Бейсик для «Микро-80»

Условный оператор IF X THEN Y — один из фундаментальных операторов, имеющихся практически в любом алгоритмическом языке программирования высокого уровня. Работа его заключается в следующем. Проверяется выполнение условия X:

— если X — истинно, то выполняются операторы, стоящие в строке после слова THEN;

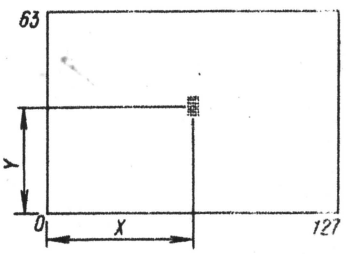
— если X — ложно, то управление будет передано следующей строке программы. (Выражение X считают ложным, если его результат равен нулю, в противном случае X истинно).

Выражение X может включать проверку самых различных условий с использованием как операций отношения, так и логических операций. Операторы, стоящие после слова THEN, также могут быть различными. В частности, если необходимо выполнить оператор GOTO, то название оператора можно опустить и просто указать номер строки программы, которой следует передать управление. Следующий пример иллюстрирует использование условного оператора:

```
ПРИМЕР №11!
===========
IF X=10 THEN 1000
IF A=0 THEN B=1
IF A=5 AND C=7 THEN GOSUB 1000
IF Z=0 OR F=0 THEN L=15:GOTO 250
IF Bx="НЕТ" THEN PRINT "ВЕРНО"
IF Kx="ДА" OR Kx="Д" THEN STOP
```

Особенностью описываемой версии интерпретатора языка Бейсик является наличие операторов, позволяющих формировать графические изображения на экране дисплея.

Оператор CLS предназначен для стирания информации с экрана и всегда должен выполняться прежде, чем начнется формирование графических изображений. В графическом режиме на экране дисплея возможно отображение 128 точек по горизонтали и 64 точек по вертикали. На рис. 1 показан формат экрана в графическом режиме. Точки адресуются в прямоугольной системе координат с началом в левом нижнем углу экрана.



Оператор PLOT X, Y, Z позволяет погасить (или высветить) точку с координатами $0 \leqslant X \leqslant 127$ по горизонтали и $0 \leqslant Y \leqslant 63$ по вертикали. Если операнд Z равен 1, то соответствующая точка засвечивается, 0 — гаснет. Например, следующая программа «рисует» на экране прямую линию с начальными координатами X=50, Y=0 и конечными X=110, Y=60 (обратите внимание на то, что диапазон изменения координат X и Y оператора CUR X, Y вдвое меньше, чем соответствующий диапазон оператора PLOT X, Y):

```
ПРИМЕР №12!
===========
10 CLS
20 FOR I=0 TO 60
30 PLOT 50+I,I,1
40 NEXT I
```

Оператор LINE X, Y позволяет вычерчивать отрезки прямых линий. Операнды X и Y задают координату конечной точки отрезка. Для задания координат начала отрезка, а также его вида (засветка или гашение) служит оператор PLOT:

```
ПРИМЕР №13!
===========
10 CLS
20 PLOT 0,0,1
30 LINE 40,40
40 STOP
```

После выполнения программы на экране появится отрезок прямой линии с координатами начала 0,0 и координатами конца 40,40.

Если следующий отображаемый отрезок должен начинаться там, где закончился предыдущий, то оператор PLOT необходим для задания координат начала только первого отрезка. Так программа выводит на экран дисплея изображение квадрата.

```
ПРИМЕР №14!
===========
10 CLS
20 PLOT 20,20,1
30 LINE 20,50:LINE 50,50
40 LINE 50,20:LINE 20,20
50 STOP
```

Используя описанные операторы, на экране можно создавать разнообразные изображения, в том числе и динамически изменяющиеся. Однако разработка графических программ не простая задача. Для более глубокого ее изучения мы рекомендуем читателю обратиться к литературе [Л].

Среди операторов Бейсика особое место занимают операторы, позволяющие получить доступ к машинным ресурсам — ячейкам памяти и портам ввода-вывода.

Оператор POKE X, Y позволяет записать в ячейку памяти, адрес которой задан выражением X, величину, равную результату выражения Y (значение результата должно находится в диапазоне от 0 до 255 (00H—FFH).

Оператор OUT X, Y позволяет выдать в порт с номером, определяемым выражением X, результат выражения Y. Ограничения на Y такие же, как и в предыдущем операторе

И, наконец, еще один оператор, имеющийся в языке Бейсик. **Оператор CLEAR X** предназначен для очистки памяти от переменных. Если параметр X не указан, то после выполнения оператора всем числовым переменным присваивается значение 0, а всем символьным — значение «пустая строка». Если же параметр X указывается, то в памяти выделяется область размером X байт, предназначенная для хранения

Окончание. Начало см. в «Радио», 1985, № 1, 2.

символьных переменных. Например, CLEAR 500 отводит 500 байт памяти под буфер для символьных переменных. По умолчанию размер этого буфера равен 50 байтам.

Функции в языке Бейсик

В языке Бейсик имеется ряд встроенных функций, которые позволяют значительно упростить написание некоторых программ. Название **встроенная** означает, что в интерпретаторе есть программа обработки данной функции. Существуют функции, работающие с числовыми и символьными данными, функции преобразования типов данных, а также функции обращения к таким машинным ресурсам, как содержимое ячеек памяти и портов ввода-вывода. В этом порядке мы и будем их рассматривать.

В выражениях обращение к функции должно находиться в правой части, например: A=SIN(X)+5.

Таким образом, для вызова функции достаточно указать ее имя в выражении. В круглых скобках указывают аргумент. Результат работы функции (в данном случае значение синуса X) используется в дальнейшем в выражении для вычисления окончательного результата и присвоения полученного значения левой части выражения. Во всех описываемых ниже функциях в качестве аргумента могут выступать константы, имена переменных и выражения, содержащие, в свою очередь, обращения к встроенным функциям.

Таблица 5

ФУНКЦИЯ	РЕЗУЛЬТАТ РАБОТЫ
SQR(X)	КОРЕНЬ КВАДРАТНЫЙ ИЗ X
EXP(X)	ЭКСПОНЕНЦИАЛЬНАЯ ФУНКЦИЯ E
LOG(X)	НАТУРАЛЬНЫЙ ЛОГАРИФМ ОТ X
ABS(X)	АБСОЛЮТНАЯ ВЕЛИЧИНА X ABS(X)= X, ЕСЛИ X>0 0, ЕСЛИ X=0 -X, ЕСЛИ X<0
SGN(X)	ЗНАК X SGN(X)= 1, ЕСЛИ X>0 0, ЕСЛИ X=0 -1, ЕСЛИ X<0
SIN(X)	СИНУС ОТ X (X В РАДИАНАХ)
COS(X)	КОСИНУС ОТ X (X В РАДИАНАХ)
TAN(X)	ТАНГЕНС ОТ X (X В РАДИАНАХ)
ATN(X)	АРКТАНГЕНС ОТ X, РЕЗУЛЬТАТ В РАДИАНАХ
INT(X)	ЦЕЛАЯ ЧАСТЬ ОТ X
RND(X)	СЛУЧАЙНОЕ ЧИСЛО В ДИАПАЗОНЕ ОТ 0 ДО 1

В табл. 5 перечислены встроенные функции языка Бейсик. Почти все они хорошо знакомы читателю по курсу математики средней школы и только две последние требуют дополнительных разъяснений.

Функция INT (X) предназначена для выделения целой части числа. Например, INT (7.6) равно 7, INT (−5.6) равно −6, INT (3) равно 3.

Функция генерации случайного числа RND (X) занимает особое место среди других функций. С ее помощью можно писать на языке Бейсик разнообразные программы моделирования и, как частный случай таких программ,— игровые. Отметим, что аргумент этой функции всегда равен 1. Результат работы функции — случайное число в диапазоне от 0 до 1.

Если необходимо случайное число в другом диапазоне, например от 0 до 100, то достаточно в программе написать A=RND (1) * 100, а чтобы сделать его целым — A=INT (RND (1) * 101).

Тригонометрические функции в Бейсике требуют, чтобы аргумент был указан в радианах. Если необходимо указать угол в градусах, то перевод угла из одной меры в другую легко произвести в программе, воспользовавшись известной формулой X=C * * π/180, где C — значение угла в градусах, X — значение его в радианах.

Как Вы, наверное, уже заметили, в языке реализованы не все тригонометрические функции, однако это не должно вызывать осложнений, так как всегда можно воспользоваться известными тригонометрическими выражениями для определения одних функций через другие.

Язык Бейсик имеет развитые встроенные средства для обработки текстов, что выгодно отличает его от других языков программирования высокого уровня. Рассмотрим встроенные функции для работы с символьной информацией.

Функция LEN (Хö). Результатом ее работы является число, равное количеству символов (длине) переменной Хö. Например, если Хö=«ПАРОХОД», то LEN (Хö) равно 7.

Функция LEFT (Хö,Y) позволяет вывести на экран строку символов длиной Y, начиная с крайнего левого символа. Например, LEFTö (Хö, 3) равно «ПАР».

Функция RIGHTö (Хö, Y) проделывает тоже самое, но начиная с крайнего правого символа. Например RIGHTö (Хö, 3) равно «ХОД».

Функция MIDö (Хö, Y, Z) позволяет вывести строку символов длиной Z, начиная с позиции Y. Отсчет позиций ведется слева направо. Например, MIDö (Хо, 3, 4) равно «РОХО».

В программах на Бейсике часто возникает необходимость перевода данных из числового вида в символьный и наоборот. Этой цели служат две специальные встроенные функции.

Функция STRö (X) служит для преобразования числовых величин в символьный вид. Аргумент X — число или арифметическое выражение, а результат работы функции — строка, являющаяся символьным представлением данного числа. Например, пусть X=10^7, тогда STRö (X) равно «1E+07»; если X=−3,15, то STRö (X) равно «−3.15».

Функция VAL(Хö) предназначена для обратного преобразования данных из символьного вида в числовой, начиная с крайнего левого символа переменной Хö. Если в строке встречается недопустимый символ (не цифра и не знак числа), то преобразование прекращается. Если уже первый символ является недопустимым, то результат работы равен 0. Например, если Хö= =«13 ШТУК», то VAL (Хö) равно 13; если Хö=«ШТУК 13», то VAL (Хö) равно 0.

Как известно, каждому символу соответствует определенный семиразрядный код (см. табл. 2 в «Радио», 1983, № 8, с. 26). В Бейсике есть две встроенные функции для работы с кодами символов.

Функция ASC (Хö) переводит код первого символа Хö в десятичный вид. Например, если Хö=«D», то ASC (Хö) равно 68.

Функция CHRö(X) позволяет вывести на экран символ, код которого равен X (аргумент функции не должен превышать 255). Например, CHRö(68) равно «D». Эту функцию удобно использовать для выдачи на экран дисплея различных управляющих символов. Например, PRINT CHRö(12) приведет к перемещению курсора в левый верхний угол экрана.

Для управления форматом печати результатов на экране дисплея служат следующие три функции.

Функция POS (1), результатом работы которой является целое число, равное номеру позиции последнего отпечатанного символа в текущей строке. Например: PRINT "ABCDE". В этом случае POS(1) равно 5.

Функция SPC(X) позволяет вставлять в печатаемую строку X пробелов (аргумент X не должен превышать 255). Например, в результате обработки строки PRINT «Транзистор»; SPC(5); «КТ315Г», будет напечатано: ТРАНЗИСТОР _____ КТ315Г (_ пробел).

Функция TAB (X) — это функция горизонтальной табуляции, которая позволяет переместить курсор в заданную позицию в строке. Аргумент X и в этом случае не должен превышать 255. Он указывает, сколько позиций необходимо отступить от левого

края строки. Например, после выполнения операторов PRINT TAB (5); «A»; TAB (10); «B» символ «A» будет напечатан в 6-м знакоместе, а «B» — в 11-м.

При разработке больших программ, а также программ обработки текстов необходимо иметь представление об объеме свободной памяти, не занятой текстом программы и переменными. Для этих целей в языке Бейсик предусмотрена функция FRE(X). Если аргумент этой функции число, то результатом будет число свободных байтов в памяти. Если аргумент — символьное выражение, то результат — число свободных байтов в буфере для символьных переменных. Например, после выполнения оператора PRINT FRE (0) на экран будет выведено число свободных байтов в памяти.

Выше уже отмечалось, что интерпретатор позволяет разрабатывать программы управления различными объектами. Управляющие программы обычно содержат критичные по времени выполнения фрагменты и требуют также возможности непосредственного манипулирования с содержимым ячеек памяти и портов ввода-вывода. Следующие встроенные функции Бейсика и служат для этих целей.

Результат работы функции PEEK(X) — десятичное число, равное содержимому ячейки памяти, адрес которой определен аргументом X. Например, в результате выполнения оператора PRINT PEEK (0) будет напечатано 49.

Обратившись к функции INP(X), получим десятичное число, равное содержимому порта ввода с номером X ($X \leqslant 255$), например, A=INP(1). Переменной A в этом случае будет присвоено значение, равное содержимому порта номер 1.

Функция USR(X) предназначена для организации связи программ, написанных на Бейсике, с подпрограммами, написанными на ассемблере или в машинных кодах. Аргумент X — это адрес ячейки памяти, начиная с которой записана программа в машинных кодах. Поэтому, если в выражении встретится обращение к функции USR(X), то управление будет передано подпрограмме, расположенной по адресу X.

В конце подпрограммы обязательно должна стоять команда RET, после выполнения которой управление возвращается программе на Бейсике. Результат работы подпрограммы (в виде одного байта) перед возвратом из нее помещается в аккумулятор. Функция USR(X) позволяет критичные по времени и специфике работы фрагменты алгоритма реализовывать на ассемблере, а основную программу писать на языке Бейсик. Для передачи параметров для подпрограмм и результатов в программу на Бейсике можно воспользоваться оператором POKE X, Y и функцией PEEK(X).

Подпрограммы целесообразно размещать в «защищенной» области памяти (т. е. в той, которую заведомо не использует интерпретатор). Если в функциях и операторах, работающих с адресами памяти, адрес превышает значение 32767 (7FFFH), то он должен указываться в виде отрицательного числа. Адресу FFFFH будет соответствовать —1, адресу FFFEH — —2 и т. д.

Рассмотрим пример. Пусть микро-ЭВМ «Микро-80» используется для обучения детей таблице умножения. Очередной вопрос появляется на экране в виде текстового сообщения. Обучаемый отвечает на вопрос, нажимая соответствующие клавиши на клавиатуре дисплея. Если ответ верен, то в качестве поощрения микро-ЭВМ «исполняет» несколько первых тактов популярной детской песенки. «Исполняет» мелодию подпрограмма, написанная на языке ассемблера, генерирующая соответствующую последовательность импульсов. Сделать это в программе на Бейсике не представляется возможным из-за временных ограничений, присущих интерпретатору. Вызов подпрограммы происходит с помощью функции USR(X) в соответствующем месте основной программы, написанной на Бейсике. Таким образом, разумно сочетая возможности программирования на ассемблере и на языке высокого уровня, можно получить требуемый результат.

Кроме перечисленных встроенных функций, в Бейсике имеется возможность вводить в текст программы определение новых функций и в дальнейшем обращаться к ним по имени в различных выражениях. Определить функцию можно в любом месте программы с помощью **оператора DEF.** Имена всех функций должны начинаться обязательно с символов FN, за которыми могут следовать один или два любых символа. Ограничения на них такие же, как и на имена переменных; например, FNA, FNX1 — допустимые имена, ES2, EKA — недопустимые.

Определим, например, функцию FNCT(X) следующим образом:

10 DEFFNCT(X)=COS(X)/SIN(X).

В соответствии с этим определением функция FNCT — это тригонометрическая функция котангенс, не реализованная в виде встроенной в интерпретаторе.

Оператор DEF можно использовать только в программном режиме. Кроме того, допускается определение функций только от одного числового аргумента. Параметр X, стоящий в операторе определения, является формальным, необходимым для обозначения функциональной зависимости. При обращении к функции вместо него указывается фактический аргумент, который заменяет формальный параметр, стоящий в правой части оператора присваивания. Например, после выполнения строки программы 20 PRINT FNCT (2) на экране будет напечатано значение котангенса для угла, равного 2 радианам.

Использование возможности определения функций самим программистом позволяет сократить текст программ и значительно улучшить их читаемость.

Сообщения об ошибках

Интерпретатор языка Бейсик позволяет в процессе выполнения программы обнаруживать и анализировать ошибки. Интерпретатор, естественно, не может обнаружить логические ошибки. Это под силу только программисту, так как только он знает, что должна делать программа. О каких же ошибках тогда идет речь? Это будет ясно из дальнейшего.

Если ошибка обнаружена в непосредственном режиме, то на экран выводится сообщение «?XX ОШИБКА», где XX — двузначный код ошибки. Если ошибка обнаружена в программном режиме, то выводится сообщение «?XX ОШИБКА В N», где XX — код ошибки, а N — номер строки, в котором она обнаружена.

После сообщения об ошибке появляется символ «=>», означающий, что интерпретатор готов к приему директив, и программист может внести изменения в программу и продолжить отладку.

Рассмотрим, какие ошибки обнаруживает интерпретатор.

Ошибка 01. В программе встретился оператор NEXT, для которого не был выполнен соответствующий оператор FOR.

Ошибка 02. Неверный синтаксис.

Ошибка 03. В программе встретился оператор RETURN без предварительного выполнения оператора GOSUB.

Ошибка 04. При выполнении программы не хватает данных для оператора READ, т. е. данных, описанных операторами DATA меньше, чем переменных в операторах READ.

Ошибка 05. Аргумент функции не соответствует области определения данной функции. Например, отрицательный или нулевой аргумент функции LOG(X), отрицательный аргумент у функции SQR(X) и т. д.

Ошибка 06. Переполнение при выполнении арифметических операций. Результат любой операции не может быть больше $+1{,}7\cdot10^{38}$ или меньше $-1{,}7\cdot10^{38}$.

Ошибка 07. Недостаточен объем памяти. Возможные причины: велик текст программы; слишком длинны массивы данных; вложенность подпрограмм и циклов больше нормы; слишком много переменных.

Ошибка 08. Нет строки с данным номером. Возникает при выполнении операторов перехода.

Ошибка 09. Индекс не соответствует размерности массива.

Ошибка 10. Повторное выполнение операторов DIM или DEF, описывающих массив или функцию, которые уже были описаны ранее.

Ошибка 11. Деление на ноль.

Ошибка 12. Попытка выполнить операторы INPUT или DEF в непосредственном режиме.

Ошибка 13. Несоответствие типов данных. Возникает при попытке символьной переменной присвоить числовое значение и наоборот.

Ошибка 14. Переполнение буферной области памяти, отведенной для хранения символьных переменных. Для расширения объема буфера служит директива CLEAR.

Ошибка 15. Длина символьной переменной превышает 255.

Ошибка 16. Выражение, содержащее символьные переменные, слишком сложно для интерпретатора.

Ошибка 17. Невозможность продолжения выполнения программы по директиве CONT.

Ошибка 18. Обращение к функции, не описанной оператором DEF.

Кроме описанных, интерпретатор выдает еще три сообщения об ошибках в случае неверного набора данных при выполнении оператора INPUT:

«?ПОВТОРИТЕ ВВОД» — указывает на ошибку в наборе данных. Вместо числа набрана строка символов и наоборот.

«? ЛИШНИЕ ДАННЫЕ» — данных набрано больше, чем переменных в операторе INPUT. Лишние данные просто игнорируются.

«??» — данных набрано меньше, чем переменных в операторе INPUT. Необходимо ввести недостающие данные.

В дальнейшем предполагается опубликовать несколько программ. Их анализ поможет Вам разобраться в методике написания программ на языке Бейсик.

Г. ЗЕЛЕНКО,
В. ПАНОВ, С. ПОПОВ

г. Москва

ЛИТЕРАТУРА

Гилой В. Интерактивная машинная графика.— М.: Мир, 1981.

Радиоконструктор «УНЧ предварительный»

Этот радиоконструктор (другое его название «Старт 7173») входит в серию наборов, которые выпускает для самостоятельного конструирования домашних радиокомплексов завод «Электроприбор» им. 60-летия СССР (г. Каменец-Подольский Хмельницкой области). В этой серии, правда, есть уже один предварительный усилитель звуковой частоты (см. «Радио», 1983, № 5, с. 57), выполненный на операционном усилителе К140УД1Б. Новый радиоконструктор во многом отличается от своего предшественника. Во-первых, в нем применены микросхемы К548УН1Б, которые разработаны специально для использования в каскадах предварительного усиления звуковой частоты и имеют низкий собственный уровень шумов. Во-вторых, печатная плата набора рассчитана на сборку двухканального усилителя, что, конечно, удобно для использования его в стереофоническом комплексе. И, наконец, в новом усилителе применены пассивные регуляторы тембра. Это может показаться странным — ведь в последние годы многие радиолюбители отдают предпочтение активным регуляторам. Однако все большее число конструкторов приходит к убеждению, что использование в звуковоспроизводящем тракте пассивных цепей частотной коррекции обеспечивает лучшие характеристики с точки зрения минимизации искажений сигнала.

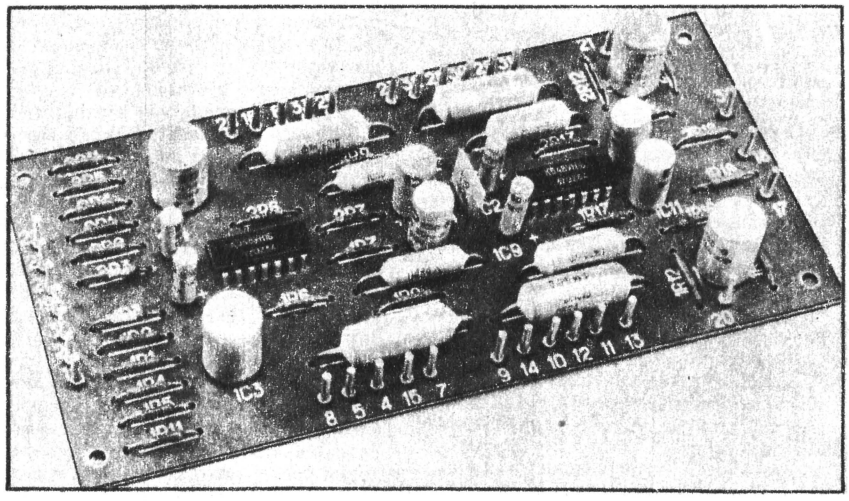
Номинальное выходное напряжение, В	1
Коэффициент гармоник на частотах 200, 5000 и 16 000 Гц при номинальном выходном напряжении, %, не более	0,05
Относительный уровень шумов, дБ, не более	—65
Глубина регулировки тембра на частотах 40 и 16 000 Гц, дБ, не менее	±12
Напряжение питания, В	12...18
Габариты, мм	95×166×27
Масса, г	150

Сопротивление нагрузки (т. е. входное сопротивление усилителя мощности) должно быть не менее 10 кОм.

Помимо регулировок тембра по высшим и низшим звуковым частотам, а также уровня сигнала, в радиоконструкторе «Старт 7173» предусмотрена регулировка стереобаланса (заметим, что переменные резисторы в набор не входят — их надо приобрести отдельно). Цена набора — 9 р. 70 к.

●

Как сообщили редакции, Центральная торговая база (ЦТБ) Роспосылторга открыла предварительный прием заказов на радиоконструкторы «Старт 7173», «Старт 7174», «Старт 7175» (выполнение заказов — по мере поступления наборов с предприятия-изготовителя).



Предварительный усилитель, собранный из набора «Старт 7173», обладает следующими техническими характеристиками:

Номинальный диапазон частот, при неравномерности АЧХ ±1,5 дБ, Гц	20...20 000
Номинальное входное напряжение, мВ (входное сопротивление, кОм), входа:	
«Звукосниматель»	250 (400)
«Радиоприемник»	20 (40)
«Микрофон»	3 (4,7)

Кроме того, ЦТБ начала предварительный прием заказов на следующую аппаратуру и наборы: «Электроника-10-стерео» (см. «Радио», 1984, № 9, с. 37) — цена 75 руб.; генератор телевизионных испытательных сигналов ГИС-01Т (см. «Радио», 1984, № 11, с. 64) — цена 75 руб.; осциллограф ОМЛ-2М (модифицированный вариант осциллографа ОМЛ-2-76; см. «Радио», 1981, № 2, с. 29) — цена 125 руб.

Адрес базы: 111126, г. Москва, Е-126, Авиамоторная ул., 50, ЦТБ Роспосылторга.