микрокомпьютер

«ЭЛЕКТРОНИКА МК 85»

«ЭЛЕКТРОНИКА МК 85М»

MK85.NAROD.RU

РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

ВНИМАНИЕ!

Если при установке выключателя питания в положение  ▸ компьютер не включился, необходимо выключить его, подождать 2-3 секунды и включить снова.

УВАЖАЕМЫЙ ТОВАРИЩ!

Вы являетесь обладателем первого отечественного микрокомпьютера карманного типа "Электроника МК 85" ("Электроника МК 85М"). Это электронно-вычислительная машина миниатюрных размеров индивидуального пользования, выполненная на базе однокристального микропроцессора и комплекта интерфейсных КМДП-микросхем, в которых реализованы функции устройств ввода-вывода, оперативной и постоянной памяти.

Бережное обращение с компьютером и соблюдение требований руководства по эксплуатации обеспечат его надежную работу на протяжении длительного времени.

Мы уверены, что микрокомпьютер будет Вашим надежным помощником при выполнении научных, инженерных и статистических расчетов.

Инженеры-предприятия-изготовителя ведут постоянную работу по улучшению технических и эксплуатационных характеристик компьютера и заранее выражают свою признательность за Ваш отзыв о его работе.

Заполненный листок предложений просим выслать по адресу: 103482, Москва, абонентный ящик № 123.

ВНИМАНИЕ!

При покупке компьютера требуйте проверки его работоспособности путем демонстрации выполнения теста самоконтроля. Чтобы перевести компьютер в режим самоконтроля, нужно набрать на клавиатуре слово TEST и нажать клавишу [EXE]. В случае его полной работоспособности на индикаторе появятся символы RUN, DEG и сообщение READY P0.

Следует учитывать, что выполнение теста самоконтроля приводит к стиранию записанных ранее программ.

Проверьте комплектность компьютера и сохранность пломбы, наличие гарантийного и отрывных талонов в руководстве по эксплуатации, убедитесь, что в них проставлены штамп продавца или его разборчивая подпись, штамп магазина и дата продажи.

Помните, что в случае утери гарантийного талона Вы лишаетесь права на гарантийный ремонт.

Отрывные талоны изымаются работниками обслуживающей организации только после того, как работа по ремонту фактически выполнена.

Для обеспечения длительной надежной работы следует избегать эксплуатации или хранения компьютера в местах, подверженных резкой смене температур, повышенной влажности, не допускать его нагрева от попадания прямых солнечных лучей, беречь от механических повреждений.

Запрещается протирать панель компьютера органическими растворителями и спиртосодержащими жидкостями.

После хранения компьютера в холодном помещении или транспортировки в зимних условиях перед эксплуатацией его следует выдержать при комнатной температуре в течение двух часов.

Прежде чем приступить к работе с компьютером, изучите данное руководство по эксплуатации.

1. КОМПЛЕКТ ПОСТАВКИ

1. Микрокомпьютер "Электроника МК 85" ("Электроника МК 85М")	1 шт.
2. Руководство по эксплуатации	1 шт.
	1 шт.
3. Футляр	1 шт.
4. Упаковочная тара	1 шт.
5. Элементы питания типа СЦ 0,18 (установлены в компьютере)	1 компл.
6. Блок питания "Электроника Д2-10К" ("Электроника Д2-37В")	1 шт.
7. Паспорт на блок питания	1 шт.
8. Накладка	1 шт.

2. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Система счисления при вводе и выводе информации	десятичная
Количество разрядов мантиссы числа	10
Количество разрядов порядка числа	4
Диапазон представления чисел	$\pm 10^{-4095}$ ... $\pm 9{,}999999999 \times 10^{4094}$
Количество адресуемых регистров памяти	26 с возможностью расширения до 178 (690) *
Объем энергонезависимой памяти, байт	2К (1221 шаг программы) 6К (5317 шагов программы) *
Количество одновременно хранимых программ (файлов)	10 (P0 ... P9)
Язык программирования	БЭЙСИК
Режимы работы	основной, калькуляторный, записи, отладки, совмещенных функций *S*, *F*, расширения функциональных возможностей, повышенного быстродействия
Индикатор	жидкокристаллический матричный 12-разрядный
Электропитание	от 4 элементов типа СЦ 0,18 (или от блока питания)
Потребляемая мощность, Вт, не более	0,02
Диапазон рабочих температур, °С	+ 5 ... + 40
Габаритные размеры, мм	13 x 166 x 73
Масса, кг, не более	0,15
Содержание драгоценных материалов, г:	
золото	0,16533 (0,17591) *
серебро	3,28

* Характеристики приведены для компьютера "Электроника МК 85М"

Допустимые диапазоны задания чисел и точность вычислений

Компьютер выполняет все операции с максимальной ошибкой ± 1 в десятом разряде числа. Допустимые диапазоны задания чисел, обеспечивающие указанную точность вычислений, приведены ниже.

Обозначение функции	Диапазон аргументов
$\sin x$, $\cos x$, $\tan x$	определен диапазоном представленных чисел
$\sin^{-1} x$, $\cos^{-1} x$	$\lvert x\rvert \leqslant 1$
$\tan^{-1} x$	$\lvert x\rvert < 10^{к*}$
$\log x$, $\ln x$	$0 < x < 10^{к}$
e^{x}	$-9429 \leqslant x \leqslant 9429$
$\sqrt{x}$	$0 \leqslant x \leqslant 10^{к}$
x^{y} $(x \uparrow y)$	$\lvert x\rvert < 10^{к}$ при $x < 0$, y — целое число

*к = 4094

Энергонезависимая память

Программы в энергонезависимой памяти сохраняются при установленных в компьютере элементах питания как при работе от блока питания, так и в автономном режиме. При замене элементов питания на новые программы сохраняются в течение 15 минут с момента начала замены.

3. УКАЗАНИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ

Внимание! При работе компьютера от сети через блок питания будьте осторожны! Запрещается вскрывать компьютер и производить ремонтные и наладочные работы при подключенном блоке питания.

4. КРАТКОЕ ОПИСАНИЕ КОМПЬЮТЕРА

В настоящем разделе Вы найдете все необходимые сведения для работы с компьютером "Электроника МК 85" ("Электроника МК 85М" работает аналогично), узнаете о его возможностях и познакомитесь с основами программирования на языке БЭЙСИК.

Микрокомпьютер предназначен для выполнения научных, инженерных и статистических расчетов и решения задач с помощью программ, составленных на языке БЭЙСИК.

Компьютер автоматически выполняет ранее записанную программу, четыре арифметических действия, вычисления прямых и обратных тригонометрических функций, десятичных и натуральных логарифмов, экспоненциальной функции, квадратного корня, абсолютной величины, определение знака числа, целой и дробной части числа, генерацию случайных чисел.

Внешний вид компьютера приведен на рис. 1, схема электрическая принципиальная — в приложении1.

Ввод данных и команд в компьютер осуществляется с помощью клавиатуры.

Наличие клавиш совмещенных функций [S] и [F], а также клавиши [MODE] позволяет использовать клавиши для выполнения нескольких операций (до семи). Обозначения совмещенных функций и символов располагаются вверху, внизу, справа от клавиши и на накладке.

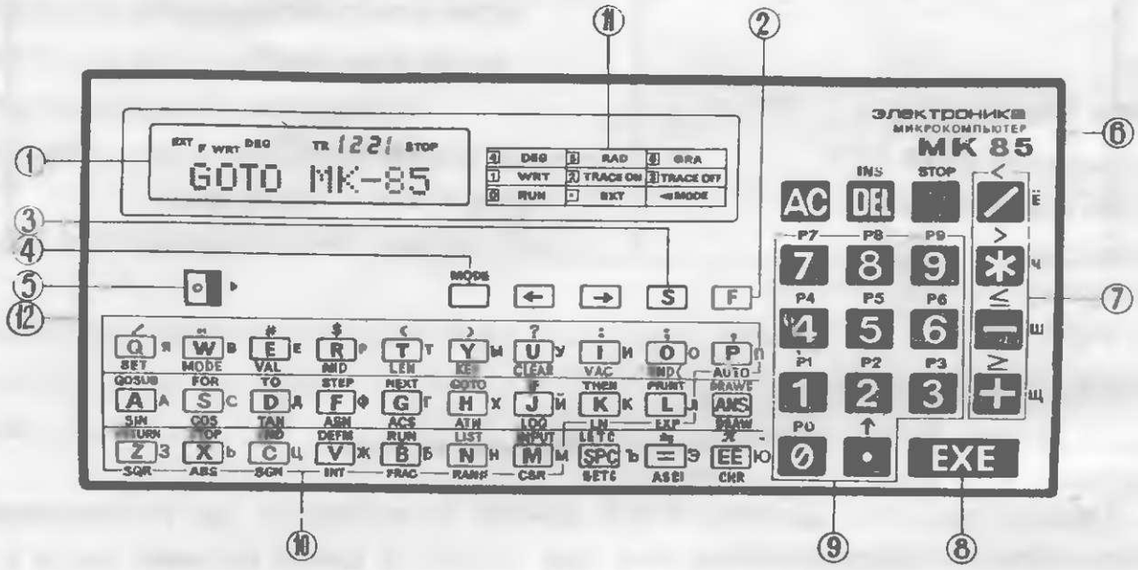
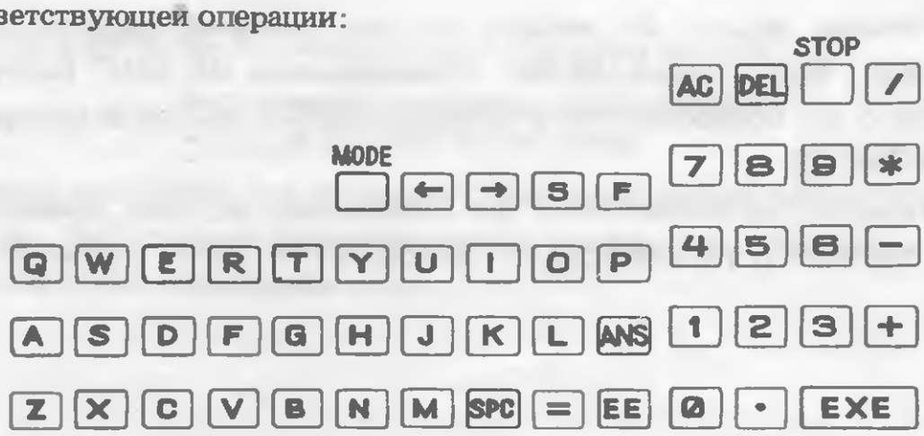


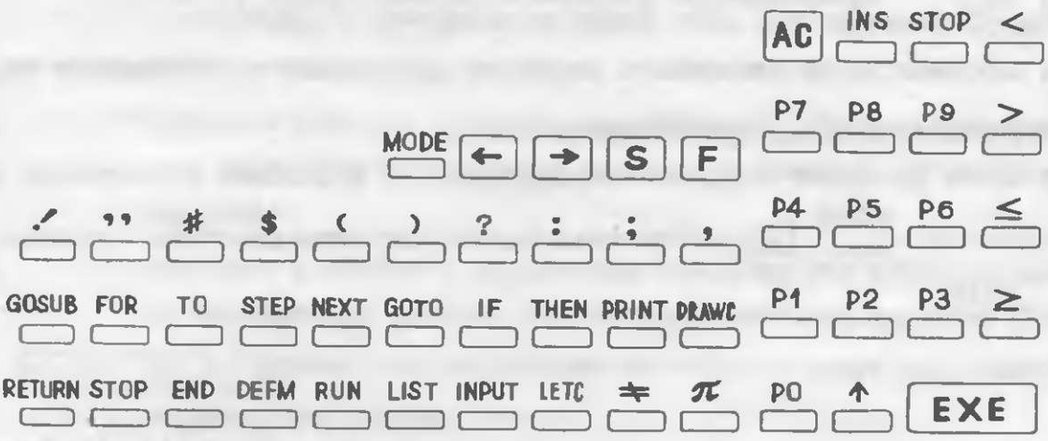
Рис. 1. Внешний вид компьютера:

1 — индикатор; 2 — клавиша совмещенной функции [F]; 3 — клавиша совмещенной функции [S]; 4 — клавиша выбора режима работы [MODE]; 5 — выключатель питания; 6 — регулятор контрастности индикатора; 7 — клавиши арифметических операций; 8 — клавиша выполнения операций; 9 — цифровые клавиши и клавиша десятичной запятой; 10 — клавиши ввода алфавита; 11 — таблица режимов работы компьютера; 12 — разъем для подключения блока питания.

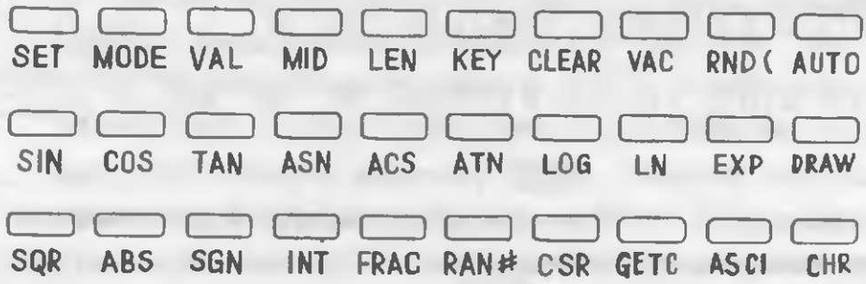
При нажатии клавиши происходит ввод нанесенного на нее символа или отработка соответствующей операции:



При нажатии клавиши [S] и одной из клавиш осуществляется ввод операторов, обозначенных на панели красным цветом, или обращение к файлам PØ ... P9 :



При нажатии клавиши [F] и одной из клавиш осуществляется ввод функций и операторов, обозначенных на панели синим цветом:



Клавиша [MODE] используется для:

- выбора режима работы компьютера и формы представления аргумента тригонометрических функций и угловых величин;
- ввода строчных букв латинского алфавита;
- ввода строчных и прописных букв русского алфавита.

При нажатии клавиши [MODE] на индикаторе одновременно появляются символы S и F.

Выбор режима работы компьютера осуществляется последовательным (рис. 1, поз. 11) нажатием клавиш. При этом на индикаторе отображаются соответствующие символы:

[MODE] [0] — режим исходного состояния и автоматического выполнения операций (RUN).

[MODE] [1] — режим записи, проверки и редактирования программы (WRT).

[MODE] [2] — режим отладки программы (TR).

[MODE] [3] — выход из режима отладки.

[MODE] [.] — режим расширения функциональных возможностей (EXT).

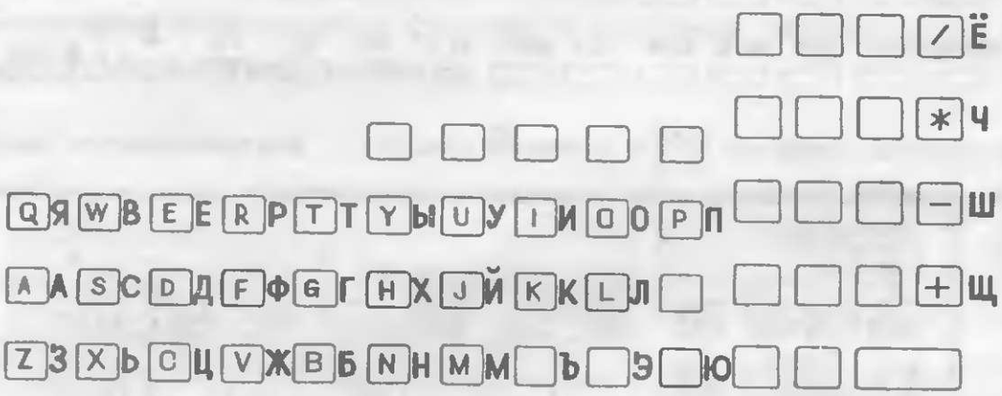
[MODE] [4] — представление угловых величин в градусах (DEG).

MODE [5] — представление угловых величин в радианах (RAD).

MODE [6] — представление угловых величин в градах (GRA).

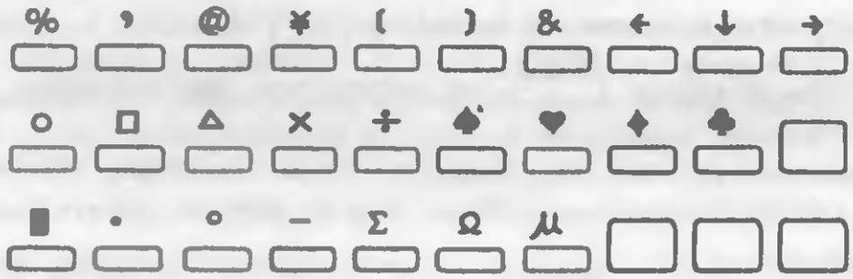
Ввод строчных букв латинского алфавита происходит после нажатия клавиши MODE и соответствующей буквенной клавиши.

Ввод букв русского алфавита осуществляется в режиме расширения (после нажатия клавиш MODE [·]). Строчные буквы вводятся нажатием клавиш [Q] Я ... [M] М :



Если перед нажатием одной из этих клавиш будет нажата клавиша MODE , то вводится соответствующая прописная буква. Строчные буквы *ю, э, ъ, ё, ч, ш, щ* вводятся в режиме расширения, но при предварительном нажатии клавиши MODE . Ввод соответствующей прописной буквы осуществляется аналогично, но в основном режиме.

В режиме расширения при нажатии клавиши [S] и одной из клавиш [A] ... [Z] вводятся символы, обозначенные на накладке :



При нажатии клавиши [F] и одной из этих клавиш в режиме расширения вводятся прописные буквы латинского алфавита.

Более подробно назначение каждой клавиши компьютера будет описано в соответствующих разделах настоящего руководства. Ниже дается лишь краткая их характеристика.

[←] , [→] — сдвиг курсора. При однократном нажатии клавиши курсор сдвигается в соответствующем направлении на одну позицию.

Если клавишу держать нажатой более 1 секунды, происходит автоматическое передвижение курсора. То же относится и к любой другой клавише. Автоматический ввод символа (оператора команды и т. д.) будет осуществляться, если клавиша нажата более 1 секунды, и прекратится после того, как клавиша будет отпущена.

[AC] — клавиша очистки регистра индикации и рабочего регистра (может использоваться и для выхода из режима сигнализации об ошибке).

INS
[DEL] — клавиша удаления с индикатора символа, под которым находится мерцающий курсор. При предварительном нажатии клавиши [S] служит для выделения свободного места для записи символа внутри строки.

STOP
[ ] — клавиша останова программы. При нажатии клавиши процесс выполнения программы приостановится, и на индикаторе появятся символ STOP и диагностическое сообщение Pn — M, где n — номер программы, а M — номер строки, которая отрабатывалась в момент нажатия клавиши [STOP].

В этом режиме можно выполнять любые промежуточные вычисления. Чтобы продолжить вычисления по программе, следует снова нажать клавишу [EXE]. Нажатие клавиши [STOP] во время отработки оператора вывода PRINT приводит к останову "бегущей" строки. Для выполнения промежуточных вычислений следует нажать клавиши [MODE] [EXE]. При этом на индикаторе появится символ STOP. Для продолжения работы программы необходимо нажать клавишу [EXE].

[EXE] — клавиша выполнения операций. В режиме вычислений используется для получения результата, в режиме WRT — для записи строк программы в память компьютера, в режиме RUN — для ввода исходных данных. Может использоваться и для возобновления вычислений в случае останова.

DRAWC
[ANS]
DRAW — клавиша вывода на индикатор результата предыдущего вычисления. В режиме сообщения об ошибках при нажатии клавиши на индикатор выводится строка, в которой была допущена ошибка. Мерцающий курсор при этом указывает на ее конкретное место. Клавиша может применяться также в случае псевдографического использования индикатора (операторы DRAW и DRAWC).

π
[EE] Ю
CHR — клавиша ввода порядка числа (при записи его в экспоненциальной форме) русской буквы Ю и символа числа π, которое может быть использовано в вычислениях в неявном виде. Для получения числового значения π следует нажать клавишу [EXE] после

ввода символа π. Клавиша используется также для преобразования целого числа в текстовый символ.

≠
[=] Э — клавиша определения числовых значений переменных, ввода русской буквы Э и символа ≠. Кроме того, используется в операторе условного перехода IF и служит для преобразования текстового формата в числовой.
ASCI

P7 P8 P9
[7] [8] [9]
P4 P5 P6
[4] [5] [6]
P1 P2 P3
[1] [2] [3]
P0 ↑
[0] [·] — цифровые клавиши и клавиша десятичной запятой. При нажатии клавиши [S] служат для обозначения номера (имени) программы или подпрограммы. Клавиша [↑ ·] в этом случае используется для возведения в степень.

≥ ≤
[+] Щ, [−] Ш,
> <
[*] Ч, [/] Ё — клавиши арифметических операций, ввода русских букв Щ, Ш, Ч, Ё и символов ≥, ≤, >, <, используемых в операторе условного перехода IF.

LETC
[SPC] Ъ — клавиша пробела, ввода русской буквы Ъ и операторов, работающих с текстовыми переменными.
GETC

Контроль ввода данных и результатов вычислений производится визуально с помощью 12-разрядного жидкокристаллического индикатора матричного типа. На индикаторе отображается весь набор данных и результатов, символы, определяющие режим, в котором находится компьютер, и число свободных шагов программы:



Если разрядность вводимых в компьютер данных превышает 12, знаки смещаются влево. В строку можно ввести 63 символа. После ввода 57-го знака в крайнем правом разряде индикатора появляется предупредительный мерцающий символ " ▮ ".

Число оставшихся шагов программы (в режиме WRT) индицируется в верхней части индикатора. Для этой цели служат четыре сегментных разряда. Во время отработки компьютером той или иной программы крайний правый из этих разрядов служит для сигнализации о том, что идет счет. В этот момент там появляется символ Р.

Регулятор контрастности

С правой стороны компьютера находится регулятор контрастности, который предназначен для регулирования напряжения, подаваемого на ЖКИ. В зависимости от угла обзора следует устанавливать оптимальную для считывания информации контрастность (при вращении ручки регулятора вниз контрастность уменьшается).

5. ПОДГОТОВКА КОМПЬЮТЕРА К РАБОТЕ

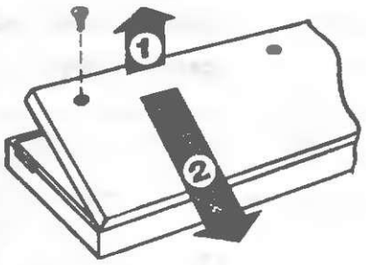
5.1. Работа от автономного источника питания

Микрокомпьютер "Электроника МК 85" поставляется с четырьмя элементами питания СЦ 0,18, установленными в компьютере. Для включения компьютера выключатель питания следует установить в положение  . На индикаторе должны появиться символы RUN и DEG, а также сообщение READY PØ, свидетельствующее о готовности компьютера к работе.

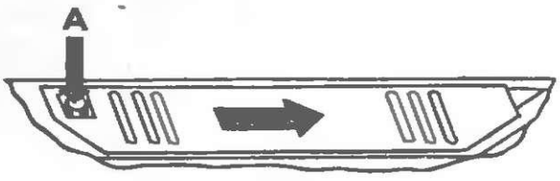
Слабое свечение индикатора при крайнем верхнем положении регулятора контрастности указывает на разряд элементов питания.

В период гарантийного срока замену элементов следует производить в гарантийных мастерских. Замену элементов производить в следующем порядке:

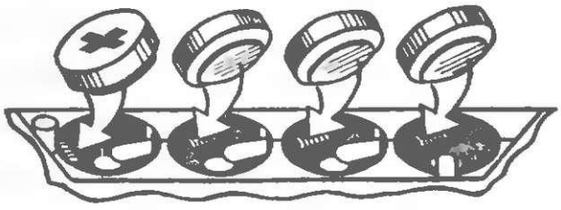
- *удалить пломбу, отвинтить винты на задней крышке и снять ее*



- *нажать в точке А и снять крышку отсека питания, сдвинув ее вправо*



- *извлечь элементы и установить новые, соблюдая полярность*



Сборку компьютера проводить в обратном порядке.

Внимание! После замены элементов питания в случае разрушения хранимых программ необходимо включить компьютер и произвести начальную установку, нажав острым непроводящим предметом кнопку "начальная установка", находящуюся справа внизу на задней крышке (нажатие кнопки — кратковременное).

5.2. Работа от блока питания

При работе компьютера от блока питания следует вставить вилку шнура блока питания в гнездо компьютера, а затем вилку блока питания — в розетку сети переменного тока напряжением 220 В. Установить выключатель питания компьютера в положение [o|▶] . Появление на индикаторе символов RUN и DEG и сообщения READY P0 свидетельствует о готовности компьютера к работе.

Для ускорения процесса вычислений в компьютере реализована возможность повышения быстродействия (за счет увеличения тактовой частоты). Для работы в этом режиме включение компьютера должно осуществляться при нажатой клавише [+] . Во избежание быстрого разряда элементов питания компьютер в режиме повышенного быстродействия рекомендуется включать через блок питания.

Внимание! После окончания работы необходимо отключить блок питания от сети и от компьютера.

6. ПОРЯДОК РАБОТЫ С КОМПЬЮТЕРОМ

6.1. Общие сведения

Прежде чем приступить к изучению программного обеспечения микрокомпьютера, необходимо усвоить ряд моментов, без учета которых работа с компьютером невоэможна.

Организация и расширение памяти

В исходном состоянии микрокомпьютер имеет 26 ячеек адресуемой памяти, обозначаемых буквами латинского алфавита, и 1221 (5317) шаг программы. В случае необходимости в процессе работы с компьютером число ячеек памяти можно увеличить. Следует, однако, учитывать, что добавление каждой новой ячейки памяти сокращает число шагов программы на 8. При максимальном числе ячеек памяти число шагов программы равно 5.

Число ячеек памяти		Число шагов программы	
"Электроника МК 85"	"Электроника МК 85М"	"Электроника МК 85"	"Электроника МК 85М"
26	26	1221	5317
27	27	1213	5309
. . .	. . .	. . .	. . .
46	46	1061	5157
. . .	. . .	. . .	. . .
100	100	629	4725
. . .	. . .	. . .	. . .
150	150	229	4325
. . .	. . .	. . .	. . .
178	690	5	5

Расширение памяти производится при помощи оператора DEFM. Например, для получения 46 ячеек памяти необходимо к исходным 26 добавить 20: DEFM 20 [EXE] . При этом оператор DEFM может быть введен как побуквенно ([D] [E] [F] [M]), так и с помощью клавиши совмещенной функции [S] DEFM ([S] [ ]). К исходному числу ячеек памяти (26) можно вернуться с помощью оператора DEFM 0. Оператор DEFM можно использовать не только для изменения числа ячеек, но и для определения количества ячеек, с которыми можно работать в данный момент:

DEFM [EXE] | * * * VAR : 46 | *

В случае, если при попытке осуществить расширение памяти число свободных шагов программы окажется недостаточным, на индикаторе появится диагностическое сообщение об ошибке — ERR 1.

При работе с компьютером можно использовать одномерные массивы, так как память имеет следующую организацию:

A = A(0)
B = A(1) = B(0)
C = A(2) = B(1) = C(0)
D = A(3) = B(2) = C(1) = D(0)
E = A(4) = B(3) = C(2) = D(1) = E(0)
.
.
.
Y = A(24) = B(23) = C(22) = ... = Y(0)
Z = A(25) = B(24) = C(23) = ... = Y(1) = Z(0)

При расширении памяти имеют место соотношения:

DEFM 1: A(26) = B(25) = C(24) = ... = Y(2) = Z(1)
DEFM 2: A(27) = B(26) = C(25) = ... = Y(3) = Z(2)
.
.
.
DEFM 152 : A(177) = B(176) = C(175) = ... = Y(153) = Z(152)

Видно, что дополнительные ячейки памяти, привлеченные в режиме расширения оператором DEFM, имеют "имена", состоящие из буквы латинского алфавита и заключенного в скобки числа. К исходным 26 ячейкам памяти можно обращаться как непосредственно (A ... Z), так и используя индексное их обозначение. При этом, однако, следует обращать внимание на используемые в одной программе или выражении ячейки памяти. Например, нельзя применять одновременно для записи различных данных ячейки F, G, A(5), A(6), так как F = A(5), G = A(6).

* Здесь и далее в прямоугольнике приведены показания индикатора.

Приоритет операций

Если математическое выражение содержит несколько операций, выполняться они будут с учетом приоритета:

1. Вычисление функции (sin, cos и т. д.).

2 Возведение в степень.

3. Умножение и деление.

4. Сложение и вычитание.

При наличии в выражении скобок в первую очередь выполняются действия, заключенные в эти скобки.

Пример. $2 + 3 * (\sin(12 + 18))^2 = 2{,}75$

1

2

3

4

5

Вычисление выражений, имеющих одинаковый приоритет, компьютер производит слева направо.

Разрядность чисел

Все вычисления, включая и промежуточные, компьютер производит, оперируя числами с 12-разрядной мантиссой и 4-разрядным порядком. Результаты вычислений выдаются на индикатор с 10-разрядной мантиссой. Если результат получен в экспоненциальной форме, мантисса будет иметь 8 разрядов.

Основные операции, выполняемые компьютером

Все символы, входящие в формулы, по которым производятся вычисления, отображаются на индикаторе. Символы арифметических действий: "+" — сложение, "—" — вычитание, "*" — умножение, "/" — деление.

Выражение 2 + 3 — 4 x 5 ÷ 6 на индикаторе будет выглядеть так: 2 + 3 — 4 * 5/6.

Кроме того, компьютер автоматически выполняет следующие функции:

ФУНКЦИЯ	ОБОЗНАЧЕНИЕ	ИМЯ ФУНКЦИИ	СПОСОБ ВВОДА
Тригонометрические функции	sin x	SIN	F SIN
	cos x	COS	F COS
	tg x	TAN	F TAN
Обратные тригонометрические функции	$\sin^{-1}$ x	ASN	F ASN
	$\cos^{-1}$ x	ACS	F ACS
	tg^{-1} x	ATN	F ATN
Квадратный корень	$\sqrt{x}$	SQR	F SQR
Натуральный логарифм	ln x	LN	F LN

Десятичный логарифм	lg x	LOG	F LOG
Экспоненциальная функция	e^x	EXP	F EXP
Модуль числа	| x |	ABS	F ABS
Целая часть числа	INT x	INT	F INT
Дробная часть числа	FRAC x	FRAC	F FRAC
Знак числа	положительное число 1 ноль 0 отрицательное число −1	SGN	F SGN
Округление числа	RND(x, y)	RND(...,...)	F RND(
Генерация случайного числа	RAN	RAN #	F RAN#

Вызов результата предыдущего вычисления

При выполнении вычислений на компьютере очень часто требуется результат предыдущего вычисления. Для этого используется клавиша [ANS] .

Пример. 741 + 852 = 1593

2341 − 1593 = 748

741 [+] 852	741 + 852
[EXE]	1593
2341 [−] [ANS]	2341 − 1593
[EXE]	748

Если результат предыдущего вычисления, полученный на индикаторе, необходимо использовать в следующем вычислении, то выполнить это можно непосредственным вводом нужной операции:

748 x 2 = 1496

[*] 2	748 * 2
[EXE]	1496

Сообщения об ошибках

Если при работе с компьютером была допущена ошибка (превысили диапазон представления чисел, пытались извлечь корень из отрицательного числа и т.д.), на индикаторе появится сообщение:

ERR2	синтаксическая ошибка	при вычислениях в автоматическом режиме или при отработке программы
ERR3	математическая ошибка	
ERR2PØ − 1Ø		

Такое сообщение читается следующим образом: синтаксическая ошибка в 10-й строке программы, записанной в файле PØ.

Более подробно типы ошибок и способы их устранения изложены ниже.

Способы ввода информации

Перед началом работы с компьютером необходимо перевести выключатель питания в положение "▶" (при работе с блоком питания сначала подключить блок). На индикаторе появится сообщение READY PØ, сигнализирующее о том, что компьютер готов к работе.

Ввод информации

Примеры.

1. Ввести ABC:

[A] [B] [C] → A B C

2. Ввести SIN:

[S] [I] [N] (или [F] [ ] (SIN)) → S I N

3. Ввести 12,3:

[1] [2] [·] [3] → 1 2 . 3

4. Ввести ? # :

[S] [?] [S] [#] → ? #

5. Ввести $7{,}89 \times 10^{15}$:

[7] [·] [8] [9] [EE] [1] [5] → 7.89 E 15

6. Ввести $-2{,}345 \times 10^{-28}$:

[−] [2] [·] [3] [4] [5] [EE] [−] [2] [8] → −2.345 E⁻ 28

7. Ввести "П р и м е р ":

MODE [ ] MODE [ ] [P] п [R] р [I] и [M] м [E] е [R] р → Пример

Исправление введенной информации

Примеры.

1. Заменить A$ на B$:

на индикаторе (курсор) → A$

При помощи клавиши [←] переместить курсор под символ A:

[←] [←] → A$

Нажать клавишу нужного символа:

[B] → B$

2. Заменить LIST на RUN:

LIST

[←] [←] [←] [←] → LIST

[R] [U] [N] [SPC] (или [S] [ ] (RUN)) → RUN

3. Удалить ошибочно введенный символ.

Для удаления используется клавиша [DEL]. Пусть необходимо устранить символ O в COOS:

	COOS
[←] [←]	COOS
[DEL]	COS

4. Удалить X, в тексте INPUT X, Y:

	INPUT X, Y
[←] [←] [←]	INPUT X, Y
[DEL] [DEL]	INPUT Y

5. Вставить пропущенный символ.

Для этого используется клавиша [INS]. Пусть необходимо вместо T = A$ записать T$ = A$:

	T = A$
[←] [←] [←]	T = A$
INS [S]	T_ = A$
$ [S]	T$ = A$

6. Заменить PRINT X на PRINT SIN X:

	PRINT X_
[←]	PRINT X
INS [S] INS [S] INS [S]	PRINT _ X
[S] [I] [N]	PRINT SIN X

6.2. Использование компьютера в калькуляторном режиме

Микрокомпьютер "Электроника МК 85" может быть использован в калькуляторном режиме.

Примеры вычислений

1. Арифметические действия

1) 23 + 4,5 − 53 = − 25,5

23 [+] 4 [·] 5 [−] 53 [EXE] | −25.5

2) 56 x (− 12) ÷ (− 2,5) = 268,8*

56 [*] [S] [(] [−] 12 [S] [)] [/] [S] [(] [−] 2 [·] 5 [S] [)] [EXE] | 268.8

3) 12369 x 7532 x 74103 = 6,9036806 x 10^{12} (= 6903680600000)

12369 [*] 7532 [*] 741Ø3 [EXE] | 6.9Ø368Ø6E12

* *В примерах такого типа скобки могут быть опущены.*

4) $1{,}23 \div 90 \div 45{,}6 = 2{,}997076 \times 10^{-4}$ (= 0,0002997076)

1 [·] 23 [/] 9Ø [/] 45 [·] 6 [EXE]	2.997Ø76Ø2 E⁻ 4

(если результат вычислений больше 10^9 или меньше 10^{-3}, на индикаторе он появится в экспоненциальной форме).

5) $12 + (2{,}4 \times 10^5) \div 42{,}6 - 78 \times 36{,}9 = 2767{,}602817$

12 [+] [S] [(] 2 [·] 4 [EE] 5 [S] [)] [/] 42 [·] 6 [−] 78 [*]	
36 [·] 9 [EXE]	2767.6Ø2817

2. Операции с использованием ячеек памяти

1) 12 x 45 = 540
12 x 31 = 372
75 ÷ 12 = 6,25

A [=] 12 [EXE]	_
A [*] 45 [EXE]	54Ø
A [*] 31 [EXE]	372
75 [/] A [EXE]	6.25

2) 23 + 9 = 32
53 − 6 = 47
−) 45 x 2 = 90
99 ÷ 3 = 33
Итого: 22

M [=] 23 [+] 9 [EXE]	
M [=] M [+] 53 [−] 6 [EXE]	
M [=] M [−] 45 [*] 2 [EXE]	
M [=] M [+] 99 [/] 3 [EXE]	
M [EXE]	22

Для предыдущего примера в случае, если необходимо получить результаты каждой строки, последовательность действий может быть такой:

23 [+] 9 [EXE]	32
M [=] [ANS] [EXE]	_
53 [−] 6 [EXE]	47
M [=] M [+] [ANS] [EXE]	_
45 [*] 2 [EXE]	9Ø
M [=] M [−] [ANS] [EXE]	_
99 [/] 3 [EXE]	33
M [=] M [+] [ANS] [EXE]	_
M [EXE]	22

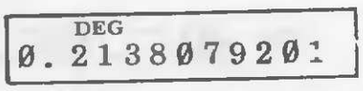
3. Тригонометрические и обратные тригонометрические функции

При вычислениях этих функций следует обращать внимание на то, в каких угловых величинах выражен аргумент.

1) $\sin 12{,}3456° = 0{,}2138079201$

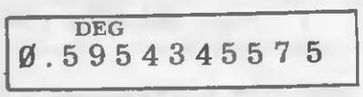
MODE [ ] [4] ⟶ D E G

SIN 12 [·] 3456 [EXE]



2) $2 \times \sin 45° \times \cos 65{,}1° = 0{,}5954345575$

2 [*] SIN 45 [*] COS 65 [·] 1 [EXE]



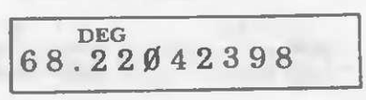
3) $\sin^{-1} 0{,}5 = 30°$

ASN Ø [·] 5 [EXE]



4) $2{,}5 \times (\sin^{-1} 0{,}8 - \cos^{-1} 0{,}9) = 68{,}22042398$

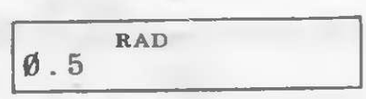
2 [·] 5 [*] [S] [( ] ASN Ø [·] 8 [−] ACS Ø [·] 9 [S] [)] [EXE]



5) $\cos\left(\frac{\pi}{3}\ \text{rad}\right) = 0{,}5$

MODE [ ] [5] ⟶ R A D

COS [S] [( ] [S] [π] [/] 3 [S] [)] [EXE]



6) $\cos^{-1} \frac{\sqrt{2}}{2} = 0{,}7853981634$

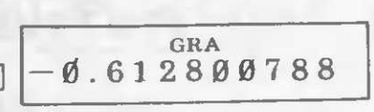
ACS [S] [( ] SQR 2 [/] 2 [S] [)] [EXE]



7) $\text{tg}(-35\ \text{gra}) = -0{,}612800788$

MODE [ ] [6] ⟶ G R A

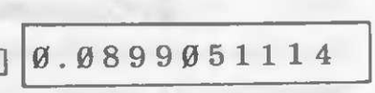
TAN [−] 35 [EXE]



4. Логарифмические и степенные функции

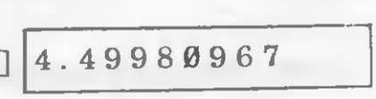
1) $\lg 1{,}23 = 0{,}0899051114$

LOG 1 [·] 23 [EXE]



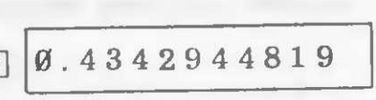
2) $\ln 90 = 4{,}49980967$

LN 9Ø [EXE]



3) $\lg 456 \div \ln 456 = 0{,}4342944819$

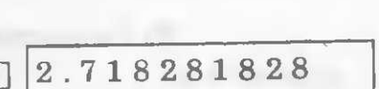
LOG 456 [/] LN 456 [EXE]



4) $e = 2{,}718281828$

(число e)

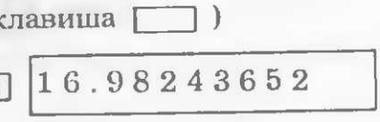
EXP 1 [EXE]



5) $10^{1{,}23} = 16{,}98243652$

(при возведении в степень используется клавиша [↑])

10 [S] [↑] 1 [·] 23 [EXE]



6) $5{,}6^{2{,}3} = 52{,}58143837$

5 [·] 6 [S] [↑] 2 [·] 3 [EXE] | 52.58143837

7) $\sqrt[7]{123}\ (= 123^{1/7}) = 1{,}988647795$

123 [S] [↑] [S] [(] 1 [/] 7 [S] [)] [EXE] | 1.988647795

8) $(78 - 23)^{-12} = 1{,}3051118 \times 10^{-21}$

[S] [(] 78 [−] 23 [S] [)] [S] [↑] [−] 12 [EXE] | 1.3Ø51118 E⁻ 21

9) $2^2 + 3^3 + 4^4 = 287$

2 [S] [↑] 2 [+] 3 [S] [↑] 3 [+] 4 [S] [↑] 4 [EXE] | 287

10) lg sin 40° + lg cos 35° = − 0,278567983

MODE [4] ⟶ D E G

LOG SIN 4Ø [+] LOG COS 35 [EXE] | −Ø.278567983

антилогарифм:

1Ø [S] [↑] ANS [EXE] | Ø.526540784 6

5. Прочие функции

1) $\sqrt{2} + \sqrt{5} = 3{,}65028154$

SQR 2 [+] SQR 5 [EXE] | 3.65Ø28154

2) Определение знака числа

(1 — положительное число, 0 — ноль, — 1 — отрицательное число)

SGN 6 [EXE]	1
SGN Ø [EXE]	Ø
SGN [−] 2 [EXE]	− 1

3) Генерация случайных чисел, равномерно распределенных в интервале от 0 до 1

RAN [S] [#] [EXE] | Ø.9Ø3173418

(или [F] [RAN#])

4) Получение результата с заданной точностью

Пусть необходимо получить результат вычисления 12,3 x 4,56 с округлением второго и последующих знаков после запятой (12,3 x 4,56 = 56,088)

RND [S] (12 · 3 * 4 · 56 [S] , − 2 [S]) EXE | 5 6 . 1 |

5) | − 78,9 ÷ 5,6 | = 14,08928571

ABS [S] (− 78 · 9 / 5 · 6 [S]) EXE | 1 4 . Ø 8 9 2 8 5 7 1 |

6) Определить целую часть отношения $\frac{7800}{96}$ (= 81,25)

INT [S] (78ØØ / 96 [S]) EXE | 8 1 |

7) Определить дробную часть отношения $\frac{7800}{96}$ (= 81,25)

FRAC [S] (78ØØ / 96 [S]) EXE | Ø . 2 5 |

8) Получить приближенное значение выражения с заданным числом разрядов мантиссы

SET 4 EXE

1ØØ / 6 EXE | 1 6 . 6 7 |

1ØØØØØ / 3 EXE | 3 . 3 3 3 E 4 |

SET 6 EXE

123 / 7 EXE | 1 7 . 5 7 1 4 |

SET 1Ø EXE

123 / 7 EXE | 1 7 . 5 7 1 4 2 8 5 7 |

6.3. Основы программирования

Микрокомпьютер "Электроника МК 85" предназначен в основном для решения математических и инженерных задач в режиме диалога человек — ЭВМ. Компьютер работает с языком БЭЙСИК, который позволяет программировать большой круг задач и при этом прост для изучения и понимания. Обозначения, используемые в языке, аналогичны математическим.

При составлении программы на языке БЭЙСИК необходимо учитывать целый ряд правил, которые будут изложены ниже в соответствующих разделах. Рассмотрим принцип построения программы на простом примере.

```
1Ø INPUT A, B      — оператор ввода
2Ø C = A + B       — оператор процедуры
3Ø PRINT C         — оператор вывода
```

операнды

команда

номер строки

Основными элементами любой программы являются:

— оператор ввода, используемый для ввода исходных данных с клавиатуры во время выполнения программы;
— оператор процедуры, дающий компьютеру команду выполнить то или иное действие с введенными данными;
— оператор вывода, используемый для вывода данных или результатов вычислений на индикатор.

Программа, составленная на языке БЭЙСИК, оформляется в виде пронумерованных строк. Выполнение программы осуществляется в порядке возрастания их номеров. Чем сложнее решаемая задача, тем больше программа, однако структура ее всегда остается неизменной.

При записи программы следует учитывать, что для разных элементов программы требуется разное число шагов:

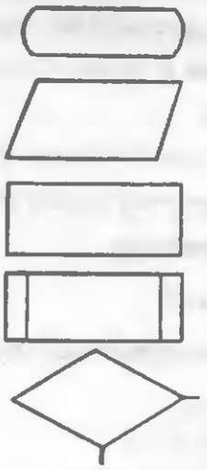
— команда (INPUT, PRINT, RETURN, GOTO и т. д.) — 1 шаг;
— функция (SIN, RAN #, FRAC и т. д.) — 1 шаг;
— номер строки (1, 10, 100 и т. д.) — 2 шага;
— символ (B, ", 1, 7, ; , ↑ и т. д.) — 1 шаг;
— клавиша EXE (обязательно должна быть нажата в конце строки для ввода ее в компьютер) — 1 шаг.

Основные этапы программирования

Процесс составления программы можно разбить на следующие основные этапы:

1) анализ поставленной задачи с разбивкой ее на отдельные элементы;
2) выбор алгоритма решения задачи и изображение его в виде блок-схемы;
3) запись программы на языке БЭЙСИК;
4) отладка и редактирование программы.

При составлении блок-схемы рекомендуется использовать следующие графические обозначения:



— запуск, окончание программы и т. д.

— ввод исходных данных и вывод результатов

— вычисления любого рода

— подпрограмма

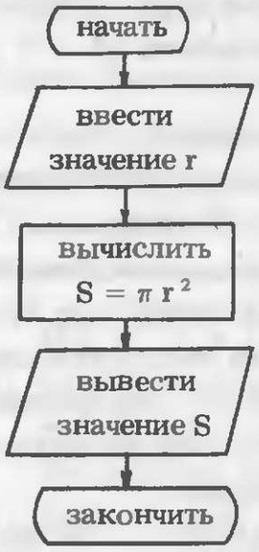
— условный оператор

Рассмотрим этапы программирования более подробно на примере. Пусть необходимо составить программу для нахождения площади круга:



$$S = \pi r^2$$

Блок-схему такой программы можно изобразить в следующем виде:



Естественно, составление блок-схемы для такого простого примера может показаться лишним. Однако при решении более сложных задач этот этап во многом упрощает процесс составления программы.

После составления блок-схемы можно переходить к следующему этапу — записи программы на языке БЭЙСИК. При этом используются определенные рабочие символы (например, "+" — сложение, "—" — вычитание, "/" — деление, " * " — умножение, " ↑ " — возведение в степень). Назначение символа "=" подробно описано ниже, а пока достаточно сказать, что в выражении типа $S = \pi r^2$ он служит для присвоения переменной S численного значения произведения πr^2.

Как уже говорилось, начинать любую программу нужно с оператора ввода. Чаще всего в качестве такого оператора используется INPUT . В нашем примере он будет иметь вид INPUT R . Для использования в программе ему теперь необходимо присвоить числовое значение — номер строки:

```
10 INPUT R
```

В качестве номера строки может быть использовано любое число от 1 до 9999, но следует помнить, что в одной программе каждая строка должна иметь свой номер. Для удобства записи и редактирования программы целесообразно обозначать строки номерами, кратными 10.

В нашем примере в соответствии с блок-схемой под следующим номером должен стоять оператор, производящий вычисление величины πr^2:

```
20 S = π*R ↑ 2
```

Для вывода результата на индикатор используется оператор PRINT :

```
30 PRINT S
```

Таким образом, программа для вычисления площади круга будет иметь вид:

```
10 INPUT R
20 S = π*R↑2
30 PRINT S
```

Переменные и константы

В версии языка БЭЙСИК, реализованной в микрокомпьютере "Электроника МК 85", используются заглавные буквы латинского алфавита (A — Z), цифры (0 — 9) и целый ряд специальных символов.

В программах применяются как постоянные величины (константы), так и переменные. Константа — заранее заданная величина, которая непосредственно записывается в программу. Так, в выражении $S = \pi * R \uparrow 2$ π и 2 — константы.

Переменная — это величина, которой можно присваивать различные значения в процессе выполнения программы. Переменные обозначаются заглавными буквами латинского алфавита (A — Z) или комбинацией одной из этих букв с символом $ (для текстовых переменных). В нашем примере S и R — переменные.

Примечание. В калькуляторном режиме переменные величины выполняют функции памяти, поэтому выше они были названы ячейками памяти.

Переменные могут принимать как различные числовые значения (далее просто переменные), так и состоять из набора символов, не имеющих смысла числового значения (далее текстовые переменные). При записи текстовых переменных символы, входящие в их состав, должны быть заключены в кавычки.

В отличие от переменных, с текстовыми переменными можно проводить только операции сравнения или "склейки" (прибавления).

Пример. Пусть A$ = "123", B$ = "456".
Тогда C$ = A$ + B$ = "123456"
(если C$ = B$ + A$, то C$="456123").

При проведении операций с текстовыми переменными следует учитывать тот факт, что максимальная длина текстовой переменной 7 символов.

Кроме 26 текстовых переменных A$ ... Z$, существует еще дополнительная текстовая переменная, обозначаемая символом $, длина которой может достигать 30 символов.

Пример. $ = "1234567890ABCDEFGHJK"

Операторы присвоения

Для записи информации в микрокомпьютере "Электроника МК 85" используются операторы присвоения, имеющие вид:

ПЕРЕМЕННАЯ = МАТЕМАТИЧЕСКОЕ ВЫРАЖЕНИЕ

Пример. A = 2*B + 3

Такая запись означает, что переменной A присваивается числовое значение выражения 2*B + 3.

Примеры операторов присвоения

1) A = B — присвоение переменной A значения переменной B. Прежнее значение A стирается.
2) N = 2*M — присвоение переменной N удвоенного значения переменной M.

3) X = Y+Z — присвоение переменной X значения суммы переменных Y и Z.

4) I = I+1 — увеличение значения переменной I на 1.

Запись программы

Ввод программы в компьютер осуществляется с помощью клавиатуры пульта управления в следующем порядке:

1) перевести компьютер в режим записи, нажав клавиши MODE [1];

2) выбрать имя файла (номер программы PØ ... P9):

[S] P0 → PØ, или [S] P1 → P1, ... , или [S] P9 → P9;

3) по строкам ввести программу (для занесения строки в компьютер после ее набора необходимо нажать клавишу [EXE]. Следует помнить, что длина строки, включая клавишу [EXE] в конце, не должна превышать 64 символа);

4) после записи программы перевести компьютер в основной режим, нажав клавиши MODE [Ø].

Рассмотрим процедуру записи программы более подробно.

Пример. Записать в файл PØ программу нахождения суммы и разности величин A и B.

```
1Ø INPUT A , B
2Ø V = A + B
3Ø W = A — B
4Ø PRINT V , W
5Ø END
```

1) Перевести компьютер в режим записи WRT:

MODE [1] P Ø 1 2 3 4 5 6 7 8 9 — 1221 — число свободных шагов программы для компьютера "Электроника МК85"

мерцание цифры означает, что запись будет производиться в файл с соответствующим номером (в исходном состоянии это PØ)

В процессе записи программы после ввода очередной строки число свободных шагов уменьшается, сигнализируя о том, какое количество их еще может быть использовано.

2) Выбрать имя файла PØ:

[S] P0 (в нашем примере этот шаг не обязателен, так как "0" уже мерцает).

Если вместо номера программы высвечивается символ " ♦ ", это означает, что соответствующий файл занят. При необходимости можно уничтожить прежнюю программу. Для этого следует обозначить номер программы и нажать клавиши [C] [L] [E] [A] [R] [EXE] (или [F] CLEAR [EXE]).

3) Записать программу в компьютер:

1Ø ␣ INPUT ␣ A [S] [,] B [EXE]	1214 1Ø INPUT A , B
2Ø ␣ V [=] A [+] B [EXE]	1206 2Ø V = A + B
3Ø ␣ W [=] A [−] B [EXE]	1198 3Ø W = A − B
4Ø ␣ PRINT ␣ V [S] [,] W [EXE]	1191 4Ø PRINT V , W
5Ø ␣ END [EXE]	1187 5Ø END

Пробелы в программе имеют чисто утилитарное значение и служат только для облегчения работы с индикатором.

Команда CLEAR уничтожает только одну предварительно обозначенную программу. Для стирания всех ранее записанных программ используется команда CLEAR A .

Выполнение программы

К выполнению программы можно приступить сразу же после ввода ее в компьютер. Запуск программы возможен как из основного режима, так и из режима записи.

Если компьютер находится в режиме записи, то к выполнению программы можно перейти следующим образом:

— очистить индикатор, нажав клавишу [AC] ;

— обозначить номер нужного файла ([S] [P0] ... [S] [P9]);

— набрать RUN (или [S] [RUN]) [EXE] .

После этого компьютер автоматически переходит в основной режим и приступает к выполнению программы.

В основном режиме ([MODE] [Ø]) перейти к выполнению программы можно либо с помощью команды RUN , либо обозначением номера файла ([S] [P0] ... [S] [P9]).

Запуск по команде RUN удобен, если есть необходимость начать выполнение программы не с начала, а с определенного этапа. Для этого следует после команды RUN ввести номер строки, с которой должна выполняться программа, а затем нажать клавишу [EXE] .

Следует иметь в виду, что при переводе компьютера из режима записи в основной режим сообщение вида READY Pn говорит о том, что команда RUN приведет к началу выполнения программы, записанной в файле с номером Pn .

Для нашего примера:

[S] [P0]

```
?
```

знак "?" означает, что компьютер отработал оператор ввода и находится в состоянии ожидания данных

Введем первое число A = 47:

4 7 [EXE]



компьютер "ждет" ввода второго числа

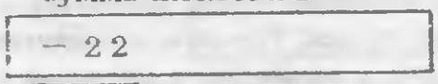
Пусть B = 69:

6 9 [EXE]



сумма чисел A и B

[EXE]



разность чисел A и B

Редактирование программы

Редактирование программы проводится с целью устранения допущенных при записи ошибок. Для вызова текста программы на индикатор используется команда LIST , причем в основном режиме RUN применение этой команды дает возможность лишь просмотреть текст программы, исправления же можно производить только в режиме записи WRT .

Если после команды LIST ввести номер строки, то просмотр программы начнется с указанной строки, например LIST 3Ø [EXE].

В основном режиме по команде LIST компьютер автоматически "пролистывает" всю программу. Если будет нажата клавиша [STOP], автоматическое движение текста прекращается и возобновляется после того, как клавиша [STOP] будет отпущена.

В режиме записи WRT по команде LIST и при нажатии клавиши [EXE] строки будут появляться на индикаторе поочередно, сменяя одна другую в порядке возрастания номеров. Если необходимо просмотреть программу в порядке убывания номеров строк, следует нажимать клавиши [MODE] [EXE] . Разберем процесс редактирования на предыдущем примере.

1. Внесение частичных изменений.

Пусть в строке 20 нашей программы необходимо заменить знак "+" на "✱".

а) перевести компьютер в режим записи WRT :

[MODE] [1]

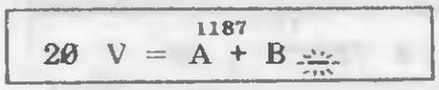


б) обозначить номер программы:

[S] [PØ]

в) вызвать на индикатор строку 20:

LIST 2Ø [EXE]



г) передвинуть курсор на место того символа, который необходимо изменить:

[←] [←]

```
1187
2Ø  V = A ± B
```

(если клавишу передвижения курсора [→] или [←] держать нажатой более одной секунды, его перемещение станет автоматическим).

д) внести изменение:

[*] [EXE]

```
1187
3Ø  W = A − B
```

(для внесения изменения нажатие клавиши [EXE] обязательно). После этого на индикаторе появляется очередная строка. Так как исправлений больше не требуется, следует нажать клавишу [AC] и перевести компьютер в основной режим: MODE [ ] [Ø]. Чтобы убедиться, что изменение внесено, можно просмотреть текст программы: LIST [EXE].

2. Замена строки.

Для того, чтобы заменить какую-либо строку, необходимо под ее номером ввести новую. Старая информация при этом аннулируется.

Пусть нужно заменить строку W = A − B на W = V/2. Переведя компьютер в режим записи (MODE [ ] [1]), введем новую строку:

3Ø W [=] V [/] 2 [EXE]

```
1187
3Ø  W = V/2
```

3. Добавление строк

Пусть в нашу программу необходимо вставить строку U = V*2 между строками 30 и 40, а строку 40 привести к виду: PRINT V, W, U.

Для новой строки выберем номер между 30 и 40, например 35. В режиме записи (MODE [ ] [1]) запишем новую строку:

35 U [=] V [*] 2 [EXE]

```
1179
35  U = V*2
```

Чтобы изменить строку 40, необходимо следующее нажатие клавиш:

LIST 4Ø [EXE]

```
1179
4Ø  PRINT V , W
```

[→] [S] [ ] U [EXE]

```
1177
5Ø  END
```

[AC]

```
1177
```

4. Удаление символов.

Пусть в строке 40 нашей программы необходимо устранить V , :

MODE [ ] [1]

```
1177
P  123456789
```

LIST 4Ø [EXE]

```
1177
4Ø  PRINT V , W
```

[←] [←]

```
1177
4Ø  PRINT V , W
```

передвинем курсор под символ, который нужно удалить

[DEL] [DEL]

```
1177
40 PRINT W.U
```

с каждым нажатием клавиши [DEL] удаляется символ, под которым находится курсор

[EXE]

```
1179
50 END_
```

[AC]

```
1179
_
```

5. Удаление строки.

Для устранения строки из текста программы необходимо в режиме записи набрать номер удаляемой строки и нажать клавишу [EXE].

Пусть необходимо удалить строку 30:

MODE [ ] [1]

```
1179
P ◆ 123456789
```

30 [EXE]

```
1187
_
```

6. Изменение порядка следования строк.

Запишем под номером P2 следующую программу:

```
10 INPUT N
20 M = N ↑ 2
30 L = N ↑ 0.5
40 PRINT M , L
50 END
```

Пусть в этой программе нужно поменять местами строки 20 и 30. Для этого следует заново обозначить строки так, чтобы номер оператора M = N ↑ 2 был больше номера оператора L = N ↑ 0.5 :

MODE [ ] [1] [S] P2 [ ]

```
1153
P ◆ 1 ◆ 3456789
```

LIST 20 [EXE]

```
1153
20 M = N ↑ 2
```

Изменим номер строки 20 на 35:

[←] [←] [←] [←] [←] [←] [←] [←]

```
1153
20.M = N ↑ 2
```

35 [EXE]

```
1145
10 PRINT M, L
```

Теперь строка M = N ↑ 2 присутствует в программе дважды – под номерами 20 и 35.

Удалим лишнюю строку:

[AC]

```
1145
_
```

20 [EXE]

```
1153
_
```

Можно убедиться в правильности исправлений:

MODE [ ] [Ø]

LIST [EXE]

```
READY  P 2
10  INPUT  N
30  L = N ↑ Ø.5
35  M = N ↑ 2
40  PRINT  M , L
50  END
READY  P 2
```

Отладка программы

Отладка программы проводится с целью выявления возможных ошибок (грамматических, алгоритмических и т. д.). Проводить отладку можно в диалоговом режиме с использованием индикатора, на котором появляется сообщение об ошибке с указанием ее типа и номера строки, в которой она допущена.

Рассмотрим процесс отладки на примере.

Пусть под номером PØ записана программа:

```
10 INPUT X
20 Y = X ↑ 2 + 3*X + 15
30 PRINT Y
40 END
```

Допустим, строка 20 записана с ошибкой: Y = X ↑ 2 + 3X + 15.

При выполнении программы будем иметь:

MODE [ ] [Ø] RUN [EXE]

```
?
```

Введем какое-нибудь значение X:

4 5 [EXE]

```
ERR 2  PØ — 2Ø
```

это сообщение означает, что в строке 20 допущена ошибка, которую необходимо устранить

Для локализации ошибки в строке следует нажать клавишу [ANS]:

[ANS]

```
2Ø  Y = X ↑ 2 + 3X +
```

Мерцающий курсор указывает на место, где допущена ошибка. В данном случае пропущен знак "*". Внесем поправку уже известным способом:

[S] INS [ ]

```
2Ø  Y = X ↑ 2 + 3 X
```

[*] [EXE]

```
2Ø  Y = X ↑ 2 + 3 * X
```

Такой способ отладки позволяет устранить те ошибки, о которых получено сообщение компьютера. Возможно, однако, что сообщения об ошибке нет, а ре-

зультаты выполнения программы отличаются от ожидаемых. В этом случае отладку можно провести одним из следующих способов.

а) Отладка с использованием команды STOP

Запишем под номером P0 программу:

```
10 Y = 0
20 INPUT N , X
30 FOR I = 1 TO N
40 Y = Y + X ^ 2
50 NEXT I
60 PRINT Y
70 END
```

Чтобы определить правильность выполнения программы, необходимо определить значение переменной Y в каждом элементе цикла. Для этого между 40 и 50 строками следует записать оператор STOP:

MODE [ ]	[1]	P? 1 2 3 4 5 6 7 8 9
45 STOP	[EXE]	45 STOP

Процесс выполнения программы теперь будет выглядеть следующим образом:

MODE [ ]	[0]	READY P0
RUN	[EXE]	?
(N = 4) ÷ 4	[EXE]	?
(X = 35) ÷ 35	[EXE]	P0 — 4 5

После появления такого сообщения можно определить, чему равно значение Y при соответствующем I:

Y	[EXE]	1 2 2 5

Для продолжения выполнения программы следует нажать клавишу [EXE]. После вычисления каждого нового значения Y программа останавливается. Так можно проверить ход выполнения всей программы.

б) Отладка в режиме TRACE (MODE [ ] [2]).

Когда компьютер находится в режиме TRACE, выполнение программы осуществляется с остановом после отработки каждой строки. В этот момент можно проверить результат ее выполнения.

Для нашего примера:

MODE [ ]	[0]	READY P0
MODE [ ]	[2]	READY P0 (TR)
[S] P0 [ ]		P0 — 1 0 (TR)
	[EXE]	P0 — 2 0 (TR) — отработана строка 10

```
                TR
      [EXE]  ?                          — отрабатывается строка 20,
                TR                        компьютер ждет ввода пер-
4     [EXE]  ?                          вого числа (N)
                TR
35    [EXE]  P 0 — 3 0
                TR
      [EXE]  P 0 — 4 0
                TR
      [EXE]  P 0 — 4 5                  — отработана строка 45, мож-
                TR                        но проверить содержимое
Y     [EXE]  1 2 2 5                    переменной Y
                TR
      [EXE]  P 0 — 4 5
                TR
      [EXE]  P Я — 5 Я
```

Таким образом можно проверить ход выполнения всей программы. Данный способ наиболее удобен при отладке сложных программ.

Для выхода из режима отладки TRACE следует нажать клавиши MODE [ ] [3]

6.4. Операторы (команды), используемые в программах

Операторы перехода

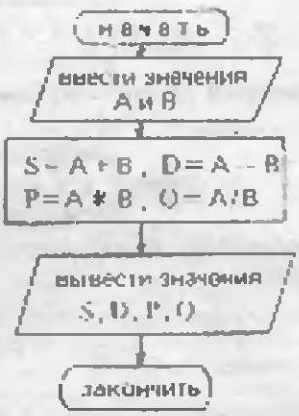
Операторы перехода можно разделить на два основных типа — безусловные (GOTO) и условные (IF).

Оператор GOTO

Этот оператор осуществляет безусловный переход к строке с указанным номером. Программа выполняется последовательно, начиная с оператора, к которому был произведен переход.

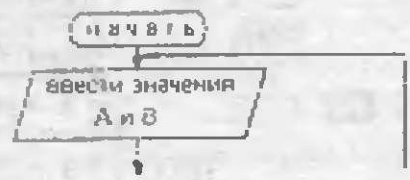
Пример.

```
10 INPUT A , B
20 S = A + B
30 D = A - B
40 P = A * B
50 Q = A/B
60 PRINT S , D , P , Q
70 END
```

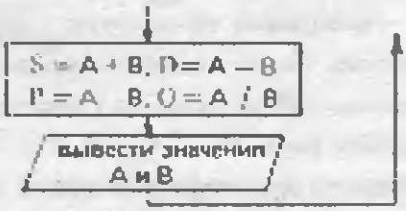


Если необходимо выполнить вычисления по программе с различными исходными данными A и B, каждый раз придется обращаться к программе (RUN [ ]). Этого можно избежать, если вместо оператора END ввести оператор GOTO.

```
10 INPUT A , B
20 S = A + B
30 D = A - B
40 P = A * B
```



```
50 Q = A / B
60 PRINT S , D , P , Q
70 GOTO  10
```

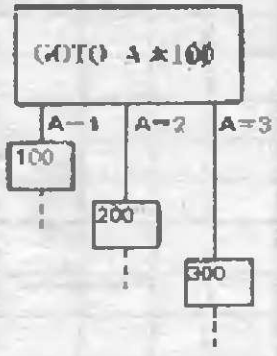


После индикации результатов вычислений для первой группы исходных данных компьютер автоматически перейдет к началу программы — будет ждать ввода новых данных.

Оператор GOTO N, где N — номер строки, однозначно определяет переход к выполнению того или иного фрагмента программы. В ряде случаев, однако, бывает необходимо осуществлять переход к фрагменту программы, начало которого определяется значением какого-либо параметра. Для этого используется косвенная адресация. Формат оператора в этом случае:

GOTO переменная или выражение

Переменная (A, B, X и т. д.) или выражение (A + B, X + 10 и т. д.) определяют номер строки, к которой должен быть осуществлен переход.



Пример 1. GOTO A*100

Здесь в зависимости от значения A после отработки оператора GOTO будет осуществлен переход к одному из фрагментов (100, 200 или 300), причем A может принимать только значения 1, 2 или 3. В противном случае произойдет ошибка.

При помощи оператора GOTO можно осуществлять переход к выполнению программы, записанной под другим номером. Формат оператора при этом может иметь вид GOTO#n или GOTO # n, N, где n — число от 0 до 9 (номер программы), N — номер строки. В первом случае произойдет переход к программе, записанной под номером Pn, начиная с первой ее строки, во втором случае — со строки под номером N.

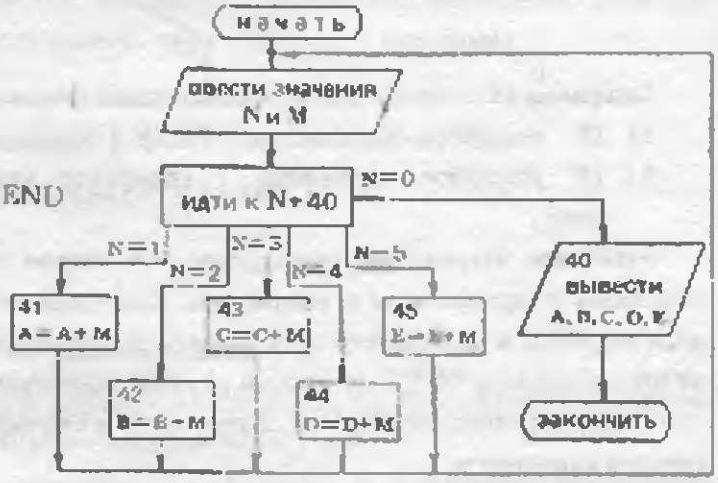
Пример 2. GOTO # 2,250

Если этот оператор стоит в программе, записанной под номером P0, то его отработка вызовет переход к 250-й строке программы, записанной под номером P2.

Пример 3.

Рассмотрим программу для подсчета суммарного количества величин, классифицированных по определенному признаку.

```
10 VAC
20 INPUT N , M
30 GOTO N + 40
40 PRINT A , B , C , D , E :  END
41 A = A + M : GOTO 20
42 B = B + M : GOTO 20
43 C = C + M : GOTO 20
44 D = D + M  : GOTO 20
45 E = E + M  : GOTO 20
```



Программа начинается с оператора VAC, производящего очистку всех переменных. Затем следует оператор ввода, фиксирующий номер классификационной группы (N), а затем соответствующее этой группе число. Оператор GOTO осуществляет переход к строке (41 ... 45), где и происходит суммирование чисел определенной классификационной группы. После ввода всех данных вместо номера N и числа M следует ввести "0". На индикаторе появятся суммарные значения чисел в каждой группе.

Пусть имеется следующий набор данных:

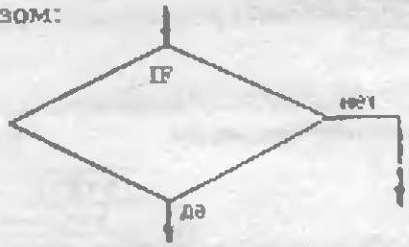
№ группы	Число
3	2870
2	1960
5	3850
5	1250
1	2500
2	2310
3	1850
5	4370
3	5360
1	2220
2	1450
4	6120
1	3100
0	0

Программа просуммирует одноименные величины:

№ группы	Число
1	7820
2	5720
3	10080
4	6120
5	9470

Разделитель ":" в строках 40 ... 45 позволяет использовать одну строку для записи нескольких операторов. Такой прием приводит к уменьшению числа шагов программы.

В отличие от GOTO оператор IF обеспечивает переход в программе в зависимости от истинности или ложности какого-либо условия. В блок-схеме он обозначается следующим образом:



Оператор IF используется в следующих форматах:

а) IF условное обозначение THEN {номер строки или выражение};

б) IF условное выражение ; {оператор, команда, математическое выражение}.

Условное выражение определяет некоторое условие, проверка выполнения которого и происходит в операторе. Если условие выполняется, то осуществляется переход к фрагменту программы, начинающемуся со строки, номер которой стоит за словом THEN (в случае *а*), или выполняется команда, оператор и т. п., стоящие за разделителем ":". В противном случае будет выполнен следующий по порядку оператор.

В качестве условного выражения могут быть использованы:

X > 10, A$ = "123",

A ≥ B, P$ ≠ Q$ и т. д.

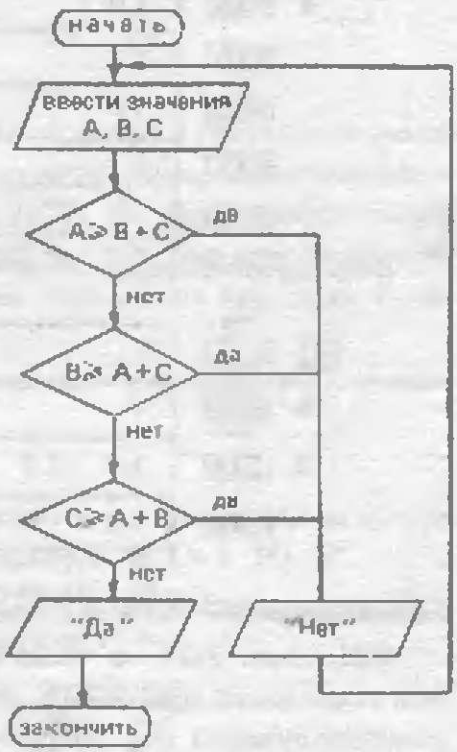
N = L + 3,

Использовать THEN или ";" в условном операторе следует в зависимости от конкретных условий:

IF A ≤ D THEN 60 . IF E = A + B ; PRINT E и т. д.

Пример. Программа, определяющая возможность построения треугольника по трем сторонам заданной длины.

```
10 INPUT A , B , C
20 IF  A ≥ B + C  THEN 70
30 IF  B ≥ A + C  THEN 70
40 IF  C ≥ A + B  THEN 70
50 PRINT "Да"
60 END
70 PRINT "Нет"
80 GOTO 10
```



Чтобы построить треугольник, длина любой его стороны должна быть меньше суммы длин двух других сторон. Проверка этих условий осуществляется в строках 20 ... 40. В случае их выполнения на индикаторе появится "Да", в противном случае — "Нет".

Оператор вывода PRINT

Оператор PRINT используется для вывода данных на индикатор. Он имеет следующий формат:

PRINT { "ТЕКСТ" / переменная / выражение } [{ ; / , } { ... }]

Использование оператора PRINT приводит к индикации числового значения переменной, выражения и пр., или текста, заключенного в кавычки:

Примеры.

PRINT A → 18 (если A = 18)

PRINT C*D ↑ 2 → 100 (если C*D ↑ 2 = 100)

PRINT "A" → A

PRINT "Ответ" → Ответ

PRINT "N=",N → N = 256 (если N = 256)

Оператор PRINT может быть использован для вывода на индикатор нескольких значений. При этом используются разделители "," и ";".

Если в качестве разделителя берется ",", то первым на индикаторе появляется

значение переменной (или части текста), непосредственно следующей за словом PRINT . Для появления очередного значения необходимо нажать клавишу [EXE] . При разделителе "; " значения или текст появляются на индикаторе одновременно.

В качестве примера рассмотрим программу, приведенную на стр. 30 .

Использование оператора 60 PRINT S , D , P , Q даст следующие результаты:

[S]	P9	?	
15	[EXE]	?	
3	[EXE]	18	(S)
	[EXE]	12	(D)
	[EXE]	45	(P)
	[EXE]	5	(Q)
	[EXE]	Ready	

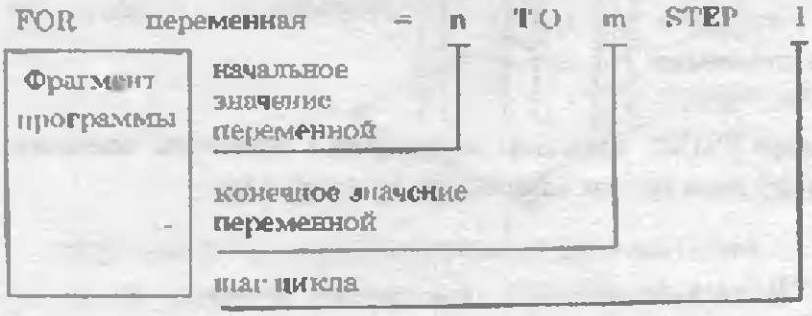
Если оператор вывода будет иметь вид 60 PRINT S ; D ; P ; Q , то:

[S]	P9	?
15	[EXE]	?
3	[EXE]	18 12 45 5
	[EXE]	Ready

Операторы цикла FOR и NEXT

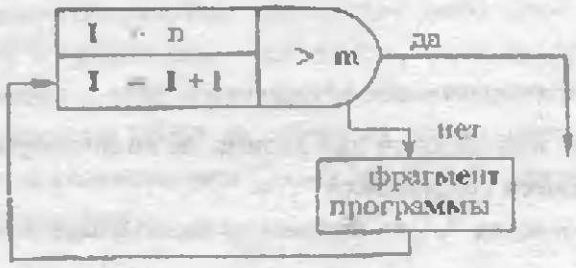
Операторы FOR и NEXT используются для обозначения начальной и конечной точек цикла программы. Все операторы, стоящие между оператором FOR и соответствующим ему оператором NEXT , будут периодически выполняться в соответствии с условием, заданным в операторе FOR .

Формат операторов цикла имеет вид:



NEXT < переменная >

Блок-схему можно изобразить в виде:



Параметры n , m и l могут быть заданы в неявном виде.

Пример. FOR I = SQR A (n) TO SIN B (m) STEP EXP C (l)

При этом допускается запись, при которой значения m и l присваиваются какой-либо переменной:

FOR A = LOG B TO C = COS D STEP E = INT F

Переменная, следующая за словом FOR в операторе цикла, рассматривается как управляющая переменная. Та же переменная должна появляться в операторе NEXT , определяющем конец цикла.

Пример. FOR I = 0 TO 10 STEP 2

...

NEXT I

Здесь переменной I присваиваются значения 0, 2, 4, 6, 8 и 10. Так как она не может иметь все эти значения одновременно, создается цикл, начинающийся с оператора FOR и заканчивающийся оператором NEXT . Операторы внутри цикла выполняются 6 раз, причем переменная I каждый раз получает новое значение. Оператор NEXT обеспечивает переход к оператору FOR до тех пор, пока I не достигнет своего конечного значения — 10.

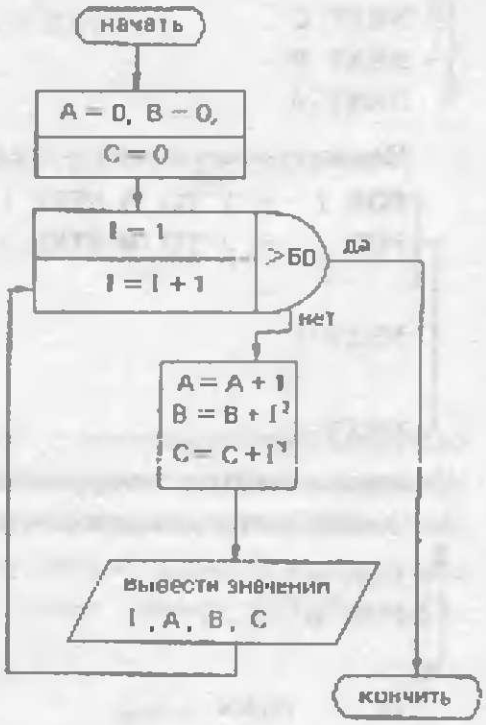
Если необходимо организовать цикл с уменьшением управляющей переменной, следует взять шаг цикла со знаком "—":

FOR I = 5 TO 1 STEP — 1

Если шаг цикла равен +1, его можно не указывать. В этом случае запись FOR I = 1 TO 10 STEP 1 будет эквивалентна записи FOR I = 1 TO 10.

Пример. Пусть необходимо вычислить суммы $\sum_{i=1}^{n} i$, $\sum_{i=1}^{n} i^2$, $\sum_{i=1}^{n} i^3$, где n изменяется от 1 до 50.

```
10 VAC
20 FOR I = 1 TO 50
30 A = A + I
40 B = B + I ↑ 2
50 C = C + I ↑ 3
60 PRINT I, A, B, C
70 NEXT I
80 END
```



Эту задачу можно решить с использованием оператора условного перехода:

```
10 VAC
20 I = 1
30 A = A + I
40 B = B + I ↑ 2
```

```
50 C = C + I^3
60 PRINT I, A, B, C
70 I = I + 1
80 IF I = 51; END
90 GOTO 30
```

Из сравнения двух программ видно, что одинаковые функции выполняют операторы:

```
20 FOR I = 1 TO 50          и   20 I = 1
                                70 I = I + 1

80 NEXT I                   и   80 IF I = 51; END
                                90 GOTO 30
```

То есть при необходимости многократного выполнения фрагмента программы предпочтение следует отдавать операторам цикла FOR и NEXT.

Вложенные циклы

Циклы, задаваемые с помощью операторов FOR и NEXT, допускается "вкладывать" друг в друга, то есть организовывать циклы внутри циклов. Их число может достигать четырех:

```
FOR A = ...
FOR B = ...
FOR C = ...
FOR D = ...
  .
  .
NEXT D
NEXT C
NEXT B
NEXT A
```

Это пример правильной организации циклов. Перекрытие их недопустимо. Внутренний цикл должен начинаться после оператора FOR внешнего цикла и заканчиваться до оператора NEXT внешнего цикла.

Недопустимым является следующее:

```
FOR I = 1 TO 5 STEP 1
FOR J = 2 TO 10 STEP 2
   ...
NEXT I
   ...
NEXT J
```

Используя оператор условного перехода IF, можно осуществить выход из цикла, "войти" внутрь его иным нельзя:

```
FOR A = ...
FOR B = ...
...
IF ... THEN ...
NEXT B
NEXT A
```

Оператор ввода INPUT

Этот оператор используется для присвоения переменной численного значения, вводимого в процессе выполнения программы после появления на индикаторе символа "?".

Формат оператора имеет вид:

INPUT ["строка текста" ,] переменная, ["строка текста" ,] переменная, ...

Строка текста может быть опущена. При наличии же текст, заключенный в кавычки, появляется на индикаторе перед символом "?".

Пример.

INPUT A



INPUT "A = ", A



Дальнейшее выполнение программы продолжается после ввода с клавиатуры численного значения переменной.

Из примера видно, что использование оператора INPUT со строкой текста позволяет избежать ошибок при вводе исходных данных.

Появление на индикаторе символа "?" свидетельствует о том, что компьютер отработал оператор INPUT и ждет ввода значения переменной. Вывести компьютер из этого состояния можно, если ввести какое-либо число или прервать выполнение программы (MODE).

Вместо численного значения переменной может быть введено математическое выражение или имя другой переменной, численное значение которой будет присвоено переменной, указанной в операторе INPUT .

Оператор KEY

По этому оператору в текстовую переменную вносится символ, соответствующий клавише, нажатой во время отработки оператора.

Формат оператора: Текстовая переменная = KEY

Пример.

```
10 A$ = KEY : IF A$ = "" THEN 10
20 IF A$ = "A" THEN 100
30 IF A$ = "B" THEN 200
40 IF A$ = "C" THEN 300
50 GOTO 10
:
```

Такого рода программа служит для перехода к выполнению определенного фрагмента программы в зависимости от нажатой во время отработки оператора KEY клавиши. Переход к какому-либо фрагменту программы из состояния ожидания ввода может быть осуществлен только после нажатия одной из клавиш, оговоренных в операторе условного перехода IF (в нашем примере – A, B или C).

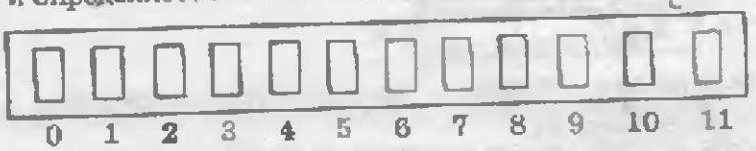
Оператор CSR

Оператор местоположения курсора CSR определяет номер позиции, начиная с которой на индикаторе будут появляться результаты вычислений (с учетом знака).

Оператор CSR имеет следующие форматы:

```
CSR n
CSR ,
CSR n ,
CSR , n
```

Номер позиции n может быть задан числом, переменной или математическим выражением и определяется в соответствии с рисунком:



В числе или результате математического выражения дробная часть игнорируется, целая часть должна быть от 0 до 11.

В первом формате вывод на индикатор происходит с n-й позиции, причём предыдущая информация сохраняется:

```
10 CSR 5 : PRINT "AB"
```



Во втором формате индикатор очищается, и информация появляется с первой позиции:

```
10 CSR , : PRINT "AB"
```



В последних двух форматах все происходит так же, как в первом, только в третьем индикатор очищается справа от указанной позиции, а в четвертом слева:

```
10 CSR 5 , : PRINT "AB"
```




```
10 CSR , 5 : PRINT "AB"
```

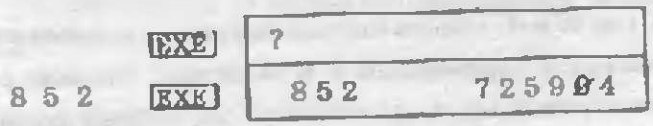


Пример.

```
10 INPUT X
20 PRINT X ;
30 CSR 5 , : PRINT X ↑ 2
40 GOTO 10
```

Программа выполняет вычисление квадрата исходного числа и воспроизводит на индикаторе оба числа одновременно. Местоположение второго числа определяет оператор CSR 5.

	EXE	?
7	EXE	7 49
	EXE	?
45	EXE	45 2025



Такой прием удобен, если необходимо, чтобы исходные данные и результаты одновременно отображались на индикаторе.

Оператор SET

Оператор SET используется при округлении чисел и определяет количество значащих цифр в числе.

Формат оператора: SET n , где n — число от 1 до 10.

Пример.

```
10 SET 4
20 MODE 4
30 FOR X = 0 TO 180 STEP 5
40 PRINT SIN X
50 NEXT X
60 SET 10
70 END
```

Программа выполняет вычисление SIN X при изменении аргумента от 0 до 180° с шагом 5°. Результаты вычислений округляются до четырех значащих цифр. После отработки программы все вычисления будут производиться с указанной точностью. Для отмены округления в программу введен оператор "60 SET 10".

Операторы LEN, MID, VAL, GETC

Эти операторы используются при выполнении операций с текстовыми переменными (A$, B$, $ и др.).

Оператор LEN служит для определения числа символов, составляющих текстовую переменную.

Формат оператора: LEN (текстовая переменная)

Пример.

```
10 A$ = "1 2 3"
20 B$ = "4 5 6"
30 C$ = A$ + B$
40 D = LEN (C$)
50 PRINT D
60 END
```

Программа определяет длину текстоной переменной C$, полученной "склейкой" переменных A$ и B$.

Оператор MID предназначен для выделения нескольких символов, входящих в состав текстовой переменной $ (применим только к этой переменной).

Формат оператора: MID (m, n), где m и n — числа или математические выражения. Оператор MID выделяет из текстовой переменной $ n символов, начиная с m-го. В случае задания n и m математическими выражениями десятичная часть отбрасывается, при этом значения n и m должны лежать в пределах от 1 до 30. Кроме того, должно выполняться условие: m + n < LEN ($) + 1.

Пример 1.

Пусть $ = "A B C D E F G H I J"

Тогда MID (3, 4) → CDEF

MID (1, 2) → AB

MID (5, 6) → EFGHIJ

Пример 2.

```
10 INPUT $
20 N = LEN ($)/2
30 X$ = MID (1, N) : Y$ = MID (N + 1, 2*N − N + 0.5)
40 PRINT X$, Y$
50 END
```

Программа делит текстовую переменную $ на две части. Сначала вычисляется половина "длины" $. Полученное число определяет номер символа, которым закончится первая из двух полученных переменных. Эти переменные записываются в X$ и Y$ соответственно и отображаются на индикаторе. Длина текстовой переменной $ может достигать 30 символов, но так как в нашем примере она поделена на две части (а длина текстовой переменной не более 7 символов), длина $ не должна превышать 14 символов.

Оператор VAL преобразует текстовую переменную в числовую.

Формат оператора: VAL (текстовая переменная)

Ясно, что применение оператора VAL для текстовой переменной, в состав которой входят какие-либо символы, кроме цифр или десятичной точки, не имеет смысла и приводит к ошибке.

Пример.

Пусть Z$ = "70693"

Тогда VAL (Z$) = 70693

Оператор GET$ выделяет один указанный символ из текстовой переменной.

Формат оператора:

$$\text{GET\$}\ (\left\{\begin{array}{c}\text{"ТЕКСТ"}\\ \text{текстовая переменная}\\ \text{выражение}\end{array}\right\}, N)$$

Оператор выделяет N-й символ из текста, заключенного в кавычки, или текстовой переменной.

Пример.

```
10 A$ = "ABCD" : B$ = "EF"
20 C$ = GET$ (A$ + B$, 5)
```

В результате выполнения программы текстовая переменная C$ примет вид:

C$ = "E"

Оператор AUTO

Оператор автоматической нумерации строк программы AUTO используется в режиме записи программы.

Формат оператора: AUTO [шаг изменения номеров строк].

Применение оператора AUTO позволяет исключить необходимость набора номеров строк программы.

Пример. Пусть необходимо записать программу, строки которой пронумерованы от 5 до 25 с шагом 5.

Переведем компьютер в режим записи:

MODE [ ] [1]



Введем оператор AUTO :

AUTO 5 [EXE]



(или [F] [ ])
AUTO

Компьютер готов к вводу строки с номером 5. Набрав необходимую информацию, следует дважды нажать клавишу [EXE] :

... [EXE] [EXE]



После первого нажатия клавиши в компьютер заносится строка, а после второго на индикаторе появляется номер следующей. Так можно записать всю программу. Если шаг изменения номеров строк не указывать, он принимается равным 10.

Подпрограммы

Часто в программу бывает нужно записать подпрограмму (группу операторов) только один раз, а затем неоднократно обращаться к ней из различных точек программы. Последним оператором подпрограммы должен быть оператор RETURN. Оператор GOSUB используется для перехода к первому оператору подпрограммы.

Формат оператора имеет вид:

1) GOSUB число (или математическое выражение);

2) GOSUB # число (или математическое выражение).

В первом случае число (или результат математического выражения) может находиться в пределах от 1 до 9999 (десятичная часть отбрасывается), оно же определяет номер строки первого оператора подпрограммы, во втором случае — от 0 до 9. Это номер файла (F0 ... F9), в котором может быть записана подпрограмма. Рассмотрим применение подпрограмм на примерах.

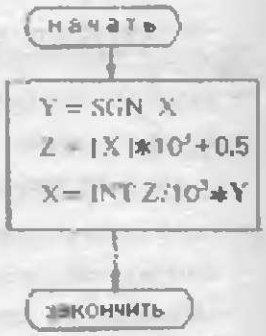
Пример 1.

Составить программу, выполняющую вычисление площади поверхности и объема тетраэдра со стороной A с точностью три знака после запятой. Округление оформить в виде подпрограммы.

$S = \sqrt{3}a^2$; $V = \frac{\sqrt{2}}{12}a^3$

```
10 INPUT A
20 X = SQR 3*A*A
30 GOSUB 1000
40 PRINT X
50 X = SQR 2/12*A*A*A
60 GOSUB 1000
70 PRINT X
80 END
1000 Y = SGN X
1010 Z = ABS X*10 ↑ 3 + 0.5
1020 X = INT Z/10 ↑ 3*Y
1030 RETURN
```





Основная программа вычисления S и V достаточно проста и не нуждается в пояснениях. Внимания заслуживает фрагмент, осуществляющий округление результатов, оформленный в виде подпрограммы. Подпрограмма округляет результаты, вычисленные на строках 20 и 50, до трех знаков после запятой.

Обращение к подпрограмме может быть осуществлено столько раз, сколько это необходимо. В нашем примере это происходит дважды — сначала для получения величины S, а затем V.

Если отказаться от использования подпрограммы и делать все вычисления в основной программе, число шагов для ее записи возрастет:

```
10 INPUT A
20 X = SQR 3*A*A
30 Y = SGN X
40 Z = ABS X*10 ↑ 3 + 0.5
50 X = INT Z/10 ↑ 3*Y
60 PRINT X
70 X = SQR 2/12*A*A*A
80 Y = SGN X
90 Z = ABS X*10 ↑ 3 + 0.5
100 X = INT Z/10 ↑ 3*Y
110 PRINT X
120 END
```

Эта программа и приведенная выше выполняют аналогичные вычисления, однако программа с использованием подпрограммы короче и проще.

Пример 2.

Составить программу, выполняющую вычисление площади поверхности и

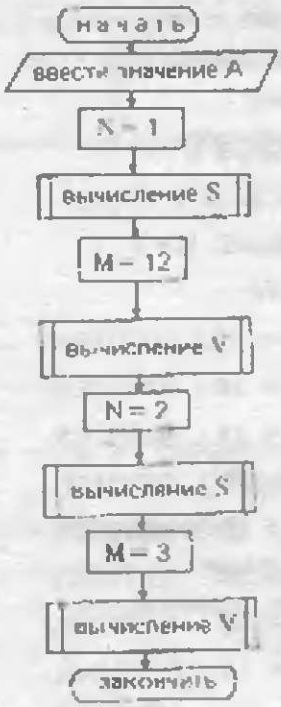
объема тетраэдра и октаэдра со стороной A. Точность вычислений — четыре знака после запятой. Подпрограмму округления оформить в виде отдельной программы (P1).

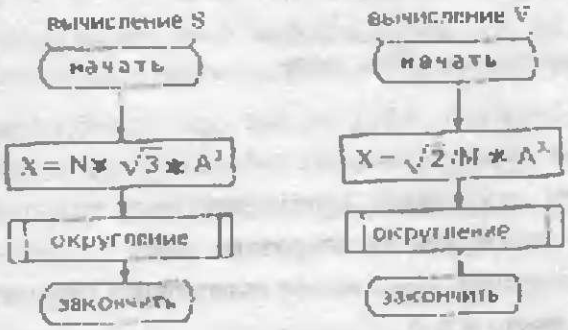
Тетраэдр: $S = \sqrt{3}\,a^2$; $V = \frac{\sqrt{2}}{12}\,a^3$

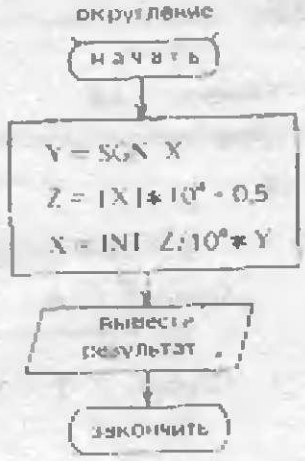
Октаэдр: $S = 2\sqrt{3}\,a^2$; $V = \frac{\sqrt{2}}{3}\,a^3$

```
P0  10 INPUT A
    20 N = 1
    30 GOSUB 1000
    40 M = 12
    50 GOSUB 2000
    60 N = 2
    70 GOSUB 1000
    80 M = 3
    90 GOSUB 2000
   100 END
  1000 X = N * SQR 3 * A * A
  1010 GOSUB # 1
  1020 RETURN
  2000 X = SQR 2 / M * A * A * A
  2010 GOSUB # 1
  2020 RETURN

P1  10 Y = SGN X
    20 Z = ABS X * 10 ↑ 4 + 0.5
    30 X = INT Z / 10 ↑ 4 * Y
    40 PRINT X
    50 RETURN
```







В этой программе использованы три подпрограммы — две для вычисления площади поверхности S и V и одна для округления результатов с заданной точностью.

Внимания заслуживает тот факт, что обращение к подпрограмме, осуществляющей округление, производится не из основной программы, а из подпрограммы, то есть мы имеем дело с вложенными подпрограммами. Уровень вложенности может достигать 8. Число подпрограмм одного уровня вложенности.

т. е. работающих независимо друг от друга, практически неограниченно.

Косвенное обращение к подпрограмме

Косвенное обращение к подпрограмме осуществляется аналогично косвенному безусловному переходу.

Пример 1.

```
10 INPUT N
20 GOSUB N + 100
30 PRINT X*6
40 END
101 X = 5 : RETURN
102 X = 10 : RETURN
103 X = 15 : RETURN
104 X = 20 : RETURN
105 X = 25 : RETURN
106 X = 30 : RETURN
107 X = 35 : RETURN
108 X = 40 : RETURN
109 X = 45 : RETURN
110 X = 50 : RETURN
```

Здесь обращение к той или иной программе осуществляется в зависимости от величины N . Если N = 1 ... 10, работает одна из подпрограмм 101 ... 110. При этом на индикаторе появляется значение соответствующего X , умноженного на 6 .

Операторы ASCI и CHR

Оператор ASCI служит для преобразования текстового символа в целое число (в соответствии с таблицей кодов символов) . Любое индицируемое число, буква или символ имеют свой код, который и появляется на индикаторе в результате отработки оператора ASCI.

Оператор ASCI имеет следующий формат: ASCI "*", где * — буква, символ, число и т. д.

Пример.

```
10 INPUT "Символ", A$
20 PRINT ASCI A$
30 GOTO 10
```

Программа выдает на индикатор код запрашиваемого символа:

```
Символ ? A
  65
Символ ? T
  84
Символ ? $
  36
```

Оператор CHR выполняет обратную по отношению к оператору ASCI функцию, т. е. преобразует целое число в текстовой символ (также в соответствии с таблицей кодов символов) . Формат оператора: CHR N , где N — код символа (целое число от 1 до 191) .

Пример

```
10 FOR  I = 1  TO  191
20 PRINT  I ; " ", CHR  I
30 NEXT  I
```

Выполнение программы сопровождается появлением на индикаторе кодов и соответствующих им символов:

1 :
2 :
3 :
и т. д.

Оператор LETC

Из таблицы кодов символов видно, что для кода 96 соответствующего символа нет. Он является резервным и его можно использовать для записи произвольного символа. Программирование произвольного символа в матрице 5 x 7 (один разряд индикатора) осуществляется построчно сверху вниз при помощи оператора LETC в соответствии с таблицей кодирования, в которой единица соответствует зажигающейся точке матрицы, нуль — невысвечивающейся.

Формат оператора: LETC " ж ж ж ж ж ж ж " , где ж — символ из таблицы кодирования. Этих символов может быть не более 7

Для вызова на индикатор закодированного символа служит оператор CHR 96.

Пример.

```
10 LETC "LALALAL" : PRINT CHR 96
```

В результате выполнения такой программы первый разряд индикатора будет заполнен точками в шахматном порядке.

Операторы DRAW и DRAWC

Оператор DRAW используется для зажигания в поле индикатора точки с указанными координатами x и y.

Формат оператора: DRAW x , y .

Координаты x и y могут быть заданы числом, переменной или выражением и должны лежать в пределах $0 \le x \le 59$, $0 \le y \le 6$.



Пример

```
10 FOR X = 0 TO 59 STEP 2
20 DRAW X, SIN(X * 18) * 3 + 3.5
30 NEXT X
```

Таблица кодирования символов

Код	Клавиша	Код	Клавиша	Код	Клавиша	Код	Клавиша
00000	0	01500	8	10000	G	11000	O
00001	1	01001	9	10001	H	11001	P
00010	2	01010	A	10010	I	11010	Q
00011	3	01011	B	10011	J	11011	R
00100	4	01100	C	10100	K	11100	S
00101	5	01101	D	10101	L	11101	T
00110	6	01110	E	10110	M	11110	U
00111	7	01111	F	10111	N	11111	V

В результате выполнения программы на индикатор выводится график функции y = sin x.

Оператор DRAWC аналогичен оператору DRAW , но служит для стирания вычерчиваемой точки.

Оператор **DEFM**

Ранее упоминавшийся оператор DEFM может быть использован и в программе, что очень удобно, так как отпадает необходимость в предварительном резервировании ячеек памяти.

Формат оператора: DEFM N , где N — число дополнительных ячеек памяти, которое может быть задано числом, переменной, выражением.

Так как число дополнительных ячеек памяти прямо пропорционально числу оставшихся шагов программы, полезно знать, какое число ячеек еще можно использовать. Для этого также служит оператор DEFM . В этом случае его формат имеет вид: DEFM , переменная.

После отработки такого оператора указанной переменной присваивается значение количества дополнительных ячеек памяти.

Пример.

10 DEFM , A : PRINT A

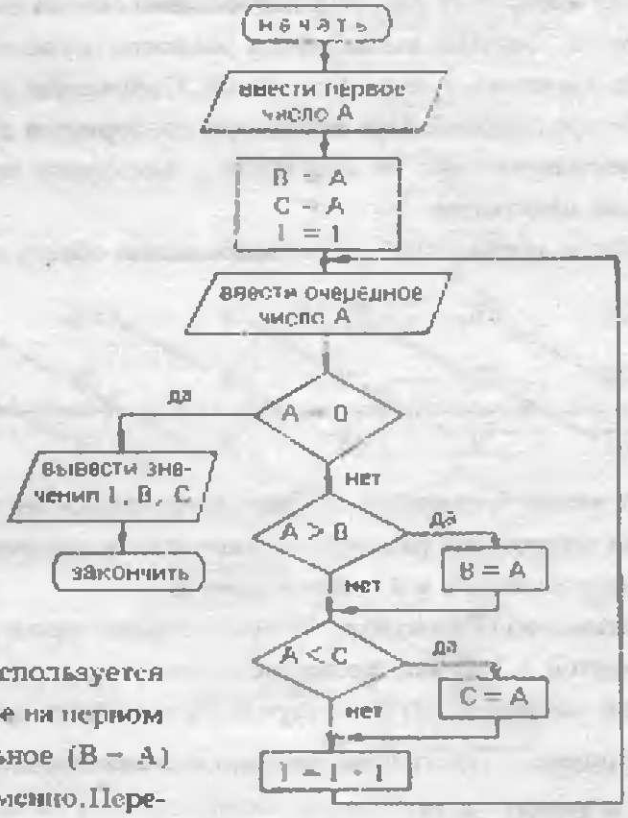
В результате выполнения программы на индикаторе появится число, равное количеству имеющихся дополнительных ячеек памяти.

6.3. Примеры программ

В настоящем разделе руководства приведен ряд программ, составленных для решения конкретных задач.

Пример 1. Программа нахождения max и min из ряда чисел.

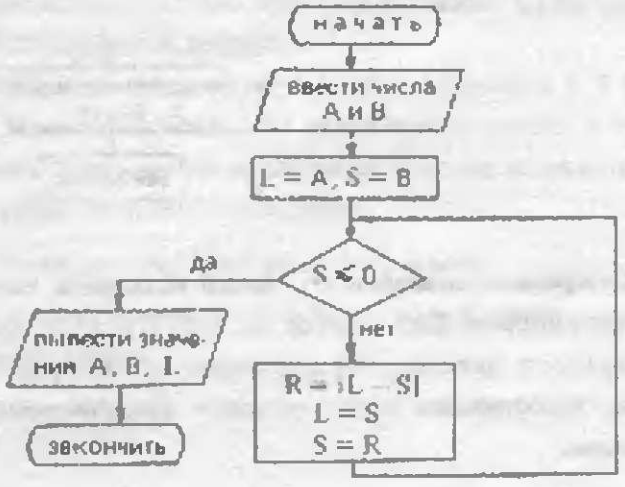
```
10 INPUT A
20 B = A
30 C = A
40 I = 1
50 INPUT A
60 IF A = 0 THEN 110
70 IF A > B ; B = A
80 IF A < C ; C = A
90 I = I + 1
100 GOTO 50
110 PRINT I ; B ; C
120 END
```



Оператор "10 INPUT A" используется для ввода первого числа, которое на первом этапе принимается за максимальное (B = A) и минимальное (C = A) одновременно. Переменная I определяет количество введенных чисел. Оператор "50 INPUT A" предназначен для ввода второго и всех последующих чисел (оператор "100 GOTO 50"). В строках 70 и 80 происходит сравнение введенного числа с максимальным и минимальным соответственно и присвоение его значения переменной B, если число больше максимального, или переменной A, если оно меньше минимального. После ввода всех чисел необходимо вместо очередного числа ввести 0. На индикаторе появятся количество чисел, максимальное и минимальное числа.

Пример 2. Программа нахождения наибольшего общего делителя (по алгоритму Евклида).

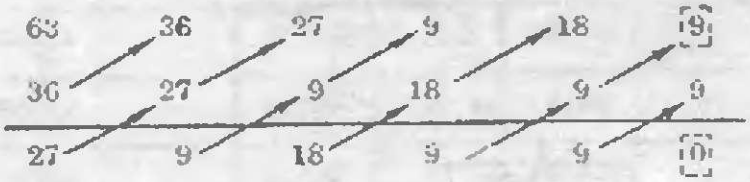
```
10 INPUT A, B
20 L = A
30 S = B
40 IF S <= 0 THEN 90
50 R = ABS (L - S)
60 L = S
70 S = R
```



```
80 GOTO 40
90 PRINT A, B, L
100 END
```

По алгоритму Евклида наибольший общий делитель определяется следующим образом. Сначала вычисляется разность первого и второго чисел. Затем эта разность вычитается из второго числа. Полученная разность в свою очередь вычитается из предыдущей. Эта процедура повторяется до тех пор, пока в результате очередного вычитания не получится 0. Последнее перед 0 число и будет наибольшим общим делителем.

Пусть необходимо найти наибольший общий делитель чисел 63 и 36:

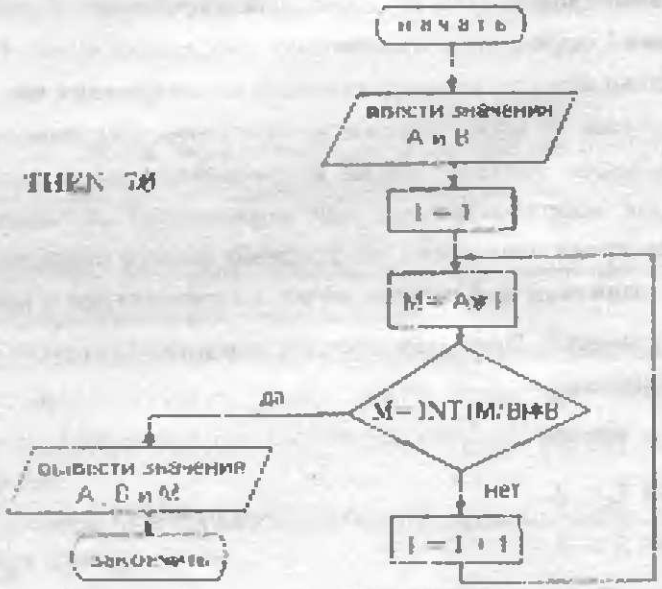


В нашей программе для получения разностей используются переменные L и S. Вновь полученная разность вычисляется в переменной R , затем значения S и R переписываются в L и S соответственно.

Оператор IF на строке 40 осуществляет проверку условия S = 0 . Если оно выполняется, в L имеем число, равное наибольшему общему делителю. В противном случае оператор GOTO на строке 80 приводит к повторению вычисления R .

Пример 3. Программа нахождения наименьшего общего кратного.

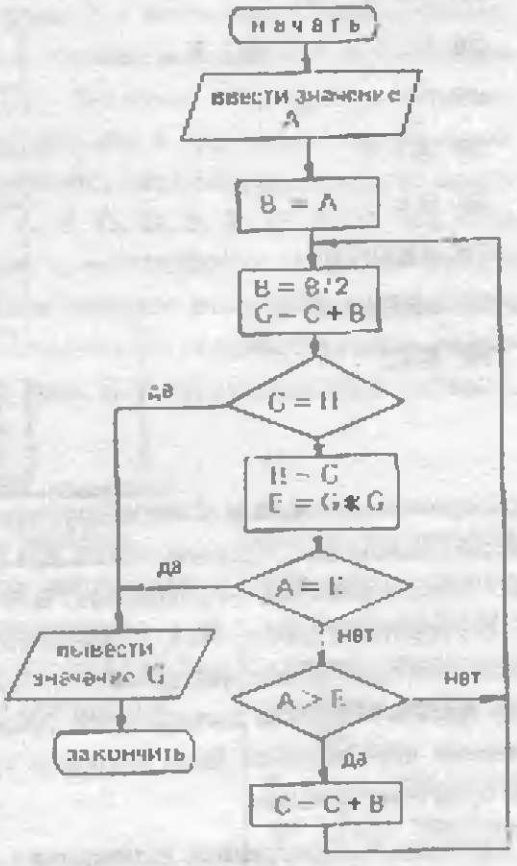
```
10 INPUT A, B
20 I = 1
30 M = A * I
40 IF M = INT (M/B) * B THEN 70
50 I = I + 1
60 GOTO 30
70 PRINT A, B, M
80 END
```



Программа начинается с ввода исходных чисел A и B, одно из которых (A) последовательно умножается на 1, 2, 3 и т. д., причем после каждого умножения проверяется, делится ли полученное произведение на второе число без остатка. В случае выполнения этого условия произведение и будет наименьшим общим кратным.

Пример 4. Программа вычисления квадратного корня методом дихотомии.

```
10 VAC
20 INPUT A
30 B = A
40 B = B/2
50 C = C + B
60 IF C = H THEN 120
70 H = C
80 E = C*C
90 IF A = E THEN 120
100 IF A > E ; C = C + B
110 GOTO 40
120 PRINT C
130 END
```



В соответствии с методом дихотомии значение квадратного корня числа A вычисляется путем последовательных приближений. За первое приближенное значение корня принимается A/2. Эта величина возводится в квадрат и сравнивается с A. Если они не равны, A/2 делится на 2 и полученное число добавляется к предыдущему значению корня (или вычитается от него). Затем вновь производится возведение в квадрат и сравнение с A. Эта процедура повторяется до тех пор, пока не будет найдено значение $\sqrt{A}$.

В программе оператор VAC производит очистку содержимого переменных. Затем вводится число A. Так как позже величина A понадобится в операторе IF, производить какие-либо операции над ней нельзя. Поэтому значение A присваивается переменной B, с которой и выполняются действия.

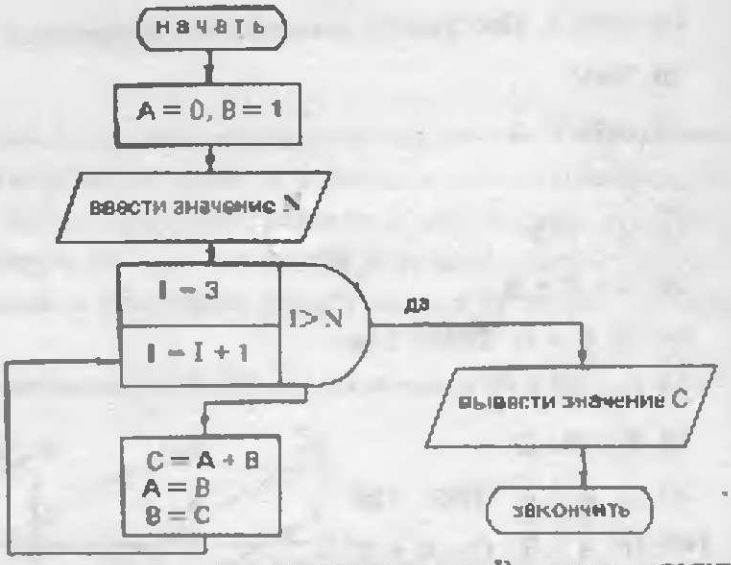
Максимальная погрешность при вычислениях не должна превышать ± 1 в десятом разряде. Переменные C и H используются для нахождения точки, в которой достигается заданная точность. Вычисления заканчиваются, когда два последовательно найденных приближенных значения совпадают.

Пример 5. Программа нахождения n – го члена ряда Фибоначчи.

Ряд Фибоначчи представляет собой последовательность целых чисел, каждое из которых равно сумме двух предыдущих:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

```
10 A = 0: B = 1
20 INPUT N
30 FOR I = 3 TO N
40 C = A + B
50 A = B
60 B = C
70 NEXT I
80 PRINT C
90 END
```

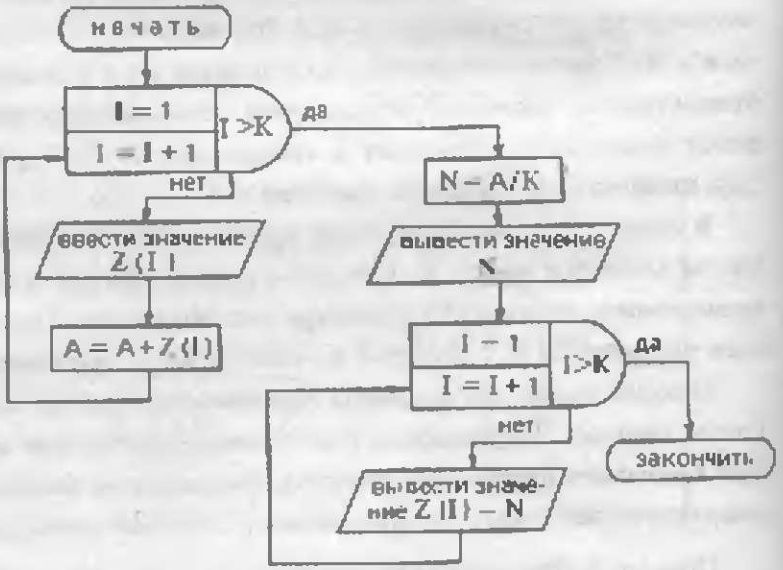


Переменные A, B и C используются для записи чисел ряда. B — число, следующее за числом A, C — сумма чисел A и B. После вычисления каждого нового числа идет переобозначение переменных: A = B, B = C.

В операторе цикла FOR переменная I принимает значения от 3 до N (N — номер числа, которое необходимо найти). Последовательность операторов 40 ... 60 менять нельзя. Сначала вычисляется сумма двух чисел и лишь затем идет переобозначение переменных. Если порядок следования операторов изменить, вычисления будут неверными.

Пример 6. Программа вычисления отклонения параметра от среднего значения (для K образцов).

```
10 A = 0
20 INPUT K
30 FOR I = 1 TO K
40  INPUT Z(I)
50 A = A + Z(I)
60 NEXT I
70 N = A/K
80 PRINT N
90 FOR I = 1 TO K
100 PRINT Z(I) - N
110 NEXT I
120 END
```



Для вычисления среднего значения некоторого параметра К образцов необходимо знать суммарное значение этого параметра. Так как оно вычисляется в переменной А, сначала следует присвоить ей значение 0, а затем уже ввести количество образцов (К). Значение параметра каждого образца заносится в соответствующий элемент одномерного массива Z (I). При большом количестве исходных данных применение массива гораздо удобнее, так как в противном случае пришлось бы использовать большое число разноименных переменных, а вместо оператора INPUT Z (I) записать оператор INPUT A, B, C, D, E, F, G, H ... При этом длина программы значительно увеличивается и ее отработка занимает больше времени. Однако при использовании массивов следует учитывать организацию памяти (см. стр. 11) и осуществлять при необходимости соответствующее ее расширение. Для нашей программы, например, если К = 50, необходимо добавить еще 50 ячеек памяти: DEFM 50.

6.6. Список сообщений об ошибках

Сообщение	Тип ошибки	Причина ошибки	Способ устранения
ERR 1	Ошибка в использовании памяти.	Недостаточное число шагов для записи программы. Превышение уровня при вычислениях по формулам.	Стереть лишние программы или сократить число зарезервированных ячеек памяти. Разбить формулы на более простые.
ERR 2	Синтаксическая ошибка.	Ошибка в формате оператора, при записи формулы, математического выражения и т. д.	Исправить ошибку в тексте программы, в формуле и т. д.
ERR 3	Математическая ошибка.	Результат вычисления превышает $10^{40/5}$. Аргумент функции не входит в допустимый диапазон. Попытка выполнения некорректной операции.	Исправить формулу или исходные данные.
ERR 4	Ошибка в использовании номера строки.	Адресация к несуществующему номеру строки в операторах GOSUB или GOTO .	Исправить номер строки.
ERR 5	Ошибка в аргументе.	Аргумент не лежит в допустимых пределах для команды, использующей аргумент.	Исправить ошибку в аргументе.

Сообщение	Тип ошибки	Причина ошибки	Способ устранения
ERR 6	Ошибка в использовании переменной.	Использование переменной, незарезервированной оператором DEFM.	Провести расширение памяти.
		Использование одной и той же переменной в качестве переменной и текстовой переменной.	Разделить имена переменных и текстовых переменных.
ERR 7	Ошибка в организации цикла или подпрограммы.	Использование оператора RETURN без обращения к подпрограмме, а оператора NEXT без оператора FOR. Уровень вложенности подпрограмм превышает 8, а уровень вложенности циклов — 4.	Устранить лишний оператор RETURN или NEXT. Установить требуемый уровень вложенности циклов и подпрограмм.
ERR 8	Ошибка в текстовой переменной	В текстовой переменной число символов превышает 7. В текстовой переменной $ число символов превышает 30.	Исправить текстовую переменную

6.7. Список команд и операторов

Команда или оператор	Формат	Область применения
INPUT	INPUT ["ТЕКСТ",] переменная [,...]	Используется для присвоения переменной численного значения, вводимого в процессе выполнения программы.
KEY	Текстовая переменная = KEY	Используется для записи в текстовую переменную символа, соответствующего нажатой во время отработки оператора клавише.
PRINT	PRINT {"ТЕКСТ" / переменная / выражение} [{; / ,} {...}]	Используется для вывода информации на индикатор.
CSR	CSR [,] [{число / переменная / выражение}] [,]	Используется для указания позиции, с которой должна начинаться индикация.

Команда или оператор	Формат	Область применения
GOTO	GOTO [#] { число / переменная / выражение } [, { ... }]	Используется для безусловного перехода к строке программы с указанным номером
IF ... { THEN / ; } ...	IF условное выражение { THEN / ; } { число / переменная / выражение }	Используется для обеспечения перехода в программе в зависимости от истинности или ложности какого-либо условия.
GOSUB	GOSUB [#] { число / переменная / выражение } [, { ... }]	Используется для перехода к первому оператору подпрограммы.
RETURN	RETURN	Используется для возвращения в основную программу после выполнения подпрограммы.
FOR	FOR $v = A_1$ TO A_2 [STEP A_3], где v — переменная, A_1, A_2, A_3 — числовые выражения	Используется для обозначения начала цикла, где переменная v принимает значения от A_1 до A_2 с шагом A_3. Фрагмент программы в цикле выполняется $(\frac{A_2 - A_1}{A_3} + 1)$ раз. Если значение A_3 не указано, оно принимается равным 1.
NEXT	NEXT v v — переменная	Используется для обозначения конца цикла. Если результат суммы $v + A_3$ не больше A_2, вычисления производятся в цикле, в противном случае выполняется следующий за NEXT оператор.
STOP	STOP	Используется для временного прерывания выполнения программы.

Команда или оператор	Формат	Область применения
END	END	Используется для обозначения конца программы.
VAC	VAC	Используется для очистки содержимого всех переменных, включая текстовые.
LIST	LIST [номер строки]	Используется для выдачи на индикатор текста программы.
RUN	RUN [номер строки]	Используется для запуска программы на счет.
CLEAR	CLEAR	Используется для стирания обозначенной программы.
CLEAR A	CLEAR A	Используется для стирания всех программ.
MODE	MODE $\left\{\begin{matrix}4\\5\\6\end{matrix}\right\}$	Используется при выборе представления угловых величин.
SET	SET n $1 \leqslant n \leqslant 10$	Используется для задания числа разрядов мантиссы.
LEN	LEN (текстовая переменная)	Используется для определения числа символов в составе текстовой переменной.
MID	MID (m, n)	Используется для выделения символов из состава текстовой переменной.
VAL	VAL (текстовая переменная)	Используется для преобразования текстовой переменной в числовую.
AUTO	AUTO [шаг изменения]	Используется для автоматической нумерации строк программы.

Команда или оператор	Формат	Область применения
GETC	GETC({"ТЕКСТ" / текстовая переменная / выражение}, {число / переменная / выражение})	Используется для выделения одного символа из текстовой переменной
LETC	LETC "ТЕКСТ"	Используется для программирования произвольного символа
CHR	CHR {число / переменная / выражение}	Используется для преобразования целого числа в текстовой символ
ASCI	ASCI "символ"	Используется для преобразования символа в целое число
DRAW DRAWC	DRAW x, y DRAWC x, y x, y — координаты	Используются для установки и стирания точки в поле индикатора соответственно

Примечание. Параметр в квадратных скобках [ ] может быть опущен. Используется один из указанных в фигурных скобках { } параметров.

6.8. Расшифровка команд и операторов

Имя команды или функции	Название	Перевод
ABS	absolute	абсолютный
AC	all clear	полная очистка
ACS	arccos	арккосинус
ANS	answer	ответ
ASCI	ASCI	код ASCI
ASN	arcsin	арксинус
ATN	arctan	арктангенс
AUTO	auto	автоматический
CHR	character	символ

Имя команды или функции	Название	Перевод
CLEAR	clear	стереть (программу)
CLEAR A	clear all	стереть все (программы)
COS	cos	косинус
CSR	cursor	курсор
DEFM	definition of memory	определение памяти
DEG	degree	градус
DEL	deletion	удаление
DRAW	draw a dot	нарисовать точку
DRAWC	clear a dot	стереть точку
EE	entry exponent	ввод в экспоненциальн. форме
END	end	конец
EXE	execution	выполнение
EXT	extension	расширение
EXP	exponent	экспонента
FOR	for	для
FRAC	fraction	дробь
GETC	get character	получить символ
GOSUB	go to the subroutine	идти к подпрограмме
GOTO	go to	идти к
GRA	gradient	град
IF	if	если
INPUT	input	ввести
INS	insertion	внесение
INT	integer	целый
KEY	key	ключ
LEN	lenght	длина

Имя команды или функции	Название	Перевод
LETC	let character	установить символ
LIST	list	листать
LN	ln	логарифм натуральный
LOG	log	логарифм десятичный
MID	middle	середина
MODE	mode	режим
NEXT	next	следующий
PRINT	print	печатать
RAD	radian	радиан
RAN	random	случайный
READY	ready	готов
RETURN	return	возвратиться
RND	rounding	округление
RUN	run	запуск
SET	set	установить
SGN	sign	знак
SIN	sin	синус
SPC	space	пробел
SQR	square root	квадратный корень
STEP	step	шаг
STOP	stop	стоп
TAN	tan	тангенс
TRACE	trace	отслеживать
VAC	variables all clear	полная очистка содержимого переменных
VAL	value	значение
WRT	write	записать

6.9. Таблица кодов символов

Код	Символ	Код	Символ	Код	Символ
0	—	32	пробел	64	@
1	◣	33	!	65	A
2	◢	34	"	66	B
3	◥	35	#	67	C
4	◤	36	$	68	D
5	⧗	37	%	69	E
6	⋈	38	&	70	F
7	{	39	'	71	G
8	}	40	(	72	H
9	◌	41	)	73	I
10	. . .	42	*	74	J
11	○	43	+	75	K
12	Σ	44	,	76	L
13	°	45	—	77	M
14	△	46	.	78	N
15	—	47	/	79	O
16	x	48	0	80	P
17	÷	49	1	81	Q
18	♠	50	2	82	R
19	♥	51	3	83	S
20	♦	52	4	84	T
21	♣	53	5	85	U
22	μ	54	6	86	V
23	Ω	55	7	87	W
24	↓	56	8	88	X
25	←	57	9	89	Y
26	→	58	:	90	Z
27	⚷	59	;	91	[
28	□	60	<	92	\
29	▪	61	=	93	]
30	♭	62	>	94	↑
31	ъ	63	?	95	←

Код	Символ	Код	Символ	Код	Символ
96		128	ю	160	Ю
97	a	129	а	161	А
98	в	130	б	162	Б
99	c	131	ц	163	Ц
100	d	132	д	164	Д
101	e	133	е	165	Е
102	f	134	ф	166	Ф
103	g	135	г	167	Г
104	h	136	х	168	Х
105	i	137	и	169	И
106	j	138	й	170	Й
107	k	139	к	171	К
108	l	140	л	172	Л
109	m	141	м	173	М
110	n	142	н	174	Н
111	o	143	о	175	О
112	p	144	п	176	П
113	q	145	я	177	Я
114	r	146	р	178	Р
115	s	147	с	179	С
116	t	148	т	180	Т
117	u	149	у	181	У
118	v	150	ж	182	Ж
119	w	151	в	183	В
120	x	152	ь	184	Ь
121	y	153	ы	185	Ы
122	z	154	з	186	З
123	ε	155	ш	187	Ш
124	ŋ	156	э	188	Э
125	ε̄	157	щ	189	Щ
126	≥	158	ч	190	Ч
127	■	159	ё	191	Ё

Гарантийные обязательства

Микрокомпьютер "Электроника МК 85" ("Электроника МК 85М") должен быть принят отделом технического контроля предприятия-изготовителя.

Изготовитель гарантирует соответствие микрокомпьютера требованиям технических условий 6К0.310.085 ТУ при соблюдении потребителем условий эксплуатации.

Гарантийный срок — 24 месяца с момента продажи микрокомпьютера. При отсутствии в гарантийном и отрывных талонах отметки торгующей организации срок исчисляется со дня выпуска компьютера предприятием-изготовителем.

В случае неисправной работы микрокомпьютера его владелец имеет право на бесплатный ремонт в период гарантийного срока. После ремонта отрывной талон остается в мастерской. Гарантийный срок продлевается на время нахождения микрокомпьютера в ремонте.

Замена микрокомпьютера осуществляется через торговую сеть по заключению ремонтного предприятия в соответствии с действующими правилами обмена.

Микрокомпьютер опломбируется двумя пломбами.

Первая пломба устанавливается на нижней крышке компьютера и ее сохранность дает право на однократную бесплатную замену разрядившихся элементов питания в гарантийный период с изъятием отрывного талона на их замену без восстановления пломбы.

Без предъявления отрывного талона на замену элементов питания и (или) при нарушении или отсутствии пломбы на нижней крышке микрокомпьютера замена элементов ремонтной мастерской производится за счет владельца компьютера или владельцем самостоятельно. Замена элементов питания производится по установленным расценкам.

Вторая пломба устанавливается внутри компьютера и ее сохранность дает право на бесплатный ремонт в гарантийный период. Без предъявления гарантийного талона и (или) при нарушении сохранности внутренней пломбы или пломбы на блоке питания претензии к качеству изделий не принимаются и их гарантийный ремонт не производится. При проведении гарантийного ремонта, не связанного с заменой элементов, при наличии отрывного талона на их замену ремонтное предприятие обязано восстановить пломбу на нижней крышке микрокомпьютера.

В случае выхода микрокомпьютера из строя покупатель должен выслать его в адрес мастерской предприятия-изготовителя.

Расходы, связанные с пересылкой микрокомпьютера для его ремонта в период гарантийного срока, относят за счет предприятия-изготовителя при предъявлении покупателем почтовой квитанции.

Сведения о гарантийных мастерских, осуществляющих ремонт микрокомпьютеров "Электроника МК 85", "Электроника МК 85М"

Мастерская предприятия-изготовителя: 103482, Москва, абонентный ящик № 123, тел. 532-07-07.

При необходимости ремонта Вашего изделия просим записать его заводской номер, обозначенный на задней крышке, в гарантийный и отрывные талоны руководства по эксплуатации.

При отправке микрокомпьютера в адрес мастерской предприятия-изготовителя упаковка должна исключать возможность его механического повреждения при транспортировке. К изделию должен быть приложен точный почтовый адрес отправителя.

Примечание. Адреса гарантийных мастерских указаны в приложении. Адреса ремонтных мастерских можно узнать в любом магазине, где продаются микрокомпьютеры.

СОДЕРЖАНИЕ

1. Комплект поставки 2
2. Основные технические характеристики 2
3. Указания по технике безопасности 3
4. Краткое описание компьютера 3
5. Подготовка компьютера к работе 9
   5.1. Работа от автономного источника питания 9
   5.2. Работа от блока питания 10
6. Порядок работы с компьютером 10
   6.1. Общие сведения 10
      Организация и расширение памяти 10
      Приоритет операций 11
      Разрядность чисел 12
      Вызов результата предыдущего вычисления 13
      Сообщения об ошибках 13
      Ввод информации 14
      Исправление введенной информации 14
   6.2. Использование компьютера в калькуляторном режиме 15
      Примеры вычислений 15
   6.3. Основы программирования 19
      Основные этапы программирования 20
      Переменные и константы 22
      Операторы присвоения 22
      Запись программы 23
      Выполнение программы 24
      Редактирование программы 25
      Отладка программы 28
   6.4. Операторы (команды), используемые в программах 30
      Операторы перехода 30
      Оператор вывода PRINT 33
      Операторы цикла FOR и NEXT 34
      Вложенные циклы 36
      Оператор ввода INPUT 37
      Оператор KEY 37
      Оператор CSR 38
      Оператор SET 39
      Операторы LEN, MID, VAL, GETC 39
      Оператор AUTO 41
      Подпрограммы 41
      Операторы ASCI и CHR 44
      Оператор LETC 45
      Операторы DRAW и DRAWC 45
      Таблица кодирования символов 46

Оператор DEFM 46
6.5. Примеры программ 47
6.6. Список сообщений об ошибках 51
6.7. Список команд и операторов 52
6.8. Расшифровка команд и операторов 55
6.9. Таблица кодов символов 58
Гарантийные обязательства 60
Гарантийный талон 61
Сведения о гарантийных мастерских, осуществляющих ремонт микрокомпьютеров "Электроника МК 85", "Электроника МК 85М" 62
Приложение. Схема электрическая принципиальная микрокомпьютера "Электроника МК 85" ("Электроника МК 85М")