

От редакции. Несмотря на довольно обширную литературу по программируемым микрокалькуляторам, выпущенную к настоящему времени, редакционная почта свидетельствует о значительном интересе к этому виду ВТ в учительской среде. Он объясняется не только недостаточными тиражами изданий по этой тематике, которые, как утверждают авторы писем, практически не доходят до «глубинки», но и тем, что подобная литература не носит методического характера. Начиная публикацию статей, отражающих один из подходов к построению уроков с ПМК, мы надеемся, что они будут полезны учителям школ, преподавателям техникумов и профтехучилищ.

Л. ШТЕРНБЕРГ,
канд. физ.-мат. наук

Уроки

с программируемыми микрокалькуляторами

82

Наиболее распространенным средством вычислительной техники, которым оснащены многие школы, является программируемый микрокалькулятор. Его вычислительные возможности невелики, но вполне достаточны для решения задач, возникающих в школьном курсе.

Рассмотрим методику проведения уроков с программируемыми микрокалькуляторами. Для школ лучше всего подходят модели БЗ-34, МК-54, МК-56, имеющие одинаковую систему команд, которая более проста и логична, чем у моделей БЗ-21, МК-46. Впрочем, методика проведения урока сравнительно мало зависит от типа калькулятора. Мы будем использовать названия клавиш модели МК-56 и МК-54 (у БЗ-34 несколько клавиш отличаются по надписям на них).

Для проведения урока необходимо иметь не менее одного калькулятора на парту (т. е. на 2 человека) и один — для учителя. Для надежного проведения урока желательно работать с машинками, включенными в сеть.

Возможно два варианта проведения уроков. Первый — по мере изучения алгоритмического языка: изучили линейные программы — изучаем их реализацию на калькуляторе, изучили ветвления — рассматриваем, как это сделать на калькуляторе, и т. д. Но можно принять и другой вариант: сначала изучаем алгоритмический язык, затем проводим уроки по программированию рассмотренных алгоритмов на калькуляторе.

Тема «Структура ЭВМ на примере программируемого микрокалькулятора»

Эта тема занимает два урока. Желательно иметь плакаты «Лицевая панель калькулятора» и «Внутреннее устройство калькулятора» (с минимальным количеством деталей: операционные и адресуемые

регистры, программная память, счетчик адреса, клавиатура, индикатор, процессор в виде блока без детализации).



Первый урок. В течение 10 мин, используя плакаты, надо описать основные блоки микрокалькулятора: процессор (мы его не видим — он внутри), память для программы и данных, устройство ввода (клавиатура), устройство вывода (индикатор). Надо отметить, что исполнитель «Калькулятор» понимает «язык», состоящий из нажатий клавиш.

Далее целесообразно сразу же переходить к работе на калькуляторах. Поскольку в школе нет возможности делить класс для этих занятий на подгруппы, то учитель вынужден проводить уроки сразу с 25—30 учениками. Если сначала объяснить им, как работать на калькуляторах, а затем предложить работать самостоятельно, то учитель не будет успевать отвечать на возникающие вопросы. Одним из возможных вариантов является проведение первых уроков с калькуляторами в «командном режиме», когда весь класс синхронно по команде учителя выполняет одинаковые действия. Учитель параллельно объясняет, что при этом происходит. Если у кого-то из-за ошибочного действия получается не то, что

надо, учитель не отвлекается на частные ошибки (не всегда можно быстро понять, что именно неправильно сделал ученик, тем более что не всякую ошибку можно быстро нейтрализовать), а предлагает перейти к работе на одном калькуляторе с соседом (при работе парами на одном калькуляторе пара распределяется по одному человеку к двум другим парам). Опыт показывает, что за урок происходит порядка 5 таких случаев.

Итак, учитель объясняет, что клавиши надо нажимать только по его команде. На вопрос: «У всех на индикаторе то-то?» — должен следовать *молчаливый* кивок ученика, если все в порядке, или же ученик должен поднять руку (и то, и другое легко фиксируется учителем). На первом уроке достаточно объяснить работу в режиме вычислений с использованием только регистров X и Y (без работы со стеком) и адресуемых регистров. При этом учитель может произнести монолог, примерно такого содержания:

— В верхнем левом углу панели вы видите рычажок включения. Включить калькулятор! На индикаторе загорается ноль. У всех есть эта индикация? (Взгляд на класс.) В микрокалькулятор числа вводятся последовательным нажатием клавиш: введем число 25 — нажмем «2», на индикаторе «2», нажимаем «5» — на индикаторе «25». (Далее следует вопрос: «У всех получилось?» — в тексте опускается, но звучать он должен постоянно.) Число 25 записано в регистре X. Клавишей «V†» число 25 засылается в регистр Y и остается в регистре X — содержимое Y мы не видим. Нажмем «3», на индикаторе — «3», теперь в регистре Y «25», а в регистре X — «3». Можем в этом убедиться: клавиша «↔» (целесообразно параллельно показывать эти клавиши на плакатах) обменивает содержимое регистров X и Y. Нажимаем ее и на индикаторе — «25».

Теперь выполним, например, сложение: нажмем «+» — на индикаторе «28». Это в X оказалась сумма X и Y. Прочие двуместные операции выполняются аналогично.

Далее можно предложить тем, кто перешел к калькуляторам соседа, вернуться к своим машинкам — это удобный момент для возврата. Учитель объясняет назначение переключателя «радианы — градусы», одностепенные операции, ввод дробных и отрицательных чисел, чисел с порядком (эти вопросы трудностей не вызывают: с ними ученики знакомы по инженерным калькуляторам), а также исправление ошибочно набранного числа.

Затем переходим к работе с адресуемыми регистрами.

— Наберем, например, число 3 — нажмем «3» — оно загорелось на индикаторе. Теперь нажмем клавишу «x→P» — число на индикаторе мигнуло — и клавишу с номером регистра, например «0», — число на индикаторе еще раз мигнуло — теперь оно записано в регистр 0 и осталось в регистре X...

Аналогично заносим число 4 в P1, 5 — в P2, затем вызываем их опять в регистр X, иллюстрируя работу команд «P→x» и «x→P».

— Теперь проведем небольшой расчет. Например, зная длины сторон треугольника вычислим косинус угла, лежащего против стороны, указанной первой. (Нужно провести расчет по одной из линейных программ из предыдущих уроков.) Допустим, длины сторон a, b, c равны соответственно 3, 4, 5. При этом число 3 находится в P0, 4 в P1, 5 в P2.

Расчет проводим по формуле:

$$\cos A = (b^2 + c^2 - a^2) / (2 \cdot b \cdot c)$$

— Нажмем «P→x» «1» — на индикаторе «4» — это на регистр X вызвано значение b . Возводим его в квадрат: «F» « x^2 » — на индикаторе «16». Вызываем в X значение c «P→x» «2» — на индикаторе «5»; калькулятор устроен так, что при вызове нового числа или его наборе результат предыдущей операции автоматически уходит в регистр Y. Сейчас у нас в X значение c , а в Y — значение b^2 . Возводим c в квадрат...

В таком стиле объясняем дальнейшие вычисления. Поскольку пока пользуемся регистрами X и Y, то значение числителя придется записать в адресуемый регистр, а затем вызвать из него. Для объяснения происходящего удобно иметь таблицу следующего вида:

Клавиши	Индикация	Комментарий
P→x 1	4	$b \rightarrow X$
F x^2	16	$X^2 \rightarrow X$; теперь $X = b^2$
P→x 2	5	$X \rightarrow Y$; $c \rightarrow X$; $X = c$; $Y = b^2$

Желательно иметь такую таблицу в виде плаката (наилучший вариант — в виде раздаточного материала), но можно рисовать и на доске.

В качестве домашнего задания можно предложить сделать аналогичную таблицу для одной из задач (например, решение квадратного уравнения при заведомо неотрицательном дискриминанте). Таблица должна отражать последовательность нажатий клавиш, ожидаемые на индикаторе результаты для каких-то простых данных, комментарий происходящих действий.

Второй урок. Тема этого урока — запись программы в память калькулятора и ее

выполнение. На уроке необходимо иметь плакаты: лицевая панель калькулятора, таблица кодов команд калькулятора (ее хорошо иметь и в виде раздаточного материала), таблица «клавиши — индикация — комментарий» из предыдущего урока, дополненная слева столбцами «адреса» и «коды» (на первом уроке эти столбцы можно закрыть или просто не рассматривать).

Этот урок целесообразно провести в «командном» режиме с контролем «у всех получилось?» на каждом шаге и описанием расположения новых клавиш. Для этого удобно использовать ту же задачу, которая была просчитана в режиме вычислений на прошлом уроке. Для напоминания техники работы в режиме вычислений занесем значения 3, 4, 5 в P0, P1, P2 соответственно — они нам пригодятся. Дальнейший рассказ учителя может выглядеть, например, так:

— Переводим калькулятор в режим программирования: нажмем «F» и «ПРГ» — на индикаторе видим два нуля на месте порядка. Теперь калькулятор будет не исполнять подаваемые нажатием клавиш команды, а запоминать их в программной памяти в закодированном виде. Коды команд вы видите в таблице (указание на плакат). На индикаторе справа вы видите счетчик адреса: он показывает, что очередной код будет запомнен в ячейке с адресом ноль. Теперь будем нажимать те же клавиши в том же порядке (этот момент надо подчеркнуть). Нажимаем «П→х» «1» — на индикаторе появился код «61» — по таблице видим, что это код команды «П→х» «1», он запомнен в ячейке 00, а в счетчике адреса горит 01: следующая команда будет запомнена в ячейке 01...

Таким образом вводим всю программу, добавив в конце «С/П», назначение которой объясняется позже. В процессе ввода необходимо сделать 4 ошибки, чтобы пока-

зать, как их исправлять. Первая исправляется сразу:

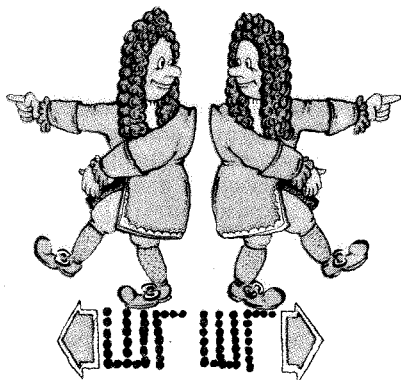
— Теперь давайте ошибемся: допустим при вводе команды «П→х» «0» мы плохо нажали клавишу «П→х» — вы ее не нажимайте вообще — нажимаем «0» — вместо кода 60 появился код 00...

Исправляем эту ошибку с помощью клавиши «ШГ». Следующую ошибку «замечаем» не сразу, а после набора еще двух команд и показываем ее исправление с помощью клавиши «ШГ» с пропуском уже набранных после нее команд клавишей «ШГ». Тем самым показывается, как клавишами «ШГ» и «ШГ» двигается «окошко», сквозь которое видны три ячейки памяти. Еще две ошибки «оставляем незамеченными».

Далее выполняется проверка программы. Нажимаем клавиши «F» «АВТ», «В/О», «F» «ПРГ» и читаем программу с помощью клавиши «ШГ». На этом этапе надо заметить и исправить одну из сделанных ошибок (вторую пока «не замечаем»).

Теперь выполняем программу по шагам: «F» «АВТ», «В/О» и многократно нажимаем клавишу «ПП». На этом этапе следует обратить внимание на то, что на каждом шаге выполняется одна очередная команда из памяти. Называем эту команду и показываем, что на индикаторе получается то же самое, что было при работе в режиме вычислений. На этом этапе у нас проявится последняя ошибка: для ее обнаружения надо после появления на индикаторе ошибочных показаний нажать «F» «ПРГ», и в левом углу индикатора мы увидим код ошибочной команды, т. е. команды, вызвавшей неправильный результат. Исправление программы выполняется обычным способом, после чего программа проверяется сначала.

После проверки программы можно ввести понятие автоматического режима работы калькулятора, когда он выполняет команды одну за другой, не дожидаясь нашей команды (нажатия клавиши). Здесь объясняется назначение команды «С/П» в программе: выход из автоматического режима и клавиши «С/П» — запуск работы в автоматическом режиме. Подготовив программу к работе (занеся ее начальный адрес в счетчик клавишей «В/О»), выполним программу, нажав «С/П». Здесь следует отметить, что подготовка программы, как мы видели, достаточно трудоемка, но теперь когда мы имеем готовую программу в калькуляторе, можно легко проводить вычисления для разных исходных данных. Здесь следует ввести в P0, P1, P2 другие значения длин сторон треугольника и получить результат для них. Таким образом следует просчитать несколько вариантов при



различных исходных данных.

Домашнее задание: при возможности организовать работу на калькуляторах во внеурочное время можно предложить провести всю работу по вводу и отладке программы из домашнего задания прошлого урока; если такой возможности нет, то выписать на листе бумаги всю последовательность нажатий клавиш при вводе программы (без умышленного введения ошибок) и показания индикатора.

Основные методические моменты. Наиболее целесообразно провести эти уроки сразу после введения понятия величины и рассмотрения первых алгоритмов работы с величинами, т. е. в начале 2-й четверти.

В заключение этих уроков полезно сделать обзор выполненной работы и отметить следующие моменты.

Калькулятор является **формальным исполнителем**: он «не понимает» смысла того, что считает, — здесь следует напомнить, как калькулятор «не замечал» ошибок, которые мы вносили в программу.

Л. ЗЕМЦОВА, А. ЛУКАНИН
НИИ школ МНО РСФСР

Форма или содержание?

Лабораторией обучения программированию и ЭВМ НИИ школ МНО РСФСР начиная с 1984/85 учебного года ведется систематическая работа по отработке методики преподавания машинного курса ОИВТ. В 1985—1986 г. совместно с Московским институтом электронной техники эксперимент проводился в школах Зеленоградского района Москвы. Экспериментальная программа курса с самого начала предусматривала максимальное использование имеющейся техники (ДВК-1 и ДВК-2), так как возможность реализации алгоритма на ЭВМ и получение результата работы машины повышает у учащихся мотивацию изучения информатики.

Для оценки правильности того пути, по которому шло экспериментальное обучение, и для своевременной коррекции содержания и методики преподавания курса в ходе эксперимента проводилось психолого-педагогическое исследование особенностей мотивации мышления и некоторых личностных характеристик учащихся. Нами использовались методики ранжирования, анкетирования, а также дидактические методики: «составление задач», «выбор», которые позволяют осуществлять диагностику психологических особенностей учащихся не в результате лабора-

Система команд исполнителя «Калькулятор» представлена в рассмотренной на уроке таблице кодов; «каждая его команда указывает... одно конкретное законченное действие, и исполнитель должен выполнить его целиком» (дискретность алгоритма), «выполнение всех команд гарантирует правильное решение задачи» (конечность алгоритма) [1].

Калькулятор с введенной программой позволяет решать множество однотипных задач (массовость алгоритма).

Таким образом, работа на калькуляторе позволяет хорошо проиллюстрировать и закрепить основные положения пройденной части курса.

Связь программ для ПМК с алгоритмами на алгоритмическом языке будет рассмотрена в следующих статьях.

Литература

1. Основы информатики и вычислительной техники. Ч. 1. М.: Просвещение, 1985.

Продолжение следует.

торного эксперимента, а в процессе выполнения учебных заданий (составление задач, выбор и решение выбранной задачи и т. д.).

При рассмотрении мотивации изучения информатики нами были выделены два основных типа мотивов — «операциональные» и «содержательные». Для того чтобы учение способствовало развитию мышления учащихся, необходимо, чтобы они стремились к овладению «схемами вещей», т. е. общими способами действий по решению классов задач. Этому соответствуют выделяемые нами среди широких познавательных мотивов учения «содержательные мотивы», показывающие степень развития мотивации «приобретения обобщенных способов».

«Содержательные» мотивы проявляются в интересе учащихся к решению задач и характеризуют ориентацию учащихся на овладение способами их решения. Однако в практике обучения нередко ЭВМ рассматривается как «игрушка», учащимся нравится «нажимать клавиши». Выделенные нами «операциональные» мотивы характеризуют интерес учащегося к технической стороне работы на ЭВМ.

Результаты первого опроса, проведенного в октябре 1985 г., показали, что у 10 % уча-

ли проложены по полу вдоль плинтуса от каждой машины к блоку А2, установленному в одном из ученических столов. Место установки блока выбирается с учетом наименьшего расхода кабеля и расположения сетевой розетки для подключения блока питания. Схема блока питания со стабилизацией на 5 В не приводится. Подобные схемы часто печатаются в журнале «Радио». Можно также блок питания заменить двумя параллельно соединенными батарейками типа «Планета» (4, 5В). Конечно, для удобства монтажа желательно использовать тонкий коаксиальный кабель. Соединение блоков между собой показано на рис. 3. В качестве линии из проводов 1, 2, 3, 4, 5, 6 использован 5-ти парный телефонный ка-

бель. Неиспользованные жилы его остаются в качестве резерва.

Вариант оформления панели пульта контроля — на рис. 4.

Если дисплейный класс расположен в помещении с небольшой площадью, то можно исключить микросхему DD6, а все устройство собрать в одном корпусе. При этом входы транзисторных ключей подсоединяются к выходам микросхемы DD5. При отсутствии светодиодов можно вместо них включить через транзисторные ключи лампочки для карманного фонарика.

Элементы схемы монтируются на печатных платах из одностороннего фольгированного стеклотекстолита.

В. ЗИНЮК, Э. БАРАНАУСКАС
г. Мурманск

ПМК на уроке труда

Последние два года в слесарной мастерской школы № 42 г. Мурманска с успехом используются микрокалькуляторы при выполнении различных практических работ. Первоначально использовались простейшие МК для вычислений при работе на токарных, фрезерных станках, разметке.

Для настройки токарных станков ТВ-4, ТВ-6 на точение заданного размера по диаметру используется формула

$$n = \frac{D-d}{2 \cdot 0,025} = 20(D-d),$$

где n — число делений лимба поперечной подачи, D — первоначальный диаметр заготовки, d — заданный диаметр детали, 0,025 — цена деления лимба.

При точении конических поверхностей тангенс угла поворота верхней части суппорта вычисляется по формуле

$$\operatorname{tg} \alpha = \frac{D-d}{2l},$$

где D — большой диаметр конуса, d — малый диаметр конуса, l — длина конуса. Затем по таблице тангенсов определяется величина угла.

На первом этапе использования МК учащиеся производят соответствующие вычисления, приобретают при этом навыки использования микропроцессорной техники для выполнения практических работ. Это сокращает время вычислений, повышает точность, качество выполняемых работ.

Расчеты упрощаются, если использовать программируемые МК.

Программа вычислений числа делений лимба для ПМК «Электроника-54» и «Электроника-56» такова: 00.— 01.2 02.0 03*04.С/П 05.БП 06.00. Конкретные данные для вычислений вводятся в следующем порядке: D† В d В/О С/П

При незначительном количестве ПМК в мастерской и разнообразии выполняемых учащимися работ целесообразно составлять сводные программы по наиболее часто используемым формулам. Вот одна из них.

	00	01	02	03	04	05	06	07	08	09
00	↑	1	—	X=0	12	С/П	—	2	0	*
10	БП	05	1	—	X=0	25	С/П	—	↔	÷
20	2	÷	tg ⁻¹	БП	16		—	X=0	36	С/П
30	—	4	0	*	БП	29	1	—	X=0	51
40	С/П	*	π	*	1	0	0	0	÷	БП
50	40	С/П	↑	1	8	0	↔	÷	sin	*
60	БП	51								

Программа вводится в ПМК перед занятием и позволяет производить расчеты по одной из пяти формул.

1. $n = 20(D-d)$.
2. $\alpha = \arctg((D-d)/2l)$.
3. $n = 40(L-l)$: число делений лимба продольной подачи суппорта при подрезании детали до заданной длины, L — первоначальная длина заготовки, l — длина детали по чертежу.

4. $V = \frac{\pi D n}{1000}$ м/мин: скорость резания при

цилиндрическом точении, D — диаметр заготовки, m , n — число оборотов шпинделя, об/мин.

5. $H = D \sin \frac{\pi}{n}$: длина хорды n -угольника, вписанного в окружность диаметра D .

Для выбора любой из приведенных формул необходимо набрать ее номер:
№ формулы В/О С/П

После этого можно вводить данные в соответствии с образцами:

1. $D \uparrow d$ С/П
2. $L \uparrow D \uparrow d$ С/П
3. $L \uparrow l$ С/П
4. $D \uparrow n$ С/П
5. $D \uparrow n$ С/П

После получения ответа можно снова ввести данные, и калькулятор выполнит вычисления по выбранной вами формуле.

Картинка — на принтер

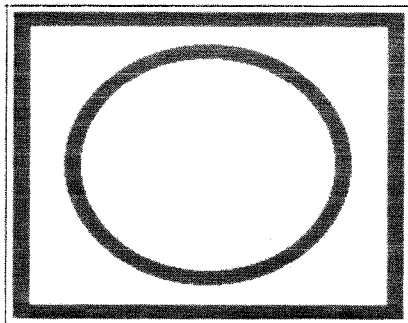
В секции прикладной математики и вычислительной техники электростальского филиала Московского института стали и сплавов найдена возможность вывода на принтер преподавательской ЭВМ машинной графики компьютеров «Электроника БК-0010Ш», работающих на Бейсике.

В журнале «Информатика и образование» (1988, № 5, с. 58) уже сообщалось о возможности печати листингов программ и алфавитно-цифровых результатов счета ЭВМ БК-0010Ш без обращения к накопителю на гибких магнитных дисках ЭВМ ДВК-2М*.

Идея печати машинной графики БК-0010Ш заключается в копировании на принтер картинки с экрана дисплея. Принтер подачей управляющих кодов может быть переведен в растровый (графический) режим работы. Определенное число поступающих затем кодов обрабатывается им следующим образом: срабатывают иглы печатающей головки принтера, соответствующие «установленным» битам в двоичном представлении анализируемого кода. Машинной графике Бейсик-системы доступна область экранного ОЗУ с адреса 17408 по 32767 размером 240 на 512 точек. Следовательно, при копировании построчно картинки с экрана дисплея на принтер надо обработать в растровом режиме 512 кодов, составленных так, чтобы иглы принтера срабатывали в необходимых местах. Подача с БК-0010Ш на принтер нулевого кода затруднительна, но его можно заменить кодом 128, настроив принтер на обработку только семибитной информации. Количество строк при построчном копировании картинки может быть равно $240:6=40$, но тогда картин-

ка получится мелкой. Крупная картинка получается при числе строк, равном 80, где каждой копируемой точке картинки соответствуют два соседних бита в двоичном представлении обрабатываемого принтером кода.

Приведем Бейсик-программу (авторы — И. А. Буров и А. С. Разумов), позволяющую получить на принтере ROBOTRON CM 6329.02-М крупную копию картинки с экрана дисплея. В ней строки 110—190 — построение картинки (в данном случае — окружности), строки 200—400 — вывод картинки из экранного ОЗУ на принтер. На принтер с БК-0010Ш необходимо передать $519 \times 80 + 5 = 41\,525$ кодов, включая управляющие коды установки интервала подачи бумаги, перевода принтера в растровый режим, а также его инициализации. Для промежуточного хранения такого объема информации не хватает буфера ДВК-2М. Поэтому вся информация делится пополам и передается частями. Поскольку программа просматривает адресное пространство экранного ОЗУ последовательно, то его первый бит должен соответствовать точке экрана в левом верхнем углу. Режим экранного «ролика» нарушает это соответствие, но его можно восстановить «смаргиванием» экрана, дважды подав управляющий код 140. Эта процедура обязательна перед построением копируемой на принтер картинки. Общее время копирова-



* К сожалению, в эту заметку вкрались опечатки: в приведенной программе аргумент оператора OPEN — не TT:RL:, а TT:LP:; вместо «занятые и встроенные функции» следует читать «занятые и встроенные функции». — Примеч. ред.

Л. ШТЕРНБЕРГ,

канд. физ.-мат. наук

Куйбышевский авиационный институт

Уроки с программируемыми микрокалькуляторами

Перевод алгоритмов
на язык микрокалькулятора

После проведения уроков, на которых ученики знакомятся с возможностями программируемого калькулятора, у них возникает естественный вопрос: как связаны программирование в командах калькулятора и запись алгоритмов на алгоритмическом языке? Ответу на него и посвящаются уроки, темы которых можно сформулировать следующим образом:

1. Перевод линейных алгоритмов на язык калькулятора.

2. Перевод ветвлений.

3. Перевод циклов.

4. Отладка программ на калькуляторе.

Первые 3 урока проводятся без калькуляторов, а 4-й является фактически лабораторной работой на калькуляторах.

При изложении этого материала учитель должен подчеркнуть, что любая ЭВМ понимает весьма своеобразный язык машинных команд, несколько похожий на коды калькулятора. А программа на языке программирования после ее ввода в ЭВМ переводится на язык машинных команд и только затем выполняется. Процесс перевода называется трансляцией. Так как калькулятор — это очень маленькая ЭВМ, то перевод с алгоритмического языка приходится выполнять вручную. Правила этого перевода и составляют содержание трех уроков.

Отметим, что наличие любой вычислительной техники (калькуляторов в том числе) заставляет учителя отойти от плана, рекомендованного для безмашинного варианта курса.

Ряд уроков необходимо отвести для работы с калькуляторами (или ЭВМ), поэтому часть материала IX класса откладывается до X класса, где имеется большой резерв времени, так как материал учебника X класса рассчитан на 34 часа, а на курс отведено 68 часов. 75

Урок «Перевод линейных алгоритмов» лучше всего провести сразу после изучения структуры калькулятора, урок «Перевод ветвлений» — после того, как рассмотрены разветвляющиеся алгоритмы обработки величин, т. е. где-то в конце 2-й четверти, уроки «Перевод циклов» и «Отладка программ» — в середине 3-й четверти, после рассмотрения циклических алгоритмов, когда на калькуляторе можно будет просчитать практически полезные программы, такие, как вычисление площади криволинейной трапеции, вычисление многочлена, определение корня методом деления пополам и т. д.

Перед тем как начать изложение правил перевода, следует рассмотреть, какие элементы калькулятора могут служить аналогами элементов алгоритма. Переменная величина может принимать разные значения, на калькуляторе могут принимать разные значения регистры, значит, переменным алгоритма должны соответствовать регистры. Все регистры в арифметических операциях работают одинаково, значит, любой из них может быть взят для переменной типа цел и вещ, а переменные типа лит обеспечить регистром невозможно: литерные величины на калькуляторе не обрабатываются. Команде (предписанию) присваивания из алгоритма соответствует серия команд калькулятора, так как исполнитель Калькулятор понимает более мелкие команды, чем исполнитель, понимающий алгоритмический язык.

Правила перевода алгоритмов удобно рассказывать, параллельно иллюстрируя их на каком-либо алгоритме, а затем сформулировав в явном виде. Они сводятся к правилам перевода отдельных команд алгоритмического языка (чтобы не путаться, мы будем говорить «предписание» алгоритмического языка и «команда» калькулятора). Проиллюстрируем их на примере алгоритма вычисления косинуса угла треугольника по заданным длинам трех его сторон:

алг косинус угла (вещ a, b, c , кос),

арг a, b, c ; **рез** кос;

нач кос: $= (b^2 + c^2 - a^2) / (2 \cdot b \cdot c)$;

кон;

Правила перевода:

1. Заголовок алгоритма, предписания **арг**, **рез**, **нач** не переводятся.

2. По описаниям типов переменных (вещ, цел и т. д.), стоящим как в заголовке, так и после **нач**, составляется таблица, в которой каждой переменной ставится в соответствие адресуемый регистр. Регистры можно распределять произвольно, рекомендуется брать под-
ряд.

Для нашего алгоритма: $a - P0$; $b - P1$, $c - P2$; кос — P3.

3. Присваивания переводятся очевидным образом. Составленная таблица распределения регистров показывает, откуда брать значение и куда помещать результат. Если понадобятся регистры для запоминания промежуточных результатов, то их берут из числа свободных.

4. **кон** переводится командой «С/П».

Программы рекомендуется всегда составлять с нулевого адреса (даже не объясняя ученикам возможность их написания с других адресов) и записывать в виде таблицы «адрес — команда — код — алг. язык — комментарий», как это сделано ниже. Стобец «код» заполняется после написания программы.

Для выполнения программы надо занести значение аргумента в P0, P1, P2 соответственно и нажать клавиши «В/О» и «С/П». После остановки считать результаты из регистров, отведенных под результаты. В нашем случае, так как результат один и он вычисляется в самом конце, он при остановке окажется на индикаторе, но так будет не всегда.

Однако калькулятор может помочь нам в занесении аргументов и выдаче результатов: он может сам считать значения в память или выдавать их на индикатор. Это одно из основных действий калькулятора, которое нельзя выразить через другое, поэтому для описания

Адрес	Код	Команда	Алгоритмический язык
		$a - P0$	
		$b - P1$	
		$c - P2$	алг косинус угла
		кос — P3	(<u>вещ</u> a, b, c , кос);
00	61	П—x 1	кос:=
01	22	F x^2	b^2
02	62	П—x 2	+
03	22	F x^2	c^2
08	44	x—P 4	
09	02	2	/ (2
10	61	П—x 1	· b
14	41	П—x 4	
15	13	—	
16	23	F 1/x	
17	43	x—P 3	
18	50	С/П	кон

таких действий в язык надо ввести специальные средства: предписания **ввод** и **вывод**.

Предписание «**ввод** список переменных» — это запрос в процессе выполнения программы значений указанных переменных, а предписание «**вывод** список выражений» — это запрос в процессе выполнения программы. **Ввод** и **вывод** можно сочетать с аргументами и результатами.

Правила их перевода таковы:

1. Конструкция **ввод** программируется командами «x—П» с номерами регистров, соответствующими перечисленным после **ввод** переменным, перед каждой из которых, кроме первой, ставится «С/П»;

2. **Вывод** программируется командами вычисления соответствующих выражений, после каждого выражения, кроме последнего, ставится «С/П».

С применением этих конструкций алгоритм принимает вид:

алг косинус; (аргументов, результатов нет, все переменные — вспомогательные) (запрашиваем значения a, b, c)
нач вещ a, b, c ;
ввод a, b, c ;
вывод $(b^2 + c^2 - a^2) / (2 \cdot b \cdot c)$
кон

Программа выглядит так (распределение регистров прежнее):

У школьников может возникнуть вопрос: почему, нарушая правила формального перевода, можно получить более короткую или быструю программу? Ответ на этот вопрос таков: мы пользуемся очень простым алгорит-

Адрес	Код	Команда	Алгоритмический язык	Комментарий
00	40	x—П 0	ввод a	Запоминаем набранное перед пуском значение a .
01	50	С/П	b	При этом останове набирается b , после пуска оно запоминается.
02	41	x—П 1		
03	50	С/П	c	Этот останов нужен для набора c , значение c запоминается, далее начинается вычисление.
04	42	x—П 2		
05	61	П—x 1		
06	22	F x^2		
...		
23	50	С/П		Когда выражение вычислено, команда С/П позволяет увидеть его значение на индикаторе.

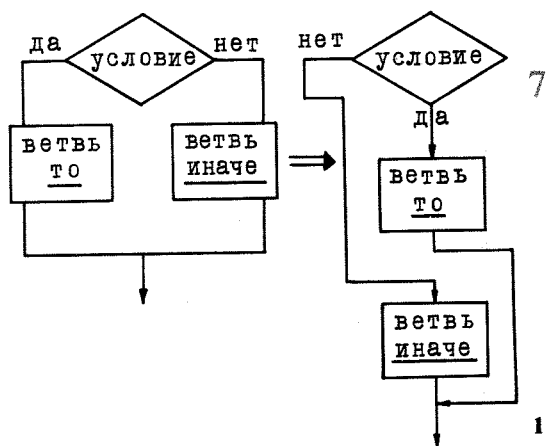
мическим языком, в котором нет ряда полезных возможностей. Если усложнить язык, то на нем можно будет описать все нюансы, дающие при переводе оптимальные программы.

Следующий «калькуляторный» урок проводится после изучения конструкции «если-то-иначе-все» и решения ряда задач с использованием этой конструкции применительно к алгоритмам работы с величинами.

Объяснение команд условного и безусловного переходов должно занимать около 10 мин. Здесь следует объяснить, что эти команды засылают в счетчик адреса значение, хранящееся во втором слове команды, подменяя то значение, которое было в счетчике адреса ранее. Тем самым происходит переход к выполнению команды с указанным адресом. Следует упомянуть, что команды условных переходов выполняются только в программном режиме и что в командах переходов нельзя исправить только адрес: надо повторно занести и команду.

Далее, перерисовав схему ветвления так, как показано на рис. 1, легко объяснить, что ромб, из которого имеется выход либо даль-

ше, либо в обход части программы, соответствует команде условного перехода (позволяющая идти либо дальше, либо в обход части программы), а ломаной стрелке, обходящей



часть «иначе», соответствует безусловный переход. Это объяснение позволяет сформулировать правила перевода ветвлений.



1. Программируется условие, приведенное к имеющемуся на калькуляторе сравнению с нулем (т. е., например, вместо $a \leq b$ рассматривается условие $b - a \geq 0$).
2. Пишется нужная команда условного перехода, за которой оставляется пустая ячейка для адреса, который выяснится позже.
3. Программируется ветвь «то».
4. Пишется команда «БП», и оставляется свободная ячейка для адреса, который также выяснится позже.
5. Программируется ветвь «иначе».
6. Когда встречается «все», пропущенный на шаге 2 адрес заполняется адресом начала ветви «иначе», а пропущенный на шаге 4 адрес — адресом очередной свободной ячейки.

Эта часть урока требует также около 10 мин. Затем правила перевода иллюстрируются на конкретном примере, например на классическом: выбор максимума из двух чисел. Пусть a находится в P_0 , b — в P_1 , максимум M размещается в P_2 . Тогда тело алгоритма и программы примет вид (фрагмент программы совершенно произвольно размещен с адреса 05):

«К НОП», заносятся исходные данные, и команда выполняется в пошаговом режиме 2 раза: при $a < b$ и при $a \geq b$. После выполнения команды 07 нажатием «F ПРГ» получаем возможность взглянуть на счетчик адреса: на индикаторе «11 61 60 08» — последние 3 команды и номер команды, которая будет выполняться. Возвращаемся в автономный режим («F АВТ»), выполняем одну команду («ПП»)

Адрес	Код	Команда	Алгоритмический язык	Комментарий
05	60	П—х 0	если	Программируем условие, приведенное к сравнению с нулем.
06	61	П—х 1	$a-b$	
07	11	—		Адрес выяснится позже.
08	59	$Fx \geq 0$	≥ 0	
09	□	□	то	
10	60	П—х 0	$M:=a$	
11	42	х—П 2		И этот адрес выяснится позже.
12	51	БП	иначе	
13	□	□		
14	61	П—х 1	$M:=b$	
15	42	х—П 2		Вот сейчас ячейку 09 можно заполнить адресом 14, а ячейку 13 — адресом 16.
16	все	

Обратите внимание учеников на строгое соответствие элементов алгоритмического языка и отчеркнутых частей программы. Полезно также рисовать стрелки переходов.

Если останется время (или же на следующем уроке), можно поработать с этой программой на калькуляторе. Для этого программа дополняется в ячейках 00—04 командами

и снова смотрим на счетчик адреса («F ПРГ»), где при $a < b$ видим «17 51 42 15», т. е. выполнен переход, а при $a \geq b$ видим «15 59 11 10», т. е. перехода не произошло.

Отметим, что этот фрагмент можно написать существенно короче, если пользоваться более развитым алгоритмическим языком.

В качестве домашнего задания можно предложить написать программу вычисления и максимума и минимума или значения функции, определяемой по разным формулам при аргументе больше нуля и меньше нуля соответственно.

На этих уроках не следует затрагивать вложенные ветвления, а также сложные условия с союзами и и или — этот материал более тяжел для понимания, а потому должен быть отложен на более позднее время, когда учащиеся лучше освоятся с калькуляторами.

Работа с циклами будет рассмотрена в следующей статье.

В завершение отметим, что статьи данного цикла предполагают, что учитель знаком с работой на ПМК, и потому в них рассматриваются только методические вопросы. Подробное изложение программирования на ПМК, техники перевода с алгоритмического языка на примерах задач школьного учебника дано в [1].

Продолжение следует.

Литература

1. Штернберг Л. Ф. Программирование на микрокалькуляторе. М.: Просвещение, 1988.

— УЧИТЕЛЬ ДЕЙСТВУЕТ
ВМЕСТЕ С КЛАССОМ...



Педагогический опыт

Л. ШТЕРНБЕРГ,

канд. физ.-мат. наук

Куйбышевский авиационный институт

Уроки с программируемыми микрокалькуляторами

Многие учителя информатики 1 сентября начнут учебный год вооруженными новой вычислительной техникой. Хотя число дисплейных классов растет, для многих школ они пока недоступны. На сегодняшний день наиболее распространенными остаются программируемые микрокалькуляторы. Как же рационально распланировать учителю время уроков с ПМК? Надеемся, что ему помогут статьи этого цикла.

В качестве одного из вариантов можно предложить такой.

IX класс

Уроки 1—6 проводятся так же, как и в безмашинном варианте. Даются основные понятия на примере простых исполнителей (с. 1—29 [1] или соответствующие разделы по [2]). Рекомендуется пропустить материал «Форма представления информации в ЭВМ» — он будет дан позже, так как на 2-м уроке он воспринимается плохо. В качестве простых исполнителей рекомендуется взять «Муравья» [4, 5], графический исполнитель (с. 77—88 [1]), исполнители «Путник» и «Резчик металла» [2].

Уроки 7—9. Понятие переменной величины, линейные алгоритмы работы с величинами (с. 29—34 [1] или соответствующий материал из [2]).

Уроки 10—14. Структура ЭВМ на примере ПМК. ПМК как исполнитель. Структура его команд. Работа в режиме вычислений, запись программ в память и их выполнение. Перевод линейных алгоритмов на «язык»

ПМК. Материал к этим урокам изложен в предыдущих статьях цикла. Технические детали подробно рассмотрены в [3] — этот материал может быть использован как учителями, так и учениками. Работа со стеком рассмотрена в [7].

Уроки 15—16.* Разветвляющиеся алгоритмы. Теоретический материал и решение задач по учебнику.

Уроки 17—18. Перевод ветвлений на язык ПМК. Методическая сторона этих уроков рассмотрена ранее, техническая изложена в [3].

Уроки 19—20. Контрольная работа и ее анализ.

Уроки 21—28. Циклы. В этой серии уроков пока не затрагиваются вложенные циклы и табличные величины. Методические вопросы рассмотрены в настоящей статье.

Уроки 29—30. Табличные величины (без обработки на ПМК).

Уроки 31—32. Контрольная работа и ее анализ.

Уроки 33—34. Экскурсия на вычислительный центр (эти уроки могут быть проведены не обязательно в конце — их можно дать в любое время).

Остальной материал дается в X классе. Как видим, пока что охвачен далеко не весь материал IX класса. Отметим изменение порядка подачи материала: в уроки 21—28 входит часть разделов «Решение задач из курса математики» и «Решение задач из курса физики» — тех, которые для решения требуют изучаемых технических приемов (циклов определенного вида).

Уроки 1—3. Повторение. Один из уроков отводится для повторения работы на ПМК.

Уроки 4—5*. Задачи на табличные величины. Материал разделов «Решение задач из курса математики» и «...физики», связанный с обработкой таблиц.

Уроки 6—8.** Косвенная адресация, обработка табличных величин на ПМК.

Уроки 9—13. Циклы с параметром. Решение задач с их применением; приведение их к виду, реализуемому на ПМК.

Уроки 14—17. Двумерные таблицы. Вложенные циклы. Реализация вложенных циклов на ПМК.

Уроки 18—19. Контрольная работа и ее анализ.

Уроки 20—22*. Вспомогательные алгоритмы. Параметры. Использование вспомогательных алгоритмов для создания библиотек.

Уроки 23—24.** Подпрограммы на ПМК. Реализация вспомогательных алгоритмов в виде подпрограмм.

Урок 25*. Основные этапы решения задач на ЭВМ [1].

Уроки 25—28. Алгоритмы вычисления значений функции и их реализация на ПМК.

Уроки 29—31. Команда выбора, ее применение в задачах. Реализацию на ПМК следует только упомянуть: программируются вложенные если, которым эквивалентен выбор.

Уроки 32—33. Контрольная работа и ее анализ.

Уроки 34—36*. Алгоритмы работы с литературными величинами.

Уроки 37—39. Алгоритмы поиска информации.

Уроки 40—47.** Устройство ЭВМ. Основной алгоритм работы процессора. Организация ветвлений и повторений на ЭВМ. Подпрограммы. Кодирование информации в памяти ЭВМ. Решение задач на составление программ для ЭВМ. (Материал в безмашинном варианте достаточно труден, но при наличии ПМК его можно рассказать быстро и понятно, проводя аналогии с ПМК.)

Уроки 48—50*. Логические элементы ЭВМ, переработка информации с их помощью. Физические принципы работы внешних устройств.

Уроки 51—52. Контрольная работа и ее анализ.

Уроки 53—60*. Язык программирования Рапира. Решение задач.

Уроки 61—66*. Роль ЭВМ в современном

обществе. Решение задач для повторения. Часть задач можно решить с использованием ПМК.

Уроки 67—68. Контрольная работа (итоговая) и ее анализ.

Обратите внимание, что на контрольные дается два урока. На 2-м уроке важно показать не только правильные решения, но и типичные ошибки, разъяснив, как исполнитель среагирует на ошибку: если ошибка синтаксическая, исполнитель не поймет, а если смысловая — понятное предписание неисполнимо либо дает не тот эффект, который нужен.

Для учителей, которые работали в IX классе по безмашинному варианту, а к X получили ПМК, можно рекомендовать план уроков в X классе: проводится один урок на повторение линейных алгоритмов, затем серия «калькуляторных» уроков на эту тему (уроки 10—14 для IX класса в описанном выше плане); один урок на повторение разветвляющихся алгоритмов, серия «калькуляторных» уроков по этой теме (уроки 17, 18 для IX класса) и т. д. При такой схеме уроков к концу полугодия будут пройдены все калькуляторные темы, далее можно идти по описанному плану для X класса.

Перейдем к рассмотрению методики проведения уроков 21—28 для IX класса. Тема «Циклы».

В учебнике [1] циклы вводятся сразу вместе с табличными величинами (§ 10) — это методический недостаток: вводятся сразу два новых понятия, поэтому материал воспринимается учениками хуже, чем хотелось бы. Рекомендуется начать циклы с наиболее простой и естественной конструкции (см. рис. 1а):

повторять М раз (или цикл М раз)
тело цикла

кц

С ее помощью гораздо проще записываются, например, алгоритмы степени (с. 50 [1]), площади (с. 64 [1]) и др.

Сравним тела алгоритмов:

цел i

y:=1; i:=1;

пока i≤N

нц y:=y·a;

кц

вещ h, x; нат i;

h:=(b-a)/n; S:=0;

x:=a;

пока i≤n

нц S:=S+f(x)·h;

x:=x+h; i:=i+1

кц

y:=1;

цикл N раз

y:=y·a

кц

вещ h, x;

h:=(b-a)/n; S:=0;

x:=a;

цикл n раз

S:=S+f(x)·h

x:=x+h;

кц

** Две звездочки означают, что методика изложения этих уроков будет рассмотрена в следующих статьях данного цикла. — Прим. автора.

Проведя два урока по изучению конструкции цикла и решению задач (можно рассмотреть задачу колебания шарика на пружине или вычисление значений какой-либо функции в равноотстоящих точках), проводим урок по реализации этой конструкции на ПМК. Объяснение работы команд цикла FL 0, FL 1, FL 2 и FL 3 (общее название — FL*i*) займет порядка 10 мин, так как они во многом похожи на ранее изученные команды переходов. Далее формулируются строгие правила программирования таких циклов.

1. Если в алгоритме имеется цикл типа раз, распределение памяти начинается с отведения регистра из P0 ÷ P3 для счетчика. (впоследствии, при изучении вложенных циклов, следует упомянуть, что для вложенных циклов такого типа потребуется несколько счетчиков.)

2. Цифра в цикл... раз программируется командами вычисления количества повторений с записью результата в регистр, отведенный для счетчика.

3. Тело цикла программируется как обычно.

4. кц для цикла типа раз программируется командой "FL*i*", где *i* — номер отведенного данному циклу регистра-счетчика, с адресом начала тела цикла.

Правила иллюстрируются на примере какого-либо алгоритма, например алгоритма *степень* (он очень удобен для пошагового

исполнения). Программа для ПМК записывается в столбец с параллельной записью алгоритма и отчеркиванием групп команд, соответствующих отдельным конструкциям алгоритмического языка. Пример оформления приведен в табл. 1. Еще раз напомним, что система записи — один из важных методических моментов.

Сразу отметим, что возможны вопросы вундеркиндов, связанные с тем, что программе можно значительно сократить, если разместить *n* там же, где счетчик, т. е. в P0, а также накапливать результат непосредственно в PX, без хранения в P3. Это действительно так, и такую оптимальную программу можно получить, если использовать более сложные правила распределения регистров (с. 66, 70 [3]). Их можно рассмотреть на кружках или в математических классах, а на уроках в обычных классах ограничиться наиболее простыми правилами, заботясь о простоте понимания, а не об оптимальности программы.

Мотивацией для написания этого алгоритма на ПМК является то, что вычисление a^n с помощью операции "F x^y" дает большую погрешность (вычислите 2^3 по программе и операцией "F x^y"). Для работы с этой программой надо заслат в память значения аргументов: *a* — в P1, *n* — в P2; после выполнения прочитать результат из P3 (в силу особенностей программы результат

Таблица 1

Адрес	Код	Команда	А — язык	Комментарий
счетчик — P0 <i>a</i> — P1 <i>n</i> — P2 <i>y</i> — P3			<u>алг</u> <i>степень</i> (<u>вещ</u> <i>a</i> , <u>нат</u> <i>n</i> , <u>вещ</u> <i>y</i>);	Сначала отводим регистр счетчику, затем всем остальным переменным — аргументам и результатам (внутренних переменных, описываемых после <u>нач</u> , в данном алгоритме нет).
			<u>нач</u>	
00 01	01 43	1 <i>x</i> — П 3	<i>y</i> := 1	Присваивание программируется обычно.
02 03	62 40	П — <i>x</i> 2 <i>x</i> — П 0	<u>цикл</u> <i>n</i> <u>раз</u>	Количество повторений цикла засылается в счетчик.
04 05 06 07	63 61 12 43	П — <i>x</i> 3 П — <i>x</i> 1 × <i>x</i> — П 3	<i>y</i> := <i>y</i> · <i>a</i> ;	Тело цикла (в данном случае состоящее из одного присваивания) программируется обычно.
08 09	5Г 04	FL 0 04	<u>кц</u>	
10	50	С/П	<u>кон</u>	<u>кц</u> программируется в соответствии с п. 4 правил: тело цикла начинается с адреса 04, счетчик — в P0.

Адрес	Команда	А — язык	Комментарий
		<u>алг</u> время прибытия (вещ S, h, a, T); <u>арг</u> S, h, a <u>рез</u> T <u>нач</u> вещ <u>путь</u>	Регистры распределены по порядку: $S — P0, h — P1$ и т. д. Здесь $S —$ заданный путь, $h —$ шаг по времени, $a —$ время старта, $T —$ время прибытия, <u>путь</u> — пройденный путь с момента a .
00 01 02 03	0 $x \leftarrow P4$ $P \leftarrow x2$ $x \leftarrow P3$	путь := 0; $T := a$	Присваивания программируются стандартно.
04 05 06	$\rightarrow P \leftarrow x4$ $P \leftarrow x0$ —	<u>пока</u> $S < \text{путь}$	Программируем цикл: вычисляем условие, приведенное к сравнению с нулем (<u>путь</u> — $S < 0$ вместо $S < \text{путь}$),
07 08	$F x < 0$ <div style="border: 1px solid black; width: 50px; height: 20px; margin: 5px auto;"></div>	<u>нц</u>	ставим команду условного перехода, оставляем пустую ячейку (заполним ее позже).
09 10 11 ...	$P \leftarrow x4$ $P \leftarrow x3$ $F x^2$...	путь := путь + $T^2 \times h$; $T := T + h$	Для вычисления конкретного значения взяли конкретную функцию: $y = x^2$. Присваивания в теле цикла программируем обычно.
20 21	БП 04	<u>кц</u> 22	<u>кц</u> программируем в соответствии с правилами.
22	$\rightarrow C/P$	<u>кон</u>	

87

при останове будет и на индикаторе).

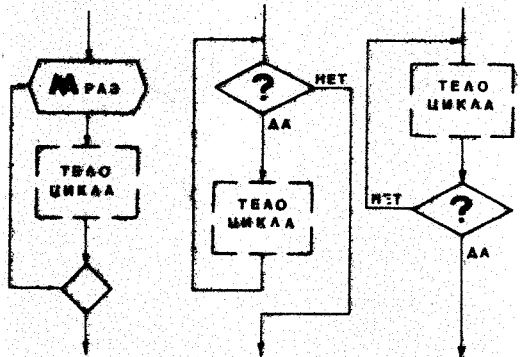
Для иллюстрации работы команд цикла рекомендуется пройти эту программу в пошаговом режиме (класс работает при этом в «командном режиме») с простыми данными, например: $a=2, n=3$. После команды с адресом 07 следует заглянуть в счетчик адреса («F ПРГ», смотрим адрес, «F АВТ») и посмотреть содержимое счетчика цикла (« $P \leftarrow x0$ »); то же самое нужно сделать после выполнения команды «FL 0».

Домашним заданием может быть перепись в виде программы для ПМК одного из ранее разобранных алгоритмов, использующих цикл типа раз.

На следующем уроке можно ввести цикл пока, отталкиваясь от задачи, в которой такой цикл действительно необходим. Ключевым отличием этого вида цикла от цикла раз является то, что количество повторений к моменту начала цикла неизвестно и определяется только в процессе его выполнения. Пример — небольшая модификация задачи о площади криволинейной трапеции: при заданной функции, значениях a и h , определить b , при котором площадь достигает заданной величины S . Алгоритм вместе с программой приведен в табл. 2.

На схемах цикл пока изображается так, как показано на рис. 1б. Из схемы нетрудно усмотреть правила его программирования.

1. Программируется условие, приведенное к сравнению с нулем.
2. Пишется соответствующая команда условного перехода, пропускается ячейка под адрес.
3. Программируется тело цикла.
4. кц для цикла пока программируется командой «БП» с адресом начала проверки



а) цикл раз б) цикл пока в) цикл до



88

условия, затем пропущенная на шаге 2 ячейка заполняется очередным свободным адресом.

Отметим, что в этой программе при оставше на индикаторе будет не результат. Его следует искать в РЗ. С программой можно провести ряд интересных машинных экспериментов, подробности которых описаны на с. 55, 145—146 в [3] и в [8].

Домашним заданием может служить составление программы для одного из ранее разобранных алгоритмов с циклом пока. Следующий урок можно провести как лабораторную работу, на которой ученики должны самостоятельно отладить программу домашней задачи и предъявить учителю ПМК с программой, выдающей правильные результаты.

На последних уроках по этой теме целесообразно ввести еще одну конструкцию цикла: цикл до, проверяющий условие окончания после тела цикла (см. рис. 1 в), который неявно применен в [1].

3



— КАЛЬКУЛЯТОР — ЭТО
ФОРМАЛЬНЫЙ ИСПОЛНИТЕЛЬ...

нц

тело цикла

до условие кц

Правила ее программирования легко выводятся из схемы:

1. Программируется тело цикла.
2. Программируется условие, приведенное к сравнению с нулем.
3. Ставится нужная команда условного перехода с адресом начала тела цикла.

Опыт показывает, что гораздо легче объяснить дополнительную конструкцию, чем то, как свести ее к уже имеющимся. Сравните тело алгоритма «уточнение корня»: (с. 95 [1]).

<u>вещ</u> $c;$	$x := (a+b)/2;$
<u>пока</u> $b-a > 2 \cdot \varepsilon$	<u>нц</u> $x := (a+b)/2;$
<u>нц</u> $c := (a+b)/2;$	<u>если</u> $f(a) \cdot f(x) < 0$
<u>если</u> $f(a) \cdot f(c) < 0$	<u>то</u> $b := x$
<u>то</u> $b := c$	<u>иначе</u> $a := x$
<u>иначе</u> $a := c$	<u>все</u>
<u>все</u>	<u>до</u> $b-a \leq \varepsilon;$
<u>кц</u>	

В варианте с циклом пока дополнительное присваивание нужно именно потому, что тело цикла пока может не выполниться ни разу, и тогда x не получит вообще никакого значения.

Технические подробности, примеры и машинные эксперименты с программой с циклом до описаны на с. 55—60 в [3].

В заключение отметим, что кабинет информатики, оснащенный программируемыми микрокалькуляторами, желательно оснастить также и плакатами, иллюстрирующими правила программирования конструкций и связь алгоритмов, схем и программ.

Продолжение следует.

Литература

1. Основы информатики и вычислительной техники: Пробное учебное пособие Ч. I, II / Под ред. А. П. Ершова, В. М. Монахова. М.: Просвещение, 1985.
2. Основы информатики и вычислительной техники / Под ред. А. П. Ершова. М.: Просвещение, 1988.
3. Штернберг Л. Ф. Программирование на микрокалькуляторе. М.: Просвещение, 1988.
4. Звенигородский Г. А. Первые уроки программирования. М.: Наука, 1985.
5. Гутман Г. Н., Карпилова О. М. Азбука программирования // Информатика и образование. 1987. № 6.
6. Шень А. Информатика в 9 классе // Информатика и образование. 1987. № 6.
7. Штернберг Л. Ф. Зачем микрокалькулятору стек? // Квант. 1986. № 4.
8. Штернберг Л. Ф. Умеет ли считать ваш микрокалькулятор? // Информатика и образование. 1988. № 5.

Экспериментируя со сдвигом тональности, можно убедиться, что слишком высокие ноты звучат хуже, слышится фальшь, вызванная ошибками округления до целого. Для сравнения распечатайте члены прогрессии с округлением и без него. Работая с параметром МНОЖИТЕЛЬ, можно изменять темп исполнения — *andante*, *allegro* и т. п.; вставляя между всеми звуками небольшие паузы, можно получить *staccato*. Вообще кодирование паузы (видимо, нулем) и ее отработку тоже обязательно надо предусмотреть. И собрать мелодии в библиотеку, и сделать каталог мелодий, и музыкальный редактор, и еще, и еще... Развивать эту тему можно бесконечно.

Конечно, в некоторых версиях Бейсика есть специальные операторы **SOUND** и **PLAY**, а в Рапире — стандартные процедуры **ЗВУК** и **НОТА**, облегчающие программирование музыки. Но, овладев ассемблером, вы можете попытаться симитировать звук выстрела или удара, шум мотора и даже синтезировать речь. Желаем успехов!

Л и т е р а т у р а

Техническое описание ПЭВМ «Агат». Книга 2. Фг.3.032.002.ТО1—3. ППП «Школьника». Описание Рапиры и Отладчика.

Морер У. Язык ассемблера для персонального компьютера Эпл. М.: Мир, 1987.

А. ГРИШАЕВ
с. Малиновка

На Луну на МКШ-2

В обстановке «повальной» компьютеризации сельские школы чувствуют себя за бортом корабля, плывущего по волнам информатики. Не случайно многие сельские учителя испытывают раздражение от нового предмета; а что говорить об учениках?

Понятно, что лучший способ приобщить детей к ЭВМ — это игра. Но нам пока не к чему приобщать — мы располагаем лишь простейшими вычислительными средствами, например ЭКВМ МКШ-2. В печати редки сообщения о разработке игр для простейших микрокалькуляторов; между тем можно играть и на них.

На такую мысль нас натолкнул анализ игровой программы «Посадка на Луну». Возможностей МКШ-2 вполне хватает, чтобы произвести необходимые вычисления; кроме калькулятора для проведения игры используется таблица исполнения, заполняемая по результатам вычислений.

Итак, на Луну падает ракета с собственной массой $M=500$ кг и запасом топлива $M1$ (в начальный момент $M1=250$ кг). Расстояние до Луны равно H (в начальный момент $H=15$ км), скорость падения V (сначала — 1 км/с). Включается тормозной двигатель и в течение времени, равного Tc , расходует L кг/с топлива; раскаленные газы вылетают из сопла двигателя со скоростью 3 км/с, создавая ускорение (замедляющее падение ракеты) A задача игрока — так подобрать расход топлива и длительность включений тормозного двигателя, чтобы ракета мягко села на Луну, т. е. $V=0$ при $H=0$.

Для решения этой задачи нужно знать еще ускорение свободного падения в окрестностях Луны; поскольку оно меняется незначительно, можно задать его таблицей:

H	15 000— 10 000	10 000— 8000	8000—4000	4000—0
G	1,58—1,59	1,59—1,6	1,6—1,61	1,61—1,62

Задав величины расхода топлива L и времени торможения T , вычисляем ускорение A (программа для МКШ-2):

3000 $\boxed{\times}$ L $\boxed{\div}$ \boxed{C} 500 $\boxed{+}$ M1 $\boxed{)}$
 $\boxed{-}$ G $\boxed{=}$ A $\boxed{X-P}$

Оно должно быть меньше 49 — иначе чрезмерная перегрузка приводит к гибели экипажа.

Если ускорение в пределах нормы, вычисляем скорость:

A $\boxed{\times}$ T $\boxed{-}$ $\boxed{+}$ $\boxed{=}$ V' $\boxed{=}$ V

где V' — предыдущее значение скорости (такое же обозначение применяем и в дальнейшем).

Вычисляем текущее значение высоты:

H' $\boxed{-}$ \boxed{C} V $\boxed{\times}$ T $\boxed{-}$
 $\boxed{+}$ \boxed{C} $\boxed{P-X}$ $\boxed{\times}$ T \boxed{F} $\boxed{x^2}$
 $\boxed{\div}$ 2 $\boxed{-}$ $\boxed{=}$ H

Остаток топлива:

$L \mid \overline{\times} \mid T \mid \overline{-/} \mid \overline{+} \mid M1' \mid \overline{=} \mid M1$

Результаты заносятся в таблицу и становятся исходными данными для следующего этапа полета.

Пример исполнения программы:

L	T	G	A	V	H	M1
4,0	12	1,58	14,42	783,7	4038	202
11	2	1,6	45,4	692,9	2561,4	180
11	2	1,61	46,9	599	1269	158
11	2	1,61	48,54	501,9	169,36	136
10,5	1	1,62	47,9	454	-309,7	125

Как видите, на этот раз наша ракета образовала на Луне новый кратер. Но решение

задачи существует, и даже с экономией топлива для последующего взлета.

Конечно, применение ЭКВМ с несколькими регистрами памяти существенно упрощает вычисления, но вопросы оптимизации подсчетов каждый может решить самостоятельно.

Когда цель достигнута и ракета благополучно прилунилась, поставьте другую задачу: наибольшая экономия топлива, наименьшее число шагов, наименьшее время посадки, достижение наибольшей высоты на остатках топлива и т. д.

Преимущества игры на ЭКВМ относительны, но они есть: поняв, что сделанный шаг неверен и ведет к катастрофе, можно не возвращаться к началу, как это заставил бы сделать компьютер, а переиграть 1—2 шага.

Счастливой посадки!

М. СТЕПАНОВА

Компьютер и самочувствие школьника

Из множества вопросов, связанных с компьютеризацией образования, пожалуй, один из самых важных — самочувствие школьника, работающего на ЭВМ.

Интенсификация и формализация учебной деятельности, значительное зрительное напряжение, увеличение гиподинамии, специфические условия окружающей среды в дисплейных классах и др. могут оказать неблагоприятное влияние на здоровье школьников.

До последнего времени медицинская наука не располагала данными о влиянии занятий с компьютером на детский организм. Хотя исследования в этом направлении начаты совсем недавно, но уже получены первые результаты (1, 2, 3). Основная задача этих исследований — установить влияние компьютерной деятельности на растущий организм и определить оптимальные условия для этой деятельности, предупреждающие переутомление школьников. Степень утомления школьников зависит от целого ряда как объективных, так и субъективных факторов (рис. 1).

Проводимые исследования посвящены главным образом изучению влияния на функциональное состояние организма школьников объективных факторов, которые, как правило, поддаются количественной оценке и гигиеническому регламентированию.

Прежде всего это относится к длительности и режиму занятий, микроклиматическим параметрам, освещенности, различного

рода излучениям, организации рабочего места, качеству изображения дисплея и др. Однако при прочих равных условиях степень утомления школьника весьма сильно зависит от субъективных факторов, работоспособности, интереса и подготовленности к занятиям.

Субъективные факторы весьма переменчивы и трудноизмеримы. Многие из них, например такие, как подготовленность, интерес к занятиям, эмоциональное состояние ученика, преимущественно зависят от педагогического воздействия.

В нашем исследовании проводилось специальное анкетирование учащихся VII—X классов (481 человек) нескольких московских школ, направленное на изучение таких важных субъективных факторов, как интерес школьников к занятиям на ЭВМ и их самочувствие после этих занятий. Старшеклассники занимались с компьютерами на уроках информатики, а учащиеся VII—VIII классов — на уроках математики. Одна часть школьников (VII—X классы — 222 человека) работали на ЭВМ ДВК-2, а другая (IX—X классы — 259 человек) на ЭВМ БК-0010, у которых в качестве дисплея использовались портативные телевизоры «Электроника-404».

Результаты анкетирования выявили, что у большинства школьников (63,5 % старшеклассников и свыше 70 % учащихся VII—VIII классов) занятия с компьютером проходили в среднем один раз в неделю. Они

Л. ШТЕРНБЕРГ

канд. физ.-мат. наук

Куйбышевский авиационный институт

Уроки с программируемыми микрокалькуляторами

Табличные величины

В соответствии с приведенным ранее вариантом поурочного планирования табличные величины даются в конце года (уроки 29—30). У школьников этот материал вызывает трудности, связанные с понятиями «индекс» и «значение». Чтобы их преодолеть, можно перед табличными величинами рассмотреть работу с литерными величинами: там проблема снимается тем, что индекс — число, а элемент величины — литера.

В любом варианте для обработки табличных и литерных величин лучше вначале использовать цикл **раз**: алгоритмы получаются короче и понятнее. Кроме того, именно так придется программировать на ПМК. Сравним тела алгоритма «сумма таблицы»:

```
сум:=0; i:=1
пока i ≤ N
  нц
    сум:=сум+а [i]; i:=i+1
  кц
```

```
сум:=0; i:=1
цикл N раз
```

```
сум:=сум+а [i]; i:=i+1
кц
```

Здесь естественным образом вводится цикл с параметром, в котором заголовок цикла вбирает в себя и начальную установку параметра, и его изменение, и проверку на окончание цикла.

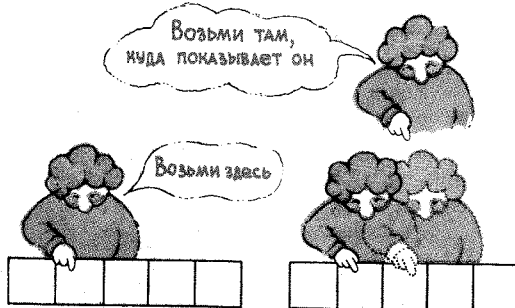
Введя табличные величины, желательно рассмотреть те задачи, которые связаны с их обработкой: они рассеяны по разделам

«Решение задач из курса математики» и «...физики»: вычисление многочлена, поиск в таблице заданного и минимального элемента, упорядочение, метод наименьших квадратов и т. д.

Проработав хотя бы часть этих алгоритмов, можно приступить к их реализации на ПМК. Эта тема достаточно сложна из-за того, что на ПМК нет средств, соответствующих индексированию (обращению к элементу таблицы) в «чистом виде». Поэтому приходится использовать преобразование алгоритма с введением дополнительных средств в алгоритмический язык (А-язык). Кроме того, необходимое средство ПМК — косвенная адресация — само по себе достаточно сложно для восприятия. Поэтому эту тему, видимо, не следует считать обязательной и добиваться ее усвоения всеми.

Из-за малой памяти ПМК практически невозможно реализовать работу с двумерными таблицами, поэтому рассмотрим работу с одномерными.

Для того чтобы команда, работающая в цикле, при каждом новом исполнении выбирала (или засылала) число из (или в) другого регистра, она должна изменяться



в процессе исполнения программы невозможно. Поэтому номер регистра, с которым работает команда, нельзя задавать в самой команде (заметьте, что вторая цифра кодов команд «П—х» и «х—П» — это и есть номер регистра), а надо задавать в таких местах, которые доступны для изменений в процессе выполнения программы, — ими являются адресуемые регистры и регистры стека. Так, учитель должен подвести к понятию косвенной адресации (на ПМК косвенная адресация осуществляется только через адресуемые регистры). Понятие косвенной адресации можно объяснить на примере, не связанном с ПМК (см. рис. 1). Далее надо напомнить, что с косвенной адресацией мы имели дело при занесении программы в память ПМК: команда попадает туда, куда указывает счетчик адреса, а затем счетчик увеличивается (только здесь он увеличивается не до, а после засылки). Для иллюстрации работы команд косвенной адресации можно провести следующий эксперимент (класс при этом работает в «командном режиме»): засылаем в регистры числа, равные номеру регистра (0 — в R0, 1 — в R1, ... 13 — в RD), и выполняем команды (см. табл. 1).

Таблица 1

Команда	Индикатор	Комментарий
К П—х 4	5.	Вызов произошел из R5; только там есть число «5»
К П—х 4	6.	Теперь та же команда вызвала число из R6
К П—х 4	7.	Та же команда вызвала число из R7
П—х 4	000 000 07.	А в R4 — уже число 7 (в необычном виде)

Аналогично проводим демонстрацию косвенной записи, косвенной работы с регистрами, работающими с автоуменьшением и без автоизменений. Полностью во всех деталях этот эксперимент описан в [1].

Итак, ясно, что если команду «К П—х М» или «К х—П М» заставить работать в цикле, то она может обрабатывать последовательность регистров, хранящих разные элементы таблицы.

Но в А-языке для обработки таблиц используется индексация, а на ПМК — косвен-

ная адресация. Следовательно, перед переводом алгоритма с А-языка на язык ПМК его нужно преобразовать так, чтобы заменить индексацию на косвенную адресацию. В учебнике понятия косвенной адресации в А-языке нет. Перед учителем возникает два варианта: а) ввести в А-язык дополнительное понятие (особенности конкретной ВТ всегда накладывают отпечаток на преподавание), б) не вводить дополнительное понятие и выходить из сложившейся ситуации объяснениями на интуитивном уровне, как это сделано в учебнике, где неявно используются сразу две не введенные в явном виде конструкции: косвенная адресация и цикл до (с проверкой условия в конце). Сомневающимся можно предложить попробовать оба варианта и лично убедиться, что проще ввести дополнительную конструкцию, чем объяснять, как обходиться без нее. Поэтому здесь рассмотрен только вариант б).

Разница между индексом и адресом заключается в том, что адрес — это *абсолютный* номер регистра, отсчитываемый от начала массива регистров, а индекс — *относительный* номер, отсчитываемый от начала той группы регистров, которую занимает данная таблица, причем счет ведется, возможно, не с 1, а с нижней границы диапазона индекса, указанного в описании таблицы. Например, для таблицы ...таб А [k:m] адрес элемента A [i] равен $m + (i - k + 1)$, где m — номер первого из регистров, отведенного для хранения таблицы. Введем в А-язык две новые конструкции:

адрес переменная

которая не нуждается в пояснениях, и косвенную адресацию, соответственно простую, с автоувеличением и с автоуменьшением, записываемую:

[k] [⁺k] [⁻k]

и означающую обработку переменной, адрес которой записан в k. Для переменных, используемых в косвенной адресации, лучше ввести и специальные типы косв, косв— и косв+, так как эти типы следует учитывать при распределении регистров.

При обработке табличных величин обычно используют цикл для. Для ПМК его придется преобразовать к циклу раз, что делается по строго формальным правилам:

для k от a до b шаг c нц
...
кц

→ { k:=a;
цикл (b - a)/c + 1 раз
... k:=k+c
кц }

Далее вводится переменная с описателем косв, которая заменяет индекс, все обраще-

ния к элементам таблицы заменяются на косвенную адресацию, а начальное присва-

ивание индексу заменяется на присваивание косвенной переменной адреса соответствующего элемента массива.

Преобразуем, например, алгоритм «сумма таблицы»:

```

алг  сумма таблицы (вещ таб А [1:6], вещ сум)
  арг А; рез сум
  нач цел i
    сум:=0;
    для i от 1 до 6 по
      сум:=сум+А [i]
  кц
кон

```

преобра-
зование
цикла

```

сум:=0; i:=1
цикл 6 раз
  сум:=сум+А [i]; i:=i+1
кц

```

Теперь переходим к косвенной адресации:

```

нач цел i
  сум:=0; i:=1
  цикл 6 раз
    сум:=сум+А [i]; i:=i+1
  кц

```

```

нач косв k
  сум:=0; k:=адрес А [1]
  цикл 6 раз
    сум:=сум+ [k]; k:=k+1
  кц

```

88

Как видим, преобразование выглядит достаточно просто. Правила программирования алгоритмов с этими конструкциями таковы:

1. Распределение памяти начинается с выделения регистров для переменных с описателями косв, косв— и косв+, так как для них подходят не все регистры.

2. Конструкция адрес программируется командами набора соответствующего числа.

3. Конструкции [k], [+k] и [—k] программируются командами «К П—х ...» или «К х—П ...» с номером отведенного переменной k регистра.

иметь свою косвенную переменную для каждой таблицы.

Если требуется обработать все элементы таблицы в порядке их расположения, то удобно применять косвенную адресацию с автоувеличением или автоуменьшением. Процесс преобразования алгоритма таков (подчеркнуты меняющиеся элементы):

```

косв k
сум:=0; k:=адрес А [1]
цикл 6 раз
  сум:=сум+ [k]; k:=k+1
кц

```

Таблица 2

Адрес	Команда	А-язык	Комментарий
	счетчик — P0 k — P7 А [1:6] — P8—PD сум — P1	<u>алг</u> сумма <u>косв</u> k (<u>вещ</u> ...А... сум)	Распределение памяти начинается с отведения регистров счетчику цикла и косвенной переменной, далее отводим место прочим переменным
00 01 02 03	0 х—П 1 8 х—П 7	сум:=0 k:= <u>адрес</u> А [1]	Адресом А [1] является 8; набираем «8» в РХ и засылаем адрес в k
04 05	6 х—П 0	<u>цикл</u> 6 <u>раз</u>	Количество повторений цикла засылаем в счетчик (обычный способ реализации цикла <u>раз</u>)
06 07 08 09	П—х 1 К П—х 7 + х—П 1	сум:= сум+ [k]	Работа с косвенной переменной реализуется командой косвенной адресации (в данном случае чтения)
10 11 12 13	П—х 7 1 + х—П 7	k:= k+1	Увеличиваем адрес, содержащийся в k
14 15 16	F L0 06 С/П	<u>кц</u> <u>кон</u>	Завершаем цикл стандартным для цикла <u>раз</u> образом

Адрес	Команда	А-язык	Комментарий
			Команды 00...01 без изменений
02 03	7 x — П4	$k :=$ <u>адрес</u> А [0]	Элемента А [0] в таблице нет, но если бы он был, то занимал бы Р7
04 05	6 x — П 0	<u>цикл</u> 6 <u>раз</u>	Заголовок цикла программируется так же
06 07 08 09	П — x 1 К П — x 4 + x — П 1	сум:= сум+ [+k]	А здесь косвенная адресация помимо чтения нужного числа еще и увеличивает значение k
10 ...	F L 0 ...	<u>кц</u> ...	Далее все без изменений

косв k

сум:=0; k:=адрес А [0]

цикл 6 раз

k:=k+1; сум:=сум+ [k]

кц

косв— k

сум:=0; k:=адрес А [0]

цикл 6 раз

сум:=сум+ [+k]

кц

Теперь для k можно взять Р4, и, оставив остальное распределение памяти без изменений, получим программу (см. табл. 3).

Здесь имеется широкое поле для оптимизации программы: хранение суммы непосредственно в РХ, размещение таблицы в обратном порядке (А [6]. — в Р8, А [5] — в Р9) и ее обработка с применением адресации с автоуменьшением и т. д. Все это описано в [1] на с. 75—79, и эти варианты можно рассмотреть в кружках или классах с математическим уклоном, для урока в обычном классе можно рекомендовать только простейший вариант (можно и не обсуждать косвенную адресацию с автоизменением, упомянув только о ее существовании и не рекомендуя использовать для нее Р0...Р6).

По этой теме для проработки на ПМК можно взять множество имеющихся в учебнике задач по обработке таблиц.

Сочетания программных конструкций

В наших примерах были рассмотрены программы, содержащие только по одной конструкции нужного нам вида: одно ветвление или один цикл. Именно на таких программах и следует пояснять работу конструкции.

После введения каждой конструкции в отдельности можно перейти к задачам, где требуется применить две или более конструкции: два цикла, цикл и ветвление и т. д. Конструкции сочетаются только двумя способами: *следованием* и *вложением*. Следование конструкции при программировании проблем не вызывает, а при программировании вложений учащиеся допускают много ошибок. Учитель должен сформулировать правило программирования сочетаний разных конструкций и следить за его выполнением, тем более что формулируется оно просто: «Каждая конструкция программируется по своим правилам, независимо от того, что следует за ней или за чем следует она, а также, что вложено в нее или во что вложена она».

Проследим реализацию этого правила на примере алгоритма подсчета числа отрицательных элементов таблицы, где алгоритм содержит цикл, в который вложено ветвление (без ветви иначе) (см. табл. 4).

Здесь возможны попытки учеников «оптимизировать» программу, рассуждая так: если элемент неотрицателен, то подсчитывать его не надо, а значит, надо просто перейти к обработке следующего элемента, т. е. с адреса 08 перейти на адрес 06. В итоге в ячейке 08 появляется адрес 06 — и в программу внесена ошибка: так как не выполнена команда «F L0», то будет неправильным число повторений цикла (вплоть до заикливания). Чтобы ошибок не было, нужно приучать учеников к строго формальному переводу алгоритмов на язык ПМК: адрес заполняется при обработке все (и не ранее) очередным значением счетчика адреса.

Возможности оптимизации программ за счет команд переходов и использования других особенностей ПМК действительно

Адрес	Команда	А-язык	
	счетчик — P0 A[1:8] — P5 — PC кол — P1 k — P4	<u>алг</u> число отриц. (<u>таб</u> A[1:8] <u>цел</u> кол) <u>нач косв</u> + k	
00 01	0 x — П 1	кол := 0	
02 03	4 x — П 4	k := <u>адрес</u> A[0]	
04 05	8 x — П 0	<u>цикл</u> <u>8 раз</u>	Элементы программы, необходимые для цикла типа <u>раз</u>
06 07 08	K П — x 4 F x < 0 13	<u>если</u> [+k] ≤ 0 <u>то</u>	Элементы программы, необходимые для программирования ветвления
09 10 11 12	П — x 1 1 + x — П 1	кол := кол + 1	
13 14	FL 0 06	<u>все</u> ← <u>кц</u> ←	
15	C/П	<u>кон</u>	

нередко появляются, но они не должны быть темой уроков, а лишь предметом рассмотрения на кружках.

Работа со стеком

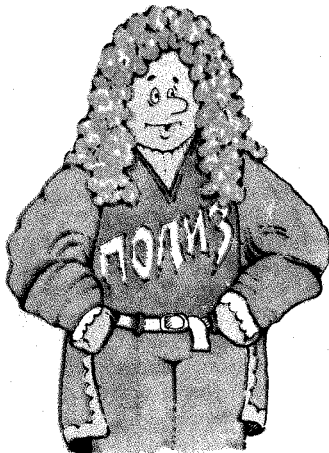
Этот материал (если учитель решит познакомить с ним учащихся) должен быть дан на 12—14-м уроке, однако рассмотрение было до сих пор отложено, так как он является дополнительным: приемы работы со

стеком — это особенность ПМК данного семейства, они выйдут из употребления вместе с уходом ПМК с «арены» школьной информатики, в то время как все остальное (за исключением разве что косвенной адресации) приложимо без изменений к переводу алгоритмов, например, на Бейсик.

Вместе с тем этот материал математически красив, легко доступен и хорошо воспринимается учащимися как удобное средство выполнения цепочечных вычислений.

Использование стека связано с математической конструкцией, называемой *польская инверсная запись* (ПОЛИЗ), изобретенной в 1921 г. польским математиком Я. Лукасевичем. Обычно в учебниках и справочниках по ПМК мало говорится о ПОЛИЗе, без объяснения, что это такое; в результате на уроках появляются «задачи»: в каком порядке вычислять сложное выражение, чтобы было меньше вспомогательных действий (запоминаний/вызовов чисел, обменов между регистрами и т. д.). Эти задачи не имеют права на существование, ибо ПОЛИЗ как раз и позволяет, не задумываясь, получать оптимальную последовательность команд.

Идея ПОЛИЗа заключается в том, что знак операции записывается после своих



операндов, причем сам операнд может быть выражением, записанным в ПОЛИЗе:

ПОЛИЗ

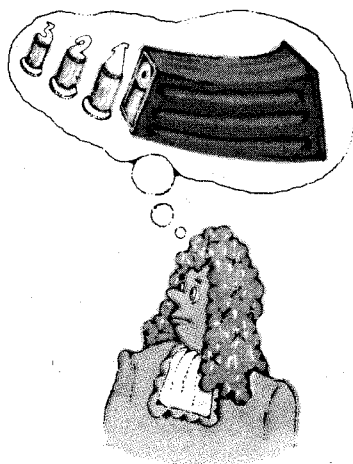
$a-b$ $a\ b-$
 $\sin(x)$ $x\ \sin$
 $a \cdot (b-c)$ $a\ b\ c-$
 $a \cdot b-c$ $a\ b\ \times c-$

Как показывают два последних примера, в ПОЛИЗе нет скобок и нет старшинства операций: они не нужны — операции выполняются в порядке их записи, что и есть основное преимущество ПОЛИЗа.

Для вычисления записанного в ПОЛИЗе выражения нужен стек, который можно представлять (эту аналогию учитель должен дать обязательно) как магазин стрелкового автомата: коробочка с пружиной, в которую только с одной стороны можно втолкнуть патрон (в нашем случае патрон будет не с пулей, а с числом). Вынуть его можно тоже только с одной стороны. Правила вычисления выражения на ПОЛИЗе:

1. Выражение просматривается один раз слева направо.
2. Числа (значения переменных или константы) помещаются в стек.
3. Если встретилась операция, то из стека выбирается столько чисел, сколько нужно для операции (одно или два), над ними выполняется операция, результат помещается назад в стек.
4. В конце в стеке остается только окончательный результат.

Но для вычисления выражения, записанного в обычной форме, нет необходимости в предварительном преобразовании в ПОЛИЗ. Рассуждая определенным образом,



можно получить оптимальную программу вычисления выражения за один его просмотр.

1. Выражение записывается в одну строку (без многострочных формул) и просматривается один раз слева направо.
2. Константы и значения переменных записываются в стек.
3. Операция выполняется немедленно после того, как в стеке оказались все ее операнды.

Рассмотрим реализацию этих правил на примере выражения

$$\frac{\sin(3 \cdot a - \frac{b}{2})}{c-d}$$

Таблица 5

Элемент	Рассуждения	Действие	Стек			
			X	Y	Z	T
\sin	Пока невыполнимо: отложим					
3	Константу заносим в стек	3	3			
x	Пока невыполнимо: отложим					
a	Значение переменной — в стек и можно выполнить умножение	П — x A ×	a 3 · a	3		
—	Пока невыполнимо: отложим					
b	Значение переменной — в стек	П — x B	b	3 · a		
/	Пока невыполнимо: отложим					
2	Значение константы — в стек и можно выполнить «/»: делим	2 ÷	2 a/2	b 3 · a	3 · a	
	А теперь можно вычитать, и можно «синусовать»	— F sin	(3 · a — — b/2) sin (...)			
c	Значение переменной — в стек	П — x C	c	sin (...)		
—	Операция пока откладывается					
d	Значение переменной — в стек	П — x D	d	c	sin (...)	sin (...)
	И остается выполнить отложенные операции	— ÷	c — d рез-т	sin (...)		

Считаем, что a находится в РА, b — в РВ, c — в РС, d — РД. Обратите внимание, что за тем, где что находится в стеке, следить не надо: все автоматически оказывается на своих местах. Поэтому закройте пока листом бумаги часть табл. 5, на которой изображен стек (на уроке это тоже следует сделать). Запись выражения в одну строку имеет вид: $\sin(3 \cdot a - b/2) / (c - d)$ (знаменатель пришлось взять в скобки). Читаем его слева направо, скобки читать не будем (они только учитываются для определения порядка действий).

Теперь можно открыть правую часть таблицы и посмотреть, как «качаются» операнды в стеке (в примере для лучшей иллюстрации взяты непрерывные операции: деление, вычитание).

ПОЛИЗ для нашего выражения имеет вид:
 $3 \ a \times \ b \ 2 \ / \ - \ \sin \ c \ d \ - \ /$

Если сравнить его с полученной программой, то становится ясно, почему говорят, что ПМК работает на идее ПОЛИЗа.

Если приходится набирать две константы подряд, то их надо разделять командой «V↑», а команды «FC» и «←→» при классической работе со стеком вообще не используются. Регистр Т потребуется в более сложных выражениях, а ситуацию, где стека не хватает, и придумать-то непросто. Технические подробности можно посмотреть в [1] на с. 11—18 или [2].

Продолжение следует.

Литература

1. Штернберг Л. Ф. Программирование на микрокалькуляторе. М.: Просвещение, 1988.
2. Штернберг Л. Ф. Зачем микрокалькулятору стек // Квант, 1986. № 4.

Е. ЛИНЕЦКИЙ

Городской компьютерный центр: опыт, проблемы, перспективы

О Свердловском городском компьютерном центре (ГКЦ) уже упоминалось в [1]. Цель настоящей статьи — более подробно рассказать о ГКЦ, описать конкретное применение предложенного в [1] подхода к обучению, обосновать оптимальность организации обучения в ГКЦ.

Наш ГКЦ объединяет под одной крышей специализированный программистский учебно-производственный комбинат, преподавание школьной информатики (для учащихся школ, не оснащенных вычислительной техникой), школу юных программистов и методический центр для учителей информатики. Мы стараемся не забывать и о сельских школьниках. С 1987 г. организуются недельные практические занятия по информатике для старшеклассников из разных районов области. Теоретические вопросы они изучают дома по тому же учебнику, что и городские школьники (см. [1]). Программа полного практического курса — 48 ч. Таким «вахтовым» способом в 1987/88 учебном году около тысячи сельских учащихся смогли поработать на ЭВМ.

В ГКЦ установлено 48 ЭВМ «Роботрон 1715» (операционная система SCP — роботроновская версия хорошо известной CP/M) — это четыре компьютерных класса. Кроме того, есть два теоретических

класса. Учащиеся группы информатики и юные программисты занимаются 2 ч в неделю, а учащиеся УПК приходят к нам раз в неделю на целый учебный день — 6 ч, из которых первые два занимаются теорией, а остальное время (с необходимыми перерывами) работают с ЭВМ. Каждый год в ГКЦ почти 400 школьников обучаются в УПК, более 1700 — в группах информатики и около 70 посещают занятия школы юных программистов.

Наша идеология курса ОИВТ изложена в [1], функции методического центра очевидны, поэтому речь пойдет в основном о профильном обучении в группах УПК и о школе юных программистов (ШЮП), хотя все формы обучения в ГКЦ взаимосвязаны.

ШЮП возникла при Свердловском пединституте в 1982 г., а в 1986 г. «переехала» в ГКЦ. Опыт работы в ШЮП во многом помог нам почти на самом старте школьной компьютеризации предложить вполне сформировавшийся подход к обучению информатике. В обучении юных программистов с самого начала стали выделяться два направления. С одной стороны, нужно было организовать увлекательный компьютерный ликбез для массы любознательных детей (в ШЮП принимаются ученики IV—VIII классов), а с другой — придумать занятие для тех юных

Л. ШТЕРНБЕРГ

Куйбышевский авиационный институт

Уроки

с программируемыми микрокалькуляторами

Подпрограммы

После изучения темы «Вспомогательные алгоритмы» можно перейти к реализации их на ПМК. На первом уроке рекомендуется дать команды «ПП» и «В/О» без упоминания стека возвратов: просто отметить, что команда «ПП» запоминает адрес возврата в некотором регистре, содержимое которого увидеть невозможно, а «В/О» выполняет переход по запомненному адресу. Объяснение работы этих команд займет не более 10 мин. Сами по себе команды «ПП» и «В/О» позволяют организовывать только подпрограммы без параметров. Такие подпрограммы либо всегда выдают один и тот же результат, либо должны работать с переменными (всегда с одними и теми же), принадлежащими основной (головной) программе.

В этом месте учитель встретится с одной методической тонкостью. В учебнике нет ни явного запрета, ни явного разрешения на использование во вспомогательных алгоритмах значений переменных из основного алгоритма, хотя неявно предполагается, что связь идет только по параметрам, а все переменные, описанные после *нач*, — *локальные* (собственные) переменные вспомогательного алгоритма: при исполнении вспомогательного алгоритма мы рисуем таблицу со своими столбцами для каждой такой переменной.

В языках программирования эта проблема решена по-разному: в Фортране, например, доступа из подпрограммы к переменной из головной программы нет, в Паскале — есть, в

Бейсике — только так и идет работа и т. д. Учителю можно порекомендовать такой вариант: если переменная не описана ни среди параметров, ни после *нач*, то предполагается, что вспомогательный алгоритм берет переменную, описанную в основном алгоритме.

В качестве примера подпрограммы без параметров можно взять подпрограмму набора физической или химической константы, известной школьникам, например числа Авогадро ($6.02252 \cdot 10^{23}$). Для ее набора нужно 10 команд: «6.0 2 2 5 2 ВП 2 3». Предположим, что она нужна в расчете более одного раза, а для ее хранения регистра не нашлось (все заняты другими данными). Тогда выделение этой последовательности команд в подпрограмму сократит программу, позволив не выписывать всю последовательность многократно.

Для иллюстрации работы команд «ПП» и «В/О» можно ввести рассмотренную выше программу в память ПМК с адреса 20, а с адреса 00 простейшую головную программу: «ПП 20 2—ПП 20» и пройти ее в пошаговом режиме (класс работает при этом в «командном режиме») с заглядыванием в счетчик адреса до и после выполнения команд «ПП» и «В/О» (табл. 1).

И так далее. После второго обращения к подпрограмме убедимся, что будет выполнен возврат «под» вторую команду «ПП».

Одна из задач написания вспомогательных алгоритмов — это «обучение» исполнителя новым последовательностям действий, которые становятся для него новыми командами (см. [1]). Этот момент следует акцентировать и проиллюстрировать на примере: после того как в памяти ПМК оказалась упомянутая выше подпрограмма, мы

Таблица 1

Клавиши	Индикатор	Комментарий
F ПРГ	00	Стоим перед выполнением команды «ПП».
F АВТ ПП	0. 0.	Выполнили команду «ПП».
F ПРГ	00 00 00	Убедились, что выполнен переход на адрес 20.
F АВТ ПП, ПП	0. (10 раз)	Проходим по шагам подпрограммы.
F ПРГ	0С 02 03	Стоим перед командой «В/О».
F АВТ ПП	6.02252 6.02252	Выполняем команду «В/О».
F ПРГ	20 53	Убеждаемся, что вернулись «под» команду «ПП».
F АВТ	6.02252	
...	...	

можем считать, что ПМК «обучен» новой команде — команда «ПП 20» мало чем теперь отличается от команды «Фл» и выполняет (как и команда «Фл») занесение на регистр X некоей специфической константы, хотя и делает это гораздо дольше, чем занесение л.

Реализацию подпрограмм с параметрами можно выполнить на ПМК в точном соответствии с тем, как это описано в учебнике [2]. Для выполнения вспомогательного алгоритма заводится своя таблица переменных, значения аргументов заполняются значениями фактических параметров, выполняется тело, затем значения результатов переписываются из таблицы вспомогательного алгоритма в соответствующие переменные из таблицы основного алгоритма. Следовательно, на ПМК для переменных, используемых в подпрограмме, надо выделить регистры, отличные от тех, которые используются в головной программе; в регистры, соответствующие аргументам, заслат значения фактических параметров, обратиться к подпрограмме, а затем из регистров, соответствующих результатам, переслать значения в регистры, отведенные переменным головной программы. Проиллюстрируем это на примере подпрограммы, выполняющей деление нацело с остатком двух чисел по алгоритму:

алг *нацело* (цел *a*, *b*, *часть*, *ост*)

—арг *a*, *b*; рез *часть*, *ост*

нач *часть* := $[a/b]$;

—ост := $a - b \cdot \text{часть}$

кон

к которому мы обращаемся командой *нацело* (*aa* — *bbb*, *s*, *ча*, *ост*)

Для используемых в головной программе переменных *aa*, *bbb*, *s*, *ча* и *ост* отведем регистры Р0 — Р4 соответственно, для используемых в подпрограмме переменных *a*, *b*, *ча*ст и *ост* — Р5—Р8. Обратите внимание (как свое, так и учеников), что у нас имеются две переменные *ост*: своя — у головного алгоритма и своя — у вспомогательного (табл. 2).

Этой реализацией обращений к подпрограмме можно воспользоваться в слабых классах, однако она страдает двумя недостатками: во-первых, неэффективностью, во-

Таблица 2

Адрес	Команда	А-язык	Комментарий
00	П—х 0	<i>нацело</i> (<i>aa</i> — <i>bbb</i> , <i>с</i> , <i>ча</i> , <i>ост</i>)	Значение разности <i>aa</i> и <i>bbb</i> пересылаем в регистр, соответствующий первому аргументу подпрограммы, значение второго аргумента — в регистр, соответствующий второму аргументу; выполнением собственно обращение к подпрограмме (адрес ее размещения пока выбран произвольно), а теперь пересылаем результаты из регистров, выделенных подпрограмме, в регистры, выделенные головной программе
01	П—х 1		
02	—		
03	х—П 5		
04	П—х 2	<i>ча</i> ст: = $[a/b]$	С адреса 20 начинается подпрограмма, которая работает только с отведенными ей регистрами, не заботясь о том, откуда взялись аргументы и куда денутся результаты. Команды вычисления остатка выписывать не будем
05	х—П 6		
06	ПП		
07	20		
08	П—х 7		
09	х—П 3		
10	П—х 8		
11	х—П 4		
...
20	П—х 5	<i>ча</i> ст: = $[a/b]$	С адреса 20 начинается подпрограмма, которая работает только с отведенными ей регистрами, не заботясь о том, откуда взялись аргументы и куда денутся результаты. Команды вычисления остатка выписывать не будем
21	П—х 6		
22	÷		
23*	F []		
24	х—П 7		
25	...	В/О	
...	...		
31	В/О		

* Для ПМК БЗ-34, МК-54, где нет команды «F []», для выделения целой части из заведомо положительного числа надо выполнить последовательность команд «х—П *m*, К П—х *m*, П—х *m*», где *m* — один из номеров регистров из диапазона 7 — D.

вторых, методической неточностью. Такая передача параметров предполагает, что головная программа «знает», где что размещается в подпрограмме, а вот это как раз она «знать» и не обязана.

Отметим (этот момент нужно акцентировать и на уроке), что для использования вспомогательного алгоритма достаточно знать его заголовок, а тело может быть написано как угодно. Например, если в приведенной выше подпрограмме и в соответствующем ей алгоритме заменить тело на цикл последовательного вычитания:

$част:=0; ост:=a;$

пока $ост>b$ **нц**

$ост:=ост - b; част:=част + 1$

кц

то обращаться к алгоритму (подпрограмме) можно точно так же, как и ранее.

Но и заголовок нужно знать не весь. Поясним это на примере заголовка алгоритма суммирования таблицы, содержащего параметры разных типов (чем он для наших целей и удобен):

алг сумма (вещ таб a [1:M], цел M, вещ сум)

$\text{— арг } a, M; \text{ рез } сум;$

Перепишем его в несколько иной форме, объединив список параметров и списки аргументов и результатов (методически очень удачная форма, используемая в [4], которую будем применять далее):

алг сумма (арг вещ таб a [1:M], цел M, рез вещ сум)

Теперь, если убрать отсюда имена переменных, то останется именно то, что надо знать для использования алгоритма:

алг сумма (арг вещ таб, цел, рез вещ)

а именно: алгоритму надо дать два аргумента — вещественную таблицу и целое число, а выдает алгоритм один вещественный результат. При этом то, как называются эти параметры внутри вспомогательного алгоритма, знать нет необходимости. При таком изложении становится очевидным, что параметры сопоставляются по порядку следования, а не по именам.

Итак, вспомогательному алгоритму (подпрограмме) надо выдать аргументы в нужном порядке, а затем забрать результаты. Наиболее естественно это сделать через стек. Применительно к рассмотренной выше подпрограмме деления с остатком получим следующее (табл. 3).

Отметим, что если бы наша подпрограмма выдавала только один результат, например остаток, то ее работа шла бы внешне по тому же принципу, что и любой двуместной операции: выбираем операнды из стека и оставляем в стеке результат, т. е., написав

Адрес	Команда	А-язык	Комментарий
00	П—х 0	наце- ло (Заносим в стек значение 1-го аргумента, заносим в стек значение 2-го аргумента, выполняем собственно обращение к подпрограмме; теперь забираем из стека оставленные там подпрограммой результаты
01	П—х 1		
02	—	aa — bbb, с,	
03	П—х 2		
04	ПП		
05	20		
06	х—П 3	ча,	
07	F b		
08	х—П 4	ост);	
...	...		
20	х—П 6		Подпрограмма выбирает из стека аргументы и засылает их туда, куда ей надо,
21	↔		
22	х—П 5		либо обрабатывает непосредственно в стеке.
23	÷	част:=	
...	Результаты оставляет в стеке в нужном порядке
...	V/O		

подпрограмму, мы как бы научили ПМК еще одной операции: «остаток».

Разобрав на уроке эти (или аналогичные примеры), следует сформулировать правила программирования вспомогательных алгоритмов.

1. Значения аргументов вызова вспомогательного алгоритма засылаются в стек в порядке их перечисления в списке параметров.

2. Ставится команда «ПП» с пока незаполненным адресом начала подпрограммы.

3. Значения результатов извлекаются из стека в порядке их перечисления в списке параметров.

4. По завершении программирования основной программы определяется адрес, с которого размещается подпрограмма (очередной свободный адрес), — этим адресом заполняются пустые адреса команд «ПП».

5. Для параметров и прочих используемых в подпрограмме переменных отводятся регистры из числа не занятых в головной программе и ранее написанных подпрограммах.

6. Подпрограмма начинает работу с извлечения из стека значений парамет-

ров в порядке, обратном порядку их перечисления в списке аргументов.

7. Тело подпрограммы программируется как обычно.

8. Результаты засылаются в стек в порядке, обратном порядку их перечисления в списке результатов.

9. кон для подпрограммы программируется командой «В/О».

Добавим к этим правилам несколько комментариев для учителя, связанных с возможными вопросами учеников.

1. Что делать, если аргументов или результатов более четырех?

Для школьных алгоритмов это ограничение не актуально, но в целом это вопрос того же типа, что и «А если программа не помещается в памяти ПМК?»: у ПМК довольно ограниченные возможности, и далеко не все на нем реализуется.

78 2. Как передать через параметры таблицу?

Малая память ПМК делает почти невозможным размещение в ней более одной таблицы, поэтому передача таблиц через параметры мало актуальна, но вопрос интересен и учитель должен знать ответ на него. Тот способ передачи параметров, который описан в учебнике и который рассмотрен здесь, носит название «передача параметров по значению-результату» и в практических языках программирования используется нечасто. Чаще используется другой прием — «передача параметров ссылкой» (применяется в Паскале, Фортране, ПЛ/1 и др.), когда подпрограмме передается адрес переменной или таблицы, а не копия ее значения. Использование этого приема на ПМК описано в [3], с. 111—113, но обсуждать его, по-видимому, целесообразно только на кружках, в математических школах или очень сильных классах.

3. Некоторые особенности ПМК заставляют отводить память для используемых в подпрограммах переменных одновременно с отведением памяти в головной программе: для используемых в подпрограмме циклов и косвенных переменных могут потребоваться регистры из числа первых.

В качестве домашнего задания следует дать одну-две задачи на реализацию на ПМК вспомогательных алгоритмов из числа разобранных ранее на уроке. Очень показательны такие задачи:

Задача 1. Написать вспомогательный алгоритм, вычисляющий по длинам сторон треугольника a , b , c угол, лежащий против стороны a , и тремя обращениями к нему вычислить все углы треугольника.

Сам алгоритм очень прост:

алг угол (арг вещ a , b , c , рез вещ $уг$)

нач $уг := \arccos((b^2 + c^2 - a^2) / (2 \cdot b \cdot c))$

кон

а три обращения к нему отличаются только порядком задания аргументов

угол (a , b , c , $угA$); угол (b , c , a , $угB$);

угол (c , a , b , $угC$)

при этом получаются разные результаты.

Этот пример прекрасно иллюстрирует сопоставление формальных и фактических параметров по порядку их следования в списках, а не по именам переменных: после того как значения аргументов засланы в стек, у подпрограммы нет никаких средств определить, откуда взялось значение, находящееся в том или ином регистре стека. Программу для слегка усложненного варианта этой задачи можно найти в [3], с. 107—110.

Задача 2. Написать вспомогательный алгоритм, вычисляющий длину отрезка по координатам вершин, и головной алгоритм, который тремя обращениями к вспомогательному получает длины сторон треугольника, координаты вершин которого заданы в таблицах $X[1:3]$ и $Y[1:3]$, где $(X[K], Y[K])$ — координата одной из вершин. Реализовать алгоритмы на ПМК.

Алгоритмы имеют вид:

алг стороны (арг вещ таб $X[1:3]$, $Y[1:3]$, рез вещ $a12$, $a23$, $a31$)

нач длина ($X[1]$, $Y[1]$, $X[2]$, $Y[2]$, $a12$)

длина ($X[2]$, $Y[2]$, $X[3]$, $Y[3]$, $a23$)

длина ($X[3]$, $Y[3]$, $X[1]$, $Y[1]$, $a31$)

кон

алг длина арг вещ ($x1$, $y1$, $x2$, $y2$, рез вещ $дл$)

нач $дл := \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$

кон

Их реализация на ПМК проблем не вызывает. При их обсуждении на следующем уроке нужно обратить внимание на то, что аргументами вспомогательного алгоритма являются вещественные переменные, а фактическими параметрами — элементы таблицы. Это допустимо, так как элемент вещественной таблицы — вещественный, а после засылки аргументов в стек никаким способом нельзя определить из каких переменных — простых или элементов таблиц — взяты значения.

После анализа домашнего задания можно уточнить работу команд «ПП» и «В/О», введя понятие стека возвратов, который позволяет из одной подпрограммы обращаться к другой. В качестве примера, где такое обращение необходимо, можно взять вспомогательный алгоритм вычисления площади криволинейной трапеции или уточнения корня методом деления пополам, которые обращаются к алгоритму вычисления функции. Соответствующие программы можно найти в [3], с. 101—105, 114—117.

Полезно отметить, что, в отличие от описанного в учебнике, на ПМК головной и вспомогательный алгоритмы отличаются тем, что кон для головного реализуется командой «С/П», а для вспомогательного — «В/О», а также по-разному передаются параметры. Но если бы команда «В/О» при пустом стеке возвратов выполняла оставов (а не переход на адрес 01), то подпрограмма не отличалась бы от головной программы. Что же касается параметров, то и для головной программы их можно задавать в стеке, а не в регистрах (такой прием называется СТ-вводом — см. [3], с. 123). Иными словами все несоответствия описанному в учебнике — это мелкие детали, вызванные особенностями используемой техники.

Следующий «калькуляторный» урок посвящается реализации на ПМК функций, но предварительно разберем предшествующий ему урок, на котором вводится это понятие, так как мотивация его введения в учебнике недостаточна.

Начнем с определения понятия функции и назначения отдельных частей ее описания.

Функция (или, более строго, алгоритм вычисления значения функции) — это способ оформления вспомогательного алгоритма, имеющего единственный результат, который не является табличной величиной.

Например, рассмотренный выше алгоритм нацело нельзя оформить как функцию, так как он выдает два результата; алгоритм, заполняющий таблицу нулями, не может быть функцией, так как его результат хоть и единственный, но табличный; а алгоритм суммирования таблицы можно оформить как функцию: он выдает единственный нетабличный результат — сумму.

Функции удобнее в использовании, так как позволяют не заводить лишних переменных и сокращают запись. Например, если у нас имеется вспомогательный алго-

ритм определения большего из двух чисел в варианте нефункции и в варианте функции с заготовками соответственно:

алг БИД (арг вещ a, b, рез вещ m)

и

алг вещ бид (вещ a, b)

то для вычисления максимума из трех нам придется писать

либо БИД (a, b, m); БИД (m, c, m);

либо m := бид (бид (a, b), c).

Это и является основным достоинством функций, но, к сожалению, не каждый алгоритм может быть оформлен как функция.

Поскольку у функции единственный параметр-результат, то роль этого параметра берет на себя имя функции. Описатель типа, стоящий перед именем функции, относится как раз к этому имени (а также к переменной с именем знач) и показывает, какого типа значения может принимать это имя. Поскольку имя функции взяло на себя роль параметра-результата, то в списке параметров остается на один параметр меньше, а так как все оставшиеся параметры — аргументы, то нет необходимости в списке аргументов: и так ясно, что все параметры являются аргументами (хотя авторы [4] и оставили слово арг в функциях, но в нем нет необходимости). Так как имя функции получает значение, то обращение к функции можно использовать в выражениях на правах простой переменной. Итак, чтобы превратить алгоритм в функцию, надо:

а) описатель типа результата поставить перед именем алгоритма;

б) убрать переменную-результат из списка параметров, а в теле алгоритма заменить ее всюду на знач,

в) убрать список аргументов и список результатов (или слово арг, если изложение дается по [4]) (см. рис. 1 а, б, в).

Собственно реализация функций на ПМК не вызывает никаких проблем: функции реализуются по общим правилам подпрограмм,

79

алг БИД (вещ a, b, (вещ m))

арг a, b; рез m

(алг БИД (арг вещ a, b, рез (вещ m)))

кон если a > b то m = a
иначе m = b

все

кон

а).

алг вещ БИД (вещ a, b, m)

арг a, b; рез m

(алг вещ БИД (арг вещ a, b, рез m))

кон если a > b то m := a
иначе m := b

все

кон

б).

алг вещ БИД (вещ a, b).

кон если a > b то знач := a
иначе знач := b

все

кон

в).

единственный результат при засылке в стек оказывается в регистре X и может быть использован не только для засылки куда-то, но и непосредственно в арифметических операциях в точном соответствии с тем, как обычно используют результат функций. Однако особенности ПМК накладывают некоторые ограничения на использование функций:

а) при засылке в стек аргументов многоаргументных функций можно вытолкнуть из стека какой-либо полезный промежуточный результат;

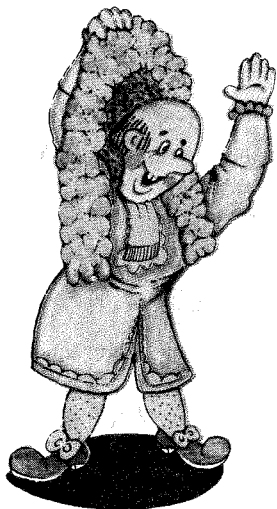
б) для уверенного программирования выражений, содержащих обращения к функциям, необходимо, чтобы функции *корректно использовали стек*, т. е. не оставляли в нем ничего лишнего (подробнее об этом термине см. [3], с. 117).

Чтобы не усложнять материал этими деталями, проще потребовать, чтобы *обращение к функции было всегда первым членом выражения*.

В качестве задач для рассмотрения на уроке и домашних заданий годятся почти все задачи учебника, в которых использованы функции.

Финальным аккордом темы «Вспомогательные алгоритмы» целесообразно провести урок по схеме «цепочка задач», иллюстрирующий, как облегчает работу наличие библиотеки подпрограмм. В качестве цепочки задач можно взять, например, такие (см. [2], часть 2, с. 52):

- 1) определить а) последнюю, б) предпоследнюю, в) К-ю цифру целого числа;
- 2) определить сумму цифр трехзначного числа;
- 3) определить, является ли «счастливым» билет с данным номером (можно написать и алгоритм подсчета числа «счастливых»



билетов, но он невыполним за разумное время на ПМК).

Задачи 1а и 1б — вводные. Общий случай дает задача 1в. С использованием рассмотренного алгоритма *нацело* алгоритм *цифраК* имеет вид

алг цел *цифраК* (цел число, к);

нач цел *часть*, *ост*;

нацело (число, 10^k , *часть*, *ост*);

нацело (*ост*, 10^{k-1} , знач, *ост*);

кон

Теперь, имея алгоритм *цифраК*, легко получить алгоритм *суммацифр*:

нач знач := *цифраК* (число, 1) + *цифраК* (число, 2) + *цифраК* (число, 3)

Итак, в нашей библиотеке алгоритмов есть и алгоритм *нацело* и алгоритм *суммацифр*. С их помощью проверка «счастья» билета выполняется очень просто: *нацело* (номер, 1000, *циф*123, *циф*456); *если* *суммацифр* (*циф*123) - *суммацифр* (*циф*456) *то*...

Программы получаются не самые оптимальные по скорости, но отлично показывают, как «набирается» алгоритм из крупных блоков вместо его построения из кирпичиков-команд.

Рассмотрев эти алгоритмы в классе и записав в виде программ для ПМК в домашних заданиях, на последнем уроке вводим алгоритм *нацело* в дальние адреса памяти (скажем, с адреса 80), а с адреса, например, 60 пишем головную программу — алгоритм *цифраК*. Убедившись, что все работает, заменяем «С/П» в программе *цифраК* на «В/О», превращая ее в подпрограмму, и пишем с адреса 40 программу *суммацифр* и т. д. (Заметим только, что 10^k необходимо вычислять циклом умножения, ибо использование команды «F 10^x » дает большую погрешность.)

Литература

1. Шень А. Информатика в 9 классе // Информатика и образование. 1987. № 6. С. 5—9.
2. Основы информатики и вычислительной техники / Под ред. А. П. Ершова и В. М. Монахова. М.: Просвещение, 1985.
3. Штернберг Л. Ф. Программирование на микрокалькуляторе. М.: Просвещение, 1988.
4. Основы информатики и вычислительной техники / Под ред. А. П. Ершова. М.: Просвещение, 1988.

От редакции. Уважаемые читатели! Этой статьей мы заканчиваем цикл «Уроки с ПМК». Автору и редакции хотелось бы получить от вас отзывы на эти публикации: понравились ли они вам,годились ли в работе, нужно ли в дальнейшем освещать эту тематику на страницах журнала? Ждем ваших писем!