Вятский государственный гуманитарный университет

ВЕСТНИК ВЯТСКОГО ГОСУДАРСТВЕННОГО ГУМАНИТАРНОГО УНИВЕРСИТЕТА

Информатика

Научно-методический журнал

№ 2

Киров
2003

ББК 74.58 я 5
В 38

Выпуск готовили: С. М. Окулов, В. А. Онегов, М. В. Петухова, Е. Н. Колодкина

Материалы выпуска подготовлены при поддержке
Министерства образования Российской Федерации, проект Г00-4.1-60

Адрес редакции: 610002, г. Киров, ул. Ленина, 111,
тел.: (8332) 678-860 (научный отдел), (8332) 673-674 (издательство)

Редакторы: Т. Котельникова, О. Коробкова
Компьютерная верстка: О. Редькина

ISBN 5-93825-098-6

Вестник
Вятского государственного гуманитарного университета

Информатика

№ 2

Научно-методический журнал

Подписано в печать 15.12.2003 г.
Формат 60×84 1/8. Бумага офсетная. Гарнитура Таймс.
Печать офсетная. Усл. печ. л. 27,90. Тираж 300. Заказ 2868

Отпечатано с готовых оригинал-макетов
в КОГУП «Кировская областная типография».
610000, г. Киров, Динамовский пр., 4

СОДЕРЖАНИЕ

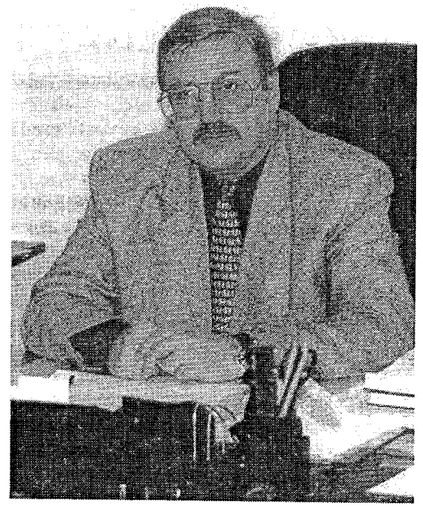
ИНФОРМАТИКА И МАТЕМАТИКА В ВУЗЕ

КОГНИТИВНАЯ ИНФОРМАТИКА

ИНФОРМАТИКА В ШКОЛЕ

ИНФОРМАТИКА И ЯЗЫК

СТУДЕНЧЕСКАЯ НАУКА



Каждый раз я с нетерпением жду выпуска «Вестника» нашего университета, посвященного научно-методическим проблемам информатики и математики. Это не связано с тем, что я сам физик и трепетно отношусь к «царице научных полей» " математике " и современной кудеснице " информатике, а скорее хочу узнать новые результаты научных вятских школ по данным отраслям знаний. Публикации эти отличаются актуальностью, новизной и оригинальностью в интерпретации тех положений, которые авторы выносят на всеобщее обсуждение. Считаю, что у журнала появилось свое лицо и стиль в сочетании с хорошим содержанием.

Надеюсь, материалы журнала займут достойное место в научном пространстве не только региона и России, но и выйдут на международный уровень, прославляя Вятскую землю.

С наилучшими пожеланиями,
ректор Вятского государственного
гуманитарного университета,
профессор, академик РАЕН

В. С. Данюшенков

ИНФОРМАТИКА И МАТЕМАТИКА В ВУЗЕ

Е. М. Вечтомов

ТЕОРЕМА ГЕДЕЛЯ О НЕПОЛНОТЕ И НАУЧНОЕ ПОЗНАНИЕ

В работе раскрывается методологическая роль классической теоремы Геделя о неполноте формальных теорий. Рассматриваются идея Лейбница об универсальном языке и программа Гильберта обоснования математики, а также выразительные возможности логико-математического языка.

Введение. В человеческом познании Мира и самопознании можно выделить три интеллектуальных уровня, которые условимся называть Разумом, Наукой и Логикой. Мы отождествляем Разум с интеллектом и знанием, Науку – с точным (проверяемым, доказательным) знанием, Логику – с рассудком и формализацией. В статье рассматриваются следующие утверждения:

I. Разум шире Науки, ибо существуют «ненаучные» формы познания, например, философская, художественная или религиозная.

II. Наука шире Логики, поскольку не каждый научный факт допускает адекватную формализацию.

Отметим, что в классической парадигме познания Науку характеризует *принцип математизации* (точного) знания [1]. Логике же соответствует конструктивная (компьютерная) математика, искусственный интеллект. Разумеется, Наука разумна, а Логика научна. Заметим также, что утверждение II представляет собой некий неформальный аналог теоремы Геделя о неполноте.

Теорема Геделя о неполноте – самое крупное достижение логики в XX веке. Она явилась выдающимся интеллектуальным событием, вошла в монографии и учебники (см., например, [3]-[6], [10], [12]), стала классикой. Велика ее роль в логике и математике, философии и теории познания, психологии и информатике. Теореме Геделя посвящены многочисленные специальные статьи и книги (укажем лишь работы [8], [9], [14]).

ВЕЧТОМОВ Евгений Михайлович – доктор физико-математических наук, профессор, заведующий кафедрой алгебры и геометрии ВятГГУ, член-корреспондент РАЕН

Мечта Лейбница – создание идеального языка, в котором строго и адекватно выражалось бы все знание, и построение такой универсальной вычислительной машины, «говорящей» на этом языке, чтобы решение любой интеллектуальной проблемы сводилось к определенным «машинным» вычислениям. Заметим, что еще Платон утверждал, что всякое знание должно быть представлено в виде точных определений, которыми может пользоваться каждый. Все, что не может быть сформулировано в виде четких правил, что опирается на интуицию, мнения или традиции, бессмысленно. Идея мышления как процесса вычисления, «подсчитывания» была сформулирована также Гоббсом.

Лейбниц ввел двоичное счисление и идею универсальной характеристики, позволяющей координатизировать понятия натуральными числами, что предвосхитило нумерацию Геделя. Ранний Лейбниц был уверен в осуществимости своей мечты (но позднее признал безусловную роль бесконечности). Это означало бы тождественность Разума и Логики, возможность замены Разума искусственным интеллектом. Если придерживаться утверждений I и II, то следует признать, что человеческий интеллект неизмеримо шире, тоньше и многограннее любого искусственного интеллекта.

Тем не менее заманчивая и великая мечта Лейбница во многом претворена в жизнь – осуществлена настолько, насколько это возможно. Созданы и развиваются математическая логика, теория алгоритмов, искусственные языки программирования, современные компьютеры, компьютерная математика. Продуктивен подход к исследованию разума с помощью структур программирования, что находит практическое воплощение при развитии интеллекта учащихся в рамках когнитивной информатики [7]. Говорят даже о «перевороте в сознании», состоящем в том, что открывается новый способ изучения мышления человека по типу компьютерного программирования. Использование с помощью компьютера принципов кибернетики может способствовать глубинному исследованию структуры знания, что позволит лучше понимать процесс познания, природу нашего сознания.

Однако не следует преувеличивать автономные возможности вычислительных машин. Они четко очерчены: компьютер по своим возможностям находится между новичком и более подготовленным начинающим и не может продвинуться дальше этой границы, поскольку мышление даже новичка включает массу процедур, непосильных для компьютера.

Поэтому необходимо «сотрудничество» компьютера с человеком для создания робота с проблесками разума. Это интересная область науки, называемая *искусственным интеллектом*. Ее цель – создание эвристических программ, дающих возможность компьютеру, перерабатывающему информацию, проявлять разумность. Проектируя робота, никто не ожидает от него, что он будет воспроизводить любое разумное действие. Можно рассчитывать только на способность человека к формализации своего поведения.

Программа Гильберта представляет собой четко продуманный план работы по безупречному обоснованию всей математики. На пути ее реализации был усовершенствован язык *математической логики*, разработанный Булем, Шредером, Фреге, Расселом, Уайтхедом, Гильбертом и его последователями, возникла *метаматематика*, зародилась математическая *теория моделей*. Полная реализация программы Гильберта означала бы совпадение математики, стало быть, и Науки, с Логикой. Однако грандиозная программа Гильберта в том финитном виде, в каком она была задумана самим Гильбертом, была обречена на неудачу. Это задолго до теоремы Геделя о неполноте отмечал Пуанкаре, указав на порочный круг: в число финитных средств неизбежно попадает метод математической индукции, который сам подлежит формализации, и, заметим, тесно связан с бесконечностью.

Рассмотрим программу Гильберта чуть подробнее. Появившиеся на рубеже XIX и XX веков элементарные логические и теоретико-множественные *парадоксы* остро поставили проблему пересмотра оснований математики, обоснования непротиворечивости математики. Среди крупнейших математиков того времени возникли споры о надежности и возможности применения в математических рассуждениях тех или иных логических средств (закона исключенного третьего, метода от противного, аксиомы выбора). В результате зародились и возникли основные методологические направления в математике – *логицизм* (Фреге, Рассел, Уайтхед), *интуиционизм* (Брауэр, Вейль, Гейтинг), *формализм* (Гильберт, Бернайс), *теоретико-множественное направление* (Кантор, Цермело, Бурбаки). В математическом плане интуиционизм есть конструктивизм, развитый создателями теории алгоритмов и ЭВМ (Пост, Черч, Тьюринг, фон Нейман, А. А. Марков).

Гильберт и формалисты видели выход в эффективной аксиоматизации не только существующей математики, но и применяемой математиками логики. К началу XX века были содержательно аксиоматизированы основные числовые системы (Гамильтон, Грассман, Дедекинд, Кантор, Пеано), евклидова геометрия (Гильберт), теория множеств (Цермело). Формализм требует, чтобы изложение велось на строго формализованном языке математической логики. Таким языком служит язык *логики первого порядка*, т.е. логика предикатов, в которой кванторы связывают только предметные переменные. Этого мнения, называемого *тезисом Гильберта*, придерживаются многие логики и математики [11]. Все богатство математики должно быть представлено в виде формальных аксиоматических теорий (систем). Во второй половине XIX века произошла *арифметизация математики* (анализа, геометрии) – сведение классической математики к натуральным числам, то есть в известной степени осуществлена пифагорейская программа «Все есть натуральное число». Поэтому одним из первых шагов программы Гильберта явилась попытка построения внутренне непротиворечивой и адекватной (полной) формальной арифметики.

Формалисты представляют математику как своеобразную игру в формулы по четко определенным правилам. Напомним, что *формальная аксиоматическая теория* (или просто теория) предполагает наличие *языка*, *аксиом* и *правил вывода*. Язык состоит из *алфавита*, содержащего знаки предметных переменных и констант, предикатов и функций, скобки, *термов* и *формул*. Термы и формулы – это грамматика теории, т.е. четко определенные слова в данном алфавите, интуитивно обозначающие соответственно предметы и высказывания о них. Аксиомы и правила вывода можно назвать дедуктикой формальной системы. Формальный язык с грамматикой и дедуктикой образуют *синтаксис*. К синтаксису относятся понятия формальной выводимости и доказательства. Смысл и истинностное значение формул принадлежат *семантике*, связанной с синтаксисом посредством *интерпретаций* и *моделей*.

Естественным образом появилась метаматематика – наука о формальных аксиоматических системах, их свойствах. Важнейшими свойствами таких теорий являются непротиворечивость, полнота, разрешимость. Метаматематические рассуждения должны быть общепринятыми, *финитными*, алгоритмическими. Школой Гильберта были доказаны непротиворечивость, полнота и разрешимость исчисления высказываний, непротиворечивость и полнота чистой логики предикатов, полнота ряда формальных алгебраических теорий. Формалисты приступили к финитному доказательству непротиворечивости и полноты формальной арифметики. И здесь их ждало большое разочарование.

Теорема Геделя о неполноте. В 1931 году австрийский математик и логик Курт Гедель доказал свою знаменитую первую *теорему о неполноте:*

В любой непротиворечивой (ω-непротиворечивой) формальной аксиоматической системе S, содержащей формальную арифметику, существует неразрешимое высказывание.

Напомним, что высказывание – это замкнутая формула логики предикатов (она не содержит свободных переменных), теорема – доказуемая формула, а неразрешимым называется высказывание A, которое нельзя

ни доказать, ни опровергнуть, то есть высказывание A и его отрицание $\neg A$ не являются теоремами в S. Противоречивые теории – в точности те, все формулы которых суть теоремы. Теория называется полной, если для любого ее высказывания доказуемо либо оно само, либо его отрицание. Сформулированная теорема Геделя утверждает, что такие достаточно богатые теории S не полны. Интересно заметить, что в S найдется недоказуемое высказывание, означающее непротиворечивость теории S. Для построения неразрешимого высказывания Гедель применил кодирование формул натуральными числами (геделева нумерация), позволившее рассуждать о высказываниях теории S (метатеория) как о натуральных числах (предметной теории S). Тем самым показано, что средствами теории S невозможно доказать ее непротиворечивость (*вторая теорема Геделя о неполноте*, см. [12]). Следовательно, финитная программа обоснования математики не выполнима. Но даже если бы программа Гильберта оказалась полностью выполнимой, то повседневная работа по гильбертовскому шаблону была бы слишком громоздкой, и в своей практике математики продолжали бы пользоваться интуицией.

Пусть S – формальная арифметика и A – некоторое ее неразрешимое высказывание. Теория S служит формализацией содержательной теории натуральных чисел, имеющей стандартную модель – обычный натуральный ряд N. Высказывание A является вполне определенным утверждением о натуральных числах, которое либо истинно, либо ложно в N. Поэтому одно из высказываний A или $\neg A$ будет истинным недоказуемым высказыванием. Получаем существование недоказуемых верных высказываний о свойствах N, причем никакое расширение теории S положение дел не меняет. Из первой теоремы Геделя следует также, что множество всех истинных в N высказываний неразрешимо, т. е. не существует алгоритма, позволяющего решить вопрос об истинности каждого высказывания формальной арифметики.

Чуть позже были доказаны *теорема Тарского о невыразимости истины* в S и *теорема Черча о неразрешимости* множества всех общезначимых формул логики предикатов. Общезначимость формулы данной теории означает ее истинность во всех моделях теории. Укажем другие фундаментальные результаты о взаимосвязи между синтаксисом и семантикой формальных систем. Во-первых, назовем *теорему Геделя* 1930 года *о полноте*, или адекватности: формула теории первого порядка является теоремой тогда и только тогда, когда она общезначима. Во-вторых, обобщенная теорема о полноте утверждает, что непротиворечивость формальной аксиоматической теории равносильна существованию модели этой теории. Отсюда выводится *теорема компактности*, впервые доказанная А. И. Мальцевым в 1936 году: множество высказываний некоторой формальной аксиоматической теории имеет модель тогда и только тогда, когда любое его конечное подмножество обладает моделью. Заметим, что в пользу приведенного выше тезиса Гильберта говорит следующий прекрасный результат Линдстрема [11]: логика первого порядка является единственной логикой, замкнутой относительно обычных логических связок и кванторов и удовлетворяющей теоремам компактности и Левенгейма-Скулема. А *теорема Левенгейма-Скулема* утверждает, что теория с бесконечной моделью имеет и счетную модель.

Поясним первую теорему Геделя о неполноте. В системе S строится высказывание A, говорящее о собственной недоказуемости в терминах геделевой нумерации, т. е. A обозначает высказывание «Это предложение недоказуемо». Здесь, как и в строгом оригинальном рассуждении Геделя, обыгрывается знаменитый *парадокс лжеца* в форме «Это предложение ложно» с заменой слова «ложно» на «недоказуемо». При любых интерпретациях непротиворечивой теории теоремы истинны. Поэтому высказывание A не может быть ложным. Значит, A – недоказуемая истина и, стало быть, неразрешимое высказывание.

Модификации формализма. Обоснование математики в гильбертовском смысле потерпело неудачу. При построении математики как формальной системы Гильберт допускал только логику первого порядка в качестве предметной теории и только финитные методы в метатеории. Но можно пытаться доказать непротиворечивость математики, либо несколько усилив язык формальной теории, либо используя некоторые достаточно убедительные нефинитные средства в метаматематических рассуждениях. Так, Генцен доказал непротиворечивость формальной арифметики, применив трансфинитную индукцию по счетным ординалам (см. [5]).

Интересна предпринятая Ю. Л. Ершовым реабилитация программы Гильберта [2]. Его подход придает новый смысл программе Гильберта и ее реализации: *«для всякой заинтересовавшей нас проблемы ищите систему с обычными правилами вывода, в которой эта проблема представима в виде осмысленной задачи, и затем ищите для такой системы финитное доказательство ее непротиворечивости в соответствии с прежним императивом программы Гильберта»*. Это определенная локализация гильбертовского глобализма.

А. Н. Паршин рассматривает геделеву нумерацию как определенную систему координат, придавая ей геометрический смысл и имея в виду ее вложение в p-адический континуум [8]. Это открывает новые возможности истолкования теоремы Геделя.

Гносеологические выводы. Теорема Геделя о неполноте ограничивает тотальную формализацию Науки, высвобождая энергию творчества. Она убедительно показывает несводимость мышления к логике, принципиальную неформализуемость Разума, необходимость интуиции.

Как подчеркивает А. Н. Паршин в своей глубокой работе [8], теорема Геделя есть фундаментальный философский факт, говорящий о каком-то глубинном свойстве мышления и указывающий на существование *подвижной* границы между формальным и интуитивным везде, в частности в самой математике, причем, эта граница «устанавливается каждый раз *заново в каждом* новом акте познания». Интересно отметить, что зачастую акт обретения нового совершается мгновенно (именно так человек начинает уметь плавать и кататься на велосипеде). Кроме того, надо сказать, что в квантовой физике имеется методологически аналогичный результат – это теорема фон Неймана о невозможности введения скрытых параметров, которые позволили бы свести квантовые системы к системам классической механики.

Открытие Геделя можно сравнить с расшифровкой генома человека, заключающего в себе уникальность и всеобщность каждой человеческой жизни. Теорема Геделя о неполноте, сама будучи вершиной логической мысли, несет огромный жизнеутверждающий заряд – ценность постепенного логического освоения научного знания. Подчеркивая непреходящее значение творчества, она укрепляет веру в бесконечный прогресс рационального познания.

Примечания

1. *Вечтомов Е. М.* Научное познание и математика // Вестник ВятГГУ. 2002. № 7. С. 8-11.
2. *Ершов Ю. Л., Самохвалов К. Ф.* О новом подходе к методологии математики // Закономерности развития современной математики. Методологические аспекты. М.: Наука, 1987. С. 85-106.
3. *Линдон Р.* Заметки по логике. М.: Мир, 1968.
4. *Мадер В. В.* Введение в методологию математики. (Гносеологические, методологические и мировоззренческие аспекты математики. Математика и теория познания). М.: Интерпракс, 1995.
5. *Мендельсон Э.* Введение в математическую логику. М.: Наука, 1971.
6. *Непейвода Н. Н.* Практическая логика. Новосибирск: Изд-во Новосиб. ун-та, 2000.
7. *Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ, 2003.
8. *Паршин А. Н.* Размышления над теоремой Геделя // Вопросы философии. 2000. № 6. С. 92-109.
9. *Подниекс К. М.* Вокруг теоремы Геделя. Рига, 1981.
10. *Смальян Р.* Теория формальных систем. М.: Наука, 1981.
11. Справочная книга по математической логике. Ч. I. Теория моделей. М.: Наука, 1982.
12. Справочная книга по математической логике. Ч. IV. Теория доказательств и конструктивная математика. М.: Наука, 1983.
13. *Тьюринг А.* Может ли машина мыслить? М.: ГИФМЛ, 1960.
14. *Успенский В. А.* Теорема Геделя о неполноте. М.: Наука, 1982. (Популярные лекции по математике).

В. А. Онегов

КОМПЬЮТЕРНАЯ МАТЕМАТИКА

В статье определяется отношение к термину «компьютерная математика» и приводятся доводы в его поддержку. Компьютерная математика рассматривается как совокупность теоретических предметов, составляющих основу информатики и одновременно опирающихся на фундаментальный аппарат математики. Указывается на динамическую связь между математикой и информатикой.

Понятие «компьютерная математика» было введено Д. Куком и Г. Бейзом [1]. Конечно, как и всякий новый термин, вводимый в структуру науки, требуется уточнение этого понятия путем очерчивания сферы, которую он охватывает. Авторы наполняют его различными разделами математики: множества, отношения, функции, основные понятия арифметики, алгебраические структуры, матрицы, автоматы, компьютерная геометрия.

В 2001 году в журнале «Информатика и образование» № 2 была опубликована статья [2], некоторые из предложений которой могут послужить темой для обсуждения. Авторы статьи также используют термин «компьютерная математика». Они даже предлагают ввести вузовскую дисциплину с таким названием, и она, по их мнению, должна состоять из разделов:

1) основания конструктивной математики, состоящие из элементов семиотики, а также математической логики и теории алгоритмов;

2) конструктивная дискретная математика;

3) численные методы (включая компьютерную алгебру);

4) компьютерная геометрия.

Первое, что вызывает несогласие, – это попытка обособить «компьютерную математику» от общего и единого материнского дерева математики. Второе: вольное обращение с термином «конструктивная математика», который в системе математических наук имеет устоявшийся и определенный смысл, отличный от смысла, придаваемого ему авторами. Третье: в состав предлагаемого курса так или иначе в качестве разделов предлагается ввести большинство предметов существующего учебного цикла. Мы считаем, что принятый сейчас учебный план более полон и содержателен, кроме того, структурирован по семестрам и по количеству отводящихся на предметы часов.

Осознание глубинной связи информатики и математики привело к тому, что в систему учебных дисциплин, входящих в учебный план подготовки учителя информатики, в 2000 году были включены предметы теоретической информатики и математические предметы, тесно с ними связанные. Принятый в

ОНЕГОВ Владислав Алексеевич – кандидат физико-математических наук, доцент, заведующий кафедрой прикладной математики ВятГГУ

2000 году Государственный образовательный стандарт предусматривает следующие предметы профессиональной подготовки учителя информатики (в порядке изучения по семестрам):

– математическая логика (4-й семестр);
– дискретная математика (4-й семестр);
– элементы абстрактной и компьютерной алгебры (5-й семестр);
– уравнения математической физики (5-й семестр);
– теория алгоритмов (6-й семестр);
– численные методы (6-й, 7-й семестры);
– теоретические основы информатики (6-й семестр);
– исследование операций (8-й семестр);
– компьютерное моделирование;
– основы искусственного интеллекта (10-й семестр).

Перечень теоретических дисциплин говорит о необходимости основательной предварительной и текущей подготовки и корреляции этих дисциплин между собой и теоретической и прикладной математикой. Ранее предпринимаемые попытки готовить специалистов по информатике без серьезных математических и теоретических (по информатике) знаний не принес успеха. Названный подход возможен лишь в отношении информационных технологий (с известными оговорками), к которым, конечно, не сводится информатика. Здесь уместно упомянуть, что в 2002 году при утверждении организационных структур Академии Наук России было выделено отделение математики в составе двух секций: 1) теоретическая математика; 2) прикладная математика и информатика. Это еще раз подтверждает единство математики и теоретической информатики – два ствола одного дерева.

С большим внутренним сопротивлением автор настоящей статьи согласился на условное использование термина «компьютерная математика» только для обозначения совокупности перечисленных предметов учебного плана, с включением в нее основ теоретической математики.

С одной стороны, этимологически ясна связь этой группы наук с принятыми в информатике разделами «компьютерная геометрия» и «компьютерная алгебра». С другой – дословный перевод термина «computer science» означает «вычислительная наука», что можно, не переводя слово «computer», трактовать как «компьютерная наука». Тогда «компьютерная математика» – это математические разделы вычислительной науки («computer science»), а вместе с этим – информатики (информатика – синоним англоязычного «computer science»). Не вдаваясь в дальнейшее обсуждение удачности или неудачности термина «компьютерная математика», воспользуемся им и будем этим условным термином называть разделы информатики, которые тесно связаны с математическими науками (теоретическая информатика) и собственно математическими науками, идеи которых постоянно питают и теоретически обосновывают собственно информатику.

Рассматривая любой из перечисленных предметов, в каждом конкретном случае можно указать сферу использования соответствующих знаний либо для решения конкретных задач, либо в качестве обоснования некоторых теоретических положений информатики. Встает вопрос о том, что является самым важным, объединяющим все предметы моментом (кроме всеобщности отношения к информатике). На наш взгляд, составители Государственного стандарта по информатике впервые последовательно проводят идею универсальности основного методологического принципа информатики: математическое, информационное и компьютерное моделирование изучаемых систем и явлений. Таким образом, выделен инвариант информатики как науки.

Информатика является одной из основополагающих наук естественнонаучного цикла, наряду с философией, математикой и физикой. В этом качестве ее способы исследования различных явлений (систем и т.п.), методы решения задач, возникающих в самых разных сферах интеллектуальной деятельности современного человека, обладают единством подхода – методологией.

Воспроизведем в виде тезисов (в нашем изложении) некоторые мысли академика А. П. Ершова, высказанные им на 6-м Международном конгрессе по математическому образованию в Будапеште в 1988 году, которые полностью могут быть отнесены к предмету обсуждения этой статьи.

1. Резкое расширение математической практики. Повсеместное применение компьютеров, строительство информационной модели мира раздвинули объем и многообразие математической практики в грандиозных масштабах. Построение знаковых систем, схематизация конкретных объектов путем выделения их свойств, атрибутов и отношений, построение моделей, дедукция, редукция и рекурсивное мышление, выделение и поддержание уровней абстракции, прогнозирование поведения, анализ законов, установлений и правил, наконец, конструирование огромного количества алгоритмов и их оценка – все это оружие современного интеллекта, каркас информационной культуры.

2. Изменение номенклатуры математических знаний. Компьютер воспроизводит человеческое поведение. Через программирование и построение информационных моделей в содержательную часть математики входят абстракции человеческой деятельности, свойства искусственных и живых систем. Это же усиливает роль и место дискретной математики. Выявляется роль разделов дискретного анализа, параллельных классическому анализу. На первый план выходит изучение связи между дискретным и непрерывным, например, в синергетике и теории катастроф. Появляются новые приемы математической деятельности – доказательные вычисления. Еще один пример неортодоксальных математических доказательств – решение задачи о четырех красках.

3. Системная роль математической теории. Понятие теории родилось в недрах математики. С другой стороны, в информатике имеется важнейшее понятие обстановки. Обстановка – это воплощенная в компьютере замкнутая модель мира, в котором предстоит действовать программируемому исполнителю. Поскольку все исходы должны быть предусмотрены, необходимо иметь полное знание обстановки и сознавать пределы этого знания в реальном мире. Все это знание должно предшествовать конкретному моделированию. Выработка этого знания составляет сущность системного анализа. Системный анализ – новый массовый вид человеческой практики.

4. Вычислительный эксперимент с математической и компьютерной моделью. Его роль в инженерной практике общеизвестна. Следует подчеркнуть, что вычислительный эксперимент все в большей степени становится источником чисто теоретических открытий. Можно отметить практичность этого подхода как нового метода в познавательной деятельности в учебной педагогической практике.

От себя отметим, что построение когнитивных методов дидактики следует начинать с построения той или иной модели обучения.

5. Визуализация абстракций. Визуальное восприятие человека является поистине магическим кристаллом, позволяющим делать открытия. Поиски того, как сделать мысль наглядной, всегда мучили ученых и педагогов. Интеллектуальная графика имеет многовековую историю. За последнее время появилась масса публикаций – образы, порожденные абстрактным знанием, оживленным союзом ученого программиста и компьютера.

Отметим, что академик А. Н. Колмогоров подчеркивал различие между формальной символической системой и содержательным (математическим) знанием. Это отлично чувствует любой математик или информатик, решая творческую задачу. Компьютер действительно по строго формализованным правилам выполняет действия над конструктивной информацией (словами некоторого алфавита или наборами символов). Законы мышления человека не таковы.

6. Динамизация математических объектов. Математика – наука об инвариантах. Познать природу инварианта можно, только осознав диалектику постоянства и изменчивости параметров этого инварианта. Увидеть в логической константе все проявления реальной жизни, описанной законом, – это значит понять закон и научиться его применять. Компьютер со своими средствами визуализации и вычислений позволяет исследователю извлечь из статической упаковки математических отношений всевозможные траектории развития динамического процесса во времени и пространстве, обогащая тем самым его опыт, интуицию и способность к предсказыванию.

7. Становление структуры из хаоса. Среди возможностей, предоставляемых вычислительным экспериментом и средствами визуализации, заслуживают особого упоминания эксперименты по наблюдению становления регулярных структур из исходного беспорядка. В их простейшем проявлении – это разнообразные конструкции, возникающие при итеративном применении некоторых нелинейных операторов к случайным исходным данным или попутным параметрам. Здесь формируется совершенно новый и исключительно мощный канал для распространения знаний на огромный класс природных явлений: движение материков, формирование береговой линии, горные ландшафты, рисунки полярных сияний, формообразование у растений, расцветка животных, развитие конфликтов и возникновение кризисов. Материал, поставляемый синергетикой и математикой нелинейного, позволяет сделать важный общенаучный вывод о принципиальной важности вычислительного эксперимента как познавательного инструмента: если источником всего нового в природе является нелинейность, то умозрительные представления экстраполяционного типа линейны по своей природе и поэтому ограничены в своей познавательной силе, как, например, любой вывод в существующей аксиоматике. Поэтому для добычи поистине нового знания необходим нелинейный синергетический процесс либо в мозгу человека, либо в памяти компьютера.

Отметим, что процесс получения новых знаний при обучении также нелинеен. Осознание этого обстоятельства наконец-то начинает проникать в современные теории когнитивной психологии и дидактики.

8. Союз информатики, математики и лингвистики. Можно дать одно ограниченное определение информатики: информатика – это наука о правилах целеустремленной деятельности. Это определение становится просто справедливым, если мы признаем компьютер в соответствии с тезисом Черча и тьюринговым понятием универсальной машины (с оракулом) всеобъемлющей моделью целеустремленной деятельности. Если мы согласимся с этим, то информатика по праву входит в союз с математикой и лингвистикой, закладывая уже в школьное образование опорный треугольник развития главных проявлений человеческого интеллекта: способность к обучению, способность к рассуждению и способность к действию. Дисциплина действия так же нужна, как и дисциплина речи. Понимая, как компьютер решает задачу, он сохраняет это понимание в себе. Наблюдая катастрофы в искусственных мирах, он многократнее и безопаснее для себя вырабатывает опыт сопоставления решений и их последствий.

Направления развития информатики обозначены А. П. Ершовым очень четко и впечатляюще. Каким образом воплотить это в сфере образования, в конкретном вузе?

На наш взгляд, «компьютерной математике», как она понимается нами, не хватает детальной проработки логических и функциональных связей между составляющими ее предметами и разделами. Следуя принятой парадигме обучения предметам цикла «ком-

пьютерная математика», с неизбежностью приходится (или придется) решать отмеченные проблемы. В связи с этим кафедра прикладной математики и ИМОИ факультета информатики выступила с инициативой по созданию экспериментальной методической площадки «компьютерная математика». Предполагается, что в процессе работы площадки будут выделены ключевые понятия и основные положения каждого из входящих в комплекс предметов, установление логических связей между ними и последовательность освоения соответствующих понятий.

Очень важным представляется выделение понятий, теорем и методов их доказательств в курсе основ теоретической математики. В настоящее время заметно некоторое несогласование разделов и тем основ теоретической математики потребностям предметов теоретической математики. В качестве одного из первых шагов автор предлагает введение на 1-м курсе пропедевтического предмета – введение в математику, в котором бы рассматривались важные понятия математики, нужные в самом начале обучения в университете и не изучаемые в средней школе. К ним стоит отнести элементы теории множеств, отношения, отображения, элементы математической логики, элементарные логические средства рассуждений и доказательств в математике, необходимые и достаточные условия, метод полной математической индукции, метод доказательства от противного, дедукция, конструктивные доказательства.

Очень важным представляется создание некоего мостика, способного помочь студентам преодолеть известные эмоциональные трудности в освоении абстрактных разделов теоретической математики. В этом плане интересна публикация перевода с английского языка монографии Р. Грэхема, Д. Кнута и О. Паташника «Конкретная математика». В предисловии к монографии Владимир Игоревич Арнольд написал, что созданная Ньютоном и Эйлером, Бернулли и Гауссом, Лейбницем и Дирихле математика оказывается вечно юной и вновь возрождается следующими поколениями математиков. Польза этого труда в том, что в ней рассматриваются конкретные примеры как дискретных, так и непрерывных систем. Противопоставление «конкретной математики» и «абстрактной математики», погоня за обобщениями оказалась столь захватывающей, что целое поколение математиков потеряло способность находить прелести в частностях, в том числе получать удовольствие от решения численных задач или оценить по достоинству роль математических методов. Абстрактная математика стала вырождаться и терять связь с действительностью. Конкретная математика – смесь континуальной и дискретной математики. Исчисление сумм, рекурентные соотношения, элементарная теория чисел, дискретная теория вероятностей представляют большой интерес с технической стороны.

Вопросы, задачи и упражнения, рассматриваемые в «Конкретной математике», не относятся, строго говоря, к «компьютерной математике». Это разделы классической математики того периода, когда не было четкого разделения теоретической и прикладной математики. Несмотря на то, что и сами задачи, и приемы их решения достаточно серьезны, доведение решения до числа или формулы снимает у читателя чувство недоступности рассматриваемого материала. Можно организовать проведение специального курса на базе материала этой монографии.

В заключение отметим, что ведущие преподаватели факультета информатики достаточно отчетливо представляют себе направление и содержание работы, которую необходимо осуществить в рамках экспериментальной площадки университета «Компьютерная математика». Имеется определенный задел в виде разработанных учебных программ по всем дисциплинам цикла, а также большое количество публикаций, связанных с намеченными работами.

Примечания

1. *Кук Д., Бейз Г.* Компьютерная математика. М.: Наука, 1990.

2. *Бороненко Т. А., Рыжова Н. И.* Компьютерная математика в педагогическом вузе и школе // Информатика и образование. 2001. № 2.

3. *Ершов А. П.* Компьютеризация школы и математическое образование // Информатика и образование. 1992. № 5-6.

4. *Грехэм Р., Кнут Д., Паташник О.* Конкретная математика. М.: Мир, 1998.

5. *Онегов В. А.* Теоретическая информатика и математика // Вестник ВятГГУ. Информатика. Киров, 2002. № 1.

С. И. Калинин

О ПОИСКЕ НЕЛИНЕЙНЫХ ЭМПИРИЧЕСКИХ ФУНКЦИЙ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

В статье описывается метод наименьших квадратов, на основе которого строятся нелинейные эмпирические функции.

В прикладных исследованиях часто зависимость между переменными величинами удается описывать лишь табличными функциями. Для изучения же глубинных закономерностей связи зависимой и независимой переменных приходится решать задачу аппроксимации табличной функции некоторой аналитически задаваемой функцией, значения которой, по возможности, мало бы отличались от данных табличных значений.

КАЛИНИН Сергей Иванович – кандидат физико-математических наук, доцент кафедры прикладной математики

Напомним, всякую аналитическую функцию, приближающую полученную опытным путем табличную функцию, называют *эмпирической функцией*, или *эмпирической формулой*.

Построение эмпирических функций состоит из двух этапов: сначала выявляют общий вид эмпирической функции, а затем определяют значения входящих в ее запись параметров так, чтобы искомая функция наилучшим образом приближала бы табличную функцию. Часто при определении значений упоминаемых параметров используют так называемый *метод наименьших квадратов*, или, по-другому, *принцип Лежандра*. Его суть состоит в том, что из функций $y=f(x)$, приближающих данную табличную, выбирается та, для которой сумма квадратов отклонений табличных значений от вычисляемых является наименьшей.

Метод наименьших квадратов достаточно просто реализуется в отношении поиска линейной эмпирической функции (линейная функция зависит всего от двух параметров – от углового коэффициента и начальной ординаты). Но в случае, когда данная табличная функция имеет широкий спектр значений, то следует ожидать, что ее лучше будет приближать нелинейная эмпирическая функция.

Для данной табличной функции

x	x_1	x_2	…	x_n
y	y_1	y_2	…	y_n

рассмотрим задачу нахождения эмпирической функции вида

$$y = a_1 x^{\alpha_1} + a_2 x^{\alpha_2} + \ldots + a_q x^{\alpha_q},\ q \geq 2, \qquad (1)$$

где у табличной функции аргумент x принимает только положительные значения, а у функции (1) величины $a_1,\ldots,a_q$ – параметры, подлежащие определению, показатели же $\alpha_1,\ldots,\alpha_q$ – произвольные фиксированные вещественные числа.

Для определения значений параметров $a_1,\ldots,a_q$ вычислим отклонения $\delta_i = \left(\sum_{j=1}^{q} a_j x_i^{\alpha_j}\right) - y_i$ по ординатам каждой из n точек графика табличной функции от соответствующих точек графика аппроксимирующей функции (1). В соответствии с методом наименьших квадратов наилучшей функцией вида (1) считаем ту, для которой сумма $S = \sum \delta_i^2$ будет наименьшей.

Перепишем S в виде:

$$S = \sum_{i=1}^{n}\left(\left(\sum_{j=1}^{q} a_j x_i^{\alpha_j}\right) - y_i\right)^2. \qquad (2)$$

Теперь S можно рассматривать как функцию q переменных $a_1,\ldots,a_q\,(a_k \in R,\ k=1,\ldots,q)$. Исследуем ее на экстремум в пространстве R^q. В силу необходимого условия экстремума должны выполняться соотношения

$$\frac{\partial S}{\partial a_k} = 0,\ k=1,\ldots,q,\ \text{или}$$

$$\sum_{i=1}^{n} 2\left(\left(\sum_{j=1}^{q} a_j x_i^{\alpha_j}\right) - y_i\right) x_i^{\alpha_k} = 0,\ k=1,\ldots,q.$$

Последние порождают следующую систему линейных уравнений относительно параметров $a_1,\ldots,a_q$:

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^{\alpha_1+\alpha_1}\right)a_1 + \left(\sum_{i=1}^{n} x_i^{\alpha_2+\alpha_1}\right)a_2 + \ldots + \left(\sum_{i=1}^{n} x_i^{\alpha_q+\alpha_1}\right)a_q = \sum_{i=1}^{n} y_i x_i^{\alpha_1} \\ \qquad\qquad\qquad\qquad\qquad\qquad k=1,\ldots,q. \end{cases} \qquad (3)$$

Следуя терминологии, используемой при построении эмпирической линейной функции методом наименьших квадратов, систему (3) назовем *нормальной системой* уравнений для определения параметров $a_1,\ldots,a_q$. Это есть неоднородная система q линейных уравнений с q неизвестными.

Пусть

$$\Delta_q = \left|\left(\sum_{i=1}^{n} x_i^{\alpha_j+\alpha_k}\right)\right| \qquad (4)$$

определитель матрицы системы (3). Если $\Delta_q \neq 0$, то система (3) будет совместной, определенной, и ее решение легко может быть найдено по правилу Крамера.

Пусть $\left(a_1^{(0)},\ldots,a_q^{(0)}\right)$ – единственное решение системы (3). Покажем, что тогда точка $M\left(a_1^{(0)},\ldots,a_q^{(0)}\right)$ в пространстве R^n будет точкой глобального минимума функции (2).

Вычислим частные производные второго порядка функции (2): $\frac{\partial^2 S}{\partial a_j \partial a_k} = 2\sum x_i^{\alpha_j+\alpha_k}$. Для установления требуемого, в силу достаточных условий экстремума функции нескольких переменных в стационарной точке (см. [1], с. 422–424), нам нужно показать, что второй дифференциал функции S в точке M_0 будет положительным, то есть будет выполняться соотношение

$$\sum_{j,k=1}^{q}\left(\sum_{i=1}^{n} x_i^{\alpha_j+\alpha_k}\right) da_j da_k > 0 \qquad (5)$$

при одновременном необращении в нуль дифференциалов $da_1,\ldots,da_q$.

Левая часть (5) есть квадратичная форма от переменных $da_1,\ldots,da_q$. Легко видеть, что она может быть представлена в виде $\sum_{i=1}^{n}\left(\sum_{j,k=1}^{q}\left(x_i^{\alpha_j} da_j + x_i^{\alpha_k} da_k\right)^2\right)$, поэтому, очевидно, будет положительной, если $da_1^2 + \ldots + da_q^2 > 0$.

Таким образом, действительно, точка M_0 является точкой глобального минимума функции (2). Отсюда имеем, что искомая эмпирическая функция есть функция

$$y = a_1^{(0)} x^{\alpha_1} + a_2^{(0)} x^{\alpha_2} + \ldots + a_q^{(0)} x^{\alpha_q}. \quad (6)$$

Замечание 1. Если в (1) показатели $\alpha_1,\ldots,\alpha_q$ таковы, что степени $x^{\alpha_1},\ldots,x^{\alpha_q}$ имеют смысл не только для положительных значений x, то в качестве аргумента x табличной и эмпирической функций могут выступать, очевидно, и неположительные значения. Так, если α_k, $k=1,\ldots,q$, – целые показатели, то $x_1,\ldots,x_n$ могут быть любыми числами, отличными от нуля. Тогда эмпирическая функция может строиться на множестве $R\setminus\{0\}$. Если же показатели α_k, $k=1,\ldots,q$, – целые неотрицательные числа, то, очевидно, эмпирическая функция может строиться на множестве R.

Пусть в (1) $\alpha_1=1$, $\alpha_2=0$, $a_3=\ldots a_q=0$. Тогда функция (1) становится линейной. Для такой функции нормальная система (3) примет вид

$$\begin{cases} \left(\sum\limits_{i=1}^{n} x_i^2\right)a_1 + \left(\sum\limits_{i=1}^{n} x_i\right)a_2 = \sum\limits_{i=1}^{n} x_i y_i \\ \left(\sum\limits_{i=1}^{n} x_i\right)a_1 + na_2 = \sum\limits_{i=1}^{n} y_i, \end{cases} \quad (7)$$

а определителем (4) будет являться определитель

$$\Delta_2 = \begin{vmatrix} \sum\limits_{i=1}^{n} x_i^2 & \sum\limits_{i=1}^{n} x_i \\ \sum\limits_{i=1}^{n} x_i & n \end{vmatrix} = n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2.$$

В силу неравенства $\sqrt{\frac{\sum\limits_{i=1}^{n} x_i^2}{n}} > \frac{\sum\limits_{i=1}^{n} x_i}{n}$ между средним квадратичным и средним арифметическим n положительных чисел $x_1,\ldots,x_n$, заключаем, что будет иметь место условие $\Delta_2 \neq 0$. Значит, система (7) будет всегда совместной, а ее единственное решение $\begin{cases} a_1 = a_1^{(0)} \\ a_2 = a_2^{(0)} \end{cases}$ будет порождать линейную эмпирическую функцию $y = a_1^{(0)} x + a_2^{(0)}$, наилучшим образом аппроксимирующую табличную функцию по методу наименьших квадратов.

Пусть теперь в (1) $a_3=\ldots a_q=0$. Тогда функция (1) в своей записи будет содержать только две степени, то есть будет иметь вид $y = a_1 x^{\alpha_1} + a_2 x^{\alpha_2}$ ($x>0$). Для нее нормальная система запишется так:

$$\begin{cases} \left(\sum\limits_{i=1}^{n} x_i^{2\alpha_1}\right)a_1 + \left(\sum\limits_{i=1}^{n} x_i^{\alpha_1}x_i^{\alpha_2}\right)a_2 = \sum\limits_{i=1}^{n} x_i^{\alpha_1} y_i \\ \left(\sum\limits_{i=1}^{n} x_i^{\alpha_1}x_i^{\alpha_2}\right)a_1 + \left(\sum\limits_{i=1}^{n} x_i^{2\alpha_2}\right)a_2 = \sum\limits_{i=1}^{n} x_i^{\alpha_2} y_i. \end{cases} \quad (9)$$

Определитель матрицы этой системы есть величина

$$\tilde{\Delta}_2 = \begin{vmatrix} \sum\limits_{i=1}^{n} x_i^{2\alpha_1} & \sum\limits_{i=1}^{n} x_i^{\alpha_1+\alpha_2} \\ \sum\limits_{i=1}^{n} x_i^{\alpha_1+\alpha_2} & \sum\limits_{i=1}^{n} x_i^{2\alpha_2} \end{vmatrix} = \left(\sum_{i=1}^{n} x_i^{2\alpha_1}\right)\left(\sum_{i=1}^{n} x_i^{2\alpha_2}\right) - \left(\sum_{i=1}^{n} x_i^{\alpha_1}x_i^{\alpha_2}\right)^2 > 0.$$

Последняя оценка вытекает из хорошо известного неравенства Коши-Буняковского

$$\left(\sum_{i=1}^{n} a_i b_i\right)^2 \le \left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right), \quad (a_i, b_i \in R, b_i \neq 0, i=1,\ldots,n),$$

в котором равенство достигается лишь тогда, когда $\frac{a_1}{b_1} = \ldots = \frac{a_n}{b_n}$. Так что система (9) всегда будет иметь единственное решение $\begin{cases} a_1 = a_1^{(0)} \\ a_2 = a_2^{(0)} \end{cases}$, а соответствующая эмпирическая функция будет иметь вид $y = a_1^{(0)} x^{\alpha_1} + a_2^{(0)} x^{\alpha_2}$.

Ясно, что последняя методом наименьших квадратов может определяться и в том случае, когда аргумент x будет принимать не только положительные значения – лишь бы имели смысл степени x^{α_1}, x^{α_2}.

Примечания

1. *Фихтенгольц Г. М.* Курс дифференциального и интегрального исчисления. Т. 1. М.: Наука, 1966. 608 с.

М. В. Петухова

ПУТИ ОПТИМАЛЬНОЙ ИНТЕГРАЦИИ ТЕОРИИ И ПРАКТИКИ ПРИ ФОРМИРОВАНИИ СИСТЕМЫ ПОНЯТИЙ КУРСА «СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ»

В статье рассматриваются вопросы формирования у студентов системы понятий дисциплины «Системы управления базами данных», предлагаются этапы организации этого процесса, а также приводятся и анализируются результаты первого этапа.

Изучение какого-либо раздела любой науки подразумевает формирование целостной системы его основных понятий, входящей в общую понятийную систему, умения практического оперирования ими, приложение их к решению задач. Структура понятийного аппарата науки адекватна структуре самой науки, является обобщенным и абстрагированным отражением ее объектов, явлений и процессов. П. А. Флоренский, например, пишет: «Не ищите в науке ничего, кроме терминов, данных в их соотношениях: все содержание науки, как таковой, сводится именно к терминам в их связях, которые (связи) первично даются определениями терминов» [4]. Утверждение это, конечно, спорно, но во многом с ним можно и согласиться.

ПЕТУХОВА Мария Владиславовна – кандидат педагогических наук, доцент, заведующая кафедрой информационных технологий и ТСО

Формирование понятий – одна из важных когнитивных функций человека. Обучение является видом познания, и формирование в процессе его научных понятий у студентов – одна из форм познавательной деятельности. В ходе обучения понятие должно задаваться, прежде всего, не в своей словесной форме, а в форме активной деятельности самого студента. Эта деятельность направляется на обнаружение внутренних связей и отношений. Усвоение понятий обучаемым плодотворно лишь тогда, когда он является не только объектом обучения, но и субъектом познания и действий. То есть важно включить студента в сам процесс производства понятия. С. Л. Рубинштейн рассматривал понятийное обобщение как основу переноса знаний и умений и в связи с этим выделял две характерные черты теоретического обобщения: вскрытие внутренней связи объектов класса на примере анализа одного факта; обобщение на основе этой связи всех других фактов данного круга явлений [2].

П. Я. Гальперин [3], отражая идею поэтапного формирования понятий, выделяет следующие этапы обучения:

1) обучаемому в готовом виде дается неполная система ориентиров и объяснений действий, проводится однократная демонстрация образцов;

2) дается в готовом виде полная ориентировочная основа действий;

3) ориентировочная основа представлена в обобщенном виде, что позволяет обучаемому самостоятельно составлять конкретные ориентиры в действии.

При разработке курса «Системы управления базами данных (СУБД)» и методики его изучения учитывались именно эти идеи.

Курс «СУБД», помимо актуальности для многих областей деятельности, интересен и тем, что сочетает в себе несколько направлений информатики: здесь и элементы теоретической информатики, и элементы программирования, и применение современных прикладных программных средств общего назначения. Понятно, что при организации его изучения встает вопрос об оптимальной интеграции теории и практики, то есть как рационально построить работу с конкретной системой управления базами данных на практических занятиях, чтобы теоретические понятия, с одной стороны, формировались в процессе деятельности студентов, а с другой стороны, стали базой для решения практических задач. Проблемы при этом связаны с тем, что многие вопросы (например, нормализации, обеспечения целостности) достаточно объемны, поэтому начинать на практике «с самого начала» – с проектирования базы данных – затруднительно.

С опорой на идею поэтапного формирования понятий были выделены следующие этапы практической стороны изучения курса «СУБД».

1. Работа с готовой базой данных (БД). На ее примере происходит знакомство с интерфейсом и объектами конкретной СУБД: таблицами, связями, формами, отчетами, запросами, страницами доступа к данным, макросами – в режимах просмотра и конструктора. Выясняется их назначение и структура. Для этого предлагается план самостоятельного изучения, в конце занятия производится обобщение.

2. Проектирование БД. Здесь главная задача – спроектировать свою БД, опираясь на «здравый смысл». Студенты под руководством преподавателя обсуждают и выбирают область реального мира, по которой будет строиться БД, выделяют объекты, свойства и т. д. В процессе обсуждения даются общие принципы и понятия БД, формулируются шаги ее проектирования в неформализованном виде, приблизительно. Область реальной жизни, по которой строится БД, должна быть хорошо знакома студентам (например, школа или университет). Понятно, что БД может получиться не полностью нормализованная. Но это и неплохо – в процессе изучения теории можно привлекать примеры из этой БД, выявляя ошибки или неточности, их причины и возможные последствия. Лучше, если эти «ошибки» будут продуманы заранее.

3. Создание и заполнение БД. При этом опытным путем устанавливается ряд важных фактов, например, что связывать можно только однотипные поля; что установленное автоматически значение по умолчанию для числовых полей, которые задействованы в связях, иногда «мешает» при заполнении БД; что запрещено использовать повторяющиеся значения ключевых полей и т.п. Это обеспечивает начальные представления о целостности данных.

4. Работа с БД, создание внешних представлений данных: запросов, отчетов и т. п.

5. Нормализация БД. Производится после рассмотрения соответствующего теоретического материала, то есть когда произошла «формализация соображений здравого смысла», произведено обобщение и облачение в точную терминологическую оболочку тех конкретных примеров и приблизительных представлений, которыми пользовались студенты при проектировании БД.

6. Проектирование и создание своей БД. Каждый студент для этого может выбрать заранее ту область реального мира, которая ему интереснее. Преподаватель четко формулирует требования к результирующей БД. Здесь студенты уже опираются на теоретические знания и практический опыт, полученный при построении и работе с общей базой данных.

Для первого этапа возникает законный вопрос: насколько точно студенты без предварительной подготовки и объяснения в процессе самостоятельного

исследования смогут выявить назначение и структуру основных объектов СУБД.

В качестве небольшого эксперимента студентам III курса факультета информатики (дисциплина «Системы управления базами данных») и V курса математического факультета (дисциплина «Информационные системы») было предложено без каких-либо предварительных объяснений на первом практическом занятии в течение 30 минут самостоятельно исследовать готовую базу данных по предоставленному примерному плану. Этот план предусматривал просмотр основных объектов СУБД (таблиц, форм, отчетов, запросов) в разных режимах (просмотра и конструктора), свойств этих объектов и схемы данных. Затем студенты заполнили анкеты, где своими словами сформулировали назначение исследованных объектов.

Результаты ответов были обобщены по категориям: «близкий к точному», «частично верный», «неверный», «не указан» (см. таблицу).

Категория ответов	Факультет, курс	
	ИНФ-3	М-5
Близкий к точному	48%	59%
Частично верный	29%	19%
Неверный	3%	3%
Не указан	20%	19%

Наглядно обобщенные результаты представлены в виде диаграмм (рис. 1, 2).

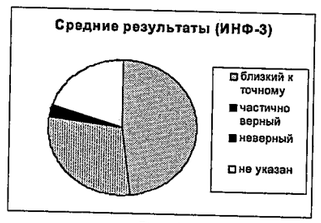


Рис. 1

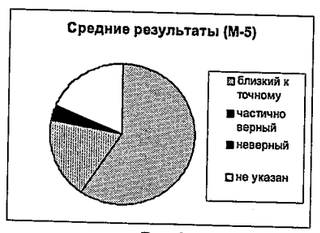


Рис. 2

Видно, что доля неверных ответов очень мала (по 3%). Огорчает, конечно, доля категории «не указан» (20% и 19%). Вообще не писали ответов, как правило, студенты, неуверенные в себе или интеллектуально неактивные («ничего не понимаю»). Это тоже отдельная проблема.

Но все же большинство студентов активно исследовали базу данных, смогли выявить основные особенности ее объектов и ответили на вопросы анкеты вполне разумно (48%+29%=77% и 59%+19%==78%).

То есть такой вариант начала изучения дисциплины вполне удачен и стимулирует познавательную активность студентов. При дальнейшей работе на последующих этапах начальные представления уточнялись, расширялись, формализовались как на лекционных, так и на практических занятиях. Исследования эффективности последующих этапов такой схемы ведутся.

Таким образом, можно предположить, что один из путей оптимальной интеграции теории и практики при формировании системы понятий курса «СУБД» представляет собой «спиральное» его изучение:

1-й «виток» – общее представление о БД (работа с готовой БД).

2-й «виток» – проектирование и создание БД, основываясь на «соображениях здравого смысла» (создание общей БД).

3-й «виток» – теоретическое обоснование всех этапов проектирования и создания БД или «формализация соображений здравого смысла» (уточнение, нормализация общей БД).

4-й «виток» – проектирование и создание БД «со знанием дела», то есть опираясь на формальные знания и собственный опыт (создание своей БД).

Такая организация дает ряд положительных моментов:

• прежде всего – деятельностный подход к формированию системы основных понятий курса, то есть вводимые и формально определяемые на лекциях понятия уже «прочувствованы» на практике и потому легче усваиваются;

• отношение к теории нормализации как к формализации соображений здравого смысла, что отражает и история ее развития [1];

• возможность при изложении теории активного использования примеров из конкретной БД, с которой студенты работают.

Изучение любой дисциплины – не только процесс усвоения некоторых знаний и формирование умений, но и средство развития человека как личности, как будущего специалиста. И деятельностный подход к этому процессу позволяет реализовать и образовательный, и развивающий потенциал дисциплины.

Примечания

1. *Дейт К.* Введение в системы баз данных. Киев; М.: «Диалектика», 1998.

2. *Рубинштейн С. Л.* Основы общей психологии. СПб.: Питер, 1999.

3. Управляемое формирование психических процессов / Под ред. П. Я. Гальперина. М., 1977.

4. *Флоренский П. А.* Термин // Мир философии. Ч. 1. М.: Политиздат, 1991.

В. А. Онегов

УВЕЛИЧЕНИЕ СКОРОСТИ СХОДИМОСТИ

Решение задачи ускорения скорости сходимости в численных методах встречается достаточно часто. Применение приемов ускорения сходимости позволяет уменьшать погрешность на порядки. Автор рассмотрел вопрос с самых общих позиций и привел три примера: ускорение сходимости в методе Эйлера, ускорение сходимости формулы трапеций численного интегрирования и ускорение сходимости предельного перехода, определяющего в качестве предела число e.

Большинство задач, решаемых вычислительной математикой, имеют одну из двух форм:

$$Ax = b \ (b \in Y) \qquad (1)$$

$$y = Ab \ \ (b \in X), \qquad (2)$$

где A – некоторый абстрактный оператор, действующий из полного нормированного пространства X в пространство Y ($A: X \to Y$).

Задачам первого типа соответствуют (в зависимости от оператора A) уравнения различного рода.

Примеры: 1) если A – вещественная функция вещественной переменной ($A: R \to R$), то речь идет о решении обычного уравнения (линейного, квадратичного, нелинейного); 2) если A – дифференциальный оператор, то решается дифференциальное уравнение (обыкновенное или с частными производными); 3) если A – матрица, то решается система линейных алгебраических уравнений.

Задачи второго типа соответствуют задачам, аналогичным вычислению значения функции (значение оператора на элементе из пространства X). Например, если A – определенный интеграл, то речь идет о нахождении значения определенного интеграла от некоторой интегрируемой функции (пространство X – множество интегрируемых на заданном интервале функций).

Основным методом решения задач (1), (2) является дискретизация, то есть замена (аппроксимация) элементов исходных пространств X и Y конечными векторами из эвклидовых пространств E^n и E^m, а оператор A заменяется (аппроксимируется) матрицей $A_{m,n}$ размерности $m*n$. Таким образом, задачи (1), (2) аппроксимируются задачами (1') и (2'):

$$A_{m,n}\, x_n = b_m \ (x_n \in E^n,\ b_m \in E^m). \qquad (1')$$

$$y_m = A_{m,n}\, b_n \ (y_m \in E^m,\ b_n \in E^n). \qquad (2')$$

Как правило, при аппроксимации задач (1), (2) задачами (1'), (2') качество аппроксимации (ее точность) характеризуется некоторым числовым параметром h, связанным с n и m. Поэтому можно обозначить $A_{m,n} = A(h)$. Для корректно поставленных задач (и исходных и аппроксимирующих) всегда справедливо утверждение о том, что если при $h \to 0$ оператор $A(h)$ в некотором смысле сходится к оператору A (по некоторой норме), то и приближенные решения x_n и y_m сходятся к точным решениям ($h \to 0$ – эквивалентно $n \to \infty$ и $m \to \infty$). Иными словами, при приближенном решении задач (1), (2) мы всегда имеем дело с процессом предельного перехода (точное решение – это предел последовательности решений аппроксимационных задач). Тогда можно говорить о скорости сходимости этого процесса и о его ускорении.

Говорят, что процесс предельного перехода имеет скорость сходимости α, если можно установить справедливость неравенства

$$|x^* - x_n| = C \cdot h^{\alpha} + O(h^{\alpha+1})$$

$$\text{или } |y^* - y_m| = C \cdot h^{\alpha} + O(h^{\alpha+1}),$$

где x^*, y^* – решения соответствующих точных задач, x_n, y_m – решения аппроксимационных задач, C – константа (не зависит от h), $|\cdot|$ – норма в соответствующем пространстве, $\alpha > 0$, $h < 1$.

В нашем контексте под ускорением сходимости будем понимать нахождение линейных комбинаций

$$\bar{x}(h) = \alpha_1 x(h_1) + \alpha_2 x(h_2) + \ldots + \alpha_p x(h_p),$$

$$\bar{y}(h) = \alpha_1 y(h_1) + \alpha_2 y(h_2) + \ldots + \alpha_p y(h_p),$$

причем $h_i \le h < 1$ при $i = 1, \ldots p$. Пусть каждое приближенное решение $x(h_i)$ и $y(h_i)$ имеет скорость сходимости α, то есть

$$|x^* - x(h_i)| = C \cdot h_i^{\alpha} + O(h_i^{\alpha+1}),$$

$$|y^* - y(h_i)| = C \cdot h_i^{\alpha} + O(h_i^{\alpha+1}).$$

Если построенные решения будет удовлетворять равенствам

$$|x^* - \bar{x}(h)| = C \cdot h^{\beta} + O(h^{\beta+1}),$$

$$|y^* - \bar{y}(h)| = C \cdot h^{\beta} + O(h^{\beta+1}) \quad \text{при } \beta > \alpha,$$

то это решение дает лучшую скорость сходимости по сравнению с решениями $x(h_i)$, из которых оно составлено (произошло улучшение сходимости).

Рассмотрим три примера ускорения сходимости в различных методах приближенных вычислений.

1. Метод Эйлера численного решения задачи Коши для обыкновенного дифференциального уравнения первого порядка

Пусть требуется найти решение задачи Коши

$$x'(t) = f(t, x),\ t \in [0; T], \text{ при } x(0) = g. \qquad (3)$$

Метод Эйлера заключается в следующем. Отрезок $[0; T]$ разбивается равноотстоящими точками (узлами) $t_k = k \cdot h$, $k = 0, 1, \ldots, n$, $h = T/n$. Это – дискретизация промежутка $[0; T]$, на котором ищется решение, – этот отрезок заменили дискретным множеством точек $\{t_k\}$. Затем в каждой узловой точке производная $x'(t_k)$ заменяется (аппроксимируется) разностным отношением

$$x'(t_k) = \frac{x(t_{k+1}) - x(t_k)}{h}.$$

ОНЕГОВ Владислав Алексеевич – кандидат физико-математических наук, доцент, заведующий кафедрой прикладной математики ВятГГУ

2 Зак. 2868

Тогда исходная задача (3) аппроксимируется дискретной задачей

$$x_{k+1} = x_k + h \cdot f(t_k, x_k),\ x_0 = g,$$
$$\text{при } k = 0, 1, \ldots, n-1. \qquad (3')$$

Алгоритмически получилась очень простая схема: по известному значению $x_0 = g$ вычисляется x_1 и т.д. Схема нахождения приближенного решения задачи Коши в узловых точках относится к рекуррентным формулам. В (3') мы воспользовались обозначением $x_k \approx x^*(t_k)$ (x_k – приближенное решение, $x^*(t_k)$ – точное решение в узловой точке).

В качестве пространства X в задаче (3) можно взять пространство непрерывно дифференцируемых на $[0; T]$ функций, Y – множество непрерывных на $[0; T]$ функций, A – дифференциальный оператор ($Ax \equiv x'(t)$). В качестве нормы и в пространстве X, и в пространстве Y возьмем $\|x\| = \max|x(t)|$, при $t \in [0; T]$.

Пространство X_n – множество векторов $x = (x_0, x_1, \ldots, x_n)$, а пространство Y_m – множество векторов $y = (y_0, y_1, \ldots, y_{n-1})$. В качестве нормы в этих пространствах возьмем первую норму вектора, согласованную с нормами в пространствах X и Y: $\|x_n\| = \max|x_k|$, по всем $k = 1, \ldots, n$. Вид дискретного оператора $A_{m,n}$ определяется тождеством $A_{m,n}x \equiv \frac{x_{k+1} - x_k}{h}$, при $k = 0, 1, \ldots, n-1$.

Выясним, какова погрешность метода Эйлера на одном шаге. Будем считать, что на k-м шаге известно точное решение, то есть $x_k = x^*(t_k)$. Тогда абсолютная погрешность $\varepsilon(t_{k+1})$ на $k+1$ шаге определяется разностью $\varepsilon(t_{k+1}) = x^*(t_{k+1}) - x_{k+1}$.

В предположении существования у точного решения $x^*(t)$ непрерывной производной третьего порядка можно $x^*(t_{k+1})$ представить в виде отрезка ряда Тейлора с остаточным членом

$$x^*(t_{k+1}) = x_k + h \cdot x^*/(t_k) + \frac{h^2}{2} \cdot \frac{d^2x^*}{dt^2}(t_k) + \frac{h^3}{6} \cdot \frac{d^3x^*}{dt^3}(q),$$

$q \in [t_k; t_{k+1}]$.

Отсюда $x^*(t_{k+1}) = x_k + h \cdot f(t_k, x_k) + \frac{h^2}{2} \cdot \frac{d^2x^*}{dt^2}(t_k) +$

$$\frac{h^3}{6} \cdot \frac{d^3x^*}{dt^3}(q) = x_{k+1} + \frac{h^2}{2} \cdot \frac{d^2x^*}{dt^2}(t_k) + \frac{h^3}{6} \cdot \frac{d^3x^*}{dt^3}(q).$$

Обозначим $\left|\frac{d^2x^*}{dt^2}(t_k)\right| = C$. Тогда имеем

$$|\varepsilon(t_{k+1})| = |x^*(t_{k+1}) - x_{k+1}| = C \cdot h^2 + O(h^3).$$

Показали, что скорость сходимости метода Эйлера на одном шаге имеет порядок 2 ($\alpha = 2$).

Выберем $h_1 = h/2$, $h_2 = h$. Опираясь на значение в точке $t_k = k \cdot h$, найдем три приближенных решения:

$$x_{k+1} = x_k + h \cdot f(t_k, x_k),$$

$$\bar{x}_{k+\frac{1}{2}} = x_k + \frac{h}{2} \cdot f(t_k, x_k),\ \bar{x}_{k+1} = \bar{x}_{k+\frac{1}{2}} + \frac{h}{2} \cdot f(t_k + \frac{h}{2}, \bar{x}_{k+\frac{1}{2}}).$$

Таким образом, в точке t_{k+1} имеем два приближенных решения: первое – вычислено при шаге разбиения h, второе – $\frac{h}{2}$. По доказанному ранее свойству имеем

$$x^*(t_{k+1}) = x_{k+1} + C \cdot h^2 + O(h^3). \qquad (4)$$

Для $\bar{x}_{k+\frac{1}{2}}$ имеем

$$x^*(t_k + \frac{h}{2}) = \bar{x}_{k+\frac{1}{2}} + (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_1) =$$

$$\bar{x}_{k+\frac{1}{2}} + (\frac{h}{2})^2 \frac{1}{2} \cdot x''(t_k) + O(h^3).$$

В точке t_{k+1} имеем

$$x^*(t_{k+h}) = \bar{x}_{k+\frac{1}{2}} + (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_1) + f(t_k + \frac{h}{2}, \bar{x}_{k+\frac{1}{2}}) +$$

$$+ (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_2) = \bar{x}_{k+1} + (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_1) + x''(\eta_2) - \frac{h}{2} \cdot$$

$$\cdot \frac{\partial f}{\partial x}(t_k + \frac{h}{2}, \xi) \cdot (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_1) = \bar{x}_{k+1} + (\frac{h}{2})^2 \cdot x''(\eta) -$$

$$- \frac{h}{2} \cdot \frac{\partial f}{\partial x}(t_k + \frac{h}{2}, \xi) \cdot (\frac{h}{2})^2 \frac{1}{2} \cdot x''(\eta_1) =$$

$$= \bar{x}_{k+1} + (\frac{h}{2})^2 \cdot x''(t_k) + O(h^3),$$

то есть

$$x^*(t_k + h) = \bar{x}_{k+1} + (\frac{h}{2})^2 \cdot x''(t_k) + O(h^3), \text{ или}$$

$$x^*(t_{k+1}) = \bar{x}_{k+1} + 2C \cdot (\frac{h}{2})^2 + O(h^3). \qquad (5)$$

Умножим равенство (4) на α_1, а (5) – на α_2 и сложим получающиеся равенства

$$\alpha_1 \cdot x^*(t_{k+1}) + \alpha_2 \cdot x^*(t_{k+1}) = \alpha_1 \cdot x_{k+1} + \alpha_2 \cdot x_{k+1} +$$

$$+ (\alpha_1 \cdot h^2 + \alpha_2 \cdot \frac{h^2}{2}) \cdot C + O(h^3).$$

Пренебрегая слагаемыми порядка h^3, выбирая α_1 и α_2 так, чтобы выполнялись два равенства

$$\begin{cases} \alpha_1 + \alpha_2 = 1, \\ \alpha_1 + \frac{\alpha_2}{2} = 0, \end{cases}$$

получаем $\alpha_1 = -1$, $\alpha_2 = 2$. С помощью линейной комбинации $-x_{k+1} + 2 \cdot \bar{x}_{k+1}$ получаем новое решение, скорость сходимости которого на единицу лучше, чем у

решений полученных при значениях шага h и $\frac{h}{2}$. Конкретнее, имеем

$$x^*(t_{k+1}1) - [-x_{k+1} + 2 \cdot \bar{x}_{k+1}] = O(h^3).$$

Итак, вдвое увеличивая количество вычислений, улучшаем точность (повышаем скорость сходимости) на порядок. Увеличение точности на порядок без приема ускорения скорости сходимости потребовало бы увеличения количества вычислений приблизительно в $1/h$ раз.

В предположении большей гладкости решения $x^*(t)$ (а значит, и функции $f(t, x)$) аналогичным образом можно повысить скорость сходимости еще на единицу и т.д.

Пример. Рассмотрим задачу Коши для обыкновенного дифференциального уравнения

$$x'(t) = -x^2 + t \cdot x + \frac{1}{2};\ x(0) = 1. \qquad (6)$$

В табл. 1 приведены результаты решения задачи (6) методом Эйлера при различных значениях шага интегрирования $h = 0.1; 0.05; 0.025; 0.0125$, причем решения распечатаны не все, а только в узлах, кратных 0.1. Через x^{**} обозначено решение, полученное методом ускорения сходимости. В четвертом столбце это решение было получено линейной комбинацией решений в методе Эйлера для $h = 0.1$ и $h = 0.05$, а в седьмом – $h = 0.025$ и $h = 0.0125$. В последнем столбце приведены точные решения. Выводы может сделать сам читатель.

2. Приближенное вычисление определенного интеграла

Пусть стоит задача вычисления определенного интеграла

$$y = \int_0^1 f(x)dx.$$

В качестве метода выберем квадратурную формулу трапеций (составную)

$$y(h) = y_n = \frac{h}{2} \cdot [f(x_0) + 2(f(x_1) + f(x_2) + \ldots + f(x_{n-1})) + f(x_n)] \qquad (7)$$

где $x_k = k \cdot h$, $k = 0, 1, \ldots, n$, $h = \frac{1}{n}$.

Найдем представление остаточного члена квадратурной формулы трапеций на отрезке $[x_k; x_{k+1}]$, то есть

$$R_1(f; x_k) = \int_{x_k}^{x_{k+1}} f(x)dx - \frac{h}{2}[f(x_k) + f(x_{k+1})].$$

В предположении существования третьей производной у подынтегральной функции $f(x)$, разлагая ее в ряд Тейлора с остаточным членом в точке x_k, используя обобщенную теорему о среднем значении интеграла, можно получить следующее представление:

$$R_1(f; x_k) = -\frac{h^3}{12} \cdot f''(x_k) + \frac{h^4}{24} \cdot [f'''(\eta_k) - f'''(\xi_k)],$$

где $\eta_k, \xi_k \in [x_k; x_{k+1}]$. Отсюда ясно, что скорость сходимости формулы трапеций на отрезке $[x_k; x_{k+1}]$ имеет порядок $O(h^3)$.

Составная формула трапеций (7) получается, как сумма простых формул трапеций при интегрировании на частичных отрезках $[x_k; x_{k+1}]$, поэтому остаточный член интегрирования по составной формуле трапеций является суммой остаточных членов на $[x_k; x_{k+1}]$, то есть

$$R_{mp}(f, h) = \int_0^1 f(x)dx - y_n = \sum_{k=0}^{n-1} R_1(f; x_k) =$$

$$-\frac{h^3}{12} \sum_{k=0}^{n-1} f''(x_k) + \frac{h^4}{24} \sum_{k=0}^{n-1} [f'''(\eta_k) - 2f'''(\xi_k)].$$

Уменьшая шаг в два раза, то есть $h_1 = \frac{h}{2}$, получим новую составную формулу трапеций с остаточным членом

$$R_{mp}(f, h_1) = \int_0^1 f(x)dx - y(h_1) = -\frac{h^3}{96} \sum_{k=0}^{n-1} f(\tilde{x}_{2k}) -$$

$$\frac{h^3}{96} \sum_{k=0}^{n-1} f(\tilde{x}_{2k+1}) + \frac{h^4}{384} \cdot \sum_{k=0}^{N-1} + [f'''(\eta_k) - 2f'''(\xi_k)].$$

Таблица 1

T	0.1	0.05	x^{**}	0.025	0.0125	x^{**}	$x^*(t)$
0.1	0.95491	0.95710	0.95931	0.95836	0.95877	0.95918	**0.95917**
0.2	0.92704	0.93059	0.93414	0.93263	0.93329	0.93395	**0.93395**
0.3	0.91265	0.91704	0.92144	0.91958	0.92041	0.92124	**0.92123**
0.4	0.90924	0.91417	0.91911	0.91703	0.91797	0.91891	**0.91890**
0.5	0.91508	0.92036	0.92566	0.92345	0.92446	0.92547	**0.92546**
0.6	0.92900	0.93452	0.94004	0.93775	0.93881	0.93987	**0.93986**
0.7	0.95017	0.95585	0.96154	0.95918	0.96028	0.96138	**0.96137**
0.8	0.97801	0.98381	0.98962	0.98722	0.98835	0.98947	**0.98946**
0.9	1.01213	1.01803	1.02392	1.02149	1.02264	1.02378	**1.02377**
1.0	1.05225	1.05822	1.06419	1.06173	1.06289	1.06405	**1.06405**

В последнем представлении $\tilde{x}_k = k \cdot \frac{h}{2}$, при $k = 0, 1, \ldots, N$, $N = 2n$. Таким образом, первые суммы (без коэффициентов) в представлениях $R_{mp}(f, h)$ и $R_{mp}(f, h_1)$ совпадают. Обозначим эту сумму через M. Вторую сумму заменим на следующую

$$\sum_{k=0}^{n-1} f(\tilde{x}_{2k+1}) = \sum_{k=0}^{n-1} f(x_k) + \frac{h}{2} \cdot \sum_{k=0}^{n-1} f'(\gamma_k) = \frac{h}{2} \cdot \sum_{k=0}^{n-1} f'(\gamma_k),$$

где $\gamma_k \in [x_k; x_{k+1}]$.

Легко заметить, что

$$n \cdot \min f(x) \le \sum_{k=0}^{n-1} f(x_k) \le n \cdot \min f(x),$$

поэтому существует такая точка $\eta \in [0; 1]$, что $\sum_{k=0}^{n-1} f(x_k) = n \cdot f(\eta) = \frac{1}{h} \cdot f(\eta)$. Рассуждая аналогично, можно получить

$$\sum_{k=0}^{n-1} [f'''(\eta_k) - f'''(\xi_k)] = \frac{1}{h} [f'''(\alpha) - f'''(\beta)],$$

$$\sum_{k=0}^{n-1} [f'''(\tilde{\eta}_k) - 2f'''(\tilde{\xi}_k)] = \frac{1}{h} [f'''(\alpha_1) - f'''(\beta_1)],$$

$$\sum_{k=0}^{n-1} f'(\gamma_k) = \frac{1}{h} f'(\gamma_k),$$

где $\alpha, \beta, \alpha_1, \beta_1, \gamma$ – некоторые числа из [0; 1]. Тогда можно записать

$$R_{mp}(f, h) = -\frac{h^2}{12} \cdot f(\eta) + O(h^3),$$

$$R_{mp}(f, h_1) = -\frac{h^2}{48} \cdot f(\eta) + O(h^3).$$

Итак, имеем $\int_0^1 f(x)dx = y(h) - \frac{h^2}{12} \cdot f(\eta) + O(h^3)$,

$$\int_0^1 f(x)dx = y(\frac{h}{2}) - \frac{h^2}{48} \cdot f(\eta) + O(h^3).$$

Умножая первое равенство на C_1, а второе – на C_2 и складывая полученные равенства, приходим к соотношению

$$(C_1 + C_2) \cdot \int_0^1 f(x)dx =$$

$$C_1 \cdot y(h) + C_2 \cdot y(\frac{h}{2}) - \frac{h^2}{12} \cdot [C_1 + \frac{1}{4} C_2] \cdot f(\eta) + O(h^3).$$

Полагая $C_1 = -\frac{1}{3}$, а $C_2 = \frac{4}{3}$, приходим к равенству

$$\int_0^1 f(x)dx = -\frac{1}{3} \cdot y(h) + \frac{4}{3} \cdot y(\frac{h}{2}) + O(h^3).$$

Получили новую формулу численного интегрирования, которая на порядок точнее двух исходных

$$\int_0^1 f(x)dx \approx -\frac{1}{3} \cdot y(h) + \frac{4}{3} \cdot y(\frac{h}{2}). \qquad (8)$$

Таким образом, рассмотренный прием улучшения сходимости квадратурной формулы трапеций можно рассматривать как один из способов получения новых неинтерполяционных квадратурных формул большей точности.

Пример. Вычислить интеграл

$$I = \int_0^1 \sin \pi x dx.$$

Решения сведены в табл. 2. В трех первых столбцах приведены приближенные значения интеграла I, полученные по составной формуле трапеций при различных значениях n, в четвертом и пятом – по формуле (8), в последнем – точное значение интеграла I.

3. Предельный переход и машинная арифметика

В качестве модельной задачи рассмотрим задачу нахождения точных знаков числа e, опираясь непосредственно на определение числа e как предела

$$(1 + \frac{1}{n})^n \xrightarrow[n \to \infty]{} e.$$

Построим последовательность чисел $\{e_n\}$, таких, что

$$e_n = (1 + \frac{1}{n})^n. \qquad (9)$$

Алгоритмически процесс построения этой последовательности прост. Если бы вычисления по формуле (9) осуществлялись точно (без округлений), то в соответствии с определением предела, построив достаточно длинную последовательность $\{e_n\}$, можно было бы получить сколь угодно точное представление числа e (с любым заданным количеством верных знаков). Однако, рассматривая фактически построенную (вычисленную на компьютере) последовательность $\{en\}$, замечаем, что характер изменения членов последовательности не совпадает с теоретически ожидаемым изменением. При увеличении n до некоторого достаточно большого N действительно происходит стабилизация верных знаков, то есть последовательность ведет себя в соответствии с теорией. Дальнейшее увеличение n не добавляет верных знаков. Наконец, наступает момент, когда верные знаки начинают замещаться сомнительными знаками, вплоть до полного искажения результата (ни одного верного знака). Эта ситуация характерна при осуществлении предельного перехода при реализации его на вычислительных устройствах.

Таблица 2

n=20	*n=40*	*n=80*	*20 – 40*	*40 – 80*	*I (точное)*
0.6353102	0.6362925	0.6365380	0.6366199	0.6366198	0.6366198

Рассмотрим структуру абсолютной погрешности $\alpha = e - e_n$. Она состоит из двух слагаемых, первое из которых определяется заменой предела последовательности e некоторым ее членом e_n, а второе – погрешностью округления и количеством цифр в представлении чисел данного вычислительного устройства (разрядностью чисел).

Рассмотрим функцию $f(n) = (1 + \frac{1}{n})^n$. Вводя новую переменную $x = \frac{1}{n}$ и доопределяя ее по непрерывности в точке $x = 0$, получаем хорошо изученную функцию $y = f(x)$, которая непрерывна в точке $x = 0$ вместе со всеми производными. Легко вычислить, что $f'(0) = \frac{e}{2}$. Таким образом, первое слагаемое в представлении абсолютной погрешности может быть записано в виде $\frac{e}{2} \cdot x (\frac{e}{2} \cdot \frac{1}{n})$. Учитывая, что мантиссы числа типа ***real*** в Pascal записываются в пяти байтах, причем приближенно (с округлением), погрешность числа $\frac{1}{n}$ равна 2^{-39}, погрешность числа $(1 + \frac{1}{n})^n$ равна 2^{-38}. Тогда вычислительная погрешность степени $(1 + \frac{1}{n})^n$ будет равна $n \cdot (1 + \frac{1}{n})^{n-1} \cdot 2^{-38} \approx n \cdot e \cdot 2^{-38}$. При больших значениях n эта погрешность будет влиять не только на знаки, в которых осуществляется округление, но и на информативные знаки. Так, например, при $n \approx 2^{38}$ величина погрешности будет около e (2, 7…), так что у соответствующего значения e_n все знаки сомнительные. В связи с этим возникает проблема, каким выбирать значение n и как видоизменить вычислительный процесс, чтобы получить максимальное количество верных знаков, (например, 11 – стандартная выдача числа типа ***real*** на Pascal). Ясно, что при очень большом n вычислительная погрешность не позволит найти даже меньше, чем одиннадцать верных знаков. С другой стороны, при небольших значениях n в принципе невозможно получить много верных знаков.

В создавшейся ситуации выходом из положения является применение приема ускорения сходимости. Вычислим при сравнительно небольших значениях n несколько ($n = n_1, n_2, \ldots, n_p$) чисел e_n. Затем с помощью линейной комбинации этих значений найдем более точное приближение. При этом, если коэффициенты линейной комбинации выбраны так, что точность ее выше точности каждого из составляющих, то получим большее количество верных знаков. Выбор небольших значений чисел n сделает вычислительную погрешность сравнительно малой.

Найдем коэффициенты упомянутых линейных комбинаций, установим точность, с которой линейная комбинация приближает число e, и проведем численный эксперимент.

Ранее введенная функция $f(x)$ может быть разложена в ряд Тейлора в точке $x = 0$ с остаточным членом

$$f(x) = f(0) + f'(0) \cdot x + \frac{f''(0)}{2} \cdot x^2 + \ldots +$$

$$+ \frac{f^{(p-1)}(0)}{(p-1)!} \cdot x^{p-1} + \frac{f^{(p)}(\eta)}{p!} \cdot x^p.$$

Возвращаясь к переменной n, учитывая, что $f(0) = e$, а $f(\frac{1}{n}) = e_n$, перепишем это равенство в следующем виде

$$e = e_n - f'(0) \cdot \frac{1}{n} - \ldots - \frac{f^{(p-1)}(0)}{(p-1)!} \cdot \frac{1}{n^p} - \frac{f^{(p)}(\eta)}{p!} \cdot \frac{1}{n^{p+1}}.$$

Составим линейную комбинацию $E = C_1 \cdot E_1 + C_2 \cdot E_2 + \ldots + C_p \cdot E_p$, выбрав в качестве C_i решения системы уравнений

$$C_1 + C_2 + \ldots + C_p = 1,$$
$$C_1 \cdot h_1 + C_2 \cdot h_2 + \ldots + C_p \cdot h_p = 0,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$
$$C_1 \cdot h_1^p + C_2 \cdot h_2^p + \ldots + C_p \cdot h_p^p = 0,$$

где через E_i обозначены e_n при соответствующих значениях n_i, а $h_i = \frac{1}{n_i}$, при $i = 1, \ldots, p$. Написанная система линейных уравнений разрешима единственным образом, так как ее определитель – это транспонированный определитель Вандермонда. Удобнее всего величины h_i выбрать равными $\frac{1}{2^i} \cdot h$ при некотором фиксированном h. Приведем системы линейных алгебраических уравнений для определения коэффициентов C_i и их решения при различных значениях параметра p.

1. p=1: $C_1 = 1$. Точность – порядка $O(h) = O(\frac{1}{n})$.

2. p=2: $\begin{cases} C_1 + C_2 = 1, \\ C_1 + \frac{1}{2} \cdot C_2 = 0. \end{cases}$ $C_1 = -1, C_2 = 2.$

3. p=3: $\begin{cases} C_1 + C_2 + C_3 = 1, \\ C_1 + \frac{1}{2} \cdot C_2 + \frac{1}{4} \cdot C_3 = 0, \\ C_1 + \frac{1}{4} \cdot C_2 + \frac{1}{8} \cdot C_3 = 0. \end{cases}$ $C_1 = \frac{1}{3}, C_2 = -2, C_3 = \frac{8}{3}.$

Таблица 3

n	P = 1	p = 2	p = 3	p = 4	p = 5
16	2,6379284974	2,7160517614	2,7182491138	2,7182815824	2,712818276
32	2,6769901294	2,7176997757	2,7182775238	2,7182818123	2,7182818283
64	2,6973449526	2,7181330868	2,7182812763	2,7182818273	2,7182818284
128	2,7077390197	2,7182442289	2,7182817584	2,7182818283	2,7182818284
256	2,7129916243	2,7182723761	2,7182818196	2,7182818284	
512	2,7156320002	2,7182794587	2,7182818273		
1024	2,7169557294	2,7182812351	2,7182818285		
2048	2,7176184823	2,7182816801			
4096	2,7179500813	2,7182817911			
8192	2,7181159362				

4. p=4: $\begin{cases} C_1+C_2+C_3+C_4=1, \ C_1=-0{,}0476190476, \\ C_1+\frac{1}{2}\cdot C_2+\frac{1}{4}\cdot C_3+\frac{1}{8}\cdot C_4=1, \ C_2=0{,}6666666667, \\ C_1+\frac{1}{4}\cdot C_2+\frac{1}{16}\cdot C_3+\frac{1}{64}\cdot C_4=0, \ C_3=-0{,}2666666667, \\ C_1+\frac{1}{8}\cdot C_2+\frac{1}{64}\cdot C_3+\frac{1}{512}\cdot C_4=0, \ C_4=3{,}0476190476. \end{cases}$

5. p=5: $\begin{cases} C_1+C_2+C_3+C_4+C_5=1, \\ C_1+\frac{1}{2}\cdot C_2+\frac{1}{4}\cdot C_3+\frac{1}{8}\cdot C_4+\frac{1}{16}\cdot C_5=0, \\ C_1+\frac{1}{4}\cdot C_2+\frac{1}{16}\cdot C_3+\frac{1}{64}\cdot C_4+\frac{1}{256}\cdot C_5=0, \\ C_1+\frac{1}{8}\cdot C_2+\frac{1}{64}\cdot C_3+\frac{1}{512}\cdot C_4+\frac{1}{4096}\cdot C_5=0, \\ C_1+\frac{1}{16}\cdot C_2+\frac{1}{256}\cdot C_3+\frac{1}{4096}\cdot C_4+\frac{1}{65536}\cdot C_5=0, \end{cases}$

C_1=0,00317460316, C_2=–0,09523809522,

C_3=0,88888888889, C_4=–3,0476190476.

Находить линейные комбинации для p>5 нет смысла, потому что при решении систем линейных алгебраических уравнений высокого порядка с коэффициентами отмеченного характера, например, методом Гаусса вычислительная погрешность чисел C_i становится больше чем 10^{-10} и улучшения результата не происходит.

В табл. 3 приведены результаты численного эксперимента.

Автор не может удержаться от комментариев по поводу поведения чисел, приведенных в этой таблице. Во-первых, числа e_n в первом столбце, начиная с n=32 и заканчивая n=512, уже содержат всю информацию, которая необходима для решения поставленной задачи (нахождение числа е с точностью одиннадцать верных знаков). При этом каждое из них дает приближение к числу е лишь с точностью два верных знака. Во-вторых, используя прием ускорения сходимости с помощью линейной комбинации пяти en при n=32, 64, 128, 256, 512, мы производим лишь 992 операции возведения в степень, тогда как нахождение приближенного значения числа *e* без ускорения сходимости дает максимально возможную точность девять верных знаков только на предельных значениях целых чисел типа ***longint*** (количество операций возведения в степень – 2^{31}).

Примечания

1. *Коллатц Л.* Функциональный анализ и вычислительная математика. М.: Мир, 1969.

2. *Березин И. С., Жидков Н. П.* Методы вычислений. Т. 1, 2. М.: ГФМЛ, 1959.

3. *Мысовских И. П.* Лекции по методам вычислений. М.: ГФМЛ, 1962.

Р. В. Быкова

ДИСТАНЦИОННОЕ ОБУЧЕНИЕ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ

В статье раскрываются особенности построения компьютерных обучающих программ при организации дистанционного обучения по двум специальностям: «Бухучет, аудит» и «Финансы, кредит» в финансово-экономическом институте.

Дистанционное обучение (ДО) – это целенаправленный процесс взаимодействия преподавателя и студентов между собой и со средствами обучения, не связанный с их расположением в пространстве и во времени.

Главной особенностью ДО является организация самостоятельной работы студентов с учебно-методическим обеспечением.

При дистанционном обучении диалог студента с преподавателем осуществляется компьютерными

БЫКОВА Раиса Викторовна – старший преподаватель кафедры АОЗИ ВЗФЭИ

средствами. К компьютерным учебным средствам относятся компьютерные электронные учебники и компьютерные обучающие программы (КОПР).

Выделим характерные особенности ДО:

1. Обучающиеся занимаются в удобное для себя время и в удобном темпе. Каждый может учиться столько, сколько ему лично необходимо для освоения курса дисциплины и получения необходимых знаний по выбранным дисциплинам.

2. В основу программ ДО закладывается модульный принцип. Каждая отдельная дисциплина (учебный курс), который освоен обучаемым, адекватен по содержанию определенной предметной области. Это позволяет из набора независимых учебных курсов формировать учебный план, отвечающий индивидуальным или групповым потребностям.

3. В ДО используются все виды информационных технологий: компьютеры, компьютерные сети, мультимедиа системы и т. д.

Перечисленные особенности определяют преимущества ДО перед другими формами получения образования и предъявляют определенные специфические требования как к преподавателю, так и к слушателю, ни в коем случае не облегчая, а подчас увеличивая трудозатраты и того и другого.

В ДО средства обучения реализуются через средства новых информационных технологий. Они включают в себя магнитофоны, видеомагнитофоны, кинопроекторы, диапроекторы, кодоскопы, компьютеры.

В ВЗФЭИ работает центр ДО с 1997 года. Институт включен во всероссийский эксперимент по дистанционному образованию.

Дистанционное обучение проводится в ВЗФЭИ по двум специальностям:

- бухучет и аудит;
- финансы и кредит.

В ВЗФЭИ ДО проводится с помощью компьютерных обучающих программ.

КОПРы предназначены для самостоятельного обучения студентов, не посещающих лекции по каким-либо причинам, либо для получения дополнительного образования. Эта программа может использоваться:

1) для коррекции усвоения теории;
2) самоконтроля;
3) контроля допуска к экзамену или зачету.

Структура КОПР:

- введение;
- блоки учебного материала;
- необходимый дополнительный материал и нормативно-справочная база;
- задания для самоконтроля и контроля;
- заключение.

Во введении коротко описаны цели и задачи изучения дисциплины.

Нумерация тем в КОПР и нумерация тем в учебной программе совпадают.

Учебный материал каждой темы структурирован поблочно, начинается с изложения теоретического материала и определений и заканчивается контрольными вопросами (рис. 1).

Основной материал представлен в максимально наглядной форме. Фрагменты текста могут быть представлены в виде таблиц, рисунка, блок-схемы (рис. 2).

КОПР полностью охватывает весь курс изучаемых дисциплин по разным специальностям (рис. 3).

КОПР использует диалог, алгоритм работы программы зависит от ответов студента. В зависимости

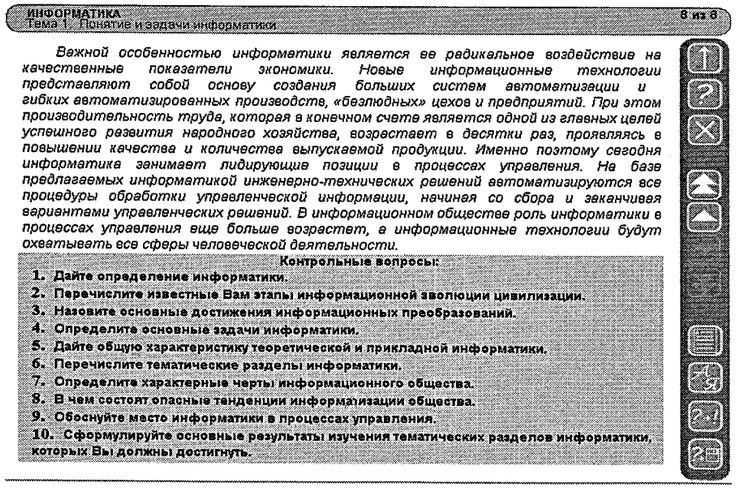


Рис. 1

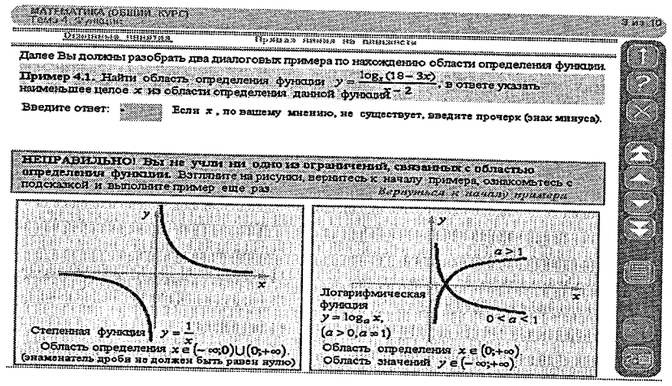


Рис. 2

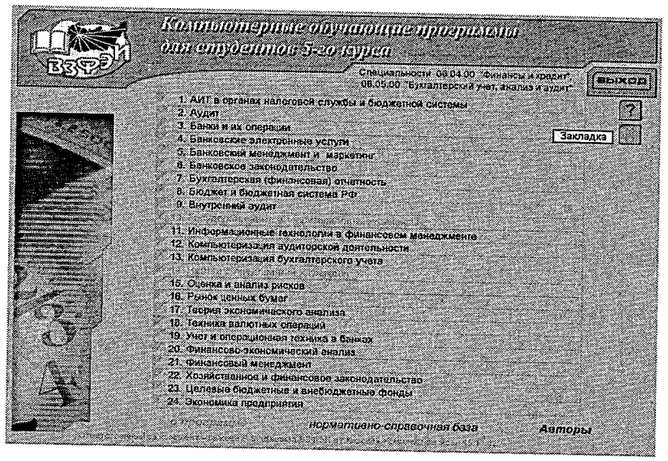


Рис. 3

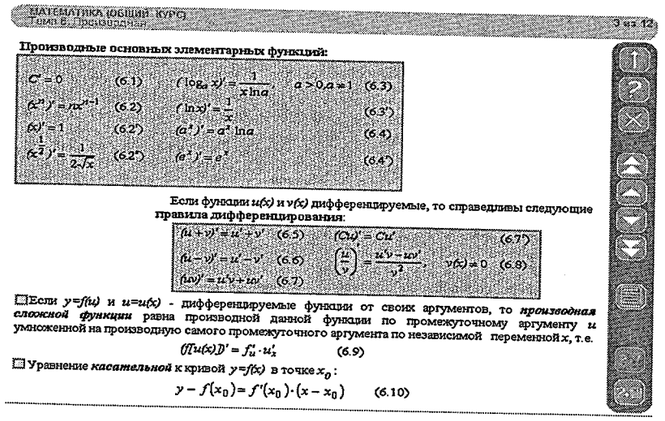


Рис. 4

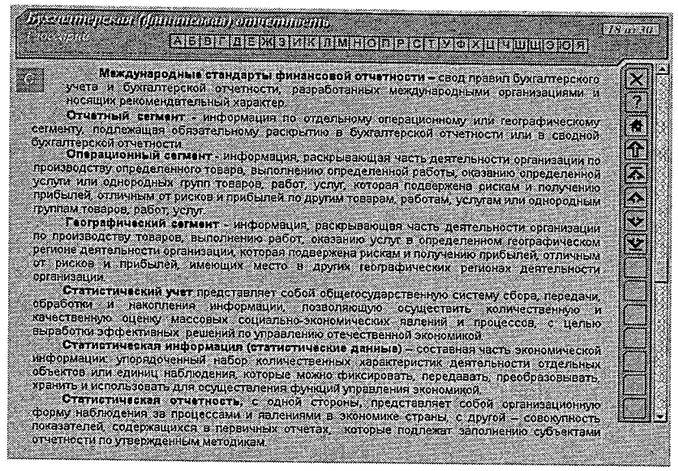


Рис. 5

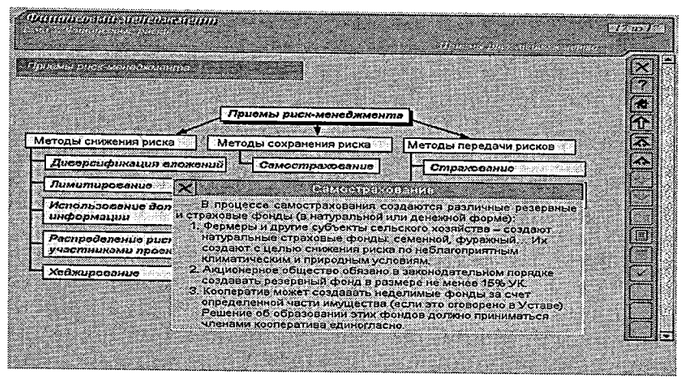


Рис. 6

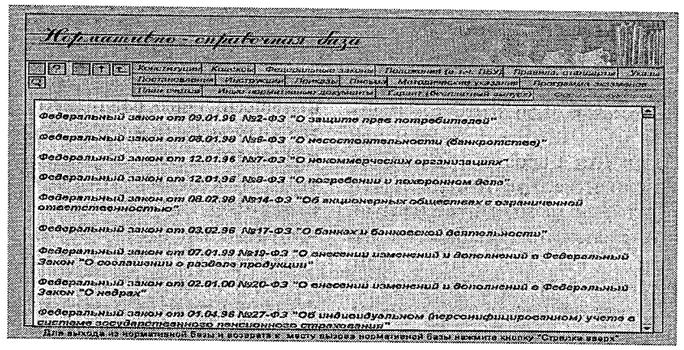


Рис. 7

от ответа программа выдает рекомендации, подсказки, пояснения.

КОПР осуществляет взаимодействие с учащимся, цель которого – обеспечить лучшее понимание материала и самоконтроль его усвоения, предполагает активную работу с учебным материалом. В программе используются мультимедиа и гипертекст.

Мультимедиа – представление информации в виде графиков и изображений, анимации, видеосюжетов с возможным звуковым сопровождением.

Гипертекстовая структура КОПР предполагает:

1) конспективное изложение материала;

2) свободное перемещение по тексту, осуществляемое с помощью экранных кнопок;

3) использование перекрестных ссылок, которые облегчают организацию обратных связей – при неправильных ответах студента в процессе разбора типовых примеров или самоконтроля выдаются корректирующие пояснения (рис. 2).

Дополнительный справочный материал содержит:

1) терминологический словарь;

2) поясняющие расшифровки по тексту;

3) нормативную базу и ссылки на рекомендуемую литературу;

4) справочные данные, необходимые при решении расчетных задач, например формулы из высшей математики (рис. 4);

5) примеры из хозяйственной практики отечественных и зарубежных фирм;

6) ситуационный анализ (анализ реальных или гипотетических экономических ситуаций, имитационные тренажеры).

Терминологический словарь охватывает все термины, которые могут быть непонятны (рис. 5).

При ссылке одного термина на другой – перекрестные ссылки.

Поясняющие расшифровки по тексту – элемент гипертекстовых компьютерных учебников. Вызываются только в случае необходимости

- расшифровки терминов;
- справки библиографической или исторической;
- содержания элементов блок-схем (рис. 6).

Нормативная база – содержит минимальный набор нормативных документов для изучения дисциплины (рис. 7).

КОПР содержит тестовые задания для самоконтроля и контроля.

1. Контроль заключается в проверке результатов теоретического и практического усвоения учебного материала. Проводится после прохождения изучаемой дисциплины и позволяет оценить полученные знания, умения и навыки.

2. Самоконтроль включает также и элементы обучения, так как позволяет узнать правильные ответы. С его помощью обучаемый может самостоятельно, в любой момент обучения, проверить полученные знания.

В целом ДО как одна из форм получения знаний может помочь решить задачи, стоящие перед системой образования по предоставлению широким слоям населения доступного и качественного образования.

В. А. Онегов, Е. С. Лосева

КОМПЬЮТЕРНАЯ МОДЕЛЬ «РАВНОВЕСИЕ ЗАКРЕПЛЕННОЙ МЕМБРАНЫ»

В статье подробно рассматривается одна из задач компьютерного моделирования – расчет куполообразного перекрытия (мембраны). Представлены все этапы составления соответствующей компьютерной модели. Результаты, изложенные в данной статье, докладывались авторами на Международной конференции по компьютерному моделированию (г. - Санкт-Петербург, июнь 2003 г.).

Уже третий год подготовка студентов по специальности 0300100 «Учитель информатики» ведется по учебным планам, в которые введено много новых учебных дисциплин: математическая логика, дискретная математика, элементы абстрактной и компьютерной алгебры, теоретические основы информатики, уравнения математической физики, теория алгоритмов, исследование операций, компьютерное моделирование, искусственный интеллект. Введение этих предметов в структуру учебной подготовки по специальности «учитель информатики» говорит о повышении роли теоретических предметов в ней. Кроме того, отчетливо прослеживается общее направление подготовки – освоение методологии информатики (математическое, логическое, информационное и компьютерное моделирование).

Перечень теоретических дисциплин говорит о необходимости основательной предварительной математической подготовки и корреляции этих дисциплин между собой и с теоретической математикой. Ранее предпринимаемые попытки готовить специалистов по информатике без серьезных математических и теоретических (по информатике) знаний не выдержали проверки жизнью. Названный подход возможен лишь в отношении информационных технологий (с известными оговорками), к которым, конечно, не сводится информатика.

В качестве иллюстрации глубокой интеграции математики и различных теоретических дисциплин по информатике вниманию читателя предлагается эта статья, являющаяся изложением одной характерной темы компьютерного моделирования «Равновесие закрепленной мембраны». При рассмотрении этой задачи, во-первых, отслеживаются все классические этапы построения компьютерной модели: постановка задачи, математическая модель, вычислительная модель, алгоритм, компьютерная модель. Во-вторых, специфика задачи связывает теоретическую и строительную механику, уравнения математической физики, численные методы, компьютерную графику.

ОНЕГОВ Владислав Алексеевич – кандидат физико-математических наук, доцент, заведующий кафедрой прикладной математики

ЛОСЕВА Елена Сергеевна – ассистент кафедры прикладной математики

Постановка задачи

При проектировании и строительстве зданий различного назначения часто используются покрытия в виде куполов. Примерами зданий такого типа являются купольные сооружения (церкви, мечети), здания цирков, крытые рынки, крытые стадионы и тому подобное. Для возведения таких покрытий используются обычные строительные материалы: металлы, композиты, железобетон. Как правило, купола жестко крепятся на основаниях (стенах). Можно поставить следующие вопросы. 1) Какую форму купола предпочтительнее выбрать? 2) Каковы предельные нагрузки на купол (метеорологические осадки, ветер, собственный вес конструкции), не приводящие к разрушению здания? 3) Какими должны быть характеристики строительного материала при выбранной форме купола?

Построить компьютерную модель, работа с которой позволяет, так или иначе, дать ответ на поставленные вопросы. Организовать вывод на экран дисплея поверхности купола при различных нагрузках и характеристиках строительного материала.

Математическая модель

Введем следующую математическую абстракцию – мембрана. Мембраной назовем натянутую пленку, которая не сопротивляется изгибу, то есть изменению формы, не вызывающему изменения площади произвольно взятого участка мембраны. Будем предполагать, что работа внешней силы, вызывающей изменение площади некоторого участка, пропорциональна этому изменению (аналог закона Гука). Положительный коэффициент пропорциональности T не зависит ни от формы этого участка, ни от его положения. Этот коэффициент называется натяжением мембраны.

Работа внутренних сил упругости равна по абсолютной величине работе внешних сил, вызывающих изменение площади, и противоположна ей по знаку.

Пусть в состоянии покоя (без натяжения) мембрана располагается в плоскости (x, y) и имеет форму некоторой плоской области G с границей L (рис. 1).

Предположим, что на мембрану действует некоторая сила, плотность которой в точке (x, y) равна $f(x, y)$, а направление перпендикулярно плоскости Oxy. Под действием этой силы мембрана прогнется и примет форму некоторой поверхности, уравнение которой запишем в виде $U = U(x, y)$. Ось U перпендикулярна плоскости Oxy.

В курсе «Уравнения математической физики» выводится уравнение, которому удовлетворяет функция $U(x, y)$. Этот вывод осуществляется при следующих ограничениях. Во-первых, предполагается, что в интересующем нас положении мембрана не сильно изогнута. Другими словами, что $\frac{\partial U}{\partial x}$ и $\frac{\partial U}{\partial y}$ малы по абсолютной величине и можно пренебречь высшими степенями этих производных. Во-вторых, предполагается, что под действием силы $f(x, y)$ точки мембраны двигаются только по перпендикулярам к плоскости Oxy, так что их координаты (x, y) при этом не меняются.

При сделанных предположениях доказано, что для положения равновесия функция $U(x, y)$ в любой внутренней точке удовлетворяет дифференциальному уравнению в частных производных

$$T \cdot \Delta U = -f, \text{ где } \Delta U \equiv \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}. \qquad (1)$$

Это уравнение называется уравнением Пуассона, а оператор ΔU – оператор Лапласа.

Для единственной разрешимости одного уравнения Пуассона недостаточно. Необходимо положить некоторые условия на границе (граничные условия). Пусть край мембраны закреплен вдоль некоторой пространственной кривой, проектирующейся в L. Иными словами, на границе L функция $U(x, y)$ принимает те же значения, что и функция $\varphi(x, y)$.

$$U(x, y)\big|_L = \varphi(x, y). \qquad (2)$$

Окончательная форма математической модели – дифференциальное уравнение в частных производных (1) и граничные условий (2). Эта математическая модель носит название задачи Дирихле. В теории уравнений с частными производными задача Дирихле полностью исследована. Доказано существование и единственность решения при некоторых ограничениях, налагаемых на гладкость границы L и граничной функции $\varphi(x, y)$.

Вычислительная модель

Вычислительная модель предполагает дискретизацию непрерывной математической модели с целью максимального ее приближения к виду, для которого возможно получение решения на компьютере.

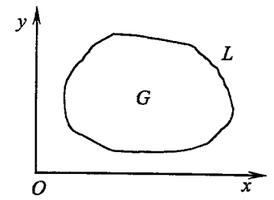


Рис. 1

1. Дискретизация области G. Выбираем h – шаг по оси Ox и τ – шаг по оси Oy. Построим две системы взаимно перпендикулярных прямых

$$x = x_k = k \cdot h,\ k = 0,1,...$$

$$y = y_i = i \cdot \tau,\ i = 0,1,...$$

Пересечение этих двух семейств прямых дает систему точек $\{(x_k, y_i)\}$ – узлов сетки (рис. 2).

Множество узлов $(x_k, y_i) \in G$ назовем $\overline{G}_{h,\tau}$. Исходную область G заменяем (дискретизируем) множеством узловых точек $\overline{G}_{h,\tau}$.

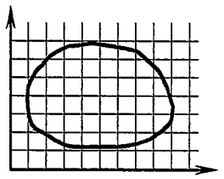


Рис. 2

2. Дискретизация производных. Множество $\overline{G}_{h,\tau}$ разобьем на два типа. Назовем внутренними узлами такие узлы (x_k, y_i) из $\overline{G}_{h,\tau}$, для которых соседние узлы $(x_{k+1}, y_i), (x_{k-1}, y_i), (x_k, y_{i-1}), (x_k, y_{i+1})$ принадлежат $\overline{G}_{h,\tau}$. Множество внутренних узлов обозначим $\overline{G}_{h,\tau}$. Остальные узлы из $\overline{G}_{h,\tau}$ будем называть граничными.

В каждом внутреннем узле сделаем замену производных разностными отношениями (аппроксимируем) в соответствии с формулами

$$\left.\frac{\partial^2 U}{\partial x^2}\right|_{x_k, y_i} \approx \frac{U(x_{k-1}, y_i) - 2 \cdot U(x_k, y_i) + U(x_{k+1}, y_i)}{h^2}$$

$$\left.\frac{\partial^2 U}{\partial y^2}\right|_{x_k, y_i} \approx \frac{U(x_k, y_{i-1}) - 2 \cdot U(x_k, y_i) + U(x_k, y_{i+1})}{\tau^2}.$$

Известно, что точность аппроксимации

$$\frac{\partial^2 U}{\partial x^2} - O(h^2), \text{ а } \frac{\partial^2 U}{\partial y^2}\ O(\tau^2).$$

Подставляя во всех внутренних точках в уравнение Пуассона (1) соответствующие разностные отношения и обозначив $U_{k,i} \approx U(x_k, y_i)$, получаем

$$T \cdot \left(\frac{U_{k-1,i} - 2 \cdot U_{k,i} + U_{k+1,i}}{h^2} + \frac{U_{k,i-1} - 2 \cdot U_{k,i} + U_{k,i+1}}{\tau^2} \right) = -f(x_k, y_i) \quad (3)$$

для всех $(x_k, y_i) \in G_{h,\tau}$.

Будем считать, что в множестве $\overline{G}_{h,\tau}$ всего N внутренних точек. Таким образом, (3) представляет собой систему из N линейных уравнений относительно $U_{k,i}$. Если N_1 – количество граничных узлов, то для того, чтобы можно было говорить о решении системы линейных алгебраических уравнений, необходимо дополнить систему (3) N_1 уравнениями. Сделаем это с помощью метода, известного как метод простого сноса учета граничных условий. Точнее, сделаем следующее:

а) если граничный узел $(x_k, y_i) \in L$, то полагаем $U_{k,i} = \varphi(x_k, y_i)$;

б) если граничный узел $(x_k, y_i) \notin L$, то полагаем

$$U_{k,i} = \varphi(x, y), \quad (4)$$

где точка (x, y) – ближайшая к граничной точке (x_k, y_i), лежащая на границе L.

Если в (4) возможны два варианта, то выбирается любой из них. Общее количество равенств N_1.

Учет граничных условий простым сносом и систем (3) приводит к системе N линейных алгебраических уравнений с N неизвестными.

Замечание. Поскольку оси координат Ox и Oy в рассматриваемой задаче равноправны, то шаги по Ox и по Oy можно выбирать одинаковыми, то есть $h = \tau$. В дальнейшем будем предполагать, что это так.

Итак, вычислительная модель представляет собой систему из N уравнений с N неизвестными, которую запишем в виде

$$U_{k-1,i} + U_{k+1,i} - 4 \cdot U_{k,i} + U_{k,i-1} + U_{k,i+1} = -\frac{h^2}{T} \cdot f(x_k, t_i) \quad (5)$$

для всех (k, i) таких, что $(x_k, t_i) \in G_h$ $(G_h = G_{h,h})$. В граничных узлах полагаем $U_{k,i} = \varphi(x, y)$, где точка (x, y) – ближайшая к точке (x_k, t_i).

Встает вопрос о методе решения этой системы. Из курса «Численные методы» известны две основные группы таких методов: 1) точные методы (типа метода Гаусса); 2) методы последовательных приближений. При обосновании выбора метода решения полученной системы проведем анализ структуры матрицы этой системы. Для простоты, не затемняющей сути дела, предполагаем, что область G есть квадрат $\{0 \le x \le 1,\ 0 \le y \le 1\}$. В этом случае все граничные точки принадлежат прямым $x = 0$, $x = 1$, $y = 0$, $y = 1$, а множество внутренних узлов $\{k \cdot h, i \cdot h\}_{k=1,i=1}^{n-1,n-1}$, где $h = \frac{1}{n}$.

Взяв $n = 4$, введем вектор неизвестных $\{X_j\}_{j=1}^{9}$, компоненты которого соответствуют неизвестным $U_{k,i}$ следующим образом

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$U_{1,1}$	$U_{1,2}$	$U_{1,3}$	$U_{2,1}$	$U_{2,2}$	$U_{2,3}$	$U_{3,1}$	$U_{3,2}$	$U_{3,3}$

Тогда исследуемая система может быть записана в векторно-матричном виде

$$A \cdot X = -F \cdot h^2,$$

где X – описанный выше вектор неизвестных,

$$F = \frac{1}{T}(f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, f_{3,3}),$$

а матрица A имеет вид:

-4	1	0	1	0	0	0	0	0
1	-4	1	0	1	0	0	0	0
0	1	-4	0	0	1	0	0	0
1	0	0	-4	1	0	1	0	0
0	1	0	1	-4	1	0	1	0
0	0	1	0	1	-4	0	0	1
0	0	0	1	0	0	-4	1	0
0	0	0	0	1	0	1	-4	1
0	0	0	0	0	1	0	1	-4

Можно заметить следующее:

1) как следствие симметричности области G матрица A симметрична;

2) в 1-й, 3-й, N-й, N-2-й строках – 3 неизвестных элемента;

3) во 2-й, 4-й, N-1-й, N-3-й – 4 неизвестных элемента;

4) в 5-й строке – 5 неизвестных элементов;

5) при уменьшении шага разбиения увеличивается значение $N = (n-1)2$, при этом свойства 1), 2), 3), 4) сохраняются, а между 4-й и $(N-3)$-й строк появляется группа строк, аналогичных по структуре 5-й строке приведенной матрицы.

Итак, наибольшее количество неизвестных в строке матрицы системы равно 5 – все остальные (подавляющее большинство!) элементы равны нулю. Матрицы, обладающие такими свойствами, называются «редкими».

Поскольку в силу точности использованной при построении модели аппроксимации, характеризующейся величиной $O(h^2)$, приемлемая точность $U_{k,i} \approx U(x_k, t_i)$ может быть достигнута только при малых значениях шага h, то есть больших n. Например, в качестве допустимого значения можно взять $h = 0{,}01$, или $n = 100$. В этом случае порядок матрицы системы линейных уравнений равен $N = 99 \times 99 = 9801$, а всего элементов матрицы $N \times N = 9801 \times 9801 = 96059601$. При этом ненулевых элементов не более $5 \cdot 9801 = 49005$, а нулевых не менее $96059601 - 49005 = 960105596$.

Проведенное исследование позволяет сделать следующие выводы в отношении применения метода Гаусса для решения систем линейных алгебраических уравнений, полученных в вычислительной модели.

1. Если область G произвольной конфигурации (не обладает симметрией), то получение матрицы A весьма трудоемко и плохо алгоритмизуемо.

2. Структура матрицы системы A такова, что подавляющее большинство хранимых элементов равно нулю.

3. Прямой ход метода Гаусса приводит исходную систему к эквивалентной системе с верхней треугольной матрицей, которая структурно значительно хуже (в отношении количества ненулевых элементов) исходной «редкой» матрицы. Это наводит на мысль, что если для решения системы с «редкой» матрицей применим метод из группы «точных», то это должен быть какой-то специальный метод, отличный от метода Гаусса.

Отмеченные недостатки присущи только первой группе методов решения систем линейных алгебраических уравнений – «точным» или конечным.

В методах второй группы – последовательных приближений – их, вообще говоря, не возникнет. Поэтому при решении систем уравнений с «редкими» матрицами предпочтение отдается методам последовательных приближений. Кроме того, эти методы отличаются устойчивостью относительно вычислительной погрешности, то есть абсолютная величина вычислительной погрешности не возрастает при проведении очередной итерации. Последнее обстоятельство очень важно при решении систем высокого порядка.

В качестве метода решения системы (5) выберем метод простых итераций, состоящий в последовательном нахождении векторов $X^{(n)} = \left\{U_{k,j}^{(n)}\right\}$, удовлетворяющих рекуррентному соотношению

$$U_{k,i}^{(n)} = \frac{U_{k-1,i}^{(n-1)} + U_{k+1,i}^{(n-1)} + U_{k,i-1}^{(n-1)} + U_{k,i+1}^{(n-1)}}{4} + \frac{h^2}{4} \cdot F_{k,i} \quad (6)$$

при $n = 1, 2, \ldots$ для всех внутренних узлов.

Процесс нахождения последовательных приближений, задаваемых формулой (6), начинается с некоторого начального приближения $X^{(0)} = \left\{U_{k,i}^{(0)}\right\}$.

В привычных векторно-матричных обозначениях систему равенств (6) можно записать в виде

$$X^{(n)} = BX^{(n-1)} + G, \quad (7)$$

где $G_{k,i} = \frac{h^2 F_{k,i}}{4}$, а структура матрицы B в условиях рассмотренного ранее примера такова:

0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0	0	0	0
$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0	0	0
0	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$	0	0	0
$\frac{1}{4}$	0	0	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0
0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0
0	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$
0	0	0	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$	0
0	0	0	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$
0	0	0	0	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0

Таким образом, алгоритм решения системы (5) чрезвычайно прост.

1. Выбирается некоторый начальный вектор $X^{(0)} = \left\{U_{k,i}^{(0)}\right\}$ во всех внутренних точках.

2. Для всех векторов $X^{(n)}$ в граничных узлах задаются граничные значения.

3. Во всех внутренних точках провести вычисления по формуле (6).

4. Повторяется пункт 3 до тех пор, пока не будут выполняться условия достижения заданной точности вычислений.

В качестве критерия точности выберем следующий прием. Пусть ε – требуемая точность вычислений. Тогда будем повторять итерации, то есть пункт 3 до тех пор, пока не выполнится неравенство

$$\left\|X^{(n)} - X^{(n-1)}\right\| = \max_{k,i}\left|U_{k,i}^{(n)} - U_{k,i}^{(n-1)}\right| \le \varepsilon.$$

Замечание. В программе нет необходимости сохранять все последовательные приближения. Достаточно иметь два массива: $X1 \sim X^{(n-1)}$ и $X2 \sim X^{(n)}$.

Однако, создавая программу решения любой задачи выбранным методом, надо быть уверенным в том, что этот метод приведет к решению данной задачи, а не к чему-нибудь совсем иному. В нашем случае речь идет о сходимости последовательности $X^{(n)}$. Если построенная последовательность сходится к X^* при $n \to \infty$, то X^* – решение системы, и все в порядке. В противном случае выбранный метод не приводит к решению.

Убедимся в сходимости предложенного процесса построения последовательности приближений $X^{(n)}$.

Из теории известно, что достаточным условием сходимости последовательности $X^{(n)}$ является выполнение неравенства

$$\|B\| < 1,$$

где $\|B\|$ – некоторая норма матрицы B, согласованная с соответствующей нормой вектора X.

В качестве нормы вектора X выберем так называемую первую норму вектора

$$\|X\|_1 = \left\|\left\{U_{k,i}\right\}\right\|_1 = \max_{k,i}\left|U_{k,i}\right|.$$

Тогда норма $\|B\|_1$ задается формулой

$$\|B\|_1 = \max_i \sum_r \left|b_{i,r}\right|. \qquad (8)$$

В формуле (8) вычисляются суммы модулей элементов матрицы B в каждой строке, а затем находится максимальная сумма.

В нашем случае $\|B\|_1 = \max\left\{\frac{1}{2}, \frac{3}{4}, 1\right\} = 1.$

Это следует из того, что имеются строки, в которых четыре элемента равны $\frac{1}{4}$. Таким образом, классическое достаточное условие сходимости не выполнено.

Проведем более тонкое исследование. Разобьем все внутренние узлы G_h на m типов:

1-й тип – множество всех узлов, у которых хотя бы один соседний узел граничный;

2-й тип – множество всех узлов, у которых хотя бы один соседний узел принадлежит первому типу; и так далее.

Введем в рассмотрение матрицы $D_j = B^j$, где B^j – j-я степень матрицы B. Установим основные неравенства, связанные с матрицами D_j. Возьмем произвольный вектор $X, \|X\|_1 \ne 0$.

1. $\left\|D_j X\right\|_1 = \left\|B^j X\right\|_1 \le \|B\|_1 \cdot \left\|B^{j-1} X\right\|_1 \le \ldots \le \|B\|_1^j \cdot \|X\|_1$

В силу того, что $\|B\|_1 = 1$, из полученного неравенства следует, что $\|D_j\|_1 \le 1$. Иными словами, при любом $j = 1, 2 \ldots$ выполнено неравенство

$$\|D_j X\| \le \|X\|_1.$$

2. Введем обозначения $\left|X\right|_{(j)} = \left|U_{k,i}\right|_{(j)}$ – абсолютная величина компоненты j-го типа. Тогда

$$\left|D_1 X\right|_{(1)} \le \frac{3}{4} \cdot \|X\|_1 = \left(1 - \frac{1}{4}\right) \cdot \|X\|_1$$

для всех узлов первого типа;

$$\left|D_2 X\right|_{(2)} \le \frac{1}{4} \cdot \max\left|D_1 X\right|_{(1)} + \frac{3}{4} \cdot \|X\|_1 \le$$

$$\le \left(\frac{1}{4} \cdot \frac{3}{4} + \frac{3}{4}\right) \cdot \|X\|_1 = \frac{15}{16} \cdot \|X\|_1 = \left(1 - \frac{1}{16}\right) \cdot \|X\|_1 = \left(1 - \left(\frac{1}{4}\right)^2\right) \cdot \|X\|_1$$

для всех узлов второго типа;

$$\left|D_3 X\right|_{(3)} \le \frac{1}{4} \cdot \max\left|D_2 X\right|_{(2)} + \frac{3}{4} \cdot \|X\|_1 \le$$

$$\le \left(\frac{1}{4} \cdot \left(1 - \frac{1}{16}\right) + \frac{3}{4}\right) \cdot \|X\|_1 = \left(1 - \frac{1}{64}\right) \cdot \|X\|_1 = \left(1 - \left(\frac{1}{4}\right)^3\right) \cdot \|X\|_1$$

для всех узлов третьего типа.

И вообще, по индукции

$$\left|D_j X\right|_{(j)} \le \frac{1}{4} \cdot \max\left|D_{j-1} X\right| + \frac{3}{4} \cdot \|X\|_1 \le$$

$$\le \left(\frac{1}{4} \cdot \left(1 - \left(\frac{1}{4}\right)^{j-1}\right) + \frac{3}{4}\right) \cdot \|X\|_1 = \left(1 - \left(\frac{1}{4}\right)^j\right) \cdot \|X\|_1$$

3. Из предыдущих пунктов следует, что

$$\left|D_j X\right|_{(1)} \le \left(1 - \frac{1}{4}\right)^j \cdot \|X\|_1$$

для всех узлов первого типа;

$$\left|D_j X\right|_{(2)} \le \left(1 - \left(\frac{1}{4}\right)^2\right)^{j-1} \cdot \|X\|_1$$

для всех узлов второго типа;

.....

$$\left|D_j X\right|_{(j-1)} \le \left(1 - \left(\frac{1}{4}\right)^{j-1}\right)^2 \cdot \|X\|_1$$

для всех узлов $(j-1)$-го типа;

$$\left|D_j X\right|_{(j)} \le \left(1 - \left(\frac{1}{4}\right)^j\right) \cdot \|X\|_1$$

для всех узлов j-го типа.

Таким образом,

$$\left\|D_j X\right\|_1 \le \max\left\{\left(\frac{3}{4}\right)^j, \left(\frac{15}{16}\right)^{j-1}, \ldots, 1 - \left(\frac{1}{4}\right)^j\right\} \cdot \|X\|_1.$$

Последнее условие позволяет оценить норму матрицы D_j, $\left\|D_j\right\|_1 \le 1 - \left(\frac{1}{4}\right)^j$.

Поскольку всего типов внутренних узлов m, постольку важной представляется оценка нормы матрицы $\|D_m\|_1 \le 1 - \left(\frac{1}{4}\right)^m = q_m < 1$.

Оказывается, что матрица $D_m = D^m$ является сжимающей матрицей (сжимающим оператором), то есть матрицей с нормой строго меньшей единицы. Тогда исходный процесс построения последовательных приближений в соответствии с равенством (7) можно переформулировать, а именно, заменить равенство (7) на равенство

$$X^{(nm)} = D_m X^{m(n-1)} - D_{m-1} g - D_{m-2} g - \ldots - D_1 g - g$$

или, обозначив $g1 = D_{m-1} g - D_{m-2} g - \ldots - D_1 g - g$, получим ту же самую последовательность приближений, удовлетворяющую рекуррентным соответствиям

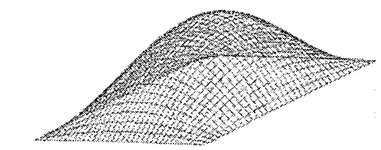
$$X^{(nm)} = D_m X^{m(n-1)} - g1 \qquad (9)$$

при $n = 1,2\ldots$.

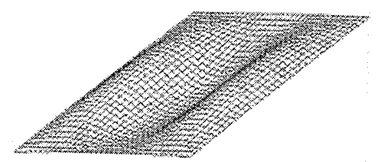
Сходимость процесса, удовлетворяющего формуле (9), гарантируется достаточным условием сходимости $\|D_m\| \le q_m < 1$.

Сделаем несколько очевидных выводов относительно скорости сходимости в предложенном методе простых итераций. Ясно, что норма $\|D_m\|$ зависит от m. Мы установили оценку $\|D_m\| \le q_m$, где

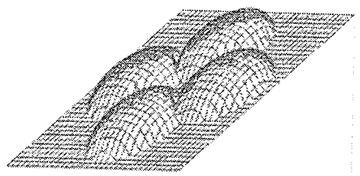
$$q_m = 1 - \left(\frac{1}{4}\right)^m < 1.$$



$$f(x,y) = 4 \cdot \sin\left(\frac{\pi \cdot x}{n}\right) \cdot \sin\left(\frac{\pi \cdot y}{n}\right)$$

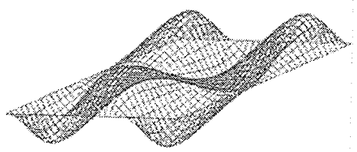


$$f(x,y) = \sin\left(\frac{\pi \cdot x}{n}\right)$$



$$f(x,y) = \sqrt{0{,}04 - \left(\frac{x}{n}\right)^2 - \left(\frac{y}{n} + 0{,}25\right)^2} + \sqrt{0{,}04 - \left(\frac{x}{n}\right)^2 - \left(\frac{y}{n} - 0{,}25\right)^2} +$$

$$+ \sqrt{0{,}04 - \left(\frac{x}{n} + 0{,}25\right)^2 - \left(\frac{y}{n}\right)^2} + \sqrt{0{,}04 - \left(\frac{x}{n} - 0{,}25\right)^2 - \left(\frac{y}{n}\right)^2}$$



$$f(x,y) = 3 \cdot \cos\left(\frac{\pi \cdot x}{n}\right) \cdot \sin\left(\frac{3 \cdot \pi \cdot y}{n}\right)$$

Рис. 3

$$f(x,y) = \sqrt{0{,}5 - \left(\frac{x}{n}\right)^2 - \left(\frac{y}{n}\right)^2} \qquad f(x,y) = \sin\left(\frac{2{,}5\cdot\pi\cdot x}{n}\right)\cdot\sin\left(\frac{\pi\cdot y}{1{,}2\cdot n}\right) + 0{,}2$$

Рис. 4

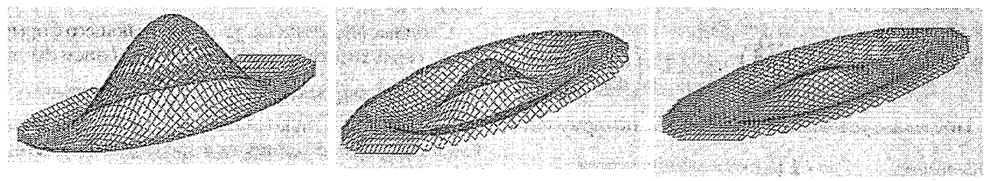


Рис. 5

При большом значении числа m величина q_m приближается к 1. Это говорит о том, что чем выше порядок системы (9), то есть чем меньше величина шага разбиения h, тем ближе q_m к единице. Это значит, что при большом количестве внутренних узлов скорость сходимости процесса приближений, оставаясь геометрической, уменьшается.

Примеры

По приведенному ранее плану найдем положения равновесия для конкретных мембран, то есть решим задачу Дирихле рассмотренным ранее методом конечных разностей, для конкретных случаев выбора области G и плотности силы $f(x, y)$. Во всех рассматриваемых далее примерах в качестве граничных условий была выбрана нулевая функция, то есть мембрана жестко закреплена на границе L.

Случай, когда область G-квадрат $\{0 \le x \le 1, 0 \le y \le 1\}$, при различном выборе внешней силы $f(x, y)$ приводит к следующим результатам (рис. 3).

В рассмотренных примерах на рисунке (рис. 3) исходное дифференциальное уравнение аппроксимировалось разностями на сетке с шагом $h = \frac{1}{35}$. При этом вычислительная погрешность каждого из полученных решений не более 10^{-2}.

Интересен случай, когда в качестве области G выбран круг $x^2 + y^2 \le 1$. При решении задачи Дирихле на круге область G дискретизировалась равномерной сеткой с шагом $h = \frac{1}{45}$. Учет граничных условий производился рассмотренным ранее способом простого сноса, погрешность которого $O(h)$. Общая погрешность решения не превосходит 10^{-1}. Результаты изображены на рисунке (рис. 4).

Хорошую наглядность имеет случай, когда в качестве внешней силы выбрана функции Бесселя первого рода нулевого порядка J_0 или ее модификации (рис. 5).

Рисунки поверхностей (положение мембраны определенной формы в пространстве) были получены с помощью трехмерной графики. На занятиях по компьютерному моделированию от студентов требуется построение такого рода рисунков более упрощенным способом, точнее, изображением только некоторых сечений поверхности плоскостями, перпендикулярными осям Ox и Oy. При этом не требуется не изображать невидимые линии.

Примечания

1. *Петровский И. Г.* Лекции об уравнениях с частными производными. М.: ГФМЛ, 1951.

2. *Тихонов А. Н., Самарский А. А.* Уравнения математической физики. М.: Наука, 1972.

3. *Березин И. С., Жидков И. П.* Методы вычислений. Т. II. М.: ГФМЛ, 1960.

Е. В. Котельников

ТЕОРИЯ АВТОМАТОВ В COMPUTER SCIENCE

В статье рассматриваются связи классической теории автоматов с другими курсами, входящими в систему подготовки специалистов по информатике.

Основные понятия. Теория конечных автоматов изучает простейшую модель вычислительного устройства – автомат, определяемый как преобразователь информации, выходная реакция которого зависит не только от входных воздействий в данный момент, но и от входной истории [1, 2]. Хотя теория автоматов изучает очень простые модели, она, как будет показано ниже, является фундаментом большого числа разнообразных приложений.

Перед изучением роли автоматов в информатике необходимо уяснить основные понятия теории автоматов.

Конечный автомат имеет множество входных $Z=\{z_1, z_2,\ldots, z_p\}$ и множество выходных сигналов $W=\{w_1, w_2,\ldots, w_m\}$ (рис. 1). Он работает в дискретном режиме и в каждый момент времени находится в некотором состоянии из конечного множества состояний $S=\{s_1, s_2,\ldots, s_n\}$. При поступлении сигнала на вход автомат выдает выходной сигнал и переходит в другое состояние либо остается в предыдущем, причем и выходной сигнал, и следующее состояние зависят как от входного сигнала, так и от состояния, в котором находился автомат.

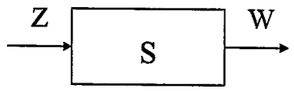


Рис. 1. Конечный автомат

Существует несколько видов конечных автоматов, самые распространенные из них – автоматы Мили и Мура, названные так по имени впервые исследовавших эти модели американских ученых G. H. Mealy и E. F. Moore. Два этих вида равнозначны по вычислительной мощности, и легко можно построить эквивалентный автомату Мура автомат Мили и наоборот. Автомат Мили соответствует классическому определению автомата, то есть выход зависит от входа и от предыдущего состояния, в то время как выход автомата Мура определяется однозначно тем состоянием, в которое автомат переходит после приема входного сигнала.

КОТЕЛЬНИКОВ Евгений Вячеславович – ассистент кафедры информатики и МОИ ВятГГУ

3 Зак. 2868

Работа автомата может быть описана двумя способами: таблицами переходов и выходов или графами переходов. Пример описания автомата Мили первым способом представлен в табл. 1 и 2.

Таблица 1

Таблица переходов автомата Мили

	a_1	a_2	a_3
z_1	a_2	a_3	a_2
z_2	a_3	a_2	a_1

Таблица 2

Таблица выходов автомата Мили

	a_1	a_2	a_3
z_1	w_1	w_3	w_3
z_2	w_2	w_1	w_1

Строки этих таблиц соответствуют входам, а столбцы – состояниям автомата. На пересечении столбца a_i и строки z_j в таблице переходов ставится состояние a_k, в которое автомат приходит из состояния a_i под воздействием сигнала z_j, а в таблице переходов – соответствующий этому переходу выходной сигнал w_k.

Автомат Мура описывается одной таблицей переходов, в которой каждому столбцу приписан кроме состояния ещё и выходной сигнал.

Задание автоматов графами переходов имеет преимущество большей наглядности и понятности. Граф переходов автомата Мили, соответствующий табл. 1 и 2, показан на рис. 2.

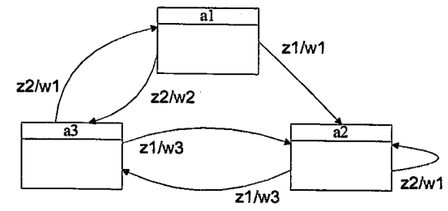


Рис. 2. Граф переходов автомата

Граф автомата – ориентированный граф, вершины которого соответствуют состояниям автомата, а дуги – переходам между ними. В вершинах указывается обозначение состояния, а для автоматов Мура – ещё и выходной сигнал в данном состоянии. На дугах для автоматов Мили отмечается выходной сигнал и для обоих видов – входной, при котором происходит данный переход.

После краткого изложения основ перейдем непосредственно к вопросу о связи автоматов с Computer Science, предварительно рассказав об ученых, положивших начало данному направлению в науке.

История развития теории автоматов. Раньше всего в теории автоматов получило развитие одно из направлений прикладной теории автоматов – комбинационный синтез релейно-переключательных схем. Ещё в 1910 году петербургский физик П. С. Эренфест указал на возможность использования алгебры логики при построении релейных схем [2]. Но началом становления теории конечных автоматов можно считать работы В. И. Шестакова и К. Шеннона (C. E. Shannon), в 1938 году применивших аппарат булевой алгебры к синтезу таких схем [3]. В 40-50-х годах теория прикладных автоматов была расширена за счет исследований в областях структурного и блочного синтеза автоматов (работы Д. Хафмена, Г. Мили, В. И. Шестакова, М. А. Гаврилова, М. Уилкса) [4].

В конце 40-х годов над абстрактной теорией автоматов начинает работать знаменитый математик Джон фон Нейман (John von Neumann). Его общая математическая (логическая) теория автоматов создавалась вместе с кибернетикой Норберта Винера, сферой теории автоматов была дискретная математика, а сферой кибернетики – непрерывная [6]. В работах фон Неймана были разработаны такие важные области теории автоматов, как надежностный синтез и самовоспроизведение автоматов. Многие из его идей до сих пор не получили должного развития.

Труды фон Неймана, а также других значительных ученых – С. К. Клини, Э. Мура, К. Шеннона – были опубликованы в известном сборнике статей «Автоматы» под редакцией К. Шеннона и Дж. Маккарти в 1956 году [6]. Этот сборник являлся первым в своем роде, он положил начало абстрактной теории автоматов и задал направление исследований в этой сфере на несколько лет вперед.

В СССР с начала 50-х годов, с появлением новой области науки – кибернетики, довольно бурное развитие получила и теория автоматов. Среди направлений стратегических исследований в области информатики советские ученые отмечали и теорию автоматов. Так, в статье В. М. Глушкова, работавшего тогда в Институте математики АН УССР, отмечается, что основой прогресса развития вычислительных машин наряду с теорией их работы, разработкой методов автоматизации проектирования ЭВМ и программирования, теорией алгоритмов и вычислительной математики должна стать и теория конечных детерминированных и стохастических автоматов [7].

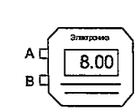
С середины 50-х годов начал работать семинар по теории автоматов на физическом факультете МГУ, руководителем которого был М. Л. Цетлин. Чуть позже в Киеве стал действовать семинар под руководством В. М. Глушкова. На этих семинарах обсуждались вопросы коллективного поведения автоматов, логического синтеза и синтеза схем, связь теории автоматов с проблемами машинного моделирования и моделями поведения живых систем. Поэтому в семинарах принимали участие не только физики и математики, но и физиологи, медики, биологи и философы. Оба ученых стали основоположниками научных школ в Москве и Киеве [8].

Еще одним замечательным ученым, работавшим в области теории автоматов и релейных устройств, был М. А. Гаврилов. В середине 50-х годов он организовал семинар в Институте автоматики и телемеханики АН СССР, а с 1964 года начали проводиться знаменитые Гавриловские школы. Их уровень был одним из самых высоких в мире: участники этих школ в 70-х годах первыми начали создавать системы автоматического проектирования дискретных систем управления [9].

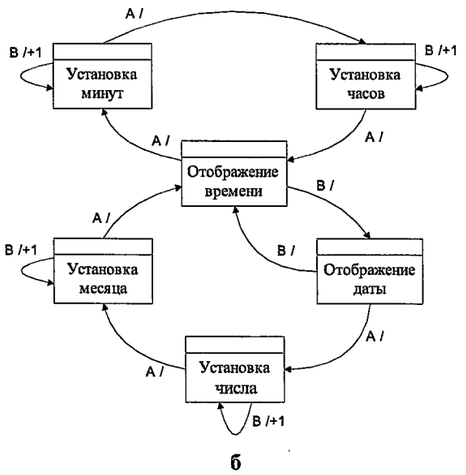
В рамках этих и других школ по теории автоматов и её приложениям работали такие ученые, как М. А. Айзерман, С. И. Баранов, В. И. Варшавский, А. Д. Закревский, В. Н. Касьянов, В. Г. Лазарев, Д. А. Поспелов, Ю. В. Поттосин, И. В. Прангишвили, А. А. Таль, Б. А. Трахтенброт и многие другие.

Благодаря этим школам и исследователям сегодня мы имеем достаточно разработанную теорию автоматов, которая, как будет показано ниже, находит все новые и новые приложения.

Синтез цифровых устройств. Одним из традиционных приложений теории автоматов является синтез цифровых устройств. Например, теория автоматов широко использовалась «при создании отечественной вычислительной техники, некоторые образцы



а



б

Рис. 3:
а – электронные часы;
б – граф переходов управляющего автомата

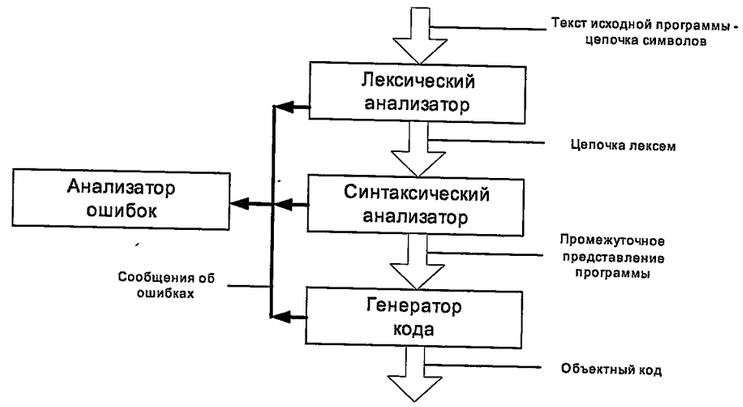


Рис. 4. Общая структурная схема транслятора

которой (например, машины М-10 и БЭСМ-6) не уступали, а во многом и превосходили зарубежные аналоги» [10].

В 90-х годах зарубежные фирмы (фирма «Модикон», США, входящая в группу AEG Schneider Electric; фирма «Сименс», Германия; фирма «General Electric Fanuc Automation», США) стали использовать в качестве языка программирования своих программируемых логических контроллеров графы переходов [11].

Для иллюстрации применения теории автоматов в этом направлении приведем граф переходов схемы управления обычных электронных часов [12]. Все функции часов реализуются конечным автоматом, который управляется кнопками на корпусе часов. Пусть для простоты автомат выполняет две функции: отображение даты и времени и их установку. Часы с двумя управляющими кнопками A и B приведены на рис. 3а, граф переходов автомата – на рис. 3б.

Начальное состояние для данного автомата – состояние «Отображение времени». При нажатии кнопки A автомат переходит в состояние «Установка минут». Если сейчас нажать кнопку B, то к минутам прибавится единица, а состояние останется прежним. Нажатие кнопки A переводит автомат в состояние «Установка часов» и т. д.

Трансляция языков программирования. Другим традиционным применением теории автоматов является построение трансляторов для языков программирования – одна из самых важных областей системного программирования.

Известно, что транслятор (компилятор или интерпретатор) состоит из следующих основных частей: лексического анализатора, синтаксического анализатора, генератора кода (в интерпретаторе вместо генератора кода используется эмулятор) и анализатора ошибок (рис. 4) [13].

Лексический анализатор преобразует цепочку входных символов в цепочку лексем (элементарные конструкции языка, например, идентификатор, действительное число, комментарий). Синтаксический анализатор осуществляет распознавание цепочки лексем, семантический анализ и построение промежуточного представления. Генератор кода на основе промежуточного представления и системы команд конкретного компьютера вырабатывает объектный код. Анализатор ошибок, обрабатывая сообщения блоков транслятора об ошибках, выдает предупреждения пользователю, иногда пытаясь исправить ошибку самостоятельно.

Важнейшими частями лексического и синтаксического анализаторов являются распознаватели – алгоритмы, определяющие некоторое множество. Распознаватели реализуются автоматами. Рассмотрим структуру лексического анализатора (рис. 5).

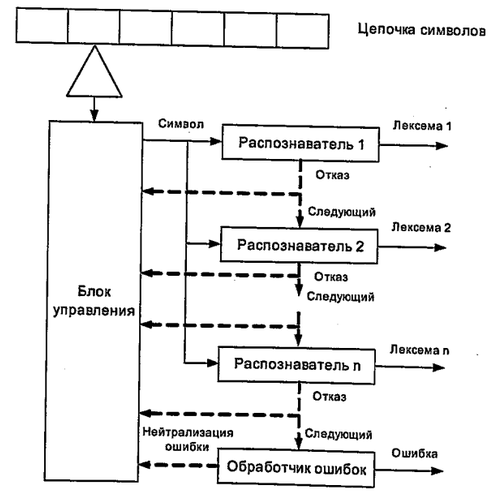


Рис. 5. Лексический анализатор

Он включает в себя блок управления, n распознавателей – конечных автоматов и обработчик ошибок. Каждый конечный автомат распознает отдельную лексему. Автоматы соединены последовательно, поэтому в случае неудачи распознавания одним автоматом символ передается следующему.

На рис. 6 изображен конечный автомат – распознаватель множества целых констант [14].

Допустимые символы в начальном состоянии: '+', '–', '0..9'. Если приходит другой символ («Иначе»), то автомат переходит в так называемое «незаключительное» состояние, сигнализирующее о том, что лексема не относится к целым константам. Если после анализа очередной порции символов автомат окажется в «заключительном» состоянии, значит, лексема распознана и является целым числом.

Синтаксический анализ выполняется с помощью более сложных методов, поэтому возможностей обычных конечных автоматов недостаточно и используют автоматы с магазинной памятью (память стекового типа).

Системы связи. Уже довольно давно автоматы применяются в системах телекоммуникаций, в частности для спецификации протоколов компьютерных сетей. В этих спецификациях используются диаграммы Харела, основанные на графах переходов.

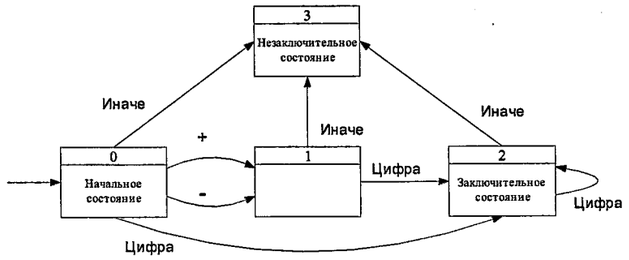


Рис. 6. Конечный автомат – распознаватель целых констант

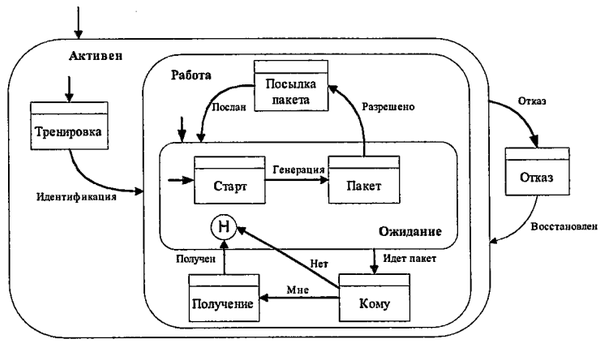


Рис. 7. Спецификация доступа в протоколе 100VG-AnyLAN

Дэвид Харел в 1987 году предложил свои диаграммы (названные им «State charts» – «карты состояний») для формального описания сложных систем [15]. Эти диаграммы расширяют возможности графов переходов конечных автоматов за счет некоторых улучшений. Например, наряду с обычными состояниями могут использоваться гиперсостояния (или суперсостояния), объединяющие несколько состояний, имеющих идентичную реакцию на одно и то же событие. С состояниями могут быть ассоциированы действия: при входе в состояние (в соответствующей вершине помечается словом ***entry***), при выходе из него (помечается словом ***exit***) и при нахождении в состоянии (помечается словом ***do***). Также введено историческое псевдосостояние (обозначается буквой ***H***), которое не является реальным состоянием и служит для запоминания вершины в гиперсостоянии, в которой система находилась в последний раз.

Приведем в качестве примера спецификацию процесса доступа к среде протокола IEEE 802.12, описывающего высокоскоростную технологию локальной сети 100VG-AnyLAN (рис. 7) [16].

Начальное состояние процесса доступа – «Тренировка» гиперсостояния «Активен». В нем данная рабочая станция идентифицируется, проверяются канал и верхний уровень. При отсутствии ошибок процесс переходит в начальное состояние «Старт» гиперсостояния «Ожидание». Если произошла генерация пользователем очередного пакета, то процесс в состоянии «Пакет» ждет разрешения посылки пакета в канал связи и при получении разрешения отправляет пакет (состояние «Посылка пакета» гиперсостояния «Работа»). Затем процесс возвращается в состояние «Старт». В любом из состояний гиперсостояния «Ожидание» процесс может быть прерван сигналом от верхнего уровня «Идет пакет», который заставляет все станции сети готовиться к приему пакета. Верхний уровень по полученному заголовку пакета определяет его адресата и сообщает процессу, надо ли принимать пакет или следует вернуться в предыдущее состояние гиперсостояния «Ожидание» (за это отвечает историческое псевдосостояние ***H***). В случае отказа процесс переходит в состояние «Отказ», а при восстановлении возобновляет работу из состояния «Тренировка».

Теория автоматов применяется не только в компьютерных, но и в других видах сетей связи. В начале

1990-х годов Международный консультативный комитет по телеграфии и телефонии (CCITT, Comité Consultatif Internationale de Télégraphie et Téléphonie) разработал язык SDL (Specification and Description Language, язык спецификаций и описаний), предназначенный для описания структуры и функционирования распределенных систем реального времени, в частности сетей связи [17]. SDL-описание представляет поведение системы как цепочку ответов на произвольную цепочку стимулов. Основным элементом такой системы является процесс – обобщение понятия «конечный автомат». Процесс описывается с помощью таблиц переходов или SDL-диаграмм, являющихся разновидностью схем алгоритмов, в которые в явном виде могут вводиться состояния. Методология SDL применяется при создании программного обеспечения телекоммуникационных систем.

Объектно-ориентированное программирование. От традиционных сфер применения теории автоматов перейдем к рассмотрению относительно новых направлений современного программирования, где конечные автоматы успели занять достойное место.

Одним из таких направлений является объектно-ориентированное программирование (ООП). Появление и развитие этой парадигмы относится к 1970-80-м годам, однако методология проектирования программ в рамках ООП была разработана Гради Бучем только в 1991 году [18]. Для описания поведения проектируемой системы и её архитектуры он предложил использовать диаграммы классов, объектов, модулей, процессов, взаимодействий и состояний и переходов. Последний вид диаграмм является применением диаграмм Харела к объектной модели.

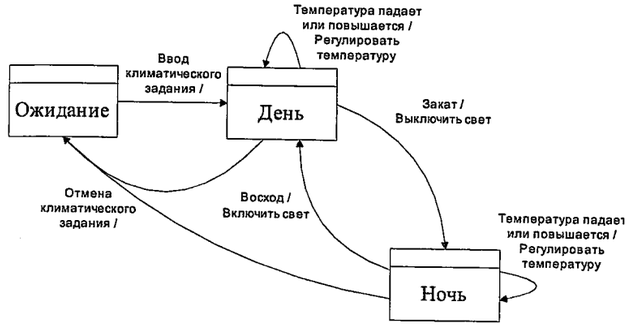


Рис. 8. Диаграмма состояний и переходов контроллера теплицы

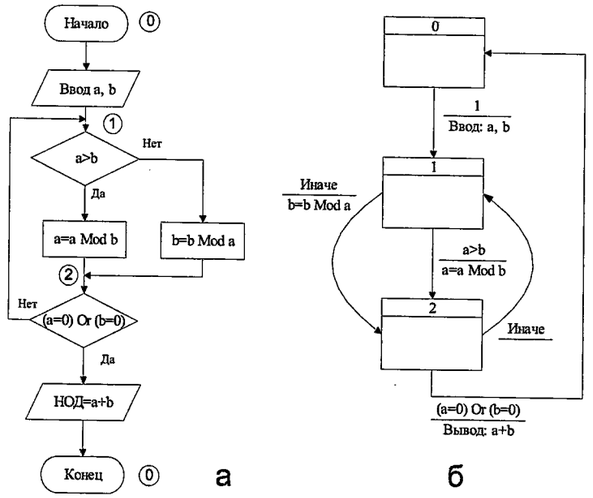


Рис. 9: **а** – граф-схема алгоритма Евклида;
б – граф переходов автомата Мили

Диаграмма состояний и переходов в методологии Г. Буча «показывает:

1) пространство состояний экземпляров данного класса;

2) события, которые влекут переход из одного состояния в другое;

3) действия, которые происходят при изменении состояния» [19].

В качестве примера на рис. 8 приведена диаграмма состояний и переходов для класса, описывающего контроллер тепличной среды [20].

Все объекты данного класса начинают существование в состоянии «Ожидание». По событию «Ввод климатического задания» объект переходит в состояние «День» (предполагается, что задание вводится днем) без выполнения действий. Затем следуют переключения между состояниями «День» и «Ночь», определяющиеся событиями «Восход» и «Закат». Переключения сопровождаются действиями по изменению освещения. Отклонение температуры от заданной приведет к вызову метода «Регулировать температуру», который является локальным для данного класса. По событию «Отмена климатического задания» происходит переход в состояние «Ожидание».

Следует отметить, что понятие «конечный автомат» и диаграммы состояний, основанные на диаграммах Харела, входят также в стандарт

языка UML (Unified Modeling Language – унифицированный язык моделирования). Этот язык был недавно (руководство пользователя опубликовано в 1998 году) создан Г. Бучем (G. Booch), Д. Рамбо (J. Rumbaugh) и А. Джекобсоном (I. Jacobson) как «коллекция наилучших инженерных нотаций, применяемых при объектном моделировании сложных систем общего назначения» [21].

Автоматное программирование. В другом молодом направлении программирования автоматы имеют основополагающее значение. Этим направлением является автоматное программирование (или SWITCH-технология), созданное А. А. Шалыто в 1991 году [22].

Назовем основные положения данной парадигмы:

• алгоритмы реализуются в виде конечного автомата, причем понятие «состояние» является ключевым;

• в качестве спецификации программного продукта используются графы переходов;

• автоматная программа строится по графу переходов формально и изоморфно, поэтому отладка, тестирование, сопровождение и изменение программы производится по графам переходов;

• при необходимости автомат может быть разложен на систему взаимосвязанных автоматов, взаимодействующих между собой по принципу вызываемости или вложенности;

• каждый автомат реализуется конструкцией SWITCH языка Си (в Паскале – CASE) или любой аналогичной, которая является телом цикла с постусловием.

А. А. Шалыто предложил метод построения по граф-схеме алгоритма графа переходов, а затем и автоматной программы [23]. Данный метод основан на способе преобразования граф-схем в графы переходов, представленном в [24].

Приведем пример построения автоматной программы по алгоритму Евклида для нахождения наибольшего общего делителя. Граф-схема алгоритма показана на рис. 9а.

В заданную схему в зависимости от выбранного для ее замены типа автомата (Мура или Мили) вводятся пометки, соответствующие номерам его состояний. При этом для автоматов обоих типов начальной и конечной вершинам схемы присваивается номер 0, что обеспечивает построение «замкнутого» графа переходов. При построении автомата Мура n-й группе соединенных последовательно операторных вершин (она может состоять также и из одной вершины) присваивается номер n. При построении автомата Мили соответствующий номер присваивается точке, следующей за последней из последовательно соединенных операторных вершин группы, причем указанные точки для различных групп могут совпадать. Это приводит к тому, что автомат Мили имеет число состояний, не превышающее их число в эквивалентном автомате Мура.

Номера, применяемые для построения автомата Мили, указаны на схеме алгоритма в кружках.

Граф переходов автомата Мили, соответствующий этой граф-схеме, приведен на рис. 9б.

Используя этот граф переходов, построим автоматную программу (листинг 1).

Листинг 1

```
Program Evklid_Automat; {Автоматная реали-
                         зация алгоритма Евклида}
Var a,b,y:Integer;       {Автомат Мили}
Begin
 y:=0;                   {Переменная y обозна-
                         чает номер состояния}
 Repeat
  Case y of
   0: Begin
       ReadLn(a,b);
       y:=1;
      End;
   1: If a>b Then Begin a:=a Mod b;
                        y:=2;   End
              Else Begin b:=b Mod a;
                        y:=2; End;
   2: If (a=0) Or (b=0) Then Begin
                      WriteLn("HOD=",a+b);
                        y:=0; End
                 Else y:=1;
  End;
 Until y=0;
End.
```

Прочие применения теории автоматов. Кроме перечисленных областей автоматы используются и в других сферах информатики.

В теории алгоритмов само понятие «алгоритм» часто определяется с помощью машины Тьюринга. В 1936 году Алан Тьюринг предложил гипотетическое устройство (которое сейчас называется машиной Тьюринга), представляющее собой формальную модель вычислителя, и определил алгоритм как программу для этой машины [25]. По сути, машина Тьюринга является результатом добавления к обычному конечному автомату потенциально бесконечной памяти. Оказывается, такое простое расширение обеспечивает существенное изменение возможностей применения машины Тьюринга по сравнению с конечными автоматами.

В операционной системе UNIX теория автоматов нашла свое применение в системных программах, таких, как протокол выбора лидера в распределенной системе процессоров, а также для поиска текста по образцу в текстовом редакторе и для построения драйверов [26].

В работе [27] автоматный подход реализован для одного из алгоритмов поиска подстрок.

Работы М. Л. Цетлина по проблемам целесообразного поведения автоматов, начатые в 60-х годах, привели к изучению коллективного и адаптивного поведения автоматов, а затем и к использованию теории автоматов в биологии, социологии, теории игр и в области искусственного интеллекта [28].

Конечноавтоматная модель применяется и для описания параллельных взаимодействующих процессов, в частности для координации их управления [29].

В [30] указывается, что автоматы используются для построения и программирования систем реального времени.

В заключение этого раздела можно привести высказывание Herve Gallaire, одного из участников семинара «Software 2000: a View of the Future», проходившего в 1994 году: «Я знаю людей из "Боинга", занимающихся системами стабилизации самолетов с использованием чистой теории автоматов. Даже трудно себе представить, что им удалось сделать с помощью этой теории» [31].

Вывод. Итак, можно сделать вывод о необходимости глубокого знания теории автоматов для специалиста в Computer Science. Причем такой вывод следует не только на основании широты применения данной теории в традиционных и новых областях информатики, но и с учетом фундаментального характера понятия «состояние». Это понятие, базовое в теории автоматов, естественным образом может использоваться для описания процессов самой различной природы. Поэтому есть все основания полагать, что теория автоматов найдет свое применение не только в уже известных сферах человеческой деятельности, но и в ещё не открытых.

Примечания

1. *Карпов Ю. Г.* Теория автоматов. СПб.: Питер, 2002; *Баранов С. И.* Синтез микропрограммных автоматов (граф-схемы и автоматы). Л.: Энергия, 1979.
2. Теория дискретных управляющих устройств / Под ред. А. Д. Закревского, И. В. Прангишвили. М.: Наука, 1982.
3. *Шалыто А. А.* У нас была Великая эпоха! // www.softcraft.ru.
4. *Глушков В. М.* Синтез цифровых автоматов. М.: Физматгиз, 1962.
5. *Нейман фон Дж.* Теория самовоспроизводящихся автоматов. М., 1971.
6. Автоматы: Сборник статей / Под ред. К. Шеннона, Дж. Маккарти. М.: ИЛ, 1956.
7. Очерки истории информатики в России / Ред.-сост.: Д. А. Поспелов, Я. И. Фет. Новосибирск: НИЦ ОИГГМ СО РАН, 1998.
8. Там же.
9. Теория дискретных управляющих устройств...; *Шалыто А.А.* Указ. соч.; Очерки истории информатики... .
10. *Шалыто А. А.* Указ. соч.
11. *Шалыто А.А.* Алгоритмизация и программирование для систем логического управления и «реактивных» систем // Автоматика и телемеханика. 2001. № 1. С. 3-39.
12. *Карпов Ю. Г.* Указ. соч.
13. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 1. Синтаксический анализ. М.: Мир, 1978; *Льюис Ф., Розенкранц Д., Стирнз Р.* Теоретические основы проектирования компиляторов. М.: Мир, 1979; *Легалов А. И.* Разработка трансляторов // www.softcraft.ru.
14. *Карпов Ю. Г.* Указ. соч.
15. Там же; *Шалыто А. А.* Алгоритмизация и программирование...; *Шалыто А. А., Туккель Н. И.* Программирование с явным выделением состояний // Мир ПК. 2001. № 8. С. 116-121; *Harel D.* Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming. 1987. Vol. 8. P. 231-274.
16. *Карпов Ю. Г.* Указ. соч.
17. *Шалыто А. А.* Алгоритмизация и программирование...; *Шалыто А. А., Туккель Н. И.* Указ. соч.; *Карабегов А. В., Тер-Микаэлян Т. М.* Введение в язык SDL. М.: Радио и связь, 1993.
18. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином; СПб.: Невский диалект, 1998.
19. Там же.
20. Там же.
21. *Шалыто А. А.* Алгоритмизация и программирование...; *Booch G., Rumbaugh J., Jacobson I.* The Unified Modeling Language. User guide. MA: Addison-Wesley, 1998. (*Буч Г., Рамбо Д., Джекобсон А.* Язык UML. Руководство пользователя. М.: ДМК, 2000); *Рамбо Д., Джекобсон А., Буч Г.* UML. Специальный справочник. СПб.: Питер, 2002.
22. *Шалыто А. А.* Алгоритмизация и программирование...; *Шалыто А. А., Туккель Н. И.* Указ. соч.; *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
23. *Шалыто А.А., Туккель Н.И.* Реализация вычислительных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2001. № 6.
24. Теория дискретных управляющих устройств...
25. *Карпов Ю. Г.* Указ. соч.
26. Там же.
27. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2000.
28. Очерки истории информатики...; *Стефанюк В. Л.* От автоматов М. Л. Цетлина к искусственному интеллекту // Новости искусственного интеллекта. 1995. № 4; *Цетлин М. Л.* Исследования по теории автоматов и моделированию биологических систем. М.: Наука, 1969.
29. *Карпов Ю. Г.* Указ. соч.; *Дейкстра Э.* Взаимодействие последовательных процессов // Языки программирования / Ред. Ф. Женюи. М.: Мир, 1972. С. 9-86.
30. *Карпов Ю. Г.* Указ. соч.; *Шалыто А. А.* Алгоритмизация и программирование...
31. *Карпов Ю. Г.* Указ. соч.

Ю.В. Икрина

О ПРОЦЕССОРЕ БАЗ ДАННЫХ СРЕДЫ ПРОГРАММИРОВАНИЯ DELPHI

У меня есть голова. Она заставляет меня одному верить, другому нет, а сказанное третьим проверять фактами.

Потоки информации, циркулирующие в мире, который нас окружает, огромны. Во времени они имеют тенденцию к увеличению. Поэтому в любой организации, как большой, так и маленькой, возникает проблема такой организации управления данными, которая обеспечила бы наиболее эффективную работу.

С появлением ЭВМ и использованием их для обработки информации появилась возможность автоматизировать решение многих информационно-справочных и расчетных задач. Эти идеи нашли свое воплощение в системах управления базами данных (СУБД).

Роль СУБД как единого средства хранения, обработки и доступа к большим объемам информации постоянно возрастает. При этом существенным является постоянное повышение объемов информации, хранимой в БД, что влечет за собой требование увеличения производительности таких систем.

На сегодняшний день практически все языки программирования имеют расширения для работы с базами данных наиболее распространённых форматов.

На рынке программных средств появляются всё более сложные и совершенные разработки, которые имеют универсальное применение. Реклама производителей зачастую представляет предлагаемые новшества как наилучший вариант решения проблем. Но только является ли этот вариант на самом деле наилучшим? Перед программистом встаёт сложный вопрос выбора средства разработки.

Одним из таких средств является среда программирования Delphi.

Среда программирования Delphi предоставляет все необходимые инструменты для создания приложений, работающих с базой данных.

Одним из достоинств программирования баз данных в Delphi является использование открытой архитектуры доступа к данным. Процессор баз данных Borland Database Engine (BDE), в основу которого положена технология интегрированной базы данных (Integrated DataBase API (IDAPI)), предоставляет возможность унифицированного подключения как к локальным, так и к удаленным базам данных.

Особенность архитектуры BDE обеспечивает ряд существенных преимуществ.

1. Реальное разделение программного кода и механизм доступа к данным. Причем сам доступ также осуществляется на нескольких уровнях-BDE, драйвера, сервера БД. Приложение Delphi для работы с БД можно настроить на использование с любой СУБД, для которой имеется соответствующий драйвер, буквально за несколько минут. При этом перекомпиляция самой программы не требуется. Плата за такую великолепную переносимость – скорость обмена данными через BDE и драйверы несколько меньше, чем напрямую между приложением и СУБД.

2. Разделение драйверов и выделение в специальную группу драйверов для серверов SQL позволило гораздо полнее использовать функциональные возможности серверов БД, а применение единого API сняло остроту проблемы интерпретации процесса выполнения транзакций разными серверами.

3. BDE идеально подходит для создания приложений для архитектуры клиент/сервер, так как обеспечивает следующие функции:

- обработка запросов в диалоговом режиме;
- представление через BDE вызовов операционной системы;
- модифицируемые запросы;
- поддержка хранимых процедур;
- управление регистрацией пользователя.

4. Все инструментальные средства баз данных Borland – Paradox, dBase, Database Desktop – используют BDE. Все особенности, имеющиеся в Paradox или dBase, «наследуются» BDE, и поэтому этими же особенностями обладает и Delphi.

5. Библиотека объектов содержит набор визуальных компонент, значительно упрощающих разработку приложений для СУБД с архитектурой клиент-сервер. Объекты инкапсулируют в себя нижний уровень – Borland Database Engine.

6. Предусмотрены специальные наборы компонент, отвечающих за доступ к данным, и компонент, отображающих данные. Компоненты доступа к данным позволяют осуществлять соединения с БД, производить выборку, копирование данных и т. п. Компоненты визуализации данных позволяют отображать данные в виде таблиц, полей, списков. Отображаемые данные могут быть текстового, графического или произвольного формата.

7. Таблицы сохраняются в базе данных. Некоторые СУБД сохраняют базу данных в виде нескольких отдельных файлов, представляющих собой таблицы (в основном все локальные СУБД), в то время как другие состоят из одного файла, который содержит в себе все таблицы и индексы (InterBase). Например, таблицы dBase и Paradox всегда сохраняются в отдельных файлах на диске. Директорий, содержащий dBase.DBF файлы или Paradox .DB файлы, рассматривается как база данных. Другими словами, любой директорий, содержащий файлы в формате Paradox

ИКРИНА Юлия Вадимовна – ассистент кафедры информатики и МОИ ВятГГУ

или dBase, рассматривается Delphi как единая база данных. Для переключения на другую базу данных нужно просто переключиться на другой директорий. InterBase сохраняет все таблицы в одном файле, имеющем расширение .GDB, поэтому этот файл и есть база данных InterBase.

Доступ к данным

Основой архитектуры доступа к наборам данных (рис. 1) является базовый класс TDataSet, который содержит абстрактное представление записей и полей набора данных, инкапсулирующий управление, навигацию и манипулирование набором данных. Некоторые методы класса TDataSet могут быть переопределены с целью создания компонента, подключаемого к определенному физическому формату данных. Исходя из этого, класс TBDEDataSet определен как производный от класса TDataSet и является основным классом источников данных, он вводит такие концепции, как BDE-базы данных и сеансы.

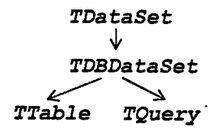


Рис. 1

Класс TTable представляет структуру и данные, содержащиеся в таблице базы данных, знает, как обрабатывать индексы и применять специальные приемы, связанные с поддержкой отношений двух таблиц типа один-ко-многим. Класс TQuery – набор дан-

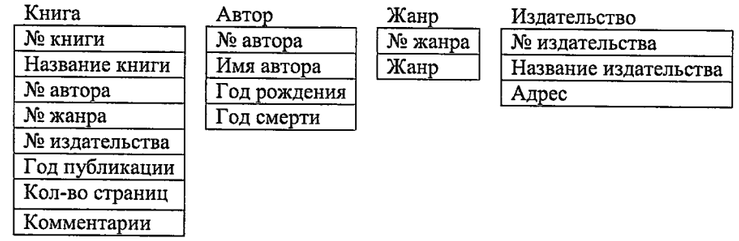


Рис. 2

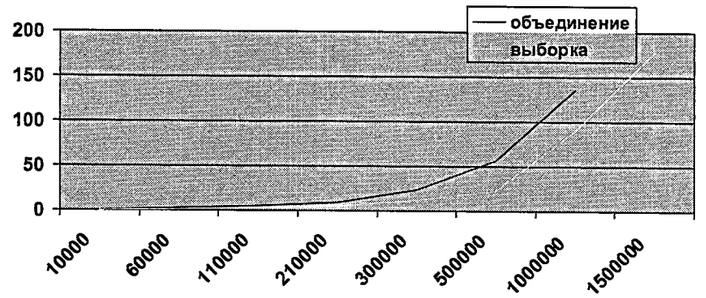


Рис. 3

Количество записей в таблицах	10000	60000	110000	210000	300000	500000	1000000	1500000
Время работы запроса на выборку данных (сек)	0,18	1,35	2,6	5,3	11,8	22,11	90,11	177,52
Время работы запроса на объединение данных (сек)	0,52	2,37	4,74	9,6	24	56,3	135,14	Запрос не выполняется

ных, содержащий информацию, возвращенную в результате выполнения SQL-запроса.

При помощи свойств и методов компонент, работающих с БД, осуществляется большинство механизмов доступа к данным, управление данными, их редактирование, выполняются различные манипуляции и осуществляется интерфейс.

Используя самое простое приложение, работающее с базой данных, проведем эксперимент. Исследуем возможности и время работы автономной базы данных на основе BDE (Borland Database Engine), с использованием запросов.

Для экспериментального исследования возможностей работы БД будем использовать БД «Книги». Данная БД является реляционной, локальной, основана на BDE, состоит из четырех таблиц (рис. 2). Количество записей в БД генерируется автоматически.

Эффективность работы БД будем определять временем работы запроса с разным количеством записей.

Исследуем два различных типа запросов:

1. Запрос на выборку данных из БД. «Выбрать из базы данных все книги Уоррена и писателей, которые родились после 1900 года».

2. Запрос с использованием объединения таблиц. «Объединить таблицы и выбрать сказки Пушкина, напечатанные в издательстве "Дрофа"».

Постепенно будем увеличивать количество записей в таблицах и подсчитывать время работы запроса соответственно (см. таблицу).

При увеличении количества записей время выполнения запроса увеличивается. Кроме того, запрос с использованием объединения таблиц прекращает работу при количестве записей равных 1500000, и система выводит сообщение об ошибке. Система также выводит сообщение об ошибке «table is full (таблица заполнена)» при попытке заполнения таблиц более чем 1500000 записей. Полученные данные отражены в графике (рис. 3).

При полученных результатах можно сделать следующие выводы:

1. Время выполнения запроса зависит от количества записей в БД и от сложности запроса. Чем больше записей в таблице и чем сложнее запрос, тем требуется больше времени на его выполнение;

2. Требуется достаточно большое количество времени на обработку запроса, что не эффективно в работе.

3. Таблицы имеют ограничение. В работе может использоваться не более чем 1 500 000 записей.

Таким образом, при больших масштабах работы, в производственной сфере реляционная БД на основе BDE работает не эффективно, так как требуется большое количество времени на обработку данных, и существует ограничение в количественном наборе данных.

Примечания

1. *Архангельский А.Е.* Программирование в Delphi 5. М.: ЗАО «Издательство БИНОМ», 2000. 1072 с.

2. *Данетеман Д. Тейлор Д.* Программирование в среде Delphi. Киев, 1995. 608 с.

3. *Обвинцев О.А., Внуковский Н.И* Обработка баз данных средствами языков программирования.

А. Н. Семенов

НЕКОТОРЫЕ КОНСТРУКЦИИ ПОЛУТЕЛ

В работе определены три новые конструкции полутел, которые будут использованы в дальнейшем исследовании полутел.

Введение

Множество S с бинарными операциями + и · и нульарными 0 и 1 называется *полукольцом* [1], если $\langle S, \cdot, 1\rangle$ – моноид, $\langle S, +, 0\rangle$ – коммутативный моноид, $c(a+b)d = cad + cbd$ и $a \cdot 0 = 0 \cdot a = 0$.

При таком определении полукольца образуют многообразие с морфизмами, сохраняющими все четыре операции, и, следовательно, переводящими противоположные элементы в противоположные, а взаимно обратные – во взаимно обратные.

Ассоциативные кольца с единицей (далее просто кольца) являются частным случаем полуколец. Из соображений универсальной алгебры каждое полукольцо обладает своим *кольцом разностей*. То есть для любого полукольца S существуют единственное кольцо $R(S)$ и морфизм $\varphi: S \to R(S)$, такие, что для любого кольца R и любого морфизма $\alpha: S \to R$ существует единственный морфизм $\beta: R(S) \to R$, для которого $\alpha = \beta\varphi$. Здесь и далее при использовании мультипликативной записи композиции отображений подразумеваем, что первым действует правое отображение. Конструкция кольца разностей достаточно проста: на множестве $S^2/\sim$, где $(a, b) \sim (c, d) \Leftrightarrow \exists e \in S: a+d+e = c+b+e$, вводятся кольцевые операции, подразумевая под парой (a, b) разность $a-b$. Понятно, что полукольцо вкладывается в кольцо разностей тогда и только тогда, когда оно удовлетворяет квазитождеству $a + c = b + c \Rightarrow a = b$, такие полукольца называются *сократимыми*. Другой крайний случай, когда кольцо разностей тривиально (одноэлементно), для этого необходимо и достаточно, чтобы любые два элемента полукольца уравнивались, т. е. $\forall a, b \in S\ \exists c \in S\ \ a+c=b+c$.

Еще один частный случай полукольца – *полутело* – множество, являющееся одновременно мультипликативной группой и аддитивной коммутативной полугруппой, причем умножение дистрибутивно относительно сложения с обеих сторон. При таком определении, строго говоря, полутело U полукольцом не является, но множество $U^0 = U \cup \{0\}$ с естественно доопределенными операциями становится полукольцом. Из некоторых соображений мы все-таки будем пользоваться данным определением. Полутело с коммутативным умножением называется *полуполем*.

СЕМЕНОВ Александр Николаевич– научный сотрудник кафедры алгебры и геометрии ВятГГУ

Так же, как и кольца, полутела образуют многообразие, но, в отличие от многообразия колец, многообразие полутел не является подмногообразием многообразия полуколец. Согласно универсальной алгебре существуют копроизведения полутел. То есть для любого семейства полутел $U_i, i \in \lambda$ существуют единственное полутело U и морфизмы $\varphi_i: U_i \to U$, такие, что для любого полутела V и любых морфизмов $\alpha_i: U_i \to V$ существует единственный морфизм $\beta: U \to V$, для которого $\alpha_i = \beta\varphi_i$ при всех i из λ. Конструкция копроизведения, в отличие от кольца разностей, более сложная и не такая явная.

Полутело называется *идемпотентным*, если в нем выполняется тождество $a + a = a$.

Полутело U назовем *зероидным*, если выполняется одно из равносильных условий:

1) $\exists\, a, b \in U\ \ a=a+b$;
2) $\forall\, a \in U\ \ \exists\, b \in U\ \ a=a+b$;
3) $\forall\, a \in U\ \exists\, c \in U\ \ b=b+a$;
4) $\forall\, a, b \in U\ \ \exists\, c \in U\ \ a+c=b+c$.

Очевидно, идемпотентное полутело является зероидным. Как показывает следующая конструкция, существуют зероидные неидемпотентные полутела [2].

Пусть U – полутело, G – линейно упорядоченная группа. На прямом произведении мультипликативных групп $G \times U$ определим операцию сложение:

$$(g_1,u_1)+(g_2,u_2)=\begin{cases}(g_1,u_1), & \text{если } g_1 > g_2;\\ (g_1,u_1+u_2), & \text{если } g_1 = g_2;\\ (g_2,u_2), & \text{если } g_1 < g_2.\end{cases}$$

Коммутативность сложения очевидна. Ассоциативность сложения и дистрибутивность умножения относительно сложения легко проверяются. Полученное полутело при нетривиальной группе G всегда зероидно, а если U – неидемпотентное полутело, то неидемпотентно.

Отметим, что строению полутел посвящена статья [3].

Сейчас приведем три новые конструкции полутел, интересные сами по себе и своими применениями.

1. Обобщенное матричное полутело

Пусть $A_0, A_1, A_2, \ldots$ – произвольные непустые непересекающиеся множества, количество которых конечно или бесконечно, и $A = \bigcup A_i$. Пусть также определены две функции $l, p: A \setminus A_0 \to A$, такие, что $a \in A_i, i > 0$ влечет $l(a), p(a) \in A_{i-1}$ и $a \in A_i, i > 1$ влечет $lp(a) = pl(a)$, где для композиций этих функций используется мультипликативная запись. Полученную конструкцию назовем формой. Сейчас каждому элементу $a \in A_0$ поставим в соответствие полутело S_a, а каждому $a \in A \setminus A_0$ кольцо S_a (возможно тривиальное) так, что для любого $a \in A \setminus A_0$ можно фиксировать два морфизма: $\alpha_a: S_{l(a)} \to S_a$ и $\beta_a: S_{p(a)} \to S_a$, для которых $\alpha_a\beta_{l(a)} = \beta_a\alpha_{p(a)}$ при каждом $a \in A \setminus (A_0 \cup A_1)$.

Последнее условие позволяет однозначно построить морфизм из $S_{p^k l^s(a)}$ в S_a для любого $a \in A_i$ и для любых неотрицательных целых k и s, $k+s \le i$. Действительно, чередуя функции l и p, из a в $p^k l^s(a)$ можно прийти разными путями. Для каждого пути строится композиция соответствующих морфизмов, но, как нетрудно видеть, все эти композиции будут совпадать. Результат применения полученного морфизма к элементу $x \in S_{p^k l^s(a)}$ будем обозначать x_a. Таким образом,

$$x_a = \alpha_a \ldots \alpha_{l^{s-2}(a)}\alpha_{l^{s-1}(a)}\beta_{l^s(a)}\beta_{pl^s(a)} \ldots \beta_{p^{k-1}l^s(a)}(x) =$$

$$= \alpha_a \ldots \alpha_{l^{s-2}(a)}\beta_{l^{s-1}(a)}\alpha_{pl^{s-1}(a)}\beta_{pl^s(a)} \ldots \beta_{p^{k-1}l^s(a)}(x) = \ldots.$$

На множестве $F = \left\{f: A \to \bigcup_{a \in A} S_a,\ f(a) \in S_a\right\}$ определим операции + и · следующим образом:

$$(f+g)(a) = f(a)+g(a).$$

Очевидно, $\langle F, +\rangle$ – коммутативная полугруппа, поскольку является прямым произведением коммутативных полугрупп $\langle S_a, +\rangle$.

$$(fg)(a) = \sum_{i,j \ge 0; i+j=n} f(l^i(a))_a\, g(p^j(a))_a \quad \text{при } a \in A_n.$$

Дистрибутивность с обеих сторон умножения относительно сложения очевидна. Докажем ассоциативность умножения в F. Пусть $a \in A_n$, тогда

$$(f(gh))(a) = \sum_{i+j=n} f(l^i(a))_a\,(gh)(p^j(a))_a =$$

$$= \sum_{i+j=n} f(l^i(a))_a \left(\sum_{k+m=i} g(l^k p^j(a))_{p^j(a)}\, h(p^{m+j}(a))_{p^j(a)}\right)_a =$$

$$= \sum_{i+j=n;k+s=n;s \ge j} f(l^i(a))_a\, g(l^k p^j(a))_a\, h(p^s(a))_a =$$

$$= \sum_{k+s=n} \left(\sum_{m+j=s} f(l^{m+k}(a))_{l^k(a)}\, g(p^j l^k(a))_{l^k(a)}\right)_a h(p^s(a))_a =$$

$$= \sum_{k+s=n} (fg)(l^k(a))_a\, h(p^s(a))_a = ((fg)h)(a).$$

Покажем, что множество F по умножению – группа. Для этого достаточно найти такой элемент $e \in F$, что для любого $f \in F$ существует $g \in F$, для которого верны равенства $fe=f$ и $fg=e$. Положим $e(a)=\begin{cases}1, \text{если } a\in A_0\\ 0, \text{если } a\in A\setminus A_0\end{cases}$. Тогда в формуле $(fe)(a)$ при $a\in A_n$ все слагаемые, кроме одного – $f(l^0(a))_a e(p^n(a))_a = f(a)$, будут нулевыми. Итак, $fe=f$. Покажем сейчас индукцией по k, что мы можем подобрать значения функции g в точках $a\in A_k$, не меняя значений в точках $a\in\bigcup_{i=0}^{k-1}A_i$ так, что $(fg)(a)=e(a)$ при $a\in\bigcup_{i=0}^{k}A_i$. При $k=0$ положим $g(a)=(f(a))^{-1}$, поскольку $f(a)$ – элемент полутела S_a, то он обратим. Очевидно, $(fg)(a)=1=e(a)$. Пусть теперь $k>0$ и функция g уже определена в точках $a\in\bigcup_{i=0}^{k-1}A_i$ и в этих точках $(fg)(a)=e(a)$. Для каждого $a\in A_k$ положим

$$g(a)=-\left(f(l^k(a))\right)_a^{-1}\sum_{i=0}^{k-1}f(l^i(a))_a\,g(p^{k-i}(a))_a.$$

Это определение корректно, поскольку $l^k(a)\in A_0$, значит, $f(l^k(a))$ – элемент полутела, кроме того, $p^{k-i}(a)\in A_i$, следовательно, функция g уже определена в этих точках. Тогда $(fg)(a)=0=e(a)$ при $a\in A_k$. При этом мы не изменили $(fg)(a)$ для $a\in\bigcup_{i=0}^{k-1}A_i$. Индукция доказана. Значит, мы можем определить функцию g во всех точках так, что $fg=e$. В действительности, можно выписать явную формулу для g, но она громоздка и вряд ли больше отражает ситуацию.

Таким образом, умножение – групповая операция, и построенное множество функций F – полутело, которое можно назвать обобщенным матричным полутелом.

2. Формальные бесконечные суммы

Хорошо известно, что для упорядоченного множества эквивалентны условия минимальности, индуктивности и обрыва убывающих цепей.

Упорядоченное множество называется множеством с условием конечной минимальности (*КМ-множеством*), если выполняется одно из равносильных условий [4]:

1. Во всяком непустом его подмножестве множество минимальных элементов не пусто и конечно.
2. Оно удовлетворяет условию обрыва убывающих цепей и всякая его антицепь конечна.
3. Во всяком бесконечном его подмножестве можно выбрать (строго) возрастающую (бесконечную) последовательность.
4. Во всякой последовательности $\{a_i\}$, $i\in\mathbf{N}$, его элементов найдутся $a_i\le a_{i+k}$, $k>0$.

Прямое произведение упорядоченных множеств A_i, $i\in\alpha\neq\varnothing$ есть $\prod_{i\in\alpha}A_i$ с порядком $a\ge b\Leftrightarrow\pi_i(a)\ge\pi_i(b)$ для всех $i\in\alpha$. Прямое произведение КМ-множеств будет КМ-множеством, если и только если неодноэлементных среди них конечное число [4].

Лемма. Если B и C – КМ-подмножества упорядоченной группы G, то

a) для любого $a\in BC$ множество $\{(b,c)\in B\times C: bc=a\}$ конечно;

b) BC есть КМ-подмножество.

Доказательство

a) Поскольку неравенство $(b_1,c_1)<(b_2,c_2)$ элементов $B\times C$ влечет неравенство $b_1c_1<b_2c_2$ элементов G, то искомое множество будет антицепью в КМ-множестве $B\times C$, значит, будет конечным.

b) Для каждого $a\in BC$ выберем элемент $(b_a,c_a)\in B\times C$, такой, что $a=b_ac_a$. Пусть A – бесконечное подмножество BC, тогда в соответствующем ему бесконечном подмножестве произведения $B\times C$ можно выбрать возрастающую последовательность (b_i,c_i), которой, в силу указанного следствия неравенств, будет соответствовать возрастающая последовательность в A.

Пусть G – произвольная решеточно упорядоченная группа, H – ее l-подгруппа, U – полутело, K – ассоциативное кольцо с единицей (возможно тривиальное), $\varphi: U\to K$ – полукольцевой морфизм, сохраняющий 1.

Рассмотрим множество F всех функций $f: G\to K\cup U$ (дизъюнктное объединение), которые переводят в U только один элемент из G – $f'\in H$, и носитель которых – $S(f)=\{a\in G: f(a)\in(K\setminus\{0\})\cup U\}$ – КМ-множество с наименьшим элементом f'. Умножение на F определим так:

$$(fg)(a)=\sum_{bc=a}f(b)\cdot g(c).$$

Мы подразумеваем, что $x\cdot y=\varphi(x)\cdot y$, если $x\in U$, $y\in K$, и $x\cdot y=x\cdot\varphi(y)$, если $x\in K$, $y\in U$. Кроме того, $x+0_K=x$ для $x\in U$, и бесконечная сумма нулей есть нуль.

Понятно, что если $a\notin S(f)S(g)$, то $(fg)(a)=0\in K$. Если же $a\in S(f)S(g)$, то можем считать, что $b\in S(f)$, $c\in S(g)$, и, в силу леммы a), в каждой сумме конечное число ненулевых слагаемых. Таким образом, fg – функция, носитель которой $S(fg)\subseteq S(f)S(g)$ в силу

леммы b) является КМ-множеством, а наименьший элемент – $(fg)'=f'g'$. Следовательно, операция определена корректно. Ее ассоциативность очевидна. С помощью нётеровой индукции несложно показать, что введенная операция будет групповой.

Сложение на F определим так:

$$(f+g)(a)=\begin{cases}f(f')+g(g'), \text{если } a=f'\vee g',\\ \left[g'\le f'\right]f(a)+\left[f'\le g'\right]g(a), \text{в других случаях,}\end{cases}$$

где результат в квадратных скобках равен 1, если там стоит верное высказывание, и 0, если неверное.

Нетрудно видеть, что операция определена корректно и она будет ассоциативной и коммутативной. Вообще:

$$\left(\sum f_i\right)(a)=\begin{cases}\sum f_i(f_i'), \text{если } a=\vee f_i',\\ \sum_i\left[\sum_{j\neq i}f_j'\le f_i'\right]f_i(a), \text{в других случаях.}\end{cases}$$

Дистрибутивность тоже нетрудно проверить. Значит, F – полутело. Если в этой формуле заменить $\vee$ на $\wedge$ и $\le$ на $\ge$, то получим другое сложение.

При частных условиях можно определить другие сложения.

1. H содержится в центре группы G:

$$(f+g)(a)=f(af'(f'\vee g')^{-1})+g(ag'(f'\vee g')^{-1}).$$

Вместо $\vee$ можно взять $\wedge$.

2. H – линейная:

$(f+g)(a)=f(a)+g(a)$ или

$$(f+g)(a)=\begin{cases}f(a)+g(a), \text{если } f'=g'\\ f(a), \text{если } f'>g'\\ g(a), \text{если } f'<g'\end{cases}$$

3. Неполное прямое произведение полутел

Пусть $A_i(i\in\alpha)$ – полутела. Мы знаем, что для каждого $i\in\alpha$ существует единственный морфизм $\varphi_i:\mathbf{Q}^+\to A_i$. Носителем элемента x прямого произведения $\prod_{i\in\alpha}A_i$ с проекциями π_i назовем множество $\{i\in\alpha:\pi_i(x)\notin\operatorname{Im}\varphi_i\}$ или, проще говоря, множество координат, значения в которых нерациональны. Нетрудно проверить, что множество всех элементов с конечным (возможно пустым) носителем будет подпрямым произведением полутел $A_i(i\in\alpha)$. Полученное полутело можно назвать неполным прямым произведением полутел $\prod_{i\in\alpha}A_i$. Аналогично, если D некоторый фильтр на α, то множество всех элементов из $\prod_{i\in\alpha}A_i$, дополнение носителя которых принадлежит D, будет подполутелом в $\prod_{i\in\alpha}A_i$.

Примечания

1. *Golan J. S.* The theory of semirings with applications in mathematics and theoretical computer science. N.Y.: Pitman, 1991.

2. *Богданов И. И.* Об аддитивной структуре полутел // Вестник МГУ. Сер. Математика, механика (в печати).

3. *Семенов А. Н.* О строении полутел // Вестник Вятского государственного гуманитарного университета. 2003. № 8. С. 105-107.

4. *Вечтомов Е. М., Варанкина В. И.* Упорядоченные множества с конечным условием минимальности // Вестник ВятГГУ. 2000. № 3–4. С. 11-12.

В. Ю. Автамонов

АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

В статье рассматривается новая парадигма программирования – аспектно-ориентированное программирование (АОП). Данная парадигма возникла недавно, и пока только определены ее основные рамки. Даются важнейшие понятия АОП, проводится сравнение АОП с объектно-ориентированным программированием.

В процессе эволюции человечество постоянно пытается решить проблему «Как организовать максимальный выход продукта при минимальных затратах?». Причём эту проблему человек пытается решить во всех областях своей деятельности, будь то производство хлебобулочных изделий или же разработка очередного спутника NASA. То же самое можно сказать и о программировании. Всю эволюцию программирования можно представить как процесс, направленный на снижение программного кода и увеличение эффективности его использования. В эпоху зарождения программирования программы представляли собой набор машинных команд. Основное занятие программистов заключалось в составлении машинных инструкций. Но компьютерная индустрия развивается динамично, растут задачи и требования. Как результат появились языки программирования более высокого уровня, позволяющие отойти от машинного кода, абстрагироваться и более визуально решать поставленные проблемы. Появились структурированные языки, здесь уже пришлось думать о том, как разбить большое задание, разработку общего поведения

АВТАМОНОВ Виталий Юрьевич – ассистент кафедры информатики и МОИ ВятГГУ

и интерфейса для взаимосвязанных концепций и изменение обычного поведения более специализированных компонентов без обращения к реализации базовой концепции. Таким образом, на данном этапе появилась возможность избежать множества ошибок, сопутствующих структурному и модульному программированию, внести большую ясность при разработке программного обеспечения. Как вполне естественный результат развития и совершенствования техники ООП возникло компонентное программирование, позволившее более эффективную организацию данных.

Так, наблюдая за развитием программирования, можно сделать следующие выводы: чем дальше идет эволюция программирования, тем большему обобщению подвергаются его составляющие. Сначала организовали набор машинных инструкций в единые блоки, далее эти блоки совместили в одном модуле. Очевидно, что чем дальше идет развитие, тем больше уровень комплексности и абстракции, тем все более становится скрытным базовая реализация тех компонентов программирования, с которыми мы работаем.

Как мы уже говорили, сложность программного обеспечения и методологий программирования постоянно возрастает, также постоянно предпринимаются попытки разорвать этот замкнутый круг. Xerox PARC, IBM и Microsoft готовят для индустрии ПО новую парадигму программирования. Она называется аспектно-ориентированное программирование (АОП, aspect-oriented programming) и призвана прийти на смену объектно-ориентированному (ООП, object-oriented programming) и компонентному (component-based programming) программированию. Основными идеологами нового программирования являются Грегор Кишалес (Xerox PARC) и Чарльз Саймони, бывший ведущий архитектор Microsoft.

Аспектно-ориентированное программирование (АОП) (aspect-oriented programming, AOP) — парадигма, изобретенная в Xerox PARC в 90-х гг. и позволяющая разработчику четко разделять задачи, которые не следует запутывать в клубок, например математические операции и обработку исключений. АОП имеет ряд преимуществ. Во-первых, оно повышает производительность, так как операции выполняются быстрее. Во-вторых, программисты тратят меньше времени на переписывание одного и того же кода. В целом, АОП обеспечивает более качественную инкапсуляцию различных процедур и облегчает взаимодействие с кодом, написанным на других языках программирования.

Состояние развития АОП похоже на состояние ООП 20 лет назад, данная методика находится на стадии исследований, ещё только определены её рамки и основные понятия. Парадигма аспектно-ориентированного программирования (АОП, aspect-oriented programming) призвана прийти на смену объектно-ориентированному (ООП, object-oriented programming) и компонентному (component-based programming) программированию. По оценкам специалистов, около 70% времени в проектах тратится на сопровождение и внесение изменений в готовый программный код. Вот почему столь важной в ближайшей перспективе становится роль АОП и подобных трансформационных подходов.

Направление развития программирования очевидно. Никому не хочется тратить время и усилия на изменение готовых разработок. Также остро стоят проблемы «повторного использования кода» и «пересекающегося кода» (code crosscutting). Проблема пересекающегося кода состоит в том, что при реализации решения задачи встречается программный код, по сути выполняющий одну и ту же функцию в разных частях программы. Это выражается в плохой трассировке, низкой продуктивности, плохом качестве программного кода, более трудной модификации и низкой эффективности повторного использования кода. Проблема повторного использования кода начала решаться с появления программирования, но нашла своё решение при появлении ООП. До этого этапа можно отметить значительный вклад в решение этой задачи модульного программирования, но здесь можно было использовать модули, если между разработчиками программ были оговорены правила их использования и применения. С появлением объектно-ориентированного программирования стало возможным организовать относительно слабосвязанную систему, компоненты которой менее зависят друг от друга, как это было ранее, кроме того, разработанные объекты стало возможным применять в программе, не задумываясь о том, кто их написал. Такое положение сейчас уже вполне естественно, каждый человек, хоть немного знакомый с программированием, сейчас без проблем может написать объектно-ориентированную программу (используя, например, язык программирования Delphi), не задумываясь о том, что за компонентом, который он выбрал, стоит не одна сотня строк программного кода. Как мы уже говорили, всё очевидно. Никто не желает выполнять лишнюю работу, под девизом «лень – двигатель прогресса» проходит вся эволюция человека, это касается всех сфер его деятельности, в том числе и программирования.

Рассмотрим новую парадигму подробнее. С точки зрения АОП в процессе разработки достаточно сложной системы программист решает две ортогональные задачи:

- разработка компонентов;
- разработка сервисов, поддерживающих взаимодействие компонентов.

Такие языки программирования, как, например, C++, VB, Delphi и т. п., ориентированы прежде всего на решение первой задачи. Код компонента представляется в виде класса, то есть он хорошо локализован и, следовательно, его легко просматривать, изучать, модифицировать, повторно использовать. С другой стороны, при *программировании процессов, в которые вовлечены различные объекты*, мы получаем код, в котором элементы, связанные с поддержкой такого процесса, *распределены по коду всей системы*. Эти

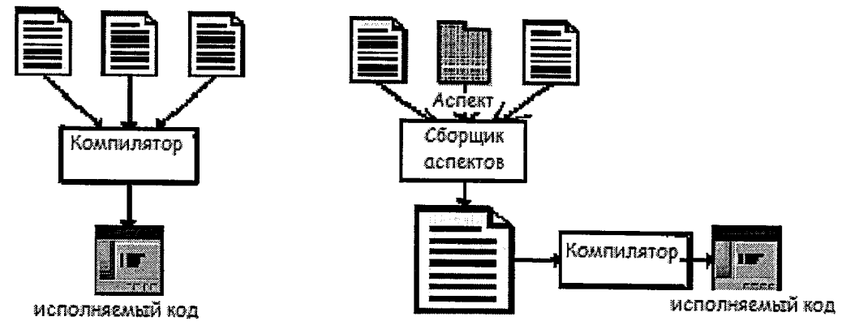


Схема работы АОП. По сравнению с традиционным подходом после этапа кодирования компонентов и аспектов на соответствующих языках выполняется автоматическое построение оптимизированного для выполнения (но не для просмотра и модификации) кода. Сборщик аспектов может автоматически вставить точку входа для любого набора

элементы встречаются в коде множества классов, их совокупность в целом не локализована в обозримом сегменте кода. В результате мы сталкиваемся с проблемой *«запутанного» и «пересекающегося» кода (code tangling и code crosscutting)*.

АОП создает систему, используя слабосвязанные модульные разработки пересекающихся моделей. Модульная часть в аспектно-ориентированном программировании называется аспектом, тогда как общая модель в ООП носит название *класс*.

Аспект (ключевое понятие новой парадигмы) представляет собой языковую концепцию, схожую с классом, но только более высокого уровня абстракции. Аспекты могут затрагивать многие классы и используют так называемые точки вставки (insertion points) для реализации регулярных действий (например, связанных с безопасностью, обработкой ошибок и т. п.), которые обычно «размазаны» по всему тексту программы. АОП отличается от ООП тем, как адресуются пересекающиеся модели. В АОП реализация каждой модели не подозревает о существовании другой, пересекающей ее структуру. Вспомним, что при использовании техники ООП создаётся система из объектов, связь между которыми осуществляется через действия.

Технология АОП позволяет выделить многократно повторяющийся код в простые блоки, называющиеся *аспектами*. Аспекты инкапсулируют множество классов в один модуль, к которому можно осуществлять обращение из любого места программного кода. Требования аспектного программирования заключаются в том, что при разработке аспект рассматривают как модель, пересекающую структуру другой модели.

В АОП каждый аспект может быть выражен в единичной форме, и далее они автоматически комбинируются в исполняемый код с помощью так называемого *сборщика (aspect weaver)*. В результате **каждый аспект** представляет реализацию некоторого **числа модулей, или объектов**, увеличивая возможности повторного использования кода. На рисунке представлена схема работы АОП.

Таким образом, АОП позволяет снизить объём кода программы и увеличить эффективность повторного использования кода.

В рамках АОП утверждается, что никакая технология проектирования не поможет решить данную проблему, если оставаться в рамках языка, ориентированного только на разработку *компонентов*. Для программирования *сервисов*, обеспечивающих взаимодействие объектов, нужны *специальные средства*, возможно, специальные языки.

Понятие *аспект* можно определить так: «некоторая модель является аспектом другой модели, если она пересекает (crosscuts) ее структуру». Иными словами, понятие аспекта относительно. Если, например, в качестве модели некоторой системы мы рассматриваем совокупность компонентов, представляющих такие сущности, как вкладчик, счет, банк, то аспектами являются сервисы, обеспечивающие синхронизацию доступа к счету, распределенные транзакции, безопасность. То есть автоматически выполняемые сервисы, обеспечивающие слаженную, надежную, безопасную работу компонентов.

Преимущества АОП

Как мы уже говорили, АОП помогает решить проблемы, касающиеся запутанного кода (code tangling) и разбросанного кода (code scattering). Специфичные преимущества АОП таковы.

- ***Модульная реализация пересекающихся моделей.*** АОП адресует каждую модель отдельно, с минимальной связностью, что отражается в модульной реализации, даже если присутствует перекрещивание моделей. Такая реализация снижает дублирование кода. Так как каждая проблема реализуется отдельно,

это позволяет снизить размер кода. В дальнейшем модульная реализация приводит к более легкому пониманию и сопровождению системы.

• ***Системы легки для развития***. Так как аспектные модули могут не знать о пересечении проблем, то легко можно добавить функциональность путем создания нового аспекта. В дальнейшем при добавлении нового аспекта в систему существующие аспекты пересекают его структуру, помогая создавать гармоничную эволюцию.

• ***Более поздняя связь с проектным решением.*** Вспомним проблему разработчиков решить проблему, придерживаясь ее рамок, или расширить ее границы. С АОП разработчики могут задержаться с разработкой проекта для будущих требований, так как они могут быть реализованы в будущем как отдельные классы.

• ***Более широкие возможности повторного использования***. Так как АОП решение – это аспекты, как отдельные модули, каждый отдельный модуль слабо связан с остальными. Например, вы можете использовать модуль взаимодействия с базой данных в отдельном аспекте регистрационного механизма с различными требованиями регистрации. В общем, слабосвязанные реализации являются ключом к широкому повторному использованию кода. АОП представляет более слабосвязанные решения, чем ООП.

Нужно ли нам АОП? Устраняет ли АОП проектировочные недостатки? В АОП каждая реализация модели означает тот факт, что она пересекает структуру другой модели, которая может не подозревать о ее существовании. Такая «забывчивость» и отличает АОП от технологий ООП. В АОП поток формируется из пересекающихся моделей в главную модель, в то время как в ООП поток работает в противоположном направлении. Заметим тем не менее, что АОП и ООП неплохо уживаются рядом. Например, можно представить смешанный класс, как модель, используя как АОП, так и ООП, только на уровне АОП это будет более удобно. В любом случае реализация пересекающейся модели смешанного класса не нуждается в том, чтобы знать, каким образом она была реализована. Например, в одном случае вы можете использовать интерфейс регистратора событий как смешанный класс, в другом – как аспект. Мы только можем говорить об эволюции ООП в АОП.

Пока АОП – это еще только новая идея, элементы которой используются в уже существующих технологиях.

Так, например, АОП можно поддерживать в рамках уже существующих языков. В частности, исследовательский центр Xerox PARC разработал систему AspectJ (*www.aspectj.org*), поддерживающую АОП в рамках языка Java. В ноябре 2002 г. вышел новый релиз AspectJ 1.1. Этот пакет встраивается в такие системы разработки, как Eclipse, Sun ONE Studio (*http://forte.sun.com/ffj/articles/aspectJ.html*) и Borland JBuilder. Другой исследовательский центр – IBM Research — выпустил альфа-версию HyperJ (*http://www.alphaworks.ibm.com/tech/hyperj*) и готовит к марту 2003 г. выпуск системы Cosmos (*http://www.research.ibm.com/AEM/mdsoc.html*) с гипертекстовой поддержкой требований и диаграмм. Cosmos является развитием MDSOC, которая, в свою очередь, опирается на идеи субъектно-ориентированного программирования (*http://www.research.ibm.com/sop/*). Помимо Java новая идеология поддерживается и в других языках, таких, как Си, Си++, Squeak/Smalltalk, Perl, Python, Ruby (*http://aosd.net/tools.html*). Пока интерес к новой парадигме проявляют несколько сотен экспертов из General Electric, Hewlett-Packard, Siemens, Motorola, Oracle и Sony. Очевидно, что их число будет расти.

Примечания

Более подробно с новой парадигмой можно ознакомиться по следующим адресам.

1. http://ccs.neu.edu
2. http://clubpro.spb.ru/articles/
3. http://creonet.cdu.edu.ua/index.html
4. http://emanual.ru/cgi-bin/show.pl?16,3
5. http://forte.sun.com/articles/index.html
6. http://newasp.omskreg.ru/intellect/f29.htm
7. http://sp.cs.msu.ru/specseminar/ds/aop/aop.pdf.gz
8. http://sp.cs.msu.ru/specseminar/ds/aop/d.pdf.gz
9. http://sp.cs.msu.ru/specseminar/ds/aop/rg.pdf.gz
10. http://www.agilealliance.com/articles/searchResults?topic=Aspect+Oriented+Programming
11. http://www.ccs.neu.edu/home/lieber/AOP.html
12. http://www.comprice.ru/logovo/2003-21.phtm
13. http://www.cs.ust.hk/~scc/comp610e/assignment/reading04.pdf
14. http://www.itbook.ru/view_book_sheet.asp?FROM=SINGLE&BOOKID=1071&EXP=27
15. http://www.osp.ru/os/1999/07-08/11.htm
16. http://www.osp.ru/pcworld/2003/01/142_3.htm
17. http://www.prefnews.ruual.ru/cgi-bin/show.pl?16,3
18. http://www-ia.tu-ilmenau.de/~czarn/aop/letter/aop.zip
19. *Иванова Г. С., Ничушкина Т. Н., Пугачев Е. К.* Объектно-ориентированное программирование. М.: МГТУ им. Н. Э. Баумана, 2001.

М. С. Шиляева

ОФИСНОЕ ПРОГРАММИРОВАНИЕ И ВОЗМОЖНОСТИ ЕГО ИСПОЛЬЗОВАНИЯ ДЛЯ ЭКОНОМИЧЕСКИХ СПЕЦИАЛЬНОСТЕЙ

В статье рассматриваются вопросы достаточно нового и перспективного направления в информатике – офисного программирования. Сочетая работу с документами среды MS Office и их настройку средствами языка VBA, данная дисциплина может быть успешно использована для студентов некоторых экономических специальностей.

ШИЛЯЕВА Марина Сергеевна – ассистент кафедры информационых технологий и ТСО ВятГГУ

Офисное программирование – это достаточно новое и перспективное направление в современном программировании, и спектр его применения очень широк – от настройки отдельных документов до создания серьезных решений масштаба предприятия. Офисное программирование – дисциплина очень молодая, а значит, как в любом новом направлении, пока еще не существует единого понимания предмета. Часто его отождествляют с программированием на VBA (Visual Basic for Application), теряя при этом основную цель технологии: создание системы документов. Конечно, пользователи, работающие в среде Office, могут создавать документы и работать с ними без всякого программирования благодаря тем стандартным возможностям, которые обеспечивает эта среда. Но рано или поздно наступает момент, когда стандартных возможностей среды не хватает для удовлетворения растущих потребностей пользователя и решения специфических задач, возникающих в той проблемной области, в которой он работает. С этого момента и начинается Офисное программирование.

Особенностью Офисного программирования является то, что документы создаются в некоторой достаточно мощной среде, классическим примером которой является среда Office, а язык программирования встроен в среду. И снова примером может служить встраивание языка Visual Basic в среду Office, после чего он и становится языком VBA (Visual Basic for Application).

Следует отметить, что среда, помимо широкого спектра возможностей, привлекающих пользователя для работы в ней, должна обладать следующими качествами:

• быть построена на принципах объектного программирования и, по существу, представлять совокупность объектов, определяющих тот мир, в котором работает пользователь;

• иметь встроенный язык программирования, позволяющий работать с объектами среды, а значит, с ее документами, без каких-либо чрезвычайных усилий. Принципиально важно, чтобы встроенный язык был обычным языком программирования, применяемым программистами и вне среды. Корпорация Microsoft сделала революционный шаг – она не стала создавать в среде Office 2000 собственный язык программирования, как это делалось ранее в большинстве известных сред (Oracle, FoxPro, AutoCad), а встроила в среду язык, уже известный программистам. Важно и то, что VBA является отчуждаемым от среды и может быть встроен в различные среды. Так что язык VBA в этом отношении становится схож с естественным языком, встраиваемым во все профессиональные области знания.

Офисное программирование имеет свою специфику и отличается от программирования в программной среде таких языков, как Visual Basic, Visual C ++ или Delphi, ориентированных на создание программных проектов различного типа: так, мощная и разнообразная среда приложений Office ориентирована в первую очередь на пользователей, а не на программистов. И программист обычно начинает работать с документами не на пустом месте, а с их заготовками, созданными пользователями. Кроме того, меняются цели, приоритеты, сам взгляд на деятельность программиста, работающего в среде Office. Ранее программист создавал приложение, понимаемое как программа, программный проект. Теперь программист является одним из участников создания системы документов. Документ, а не программа, становится целью разработки. Программный проект — это лишь часть документа. В Office программный проект неразрывно связан с документом, хранится как часть документа и не может существовать независимо от него.

Необходимо отметить, что возможность настроить стандартный документ Office, сделать его «по заказу», снабдить новыми функциями, учитывающими специфику решаемой задачи — это одна из важнейших особенностей Офисного программирования. Настройка может быть очень простой и состоять в том, что стандартный документ получает дополнительные полезные свойства, расширяющие его возможности или изменяющие внешний вид. Обычно так начинают свой путь в Офисное программирование продвинутые пользователи Office. Но настройка документа может быть и очень сложной, а документ, сделанный по заказу, может ничем не напоминать обычный стандартный документ Office.

Изучение Офисного программирования может быть успешно применено при изучении курса «Разработка и применение пакетов прикладных программ», входящих в программу высшего образования студентов некоторых экономических специальностей. Специфика данных специальностей такова, что, обладая базовыми знаниями по курсам «Информатика» и «Программное обеспечение ЭВМ» и неплохо владея программным пакетом Microsoft Office, студентам не хватает элементарных знаний основ программирования. Поэтому одним из вариантов решения данной проблемы может быть знакомство с основными принципами и приемами Офисного программирования, которое позволяет соблюсти баланс интересов между «хорошо знакомой» средой создания документов и «незнакомым» программированием.

Естественно, что современный рынок программных средств и продуктов предлагает широкий выбор специализированных финансовых пакетов, таких, как 1С Бухгалтерия – для проведения бухгалтерского учета, Microsoft Project – для организации календарного планирования и управления и т.д. Однако наиболее приемлемым и эффективным для использования в учебном процессе является интегрированный пакет Microsoft Office. К преимуществам данного программного продукта можно отнести его доступность: на

любой персональный компьютер в первую очередь устанавливается пакет Microsoft Office. Второе преимущество заключается в том, что при использовании Microsoft Office для решения финансовых, организационных, управленческих задач пользователю самому приходится вводить расчетные формулы, что делает «прозрачными» алгоритмы обработки данных. И это, несомненно, очень важно в процессе обучения, в то время как перечисленные выше специализированные программные продукты обеспечивают ввод данных и вывод результата, скрывая от пользователя сам процесс обработки информации.

Особенностью Офисного программирования является возможность создать программный проект или, по крайней мере, его отдельные компоненты без программирования. Эта возможность основана на использовании такого характерного для Офисного программирования инструмента, как MacroRecorder. MacroRecorder — это транслятор действий, записывающий действия пользователя при работе вручную и транслирующий их в программу на языке VBA. В итоге пользователь получает документы, обладающие новыми функциями и способные решать задачи, характерные для проблемной области пользователя. Далее пользователь сам, не будучи программистом, способен создавать простые программируемые документы Office, постепенно совершенствуясь в этой деятельности. Кроме того, идеи визуального и событийно-управляемого программирования получают в Офисном программировании свое дальнейшее развитие, а значит, пользователь в полной мере берет на себя управление своим документом.

Таким образом, в рамках курса Офисное программирование студенты знакомятся с основами программирования (основные алгоритмические конструкции и типы данных), с понятием объектно-ориентированного программирования (объекты и их семейства, свойства и методы, события) и применяют полученные знания для создания систем документов, которые могут быть применены в сфере их профессиональной деятельности.

Примечания

1. *Биллиг В. А., Дехтярь М. И.* VBA и Office 97. Офисное программирование. М.: «Русская Редакция», 1998.

2. *Биллиг В. А.* VBA в Office 2000. Офисное программирование. М.: «Русская Редакция», 1999.

3. *Биллиг В. А.* Введение в офисное программирование // Компьютер-ИНФО. 2000. № 30 (214).

4. Информатика: Учебник / Под ред. Н. В. Макаровой. М.: Финансы и статистика, 1997.

5. Практикум по информатике: Учебник / Под ред. Н. В. Макаровой. М.: Финансы и статистика, 1997.

Н. В. Исупова

К ВОПРОСУ О ПРЕПОДАВАНИИ КУРСА «СОЦИАЛЬНАЯ ИНФОРМАТИКА»

Статья посвящена новому разделу в информатике – социальной информатике. Обосновывается целесообразность преподавания данной дисциплины в университете. В статье предлагаются возможные темы теоретических и практических занятий по курсу «Социальная информатика».

С развитием вычислительной техники, информационных технологий, телекоммуникаций наметилась тенденция к самостоятельному развитию в информатике различных отраслевых направлений, таких, как правовая, экономическая, педагогическая, социальная, медицинская информатика, информационная психология и т. п.

Все эти направления поддерживают информационные технологии определенных предметных областей деятельности человека. И это понятно – общество увидело громадные возможности и преимущества использования информационных технологий и телекоммуникационных средств в своих профессиональных сферах. Так, например, правовая информатика занимается вопросами компьютерной поддержки правовой стороны жизни общества, разработкой различных правовых информационно-поисковых систем; информационная психология – вопросами влияния информационных технологий на сознание и психику человека; педагогическая информатика занимается проблемами создания и реализации концепции образования людей, которым предстоит жить в информационном обществе.

Очевидно, что данные направления уже начинают оформляться в виде отдельных научных дисциплин и они, несомненно, показывают гигантскую роль информационных технологий и телекоммуникационных средств в переработке, анализе и использовании информации в конкретной предметной области для получения знаний нового качества и их применения для выработки какого-либо решения.

Из этих отраслевых направлений можно выделить те, результаты исследований которых могут быть использованы в любой предметной области. Можно провести аналогию с информационными технологиями общего пользования, как, например, текстовый процессор, табличный процессор и т.п. При ясном и правильном понимании возможностей этих технологий они могут быть использованы в любой области, но с учетом ее специфики. Так, например, бухгалтеры могут использовать электронную таблицу для ведения каких-либо математических расчетов, социологи для построения графиков и анализа данных по ним, учителя для ведения электронных классных журналов.

ИСУПОВА Наталья Валентиновна – старший преподаватель кафедры информационных технологий и ТСО ВятГТУ

Очевидно, что такой научной дисциплиной может стать ***социальная информатика как наука, изучающая комплекс проблем, связанных с прохождением информационных процессов в социуме***. Предметом изучения социальной информатики как науки являются процессы информатизации общества, а также их воздействие на социальные процессы, в том числе на развитие и положение человека в обществе, на изменение социальных структур общества под влиянием информатизации [1]. За последние десять лет из технической дисциплины о методах и средствах обработки данных при помощи вычислительной техники информатика все больше превращается в фундаментальную науку об информации и информационных процессах не только в технических системах, но также в природе и обществе [2].

Термин «социальная информатика» был впервые предложен профессором А. В. Соколовым в середине 70-х годов для обозначения новой научной дисциплины, «изучающей посредством информационного подхода общественное знание, социальную коммуникацию и управление обществом» [3]. Один из основоположников социальной информатики академик А. Д. Урсул рассматривает ее как научную базу формирования зарождающегося информационного общества. В своих работах А. Д. Урсул убедительно показал социальный, по сути, характер процесса информатизации общества, в котором вычислительная техника и новые информационные технологии являются лишь средствами для более эффективного овладения информацией в целях социального прогресса [4].

Само определение этой науки показывает на целесообразность ее преподавания в вузе, так как все мы переживаем переход к жизни в информационном обществе, где все отношения пронизываются современными информационными технологиями и телекоммуникационными средствами. Российская система образования не может далее не замечать этого глобального процесса, который уже охватил практически все передовые страны мирового сообщества, не видеть темпов развития этого процесса и его возможных геополитических и социально-экономических последствий [5]. Как информационные технологии влияют на отдельную личность, на отношения общество – личность, личность – социальная группа, общество – социальная группа? Какие возникают при этом проблемы? Каковы пути разрешения этих проблем? Все это направления исследований социальной информатики.

Информатизация общества оказывает революционное воздействие на все сферы жизнедеятельности общества, кардинально изменяет условия жизни и деятельности людей, их культуру, стереотип поведения, образ мыслей. Именно поэтому разворачивающийся на наших глазах процесс информатизации общества следует квалифицировать как новую *социотехническую революцию*, информационную основу которой составляет шестая информационная революция, результатом которой станет формирование на нашей планете новой цивилизации – *информационного общества* [6].

Исходя из всего вышеперечисленного можно сделать вывод о целесообразности преподавания данной дисциплины в вузе, так как наша цель – это не только подготовка высококвалифицированного специалиста, но и формирование у него необходимых знаний по использованию современных информационных технологий, возможных проблем их внедрения и использования в своей профессиональной деятельности для более органического вхождения в современную информационную среду.

Здесь можно возразить, что информационные технологии изучаются в курсе «Информатика». Но эти занятия как бы «выравнивают» знания студентов, и, потом, времени учебных занятий по информатике на гуманитарных специальностях не хватает на изучение основных возможностей наиболее распространенных программных продуктов общего назначения.

Когда же студенты имеют определенный «багаж» информационных знаний, то углубление этих знаний, закрепление их, видение новых возможностей этих знаний в своей профессиональной деятельности – это может стать целью преподавания данной дисциплины.

Информационные технологии и телекоммуникационные средства стремительно развиваются и совершенствуются, студенты технических специальностей по своей сути стараются быть в курсе последних новинок, тогда как студенты-гуманитарии в этом отстают, поэтому для последних учебных занятий на ПК никогда не будет много, будь то занятия по информатике, информационным технологиям, социальной информатике, телекоммуникационным средствам, ТАСО и т. п.

Если взять содержательную сторону учебного курса «Социальная информатика» и возможные варианты преподавания этого курса, то схема может быть следующей.

1. Можно ввести спецкурс «Социальная информатика» на специальностях социальная работа, социальная педагогика, социология, журналистика, психология и др., то есть на специальностях, затрагивающих вопросы межличностных отношений.

2. Можно ввести в курс «Информатика» отдельный раздел «Социальная информатика» на гуманитарных педагогических специальностях.

3. Также возможно в курсе ТАСО знакомить студентов с некоторыми вопросами социальной информатики, например проблемами информационного неравенства, социальными предпосылками и последствиями информатизации общества с целью форми-

рования у студентов представления об обществе, в котором им предстоит в будущем работать.

Лекционный материал может быть представлен следующими темами.

1. Предмет, цели, задачи, понятия социальной информатики.
2. Информационные ресурсы общества.
3. Формирование информационной среды общества, информационный потенциал общества, информационная культура.
4. Информатизация общества: социальные условия, предпосылки, последствия.
5. Проблемы и возможности личности в условиях информационного общества.
6. Социальная структура в информационном обществе.
7. Специфика трудовой деятельности в информационном обществе.
8. Проблемы адаптации информационных технологий в конкретной предметной области, например для юристов – компьютерная этика, компьютерное право; для журналистов – информатизация и СМИ; для психологов – проблемы взаимодействия системы «человек-машина» и т. п.

Содержание *практических занятий* в зависимости от материально-технической базы, программного обеспечения, знаний преподавателя может включать использование следующих программ:

1. Программы общего назначения в приложении к определенной предметной области. Например, для социальных работников создание электронных форм заполнения различных анкетных данных средствами текстового процессора Word; создание различных баз данных и манипулирование данными в них по работающим, пенсионерам, трудным детям и т.п. с помощью СУБД Access и т. п.

2. Программы специального назначения конкретной предметной области, например для журналистов – издательские системы, для социологов – статистические программы, для психологов – компьютерные тестирующие программы и т. п.

В предлагаемом содержании лекционного и практического материала можно увидеть некое сходство между данной дисциплиной и преподаваемыми сейчас дисциплинами, такими, как «ИТ в управленческой деятельности», «ИТ в социальной сфере» и т. п. Очевидно, что так и должно быть, поскольку основа этих знаний одинакова – переработка знаний с помощью вычислительной техники. Однако, на наш взгляд, социальная информатика имеет более глобальное значение в условиях становления информационного общества, так как позволяет проанализировать социальные аспекты внедрения информационных технологий в различные сферы профессиональной и общественной деятельности человека.

В настоящее время в нашем вузе на некоторых специальностях основы данной дисциплины постепенно вводятся. Так, на специальности «Социальная работа» социально-гуманитарного факультета введен курс «Информационные технологии в социальной работе», в лекционный материал которого вводятся некоторые вопросы социальной информатики. Несомненно, что по мере изучения и освоения данной дисциплины будут совершенствоваться содержательная и методическая стороны ее преподавания.

В заключение отметим, что дисциплина эта довольно новая и как любая новая дисциплина потребует и значительных затрат (временных, людских, материальных), и различных методических аспектов ее преподавания. И наверняка скоро появятся новые научные дисциплины, консолидирующие информатику с каким-нибудь знанием. Пока мы даже не подозреваем об этих дисциплинах, но они неизбежно должны появиться в развивающемся информационном обществе.

Примечания

1. *Колин К. К.* Социальная информатика – научная база постиндустриального общества // Социальная информатика-94. М., 1994.
2. *Колин К. К.* О структуре и содержании образовательной области «Информатика» // Информатика и образование. 2000. № 10.
3. *Колин К. К.* Фундаментальные основы информатики: социальная информатика: Учеб. пособие для вузов. М.: Академический проект; Екатеринбург: Деловая книга, 2000.
4. Там же.
5. *Колин К. К.* О структуре...
6. *Колин К. К.* Фундаментальные основы...

КОГНИТИВНАЯ ИНФОРМАТИКА

С. М. Окулов

О ПОНЯТИИ «КОГНИТИВНАЯ ИНФОРМАТИКА»

В статье обосновывается введение термина «когнитивная информатика» для обозначения новой точки зрения на обучение информатике, в рамках которой осуществляется целенаправленная деятельность по развитию интеллекта школьника. Выделены особенности программирования как особого вида деятельности, обеспечивающей эффективное формирование того, что когнитивные психологи называют ментальным опытом человека, или его интеллектом.

В предлагаемой статье рассматривается проблема (ей посвящена одна из последних работ автора [1]) – «как деятельность, связанная с изучением программирования, и в целом информатики, влияет на развитие интеллекта школьника», решение которой позволит в определенной степени уточнить цели и задачи курса информатики. Естественно, что при любом обучении, любому предмету, так или иначе, интеллект развивается, но это малопродуктивное утверждение. В нашем процессе есть компьютер – активный элемент среды обучения, особенно при работе в системах программирования. Возможна ли целенаправленная деятельность на развитие интеллекта с учетом тех результатов, которые получены в смежных науках, и потенциала самого предмета? Проблема не возникла сегодня, а является закономерным итогом процесса развития образовательной информатики. Характеризуя её нынешнее состояние, академик РАО А. А. Кузнецов утверждает: «... Главной целью образования становится формирование целостного мировоззрения, предполагающего *новый способ мышления и деятельности человека* (курсив автора). Роль изучения информатики в формировании такого мировоззрения трудно переоценить. Именно поэтому формирование научной картины мира и становится сейчас приоритетной задачей в системе задач изучения информатики в школе. Не замечать эту тенденцию или сводить мировоззренческие аспекты изучения информатики к роли информационных технологий в развитии общества (как это пытаются делать некоторые авторы) уже нельзя» [2]. Итак, требуется формировать новый способ мышления, соответственно развивать мышление (интеллект в действии), и осмысление этого положения, на наш взгляд, первоочередная задача педагогической науки и практики. Выскажем следующую гипотезу – дальнейшее развитие целей образования школьников в области информатики будет происходить, с учетом всех предыдущих достижений (и в этом заключается диалектика процесса), в направлении целенаправленного развития интеллекта школьника. Не отвергать предыдущие достижения (алгоритмическую культуру, математическое моделирование, информационные технологии и т.д.), а отрицать, использовать их в новом качестве на новом витке развития. Используя прием М. П. Лапчика, изобразим схематично эволюцию целей образования школьников в области информатики: *алгоритмическая культура* ⇒ *компьютерная грамотность* ⇒ *(информационная культура + развитие интеллекта)*.

Понятие «интеллект» как одну из предельных абстракций трудно определить. Оно имеет длительный процесс эволюции, начиная с Платона. Против рассмотрения интеллекта как некой «фиксированной величины», которую можно измерить в лабораторных условиях, возражал один из выдающихся психологов XX столетия Л. С. Выготский. Он считал, что наилучшим показателем интеллекта является то, как люди осваивают новое, а не уровень знаний, которые они накопили к определенному моменту времени. Современная точка зрения сводится к тому, что интеллект не является некой единой унитарной конструкцией. Считается, что *интеллект есть некая суперпозиция всех его многообразных форм*: сенсомоторных, образных, вербальных, знаково-символических, дискурсивных и пр. С интеллектом связаны *способности* формировать понятия, рассуждать, решать задачи (в том числе и творческие), запоминать и воспринимать (высшие формы познания человеком действительности). Однако определять нечто, через его свойства, конечно, допустимо, но, в соответствии с современной трактовкой понятия когнитивными психологами, считаем, что интеллект есть некая психическая реальность и обладает как нечто целое определенными связями, структурой, которые не определяются отдельными свойствами.

ОКУЛОВ Станислав Михайлович – кандидат технических наук, доцент, заведующий кафедрой информатики и МОИ ВятГГУ

Когнитивные психологи исследуют принципы и методы, которыми управляется феномен человеческого познания. Познание охватывает ментальные процессы, такие, как восприятие, мышление, память, оценка, планирование и организация. Когнитивная психология – один из срезов многогранного явления под названием психология. Это не теория личности как таковая, она не есть нечто единое, сцементированное, она скорее объединяет в себе множество теорий, общей базой которых является «картография» структуры интеллекта. Исследования когнитивных психологов приобрели совершенно новое звучание, после того как А. Ньюэллом и Г. Саймоном в 1958 году высказана гипотеза о том, что интеллект можно рассматривать как систему обработки информации наподобие компьютера (компьютерная метафора). Это породило лавину исследований и теоретических формулировок, основанных на компьютерных моделях. Не обошли они и отечественные издания, вплоть до последнего времени [3]. Не вдаваясь в анализ научной (и не очень) дискуссии под названием – «могут ли машины мыслить», отметим суть, а крайности – они присутствуют при обсуждении любого явления. Для того, чтобы осуществлять компьютерное моделирование работы интеллекта, требуется понять, как работает интеллект. При этом создание модели и её «прокатка» вносят новое в понимание того, что собой представляет интеллект. Это как бы два участника совместного танца, и отсутствие партнера приводит к прекращению номера, а так танец бесконечен, как, видимо, любой познавательный процесс. Главное, за последние десятилетия когнитивные психологи, используя достижения информатики, действительно продвинулись вперед в понимании природы интеллекта. А сейчас, если так можно выразиться, сделаем мысленно обратный ход, – используя результаты когнитивных психологов, можно ли понять, как деятельность в информатике, точнее, обучение информатике, оказывает воздействие, влияет на развитие интеллекта. Естественно, когнитивная психология является одним из «срезов» психологии. Понимая это и понимая, что построения когнитивных психологов в определенной степени носят редукционистский характер, следует сознательно уменьшить область рассмотрения только интеллектом и не рассматривать на данном этапе, например, теории личности в целом. Однако, так или иначе, если в результате будет построена система обучения информатике, учитывающая достижения когнитивных психологов в области понимания природы становления (развития) интеллекта, отражающая, на фундаментальном уровне, состояние настоящей информатики, то имеем ли мы право назвать эту систему «когнитивной информатикой»?

Не останавливаясь на обзоре различных теорий интеллекта (это сделано в работе автора [4]), считаем (основываясь на исследовании М. А. Холодной [5], в котором интеллект рассматривается как некая цельная психическая структура), что интеллект есть форма организации ментального (умственного) опыта. Понятие опыта трактуется не как чувственно-эмпирические формы познания действительности и не сводится к полученным знаниям, умениям, навыкам. Ментальный опыт трактуется как сформированные психические образования (структуры) человека, обеспечивающие: хранение, упорядочения и преобразования наличной и поступающей информации; осознанную и неосознанную регуляцию деятельности человека (планирование, предвосхищение, оценку, «притормаживание», выбор стратегии); выбор направления поиска при решении творческих проблем. Итак, если будет создана среда обучения для эффективного формирования ментального опыта школьника, то можно ли говорить о том, что в результате мы формируем его интеллект?

Результативность среды, её действительная эффективность, обеспечивается в том случае, если она: работает в реальном масштабе времени (временных задержек между любым действием школьника и реакцией среды на действие быть не должно); имеет дело с динамическими моделями (программа – это динамическая модель конкретной задачи); создает условия для реализации прохождения любого познавательного процесса, как последовательного восхождения по сходящейся спирали [6] (процесс отладки программ, её последовательное уточнение имитирует этот процесс).

В содержательном наполнении среды должен отражаться исторический процесс развития одной из сфер деятельности человека (название предмета) и её научного осмысления. Программирование, видимо, одна из ключевых информационных технологий. Понимание этого положения отражено в образовательных стандартах высшей школы [7], но, к сожалению, в школьной информатике ему уделяется все меньше внимания. Отметим, что программирование является одной из немногих сфер деятельности человека, в которой ему, не вставая из-за рабочего стола, приходится иметь дело со сверхсложными системами [8]. Деятельность, связанная с разработкой сверхсложных систем, принципиально отличается от деятельности, например, математика, при решении сложной творческой задачи. Отличие программы от любой даже сложной механической системы в том, что число взаимодействующих частей в программе настолько велико, что не поддается никакому разумному объяснению и проверить работу программы, перебрать все возможные способы взаимодействия её частей немыслимо даже на свербыстродействующих компьютерах в разумные строки.

Выделим особенности программирования как учебного вида деятельности (общая характеристика). Есть задача, проблема. Ученику требуется найти решение путем разработки соответствующей программы. Если решение известно, прорешивались аналогичные задачи, то задействуется ассоциативная составляющая интеллекта, работа сводится к набору

программы и её отладке. Мы рассматриваем творческие задачи. В этом случае за постановкой задачи следует гипотеза и разработка первого варианта программы. Затем она подвергается исследованию, экспериментальной проверке с помощью системы тестовых проверок – сравнению ожидаемых результатов и полученных. Ученику мысленно следует предсказать, предвидеть результаты работы. Наступает фаза или экспериментального опровержения, или экспериментального подтверждения. Итак, программирование можно рассматривать как разработку плана будущих действий по решению задачи (проблемы). Необходимо предвидеть эти будущие действия во всем многообразии возникающих вариантов. Этот тип мышления называют алгоритмическим, но это не так, точнее – это характеристика действительно интеллектуальной деятельности (творческого мышления). Алгоритмический аспект – это управление действиями (в психологии называют это процедурными знаниями). Понятие программа более широкое, чем набор управляющих конструкций. Деятельность при программировании характеризуется следующими признаками.

• *Готовность к планированию.* Два типа школьника: один при получении задания сразу «хватается за компьютер» и начинает что-то делать; второй, продумав, составив план в общих чертах, приступает к работе. «Дурные» привычки у школьников первого типа быстро изживаются, ибо постоянно приводят к отрицательному результату. Получить работоспособную программу ему удается достаточно редко. Второй тип деятельности соответствует типу деятельности профессионалов в программировании.

• *Гибкость.* Отсутствие гибкости (ригидность) и догматизм характеризуют «ограниченный ум». Гибкая позиция – это готовность рассматривать новые варианты, пытаться сделать что-то иначе, менять свою точку зрения. Программирование в своей сути обязывает не торопиться с окончательным решением, проверить программу еще при одних исходных данных, еще при одних и т. д. Программирование обязывает четко определить допустимую область значений исходных данных, при которых данный вариант программы работоспособен. Формируются качества, если так можно выразиться, открытого ума, способного подождать с вынесением суждений, собрать больше информации, прояснить для себя более сложные вопросы.

• *Настойчивость.* Отношение к решению задач, к разработке программы. Даже простая программа требует отладки. Первый тип учеников бросает доведение любой программы до работоспособного состояния, если она сразу не выдала какой-то результат, или могут исправлять только простейшие типы ошибок. Второй тип учеников получает удовольствие от процесса тестирования программы и поиска ошибок. Они обычно создают несколько вариантов программы, исследуя проблему. В процессе обучения первый тип плавно перетекает во второй, ибо этого требует среда – она требует доводить дело до конца, требует терпенья и настойчивости, ибо действительное мышление – напряженнейший труд с полной самоотдачей.

• *Готовность исправлять свои ошибки* (контролируемостью). «Два раза наступать на одни и те же грабли» – признак дурного тона в мышлении. Заниматься процессом оправдания своих ошибок бессмысленно, ибо для компьютера это не имеет никакого значения. Их требуется исправлять и не повторять. Приходится отвергать свои решения, как бы вы ни были влюблены в них. После этого, естественно, будешь относиться гибче и к мнению окружающих и к противоположным точкам зрения – искать в них рациональное зерно, то есть совершенствовать своё мышление.

• *Осознание или метапознание.* При программировании четко прослеживается, что я как действующий за компьютером знаю, что я понимаю. Без сосредоточения на собственном мыслительном процессе, на результатах собственного мышления, другими словами – на критической оценке полученных результатов, программу (решение) просто-напросто не сделать. Дидактический потенциал этапа тестирования программ просто ещё не оценен. Это один из мощнейших инструментов формирования ментального опыта школьника.

• *Поиск различных вариантов решения задач.* Это естественное качество работы программиста, ибо у каждой программы есть ограничения и она создается с использованием ограниченного инструментария. Например, изменение размерности входных данных требует, как правило, поиска других методов решения. Отметим еще одну возможность (не индивидуальную) при написании программ. Если задача решается в классе, то происходит обмен идеями, методами между школьниками. Ищется наилучший вариант решения, оценивается время его работы и т. д. Развиваются умения слушать и слышать другого, коммуникативные навыки. Практика программирования первой пришедшей на ум идеи уходит в прошлое уже через полгода работы.

Несколько слов о структурном принципе деятельности в программировании. Принцип структуризации лежит в основе любой интеллектуальной деятельности, то есть он универсален. Выскажем утверждение о том, что любая деятельность может быть описана с помощью ограниченного числа структурных конструкций, логических инвариантов этого вида деятельности. Программа обязана иметь хорошую структуру, хороший гештальт, что облегчает её понимание как сверхсложной системы и упрощает работу с ней. Мы получаем укрупнение оперативных единиц восприятия (семантически целостностных образований, обеспечивающих возможность практически одноактного восприятия объектов внешнего мира независимо от числа содержащихся в них признаков). Эта мысль, начиная с работ классиков, пронизывает все развитие технологий программирования. Исторически про-

граммирование – первый тип деятельности, к которому был применен в явном виде принцип структуризации. В чем его суть? Человеческие знания, выраженные с помощью любого письменного языка, можно разбить на две части: императивные и декларативные. Императивные (процедурные, алгоритмические, операторные) знания содержат сведения о последовательности действий. Декларативные (дескриптивные, атрибутивные, описательные) – это знания не о действиях, а об описаниях информационных объектов. Если на втором витке развития технологий программирования речь шла о структуризации императивных знаний (в основном), императивной части программы, то, начиная с объектно-ориентированных технологий, идет структуризация по данным, а затем и структуризация по интерфейсу.

Структурированная программа, её фрагменты воспринимаются как нечто целое, а не на уровне отдельных управляющих конструкций или типов данных. Элементы структуры (в частности, процедуры, функции) воспринимаются и как некий единый языковый знак, и как некий наглядный образ, и как некое действие (или действия). Развитие интеллекта, как отмечал Дж. Брунер, осуществляется по мере овладения этими тремя формами представления информации [9]. Мы уходим только от вербальной формы подачи информации (как в обычном традиционном обучении) и работаем в системе трех модальностей: через знак, через образ и на сенсорном уровне. Структурный принцип деятельности обеспечивает как улучшение понимаемости (когнитивное качество) программы, так и уменьшение интеллектуальных усилий (принцип Р. Декарта [10]), требуемых на её создание, получение результата решения проблемы. Еще один аспект. При структурном программировании достигается определенная согласованность когнитивных характеристик восприятия человеком информации и текстом программы, достигается как бы взаимная адаптация, что, безусловно, влияет на продуктивность работы интеллекта, на его развитие.

Деятельность при программировании можно назвать направленной на получение желаемого результата. Она не просто активна, она сверхактивна, и мы видим возможность реализации концепции развивающего обучения в полном объеме. Обеспечивается не только управление мыслительными процессами школьника извне, но и рациональное самоуправление познающего субъекта в процессе учебной деятельности (по Н. А. Менчинской [11]). Развивается культура внутренних процессов (по С. Л. Рубинштейну [12]). Происходит процесс осознанной саморегуляции субъекта – основа становления общих умственных способностей человека, его одаренности (по Н. С. Лейтесу [13]). Деятельность при программировании показывает школьнику, как он должен думать, а не только то, что он должен думать, хотя и это, естественно, подразумевается. Проблема (задача) постигается не через её наглядное, внешнее сходство с другими (ассоциативная теория), а через её скрытые конкретные взаимосвязи, через противоречивый путь её внутреннего развития (по Д. Б. Эльконину и В. В. Давыдову [14]), естественно, при соответствующем построении курса обучения. И, резюмируя, при обучении программированию есть возможность создать условия, обеспечивающие эффективное формирование того, что когнитивные психологи называют ментальным опытом человека.

Данная точка зрения на обучение информатике не появилась на пустом месте, она основана на конкретной, практической работе. По её результатам изданы книги [15], в которых нашло частичное отражение этой деятельности.

Термин «когнитивная информатика» введен для обозначения новой точки зрения на обучение информатике, в рамках которой осуществляется целенаправленная деятельность по развитию интеллекта школьника. Не претендуя, безусловно, на полный охват этого многогранного явления, автор надеется привлечь внимание специалистов к этому многообещающему направлению развития образовательной информатики.

Примечания

1. *Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ. 2003. 224 с.

2. *Кузнецов А. А.* О концепции содержания образовательной области «Информатика» в 12-летней школе // Информатика и образование. 2000. № 7.

3. *Сергин В.* Мозг как вычислительная система//Информатика и образование. 1987. № 6; *Усольцев А. П.* Информационная модель мышления // Информатика и образование. 2002. № 4.

4. *Окулов С. М.* Когнитивная информатика. Киров. Изд-во ВятГГУ. 2003. 224 с.

5. *Холодная М. А.* Психология интеллекта. Парадоксы исследования. СПб.: Питер, 2002. 272 с.

6. *Абдеев Р. Ф.* Философия информационной цивилизации. М.: Владос, 1994. 336 с.

7. *Окулов С. М.* Стандарты по информатике в вузе: пути совершенствования // Стандарты и мониторинг в образовании. 2002. № 1.

8. *Буч Г.* Объектно-ориентированное проектирование с примерами применения. М.: Конкорд, 1992. 519 с.

9. *Брунер Дж.* Психология познания. М.: Прогресс. 1977.

10. Р. Декарт писал в своем философском учении: «Под методом же я разумею достоверные и легкие правила, строго соблюдая которые человек никогда не примет ничего ложного за истинное» и сможет добывать новое знание — все, что он способен познать — «*без излишней траты умственных сил*». Для сферы образования это положение Р. Декарта можно сформулировать как требование минимизации умственных усилий учащегося, затрачиваемых на единицу прочно усваиваемых знаний, умений и навыков. Декарт утверждал, что в основании всех наук лежит одна и та же тождественная себе человеческая мудрость, относящаяся к разным наукам, как солнце к различным освещаемым предметам. Для образования, следовательно, было бы гораздо полезнее, чем обучать «многознанию», обратиться к обучению, хотя бы в рамках отдельных предметов, законам самой этой мудрости. На современном языке мудрость – это разум, интеллект. Так что лучше было бы сказать не на единицу знаний, умений, навыков, а на единицу развития интеллекта. Правда, «линейки» для измерения второго, вероятно, нет, если не рассматривать методы факторного анализа интеллекта, результаты которого носят относительный характер.

11. *Менчинская Н. А.* Проблема учения и умственного развития. М.: Педагогика, 1989.

12. *Рубинштейн С. Л.* Основы общей психологии. СПб.: Питер, 1999.

13. *Лейтес Н. С.* Умственные способности и возраст. М.: Педагогика, 1971.

14. *Давыдов В. В.* Теория развивающего обучения. М.: Педагогика, 1996.

15. *Окулов С. М.* Основы программирования. М.: ЮНИМЕДИАСТАЙЛ, 2002. 424 с.; *Окулов С. М.* Программирование в алгоритмах. М.: БИНОМ Лаборатория знаний, 2002. 341 с.

С. М. Окулов

МЕЖДИСЦИПЛИНАРНЫЕ АСПЕКТЫ ДИССЕРТАЦИОННЫХ ИССЛЕДОВАНИЙ В ОБРАЗОВАТЕЛЬНОЙ ИНФОРМАТИКЕ

В статье сделана попытка обоснования исчерпанности парадигмы сверхспециализации исследований. Рост числа аномалий, уход в сторону междисциплинарных работ является одним из подтверждений этого тезиса.

Э. Тоффлер, характеризуя век индустриализации (вторую волну), писал о шести основополагающих принципах, одним из которых является специализация [1], распространяющаяся на все сферы деятельности человека, включая и науку. Эта специализация – веление времени, но, может быть, пришло время «собирать камни». А. И. Ракитов [2] говорит о том, что каждая наука характеризуется определенной совокупностью объектов познания, взаимодействующих между собой, находящихся в каких-то отношениях. Эти последние – отношения, взаимодействия, преобразования – составляют суть, предмет *данной* науки и определяют круг проблем, рассматриваемых и понятных специалистам *данной* науки. Кроме того, и критерии истины определяются внутри *данной* науки, и методы исследования формулируются в *данной* области познания под хорошо сформулированные проблемы, под известный эмпирический базис *данной* науки. Получается как бы «замкнутая система», функционирующая по своим законам, по своим «правилам игры», и это есть в определенной степени следствие специализации. Рассмотрим действия исследователей в «замкнутой системе», определяемой специальностью 13.00.02 – теория и методика обучения и воспитания (информатика) по двум параметрам – методам ухода от замкнутости (междисциплинарный аспект) и критериям истинности результатов исследований. Все рассматриваемые исследования с точки зрения категориального аппарата (проблема, тема, актуальность, объект и предмет исследования, задачи, гипотеза, задачи и защищаемые положения, значение для науки, значение для практики) выполнены безукоризненно в соответствии с общепризнанной парадигмой. По Т. Куну, наука, развивающаяся в рамках общепризнанной парадигмы, является «нормальной»: « …исследования в нормальной науке направлены на разработку тех явлений и теорий, существование которых парадигма заведомо предполагает»[3] . До тех пор, пока парадигма выступает как надежный инструмент решения научных проблем, в науке происходит процесс накопления знания. Работа ученого при этом заключается в совершенствовании самой парадигмы и решении проблем, для которых существует гарантированное решение, получаемое некоторым предписанным путем. Затем начинает расти число проблем, которые нельзя решить средствами существующей парадигмы – «аномалии». Первыми признаками возникновения аномалий являются попытки уйти от замкнутости.

А. Г. Гейн [4] привлекает аппарат других дисциплин естественнонаучного цикла и считает, что в этом случае будет реализован общеобразовательный потенциал курса информатики. И. В. Симонова [5] строит курс информатики для школьников, имеющих трудности в обучении с учетом достижений в *Computer Science*, а именно, Интернет-технологий. Н. И. Пак [6] привлекает понятийный аппарат синергетики и Интернет-технологий для обоснования курсов информатики и информационных технологий. Е. В. Баранова [7] использует объектно-ориентированные технологии проектирования программных продуктов в *Computer Science* для разработки содержания обучения средствам информационных технологий. И. Б. Готская [8] исследует влияние экономических факторов на развитие методической системы обучения информатике студентов педвузов. О. Г. Смолянинова [9] привлекает аппарат мультимедиа–технологий из *Computer Science* для разработки системы формирования компетенций учителя информатики.

Если в перечисленных исследованиях попытки ухода от замкнутости достаточно очевидны, то, например, в работах А. В. Петрова [10], Е. А. Ракитиной [11] они выполняются более сложным образом. В первом случае используется аппарат не дисциплинарной методологии (как обычно), а методологии в целом как составной части гносеологии. Во втором случае используется «срез» целостной психологической структуры деятельности в педагогической терминологии для конструирования системы обучения информатике.

Возьмем отдельную проблему – развитие, становление интеллекта (мышление – это интеллект в дей-

ОКУЛОВ Станислав Михайлович – кандидат технических наук, доцент, заведующий кафедрой информатики и МОИ ВятГГУ

ствии) ребенка. В педагогике, прямо или косвенно, с ней связано огромное количество работ. О психологии говорить не приходится – от общих теорий личности до работ Ж. Пиаже и, наконец, до отдельного направления под названием «когнитивная психология», в котором эта проблема является ключевой. В *Computer Science* отрасль искусственного интеллекта связана с моделированием и исследованием закономерностей работы мозга человека. К этому перечню можно добавить логику, лингвистику, эпистемологию и т. д. Естественно, что каждая наука абстрагируется от тех его сторон, механизмов, которые не существенны с точки зрения ее предмета и не могут быть исследованы с помощью присущих ей методов анализа. Например, психология исследует главным образом общие закономерности мышления, обнаруживающиеся при решении задач самого различного содержания. При этом творческое мышление рассматривается только как частный случай проявления этих общих закономерностей. В логике создан мощнейший аппарат, позволяющий эффективно оперировать уже имеющимися знаниями, изучать механизмы выводного знания. Но соблюдение законов и правил логики не гарантирует успеха в решении многих задач, ибо решение не может быть найдено путем прямого логического вывода и не определяется совокупностью заранее заданных алгоритмов (точных предписаний мыслительных действий, ведущих от некоторых начальных данных к искомому результату). Можно продолжить, однако резюмируем. На наш взгляд, конструктивный «сдвиг» педагогической проблемы развития интеллекта может быть получен только в рамках междисциплинарных исследований. При этом упрощение проблемы до «процессов отражения внешнего мира в нашем сознании», а именно на этом постулате выстраиваются, в основном, дидактические схемы, приведет к упрощенным и непродуктивным решениям. Таким образом, разрешение проблем аномалий возможно только в рамках междисциплинарных исследований.

Остановимся на критериях истинности научных результатов. О теории говорят, что она должна быть полной и непротиворечивой, хотя, конечно, эти понятия относительны согласно основополагающим результатам (их философскому осмыслению) К. Гёделя (любая достаточно сложная теоретическая система будет, с одной стороны, неполна, с другой стороны, её непротиворечивость не может быть полностью доказана в рамках данной системы). Жизненность теории определяется ее объяснительным потенциалом (критерий реальности). Цель любой науки – «понять реальность через объяснения» [12]. Рациональная критика (экспериментальная проверка) сравнивает конкурирующие теории и выбирает ту, которая дает лучшее объяснение фактам. Например, в когнитивной психологии рассматриваются различные теории представления (репрезентации) знаний в памяти человека: теории абстракций (семантические сети, когнитивные карты и т. д.) и теории образа или примеров (хранится не какое-либо центральное понятие, как в теориях абстракций, а лишь определенные примеры, которые при деятельности оцениваются степенью близости с образцом). Несмотря на определенную диаметральную противоположность теорий, степень близости предсказаний в широком диапазоне экспериментов удивляет психологов. Таким образом, мы имеем не теории, а некие модели, гипотезы. А в педагогических исследованиях? Есть ли хотя бы одна теория или концепция развития методической системы обучения информатике, не подтвержденная экспериментом? В лучшем случае говорят, что в эксперименте идет речь о новом содержании, методах и средствах обучения, поэтому его оценки носят качественный характер. В основном пишется о констатирующем, формирующем и контрольно-оценивающем этапах эксперимента. Таким образом, рациональная критика строится на сравнении как бы обычной и разработанной теорий (концепций, систем обучения). В качестве обоснования приводятся результаты статистического анализа. Критерий реальности не работает, как нет и объяснительного потенциала, и это не следствие недостатков конкретных исследований, а исчерпанность парадигмы из-за узкой специализации последней. Из принципов научного познания работают принципы детерминизма и дополнительности, «хромает» принцип соответствия. Сведение принципа соответствия в гуманитарных исследованиях к преемственности теорий вряд ли правомочно. Например. Объяснительный потенциал теорий личности (гуманитарных исследований) Л. С. Выготского, С. Л. Рубинштейна, З. Фрейда, К. Юнга огромен, а педагогика – это теория личности в образовательном процессе, ибо, говоря «наука о воспитании и обучении подрастающего поколения», мы, надеюсь, говорим не о роботах, предназначенных для выполнения конкретных операций, а о личности.

Примечания

1. *Тоффлер Э.* Третья волна. М.: ООО «Издательство АСТ», 2002.
2. *Ракитов А. И.* Историческое познание: системно-гносеологический подход. М.: Политиздат, 1982.
3. *Кун Т.С.* Структура научных революций. М.: Наука, 1975.
4. *Гейн А. Г.* Изучение информационного моделирования как средство реализации межпредметных связей информатики с дисциплинами естественнонаучного цикла: Автореф. дис. … д-ра пед. наук. М., 2000.
5. *Симонова И. В.* Концептуальные модели обучения практико-ориентированных учащихся в условиях интернет-образования: Автореф. дис. …д-ра пед. наук. СПб., 2000.
6. *Пак Н. И.* Нелинейные технологии обучения в курсах информатики и информационных технологий: Автореф. дис. … д-ра пед. наук. Красноярск, 2000.
7. *Баранова Е. В.* Теория и практика объектно-ориентированного проектирования содержания обучения средствам информационных технологий: Автореф. дис. … д-ра пед. наук. СПб., 2000.
8. *Готская И. Б.* Методическая система обучения информатике студентов педвузов в условиях рыночной экономики: Автореф. дис. … д-ра пед. наук. СПб., 1999.
9. *Смолянинова О. Г.* Развитие системы формирования информационной и коммуникативной компетентности будущего учителя на основе мультимедиа-технологий: Автореф. дис. … д-ра пед. наук. СПб., 2002.
10. *Петров А. В.* Методологические основы моделирования системы подготовки учителя информатики.: Автореф. дис. … д-ра пед. наук. М., 2001.
11. *Ракитина Е. А.* Построение методической системы обучения информатике на деятельностной основе. Автореф. дис. … д-ра пед. наук. М., 2002.
12. *Дойч Д.* Структура реальности. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.

С. М. Окулов

КОМПЬЮТЕР КАК ИНСТРУМЕНТ СОЗДАНИЯ НЕЛИНЕЙНОЙ СРЕДЫ ОБУЧЕНИЯ

В статье развивается тезис о том, что только определенное соответствие двух нелинейных сред – интеллекта и обучения – является необходимым условием развития первого. Нелинейность среды обучения обеспечивается как содержанием, так и технологией обучения, и компьютер (система программирования) выступает как элемент создания этой среды.

Современное общество развивается в направлении возрастающей интеллектуализации Человека и Человечества. К концу XX века полностью осознана глубокая зависимость современной цивилизации от тех способностей и качеств личности, которые формируются в образовании. Именно поэтому поиск новых путей на целенаправленное развитие интеллекта школьника представляется актуальным. Развитие образовательной информатики связано так или иначе с компьютером, но роль компьютера как инструмента создания нелинейной среды обучения и развития интеллекта, со всеми вытекающими из этого положения следствиями, вероятно, педагогической наукой, не оценена, что ведет к «выплескиванию ребенка».

Истоки традиционного обучения. Первый исторический расцвет и распространение науки связаны с достижениями классической механики XVII-XVIII веков. В классической механике были выработаны определенные представления о материи, движении, пространстве, времени, причинности, развитии и т. д., согласно которым вся Вселенная (от атома до планет) виделась как замкнутая механическая система, состоящая из неизменных элементов, движущихся по законам этой науки. Законы считались универсальными и распространялись на все виды движения материи. Места случайности не было, а понятия необратимость и вероятность связывались с неполнотой знания. Любое явление имело причину и, одновременно, было причиной других явлений. Выстраивалась цепь событий (причина, действие), проходящая из прошлого, через настоящее, в будущее. Развитие и ретросказуемо, и предсказуемо. Просматривалась четкая предопределенность всех происходящих в мире процессов, что требовало поиска первопричины (бога) или «кирпичиков», открыв которые можно с точностью построить и предсказать все происходящие процессы. В научном познании эти поиски оформились как определенные мировоззренческие и методологические принципы: рационализм, детерминизм, механицизм и редукционизм.

Отражением данного понимания мира в дидактике стало появление теории немецкого философа, психолога и педагога Иоганна Фридриха Гербарта (1776 – 1841). Это классический пример авторитарной педагогики, в которой обучаемый рассматривается в качестве объекта внешнего воздействия и тщательно разрабатывается вся система средств управления ребенком, жестко регламентируются все действия (учителя и ученика). Философское осмысление, обоснование авторитарной системы обучения есть и в работах таких классиков мысли, как И. Кант (дисциплина – это средство уничтожить в человеке дикости…), Г. Гегель. Так, Г. Гегель, например, утверждал и обосновывал необходимость подавления и отчуждения личности ради высших государственных целей и считал, что не только формальная школьная наука, отчуждающая личность от самой себя, но и военная муштра способствует развитию духа, так как противостоит природной лени и вынуждает с точностью выполнять чужие распоряжения.

Итак, механистический взгляд на природу вещей требует наличия универсальных законов и истин, которые Человек определяет с помощью опыта, эксперимента, специальных методов, и знание в этом случае есть сумма истин, определяемых наукой и подлежащих, в определенной интерпретации (содержание образования) усвоению в образовании. Это исходное положение определяет и все принципы организации и структуризации образовательного процесса. Не останавливаясь подробно на характеристике традиционного обучения [1], выделим основные моменты, требуемые нам для дальнейшего обсуждения.

Накапливая (аддитивно) знания, развивая науку и технику, Человек обеспечивает господство над природой с целью удовлетворения своих потребностей (логика мышления техногенной цивилизации). А в дидактике? Ученик накапливает знания (основы наук), умения, навыки (ЗУН), тем самым осваивая предмет и развивая рациональное, преимущественно логико-вербальное, мышление. И, что поистине удивительно, как бы параллельно образованию идет процесс

ОКУЛОВ Станислав Михайлович – кандидат технических наук, доцент, заведующий кафедрой информатики и МОИ ВятГГУ

воспитания. Во времени этот процесс (мы рассматриваем изучение отдельного предмета) выглядит так, как показано на рис. 1.

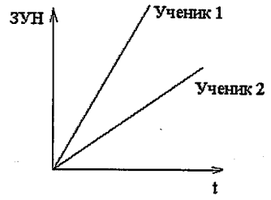


Рис. 1. Аддитивная зависимость формирования знаний, умений, навыков

В традиционной дидактике основное внимание обращается на логическое построение изучаемого материала (принцип систематичности и последовательности), отражаемого и в структуре учебников, и в структуре занятий. Это приводит к достаточно жесткой регламентации деятельности учителя, выражаемой, например, в формальном вопросно-ответном методе обучения. Цели управляющих воздействий, ибо бесцельного управления не бывает, определяются из исходных установок, и учитель, как правило, становится составной частью учебной машины, её передающим устройством. Схема взаимодействия участников образовательного процесса при традиционном обучении показана на рис. 2. Общая характеристика связей. Тип (А) описывает действия учителя и ту информацию, которую ученик «черпает» из учебника. Тип (Б) определяет то, что из полученной информации влияет на ученика как на личность, на развитие его интеллекта. Тип (В) – обратная связь от ученика к учителю при решении творческих, интеллектуальных задач. Тип (Г) – обратная связь от ученика к учителю при обычных проверочных мероприятиях опросно-ответного метода обучения. Тип (Д) определяет ту часть входных воздействий, которая проходит мимо ученика, никак не влияя на его развитие и не затрагивая его развития. Рассмотрение ученика как объекта и как субъекта основано на известной истине о том, что знание не пересаживаемо из головы в голову. Это следует, как отмечал М. К. Мамардашвили, «в силу простого обстоятельства: никто не может вместо другого ничего понимать, понять должен сам, и, более того, если уже не понял, то вообще не поймешь сообщаемое, понять можно

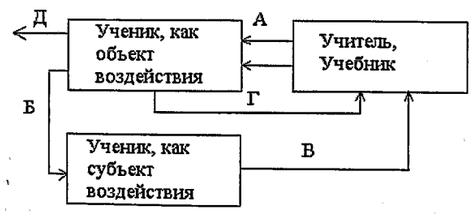


Рис. 2. Схема взаимодействия участников образовательного процесса

лишь то, что уже понял. И этот акт понимания «самим» не выводим ни из какой цепи обусловливания этого понимания, он должен совершиться или не совершиться, т. е. знание не перекачиваемо в другую голову, как в некую пустоту перекачивалась бы жидкость. Я могу пройти максимально далеко, максимально сузить воронку, внутри которой должен вспыхнуть акт понимания, но акт понимания – он должен вспыхнуть, и он не выводим из всего того, как я суживал воронку» [2].

Пропуская влияние квантовой теории на мировоззренческие и методологические принципы, в частности, на трактовку (или интерпретацию) принципа дополнительности в дидактике, перейдем на краткую характеристику синергетического взгляда на мир, сформировавшегося во второй половине XX века. Он появился в результате изучения большинством фундаментальных научных дисциплин самоорганизующихся и саморазвивающихся систем, ибо, в действительности большинство существующих в природе и в обществе систем относятся к этому классу и являются открытыми. Между системами происходит постоянный обмен энергией, веществом, информацией, и, вследствие этого, их характеризует изменчивость, стохастичность. Согласно терминологии И. Пригожина, все системы содержат подсистемы, которые постоянно флуктуируют. В некоторые моменты времени отдельная флуктуация, или их комбинация (действия ученика, учителя, компьютера. – *С. О.*), может оказаться настолько сильной, что существующая прежде организация (уровень развития интеллекта. – *С. О.*) не выдерживает и переходит в новое состояние, на новый, более высокий уровень упорядоченности (организации). Для каждого ученика этот переломный момент (точка бифуркации) индивидуален и её точное предсказание вряд ли возможно. Понятие нелинейности ассоциируется с определенным видом математических уравнений, содержащих искомые величины в степенях выше первой, или коэффициенты, зависящие от свойств среды, с помощью которых описывается поведение системы. Отсюда суть – множеству решений нелинейных уравнений соответствует множество путей эволюции системы, описываемой этими уравнениями. «Открытые нелинейные системы – это системы, способные к самоорганизации… Нелинейные системы (или состояния систем) – это нестабильные, неустойчивые, хаотизированные системы (или состояния), готовые к качественным изменениям и структурным перестройкам» [3]. Практически все существующие системы являются нелинейными и открытыми. Синергетический взгляд на мир (он не дает готовых рецептов для прямых действий и не является «инструментом для получения заданных результатов, а служит дверью, открытой в природную и человеческую реальность, от которой следует ждать ответа» [4]) характеризуется следующими идеями (принципами): о нелинейности, открытости и неравновесности систем; о конструктивной роли хаоса; о

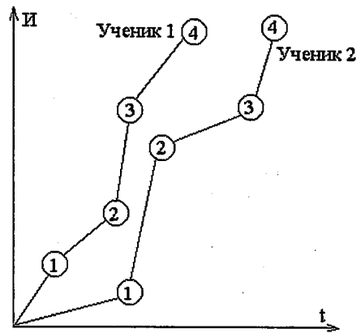


Рис. 3. Нелинейная зависимость развития интеллекта

значении устойчивости и неустойчивости, необходимости и случайности; о невозможности полного и точного прогноза; о резонансном воздействии и др.

Позволит ли ввод компьютера в обучение организовать среду, действующую по нелинейным принципам? Вопрос остается пока открытым, но прежде определим целевую установку. Нам хотелось бы создать среду по целенаправленному развитию интеллекта (И) так, чтобы каждый ученик «шел» по своей экспоненциальной траектории индивидуального развития – рис. 3 (в кружочках обозначены аттракторы [5] среды обучения). Не останавливаясь на обзоре различных теорий интеллекта (это сделано в работе автора [6]), считаем (основываясь на исследовании М. А. Холодной [7], в котором интеллект рассматривается как некая цельная психическая структура), что интеллект есть форма организации ментального (умственного) опыта. Понятие опыта трактуется не как чувственно-эмпирические формы познания действительности и не сводится к полученным знаниям, умениям, навыкам. Ментальный опыт трактуется как сформированные психические образования (структуры) человека, обеспечивающие: хранение, упорядочение и преобразование наличной и поступающей информации; осознанную и неосознанную регуляцию деятельности человека (планирование, предвосхищение, оценку, «притормаживание», выбор стратегии); выбор направления поиска при решении творческих проблем. Главное то, что интеллект есть сложная, самоорганизующая и саморазвивающая система, действующая по синергетическим принципам, и, создавая нелинейную среду обучения, мы тем самым достигаем желаемого, а именно, *соответствия между двумя нелинейными средами*.

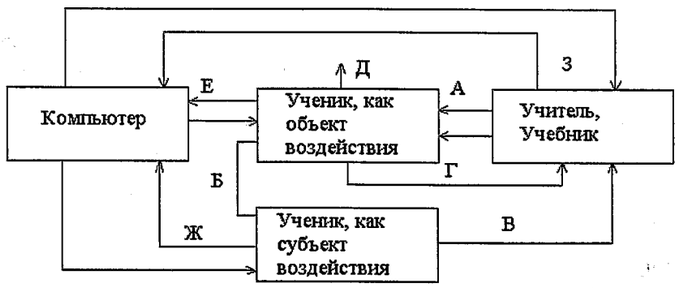


Рис. 4. Схема взаимодействия участников образовательного процесса

Ввод компьютера в процесс обучения изменяет существующие связи (рис. 4) между участниками образовательного процесса, однако для создания нелинейной среды обучения этого недостаточно. Компьютер есть инструмент, и только. С его помощью следует изменить принципы, содержание, технологию (формы, методы, средства, контроль и оценку) обучения, а это уже дидактика [8]. В этом случае как традиционные связи А – Д, так и новые Е – З (рис. 4), имеют другой смысл и другую нагрузку. И не только. Требуется отказаться от традиционного понимания управления учеником в ходе образовательного процесса, т. е. связи на рис. 4 есть связи не по управлению, а скорее по обмену информацией. Следует «говорить не об управляемом, а о направляемом развитии, полагая, что наши воздействия способны лишь обеспечить желаемые тенденции или избежать тех или иных подводных камней, которые способны увести в сторону поток развития событий» [9]. Дополняя это положение, сошлемся на мысль великого педагога Г. Нейгауза о том, что «таланты воспитать нельзя, можно создать только среду для произрастания талантов».

Несколько слов о наполнении нелинейной среды обучения (обсуждение технологии – предмет отдельного разговора) – это программирование [10]. Естественно, что содержание может быть и другим. Выбор программирования обусловлен следующими факторами.

• Анализ развития технологий программирования показывает, что в рамках этого вида деятельности человека накоплен уникальный опыт решения сложнейших проблем (мы имеем, может быть, единственный школьный предмет, когда ребенок, не вставая из-за рабочего стола, имеет дело со сверхсложными системами), при этом процесс разработки любой программы носит нелинейный характер.

• Совершенствование технологий программирования соответствует диалектике развития любого сложнейшего процесса; к технологиям про-

граммирования применимо представление как о сложной открытой нелинейной самоорганизующейся системе, при этом самоорганизация (развитие системы) происходит в направлении уменьшения хаоса в процессе разработки программ.

• Технологии программирования в процессе своего развития следовали критерию Р. Декарта – с наименьшими затратами, согласно данной конкретной технологии получить качественный программный продукт. Каждый виток характеризуется: совершенствованием метода анализа решаемых задач; уровнем абстрагирования как в создании программного продукта, так и в доказательстве его работоспособности; изменением методов анализа и синтеза программ.

Итак, логика выбора наполнения заключается в том, что среда не может быть нелинейной, если процесс деятельности с её содержимым подчиняется линейным законам, если традиции работы с этим наполнением носят линейный характер. Деятельность в программировании (процесс разработки программы) изначально носила нелинейный характер. Возможно ли другое наполнение? Вероятно, да, но это требует пересмотра большинства дидактических установок в преподавании предмета и, в любом случае, без компьютера, точнее, без создания сложных компьютерных сред, это вряд ли возможно.

Далее, говоря о содержании, следует упомянуть о том, что человек, согласно исследованиям когнитивных психологов, помнит только «суть», оперирует только сущностной основой явлений, объектов, а не деталями. Поэтому строить образовательный процесс следует в окрестностях фундаментальных понятий [11] (они на рис. 3 обозначены кружочками с цифрами – первое, второе и т. д.), при этом понятия не даются как прописные истины последней инстанции (мы только «сужаем воронку»), их осознание достигается самостоятельной деятельностью в среде. Перечень понятий должен быть ограничен и определяется исходя из логики развития изучаемого предмета. Именно они являются аттракторами (точнее, прохождение по ним), сформированность которых позволяет самоорганизовываться системе. Оформление их в устоявшиеся структуры не достигается одноразовым прохождением по темам. Процесс является последовательным восхождением по сходящейся спирали [12]. Витки отличаются уровнем сложности задач, задачным (проблемным) наполнением. Например, рассмотрим понятие – «отношение порядка». Первый виток – отношение порядка на множестве целых чисел. Второй виток – упорядоченность элементов массива (алгоритмы сортировки). Третий виток – упорядоченность состояний в пространстве решений переборной задачи и т. д.

Возвращаясь к рис. 4, дадим общую характеристику взаимодействия участников образовательного процесса. 4. Воспользуемся образом слоя [13], в каждой точке соприкосновения которого происходит обмен информацией. В нашей среде три слоя: ученик - учитель, ученик – компьютер и учитель – компьютер. Слой «ученик – компьютер» (связи Е, Ж). Отметим её многогранность. Экран один, клавиатура одна, а точек соприкосновения, обмена информацией много, особенно в режиме тестирования программы, когда оказывается, что, казалось бы, работающая программа решает задачу не при всех исходных данных. Простая рекурсивная процедура решения задачи о ханойских башнях [14], помещающаяся на экране, или вывод результатов её работы вызывает огромные «пласты» рассуждений, ассоциаций и т.д. Слой «учитель – ученик» (связи А, В, Г) не ограничивается единичным взаимодействием. Изложение нового материала не главная составляющая занятия. Учитель выступает и как эксперт, и как помощник, и как воспитатель (ученик не справляется с проблемой, ему требуется помощь, его нужно просто или поддержать, или «побить»). Его роль в этой среде многогранна, если коротко формулировать суть, то он работает по «принципу кормчего» [15], по принципу обеспечения направленного развития. Слой «учитель – компьютер» (связь З). Учитель «вкладывает» в компьютер через задачи различное содержание, различные реакции компьютера на воздействия учеников. Учитель, за счет особой организации связей А, В, Г, З, определяет и поддерживает ситуации «разрастания малого», он как бы запускает ученика «в разнос», поддерживая эффект «напряженной потребности» (создает объемную положительную обратную связь). Ученика не удовлетворяет достигнутое, устойчивости нет, он постоянно входит в режим очередного обострения при решении очередной задачи. Достигается индивидуальное резонансное воздействие [16] на каждого ученика, выбирается тот камень, который вызывает лавину. «По всей видимости, режимы с обострениями играют не последнюю роль в функционировании мозга, в разрушении и образовании структур на различных его участках… А, следовательно, здесь, на наш взгляд, необходимо искать объяснение механизмов ускорения в процессе повышения эффективности обучения» [17].

Требование определенного соответствия двух нелинейных сред – интеллекта и среды обучения, необходимое условие развития первого. Нелинейность среды обучения обеспечивается как содержанием, так и технологией обучения, и компьютер (система программирования) выступает как инструмент создания этой среды.

Примечания

1. См., напр.: *Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ, 2002. 224 с. Заметим, что речь ни в коем случае не идет о том, чтобы отвергать традиционное обучение. Речь идет об отрицании, то есть о другом витке его развития – и только.
2. *Мамардашвили М. К.* Эстетика мышления. М.: Московская школа политических исследований, 2000.
3. *Назарова Т. С., Шаповаленко В. С.* Парадигма нелинейности как основа синергетического подхода в обучении // Стандарты и мониторинг. 2003. № 1.
4. *Князева Е. Н., Курдюмов С. П.* Синергетика как новое мировидение: диалог с И. Пригожиным // Вопросы философии. 1992. № 12.
5. Аттрактор – это относительно конечное, устойчивое состояние системы, которое как бы притягивает к себе все множество «траекторий» движения (развития) системного объекта.
6. *Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ, 2003. 224 с.
7. *Холодная М. А.* Психология интеллекта. Парадоксы исследования. СПб.:Питер, 2002. 272 с.
8. *Хуторской А. В.* Современная дидактика. СПб.: Питер, 2001. С. 21.
9. *Моисеев Н. Н.* Универсальный эволюционизм (позиция и следствия) // Вопросы философии. 1991. № 3.
10. Представление о программировании, как о кодировании на одном из языков, соответствует, на наш взгляд, представлениям о том, что Солнце вращается вокруг Земли.
11. Более подробно логика выбора фундаментальных понятий описана в работе автора (*Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ, 2003).
12. *Абдеев Р. Ф.* Философия информационной цивилизации. М.: Владос, 1994. 336 с.
13. *Рыжаков М. В.* Образование как сложная открытая нелинейная самоорганизующаяся система // Стандарты и мониторинг в образовании. 2000. № 1.
14. *Окулов С. М., Веснин Р. А.* Задача о ханойских башнях (о неисчерпаемости задач) // Информатика и образование. 2003. № 10.
15. *Моисеев Н. Н.* Универсальный эволюционизм (позиция и следствия) // Вопросы философии. 1991. № 3.
16. Вспомним, казалось бы, абсурдную мысль Лао-цзы, основателя даосизма, о том, что тихое побеждает громкое, мягкое побеждает твердое, слабое побеждает сильное.
17. *Назарова Т. С., Шаповаленко В. С.* Парадигма нелинейности как основа синергетического подхода в обучении // Стандарты и мониторинг. 2003. № 1.

Е. А. Васенина

ОБРАЗОВАТЕЛЬНАЯ ИНФОРМАТИКА И РАЗВИТИЕ ИНТЕЛЛЕКТА

В статье обосновывается факт слабого использования образовательного потенциала информатики, особенно в части развития интеллекта ученика. Развитие интеллектуальных способностей в школе возможно и необходимо. С этой точки зрения следует посмотреть на преподавание информатики, поскольку, несомненно, ее изучение влияет на развитие ума.

В настоящее время вряд ли кто-то рискнет публично усомниться в том, что информатика как общеобразовательный предмет оказывает значительное влияние на развитие мыслительных способностей учащихся и что в наибольшей степени такому развитию способствует обучение разработке алгоритмов и программ. Действительно, с одной стороны, развитие мышления школьников заявлено как вторая из трех основных целей изучения информатики в школе [1]. С другой стороны, на уровне деклараций роль алгоритмизации и программирования в школьном курсе информатики видится именно в формировании алгоритмического, или структурного, стиля мышления, для которого характерна способность к планированию деятельности, к ее структурированию, т.е. выделению отдельных самостоятельных блоков, к прогнозированию возможных результатов деятельности и т.д. Однако происходит это именно на уровне деклараций, поскольку в практическом преподавании информатики преобладает пользовательский подход. Это проявляется в следующем.

1. Основное время и силы ученика направлены на изучение информационных технологий, а точнее, прикладных программных средств, их реализующих (в подавляющем большинстве, Microsoft Office). Обучение разработке алгоритмов и программ отодвигается на задний план как по времени, на него отводимому, так и по содержанию. Учеников знакомят с базовыми алгоритмическими конструкциями, упоминают о подпрограммах (работа с которыми является краеугольным камнем структурного программирования, а значит, и структурного мышления) и практически не рассматривают структуры данных, даже столь важную для понимания основ информатики, как массив. Хотя работа именно со структурами данных способствует формированию таких мыслительных процессов, как анализ, синтез, абстрагирование.

2. В работе с информационными технологиями преобладает вовсе не обучение решению с их помощью тех или иных задач (т.е. структурированию данных и действий). Акцент делается на освоение интерфейса, работу со средой того или иного программного средства. Это ярко иллюстрирует вопрос одного из тестов, призванных административно проверить уровень усвоения основ информатики – «Что отображается в строке состояния табличного процессора Excel?» Такой утилитарный подход отнюдь не способствует активизации мышления, а ориентирует на запоминание конкретных фактов, которые, по большому счету, можно найти в любом справочнике.

3. В изучении раздела «Устройство ЭВМ» немалое число учебных пособий нацеливает учителя и учеников не на исследование идей и принципов работы основных функциональных блоков компьютера, а на их конструктивное воплощение. Вопрос «Как и почему оно работает?» подменяется вопросом «Из чего оно сделано?». Но «сделано» оно на разных этапах из разного. И если утверждать, что главной частью компьютера является системный блок, то надо забыть времена, когда системного блока не было, а компьютеры были, и постараться не думать о будущем, когда системного блока, возможно, не будет, а

ВАСЕНИНА Елена Александровна – кандидат педагогических наук, доцент кафедры информатики и МОИ ВятГГУ

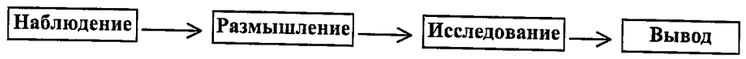


Рис. 1

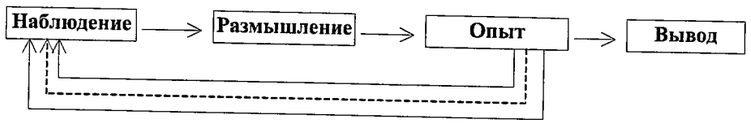


Рис. 2

компьютеры, опять-таки, будут. То есть вновь конкретное знание преобладает над формированием способности к абстракции, к выявлению общих закономерностей.

4. Изучение фундаментальных вопросов информатики (информация, ее представление и измерение, системы счисления, информационные процессы, информация и управление и др.) ведется в теоретическом ключе по принципу «выслушал учителя – выучил параграф – ответил (или не ответил)». На это настраивает учебник. К этому привыкли учителя и ученики. Так устроена традиционная школа. Хотя в свое время предполагалось, что уроки информатики, их проведение с использованием компьютера могут изменить традиционные подходы к преподаванию, приблизят обучение к реальному познанию, когда «новый материал» не «излагается» а исследуется учеником в ходе компьютерного эксперимента, результаты которого обобщаются в совместном обсуждении с учителем.

Однако мощный образовательный потенциал информатики весьма в малой степени реализуется в практическом преподавании информатики, особенно в части развития интеллекта ученика. И учителей можно понять. С одной стороны, их к этому ведут органы, управляющие образованием, как на уровне страны, когда на всю (!!) школьную информатику федеральный компонент учебного плана отводит 68 часов в старших классах – время, за которое можно выучить слова Windows, Word и Excel (не всегда понимая разницу), научиться худо-бедно набирать текст и отличать системный блок от всего остального, так и на уровне города, когда чиновники бдительно следят, чтобы учитель не переборщил с программированием в ущерб Word и Excel. С другой стороны, родители (да и дети) желают, чтобы в школе учили «чему-нибудь практическому». Когда ребенок набрал текст особым шрифтом и украсил его картинкой – это понятно, «умеет» чадо «на компьютере», глядишь, и на престижную работу устроится. А то, что ребенок чуть-чуть иначе думает, быстрее ориентируется, дальше видит – этого не заметно, ничем не измеришь, не потрогаешь и друзьям не покажешь.

Из сказанного следует, что связи между информатикой (и в особенности, программированием), образовательной деятельностью и развитием интеллекта необходимо хотя бы обсуждать, ибо сказано «толците – и отверзется».

Образование, как среднее, так и высшее, можно рассматривать как подготовку к умственной, интеллектуальной деятельности. А это работа с информацией. В то же время сам предмет науки информатики – закономерности протекания информационных процессов в системах различной природы, их реализация посредством компьютера как инструмента автоматизации интеллектуальной работы.

Человек пытался в работе компьютера в той или иной степени моделировать работу своего мозга, пытался понять, как мозг работает с информацией, чтобы создать аппарат, автоматизирующий эту работу. Но возможно и обратное – моделирование информационных процессов на компьютере позволяет лучше понять, как работает наш собственный мозг. Стремление к созданию искусственного интеллекта позволяет исследовать феномен интеллекта «естественного». Недаром Computer Science дала мощный толчок развитию когнитивной психологии – психологии познания, психологии интеллекта.

Возникает вопрос: если человек занимается (в школе или вузе) проблемами работы с информацией – исследует способы поиска информации, ее представления, хранения и передачи, наконец, обработки и использования – становится ли он от этого умнее? Эффективнее ли работает наш мозг оттого, что мы знаем и применяем на практике методы эффективной работы с информацией?

Обратимся к модели процесса познания, который можно схематично представить как цепочку (см. рис. 1).

Исследование здесь не что иное, как проведение опыта и наблюдение за его результатами. Причем, опыт проводится многократно, и по результатам данного опыта принимается решение о том, каким должен быть следующий опыт. То есть в схеме появляется обратная связь (см. рис. 2).

Если суметь реализовать такую схему и в процессе обучения, это непременно должно повлиять на формирование способности к мыслительной деятельности.

Рассмотрим пример. Некий школьник, углубленно изучающий английский язык и литературу, влюбился. В этом возвышенном состоянии он столкнулся с сонетами Шекспира, которые были созвучны душе и произвели сильное впечатление. Накануне наш школьник изучал английскую литературу дошекспировского периода и вдруг заметил некую существенную разницу (возможно, волнение сердца сделало его внимательнее). Возникло ощущение, что в стихах Шекспира использовано больше слов. Любознательный школьник решил проверить, так ли это. Исследование состояло в подсчете слов, используемых Шекспиром и его современниками.

Посмотрим, изменится ли интеллектуальный багаж школьника, если он захочет автоматизировать свою работу и попытается создать программу подсчета слов в текстах различных авторов. Очевидно, для этой цели ему потребуется мощное знание о структуре текста, его грамматике, синтаксисе и т. д. Слова по составу придется разбирать не по требованию учительницы русского языка, а для дела. Следовательно, опыт, исследование становится богаче, интереснее, и исследователь, который составлял программу работы с текстом, выиграл больше, чем тот, кто просто вручную считал слова.

Лучший способ чему-то хорошо научиться – научить этому другого. Составляя программу, человек учит компьютер выполнять интеллектуальную работу, учит думать. Можно предположить, что при этом он сам «обучается думанию», развивает интеллект, обогащает свой ментальный опыт, если воспользоваться термином, введенным М. А. Холодной в ее исследовании, посвященном психологии интеллекта [2]. В дальнейших рассуждениях будем опираться на идеи, высказанные в данной книге. В частности, вслед за М. А. Холодной зададимся вопросом, нужен ли для жизни человеку интеллект. Хорошо ли быть очень умным? И ответ не будет однозначным.

С одной стороны, безусловно, да.

Это нужно государству, ибо интеллектуально одаренные люди обеспечивают развитие наукоемкого производства, создают интеллектуальную собственность, которая становится ведущей формой собственности в современном мире.

Это нужно обществу, ибо интеллектуально развитый человек более успешно противостоит низменным инстинктам, агрессии, бездуховности.

Это нужно самому человеку, ибо интеллект обеспечивает самостоятельность мысли, позволяет противостоять внешним влияниям, гарантирует личную свободу и, в определенной мере, судьбу.

С другой стороны, вовсе нет.

Государству сложно осуществлять свои функции по организации жизни граждан, если граждане обладают большим интеллектуальным потенциалом и сами в состоянии организовать свою жизнь. Это предъявляет на порядок более высокие требования к уровню государственных управленческих решений. Врачи знают: чтобы лечить врача, нужен очень компетентный врач. Отсюда двойственное отношение к умному на всех уровнях руководства: вроде бы нужен, но потенциально опасен.

«С умным хлопотно, с дураком плохо» (Б. Ш. Окуджава).

На бытовом уровне все еще сложнее. Часто умный человек вызывает неприязнь, раздражение, порой даже тем, что в силу своей самодостаточности никак на эту неприязнь не реагирует.

И даже психологи, которые долгое время понимали интеллект как способность к решению специальных задач (интеллектуальных тестов IQ), в определенный момент поставили под сомнение само понятие интеллекта, ибо весьма часто оказывалось, что обладатели высокого IQ испытывают проблемы в обыденной жизни и даже профессиональной сфере, особенно если она связана с творчеством.

Такое «легкомысленное» отношение к интеллекту со стороны науки, общества, человека чревато *функциональной глупостью*, которую М. А. Холодная определяет как рост в общей массе людей числа лиц со средним и низким уровнем интеллектуальных возможностей и которая имеет под собой большое количество как генетико-биологических, так и социально-экономических причин. Если не противостоять неблагоприятным факторам, ситуация может стать необратимой, и цена, которую заплатит общество, будет очень высока – страна может просто не вписаться в классификацию японских футурологов, которые выделили три группы стран, в зависимости от того, что они могут предложить на мировом рынке: I – идеи и технологии, II – наукоемкую технику, III – машины, пищу, сырье. В какую группу попадет страна, которая сможет предложить только дешевую рабочую силу, к тому же не обремененную интеллектом?

В ряду социально-экономических факторов, обусловливающих феномен функциональной глупости, для нас особую важность представляет снижение уровня образования, поскольку здесь есть реальная возможность существенно его нейтрализовать. Сейчас уже стало достаточно ясно, что российская (в прошлом, советская) система образования с ее приматом фундаментальных знаний и ставкой на способность к решению проблем в противовес знаниям «практически полезным» есть признанная в мире ценность, которую следует всемерно хранить, лелеять и беречь, осторожно избавляясь от некоторых явных недостатков: формализма, оценкомании, авторитаризма.

Если принять как руководство к действию кажущуюся парадоксальной идею, что *жизнь страны определяют троечники*, становится ясным, что необходимо всемерно поднимать планку среднего уровня развития, интеллектуального в том числе. Наши дети

очень разные по своим интеллектуальным ресурсам, но они равны с позиции права на развитие интеллекта по максимуму этих возможностей.

Развитие интеллектуальных способностей в школе возможно и необходимо. С этой точки зрения следует посмотреть на преподавание информатики – поскольку ее изучение влияет на развитие ума, надо использовать этот ее образовательный потенциал.

Если человек интеллектуально воспитан, меняется характер его познавательного отношения к миру: то, как он *воспринимает*, *понимает* и *объясняет* происходящее. М. А. Холодная выделяет ряд показателей интеллектуальной зрелости личности. Рассмотрим, как обучение информатике, и в особенности программированию, может воздействовать на эти показатели.

1. Широта умственного кругозора (в противовес «закапсюлированному сознанию»).

Здесь важен мировоззренческий аспект информатики – умение видеть общее в разнородном: например, единство процессов управления в системах различной природы, возможность использования одной и той же структуры данных для описания вроде бы несхожих объектов. Важно также умение видеть явления с неожиданной стороны: например, возможность использования отрицательных цифр в алфавите системы счисления.

2. Гибкость и многовариантность оценок происходящего (в противовес «черно-белому» сознанию).

Это качество в особенности и проявляется, и воспитывается в процессе разработки программ. Обучение программированию воспитывает готовность к поиску новых вариантов решения, пригодных в различных ситуациях, готовность к обмену идеями с другими учениками и учителем, к восприятию «чужого» решения. Важным в этом смысле является отношение к ошибке как к естественному этапу в процессе познания и, следовательно, готовность к ее поиску и исправлению. Своеобразным воспитывающим началом здесь выступает сам компьютер. С ним бесполезно спорить и кричать, что ты прав, – программа просто не работает.

3. Готовность к принятию необычной, противоречивой информации (в противовес догматизму).

Поскольку компьютер обрабатывает данные по законам формальной логики, результаты работы программы в определенных условиях могут противоречить нашим обычным представлениям. Например, при сложении положительных чисел может получиться отрицательное число и т. п. Чаще всего это происходит на границе диапазонов, определяемых типами данных. Опыт преодоления подобных трудностей приучает не только искать разумное объяснение необычным, на первый взгляд, явлениям, но и находить пути их применения в полезном для своих задач направлении.

4. Умение осмыслить происходящее одновременно с позиций прошлого (причин) и в терминах будущего (последствий) – в противовес склонности мыслить в категориях «здесь и теперь».

В программировании важнейшую роль играет умение просчитывать варианты, предвидеть последствия принятых решений, прогнозировать работу программы в различных ситуациях, тестировать программу, подбирая наборы данных, моделирующие возможное поведение пользователя и продумывая реакцию компьютера на все его непредвиденные действия. Этому можно учить детей, предлагая не только решение дидактических или даже эвристических задач, но разработку проектов «под ключ» с возможным использованием созданного продукта на практике, например в преподавании других предметов.

5. Ориентация на выявление существенных, объективно-значимых аспектов происходящего (в противовес субъективированной, эгоцентрической познавательной позиции «мне это не надо»).

Ситуация «мне это не надо» довольно часто возникает на уроках информатики. Особенно это относится к программированию и другим темам, сложным для понимания и не имеющим очевидного практического выхода. Укоренению подобного взгляда способствует малоразумный «интерфейсный» подход к изучению информационных технологий – вот оно, тожество практической пользы. Но на поверку и практической пользы оказывается немного. Интерфейс, технические приемы управления программным средством освоены, а задачу с его помощью решить не получается – нужен иной подход, теория, «ненужное» фундаментальное знание. В то же время теория, преподнесенная как знание, данное свыше, как правило, кажется ненужной, а вот самостоятельно добытая в процессе решения проблем и преодоления трудностей осознается глубоко, помнится долго и применяется всегда.

Иногда студенты жалуются, что они много времени уделили программированию, а текст набирать не научились, а вот для курсовой требуется… Помочь им увидеть целостную картину и правильно выбрать приоритеты можно вопросом, за какое, примерно, время они самостоятельно освоили текстовый редактор в объеме, достаточном для набора курсовой, и удалось ли бы им столь же самостоятельно освоить азы программирования, если бы вместо разработки программ, планирования и структурирования действий и данных им долгое время преподавали тонкости компьютерной верстки текстовой информации, а в качестве курсовой предложили создать несложную обучающую программу по типу вопрос-ответ. Такая перспектива обычно вызывает улыбки, показывающие, что студенты верно оценивают значимость фундаментального и прикладного знания и с интеллектом у них все в порядке.

В преподавании информатики, где велик соблазн изучать то, что требуется «здесь и сейчас», особенно важно соблюдать баланс, отдавая приоритет вечному перед сиюминутным, и постоянно объяснять, мотивировать, доказывать правильность этого выбора перед учениками.

6. Склонность мыслить в категориях вероятного «как если бы» (в противовес игнорированию невозможных событий).

С такими событиями ученики сталкиваются достаточно часто при разработке программ. Например, вполне возможно, что корень квадратный из четырех при вычислении не окажется равным двум. Нестандартный взгляд на вещи заметно помогает разработке программ. Например, строка трактуется как массив символьных переменных, но в то же время ее можно рассматривать как целостный объект и обрабатывать соответственно. Можно число превратить в строку и работать с ним, как со строкой (учитывая, безусловно, все возможные последствия). Порой ученики придумывают столь интересные, нестандартные и эффективные решения, что учитель надолго задумывается над ними. Однако это происходит не очень часто, а чтобы произошло хотя бы когда-нибудь, учителю не следует с порога отвергать самостоятельное изобретение ученика, каким бы корявым оно ни выглядело. После двух десятков корявых решений может возникнуть изящнейшее в своей необычности.

7. Способность мысленно видеть отдельное явление в контексте его целостных связей с множеством других явлений (в противовес однолинейному взгляду на мир).

В основе любой интеллектуальной деятельности лежит принцип структурирования – он универсален. Именно его освоению, включению в состав ментального опыта в огромной степени способствует обучение структурному программированию. Именно такой подход позволяет сначала увидеть целое как совокупность связанных, достаточно крупных частей, затем подробнее рассмотреть, детализировать эти части, выделив в их составе, в свою очередь, некую сложную структуру, и так до полного решения задачи.

Изучение новых технологий программирования (объектно-ориентированное и визуальное программирование) позволяет детям научиться рассматривать объект во всем многообразии его связей и взаимодействий с окружающей средой.

На формирование данного качества ума влияет изучение информатики в целом, поскольку для этой дисциплины характерны очень мощные внутрипредметные связи, когда вновь изучаемое понятие не только опирается на ранее изученные, но в то же время позволяет взглянуть на них в новом ракурсе, добавить новые черты к их содержанию. Следует только учитывать это обстоятельство при выборе методики.

Обобщая сказанное, можно сделать следующие выводы.

• Изучение информатики дает понимание принципов работы с информацией, т. е. интеллектуальной работы, что стимулирует развитие ума.

• Интеллект есть необходимое условие качественной жизни как личности, так и общества.

• Работа с компьютером, особенно обучение компьютера думать, т.е. программирование, служит развитию интеллекта, позволяет активизировать деятельность ученика, его самостоятельность в познании. приближает обучение к реальному познанию, способствует становлению интеллектуально зрелой личности.

Примечания

1. Проект федерального компонента Государственного образовательного стандарта начального общего, основного общего и среднего (полного) образования. Образовательная область «Информатика» // Информатика и образование. 1997. № 1.

2. *Холодная М. А.* Психология интеллекта. Парадоксы исследования. 2-е изд., перераб. и доп. СПб.: Питер, 2002.

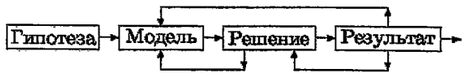
Н. А. Бушмелева

КОГНИТИВНАЯ ФУНКЦИЯ КОМПЬЮТЕРНОЙ ГРАФИКИ

В статье на примере преподавания компьютерной графики показывается, как можно развивать творческое мышление студентов.

Во все времена перед образованием стояла и будет стоять задача формирования культуры мышления будущего специалиста-профессионала, заключающейся в умении творчески, нестандартно мыслить.

Одним из механизмов мышления является образное, геометрическое, интуитивное мышление, обеспечивающее работу с чувственными образами и представлениями об этих образах. Значительное воздействие на него оказало развитие средств компьютерной графики, что привело к появлению нового, пока еще не достаточно очерченного направления, так называемой когнитивной (т. е. способствующей познанию) компьютерной графики. Ее важным моментом является непосредственное, целенаправленное, активирующее воздействие на подсознательные интуитивные механизмы образного мышления человека, что позволяет активизировать свойственную человеку способность мыслить сложными пространственными образами, а также способствует в процессе познания появлению нового знания.



БУШМЕЛЕВА Наталья Александровна – кандидат педагогических наук, доцент кафедры прикладной математики ВятГТУ

Известно, что процесс познания состоит из нескольких, иногда циклично повторяющихся, этапов: гипотеза, модель изучаемого объекта или явления и решение, результатом которого является новое знание (см. рисунок).

Он во многом укладывается в схему «интуиция, абстракция, символическая интерпретация». В этом контексте основная проблема и задача когнитивной компьютерной графики – создание таких моделей представления знаний, в которых можно было бы однообразно представлять как объекты, характерные для логического (символического, алгебраического) мышления, так и объекты, характерные для образного мышления. При этом такое представление должно, во-первых, быть наглядным, «осязаемым», то есть «прояснять» абстрактные построения сложных явлений и процессов (в частности, понятий, гипотез, теорий) средствами графических интерпретаций, во-вторых, позволить увидеть связи и значения, которые до сих пор были скрыты от человека, а в-третьих (как следствие), способствовать выявлению сути абстракции.

Эта проблема (полностью или частично) решается в процессе построения разных изображений, в котором выделяется два основных этапа – моделирование и визуализация. *Моделирование* представляет собой описание объектов различной природы (геометрические объекты на плоскости и в пространстве, реальные природные объекты, например деревья, горы, облака, большие числовые массивы, используемые в экспериментальных исследованиях и др.) с помощью математических методов. На этапе *визуализации* происходит преобразование полученных на предыдущем этапе моделей объектов в одно или несколько статических изображений, результатом чего является построение на плоском экране компьютера реалистических изображений объемного мира.

Для получения натурального изображения объемных объектов на экране компьютера необходимо мастерски владеть методами моделирования геометрических форм и средствами достоверной визуализации. В основе этого лежат фундаментальные дисциплины – аналитическая геометрия, оптика и программирование.

В процессе изучения компьютерной графики реализуются следующие цели: 1) систематизация знаний студентов аналитической геометрии и дополнение их новыми, интересными фактами; 2) закрепление теоретических знаний и выработка навыков их применения в практической деятельности; 3) формирование умения решать нестандартные задачи; 4) развитие навыков самостоятельной работы.

Наилучший способ обучения, дающий сознательные и прочные знания учащимся и обеспечивающий одновременно их умственное развитие, заключается в том, что перед учащимися ставятся одна за другой последовательно связанные посильные теоретические и практические задачи, решения которых дают новые знания. Обучение на немногочисленных, но хорошо подобранных задачах, решаемых учащимися в основном самостоятельно, способствует вовлечению их в творческую исследовательскую работу, последовательно проводя через этапы научного поиска, развивает логическое мышление. Усвоение материала курса через последовательное решение учебных задач происходит в едином процессе приобретения новых знаний и их применения, что способствует развитию познавательной самостоятельности и творческой активности.

С дидактической точки зрения в процессе решения задач компьютерной графики заложены большие (по сравнению с решением алгебраических задач) возможности развития мышления. Решение этих задач существенно и позитивно влияет на развитие логического мышления, пространственного воображения, конструктивно-геометрического мышления, способности к анализу и синтезу пространственных форм и отношений на основе графических моделей пространства, практически реализуемых в виде изображений конкретных пространственных объектов.

В то же время в значительно меньшей степени при решении задач компьютерной графики может быть использован образец решения некоторого класса задач, тем самым осуществляется дестандартизация как самих задач, так и методики их использования в обучении.

Вышесказанное объясняет план занятий: 1) формулировка задачи, постановка проблемной ситуации; 2) анализ имеющихся данных; 3) выбор оптимального из предлагаемых методов решения данной задачи; 4) формулировка теоретического положения, необходимого для решения задачи; 5) применение этого положения при решении задач; 6) реализация решения посредством программы.

По нашему мнению, программирование играет ключевую роль в реализации когнитивной функции компьютерной графики, поскольку оно позволяет получить ответы на многие вопросы типа «почему именно так». Именно детальное изучение алгоритмов и методов компьютерной графики и их реализация способствуют более полному пониманию рассматриваемых объектов и явлений, их связей и закономерностей создания реалистических (как статических, так и динамических) изображений.

Как показал опыт, обучение компьютерной графике через задачи обеспечивает развитие самостоятельности и творческой активности учащихся, способствует приобретению прочных и осознанных знаний, развивает умение сравнивать, обобщать, делать творческие выводы из решенных задач, поддерживает учебно-познавательный интерес учащихся к математике (в частности, к геометрии) и к информатике (в частности, к программированию).

Таким образом, когнитивные функции изучения элементов компьютерной графики состоят:

1) в активизации мыслительной деятельности человека, повышении интереса к изучаемым фундаментальным дисциплинам, в частности к аналитической геометрии, росте уровня остаточных знаний;

2) в развитии пространственного представления и воображения, конструктивно-геометрического мышления, способностей к анализу и синтезу пространственных форм и отношений, изучению способов конструирования различных геометрических пространственных объектов.

Необходимо также заметить, что при изучении компьютерной графики возрастает роль преподавателя. Процесс обучения становится более контролируемым, активно работает прямая и обратная связь «преподаватель – студент». Преподаватель видит «слабые места» студента в том или ином разделе аналитической геометрии, компьютерной графики, обращает на это внимание студента, еще раз разъясняет ошибки.

Развитие когнитивной компьютерной графики открыло для сферы обучения принципиально новые возможности, а именно, увеличение скорости передачи информации учащимся, повышение уровня ее понимания, а также развитие таких важных для специалиста любой отрасли качеств, как интуиция, профессиональное «чутье», образное мышление.

Использование достижений когнитивной компьютерной графики в обучении способствует интеллектуальному процессу получения нового: учащиеся при анализе изображений могут динамически управлять их содержанием, формой, размерами и цветом, добиваясь получения нового, т. е. еще не существующего в их голове, знания (иначе говоря, учащиеся «добывают» знания с помощью исследований на математических моделях изучаемых объектов). Причем графические изображения позволяют каждому учащемуся сформировать свой, сугубо личностный, образ изучаемого объекта или явления во всей его целостности и многообразии связей.

Таким образом, когнитивная компьютерная графика представляет собой эффективный технический инструмент воздействия на интуитивное образное мышление человека.

М. Л. Владимирова

О РАЗВИТИИ КРИТИЧЕСКОГО МЫШЛЕНИЯ В КУРСЕ ПРОГРАММИРОВАНИЯ

В статье раскрывается суть термина «критическое мышление» и на простых примерах из курса программирования показывается, как развивать такое мышление.

ВЛАДИМИРОВА Мария Львовна – ассистент кафедры информатики и МОИ ВятГГУ

Предметом программирования как образовательной дисциплины является программирование как теоретическая и практическая деятельность по обеспечению программного управления обработкой данных, включающая создание программ, а также выбор структуры и кодирования данных [1 : 97]. При этом для реализации такого вида деятельности необходимо уметь:

- определить возможность решения поставленной задачи,
- выделить исходные данные задачи,
- проанализировать возможные методы получения правильного конечного результата и выбрать из них оптимальный метод,
- исходя из выбранного метода решения, продумать структуры данных, оптимальные для решения конкретной задачи, составить алгоритм решения задачи и записать его на языке программирования,
- проверить правильность и оптимальность полученного решения.

К сожалению, практика подготовки студентов на факультете информатики Вятского государственного гуманитарного университета показывает, что не все студенты уже обладают вышеперечисленными умениями и хорошо справляются с решением поставленных перед ними задач. Отсюда, целесообразно скорректировать методику преподавания курса программирования с учетом использования на занятиях приемов, направленных на совершенствование мыслительных процессов студентов.

Для решения этой проблемы обратимся к когнитивной психологии, ведь именно эта наука изучает природу мышления и механизмы его работы. При этом необходимо учесть, что в курсе программирования важно развивать такое мышление, к которому прибегают при решении задач, формулировании выводов и принятии решений. Такой вид мышления в когнитивной психологии определяется как критическое мышление. Вот одно из самых простых определений критического мышления, передающее суть идеи: «Критическое мышление – это использование когнитивных техник или стратегий, которые увеличивают вероятность получения желаемого конечного результата» [2 : 22]. В этом определении важно подчеркнуть направленность критического мышления на получение желаемого результата, другими словами, направленность мышления на решение конкретной когнитивной задачи. Таким образом, когда студент продумывает и реализует процесс решения некоторой задачи, он мыслит критически. Формулу критического мышления можно записать следующим образом:

Установка + Знания + Навыки мышления = Критическое мышление

Действительно, невозможно стать критически мыслящим человеком, не выработав у себя установку на критическое мышление, подразумевающую развитие таких качеств, как умение планировать свои действия, гибкость мышления, настойчивость, готов-

Таблица 1

Качества, свойственные человеку, мыслящему критически	*Этап решения задачи в курсе программирования, требующий развития соответствующих качеств*	*Деятельность студента на соответствующем этапе решения задачи*
Готовность к планированию	Построение алгоритма решения задачи	Необходимо определить точный порядок действий для достижения результата: вначале составляется последовательность из небольшого числа достаточно крупных шагов, затем выполняется более подробное описание каждого шага – детализация алгоритма
Гибкость (готовность рассматривать новые варианты), поиск компромиссных решений	Постановка задачи, построение информационной модели. Формализация задачи. Получение и анализ результатов	Необходимо определить, что известно и что является результатом решения, а также как связаны исходные данные и результаты. При этом важно уметь выбрать форму представления данных, оптимальную для компьютерной обработки. Необходимо уметь оценивать правильность и рациональность полученного решения и при необходимости изменить саму информационную модель либо алгоритм с целью оптимизации решения
Настойчивость, готовность исправлять свои ошибки	Отладка и тестирование программы	Необходимо устранить ошибки программирования. При этом важно уметь разработать систему тестов, предусмотрев разнообразные варианты хода вычислительного процесса, а также действия пользователя, и, таким образом, защитить работу программы от ввода некорректных данных, неверных значений и т.д.
Осознание	Все этапы решения задачи	Важно оценивать, как протекают мыслительные операции и их конечный результат – полученное решение. Необходимо уметь оценить время и усилия, требуемые для решения задачи и подзадач, а также осуществлять постоянный контроль за процессом продвижения к поставленной цели

ность исправлять свои ошибки, осознание, наблюдение за мыслительным процессом и поиск компромиссных решений. Если провести параллель с программированием, то несложно заметить, что большинство из перечисленных выше качеств будут планомерно развиваться в процессе решения задач (табл. 1).

Еще одной составляющей критического мышления являются навыки мышления, т.е. набор приемов или операций, позволяющих найти путь к поставленной цели. Так, Д. Халперн, опираясь на богатейший теоретический и фактический материал, накопленный когнитивной психологией при изучении познавательных процессов, выделила ряд определенных навыков и приемов мышления, овладев которыми студенты начинают мыслить критически [2]. Отсюда, целесообразно предположить, что изучение опыта когнитивных психологов по развитию критического мышления и осуществление переноса соответствующих когнитивных стратегий по совершенствованию мыслительных операций на процесс обучения программированию позволит добиться повышения успеваемости студентов в ходе решения задач.

Ниже приводится примерное занятие по теме «конструкция ветвления» в рамках курса программирования, учитывающего методы формирования критического мышления, предложенные когнитивными психологами.

*1-й этап – **введение новой алгоритмической конструкции***

На этом этапе занятия, при объяснении нового материала, целесообразно использовать средства повышения степени понимания материала студентами. В этом случае когнитивная психология предлагает целый ряд различных стратегий понимания, нацеленных

на то, чтобы помочь сделать более понятной информацию, переданную вербально (обычным языком).

Прежде всего, в качестве огромного подспорья вербальному изложению информации выступают графические систематизаторы. С их помощью можно изобразить структуру знаний изучающего и показать, каким образом новая информация встраивается в то, что уже известно. В информатике таким графическим систематизатором может служить блок-схема. При введении новой алгоритмической конструкции целесообразно ввести запись конструкции ветвления не только на языке программирования, но и в виде блок-схемы (рис. 1).

Следующий когнитивный метод, облегчающий понимание материала, – это *метод взаимных вопросов и ответов.* В связи с тем, что изучение нового материала проходит наиболее эффективно тогда, когда по нему задают правильно поставленные вопросы, а затем при ответах выявляется степень понимания материала, после введения конструкции ветвления и объяснения принципа ее использования рекомендуется обсудить новую конструкцию со студентами в форме беседы. При этом следует учитывать, что умение задавать вопросы – это тот навык, которому следует учить, поскольку большинство людей привыкло задавать примитивные вопросы, требующие при ответе на них лишь небольшого напряжения памяти. Психолог А. Кинг разработала серию вопросов, при ответе на которые студенты из экспериментальной группы запоминали и понимали материал гораздо лучше, чем студенты из контрольной группы, которых приучали либо знакомиться с материалом самостоятельно, либо задавать вопросы, но не показывали им при этом, как следует правильно задавать вопросы (табл. 2) [2 : 140].

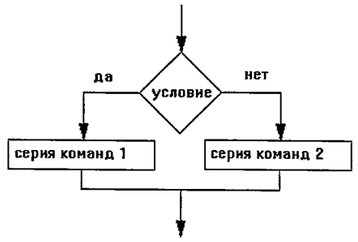


Рис. 1. Блок-схема полной формы конструкции ветвления

В рамках темы конструкция ветвления в ходе беседы рекомендуется предложить студентам следующие вопросы:

- В чем разница между полной и неполной формами записи конструкции ветвления?
- Приведите примеры использования полной формы конструкции ветвления в повседневной жизни?
- Приведите примеры использования неполной формы конструкции ветвления в повседневной жизни?
- Можно ли при решении задач использовать только неполную форму конструкции ветвления?

Таблица 2

Вопросы, направляющие ход мышления

Вопросы	*Мыслительные операции*
Приведите пример …	Приложение
Каким образом можно … использовать для …?	Приложение
Что случится, если …?	Предположение/Выдвижение гипотезы
Что подразумевается под …?	Анализ/Заключение
В чем сильные и слабые стороны …?	Анализ/Заключение
Что мы уже знаем о …?	Активизация ранее приобретенных знаний
Каким образом … влияет на …?	Активизация причинно-следственных отношений
Каким образом … связано с тем, что мы изучили ранее?	Активизация ранее приобретенных знаний
Объясните, почему …?	Анализ
Объясните, как …?	Анализ
Почему важно …?	Анализ значимости
В чем разница между … и …?	Сравнение–противопоставление
Как можно применить … в повседневной жизни?	Применение в реальном мире
Какой аргумент можно привести против …?	Контраргументация
Какими могут быть возможные решения задачи?	Синтез идей
Сравните … и … на основании …	Сравнение–противопоставление
Что, на ваш взгляд, является причиной … и почему?	Анализ причинно-следственных связей
Согласны ли вы с утверждением, что … ?	Оценка и ее обоснование
Чем вы можете аргументировать свой ответ?	Оценка и ее обоснование

• В чем преимущество использования полной формы конструкции ветвления?

• В чем отличие конструкции ветвления от конструкции следования?

• Что вы уже знаете об условиях, что может выступать в качестве условия?

• Каким образом можно использовать конструкцию ветвления для определения возрастной группы человека?

• Какие операторы можно использовать в конструкции ветвления?

• Что может привести к возникновению неоднозначности в способе записи конструкции ветвления?

2-й этап – ***отработка использования новой алгоритмической конструкции на примере решения задач.***

Когнитивные психологи разработали универсальный алгоритм, или руководство, помогающее направить процесс мышления на определенную задачу. Следующие вопросы служат для упорядочения процесса мышления.

1. Какова цель?
2. Что известно?
3. Какие навыки мышления позволяют вам достичь поставленной цели?
4. Достигнута ли поставленная цель?

При разборе решения задач важно вместе со студентами продумать ответы на поставленные вопросы. Так, разбор конструкции ветвления можно выполнить на следующем простом примере.

Даны два конверта прямоугольной формы с длинами сторон (a,b) и (c,d). Определить, можно ли первый конверт вложить во второй?

После определения цели задачи и исходных данных целесообразно использовать различного рода графические изображения, являющиеся отличной когнитивной стратегией решения задач. Для нашего примера удобно использовать блок-схему или древовидную диаграмму. Первоначальный алгоритм решения задачи, предлагаемый студентами, отображен на блок-схеме (рис. 2). Ниже приведено решение, записанное на языке Паскаль.

```
Program Task1;
Var a, b, c, d: Integer;
Begin
 WriteLn('Введите размеры первого
                                конверта');
 ReadLn(a, b);
 WriteLn('Введите размеры второго
                                конверта');
 ReadLn(c, d);
 If (a<c)AND(b<d) Then
     WriteLn('Первый конверт входит
                                во второй')
 Else WriteLn('Первый конверт не входит
                                во второй');
 ReadLn;
End.
```

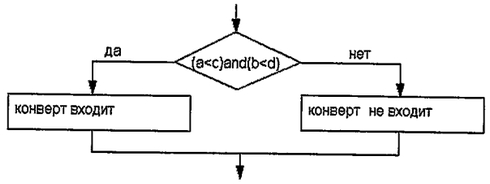


Рис. 2

Заключительный, но не менее важный *этап* в решении любой задачи заключается в ***оценке качества решения.*** Когнитивные психологи утверждают, что на стадии проверки правильности решения задачи большинство из нас выносят ошибочные суждения в силу так называемой тенденции к подтверждению. Оказывается, при принятии решений людям свойственно подбирать ту информацию, которая соответствует их представлениям. Так, при проверке правильности решения задачи большинство студентов демонстрирует неумение тестировать написанные программы, ограничиваясь одним-двумя вариантами входных данных, подтверждающих корректность работы программы. Поэтому важно учить студентов обнаруживать такие варианты входных данных, для которых написанная программа будет работать неверно. Так, для рассмотренной задачи примером входных данных, приводящих к некорректной работе программы, могут служить следующие значения: a=15, b=3, c=4, d=20. В этом случае, в соответствии с приведенным выше решением, на экран будет выведено сообщение о том, что первый конверт во второй не входит, хотя в действительности это не так. Следовательно, полученное решение не является правильным и требует доработки. Блок-схема модифицированного решения приведена на рис. 3.

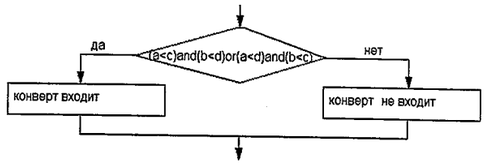


Рис. 3

Ниже приведено решение задачи на языке Паскаль.

```
Program Task1_2;
Var a, b, c, d: Integer;
Begin
 WriteLn('Введите размеры первого
                                конверта');
 ReadLn(a, b);
 WriteLn('Введите размеры второго
                                конверта');
 ReadLn(c, d);
```



```
 If  (a<c)AND(b<d)OR(a<d)AND(b<c)  Then
WriteLn('Конверт входит')
 Else WriteLn('Первый конверт не входит во
второй');
 ReadLn;
End.
```

При дальнейшей отладке программы определяется еще один вариант входных данных, приводящих к некорректной работе программы – ввод отрицательных чисел. По условию задачи входные данные являются длинами сторон конвертов, а так как длина не может быть отрицательной величиной, то при вводе таких значений необходимо вывести сообщение о некорректности ввода и завершить работу программы. Тогда решение задачи на языке Паскаль будет иметь вид:

```
Program Task1_3;
Var a, b, c, d: Word;
Begin
 WriteLn('Введите размеры первого
                                конверта');
 ReadLn(a, b);
 WriteLn('Введите размеры второго
                                конверта');
 ReadLn(c, d);
 If (a<=0)OR(b<=0)OR(c<=0)OR(d<=0) Then
     WriteLn('Некорректный ввод данных')
 Else
  If  (a<c)AND(b<d)OR(a<d)AND(b<c)  Then
WriteLn('Конверт входит')
 Else WriteLn('Первый конверт не входит
                                во второй');
 ReadLn;
End.
```

Далее целесообразно модифицировать условие задачи следующим образом.

Даны два конверта прямоугольной формы с длинами сторон (a,b) и (c,d). Определить, можно ли один конверт вложить в другой?

Чаще всего в процессе решения задачи продвижение к цели не происходит по прямой «вымощенной дороге». Если цель не может быть достигнута сразу, нередко приходится идти обходными путями или разбивать задачу на более мелкие части – так называемые подзадачи, каждая из которых имеет свою цель

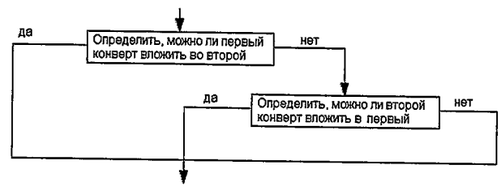


Рис. 4

или подцель. Процедура, согласно которой люди определяют подцели и используют их для продвижения к основной цели, в когнитивной психологии называется анализом целей и средств. Эта когнитивная стратегия является одним из основных, очень мощных средств решения задач.

Таким образом, после определения цели задачи необходимо разбить задачу на подзадачи. При этом для облегчения процесса разбиения задачи на подзадачи и поиска решения опять же удобно использовать графические изображения, например иерархическое дерево (рис. 4).

В результате задача разбивается на две подзадачи, решение которых уже найдено. Ниже приведено решение задачи.

```
Program Task2;
Var a, b, c, d: Word;
Begin
 WriteLn('Введите размеры первого
                                конверта');
 ReadLn(a, b);
 WriteLn('Введите размеры второго
                                конверта');
 ReadLn(c, d);
 If (a<=0)OR(b<=0)OR(c<=0)OR(d<=0) Then
     WriteLn('Некорректный ввод данных')
 Else
 If (a<c)AND(b<d)OR(a<d)AND(b<c) Then
WriteLn('Первый конверт входит во второй')
 Else
 If (a>c)AND(b>d)OR(a>d)AND(b>c) Then
WriteLn('Второй конверт входит в первый')
 Else WriteLn('Один конверт вложить
                                в другой нельзя');
 ReadLn;
End.
```

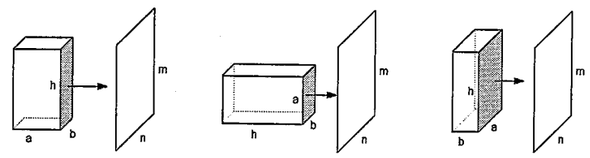


Рис. 5

Еще одной модификацией данной задачи может служить следующий пример.

*Определить, войдет ли прямоугольный шкаф размером a*b*h в дверной проем размером n*m.*

Для решения этой задачи опять же полезно применить метод анализа целей и средств и, в зависимости от способа занесения шкафа в дверной проем, разбить задачу на три подзадачи (рис. 5).

Кроме того, с целью приведения каждой из подзадач к уже решенной задаче, при построении иерархического дерева уместно прибегнуть к такой когнитивной стратегии решения задач, как переформулировка (рис. 6).

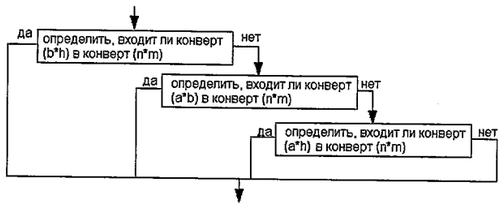


Рис. 6

Ниже приведено решение задачи.

```
Program Task3;
Var a, b, h, n, m: Word;
Begin
 WriteLn('Введите размеры первого конверта');
 ReadLn(a, b);
 WriteLn('Введите размеры второго конверта');
 ReadLn(c, d);
 If (a<=0)OR(b<=0)OR(h<=0)OR(n<=0)OR(m<=0)
Then
     WriteLn('Некорректный ввод данных')
 Else
  if (b<n)AND(h<m)OR(b<m)AND(h<n) Then
      WriteLn('входит')
  Else
  If (a>n)AND(b>m)OR(a>m)AND(b>n) Then
      WriteLn('входит')
  Else
  If (a>n)AND(h>m)OR(a>m)AND(h>n) Then
      WriteLn('входит')
  Else WriteLn('не входит');
 ReadLn;
End.
```

Итак, в приведенной выше схеме занятия по теме «Конструкция ветвления» использованы метод графических систематизаторов и метод взаимных вопросов и ответов для облегчения усвоения студентами нового материала. Кроме того, разобраны три задачи, на примере которых студенты учатся, во-первых, использовать графические изображения (блок-схема и иерархическое дерево) для облегчения поиска решения, и, во-вторых, знакомятся с когнитивными стратегиями решения задач – анализом целей и средств и переформулировкой задачи. Такая методика преподавания курса программирования, ориентированного на использование когнитивных методик и стратегий развития критического мышления на каждом занятии, позволит повысить способность студентов решать возникающие задачи.

Примечания

1. *Окулов С. М.* Когнитивная информатика. Киров: ВятГГУ, 2003.

2. *Халперн Д.* Психология критического мышления / Пер. с англ. СПб.: Питер, 2000.

Н. И. Исупова

УЧЕТ ПСИХОЛОГИЧЕСКИХ ОСОБЕННОСТЕЙ ПРОЦЕССА ОБРАЗОВАНИЯ ПОНЯТИЙ ПРИ ОРГАНИЗАЦИИ ОБОГАЩАЮЩЕГО ОБУЧЕНИЯ

Основное содержание статьи составляет описание особенностей процесса образования понятий при обучении информатике. В работе обосновывается ведущая роль понятий во всей интеллектуальной деятельности человека. При этом акцент ставится на интегративный характер понятийных структур. Рассматриваются основные компоненты понятийного мышления в рамках онтологического подхода.

...Образование понятий подростка никогда не идет тем логическим путем, как его рисует традиционная схема.

Л. С. Выготский

Основная цель, которую ставит перед собой обогащающее обучение – это развитие интеллекта учащегося. В качестве психологической основы такого интеллектуального воспитания выступает обогащение ментального (умственного) опыта каждого ребенка. Поэтому процесс обучения должен содействовать учету и формированию компонентов ментального опыта (на уровне его когнитивного, метакогнитивного и интенционального опыта).

Когнитивный опыт – это ментальные структуры, которые обеспечивают хранение, упорядочение и преобразование наличной и поступающей информации, способствуя тем самым воспроизведению в психике познающего субъекта устойчивых, закономерных аспектов его окружения [1]. Их основное назначение –

ИСУПОВА Наталья Ивановна – аспирант кафедры информатики и МОИ ВятГГУ

оперативная переработка текущей информации об актуальном воздействии на разных уровнях познавательного отражения. К ментальным структурам, образующим состав когнитивного опыта, можно отнести архетипические структуры, способы кодирования информации, когнитивные схемы, семантические структуры, и, наконец, особое место здесь занимают понятийные структуры, которые можно рассматривать как результат интеграции вышеуказанных базовых механизмов переработки информации.

Не секрет, что формирование понятий всегда рассматривалось как одна из наиболее важных целей и особенностей обучения в любой модели, любой технологии. Многие ученые и исследователи признавали особую роль понятийного мышления в структуре интеллекта, рассматривая способность к понятийному отражению как высшую стадию интеллектуального развития, а понятийную мысль - как один из наиболее эффективных познавательных элементов. Так, Л. С. Выготский, изучая закономерности умственного развития ребенка, пришел к заключению, что именно образование понятий является ключом к пониманию процессов психологического (в том числе интеллектуального) развития подростка [2]. В работах О. Харви, Д. Ханта и Х. Шродера основное внимание также уделялось структурным аспектам организации индивидуальных понятийных схем, а основной единицей анализа выступало «понятие» (или «концептуальная схема») как «устройство, фильтрующее опыт», посредством которого субъект воспринимает, преобразует и оценивает то или иное воздействие. Авторы отмечают, что для нормального развития личности роль понятийной системы исключительно велика, ибо «разрыв концептуальных связей между субъектом и объектами, с которыми он взаимодействует, будет способствовать деструкции Я, уничтожению той пространственной и временной опоры, от которой зависят все определения его существования» [3].

Много позже Л. М. Веккер при разработке теории психических процессов снова возвращается к ведущей роли понятийных структур в системе интеллекта. Он указывает, что понятийные образования (концепты) включают в себя все нижележащие уровни когнитивных структур, выступают в качестве «формы интегральной работы интеллекта», а сам концепт выступает как «интеллектообразующая интегративная единица» [4].

Все эти и другие работы не оставляют сомнений, что в процессе обучения особое внимание нужно обращать на учет процесса формирования понятий. Образование понятий – это длительный процесс. И хотя отдельные элементы этого процесса можно зафиксировать на самых ранних стадиях онтогенеза (например, уже сам факт, что ребенок овладел словом, свидетельствует о зарождении способности к обобщению), тем не менее собственно понятия появляются только в переходном (подростковом) возрасте примерно с 11-12 лет. Об этом свидетельствуют труды некоторых психологов. Например, Ф. Римат посвятил специальное, очень обстоятельное исследование процессу образования понятий у подростков и пришел к следующему выводу: «Мы можем твердо установить, что лишь по окончании 12-го года жизни обнаруживается резкое повышение способности самостоятельного образования общих объективных представлений... Мышление в понятиях, отрешенное от наглядных моментов, предъявляет к ребенку требования, которые превосходят его психические возможности до 12-го года жизни» [5].

Ж. Пиаже также давал детям 10-12 лет задачи на совмещение двух признаков одновременно – животное имеет длинные уши и короткий хвост или короткие уши и короткий хвост. В результате таких экспериментов он сделал вывод, что ребенок решает задачи, имея в поле внимания только один признак, оперировать понятием как системой он не может; он овладевает всеми признаками, входящими в понятие, но всеми – порознь, он не владеет тем синтезом, при котором понятие действует как единая система [6].

Аналогично, в ходе проведения ряда экспериментов на основе функциональной методики двойной стимуляции Л.С. Выготский заключает, что «развитие процессов, приводящих впоследствии к образованию понятий, уходит своими корнями глубоко в детство, но только в переходном возрасте вызревают, складываются и развиваются те интеллектуальные функции, которые в своеобразном сочетании образуют психическую основу процесса образования понятий» [7].

Таким образом, именно переходный (подростковый) возраст должен быть объектом пристального внимания педагогов в процессе обучения. Для современной практики обучения особый интерес представляет поиск ответа на вопрос о том, почему именно с образованием понятий Л. С. Выготский связывал коренную перестройку всей интеллектуальной деятельности подростка, а также существенные изменения содержания его сознания в целом.

Во-первых, благодаря понятиям подросток начинает понимать связи, отношения, взаимозависимости, скрытые за поверхностью видимых явлений, и, следовательно, постигать закономерности, управляющие действительностью. Эту мысль подтверждают слова Л. С. Выготского: «...когда испытуемый решает задачу на образование новых понятий, то сущность процесса, происходящего при этом, заключается в установлении связей; когда испытуемый подыскивает к предмету ряд других предметов, он ищет связи между этим предметом и другими. ...Понятие заключается не в коллективной фотографии, не в том, что стираются индивидуальные черты предмета, а в том, что предмет познается в его отношениях и связях» [8]. Кроме того, понятия – это средство упорядочения воспринимаемого мира с помощью «сетки» категориальных и логических отношений, то есть это тот

интеллектуальный инструмент, который помогает справиться с хаосом эмпирических впечатлений и организовать их на уровне разумной картины мира.

Во-вторых, с помощью понятий происходит расширение среды сознания подростка. Иными словами, средой для мышления подростка становится весь мир в его многообразии и целостности.

В-третьих, происходит перестройка («интеллектуализация») элементарных познавательных функций на основе их синтеза с функцией образования понятий: восприятие фактически превращается в наглядное мышление, запоминание начинает опираться на смысловые связи, внимание приобретает произвольный характер и т. д. Недаром Л. С. Выготский отмечает: «Образование понятия или приобретение словом значения является результатом сложной активной деятельности (оперирование словом или знаком), в которой участвуют все основные интеллектуальные функции в своеобразном сочетании» [9].

В-четвертых, понятия выступают в качестве средства адекватного и полного усвоения исторически сложившегося опыта человечества. Фактически только через понятия индивидуум открыт культуре, и, таким образом, только через понятия осуществляется наиболее эффективная социализация (очеловечивание) индивидуального интеллекта, что создает предпосылки для понимания других людей (и других вариантов культуры).

В-пятых, благодаря формированию понятийного мышления (владению понятиями) содержание мышления становится внутренним убеждением подростка, его интересом, желанием и намерением. Переплетаясь со сложными внутренними моментами личности, содержание мышления становится «достоянием личности, начинает участвовать в общей системе движения этой личности» [10].

В-шестых, понятийный опыт — это основа самопознания, ибо, по словам Л. С. Выготского, «только с образованием понятий наступает интенсивное развитие самовосприятия, самонаблюдения, интенсивное познание внутренней действительности, мира собственных переживаний» [11].

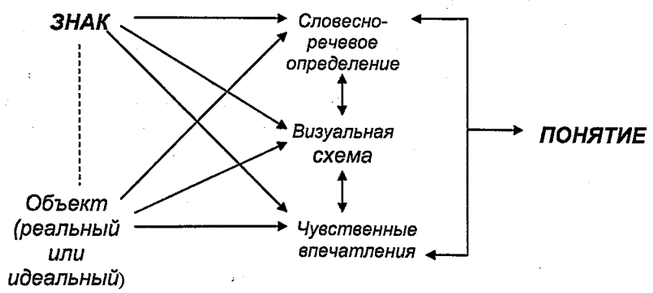
Таким образом, мышление в понятиях обеспечивает возможность нового типа понимания объективного мира, возможность понимания других людей и, наконец, возможность понимания самого себя.

Не удивительно, что задача формирования понятийного мышления — это одновременно и задача развития личности и ее отношений с окружающим миром. Именно поэтому в центре обогащающей технологии преподавания оказывается требование формирования понятийного мышления учащихся как психологической основы компетентности и важнейшего условия их интеллектуального роста.

В данном случае важно подчеркнуть следующий момент: усвоение понятий (как внешних ребенку единиц научного знания) и образование понятий (как когнитивных структур) — это не тождественные явления. С психологической точки зрения образование понятий — это процесс превращения определенных единиц объективно существующего знания в субъективные ментальные структуры, существующие уже «внутри» опыта человека в качестве психических новообразований [12]. Беспокоиться, следовательно, нужно не просто об усвоении понятий, а о выстраивании в ментальном опыте ребенка понятийных психических структур как психологических носителей понятийного знания.

Понятийные психические структуры – это интегральные когнитивные структуры, особенности устройства которых характеризуются включенностью разных способов кодирования информации, представленностью визуальных схем разной степени обобщенности и иерархическим характером организации семантических признаков [13]. Интегральный характер понятийных структур заключается в том, что большинство авторов выделяют в их составе несколько когнитивных компонентов. Чаще всего в качестве последних выступают три модальности опыта — словесно-речевая, визуальная и чувственно-сенсорная. Схематически образование понятий можно представить следующим образом (см. рисунок).

Однако некоторые психологи считают такую схему неполной, например М. А. Холодная [14] к перечисленным компонентам добавляет еще три: операционально-логический, мнемический и аттенционный, кроме того, все шесть элементов достаточно тесно и в то же время избирательно взаимосвязаны. Дж. Брунер отмечает, что развитие интеллекта (в том числе и процесс образования понятий) идет «под влиянием тех способов, с помощью которых люди постепенно научаются представлять себе мир, в котором оперируют, а именно через действия, образы и символы» [15].



Л. С. Выготский, рассуждая об образовании понятийных психических структур, в качестве основных выделяет словесно-речевой и чувственно-сенсорный компонент: «...чувственный материал и слово – необходимые элементы процесса образования понятий, и слово, оторванное от этого материала, переводит весь процесс определения понятия в чисто вербальный план, не свойственный ребенку» [16]. В то же время он подтверждает мысль Дж. Брунера о роли действия при формировании понятий, в частности Л. С. Выготский пишет: «...только при возникновении известной потребности, надобности в понятии, только в процессе какой-то осмысленной целесообразной деятельности, направленной на достижение известной цели или решение определенной задачи, может возникнуть и оформиться понятие» [17].

Таким образом, обогащающая модель обучения строится на том положении, что процесс образования понятий должен предполагать специально разработанную систему заданий, ориентированных на разные составляющие понятийных структур. Выполнение таких заданий в рамках усвоения той или иной темы должно обеспечивать подключение чувственно-сенсорных впечатлений учащихся, обратимые переводы информации с языка знаков и символов на язык образов (визуальных схем разной степени обобщенности), работу с определениями понятий и их признаками, уяснение связей с другими понятиями, а также формирование базовых мыслительных операций.

Далее, содержательное, осмысленное усвоение понятий – это развернутый во времени процесс, в котором могут быть выделены определенные фазы движения мысли, в том числе: *мотивировка, категоризация, обогащение, перенос, свертывание.* Соответственно последовательность изложения учебного материала должна строиться таким образом, чтобы при этом учитывалась внутренняя динамика мысли ребенка при его постепенном переходе от знания значения нового знака (формулы, символического обозначения, словесного определения) к собственно понятийному обобщению этого нового знания.

Наконец, необходимо иметь в виду, что образование понятий осуществляется только за счет интериоризации готовых сведений об окружающей действительности, но и на основе интеллектуальной самодеятельности ребенка. Учебный текст должен «отпускать» ученика вперед, давать ему возможность самому формулировать определения, вводить и обосновывать признаки понятий и т. п. И тогда выясняется, что дети гораздо умнее, чем нам кажется.

Таким образом, при конструировании учебных текстов нужно учитывать три аспекта процесса образования понятий:

- *во-первых,* в виде подбора и разработки обучающих заданий, каждое из которых будет иметь своим психологическим адресатом важнейшие характеристики понятийных психических структур;
- *во-вторых,* в виде выстраивания последовательности учебного материала, отвечающей требованию пофазового формирования субъективного образа содержания понятия;
- *в-третьих,* в виде предоставления ученику возможности самостоятельно участвовать в процессе усвоения понятий и наполнения их соответствующим содержанием.

В связи с этим выделяются учебные задания следующих типов.

1. Задания на подключение предметного (житейского) опыта детей. По мнению Дж. Брунера, образование понятий уходит своими корнями в глубинные структуры индивидуально-ментального опыта. Поэтому, добиваясь взаимодействия житейского опыта ребенка и тех научных знаний, которые предлагаются ему в учебном процессе, мы одновременно решаем две задачи: с одной стороны, под влиянием научного знания происходит актуализация и обогащение чувственно-сенсорных впечатлений ребенка и, с другой стороны, сами чувственно-сенсорные впечатления начинают оказывать активное влияние на процесс образования научных понятий, что в целом обусловливает возможность появления «личностного знания» [18].

Например, на уроках информатики при введении понятия алгоритма можно предложить учащимся такую ситуацию: «Вы купили книжный шкаф, который вам доставили в разобранном виде. Необходимо собрать шкаф. Ваши действия?». Вероятнее всего, дети ответят, что воспользовались бы прилагающейся инструкцией. А что представляет собой инструкция? – Это описание некоторой последовательности действий, правильное выполнение которой должно привести к нужному результату. Но это и есть алгоритм (нестрого говоря).

2. Задания на формирование способности к словесно-образному переводу, то есть переводу научной информации со знаково-символического «языка» на «язык» рисунков-образов в виде схем, графиков, моделей, предметно-индивидуальных образов. В данном случае речь идет не только о развитии способности к визуализации знания (в частности, за счет использования различных визуальных схем), но и о возможности одновременной, взаимообусловленной работы двух основных субъективных способов кодирования информации – словесного и образного – как базового механизма мышления [19].

При изучении информатики такую способность развивают, например, задания на оформление алгоритма в виде блок-схемы. Кроме того, богатый материал для развития мышления в данном аспекте могут дать задачи на перебор, в которых часто используется графическое изображение переборной схемы, а также алгоритмы на графах, предполагающие перевод словесно-символьной информации в визуально-образную.

3. Задания на выделение признаков усваиваемого понятия, ориентирующие ребенка на выявление множества возможных признаков, их дифференциацию, соотнесение различных признаков по степени их значимости и степени обобщенности, систематизацию наиболее существенных признаков и понимание того, что мера существенности или несущественности признака может меняться в зависимости от цели деятельности.

Такие задания можно формулировать прямо по ходу усвоения понятия в форме небольших устных вопросов. Так, после определения алгоритма выводятся его свойства. Лучше всего это можно сделать на конкретном примере алгоритма: изменяя алгоритм (так, чтобы построить контрпримеры на каждое свойство), можно легко показать его несостоятельность и тем самым вывести требования, необходимые для правильной работы алгоритма. Кроме того, при решении более или менее сложной задачи ученик, анализируя ее условия, так или иначе сталкивается с проблемой определения существенных (которые необходимо учесть при составлении алгоритма) и несущественных условий.

4. Задания на включение исходного понятия в систему связей с другими понятиями. Принимаются во внимание связи научных понятий с понятиями из других областей знания (математики, физики, географии, биологии, экономики). Кроме того, межпонятийные связи прослеживаются за счет анализа развития того или иного понятия в истории информатики и т. д.

Очень важно при введении понятия опираться на имеющиеся у учащихся знания об этом понятии из других предметных областей. В частности, когда изучается понятие величины, можно провести аналогию с известными детям величинами в алгебре, физике, например вспомнить, что они уже встречали величины как постоянные (U = 220В; p =3,14 и т. д.), так и переменные (x = -5; a = 0,3 и т. д.). Отсюда можно перейти к вопросу об основных характеристиках величины (имя и значение).

5. Задания на развитие мыслительных операций, лежащих в основе образования понятий (таких, как анализ, синтез, обобщение, сравнение, конкретизация, абстрагирование). Учитывается, в соответствии с позицией Ж. Пиаже [20], что субъективной мерой овладения мыслительными операциями является их системность и обратимость.

На самом деле, предмет информатики построен таким образом, что само решение задачи, составление алгоритма уже предполагает развитие мыслительных операций. Сколько раз ученик должен проанализировать условия задачи, синтезировать имеющиеся знания, обобщить, сравнить, проверить результаты, наконец, абстрагироваться от несущественных условий и ситуаций, чтобы прийти к лаконичному, универсальному, правильному решению и «дойти до цельного восприятия и понимания программы как знака (динамической модели) решаемой проблемы» [21]! Решая задачи, совершенствуя программы, ребенок совершенствует свой интеллект.

Актуализируя и развивая те компоненты ментального опыта ребенка, которые выступают в качестве основы процесса образования понятий, мы, кроме того, должны «собрать» их воедино с тем, чтобы можно было говорить о действительной сформированности понятийных структур «внутри» опыта ученика. Этому способствует, по мнению М. А. Холодной [22], такая форма организации текста, которая позволяет последовательно выстраивать субъективный образ содержания соответствующего понятия. В учебных текстах должны учитываться следующие основные фазы образования понятия:

1) *мотивировка* — создание условий для осознания учащимися необходимости нового способа описания своего предыдущего опыта (житейского, физического, арифметического), например, за счет создания эффекта «невозможности» разрешения ситуации в силу отсутствия на данный момент адекватных понятийных средств ее анализа;

2) *категоризация* — введение знаково-символического и визуального обозначения понятия с последующим постепенным увеличением степени обобщенности знаково-символического и визуального «языков» представления его содержания, а также ориентация ребенка на выделение отличительных частных и общих (несущественных и существенных) признаков соответствующего понятия;

3) *обогащение* — накопление и дифференциация опыта оперирования вводимым понятием, расширение возможных ракурсов осмысления его содержания (за счет включения разных вариантов его интерпретации, увеличения числа межпонятийных связей, использования альтернативных контекстов его анализа и т. д.);

4) *перенос* — применение усваиваемого понятия в разных ситуациях, в том числе в условиях самостоятельного выстраивания отдельных аспектов его содержания;

6) *свертывание* — экстренная реорганизация всего множества имеющихся у ученика сведений относительно данного понятия и превращение их в обобщенную единицу знания. Иными словами, развернутый на предыдущих фазах субъективный образ понятия на этой фазе должен быть представленным в сжатой концентрированной форме, что на уровне учебного текста может обеспечиваться такими приемами, как создание «бессмысленных» ситуаций (например,

в условиях вынужденного выполнения долгих, громоздких вычислений), работа с предельно «открытыми» заданиями типа: «Составь рекламу для изучения фильтрации данных в электронных таблицах», составление конспектов, введение жесткого ограничения времени на выполнение определенных заданий и т. д.

В качестве примера рассмотрим содержание основных фаз формирования понятия «файла».

Мотивировка. Компьютер – устройство для работы с информацией. В реальной жизни носителями информации являются вещественные объекты волны и поля, состояние вещества и т. д. А каким образом организовать хранение информации в компьютере, чтобы можно было без труда найти интересующие нас сведения?

Категоризация. Вводится новые термины – файл, имя, тип (расширение), значок файла, свойства файла, правила формирования имени файла, первое упоминание о программах, в которых создаются файлы различных типов и т. д.

Обогащение. После знакомства с файлом логично рассмотреть понятие папки как группы файлов. Затем формируется представление об иерархической (древовидной) структуре организации папок и файлов. Вводится понятие полного имени файла (пути к файлу). Чтобы уверенно ориентироваться в мире папок и файлов, ученику приходится освоить целый ряд операций над файлами: создание, перемещение, копирование, удаление, переименование и т. д., что потребует изучения определенных программ. Далее, обогащению содержания понятия файла способствует знакомство ребенка с ситуацией, в которой он должен найти файлы по определенным критериям, а также организовать быстрый доступ к какому-либо объекту (в виде создания ярлыков).

Перенос. В процессе обучения должны создаваться условия для самостоятельного переноса усвоенного понятия в других ситуациях, например, при изучении новых программ особого внимания по работе с файлами уже не уделяется, а изучаются только собственно особенности этой программы. То есть понятие файла изучается только один раз, а вся дальнейшая работа с файлами в других программах должна носить форму переноса ранее приобретенных знаний и навыков.

Свертывание. Накопленные сведения о файлах должны быть свернуты на уровне единого – целостного, обобщенного и динамичного – представления о сути соответствующего объекта. Решению этой задачи может способствовать специальный вопросник, предлагаемый детям в конце данного учебного раздела. Каждый вопрос, сформулированный в заведомо парадоксальной форме, выступает в качестве катализатора процесса свертывания знаний ребенка. Например, вопросы типа: «Файл TEXT.TXT находится на диске А. Путь к этому файлу: РОТ\8О8\. Каково полное имя файла?», «Какого типа файл с именем Работа. jpg?», «Как узнать размер файла?», «Назовите общее и различия папки и файла», «На Рабочем столе находится папка «1» и файл «Работа». Пользователь прижал клавишу CTRL, выделил мышью файл «Работа» и переместил его на значок папки «1». Что при этом произошло?», «Какое из предложенных ниже полных имен файла записано правильно?» и т.д.

Фаза «свертывания», по сути, завершает процесс «кристаллизации» опыта относительно определенной сферы научного знания. Этот уровень организации понятийного опыта можно рассматривать основой компетентности как одного из показателей уровня интеллектуального развития личности. Отметим, что аналогичной точки зрения придерживаются В. А. Крутецкий, рассматривающий эффект «свертывания» как ключевой признак математических способностей [23], а также Дж. Уолтерс и Х. Гарднер, объясняющие экстраординарные интеллектуальные достижения эффектом «кристаллизации» индивидуального опыта [24].

Что касается **интеллектуальной самодеятельности** учащихся в процессе усвоения новых понятий, то в самом учебном тексте должны быть предусмотрены такие формы организации учебной информации, которые позволяют ученику мысленно участвовать в процессе рождения нового понятия, пересматривать его содержание по мере углубления представлений о соответствующих объектах вплоть до самостоятельного выстраивания нового понятия на базе некоторых исходных понятийных знаний.

Например, после изучения темы форматирования символов можно предложить учащимся самостоятельно изучить, а затем самим написать параграф по теме «Форматирование абзацев документа». Каждый ученик на основе предложенных рекомендаций и советов должен самостоятельно подготовить текст с использованием нужного, с его точки зрения, материала. Дети придумывают задания, для которых могут понадобиться новые знания, составляют тренировочные упражнения с их обоснованием, контрольные работы, вопросы для самопроверки и т. д. Каждый ученик для описания нового объекта сам выбирает жанр, который ему больше нравится (в виде истории в картинках, научного отчета, раздела учебника, пьесы и т. д.). В ходе такой работы ученик должен не только сам разобраться в теме, но и суметь изложить ее так, чтобы было понятно другому. В рамках обогащающей модели обучения учащиеся будут психологически подготовлены к этой работе, так как они уже привыкнут иметь дело с вариативными и неопределенными ситуациями, прогнозировать, анализировать и оценивать свои интеллектуальные действия, доверять собственной интуиции и т. д.

Итак, поскольку процесс образования понятий имеет одно из важнейших значений в развитии интеллекта, необходимо учитывать его психологические особенности при обучении. Поэтому, как считают представители обогащающей модели обучения, формирование понятийных структур, выступающих в

качестве носителей понятийного знания, должно учитывать три основных аспекта. Во-первых, необходимо помнить, что понятийные психические структуры – это интегральные когнитивные образования, включающие в себя словесно-речевой, визуальный и чувственно-сенсорный компоненты. Значит, необходима специальная система заданий, ориентированных на разные составляющие понятийных образований. Во-вторых, процесс обучения должен учитывать фазовую динамику процесса образования понятий (учет основных фаз движения мысли), в том числе: мотивировку, категоризацию, обогащение, перенос, свертывание. Наконец, нужно создавать условия для интеллектуальной самодеятельности учащихся в ходе порождения новых понятий. Учет и выполнение всех этих требований позволит более эффективно организовать процесс овладения понятиями, а значит, и интеллектуальное развитие учащихся.

Примечания

1. *Холодная М. А.* Психология интеллекта. Парадоксы исследования. СПб.: Питер, 2002. С. 108.
2. *Выготский Л. С.* О психологических системах // Собр. соч. Т. 1. М.: Педагогика, 1982. С. 120.
3. *Harvey O. J., Hunt D. E., Schroder H. M.* Conceptual system and personality organization. N. Y.: Wiley Sons, 1961. P. 7.
4. *Веккер Л. М.* Психика и реальность: единая теория психических процессов. М.: Смысл, 1998. С. 661-662.
5. *Rimat F.* Intelligent zu Untersuchungen ansschliessend und die Ach'sche Suchmethode. Leipzig, 1925.
6. *Пиаже Ж.* Психология интеллекта // Избранные психологические труды. М.: Просвещение, 1969.
7. *Выготский Л. С.* Мышление и речь // Собр. соч. Т. 2. М.: Педагогика, 1984. С. 130.
8. *Выготский Л. С.* О психологических системах... С. 121.
9. *Выготский Л. С.* Мышление и речь... С. 131.
10. *Выготский Л. С.* Психология подростка // Собр. соч. Т. 4. М.: Педагогика, 1982. С. 71.
11. Там же. С. 65.
12. *Веккер Л. М.* Психические процессы. Мышление и интеллект. Т. 2. Л.: Изд-во Ленингр. ун-та, 1976; *Холодная М.А.* Интегральные структуры понятийного мышления. Томск: Изд-во Том. ун-та, 1983.
13. *Холодная М. А.* Психология интеллекта... С. 119.
14. *Холодная М.А.* Интегральные структуры...
15. *Брунер Джс.* О познавательном развитии. Ч. 1, 2 // Исследование развития познавательной деятельности / Под ред. Дж. Брунер и др. М.: Педагогика, 1971.
16. *Выготский Л. С.* Мышление и речь... С. 119.
17. Там же. С. 127.
18. *Полани М.* Личностное знание: на пути к посткритической философии. М.: Прогресс, 1985.
19. *Веккер Л. М.* Психика и реальность...; *Он же.* Психические процессы...
20. *Пиаже Ж.* Указ. соч.
21. *Окулов С. М.* Когнитивная информатика. Киров: Изд-во ВятГГУ, 2003. С. 68.
22. *Холодная М. А.* Психология интеллекта...
23. *Крутецкий В.А.* Психология математических способностей. М.: Просвещение, 1968.
24. *Gardner H., Walters P.* Frames of mind: The theory of multiply intelligences. L.: Heinemann, 1986.

Н. И. Исупова

КОГНИТИВНЫЙ АСПЕКТ ИНТЕЛЛЕКТУАЛЬНОГО РАЗВИТИЯ УЧАЩИХСЯ ПРИ ОБУЧЕНИИ ИНФОРМАТИКЕ

В работе рассматривается онтологический подход к понятию интеллекта, согласно которому интеллект представляет собой особую форму организации ментального опыта человека. По этой теории, развитие интеллекта означает обогащение и наращивание умственного опыта. В статье приводятся основные линии такого обогащения и возможности их формирования средствами информатики.

Образование – это то, что остается, когда все выученное забудется.

Д. Гранин

Интеллектуальное развитие учащихся волновало российскую школу практически на протяжении всей истории ее развития. И в настоящее время эта проблема не теряет своей актуальности, более того, возникают все новые аспекты ее рассмотрения. Одним из таких аспектов является мысль, что для осуществления целенаправленного влияния на развитие интеллекта учащегося необходимо знать, как устроен человеческий интеллект и как он работает, иными словами, каким образом происходит познание человеком окружающей действительности. Этим вопросом и занимается когнитивная психология, исследующая принципы и методы, которыми управляется феномен человеческого познания. Познание охватывает ментальные процессы, такие, как восприятие, мышление, память, оценка, планирование и организация. Когнитивная психология объединяет в себе множество теорий, общей базой которых является изучение структуры интеллекта. Исследования когнитивных психологов приобрели совершенно новое звучание после того, как А. Ньюэллом и Г. Саймоном в 1958 году была высказана гипотеза о том, что интеллект можно рассматривать как систему обработки информации наподобие компьютера. Это породило лавину исследований и теоретических формулировок, основанных на компьютерных моделях. И нужно отметить, что за последние десятилетия когнитивная психология, применяя достижения в *Computer Science*, действительно продвинулась далеко вперед в понимании природы интеллекта. Но нам хотелось бы, если так можно выразиться, сделать обратный ход – используя результаты когнитивных психологов, понять, как деятельность в *Computer Science*, точнее, обучение информатике, влияет на развитие интеллекта.

ИСУПОВА Наталья Ивановна – аспирант кафедры информатики и МОИ ВятГГУ

В связи с этим, прежде всего, требуется определить, что такое интеллект и какова его структура. Существует множество различных взглядов, теорий относительно природы интеллекта. Таких подходов в настоящее время можно насчитать не менее десяти: это тестологический, феноменологический, генетический, социо-культурный, процессуально-деятельностный, образовательный, информационный, функционально-уровневый, регуляционный подходы и т. д. Все они с разных сторон выявляют и описывают достаточно богатую феноменологию проявления интеллектуальной деятельности, однако их взаимопересечения и взаимодополнения огромны. Это естественно при изучении сложнейшего явления: наука должна накопить фактографический материал. Но рано или поздно должен начаться процесс интеграции, обобщения всех имеющихся знаний, и именно это обобщение должно привести, или хотя бы приблизить, к единому и непротиворечивому пониманию природы интеллекта.

В качестве такой интегративной теории интеллекта может быть рассмотрена модель М. А. Холодной – Р. Стенберга – Л. М. Веккера, которая представляет собой так называемый онтологический подход. В рамках данного направления исследований делается попытка ответить не на вопрос «Что такое интеллект?» (с последующим перечислением его свойств), а на вопрос «Что представляет собой интеллект как психический носитель своих свойств?». Эта идея о неправомерности описания психической реальности через совокупность её свойств впервые сформулирована Л. М. Веккером [1]. По его мнению, изучать психические свойства можно до бесконечности, однако теоретического «перелома» – уяснение действительной природы изучаемого явления – при этом не возникает. Задача научного психологического анализа заключается в объяснении свойств исходя из особенностей устройства и функционирования их психического носителя. Следовательно, вся совокупность познавательных процессов, образующих состав интеллекта, должна рассматриваться как иерархия разноуровневых когнитивных структур, которые на основе когнитивного синтеза «снизу» и «сверху» образуют единую структуру человеческого интеллекта.

По своему назначению интеллект - это общая познавательная способность, которая проявляется, во-первых, в том, как человек воспринимает, понимает и объясняет происходящее, и, во-вторых, в том, какие решения он принимает и насколько эффективно действует в той или иной ситуации. *«Интеллект – это особая форма организации индивидуального ментального опыта в виде наличия ментальных структур, порождаемого ими ментального пространства отражения и строящихся в рамках этого пространства ментальных репрезентаций происходящего»* [2]. Свойства интеллектуальной деятельности (как измеряемые с помощью психодиагностических методик, так и проявляющихся в условиях реальной жизнедеятельности) производны по отношению к особенностям состава и строения ментального опыта субъекта. Ментальный опыт – это система наличных психических образований и инициируемых ими психических состояний, лежащих в основе познавательного отношения человека к миру и обусловливающих конкретные свойства его интеллектуальной деятельности. Узкая трактовка опыта как чувственно-эмпирической формы познания действительности или сведение его к знаниям, умениям, навыкам, как отмечал Д. Н. Завалишин, является неприемлемой. Он писал: «... опыт человека перестает выступать как второстепенный компонент интеллекта... но скорее становится его ведущим компонентом, потенциальным резервуаром новых операциональных и предметных знаний, зачастую всплывающих в затрудненных условиях деятельности в виде неинструментальных сигналов и интуитивных механизмов» [3]. *Ментальные структуры* – относительно устойчивые психические образования, обеспечивающие в процессе познания возможность работы с информацией (поступление, преобразование, переработку). *«Ментальное пространство* – это субъективный диапазон отражения, в рамках которого возможны разного рода мысленные перемещения. Ментальное пространство представляет собой динамическую форму ментального опыта, поскольку оно, во-первых, развертывается наличными ментальными структурами в условиях актуального интеллектуального взаимодействия субъекта с миром и, во-вторых, обладает способностью к одномоментному изменению своей топологии и метрики под влиянием субъективных и объективных факторов» [4]. *«Ментальная репрезентация* – это актуальный умственный образ того или иного конкретного события, то есть субъективная форма видения происходящего. Это не та или иная форма хранения знаний (в виде прототипа, следов памяти, фрейма и т. д.), а инструмент приложения знаний к определенному частному аспекту действительности. Ментальная репрезентация – это конструкция, зависящая от обстоятельств и построенная в конкретных условиях для специфических целей. Она является оперативной формой ментального опыта и модифицируется по мере изменения ситуации и интеллектуальных усилий субъекта, являясь специализированной и детализированной умственной картиной события» [5].

Таким образом, развитие интеллекта (интеллектуальной одаренности) есть процесс построения, конструирования индивидуального ментального опыта. А наша задача заключается в том, чтобы при обучении информатике создать среду, условия для «произрастания», становления талантов и выявить закономерности в формировании ментального опыта. Итак, основная идея состоит в том, чтобы сделать обучение информатике обогащающим, то есть направленным на интеллектуальное воспитание учащихся за счет актуализации и усложнения ментального (умственного) опыта ребенка. Это означает, что процесс

6 Зак. 2868

обучения должен, во-первых, содействовать учету и формированию основных компонентов ментального опыта учащихся, и, во-вторых, позволять детям с разными типами ментального опыта (в том числе с разными познавательными стилями) выбирать наиболее подходящую для себя линию обучения.

Обогащение ментального опыта целесообразно проводить по основным линиям, которые соответствуют его строению. Анализ ментальных структур позволяет выделить три уровня (или слоя) опыта, каждый из которых имеет свое назначение.

1. *Когнитивный опыт* – это ментальные структуры, которые обеспечивают хранение, упорядочение и преобразование наличной и поступающей информации, способствуя тем самым воспроизведению в психике познающего субъекта устойчивых, закономерных аспектов его окружения. Их основное назначение - оперативная переработка текущей информации об актуальном воздействии на разных уровнях познавательного отражения.

2. *Метакогнитивный опыт* – это ментальные структуры, позволяющие осуществлять непроизвольную и произвольную регуляцию интеллектуальной деятельности. Их основное назначение – контроль за состоянием индивидуальных интеллектуальных ресурсов, а также за процессами переработки информации.

3. *Интенциональный опыт* – это ментальные структуры, которые лежат в основе индивидуальных интеллектуальных склонностей. Их основное назначение — формирование субъективных критериев выбора относительно определенной предметной области, направления поиска решения, источников информации и способов ее переработки и т. д.

Соответственно к оценке индивидуального интеллекта следует подходить, одновременно принимая во внимание четыре аспекта его работы (с учетом четырех горизонтальных уровней представленной модели):

- как человек перерабатывает поступающую информацию (I уровень),
- может ли он контролировать работу своего интеллекта (II уровень),
- почему именно так и именно об этом он думает (III уровень),
- как он использует свой интеллект (IV уровень).

Коротко остановимся на психологической характеристике особенностей организации и обогащения (в условиях обучения информатике) каждого из трех слоев ментального опыта.

1. Когнитивный опыт

К ментальным структурам, образующим состав когнитивного опыта, можно отнести способы кодирования информации, когнитивные схемы, семантические структуры и, наконец, понятийные структуры как результат интеграции вышеуказанных базовых механизмов переработки информации.

Способы кодирования информации – это субъективные средства, с помощью которых развивающийся человеческий индивид представляет (отображает) в своем опыте окружающий мир и которые он использует в целях организации этого опыта для будущего поведения. Психологическое исследование способов кодирования информации впервые было предпринято Дж. Брунером [6], который говорил о существовании трех основных способов субъективного представления мира: в виде действий, наглядных образов и языковых знаков. То есть когда мы нечто понимаем, мы это словесно определяем, мысленно видим и чувствуем. Если проводить аналогию с программированием, точнее, со структурным программированием, то можно отметить, что «структурная программа воспринимается не как сплошной текст, а как некий единый паттерн» [7]. Это утверждение основано на практических наблюдениях за тем, как проученный школьник анализирует программы. Очень быстро формируется общая структура программы в кратковременной зрительной памяти. Строится как бы дерево решения, в котором отражены все взаимосвязи процедур, функций, их вложенности, выделяются особенности программы. То есть структурный стиль написания программ соответствует когнитивным характеристикам восприятия человека, и максимальный эффект достигается за счет того, что воспринимается не отдельное слово, не отдельный рисунок, а нечто целое, сочетающее в себе и слово, и образ, и действие.

Обогащению когнитивного опыта учащихся в этом аспекте способствует и то, что при обучении информатике они приобретают опыт использования разных способов кодирования информации. Так, при написании программы ученик постоянно осуществляет перевод с родного языка на язык программирования, что содействует овладению *словесно-символическим* способом кодирования информации. *Визуальный* способ кодирования информации учащиеся осваивают, например, при оформлении алгоритма в виде блок-схемы или при решении задач на перебор, в которых часто используется графическое изображение переборной схемы, а также при составлении алгоритмов на графах, где используется перевод словесно-символьной информации в визуально-образную. Овладению *предметно-практическим* способом кодирования информации способствуют задания на привлечение жизненного опыта учащихся, задачи с практическим содержанием. Например, когда младшие школьники работают в среде ЛОГО, им предлагается выполнять конкретные действия: поиграй в Черепашку, чтобы понять, как бы ты это сделал сам, то есть какие-либо абстрактные понятия дети познают через знакомые им ситуации и действия. При обучении информатике происходит и развитие *чувственно-сенсорного* способа кодирования информации благодаря наличию в учебном процессе «красивых задач» (например, задача о ханойских башнях или «гипотеза Сиракуз»), рациональных или громоздких решений одной и той же задачи, а также «красивых» идей (одной из которых является идея рекурсии). С. Пейперт писал: «Из

всех идей, с которыми я познакомил детей, рекурсия вызвала у них особенно сильную реакцию. Я думаю, это происходит отчасти потому, что эта идея захватила детскую фантазию, а отчасти и потому, что она уходит своими корнями в народную культуру... Подобные изображения приводили детей в состояние шока, и они часто проводили долгие часы за занятиями, в которых размышления над числами и геометрическими фигурами переплетались с эстетическими размышлениями» [8]).

Когнитивная схема - это обобщенная и стереотипизированная форма хранения прошлого опыта относительно строго определенной предметной области (знакомого объекта, известной ситуации, привычной последовательности событий и т. д.). Когнитивные схемы, таким образом, отвечают за прием, сбор и преобразование информации в соответствии с требованием воспроизведения устойчивых, нормальных, типичных характеристик происходящего (в том числе прототипы, предвосхищающие схемы, когнитивные карты, фреймы, сценарии и т. д.).

Одной из когнитивных схем является прототип – это когнитивная структура, которая воспроизводит типичный пример данного класса объектов или пример определенной категории. Так, исследования показали, что у большинства испытуемых наиболее типичным примером для категории «мебель» является «стул», а наименее типичным – «телефон»; для категории «фрукт» – «апельсин» и «фруктовое пюре» соответственно; для категории «транспорт» – «автомобиль» и «лифт» соответственно [9]. Таким образом, прототип – это обобщенное визуальное представление, в котором воспроизведен набор общих и детализированных признаков типичного объекта и которое выступает в качестве основы для идентификации любого нового впечатления или понятия.

Что будет воспринято и какова будет первичная интерпретация воспринятого, определяется, в частности, и такой разновидностью когнитивных схем, как фреймы [10]. Фрейм является формой хранения стереотипных знаний о некотором классе ситуаций: его «каркас» характеризует устойчивые, всегда имеющие место отношения между элементами ситуации, а «узлы» (или «слоты») этого каркаса – вариативные детали данной ситуации. При извлечении наличного фрейма он оперативно приводится в соответствие с характеристиками ситуации путем дозаполнения своих «узлов» (например, фрейм жилой комнаты имеет некоторый единый каркас в виде обобщенного представления о жилой комнате вообще, узлы которого каждый раз, когда человек воспринимает жилую комнату или думает о ней, могут заполняться новой информацией). Согласно Минскому, если мы говорим о человеке, что он умен, то это означает, что он обладает способностью чрезвычайно быстро выбирать в сложившихся обстоятельствах наиболее подходящий фрейм.

Говоря в связи с этим о программировании, уместно вспомнить объектно-ориентированное программирование, в котором понятие «объект», или «класс», ассоциируется с вышеупомянутыми фреймами, а основные идеи объектно-ориентированного программирования – полиморфизм и наследование – напоминают процесс извлечения фреймов из памяти при интеллектуальной деятельности.

Еще одним компонентом когнитивного опыта, согласно рассматриваемой модели, являются *семантические структуры*. В процессе взаимодействия со своим окружением у человека формируется особый механизм отражения действительности — индивидуальная система значений. Все элементы мира, с которыми человек в свое время непосредственно сталкивался, о которых ему рассказывали и о которых он задумывался когда-либо сам, начинают для него нечто значить: человек знает значение вещей, жестов, слов, событий и т. п. Таким образом, семантические структуры – это индивидуальная система значений, характеризующая содержательный строй индивидуального интеллекта. Благодаря этим психическим образованиям знания, будучи представленными в ментальном опыте конкретного человека в специфически организованном виде, оказывают активное влияние на его интеллектуальное поведение.

В целом ряде исследований было показано, что индивидуальная система значений на уровне вербальных и невербальных семантических структур обнаруживает себя в экспериментальных условиях в виде устойчивых словесных ассоциаций, «семантических полей», «вербальных сетей», «семантических, или категориальных, пространств», «семантико-перцептивных универсалий» и т. п. В частности, эксперименты А. Р. Лурии и О. С. Виноградовой [11] показали не только наличие определенных семантических структур в виде «семантических полей» с выделением в последних «семантического ядра» и «семантической периферии», но и тот факт, что сами испытуемые не осознавали столь очевидных и устойчивых межсловесных связей. Принцип организации и функционирования «вербальной сети» таков, что активизация основного слова приводит к одновременной, последовательной либо избирательной актуализации других элементов этой вербальной сети. Это говорит о том, что мы храним выделенные путем абстрагирования общие свойства явления, объектов, изученных примеров. Но то же самое мы видим и в программировании, где данные и действия с данными интегрированы в единое целое (понятие запись в среде программирования Pascal, понятие объекта в среде программирования Delphi). Все это говорит о том, что «понимание когнитивными психологами вопросов представления информации в мозгу человека практически полностью соответствует тому, что наработано в *Computer Science* для описания логики функционирования информационных систем, в частности описа-

нию данных и действий в программировании. Это положение подтверждает мысль о том, что занятия программированием соответствуют когнитивной сущности человека» [12].

Центральную же роль в становлении интеллекта играют *понятийные структуры*, так как понятийные структуры (концепты), включая в себя все вышеперечисленные уровни когнитивных структур, выступают в качестве «формы интегральной работы интеллекта», а сам концепт выступает как «интеллектообразующая интегративная единица» [13]. «Понятийные психические структуры – это интегральные когнитивные структуры, особенности устройства которых характеризуются включенностью разных способов кодирования информации, представленностью визуальных схем разной степени обобщенности и иерархическим характером организации семантических признаков» [14]. Поэтому к ним в полной мере относятся все вышеупомянутые примеры и аналогии относительно программирования.

2. Метакогнитивный опыт

Метакогнитивный опыт – это психические механизмы, обеспечивающие управление собственной интеллектуальной деятельностью (в том числе непроизвольный и произвольный интеллектуальный контроль, метакогнитивная осведомленность, открытая познавательная позиция). *Контроль за работой собственного ума* предполагает способность к непроизвольной и произвольной саморегуляции своей интеллектуальной деятельности. Эта функция как нельзя лучше может быть реализована при обучении информатике, так как здесь появляется новое средство обучения – компьютер. И важно отметить, что этот инструмент является активным, это своего рода соучитель, который является активизатором обратной связи в цепочке «учитель – ученик». Компьютер служит помощником ученику в обогащении его метакогнитивного опыта: контролируя и оценивая работу написанной программы, ученик в некоторой степени контролирует и работу собственного ума, а работа по нахождению ошибок в алгоритме формирует умение видеть свои ошибки, выяснять их причины, предупреждать появление новых ошибок и т.д. Кстати сказать, любая работа за компьютером позволяет ошибаться, и это хорошо. «Позволяя ошибаться, разрешая ошибаться, создавая возможности ошибаться, мы даем возможность познавать через противоречия, ибо ошибка – это источник противоречия» [15]. При изучении информатики бессмысленно заниматься оправданием своих ошибок, так как для компьютера это не имеет никакого значения: программа от этого правильно работать не будет. Ошибки требуется искать, исправлять и не повторять. В результате ученик будет относиться гибче и к мнению окружающих, и к противоположным точкам зрения – искать в них рациональное зерно, то есть совершенствовать свой интеллект.

Обогащение метакогнитивного опыта учащихся предполагает также формирование их *метакогнитивной осведомленности* – системы представлений о том, как устроены научные знания и каковы особенности разных методов познания, сведений о своих собственных качествах ума и способах их эффективного использования. Интеллектуальное развитие ребенка предполагает не только усвоение знаний «о том, что» и знаний «о том, как», но и знаний «о том, какой Я». При программировании четко прослеживается, что Я как действующий за компьютером, знаю, что Я понимаю. Без сосредоточения на собственном мыслительном процессе, на результатах собственного мышления хорошую, правильную программу не сделать. Еще одним инструментом формирования метакогнитивного опыта учащихся является этап тестирования программ. На каждом шаге работы ученик «имеет возможность осознавать (оценивать), насколько правильно принятое решение, насколько верен ход рассуждений, все ли факторы учтены при принятии решения и т. д.» [16]. Кроме того, повысить уровень метакогнитивной осведомленности учащихся можно, включив в учебные пособия специальные разделы под названием «Психологический комментарий», в каждом из которых излагаются общие сведения об определенных проявлениях человеческого интеллекта с использованием простейших процедур интеллектуальной самодиагностики и интеллектуального тренинга. Например, темой таких разделов может быть рассмотрение основных интеллектуальных способностей (способность оперировать образами, способность к запоминанию, способность выполнять мыслительные операции, способность быть внимательным и т. д.). В процессе работы с такими психологическими разделами создаются условия для того, чтобы ребенок мог достаточно быстро почувствовать эффект усиления того или иного интеллектуального свойства (в виде увеличения объема запоминания при опоре на смысловые связи, большей легкости понимания научных понятий при использовании «своего» познавательного стиля, умения преодолевать психологическую инерцию собственного мышления и т. д.). Предполагается, что и при проработке материала по предмету эти проявления роста метакогнитивной осведомленности будут закрепляться и использоваться.

Еще одним компонентом метакогнитивного опыта является *открытая познавательная позиция*. Она предполагает вариативность и разнообразие способов анализа происходящего, а также готовность воспринимать необычную, парадоксальную, «невозможную» информацию. Деятельность при программировании удовлетворяет и этому условию, так как характеризуется поиском различных вариантов решения задач. Это естественное качество работы программиста, так как у каждой программы есть ограничения и она создается и использованием ограниченного инструментария. Например, изменение размерности входных данных требует, как правило, поиска других методов решения. Кроме того, если задача решается в классе, то происходит обмен идеями, методами решения между школьниками. Ищется наилучший вариант решения, оценивается время его работы и т. д. А значит, развиваются умения слушать и слышать другого, воспринимать и уважать альтернативное мнение, уметь отстаивать свою точку зрения и принимать точку зрения оппонента.

Формированию открытой познавательной позиции способствуют также задания и тексты:

- дающие учащимся возможность осознать существование нескольких подходов к одной и той же ситуации и работать в рамках разных, в том числе альтернативных подходов;
- содержащие противоречивые данные;
- развивающие способность воспринимать неожиданную информацию;
- стимулирующие готовность принимать и обсуждать необычные идеи;
- дающие возможность видеть перспективу в изучении информатики и обращаться к уже изученному материалу с новой точки зрения, и т. д.

3. Интенциональный опыт

Интенциональный опыт — это психические механизмы, предопределяющие избирательность индивидуальной интеллектуальной деятельности (в том числе интеллектуальные предпочтения, убеждения, умонастроения). Эти механизмы формируются на основе так называемых интеллектуальных интенций, то есть особых субъективных состояний (состояний направленности ума), которые по своим механизмам являются продуктом эволюции индивидуального ментального опыта, а по форме выражения - неопределенно переживаемыми и в то же время чрезвычайно устойчивыми чувствованиями.

Многие психологи и педагоги, основываясь на собственных наблюдениях, считают, что если интенциональный опыт ребенка игнорируется либо полностью отторгается, то темп интеллектуального развития школьника резко замедляется и, что самое печальное, снижается творческий потенциал ребенка. Что не удивительно, поскольку интенциональный опыт, как можно предположить, является одним из мощнейших источников интуиции.

Обогащению интенционального опыта способствует такой процесс обучения, который в той или иной мере активизирует участие в интеллектуальной работе ребенка его личных переживаний, сомнений, эмоциональных оценок, догадок и т. д. Об этом же говорил еще З. Фрейд, вводя понятие «синтонное я». Этот термин использовался им для описания инстинктов или представлений, приемлемых «я», то есть совместимых с целостностью «я», с его требованиями. «Синтонное я означает созвучность представлениям детей о себе как людях с определенными целями, намерениями, желаниями, симпатиями и антипатиями. На практике при изучении информатики это положение реализуется через использование среды ЛОГО, разработанной с участием С. Пейперта. Работа с Черепашкой в среде ЛОГО соответствует уровню развития ребенка. «Геометрия Черепашки» пригодна для изучения именно потому, что она синтонна. И она является средством изучения других вещей, поскольку поощряет детей к их сознательному, продуманному использованию при решении проблем.

Но важно также учитывать интенциональный опыт учащихся и при подборе учебного материала. В связи с этим научные сведения должны излагаться с использованием историко-культурных материалов, размышлений представителей других областей знаний. Учащимся должна предоставляться возможность получать новые знания, используя имеющиеся правила, алгоритмы, справочники; проводить самостоятельное исследование проблем, выдвигать гипотезы и проверять их. Особое внимание должно уделяться актуализации интуитивного опыта детей: должны поощряться высказывания ими своих личных убеждений, «опережающих» идей, эмоциональное отношение к учебному материалу и т. д.

Такую возможность в полной мере предоставляет занятие программированием. При решении творческих задач деятельность учащегося развивается по плану: за постановкой задачи следует гипотеза и разработка первого варианта программы. Затем она подвергается исследованию, экспериментальной проверке с помощью системы тестовых проверок – сравнению ожидаемых результатов и полученных. Ученику мысленно следует предсказать, предвидеть результаты работы. Наступает фаза или экспериментального опровержения, или экспериментального подтверждения. Дальнейшее исследование задачи с помощью компьютера – это неоднократные попытки написания различных версий программ, проведение экспериментов, возврат на начальные стадии работы, возникновение новых проблем и т. д. При попытке разрешить неясную, незавершенную ситуацию происходит очередная мобилизация творческих сил. Главным для учителя в триаде «ученик – учитель – компьютер» становится не изложение новых истин, а поддержание у ученика состояния творческой напряженности, состояния поиска. Из такой специфики действий, особенностей познавательного процесса в связке «человек - компьютер» вытекают отличия школьного предмета «информатика» по своим дидактическим возможностям для развития интеллекта учащегося от всех остальных школьных предметов, включая математику.

Вторым аспектом обогащения ментального (умственного) опыта учащихся – наряду с формированием основных компонентов когнитивного, метакогнитивного и интенционального опыта – является *создание условий для раскрытия и роста индивидуального своеобразия склада ума учащихся*. Таким образом, индивидуализация обучения – это важнейшее средство интеллектуального воспитания учащихся, поскольку помогает учителю увидеть в каждом ученике уникальность его интеллектуальных возможностей. Дело в том, что в процессе обучения школьникам постоянно требуется обратная связь по оценке резуль-

татов их деятельности, постоянный контроль за правильностью действий, указание на то, в чем заключается их ошибка и т. д. Поэтому эффективны любые формы индивидуального обучения. Но для их реализации требуется инструмент, который способствовал бы формированию ментального опыта учащихся в выбранной сфере деятельности. В традиционной педагогике таким инструментом выступает учитель, в нашем случае (при обучении информатике) этот инструмент – сочетание «учитель – компьютер». Именно в этом, еще раз подчеркнем, дидактическая сила нового школьного предмета – информатики.

В общем случае индивидуализация обучения информатике предполагает:

• учет индивидуальных интеллектуальных особенностей детей с последующей адаптацией учебного процесса (в том числе учет индивидуальных познавательных склонностей, предпочитаемых способов познания, избирательности в самостоятельном изучении тех или иных тем, выборе наиболее подходящих форм контроля, степени сложности заданий и т. д.);

• оказание каждому ребенку индивидуализированной педагогической помощи с целью развития его исходных психологических возможностей (в том числе создание условий для проявления присущих разным детям разных познавательных стилей, текущая учебная диагностика уровня обученности каждого ребенка, формирование навыков самообучения и т. д.).

Таким образом, если основной целью при обучении информатике считать интеллектуальное развитие учащихся, то, по мнению когнитивных психологов (а именно: М. А. Холодной, Р. Стенберга, Л. М. Векера), необходимо создать условия, обеспечивающие формирование их ментального опыта. Деятельность, связанная с программированием, как показано выше, положительно влияет на структурные элементы (когнитивный, метакогнитивный, интенциональный) ментального опыта, что дает основания утверждать о возможности построения процесса обучения информатике по типу обогащающего обучения.

Результатом такой организации обучения должен стать рост интеллектуальных возможностей каждого ребенка. При этом встает вопрос о том, что является критериями интеллектуальной воспитанности, как оценить степень развития интеллекта? В рамках традиционной школы годами формировалось убеждение, что главный показатель эффективности школьного обучения – это уровень усвоения знаний, умений, навыков. Те самые ЗУН (знания – умения – навыки), вокруг которых так или иначе строится и система контроля успеваемости школьников, и система аттестации учительских кадров.

Несомненно, ЗУН – важная сторона тех изменений, которые происходят с учениками на всем протяжении школьного обучения. Вопрос в другом: достаточно ли ЗУН для реализации задач интеллектуального воспитания учащихся? Думается, что нет.

Что меняется в человеке, если он интеллектуально (умственно) воспитан? Меняется, по-видимому, характер познавательного отношения к миру: то, как человек воспринимает, понимает и объясняет происходящее. Таким образом, интеллектуальное воспитание заключается не только в формировании системы знаний, умений и навыков или развитии теоретического мышления, но, скорее, в обогащении индивидуального ментального (умственного) опыта ребенка, которое и выступает в качестве психологической основы интеллектуального роста личности.

Следовательно, чем выше уровень интеллектуального развития человека, тем более субъективно богатой и в то же время объективированной является его индивидуальная «картина мира». Соответственно в качестве показателей интеллектуальной зрелости (воспитанности) можно рассматривать характеристики индивидуального умозрения (или типа репрезентации происходящего). В частности, такие:

• широта умственного кругозора (в противовес «закапсюлированному» мировосприятию);

• гибкость и многовариантность оценок происходящего (в противовес «черно-белому мышлению»);

• готовность к принятию необычной, противоречивой информации (в противовес догматизму);

• умение осмысливать происходящее одновременно в терминах прошлого (причин) и в терминах будущего (последствий) (в противовес склонности мыслить в терминах «здесь-и-теперь»);

• ориентация на выявление существенных, объективно значимых аспектов происходящего (в противовес субъективированной, эгоцентрической познавательной позиции);

• склонность мыслить в категориях вероятного в рамках ментальной модели «как если бы» (в противовес игнорированию возможности существования «невозможных» событий);

• способность мысленно видеть отдельное явление в контексте его целостных связей с множеством других явлений (в противовес однолинейному взгляду на мир) и т. д.

С учетом сказанного в образовательном процессе на первый план – наряду с ЗУН выходит проблема формирования базовых интеллектуальных качеств личности, таких, как компетентность, инициатива, творчество, саморегуляция и уникальность склада ума (КИТСУ). Коротко сущность этих показателей интеллектуального развития можно выразить так:

Интеллектуальная компетентность – это особый тип организации знаний, обеспечивающий возможность принятия эффективных решений в конкретной области деятельности.

Интеллектуальная инициатива – это желание самостоятельно, по собственному побуждению отыскивать новую информацию, выдвигать те или иные идеи, осваивать другие области деятельности.

Интеллектуальное творчество – это процесс создания субъективно нового, основанный на способности порождать оригинальные идеи и использовать нестандартные способы деятельности.

Интеллектуальная саморегуляция – это умение произвольно управлять собственной интеллектуальной деятельностью и, главное, целенаправленно строить процесс самообучения.

Уникальность склада ума – это индивидуально своеобразные способы интеллектуального отношения к происходящему, в том числе индивидуализированные формы взаимокомпенсации слабых и сильных сторон своего интеллекта, выраженность индивидуальных познавательных стилей, сформированность индивидуальных интеллектуальных предпочтений.

Таким образом, КИТСУ – это определенная система показателей интеллектуального развития личности, в которых отражаются особенности индивидуального ментального опыта и которые характеризуют уровень развития индивидуальных интеллектуальных возможностей.

В качестве главного вывода статьи мы приводим наше предположение (гипотезу). По нашему мнению (а оно основывается, как было показано, на основных положениях когнитивной психологии), реализация обогащающего обучения информатике позволит выстроить систему индивидуальных интеллектуальных средств, способствующих росту интеллектуальных возможностей каждого ребенка. В частности, можно будет обеспечить обогащение индивидуального ментального опыта в направлении формирования его когнитивных, метакогнитивных и интенциональных компонентов, а также за счет создания условий для роста индивидуального своеобразия склада ума. Подобного рода обогащение ментального опыта учащихся приведет к тому, что их индивидуальные интеллектуальные возможности к концу завершения образования в средней школе будут в той или иной мере отвечать КИТСУ-критериям (критериям компетентности, инициативы, творчества, саморегуляции, уникальности склада ума).

Примечания

1. *Веккер Л. М.* Психические процессы. Мышление и интеллект. Т. 2. Л.: Изд-во Ленингр. ун-та, 1976.
2. *Холодная М. А.* Психология интеллекта. Парадоксы исследования. Спб.: Питер, 2002.
3. *Завалишин Д. Н.* Психологический анализ оперативного мышления. М.: Наука, 1985.
4. Когнитивная психология / Под ред. В. Н. Дружинина, Д. В. Ушакова. М.: ПЕР СЭ, 2002.
5. *Ришар Ж. Фр.* Ментальная активность. Понимание, рассуждение, нахождение решения. М.: ИП РАН, 1998.
6. *Брунер Дж.* О познавательном развитии. Ч. 1, 2 // Исследование развития познавательной деятельности / Дж. Брунер, Р. Олвер, П. Гринфилд. М.: Педагогика, 1971; *Он же.* Психология познания. М.: Прогресс, 1977.
7. *Окулов С. М.* Когнитивная информатика: Монография. Киров: Изд-во ВятГГУ, 2003. С. 136.
8. *Пейперт С.* Переворот в сознании: Дети, компьютеры и плодотворные идеи. М.: Педагогика, 1989.
9. *Rosch E.* Natural categories // Cognitive Psychology. V. 4. 1973. P. 326-350; *Rosch E.* Principles of categorization // Cognition and categorization / Eds. E. Rosch, B. L. Lloyd. N. Y.: Lawrence Erlbaum association, 1978. P. 27-48.
10. *Минский М. С.* Структура для представления знания // Психология машинного зрения / Пер. с англ. М.: Мир, 1978. С. 249-320.
11. *Лурия А. Р., Виноградова О. С.* Объективное исследование динамики семантических систем // Семантическая структура слова. М.: Наука, 1971.
12. *Окулов С. М.* Указ. соч. С. 159.
13. *Веккер Л. М.* Психика и реальность: единая теория психических процессов. М.: Смысл, 1998.
14. Там же.
15. *Окулов С. М.* Указ. соч. С. 158.
16. *Окулов С. М.* Указ. соч. С. 127-128.

А. В. Козвонина

О МЕТОДИКЕ ФОРМИРОВАНИЯ ПОНЯТИЙ

Как формируются понятия при обучении и что такое понятийное мышление? В статье сделана попытка ответить на эти вопросы и рассмотреть методику работы с понятиями на примере материала конкретного учебника.

На примере работы [1] в данной статье проводится анализ методики формирования понятий. Определим, что такое «понятие» и каким образом его можно сформировать.

В целом мышление в своем становлении проходит две стадии: допонятийную и понятийную (не будем останавливаться на понятии мышления, так как до сих пор отсутствует единое понимание его сути). Согласно Ж. Пиаже [2], допонятийное мышление присуще ребенку до 5 лет. Оно характеризуется нечувствительностью к противоречиям, синкретизмом (тенденцией связывать все со всем), трансдукцией (переходом от частного к частному), отсутствием представления о сохранении количества.

Понятийное мышление развивается постепенно от простого складывания ребенком предметов через установление сходства и различия между ними до собственно понятийного, которое формируется к 16-17 годам.

В качестве основных признаков понятийного мышления можно выделить следующие [3].

1. Допонятийное мышление отличается эгоцентризмом, понятийное же мышление способно к интеллектуальной децентрации, то есть на стадии понятийного мышления человек в состоянии с помощью вербальных операций встать на любую точку зрения, видеть и понимать предмет под любым углом зрения, в любой «системе мыслительных координат».

КОЗВОНИНА Анастасия Валерьевна – аспирант кафедры информатики и МОИ ВятГГУ

2. Для понятийного мышления характерна согласованность объема и содержания понятий (об объеме и содержании понятий – см. ниже), понятийные структуры выступают как собственно строгие логические классы. В предпонятийной мысли содержание и объем не согласованы.

3. Понятийное мышление использует правильно построенные индуктивно-дедуктивные умозаключения.

4. В понятийном мышлении господствует иерархизированность мыслительных конструкций. Понятия образуют определенную иерархию по степени общности от конкретных единичных понятий к категориям.

5. Существенным признаком понятийного мышления является полнота понятия и высший его уровень. На стадии понятийного мышления достигается осознание понятий (Л. С. Выготский). Это означает, что субъект отдает себе отчет о содержании и объеме используемых понятий, их отношениях друг к другу, операциях над ними. При этом осмысливается не только мыслительно отображаемая реальность, но и собственная структура и состав операций мышления.

Понятийное мышление является высшим уровнем понимания и высшей ступенью отделения субъекта от объекта познания [4]. Оперирование понятиями как строгими логическими классами позволяет преодолеть ограниченность допонятийной мысли. Понятийное мышление оказывает регулирующее влияние на низлежащие когнитивные уровни, перестраивая их работу и обеспечивая более высокий уровень понимания на этих уровнях.

Мыслительный процесс человека осуществляется в двух основных формах: формирование и усвоение понятий, суждений и умозаключений и решение проблем (мыслительных задач).

Что же такое понятие? В его определении нет однозначности. С одной стороны, понятие – это форма мышления, отражающая существенные свойства, связи и отношения предметов и явлений, которая выражена словом или группой слов (или же, пользуясь определением В. Ю. Лысковой и Е. А. Ракитиной, понятие – это «форма мышления, в которой отражаются отличительные существенные признаки предметов»). С другой стороны, понятие – это внутреннее, психологическое представление общих (существенных) свойств (ведь нигде в мире, кроме как в сознании людей, не определены понятия каких-либо предметов или явлений).

Существенными называются такие свойства, каждое из которых, взятое отдельно, необходимо, а всех вместе достаточно, чтобы с их помощью отличить данный предмет (явление) от всех остальных.

Понятие имеет две основные характеристики: содержание и объем. Содержание понятия – совокупность существенных признаков, отраженных в данном понятии, а объем понятия – это множество предметов, каждому из которых принадлежат признаки, составляющие содержание понятия.

Понятия принято различать по степени отвлеченности на конкретные и абстрактные. Когда из всех признаков предмета выделяется определенная совокупность признаков, характеризующая именно этот предмет или группу ему подобных, мы имеем дело с конкретным понятием (например, «город», «мебель»). Если же при помощи отвлечения в предмете выделяется определенный признак и этот признак становится предметом изучения и, кроме того, рассматривается как особый предмет, то возникает абстрактное понятие (например, «справедливость», «равенство»).

Формирование понятий – это «выявление свойств, общих для некоторого класса стимулов и обнаружение законов, связывающих эти свойства» [5]. Формирование понятий – это образование в сознании человека понятий, на основе которых затем организуются понятийные структуры.

Наличие сформированных понятий является важным условием обученности по всем школьным и вузовским предметам.

Как же сформировать *понятия*? Этот вопрос решают давно, решают относительно разных предметов и наук, но однозначно верного ответа на поставленный вопрос никто дать не может. Поэтому можно сказать, что существует множество методик формирования понятий и каждый педагог пользуется своей разновидностью методики. Выделим основные моменты в этом процессе (а формирование понятий есть процесс).

1. Поскольку понятие состоит из содержания и объема, обе эти составляющие и должны быть раскрыты учителем. В большинстве случаев, конечно, раскрываются они далеко не в полном объеме, но это не противоречит принципам обучения, в частности, принципу научности (в соответствии с которым, можно давать ученикам неполные знания, но обязательно истинные). Ведь нереально же, например, раскрыть полный объем понятия «алгоритм», но данное понятие должно быть сформировано у каждого ученика, изучающего информатику.

2. В отличие от объема, содержание понятия должно быть раскрыто практически полностью, т. е. необходимо изучить всю совокупность существенных признаков, отраженных в понятии. Кроме того, должна быть раскрыта связь формируемого понятия с уже сформированными.

3. Сложность формирования понятий информатики заключается в том, что большинство понятий информатики являются абстрактными. Ведь гораздо легче объяснить ученику, что такое «апельсин», чем дать понять, что есть «алгоритм».

4. Следующим шагом в формировании понятия является (при необходимости) его коррекция, а также его закрепление (осуществляемое различными способами).

Обратимся к работе [6]*. Данное пособие знакомит с одним из разделов информатики – элементами математической логики и логическими основами ЭВМ. Авторы рассматривают основные способы решения логических задач, алгоритмы построения таблиц истинности и получения совершенных дизъюнктивных и конъюнктивных нормальных форм по таблицам истинности. Кроме того, в книге разобрано построение функциональных схем для основных логических элементов ЭВМ. Пособие предназначено для учителей информатики, преподавателей, студентов.

По мнению авторов, «обучение школьников основам информатики, изучение ими такого важного понятия, как «алгоритм», невозможно без развития у них логического мышления, умения оперировать понятиями и символикой математической логики» [с. 4]. Предметным обоснованием к вводимым новым понятиям в книге являются логические содержательные задачи.

В конце книги приведена логическая схема понятий курса «Элементы математической логики и логические основы ЭВМ», делающая наглядной структуру курса.

Формирование практически любого понятия в анализируемой книге начинается с приведения четкого определения, за которым следуют поясняющие (раскрывающие) понятие примеры. Таким образом, раскрывается как содержание, так и объем понятия (естественно, частично).

В работе дается понятие простого высказывания. «Простым называется высказывание, которое не содержит в себе других высказываний.

Примеры простых высказываний:

1) Идет дождь.

2) Нам живется весело» [с. 40].

К сожалению, иногда приводятся простые, наивные примеры, слишком упрощающие раскрываемые понятия. К тому же некоторые определения весьма упрощены, примитивны.

В конце каждой главы имеется список основных понятий и определений. Кроме того, приводится ряд вопросов и заданий, с помощью которых осуществляется коррекция и закрепление сформированных понятий. Стоит отметить то, что задания весьма многочисленны, разнообразны и интересны.

Еще одним достоинством книги является то, что она богата наглядными иллюстрациями, допустим некоторых логических отношений, а также таблицами и схемами. Весьма эффективно используются различные шрифты, акцентирующие внимание на наиболее важных моментах.

Продемонстрируем это на небольшом примере [с. 16] (см. рисунок).

Авторами приведена система лабораторных занятий и контрольных работ, при выполнении которых закрепляются сформированные понятия. В конце книги приводятся ответы и решения.

В ряде случаев, когда есть необходимость разбора изучаемого материала с использованием языков программирования, например при программировании построения таблиц истинности, приводятся нужные программы на языках, наиболее распространенных с современной школе, – Паскаль и Бейсик.

Разберем процесс формирования понятий на примере понятия «логическое отрицание (инверсия)» по тем этапам, которые были перечислены выше.

Итак, во-первых, должны быть раскрыты объем и содержание понятия [с. 23-24].

«Логическое отрицание (инверсия) образуется из высказывания с помощью добавления частицы «не» к сказуемому или использования оборота речи «неверно, что…».

Далее приведены примеры образования логического отрицания (см. таблицу).

Кроме примеров авторы приводят некоторые их пояснения.

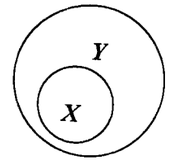
«А=У меня есть приставка Dendy – высказывание.

Пусть у Вас ее нет. Тогда это высказывание ложно (А=0). Инверсия А – это высказывание *У меня не есть приставка Dendy* или высказывание *Неверно, что у меня есть приставка Dendy*. Более правильным в русском языке является предложение *У меня нет приставки Dendy*, и это высказывание будет истинным».

Далее В. Лыскова и Е. Ракитина приводят способы обозначения инверсии, а также таблицу истинности с необходимыми пояснениями. Кроме того, авторы дают ученикам мнемоническое правило и графическую иллюстрацию инверсии с помощью диаграмм Эйлера – Венна [с. 24-25].

Высказывание А	**Значение высказывания А**	**Инверсия высказывания А**	**Значение инверсии высказывания А**
У меня есть приставка Dendy	0	У меня нет приставки Dendy	1
Я не знаю китайского языка	1	Неверно, что я не знаю китайского языка. (Я знаю китайский язык)	0

Подчинение
(*X* подчинен *Y*)



X – лев
Y – хищник

* Далее указываются страницы этого издания.

Благодаря описанным выше этапам формирования понятия, особенно примерам и графической иллюстрации, несколько снимается сложность этого процесса, связанная с абстрактностью понятия.

Как уже было написано выше, следующим шагом в формировании понятия является его коррекция и закрепление. Что касается коррекции, то она может происходить на этапе формирования следующих вводимых понятий, таких, например, как логическое умножение (конъюнкция), логическое сложение (дизъюнкция), логическое следование (импликация), логическое равенство (эквивалентность) и др., процесс формирования которых аналогичен процессу формирования понятия «инверсия». Кроме того, авторами построена сводная таблица логических функций двух переменных [с. 39].

Для закрепления формируемого понятия в приложении приводятся лабораторная работа по теме «Построение таблиц истинности с помощью электронных таблиц Excel» [с. 127], а также контрольная работа по теме «Элементы математической логики» [с. 133].

Таким образом, в целом методика формирования понятий, используемая В. Лысковой и Е. Ракитиной, традиционна, линейна, но детально продумана. К достоинствам ее можно отнести также использование компьютерных технологий при изучении практически всех вводимых понятий.

Мы рекомендуем использовать эту книгу всем учителям, преподающим в курсе информатики элементы математической логики и логические основы ЭВМ. Кроме того, советуем студентам и школьникам, интересующимся этими темами, познакомиться с книгой В. Лысковой и Е. Ракитиной, поскольку работать с пособием «Логика в информатике» можно и самостоятельно.

Примечания

1. *Лыскова В.Ю., Ракитина Е.А.* Логика в информатике. М.: Лаборатория Базовых Знаний, 2001. 160 с.

2. *Пиаже Ж.* Избранные психологические труды. Психология интеллекта. Генезис числа у ребенка. Логика и психология. М.: Просвещение, 1969. 659 с.; *Пиаже Ж., Инельдер Б.* Генезис элементарных логических структур. Классификация и сериация / Пер. с фр. Э. Пчелкиной. М.: Издво ЭКСМО-Пресс, 2002. 416 с.

3. Сознание и диалектика процесса познания. Межвуз. сб. науч. тр. Иваново, 1979. 164 с.

4. Психология и педагогика: Учеб. пособие / Отв. ред. В. М. Николаенко. М.: ИНФРА-М; Новосибирск: НГАЭиУ, 2000. 175 с.

5. *Солсо Р. Л.* Когнитивная психология / Пер. с англ. М.: «Тривола», М.: «Либерея», 2002. 600 с.

6. *Лыскова В. Ю., Ракитина Е. А.* Указ. соч.

Е. С. Лосева

О РЕАЛИЗАЦИИ ИДЕЙ ДОКТОРА СЕЙМУРА ПЕЙПЕРТА В ОБРАЗОВАТЕЛЬНОЙ ИНФОРМАТИКЕ

В статье показывается, что разработки С. Пейперта успешно используются в образовательной информатике многих стран. Вызывает удивление тот факт, что на официальном уровне в России они как бы не признаны.

За последние 20 лет электронная технология проникла практически во все сферы нашей жизни. Но эти бурные технологические изменения почти не повлияли на образовательные системы. Существует точка зрения, что за прошедшие 100 лет не произошло никаких существенных изменений в методах обучения, да и вообще в самой организации процесса обучения. С этим можно не согласиться, но нельзя отрицать, что необходимы улучшения в сфере обучения и образования, нужны новые учебные планы, новые исследования особенно в области применения компьютера в учебном процессе.

Одним из таких плодотворных исследований является программа, выполненная лабораторией Массачусетского технологического института под руководством профессора С. Пейперта, активно развиваемая в настоящее время.

Профессор-математик, доктор С. Пейперт известен как создатель компьютерного языка Лого и эксперт в области образования детей. Он посвятил годы исследований разработке методики, использования технологии Лого, позволяющей ребенку активно конструировать собственные знания.

Доктор С. Пейперт и его ученики (для профессора специально была создана кафедра) придерживаются теории образования, называемой «конструктивизм». В ее основе лежат две идеи понимания «конструкции» знания. Первая из них основана на том, что дети обучаются, активно конструируя новое знание, не имея никакой информации, заранее вложенной в их головы. Вторая утверждает, что эффективное обучение происходит, когда ученик вовлекает в конструирование новых знаний лично значимые артефакты, представляющие его собственное знание. Так, создание компьютерной мультипликации, роботов, игр или стихотворений лучше способствует процессу обучения, чем механическое выполнение упражнений.

С. Пейперт исходит из того, что использование очень мощной компьютерной техники и ее идей открывает новые возможности в учении, мышлении, в эмоциональном и когнитивном развитии. При этом компьютер используется не просто как вычислительное устройство, а скорее как устройство, позволяющее изменить стереотипы в усвоении знаний и в самом мышлении. Благодаря такому опыту человек уже в годы своего ученичества мог бы научиться думать о мышлении, анализировать его стили и приемы, то есть быть эпистемологом.

ЛОСЕВА Елена Сергеевна – ассистент кафедры прикладной математики ВятГГУ

Для развития этой способности группа С. Пейперта нашла необычный прием: с помощью компьютера и серии игр в реальном мире с реальными вещами создавались микромиры. Критериями построения таких микромиров стали возможность изучать на основе личного опыта и хорошо известной «геометрии собственного тела» законы движения, работая с простыми доступными примерами, осуществлять деятельность по этим законам и определять понятия на основе работы в этом микромире [1].

Деятельность в таких микромирах существенно стимулирует творчество учащихся, так как создает такую среду, в которой проблема истинности или ложности знания не является особо значимой, по сравнению с возможностями исследовать различные, в том числе «ошибочные» теории, которые категорически отвергает обычная школа. Но Ж. Пиаже, принципам которого следует С. Пейперт, показал, что ошибочные теории, в том числе создаваемые самими учащимися, являются своего рода «полигоном» мыслительной деятельности и потому могут и должны присутствовать в учебном процессе.

Построение и использование микромиров предполагает нетрадиционное использование компьютера для развития мышления и переработки информации. При таком использовании создается возможность поупражняться в качественном, по выражению С. Пейперта, мышлении, тогда как при стандартном использовании закрепляется количественная сторона знаний, поскольку компьютер позволяет производить сложные вычисления. За проблемой объединения качественной и количественной сторон знания стоит необходимость преодолеть разобщенность естественнонаучного и гуманитарного знания, а главным следствием этого процесса будет новый образ знания.

Таким образом, компьютер используется для выявления связи научного знания с личностным, для приближения научного знания к знанию человека, а не к знанию факта или владения навыком. «Переворот в сознании», о котором пишет С. Пейперт, состоит не просто в обращении к компьютеру, а в том, что открывается новый способ подхода к мышлению по типу компьютерного программирования [2].

Как уже упоминалось, учение С. Пейперта основывается на теории Ж. Пиаже о когнитивном развитии детей. Швейцарский психолог и философ Ж. Пиаже (1896-1980) известен как создатель концепции развития интеллекта и генетической эпистемологии. По образованию Ж. Пиаже – зоолог, и когда он изучал поведение человека, то пытался включить его в рамки более широкого контекста поведения других живых существ. Ключевым для него был вопрос о приспособлении (аккомодации) животных к среде обитания. Интеллект человека, таким образом, рассматривался как одно из средств обеспечения приспособления. Эти идеи воплотились в его операциональной концепции. Согласно ей, функционирование и развитие психики совершаются вследствие адаптации индивида к среде – ассимиляции нового наличными схемами поведения индивида и аккомодации этих схем к конкретным ситуациям. Высшей формой уравновешивания субъекта и объекта является образование операциональных структур. Операцией, по Пиаже, называется «внутреннее действие» субъекта, генетически производное от внешнего, предметного, действия и скоординированное с другими в определенную систему. Пиаже стремился открыть и объяснить, как происходит нормальное развитие человека. Он был убежден, что нормальный ход развития существует, то есть существует последовательность этапов, через которые, хотя и с разной скоростью, проходят все люди, причем одни проходят по этому пути дальше других. Согласно Ж. Пиаже существует четыре основные стадии развития интеллекта: сенсомоторная (от момента рождения до овладения языком 0–2 года), интуитивная, или предоперациональная (2–7 лет), конкретно-операциональная (7–12 лет), формально-операциональная (12–15 лет) [3].

В зарубежной литературе есть точка зрения, что концепция языка Лого может быть обоснована и в рамках других психологических теорий (теории психосоциального развития Эрика Эриксона, теории множественности интеллекта Говарда Гарднера и культурно-исторической теории развития психики Льва Выготского). Главным аргументом при этом является то, что в процессе работы с Лого происходит не только когнитивное развитие, то есть развитие мышления и мыслительных способностей [4].

Также Лого-программирование не менее эффективно помогает и психосоциальному развитию детей. В этом аспекте можно обнаружить удивительное соответствие Лого-программирования и теории психосоциального развития Эрика Эриксона. Например, работа с Лого помогает в стадиях психосоциального развития, обозначенных Э. Эриксоном как «инициатива (предприимчивость) – вина» (3–5 лет) и «трудолюбие – неполноценность» (6–11 лет). Действия в среде Лого дают детям возможность реализовывать их идеи и знания множеством различных способов, что способствует более прочному усвоению знаний и помогает преодолеть основные противоречия, возникающие на этих стадиях развития.

Одной из наиболее распространенных сейчас на Западе теорий начального образования является теория множественности интеллекта Говарда Гарднера. Согласно этой теории интеллект не является единым целым, а состоит из совокупности интеллектов (multiple intelligence). Известно 8 видов интеллекта: лингвистический, музыкальный, логико-математический, пространственный, натуралистический, телесно-

кинестический, межперсональный и внутриперсональный. Лого-программирование помогает развитию почти всех видов интеллекта, за исключением натуралистического (так как технология – по определению не природа) [5].

Общение учеников с учителями и со сверстниками во время занятий с Лого хорошо объясняется с точки зрения культурно-исторической теории Л. С. Выготского. По мнению Выготского, дети учатся в контексте социального взаимодействия, обучение ведет и определяет развитие, и процессу обучения надо помогать только в тех пределах, в которых учащиеся не могут решить задачу самостоятельно. А именно таким образом и осуществляется помощь на уроках.

Следовательно, несмотря на авторитетность теории Ж. Пиаже, можно сделать вывод, что концепция С. Пейперта получает не менее строгое (возможно даже более строгое) обоснование в рамках теорий Э. Эриксона, Г. Гарднера и Л.С. Выготского.

Активная работа в области Лого продолжается и в настоящее время. Существуют системы ЛогоМиры (эта программа является русской версией программы MicroWorlds, разработанной в LSCI, 1993 г.) и ПервоЛого (совместная разработка LCSI и ИНТ, 1996 г.). Среда Control Lab (русская версия этой программы называется ЛЕГО-Лаборатория), разработанная компанией LEGO Dacta, является естественным развитием идей, заложенных в системе LEGO TC Logo. Возникают и новые перспективные области исследований: среди таких проектов укажем, например, LEGO-роботы (Programmable Brick) – дальнейшее развитие систем LEGO TC Logo и Control Lab, позволяющее создавать автономных роботов, управляемых специальным миниатюрным компьютером.

Лого, конечно, отнюдь не сводится к созданию и распространению компьютерных программ. Существует мировое сообщество учителей и исследователей, заинтересованных в развитии и распространении педагогической философии Лого. Публикуется различная литература, посвященная как проблемам использования Лого в преподавании конкретных школьных дисциплин (таких, как математика, язык и т. д.), так и общим вопросам использования Лого в школьном и внешкольном образовании. Уже более 10 лет в США существует специальная некоммерческая организация Logo Foundation, координирующая деятельность мирового Лого-сообщества. В России поддержкой и развитием Лого-сообщества занимается Институт новых технологий образования [6].

Но Лого – это не просто язык общения или язык программирования, это еще и своего рода возрастной стандарт. В США обучение Лого 10-11-летних детей уже давно стало традицией, причем возрастные границы изучения Лого продолжают расширяться. Есть проекты изучения Лого с детьми младшего возраста [7]. Многие образовательные учреждения США оборудованы наборами Programmable Brick. И как показывает опыт, изучение Лого дает свои положительные результаты.

Примечания

1. *Пейперт С.* Переворот в сознании: Дети, компьютеры и плодотворные идеи / Пер. с англ. М.: Педагогика. 1989. 224 с.

2. *Микешина Л. А., Опенков М. Ю.* Новые образы познания и реальности. М.: РОССПЭН. 1997. С. 140-149.

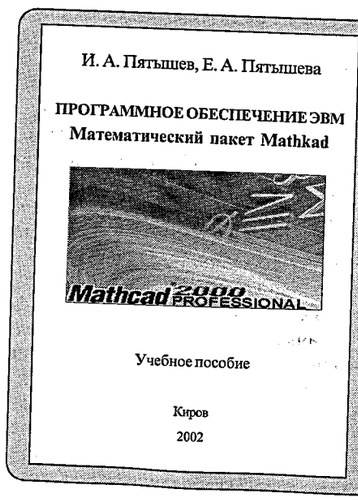
2. О Лого: *//http://www.int-edu.ru.*

3. *Яковлев В. А.* Теория познания Жана Пиаже и эволюционная эпистемология (научно-аналитический обзор) // Современные теории познания: Сб. обзоров и рефератов. М., 1992. С. 9-80.

4. *Gillespie, C., Beisser, S.* Developmentally Appropriate LOGO Computer Programming with Young Children. (2001). [Online]. Available: *http://www.aace.org/dl/files/ITCE/.*

5. Там же.

7. *Gillespie, C., Beisser, S.* Указ. соч.



Пятышев Илья Алексеевич
Пятышева Елена Анатольевна
Программное обеспечение ЭВМ. Математический пакет Mathcad: Учебное пособие. – Киров: Изд-во ВГГУ, 2002. – 64 с.

Пособие предназначено для изучения математического пакета на примере Mathcad 2000 в рамках таких курсов, как программное обеспечение ЭВМ (ПО ЭВМ) и информационные технологии в образовании (ИТО).

Основные темы, включенные в пособие: задачи вычислительного характера, построение графиков в различных системах координат, символьная математика (начало математического анализа), вектора и матрицы.

Пособие позволяет изучить материал, непосредственно выполняя упражнения за компьютером. Для самостоятельного выполнения предлагаются задания.

Книга будет полезна студентам первых курсов высших учебных заведений, а также учащимся и учителям информатики школ с углубленным изучением информатики.

ИНФОРМАТИКА В ШКОЛЕ

Е. В. Ведерникова

ОСОБЕННОСТИ ОБУЧЕНИЯ ИНФОРМАТИКЕ В КИРОВСКОМ ФИЗИКО-МАТЕМАТИЧЕСКОМ ЛИЦЕЕ

В течение более чем 10 лет лицей является одним из лидеров в России по работе с одаренными школьниками в области информатики. В статье рассматриваются особенности обучения, дающие такие результаты.

Кировский физико-математический лицей – муниципальное образовательное учреждение с углубленным изучением математики и физики. В этом году физико-математической школе, которая была предшественником лицея, исполняется 15 лет. Может показаться, что это не очень большой срок для учебного заведения, но тем ценнее результаты, достигнутые лицеем в образовательной деятельности. Немного фактов.

В 2001-2002 году лицей, как и в предыдущие годы, остался на позициях второго места среди школ России по результатам завоеванных дипломов на российских олимпиадах школьников. Это дипломы по предметам углубления – математике, физике, а также по информатике.

Нужно сказать, что информатика в лицее играет ведущую роль, подтверждением тому являются результаты олимпиад различного уровня.

Международные олимпиады

1993г	Лапунов А.	бронзовая медаль
1994 г	Лапунов А. Беров В.	золотая медаль серебряная медаль
1995 г	Беров В.	бронзовая медаль
1996 г	Матюхин В.	золотая медаль

Российские олимпиады по информатике:
с 1993 по 2003 годы: дипломы 1-й степени – 10,
дипломы 2-й степени – 13,
дипломы 3-й степени – 5.

Более 50 процентов выпускников лицея выбирают профессии и специальности для обучения в вузах, связанные с вычислительной техникой и программированием.

МГУ (Московский государственный университет): факультеты мехмат и ВМК – 64 выпускника лицея;

СПБГУ (Санкт-Петербургский государственный университет): факультет матмех – 8 выпускников;

СПБИТМО (Санкт-Петербургский институт точной механики и оптики): факультет информатики – 3 выпускника;

МГТУ им. Баумана (Московский государственный технический университет): факультет информационных систем – 25 выпускников;

ВГУ (Вятский государственный университет): факультет ФАВТ – 148 выпускников.

Из уже закончивших вуз наших выпускников многие продолжают обучение в аспирантуре, работают программистами в крупнейших компьютерных фирмах Microsoft, ABBYY, Nival Interactive , системными аналитиками, директорами и менеджерами в компаниях MTS, ГУ ЦБ РФ, ЦПС ФАПСИ, ИПИ РАН, ОАО «Интеллектуальные сети», ЦАИТ «Волга-Телеком».

Закономерны вопросы: повлияло ли обучение информатике в лицее на выбор будущей специальности и формирование профессиональных интересов наших выпускников и каковы же особенности преподавания курса информатики в лицее?

Первая особенность, которую стоит выделить и о которой стоит говорить – это построение курса информатики в лицее. Главная идея заключается в том, что школьники с высокими мыслительными способностями и интеллектуальными умениями требуют повышенной подготовки и в области информатики. А это значит, что учебный материал должен быть отобран, а методика преподавания разработана таким образом, чтобы занятия по информатике способствовали дальнейшему развитию интеллектуального и творческого потенциала школьника.

Основа программы по информатике – изучение программирования. Для чего нужно изучать программирование? Очевидно, что процесс проектирования программ - это процесс решения проблем человеком. Программирование подразумевает описание некоторой проблемы на определенном языке и последую-

ВЕДЕРНИКОВА Елена Витальевна – заместитель директора физико-математического лицея г. Кирова, учитель информатики

щее многократное моделирование с целью проверки модели. Эффективное описание проблемы для моделирования требует:

- овладение основными методами проектирования алгоритмов;
- развитого логического мышления;
- привития исследовательских навыков;
- формирования приемов умственной деятельности творческого типа;
- использование соответствующего языка программирования и среды.

Это уровень умений и навыков, который соответствует школьникам со способностями в области математики и физики.

На уровне младших классов (5-6) знакомство с программированием начинается со среды Логомиры. Разработанная на базе языка Лого, созданного С. Пейпертом в целях обеспечения комплексного развития образного, логического и алгоритмического компонентов мышления, среда Логомиры позволяет даже школьникам младших классов легко и естественно оперировать с информацией разного вида: графической, текстовой, звуковой. Мультимедийность этой среды вызывает повышенный интерес у школьников и способствует развитию познавательного интереса учащихся.

В средних и старших классах программирование осуществляется в среде Borland Pascal и Delphi. Изучается алгоритмический язык Паскаль, алгоритмы обработки данных, обзор технологий программирования.

Программа по информатике является авторской. Она разработана под руководством С. М. Окулова. Сотрудничество лицея и факультета информатики университета является плодотворным на протяжении многих лет, и достижения лицея в олимпиадах по программированию – результат данного сотрудничества.

Кроме обычных уроков по информатике в лицее работает система спецкурсов. Это в первую очередь спецкурсы «Олимпиадные задачи по информатике», которые работают на параллелях, начиная с 7-го класса, а также спецкурсы по издательским системам, компьютерному монтажу видеофильмов и т. д.

Таким образом, каждый школьник может реализовать свой интеллектуальный и творческий потенциал как на уроках информатики, так и на спецкурсах.

Вторая особенность, которую необходимо выделить, – это то, что наш лицей находится в Кировской области, в которой имеются устоявшиеся традиции (школа) развития учеников с использованием возможностей информатики как школьного предмета. Так, на последней российской олимпиаде по информатике именно Кировская область получила грамоту Министерства образования России за 100%-ный результат. Из четырех участников олимпиады, учеников физматлицея, все четыре школьника получили дипломы: один диплом 1-й степени, два диплома 2-й степени, один диплом 3-й степени.

Третья особенность – это построение системы олимпиад. Учебный год начинается Открытой командной олимпиадой лицея по программированию, которая в этом году проводилась уже в пятый раз и получила статус межрегиональной. Именно открытость олимпиады и высокий уровень решаемых задач привлекает к участию в ней сильные команды из многих регионов России.

Городская олимпиада по программированию дает возможность «новичкам» попробовать свои силы и отобрать достойных для выступления на областной олимпиаде.

Областная олимпиада проводится также в варианте открытой олимпиады. И это имеет большое значение. Высокий уровень подготовки участников, интересные задачи, четкая организация олимпиады – все это способствует привлечению многочисленных желающих попробовать свои силы на олимпиаде и создает определенную интеллектуальную конкурирующую среду.

В этом году впервые в лицее была проведена неделя информатики, которая предусматривала проведение в каждом классе и параллели внеклассных мероприятий по информатике (это игры, викторины, конкурсы), лицейских конкурсв WEB-сайтов, разработки и реализации игровых стратегий. Закончилась неделя лицейской командной олимпиадой по информатике. В ней мог принять участие любой ученик лицея, достаточно было только найти еще двоих единомышленников и сформировать команду. Олимпиада вызвала большой интерес и со стороны участников олимпиады, и со стороны жюри, которым являлись ученики старших классов, придумавшие задачи и систему тестов. Подобные олимпиады заслуживают должного внимания, и проводить их мы намерены гораздо чаще, например ежемесячно.

А в продолжение всех занятий в течение учебного года существует летняя компьютерная школа, которая до последнего времени организовывалась лицеем. Два отделения работают в этой школе: отделение олимпиадных задач и информационных технологий. Эта школа собирает многих способных ребят из разных регионов России, ведь занятия на олимпиадном отделении проводят студенты МГУ, победители российских и международных олимпиад по информатике среди школьников и студентов, многие из которых являются выпускниками лицея. Эта преемственность в обучении, возможность общаться и учиться у звезд олимпиадной информатики имеет большое значение.

Вот такая система обучения информатики существует в лицее, а точнее сказать, существовала до нынешнего года. Учебный план общеобразовательного учреждения – до предела «плотный» документ. С учетом общей тенденции к разгрузке школьника, изъятия из содержания обучения всего второстепенного,

информатика из плана лицея была убрана и заменена образовательной областью «Технология». На сегодняшний день информатика по учебному плану осталась только в 10-11-х классах, один час в неделю. Вот уж действительно, «что имеем не храним, потерявши плачем». Горько осознавать, что школа, имеющая систему и опыт работы, высокие результаты в области программирования, поставлена в условия выживания курса информатики. Сегодня лицей вырабатывает свою стратегию поддержания прежнего уровня обучения программированию и решению олимпиадных задач. Сохраняется вся линия спецкурсов, система олимпиад, разрабатывается программа курса «Технология» с использованием компьютерных средств. Мы надеемся, что наши усилия принесут свои плоды.

Е.В. Разова

ТЕОРИЯ ЧИСЕЛ НА УРОКЕ ИНФОРМАТИКИ

Статья посвящена вопросу реализации межпредметных связей на уроках информатики. На примере двух задач показано, как понятия и теоремы теории чисел могут быть использованы на уроке информатики для разработки и доказательства правильности алгоритмов и для их анализа. Рассмотрение таких задач способствует реализации принципов уровневой дифференциации, активизации ситуативно нестимулированной продуктивной деятельности, которая позволяет выходить за пределы заданного и видеть «непредвиденное».

В наше время, в условиях информатизации, умение применять вычислительную технику для решения разнообразных задач превращается в особо ценное качество человека. Очевидно, что это умение не приходит само по себе, даже при наличии самого совершенного компьютера. Этому человека надо целенаправленно обучать.

К сожалению, в настоящее время задачи и упражнения каждого предмета решаются только специфическими для него методами, при этом решения методами других дисциплин игнорируются, т. е. отсутствуют межпредметные связи.

При изучении программирования одна из проблем, с которой приходится сталкиваться, – это подбор задач для закрепления навыков работы с основными структурами данных и алгоритмическими конструкциями. Условия задач должны иметь оригинальную формулировку и практическую ценность, способную заинтересовать учащихся. В качестве таких задач могут быть выбраны задачи теории чисел, решение большинства из которых сводится к выполнению конечного набора определенных операций, то есть к выполнению некоторого алгоритма (например, вычисление НОД, НОК, генерация простых чисел и т. д.).

Использование задач теории чисел на занятиях по программированию позволяет не только получить качественный материал для изучения и отработки навыков работы с основными алгоритмическими конструкциями, использования рекурсивного программирования при решении задач, но и на новом уровне рассмотреть знакомые факты из теории чисел, углубить и расширить имеющиеся знания.

При таком подходе, решая многие задачи теории чисел с использованием вычислительной техники, с одной стороны, появляется возможность получить на первых порах неэффективное и нестрогое с точки зрения математики решение, возможность предугадать ответ. С другой стороны, привлечение математического аппарата для построения более эффективного алгоритма, дает возможность познакомить учащихся с проблемами анализа алгоритмов.

Например, всегда вызывала интерес работа с простыми числами. Простые числа будоражили воображение математиков с древнейших времен. Еще Евклид доказал, что простых чисел бесконечно много, но их распределение носит весьма неравномерный характер. Есть простые числа-близнецы, разность между которыми равна 2, но существуют сколь угодно большие промежутки, в которых простых чисел нет [1].

Учащихся привлекает постановка задач, тесно связанных с именами древнейших и современных известных математиков, занимавшихся поиском интересных чисел, естественно «вручную», и возможность решать считавшиеся сложными для великих умов человечества задачи просто, быстро и «элегантно» с помощью компьютерных программ.

Рассмотрим следующую задачу.

Задача 1. Найти пять подряд идущих составных чисел. [2]

Может быть предложен следующий алгоритм решения данной задачи.

```
Uses Crt;
Const n=5;
Var x, k, i : Longint;
Function Simple(x : Longint) : Boolean;
{Функция проверки, является ли число простым. Число простое, если в у него нет делителей в промежутке [2, √x ].}
Var d : Longint;
Begin
  d:=2;
  While (d<=Sqrt(x)) And (x Mod d <> 0) Do Inc(d);
  Simple:=(d>Sqrt(x))
End;
Begin
  ClrScr;
```

РАЗОВА Елена Владимировна – аспирант кафедры информатики и МОИ ВятГГУ

```
  x:=3;
  k:=0;
  Repeat {Повторять обработку очередного числа
          до тех пор, пока не будет найден отрезок
          из пяти (n=5) подряд идущих составных
          чисел или не будет исчерпан диапазон
          Longint.}
    Inc(x);
    {Если число составное, то увеличение k – числа
    подряд идущих составных чисел, иначе – «сброс»
    значения k (начинается поиск новой последова-
    тельности).}
    If Not(Simple(x)) Then Inc(k) Else k:=0
  Until (x=MaxLongint) Or (k=n);
  {Вывод найденной последовательности.}
  If k=n Then For i:=k-1 DownTo 0 Do Write(x-i,' ')
    Else Writeln(n,'подряд идущих простых чисел в
    диапазоне Longint нет.');
  Readln
End.
```

Решение этой задачи с алгоритмической точки зрения не вызывает труда, но дальше у ученика возникает естественное желание поэкспериментировать, например найти как можно более длинный отрезок из идущих подряд составных чисел (например, найти отрезок из ста таких чисел). А здесь уже возникают проблемы, связанные со временем выполнения алгоритма (временной сложностью алгоритма). Таким образом, возникает необходимость оптимизации алгоритма, а здесь может быть полезна математика.

Идея доказательства Евклида о бесконечности простых чисел [3] дает подсказку для получения реального способа генерации искомых чисел – это числа (n+1)!+2, (n+1)!+3, …, (n+1)!+(n+1). Действительно, эти числа являются составными, так как число (n+1)!=1·2·3·…·n·(n+1) делится на 2, 3, …n и (n+1), откуда первое число (n+1)!+2 делится на 2, второе число (n+1)!+3 делится 3, …, n-е число (n+1)!+(n+1) делится на n+1.

Но здесь возникает вторая проблема – представление «длинных» чисел. Например, вычисляя факториал ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$), в диапазоне представления величин типа Longint удастся правильно получить только 12! = 479001600. Для больших значений, конечно, можно использовать действительные типы данных, но это уже не гарантирует точного результата. Поэтому полезно для получения точных значений при действиях с многозначными числами разработать другие способы представления таких чисел, алгоритмы выполнения арифметических и других операций, процедуры ввода и вывода результатов и т. д.

Наиболее естественным способом представления многозначного числа является запись каждого его разряда в виде отдельного элемента линейного массива, начиная с разряда единиц, т.е. цифра из разряда единиц – элемент массива с номером один, цифра из разряда десятков – элемент массива с номером два и т. д.

Определим для работы с «длинными» неотрицательными числами тип данных:

```
  Const MaxDig=10000;
        Osn=10000;
  Type Tlong=Array[0..MaxDig] Of Integer;
```

Для решения поставленной задачи необходимо уметь выполнять следующие действия:

1) ввод «длинного» числа (процедура ReadLong);

2) умножение «длинного» числа на «короткое» (процедура MulLongShort);

3) прибавление «короткого» числа к «длинному» (процедура SumLongShort);

4) вывод «длинного» числа (процедура WriteLong).

Описание и детальный разбор этих процедур можно найти в учебном пособии [4]. Основная логика решения данной задачи с использованием указанных процедур очевидна.

```
Const MaxDig=10000;
      Osn=10000;
Type Tlong=Array[0..MaxDig] Of Integer;
Var A,c : Tlong;
    i,n : Longint;
{Основная программа.}
Begin
  Readln(n); {Ввод количества искомых подряд
              идущих составных чисел.}
  FillChar(A,SizeOf(A),0);
  A[0]:=1; A[1]:=1; {Начальное значение «длинного»
                     числа – 1.}
  For i:=2 To n+1 Do MulLongShort(A,i);
                     {Нахождение (n+1)!}
  For i:=2 To n+1 Do Begin
    SumLongShort(A,i,c); {Вычисление значения
                          числа c=(n+1)!+i.}
    WriteLong(c); {Вывод полученного числа с
                   на экран.}
  End;
  ReadLn
End.
```

Несмотря на значительное усложнение программы из-за использования алгоритмов «длинной» арифметики, ответ в этом случае получается мгновенно.

Таким образом, рассмотренная задача позволяет выявить две основные проблемы программирования: временная сложность алгоритма и ограниченность диапазонов значений числовых типов. Решение этих проблем дает два пути усложнения данной задачи, способствует совершенствованию техники программирования, формированию навыков творческой исследовательской деятельности.

Доказательство теоремы о бесконечном количестве простых чисел дает один из способов решения следующей задачи.

Задача 2. Найти набор из n различных натуральных чисел, в котором любые два числа взаимно просты, а любые несколько чисел дают в сумме составное число. [5]

Ответом данной задачи будут n чисел вида $a_i=i*n!+1$, i=1, 2, … n. Следовательно, алгоритм решения данной задачи будет незначительно отличаться от второго способа решения первой задачи и может быть реализован практически каждым. Но если не прибегать к использованию данного математического факта, решение задачи становится достаточно трудоемким с алгоритмической точки зрения, а именно, возникает необходимость решения следующих алгоритмических проблем:

1) проверка взаимной простоты двух натуральных чисел (алгоритм Евклида нахождения наибольшего общего делителя двух чисел);

2) генерация всех подмножеств выбранного набора чисел;

3) проверка, является ли сумма элементов подмножества составным число (проверка простоты числа).

Как видно, данный способ решения может вызвать затруднения даже у достаточно сильных учеников.

Таким образом, методически продуманный отбор заданий для практики по программированию позволяет наряду с изучением информатики активизировать и углубить знания учащихся по теории чисел. При этом понятия и теоремы теории чисел используются для разработки и доказательства правильности алгоритмов и для их анализа, т. е. приобретают практический, прикладной характер. Кроме того, данные задачи способствуют реализации принципов уровневой дифференциации, когда, решая вроде бы одну и ту же задачу, каждый ученик имеет возможность максимально реализовать свой потенциал. При решении таких задач происходит развитие учащихся, активизация ситуативно нестимулированной продуктивной деятельности, которая позволяет выходить за пределы заданного и увидеть «непредвиденное». В этой ситуативно нестимулированной деятельности и кроется тайна высших форм человеческого творчества.

Примечания

1. *Виноградов И. М.* Основы теории чисел. М.: Наука, 1982; *Сергеев И. Н., Олехник С. Н., Гашков С. Б.* Примени математику. М.: Наука; Гл. ред. физ.-мат. лит., 1990.

2. *Сергеев И. Н., Олехник С. Н., Гашков С. Б.* Примени математику. М.: Наука; Гл. ред. физ.-мат. лит., 1990.

3. *Виноградов И. М.* Основы теории чисел. М.: Наука, 1982

4. *Окулов С. М., Пестов А. А.* 100 задач по информатике. Киров: Изд-во ВГПУ, 2000.

5. *Васильев Н. Б., Егоров А. А.* Задачи Всесоюзных математических олимпиад. М.: Наука; Гл. ред. физ.-мат. лит., 1988.

7 Зак. 2868

С. В. Шедов

МЫТИЩИНСКАЯ ШКОЛА ПРОГРАММИСТОВ

Учить программированию можно и должно. В статье на примере работы коллектива талантливых педагогов из г. Мытищи показывается, как это делать.

Мытищинская школа программистов (МШП) – некоммерческая образовательная организация, созданная учителями-энтузиастами лицея № 23 г. Мытищи Московской области С. В. Шедовым, И. Н. Рутковской и А. Ю. Бауровым в 2001 году. В этом году школа будет проводить уже четвертый набор.

Основными задачами, которые ставит перед собой школа, являются поиск и привлечение способных к программированию учащихся Мытищинского района, подготовка к олимпиадам по информатике районного, областного и российского уровня, раскрытие творческих способностей учащихся, пропаганда информационных технологий среди молодежи, профессиональная ориентация учащихся в области программирования.

ПРИЕМ В ШКОЛУ ПРОГРАММИСТОВ

Прием в Школу программистов проходит на основе конкурсного отбора. В сентябре каждого учебного года для всех желающих организован вступительный экзамен. Варианты задач различаются для 6-7-х классов, 8-го класса и 9-10-х классов. Вступительный экзамен состоит из заданий по математике, логике и алгоритмике.

Для решения задач не требуется углубленных знаний школьной программы, основной упор делается на умение ребенком находить свои, порой нестандартные способы решения задач. Часть задач направлена на проверку математической культуры школьников и на выявление его уровня развития. Продолжительность экзамена – от 3 до 4 часов.

Ниже приводятся примеры задач, которые предлагались на вступительных экзаменах прошлых лет и поясняются цели, которые преследовала та или иная задача.

1. Следующая задача простая (решается в уме), однако по своей постановке является нестандартной в сравнении с аналогичными задачами, которые решают школьники на уроках.

• *(6-7-й класс, 2001 год) Экспресс из Баслтауна в Айрончестер идет со скоростью 60 км/ч, а экспресс из Айрончестера в Баслтаун, который выходит одновременно с ним, - со скоростью 40 км/ч. К сожалению, разные справочники дают разные расстояния между городами. Однако доподлинно известно, что расстояние между ними больше 240 км и меньше*

ШЕДОВ Сергей Валерьевич – учитель информатики (г. Мытищи, Московская область)

320 км. На каком расстоянии друг от друга будут находиться поезда за час до встречи?

Несмотря на «утешительный» характер задачи, некоторых школьников она ставит в тупик. Сбивает нечеткая формализация задачи и некоторая расплывчатость исходных данных. Многие преодолевают этот барьер, для них ответ задачи становится очевиден: 60+40=100 км.

2. Выявить мыслящих школьников хорошо помогают подобные задачи:

• *(8-й класс, 2002 г.) После заготовки дров работник подсчитал, что он выполнил 53 распила и в результате получил 72 полена. Сколько бревен было вначале? (Ответ: 19.)*

• *(7-й класс, 2001 г.) Кусок мыла, лежащий на умывальнике, имеет форму прямоугольного параллелепипеда. Мыло расходуют равномерно каждый день. Спустя неделю размеры мыла уменьшились вдвое. На сколько еще дней хватит этого мыла, если им будут пользоваться так же интенсивно? (Ответ: на один день.)*

3. Следующая задача хороша тем, что доказать невозможность того, что числа станут равными, под силу даже 7-класснику.

• *(6-7-й класс, 2001 г.) На доске написаны числа 0, 1, 0, 0. За один шаг разрешается прибавлять единицу к любым двум из них. Можно ли, повторяя эту операцию, добиться, чтобы все числа стали равными? Если можно, то выпишите последовательность прибавлений.*

4. В основе многих задач лежат математические игры-головоломки, для успешного решения которых учащийся, как правило, должен найти выигрышную стратегию для того или иного игрока.

• *(8-й класс, 2002 г.) Вася и Петя играют на доске 10 x 10 в такую игру. У Васи есть много квадратиков в одну клетку, у Пети есть много уголков из трех клеток (см. рис.) Они ходят по очереди – сначала Вася кладет на доску свой квадратик, затем Петя – свой уголок, потом Вася кладет еще квадратик и т. д. (Класть фигуры поверх других нельзя.) Проигрывает тот, кто не может сделать очередной ход. Вася утверждает, что он всегда сможет выиграть, как бы ни старался Петя. Прав ли Вася?*

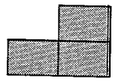


Рис. 1

С такой задачей обычно справляется не более 20% школьников. Главная идея задачи – разбить поле на квадраты размера 2x2, и Пете остается дополнять своим уголком до этого квадрата любой квадратик Васи, пока не заполнится вся доска. В этом и заключается стратегия игры Пети.

5. Более сложная задача, в которой надо получить и обосновать выигрышную стратегию.

• *(7-8-й класс, 2001 г.) Есть 9 запечатанных коробок, в которых лежит по 1, 2, 3, … ,9 фишек соответственно (на каждой коробке написано, сколько в ней фишек). Двое играющих по очереди берут по одной фишке из любой коробки, распечатывая, если необходимо, коробку. Проигрывает тот, кто последним распечатает коробку. Кто из игроков может всегда выигрывать независимо от игры противника? Докажите.*

Варианты вступительных экзаменов содержат также логические задачи, нестандартные геометрические задачи, простейшие комбинаторные и алгоритмические задачи.

Опыт показал, что подобный набор задач позволяет достаточно эффективно разделить детей на группы. Учащиеся, успешно выдержавшие вступительный экзамен и поступившие в МШП, распределяются по группам преимущественно с учетом возраста и степени подготовленности. Если начальные знания позволяют это сделать, то ученик может быть сразу зачислен на 2-й год обучения. Каждый год из показавших наилучшие результаты формируется группа наиболее способных учеников. Школьники из этой группы обучаются полностью бесплатно, за счет средств местного бюджета. Основное направление, которое интересует ребят из этой группы, – олимпиадная информатика, поиск решений и подходов к нестандартным и сложным задачам, изучение нетривиальных алгоритмов.

МЕТОДИКА РАБОТЫ ШКОЛЫ ПРОГРАММИСТОВ

Методика работы Школы программистов необычна для средней школы и заимствует некоторые подходы университетского образования. Ученики МШП посещают несколько обязательных и факультативных курсов. Каждый школьник имеет возможность сам формировать индивидуальную программу обучения – выбирая и посещая те курсы, которые соотносятся с его интересами и возможностями.

Обязательные курсы составлены методическим советом Школы программистов таким образом, чтобы они образовывали непрерывный цикл трехгодичного обучения программированию. Обязательные курсы закладывают основы правильного алгоритмического мышления будущего программиста, вырабатывают парадигму программирования и являются базисом для обучения любым языкам и системам программирования. Школьники глубоко изучают алгоритмический язык Pascal, а также основы построения и анализа нетривиальных алгоритмов.

Кроме того, в число обязательных курсов входят некоторые разделы математики, выходящие за рамки школьной программы, но тем не менее необходимые для успешного освоения программирования. Это в первую очередь дискретная математика и ее разделы: теория множеств, математическая логика, теория чисел, комбинаторика, теория графов, а также элементы линейной алгебры, аналитической геометрии, теории игр и криптографии. Обязательные курсы составляют учебную нагрузку, равную, как правило, четырем академическим часам в неделю.

Программа первого семестра (октябрь – январь) общая для всех групп и включает только следующие обязательные курсы:

1-й год обучения (табл. 1, 2).

В конце января, по окончании первого семестра, для всех школьников проходит Закрытая Олимпиада Школы Программистов – основная форма контроля достигнутых результатов. Приведем несколько, на наш взгляд, наиболее удачных задач, предлагавшихся на прошлых олимпиадах для первого года обучения.

1. Практика показала, что следующая задача является одной из лучших на тему ветвление. По нашему мнению, с дидактической точки зрения, трудно найти задачу лучше этой, которая бы ввела школьника в проблематику «олимпиадного программирования». Для решения не требуется никаких специальных знаний и сложного программирования, однако при всей кажущейся простоте задачи, на ней могут «сесть в лужу» даже самые сильные школьники. Мало у кого проходит самый сложный тест к задаче – 134 (ответ должен быть 1 0 0)

• *(Московская городская олимпиада по программированию, 1988 г.)* [3]. Пара носков стоит 1.05 руб., связка (12 пар) стоит 10.25 руб., а коробка (12 связок) стоит 114 руб. Составьте программу, которая по числу ***n*** пар носков, которые хочет купить покупатель, вычисляет числа ***n1***, ***n2***, ***n3*** коробок, связок и пар носков, которые ему следует (выгоднее всего) купить. Пример входных данных: 11. Пример выходных данных: 0 1 0 (вместо 11 пар носков следует покупать связку – это обойдется дешевле).

2. В задаче «Волшебная последовательность» есть над чем поломать голову, однако решается она гениально просто – достаточно одного цикла FOR. Лучшей олимпиадной задачи для начинающих на тему «циклы» трудно придумать. Главное здесь – догадаться до идеи, как строить последовательность. (Ответ: N-2 единиц, 2 и N.)

Таблица 1

Программа обучения в Мытищинской Школе программистов, 1-й год обучения, 1-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Алгоритмическое программирование на языке Pascal*	2
2	*Введение в алгоритмы*	1
3	*Дискретная математика*	1

Таблица 2

Программа обучения в Мытищинской Школе программистов, 1-й год обучения, 2-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Алгоритмическое программирование на языке Pascal*	2
2	*Комбинаторика для программистов*	1
3	• *Построение нетривиальных алгоритмов и программ (группы 1-го уровня)* • *Алгоритмы и алгоритмические языки (группы 2-го уровня)* • *Развивающие задачи (группы 3-го уровня)*	1
4	*Спецкурсы по выбору*	до 2 часов

Таблица 3

Программа обучения в Мытищинской Школе программистов, 2-й год обучения, 3-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Структурное программирование на языке Pascal*	2
2	*Начала линейной алгебры и аналитической геометрии (группы 1-го уровня)*	1
3	*Построение нетривиальных алгоритмов и программ (группы 1-го уровня)*	1
4	*Спецкурсы по выбору*	до 4 часов

Таблица 4

Программа обучения в Мытищинской Школе программистов, 2-й год обучения, 4-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Структурное программирование на языке Pascal*	2
2	*Спец. главы дискретной математики (группы 1-го уровня)*	1
3	*Построение нетривиальных алгоритмов и программ (группы 1-го уровня)*	1
4	*Спецкурсы по выбору*	до 4 часов

• *(I Всероссийская Командная Олимпиада по программированию)* Недавно Петя научился считать. Он тут же заметил, что число 2 обладает замечательным свойством 2+2=2x2. Его старший брат Ваня тут же объяснил ему, что дело не в двойке. «Дело в том, что последовательность 2, 2 – волшебная, – сказал Пете Ваня. – Волшебная последовательность – это такая последовательность натуральных чисел, что сумма ее членов равна их произведению. Например, последовательность 1, 2, 3 – тоже волшебная». Петя тут же сложил 1, 2 и 3, потом перемножил их и обрадовался. Теперь Петя хочет найти более длинные волшебные последовательности. Помогите ему!

Таблица 5

Программа обучения в Мытищинской Школе программистов, 3-й год обучения, 5-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Эвристическое программирование на языке Pascal*	2
2	*Спецкурсы по выбору*	до 6 часов

Таблица 6

Программа обучения в Мытищинской Школе программистов, 3-й год обучения, 6-й семестр

№	Курс	Учебная нагрузка (в неделю)
1	*Объектно-ориентированное программирование на языке Object Pascal*	2
2	*Спецкурсы по выбору*	до 6 часов

3. В олимпиаде для первого семестра традиционно предлагается много задач целочисленной арифметики, в решениях которых запрещено использование логических выражений и условных операторов. Главное здесь – при ограниченных ресурсах (фактически div и mod) выписать формулы для решения задачи, которые должны, ко всему прочему, работать при любых входных данных, включая граничные случаи. Задача «Часы» – одна из самых сложных задач, она предлагалась на I Закрытой Олимпиаде Школы программистов. Об этой задаче в Школе программистов ходят легенды – вывод формулы, которая решает последние два пункта задачи и работает при любых входных данных, на разборе задач занял около часа!

• С начала суток прошло *n* секунд. Определить:

– сколько полных часов прошло с начала суток;

– сколько полных минут прошло с начала очередного часа;

– сколько полных секунд прошло с начала очередной минуты;

– сколько полных минут пройдет до того момента, когда часовая и минутная стрелки на циферблате впервые совпадут;

– сколько полных минут пройдет до того момента, когда часовая и минутная стрелки на циферблате впервые расположатся точно перпендикулярно друг другу.

• *Примечание: часовая, минутная и секундная стрелки перемещаются по циферблату с равными дискретными шагами в 1/60 циферблата (циферблат разбит на 60 частей). При вычислении положения минутной и часовой стрелки округления производятся в меньшую сторону, то есть стрелки не занимают промежуточных положений и не перемещаются на следующую позицию раньше времени. Например, если с начала суток прошло 119 секунд, то минутная стрелка указывает на 1 минуту, часовая – на 0 часов.*

По результатам олимпиады учащиеся разделяются на группы трех уровней, обучение в которых далее осуществляется по различным программам.

Начиная со второго года (табл. 3, 4) обучения доля основных курсов начинает уменьшаться, а доля спецкурсов постепенно увеличивается.

На третьем году (табл. 5, 6) обучения у школьников остается только один обязательный курс – основной курс «Программирование на языке Pascal», всю остальную программу обучения школьники формируют сами.

МЕТОДИКА ПРЕПОДАВАНИЯ ОСНОВНОГО КУРСА – «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PASCAL»

Алгоритмический язык Pascal преподается в Школе программистов по авторской методике к.т.н., декана факультета информатики ВятГГУ С. М. Окулова. Основная идея автора заключается в том, что «занятия по информатике должны в корне отличаться от традиционных занятий по любому другому предмету: здесь должна поощряться ошибка, ибо только через ошибку можно прийти к результату; стиль мышления программиста свой, отличающийся от стиля мышления как математика, так и любого другого специалиста, – он настроен на борьбу с хаосом. Основной методический принцип обучения – все познается через труд, через процесс решения задач, через преодоление собственных ошибок. Этот принцип определяет структуру занятий по Pascal: вводная часть > обсуждение нового материала > эксперименты с заготовками решения задач > самостоятельное решение задач». [6]

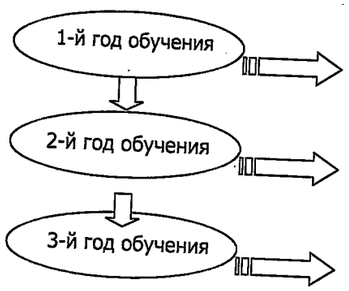


Рис. 2

Основным учебником для первого, второго и третьего года обучения является книга С. М. Окулова «Основы программирования» [3], для второго и третьего года используется также книга того же автора «Программирование в алгоритмах». Эти книги входят в состав обязательной литературы, которую должен иметь каждый ученик Школы программистов.

Главная цель данного курса – развитие мышления ученика. На первом году обучения речь идет об алгоритмическом мышлении, на втором – о структурном, на третьем – об эвристическом. Причем элементы обучения синтезированы в одно целое, часто многие аспекты рассматриваются одновременно. Алгоритмическим мышлением мы называем умение разработать алгоритм решения задачи. Структурный стиль мышления подразумевает умение «расчленить» задачу; программы становятся простыми и ясными, в них используются только основополагающие конструкции, каждый блок в идеале имеет только одну точку входа и выхода. Умение находить истину, доказывать факт правильности решения задачи назовем эвристическим стилем мышления. Схематично можно выделить три этапа данного трехгодичного курса (см. рис. 2).

Может возникнуть вопрос, почему именно при занятиях программированием происходит развитие этих видов мышления, почему традиционные предметы, на которых тоже решаются задачи (физика, математика) не всегда способны справиться с этой задачей? По нашему мнению, это происходит из-за наличия жесткой обратной связи с компьютером, связи объективной и лишенной эмоций, которая представляет собой мощный инструментарий индивидуального и развивающего обучения.

Теперь покажем, какими методами достигаются поставленные цели.

На первом году обучения изучаются основные алгоритмы и реализующие их операторы языка программирования Turbo Pascal (ввод и вывод данных; ветвление в неполной и полной формах, а также вложенное; неопределенные и определенный циклы), но не только. Задачи с первых же занятий предлагаются нетривиальные, с серьезным математическим наполнением. В принципе отсутствуют «жевательные» задачи типа «Hello, world». Учащиеся МШП учатся составлять программы, используя минимальное число конструкций. Школьники вырабатывают умение видеть динамику работы программы. С этой целью в начале применяется ручная трассировка работы программы, а затем – средства отладчика среды программирования Borland Pascal 7.0.

I семестр посвящен линейным алгоритмам и различным формам ветвления, определенным и неопределенным циклам; учащиеся знакомятся со всеми подтипами целого типа данных и с логическим типом.

Задачи, решаемые в этот период, в основном посвящены целочисленной арифметике:

• выделение и обработка цифр числа,

• побитовые операции с целыми числами, операции сдвига,

• поиск наибольшего и наименьшего из нескольких целых чисел,

• подсчет сумм и количеств,

• поиск чисел по условию (пифагоровы числа: $a^2+b^2=c^2$, числа Армстронга: $\overline{xyz} = x^n+y^n+z^n$, числа Мерсена: 2^p-1, где p – простое, автоморфные числа – равные последним цифрам своего квадрата, совершенные числа и др.),

• делители, кратные, простые числа.

Другой тип задач – на формирование сложных условий, знание законов алгебры, логики, упрощение логических выражений.

II семестр в основном посвящен вложенным циклам и работе с одномерными массивами. Задачи целочисленной арифметики, решаемые с использованием циклов, усложняются. Многие задачи на обработку чисел предполагают несколько вариантов решения – простые, с быстродействием $O(N)$ и более сложные, которые работают уже за $O(\log N)$ операций. Завершают год задачи на обработку элементов массива, вывод и выборку элементов по условию, суммирование элементов, поиск и анализ max и min.

Таким образом, первый год обучения посвящен развитию алгоритмического мышления ученика.

III семестр (2-й год обучения) посвящен механизму использования процедур и функций, создания блоков логики с одной точкой входа и одной точкой выхода; а также такой фундаментальной структуре данных, как массив. Учащиеся еще раз возвращаются к работе с элементами массива, но уже на более качественном уровне. После освоения процедур и функций они в состоянии писать хорошо структурирован-

ные программы, оформляя в виде отдельных процедур основные этапы решения задачи.

В этот же период учащиеся знакомятся с вещественным, символьным и строковым, а также множественным типами данных. В ходе изучения вещественного типа школьники впервые сталкиваются с задачами численной математики – методами приближенного решения уравнений, нахождения приближенного значения интеграла как площади криволинейной трапеции, вычислением значений элементарных функций путем разложения в ряд. Тогда же изучается и рекурсия как метод программирования, рассматриваются такие изумительные объекты, как фракталы.

Преподавание программирования по данной методике ведется на высоком уровне сложности, но с учетом возможностей группы: варьируется темп освоения нового материала, количество и сложность решаемых задач. Используются и элементы концентрического обучения, когда одна и та же или похожая задача вводится в рассмотрение снова – но уже на другом, более высоком уровне обобщения, с применением более элегантных методов программирования.

Хорошим примером может служить задача вычисления степени числа (как известно, Pascal не имеет операции возведения в произвольную степень). Впервые ученики Школы программистов сталкиваются с ней уже на третьем занятии, при изучении линейного алгоритма. Рассматривается частный случай – вычисление целой степени числа от нулевой до десятой. Затем предлагается модифицировать программу так, чтобы она вычисляла степень за минимальное число операций умножения. После некоторых размышлений учащиеся приходят к выводу, что для «экономного» возведения в четную и нечетную степень нужно использовать разные формулы. Затем та же задача (или ее модификации) решается уже с использованием цикла, затем рекурсии, а затем и динамического программирования.

Вообще подбор задач – одна основных из особенностей данной методики обучения, они не простые, но под силу как ученику старших классов, так и логично мыслящему 6-7-класснику.

Таким образом, на втором году обучения у школьника развивается структурный стиль мышления.

В **IV семестре** учащиеся очень подробно знакомятся с классической задачей программирования - сортировкой массива. На этом примере показывается все многообразие способов, которыми можно решать одну и ту же задачу. Исследуется эффективность и производительность каждого алгоритма как по быстродействию, так и по затратам памяти. Вначале учащиеся знакомятся с простейшими методами сортировки массивов: сортировка простым выбором, сортировка простым обменом («пузырек»), сортировка простыми вставками, сортировка подсчетом, шейкер-сортировка, сортировка с убывающим шагом (сортировка Шелла), сортировка Фрэнда. Причем четыре последних метода школьники осваивают самостоятельно. Затем упор делается на методы быстрой сортировки массивов: сортировка слияниями, быстрая сортировка (сортировка Хоара), пирамидальная сортировка, k-путевая сортировка.

Тема «Сортировка массивов» логично продолжается разделом «Поиск данных». Учащиеся знакомятся с алгоритмами линейного, бинарного и случайного поиска, а также с алгоритмом Бойера-Мура поиска подстроки в строке.

Семестр завершается темой «Арифметика многоразрядных («длинных») чисел». По окончании данной темы учащиеся впервые сталкиваются с новой формой контроля знаний, с которой им придется постоянно сталкиваться на третьем году обучения – контрольный тренинг. Цели контрольного тренинга – не только (и не столько!) в проверке тех или иных знаний учащихся, а в закреплении пройденного материала, выработке техники программирования, тренировке психологической устойчивости к стрессовым состояниям, способности программировать в режиме цейтнота, с которым, в связи со спецификой IT-отрасли, безусловно, придется столкнуться школьникам в своей будущей работе программиста или системного администратора. За 2-2,5 часа ученику необходимо написать «с нуля» от 10 до 14 работающих подпрограмм, реализующих следующие функции с многоразрядными числами: ввод числа, вывод числа, сравнение чисел, сложение чисел, разность чисел, умножение «длинного» на «короткое», умножение «длинного» на «длинное», деление «длинного» на «короткое», деление «длинного» на «длинное». В зависимости от подготовки группы литературой пользоваться или полностью запрещено, или же разрешено сделать несколько подходов к книге вне компьютера, при этом любые записи делать не разрешается.

В **V семестре** (3-й год обучения) изучается материал, обычно не затрагиваемый при работе с учащимися средних школ. Семестр полностью посвящен динамическим структурам данных. Учащиеся подробно рассматривают ссылочный тип данных, пишут модули для работы со списком, стеком и очередью. Большая часть времени посвящается работе с деревьями, школьники изучают двоичные деревья, АВЛ-деревья и Б-деревья. По мнению автора методики, без материала этого семестра «невозможен переход на следующий «виток» обучения программированию. В объектно-ориентированных технологиях он является одним из стержневых, без которого понимание технологий остается «любительским».

Необходимо отметить, что задачи на динамический тип данных чрезвычайно трудно отлаживать. Для этих задач неэффективна встроенная система отладки среды программирования Borland Pascal 7.0, а о ручной трассировке программ и говорить не приходится. Поэтому предвидеть и рассмотреть все случаи работы программы становится невероятно сложно. Школьники приобретают новые навыки доказательного программирования и верификации программ.

Таким образом, формируется эвристический стиль мышления, он развивается на протяжении всего третьего года обучения.

Наконец, в **VI,** заключительном, **семестре** школьники глубоко знакомятся с объектно-ориентированным программированием на языке Object Pascal. В основу курса положены методические рекомендации [5]. По окончании семестра школьники получают последнюю порцию знаний того стержня, который объединен в МШП в единый трехгодичный курс под общим названием «Программирование на языке Pascal». После прохождения этого курса изучение любых технологий программирования, любых современных языков и систем программирования становится обыденным делом, мы уверены, что у успешно справившихся с программой школьников не возникнет никаких проблем с программированием ни в вузе, ни в будущей профессиональной деятельности.

СПЕЦКУРСЫ

Факультативные курсы охватывают разнообразные сферы информационных технологий, программирования и современной математики. Учащиеся могут выбирать спецкурсы уже начиная со II семестра. Разумеется, спецкурсы ориентированы на тот или иной уровень подготовки детей – для разных годов обучения спецкурсы предлагаются разные. Некоторые спецкурсы не требуют предварительной подготовки (например, архитектура ЭВМ), остальные факультативные курсы рассчитаны на определенный год обучения.

Существуют три основных направления факультативных курсов.

I. Математическое направление:

1) Теория множеств

2) Математическая логика

3) Теория графов

4) Системы счисления и компьютерная арифметика

5) Вычислительная геометрия

II. Языки и технологии программирования:

1) Визуальное программирование на языке *Visual Basic 6.0* для начинающих *(1-й и 2-й год обучения)*

2) Специальное программирование на языке *Visual Basic 6.0 (2-й год обучения)*

3) Объектно-ориентированное программирование в среде *Delphi 7.0 (2-й и 3-й год обучения)*

4) Программирование на языке C *(3-й год обучения)*

5) Функциональное программирование на языке Lisp *(3-й год обучения)*

6) Особенности программирования под различные операционные системы *(DOS, Windows, Unix) (3-й год обучения)*

7) Архитектура ЭВМ и низкоуровневое программирование *(Assembler) (3-й год обучения)*

8) Базы данных *(SQL) (3-й год обучения)*

III. Internet-программирование и сетевая безопасность

1) Internet-конструирование для начинающих *(1-й год обучения)*

2) Internet-программирование (JScript/VBScript, Java, PHP) *(2-й и 3-й год обучения)*

3) Компьютерные сети *(протоколы, безопасность, шифрование, защита от взлома) (2-й и 3-й год обучения)*

ЗАКЛЮЧЕНИЕ

В Школе программистов работают высококвалифицированные преподаватели с университетским образованием по математике, физике и программированию. Кроме того, некоторые спецкурсы проводятся приглашенными специалистами, работающими в сфере информационных технологий. В настоящий момент в МШП работают 7 преподавателей.

Школа программистов проводит Открытые Командные Олимпиады по программированию. В олимпиадах, помимо школьников города, принимают участие команды из других городов Московской области и г. Москвы. Кроме того, команда школы программистов регулярно участвует во всех командных соревнованиях по программированию, включая всероссийскую олимпиаду. Ученики Школы программистов ежегодно участвуют в районных, областных и всероссийских олимпиадах по информатике. На протяжении двух последних лет они занимают все призовые места в районной олимпиаде по информатике, ее ученики завоевали 4 диплома на Московской областной олимпиаде и 3 диплома на Открытой Кировской областной олимпиаде по информатике. Школьники регулярно представляют свои творческие проекты на различных конкурсах программистов. Некоторые работы даже вошли в состав учебников по информатике для средней школы.

В дни школьных каникул лучшие ученики Школы программистов направляются в специализированные компьютерные лагеря – Московский областной компьютерный лагерь (г. Троицк) и Летнюю компьютерную школу (г. Киров).

Дополнительную информацию о работе МШП вы можете найти на сайте www.informatics.ru.

Примечания

1. *Брудно А. Л., Каплан Л. И.* Московские олимпиады по программированию. М.: Наука, 1990.

2. *Окулов С. М.* Основы программирования. М.: Лаборатория базовых знаний, 2002.

3. Там же.

4. *Окулов С. М.* Программирование в алгоритмах. М.: БИНОМ. Лаборатория знаний, 2002.

5. *Окулов С. М., Торгашова Н. Э., Кедров А. В.* Объектно-ориентированное программирование. Киров: Изд-во КПГУ, 1994.

М. В. Овсянникова

ОБ ОБРАЗОВАТЕЛЬНОЙ ЦЕННОСТИ МОДЕЛЕЙ, ДОПУСКАЮЩИХ УПРАВЛЕНИЕ

Главная задача курса компьютерного моделирования – овладение учащимися моделированием как методом познания. Этому способствует использование в курсе компьютерного моделирования «мягких» моделей, допускающих возможность управления посредством изменения параметров. В качестве примера рассмотрена модель однородной популяции.

Главная задача курса компьютерного моделирования – овладение учащимися моделированием как методом познания.

Например, в вводной лекции к курсу «Компьютерное математическое моделирование» [1] при выделении основных задач курса говорится: «Основной упор необходимо сделать на выработку общего методологического подхода к построению математической модели и работе с ней. Здесь необходимо продемонстрировать, что моделирование в любой области знаний имеет схожие черты, зачастую для различных процессов удается получить очень близкие модели (например, модели межвидовой конкуренции и гонки вооружений). Таким образом, необходимо показать, как заметил А. Б. Горстко [2], что "ни ЭВМ, ни математическая модель, ни алгоритм ее исследования порознь не могут решить достаточно сложную исходную задачу. Но вместе они представляют ту силу, которая позволяет познавать окружающий мир, управлять им в интересах человека"».

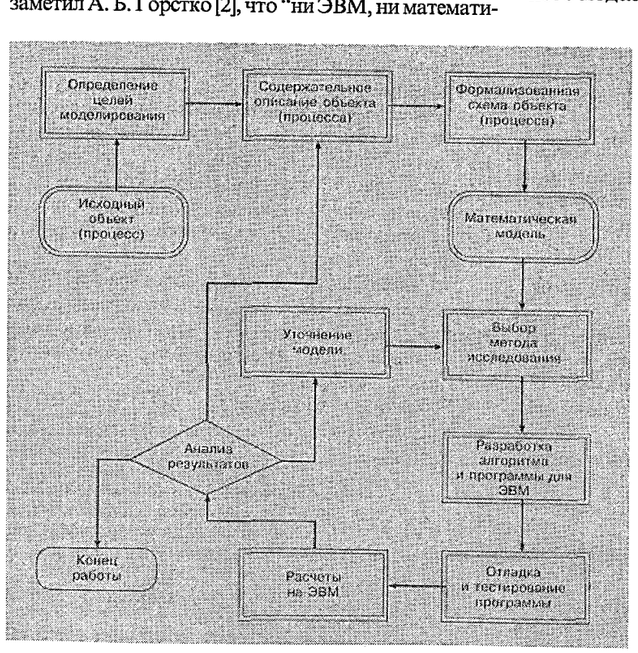


Рис. 1

В [1] предложена следующая схема компьютерного эксперимента (рис. 1).

Для выработки практических навыков компьютерного моделирования необходимо на примере ряда моделей проследить все этапы моделирования.

Остановимся более подробно на этапах «Расчеты на ЭВМ» – «Анализ результатов». Рассмотрим взаимодействие ученика и компьютерной модели, положив в основу общую схему управления (рис. 2).

Рис. 2

Ученик является субъектом управления, компьютерная модель – объектом управления. Наличие канала обратной связи дает возможность достигнуть цели моделирования – получить информацию о модели и тем самым об объекте моделирования. Если ученик имеет возможность вырабатывать управляющее воздействие (т. е. если модель допускает возможность управления), то работа с моделью будет циклической: управляющее воздействие – информация от модели – анализ информации – управляющее воздействие с учетом полученной информации – новая информация от модели и т. д. В этом случае возможно получение более полной информации об объекте моделирования.

Поэтому схему компьютерного эксперимента дополним возвратом к этапу «Расчеты на ЭВМ» с этапа «Анализ результатов» (рис. 3).

Исходя из вышеизложенного можно сделать вывод: в курсе компьютерного моделирования необходимо рассматривать «мягкие» модели, т. е. модели с параметрами, которые допускают возможность управления посредством изменения параметров. Работа с такими моделями происходит циклично, что дает возможность получить такие сведения о модели, которые были бы невозможны без многократной «прокрутки».

ОВСЯННИКОВА Марина Владимировна – аспирант кафедры информатики и МОИ ВятГГУ

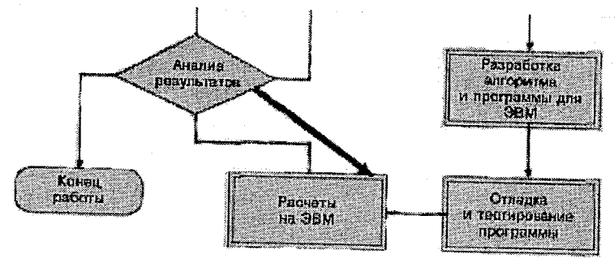


Рис. 3

Использование «мягких» моделей в преподавании курса компьютерного моделирования ставит перед учителем как минимум две дополнительные задачи. Первая задача: для конкретной модели необходимо сформулировать задачу и цель моделирования таким образом, чтобы возникла необходимость в «прокрутке». Это должна быть задача с элементами исследования. Задача вторая: научить исследовать модель посредством «грамотного» управления параметрами. Часто учащиеся сталкиваются с определенными трудностями из-за того, что изменяют параметры модели случайным образом, бессистемно. Вследствие этого они делают неправильные выводы или не могут сделать выводов вообще.

Управление параметрами модели может строиться по следующей схеме.

Подготовительный этап. Разделим параметры на две группы: начальные данные и параметры внешней среды.

Первый этап. Полагаем, что параметры внешней среды изменению не подлежат, и рассматриваем поведение модели при различных начальных данных (по возможности модель должна отражать поведение объекта на какой-либо совокупности начальных данных), делаем выводы. Так мы реализуем первый виток работы с моделью – получаем информацию о том, какие режимы могут установиться в системе по прошествии достаточно большого времени. Например, в химическом реакторе при различных начальных концентрациях вещества реакция может либо прекратиться, либо идти в пульсирующем режиме, либо протекать с постоянной скоростью. Кроме того, по возможности выяснить, будут ли эти режимы устойчивы, а также установить наличие или отсутствие аттракторов.

Второй этап. Выбираем один из параметров внешней среды и начинаем изменять его значение (остальные параметры зафиксируем). С каждым новым значением параметра проводим работу, описанную на первом этапе (реализуем второй, третий и т.д. витки). Выясняем: претерпевают ли качественные изменения режимы, которые могут установиться в системе (например, стационарный режим сменяется пульсирующим), если да – необходимо описать эти изменения и найти критические значения параметра.

Такое исследование можно проводить на достаточно простых моделях. Рассмотрим, как, например, может быть исследована модель однородной популяции. Стандартный подход: выбирается зависимость, описывающая динамику развития популяции, строится график, показывающий изменение численности при заданном начальном значении. Вывод, который делает учащийся: если начальная численность… то численность популяции не изменяется (возрастает, убывает).

Постановка задачи

Имеется некоторая популяция, численность которой изменяется с течением времени. Построить компьютерную модель, отражающую динамику развития этой популяции.

Математическая модель

Введем величины, характеризующие популяцию:

$x(t)$ – численность популяции в момент времени t;

$N=x(0)$ – численность популяции в начальный момент наблюдения.

Тогда $x'(t)$ – скорость изменения численности.

Будем исходить из предположения, что скорость изменения численности в момент времени t зависит от численности в этот момент времени, то есть $x'(t)=f(x(t))$.

Таким образом, для данной задачи математической моделью является дифференциальное уравнение первого порядка с начальным условием $x(0)=N$.

Пока мы еще не задали точный вид правой части дифференциального уравнения – функции $f(x)$. В правую часть уравнения помимо фазовой переменной могут входить параметры, определяемые внутренними свойствами популяции или внешними условиями (например, вылов).

Вычислительная модель

Полученная математическая модель является непрерывной моделью. Для построения вычислительной модели необходимо создать ее дискретный аналог. Будем использовать метод Эйлера численного решения дифференциального уравнения. Разобьем промежуток времени, в течение которого производится наблюдение, равноотстоящими точками.

Получим $t_{i+1}=t_i+\Delta t$, где Δt – фиксированная величина.

Для удобства обозначим $x(t_i)$ как x_i. Рассмотрим значение производной в узловой точке. Если Δt мало, то производную можно приближенно представить разностным отношением

$$x_i' \approx \frac{x_{i+1} - x_i}{\Delta t}$$

Заменим во всех узловых точках значение производной на разностное отношение $\frac{x_{i+1} - x_i}{\Delta t} = f(x_i)$.

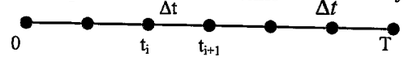


Рис. 4

Откуда получаем соотношение $x_{i+1}=x_i+\Delta t f(x_i)$

Это рекуррентное соотношение позволяет вычислить численность популяции в следующий момент времени, зная ее значение в предыдущий.

Компьютерная модель

Наша цель – построить компьютерную модель, которая позволяет численно решать дифференциальное уравнение $x'(t)=f(x)$ и построить несколько траекторий при различных начальных значениях численности.

Далее, изменяя в программе функцию $f(x)$, получим информацию:

- о поведении системы при фиксированных параметрах модели, иначе говоря, выясним, какие режимы могут установиться по прошествии достаточно большого времени. Это выясним не аналитически (исследуя $f(x)$), а опытным путем;
- о реакции системы на малое случайное колебание численности, иначе говоря, выясним, является ли система устойчивой;
- о поведении системы при изменении параметров, по возможности определим значение параметров, при которых система претерпевает качественные изменения, опишем эти изменения.

Работа с моделью

С целью исследования динамики развития популяции будем изменять в программе функцию $f(x)$.

1) Пусть $f(x)=kx$, $k>0$ (скорость изменения численности прямо пропорциональна численности, отрицательных факторов нет – модель Мальтуса).

Выводы

Поведение системы при фиксированном k: при любом начальном значении k>0 численность популяции неограниченно возрастает (рис. 5).

Зависимость поведения системы от параметров модели: поведение системы в целом не зависит от выбора (чем больше k, тем выше скорость роста и, соответственно, «круче» график).

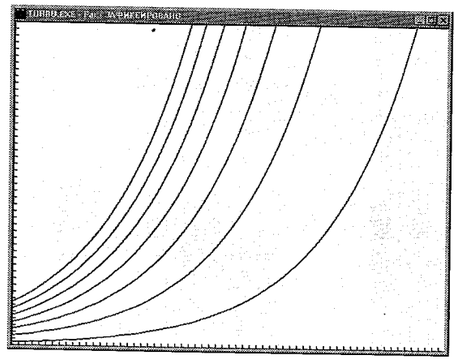


Рис. 5

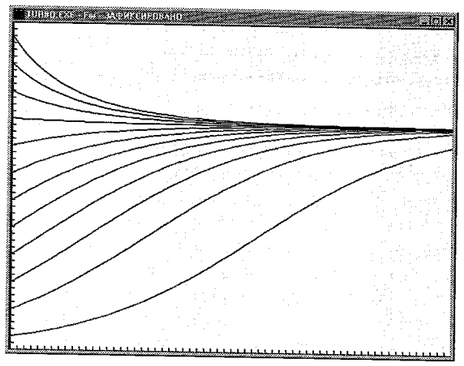


Рис. 6

Реакция системы на малое случайное колебание численности: небольшое изменение численности не влияет на поведение системы – численность популяции будет неограниченно возрастать.

Известно, что в природе не существует таких популяций.

2) Из опыта известно, что скорость роста популяции уменьшается при увеличении ее численности (сказывается нехватка территории и пищи, болезни). Поэтому в качестве $f(x)$ возьмем, например, функцию $ax-bx^2$. (a>0, b>0). Такой выбор приводит к логистической модели.

Выводы

Поведение системы при фиксированных a и b: численность популяции с течением времени стабилизируется на определенном уровне, не зависящем от начальной численности (рис. 6).

Зависимость поведения системы от параметров модели: при некоторых значениях параметров может получиться, что уровень, на котором стабилизируется численность, меньше нуля. Это означает, что при любой начальной численности популяция вымрет.

Реакция системы на малое случайное колебание численности: небольшое изменение численности не влияет на поведение системы. Если численность уже стабилизировалась и в какой-то момент ее значение немного изменилось (стало выше или ниже уровня стабилизации), то с течением времени система вернется в состояние равновесия.

Поведение системы при изменении параметров: если уменьшать параметр *a* (или увеличивать параметр *b*), уровень, на котором стабилизируется численность, будет понижаться и наоборот: с увеличением *a* (или уменьшением *b*) уровень стабилизации повысится. Можно сказать, что коэффициент «весомее»: уменьшение коэффициента *a* даст меньший сдвиг состояния равновесия, чем сдвиг, вызванный увеличением коэффициента *b* на ту же величину.

3) Пусть $f(x)=ax-bx^2-c$. На популяцию действует посторонний фактор –производится вылов с интенсивностью *c*.

Зависимость поведения системы от параметров модели: при небольшом значении параметра *c* система имеет два равновесных состояния M и N (M<N) (рис. 7). Если начальная численность меньше M (даже если отличие и невелико), то популяция неизбежно вымрет. Если начальная численность больше M, то с течением времени ее численность стабилизируется на уровне N (который немного меньше, чем в случае отсутствия вылова).

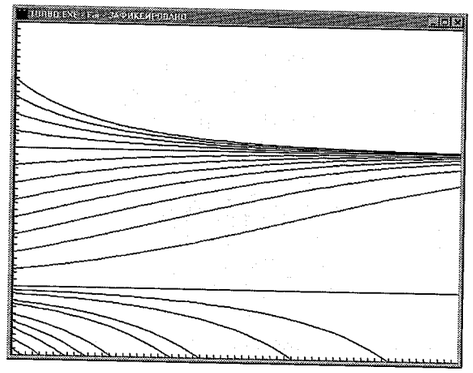


Рис. 7

Для этого случая *реакция системы на малое случайное колебание численности* будет такой: если вследствие малого случайного колебания численность станет меньше M, то популяция вымрет. Это может произойти только в том случае, если начальная численность лишь немного превосходит M и только если это колебание численности происходит в начальный период (так как с течением времени численность растет и сделать ее значение меньше M *малым* случайным изменением будет невозможно). Поэтому можно сделать вывод: если начальная численность больше M, то с течением времени система стабилизируется и вывести ее из состояния равновесия малым случайным изменением численности невозможно. Аналогично: если начальная численность была меньше M, то популяция вымрет за исключением тех случаев, когда начальная численность незначительно меньше M и в начальный период случайное колебание делает численность больше M (рис. 7).

При увеличении параметра *c* уровни M и N будут стремиться друг к другу. Если интенсивность вылова слишком велика – при любом начальном значении популяция вымрет. В этом случае малое случайное колебание численности не влияет на поведение системы (рис. 8).

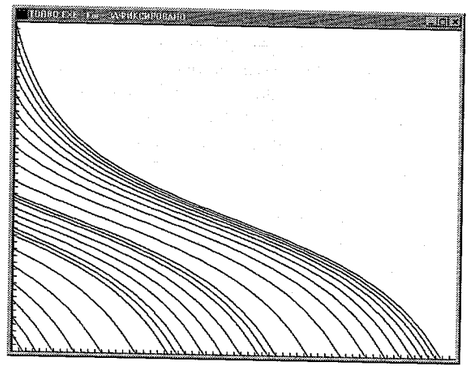


Рис. 8

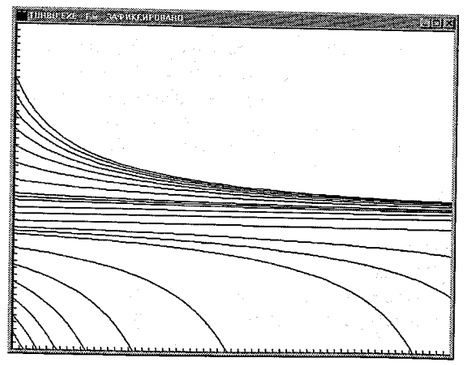


Рис. 9

Оптимальный с экономической точки зрения вариант – уровни M и N совпадают (вылов максимален, уничтожения популяции не происходит) (рис. 9). Если начальная численность меньше M, то популяция вымрет, а если больше – численность будет уменьшаться и стабилизируется на уровне M.

Для этого случая *реакция системы на малое случайное колебание численности* будет следующей: если

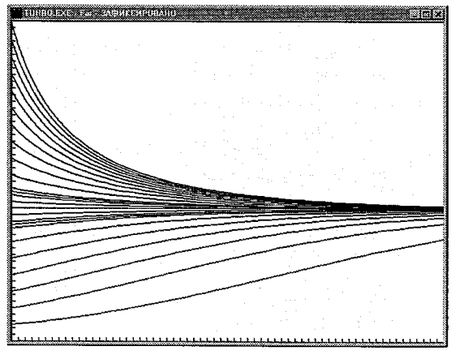


Рис. 10

вследствие малого случайного колебания численность станет меньше M, то популяция вымрет. Это может произойти в любой момент времени (в отличие от предыдущего случая, когда катастрофа была маловероятна и могла произойти только в начальный момент времени). Поэтому можно сделать вывод: если начальная численность больше M, то с течением времени система стабилизируется, однако это состояние не является устойчивым. Если вследствие каких-либо причин численность популяции немного уменьшится и станет меньше M (что вполне вероятно в реальных условиях) – популяция будет уничтожена, а если численность станет больше M – система вернется в состояние равновесия.

Поведение системы при изменении параметров. Будем изменять только коэффициент *c* при фиксированных значениях *a* и *b*. При небольших значениях *c* система имеет два равновесных состояния – устойчивое и неустойчивое. При увеличении коэффициента происходит сближение уровней стабилизации, затем происходит качественное изменение системы – два равновесных состояния сливаются в одно. Дальнейшее увеличение *c* ведет к исчезновению равновесия, популяция вымирает.

4) Заменим жесткий план вылова *c* величиной, пропорциональной фактической численности популяции (введем обратную связь).

$f(x)=ax-bx^2-kx$

Рассмотрим только оптимальный вариант – уровень вылова максимален, уничтожения популяции не происходит.

В отличие от рассмотренной выше системы, при любой начальной численности (даже достаточно низкой) популяция не будет уничтожена.

И *реакция на малое случайное колебание численности* будет другой: если вследствие малого случайного колебания численность станет меньше M, то популяция не вымрет, а с течением времени снова восстановит свою численность до уровня M. Система устойчива, случайное уменьшение численности к катастрофе не приведет.

Таким образом, было проведено исследование динамики развития однородной популяции. Рассмотрены различные функции $f(x(t))$, входящие в правую часть уравнения $x'(t)=f(x(t))$, сделаны выводы о зависимости поведения системы от параметров модели, получена информация о реакции системы на малое случайное колебание численности, изучено поведение системы при изменении параметра. Все выводы были получены только благодаря использованию компьютерной модели, аналитическое исследование функции $f(x(t))$ не проводилось. Поэтому данную модель можно исследовать даже в школьном курсе информатики, так как в этом случае не требуется специальных знаний по математике, кроме знания производной.

Примечания

1. *Шестаков А.П.* Профильное обучение информатике в старших классах средней школы (10-11-е классы) на основе курса «Компьютерное математическое моделирование» // Информатика. Прилож. к газ. «Первое сентября». 2002. № 34. С. 7-12.

2. *Горстко А.В.* Познакомьтесь с математическим моделированием. М.: Знание, 1991. 160 с.

А. А. Кочергин

ОСОБЕННОСТИ ПРЕПОДАВАНИЯ ИНФОРМАТИКИ В ОБЫЧНОЙ ШКОЛЕ

Работая в обычной школе, не трудно заметить, что как в отдельно взятом классе, так и в параллели могут учиться и способные дети, и слабоуспевающие – разброс между детьми и по скорости мышления и по уровню интеллекта достаточно велик. Но есть общее, что их объединяет в одну группу: у всех огромное желание обучаться информатике. Информатику любят все изначально, это надо использовать, и наша задача сводится к тому, чтобы показать ученикам направление, в котором необходимо двигаться. Мне кажется, что альтернативы программированию нет.

1. Программирование – инструмент развития ума. Не компьютер, а именно программирование.

2. Никакие информационные технологии не могут установить те межпредметные связи, которые устанавливает программирование, если, конечно, вы не понимаете под межпредметными связями использование компьютера в качестве ТСО. Для тех, кто именно так считает, поясню не таком примере: прослушивание музыки не приводит к тому, что Вы научитесь играть на музыкальном инструменте и тем более сочинять музыку.

3. Программирование воспитывает, потому что это труд.

4. Воздействие всегда труднее, чем созерцание «что делает компьютер». Преодолевая трудности, ребенок развивается.

Проблема в обычной школе может быть только одна: учить всех детей на одинаковом уровне нет возможности. А это уже проблема дифференциации, которая в свою очередь сводится к проблеме выбора учебно-методического материала. При его правильном подборе приходим к мысли, что языком программирования должен стать Паскаль.

В основе преподавания программирования лежит обратная связь (учитель – ученик), она рассматривается как **средство развития и воспитания.**

Не исключено, что это всего лишь особенность преподавания в конкретной школе конкретным учителем, тем не менее попробую это обосновать.

Отработав в школе несколько лет, практически любой учитель приходит к выводу, что в рамках традиционного обучения научить становится все труднее. Это связано, прежде всего, с огромным объемом информации, которую надо выдать ученику. Произошел информационный взрыв практически во всех областях человеческого знания. Ученик не успевает переваривать огромные объемы информации.

В рамках существующей классно-урочной системы сделать обучение развивающим не просто, а может быть, и невозможно, потому что обучаем КЛАСС, а не ученика.

Нужно перестраивать классно-урочную систему, точнее, отношения в ней.

Сейчас могу сказать, что это можно сделать не только на уроках информатики, но и на уроках математики и физики.

Итак, отношения учитель – ученик субъект-объектные отношения.

Если посмотреть историю развития человеческой цивилизации, то можно заметить, что развивающее обучение применялось и использовалось еще в древнегреческих академиях, а отношения учитель-ученик были **субъект-субъектные.**

Когда мы говорим УЧИТЕЛЬ и УЧЕНИК, то здесь нужно говорить о двух взаимодействующих субъектах, исследующих ту или иную область человеческого знания. Учитель, если говорить грубо, вообще «научать» не должен, «он должен накрыть стол».

И аргументом в защиту этого тезиса является то, что процесс запоминания, процесс мышления, творчества – это сугубо индивидуальный процесс.

Основные задачи, необходимые для организации преподавания на уроках информатики:

1. Организация процесса быстрого репродуктивного усвоения знаний (не более10 минут).

2. Создание условий для сотрудничества. Решение проблемы психологического взаимодействия учитель – ученик.

3. Создание условий для творческой деятельности ученика (опираясь на два предыдущих пункта).

Развивающее обучение, на наш взгляд, было реализовано еще в Древней Греции.

Здесь четко просматривается, что учитель передавал не только знания, но и определенный образ мышления, или то, что сейчас называется парадигмой мышления.

А как же шел процесс обучения? И самое главное – как греки передавали парадигму мышления?

Напомним, что в свое время Греция заняла ведущие позиции в культуре, а в науке – заглянули вперед на столетия и предопределили развитие современной Европы.

Историки описывают занятия в греческих Академиях как «прогулки по аллеям Академа под тенистыми деревьями, учитель вел *беседы* с учениками. У греков была организована сильнейшая обратная связь (учитель – ученик) в режиме реального времени, где ученик не только запоминал факты, получал знания но и, беседуя с учителем, перенимал логику его рассуждений, овладевал определенной характерной для своего времени ПАРАДИГМОЙ МЫШЛЕНИЯ.

Изучая культуру и историю древней Греции, можно указать на несколько замечательных моментов.

1. Именно культура Греции породила таланты, которые и предвосхитили многие открытия в математике, астрономии, философии. **Значит, истоком всего является культура.**

2. По древнегреческой технологии обучения ученик максимально был приближен к учителю и считал себя скорее собеседником, чем обучаемым. Таким образом была решена психологическая проблема (учитель – ученик).

3. Были созданы условия для творчества, так как ученик мог беседовать с учителем, обмениваться мнениями и в том числе оказывать влияние на учителя. Только в беседе вопрос может задаваться под конкретного ученика, с учетом его знаний и взглядов на проблему.

Значит, обучение было дифференцированное. Видим, что в греческой школе между учителем и учеником были **субъект-субъектные отношения.**

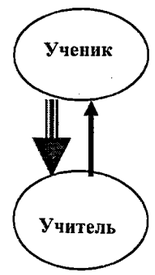


Рис. 1

Схема процесса познания выглядела так (рис. 1).

В качестве дополнительного примера можно вспомнить беседы Сократа со своими учениками, когда с помощью вопросов он подводил их к правильному решению.

И если сейчас рассмотреть происхождение и первоначальный смысл слова КУЛЬТУРА, то все становится ясным и понят-

КОЧЕРГИН Анатолий Анатольевич – учитель информатики школы № 65 г. Кирова

ным. Лат: Cultura – *возделывание, обрабатывание, уход,* а также воспитание и образование. В широком смысле все, что создано человеческим трудом, в противоположность тому, что получено в готовом виде от природы.

Подводя итог, можно сказать, что греки обрабатывали, возделывали человека, посредством той технологии, которой они владели.

А что сейчас? – Обучение, которое имеет три составляющих метода:

- показ;
- объяснение;
- контроль.

(Все остальное, то, что мы часто называем «новыми технологиями», это различные варианты к вышеописанной схеме, это скорее совершенствование вышеперечисленных трех методов, выдаваемое за новые технологии.)

Взаимодействие ученика и учителя здесь сводится до минимума. А когда и как передать парадигму мышления? Ведь итогом образовательного процесса образования должны стать не только усвоенные знания, но и способность самостоятельно оперировать ими, выносить суждения и, наконец, способность научить другого.

Процитируем в связи с этим профессора Ильенкова: «…ум – это не естественный дар природы, а результат **социально-исторического** развития человека». **Значит роль учителя в первую очередь – это работа над этой социальной составляющей,** (именно так и работали в греческом Академе). Социальная составляющая, как мне видится, – это *взаимодействие* людей, их общение, обмен мнениями, идеями, их культурное влияние друг на друга. Значит, можно сделать вывод, что умственные способности ребенка можно развить на основе общения (взаимного действия) учителя и ученика. Итак, мы вновь подошли к тому, чтобы вернуться к самой старой технологии преподавания, но уже в современных условиях. (Как мне кажется, реформа образования даст результат только тогда, когда вместе с разгрузкой учебного плана, будут введены новые технологии ведения занятий, иначе мы придем к тому же, но уже с меньшим багажом знаний, и вновь придется сокращать учебный план).

Развитие учащихся на уроках информатики посредством обратной связи

Согласно перечисленным выше задачам, за 16 лет в школе выстроилась определенная технология преподавания информатики, она основывается на следующих, сопутствующих информатике моментах.

- Деление класса на группы позволяет усилить и увеличить по времени контур обратной связи **учитель – ученик.**
- Как никакие другие занятия, они строятся на принципе обратной связи **компьютер-ученик,** действующей наиболее динамично и в реальном режиме времени. А в связи с тем, что принцип коммуникативности – одна из основных составляющих развития и творческой деятельности, можно сделать вывод, что информатика – это предмет, наиболее сильно воздействующий на ученика. Это **инструмент развития.**

Как построить уроки (занятия)?

Изложение теоретического материала, как правило, не должно выходить за рамки 10 минут. А дальше ученик должен закрепить эту теорию посредством решения одной-двух задач (как можно быстрее включаем обратную связь – **компьютер-ученик**), и по мере выхода учеников из первого контура сразу подключаем второй контур **– обратную связь (учитель – ученик).**

Поясним, как это сделать.

Необходимо, чтобы *каждый* ученик рассказал, как он решил задачу, коротко рассказал теоретическую часть своими словами. Здесь возможна и очень полезна «ручная» трассировка задачи. С этой целью учитель должен подойти и побеседовать с каждым учеником. В процессе рассказа, делаются необходимые поправки, могут задаваться любые вопросы в рамках изложенной ранее теории, уточняется, насколько ученик владеет теорией и применяет ее на практике (напомним, что вся теория уместилась в оперативной памяти ученика). На этом этапе учитель выступает в роли слушателя.

Назовем вышесказанное ***первым этапом*** занятия.

Положительные моменты:

- задействовав обратную связь *учитель – ученик*, устанавливаются личные контакты с учениками;
- все замечания будут сказаны в личной беседе, с учетом личных особенностей ученика;
- устанавливаем репродуктивный минимум, который он должен знать, а тех, кто не знает, ставим на «контроль.
- Ученик начинает **ГОВОРИТЬ**.

Ученик должен говорить, говорить и еще раз говорить. Лишь на первый взгляд информатика не разговорный предмет – здесь все можно изложить машинным кодом, но ученик не просто говорит, он говорит языком информатики, перестает бояться новых, порой не совсем понятных слов, ученик привыкает употреблять научные термины в своей живой речи и наконец запоминает материал.

Второй этап занятия (начало дифференциации).

После прослушивания учащегося и уяснения уровень его знаний, дается либо подобная задача, но в другой формулировке – для одних, либо задача исследовательского характера – для других.

Вопросы для беседы:

- «проверить, что будет, если мы сделаем, например вот так…»;
- «объясни полученный ответ…», или «объясни неверный ответ», если ответ не вписывается в рамки наших представлений об объекте.

Ошибки – источник противоречий. Не исправляйте ошибки механически, помните, как Сократ через ошибки, через противоречия между действительностью и моделью подводил собеседников к тому, что собеседник сам находил ошибки, делал правильный вывод. (Сделав такой вывод, ученик развивает ум, ибо ум – это прежде всего **умение самостоятельно** выносить суждения.)

В задачах повышенной трудности вопрос может быть поставлен, например такой: поясни идею решения задачи, какие подзадачи включены в данную задачу, где в природе, технике может встретиться такая или подобная задача. И наконец, составить тест для проверки и попытаться доказать, что тест должен быть именно таким. Время на собеседование с учеником на втором этапе занятия сокращается, беседа носит более общий характер. По времени обратная связь **компьютер – ученик** более продолжительна, чем **учитель – ученик.**

Очень важное замечание: Любой вопрос не должен содержать слово «ПОЧЕМУ». В вопросе «почему», и тем более «А почему», заложен элемент агрессивности, с какой бы интонацией Вы их ни произнесли.

В основе взаимодействия людей всегда лежит общение, точнее сказать, культура общения. На уроке же всегда есть общая тема разговора (есть общее дело – задача), рассказывая, ученик приобщается к общению, перенимает культуру его поведения и, самое главное, перенимает **его парадигму** мышления. Многим детям к тому же в семье не хватает именно общения.

Наконец, **третий этап** занятия – контроль знаний учащихся.

Традиционный метод – контрольная работа. Откажемся от такого способа – нет обратной связи (учитель – ученик). Для чего нужен контроль? Выявить уровень знаний и умений учащихся? Вы его уже выявили на первых двух этапах. Значит, этот этап надо превратить в творческую работу ученика, с кем-то – в собеседование, конкурс придуманных задач, а для кого-то нужна и традиционная контрольная работа. И снова, **обратная связь: учитель – ученик.**

Еще одним доводом для подобной технологии является то, что при подобной работе снижается уровень тревожности учащихся, а именно тревожность является тормозом в обучении, что особенно касается контрольных работ.

Схема новой технологии

Схема для реализации процесса познания с появлением нового инструмента – компьютера – сейчас выглядит следующим образом (рис. 2).

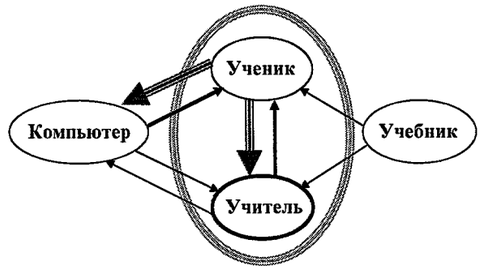


Рис. 2

(В центре выделена социальная составляющая, которую пытаемся усилить).

Процесс обучения становится **индивидуальным,** ибо *процесс познания, процесс мышления – это сугубо индивидуальные процессы.*

Маленькие замечания

1. Контур обратной связи (компьютер – ученик) наиболее динамичен, поэтому необходимо, особенно на первом этапе занятия, вести постоянный мониторинг слабо успевающих учеников.

2. Второй контур обратной связи (учитель – ученик) менее динамичен, но позволяет «вытаскивать» слабоуспевающего ученика и заинтересовать предметом.

Поэтому в целом повышение качества обучения в обычной школе, как мне кажется, может быть только в части усиления обратной связи (учитель – ученик).

3. Контур обратной связи **(компьютер-ученик)** – сильнейший инструмент развития личности и интеллекта ребенка, инструмент дифференциации обучения.

4. **Саму же парадигму мышления можно передать только посредством поставленной задачи.**

Предлагаемая технология уроков полностью согласуется с Федеральной программой развития образования (см.: «Учительская газета» от 13 ноября 2001 г.). Цитируем: «…На наших глазах происходит невиданное ускорение информационного взаимодействия людей… для школы это означает необходимость значительного усиления коммуникативных компетенций, будь то традиционные аспекты языковых коммуникаций или более современные направления использования информационных технологий. С этой целью в проекте существенно усилена филологическая подготовка школьников…».

Мы предлагаем делать это не только на гуманитарных предметах, но и на уроках информатики, физики, и других.

Еще одна цитата: «… необходимо формировать социальные умения общения, связанные с осознанием многообразия позиций. Способностью выслушать и понять другую точку зрения и терпимостью, критическим анализом аргументов и т. п.».

Ссылка на данный документ вовсе не означает, что автор полностью согласен с тем проектом содержания образования, который предлагается. Концепция разгрузки должна находится не в сфере сокращения объема знаний, а в сфере более эффективного их изложения и усвоения, в сфере внедрения таких технологий урока, при которых обучение должно происходить только на уроке.

В самом деле, где как не на уроке, общаясь с учителем, доказывая и рассказывая ему или своему товарищу, можно сформировать указанные в вышеприведенной цитате умения, и критически оценить уровень своей подготовки?

Еще один тезис в защиту такой технологии, а именно, попытаемся доказать то, что с ребенком нужно говорить, говорить и еще раз говорить.

Как мыслит человек – загадка, но то, что человек мыслит СЛОВОМ, броской, запоминающейся цитатой, – это факт. Что возникает сначала, мысль или слово? Так и хочется сказать, что сначала возникает мысль, а затем она мгновенно трансформируется в слово, но если вдуматься, то ЯЗЫК – это не только инструмент для общения, это ОПЕРАЦИОННАЯ СИСТЕМА человека: человек мыслит словами (говоря, он использует свою операционную систему). Значит, ребенка нужно приучать говорить. Он должен уметь «переводить» сказанное языком математики, информатики или другого предмета в мысль. Не владея СЛОВОМ, не понимая сути этого слова, наконец, просто не зная каких-то терминов, ребенок не сможет ими оперировать, а значит, описать окружающий мир, явления, или даже просто что-то воспроизвести.

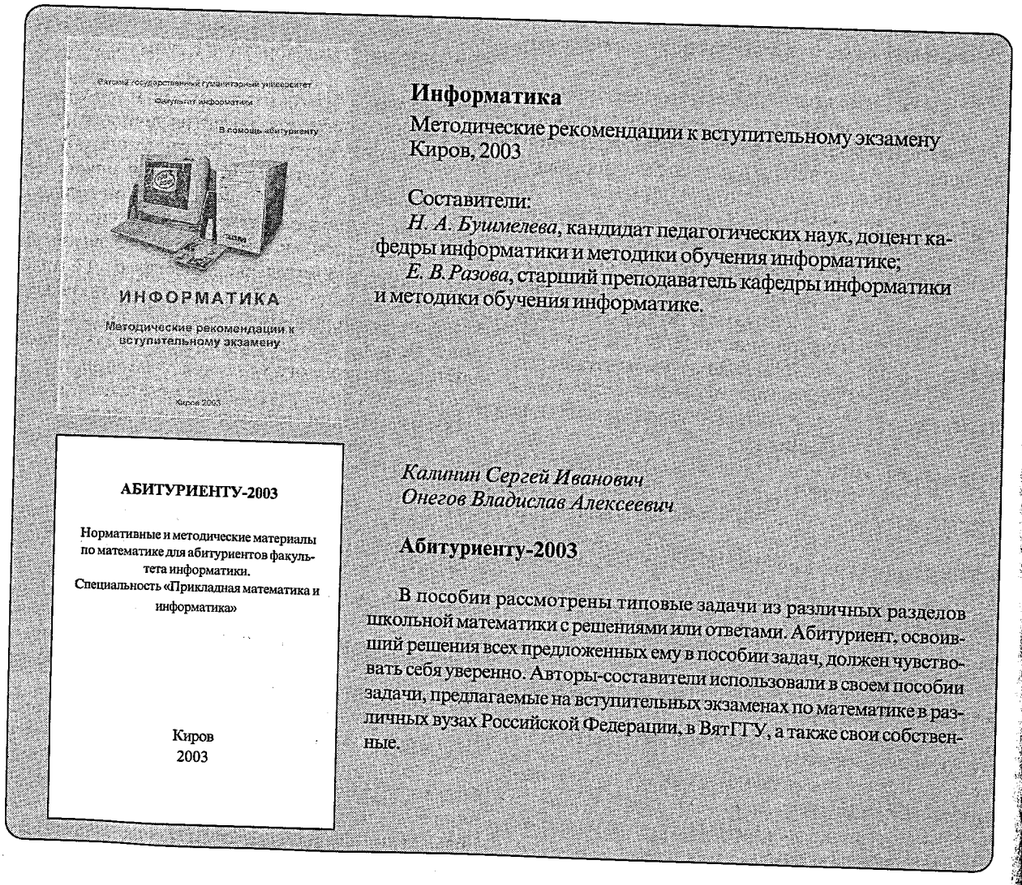
Как информация записывается в память человека? Словом, слово несет в себе ИНФОРМАЦИЮ – суть явления, значит, учитель должен разговаривать с учениками, учить их говорить, и не важно – язык ли это математики, физики или информатики.

Результат

1. Обучаясь на факультетах, связанных с информатикой, студенты не испытывают затруднений с предметом.

2. Будучи студентами, они могут подойти к любому преподавателю, с любым вопросом, при этом всегда корректны и не испытывают чувства «страха» при общении.

3. Имеются победители и призеры городских олимпиад), призеры открытых областных олимпиад.



Информатика

Методические рекомендации к вступительному экзамену
Киров, 2003

Составители:

Н. А. Бушмелева, кандидат педагогических наук, доцент кафедры информатики и методики обучения информатике;

Е. В. Разова, старший преподаватель кафедры информатики и методики обучения информатике.

Калинин Сергей Иванович
Онегов Владислав Алексеевич

Абитуриенту-2003

В пособии рассмотрены типовые задачи из различных разделов школьной математики с решениями или ответами. Абитуриент, освоивший решения всех предложенных ему в пособии задач, должен чувствовать себя уверенно. Авторы-составители использовали в своем пособии задачи, предлагаемые на вступительных экзаменах по математике в различных вузах Российской Федерации, в ВятГГУ, а также свои собственные.

ИНФОРМАТИКА И ЯЗЫК

А. И. Бардовская

СИНЕСТЕТИЧЕСКИЕ СЛОВОСОЧЕТАНИЯ С КОМПОНЕНТАМИ, ОПИСЫВАЮЩИМИ ТЕМПЕРАТУРНЫЕ ОЩУЩЕНИЯ

В статье приводится краткий обзор различных точек зрения на проблему синестезии и описывается подготовка к психолингвистическому исследованию синестетических словосочетаний на материале русского языка.

Одной из актуальных проблем современного лингвистического исследования является синестезия, играющая важную, но малоизученную роль в когнитивных процессах индивида [1]. Несмотря на разнообразие подходов и точек зрения, представляется возможным выделить общее в рассуждениях лингвистов: синестезия - явление, основанное на переносе, то есть представляет собой разновидность метафоры [2], особенностью которой является то, что сдвиг в значении слова происходит в сфере чувственного восприятия. Лингвистический анализ синестезии направлен на выявление ее особенностей как языкового феномена, структуры и значения синестетических переносов, роли в системе языка и его функционировании, и в этом направлении проделана немалая работа [3]. Однако можно утверждать, что исследование синестезии до сих пор не вышло за пределы первоначальной стадии, и множество вопросов, связанных с этим сложным явлением, остаются без ответа [4].

Проблема синестетического переноса, решение которой невозможно без рассмотрения таких вопросов, как лексическое значение в структуре многозначного слова и его изменение, представляет для нас наибольший интерес. В современной лингвистической парадигме на первое место выходит отношение «значение – внутренний мир индивида» [5], и потому необходим поиск методик, позволяющих исследовать значение как достояние индивида, переходить от исследования значения как абстрактной сущности к изучению концепта как сущности ментальной прежде всего. По мысли А.А. Залевской, «значение слова должно рассматриваться в совокупности с процессами его функционирования и с проблемой организации не только внутренней структуры значения, но и всего того, что связано с отдельным словом и с лексиконом в целом, с местом последнего в языковом/речевом механизме человека и в системе познания» [6].

По мнению М. Я. Сабанадзе [7], далее других в объяснении синестетического переноса как явления, с одной стороны, психофизиологического, а с другой – лингвистического продвинулись представители звукосимволизма, в связи с чем одним из аспектов нашего исследования мы предполагаем сделать фоносемантический анализ.

Явления окружающего мира обладают множеством аспектов, тем не менее взаимодействие перцептивных систем обеспечивает их целостное восприятие, восприятие мира в единстве [8]. По мнению В. П. Зинченко, язык синестезий – один из образных языков, с помощью которых строится образно-концептуальная модель мира человека [9]. Таким образом, можно предположить, что человек постоянно сравнивает, оценивает свои ощущения, благодаря чему строится его концептуальная картина мира. Являясь сложным, субъективным, эмоциональным явлением, синестезия, несомненно, связана с рядом важных психических процессов, и потому нам представляется интересным рассмотреть данный феномен, связав его с анализом ментальных репрезентаций.

По мысли П. Гэрденфорса [10], чья теория обладает большой объяснительной силой относительно образования метафор и, в частности, синестетических метафор, основой репрезентаций являются концептуальные пространства. Концепты взаимосвязаны, зависимы друг от друга и структурируются в домены. В теории концепта данного исследователя ключевым понятием становится качественное измерение, главная роль которого – образование домена, необходимого для репрезентации концептов. К числу простейших качественных измерений П. Гэрденфорс относит температуру, вес, яркость, высоту звука и три простых пространственных измерения – высоту, ширину и глубину, тесно связанные с нашими рецепторами, воспринимаемые разными органами чувств. Между тем в работе отмечается, что существует ряд качественных измерений абстрактного, нечувственного характера. Измерения образуют систему, оценивающую качества предметов и определяющую отношения сходства и различия между ними. Концептуальные структуры «телесны» (т. е. значения связаны

БАРДОВСКАЯ Анастасия Игоревна – аспирант кафедры германских языков

8 Зак. 2868

с восприятием и телесным опытом), и должны существовать уровни ментальной репрезентации, где информация, выражаемая языком, совмещается с информацией, получаемой периферийными системами: зрением, слухом, обонянием, кинестезией и т. д. Если бы этих уровней не существовало, было бы невозможно использовать язык для передачи чувственных ощущений.

При отборе экспериментального материала, то есть лексики, описывающей различные ощущения и восприятия (качественные измерения, в терминологии П. Гэрденфорса), а следовательно, потенциально участвующей в синестетических словосочетаниях, мы руководствовались положениями П. Гэрденфорса относительно качественных измерений, а также классификацией, предложенной Е.А. Елиной [11]. Данный автор выделяет следующие виды ощущений и восприятий: слуховые (звуковые, музыкальные и языковые), обонятельные, вкусовые, тактильные, температурные, двигательные, гравитационные, пространственные, временные, вибрационные, болевые, степени интенсивности, психических состояний (эмоции и характер). Нами были использованы толковые словари русского языка [12], а также материалы исследований О. И. Кундик [13]. Методом сплошной выборки были отобраны более 400 синестетических словосочетаний русского языка, ярко демонстрирующих различные интермодальные ощущения (например, совмещение вкусовых и обонятельных ощущений, тактильных и вкусовых, болевых и слуховых и т. д.).

Анализ литературы показывает, что в исследованиях синестезии значительное место отводится изучению синестетических словосочетаний, а именно синестетических словосочетаний синтаксической конструкции прилагательное + существительное, что дает возможность утверждать, что именно прилагательное обладает наибольшим потенциалом к образованию синестетических значений [14]. Однако лексико-семантическое поле восприятия в системе языка не ограничивается прилагательными [15], и мы стремились включить в предполагаемый экспериментальный материал синестетические словосочетания с различными типами грамматической связи, например, синтаксические конструкции глагол + существительное.

Как одно из изобразительных средств языка синестезия классифицируется по степени выразительности. Так, ряд изученных нами работ по проблеме синестезии посвящен так называемой узуальной, традиционной синестезии [16], объектом интереса других исследователей является синестезия специальных подъязыков (музыковедения, искусствоведения) [17], третьи изучают авторскую синестезию, присущую индивидуальному стилю писателей [18. По нашему мнению, в данном случае можно говорить о противопоставлении так называемых «мертвых» и «живых» метафор, представляющем еще один аспект изучения специфики значения. Как отмечает Е. Ю. Мягкова [19], «живые», новые метафоры отражают изменение в отношении людей к тому, о чем они думают и говорят, в то время как «мертвые», старые метафоры бессознательно встроены в язык давно установившимися привычками и потому являются самыми важными с когнитивной точки зрения. Одной из целей нашего исследования является проверка того, каким образом отобранные, по нашему предположению, узуальные синестезийные словосочетания функционируют в индивидуальном сознании и в какой степени они могут быть отнесены к узуальным. Для исследования этого аспекта явления мы предполагаем использовать метод субъективных дефиниций.

Как отмечает А. В. Житков [20], специалисты значительно большее внимание отводят лексике зрительного и слухового восприятия, чем лексике со значением осязания, вкуса, запаха. Нам представилось интересным рассмотреть синестетические словосочетания с компонентами, описывающими температурные ощущения, представляющие, согласно теории П. Гэрденфорса, одно из простейших качественных измерений. Температура – физическое свойство всех предметов окружающего нас мира, но что есть температурные качества, когда их оценивает, сравнивает переживающий, чувствующий человек?

По мысли П. Гэрденфорса, универсального способа репрезентации измерений не существует: каждая культура и отдельно взятый индивид обладает собственной системой измерений. Анализ пятидесяти отобранных нами словосочетаний русского языка позволяет представить концептуальное измерение температуры в виде восьмиугольника, гранями которого будут являться зной, жар, «горячесть», теплота, прохлада, холод, лед и мороз.

На данном этапе исследования нам представляется возможным сгруппировать экспериментальный материал по принципу совмещения ощущений и восприятий семи различных модальностей с температурными ощущениями.

Слух	Вкус	Зрение	Время	Эмоции и психические состояния	Интенсивность	Боль
3	6	12	4	21	3	1

Таким образом, можно говорить о разной степени совмещения температурных ощущений с ощущениями других модальностей. То же самое наблюдается и относительно интермодальных ощущений, в которых задействуются различные грани концептуального измерения температуры.

Зной

Зрение	Эмоции и психические состояния	Интенсивность
Знойный взгляд	*Знойная страсть*	*Палящий зной*
1	1	1

Жар

Зрение	Эмоции и психические состояния
Жаркий цвет	*Жарко спорить*
3	2

«Горячесть»

Вкус	Время	Эмоции и психические состояния
Горячие напитки	*Горячая пора*	*Горячо любить*
3	4	7

Теплота

Зрение	Эмоции и психические состояния	Слух
Теплые тона	*Теплота чувств*	*Теплый голос*
2	4	1

Прохлада

Эмоции и психические состояния
С прохладцей
2

Холод

Вкус	Зрение	Интенсивность	Эмоции и психические состояния	Боль
Мята холодит	*Холод в глазах*	*Резкий холод*	*Холодный нрав*	*Колючий холод*
2	5	1	3	1

Лед

Слух	Вкус	Эмоции и психические состояния	Зрение
Лед в голосе	*Ледяная вода*	*Леденить сердце*	*Ледяной взгляд*
2	1	2	1

Мороз

Интенсивность
Крепкий мороз
1

Остается нерешенным вопрос о том, каким образом происходит совмещение температурных измерений с другими, благодаря чему становятся возможными «теплые оттенки», «горячие чувства», «ледяной тон» и т. п. В [21] Р. Браун говорит о том, что связь между температурой, манерами, выражением лица и голосом может устанавливаться благодаря некой общей ассоциации. Проверить существование подобных ассоциаций мы планируем, используя ассоциативный эксперимент.

Исследователями признается, что один из компонентов синестетического словосочетания – синестезирующее, а второй – синестезируемое [22]. С целью анализа того, что в отобранных нами словосочетаниях будет являться в индивидуальном сознании носителя языка синестезирующим, а что – синестезируемым, мы планируем использовать метод мысленных образов.

Метафора упоминается во всех работах, связанных с изучением эмотивности лексики [23]. Возможно, одним из факторов, позволяющих говорить о «ледяной воде» и «ледяном спокойствии», будет общность эмоциональной оценки, эмоциональная оценка будет являться медиатором синестетического переноса. Для проверки этой гипотезы мы планируем использовать метод субъективного шкалирования.

Многие аспекты синестезии, касающиеся как самого объекта исследования, так и его признаков, механизмов и закономерностей функционирования, еще не нашли своего окончательного разрешения. Выбор целого комплекса методик исследования обусловлен сложностью интересующего нас феномена. Но именно его сложность и неоднозначность позволяют говорить о перспективности более углубленного исследования явления в свете междисциплинарного подхода к изучению языка как достояния индивида.

Примечания

1. *Бардовская А. И.* Разные подходы к синестезии // Психолингвистические исследования: слово и текст. Тверь: ТвГУ, 2002. С. 16-22; *Дунаев В.* Цвет головной боли. Синестезия: аномалия или дар природы? // eidos.kiam.ru/group/cvee.html.

2. *Агалакова Т. Б., Межбурд М. Е.* Некоторые особенности прилагательных синестезической семантики в современном английском языке // Вопросы романо-германской филологии. Киров: Изд-во ВГПУ, 2000. С. 3-8; *Бардовская А. И.* Указ. соч.; *Елина Е. А.* Вербальные интерпретации произведений изобразительного искусства. Саратов, 2002. 255 с.; *Житков А. В.* Функционально-семантическое поле восприятия запаха и синестезии одорической лексики в произведениях И. А. Бунина: Автореф. дис. ... канд. филол. наук. Екатеринбург, 1999. 19 с.; *Мягкова Е. Ю.* Проблемы исследования метафоры // Языковое сознание: формирование и функционирование. М., 2000. С. 123–128; *Петренко В. Ф.* Психосемантика сознания. М.: Изд-во Моск. ун-та, 1988. 208 с.; *Сабанадзе М. Я.* Синестезия в подъязыке музыковедения (на материале англ. яз.): Автореф. дис. … канд. филол. наук. Л.: ЛГУ им. А. А. Жданова, 1987. 21 с.; *Ullmann S.* The Principles of Semantics. Glasgow, 1959. 348 p.

3. *Агалакова Т. Б., Межбурд М. Е.* Указ. соч.; *Житков А. В.* Указ. соч.; *Петренко В. Ф.* Указ. соч.; *Ullmann S.* Cit. op.

4. *Бардовская А. И.* Указ. соч.

5. *Сазонова Т. Ю.* Различные подходы к трактовке концепта // Слово и текст в психолингвистическом аспекте. Тверь: ТвГУ, 2000. С. 70–76.

6. *Залевская А. А.* Значение слова и возможности его описания // Языковое сознание: формирование и функционирование. М., 2000. С. 50.

7. *Сабанадзе М. Я.* Указ. соч.

8. *Величковский Б. М., Зинченко В. П., Лурия А. Р.* Психология восприятия. М.: Изд-во Моск. ун-та, 1973. 246 с.

9. *Зинченко В. П.* Мышление и язык: Учеб. пособие. Дубна, 2001. 140 с.

10. *Gaerdenfors P.* Conceptual Spaces. The Geometry of Thought. Cambridge: MA; London: The MIT Press, 2000.

11. *Елина Е. А.* Указ. соч.

12. *Даль В. И.* Толковый словарь живого великорусского языка. Т. 1-4. М.: Рус. яз., 1978; *Ефремова Т. В.* Новый словарь русского языка (толково-словообразовательный).

Т. 1-2. М.: Рус. яз., 2000. 1209 с.; *Ожегов С. И.* Словарь русского языка. М.: Рус. яз., 1988. 750 с.

13. *Кундик О. И.* Специфика синестезийных словосочетаний в русском и английском языках: Автореф. дис. … канд. филол. наук. Саратов, 1997. 18 с.

14. *Агалакова Т. Б., Межбурд М. Е.* Указ. соч.; *Кундик О. И.* Указ. соч.; *Сабанадзе М. Я.* Указ. соч.; *Brown R.* Words and Things. Illinois: Glencoe, 1958. 398 p.; *Ullmann S.* Cit. op.

15. *Житков А. В.* Указ. соч.

16. *Агалакова Т. Б., Межбурд М. Е.* Указ. соч.; *Кундик О. И.* Указ. соч.; *Ullmann S.* Cit. op.

17. *Горелов Н. И.* Синестезия и мотивированные знаки подъязыков искусствоведения // Проблемы мотивированности языкового знака. Калининград, 1976. С. 74-81; *Елина Е. А.* Указ. соч.; *Сабанадзе М. Я.* Указ. соч.

18. *Галеев Б. М.* Синестезия – чудо поэтического мышления // Научный Татарстан. 1999. № 3. С. 19-23; *Житков А. В.* Указ. соч.

19. *Мягкова Е. Ю.* Указ. соч.

20. *Житков А. В.* Указ. соч.

21. *Brown R.* Cit. op.

22. *Сабанадзе М. Я.* Указ. соч.

23. *Мягкова Е. Ю.* Указ. соч.

Е. Л. Воробьева

ПСИХОЛИНГВИСТИЧЕСКОЕ ИССЛЕДОВАНИЕ ПРЕЦЕДЕНТНЫХ ФЕНОМЕНОВ В ОБЛАСТИ COMPUTER SCIENCE

Имена 10 деятелей в области информатики рассматриваются как прецедентные феномены. В ходе анализа результатов психолингвистического эксперимента выявлено десять моделей идентификации данной группы антропонимов.

В настоящее время в связи с изменением общей направленности лингвистических исследований, а именно с переносом акцента с системно-структурных особенностей слова и текста на когнитивные аспекты языка, многие традиционные вопросы лингвистики рассматриваются под новым углом зрения, а именно через их преломление индивидуальным человеческим сознанием, что позволяет включить их в современный лингвистический процесс и использовать при построении новейших концепций понимания слова и текста.

В связи с этим настоящее исследование представляется, безусловно, актуальным, поскольку позволяет впервые рассмотреть, как создается в голове индивида, знающего Computer Science, своеобразная «проекция» образа деятеля в области информатики и ключевых понятий данной области знания. Мы опираемся на концепцию прецедентности, имеющую большую объяснительную силу при решении сложных вопросов взаимоотношения языка, культуры и сознания. Прецедентные феномены (далее – ПФ) являются основными компонентами общего для всех членов лингвокультурного сообщества ядра знаний и представлений. ПФ – это «культурные предметы», которые отражают и определяют специфику национального характера, этнического и языкового сознания. Это – прецеденты, образцовые факты, служащие моделью для воспроизводства сходных фактов и представленные в речи определенными вербальными сигналами, актуализирующими стандартное содержание, которое не создается заново, но воспроизводится.

Понятие прецедентного текста еще в 1987 году ввел в научный обиход Ю. Н. Караулов, понимая под этим «тексты, значимые для той или иной личности в познавательном и эмоциональном отношении, имеющие сверхличностный характер, то есть хорошо известные широкому окружению данной личности, включая ее предшественников и современников, и, наконец, такие, обращение к которым возобновляется неоднократно в дискурсе данной языковой личности» [1]. Среди ПФ помимо прецедентных текстов принято выделять также прецедентные высказывания и прецедентные имена.

Материалом нашего экспериментального исследования послужили ПФ всех вышеназванных трех уровней: 10 прецедентных имен-антропонимов, а именно имен ученых, определивших возникновение и становление информатики, 5 ключевых понятий, точнее ключевых копметенций курса информатики в школе [о ключевых компетенциях см.: 2] и прецедентный текст – программа, написанная на языке «Turbo Pascal», осуществляющая поиск подстроки в строке (алгоритм Кнута-Мориса-Пратта).

В данной статье мы остановимся на анализе прецедентных имен и на основе интерпретации результатов свободного ассоциативного эксперимента рассмотрим особенности идентификации антропонимов – имен деятелей информатики.

Список из 10 исследуемых имен был составлен методом экспертной оценки. Экспертами выступали 15 студентов IV курса факультета информатики. Из индивидуальных списков экспертов были отобраны 10 самых частотных слов – имен деятелей информатики:

1. ***Никлаус Вирт*** (в 1968-1969 гг. разработал язык программирования «Pascal»).

2. ***Кемени*** (работал преподавателем в Дартмутском колледже, изобрел язык программирования «Basic»).

3. ***Бэкус*** (возглавлял группу для разработки языка «Фортран» для решения математических задач).

4. ***Б. Гейтс*** (вместе с Полом Аленом создали «Basic» в машинных кодах, то есть интерпретатор Basic. Образовали свою фирму «Microsoft»).

ВОРОБЬЕВА Елена Леонидовна – студентка V курса факультета информатики ВятГТУ

5. ***Джобс*** (придумал компьютеры «Macintosh», «NeXT»).

6. ***Филипп Кан*** (разработал компактный, моментально срабатывающий компилятор «Turbo Pascal». Он основал фирму «Borland International»).

7. ***Барнеби*** (вместе с Рубинштейном реализовали идею «Word Star» – текстовый редактор, который быстро завоевал рынок и стал стандартом для изготовления других текстовых редакторов).

8. ***Дэниел Бриклин*** (разработал программу – табличный процессор (электронной таблицы)).

9. ***Энгельбарт*** (изобрел «мышь» – указатель X,Y – позиции для дисплея).

10. ***Сеймур Пейперт*** (реализовал идею «обучение во время игры»: язык программирования «Черепашка»).

В качестве второй методики был использован ассоциативный эксперимент в его свободной модификации. В рамках свободного ассоциативного эксперимента испытуемым при предъявлении слова-стимула предлагалось как можно быстрее написать от двух до четырех слов-ассоциаций, которые бы характеризовали прецедентное имя или высказывание. Чтобы испытуемым было более понятно задание эксперимента, им был дан пример слов-ассоциаций на прецедентное имя, не входящее в экспериментальный набор. В дополнение было сказано, что ассоциациями могут быть не только слова, но и словосочетания, и предложения, в общем, любые слова-реакции, пришедшие в голову.

Всего было получено 174 ассоциации на русском языке. Приведем списки полученных ассоциаций (ассоциативное поле) на стимул НИКЛАУС ВИРТ – «Pascal» (11 реакций) (здесь и далее слово-стимул пишем заглавными буквами; слово-ассоциат – строчными), язык программирования (6), программирование (4), структурное программирование (4), зарезервированные слова (1), операторы (1), нет ассоциаций (1) (всего 26 реакций).

Обратимся к анализу полученных ассоциативных полей и выделим основные стратегические идентификации имен деятелей информатики. Под идентификацией мы, вслед за А. А. Залевской [3] понимаем процесс соотнесения слова с разнообразной информацией, которая стоит за словом в индивидуальном сознании. Стратегические модели позволяют объяснить, как в рамках внутреннего когнитивного контекста человек интегрирует и организует новую информацию, заполняет пробелы в случае поступления недостаточной или неполной информации, осуществляет доступ к конкретным аспектам вербальной информации. При идентификации имен литературных персонажей и исторических личностей используется ряд специфических стратегий, благодаря которым эпоним включается в обширную систему связей по линии как энциклопедических, так и языковых знаний, сопровождаемых эмоционально-оценочным маркированием [4].

С. С. Хватова отмечает, что имена собственные как единицы речи, служащие для выделения именуемого ими объекта из ряда подобных, как ничто другое отражают историю, культуру и сознание народа. Но еще более точно и глубоко отражают картину мира прецедентные имена. Прецедентное имя является не только вербальным феноменом, оно обозначает и то, что стоит за ним: совокупность дифференциальных признаков и атрибутов. Дифференциальные признаки могут выражать характеристику предмета по чертам характера (Шерлок Холмс – *умный, проницательный*), или по внешности (Квазимодо – *уродливый горбун*), они могут актуализироваться через прецедентную ситуацию (Леди Макбет – *убийство*). Атрибутами считаются «элементы», тесно связанные с прецедентными именами, но не необходимые для его сигнификации (*Трубка Шерлока Холмса, шарф Остапа Бендера, диван Обломова* и т.д.) [5].

С. С. Хватова вслед за Ю. Н. Карауловым и Э. Е. Каминской считает, что знакомое имя открывает человеку доступ к широчайшему пласту экстралингвистических знаний, связанных с особенностями национальной культуры и истории, чертами национального характера, спецификой речевого и сопутствующего ему поведения данной общности людей. Особое место в ассоциативно-вербальной сети любого носителя языка принадлежит антропонимам, среди которых выделяется группа «знаменитых» имен и фамилий (исторические деятели, выдающиеся представители искусства культуры, литературные персонажи). Последние задают уникальный экстралингвистический фон, являющийся неотъемлемой частью информационного тезауруса члена данной языковой общности и выступающий непреодолимым препятствием для иностранца. Картирование подобных культурологических стереотипов должно способствовать прогнозированию и нейтрализации наиболее вероятных критических моментов в межэтнических контактах [5].

Итак, рассмотрим полученные идентификационные модели имен деятелей информатики:

1) идентификация через название изобретения/достижения (53,45%):

- ВИРТ – «Pascal»;
- КЕМЕНИ – язык программирования «Basic»;
- БЭКУС – «Фортран»;
- ГЕЙТС – интерпретатор языка «Basic», «Microsoft Word», «Microsoft», «Microsoft Excel», «Windows»;
- ДЖОБС – «Macintosh», «NeXT», «Macintosh LC-50», язык программирования «Post Script», операционная система «Next Step»;
- КАН – «Turbo Pascal»;
- БАРНЕБИ – «Word Star»;
- БРИКЛИН – табличный процессор;
- ЭНГЕЛЬБАРТ – «мышка»;
- ПЕЙПЕРТ – «Черепашка», «Лого»;

2) идентификация через дефиницию изобретения (15,52%):

• ВИРТ – язык программирования;
• КЕМЕНИ – язык программирования;
• БЭКУС – язык программирования;
• БАРНЕБИ – текстовый редактор;
• ЭНГЕЛЬБАРТ – указатель позиции курсора;
• ПЕЙПЕРТ – язык программирования;

3) идентификация через действие, на которое направлено изобретение (6,32%):
• ВИРТ – программирование;
• КЕМЕНИ – программирование;
• БАРНЕБИ – редактирование текстов;

4) идентификация через характеристику действия, на которое направлено изобретение (5,17%):
• ВИРТ – структурное программирование;
• КАН – структурное программирование, объектно-ориентированное программирование;
• ПЕЙПЕРТ – программирование для детей;

5) идентификация через характеристику изобретения (4,02%):
• БЭКУС – язык высокого уровня;
• ДЖОБС – компьютеры, ориентированные на среду образования, перые мультимедийные компьютеры;
• ПЕЙПЕРТ – обучение + игра;

6) идентификация через название объектов, на работу с которыми направлено изобретение (8,62%):
• БЭКУС – математические задачи, программа;
• БАРНЕБИ – текст;
• БРИКЛИН – электронные таблицы, таблица;

7) идентификация через название качеств, характеризующих деятеля (1,15%):
• ГЕЙТС – ум, всемирная известность;

8) идентификация через имя другого деятеля (0,57%):
• БАРНЕБИ – Рубинштейн;

9) идентификация через название частей изобретения (5,17%):
• ВИРТ – зарезервированные слова, операторы;
• КАН – оператор, цикл, массив, зарезервированные слова, процедуры, функции.

Итак, самой актуальной стратегией идентификации оказалась стратегия 1, набравшая 53,45%. Актуальными стратегиями идентификации являются также стратегии 2, 3, 4, 5, 6, 9, набравшие от 5,17 до 15,52%, в то время как стратегии 7, 8 играют лишь вспомогательную роль и набирают 1,15 и 0,57% соответственно.

Таким образом, анализ выделенных стратегий идентификации имен выдающихся деятелей в области информатики продемонстрировал, что их опознание произошло через стратегии, связанные с Computer Science. Как показывают результаты эксперимента, в первую очередь человек воспринимает выдающегося деятеля в области информатики через название его изобретения. Это доказывает значимость данных изобретений для различных областей программирования и неразрывность имени изобретателя и его изобретения в индивидуальном сознании студента факультета информатики.

Примечания

1. *Караулов Ю. Н.* Русский язык и языковая личность. М.: Наука. 1987. 264 с.
2. *Окулов С. М.* Когнитивная информатика: Монография. Киров: Изд-во ВятГГУ, 2003. 234 с.
3. *Залевская А. А.* Введение в психолингвистику. М.: Рос. гос. гуманит. ун-т, 2000. 382 с.
4. *Залевская А. А.* Индивидуальное знание: специфика и принципы функционирования. Тверь: Твер. гос. ун-т, 1992. 135 с.
5. *Хватова С. С.* Лингвокультурная информативность прецедентного имени // Слово и текст: психолингвистический подход. Вып. 1. Тверь, 2003. С. 153-157.

В. С. Гуляева, С. Ю. Иванов

СЦЕНАРНЫЕ УРОКИ ИНОСТРАННОГО ЯЗЫКА: МОДА ИЛИ НЕОБХОДИМОСТЬ?

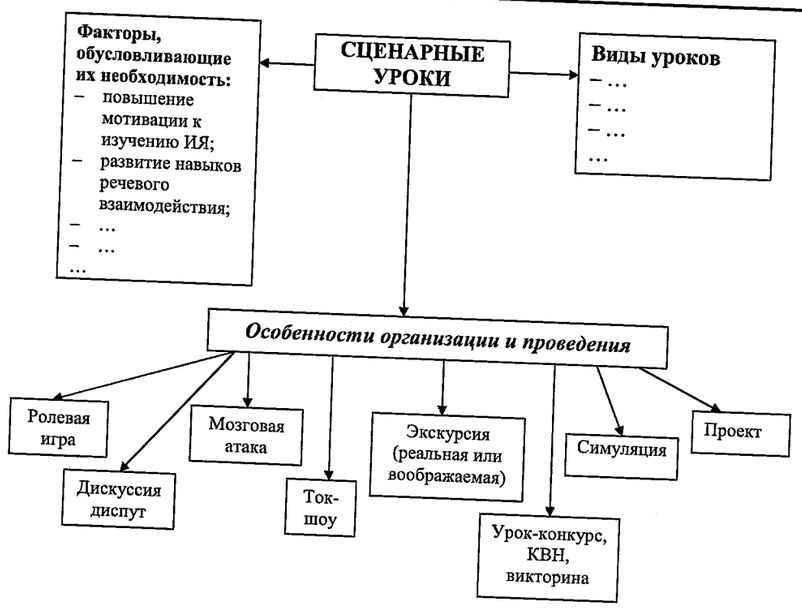
Процессы обновления в сфере обучения иностранным языкам в средней школе создают ситуацию, в которой педагогам предоставлены право и возможность самостоятельного выбора моделей построения процесса обучения, учебных пособий и других обучающих средств. В этой ситуации необходимо подходить к решению методических проблем с точки зрения активизации деятельности всех участников педагогического процесса. Этому способствуют и сценарные уроки иностранного языка, о чем и идет речь в статье.

Для обсуждения данной проблемы на практических занятиях по методике обучения иностранному языку (ИЯ) студентам предлагаются, например, такие вопросы:

1. Что такое сценарный урок?
2. Как выбрать вид сценарного урока, адекватный поставленной цели?
3. Как методически грамотно организовать и провести уроки данного вида?
4. Какова роль учителя в организации и проведении сценарных уроков?

Задача организовать общение школьников на уроке ИЯ часто представляется несложной и сводится к тому, чтобы дать возможность каждому ученику сформулировать свое высказывание и принять участие в общем разговоре. Однако решить эту задачу на практике оказывается не просто. Наблюдения за учебным процессом показывают, что на уроке ИЯ нередко произносятся заранее заданные последовательности реплик, т. е. каждый ученик знает свою «роль». В подобной «инсценировке» отсутствует важное свойство общения – речевое взаимодействие его участников.

ГУЛЯЕВА Валентина Семеновна – кандидат педагогических наук, доцент ВятГГУ
ИВАНОВ Сергей Юрьевич – студент V курса факультета информатики ВятГГУ



Речевое взаимодействие – это объединение, координация и взаимодополнение усилий участников общения для определения, приближения и достижения коммуникативной цели и результата речевыми средствами. Речевое взаимодействие является важным условием организации учебной работы школьников на уроке ИЯ, с помощью которого можно рационально использовать учебное время, активизировать речемыслительную деятельность учащихся, повышать развивающий эффект обучения.

Речевое взаимодействие не может быть представлено в виде готового и заранее заданного текста, поскольку любой текст есть итог монологической, диалогической или групповой речемыслительной деятельности. Ограниченность репродуктивного обучения как раз в том, что вместо речемыслительной *задачи* учащимся предъявляется *способ ее решения*. Для того чтобы организовать речевое взаимодействие учащихся, недостаточно предъявить им речевую задачу, образцы речевых действий. Нужны такие методические приемы, которые обеспечивали бы необходимое речевое взаимодействие учащихся на ИЯ [1]. Эти методические приемы и легли в основу деления сценарных уроков на различные виды. Студентам предлагается обобщить результаты изучения проблемы в виде опорного конспекта (см. рисунок).

Рамки статьи не позволяют дать подробные методические рекомендации по организации и проведению различных видов сценарных уроков. Остановимся на некоторых из них.

Суть приема ***«мозговая атака»*** (brain-storming) заключается в том, что учащимся необходимо найти оптимальное решение конкретной проблемы в течение ограниченного периода времени. С этой целью группа делится на три подгруппы: *генераторы идей, критики* и *эксперты*. Основные условия организации и проведения «мозговой атаки»:

1. *Генераторы идей* могут высказывать любую мысль без опасения, что она будет признана неудачной или нерациональной.
2. Любая инициатива поощряется, принимается любая идея, даже нелепая, на первый взгляд.
3. Важно стремиться к наибольшему количеству идей.
4. Важно изменять, комбинировать, улучшать предложенные идеи (как свои, так и чужие).
5. В случае, если творческая активность *генераторов идей* снижается, учитель может стимулировать работу подгруппы путем постановки вопросов, предложения собственных идей, новых подходов и др.
6. По истечении времени, заданного учителем для выдвижения идей, приступает к работе подгруппа критиков.
7. *Критики* анализируют, оценивают, синтезируют предложенные идеи и включают в список те, которые, на их взгляд, обеспечивают решение проблемы.
8. *Эксперты* изучают предложенные в списке идеи, классифицируют их, дают качественную оценку каждой и выбирают 3 наиболее оптимальные. (Они также аргументируют свой выбор для всей группы, защищая идеи от привычных аргументов («так мы еще не делали».)

В заключение группа может выбрать наиболее удачное решение проблемы из предложенных вариантов, т. е. мозговая атака – прием стимулирования творческой активности учащихся для продуктивного решения проблемы, который предусматривает создание условий, способствующих возникновению новых оригинальных идей в противовес привычным стереотипным формам принятия решений.

Симуляция (simulation) – речевая деятельность учащихся, осуществляемая в ситуациях, максимально приближенных к реальным, и стимулируемая коммуникативной задачей или проблемой, которые требуют личностного отношения обучаемых к фактам и событиям. Необходимо учитывать, что предоставленные участникам основные факты, имеющие отношение к данной ситуации (проблеме), не должны быть домыслены или изменены. Проигрывание полученных ролей происходит в соответствии с собственными взглядами, жизненным и языковым опытом.

Обобщение результатов обсуждения осуществляется в виде устного или письменного сообщения.

Первый шаг в каждой *дискуссии* – определить желаемый результат и понять точку зрения другой стороны. Для того чтобы управляемая дискуссия прошла успешно, к ней надо тщательно готовиться, учитывая при этом ее особенности и интересы участников. К технологии проведения дискуссии относятся:

- четкое определение позиций и реакций оппонентов;
- прогнозирование своего поведения и выбор вопросов;
- слушание и понимание иных позиций;
- обобщение, заключение.

Основные задачи учителя, ведущего дискуссию:

- создать атмосферу, в которой может развиваться свободная дискуссия;
- стимулировать высказывание идей, мнений;
- убедиться, что все имеют возможность высказываться;
- поддерживать дискуссию в нужном направлении;
- разъяснять запутанные вопросы;
- сделать заключение.

Ведущий дискуссию должен уважать чужие мнения, даже если он с ними не согласен, контролировать ее ход с помощью вопросов. Хороший вопрос характеризуется краткостью, ясностью формулировки, его легко понять, он связан только сутью дела, логически последователен. Наиболее полезны открытые вопросы, начинающиеся с вопросительных слов *почему*, *что* и *как*, на которые нельзя ответить просто *да* или *нет*. Как правило, полезнее ставить вопросы, на которые может ответить любой участник группы, чем задавать индивидуальные вопросы. Общие вопросы заставляют готовить коллективные ответы.

Индивидуальные вопросы полезны в том случае, если нужно включить отмалчивающегося в дискуссию. Кроме того, они способствуют поддержанию дискуссии, если она начинает угасать или надо узнать конкретный личный опыт. Иногда следует трансформировать общие вопросы в индивидуальные, т. е., поставив вопрос перед группой, ведущий делает паузу для обдумывания и адресует его конкретному лицу.

Ведущий должен побуждать участников дискуссии к обмену мнениями. Его задача – представить проблему, а затем следить за ходом ее развития.

Во время дискуссии важно периодически резюмировать промежуточные результаты, что позволяет акцентировать основное положение, убедиться в прогрессе, достигнутом при обсуждении проблемы, удерживать дискуссию на правильном пути, предоставлять участникам возможность сделать те замечания, которые они не успели сделать раньше, мотивировать обучаемых к дальнейшему прогрессу.

Особенно важно обращать внимание на тот момент, когда высказываются спорные или ошибочные мнения. Лучшей тактикой в случаях несогласия с определенным суждением является выдвижение вопроса для обсуждения. Если и несколько последующих высказываний приводят к тому же выводу, ведущий ни в коем случае не должен указывать всем, что они заблуждаются. Слабое место уже известно, и позднее можно предпринять меры по формированию правильного представления.

Если дискуссия отклоняется от темы и группа уходит от обсуждения основной проблемы, учителю следует указать, как и когда можно продолжить решение данного вопроса, и предложить вернуться к главной теме.

Очень важно в конце дискуссии сделать обобщение, сформулировать единые выводы, показать все интересные идеи группы.

Проведению дискуссии должно предшествовать овладение учащимися средствами речевого взаимодействия (выражение согласия/несогласия, уточнение, аргументирование, одобрение и др.).

Особую популярность как среди учащихся, так и среди преподавателей иностранного языка получила проектная технология.

Метод проектов возник в начале девятнадцатого столетия в США. Его основателями считаются американский философ и педагог Дж. Дьюи и его ученик В.Х. Килпатрик. Этот метод называли также методом проблем. Дж. Дьюи предлагал строить обучение на активной основе, через целесообразную деятельность ученика, сообразуясь с его личным интересом именно в этом знании. Отсюда чрезвычайно важно было показать детям их личную заинтересованность в приобретаемых знаниях, которые могут и должны пригодиться им в жизни. Для этого необходима проблема, взятая из реальной жизни, знакомая и значимая для ребенка, для решения которой ему необходимо приложить полученные знания, новые знания, которые еще предстоит приобрести [2].

В основе метода проектов лежит развитие познавательных навыков учащихся, умений самостоятельно конструировать свои знания, умений ориентироваться в информационном пространстве, развитие критического и творческого мышления. Поэтому, если мы говорим о методе проектов, то имеем в виду именно способ достижения дидактической цели через детальную разработку проблемы, которая должна завершиться вполне реальным, осязаемым практическим результатом, оформленным тем или иным образом. В основу метода проектов положена идея. Составляющая суть понятия «проект», его прагматическая направленность на результат, который можно получить при решении той или иной практически или теоретически значимой проблемы. Этот результат можно увидеть, осмыслить, применить в реальной практической деятельности. Чтобы добиться такого результата необходимо научить учащихся или студентов самостоятельно мыслить, находить и решать проблемы, привлекая для этой цели знания из разных областей, умения прогнозировать результаты и возможные последствия разных вариантов решения, умения устанавливать причинно-следственные связи [3].

Метод проектов всегда ориентирован на самостоятельную деятельность учащихся – индивидуальную, парную, групповую, которую учащиеся выполняют в течение определенного отрезка времени. Учитель может подсказать источники информации, а может просто направить мысль учеников в нужном направлении для самостоятельного поиска. Но в результате ученики должны самостоятельно и в совместных усилиях решить проблему, применив необходимые знания подчас из разных областей, получить реальный и ощутимый результат.

Основные требования к использованию метода проектов:

1. Наличие значимой в исследовательском, творческом плане проблемы/задачи, требующей интегративного знания, исследовательского поиска для ее решения. Например, исследование истории возникновения праздников в англоязычных странах: St. Patrick's Day, Thanksgiving Day, Halloween, Christmas, Mother's Day; организация путешествий в разные страны; проблемы семьи; проблема свободного времени у молодежи; проблема обустройства дома; проблема отношений между поколениями; проблема организации спортивных мероприятий.

2. Практическая, теоретическая значимость предполагаемых результатов. Например, доклад в соответствующие службы о демографическом состоянии данного региона, факторах, влияющих на это состояние; совместный выпуск газеты, альманаха с репортажами с места событий; программа туристического маршрута; план обустройства дома, парка, участка; планировка и обустройство квартиры.

3. Самостоятельная (индивидуальная, парная, групповая) деятельность учащихся на уроке или во внеурочное время.

4. Структурирование содержательной части проекта (с указанием поэтапных результатов и распределением ролей).

5. Использование исследовательских методов: определение проблемы, вытекающих из нее задач исследования; выдвижение гипотезы их решения; обсуждение конечных результатов; анализ полученных данных; подведение итогов, корректировка, выводы (использование в ходе совместного исследования метода «мозговой атаки», «круглого стола», творческих отчетов, защиты проекта и т. п.).

Теперь рассмотрим общие подходы к стимулированию проекта:

1. Выбор темы проекта, его типа, количества участников.

2. Учителю необходимо продумать варианты проблем, которые важно исследовать в рамках намеченной тематики. Сами же проблемы выдвигаются учащимися с подачи учителя (наводящие вопросы, ситуации, способствующие определению проблем, видеоряд с той же целью и т. д.). Здесь уместна «мозговая атака» с последующим коллективным обсуждением.

3. Распределение задач по группам, обсуждение возможных методов исследования, поиск информации, творческих решений.

4. Самостоятельная работа участников проекта по своим индивидуальным или групповым исследовательским, творческим задачам.

5. Промежуточные обсуждения полученных данных в группах (на уроках или на занятиях в научном обществе, в групповой работе в библиотеке, медиатеке).

6. Защита проектов, оппонирование.

7. Коллективное обсуждение, экспертиза, результаты внешней оценки, выводы.

Для топологии проектов предлагаются следующие топологические признаки:

1. Доминирующая в проекте деятельность: исследовательская, поисковая, творческая, ролевая, прикладная (практико-ориентированная), ознакомительно-ориентировочная. В соответствии с этим выделяют исследовательский проект, ролево-игровой, практико-ориентированный, творческий, информационный.

2. Предметно-содержательная область: монопроект (в рамках одной области знания); межпредметный проект.

3. Характер координации проекта: непосредственный (жесткий, гибкий), скрытый (неявный, имитирующий участника проекта, характерно для телекоммуникационных проектов). Соответственно, выделяют проекты с открытой (явной) и со скрытой координацией.

4. Характер контактов (среди участников одной школы, класса, города, региона, страны, разных стран мира). Здесь выделяют внктренние и международные проекты.

5. Количество участников проекта – личностные, среднесрочные и долгосрочные проекты.

Разумеется, в реальной практике чаще всего приходится иметь дело со смешанными типами проек-

тов, в которых имеются признаки исследовательских и творческих проектов. Каждый тип проекта имеет тот или иной вид координации, сроки исполнения, этапность, количество участников. Поэтому, разрабатывая проект, надо иметь в виду признаки и характерные особенности каждого из них.

Важную роль в преподавании иностранного языка занимают создание естественной языковой среды и формирование потребности в языковом общении. В этом плане особое место занимают международные проекты, которые позволяют решить эти задачи. Для организации международных проектов сегодня все шире применяются телекоммуникации, в частности широко используется Интернет в силу все большей доступности и эффективности. Этим обусловливается распространение телекоммуникационных международных проектов, которые создают условия для межкультурного общения. Подобного рода проекты помогают не только в изучении определенной темы, но и стимулируют изучение культуры страны, с которой проводится проект, способствуют овладению различными техническими средствами (Интернет).

Множество предложений Интернет-проектов можно найти через I*EARN, International Education and Resource Network – «Международное образование и ресурсы компьютерной среды» (<http://www.iearn.org>). Здесь предлагаются проекты со всего мира. Опыт использования данного информационного ресурса и проектов с него описан в статье Н. М. Коптюг [4]. Примером одного из таких проектов является проект под названием «Kindred» – «Родственники (Семья)», который был выбран Н. М. Коптюг для проведения в своем классе. Учащимся необходимо было написать небольшое сочинение на английском языке о своих родных, набрать его на компьютере и отослать координатору в Австралию. При реализации этого проекта возникали различные трудности: непривычной казалась сама идея проекта; почти никто из учеников не знал, как обращаться с компьютером и работать через Интернет; многие просто испугались. Одно дело – писать сочинение на уроке английского языка, другое дело – послать свою работу в страну, где английский язык является родным. В итоге были отосланы всего два сочинения. Оба были напечатаны в 1998 г. в Австралии. После того как учащиеся увидели книгу с работами их товарищей, у них появился интерес, они поняли, что не требуется ничего сверхъестественного и подобная работа вполне посильна.

Изложенное позволяет нам утверждать, что правильный выбор и методически грамотная организация сценарных уроков актуализирую собственно учебно-познавательную деятельность и учащегося как субъекта данной деятельности, реализуют его личностный потенциал. Это создает условия для развития креативности учащегося, обеспечивает по-настоящему продуктивный характер учебной деятельности и продуктивность образовательного процесса.

Примечания

1. *Мильруд Р. П.* Обучение школьников речевому взаимодействию на уроке иностранного языка // ИЯШ. 1991. № 6. С. 3-8; *Полат Е. С.* Метод проектов // Сайт Российской Академии образования, лаборатория дистанционного обучения: http://www.ioso.ru/distant/

2. *Полат Е. С.* Метод проектов на уроке иностранного языка // ИЯШ. 2000. № 3. С. 3; *Он же.* Там же. № 3. С. 3.

3. *Полат Е. С.* Указ. соч.

4. *Коптюг Н. М.* Интернет-проект как дополнительный источник мотивации учащихся // ИЯШ. 2003. № 3. С. 62.

Л. С. Гуржапова

ИСПОЛЬЗОВАНИЕ ИГРЫ НА УРОКАХ АНГЛИЙСКОГО ЯЗЫКА В МЛАДШИХ КЛАССАХ

Известно, что игра является одном из наиболее эффективных, гибких и универсальных приемов обучения. Она призвана активизировать процесс обучения, сделать его более продуктивным, а также формировать и развивать мотивацию учения. Автор статьи показывает, как детально продуманная и методически грамотно организованная игра позволяет комплексно решать задачи как практического, так и воспитательного, развивающего и образовательного характера на начальной ступени обучения английскому языку.

«Я буду изучать иностранный язык! Я смогу общаться с настоящими иностранцами!» – такие восторженные мысли обычно возникают у школьников, начинающих изучать иностранный язык. Но получается так, что английский, который казался вроде бы простым, вдруг на самом деле является очень трудным языком (Курт Тухольский).

И то радостное настроение, с которым они начинали посещать уроки иностранного языка, постепенно угасает, а иногда и вовсе пропадает. И часто мы слышим: «Я не могу запомнить слова. Мне трудно произнести».

Для нас, преподавателей, это огромная проблема. Настоящий урок – это не тот, что просто прошел, а тот, что запомнился детям как интересный, познавательный; урок, который оставил след в памяти учеников. Урок, на который хочется прийти еще раз.

Поэтому преподаватели постоянно ищут наилучшие способы взаимодействия с учащимися и преподавания материала.

Одним из таких приемов является игра.

Общение на любом языке требует большого словарного запаса, который накапливается в течение нескольких лет. Отсюда следует, что изучать язык важно начинать с раннего детства. Игра позволяет детям эффективно и качественно, а главное, с интересом изучать иностранные языки. Учитывая тот факт, что интерес является лучшим стимулом к обучению, необходимо стараться использовать каждую возможность, чтобы разгрузить ребёнка посредством игровой деятельности в процессе обучения языку.

ГУРЖАПОВА Любовь Сергеевна – студентка V курса факультета информатики ВятГГУ

Психологами и педагогами установлено, что прежде всего в игре развивается способность к воображению, образному мышлению.

Большое влияние оказывает игра на развитие у детей способности взаимодействовать с другими людьми. Кроме того, что ребёнок, воспроизводя в игре взаимодействие и взаимоотношения взрослых, осваивает правила, способы этого взаимодействия, в совместной игре со сверстниками он приобретает опыт взаимопонимания, учится пояснять свои действия и намерения, согласовывать их с другими детьми.

Игра очень эффективна при обучении английскому языку детей младшего школьного возраста.

В этом возрасте происходят большие изменения в познавательной сфере ребенка [1]. Память приобретает ярко выраженный познавательный характер. В этот период развиваются формы мышления, обеспечивающие в дальнейшем усвоение системы научных знаний, развитие научного, теоретического мышления [2].

Во-вторых, в младшем школьном возрасте идет интенсивное формирование приемов запоминания.

В области восприятия происходит переход от непроизвольного восприятия к целенаправленному произвольному наблюдению за объектом.

В этом возрасте формируется способность сосредотачивать внимание на малоинтересных вещах. Эмоциональные переживания приобретают более обобщающий характер.

Таким образом, младший школьный возраст – это возраст интенсивного развития [3].

В то же время иностранный язык – сложный предмет, требующий от учащихся напряженной умственной деятельности. Специфика иностранного языка такова, что он, характеризуясь чертами, присущими вообще языку как знаковой системе, также определяется целым рядом отличительных от родного языка особенностей овладения и владения им.

Специфической особенностью иностранного языка является сформированное к нему негативное, субъективное отношение учащихся как к очень трудному, практически не поддающемуся в условиях школьного обучения овладению предмету.

Иностранный язык, действительно, требует работы – ежедневной, систематической, мотивированной. Ученик должен иметь четко поставленную конкретную цель изучения иностранного языка.

Вот почему игра – наилучший способ обучения детей. Она оказывает огромное влияние на развитие и становление личности. Многие психологи считают, что в игре происходят главнейшие изменения в психике ребенка. Игра формирует качества, нужные для успешного обучения ребенка иностранному языку.

С помощью игры хорошо отрабатывается произношение, активизируется лексический и грамматический материал, развиваются навыки аудирования, устной речи.

В игре развиваются творческие, мыслительные способности ребенка. В ней предполагается принятие решения: как поступить, что сказать, как выиграть.

Обучающие игры помогают сделать процесс обучения иностранному языку интересным и увлекательным. Чувство равенства, атмосфера увлеченности дают возможность ребятам преодолеть стеснительность, скованность, снять языковой барьер, усталость.

В любой вид деятельности на уроке можно внести элементы игры, и тогда даже самое скучное занятие приобретает увлекательную форму.

Д. Б. Эльконин [4] считает, что игра выполняет четыре важнейшие для человека функции:

- средство развития мотивационно-потребностной сферы;
- средство познания;
- средство развития умственных действий;
- средство развития произвольного поведения.

Игра всегда предполагает принятие решения – как поступить, что сказать, как выиграть. Это обостряет мыслительную деятельность учащихся. Именно в игре дети усваивают общественные функции, нормы поведения. Игра, как говорил Л. С. Выготский [5], ведет за собой развитие. Развивающее значение игры заложено в самой ее природе, ибо игра – это всегда эмоции, а там, где эмоции, – там активность, там внимание и воображение, там работает мышление.

Таким образом, игра – это

1) деятельность;
2) мотивированность;
3) индивидуализированная деятельность, личная;
4) обучение и воспитание в коллективе и через коллектив;
5) развитие психических функций и способностей;
6) «учение с увлечением» (С. Л. Соловейчик) [6].

Применять игру как средство обучения можно при прохождении любой темы, грамматической структуры. Таким образом, и классификация игр может быть разная.

В данной классификации главным принципом послужили определенные уровни, или аспекты языка:

1) игры с буквами;
2) игры со словами;
3) игры с предметами;
4) грамматические игры;
5) поэтические игры;
6) загадки, ребусы.

Эта классификация может быть расширена, так как на уровне слов, предложений, текстов могут выполняться и лексические, и грамматические, и стилистические, и орфографические, и фонетические игры.

Применяя языковые и речевые игры на уроке, учитель должен помнить следующее.

1. Выбор формы игры должен быть педагогически и дидактически обоснован. Учитель должен знать, с какой целью он проводит ту или иную игру.

2. В играх должно быть задействовано как можно больше учащихся.

3. Игры должны соответствовать возрастным и языковым возможностям учащихся.

4. Языковые игры служат развитию всех видов речевой деятельности. На первом плане – говорение. Есть игры, предполагающие письменное выполнение, а также ряд заданий, которые могут по усмотрению учителя выполняться устно или письменно.

5. Не всегда удаётся «вписать» языковую игру в тематику урока. Но учитель должен стараться это сделать.

6. Ведущий, обычно учитель, должен быть по возможности на заднем плане. Роль ведущего могут играть и ученики.

7. Затраты времени на подготовку и проведение игры и её польза должны находиться в оправданном соотношении друг с другом.

8. Среди многих видов упражнений есть такие, которые являются предметом педагогических дискуссий: корректировочные упражнения. Упражнения, включающие поиск ошибок, полезны тем, что они дарят радость открытия и стимулируют активность учащихся.

9. Языковые игры должны проводиться преимущественно на иностранном языке.

Поскольку целью обучения иностранному языку является формирование навыков и развитие умений, можно классифицировать игры и игровые упражнения в соответствии с этой целью обучения [7].

I. Языковые упражнения, направленные на формирование навыков и развитие умений:

1. Игры для обучения произношению.

2. Игры для обучения слушанию и говорению.

II. Игры, направленные на усвоение определенной лексики.

Куда интереснее вместо обычного заучивания слов по теме, например, на уроке наряжать елку разноцветными шарами, садить бумажные цветы, искать сбежавших из зоопарка зверей, кормить голодного Робина Бобина или считать веснушки у Незнайки.

Когда я проходила практику в Вятской гуманитарной гимназии, меня попросили провести уроки английского языка в 7-х классах, так как преподаватель заболел. За день до уроков мне сообщили тему урока, дали учебник, рабочую тетрадь и добрые напутствия. Весь оставшийся вечер я придумывала, как занять ребят, привлечь их внимание и преподнести материал. Ведь это был мой первый опыт преподавания английского языка.

На помощь пришла игра. В течение нескольких уроков мы играли с ребятами, изучая тему «Животные». Результат превзошел все наши ожидания.

Ребята с удовольствием рассказывали о своих домашних любимцах, изображали различных зверюшек, загадывали друг другу загадки, участвовали в «Поле чудес», «Угадайке».

По собственному желанию школьники выучили названия других животных и приготовили интересные доклады.

Достаточно лишь внести элемент игры или создать игровую ситуацию, чтобы ребенок сам захотел овладеть изучаемой темой, структурой, запомнить нужные правила.

Игра позволяет сделать учебный процесс привлекательным и интересным, заставляет учащихся волноваться и переживать.

Это мощный стимул к овладению языком.

Примечания

1. Возрастная и педагогическая психология / Под ред. А. В. Петровского. М.: Просвещение, 1973.

2. *Зимняя И. А.* Психология обучения иностранным языкам в школе. М.: Просвещение, 1991.

3. *Обухова Л. Ф.* Возрастная психология. М.: Роспедагентство, 1996; *Петрунек П., Таран Л.* Младший школьный возраст. М.: Знание, 1981.

4. *Эльконин Д. Б.* Психология игры. М.: Педагогика, 1978.

5. *Верхогляд В. А.* Игры на английском языке для внеклассной работы // Иностранные языки в школе. 1981. № 5.

6. *Савченко О. Ю.* Игры на уроках английского языка // Иностранные языки в школе. 1992. № 2.

7. *Колесникова И. Е.* Игры на уроках английского языка в 5-м классе. М., Народная Асвета, 1990.

А. А. Зубарев

ПРИМЕНЕНИЕ ВИДЕОМАТЕРИАЛОВ НА УРОКАХ АНГЛИЙСКОГО ЯЗЫКА

В статье описаны упражнения для работы с видеофильмами на занятиях по английскому языку в средней школе и показана их роль в развитии коммуникативной компетенции учащихся.

В достаточной степени овладеть иностранным языком, не находясь в стране изучаемого языка, как известно, очень трудно. Поэтому важной задачей учителя является создание реальной и воображаемой ситуаций общения на уроке, также не менее важно приобщение школьников к культурным ценностям народа – носителя языка.

Использование видеофильмов помогает ученику усваивать язык более осмысленно и с большим интересом. Один и тот же языковой материал усваивается при помощи видеофильмов не только более прочно,

ЗУБАРЕВ Алексей Анатольевич – студент V курса факультета информатики ВятГГУ

но и при меньшем количестве затраченного времени.

При использовании видеофильмов совершенствуется психическая деятельность учащихся. Во время просмотра в классе возникает атмосфера познавательной совместной деятельности. В этих условиях даже невнимательный ученик становится внимательным. Для того чтобы понять содержание фильма, школьникам необходимо приложить определенные усилия. Так, непроизвольное внимание переходит в произвольное, его интенсивность оказывает влияние на процесс запоминания. Использование различных каналов поступления информации (слуховое, зрительное, моторное восприятие) положительно влияет на прочность запечатления языкового материала.

Наряду с выдачей учебной информации фильм включает ученика в воспроизводимое событие, способствует более полному запоминанию учебного материала, а также соединяет все особенности учебного кино, телевидения и радио. 4-5-минутная трансляция предоставляет достаточно материала для обсуждения в течение всего урока; сохраняет нормальную рабочую обстановку, не нарушающую психологический контакт между учителем и учениками.

Давайте рассмотрим методику применения видеоматериалов в 8-м классе на примере работы с короткометражным фильмом «В кафе самообслуживания».

Целью применения художественного фильма являлось развитие навыков говорения, увеличение словарного запаса, активизация лексики, повышение интереса к изучению иностранного языка, развитие внимания, памяти и др.

На предыдущем уроке ученикам было задано домашнее задание: по портрету описать внешность, характер, краткую биографию двух человек и придумать им имена (были предложены портреты двух действующих лиц фильма).

Продолжительность фильма около 9 минут, но в целях наилучшего восприятия он был разделен на два эпизода (4 и 5 минут), работа над которыми проходила по отдельности.

(Для наилучшего усвоения лексики и развития речи у учащихся мы бы порекомендовали провести два урока по этой теме.)

Главной особенностью фильма является то, что в нем отсутствует звук. Отсутствие звукового сопровождения помогает ученикам осмыслить ситуацию в целом, не отвлекаясь на попытки понять незнакомые слова.

1. Допросмотровый этап

Проверяется домашнее задание. Ученики в группах рассказывают друг другу биографии, которые они придумали по портретам.

После этого учитель сообщает о том, что на уроке учащиеся будут смотреть фильм, в котором участвуют эти персонажи. Чтобы потом обсуждать фильм, им нужно запомнить некоторые новые слова и фразы. Слова и фразы могут быть записаны на карточках или на доске (в них необходимо включить названия различных блюд, овощей и фруктов).

Pea soup – гороховый суп
Condiment – приправа
Sausage – сосиска
Black man – негр
Coat – пальто
Beer – пиво
Thick soup – суп-пюре
Suddenly – внезапно
Nevertheless – тем не менее
Loss – пропажа
Café – кафе
Smile – улыбаться

Желательно разъяснить ученикам разницу между словами «Black man» и «Negro».

В зависимости от класса или от своих предпочтений учитель может дать другие слова. Главное, чтобы ученики могли рассказать о том, что они думают.

2. Просмотровый этап

Затем ученикам предлагаются вопросы для активизации внимания, на которые они должны найти ответы во время просмотра фильма (вопросы написаны на доске или на карточках).

1. What did the Old Lady have for dinner?
2. What was The Black man doing when the Old Lady saw him?
3. What did The Old Lady take from the Black man?
4. What did the Black man gave her?
5. Why did The Old Lady call up to police?

Демонстрируется первый эпизод видеофильма. Затем проводится беседа по вопросам, которые были заданы (проверка понимания просмотренного отрывка). Для закрепления материала ученики выполняют задание «Правильно – неправильно» (индивидуально). На доску выписаны следующие фразы:

1. The Old Lady saw The Métis on her place.
2. The Old Lady had only tea for dinner.
3. The Old Lady saw that The Black man eaten her dinner.
4. The Old Lady took off her coat before eating.
5. The Old Lady smiled to see The Black man.

После выполнения упражнения проводится взаимоконтроль, затем правильные варианты ответов ученики зачитывают вслух по цепочке.

На дом предлагается задание написать возможный вариант развития событий.

Следующий урок начинается с проверки нескольких работ. Учитель с учениками вместе обсуждают различные варианты развития событий.

Ученики повторяют лексику с прошлого урока, выполняют упражнение на повторение материала (в парах). Нужно расположить предложения в правильной последовательности:

(2) The Old Lady gave money.
(5) The Old Lady returned.
(1) The Old Lady took dinner.
(6) The Old Lady saw The Black man.

(4) The Old Lady took off her coat.

(3) The Old Lady put her handbag.

Учитель просит учеников зачитать предложения в правильной последовательности вслух по цепочке.

Далее предлагается посмотреть второй эпизод.

3. Послепросмотровый этап

Чтобы проверить понимание сюжета, ученикам необходимо ответить на вопросы, которые были поставлены перед ними непосредственно перед просмотром фильма.

Проводят беседу, в которой обсуждают данную ситуацию и возможные пути развития данного сюжета. Беседа может быть фронтальной или в парах (развитие речи каждого ученика). Можно дать ученикам задание рассказать, как бы они поступили на месте того или иного героя.

Можно порекомендовать исправить характеры героев и их биографии.

Можно инсценировать данный фильм с продолжением, а можно посмотреть его еще раз, причем ученики будут озвучивать действия героев.

В качестве домашнего задания можно порекомендовать пересказ фильма в различных вариантах: от первого лица старушки, от лица негра, от третьего лица (кассира или посетительницы).

Примечания

1. *Андриевская В.В.* Психология усвоения иностранного языка на среднем этапе обучения в школе // Иностранные языки в школе. 1985. № 6. С. 3-8.

2. *Барменкова О.И.* Видеозанятия в системе обучения иностранной речи // Иностранные языки в школе. 1999. № 3. С. 20-25.

3. *Барменкова О.И.* Использование кинофрагментов на уроках английского языка // Иностранные языки в школе. 1987. № 5. С. 89-90.

4. *Гуляева В.С.* Использование видеофильмов для развития коммуникативной компетенции студентов // Совершенствование преподавателя иностранных языков в школе и в вузе. Киров, 1998. С. 52-57.

5. *Кашкуревич Л.Г.* Формирование различных видов речевой деятельности с помощью кинофильмов // Иностранные языки в школе. 1984. № 1. С. 57-58.

Е. В. Кипрская

ПСИХОЛИНГВИСТИЧЕСКОЕ ИССЛЕДОВАНИЕ АГЛОЯЗЫЧНЫХ ПОЛИТИЧЕСКИХ ЭВФЕМИЗМОВ

Статья посвящена описанию процедуры психолингвистического эксперимента, который проводился со студентами IV курса факультета информатики. Экспериментальным материалом послужили аглоязычные политические эвфемизмы, отобранные из словарей.

КИПРСКАЯ Екатерина Викторовна – аспирант кафедры германских языков ВятГГУ

В настоящее время политический дискурс является объектом пристального внимания не только в лингвистике, но и в таких дисциплинах, как философия, психология, культурология и т. д. Возрастает интерес к данному феномену и в современном обществе, поскольку решение многих политических и социальных проблем зависит от их правильной интерпретации. Сложность интерпретации политического дискурса, как нам представляется, объясняется его «специфической эзотеричностью», обусловленной тем, что обладание тайной (эксклюзивное право на информацию) является одной из сил, составляющих власть [1]. Хью Роусон [2] приводит точку зрения известного английского писателя Джорджа Оруэлла, который в своем эссе «Политика и английский язык» отмечает, что язык политики сконструирован таким образом, чтобы ложь звучала более правдоподобно, а убийство выглядело вполне благопристойно, а также чтобы придать весомость пустой болтовне. Эвфемизмы, как указывает Х. Роусон, всегда несут в себе скрытый смысл, но все предпочитают считать, что данный скрытый смысл не существует. Такое положение вещей может оказаться необходимым, когда идет речь о стремлении избежать или вовсе закамуфлировать некоторые события и явления политической и социальной жизни или, как говорит автор, «использовать язык, чтобы направить мысли людей в политически верное русло» [3]. Близкой точки зрения придерживается Е.И. Шейгал, которая считает, что эвфемистическое переименование является следствием сознательного воздействия на язык как проявление его магической функции [4]. По ее мнению, «расшифровка эвфемистического переименования позволяет выявить систему ценностей общества» и «в эвфемизме по сравнению с исходным, "табуированным" наименованием снимается коннотация социальной неприемлемости» [5].

Ранее нами предпринимался анализ публикаций, посвященных эвфемизмам в таких лингвистических дисциплинах, как семантика, стилистика и лексикология [6]. Но, как указывает И. А. Стернин [7], уровень традиционного лингвистического описания предполагает обобщенное описание значений и употреблений языковых единиц и структур в отвлечении от психологии говорящего человека, а фиксация значений в словарях является результатом обобщения и отвлечения от лингвистической реальности. Поэтому «описание слова в словаре есть результат отвлечения от реальных употреблений слов, определенный лингвистический конструкт, который нельзя абсолютизировать» [8]. Проблема разграничения конструкта как продукта научного описания возможных схем репрезентации действительности у человека и концепта как спонтанно функционирующего в речемыслительной деятельности индивида базового перцептивно-когнитивно-аффективного образования динамического характера ставится и последовательно решается А. А. Залевской [9].

Поскольку теоретическим основанием нашего исследования являются психолингвистическая теория слова и концепция специфики функционирования индивидуального знания, разработанные А. А. Залевской [10], мы планируем осуществить исследование политических эвфемизмов через обращение к индивидуальному сознанию человека. Еще раз отметим, что данный лексический пласт изучался, в основном, на лингвистическом уровне описания языка.

В основе предстоящего эксперимента лежит рабочая гипотеза о том, что трактовка современными носителями языка политических эвфемизмов, несущих в себе глубокий (скрытый) смысл и являющихся одним из способов камуфлирования действительности, не совпадает с их словарными дефинициями.

Цель данного эксперимента – установить, в какой степени политические эвфемизмы знакомы современным носителям русского языка, причисляют ли они их к исследуемой категории, и сравнить специфику реального значения эвфемизмов в сознании человека с данными, представленными в словарях. В соответствии с целью эксперимента были поставлены следующие практические задачи:

1) установить корпус эвфемизмов для предъявления их испытуемым в качестве экспериментального материала, а также выбрать процедуру эксперимента, которая бы максимально соответствовала его цели;

2) определить, какие эвфемистические слова и словосочетания известны современным носителям языка;

3) выяснить, которые из них, по мнению испытуемых, принадлежат к категории эвфемизмов;

4) осуществить качественный и количественный анализ экспериментальных данных для того, чтобы выяснить, какой смысл современные носители языка вкладывают в эвфемизмы;

5) сопоставить результаты эксперимента с дефинициями эвфемизмов, представленными в словарях, чтобы найти совпадения и различия между пониманием данных лексических единиц «наивным» носителем языка и их словарной трактовкой;

6) установить факторы, влияющие на идентификацию эвфемизмов современными носителями языка.

При выборе методов мы руководствовались спецификой материала и задачами предстоящего экспериментального исследования. В качестве основного был выбран метод психолингвистического эксперимента с использованием комплексной методики, включающий свободный ассоциативный эксперимент и метод субъективных дефиниций. В качестве дополнительного метода планируется использование смыслового анализа экспериментального материала и словарных статей к исследуемым эвфемизмам. Отбор корпуса эвфемистических слов и словосочетаний осуществлялся методом сплошной выборки из словарей. Материалом для предстоящего экспериментального исследования послужили 25 английских эвфемистических слов и словосочетаний. За основу при отборе был взят словарь Х. Роусона «Rawson's dictionary of euphemisms and other doubletalk» [11]. Представляется уместным привести точку зрения Уильяма Сафира [12], который утверждает, что за последние 15-20 лет в связи с окончанием «холодной войны», развалом Советского Союза и сменой нескольких администраций в Белом Доме, политический дискурс претерпел серьезные изменения. Появилось много новой лексики, которая берет начало из Библии: «all things to all men», «fight the good fight», «Armageddon»; из поэзии: «strange bedfellow» Шекспира, «wasteland» Томаса Элиота; из военного дискурса: «campaign», «boom», «regular», «escalation», «purify» и т. д. Интересно отметить, что, по мнению автора [13], к языку политики не следует причислять те слова, у которых есть «официальная» дефиниция, приведенная в словарях и не несущая в себе скрытого смысла. У. Сафир дает свое определение политическим эвфемизмам: «Это стремление избежать неприятных, тяжелых слов» [14]. Примером самого циничного эвфемизма в 1970-х годах, по мнению У. Сафира, может служить «protective reaction» («бомбардировка»), термин из военного дискурса, означающий бомбардировку наземных средств противовоздушной обороны. Мы посчитали необходимым принять во внимание точку зрения составителей словарей [15], и при отборе материала руководствовались данными дефинициями политических эвфемизмов. Ниже приводится список слов и словосочетаний, которые предполагается предъявить испытуемым. Мы считаем, что для последующей интерпретации экспериментальных данных полезно также привести этимологические данные некоторых исследуемых эвфемизмов (далее – Э).

1. «Antipersonnel weapon» («оружие массового уничтожения»). Это словосочетание впервые вошло в употребление во время Второй мировой войны.

2. «Assets» («оружие»).

3. «Bite the dust» («пасть в бою»). Данный Э появился в английском языке благодаря переводу «Илиады» Гомера Ричмондом Латтимором в 1951 году. Возможны следующие варианты: «bite the ground», «bite the sand» и более современный, принадлежащий скорее сленгу, «kiss the ground».

4. «Collateral damage» («случайные жертвы среди мирного населения при бомбежке военных объектов»). Впервые это словосочетание появилось в печати 12 июня 1979 года в «New York Times». Получило широкое распространение во время войны в Персидском заливе в 1991 году.

5. «Collective indiscipline» («отказ солдат выполнять приказы командования»). Термин известен с Первой мировой войны.

6. «Contribution» («налоги»). Данный Э был введен в употребление президентом США Биллом Клинтоном.

7. «Economical with the truth»(«уклоняющийся от истины», «сокрытие нелицеприятных фактов»).

8. «Escalate» («эскалация»). Стало очень популярным словом во времена войны во Вьетнаме.

9. «Ethnic cleansing» («этническая чистка»). Примером использования могут служить события в Сербии, Боснии и Герцеговине.

10. «Fundamental truth» («полуправда», «ложь»). Появилось в печати в 1978 году в интервью, данном президентом США Ричардом Никсоном Дэвиду Фросту. Возможные варианты: «essential truth» и «symbolic truth».

11. «Inoperative» («неверный, неправильный»). Вошло в употребление после знаменитого Уотергейтского скандала.

12. «Liberate/liberation» («разрушить», «уничтожить»). Данный Э впервые появился во время Второй мировой войны.

13. «Liquidate/liquidation» («ликвидировать», «ликвидация»). Интересно отметить, что данные Э впервые вошли в употребление в России после революции, а затем с помощью конверсии закрепились в английском языке.

14. «Military government» («правление хунты»).

15. «Minority» («этнические меньшинства»). Традиционный Э для Афро – Американцев и выходцев из Латинской Америки. Возможные варианты: «black» и «people of color».

16. «Misadventure» («ошибка»). Данный Э использовался американским военным командованием во время военных действий во Вьетнаме в случае гибели американских солдат от собственной артиллерии.

17. «Misspeak» («оговориться»). Стал особенно популярным во времена правления президента Никсона.

18. «Misstatement» («ложь»). Данный Э также появился на свет во времена Уотергейтского скандала.

19. «Neutralize/neutralization» («нейтрализовать, нейтрализация»).

20. «Precision bombing» («бомбовые удары, направленные на определенные цели»). Возможный вариант «discriminate deterrence», который использовался во время войны в Персидском заливе в 1991 году.

21. «Protective reaction» («бомбардировка»). Приведенное словосочетание было впервые использовано в 1969 году, чтобы оправдать возобновленные американскими военными бомбардировки в Северном Вьетнаме.

22. «Purify» («зачистка»).

23. «Strategic» («стратегический»). Примерами словосочетаний с данным Э могут служить следующие: «strategic bombing», которое во время Второй мировой войны означало бомбежку городов и которое позднее трансформировалось в идею «strategic nuclear war».

24. «Sweeping operation» («операция по зачистке»). Варианты: «search and clear», «search and sweep», «sweep and clear», «sweeping operation», «reconnaissance forces sweeping operation».

25. «White lie» («ложь»).

За время от 40 до 50 минут испытуемым будет предложено выполнить следующие задания:

1) указать, знакомо или нет данное слово или словосочетание;

2) отметить, причисляют ли испытуемые данные высказывания к эвфемизмам;

3) дать перевод без помощи контекста;

4) дать перевод в контексте;

5) записать возникшие в связи со стимулом ассоциации.

Эвфемистические слова и словосочетания, а также задания к ним будут напечатаны на специальных бланках, которые предстоит заполнить. Перед началом эксперимента будет дано подробное объяснение каждого задания.

Примечания

1. *Шейгал Е. И.* Рефлексивы в политической коммуникации // Аспекты метакоммуникативной деятельности. http://tpl1999.narod.ru.

2. *Rawson H.* Rawson's dictionary of euphemisms and other doubletalk. N. Y., 1995.

3. Ibid. P. 3.

4. *Шейгал Е. И.* Семиотика политического дискурса: Автореф. дис. ... д-ра филол. наук. Волгоград, 2000.

5. Там же. С. 21.

6. *Кипрская Е. В.* Указ. соч.

7. *Стернин И. А.* Уровни описания языкового сознания // Филология и культура. Тамбов, 2003. С.11-14.

8. Там же. С. 12.

9. *Залевская А. А.* Введение в психолингвистику. М., 2000; *Залевская А. А.* Концепт как достояние индивида // Психолингвистические исследования слова и текста. Тверь, 2000.

10. *Залевская А. А.* Введение...

11. *Rawson H.* Cit. op.

12. *Safire W.* Safire's new political dictionary. N. Y., 1995.

13. Ibid.

14. Ibid. P. 226.

15. *Rawson H.* Cit. op.; *Safire W.* Cit. op.

Е. Н. Колодкина, Е. М. Лобанова

ПСИХОЛИНГВИСТИЧЕСКОЕ ИССЛЕДОВАНИЕ ГЕНДЕРНЫХ АСПЕКТОВ ЯЗЫКА

В статье высказывается предположение, что гендерный компонент является неотъемлемой частью структуры слова. В ходе анализа результатов психолингвистического эксперимента были выделены основные модели понимания существительных с ярко выраженным гендерным компонентом.

Гендерология – новая, бурно развивающаяся отрасль гуманитарного знания. Ученые изучают психологические основы гендерологии, её влияние на человеческую культуру. Гендером начинают интересоваться и лингвисты, так как в языке отражаются все психологические и культурные процессы, при этом отображение человека в языке происходит двояко, поскольку «человек представлен в двух ипостасях – мужчина и женщина» [1].Основная задача гендерной лингвистики заключается в изучении мужского и женского языка, а также особенностей языкового поведения мужчины и женщины. Как указывает Е. И. Горошко, при изучении проблемы взаимоотношения языка и гендера можно выделить три подхода [2].

1. Первый подход сводится к трактовке исключительно социальной природы языка женщин и мужчин. В рамках этого подхода выявляются семантические различия мужского и женского языка, которые объясняются перераспределением социальной власти в обществе.

2. В рамках второго подхода «мужской» и «женский» языки рассматриваются как особенности языкового поведения представителей обоих полов. Среди методов исследования данного подхода выделяются определение статистических показателей, средних параметров.

3. Третий подход делает упор на когнитивном аспекте различий мужского и женского речевых рисунков.

Представляется, что второй и третий подходы к проблеме взаимодействия гендера и языка, выделенные Е. И. Горошко, можно объединить в единый подход, поскольку оперирование статистическими данными отнюдь не исключает объяснение нестандартных, нетипичных явлений. Более того, когнитивная научная парадигма обладает большой объяснительной силой при изучении специфики ментального лексикона. В фокусе внимания современных исследований лексикона находится концепт, представляющий собой «спонтанно функционирующее в речемыслительной деятельности индивида базовое перцептивно-когнитивно-аффективное образование динамического характера, отличающееся от понятий и значений по ряду параметров» [3].

Принято считать, что концепт представляет собой совокупность разнообразных вербальных и невербальных компонентов. Рассматривая структуру концепта, Е. В. Лукашевич выделяет следующие компоненты: понятие, представление, эмоция и оценка, предметное содержание и ассоциативные связи [4]. Нам представляется, что указанные компоненты не исчерпывают структуру концепта. Предложенная А. А. Залевской концепция внутреннего лексикона человека исходит из множественности и разнообразия принципов его организации: основаниями для связи между его единицами могут служить признаки и признаки признаков по любому из направлений переработки разнообразной информации [5]. Можно предположить, что гендер следует рассматривать в качестве одной из составляющих как структуры концепта, так и структуры значения слова.

Объектом нашего психолингвистического исследования послужили английские слова с ярко-выраженным гендерным компонентом в значении. Целью экспериментального исследования было выявление специфики образов языкового сознания испытуемых, изучающих английский язык, и через полученные ассоциативные поля выделение основных стратегий понимания данных слов-стимулов. Мы предположили, что гендерный компонент найдет свое проявление как в ассоциативных полях, так и в моделях идентификации исследуемых слов.

В экспериментальный список вошли слова, отражающие возрастные характеристики мужчин и женщин (BOY (здесь и далее прописными буквами обозначаем экспериментальный стимул, строчными – слово-реакцию), GIRL, TEENAGER, MAN, WOMAN), семейный статус (FATHER, GRANDFATHER, GRANDMOTHER, MOTHER), профессиональную деятельность (HEADMASTER, HEADMISTRESS), имеющие стилистическую окраску (GENTLEMAN, LADY, MAIDEN).

Испытуемым предлагалось записать первые 3 пришедших в голову ассоциации к предложенным словам. Слова-стимулы предъявлялись испытуемым в письменном виде, а инструкция зачитывалась устно. Теоретически в результате проведённого эксперимента мы могли бы получить 1323 реакции на слова-стимулы. Однако на некоторые слова со стороны испытуемых были приведены одна или две реакции вместо трех, отказов же от реакций на предлагаемые слова не было. Таким образом, в ходе ассоциативного эксперимента было получено 586 реакций.

Наибольшую трудность у испытуемых вызвали такие слова-стимулы, как HEADMISTRESS (25); HEADMASTER, GENTLEMAN (27); MAIDEN (30); LADY (35). Можно предположить, что сложность ассоциирования связана с тем, что слова-стимулы неизвестны испытуемым (HEADMISTRESS, HEADMASTER, MAIDEN), а также с тем, что некоторые слова не свойственны для русской культуры. Примером таких слов могут служить слова GENTLEMAN и LADY, чисто английские реалии.

Что же касается слов, к которым все испытуемые написали по три реакции, то таких слов только два: MOTHER и TEENAGER. Причинами этого может быть то, что мать является самым близким для любого человека, а подростками испытуемые сами были совсем недавно.

В ходе структурного анализа мы проанализировали ассоциативные поля исследуемых слов-стимулов, среди слов реакций были выделены отдельные слова (134) и словосочетания (18). Среди словосочетаний можно выделить 4 модели:

1) adjective + noun: bad marks, English woman, harmful habits, old age, small machines;

2) noun + noun: baby's dummy, business lady, ladies' saint, lady Diana, snot nose;

КОЛОДКИНА Елена Николаевна – кандидат филологических наук, доцент, заведующая кафедрой германских языков факультета информатики ВятГГУ
ЛОБАНОВА Екатерина Михайловна – студентка V курса факультета информатики ВятГГУ

3) noun + preposition + noun: care of the child, example for an imitation, head of the family, prosperity of the family;

4) verbal + noun: swaddling clothes.

Большинство реакций-словосочетаний было получены на слова-стимулы BOY, FATHER, MOTHER (по 3 реакции-словосочетания). Также следует отметить, что большинство реакций-словосочетаний относится к детям (baby's dummy, bad marks, care of the child, small machines, snot nose, swaddling clothes).

Группа реакций-слов намного больше, в нее вошли 134 (103 различных) слова. Причем среди них 100 (75 различных) существительных, 31 (26) прилагательных и 3 (2) глагольные формы.

Если рассматривать полученные реакции-прилагательные по способу образования, то мы можем найти все три вида прилагательных: простые (kind, small, strong), производные (beautiful, trustful) и сложные (well-bred). Все производные прилагательные образованы путём прибавления к основе слов (в основном существительных) суффикса. Количество простых прилагательных – 18, производных – 6, составных – 2.

Большинство прилагательных имеют нейтральную коннотацию (adult, strong, young) или позитивную (elegant, kind, polite) коннотацию и только 3 прилагательных (fidgety, selfish, wild) имеют негативную окраску.

Далее рассмотрим реакции-существительные, которые чаще всего встречаются в реакциях испытуемых. Следует отметить, что среди 75 различных существительных можно выделить абстрактные (beauty, fidelity, youth) и конкретные (armchair, cane, spectacles). Количество абстрактных существительных – 25, а конкретных – 50.

Если классифицировать существительные по способу образования, то можно выделить 54 простых существительных (age, beer, skirt), 11 производных (kindness, motherhood, player), 9 составных (housewife, pancake, schoolgirl) и 1 сокращение (TV).

Рассматривая имена существительные с позиции их стилистической окраски, можно выделить 58 существительных с нейтральной (bag, car, milk), 9 – с положительной (firewood, kindness, wisdom) и 8 – с негативной коннотацией (alcohol, drag, impudence).

Мы подробно рассмотрели ассоциативные поля каждого слова-стимула. Приведем примеры анализа. Так, к слову GIRL было приведено 11 ассоциаций: child, school, bows, dolls, childhood, cosmetics, small, beautiful, trustful, schoolgirl, skirt. У всех слов позитивная или нейтральная коннотация. Идентифицируя слово GIRL, акцент делали прежде всего на юный возраст, что подтверждают такие ассоциации, как bows, child, childhood, dolls, small. Также обращали внимание на школьный статус (school, schoolgirl), внутренние качества (trustful), внешние данные и одежду (beautiful, skirt).

Слово BOY ассоциируется с shorts, fidgety, scuffles, child, school, small machines, bad marks, small, snot nose, wild. Отметим, что испытуемые оказались более критичны к мальчикам. Так, если при оценке девочек все слова имели позитивную или нейтральную коннотацию, то среди реакций к слову-стимулу BOY есть слова с негативной окраской. Это такие реакции, как bad marks, fidgety, scuffles, wild. Так же, как и в предыдущем случае, делается акцент на возраст (child, school, small, small machines, snot nose). Что же касается одежды мальчиков, то она ассоциируется, прежде всего, с shorts.

Таким образом, при сопоставлении пары слов-стимулов GIRL – BOY мы видим, что отношение к девочкам у испытуемых более положительное, чем к мальчикам. Кроме того, мы можем предположить, что слова GIRL и BOY ассоциируются прежде всего с детьми младшего школьного возраста.

Следующая пара слов-стимулов TEENAGER – MAIDEN характеризует более старшую возрастную группу. Анализ экспериментальных данных показывает, что характеристика возрастного периода от тринадцати до девятнадцати лет имеет ярко выраженную негативную окраску. Данный возрастной период характеризуется словами disco, drug, player, jeans, young, impudence, daredevil, alcohol, harmful habits, music, и половина из перечисленных слов имеет негативную коннотацию (alcohol, daredevil, drug, harmful habits, impudence). По данным эксперимента, основное увлечение подростков – музыка (disco, music, player).

Главным этапом нашего анализа было выделение основных моделей идентификации слов полученного ассоциатиного поля, посредством которых осуществляется соотнесение слова стимула с тем или иным фрагментом ментального лексикона человека, и, в конечном итоге, понимание слова. Мы выделили 8 основных стратегий идентификации.

1. Идентификации через указание на возраст: BOY – child; GIRL – childhood; MAN – adult; GRANDMOTHER – old age.

2. Идентификации через роли, занимаемые человеком в семье: FATHER – head of the family; WOMAN – housewife; WOMAN – wife.

3. Идентификации через основные занятия: BOY – school; GIRL – schoolgirl; FATHER – work; GRANDFATHER – firewood; GRANDMOTHER – knitting.

4. Идентификации через одежду: BOY – *shorts;* GENTLEMAN – *tie;* LADY – jacket; MAN – suit.

5. Идентификации через конкретный предмет, характерный для данного слова-стимула: BOY – small machines; FATHER – car; GENTLEMAN – flowers; GIRL – dolls; GRANDFATHER – cane; GRANDMOTHER – spectacles; TEENAGER – player; WOMAN – bag.

6. Идентификации через внешние данные: GENTLEMAN – tidy; LADY – elegant; MAIDEN – beauty; MAN – muscular.

7. Идентификации через внутренние качества: BOY – fidgety; GENTLEMAN – polite; GRANDFATHER – grumbler; HEADMASTER – strictness; HEADMISTRESS – responsibility; MAN – courage.

8. Идентификации через эмоциональную оценку: BOY – snot nose; BOY – wild; TEENAGER – daredevil.

Необходимы дальнейшие исследования гендерного компонента в структуре значения слова. Представляется целесообразным определить его роль, место и специфику, а также особенности его взаимодействия с другими компонентами структуры значения.

Примечания

1. *Маслова В. А.* Лингвокультурология. М.: Academia, 2001. С. 121.

2. *Горошко Е.И.* Пол, гендер, язык // Женщина, гендер, культура. М., 1999. С. 109.

3. Там же. С. 16.

4. *Лукашевич Е.В.* Когнитивная семантика: эволюционно-прогностический аспект. М.; Барнаул: Изд-во Алт. ун-та, 2002. 234 с.

5. *Залевская А.А.* Концепт как достояние индивида // Психолингвистические исследования слова и текста. Тверь, 2002. С. 5-18.

Е. Н. Колодкина, О. С. Рублева

ОСОБЕННОСТИ УПОРЯДОЧЕНИЯ СЛОВ ПО ПАРАМЕТРАМ КОНКРЕТНОСТИ И ОБРАЗНОСТИ В ИДИОЛЕКСИКОНЕ

В статье высказывается предположение, что одним из принципов организации идиолексикона человека является упорядочение по степени выраженности параметров конкретности и образности в структуре значения слова. Предпринят количественный и качественный анализ английских существительных с различными показателями коэффициентов конкретности и образности.

Данное исследование выполнено в рамках параметрического подхода к значению слова, рассматривающего значение как совокупность компонентов, степень выраженности которых поддается количественному измерению. Впервые параметрический подход как один из основных подходов к значению внутри когнитивной парадигмы выделяет А. А. Залевская [1]. В обзоре [2] приводятся отечественные и зарубежные исследования, сделанные в рамках параметрического подхода к значению. Там же сделан вывод о том, что параметры конкретности (далее К) и образности (далее О) – это разные, но взаимосвязанные психолингвистические феномены, взаимодействие которых во внутреннем лексиконе еще предстоит изучить.

Исследование внутреннего лексикона относится к числу актуальных проблем психолингвистики и когнитивной науки в целом. Как указывает Н. О. Золотова [3], динамика подходов к трактовке ментального лексикона индивида соотносится с общими тенденциями развития психолингвистики и других наук о человеке. В центре внимания ученых обычно оказывается ряд вопросов, связанных с тем, что понимается под значением слова как единицы ментального лексикона, какие именно функции оно выполняет в когнитивных процессах, как организован ментальный лексикон, каким образом осуществляется доступ к его единицам.

В работе [4], написанной еще в 1977 году, исходя из психофизиологических особенностей речевой организации человека (Л. В. Щерба [5]), даются определения внутреннего лексикона и впервые последовательно излагаются принципы организации лексикона. Дальнейшее развитие этих идей с позиций психолингвистики послужило основанием для трактовки ментального лексикона как функциональной динамической системы, самоорганизующейся вследствие постоянного взаимодействия между процессами и продуктами переработки и упорядочения разностороннего (перцептивного, когнитивного, аффективного; вербального и не поддающегося вербализации) опыта человека в окружающем его мире [6]. Наиболее эксплицитно принципы речевой организации индивида изложены А. А. Залевской в [7] в системе из 20 оппозиций, характеризующих язык как достояние индивида и язык как абстрактный конструкт соответственно.

Можно предположить, что одним из принципов организации лексикона является его упорядочение по степени выраженности параметров К и О.

Под К в психолингвистических исследованиях понимается способность слова быть измеренным при помощи конкретных физических процедур. О трактуется как способность слова вызывать в сознании образ слова. Если слово обозначает предмет, который представляется носителям языка как материальный, то есть может быть легко измерен, взвешен, ощупан и т. д., то это слово получит максимальный коэффициент по параметру К. И наоборот, если слово обозначает абстрактное понятие, которое не выражено материально, то оно получит низкие показатели по шкале К. Если слово легко и быстро вызывает яркий мыслительный образ, то оно получит высокие баллы по шкале О. И напротив, если слово трудно или не-

КОЛОДКИНА Елена Николаевна – кандидат филологических наук, доцент, заведующая кафедрой германских языков факультета информатики ВятГГУ
РУБЛЕВА Ольга Сергеевна – ассистент кафедры германских языков ВятГГУ

возможно представить в образной форме, то оно получит низкие показатели по шкале О.

В психологических и психолингвистических исследованиях, проведенных по методике субъективного шкалирования применяют градуальную шкалу. В исследовании А. Пайвио [8] использовалась семибалльная градуальная шкала с делениями от одного до семи, где цифре один соответствует минимальное значение исследуемого параметра, а цифре семь соответственно – максимальное значение параметра.

В семантических нормах А. Пайвио приводятся коэффициенты К и О для 925 английских существительных. Экспериментальные данные А. Пайвио предназначались для верификации его гипотезы о двойном – вербальном и образном – кодировании информации. Количественные данные, полученные А. Пайвио, представляют обширный фактический материал для семантических исследований различного рода. В данной статье мы хотим рассмотреть специфику взаимосвязи параметров К и О в структуре значения английского слова. Укажем, что всем исследуемым словам были приписаны те или иные величины К и О, в том числе и словам, которые традиционно в лингвистике трактуются как абстрактные и не обладающие образностью.

В ходе количественного анализа коэффициентов, представленных в семантических нормах, мы выделили группы слов с коэффициентами в один шаг семибалльной шкалы. Мы назвали выделенные группы К1, К2, К3, К4, К5, К6 и О1, О2, О3, О4, О5, О6 соответственно. Количество слов, выделенных в каждой группе, представлено на рисунке.

В К1 вошло 79 слов со значением К от 1 до 2 балла. Эта группа представлена исключительно абстрактными словами – *hope, unreality, moral, idea, chance, mood, truth, memory, intellect, nonsense, theory, criterion, freedom, soul*. Среди абстрактных слов возможно особо выделить группу названий чувств и эмоций – *jealousy, ignorance, honor, greed, shame, anger, passion, joy, kindness, anxiety, hatred, pride*.

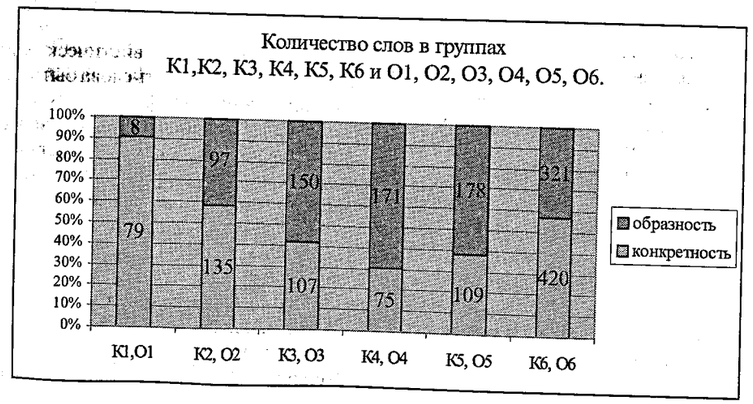
К2 (135 слов) представлена, в основном, абстрактными словами – *illusion, intimate, advice, impulse, quality, charm, interest, effort, method, advantage, miracle, trouble, economy, duty, perception, satire, madness, capacity, fun, time, chaos, moment*, среди которых можно выделить группу названий чувств и эмоций – *pleasure, affection, gaiety, hostility, humor, misery, irony* и группу названий вымышленных существ – *devil, demon*.

В К3 (107 слов) вошли существительные, обозначающие названия лиц, – *creator, prayer*; некоторые юридические термины – *crime, homicide, brutality, evidence*; абстрактные слова, обозначающие категории состояния и отношения качества, – *cost, health, hearing, silence, agony, poverty*.

В К4 (75 слов) можно выделить слова, обозначающие названия лиц, – *unbeliever, maker, arbiter, peacemaker, lord*; абстрактные понятия – *disaster, fatigue, unit, permission, answer, lecture, code, robbery, meeting, anecdote, speech*.

В К5 (109 слов) можно выделить слова, характеризующие названия лиц, – *comforter, disturber, outsider, dreamer, charlatan, originator, cuisine, comrade, baron, dweller, leader, prosecutor, retailer, owner*; абстрактные слова – *gift, volume, angle, salary*; названия лиц, дифференцированные по иерархическому признаку, – *master, chief*.

В К6 (420 слов) представлены слова, среди которых можно выделить названия конкретных лиц, дифференцированные по степени родства, – *child, baby, grandmother, wife, mother, nephew;* по титулу и рангу – *sultan, doorman, monarch, slave, queen, admiral, king, officer, prisoner, professor*; возрасту – *girl, student, pupil*; названия профессий – *builder, artist, singer, acrobat, student, teacher, butcher, fisherman, musician, pianist, painter, physician, doctor, pupil, policeman, blacksmith*; названия природных явлений – *typhoon, storm, hurricane, volcano*; названия природных объектов – *ocean, river, lake,* soil; названия артефактов – *book, fork, bagpipe, steamer, clock, paper, ship, tablespoon,*



magazine, picture, palace, journal, toy, arrow, automobile, car, chair, door, kettle, shoes, refrigerator, table, tree, umbrella, pencil, window; названия животных и растений – cat, fox, leopard, potato, apple, oats.

Качественный анализ слов по параметру О показал, что в группу О1 вошли восемь абстрактных слов, в основном научные термины – *criterion, inanity, impropriety, concept, gist, interim, functionary*.

В группу слов О2 вошли 97 слов. Среди них преобладают абстрактные слова – *truth, idea, advantage, chance, moment, situation, ability, unreality, soul*, среди которых можно выделить группу научных терминов – *method, theory, deduction*; и группу слов – названий лиц – *arbiter, unbeliever, reminder*.

О3 представлена 150 словами, среди которых наиболее представительна группа абстрактных слов, – *opportunity, mind, nonsense, advice, duty, opinion, effort, miracle, promotion, malady, custom, illusion, trouble*; слова, обозначающие чувства и эмоции, – *justice, obedience, boredom, hatred, honor, greed*; слова – названия лиц – *creator, originator, charlatan, maker*.

В группе слов О4 171 существительное. Среди них слова, обозначающие названия лиц, – *peacemaker, retailer, dweller, author, lord, master, disturber, owner, nephew*; абстрактные слова – *code, chaos, dream, salary, demon, charm, time*; названия чувств и эмоций – *kindness, sadness, jealousy, pleasure, brutality, pride, misery, humor*.

В группе О5 178 слов, среди которых представлены слова, обозначающие названия лиц по профессиональному признаку, – *teacher, painter, builder, artist, lecturer*; слова, характеризующие названия лиц, – *comforter, prosecutor, racketeer, leader, prayer, poet, dreamer, performer, comrade, speaker, bandit;* юридические термины – *robbery, victim, murder, injury, thief, bandit*; названия чувств и эмоций – *fatigue, happiness, exhaust, passion, love, gaiety*;

В О6 – 321 слов, среди которых выделена группа конкретных слов – *iron, kettle, stone, elbow, dove, arrow, apple, bottle, lip, butter, potato, umbrella, chair, car*; группа слов, обозначающих названия артефактов, – *furniture, pencil, tablespoon, book, clock, fork*; группа слов, обозначающих природные объекты, – *forest, grass, river, ocean, tree*; группа слов, дифференцированных по степени родства, – *wife, boy, girl, mother, infant, child, grandmother*; группа слов, обозначающих названия животных, – *fox, frog, leopard, snake, butterfly*; группа слов – названий лиц, дифференцированных по профессиональному признаку, – *blacksmith, singer, butcher, student, doctor, doorman, pianist, fisherman*; названия лиц, дифференцированные по иерархическому признаку, – *monarch, king, admiral, officer*.

Качественный анализ групп слов с различной выраженностью параметров К и О показал, что слова, относящиеся к одной тематической группе, встречаются в нескольких смежных группах, разделяемых величиной шага в один балл. Например, слова, обозначающие чувства и эмоции, можно встретить в группах О3, О4 и О5, однако наполнение групп будет различным, если в О3 – 5 слов, в О4 – 8 слов, в О5 – 6 слов. Таким образом, можно говорить о полевой структуре слов, относящейся к тематической группе с четко выделяющимся ядром и ближней и дальней периферией.

Безусловно, распределение слов по выделенным группам носит субъективный характер. Более того, само выделение тематических групп является весьма условным. Нам представляется, что континуальный характер упорядоченности параметров К и О во внутреннем лексиконе объясняется чрезвычайной сложностью и многомерностью как самого лексикона, так и окружающего нас мира, который специфически отражается во внутреннем лексиконе.

Проведенный анализ и наши выводы относительно упорядоченности внутреннего лексикона по К и О особенно ярко подтверждают некоторые из 20 выделенных А.А. Залевской принципов организации лексикона [9], а именно: субъективность, континуальность, постоянную динамику, предметность, обладание характеристиками открытой сверхбольшой динамической системы, многообразие оснований для различных связей и наличие расплывчатых множеств с нечеткими границами.

Примечания

1. *Залевская А. А.* Введение в психолингвистику. М.: Российск. гос. гуманит. ун-т, 2000. 382 с.

2. *Колодкина Е. Н.* Экспериментальное параметрическое исследование конкретности и образности в структуре значения слова // Вестник Вятского государственного гуманитарного университета. Информатика, Киров, 2002. С. 68-69.

3. *Золотова Н. О.* Ментальный лексикон, его ядро и функции единиц как проблема языкового сознания // Филология и культура: М-лы IV Междунар. науч. конф. 16-18 апр. 2003 года. Тамбов: Изд-во ТГУ, 2003. С. 25-27.

4. *Залевская А. А.* Различные подходы к трактовке языка // Слово и текст: психолингвистический подход. Вып. 1. Тверь, 2003. С. 48-55.

5. *Щерба Л. В.* О трояком аспекте языковых явлений и об эксперименте в языкознании // Языковая система и речевая деятельность. Л., 1974. С. 23-39.

6. *Залевская А. А.* Текст и его понимание. Тверь: Изд-во Твер. ун-та, 2001. 177 с.

7. *Залевская А. А.* Различные подходы...

8. *Paivio A., Yuille J.C., Madigan S.A.* Concreteness, imagery and meaningfulness values for 935 nouns // Journal of Experimental Psychology. Monograph Supplement, 1968. Vol. 76. No. 1. Part 2. P. 1-25.

9. *Залевская А. А.* Различные подходы...

Л. П. Кудреватых

ДВУСТОРОННИЕ СОСТАВЛЯЮЩИЕ СЛОВА. ИСТОРИЯ ВОПРОСА

Рассматриваются существующие классификации значений. Учет комплекса критериев позволяет выполнить дифференцирование описания двусторонних лексико-семантических составляющих слова (словозначений, лексико-семантических вариантов, синкретов и прономинализаций). Семантические типы отношений между выделяемыми единицами отражают своеобразие семантических типов слов в языке.

Семиологический подход к анализу слова как знака предполагает признание единства внешней формы (звучания) и внутреннего содержания (значения). В связи с этим в семантической структуре слова выделяются двусторонние лексико-семантические составляющие слова.

В современной лингвистике для обозначения лексико-семантических составляющих значения слова используются разные термины: словозначения (М. В. Никитин), лексико-семантические варианты (А. А. Уфимцева, Е. Г. Беляевская), семемы (Л. А. Новиков), отдельные значения слова (Д. Н. Шмелев). Однако отмечается недифференцированное употребление этих терминов, часто их синонимическое использование, иногда – лишь интуитивное разграничение.

Терминологическое упорядочение в этой сфере было проведено В. П. Конецкой, которой удалось вывести системные характеристики двусторонних лексико-семантических составляющих слова и доказать наличие различных типов этих единиц: словозначения, лексико-семантические варианты, синкреты, словоморфемы (в нашей редакции «прономинализации»).

Впервые понятие двусторонней лексико-семантической единицы было определено А. И. Смирницким, который обозначил наличие «лексико-семантического варианта» (ЛСВ) в ряду с фономорфологическими единицами. Под ЛСВ понимается двусторонняя языковая единица, являющаяся единством звучания и значения и сохраняющая неизменность лексического значения в пределах присущих ему синтагматических связей и парадигм. Таким образом, уже по определению предлагается различать «значение слова», репрезентирующее внутреннюю сторону семантики слова, и ЛСВ как двустороннюю единицу, представляющую единство внешней формы и внутреннего содержания. По мнению В. П. Конецкой, в последующих редакциях трудов А. И. Смирницкого произошло терминологическое смешение и понятиями «значение» и «ЛСВ» стали пользоваться синонимически.

КУДРЕВАТЫХ Людмила Петровна – доктор филологических наук, профессор ВятГГУ

Очевидно, что лексико-семантические составляющие слова – словозначения, ЛСВ, синкреты и прономинализации – имеют ярко выраженные различия. Однако основная проблема здесь заключается в отсутствии объективных критериев их разграничения в структуре слова (В. В. Левицкий, И. А. Стернин) и в признаках формализации понятий семантического сходства и различия значений слова (Ю. Д. Апресян).

В российском языкознании проблема формализации признаков сходства и различия семантического содержания слова разрабатывалась в связи с обнаружением типов значений слова. В построении семантической типологии языковых значений выделяется несколько подходов.

Одна из первых классификаций значений слова была проведена В. В. Виноградовым с учетом лексических, грамматических, функциональных и деривационных параметров слова. Здесь различаются основные и производные значения, первичные и вторичные значения, прямые и переносные, синтаксически свободные и несвободные значения и т. д.

Учет семантических и функциональных характеристик значения проводится при функциональном подходе к анализу природы слова. В этом направлении наиболее типичной является классификация Л. М. Васильева, который предлагает различать значения знаков-десигнаторов и знаков-формантов, предметные и признаковые значения, номинативные и указательные значения.

Когнитивный подход, положенный в основу классификации М. В. Никитина, сделал возможным выявить иерархию логических отношений части и целого, родо-видовых отношений в слове. Здесь предполагается, что типы значений обусловливают и их языковой статус, который определяется соотносительными характеристиками ЛСВ как единицы номинации. В целом же семантическая структура слова определяется в этой работе как схема содержательных связей между словозначениями в совокупности их статутных характеристик. По мнению автора, основными концептуальными связями словозначений в слове являются импликационный и классификационный типы. Выделяются разновидности импликационной связи – конверсивная связь и классификационной связи – гиперо-гипонимическая и симилятивная связи.

В более поздних редакциях М. В. Никитин уточняет предложенную классификацию значений и предлагает учитывать характер информационной связи, степень субъективности и объективности оценки мира и степень зависимости от объема понятия. В связи с этим предлагается различать импликационный и знаковый типы значений. В свою очередь в когнитивном типе различают экстенсиональное и контенсиональное значения, последнее подразделяется на денотативное и сигнификативное значения.

В теории регулярной многозначности различают метафорическое и метонимическое значения, конверсивное значение, соотносящие типы значения с типом их связей в структуре слова.

Известны и более частные типологии значений, учитывающие формальные признаки значений: 1) по соотносительной близости или удаленности от основного значения различают значения контрастирующие с основным или близкие к основному (Н. Д. Гарипова), 2) по признаку возможности или невозможности сочетания разных разрядов слов различают слова со значениями неограниченно широкой семантики, широкой семантики, ограниченной семантики, узкой и единичной семантики (И. М. Деева).

Многообразие классификаций значений слова отражает многоплановость, подвижность, семантическую неоднородность и многофункциональность слова. Однако, вопрос о семантическом единстве слова не вызывает сомнения. Различают следующие параметры, обеспечивающие единство слова: семантический стержень, общее понятие, семантический центр, общее значение лексемы (Е. Г. Беляевская).

В настоящее время распространено восприятие структуры слова как семантического поля, имеющего центр, периферию и варьирующую часть, позволяющие объединить значения в семантическую структуру слова. При этом основное значение трактуется как некоторый инвариант или так называемое «обобщенное» значение слова (Э. В. Кузнецова), которое понимается как некоторое объективное содержание, учитывающее информационный потенциал слова (И. В. Сентенберг).

В современных лингвистических исследованиях признается, что семантическое единство слова обеспечивается семантическим типом отношений между двусторонними лексико-семантическими составляющими слова. Схема этих отношений признается универсальной и представлена в моделях регулярной полисемии.

Иерархическое соподчинение значений в семантической структуре слова выводится в рамках регулярной многозначности, получившей наиболее детальное описание в трудах Ю.Д. Апресяна. Признается, что целостность слова обусловлена тремя аспектами: парадигматическим, синтагматическим и эпидигматическим. Парадигматические связи отражают системные характеристики слова. Синтагматические связи определяют семантические валентности слова. Эпидигматические связи отражают своеобразие формальной и смысловой соотнесенности слова и его деривационных отношений (Д. Н. Шмелев).

Дифференцированное описание двусторонних лексико-семантических составляющих слова требует учета комплекса критериев: 1) своеобразие семантической деривации, 2) различия в собственных или несобственных категориях разряда, 3) наличие или отсутствие семантического сдвига значений, 4) автосемантизм или синсемантизм единицы (В. П. Конецкая).

Учет перечисленных критериев позволил обнаружить системные характеристики рассматриваемых единиц. Так, для словозначений характерны наличие семантической деривации, отличия в собственных катериях разряда, автосемантизм. Связи деривации представлены в формулах регулярной многозначности. Наиболее типичной для словозначений является их категориальная нетождественность. Так, в существительном *«галерка»* 1) место в театре и 2) люди, находящиеся на этих местах, оба словозначения различаются в собственных категориях разряда «одушевленность-неодушевленность», «лицо-нелицо». Им свойствен импликационный перенос. В прилагательном *«сухой»* 1) обезвоженный, 2) лаконичный (о стиле), 3) недоброжелательный (о человеке) отмечаются различия в собственных категориях разряда и семантической симиляционной неоднородности. Прямое значение прилагательного относится к разряду качественных, а переносные значения – к разряду качественно-оценочных: «приятный – неприятный».

Лексико-семантическим вариантам свойственна семантическая непрерывность, представляющая собой развитие некоторого обобщенного инвариантного признака, различия в несобственных катериях разряда, отсутствие актуальной деривации, синсемантизм. Так, своеобразие связей ЛСВ в структуре прилагательного «мягкий»: мягкий – не яркий (о свете), мягкий – не острый (о вкусе), мягкий – не сильный (о дожде), мягкий – теплый (о климате)» проявляется в отсутствии семантического сдвига значений и наличии варьирования основного признака, обозначаемого прилагательным «мягкий – не резкий в проявлениях». ЛСВ не образуют иерархической пирамиды соподчиненных единиц и являются совокупностью равноправных вариантов.

Синкреты представляют собой единицы, обнаруживающие совмещение семантических признаков категориального, интегрального или дифференциального уровней, предполагающих их совместное присутствие в структуре единицы и актуализацию одного из признаков с одновременным погашением другого признака: *«добираться»* – ехать, плыть, лететь на чем-либо; *«вахта»* – служебная обязанность и процесс ее реализации.

Прономинализациям свойственно отсутствие прямых связей деривации, различия в категориальных и интегральных признаках, в собственных категориях разряда и семантическом содержании, выполнение несобственных функций разряда. В связи с этим прономинализационные характеристики свойственны глаголам-связкам типа *«быть, становиться, превращаться»* и пр., ср. *«быть внимательным, стать серьезным»*, а также существительным, которые могут употребляться с целью обобщенной характеристики одушевленных лиц: *«мокрая курица, известная штучка»*.

Таким образом, двусторонние лексико-семантические составляющие слова репрезентируют один из аспектов семиотики: словозначения отражают отношения, регулируемые семантическим аспектом, ЛСВ и синкреты – аспекты синтактики, а прономинализа-

ции – аспект прагматики. Семантические типы отношений между выделенными единицами отражают своеобразие семантических типов слов в языке.

Примечания

1. *Апресян Ю. Д.* Лексическая семантика. Синонимические средства языка. М., 1974. 367 с.
2. *Беляевская Е. Г.* Семантика слова. М., 1987. 128 с.
3. *Виноградов В. В.* Избранные труды. М., 1953.
4. *Васильев Л. М.* Современная лингвистическая семантика. М., 1990. 176 с.
5. *Гарипова Н. Д.* Наблюдения над семантической структурой многозначных слов разных частей речи // Исследования по семантике. Уфа, 1976. С. 90-98.
6. *Деева И. М.* Полисемия и моносемия в сфере английских прилагательных. Л., 1988. 68 с.
7. *Конецкая В. П.* Введение в сопоставительную лексикологию германских языков. М., 1993. 201 с.
8. *Кузнецова Э. В.* Лексикология русского языка. М., 1989. 216 с.
9. *Левицкий В. В., Стернин И. А.* Экспериментальные методы в семасиологии. Воронеж, 1989. 192 с.
10. *Никитин М. В.* Основы лингвистической теории значения. М., 1988.
11. *Новиков Л. А.* Семантика русского языка. М., 1982. 272 с.
12. *Сентенберг И. В.* Динамический аспект лексической семантики английского глагола. Волгоград, 1991. 378 с.
13. *Смирницкий А. И.* Лексикология английского языка. М., 1956.
14. *Уфимцева А. А.* Лексическое значение. Принцип семиологического описания лексики. М., 1986. 239 с.
15. *Шмелев Д. Н.* Современный русский язык. Лексика. М., 1977. 335 с.

Л. П. Кудреватых

РОЛЬ КОНТЕКСТА В АКТУАЛИЗАЦИИ СЕМАНТИКИ СЛОВОЗНАЧЕНИЙ

В статье описаны 4 типа категорий, определяющих семантику слова. Категориально-семантическая разрядность слова определяет контекстуальные средства его семантической детерминации. Лексико-семантические, грамматические и синтаксические средства контекста обусловливают категориальное и семантическое своеобразие единиц.

В определении семантического содержания слова общеизвестны функции актуализации, интенсификации и дифференциации контекста. Актуальным является решение вопроса об определении семантического своеобразия двусторонних единиц, образующих семантическую структуру слова. Известно 4 типа двусторонних лексико-семантических составляющих слова: *словозначение, лексико-семантический вариант, синкрет, прономинализация.*

Объектом нашего внимания были словозначения, типологическими чертами которых являются признаки семантической деривации, наличия семантического сдвига в слове, актуализации собственных категорий разряда, автосемантизм единицы (В. П. Конецкая). Следует учитывать также категориально-семантическую специфику единиц разных частеречных разрядов. Так, в семантике слова следует различать 4 типа категорий:

1) концептуальные категории, отражающие наиболее обобщенные частеречные признаки: *субстанциональность* (у существительного) и *признаковость* (у глагола и прилагательного);

2) логические категории, отражающие внеязыковую действительность – физический мир и мир ментальный – в сознании людей, что проявляется в категориях *пространства* и *времени, конкретности и абстрактности*, своеобразно сопрягающихся в словах разных частей речи;

3) когнитивные категории, отражающие результаты обобщающей аналитической деятельности человека и закрепленные в языке в виде категориальных признаков дифференцирующих частеречный разряд слов: *предметность – непредметность* для существительного, *действие, процесс, состояние, события* для глагола; *качественность, количественность, оценка* для прилагательного;

4) онтологические категории, отражающие лексико-грамматические характеристики двусторонних составляющих слова, специфически представленные в структуре разных частей речи: *одушевленность – неодушевленность, исчисляемость – неисчисляемость* для существительного; категориальные признаки, обусловленные адвербиальной матрицей для глагола – *субъектность, объектность, адвербиальность (аспектуальность, залоговость, темпоральность, локативность* и др.); категориальный признак *компаративности* для прилагательного (Л. П. Кудреватых).

Материал показал, что категориальное своеобразие слова обусловливает его (слова) категориально-семантическую разрядность: отнесенность к категориально-семантическому разряду, парадигме или группе. Так, различия словозначений в логических категориях определяют их отнесенность к разным категориально-семантическим разрядам: *market* – 1) *базар, рынок*, 2) *экономическая ситуация*. Различия словозначений в когнитивных категориях обусловливают отнесенность их к разным категориально-семантическим парадигмам: *birch* – 1) *розга*, 2) *наказание розгами*. Различия словозначений в онтологических категориях обусловливают их отнесенность к различным категориально-семантическим группам: *gut* – 1) *кишка*, 2) *мужество* (неисч., мн. ч.).

По нашим наблюдениям, категориально-семантическая разрядность слова или словозначения определяет контекстуальные средства их семантической детерминации.

КУДРЕВАТЫХ Людмила Петровна – доктор филологических наук, профессор ВятГГУ

Материал позволил дифференцировать работу контекста, сочетаемости слова и проявления семантических валентностей слов разных частей речи. Соответственно, можно выделить три уровня контекста:

1) контекст, дифференцирующий категориально-семантическую разрядность словозначения;

2) контекст, дифференцирующий семантическое своеобразие словозначения;

3) контекст, дифференцирующий коннотативные особенности словозначения.

В структуре именных единиц первый тип контекста употребляется чаще всего для актуализации категориальных признаков одушевленности/ неодушевленности, которые отражают различия словозначений в категориально-семантических группах. Это валидно для существительных, тождественных в категориально-семантических разрядах и парадигмах, а также для словозначений, которые тождественны в их категориально-семантических признаках всех уровней.

В том случае, если именные словозначения относятся к разным категориально-семантическим группам, семантическая валентность отражает эти различия. Валентности разных семантически и синтаксически зависимых частей речи коррелируют с признаками одушевленности/ неодушевленности и обозначаются, например,

1) глаголами речи, умственной, социальной и профессиональной речи: *The gallery applauded the celebrities;*

2) личными местоимениями: *He's a rare bird;*

3) прилагательными оценочными, обозначающими характеристики сферы человека: *There were two smashing birds there.*

Категориальный признак неодушевленности может актуализируется в синтагме существительных, употребленных в функции обстоятельства места или образа действия: *He saw the play from the gallery.*

Именные словозначения, относящиеся к сходным категориально-семантическим группам, но различным категориально-семантическим парадигмам и разрядам, получают конкретизацию в контесте путем актуализации категориальных признаков более высокого уровня – признаков *«предметность/ непредметность»*. Так, уточнение семантики существительного *birch «палка, сделанная из березы»* и *«наказание такой палкой»* проходит средствами антонимического категориального противопоставления существительному *stroke (12 strokes of birch),* обозначающему явления непредметной сферы. Признак непредметности может проявляться средствами экстралингвитического контекста, когда предметное прочтение словозначения исключается.

Дифференциация признаков *«лицо/ нелицо»* проходит средствами контекста, уточняющими отнесенность предмета к миру живых и неживых объектов: *All birds singing in the trees…*

По нашим данным, для словозначений, которые различаются на всех трех уровнях, т. е. относятся к разным категориально-семантическим парадигмам, разрядам и группам, наибольшую актуальность имеют средства контекста, акцентирующие признаки *«одушевленность/ неодушевленность»*, что еще раз подтверждает вывод о важности вышеупомянутого признака для семантики существительного.

В дифференциации глагольной семантики важной является их категориально-семантическая разрядность: отнесенность к субъектным, субъектно-объектным или субъектно-объектно-адвербиальным единицам (А. В. Бондарко). В этом отношении существенной является синтаксическая зона словозначения, его левые и правые валентности. Так, адвербиальные распространители, имеющие разнообразные синтаксические трансформации, отражают особенности словозначений в признаках адвербиала: *The depths to which newspapers will stoop to get a story never fails to astound me.*

По нашим наблюдениям, возможности подобной синтаксической зависимости являются первой ступенью на пути к категориально-семантической дифференции. Более тонкие семантические различия актуализируются синонимическими и антонимическими средствами контекста.

По мнению Ю. Д. Апресяна, важным является не простое знание способности глагола, например, к грамматическому подчинению слов определенного частеречного разряда, но и его семантические валентности. Так, анализ глагола *«арендовать»* позволил выделить 5 его валентностей: субъект, первый объект, контрагент, второй объект, сроки. Факультативны для него валентности типа *причина, место, время, цель аренды.*

Подобная факультативная валентность дифференцирует семантические признаки интегрального уровня, обусловливая семантическую дифференциацию значения, что дает возможность более четкого отражения описания логико-предметного содержания глагольного словозначения. Так, адвербиальные зависимости обстоятельства образа действия в категориально-семантических признаках «направление» способствуют категориально-семантической дифференциации словозначения глагола *carry: 1) He picked up his suitcase and carried it into the bedroom* (VO), *2) A gentle current carried him slowly down stream. The ship was carried along by the prevailing tides* (VOA). В приведенном примере первое словозначение, относящееся к разряду субъектно-объектных, имеет в качестве значимой зависимости объект глагольного действия *«нести что-либо»*. Второе словозначение относится к разряду субъектно-адвербиальных, что обусловливает его изменения в категориальной разрядности и его отнесенность к другой категориально-семантической группе в отличие от первого субъектно-объектного словозначения. Понятие направленности действия в этом случае является содержанием семантической валентности действия второго глагольного словозначения, а не его чисто грамматической зависимостью

и определяет его категориально-семантическую разрядность. Одновременно адвербиально зависимые слова дифференцируют семантику словозначения в семантических признаках: *нести вниз, нести вдоль, нести вместе с чем-либо/кем-либо*. Контекст в этом случае актуализирует имплицитное содержание адвербиальной зависимости в структуре единицы: *carry* (VOA) *disease, virus*. Семантика существительных дифференцирует логико-предметное содержание словозначения *«переносить болезнь»*.

В том случае, когда словозначения относятся к сходным категориально-семантическим разрядам, парадигмам и группам, средства контекста помогают дифференцировать интегральную семантику лексемы: *When the water has boiled let it cool. She doesn't know how to boil an egg.* Здесь словозначения *«кипятить»* и *«варить»*, проявляющие связи импликации, относятся к сходным категориально-семантическим разрядам, парадигмам и группам, проявляя тем самым тождественную категориально-семантическую разрядность. Семантическая дифференциация глагола в этом случае актуализируется семантикой зависимых существительных в синтагме: *кипятить воду, варить яйца/ бульон, отваривать капусту* и пр. В структуре глагола *stoop* субъектно-адвербиальные словозначения *«сутулиться»*, с одной стороны, и *«наклоняться»* – с другой могут дифференцироваться потенциальными валентностями глагола: *The girl stooped down and picked up the 2 pebbles.* Валентности цели (*pick up*) и направления (*down*) действия уточняют семантику глагола *stoop*.

Своеобразие прилагательных проявляется в том, что адвербиальные словозначения, связанные отношениями симиляции и импликации, как правило, относятся к различным категориально-семантическим разрядам и парадигмам, но сходным группам. Можно предположить, что средства контекста должны отражать категориально-семантическую разрядность словозначений в признаках *«конкретность – абстрактность»*, с одной стороны, и признаках *«качественность, количественность, оценка и отношение»* – с другой. Так, в словосочетаниях *wide shoulders* и *wide range of interest* семантические валентности словозначения *«широкий»* (о размере) и *«обширный»* (об уровне интересов) служит двойной цели: категориально-семантической и собственно семантической дифференциации словозначений.

Существительное при прилагательном играет роль семантической дифференциации логико-предметного содержания, что отличает его от именных и глагольных словозначений. Важной здесь является отнесенность существительных к различным категориально-семантическим парадигмам в обозначении предметных объектов и непредметных явлений действительности. Так, в сочетаниях прилагательного *brittle* с существительными, обозначающими политические объединения людей типа *empire*, реализуется словозначение *«хрупкий – легко распадающийся»*. В сочетаниях с существительными – названиями человека, его характер и поведение *brittle* обозначает *«небрежный, бесчувственный»*: *«Why not», she said wanting to be as bright and brittle as all the other people.* В приведенном примере контекст не дифференцирует семантику прилагательного и лишь сочетаемостные валентности с существительным *people* поддерживают идею характеристики нравственного лица человека. В сочетаниях с существительными со значением «звук» *brittle* имеет значение *«короткий и громкий, резкий»*: *There was a sharp brittle tinkling.*

Таким образом, анализ речевых употреблений словозначений показал, что актуальными для дифференциации словозначений являются средства контекста, дифференцирующие категориально-семантические признаки собственного разряда и его логико-предметное содержание. Такими средствами являются семантические валентности словозначений, обусловленные его связями с зависимыми членами предложения. Средства контекста выполняют также функцию семантической дифференциации словозначений. Лексико-семантические, грамматические и синтаксические средства контекста обусловливают категориальное и семантическое своеобразие единицы. Знание семантических валентностей словозначений является необходимым для дифференциации специфики его содержания.

Примечания

1. *Апресян Ю. Д.* Лексическая семантика. Синонимические средства языка. М., 1974. 367 с.
2. *Бондарко А. В.* Функциональная грамматика. Л., 1984.
3. *Конецкая В. П.* Введение в сопоставительную лексикологию германских языков. М., 1993. 201 с.
4. *Кудреватых Л. П.* Семиологические основания семантических типов слов. М., 1996. 405 с.

Г. А. Лекомцева, А. В. Баранова

УПОТРЕБЛЕНИЕ ПЕРФЕКТНЫХ ФОРМ В СОВРЕМЕННОМ АНГЛИЙСКОМ ЯЗЫКЕ

В результате проведенного исследования на материале газеты «The Times» было отмечено, что в газетном стиле из всех перфектных форм используется в основном Present Perfect, употребляемое в своем типичном значении. Авторы критически проанализировали книгу Г.В. Вейхмана «Новое в грамматике современного английского языка» и выразили свое несогласие с некоторыми положениями, касающимися перфектных форм, в частности о принадлежности to be + Past Participle к временам группы Perfect и о месте перфектных форм в газетном тексте.

ЛЕКОМЦЕВА Галина Александровна – старший преподаватель кафедры германских языков ВятГГУ
БАРАНОВА Анна Владимировна – студентка V курса факультета информатики ВятГГУ

Язык постоянно эволюционирует, поэтому необходимо следить за его развитием, чтобы адекватно использовать его в коммуникации. Особенно это важно для изучающих иностранные языки, в частности английский язык. Сравнивая его с русским языком – эталоном для наших учащихся, необходимо отметить наличие типов, не имеющих аналогов в русском языке. Одной из таких форм является перфект. Перфект (от латинского perfectus – совершенный) – видовременная форма глагола (собственно Перфект), обозначающая состояние в настоящем как результат предшествующего действия, изменения; и/или действие, событие, состояние прошлого, чем-либо важное для настоящего, рассматриваемое с точки зрения настоящего, вне связи с другими фактами прошлого.

Перфектные глагольные формы заключают в себе непосредственное сочетание причастия второго с какой-либо формой глагола have (иметь). Большинство грамматистов признают это сочетание как единственный способ образования перфекта, но в последнем исследовании Г. А. Вейхмана «Новое в грамматике современного английского языка» (М., 2002) утверждается, что «формы Perfect со вспомогательным глаголом have передают действие, а со вспомогательным глаголом be обозначают достигнутое состояние. Формы с be были более распространены в XVIII – XIX веках, некоторые из них (особенно gone) употребляются до сих пор» [1]

Автор не определяет частотность употребления этих форм, он просто констатирует их наличие в современном английском языке, поэтому мы не можем согласиться, что глагол be (быть) образует времена группы Perfect наряду с have (иметь) в качестве вспомогательного глагола.

Для подтверждения нашей точки зрения обратимся к работе авторитетного автора З. Я. Тураевой. Она пишет, что «изучение языкового материала – анализ глаголов, вступавших в сочетание с глаголом «быть» в различные периоды развития языка, анализ структурных и семантических особенностей данного построения и его синтаксических связей – свидетельствует о том, что сочетание глагола "быть" со вторым причастием ни в один период развития языка не достигало такой степени грамматизации, как сочетание глагола "иметь" со второй формой и не являлось формой глагола» [2]. И далее: «...синтаксические связи сочетания глагола "be" со вторым причастием типичны для составного именного сказуемого, а не для аналитической формы глагола» [3].

Как уже было отмечено, алломорфизм перфекта создает определенные трудности при изучении русскоязычными студентами английского языка, особенно формы Present Perfect. Хотя вспомогательный глагол в нем и имеет форму настоящего времени (have, has), оно обычно сближается с прошедшим неперфектным (простым, Simple Past) и обозначает, как и эта последняя форма, тот или другой процесс в прошлом.

Сравним: *He has found the key.* – Он нашел ключ.
He found the key. – Он нашел ключ.

Вместе с этим замечается и специфическая особенность значения перфектности, отличающая это значение от значения прошедшего времени: значение перфектности – это осуществление процесса до момента данной речи. Такое различие с наибольшей четкостью выступает при сопоставлении будущего времени перфекта с обычным прошедшим:

He will have finished it by that time. – Он окончит это к тому времени.
He finished it. – Он окончил это.

В будущем перфекте очень четко и ясно значение перфектности характеризуется как значение относительного прошедшего. При изучении Present Perfect специфическое затруднение вызывает определение смыслового отличия настоящего времени перфекта от обычного (неперфектного) прошедшего. Различие здесь оказывается очень тонким, и его можно легко упустить из виду. Так, в американском варианте английского языка наблюдается большее или меньшее стирание различия между обеими формами. Но в литературном английском языке настоящее время перфекта продолжает регулярно и достаточно устойчиво отличаться по употреблению от прошедшего обычного. В некоторых грамматиках дифференцированность в употреблении настоящего перфектного и прошедшего обычного (Present Perfect и Simple Past) объясняется в большей или меньшей мере словарным контекстом. Например, при just употребляется Present Perfect, при *just now* – Simple Past (Past Indefinite): *He has just come. – He came just now.* – Он только что пришел.

Поэтому Г. А. Вейхман в своей работе уделяет большое внимание таким словам, называемым им маркерами. Автор дает уточнения и дополнения, полученные в результате изучения различных исследований за последние сорок лет. Он отмечает, что связь маркеров с соответствующими временами ослабевает. Это справедливо и для Present Perfect. Его маркеры употребляются с глаголами в 1) Past Indefinite, 2) Present Indefinite, 3) Present Continuous. Это объясняется многозначностью соответствующих маркеров, а также влиянием контекста или ситуаций. Например, recently – в значении «за последнее время» употребляется с Present Perfect, а в значении «недавно» с Past Indefinite.

Подобная многозначность характерна также для *until recently, just now, ever, never, lately, since, since when.* С другой стороны, Present Perfect употребляется с маркерами, характерными для Present и Past Indefinite.

e.g. *She's moved **long ago** ...*
When have I been harsh, tell me?

Таким образом, даже наличие соответствующих маркеров не дает точного указания на употребление Present Perfect. Следующей трудностью для русско-

язычных студентов является употребление Past Perfect, так как в русском языке нет предпрошедшего времени. Как отмечает Г. А. Вейхман [4], в современном языке резко сокращается употребление форм Past Perfect. Автор дает рекомендации по употреблению времен в сложноподчиненном предложении. Средствами передачи предшествования могут быть союзы, наречия, контекст. Употребление Past Indefinite вместо Past Perfect характерно для глаголов know и understand. Итак, Г. А. Вейхман дал глубокий анализ современного употребления времен группы Perfect и изменений в их системе, включая стилистическую сферу употребления Present Perfect. Он отмечает, что в британском английском формы Present Perfect употребляются в основном в разговорном стиле (в беседах, состоящих из вопросов и ответов, в газетах, радиопередачах и т. д.). В газетах и радиопередачах Present Perfect обычно используется в первом предложении для сообщения главной новости, а затем подробности описываются в форме Past Indefinite.

Чтобы продолжить исследование современного состояния времен группы Perfect, мы использовали материал газеты «'The Times' Weekend» от 5 января 2002 г. на 16 страницах [5]. Выделяют несколько функций газеты: информационную, просветительскую, воспитательную, организаторскую, развлекательную. Однако основной ее функцией является воздействующе-информационная, и доля средств и способов достижения экспрессивности оказывается в публицистической речи весьма высокой, т. е. содержание статьи, ее тематика влияет на используемые языковые средства. Так, в данной газете времена группы Perfect отмечены в 23 статьях из 37. Это в основном информационные статьи, а также просветительские и развлекательные. В объявлениях и рекламе Perfect не отмечен. В заголовках встречается Present Perfect, 1% от общего числа форм группы Perfect.

В результате исследования было выявлено, что в данном номере газеты есть все формы группы Perfect, причем 69% составляют Present Perfect, 24% – Past Perfect, 7% – Future Perfect, что соответствует выводам Г. А. Вейхмана о преимуществе употребления Present Perfect в английском языке. Однако наше исследование не подтверждает точку зрения Г. А. Вейхмана, что Present Perfect употребляется только в начале статьи. В данной газете эта форма встречается также и в середине, и в конце статей. Далее мы проследили наличие маркеров Present Perfect, следует отметить, что они используются редко, в 19,4% всех случаев употребления Present Perfect. Наряду с известными *since, ever since, yet, for more than a century, for years, long, always, never, just, this year,* встретились маркеры *«to the 21st century, over 15 million years, over the years»*:

1) *It* ***has been rebuilt*** *four times since it opened in 1705.*

2) *A folded cotton T-shirt that has* ***not yet*** *been worn.*

3) ***For years*** *you've been told that a firm bed is better for your back.*

4) *Magnums and half bottles have* ***long*** *been an L&W sale treat.*

5) *It hasn't* ***always*** *been like this.*

6) *London's theatres* **have suffered** *a rough start* ***to the 21st century.***

Поскольку основным значением времен группы Perfect является предшествование или результат, их употребление зависит от содержания статьи или контекста. Данный выпуск газеты предназначен для чтения в выходные дни, т.е. тематика статей разнообразна и интересна. Тем не менее не везде необходимо обозначение предшествования или результата, поэтому распределение форм Perfect в газете неравномерно.

Так, в статье «Theater of dreams» [6] рассказывается о Лондонских театрах, их истории и современном состоянии, e. g.:

1) *London's old stages face tough times, but their gilt gods and cherubs* ***have seen*** *far worse – fire, riot and collapse.*

2) *Theatres* ***have often proved*** *as ephemeral as their productions.*

3) *The Royal Opera house* ***has thrice suffered*** *fire.*

Поэтому 8 случаев употребления Present Perfect в данной статье вполне соответствуют содержанию. В подборке статей «Live like guests in your own home» [7], советующей лондонцам покупать дешевые старые отели, чтобы проводить в них уикенд на Юге, употребляются Present Perfect 8 раз, а Past Perfect – 3 раза, e. g.:

1) *There are people who* ***have realized*** *that they don't need to pack their bags.*

2) *It* ***has*** *all* ***worked*** *out far better than we* ***had hoped.***

3) *…The service* ***will have deteriorated*** *somewhat from the standards set by the previous owners.*

Наряду с этими примерами в статьях, посвященных кулинарным рецептам, описанию деревьев, рекламе, перфект не употребляется.

Это еще раз подтверждает, что использование перфектных времен в большой степени зависит от контекста.

Итак, в результате исследования можно сделать вывод, что в современной британской прессе употребляются времена группы Perfect. Если Present Perfect встречается в 72 случаях, то Past Perfect используется в 20 случаях. Причем значение этих форм в основном соответствует описанию, данному Г. А. Вейхманом в его книге «Новое в грамматике английского языка». Это предшествование и результат. В газете встречаются типичные маркеры перфекта, но есть и вновь образованные. Они встречаются 14 раз. Некоторые положения Г. А. Вейхмана можно оспорить. Например, формы Perfect употребляются не только в начале абзаца, как считает Г. А. Вейхман, но и в середине и конце статьи. Кроме того, не встретилось случаев Present Perfect со вспомогательным глаголом be (быть), так как эта форма потеряла грамматическое значение и встречается чрезвычайно редко.

Примечания

1. *Вейхман Г. А.* Новое в грамматике английского языка. М., 2002. С. 162.
2. *Тураева З. Я.* О развитии перфектных форм в истории английского языка // Вопросы английской филологии. Л., 1962. С. 376.
3. Там же. С. 379.
4. *Вейхман Г. А.* Указ. соч. С. 174.
5. «The Times» Weekend Saturday. 2002. January 5.
6. Ibid. P. 1.
7. Ibid. P. 12.

Г. А. Лекомцева, Л. А. Буркова

ЭВОЛЮЦИЯ МОДАЛЬНЫХ ГЛАГОЛОВ В СОВРЕМЕННОМ АНГЛИЙСКОМ ЯЗЫКЕ

В ходе анализа английской газеты «The Guardian» были выделены модальные глаголы, проанализированы их значения. Установлено, что в основном в английском языке преобладает устоявшаяся система модальных глаголов, которая претерпевает минимальные изменения, что соответствует точке зрения Г. А. Вейхмана, выраженной в его книге «Новое в грамматике современного английского языка».

Модальность (лат. modus – способ, мера) – функционально-семантическая категория, выражающая разные виды отношения высказывания к действительности, а также разные виды субъективной квалификации сообщаемого. Модальность является языковой универсалией, она принадлежит к числу основных категорий естественного языка, «в разных формах обнаруживающихся в языках разных систем … , в языках европейской системы она охватывает всю ткань речи» [1]. Категорию модальности большинство исследователей дифференцируют, один из аспектов дифференциации – противопоставление объективной и субъективной модальности.

Объективная модальность – обязательный признак любого высказывания, одна из категорий, формирующих предикативную единицу – предложение. Объективная модальность выражает отношение сообщаемого к действительности в плане реальности (осуществляемости или неосуществляемости) и ирреальности (неосуществленности). Главным средством оформления модальности в этой функции является категория глагольного наклонения.

Субъективная модальность, т. е. отношение говорящего к сообщаемому, в отличие от объективной модальности, является признаком высказывания.

Смысловую основу субъективной модальности образует понятие оценки в широком смысле слова, включая не только логическую (интеллектуальную и рациональную) квалификацию сообщаемого, но и разные виды эмоциональной реакции. Субъективная модальность реализуется по-разному.

Основным способом выражения модального отношения между субъектом действия и действием в английском языке (как и в других германских языках) являются модальные глаголы.

Этот способ – лексико-синтаксический, поскольку модальное значение передается посредством лексического значения модального глагола, входящего в составное модальное сказуемое, представляющее собой лексическую единицу.

Вопрос о том, какие глаголы необходимо считать модальными, грамматисты решают неоднозначно. В соответствии с более традиционной точкой зрения к модальным глаголам относят ограниченную исторически сложившуюся в германских языках группу глаголов. Для английского языка это глаголы *can, may, must, ought, shall, dare, need, will*. Эти глаголы по своему лексическому значению и ряду структурных признаков (способность соединяться только с инфинитивом, отсутствие неличных форм и др.) отличаются от полнозначных глаголов и выполняют служебную функцию. Отдельные лингвисты расширяют круг модальных глаголов в очень большой степени. Например, И. Такутани [2] насчитывает 19 модальных глаголов, включая в их число, помимо перечисленных выше, такие глаголы и сочетания, как *have, be, used to, (had) better, (had) best, be going to, be about to* и др. [3]

Второй вопрос – об употреблении и значении модальных глаголов – сложный, так как в разных условиях контекста эти глаголы могут выражать разные модальные значения. Более традиционным является мнение о полисемантичности модальных глаголов. Г. А. Вейхман в своей книге «Новое в грамматике современного английского языка» [4] подробно рассматривает значение модальных глаголов, связь между значениями одного глагола и факторы, влияющие на интерпретацию значений модальных глаголов. Так, в значении «необходимость» употребляются глаголы *need* и *must*. Но вопросы с ***need*** *«Need I go, Mother?»* – «Неужели мне надо идти?» отличаются от вопросов с ***must*** в *«Must I go, Mother?»* – «Мне надо идти?» тем, что говорящий надеется получить отрицательный ответ.

В значении «разрешение» глаголы ***can*** и ***may*** различаются:

1) *стилистически*: здесь отмечается, что глагол *can* имеет более разговорный стилистический оттенок. Вопрос *«How may I help you?»* звучит вежливее, чем *«How can I help you?»*. Употребление *can* вместо *may* со значением разрешения сделать что-либо распространяется из разговорного в нейтральный стиль. Например, преподаватель – студентам: *You can go now;*

ЛЕКОМЦЕВА Галина Александровна – старший преподаватель кафедры германских языков ВятГГУ
БУРКОВА Лариса Алексеевна – студентка V курса факультета информатики ВятГГУ

2) *оттенками значения:* так, в просьбе о разрешении, если после танцев мужчина спрашивает девушку *«May I see you home?»*, он не только подчеркивает свое желание, сколько стремится быть вежливым. Если он спрашивает *«Can I see you home?»*, он вполне ясно дает понять, что хочет сделать это.

В значении «предоставление разрешения» используются ***can*** и ***may***, но не ***could*** и ***might***, даже если они были употреблены в просьбе.

Could I use your phone? – Yes, of course, you can.
Might I trouble you for a light? – You may indeed.

Для сообщения о получении разрешения обычно употребляют *can*, а не *may*. E.g. *Joey can stay up till ten.*

Для передачи разной степени вероятности используются (в порядке нарастания степени вероятности): *might, could, may, should, must, will.* Например, после стука в дверь: *That might/could be Sydney* (наименьшая степень вероятности); *That must be Sydney* (очень высокая степень вероятности). Один модальный глагол может иметь несколько значений. Глагол ***may***, например, имеет основное значение «возможность», у которого пять вариантов:

1) возможность налицо: *They **may** be distinguished by their markings;*

2) гипотетическая возможность, которая может или могла не реализоваться: *It **may rain**. It **may have rained**;*

3) пожелание о возможности: ***May** your days be bright;*

4) побуждение к реализации возможности: *You **may** expect me at six;*

5) разрешение возможности: *You **may** hold the baby.*

На интерпретацию значения модальных глаголов влияют также ситуация и контекст.

Г. А. Вейхман [5] проследил эволюцию системы модальных глаголов. Он отмечает, что:

1) глагол *may* встречается все реже. Круг предложений с *may* со значением вероятности и разрешения неуклонно сужается. Наибольшую устойчивость проявляют клише, выражающие вероятность, типа: *What may that mean?*;

2) *must* постепенно вытесняется его эквивалентами *have to* и *have got to*;

3) в утвердительных предложениях *ought to* отличается от *should* не только значением, но и более разговорным характером: *You ought to (oughtta) tell your folks about it*;

4) появляется новый модальный глагол *want* со значением долженствования, предупреждения или совета, который все чаще употребляется вместо *ought to, must* и *should* (пока только в разговорном стиле): *You want to take it easy*.

Модальным по значению является и глагол ***bear*** – синоним глаголов *need* и *want*: *The boys bear watching* (=*should be watched*).

После рассмотрения всех точек зрения на употребление модальных глаголов в английском языке возникла необходимость изучить их использование в современных источниках. Наиболее подходящим материалом для этого является публицистика, в частности газета. Нередко выделяют функции газеты: информационную, просветительскую, воспитательную, организаторскую, развлекательную. Основной ее функцией является пропагандистско-информационная.

Газетная лексика неоднородна. Она включает большой круг книжных слов, но здесь используется и разговорная, а изредка и просторечная лексика, как правило, со специальными стилистическими целями. При изучении газетных материалов, опубликованных в «The Guardian» от 1 сентября 2001 года, мы 1) провели количественный анализ употребления модальных глаголов; 2) установили частотность использования разных форм инфинитива, следующих за модальными глаголами, что меняет их значение, а также 3) определили влияние контекста на значение модальных глаголов.

Сфера нашего исследования ограничена глаголами *can/could, may/might, must, be able to, to be allowed to, nave/had to, need, to be to.* Из табл. 3 [6] видно, что наиболее употребительны модальные глаголы – *can (cannot, can't could not, couldn't), may (might), to be able to u to be allowed to.*

Таблица 1

MV	Сочетания MV + Infinitive	
	Абсолютная частота	Удельный вес, %
CAN	32	17,77
CANNOT	8	4,44
CAN'T	10	11,11
COULD	39	21,67
COULD NOT	1	0,55
COULDN'T	8	4,44
MAY	18	10
MIGHT	6	3,33
TO BE ABLE TO	13	7,22
TO BE ALLOWED TO	13	7,22
HAVE TO	12	6,67
HAD TO	4	2,22
MUST	9	5
NEED	4	2,22
TO BE TO	3	3,36

Важным, на наш взгляд, моментом является форма инфинитива, следующая за модальным глаголом. Если употребляется Indefinite Infinitive, это не составляет особой трудности для учащихся при понимании и переводе глагольного сказуемого, например: *He can't go there.* – Он не может пойти туда (невозможность); но если после *can't* употребляется Perfect Infinitive или Continuous Infinitive, то значение модального глагола и структура русского предложения меняются существенно, например: *He cant' have gone there.* – Не может быть, что он ходил туда (сомнение); *She cannot be waiting for us.* – Не может быть, чтобы она нас ждала (сомнение).

Частота употребления тех или иных форм в качестве одного из конституентов таких сочетаний существенно отличается (см. табл. 2).

Таблица 2

	Сочетания MV + Infinitive		Without an Infinitive	Всего сочетаний
	Indefinite	Perfect		
Абсолютная частота	158	20	2	180
Удельный вес, %	87,78	11,11	1,11	100

Из таблицы видно, что сочетания MV+Perfect Infinitive встречаются в 8 раз реже, чем MV+ Indefinite Infinitive, на долю которых приходится более 87% общего числа отмеченных сочетаний. Например:

MV + Indefinite Infinitive – *He is now aware of his surroundings and can communicate with his family* (возможность);

MV + Indefinite Infinitive – *This fact must never be forgotten* (запрет);

MV + Perfect Infinitive – *Presumably she must have known that. We would be left with no option but to charge her* (возможность).

Употребление модальных глаголов главным образом в сочетаниях с инфинитивом действительного залога является характерной особенностью английского газетного стиля. Поскольку пассивные сочетания обычно используются при описании действий обобщенных, между ними и сочетаниями действительного залога наблюдаются некоторые семантические различия. Анализ исследования показывает, что семантика таких сочетаний значительно уже, в большей степени утрачивается элемент эмоциональности [7].

Таблица 3

	Сочетания MV + Infinitive				Всего сочетаний
	Indefinite		Perfect		
	Active	Passive	Active	Passive	
Абсолютная частота	150	10	16	4	180
Удельный вес, %	83,33	5,55	8,89	2,23	100

Как уже было отмечено, контекст и ситуация влияют на выбор модального глагола. Так в проблемной статье «Winfor woman in suicide bed», обсуждающей возможность легализации эфтаназии и различные точки зрения на эту проблему, употребляются различные модальные глаголы со значением возможности (*could*), позволения и разрешения (*might, to be allowed to*), например:

*But there is also the fear that patients who **could** possibly recover **might be allowed** to die.*

*Since then 20 more such patients **have been allowed** to die.*

*In Oregon, doctors are only allowed to provide terminally ill patients with the lethal drugs to end their lives and **cannot** administer them.*

Еще одна статья, связанная с эмоциональным стрессом и напряжением – «Daughter admits trying to kill woman in torment». 69-летняя дочь попыталась отравить, добавляя в пищу опасные для жизни таблетки, свою 90-летнюю прикованную к постели мать. Причиной этому послужило доведенное до отчаяния состояние дочери, поэтому для выражения этих эмоций использовались модальные глаголы со значением невозможности, невыносимости, а также выражающие умственную способность.

*She **could nor bear** to see (mother) her suffer.*

*She **is not allowed** to visit her mother unless supervised by a police officer.*

Модальный глагол *to be allowed to* употреблен здесь в настоящем времени, хотя это не характерно для данного эквивалента, выражающего разрешение, но сделано с такой целью, чтобы подчеркнуть его значение.

В газетных статьях встречаются любопытные сочетания модального глагола с эквивалентом другого модального глагола, например:

1) *Policyholders hoping for news about possible compensation **may have to** wait for years until the investigation is completed;*

2) *Ordinary Turks may **not be able to afford** to be (impressed);*

3) *«I suffer from chronic conscientiousness», he says, clearly fretting that he **may not be able** to find a new outlet for his restless energy.*

При отборе примеров с модальными глаголами мы определили также значения этих глаголов [8].

Вероятность:

*Minister hope that a 10000 interest free loan **could** provide the deposit on a home.*

*Some western sites were already practicing self-censorship by not putting articles that **might** offend Baying on their sites.*

Возможность:

*We **may** monitor and record calls.*

Вынужденность:

*After Scotland, I **had to** go to Norwich by train.*

*Variety staff are not allowed to sell scripts, because they **have to** write about the potential buyers, but the article suggested that Bart had done so.*

Мысленные процессы:

*He **could** understand the frustration of passengers planning to take part in a national passenger boycott of train services.*

Необходимость:

*Eurotunnel **need to** have a more comprehensive system for preventing people entering tunnels.*

*My grandmother said: «You **need to** give your body to your husband and you soul to Jesus».*

*This possibility **must** clearly be investigated.*

Неизбежность:

*You **must** be aged 18 or over and give us a mortgage over your property.*

Обязанность:

*Additional Mortgage Security fee you **have to** pay.*

Разрешение:

*In Oregon, doctors **are** only **allowed to** provide terminally ill patients with lethal drugs to end their lives.*

*A condition of bail is that she **is not allowed to** visit her mother of the stress hormone cortisol.*

В статьях, где контекст не требует выражения этих значений, модальные глаголы не употребляются. Итак, исследовав теоретическую литературу о модальных глаголах, в том числе и последнее издание книги Г. А. Вейхмана «Новое в грамматике современного английского языка», мы установили, что в основном в английском языке преобладает устоявшаяся система модальных глаголов, которая претерпевает минимальные изменения.

Проанализировав употребление модальных глаголов в современной прессе на материале газеты «The Guardian», мы определили следующее:

1) частотность употребления модальных глаголов;

2) формы инфинитива, следующие за модальными глаголами и их частотность;

3) значения модальных глаголов в зависимости от контекста и ситуации;

4) существование незначительных изменений в семантике модальных глаголов;

5) влияние Perfect Infinitive и Continuous Infinitive на изменение значений предшествующих модальных глаголов и трудность их перевода на русский язык.

Примечания

1. *Виноградов В. В.* О категории модальности и модальных словах в русском языке. М., 1980. С. 4.

2. *Буркова Л. А.* Эволюция модальных глаголов в современном английском языке: ВКР (в рукописи). Киров, 2003.

3. Там же. С. 8.

4. *Вейхман Г. А.* Новое в грамматике современного английского языка. М., 2002.

5. Там же.

6. *Буркова Л. А.* Указ. соч. С. 64.

7. Там же. С. 6.

8. Там же. С. 64.

Е. М. Лобанова

К ВОПРОСУ СОЗДАНИЯ СЛОВАРНОЙ СТАТЬИ COMPUTER SCIENCE В АССОЦИАТИВНОМ СЛОВАРЕ

Ассоциативные словари, в отличие от обычных лексикографических, более полно отражают «реальную жизнь слова». В статье представлено ассоциативное поле термина Computer Science, дается его количественный анализ, интерпретация стереотипных реакций, перечень основных моделей идентификации. Результаты анализа могут служить основой для соответствующей словарной статьи ассоциативного словаря.

Словарь – это справочная книга, которая содержит слова, расположенные в определенном порядке, объясняет значения описываемых единиц, дает различную информацию о них или их перевод на другой язык либо сообщает сведения о предметах, обозначаемых ими [1].

К сожалению, существующие словари, организованные по принципу «минимального толкования», во многом являются лишь «памятниками лексике», не отражая всей совокупности оттенков значения слова и случаев его употребления. Как указывает И. А. Стернин, фиксация значений в словарях является результатом обобщения и отвлечения от лингвистической реальности и поэтому «нельзя утверждать, что то или иное слово в данном языке не имеет определенного значения или семантического компонента, поскольку это не отражено в имеющихся словарях. К тому же «словари часто запаздывают» [2]. Однако у человека, пользующегося словарем, часто возникает необходимость учитывать «реальную жизнь слова», не зафиксированную в обычных лексикографических словарях. Это особенно актуально для терминов новых наук, активно развивающихся в последние годы. Подобную полную информацию возможно получить в ассоциативных словарях. Обращение к индивидуальному сознанию носителей языка позволяет получить ассоциативные поля, в которых «реальная жизнь слова» представлена полно и разнообразно. К сожалению, современные ассоциативные словари не предназначены для рядового пользователя словарем, поскольку рассчитаны, в первую очередь, для исследования языкового сознания. Тем не менее перед лексикографией стоит задача интегрировать данные традиционной лексикографии и психолингвистики и создать современный словарь с учетом последних научных достижений. Данная задача особенно актуальна для отраслевых словарей новых, активно развивающихся научных направлений.

Целью нашего экспериментального исследования было установление специфики понимания термина

ЛОБАНОВА Екатерина Михайловна – студентка V курса факультета информатики ВятГГУ

«COMPUTER SCIENCE». Мы полагаем, что результаты исследования могут быть положены в основу словарной статьи современного отраслевого словаря.

В ходе свободного ассоциативного эксперимента испытуемым – 21 студенту V курса факультета информатики (специальность «информатика и английский язык») Вятского государственного гуманитарного университета – носителям русского языка, изучающим английский язык как специальность, было предложено записать 10 английских слов-ассоциаций на термин «COMPUTER SCIENCE». Теоретически в результате эксперимента мы могли бы получить 210 реакций. Однако не все испытуемые написали по 10 реакций. Таким образом, в итоге мы получили 192 реакции. Список полученных ассоциаций представляет собой ассоциативное поле исследуемого термина «COMPUTER SCIENCE».

Представим полный список реакций: *computer* (количество реакций – 20); *net* (15); *mouse* (12); *Internet, program* (8); *Windows* (7); *algorithm* (6); *Pascal* (5); *keyboard, informatics, programming* (4); *disk, diskette, file, monitor, notebook, operation system, processor, server, software* (3); *Gates, bite, database, Delphi, driver, CD-ROM, Chekanov, e-mail, Excel, faculty, games, hardware, information, interface, Microsoft Office, motherboard, Okulov, pixel, procedure, programming language, winchester, Word* (2); *array, access, button, cable, calculation, chat, client, communication, cursor, HTML, Intel, memory, message, Onegov, Pentium, Photoshop, printer, protocol, reset, science, service, site, socket, speed, SQL, system, table, TCP/IP, user, var, VBA* (1).

В ходе структурного анализа мы изучили ассоциативные поля и выделили в них отдельные слова и словосочетания. Количество реакций-словосочетаний – 7 (3 различных), отдельных слов – 185 (70). Представим список реакций-словосочетаний: *operation system, programming language и Microsoft Office.* Среди словосочетаний можно выделить две различные структуры: 1) noun + noun: *operation system и Microsoft Office;* 2) verbal + noun: *programming language.* Реакции-слова составляют большую часть реакций. Причем среди них можно выделить 181 (69 различных) существительных и 4 (1) глагольные формы: *programming*. Следует также отметить такие группы имен существительных, как сокращения – 7 (6 различных), имена собственные – 6 (13), из них фамилии – 7 (4), название фирм, различных программных продуктов – 29 (9).

Одним из традиционных методов анализа результатов ассоциативного эксперимента является анализ стереотипных реакций. Каждое слово-стимул может вызвать одинаковые, то есть стереотипные, реакции. В нашем случае будем считать реакцию стереотипной, если она встречается не менее пяти раз. Таким образом, мы получили 81 стереотипную реакцию: computer (202), net (153), mouse (124), Internet (85), program (86), Windows (77), algorithm (68), Pascal (15). Именно эти слова составляют ядро ассоциативного поля COMPUTER SCIENCE. Полученный список слов-ассоциаций можно условно разделить на три группы.

1. В первую группу включены слова, которые относятся к компьютеру, характеризуя его как электронно-вычислительную машину. Сюда относятся такие слова, как *bite, button, cable, CD-ROM, client, computer, cursor, disk, diskette, driver, file, hardware, Intel, interface, keyboard, memory, monitor, motherboard, mouse, notebook, operation system, Pentium, pixel, printer, processor, protocol, reset, server, service, socket, software, speed, system, TCP/IP, user, winchester, Windows.* Они составляют примерно 51% от общего числа перечисленных слов.

2. Ко второй группе можно отнести такие слова, как *access, algorithm, array, Bill Gates, calculation, database, Delphi, Excel, HTML, Microsoft Office, information, Pascal, Photoshop, procedure, program, programming, programming language, SQL, table, var, VBA, Word,* в эту же группу можно включить слова *faculty, informatics, science*, а также фамилии преподавателей *Chekanov, Okulov, Onegov*. Полученный список слов можно охарактеризовать как перечень слов, относящихся к информатике как к науке. Подобные слова составляют примерно 38% от общего числа слов.

3. И, наконец, третья группа включает в себя слова, которые относятся к компьютеру и характеризуют его как средство общения и развлечения. Сюда вошли слова *chat, communication, e-mail, games, Internet, message, net, site*, то есть всего 1% от всех перечисленных слов.

Таким образом, можно сделать вывод, что термин «COMPUTER SCIENCE» у студентов V курса Вятского государственного гуманитарного университета ассоциируется прежде всего с компьютером как электронной вычислительной машиной, а только потом уже с информатикой как наукой.

В ходе анализа полученного ассоциативного поля мы выделили следующие идентификационные стратегии (модели):

1) идентификация через составляющие компьютера: *keyboard, monitor, mouse, processor, wincheste;*

2) идентификация через основное назначение компьютера: *communication, message, programmin;*.

3) идентификация через название программных продуктов: *Excel, Delphi, Pascal, Photoshop, Word;*

4) идентификация через конкретное лицо, имеющее отношение к слову-стимулу: *Chekanov, Gates, Okulov, Onegov;*

5) идентификация через простое слово – часть сложного слова-стимула: *computer, science.*

Именно эти идентификационные модели определяют специфику понимания исследуемого термина и могут быть использованы для составления словарной статьи современного словаря.

10 Зак. 2868

Примечания

1. Лингвистический энциклопедический словарь / Гл. ред. В. Н. Ярцева. М.: Сов. энцикл., 1990. 462 с.
2. *Стернин И. А.* Уровни описания языкового сознания // Филология и культура. Тамбов, 2003. С. 11-14.

А. А. Свицова

ПСИХОЛИНГВИСТИЧЕСКОЕ ИССЛЕДОВАНИЕ ПОСЛОВИЦ НА МАТЕРИАЛЕ РУССКОГО И АНГЛИЙСКОГО ЯЗЫКОВ

В статье описывается подготовка к психолингвистическому эксперименту, материалом для которого являются пословицы с темой «Дом. Родина. - Чужбина» и их русские корреляты. Использование различных методик позволит выявить механизмы понимания таких сложных явлений лингвистики и фольклора, как пословицы.

Особенности национальной картины мира невозможно постичь, не изучив сознание человека, зафиксированное с помощью языка. Проблемы взаимоотношения языка, культуры и сознания занимают одно из центральных мест в проблематике отечественной психолингвистики. Теоретической основой исследований служит представление о том, что явления реальной действительности, воспринимаемые человеком в структуре деятельности и общения, отражаются в его сознании таким образом, что это отображение фиксирует причинные, пространственные связи явлений и эмоций и образ мира меняется от одной культуры к другой [1]. Возможны различные способы выявления специфики образов сознания носителей той или культуры. Однако даже «самое тщательное детальное сопоставление языковых систем не может обеспечить достижение целей межязыковых и межкультурных сопоставлений» [2]. Более того, лишь психолингвистический подход, осуществляемый через обращение к носителям языка, акцентирует внимание на том, как функционирует язык в культуре и через культуру. [3] Из универсального характера функционирования языка вытекает положение, согласно которому язык не может быть не связан с национальной культурой, так как одна из целей деятельности общества – создание культуры [4]. Воздействие культуры на язык проявляется в своеобразии самого процесса общения в разных культурах, что откладывается в различных ярусах языка, а также в нормативно-стилистическом укладе языка [5]. В. Н. Телия [6] акцентирует внимание на том, что пословицы и поговорки, будучи в значительной степени реализацией кумулятивных функций языка, отражают стереотипы, стандарты, нормы и идеалы как часть культурно-исторического наследия, создаваемого в процессе практического освоения мира, относятся к области культурного знания, вооружающего носителя языка той системой ценностных ориентиров, которая принадлежит к концептуальной картине мира в ее обиходно-бытовом варианте, поскольку создание культуры – одна из целей создания общества [7]. Как указывает Ю. Н. Караулов [8], принадлежность людей к той или иной культуре определяется также знанием прецедентных текстов, тогда как незнание, наоборот, есть предпосылка отторженности от соответствующей культуры. Прецедентными текстами, безусловно, являются паремии, которые в сжатой форме передают из поколения в поколение наиболее важные идеи общества.

В. И. Карасик [9], анализируя лингвокультурные доминанты – системы особых концептов, раскрывающих ценностные приоритеты соответствующей культуры, указывает, что они могут быть «измерены», а их этнокультурная специфика может быть выявлена путем сопоставления ценностных суждений, вытекающих из стереотипов поведения, зафиксированных в значениях слов, устойчивых выражений, прецедентных текстов.

Одним из научных направлений, связанных с человеком как носителем «языкового сознания», «когнитивной базы», «картины мира», как представителем той или иной культуры является перевод. Как известно, проблемы перевода пословиц считаются одной из самых сложных проблем в теории перевода. Здесь принято выделять следующие основные типы перевода пословиц: при помощи соответствия в другом языке, создания подобия, воспроизводящего вещественный смысл составляющих пословицу слов и вместе с тем сохраняющих ее общий смысл и характер как определенной и единой формулы; создание своего варианта, представляющее известное видоизменение вещественного смысла отдельных составных частей словесной формулы подлинника, не приводящей к совпадению с уже существующей в языке перевода пословицей, но вызывающей впечатление сходства с существующими речениями этой категории [10].

Однако традиционный подход к переводу не учитывает специфику перевода в контексте порождения речи в целом. Новое направление в переводе – когнитивная транслятология, разрабатываемое Т. А. Фесенко, – занимается именно этой проблемой [11]. Особенностью является то, что переводные материалы рассматриваются не как «трансляция образов сознания исходного культурного пространства, но проекция образов сознания культуры реципиента» [12]

Цель нашего исследования, выполненного в рамках психолингвистической теории слова и внутреннего контекста, разработанных А. А. Залевской [13], – выявление особенностей идентификации пословиц современными носителями языка и сопоставление

СВИЦОВА Анна Альбертовна – аспирант кафедры германских языков ВятГГУ

«живого» понимания национально-культурной специфики русских и английских пословиц с их словарными трактовками.

Для достижения этой цели мы поставили несколько практических задач:

1) установить корпус исследуемых пословиц с тематической группой «дом – родина – чужбина» для предъявления их испытуемым;

2) выявить осведомленность испытуемых об этих пословицах; выяснить, какой смысл вкладывают в эти пословицы испытуемые;

3) сопоставить результаты эксперимента с целью выявления совпадений и различий в «живом» и словарном понимании пословиц;

4) попытаться выяснить, в чем особенности понимания национально-культурной специфики словарных пословиц коррелятов.

Для решения поставленных задач мы планируем использовать метод субъективных суждений, метод субъективных дефиниций и метод вербального описания.

Чтобы отобрать материал для исследования, мы обратились к русским и англо-русским словарям пословиц, так как целью было выбрать корпус английских пословиц и их словарными русскими коррелятами. Анализ словарей, организованных по тематическому принципу, позволил выделить круг тем, вокруг которых группируются пословицы и которые определяют многообразие пословичных объектов. Темами пословиц являются абстрактные свойства объектов и чисто абстрактные понятия, в которые, несомненно, входят «дом – родина – чужбина». А. П. Бабушкин [14] определил концепт абстрактного имени как «калейдоскопический», т. е. ему могут приписываться самые различные характеристики. В пословицах, с одной стороны, происходил отбор объектов, через которые абстрактное понятие могло быть определено в отдельной национальной культуре, с другой стороны, в них сохранились наиболее значимые для национальной культуры характеристики абстрактных понятий и объектов, их обозначающих. Как показывают исследования, список абстрактных понятий, вокруг которых группируются пословицы, является открытым списком и зависит от доминант, которые существуют в культурах разных народов.

Основным принципом группировки пословиц является тематический принцип. Мы остановились на группе пословиц, которые могут быть объединены в рамках тематической группы «дом – родина – чужбина». Выбор данного круга пословиц объясняется тем, что как в русском, так и в английском языках подобные пословицы широко представлены, хотя в русском языке их количество значительно превышает количество таковых в английском языке. Пословицы, репрезентирующие концепты «дом – родина – чужбина», относятся к культурным доминантам, являющимся ориентирами поведения человека в обществе и лежащим в основе его интерпретации окружающего мира.

Важным является также то, что слова, за которыми стоят концепты «дом – родина – чужбина», относятся к ядру лексикона человека [15], а единицы ядра выполняют роль опор в процессах опознания других слов.

Н. В. Уфимцева [16], рассматривая различные способы выявления специфики образов сознания носителей той или культуры, обращается к анализу ядра лексикона, выполненному А. А. Залевской. [17]. Н. В. Уфимцева [18] приписывает 2-й ранг слову «дом», 68-й ранг – слову «домой» языкового сознания, 22-й ранг – слову «house» и 25-й ранг – слову «home» в ядре английского языкового сознания

Мы выбрали 10 английских пословиц, относящихся к теме «дом – родина – чужбина», и подобрали к каждой английской пословице оригинальное русское соответствие, которое является ее словарным коррелятом.

1. *There is no place like home.* – В гостях хорошо, а дома лучше.
2. *Home is home though it be never so homely.* – Своя земля и в горсти мила.
3. *Dry bread at home is better than roast meat abroad.* – Дома и солома съедома.
4. *An Englishman's house is his castle.* – На своей печи – сам себе голова.
5. *Every bird likes its own nest.* – Всяк кулик свое болото хвалит.
6. *It is a foolish bird that soils its own nest.* – Худая та птица, которая свое гнездо марает.
7. *The wider we roam, the welcomer home.* – Всякому мила своя сторона.
8. *He has no home whose home is everywhere.* – Чужие стены не греют.
9. *Every dog is a lion at home.* – Всяк кулик на своем болоте велик.
10. *Men make houses, women make homes.* – Муж – дому строитель, нищете отгонитель.

Эксперимент планируется провести в несколько этапов. В начале испытуемые – студенты IV курса Вятского государственного гуманитарного университета, изучающие английский язык как специальность, – должны вспомнить и записать в течение 10 минут все известные пословицы с данной тематической группой. Одна группа испытуемых пишет английские пословицы, вторая группа – английские. В ходе следующего задания испытуемые должны просмотреть корпус предъявленных пословиц и отметить знаком «+» известные, а знаком «–» – неизвестные.

Таким образом, мы выясним осведомленность испытуемых об английских и русских пословицах. В ходе второго этапа эксперимента испытуемым зачитывается пословица. Они должны представить внутреннюю картину, возникшую при восприятии данной пословицы, и дать ее описание в письменном виде. Затем мы предлагаем испытуемым объяснить своими словами значение пословицы и подобрать ее эквивалент. Далее предъявляется английская пословица и

русский словарный коррелят и предлагается объяснить, в чем испытуемые видят разницу. На основе полученных результатов мы сопоставим словарное значение и «живое» значение пословиц и выясним, совпадают ли эквиваленты, которые подобрали испытуемые, со словарными коррелятами данных пословиц и попытаемся выявить особенности идентификации русских и английских пословиц.

Примечания

1. *Тарасов Е. Ф., Уфимцева Н. В.* Предисловие // Этнокультурная специфика языкового сознания. М., 1996. С. 5-6.
2. *Залевская А. А.* Вопросы теории и практики межкультурных исследований // Этнокультурная специфика языкового сознания. М., 1996. С. 23-39.
3. Там же. С. 25.
4. *Тарланов З. К.* Язык и культура. Петрозаводск, 1984.
5. *Мечковская Н. Б.* Социальная лингвистика. М.: Аспект Пресс, 1996. 206 с.
6. *Телия В. Н.* Метафора как модель смысла производства и ее экспрессивно-оценочная функция // Метафора в языке и тексте: Сб. науч. тр. М., 1988.
7. *Тарланов З. К.* Указ. соч.
8. *Караулов Ю. Н.* Роль прецедентных текстов в структуре и функционировании языковой личности. М., 1986.
9. *Карасик В. И.* Языковой круг: личность, концепты, дискурс. Волгоград, 2002.
10. *Федоров А. В.* Основы общей теории перевода. М.: Высш. шк., 1983. 165 с.
11. *Фесенко Т. А.* Специфика национального культурного пространства в зеркале перевода: Учеб. пособие. Тамбов: Изд-во ТГУ, 2002. 228 с.
12. *Фесенко Т. А.* Принципы когнитивной транслятологии // Реальность, язык и сознание: Междунар. сб. науч. тр. Вып. 2. Тамбов, 2002. С. 27.
13. *Залевская А. А.* Введение в психолингвистику. М.: Рос. гос. гум. ун-т, 1999. 381 с.
14. *Бабушкин А. П.* Типы концептов в лексико-фразеологической семантике языка. Воронеж: Изд-во Воронеж. гос. ун-та, 1996.
15. *Залевская А. А.* Проблема организации внутреннего лексикона. Калинин: Изд-во Калинин. гос. ун-та, 1997. 84 с.; *Золотова Н. О.* Ядро лексикона человека: формирование в отногенезе // Психолингвистические исследования: слово и текст. Тверь, 2002. С. 46-52.
16. *Уфимцева Н. В.* Русские: Опыт еще одного самопознания // Этнокультурная специфика языкового сознания. М., 1996. С. 139-162.
17. *Залевская А. А.* О комплексном подходе к исследованию закономерностей функционирования языкового механизма человека // Психолингвистические исследования в области лексики и фонетики. Калинин, 1981. С. 28-44.
18. *Уфимцева Н. В.* Русские...

Ю. А. Скурихина

ПСИХОЛИНГВИСТИЧЕСКОЕ ИССЛЕДОВАНИЕ БЕЗЭКВИВАЛЕНТНОЙ ЛЕКСИКИ

В статье приводятся различные классификации и способы перевода безэквивалентной лексики. В ходе обсуждения результатов психолингвистического эксперимента выделены три модели идентификации (конкретная, абстрактная, формальная). Дается описание способов перевода данных групп лексики испытуемыми в сравнении с общепринятыми способами.

В конце 60-х – начале 70-х гг. в практике преподавания иностранных языков сложилось новое направление, в основу которого был положен лингвострановедческий принцип. В соответствии с этим принципом изучение языка должно идти в тесной связи с изучением культуры народа, говорящего на этом языке. Такое направление получило название «лингвострановедение». Оно сочетает в себе элементы лингвистики (изучение языковых единиц) с элементами страноведения (изучение реалий культуры страны через обозначающие их слова).

Объектом лингвострановедения являются фоновые знания и лексика с ярко выраженной национально-культурной семантикой (к ней относят безэквивалентную, коннотативную и фоновую лексику). Наше исследование посвящено безэквивалентной лексике (реалиям).

Безэквивалентная лексика сравнительно легко опознается при сопоставлении языков, ибо в ней нагляднее всего проявляется специфика культуры страны данного языка. Безэквивалентными являются слова, служащие для выражения понятий, которые отсутствуют в иной культуре, и, как правило, не переводятся на другой язык одним словом, не имеют эквивалентов за пределами языка, к которому они принадлежат.

Реалии – названия присущих только определенным нациям и народам предметов материальной культуры, фактов истории, государственных институтов, имена национальных и фольклорных героев, мифологических существ и т. п. [1].

Например, всем американцам известно, что *Oval Cabinet* – «Овальный кабинет» – это рабочий кабинет президента США в Белом доме; *«Honest Abe»* – «честный Эйб» – прозвище президента Авраама Линкольна (Abe – уменьшительно-ласкательное от Abraham); *«Gettisburg Address»* – «Геттисбергская речь» Линкольна, в которой им было дано определение демократии. Всем американцам с детства знакомы подвиги легендарного лесоруба Пола Баньяна (Paul Bunyan) и его верного друга Голубого быка (Blue Ox); приключения лихого шерифа времен «дикого Запада» Бэта Мастерсона (Bat Masterson); названия сравнительно небольших по размерам населенных пунктов Конкорда (Concord) и Лексингтона (Lexington), которые вошли в американскую историю как места вооруженных столкновений американских повстанцев с английскими войсками, положивших начало войне за независимость, и т. п.[2]

СКУРИХИНА Юлия Александровна – выпускница факультета информатики ВятГГУ

Существуют различные способы классификации лексики, обозначающей реалии. Наиболее подробной является тематическая классификация, предложенная Г. Д. Томахиным [3]. В соответствии с ней выделяют следующие лексические группы.

I. Этнографические реалии. Реалии быта. Речевой этикет и нормы поведения.

Например, *wigwam* – вигвам (жилище северо-американских индейцев), *granfather's clock* – высокие старинные напольные часы, *«happy hour»* – время в барах, когда цены значительно снижаются, *«Treat or Trick», «Beggar's night», jack-o'-lantern* (реалии, связанные с Днем Всех Святых).

II. Географические реалии.

К этой группе можно отнести топонимы. Иногда их относят к ономастическим реалиям. Мы же будем рассматривать топонимы, которые являются географическими названиями, как географические реалии (а остальные – как ономастические). Также к этой группе относят названия растений, животных, птиц, полезных ископаемых и различные словосочетания, где эти названия встречаются.

Bald Eagle – белоголовый орлан, символ США, *gold rush* – золотая лихорадка.

III. Общественно-политические реалии.

Общественно-политическая лексика является одним из основных центров максимальной концентрации национально окрашенной лексики, отражающей особенности жизни народа – носителя языка. Здесь можно отметить такие группы слов: слова, обозначающие государственные символы и символы штатов: *star-spangled* – усеянный звездами (о флаге США), настроенный ура-патриотически (неодобр. об американцах); реалии, связанные с конституцией, законодательной властью, президентом и аппаратом Белого Дома: *pocket veto* – «карманное вето» – косвенное вето президента.

IV. Реалии системы образования, религии и культуры.

Это реалии школьного и высшего образования (очень важно отметить здесь различные звания). К этой группе также относятся слова, связанные с культом, религией, литературой (названия жанров, афоризмы, наиболее известные персонажи), театром, кино, музыкой, средствами массовой информации. Например, *A, B, C, D, E* – оценки по пятибалльной системе, *the three R's* – основы начального образования: чтение (*reading*), письмо (*writing*) и арифметика (*arithmetic*), *campus* – территория университета, колледжа, включая и парк, разг. – университет, колледж; *flunkey* – заваливший экзамен, неудачник, *BA* – *bachelor of arts* – бакалавр искусств.

V. Ономастические реалии (имена собственные).

К ономастическим реалиям относят репрезентативные имена (наиболее известный пример – *Uncle Sam* в значении «США»); аллюзивные – имена персонажей различных произведений: библейских, античной мифологии, мировой литературы, таких, как *Solomon* – Соломон, царь, отличавшийся мудростью, *Hercules* – Геракл, *Scrooge* – Скрудж, персонаж «Рождественской песни» Чарльза Диккенса, скупец; переносные значения имен собственных (*Rockefeller* – Рокфеллер, богач, *Lincoln* – Линкольн, олицетворяет честность).

Также к ономастическим реалиям относят топонимы. Мы отнесли названия штатов, островов, городов к географическим реалиям. А к данной группе отнесем названия кварталов, площадей, улиц, зданий и архитектурных комплексов (*Madison Avenue* – Мэдисон Авеню, центр рекламного бизнеса, *the Oval Office* – Овальный кабинет, рабочий кабинет президента).

Лингвострановедение обеспечивает решение целого ряда проблем, в частности главной лингвистической проблемы – адекватного понимания текста, поэтому оно выступает в качестве лингвистической основы перевода.

Перевод реалий достаточно сложен и требует от переводчика большой эрудиции и мастерства. Недостаточное знание истоии страны, политических и исторических деятелей, ассоциаций, связанных с географическими названиями, приводит к непониманию сравнений, исторических ссылок и, как результат, к ошибкам в переводе. Например:

– *«Reckon old Dill'll be coming home tomorrow», I said.*

– *«Probably day after», said Jim. «Mississippi turns 'em loose a day after".*

Последняя фраза означает, что в штате Миссисипи школьников отпускают на каникулы на день позже. В переводе находим:

– Может, завтра приедет Дилл, – сказал я.

– Наверное, послезавтра, – сказал Джимм. – Им ведь еще через Миссисипи переправляться.

Также проблемой является неумение «увидеть» реалию (особенно фразеологизм), что приводит к буквализму в переводе.

Примером может служить следующий перевод предложения ***Bread lines** in the cities grew longer* – «В городах росли очереди за хлебом». Однако реалия *bread-line* означает очередь безработных за получением бесплатного питания, и смысл предложения в целом передан не точно.

При адаптации текста необходимо учитывать функциональную роль реалии в сообщении. Иногда возможно устранение реалии, если она несущественна, как, например, при переводе названия книги Марка Твена «Connecticut Yankee at King Authur's Court».

Connecticut Yankee воспринимается в США как самый ловкий из всех янки, так как название штата *Connecticut* ассоциируется с искусными ремесленниками и ловкими торговцами, которые вытачивали мускатные орехи из дерева и продавали их как настоящие. Однако русскому читателю непонятны ассоциации, связанные со штатом Коннектикут, и поэтому вполне оправдан перевод «Янки при дворе короля Артура».

Практикуется также замена реалии эквивалентом в другой культуре (например, клюквенный → малиновый → темно-красный).

Нужно отметить, что устранение реалий снижает художественную ценность произведения, как и подмена иноязычных реалий в процессе перевода, которая искажает исходное сообщение.

Существует несколько способов передачи иноязычных реалий.

• *Транслитерация* (передача на уровне графем) и *транскрипция* (передача на уровне фонем).

• *Калькирование* – буквальный перевод слова или словосочетания.

• *Описание или разъяснительный перевод*.

• *Приближенный (приблизительный) перевод* – при помощи «аналога».

• *Трансформационный* (контекстуальный перевод).

Задача переводчика состоит в том, чтоб подобрать наиболее удачный способ. Необходимо помнить о том, что реалиям присущ временной и национальный колорит, которые должны быть сохранены в переводе.

Для нашего психолингвистического исследования безэквивалентной лексики были отобраны 20 реалий (слов и словосочетаний) в соответствии с тематической классификацией. Испытуемыми выступали 20 студентов V курса факультета информатики ВятГГУ (специальность «информатика и английский язык»), которые приняли участие в двух экспериментах.

В качестве основного экспериментального метода использовался психолингвистический эксперимент, а именно свободный ассоциативный эксперимент. Свободный ассоциативный эксперимент подразумевает отсутствие ограничений при выборе вербальной реакции, представляющей собой первую пришедшую на ум реакцию. Слова-реакции могут принадлежать к совершенно другой грамматической категории, являться синонимами, антонимами, могут быть представлены собственными именами, являться фразеологизмами, литературными аллюзиями и т.п. При сравнительно простой технике проведения ассоциативного эксперимента можно получить интересные результаты, способствующие выявлению различных характеристик слов и их отношений в системе языка.

Целью первого эксперимента является исследование стратегий идентификации безэквивалентной лексики в индивидуальном лексиконе.

Идентификация слова – процесс соотнесения носителем языка той информации, которая стоит за словом в индивидуальном сознании и подсознании. В результате эксперимента были получены ассоциативные поля слов-стимулов. Под ассоциативным полем мы понимаем всю совокупность вербальных реакций, полученных на одно слово-стимул. Ассоциативное поле слова отражает внутреннюю структуру индивидуального лексикона – сложной иерархической структуры взаимопересекающихся ассоциативных полей, в которых как в вербальной, так и в невербальной форме хранится вся языковая и неязыковая информация о мире. [1]

1. Структурный анализ, в ходе которого были изучены ассоциативные поля исследуемых слов, среди реакций на слова-стимулы были выделены отдельные слова, словосочетания и предложения.

Количество реакций-предложений – 4; словосочетаний – 64 (50 различных); отдельных слов – 329 (182 разных).

Соотношение различных типов реакций представлено в виде диаграммы на рис. 1.

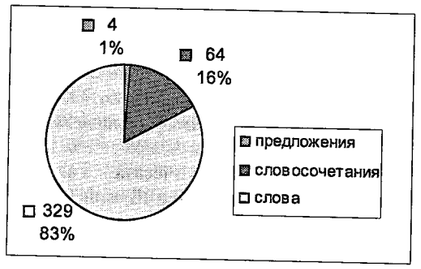


Рис. 1

Список полученных реакций-предложений приведен в табл. 1.

Таблица 1

№	Реалия	Предложение-ассоциация	Тип предложения
1	«Treat or trick»	The weather is fine	Простое, нераспространенное, двусоставное, повествовательное, утвердительное
2	Monday feeling	I want to sleep	
3	Auld Lang Syne	Merry Christmas!	Простое, нераспространенное, односоставное, восклицательное
4	TGIF	Hurra!	

Всего получено 4 реакции-предложения, причем это реакции на слова-стимулы, относящиеся к реалиям быта: 2 – к группе «праздники, традиции» (которая наиболее знакома испытуемым), 2 – к реалиям, связанным с периодом времени (начало и конец недели). Реакции-словосочетания тоже составляют малую долю всех реакций, причем среди них выделяется довольно-таки большая группа ономастических реакций – имен известных людей (киногероев) и названий стран, городов, улиц. Самую большую группу составляют реакции-слова. Причем среди них 322 существительных (156 различных), 31 прилагательное

(22) и 6 глагольных форм (4). Следует отметить такие группы имен существительных: сокращения – 26 (3 различных слова), имена собственные – 49 (8), из них названия стран – 30 (4), названия городов и улиц – 3 (2), название праздников – 15 (1), дней недели – 1 (1).

2. Анализ отказов от ассоциирования

В ходе ассоциативного эксперимента было получено 397 реакций (из них по 1 реакции на 273 слова и по 2 реакции на 62 слова) и 403 отказа от реакций (273 отказа от одной реакции из двух и 130 – от двух реакций, т. е. на 65 слов). Процентное соотношение возможных и полученных реакций представленно в виде круговой диаграммы на рис. 2.

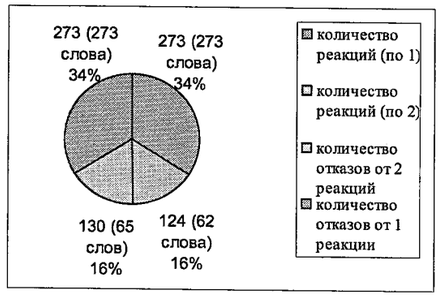


Рис. 2

Слова, получившие наибольшее количество отказов: *Gettysburg Address, the Star Spangled Banner, the «starter house», «happy hour», Honest Abe, TGIF.*

3. Анализ стереотипных реакций на предложенные реалии-стимулы, в ходе которого выявлены модели идентификации безэквивалентной лексики.

• Через объект, предмет, имеющий данное название: *Berkley – university (institute), Jack o'Lantern – pumpkin, Broadway – street.*

• Через название места, где встречается данная реалия: *Auld Lang Syne – Scotland, Bauld Eagle – USA, Oval Cabinet – White House.*

• Через время, когда происходит событие или используется предмет, обозначаемые реалией: *«Treat or trick» – Halloween, Indian summer – autumn.*

• Через причинно-следственную цепочку (реалия как причина или результат процесса): *TGIF – rest, Monday feeling – work (sleep).*

Также хотелось бы указать стратегии, не являющиеся стереотипными, но представляющиеся интересными.

• Через учебный предмет, дисциплину, с которым связана реалия: *Berkley – Computer science, Magna Carta – Country study.*

• Через эмоциональную оценку: *TGIF – hurrah!*

• Через развертывание ситуации: *Monday feeling – I want to sleep.*

Среди стратегий идентификации стереотипных реакций безэквивалентной лексики можно выделить три группы моделей: «конкретные», «абстрактные» и «формальные» (см. табл. 3).

Экспериментальным путем было установлено, что большинство реакций на предложенные реалии относятся к конкретным моделям.

Таблица 2

Идентификационные модели

«конкретные»	*«абстрактные»*	*«формальные»*
– Через объект, предмет, имеющий данное название	– Через причинно-следственную цепочку (реалия как причина или результат какого-либо процесса)	– Через простое слово – часть сложного слова-реалии
– Через название места, где встречается данная реалия		
– Через время, когда происходит событие или используется предмет, обозначаемые реалией		
– Через конкретный пример – типичное действующее лицо ситуации, в которой встречается данная реалия		
– Через конкретный пример – предмет, который входит в группу предметов, описываемых реалией или является его составной частью (элементом)		
– Через атрибутивную характеристику		

Целью второго эксперимента было выявление основных способов, используемых при переводе реалий, и их соотношения с общепринятыми способами перевода.

1. При анализе результатов были выявлены *способы, наиболее часто используемые при переводе:*

• калькирование («happy hour» – *счастливый час*, Monday feeling – *чувство Понедельника*, Stars and Stripes – *звезды и полоски*, Gettysburg Address – *Геттисбергский адрес*, Honest Abe – *Честный Эйб*);

• приближенный перевод /с помощью аналога (drug-store – *аптека*, Jack o'Lantern – *фонарь*, Monday feeling – *«Понедельник – день тяжелый!»*, Indian summer – *бабье лето*);

• объяснение (TGIF – *последний рабочий день*, Bald Eagle – *символ США*, Broadway – *улица*);

• транскрипция (Berkley – *Беркли*, Magna Carta – *Магна Карта*, Broadway – *Бродвей*);

• транскрипция + объяснение (Berkley – *университет Беркли*).

Соотношение используемых способов перевода представлено в виде диаграммы на рис. 3.

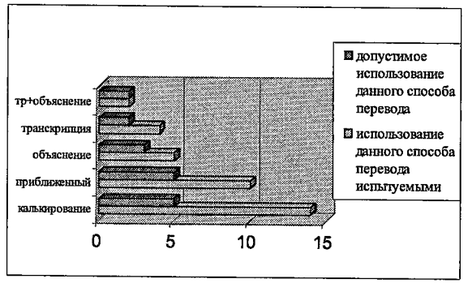


Рис. 3

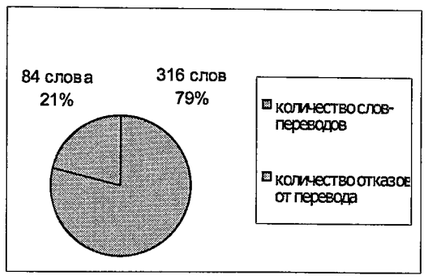


Рис. 4

Из диаграммы видно, что самыми популярными способами являются калькирование и приближенный перевод, однако в большинстве случаев использование их не оправдано. Также часто (в большем количестве случаев, чем необходимо) слова переводились при помощи транскрипции. Способ, который должен применяться при переводе большинства реалий (калькирование+описание), испытуемыми вообще не использовался.

2. Анализ отказов от перевода. Итак, в ходе эксперимента было получено 316 реакций и 84 отказа от реакций (рис. 4).

Наибольшее количество отказов было получено на слова *Gettysburg Addres, the «starter house», the Star Spangled Banner, Honest Abe, Auld Lang Syne.*

При сравнении данных обоих экспериментов (в том числе и по количеству отказов) оказалось, что наиболее сложными для перевода и ассоциирования оказались реалии быта и общественно-политические реалии, так как эта группа, по-видимому, менее всего знакома испытуемым. Легче всего идентифицируются географические реалии и реалии, связанные с традициями и праздниками, так как информацию о них можно получить из многих источников. Трудности также вызывают слова-аббревиатуры и словосочетания, смысл отдельных слов в которых не дополняет друг друга, а иногда и противоречит.

Примечания

1. *Томахин Г. Д.* Реалии-американизмы. М.: Высш. шк., 1988.

2. *Томахин Г. Д.* Америка через американизмы. М.: Высш. шк., 1982.

3. *Ахманова О. С.* Словарь лингвистических терминов. М.: Сов. энцикл., 1969.

О. С. Ульянова

ПОЛЕВАЯ СТРУКТУРА ТЕРМИНОВ COMPUTER SCIENCE В ИНДИВИДУАЛЬНОМ СОЗНАНИИ СТУДЕНТОВ ФАКУЛЬТЕТА ИНФОРМАТИКИ

В ходе интерпретации результатов двух психолингвистических экспериментов была установлена специфика полевой структуры терминов «Computer Science», определены структура и наполнение поля.

Моделирование системы языка – чрезвычайно эффективный прием ее познания. Желание исследователя наглядно, зримо представить себе, как «выглядит язык» в мозгу человека, в какой форме, «упаковке» заложен он в память и обеспечивает речевую деятельность, вытекает из особенностей познавательной деятельности людей. Видеть объект, иметь возможность воспринимать его всеми органами чувств – это наиболее привычный для человека способ познания действительности.

УЛЬЯНОВА Ольга Сергеевна – студентка V курса факультета информатики ВятГГУ

Одна из первых моделей системы языка – модель уровней, которую представляли в виде этажерки, спирали, цепочки и других материальных фигур, – дала лишь самую приблизительную разбивку состава языковой системы на классы элементов (фонем, слов, словоформ, словосочетаний и др.). В последнее время в лингвистической литературе все большее распространение получает полевая модель системы языка. Лингвистика исходит из того, что слово – это целая система связей между акустическим образом и смысловыми образами; следует добавить сюда связи и ассоциации между разными словоформами этого же слова, а также другими словами, имеющими с первым какие-либо общие компоненты. Группировка слов и словоформ может иметь очень сложный характер. Принято различать парадигматические сцепления слов и словоформ: лексико-семантические, лексико-грамматические, словообразовательные и некоторые другие. На возможность существования разных типов лексических объединений ученые обратили внимание еще в XIX веке (М. М. Покровский), некоторые особенности полевой структуры лексики были отмечены при построении первых словарей тезаурусного типа (П. Роже, В. фон Вартбург).

Термин «поле» был введен в лингвистику Й. Триром и Г. Ибсеном, которые работали с семантическим полем. Полевая модель утверждает представление о языке как системе подсистем, между которыми происходит взаимодействие и взаимопроникновение. По этой модели язык предстает как функционирующая система, в которой происходят постоянные перестройки элементов и отношений между ними. В процессе полевого структурирования раскрываются диалектические связи между языковыми явлениями и внеязыковой действительностью, механизм этой связи и его закономерности, выявляются особенности языкового сознания, раскрываются его национально-специфические черты. Помимо семантического поля в истории лингвистики были выделены морфологические поля (П. Гиро), грамматические поля, например залоговое поле (М. М. Гухман, А. В. Бондарко), синтагматические поля (В. Порциг). Близкими к семантическим полям являются лексико-грамматические поля, рассматриваемые как фрагмент ассоциативно-вербальной сети системы языка (О. М. Воевудская).

А. А. Залевская, создавшая в 1977 году модель лексикона индивида [1], указывает, что внутренний лексикон представляет собой чрезвычайно сложную систему многоярусных многократно пересекающихся полей, с помощью которых упорядочивается и хранится разносторонняя информация о предметах и явлениях окружающей действительности, об их свойствах и отношениях, об их оценке индивидом и о лингвистических особенностях обозначающих их вербальных единиц. Более того, А.А. Залевская на основании анализа внутриязыковых и межъязыковых ассоциативных связей слова приходит к выводу, что «ассоциативное поле интегрирует все известные в настоящем времени виды полей, в то время как описанные в лингвистических работах поля отражают лишь отдельные аспекты организации лексикона человека» [2].

Большинство лингвистов едины в том, что поле имеет сложную структуру: оно состоит из ядра, имеющего, как правило, обобщенное значение, и периферийных зон, включающих лексемы с дифференциальными признаками [3]. Однако в определении структуры поля у ученых единообразия нет. Самой распространенной моделью описания поля является описание его в виде опоясывающих друг друга кругов. Многие лингвисты представляют поле как иерархическую, многоуровневую систему.

Целью нашего исследования было выявление специфики полевой структуры терминов Computer science. Для формирования списка терминов был проведен эксперимент по методике экспертной оценки. В качестве экспертов выступали 15 студентов пятого курса факультета информатики ВятГГУ, изучающих английский язык. Им было предложено назвать 10 английских слов, относящихся к Computer science. Из полученного списка в 51 слово было выбрано 10 самых частотных (см. таблицу).

Список слов	Частота		Список слов	Частота
computer	14	6	Windows	6
net	10	7	algorithm	6
mouse	9	8	Pascal	5
program	8	9	programming	4
Internet	7	10	informatics	4

Именно эти 10 слов были использованы в качестве слов-стимулов в свободном ассоциативном эксперименте с ограниченным количеством ассоциаций. Испытуемым предлагалось записать первые три пришедшие в голову ассоциации на слово-стимул. Было получено 89 слов-реакций, среди них значительное количество повторяющихся (стереотипных) реакций. Ассоциативные поля каждого слова-стимула подвергались количественной и качественной интерпретации. Приведем для примера ассоциативное поле стимула INTERNET: *www* (7 реакций) (заглавными буквами обозначаем слово-стимул, строчными – ассоциативную реакцию), *computer networks* (7), *Yandex* (6), *network* (5), *search* (3), *global* (3), *e-mail* (3), *modem* (3), *computer* (2), *Outlook Express* (1), *IP address* (1), *browser* (1), *Explorer* (1), *Bill Gates* (1), *card* (1).

Итак, в результате двух экспериментов были получены два списка слов: тематический и ассоциативный. Сравнение этих списков показывает, что в них есть несовпадающие единицы. В тематическом списке отсутствуют 64 слова из списка ассоциативного, например *hardware, button, modem, Windows 95, coding, www*. В то же время в списке ассоциаций нет слов: *server, dish, winchester, database, file, diskette, FAT 33, Excel, driver, Intel, TCP/IP, socket, communication, processing, calculation, client, motherboard, HTML,*

compiles, CD-ROM, exhibition, speed, которые вошли в тематический список. Мы предлагаем считать ядром поля Computer Science единицы, зафиксированные в обоих списках: *computer, net, mouse, program, Internet, Windows, algorithm, Pascal, programming, informatics, monitor, operation system, keyboard, faculty, software, information, processor, e-mail, language, Delphi, Microsoft Office, Word, Bill Gates, Okulov, protocol, cable, science, array, interface.* Они будут составлять ядро исследуемого поля.

В ядре выделился центр: *computer, Internet, programming.* Эти слова относятся к центру по следующим признакам:

1) они есть во всех списках – тематическом и ассоциативном;

2) имеют большой показатель включенности в тематическую группу;

3) первыми внесены в список на формирование тематической группы информантами;

4) имеют взаимные ассоциативные связи: COMPUTER – programming, COMPUTER – Internet.

Слово *Computer* в составе центра является доминантой. В исследуемой полевой структуре можно выделить ближнюю периферию. В нее вошли слова:

1) с меньшей языковой частотностью и тематической включенностью в группу;

2) однозначные;

3) употребляющиеся в одном из двух списков.

Этим признакам удовлетворяют единицы: *winchester, diskette, Excel, TCP/IP, Socket, motherboard, HTML, compiler, CD-ROM, hardware, RAM, network, OSI, Ethernet, Web, site, WWW, Yandex, modem, Outlook Express, IP address, browser, Explorer, Java.*

Слова, вошедшие в дальнюю периферию, определяются по следующим признакам:

1) есть в одном из списков;

2) имеют меньший показатель включенности в тематическую группу;

3) некоторые слова многозначны.

Это такие единицы, как *integer, coding, Fano, adapter, connector, switch, Fiber optic, personal, FAT 32, processing, driver, Intel, file, database, server, Operations Research, button, disk, API.*

Крайнюю периферию составляют слова, включенные некоторыми информантами информатики в тематическую группу «computer science», выделенные в ассоциативных списках и имеющие не только информационное значение, а чаще употребляемые в других сферах. Эти единицы имеют единичную частотность: *subject, method, troubles, difficult, work, blue screen, scientist, important, structure, function, problem, command, card, search, global, game, cycle, begin, end, procedure, local, click, ball, roller, optical, institute, machine, exhibition, speed, client, calculation, communication.*

Итак, в ходе анализа экспериментального материала мы выяснили, что элементы computer science в лексиконе студентов факультета информатики организованы в структуру полевого типа. Детализованная полевая модель терминов computer science состоит из ядра, окруженного ближней, дальней и крайней периферией. При этом периферия существенно преобладает над ядром, причем дальняя периферия в количественном отношении превосходит ближнюю, а крайняя – дальнюю.

Примечания

1. *Залевская А.А.* Проблемы организации внутреннего лексикона индивида. Калинин: Калинин. гос. ун-т, 1977. 72 с.

2. *Залевская А.А.* Слово в лексиконе человека: психолингвистическое исследование. Воронеж: Изд-во Воронеж. ун-та, 1990. С. 157.

3. Полевые структуры в системе языка / Под ред. З.Д. Поповой. Воронеж: Изд-во Воронеж. ун-та, 1989. 199 с.

С. В. Шабардина

СИНОНИМЫ В ТЕРМИНОСИСТЕМЕ ПРАВА АНГЛИЙСКОГО ЯЗЫКА

Термины-синонимы входят в состав терминосистемы права. В плане тождества семантики терминов выделяются абсолютные и относительные синонимы. В плане структурного тождества – однокорневые, разнокорневые, синтаксические и простые/составные синонимы. Удельный вес среди терминов права синонимов вышеперечисленных семантических и структурных типов различен.

Материалом для исследования послужила правовая лексика современного английского языка (11 760 терминов права) [1], извлеченная методом сплошной выборки из специальных толковых словарей английского языка, двуязычных юридических словарей и текстов, представляющих различные жанры научной литературы: монографии, сборники действующих судебных решений и законодательных актов [2].

Как показало проведенное исследование, в состав терминосистемы права [3] современного английского языка входят правовые термины-синонимы, число которых достаточно велико и встречаются они во всех подсистемах терминологической системы права.

Наличие синонимических отношений внутри терминологий отмечается такими терминоведами, как Б. Н. Головин, С. В. Гринев, В. П. Конецкая, А. А. Реформатский, В. А. Татаринов, Е. И. Чупилина и др. [4]. Если изначально считалось, что синонимия – это признак, порочащий терминосистему (Д. С. Лотте, С. И. Коршунова) [5], то в настоящее время большинство терминоведов рассматривают синонимию как естественное явление, присущее терминологии как

ШАБАРДИНА Светлана Викторовна – кандидат филологических наук, доцент кафедры германских языков факультета информатики ВятГГУ

языковой системе [6]. В реально функционирующих терминосистемах различных областей знания выделяются отдельные синонимические ряды. Сущность синонимических отношений в лексике заключается в том, что тождество и различие семантических признаков слов взаимообусловлено, поэтому синонимические отношения рассматриваются как подтип системных связей [7].

При исследовании синонимии терминов важно учитывать различную степень тождества и различия семантики терминов и, кроме того, их структурное тождество/нетождество.

В плане тождества и различия семантики терминов нами выделяются: 1) абсолютные, или полные синонимы (ряд исследователей называют такие синонимы вариантами); 2) относительные, или частичные синонимы. Критерием этих двух типов синонимов является степень тождества компонентов их значения.

Абсолютные синонимы (или варианты) составляют 39% всех синонимов английского права. Абсолютные синонимы тождественны в плане семантики, но различаются по своей структуре: *law-charges/law-costs/law-expenses* – судебные издержки.

К относительным (частичным) синонимам относятся термины, в значениях которых выявлено тождество отдельных ЛСВ значений: ***false*** – 1) *wrong, incorrect*; 2) *deceitful, lying;* 3) *not genuine, artificial*; ***faked*** – *looking genuine, but is not.*

К относительным синонимам относятся также правовые термины, имеющие одно и то же ядро значения, но разные периферийные компоненты (признаки): ***plaintiff*** – *person who brings an action at law*; ***suitor*** – *person bringing a lawsuit*; ***claimant*** – *person, who makes a claim, especially in law.*

Согласно нашему анализу относительные синонимы значительно более частотны, чем абсолютные синонимы во всех подсистемах права. В общей сложности на их долю приходится 61% всех исследованных нами правовых терминов-синонимов.

В плане структурного тождества/нетождества все синонимы могут быть разделены на ряд типов: 1) однокорневые; 2) разнокорневые; 3) синтаксические; 4) простые/составные. Удельный вес терминов права-синонимов в структурных типах различен.

1. На долю *разнокорневых синонимов* приходится 61,5% всего объема синонимических терминов права: *plaintiff/claimant* – истец. Специфической чертой большинства разнокорневых синонимов английского права является их принадлежность к типу относительных синонимов (с точки зрения их семантики).

2. *Синтаксические (составные/составные) синонимы:* структурное различие этих правовых терминов-синонимов в том, что одно и то же правовое понятие выражается разными составными терминами, т. е. терминосочетаниями: *law of equity/equity law* – право справедливости. Синтаксические синонимы насчитывают 22,5%.

3. *Однокорневые синонимы* составляют 11% всего объема синонимов терминосистемы английского права. Среди однокорневых синонимов можно выделить ряд подтипов:

а) аффиксальные синонимы (варианты): структурное нетождество этих синонимов – в использовании разных суффиксов или префиксов: *pledger/pledgor* – залогодатель, поручитель; *unitarian/unitary* – унитарный, единый. На их долю приходится 5,5%;

б) морфологические синонимы (варианты): *visum/visa* – виза; *loss-losses* – потери, убытки. Они составляют 3%;

в) орфографические синонимы (варианты): *judgement/judgment* – приговор, решение суда. На долю этих синонимов-вариантов приходится 2,5%.

Отличительной чертой однокорневых синонимов является их отнесенность к разряду абсолютных синонимов.

4. *Простые/составные синонимы.* Структурное отличие этих синонимов состоит в том, что один синоним представлен простым по структуре правовым термином, а другой – составным, т.е. терминосочетанием. Эти синонимы составляют 3% всех синонимов: *leaseholder/holder of the lease* – арендатор.

Полученные нами результаты могут быть представлены в виде таблицы.

Таким образом, можно утверждать, что синонимия как подтип системных отношений характерна для терминосистемы права английского языка. В плане семантики различают два типа синонимов: относительные и абсолютные синонимы (или варианты). Отно-

Структурные типы синонимов терминосистемы английского права

Типы синонимов	Однокорневые			Разнокорневые	Синтаксические Составные / составные	Простые / составные
	Аффиксальные	Морфологические	Орфографические			
Доля синонимов, %	5,5	3	2,5	61,5	22,5	3

сительные синонимы отмечены наибольшей частотностью (61,5%). В плане структурных различий выделяется 4 структурных типа синонимов: однокорневые, разнокорневые, составные/составные и простые/составные. По нашим данным, преобладают разнокорневые синонимы.

Примечания

1. *Правовой термин* понимается как слово или словосочетание, соотнесенное с определенным понятием права.

2. Англо-русский юридический словарь / Сост. Г. А. Командин. М., 1993; Англо-русский юридический словарь / Под ред. С. Н. Андрианова. 2-е изд. М., 1998; *Eddey K.* The English legal system / Ed. by K. Eddey. L., 1987; *James Ph. S.* Introduction to English law. 12-th ed. L., 1989; *Jowit W. A.* The dictionary of English law. L., 1959; *Maley J.* The language of the law // Language and law / Ed. by J. Gibbons. L., 1994; *Schlauch M.* The English law in modern times (since 1400). Warszawa, 1959; *Stephen J. F.* A digest of the Criminal Law (indictable offence). 9-th ed. L., 1950; *Walker D. M.* The Oxford Companion to Law. Oxford, 1980.

3. Вслед за В. А. Татариновым, давшим определение терминовистемы в целом [14], понятие *правовая терминосистема* рассматривается как соотнесенный с правовой наукой, взаимосвязанный, существующий в равновесии набор правовых терминологических единиц всех существующих типов, связанных друг с другом на понятийном и языковом уровнях.

Поскольку система английского права охватывает различные отрасли (уголовное право, конституционное право, деликтное право, доверительное право, контрактное право, коммерческое право, семейное право, судебную систему Англии, систему судебного производства по уголовным и гражданским делам и т.д.), то под *английской правовой терминологией* в данном исследовании понимается совокупность правовых терминов, соотнесенных с понятиями всех отраслей английского права.

4. *Головин Б. Н., Кобрин Р. Ю.* Лингвистические основы учения о терминах. М., 1987; *Гринев С. В.* Введение в терминоведение. М., 1986; *Конецкая В. П.* О системности лексики // Вопросы языкознания. 1984. № 1; *Реформатский А. А.* Термин как член лексической системы языка // Проблемы структурной лингвистики. М., 1967. С. 103-125; *Татаринов В. А.* Теория терминоведения: В 3 т. Теория термина: история и современное состояние. М., 1996; *Чупилина Е. И.* О терминологической синонимии // Семиотич. проблемы языков науки, терминологии, информатики. М., 1971.

5. *Лотте Д. С.* Образование системы научно-технических терминов. Элементы термина // Изв. АН СССР. ОТН. 1948. № 5. Перепеч. в: *Лотте Д. С.* Основы построения научно-технической терминологии. М., 1961.

6. *Головин Б. Н., Кобрин Р. Ю.* Указ. соч.; *Гринев С. В.* Указ. соч.; *Конецкая В. П.* Указ. соч.; *Реформатский А. А.* Указ. соч.; *Татаринов В. А.* Указ. соч.

7. *Конецкая В. П.* Указ. соч.

А. М. Шиляева

ИЗ ОПЫТА ИСПОЛЬЗОВАНИЯ ПРОЕКТНОЙ МЕТОДИКИ (ФЕСТИВАЛЬ-КОНКУРС «АНГЛИЯ. ИРЛАНДИЯ. ШОТЛАНДИЯ. УЭЛЬС»)

В статье представлен практический опыт использования метода проектов на внеклассном мероприятии второго курса факультета информатики. Описаны этапы подготовки и проведения фестиваля конкурса. Дается анализ положительных и возможных проблемных сторон мероприятия. Предложены сферы употребления данной творческой методики.

Обучение иностранному языку – сложный процесс, связанный с такими компонентами психической и ментальной деятельности, как восприятие и память, мышление и языковое сознание, механизмы переключения с одного языка на другой, способы хранения лексики в памяти обучаемого, а также с множеством других психолингвистических факторов обучения. При этом обучение иностранному языку протекает наиболее успешно, когда оно имеет коммуникативный характер, т. е. когда изучающие его вовлечены в творческую деятельность. Конечно, в современном нам мире интерес к овладению английским языком достаточно высок, и студенты, поступившие на факультет информатики с дополнительной специализацией «Английский язык», осознают его важность для своей будущей жизни и карьеры. Однако у части студентов положительная мотивация недостаточна, так как при изучении языка они сталкиваются с трудностями и не усваивают материал в полной мере в силу своих психологических особенностей (особенностей восприятия, памяти, мышления). Именно поэтому мы решили воспользоваться методом проектов и провести фестиваль-конкурс «Англия. Ирландия. Шотландия. Уэльс» с участием всех студентов второго курса. Такое мероприятие позволило решить проблему мотивации и создать положительный настрой к изучению английского языка, дало возможность студентам активно применить свои знания на практике и раскрыть потенциальные таланты каждого.

Проведение фестиваля-конкурса стало традицией на факультете информатики, так как уже в течение двух лет мы успешно проводили такое нестандартное занятие, завершающее тему «География Британских островов». Задумывая проведение этого мероприятия, мы выдвигали следующие задачи.

1. Содействовать распространению английского языка и англоязычной культуры среди молодежи.

2. Пробуждать и поддерживать интерес к расширению знаний в области страноведения.

ШИЛЯЕВА Анна Михайловна – старший преподаватель кафедры германских языков ВятГГУ

3. Поддерживать традиции изучения в России языка, литературы и культуры Британии.

4. Развивать фантазию, воображение, творческие способности студентов.

5. Расширять их общий кругозор.

Помимо упомянутого немаловажного воспитательного значения, такая внеклассная работа, безусловно, помогает лучше решать задачи учебного процесса, а именно:

- усиливать мотивацию к изучению английского языка;
- способствовать практическому владению речевой деятельностью, т.е. тренирует участников в аудировании и говорении;
- увеличивать активный словарный запас студентов;
- совершенствовать их произношение;
- стимулировать в целом их интеллектуальную и языковую активность.

Проведение фестиваля-конкурса проходило в три этапа. Приведем примеры работы на каждом из них.

На ***первом этапе*** (на занятии) студентам предлагается в скрытом виде проблемный вопрос, который нужно выявить и сформулировать. Для этого перед объявлением об участии в фестивале-конкурсе и началом учебной темы «География Британских островов» мы провели устную анкету среди студентов, попросив их назвать все, что ассоциируется у них с отдельными частями Великобритании. В связи с тем, что количество полученных ассоциативных связей было очень неравномерным (так, например, Англия оказалась самой простой, а Уэльс – самым трудным для ассоциирования), был сформулирован проблемный вопрос, на который впоследствии предстояло ответить, чем знамениты разные части Великобритании?

Чтобы решить проблему, которая лежит в основе проекта, участники изначально должны обладать определенными интеллектуальными, творческими и коммуникативными умениями. К ним можно отнести умение работать с текстом, анализировать информацию, делать обобщения; умение работать с разнообразным справочным материалом. Непосредственное же участие в фестивале-конкурсе требует от студентов высокого уровня речевой подготовленности: они должны свободно владеть активной лексикой и грамматикой в рамках учебной темы. Отсюда следует необходимость во втором, подготовительном этапе, который должен обеспечить языковые и речевые умения студентов.

Второй этап проходил как во время запланированных программой занятий, так и внеурочно. Работа над проектом занимала 3-4 недели, при этом параллельно шло поэтапное освоение материала по теме «География Британских островов» на базе текстов и упражнений из учебника, а также самостоятельная поисковая деятельность студентов. Участникам был заранее объявлен примерный тематический план мероприятия, в соответствии с которым в итоговой защите проектов предстояло творчески представить следующее:

а) визитную карточку страны (флаг, герб, национальный музыкальный инструмент, растение, праздник, святой покровитель) – в виде театрализованного представления;

б) блиц-опрос (географическая викторина);

в) инсценировку сказки, легенды или исторического эпизода;

г) загадки об известных людях;

д) представление традиций, национального блюда и костюма;

е) музыкальный / танцевальный конкурс.

Длительность представления каждого задания программы фестиваля-конкурса заранее оговаривалась и ограничивалась 5-7 минутами.

Далее была проведена жеребьевка среди четырех групп второго курса, чтобы определить, какая команда будет представлять ту или иную часть Великобритании. С этого момента началась непосредственная подготовка к фестивалю-конкурсу. Очень важным было то, что все студенты могли принять посильное участие в этой работе. Поскольку поисковая работа велась во внеурочное время, студенты самостоятельно распределяли между собой обязанности по выполнению определенных задач и были ответственны за их выполнение перед своими товарищами. К преподавателям они обращались, лишь сталкиваясь с той или иной трудностью. Таким образом, мы оказывали помощь студентам на основе доверия и сотрудничества, не предлагая готовых решений. Конечно, было важно проследить за логичностью, доступностью и грамматической корректностью составляемых вопросов и загадок, а также скоординировать сроки готовности и организации заданий фестиваля-конкурса. В остальном же участникам предоставлялась полная свобода в подготовке материала и способа его подачи при защите проекта. В поисках информации для фестиваля-конкурса студенты обращались к разнообразным журналам, книгам по страноведению, а также активно пользовались материалами, полученными через Интернет. Таким образом, на подготовительном этапе проектной работы совершенствовались сформированные ранее навыки иноязычного общения и закладывались основы будущих самостоятельных высказываний студентов.

Через 3-4 недели интенсивной работы каждой из групп над своим проектом был назначен день защиты. Для презентаций проектов требовалось два полных занятия (из-за сложности координации расписания в четырех группах их пришлось провести дополнительно), и местом проведения была выбрана большая аудитория.

Третий этап состоял из заключительной защиты проектов. Несомненно, самым важным и ответствен-

ным днем для всех участников фестиваля-конкурса стал день презентации подготовленных группами проектов. Представление традиций, обычаев и национальных особенностей каждой из составляющих частей Великобритании должно было быть показано экспертному жюри, состоящему из преподавателей кафедры германских языков. Волнение студентов усиливал и тот факт, что на это общекурсовое событие были приглашены все желающие, о чем извещало объявление на расписании факультета. Очередность выступления команд-участниц внутри каждого конкурса была определена жеребьевкой (по принципу проведения КВН).

Участники сами выбирали, каким образом интересно и ярко представить официальные и в целом общеизвестные факты для презентации своей части Великобритании. Для красочного и познавательного конкурса «Визитная карточка страны» были использованы и автобусная экскурсия, и урок, и посиделки в кафе. Свою эрудицию в отношении географии Британских островов соперничающие команды показали, отвечая на максимально большое количество вопросов оппонентов в течение 1 минуты. Чтобы инсценировать сказку, легенду или исторический эпизод, участники подбирали какой-то оригинальный сюжет. Однако студенты не просто заимствовали ту или иную напечатанную пьесу, а подходили к этой работе творчески: вносили коррективы, дополняли, находили соответствующие песни, разучивали танцевальные элементы. Кроме того, все участники преобразились, надев исторические и национальные костюмы. Легкоузнаваемыми здесь были, конечно, представители Шотландии в килтах и Ирландии в одежде всех оттенков зеленого, но «англичане» и «валлийцы» тоже делали все возможное, чтобы выглядеть цельными и заметными командами. Большую помощь в этом перевоплощении оказала костюмерная драматического кружка нашего университета. При оформлении сцены использовались флаги, гербы и национальные эмблемы, нарисованные самими участниками. Решая загадки об известных уроженцах разных частей Великобритании, студенты продемонстрировали свою начитанность и общий кругозор, хотя иногда формулировки загадок были действительно непростыми. Пожалуй, самым популярным конкурсом оказалось представление национальных традиций, блюда и костюма, где эти особенности Англии, Уэльса, Ирландии и Шотландии были показаны зрителям в небольших театрализованных представлениях. В последнем задании все участники проявили музыкальные и танцевальные таланты.

Жюри фестиваля-конкурса подвело итоги и определило победителей. В устных выступлениях учитывалась как содержательная сторона выступления каждой группы (логичность, полнота изложения), так и лексическое наполнение, грамматическая верность высказываний, а также фонетическое звучание, т.е. интонация, паузы, логическое ударение, беглость речи. Существенно влияло на начисление баллов также и умение участников логично и правильно реагировать на вопросы, заранее составленные соперниками. В театрализованных представлениях оценивались не только содержательные и языковые качества инсценировок, но и артистизм их исполнения, так как развитие презентационных навыков сейчас считается очень важным аспектом обучения устной речи.

В результате все представленные проекты были признаны жюри интересными, познавательными и отвечающими поставленным требованиям, однако при этом в течение двух лет чемпионами оказывались команды, представлявшие культуру и традиции Уэльса. А в целом, мы считаем, все участники вполне успешно справились с поставленной перед ними задачей – изучить самим яркие национальные особенности разных частей Великобритании и ознакомить с ними других.

Польза такого рода мероприятий несомненна, так как конкурсный характер фестиваля повышает мотивацию к освоению учебного материала и изучению английского языка вообще; помогает выработать здоровое отношение к соревнованию, раскрепощает творческие способности студентов, что далеко не всегда возможно на уроке. Особенно привлекательно то, что проектное обучение активно влияет на мотивационную сферу студентов – позволяет даже самым застенчивым поверить в себя. Совместное участие в такой необычной форме работы помогает студентам проявить себя совершенно неожиданно, у них есть возможность показать свои организаторские способности и другие скрытые таланты – художественные, театральные, аналитические, поисковые. Процесс работы над проектом стимулирует студентов быть деятельными, развивает у них воображение, творческое мышление, самостоятельность. Наличие элементов поисковой деятельности и творчества создает условия для взаимообогащающего общения как на родном, так и на иностранном языке. Очень важно также и то, что в работе над проектом студенты учатся сотрудничать, а обучение в сотрудничестве воспитывает в них такие нравственные ценности, как взаимопомощь, желание и умение успешно, активно и творчески работать вместе, уважение к мнениям других участников проекта и т. д. Кроме того, во время работы над проектом формируются новые отношения преподавателя и студентов, здесь тоже возникает благотворная атмосфера сотрудничества. Преподаватель уже не является для студентов единственным источником учебной информации, он становится для них консультантом и помощником, предоставляя студентам свободу самовыражения в рамках объявленной темы. Особым достоинством проектной методики является то, что формы работы не остаются чем-то неизменным. Каждый год творчески мыслящие студенты привносят что-то свое, новое, интересное.

Использовав в своей работе такую проектную методику, мы пришли к выводу, что она наиболее эффективна при обобщении, закреплении и повторении учебного материала, а особенно при организации его практического применения. Углубленно работая над заданной темой, студенты расширили свои страноведческие познания, развили коммуникативные умения, а также получили ценный опыт проведения внеклассных мероприятий. Усилия не были потрачены зря, так как все эти навыки в дальнейшем несомненно пригодятся нашим студентам и на экзаменах, и в образовании, и в жизни.

Следует отметить, что при такой форме работы наряду с преимуществами имеются и определенные трудности. Они обусловлены непривычностью подобной формы работы (как для студентов, так и для преподавателей), недостатком времени (по учебному плану выделяется не более 24 часов на каждый раздел), неравномерным уровнем языковой подготовки студентов в отдельных группах, отсутствием навыка выступления перед большой аудиторией. Заметно не хватало и «актерского мастерства», участникам следовало больше внимания уделять репетициям, а особенно тренировке громкости голоса и отчетливости произношения.

Несмотря на все это, как мы видим, метод проектов оказался весьма многообещающей формой работы на занятиях английского языка, так как он четко ориентирован на реальный практический результат, значимый для студента. Нам, преподавателям, нужно подкорректировать план мероприятия и требования к нему, более четко продумать организацию и ход фестиваля, например сократить количество конкурсов, чтобы сохранить динамику.

Мы планируем ввести краткий письменный опрос студентов после фестиваля-конкурса, чтобы узнать их мнение и впечатления от участия в нем. Однако уже сейчас очевидно, что таких общекурсовых мероприятий, где царит дух праздника, радости творчества и общения, должно становиться больше на каждом курсе. Поверив в свои силы, наши самостоятельные и самодеятельные студенты и дальше будут продолжать развивать и совершенствовать свои иноязычные коммуникативные умения.

Примечания

1. Практический курс английского языка. II курс: Учеб. для пед. вузов по спец. «Иностр. яз.» / Под ред. В. Д. Аракина. 5-е изд., перераб. и доп. М.: ИЦ ВЛАДОС, 1999.

2. Знакомимся с Британией: Кн. для чтения по страноведению / Авт. и сост. В. М. Павлоцкий. СПб.: Игрек М, 1996.

3. Великобритания: Лингвострановедческий словарь / Сост. Г. Д. Томахин. М.: ООО «Издательство АСТ», 2001.

СТУДЕНЧЕСКАЯ НАУКА

Р. А. Веснин

ОБ ОБРАБОТКЕ ТЕКСТОВОЙ ИНФОРМАЦИИ

Можно ли в курсе программирования решать задачи, связанные с «примитивами» обработки информации в любом текстовом процессоре? Оказывается, да. Это показано на примере простых задач, перечень которых может быть расширен.

Сформулируем проблему. Как изучать принципы обработки текстовой информации? Обычно изучается конкретный текстовый процессор, например MS Word, на уровне пользовательских навыков. Возможен другой взгляд на проблему, а именно, в курсе программирования за счет специально подобранных задач изучаются фундаментальные основы обработки текстовой информации. Статья посвящена разработке подборки (части) задач для реализации второй точки зрения.

1. Выравнивание текста

Задача:

Дан текст в файле в виде строки. Требуется отформатировать текст, выполнив

1) выравнивание по ширине;

2) выравнивание по центру;

3) выравнивание по левому краю;

4) выравнивание по правому краю;

и вывести текст в файл в виде нескольких строк (ширина строк после форматирования не должна превышать W символов). Выравнивание текста производится путем вставки пробелов. Слова переносятся на следующую строку только целиком. Слово – последовательность символов, не содержащая пробелов.

Исходный текст содержится в файле в виде строки, его требуется отформатировать и вывести в несколько строк. Рассмотрим процедуру Solve:

```
procedure SOLVE;
begin
 while READS do
  begin
  FORMATING(W);
   PRINT;
  end;
 PRINT;
end;
```

ВЕСНИН Роман Александрович – студент II курса факультета информатики ВятГГУ

Данная процедура обращается к функции READS, которая записывает в строку S (глобальная переменная) текст из файла блоками длиной по W (глобальная переменная) символов. Далее процедура FORMATING производит форматирование строки, которая затем выводится процедурой PRINT. Этот процесс повторяется до тех пор, пока функция READS возвращает значения ИСТИНА – не конец файла. В завершении выводится остаток исходного текста без форматирования.

```
function READS: boolean;
var c: char;
begin
 delete(s,1,w);
 while (length(s)>0)and(s[1]=' ') do
  delete(s,1,1);
 while (not eoln) and (length(s)<=w) do
 begin
  read(c);
   s:=s+c;
 end;
 if eoln then reads:=false
 else reads:=true;
end;
```

Функция READS удаляет первые W символов из строки, если данная функция вызывается не первый раз, и удаляет все пробелы, находящиеся в начале строки. Затем, как уже было сказано, строка S дополняется до W символов. В случае, если достигнут конец строки файла, функция принимает значение ЛОЖЬ, чтобы вывести остаток исходного текста без форматирования.

Следует отметить, так как мы работаем с текстовым режимом отображения информации, то весь анализ проводится на основе анализа символьного содержания текста без использования дополнительной (мета)информации.

1.1 Выравнивание текста по ширине

```
procedure FORMATING;
var i,j,k:integer;
begin
 j:=length(s);
 while (j>0) and (s[j]<>' ') do
  dec(j);
 dec(j);
 k:=0;
 while j<>k do
 begin
  i:=1;
  k:=j;
 while (i<j)and(j<w) do
  begin
   if s[i]=' ' then
    begin
     insert(' ',s,i);
    while (i<length(s))and(s[i]=' ') do inc(i);
     inc(j);
    end;
   inc(i);
  end;
 end;
end;
```

Логика данной процедуры довольно проста.

1. Необходимо найти позицию, с которой начинается слово, не умещающееся в строке.

2. Находим в интервале от 1 до найденного выше значения промежуток между словами и добавляем один пробел.

3. Шаг (2) повторяется до тех пор, пока последнее слово в строке не выйдет за границу W. Это слово будет перенесено в следующую строку.

1.2. Выравнивание текста по центру

```
procedure FORMATING (t:integer);
var i,j:integer;
begin
 if (t=w) then inc(t);
 j:=t;
 while (j>0) and (s[j]<>' ') do
  dec(j);
 dec(j);
 if (j<=0) then j:=t;
 for i:=1 to (w-j) div 2 do
  insert(' ',s,1);
 inc(j,i);
 for i:=j+1 to w do
  insert(' ',s,j+1);
end;
```

Процедура ищет начало слова, которое не вмещается в строку. Вся предыдущая часть строки дополняется одинаковым количеством пробелов с начала и с конца, таким образом, получается текст, выровненный по центру.

1.3. Выравнивание текста по левому краю

```
procedure FORMATING (t:integer);
var i,j:integer;
begin
 if (t=w) then inc(t);
 j:=t;
 while (j>0) and (s[j]<>' ') do
  dec(j);
 dec(j) ;
 if (j<=0) then j:=t;
 for i:=1 to (w-j) do
  insert(' ',s,j+1);
end;
```

Процедура ищет начало слова, которое не вмещается в строку. Перед этим словом добавляются пробелы так, что получается текст, выровненный по левому краю. Последнее слово будет перенесено в следующую строку.

1.4. Выравнивание текста по правому краю

```
procedure FORMATING (t:integer);
var i,j:integer;
begin
 if (t=w) then inc(t);
 j:=t;
 while (j>0) and (s[j]<>' ') do
  dec(j);
 dec(j);
 if (j<=0) then j:=t;
 for i:=1 to (w-j) do
  insert(' ',s,1);
end;
```

Процедура ищет начало слова, которое не вмещается в строку. В начало строки добавляются пробелы так, что получается текст, выровненный по правому краю. Последнее слово будет перенесено в следующую строку.

2. Формирование содержания

Задача:

Дан текст в файле в виде строк. Текст содержит заголовки разделов, глав, параграфов. Требуется составить содержание по этим заголовкам, имеющим следующий вид:

- *«$X Заголовок» – заголовок первого уровня;*
- *«$X.X Заголовок» – заголовок второго уровня и т.д.*

Перед символом "$" не должно быть никаких символов, кроме пробелов.

Рассмотрим процесс создания содержания. Предполагается, что содержание формируется из заголовков, имеющих вид:

Для хранения элементов содержания используется комбинированный тип – RECORD, содержащий заголовок, номер страницы и ссылку на следующий пункт содержания. Также используется ссылочный тип на вышеуказанный.

```
type tpnt = ^trec;
   trec = record
        header:string;
        page:integer;
        next:tpnt;
       end;
```

Процедура SOLVE выглядит следующим образом:

```
procedure SOLVE;
var i,num,num_line:integer;
   s:string;
begin
 num_line:=0;
 while not eof do
 begin
  readln(s);
  inc(num_line);
  num:=pos("$",s);
```

```
    i:=1;
    while (i<num)and(s[i]=' ") do
     inc(i);
    if i=num then ADD (s,num_line);
   end;
  item.next:=nil;
 end;
```

Поясним работу процедуры SOLVE: из файла читается строка; проверяется наличие в строке символа «$» – признак заголовка; проверяется начало строки на наличие «посторонних» символов; если строка является заголовком, то процедурой ADD формируется очередной элемент содержания.

```
 procedure ADD (s:string;lin:integer);
 var page:integer;
 begin
  new(item.next^.next);
  item.next:=item.next^.next;
  page:=lin div lpp; {lpp – определяет количество
строк на странице - константа}
  if lin mod lpp>0 then inc(page);
  item.next^.header:=s;
  item.next^.page:=page;
 end;
```

Теперь осталось реализовать вывод содержания в отформатированном виде – все элементы содержания, соответствующие заголовкам одного уровня, имеют одинаковый отступ слева. Это выполняет процедура PRINT.

```
 procedure PRINT;
 var s,p:string;
    len,i:integer;
 begin
  writeln("Содержание:");
  item.next:=first^.next;
  while item.next<>nil do
  begin
   item:=item.next^;
   s:=item.header;
   str(item.page,p);
   while s[1]=' " do delete(s,1,1);
   i:=3;
   while s[i]<>' " do
   begin
    if s[i]='.' then
     begin
      insert(" ",s,1);
      inc(i,2);
     end;
    inc(i);
   end;
   len:=length(s)+1;
   s:=s+p;
   while length(s)<w do insert(".",s,len);
   writeln(s);
  end;
 end;
```

3. Взаимное преобразование нумерованных и маркированных списков

Задача:

Дан текст в файле в виде строк. Текст содержит нумерованные/маркированные списки. Требуется преобразовать все нумерованные (маркированные) списки в маркированные (нумерованные) и записать новый текст в файл. Элемент маркированного списка имеет вид " ТЕКСТ ", элемент нумерованного списка имеет вид "X) ТЕКСТ ". Элемент списка может занимать более одной строки. Все элементы одного нумерованного/маркированного списка начинаются с новой строки и имеют одинаковый отступ слева.*

Разберем процедуру преобразования нумерованного списка в маркированный.

```
 procedure SOLVE;
 var p:integer;
 begin
  uk:=1;
  while not eof do
  begin
   readln(s);
   if (uk>1) then SRCHNEXT (p);
   SRCHFRST (p);
   writeln(s);
  end;
 end;
```

Каждая строка проверяется на наличие признака нумерованного списка. Процедура SRCHFRST ищет первый элемент списка и возвращает параметр форматирования этой строки. Процедура SRCHNEXT ищет следующие элементы нумерованного списка с данным параметром форматирования. Обе эти процедуры при положительном результате заменяют номер элемента списка на маркер.

```
 procedure SRCHFRST
       (var p:integer);
 var mkr:string;
    pp,i:integer;
 begin
  str(1,mkr);
  mkr:=mkr+')';
 end;
  if pp <> 0 then
  begin
  p:=pp;
   delete(s,p,Length(mkr));
   insert("*",s,p);
   uk:=2;
 end;
 end;
 procedure SRCHNEXT
          (p:integer);
 var mkr:string;
    pp,i:integer;
 begin
  str(uk,mkr);
  mkr:=mkr+')';
  pp:=pos(mkr,s);
  if pp>1 then
  begin
   i:=pp-1;
   while s[i]=' " do dec(i);
   if i>0 then pp:=0;
  pp:=pos(mkr,s);
  if pp>1 then
  begin
   i:=pp-1;
   while s[i]=' " do dec(i);
   if i>0 then pp:=0;
  end;
  if pp=p then
  begin
   delete(s,p,Length(mkr));
   insert("*",s,p);
   inc(uk);
  end;
 end;
```


Процедуры обратного преобразования списков подобны вышеописанным. Разница лишь в том, что эти процедуры ищут признаки маркированного списка и производят вставку нумерации элементов списка.

4. Перенос слов по слогам

Задача:

Дан текст в файле в виде строки. Требуется выполнить разбиение всего текста на строки с расстановкой переносов в словах так, чтобы длина каждой строки не превышала W символов. Расстановку переносов в словах следует выполнять по следующему правилу: перенос ставится между сочетаниями букв «гг», «гс», «сс» и «зс», где «г» – гласная буква, «с» – согласная буква, «з» – все остальные символы. Слово должно содержать не менее одной гласной буквы до и после переноса. Слово – последовательность символов, не содержащая пробелов.

```
 procedure SOLVE;
 var s,wd:string;
    p,pp:integer;
 begin
  readln(s);
  while length(s)>w do
  begin
   GETWORD (s,wd,p);
   if (wd='') then pp:=0
   else INSRMV (wd,w-p,pp);
   wd:=copy(s,1,p+pp);
   if (pp>0) then wd:=wd+'-';
   writeln(wd);
   delete(s,1,p+pp);
  end;
  writeln(s);
 end;
```

Задача состоит в том, чтобы вывести строку S в файл в несколько строк по указанной ширине с переносом слов. Процедура GETWORD возвращает слово, которое находится в конце и должно быть перенесено, и позицию, с которой начинается слово. Процедура INSRMV вставляет в слово перенос не далее, чем в указанной позиции. Затем в файл выводится очередная строка.

```
 procedure GETWORD (s:string;var wd:string;var
p:integer);
 var j:integer;
 begin
  wd:='';
  p:=w;
  while s[p]<>' " do dec(p);
  j:=w;
  while s[j]<>' " do inc(j);
  if j=w+1 then p:=j
  else wd:=copy(s,p+1,j-p-1);
 end;
 function TIP (c:char):byte;
 begin
  if c in gl then tip:=0
  else if c in sgl then tip:=1
   else tip:=2;
 end;
 procedure INSRMV (wd:string;p:integer;var
pp:integer);
 var len,af,bf,t,i,pair:integer;
 begin
  len:=length(wd);
  if tip(wd[len])=2 then dec(len);
  pp:=0;
  while len-p<1 do dec(p);
  if (len>=4)and(p>=2) then
  begin
   bf:=0;
   for i:=p downto 1 do
    if (wd[i] in gl) then inc(bf);
   af:=0;
   for i:=p+1 to len do
    if (wd[i] in gl) then inc(af);
   if (af+bf>1)and(bf>0) then
   begin
    i:=p-1;
    pair:=tip(wd[p]);
    repeat
     pair:=(pair * 10)mod 100;
     t:=tip(wd[i]);
     if (pair div 10=0)then
      begin
       inc(af);
       dec(bf);
      end;
     pair:=pair + t;
    if (pair in [0,10,11,12])and(af*bf>0)then pp:=i;
     dec(i);
   until (i<2)or(pp<>0);
   end;
  end
  else pp:=-p+1;
 end;
```

Для определения типа символа (гласная/согласная буква или прочие символы) используется функция TIP путем проверки принадлежности символа множествам (множество гласных букв / множество согласных букв), которые инициализируются перед вызовом процедуры SOLVE.

5. Замена текста

Задача:

Дан текст в файле в виде строк, выровненный по ширине/по центру/по левому краю/по правому краю. Требуется выполнить поиск и замену всех вхождений строки S1 строкой S2 в тексте без нарушения выравнивания текста.

При замене текста возможно нарушение форматирования текста. Для сохранения вида выравнивания текста рассмотрим процедуру определения параметров форматирования текста при условии, что весь текст имеет одинаковые параметры форматирования.

```
procedure REPLACE;
var i,j:integer;
begin
 llen:=length(s1);
 i:=1;
 while i<=len-llen do
 begin
  j:=0;
 while (j<llen)and(s[i+j]=s1[j+1]) do inc(j);
  if j=llen then
  begin
   delete(s,i,llen);
   insert(s2,s,i);
   len:=len-llen+length(s2);
   inc(i,length(s2));
   end;
  inc(i);
 end;
end;
```

Процедура REPLACE заменяет текст в глобальной переменной S. S1 – текст, который будет заменен; S2 – текст, на который заменяется искомый текст.

```
procedure SOLVE;
var j,i,n,m:integer;
begin
 len:=0;
 while reads do begin
  replace;
  case typ of
   0:format_c(w);
   1:format_l(w);
   2:format_r(w);
   3:format_j(w);
  end;
  out;
end;
```

```
  replace;
  case typ of
   0:format_c(len);
   1:format_l(len);
   2:format_r(len);
   3:format_l(len);
  end;
  out;
end;
```

Пока конец файла не достигнут, процедура *SOLVE* читает из файла строку (процедура READS), производит поиск и замену текста с помощью процедуры REPLACE и форматирует текст согласно начальным параметрам форматирования текста.

Теперь рассмотрим процедуру определения параметров форматирования текста. Принцип работы процедуры основан на статистическом анализе, что позволяет достаточно точно определить параметры первоначального отображения текста.

```
 lc:=0;
 rc:=0;
 wc:=0;
fillchar(wp,sizeof(wp),0);
fillchar(lp,sizeof(lp),0);

fillchar(rp,sizeof(rp),0);
```

```
  if i<=len then
  begin
   fld:=i;
   i:=1;

   while (i<=lc)and
   (fld<>lp[i].val) do inc(i);
  if i>lc then lc:=i;
 while (not eof) do
 begin

  readln(s);
  len:=length(s);


  fld:=len;
  i:=1;

 while (i<=wc)and
    (fld<>wp[i].val)
          do inc(i);
 if i>wc then wc:=i;
 wp[i].val:=fld;
inc(wp[i].num);

 i:=1;
 while (i<=len) and
   (s[i]=' ') do inc(i);
```

```
  if i>lc then lc:=i;
 lp[i].val:=fld;
inc(lp[i].num);

  i:=len;
  while  (i>0) and
      (s[i]=' ') do dec(i);
 fld:=i;
 i:=1;
while (i<=rc)and
     (fld<>rp[i].val)
           do inc(i);
if i>rc then rc:=i;


rp[i].val:=fld;
inc(rp[i].num);

 end;

 end;
```

Данный фрагмент программы реализует «сбор сведений» о форматировании текста. Переменные lp, rp, wp имеют тип record, содержат поля val и num, где val – значение параметра, num – количество объектов с данным значением, lp содержит информацию об отступах слева, rp содержит информацию об отступах справа, wp содержит информацию о ширине строк. Переменные lc, rc и wc определяют количество различных значений для соответствующих параметров. Теперь необходимо провести анализ данных и определить тип форматирования текста после замены.

```
 typ:=0;
 max:=0;
 imax:=0;
 cnt:=0;

if lc<=2 then
begin
  typ:=typ+1;
  if lc=1 then sh:=0;
  else sh:=lp[1].val;

end;
```

```
 for i:=1 to rc do
  begin
  inc(cnt,r[i].num);
   if max<rp[i].val then
    begin
     max:=rp[i].val;
     imax:=i;
    end;
 w:=max;
   if (rp[ind].num*2>=
   fld) then  typ:=typ+2;
```

Переменная w содержит значение ширины текста, sh задает отступ красной строки. Значение переменной typ определяет выравнивание текста (см. табл.).

Значение typ	0	1	2	3
Выравнивание	По центру	По левому краю	По правому краю	По ширине

6. Статистика документа

Задача:

Дан текст в файле в виде строк. Требуется подсчитать количество страниц, строк, слов, знаков (без пробелов), знаков (с пробелами) в тексте. Слово – последовательность символов, не содержащая пробелов.

```
procedure SOLVE;
begin
 pge:=0;
 wrd:=0;
 sim:=0;
 sps:=0;
 lin:=-1;
 while not eof do
 begin
  readln(s);
  sp:=pos(" ",s);
  inc(lin);
   s:=s+' ';
  while (length(s)>0) do
   begin
    sp:=pos(" ",s);
    inc(sps);
    if sp>1 then inc(wrd);
    inc(sim,sp-1);
    delete(s,1,sp);
   end;
  dec(sps);
 end;
 pge:=lin div lpp + 1;
 inc(lin);
end;
```

7. Преобразование кодировки текста

Задача:

Дан текст в файле. Требуется выполнить преобразование текста из кодировки ASCII (DOS) в кодировку ANSI (WINDOWS) и обратно.

Преобразование кодировки текста из DOS в WINDOWS и обратно происходит путем замены кодов символов одной кодировки кодами соответствующих символов другой кодировки.

Преобразование текста из DOS в WINDOWS:

```
procedure SOLVE;
var i:integer;
   s:string;
begin
 while not eof do
 begin
  readln(s);
  for i:=1 to length(s) do
   if s[i] in [#128..#175] then
    s[i]:=chr(ord(s[i])+64)
    else
     if s[i] in [#224..#239] then
      s[i]:=chr(ord(s[i])+16);
  writeln(s);
 end;
end;
```

Преобразование текста из WINDOWS в DOS:

```
procedure SOLVE;
var i:integer;
   s:string;
begin
 while not eof do
 begin
  readln(s);
  for i:=1 to length(s) do
   if s[i] in [#192..#239] then
    s[i]:=chr(ord(s[i])-64)
    else
     if s[i] in [#240..#255] then
      s[i]:=chr(ord(s[i])-16);
  writeln(s);
 end;
end;
```

Р. А. Веснин

ЗАДАЧА ОБ ИЗОМОРФИЗМЕ ГРАФОВ (ПОПЫТКА РЕШЕНИЯ)

В статье рассмотрена классическая, достаточно сложная задача. Попытка её приближенного решения – демонстрация принципа неисчерпаемости любой содержательной задачи.

Задача:

Даны два графа (ориентированные или неориентированные). Требуется определить, являются ли эти графы изоморфными. (Два графа считаются изоморфными, если они одинаковы и различаются лишь нумерацией вершин.)

Прежде всего отметим, что задачу следует решать для графов, в которых одинаковое количество вершин и ребер, в противном случае результат решения заранее будет отрицательным.

Данный метод решения задачи основан на приведении матриц смежности графов к определенному «состоянию», после чего равенство матриц говорит об изоморфизме данных графов.

Чтобы подойти к конечному «состоянию» матрицы смежности, в ней проводится три последовательные сортировки:

ВЕСНИН Роман Александрович – студент II курса факультета информатики ВятГТУ

1. Первая сортировка в матрице смежности позволяет объединить в группы вершины по возрастанию количества исходящих ребер. Теперь каждая вершина принадлежит своей группе и не может быть помещена за ее пределы.

2. Вторая сортировка устанавливает отношение порядка между вершинами внутри каждой группы за счет их связей со смежными вершинами.

3. Последняя сортировка позволяет определить окончательное соответствие между вершинами, если таковое не было достигнуто первыми двумя пунктами.

Чтобы данный метод мог быть применен без опасений к ориентированным графам, каждое ребро между вершинами графа дополняется обратным («мнимым») в случае его отсутствия. В матрице смежности такое ребро обозначается "–1", чтобы его можно было отличить от действительных ребер.

Для реализации потребуются следующие структуры:

```
const nmax=50;
type Tstr = string[nmax];
   Trow = record
          num:byte;
          cnt:byte;
          lin:Tstr;
         row:array[1..nmax] of shortint;
        end;
  Tgr = array[1..nmax] of trow;

Var A:array[1..2] of tgr;
  n:byte;
```

Тип Trow содержит поля: num – номер вершины в исходном графе; cnt – количество ребер, исходящих из вершины; lin – характеризует вершины, смежные данной; row – строка матрицы смежности, показывающая исходящие ребра из вершины. Тип Tgr полностью описывает граф и его вершины.

Массив A содержит две матрицы смежности сравниваемых графов, n – количество вершин в каждом графе.

Процедура Init производит ввод данных из файла с подсчетом количества исходящих ребер для каждой вершины.

```
procedure init;
var j,i,m:integer;
begin
 fillchar(a,sizeof(a),0);
 readln(n);
 for m:=1 to 2 do
  for i:=1 to n do
   begin
    while not eoln do
    begin
     read(j);
     a[m][i].row[j]:=1;
     inc(a[m][i].cnt)
     end;
    a[m][i].num:=i;
    readln;
   end;
end;
```

Процедура Addition дополняет граф до неориентированного, если это требуется, производит пересчет количества смежных вершин и заменяет единицы в матрице смежности на числа, равные количеству смежных вершин у той вершины, в которую идет ребро из данной вершины. Знак "–" у «мнимых» ребер сохраняется.

```
procedure Addition;
var m,i,j:integer;
begin
 for m:=1 to 2 do
  begin
   for j:=1 to n do
    for i:=j to n do
    if abs(a[m][i].row[j])<>abs(a[m][j].row[i]) then
     if a[m][i].row[j]=0 then
       begin
       inc(a[m][i].cnt);
       a[m][i].row[j]:=-1;
       end
       else
       begin
      inc(a[m][j].cnt);
      a[m][j].row[i]:=-1;
      end;
   for j:=1 to n do
    for i:=1 to n do
    a[m][i].row[j]:=a[m][i].row[j]*a[m][j].cnt;
  end;
end;
```

Функция Lin в графе с номером m для вершины i подсчитывает ее характеристику, состоящую из элементов строки матрицы смежности без нулей. Выбор строкового типа для lin объясняется тем, что строковые переменные имеют свои особенности сравнения и установления отношения порядка.

```
function Lin(m,i:byte):tstr;
var j:byte;
   st:tstr;
begin
 st:='';
 for j:=1 to n do
  if a[m][i].row[j]<>0 then
   st:=st+chr(abs(a[m][i].row[j]));
 lin:=st;
end;
```


Функция Less в графе m для вершин с номерами r1 и r2 с одинаковыми характеристиками lin устанавливает отношение порядка по строкам матрицы смежности.

```
function Less(m,r1,r2:byte):boolean;
var i,sh:byte;
   ok:boolean;
begin
 i:=0;
 repeat
  inc(i);
  if (i=r2)and(a[m][r1].row[i]=a[m][r2].row[i+1])
                                  then inc(i,2);
 until (i>n)or (a[m][r1].row[i]<>a[m][r2].row[i]);
 if i=r2 then sh:=1 else sh:=0;
 if i<=n then ok:=a[m][r1].row[i]>
                          a[m][r2].row[i+sh]
 else ok:=false;
 less:=ok;
end;
```

Процедура Transp в графе m выполняет перенумерацию вершин с номерами r1 и r2. Для равноценного преобразования необходимо поменять местами строки и столбцы с данными номерами.

```
procedure Transp(m,r1,r2:byte);
var i,s:shortint;
   w:trow;
begin
 w:=a[m][r1];
 a[m][r1]:=a[m][r2];
 a[m][r2]:=w;
 for i:=1 to n do
 begin
  s:=a[m][i].row[r1];
  a[m][i].row[r1]:=a[m][i].row[r2];
  a[m][i].row[r2]:=s;
 end;
end;
```

Процедура Sort выполняет три вышеобозначенных пункта сортировки в матрицах смежности сравниваемых графов: первая сортировка по значению cnt; вторая сортировка по значению lin; третья сортировка по возвращаемому значению функции less.

```
procedure Sort;
var m,i,j,s,e:byte;
begin
 for m:=1 to 2 do
 begin

 {1-й блок}
  for i:=n downto 2 do
   for j:=2 to i do
    if a[m][j].cnt<a[m][j-1].cnt then
      transp(m,j,j-1);
  for i:=1 to n do
   a[m][i].lin:=lin(m,i);

 {2-й блок}
  e:=1;
  repeat
   s:=e;
  while a[m][e].cnt=a[m][s].cnt do
   inc(e);
  for i:=e-1 downto s+1 do
   for j:=s+1 to i do
   if a[m][j].lin<a[m][j-1].lin then
     transp(m,j-1,j);
  until e>n;

 {3-й блок}
  for i:=n downto 2 do
   for j:=2 to i do
   if a[m][j].lin=a[m][j-1].lin then
    if less(m,j,j-1) then
     transp(m,j-1,j);
 end;
end;
```

Процедура сравнения матриц смежности по завершению сортировки. В случае изоморфизма графов выводится соответствие вершин первого и второго графов.

```
procedure Compare;
var i,j:integer;   ok:boolean;
begin
 ok:=true;
 for i:=1 to n do
  for j:=1 to n do
   if a[1][i].row[j]<>a[2][i].row[j] then ok:=false;
 if ok then
  for i:=1 to n do
   writeln(a[1][i].num,'- ',a[2][i].num)
 else writeln("Not possible");
end;
Процедура Solve :
procedure Solve;
begin
 addition;
 sort;
 compare;
end;
```

Таким образом, получается, что временная сложность алгоритма пропорциональна N^3.

Рассмотрим пример (рис. 1, 2):

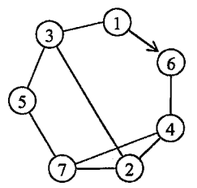


Рис. 1. Ребро 1-6 является направленным (граф 1)

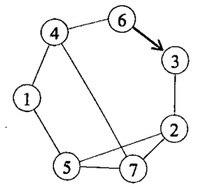


Рис. 2. Ребро 6-3 является направленным (граф 2)

Начальное содержимое матрицы смежности

num	1	2	3	4	5	6	7	cnt	lin
1			1			1		2	
2			1	1			1	3	
3	1	1			1			3	
4		1				1	1	3	
5			1				1	2	
6				1				1	
7		1		1	1			3	

Матрица смежности после выполнения процедуры Addition

num	1	2	3	4	5	6	7	cnt	lin
1			3			2		2	
2			3	3			3	3	
3	2	3			2			3	
4		3				2	3	3	
5			3				3	2	
6	-2			3				2	
7		3		3	2			3	

Матрица смежности после выполнения первого блока сортировки

num	1	5	6	4	2	3	7	cnt	lin
1			2			3		2	23
5						3	3	2	33
6	-2			3				2	23
4			2		3		3	3	233
2				3		3	3	3	333
3	2	2			3			3	223
7		2		3	3			3	233

Матрица смежности после выполнения второго блока сортировки

num	1	6	5	3	7	4	2	cnt	Lin
1		2		3				2	23
6	-2					3		2	23
5				3	3			2	33
3	2		2				3	3	223
7			2			3	3	3	233
4		2			3		3	3	233
2				3	3	3		3	333

Матрица смежности после выполнения третьего блока сортировки

num	1	6	5	3	4	7	2	cnt	lin
1		2		3				2	23
6	-2				3			2	23
5				3		3		2	33
3	2		2				3	3	223
4		2				3	3	3	233
7			2		3		3	3	233
2				3	3	3		3	333

Начальное содержимое матрицы смежности

num	1	2	3	4	5	6	7	cnt	lin
1				1	1			2	
2			1		1		1	3	
3		1						1	
4	1					1	1	3	
5	1	1					1	3	
6			1	1				2	
7		1		1	1			3	

Матрица смежности после выполнения процедуры Addition

num	1	2	3	4	5	6	7	cnt	lin
1				3	3			2	
2			2		3		3	3	
3		3				-2		2	
4	2					2	3	3	
5	2	3					3	3	
6			2	3				2	
7		3		3	3			3	

Матрица смежности после выполнения первого блока сортировки

num	1	6	3	4	5	2	7	cnt	lin
1				3	3			2	33
6			2	3				2	23
3		-2				3		2	23
4	2	2					3	3	223
5	2					3	3	3	233
2			2		3		3	3	233
7				3	3	3		3	333

Матрица смежности после выполнения второго блока сортировки

num	3	6	1	4	5	2	7	cnt	lin
3		-2				3		2	23
6	2			3				2	23
1				3	3			2	33
4		2	2				3	3	223
5			2			3	3	3	233
2	2				3		3	3	233
7				3	3	3		3	333

Матрица смежности после выполнения третьего блока сортировки

num	6	3	1	4	2	5	7	cnt	lin
6		2		3				2	23
3	-2				3			2	23
1				3		3		2	33
4	2		2				3	3	223
2		2				3	3	3	233
5			2		3		3	3	233
7				3	3	3		3	333

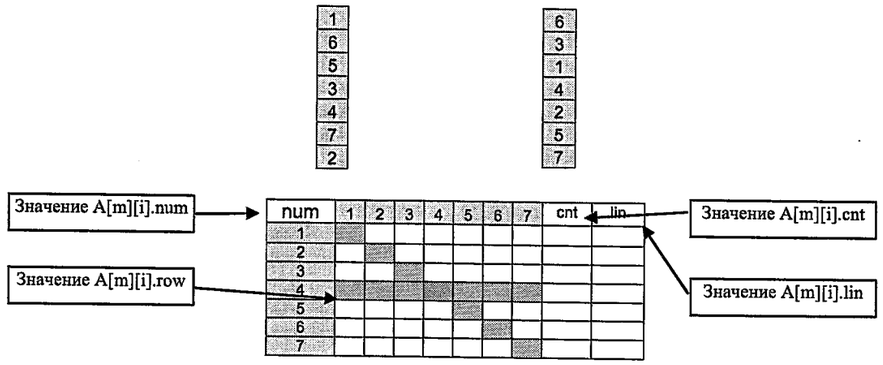


Рис. 3. Сопоставление вершин
(где m – номер матрицы смежности A, i – номер строки в матрице смежности A)

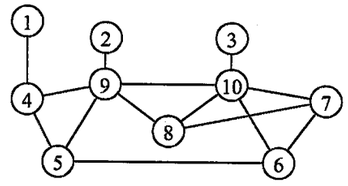


Рис. 4

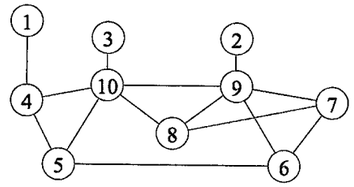


Рис. 5

num	1	2	3	4	5	6	7	8	9	10	cnt	lin
1				3							1	3
2									5		1	5
3										5	1	5
4	1				3				5		3	135
5				3		3			5		3	335
6					3		3			5	3	335
7						3		3		5	3	335
8							3		5	5	3	355
9		1		3	3			3		5	5	13335
10			1			3	3	3	5		5	13335

num	1	2	3	4	5	6	7	8	9	10	cnt	lin
1				3								3
2									5			5
3										5		5
4	1				3					5	1	135
5				3		3				5		335
6					3		3		5			335
7						3		3	5			335
8							3		5	5		355
9		1				3	3	3		5		13335
10			1	3	3			3	5			13335

На основании совпадения матриц смежности первого и второго графов можем привести следующее сопоставление вершин (см. рис. 3).

Однако этот алгоритм для некоторых графов может давать неверный результат. Для графов, являющихся неизоморфными, данный алгоритм этот факт определяет всегда. Но для некоторых графов существует зависимость определения их изоморфности друг другу от нумерации вершин. Пример такого графа приведен на рис. 4, 5.

Обе матрицы смежности удовлетворяют условиям сортировки, но тем не менее они не совпадают. Это объясняется тем, что каждый из данных графов обладает элементами симметрии – две пары вершин (2,9) и (3,10) с их «окружением». Вершины (2) и (3) имеют одинаковые характеристики lin и имеют смежные ребра соответственно с вершинами (9) и (10), также с равными характеристиками lin. Отсюда вытекает неоднозначность данного метода решения задачи.

В решении этой проблемы поможет модификация функции Less, которая, прежде чем окончательно сравнить строки 2 и 3 в матрице смежности, сравнит строки 9 и 10 без учета первых единиц применительно к вышеприведенному примеру. Таким образом, матрица смежности второго графа будет изменена и совпадет с матрицей первого графа.

```
function Less(m,r1,r2,skip:byte):boolean;
var i,sh:byte;
   ok:boolean;
begin
 i:=0;
 repeat
  inc(i);
  if (i=r2)and(a[m][r1].row[i]=a[m][r2].row[i+1])
                                   then inc(i,2);
  if i=skip then inc(i,2);
  if i=skip+1 then inc(i);
 until (i>n) or (a[m][r1].row[i]<>a[m][r2].row[i]);
 if i=r2 then sh:=1 else sh:=0;
 if i>n then ok:=false
 else
  if a[m][r1].row[i]=0 then
   if (sh=0)and(a[m][i].lin=a[m][i+1].lin) then
    ok:=Less(m,i+1,i,r2)
    else ok:=false
   else
    if a[m][r2].row[i]=0 then
     if (sh=0)and(a[m][i].lin=a[m][i+1].lin) then
      ok:=not Less(m,i+1,i,r2)
      else ok:=true
    else ok:=a[m][r1].row[i]>a[m][r2].row[i+sh];
 less:=ok;
end;
```

При вызове функции Less из процедуры Sort параметр skip может быть произвольным, но не должен входить в диапазон значений [0..n].

Ю. К. Дудова

ОПТИМАЛЬНОЕ ДВОИЧНОЕ КОДИРОВАНИЕ И ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

В статье исследуется вопрос построения оптимальной схемы двоичного алфавитного кодирования, известного как кодирование Хаффмана.

Изучая историю появления алгоритма Хаффмана (1952) и обоснование того, что схема кодов, полученных по алгоритму Хаффмана, оптимальна, мы заметили, что вначале появился алгоритм Хаффмана, затем теоретиками были установлены свойства оптимальных кодов (если они вообще существуют), а уж потом то, что коды Хаффмана удовлетворяют этим условиям. Поскольку кодирование Хаффмана оптимально, постольку вполне естественен вопрос о постановке задачи на нахождения оптимума (минимума) некоторой целевой функции при соответствующих условиях. Задачи такого типа относятся к исследованию операций.

Сначала формализуем задачу нахождения оптимальной схемы двоичного алфавитного кодирования.

Пусть подлежат кодированию символы алфавита $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ словами алфавита $B = \{0, 1\}$. Известно распределение вероятностей появления каждого символа в сообщении – $p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_n)$ соответственно. Будем Σ называть схемой кодирования, если заданы соответствия:

$$\alpha_1 \to b_1,\ \alpha_2 \to b_2, \ldots,\ \alpha_n \to b_n,$$

где b_i – двоичные слова, длина которых $l_i = l(b_i)$. Будем считать, что с помощью предложенной схемы можно закодировать любое слово в алфавите A (обозначим множество всех таких слов через $S(A)$). Обозначим $S(B)$ – образ множества $S(A)$ (множество всех двоичных кодов слов из A, полученных кодированием по схеме Σ). Если схема Σ обладает свойством взаимной однозначности (например, свойством префиксности), то возможно декодирование, то есть по коду $b \in S(B)$ однозначно восстанавливается слово $a \in S(A)$.

Введем основную характеристику схемы кодирования Σ. Обозначим через $l_{ср}(\Sigma)$ среднее взвешенное по вероятности (математическое ожидание) количество двоичных символов, приходящихся при кодировании по схеме S на один символ алфавита A:

$$l_{ср}(\Sigma) = p_1 \cdot l_1 + \ldots + p_n \cdot l_n.$$

Тогда можно поставить вопрос о нахождении оптимальной схемы кодирования Σ^*, то есть такой, в которой

$$l_{ср}(\Sigma^*) = min\{l_{ср}(\Sigma)\}: \text{ по всем схемам } \Sigma\}. \quad (1)$$

Качество (в отношении близости к оптимальной) любой схемы кодирования Σ характеризует отношение $\frac{I}{l_{ср}(\Sigma)}$. Из установленного К. Шенноном свойства любого двоичного кодирования следует, что

$$\frac{I}{l_{ср}(\Sigma)} \le 1.$$

Исторически первой схемой кодирования, для которой величина $\frac{I}{l_{ср}(\Sigma)}$ довольно близка к единице, была схема Шеннона-Фано. Однако она не оптимальна.

Нами уже была сформулирована оптимизационная задача (1). Поиск минимума в ней ведется по всевозможным схемам Σ, поэтому трудно описать условие принадлежности схемы кодирования к тем, среди которых в действительности надо отыскивать минимум. Выделение среди всевозможных схем двоичного кодирования Σ, их подмножеств, которому принадлежит оптимальная схема кодирования Σ^*, можно сделать, опираясь на неравенство Макмиллана.

ДУДОВА Юлия Константиновна – студентка V курса факультета информатики ВятГГУ

Неравенство Макмиллана. Для любой схемы Σ двоичного кодирования, обладающей свойством взаимной однозначности, справедливо неравенство

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \ldots + \frac{1}{2^{l_n}} \le 1.$$

Изучение свойств оптимальной схемы кодирования и алгоритма Хаффмана приводит к тому, что на оптимальной схеме Σ^* неравенство Макмиллана принимает вид равенства. Таким образом, можно сформулировать оптимизационную задачу для нахождения Σ^* так:

$$\text{найти} \to min\{l_{ср}(\Sigma) = p_1 \cdot l_1 + \ldots + p_n \cdot l_n:$$
$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \ldots + \frac{1}{2^{l_n}} = 1\}.$$

Словесная формулировка поставленной задачи: «Найти минимум функции $l_{ср}(\Sigma) = p_1 \cdot l_1 + \ldots + p_n \cdot l_n$ при условии $\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \ldots + \frac{1}{2^{l_n}} = 1$». Так сформулированная задача очень напоминает задачу динамического программирования – распределение ограниченного ресурса. Главное отличие – в нашей задаче функция цели линейна, а условие нелинейно (в задаче распределения ресурса – наоборот). В целях максимального приближения формулировок обеих задач введем новые переменные $x_i = \frac{1}{2^{l_i}}$, $i=1, \ldots, n$. Тогда $l_i = -\log_2 x_i = \log_2 \frac{1}{x_i}$, а оптимизационная задача для нахождения Σ^* формулируется так:

$$\text{найти} \to min\{l_{ср}(\Sigma) =$$
$$= -p_1 \cdot \log_2 x_1 - \ldots - p_n \cdot \log_2 x_n : x_1 + \ldots + x_n = 1\}. \quad (2)$$

Сейчас задача распределения ресурса (минимизация функции цели) и задача оптимального кодирования по форме совпадают за одним важным исключением – в задаче оптимального кодирования переменные x_i могут принимать только дискретные значения (степени $\frac{1}{2}$). Решение поставленной задачи методом динамического программирования при непрерывных x_i приводит к хорошо известному решению – формуле К. Шеннона для представления количества средней информации, приходящейся на один символ рассматриваемого алфавита A:

$$I = H = -p_1 \cdot \log_2 p_1 - p_2 \cdot \log_2 p_2 - \ldots - p_n \cdot \log_2 p_n.$$

Поскольку множество, на котором ищется минимум в задаче (2), содержится в множестве, на котором в качестве минимума получена формула К.Шеннона, то, очевидно, имеем:

$$I = H \le min\{l_{ср}(\Sigma)\} = l_{ср}(\Sigma^*) \le l_{ср}(\Sigma).$$

Последнее неравенство известно как теорема К. Шеннона. Поскольку в то время, когда была доказана эта теорема, еще не было известно, что существует схема оптимального кодирования, постольку ее можно было бы формулировать так: «Если существует оптимальная схема кодирования, то среднее количество двоичных разрядов в ней не меньше, чем количество энтропии (информации), приходящееся в среднем на один символ кодируемого алфавита A». Мы получили этот же результат как бы между прочим.

Решение задачи (2), в силу дискретности множества допустимых значений переменных x_i, не может в точности повторять схему решения аналогичной задачи с непрерывным изменением этих переменных (методом динамического программирования). Это, в первую очередь, связано с тем, что не существует значений целевых функций в промежуточных точках (между допустимыми значениями). Во-вторых, не ясен вопрос, с какого минимального значения $\frac{1}{2^m}$ начинать построение таблицы решений (функция $\log_2 x$ не определена в точке $x = 0$).

Введем обычно принятые в динамическом программировании обозначения. Пусть функция цели, подлежащая минимизации, имеет вид:

$$F(x_1, x_2, \ldots, x_n) = \varphi_1(x_1) + \varphi_2(x_2) + \ldots + \varphi_n(x_n),$$
$$\text{при } \varphi_1(x_1) = -p_1 \cdot \log_2 x_1,\ \varphi_2(x_2) =$$
$$-p_2 \cdot \log_2 x_2, \ldots, \varphi_n(x_n) = -p_n \cdot \log_2 x_n.$$

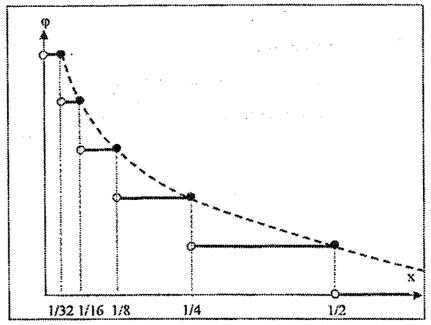
Выберем шаг $h = \frac{1}{2^m}$ таблицы решений в динамическом программировании. Определим множество допустимых решений $\{\frac{1}{2^m}, \frac{2}{2^m}, \frac{2^2}{2^m}, \ldots, \frac{1}{2}\}$ для каждой переменной x_i. Поскольку целевые функции $\varphi_j(x_j)$ определены только на множестве допустимых значений (дискретно), постольку доопределим их на промежуточных точках следующим образом: разобъем отрезок $[\frac{1}{2^m}; \frac{1}{2}]$ на частичные промежутки

$$[\frac{1}{2^m}], [\frac{1}{2^{m-1}}], (\frac{1}{2^{m-1}}; \frac{1}{2^{m-2}}], (\frac{1}{2^{m-2}}; \frac{1}{2^{m-3}}], \ldots, (\frac{1}{4}; \frac{1}{2}],$$

а значения функций $\varphi_j(x_j)$ на каждом частичном промежутке положим равными значениям на правом конце промежутка. Таким образом, каждой целевой функции $\varphi_j(x_j)$ поставили в соответствие некоторую ступенчатую функцию, которую в дальнейшем будем обозначать так же.

Замечание. Первые два подынтервала содержат всего по одному значению.

Покажем, какой вид имеют введенные описанным способом ступенчатые функции.



На рисунке изображен график ступенчатой функции

$$\varphi(x)=0{,}5\cdot\log_2 x,$$

причем начальное значение $\frac{1}{2^m}=\frac{1}{32}$. Штриховой линией показан график этой же функции при непрерывном изменении переменной x.

Далее, в соответствии с методом решения такого типа задач (распределение ресурса) строится таблица (см. табл. 1):

Таблица 1

X	$\varphi_1(x)$	$F_1(x)$	$\varphi_2(x)$	$F_2(x)$	$\varphi_3(x)$	$F_3(x)$	$\varphi_4(x)$	$F_4(x)$
X_1								
X_2								
x_n								

В табл. 1 в соответствии с выбранным шагом $h=\frac{1}{2^m}$ вычисляются значения узловых точек $x_j=j*h$, $j=1, 2, \ldots, n;\ n=2^{m-1}$. Затем заполняются столбцы со значениями ступенчатых функций $\varphi_i(x)$ во всех узлах. Столбец для функции $F_j(x)$ совпадает со столбцом для $j_1(x)$. Столбцы для функций $F_i(x)$ при $i>1$ вычисляются по рекуррентным формулам:

$$F_i(x)=min[\varphi_i(y)+F_{i-1}(x-y+h)],\ i=2, \ldots, k,$$

причем минимум ищется по y ($h \le y \le x$).

Пример 1

Дано распределение вероятностей появления символов в некотором тексте: $p_1=0{,}3$, $p_2=0{,}20$, $p_3=0{,}25$, $p_4=0{,}1$, $p_5=0{,}1$, $p_6=0{,}05$.

В результате прямого хода построена табл. 2. Величины, найденные в обратном ходе, в таблице выделены подчеркиванием.

В результате получили следующую схему кодирования: 1-й символ – 11, 2-й символ – 10, 3-й символ – 00, 4-й символ – 010, 5-й символ – 0111, 6-й символ – 0110.

Полученные коды совпадают с кодами Хаффмена.

Среднее количество двоичных разрядов, приходящееся на один символ:

4*(0,05+0,1)+3*0,1+2*(0,2+0,25+0,3)=2,4

Пример 2

Дано распределение вероятностей появления символов в некотором тексте: $p_1=0{,}2$, $p_2=0{,}2$, $p_3=0{,}2$, $p_4=0{,}2$, $p_5=0{,}2$

В результате прямого хода построена табл. 3. Величины, найденные в обратном ходе, в таблице выделены подчеркиванием.

Получили следующую схему кодирования: 1-й символ – 11, 2-й символ – 10, 3-й символ – 00, 4-й символ – 010, 5-й символ – 011.

Таблица 2

X	φ_1	F_1	φ_2	F_2	φ_3	F_3	φ_4	F_4	φ_5	F_5	φ_6	F_6
1/16	1.20	1.20	0.80	2.00	1.00	3.00	*0.40*	3.40	*0.40*	3.80	*0.20*	4.00
2/16	0.90	0.90	*0.60*	1.70	0.75	2.70	0.30	3.10	0.30	3.50	0.15	3.70
3/16	0.60	0.60	0.40	1.40	*0.50*	2.40	0.20	2.80	0.20	3.20	0.10	3.40
4/16	0.60	0.60	0.40	1.20	0.50	2.15	0.20	2.55	0.20	2.95	0.10	3.15
5/16	0.30	*0.30*	0.20	1.00	0.25	1.90	0.10	2.30	0.10	2.70	0.05	2.90
6/16	0.30	0.30	0.20	*0.90*	0.25	1.70	0.10	2.10	0.10	2.50	0.05	2.70
7/16	0.30	0.30	0.20	0.70	0.25	1.50	0.10	1.90	0.10	2.30	0.05	2.50
8/16	0.30	0.30	0.20	0.70	0.25	*1.40*	0.10	*1.80*	0.10	*2.20*	0.05	*2.40*

Результаты рассмотренных примеров показывают, что предложенный метод дает оптимальные коды. Однако в ходе эксперимента выявились некоторые обстоятельства, требующие дополнительных исследований.

Таблица 3

X	φ_1	F_1	φ_2	F_2	φ_3	F_3	φ_4	F_4	φ_5	F_5
1/8	0.6	0.6	0.6	1.2	*0.6*	1.8	*0.6*	2.4	*0.6*	3.0
2/8	0.4	0.4	*0.4*	1.0	0.4	1.6	0.4	2.2	0.4	2.8
3/8	0.2	*0.2*	0.2	0.8	0.2	1.4	0.2	2.0	0.2	2.6
4/8	0.2	0.2	0.2	*0.6*	0.2	*1.2*	0.2	*1.8*	0.2	*2.4*

Примечания

1. *Новиков Ф. А.* Дискретная математика для программистов. СПб.: Питер, 2001.
2. *Онегов В. А.* Исследование операций: задачи, методы, алгоритмы. Киров: Изд-во ВГГУ, 2001.
3. *Яблонский С. В.* Введение в дискретную математику. М.: Высш. шк., 2001.

И. А. Жалдак

ЗАДАЧА РАСПОЗНАВАНИЯ И БУЛЕВЫ УРАВНЕНИЯ

Решение многих задач, связанных с распознаванием объектов, может быть сведено к нахождению решений булевых уравнений с одним (или более) неизвестным. В статье рассматривается одна, предложенная автором, задача. Это задача распознавания некоторой конкретной ситуации, решенная с помощью ее логической модели – булева уравнения.

Примером булева уравнения с одним неизвестным может служить соотношение

$$X\cdot(A+B)=A\cdot B\cdot C,$$

где X – некоторая булева функция, зависящая от A, B, C, которую требуется найти, так чтобы в результате подстановки $X(A, B, C)$ в данное уравнение оно обращалось в тавтологию (тождественно истинная формула).

Примечание. Здесь и в дальнейшем будем использовать обозначения: • – знак конъюнкции, «+» – знак дизъюнкции, $\overline{A}$ – отрицание.

Для решения приведенного уравнения перейдем от элементов X, A, B, C к их изображающим числам относительно базиса $b[A, B, C]$:

$\#A\cdot B\cdot C$ ==0000 0001;

$\#(A+B)$ ==0111 0111.

Следовательно, $\#X$ должно быть таким, чтобы выполнялось равенство (0111 0111) • $(\#X)$ ==0000 0001, откуда $\#X$= ?000 ?001, где вместо «?» можно написать как 0, так и 1.

Таким образом, рассматриваемое уравнение имеет четыре решения, соответствующие изображающим числам 0000 0001, 1000 0001, 0000 1001 и 1000 1001, то есть

$$X_1=A\cdot B\cdot C,\ X_2=A\cdot B\cdot C+\overline{A}\cdot\overline{B}\cdot\overline{C},$$

$$X_3=C\cdot(A\cdot B+\overline{A}\cdot\overline{B}),\ X_4=\overline{A}\cdot\overline{B}+A\cdot B\cdot C.$$

Изложим общий метод решения на примере следующей задачи.

ЖАЛДАК Ирина Анатольевна – студентка V курса факультета информатики ВятГГУ

Формулировка задачи

Две враждующие страны, опасаясь нападения со стороны друг друга, вынуждены осуществлять постоянный контроль за приграничной зоной.

Однажды утром разведчики стороны **A** доложили, что, возможно, сторона **B** готовит нападение. Командование провело подробный допрос западной и восточной группы разведчиков по этому вопросу.

Систематизация результатов допроса разведчика западной группы показала следующее:

– в небе были замечены очертания самолетов, их появление сопровождалось гулом;

– на линии горизонта, возможно, появились танки, что сопровождалось громом, но самих танков замечено не было;

– вблизи приграничной зоны было замечено передвижение войск, гула и грома не было замечено.

Сведения, полученные при допросе разведчиков восточной группы, можно объединить в следующие группы:

– в небе были замечены очертания самолетов, гула и грома замечено не было;

– около приграничной зоны отмечено передвижение войск, были видны очертания танков и слышался гул;

– был слышен гром, но очертаний самолетов в небе замечено не было.

На основании этих данных требовалось определить, следует ли принимать всерьез заявление некоторых разведчиков о скором нападении противника или же представления о появлении техники противника могли быть вызваны комбинированными действиями атмосферных явлений и передвижением гражданских войск или невниманием разведчиков.

Решение

Введем необходимые обозначения для решения и формализации данной задачи. Следуя полученной информации, можно определить круг обозначений:

1) техника, с помощью которой производится нападение: танки (X), самолеты (Y);

2) атмосферные и другие явления: гул (A), гром (B), передвижение войск (C).

Используя введенные обозначения, результаты опроса представим в виде следующего булева уравнения:

$$Y\cdot A+X\cdot\overline{Y}\cdot B+C\cdot\overline{A}\cdot\overline{B}=$$
$$=Y\cdot\overline{A}\cdot\overline{B}+C\cdot A\cdot X+\overline{Y}\cdot B \qquad (*)$$

Решение вопроса о начале войны, то есть о нападении стороны B с помощью технических средств X и Y сводится к определению возможности разрешить уравнение (*) относительно неизвестных X и Y и выразить их как функции от элементов A, B, C.

Вычислим изображающие числа левой и правой частей уравнения.

Левая часть

1. Изображающие числа элементов A, B, C будем вычислять относительно базиса b[A, B, C]:

0123 4567, #A = 0101 0101,

#B = 0011 0011, $\#C \bullet \overline{A} \bullet \overline{B}$ = 0000 1000.

2. Изображающие числа неизвестных функций будем высчитывать относительно базиса b[X,Y]:

0123, #Y = 0011, $\#X \bullet \overline{Y}$ = 0100, #I = 1111.

Правая часть

1. Изображающие числа элементов A, B, C:

0123 4567, $\# \overline{A} \bullet \overline{B}$ = 1000 1000,
#C • A = 0000 0101, #B = 0011 0011.

2. Изображающие числа неизвестных функций:

0123, #Y = 0011, #X = 0101, $\# \overline{Y}$ = 1100.

Дальнейшие вычисления будем производить по следующей схеме.

1. Возьмем первую пару значений функций X, Y из базиса b[X, Y]. Это значения (0,0). Данной паре значений отвечает колонка 0 в наборе изображающих чисел неизвестных функций для левой части уравнения (*) и колонка 0 в наборе изображающих чисел неизвестных функций для правой части уравнения.

Умножим нулевую колонку изображающих чисел неизвестных функций левой части уравнения на нулевую колонку в наборе изображающих чисел элементов левой части уравнения

0 0 1 • 0 0 0 = 0 + 0 + 0 = 0.

Умножим нулевую колонку изображающих чисел неизвестных функций правой части уравнения на нулевую колонку в наборе изображающих чисел элементов правой части уравнения

0 0 1 • 1 0 0 = 0 + 0 + 0 = 0.

Поскольку суммы произведений колонок совпадают, можем записать в нулевые *(i=0)* разряды #X и #Y по отношению к *b[A, B, C]* значения элементов X и Y относительно b[X, Y] (это значения 0, 0), как будет показано ниже в таблице результатов.

2. Возьмем вторую пару значений функций X, Y из базиса b[X, Y]. Это значения (1,0). Данной паре значений отвечает колонка 1 в наборе изображающих чисел неизвестных функций для левой части уравнения (*) и колонка 1 в наборе изображающих чисел неизвестных функций для правой части уравнения.

Умножим первую колонку изображающих чисел неизвестных функций левой части уравнения на нулевую колонку в наборе изображающих чисел элементов левой части уравнения

0 1 1 • 0 0 0 = 0 + 0 + 0 = 0.

Умножим первую колонку изображающих чисел неизвестных функций правой части уравнения на нулевую колонку в наборе изображающих чисел элементов правой части уравнения

0 1 1 • 1 0 0 = 0 + 0 + 0 = 0.

Поскольку суммы произведений колонок совпадают, можем записать в нулевые *(i=0)* разряды #X и #Y по отношению к *b[A, B, C]* значения элементов X и Y относительно b[X, Y], находящиеся в колонке 1, это значения (1,0).

Выполняя вычисления далее, в разряде *i*=0 запишем две пары значений.

Затем те же операции выполняются с остальными колонками набора изображающих чисел элементов, начиная со второй.

Итак, в общем случае для проверки *i*-й пары значений *(X, Y)* колонки набора изображающих чисел неизвестных функций последовательно умножаются на *i*-ю колонку набора изображающих чисел элементов и результат логически складывается, при совпадении значений сумм для правой и левой частей уравнения, соответствующие *i*-й колонке неизвестные значения *(X, Y)* подставляются в *i*-й разряд таблицы результатов.

Ниже показана таблица результатов вычислений.

Согласно этой таблице имеется 2 • 2 • 3 • 1 • 2 • 2 • 3 • 2 = 288 различных неэквивалентных пар решений уравнения *(*)*. Например, если, в частности, взять числа, стоящие первыми в каждом разряде *(i=0, 1, ..., 7)* таблицы результатов, то получим:

#X = 0010 0011, #Y = 0001 1000.

В соответствии с совершенной нормальной дизъюнктивной формой заключаем:

$$\#X = (\overline{A} \bullet B \bullet \overline{C}) + (\overline{A} \bullet B \bullet C) + (A \bullet B \bullet C) = (\overline{A} \bullet B) + (A \bullet B \bullet C)$$

$$\#Y = (A \bullet B \bullet \overline{C}) + (A \bullet \overline{B} \bullet C)$$

Таким образом, на основании полученных данных в этом случае можно заключить, что заявление разведчиков о появлении танков может быть вызвано слышавшимся около приграничной зоны громом либо всеми явлениями, происходившими там во время наблюдения. Заявление о появлении самолетов противника может быть вызвано либо услышанным громом и гулом, либо замеченным передвижением, по всей видимости, гражданских сил противника. Поэтому нет серьезных оснований считать, что противник действительно готовит нападение.

Процедуру отбора пар значений *(X, Y)*, удовлетворяющих уравнению (*), можно значительно упростить, если ввести в рассмотрение операции над *матрицами.*

Отметим, что последовательное умножение с последующим сложением каждого столбца в наборе изображающих чисел элементов на все столбцы набора изображающих чисел неизвестных функций эквивалентно умножению булевой матрицы, полученной транспонированием таблицы изображающих чисел элементов, на булеву матрицу, составленную из изображающих чисел неизвестных.

Изображающие числа	Номер столбца в b[A, B, C]							
	0	1	2	3	4	5	6	7
#X	0, 1	0, 1	1,0,1	0	0,1	0,1	1,0,1	1,1
#Y	0, 0	0, 0	0,1,1	1	1,1	0,1	0,1,1	0,1

То есть для решения нашего примера необходимо:

1. Найти транспонированную матрицу набора изображающих чисел элементов левой части уравнения (*).

2. Умножить найденную матрицу на матрицу, соответствующую набору изображающих чисел неизвестных функций левой части уравнения.

3. Проделать те же операции для правой части уравнения.

4. Сравнить матрицы, полученные в результате умножения. Если какие-то элементы матриц совпадают, то есть $\|c_{ij}\| = \|d_{ij}\|$, то в соответствующий столбец *i* таблицы результатов записываются значения X и Y, отвечающие данным значениям, то есть значения колонки *j* базиса *b[X, Y]*.

Выполняя эти вычисления для левой части уравнения (*), найдем булеву матрицу:

$$\begin{Vmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{Vmatrix} \times \begin{Vmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{Vmatrix} = \begin{Vmatrix} \underline{\underline{0}} & \underline{\underline{0}} & 0 & 0 \\ \underline{\underline{0}} & \underline{\underline{0}} & 1 & 1 \\ \underline{\underline{0}} & \underline{1} & \underline{\underline{0}} & \underline{\underline{0}} \\ 0 & \underline{1} & 1 & 1 \\ 1 & 1 & \underline{\underline{1}} & \underline{\underline{1}} \\ \underline{\underline{0}} & 0 & 1 & \underline{\underline{1}} \\ \underline{\underline{0}} & \underline{1} & \underline{\underline{0}} & \underline{\underline{0}} \\ 0 & \underline{1} & 1 & \underline{\underline{1}} \end{Vmatrix}$$

Аналогично для правой части находим

$$\begin{Vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{Vmatrix} \times \begin{Vmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{Vmatrix} = \begin{Vmatrix} \underline{\underline{0}} & \underline{\underline{0}} & 1 & 1 \\ \underline{\underline{0}} & \underline{\underline{0}} & 0 & 0 \\ 1 & \underline{1} & \underline{\underline{0}} & \underline{\underline{0}} \\ 1 & \underline{1} & 0 & 0 \\ 0 & 0 & \underline{\underline{1}} & \underline{\underline{1}} \\ \underline{\underline{0}} & 1 & 0 & \underline{\underline{1}} \\ 1 & \underline{1} & \underline{\underline{0}} & \underline{\underline{0}} \\ 1 & \underline{1} & 0 & \underline{\underline{1}} \end{Vmatrix}$$

Одинаковые элементы двух результирующих матриц подчеркнуты. Анализ матриц проводится аналогично описанному выше. Например, в строке 0 совпадают элементы в столбцах 0 и 1. Таким образом, в таблицу результатов в разряд i=0 записываются значения колонок 0 и 1 базиса b[X, Y]. Полученная таким образом таблица результатов будет полностью соответствовать таблице, показанной выше.

Таким образом, можно заметить, что в результате применения метода решения булевых уравнений с помощью умножения булевых матриц можно точно и полностью получить все многообразие возможных решений. Уменьшение количества этих решений возможно только при введении дополнительных условий (параметров) задачи.

Примечания

Горелик А. Л., Скрипник В. А. Методы распознавания. М.: Высш. шк., 1989.

С. В. Зяблицев

О ВЫПОЛНЕНИИ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ С ДЛИННЫМИ ОТРИЦАТЕЛЬНЫМИ ЧИСЛАМИ

Приводятся особенности реализации арифметических операций с длинными отрицательными числами.

В работе [1] описаны арифметические действия с положительными длинными числами. Цель статьи заключается в том, чтобы изменить этот материал и для работы с отрицательными длинными числами.

Основные моменты логики программы.

Ввод данных

Возможно считывание данных как с клавиатуры, так и из файла. Данные считываются посимвольно и записываются в массив.

Считывая первый символ, мы смотрим, не является ли он «-». Если так, то мы записываем его в переменную Sign(:Integer).

```
Read(ch);
If ch='-' Then Begin Sign:=1; Read(ch) End
                                Else Sign:=0;
```

Далее воспользуемся процедурой ReadLong(Var A:TLong);

Затем после последнего разряда в массиве A мы ставим знак. *A[A[0]+1]:=Sign;*

Вывод данных

Возможен вывод как в файл, так и на экран. В процедуре WriteLong(Const A:TLong) учитываем знак в A[0]+1 разряде.

Перевод чисел в дополнительный код

Перевод числа в дополнительный код (Procedure Add_cod(Var A:TLong);) является важной процедурой для сложения отрицательных чисел и считается математически обоснованным и теоретически доказанным шагом. Состоит он в следующем.

ЗЯБЛИЦЕВ Сергей Васильевич – студент II курса факультета информатики ВятГГУ

Каждой цифре числа присваиваем значение разности максимальной возможной цифры (в данной системе счисления) и значения цифры.

```
For i:=1 To A[0] Do A[i]:=(Osn-1)-A[i];
                {*где Osn-1 максимальная цифра в
                Osn-ной системе счисления*}
```

К полученному числу мы прибавляем 1.

```
A[1]:=A[1]+1;
For i:=1 To A[0] Do Begin {*Производим возможный перенос 1 в другие разряды *}
 A[i+1]:=A[i+1]+(A[i]) Div Osn;
 A[i]:=(A[i]) Mod Osn;
End;
```

Кроме того, для нашей программы мы должны внимательно смотреть за знаком. То есть если значение знака четное, то число положительное, иначе оно отрицательное.

```
A[A[0]+1]:=A[A[0]+1] Mod2;
```

Замечание: в пунктах 3a и 3b мы абсолютно не касаемся знакового разряда.

Выравнивание длин чисел

Так как числа у нас задаются с плавающей длинной, то перед выполнением сложения или вычитания необходимо выровнять их длину.

```
Procedure Leveling_length(Var A,B:TLong);
 Var k,Sign:Integer;
  Begin
  If A[0]>B[0] Then k:=A[0] Else k:=B[0]; Inc(k);
      {*Так как данная процедура необходима
       только при сложении и вычитании, мы
       умышленно увеличиваем размерность чисел
       на 1 разряд, оставляя его пустым, иначе в
       процессе сложения или вычитания может
       произойти «непроизвольная смена знака»*}
    Sign:=A[A[0]+1]; A[A[0]+1]:=0; A[0]:=k;
A[A[0]+1]:=Sign;
    Sign:=B[B[0]+1]; B[B[0]+1]:=0; B[0]:=k;
B[B[0]+1]:=Sign;
  End;
```

Суммирование

Используем процедуру Summ(Const A,B:TLong; Var C:TLong);

Однако суммируем не до последнего разряда, а до следующего, то есть мы складываем и знаковые разряды!!!

Длина результата должна строго совпадать с первым или вторым числом.

Сложение

```
Leveling_length(A,B);
If A[A[0]+1]<>0 Then Add_cod(A); {*Если число
            отрицательное, мы переводим его
            в дополнительный код*}
If B[B[0]+1]<>0 Then Add_cod(B);
Summ(A,B,C);
If C[C[0]+1]<>0 Then Add_cod(C);
```

Вычитание

```
Leveling_length(A,B);
If A[A[0]+1]<>0 Then Add_cod(A); {*Если число
            отрицательное, мы переводим его
            в дополнительный код*}
If B[B[0]+1]<>0 Then Add_cod(B);
Inc(B[B[0]+1]); {*Переводим в дополнительный
            код, не спрашивая знака*}
Add_cod(B);
Summ(A,B,C);
If C[C[0]+1]<>0 Then Add_cod(C);
```

Длину результата можно подровнять

```
Procedure Justification(Var A:TLong);)
Var Sign:byte;
   i:Integer;
Begin
 Sign:=A[A[0]+1];
 A[A[0]+1]:=0;
 i:=A[0];
 While (i>1) And (A[i]=0) Do Dec(i); {*В числе все
       равно должен быть хотя бы 1 разряд*}
 A[0]:=i;
 A[A[0]+1]:=Sign;
End;
```

Функция больше возвращает истину или ложь

```
Function More(Const A,B:TLong):Boolean;
Var i:Integer; p:Boolean;
 Begin
 Case (A[A[0]+1] Xor B[B[0]+1]) Of
                    {*Учитываем знак*}
 0:Case A[A[0]+1] Of
                    {*Сравниваем положительные*}
  0:If A[0]>B[0] Then p:=True Else
     If A[0]<B[0] Then p:=False Else Begin
    i:=A[0];
    While (i>0)And(A[i]=B[i])Do Dec(i);
    If (A[i]>B[i])And(i>0) Then p:=True Else
                                  p:=False;
     End;
  1:If A[0]<B[0] Then p:=True Else
                    {*Сравниваем отрицательные*}
   If A[0]>B[0] Then p:=False Else Begin
    i:=A[0];
    While (i>0)And(A[i]=B[i])Do Dec(i);
    If (A[i]<B[i])And(i>0) Then p:=True Else
                                  p:=False;
     End;
     End;
  1:If A[A[0]+1]=0 Then p:=True Else p:=False;
 End;
 More:=p;
End;
```

Функция равенства аналогична приведенной в [1].

Процедура произведение «длинного» на «длинное» немного исправленная:

```
Procedure MulLong(A,B:TLong; Var C:TLong);
 Var Sign,i,j:Integer;
  Begin
  Sign:=A[A[0]+1] Xor B[B[0]+1]; A[A[0]+1]:
    =0; B[B[0]+1]:=0; {*Учитываем знак*}
```



```
 If More(b,a) Then Begin c:=a; a:=b; b:=c; End;
 FillChar(c,SizeOf(c),0); C[0]:=1;
  For j:=1 To B[0] Do Begin
   For i:=1 To A[0] Do Begin
    C[i+j]:=C[i+j]+(Longint(A[i])*B[j]+
                        +C[i+j-1]) Div Osn;
    C[i+j-1]:=(Longint(A[i])*B[j]+
                        +C[i+j-1]) Mod Osn;
   End;
 If C[A[0]+1]>0 Then C[0]:=A[0]+1 Else
                        C[0]:=A[0];
  End;
 C[C[0]+1]:=Sign;
End;
```

Умножение на «короткое» число, не превосходящее основание, аналогично приведенной в [1], но с учетом знака.

Вариант процедуры деления без остатка

```
Procedure Fission(A,B:TLong; Var DivLong:TLong);
 Var Sign,i,j,nac,con:Integer;
   C,D:TLong;
 Begin
  FillChar(DivLong,SizeOf(DivLong),0);
          {*Инициализация длинного частного*}
 Sign:=A[A[0]+1]XorB[B[0]+1];   A[A[0]+1]:=0;
          B[B[0]+1]:=0; {*Учет знака*}
  FillChar(D,SizeOf(D),0); {*Инициализация
                    дополнительного массива*}
  If More(A,B) Then {*Если А больше В,
              то записываем несколько старших
              разрядов в D и убираем их из А*}
   For j:=1 To B[0]-1 Do Begin
    Inc(D[0]);
    For i:=D[0] DownTo 2 Do D[i]:=D[i-1];
    D[1]:=A[A[0]];
    A[A[0]]:=0;
    Dec(A[0]);
   End;
 While A[0]>0 Do Begin
          {*Записываем очередной разряд в D*}
   Inc(D[0]);
   For i:=D[0] DownTo 2 Do D[i]:=D[i-1];
   D[1]:=A[A[0]];
   A[A[0]]:=0;
   Dec(A[0]);
   While More(B,D) Do Begin {*Если D меньше
          B, то увеличиваем D и увеличиваем
          DivLong на пустой разряд*}
   Inc(DivLong[0]);
    For i:=DivLong[0] DownTo 2
            Do DivLong[i]:=DivLong[i-1];
    DivLong[1]:=0;
   Inc(D[0]);
   For i:=D[0] DownTo 2 Do D[i]:=D[i-1];
   D[1]:=A[A[0]];
   A[A[0]]:=0;
   Dec(A[0]);
    End;
```

```
 {*Подбираем цифру, при умножении на которую
  число B*i максимально «близко» к D, но
  не превышает его*}
    nac:=1; con:=Osn-1;
    While nac+1<con Do Begin
     i:=(nac+con) Div 2;
    Umn(B,i,C);
    If More(D,C) Then nac:=i Else
     If EqLong(D,C) Then Begin nac:=i; con:=i;
End
        Else con:=i;
    End;
{*Изменяем D. D:=D-B*i *}
   Umn(B,nac,C);
   Minus(D,C,D);
   Dec(D[0]);
{*Приписываем i в DivLong*}
   Inc(DivLong[0]);
   For i:=DivLong[0] DownTo 2 Do DivLong[i]:
                        =DivLong[i-1];
   DivLong[1]:=nac;
 End;
 DivLong[DivLong[0]+1]:=Sign;
End;
```

Примечания

1. *Окулов С. М.* Программирование в алгоритмах. М.: Бином, 2002.

2. *Окулов С. М., Пестов А. А.* 100 задач по информатике. Киров: Изд-во ВГПУ, 2000.

С. Ю. Иванов

МЕТОД ВОЛНЫ: ОТ ПРОСТОГО К СЛОЖНОМУ

Показано, как «раскручивать» один из базовых алгоритмов информатики при работе со школьниками.

В статье рассматривается один из базовых алгоритмов решения задач, связанных с нахождением пути, содержащего минимальное количество ходов. Суть этого алгоритма заключается в следующем. Имеется двумерный массив, в нем задана начальная и конечная клетки, а также в нем есть некоторые препятствия – клетки, ходить по которым нельзя. Из начальной точки запускаем волну: все клетки, в которые можно попасть из начальной за 1 шаг, получают значение, равное 1. Затем перебираем все точки, в которые можно попасть за 1 шаг, и соседним с ними клетками присваиваем значение 2. Процесс продол-

ИВАНОВ Сергей Юрьевич – студент V курса факультета информатики ВятГГУ

жается до тех пор, пока не дойдем до конечной клетки (нашли путь) или пока можем совершить очередной ход, если сходить не удается, то пути из начальной в конечную клетку не существует. Такой алгоритм получил название метода волны, или поиска в ширину.

0	1	2	■	
1	2	■	8	
2	■	■	7	8
3	4	5	6	7
4	5	6	■	8

Рис. 1

Пример метода волны приведен на рис.1. Черными клетками обозначены препятствия. Волну запускаем из верхней левой клетки, конечная – нижняя правая. Ходить можно либо по горизонтали, либо по вертикали, по диагонали ходы запрещены. Получаем, что в конечную клетку можно попасть за 8 ходов.

Существуют различные задачи, для решения которых подходит данный алгоритм.

1. ***Задача о лабиринте.*** Дано: двумерный массив с препятствиями, начальная и конечная клетки. Найти путь с минимальным количеством ходов. Идея решения уже рассмотрена выше. Данный метод часто применяется в компьютерных играх, как правило, в стратегиях, так как на каждом шаге необходимо заново запускать волну, т. е. находить снова лучший путь после изменения положения некоторых объектов на поле, а это требует значительных временных затрат.

2. Для увеличения быстродействия метода волны можно применить некоторое его улучшение – ***двойной поиск в ширину***. Его отличие в том, что запускается сразу две волны, одна – из начальной клетки, другая – из конечной. Серыми клетками обозначены начальная и конечная клетки. Слева (рис. 2а) пример с запусканием одной волны, справа (рис. 2б) запускаются сразу две волны. Теоретически вторая волна может дать улучшение в 3-4 раза, но на практике, как всегда, все оказывается «немного» не так.

3. ***Задача о замке.*** Имеется план замка, т. е. известно, где расположены стены внутри замка (рис. 3). Необходимо определить, сколько комнат в замке, найти комнату с наибольшей площадью и определить, какую перегородку необходимо убрать, чтобы получить комнату с максимально возможной площадью.

2	3	4	5	
1	2	3	4	5
0	1	■	5	
1	2			
2	3	4	5	

а. Пример с запуском одной волны

2	2	1	0	1
1	2	2	1	2
0	1	■	2	
1	2	■		
2				

б. Пример с запуском двух волн

Рис. 2

С помощью метода волны эту задачу можно решить следующим образом. Ищем в массиве «пустую» клетку, запускаем из нее волну и считаем, сколько всего ходов сделали, т. е. сколько клеток заполнили числами, это и будет площадью комнаты. Повторяем процесс, пока есть свободные клетки. Количество комнат равно количеству запусков волны.

0	1	2	■	0
1	2	■	2	1
■	■	■	■	■
0	1	■	0	1
1	2	3	■	2

Рис.3. Решение задачи о замке методом волны

4. Еще одна модификация задачи о лабиринте – ***задача о пути с минимальным количеством поворотов***. Дано: двумерный массив с препятствиями, начальная и конечная клетки. Из начальной клетки можно двигаться в любом направлении по горизонтали или вертикали, но затем направление можно поменять только «столкнувшись» с препятствием или границей лабиринта. Необходимо найти путь с минимальным количеством поворотов.

Эту задачу удобно решать методом волны. Единственной модификацией алгоритма здесь будет «необычный» ход. «Необычный» ход представляет собой следующее: из текущей клетки во всех направлениях ищем препятствие или границу лабиринта, как только нашли, определяем, стоим на непосещенной клетке или на ней мы уже были, если клетка свободна, то заносим туда соответствующее значение – номер шага. Рис. 4 демонстрирует прохождение лабиринта с применением такого «необычного» хода.

0	3	1	■	
			4	
3		2	3	■
1	2	■		
■	3		4	4

Рис. 4. Метод волны с «необычным» ходом

5. ***Задача о «блохах».*** На шахматной доске размером NxM имеется Q блох и имеется кормушка в клетке с координатами (x,y) – рис. 5а (X-кормушка, Б-блоха). По полю блохи передвигаются ходом шахматного коня. Необходимо определить для каждой блохи, за какое минимальное количество прыжков она сможет добраться до кормушки, или определить, что это сделать невозможно. (1<N<250, 1<M<250, 0<Q<10000).

Можно брать координаты каждой блохи как начальную клетку, а координаты кормушки – как конечную и запускать волну Q раз. Но при Q=10000, данный алгоритм не будет проходить по времени. Здесь целесообразно запустить волну из кормушки, если на

очередном ходе попадаем в клетку с блохой, то мы нашли длину ее пути до кормушки. Результат виден на рис. 5б: черные клетки – первоначальные места расположения блох, клетка, в которой записан 0, представляет собой кормушку.

Б			Б	Б
	X			
Б				
Б		Б		
			Б	Б

а. Задача о блохах

4	3	2	1	2
3	0	3	2	3
2	3	2	1	2
1	2	1	4	3
2	3	2	3	2

б. Задача о блохах. Волна запускается из кормушки

Рис. 5

6. Можно также модифицировать метод волны на ***трехмерное пространство*** (рис. 6), для этого потребуется трехмерный массив с лабиринтом, и на каждом шаге просматривается другое количество соседних клеток. Можно модифицировать данный алгоритм для тел произвольной формы. Для этого, естественно, надо будет изменить логику установления факта встречи с препятствием и при необходимости предусмотреть возможность поворота объекта для прохождения в узких местах лабиринта.

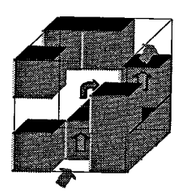
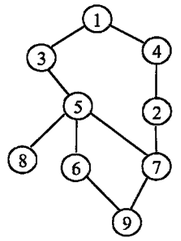
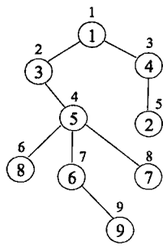


Рис. 6. Трехмерный лабиринт



а. Исходный граф

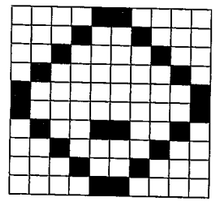


б. Очередность просмотра вершин графа

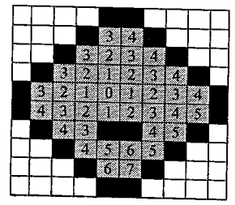
Рис. 7

7. Метод волны используется также и ***в алгоритмах на графах*** – поиск в ширину, для нахождения кратчайших путей между исходной вершиной и конечной (или всеми остальными). Дано: граф (рис. 7а). Необходимо найти путь от первой вершины до девятой, содержащий минимальное количество ребер. На рис. 7б показана очередность просмотра вершин графа.

8. В компьютерной графике метод волны может применяться ***для закрашивания областей произвольной формы*** или для выделения определенного цвета в заданной замкнутой области. Для этого другие цвета будут рассматриваться как препятствия, а искомый цвет с допустимыми отклонениями – как свободные клетки. По сути, это задача о замке, так как в обоих случаях имеются замкнутые области, которые надо полностью просмотреть. Дано: произвольная область (рис. 8а). Требуется закрасить серым цветом всю белую область внутри черной границы. На рис. 8б показан результат, а также ход выполнения методом волны. На каждом шаге данного алгоритма мы просматриваем клетку: если она белая, то закрашиваем, в противном случае ищем среди соседних клеток белые, куда можно сходить.



а. Произвольная область для закраски



б. Область после закраски

Рис. 8

9. Данный метод также применяется для ***векторизации изображений***. Допустим, у нас имеется растровое изображение (для каждой точки задан определенный цвет), мы хотим преобразовать его в векторную форму (изображение состоит из векторов, т.е. известно положение двух точек, между которыми строится прямая). Для простоты будем полагать, что фон изображения белый, а сам рисунок изображен только черным цветом.

На рис. 9 приводится пример растрового изображения (черным цветом) и построенного векторного (серым цветом). Векторизация заключается в построении графа, т. е. нам необходимо найти в растровом изображении «контрольные точки» (вершины в графе) и соединить их отрезками (ребра в графе). Данный метод основан на том, что мы предполагаем определенную ширину линий и, основываясь на этом, пытаемся выделить ее в растровом изображении. Необходимо также учитывать, что «линии» в растровом изображении могут пересекаться и соединяться (как показано на рис. 9). Для определения этих линий и мест их пересечения и используется метод волны. Запускаем волну из какого-либо места растро-

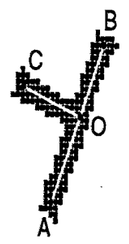


Рис. 9

вого изображения, через 2*N шагов (N – ширина «линии» в пикселях) волна принимает устойчивый характер, на каждом шаге нам необходимо отслеживать «ширину» волны, т. е. количество точек, образующих текущую генерацию волны. В местах пересечения «линий» растрового изображения наша волна будет разделяться на две дочерние волны. Следовательно, перед разделением волны ее «ширина будет увеличиваться», а затем произойдет разделение волны. Таким образом можно определить точку соединения или пересечения «линий» в растровом изображении. После построения такого каркаса возможна дальнейшая оптимизация, например уточнение места положения точек пересечения, удаление лишних точек и т. д., но это не входит в задачу данной статьи. Этот метод может применяться для создания программ распознавания текста, конструкторской документации, САПР.

Применение метода волны для решения задач на графах или задач о лабиринте и близких к ним подробно описаны в работах [1]. Применение волнового алгоритма и его модификации для компьютерных игр рассмотрены в статьях [2].

Примечания

1. *Окулов С. М., Пестов А. А., Пестов О. А.* Информатика в задачах. Киров: Изд-во ВГПУ, 199; *Окулов С. М., Пестов А. А.* 100 задач по информатике. Киров: Изд-во ВГПУ, 2000.

2. Информационный ресурс Интернет http://algolist.manual.ru; Информационный ресурс Интернет http://www.ocrai.narod.ru

С. Г. Кипров

ПОИСК НАИБОЛЬШЕЙ ОБЩЕЙ ПОСЛЕДОВАТЕЛЬНОСТИ

В связи с запросами практики проблеме решения задачи за минимальное время придается большое значение. Особенность входных данных определяет то, какой из алгоритмов лучше использовать. Анализу этого соответствия и посвящена статья.

Задача нахождения общей подпоследовательности двух последовательностей широко используется во многих областях, например: проверка правописания, сравнение файлов, изучение мутации генов. И если для первых двух областей достаточно сказать, одинаковые последовательности или нет (или количество общих элементов) – для этого подойдет алгоритм 2а, то в задачах, связанных с генетикой, необходимо найти общую подпоследовательность максимальной длины. Для этой цели мог бы подойти любой алгоритм, $O(N^2)$ времени и использующий двумерный массив, но для его хранения памяти может быть недостаточно, так как длина цепочки ДНК может достигать нескольких сотен тысяч, а иногда и миллионов символов (обозначающих кислоты).

Для подобных задач как нельзя лучше подойдет алгоритм 2б.

В статье кроме алгоритмов 2а и 2б будут рассмотрены алгоритмы, им предшествовавшие, работающие над той же проблемой, но с меньшей производительностью.

Итак, начнем!

Алгоритм 1. Малого объема памяти, но обратно пропорционального ему времени

Данный алгоритм достаточно прост и может быть применен на небольших похожих последовательностях, в приложениях, не часто использующих данную операцию. Суть метода заключается в том, что программа использует символы двух исходных последовательностей для построения графа (возможно циклического), одновременно ищет путь до конечной точки (на рис. 1 – большая красная точка) с минимальным количеством несовпадений (на рис. 1 несовпадения – невертикальные ребра).

Временная оценка метода. Временная сложность данного алгоритма $-((2*(N+M))!/((N+M)!(N+M)!))$ времени. Данный алгоритм нахождения максимальной общей подпоследовательности похож на переборный вариант решения задачи о минимальном пути черепашки, поэтому за подробностями оценки можно обращаться к [1].

Суть метода: Метод основан на обходе графа в глубину с возвратом. На каждом шаге программы прибавляется только одно не совпадающее ребро и следующие за ним совпадающие ребра (если таковые существуют). По достижении конца графа возвращаемся на начальное значение предыдущего шага, и если первый раз на данном шаге поиска в графе путь «уходил» вправо, то при возврате путь «уходит» влево, создавая дальнейший путь (построение графа показано на рис. 1).

Усовершенствование: Для некоторого ускорения в работе программы можно учитывать еще количество шагов по первому и дальнейшему достижению конечной точки. И если в текущем пути количество невертикальных ребер превосходит количество невертикальных ребер лучшего пути, то построение данной ветки в графе закончено.

Часть алгоритма с некоторым усовершенствованием представлена ниже:

```
Procedure Solve(x,y:integer{length(a),length(b)});// A,B – последовательности, представленные в виде массива
 begin
  delete(Res,((x+y-step)div 2)+1,length(Res));
  // проходим по вертикальным ребрам
  while (A[x+1]=B[y+1])and(x<Length(A))
  and(y<Length(B))and(Step<=BestStep) do
  begin
  inc(x);
  inc(y);
  Res:=Res+A[x];
  end;
 //вызов рекурсии с новыми значениями индекса сравниваемых символов
  if((x<Length(A))or(y<Length(B)))
                    and(Step<=BestStep)then
  begin
  if x<Length(A)then
   begin inc(Step); Solve(x+1,y); end;
  if y<length(B)then
   begin inc(Step); Solve(x,y+1);
    end;
 //улучшение и хранение оптимального пути
  end else
  begin
  if(x>=length(A))and(y>=Length(B)) then
   begin
   if length(Res)>Length(BestRes) then
    begin
     BestStep:=Step;
     BestRes:=Res;
    end;
   end;
  end;
 dec(step);
end;
```

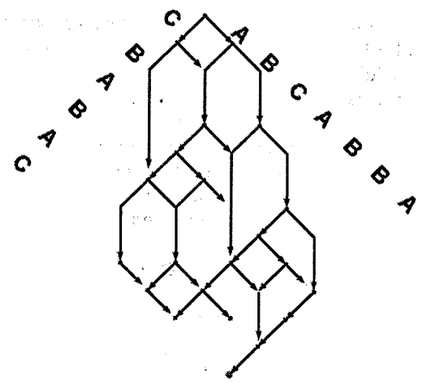


Рис. 1. Ориентированный граф для алгоритма 1 для последовательностей «АВСАВВВА» и «СВАВАС»

Алгоритм 2. O(ND) временной алгоритм

Временных O(ND) алгоритмов существует несколько, мы не будем затрагивать все из них, только укажем, что данный алгоритм 2а (алгоритм Myers'a [2]) несколько напоминает алгоритм Нудельмана-Вунша [3] и по времени работы, и по объему занимаемой памяти (если требуется вывод всех путей), но в некоторых случаях он может быть гораздо производительней – при небольших различиях в последовательности (это существенно заметно, если количество различий не превосходит 25% символов). В случае, если требуется только посчитать количество элементов в подпоследовательности, данный алгоритм может быть очень хорош, так как в этом случае он занимает линейный объем памяти (особенно эффективен при больших длинах примерно одинаковых последовательностей).

Алгоритм 2а O(ND) временной алгоритм

Предварительные разъяснения. Редактируемый граф

Пусть $A=a_1a_2\ldots a_n$ и $B=b_1b_2\ldots b_y$ — последовательность длины N и M соответственно. Редактируемый граф A и B обладает вершинами в каждой точке сети (x,y), где $x\in[0,N]$, $y\in[0,M]$. Редактируемый граф обладает вертикальными, горизонтальными и диагональными ребрами. Вертикальные ребра соединены с соседними вершинами справа, т. е. $(x-1,y)\rightarrow(x,y)$, где $x\in[1,N]$, $y\in[0,M]$. Горизонтальные ребра соединены с соседними вершинами сверху, т. е. $(x,y-1)\rightarrow(x,y)$, где $x\in[0,N]$, $y\in[1,M]$. Если $a_x=b_y$, тогда образуется диагональное ребро, соединяющее диагональных соседей так, что $(x-1,y-1)\rightarrow(x,y)$. Точки (x,y), для которых $a_x=b_y$, назовем равными точками. На рис. 2 представлен граф для двух последовательностей (последовательности отличны от последовательностей алгоритма 1).

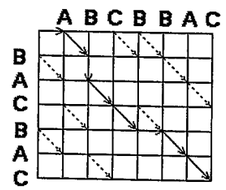


Рис. 2

Путь отмечен непрерывной стрелкой, прерывающаяся стрелка – совпавшие символы (диагонали), наибольшая общая последовательность – BCBAC

Путь длиной L – это последовательность из L равных точек, $(x_1,y_1)(x_2,y_2)\ldots(x_L,y_L)$, таких, что $x_i<x_{i+1}$ и $y_i<y_{i+1}$ для следующих точек (x_i,y_i) и (x_{i+1},y_{i+1}), $i\in[1,L-1]$. Каждый шаг является точным соответствием диагонального ребра в пути в редактируемом графе из (0,0) в (N,M). Построить путь по шагам можно, собрав последовательность из диагональных ребер соответствующих совпавшим точкам по шагам и соединенных со следующим рядом вертикальных или горизонтальных ребер. Это всегда может быть сделано, так как $x_i<x_{i+1}$ и $y_i<y_{i+1}$ до следующих совпавших точек. Заметьте, что несколько путей отличаются только их недиагональным ребрами, может быть, эквивалентными в данном пути. Рис. 2 иллюстрирует описание создания пути и шагов.

Проблема нахождения наибольшей общей подпоследовательности равняется нахождению пути из (0,0) в (N,M) с наибольшим количеством диагональных ребер. А проблема нахождения кратчайшей редакти-

КИПРОВ Сергей Геннадьевич – студент IV курса факультета информатики ВятГГУ

руемой последовательности в свою очередь равняется нахождению пути из (0,0) в (N,M) с минимальным количеством недиагональных ребер. Если мы присвоим диагональным ребрам вес 0, а не диагональным – 1, то данную задачу можно решить за счет нахождения пути с минимальной стоимостью во взвешенном графе.

Суть. Пусть D-путь начинается в (0,0), который содержит D недиагональных ребер. 0 путь должен содержать только диагональные ребра. На следующем шаге мы добавляем к имеющемуся пути еще одно либо вертикальное, либо диагональное ребро (вызываем либо (x+1,y), либо (x,y+1)) и следующие за ним диагональные ребра (если таковые имеются). Таким образом, D-путь должен состоять из D-1 следующего недиагонального ребра и возможно пустой последовательностью диагональных ребер, названных змейкой. Прохождение по графу будет продолжаться до тех пор, пока (x,y) не достигнет правого нижнего угла (если смотреть на рисунок так, как задумывал автор).

Необходимо упомянуть два важных момента для создания алгоритма:

1) за один шаг невозможно пройти более 1 недиагонального ребра;

2) при добавлении змейки к D-пути необходимо брать наибольшее число диагональных ребер, иначе полученный путь может оказаться не наибольшей общей подпоследовательностью;

3) из 1 следует, что любой D-путь содержит ровно (D-1) недиагональное ребро (так как всего D шагов, на каждом из которых, кроме первого, содержалось 1 недиагональное ребро).

Представленная ниже процедура предлагает нахождение длины максимальной общей последовательности.

```
Procedure Solve;
var x,y,k,D,Dout:integer;
begin
    Res[1]:=0; //вспомогательное значение
                                для 0 пути
//внешний цикл For – расширяет область
                    диагоналей на каждом шаге
    For D:=0 to length(A)+Length(B)do
    begin
//внутренний цикл For – начинает проход с самой
левой диагонали и
//заканчивает на самой правой
    For k:=-d to d do
//проход по диагоналям, находящимся через
                    1 диагональ с самой левой
//об этом позже в леммах 1 и 2
    If (abs(k) mod 2)=(d mod 2)then begin
//выбор диагонали (D-1)-пути (пути-предка)
    If (k=-d)or(k<>D)and(Res[k-1]<Res[k+1])
    then x:=Res[k+1]
    else x:=Res[k-1]+1;
    y:=x-k;
//создание змейки максимальной длины
    While x<length(A))and(y<Length(B))
                    and(A[x+1]=b[y+1])do
    begin inc(x);
    inc(y); end;
    Res[k]:=x;
//проверка условия выхода из процедуры
    If (x>=length(A))and(y>=length(B))then
    begin
    Res[k] – длина максимальной
                    последовательности};
    exit
    end
    end;
end;
```

На практике алгоритм ищет D-пути, где D<= максимуму и если такой путь достигает (N,M) тогда любая редактируемая последовательность для A и B должна по строке exit быть больше чем максимальная длина последовательностей (далее просто максимум). Полагая, что максимум эта сумма двух длин последовательностей, алгоритм гарантированно найдет максимальную длину общей последовательности. Рис. 3 показывает найденный D-путь, когда алгоритм применен к примеру на рис.1. Заметим, что несуществующая точка (0,-1) была введена для нахождения змейки 0-пути. Также заметим, что D-пути не увеличиваются, достигая правой и нижней границ в редактируемом графе при работе алгоритма. Эта пограничная ситуация обработана тем условием, что после граничных ситуаций в одной из последовательностей нет символов.

Чтобы не быть голословным, докажем утверждения 1 и 3 (они также упоминаются в описании процедуры Solve):

Лемма 1. D-путь должен заканчиваться на диагонали kO{-D,-D+2,…D-2,D}.

Доказательство. 0 путь состоит исключительно из диагональных ребер и начинается на диагонали 0. Отсюда он должен заканчиваться на диагонали 0. Предположим, что D путь должен заканчиваться

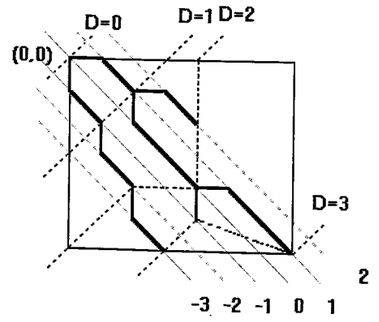


Рис. 3
Найденные за D-шагов пути (выделены жирным)

на диагонали kO{-D,-D+2,…D-2,D}. Каждый (D+1) путь состоит из предыдущего D пути, заканчивающегося, как сказано, на диагонали k, недиагонального ребра, заканчивающегося на диагонали k-1 (k+1), и змейки, которая также должна заканчиваться на диагонали k-1 (k+1). Из этого следует, что каждый (D+1) путь должен заканчиваться на {(-D)±1,(-D+2) ±1,… (D-2)±1,D±1}={-D-1,-D+1,-D+3,…D-1,D+1}. Таким образом, можно продолжить рассуждения для следующих D+1+l-путей и для предыдущих D-l-путей, где l-произвольное целое число. Лемма подразумевает, что D путь заканчивается исключительно на нечетной диагонали, когда D нечетное, и на четной, когда D четное.

D путь является самым длинным на диагонали k тогда и только тогда, когда один из D путей заканчивается на диагонали k, чья конечная точка содержит наибольшее возможное число рядов (столбцов) всех таких путей. Другими словами, из всех D путей, заканчивающихся на диагонали k, он заканчивается на самом длинном пути из всех начинавшихся в (0,0). Следующая лемма дает характеристику длиннейшего D пути и реализует жадный принцип: длиннейший D путь получается путем увеличения длиннейшего (D-1) пути.

Лемма 2. Самый длинный 0 путь заканчивается в (x,x), где x – минимум (z-1Π $a_x \neq b_z$ или z>M или z>N). Самый длинный D путь на диагонали k может быть заменен без потерь на самый длинный D-1 путь на диагонали k-1, горизонтальное ребро и наиболее длинную змейку, или он может быть заменен на самый длинный D-1 путь на диагонали k+1, вертикальное ребро и наиболее длинную змейку.

Доказательство. Основа 0 путей проста. Как замечалось ранее, D путь содержит D-1 путь, недиагонального ребра и змейки. Если D путь заканчивается на диагонали k, из этого следует, что D-1 путь должен заканчиваться на диагонали k±1 в зависимости от того, горизонтальное или вертикальное ребро было выбрано перед финальной змейкой. Финальная змейка должна быть максимальной, так как D путь не будет максимальным, если змейку можно будет увеличить. Предположим, D-1 путь не самый дальний на этой диагонали. Но тогда самый дальний D-1 путь может быть связан с последней змейкой D пути данным недиагональным шагом. Таким образом, D путь по желанию может быть всегда разложен на D-1-путь, недиагональное ребро и змейку (если хранить предыдущее значение).

Время выполнения процедуры. Данная процедура обладает временем O((M+N)max(N,M)), складывающимся из следующих действий:

- внешний цикл повторяется в худшем случае M+N раз;
- внутренний цикл For k:=-D to D повторяется 2D+1 раз;
- обращение ко всем внутренним операциям во внутреннем цикле For k:=-D to D происходит за счет условия (abs(k) mod 2)=(d mod 2), которое определяет шаг ((D div 2)+(D mod 2)) раз;
- таким образом, внутренние операции, кроме цикла While, повторяются D/2 раз (округление);
- если цикл While повторится хотя бы один раз, то время выполнения процедуры будет улучшено, поэтому его повторение мы будем учитывать только для худшего случая, таким образом, он будет пропорционален O(1);
- исходя из всех вышеперечисленных пунктов общее время складывается только из двух циклов For и времени, затрачиваемого на внутренние операции, и в худшем случае на данную процедуру уйдет O((M+N)max(N,M)) времени.

Замечание. Как было сказано выше, для вывода обратного пути итогового массива D недостаточно, необходимо знать еще и предыдущее значение. Поэтому для вывода обратного D-пути следует хранить все исходные массивы, что иногда может и не удовлетворять условиям задачи (на больших последовательностях), но в некоторых случаях он может быть полезен.

Алгоритм 2б. Улучшенный алгоритм 2а с линейным объемом памяти

Усовершенствованный алгоритм Mayers'a [4].

Как было сказано выше, для вывода пути необходимо запоминать массив D на каждом повторении внешнего цикла For. Таким образом, для вывода общей последовательности необходим квадратичный объем памяти. Поэтому попробуем уменьшить объем до линейного. Для этого немного изменим структуру алгоритма. Алгоритмом, представленным выше, определяется количество недиагональных ребер в максимальной общей последовательности.

Для дальнейшей сборки алгоритма используем тип алгоритма «разделяй и властвуй». Разделим максимальную последовательность пополам, пусть мы остановились на точке (x,y), затем найдем максимальную последовательность от (0,0) до (x,y) и от (x',y') до конечной точки. Следующим шагом повторим вышеописанные действия. Так повторяем до тех пор, пока в общей последовательности не останется недиагональных ребер ≤ 1. Для доказательства возможности использования данного алгоритма воспользуемся леммой 3 и ее доказательством.

Лемма 3. D-путь из (0,0) до (N,M) существует тогда и только тогда, когда существует [D/2]-путь из (0,0) в (x,y) и [D/2]-путь из некоторой точки (u,v) до (N,M), такой, что:

{выполнимость} $(u + v) \geq [D/2]$ и
$(x + y) \leq (N+M-[D/2])$ и
{частичное совпадение} $x - y = u - v$ и $x \geq u$

Более того, оба [D/2]-пути составляют D-путь из (0,0) до (N,M)

Доказательство. Предположим, что существует D-путь из (0,0) до (N,M). Он может быть разделен от начала (0,0), до точки (x,y),что является серединой

змейки [D/2]-пути из (0,0) до (x,y) и [D/2]-пути от (u,v) до (N,M), где (u,v)=(x,y). Путь из (0,0) до (u,v) может содержать максимум u + v недиагональных ребер и [D/2]-путь до (u,v) предполагает, что u + v ≤ [D/2]. Путь их (x,y) до (N,M) может содержать (N+M)-(x+y) недиагональных ребер и [D/2]-путь подразумевает, что x+y ≤ N+M-[D/2]. И наконец, u-v=x-y и u ≤ x, так как (x, y)=(u,v).

Обратно, предположим [D/2] и [D/2]-пути существуют. Но u ≤ x подразумевает, что существует k-путь из (0,0) в (u,v), где k ≤ [D/2]. По лемме 1, D=[D/2]-k кратно 2, так как оба пути и k и [D/2] заканчиваются на одной диагонали. Более того, k-путь содержит (u + v – k)/2 ≥ D/2 диагоналей, так как u + v ≤ [D/2]. Заменив каждую D/2 диагонали в k-пути с парой вертикальных и горизонтальных ребер, [D/2] путь из (0,0) до (u,v) найдется. Но тогда будет существовать D-путь из (0,0) до (N,M), состоящий из этого [D/2]-путь до (u,v) и и данного [D/2]-путь из (u,v) до (N,M). Заметьте, что [D/2]-путь – это часть D-пути. И симметричный [D/2]-путь – это также часть D-пути из (0,0) до (N,M).

Ниже приведены процедуры для алгоритма, использующего вышеописанный метод.

```
//процедура нахождения среднего пути для двух
//подпоследовательностей
function MiddlePoint(var x,y,u,v:integer):Integer;
 var D,k,l,delta:integer;
   way,back:MyArray;
 begin
 delta:=length(AA)-Length(BB);
 FillChar(Way,SizeOf(Way),0);
 FillChar(Back,SizeOf(Back),10);
 Way[1]:=0;
Back[delta+1]:=Maxim(Length(AA),Length(BB))+1;
 For D:=0 to ((length(AA)+Length(BB))div 2)+1 do
   begin
//прямой путь
     For k:=-d to d do
    If (abs(k) mod 2)=(d mod 2)then
       begin
      If (k=-d)or(k<>D)and
                        (Way[k-1]<Way[k+1])
       then x:=Way[k+1]
       else x:=Way[k-1]+1;
       y:=x-k;
     While (x<length(AA))and(y<Length(BB))
                  and(AA[x+1]=BB[y+1])do
         begin inc(x);
          inc(y); end;
       Way[k]:=x;
//выход из функции нахождения средней точки
   If (x>=length(AA))and(y>=length(BB))
                        or (Way[k]>=Back[k])
        then begin
        u:=Back[k];
        v:=Back[k]-k;
        MiddlePoint:=2*D-1;
        exit;end;
     end;

//обратный путь
     For k:=-d to d do
    If (abs(k) mod 2)=(d mod 2)then
       begin
// переопределение нумерации диагоналей
                                в обратном пути
// для боле легкого понимания
       l:=k+delta;
      If (k=-d)or(k<>D)and
                        (Back[l-1]>Back[l+1])
       then u:=Back[l+1]-1
       else u:=Back[l-1];
       v:=u-l;
     While (u>0)and(v>0)and(AA[u]=BB[v])do
         begin dec(u);
          dec(v); end;
      Back[l]:=u;
//выход из функции нахождения средней точки
     If (u<=0)and(v>=0)or (Way[l]>=Back[l])
        then begin
        x:=Back[l];
        y:=Back[l]-l;
        MiddlePoint:=2*D;
        exit;end;
     end;
  end;
end;

// функция нахождения подпоследовательности
                                из имеющейся
// последовательности координат начала
                    и конца подпоследовательности
function out(Script:string;x,u:integer):string;
 var i:integer;
   st:string;
 begin
 st:='';
 for i:=x to u do
 st:=st+Script[i];
 out:=st;
 end;

// основное тело процедуры решения
 Procedure Solve(AA,BB:string);
 var x,y,u,v:integer;
 begin
 if (length(AA)>0)and(Length(BB)>0) then
   begin
// если в пути до срединной точки более
                    одного недиагонального ребра,
// то делим путь дальше
   if MiddlePoint(x,y,u,v)>1 then
     begin
```



```
     Solve(out(AA,1,u),out(BB,1,v));
     if (x>u)and(y>v)then write(Out(AA,u+1,x));
    Solve(out(AA,x+1,length(AA)),out(BB,y+1,length(BB)));
     end else
// если путь содержит менее двух различий,
                                то выводим
// последовательность минимальной длины
    if length(AA)<=length(BB) then
     write(AA) else write(BB)
   end;
 end;
```

По примерным подсчетам, времени на работу данной процедуры требуется в два раза больше, чем для описанной выше процедуры с циклами *For* и *While*, также время тратится на проверку всех условий внутри рекурсии. Время работы программы, как было сказано выше, не намного хуже лучших вариантов реализации. Теперь подсчитаем объем памяти. На хранение всех последовательностей в стеке рекурсии требуется объем равный 2 объемам, необходимым для хранения исходных последовательностей (так как алгоритм делит последовательность в среднем случае пополам, в худшем – сначала сохраняется большая часть, затем меньшая). Также для хранения координат точек в стеке требуется два массива на D элементов. В итоге необходимый объем памяти составляет: O((M+N)lgS+(M+N)lg(M+N)), где

(M+N)lgS – объем, необходимый для хранения последовательностей, (M+N) – длины последовательностей, lgS – байты, необходимые для хранения символа;

(M+N)lg(M+N) – объем, необходимый для хранения координат точек, (M+N) – общее количество точек в худшем случае, lg(M+N) – байты, необходимые для хранения координат символов.

На рис. 3 представлены две последовательности, разбитые алгоритмом.

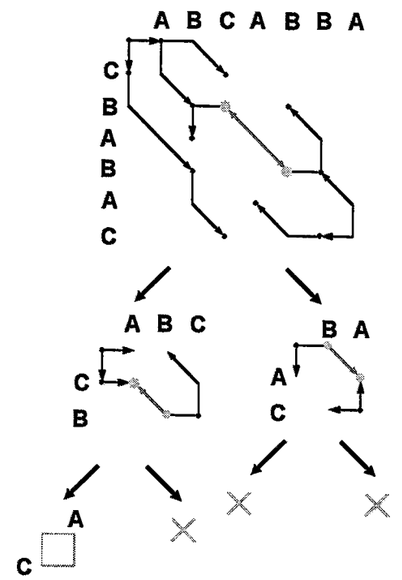


Рис. 3
Нахождение общих последовательностей по алгоритму 2б

Замечания

Приведенные выше алгоритмы достаточно широко используются для решения разнообразных задач. Но ни один из приведенных алгоритмов не является универсальным для различных последовательностей (кстати, элементами последовательности могут быть не только символы, но и слова, а также в роли последовательности может выступать массив чисел или состояний). Поэтому при решении задач следует комбинировать алгоритмы в зависимости от результата и входных данных задачи.

В случае, если набор входных данных неизвестен или вы уверены, что около 30-50% символов совпадут, то лучше всего воспользоваться квадратичным алгоритмом. Тогда он будет оптимальным алгоритмом по времени.

Если вам известно, что последовательности слишком большие для хранения их в двумерном массиве, то для решения задачи придется пожертвовать временем (существует измененный алгоритм Нудельмана-Вунша, работающий за большее время, но с линейным объемом – обращаться к автору статьи).

Алгоритм 2б хорошо подходит для последовательностей любой длины, но при условии, что две последовательности достаточно сильно совпадают (он использовался для нахождения мутации генов ДНК длиной более 200 000 элементов, но различия между двумя последовательностями не превышали 15%), в противоположном случае (совпадений менее 70%), на последовательностях большой длины, алгоритм существенно замедляется.

Для подобного случая хорошо подходит другой алгоритм Hunt'a и Szimansk'ого[5], который хорошо работает на последовательностях, отличающихся друг от друга, или последовательностях, алфавит которых достаточно широк.

Примечания

1. *Окулов С. М.* Программирование в алгоритмах. М.: БИНОМ. Лаборатория знаний, 2002.

2. *Eugene M.* An O(ND) Difference Algorithm and its Variations, www.algolist.manual.ru

3. *Окулов С. М.* Указ. соч.

4. *Eugene M.* Cit. op.

5. Алгоритм Ханта – Зиманского. www.algolist.manual.ru /Поиск/Общие подпоследовательности. Дистанция/ Алгоритм Ханта –Зиманского

С. Г. Кипров

АЛГОРИТМ ХАНТА – ЗИМАНСКОГО

В статье рассматриваются особенности одного из последних (по времени появления) алгоритма поиска наибольшей общей последовательности.

Метод выделения наибольшей общей последовательность LCS (longest common sequence) для двух входных строк X и Y Ханта – Зиманского зависит от количества совпадений символов (элементов) и длины общей последовательности, а время затраченное на работу данного алгоритма, вычисляется O(C*LogN), где C – количество общих символов, N – длина одной из двух сравниваемых последовательностей.

Входные данные. Даны две строки X и Y длин N и M.

Суть алгоритма. Сканируется одна из строк (лучше меньшей длины, для того чтобы время LogN было меньше). Если символ (элемент) находится в другой строке, то запоминаем его индекс и позицию, которую он займет в максимальной общей последовательности. Затем из полученного набора выбираем максимальную общую последовательность. Ниже представлен набросок алгоритма.

```
// инициализация
kk0 = 0
for s = 1 to n
    kks = n + 1
//вычисление |LCS(X, Y)|
for i = 1 to n
    for j = n downto 1
    if xi = yi

//найти s, для которого kks-1 < j < kks
    kks = j
print max s такое, что kks =/=n + 1 (?????)
```

Теперь все более подробно.

Инициализация № 1

Если последовательно будет производиться сравнение одной последовательности с несколькими, то имеет смысл произвести инициализацию № 1 только один раз, а затем использовать полученный результат.

Для ответа на вопрос о нахождении одного символа (элемента) последовательности в другой можно использовать линейный поиск символа (подстроки) в последовательности (алгоритмы Бойра-Мура, Рабина, Кнута-Морриса-Пратта в этом случае будут менее эффективны [1]), но это займет достаточно большое количество времени. Поэтому лучше ввести некоторую новую переменную множественного типа (*set of Char*) или массив бинарных элементов (*array [Char] of Boolean*).

Инициализация осуществляется проходом по строке и внесением текущего символа (элемента) в набор существующих (????)

```
Var
    MaskSet1 : Set of Char;
    MaskSet2 : array [Char] of Integer;

procedure SetInitialization;
begin
    for i := 1 to Length(Mask) do begin
// внесение элемента во множество просмотренных
    MaskSet := MaskSet1 + [Mask[i]];
    MaskSet2[Mask[i]] := True;
    end;
end;
```

Mask – одна из последовательностей.

MaskSet1, MaskSet2 – две переменные, отвечающие за нахождение символа в Mask.

Далее данная процедура будет немного изменена, для хранения индекса символа Mask. Таким образом, при введении данной процедуры сокращается время, потраченное на поиск символа в последовательности, в N раз (для худшего случая).

На рис. 1 представлен массив Boolean элементов, и результат после его заполнения.

Рис. 1
На рисунке пустые элементы равны False; элементы, равные True, присутствуют в исходной строке «BYB»

Инициализация № 2 (Массив указателей)

Для нахождения общей подпоследовательности символов максимальной длины (LCS) сначала необходимо найти все общие символы в последовательностях. Для этого просканировать вторую последовательность и при нахождении символа, принадлежащего первой, записать его в массив указателей [2].

Необходимо отметить, что в данной инициализации все элементы записаны в обратном порядке. Это необходимо для того, чтобы найти все вхождения одинаковых символов и записать их позиции в обратном порядке, для того чтобы не внести один элемент из одной последовательности, встречающийся в другой дважды, в максимальную общую последовательность. Таким образом, при использовании других структур данных необходимо записывать их в обратном поступлению элементов порядке, или идти с кона в начало в описанной ниже процедуре.

Ниже приведена процедура записи в массив.


```
type
    PElement = ^TElement;
    TElement = packed record
    SymbolIndex  : Integer;
    Next    : PElement;
    end;

    TMaskSet = array [Char] of Integer;

var
    PLetterArray : array [Char] of PElement;
    MaskSet      : TMaskSet;

procedure InitSymbolsIndexes;
var
    i       : Integer;
    Index   : Integer;
begin
    FillChar(PLetterArray, SizeOf(PLetterArray), 0);
    for i := 1 to Length(Buf) do begin
// берем общий элемент, получаем его индекс
    Index := MaskSet[Buf[i]];
    if Index > 0 then begin
// добавление элемента в стек
    AddElement(PLetterArray[Buf[i]], i);
    end;
    end;
end;
```

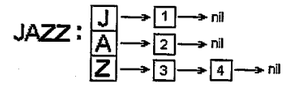


Рис. 2.
Формирование массива указателей для последовательности «JAZZ» и исходной «JAZZMAN»

Таким образом, в PLetterArray хранятся индексы общих символов для сравниваемых последовательностей. Объем, требуемый для PLetterArray, зависит только от количества общих символов. Поэтому, если не имеется ни одного общего символа, все указатели будут равняться nil, следовательно, PLetterArray будет занимать объем памяти только для собственного хранения.

Сборка общей подпоследовательности

Для сборки общей подпоследовательности символов будет использоваться ранее сформированный PLetterArray. Проходя по элементам PLetterArray, выбираем не нулевые указатели, которые дают нам позицию символа (SymbolIndex). Далее мы будем ставить его на свое место в общей подпоследовательности (на самом деле мы только поставим его в очередь на вхождение в общую подпоследовательность). Поиск индекса символа в массиве можно осуществлять как прямым, так и бинарным поиском. Для хранения информации о найденном элементе опять используем указатели [3].

После данной процедуры мы будем иметь связный список (нечто наподобие массива), в котором будут храниться индексы символа в исходных последовательностях и общей подпоследовательности.

Ниже приведены процедуры и структуры данных, реализующие вышесказанное.

```
    type
        PByteArray = ^TByteArray;
        TByteArray = array [0..MaxInt div 16] of Integer

        PCoinsiding = ^TCoinsiding;
        TCoinsiding = record
        Step    : Integer;
        CodeLetter    : Integer;
        CodeLetterMask      : Integer;
        Next    : PCoinsiding;
        end;

    var
        CommonPath: PByteArray;
        PLetterArray : array [Char] of PElement;
        Temp    : PElement;
        Coinsidings    : PCoinsiding;
    // максимальная длина подпоследовательности
        MaxLengthLCS        : Integer;

    // бинарный поиск элемента в массиве
    procedure FindPosition(x, y, j : Integer;
                                    var s : Integer);
    begin
        while y > x do begin
        if CommonPath^[(x + y) div 2] >= j then y :=
                                    (x + y) div 2
        else x := ((x + y) div 2) + 1;
        end;
        s := x;
    end;

    procedure Solve;
    var
        i, j, s    : Integer;
    begin
        MaxLengthLCS := 1;
    // проход по последовательности
        for i := 1 to Length(Mask) do begin
        Temp := PLetterArray[Mask[i]];
        while Temp <> nil do begin
        j := Temp^.SymbolIndex;
        s := Length(Buf);
        FindPosition(0, MaxLengthLCS, j, s);
        if (j < CommonPath[s]) then begin
        CommonPath[s] := j;
        MaxLengthLCS := Max (MaxLengthLCS, S + 1);
    // добавление элемента в стек
                    AddConsiding(Coinsidings, i, j, s);
```

```
    end;
    Temp := Temp^.Next;
    end;
    end;
end;
```

На рис. 3 представлен связный список для двух последовательностей «zeitgeost» и «preterit», сформированный по вышеописанной процедуре.

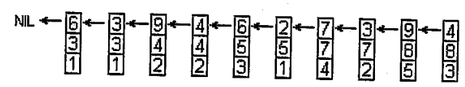


Рис. 3
Связанный список одинаковых элементов для двух последовательностей «zeitgeost» и «preterit» (первая ячейка – индекс символа в первой строке, вторая – индекс символа во второй строке, третья – индекс символа в LCS)

Сборка общей подпоследовательности максимальной длины

Имея связный список, в котором хранятся индексы символа в общей подпоследовательности можно легко получить всю подпоследовательность:

• так как нам уже известна длина LCS (*MaxLengthLCS*), находим в связном списке первый символ с этим индексом;

• далее идем по связному списку, пока не встречаем элемент с индексом, меньшим текущего индекса, запоминаем его и уменьшаем текущий индекс на 1;

• продолжаем поиск до тех пор, пока текущий индекс не станет равным 0.

Процедура нахождения одной общей подпоследовательности максимальной длины.

```
function GetSubSequence (MaxCommon: Integer): String;
  var
    SubString      : String;
  Begin
  // инициализируем общую строку
    SubString := '';
  // ищем в связном списке правый символ из LCS
    while Coinsidings ^.Step <> MaxCommon do
          Coinsidings := Coinsidings^.Next;
  // продолжаем поиск элемента с текущим
                              индексом LCS, пока не
  // соберем всю LCS и не кончится связный список
    while (MaxCommon > 0) and (Coinsidings <> nil)
                                        do begin
    if Coinsidings^.Step = MaxCommon then begin
  // если нашли символ с текущим индексом,
                        то добавляем его в строку и
  // уменьшаем текущий индекс
    SubString := Mask[Coinsidings^.
                              CodeLetterMask]
    Dec(MaxCommon);
    end;
    Coinsidings := Coinsidings^.Next;
    end;
    Result := SubString;
end;
```

Описанная выше процедура GetSubSequence использует связный список и выдает общую последовательность символов (рис. 4).

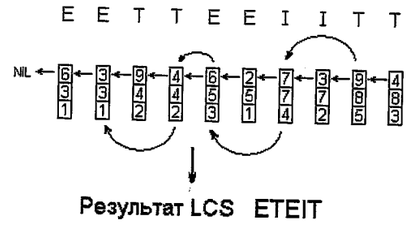


Рис. 4
Выделение LCS из связного списка для последовательностей «zeitgeist» и «preterit»

Ниже приведен пример вычисления LCS(preterit, zeitgeist) для описанного выше алгоритма, который выдает следующий список пар (i, j) для LCS (в обратном порядке): (8, 9), (7, 7), (5, 6), (4, 4), (3, 2).

.	j	1	2	3	4	5	6	7	8	9
i		z	e	i	t	g	e	i	s	t
1	p									
2	r									
3	e		●				o			
4	t				●					o
5	e		o				●			
6	r				\					
7	i			o				●		
8	t	\			o					●

Это проиллюстрировано в таблице: узлы указывают точки (i, j), такие, что $X_i = Y_j$. LCS формируется символами в соответствии с заполненными узлами, а именно, eteit.

Проведем анализ алгоритма:

• **объем**: при своевременном освобождении указателей объем памяти не превысит объем памяти необходимый для хранения всех общих символов, в противном случае понадобится до трех подобных объемов;

• **время**: алгоритм, приведенный ниже, достаточно хорошо оптимизирован и требует время, равное O(CLogN + Nk), где C – количество общих символов, N – длина одной из последовательности, k – некий постоянный коэффициент. Коэффициент k может быть еще немного уменьшен. Алгоритм Ханта – Зиманского быстро работает на последовательностях с широким алфавитом и на последовательностях с небольшим количеством одинаковых символов;

• **алгоритм** Ханта – Зиманского может быть применен при неточном поиске файлов, текста, поиске ошибок, но в случае, когда алфавит узок, время работы алгоритма может превысить квадрат (например, на последовательностях X ='аааааааа' и Y =' аааааааа' время работы будет O(N^2*LogN)).

Заметки

• заметим, что все процедуры были даны без освобождения указателей;

• вышеописанные процедуры находят только одну LCS, но данный алгоритм легко изменить так, чтобы он находил все LCS (для этого необходимо изменить структуру PCoinsiding до дерева указателей, лист дерева должен обладать списком указателей на нижележащий уровень).

Примечания

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 1999.

2. *Окулов С. М.* Основы программирования. М: ЮНИМЕДИФСТАЙЛ, 2002.

3. Там же.

В. Ю. Колеватов

О ЗАДАЧЕ НАХОЖДЕНИЯ МАКСИМАЛЬНО-НЕЗАВИСИМЫХ МНОЖЕСТВ ГРАФА

В статье приводится конструктивная критика метода объяснения алгоритма в работе С. М. Окулова «Конспекты занятий по информатике (алгоритмы на графах)» [1].

Теория. Дан неориентированный граф G=(V,E). Независимое множество вершин есть множество вершин графа G, такое, что любые две вершины в нем не смежны, т. е. никакая пара вершин не соединена ребром.

Независимое множество называется максимальным, когда нет другого независимого множества, в которое оно бы входило.

У максимально-независимого множества графа есть такое свойство:

S – максимально-независимое множество графа G,

U – множество вершин, смежных с вершинами множества S

G=S+U

Отсюда следует и метод решения задачи нахождения максимально-назависимых множеств (МНМ), то есть находится такое множество несмежных вершин графа, которое в совокупности со своими смежными вершинами образует граф. Основной идеей сокращения перебора является последовательное расширение независимого множества. Если его нельзя будет расширить, то это множество и есть максимально- независимое.

В работе [1] приведена следующая логика решения данной задачи:

Переменные

Type Sset=set of 1..N;

a:Array[1..N] of set of 1..N; i-й элемент массива множество вершин, смежных с вершиной i. *

Ss – массив решения. Решением являются первые k элементов;

k – переменная, указывающая на порядковый номер вершины, которую мы хотим включить в решение;

qp – множество вершин, которые еще не были рассмотрены в процессе перебора;

qm - множество вершин, которые уже были просмотрены в процессе перебора;

Gg – множество кандидатов на расширение текущего независимого множества.

Опишем логику вывода решения из массива Ss:

```
Procedure Print(k:byte);
  Var i:integer;
  Begin
        Writeln;
        For i:=1 to k do write(Ss[i]:2);
  End;
```

Для решения нам понадобится функция подсчета количества элементов в множестве.

```
function number(a:sset):byte;
  var i,cnt:byte;
begin
  cnt:=0;
  for i:=1 to n do if i in A then inc(cnt);
  number:=cnt;
end;

{основная процедура решения задачи}
Procedure FIND (k:integer; qp,qm:sset);
  var i,delt,j:byte;
    gg:sset;
  begin
```

КОЛЕВАТОВ Василий Юрьевич – студент IV курса факультета информатики ВятГТУ

* Данный метод описания графа используется, так как нам часто придется рассматривать вершины, смежные с данной.

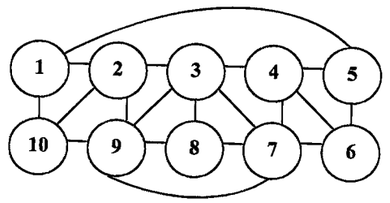
№	K	Qp	Qm	Gg	Ss	Примечания
1	1	[1..10]	[]	[1..10]	(1)	Вбираем первую вершину
2	2	[3,4,6..9]	[]	[3,4,6..9]	(1,3)	Включаем вершину 3
3	3	[6]	[]	[6]	(1,3,6)	Включаем вершину 6
4	4	[]	[]	[]	(1,3,6)	Вывод первого мн-ва и выход в предыдущую копию процедуры FIND
5	3	[]	[6]	[]	(1,3,6)	Выход в предыдущую копию Find
6	2	[4,6..9]	[3]	[4,7..9]	(1,4)	Исключаем вершину 3 из Ss и включаем ее в Qm. Продолжает работу цикл while. Он включает следующую вершину – 4.
7	3	[8,9]	[]	[8,9]	(1,4,8)	Вывод второго МНМ
8	3	[9]	[8]	[9]	(1,4,9)	Вывод третьего МНМ и выход в предыдущую копию Find
9	2	[6..9]	[3,4]	[6,7]	(1,6)	
10	3	[8,9]	[3]	[]	(1,6)	Первое использование блока АА. Этот блок используется для сокращения перебора. Он формирует множество Gg как пересечение множеств Qp и A[j], где j из Qm. Причем мощность множеств минимальна.
	3	[8,9]	[3]	[8,9]	(1,6)	В итоге формирования I приняло значение 3
	3	[8,9]	[3]	[8,9]	(1,6,8)	
11	4	[]	[]	[]	(1,6,8)	Вывод МНМ
12	3	[9]	[3,8]	[9]	(1,6,9)	
13	4	[]	[]	[]	(1,6,9)	Вывод МНМ
14	2	[7..9]	[3,4,6]	[7]	(1,7)	
15	3	[]	[]	[]	(1,7)	Вывод МНМ
16	1	[2..10]	[1]	[2,5,10]	(2)	
17	2	[4..8]	[]	[4..8]	(2,4)	
18	3	[8]	[]	[8]	(2,4,8)	
19	4	[]	[]	[]	(2,4,8)	Вывод МНМ
20	2	[5..8]	[4]	[5..7]	(2,5)	
21	3	[7..8]	[]	[7..8]	(2,5,7)	
22	4	[]	[]	[]	(2,5,7)	Вывод МНМ
23	3	[8]	[7]	[8]	(2,5,8)	
24	4	[]	[]	[]	(2,5,8)	Вывод МНМ
25	2	[6..8]	[4,5]	[6]	(2,6)	
26	3	[8]	[]	[8]	(2,6,8)	
27	4	[]	[]	[]	(2,6,8)	Вывод МНМ
28	1	[3..10]	[1,2]	[2,5,10]	(5)	
29	2	[3,7..10]	[2]	[3,9,10]	(5)	Вновь включается в работу блок АА. Qm состоит из1го элемента – 2, поэтому он и будет выбран как минимальный.
	2	[3,7..10]	[2]	[3,9,10]	(5,3)	
30	3	[10]	[]	[10]	(5,3,10)	
31	4	[]	[]	[]	(5,3,10)	Вывод МНМ


```
if (qp<>[]) or (qm<>[]) then begin
    {форомирование множества Gg}
    if qm<>[] then begin
        delt:=n+1;
        for j:=1 to n do
            if j in qm then
            if number(a[j]*qp)<
                        delt then begin
                i:=j;
                delt:=number(a[j]*qp);
            end;
        gg:=qp*a[i];
    end
    else gg:=qp;
    {поиск следующей вершины}
    i:=1;
    while i<=n do begin
        if i in gg then begin
            ss[k]:=i;
            find(k+1,qp-a[i]-[i],
                        qm-a[i]-[i]);
            qp:=qp-[i];
            qm:=qm+[i];
            if number(qp*a[i])<
                        number(gg)
                then gg:=qp*a[i];
    end;
    inc(i);
    end;
end else print(k);
end;
```

АА

Б

Но в вышеуказанной книге приведен пример, который не в полной мере раскрывает необходимость использования блоков АА и Б, а также целесообразность введения множества Qm. Поэтому попытаемся привести новый пример и уточнить вышеуказанные моменты методом трассировки.



Рассмотрим граф на рисунке. Матрица А для него будет выглядеть следующим образом:

A[1]=[2,5,10]
A[2]=[1,3,9,10]
A[3]=[2,4,7..9]
A[4]=[3,5..7]
A[5]=[1,4,6]
A[6]=[4,5,7]
A[7]=[3,4,6,8,9]
A[8]=[3,7,9]
A[9]=[2,3,7,8,10]
A[10]=[1,2,9]

Используем метод трассировки (см. таблицу).

Примечание. В блоках АА и Б выбирается минимальное по мощности множество, так как, выполняя такой шаг, мы каждый раз, сокращая множество Gg, сокращаем количество итераций в рассмотрении, а значит, и сокращаем перебор.

Примечания

1. *Окулов С. М.* Конспекты занятий по информатике (алгоритмы на графах). Киров, 1996.

И. В. Малкова

ОБ АЛГОРИТМАХ СОРТИРОВКИ

В статье уточняется методика изложения темы, приведенная в ряде работ С. М. Окулова [1].

Алгоритм в программировании – это описание метода решения задачи. Цель данной статьи заключается в том, чтобы показать, как можно улучшить изложение материала об алгоритмах сортировки из работы [2]. Рассмотрим сортировку методом вставки.

В процессе анализа попытаемся ответить на несколько вопросов:

- Насколько хороши разработанные алгоритмы?
- Как улучшить имеющиеся алгоритмы и программы?
- Эффективна ли программа с точки зрения затрат времени и памяти?
- Какие ограничения можно ввести, чтобы улучшить алгоритм?

Идея сортировки методом простой вставки не сложная: считается, что перед рассмотрением элемента a[i] предыдущие элементы a[1], a[2], ..., a[i-1] уже упорядочены, и подбирается место для a[i] записи.

Сортировка массива начинается со второй записи. На i-м шаге a[i] сравнивается по очереди с a[i-1], a[i-2],... до тех пор, пока выполняется условие a[i] < a[j] (j = i-1,i-2, ...), или достигнут левый конец упорядоченного подмассива (j = 1, a[i] < a[1]). Выполнение условия a[i] >= a[j] означает, что запись a[i] нужно вставить между a[j] и a[j+1]. Тогда записи a[j+1], a[j+2], ..., a[i-1] сдвигаются на одну позицию и запись a[i] помещается в позицию j+1. На каждом шаге отсортированная часть массива увеличивается. Сортировка закончится на N-1 шаге, если в массиве N элементов.

Программа сортировки может выглядеть так:

МАЛКОВА Инна Владимировна – студентка III курса факультета информатики ВятГТУ

```
Procedure Sort;
 var i, j :integer;
 Begin
  For i := 2 to N do Begin
     j := i;
    While ( j > 1 ) and ( a[j] < a[j-1] ) do Begin
       Swap(a[j], a[j-1]);
        Dec(j);
      end;
  end;
 end;
```

Массив а опишем так:

```
Type mas = array[1..N] of integer;
Var a: mas;
```

Процедура Swap содержит следующие операторы присваивания:

```
t : = a[j];
a[j] := a[j-1];
a[j-1] := t;
```

Обратим внимание на переменную t в процедуре Swap, в нее каждый раз посылается одно и то же значение – то, которое было в начальный момент в ячейке a[j]. Поэтому из первого варианта программы можно убрать процедуру Swap и изменить оператор сравнения.

```
Procedure Sort;
 var i, j, t : integer;
 Begin
  For i := 2 to N do Begin
     j:= i;
     t:= a[j];
    While (j > 1) and (a[j-1] > t) do Begin
       a[j]:= a[j-1];
        Dec(j);
      end;
     a[j]:= t;
  end;
 end;
```

Попробуем поэкспериментировать со вторым условием в цикле: While **(a[j-1] > t)**. Изменим это условие на **a[j-1]>=t.**

На первый взгляд, программа работает так же, но мы будем лишний раз заходить в цикл While и менять местами одинаковые элементы. Наша модификация не улучшит работу программы, значит, оставим это условие без изменений. Если записи с одинаковыми ключами не меняются местами, говорят, что сортировка **устойчивая**. Сделаем вывод, что при правильном написании программного кода сортировку методом простой вставки можно назвать устойчивой.

Ещё раз взглянем на предыдущий вариант процедуры. Нужна ли нам переменная t? При использовании **t** происходит бесполезная трата времени на её запоминание и последующее чтение с целью обмена значениями.

Чтобы избавиться от ненужной переменной, изменим наш массив A, сделав его типом **array[0..N] of integer**.

Рассматриваемый i-й элемент теперь будет записываться на нулевое место и играть роль «барьера». Изменить процедуру простой вставки на процедуру с барьерным элементом очень просто.

В общем случае время выполнения алгоритма пропорционально количеству операций сравнения (a[j-1] > t) и числу перемещений элементов по массиву (тело цикла While).

У нас в массиве **N** элементов. Основной цикл **(For i:=…)** идет от 2 до N. Второй цикл **(While)** идет от **i** по убывающей до единицы. В худшем случае самый маленький элемент стоит на последнем, N-м, месте в массиве. Значит, надо сделать **N-1** сравнение и перемещение, чтобы поставить элемент на свое (первое) место. Получается, что в худшем случае сортировка находится в квадратичной зависимости от числа элементов массива, то есть $C = O(N^2)$.

Изменения, которые были сделаны, незначительны. Но для небольших массивов и этих модификаций будет достаточно: подобные массивы отсортируются быстро. Этот метод также эффективен, когда записи частично упорядочены, т.е. записи находятся недалеко от своих конечных положений.

Когда при сортировке методом простых вставок обрабатывается i-й элемент, его нужно сравнить в среднем примерно с i /2 ранее отсортированными элементами.

При больших N этих сравнений получается слишком много. Поэтому в 1946 году *Джон Мочли* (John Mauchly) предложил метод бинарных вставок, чтобы сократить число сравнений.

Этот способ основывается на том факте, что мы вначале ищем место, куда нужно вставить элемент, а затем уже вставляем его. Это дает существенный выигрыш по числу сравнений.

Поскольку все элементы массива перед записью a[i] уже упорядочены, то можно использовать более эффективный метод поиска места для a[i]. В данном случае это алгоритм бинарного поиска. Например, если вставляется 64-я запись, то можно сравнить ее с 32-й, и если 64-я запись окажется меньше, то сравнить ее уже с 16-й, а если окажется больше, то сравнить с 48-й и т. д. Таким образом, место 64-й записи будет определено в худшем случае за шесть сравнений.

Для реализации этого алгоритма, кроме основной процедуры, нам потребуется вспомогательная (Change ()), которая будет находить место в массиве для нового элемента. Логика поиска нужного места такая же, как в методе простой вставки.

Основная процедура сортировки может выглядеть так:

```
Procedure Sort;
var i,j,r,m:integer;
Begin
m:=1;
For i:=2 to N do Begin
```



```
 If a[i]<a[i-1] then Begin
   j:=1; r:=i-1;
  While r-j>1 do Begin
    m:=(r+j) div 2;
    If a[m]>a[i] then r:=m
      else j:=m;
  end;
 If (a[m]>a[i]) then
  If (a[m-1]>a[i]) and (m<>1) then Change
                                   (a,i,m-1)
    else Change(a,i,m)
  else Change(a,i,m+1);
 end;
end;
```

Зачем нам в основной процедуре условие

If **(a[m-1]>a[i])** *and* **(m<>1)** *then… ?*

Попробуем обойтись без него и запишем последнее условие так:

```
…
If (a[m]>a[i]) then Change(a,i,m-1)
 else Change(a,i,m+1);
…
```

С таким условием программа будет работать примерно в половине случаев. Все будет зависеть от того, как мы подберем элементы массива.

Если первый элемент больше второго, их необходимо поменять местами. И менять мы будем сначала второй элемент с первым, а потом получится, что надо будет поменять первый с нулевым, таким образом, мы выйдем за пределы массива. Программа будет работать неправильно.

Может возникнуть ещё один вопрос: может быть, при условии, что a[m]>a[i], вызывать процедуру Change(a, i, **m**)? Но при таком условии первый элемент всегда будет оставаться на своём месте, не меняясь ни при каких обстоятельствах.

Метод бинарных вставок решает временную проблему лишь наполовину. Когда мы находим место для i-го элемента, нам все равно приходится делать перестановки. Время, которое мы тратим на перестановки, так и остается O(N), как и в методе простых вставок. Зато вполовину уменьшается число сравнений, то есть «отсеивается» половина данных, что займет O(lg N) времени. Получается, что сортировка бинарными вставками работает за время, пропорциональное N*Lg N.

Так как число перестановок остается по-прежнему достаточно большим, Х. Неглер (H. Negler) показал, что метод бинарных вставок не рекомендуется использовать при сортировке более N=128 элементов, так как с ростом N зависимость от N^2 начнет преобладать.

Чтобы исправить недостаток сортировки бинарными вставками, необходим метод, в котором элементы перемещаются большими скачками. Такой метод предложил Д. Шелл (D. Shell) в 1959 году.

Идея состоит в обмене элементов, расположенных не только рядом, как в алгоритме простых вставок, но и далеко друг от друга, что значительно сокращает общее число операций перемещения элементов. Для примера возьмем файл из 16 элементов. Сначала просматриваются пары с шагом 8. Это пары элементов 1-9, 2-10, 3-11, 4-12, 5-13, 6-14, 7-15, 8-16. Если значения элементов в паре не упорядочены по возрастанию, то элементы меняются местами. Назовем этот этап 8-сортировкой. Следующий этап - 4-сортировка, на котором элементы в файле делятся на четверки: 1-5-9-13, 2-6-10-14, 3-7-11-15, 4-8-12-16. Выполняется сортировка в каждой четверке. Следующий этап – 2-сортировка, когда элементы в файле делятся на 2 группы по 8: 1-3-5-7-9-11-13-15 и 2-4-6-8-10-12-14-16. Выполняется сортировка в каждой восьмерке. Наконец, весь файл упорядочивается методом простых вставок. Поскольку дальние элементы уже переместились на свое место или находятся вблизи от него, этот этап будет значительно менее трудоемким, чем при сортировке вставками без предварительных «дальних» обменов.

Процедуру этой сортировки для приращений 8-4-2-1 можно реализовать так:

```
Procedure Sort;
 var i, j, k, up, t, help :integer;
 Begin
   k:=n div 2;
   While k>0 do Begin
      up:=n-k;
      repeat
        t:= 0;
        For i:= 1 to up do
        If a[i]>a[i+k] then Begin
            help:=a[i];
           a[i]:=a[i+k];
           a[i+k]:=help;
            t:=i;
         end;
       up:=t-k;
     Until t = 0;
     k:=k div 2;
  end;
 end;
```

Если изменить условие **a[i] > a[i+k]** на **a[i] >= a[i+k]**? В этом условии мы меняем местами элементы, если **a[i+k]** больше **a[i]**. Значит, при изменении условия, у нас будут меняться местами одинаковые элементы. Такое условие не принесет никакой пользы.

Хотя этот метод легко объяснить, его формальный анализ довольно труден. До сегодняшнего момента никому не удалось выполнить точный анализ алгоритма. Временная оценка сортировки зависит от выбранной последовательности шагов. Но пока также не удалось определить и наилучшую последовательность. Д. Кнут описал математические исследования, по которым временная оценка алгоритма близка к $N^{1,25}$.

13 Зак. 2868

Ещё одним важным этапом улучшения программы является подбор подходящих структур данных. Выбор «правильных» структур данных позволит избежать ненужных операций. Связный список прекрасно подходит для реализации сортировки методом простых вставок. Так как мы просматриваем элементы в одном направлении, можно использовать список с однонаправленной связью. Чтобы вставить элемент на своё место, надо просто изменить несколько связей.

Рекомендации по выбору правильного метода сортировки

Мы рассмотрели несколько видов сортировок. Необходимо уметь выбирать правильные методы для конкретной задачи. Ниже приведено несколько рекомендаций по выбору метода сортировки:

• Если сортируется небольшой объем данных (N < 100), то рекомендуется выбирать простой метод сортировки, так как сложные алгоритмы имеют сложный код и не эффективны при малом количестве сортируемых элементов.

• Если сортируются элементы большого размера, то рекомендуется использовать специальные таблицы, с помощью которых осуществляется доступ к самим элементам (например, это могут быть указатели на элементы или их порядковые номера), причем ключ, по которому сортируются данные, может входить или не входить в данные такой таблицы. После сортировки таблицы можно или переставить исходные данные по таблице, или оставить все как есть и осуществлять дальнейший доступ к данным по уже отсортированной таблице.

• Если сортируются данные в файле, то необходимо учитывать, что большая часть времени будет тратиться на чтение, запись элемента и перемещение по файлу. В такой ситуации методы вставок являются не эффективными, так как требуют большое число перестановок.

• Если используются структуры данных, отличные организацией доступа от массива, то необходимо выбирать метод сортировки, наиболее подходящий для данной структуры.

• Рекомендуется выбирать алгоритм сортировки с учетом предполагаемого входного состояния данных (является таблица частично отсортированной, отсортированной в обратном порядке и т. д.).

Примечания

1. *Окулов С. М., Пестов А. А.* 100 задач по информатике. Киров, 2000; *Окулов С. М.* Основы программирования. М., 2002.

2. *Кнут Д.* Искусство программирования. Т. 3. Сортировка и поиск. М.; СПб.; Киев, 2001.

А. Г. Меркулов

ОБ ИГРЕ «ЖИЗНЬ»

В статье рассматривается вопрос о том, при каких начальных комбинациях «жизнь» продолжается бесконечно или вырождается, остался открытым.

В эту игру играют на неограниченном клеточном поле. Каждая клетка может иметь два состояния: либо в ней есть жизнь, либо она безжизненная. Данная клетка имеет 8 соседей: клетки по горизонтали, вертикали и диагонали, которые имеют с клеткой общее ребро или вершину. Жизнь течет поколениями. За одно поколение в каждой клетке происходит изменение. Если количество ее соседей составляет 0 или 1 живых клеток, то данная клетка умирает от одиночества. Если у клетки 2 соседа, то ее состояние остается прежним, а если с ней граничат 3 живые клетки, то в ней появляется жизнь. И наконец, если у клетки более 3 соседей, то жизнь в клетке прекращается от «перенаселения».

Реализация этой программы языком программирования достаточно очевидна. Используется двумерная матрица, в которой хранится игровое поле. В каждой клетке может стоять либо 1, если есть жизнь, либо 0, если, соответственно, жизни нет.

```
type MyArray = array [0..nmax+1, 0..nmax+1] of 0..1;
```

Далее для проверки состояние каждой ячейки поля при переходе в новое поколение нам необходима процедура, в которой будет производиться пересчет значений.

```
for i := 1 to xmax do begin   {цикл пробега по всем строкам}
for j := 1 to ymax do begin   {цикл пробега по всем столбцам}
 Count := 0;
 for k:=1 to 8 do
 if a[i+DX[k],j+DY[k]]=1 then Inc(Count);
                              {считаем всех соседей}
 case Count of
 0..1, 4..8 : lower[j] := 0;
 2          : if a[i,j] = 1 then lower[j] := 1
 else lower[j] := 0;
 3          : lower[j] := 1;
 end;                         {меняем значение клетки}
 end;
 for j := 1 to ymax do begin
 a[i-1, j] := upper[j];
 upper[j]  := lower[j];
 end;
 end;
 for j := 1 to ymax do a[xmax,j] := lower[j];
```

МЕРКУЛОВ Алексей Георгиевич – студент IV курса факультета информатики ВятГТУ

Причем стоит помнить, что при вычислении значения клетки нам необходимо пользоваться данными только текущего поколения, и результаты пересчета предыдущей строки не должны оказывать никакого влияния на вычисления, для этого в программу были добавлены два одномерных массива:

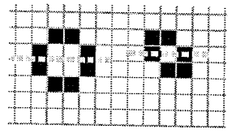
```
upper, lower : array [0..nmax+1] of 0..1;
```

в которых хранятся значения клеток в следующем поколении.

Ввод начального и вывод текущего поколения игрового поля осуществляется через файл.

Множество устойчивых фигур

Прежде чем приступать к нашему исследованию, давайте введем понятие симметричности. *Симметричной* называется фигура, одна из частей которой является зеркальным отражением другой. Соответственно, если существует отражение одной из частей, то должна существовать и ось, которая разделяет эти две половины. В зависимости от того, как расположена эта ось, выделим два вида отражения или симметричности: горизонтальная и диагональная. *Горизонтальным* будем называть отражение, когда ось проходит параллельно границам клеток поля. *Диагональным* отражением соответственно считается отражение, когда ось пересекает клетки поля по диагонали. Приведем пример. Фигуры, изображенные на рис. 1, являются горизонтально симметричными. Теперь рассмотрим пример с фигурой, обладающей диагональной симметрией (рис. 1б). Эта фигура также примечательна тем, что не обладает горизонтальной симметрией, в отличие от предыдущего примера, где фигуры были «двойные» (рис. 1а).



а

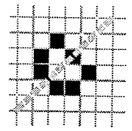


Рис. 1

б

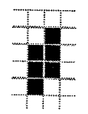
Еще один важный момент. Все остальные виды симметрии, которые характерны для устойчивых фигур, могут быть получены путем поворота оси симметрии на 90° либо путем комбинации двух видов отражения.

От теории перейдем к непосредственному изучению устойчивых фигур. Но прежде чем начать с простейших форм жизни, которые состоят из трех или четырех элементов, стоит сразу сказать, что параллельно мы рассмотрим устойчивость этих фигур к изменениям, к добавлению новых (дополнительных) элементов.

1. Итак, первая фигура – это куб (рис. 2а). Фигура достаточно примечательная. Как вы можете заметить, она обладает двумя симметриями – горизонтальной и диагональной. Поэкспериментируем с этой фигурой, добавив к ней лишних клеток. Если лишняя клетка имеет с кубом общую сторону, то через несколько поколений куб пропадает, если же общей является точка по диагонали, то куб выразится в фигуру 4. Уже сейчас можно выдвинуть предположение: **«Если фигура симметрична, то при добавлении симметричных элементов ее развитие будет либо продолжаться бесконечно, либо она выродится в набор устойчивых фигур»**. И действительно, добавление клетки по диагонали сохраняет симметрию фигуры, а клетка со смежной стороной делает ее асимметричной. А если добавить две точки симметрично, то куб превращается в устойчивую фигуру 10 (рис. 2б).

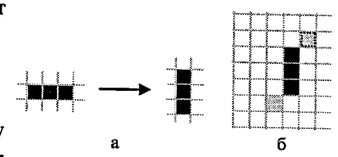
а

Рис. 2



б

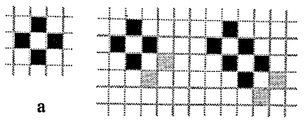
2. Следующая фигура имеет два состояния, переходя из одного в другое в единицу времени (рис. 3а). Давайте проверим наше утверждение. Фигура обладает горизонтальной симметрией и не обладает диагональной. Т. е., добавив два симметричных элемента по диагонали, мы получим вырождение фигуры уже через пару поколений (рис. 3б). В связи с этим к утверждению стоит внести небольшое дополнение. Итак **«Если фигура обладает диагональной симметрией, то добавление элементов, нарушающих эту симметрию, приведет к вырождению фигуры»**.



а б

Рис. 3

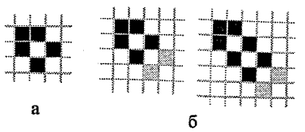
3. Эта фигура (рис. 4а) интересна в том плане, что добавлением нескольких элементов можно получить бесконечное множество фигур, причем все они будут устойчивыми, независимо от числа (рис. 4б). Это свойство фигуры только подтверждает нашу теорию, и многочисленные эксперименты являются тому примером. Перечислять их не имеет смысла, можете поэкспериментировать сами.



а б

Рис. 4

4. Фигура (рис. 5а) очень похожа на фигуру 3, и также можно получить бесконечное множество устойчивых фигур, используя тот же при-

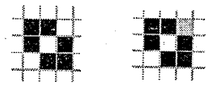


а б

Рис. 5

ем (рис. 5б). Как вы можете заметить, она диагонально симметрична и новые элементы не нарушают ее симметрии.

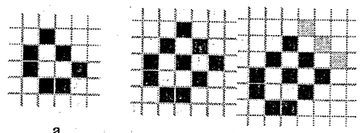
5. Следующая фигура (рис. 6а) с двойной симметрией. При добавлении одного элемента (рис. 6б) через поколение переходит в фигуру 6.



а б

Рис. 6

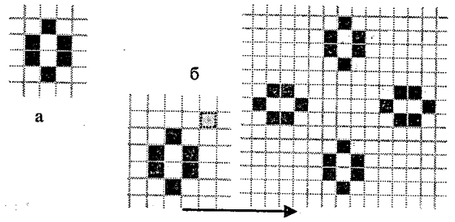
6. Также устойчивая фигура (рис. 7а), но прием добавления нескольких симметричных элементов (как и у фигур 3 и 4) работает лишь в самом первом случае, остальные же фигуры не являются устойчивыми (рис. 7б).



а б

Рис. 7

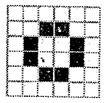
7. По поводу этой фигуры (рис. 8а) стоит сказать, что добавление элемента позволяет наблюдать удивительный процесс развития, который происходит симметрично и, в конце концов, вырождается в четыре исходные фигуры (рис. 8б).



а б

Рис. 8

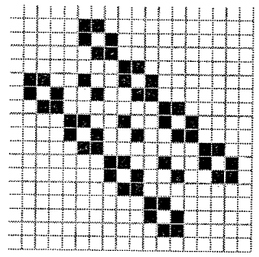
8. Фигура (рис. 9а), похожая по своим свойствам на предыдущую. Если добавить 2 клетки по диагонали (рис. 9б), то фигура изменится, и через несколько поколений перейдет другую устойчивую конструкцию (рис. 9в).



а

Рис. 9

Если вам стало интересно, то можете в домашних условиях поэкспериментировать и посмотреть, во что превращается фигура, изображенная на рисунке (для успешного проведения опыта вам необходимо достаточно большое поле – 50x50).



в

Рис. 9

Несколько слов о динамических фигурах, которые с течением времени меняют свою форму, но которые также могут считаться устойчивыми формами жизни. Об одной из таких фигур мы уже говорили, это фигура 2. Приведем еще несколько новых.

9. Данная фигура (рис. 10) имеет два состояния, которые поочередно принимает с течением времени.

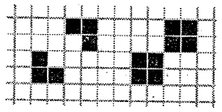


Рис. 10

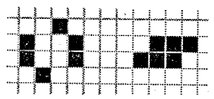


Рис. 11

10. Фигура, у которой тоже два состояния (рис. 11).

11. А теперь еще одна динамическая фигура, которая заметно отличается от всех рассмотренных ранее динамических фигур (рис. 12). Дело в том, что у эта фигура, которая обладает четырьмя устойчивыми состояниями, способна передвигаться по клеточному полю, переходя последовательно из одного состояния в другое. Направление ее движения зависит от того, как расположены элементы.

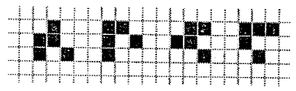


Рис. 12

12. А теперь пару слов о фигуре, которая достаточно громоздка по своей величине (рис. 14а), но она может быть поставлена в ряд с фигурами 3 и 4. Несмотря на такие размеры, она обладает устойчивостью. Можно заметить, что при добавлении симметричных элементов она по-прежнему будет оставаться устойчивой (рис. 14б). Эта фигура получается путем добавления всего лишь одной клетки к фигуре, изображенной на рис. 13. Причем стоит сказать, что та точка, которая была добавлена, не нарушила диагональной симметричности данной фигуры.

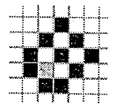
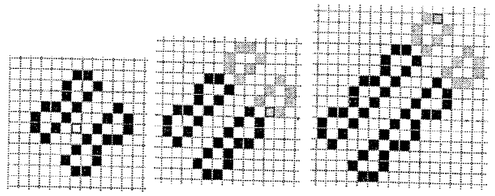


Рис. 13

Итак, наше предположение о том, что добавление симметричных элементов к симметричной фигуре не нарушает устойчивости, подтвердилось. Стоит также отметить, что эффект симметричности заметен также и в других элементах игры. Если последить за развитием фигуры, то можно заметить такую закономерность. Если фигура обладает диагональной симметрией, то ее развитие будет проходить симметрично, относительно ее оси симметрии, такая же ситуация и с горизонтально-симметричной фигурой.



а б

Рис. 14

М. В. Перевозчиков

ОСОБЕННОСТИ РАБОТЫ С АВЛ-ДЕРЕВЬЯМИ

Предложен алгоритм работы с АВЛ-деревом отличающийся от алгоритма Н. Вирта, классика Computer Science.

При работе с большими массивами данных зачастую используют некоторые сравнительные свойства этих данных для ускорения поиска в таких массивах. Примером может служить сортировка массива чисел (численные данные) для использования бинарного поиска на этом массиве, который, как известно, значительно быстрее, чем линейный.

Суть бинарного поиска представляется как поиск по двоичному дереву, работа с которым зачастую более удобна, нежели работа с массивом, так как при вставке/удалении элемента приходится «сдвигать» или «раздвигать» массив, что также иногда неприемлемо, потому что быстрым остается лишь поиск по массиву, т. е. массив удобен лишь тогда, когда он абсолютно статичен, чего в природе практически не встречается. Еще одним аргументом против линейных массивов является статичность их размеров, что означает сложность работы с ними при произвольном и неограниченном росте объема накапливаемых данных. Примером может служить учет файловых операций на жестком диске или научных открытий. Оба процесса носят почти случайный характер, и их объем ограничен лишь футуристическими предположениями, которые, кстати, если и верны, то на небольшой (по крайней мере, конечный) промежуток времени. Таким образом, хранение данных в массиве может быть очень неудобным, поэтому часто используется представление данных в виде деревьев.

Наиболее простой пример дерева – *идеально сбалансированное двоичное дерево*. Его идеальность заключается в том, что для каждого узла выполняется условие: количество левых и правых потомков узла совпадает. Такой критерий делает рассматриваемую структуру больше теоретической, чем практически применимой, так как при малейшем изменении в структуре дерева необходимо перестроить почти все дерево, что порой не быстрее, чем работа с массивом, не говоря уже о сложности этой операции. Но опять же при статичности идеально сбалансированное дерево дает результат такой же, как при использовании массива, с той лишь разницей, что если данные все-таки добавляются или удаляются, но достаточно редко, то дерево все же более «поворотливо» в этом плане, так как деревья чаще всего представляются в памяти как динамические структуры и проблем с «раздвижкой», как в массиве, не возникает. Но, как уже отмечалось, из-за жесткости критерия своей идеальности такие деревья почти всегда теоретические и на практике применяются редко.

Самым простым методом довести двоичные деревья до практического использования явилось ослабление критерия идеальности, что, несомненно, приводило к незначительным (по сравнению с линейным поиском) потерям в скорости поиска по таким деревьям. Наиболее любопытным оказался перенос критерия с количества узлов на высоту левого и правого поддеревьев узла. Примерами могут служить *АВЛ–деревья*, в которых используется высота непосредственно, или *красно-чёрные деревья*, в которых используется так называемая «черная высота». Такие деревья достаточно удобны в использовании и пренебрежимо мало уступают в скорости идеально сбалансированным.

В 1962 г. советские ученые Г. М. Адельсон-Вельский и Е. М. Ландис предложили упрощение критерия идеальной сбалансированности: дерево сбалансировано тогда и только тогда, когда для любого узла число дочерних уровней левого поддерева отличается от числа дочерних уровней правого поддерева не более чем на единицу. Деревья, подходящие под такой критерий, были названы АВЛ-деревьями (по первым буквам фамилий авторов).

Такая формулировка критерия сбалансированности существенно облегчает реализацию балансировки при работе с АВЛ-деревьями, кроме того, балансировка АВЛ-деревьев требует гораздо меньшего времени, чем построение идеально сбалансированного дерева. Также очевидно, что АВЛ-балансировка обеспечивает достаточно скоростной поиск по дереву.

Идея балансировки дерева по АВЛ очень проста: *если при добавлении/удалении узла обнаружи-*

ПЕРЕВОЗЧИКОВ Максим Валерьевич – студент III курса факультета информатики ВятГГУ

вается, что для какого-либо узла высота одного поддерева превзошла высоту другого более чем на единицу, то необходимо «повернуть» дерево относительно такого узла в сторону меньшей высоты.* То есть вся балансировка сводится к таким поворотам. Также следует заметить, что в методе фигурирует только понятие разницы высот левого и правого поддеревьев, что дает надежду, что если для выполнения поворотов не понадобятся высоты, то достаточно будет хранить лишь разность высот левого и правого поддеревьев для каждого узла.

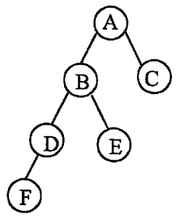


Рис. 1

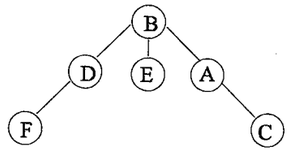


Рис. 2

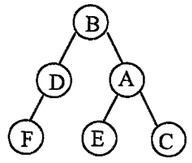


Рис. 3

Что же такое поворот? Для определенности положим, что после какой-либо операции для узла A высота левого поддерева превзошла высоту правого более чем на единицу. Так как операция выполнялась над уже сбалансированным деревом, то справедливо будет утверждение, что разность высот правого и левого поддеревьев равна -2 (рис. 1). Для восстановления сбалансированности дерева относительно данного узла необходимо добавить в правое поддерево хотя бы один уровень так, чтобы выполнялось условие о ключах узлов**. Для этого можно «позаимствовать» один узел из левого поддерева (в нем достаточно узлов для данной операции). Визуально этот процесс можно представить, что дерево «тянут» за правый дочерний узел, при этом узел B встает на место узла A, а узел A в свою очередь стает правым дочерним узлом B. После этой процедуры, направив ветку B->A вниз (так как теперь узел A дочерний от B), получится, что узел B теперь имеет три дочерних узла (рис. 2), а узел A в свою очередь не имеет дочерних узлов слева – дерево попрежнему несбалансированно. Завершением балансировки является «перекидывание» ветки B->E в A->E (рис. 3). Теперь дерево сбалансировано. Для поворота такого типа также необязательно знать высоту дерева, достаточно лишь разности высот, и новые разности получатся путем рассмотрения частных случаев, о чем будет сказано ниже.

* Далее так будем называть количество уровней.

** Для любого узла ключ слева не должен превосходить, а ключ справа должен превосходить ключ узла.

Теперь рассмотрим только что описанное действие над деревьями более глубоко. При таком повороте* заимствовался узел, который принадлежал множеству левых узлов узла A за исключением **правой ветки** левого дочернего узла A. Рассмотрим, что может быть при различных значениях баланса левого узла** (назовем его bal):

1) bal = -1. Как видно из вышеприведенного примера, баланс узла A станет равным нулю, так как правое поддерево останется неизменным, а в правом поддереве узла B количество узлов будет точно таким же. А баланс узла B, который теперь стоит на месте A, очевидно, обнулится, так как из левого поддерева был заимствован один узел, а в правое поддерево он был добавлен***;

2) bal = 0. Здесь баланс узла A будет на единицу больше, чем в предыдущем случае, а заимствование узла из левой ветки A не влечет за собой изменение высоты левого поддерева, поэтому баланс узла с ключом B станет равным 1;

3) bal = 1. Этот случай самый интересный, так как он не дает в результате сбалансированного дерева, что легко показывается рассуждениями, аналогичными предыдущим двум случаям. Заимствование узла влечет за собой уменьшение высоты левой ветки, а перенос правой ветки узла B еще увеличивает правое поддерево, поэтому баланс узла A примет значение -1, а баланс узла B будет равным 2, то есть такой поворот в данном случае ничего не дает, и данная логика работать не будет (рис. 4, 5).

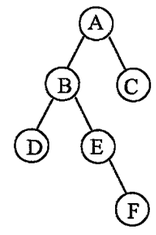


Рис. 4

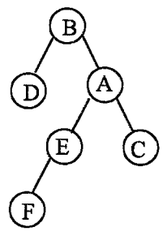


Рис. 5

В продолжении рассмотрения третьего случая легко прийти к выводу, что путем другого поворота, аналогичного рассмотренным выше****, с той лишь разницей, что заимствование узла производится из правого поддерева узла, относительно которого производится поворот, третий случай сводится к первым двум (рис. 4, 6, 7). Но для окончания балансировки необходимо определить баланс всех узлов, которые подверглись изменениям. Проблема появляется сразу же после выполнения правого поворота. Только что разобранный метод в данном случае неприменим, так как поворот выполняется для уже сбалансированного дерева, то есть вышеописанная балансировка неприменима. Самым простым решением явился бы отказ от использования баланса и использование вместо него количества дочерних уровней. Это не изменило бы идеи АВЛ–деревьев, а только лишь изменило бы метод их реализации, причем без принципиальных изменений. Недостатком данного метода является большая потребность в памяти и ограничение на количество узлов (в случае с балансом количество узлов ограничивалось лишь объемом памяти машины)*. Но метод имеет и преимущества. Во-первых, все операции балансировки реализовывались бы в общем виде (без рассмотрения частных случаев), что, естественно, более предпочтительно. Кроме того, такой метод более удобен для понимания. Рассмотрение же частных случаев приводит к появлению множества операторов условия, которые более медленны, чем вычислительные операции.

* В работе С. Д. Кондратьевой [1] данное действие называлось *малым левым поворотом.*

** *Балансом* узла условимся называть разность между высотами правой и левой веток узла.

*** В данном случае удаление узла влечет за собой изменение высоты ветки.*

**** Этот поворот называют *малым правым поворотом.*

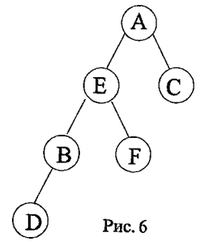


Рис. 6

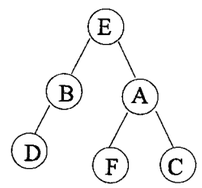


Рис. 7

Попытки реализовать общее решение малого поворота для сбалансированного и несбалансированного дерева, если и увенчаются успехом, то будут очень медленны, так как операция сложения, которая будет использоваться для вычисления количества уровней, гораздо более быстрая по отношению к операции условия. Кроме того, очевидно, что решение, использующее только баланс, содержит достаточно много условных операторов.

Итак, какие действия необходимо производить при выполнении операций вставки/удаления узлов в АВЛ-деревьях? Прежде всего, необходимо определить структуру их хранения, а иными словами, необходимо представлять структуру самого узла и структуру связей узлов. Свойствами каждого узла являются:

- ключ узла, чаще всего это некоторое число;
- предок узла;
- два его потомка;
- выполняемость критерия АВЛ-дерева, то есть высоты левого и правого поддеревьев должны отличаться друг от друга не более чем на единицу, поэтому необходимо хранить высоты левого и правого поддеревьев.

Но такая структура с точки зрения ее программной реализации очень «неповоротлива», поэтому нуждается в рационализации. Во-первых, просмотр дерева чаще всего осуществляется рекурсивно сверху вниз, поэтому предка узла хранить не обязательно (за исключением конкретных случаев, где это необходимо), во-вторых, было бы рационально в полях потомков узлов хранить указатели на структуры, характеризующие, это поможет избавиться от статичности структуры. В-третьих, для еще большего уменьшения структуры высоты левого и правого потомков рациональнее хранить в структурах, характеризующих самих потомков, но, так как любой узел, кроме корня, сам является потомком, необходимо хранить высоту поддерева с корнем в данном узле, которая на единицу больше максимума высот левого и правого поддерева. Но при такой структуре появляется необходимость проверки на наличие потомков при вычислении высоты, что влечет за собой условные операторы, которые, как известно, существенно замедляют работу, поэтому здесь приходится выбирать между ресурсами памяти и ресурсами времени. После рассуждений, изложенных выше, напрашивается следующая структура узла дерева, которая решает как проблему характеристики узла, так и проблему описания связей узлов в дереве:

```
type lpList = ^List ;
THeight = WORD ;
List = record
Height : THeight ; {высота поддерева}
Value : LongInt ; {ключ узла}
left  : lpList ;  {указатель на левого и правого
                                потомков}
right  : lpList ;
end ;
```

Теперь, когда метод машинного представления дерева вполне определен, можно приступить к проработке операций с этим деревом. Сначала рассмотрим действия, необходимые для вставки нового узла в дерево. Очевидно, что логика вставки рекурсивна, поэтому рассмотрим ее на некотором шаге. Пусть ключ узла будет x, а рассматриваемый узел дерева q. Тогда необходимо сравнить ключ узла q с новым ключом x, и если x меньше или равен, то перейти к рассмотрению левого потомка узла q, иначе перейти к правому потомку. Такая логика обеспечит поиск места для нового узла, которое будет найдено, когда логика перейдет к рассмотрению несуществующего потомка, а так как такое возможно, то перед сравнением необходимо проверить существование данного узла, и если узел не существует, то создать на этом месте узел с новым ключом. Здесь не следует забывать, что

* Здесь если размер стека не превосходит 64 Кбайт, то данный недостаток перестает быть актуальным

в результате обработки потомка узла, его высота могла измениться, так как в силу конечности количества узлов новый узел был вставлен именно в обрабатываемое поддерево, поэтому потребуется еще и флаг изменения высоты, который устанавливается при создании узла, а сбрасывается при неизменности высоты какого-либо узла до и после балансировки (относительно всех узлов – предков такого узла – дерево сбалансировано). После обработки потомка узла необходимо проверить этот флаг, и, если он установлен, то провести балансировку, если она требуется, и, если после этого высота поддерева не изменилась, сбросить флаг.

Суть балансировки – в получении разницы высот левого и правого поддеревьев и проверка ее на принадлежность диапазону [-1;1]*. Если это значение выходит за пределы этого диапазона, то необходима балансировка, при этом если оно больше единицы, то необходимо выполнить правый поворот, иначе – левый. Также необходимо учитывать, что при выполнении левого поворота относительно узла q необходимо, чтобы высота левого поддерева левого потомка (который, очевидно, существует) была не меньше высоты правого поддерева этого потомка. Аналогично и для правого поворота. Приведем пример реализации логики балансировки (Приложение А).

После реализации логики балансировки нетрудно реализовать простейшие операции с деревьями. Рассмотрим реализацию процедуры балансировки. Во-первых, данная логика должна найти место для вставляемого узла и собственно вставить его. Реализация такой логики не требует объяснений (Приложение Б).

Очевидно, что данная логика для АВЛ-деревьев неприменима. Исходя из того, что рано или поздно дерево перестанет быть сбалансированным, так как после добавления узла в дерево относительно какого-либо узла его поддерево (правое или левое) может изменить свою высоту (в рассматриваемом случае высота может только увеличиться). Вследствие этого может нарушиться АВЛ-сбалансированность дерева. Для того чтобы избежать этого, необходимо для каждого узла, рассматриваемого данной логикой, проверять принадлежность баланса рассматриваемого узла промежутку [-1;1] и проводить балансировку, если она требуется. Добавим такую проверку в существующую логику. Для этого после каждого рекурсивного вызова будем вызывать процедуру балансировки. Но написанная процедура балансировки выполняет слишком много действий, порой ненужных, так как если после балансировки какого-либо узла, выяснится, что высота поддерева, имеющего корнем рассматриваемый узел, не изменилась, то для всех узлов, для которых такой узел является дочерним, балансировка более не требуется, поэтому воспользуемся глобальным флагом, с которым работает балансировка. После этого логика вставки примет вид (Приложение В).

В своей работе [2] Никлаус Вирт рассматривает частные случаи балансировки АВЛ-деревьев и работает не с высотой поддерева, а уже с его балансом (Приложение Г).

Как видно из вышеприведенного программного кода, автор не использует высоту поддерева, а хранит лишь баланс каждого узла, что дает некоторый выигрыш в памяти и ограничивает количество узлов дерева лишь количеством доступной памяти машины. Но такой подход делает невозможной реализацию общего случая поворота и заставляет «разбивать» поворот на два случая, один из которых по сути своей является двукратным другим, но в силу отсутствия высот узлов несводим с нему.

Нетрудно показать, что общий случай поворота (хотя бы малого), используя только лишь баланс, реализовать невозможно. К такому выводу можно прийти, обозначив количество уровней в каком-либо поддереве узла, относительно которого производится поворот, за некоторую переменную. Через эту переменную можно выразить количество левых или правых уровней для любого из узлов рассматриваемого поддерева, для которого узел, относительно которого выполняется поворот, является предком. Очевидно, что нетрудно выразить через эту же переменную и количество уровней в дереве после выполнения поворота. Если бы задача о реализации общего случая поворота с использованием только баланса имела решение, то, выразив баланс через данную переменную для любого узла, подвергшегося изменениям в результате поворота, получилось бы выражение, которое не зависело бы от введенной переменной. Проделав вышеописанные действия, введя переменную x и выразив баланс узла, который до поворота являлся вершиной поворачиваемого поддерева, получим выражение вида: $bal = \pm\left[\frac{x+\alpha}{2}+\beta\right]\mp x$,

где α и β – числа, которые являются суммой балансов. Из вида выражения понятно, что оно зависит от x, что подтверждает зависимость вновь вычисленного баланса от высоты уровня.

Но как же производится поворот с использованием только лишь баланса? Так как в общем виде поворот не может быть выполнен, нужна дополнительная информация о рассматриваемом случае. Эту информацию и несет в себе флаг изменения высоты и структура построения логики Никлауса Вирта. А именно, после добавления нового узла в поддерево, с корнем в рассматриваемом узле, известно, высота какого поддерева (левого или правого) изменилась и, кроме того, известно, увеличилась она или уменьшилась и на сколько*. Таким образом, повороты разбиваются на два случая («правый» и «левый» повороты), не имеющие между собой ничего общего, что может повлечь увеличение размера кода программы, а вместе с тем

* Здесь положительную разницу условимся понимать как большую высоту правого поддерева.

и сложность. В предыдущей же логике повороты различаются лишь работой с указателями.

Рассмотрим несколько примеров работы той и другой логики. Приведем простейшие случаи поворотов дерева. Во-первых, рассмотрим случай, когда после вставки или удаления получилось так, что относительно некоторого узла высота левого его поддерева превысила допустимое значение на единицу, то есть правая ветка короче левой на два уровня. Для этого рассмотрим дерево, показанное на рис. 8, которое, очевидно, удовлетворяет всем критериям АВЛ-дерева. Теперь вставим в него новый узел с ключом, равным 1. Очевидно, новый узел будет левым потомком узла с ключом 2 (рис. 9). После вставки нового узла флаг изменения высоты будет выставлен и произойдет выход из рекурсивной процедуры, после чего в работу вступит следующий фрагмент логики (Приложение Д).

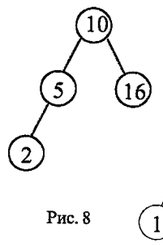


Рис. 8

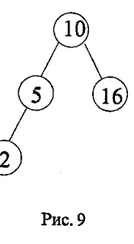


Рис. 9

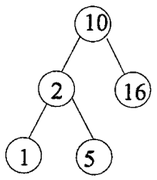


Рис. 10

Теперь p указывает на узел с ключом 2, **прежний** баланс этого ключа равен 0, поэтому новый станет равным -1 и снова произойдет выход из рекурсивной процедуры Search в тот же фрагмент кода. Но p указывает уже на узел с ключом 5, баланс которого равен -1, но по логике увеличилась высота левого поддерева данного узла на единицу, то есть новый баланс этого узла должен уменьшиться на единицу**. Новое значение лежит уже за допустимым пределом, поэтому требуется поворот, который должен либо увеличить высоту правого поддерева узла с ключом 5, либо уменьшить высоту его же левого поддерева. Такое действие было названо левым поворотом. Оно осуществляется с помощью замены рассматриваемого узла (назовем его корнем рассматриваемого поддерева) на его левого потомка, а рассматриваемый узел становится правым потомком корня. Но нынешний корень мог иметь непустое правое поддерево, которое теперь становится левым поддеревом «бывшего» корня. Таким образом, в результате левого поворота, в случае, если баланс левого потомка корня равен -1, высота правого поддерева увеличивается на единицу, а высота левого поддерева уменьшается на единицу и баланс узла становится равным нулю, так как баланс до поворота равнялся -2 (рис. 10). Такой поворот был назван *малым левым поворотом*. В случае же, если баланс левого потомка не равен -1, то малый поворот ничего не даст (это было показано выше). В этом случае выполняется *большой левый поворот*. Сущность его заключается таком повороте поддерева, с корнем в левом потомке узла, нуждающегося в повороте, после которого баланс корня этого поддерева станет равным -1 или даже -2. Такой поворот аналогичен малому левому и назван *малым правым поворотом*. Как и малый левый поворот, малый правый изменяет баланс узла на 2, но если левый поворот увеличивал, то правый уменьшает на 2. Малый правый поворот выполняется точно так же, как и левый, с той лишь разницей, что он предполагает превышение высоты правого поддерева над левой на 2. Его задачей является увеличение левого поддерева и уменьшение правого. После выполнения такого поворота над левым потомком узла, нуждающегося в повороте, его баланс станет равным либо -2, либо -3 (рис. 11*, 12), в зависимости от баланса левого его потомка. А левый малый поворот в свою очередь увеличит этот баланс на 2 (рис. 13). Таким образом, рассматриваемое поддерево будет сбалансированным.

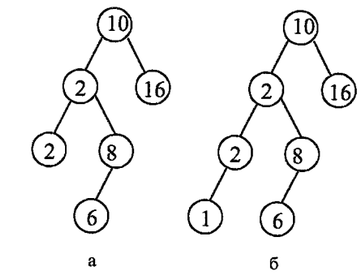


Рис. 11

Случаи, когда высота правого поддерева узла превосходит допустимое значение, аналогичны рассмотренным.

Также следует заметить, что балансировка Н. Вирта работает только тогда, когда баланс любого узла дерева выходит за пределы допустимого диапазона не далее чем на единицу, то есть такая логика применима лишь для последовательной вставки/удаления узлов. Но сбалансированные деревья изначально были созданы для организации баз данных, где может встать задача об удалении узла и всех его потомков, или ее можно сформулировать как удаление узлов, ключи которых лежат в заданном диапазоне. Последовательное удаление узлов из дерева не только требовательно к времени (так как возможны невостребованные

* В данном случае высота всегда изменяется на единицу.

** Это как раз те данные, которые позволяют рассматривать частные случаи поворотов.

* Случай, показанный на рис. 11б, может возникнуть только при удалении узла из правого поддерева.

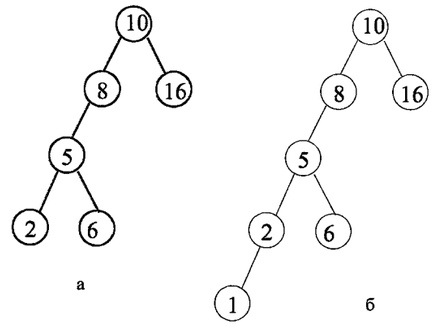


Рис. 12

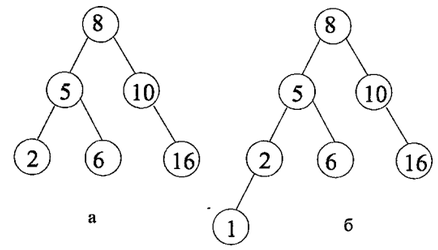


Рис. 13

повороты), но и, скорее всего, потребует создания списка удаляемых узлов, так как после удаления одного узла потомок узла может стать его предком, что делает невозможным дальнейшее удаление без предварительно составленного списка. Таким образом, данная логика еще и требовательна к памяти. Предлагаемая же логика легко оптимизируется под задачи такого типа путем простого зацикливания процедуры балансировки, то есть необходимо выполнять балансировку до восстановления допустимого баланса узла.

Приложение А

```
Function max(a, b : THeight) : THeight ;
                { функция вычисления максимума }
begin if( a>b) then max:=a else max := b end ;
Procedure GetNewHeight( q : lpList) ;
              { функция вычисления изменившейся}
var a, b : THeight ;              { высоты поддерева }
begin
   if( q^.left <> nil) then a := q^.left^.height
                                        else a := 0 ;
   if( q^.right <> nil) then b := q^.right^.height
                                        else b := 0 ;
 q^.height := max(a, b) + 1 ;
end ;

Function GetBallance( q : lpList) : ShortInt ;
                        { вычисление разницы высот }
var a, b : THeight ;
                  { левого и правого потомков узла}
begin
   if( q^.left <> nil) then a := q^.left^.height
                                        else a := 0 ;
   if( q^.right <> nil) then b := q^.right^.height
                                        else b := 0 ;
 GetBallance := b - a ;
end ;
Procedure Ballance( var q : lpList) ;
                        { собственно балансировка}
var bal : ShortInt ;
old_height : LongInt ;
begin
old_height := q^.height ; { сохранение высоты
                        для последующего сравнения}
GetNewHeight( q) ;    { вычисление новой высоты,
                                учитывая изменения}
   bal := GetBallance( q) ;  { вычисление разницы
                                              высот}
   if( bal > 1) then begin    { правое поддерево
                        выше допустимого уровня}
   if( GetBallance(q^.right) < 0)
                                then TurnLL( q^.right) ;
TurnRR( q) ;
      { выполняется правый поворот с
       предшествующей проверкой на наличие
       узла для заимствования}
          if( q^.height = old_height) then h :=
                                false ; { сбрасывание флага}
end else if( bal < -1)then  begin
                 { левое поддерево выше допустимого}
if( GetBallance(q^.left) > -0) then TurnRR( q^.left) ;
TurnLL( q) ;
{ выполняется правый поворот с предшествующей проверкой на наличие узла для заимствования}
          if( q^.height = old_height) then h :=
                                false ; {сброс флага}
end ;
end ;
```

Приложение Б

```
Procedure Insert( x : LongInt ; var q : lpList) ;
Var t1, t2 : lpList ;
Bal        : ShortInt ;
Begin
   If( q = nil) then begin
                 {место для вставки узла найдено}
          new(q) ;
with  q^ do begin
   value := x ;
   left := nil ;
   right := nil ;
   height := 1 ;
end ;
   End else
          If x < q^.value then          { новый узел
                должен принадлежать }
          Insert( x, q^.left)    { левому
                поддереву данного узла }
      Else Insert( x, q^.right) ;  { или правому }     ;
End ;
```


Приложение В

```
Procedure Insert( x : Byte ; var q : lpList) ;
var t1, t2 : lpList ;
bal   : ShortInt ;
begin
if( q = nil) then begin
   new(q) ;
h :=  true ;
with Q^ do begin
value :=  x ;
left := nil ;
right := nil ;
height := 1 ;
end ;
end else
if x < q^.value then begin
Insert( x, q^.left) ;
if( h) then Ballance( q) ;
end else begin
Insert( x, q^.right{, h}) ;
if( h) then Ballance( q) ;
end ;
end ;
```

Приложение Г

```
Procedure Search( x : Integer ; var p : ref ;
                                var h : boolean) ;
   Var p1, p2 : ref ;
Begin
   If( p = nil) then begin
         New(p) ; h := true ;
With p^ do begin
   Key := x ;
   Count := 1 ;
   Left := nil ;
   Right := nil;
   Bal := 0 ;
End
   End else
      If( x < p^.key) then begin
         Search( x, p^.left, h) ;
       If( h) then
          Case p^.bal of
          -1: begin p^.bal := 0 ;  h := false end ;
          0 : p^.bal := 1 ;
          1 : begin       p1 := p^.right ;
                if p1^.bal = -1 then
                begin
                     p^.left:=p1^.right; p1^.right:=p;
                     p^.bal := 0 ;
                     p := p1 ;
                end else begin p2 := p1^.left;
                     p1^.left:=p2^.right; p2^.right:=
                                                p1 ;
                     p^.right:=p2^.left; p2^.left := p ;
                     if p2^.bal=1 then p^.bal:=-1
                                else p^.bal:=0;
                     if p2^.bal=-1 then p1^.bal:=1
                                else p1^.bal:=0;
                     p := p2 ;
                end ;
                p^.bal := 0 ;  h := false ;
          end ;
       end ;
   end else begin p^.count := p^.count + 1 ;
                                h := false end ;
End ;
```

Приложение Д

```
   If( x < p^.key) then begin
      Search( x, p^.left, h) ;
      If( h) then
          Case p^.bal of
          1 : begin p^.bal := 0 ;  h := false end;
      0 : p^.bal := -1 ;
      -1: begin     p1 := p^.left ;
          if p1^.bal = -1 then
      begin
         p^.left:=p1^.right; p1^.right:=p;
         p^.bal := 0 ;
         p := p1 ;
      end else begin p2 := p1^.right;
        p1^.right := p2^.left ;  p2^.left:=p1 ;
         p^.left:=p2^.right; p2^.right := p ;
```

```
            if p2^.bal=-1 then p^.bal:=1
                                  else p^.bal:=0;
          if p2^.bal=1 then p1^.bal:=-1
                                  else p1^.bal:=0;
            p := p2 ;
        end ;
        p^.bal := 0 ;  h := false ;
    end ;
end ;
```

Примечания

1. *Кондратьева С. Д.* Введение в структуры данных: лекции и упражнения по курсу. М.: Изд-во МГТУ им. Н. Э. Баумана, 2000.

2. *Никлаус В.* Алгоритмы + структуры данных = программы. М.: «Мир», 1985.

Д. А. Подгорный

ТЕОРИЯ ОПТИМИЗАЦИИ: ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

В статье сделана попытка дать аналитический обзор одному из «новомодных» направлений теории оптимизации – генетическим алгоритмам.

В данной статье приводится суть и анализ текущего состояния научной области генетических алгоритмов (далее – ГА), которые реализуют собой один из интереснейших и весьма актуальных подходов к решению задач оптимизации – эволюционный подход. Предполагается, что читатель знаком с проблемами и основными методами решения задач оптимизации.

ГА – это метод приближенного решения задач оптимизации, а точнее, это алгоритм управляемого параллельного стохастического перебора. Это значит, что ни один ГА не гарантирует решения какой-либо задачи за полиномиальное время, или нахождение самого лучшего результата, тем не менее они весьма эффективны, когда необходимо получить приближенный результат за малое время. Следует отметить, что в теории ГА появились достаточно давно. Автором идеи считается Джон Холланд (Holland J. H.), который делал первые публикации по ГА еще в начале 60-х. ГА получили признание в 1975 г., после выхода в свет книги «Адаптация в естественных и искусственных системах» [1]. Но тогда ГА не представляли практического интереса из-за малых мощностей компьютеров того времени и развивались относительно медленно. Сейчас это мощный аппарат для эффективного решения целых классов задач, как тех, которые не могли быть решены ранее, так и тех, которые решались, но не достаточно эффективно. В частности, «в компетенцию» ГА входят все задачи оптимизации, от комбинаторной до многопараметрической. Интересно, что, являясь универсальным методом, ГА часто дают результаты не хуже, а иногда даже лучше, чем специальные методы. Как правило, это связано с тем, что при росте размерности задачи, рост объема вычислений у ГА значительно ниже, чем у других методов.

Скажем несколько слов об основе ГА – эволюционном подходе к решению задачи. Такой подход предполагает, что в процессе решения решающий будет опираться на некоторые принципы естественного (или эволюционного) развития. Как правило, это принцип наследственности и принцип естественного отбора. В настоящее время применение такого подхода отнюдь не ограничивается задачами оптимизации, но мы остановимся только на ГА как на четкой конкретизации этого подхода для решения оптимизационных задач.

В общих чертах ГА представляет собой модель естественного развития некоторой популяции особей (индивидов), где заданы все основные природные процессы: фактор приспособленности особи к окружающей среде, воспроизводство и естественный отбор особей в соответствии с их приспособленностью. При работе алгоритма мы должны пронаблюдать эволюцию нашей популяции по фактору приспособленности и в результате получить самого приспособленного индивида.

Для конкретной задачи оптимизации каждый индивид является решением задачи, записанным в форме какого-либо единого представления параметров, например строки. Отметим, что под решением мы понимаем некие значения параметров, соответствующие условиям задачи и дающие какой-либо, не обязательно желаемый, результат целевой функции. Это представление параметров играет роль цепочки генов для данного индивида. Именно с ней и происходят все действия. Отсюда и название – генетический алгоритм. Фактор приспособленности индивида есть мера «пригодности» данного решения (т. е. насколько оно удовлетворяет запросам составителя алгоритма), и роль его является ведущей. Как правило, это значение целевой функции. Процесс воспроизводства есть некоторый оператор получения новых решений из уже имеющихся, для организации перебора. Следует сказать, что, в общем понимании, ГА – это именно схема для построения алгоритма, а не конкретный алгоритм.

Генетический алгоритм в изложении Холланда, который принято считать классическим, выглядит следующим образом:

```
НАЧАЛО /* генетический алгоритм */
Создать начальную популяцию
Оценить приспособленность каждого индивида
останов := FALSE
```

ПОДГОРНЫЙ Дмитрий Александрович – студент V курса факультета информатики ВятГГУ


```
ПОКА НЕ останов ВЫПОЛНЯТЬ
НАЧАЛО /*создать популяцию нового поколения*/
ПОВТОРИТЬ (размер_популяции/2) РАЗ
НАЧАЛО /* цикл воспроизводства */
Выбрать двух индивидов с высокой приспособленностью из предыдущего поколения для скрещивания
Скрестить выбранных индивидов и получить двух потомков
Оценить приспособленности потомков
Поместить потомков в новое поколение
КОНЕЦ
ЕСЛИ все индивиды одинаковы или пройдено заданное количество поколений ТО останов := TRUE
КОНЕЦ
КОНЕЦ
```

Этот ГА был представлен Д. Холландом в одной из первых работ и создавался исключительно с теоретической целью. Применяемые на практике алгоритмы, как правило, сильно отличаются от приведенного выше.

Принцип действия алгоритма прост: выживает сильнейший, но реальный механизм его работы весьма сложен. Вполне понятно, что за счет замены «плохих» решений «хорошими», и составления новых решений, мы будем получать все более и более «хорошее» множество решений. Однако совсем не очевидно, как это происходит и можем ли мы гарантировать, что в конце концов доберемся *достаточно близко* к самому лучшему решению? Следует отметить, что ГА представляет собой сложную нелинейную динамическую структуру и его поведение поддается математическому описанию, так же как и предсказанию, очень относительно. Но тем не менее доказать, что он *будет* работать, можно. Такое доказательство опубликовано Д. Холландом в 1975 г. и носит название «теорема схем» [2]. Также в 1989 г. Дэвид Голдберг (Goldberg D. E.), один из основоположников ГА, выдвинул «гипотезу строящих блоков» [3]. В этих же источниках впервые наглядно показывается схема параллельных вычислений ГА. Мы не будем приводить здесь полное математическое изложение этих доказательств, ограничившись только сутью.

Объектом всех действий ГА является решение, закодированное в виде последовательности параметров (генов). На ее основе и осуществляется воспроизведение новых решений из уже имеющихся, а также оценка и отбор решений. Принцип наследственности предполагает, что решение-потомок должно сочетать в себе свойства обоих решений-родителей. Фактически ГА выделяет неудачные комбинации генов и пресекает их развитие. Т. е. происходит структуризация пространства поиска, и отсекаются области, содержащие плохие решения. За счет этого объем поиска (здесь и далее под объемом поиска имеется в виду «просмотренная» алгоритмом область пространства поиска) очень сильно сокращается, а ГА продолжает поиск в других областях.

Данная схема дает нам понимание того, как ГА работает. Но работать не всегда значит хорошо работать. От чего зависит эффективность ГА? Мы можем сказать, что для получения хорошего результата, пользователь должен:

- подобрать некоторое представление для рассматриваемого решения в виде строки, дерева и т. д.;
- выбрать или разработать операторы, которые формируют новые решения на основе имеющихся, так, чтобы наилучшим образом учесть особенности пространства поиска (например, симметрию), если о нем есть какая-либо информация, и как можно более полно соблюсти принцип наследственности;
- выбрать подходящий объем и структуру популяции;
- отобрать индивидов для выживания и/или скрещивания на основе их «пригодности» (например, по значению целевой функции);
- применить различные операторы и установить соответствующие значения параметров, чтобы обеспечить сходимость процедуры поиска в важных областях пространства поиска, избегая при этом сходимости к субоптимальным решениям (локальным оптимумам).

Но так ли это? Безусловно, все вышеперечисленные шаги необходимы для получения хорошего результата, но, увы, вовсе не достаточны. На сегодняшний день очень трудно дать определенный ответ на целый ряд практических и теоретических вопросов, касающихся функционирования ГА. Фактически сейчас нет универсальных методов, способных дать четкий ответ, будет ли данный ГА эффективен и насколько. Тем не менее существуют некоторые методы оценки и даже предсказания ГА, но их результативность во многом зависит от сложности и структуры данного ГА. Это методы описания ГА с помощью марковских цепей [4] и статистической механики [5].

При отсутствии оценочного аппарата разработка новых решений в создании ГА становится чрезвычайно трудоемким делом. Все результаты, как правило, получены с помощью эксперимента. Сюда относится и основная задача управляемости ГА – задача о том, каким образом достичь баланса «исследование – освоение», т. е. объема поиска ГА и его скорости схождения. Добиться того, чтобы алгоритм не останавливался в субоптимальных решениях и не делал чрезмерно большого поиска при произвольных условиях, а возможно, и постановке задачи. В данный момент разработано множество стратегий, на основе которых создаются операторы, оказывающие давление на ГА с той или иной стороны. Часто это модели природных процессов, таких, как мутация (случайные изменения способствуют увеличению объема поиска, а главное, позволяют алгоритму «выходить» из локальных оптимумов), или такие модели выбора родителей, как inbreeding (инбридинг) и outbreeding (аутбридинг) (для создания потомков выбираются наиболее похожие (inbreeding) и непохожие (outbreeding) родители, что тоже влияет на объем поиска). Но если воздействие одного оператора на поведение алгорит-

ма можно предположить, руководствуясь здравым смыслом, а иногда и предсказать теоретически, то комбинации двух и более операторов, в общем случае, почти непредсказуемы, и мало изучены. Также отдельной темой для изучения являются так называемые адаптивные операторы. Т. е. такие операторы, которые имеют двустороннюю связь с алгоритмом и могут изменять свое действие в зависимости от результатов.

Кроме оценки эффективности ГА существует проблема оценки задачи. Исследователями постоянно предпринимаются усилия по четкому выделению ГА-разрешимых и ГА-неразрешимых задач, а также выведению свойств, необходимых и достаточных для вхождения в первый или второй класс. Эта проблема решена лишь отчасти. На сегодняшний день есть только один класс задач, для которого достигнуто более глубокое понимание поведения ГА. Это задачи, где ландшафт целевой функции не зависит от времени, а приспособленность каждого индивида может быть вычислена независимо от других членов популяции.

Но такое понимание исчезает, когда мы обращаемся к задачам, ландшафт целевой функции которых не удовлетворяет традиционным допущениям. Существует как минимум три важных класса задач, которые остро нуждаются в разработке методики решения:

1) задачи с автономно меняющимися ландшафтами,
2) задачи развития кооперативного поведения,
3) экологические проблемы.

Задачи с автономно меняющимися ландшафтами

Задачи такого типа возникают, когда целевая функция определяется в терминах факторов окружающей среды, например времени. Как правило, это задачи с участием механических устройств, которые изнашиваются, и их приспособленность со временем падает. Например, построение схемы грузоперевозок при определенном количестве машин с учетом того, что на некоторых дорогах машины изнашиваются быстрее и их «эффективность» падает.

Задачи развития кооперативного поведения

Этот класс составляют задачи, где приспособленность одного индивида зависит от того, насколько хорошо ее дополнят другие. Цель не искать лучшего индивида, а выявить группу индивидов, коллективно решающих проблему. В качестве примера можно привести задачу о проектировании некой системы защиты. Индивидом будет являться один элемент защиты с некоторыми характеристиками, предотвращающий все или некоторые типы атаки. Тогда популяция будет являть собой целую систему защиты. В ней будет происходить развитие индивидов от большого количества универсалов до меньшего количества «специалистов», взаимодополняющих друг друга.

Экологические проблемы

Такие задачи изучены меньше всего. Это задачи, ландшафт которых подвержен влиянию эволюционных факторов, т. е. изменяется в зависимости от действий алгоритма. Например, задача об освоении с максимальной выгодой некоторой посевной площади. «Потенциал» почвы, а значит, и прибыль, может снижаться или расти в зависимости от того, что и как мы будем садить.

Стоит отметить, что много внимания сейчас уделяется выбору формы представления решений. При выборе определенного представления мы создаем пространство поиска для ГА путем отображения точек реальной задачи в точки пространства выбранного представления. Соответственно от этого выбора во многом может зависеть успех ГА. Используются различные представления, начиная с классической двоичной кодировки и заканчивая проблемно-ориентированными кодировками, для специфических задач. Конечно, универсальные кодировки отличаются более широким диапазоном применимости, но, как правило, уступают проблемно-ориентированным, которые хоть и более сложны в использовании, но несут дополнительную информацию о задаче. Пока еще никому не удалось разработать полную и исчерпывающую теорию представлений, хотя было предпринято несколько серьезных попыток и получены весьма ценные результаты [6].

Много вопросов еще остается по построению самой популяции и моделированию ее развития. Конечно, все представляют эволюционный процесс в целом, абстрагируясь от деталей. Но мнения по построению математической модели сильно расходятся. Существует достаточно много вариаций моделей эволюционного процесса. В основном они различаются решением нескольких принципиальных вопросов о составлении модели, таких, как:

1) выбор размера популяции,
2) стратегии удаления,
3) отбор родителей,
4) воспроизводство и наследование.

Выбор размера популяции

В подавляющем большинстве моделей размер популяции является входным параметром (N), определяемым пользователем, и является величиной постоянной. Здесь выделяют два типа моделей: модели поколенческие и стационарного состояния. Модели стационарного состояния жестко задают ограничение популяции в N индивидов, и, когда их количество становится N+1, включается механизм отбора, призванный уменьшить популяцию до размера N. Поколенческие же модели допускают достаточную гибкость в этом плане, разрешая появление всего нового поколения (т. е. до N потомков), перед тем как придет в действие механизм отбора. Несложно догадаться, что большой размер популяции способствует большей «поисковой мощности» алгоритма, но снижает реальную скорость его работы, и наоборот. В данный момент мы располагаем лишь начальными теоретическими результатами, которые тем не

менее позволяют давать некоторые априорные рекомендации по выбору подходящего фиксированного размера популяции и дают некоторое представление о необходимом уровне гибкости этой величины. Но при этом нет почти никаких теоретических результатов о том, как добиться высокой эффективности при построении моделей с динамической адаптацией размеров популяции.

Стратегии удаления

Стратегии, применяемые для удаления членов популяции, значительно варьируются от одной модели к другой и включают такие стратегии, как случайное равновероятное удаление («равномерная рулетка»), удаление наихудших индивидов (стратегия «элитизма») и удаление, обратно пропорциональное значению целевой функции («пропорциональная рулетка»). Ясно, что стратегии, сохраняющие элиту, способны привести к преждевременной утрате популяцией разнообразия, а значит, к субоптимальным решениям. Столь же ясно, что сохранение в популяции слишком большого числа слабых индивидов приведет к бесцельному блуждающему поиску. Найти нужное равновесие важно, но это трудно сделать априори при нынешнем состоянии теории.

Отбор родителей

Аналогичные трудности возникают и при отборе родителей, которые будут давать потомство. Слишком сильное предпочтение наилучшим индивидам приведет к чрезмерному сужению целей поиска, тогда как недостаточное внимание к ним сделает эти цели размытыми. Современные методы включают случайный равновероятный (равномерный) отбор, ранговопропорциональный отбор и отбор пропорционально значению целевой функции. Эти стратегии достаточно хорошо изучены порознь. Однако ясно, что стратегии отбора родителей и удаления индивидов должны дополнять друг друга в смысле общего влияния, которое они оказывают на баланс «исследование – освоение». Существуют некоторые теоретические результаты для частных случаев, например, «оптимальное расположение проб» Д. Холланда [7] и «правило 1/5» [8], но требуются гораздо более сильные результаты.

Воспроизводство и наследование

Наряду с упомянутыми выше процессами отбора, на баланс «исследование – освоение» влияет также и механизм воспроизводства. Чтобы охарактеризовать это влияние, рассмотрим две крайние точки зрения на механизм воспроизводства. С одной стороны, можно представить систему, в которой потомки являются точными копиями родителей (неполовое воспроизводство без мутаций), что влечет быстрый рост доли наилучших индивидов в популяции, но исключает исследование пространства вне множества членов начальной популяции. Противоположный же случай представляет собой система, в которой потомки имеют мало общего с родителями, что максимизирует освоение в ущерб наследованию полезных свойств родителей. В результате усилий исследователей было установлено, что улучшения результата ГА можно достигнуть при использовании репродуктивного механизма, состоящего из родительского воспроизводства и мутации. Причем в некоторых случаях результаты улучшались при отказе от мутации, либо при переходе к более мощным операторам мутации [9]. Здесь, как и в предыдущем случае, есть многообещающие начальные теоретические результаты, способствующие пониманию использования и направлений дальнейшего развития репродуктивных механизмов [10]. Однако быстрое расширение области исследований подвергает эти теории серьезным испытаниям, предъявляя им «аномальные результаты» [11] или новые направления, не охваченные современной теорией. Одним из важных, но не вполне понятных вопросов является вопрос о преимуществах адаптивных операторов воспроизводства. Имеются различные экспериментальные результаты, свидетельствующие об эффективности различных адаптивных операторов [12].

Отметим некоторые перспективные идеи и направления исследования, которые, скорее всего, сыграют важную роль в ближайшем будущем.

В разделе, посвященном вопросам представления решений, обсуждались плюсы и минусы проблемно-независимых и проблемно-ориентированных представлений. С этим вопросом тесно связано биологическое различие между более универсальным генотипическим описанием индивидов в виде схем их генерирования (т. е. некоторая форма кодирования внешних признаков, например ДНК) и фенотипическими описаниями реально сгенерированных структур (внешние признаки как таковые, например наличие ног, рук). В природе, как известно, все эволюционные операторы работают на уровне генотипа и реализуются через рост и созревание индивида. Если мы хотим учитывать такие сложные процессы, нам, возможно, понадобится использовать более универсальные процедуры кодирования в сочетании с формообразовательными процессами (морфогенезом) [13].

Еще одним интересным новшеством является включение ламарковских свойств. Хотя ГА берут начало из биологических систем, они приобретают множество интересных свойств при включении особенностей, отсутствующих в исходных системах. Например, ламарковских операторов, разрешающих наследование признаков, приобретенных индивидом за его жизненный цикл. Модели с этими операторами построены так, что индивиды проходят через фазу обучения и/или адаптации, рассматриваемую как часть процедуры оценивания их качества, а результаты такой адаптации передаются их потомкам [14]. Хотя первые полученные экспериментальные результаты вдохновляют, пока что нет хорошего способа проанализировать такие системы на более абстрактном уровне.

Перспективной идеей является также и идея неслучайного спаривания и видообразования. В насто-

ящее время почти все эволюционные модели построены на схеме случайного спаривания, в которой вид или пол индивида не играет роли. Но с переходом к более сложным системам, которые пытаются выработать кооперативное поведение и в которых может протекать не один, а несколько эволюционных процессов одновременно, роль неслучайного спаривания и видообразования становится вопросом особой важности.

Предложено несколько подходов к решению подобных задач:

1) вытеснение (crowding) [15],
2) разделение (sharing) [16]
3) мечение (tagging) [17].

Вытеснение

Стратегия вытеснения определяет порядок замены индивидов в популяции и сводится к следующему: из популяции случайно выбирается некоторое количество индивидов и заменяется тот из них, кто наиболее похож на того, которого мы хотим включить в популяцию.

Разделение

Стратегия разделения сводится к обратному, но реализуется несколько по-другому: значение целевой функции зависит не только от представления индивида, но и от представлений других индивидов так, что похожие индивиды «получают» меньшую приспособленность. Таким образом, поддерживается разнообразие популяции.

Мечение

Стратегия мечения может использоваться для разных целей и заключается в следующем: представлениям индивидов приписываются определенные «метки», на основании которых действуют другие операторы, такие, как скрещивание или отбор.

К сожалению, для этих решений характерны довольно сильные предположения, такие, как число видов и/или распределение ниш в среде. Для некоторых задач такие предположения оправданны. Однако во многих случаях подобные свойства априорно неизвестны и должны быть еще разработаны [18].

Стоит еще обратиться к созданию так называемых децентрализованных и параллельных моделей. В силу врожденного естественного параллелизма эволюционных моделей большое число позднейших публикаций посвящено описанию специализированных параллельных вычислительных систем, реализующих как ГА, основанные на концепции «островов» (coarse-grain или island parallel GA), так и параллельные ГА со скрещиванием в локальной области (fine-grain parallel GA). Первая модель предполагает наличие нескольких различных популяций, которые развиваются независимо друг от друга, но периодически обмениваются определенным объемом «генетического материала». Идея второй модели заключается в том, что индивиды рассматриваются как точки в некотором пространстве и скрещиваться могут только те из них, кто находится рядом. Подобные реализации обещают существенное сокращение времени выполнения ГА. Более того, такие модели открывают огромные возможности по реализации особых природных эволюционных эффектов, таких, как видообразование, формирование ниш, миграции видов и т. п. Однако каждое такое изменение ГА существенно меняет его семантику и конечное поведение [19]. К сожалению, относительно слабо развитая даже для традиционных ГА теория мало что дает для понимания таких параллельных реализаций.

Другим не менее интересным направлением является создание коэволюционных систем. Есть некоторые работы об улучшениях, которые могут быть достигнуты при совместном развитии добавочных индивидов-паразитов вместе с исследуемыми индивидами, которые позволяют оценить всю поведенческую сложность и эффективность таких методов [20]. Предложенная Д. Холландом система Echo [21] отражает еще более сложную экологическую ситуацию, включающую обновляемые ресурсы и хищников. Также были показаны преимущества «кооперативных» коэволюционных моделей [22]. Каждая из этих систем предполагает важную будущую роль коэволюции в эволюционных алгоритмах, но подобные системы ставят больше вопросов, чем дают ответов относительно принципов их конструирования, а также типов задач, для которых этот дополнительный уровень сложности является как необходимым, так и эффективным.

Еще одна из наиболее интересных тем для исследования – это самоадаптирующиеся системы, о которых немного упоминалось выше. Другая смежная все чаще появляющаяся тема – это введение в эволюционные алгоритмы механизмов самоадаптации для управления параметрами самого алгоритма, включая внутреннее представление, мутацию, воспроизводство и размер популяции. Частично эта тенденция вызвана отсутствием сильной прогностической теории, которая определяла бы значения таких параметров априори. Данная тенденция отражает и тот факт, что эволюционные алгоритмы применяют к более сложным и изменяющимся во времени ландшафтам. Некоторые важные вопросы, требующие решения, касаются самого механизма самоадаптации. Например, нужно ли использовать эволюционный алгоритм или какой-то иной механизм? Если применять эволюционный алгоритм, то как использовать для самоадаптации значение целевой функции в качестве характеристики обратной связи? Положительный аспект состоит в том, что эффективность самоадаптации для мутаций и воспроизводства для ГА уже показана экспериментально [23].

Несмотря на то что теория ГА развита еще очень слабо, сегодня ГА успешно применяются для решения ряда больших и экономически значимых задач в бизнесе и инженерных разработках. С помощью ГА

были разработаны промышленные проектные решения, позволившие сэкономить миллионы долларов. Финансовые компании широко используют эволюционные методы прогнозирования развития финансовых рынков для управления портфелями ценных бумаг. Также ГА используются для оценки значений непрерывных параметров моделей большой размерности, для решения нп-полных комбинаторных задач, для оптимизации моделей, включающих одновременно непрерывные и дискретные параметры, в системах извлечения новых знаний из больших баз данных (дата мининг), для построения и обучения стохастических и нейронных сетей и оценки параметров в задачах многомерного статистического анализа. Учитывая усилия исследователей, прилагаемые в области ГА, с каждым годом их эффективность растет. Логично предположить, что уже очень скоро ГА обретут большую популярность и прочно войдут в инструментарий разработчика решений. Возможно, осмысление механизма, сделавшего из обезьяны человека станет одним из серьезнейших достижений человечества.

Примечания

1. *Holland, J. H.* Adaptation in Natural and Artificial Systems. Ann Arbor, Michigan: Univ. Michigan Press, 1975.
2. Там же.
3. *Goldberg, D. E.* Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.
4. *Vose, M. D., Liepins, G. E.* Complex Systems. 1991. V. 5. P. 31; *Nix, A., Vose, M. D.* Ann. Math. and Artificial Intelligence. 1991. V. 5. P. 88.
5. *Shapiro, J., Rattray, M., Pruegel-Bennett, A.* The Statistical Mechanics Approach to the Study of Genetic Algorithm Dynamics // First International Conference on Evolutionary Computation and Its Applications. 1996. P. 121-134. Publisher: EvCA'96.
6. *Holland, J. H.* Cit. op.; *Jones, T. C.* Evolutionary Algorithms, Fitness Landscapes, and Search: Ph. D. Thesis. Albuquerque, NM: University of New Mexico, 1995; *Manderick, B., de Weger, M., Spiessens, P.* The genetic algorithm and the structure of the fitness landscape // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991; *Radcliffe, N. J.* Forma analysis and random respectful recombination // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991.
7. *Holland, J. H.* Cit. op.
8. *Schwefel, H.-P.* Numerical Optimization of Computer Models. N. Y.: Wiley, 1981.
9. *De Garis, H.* Genetic programming: modular evolution for Darwin machines // Proceedings of the 1990 International Joint Conference on Neural Networks. Washington, DC: Lawrence Erlbaum, 1990. P. 194-197; *Koza, J. R.* Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press, 1992; *Eshelman L. J., Schaffer, J. D.* Preventing premature convergence in genetic algorithms by preventing incest. — In.: Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991. P. 115-122.
10. *Holland, J. H.* Cit. op.; *Vose, M. D., Liepins, G. E.* Schema disruption // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991. P. 237-242; *Booker, L.* B. Recombination distributions for genetic algorithms // Proceedings of the Workshop on Foundations of Genetic Algorithms. Vail, CO: Morgan Kaufmann, 1992; *De Jong, K.A., Spears, W.* A formal analysis of the role of multi-point crossover in genetic algorithms. Ann. Math. Artificial Intelligence. 1992. V. 5. № 1. P. 1-26; *Spears, W. M.* Crossover or mutation? // Proceedings of the Workshop on Foundations of Genetic Algorithms. Vail, CO: Morgan Kaufmann, 1992.
11. *Forrest, S., Mitchell, M.* Relative building-block fitness and the building block hypothesis // Proceedings of the Second Workshop on Foundations of Genetic Algorithms. Vail, CO: Morgan Kaufmann, 1992. P. 109-126.
12. *Fogarty, T. C.* Varying the probability of mutation in the genetic algorithms // Proceedings of the Third International Conference on Genetic Algorithms. Fairfax, VA: Morgan Kaufmann, 1989. P. 104-109; *Baeck, T., Hoffmeister, F., Schwefel, H.-P.* A survey of evolution strategies // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991. P. 2–9; *Schaffer, J. D., Morishima, A.* An adaptive crossover distribution mechanisms for genetic algorithms // Proceedings of the Second International Conference on Genetic Algorithms. Cambridge, MA: Lawrence Erlbaum, 1987. P. 36-40; *Davis, L.* Adapting operator probabilities in genetic algorithms // Proceedings of the Third International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1989. P. 60–69.
13. *Dawkins, R.* The Blind Watchmaker. N.Y.: W. W. Norton & Co., 1987; *Harp, S., Samad, T., Guha, A.* Towards the genetic synthesis of neural networks // Proceedings of the Third International Conference on Genetic Algorithms. Fairfax, VA: Morgan Kaufmann, 1989. P. 360-369.
14. *Grefenstette, J. J.* Lamarckian learning in multi-agent environments // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991. P. 303-310.
15. *De Jong, K. A.* An analysis of the behavior of a class of genetic adaptive systems: Doctoral Thesis. Department of Computer and Communication Sciences. Ann Arbor: Univ. of Michigan, 1975.
16. *Goldberg, D. E., Richardson, J.* Genetic algorithms with sharing for multimodal function optimization // Proceedings of the Second International Conference on Genetic Algorithms. Cambridge, MA: Lawrence Erlbaum, 1987. P. 41-49.
17. *Booker, L.* B. Intelligent Behavior as an adaptation to the task environment: Doctoral Thesis. Department of Computer and Communication Sciences. Ann Arbor: Univ. Of Michigan, 1982.
18. *Spears, W. M.* A simple subpopulation scheme // Proceedings of the Fourth Conference on Evolutionary Programming. San Diego, CA: MIT Press, 1994. P. 296-307.
19. *De Jong, K., Sarma, J.* On decentralizing selection algorithms // Proceedings of the Sixth International Conference on Genetic Algorithms. Pittsburgh, PA: Morgan Kaufmann, 1995. P. 17-23.
20. *Hillis, D. W.* Co-evolving parasites improve simulated evolution as an optimization procedure // Physica. 1990. V. D42. P. 228-234.
21. *Holland, J. H.* Adaptation in Natural and Artificial Systems. 2nd ed. Cambridge, MA: MIT Press, 1992.
22. *Potter, M., De Jong, K., Grefenstette, J.* A coevolutionary approach to learning sequential decision rules // Proceedings of the Sixth International Conference on Genetic Algorithms. Pittsburgh, PA: Morgan Kaufmann, 1995. P. 366-372.
23. *Goldberg, D. E., Deb, K., Korb, B.* Don't worry, be messy / / Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991;

14 Зак. 2868

Schraudolph, N. N., Belew, R. K. Dynamic parameter encoding for genetic algorithms // Machine Learning J. 1992. V. 9. № 1. P. 9-22. P. 24-30; *Shaefer, C. G.* The ARGOT strategy: adaptive representation genetic optimizer technique // Proceedings of the Second International Conference on Genetic Algorithms. Cambridge, MA: Lawrence Erlbaum, 1987. P. 50-58; *Whitley, D., Mathias, K., Fitzhorn, P.* Delta coding: an iterative search strategy for genetic algorithms // Proceedings of the Fourth International Conference on Genetic Algorithms. La Jolla, CA: Morgan Kaufmann, 1991. P. 77-84.

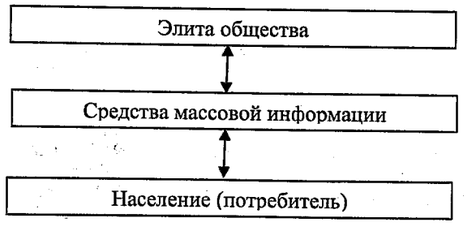
А. Н. Ронжин

НОВОЕ ПОЗИЦИОНИРОВАНИЕ КОМПЬЮТЕРНЫХ КОМПАНИЙ НА РЫНКЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

В статье анализируется рекламная стратегия компьютерных фирм в современных условиях.

Основы формирования стратегических рекламно-маркетинговых кампаний

Стремительный прогресс в области информационно-коммуникационных технологий поставил перед компьютерными компаниями, действующими на рынке, новые цели и задачи. Одной из основных задач на данный момент является более глубокая интеграция в общественную жизнь, видение путей развития общественных связей компании и пропагандирования ее ценностей [1]. Рекламная стратегия, приведенная в настоящей работе, применима к компаниям, работающим с корпоративными клиентами: заводами, предприятиями, банками, государственными учреждениями и крупными коммерческими фирмами.



Основной идеей при формировании стратегической рекламно-маркетинговой кампании должна быть идея «трехэтажного пиара». В упрощенном изложении она описывает процессы общественной коммуникации как взаимодействие трех составляющих – элиты общества, обладающей наибольшим влиянием, финансовым и политическим потенциалом; населения, воспринимающего нормы поведения, потребления и идеологии, диктуемые элитой; СМИ, являющихся посредниками при передаче сообщений от элиты – к населению и обратно (см. рисунок).

Дело в том, что массовые коммуникации не могут ориентироваться исключительно на один сегмент задачи – только на элиту, только на население или только на СМИ. Мнение элиты – это мнение референтной группы, публикации в СМИ – лишь факт присутствия данного продукта в «общественном сознании», а мнение потребителя подвержено влиянию как первых двух групп, так и факторов слухов и установок.

Поэтому для того, чтобы сформировать по-настоящему эффективную стратегическую кампанию, ориентированную на массового потребителя, необходим учет всех видов взаимодействия и оценка эффективности проводимых действий с точки зрения каждого из «этажей». В нашем случае это будет означать сочетание массовых мероприятий (семинаров, конференций, акций), «элитной продукции» (презентационные фильмы, буклеты, «реклама рекламы») и ориентированных на широкую публику материалов в СМИ.

Рекламно-маркетинговые кампании на кировском рынке информационных технологий

Рынок информационных технологий г. Кирова, в особенности в сегменте высокотехнологичных решений, поделен между несколькими компьютерными компаниями. Необходимость проведения стратегической рекламной кампании определяется не столько их конкуренцией на местном рынке, сколько необходимостью выхода компаний на рынки соседних регионов.

Рассматривая вводные к данной задаче, мы обнаруживаем, что большинство компаний, работающих на рынке корпоративных услуг, не имеют своего лица, т. е. не определены ценности компании, отсутствует идеология и миссия. В целом непонятно, чем одна компания отличается от другой. Можно сказать, что торговой маркой такой компании является ее директор.

Потенциальный заказчик компании ориентируется преимущественно на свою референтную группу – отзывы и советы своих друзей, имеющих опыт работы с данной компанией. Новые клиенты приобретаются по рекомендациям бывших клиентов. Проблемы возникают при выходе компании на ближайшие регионы даже в рамках Приволжского федерального округа. Компания не воспринимается как торговая марка. В этой ситуации возникает задача создания внутрикорпоративной стратегии принятия решений по формированию внешнего имиджа компьютерной компании и стратегических маркетинговых коммуникаций в целом [2].

В большинстве случаев компании не заботятся о собственном имидже. Фирменный стиль у компаний, за редким исключением, отсутствует: наиболее наглядно это отражается на документообороте компа-

РОНЖИН Андрей Николаевич – студент IV курса факультета информатики ВятГГУ

нии или в оформлении офиса. Практически невыполнимой задачей для руководства компаний является решение о смене названия компании или логотипа, хотя очевидно, что при выходе на рынок ПФО название компании, позиционированное на местный рынок (например, «Вятские Информационные Технологии»), является скорее недостатком, чем достоинством.

Акции, проводимые компаниями, ограничиваются статусными мероприятиями, демонстрирующими исключительно сам факт существования компании на рынке. Их позиционирование, даже в таких очевидных элементах, как уровень технических решений и перспективы развития, не находятся в фокусе руководства.

Презентационный пакет (в виде буклетов и/или презентационных фильмов) отсутствует; некоторые из компаний не имеют даже интернет-сайтов. Существующие рекламные материалы сформированы с позиции самой компании, а не с позиции восприятия ее клиентов. Иными словами, компания описывает свою деятельность с точки зрения того, что, как ей представляется, необходимо клиентам [3]. Кроме того, при их прочтении возникает необходимость в консультации у специалистов по компьютерной технике.

Формирование стратегической рекламной кампании

Наиболее верным решением вышеозначенных проблем может стать проведение стратегической рекламной кампании, целью которой является создание нового имиджа компании, обеспечивающего не только консолидированность рекламных акций и увеличения объема продаж, но и успешное продвижение услуг компании на рынке соседних регионов [4].

Основной задачей рекламной кампании является формирование имиджа компьютерной компании как: обладающей устойчивой клиентурой, применяющей комплексный подход и индивидуально относящейся к клиенту.

Формирование рекламных материалов должно идти двумя путями:

1) создание имиджа, при котором происходит жесткая отстройка компании от конкурентов, показывается ее уникальность;

2) конкретизация ценностных линий компании.

Миссия компании является основой корпоративной идеологии, позволяющей сформировать единый стиль деятельности, выявить базовые ценности, минимизировать коммуникативные издержки [5].

Можно выделить следующие уровни понимания миссии компании:

1) предназначение: перспективы развития компании в рамках собственного региона и успешное продвижение услуг на рынке соседних областей, т. е. компания становится представителем региона за его пределами;

2) идентичность: стать признанным лидером в области компьютерных и информационных технологий;

3) ценности: высокопрофессиональная, активно расширяющаяся, обладающая устойчивой клиентурой, индивидуальным подходом к клиенту компания.

Выстраивая рекламную кампанию, следует использовать стратегию «движения изнутри», т. е. первоначальными элементами должны стать внутрикорпоративные нормы и правила, основанные на идеологии компании [6].

В данном случае под идеологией компании понимается изложение в краткой, тезисной форме предназначения компании. Сформированная идеология упрощает поиск и принятие решений в таких областях, как построение стратегических рекламных кампаний, работа с персоналом, взаимодействие с клиентами.

На основе миссии и идеологии создается фирменный стиль компании:

1) Логотип компании. Должен отражать идею выхода компании на новые рынки. Кроме того, две второстепенные ценностные линии – профессионализм и комплексное индивидуальное обслуживание.

2) Девиз и слоган компании. Девиз компании, в отличие от слогана, является более долговременным образованием. Слоган должен отражать именно текущую задачу – позиционируя компанию как активно продвигающую свои услуги на рынках соседних регионов.

3) Серия базовых текстов. История компании, информация о ее руководстве, ведущие направления работы и перспективы развития. Базовые тексты могут использоваться на протяжении всей рекламной кампании не только как основа для публикаций, но и для создания буклета, а также для текстовых сообщений при переговорах и обслуживании клиентов.

Буклет компании должен стать частью презентационной продукции, т. е. его необходимо распространять на выставках, семинарах, дарить клиентам, делать адресную рассылку. **Примерное содержание буклета:** краткая справка о компании, основные клиенты, уникальность и сила компании, интервью с директором, наименование предоставляемых услуг, выгоды, которые клиент получает при сотрудничестве с конкретной компанией. Необходимы качественные фотографии офиса, сотрудников, техники. Дизайн буклета должен быть выдержан в стиле компьютерного века.

Презентационный фильм также является важной частью рекламной продукции компании. **Основные требования к фильму:** он должен смотреться, увлекать, быть живым и динамичным, понятным дилетанту, он должен вызывать желание работать с компанией. Обязательно нужны динамичные проходы – по офису фирмы и по предприятиям клиентов. Весь фильм должен смотреться захватывающе – это главное. Должна быть энергичная, воодушевляющая постановка проблемы, конкретное, профессиональное решение, короткие, эффектные графические заставки.

Рекламная политика должна отражать несколько основных ценностных линий компании. Таким обра-

зом, статьи в прессе должны содержать: интервью с клиентами, историю создания компании, основные виды услуг, основные отличия от конкурентов. Рекламная информация должна содержать основную линию рекламной кампании – «Выход конкретной компании на рынки соседних регионов» и ценностную идею – «Мы не торгуем, мы создаем и заботимся». Также возможно размещение обзорных статей по проблемам развития компьютерного рынка и применения информационных технологий в современном бизнесе.

Одним из мощных информационных поводов должно стать проведение регулярных бесплатных семинаров в области информационных технологий. Помимо участия сотрудников самой компании, необходимо рассмотреть возможность приглашения специалистов из Нижнего Новгорода, Москвы, Санкт-Петербурга. Семинар должен стать не просто лекцией, которую специалисты читают для специалистов, а знаковым событием. На семинарах можно распространять всю рекламную продукцию и оказывать бесплатные услуги по консалтингу потенциальных клиентов.

Готовые рекламные материалы становятся (до их размещения и публикации) основой для **«рекламы рекламы»**. Другими словами, перед размещением материалов выставки, семинара, рассылки потенциальным клиентам необходимо ознакомить с образцами рекламы клиентов компании – в режиме «не могли бы вы высказать свое мнение по поводу нашего нового рекламного проекта?». Из этой процедуры возникает и размещение отзывов в буклете, и приглашение на семинар, и информационный повод контакта с клиентом.

В процессе создания стратегической имиджевой кампании можно достигнуть нескольких существенных для фирмы результатов.

Во-первых, благодаря детальному обсуждению миссии компании могут быть выявлены те базовые ценности и установки, которые позволяют в дальнейшем руководству компании уверенно проводить разработку рекламных акций и ориентироваться в вопросе бюджетирования рекламных кампаний.

Во-вторых, могут быть сформированы базовые тексты, способные служить основой при подготовке печатной продукции.

В результате многосоставных и многоступенчатых мероприятий компания сможет стать в общественном сознании лидирующей фирмой в области компьютерных технологий: лидерство это формируется как лидерство «специалистов», в противовес лидерству «успешных торговцев» [7].

Создание фирменного стиля, рассылка буклетов и диска с презентационным фильмом компании, а также проведение семинаров, безусловно, покажет передовой подход к информированию клиентов компании, что отлично вписывается в общую концепцию современного стиля культурной и деловой активности, пропагандируемого Полпредством ПФО.

Примечания

1. *Траут Д.* Новое позиционирование. СПб.: «Питер», 2002.
2. *Райс Э., Траут Д.* Маркетинговые войны. СПб.: «Питер», 2001.
3. *Джулер А. Дж., Бонни Л. Д.* Креативные стратегии в рекламе. СПб.: «Питер», 2002.
4. *Домнин В. Н.* Брэндинг: Новые технологии в России. СПб.: «Питер», 2002.
5. *Райс Э., Траут Д.* Позиционирование. Битва за узнаваемость. СПб.: «Питер», 2001.
6. *Минцберг Г.* Структура в кулаке. СПб.: «Питер», 2002.
7. *Д'Алессандро Д.* Война брэндов. СПб.: «Питер», 2002.

А. В. Скурихин
Ю. А. Скурихина

КРАСНО-ЧЕРНЫЕ ДЕРЕВЬЯ (RB-TREES)

Рассматриваются основные операции на абстрактной структуре данных, которая слабо отражена в литературе по алгоритмистике.

Основные операции с двоичным деревом поиска высоты h (вставка, удаление, поиск, следующий, предыдущий) могут быть выполнены за O(h) действий. Деревья эффективны, если их высота мала. Но малая высота не гарантируется, и в худшем случае деревья не более эффективны, чем списки. Проблему баланса будем решать нижеописанным способом.

Свойства

Красно-черные деревья [1], так же как и AVL-деревья [2], принадлежат к классу «сбалансированных» с высотой порядка **O(log *n*)**, но, в отличие от последних, имеют максимальную высоту **2log(*n*+1)***. Это обеспечивается следующими свойствами:

1) все вершины, помимо «ключа», имеют «цвет» O {красный, черный}**;

2) дети – **nil** – всегда черные***;

3) если вершина красная, то ее дети черные;

4) все пути от вершины до любого **nil** содержат равное количество черных вершин – их черные высоты равны.

* У AVL-деревьев высота не превосходит log *n*.

** Или {red, black}.

*** Мы заменили nil на nul – фиктивный элемент, у которого: c=black; el=<любой>; l=r=nil; p=<принимает значение p удаляемого элемента>. Во всех процедурах, кроме удаления элемента, nul равносильно nil.

СКУРИХИН Андрей Владимирович – студент III курса факультета информатики ВятГГУ
СКУРИХИНА Юлия Александровна – выпускница факультета информатики ВятГГУ

Вершина содержит три указателя типа **t_rb** (родитель **p**, правый **r** и левый **l** дети) - каркас дерева, ключ **el** – предмет упорядочивания, тип которого зависит от **t_el**, и цвет **c** – «балансировка» (см. *табл. 1* в *Приложении*).

Добавление вершины и вращение

При обычном добавлении элемента в дерево процедурой **tree_ins** (см. *табл. 2*), которая осуществляет эту операцию за время **O(log *n*)**, RB-баланс нарушается. Поэтому следует выполнять ряд действий для его восстановления. Для этого мы будем использовать «вращения», смысл которых ясен из *рис 1.*

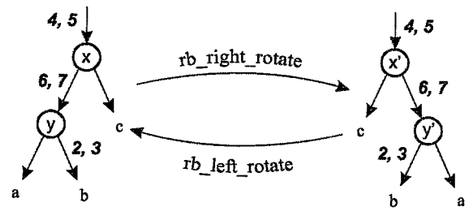


Рис. 1
Цифры рядом со стрелками соответствуют номерам строк процедур табл. 3

Как мы видим, правое и левое вращения симметричны (см. *табл. 3* и *рис. 4, 5 в Приложении*), а время выполнения есть **O(1)**.

Собственно добавление элемента осуществляется с помощью процедуры **tree_ins**, а балансировка дерева – основным кодом процедуры **rb_ins** (см. *табл. 4*). После выполнения строк 1, 2 наше дерево соответствует всем RB свойствам, кроме 3-го: у **x** – красной вершины может быть родитель красного цвета. На другие свойства вставка красной вершины не влияет.

В цикле рассматриваются три случая, каждый из которых распадается на пару симметричных, и определяются логической переменной **b**. Основной цикл конечен, так как вершина **x** каждую итерацию основного цикла поднимается не менее чем на один уровень, время выполнения сравнимо с **O(log *n*)**.

Для уменьшения количества граничных случаев мы считаем, что корень у RB-дерева всегда черный. Поэтому вершина **x^.p^.p^.l** существует.

Все возможные случаи, возникающие в ходе выполнения процедуры, наглядно иллюстрируются *рис. 2.*

Удаление элемента из RB-дерева

Осуществляется посредством процедуры **rb_delete** (см. *табл. 6*).

Именно здесь приобретает силу замена **nil** на **nul**, которая позволяет обходить некоторые граничные случаи. Теперь **«nil»**-лист логически – вершина, у которой действителен указатель на предка **p** и цвет **c**.

Начинаем с обычного удаления вершины из дерева. После чего возникает два случая:

1) удаление красной вершины не противоречит ни одному условию RB;

2) удаляемая вершина черная – для ее родителя черные высоты детей будут различны*. Требуется балансировка.

Второй случай приводит нас к процедуре **rb_delete_fixup** (*табл. 5*). Основной цикл продолжает работу до тех пор, пока вершина, на которую указывает **x**, не станет черной, или, корнем дерева. Однотипные случаи (симметричные) разделяются строчкой (1) кода.

После удаления в ветви дерева, на родителя которой указывает **x**, не хватает единицы черноты – исправляем посредством ее брата или родителя. Для подробного разбора кода мы приводим *рис. 3*, который наглядно иллюстрирует динамику изменения структуры дерева после преобразований.

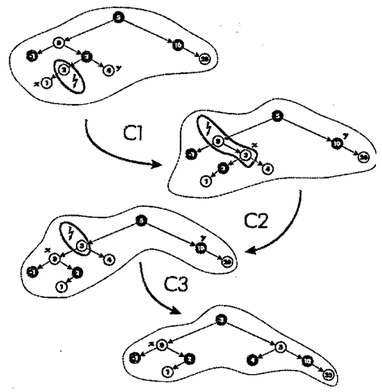


Рис. 2

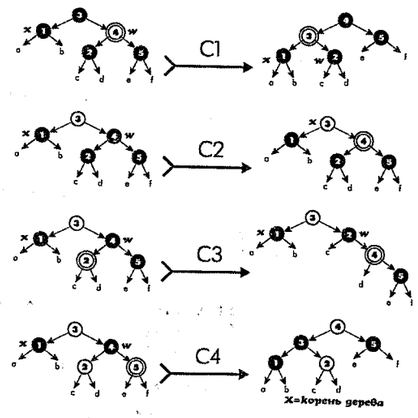


Рис. 3

* Частный случай: родитель удаляемой вершины отсутствует, тогда независимо от цвета этой вершины черная высота правого и левого ее детей будет одинаковой

Примечания

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2000.

2. *Вирт Н.* Алгоритмы + Структуры данных = Программы. М.: Мир, 1985.

3. *Окулов С. М.* Основы программирования: Учеб. пособие. Ч. 4. Киров: Изд-во ВГПУ, 2001.

Приложение

Индексация | **Код** | **№ таблицы и комментарий**

Таблица 1

```
   Const
    red=true
    black=false
  Type
     t_el=integer;
     t_rb=^t_rb_rec;
     t_rb_rec=record
       el : t_el;
        l,r,p:t_rb;
        c: boolean;
end;
  function rb_new(el:t_el; nul:pointer; c:boolean):t_rb;
var
   t:t_rb;
begin
   new(t);
    t^.l:nul;
    t^.r:nul;
    t^.el:nul;
    t^.p:nul;
    t^.c:nul;
    rb_new:=t;
end;
```

Таблица 2

t – корень дерева
z – указатель на вставляемый элемент
Здесь формируется вызовом функции rb_new. nul:=rb_new(0,nil,black), а <любой элемент дерева>:= rb_new (el, nul, red), что избавляет нас от написания дополнительного для инициализации самого nul

```
 procedure free_ins(var t:t_rb; z:t_rb);
var
 y,x:t_rb;
begin
    y:=nul;
    x:=t;
     while x<>nul do begin
       y:=x;
     if z^.el<x^.el then x:=x.1
     else x:=x^.r;
     end;
     z^.p:=y;
     if y=nul then t:=z
     else begin
        if z^.el<y^.el then y^.1:=z
       else y^.r:=z;
    end;
 end

 procedure rb_left_rotate (var t:t_rb; x:t_rb);
var
   y:t_rb;
 begin
1    y:=x^.r;
2    x^.r=y^.1;
3     if y^.1<>nil then y^.1^.p:=x;
4    y^.p:=x^.p;
5    else begin
5.1      if x=x^.p^.1 then x^.p^.1:=y
5.2     else x^.p^.r:=y;
   end;
6    y^.1:=x;
7    x^.p:=y;
 end;
```

Таблица 3

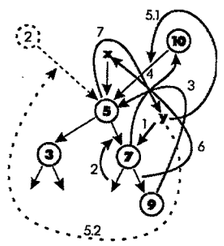


Рис. 4

Индексация | **Код** | **№ таблицы и комментарий**

```
  procedure rb_right_rotate (var t:t_rb; x:t_rb);
var
  y:t_rb;
begin
1   y:=x^.1;
2   x^.1:=y^.r;
3    if y^.r<>nil then y^.r^.p:=x;
4   y^.p=x^.p;
5   if x^.p=nil then t:=y
    else begin
5.1     if x=x^.p^.r then x^.p^.r:=y
5.2    else x^.p^.1:=y;
    end;
6    y^.r:=x;
7    x^.p:=y;
end;

 procedure rb_ins(var t:t_rb; x:t_rb);
var
   y:t_rb;
    b:boolean;
begin
1    tree_ins(t,x);
2    x^.c:=red;
       while (x<>t) end(x^.p^.c=red) do begin
         b:=(x^.p=x^.p^.p^.1);
C1-C1'  if b then y:x^.p^.p^.r
        else y:=x^.p^.p^.1;
         if (y<>nul) end(y^.c=red) then begin
           x^.p^.c:=black;
           y^.c:=black;
           x^.p^.p^.c:=red;
           x:=x^.p^.p;
        end else begin
C2-C2'     if x=x^.p^.1 then begin
             x:=x^.p;
              rb_right_rotate(t,x);
           end;
         end;
C3       x^.p^.c:=black;
         x^.p^.p^.c:=red;
C3-C3'   if b then rb_right_rotate(t,x^.p^.p)
          else rb_left_rotate(t,x^.p^.p);
     end;
  end;
   t^c:=black;
end;

 procedure rb_delete_fixup (var t:trb; x: trb);
var
  w:t_rb;
begin
    while (x<>t) end(x^.c=black) do begin

(1)    if x=x^.p^.1 then begin
         w:=x^.p^.r;
C1        if (w^.c=red) then begin
            w^.c:=black;
            x^.p^.c:=red;
             rb_left_rotate(t,x^.p);
            w:=x^.p^.r;
          end;
C2          if (w^.1^.c=black) end(w^.r^c=black) then begin
            w^.c:=red;
            x:=x^.p;
          end else begin
              if (w^.r^.c=black) then begin
```

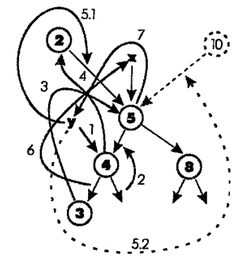


Рис. 5

Таблица 4

Таблица 5

Индексация	Код	№ таблицы и комментарий

```
                    w^.1^.c:=black;
                    w^.c:=red;
                     rb_right_rotate(t,w);
                    w:=x^.p^.r;
                  end;
                   w^.c:=x^.p^.c;
                   x^.p^.c:=black;
                   w^.r^.c:=black;
                    rb_left_rotate(t,x^.p);
                   x:=t;
                end;
              end else begin;
                 w:=x^.p^.l;
                  if (w^.c=red) then begin
                    w^.c:=black;
                    x^.p^.c:=red;
                     rb_right_rotate(t,x^.p);
                    w:=x^.p^.l;
                  end;
                    if (w^.1^.c=black) end(w^.r^.c=black) then begin
                     w^.c:=red;
                     x:=x^.p;
                  end else begin
                      if (w^.1^.c=black) then begin
                       w^.r^.c:=black;
                       w^.c:=red;
                       rb_left_rotate(t,w);
                       w:=x^.p^.l;
                    end;
                     w^.c:=x^.p^.c;
                     x^.p^.c:=black;
                     w^.1^.c:=black;
                      rb_right_rotate(t,x^.p);
                    x:=t;
                  end;
               end;
            end;
           x^.c:=black;
         end;

          function tree_next(x:t_rb):t_rb;
        var
           y:t_rb;
        begin
            if x^.r<>nul then begin
              x:=x^.r;
               while x^.l<>nul do x:x^.l;
               tree_next:=x;
            end else begin
               y:=x^.p;
                 while (y<>nul) end(x=y^.r) do begin
                  x:=y;
                   y:=x^.p;
               end;
                tree_next:=y;
           end;
        end;
          function rb_delete(var t:t_rb; z:t_rb):t_rb;
        var
           y,x:t_rb;
        begin
            if z<>nul then begin
               if (z^.l=nul)or(z^.r=nul) then y:=z
               else y:=tree_next(z);
               if y^.l<>nul then x:=y^.l
              else x:=y^.r;
              x^.p:=y^.p;
               if y^.p=nul then t:=x
               else begin
```

Таблица 6

Индексация	Код	№ таблицы и комментарий

```
                 if y=y^.p^.l then y^.p^.l:=x
                 else y^.p^.r:=x;
              end;
                if y<>z then z^.el:=y^.el;
                if y^.c=black then rb_fix(t,x);
               rb_delete:=y;
           end else rb_delete:=z;
        end;
```

В. И. Умнов

РАЗБОР ЗАДАЧ XV ОБЛАСТНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ

В статье приводится детальный разбор достаточно интересных олимпиадных задач по информатике.

Уже давно стало традицией организовывать областную олимпиаду по информатике открытой, и этот год не стал исключением. Достойный уровень задач привлекает школьников из разных регионов. В XV олимпиаде принимали участие школьники из Кировской области, Казани, Жуковского, Мытищ, Челябинска и Москвы. Открытость олимпиады позволяет сравнивать уровень и знания школьников, знакомить школьников разных регионов.

Задача 1. Текст (30 баллов. Ограничение по времени на каждый тест – 1 секунда)

Дан текст, состоящий из слов, знаков препинания и других символов. Словом в тексте считается последовательность символов из прописных и строчных букв латинского алфавита. Требуется перевернуть (записать в обратном порядке) все слова текста, оставив знаки препинания и другие символы, включая буквы русского алфавита, без изменений. В строке не более 255 символов, строк в файле не более 1000.

Входной файл: input.txt. Выходной файл: output.txt.

Пример:

входной файл	выходной файл
This is an example Prim333primяят	sihT si na elpmaxe mirP333mirряятr

Статистика решения: *45,12% участников полностью решили задачу, 18,29% набрали 0 баллов.*

Решение задачи

Эта задача, одна из двух, которая решалась «в лоб» и не содержала подвохов.

УМНОВ Владимир Игоревич – студент V курса факультета информатики ВятГТУ

1-й способ. При решении задачи на Pascal-е можно было использовать тип **string**, так как максимальная длина строки 255 символов. После считывания очередной строки необходимо изменить ее следующим образом: каждое найденное «слово» перевернуть на месте. Далее вывести эту строку.

2-й способ. Более элегантный подход заключатся в том, что не надо анализировать текст по строкам, можно представить его в виде последовательности символом с переводами кареток. Для переворота «слов», можно использовать концепцию работы стека, накапливая символы в нем. Рассмотрим его реализацию:

```
var s : string;
    c : char;
begin
 assign(input, "input.txt");
 reset(input);
 assign(output, "output.txt");
 rewrite(output);
 s := "";
 while not eof(input) do begin
  read(c);
   if c in ["A"..'Z', "a"..'z'] then
    s := c + s
   else begin
    write(s);
    s := "";
    write(c);
  end;
 end;
 write(s);
 close(input);
 close(output);
end.
```

Здесь до конца файла считывается по одному символу. Если текущий символ латинский, то он является частью слова, поэтому он приписывается в начало строки s. Если символ не латинский, то на данной итерации либо конец «слова», либо просто продолжение текста. В последнем случае необходимо вывести перевернутое слово, очистить строку s, а

также вывести только что считанный символ текста, уже не являющийся частью слова.

Комментарии

Многие школьники не учитывали тот случай, когда в конце файла находится «слово», но нет перевода каретки. В результате они теряли последнее слово. Было несколько школьников, которые забывали, что латинские буквы бывают заглавными.

Задача 2. «Блохи» (30 баллов. Ограничение по времени на каждый тест – 1 секунда)

На клеточном поле, размером NxM ($2 \le N, M \le 250$) сидит Q ($0 \le Q \le 10000$) блох в различных клетках. «Прием пищи» блохами возможен только в кормушке – одна из клеток поля, заранее известная. Блохи перемещаются по полю странным образом, а именно, прыжками, совпадающими с ходом обыкновенного шахматного коня. Длина пути каждой блохи до кормушки определяется как количество прыжков. Определить минимальное значение суммы длин путей блох до кормушки или, если собраться блохам у кормушки невозможно, сообщить об этом. Сбор невозможен, если хотя бы одна из блох не может попасть к кормушке.

Входной файл: input.txt.

В первой строке входного файла находится 5 чисел, разделенных пробелом: N, M, S, T, Q.

N, M – размеры доски (отсчет начинается с 1); S, T – координаты клетки-кормушки (номер строки и столбца соответственно), Q – количество блох на доске. И далее Q строк по два числа – координаты каждой блохи.

Выходной файл: output.txt.

Содержит одно число – минимальное значение суммы длин путей или –1, если сбор невозможен.

Примеры:

входной файл	выходной файл
4 4 1 1 3 2 3 3 2 3 3	6
5 5 3 4 0	0

Статистика решения: *10,98% участников полностью решили задачу, 42,68% набрали 0 баллов.*

Решение задачи

Для решения задачи необходимо использовать двумерный массив **board**. Элемент массива **board[x][y]** хранит минимальное расстояние, за которое можно добраться от клетки с координатами (x; y) до кормушки (S; T). Элемент этого массива не превысит значения 250, так как блохи прыгают конем, прыгают «в ширь», а также не прыгают на уже достигнутые клетки. Удобнее решать задачу с конца, то есть находить не расстояние от каждой блохи до кормушки, а расстояние от кормушки до всех потенциальных позиций блох на доске. Рассмотрим решение на первом примере (рис. 1-6):

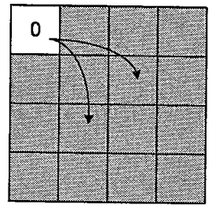


Рис 1. Исходная доска

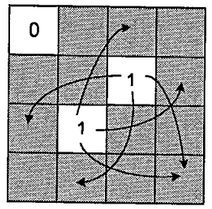


Рис 2. Первый шаг

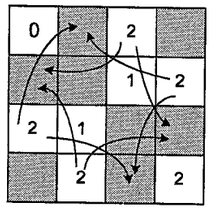


Рис 3. Второй шаг

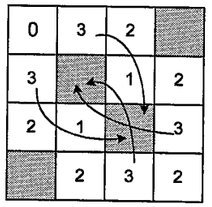


Рис 4. Третий шаг

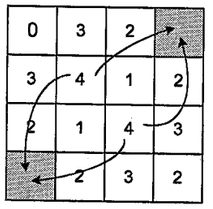


Рис 5. Четвертый шаг

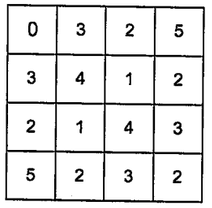


Рис 6. Конечная позиция

Закрашенные клетки – на текущий момент это недостижимые клетки. В начале работы алгоритма массив **board** представлен на рис. 1. На первом шаге (рис. 2), по алгоритму волны надо просмотреть возможные пути конем в закрашенные клетки из ячеек, содержащих 0, и присвоить им значение 1. На втором шаге (рис. 3) надо просмотреть все возможные пути конем в закрашенные клетки из ячеек, содержащих 1. Таким образом, каждый раз надо увеличивать длину пути на единицу. В итоге (рис. 6) получена доска с расстояниями от каждой клетки до кормушки.

Для обоих рассматриваемых ниже реализаций этой идеи целесообразно завести массив с запасом на 2 ячейки по горизонтали и по вертикали, пометив их как запретные, чтобы было удобнее контролировать ходы за границу доски.

1-й способ. Решение этой задачи сводится к алгоритму волны. Псевдокод будет выглядеть следующим образом:

```
while <были проведены изменения> do begin
 for i := 1 to N do
  for j := 1 to M do
   if board[i][j] = <текущий шаг> then
     <незанятые клетки «на расстоянии
       коня»> := <текущий шаг> + 1;
 <увеличить текущий шаг>;
end;
```


Осталось только посчитать сумму длин путей.

2-й способ. Очевидна неэффективность первого способа. Применение очереди дает значительный выигрыш, а сложность алгоритма будет оцениваться. Суть заключается в следующем:

1) изначально в очередь ставится клетка с координатами кормушки, **board[S][T]** присваивается 0;

2) взять из очереди следующий элемент, далее по порядку заполняются незанятые клетки «на расстоянии коня» в **board** и одновременно, в случае заполнения, эта клетка ставиться в очередь;

3) делать пункт 2, пока очередь не пуста.

Код программы, реализующий второй способ:

```
const
 MaxN    = 252;        {Максимальный размер
                        доски по обеим
                        координатам}
 MaxQueue = 10000;     {Максимальный
                        размер очереди}
 plEmpty  = 253;       {Обозначение незанятой
                        клетки}
 plNone   = 254;       {Обозначение границы вне
                        доски на расстоянии
                        2 клеток}
 {Смещение по горизонтали и вертикали
                  для 8 возможных ходов конем}
 dx: array[1..8] of integer=(-1,-1, 1, 1,-2,-2, 2, 2);

 dy: array[1..8] of integer=(-2, 2,-2, 2,-1, 1,-1, 1);
type
 TBoard = array [-1..MaxN, -1..MaxN] of byte;
 TPoint = record
  x, y : integer;
 end;
var
 board     : ^TBoard;  {Доска}
 queue     : array [0..MaxQueue-1] of TPoint;
                                   {Очередь}
 qr,qw     : integer;  {Указатель чтения
                        и записи для очереди}
 n,m,s,t,q : integer;

{
Процедура, осуществляющая ход конем:
• помечает незанятые клетки «на расстоянии коня» на доске значением на единицу больше;
• добавляет в очередь рассмотренные клетки;
• осуществляет работу с циклической очередью для экономии памяти;
}
procedure move(p : TPoint);
var i, tt, x, y : integer;
begin
 tt := board^[p.x][p.y] + 1;
 for i := 1 to 8 do begin
  x := p.x+dx[i];
  y := p.y+dy[i];
  if board^[x][y] = plEmpty then begin
   board^[x][y] := tt;
   queue[qw].x := x;
   queue[qw].y := y;
   qw := (qw+1) mod MaxQueue;
  end;
 end;
end;


{
Процедура инициализации:
• выделение памяти под доску;
• чтение входных данных;
• определение всех клеток пустыми, и установка граничных клеток;
}
procedure init;
var i, j : integer;
begin
 new(board);
 assign(input, "input.txt");
 assign(output, "output.txt");
 rewrite(output);
 reset(input);
 read(n, m, s, t, q);
 for i := -1 to m+2 do
  for j := -1 to n+2 do
   if (j<1) or (i<1) or (j>n) or (i>m) then
    board^[j][i] := plNone
    else
    board^[j][i] := plEmpty;
end;

procedure done;
begin
 dispose(board);
 close(input);
 close(output);
 halt(0);
end;

{
Процедура заполнения доски:
• изначально ставит в очередь клетку с кормушкой;
• пока очередь не пуста, для каждого следующего элемента делает ходы;
}
procedure prepare;
begin
 qr := 0; qw := 1;
 with queue[0] do begin x := s; y := t; end;
 board^[s][t] := 0;
 while qw <> qr do begin
  move(queue[qr]);
  qr := (qr+1) mod MaxQueue;
 end;
end;
```

```
{
Процедура подсчета суммы длин путей:
• продолжение чтения исходных данных: координаты блок;
• суммирование путей;
• если блоха стоит на пустой клетке, значит, в нее попасть нельзя, вывод -1 и выход;
}
procedure solve;
var ss, i : longint;
   x, y : integer;
begin
 ss := 0;
 for i := 1 to q do begin
  read(x, y);
  if board^[x][y] <> plEmpty then
   inc(ss, board^[x][y])
   else begin
   writeln(-1);
    done;
   end;
 end;

 writeln(ss);
end;

begin
 init;
 prepare;
 solve;
 done;
end.
```

Задача 3. Отчет (40 баллов. Ограничение по времени на каждый тест – 3 секунды)

Один из цехов завода производит продукцию в течение N месяцев. Начальнику цеха было поручено составить отчет о росте производительности данного цеха и об уменьшении доли некачественной продукции в процентном соотношении (точность доли процента до одного знака после запятой, например, 2/7=0.(285714)≤ 28.6%). При этом в отчет должна войти информация за возможно большее число месяцев K (K ≤ N) работы цеха. Начальник цеха решил, что он включит в отчет данные только по тем месяцам (не обязательно взятым подряд, но обязательно в хронологическом порядке), по которым наблюдается строгий рост количества производимой продукции и строгий спад доли бракованных товаров по сравнению с данными предыдущего месяца, вошедшего в отчет. Определить, какое максимальное количество месяцев удовлетворяет этим условиям и сколько есть возможных вариантов составления отчета.

Входной файл: input.txt.

Первая строка файла содержит число N (1 ≤ N ≤ 40) – количество месяцев работы цеха. Далее следует N строк, содержащих целые числа vi (1 ≤ vi ≤ 10000) и bi (1 ≤ bi ≤ vi); vi – объем продукции, произведенной цехом за i-й месяц; bi – количество бракованной продукции в i-м месяце.

Выходной файл: output.txt.

Первая строка файла содержит число K – количество месяцев, по которым будет включена в отчет информация о работе цеха. Вторая строка содержит число P – количество возможных вариантов составления отчета с максимальным содержанием.

Пример

входной файл	выходной файл
10 10 3 20 5 15 5 45 9 65 5 70 14 68 17 64 16 70 16 70 15	4 1

Статистика решения: 10,98% *участников полностью решили задачу,* 65,85% *набрали 0 баллов.*

Решение задачи

Рассмотрим решение задачи с использованием динамического программирования. Для наглядности рассмотрим алгоритм на исходном примере. Представим условие задачи в виде матрицы смежности **a**. Элемент **a[j][i]** будет равняться 0 или 1, причем равенство 1 означает, что из **j** месяца можно «попасть» в **i**, то есть производительность цеха в месяце **j** больше производительности в месяце **i** и процент брака с точностью до одной десятой процента в месяце **j** меньше, чем в **i**. Матрица **a** выглядит следующим образом:

i

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	1	1	1	1	1	1
2	0	0	0	1	1	1	0	0	1	1
3	0	0	0	1	1	1	1	1	1	1
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

По условию задачи будет использована только правая верхняя половина матрицы, так как должна сохраняться хронологическая последовательность месяцев. Введем еще одну ключевую матрицу **d**. Элементом **d[i][k]** будет являться количество вариантов,

с помощью которых можно построить последовательность длины **k**, заканчивающуюся в **i** месяце. Рассмотрим алгоритм заполнения такой матрицы. Очевидно, что первая строка будет единичной, так как из одного месяца мы можем всегда сделать последовательность длины 1, заканчивающуюся в том же самом месяце. Далее начнем заполнять вторую строку:

• очевидно, что не существует последовательности длины 2, заканчивающейся в 1-м месяце;

• последовательность длины 2, заканчивающуюся во втором месяце, можно получить одним способом, так как последовательность длины 1, идущая ранее в хронологическом порядке только 1, а также из первого месяца можно «попасть» во второй;

• в третий месяц «попасть» нельзя ни из первого, ни из второго, поэтому таких последовательностей нет;

• в четвертый месяц можно «попасть» из всех трех предыдущих, поэтому будет три таких последовательности и т.д.

Можно записать формулу для вычисления **d[i][k]**:

$$d_{i,k} = \sum_{j=1}^{i-1} d_{j,k-1} \cdot a_{j,i}, \quad \text{где } i \geq k.$$

Применительно к данному примеру матрица **d** будет выглядеть следующим образом:

i – последний месяц последовательности

k – длина последовательности	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2		1	0	3	4	3	2	2	5	5
3			0	1	4	1	0	0	5	5
4				0	1	0	0	0	0	0
5					0	0	0	0	0	0
6						0	0	0	0	0
7							0	0	0	0
8								0	0	0
9									0	0
10										0

Для получения ответа остается найти первую ненулевую с конца строку и сосчитать в этой строке сумму элементов. Приведенный алгоритм решает более широкую задачу – он позволяет найти не только количество вариантов с максимальным K, но и количество вариантов с любым K. Программный код, решающий эту задачу:

```
const MaxN = 40;
var a    : array [1..MaxN, 1..MaxN] of byte;
  d    : array [1..MaxN, 1..MaxN] of longint;
  v, b  : array [1..MaxN] of integer;
  n    : integer;

procedure init;
var i, j : integer;
begin
 assign(input, "input.txt");
 assign(output, "output.txt");
 reset(input);
 rewrite(output);
 read(n);
 for i := 1 to n do
  read(v[i], b[i]);
 fillchar(a, sizeof(a), 0);
 fillchar(d, sizeof(d), 0);
 for i := 1 to n-1 do
  for j := i+1 to n do
   if (v[j] > v[i]) and (round(b[j]/v[j]*1000) <
                round(b[i]/v[i]*1000)) then
    a[i][j] := 1;
end;

procedure solve;
var i, j, k : integer;
begin
 for i := 1 to n do
  d[i][1] := 1;
 for k := 2 to n do
  for i := k to n do
   for j := 1 to i-1 do
    d[i][k] := d[i][k] + d[j][k-1]*a[j][i];
end;

procedure done;
var i, k : integer;
   res  : longint;
begin
 for k := n downto 1 do begin
  res := 0;
  for i := 1 to n do
   res := res + d[i][k];
  if res <> 0 then begin
   writeln(k);
   writeln(res);
   break;
  end;
 end;
 close(input);
 close(output);
end;

begin
 init;
 solve;
 done;
end.
```

Задача 4. Точки (30 баллов. Ограничение по времени на каждый тест – 1 секунда)

На окружности расположено N точек. Их положение определяется углом φ между осью OX и радиусом, проведенным от центра окружности к этой точке. Угол задается в градусах. Никакие две точки на

окружности не совпадают. Требуется среди данных точек найти такие, чтобы сумма расстояний по окружности от каждой из этих точек до всех остальных была минимальна. Расстояние по окружности пропорционально минимальному углу между радиусами, проведенными к этим точкам, поэтому сумму расстояний следует вычислять как сумму углов.

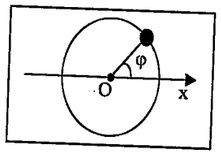


Рис. 1

Входной файл: input.txt.

Первая строка файла Input.txt содержит целое число N ($1 \le N \le 360$) – количество точек. Далее следует N строк: каждая строка содержит целое число φ ($1 \le \varphi \le 360$), определяющее положение точки на окружности.

Выходной файл: output.txt.

Первая строка файла Output.txt содержит число K – количество точек, удовлетворяющих условию задачи. Далее следует K строк, содержащих номера этих точек в порядке считывания данных из файла. Номера точек требуется вывести в порядке возрастания номеров.

Пример

входной файл	выходной файл
4 315 350 47 12	2 2 4

Статистика решения: *58,54% участников полностью решили задачу, 13,41% набрали 0 баллов.*

Решение задачи

Для решения задачи необходимо считать входные данные и вывести выходные. Все. А теперь серьезно. Чтобы посчитать расстояние до точки, можно воспользоваться формулой $r \cdot \alpha$, где r – радиус окружности, а α – угол в радианах, но и это здесь не требуется, так как в условии задачи явно указано, что расстояние можно определять углом. Вычислить расстояние между парами точек можно с помощью формулы $\min(|\alpha_2 - \alpha_1|, 360 - |\alpha_2 - \alpha_1|)$. Далее считается расстояние от i точки до всех остальных и суммируется. И так для каждой точки. Из всех сумм выбираются минимальные, и выводятся номера соответствующих точек в возрастающем порядке. Сложность такого алгоритма $O(n^2)$.

Текст программы:

```
var n, i, j, k, t, min, sum : longint;
   a, b : array [1..360] of integer;

function minimum(a, b : longint) : longint;
begin
 if a < b then minimum := a else minimum := b;
end;

begin
 fillchar(a, sizeof(a), 0);
 assign(input, "input.txt");
 reset(input);
 read(n);
 for i := 1 to n do begin
  read(a[i]);
 end;
 close(input);

  k := 0;
 min := 360*360;
 for i := 1 to n do begin
  sum := 0;
  for j := 1 to n do
    if i<>j then
    inc(sum, minimum(360 - abs(a[i] - a[j]),
                                    abs(a[i] - a[j])));
 if (sum < min) and (sum > 0) then begin
    min  := sum;
    k    := 1;
    b[k] := i;
 end else if sum=min then begin
  inc(k);
  b[k] := i;
 end;
end;
 if n = 1 then begin
  k   := 1;
  b[k] := 1;
 end;
 assign(output, "output.txt");
 rewrite(output);
 writeln(k);
 for i := 1 to k do
  writeln(b[i]);
 close(output);
end.
```

Примечание

Многие участники попытались найти линейное или хотя бы логарифмическое решение, но из статистики видно, что оно совсем не тривиальное и у них не хватило техники на его реализацию. Некоторые участники некорректно обрабатывали случай, когда точка всего одна.

Задача 5. Город Киров (30 баллов. Ограничение по времени на каждый тест – 2 секунды) (рис. 2)

Через г. Киров проходит железная дорога (считать, что она не имеет ответвлений), расстояния на которой отсчитываются от г. Москвы. Новый министр железнодорожного транспорта с целью придания единообразия приказал переименовать все небольшие станции. После этого станции стали иметь названия – такой-то километр от г. Москвы (например, «910-й км»).

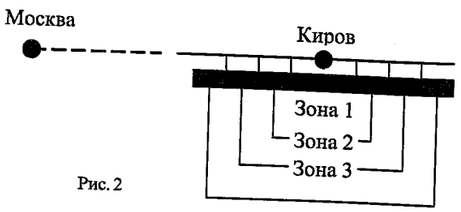


Рис. 2

Местный заместитель министра, с целью увеличения поступлений в местный бюджет, все пригородные станции распределил по зонам (стоимость проезда до всех станций в одной зоне одинакова, независимо от расстояния), с одинаковой протяженностью L. Если станция находится на границе двух зон, то она может быть отнесена к любой из них в зависимости от настроения кассира, продающего билеты.

Требуется по N станциям, для которых известны расстояния от г. Москвы и номера зон, которым они принадлежат, определить расстояние (в км) от г. - Москвы до г. Кирова. (Нет совпадающих станций, г. Киров не рассматривается как станция, все расстояния – целые числа.)

Входной файл: input.txt.

В первой строке числа N (0<N<201), L, затем для каждой станции расстояние от г. Москвы и номер зоны. (Все расстояния < 1000000001.)

Выходной файл: output.txt.

В выходной файл вывести расстояние от г. Москвы до г. Кирова (одно из возможных) или -1, если задача не имеет решения.

Примеры

входной файл	выходной файл
5 10 910 1 900 3 925 1 940 2 977 6	920
4 9 919 1 910 2 929 2 920 1	-1

Статистика решения: *7,32% участников полностью решили задачу, 26,83% набрали 0 баллов.*

Решение задачи

Очень важно осознать условие задачи и ограничения. Переформулировав условие, можно сказать, что требуется определить правильность расстановки станций по зонам. Если они расставлены правильно, то можно определить возможные границы первой зоны, если нет, то надо просто вывести -1. В условии явно не сказано, что г. Киров лежит в центре первой зоны, хотя это и отображено на рисунке. Существенно и то, что г. Киров не может быть одной из станций. Ограничения на расстояние и время работы отсекают попытку перебрать все возможные точки между станциями, где может лежать г. Киров (их можно получить, отсортировав станции по зонам, далее, идя слева, наткнувшись на возрастание номера зоны, определить правую границу, а левая граница будет первая отличающаяся по номеру зоны станция, находящаяся левее), и сопоставлением расстановки зон на имеющиеся станции. Рассмотрим способы решения задачи на первом входном примере.

1-й способ. Относительно каждой станции можно выдвинуть первое утверждение: первая зона лежит либо слева – отрезок $[d - k \cdot L; d - (k-1) \cdot L]$, либо справа – отрезок $[d + k \cdot L; d + (k-1) \cdot L]$, где d – расстояние от Москвы до текущей станции, а k – номер зоны текущей станции. Сделав такие утверждения относительно всех станций, можно их совместить, то есть пересечь отрезки. Очевидно, что после пересечения всех отрезков по первому утверждению может получиться не более двух результирующих отрезков. Вот, кажется, и все решение задачи, но нельзя торопиться. Если станция находится в 1-й зоне, то, воспользовавшись первым утверждением, мы не учтем того, что г. Киров не может быть станцией, поэтому утверждение изменится на $[d - k \cdot L; d - (k-1) \cdot L]$ и $[d + k \cdot L; d + (k-1) \cdot L]$ соответственно (так как числа целые, то, уменьшив правую границу первого выражения на 1 и увеличив левую границу второго выражения на один, можно снова получить вид, как в первом утверждении). Итак, если все станции находятся в первой зоне, то может получиться n+1 отрезок. Для реализации потребуется набор отрезков в текущий момент времени, отражающий возможное положение первой зоны, с которым для каждой зоны будем производить пересечение с парой гипотез о нахождении первой зоны. В частности, для первого примера этим способом можно показать, что только одна точка может быть ответом (рис. 3).

Рассмотрим случай, когда несколько городов лежит в 1-й зоне и образуется более двух отрезков (рис. 4).

Слева на рисунке отображены исходные отрезки, справа отображен процесс пересечения. Это худший случай – получилось n+1 отрезков. Такое решение будет работать со сложностью от O(n) до $O(n^2)$, что в принципе не много. Текст программы с комментариями представлен ниже:

```
const
  MaxN = 201; {максимальное количество
                                    отрезков}
  infinity = MaxLongint;  {обозначение
                                    бесконечности}
type
  {тип для описание отрезка}
  sector = record
```

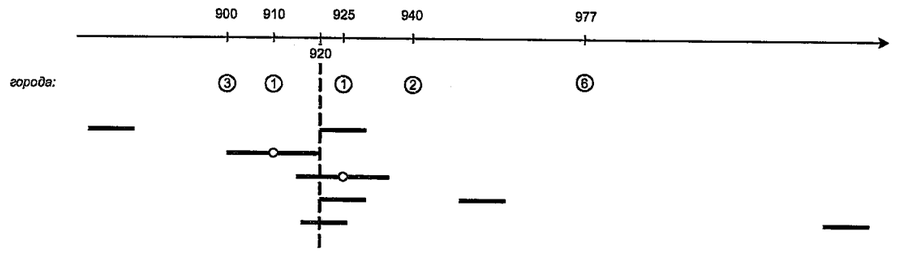


Рис. 3

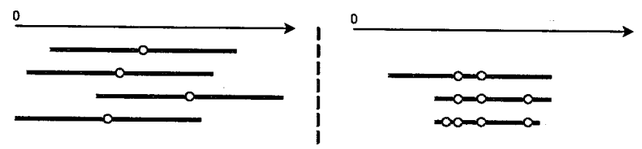


Рис. 4

```
    lx, rx : longint;
  end;
  sectors = array [1..MaxN] of sector; {тип для массива отрезков}
  var n, l, cnt : longint;    {кол-во станций, длина зоны, кол-во отрезков}
      s       : sectors;
                {массив отрезков}
```

{
Процедура инициализации:
• чтение n, l;
• вначале в массиве отрезков есть только один: от 0 до бесконечности – нужно для правильной работы алгоритма;
}

```
procedure init;
begin
  assign(input, "input.txt");
 reset(input);
 read(n, l);
 cnt := 1;
 s[cnt].lx := 0; s[cnt].rx := infinity;
end;
```

{
Процедура вывода результата:
• если после пересечения получено пустое множество, то нет решения;
}

```
procedure done;
begin
 close(input);
 assign(output, "output.txt");
 rewrite(output);
 if cnt > 0 then
  writeln(s[1].lx)
 else
  writeln(-1);
 close(output);
end;
```

{
Процедура пересечения отрезков:
• полученный отрезок от пересечения a и b записывается в r;
• если отрезки не пересекаются, то r.lx будет r.rx
}

```
procedure intersect(const a, b : sector; var r : sector);
begin
  r := a;
  if b.lx > r.lx then r.lx := b.lx;
  if b.rx < r.rx then r.rx := b.rx;
end;
```

{
Процедура получения пересечений двух отрезков со всеми остальными
}

```
procedure work(const a, b : sector);
var ncnt, i : longint;
    ns      : sectors;
    t       : sector;
begin
 ncnt := 0;
 for i := 1 to cnt do begin
  intersect(a, s[i], t);
  if t.lx <= t.rx then begin inc(ncnt); ns[ncnt] := t;
end;
```



```
    intersect(b, s[i], t);
    if t.lx <= t.rx then begin inc(ncnt); ns[ncnt] := t;
end;
    end;
    s   := ns;
    cnt := ncnt;
  end;
```

{
Процедура основной логики программы:
• считывает очередной город: расстояние от Москвы (d) и номер зоны (k);
• формируем два отрезка: один отрезок [d-n*l;d-(n-1)*l], а второй отрезок [d+(n-1)*l;d+n*l];
• если город в 1-й зоне, то нужно укоротить соответствующие края пары отрезков;
• эти два отрезка накладываются на имеющееся множество отрезков s (work);
}

```
procedure solve;
var d, k, i : longint;
   t1, t2  : sector;
begin
 for i := 1 to n do begin
  read(d, k);
  t1.lx := d + (k-1)*l;
  t1.rx := d + k*l;
  t2.lx := d - k*l;
  t2.rx := d - (k-1)*l;
  if k = 1 then begin
   inc(t1.lx);
   dec(t2.rx);
  end;
 work(t1, t2);
 end;
end;

begin
 init;
 solve;
 done;
end.
```

2-й способ. Является улучшением первого и позволяет решать задачу почти за линейное время, но, к сожалению, вывод результата в этом случае иногда становится квадратичным. Что будет происходить, если не учитывать ситуацию со станциями в 1-й зоне? А ничего, кроме возможного попадания г. - Кирова в уже существующие станции. Это можно легко поправить, сохранив расстояние до станций в 1-й зоне, и при выводе результата их учесть. Тогда будет только 3 возможных случая: на текущий момент либо 2 отрезка, либо 1, либо пустое множество, причем если уже был 1 отрезок, то 2 далее не получится, так же и с пустым множеством. Рассмотрим изменения в исходном коде 1-го способа:

Переменные и константы (изменение отмечено жирным шрифтом):

```
const
  MaxN = 200;  {Максимальное количество городов}
  MaxS = 2;    {Максимальное количество возможных отрезков}
type
 {Тип Отрезки}
 sectors = array [1..MaxS] of sector;
var
 s     : sectors;   {Отрезки}
 bad   : array [1..MaxN] of longint;
                    {Города из 1-й зоны}
 bcnt  : longint;   {Кол-во городов из 1-й зоны}
```

Добавленная и переписанные функции:

{
Функция определения совпадения точки с точками из 1-й зоны}

```
function belong(x : longint) : boolean;
var i : longint;
begin
 belong := false;
 for i := 1 to bcnt do
  if bad[i] = x then begin
   belong := true;
   break;
  end;
end;
```

{
Процедура вывода результата:
• если отрезков 0, то нет решения задачи;
• если отрезков больше 0, то ищется любая точка из возможных 2 отрезков, не совпадающая с точками из 1-й зоны, и если таковая найдена, то это и есть один из возможных ответов, иначе задача решения не имеет
}

```
procedure done;
var i, x : longint;
begin
 close(input);
 assign(output, "output.txt");
 rewrite(output);
 if cnt > 0 then begin
  i := 1;
 while (i <= cnt) and (belong(s[i].lx)) do begin
  inc(s[i].lx);
  if s[i].lx = s[i].rx then begin
   inc(i);
  end;
 end;
```

```
    if s[i].lx <= s[i].rx then
     writeln(s[i].lx)
    else
     writeln(-1);
   end else
    writeln(-1);
  close(output);
end;
```

{
Процедура основной логики программы:
• считывает очередной город: расстояние от Москвы (d) и номер зоны (k);
• если город в 1-й зоне, то заносим его в массив bad;
• формируем два отрезка: если город в 1-й зоне, то один отрезок [d-l;d+l], а второй отрезок пустой (где-то там); иначе один отрезок [d-n*l;d-(n-1)*l], а второй отрезок [d+(n-1)*l;d+n*l];
• эти два отрезка накладываются на имеющееся множество отрезков s (work);
}

```
procedure solve;
var d, k, i : longint;
    t1, t2  : sector;
begin
 bcnt := 0;
 for i := 1 to n do begin
  read(d, k);
  if k = 1 then begin
   t1.lx := d - l;
   t1.rx := d + l;
   t2.lx := -200000001; t2.rx := -200000001;
   inc(bcnt);
   bad[bcnt] := d;
  end else begin
   t1.lx := d + (k-1)*l;
   t1.rx := d + k*l;
   t2.lx := d - k*l;
   t2.rx := d - (k-1)*l;
  end;
  work(t1, t2);
 end;
end;
```

Хотя уже и говорилось о недостатках этого метода, он все же работает быстрее первого, так как сущность выполняемых операций в квадратичной части намного проще, чем у первого способа (цикл и условие против получения пересечений двух и вплоть до n отрезков).

3-й способ. Он кардинально отличается от первых двух. Он действительно дает линейный алгоритм работы, если не считать, что надо предварительно отсортировать станции по расстоянию от Москвы. Итак, станции расположены по порядку от Москвы. Так как первая станция самая левая, то можно предположить, что Киров лежит справа. Аналогично для следующих станций. Будем хранить в массиве **lsec[i]** пересечение **i**-го предположения и предположений для всех лежащих левее станций, это можно сделать с помощь пересечения этого промежутка с предыдущим, то есть **lsec[i-1]**. Сделаем аналогичное предположение и с правой стороны и сохраним эти пересечения в **rsec**, но в отличие от заполнения **lsec**, **rsec[i]** не будет включать предположение об **i**-й станции. Это делается за 2N операций пересечения. Далее самое интересное. Зададимся вопросом, как можно определить, находится ли Киров между двумя соседними станциями? С помощью введенных нами структур это делается очень легко. Для **i** от 0 до N:

• есть обобщенное предположение, что г. Киров лежит правее **i**-й станции в отрезке **lsec[i]** (отрезок включает предположение относительно **i**-й станции);
• есть обобщенное предположение, что г.Киров лежит левее **(i+1)**-й станции в отрезке **rsec[i]** (отрезок не включает предположение относительно **i**-й станции, а только относительно станций правее **i**);

Тогда для определения истинности высказывания «Киров находится между **i** и **i+1** станциями» надо пересечь **lsec[i]** и **rsec[i]**. Если получен не пустой отрезок, то в нем наверняка будет лежать Киров. Для того, чтобы быть уверенным, что Киров не является станцией, надо пересечь его еще с отрезком между **i** и **i+1** станциями, не включая самих станций. И если этот отрезок не пуст, то ответ найден! Ниже приведено решение этим способом:

```
const
 MaxN = 200; {Максимальное количество городов}
 infinity = MaxLongint; {Обозначение
                                бесконечности}
type
 {Тип Отрезок}
 sector = record
  lx, rx : longint;
 end;
 {Тип станция}
 station = record
  d, k : longint;
 end;
var n, l     : longint;   {Количество городов,
                                длина зоны}
 lsec, rsec : array [0..MaxN+1] of sector;
                {Массив для левосторонних и
                 правосторонних предположении}
 s      : array [0..MaxN+1] of station;
                                {Массив станций}
 line     : sector;
```

{
Процедура инициализации: чтение входных данных.
}

```
procedure init;
var i : longint;
begin
 line.lx := 0; line.rx := infinity;
 assign(input, "input.txt");
 reset(input);
 read(n, l);
 for i := 1 to n do
  read(s[i].d, s[i].k);
 close(input);
end;
```

{
Процедура вывода результата и завершения программы.
}

```
procedure done(d : longint);
begin
 assign(output, "output.txt");
 rewrite(output);
 writeln(d);
 close(output);
 halt(0);
end;
```

{
Процедура пересечения отрезков:
• полученный отрезок от пересечения a и b записывается в r;
• если отрезки не пересекаются, то r.lx будет r.rx
}

```
procedure intersect(const a, b : sector; var r : sector);
begin
 r := a;
 if b.lx > r.lx then r.lx := b.lx;
 if b.rx < r.rx then r.rx := b.rx;
end;
```

{
Процедура подготовки к вычислению ответа:
• сортировка станций по расстоянию от Москвы;
• так как Киров может лежать и левее самой левой станции, то s[0] присваивается 0, аналогично Киров может лежать правее самой правой, s[n+1] присваивается бесконечность;
• для запуска алгоритма пересечений необходимо, чтобы изначально начали пересекать с прямой; для этого достаточно использовать псевдопрямую от 0 до бесконечности;
• получение пересечений в lsec, предполагая, что Киров справа;
• получение пересечений в rsec, предполагая, что Киров слева;
}

```
procedure prepare;
var
 i, j : longint;
 t    : station;
 p    : sector;
begin
 for i := 1 to n - 1 do
  for j := i + 1 to n do
   if s[j].d < s[i].d then begin
    t    := s[i];
    s[i] := s[j];
    s[j] := t;
   end;
 s[0].d := 0;
 s[n + 1].d := infinity;
 lsec[0] := line; rsec[n] := line;

 for i := 1 to n do begin
  p.lx := s[i].d + (s[i].k - 1)*l;
  p.rx := p.lx + l;
  intersect(lsec[i - 1], p, lsec[i]);
 end;
 for i := n downto 1 do begin
  p.lx := s[i].d - s[i].k*l;
  p.rx := p.lx + l;
  intersect(rsec[i], p, rsec[i - 1]);
 end;
end;
```

{
Процедура, реализующая основную логику:
• проверка возможности наличия Кирова между двумя соседними станциями, без учета самих станций;
• вывод результата и завершение программы в случае успеха;
}

```
procedure solve;
var i     : longint;
   u, p, t : sector;
begin
 for i := 0 to n do begin
  u.lx := s[i].d + 1;
  u.rx := s[i + 1].d - 1;
  intersect(lsec[i], rsec[i], t);
  intersect(u, t, p);
  if p.lx <= p.rx then
   done(p.lx);
 end;
end;

begin
 init;
 prepare;
 solve;
 done(-1);
end.
```

Примечание

Были неудачные попытки решить задачу дихотомией, определив, между какими городами может лежать г. Киров, последовательно деля расстояние пополам найти точное расположение за log(x) операций, где x – максимальное расстояние между парой городов. Но, похоже, не смогли решить проблему до кон-

ца с определением направления, в которое нужно двигаться на каждом шаге.

Кировская область остается одним из лидеров в России по подготовке школьников в олимпиадной информатике. В организационном плане ежегодно проводятся следующие мероприятия: областная открытая олимпиада, открытый командный турнир по программированию, летняя компьютерная школа. Все это стало возможным благодаря позиции, занимаемой в этой образовательной области Вятским государственным гуманитарным университетом и Кировским физико-математическим лицеем.

Шестая задача рассмотрена в статье Р. В. Шарыгина (см. с. 230).

Все материалы олимпиады могут быть получены по адресу в Интернет: http://www.olympiads.ru/regional/kirov/2003/downloads.shtml.

М. А. Ходыкина

ОБ АЛГОРИТМИЧЕСКОЙ РЕАЛИЗАЦИИ ТЕОРЕМЫ МЕНГЕРА

В статье рассмотрены особенности алгоритмической реализации известной теоремы теории графов.

Теория графов является важной частью вычислительной математики. С помощью этой теории решается большое количество задач из различных областей. Граф состоит из множества вершин и множества ребер, которые соединяют между собой вершины. С точки зрения теории графов не имеет значения, какой смысл вкладывается в вершины и ребра. Вершинами могут быть населенные пункты, а ребрами – дороги, соединяющие их, или вершинами могут являться подпрограммы; соединение вершин ребрами означает взаимодействие подпрограмм.

Проблема связности графа является ключевой в данной теории. А теорема Карла Менгера (1840-1921) придает проблеме связности графа более точный и строгий смысл. Хотя интуитивно очевидно, что граф тем более связен, чем больше существует различных цепей, соединяющих одну вершину с другой, и тем менее связен, чем меньше нужно удалить промежуточных вершин, чтобы отделить одну вершину от другой, теорема Менгера помогает нам установить точное число вершинно-непересекающихся цепей, разделяющих вершины *u* и *v*.

Пусть *G(V,E)* – связный граф, *u* и *v* – две его несмежные вершины.

ХОДЫКИНА Мария Александровна – студентка IV курса факультета информатики ВятГГУ

Две цепи **<*u,v*>** называются вершинно-непересекающимися, если у них нет общих вершин, отличных от *u* и *v*.

Пример вершинно-непересекающихся <u,v>-цепей (рис. 1):

Между вершинами *u* и *v* существует 4 цепи: 1-2-4-6; 1-3-5-6; 1-2-5-6; 1-3-5-2-4-6. Но только первая и вторая являются вершинно-непересекающимися.

Две цепи называются реберно-непересекающимися, если у них нет общих ребер. Цепи могут пересекаться вершинно, но не пересекаться реберно, например цепи 1-2-**4**-6-7 и 1-3-**4**-5-7 (рис. 2).

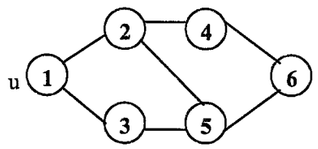


Рис. 1

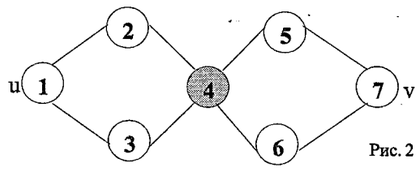


Рис. 2

Если две цепи вершинно не пересекаются, то они и реберно не пересекаются.

Теорема Менгера

Пусть *u* и *v* – несмежные вершины в графе *G*. Наименьшее число вершин в множестве, разделяющем *u* и *v*, равно наибольшему числу вершинно-непересекающихся простых <*u,v*>-цепей. (Доказательство теоремы можно найти в [1])

Варианты теормы Менгера

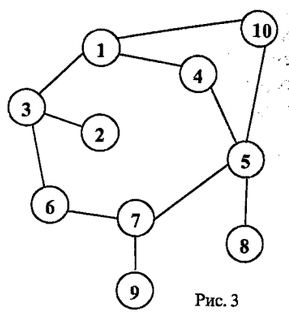
Теорема Менгера представляет собой весьма общий факт, который в разных формулировках встречается в различных областях математики.

Вариант 1. Для любых двух несмежных вершин *u* и *v* графа *G* наибольшее число реберно-непересекающихся <u,v>-цепей равно наименьшему числу ребер в (u,v)-разрезе (разделяющее множество ребер).

Вариант 2. Чтобы граф *G* был *k-связным*, необходимо и достаточно, чтобы любые две несмежные вершины были соединены не менее чем *k* вершинно-непересекающимися простыми цепями.

Наша цель состоит в том чтобы написать программу для решения задачи нахождения для каждой пары вершин графа наибольшего числа реберно-непересекающихся <u,v>-цепей.

Для примера рассмотрим граф ***G*** , хранящийся в файле в виде матрицы смежности *A* (рис. 3):




```
 10
0011000001
0010000000
1100010000
1000100000
0001001101
0010001000
0000110010
0000100000
0000001000
1000100000
```

Рис. 3

Идея решения.

Для решения задачи нам нужно просмотреть все пары вершин *u* и *v*. Из вершины *u* идем в глубину к вершине *v*. Как только находим вторую, возвращаемся и продолжаем поиск новой цепи. Вершины, которые находятся в первой <u,v>-цепи, запоминаем, чтобы они не встречались в следующей цепи, иначе столкнемся с пересечением этих цепей.

Возможны 3 случая.

1. Если вершины смежные (A[i,j]=1), то рассматривать этот случай не нужно.

2. Если валентность вершины = 1 и вершины не смежные (A[i,j]<>1), то очевидно, что существует только одна вершинно-непересекающаяся цепь от этой или к этой вершине.

3. Если валентность вершины не равна 1 и вершины не смежные. (Самый интересный случай.)

Для организации хранения данных нам потребуется:

```
Program Menger;
Const maxn =15;
Type matrix = array [1..maxn,1..maxn] of integer;
     Boolmas = array [1..maxn] of boolean;
     mas = array [1..maxn] of integer;
var N: integer;     {количество вершин в графе}
   A: matrix;     {матрица смежности}
   Val_V: mas;  {массив для запоминания
                              валентности вершин}
```

Также нам потребуется логический массив *B* для запоминания «посещаемости вершин»: были мы в этой вершине или нет. И потребуется массив *Num* для хранения и выведения вершин, входящих в <u,v>-цепь. Но их мы опишем в основной процедуре ***Solve***.

Напишем костяк процедуры Solve:

```
Procedure Solve;
var B: boolmas;
   Num: mas;
   u,v: integer;

begin
  for u:=1 to N-1 do
    for v:=u+1 to N do begin
      write (u,' - ',v);
      if A[u,v]=1 then writeln (" смежные ")
                                   {1-й случай}
      else if (Val_V[u]=1) or (Val_V[v]=1) then
                writeln (" 1")  {2-й случай}
           else begin
             ...                   {3-й случай}
           end;
    end;
end;
```

Как уже было сказано, 3-й случай – самый интересный. Для его реализации будем использовать модификацию рекурсивного поиска в глубину. (Подробное рассмотрение логики поиска в глубину можно найти в [2])

```
Procedure Search_Depth (i:integer; var k:integer);
var j:integer;
begin
 Num[k]:=i;
 B[i]:=True;
 for j:=1 to N do
  if (A[i,j]=1) and (B[j]=false)
    then begin
      inc(k);
      Search_Depth(j,k);
    end;
end;
```

При написании программы следует учесть некоторые особенности.

• Во-первых, сам способ выведения <u,v>-цепей на экран. Для запоминания цепей мы должны формировать массив Num заново.

Решение: каждый раз, как мы выходим из рекурсии, уменьшаем *k* (индекс вершины в <u,v>-цепи):

Dec (k).

• Во-вторых, нам при формировании цепи не имеет смысла заходить в вершины, валентность которых равна 1. В <u,v>-цепях такие вершины являются лишними.

Решение: перед тем как пойти к какой-либо вершине, мы проверяем ее валентность. И если валентность вершины равна 1, то пропускаем ее:

If Val_V[i]<>1 then begin … end;

• В-третьих, например, идя из вершины 3 в вершину 5, нужно исключить возможность формирования вершинно-пересекающихся цепей 3-1-4-5 и 3-1-10-5.

Решение: как только была найдена нужная нам вершина, мы должны вернуться к исходной вершине и продолжить поиск, но уже по другому ребру. Для этого вводим специальную логическую переменную *pp*, которая будет отвечать за нахождение вершины.

Изначально *pp:=false*, как только нашли вершину *v*, *pp:=true*. Далее идем назад, пока не встретим снова вершину *u*. Нашли ее, *pp:=false*, и снова продолжаем поиск и формирование <u,v>-цепи.

После внесенных изменений процедура Solve будет выглядеть так:

```
Procedure Solve;
var B:boolmas;
    Num:mas;
   u,v,t,ways:integer;
    pp:boolean;

    Procedure Search (i:integer; var k:integer);
     var j:integer;
     begin
       Num[k]:=i;
       B[i]:=True;
        j:=0;
       if A[i,v]=1 then begin
                          pp:=true;
                        Print(num,k);
                         inc(ways);
                      FillChar (Num,Sizeof(Num),0);
                          end
        else
       while (j<N) and not pp do begin
         inc(j);
       if (A[i,j]=1) and (B[j]=false) then
          if Val_V[i]<>1 then begin
                 inc(k);
                Search(j,k);
                 end;
      end;
      dec (k);
      if k=1 then pp:=false;
     end;

begin
 for u:=1 to N-1 do
  for v:=u+1 to N do begin
   write (u,' - ",v);
   if A[u,v]=1 then writeln (" смежные")
   else if (Val_V[u]=1) or (Val_V[v]=1) then writeln
                                          (" 1")
          else begin
              pp:=false;   t:=1;   ways:=0;
            FillChar (B,Sizeof(B),0);
             Search (u,t);
             writeln ("ways:  ",ways);
              end;
   end;
end;
```

Временная оценка работы программы

Общее количество вызовов рекурсивной процедуры Search: C = N-1+N-2+…+1 = N*(N-1)/2. Стандартная рекурсивная процедура поиска в глубину (Search_Depth) имеет временную оценку порядка N^2; мы имеем N вершин и для каждой вершины – N вариантов путей поиска, таким образом, C = N*N. Наша измененная процедура Search имеет такую же временную оценку, однако в большинстве случаев временная сложность процедуры равна N, потому что мы не рассматриваем смежные вершины и вершины, валентность которых равна 1.

Посчитаем временную сложность алгоритма всей процедуры Solve:

$$C = N*N*(N-1)/2 = O(N^3).$$

Для процедуры, позволяющей выводить на экран не только максимальное количество вершинно-непересекающихся <u,v>-цепей для каждой пары несмежных вершин, но и сами <u,v>-цепи, это хорошая временная характеристика.

Заключение

У каждого учащегося есть свое понимание понятия связности графа. Чаще всего это понимание строится на основе неформальных наблюдений и не всегда является четким, ясным и даже правильным. Теорема Менгера придает этим наблюдениям точный и строгий смысл. Поэтому целесообразным будет включение темы «Алгоритмическая реализация Теоремы Менгера» в раздел учебного курса по информатике.

Примечания

1. *Новиков Ф. А.* Дискретная математика для программистов. СПб.: Питер, 2001. С. 215-217.
2. *Окулов С. М.* Конспекты занятий по информатике (алгоритмы на графах). Киров, 1996. С. 10-11.
3. *Окулов С. М.* Основы программирования. Ч. 4. Киров, 2001. С. 37-40.

Р. В. Шарыгин

ОБ ОДНОЙ ЗАДАЧЕ ПОИСКА ДАННЫХ

Представление данных – это действительно суть программирования.
Ф. Брукс

Неисчерпаемость обычной задачи поиска данных приводит в изумление

В настоящее время задачи поиска встречаются во многих приложениях. Программа проверки правописания ищет слово в словаре, сервер ищет имя узла в свой базе данных, чтобы определить его IP-адрес, поиск абонента в телефонном справочнике и т. д. Можно привести множество примеров, где используется поиск в наборе данных для получения необходимой информации. Рассмотрим задачу поиска в наборе данных, а точнее, ее постановку, выбор подходящей структуры данных для ее решения и оценки эффективности работы каждого из способов решения.

ШАРЫГИН Роман Владимирович – студент III курса факультета информатики ВятГТУ

Задача

Секретная корпорация, занимающаяся поиском инопланетных жизненных форм, обнаружила на одной из планет созвездия Альфа живые организмы. Она приняла решение вести наблюдение за развитием и изменением численности организмов, с этой целью на орбиту планеты был послан спутник-наблюдатель, который мог следить за изменениями численности организмов. Недостаток этого «наблюдателя» в том, что он может отслеживать изменения только на той территории планеты, которая находится непосредственно под ним.

Таблица

Событие	Параметры	Описание
1	*X, Y*	Изменение количества организмов в сегменте с номером *X* на *Y* единиц.($-2^{15} \le Y \le 2^{15}-1 = 32767$)
2	*L, R*	Запрос суммарного количества организмов с *L* по *R* сегмент
0		Завершение работы

С этой целью его траектория была разбита на равные сегменты. Они пронумерованы от 1 до *N*. По запросу с Земли о количестве живых форм в сегментах с *L* по *R* (*L* J *R*) – спутник должен, пролетая над ними (*L*, *L+1*, …,*R-1*, *R* сегментами), произвести подсчет и затем, в ответ на запрос, отправить полученные данные. Но количество организмов постоянно изменяется: в некоторое время в *X* сегменте на *Y* единиц.

Помогите написать программу для спутника, которая будет отвечать на запросы и отслеживать количество единиц жизни в каждом сегменте.

Формат входных данных

Во входном файле первым записано число *N* (*1* J *N* J 2^{13}= *8192*). Затем записана последовательность событий (табл. 1):

Количество событий не превосходит *100000*.

Формат выходных данных

В выходной файл записывать только ответы на запросы.

Как мы видим, формулировка задачи достаточно большая, но выделим из этого описания только нужные нам моменты для решения задачи: дан одномерный массив, элементами массива могут быть только неотрицательные целые числа, элементы массива могут менять сове значение в процессе работы, требуется нахождение суммы элементов некоторого непрерывного отрезка массива. Попробуем решить поставленную перед нами задачу (и попутно ослабим ограничение на количество событий).

Решение

Простые алгоритмы

Первый вариант решения (А), который приходит в голову, достаточно прост. Назовем событие изменения значения элемента массива `UpDate(I, J)`, где `I` – индекс изменяемого элемента массива, а `J` – значение изменения. Аналогично для события запроса суммы элементов отрезка массива – `Report(I, J)`, где [I, J] отрезок массива.* Приведем листинг этих процедур:

```
Procedure UpDate(Const I, J: Integer);
 Begin
 Inc(X[I], J);
 End;

Procedure Report(Const I, J: Integer);
 Var K: Integer; Sum: LongInt;
 Begin
  Sum := 0; For K := I To J Do Inc(Sum, X[K]);
  WriteLn(Sum);
 End;
```

В зависимости от события вызывается соответствующая ему процедура. Программа получается короткой и ее легко понять. К сожалению, она никуда не годится – она медленно работает при больших размерностях задачи (в частности, при увеличении числа запросов, а тем более числа событий).

Есть очевидный способ сделать быстрее вычисление суммы отрезка.* На этот раз будем хранить в X[I] сумму отрезка [0, I]. А вычисление суммы отрезка [I, J] сведем к вычислению разности [0, J] – [0, I-1].** Наше нововведение заметно ускорило вычисление суммы отрезка, но также и замедлило изменение массива, теперь, в случае изменения I-го элемента, кроме X[I] потребуется изменить еще и все X[J], (I<J<=N). Приведем листинг соответствующих процедур:

```
Procedure UpDate(Const I, J: Integer);
 Var K: Integer;
 Begin
  For K := I To N Do Inc(X[K], J);
 End;

Procedure Report(Const I, J: Integer);
 Var Sum: LongInt;
 Begin
  Sum := X[J] – X[I – 1];
  WriteLn(Sum);
 End;
```

* Далее и в других вариантах решения события и соответствующие им процедуры будут называться аналогично.

* Конечно же, суммы элементов отрезка массива, но для простоты далее будем ограничиваться названием «сумма отрезка».

** В начале работы алгоритма массив X инициализирован нулями, таким образом, нам не требуется делать никаких дополнительных вычислений. (X[0]=0 – барьерный элемент, он используется для того, чтобы исключить написание дополнительных условий проверки при вычислении суммы отрезка).

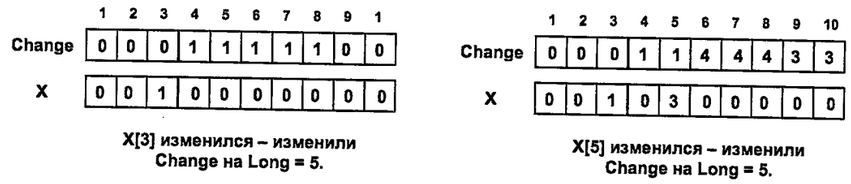


Рис. 1

У описанного выше алгоритма (B) существует очевидный недостаток (о нем говорилось выше) – обновление массива отнимает большое количество времени. К примеру, обновляем только первый элемент нашего массива, происходит также пересчет и всех остальных элементов массива. Попробуем написать алгоритм, учитывающий недостатки обоих вышеуказанных алгоритмов.

Теперь идея заключается в следующем: будем хранить изменение элемента на некоторое расстояние. Для этого нам потребуется второй массив Change той же размерности, что и X.* Длину, на которую мы будем хранить изменение, назовем Long. Допустим, происходит изменение I-го элемента массив, требуется также изменить X[I] и элементы Change[I+1], …, Change[I+Long].** Сбор суммы занимает меньшее время: нам требуется собрать суммы наших отрезков, длины Long и затем добавить элементы не входящие в отрезок. Более ясно станет, если разобрать листинг соответствующих процедур (рис. 1):

```
Procedure UpDate(Const I, J: Integer);
 Var K: Integer;
 Begin
 Inc(X[I], J);
 For K := I + 1 To Min(N, I + Long) Do
                              Inc(Change[K], J);
 End;

Procedure Report(Const I, J: Integer);
 Var Sum: LongInt;
 Begin
 Sum := 0;
 While I <= J – Long Do Begin
 Inc(I, Long); Inc(Sum, Change[I]);
 End;
 While I <= J Do Begin
  Inc(Sum, X[I]); Inc(I);
 End;
 WriteLn(Sum);
End;
```

В процедуре `Report` первым циклом собирается сумма с отрезков длины Long, а вторым циклом – элементы массива X, не вошедшие в отрезок. Таким образом, мы получаем алгоритм (C), который работает на достаточно больших размерностях.* Выбор значения для переменной Long предоставляется читателю (от выбора ее значения также зависит и время работы данного алгоритма).

Дальнейшее совершенствование алгоритмов, я думаю, не даст ощутимого увеличения производительности. Попробуем двигаться в другом направлении, а именно, изменим структуру данных.

Сложные алгоритмы

Что же следует изменить в структуре данных? Все просто, изменим само представление нашего исходного массива X (рис. 2). Теперь массив X будет хранить в себе дерево сумм исходного массива.** Если сейчас все не очень ясно, то далее вопросы исчезнут.

Рис. 2

В статье [1] подробно разобран и приведен алгоритм решение такой же задачи для двумерного случая. На рисунке можно увидеть, как массив X хранит частичные суммы исходного массива. В X[I] хранится сумма элементов из отрезка [I-2^k+1, I], где k – конечное число нулей в двоичном представлении числа I.* Например, для I=6 имеем $6_{10}=110_2$, k=1, таким образом, X[6] хранит сумму элементов исходного массива на отрезке [5, 6].

Теперь с хранением сумм все ясно.

Перейдем к процедуре изменения значения исходного массива. Так как храним мы только массив частичных сумм, то нам нужен способ обновления их, при изменении значения любого элемента исходного массива. Это сделать совсем не сложно: вслед за изменением I элемента исходного массива изменим X[I], но этот элемент содержится в узле нашего дерева, т.е. нам надо вычислить новый узел, который содержит в себе текущий измененный. Для этого мы к индексу I текущего элемента прибавляем 2^k (предварительно вычислив k для I) и т. д., пока мы находимся в пределах массива X. Например, изменяем значение 2-го элемента исходного массива. Соответственно изменим X[2], затем $2_{10}=10_2$, $2+2^1=4$, изменим X[4], далее $4_{10}=100_2$, $4+2^2=8$, обновляем X[8]. Все наше дерево частичных сумм обновлено и готово к следующему событию.

Перейдем теперь к способу получения суммы отрезка [I, J]. Операция вычисления суммы сводится к вычислению разности отрезков [1, J] – [1, I-1]. Рассмотрим получение суммы отрезка [1, I]. Как легко заметить, сумму отрезка мы можем получить, собирая частичные суммы нашего дерева. Соответственно суммируем сам элемент X[I], затем найдем ближайший к нему узел, хранящий сумму предшествующих I элементов, т.е. I-2^k новый номер узла и т. д. до первого элемента. Пусть есть отрезок [1, 5], получим сумму этого отрезка: суммируем X[5], затем вычисляем новый индекс $5_{10}=101_2$, $5-2^0=4$, прибавляем X[4], далее $4_{10}=100_2$, $4-2^2=0$ – закончили суммирование, так как 4-й элемент хранит сумму отрезка [1, 4]. Получать суммы отрезков мы научились, теперь осталось написать алгоритмы UpDate и Report.

```
Procedure UpDate(Const I, J: Integer);
 Begin
  While I <= N Do Begin
  Inc(X[I], J);
  Inc(I, Dispose(I));
  End;
 End;

Procedure Report(Const I, J: Integer);
 Var Sum: LongInt;
 Begin
  Sum := 0;
  While J >= 1 Do Begin
  Inc(Sum, X[J]);
  Dec(J, Dispose(J));
  End;
  Dec(I);
  While I >= 1 Do Begin
  Dec(Sum, X[I]);
  Dec(I, Dispose(I));
  End;
  WriteLn(Sum);
 End;
```

Здесь использована процедура `Dispose(I)`, которая возвращает значение 2^k для заданного числа I. Ее листинг, полагаю, приводить не надо, вы сами без труда сможете ее воспроизвести. Следует также указать, что для работы с этой структурой данных нам требуется массив X, такой, что его размер есть число 2 в какой-то степени, превосходящее либо равное размеру исходного массива. Подобрать его можно простым циклом.

```
While N > Nm Do Nm := Nm Shl 1;
N := Nm;
```

N – размер исходного массива, Nm – переменная для подбора подходящего размера массива X. Затем заменяем значение N на найденное Nm (в алгоритмах использовано единое представление длины массива X переменной N). Приведенный алгоритм (D) работает на больших размерностях, чем все ранее использованные алгоритмы.*

Оценки

Для простых алгоритмов, особенно для (A) и (B), оценка времени работы пропорциональна O(N) для блока обновления или запроса. Это нас не устраивает, так как для достаточно больших N мы просто не дождемся, когда алгоритм возвратит конечный результат, чего не скажешь об алгоритме (D). Оценка его времени работы $O(\log_2 N)$, что более приемлемо для нашей задачи.

Алгоритм	Время работы (сек).
A	*6~7*
B	*7~8*
C (для `Long` = *5, 25, 50, 100*)	*3~4, 1~2, 1*, менее *1*
D	Менее *1*

Тестирование выполнено для N = *8192*, количество событий *100000*, *55055* запросов (на P III 668 Mhz). Однако этими алгоритмами еще не заканчивается решение данной задачи. Существует множество решений, но решений с меньшей верхней оценкой я не нашел.

В заключение скажем, что приведенным здесь примером мы подтвердили, что сложные алгоритмы зачастую дают огромное увеличение производительности.

Примечания

1. *Андреева Е. В.* Решение задач XIII международной олимпиады // Информатика. 2001. № 37.

2. *Бентли Д.* Жемчужины программирования. 2-е изд. СПб.: Питер, 2002.

* Введение фиктивного элемента с индексом 0 не требуется

** Если таковые существуют

* Имеется в виду количество событий в условии задачи: запросов и изменений

** Название *дерево* вовсе не означает бинарное дерево, просто эта структура хранения сумм очень похожа на деревья

* Далее k используется в том же значении для конкретных чисел.

* Время его работы незначительно даже для количества событий, указанных в условии задачи.

Вятский государственный гуманитарный университет

Факультет информатики

Факультет информатики стал самостоятельным подразделением Вятского государственного гуманитарного университета в 1999 году.

Декан факультета – кандидат технических наук, доцент кафедры информатики и методики обучения информатике Окулов Станислав Михайлович.

Зам. декана по учебно-воспитательной работе – кандидат педагогических наук, доцент кафедры прикладной математики Бушмелева Наталья Александровна.

Зам. декана по заочному отделению факультета – кандидат педагогических наук, доцент кафедры информатики и методики обучения информатике Васенина Елена Александровна.

Обучение ведется по трем специальностям

«030100 – Информатика с дополнительной специальностью иностранный язык» с квалификацией специалиста – учитель информатики и английского языка.

Основные направления обучения

1. Развитие мыслительных способностей в процессе столь специфической деятельности, как программирование, включающее в себя профессиональное владение основными технологиями программирования – процедурное, объектно-ориентированное, визуальное программирование (Pascal, Delphy, Visual Basic и др.).
2. Развитие способности к быстрому самостоятельному освоению новых информационных технологий и программных средств, профессиональное владение основными информационными технологиями общего назначения (обработка текстов и электронных таблиц, системы компьютерной графики и многое другое).
3. Свободное владение английским языком, в частности соединение профессиональных знаний в области информатики с языковой подготовкой, что дает качественно новый уровень подготовки специалиста.
4. Владение основами педагогики, психологии, теории познания, дидактики и методики.

«010200 – Прикладная математика и информатика» с квалификацией специалиста – математик, системный программист.

Основные направления специализации

1. Серьезная математическая подготовка. В дополнение к традиционной высшей математике изучаются курсы дискретной математики, теория алгоритмов, методы оптимизации, теория игр, исследование операций и другие.
2. Глубокая подготовка в области программирования. Акцент делается на системное программирование, методы трансляции, программное обеспечение компьютерных сетей.

Изучаются курсы:

- архитектура компьютерных сетей;
- сетевое администрирование и управление сетью;
- сетевые операционные системы;
- методы защиты информации;
- распределенные базы данных;
- глобальные сети и др.

3. Серьезная языковая подготовка специалистов в области сетевого и системного программирования, что на порядок повышает их конкурентоспособность на рынке труда.

«351400 – Прикладная информатика (в экономике)» с квалификацией специалиста – информатик-экономист.

Информатик-экономист – это специалист широкого профиля, получивший фундаментальные знания в области информатики и профессиональные знания в области экономики. Основное назначение такого специалиста – создание и внедрение профессионально-ориентированных информационных систем в экономике. Он занимается решением функциональных задач и управлением информационных, материальных и денежных потоков в конкретном секторе экономики с помощью информационных систем. Информатик-экономист анализирует, прогнозирует, моделирует и создает информационные процессы и технологии в рамках профессионально-ориентированных информационных систем.

Подготовка информатиков-экономистов ориентирована на экономическую грамотность и их возможную будущую деятельность в областях практической экономики, в областях экономической науки и реального экономического анализа.

В процессе обучения студенты изучают модели экономических систем, проектирование бизнеса, программное обеспечение банковских систем, математические и информационные технологии.

Выпускники способны:

- проектировать новые информационные системы для решения прикладных задач в экономике;
- быстро адаптироваться к структурным изменениям в рыночной экономике.

Выпускники могут работать:

- в органах государственного и местного управления (информационное обеспечение организационно-экономических механизмов управления);
- в научно-исследовательских организациях и коммерческих структурах (разработка компьютерных экономических систем для банков, холдингов, акционерных обществ);
- в академических институтах (проектирование информационных систем в экономике и управлении).

Поскольку две последние специальности не являются педагогическими, здесь нет специальной подготовки в области педагогики и методики. Однако по желанию студент может выбрать для изучения курсы по психологии, социологии, экономике, государственному праву, что также расширяет сферу его профессиональной подготовленности.

Важно и то, что на факультете информатики готовят всесторонне образованных людей, которые знают мировую историю и культуру, философию, концепции современного естествознания.

Сегодня на факультете обучается 405 студентов (по специальностям соответственно 220, 97, 51), из них 47 – заочно.

В настоящее время в рамках факультета существует ***4 кафедры:***

- *кафедра прикладной математики:* заведующий – канд. физ.-мат. н., доцент В. А. Онегов;
- *кафедра информатики и методики обучения информатике (ИМОИ):* заведующая – канд. пед. н., доцент кафедры И. А. Бабушкина;
- *кафедра информационных технологий:* заведующая – канд. пед. н., доцент М. В. Петухова;
- *кафедра германских языков: заведующая* – канд. филол. н., доцент Е. Н. Колодкина.

На факультете работает 41 преподаватель. Средний возраст преподавателей – 27 лет.

Научные направления. На факультете открыта ***аспирантура*** по двум специальностям:

- специальность **«130002 – Теория и методика обучения и воспитания (информатика)»**. На сегодняшний день аспирантуру закончили и защитили кандидатские диссертации 3 человека, продолжают обучение 4 человека.

Руководит работой аспирантов доктор педагогических наук, профессор, академик-секретарь Отделения общего среднего образования РАО ***Александр Андреевич Кузнецов***, известный в России и за рубежом специалист в области методики преподавания информатики в школе и в вузе. Александр Андреевич Кузнецов возглавляет научную школу по методике обучения информатике, еще в конце 60-х годов он сформулировал и обосновал пути развития образовательной информатики в школе. Решил ряд ключевых проблем преподавания информатики, в частности содержания обучения информатике, образовательного стандарта по этой дисциплине, методической подготовки учителя информатике, проверки и оценки учебных достижений школьников и др. Он является автором более 250 научных работ, в том числе более 10 монографий и учебных пособий. Под руководством А. А. Кузнецова 6 человек защитили докторские диссертации, 27 человек – кандидатские диссертации.

- специальность **«100219 – Теория языка»**

Руководитель – доктор филологических наук, профессор, заслуженный деятель науки Российской Федерации ***Александра Александровна Залевская*** – основатель и руководитель Тверской психолингвистической школы – особого направления в отечественной психолингвистике, автор многочисленных статей и монографий, большинство из которых являются бестселлерами в научных кругах. 10 докторантов и 35 аспирантов защитили свои диссертации под ее научным руководством.

Тематика научных исследований аспирантов и соискателей охватывает широкий круг проблем психолингвистики: становление и функционирование слова в ментальном лексиконе индивида с учетом разнообразных форм репрезентации и организации как языковых, так и энциклопедических знаний, определение стратегий, используемых при понимании слова; изучение процессов и механизмов продуцирования речи и понимания текста на родном и иностранном языках; определение опор, используемых при принятии переводческих решений.

Научно-исследовательская работа: научная деятельность преподавателей и студентов направлена на решение следующих основных задач: развитие фундаментальных научных исследований, повышение квалификации преподавательских кадров.

На факультете сложился ряд перспективных направлений научных исследований:

- методология информатики (на примере отдельных дисциплин информационного цикла);
- психолого-педагогические вопросы методики преподавания информатики в школе и в вузе;

- внедрение в учебный процесс новых информационных технологий, их использование в преподавании гуманитарных и социально-экономических дисциплин;
- структурно-семантический аспект германских языков;
- психолингвистические проблемы понимания слова и текста;
- теория средних величин и др.

На кафедре прикладной математики обучения информатике регулярно работает научно-методический семинар (рук. доцент В. А. Онегов), на котором рассматриваются проблемы, связанные с взаимодействием предметов информационного цикла в связи с введением новых образовательных стандартов.

Работа в помощь школе и органам образования проводится по следующим направлениям: проведение педагогических практик в школах и УПК г. Кирова и области; написание статей и учебных пособий по актуальным вопросам информатики и др.

Преподаватели факультета читают лекции для учителей школ и преподавателей других вузов, проводят семинары, выступают с докладами на областных и всероссийских конференциях.

Лучшие работы студентов представляются на университетские конкурсы. Ежегодно на факультете проводится декада студенческой науки, в рамках которой организуются олимпиады по информатике. Ежегодно студенты факультета занимают призовые места на республиканских олимпиадах по информатике.

Работа со школьниками. Студенты и преподаватели занимаются со школьниками в кружках, в профильных классах, на семинарах и спецкурсах. Для учащихся ежегодно организуются областные олимпиады. Школьники имеют также возможность обучаться в школе программирования «Аналитик», созданной при факультете.

Внеучебные занятия. В организации студенческой жизни во многом помогают традиции: учебно-методический сбор «Первокурсник», КВН между студентами I и II курсов, проводимый к Дню учителя, факультетский День здоровья, «Студенческая весна» с обязательной подготовкой в течение всего года в танцевальном коллективе, вокальном ансамбле; и др.

Вступительные экзамены

Прием документов: 15.06.2004 – 15.07.2004 (г. Киров, ул. Ленина, д. 111, каб. 319)
Экзамены – 16.07.2004 – 31.07.2004

Специальность «Информатика и английский язык»
- Информатика (письменно)
- Иностранный язык (устно)
- Русский язык и литература (сочинение)

Специальность «Прикладная математика и информатика»
- Математика (письменно)
- Информатика (письменно)
- Русский язык и литература (сочинение)

Специальность «Прикладная информатика (в экономике)»
- Информатика (письменно)
- Математика (письменно)
- Русский язык и литература (сочинение)

Зачисление – 1.08.2004

На факультете осуществляется целевой внеконкурсный прием по направлениям департамента образования.

На период занятий на подготовительных курсах и экзаменов абитуриенты обеспечиваются общежитием.

Справки

по телефону приемной комиссии – 67-87-78
по телефону деканата факультета информатики – 67-06-11

Окулов Станислав Михайлович
Основы программирования
М.: ЮНИМЕДИАСТАЙЛ, 2002. – 424 с.: ил.
ISBN 5-94774-003-6

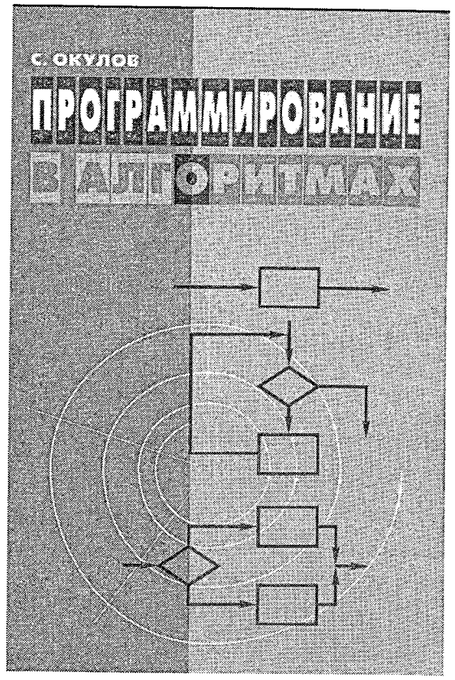
Серия «Технический университет». Учебное издание.

В учебнике рассмотрены основные управляющие конструкции системы программирования Турбо Паскаль, процедуры и функции, строковый, вещественный и файловый типы данных. Приводится материал для изучения массивов, методов сортировки и поиска, а также по динамическим структурам данных. Рассмотрены следующие структуры данных: списки, стеки, очереди, двоичные деревья, АВЛ-деревья и Б-деревья. В материалах для чтения обсуждаются практически все вопросы, входящие в школьный минимум знаний по информатике.

Книга является достаточно полным учебником по программированию, реализующим сложную задачу – формирование у читателя структурного стиля мышления. Учебным материалом является система программирования Турбо Паскаль, а также большое число задач, включая задачи на алгоритмы сортировки и поиска.

Достаточно подробно рассмотрена работа с динамическими структурами данных.

Книга рассчитана на широкий круг читателей от школьника и студента до специалиста, решающего с помощью программирования прикладные задачи.



Окулов Станислав Михайлович
Программирование в алгоритмах
М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.: ил.
ISBN 5-94774-010-9

Искусство программирования представлено в виде учебного курса, раскрывающего секреты наиболее популярных алгоритмов. Освещены , такие вопросы, как комбинаторные алгоритмы, перебор, алгоритмы на графах, алгоритмы вычислительной геометрии. Приводятся избранные олимпиадные задачи по программированию с указаниями к решению. Практические рекомендации по тестированию программ являются необходимым дополнением курса.

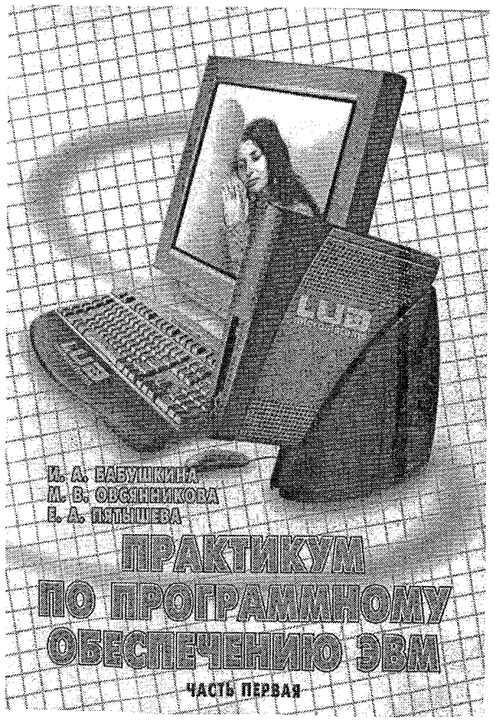
Предназначен для школьников, студентов и специалистов, серьезно изучающих программирование, а также для преподавателей учебных заведений.

Онегов Владислав Алексеевич

Исследование операций: задачи, методы, алгоритмы

Учебное пособие
Киров: Изд-во ВГПУ, 2001. – 224 с.
ISBN 5-93825-004-8

В учебном пособии представлены традиционные разделы науки Исследование операций. Основной упор сделан на методы и алгоритмы решения разнообразных задач. Все алгоритмы доведены до отлаженных Pascal-программ. Материал излагается таким образом, что возможна работа с ним в самостоятельном режиме. Пособие предназначено студентам, специализирующимся по информатике и математическим методам в экономике и всем желающим познакомиться с идеями и задачами, решаемыми методами исследования операций. Пособие рекомендовано УМО Министерства образования РФ по специальностям педагогического образования в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности 030100 – информатика.



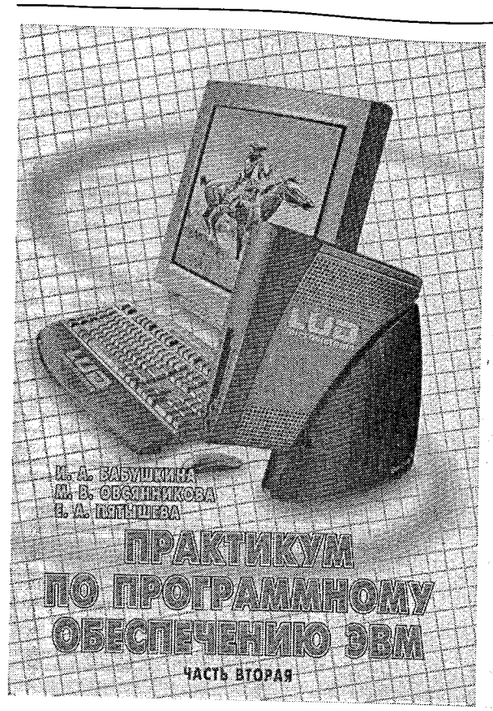
Бабушкина Ирина Анатольевна
Овсянникова Марина Владимировна
Пятышева Елена Анатольевна

Практикум по программному обеспечению ЭВМ

Часть первая
Киров: Изд-во ВГПУ, 2001. – 144 с.
ISBN 5-85271-105-5

Пособие содержит материалы занятий по курсу «Программное обеспечение ЭВМ». В первой части изложены основные сведения об операционной системе Windows NT (главы 1,2), антивирусных программных средствах (глава 3), архивировании файлов (глава 4). Главы с пятой по тринадцатую посвящены основам работы в текстовом процессоре Word 2000: настройке текстового процессора, набору и форматированию текста, подготовке документа к печати.

Учебное пособие написано по материалам практических занятий со студентами педагогического университета и со школьниками.



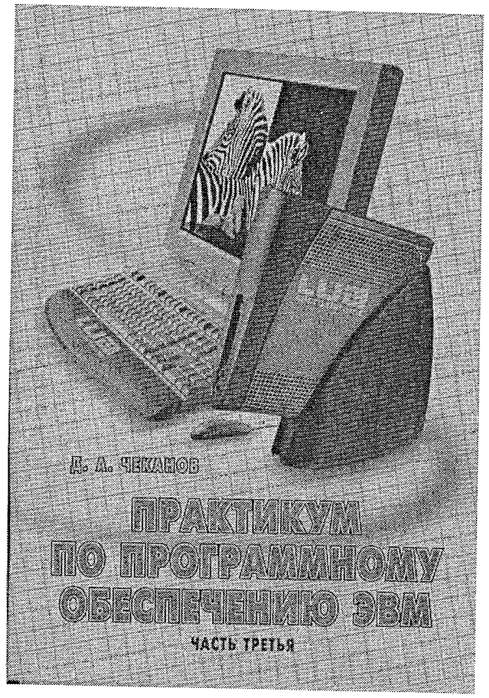
Бабушкина Ирина Анатольевна
Овсянникова Марина Владимировна
Пятышева Елена Анатольевна

Практикум по программному обеспечению ЭВМ

Часть вторая
Киров: Изд-во ВГПУ, 2001. – 144 с.
ISBN 5-85271-105-5

В книге приводятся материалы для занятий по информатике (курс «Программное обеспечение ЭВМ»). Во второй части содержатся главы, в которых рассматриваются дополнительные возможности текстового процессора Word 2000: графические объекты (главы 1-2), таблицы (глава 3), диаграммы (глава 5), формулы (глава 6), сноски (глава 9), средства автоматизации работы с документами: автотекст и автозамена (глава 4), шаблоны (глава 7), типовые письма (глава 8).

Учебное пособие написано по материалам практических занятий со студентами педагогического университета и со школьниками.



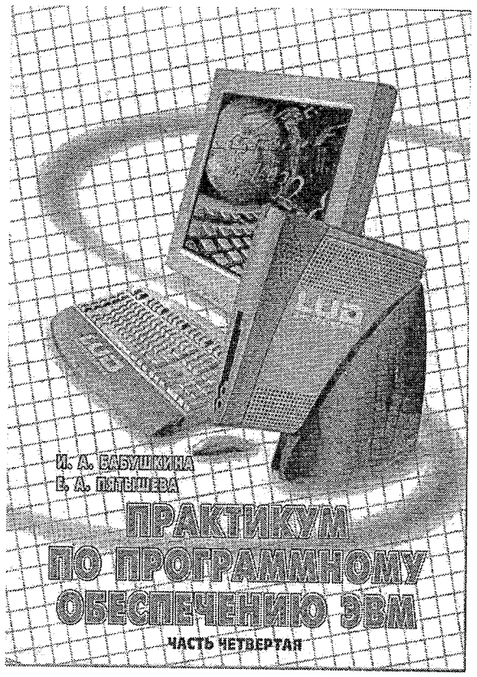
Чеканов Дмитрий Александрович

Практикум по программному обеспечению ЭВМ

Часть третья
Киров: Изд-во ВГПУ, 2001. – 96 с.
ISBN 5-85271-102-0

Пособие является продолжением первой и второй частей «Практикума по программному обеспечению ЭВМ» И.А. Бабушкиной, М.В. Овсянниковой и Е.А. Пятышевой. Содержит материалы для занятий по курсу «Программное обеспечение ЭВМ». В первой главе изложены основные сведения о сети Интернет и навигации в ней. Вторая глава посвящена поисковым серверам. Третья и четвертая главы знакомят с электронной почтой и конференциями Интернета. В пятой главе рассматривается работа с серверами FTP. Шестая глава посвящена общению в реальном времени. В седьмой главе идет речь об Интернет-пейджере ICQ.

Учебное пособие написано по материалам практических занятий со студентами педагогического университета и со школьниками.



Бабушкина Ирина Анатольевна
Пятышева Елена Анатольевна

Практикум по программному обеспечению ЭВМ

Часть четвертая.
Киров: Изд-во ВГГУ, 2003. – 240 с.
ISBN 5-85271-081-4

Учебное пособие предназначено для изучения электронных таблиц на примере Excel 2000 в рамках таких курсов, как программное обеспечение ЭВМ (ПО ЭВМ) и информационные технологии в образовании (ИТО).

Основные темы, включенные в пособие: форматирование таблиц, расчет по формулам, массивы, формирование связанных книг, построение диаграмм, использование таблиц в качестве базы данных, создание сводных таблиц, подбор параметров и поиск решения с помощью сценария.

Пособие позволяет изучить материал, непосредственно выполняя упражнения за компьютером. Для самостоятельного выполнения предлагаются задания.

Книга предназначена для студентов первых курсов высших учебных заведений, а также учащихся и учителей школ с углубленным изучением информатики.

Васенина Елена Александровна

Информатика для абитуриентов

Учебное пособие
Киров: Изд-во ВГПУ, 2002. – 168 с.
ISBN 5-900185-72-9

Учебное пособие предназначено для учащихся и учителей информатики средней общеобразовательной школы и содержит теоретический материал для подготовки к вступительным экзаменам по информатике в вуз. Книга будет полезна также студентам вузов различных профилей, начинающим изучать информатику.