МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ УКРАИНЫ
КРИВОРОЖСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

КОМПЛЕКС ПРОГРАММНЫХ СРЕДСТВ ПЕДАГОГИЧЕСКОГО И ПРОИЗВОДСТВЕННОГО НАЗНАЧЕНИЯ

Дипломная работа
студента группы МИ-93-2
С.А. СЕМЕРИКОВА

Руководитель
доцент, к.т.н., с.н.с.
А.П. ПОЛИЩУК

КРИВОЙ РОГ
1998

СОДЕРЖАНИЕ

Введение

Данная дипломная работа представляет собой своего рода «отчёт о проделанной работе», интегрировав в себе основные результаты работы дипломанта в области программирования в период обучения в педвузе. Структурно работа делится на три части.

В Части 1 - «Разработка системы обучения и проверки знаний» - рассматриваются понятие педагогического контроля и оценки результатов учебно-познавательной деятельности учащихся (как школьников, так и студентов), принципы построения обучающих систем, возможные подходы к классификации систем обучения. Выработанные требования к обучающей системе воплощены в инструментальной системе обучения и тестирования, рассмотрению которой посвящён остаток первой части. Приводятся основные результаты работы с последней по результатам апробации на студентах потоков специальности «Математика и информатика».

Часть 2 - «Использование СУБД для решения задачи расписания на примере расписания занятий» - посвящена такой актуальной проблеме администрирования учебной деятельности, как составление расписания. В этой части формулируются основные требования к системам управления базами данных, рассматриваются терминология и модели построения СУБД; акцент при этом делается на реляционной модели, для которой формулируются правила алгебры отношений Кодда. В практическом разделе рассматриваемой части описаны особенности использования настольной СУБД семейства xBase для построения и обслуживания БД «Расписание занятий» в период вычислительной практики.

Последняя, третья часть - «Управление измерительным оборудованием» рассматривает данную задачу на примере программно-аппаратного комплекса для гамма-спектрометрии, доработанного дипломантом по заказу изотопной лаборатории КМК «Криворожсталь». По результатам внедрения сформулированы рекомендации по работе с прилагаемой программой обработки спектров от анализатора АМА-03Ф, приведено описание возможных отклонений от нормального режима работы и способов их устранения.

1. Разработка системы обучения и проверки знаний

1.1. Педагогический контроль и оценка результатов учебно-познавательной деятельности учащихся

Слово **контроль** (от французского *controle*) имеет несколько значений. В дидактике его понимают как наблюдение и проверку успеваемости учащихся.

Цели, содержание и методы обучения определяют уровень и результаты контроля.

Структура контроля знаний состоит из проверки (выявления) и оценки (процесса и результата).

Функции контроля знаний делятся на специфические (контролирующие) и общие. Специфические функции - это выявление и оценивание знаний. К общим функциям относятся:

- *обучающая,*
- *стимулирующая,*
- *воспитательная и*
- *диагностическая.*

Объектом контроля в процессе обучения является:

1) знание учащимися основных категорий, правил, принципов, закономерностей изученного предмета, а также фактов и событий в их тесной взаимосвязи;
2) их умения и навыки оперировать полученными знаниями, включая суждения, доказательства, умение находить оригинальное и правильное решение учебно-познавательных задач.

Основными принципами контроля знаний являются объективность и регулярность.

Принято выделять: **предварительный, поточный (текущий), тематический, периодический, итоговый контроль и самоконтроль**.

Предварительный контроль выполняет, в основном, диагностическую функцию и проводится с целью выявления определенного запаса знаний учащихся.

Поточный контроль проводится на всех этапах процесса обучения и дает информацию о том, как учащиеся усваивают учебный материал.

Тематический контроль проводится с целью проверки и оценки знаний учащихся по каждой теме учебного предмета.

Периодический контроль проводится с целью определения, насколько успешно учащиеся овладевают системой знаний, умений и навыков.

Самоконтроль позволяет каждому учащемуся оценить правильность своих действий, скорректировать их, внести необходимые поправки.

Итоговый контроль - это экзамены и зачеты, в ходе которых происходит всесторонняя проверка знаний, умений и навыков учащихся.

Методы контроля знаний - это способы общей деятельности преподавателя и учащегося, направленной на выявление и оценивание знаний, а также фиксацию их результатов.

Традиционная **классификация методов контроля знаний** базируется на учете цели проверки знаний, умений и навыков.

Существуют следующие методы контроля:

1) *наблюдение* за различными видами деятельности учащихся;
2) устная проверка (*опрос*);
3) *письменная проверка* (контрольные работы, письменные домашние задания etc.);
4) *проверка практикой* (практические и лабораторные работы);
5) *тестовая проверка* (использование тестов и контрольных заданий).

Оценка - это количественный показатель качества результатов учебно-познавательной деятельности учащихся.

Единые требования к оценке знаний, умений и навыков формулируются в виде соответствующих критериев и норм.

Критерий оценки - это мера для определения уровня усвоения знаний, умений и навыков.

Основой критерия оценки является:

1. *характер усвоения материала* - объем, полнота, правильность и точность знаний, уровень их осмысления, системность, умение использовать изученное на практике и в нестандартных ситуациях;
2. *особенности выполнения работы* - оформление, темп, старательность и т.д.;
3. *качество ответа учащегося* - обоснованность, логичность, последовательность изложения, степень самостоятельности в суждениях, культура речи и т. д.

В соответствии к этим критериям определяются **нормы оценок** - отдельно для каждой учебной дисциплины.

1.2. Принципы построения и классификация обучающих систем

Одним из направлений повышения эффективности компьютерной поддержки учебного процесса является предоставление преподавателю - предметнику простой и удобной в обращении интерактивной инструментально-исполнительной системы с набором обучающих материалов и тестирующих заданий для проверки знаний.

Наличие подсистемы тестирования и учета успеваемости позволяет преподавателю решить проблему одновременного сплошного опроса по всем разделам изучаемых тем (при проведении занятий в компьютерном классе). Ученик, в свою очередь, получает возможность самостоятельного повторного тестирования для исправления не удовлетворяющих его оценок по тем или иным разделам уже пройденных тем.

Исторически попыток создания таких систем было немало. Первоначально разработка обучающих систем осуществлялась в крупных научных и учебных центрах. В США, например, такими центрами были Дартмундский колледж, Иллинойский и Стэнфордский университеты и фирма IBM. В Великобритании основные проекты по компьютеризации обучения осуществлялись в университетах Глазго и Лидса, а также в Эдинбургском колледже. У нас первые обучающие системы были разработаны в Рижском политехническом институте, Белорусском университете, в ВЦ АН СССР в Москве, в научных центрах Киева и т. д. Первые такие системы в большинстве своем были ориентированы на обучение программированию, поэтому в них компоненты

программного обеспечения компьютера использовались в учебных целях. К настоящему времени почти все созданные в 60-е годы обучающие системы, кроме PLATO (Programmed Logic for Automated Teaching Operation - программированная логика для автоматизированных обучающих операций), потеряли свое практическое значение. По своим дидактическим возможностям они мало чем отличались от систем, использовавших простейшие технические средства обучения и предполагавших жесткую, практически исключающую диалог детерминацию деятельности учащихся. Но именно первые разработки стимулировали интерес к компьютерному обучению, активизировали работу по созданию обучающих систем.

Если начало 60-х годов ознаменовалось оптимистическими заявлениями, что в ближайшие 10-15 лет компьютер займет ведущее положение в учебном процессе не только вузов, но и школ, то уже к концу 60-х годов стало ясно, насколько необоснованны были эти предположения. Разработка эффективных обучающих систем потребовала решения весьма сложных психолого-педагогических проблем, изучению которых препятствовали и такие факторы, как довольно высокая стоимость эксплуатации компьютерных систем обучения, а также предубежденность учителей. Учителя могли использовать компьютеры в обучении только с помощью программистов, а те, как правило, не имели достаточных знаний в области психологии и педагогики. Как следствие, ни учителя, ни программисты не могли найти общий язык, и их диалог напоминал разговор глухих.

Так продолжалось до начала 70-х годов, когда резко возросло количество выпускаемых компьютеров и стала снижаться их стоимость. Все это сопровождалось неразберихой в совместимости программного обеспечения компьютеров различных систем, что также укрепило настороженное отношение к развитию компьютерного обучения педагогической общественности.

В начале 80-х годов, по мнению специалистов, наступил новый этап применения компьютера в сфере образования. Отличительные особенности этого этапа определяются появлением недорогих, удобных в эксплуатации персональных компьютеров, изменивших представление о компьютере как о сложном и экзотическом устройстве, работать с которым могут лишь избранные.

Развитие технического и программного обеспечения персональных компьютеров привело к расширению возможностей их использования в обучении. Во-первых,

резко возросли способы предъявления учащимся информации. Во-вторых, были созданы языки программирования, близкие к естественным, а также графические и эскизные средства связи с ЭВМ. В-третьих, совершенствование инструментария (авторских систем), существенно облегчающего составление обучающих программ, значительно расширило круг лиц, которые могут составлять такие программы, даже не имея специальной подготовки.

В конце 80-х годов стали разрабатываться пакеты программ, охватывающие значительные части учебного курса. Данное направление весьма плодотворно, поскольку такие программы направлены на достижение не только ближайших, но и отдаленных целей обучения, кроме того, в них более полно реализуется индивидуальный подход, а также рефлексивное управление обучением, предполагающее построение модели обучаемого. Правда, свыше 80% из существующих программ, а по другим данным еще больше, не обеспечивают достижения даже ближайших учебных целей.

Не случайно в последнее время столь возрос интерес к разного рода инструментальным оболочкам - Framework, Symphony, Eureka, LinkWay etc, однако реализация в них учебных задач происходит не всегда адекватно, а зачастую их вообще нельзя втиснуть в рамки этих систем. Другая их разновидность - авторские оболочки - страдают массой недостатков, вызванных тем, что большинство из них, как правило, программы-однодневки, предназначенные для если не разового использования, то для узкого круга пользователей. Более того, даже с такими системами познакомиться практически невозможно, так как они обычно не выходят за рамки учебного заведения, в котором они разработаны.

Обычным требованием к обучающей системе является то, что инструментальная часть системы должна обслуживать функции создания, удаления, корректировки обучающих материалов и тестирующих заданий, чтобы каждый преподаватель мог формировать изучаемый курс под свои методы и конкретные условия работы. В то же время поставка «пустых» оболочек, даже очень совершенных, малоэффективна, так как процесс их наполнения слишком трудоемок и длителен. Например, вложенный нами в разработанную оболочку элементарный курс информатики включает 100 уроков и 1000 тестов общей емкостью свыше 1 Мбайт.

Конечно же, такая система должна быть хорошо документирована, и иметь хотя бы примитивную систему помощи, желательно - контекстно-зависимой. Она долж-

на быть открытой и уметь обмениваться данными с другими программами, не принуждая пользователя составлять обучающий материал только в ней. Обязательным требованием является наличие интуитивно понятных органов управления, например, в стандарте Commom User Access фирмы IBM.

Компьютерные программы, разработанные для применения учебного процесса, условно можно разделить на два класса.

1. Компьютерные обучающие программы (КОП) которые могут служить пособием для преподавателя и учащегося, хотя они могут не обучать, а, например, комментировать, выдавать справочную информацию или как-то иначе помогать преподавателю или учащемуся.
2. Программы предназначенные для разработки КОП, т.е. всякого рода инструментальные системы, предметно-ориентированные среды, прикладные пакеты (ПП) и т.п.

Далее мы будем использовать собирательный термин: компьютерная программа учебного назначения (КУН). КУН - это программное средство, специально разработанное или адаптированное для применения в обучении.

КУН могут быть классифицированы по назначению следующим образом:

- компьютерные учебники - КУ;
- предметно-ориентированные среды (микромиры, моделирующие программы, учебные пакеты) - ПОС;
- лабораторные практикумы - ЛП;
- тренажеры;
- контролирующие программы - КП;
- справочники, базы данных учебного назначения - УБД.

Компьютерный учебник - это программно-методический комплекс, обеспечивающий возможность самостоятельно освоить курс или его большой раздел. КУ соединяет в себе свойства обычного учебника, справочника, задачника и лабораторного практикума. Данные в компьютерном учебнике, как правило, организованы в виде гипертекста.

Предметно-ориентированные среды - это учебный пакт программ, позволяющий оперировать с объектами определенного класса. Среда реализует отношения

между объектами, операции над объектами, обеспечивает наглядное представление объектов и их свойств.

Лабораторный практикум служит для проведения наблюдения над объектами, их взаимосвязями или некоторыми их свойствами, для обработки результатов наблюдений, для их численного и графического представления и для исследования различных аспектов использования этих объектов на практике.

Тренажеры служат для отработки и закрепления технических навыков решения задач. Они обеспечивают получение информации по теории и приемам решения задач, тренировку на различных уровнях самостоятельности, контроль и самоконтроль. Предоставляют вспомогательные средства (калькулятор, таблицы, автоматическое решение подзадач и т.п.). Как правило, включают режимы: *теория, демонстрация примеров, работа с репетитором, самостоятельная работа, самоконтроль.*

Контролирующие программы - это программные средства, предназначенные для проверки (оценки) качества знаний. Требования к контролирующим программам (КП):

- КП должны предоставлять возможность ввода ответа в форме, максимально приближенной к общепринятой;
- КП должны обеспечить адекватный анализ ответа, отличающий опечатку от ошибки и распознающий правильный ответ в любой из эквивалентных форм его представления;
- КП не должны предлагать учащемуся выбрать ответ из списка, содержащего заведомо неверные утверждения;
- КП должны быть обеспечены фиксация результатов контроля, их сбор, распечатка и статистический анализ.

Контролирующие программы специально рассчитаны на проведение текущего или итогового опроса учащихся. Они позволяют установить необходимую обратную связь в процессе обучения, способствуют накопляемости оценок, дают возможность проследить в динамике успеваемость каждого учащегося, соотнести результаты обучения с трудностью предлагаемых заданий, индивидуальными особенностями обучаемых, предложенным темпом изучения, объемом материала, его характером.

Справочники, базы данных учебного назначения предназначены для хранения и предъявления учащемуся разнообразной информации справочного содержания.

Для них характерна иерархическая организация материала и быстрый поиск информации по различным признакам или по контексту. Требования к компьютерным справочникам, базам данных учебного назначения:

В них

- должна использоваться стандартная форма представления знаний;
- должна быть обеспечена возможность получения необходимой справки из любого места программы;
- должна быть обеспечена возможность сохранения и вывода полученной справки;
- должны быть выдержаны стандартные требования к интерфейсу;
- должна быть обеспечена возможность получения комплексных справок со сведениями из нескольких разделов курса;
- количество информации на экране не должно превышать норм, определяемых психолого-педагогическими и гигиеническими требованиями.

Общие требования к обучающим программам:

1. Эффективность компьютерной поддержки: экономия времени учащегося, глубина трактовки вопросов программы, предоставление возможностей для создания методик преподавания и модернизации содержания учебных курсов.
2. Методические свойства: отсутствие грамматических и синтаксических ошибок, простота освоения программы и простота работы с ней, соответствие стандартным требованиям к интерфейсу, открытость, т.е. возможность расширения круга задач и т.д.
3. Качество экранного дизайна: соблюдение требований к интерфейсу, обоснованность цветовых решений, оптимальное количество информации на экране.
4. Экономическая обоснованность: круг предлагаемых пользователей, конкурентоспособность, открытость для модификаций и дополнений последующими версиями.

1.3. Описание инструментально-исполнительной системы обучения и тестирования TUTOR

Система TUTOR в одном из своих вариантов разработана в среде Borland C++ 4.0 как полноценное Windows-приложение и в этой версии может функционировать на IBM-совместимых PC-386 и выше под управлением Windows for Workgroups 3.11 + (русская версия). Это означает, что она может выполняться одновременно с другими программами, не мешая им, а человек, работающий с ней, может переключаться в нее только при необходимости. Так, скажем, при изучении курса численных методов типичными программами, совместно работающими на на ЭВМ, были

(a) Диспетчер программ (progman) - оболочка, из которой происходит запуск всех остальных программ,

(b) Pri/Sec (ruslat) - русификатор Windows for Workgroups,

(c) Borland C++ 3.1 for Windows (bcw) - интегрированная среда языка C++, в которой происходило выполнение заданий по ЧМ, оформленных в виде классов, представляющих абстрагированные модели объектов исследования.

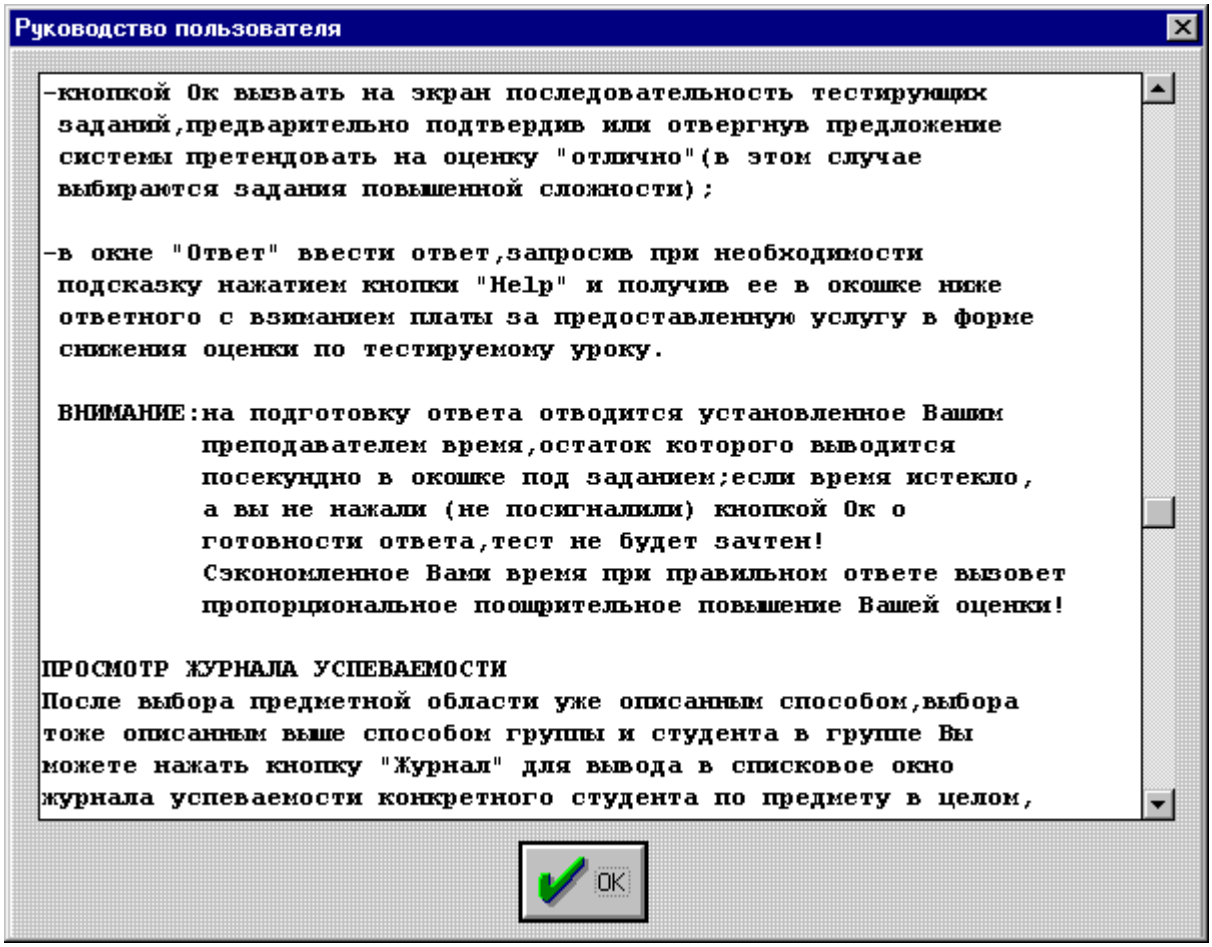
(d) Инструментально - исполнительная система обучения и тестирования (TUTOR), в которой содержалась теория по методу, задание на лабораторную работу, методическая помощь, в особо трудных случаях включающая в себе отдельные фрагменты программной реализации изучаемого метода.

Размер исполняемого модуля системы 220 Кбайт без DLL-файлов, размеры файлов обучающих материалов, тестов и вспомогательных справочников определяются объемом соответствующей предметной области..

Система предоставляет пользователю 2 основных режима работы: «Разработка» и «Эксплуатация» (обучение и тестирование). Режим разработки предназначен для преподавателя и защищен от случайного использования кнопкой пароля, проверяющей наличие в дисководе ключевой дискеты, а при ее отсутствии запрашивающей пароль с клавиатуры.

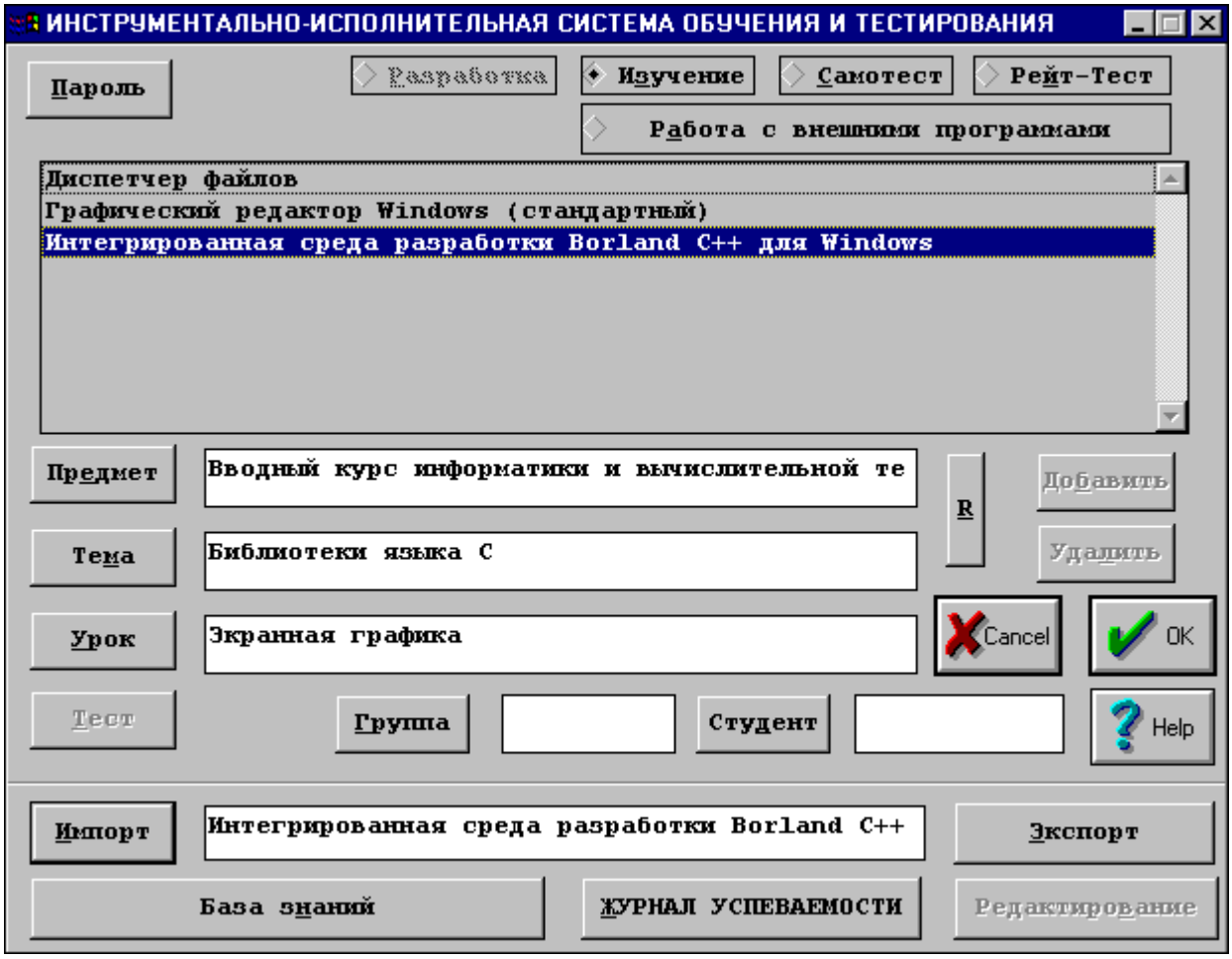
Запуск системы осуществляется из среды ДОС (с попутным автоматическим запуском Windows) или непосредственно из Windows. Windows - система сложная, но только на первых порах, по крайней мере в использовании, поэтому запуск из

Windows-программы из ДОС - возможность, облегчающая ее эксплуатацию. Как показала практика, для успешного начала курса лабораторных работ по численным методам в Windows студентам, абсолютно с ней не знакомым, достаточно одной пары, во время которой они из командной строки ДОС производят запуск Windows в расширенном режиме 386-го процессора, учебника по Windows и системы TUTOR: c:\>win /3 wintutor TUTOR В учебнике студенты осваивают мышь и манипуляции с основными управляющими элементами Windows, затем они переходят в систему TUTOR и, пользуясь подсказкой на ее заставке, по клавише F1 или кнопке Help получают информацию о том, как пользоваться этой системой. После пуска по умолчанию устанавливается режим эксплуатации.



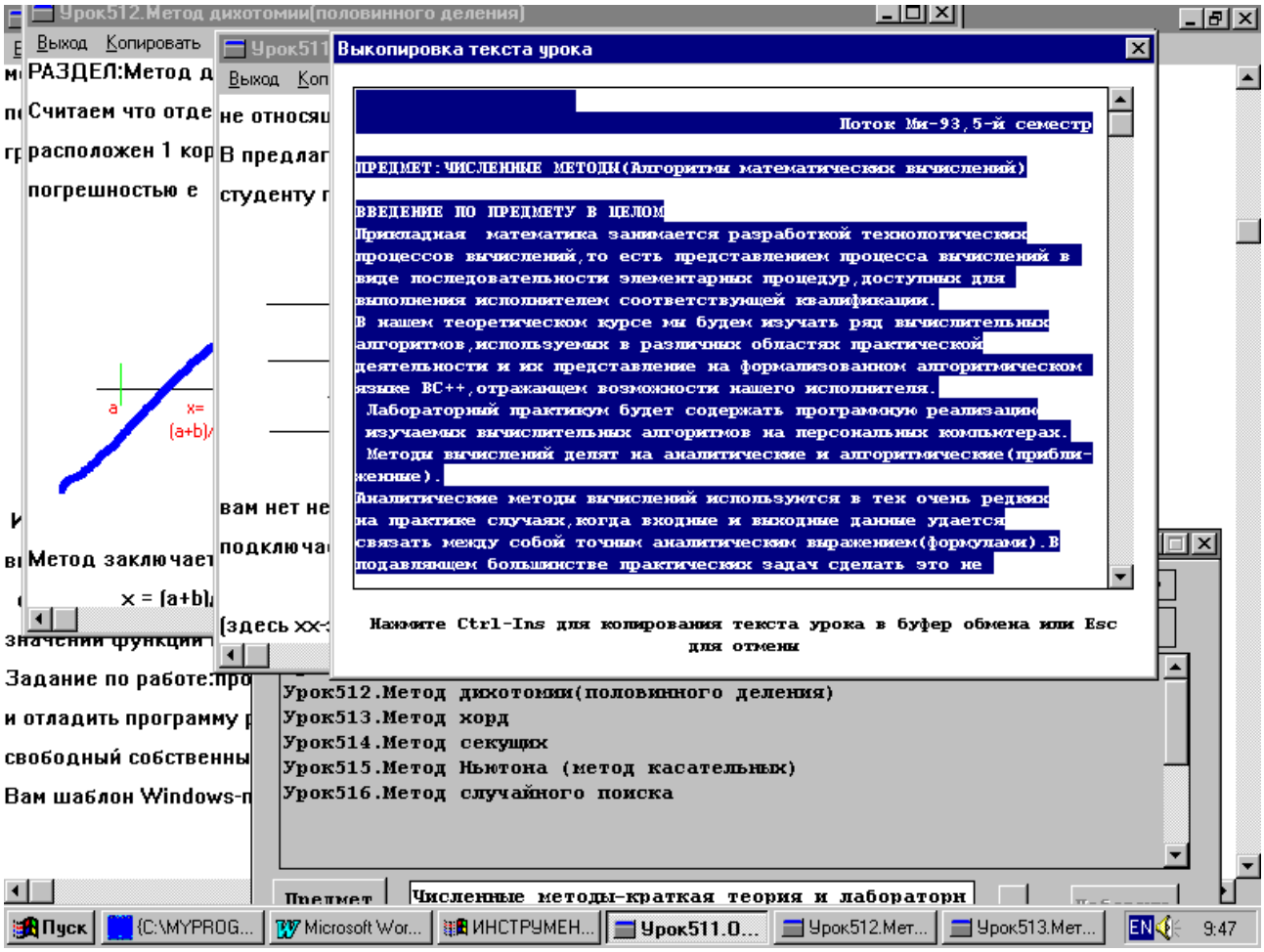
В системе отсутствуют многоуровневые меню выбора режимов и различных установок, столь привычных для сложных систем - с целью упрощения все органы управления сосредоточены в главном диалоговом окне, имеющем вид панели управления с набором кнопок и текстовых субокон. Такого рода инженерный подход - создание панели управления - позволяет постоянно иметь перед глазами режим работы,

выбранную предметную область, группу и имя студента, что, как показал опыт эксплуатации, ускоряет работу, в отличие от меню-ориентированных систем.



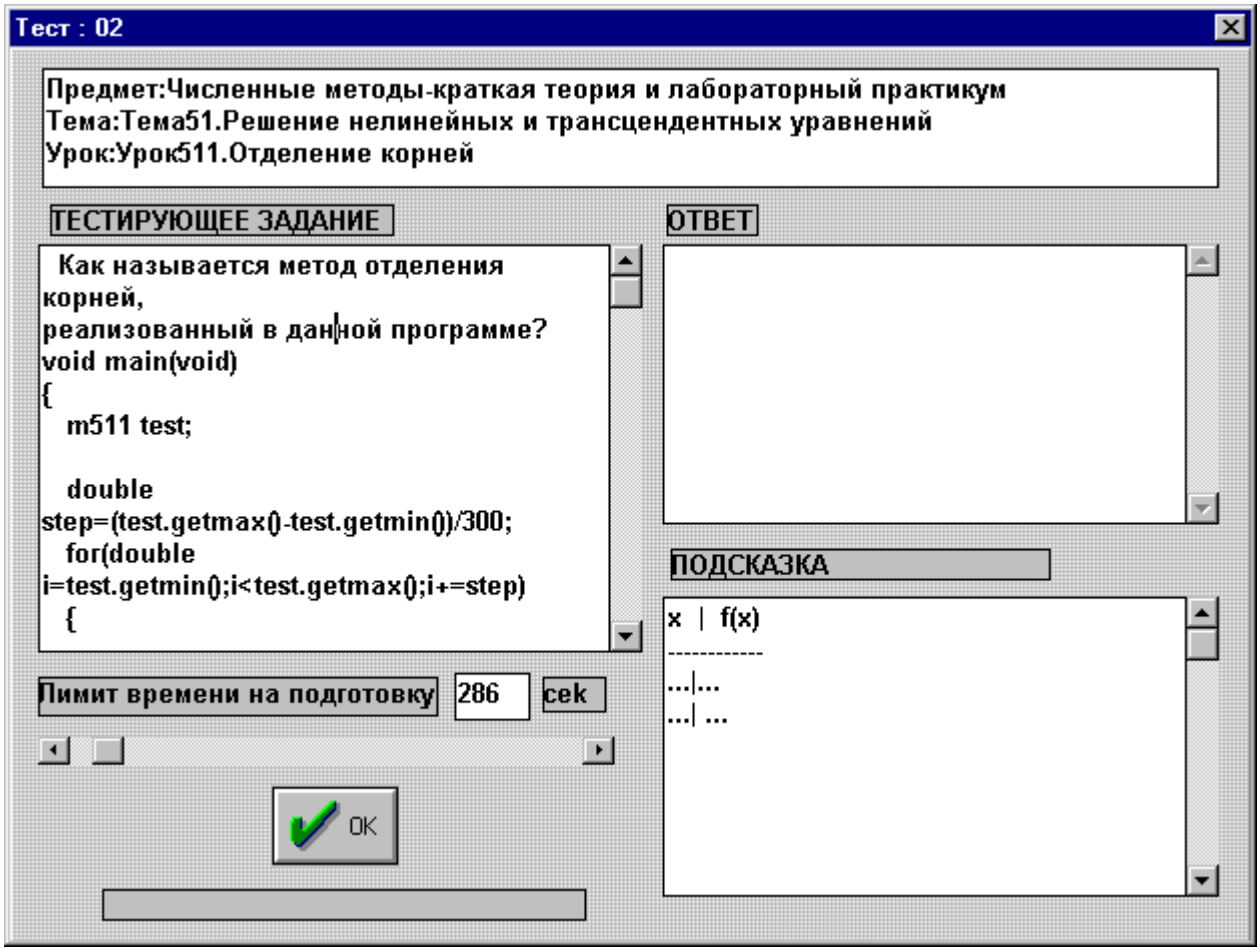
Для выбора вида работ в режиме эксплуатации в правой верхней части панели расположен набор зависимых кнопок, позволяющих выбрать один из четырех видов работы с помощью мыши или клавиши «Tab». В средней части панели размещено большое окно для отображения различных режимно-зависимых списков (с возможностью выбора элемента списка) или журнала успеваемости для просмотра. Режим «Обучение» реализуется после задания предметной области, выбираемой по условной трехуровневой схеме - иерархии типа «Предмет-тема-урок». Нажатие кнопки «Предмет» приводит к выводу в центральное окно списка зарегистрированных предметов (или разделов верхнего уровня). Выбор в этом списке переносит название предмета в связанную с кнопкой текстовую строку и позволяет перейти к следующему уровню «Тема» и далее к уровню «Урок». На любом уровне нажатие кнопки «ОК» вызывает переход в окно демонстрации обучающего материала - по одному или последовательно по всем урокам темы или предмета.

Названия всех уровней предметной области выводятся в верхней части демонстрационного окна. В режиме обучения доступны все стандартные функции просмотра (редактирование обучающего материала в этом режиме недоступно), выбора и переноса блоков через Clipboard - буфер обмена Windows, который является одним из способов обмена данными между различными приложениям. Скажем, любой человек, находясь в режиме изучения урока, может с помощью клавиш Shift-Down & Shift-Left выделить блок текста (он инвертируется по сравнению с невыделенными участками - скажем, стает черным), затем комбинацией клавиш Ctrl-Ins запомнить в буфере обмена выделенный блок, и, наконец, нажатием Shift-Ins в любом текстовом редакторе, работающим под Windows (Блокнот [Notepad], Write, MS Word, IDE BC++, Borland Pascal и многих других), запомненный текст переносится в окно редактирования, откуда его можно распечатать или запомнить в каком-нибудь файле.



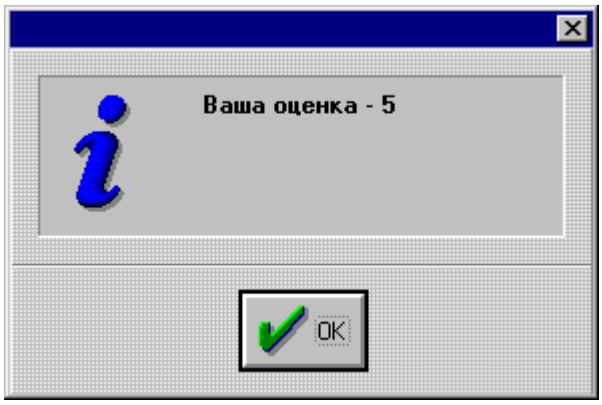
К сожалению, оценка, несмотря на опыт Амонашвили, до сих пор остается мерилом знаний обучаемого, поэтому была принята попытка совместить как тестирование «для себя», так и «для учителя». Режим «Самотестирование» служит в качестве

вспомогательного средства обучения и анонимной проверки знаний без занесения оценки в журнал успеваемости. В этом режиме после выбора предметной области осуществляется переход в диалоговую панель тестирования, содержащую окно с тестовым заданием, окно для набора текста ответа, окно-индикатор остатка времени на подготовку ответа, кнопки запроса подсказки и готовности ответа. В верхней части выводится полное название всех уровней выбранной предметной области. Набор текста ответа осуществляется с клавиатуры или копированием из буфера обмена (туда может быть занесена трудно запоминающаяся часть урока или весь урок в режиме обучения, так что обучающий материал при тестировании, что называется, «под рукой»).



Система оценивания тестового задания мягка и демократична - сэкономленное время на подготовку приводит к повышению оценки с заданным коэффициентом, а просроченное - к незачету теста. Оценка выставляется после прохождения определенного количества тестов по данному уроку (3-х по умолчанию). Нажатие кнопки запроса подсказки выводит окно с текстом помощи к данному тесту, подготовленным

преподавателем (за что, однако, приходиться расплачиваться некоторым снижением оценки), а кнопка готовности ответа вызывает подпрограмму анализа текста ответа.



Алгоритм этой подпрограммы - наиболее трудная и ответственная часть системы. Если бы она делалась традиционно - то есть была бы закрытой и ограничивалась какой-либо одной областью знаний, то проблем бы не возникло. В литературе упоминается, к примеру, система SOPHIE (sophisticated instructional environment). Это имитационная программа, предназначенная для обучения поиску неисправностей в электронной цепи. В рамках данной очень узкой области эта программа позволяет анализировать ответ тестируемого, указывая не только на ошибку, а и на последовательность операций, приведших к ней.

Распознавание и диагноз ошибок обучаемых является одной из важных предпосылок успешного функционирования интеллектуальных обучающих систем, однако, как уже было показано выше, это возможно лишь в рамках узкоспециализированных систем. Сегодня не существует эффективных методов оценки смыслового содержания ответа по абсолютно произвольной тематике и приходится полагаться на формальный анализ текста на совпадение с эталоном. Существует ряд классификаций тестов, но все они, от примитивных выборочных до конструкций из последовательности слов и фраз, набираются с клавиатуры и первичный материал для анализа - всегда текст.

Очевидно, что случайная опечатка в одном символе при наборе не должна приводить к незачету теста. Скажем, возьмем какой-нибудь первоапрельский тест на тему «Самое...»:

Вопрос: Напишите фамилию русской советской актрисы, состоящую из более чем 30 букв.

Эталонный ответ: Архневолоточерепопиндрюковская.

Думаю, самые заядлые балетоманы вряд ли с первого раза смогут написать его правильно (не говоря уже о том, чтобы произнести). Кроме того, в отдельных случаях приходится допускать возможность перестановки слов в фразе, не искажающую смысла ответа. Поэтому в рассматриваемой системе используется многоуровневый

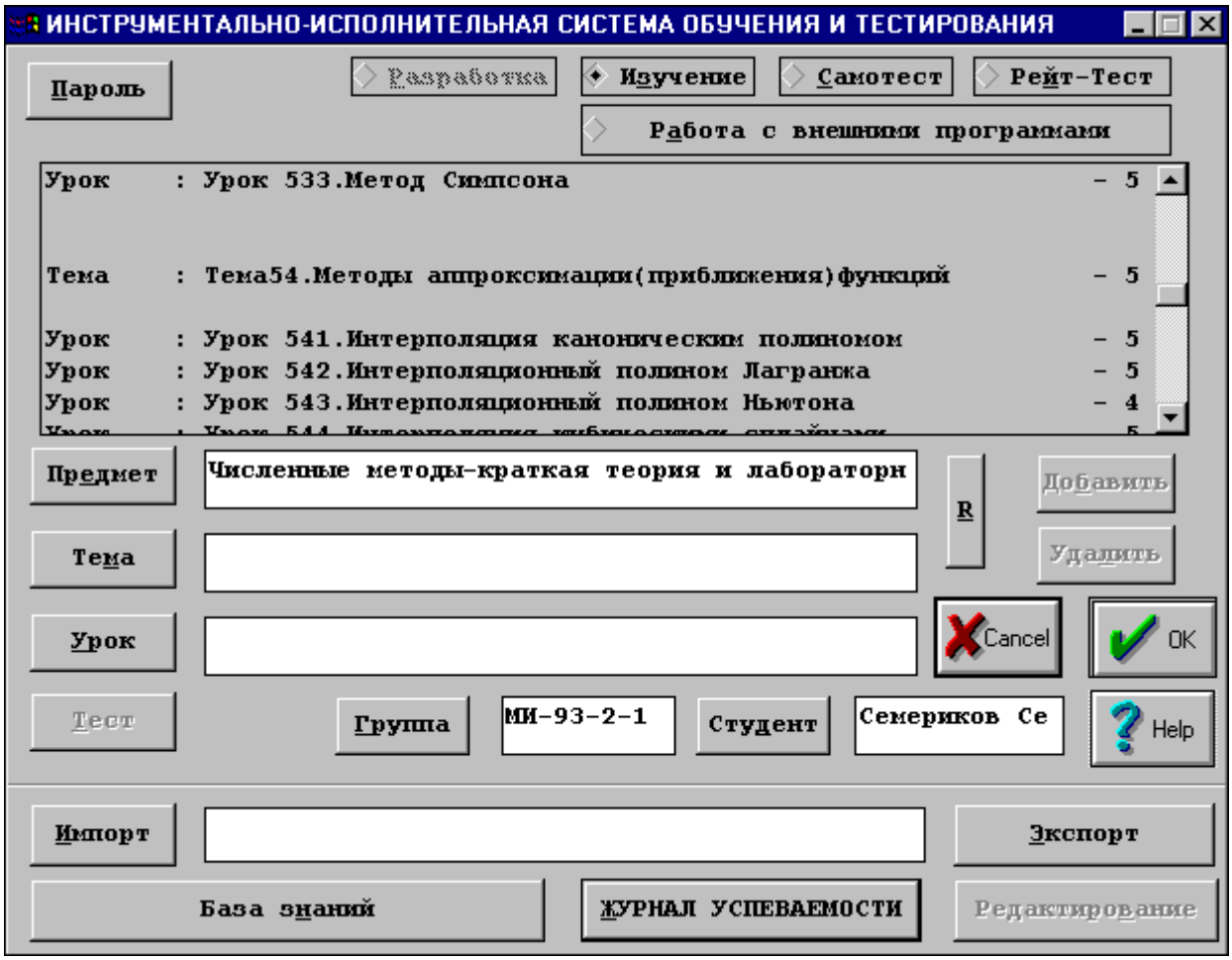
анализ. Если после удаления пробелов, возвратов каретки и переводов строки нет полного совпадения с эталонным ответом, для тестов с взведенным флагом сложности осуществляется его разбивка на лексемы и проверка на наличие всех эталонных лексем в тексте ответа.

Лексемы ответа, не нашедшие аналога в эталоне, подвергаются проверке на посимвольное несовпадение с «незанятыми» лексемами эталона. Если найденная пара с минимальным числом несовпадающих символов содержит не более заданного количества (у нас - одного) несовпадений, то фиксируется совпадение с незначительным снижением оценки по тесту.

Снижение оценки осуществляется и при нарушении порядка следования слов, ведь, скажем, некоторая разница между фразами «не забуду мать родную» и «забуду не родную мать» все же есть. Наконец, если тест не зачтен, в отдельном окне выводится эталонный ответ и обучаемый имеет возможность оценить степень искажения смысла ответа. Но и здесь ему есть возможность подать на апелляцию - если он считает, что автоматический анализатор «не прав», он может привлечь преподавателя, у которого есть приоритетный доступ к оценке теста: после нажатия кнопки «Abort» проверяется наличие в дисководе ключевой дискеты с кодом пароля. Эту возможность приходится оставлять, так как составление теста, «загоняющего» тестируемого в абсолютно однозначный ответ, затруднительно, особенно в гуманитарных предметах и в тестах на многословные определения. Разумеется, в тестах с числовыми ответами, ответами с указанием номера правильного (неправильного) варианта и тому подобными проблем не возникает.

Тесты генерируются в заданном количестве на каждый раздел темы в случайной последовательности из имеющегося набора готовых к эксплуатации (этот набор может пополняться постепенно). После прохождения заданного количества тестов для выбранного раздела курса выводится информационное окно с общей оценкой. Режим рейтинг-тестирования с занесением оценок в журнал реализуется после выбора группы из списка зарегистрированных в системе (список групп выводится по кнопке «Группа») и фамилии ученика в списке (кнопка «Студент»). В остальном этот режим полностью аналогичен самотестированию. Разница между этими режимами несущественная, потому что, по умолчанию, ни разу не тестировавшемуся студенту по всем урокам в журнале стоит оценка «2», и его задача - не набирать хорошие оцен-

ки, чтобы, протестировавшись на отлично по одному уроку, получить высший балл по всему предмету (как это сделано в некоторых оболочках), а исправлять эти двойки. Такой подход может показаться жестоким, однако альтернативы ему нет - только так можно обеспечить прохождение студентов тестов если не по всем, то хотя бы по большей части уроков (тем не менее, авторы подсластили и эту пилюлю - при спорной оценке округление всегда идет в пользу студента). Благодаря такому подходу перед обучаемым нет психологического барьера получения плохой оценки - она у него уже стоит. Для просмотра журнала успеваемости необходимо выбрать предметную область до нужного уровня иерархии, группу и фамилию в группе и нажать кнопку «Журнал». При выборе только верхнего уровня «Предмет» вывод будет содержать названия и оценки всех тем и разделов, а при выборе самого нижнего («Урок») - только одну оценку. Вывод журнала производится не в отдельное окно, а в системный список, где его можно просмотреть.



Основная работа в Windows производится с помощью манипулятора типа «мышь», поэтому и в TUTOR 1.1 она используется очень широко, однако нет таких

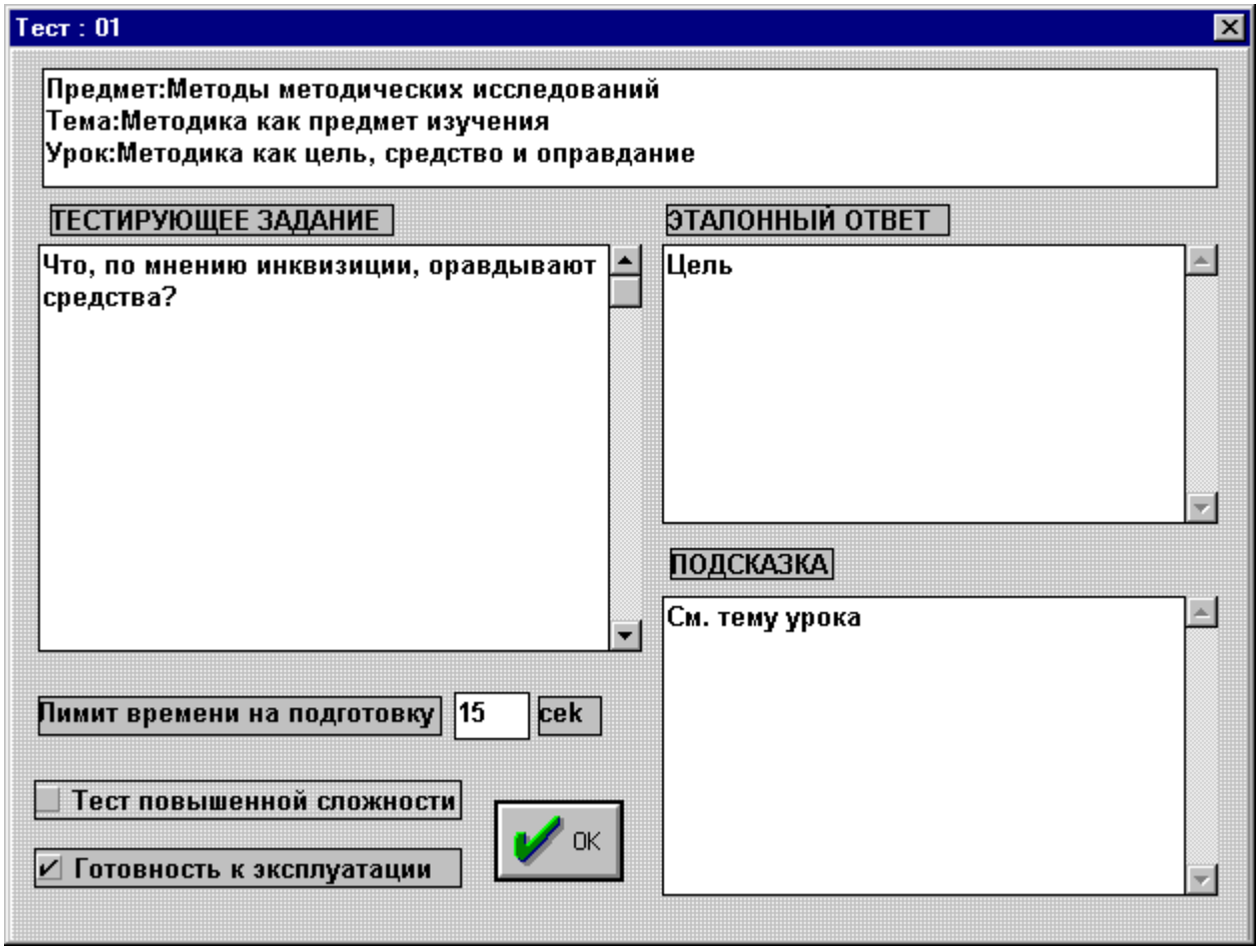
операций с мышью, которые нельзя бы было проделать с помощью клавиатуры. Например, любое окно можно прокрутить как с помощью мыши, так и клавишами управления курсора, правая кнопка мыши дублирует одновременно клавишу F1 и кнопку Help, а двойной щелчок на элементе списка является аналогом нажатия Enter'a на этом элементе - оба эти способа производят выбор.

Режим «Работа с импортированными программами» предназначен для объединения «под одной крышей» всех имеющихся в наличии и необходимых для изучения курса вспомогательных инструментальных средств - это могут быть другие обучающие программы по отдельным разделам, анимационные программы и обучающие игры, трансляторы, используемые при изучении языков программирования и т.д. Список зарегистрированных в оболочке путей доступа к таким программам выводится при нажатии кнопки «Import», выбор в списке - перенос конкретного пути в соответствующее субокно, а нажатие кнопки «ОК» - к выполнению программы. Программа выполняется параллельно с основной системой, и при этом можно воспользоваться всеми преимуществами многозадачности Windows (это относится и к ДОС-программам, при отсутствии к ним соответствующего PIF- файла запускаемых в окне с возможностью работы в фоновом режиме).

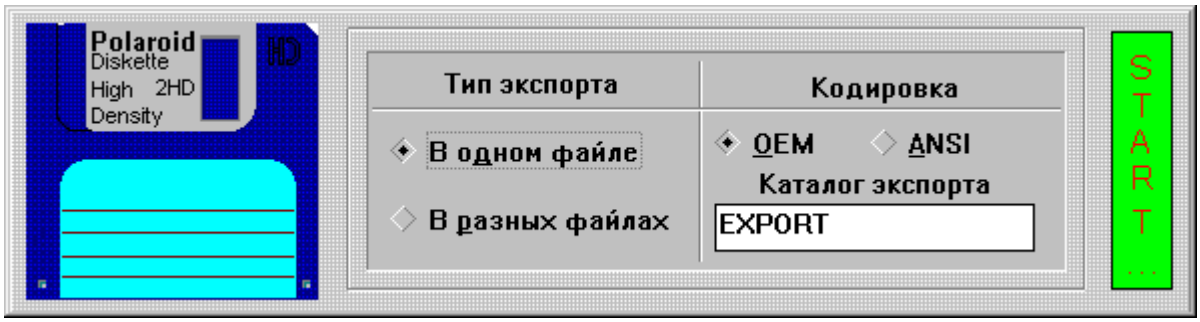
Режим «Разработка» становится доступен после нажатия кнопки «Пароль» при наличии в дисководе ключевой дискеты или при ее отсутствии - после набора пароля. Переход в этот режим убирает с панели управления кнопки выбора режимов эксплуатации и добавляет кнопки «Добавить», «Удалить», «Редактирование», «Тест» и т.д. Выбор предметной области осуществляется так же, как и в режиме эксплуатации. Отличие - в доступности функций корректировки и редактирования списков предметов, тем, уроков, тестов, групп и их составов, импортируемых программ, текстов уроков и тестирующих заданий. В соответствующих окнах редактирования уроков и тестов присутствует кнопка отметки готовности к эксплуатации.

При наборе тестов преподаватель составляет вопрос или задание, эталонный ответ, подсказку-помощь, задает в секундах время на подготовку ответа и признак сложности теста, используемый при анализе ответа и формировании оценки. Для формирования уроков могут использоваться уже готовые текстовые файлы в кодировке ANSI или OEM (после перекодировки утилитой fconvert или через стандартый Windows-редактор Write и буфер обмена). Эта возможность оказывается не только

полезной, но зачастую и незаменимой в тех случаях, когда курс лекций уже набран ранее (как это, к примеру, было с курсом «Основы научных исследований»).

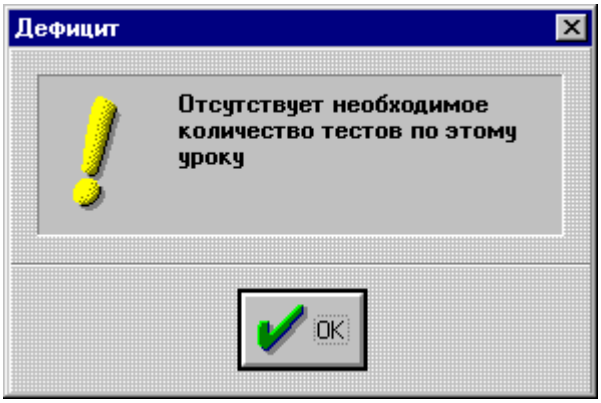


В оболочку включена также возможность вывода уроков и тестов в файл в ANSI или OEM кодировке (только текст) или в формате Windows Metafile (текст + графика). Получение твердой копии из файла оставляется на усмотрение пользователя: текст можно распечатать и из ДОС - любым способом -, а метафайлы (.WMF, .MET) экспортируются текстовым процессором WinWord.



Переход из режима разработки в режим эксплуатации осуществляется по кнопке <R>estart; нажатие этой кнопки в режиме эксплуатации приводит к очистке всех текущих установок: полей выбора, окна списков etc. Эта кнопка позволяет преподава-

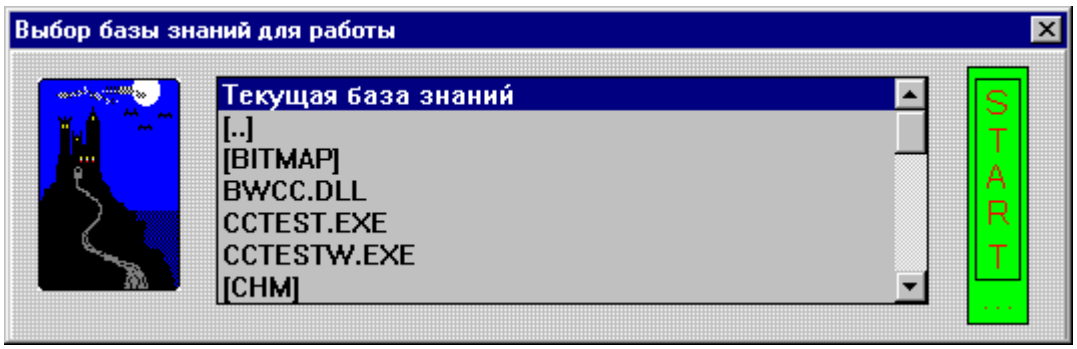
телю тут же опробовать набранные уроки и тесты. Такую возможность следует оставлять, так как представление урока, имеющего графические элементы, в режиме разработки и в режиме изучения очень отличается.



Неправильные действия пользователя (например, нарушение последовательности операций) сопровождаются во всех режимах выводом информационных сообщений о характере ошибки и, если это возможно, способе ее исправления. При удалении из списка элементов предметной области требуется предварительно удалить все элементы подчиненного уровня, удаление уроков и тестов требует дополнительного подтверждения пользователя. Следует заметить, что, в отличие от всех остальных кнопок, удаляется тот элемент, что подсвечен в окне списка, а не тот, что был выбран двойным щелчком.

Система составлена в объектно-ориентированной методологии. Доступ к объектам классов «Урок» и «Тест» на диске - прямой, использующий небольшие по объему ассоциативные справочники с элементами «код - позиция в файле». Справочники загружаются в оперативную память при пуске и непрерывно корректируются в процессе вносимых изменений. Корректировка дисковых файлов осуществляется при закрытии окон разработки уроков и тестов. Для ускорения дисковых операций запись каждый раз производится в конец соответствующего файла, так что в процессе работы на диске может оказаться несколько копий одного и того же урока или теста разных степеней готовности. Чтобы избежать такого расточительного расхода дискового пространства - а иногда это многие сотни килобайт - , в конце сеанса работы с системой пользователю предлагается произвести пересортировку данных с целью «вырезки» данных, на которые нет ссылок в справочниках (удаленных или резервных копий). Эта операция опциональна, так как отнимает при больших объемах данных немало времени (например, 900 килобайт пересортировываются 3 минуты).

Чтобы не смешивать несвязанные между собой темы в одном файле, их, как правило, помещают в различных каталогах - базах знаний, каждая из которых имеет свои журналы, списки и т.д.



Коды элементов предметной области побитно упакованы (4 бита на каждый уровень иерархии) для экономии памяти. Этим объясняется системное ограничение - до 15 элементов на каждый уровень (до 15 предметов, до 15 тем на каждый предмет и т.д.). Это ограничение может быть снято или расширено до 255 элементов на каждый уровень, но необходимость в этом не просматривается. Это же ограничение действует на количество групп и обучаемых в группе. Создание каждого нового предмета сопровождается созданием 4-х файлов - уроков, тестов и соответствующих им двух ассоциативных справочников, однако решение поступать именно так является волевым и мало обоснованным; возможно, в дальнейшем оно будет пересмотрено. Побитовая упаковка используется также для оценок в журнале успеваемости - 2 бита на оценку, т.е. используется четырехбальная система. Такое кодирование продиктовано огромным количеством возможных оценок даже для одного предмета - до 50625 при упомянутых ограничениях. Количество импортируемых программ в системном списке не ограничивается. Четырехбальная же система в обосновании, по видимому, не нуждается - мало кто из преподавателей делает разницу между «двойкой» и «единицей».

В настоящее время обсуждаемая оболочка наполнена уроками и тестами по предметам: «Численные методы», «Элементарный (вводный) курс информатики», «Основы научных исследований», «Основы событийного программирования» и некоторым другим. Обкатка системы проводилась в четырех группах студентов 3-го курса (поток МИ-93) и, помимо устранения ряда ошибок и недоработок, показала следующие результаты.

Освоение органов управления достигается за 10-20 минут при условии хотя бы минимального опыта работы с Windows; если же таковой отсутствует, занятие начинается с учебника по Windows (по изложенному выше образцу). При серьезном отношении преподавателя к оценкам в системном журнале успеваемости последний играет ключевую роль в активном использовании студентами обучающих материалов и

посещений дополнительных занятий в компьютерном классе. Это особенно эффективно для студентов, имеющих проблемы с освоением курса - наблюдаются попытки многократного тестирования, конспектирования эталонных ответов при незачетах тестов с целью будущего использования, активный поиск ответов в текстах уроков - студент невольно втягивается в процесс «самонатаскивания» по трудно поддающимся освоению разделам (в свободное от занятий время).

Одной из проблем в начальной версии системы было рассогласование журналов на различных машинах, когда студент, протестировавшись на одной машине, не имеет результатов тестирования в журналах других компьютеров. При работе в сети Windows for Workgroups эта проблема решается установкой обсуждаемой системы на одной машине, когда все студенты работают как клиенты. Однако сеть у нас встречается нечасто, поэтому была создана специальная программа сравнения журналов: раз в 2 недели преподаватель «обходит» все компьютеры с ключевой дискетой, на которой собирается информация об изменениях в оценках - так изменение на одной машине переносится на все другие.

Другой эффект - повышение производительности, например, при обучении программированию, - возникает за счет копирования фрагментов программ из приводимых в уроках методических пособий в интегрированную среду разработку. Это дает возможность увеличить сложность заданий по программированию за счет «сборки» программ из рекомендуемых фрагментов.

2. Использование СУБД для решения задачи расписания на примере расписания занятий

2.1. Общие требования к системам управления базами данных

База данных как понятие характеризуется двумя основными аспектами - информационными и манипуляционными. Первый аспект отражает такую структуризацию данных, которая является наиболее подходящей для обеспечения информационных потребностей, возникающих в предметной области (ПО). С каждой ПО ассоциируется совокупность «информационных объектов», связей между ними, а также задач их обработки. Манипуляционный аспект БД касается смысла тех действий над структурами данных, с помощью которых осуществляется выборка из них различных компонентов, добавление новых, удаление и обновление устаревших компонентов структур данных, а также их преобразования.

Под системой управления БД (СУБД) понимается комплекс средств (языковых, программных и, возможно, аппаратных), поддерживающий определенный тип БД. Главное назначение СУБД, с точки зрения пользователей, состоит в обеспечении их инструментарием, позволяющим оперировать данными в абстрактных терминах (именах и/или характеристиках информационных объектов), не связанных со способами хранения данных в памяти ЭВМ. Следует заметить, что средств СУБД может, вообще говоря, не хватать для решения всех задач той или иной ПО. Поэтому на практике приходится адаптировать (дополнять, настраивать) средства СУБД для обеспечения требуемых возможностей. Системы, получаемые путем адаптации СУБД к данной ПО, относятся к автоматизированным информационным системам (АИС). Жизнеспособная АИС, т.е. способная поддерживать модель БД с учетом динамики развития ПО, по необходимости должна в качестве своего ядра содержать СУБД. Вы-

работанная на сегодняшний день методология проектирования АИС включает четыре основные задачи:

1) системный анализ предметной области (ПО), спецификацию информационных объектов и связей между ними (в результате вырабатывается так называемая концептуальная, или семантическая, модель ПО);
2) построение модели БД, обеспечивающей адекватное представление концептуальной модели ПО;
3) разработку системы управления базами данных (СУБД), поддерживающей выбранную модель БД;
4) функциональное расширение (посредством некоторой системы программирования) СУБД с целью обеспечения возможностей решения требуемого класса задач, то есть задач обработки данных, характерных для данной ПО.

На практике в каждом рассматриваемом случае пути решения этих задач выбираются исходя из специфики ПО, функциональных возможностей доступных СУБД и вычислительных систем, допустимых затрат на создание АИС и др.

Основная масса работ по теоретическим и теоретико-прикладным исследованиям в области АИС за весь период ее развития была связана с моделированием БД, разработкой архитектур СУБД и их языковых средств. Среди вопросов программной реализации СУБД основное внимание уделялось разработке способов эффективного размещения данных в памяти ЭВМ. Рассмотрим основные направления исследований в области БД.

На раннем этапе (до 1975 г.) развития этой области наиболее характерными являются исследования способов структуризации данных, не зависящих от специфики организации среды хранения (памяти ЭВМ). Разработанные способы структуризации данных получили название моделей данных. Каждая такая модель характеризуется определенными средствами и методами структурирования данных. Наиболее известны так называемые иерархическая, сетевая и реляционная модели данных.

Реляционная модель (разработанная Э.Ф. Коддом в 1970 г.) выгодно отличается от всех других моделей данных за счет простоты, математической строгость и практической полезности. Основной прагматической направленностью реляционного подхода было построение средств структурирования данных и манипулирование ими, которые позволяли бы решать задачи проектирования СУБД, языков манипулирова-

ния данными, отвлекаясь при этом от всего того, что связано с организацией среды хранения данных, адресации и доступа к ним. Структуры данных в этой модели уточняются через конечные отношения, заданные на доменах (множествах) первичных данных. Важным моментом реляционного подхода явилась также реляционная алгебра, рассматриваемая как аппарат построения одних структур данных из других, уже ранее построенных или первоначально заданных. Все это послужило толчком к обработке новых и интересных с практической точки зрения языков манипулирования данными и СУБД. На втором этапе (1975-1980 г.) развития теории БД акценты заметно смещаются в сторону анализа прагматико-смысловых интерпретаций данных.

Практика показала, что «хорошая» АИС должна обеспечивать возможности манипулирования и обработки данных в соответствии с пониманием (пользователем) их смысла. Известные на это время модели данных оказались неадекватными с точки зрения информационного представления в БД знаний о фрагменте действительности - предметной области АИС. Все это привело к формированию нового направления исследований, ориентированного на вскрытие структур и закономерностей ПО. Для этого периода характерно построение моделей данных путем расширения иерархической, сетевой и реляционной моделей за счет включения элементов «смысловой интерпретации» информационных объектов ПО, связей и зависимостей между ними. Однако наиболее удачной в смысле математической строгости и прагматической полноты оказалась расширенная реляционная модель, предложенная всё тем же Э.Ф. Коддом в 1979 г. Эта модель, в отличие от других, интересна еще и тем, что при ее разработке была удовлетворительно решена задача уточнения как информационного, так и манипуляционного аспектов БД. Важность манипуляционного аспекта автор этой модели БД подчеркивает словами: «...структуры данных без способов и правил манипулирования ими похожи на анатомию без физиологии». В этот период были получены интересные результаты, связанные с различными типами зависимостей между компонентами структур данных в реляционных БД. Такие зависимости, называемые ограничениями целостности БД, играют роль информационных инвариантов ПО и в связи с этим должны поддерживаться в БД на протяжении всего периода ее использования. Однако попытки реализации моделей БД с достаточно богатыми типами зависимостей натолкнулись на трудности, связанные с недостаточно глубоко исследованными механизмами их поддержания.

В первой половине 80-х годов внимание исследователей сконцентрировалось на построении самостоятельных, автономных моделей ПО и выработке методов отображения их в модели БД. Что же касается структур данных и средств манипулирования ими, то они данном случае полностью или частично выпали из поля зрения исследователей. Ввиду большой сложности проблемы моделирования ПО предложенные языковые средства моделирования представляют собой, как правило, набор синтаксических структур без строгой семантической их интерпретации. Поэтому, как и следовало ожидать, исследования по моделированию ПО не привели к появлению глубоких результатов в теории БД, хотя и обеспечили богатый фактологический материал, касающийся этой проблематики.

В целом проблематика БД и АИС развивалась по экстенсиональному пути. Это означает, что в основное внимание уделялось широте охвата вопросов, не придавая особого значения расстановке акцентов на них, не отделяя главные, носящие принципиальный характер, вопросы от второстепенных. Большое количество публикаций в этой области, разобщенность взглядов авторов, интуитивная трактовка понятий породили своего рода «терминологический Вавилон». В настоящее время исследования в области БД заметно смещаются в сторону алгебрологического моделирования БД, направленного на адекватное (полное) уточнение на разных уровнях абстракции информационных структур и средств манипулирования их компонентами, а также вскрытие сущности различных типов знаний (предметно-понятийных, функциональных, логических) о ПО и способов вывода производных знаний на основе заданных базовых знаний. Эти исследования позволят теснее связать проблематику БД, языков манипулирования данными, с одной стороны, и проблематику современных языков программирования и поддерживающих их систем - с другой. Одним из основных требований, предъявляемых к современным АИС, является высокий уровень их «интеллектуальности» во взаимодействии с пользователем.

Однако успехи указанных исследований, глубина получаемых результатов, полезность разрабатываемых проектов АИС во многом зависят от того, насколько удачными и точными будут разработаны сами основания теории АИС. Это прежде всего система таких понятий, на которых возможно развитие самой теории БД как вглубь, так и вширь. Выработка оснований теории даст возможность по-новому переосмыслить накопленные теоретические результаты, богатый фактологический материал,

опыт построения и использования реальных СУБД, АИС и на основе всего этого глубже взглянуть на природу баз данных, баз знаний, построить высокопродуктивные информационные технологии.

Рассмотрим конкретнее требования к СУБД высокого уровня, которые предложил Э.Ф. Кодд. Подавляющее большинство систем управления базами данных (СУБД) для персональных компьютеров принято считать реляционными, хотя на самом деле далеко не все из них являются таковыми. Помимо представления данных в виде двумерных таблиц, принадлежность к разряду реляционных систем определяется целым набором признаков, сформулированных Коддом и получивших наименование правил Кодда. Правила Кодда. Кодд предложил двенадцать правил, которые определяют требования к реляционным системам.

1. Явное представление данных.

Вся информация в базе данных должна быть представлена на логическом уровне в виде значений в отношениях.

Это правило определяет собственно реляционную модель данных. Если данные явно не представлены в базе данных, то необходимо интерпретировать их определенным образом в рамках прикладной программы. Например, значения фамилии сотрудника может интерпретироваться для определения пола сотрудника. При этом пол сотрудника не будет явно представлен в базе данных. Алгоритмы интерпретации могут быть неэквивалентны в различных приложениях. Совместная эксплуатация таких приложений может привести к разрушению информации в базе данных или к искажению результата. Для явного представления данных реляционная система должна поддерживать различные типы значений атрибутов: числа, строки символов, даты, время, логические значения и т.п.

2. Гарантированный доступ к данным.

Вся информация в базе данных должна быть доступна для приложения. Выделение любого значения в реляционной базе данных выполняется при указании имени отношения, значение первичного ключа для идентификации кортежа в выбранном отношении и имени атрибута для идентификации значения в выбранном кортеже. Атрибут или имя атрибута - свойство некоторой сущности. Например, «Фамилия», «Возраст» свойства сущности «сотрудники».

Отношение - конечное множество кортежей, составленных из допустимых значений атрибутов схемы отношения. Отношение ассоциируется с таблицей, имена атрибутов - с именами столбцов таблицы, кортежи - со строками таблицы. Схема отношений - конечное множество имен атрибутов, определяющих сущности. Таким образом, задается единый способ доступа к данным: <имя отношения>, <значение ключа>, <атрибут>. Для обеспечения такого способа доступа каждое отношение должно иметь имя и первичный ключ. Каждый кортеж отношения должен отличаться от других кортежей значением первичного ключа. Каждое элементарное значение в кортеже должно соответствовать определенному атрибуту. Альтернативой прямому доступу по ключу является доступ по абсолютному номеру кортежа. В этом случае доступ к данным будет гарантированным только при статическом расположении кортежей в отношении.

3. Полная обработка неопределенных значений.

Неопределенные значения, отличные от любого определенного значения, должны поддерживаться для всех типов данных при выполнении всех операций.

Реляционные операции - операции над отношениями. Результатом любой реляционной операции является также отношение. Основными реляционными операциями являются селекция, проекция и соединение. Операцией селекция - выбор кортежей, удовлетворяющих условию. Операция проекции - проекция отношения на заданную схему. Операция соединения - комбинирование кортежей двух исходных отношений в одно общее.

Третье правило в первую очередь относится к неопределенным значениям атрибутов. Если понятие неопределенного значения не поддерживается, то пользователь вынужден либо отложить операцию, либо ввести фиктивное значение, о котором впоследствии может забыть. Фиктивное значение в общем случае невозможно идентифицировать. Во избежание наполнения базы данных фиктивной информацией используются неопределенные значения, которые независимо от их типа можно легко идентифицировать в базе данных.

4. Доступ к описанию базы данных в терминах реляционной модели.

Описание базы данных (перечень отношений, определение схем отношений и ключей, статистическая информация и т.п.) должно рассматриваться на реляционном уровне. Пользователь должен иметь доступ к этой информации с помощью реляцион-

ного языка, как к обычным данным. Фактически четвертое правило обуславливает возможность администрирования баз данных независимо от приложений. Администратор базы данных должен обеспечивать надежность эксплуатации базы данных всеми приложениями, иметь возможность анализировать статистику использования данных для оценки качества схемы базы данных, оказывать консультации при разработке различных приложений различными программистами.

5. Полнота подмножества языка.

Реляционная система может поддерживать несколько языков, по крайней мере, язык определения данных и язык манипулирования данными. Однако хотя бы один из языков должен иметь синтаксис предложений, в полной мере поддерживающий следующие понятия:

- определение данных (отношения, атрибуты, домены, ключи, ограничения целостности).

Целостность базы данных - свойство базы данных, при наличии которого база данных содержит полную и непротиворечивую информацию, необходимую и достаточную для корректного функционирования приложений. Ограничение целостности - набор условий, определяющих состояние целостности базы данных. Различают ограничения диапазонов возможных значений атрибутов и структурные ограничения (ограничения на кортежи, имеющиеся в отношениях). Частным случаем ограничения диапазонов является фиксация доменов. Домен атрибута - множество допустим значений, которые может принимать атрибут. Частным случаем структурных ограничений является определение первичных ключей.

- определение представлений;
- манипулирование данными (интерактивное или программное);
- ограничения целостности;
- санкционированный доступ к данным;
- управление транзакциями (начало транзакции, фиксация выполнения, отказ от выполнения транзакции).

Транзакция - группа операций над базой данных, сохраняющая целостность базы данных. В интервале времени выполнения транзакции база данных может находиться в нецелостном состоянии. Язык определения данных должен обеспечить вы-

полнение первого и второго правил в самых сложных ситуациях. Например, если некоторое значение вычисляется на основе значений нескольких атрибутов, то оно может быть явно представлено в одном из виртуальных представлений. Виртуальное представление формируется на основе некоторого предложения реляционного языка. Виртуальное представление может использоваться для доступа как обычное отношение базы данных. Корректность информации, доступной через виртуальное представление, обуславливается шестым правилом.

6. Обновляемость представлений.

Все представления, являющиеся теоретически обновляемыми, должны автоматически обновляться при модификации данных в базовых отношениях.

7. Наличие высокоуровневых языков или языка манипулирования данными.

Операции вставки, обновления и удаления должны применяться к отношению в целом. В данном случае предполагается единственно разумный подход к обеспечению контроля целостности базы данных при выполнении различных операций модификации данных. Если вставка применяется к отношению в целом, то при ее выполнении очень просто проверить сохранение уникальности первичного ключа, корректность вводимых значений и т.п. Контроль целостности выполняется на основе проверок ограничений целостности, заданных с использованием языка определения данных.

8. Физическая независимость данных.

Прикладные программы не должны зависеть от используемых способов хранения данных на носителях информации и методов доступа к ним. Физическая независимость данных обеспечивает работоспособность приложений, например, при изменении места расположения отношения среди узлов локальной сети компьютеров. При обеспеченной физической независимости данных нет необходимости изменять текст приложений при изменении формата файлов данных и индексных структур.

9. Логическая независимость данных.

Прикладные программы не должны зависеть от реализации любого из используемых представлений. Определив виртуальное отношение в рамках базы данных, можно разрабатывать приложения, использующие это отношение, не беспокоясь о том, что структура базы данных изменится и виртуальное отношение будет строится на основе другого реляционного выражения.

10. Независимость контроля целостности.

Все ограничения целостности и внешние представления должны определяться не в приложениях, а должны быть определены с помощью языка определения данных и сохранены в каталоге базы данных. Доступ к каталогу базы данных выполняется с помощью реляционного языка. Язык определения данных - язык, позволяющий создать базу данных с требуемой структурой и описать структуру базы. Все приложения, использующие базу данных, должны контролировать эквивалентный набор ограничений целостности. Обеспечить это можно, выделив определение ограничений из прикладной программы в каталог базы данных. Поскольку все приложения имеют доступ к каталогу базы данных, то они будут эквивалентно работать с общими данными.

11. Дистрибутивная независимость.

Реляционная система должна быть распространяема и переносима. Реляционный язык должен позволять манипулировать данными, размещенными в различных компьютерных системах. Требования по «коммуникабельности» имеет особое значение. Дистрибутивная независимость предполагает полную реализацию СУБД для различных платформ или реализацию коммуникационных блоков в составе СУБД, позволяющих обмениваться данными между различными СУБД.

12. Согласование языковых уровней.

Если реляционная система имеет низкоуровневый (элемент доступа - запись) и высокоуровневый (элемент доступа - таблица) языки, то выполнение низкоуровневых команд должно согласовываться с контролем целостности данных аналогично высокоуровневым командам. Реляционные системы могут поддерживать несколько языков. При этом необходимо обеспечить эквивалентность обработки данных в этих языках. Особое внимание обычно уделяется эквивалентности записе-ориентированных и таблице-ориентированных реляционных языков.

В 1993 году Кодд в своей новой работе изложил уже 300 (!) правил для построения реляционных баз данных, но даже предложенные 12 в повседневной работе являются избыточными. Для решения поставленной задачи в заголовке раздела мы будем использовать настольную СУБД FoxPro, в которой из 12 правил правила 2, 3, 4, 5, 6, 7, 8, 9, 10 не выполняются или выполняются не полностью, однако для нас хватает и трёх оставшихся.

2.2. Использование СУБД FoxPro 2.0 для построения и обслуживания БД «Расписание занятий»

Программа составления расписания занятий - не лучшая иллюстрация к системам управления базами данных, обычно их иллюстрируют задачами складского учета, учета автомобилей в автоинспекции, больных в стоматологии или служащих на предприятии. Наш выбор обусловлен, с одной стороны, приобретаемой специальностью школьного учителя, для которого маловероятно занятие учетными задачами приведенных классов, а с другой стороны тем, что составление расписания относится к «неудобным» задачам (например, в части формирования экранного отображения базы данных) - дело в том, что большинство практических задач отличаются от стандартных иллюстраций именно «неудобностью» и мы хотели использовать неклассический вариант для освоения некоторых методов преодоления возникающих трудностей.

Будем считать, что перед нами стоит задача создать компьютерную программу, позволяющую составлять, изменять и оптимизировать по некоторым критериям расписание занятий на один семестр в высшем учебном заведении, включающем некоторое множество факультетов, каждый из которых ведет обучение по нескольким специальностям в течение ряда семестров. Каждая специальность на отдельно взятом курсе обучения может содержать несколько групп обучаемых, а каждая группа может разбиваться на подгруппы для проведения, например, лабораторных занятий в специализированных лабораториях.

Составить расписание для конкретного факультета - значит определить для каждой подгруппы на каждую 2-часовку (или более - например 4-часовое занятие) каждого дня недели изучаемый предмет, проводящего занятие преподавателя и соответствующую аудиторию. При этом и преподаватель, и аудитория в соответствующей 2-часовке не могут фигурировать в расписании занятий других факультетов и любых подразделений обучаемых - не должно быть так называемых «накладок». Оптимизация расписания состоит в исключении «дыр» в занятиях студентов и в загрузке преподавателей, в размещении предметов по 2-часовкам каждого дня занятий в соответ-

ствии с категорией сложности предметов - более трудные должны идти раньше легких, в расстановке занятий в разных корпусах через большую перемену для обеспечения времени на перемещение студентов и преподавателей и пр.

Программа должна предоставлять серию необходимых услуг по работе с готовым расписанием - например, по запросам пользователя выдавать на экран и принтер расписание занятий любой заданной группы учащихся на четную или нечетную неделю, такое же расписание для любого заданного в запросе преподавателя или всех преподавателей кафедры, недельную аудиторную нагрузку преподавателя или группы студентов и пр.

Совершенно очевидно, что нам предстоит обрабатывать некоторый файл записей, каждая из которых содержит «ведущие» или ключевые поля, одни из которых определяют «столбец» в табличной форме расписания:

- ❖ курс;
- ❖ специальность;
- ❖ группа;
- ❖ подгруппа;

другие поля определяют строку в таблице расписания занятий:

- день недели;
- номер 2-часовки (пары, ленты и пр.),

а некоторые поля являются как бы зависимыми, ведомыми:

- ❑ предмет;
- ❑ преподаватель;
- ❑ аудитория.

Еще одна группа полей может носить общий характер:

- ➢ факультет;
- ➢ тип недели (четная - нечетная, числитель- знаменатель и пр.).

Таким образом, мы будем работать с файлом табличной структуры и должны обеспечить возможность выполнения в этом файле ряда стереотипных действий - создание файлов и заполнение записей, их корректировку при необходимости, поиск записей с заданным содержимым некоторых полей, обработку заданных последовательностей записей по некоторому алгоритму и пр.

Очевидно, что мы могли бы обойтись в нашем случае одним файлом с перечисленными выше полями, но в этом случае имела бы место нерациональная избыточность по расходу памяти - например названия факультетов, специальностей, фамилии преподавателей и многое другое многократно повторялись бы в различных записях в неэкономичной текстовой форме. Поэтому такие повторяющиеся многократно сведения обычно размещают в отдельных файлах списков совместно с их числовыми кодами, а в основном файле базы данных оперируют только этими более экономичными в части расходования памяти кодами. Таким образом, база данных представляет собой обычно совокупность многих взаимосвязанных файлов.

Опишем вкратце некоторые алгоритмы и структуры данных, использованные в нашей СУБД.

Как показывает практика, в СУБД самое важное - это продуманные структуры данных, поэтому проектирование системы для обслуживания расписания мы начнем с составления структур баз данных. Следующим этапом будет заполнение этих баз, и затем - самая нужная часть, ради которой делается львиная доля работы: обслуживание базы данных, под которым мы будем понимать удовлетворение различных запросов к ней.

Вспомним, как выглядит обычное расписание. На большом листе чертится или отпечатывается большое количество клеточек. Строки в расписании соответствуют номеру пары в какой-то из дней недели, столбцы определяют соответственно потоки, группы и подгруппы. А в клеточке мы видим название предмета, имя преподавателя и номер аудитории. Таким образом, для того, чтобы однозначно определить эту клеточку на стене в базе данных, нам необходимо пять полей: группа, пара, предмет, аудитория и преподаватель.

Естественно, заносить эти данные в базу так как это делает диспетчер, по меньшей мере нерационально - может быть, ей и приятно раз 15 написать чьи-то реквизиты, но не лучше ли обозначить их каким-нибудь кодом и заносить в расписание не строку «Великий инквизитор», а какое-то однозначно определяемое число, и так во всем. Конечно, где-то придется держать список пар типа код - преподаватель (а может и не пар) - в другой базе, к примеру.

Итак, в базе расписания мы введем 5 числовых полей, в которые будем заносить коды групп, пар, предметов, аудиторий и преподавателей.

Код группы мы будем формировать следующим образом:

код_группы=код_потока*100+номер_группы*10+номер_подгруппы. При этом предполагается, что групп на потоке не может быть больше 9 (в противном случае код теряет смысл или, что еще хуже, приобретает неверный), а код потока формируется в отдельной базе потоков.

Чем удобен именно такой код? Легкостью в поиске - если номер группы в коде равен нулю, то эта пара соответствует лекции, если номер группы ненулевой, а номер подгруппы нулевой, то это практичка, иначе - лабораторка. Следовательно, общий код мы никогда не будем делать нулевым, и 0 в коде рассматривать как ошибку.

Обозначив числитель через 1, знаменатель - через 2 и пронумеровав дни недели от 1 до 5, мы получим удобный код пары:

код=номер_недели*100+номер_дня*10+номер_пары

Коды предметов, аудиторий и преподавателей мы будем формировать в соответствующих базах.

База данных по предмету будет, естественно, содержать код и название предмета; кроме того, для удобства возложим на нашу программу обязанность старост - подсчитывать запланированное и реальное число часов, в неделю и общее. Дополнительно введем признак связи предмета с аудиторией, необходимый для того, чтобы, скажем, лабораторные работы по химии проводились в хим. лаборатории, а не в спортзале.

Аудиторию мы будем характеризовать номером корпуса, номером аудитории, наличием какой-либо специализации (например, компьютерный класс) и емкостью - поток, группа, подгруппа. Некоторые аудитории настолько специфичны, что этих данных для кодирования недостаточно; не каждый догадается, что аудитория номер 1 в шестом корпусе - это мастерская ИПФ, поэтому введем комментаторное поле переменной длины, в котором и прокомментируем эти особенности.

Поток традиционно характеризуется специальностью - кодом, номером курса и количеством групп. У потоков филфака есть интересная особенность: часть групп занимается с 13.00, часть - с 7.30 и т.д., причем те, кто занимаются с часу дня, считают свою пару первой, хотя для нас она - третья. Для таких вот экзотических специальностей введем особое поле смещения, показывающее, на сколько пар они отстали от жизни.

Каждый преподаватель имеет имя, звание, приписан к какой-то кафедре; кроме этого, в институте он занимает какую-то должность, что тоже надо учитывать при составлении расписания.

Факультет, специальность и кафедру можно охарактеризовать кодом и именем.

В различных рабочих областях откроем необходимые нам при работе базы данных вместе со своими структурными индексными файлами, индицируя каждое действие соответствующим сообщением на экране. Если файл базы данных не существует, то мы его создаем средствами SQL, попутно индексируя необходимые нам поля.

Рассмотрим подробнее некоторые соглашения об именах, применяемых в нашей базе.

Если в начале массива стоит слово list_, это означает, что данный массив будет использоваться для хранения некоторого списка. Если переменная начинается с cur - то в ней будет содержаться некоторый выбор (обычно код) или 0 в случае его отсутствия. Переменная, заканчивающаяся на txt, содержит строку, соответствующую этому коду.

Конкретный вид данных определяется фразами

aud - аудитория

prep - преподаватель

kuf - кафедра

fac - факультет

spec - специальность

pot - поток

pred - предмет

grp - группа

pg - подгруппа

Кроме того, есть переменные и массивы, не укладывающиеся в эту схему:

WasChanged(10) - каждый элемент этого массива отражает состояние базы данных в рабочей области с соответствующим номером. Если в базе данных в области <number> что-то изменяется, в элемент WasChanged(<number>)=.T. . В конце программы эта информация используется для определения того, какую базу данных необходимо упаковать.

tempos(10) - в этом массиве в нем хранятся числительные от 1 до 9 - номера групп в символьном формате. Они понадобятся для «красивого вывода» номера группы.

days_arr(5) - содержит названия учебных дней недели, а

week_arr(2) - названия самих недель.

mazm(30,3) - массив с таким именем является одним из самых важных. В расписании основными данными у нас являются предмет, преподаватель и аудитория - 3 ячейки на одну пару. Максимальное число пар в день - 5, подгрупп - 2, 3x5x2==30 - именно таким числом полей мы можем представить на экране данные по расписанию для одной группы на один день, причем элементы с номерами вида 1+3*n отвечают за данные по предмету, 2+3*n - по преподавателю, и 3+3*n - по аудитории. Первая ячейка каждой строки массива содержит код, вторая - соответствующий ему текст (списки, кстати, организованы наоборот - сначала текст, а затем уже код), третья имеет смысл только для поля предмета и содержит уровень дробления потока:

1 - поток,

2 - группа,

3 - подгруппа.

first - это всего лишь флаг первого вхождения в окно расписания. Он необходим для того, чтобы отличит инициализационные действия от многократно повторяющихся.

IsQuitFlag - определяет условие выхода из цикла обработки сообщений от меню. VALID-функция для этого цикла возвращает значение этого флага. Если он очищен, цикл продолжается, при взводе - завершается.

Наиболее предпочтительным методом включения вертикальных меню в приложении является использование системного горизонтального меню FoxPro и его вертикальных меню. При использовании вертикальных меню системного горизонтального меню нет необходимости вначале определять вертикальные меню, более того - это приведет к ошибке. _MSYSMENU - это название системного меню. Определив меню и процедуры - реагенты, мы организуем безоконный цикл чтения состояния меню до тех пор, пока VALID-функция IsQuit не прервет его. Функция IsQuit нигде в программе непосредственно не вызывается; она является VALID-условием, зацикливающим объединяющий READ, обрабатывающий сообщения от меню.

Уйти красиво в программе бывает даже важнее, чем в жизни. Поэтому мы и завели процедуру, реагирующую на выбор в меню пункта Quit. У нас 9 баз данных, размещенных в разных рабочих областях и перед выходом мы, естественно, должны их закрыть; если в базе были сделаны какие-либо изменения, по правилам хорошего тона ее желательно упаковать.

Изменялась ли база, мы можем узнать, просмотрев массив WasChanged, поэтому организуем цикл Просмотр массива - Упаковка (только если были изменения) - Закрытие базы данных.

Команда SELECT не может принять в качестве параметра временную переменную, поэтому воспользуемся описанной ранее функцией макроподстановки. Для этого переведем номер рабочей области в строку и подставим в SELECT содержимое этой строки.

Затем мы восстановим стандартное системное меню, выведем системной функцией на экран заставку FoxPro, и очистим все READ - циклы. Для объединяющего READ, обрабатывавшего наше меню, мы взведем флажок выхода - мол, хватит работать, после чего наша программа благополучно завершится.

Нижеприведенный алгоритм позволит нам заполнить/модифицировать базу данных по преподавателям.

Перейдя в рабочую область БД по преподавателям, мы для удобства редактирования добавим в конец пустую запись. Затем, после необходимых проверок, заполним список названиями и кодами кафедр. При этом, если БД по кафедрам уже изменялась, ее надо упаковать, так как команда копирования базы в массив не обращает внимания на то, что запись может быть пустой или помеченной к удалению, а занесение такой записи в список кафедр (обычно имеющей вид .F.) нежелательно. Далее, найдя в списке кафедр ту, которая соответствует коду в первой записи БД по преподавателям, мы запоминаем этот номер, чтобы в дальнейшем подсветить его в списке, но уже на экране.

Сконструировав список на экране из элементов массива list_kuf, мы будем заносить выбор в переменную ch_kuf. VALID - функция для списка будет вызываться при выборе его элемента с двумя параметрами: кодом кафедры и ее названием. Эта функция код занесет в базу данных, а название отобразит на экране.

В поля БД с именами rank и zvan мы будем заносить номер элемента в соответствующих кнопках - меню, а в поле fio - информацию из окна редактирования.

Сбоку экрана создадим серию из 5 кнопок, одна из которых будет по умолчанию (\!), то есть она будет «нажиматься» по комбинации Ctrl+Enter, а другая (\?) будет реагировать на нажатие клавиши Esc. Для отдельной обработки каждой кнопки в VALID - функцию мы передадим ее номер.

Свои действия в этой функции мы будем сопровождать разнообразными проверками: на выход за пределы базы, на пустоту записи и т.д.

После формирования всех полей на экране и заполнения их информацией из первой записи мы отключим системное меню и организуем цикл обработки сообщений, прерываемый по кнопке «Выход». Восстановив и отобразив меню, мы очищаем экран и занимаемся гигиеной: ищем пустые записи и удаляем их из базы.

Казалось бы, мы сделали уже все, однако фамилию преподавателя мы ввели, а код - нет! И не надо: код мы сформируем программно, по одному и тому же алгоритму во всех местах, суть которого заключается в следующем.

Если код у нас не сформирован, то в соответствующем поле базы стоит нолик, поэтому мы ищем в базе запись с нулевым кодом. Найдя ее, заводим переменную с начальным значением 1 и наращиваем ее до тех пор, пока она совпадает хотя бы с одним кодом из уже имеющихся в базе. Главный критерий кода - уникальность, поэтому, найдя число, для которого нет совпадений среди уже имеющихся кодов, мы можем его принять за handle для данной записи, и заменить пустой (нулевой) код этим числом.

Обслуживание базы данных по факультетам ведётся очень примитивным образом: мы вначале переходим в рабочую область, где открыта эта база, а затем мы открываем BROWSE - окно для ее модификации. Изменять мы позволяем только одно поле - name (название кафедры), так как ее код мы сформируем программно. Вместо имени поля, как это делает BROWSE по умолчанию, мы выводим надпись «Название факультета», а сама команда BROWSE выполняется в специально определенном ранее окне с именем BrowseWin. По окончании сеанса редактирования записей мы проделываем традиционные действия - помечаем к удалению имеющиеся, но незаполненные записи и формируем коды для новых записей.

Структура базы данных по кафедрам ничем не отличается от структуры БД по факультетам, поэтому и процедуры их обслуживания практически идентичны.

Каждая специальность привязана к какому - либо факультету, поэтому мы должны это сделать и в базе. Для этого в БД по специальностям заносится код факультета и название специальности.

Перейдя в рабочую область БД по специальностям, мы для удобства редактирования добавим в конец пустую запись. Затем, после необходимых проверок, заполним массив названиями и кодами факультетов. При этом, если БД по факультетам уже изменялась, ее надо упаковать, так как команда копирования базы в массив не обращает внимания на то, что запись может быть пустой или помеченной к удалению, а занесение такой записи в список факультетов (обычно имеющей вид .F.) нежелательно. Далее, найдя в списке факультетов тот, который соответствует коду в первой записи БД по специальностям, мы запоминаем этот номер, чтобы в дальнейшем подсветить его в списке на экране.

Все наши действия мы будем проводить не на экране, а в окне, которое мы определим и активизируем так, чтобы создаваемы нами экранные объекты отображались на экране только после повторной активации окна. Это делается для того, чтобы прорисовка окна, кнопок, списка и т.д., выполняемые FoxPro в процессе создания экранных объектов очень медленно, не были видны.

Сконструировав список на экране из элементов массива list_fac, мы будем заносить выбор в переменную ch_fac. VALID - функция для списка будет вызываться при выборе его элемента с двумя параметрами: кодом факультета и его названием. Эта функция код занесет в базу данных, а название отобразит на экране.

В поле БД с именем name занесем информацию из окна редактирования.

Сбоку экрана создадим серию из 5 кнопок, одна из которых будет по умолчанию (\!), то есть она будет «нажиматься» по комбинации Ctrl+Enter, а другая (\?) будет реагировать на нажатие клавиши Esc. Для отдельной обработки каждой кнопки в VALID - функцию мы передадим ее номер. Свои действия в этой функции мы будем сопровождать разнообразными проверками: на выход за пределы базы, на пустоту записи и т.д.

После формирования всех полей на экране и заполнения их информацией из первой записи мы отключим системное меню и организуем цикл обработки сообще-

ний, прерываемый по кнопке «Выход». Восстановив и отбразив меню, мы удаляем окно и занимаемся гигиеной: ищем пустые записи и удаляем их из базы.

Для заполнения БД по предметам мы могли бы воспользоваться и @...GET - объектами, но не менее красиво это можно сделать одной из разновидностей команды BROWSE - CHANGE. Если BROWSE предъявляет данные в виде таблицы, то CHANGE - в виде своего рода набора карточек: каждая запись выводится в несколько строк, в каждой строке находится поле в виде «название - содержимое».

По закрытию окна редактирования мы, как обычно, удаляем пустые записи (пустой будем в дальнейшем считать запись, у которой не заполнено хотя бы одно поле) и формируем коды для новых специальностей.

Окно, в общем то, вещь достаточно условная - с помощью «изобразительных средств» самого FoxPro (@... - команд) его можно сымитировать с достаточно высокой степенью достоверности, что мы и попробуем сделать на примере обслуживания БД по аудиториям.

Вначале мы перейдем в рабочую область, где расположена база данных по аудиториям, добавим в конец пустую запись (для удобства редактирования) и установим признак изменения данных в массиве WasChanged. Заметим, что признак специализации аудитории у нас поле логическое, а в случае использования управляющих элементов в стиле Windows мы вынуждены использовать только числовые поля; поэтому заведем в программе временную переменную, в которой будем помещать 1, если аудитория не специализирована и двойку при специализации.

На экране командой @1,5 TO 23,69 DOUBLE мы начертим окно без тени с двойной рамкой, в котором расположим поля ввода номеров корпуса и аудитории, выпадающее меню для выбора емкости аудитории и две радио - кнопочки для выбора типа аудитории (признака специализации). Для корректной замены данных по специализации в базе мы придадим этим кнопкам VALID - функцию swap, преобразующих число из переменной temp в логическое значение по описанному выше признаку.

В отдельном окошке у нас будет особое мемо-поле, содержимое которого не контролируется, но всегда отображается.

После отработки цикла сообщений мы, как обычно, очищаем базу от пустых записей и формируем код.

Основная база данных, содержащая расписание занятий, своими полями имеет только коды, и это не удивительно - текстовое представление этих полей хранится во вспомогательных базах.

При заполнении расписания очень желательно, чтобы вид экрана был максимально приближен к привычной нам (а особенно диспетчеру) форме; это позволит, возможно, сделать рассматриваемую нами программу используемой практически, а не только в качестве учебного пособия. Для этого мы сделаем наш экран рамкой, «ползущей» вверх/вниз по дням и влево/вправо по группам; тогда мы сможем на нем видеть обе подгруппы для одной группы и один день, то есть в общей сложности 2*5=10 клеточек расписания (если брать в расчет только одну неделю). В каждой клеточке у нас будет 3 поля: предмет, преподаватель и аудитория.

За единицу редактирования возьмем один поток. Тогда нам понадобится предварительно ввести сведения о факультете, специальности на этом факультете, потоке по специальности и виде недели (числитель/знаменатель).

После того, как все эти данные нам известны, мы можем выдать на экран окно с EDIT - полями и запустить цикл обработки сообщений от кнопок. В процессе работы с окном расписания мы производим переопределение функциональных клавиш F4 - для изменения данных в поле и F8 - для удаления нескольких полей. Для того, чтобы по окончания программы у нас по этим клавишам выполнялись стандартные действия, нам необходимо перед командой READ сохранить значения функциональных клавиш во внутреннем стеке FoxPro, а после того, как она отработает - восстановить.

Процедура освежения экрана, вызывающаяся при начальной инициализации и любых других изменениях базы/экрана, имеет следующий алгоритм:

Прежде всего мы заносим на экран номер группы, день недели и очищаем все поля нашего массива, отвечающего за EDIT - поля в окне, после чего начинаем заносить данные по лабораторным занятиям, делая это в очень простом цикле по номеру подгруппы. Найдя запись по номеру подгруппы, мы вычисляем ее номер и проверяем, в этот ли день проходит данная пара. Если день и неделя совпали, то мы переписываем все необходимые нам коды во временные переменные и по этим кодам в соответствующих базах данных формируем текстовые поля.

Текстовое поле для аудитории мы формируем из номера этой аудитории и номера корпуса, в которой она располагается, а текст для преподавателя - с помощью

специальной функции, которая формирует строку вида «Доц. Полищук А.П.» из полного имени и кода ученого звания.

Коды и текстовые поля мы заносим в соответствующие элементы массива. Между номером подгруппы, номером пары, номером EDIT - поля в паре (фактически определяющим, какого рода данные находятся в данном поле) и индексом элемента в массиве существует взаимно однозначное соответствие, устанавливаемое функцией GetObject. В дополнительный, третий столбец строки предмета мы заносим цифру 3 - признак лабораторного занятия. Сформировав полностью данные в массиве, мы командой show get принудительно отображаем их в окне.

Данные по практичкам и лекциям заносятся также, за одним исключением: на экране светятся у нас две подгруппы, а данные по практичкам/лекциям идут на группу/поток, поэтому для редактирования мы делаем доступными данные лишь в одной подгруппе - во второй они будут изменяться автоматически. Во избежание случайного ручного изменения в поля-дубликаты заносятся только тексты, без кодов, и редактирование их запрещено.

По нажатию клавиши F8 мы удаляем данные по паре. Делать будем это так:

В FoxPro есть специальная системная переменная _CUROBJ, в которой хранится номер текущего GET - объекта. Всего на экране 5*2*3==30 EDIT - полей и 5 кнопок. Нам интересен лишь тот случай, когда данная функциональная клавиша нажата на одном из полей редактирования. По его номеру мы с помощью функции GetWhat определяем тип поля, в котором мы находимся - то ли это поле с данными по предмету, то ли по преподавателю, то ли по аудитории, и в зависимости от типа поля корректируем переменную _i так, чтобы она указывала на поле предмета.

Сформировав индекс поля предмета в массиве, мы проверяем код на пустоту - а занесены ли там данные по предмету, и, если они присутствуют, а пользователь уверен в том, что удалить эту пару ему жизненно необходимо, мы, в зависимости от типа пары формируем код группы и пары для поиска. Найдя искомую пару в БД, мы ее удаляем, освежаем экран и восстанавливаем позицию курсора на том же месте, с которого начали.

3. Управление измерительным оборудованием

3.1. Приём и обработка спектрометрической информации

В 1989 г. лаборатория радиационной автоматики и радиоактивных изотопов (ЛРИРА) Криворожского металлургического комбината «Криворожсталь» была автоматизирована специалистами Института ядерных исследований с помощью микропроцессорного комплекса на базе анализатора АМА-03Ф и компьютера ДВК-2. В связи с выходом из строя сначала платы сопряжения анализатора и ЭВМ, а затем и самой ЭВМ начиная с 1993 г. все данные с анализатора снимались и обрабатывались вручную, что значительно влияло на качество измерений и степень достоверности делаемых выводов. Такая ситуация привела к необходимости восстановления МПК на новой основе, посредством сопряжения анализатора с ПЭВМ типа IBM PC AT. Распространённость последних даёт шанс на то, что такая «связка» будет служить дольше, чем предыдущая, снимая вопрос с заменой устаревшей и/или вышедшей из строя техники.

Первой задачей, которую необходимо было решить, была организация связи анализатора с ЭВМ. На плате анализатора имелся выход ИРПС, а ЭВМ не имела соответствующей платы сопряжения этого интерфейса с интерфейсом RS-232C. В связи с этим была разработана плата развязки, преобразующая соответствующие значения токовых сигналов ИРПС в сигналы для COM-порта компьютера и наоборот. Мы не будем останавливаться на технических аспектов реализации интерфейса и приёма данных с анализатора на ЭВМ, а сразу перейдем к описанию алгоритмов программы их обработки, написание которой и было целью работы.

Программа «GAMMA» предназначена для проведения измерений активности проб по гамма-излучению, визуального наблюдения информации и автоматической обработки спектров энергии гамма-излучения. Обработка заключается в извлечении информации о положении и интенсивности спектральных линий (по площади фото-

пиков), их идентификации по энергиям с радионуклидами каталога, расчета удельной активности в образцах.

Интерфейс с пользователем реализован как многослойный. Это позволяет работать с программой практически неподготовленному пользователю, т.к. весь цикл операций от считывания спектра из анализатора до выдачи итогового протокола на любое устройство может выполняться выбором одного режима АКТИВНОСТЬ, что особенно удобно при измерении большого потока однотипных проб. Для квалифицированного специалиста по мере освоения программы она может стать гибким и удобным инструментом для любых спектрометрических измерений.

Обработка заключается в извлечении информации о положении и интенсивности спектральных линий (по площади фотопиков), их идентификации по энергиям с радионуклидами каталога, расчета удельной активности в образцах. Конечный результат может вычисляться в любых единицах (Ки и Бк на единицы веса, объёма, площади), на момент измерения либо на заданный момент времени.

Имеется возможность передавать результаты обработки в базу данных произвольной структуры прямо из программы, избежав утомительного ручного ввода и ошибок, связанных с этим. Программа обладает значительной универсальностью За счет возможности подключать один из двух вариантов автоматического поиска пиков, изменять параметры поиска, калибровки спектрометра, библиотеки радионуклидов, режима ручной разметки спектра в графическом режиме и т.д.

Анализатор АМА-03Ф обладает своей сенсорной клавиатурой, графическим дисплеем и широким набором функций управления измерениями, визуализации спектра и его обработки. Интерфейс анализатора с ЭВМ позволяет посылать с ЭВМ все команды сенсорной клавиатуры и пересылать спектр и служебную информацию о нем из анализатора в ЭВМ. Т.к. программа GAMMA имеет собственную графику и более развитую обработку спектра, чем с помощью функций анализатора, то в нее включены только основные функции управления измерениями, передачи спектра и настройки анализатора. Следует обратить внимание на то, что анализатор может управляться одновременно от собственной клавиатуры и ЭВМ, но если включен режим НАБЛЮДЕНИЕ с клавиатуры анализатора во время набора спектра, то команды от ЭВМ не обрабатываются, т.к. процессор анализатора постоянно занят обслуживанием этого режима.

3.2. Описание программы управления анализатором и обработки спектров

3.2.1. Режимы работы программы GAMMA

Выполняемые из программы режимы:

СЧИТЫВАНИЕ СПЕКТРА. Время экспозиции на момент последнего останова.

СЧИТЫВАНИЕ СПЕКТРА ПРИ ИЗМЕРЕНИИ. Выполняется останов измерения, режим «наблюдение», передача спектра в ЭВМ и запуск измерения дальше. Время измерения передается на момент останова.

ОЧИСТКА. Очистка буфера накопления анализатора АМА03Ф.

СТАРТ. Запуск измерения на АМА03Ф в установленном режиме.

СТОП. Останов измерения на АМА03Ф.

НАБЛЮДЕНИЕ. Выдача содержимого буфера накопления АМА03Ф на дисплей анализатора.

АВТОМАСШТАБ. Установка автоматического масштаба отображения спектра по вертикали на АМА03Ф.

ВРЕМЯ ЖИВОЕ. Задание экспозиции по живому времени. 0 - без задания экспозиции.

ПОРТ, БУФЕР. Установка номера порта, к которому подсоединен анализатор и длины буфера накопления в АМА03Ф (1, 2, 4, 8K). Установка длины буфера происходит только при изменении текущего состояния.

Движение по строке основного меню осуществляется клавишами управления курсором. Для движения внутри подменю используются вертикальные стрелки. Чтобы сообщить программе на каком пункте меню остановлен выбор, следует подвести под него курсор и нажать <ВВОД>. После этого программа начнет выполнять выбранный режим или выведет меню следующего уровня. Иногда в меню третьего уровня предлагаются альтернативные возможности (выбрать тип спектра, дисковод, спектр для обработки и т. д.). В других случаях можно выбрать несколько пунктов одновременно (для настройки параметров поиска пика, расчета активности и т. д.).

Кроме удобных средств движения по меню, в программе GAMMA предусмотрены стандартные средства редактирования вводимой информации.

Клавиша Ф10 почти всегда прерывает начатую операцию с выходом в меню.

Рассмотрим работу всех режимов программы, указанных в меню.

3.2.2. Управление файлами

Этот режим позволяет работать с файлами. На втором уровне он имеет следующие возможности.

СЧИТЫВАНИЕ СПЕКТРА С ДИСКА. Выбранный спектр считывается в оперативную память и становится доступным для обработки. При приёме спектра с анализатора каждому каналу соответствует трёхбайтное значение, которое, в связи с отсутствием 3-хбайтного типа, сохраняется на диске как длинное целое размером в 4 байта.

ЗАПИСЬ СПЕКТРА НА ДИСК. Позволяет записать спектр, находящийся в оперативной памяти на указанный пользователем диск (по умолчанию активный) в формате программы GAMMA.

КОМАНДЫ DOS. Данный выбор открывает меню третьего уровня и позволяет без выхода из программы выполнять все команды MS DOS, запускать при наличии свободной памяти другие программы, редактировать, просматривать и выводить на принтер текстовые файлы и т.д.

ДРУГОЕ МЕНЮ. При загрузке программы выдается основное меню, достаточное для обработки результатов серийных измерений в случае настроенной программы. Данный режим позволяет перейти к полному меню (калибровки, просмотр спектра и т.д.) и обратно. Необходимо помнить, что режимы, не включенные в основное меню, требуют достаточно высокой квалификации в спектрометрии и при неумелом их использовании могут привести к искажению величин, влияющих на конечный результат.

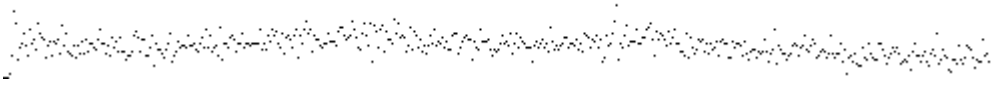
ВЫХОД В DOS. Завершает работу программы и возвращает управление ОС.

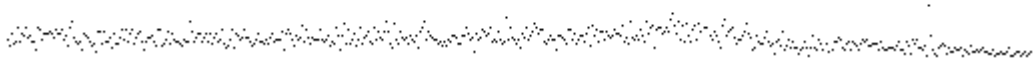
3.2.3. Расчёт активности.

Выбор позволяет произвести полную автоматическую обработку спектра.

АКТИВНОСТЬ. Этот режим позволяет при установке соответствующих параметров проводить полный цикл обработки спектра от считывания из буфера до вывода документа в одном из четырех возможных форматов на принтер, в зависимости от установленных значений. Если поиск пиков в спектре не производился другими режимами, то после отработки режима (Поиск 1 или 2 в зависимости от настройки) на дисплее появляется сообщение об этом в виде таблицы. Перед расчетом активности выдаются значения всех параметров, участвующих в расчете и спрашивается, устраивают ли нас эти значения. При отрицательном ответе предоставляется возможность ввести последовательно значение всех параметров либо при задании номера изменяемого параметра ввести только его. Программа производит анализ корректности задания параметров (вес пробы, время измерения и эффективность для заданной геометрии не равны нулю) и в случае невыполнения условий выдает сообщение и запрашивает правильное значение параметра. В более удобном варианте параметры могут изменяться в режиме ПАРАМЕТРЫ ДЛЯ АКТИВНОСТИ, где и описано назначение каждого из них. Расчет активности происходит по полной формуле с учетом распада короткоживущих радионуклидов за время измерения и с учетом распада с момента отбора если заданно ненулевое время выдержки. Если в библиотеке для данного радионуклида занесено несколько энергий гамма-переходов, то сначала рассчитывается активность по каждому идентифицированному с данным нуклидом пику, а затем вычисляется средневзвешенная активность нуклида с весами обратно пропорциональными стат. погрешностям площадей пиков.

Если измерения происходят в условиях значительного фона или измеряемые активности близки к фоновым, то для расчета активности нужно пользоваться этим режимом. Программа производит автоматическое вычитание фона, предварительно записанного режимом ЗАПИСЬ ФОНА НА ДИСК, в случае включения соответствующего флага в режиме НАСТРОЙКА РАСЧ. АКТ. Примеры фоновых спектров:





ПАРАМЕТРЫ ДЛЯ АКТИВНОСТИ. Позволяет изменить параметры, для расчета активности в оконном режиме.

ВРЕМЯ ИЗМЕРЕНИЯ. Позволяет изменить время измерения спектра. Устанавливается автоматически при считывании спектра с анализатора, но для анализатора АМА эта величина не всегда соответствует реальному времени измерения (добор экспозиции, считывание во время измерения). В этом случае время заносится вручную.

ВРЕМЯ ВЫДЕРЖКИ. Если этот параметр не равен 0, то активность рассчитывается не на момент измерения, а на момент отбора, отстоящий от момента измерения на заданное количество минут. Используется в задачах технологического контроля по короткоживущим радионуклидам либо при проверке калибровки по эффективности для приведения активности контрольного источника на момент его аттестации.

ОБЪЕМ (ВЕС) ПРОБЫ. Задается объем либо вес измеряемой пробы для получения объемной или удельной активности. При необходимости получить абсолютную активность пробы нужно задать 1.

ЕДИНИЦЫ. Единицы, в которых задается вес или объем. На расчеты не влияет и имеет смысл комментария в выходном документе. Будет указано, что активность рассчитана в Ки и Бк на заданную величину (кг, л и т.д.).

ПОЛНЫЙ ВЕС ПРОБЫ ДЛЯ КИ/КМ2. СУММАРНАЯ ПЛОЩАДЬ ОТБОРА. Параметры для расчета поверхностной активности (Ки/ и Бк/ на квадратный километр). Предполагается отбор пробы методом конверта (пять уколов на участке площадью 1 $м^2$ на глубину 25-30 см пробоотборником известной площади, полученный грунт тщательно перемешивается и после этого отбирается одна или несколько проб для измерения). Полный вес - суммарный вес всех уколов; суммарная площадь отбора - площадь пробоотборника в $см^2$, умноженная на количество уколов.

ГЕОМЕТРИЯ. Номер, под которым занесена информация по эффективности для данной геометрии измерения.

ГРУППА НУКЛИДОВ. Если =0, то в спектре ищутся все нуклиды из активной библиотеки, иначе только нуклиды из указанной группы.

НАСТРОЙКА РАСЧЕТА АКТИВНОСТИ. Параметры определяют действия, выполняемые программой по режиму АКТИВНОСТЬ. Позволяет выбором одного режима выполнять полный цикл действий, что особенно удобно при измерении большого потока однотипных проб. Установка ниже перечисленных параметров включает или отключает соответствующее действие при выполнении режима АКТИВНОСТЬ.

СЧИТЫВАНИЕ СПЕКТРА ИЗ БУФЕРА. Если =1, то перед обработкой происходит автоматическое считывание спектра из буфера анализатора.

ПРОТОКОЛ НА ПЕЧАТЬ/ДИСК. После расчета активности предлагается вывод итогового документа на принтер или в файл на диск. Есть возможность отказаться от вывода.

ПРОТОКОЛ ПОЛНЫЙ/КРАТКИЙ. Определяет объем информации, выводимой в итоговый документ. В краткий протокол выводятся значения параметров, используемых для расчета активности и итоговая таблица активностей идентифицированных в спектре нуклидов с указанием погрешностей и процентного вклада нуклида в общую активность пробы. В полный протокол, кроме этого выводится верхняя и нижняя шапки (содержимое текстовых файлов TOPDOC.DAT и ENDDOC.DAT, которые могут редактироваться пользователем программы), полная таблица найденных в спектре пиков (положение, энергия, площадь, погрешность определения площади, идентифицированный с данным пиком нуклид и его активность).

АКТИВНОСТЬ/СИЧ. Изменяет режим расчета активности и форму итогового протокола. При измерении на СИЧ рассчитывается содержание CS-137 в мкКи в человеке и годовая доза по следующей формуле:

$$D(\text{бэр/год}) = \frac{A(\text{мкКи})}{6.6} \cdot \frac{70}{m(\text{кг})}$$

Перед расчетом предоставляется возможность ввести новый вес или оставить старый.

ВЫЧИТАНИЕ ФОНА. При расчете активности если для пика с данной энергией имеется аналогичный пик в таблице интенсивностей фоновых пиков (создается режимом ЗАПИСЬ ФОНА), то рассчитывается возможный вклад от фонового пика и вычитается из площади найденного. Если результат отрицательный или меньше дисперсии найденного пика, то активность по этому пику дается равной 0.

РАСЧЕТ МДА. При включении этого флага после итоговой таблицы активностей идентифицированных в спектре нуклидов под заголовком «Активность не больше:» выдается список не найденных в спектре нуклидов из активной группы библиотеки и величина их минимально детектируемой для данного спектра активности (МДА). Расчет МДА выполняется следующим образом: из калибровки по энергии находится место в спектре с энергией библиотечного пика, из калибровки по энергетическому разрешению спектрометра определяются возможные границы пика в данном месте и определяется площадь возможного пика над фоновой подставкой. Если эта площадь превышает два корня из фона, то для расчета МДА берется она (ужесточением критериев отбора пиков мы можем не включить в таблицу найденных даже пики значительной площади и эта возможность таким образом учитывается); в противном случае для расчета МДА в качестве площади пика берется два корня квадратных из фоновой подставки.

ДОКУМЕНТ. Позволяет вывести на принтер и (или) записать на диск всю полученную в результате отработки программы информацию. Можно ввести любые комментарии, которые запоминаются и при следующем выводе могут быть отредактированы.

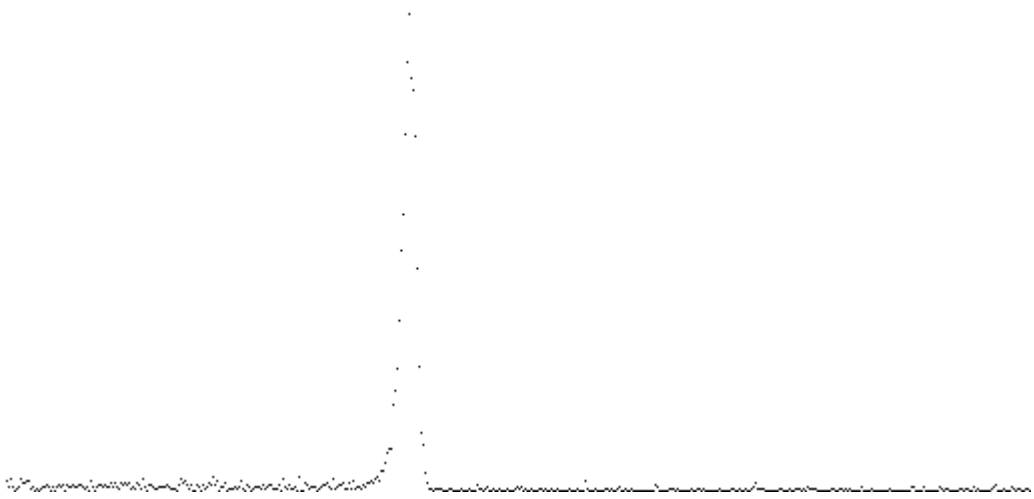
СЧИТЫВАНИЕ СПЕКТРА ИЗ БУФЕРА. Читает спектр из буфера анализатора АМА-03Ф4. При установке соответствующего флага в режиме ПАРАМЕТРЫ ДЛЯ РАСЧЕТА АКТ. производится автоматическое выполнении режима АКТИВНОСТЬ.

3.2.4. Обработка спектра

Содержит режимы обработки спектра, выдачи информации о спектре и результатах обработки спектра и т.д.

ПОИСК ПИКОВ 1. Автоматический поиск пиков в спектре и определение их площади путем анализа сглаженных первых и вторых производных и их погрешностей (см. рис.). Хорошо работает для пиков с полной шириной на половине высоты (ПШПВ) не превышающей 10-20 каналов, что в большинстве случаев реализуется для спектров, полученных с помощью полупроводникового детектора. Параметры, влияющие на поиск, могут изменятся в режиме НАСТРОЙКА ПРОГРАММЫ.

На дисплее печатает таблицу, содержащую: N п/п; служебную информацию; положение пика в каналах; его энергию в кэВ; площадь пика.



ПОИСК ПИКОВ 2. Для поиска пиков используется модифицированный алгоритм бегущего отрезка, длина которого вычисляется автоматически из зависимости энергетического разрешения от энергии, полученной при калибровке тракта в режиме ЭНЕРГИЯ, ПШПВ. Точность определения площадей найденных пиков также зависит от этой калибровки. Хорошо работает для пиков любой ширины и площади вне зависимости от формы вершины пика. Рекомендуется для обработки спектров в случае пиков очень маленькой площади с несформированной вершиной на пределе обнаружения.

ИНИЦИАЛИЗАЦИЯ ПОИСКА 2. Используется, если найденные пики не соответствуют реальному спектру. Это происходит при несоответствии калибровки по энергетическому разрешению реальным параметрам тракта. Режим устанавливает калибровку по энергии энергия=номеру канала и по энергетическому разрешению такую, что во всем интервале обработки ПШПВ равно введенной величине. На запрос: ПОЛНАЯ ШИРИНА ПИКОВ НА ПОЛОВИНЕ ВЫСОТЫ нужно ввести ширину интервала обработки в каналах с точностью 1-3 канала. После этого положение пиков будет определяться нормально, но правильный расчет их площади возможен только после калибровки спектрометра по энергетическому разрешению.

ОПРЕДЕЛЕНИЕ ПШПВ. Определяет полную ширину на половине высоты (ПШПВ) для найденных в режиме ПОИСК ПИКОВ 2 пиков (при ПОИСКЕ 1 это выполняется автоматически). Необходимо помнить, что алгоритм автоматического определения ПШПВ успешно работает на хорошо сформированных пиках и предна-

значен для облегчения калибровок по энергетическому разрешению. Для пиков малой статистики или большой ширины величина ПШПВ может быть значительно занижена. Более точные значения в этом случае можно получить в режиме ГРАФИКА с помощью ручной разметки границ пика.

ПЕЧАТЬ ПИКОВ. На дисплей выдается таблица найденных в спектре пиков с указанием положения, энергии, площади, дисперсии площади, ПШПВ (в случае ПОИСКА1) и ПШПВ по калибровке (в кэВ) и их разницу в %.

ПЕЧАТЬ СПЕКТРА. Можно распечатать на дисплей или принтер (по желанию оператора) количество импульсов в любом канале анализатора.

ГРАФИКА. Позволяет просмотреть спектр в графическом режиме. Клавиши управления графикой:

Стрелки вправо, влево	движение маркера
Стрелки вверх, вниз	увеличение, уменьшение масштаба по вертикали. При перелистывании спектра (< , >) возвращается автомасштабирование по вертикали;
F1	помощь;
F3, F4	увеличение/уменьшение скорости маркера по 10 каналов
F5	скорость маркера равна 1
F7	рисовать точками/линиями
F8	выключение/включение звукового сопровождения
F10	прерывание любой операции с выходом в меню (во всей программе);
+, -	растяжение, сжатие спектра по горизонтали с центром изображения в положении маркера;
<, >	смещение (перелистывание) экранного окна влево, вправо по спектру с шагом равным количеству каналов на экране масштаб по вертикали автоматический, спектр замкнут по кольцу;
Z	смещение экранного окна в начало спектра либо маркера в нулевой канал если спектр изображен с начала;
0	переключение режима масштабирования изображения от 0 по вертикали

	на масштабирование от минимума в окне и обратно;
C	маркерный канал в центр экрана;
1,2	положение маркера запоминается как границы пика или участка спектра для малого окна;
O	выделенный участок изображается в малом окне в правом верхнем углу экрана. При выводе линиями (F5) закрашивается спектр. Маркер переходит в малое окно без контроля границ. В правом верхнем углу экрана выдаются граничные каналы окна и сумма импульсов в выделенном окне.
X	задание экранных координат левого нижнего угла малого окна в абсолютных координатах экрана (0-верхний левый угол, по горизонтали - 640 точек, по вертикали - 200);
R	восстановление большого окна в прежнем виде. Очищается экран и выводится содержимое большого окна с управляемым маркером. Используется после режимов P или O где управление маркером передается в малое окно. Аналогично действуют команды +, -, <, >.
P	выделенный участок обрабатывается как пик. Фон определяется по трем точкам слева и справа от границ. На экран выдается изображение малого окна с проведенной фоновой линией и текстовая информация: положение модального канала; ПШПВ; площадь пика (если <1 то =1); относительная дисперсия площади (в долях); границы участка.
T	внесение информации о пике в таблицу обработки с упорядочением. Выдается номер пика в таблице. Созданная таким образом таблица пиков может быть использована в основном меню аналогично полученной автоматическим поиском («Печать пиков», «Активность», «Протокол»). Если в таблице пиков уже был пик с таким же положением, то его параметры заменяются на полученные по режиму P.
TAB	по нажатию клавиши «Табуляция» в малом окне высвечиваются пики, в границах, выбранных программой. Перемещение по таблице найденных пиков в порядке возрастания. Shift +TAB - в порядке убывания.

3.2.5. Настройка программы

Содержит режимы, позволяющие ввести или изменить различные параметры и калибровочные коэффициенты, используемые при обработке спектров.

НАСТРОЙКА ПРОГРАММЫ. Позволяет изменить настроечные параметры программы.

НАЧАЛО и КОНЕЦ УЧАСТКА ОБРАБОТКИ. Определяет интервал спектра для поиска пиков и позволяет сократить время обработки спектров. Значения в каналах. Начало >=6, конец участка не должен превышать длину спектра -6. Контроль правильности производится автоматически при выходе из настройки.

ДЛИНА СПЕКТРА. Влияет на длину записываемого на диск спектра и на доступную для просмотра часть спектра в режиме ГРАФИКА. При считывании спектра из буфера или с диска значение длины автоматически заменяется на соответствующее читаемому спектру.

КРИТЕРИЙ КРИВИЗНЫ ВЕРШИНЫ К ПОИСКУ 1. Отношение величины сглаженной второй производной к ее ошибке в точке вершины возможного пика. Позволяет отсеивать комптоновские края и пикоподобные флуктуации фона, имеющие большую кривизну, чем пики полного поглощения. Значения подбираются опытным путем на спектре, изученном визуально. Рекомендуемый диапазон 0.4-3.

ОКНО ИДЕНТИФИКАЦИИ. Максимальное отклонение энергии найденного пика от библиотечной энергии, при котором пик идентифицируется как принадлежащий данному нуклиду. При заниженном значении параметра пики могут не идентифицироваться за счет временной нестабильности тракта по энергетической калибровки; при завышенном значении и наличии в библиотеке нуклидов с близкими энергиями пик может быть идентифицирован с несколькими нуклидами и в этом случае будет использован при расчете активности каждого из них.

КРИТЕРИЙ ОТБОРА ПИКОВ (N корней из фона). Критерий отбора пиков по превышению N дисперсий фона. Рекомендуемые значения 6-9 при работе с пиками нормальной статистики и 2-4 при поиске пиков на пределе обнаружения.

ИМЯ БИБЛИОТЕКИ НУКЛИДОВ. Изменение имени активной библиотеки радионуклидов. Библиотека с указанным именем должна находиться на диске. При выходе из режима настройки она будет считана в программу взамен предыдущей. При

запуске программы считывается библиотека с именем, которое было установлено перед последним выполнением режима СОХРАНИТЬ НАСТРОЙКУ.

В РЕЖИМЕ 'АКТИВНОСТЬ' ПОИСК 1 ИЛИ 2. При выполнении режима АКТИВНОСТЬ без предварительного поиска пиков будет выполняться поиск 1 или 2 в зависимости от значения этого параметра.

ЭНЕРГИЯ. ПШПВ. Режим дает возможность откалибровать спектр по энергиям и по полуширине пика на ½ его высоты. С целью повышения надежности определения параметров пиков малой площади и искаженной формы (что часто встречается при малой статистике), в данной программе в качестве алгоритма ПОИСК2 используется метод бегущего отрезка. Площадь определяется с использованием информации об энергетическом разрешении тракта в месте расположения пика. Производится расчет энергетического разрешения калибровочных пиков и вычисляются коэффициенты зависимости разрешения от энергии.

Алгоритм определения энергетического разрешения рассчитан на изолированные пики с хорошей статистикой. Поэтому в качестве калибровочных можно брать пики, полученные с помощью ОСГИ и удовлетворяющие этим требованиям. Загрузка спектрометрического тракта должна быть в пределах нормы, т.к. иначе возможно значительное уширение пиков и искажение их формы.

Программа предлагает для анализа таблицу, содержащую: положение пика в каналах; его энергию в кэВ; площадь пика; ПШПВ в каналах; ПШПВ в кэВ.

На запрос программы: ПОДГОНКА ПОЛИНОМОМ СТЕПЕНИ 1 ИЛИ 2? вводим число 1 или 2.

Просматривая по одному найденные пики, программа предоставляет возможность выбрать точки для калибровки, исходя из положения пика в каналах, его площади и ПШПВ. Если пик пригоден для калибровки - вводим значение энергии в кэВ. Для каждого отобранного пика производится расчет ПШПВ. Далее указывается число пиков, отобранных для калибровки, уравнение для зависимости канал-энергия и энергия - энергетическое разрешение. По нажатию любой клавиши, получим график канал - энергия. Если полученный результат нас удовлетворяет - по запросу программы можем записать его на диск.

ЭФФЕКТИВНОСТЬ. Позволяет рассчитать эффективности измерения для различных геометрий измерения и под номерами от 1 до 20 записать их на диск для

дальнейшего автоматического считывания программой. Режим имеет следующие возможности.

РАСЧЕТ ПО АКТИВНОСТИ. На запрос вводим число, соответствующее номеру, под которым данная калибровка будет записана в файл. Отвечаем на запросы программы:

ЭНЕРГИЯ nnn

АКТИВНОСТЬ, ПОЛУЧЕННАЯ ПРИ РАСЧЕТЕ? nnn

АКТИВНОСТЬ ИСТИННАЯ nnn

вводя соответствующие значения. Для прекращения ввода на очередной запрос ЭНЕРГИЯ нажимаем <ВВОД>. По нажатию любой клавиши, получим график зависимости эффективности от энергии. Если полученный результат нас удовлетворяет - по запросу программы можем записать его на диск.

ВВОД ЭНЕРГИЯ-ЭФФЕКТИВНОСТЬ. Отвечая на запросы программы, вводим энергию линии гамма-излучения и соответствующую ей эффективность измерения. Для прекращения ввода на очередной запрос ЭНЕРГИЯ нажимаем <ВВОД>. На дисплее появляется таблица, содержащая значения: энергии; измеренной эффективности; рассчитанной эффективности; отклонения этих величин в %.

ВВОД КОЭФФИЦИЕНТОВ. Режим дает возможность записать коэффициенты эффективности в калибровочный файл.

ПРОСМОТР КОЭФФИЦИЕНТОВ. Выводит на дисплей таблицу коэффициентов эффективности для всех 20 геометрий.

РАСЧЕТ ЭФФЕКТИВНОСТИ ПО ЭНЕРГИИ. Позволяет рассчитать эффективность измерения для любых энергий и любой из записанных в калибровочный файл геометрий.

БИБЛИОТЕКА. Позволяет просматривать, пополнять библиотеку нуклидов, указанную в режиме НАСТРОЙКА ПРОГРАММЫ и записывать измененную библиотеку под любым именем на диск. Библиотека записывается в текстовом файле с расширением .DAT и указанным именем и может быть создана любым текстовым редактором при соблюдении следующего формата:

1-я строка - целое число, равное количеству радионуклидов в файле;

2-я строка - целое число, соответствующее выделяемой группе нуклидов;

символьная константа, заключенная в двойные кавычки « и являющаяся именем радионуклида, напр. «CS-137»;

вещественное число, являющаяся постоянной распада данного радионуклида в час-1;

целое число, равное количеству последующих строк с данными о гамма-переходах этого нуклида; величины в строке разделены между собой запятыми

3-я строка - вещественное число, равное первой энергии гамма-перехода этого нуклида;

вещественное число, равное выходу гамма-квантов на распад для данной энергии в долях;

4-я строка - аналогична 3-й, если для нуклида дается больше одной энергии или аналогична 2-й, если начинаются данные для следующего нуклида и т.д.

Пример библиотеки:

6

2, «CE-144»,.000102,1

134,.111

2, «RU-106»,.0000785,1

1050,0.015

1, «CS-134»,.0000385,1

795,.94

1, «CS-137»,2.64E-06,1

662,.851

1, «K-40»,6.18E-14,1

1460,.107

2, «EU-152»,5.82E-06,5

344,.27

779,.13

964,.14

1112,.13

1408,.206

СОХРАНИТЬ НАСТРОЙКУ. Производит запись параметров настройки на активный диск. Информация используется при вычислении активности в случае если параметр ВЫЧИТАНИЕ ФОНА в режиме ПАРАМЕТРЫ ДЛЯ АКТ. равен 1.

СПРАВКА. На дисплей выводится информация о калибровках по энергии и по ПШПВ, сообщение о количестве пиков и критериях, использованных при их поиске. Информирует о количестве нуклидов в библиотеке.

3.2.6. Возможные отклонения от нормального выполнения режимов

1. Время измерения спектра =0. В этом случае перед расчетом активности вместо запроса веса выдается полная таблица параметров (время, вес, номер геометрии и т.д.) и вопрос ХОТИТЕ ИЗМЕНИТЬ? (Y/N). Если нажать клавишу Y, появится сообщение «Время измерения = 0» и будет предложено ввести время измерения в секундах и время выдержки в минутах. После этого произойдет расчет активности, либо (что более вероятно) будет сообщение, что весь спектр равен 0. Ситуация может возникнуть при работе, если вы не сделали начальный запуск анализатора, или он находился в режиме ON (с остановом по времени экспозиции) и время истекло. Т.е. была произведена очистка спектра, а дальнейший набор спектра не производился.	Устранение: перейти в меню программы (F10) и с помощью подменю УПРАВЛЕНИЕ АНАЛИЗАТОРОМ устранить причину, переведя анализатор в требуемый режим.
2. Весь спектр равен нулю. Может сопровождать ошибку 1 с теми же причинами, либо в случае отсутствия контакта при подключении детектора к анализатору. Убедиться в этом можно в режиме ГРАФИКА либо ПЕЧАТЬ СПЕКТРА.	Устранение: проверить подключение детектора и повторить набор спектра.

Литература

1. Анализаторы многоканальные амплитудные АМА-03Ф. Техническое описание и инструкция по эксплуатации еФ1.287.004 ТО.
2. Амонашвили Ш.А. Воспитательная и образовательная функции оценки учения школьника. - М.: Педагогика, 1984.
3. Архитектура среды разработки приложений. - К.: Диалектика, 1992.
4. Бабак В.П., Хандецький В.С., Шрюфер Е. Обробка сигналів: Підручник - К.: Либідь, 1996.
5. Брябрин В.М. Программное обеспечение персональных ЭВМ. - М.: Наука, 1989.
6. Вайнер Ричард, Пинсон Льюис. C++ изнутри. - Киев.: ДиаСофт, 1993.
7. Галушкин А.И., Зотов Ю.Я, Шикунов Ю.А. Оперативная обработка экспериментальной информации. - М.: Энергия, 1972.
8. Гершунский Б.С. Компьютеризация в сфере образования: проблемы и перспективы. - М.: Педагогика, 1987.
9. Данкан Рэй. Профессиональная работа в MS-DOS. - М.: Мир, 1993.
10. Джордейн Роберт. Справочник программиста персональных компьютеров типа IBM PC,XT и AT. - М.: Финансы и статистика, 1992.
11. Дьюхарст Стивен, Старк Кэти. Программирование на C++. - К.: ДиаСофт, 1993.
12. Касаткин А.И., Вальвачев А.Н. От Turbo C к Borland C++. - Минск: Вышейшая школа, 1992.
13. Касаткин А.И. Управление ресурсами. - Мн.: Вышэйшая школа, 1992.
14. Касаткин А.И. Системное программирование. - Минск: Вышейшая школа, 1993.
15. Корытин А.М., Петров Н.К., Радимов С.Н., Шапарев Н.К. Автоматизация типовых технологических процессов и установок. - М.: Энергоатомиздат, 1988.

16. Кукуруза П.В. Полнота реляционных систем. // «Компьютеры +программы», № 2/1993, с. 4-8.
17. «Компьютеры +программы», № 6/1995, с. 68-71.
18. Лукас Пол. C++ под рукой. - Киев: ДиаСофт,1993.
19. Машбиц Е.И. Психолого-педагогические проблемы компьютеризации обучения: Педагогическая наука - реформе школы. - М.: Педагогика. 1988.
20. Нортон Питер. Программно-аппаратная организация IBM PC. - М.: Радио и связь, 1991.
21. Нортон Питер, Уилтон Ричард. IBM PC и PS/2. Руководство по программированию. - М.: Радио и связь, 1994.
22. Полищук А.П. Персональный компьютер и его программирование (C, C++, Паскаль). Учебно-справочное пособие. - Кривой Рог, 1997.
23. Полонский В.М. Оценка знаний школьников. - М.: Знание, 1981.
24. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль (версия 5.5.). - М.: МАИ, А/О «РосВузНаука», 1992.
25. Разработка методики гамма-спектрометрии лабораторных образцов и внедрение ее на КМК «Криворожсталь». Отчёт по договору №6/91. Киев, 1991.
26. Ривкинд И.Я., Маргулис Е.Д. Компьютер в школе: Кн. для учителя. - К.: Рад. шк., 1991.
27. Страуструп Бьярн. Язык программирования C++: В 2-х частях. - К.: Диасофт, 1993.
28. Тихомиров О.К. Психологическая структура диалога «человек-ЭВМ» // Вестник Московского университета: Психология, №2, 1984.
29. Фейсон Тэд. Объектно-ориентированное программирование на Borland C++ 4.5. - Киев: Диалектика, 1996.
30. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC. Ч. 1,2 - М.: Диалог-МИФИ, 1992.
31. Фролов А.В., Фролов Г.В. Программирование видеоадаптеров. - М.: Диалог-МИФИ, 1992.
32. Харламов И.Ф. Педагогика: Учеб. Пособие. - М.: Высшая школа, 1990. - 576с.
33. Шилдт Герберт. Си для профессионалов. - М.: Мир, 1989.