А.И. Бочкин

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ

Курс лекций для специальности «Математика и информатика»

2003

Список сокращений

АРТ	– ассоциативно-рефлекторная теория
АСУ	– автоматизированная система управления
ВТ	– вычислительная техника
ДС	– диалоговые среды
ИнфО	– журнал «Информатика и образование»
ИП	– информационный поиск
МК	– микрокалькулятор
ППЭВМ	– профессиональная ПЭВМ
ПО	– программированное обучение
ПОД	– поисковый образ документа
ПОЗ	– поисковый образ запроса
ППС	– педагогические ПС
ПС	– программные средства
ПФУД	– поэтапное формирование умственных действий
ПЭВМ	– персональная ЭВМ
РМП	– рабочее место преподавателя
РМУ	– рабочее место ученика
СЛС	– структурно- логическая схема
УП	– учебный процесс
ЭС	– экспертная система

В В Е Д Е Н И Е

Цели обучения. Курс «Компьютерные технологии в образовании» читается студентам выпускных курсов специальности «Математика и информатика».

При чтении курса преследуются следующие цели образования студента:

– расширить знания студента о возможностях применения ВТ в связи с учебным процессом;

– научить использованию этих возможностей, взятых в готовом виде;

– научить разработке педагогических программных средств (ППС), реализующих эти возможности.

Цели развития:

– сформировать способность к анализу и оценке учебного процесса с точки зрения возможностей его компьютеризации;

– сформировать приемы переноса компьютерных знаний и умений в конкретные учебные задачи и, наоборот, выделение из задач компоненты, поддающейся формализации и, следовательно, компьютеризации;

– научить видеть и находить новые, непредусмотренные ранее способы использования ВТ в УП;

– развить способность к системному мышлению на основании осознания или поиска внутри- и межпредметных связей информатики и других школьных дисциплин.

Цели воспитания:

– сформировать уверенность в своих силах как педагога на основе овладения ППС качестве пользователя и разработчика;

– развить потребность в творческой деятельности при компьютеризации новых учебных задач и предметов;

– развить потребность в рефлексии, критическом переосмыслении своей деятельности, знаний и умений в условиях непрерывного продвижения школы к компьютерным технологиям обучения.

Задачи дисциплины. В результате изучения курса «Использование ВТ в учебном процессе» будущий учитель должен:

– стать квалифицированным пользователем инструментальных и педагогических программных средств (ПС):

– видеть возможности компьютеризации конкретных дисциплин, управления образованием и конкретных видов своей и ученической деятельности;

– уметь формулировать педагогические требования к ППС и на основании этого правильно их отбирать;

– уметь поставить задачу программисту на разработку ППС в виде сценария, педагогически обосновано и технически реально;

– уметь самому разрабатывать несложные ППС по готовому сценарию;

– уметь разрабатывать методическую документацию на готовые ППС;

– быть готовым к взаимодействию с преподавателями других дисциплин (физики, химии и т.д.) с целью повышения качества учебного процесса за счет его компьютеризации;

– взаимодействовать с администрацией школы с целью повышения качества управления образованием через его компьютеризацию;

– знать состояние и перспективы компьютеризации образования.

По мере накопления школами ППС акценты курса будут смещаться с разработки ППС на их адаптацию и применение.

Связь курса с другими дисциплинами. В методике преподавания информатики компьютер и программные средства (ПС) выступают как объект изучения; в данном курсе – как средство. Таким образом, эти курсы взаимно дополняют друг друга. Пользовательский курс информатики ближе к данному курсу в освоении универсальных видов деятельности.

В предлагаемом курсе привлекаются понятия общей дидактики при формировании заданий на разработку или оценке готовых ППС.

Привлекаются сведения из методик преподавания конкретных дисциплин: тех, которые компьютеризуются, особенно по второй специальности студента (математика или физика).

Знания по программированию как части курса информатики нужны для разработки несложных ППС и общего понимания работы отбираемых ППС, для разработки реализуемых сценариев ППС.

Курс «Численные методы» предлагает математический аппарат для построения ППС типа учебных моделей для других дисциплин. Кроме того, могут ставиться задачи на компьютеризацию изучения самих численных методов.

Программные средства. В курсе приведены тексты открытых программ-примеров по основным разделам курса. Эти программы иллюстрируют устройство соответствующего класса программ и предназначены для модификации студентом или учеником. Основной язык – Бейсик. Он выбран с учетом того, что приводимые примеры должны быть обозримы, невелики, а Бейсик вообще весьма лаконичен. Для исполнения программ из книги заведомо подходят Бейсик–среды типа Turbo, Power, Visual. Автор по возможности избегал нумерации строк как архаизма, отвлекающего от сути алгоритмов. Но во всех примерах использовано инвариантное ядро Бейсиков, так что программы можно выполнить и в старых средах типа GW, MSX или даже Вильнюс-Бейсик, занумеровав строки.

ЧАСТЬ I. ПЕДАГОГИЧЕСКИЕ ПРОГРАММНЫЕ СРЕДСТВА

Г Л А В А 1. ВВЕДЕНИЕ В ППС

1.1. Понятие ППС

Среди программных средств (ПС) можно выделить класс педагогических ПС (ППС), как программ, обеспечивающих применение компьютера как средства обучения или учебной деятельности. Это определение несколько расширяет класс ППС, к которому теперь можно отнести не только ПС, предназначенные непосредственно для учебного процесса, но и ПС, которые обеспечивают эффективность занятий по информатике и другим дисциплинам с использованием ЭВМ. Так, обычный текстовый редактор при использовании на уроках родного языка становится уже не только универсальным, но и педагогическим ПС. «Чистое» ППС не может использоваться вне учебного процесса. Это, например, программа контроля знаний.

Итак, ППС – это средство деятельности учащегося или учителя.

1.2. Технологии и творчество в учебном процессе

Особенность информатики как учебного предмета и области интеллектуальной деятельности состоит в том, что она пронизана алгоритмами в виде готовых программ и приемов работы пользователя с готовыми ПС. Важные свойства алгоритмов вообще – результативность и массовость. Это значит, что после того, как алгоритм реализован на ЭВМ, при известных требованиях к памяти и быстродействию ЭВМ он дает массовые и гарантированные результаты, которые можно понимать как функцию от исходных данных, а сам алгоритм – как воплощение этой функции. В этом смысле использование готовых алгоритмов – это и есть новые информационные технологии. Ведь признак всякой технологии – массовость и воспроизводимость изделий, алгоритмичность, автоматизация процесса производства, независимость результата от личности работника. В этом смысле алгоритм противостоит творчеству так же, как цех с конвейером – мастерской художника или изобретателя.

Оптимальный результат взаимодействия творческого мыслительного процесса и его алгоритмического воплощения – это переход завтра к массовому, промышленному производству и использованию того, что создано в единичном экземпляре человеком-творцом сегодня. История техники, а теперь и информатики полна таких примеров. Так, собранный, согласно легенде, в гараже, первый компьютер APPLE вскоре стал массовой ПЭВМ в американских школах; результат творческого озарения Н. Вирта, язык Паскаль стал массовым учебным, а затем и промышленным языком. В ин-

форматике, как нигде в других областях человеческой деятельности, является коротким путь от акта творчества к массовому производству и использованию результатов творчества.

Отсюда ясно, насколько важно создать, образно говоря, технологию творчества в учебном процессе.

1.3. Классификация ППС

Назначение классификации. Назначение предлагаемой классификации – дать основания для правильной оценки эффективности применения конкретных ППС в УП.

Наиболее важное основание для классификации – вид деятельности человека, применяющего ППС. Можно различать, например, игру, учебу и производительный труд и, соответственно, ПС для этих целей. В свою очередь, в учебной деятельности можно различить предъявление и усвоение информации, тренировку навыком и контроль.

Универсальными видами деятельности являются чтение, письмо, счет, рисование и черчение, музицирование, управление исполнителями, поиск информации.

Итак, вид деятельности человека за ЭВМ – наиболее естественное и важное основание для классификации ППС. Содержание данного пособия и построено по видам деятельности. Рассмотрим и другие основания: по техническим, программным и педагогическим характеристикам ПС.

Технические характеристики. Технические характеристики ППС нередко имеют большое значение и с точки зрения ведения учебного процесса. Рассмотрим наиболее важные из них.

1) Объем необходимой памяти. Эта характеристика определяет потребную мощность ЭВМ.

2) Зависимость от внешних запоминающих устройств. Наличие такой зависимости программы ограничивает применение ПС в УП в случае, если несколько ученических ПЭВМ обращаются к одному тому же дисководу. При отсутствии винчестера постоянная работа с дискетой снижает надежность функционирования программной среды.

3) Время счета для пакетных и время отклика для диалоговых программ. Время ответа – не только техническая, но психологическая характеристика ПС. Известно, что ожидание реакции программы в диалоге с ней свыше 3-4 сек. утомляет и раздражает пользователя.

Программистские характеристики. В этой группе также есть параметры, важные, прежде всего, для ведения учебного процесса.

1. Режим работы (диалоговый или пакетный). Для учебных целей предпочтителен диалоговый режим. Но и профессиональный пользователь не любит ждать у терминала и может запустить, например, программу печати в фоновом режиме.

2. Язык реализации программы. С точки зрения конечного пользователя – это как будто неважно. Но не будем спешить с выводами. Если программа написана на Бейсике, то ввод, скорее всего, выполняется через запятую, на Паскале – через пробел. Перехватить ошибки времени исполнения проще из Бейсика. Пример показывает, что черты языка, на котором разработана программа, отражаются и на конечном продукте, этой программе и на диалоге с пользователем.

3. Язык управления программой. Эта характеристика важна, по крайней мере, для оценки, нужно ли предварительно учиться ведению диалога с программой или нет. Так, WINDOWS-интерфейс делает малополезным чтение книг, предназначенных для «чайников», которые и стоят к тому же примерно как небольшой компьютер. Проще сразу сесть за терминал и попробовать.

4. Устойчивость к ошибкам пользователя. Здесь возможны три ситуации: устойчивость полная, частичная, и ее отсутствие. Важность этого параметра очевидна. Нормально защищенная от ошибок программа должна корректно работать при любых действиях пользователя.

5. Наличие документации или помощи. Если не учитывать экзотическую задачу исследования управления программой вслепую, без описания, отсутствие документации или помощи при нетривиальном управлении делает программу практически бесполезной. Но не так редко приобретение программы путем «копирования» у коллег как раз и приводит к ситуации отсутствия описания.

6. Авторское сопровождение программы. Наличие такого сопровождения – наиболее удобный для пользователя и хлопотный для автора вариант. По мере тиражирования ПС эта возможность становится все более редкой. Впрочем, развертывание компьютерных сетей увеличивает и возможности для диалога с автором.

7. Способ приобретения ПС: куплено, выменяно, «скопировано». Здесь учитель может показать пример корректного отношения к чужим программам. Даже если ПС «скопировано», желательно помнить о его авторе и выполнять его пожелания, если таковые имеются.

8. Номер версии. Практика показывает, что наиболее удачны, так сказать, «вторые» или четные версии программ. В первых версиях нередки ошибки, а в третьих обычно сильно расширены возможности и при этом сделаны новые ошибки.

9. Время создания. В общем, в информатике не так много долгожителей среди программ. Поэтому время создания – критерий современности ПС.

Педагогические характеристики ПС

1. Дидактическая функция. Программы можно различать по дидактической функции: предъявление информации, тренировка навыков и выработка умений, контроль, свободная творческая деятельность.

2. Тип обратной связи. Обратная связь может быть немедленная и отсроченная. В качестве примера упомянем одну обучающую программу, которая на любые ответы обучаемого реагировала так: «Ответ принят!». А в конце диалога сообщалось, что обучаемый мало что понимает в материале, и его оценка в баллах смехотворна низка. Так что, по крайне мере, восклицательный знак в приведенной реакции программы явно неуместен.

3. Активность в диалоге. Следует различать ситуации, где активнее ученик, а где программа. Так, в работе с текстовым редактором инициатива принадлежит человеку. Классические обучающие программы сковывали инициативу обучаемого и навязывали свой диалог.

4. Адаптивность программы. Это важнейшая характеристика, позволяющая судить, насколько может быть индивидуальна деятельность ученика за ЭВМ. Персональность ЭВМ без проблем обеспечивает лишь индивидуальность темпа деятельности. Но в принципе возможна и более глубокая адаптация, например, по сложности заданий.

5. Язык сообщений – родной или английский. Английский язык справедливо считается компьютерной латынью. Практически же рост возможностей программных средств сопровождается массированным расширением околокомпьютерной англоязычной лексики, подключением все больших пластов живого английского языка. Поэтому призрачна надежда на освоение компьютерной базовой лексики, скажем, из 400 слов – она непрерывно расширяется. (Слово «Бейсик» появилось как обозначение «мининглийского», 400 слов, языка для туземной администрации в колониях). Выходов здесь два: всерьез учить английский язык или использовать переводные версии программ. При «диком», нефирменном переводе нередко делаются ошибки в самих программах.

Итак, наличие помощи или документации на родном языке – важное требование, особенно в таких богатых возможностями программ, как, например, WINDOWS.

6. Диагностика ошибок пользователя. Эта характеристика актуальна особенно для программ с языковым интерфейсом, например, для сред, допускающих программирование. Пример «полудиагностики» – сообщение Бейсика типа «Syntax error in line 120». Ведь строка 120 может содержать 255 символов и среди них разнообразные ошибки.

7. Тип программы – открытая или закрытая. Понятие открытой программы (ОП) введено в [5]. Поэтому ограничимся указанием на то, что ОП предназначена не только для использования, но и для изучения и модификации обучаемым. Лучший способ понять какую-либо программу – это выяснить, как она работает, а для этого полезно разработать простейший ее аналог.

8. Зависимость деятельности на рабочем месте ученика (РМУ) от рабочего места преподавателя (РМП). Этой зависимости лучше избегать, так как возможны проблемы с одновременным взаимодействием нескольких

РМУ с РМП, снижение «живучести» компьютерного класса. Исключением является ситуация, когда именно взаимодействие учащихся в компьютерной сети необходимо для решения задачи, задача именно так и поставлена.

Таковы основания для классификации ППС. Из важнейшего основания – дидактической функции следует такая классификация ППС: обучающие программы (предъявление материала), программы – тренажеры (выработка знаний и умений), контролирующие программы и, наконец, диалоговые учебные среды (свободная творческая деятельность).

Г Л А В А 2. ЭВОЛЮЦИЯ ППС

2.1. Программированное обучение. Начало

Попытки применить компьютер в обучении были предприняты еще в 60-е годы и восходят к так называемому программированному обучению (ПО) [52]. Суть концепции в следующем. Информация, подлежащая усвоению, разбивалась на малые фрагменты, по возможности простые и обязательно логически завершенные. В конце каждого фрагмента предлагались вопросы и задания, как правило, несложные. Ответы никак не обрабатывались, и в предположении о прилежности ученика ему выдавалось словесное поощрение независимо от результатов его деятельности (даже если он и не пытался отвечать на вопросы). Фактически учащийся выполнял при движении по материалу линейный алгоритм с шагами: информация, размышление и, возможно, самоконтроль.

Очевидная слабость такого подхода в части контроля вызвала следующий шаг в развитии ПО – зависимость деятельности, предлагаемой учащемуся от ответа на контрольные вопросы. Самое простое решение – предложить ученику выбрать один из предложенных готовых ответов как правильный. В зависимости от выбранного ответа учащемуся предлагалось перейти в заданное место обучающего текста либо выполнить дополнительные задания. Соответствующий алгоритм обучения был назван ветвящимся. Отметим, что в терминах, принятых в современном программировании, это также и циклический алгоритм, поскольку возможен переход по тексту назад. Смысл слова «ветвящийся» здесь состоит в том, что траектория движения по материалу разделялась на несколько. Кроме того, при переходе назад и при повторном контроле, очевидно, что обучаемый не будет выбирать неверный ответ, который он уже выбирал, и зацикливание невозможно.

Технически этот подход к контролю был реализован на автоматизированных средствах контроля типа «КИСИ», «Аккорд» и т.п. Учащийся просто должен был нажать кнопку, соответствующую верному ответу. Более внимательные учащиеся умели на слух различать реакции устройства при правильном и неправильном ответе...

2.2. Вторая волна программированного обучения

Другим средством реализации, наиболее полно воплотившим ветвящиеся алгоритмы обучения, стали так называемые перепутанные учебники. В таком учебнике кванты материала размещены вперемешку, и единственным средством навигации по тексту являются контрольные вопросы и указания к ним, на какую страницу перейти для каждого варианта ответа. Заметим, что внешне и по структуре такой учебник напоминает дамп памяти средней ЭВМ 60-х годов.

В перепутанных учебниках наиболее ярко отразилось основное противоречие ПО, не преодоленное и сегодня. С одной стороны, учащемуся надо задать вопросы так, чтобы ответы он не подсматривал. Для этого учебник и перепутывается. С другой стороны, поскольку никакой автоматизированной обработки ответа нет и в помине, учащийся должен все же получить комментарий к своему ответу и рекомендацию, что же делать дальше. Для этого автором учебника создается алгоритм, фактически «распутывающий» учебник. В общем, при наличии интереса и волевом усилии (а это необходимо при любой технологии обучения) учащийся может пойти вслед за автором по всем предусмотренным переходам и пройти весь курс обучения. Но нужно ли перепутывать учебник для такого учащегося? С другой стороны, учащийся без интереса к делу может свести свое обучение к бездумному перелистыванию страниц, возможно, ворча про себя о тех дурных хлопотах с интенсивным листанием взад-вперед, которые устроил ему автор перепутанной книги.

Трудности этих этапов ПО связаны в известной мере с, так сказать, идеологическим обеспечением метода, известным под названием «бихеиворизм» (от английского behaivor – поведение). Для этого течения в психологии характерен отказ от исследования психических процессов в процессе познания, упрощения их до пары «стимул – реакция». Упрощая ситуацию, можно сказать, что согласно этому подходу, если поведение человека или животного отвечает стандартам, значит человек (животное) «нормален». Если ответ верный, неважно как он получен или угадан, процесс познания не учитывается. Это – психология без психики, игнорирующая деятельность и разумность человека. Человек рассматривается кибернетически, как «черный ящик».

Рассмотренные два этапа ПО выполнялись практически без использования ЭВМ, современных им ЭВМ второго поколения. Дороговизна машинного времени, отсутствие средств диалога (дисплеев) и обработка практически только числовой информации делали такое обучение с применением ЭВМ второго поколения весьма затруднительным.

Итогом этих этапов ПО стал определенный опыт разработки учебных курсов, но также и волна разочарования педагогов в возможностях даже автоматизированного контроля при помощи технических средств

(ТСК). Главное, что было отмечено – низкий уровень познавательной деятельности учащихся. Ведь для того, чтобы выбрать правильный ответ, достаточно его мысленно противопоставить остальным, распознать, даже не воспроизводя по памяти.

Оригинальной реакцией на ПО в том виде была реакция С. Пейперта [42]. Он создал язык детского творчества – язык Лого под девизом: не компьютер учит ребенка, а ребенок учит компьютер. Подробнее об этом сказано ниже.

2.3. Третья волна программированного обучения

По мере распространения средних ЭВМ третьего поколения стало ясно, что они предоставляют значительный потенциал для реализации ПО. Большая мощность, наличие внешней памяти и особенно дисплейных классов сделали принципиально возможным не только в чистом виде реализовать уже опробованные в дидактике рассмотренные выше подходы, но и ввести определенную логическую обработку ответа, вести статистку итогов обучения и т.д. Соответствующие программные средства получили название «автоматизированные обучающие системы» (АОС). Примерами являются АСУ-ВУЗ, ДИСПАТОС и другие [41], [55]. Для них характерно следующее.

Обучаемые, обычно студенты, (в школе таких ЭВМ, конечно, не было) вели непосредственный диалог с обучающей программой. Инициатива в диалоге принадлежала, как правило, программе. ЭВМ с одним процессором обслуживала целый дисплейный класс или несколько дисплейных классов. Именно перегрузка центрального процессора привела к идее ввести дополнительный специальный процессор, работающий с классом (АОС «ДИСПАТОС»). Обмен с ЭВМ шел преимущественно текстовой, частично числовой информацией. Графики не было, цвет заменяло выделение ярким шрифтом наиболее важных сообщений ЭВМ. Достаточно низкая надежность средних ЭВМ типа ЕС (обычно до часа безотказной работы) приводила к частым перезагрузкам операционной системы, иногда с потерей информации, связанной с текущим моментом деятельности обучаемых. Это резко снижало интерес к делу, вело к расточительным потерям учебного времени и довольно дорогого машинного времени (50-100 у.е. за час).

Один центральный процессор находился в режиме постоянного ожидания прерываний от дисплеев обучаемых. При прерывании процессор копировал буфер дисплея (2 Кб), с которого пришло прерывание, в оперативную память, анализировал, составлял новый буфер и посылал его обратно на дисплей. Образно говоря, процессор работал как преподаватель во время индивидуальных консультаций, отвечая вперемешку всем на все вопросы. Это радикально отличается от работы за персональной ЭВМ, где один процессор с одним учеником занимается «персональным репетитор-

ством». Желание снизить нагрузку на процессор вело порой к дидактически неоправданному решению: максимально заполнять буфера экранов, с тем, чтобы обучаемый, получив заполненный экран, «заглох» хотя бы минут на 5 и не «тормошил» процессор.

Разработчиками обучающих программ достаточно рано было понято, что можно и нужно добиваться универсальности АОС за счет выделения функций, не связанных напрямую с изучаемым материалом. Были выделены логические блоки: учебная информация в виде кадров и файлов из них, алгоритм обучения – порядок предъявления информации, сервисная часть, например, сбор статистики и т.д. Высокая программная сложность систем вынуждала всерьез заниматься их проектированием, учитывать результаты педагогики и психологии и не программировать всякий раз с нуля. Была понята необходимость создания специальных языков для автора курса и обучаемого, ведущего диалог с программой. Авторские языки порой оказывались не проще ассемблера для средних ЕС ЭВМ.

В идейном плане использование ЭВМ дало не так много по сравнению с докомпьютерным ПО. Здесь имела место все та же покадровая организация информации, переходов в зависимости от варианта ответа. Но появились и новые дидактические функции. Это – генерация заданий на ЭВМ и соответственно, программы-тренажеры. Это упомянутый выше сбор статистики обучения. Наконец, были опробованы новые варианты ответов – ответ – число, ответ – формула и т.д.

Анализ результатов ПО привел А.Г. Молибога к выводам об основных противоречиях ПО [35]:

1) отсутствие развития речи, свободного изложения мыслей;

2) формальность разбиения информации на кванты – экраны;

3) необходимость безмашинного итогового контроля;

4) выбираемые из списка готовые ответы не отражают сложности состояния знаний обучаемого;

5) изоляция обучаемого от коллектива (один на один с ЭВМ).

Основной недостаток – пассивность обучаемого, ведущая к «дрессировке», хотя и скрытая за большими возможностями компьютерной системы.

Все же нельзя отрицать основной результат. Была показана возможность автоматизации хотя бы части рутинного педагогического труда. Были сформулированы проблемы для педагогики и психологии компьютерного обучения. Была выявлена и поставлена грандиозная задача содержательного контроля правильности ответа обучаемого.

2.4. Состояние ППС сегодня

Современные ППС связаны технически с персональными ЭВМ, а идейно – с новыми результатами в педагогике и психологии компьютерного обучения.

Компьютерная «персональная» революция смела «динозавров» – средние ЭВМ с их низкой надежностью и большими машинными залами. Заодно были сметены и результаты АОС, разработанных на машинах третьего поколения. Предельная доступность ПЭВМ привела к тому, что на первых порах обучающие курсы стали писать все, кто знал «три аккорда» на Бейсике: IF, GOTO и PRINT.

Итак, произошло следующее.

1. Откат, отказ от авторских языков. Создаваемые программы писались всякий раз с нуля, чаще всего на Бейсике.

2. Погоня за внешними эффектами. Нередко красивая заставка к программе рисовалась с полминуты. Первый раз – интересно, второй – раздражает, третий – хочется выключить.

3. Пренебрежение порой элементарными требованиями защиты от ошибок. Программы, нарисовав за минуту красивую заставку, нередко зависали или прерывались при вводе «не того» ответа.

С другой стороны, были предприняты и попытки механического переноса АОС со средних ЭВМ (АОС ВУЗ) на микроЭВМ (АОС АСТРА для ДВК-2М). При этом не учитывались возможности персональных ЭВМ, наличие процессора на каждом рабочем месте.

В настоящее время оба этих крена в основном преодолены и разработаны АОС нового поколения, например, АОС АССИСТЕНТ. Все же при написании дипломных проектов студенты при отсутствии квалифицированного руководства или консультирования нередко пытаются писать обучающие программы прямо на Бейсике.

Произошло расширение типов обучающих программ и используемых средств. Почти всегда используется графика и даже интерактивная. Это было почти невозможно на средних ЭВМ – для диалоговой графики нужен отдельный, персональный процессор. Соответственно, появились моделирующие программы, как правило, невозможные без графики и персонального процессора. Более «плотным» стал диалог. Приведем пример: Раньше единственный центральный процессор не мешал и не мог помешать писать на экране терминала все что угодно. Он забывал о пользователе, пока тот не нажал клавишу «ВВОД». Теперь процессор стал персональным. Он может просто не пустить, не дать написать не ту букву, например, не букву «и» в слове «пр_ставка» на месте пропуска.

Важный результат – улучшение надежности работы персональных ЭВМ. Теперь учебный процесс обычно прерывается не перезагрузкой, а звонком конца занятия.

Появился новый тип программных средств – компьютерные диалоговые среды. Это программы, поддерживающие деятельность человека за компьютером: активен человек, а не программа!

Минус начального этапа персонализации – не всегда достаточное быстродействие ПЭВМ, особенно при реалистичной графике. Труднее стало организовать сбор статистики обучения всего класса или группы.

2.5. Проблемы компьютерного обучения

Основные трудности компьютерного обучения связаны с отсутствием разработанной психолого-педагогической теории такого обучения. Но она и не могла быть создана без практики. В идеальной ситуации ППС должны разрабатывать вместе психолог, методист-предметник и программист.

Частичная, но важная причина неудач, и не только в образовании – стратегическая ошибка, установка в 70-е годы на развертывание производства средних ЭВМ типа ЕС (IBM), к тому же на плохой элементной базе, причем в это время Запад от средних ЭВМ типа IBM-360 уже отказывался. В это время уже существовали прекрасные отечественные ЭВМ второго поколения БЭСМ-6, но они были забыты.

Г Л А В А 3. ПСИХОЛОГИЯ И КОМПЬЮТЕРНОЕ ОБУЧЕНИЕ

3.1. Компьютер – обучению

С точки зрения психологии, применение компьютера для обучения дает следующее [32]:

1. Расширены возможности предъявления информации: графика, динамика, текст, звук. Компьютер уже не уступает в этом радио и телевидению, а в силу возможности вмешательства обучаемого в сюжет потенциально превосходит их.

2. Усиливается мотивация учения. Новизна самой среды, дифференциация сложности заданий обеспечивают это. Скорее, нужно даже снимать эффект новизны.

3. Увеличивается глубина контроля (увы, в пределах формальной правильности ответа).

4. Развивается способность к рефлексии своей деятельности, способность взглянуть со стороны на себя и свою работу. (Этот процесс порой бывает болезненным и обучаемыми высказываются крайние точки зрения: либо машина, как и автоинспектор, всегда права либо она «вообще ничего не понимает»).

5. При передаче части рутинных функций компьютеру педагог может больше внимания уделить воспитанию.

3.2. Теория поэтапного формирования умственных действий

В качестве идейной основы классических АОС нередко объявляется теория поэтапного формирования умственных действий [52]. Но следует отличать провозглашение следования данной теории как дань моде и высокой науке и фактическую реализацию ее в конкретной обучающей программе.

Согласно этой теории, процесс формирования новых знаний проходит через следующую последовательность этапов.

1) создание мотивации;
2) разъяснение или выделение ориентировочной основы для действий;
3) формирование действия в материальной форме;
4) формирование действия в громкой речи;
5) формирование действия во внутреннем плане.

Отметим сразу же, что знание здесь заменено действием, умением, а это не одно и то же.

Эти этапы можно описать в терминах программирования. Так, этапы 3 и 4 есть свертывание усваиваемой последовательности действий в подпрограмму, вспомогательный алгоритм. На этапе 5 эта подпрограмма вызывается человеком из своей памяти и выполняется как целое, не задумываясь, возможно, во время переключения внимания на другую деятельность, в «фоновом» режиме.

Вспомним, например, как обучаемые осваивают новую диалоговую среду, например, текстовый редактор или Бейсик. Вначале читается конспект или инструкция. Затем начинается «рабочий шумок», действие осваивается во внешней речи. Если за компьютерами работают по двое малышей, это сопровождается хватанием друг друга за руки, возможно и легкими взаимными оскорблениями. После освоения действия работа выполняется молча, целые последовательности элементарных действий (например, ввод слова RUN в диалоге с Бейсиком) выполняются автоматически. Здесь интересно отметить, что если умение оказалось непрочным, то через неделю снова начинается громкая речь, обсуждение действий и затем замолкание. Вот почему на начальной стадии усвоения нового материала полезнее работа вдвоем. Работающий один на один с компьютером все равно шевелит губами, формируя действие во внешней речи.

Данная теория, безусловно, полезна при описании процесса выработки некоторых операционных навыков. Так, при разработке клавиатурного тренажера полезно подумать, через какие этапы пройдет учащийся. Это подход глубоко идейно связан с программированием. Фактически программируется ученик.

Но теория поэтапного формирования умственных действий не уделяет достаточного внимания предметной стороне действия, содержанию формируемых понятий. В крайнем смысле – это теория деятельности без

предмета деятельности. Блестящий пример на эту тему – понятие о множестве, сформулированное Алисой в Стране чудес: «множество из ничего – просто множество».

Рассматриваемая теория хорошо описывает ситуации, когда действия важнее, чем предметы, с которыми человек действует, т.е. действия универсальны. Так, например, малыш, сопя и путаясь, усваивает один-единственный тип узла для завязывания шнурков. А взрослый человек привязывает этим же узлом все что придется, и бельевую веревку, и рыболовный крючок, если только не освоит другие узлы.

Заслуга теории ПФУД в том, что было обращено внимание на механизмы формирования действий как результат и как способ научения. Но она недооценила объект деятельности, свела знание к умению действовать. Недооценка ситуации для действия приводит порой к смешным и странным результатам. Так, сказочный герой Иван-дурак бывал бит именно за то, что не мог различить свадьбу и похороны, и пускался в пляс (действовал!) там, где надо бы поплакать. Известен и анекдот о том, как слабый студент на педпрактике учил детей сложению дробей – числитель с числителем, знаменатель со знаменателем. Налицо механический перенос действия сложения с целых чисел на дроби.

Отметим, что перенос как важнейшее умение, возникающее на высоком уровне усвоения знаний, как раз может выявить формальность усвоения процедур, недооценку обстановки, в которую перенос выполняется.

Какова же роль теории ПФУД в компьютеризованном обучении? Видимо, она в основном ограничена программами тренажерного типа. Ведь не только действия, но и знания, данные являются содержанием компьютеризованной деятельности. Другое дело, что теория ПФУД сама насквозь алгоритмична, в этом ее глубокое сходство с программированием (можно сказать, с программированием ученика на новый тип действий). Характерен сдвиг в применении теории, выполненный в [53]: именно в деятельности детей младших возрастных групп преобладает алгоритмическая составляющая, там эта теория наиболее эффективна и практична.

3.3. Ассоциативно-рефлекторная теория

Продолжим аналогию с программированием. Итак, в программировании есть алгоритмы, но есть и данные, которые обрабатываются алгоритмами, структуры и типы данных. При этом в современном программировании данным уделяется все больше внимания. В базах знаний уже не алгоритмы применяются к некоторым данным, а данные выбирают себе алгоритмы для обработки.

Аналогом данных в обучении являются собственно знания, понятия, взятые в системе взаимосвязей между ними, т.е. в ассоциациях. Есть поэтому и соответствующая точка зрения, и теория: ассоциативно-

рефлекторная (АРТ). Кстати, «старая» психология уделяла ассоциациям больше внимания.

Согласно АРТ [27], [61], в основе психической деятельности лежат ассоциации и их цепи, а мышление основывается на установлении или отбрасывании связей между предметами и явлениями, взятыми в понятиях или суждениях.

Выделяют ассоциации по сходству (Паскаль и Бейсик – языки программирования), противопоставлению (белое – черное), по смежности – соседству в пространстве-времени (Шел дождь, новый фильм и два студента; один в пальто, другой – в кино.) Сходство позволяет распознать новый предмет или явление, выделить его из многообразия окружающего мира, различие после этого указывает на именно данный объект. Отношения сходства - различия тесно связаны с понятиями род и вид (например, род – человеческий, вид – китаец).

Рассмотрим практический пример. Говоря о клавиатуре компьютера, сначала следует указать на сходство с клавиатурой пишущей машинки: расположение клавиш, пробел, возврат каретки, а лишь затем и на различия – клавиши со стрелками, вставка/удаление, функциональные клавиши. Заметим, что сильный учитель, владеющий материалом, обязательно начнет со сходства, установления связей с уже известным, и лишь потом укажет на различия. Троечник-студент на педпрактике, слабо владеющий материалом, укажет на какую то деталь – отличие, на новый термин. Так проще сохранить лицо перед классом, формируя свой образ как специалиста.

Итак, знания в разуме человека существуют в виде понятий, связанных сетью ассоциаций. Новое знание возникает при установлении новых понятий и связей между ними. Важным вкладом в эту теорию является разработанная [61] идея о том, новые знания возникают не только в результате образования, но и разрыва ассоциаций. Так, инерция мышления привела к тому, что после изобретения паровой машины она вначале использовалась для того, чтобы лить воду ... на водяную мельницу! Понадобились десятилетия для того, чтобы разрушилась ассоциация «двигатель – водяная мельница» и возникла ассоциация «двигатель – паровая машина».

Учет ассоциаций крайне важен именно в компьютерном обучении. Ведь информация по своей природе многолика, одно и то же знание можно представить массой способов: текстом, графикой, миганием, звуком. В свою очередь, графические способы представления информации тоже весьма многообразны (график-кривая, столбчатая диаграмма, круговая диаграмма и т.д.). Внутри- и межпредметные связи, важность которых общепризнана, так же реализуются на основе развернутых цепей ассоциаций. Наконец, творчество, и не только техническое, широко использует цепи ассоциаций. Не случайно профессиональные изобретатели активно работают со словарями синонимов, подбирая различные слова для близких по смыслу действий [61]. Например, к слову «колоть» (ледокол) подбираются

замены и соответственно, способы прохождения через льды: «пороть», подсовывая корпус судна под лед, «пилить», «плавить» горячим корпусом судна и т.д. (Кстати, наиболее радикальное решение – ничего со льдом не делать, плыть подо льдом). Именно ассоциации позволяют более корректно ставить вопросы при компьютерном контроле, резко упрощая реализацию. Рассмотрим пример.

Изучается тема «Автор – ЭВМ – исполнитель». Локальная цель обучения – сформировать пару понятий: автор алгоритма и исполнитель. Поставим вопрос: Кто сочиняет алгоритм? Желательный ответ – автор. Но и другие правильные ответы на поставленный вопрос весьма многообразны: человек, ученик, программист, учитель и т.д. Выполним поэтому двойную операцию – ассоциативное расширение/сужение вопроса: Кто сочиняет книги, музыку, программы для ЭВМ и получает за это гонорар? Здесь уже трудно придумать ответ, кроме слова «Автор».

Из изложенного следует, что ассоциации играют важнейшую роль при компьютеризованном обучении. Именно логическая однородность компьютерных «миров» позволяет смело выполнять операции типа переноса умений в новую обстановку, строить ассоциативные связи между различными компьютерными средами. Среды, не поддерживающие единообразие команд и понятий, не получают распространения, непопулярны.

Приведем простой пример. Клавиши Ctrl-Ins в текстовом редакторе среды FAR запоминают выделенный блок. Действуя по инерции (перенос!), обучаемый выполнит то же и в среде WINDOWS и будет прав.

Ассоциации – важнейший инструмент закрепления знаний. Понятие, связанное нитями ассоциаций в систему с другими понятиями, труднее забыть или удалить из оперативной памяти человека. Чем больше связей, тем богаче и прочнее понятие.

3.4. Два типа мышления

Интересно наблюдение, выполненное автором за работой двух студентов. Первый их них, горожанин, разместил все команды, необходимые для решения задачи, в тексте программы в произвольном порядке: получилось нечто вроде супа из команд. Видя явное замешательство преподавателя, студент недоуменно спросил: а разве ЭВМ сама не разберется, в каком порядке выполнять команды?.. Второй персонаж, студентка из села, испытывала непреодолимые трудности при изучении фактической стороны вопросов, конкретных сведений по дисциплине «Численные методы». Но если ей было указано, каким алгоритмом пользоваться при решении конкретной задачи, все шло без проблем. Это две крайние ситуации. Многолетние наблюдения автора за сельскими и городскими студентами привели к выводу, что реально существуют и два типа (уклона) мышления – назовем их сельское и городское. Студенты из села лучше усваивают операци-

онную сторону знаний, составляют и выполняют алгоритмы. Студенты – горожане лучше усваивают содержательную сторону знаний, уступая сельчанам в операционной компоненте.

Автор пришел к выводу, что дело здесь в обстановке, в которой воспитывается и растет человек. В сельском доме меньше вещей, но все они являются инструментами для действий: подойник – чтобы доить, веник, чтобы мести и т.д. Итак, преобладает деятельность. В городской квартире ребенка окружает больше вещей «ненужных»: например, много книг, пуфиков, картин и т.д., преобладает информация. Отсюда и соответствующий крен в формировании мышления. У студентов с ярко выраженными способностями в области информатики подобных кренов отмечено не было.

Итак, на сегодня не существует законченной или хотя бы развитой всесторонней психолого-педагогической теории компьютерного обучения. Нередко авторы программных систем действуют по интуиции, на свой страх и риск. Тем более осторожно следует относиться к рекламным объявлениям об использовании при разработке ППС каких-то психолого-педагогических теорий.

Г Л А В А 4. ЭВМ КАК СРЕДСТВО ИНФОРМАЦИИ

4.1. Предъявление информации в АОС. Гипертексты

Текст на экране и на столе. Среди материалов, предъявляемых при помощи ЭВМ, можно различать следующие: учебник, справка, помощь, тексты заданий, сообщения об ошибках и т.д.

Традиционно учебная информация предъявляется в виде текста. Новым моментом сегодня по сравнению с первыми компьютерными текстами является возможность воздействия обучаемого на процесс предъявления информации. Простейший прием – нажатие клавиши для смены кадра. Программы, предъявляющие новый текст при нажатии клавиши, так и называются – электронными листальщиками страниц. Учитывая излучение, идущее от экрана ЭВМ, вряд ли целесообразно листать тест на экране, проще взять книгу или распечатку.

Гипертекст и мультимедиа. Заслуживает интереса возможность более сильного влияния обучаемого на процесс предъявления информации, которое предоставляют так называемые гипертексты. Здесь по встроенной системе ссылок обучаемый может переходить в интересующее его место, снова углубляться, возвращаться и т.д. Работа напоминает по стилю чтение энциклопедического словаря, с той важной разницей, что компьютер автоматически запоминает сделанные переходы и позволяет без труда переходить сколь угодно далеко вперед и назад по тексту.

Недостатком гипертекстовых материалов является то, что система ссылок жестко фиксирована при составлении такого текста и не зависит от того контекста содержания терминов – ссылок, который существует в голове обучаемого в конкретный момент учебного времени. Здесь помог бы развернутый поисковый запрос.

В случае, если гипертекст содержит графическую информацию, сопровождается звуком, говорят о мультимедиа-среде. Наконец, гипертекст может содержать и задания и упражнения.

Что же изменилось по сравнению со временем перепутанных учебников? Стало легче и удобнее перемещаться по такому тексту. Расширился спектр средств воздействия: графика, цвет, звук. Но особенно важно, что решение о переходе к какому-то месту текста теперь принимает не программа, а человек. Потенциальная важная возможность – организация контекстного поиска на гипертекстовом файле.

4.2. Логическая организация текста

Структурно-логические схемы, их построение. Для того, чтобы создать АОС или хотя бы гипертекст, необходимо описать для себя его структуру. Этой цели служат структурно-логические схемы (СЛС). Графически – это сеть из узлов. Узлы представляют собой единицы (кванты, модули) информации, а линии, связывающие узлы, указывают на логические связи в представляемом материале. Если некоторый квант В по времени изучения должен следовать за квантом А, линия превращается в стрелку, учитывается направление связи.

Формирование модулей информации трудно автоматизировать. Дело в том, что они должны обладать тем, что в разработке больших программных систем называется информационной прочностью модуля. Это значит, что информационные связи внутри модуля должны быть сильнее его внешних связей. Единицей измерения связи между парой модулей текста может являться некоторое важное для данного курса понятие, которое встречается в обоих модулях. Если некоторое понятие впервые вводится в модуле А, то модуль В, использующий это понятие, должен изучаться после модуля А.

Итак, после разбиения на модули нужно текст упорядочить. Даже если содержащееся в нем знание представляет собой сеть и именно так должно отложиться в голове обучаемого, для «ввода» его в память нужна определенная последовательность. Модули нужно расставить в линию, сохраняя связи. Линия упорядочивания – это линия времени, позиции модулей – моменты их изучения. Очевидно, что формально расстановку можно сделать комбинаторно большим, практически необозримым числом способов. Какой же лучше? Здесь на помощь приходят общие принципы дидактики и просто здравый смысл. Ясно, что сильно связанные модули должны

изучаться в близкие моменты времени. Кроме того, стрелки должны быть ориентированы вправо: в общем, нельзя свободно пользоваться понятием до его определения, особенно в математике. Для упорядочивания СЛС можно применить и ЭВМ. В случае, если СЛС содержит до десятка блоков, возможен простой перебор вариантов порядка, минимизирующий, например сумму длин ребер полученного графа при его отображении на ось времени.

Логические круги, их снятие. В случае дисциплин с сильными внутрипредметными связями (информатика) задача упорядочивания материала может показаться порой неразрешимой. Так, модуль В должен следовать за А, А – за С, и С – за А. Выходов из этой ситуации несколько. Во первых, следует проанализировать модули на прочность и, если можно, разделить один из них на 2, например С на С1 и С2. Возможно, этого будет достаточно, если С1 удастся перенести в начало циклической цепочки. Второй прием менее очевиден и основан на приеме опережающего обучения. Так ли уж опасно использовать понятие до его определения? Возможно, что, упростив контекст, его можно оставить на месте. В обоих случаях будет иметь место циклическое изучение материала, обобщающее повторение Учебник [25] полон примеров и первого и второго вида. Так, понятие вспомогательного алгоритма появляется неоднократно в течение курса, обогащаясь все новыми контекстами. Без определения понятия переменной используются команды Чертежнику сметиться_на_вектор (А, В) и т.д. При таком цикличном изложении учебник сохраняет и логическую стройность.

Итак, построение СЛС – важный этап разработки обучающей программы. Не стоит заниматься программированием до составления СЛС курса.

ГЛАВА 5. ГЕНЕРАЦИЯ ЗАДАНИЙ В АОС. ТРЕНАЖЕРЫ

5.1. Принципы генерации заданий

Пассивность обучаемого в классических АОС была отмечена давно. Так же давно были начаты разработки программ, предъявляющих задания и тем самым активизирующих познавательную деятельность обучаемых. Они получили название «тренажеры».

Легче всего тренировать деятельность, сводящуюся к моторным навыкам, например, навык десятипальцевого метода работы на клавиатуре. Поэтому так много существует клавиатурных тренажеров. Они легко превращаются в игры – кто быстрее выполнит задания. Сложнее дело обстоит с формированием навыков решения задач для ума.

Задания, предъявляемые обучаемому, должны быть связаны по принципу «цель – средство»: решенная задача должна содержать средства для решения следующей.

Автоматизация формирования заданий связана с большими трудностями концептуального характера, так как требует привлечения трудно формализуемого педагогического опыта. Поэтому на сегодня существует два основных способа предъявления задач – случайная выборка готовых задач из банка задач либо порождение варианта задания из так называемой метазадачи путем варьирования ее параметров.

В первом случае законченные задачи, ранжированные по трудности, вводятся в память ЭВМ и выбираются случайным образом. Недостаток метода – проблемы с исключением повторений условий при повторных попытках обучаемого, расточительное отношение к банку задач. Его создание – ручная и весьма трудоемкая деятельность. При отладке обучающей программы машинные «случайные» числа должны повторяться при каждом запуске, при эксплуатации – обновляться командой типа RANDOMIZE.

Во втором случае задача снабжается переменными параметрами и тем самым расширяется до целого класса. Будем называть ее метазадачей. Приведем пример, не очень серьезный:

Бригада из 12 школьников собрала 5 тонн свеклы
Группа из 25 студентов затоптала 9 тонн моркови
Звено перебрало 12 тонн картофеля.
Сколько переработал один школьник/студент?

Здесь слова и числа в столбиках – варианты параметров метазадачи. Контроль решения конкретного варианта легко выполнить, взяв в качестве эталонного ответа отношение 5-го и 2-го параметров (чисел). Тем самым метазадача может иметь единое общее решение для всех вариантов.

Ясно, что число вариантов быстро растет с увеличением числа параметров – в 2–4 раза на каждый параметр. Особенно удобен метод параметризации в вычислительных задачах. Пример: вычислить приближенно интеграл от функции (...) на промежутке (...) при помощи метода (...) с точностью до (…).

При таком методе генерирования задач возникают две проблемы.

1. Сходство задач. Его можно уменьшить, если не пропускать задачи, которые отличаются только одним параметром. Этого легко добиться, если хотя бы один из параметров выбирается не случайно, а является функцией других параметров.

2. «Патологические» сочетания параметров, при которых решения нет. Отличный пример – задачи на решение треугольников, где параметрами являются длины сторон. Методисты-математики считают, что несуще-

ствование треугольника – тоже важная ситуация. Но проблема для генерации здесь состоит в том, что если не следить за случайно выбранными параметрами, такие вырожденные случаи могут составить подавляющее большинство.

5.2. Пример тренажера

Довольно много есть тренажеров на операции умножения и сложения. Они просто устроены и существуют давно. Теперь они перенесены под WINDOWS с окнами, кнопками и т.д. Рассмотрим такой простой тренажер на Бейсике.

```
      PR = 4                          ' произведение
      NP = 15                         'максимум числа попыток
10    IF PR < 4 THEN PR = 4           'фильтр малых произведений
15 A = INT(RND * PR)                  'генератор числа A
      IF A <= 1 THEN 15               'фильтр A
      B=INT(PR / A) + 1               'генератор числа B
      IF B <= 1 THEN 15               'фильтр B
      ?A; «*»; B; «=«;                'вывод задания
      INPUT C$                        'строковый ввод ответа
      C=VAL(C$)                       'преобразовать в число
      IF C$ = «« THEN END             'отказ отвечать (ENTER)
      IF C = A * B THEN 20 else 30 'если ответ верный
20 PR = PR + 1                        'сложность увеличить
      S = S + 1                       ' попытку учесть
 30   GOTO 40                         'иначе
      PR = PR - 1                     'сложность уменьшить
      N = N + 1                       'счетчик попыток
 40 IF N <= NP THEN 10                'еще мало упражнений
      ?»Вы набрали»; S; «очков из»; N 'статистика
      END
```

Даже в этой простой программе видны типичные для тренажеров следующие логические блоки и их функции.

1. Блок генерации условий. Работает на основе датчика случайных чисел.
2. Фильтры. Не пропускают слишком простые варианты.
3. Предъявление задания в текстовом виде.
4. Ввод ответа.
5. Проверка правильности ответа.
6. Выход из диалога с тренажером.
7. Адаптация сложности следующего задания.
8. Учет числа правильных ответов.

9. Учет числа попыток.
10. Итоговая выдача результата.

Итак, блоков в этом тренажере оказалось чуть меньше, чем строк.

5.3. Учет фактора времени

Для тренажера, работающего в режиме реального времени, важно ограничивать время на ответ. Это делается при помощи функции TIME. Эта функция возвращает текущее время. Рассмотрим ее использование на примере клавиатурного тренажера.

```
        CLS                             'очистить экран
 ?»клавиатурный тренажер с ожиданием ответа по таймеру»
        RANDOMIZE TIMER                 'встряхнуть случайные числа»
10 Z = INT(RND * 32)                'случайный номер символа
        S$ = CHR$(Z + ASC(«A»))         'преобразовать в литеру
        ?»нажмите «; S$                 'выдать задание
        T1 = TIMER                      'значение T до ввода
        DT = 2                          'интервал ожидания
 20 A$ = INKEY$                     'проверить нажатие клавиши
        T2 = TIMER                      'новое время
        IF T2 - T1 > DT THEN 10         'время ожидания истекло
        IF A$=«« THEN 20                'ввода не было
        ?A$                             'выдать нажатую клавишу
        IF A$ = S$ THEN S = S + 1       'число удачных ответов
        N = N + 1                       'всего попыток
        ?»У Вас»; S; «из»; N; «ответов»'выдать статистику
        IF ASC(A$)<>27 THEN 10          'нажато не Esc, продолжать
 30 END
'Упражнения:
'генерировать малые буквы
'генерировать функциональные клавиши
'менять время ожидания в зависимости от правильности
'менять время ожидания в зависимости от успевания отвечать
```

5.4. Задачи с обращением условия

Значительно разнообразнее выглядят задачи, где меняются местами известное и неизвестное, условие обращается. В простейшем случае базой для такой задачи является одна формула. Рассмотрим пример.

Пусть дана физическая формула s = v * t, где s – путь, v – скорость, t – время. Отсюда можно получить 3 задачи с неизвестными s, v, t соответственно:

Дано v, t. Найти s.

Дано s, v. Найти t.

Дано s, t. найти v.

Все варианты задачи проверяются подстановкой значений в одну формулу: s = v * t.

```
'Генератор с инверсиями условия S = VT
RANDOMIZE TIMER              'встряхнуть случайные числа
D = 1                        'точность вычислений
P$(0) = «скорость»           'имена трех параметров
P$(1) = «время»              'в кольцевом списке
P$(2) = «путь»               '
I0=INT(RND * 33) MOD 3       'кольцо номеров параметров
I1=(I0 + 1) MOD 3 '
I2=(I0 + 2) MOD 3 '
P(I0)=ROUND(RND * 10, D)     'случайные параметры
P(I1)=ROUND(RND * 10, D)     'округлены до D знаков
?»Дано: «; P$(I0) + «=«; P(I0);       'печать условия
?» и «+P$(I1) + «=«; P(I1) '
?»Надо найти «; P$(I2)       'выдача задания
?»Ваш ответ: «:INPUT P(I2)   'неизвестная величина
P(I2) = ROUND(P(I2), D)      'округлить до D знаков
IF P(2) = P(0) * P(1) THEN   'формула S=V*T
?»верно»                     'диагноз
ELSE '
?»неверно»                   '
ENDIF
END
```

Интересно может выглядеть диалог обучаемого с такой программой. Работая в DOS под WINDOWS и получив на экране задание, обучаемый может отложить диалог путем Ctrl-Esc, запустить все тот же BASIC или калькулятор, вычислить с его помощью ответ, и вернуться к диалогу с тренажером. Так в WINDOWS попутно решена непростая задача обеспечения диалога с АОС вспомогательным средством для счета.

5.5. Случай нескольких неизвестных

На практике в задачах может быть и несколько неизвестных. Богатым объектом для примеров такого рода является треугольник. Но, оставаясь на материале механики, можно рассмотреть более простой случай: две формулы, связывающие 4 величины: t, v, a, s

s = v * t / 2, v = a * t

для случая равноускоренного движения.

Выбор системы 2 базовых формул неоднозначен, (есть, например и формула s = a * t ** 2 / 2), но это не создает трудности. Важно, чтобы формулы были независимыми. В данном случае они порождают 6 условий задач (число сочетаний из 4 по 2). Контроль выполняется подстановкой известных и неизвестных в две базовые формулы, приведенные выше.

Г Л А В А 6. РЕАЛИЗАЦИЯ КОНТРОЛЯ В АОС

6.1. Состояние проблемы контроля ответа

Контроль – видимо, наиболее трудная проблема в компьютеризованном обучении, что было понято достаточно рано. Тем не менее, продолжается разработка АОС с нередко старыми приемами контроля ответов на технике нового поколения микроЭВМ, с повторением старых дидактических ошибок. В то время как в литературе, описывающей перспективные разработки [60], из года в год повторяются термины типа «семантическая сеть», «фрейм», и т.д., в реально используемых в учебном процессе разработках мы снова встречаем все тот же выбираемый ответ, ответ, проверяемый по ключам, на совпадение с эталоном. Впрочем, есть и нововведения – «наборное меню».

Ответ обучаемого есть, строго говоря, некоторое утверждение о свойствах изучаемых объектов. Оно может быть истинным, ложным, неточным, неполным и т.д. Но сам по себе ответ не является командой для обучающей программы. Решение о том, что делать дальше, принимает обычно программа, точнее, ее автор, на основе обработки ответа обучаемого.

Итак, основным элементом обратной связи в «словесных» системах обучения является ответ обучаемого. Рассмотрим основные типы ответов и реальные возможности автоматизации их контроля.

Можно классифицировать ответы по 2 основаниям:

1) по активности обучаемого – ответ, выбираемый из меню, свободно конструируемый, частично конструируемый.

2) по типу информации – число, текст, формула, график.

6.2. Выбор ответов из готовых вариантов

Выбор единственного ответа. Итак, на экране в современном стиле представлено меню ответов. Какие здесь есть технические возможности и каков их дидактический результат?

Простота технической и программной реализации сделали выбираемый необычайно живучим. Он прошел интересную эволюцию. В перепутанном учебнике нужно было перейти на указанную страницу по номеру выбранного ответа. В АОС на базе средних ЭВМ нужно было напечатать на экране номер выбранного ответа. На персональных ЭВМ ответ уже выбирается из списка при помощи меню и подтверждается нажатием кнопки «Ok». «Прогресс» налицо...

Но давно замечено психологами, что нередко происходит непроизвольное запоминание всего, что имеется на экране, правильных и неправильных ответов. Возможно, через какое-то время обучаемый вспомнит все ответы, которые он видел на экране, но забудет, который из них был правильным...

Основной же недостаток такого ответа в том, что он обеспечивает в лучшем случае распознавание уже виденного, даже не воспроизведение знаний, а самый нижний уровень усвоения материала. Отметим, что этих уровней можно выделить четыре:

1) распознавание информации («Это я видел»);
2) воспроизведение информации («Это я помню);
3) применение знаний в стандартной ситуации («Это я умею»)
4) перенос («Этим я владею свободно»).

Есть нечто противоестественное в том, что педагог сам придумывает за учащегося ошибочные ответы, вместо того, чтобы стремиться вообще избавиться от них.

Этот способ контроля должен быть ограничен ситуациями, где выбор вытекает из существа дела, например, выбор подходящего инструмента для работы из множества в принципе возможных для использования в этой работе. Но тогда хорошо бы и обосновать выбор, а этого программа и не понимает.

Выбор неупорядоченного подмножества ответов. Некоторые новые возможности предоставляет ответ, технически реализованный в виде «наборного меню». Здесь возможны два подхода. Обучаемый выбирает несколько вариантов ответа. Выбор считается правильным, если ничего не пропущено. Этот прием позволяет давать задания на разделение некоторых объектов на 2 группы. Приведем пример.

Укажите номера веществ, которые являются металлами:

1) фтор
2) бериллий
3) цирконий

4) углерод
5) бром
6) ртуть

Порядок правильных ответов, конечно, произвольный.

Вероятность случайного угадывания равна 1/2^N. Здесь N – полное число ответов.

Выбор упорядоченной серии ответов. Вторая возможность связана с фиксацией порядка выбора пунктов меню. Ответ считается верным, если выбраны те и только те пункты меню, которые нужно и притом в правильном порядке.

Тогда наборный ответ можно скорректировать в более полезную форму. Можно предложить набор вопросов, близких по смыслу и параллельно с ними выдать на экран случайным образом переставленный набор готовых правильных ответов. Если способность к репродукции знаний у обучаемого уже сформирована, такое задание потребует от него умственного усилия с целью дифференцировать эти тонкие различия между ответами. При этом автору контролирующей программы не нужно придумывать ложные альтернативы. Приведем пример.

Вопрос: какой командой нужно запустить Бейсик?	Ответы:
1) с сохранением значений переменных и с начала;	RUN
2) после прерывания по Ctrl- Stop;	GOTO 0
3) для продолжения после 60 STOP в программе;	CONT
4) с обнулением переменных и с первой строки;	GOTO 70

Введите номера ответов в правильном порядке.

Правильный ответ будет таким: 2 4 3 1.

Отметим, что вероятность угадывания уменьшается с 1/N в случае простого выбора до 1/N! в случае задания на упорядочивание ответов и для 4 вопросов составит всего 1/24. Конечно, часть ответов может быть получена методом исключения, но при этом учащийся получит или проявит дополнительные знания, а не даст заведомо ложные ответы. Кроме того, диалог становится более насыщенным.

Таким же образом можно контролировать усвоение правильного порядка действий, например, алгоритм включения дисплейного класса.

Подчеркнем, что все это варианты контроля относятся в лучшем случае к репродуктивной деятельности либо просто к узнаванию объекта.

Двоичный ответ (да/нет). Плюс ответа «да/нет» – его близость к естественному способу диалога. Набором большого числа двоичных ответов можно в принципе хорошо продифференцировать знания. Недостаток его в том, что не учитываются оттенки типа «да, но...», «не совсем так». Здесь

надо бы выслушать обоснование, задать вопрос «Почему Вы так считаете?», но программы на это пока не способны. Есть и опасность навязывания небесспорного мнения автора контролирующей программы думающему обучаемому. Требуется и полная точность вопросов. Велика вероятность угадывания: 50%. Выход заключается в серийности вопросов, их значительном количестве. С дидактической точки зрения и с учетом оговорок такой ответ удовлетворительно контролируется компьютером и позволяет в принципе «вести» обучаемого с помощью хорошо продуманной системы вопросов к некоторым выводам, подобно тому, как это делал Сократ. Но это уже не задача контроля.

Троичный ответ. Излишняя однозначность двоичного ответа как будто может быть преодолена при переходе к трехвариантному ответу: «да», «не совсем так», «нет». Но ясно, что любой из крайних ответов почти всегда можно заменить ответом «не совсем так», учитывая какие-то обстоятельства, если отвечающий обладает эрудицией. Особенно это касается гуманитарных дисциплин. Например, вопрос таков: А.С. Пушкин – великий русский поэт? Напрашивается ответ «да», но и «не совсем так» тоже правильно: А.С. Пушкин – еще и достояние мировой культуры. Перечень примеров легко продолжить: Паскаль – язык программирования? Не совсем так: это и язык публикации алгоритмов, и язык для обучения программированию. Поэтому троичный ответ применим без оговорок в довольно редких ситуациях с вопросом типа 2 * 2 = 4? Но даже и здесь сильный школьник или студент может указать на альтернативу: 2 * 2 = 10 в двоичной системе счисления. Поэтому ответ «не совсем так» – подобен, образно говоря, дырке в сети из вопросов, через которую уйдет вся «рыба»: и мелко плавающий троечник, и глубокий, мудрый эрудит. Именно троичный ответ подчеркивает беспомощность компьютерного контроля, если нужно знать причины конкретного ответа, его скрытый смысл.

6.3. Свободно конструируемые ответы

Ответ-число. Поскольку ответ-число может быть результатом значительной деятельности обучаемого и легко проверяется, его следует признать одним из лучших для реализации в АОС. Если это число не ноль и не единица, вероятность угадывания близка к нулю: чисел на числовой оси бесконечно много. Пример. Чему равен интеграл по формуле

трапеций от функции Y = X на промежутке от 0 до 1 с шагом H = 1?

Ответ: 0.5.

Для получения этого ответа обучаемому «всего лишь» надо знать:

1) формулу трапеций для численного интегрирования, ее геометрический смысл,

2) частный вид формулы для единственного разбиения промежутка,

3) при нулевой длине одной из сторон трапеция превращается в треугольник, в данном случае – прямоугольный,

4) площадь этого треугольника с катетами, равными 1.

После того, как ответ найден, вопрос кажется простым, но в ответе содержится в свернутом виде значительная деятельность обучаемого.

На практике учитывают приближенность многих ответов и допускают числовую погрешность, зависящую от задачи.

В заключение этого пункта заметим, что 1/7 с точки зрения обычной математики – число, но для контролирующей программы – это уже формула.

Ответ-формула. Легко видеть, что даже простые формулы вроде E = m * v2 / 2 могут быть записаны в силу законов комбинаторики значительным, порой астрономическим числом способов:

m * v * v * 0.5, v2 * m / 2, ...

Обучаемый, решивший «поиздеваться» над компьютерными средствами контроля формул, идущими путем сравнения их с эталонами, может приписать к своему правильному ответу «+ 0.0» и будет прав, хотя ответ будет программой, скорее всего, забракован. Поэтому рассчитывать на перечисление всех допустимых способов записи не следует.

Вместо этого необходимо:

1) ввести формулу обучаемого и вычислить значение выражение для эталонной формулы-ответа при некоторых «некруглых» значениях параметров;

2) вычислить значения при тех же параметрах для формулы обучаемого;

3) допустить отклонение из-за ошибок округления, проявляющихся при изменении порядка действий из-за неоднозначности записи.

Здесь вероятность попасть в нужную точку на числовой оси теоретически равна нулю, а в силу конечности количества чисел, представимых в ЭВМ – весьма близка к нулю.

Для реализации такого ответа необходимо интерпретировать формулу во время работы программы. Даже для языковых сред интерпретирующего типа эта возможность – большая редкость. Приятным исключением являются среды семейства DBASE, где во время работы программы можно выполнить макрокоманду &, или язык Perl. Поясним это примером на Foxpro.

```
a = «m * v**2 / 2»      && запомнить в строку выражение
m = 1.2                 && задать тестовые значения параметрам
v = 3.4                 && числа берутся некруглые
?&a                     && отпечатать значение выражения
```

В современных средах типа WINDOWS можно сообщить обучаемому значения подставляемых параметров и попросить его вычислить значение по его формуле в некоторой среде, параллельно с диалогом с контролирующей программой. Заодно снимется проблема произвола в обозначениях.

Ответ-слово. Слово – наиболее типичная форма для кратких устных ответов. Основной способ контроля здесь состоит в сравнении с эталонами или ключами.

Сравнение со словами-эталонами предполагает, что список эталонов полон. Но в силу синонимии и множества словоформ живого языка это принципиально недостижимо – живой язык неоднозначен. Всегда найдется в принципе правильный ответ, не предусмотренный в программе. Частичный выход – в усилении точности вопроса через раскрытие контекста, например, вопрос, приведенный выше:

Как называется человек, который сочиняет алгоритм или оперу или роман и получает гонорар?

Здесь уже с большей вероятностью можно ожидать ответа «автор».

Можно рассчитывать лишь на некоторую вероятность предугадывания разработчиком контроля всех возможных ответов и накапливать новые правильные варианты по мере эксплуатации программы. Вряд ли это устроит обучаемых.

При контроле по ключам легко засчитать как правильный и такой ответ: «не автор»...

Частичное решение проблемы – вести список ключей, которые не должны встречаться в ответе обучаемого, в данном случае – слово «не». (Но как быть со словами «НЕвеста», «сНЕг»?)

Перед контролем ответ и ключи нужно привести к единому стандартному виду – удалить крайние пробелы, перевести все литеры к одному, обычно верхнему регистру. Это называется канонизацией ответа. Отметим, что в случае ответа-числа такая обработка всегда выполняется автоматически: 1 =1.0.

Ответ-предложение. Комбинаторное богатство языка сразу обрекает на неудачу попытки предусмотреть все варианты такого ответа. «Нормально» ответ должен обрабатываться на основе базы знаний средствами синтаксического и семантического анализа, которые относятся к области, традиционно именуемой «искусственный интеллект» и где пока проблем обнаруживается больше, чем результатов [34]. (Впрочем, редактор WORD уже располагает средствами поиска опечаток, и даже правки стиля, что вселяет некоторые надежды). На практике АОС предлагают контроль по ключевым словам, даже с учетом их порядка в ответе обучаемого. К сожалению, с этой точки зрения, утверждения «ГРЕки ПОБИЛИ РИМлян» и «ГРЕков ПОБИЛИ РИМляне» окажутся, скорее всего, равноценными

(Прописные буквы относятся к вероятным ключам, составленным с учетом неустойчивости вариантов словоформ).

Что же на самом деле контролируется в предложении? Контролируется знание обучаемого о факте события: кто-то с кем-то воевал, а вот кто кого победил – неизвестно! Такая точность информации допустима в информационно-поисковых системах (ИПС), где как раз и может быть нужно узнать от ЭВМ, кто же кого победил. При контроле такая точность ответа хорошо описывается пословицей: «слышал звон...».

Итак, на сегодня контроль ответа-предложения недоступен компьютеру, по крайней мере, школьному, если только этот ответ – не цитата из классиков.

Учитывая изложенное, целесообразно при отрицательной оценке ответа обучаемого «открыть» процедуру контроля, чтобы он мог убедиться в объективности оценки (или наоборот, ее неправильности).

Ответ-группа слов. Это не так безнадежно, как ответ-предложение, если слова четко задаются. Примеры задания: расшифруйте сокращение – ППС. Или: какая наука занимается проблемами преподавания информатики? Конечно, полезно расширить ответ синонимами к каждому слову и засчитывать его, если в ответе ученика есть хотя бы по одному слову из каждой группы синонимов: «методика И (обучения ИЛИ преподавания) И (информатики ИЛИ программирования). Здесь обнаруживается внутрипредметная связь с темой: «информационный поиск».

6.4. Частично-конструируемые ответы

Трудности смыслового анализа ответа обучаемого привели разработчиков средств контроля в АОС к идее частично конструируемого ответа. При этом ответ собирается из заранее подготовленных элементов. Возможности обучаемого фантазировать резко сужаются, и программа скорее найдет с ним общий язык. Видимо, это и имел в виду А.П. Ершов, приводя в качестве примера языка общения с ЭВМ язык деловой прозы. Возможен и такой пример – обучаемому предлагаются на экране элементы молекулы сложного вещества и возможность перемещать их с целью правильного соединения. Здесь имеет место уже не чистый контроль правильности сборки, а предварительная деятельность обучаемого в некоторой среде с фиксированными возможностями для изменений. На наш взгляд – это шаг далеко в сторону (и вперед) из проблемы контроля ответов. Приведем пример.

Проверяется правописание слова «Пр.ставка». (буква пропущена. Классическая АОС предложила бы на выбор две буквы: «е» и «и». Но следуя логике частично-конструируемого ответа, нужно выдать на экран слово и не дать вводить вместо пропуска ничего, кроме правильной буквы.

```
      CLS                                 'очистить экран
      ?»Вставьте пропущенную букву»       'подсказка
      ?»пр.ставка»                        'слово с пропуском
      LOCATE 2, 3                         'курсор – на пропуск
10    K$ = INKEY$                         'осмотреть клавиатуру
      IF K$ = «« THEN 10                  'ничего не нажато – ждать
      IF K$ <> «и» THEN 10                 'нажата не та клавиша
      IF ASC(K$) = 27 THEN 20             'отказ отвечать
      ?K$                                 'выдать правильную букву
20    END                                 'конец
```

Заметим, что все рассмотренные способы контроля относятся к результату деятельности обучаемого, а не к ее процессу, что было бы гораздо полезнее. Но такой контроль на порядок труднее выполнить хотя бы потому, что количество правильных траекторий деятельности, т.е. ведущих к правильному результату, комбинаторно велико. Кроме того, возможны нестандартные, оригинальные пути решения задачи, которые компьютерный контроль по способу деятельности скоре всего забракует как ошибочные.

6.5. Адаптивное определение числа испытаний

Компьютерное тестирование все шире применяется в практике современного образования и особенно в дистанционном обучении. Однако, при этом практика заметно опережает теорию, и некоторые концептуальные вопросы остаются неясными. Нами предлагается метод определения достаточной длины теста, которая позволяет достигнуть необходимой и объективной меры уверенности в результатах тестирования и избежать как преждевременного прерывания теста, так и излишней нагрузки на испытуемого. Метод основан на Байесовом пересчете вероятностей гипотез об уровне знаний тестируемого в процессе самого теста.

Пусть тест состоит из заданий на выбор одного из 4 готовых вариантов ответов (весьма типичная для практики ситуация). Пусть доля знаний испытуемого равна D. Тогда вероятность знакомого ответа составит D. В противном случае испытуемый будет давать ответ случайно. Итак, (в обозначениях языка Бейсик) вероятность верного ответа на одно задание такого теста равна:

PT(D) = D+(1-D)/4

Следуя Байесовской идее о пересчете вероятностей, рассмотрим семейство гипотез об уровне знаний D тестируемого. Каждой гипотезе соответствует своя доля D знаний испытуемого и своя вероятность гипотезы P(D). Ее и будем определять. Для начала естественно принять, что все ги-

потезы об уровне знаний равновероятны: P(D)=1 (ситуация полной неопределенности уровня знаний).

Проведем один тест. Если он прошел успешно, вероятности гипотез по формуле Байеса надо умножить на PT(D). Если испытуемый ошибся, вероятности гипотез надо домножить на вероятность противоположного события: 1-PT(D). После M положительных и N отрицательных исходов тестов распределение вероятностей гипотез будет таким:

PB(D, M, N)=PT(D)^M*(1-PT(D))^N,

с точностью до нормирующего множителя. Важно, что результат не зависит от порядка заданий и слабо зависит от исходного распределения. Так, если начать в распределения P=D (в пользу испытуемого), это практически сведется к одному лишнему положительному ответу. Впрочем, идея об исходной равновероятности всех долей знаний выглядит разумной и обычно подразумевается объективными педагогами на экзаменах и без ЭВМ.

Рассчитана плотность распределения вероятностей гипотез для случая M=36 и N=12.

Итак, для данного случая наиболее вероятны гипотезы о доле знаний тестируемого D диапазоне от 48% до 80%. Несмотря на значительное число испытаний (48), очевидна высокая неопределенность результата данного теста, а таким результатом является информация о доле знаний. Напрашивается вывод: тестирование надо продолжить. Но до каких пор?

Следуя принятому в статистике приему, введем доверительный интервал, исключив хвосты распределения, с суммарной вероятностью в 0.05 – обычный для педагогики уровень надежности выводов. Теперь ясно, что тестирование можно прекращать, когда доверительный интервал станет меньше заранее заданной величины, например, 20% либо (более жесткое требование) целиком войдет в диапазон, отведенный для какой-либо оценки в баллах. (Автору не удалось выяснить, откуда берутся эти диапазоны в реальных тестах. В публикациях порой смешивают долю знаний испытуемого и вероятность правильного ответа).

Предлагаемый подход позволяет довольно быстро прекратить тестирование отличника (при 16 верных ответов подряд без единой ошибки). Отделить оценку '3' от '4' значительно труднее. Это, кстати, давно известно педагогам-практикам.

Метод легко переносится на случай неоднородных тестов, если число готовых вариантов ответов меняется от задания к заданию. Достаточно изменить число вариантов (4) в знаменателе формулы для PT на текущее значение. В случае двоичного ответа «да/нет» велика вероятность угадывания.

Численные эксперименты с моделью подтвердили в целом медленную сходимость доверительного интервала с ростом числа испытаний. Как

и следовало ожидать, увеличение числа испытаний в 4 раза уменьшает неопределенность только вдвое. Для получения дисперсии в 10% нужно порядка 100 испытаний, а доверительный интервал обычно больше трех дисперсий. Интересно, что если после серии только верных ответов встречается один неверный, неопределенность для доли знаний D снова возрастает!

Итак, надежные выводы требуют длинных серий испытаний. Учитывая возможность чисто случайных ошибок из-за утомления испытуемых, это ставит под сомнение вообще целесообразность практического применения тестов с ответом, выбираемым из готовых, несмотря на их популярность. Нужны дидактически и статистически более совершенные тесты.

Г Л А В А 7. НОВЫЕ ТИПЫ ППС

7.1. Учебные диалоговые среды

Сравнительно новым типом ППС являются учебные диалоговые среды (ДС). Они предназначены для преодоления традиционной пассивности обучаемого и «переворачивают» отношение человек-компьютер: не компьютер учит ребенка, а ребенок учит компьютер [42]. Первой средой такого рода стал язык Лого, созданный С. Пейпертом еще в 60-е годы. Примерный диалог с программой может выглядеть так (для понятности использованы русские обозначения в командах).

Ребенок хочет нарисовать квадрат и печатает на экране слово «квадрат».

Программа понимает это как имя библиотечной процедуры и просматривает свою библиотеку. Не обнаружив в каталоге библиотеки слово «квадрат», программа сообщает «Не знаю, что такое квадрат». В ответ на это ребенок пишет:

```
Это квадрат:
нц 4 раз
вперед 1
направо 90
кц
```

Программа ищет синтаксические ошибки. Не обнаружив их, она сообщает: «Понятно, что такое квадрат» и помещает процедуру в библиотеку. Так строится библиотека и одновременно рисунок.

Основной объект языка – исполнитель Черепашка, понимающий команды «вперед», «направо», «налево», которые позволяют чертить на экране замысловатые ломаные линии. Удивительно легко получаются достаточно сложные и интересные узоры.

В наше время мир языка LOGO сильно обогатился [31]. Черепашка может принимать форму любой фигуры, нарисованной в текстовом редакторе, копироваться на экран, изображая, например, стаю летящих птиц. Черепашек стало несколько, и они могут взаимодействовать между собой. Можно сказать, что в Лого отражаются новейшие достижения «большой информатики» – взаимодействующие параллельные процессы. Возможно, этот язык сменит своего отдаленного потомка – Кумир, (который, увы, задержался в развитии) в качестве первого языка общения человека с компьютером.

Итак, рассмотрим общие признаки программ класса «диалоговые среды» (ДС). Из самого термина видно, что ДС – это программы, организующие для человека некоторую среду с заданными свойствами. Человек, воздействуя на эту среду, приводит ее в нужное состояние. Активен человек, программа лишь реагирует на команды. Фактически программа является инструментом деятельности и учащегося и профессионала. Для бухгалтера среда – это так называемая «пустографка», для математика – электронная таблица, для писателя – настольная издательская система.

При таком понимании ДС ими оказываются не только Лого и текстовый редактор, но и среда программирования типа Турбо Паскаль. Можно говорить не только о типе программ, но и о типе диалога с ЭВМ.

Становится теперь понятным, почему так живуче обучение программированию на уроках информатики. Ведь учебных диалоговых сред не так уж много, а работа с реальной ДС-компилятором интересна и полезна.

В начале эволюции Лого-технологии обучения С. Пейперт полагал, что богатство возможностей ДС и высокий уровень интереса автоматически обеспечат продвижение ребенка в освоении ее возможностей даже без заданий учителя. Практика показала, что это не так. Обучаемый не осваивает порой даже зону своего ближайшего развития без помощи учителя. В настоящее время принято поддерживать работу обучаемых в ДС системой заданий. Подчеркнем, что в самой среде их может и не быть, их привносит преподаватель.

Итак, даже традиционное изучение программирования при помощи современных сред типа Турбо оказывается, подобно говорению прозой мольеровского героя, весьма современным видом учебной деятельности, если иметь в виду стиль обучения и диалога.

7.2. Экспертные системы и АОС

Другим новым типом ПС стали экспертные системы (ЭС) [9], [45], [60]. По своему замыслу – это компьютерная модель опыта профессионала в какой либо области: медицина, химия, геология.

Признаком ЭС является развернутый диалог и возможность объяснения системой, почему она пришла к тем или иным выводам. Типичный

фрагмент базы знаний ЭС-модели врача, например, таков: «если есть насморк и температура небольшая, спросить о головной боли. Увеличить вероятность гриппа».

Известны впечатляющие примеры использования ЭС, например, в геологии, в оптимизации конфигурации продаваемых компьютеров. Все же надежды, возлагаемые на ЭС педагогами, пока не оправдались. Проблема заключается в сложности формализации опыта специалиста, преобразования его в систему строго упорядоченных взаимосвязанных вопросов, ответы на которые и позволяют поставить диагноз, открыть месторождение, поставить оценку за ответ и т.д. Чем формализованее область знания, в которой создается ЭС, тем проще ее разработать. Но, кстати, до сих пор нет ЭС, которые помогали бы искать смысловые ошибки в программах, и сапожники остаются, так сказать, без сапог. По причине сложности формализации остаются единичными примеры применения технологии ЭС в образовании. Ведь мыслительная деятельность педагога – это причудливая комбинация психологии, философии и других трудно формализуемых областей знания.

Тенденции в развитии экспертных систем – значительное расширение набора утверждений базы знаний. Есть мнение [45], что только при выходе числа утверждений системы на числа порядка миллиона система станет проявлять признаки человеческого интеллекта.

Продолжая аналогию с прозой мольеровского героя, заметим, что в преподавании информатики давно используются программы, имеющие некоторые черты ЭС. Это – компиляторы языков программирования. Они выполняют достаточно глубокий разбор входного текста (Бейсик с его лаконичным «Ошибка в строке 120» не в счет) и указывают (или даже предупреждают на всякий случай, как языки Perl или Си) на ошибки. Парадокс в том, что переход к диалоговой компиляции на ПЭВМ привел к упрощению компиляторов и утрате ими части, так сказать, интеллекта. Старые компиляторы, например, с Фортрана или Паскаля, и особенно с языка ПЛ-1, обнаружив ошибку, не успокаивались и пытались продолжить разбор программы, исправив найденную ошибку. Типичная фраза была такой: «возможно, данная ошибка является следствием предыдущей»...

Итак, снова есть довод в сторону полезности работы с компиляторами, т.е. изучения программирования, не дожидаясь появления специальных ЭС для целей образования.

7.3. Учебное компьютерное моделирование

Еще один сравнительно новый тип ППС – учебные модели. Этап персонализации привел к резкому росту графических возможностей ЭВМ и возможности имитировать на экране совершенно новые для информати-

ки процессы и явления [1], [3]. Сейчас компьютерное моделирование переживает пик роста.

Бесспорно значение компьютерных моделей в тех случаях, когда изучаемое явление протекает в микро или макромире, слишком быстро или медленно. Но следует предостеречь от скоропалительной разработки моделей там, где реальность доступна для непосредственного изучения. Автору встречалась моделирующая программа для изучения правила ... рычага! Ее разработчик потратил изрядную долю усилий на изображение (кстати, с помощью MSX-Бейсика) качающегося рычага и гирь. Но не полезнее и не проще ли для учащегося и учителя физики воспользоваться реальными гирями и рычагом и переоткрыть закон Архимедова рычага, взаимодействуя с реальностью? Итак, не все, что доступно, стоит моделировать.

Доводом в пользу моделирования такого рода может быть «очистка» реального явления от второстепенных факторов, изучение основной закономерности. Но умение выделить главное – это важнейшее умение того, кто строит и изучает реальное явление. Видимо, такое препарирование реальности нецелесообразно. Оно не отвечает и истории развития наук: ведь первооткрыватель не имел перед собой знание, закон в чистом виде, а выделял его из реальности. Так, сопротивление воздуха изменяло траектории легких пушечных ядер и долго мешало артиллеристам принять галилееву параболу.

Другое возражение против «бездумного» моделирования более серьезно. Модель всегда отличается от реальности, обладает собственной активностью и параметрами. При неточном моделировании обучаемому может быть навязана качественно неверная, с точки зрения реальности, картина поведения модели. Приведем и здесь реальный пример. Предположим, модель изображает падение элементарных частиц на экран некоторого устройства. Координаты точек попадания, естественно, случайны. А интервал времени между падениями отдельных частиц в модели, оказывается, – постоянный! Это грубая физическая и дидактическая ошибка, подкрепляемая авторитетом компьютера.

Обширное поле для моделирования – компьютерные игры. Здесь можно моделировать как существенные черты модели, например, движение взаимодействующих объектов с учетом масс и сил, так и внешнее оформление, изображение предметов средствам компьютерной графики. Пример упрощенной моделирующей программы приведен в разделе, посвященном компьютерным играм.

7.4. Перспективы развития ППС

Этап освоения возможностей ППЭВМ уже, в общем, завершен. В разработке АОС, продолжающих классическую традицию, все больше внимания уделяется деятельности обучаемого. После предъявления кадра

информации ему все чаще предлагается что-либо выполнить, как минимум нужно воспроизвести информацию, относящуюся к действиям.

Возможна разработка АОС графического типа, где единицей обучения явится уже не текстовый, а графический файл.

Продолжается быстрое продвижение диалоговой среды Лого в школы. Велик поток публикаций о педагогическом опыте ее применения, особенно в России.

Нарастают разработки моделирующих программ, все больше появляется моделей, основанных на глубоком понимании и интерпретации законов природы. Так, обучающая среда «Оптика», разработанная на кафедре физики МГПУ, основана на прямом моделировании распространения света в среде с переменными характеристиками; модели линз имеют реальную толщину.

Программы-модели распадаются на два класса: высокоточные имитационные для изучения по модели реальных явлений и простые открытые программы-модели для начального изучения собственно вопросов моделирования. Модели второго типа могут развиваться, усложняясь, в направлении первого.

ЧАСТЬ II. ВТ В УПРАВЛЕНИИ ОБРАЗОВАНИЕМ

Г Л А В А 8. АСУ В ОБРАЗОВАНИИ

8.1. Задачи АСУ

Автоматизированные системы управления (АСУ) пережили пик развития и интереса еще в 70-е годы. Постепенно было понято, что целью АСУ должно быть повышение качества деятельности лиц, принимающих управленческие решения на основания полной и точной информации. Компьютер не принимает решение, но обеспечивает его оптимальность. Компьютер также разгружает систему управления от рутинной работы с документами.

Примеры: для правильного построения кадровой политики в школьном образовании необходимо, в частности, знать долю учителей, преподающих информатику и не имеющих базового образования. Необходимо оптимально распределять ВТ по школам: в одном районе – один тип техники. Отсутствие обоснованной политики в это области привело к тому, что сегодня в одном районе можно встретить почти все типы школьных ЭВМ.

Приведем типы управленческих задач в образовании:

1) создание баз данных по кадрам учителей, обеспеченности учебной литературой, оборудованию школ;

2) поддержка управленческих решений в связи с результатами обучения, обработанными на ЭВМ;

3) поддержка рутинной управленческой деятельности: формирование типовых документов.

8.2. Уровни АСУ

В настоящее время АСУ существуют на разных уровнях и лишь постепенно объединяются в систему. Де-факто технической базой АСУ стали IBM-совместимые компьютеры, средством программирования – DBASE-совместимые языки, особенно FOXPRO. Сейчас отметим переход к среде Excel.

Приведем примеры задач, решаемых АСУ разных уровней.

1) АСУ «Школа». Предназначена для ведения баз данных по школьному оборудованию, кадрам учителей, подготовки документации.

2) АСУ «ВУЗ», подсистема – «Кафедра». Выполняет расчет нагрузки по кафедре в целом и по преподавателям, оценку нагрузки по кафедре в перспективе. Поддерживает ведение документации кафедры.

3) АСУ « ВУЗ – деканат». Выполняет ведение базы данных по студентам, оборудованию общежитий; ведение текущей документации: заявления, справки студентам, экзаменационные ведомости; ведение баз данных по учебным планам, особенно в условиях их нестабильности, перехода на новые планы.

4) АСУ «ОБЛУНО». Ведет базы данных по распределению школьного оборудования, кадрам учителей области для принятия кадровых решений: планирования повышений квалификации, заказ вузам на профили подготовки новых учителей, ведет текущую документацию.

5) АСУ «Министерство образования». Выполняет информационную поддержку стратегических решений о направлении и темпах развития сферы образования: переоснащение школ и Вузов оборудованием, динамика успеваемости по годам и дисциплинам, разработка сетевых графиков переоборудования школ, подключения областей к компьютерным сетям.

Подчеркнем, что фактически объединение АСУ выполняется снизу, по мере освоения компьютерных возможностей сотрудниками систем управления.

ГЛАВА 9. ВТ КАК СРЕДСТВО ПРОФОРИЕНТАЦИИ

9.1. Две задачи профориентации

Можно указать на две основные задачи профориентации, касающиеся учителей информатики: профориентация школьников в зависимости от профилей их способностей и склонностей, и частная задача – профориентация на педагогическую профессию.

Решение этих задач поддерживается программными средствами для исследования личности, всевозможными психологическими тестами. Роль учителя информатики состоит в предъявлении тестов и правильном их проведении во взаимодействии со школьным психологом.

9.2. Ситуации допроса и клиента

При любом (не только компьютерном) изучении личности следует различать две ситуации – ситуацию допроса и ситуацию клиента. Результаты тестирования в этих двух ситуациях могут отличаться радикально при одном и том же тесте.

В ситуации «допрос» или «обследование» испытуемый проверяется на объективное наличие или отсутствие у него определенных свойств и качеств для принятия решения, например, о наличии у него данных для профессии педагога. Испытуемый, зная о цели испытания и будучи заинтересованным в благоприятном исходе испытания, может давать неискренние ответы на вопросы анкеты. Решение на основании анкетирования может оказаться совсем неверным. (Такой случай имел место в Витебском педагогическом институте при компьютерном тестировании абитуриентов на наличие качеств, необходимых для того, чтобы стать учителем. Так, на вопрос, можете ли Вы ударить животное, ответ испытуемого был фактически предопределен и заранее очевиден: «конечно, нет!», и все абитуриенты «успешно» проходили тестирование).

В ситуации «клиент» испытуемый заинтересован в том, чтобы получить помощь от эксперта. Поэтому он искренне отвечает на вопросы, даже не очень приятные. Ситуацию клиента труднее создать, но пользы от такого теста будет гораздо больше.

Чтобы исключить неискренность тестируемого, применяется следующий прием: часть вопросов анкеты ставится так, чтобы искренний ответ на них был однозначным. Пример: «Иногда мне хочется выругаться». Испытуемый, естественно, не знает о наличии и номерах таких вопросов. Если он неискренне отвечает на них, выдается предупреждение и тестирование прекращается.

9.3. Факторные нагрузки вопросов

Чтобы правильно проводить тестирование, нужно иметь хотя бы начальное представление об устройстве тестирующих программ. Каждый вопрос теста связан обычно с несколькими числами (кроме, конечно, проверок на искренность). Каждое из этих чисел связано с исследуемой чертой личности. В простейшем случае бинарного опроса программа вычисляет сумму баллов, в более сложных тестах – несколько сумм. Пороговые значения для них строятся при шкалировании теста во время его разработки. При этом тест выполняется людьми, заведомо имеющими исследуемые качества, и на основании обработки их ответов вопросам анкеты приписываются векторы весов. Фактически анкета работает как классификатор: «распознает образ» анкетируемого и относит его к одному из классов со степенью уверенности, зависящей от суммы весов.

Г Л А ВА 10. ПРОВЕРКА ПЕДАГОГИЧЕСКИХ ГИПОТЕЗ

10.1. Роль статистики в управлении образованием

В практике учебного процесса школ и вузов нередко возникает необходимость оценить правомерность педагогической гипотезы. Но, увы, пока не редкость в отчетах с трибун утверждения типа: «Успеваемость за год возросла на 1.3% и составила 76.5%». И это говорится применительно к группе из 20-25 человек. У внимательного слушателя такого отчета должен возникнуть резонный вопрос: является ли это «улучшение» достоверным, нет ли здесь случайного колебания? Для корректного ответа на этот вопрос нужно привлечь прикладную математическую статистику. Без нее такой вывод является абстрактным и в целом спорным, и не является базой для управленческого решения.

Полагаем, что привлечение аппарата прикладной статистики может и должно резко улучшить качество управленческих педагогических решений на всех уровнях АСУ. Роль учителя информатики – научить коллег, завуча и директора пользоваться этим аппаратом, взятым в готовом виде.

Математическая статистика в целом – это вполне «высшая» математика и изощренная, нередко фактически многозначная логика. В то же время область ее приложений – повседневная реальность, а на основании ее рекомендаций могут приниматься важнейшие для общества и образования решения.

Поэтому, и в духе пользовательского курса информатики, можно говорить и о пользовательской статистике. Это постоянно реализуется в ряде вузовских нематематических специальностей. Нередко, например, на биологических специальностях проводится чисто рецептурный подход: как

вычислить, что, куда подставить, с чем сравнить. Фактически и логический вывод уже навязан такой процедурой-алгоритмом, хотя компьютер может и не привлекаться. Но некоторые концептуально важные моменты могут остаться за кадром. Особенно это относится к понятию нуль-гипотезы и логическим операциям с ней.

Анализ практики управления образованием показывает, что основной объем мыслительной деятельности лица, принимающего решения проходит на качественном уровне, без числовой информации (вывод о росте успеваемости, приведенный выше, конечно, не в счет).

Применение компьютера может изменить ситуацию, убрать рутинную часть, вычисления и остановиться на важном: педагогических гипотезах и их проверке. Наблюдение, измерение и его результат – число может и должно быть полноценным, весомым элементом деятельности управленца и педагога.

В качестве важного результата адаптированного введения компьютеризованной обработки данных в практику управления мы предполагаем не абстрагирование, отвлечение от реальности, а наоборот, большую практичность, основанную на возможности немедленного сопоставления данных наблюдений с гипотезой. Компьютер, снимая проблему расчетов, окажется мостиком к реальности, и реальности совсем не виртуальной. Ведь в средние века именно практика (увы, игра в кости и карты) породила теорию вероятностей.

Поэтому возможные возражения специалистов против такого введения элементов статистики должны учитывать и важность цели, изменения стиля управленческих решений: от практики – к теории. Сама процедура статистического вывода ни в коем случае не должна затенять основное содержание проверяемой гипотезы. Поэтому вводятся лишь те понятия статистики, которые совершенно необходимы.

10.2. Введение основных понятий

Итак, необходимые понятия статистики вводятся «без нажима» на математику и определения, на примерах. В данном анализе примеры берутся в основном из области деятельности учителя: оценки и контроль знаний.

Понятия вероятности, случайного события можно неформально ввести на классических примерах (бросание монеты), но опираясь и на современное и понятное представление о симметрии в самом широком смысле. (Тяжелые споры о субъективной и объективной вероятности оставим специалистам).

Работа со статистическим материалом начинается с выдвижения гипотезы, например: успеваемость по математике повысилась. Но проверяется обычно не положительное утверждение, а его отрицание – нуль гипотеза.

Нуль-гипотезу можно понимать как гипотезу об отсутствии эффекта, связи, зависимости и т.д. Пример нуль-гипотезы:

1. Данный ученик при опросе отвечает наугад: ответы да/нет равновозможны. Хотелось бы убедиться в обратном.

2. Успеваемость у двух преподавателей математики не различается.

Обычно «тайный» замысел автора нуль-гипотезы – убедиться в ее ошибочности, т.е. обнаружить эффект (и доложить об этом в научной или школьной печати). Метод нуль-гипотез логически сходен с доказательством от противного.

Возможная аналогия: следователь должен исходить из невиновности человека и попытаться это опровергнуть, а адвокат наоборот! – искать опровержения гипотезы о виновности. Нуль-гипотеза – это позиция этичного и логичного следователя.

Отказ от нуль-гипотезы в нашем случае резонно выполнять на основе односторонних критериев. Для практики, тем более школьной, часто важно не просто отвергнуть нуль-гипотезу, а указать эффект, связь, знак этого эффекта. Так, явно мало отвергнуть гипотезу об одинаковой успеваемости у двух учителей. Первый же резонный вопрос завуча – у кого успеваемость выше.

Понятие «статистика» (как число, параметр) вводится как числовая мера отклонения данных опыта от того, что ожидается, если верна нуль-гипотеза. Оно конкретизируется как понятие и вычисляется программой лишь тогда, когда имеет ясный смысл: средний номер или сумма номеров (рангов) наблюдений в ранговых критериях, доля верных ответов при серийном опросе и т.д.

Понятие «уровень значимости» можно и не вводить, оставив его на статистической «кухне». Заменим его весьма близким, но более простым и содержательным термином: вероятность допустить ошибку, если отказаться от нуль-гипотезы на основе имеющихся данных.

По возможности избегаем и использования статистических таблиц. Если предъявить на экране только статистику-число, то с ним еще надо что-то делать, сравнивать с табличным значением по самым разным правилам. Гораздо проще и понятнее получить от компьютерной программы сразу же «живую» вероятность, отвечающую этому параметру-статистике. Имея компьютер, стоит вообще избегать таблиц. Пора таблиц синусов уже миновала.

Предпочтение отдаем непараметрической статистике. Это позволяет избежать опоры на нормальный закон распределения, который, как и положено вероятностному закону, нигде и никогда точно не выполняется, но на который, тем не менее, всегда ссылаются. Непараметрические методы более просты для понимания, и нередко являются и более общими по области применения. Их устойчивость по отношению к грубым ошибкам (выбросам) особенна ценна для школьной практики.

Анализ числового материала, связанного с гипотезами об успеваемости школьников и студентов, показывает, что для большинства приложений можно ограничиться проверкой следующих статистических гипотез (для каждой из них предлагается использовать один критерий).

10.3. Гипотеза о виде распределения

Гипотеза формулируется в виде равновозможности двух исходов, и для ее проверки используется критерий знаков.

Критерий знаков – один из простейших. Это – крепкий мостик для учителя и управленца, входящего в прикладную статистику и одновременно хороший практический критерий. Он вполне прозрачен, вероятность легко вычисляется, а при малом числе наблюдений – вообще без ЭВМ.

Пример нуль-гипотезы: рождения мальчиков и девочек – равновозможны. Но остановимся на работе с этим критерием в связи с другой, тоже реальной, но уже педагогической ситуации.

Итак, идет компьютеризованный опрос: ответы учеников «да (нет)» на 5 вопросов. Полное число возможных исходов такого опроса:

2 * 2 * 2 * 2 * 2 = 32

– всевозможные комбинации из «да» и «нет» длиной в пять слов.

Пусть ученик ответил правильно на все 5 вопросов из 5. Это значит, что исход его опроса – всего один из 32 возможных, и других подобных исходов нет. Резонное намерение – поставить ученику пять. Но для этого учитель сначала выдвигает нуль-гипотезу: «ученик отвечает наугад». Тогда число правильных и неправильных ответов должно быть равно примерно(!) 2.5, отличаясь на случайную величину. При полученном исходе опроса вероятность учителю ошибиться, отбросив нуль-гипотезу о гадании, составляет всего 1/32= 3%. Итак, гипотезу о гадании можно спокойно отбросить и принять, что этот ученик не гадает, но неизвестно, что доля его знаний >0.

Пусть другой ученик дал лишь один верный ответ из 5. Это значит, что из всех 32 равновозможных исходов опроса им реализован один из шести с таким же или меньшим числом верных ответов, (1 – да, 0 – нет). Возможные исходы с одним верным ответом таковы:

10000, 01000, 00100, 00010, 00001.

К этим 5 исходам добавляем (!) еще и исход 00000 – ни одного ответа верного. (Так строится критическая область критерия, кстати, в данном случае – в пользу ученика).

Итак, целых 6 из 32 исходов говорят в пользу гадания, т.е. нуль-гипотезы. Отбрасывать ее пока рано, придется продолжить опрос, и возможно – до бесконечности...

Пусть третий ученик не дал вообще ни одного верного ответа из 5. Но не будем спешить с оценкой. Ведь это значит, что снова произошло редкое событие: вероятность такого исхода при случайном гадании равна лишь 1/32. Значит, снова логично отказаться от нуль-гипотезы, как и в случае отличника, но уже в другую сторону: ученик не гадает, а отвечает строго наоборот (по подсказкам недоброжелателя или сам, но назло учителю, – это уже нестатистические гипотезы).

Теперь поставим вопросы: какой оценки заслуживает ученик, знающий ровно половину материала? Ученик, дающий три четверти верных ответов? Это одно и то же или нет?

Это знакомая практику-педагогу ситуация: ученик знает ровно половину материала. Но тогда половина его ответов будет правильной, а половина – случайной, «наполовину» правильной. Итого – около 3/4 (!) правильных ответов, например 4 из 5 или 6 из 8. Ну так и хочется поставить четверку, а то и с плюсом...

10.4. Гипотеза об однородности групп наблюдений

Весьма типична задача сравнения двух групп наблюдений: сравнение успеваемости у двух классов или у разных учителей. Полезный и весьма общий критерий для этого – критерий Манна-Уитни (Уилкоксона) [30]. Важно, что численности этих групп могут отличаться.

Полагаем, что проверку на однородность двух групп измерений не стоит всегда сводить «по старинке» к сравнению средних значений, отнесенных к дисперсиям. Критерий Манна-Уитни тоже полно учитывает имеющуюся информацию, хотя ничего не «знает» о законе распределения величин, и может применяться к самым разным данным: размеры, экспертные (школьные) оценки, частоты, проценты и т.д. Нормальный закон здесь не нужен.

Для понимания критерия достаточно усвоить следующую идею. Если две группы наблюдений-чисел однородны (нуль-гипотеза), то в объединенной, упорядоченной и пронумерованной последовательности всех чисел средний номер (ранг) для элементов каждой подгруппы должен быть близок к середине.

Тогда станут понятны и знак, смысл и величина большого различия этих средних номеров: одна группа наблюдений смещена относительно другой влево или вправо. Значит, данные неоднородны и это противоречит нуль-гипотезе. Тогда можно «с удовольствием» ее отбросить и сказать: эффект есть – первый учитель ставит оценки выше второго. О знаниях здесь речи не идет.

Для принятия такого решения компьютерная программа должна выводить не только статистику, но и вероятность такого, как в опыте, или большего отличия средних номеров групп.

Критерий легче понять, а вероятность вычислить при малом числе наблюдений.

Пусть оценки 3, 3, 2, 3 ученик получил по информатике в первой четверти, а оценки 4, 4, 5 – во второй. Можно ли утверждать, что он стал лучше учиться? Ответ учителя очевиден, но что скажет статистика? Итак, нуль-гипотеза: разницы в оценках нет, ученик учится по-прежнему.

Объединенная серия «наблюдений» выглядит так (буквы отмечают первую группу А или вторую – В):

Наблюдения 2, 3, 3, 3, 4, 4, 5
Группы A A A A B B B
Номера (ранги) 1 2 3 4 5 6 7

Средние номера: SA = (1 + 2 + 3 + 4) / 4 = 2.5
SB = (5 + 6 + 7) / 3 = 6

Такой или больший средний номер для второй серии может быть получен здесь лишь одним способом: при расположении всех чисел второй серии в конце, справа. Всего же можно составить 7*6/2=21 серию, отличающиеся положением чисел второй группы в общей серии, например:

A B A A B A A
B B A A A A A
....

Итак, в пользу нуль-гипотезы говорит вероятность лишь 1/21 = 0.048. Учитель прав: оценки, (а может быть и знания) стали лучше.

10.5. Гипотеза о независимости. Сопряженность

Для оценки связи пары величин, каждая из которых принимает два значения, есть простой и наглядный метод – таблицы сопряженности (критерий хи-квадрат).

Этот метод интересен и важен тем, что позволяет судить о связи, о зависимости или ее отсутствии. И здесь для таблицы 2*2, кроме статистики хи-квадрат, и даже вместо нее, программа должна вычислить вероятность по точной формуле Фишера [30] и знак связи. Про хи-квадрат можно и не упоминать.

Для понимания метода достаточно усвоить идею, что при независимости двух признаков, строки (или, что то же, столбцы) таблицы должны составлять «почти» пропорцию.

Пример: рассмотрим результаты контрольного среза успеваемости в двух районах (данные условные). Нуль-гипотеза: результаты контрольной работы по районам не отличаются.

число оценок учащихся оценка:	положительных	неудовлетворительных	всего
первый район	111	41	152
второй район	130	12	142
всего	241	3	294

Вероятность такого или менее вероятного исхода – 0.00002.

Вывод: нуль – гипотеза неверна; результаты районов существенно отличаются.

Здесь стоит обратить внимание на логическую тонкость, весьма важную. Если имеется некоторая гипотеза и факты ей не противоречат, это еще не значит, что гипотеза верна. Правильное утверждение: гипотеза не противоречит фактам.

10.6. Гипотеза о независимости. Корреляция

Если данные наблюдений за двумя параметрами – величины, а не частоты, как в предыдущем примере, можно рассмотреть связь как корреляцию этих величин. Исходные данные – набор пар наблюдений.

Можно указать сначала на сходство с математикой: корреляция – это зависимость, связь величин. И затем указываем на различие: связь эта – не жесткая, не однозначная, а случайная.

Не стоит упрощать дело в духе идей XVIII–XIX веков таким образом, будто «на самом деле» связь эта однозначная, как в математике или астрономии, а злополучные случайности ее лишь затемняют, но аккуратно складывают свои влияния в форме нормального закона. Бывает и так, что связь вообще может проявиться только через ряд случайных событий; и случайность – не мешающий фактор, а неотъемлемая часть такой зависимости.

Пример. Успеваемость в учебе отцов связана с успеваемостью детей. (Нуль-гипотеза – связи нет!) Ясно, что на это влияет масса факторов, и успеваемость отца – лишь один из них. Нет в организме отца «точной формулы» для успеваемости сына, более того, оценки самого отца тоже получились такими в результате действия множества причин.

Поэтому чаще встречается логическая симметрия корреляционной связи: не Y является функцией X, как в математике, а X и Y просто связа-

ны, изменяются совместно. К тому же в природе сплошь и рядом имеются обратные и кольцевые связи. Так, волки едят зайцев, но их число – не «аргумент функции»: численность зайцев обратно влияет на численность волков. Всеобщим аргументом не является и время: – ему, времени, вообще все равно, кто кого ест.

Итак, понятие связи здесь значительно богаче, чем в привычной школьной математике.

Ранговый критерий Кендалла позволяет, как и ранее, избежать знаний о распределении и проверить связанность величин любых типов. Так, например, оценки по физкультуре за прыжки в длину – целые, а рост учащегося – вещественные числа. Сам коэффициент корреляции зависит от числа наблюдений и значение, скажем, 0.22 может показаться малым по сравнению с единицей. Поэтому для оценки нуль-гипотезы снова необходимо вывести на экран ПЭВМ вероятность ошибки отказа от гипотезы о независимости величин, и знак связи при наличии зависимости.

Для понимания сути этого критерия достаточно следующего. Пусть наблюдения одной из групп упорядочены и занумерованы, наблюдения второй группы – переставлены в связи с первыми: пары сохранены. Тогда номера парных наблюдений при отсутствии связи полностью случайны, «перемешаны», при полной зависимости – полностью упорядочены. Если связь положительная, то порядок номеров второй группы – прямой, при отрицательной связи – обратный. Мерой перемешанности является количество перестановок номеров, которое нужно сделать, чтобы восстановить порядок и во второй группе наблюдений. Вычисляет это, конечно, программа.

10.7. Средства реализации статистических алгоритмов

Наряду со школами, хорошо оснащенными компьютерами, гораздо больше школ с маломощной вычислительной техникой. Им недоступны стандартные, профессиональные пакеты по математической статистике. Но, с другой стороны, даже студенту непросто начать диалог с подобным пакетом, который хорошо «нагружает» пользователя терминологией, окнами, помощью, справками и так далее. Поэтому началу использования ЭВМ вне информатики может помешать не только слабость наличной школьной вычислительной техники или программных средств, но, как ни странно, и их мощность.

Полагаем, что оптимальное (пока) решение – реализация рассмотренных критериев простейшими, общедоступными программными средствами. Это – все тот же, вездесущий Бейсик, (теперь уже и VISUAL-BASIC), либо электронные таблицы и такие простые программы-критерии в этих средах, которые выдают только понятные числа и термины. Знать программирование пользователю программы – педагогу или завучу – не

нужно. Учителю информатики достаточно знать, как вести диалог с программой. Приведем пример – программа для расчета доверительной вероятности в случае критерия знаков.

```
? «К Р И Т Е Р И Й З Н А К О В» '
ML = 500                                'Максимум исходов
DIM AL(ML)                              'Таблица логарифмов
AL(ML) = 0                              '0! = 1
FOR I = 1 TO ML                         '
AL(I) = AL(I - 1) + LOG(I)              'Логарифмы факториалов
NEXT I '
?»Сколько исходов первого вида»;  '
INPUT P                                 'число исходов вида 1
?»Сколько исходов второго вида»;   '
INPUT M                                 'число исходов вида 2
N = P + M                               'всего испытаний
S = 0                                   'сумма вероятностей
IF P < M THEN G = P ELSE G = M   'выбор хвоста распр-ния
Z = LOG(2) * N                          'ln(2^N)
FOR I = 0 TO G                          '
C= AL(N) - AL(N - I) - AL(I)            'показатель для EXP
S = S + EXP(C - Z)                      'сумма C(N, G) / 2^N
NEXT I                                  '
?»Вер-сть такого отклонения=«; S '
?»Вывод: исходы равновозможны?» 'нуль-гипотеза
END
```

10.8. Цели применения статистики в управлении

Теперь, обозначив группу простых статистических критериев, их методически оптимальную реализацию на ЭВМ, и основные приемы учебно-исследовательской работы с ними, вернемся к целям привлечения статистики к выработке управленческих решений.

Для уверенности в себе педагогу и управленцу достаточно понимать, как в принципе оценивается некоторая величина. То, что компьютер сделает это быстрее и без ошибок, их вполне устроит, без ущерба для самооценки. Гораздо важнее, что логический вывод из расчета делает человек сам, а не машина за него.

Полагаем, что в результате освоения элементов статистики в связи с проверкой педагогических гипотез необходимо:

1) научить управленца логически корректно, и четко во внешней речи формулировать утверждения о свойствах данных наблюдений в виде гипотез;

2) научить его оперативно и точно проверять эти гипотезы-утверждения, вычисляя при помощи готовых программ вероятность ошибки при отбрасывании гипотезы; правильно строить выводы;

3) формировать у учителя и управленца интерес к такой исследовательской деятельности.

При этом не ставится цель немедленного пересмотра форм и методов управления учебным процессом. Это – дело времени, практики. Полагаем более важным усиление научности управления, рост интереса лиц, принимающих решения к информатике, к научным методам принятия решений.

ЧАСТЬ III. ЭВМ И УНИВЕРСАЛЬНЫЕ ВИДЫ ДЕЯТЕЛЬНОСТИ

Г Л А В А 11. СЧЕТ НА ЭВМ И МК

11.1. Место счета в ряду умений человека

Вычисления – исторически первый способ использования ВТ. В наше время небольшие вычислительные задачи часто решаются на микрокалькуляторах (МК). Даже в мощных средах типа WINDOWS, FOXPRO имеется модель калькулятора. Поводом для счета на калькуляторе может быть и момент интеллектуального соревнования: «на Бейсике считать и глупец сможет». Кроме того, по стилю деятельности человека, программирующего на ЭВМ и МК, близки. Это – преодоление ограничений на память, количество вводимых данных, поиск более короткого алгоритма и т.д. МК позволяют ставить и решать такие творческие задачи. Работа с МК может быть пропедевтикой программирования.

11.2. Характерные черты счета

Счет – очень древняя деятельность человека вообще. Ее характерные черты таковы.

1. Массовость и повседневность.

2. Сложность и громоздкость счета без ЭВМ из-за неудачной, десятичной системы счисления. Можно предположить, что если бы у людей было по 4 пальца на руках и ногах, двоичную систему и компьютеры они изобрели бы на пару веков раньше.

3. Эта деятельность абсолютно формальна и легко автоматизируется.

Компьютер обеспечивает скорость и точность вычислений. Распространение микрокалькуляторов привело к ослаблению навыков устного

счета. Внимание вычислителя и его мышление смещается от пооперационного выполнения действий к оценке правдоподобности результатов проводимых вычислений. Существует опасность слепого доверия к компьютерным вычислениям, сопровождающимися не только ошибками округления, которые, как правило, малы, но и просто «опечатками» – неверным вводом исходных данных. Поэтому при расчете на МК возникает новая для информатики и математики интересная задача – преобразование выражений к виду, позволяющему вводить данные для счета однократно.

11.3. Требования к ВТ для вычислений

Можно сформулировать ряд требований к средствам для счета. Эти требования могут быть различными в учебных и профессиональных вычислениях.

1. Естественность записи и ввода формул и алгоритмов. Здесь следует указать на необходимость поддержки старшинства операции, алгебраичность записи формул в учебных вычислениях на МК инженерного типа. С другой стороны, обратная польская запись формул на МК-54 близка к доалгебраическим вычислениям младшего школьника: сначала пишутся (вводятся) числа и только после этого выполняются действия. В этом смысле польская логика МК-54 выглядит вполне естественной.

2. Простота логики, отсутствие исключений из правил. Антипримером мог быть все тот же МКШ-2: вычисление функции при открытых скобках портило числа в скрытых регистрах без всякого предупреждения. И это происходило на школьном МК!

3. Сохранение результатов при сбоях питания. Увы, единственным современным (и древним) устройством такого рода являются конторские счеты – устройство сохраняет введенные числа, пока не будет выполнена команда «сброс» резким наклоном кисти.

4. Надежность ввода. Некоторые МК «попискивают» при нажатии клавиш, но некоторые дублируют вводимую цифру при задержке нажатой клавиши. Наиболее надежен и удобен для исправлений ввод на компьютере.

5. Наличие памяти. На МК с регистром памяти можно выполнять достаточно оригинальные и краткие алгоритмы, сокращать количество моментов ввода данных.

6. Однородная точность вычислений. Все функции должны вычисляться с числом разрядов, равным разрядности чисел.

11.4. Некоторые эффективные приемы работы на МК

Использование клавиши С (сброс) при вводе чисел на арифметических МК позволяет сохранить результаты при ошибках ввода чисел – нажатиях клавиш-цифр.

Пусть надо вычислить 34*56-78. Составим таблицу шагов вычисления:

Клавиши Комментарий
34
x
57 ой, ошибка, надо было 56
C сброс числа
56 правильный ввод
-
77 ой, ошибка, надо было 78
C сброс
78 правильный ввод
=

Рассмотрим исправление ошибок при вводе операций на том же примере.

34
+ ой, не то нажал, надо было умножить
0 так исправить проще всего
*
56
/ ой, не то нажал
1 так исправить проще всего
-
78
=

Использование режима константы. При нажатии клавиши «=» (равно) арифметический МК запоминает последнее действие и последний операнд. Повторное нажатие клавиши «равно» повторяет эту операцию. Так легко вычислять члены геометрической прогрессии:

3 * 4 = = = = = = =

11.5. Моделирование с помощью МК

Заметное упрощение математики, которое происходит при компьютерном моделировании, позволяет порой применять здесь простые МК. Так, известное уравнение для массы вещества в условиях радиоактивного распада при переходе от дифференциального уравнения

dm/dt = -k * m

к дискретной конечно-разностной модели позволяет вообще не упоминать о производной:

(m2 - m1) / dt = -k * m1.

Здесь m1 – предыдущее значение массы, m2 – новое значение.
Отсюда следует рекуррентная формула:

m := m * (1 -k * dt)

Далее дело сводится к вычислению на МК геометрической прогрессии:

m * (1 - (k * dt)) = = = =

Здесь нажатие клавиши «=» переводит МК в режим константы – умножения на выражение 1 - (k * dt). При каждом нажатии клавиши «равно» для модели проходит конечное время dt, и можно снимать с индикатора МК текущее значение массы. Компьютер не нужен.

11.6. Двухпроцессорные вычисления на арифметическом МК

Освоение возможностей МК может быть хорошей школой для развития комбинаторного мышления. Приведем пример реализации на МК с регистром памяти «двухпроцессорного вычисления». Нужно вычислить параллельное сопротивление:

R = R1 * R2 / (R1 + R2)

Требование однократности ввода данных ведет к преобразованию формулы к виду:

R = 1 / (1 / R1 + 1 / R2)

Если же на данном МК нет обратной функции «F1/X», но есть клавиша «П+», то для избежания повторного ввода R1 и R2 (ввод данных – источник ошибок при счете на МК!) можно выполнить два переплетающихся вычисления: умножение и деление выполнять над числами на индикаторе, а сложение R1 и R2 выполнить в памяти:

СП R1 П+ * R2 П+ / ИП =

Рассмотрим этот процесс по шагам.

Команда	Комментарий
СП	очистить память: П := 0
R1	ввести на индикатор R1
П+	прибавить R1 к нулю в памяти. П := П + R1
*	скопировать R1 в рабочий регистр
R2	ввести R2 на индикатор
П+	П := П + R2 (П = R1 + R2)
/	умножить на индикаторе RX:=RX*R2
ИП	вызвать сумму из памяти на индикатор
=	делить произведение на сумму

Заметим, что подобные изощренные, переплетающиеся вычисления можно выполнять и на языке Си (команды типа «j = i++; «). Тем самым освоение этой возможности на МК может быть пропедевтикой для освоения изюминок языка Си.

Используем тот же прием переплетения для вычисления суммы геометрической прогрессии. На индикаторе будем получать члены суммы, а в регистре памяти накапливать сумму:

Команда	Комментарий
СП	Приготовиться к суммированию
A	начальное значение
П+	добавить в память
*	
Q	множитель
=	режим константы
П+	s:=s+a
=	a:=a*q
П+	s:=s+a
=	a:=a*q
...	и т.д.

11.7. Программные средства для диалоговых вычислений

Встроенные в диалоговые среды МК могут быть полезны, если нужно дать контролирующей программе числовой ответ. Все же при работе на компьютере модель МК – известная дань консерватизму пользователя. Удобнее выполнить диалоговое вычисление, конечно, с помощью интерпретатора типа BASIC или FOXPRO.

Укажем на диалоговый режим языка Бейсик. Имеется в виду интерпретаторы: старый Бейсик- MSX или GW-Бейсик или даже Вильнюс - Бейсик. Нередко эта возможность недооценивается и интерпретатор языка привлекается сразу для программирования. Диалог с программой может выглядеть так (комментарии справа можно не вводить):

```
A = 3                  'обозначим сторону через A
 Ok                    'готово
B = 4                  'вторая сторона
 Ok
C = 5                  'третья сторона
 Ok                    '
P = (A + B + C) / 2 'полупериметр треугольника
 Ok                    '
S = SQR(P * (P-A) * (P-B) * (P-C)'площадь по формуле Герона
?S                     'Выдать ответ на экран
6
Ok
```

Возможность обозначать величины буквами делает такой «калькулятор» весьма мощным и удобным.

Заслуживает внимания диалоговый режим сред типа DBASE. Здесь в диалоговом режиме есть две важных возможности:

1) команда DISP MEMO выводит на экран все переменные памяти;

2) команда ???A выводит значение переменной A не на экран, а сразу на принтер.

Разработчики компиляторов типа Турбо Паскаль предусмотрели диалоговый счет с помощью команды Ctrl F4. Правда, в этом режиме, без программирования, нельзя создать переменные.

В случае освоения обучаемыми сред интерпретирующего типа весьма полезно сначала выполнить задание на диалоговый счет с помощью интерпретатора, без програмирования.

11.8. Сравнение различных средств для вычислений

Приведем пример, показывающий возможности различных средств для вычислений.

Известно, что exp(x)=1 + x + x^2/2! +x^3/3! +...

Будем суммировать этот ряд до предельной точности, пока член ряда не обратится в машиный ноль.

```
'Вычисление EXP(X) на Бейсике
 cls
 s=1                   'сумма
 a=1                   'член ряда
```

```
 i=1                  'номер члена
 x=2.3                'аргумент
while a<>0            'нц пока a<>0
        a=a*x/i
        s=s+a
        i=i+1
wend
?»Сумма=«; s, «точное exp=«, exp(x)

// Вычисление EXP(X) на C
#include <stdio.h>
#include <math.h>
main ()
{                  // переплетающиеся вычисления
float s=1;     // сумма
float a=1;     // член
float i=0;             // номер
float x=2.3;           // аргумент
while (a! =0)          // нц пока A<>0 {
        a=a*x/(++i); // опережающее присваивание i=i+1;
        s+=a;          // s:=s+a на Си
}
printf(«i=%d\n», i);
printf(«Сумма=%20.10g\n», s);
printf(«Точное значение=%20.10g\n», exp(x)); }
```

Инженерный калькулятор с памятью:
Пусть в регистре памяти находится x. Тогда по схеме Горнера:
ИП/8+ИП/7+ ИП/6+ИП/5+ ИП/4+ИП/3+ ИП/2+ИП/1+1=

11.9. Области применения счета вне информатики

Отметим, что счет вовсе не является «собственностью» информатики. Информатика все более занимается нечисловыми применениями компьютера, передавая счет специальным дисциплинам. Вот некоторые примеры:

– расчет погрешностей физического эксперимента;

– задачи с числовыми ответами в математике и физике;

– моделирование;

– исследование графика функции по точкам;

– проверка правильности тождественных преобразований, выполненных вручную;

– оценка значений пределов последовательностей и функций подстановкой близких к предельным значений (пример: sin(x)/x, x=0.01).

Этот ряд примеров легко продолжить.

Г Л А В А 12. ЧТЕНИЕ И ПИСЬМО ЗА ЭВМ

12.1. Место чтения и письма в ряду умений человека

Чтение и письмо – ровесники счету как виду человеческой деятельности. Их характерные черты таковы.

1. Массовость и повседневность. Пишут и читают сегодня практически все.

2. Сравнительно невысокая степень автоматизации письма. Студенты на лекциях и школьники на уроках продолжают писать шариковыми ручками. Отсюда – низкая скорость письма и чтения рукописного текста. В перспективе следует ожидать оснащения рабочего места студента и школьника клавиатурой, если не ПЭВМ.

3. Полная формализуемость письма, возможность его компьютеризации.

Компьютер дает здесь легкость ввода: нажать клавишу проще и быстрее, чем выписывать, рисовать букву. Легко и исправлять ошибки. Но компьютер и отнимает навыки письма. Почерк хакеров, компьютерных фанатов, испорчен уже сегодня. По мере распространения клавиатур возможен и своеобразный регресс: грамотный, быстро, десятью пальцами печатающий человек сможет лишь с трудом, как ребенок, нарисовать свои инициалы. Мышление человека, работающего с текстом за ЭВМ, смещается к организации этого текста, его структуре.

12.2. Требования к компьютерным средствам

1. Наличие алфавита на родном языке. Вообще многообразие алфавитов создает массу проблем разработчикам компьютерной техники. Даже в странах с латинскими алфавитами всегда имеется пара-тройка «национально особенных» букв, отсутствующих на клавиатуре. Обойтись без них, или заменить, естественно, никак нельзя, ибо это сильно ущемляет суверенитет. Возможно, выходом станет переход к коду Unicod.

2. Надежность ввода. При массированном вводе текстов так называемый дребезг клавиш недопустим. Аппаратура должна поддерживать десятипальцевый метод ввода.

3. Скорость ввода символов должна ограничиваться только возможностями человека.

4. Достаточный объем памяти. Для ввода обычно памяти достаточно, но при быстром перемещении по читаемому тексту требования к памяти повышаются. На ППЭВМ ее достаточно.

5. Вывод на принтер. Это совершенно обязательное для профессионала условие важно и в школьной практике: важно удержать интерес, этому и служит вещественная распечатка школьником результатов своего труда.

6. Принцип непосредственного редактирования. Программа-редактор должна поддерживать у человека иллюзию, что он редактирует непосредственно файл на диске: никаких рабочих текстов, буферов и т.д. Не следует умножать сущности сверх необходимого (принцип бритвы Оккама).

7. Перенос фрагмента по документу и в другой документ. Это одна из самых распространенных операций при ручном и компьютерном редактировании.

8. Наличие управления макрокомандами. Если для профессионала это желательное удобство, то для школьника и студента – возможность управлять редактором как исполнителем, пропедевтика алгоритмизации. Умение составлять и использовать макросы – признак компьютерной образованности.

9. Работа с ОС. Сегодня де-факто это уже рядовая возможность в среде WINDOWS. Достаточно нажать, например, Ctrl + Esc.

10. Чтение нетекстовых файлов необходимо только профессиональному программисту для работы с информацией на физическом уровне.

11. Гипертекстовые переходы. Пока что это не стало повседневностью, хотя еще в 1987 году школьный текстовый редактор MIM позволял это делать.

12. Одновременная работа с несколькими текстами совершенно необходима для профессионала и интересна для школьника.

13. Наличие математических символов необходимо для тех, кто пишет статьи и учебники.

14. Откатка – отказ от изменений, прямо следует из принципа непосредственного редактирования. Ведь так легко нажать не ту клавишу. Откатка также необходима в начальной стадии освоения редактора, когда делается много ошибок.

15. Форматирование – типично издательская операция, которую, кстати, не так-то просто запрограммировать. Эстетическое воздействие форматирования на учащегося очевидно. Спорным является применение форматирования с переносом. Перенос затрудняет быстрое чтение.

16. Средства поиска по тексту заслуживают особого внимания, так как это одна из типично компьютерных возможностей ускорения работы с информацией.

17. Защита от изменений нужна при работе с учебными текстами, когда внесение изменений может быть случайным. Если же речь идет о по-

луфабрикате конспекта, который рассылается обучаемым, изменения должны, конечно, допускаться.

18. Отражение сложной структуры данных (разделы, главы, страницы). Это обязательное для профессиональной работы качество среды полезно и школьнику, так как заставляет думать над структурой документа.

12.3. Общая схема диалоговых программ

Для понимания принципов работы текстового редактора создадим предельно упрощенный аналог. Но, прежде чем рассматривать устройство простейшего текстового редактора, укажем на общую схему диалоговых программ:

```
нц
        ввести команду
        выбор
                при команда = к1: делать серию 1
                при команда = к2: делать серию 2
        при ...
        все
кц
```

Смысл ее прост. В бесконечном цикле вводится очередная команда. Делается соответствующая ей серия, и цикл повторяется.

Закрепим это на примере шуточной программы «Супербейсик». Она на все действия пользователя отвечает как и положено: «Ок». На Паскале она выглядит так:

```
program superbasic;      {модель SUPERBASIC}
var s:string;            {для ввода }
begin
 repeat                  {нц }
 readln(s);              {ввести строку }
 writeln('Ok');          {Это – Бейсик }
 until false             {кц }
end.
```

Почти то же самое на Бейсике выглядит так:

```
'BASIC.ASC 'Супербейсик
10 ?                          'нц
 ?»Ok»                        'Главная выдача
30 A$ = INPUT$(1)             'Взять 1 байт с клавиатуры
 IF A$ = CHR$(13) THEN 10 'конец строки
 ?A$;                         'текущая выдача
GOTO 30                       'кц
```

Конечно, можно усилить сходство с Бейсиком, но для шутки и этого достаточно.

12.4. Китайский текстовый редактор

Рассмотрим, как устроен редактор на примере китайского – символы выводятся в столбик. Это сделано с целью подчеркнуть тот факт, что ввод обрабатывается, а не просто копируется на экран.

```
      CLS                          'очистить экран
      X = 1
      Y = 1                        'координаты курсора 10 'нц
      IF Y <> 24 THEN 15           'столбец заполнен
      Y = 1
      X = X + 1                    'переход в новый столбец
15     LOCATE Y, X                 'переставить место вывода
20    A$ = INKEY$                  'осмотреть клавиатуру
      L = LEN(A$)                  'сколько введено байт
      IF L = 0 THEN 20             'ничего не нажато
      A=ASC(A$)                    'код клавиши
      IF A = 0 THEN 30             'двойной код клавиши
      ?A$                          'ее на экран
      Y = Y + 1                    'в следующую строку
      GOTO 10                      'одиночный код
30    A1 = ASC(MID$(A$, 2, 1))     'двойной код
      IF A1 = 75 THEN X = X – 1    'стрелка влево
      IF A1 = 77 THEN X = X + 1    ' вправо
      IF A1 = 80 THEN Y = Y + 1    'вниз
      IF A1 = 72 THEN Y = Y - 1    'вверх
      GOTO 10                      'кц
' Упражнение. Написать арабский редактор текста
```

Конечно, настоящие редакторы не просто копируют ввод на экран, а запоминают его в массивах и файлах.

Г Л А В А 13. РИСОВАНИЕ И ЧЕРЧЕНИЕ ЗА ЭВМ

13.1. Место деятельности в ряду умений человека

Исторически рисование (на стенах пещер) предшествовало письму и счету. В таком же порядке осваивает эти виды деятельности и ребенок. Освоение этих видов деятельности компьютером идет в обратном порядке.

Черты этой деятельности таковы.

1. Массовость, повседневность. Чем меньше возраст ребенка, тем больше преобладает рисование над счетом и письмом.

2. Большая трудоемкость. Выполнение чертежей вручную – недопустимо медленная операция в век ЭВМ.

3. Низкая автоматизация. По-прежнему карандаш и кисти, линейка и ластик остаются основными инструментами.

Компьютер дает здесь возможность, прежде всего, простых исправлений на экране и легкого тиражирования рисунков.

При массированном использовании ЭВМ для создания изображений возможна утрата навыков ручной работы, и в перспективе взрослый, профессионально создающий изображения на экране, сможет лишь с трудом провести прямую линию на бумаге при помощи линейки и, возможно, обведет выступивший за линейку палец так, как он делал это в детстве. Аналогично, рисуя окружность с помощью циркуля, он переставит пару раз иголку циркуля и получит спираль. Но пока это дело будущего.

Компьютер заставляет перестраивать мышление в направлении расшифровки экранных изображений. Так, схема-проект дизайна автомобиля состоит из всех видимых и невидимых линий, все они присутствуют на чертеже на экране. Увидеть в них автомобиль не так то просто.

13.2. Требования к компьютерной графике

1. Наличие графического экрана. Если в школе есть лишь ДВК-2 без графики, придется работать с псевдографикой. Единственный, но немаловажный плюс здесь – возможность рассмотреть экран как состоящий из отдельных пикселов (в данном случае знакомест), увидеть дискретность компьютерной графики.

2. Наличие графопостроителя в школе желательно потому, что это, кроме всего, вещественный исполнитель типа «Чертежник».

3. Высокая скорость выполнения рисунков необходима, если не ставится цель проследить процесс создания чертежа или рисунка, для поддержки внимания и интереса обучаемых.

4. Возможность включения в рисунок текста нужна хотя бы при подготовке раздаточных материалов, изучении геометрии.

Сразу же обратим внимание обучаемых на дискретность компьютерной графики, наличие на экране пикселов – окрашенных точек.

13.3. Способы хранения графической информации

Известны два основных способа хранения графики – в виде битовой карты, т.е. «живьем» и в форме алгоритма.

При побитном хранении состояние каждого пиксела экрана запоминается в файле. В простейшем случае пикселу соответствует один бит. Состояние ноль соответствует черному цвету, 8 – белому. При раскрашивании пиксела в цвета объем памяти увеличивается, но медленно, так как каждому цвету соответствует комбинация битов. Оценим объем памяти для черно-белого рисунка. Типичные данные таковы: 600 точек по горизонтали на 400 по вертикали. Это потребует 240 Кбит или 30 Кбайт на одну картинку. Если графика динамична, в секунду надо сменять 20 кадров, итого 600 кбайт в секунду. Ясно, что памяти так «не напасешься»: 1000 секунд на 1 CDROM-диск, причем без оттенков!

Графика реальных изображений расположена, образно говоря, в пространстве и времени. Ее можно сжать.

Если картинка содержит большие одинаково закрашенные области, можно запоминать цвет точек и количество одинаковых идущих подряд по горизонтали точек.

Если некоторая часть рисунка сохраняется на разных кадрах, ее можно хранить один раз и помнить, на сколько кадров она распространяется.

Рассмотренный способ хранения графики побитно прост и быстр, но расточителен. Другой крайний способ – хранение алгоритма, создающего рисунок. Так, вместо битов закрашенного прямоугольника можно хранить несколько байт команды вида:

```
LINE (A, B)-(C, D), E
```

Этот способ весьма экономен, если рисунок прост и состоит из однородных по цвету частей или линий.

На практике оба метода комбинируются: например, строится неподвижный фон из битовой карты, а на нем движутся вычисляемые небольшие изображения. Можно перемещать и небольшой блок графической информации – картинку, называемую спрайтом.

13.4. Простейший графический редактор

Рассмотрим открытую программу – модель графического редактора. Этот мини-редактор всего лишь оставляет на экране след от движущейся точки. Его возможности умышленно не развертываются – это дело обучаемого.

```
SCREEN 9                    'графический малый редактор
SL = 75
SP = 77
SV = 72
```

```
      SN = 80                      'вторые коды стрелок на IBM
10 A$ = INKEY$                  'опрос клавиатуры
      L = LEN(A$)                  'сколько байт пришло
      IF L = 0 THEN 10             'ничего не нажато
      A = ASC(A$)                  'первый код
      IF A <> 0 THEN 10            'пропустить одиночный код
      A=ASC(MID$(A$, 2, 1))        'второй код клавиши (стрелки)
      IF A = SL THEN X = X - 1     'движение точки рисования
      IF A = SP THEN X = X + 1     'нажата стрелка вправо
      IF A = SN THEN Y = Y + 1     ' вниз
      IF A = SV THEN Y = Y - 1     ' вверх
      PSET (X, Y), 15              'отобразить точку
      GOTO 10                      'бесконечный цикл диалога
```

В связи с освоением этой программы можно сформулировать следующие задания на ее модификацию:

1. Ввести ограничения на диапазон значений координат.
2. Ввести режимы «поднять/опустить перо» для холостого хода.
3. Ввести команду запоминания координат текущей точки для рисования отрезка и окружности.
4. Вести команду мигания курсором для визуализации его положения.
5. Вывести на экран справку о текущем состоянии – режим, координаты курсора.

Перечень заданий легко продолжить.

13.5. Алгоритмы рисования линий на экране

Простой и дидактически полезный метод рисования линий – моделирование механического движения точки по экрану: координаты точки изменяются как функции времени, через шаг dt точка выводится. Рисуем окружность:

```
нц для t от 0 до 2 шаг dt
      точка(x0 + r * cos(t), y0 + r * sin(t)), цвет
кц
```

Этот алгоритм имеет два существенных недостатка:

1) если шаг dt велик, линия получается «дырявой», если мал – многие точки выводятся повторно, ибо их вещественные координаты округляются с точностью до размеров пиксела.

2) на старых школьных ЭВМ и тем более в старом Бейсике многократное, массовое вычисление тригонометрических функций резко замедляет процесс рисования.

Итак, алгоритм рисования должен быть быстрым. Рассмотрим популярный в компьютерной графике алгоритм, основанный на методе минимального отклонения точки. Идея метода в следующем.

Пусть кривая описана в неявном виде уравнением f(x, y) = 0. Тогда по одну сторону от кривой лежат точки такие, что f(x, y)>0, а по другую – f(x, y)<0. Сама кривая может не проходить ни через одну из экранных точек с целочисленными координатами. Значит, ее следует изобразить близкими к кривой пикселами экрана. Начав с некоторой точки, следует перемещаться из нее в ту из соседних, для которой отклонение модуля функции формы от 0 будет наименьшим. Найдем приращения функции формы для случая окружности:

f(x, y) = x**2 + y**2 - r**2 = 0

при смещении на dx = -1 имеет место:

dfx = (x - 1)**2 + y**2 - (x**2 + y**2) = -2 * x + 1

Аналогично при смещении на dy = 1 имеем:

dfy = x**2 + (y + 1)**2 -(x**2 + y**2) = 2 * y +1

Начнем строить окружность с крайней правой точки с координатами (r, 0):

```
x := r; y := 0; f := 0
нц пока x>y
        fx := f - 2 * x + 1;
        dfy := f + 2 * y + 1;
        если abs(fx) > abs(fy)
                то      f := fx; x := x - 1;
                иначе f := fy; y := y + 1;
        все
        точка (x, y), цвет
кц
```

Осталось добавить к координатам x0, y0 – положение центра окружности. Составленный алгоритм рисует 1/8 окружности. Остальные дуги получаются семью зеркальными отражениями координат, например:

точка(x0 - y, y0 + x), цвет.

На медленно работающих компьютерах можно заметить, что окружность возникает сразу их четырех точек, в виде 4 удлиняющихся дуг, которые затем смыкаются.

Полученная выше окружность будет толстоватой и угловатой потому, что смещения выполняются только под прямым углом: только по x или по y. Но можно ввести и смещение сразу и по x и y:

dfxy := dfx + dfy;

и рассмотреть его в качестве третьего альтернативного варианта смещения.

Отрезок прямой строится аналогично, но алгоритм получается несколько более громоздким из-за учета направления движения по x и y, знаков приращений dx и dy.

13.6. Реалистические изображения на экране

Рисование командами типа LINE и CIRCLE – это стиль детского рисунка: небо – синее, солнце – красное, освещенность задается однородной с помощью команды типа PAINT. В то же время реалистичность изображений возникает из-за того, что линия появляется на рисунке не сама по себе, а только как граница света и тени (или разных цветов). В пределах одного изображения имеют место оттенки от черного до белого. Итак, нужно просмотреть все точки экрана и покрасить их в промежуточные серые цвета.

Для определения освещенности точки привлекаются методы оптики. Если на точку поверхности падает пучок параллельного света, то ее освещенность равна cmax * cos(n, s), где cmax – максимальная освещенность, n – вектор нормали к поверхности, s – единичный вектор направления на источник света. При угле между векторами, большем 90 градусов точки окрашиваются в черный цвет. Вектор нормали к поверхности получается нормировкой вектора – градиента функции f(x, y, z), задающей поверхность:

f(x, y, z) = 0

Рассмотрим алгоритм рисования Луны. Будем считать, что образ Луны – шар радиуса r0. Свет падает перпендикулярно на экран с изображением, Так же ориентирована ось z.

Тогда вектор нормали равен ((x-x0)/r0, (y-y0)/r0, z/r0)), скалярное произведение его на вектор света равно z/r0.

```
нц для y от 1 до maxy                ! перебор точек экрана
    нц для x от 1 до maxx            ! по x и y
        r := sqrt(x - x0)**2 + (y - y0)**2! расстояние до оси z
        если r<r0                        ! поверхность Луны
        то c := cmax * z / r0        ! освещенность
        иначе c := 0                 ! черное небо
```

```
            все !
        точка (x, y, ), c                    ! вывести пиксел
        кц !
кц !
```

Итак, наиболее освещена точка, лежащая на оси z, проходящей через центр экрана и перпендикулярно ему.

13.7. Имитация оттенков при помощи случайных чисел

Проблема состоит в том, что вычисленная освещенность – это число вещественное, а оттенков в худшем случае всего 2: черное и белое. На помощь приходят случайные числа. Будем окрашивать точку в черный или белый цвет, но цвет выбирать случайно, в зависимости от освещенности. Если она близка к максимальной, условно принятой за 1, будем чаще выдавать белый цвет, иначе – черный. Алгоритм таков:

```
w := rnd(1)              ! возьмем случайное число
если w<c                 ! меньше освещенности
        то color := 1    ! белый цвет
        иначе color := 0 ! черный цвет
все                      !
```

Такой прием позволяет строить изображение с зернистой структурой поверхности, что весьма похоже на естественную поверхность Луны.

Метод оптического моделирования освещенности тел приводит к весьма реалистичным изображениям, особенно если еще учесть отбрасываемые тени. Недостаток метода – большое время счета. Ведь для каждой точки поверхности вычисляется градиент от сложной функции, нормаль и косинус, что особенно при отсутствии математического сопроцессора ведет к большим затратам времени. Тем не менее метод широко применяется в компьютерной мультипликации (фильмы типа «Звездные войны»). Рисование одного кадра фильма на суперкомпьютере CRAY занимает порядка минуты.

Г Л А В А 14. РАБОТА С ЦВЕТАМИ НА ЭКРАНЕ

14.1. Роль цвета в обучении

Работа с цветом заслуживает особого внимания ввиду больших дидактических возможностей цветового кодирования. Компьютер здесь дает следующие возможности:

1) миллионы цветов и сотни оттенков на профессионально ориентированных ЭВМ;

2) прямое управление цветом на экране вместо кустарного смешивания красок на доске с дыркой, именуемой палитрой.

3) удобство редактирования и хранения изображений.

Вообще компьютер, как и цветной телевизор, поддерживает самообман человека: ведь восприятие цвета человеком ограничено красным, синим, зеленым цветами и их комбинациями. Колбочки на дне глазного яблока ничего другого и не воспринимают. Но это облегчает техническую имитацию оттенков на базе все тех же трех цветов.

Основная возможность для педагогики – подкрепление цветом текстовой или графической информации. Известно, что усвоение материала происходит тем лучше, чем больше органов чувств задействовано при восприятии.

Простейшее применение цвета – высвечивание красным цветом места ошибки. Это было выполнено еще на компьютерах «Агат» в системе программирования «Рапира». При отсутствии цветов их роль выполняло выделение яркостью (СКП ФОКУС для ЕС ЭВМ). Типичное же цветовое кодирование таково: красный цвет – ошибка, желтый – внимание, зеленый – все нормально. Цветовое кодирование особенно важно для тех, кто работает преимущественно с графикой, т.е. для малышей.

В наше время цветовое кодирование широко применяется и в работе с текстом. Так, редактор среды Турбо Паскаль по-разному раскрашивает авторский текст программы, ключевые слова и комментарии. Мощно представлено цветовое кодирование в оболочках FAR и DOS NAVIGATOR. Текстовый процессор WORD подчеркивает незнакомые слова красной линией.

При моделировании, решении задачи о нагреве пластины [38] можно раскрашивать клетки в зависимости от их температуры от темно-красного до белого, показывать процесс нагрева.

14.2. Профессиональные и учебные требования к цвету

Для профессионального или будущего художника необходимое условие – возможность так называемой растяжки цвета – получение плавного перехода от одного цвета к другому. Для учебных целей общеобразовательной школы столь тонкие оттенки не обязательны, но три базовых цвета – красный, синий и зеленый необходимы.

Профессиональному художнику или дизайнеру необходимы плавные переходы между оттенками. Для этого Windows-приложения предлагают сегодня по 1 байту на каждый из базовых цветов: три байта на пиксель. Итого 256 уровней интенсивности каждого базового цвета и 2^24 цветов-

смесей. Конкретный цвет получается как функция RGB(R, G, B) от интенсивностей красного, синего и зеленого цветов.

Экзотической остается возможность получения твердой цветной копии рисунка. Соответствующие принтеры стоят дорого, а графопостроитель не смешивает цвета и ориентирован лишь на линейчатые, в детском стиле рисунки.

При чтении с экрана текста весьма важна правильная раскраска фона и букв. Так, наличие более чем 4 цветов утомляет зрение. Гармоническими парами цветов являются желтый (буквы) и черный фон, зеленый (буквы) и черный фон. В среде WINDOWS при стандартной настройке количество цветов ограничено: основная пара – черные буквы на белом (сером фоне). Белый фон напоминает бумагу, но связан с более сильным излучением и выгоранием монитора. Утомляет зрение голубой цвет.

14.3. Генерация узоров и орнаментов на ЭВМ

Орнамент с точки зрения математики – некоторое изображение, обладающее симметрией, т.е. инвариантное по отношению к переносу, вращению и т.д. Повторяющаяся часть рисунка называется «раппорт». В принципе любой орнамент можно нарисовать при помощи графического редактора, работающего с цветами. Но именно повторяемость рисунка и позволяет эффективно применить ЭВМ. Более интересно не только редактировать, а генерировать орнаменты. Для этого достаточно рассмотреть цвет как некоторую функцию экранных координат. Тогда периодичность этой функции или ее четность/нечетность и породят узор. Остается преобразовать произвольную функцию в номер цвета. Это можно сделать так: c:=abs(f(x, y)) mod mcol. Здесь mcol – максимальное число цветов. Итак, снова поточечное сканирование экрана:

```
нц для y от 1 до maxx
 нц для x от 1 до maxy
        c:=abs(f(x, y) mod mcol
 точка (x, y), c
 кц
кц
```

Рассмотрим важные частные случаи.

1) f = 0. Весь экран закрашивается в нулевой – черный цвет.

2) f = x. Номера цветов возрастают по горизонтали, после наибольшего снова идет нулевой. Экран раскрашивается в вертикальную полоску. Функция f = y порождает горизонтальные полосы.

3) f = x + y порождает диагональные полосы.

4) f = abs(x) порождает симметрию относительно вертикальной оси.

5) f = abs(x) + abs(y) порождает ромбы.

6) f = 0.5 * x растягивает полосы до двойной ширины.

7) При f = 0.1 * x**2 + 0.2 y**2 генерируются эллипсы.

Для перехода к узорам, имеющим прямую технологическую ценность для вышивки, плетения, вязания т.д., можно рассмотреть не графические, а псевдографические, знакоместные узоры. Тогда знакоместо на экране может соответствовать некоторому узлу или кресту вышивки или точке плетения ткани (переплетение «утка с основой»). Такой узор строится на порядок быстрее чисто графического орнамента. Учитывая целочисленность координат, можно ограничиться целочисленными функциями, которые получаются при применении к целочисленным координатам и константам логических операций AND, OR, XOR, NOT, DIV, MOD, например:

f = (a * x + b * y + c) and (y * y + x * x) mod d

Здесь a, b, c, d – целочисленные константы, x, y – координаты знакомест на экране.

Характерно, что практически любой набор операций и констант в выражении все равно порождает некоторый узор.

Для изучения математики этот подход дает обогащение понятия функции, таких свойств, как четность, нечетность, периодичность.

Для изучения информатики важно, что метод следует идеологии построения изображений сканированием экрана – наиболее современному и мощному методу создания графических образов на экране ЭВМ.

Для дисциплин «обслуживающий труд» и «художественное конструирование» открывается поле для практически неограниченного доступа к разнообразным схемам технологических узоров. Знать программирование, естественно, не нужно, достаточно варьировать параметры в числовых формулах – выражениях для функции.

Ниже приводится простейший вариант генератора узоров.

```
    SCREEN 9                    'генератор узоров.
    FOR X=0 TO 600              'нц
    FOR Y=0 TO 400              'сканирование экрана
    A=.02                       'параметр узора. Менять!
    X1=X-64                     'переход в центр картинки
    Y1=Y-48                     'по обеим координатам
    F=A*(Y1 * Y1 -X1 * X1)      'функция-генератор: любая
    F=ABS(F) mod 7              'цвет всегда > 0
    PSET(X, Y), F               'окрасить пиксел
    NEXT Y, X                   'кц
70 GOTO 70                      'подождать прерывания
```

Г Л А В А 15. КОМПЬЮТЕР И ЗВУК

15.1. Звуковые возможности компьютера

Восприятие и создание звуков – деятельность еще более древняя, чем рисование, и доступна и животным. Для человека характерна речь как сложнейшая деятельность по созданию и восприятию звуков. Компьютер «учится» этому в наши дни в последнюю очередь после, например, счета. Генерирование звуков путем синтеза речи настолько же сложно, насколько просто сегодня записать и воспроизвести речь. То же относится и к созданию музыки.

Компьютер предоставляет весьма интересную возможность коррекции высоты звука при освоении ребенком пения. По мнению автора, практически не существует людей с полным отсутствием музыкального слуха, если есть слух обычный. Просто в тот момент, когда ребенок начинает петь, нужно обратить его внимание на связь своего звука с эталоном и показать, что нужно сделать – петь «тоньше» или «толще». Компьютер, оснащенный микрофоном, может, как идеальный репетитор, терпеливо выдавать в графическом и даже цветовом коде степень рассогласования звука издаваемого и эталонного и, конечно, знак этого рассогласования. Чтобы петь правильно, любому ребенку достаточно связать звук, который он слышит и звук, который он издает. Возможно, что при оснащении обычных детских садов необходимой техникой и ПС прекратится то мяуканье, которое сейчас называется детским пением и так умиляет родителей, в свое время научившихся «петь» точно так же.

Компьютер также дает удобство хранения и редактирования мелодий. Специфической операцией является транспонирование: параллельный перенос мелодии на несколько интервалов выше или ниже. Это чисто формальное преобразование, аналогом которому является переход в тексте от строчных букв к прописным.

15.2. Три способа применения ЭВМ

Можно использовать ЭВМ в качестве музыкального инструмента, связав клавиши с конкретными звуками. Можно использовать ЭВМ как проигрыватель после того, как мелодия записана. Наконец, можно генерировать на ЭВМ новые мелодии автоматически. Кстати, в [17] описан эксперимент распознавания машинных мелодий и мелодий, созданных человеком. Эксперты-музыканты дружно заявили, что смогут точно распознать человеческую музыку. Так же дружно эксперты назвали машинную музыку – человеческой и наоборот.

15.3. Основные сведения о нотах и звуках

Звук вообще есть колебание плотности воздуха. Чистый звук – это синусоидальное колебание:

Z(t) = A * sin (w * t), 0 <= t <= T0.

Он характеризуется тремя параметрами: амплитудой (громкостью) A, частотой (высотой) w и длительностью T0. Реальный звук есть смесь звуков разных частот и амплитуд.

Чистый звук в музыке описывается названием ноты: ля (A), си (B), до (C), ре (D), ми (E) фа(F) соль (G). Но нот всего семь, а звуков, с учетом вещественности типа частоты, бесконечно много. Как быть?

Во-первых, в музыкальном звукоряде рассматриваются не все звуки, а только те звуки, частоты которых отличаются в заданное количество раз. Так, минимальное относительное расстояние между частотами соседних звуков связано с числом 12. Точнее, частоты музыкального звукоряда задаются формулой

w[i] = w[0] * 2 ** (i / 12)

Здесь i – номер звука, w[0] – базовая частота настройки инструмента. Расстояние (интервал) между соседними звуками называется «полутон». Это минимальный музыкальный интервал. В терминах языка Паскаль частота вычисляется так:

w[i] := round(440 * exp(ln(2) * i / 12))

Легко видеть, что через 12 шагов частота повышается вдвое. Соответствующий музыкальный интервал (соотношение частот пары звуков) называется октавой. Звуки, отличающиеся на 12 шагов (в два раза по частоте), обозначаются одним и тем же словом, например, «ре». Интервал октава – самый чистый. Гости, мужчины и женщины, за столом могут дружно петь в октаву и не замечать этого – настолько похожи звуки, отличающиеся ровно в два раза по частоте. Можно сделать такой полезный опыт: пережать гитарную струну ровно посередине. Частота колебаний удвоится, звук станет на октаву выше.

Другим важным интервалом является квинта (7 минимальных шагов звукоряда), отношение частот составляет 2 ** (7 / 12) = 1.498 и равно примерно 3/2. На слух относительная разница в 0.002 едва ли заметна даже профессиональному музыканту. Читателя не должно беспокоить рассогласование между названием интервала («квинта» соответствует латинскому названию числа «5») и числом шагов в нем по частоте (7). Просто в музыке

принята удивительно архаичная, запутанная и при этом устоявшая в веках система именования звуков на основе названий только белых клавиш фортепиано (ля, си, ...) и не менее архаичная запись нот на 5 параллельных линиях, которых почти всегда не хватает. Ее неудобство сравнимо разве что со свойствами десятичной системы счисления. (Приятным исключением является клавиатура баяна – по три звука в ряд). Итак, звуков в октаве 12, а названий для нот только 7 (плюс знаки повышения/понижения ноты на полтона.) Мы будем пользоваться преимущественно номерами звуков, а не нотами.

Если на чистый звук накладывается колебания с кратными частотами, возникает окраска звука – обертоны. Чистый синусоидальный звук напоминает вой, он неприятен. Именно обертоны придают звуку окраску, характерную для различных инструментов и голоса. Человеческая речь – сложнейшая система звуков, в значительной части шумов. Шум (белый) есть смесь звуков со случайными амплитудами и частотами. Наиболее сложно устроены звуки, промежуточные между белым шумом и чистым синусоидальным колебанием.

15.4. Простые применения звука. Команда BEEP

Команда Бейсика BEEP выдает короткий чистый звук одной частоты и амплитуды. На Паскале ее аналогом является подпрограмма:

```
procedure beep;
begin
      sound(400);  {включить звукогенератор на частоте 400 гц}
      delay(200);  {задержка программы на 0.2 сек }
      nosound      {выключить генератор звука }
 end;
```

У команды BEEP возможна масса применений. Только инерция мышления программистов этому мешает. Приведем примеры.

1. Озвучивание нажатия «не той клавиши»:

```
10 A$ = INKEY$                  'осмотреть клавиатуру
IF LEN(A$) = 0 THEN 10          'ничего не нажато
BEEP                            'подтвердить нажатие
```

2. Привлечение внимания к окончанию работы программы:

```
BEEP
END
```

3. Указать на ошибку при вводе числа:

```
10 ?»Значение сопротивления»
INPUT R                  'ввод
IF R >= 0 THEN 70        'все правильно
BEEP                     'была ошибка
GOTO 10                  'вернуться на ввод
70'...                    'считаем дальше
```

4. Озвучить факт захождения программы в команду IF:

```
IF PI = 0 THEN BEEP 'Если Бейсик не знает PI, то гудок
```

5. Озвучить итерацию цикла:

```
10 FOR I = 1 TO 0 'часть Бейсиков делают FOR хотя бы 1 раз
20 BEEP           'проверим данную версию на корректность
30 NEXT
```

6. Сопровождение звуком события на экране

```
IF H <= 0 THEN BEEP 'высота полета стала меньше нуля. Удар!
```

15.5. Требования к звуку на ПЭВМ

Перечислим требования к возможностям звука на компьютере.

1. Многоголосие совершенно необходимо профессионалу и желательно для учебных целей.

2. Управление длительностью звука необходимо профессионалу и желательно для учебного процесса.

3. Управление тембром звука – профессиональное требование.

4. Управление звуком в стиле игры на инструменте (баян) может повысить интерес учащихся. Клавиатура ЭВМ вполне подходит для такого «баяна».

5. Запоминание мелодии в файл позволит профессионалу создавать базу мелодий, а обучаемому – эффективно поработать более одного урока.

6. Команда BEEP незаменима для учебных целей.

7. Возможность сочетания звука с цветом обогащает ассоциативное мышление обучаемого.

8. Прямое управление звуком через звукогенератор подобно программированию на ассемблере: возможностей много, но до них не так просто добраться. Все же имитация, например, взрыва весьма интересна для обучаемых.

15.6. Команда PLAY

Во всех версиях Бейсика есть команда PLAY. Среди ее возможностей выделим простейшие.

Команда PLAY «L16» устанавливает скорость игры – 16 нот в единицу времени (можно «L1», «L2», «L4» и т.д.).

Команда PlAY «N23» выдает ноту номер 23 (можно играть ноты с номерами от 1 до 84).

Команда PLAY «B» сыграет ноту Си (см. выше обозначения латинскими буквами).

Пример:

```
10 I = 78              'чтобы выдать ноту номер I
20 I$ = STR$(I)    'превратим I в строку
30 PLAY «N»+I$  'сыграть ноту номер I
```

Монохроматической гаммой называется последовательность всех звуков звукоряда. Чтобы ее выдать, надо перебрать все звуки:

```
PLAY «L16»        'установить длительность нот в 1/16
FOR I = 1 TO 84   'монохроматическая гамма – все звуки
I$ = STR$(I)        'превратим I в строку
PLAY «N» + I$     'сыграть ноту номер I N
NEXT I
END
```

15.7. Озвучивание клавиш. ПЭВМ как баян

Простейший способ озвучить клавишу – сыграть ноту с номером, равным ее коду. Но так как кодов клавиш – 256, а нот 84, нужно при помощи вычисления остатка по модулю привести код клавиши в диапазон номеров нот:

```
10 A$ = INKEY$                  'осмотреть клавиатуру
IF LEN(A$) = 0 THEN 10      'ничего не нажато
NK = ASC(MID$(A$, 1, 1))    'первый код клавиши
NN = NK MOD 84+1              'преобразование в номер ноты
NN$ = STR$(NN)                'преобразование в строку для PlAY
PLAY «N» + NN$                'сыграть ноту
GOTO 10                           'ждать следующей
```

Расположение буквенных клавиш на клавиатуре ПЭВМ полностью соответствует баяну – три ряда клавиш под углом в 60 градусов. Значит, достаточно связать с каждой клавишей звук нужной высоты:

```
10 DIM A(256)                    'номера нот для кодов клавиш
   PLAY «L16»                    'длительность нот – 1/16
   FOR I = 1 TO 24               'минибаян – 24 клавиши
   READ S$                       'читать название клавиши
   S = ASC(S$)                   'ее код
   A(S) = I                      'ее ноту запомнить
   NEXT I
   DATA Q, A, W, Z, S, E, X, D, R, C, F, T, V, G, Y, B, H, U, N, J, I, M, K,
O 20 K$ = INKEY$                 'нажата ли клавиша
   IF LEN(K$) = 0 THEN 20        'не нажата
   PK$ = MID$(K$, 1, 1)          'первый код клавиши
   NN = A(ASC(PK$))              'номер ноты
   NN = NN + 20                  'повысить для приятности звучания
   NN$ = STR$(NN)                'строка для команды PLAY
   PLAY «N» + NN$                'сыграть звук
   GOTO 20                       'идти за следующей нотой
```

15.8. Принципы генерации мелодий на ЭВМ

Татарский математик и композитор Р.Х. Зарипов [17] сформулировал ряд интересных идей для генерации мелодий на ЭВМ. Суть их – в варьировании второстепенных признаков при сохранении ядра мелодии. Так, вольно или невольно, поступают иногда и профессиональные композиторы. Достаточно сравнить песню «Молодежная» И.О. Дунаевского и старинную казачью песню «По Дону гуляет». Даже не будучи музыкантом и не зная нотной грамоты, из графической записи мелодий видно, что записана одна и та же мелодия, только у Дунаевского каждая нота удвоена. Если и это является творчеством, то компьютеру такое творчество вполне под силу.

Уточним приемы варьирования мелодии:

1) транспонирование части мелодии – перенос вверх или вниз;

2) замена звука гармонически допустимым – типично джазовый прием при импровизации;

3) изменение размера: вальс превращается в марш или наоборот;

4) ввод случайных изменений и проверка их на гармоническую допустимость.

Последний прием роднит компьютерное музыкальное творчество с генераторами задач и орнаментов. Там так же случайно выбирались параметры и проверялись на допустимость.

Из указанных приемов видно, что возможна постановка и обратной задачи – выявление при помощи ЭВМ сходства различных мелодий, особенно у разных авторов.

15.9. Простейший генератор мелодии

Рассмотрим простую программу – генератор мелодий. На компьютере Ямаха можно выдавать три звука одновременно. Это позволяет сгенерировать трехголосое «пение»:

```
 5                                  'музыкальный генератор
 10 FOR T = 1 TO 8                  'цикл по тактам
20 FOR N = 1 TO 7                        'цикл по нотам
30 Z1 = (T AND N) MOD 7                  'генерация первой мелодии
35 Z1$ = CHR$(Z1 + ASC(«A»))             'название ноты для команды PLAY
40 Z2 = (T * N) MOD 7                    'нота второй мелодии
45 Z2$ = CHR$(Z2 + ASC(«A»))             '
50 Z3 = (T HOR N) MOD 7 :                '
55 Z3$ = CHR$(Z3 + ASC(«A»))             ' третьей
60 PLAY Z1$, Z2$, Z3$                    'Выдать три голоса одновременно
70 NEXT
```

Основные идеи программы таковы. Звуки выбираются из базовых нот в тональности «ля минор». Это уменьшает число негармоничных сочетаний звуков. Генерация мелодии выполняется с помощью трех, в общем, случайно составленных функций. Длительности нот фиксированы. В конце такта пауза для ритмического разбиения мелодии.

Итак, компьютер порождает ряд новых возможностей для музыкального творчества, с которыми полезно ознакомиться как студенту, так и школьнику.

15.10. Звук и цвет, способы связи

Чистый звук и цвет роднит то, что оба являются колебаниями синусоидального типа. Поэтому связать звук и цвет формально достаточно легко: нужно как-то выразить частоту одного через частоту другого или, что проще, формально связать номер цвета с номером звука:

```
c := z mod 15
```

Здесь c – номер цвета в диапазоне от 0 до 15, z – номер звука. Некоторые композиторы (Скрябин) считали, что цвет присущ нотам более сильно, например, нота ре – желтого цвета, ми – синего и т.д. Но если учитывать относительность точки отсчета частот, эти «окраски» можно сопоставить только звукам, взятым в отношении к базовому уровню частот, который, вообще говоря, произволен: звук, который является нотой «ре» при одной настройке инструмента, будет звуком «ми» при другой, но близкой по частоте базовой ноты настройке. Поэтому на сегодня предлагаемое

формальное связывание номеров звуков и цветов – неплохое упражнение для развития ассоциативного мышления, установления межпредметной связи с физикой. В целом эту тему полезно рассматривать с обучаемыми именно в контексте физики, точнее акустики.

Г Л А В А 16. УПРАВЛЕНИЕ ИСПОЛНИТЕЛЯМИ

16.1. Два типа управления

Управление исполнителем – весьма древний и в чем-то первичный вид деятельности. Ведь и животное выступает как субъект управления по отношению к объекту – собственным ногам. Акты управления совершаются и человеком, непрерывно и чаще всего незаметно. Так, человек не осознает деталей движения руки при написании букв слова, например, во время урока. Управлению такого типа тысячи лет. Его главный признак состоит в том, что субъект управления погружен в процесс непосредственного управления, управляет сам.

Компьютер породил новый тип управления. Теперь от субъекта требуется воздействовать не прямо на субъект, а на управляющий компьютер, и не действием, а словом – передать компьютеру управляющий алгоритм. Научить такому стилю управления и мышления – одна из главных целей курса школьной информатики. Такому стилю управления всего около сорока лет.

Объект управления будем называть исполнителем.

Итак, если в случае непосредственного управления результатом деятельности человека с помощью исполнителя является, например, вспаханное поле, то при компьютеризованном управлении результатом человека является текст алгоритма, позволяющего вспахать множество полей.

Исполнителя, следуя учебнику [25], можно представить в виде устройства с кнопками: одна кнопка – одна команда.

Совокупность кнопок – команд образует систему команд исполнителя (СКИ). Исполнитель «умеет» выполнять только те команды, которые входят в его СКИ. Поэтому исполнитель не понимает слов «пока» «если» и т.д. Это слова предназначены для управляющей ЭВМ. У исполнителя они вызывают либо отказ, либо просто пропускаются.

Разделение труда между ЭВМ и исполнителем таково. ЭВМ обрабатывает текст алгоритма. Те команды, которые ей непонятны, она передает исполнителю.

Рассмотрим требования к учебным исполнителям.

1. Простота управления для начинающего изучать алгоритмизацию.

2. Наличие двух режимов работы: непосредственное управление и управление через программу.

3. Желательна вещественность, материальность самого исполнителя. В этом отношении хорош, например, микрокалькулятор.

4. Наглядность результатов работы, их вещественность.

5. Безопасность для людей и аппаратуры.

Не стоит использовать для изучения управления лифт, хотя дети и любят ездить «на 1235-ый этаж».

16.2. Основной алгоритм работы исполнителя

Исполнитель в общем работает так. Читается очередная команда. Если она входит в СКИ, исполнитель ее выполняет и ждет ввода следующей:

```
Алг Исп
нц
получить команду к
выбор
при к=к1: серия1
при к=к2: серия 2
...
все
кц
кон
```

Здесь выполняется бесконечный цикл. Очевидно, что этот алгоритм совпадает с общим алгоритмом работы диалоговых программ. Поэтому любая диалоговая программа является одновременно и исполнителем.

Для программного управления исполнителем необходимо на его вход подать поток команд, например, из файла:

```
нц пока не конец файла
        прочитать команду из файла
        передать исполнителю
кц
```

16.3. Редакторы как исполнители. Макрокоманды

Очевидно, что редакторы – частный случай исполнителей. Есть и клавиши-кнопки, например, со стрелками. Но в случае редакторов, кроме ввода команд с клавиатуры или из файла, возможен другой, более удобный стиль программного управления – при помощи макрокоманд.

Макрокоманда – это последовательность команд, точнее, кодов клавиш-команд, выполняющаяся как единое целое. Она может запускаться одной-двумя («аккорд») клавишами.

Типичная последовательность создания макрокоманды такова:

включить запись;

выполнить шаги макрокоманды в непосредственном режиме;

закончить запись;

отредактировать при необходимости макро как текст алгоритма.

16.4. Графопостроитель как исполнитель

Графопостроитель является вещественным аналогом исполнителя Чертежник из учебника [25]. Более того, в его СКИ входят буквально все команды Чертежника. Дополнительная команда – «взять карандаш номер N», где N равно, например, 1, 2, 3 или 4. Устройство для рисования – головка с захватом для карандаша, скользящая по вертикальной рейке. Рейка перемещается параллельно верхнему краю листа бумаги, горизонтально.

Графопостроитель допускает оба способа управления – при помощи кнопок, перемещающих рейку и скользящую по ней головку с пишущим элементом, и при помощи последовательности команд, вводимых из файла.

Любопытно и поучительно ведет себя графопостроитель, если перед командой «взять карандаш» из нужного команде гнезда «украсть» карандаш. Алгоритм его действий следующий.

```
алг Взять карандаш( новый_номер )
нач
        если номер карандаша в головке = новый_номер
        то выход
все
запомнить положение головки
подвести головку к гнезду для старого карандаша
вставить карандаш на старое место
подвести головку к месту для нового карандаша
нц пока карандаш не взят
        взять карандаш из держателя в головку
кц
вернуть головку на место
кон
```

Графопостроитель будет пытаться взять карандаш, исполняя бесконечно последний цикл.

Важное отличие графопостроителя от экранных исполнителей – непрерывность рисуемых линий. Это точнее соответствует смыслу команд Чертежнику типа «сместиться в точку» или «сместиться на вектор».

16.5. Принтер как исполнитель. ESC – команды

Обучение будет более успешным, если исполнитель, которым управляет ЭВМ, будет вещественным, а не экранным, модельным, и результаты работы будут тоже вещественными. Поэтому микрокалькулятор, будучи и вещественным, и доступным, и безопасным исполнителем, все же не совсем годится на эту роль – нет вещественных результатов работы. Далее, не каждая школа может приобрести, например, графопостроитель – отличный аналог исполнителям типа «Чертежник». Его соединение с ЭВМ представляет определенные технические трудности; нужен также специальный драйвер для этого нестандартного внешнего устройства.

Решение, тем не менее, есть и оно почти очевидно. Вместе с компьютером в школе имеется и принтер. А он и есть нужный вещественный, наглядный исполнитель, понимающий специальную группу команд, называемых Esc-командами.

Суть дела в следующем. Если на принтер послать просто символ, например «G», он и будет напечатан, возможно, с задержкой до прихода символа, обозначающего конец файла. Если же послать код 27 клавиши Esc (клавиша «ключ» на ряде отечественных клавиатур), то принтер ничего не напечатает, но будет ждать еще одного, второго кода, иногда и третьего. Символ с этим кодом он тоже не печатает, но понимает уже как управляющий. Этот код и переводит принтер в разные режимы, например, в режим печати жирным шрифтом, двойного прохода, графический режим и т.д.

Очевидное и явно интересное для всех, школьников и студентов упражнение – изготовить хорошо оформленную собственную визитку.

Ввиду низкой наглядности Esc-команд рекомендуется использовать открытую программу-интерпретатор Esc-команд. При этом во входной поток интерпретатора добавляются строки вида:

#ЗВОНОК
#ЖИРНО
#ПОДЧЕРКНУТЬ

и так далее. Символ «#» указывает, что это управляющая строка-команда, и ее не надо печатать. Вместо таких более понятных и наглядных команд интерпретатор подставляет Esc – команды. Приведем текст интерпретатора для принтера Epson Lx-800. Язык Паскаль здесь выбран из-за удобной возможности переадресации ввода из файла.

```
program printer; {интерпретатор команд принтеру }
 var    com: string[2];                          {команда принтеру }
```

```
        st: string;                                  {входная текущая строка }
begin
        while not eof do begin                       {пока не конец файла }
        readln(st);                                  {читать очередную строку}
        if pos('#', st) =1then begin                 {это команда}
        com := copy(st, 2, 2);                  {ее вырезать}
                if com = 'ЗВ' then write(#27, #7);       {звонок }
                if com = 'ЧЕ' then write(#27, #120, #0); {черновик }
                if com = 'ЧИ' then write(#27, #120, #1); {чистовик }
                if COM = 'ПО' then write(#27, #33, #128); {подчерк }
end else writeln(st)                                 {копии строк}
end;
end. {Упражнение. Составьте файл для печати своей визитки }
```

Соответствующий «командный» файл print.txt может выглядеть так:

```
#ЗВОНОК
#ЧИСТОВИК
Петров Петр Петрович
#ПОДЧЕРК эсквайр
#ШИРОКО инженер-программист ПКБ АСУ ...
```

и представляет собой смесь строк текста с управляющими строками с командами.

Вызов интерпретатора для печати на принтере выполняет переадресацию ввода/вывода и выглядит так:

```
print.exe <print.txt >prn
```

Здесь prn – имя устройства (принтер).

Очевидное упражнение – расширить число команд интерпретатора. Впрочем, именно это и делают стандартные драйверы печати. Разработка простого аналога драйверу не может не сопровождаться объективным ростом самооценки и студента, и школьника. Отметим и типичность приведенного выше текста для исполнителя – смесь команд исполнителю с тем, что должно им обрабатываться. Очевидна связь с языком HTML.

.

Г Л А В А 17. ПОИСК ИНФОРМАЦИИ НА ЭВМ

17.1. Роль информационного поиска в ряду умений

Информационный поиск (ИП) – не такой старый вид деятельности, как виды, рассмотренные выше, а поиск за компьютером еще не стал всеобщим достоянием культуры. Тем не менее, все чаще пользователь вынужден прибегать к поиску даже на своем компьютере: информация на диске заведомо есть, но как называется нужный файл – забыто. Далее, одна из тенденций развития мирового сообщества и систем образования – подключение растущего числа пользователей к глобальной сети на основе INTERNET. В любом месте земного шара становится принципиально доступным невиданный ранее объем информации. При этом нередко улучшение программных и технических характеристик средств доступа не решает, а, скорее, обостряет проблему эффективного отбора релевантной (подходящей) информации, показывая неготовность и обучаемых, и учителей к использованию открывающихся возможностей. Возникает парадокс: чем больше информации доступно, тем труднее ее отобрать.

Можно предположить, что не столько графика, сколько ИП станет пожирателем памяти и особенно быстродействия новых поколений ЭВМ. Возможно, решением станет прямой параллельный поиск на мультипроцессорных ЭВМ. Уже сегодня поиск на гигабайтном винчестере файла с указанным фрагментом слова выполняется неприемлемо долго.

Итак, умение успешно выполнять качественный информационный поиск приобретает все большее значение. В то же время в школьных, да и вузовских курсах информатики совершенно недостаточно внимания уделяется обучению информационному поиску. Вне поля зрения остались такие важные понятия классической информатики, как потери и шум, поисковый образ документа (ПОД) и запроса (ПОЗ), полнота и точность поиска и т.д. Чтобы говорить об этом, необязательно иметь выход в сеть. Но действующие программы по информатике обычно ограничиваются элементами поиска при помощи СУБД, что далеко не соответствует поисковым возможностям не только сетевых серверов, но и стандартных средств, имеющихся на каждой ПЭВМ.

Что же дает компьютер для ИП?

Во первых – рост объемов поискового массива до астрономических размеров. Во-вторых, относительное ускорение поиска. Строго говоря, поиск в массиве объемом даже в десяток мегабайт без ПЭВМ практически не возможен. Но компьютер отнимает непосредственность поиска, возможность просмотреть поисковый массив. При терпении и навыке поиск в библиотечном каталоге можно выполнить методом «бульдозера» – просмотром напрямую «подозрительных» ящиков с карточками. (Заодно об-

наружится удивительный беспорядок в размещении карточек по темам). При работе с мегабайтными дисками это невозможно.

Итак, единственным окном в мир доступной информации оказывается поисковый запрос. Умение выражать свою информационную потребность, строить и динамично корректировать запрос становится важнейшим элементом информационной культуры человека. Поэтому ожидаемым следствием распространения мощных ППЭВМ явится развитие таких черт мышления пользователя, как системность и комбинаторность, умение работать с синонимами и омонимами – этими важнейшими спутниками всякого живого языка. Ведь даже на языке программирования одно и то же можно записать массой способов. Что уж говорить о естественных языках.

В настоящее время известно более 110 алгоритмов информационного поиска [59]. Это говорит не только о кажущейся простоте разработки соответствующего программного обеспечения, но и указывает на противоречие между желанием дать пользователю простые средства управления поиском, упростить язык поисковых запросов и в то же время обеспечить высокие характеристики (полноту и точность) поиска. Рискнем утверждать, что именно здесь «без труда не вытащить и рыбку из пруда». Не описав в запросе достаточно точно и полно свою информационную потребность, пользователь не вправе ожидать удовлетворительных результатов поиска.

17.2. Терминология информационного поиска

Сфера информационного поиска имеет развитую терминологию, которая рассредоточена в литературе. Ниже приводятся наиболее важные термины их смысл.

Документ – единица хранения информации. Для СУБД это запись, для текстового поиска это файл. Если файл велик, то под документом можно понимать раздел или параграф.

Поисковый образ документа (ПОД) – набор значений атрибутов и связей между ними, существующие в документе. Следует отличать документ от ПОД. Так, например, в библиотеке документ – это книга, а ПОД – карточка в каталоге.

Поисковый образ запроса (ПОЗ) – набор значений атрибутов и связей между ними в запросе.

Поисковый признак, атрибут – слово или часть слова, используемая при запросе.

Ключ – часть слова, заданная в запросе. Проверяется вхождение ключа в ПОД.

Время доступа – типичное время поиска. Различают поиск до первого подходящего документа и полный поиск всех документов.

Поисковый запрос – некоторое утверждение о свойствах данных. Поиск состоит в отборе документов, для которых это утверждение истинно.

Поиск по логике «И» – поиск документа, в котором есть все атрибуты запроса.

Поиск по логике «ИЛИ» – поиск документа, в котором есть хотя бы один атрибут запроса.

Поиск с «мешочной грамматикой» – такая обработка запроса, при которой игнорируются грамматические связи между атрибутами.

Нормальная дизъюнктивная форма – представление запроса в виде:

(А И В) ИЛИ (С И D)

Нормальная конъюнктивная форма – представление запроса в виде:

(А ИЛИ В) И (С ИЛИ D)

Критерий выдачи – правило отбора подходящих документов.

Толерантность – соответствие документа информационной потребности ищущего.

Полнота поиска – отношение числа подходящих среди найденных документов к числу всех подходящих, в том числе и ненайденных.

Точность поиска – отношение числа подходящий среди найденных документов к числу всех найденных.

Информационный шум – выдача при поиске ненужной информации. Шум ухудшает точность поиска. Вычисляется по формуле: число выданных толерантных документов/число всех выданных.

Информационные потери – невыдача части нужной информации. Потери ухудшают полноту поиска. Вычисляются по формуле: число найденных по запросу толерантных документов/ число толерантных поисковом массиве.

Тезаурус («сокровище») – словарь терминов предметной области с указанием синонимов и связей между терминами. Используется для автоматического пополнения запроса синонимами.

17.3. Требования к средствам учебного поиска

Итак, поиску надо учить. Сформулируем требования к учебному поиску и к поисковому языку.

1. Цель учебного поиска – сам поиск, учиться искать. При профессиональном поиске целью является получаемая информация, при учебном – процесс и логика поиска, итеративное сравнение прогноза и реальных результатов.

2. Малое время отклика. Если при профессиональном поиске, например, ученый-математик или химик может и подождать с полчаса или запустить поисковую программу в фоновом режиме, то учащийся не должен отвлекаться, да и время занятия не так велико. Известно, что время ожидания ответа большее 3-4 секунд снижает интерес к работе.

3. Обозримость поискового массива. Это требование связано с необходимостью объективной апостериорной оценки качества поиска. Результат учебного поиска должен быть в принципе понятен. Поэтому хорошим учебным поисковым массивом может быть введенный в компьютер текст книги, содержание которой в общем известно учащимся.

4. Поисковый массив есть просто текстовый файл. Это наиболее типичная для практики ситуация.

5. Отсутствие действий поисковой системы, выполняемых по умолчанию, без ведома и санкции пользователя. Профессиональные поисковые системы могут «без спросу» модифицировать запрос, например, дополняя ключевые слова вариантами их словоформ [59] из тезауруса. В учебном поиске перечисление словоформ и особенно построение рядов ассоциаций имеет большую самостоятельную познавательную ценность для развития мышления и внешней речи обучаемого. Отметим, что дополнение запроса не только по словоформам, но прежде всего, по рядам синонимов и особенно по цепочкам ассоциаций является само по себе мощным средством уточнения информационной потребности того, кто ищет. В то же время ассоциативное уточнение запроса трудно формализовать и передать ЭВМ, так как при этом важен смысл и контекст запроса.

6. Реализация логических формул над ключами. Общеизвестная плохая масштабируемость поиска при использовании логических операций (резкое сужение при введении операции AND и, наоборот, расширение при OR) не должна вести к отказу от них. Ведь фактически синонимы при запросе все равно связаны через OR, а отдельные ключи – через AND. Приведем пример.

у (ПОПа OR ДЪЯКОНа) AND была (СОБАКа OR ПЕС)

Здесь и далее логические связки и ключи пишутся прописными буквами, а варьируемые части слов, например, окончания пишутся строчными и в запрос не вводятся. Приведенное утверждение записано, кстати, в нормальной конъюнктивной форме. Отметим, что это весьма общая форма запроса: синонимы связываются через AND, группы синонимов через OR.

Итак, учебный поиск должен выполняться быстро, над текстом умеренного объема, содержание которого известно обучаемому, с полностью явно задаваемым запросом, с учетом логики утверждения.

Поисковый образ документа (ПОД), в нашем случае фрагмента текста удобно представить в виде отрезка текста заданной длины, например, в

5 строк. Применяемое нередко в ИПС сечение текста до уровня предложения связано с техническими трудностями (например, инициалы всегда содержат и прописные буквы, и точки). Далее, частая замена ключа местоимением в одном из 2 рядом стоящих предложений ведет к потере информации при поиске. Что касается сечения на абзацы, то фрагмент длиной в абзац у некоторых авторов, например, у И.С. Тургенева, простирается на 1-2 страницы. Это может вести к ложной координации слов запроса и информационному шуму.

17.4. Линейная весовая форма запроса

Собственно запрос удобно представлять не в виде логической формулы со скобками, а при помощи формулы с весами. Если сумма весов слов, встречающихся в ПОД, превышает заданное пороговое значение, поиск считается успешным. Пример:

1 из ЭВМ 1 компьютер 1 калькулятор 1.

Здесь левая единица – пороговое значение, числа после ключей – их веса. Запрос будет успешным, если сумма весов достигнет порога – единицы, т.е. в данном случае, если в ПОД найдется хотя бы одно ключевое слово. Это эквивалентно логической формуле:

ЭВМ OR компьютер OR калькулятор.

Операция «И» также реализуется просто:

3 из ЭЛЕКТРОННая 1 ВЫЧИСЛИТЕЛЬНая 1 МАШИНа 1

Здесь 3 – пороговое значение суммы весов. Запрос будет успешным, если сумма весов достигает 3, т.е. в отрезке текста длиной, скажем, в 5 строк встретятся все три ключа. Эта весовая формула эквивалентна следующей логической:

ЭЛЕКТРОННая AND ВЫЧИСЛИТЕЛЬНая AND МАШИНа

Легко учесть и связку NOT помощи отрицательного веса:

2 из ПРИМЕНЕНИе 1 ЭВМ 1 СЧЕТ -3
что эквивалентно

ПРИМЕНЕНИе AND ЭВМ AND NOT СЧЕТ

Здесь слово с отрицательным весом, если оно встречается в ПОД, «выключает» ответ, делая сумму весов отрицательной. Формулы с весами обладает еще одним важным свойством. Если пороговое значение суммы весов уменьшить, например, на 1, выдача может возрасти, запрос станет «мягче». Это – так называемая «серая» логика (в отличие от черно-белой «Да»/»Нет»). В рассмотренном выше случае с пороговым значением, равным трем, уменьшение его на 1 означает, что поиск будет успешным, если в ПОД встретятся хотя бы два из списка трех ключей. Эта совершенно прозрачная и полезная идея уже не так просто описывается логической формулой. Фактически речь идет о формуле типа

(A AND B) OR (B AND C) OR (C AND A).

Этот прием позволяет также упорядочить (эшелонировать) выдачу по убыванию значения суммы весов.

Далее, при ассоциативном расширении запроса не ясно, какими логическими связками «привязать» слова, взятые по ассоциации (например, в запросе о попе и его собаке, слово «МЯСо»). Снижение порогового значения позволяет отчасти обойти и эту проблему:

3 из ПОПа 1 ДЪЯКОНа 1 СОБАКа 1 ПЕС 1 МЯСо 1

правда, за счет допуска большего шума. На этом возможности линейной весовой формулы, в общем, исчерпываются.

Линейная формула допускает некоторое приближение к нормальной конъюнктивной форме. Рассмотрим пример:

3 из методика 1 преподавания 1 информатики 1 МПИ 3

Это эквивалентно логической формуле:

(методика AND преподавания AND информатики) OR МПИ

17.5. Табличная весовая форма запроса

Для реализации более сложных запросов, в частности, обеих нормальных форм нами предлагается новая, табличная весовая формула запросов. При этом каждому ключу сопоставляется не 1 вес, а строка с весами. Веса для слов, найденных в документе, суммируются по вертикалям и сравниваются с пороговыми значениями столбцов, записанными под столбцами. Если сумма элементов столбца, отвечающих найденным ключам, дальше от нуля, чем пороговое значение, столбец «вырабатывает» пороговое значение, иначе ноль. Далее то же делается со строкой полученных пороговых значений (строка под таблицей). Она суммируется по чис-

лам-результатам столбцов, и если сумма превышает пороговое значение всей формулы (число под ключами), то поиск считается успешным. Рассмотрим примеры. Ниже ключи выделяются большими буквами.

Нормальная конъюнктивная формула запроса, например такова:

у (ПОПа OR ДЪЯКОНа) AND была (СОБАКа OR ПЕС OR ЩЕНОК)

Эту формулу можно описать таблицей:

ПОПа		1 0
ДЪЯКОНа		1 0
СОБАКа		0 1
ПЕС		0 1
————		
#пороги	2	1 1

Символ '#' – признак строки порогов

Для дизъюнктивной формулы:

(ЭЛЕКТРОННая AND ВЫЧИСЛИТЕЛЬНая AND МАШИНа) OR (ПЕРСОНАЛЬНЫЙ AND КОМПЬЮТЕР)

таблица будет такой:

ЭЛЕКТРОННая	1 0
ВЫЧИСЛИТЕЛЬНая	1 0
МАШИНа	1 0
ПЕРСОНАЛЬНый	0 1
КОМПЬЮТЕР	0 1
#пороги	2 3 2

Итак, таблица весов позволяет описать стандартные конъюнктивную и дизъюнктивную формулы, а значит и любое утверждение. Сохраняется и важная возможность снижения пороговых значений.

Отметим очевидное сходство таблично-весовой формы запросов с методом голосующих комитетов в теории распознавания образов. Здесь роль комитетов выполняют группы ключей, имеющие ненулевые веса в одном и том же столбце. Именно они при наличии «кворума» – достаточно большой суммы по столбцу включают пороговое значение как «голос» всего комитета. Особенностью формулы является то, что один и тот же ключ может «голосовать» в нескольких столбцах - комитетах по-разному.

При небольшом навыке мышление лица, ведущего поиск, перестраивается так, что запрос обдумывается непосредственно в терминах весовых

формул, без их интерпретации или предварительной записи с логическими связками. Более того, запрос, вполне понятный в терминах весов, бывает не всегда просто перевести в логическую форму. Это означает, что весовая форма является более общей. С учетом возможности снижения пороговых значений она является и более и гибкой, чем логическая форма. Приведем еще пример (для наглядности веса – нули здесь заменены точками):

```
ИНФОРМАЦИОННый      1 . . . .
ПОИСК               . 1 . . .
ИПС                 1 1 . . .
СУБД                . . . -5 .
БАЗы                . . . . -5
ДАННых              . . . . -5
ШУМ                 . . 1 . .
ПОТЕРи              . . 1 . .
#                 3 1 1 1 -5 -7
```

Это означает:

((ИНФОРМАЦИОННый AND ПОИСК) OR ИПС) AND
(ШУМ OR ПОТЕРи)
AND NOT СУБД AND NOT (БАЗы AND ДАННых)

Здесь отрицательные веса обрабатываются симметрично положительным весам относительно нуля: если модуль суммы активных весов больше модуля порога (внизу столбца), столбец «голосует» отрицательным пороговым значением. Отметим, что аббревиатура ИПС «голосует» в двух столбцах.

17.6. Реализация поисковой системы

Обсуждаемый алгоритм поиска несложно реализовать на любом языке высокого уровня, например, Паскале или Си. Запрос записывается предварительно при помощи любого текстового редактора в отдельный файл, после чего выполняется поиск по текущему каталогу или выбранному файлу. Использование стандартного NORTON – меню вида

Составить запрос
edit zapros.txt выполнить поиск
isktab.exe zapros.txt ! .! > result.txt просмотреть результат
edit result.txt

с переадресацией вывода найденных фрагментов в файл result.txt придает этой нехитрой системе удобный, законченный вид. Впрочем, на менее мощных ПЭВМ сократится лишь сервисная часть.

17.7. Методические рекомендации

Начать можно с проблемы – как найти нужную информацию, если она находится «где-то» на компьютере, на дискете или винчестере. Упомянув о штатных поисковых средствах в средах WINDOWS или NORTON COMMANDER, а также в текстовых редакторах, ставим задачу: информация описана некоторым набором слов, возможны синонимы, так что «лобовой» поиск на вхождение одного из ключей в текст ничего не даст, точнее, даст неприемлемые потери и шум. Так объясняется необходимость записи запроса в виде поисковой формулы.

Для лучшего усвоения целесообразно начинать с линейной весовой формулы и получить при ее помощи простые конъюнкцию и дизъюнкцию. Обязательно следует вначале записывать запрос параллельно в обоих формах: логической и весовой.

Здесь можно указать на множественность форм слова-ключа, по крайней мере, в родном языке. Это подводит к необходимости выделять в слове инвариантную часть и перечислять в запросе далекие по записи словоформы: «идти, пойду». Даже если реальная промышленная поисковая система, с которой будет работать учащийся в дальнейшем, сама выполняет подобное расширение запроса, пропускать этот шаг не стоит, так как это полезно для развития речи учащегося, формирования у него системно-комбинаторного мышления.

Особо следует обратить внимание на свойства синонимии и омонимии «живого» языка, выяснить, к чему это ведет при поиске: синонимия – к потерям, омонимия – к шуму.

Заслуживает интереса и всяческого стимулирования деятельность учащегося по составлению цепочек ассоциаций, связанных с запросом. Именно ассоциации помогают эффективно уточнять и запрос, и поисковую потребность.

Следует также после выполнения поиска попросить объяснить причины, ведущие к потере информации или информационному шуму. Если шум может присутствовать в результате поиска явно, то потери информации можно оценить, только ослабив поисковый запрос (уменьшив пороговый вес и повторив поиск). Вместе с возросшим шумом появятся пропущенные при предыдущем поиске, но нужные фрагменты текста.

Г Л А В А 18. КОМПЬЮТЕР КАК СРЕДСТВО ДОСУГА

18.1. Место игры среди видов деятельности

Игра есть первый из сложных видов деятельности человека. За ней следуют учеба и производительный труд. Основная социальная роль игры – подготовка ребенка к реальной деятельности. Для взрослого игра – средство отдыха, переключения. Основные признаки игры таковы.

1. Отсутствие продукта деятельности – видимого полезного результата. Так, например, если пожилой программист составил хитрую программу, которая оказалась никому не нужна, то это есть игра в работу.

2. Игра – это деятельность с заменителями вещей, со знаками. Для ребенка палочка – это боевой конь.

3. Занимательность, наличие интереса играющего. Работа может быть неинтересной, но ее делают. Неинтересную игру сразу бросают.

Элементы игры могут присутствовать и в учебе, и в любой работе, так легче выполнять необходимую деятельность.

По мнению автора, мировая тенденция развития такова: идет повсеместное увеличение досуга и сращивание игры, учебы и работы. Когда -то, у древних людей эти виды деятельности не разделялись, а теперь снова объединяются.

Рассмотрим компьютер как средство для игры [2].

1. Компьютер обычно не связан напрямую с реальностью (если это не компьютер, управляющий обороной страны или атомной станцией). Значит, с его помощью возможна деятельность, не имеющая сиюминутного результата.

2. Знаковая природа информации, многообразие способов представления на экране одной той же информации отлично согласуется со знаковой природой игры.

3. Очевидна занимательность деятельности, связанной с ЭВМ, если она не превратилась еще в рутину.

Итак, компьютер – идеальное средство для игры.

Рассмотрим, что дает компьютер для игр.

1. Богатство воздействия на органы чувств, многообразие эффектов.

2. Потенциально неограниченная сложность сюжета.

3. Безопасность приключений – после нажатия кнопки RESET все «воскресают», и игра возобновляется. При этом может выработаться опаснейший динамический стереотип поведения: искать кнопку перезагрузки при попадании в аварийную ситуацию, и не только на экране. Это может означать, что в условиях сильного дефицита времени в реальной аварийной ситуации, когда все решают доли секунды, водитель реальной машины, бывший заядлый компьютерный игрок непроизвольно станет нащупывать на пульте управления автомобилем кнопку перезагрузки.

Итак, компьютер отнимает у играющего реальность, особенно реальность движения. «Виртуальная реальность» лишь усиливает точность имитации. Все нередко сводится к 3 клавишам: стрелки вправо и влево, пробел, обычно для выстрела или удара. Если нажатие пробела влечет на экране удар ногой по голове, то увлеченный игрок, особенно малыш, бьет по клавише пробел примерно с таким же усилием, как ногой. Поэтому автор категорический противник такого рода игр. Они уничтожают клавиатуру.

18.2. Основные функции игры

1. Социально-имитационная функция: подмена изучения информатики недидактическими играми. Это все еще встречается у совсем слабых учителей при отсутствии должного контроля. Дисплеи горят, клавиши щелкают, детей не оторвать, с виду – полная компьютеризация. В принципе цель идеальной методики состоит в том, чтобы получить такую картину с помощью специально ориентированных, дидактических игр. Но сегодня их крайне мало. Пример обучающей игры – стрельба по острову с целью научить методу половинного деления (он же метод артиллерийской «вилки»).

2. Функция разгрузки, отдыха, переключения внимания. Выполняется в кабинетах психологической разгрузки на предприятиях. Полезно выделять для этого время в школе для учителей – не информатиков, особенно пожилых скептиков.

3. Функция самоиспытания. Если в реальности нет возможности поуправлять самолетом, то это легко выполнить на компьютерном тренажере. Особенности испытания на компьютере таковы:

– возможная секретность; никто не видит, разбился ли самолет испытуемого или нет, смеяться некому. Дети бывают весьма чувствительны у оценке сверстников;

– возможность анонимного сравнения, если игра сохраняет лучшие результаты – рекорды;

– безопасность. Проверять себя в реальном бою с чемпионом-каратистом как-то не хочется, а на экране – сколько угодно;

– сочетание работы с игрой; не просто скучное тестирование, а борьба за очки, возможно, в виде соревнования;

– оперативность обучения; можно сразу же повторить упражнение.

4. Функция психической тренировки и развития таких черт характера, как настойчивость, решительность и при специально поставленном обучении даже ответственность за свои действия (не бросать штурвал до конца испытания). Так учат, например, летчиков на тренажерах. Здесь важно различать два отношения к игре, две различных цели: тренировка или самоутверждение через псевдокомпенсацию, бегство от реальности. Для второй цели характерно то, что игрок предпочитает одну, от силы две

игры, именно те, в которых у него что-то получается. Он «зацикливается» на этой игре и игнорирует те, где успехов меньше. Нормальный, тренирующийся игрок будет с интересов пробовать новые игры.

5. Функция соревнования. Она присуща и безкомпьютерным играм.

6. Функция образования. Это постепенно становится не пожеланием, а реальностью. Так, игра-путешествие по древнему Средиземноморью есть, по сути, путешествие по базе данных древних городов. Но игрок об этом может не догадываться.

В принципе компьютер позволяет игроку управлять сюжетом, но здесь огромны трудности с разработкой такого гибкого сюжета, с массой возможных сюжетных линий. Конечно, учащемуся интересно знать, что будет, если в момент выстрела Дантеса толкнуть под локоть, нажав все ту же клавишу – пробел. Важно, что пробел здесь – именно промах, а не выстрел!

Внесение в игру учебной информации может быть искусственным и естественным. В первом случае игра время от времени прерывается, и появившийся «Волшебник» занудно просит сообщить что-нибудь поучительное, например, чему равен синус 90^0. Важно, что эти сведения не имеют к игре никакого отношения. Так довольно легко можно нагрузить учебой любую игру, написав небольшую резидентную программу, которая прерывает любую выполняемую программу, чтобы задать свои вопросы о синусах. Но минус здесь в том, что учеба оказывается просто помехой на пути к истинной цели – игре, и вызывает у игрока только раздражение.

Ситуация резко меняется, если в игре нужно определить угол для стрельбы с наибольшей дальностью. Здесь вопрос о синусе оказывается не только уместным, но уже и подсказкой, так как половина от прямого угла – как раз нужный угол наклона траектории в точке старта. Этот прием сращивания игры и учебы гораздо сложнее, но эффективнее с точки зрения опоры на интерес играющего.

7. Функция вознаграждения. Здесь учитель может сказать честно: Дети, 25 минут занимаемся, затем играем. Как и в предыдущем примере, учеба оказывается помехой на пути к игре. Правда, если в учебе есть собственные интересные моменты, до игры дело может не дойти.

8. Функция игры как объекта изучения. Для учителя ситуация очевидна: прежде всего он должен оценить дидактические и воспитательные функции игры. Но замечено, что сильные обучаемые довольно быстро утрачивают интерес к диалогу с игрой и пытаются понять, как устроена программа, реализующая игру, или просто пишут собственные аналоги. По результативности с точки зрения профессионализма выполнения программы такая работа почти безнадежна: надо в совершенстве владеть графикой, программированием параллельных процессов для нескольких взаимодействующих персонажей и т.д. Но сама тенденция крайне важна и говорит о творческих задатках личности игрока. Идеальным результатом здесь явля-

ется написание учащимся хотя бы сильно упрощенного аналога игры. Полезно посоветовать обучаемому не гнаться сразу за внешними эффектами, а закодировать события какими-то знаками, скажем, столкновение автомобиля с препятствием передавать просто командой BEEP. Гораздо важнее логика и моделирование.

9. Функция воспитания. То, что дети в определенном возрасте проходят через конфликтные игры – в войну, в казаков-разбойников, отражает наличие атавизмов в человеческой психике, того, что роднит человека с агрессивной обезьяной, умевшей к тому же изготавливать оружие. Затем происходит социальная компенсация этих черт личности, ребенок в нормальном развитии «перерастает» эти игры. Заметим поэтому, что агрессивные компьютерные игры могут возвращать развитие ребенка на шаг, а то и на два назад, снова к агрессии, бегству и т.д., закреплять и даже развивать такие реакции.

Не стоит забывать об особенно действенном, ненавязчивом воспитании. В эпоху холодной войны появились игры типа ядерной бомбежки Москвы. Такая игра потихоньку формирует определенное отношение к этому городу и его населению.

Воспитывающая сила компьютерных игр, их воздействие на человека на порядок сильнее, чем у кино, телевидения, книг. Известно, что заинтересованный читатель или зритель подсознательно отождествляет себя с одним из персонажей. Но кино – это просто наблюдение, а компьютерная игра – возможность воздействия. Степень отождествления себя с персонажем здесь на порядок выше. Действия, даже если это просто нажатия клавиш, происходят в контексте некоторого сюжета, и закрепляются вместе с ассоциативными связями. Вот почему фиксация на агрессивных играх не проходит бесследно для личности.

Здесь важно различать два уровня деятельности: знакомство с игрой и интенсивный диалог, фактически тренинг. Знакомство безопасно и может вызвать отвращение к злу, как и хороший фильм. Но не в этом цель игры. А при интенсивной игре происходит вдалбливание в сознание и особенно в подсознание ассоциативных рядов, характерных для данной игры.

10. Диагностирующая функция игры тесно связана с предыдущей. Если подросток и тем более юноша продолжает увлечено играть в агрессивные игры, это признак патологического развития личности; к игроку надо присмотреться. Возможно, что это не асоциально ориентированный тип, а слабый духом ребенок, компенсирующий свою беспомощность успехами в иллюзорном мире компьютерной игры. Это уже повод для беспокойства школьному психологу.

18.3. Классификация игр

Ввиду многообразия игр возможна многопараметрическая классификация. Ее цель – облегчить учителю дидактическую оценку конкретной игры. Одна игра может быть носителем ряда признаков.

1. По виду информации можно различить графические, цветовые, текстовые, звуковые программы-игры. Современные игры обычно используют весь спектр типов информации. Здесь стоит оценить, какой тип данных преобладает в данной игре.

2. По темпу различают игры позиционные и динамические. Позиционная игра (шахматы, бизнес) привлекает мышление, вторую сигнальную систему. Чисто динамическая игра проходит на уровне рефлексов «бей – беги». Сопутствующие состояния игрока – это досада, страх, злость и т.д.

3. По количеству участников игры бывают сольные, парные и групповые. Первых гораздо больше. Но сравнительно недавно появились групповые игры, использующие локальную сеть.

4. По сложности управления можно выделить игры под условным названием «вправо – влево – пробел» и более сложные, например, шахматы.

5. По сложности сюжета можно выделить полюса: игры – поединки и игры – путешествия.

6. По типу взаимодействия игроков различаются игры – конфликты (бой с чудовищем), игры нейтральные (TETRIX), игры кооперативные («монополия»). Возможны комбинации.

7. По языку сообщений различаются игры «переводные» и первичные. При переводе игры на родной язык нередки ошибки, так как делается это обычно пиратскими методами, а не фирмами-разработчиками.

8. По наличию цели можно отличить игры с готовой целью (шахматы) и игры без видимой цели (редактор орнаментов). В последнем случае цель привносится игроком или преподавателем.

Основной вопрос для педагога – чему данная игра учит и что воспитывает, учит ли она вообще чему-нибудь.

18.4. Метод обратного анализа позиционных игр

Шашки-шахматы. Вспомним, как рассуждает новичок при игре в шашки или шахматы: «я туда, он сюда». Но почему именно «туда и сюда»? Ответ очевиден: так я выигрываю пешку или шашку, улучшаю позиции фигур, а это полезно.

Но цель игры – не выиграть пешку, а поставить мат. Для этого порой приходится идти на жертвы и это, кстати, оценивается как красивая игра. Легко видеть, что попытка анализировать игру перебором всех возможных ходов ведет к моментальному комбинаторному взрыву числа вариантов.

Если на каждый из 10 ходов белых есть 10 ходов черных, то при анализе игры на глубину в 10 ходов придется просмотреть 20^10 вариантов. Это слишком много и для нынешних и для будущих компьютеров. Но люди в шахматы все же играют и порой неплохо. Как выясняется, дело отчасти в том, что люди оценивают не ходы, а позиции, относя их к определенным классам. Позиций много, но гораздо меньше, чем цепочек ходов – переходов между позициями. По мере роста мощности ЭВМ, особенно памяти, и компьютеры научились оценивать позиции, но двигаясь от конца. Чтобы понять основные идеи такого анализа, рассмотрим простые примеры.

Шашка – в дамки. На шашечной доске стоит шашка. Сколько для нее существует путей в дамки? Анализ «в лоб» ведет примерно к удвоению числа путей за каждый ход и неочевидно, как учитывать края доски. Но задача легко решается с конца. Сопоставим каждой последней клетке (ряд превращения в дамки) число 1: один путь. В нее можно попасть из клеток предыдущего ряда, одним (на краю) или двумя способами и так далее:

1 1 1 1 1 1 1 1 число путей последнего ряда
1 2 2 2 2 2 2 2 предпоследний ряд
3 4 4 4 4 4 4 2
3 7 8 8 8 8 8 6
? заданная позиция шашки

Отметим, что число путей есть величина порядка 28=256, а число позиций, положений шашки здесь равно всего лишь числу черных клеток, т.е. 32.

Игра Баше. Эта игра рассмотрена в первом учебнике информатики под ред. А.П. Ершова. Дана куча камней. Игроки берут поочередно один или два камня. Проиграет тот, кто берет последний камень. Нужно разработать наилучшую стратегию игры.

Снова будем анализировать игру с конца, отождествив себя с одним из игроков. Если при моем ходе остался один камень, я проиграл. Отмечаем эту позицию знак минус как проигранную. Смещаемся влево. Если осталось два камня, а беру один и перевожу противника в проигранную позицию. Отмечаем позицию знаком плюс. Если осталось три камня, я беру два и перевожу противника в позицию с одним камнем при его ходе, т.е. проигранную им:

4 3 2 1 число камней
? + + - признак позиции

Дальше все чуть сложнее и интереснее. Если камней 4, то есть две возможности: взять один камень или два. В обоих случаях я перевожу про-

тивника в выигранную им позицию. Значит, отмечаем ситуацию «4 камня» знаком минус. Если камней осталось 5, то существует ход (взять один камень) переводящий противника в проигранную позицию – 4 камня при его ходе. Если камней осталось 6, то надо взять два и перевести противника в проигранную позицию с 4 камнями. Дальше уже видно по индукции общее правило:

8 7 6 5 4 3 2 1 число камней
+ - + + - + + - признак позиции

«Соль» приведенного рассуждения в том, что оно опирается на сведение новой позиции к уже оцененным, а не на перебор в стиле «я – туда, он – сюда» с удвоением вариантов на каждом полуходе.

Примерно так же оцениваются с помощью ЭВМ шахматные окончания. В память ЭВМ заносятся все (!) возможные позиции с данным или меньшим числом фигур (на сегодняшних компьютерах уже можно рассмотреть до 7 фигур). Прямым просмотром часть из позиций оценивается как матовые или ничейные. Далее остальные позиции поочередно сводятся к уже оцененым. Заметим, что ЭВМ не выигрывает фигуры, пешки или пространство, она просто «тупо» перебирает позиции и не имеет понятия о возможном существовании общих, индуктивных правил типа того, которое для нас, людей стало очевидно при обратном анализе игры Баше.

Упражнения. Модифицируйте стратегию в игре Баше на следующие случаи:

1) можно брать до трех камней;
2) тот, кто берет последний камень, выигрывает;
3) играют трое.

18.5. Пример динамической игры

Рассмотрим следующую игру. Пусть волк гонится за зайцем (а ракета – за самолетом). Заяц при нажатии клавиш со стрелками может прыгать в сторону. Пусть волк бежит прямо на зайца (в ракетной технике такой способ преследования называется «собачья погоня»: куда смотрю, туда бегу). Составить программу – игру. Игрок может управлять зайцем, моментом и направлением его прыжков.

```
SCREEN 9                    'погоня волка за зайцем
HZ = 10                     'длина прыжка зайца
HV = 5                      'расстояние захвата зубами
DT = .001                   'шаг по времени
XZ = 100                    'начальные координаты зайца
YZ = 100 '
```

```
      VV = 10                        'скорость волка
10 A$ = INKEY$                       'клавиша болельщика зайца
      L = LEN(A$)                    'сколько байт введено
      IF L = 0 THEN 20               'ничего не нажато
      IF L = 1 THEN 20               'не стрелка – пропустить
      A = ASC(MID$(A$, 2, 1))        'второй код клавиши
      IF A = 77 THEN XZ = XZ + HZ 'прыжок зайца вправо
      IF A = 75 THEN XZ = XZ - HZ 'прыжок влево
      IF A = 80 THEN YZ = YZ + HZ ' вниз
      IF A = 72 THEN YZ = YZ - HZ ' вверх
20 DX=XZ - XV                        'волк направлен на зайца
      DY=YZ - YV '
      DL=SQR(DX^2 + DY^2) '
      XS = XV                                'старые координаты волка
      YS = YV                                '
      XV = XV + DX / DL * VV * DT            'перемещение волка за DT
      YV = YV + DY / DL * VV * DT '
      PSET(XZ, YZ), 15                       'где заяц (белый –15)
      LINE (XS, YS) - (XV, YV), 7            'где волк (серый – 7)
      R = (XV - XZ)^2 + (YV - YZ)^2          'расстояние^2 волк-заяц
      IF R < HV^2 THEN 29                    'если захватил, то и съел
      GOTO 10                                'следующий шаг волка
29 BEEP                                      'писк съедаемого зайца
      END                                    'конец погони
```

Упражнения:

1) Учтите конечность экранных координат.

2) * (сложное задание): учтите инерцию (массу) волка.

З А К Л Ю Ч Е Н И Е

Итак, рассмотрены специфические и универсальные виды деятельности педагога и учащихся в связи с использованием ПЭВМ. Основной вывод состоит в том, что компьютер может существенно изменять стиль деятельности, используемые понятия, приемы работы и т.д. Конечно, там, где деятельность изменяется не сильно (музыка), люди быстрее осваивают компьютер, но зато там, где изменения заметны, более заметны и результаты (графика). При этом возможен и своеобразный регресс, утрата навыков работы без компьютера.

Популярное возражение против компьютеризации, например, таково: «С этими компьютерами дети разучатся считать, а умение считать – признак образованного человека, элемент его культуры». С первой частью автор согласен: не только считать, но и писать (рисовать буквы) и чертить карандашом и линейкой дети действительно разучатся, точнее, не научатся, и чем скорее, тем лучше. Что же касается образованности и культуры, то дворянин средних веков, человек образованный и культурный, должен был уметь скакать верхом и совать в живот ближнему своему острый металлический предмет, именовавшийся шпагой. А много ли сегодня наездников и фехтовальщиков среди хороших вычислителей, учителей математики? Ушла эпоха – и унесла с собой свою культуру, ее специфику.

Итак, понятие культуры глубоко исторично, изменчиво, и дети следующего поколения, как и, будем надеяться, их учителя наверняка обогатят общечеловеческую культуры новыми достижениями, полученными при помощи ЭВМ.

Л И Т Е Р А Т У Р А

1. Амелькин В.В., Садовский А.П. Математические модели и дифференциальные уравнения. Мн., 1982.
2. Бейда А.А. Королев С.А. Игровые задачи на ЭВМ. Мн., 1991.
3. Белошапка В., Лесневский А. Основы информационного моделирования // ИнфО, 1989, № 3.
4. Бестужев-Лада И.В. К школе XXI века: Размышления социолога. М., 1988.
5. Бочкин А.И. Методика преподавания информатики: Учеб. пособие. Мн., 1998.
6. Брушлинский А.В., Пономарев В.А., Мышление и общение. Мн., 1990.
7. Буза М.К., Певзнер Л.В., Хижняк И.А. Операционная система WINDOWS и ее приложения. Мн., 1997.
8. Вейценбаум Дж. Возможности ЭВМ и человеческий разум. М., 1980.
9. Выявление экспертных знаний (процедуры и реализации) / О.И. Ларичев, А.И. Мечитов, Е.М. Мошкович, Е.М. Фуремс. М., 1989.
10. Глушков В.М. Основы безбумажной информатики. М., 1987.
11. Громко И.М. Введение в страну ЭВМ. М., 1989.
12. Денинг В., Эссиг Г., Маас С. Диалоговые системы «человек–ЭВМ»: адаптация к требованиям пользователя. М., 1984.
13. Джорж Ф. Основы кибернетики. М., 1984.
14. Дуайер Д. APPLE – классы завтрашнего дня: Чему мы научились // ИнфО, 1995, № 3.
15. Ефимов А.Н. Информационный взрыв: проблемы реальные и мнимые. М., 1985.
16. Житомирский В.Г., Долгий А.В. Мини- и микроЭВМ. Использование в учебном процессе. Свердловск, 1987.
17. Зарипов Р.Х. Машинный поиск вариантов при моделировании творческого процесса. М., 1983.
18. Извозчиков В.А., Ревунов А.Д. Электронно-вычислительная техника на уроках физики в средней школе. М., 1988.
19. Иванов В.Л. Электронный учебник: системы контроля знаний // ИнфО, 2002, № 1.
20. Клейман М. Школа будущего – компьютеры в процессе обучения. М., 1987.
21. Ковалев М.М., Курбацкий А.Н., Листопад Н.И. Концепция информатизации образования. Мн., 1991.
22. Кондратов А. Электронный разум. М., 1987.
23. Косневски Ч. Занимательная математика и персональный компьютер. М., 1987.
24. Кулакин Е.Д. и др. Микрокалькуляторы в курсе математики. М., 1989.

25. Кушниренко А.Г., Лебедев Г.В., Сворень Р.А. Основы информатики и вычислительной техники. М., 1993.
26. Кушниренко А.Г., Эпитектов М.Г. Активные гипертекстовые среды на уроках информатики // ИнфО, 1994, № 1.
27. Леонтьев А.Н. Деятельность, сознание, личность. М., 1975.
28. Лернер И.Я. Процесс обучения и его закономерности. М., 1980.
29. И.Ликеш, Й.Ляга. Основные таблицы математической статистики. М., 1985.
30. Малышева М.А. Дети и Лого // ИнфО, 1995, № 5.
31. Машбиц Е.И. Психолого-педагогические проблемы компьютерного обучения. М., 1988.
32. Милитарев В.Ю., Яглом И.М. Информационная культура эпохи НТР // Информатика и культура. Новосибирск, 1990.
33. Мичи Д., Джонсон Р. Компьютер – творец. М., 1987.
34. Молибог А.Г. Вопросы научной организации труда в высшей школе. М., 1971.
35. Молибог А.Г. Програмированное обучение. М., 1967.
36. Невуева Л., Сергеева Т. О перспективных тенденциях разработки ППС // ИнфО, 1990, № 5.
37. Основы информатики и вычислительной техники: В 2 ч. / А.П. Ершов, В.М. Монахов, А.А. Кузнецов и др. / Под ред. А.П. Ершова, В.М. Монахова. М, 1985.
38. Павловский А.И. Использование вычислительной техники в учебном процессе: Тексты лекций. М., 1990.
39. Паўлоўскі А.І. Інфармацыйная культура настаўніка інфарматыкі // Нар. асвета, 1992, № 8.
40. Пасхин Е.Н., Митин Л.И. Автоматизированная система обучения «ЭКСТЕРН». М., 1985.
41. Пейперт С. Переворот в сознании: Дети, компьютеры и плодотворные идеи. М., 1989.
42. Персональный компьютер – рабочее место профессионала / Под ред. Моисеева Н.Н. М., 1989.
43. Принятие интеллектуальных решений в диалоге с компьютером. М., 1990.
44. Приобретение знаний / Под ред. С. Осуги, Ю. Саэки. М., 1990.
45. Рубинштейн С.Л. О мышлении и путях его исследования. М., 1958.
46. Романовский Т. Микрокалькулятор в школе. Рига, 1987.
47. Реальность и прогнозы искусственного интеллекта: Сб. статей / Под ред. и с предисл. В.Л. Стефанюка. М., 1987.
48. Сайков Б.П. Excel: создание тестов // ИнфО, 2001, № 9.
49. Самарин Ю.А. Очерки психологии ума. Особенности умственной деятельности школьников. М., 1962.

50. Справочник по прикладной статистике / Под ред. Э. Ллойда, У. Ледермана. М., 1989.
51. Талызина Р.Ф. Теоретические основы программированного обучения. М., 1969.
52. Талызина Р.Ф. Формирование познавательной деятельности младших школьников. М., 1988.
53. Теннат-Смит Дж. Бейсик для статистиков. М., 1988.
54. Терещенко Л.Я. Управление обучением с помощью ЭВМ. Л., 1981.
55. Трояновский В.М. Автоматизированный контроль знаний о системе взаимодополняющих понятий // ИнфО, 2002, № 3.
56. ЭВМ и непрофессиональные пользователи. Организация взаимодействия. М., 1989.
57. Хантер Б. Мои ученики работают на компьютерах. М., 1989.
58. Храмцов П. Моделирование и анализ работы информационно-поисковых систем Internet // Открытые системы, № 6(20)/96. С. 46-55.
59. Элти Дж., Кумбс М. Экспертные системы: концепции и примеры. М., 1987.
60. Эсаулов А.Ф. Активизация познавательной деятельности студентов. М., 1982.

С О Д Е Р Ж А Н И Е