ЭФФЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ РЕСУРСОВ КОМПЬЮТЕРА ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ (ФАКУЛЬТАТИВНЫЙ КУРС)

В.А. Юрченко, С.А. Семериков
г. Кривой Рог, Криворожский государственный педагогический университет

Несколько лет назад нами была поставлена задача: разработать двухгодичный факультативный курс по информатике, который можно было бы использовать в старших классах школ нового профиля. В качестве экспериментальной площадки для его апробации была выбрана Центрально-Городская гимназия г. Кривого Рога.

С самого начала, однако, было совершенно неясно, какой язык программирования выбрать в качестве основного, поэтому, с учётом индивидуального характера занятий, было решено составлять программу, привязанную не к языку программирования, а к задачам, которые необходимо было решить в процессе обучения на факультативе. Это пожелание было высказано самими учениками, что достаточно усложнило работу – построенный курс, фактически, существует в двух вариантах – на языке Си и на языке Паскаль.

Первый год обучения на факультативе – пропедевтический. В процессе обучения ученики осваивают тот язык программирования, который они выбирают. Курс построен таким образом, что первый год обучения могут вести два разных преподавателя – один изучает со своей группой язык Паскаль, а другой – Си. Разумеется, конечный результат обучения двух преподавателей, даже по сходным программам, может очень сильно отличаться. В связи с этим в начале второго года обучения, когда, по нашему предложению, лучшие обучаемые из обеих групп сливаются в одну, возникает необходимость в проверке и коррекции усвоенных ими знаний, своего рода выравнивании их знаний перед тем, как перейти непосредственно к усвоению нового материала. Это достигается путём выдачи индивидуальных заданий по решению 10-15 стандартных алгоритмических задач. Каждый ученик получает шаблон решения 2-3 задач на том языке программирова-

ния, который он изучал в течение первого года обучения. Задания выполняются под контролем преподавателя, но в свободной манере; новый учитель не сковывает инициативу своих учеников, желающих, к примеру, приукрасить свою программу, а консультирует их по всем интересующим их вопросам. При этом решаются две дидактические задачи:

1. С одной стороны, учитель в процессе выполнения заданий получает максимум информации об уровне знаний, умений и навыков своего будущего обучаемого, не прибегая к традиционным формам контроля и оценки знаний (т.е. не проводя опроса, среза и т.п.).
2. С другой стороны, предлагаемые шаблоны как бы «навязывают» ученикам свойственный преподавателю стиль изложения материала, помогая им адаптироваться к нему ещё до того, как они начнут решать более серьёзные задачи.

Таким образом, вступительное тестирование (которое, кстати, может быть проведено и в конце первого года обучения совместно с преподавателями языка программирования) помогает преодолеть ученикам известный психологический барьер «нового учителя», а учителю – поближе познакомиться со своими подопечными.

Первая тема, рассматриваемая на факультативе, относится к теории кодирования информации. Будучи фундаментальной во всем курсе информатики, она имеет важное практическое значение при решении любой прикладной задачи: ведь, по выражению Никласа Вирта, программы есть алгоритмы + структуры данных, с которыми работают эти алгоритмы. Одной из важных проблем является ограниченность ресурсов компьютера, в частности в том, что касается дисковой памяти. Поэтому естественно возникает желание, чтобы данные, обрабатываемые программой, занимали как можно меньше места на диске. Это и обуславливает необходимость изучения алгоритмов СЖАТИЯ ИНФОРМАЦИИ для её компактного хранения.

Учитывая, что большинство школ оснащены достаточно слабой техникой – за редким исключением, в лучшем случае класса ХТ без винчестера, основная работа происходит с дискетами (там, где есть дисководы на рабочих местах ученика), поэтому проблема эффективного хранения, а главное – использо-

вания информации (текстов, картинок и т.п.) стаёт особенно актуальной. Стандартные средства – архиваторы – позволяют эффективно хранить информацию, но в этом случае об эффективном её использовании не может быть и речи. Это и является причиной изучения данного раздела; алгоритмы, предлагаемы для изучения, достаточно просты, чтобы быть усвоены школьниками, позволяя создать встраиваемые в программы функции сжатия/расжатия используемой в данной программе информации. Для данных, в которых есть длинные последовательности повторяющихся элементов (например, картинки или отдельные тексты), мы предлагаем метод RLE и Хаффмана, эффективность которых можно оценить из предоставляемой демонстрации. Ученики при этом выбирают тот метод, которые лучше всего решает поставленную задачу.

Одним из двух основных средств ввода информации в компьютер является клавиатура. Основные функции работы с клавиатурой осваиваются, как правило, на самых ранних стадиях изучения языка программирования. Более того, они признаны настолько важными, что, к примеру, в языке Паскаль функции типа read, readln входят в стандарт зыка, составляя его ядро. Поэтому эффективность использования клавиатуры нужно искать не в самом факте его использования, а в тех проблемах, которые мы можем решить, детально разобравшись в структуре нижнего уровня – буфере клавиатуры. Именно непосредственный доступ к буферу клавиатуры позволяет нам операции, которые обычными средствами сделать затруднительно или просто невозможно (например, «подсмотреть» символ, который функцией ввода ещё не считан, но в буфере уже лежит). Одним из ярких примеров эффективного использования буфера клавиатуры является написание демонстрационных программ. Обычно по демонстрации можно узнать, как работает программа в режиме диалога или, к примеру, в режиме «сценария», который проигрывает редактор «Слово и Дело» при первой установке. Поэтому гораздо эффективнее написать сценарий демонстрации для уже существующей программы, чем писать её демо-версию. Именно такого рода обучающие сценарии и являются наиболее эффективным средством ознакомления с программным продуктом. В этом случае преподавателю нет необходимости одновременно рассказывать и

показывать – обучающий сценарий сам, через заданные промежутки времени, заносит в буфер клавиатуры коды клавиш, воспринимаемые демонстрируемым программным продуктом как реальные нажатия клавиш. В качестве примера такого сценария можно привести фрагмент демонстрации некоторых возможностей файлового монитора.

Наибольшим дефицитом в прикладной программе является, конечно, оперативная память. Как правило, большинство программистов используют её нерационально, распределяя малыми блоками, фрагментируя её и, таким образом, уменьшая реально возможный используемый объём. Кроме того, многие задачи требуют хранения данных в одном большом блоке памяти – это может быть, к примеру, графический файл, не умещающийся в пределах сегмента (64 Кб). Использование рассмотренных нами алгоритмов для компрессии оперативной памяти резко увеличивает время выполнения программы, приводя к полной потере производительности последней. Самый простой выход в этом случае – не дробить оперативную память на мелкие фрагменты, а использовать один большой блок памяти для хранения данных. Примером ситуации, когда это бывает необходимо, может служить созданная нами программа проигрывания звуковых файлов. Параметром этой программы является имя достаточно большого (около 700 Кб) звукового файла, не помещающегося в ОЗУ. Однако количество обращений к диску (а, следовательно, и перерывов в работе программы) мы можем минимизировать, распределив всю доступную оперативную память как один блок под данные из файла, прочитать туда столько байт, сколько помещается, и воспроизвести; затем прочитать следующий такой большой фрагмент и т.п. Проблема заключается в реализации этой идеи – то, что легко сделать средствами языка Си, невозможно сделать стандартными библиотечными функциями языка Паскаль. Эта проблема была решена путём обращения непосредственно к функциям ДОС для работы с памятью, которые позволили освободить оперативную память, занятую Паскаль-программой, и распределить её как единый блок.

Во многих школах, оснащённых устаревшей техникой, в качестве видеоадаптера, как правило, выступает отечественный CGA-монитор МС-6105. Путём небольших изменений в аппа-

ратной части и соответствующего программного обеспечения этот монитор, тем не менее, может служить при наличии VGA-платы VGA-монитором. Однако приемлемое изображение этот монитор начинает отображать только после загрузки соответствующего драйвера, меняющего значения сигналов кадровой и строчной развёртки. Всё, что происходит ДО загрузки драйвера, остаётся вне пределов зрения пользователя машины с такой конфигурацией, а среди всего прочего это – возможность входа в программу установки параметров базовой системы ввода-вывода. Это приводит к необходимости решения проблемы чтения и записи данных в энергонезависимой памяти, хранящей настройки системы (CMOS). Зная структуру энергонезависимой памяти, мы можем осмысленно менять те её значения, которые отвечают за необходимые нам параметрами – дату, время, тип диска и т.п. – и всё это – программно. Наличие программ чтения и записи в энергонезависимую память, структура которой детально разбирается в предлагаемом, позволяет, кроме того, сохранять критически важные настройки BIOS в файле с возможностью их последующего восстановления в случае утраты, очищать энергонезависимую память с целью снятия забытых паролей и т.п. (некоторые вирусы, к примеру, могут хранить в энергонезависимой памяти свои служебные данные).

Информация в компьютере не только хранится и обрабатывается, но и подготавливается к визуализации графическими и текстовыми редакторами. Это, однако, не означает, что нельзя распечатать данные, порождаемые в программах – коды стандартного матричного принтера семейства Epson, формирующие самые разнообразные шрифты, достаточно известны. Гораздо более универсальным и эффективным способом получения твёрдой копии является ГРАФИЧЕСКАЯ ПЕЧАТЬ, и именно ей посвящена последняя часть рассматриваемого курса. В качестве демонстрации графической печати можно воспользоваться программой моделирования случайных трёхмерных поверхностей, написанной учеником Центрально-Городской гимназии (ныне студентом физико-математического факультета КГПУ) И. Каплуном. Поверхность, построенная в программе, преобразуется в градации серого, а затем, используя их, выводится на печать.