сконструировать новый или изменить существующий сценарий обучения и контроля, выбрав иную последовательность подачи учебных кадров на экран, не изменяя при этом общей конфигурации системы.

Гипертекстовый способ представления информации, используемый в АОС АРГУС, является мощным средством организации адаптивного процесса обучения, создания гибких, легко настраиваемых на психофизические характеристики обучаемого пользователя системы, обучающих программ.

АЛГОРИТМЫ + ... = БЛОК-СХЕМЫ

Т.И. Михеева, С.И. Парфенов

Современное общество все в большей степени становится "информационным обществом", а его информационный аспект - одним из самых важных аспектов. Более 10 лет назад с легкой руки академика А.П.Ершова - "крестного отца" школьной информатики началось активное внедрение информатики, как общеобразовательного предмета, в школы и вузы России. В основе лозунга, выдвинутого А.П.Ершовым. "Программирование - вторая грамотность", лежит, во-первых, развитие алгоритмического мышления, во-вторых, освоение компьютера, приобретение навыков работы с вычислительной техникой.

Быстрое развитие технической базы настоятельно требует активной разработки учебно-методического обеспечения: компьютерных обучающих и тестирующих программ, тренажеров. При этом обучающие программы не должны являться эксклюзивной формой обучения, исключающей общение ученика и учителя, они должны быть иллюстративной поддержкой традиционных видов учебного процесса: лекций, практических и лабораторных занятий, контролируемых самостоятельных занятий обучаемых.

В процессе изучения информатики - науки о создании компьютерно-ориентированных математических моделей процессов любой природы - обучаемого необходимо познакомить с этапами решения задачи на ЭВМ. Создание моделей в конкретных предметных областях - этап, обязательно предшествующий решению задач, которые и возникают в рамках таких моделей. При этом процесс обучения приобретает характер рекурсии, появляется возможность определять

новые понятия и категории в терминах уже определенных, и тем самым достигается "спуск" с уровня общих абстрактных определений до уровня детальных знаний и представлений, связанных с организацией интерпретирующей среды.

Процесс построения математических моделей для различных видов задач - отдельная тема для обсуждения. При построении математических моделей широко применяются знания, полученные при изучении других предметов, не относящихся к информатике, как то: математика, физика, химия, экономика и т.д., в зависимости от специфики решаемой задачи. На уроках информатики необходимо ознакомить ученика с основами построения моделей, с описанием свойств объекта с помощью чисел и математических отношений, пригодных для обработки на ЭВМ. Но все же данный этап решения задачи ближе не к информатике, а к той области науки, куда относится решаемая задача.

Когда модель построена, необходимо приступать к следующему этапу решения задачи - составлению алгоритма. Что такое алгоритм, как он функционирует, каковы его свойства - все эти понятия должны быть изучены на уроках информатики.

Понятие алгоритма в информатике является фундаментальным. В школьном учебнике информатики он описывается как "организованная последовательность действий". Если представить алгоритм в виде временного ряда операций, то основной вопрос состоит в том, как управляется этот поток операций, как компьютер "узнает", какой оператор надо выполнить следующим, после того, как исполнение программы дошло до определенного оператора?

Установлено, что для описания любого алгоритма достаточно лишь трех принципов управления.

Первый принцип - это принцип последовательности. Он столь очевиден, что зачастую остается незамеченным.

Вторым принципом является условное исполнение или ветвление, когда в зависимости от выполнения или невыполнения некоторого условия совершается либо одна, либо другая последовательность действий.

Третий принцип составляет повторное исполнение или цикл, т.е. такая форма организации действий, при которой одна и та же последовательность действий совершается несколько раз (или ни разу) до тех пор, пока выполнятся некоторое условие

Одним из наиболее важных элементов программирования является концепция процедуры. Это главный механизм абстракции, процесса перехода от частных примеров к обобщенным понятиям

Для того, чтобы более наглядно представлять те или иные формы организации действий (алгоритм), можно воспользоваться блок-схемой, описывающей данный алгоритм. Блок-схема позволяет наглядно проследить пути выполнения алгоритма по ветвям условия или в цикле, определить места вызова подалгоритмов (процедур или функций), а также получить представление об алгоритме в целом, о его структуре и принципе работы. Каждое действие алгоритма при этом помещается в многоугольник, причем вид многоугольника определяется этим действием. В соответствии с ГОСТ 19.701-90 (ИСО 5807-85) определены следующие основные символы процесса:

- процесс обработки данных;
- предопределенный процесс (процедура/функция);
- процесс, выполняемый человеком;
- подготовка и модификация команды/группы команд с целью воздействия на нее последующей функцией;
- решение;
- цикл (для цикла вводится два символа: символ начала и символ конца цикла, причем в зависимости от вида цикла условие цикла может записываться или в начальном, или в конечном символе);
- передача управления процессу;
- вход/выход в/из части схемы,
- терминатор (начало и конец алгоритма);
- процесс ввода/вывода.

Последовательность действий при выполнении алгоритма указывается с помощью линий, соединяющих соответствующие символы. В случае движения по линии против привычного движения (справа налево или снизу вверх) линии снабжаются соответствующими стрелками, указывающими направление движения по схеме. Место соединения линий выделяется крупной точкой.

В школьном курсе информатики блок-схемам, к сожалению, уделяется мало внимания, больший упор делается на изучение какого-либо языка программирования. Это происходит по ряду причин.

Во-первых, блок-схемы более громоздки, чем запись соответствующего алгоритма на языке программирования, их труднее рисовать (кроме самого действия надо начертить соответствующий блок и соединяющие линии). И во-вторых блок-схема является неким изображением алгоритма, в то время как текст на алгоритмическом языке - уже конкретной его реализацией, которую можно использовать непосредственно на ЭВМ.

У такого подхода есть свои недостатки. При овладевании учеником компьютерной грамотностью важнейшим является освоение понятия алгоритма, приобретение навыков постепенной формализации алгоритма. Изучение какого-либо алгоритмического языка, при всей своей важности, на данном этапе несущественно, важно показать, что последовательность действий не меняется при изменении языка, описывающего эту последовательность. Языков программирования и их диалектов огромное множество, у каждого языка своя грамматика и синтаксис, своя манера выражения понятий. В принципе, большинство вычислительных задач можно решить, пользуясь любым языком, хотя программы при этом выглядели бы совсем по-разному. Кроме того, для какой-то конкретной задачи написать программу на одном языке бывает гораздо проще, чем на другом. Поэтому надо дать ученику представление о всем многообразии языков программирования, не углубляясь в изучение какого-то конкретного, причем делая упор на то, что суть алгоритма данной задачи остается неизменной при использовании разных языков программирования.

У современных компьютеров появляются новые возможности, повышающие удобство работы и производительность труда. Широко внедряется принцип функциональных клавиш, позволяющих до минимума сократить работу с клавиатурой. Нажатие одной лишь клавиши вызывает выполнение целой конструкции. Применение системы меню позволяет освободить пользователя от запоминания синтаксиса команд. Широчайшие графические возможности ЭВМ позволяют вообще избавиться от медленного и неудобного для большинства пользователей процесса работы с клавиатурой, а использовать графическое изображение операции, выбираемое с помощью манипулятора "мышь". С помощью простых картинок (пиктограмм) изображаются привычные предметы, такие, как папки, корзины для бумаг, калькуляторы, часы и т.д., действия, такие как "скопировать", "сохранить на диск", "удалить" и т.п. Путем указания на эти символы можно выбрать функции,

которые они обозначают. Такой интерфейс более удобен для большинства людей, поскольку когда все эти графические элементы правильно подобраны, они кажутся более естественными и простыми в изучении и использовании, требуют меньше памяти (человеческой) и вызывают меньшее число ошибок. Будущие программисты и просто пользователи ЭВМ сегодня сидят за школьной партой, и уже сегодня их следует приобщать к ЭВМ при помощи программных средств, использующих такой принцип взаимодействия человека и компьютера.

Одним из таких программных продуктов может стать "Электронный конструктор и анализатор схем алгоритмов", разработанный на кафедре информационных систем и технологий Самарского государственного аэрокосмического университета

Для чего же он нужен, спросите Вы?. Как следует из названия, он предназначен для конструирования (создания и редактирования) схем алгоритмов, то есть блок-схем. Его использование позволяет значительно упростить процесс работы над блок-схемой, а именно, он может использоваться в качестве удобного инструмента для рисования блок-схемы на экране компьютера, которая впоследствии может быть распечатана или использована как графическая иллюстрация какого-либо алгоритма. Пользователем "Электронного конструктора и анализатора схем алгоритмов" может быть, как ученик, так и учитель.

Построение блок-схемы осуществляется на виртуальном листе практически неограниченных размеров. Блок выбирается из набора символов (блоков-заготовок), соответствующих ГОСТ 19.701-90, который был описан выше.

Ввод блока осуществляется выбором соответствующего символа, заданием его местоположения на листе со схемой и последующим вводом текста операции в символ. Исправление блока заключается в нахождении места исправления, выборе требуемого действия (удаление блока / замена блока / изменение текста) и выполнении выбранного действия. После установки блок соединяется линиями с другими блоками для указания последовательности выполнения алгоритма.

На протяжении всей работы редактор проверяет допустимость использования символов и соединяющих линий, и не допускает ввода некорректной схемы алгоритма (например: неединственность блока "НАЧАЛО" и "КОНЕЦ", блок без текста соответствующей операции, более одной исходящей линии для простых

блоков и более двух - для блока ветвления, нигде не оканчивающиеся соединительные линии и т.п).

Используя операции "Сохранение" и "Загрузка" блок-схемы, можно осуществлять разделенное по времени и по месту редактирование.

Часто, при создании довольно большой схемы, она не помещается целиком на экран, тогда полезной может оказаться функция масштабирования, которая осуществляет масштабирование размеров блоков с целью полного расположения схемы на экране. При достаточно сильном уменьшении текст операции внутри символа может стать слишком мелким для чтения. Функцию масштабирования хорошо использовать в целях визуализации общей структуры алгоритма.

На протяжении всей работы с редактором пользователю доступна встроенная справочная система по функциям редактора и режимам работы, которая может быть активизирована в любой момент с помощью клавиши F1 или выбора соответствующего пункта меню. Кроме того, в нижней строке экрана всегда отображается информация о текущем режиме работы и краткая справка о возможных действиях.

Помимо непосредственно самого редактора схем алгоритмов, позволяющего создавать и редактировать блок-схемы, существует еще и программа-анализатор схем алгоритмов. Этот инструмент используется для обучения, и позволяет решать небольшие практические задачи, когда обучаемый достиг уровня полной формализации алгоритма. Анализатор осуществляет проверку корректности составленной блок-схемы и ее логического соответствия тексту алгоритма на одном из алгоритмических языков. По завершении анализа ученику выдается сообщение о результатах проверки: выдается список обнаруженных в блок-схеме ошибок, с указанием их местоположения на схеме и типа ошибки, или сообщается о правильности составления блок-схемы. При необходимости, блок-схема может быть исправлена и вновь представлена для проверки.

А теперь, внимание, фокус! В принципе, можно проверить соответствие блок-схемы алгоритму *НА ЛЮБОМ АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ*, и даже, может быть, на русском. Дело в том, что программа-анализатор не является встроенной программой, а является внешним выполняемым файлом, который вызывается редактором с передачей ей определенных параметров. Поэтому в качестве такой программы может вызываться любая, составленная с учетом принятых соглаше-

ний на формат данных для обмена между редактором и анализатором. Возможно даже создание своего банка программ анализаторов для разных языков или их реализаций. Эти программы могут быть написаны студентами младших курсов или одаренными школьниками в кружке информатики.

Как уже говорилось, описываемый конструктор схем алгоритмов может работать в двух режимах: режим ученика и режиме учителя.

Ученику предоставляются в распоряжение все вышеописанные функции. У учителя более богатые возможности. Загружая конструктор со специальным идентификационным ключем, учитель получает возможность для составления сценария занятия.

Учитель может:

• сформулировать и расположить на экране задание на урок (например: "Составить блок-схему алгоритма нахождения максимального нечетного элемента в массиве");

• загрузить и расположить на экране текст алгоритма (образец), по которому учеником впоследствии будет строиться блок-схема и с которым построенная схема затем будет сравниваться программой-анализатором;

• выбрать требуемую программу-анализатор (см. выше);

• загрузить некоторую блок-схему для ее дальнейшего редактирования учеником.

При использовании данного конструктора совместно с автоматизированной учебной системой АРГУС, учитель может сформировать учебный кадр, содержащий графическое изображение блок-схемы, пригодное для использования в качестве статической иллюстрации какого-нибудь алгоритма.

Широкие графические возможности, простота и интуитивная понятность интерфейса, обширная и легкодоступная справочная система данного конструктора позволяет легко работать с ним начинающим пользователям. Наглядность оформления и визуального изображения повышает интерес, удерживает внимание обучаемого в течение урока и способствует более глубокому усвоению материала.

ЛИТЕРАТУРА

1. Использование компьютерных технологий в обучении: Сб.науч.тр./АН УССР. Ин - т кибернетики им.В.М.Глушкова,Науч.совет АН СССР по пробл "Кибернетика".-Киев, 1990.

2 Михеева Т.И. Обучение программированию с визуализацией алгоритмов / Роль ВУЗов в формировании творческой интеллигенции на этапе экономических реформ // Тезисы докладов научно-практ. конф. -Пенза: Приволжский дом знаний. 1995.

3 Михеева Т.И. Автоматизированное обучение программированию на основе использования информационных процессов рекурсивного типа/Математика Компьютер. Образование //Труды II международной конф. - Пущино: МГУ,- 1995.

4 Михеева Т.И., Парфенов С. И . Коцюбинский Д С. Обучение алгоритмизации с использованием конструктора схем алгоритмов / Тезисы конф. - Самара: СГАУ, 1995.

5. Рынгач В.Д., Краснов М.А., Лысиков А.Ф., Мазурук А.А. Проектирование АОС для обучения алгоритмизации задач и программирования // Исследование и применение АОС в учебном процессе Тематический сборник научных трудов М 1985.

ГИПЕРТЕКСТ КАК СПОСОБ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В АРГУС

Т.И. Михеева, С.Ю. Курилкин, С.В. Михеев

В последнее время среди большого разнообразия программных средств и информационных систем учебного назначения все большее внимание привлекают системы, имеющие гипертекстовую структуру организации информации, гиперсреды. Возможности, открываемые ими, весьма привлекательны, поэтому с подобными системами связывают надежды на качественный скачок в технологиях обучения.

Компьютерные гипертекстовые технологии основываются на графовом способе представления и обработки неструктурированной информации и являются,