



В. К. АЛИЕВ

В.К. АЛИЕВ

Языки Бейсик

Данная книга - самое полное справочное пособие по языку Basic, одновременно является учебником по программированию на языке QBASIC. Особенное внимание уделено практическим примерам программирования - примеры программ (написанные учителями и учащимися старших классов), приведенные в книге, отлажены и проверены в средах Visual Basic и QBasic, поэтому книгу можно использовать как справочник по технике и навыкам программирования.

Предназначена для самого широкого круга читателей-пользователей персонального компьютера, приступающих к изучению или занимающихся программированием - школьников старших классов, учащихся средних специальных учебных заведений, студентов техникумов и вузов, аспирантов, программистов, специализирующихся на VISUAL Basic, независимо от версии.

Данная книга принимала участие в ярмарке педагогических достижений во Владивостоке в октябре 1999 года, где завоевала диплом второй степени в номинации "Учебные пособия и методические разработки".

Издательство "Солон-Р"

129337, г. Москва, а/я 5

Телефоны:

(499) 254-44-10, (499) 252-36-96

E-mail: Solon.Pub@relcom.ru

Ответственный за выпуск С. Иванов

Макет и верстка В. Смирнов

Обложка Е. Жбанов

ISBN 5—93455—054—3

© "СОЛОН - Р", 2009

© В.К. Алиев, 2009

Глава V. Вывод данных на экран монитора

Помимо приведенных в главе 3.3 операторов PRINT и CLS используются еще несколько операторов вывода данных на экран дисплея.

5.1. Оператор LOCATE

Оператор LOCATE устанавливает курсор в нужное место на экране дисплея и позволяет управлять параметрами самого курсора.

Формат записи оператора LOCATE:

```
LOCATE [номер_строки] [ , [номер_колонки] ] [{ , 0, 1 }]  
      [ , [начало_курсора] [ , конец_курсора] ]
```

где

Номер_строки — задает место, куда помещается курсор. Значение

Номер_колонки — числового выражения, задающее номер_строки, находится в пределах от 1 до 25; значение числового выражения, задающего номер_колонки задается в пределах от 1 до 40 или 80 в зависимости от установленной длины строки экрана дисплея.

Если третий позиционный параметр равен 0, то курсор будет невидим на экране во время выполнения программы; 1-видим. По умолчанию этот параметр равен 1.

Параметры – **начало_курсора** (верхняя линия) и **конец_курсора** (нижняя линия) задают количество линий сканирующего луча, которые формируют толщину курсора. Значения этих параметров лежат в пределах от 0 до 31. Линии нумеруются сверху вниз от 0 до 13 для алфавитно-цифрового и от 0 до 7 для графического дисплеев.

В результате выполнения оператора LOCATE курсор будет установлен в указанную строку и колонку с заданной видимостью и толщиной. Толщиной курсора можно управлять только в текстовом режиме. Если какой-то операнд опущен, то будет использоваться значение, которое установлено по умолчанию или предыдущим оператором LOCATE.

5.2. Работа с графической информацией

Программные средства языка Бейсик позволяют работать не только с текстовой, но и с графической информацией.

Элементарным изобразительным средством графического экрана является черно-белая или цветная точка разной степени свечения. Координаты точки определяются прямоугольной системой координат и разрешающей способностью экрана дисплея. Разрешающая способность экрана задается оператором SCREEN.

5.2.1. Оператор SCREEN

Оператор SCREEN устанавливает режим и параметры работы экрана дисплея.

Формат записи оператора SCREEN:

```
SCREEN {1,2} [, [c1] [, [активная_страница] [, отобр_стр] ] ]
```

где

первый операнд, равный 1, устанавливает графический режим с разрешающей способностью 320 точек по горизонтали и 200 точек по вертикали. Число 2 устанавливает графический режим с разрешающей способностью 640x200. Изображение на экране может быть черно-белым или цветным в зависимости от операнда c1. Если c1=0, то изображение цветное

При разрешающей способности 320x200 горизонтальная строка содержит 320 точек, которые нумеруются слева направо от x=0 до x=319 и вертикально можно расположить 200 точек, которые нумеруются сверху вниз от y=0 до y=199.

Местоположение каждой точки задается ее координатами. Существуют два способа задания координаты точки: абсолютная и относительная.

Абсолютная форма задания координаты точки имеет вид (x,y). Координата точки, которая находится в левом верхнем углу экрана, равна (0,0); координаты точки в правом верхнем углу экрана при SCREEN 1 равны (319,0) и (639,0) при SCREEN 2; координата точки в нижнем левом углу при любом SCREEN равна (0,199).

Относительная форма предполагает задание координат очередной точки относительно последней выведенной точки, осуществляемое с помощью оператора STEP (x1,y1), где x1 и y1 — значения, которые прибавляются к координатам последней выведенной точки.

5.2.2. Оператор PSET

Оператор PSET (Point SET-установить точку) позволяет изобразить точку в указанной позиции экрана дисплея заданного цвета.

Формат записи оператора PSET:

```
PSET (x,y) [, цвет]
```

где

x и y — координаты точек в абсолютной или относительной форме;
цвет — целое число от 1 до 31, которое определяет цвет и насыщенность точки:

0-черный	1-голубой	2-зеленый
3-бирюзовый	4-красный	5-малиновый
6-коричневый	7-белый	8-серый
9-светло-голубой	10-светло-зеленый	11-светло-бирюзовый
12-светло-красный	13-светло-малиновый	14-желтый

15-белый насыщенный	16 и выше-вариации белого	
---------------------	---------------------------	--

5.2.3. Оператор PRESET

Оператор PRESET (Point RESET-изменить яркость точки) позволяет изобразить точку в указанной позиции экрана дисплея заданного цвета.

Формат записи оператора PRESET:

```
PRESET (x, y) [, цвет]
```

где

x и **y** — координаты точек в абсолютной или относительной форме;
цвет — целое число от 1 до 31, которое определяет цвет и насыщенность точки:

Операторы PSET и PRESET отличаются тем, что в первом операторе цвет соответствует цвету переднего плана, а во втором операторе параметр цвет соответствует цвету фона, то есть, если в операторе PRESET цвет не указан, то он гасит точку, изображенную на этом месте оператором PSET.

В случае ошибочного задания координат эти операторы не выполняются. Сообщения об ошибке не выдаются.

5.2.4. Оператор LINE

Оператор LINE позволяет начертить отрезок (линию) или прямоугольник.

Формат записи оператора LINE:

```
LINE [(x1, y1)] - (x2, y2) [, [цвет] [, B[F]] [, маска]]]
```

где **(x1,y1)** и **(x2,y2)** — координаты начала и конца отрезка, заданные в абсолютной или относительной форме. Если параметры **B** и **F** опущены, то в результате выполнения оператора LINE начальная и конечная точки соединяются прямой линией. Если операнд **(x1,y1)** отсутствует, то в качестве начальной точки используются координаты точки, которая последней участвовала в обработке. После выполнения оператора LINE система запоминает координаты точки **(x2,y2)**.

— параметр **цвет** определяет цвет линии, если до этого оператором SCREEN был установлен соответствующий режим экрана.

— если указан параметр **B**, то система строит прямоугольник. В этом случае параметры **(x1,y1)** и **(x2,y2)** задают координаты противоположных вершин прямоугольника.

— Параметр **F** окрашивает внутреннюю область прямоугольника в указанный цвет. В этом случае параметр цвет на цвет линии не действует.

Пример 5.2.4(1) Построение прямоугольника.

```
'Построение прямоугольника с помощью четырех линий
CLS                                'очистка экрана
SCREEN 12                         'установка параметров экрана
```

```

LINE (200, 50)-(400, 50), 5 'верхняя линия-малиновая
LINE - (400, 100), 1         'правая сторона-голубая
LINE - (200, 100), 2         'нижняя линия-зеленая
LINE - (200, 50), 3          'левая сторона-бирюзовая
LINE (200, 50)-(400, 100), 6 'диагональ с правого верхнего
                               угла-коричневая
END

```

Пример 5.2.4(2) Построение вложенных прямоугольников.

```

'Построение вложенных прямоугольников
CLS                               'очистка экрана
SCREEN 12                        'установка параметров экрана
LINE (200, 155)-(600, 400), 1, BF 'построение голубого
                                   прямоугольника
LINE (240, 165)-(560, 390), 2, BF 'построение зеленого
                                   прямоугольника
LINE (280, 175)-(520, 380), 6, BF 'построение коричневого
                                   прямоугольника
LINE (320, 185)-(480, 370), 4, BF 'построение красного
                                   прямоугольника
END

```

Пример 5.2.4(3) Построение осей координат.

```

'Построение осей координат для графиков
CLS                               'очистка экрана
a1: INPUT "введите крайние симметричные значения по оси"1_
      X(X<0 и X>0)"; x1, x2
IF x1 = 0 OR x2 = 0 THEN GOTO a1 'проверка условий
a2: INPUT "введите крайние симметричные значения по оси"1_
      Y(Y<0 и Y>0)"; y1, y2
IF y1 = 0 OR y2 = 0 THEN GOTO a2 'проверка условий
SCREEN 12: CLS                  'установка параметров экрана
LINE (0, 199)-(639, 199)       'построение оси X
LINE (319, 0)-(319, 399)       'построение оси Y
'ПОСТРОЕНИЕ ЕДИНИЧНЫХ ОТРЕЗКОВ ПО ОСИ X
LINE (0, 197)-(0, 201)         'построение первого отрезка
FOR i = 15 TO 639 STEP 16      'начало цикла формирования
                               отрезков длиной 4
'построение единичных отрезков по оси X
LINE (i, 197)-(i, 201)

```

¹ Командная строка Бейсика допускает длину 256 символов, что вполне хватает для любого выражения Бейсика, хотя длинные строки менее комфортны на экране. В данной же книге автор постоянно сталкивается с проблемой переноса программных строк, зная по собственному опыту, как порой один лишний символ в ненужном месте делает хорошо отлаженную программу-малопонятным набором инструкций. Именно поэтому все тексты программ автором лично проверены на работоспособность и до сих пор переносы делались только в комментариях. Но все-таки этот момент наступил и автор вынужден идти на некоторое соглашение, а именно: в дальнейшем, как и в данном случае, для переноса программных строк будет применяться символ “_” — подчеркивание, который выбран не случайно, а потому, что именно так переносятся строки в Visual Basic.

```

NEXT i 'конец цикла
'ПОСТРОЕНИЕ ЕДИНИЧНЫХ ОТРЕЗКОВ ПО ОСИ Y
LINE (317, 0)-(321, 0) 'построение первого отрезка
FOR i = 9 TO 399 STEP 9 'начало цикла формирования
                        отрезков с шагом 9
'построение единичных отрезков с длиной 4
LINE (317, i)-(321, i)
NEXT I 'конец цикла
'ВЫВОД ЗНАЧЕНИЙ КРАЙНИХ ТОЧЕК КООРДИНАТ
LOCATE 12, 1: PRINT x1 'установка курсора в точку x1
LOCATE 12, 78: PRINT x2 'установка курсора в точку x2
LOCATE 24, 42: PRINT y1 'установка курсора в точку y1
LOCATE 2, 42: PRINT y2 'установка курсора в точку y2
'Расчет цены деления
xx = x2 / 10: yy = y2 / 10
LOCATE 26, 52: PRINT "цена деления по оси X — "; xx
LOCATE 27, 52: PRINT "цена деления по оси Y — "; yy
LOCATE 1, 1 'установка курсора в начало экрана
END

```

5.2.5. Оператор CIRCLE

Оператор CIRCLE позволяет получить на экране дисплея эллипс(круг).

Формат записи оператора CIRCLE:

```
CIRCLE (x,y), r[, [цвет] [, [начало] [, [конец] [, [сжатие] ] ] ]
```

где

- (x,y) — координаты центра эллипса, которые можно задавать в абсолютной или относительной форме;
- r — любое числовое выражение, значение которого задает большую полуось эллипса.

Значение цвета выбирается из текущей палитры с номером цвета.

0-черный	1-голубой	2-зеленый
3-бирюзовый	4-красный	5-малиновый
6-коричневый	7-белый	8-серый
9-светло-голубой	10-светло-зеленый	11-светло-бирюзовый
12-светло-красный	13-светло-малиновый	14-желтый
15-белый насыщенный	16 и выше-выриации белого	

Операнд **начало** задает начальную точку дуги, **конец** — конечную.

Операнды **начало** и **конец** могут быть любыми числовыми выражениями в диапазоне от -2π до $+2\pi$, где $\pi=3.141593$. Эти операнды позволяют получить дугу эллипса.

Операнд **сжатие** задает отношение радиуса, параллельного оси у, к радиусу, параллельному оси х. Оси эллипса всегда параллельны осям координат.

Если значения операндов начало и конец отрицательные, то используются их абсолютные величины и крайние точки дуги эллипса соединяются линией с центром. Если операнды начало и конец опущены, то строится полный эллипс.

Пример 5.2.5(1)

```
'Построение окружностей
SCREEN 12
CLS
CIRCLE (200, 200), 200
CIRCLE (200, 200), 200, 5, , , 3 / 10
CIRCLE (200, 200), 200, 2, , , 4 / 10
CIRCLE (200, 200), 200, 1, , , 5 / 10
END
```

Пример 5.2.5(2)

```
'Построение серии окружностей в цикле
CLS 'Очистка экрана
SCREEN 12 'установка разрешающей способности
'экрана
FOR r = 1 / 20 TO 10 STEP .1
с = с + 1
IF с > 12 THEN с = 0
CIRCLE (320, 200), 150, 1 + с, , , r 'построение эллипса с цветом
'1+с
NEXT r
PSET (320, 200), 5
END
```

Пример 5.2.5(3)

```
'Построение серии эллипсов в цикле
CLS 'Очистка экрана
SCREEN 12 'установка разрешающей способности экрана
FOR r = 1 TO 1 / 10 STEP -.1 'начало цикла построение эллипсов
с = с + 1 'переменная для изменения цвета
IF с > 12 THEN с = 0 'которая изменяется по модулю 12
CIRCLE (320, 200), 150, 1 + с, , , r 'построение эллипса с цветом
'1+с
PAINT (320, 200), 1 + с, 1 + с 'окрашивание эллипса
NEXT r 'конец цикла
PSET (320, 200), 5 'устанавливаем точку в центре
'эллипса
END
```


Глава VI. Операторы работы с экраном монитора

6.1. Преобразование системы координат экрана дисплея

Оператор VIEW(поле зрения) задает прямоугольную область (“окно”) на экране монитора, которое становится текущим экраном. В этом экране будут выполняться все графические построения. Оператор WINDOW устанавливает новую систему координат окна.

6.1.1. Оператор VIEW

Оператор VIEW выделяет на экране монитора “окно”, которое становится новым экраном, доступным для работы. В пределах этого “окна” выполняются все графические построения, а остальные точки экрана становятся недоступными.

Формат записи оператора VIEW:

```
VIEW [ [SCREEN] [ (x1,y1) - (x2,y2) [, [цвет_окна] [, [цвет_контура] ] ] ] ]
```

где

(x1,y1) и (x2,y2) — координаты левого верхнего и правого нижнего углов окна. В операторе можно использовать любые допустимые значения координат x и y, при условии, что $x1 \neq x2$ и $y1 \neq y2$. Если все операнды опущены, то “окном” является весь экран.

6.1.2. Оператор WINDOW

Оператор WINDOW устанавливает систему координат окна.

Формат записи оператора WINDOW:

```
WINDOW [ [SCREEN] [ (x1,y1) - (x2,y2) ]
```

Если операнд Screen задан, то создается стандартная система координат, то есть x1 и y1 задают координату левого верхнего угла окна, а x2 и y2 задают координату правого нижнего угла окна. Если же операнд Screen опущен, то создается декартова система координат, где x1 и y1 задают координаты левого нижнего угла окна.

При выполнении оператора WINDOW система изменяет координаты x и y независимо от их написания так, чтобы наименьшие значения x и y стали значениями (x1,y1). В операторе можно использовать любые допустимые значения x и y при условии $x1 \neq x2$, $y1 \neq y2$.

Операторы VIEW и WINDOW без операндов отменяют систему координат, определенную ранее оператором WINDOW с операндами.

Пример 6.1.2

```
'программа поиска корней многочлена на числовом отрезке
CLS
```

```

DIM min AS DOUBLE 'начало интервала поиска корня
DIM max AS DOUBLE 'конец интервала поиска корня
DIM x AS DOUBLE   'переменная
DIM y1 AS DOUBLE  'функция
'DIM y2 AS DOUBLE  'функция
DIM h AS DOUBLE   'точность вычислений
DIM x1 AS DOUBLE  'корень
SCREEN 9
VIEW (340, 10)-(620, 200), 6, 7 'правое верхнее окно
WINDOW (-14, -9)-(14, 9) 'устанавливаем систему координат
LINE (-14, 0)-(14, 0), 3 'строим оси
LINE (0, -9)-(0, 9), 3 'координат
FOR i = -13 TO 13 'координатные отрезки
LINE (i, -.5)-(i, .5), 4 'по оси x
NEXT i
FOR x = -10 TO 10 STEP .01 'построение графика функции
y = x ^ 3 + 5 * x ^ 2 + 3 * x - 8 'в окне
PSET (x, y), 1
NEXT x
'поиск корней с указанной точностью
LOCATE 15,1
a: PRINT "введите интервал": INPUT "MIN="; min: INPUT "MAX=";
max
a001: PRINT "введите точность": INPUT "шаг="; h
IF h = 0 THEN
PRINT "ТОЧНОСТЬ СЛИШКОМ ВЕЛИКА ДЛЯ МЕНЯ"
GOTO a001
END IF
IF min >= max THEN
PRINT "min должен быть меньше max! Повторите ввод!"
GOTO a
END IF
y1 = min ^ 3 + 5 * min ^ 2 + 3 * min - 8 'вычисляем значение
'ф-ции в начале интервала
y2 = max ^ 3 + 5 * max ^ 2 + 3 * max - 8 'вычисляем значение
'ф-ции в конце интервала
IF y1 * y2 < 0 THEN
PRINT "в этом интервале существует по крайней мере один корень"
ELSE
PRINT "в этом интервале корней нет": GOTO a
END IF
'поиск корня — начало цикла — приближение к корню слева
a01: DO WHILE y1 * y2 < 0
min = min + h
y1 = min ^ 3 + 5 * min ^ 2 + 3 * min - 8
x1 = min
LOOP 'конец цикла
a1: PRINT "функция="; y1
PRINT "при x="; x1
INPUT "будете уточнять?(0-нет, точность вычисления — да)"; h
min = x1
IF h = 0 THEN GOTO m2 'больше не вычисляем

```

```
IF flag% = 0 THEN flag% = 1: GOTO m1 ELSE flag% = 0: GOTO a01
'приближение к корню справа
m1: DO WHILE y1 * y2 > 0
    min = min - h
    y1 = min ^ 3 + 5 * min ^ 2 + 3 * min - 8
    x1 = min
    LOOP: GOTO a1
m2:
    END
```

Глава VII. Управляющие операторы языка Бейсик

До сих пор, рассматривая операторы языка Бейсик и примеры программ, написанных на этом языке, мы обходили молчанием некоторые важные моменты, без понимания которых программирование не существует. Одним из таких основополагающих понятий является ПРОГРАММА для вычислительной машины, основные принципы которой разработала леди Байрон, дочь английского драматурга. Главное, что надо усвоить из этих принципов, которые по существу являются открытиями, это то, что:

1. программа является строго определенной последовательностью команд;
2. команды, из которых состоит программа, условно делятся на выполняемые и управляющие;
3. вычислительная машина выбирает команды программы для выполнения последовательно одну за другой;
4. последовательное выполнение команд может быть нарушено управляющей командой, которая указывает, какая команда выполняется следующей.

Таким образом вводятся понятия выполняемых и управляющих операторов языка программирования.

Выполняемый оператор — это такой оператор, который не нарушает последовательность выполнения операторов программы.

Управляющий оператор — это такой оператор, который может изменить последовательность выполнения операторов программы. И пусть читателя не смущает выражение “который может изменить последовательность выполнения”, которое введено лишь для того, чтобы учесть ситуацию, когда управление передается на оператор, следующий за управляющим.

В языке Бейсик довольно приличный набор управляющих операторов. Но поскольку в задачи данной книги не входит научить писать структурированные программы, а поставлена цель научить пользоваться операторами языка Бейсик, мы рассмотрим следующие управляющие операторы. Наиболее полно операторы Бейсика представлены в главе 11.3. “Ключевые слова Бейсика” и главе XII “Алфавитный указатель операторов, ключевых слов, функций языка Бейсик”.

Табл.7.1.

Оператор	Выполняемое действие
END	Закончить выполнение программы
GOTO	Обеспечить безусловный переход
IF-GOTO	Условный переход
IF-THEN-ELSE	Условный оператор
FOR-TO-NEXT	Арифметический цикл
WHILE-WEND	Итерационный цикл1

DO-LOOP	Безусловный цикл
DO-WHILE(UNTIL)-LOOP	Итерационный цикл2
DO-LOOP-WHILE(UNTIL)	Итерационный цикл3
SELECT-CASE-ENDSELECT	Структура с условием
GOSUB-RETURN	Перейти в(вернуться из) подпрограммы
ON-GOTO	Выбрать безусловный переход
ON-GOSUB	Выбрать подпрограмму
ON-ERROR	Перейти к подпрограмме обработки ошибок
STOP	Временно остановить выполнение программы
DEF	Определить функцию пользователя

7.1. Оператор END

Оператор END заканчивает выполнение программы. При его выполнении закрываются все файлы, которые использовались в программе. Если оператор END отсутствует в программе, то программа заканчивается при выполнении последнего выполняемого оператора.

Формат оператора END

END

Параметров оператор не имеет. В зависимости от структуры программы оператор END в конце программы можно не указывать. В этом случае выполнение программы закончится, когда не будет операторов для выполнения.

7.2. Оператор GOTO-безусловный переход

Оператор GOTO обеспечивает безусловный переход в любое именованное место программы.

Формат оператора GOTO:

GOTO метка_выражения

Где

Метка_выражения — имя-переменной типа метки, с помощью которой помечено выражение, которому передается управление. Если управление передано на невыполняемый оператор(REM, DATA и т.п.), то выполнение программы будет продолжено с первого встретившегося выполняемого оператора.

Пример 7.2.1.

Составить программу вычисления выражения $y=2^x$ для значений $x<999$. Значение x вводится с клавиатуры.

```
CLS 'Очистка экрана монитора
vvod: INPUT "Введите значение X<999"; x 'ввод показателя степени
IF x >= 999 THEN GOTO метка 'проверка введенного числа и
```

```
'если ошибка, то переход на метка
PRINT "2 в степени"; x; "="; 2 ^ x 'печать полученного
результата
'нужно ли еще вычислять?
INPUT "Закончить работу (0-да,#0-нет)"; x%
IF x% <> 0 THEN GOTO vvod 'если не 0, то переход на vvod
END
'обработка ошибки
метка: PRINT "Неверно. Обратите внимание на условие"
GOTO vvod
```

7.3. Оператор IF-THEN-ELSE-условный оператор

Оператор IF-THEN-ELSE (если-то-иначе) обеспечивает принятие двоичного решения.

Формат оператора IF-THEN-ELSE:

```
IF условие1 THEN
    [блок_операторов_1]
[ELSEIF условие2 THEN
    [блок_операторов_2]] ...
[ELSE
    [блок_операторов_n]]
END IF
```

где

условие1 Любое выражение Бейсика, которое может быть оценено,
условие2 как истинное(не ноль) или ложное(ноль)

блок_операторов_1 один или несколько операторов в одной
блок_операторов_n или нескольких строках

Другой формат оператора IF-THEN-ELSE:

```
IF условие THEN операторы_1 [ELSE операторы_2]
```

где

условие Любое выражение Бейсика, которое может быть оценено, как истинное(не ноль) или ложное(ноль)

операторы Один или несколько операторов, разделенных двоеточием.

Если отсутствует часть ELSE и условие истинно, то выполняется последовательность операторы_1. После этого управление передается оператору, следующему за оператором IF-THEN-ELSE. Если условие ложно, то оператор IF-THEN-ELSE не выполняется.

Если в операторе IF-THEN-ELSE присутствует часть ELSE и условие истинно, то выполняется последовательность операторов_1. После этого управление передается оператору, следующему за оператором IF-THEN-ELSE. Если условие ложно, то выполняется последовательность операторов_2.

Пример 7.3.1.

```
CLS
a1: INPUT "1 или 2?", i%
IF i% = 1 OR i% = 2 THEN
PRINT "ok"
ELSE
PRINT "вне интервала"
GOTO a1
END IF
```

7.4. Оператор IF-GOTO — оператор условного перехода

Оператор IF-GOTO обеспечивает либо переход GOTO, либо выполнение последовательности операторов, следующих за оператором IF-GOTO.

Формат оператора IF-GOTO:

IF **условие** GOTO **метка_выражения**

где

Условие любое выражение Бейсика, которое может быть оценено, как истинное(не ноль) или ложное(ноль)

Метка_выражения имя переменной типа метки, которой помечено выражение, которому передается управление. Если управление передано на невыполняемый оператор(REM, DATA и т.п.), то выполнение программы будет продолжено с первого встретившегося выполняемого оператора.

Пример 7.4.1.

```
CLS 'очистка экрана монитора
IF a = 0 GOTO b 'условный переход на метку b
PRINT "не работает"
DO: LOOP WHILE INKEY$ = "" 'остановка до нажатия любой клавиши
b: PRINT "работает"
END
```

7.5. Оператор FOR-TO-NEXT-арифметический цикл.

Операторы FOR-TO (для — к) и NEXT(следующий) объединяют в один блок группу операторов, предназначенных для многократного выполнения.

Формат записи оператора:

```
FOR перем._цикла=нач._знач. TO кон._знач. [STEP знач_шага_цикла]
.....
тело цикла
.....
NEXT [перем._цикла]
```

где

Перем._цикла	является числовой переменной.
Нач._знач. и кон._знач.	могут быть числовой константой или арифметическим выражением.
Знач_шага_цикла	может быть как числовой константой, так и числовой переменной, положительной или отрицательной. Если операнд STEP опущен, то значение шага равно 1.

При обращении к оператору FOR-TO сначала вычисляется шаг, начальное и конечное значение. Если при положительном значении шага полученное значение переменной цикла меньше или равно конечному значению, то выполняется последовательность операторов, следующая за оператором FOR-TO (тело цикла). Если полученное значение переменной цикла больше конечного значения, то происходит выход из цикла и управление передается оператору, следующему за NEXT. При отрицательном значении шага значение переменной цикла уменьшается с каждым циклом. Цикл выполняется до тех пор, пока значение переменной цикла не станет меньше конечного значения. Если при входе в блок FOR-TO заданное условие не выполняется, то управление сразу передается оператору, следующему за оператором NEXT. Блоки FOR-TO могут быть вложенными друг в друга.

Пример 7.5.1.

Посчитать число счастливых билетов в катушке автобусных билетов с номерами от 000 000 до 999 999. Счастливым считается билет, у которого сумма первых трех цифр равна сумме последних трех цифр.

```
'Программа подсчета счастливых билетов
FOR i = 0 TO 9 'цикл на первую цифру
  FOR j = 0 TO 9 'цикл на вторую цифру
    FOR k = 0 TO 9 'цикл на третью цифру
      FOR l = 0 TO 9 'цикл на четвертую цифру
        FOR m = 0 TO 9 'цикл на пятую цифру
          FOR n = 0 TO 9 'цикл на шестую цифру
            'если совпали, то суммируем
            IF i + j + k = l + m + n THEN s = s + 1
          NEXT n 'конец шестого цикла
        NEXT m 'конец пятого цикла
      NEXT l 'конец четвертого цикла
    NEXT k 'конец третьего цикла
  NEXT j 'конец второго цикла
NEXT i 'конец первого цикла
CLS 'очистка экрана
PRINT "число счастливых билетов="; s 'печать числа счастливых билетов
END 'конец программы
```

По концу работы программа напечатает число 55 252.

Цикл называется арифметическим потому, что до начала цикла переменная цикла должна быть определена, то есть цикл выполняется определенное число раз.

Если в блоке не указан оператор FOR-TO или NEXT, то выдается сообщение:

```
NEXT без FOR
```

или

```
FOR без NEXT
```


Содержание

От автора	3
Предисловие редактора	4
Глава I. ЭВМ.	6
1. ЭВМ (PC, компьютер)	6
Глава II. Язык программирования	8
2.1. Алфавит	8
2.2. Лексические единицы	8
2.2.1. Операторы языка	8
2.2.2. Переменная	9
2.2.3. Разделители	9
2.2.4. Константы	9
2.3. Выражения	10
2.3.1. Арифметические операции	10
2.3.2. Операции сравнения	11
2.3.3. Операции сцепления	12
2.3.4. Операнды	12
2.3.5. Виды выражений	12
2.3.5.1. Условные обозначения	12
2.3.5.2. Правила записи выражений	13
Глава III. Данные	15
3.1. Оператор присваивания LET	15
3.2. Оператор обмена SWAP	15
3.3. Операторы вывода данных на экран монитора CLS и PRINT	16
3.4. Оператор DATA	16
3.5. Оператор READ	17
3.6. Оператор RESTORE	17
3.7. Оператор CONST	18
3.8. Типы данных	18
3.8.1. Оператор DIM	18
3.8.1.1. Массивы (индексированные переменные)	19
3.8.2. Оператор REDIM	21
3.8.3. Оператор ERASE	22
3.8.4. Оператор OPTION BASE	22
3.8.5. Тип данных, определенный пользователем	22
Оператор TYPE	23
Глава IV. Ввод информации	24
4.1. Ввод данных с клавиатуры во время выполнения программы	24
4.1.1. Оператор INPUT	24

4.1.2. Оператор LINE INPUT	25
4.1.3. Функция INPUT\$(n)	26
4.1.4. Функция INKEY\$	27
Глава V. Вывод данных на экран монитора	28
5.1. Оператор LOCATE	28
5.2. Работа с графической информацией	28
5.2.1. Оператор SCREEN	28
5.2.2. Оператор PSET	29
5.2.3. Оператор PRESET	29
5.2.4. Оператор LINE	30
5.2.5. Оператор CIRCLE	32
Глава VI. Операторы работы с экраном монитора	34
6.1. Преобразование системы координат экрана дисплея	34
6.1.1. Оператор VIEW	34
6.1.2. Оператор WINDOW	34
Глава VII. Управляющие операторы языка Бейсик	37
7.1. Оператор END	38
7.2. Оператор GOTO-безусловный переход	38
7.3. Оператор IF-THEN-ELSE-условный оператор	39
7.4. Оператор IF-GOTO — оператор условного перехода	40
7.5. Оператор FOR-TO-NEXT-арифметический цикл	40
7.6. Оператор WHILE-WEND-итерационный цикл1	42
7.7. Оператор DO-LOOP-безусловный цикл	43
7.8. Оператор DO-WHILE(UNTIL)-LOOP-итерационный цикл2	44
7.9. Оператор DO-LOOP-WHILE(UNTIL)-итерационный цикл3	45
7.10. Оператор SELECT-CASE-ENDSELECT — структура с условием	46
7.11. Оператор GOSUB-RETURN-выполнить подпрограмму	47
7.12. Оператор ON-GOTO — выбрать безусловный переход	48
7.13. Оператор ON-GOSUB-выбрать подпрограмму	48
7.14. Оператор ON-ERROR-выполнить подпрограмму обработки ошибок	49
7.15. Оператор RESUME-продолжить выполнение программы после подпрограммы обработки ошибки	51
7.16. Оператор STOP-временно остановить выполнение программы	51
7.17. Оператор DEF-определить функцию пользователя	51
Глава VIII. Математические функции	54
8.1. Функции ABS и SGN	55
8.2. Функции CDBL и CSNG	55
8.3. Функции CINT и CLGN	55
8.4. Функции CVDMBF, CVSMBF и MKSMBF\$, MKDMBF\$	56
8.5. Функции EXP и LOG	57
8.6. Функции INT и FIX	57

8.7. Функции RANDOMIZE и RND	57
8.8. Тригонометрические функции	60
8.9. Функция SQR	61

Глава IX. Работа с символьными данными 63

9.1. Преобразование кодов.	63
9.1.1. Функция CHR\$	64
9.1.2. Функция ASC	64
9.1.3. Функции LCASE\$, UCASE\$	64
9.2. Выделение части символьного выражения	65
9.2.1. Функции LEFT\$, RIGHT\$	65
9.2.2. Функция MID\$	65
9.3. Замена части символьного выражения	66
9.3.1. Оператор MID\$.	66
9.3.2. Функции LTRIM\$, RTRIM\$	66
9.4. Формирование строк одинаковых символов	67
9.4.1. Функция STRING\$	67
9.4.2. Функция SPACE\$	67
9.5. Обработка числовой информации в тексте	68
9.5.1. Функция VAL	68
9.5.2. Функция STR\$	68
9.6. Перевод чисел из одной системы счисления в другую	68
9.6.1. Функция OCT\$	68
9.6.2. Функция HEX\$	69
9.7. Количественные характеристики символьных строк	69
9.7.1. Функция LEN	69
9.7.2. Функция INSTR	69
9.8. Пересылка символов в текстовую строку. Функции LSET и RSET	70

Глава X. Файлы данных 72

10.1. Средства обработки файлов	73
10.2. Оператор OPEN	74
10.3. Оператор CLOSE	76
10.4. Оператор WRITE#	77
10.5. Оператор INPUT#	77
10.6. Оператор PRINT# USING	79
10.6.1. Шаблоны для символьных данных.	79
10.6.2. Вывод числовой информации	80
10.6.2.1. Шаблоны для вывода целых чисел имеют вид:	80
10.6.2.2. Шаблоны для вывода чисел с фиксированной точкой имеют вид:	81
10.6.2.3. Шаблоны для вывода чисел с плавающей точкой имеют вид:	82
10.7. Оператор PRINT#	84
10.8. Оператор LINE INPUT#	85
10.9. Функция INPUT\$#	86
10.10. Оператор FIELD	86

10.11. Функции MKI\$, MKS\$, MKD\$, MKL\$	86
10.12. Операторы LSET и RSET	87
10.13. Оператор PUT	88
10.14. Оператор GET	90
10.15. Функция LOC	94
10.16. Функция LOF	95
10.17. Функция EOF	95
10.18. Функции CVS, CVI, CVD, CVL	95
10.19. Функция SEEK. Оператор SEEK	96
10.20. Оператор FILEATTR	96
10.21. Оператор FREEFILE	97
10.22. Оператор LOCK, UNLOCK	97

Глава XI. Справочно 99

11.1. **СИНТАКСИС**	99
11.2. **НАБОР ОСНОВНЫХ СИМВОЛОВ**	99
11.3. **КЛЮЧЕВЫЕ СЛОВА БЕЙСИКА**	100
11.4. **БЫСТРЫЕ КЛАВИШИ**	101
11.5. **КЛАВИШИ РЕДАКТИРОВАНИЯ**	102
11.5.1. Клавиши перемещения курсора	102
11.5.2. Клавиши прокрутки текста	102
11.5.4. Клавиши вставки и замены	103
11.5.5. Клавиши удаления	103
11.6. **КЛАВИШИ ПРОСМОТРА И ПОИСКА**	104
11.7. **КЛАВИШИ ЗАПУСКА И ОТЛАДКИ**	104
11.8. **КОДЫ СИМВОЛОВ**	104
11.8.1. Стандартная таблица ASCII (коды символов 0 — 127)	104
11.8.2. Расширенная таблица ASCII (коды символов 128 — 255)	105
11.8.3. Атрибуты и значения цвета.	106
11.8.4. Режимы экрана	107
11.9. **СКАН-КОДЫ КЛАВИАТУРЫ**	109
11.10. **КОДЫ ОШИБОК ВЫПОЛНЕНИЯ**	110

Глава XII. Алфавитный указатель операторов, ключевых слов, функций языка Бейсик 112

ABS-функция	112
ABSOLUTE-ключевое слово	112
ACCESS-ключевое слово	112
AND — оператор	113
ANY-ключевое слово	113
APPEND-Ключевое слово	114
AS — ключевое слово	114
ASC — функция	114
ATN — функция	115
Base-ключевое слово	115
BEEP — оператор	115

BINARY — ключевое слово	115
BLOAD — оператор	115
BSAVE-оператор	116
CALL-оператор	116
CALL ABSOLUTE-оператор	116
CASE-ключевое слово	116
CDBL-функция.	117
CHAIN-оператор	117
CHDIR-оператор	117
CHR\$ — функция	118
CINT — функция	118
CIRCLE-оператор	118
CLEAR-оператор	119
CLNG — функция.	119
CLOSE — оператор	119
CLS — оператор	120
COLOR — оператор	120
COM — оператор	121
COMMON — оператор	122
CONST — оператор	122
COS — функция	123
CSNG — функция	123
CSRLIN — функция	123
CVD — функция	123
CVDMBF — функция	124
CVI — функция.	125
CVL — функция	125
CVS — функция	125
CVSMBF — функция.	125
DATA — оператор	125
Функция DATE\$, Оператор DATE\$	126
DECLARE — оператор	127
DEF FN-оператор.	127
DEF SEG-оператор	128
DEFINT-оператор	128
DEFLNG-оператор	129
DEFSNG оператор	129
DEFDBL-оператор	129
DEFSTR-оператор	129
DIM-оператор	129
DO LOOP-оператор	130
DOUBLE-ключевое слово	131
DRAW-оператор	131
\$DYNAMIC-метакоманда.	132
ELSE-ключевое слово	133
ELSEIF-ключевое слово	133
END-оператор	133

ENVIRON\$-функция, ENVIRON-оператор	134
EOF-оператор	134
EQV-ключевое слово	135
ERASE-оператор	135
ERDEV-функция, ERDEV\$-функция	135
ERR-Функция, ERL-функция.	135
ERROR-оператор.	136
EXIT-оператор.	137
EXP-оператор	137
FIELD-оператор	137
FILEATTR-оператор	138
FILES-оператор	138
FIX-оператор	138
FOR-оператор	139
FRE оператор	139
FREEFILE-оператор	140
FUNCTION-оператор	140
GET-оператор(файл).	141
GET-оператор(графический)	142
GOSUB-оператор	143
GOTO-оператор	144
HEX-функция	144
IF-оператор	145
IMP-ключевое слово	145
INKEY\$-оператор	146
INP-оператор	146
INPUT-оператор	146
INPUT\$-функция	147
INSTR-функция	148
INTEGER-ключевое слово	148
IOCTL-оператор	148
IOCTL\$-функция	149
IS-ключевое слово.	149
KEY-оператор(отображение).	150
KEY-оператор(отслеживание)	150
Объявление определенных пользователем клавиш	152
KILL-оператор.	152
LBOUND-функция	153
LCASE\$-функция	153
LEFT\$-функция	153
LEN-функция	154
LINE-оператор.	154
LINE INPUT-оператор	155
LIST-ключевое слово	155
LOC-функция	155
LOCATE-оператор	155
LOCK-оператор	156

LONG-ключевое слово	156
LOOP-ключевое слово	156
LPOS-функция	156
LPRINT-оператор	157
LPRINT USING-оператор.	157
LSET-оператор.	158
LTRIM\$-функция.	159
MID\$-Функция, MID\$-Оператор	159
MKD\$-функция	160
MKDIR-оператор.	160
MKDMBF\$-оператор	160
MKI\$-функция	160
MKL\$-функция	160
MKSS\$-функция.	160
MKSMBF\$-функция	161
MOD-оператор.	161
NAME-оператор	161
NEXT-ключевое слово	161
NOT оператор	162
OCT\$-оператор	162
OFF-ключевое слово	162
ON COM-оператор	162
ON ERROR-оператор	162
ON-ключевое слово	163
ON KEY-оператор	163
ON PEN-оператор.	163
ON PLAY-оператор	164
ON STRING-оператор	164
ON TIMER-оператор	165
ON GOSUB-оператор	166
ON GOTO-оператор	167
OPEN-оператор	167
OPEN COM-оператор	168
OPTION BASE-оператор	170
OR-оператор	170
OUT-оператор	170
OUTPUT-ключевое слово	170
PAINT-оператор	170
PALETTE-оператор	171
PCOPY-оператор	172
PEEK-функция.	172
PEN-функция	173
PEN-оператор	173
PLAY-функция.	174
PLAY-оператор	174
PLAY-отслеживание событий	175
PMAP-функция	175

POINT-функция	176
POKE-функция.	176
POS-функция	177
PRESET-оператор	177
PRINT-оператор	177
PRINT USING-оператор	178
PSET-оператор.	178
PUT(В/ВВ файлов)-оператор	178
PUT(графика)-оператор	178
RANDOM-ключевое слово	178
RANDOMIZE-оператор	178
READ-оператор	179
REDIM-оператор	179
REM-оператор	179
RESET-оператор	180
RESTORE-оператор	180
RESUME-оператор	180
RETURN-оператор	180
RIGHT\$-оператор	181
RMDIR-оператор.	181
RND-оператор	181
RSET-оператор	181
RTRIM\$-функция.	181
RUN-оператор	181
SCREEN-функция	182
SCREEN-оператор	182
SEEK-функция	183
SEEK-оператор	183
SELECT CASE-оператор	184
SGN-оператор	185
SHARED-оператор	185
SHELL-оператор	185
SIN-функция	185
SINGLE-ключевое слово	186
SLEEP-оператор	186
SOUND-оператор.	186
SPACE\$-функция	186
SPC-функция	187
SQR-функция	187
STATIC-функция	187
\$STATIC-метафункция	187
STEP-ключевое слово	188
STICK-оператор	188
STOP-оператор.	188
STR\$-функция	189
STRIG-функция	189
STRIG-операторы	190

STRING-ключевое слово	190
STRING\$-функция	191
SUB-оператор	191
SWAP-оператор	191
SYSTEM-оператор	192
TAB-функция	192
TAN-функция	192
THEN-ключевое слово	192
TIMES-оператор TIMES-функция	192
TIMER-функция	193
TIMER-оператор	193
TO-ключевое слово	194
TRON, TROF-операторы	194
TYPE-оператор	194
UBOUND-функция	195
UCASE\$-функция	195
UNLOCK-оператор	195
UNTIL-ключевое слово	196
USING-ключевое слово	196
VAL-функция	196
VARPTR-функция	196
VARPTR\$-функция	197
VARSEG-функция	197
VIEW-оператор	197
VIEW PRINT-оператор	198
WAIT-оператор	198
WEND-ключевое слово	199
WHILE-WEND-оператор	199
WIDTH-оператор	199
WINDOW-оператор	200
WRITE-оператор	201
XOR XNACK-булева операция	201

Глава XIII. Примеры программ 203

13.1. Программа сортировки чисел	203
13.2. Программа сортировки символьных констант	203
13.3. Программа построения графиков	204
13.4. Программа построения окружностей	205
13.5. Программа построения модели солнечной системы. Планеты земной группы	205
13.6 Программа построения модели солнечной системы. Внутренние планеты солнечной системы	207
13.7. Программа построения модели солнечной системы	208
13.8. Программа поиска корней многочлена	210
13.9. Программа построения трехмерной фигуры	211
13.10. Программа “Игра в отгадывание слов” (самообучающаяся программа)	212