

**Basic**

# АЛГОРИТМЫ.

## Свойства алгоритмов:

1. Дискретность (алгоритм должен состоять из конкретных действий, следующих в определенном порядке);
2. Детерминированность (любое действие должно быть строго и недвусмысленно определено в каждом случае);
3. Конечность (каждое действие и алгоритм в целом должны иметь возможность завершения);
4. Массовость (один и тот же алгоритм можно использовать с разными исходными данными);
5. Результативность (отсутствие ошибок, алгоритм должен приводить к правильному результату для всех допустимых входных значениях).

## **Виды алгоритмов:**

1. **Линейный алгоритм** (описание действий, которые выполняются однократно в заданном порядке);
2. **Циклический алгоритм** (описание действий, которые должны повторяться указанное число раз или пока не выполнено задание);
3. **Разветвляющийся алгоритм** (алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий);
4. **Вспомогательный алгоритм** (алгоритм, который можно использовать в других алгоритмах, указав только его имя).

## **Стадии создания алгоритма:**

1. Алгоритм должен быть представлен в форме, понятной человеку, который его разрабатывает.
2. Алгоритм должен быть представлен в форме, понятной тому объекту (в том числе и человеку), который будет выполнять описанные в алгоритме действия.

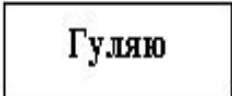

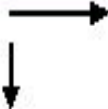
**Исполнитель** - объект, который выполняет алгоритм.

Идеальными исполнителями являются *машины, роботы, компьютеры...*

**Компьютер** – автоматический исполнитель алгоритмов.

Алгоритм, записанный на «понятном» компьютеру языке программирования, называется **программой**.

Для более наглядного представления алгоритма широко используется графическая форма - блок-схема, которая составляется из стандартных графических объектов.

Вид стандартного графического объекта	Назначение
	Начало алгоритма
	Конец алгоритма
	Выполняемое действие записывается внутри прямоугольника
	Условие выполнения действий записывается внутри ромба
	Счетчик кол-во повторов
	Последовательность выполнения действий.

# Арифметические операции на языке Basic.

Операция	Обозначение	Пример	Результат
Сложение	+	2+5	7
Вычитание	-	10-8	2
Умножение	*	3*4	12
Деление	/	15/3 15/4	5 3.75
Целочисленное деление	\	15\4	3
Возведение в степень	^	2^3	8
Остаток от деления	MOD	13 MOD 5	3

## Математические функции на языке Basic.

Корень	SQR(X)
Модуль числа	ABS(X)
Синус	SIN(X)
Косинус	COS(X)
Тангенс	TAN(X)
Целая часть числа	INT(X)
Натуральный логарифм	LOG(X)

## Некоторые операторы языка Basic.

**REM** – оператор комментария. Все что следует после этого оператора до конца строки игнорируется компилятором и предназначено исключительно для человека. Т.е. здесь можно писать что угодно. Удобно использовать комментарий в начале программы для указания её названия и назначения.

**пример:**

REM Это комментарий

можно и так:

' Это тоже комментарий

**CLS** - очистить экран. Вся информация, которая была на экране стирается.

**PRINT** (вывод, печать) – оператор вывода.

**пример:**

```
PRINT "Привет! Меня зовут Саша."
```

*На экран будет выведено сообщение: Привет! Меня зовут Саша.*



**INPUT** (ввод) – оператор ввода. Используется для передачи в программу каких-либо значений.

**пример:**

**INPUT a**

*На экране появится приглашение ввести данные*

*(появится знак "?") и компьютер будет ждать их ввода.*

*Для ввода необходимо ввести данные с клавиатуры и нажать ввод (enter).*

**INPUT "Введите число a: ", a**

*Компьютер выведет на экран: 'Введите число a:' и будет ждать ввода данных.*

**DIM** – оператор описания типа переменной.

Под **переменной** в языках программирования понимают программный объект (число, слово, часть слова, несколько слов, символы), имеющий имя и значение, которое может быть получено и изменено программой.

**пример:**

```
DIM a, b, chislo1 AS INTEGER
```

*Integer – целые числа от -32768 до 32768*

Если в программе используются переменные не описанные с помощью оператора DIM, то компьютер будет рассматривать их как универсальные переменные. Это может привести к неэффективному использованию оперативной памяти. К тому же, такие программы не всегда легки для восприятия - плохо читаемы.

Для задания значения переменной служит оператор присваивания. Он записывается так:

**LET** переменная = значение (или просто: переменная = значение)

**пример:**

LET a = 3

chislo1 = 15

**END** – оператор конца программы.

# ЛИНЕЙНАЯ СТРУКТУРА ПРОГРАММЫ.

Программа имеет линейную структуру, если все операторы (команды) выполняются последовательно друг за другом.



**Пример:** программа, выводящая на экран сообщение:  
Привет! Меня зовут Саша!

```
REM Первая программа
```

```
PRINT "Привет! Меня зовут Саша!"
```

```
END
```

**Пример:** программа,  
складывающая два числа

REM Сумма двух чисел

a = 5

b = 6

c = a + b

PRINT "Результат: ", c

END

**Пример:** Вычислите площадь прямоугольника по его сторонам.

```
REM Площадь прямоугольника  
INPUT "Введите сторону a", a  
INPUT "Введите сторону b", b  
s = a * b  
PRINT "Площадь равна: ", s  
END
```

**Пример:** Вычислите длину окружности и площадь круга по данному радиусу.

REM Вычисление длины окружности и площади  
круга

INPUT "Введите радиус ", r

PI = 3.14

l = 2 \* PI \* r

s = PI \* r \* r

PRINT "Длина окружности равна: ", l

PRINT "Площадь равна: ", s

END



**Пример:** Вычислить выражение

$$c = \frac{\sqrt{2ab}}{a+b}$$

REM Вычисление выражения

INPUT "Введите a", a

INPUT "Введите b", b

c = SQR(2\*a\*b)/(a+b)

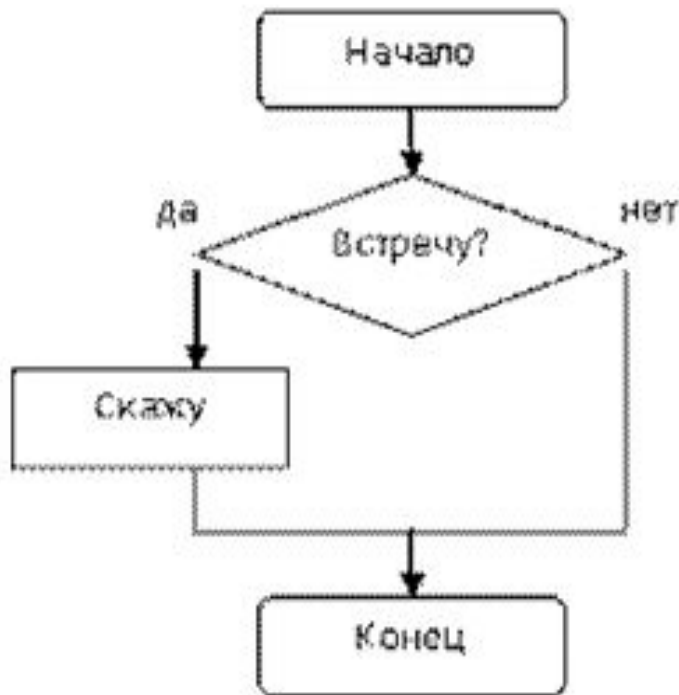
PRINT "Площадь равна: ", c

END

# ВЕТВЛЕНИЕ В АЛГОРИТМАХ И ПРОГРАММАХ.

**Разветвляющийся алгоритм** – это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Во многих случаях требуется, чтобы при одних условиях выполнялась одна последовательность действий, а при других - другая.



Вся программа состоит из команд (операторов).

Команды бывают простые и составные (команды, внутри которых встречаются другие команды).

Составные команды часто называют управляющими конструкциями.



## Условный оператор на языке Basic.

*Простая форма оператора выглядит следующим образом:*

**IF <УСЛОВИЕ> THEN <ОПЕРАТОР>**

или

**IF <УСЛОВИЕ>**

**<ОПЕРАТОР 1>**

**<ОПЕРАТОР 2>**

...

**<ОПЕРАТОР N>**

**END IF**

Если условие справедливо, то программа выполняет тот оператор, который стоит после ключевого слова **THEN** (или серию операторов от ключевого слова **THEN** до **END IF**), и дальше руководствуется обычным порядком действий. Если условие не справедливо, то оператор, стоящий после **THEN** (или серия операторов от **THEN** до **END IF**) не выполняется, и программа сразу переходит к обычному порядку действий.

Конструкция **IF...THEN** позволяет в зависимости от справедливости условия либо выполнить оператор, либо пропустить этот оператор.

Конструкция **IF...THEN...END IF** позволяет в зависимости от справедливости условия либо выполнить группу операторов, либо пропустить эту группу операторов.

Условия - еще один тип логических выражений. В них используются следующие *операторы сравнения*:

=	равно
<>	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Справа и слева от знака сравнения должны стоять величины, относящиеся к одному типу. В результате сравнения получается логическая величина, имеющее значение ИСТИНА (TRUE) или ЛОЖЬ (FALSE).

Пример:

$5 < 7$  - ИСТИНА;

$8 = 12$  - ЛОЖЬ (проверяем равно ли 8 12, именно проверяем, а не утверждаем, что  $8 = 12$ );

Чтобы вычисления могли разветвляться по нескольким направлениям, служит конструкция **IF...THEN...ELSE...END IF**.

```
IF <УСЛОВИЕ> THEN  
<ОПЕРАТОРЫ 1>  
ELSE  
<ОПЕРАТОРЫ 2>  
END IF
```

Если условие справедливо (ИСТИНА), то выполняются <операторы 1> (стоящие между **THEN** и **ELSE**), а <операторы 2> (стоящие между **ELSE** и **END IF**) будут пропущены.

Если условие не справедливо (ЛОЖЬ), то <операторы 1> игнорируются и выполняются <операторы 2>.

*IF - если, THEN - тогда, ELSE - иначе.*

**Если** в комнате темно, **тогда** надо включить свет.

**Если** пойдет дождь, **тогда** надо взять зонтик, **иначе**, зонтик не брать.

**Пример:** Проверить, равно ли введенное число некоторому значению, и в случае равенства выдать на экран сообщение о равенстве чисел.

```
REM сравнить число со каким-то значением  
INPUT "Введите a", a  
IF a=7 THEN PRINT "Числа равны"  
END
```

После запуска программы проверяется равно ли введенное значение семи или нет. Если равно, то на экран выводится сообщение 'Числа равны'.

**Пример:** Решение квадратного уравнения.

Решение квадратного уравнения зависит от значения дискриминанта.

```
REM Решение квадратного уравнения
INPUT "Введите коэффициент a: ", a
INPUT "Введите коэффициент b: ", b
INPUT "Введите коэффициент c: ", c
d=b*b-4*a*c
IF d<0 THEN
PRINT "Корней нет"
ELSE
IF d=0 THEN
x=-b/(2*a)
PRINT "корень уравнения: ", x
ELSE
x1=(-b-SQR(d))/(2*a)
x2=(-b+SQR(d))/(2*a)
PRINT "корни уравнения: ", x1, x2
END IF
END IF
END
```



## Структура "Выбор".

Структура **IF...** позволяет выбрать между двумя вариантами. Если требуется осуществить выбор между большим числом вариантов, то это можно организовать используя лишь структуру **IF...** Но можно (что чаще проще) и с помощью структуры "Выбор". Эта структура имеет вид:

```
SELECT CASE <Выражение>  
  CASE <условие 1>  
    <серия 1>  
  CASE <условие 2>  
    <серия 2>  
    ...  
  CASE ELSE  
    <серия иначе>  
END SELECT
```

Выражение, заданное после ключевых слов **SELECT CASE**, сравнивается с определенными значениями - условиями и если они истинны, то выполняется соответствующая серия команд. Если не одно условие не истинно, то выполняется серия команд между **CASE ELSE** и **END SELECT**.

**Пример:** Выдать словесное значение числа

REM Преобразование чисел в слова

```
INPUT "Введите число", a
```

```
SELECT CASE a
```

```
  CASE 1
```

```
    PRINT "один"
```

```
  CASE 2
```

```
    PRINT "два"
```

```
  CASE 3
```

```
    PRINT "три"
```

```
  ...
```

```
  CASE 10
```

```
    PRINT "десять"
```

```
  CASE ELSE
```

```
    PRINT "это число не могу перевести"
```

```
  END SELECT
```

```
END
```

В данном примере введенное число сравнивается с числами от 1 до 10 и если наше число равно одному из этих чисел, то на экран выводится словесное значение числа. Если это не так на экран выводится сообщение: "это число не могу перевести".

# ЦИКЛЫ В АЛГОРИТМАХ И ПРОГРАММАХ.

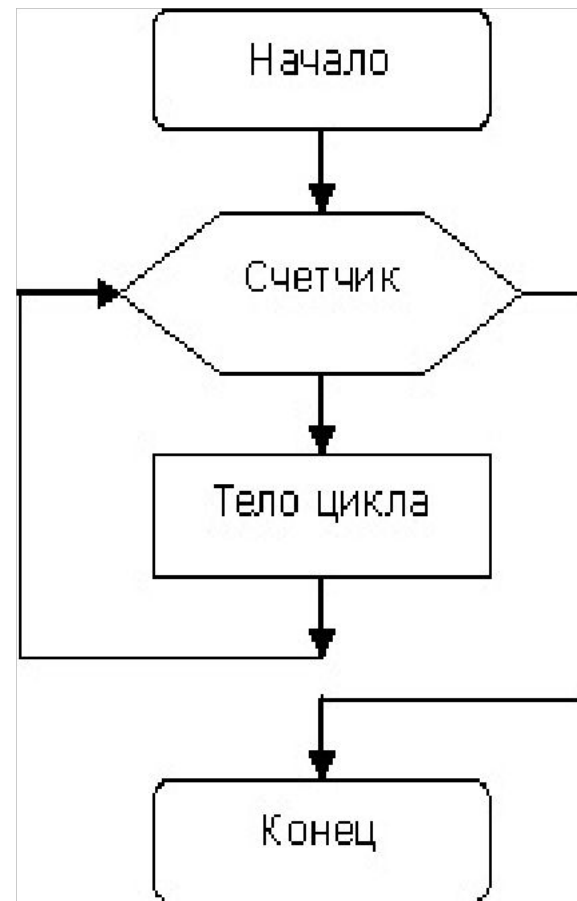
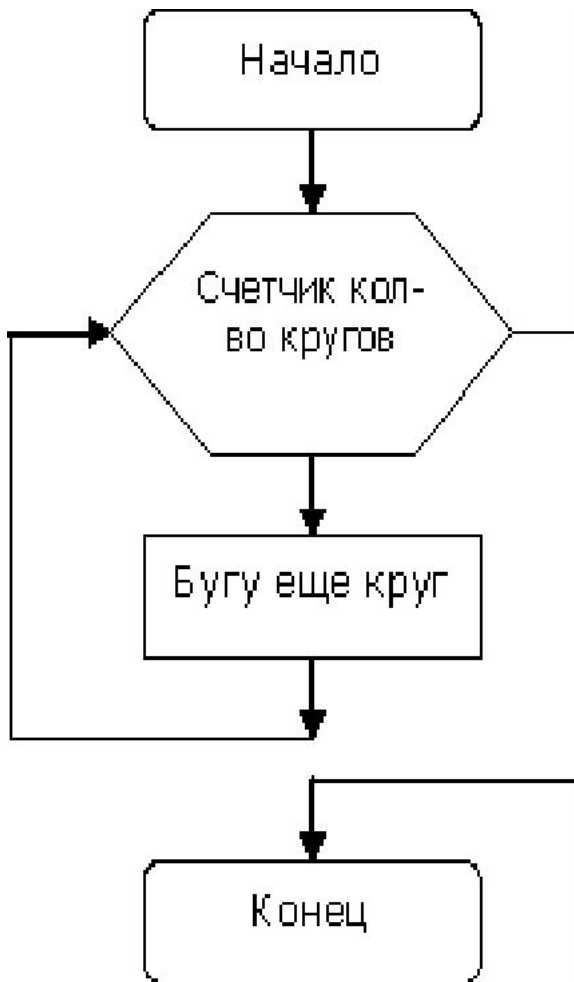
**Циклический алгоритм** - описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Перечень повторяющихся действий называют **телом цикла**.

- циклы со счетчиком.
- циклы с предусловием.
- циклы с постусловием.

## Цикл со счетчиком.

**Пример:** на уроке физкультуры вы должны пробежать некоторое количество кругов вокруг стадиона.



На языке Basic они записываются следующим образом:

```
FOR Счетчик=НачЗнач TO КонЗнач [STEP шаг]  
    тело цикла  
NEXT [Счетчик]
```

Параметры указанные в квадратных скобках являются не обязательными (их можно не записывать). По умолчанию шаг цикла равен одному, т.е. каждый раз после прохождения тела цикла счетчик увеличивается на единицу.

**Пример:** вычислить факториал числа  $a$  (записывается так:  $a!$ ).  
Факториал - это произведение чисел от 1 до  $a$ . Например,  $5!$   
(факториал пяти) - это  $5! = 1 * 2 * 3 * 4 * 5$

REM Вычислить факториал числа

a=5

f=1

FOR I=1 TO a

f=f\*I

NEXT

PRINT f

END

Вы, конечно, заметили, что до начала цикла мы присвоили переменной  $f$  значение равное единице. Иначе бы мы получили в результате ноль.

**Пример:** Вычислить сумму двухзначных натуральных чисел.

REM Вычислить сумму двухзначных натуральных чисел

FOR I=10 TO 99

s=s+I

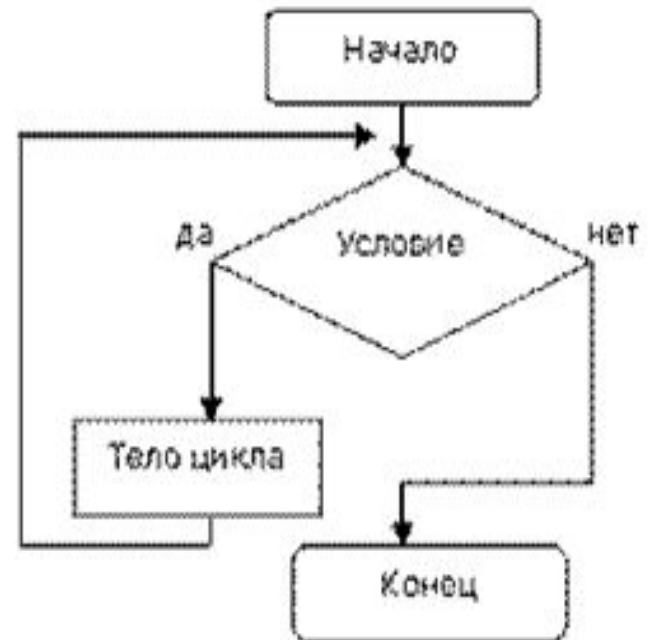
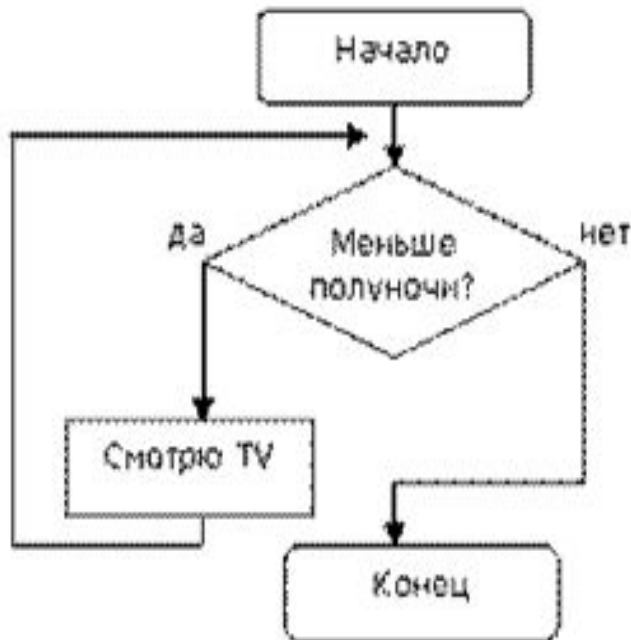
NEXT

PRINT "Результат = ",s

END

## Цикл с предусловием.

**Пример:** В субботу вечером вы смотрите телевизор. Время от времени поглядываете на часы и если время меньше полуночи, то продолжаете смотреть телевизор, если это не так, то вы прекращаете просмотр телепередач.





На языке Basic они записываются следующим образом:

**DO WHILE** условие

Тело цикла

**LOOP**

В этом цикле проверяется условие и если оно выполняется (ИСТИНА), то выполняется тело цикла до ключевого слова **LOOP**, затем условие проверяется снова ... и так до тех пор пока условие истинно.

**DO UNTIL** условие

Тело цикла

**LOOP**

Этот цикл отличается от предыдущего только тем, что он выполняется до тех пор пока условие не истинно (т.е. совсем наоборот).

**Пример:** Вывести все натуральные числа меньше данного.

REM Вывод всех чисел меньше данного

a=0

chislo=10

DO WHILE a<chislo

PRINT a

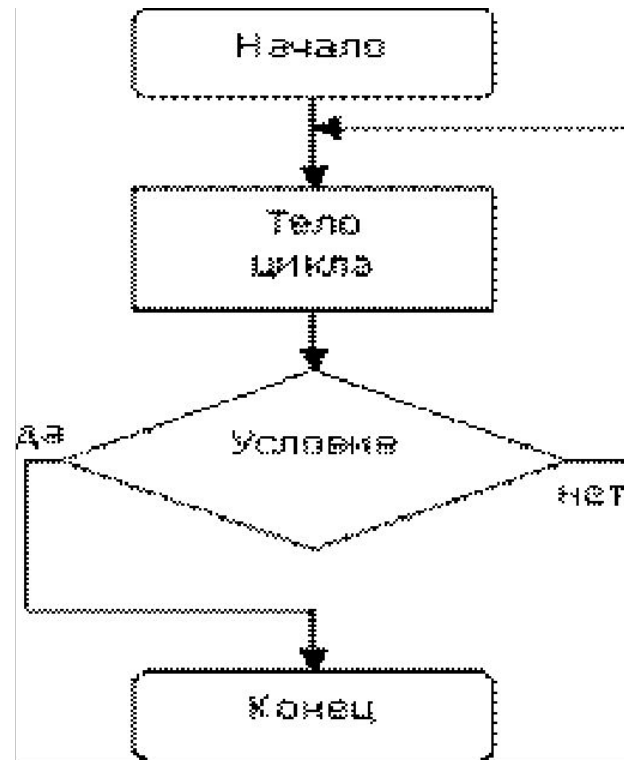
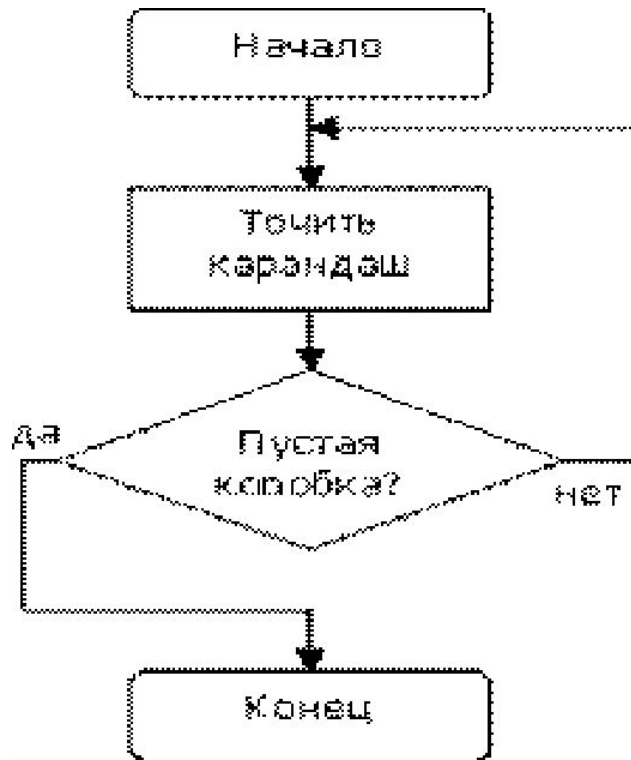
a=a+1

LOOP

END

## Цикл с постусловием.

Вам надо поточить все карандаши в коробке. Вы точите один карандаш и откладываете его в сторону. Затем проверяете, остались ли карандаши в коробке. Если условие ложно, то снова выполняется действие 'заточить карандаш'. Как только условие становится истинным, то цикл прекращается.



На языке Basic они записываются следующим образом:

**DO**

**Тело цикла**

**LOOP WHILE условие**

**DO**

**Тело цикла**

**LOOP UNTIL условие**

Циклы такого рода отличаются тем, что хоть один раз, но тело цикла будет выполнено вне зависимости от условия. Условие проверяется после первого выполнения тела цикла.

**Пример:** Вычислите сумму цифр в числе.

REM Сумма цифр числа

DIM a, chislo, s AS INTEGER

INPUT "Введите число: ", chislo

a=chislo

DO

s=s+a MOD 10

a=a/10

a=INT(a)

LOOP UNTIL a=0

PRINT "Сумма цифр числа ",chislo ," равна: ", s

END

# МАССИВЫ. ОДНОМЕРНЫЕ МАССИВЫ.

**Массив** - это набор переменных, имеющих одинаковое имя (идентификатор), но различающихся порядковыми номерами (индексами).

Массивы применяют для группировки переменных, имеющих много общих свойств.

**Пример:** Если в классе 30 учеников, то имя каждого ученика можно было бы сохранить в отдельной строковой переменной: name1, name2, ... Но вводить 30 новых переменных крайне неудобно. Можно сделать проще: объявить один массив name(), имеющий 30 элементов. В скобках проставляется индекс когда надо обратиться к какому-то конкретному элементу.

Отсчет элементов массива во многих языках начинается с нуля. Поэтому имя первого (по классному журналу) ученика будет храниться в переменной name(0), второго - в переменной name(1), а последнего (тридцатого) - в переменной name(29).

Для того чтобы использовать массив его надо сначала объявить в программе. Для этого используют оператор **DIM**. По умолчанию (если нет оператора **DIM** в программе) считается заданным массив из 10 элементов.

**DIM a(100) AS INTEGER**

Это массив из ста элементов, каждый из которых может быть целым числом.

**DIM name(30) AS STRING**

**DIM mas(20)**

Это массив из 20 элементов, тип переменных явно не указан.

**Пример:** Вывести количество отрицательных элементов массива.

**REM Вывести количество отрицательных элементов**

**INPUT "Введите число элементов массива", n**

**DIM mas(n) AS INTEGER**

**FOR I=0 TO n-1**

**INPUT "Введите элемент массива", mas(I)**

**NEXT**

**CLS**

**PRINT "Вывод массива"**

**FOR I=0 TO n-1**

**PRINT mas(I);**

**NEXT**

**FOR I=0 TO n-1**

**IF mas(I)<0 THEN k=k+1**



**Пример:** Составить программу сортировки массива по возрастанию.

**REM сортировка массива**

**INPUT "Введите число элементов массива", n**

**DIM mas(n) AS INTEGER**

**FOR I=0 TO n-1**

**mas(I)=1+INT(RND\*10)**

**NEXT**

**CLS**

**PRINT "Вывод массива"**

**FOR I=0 TO n-1**

**PRINT mas(I);**

**NEXT**

**REM сортировка массива**

**FOR I=0 TO n-2**

**FOR J=I+1 TO n-1**

**IF mas(I)>mas(J) THEN**

**REM если нашли меньший элемент, то обменяем их местами**

**a=mas(I)**

Для ввода данных удобно использовать операторы **DATA** и **READ**.

**DATA** указывает значения для чтения последующими операторами **READ**. **READ** считывает эти значения и присваивает их переменным. **RESTORE** позволяет **READ** заново считать значения в указанном операторе **DATA**.

**DATA** константы  
**READ** переменные

**Пример:** Ввод массива с использованием оператора DATA.

**REM Ввод данных из DATA**

**DIM mas(5) AS INTEGER**

**DATA 2, -4, 1, 5, 9**

**REM ввод массива**

**FOR I=0 TO 4**

**READ mas(I);**

**NEXT**

**REM вывод массива**

**FOR I=0 TO 4**

**PRINT mas(I);**

**NEXT**

**END**

# МАССИВЫ. ДВУМЕРНЫЕ МАССИВЫ.

Двумерные массивы - таблицы, в ячейках которых хранятся значения элементов массива, а индексы элементов массива являются номерами строк и столбцов.

Объявляются двумерные массивы так же, как переменные и одномерные массивы.

**Например**, целочисленный числовой массив, содержащий 3 строк и 4 столбца объявляется следующим образом:

```
DIM tabl(3 ,4) AS INTEGER
```

С помощью двумерного массива 9x9 и двух вложенных циклов можно легко составить программу, реализующую таблицу умножения. Сомножителями будут значения индексов строк и столбцов, а их произведения будут значениями элементов массива.

**DIM tablum(1 TO 9 ,1 TO 9) AS INTEGER**

**tablum**

1 2 3 4 5 6 7 8 9

1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

**REM Таблица умножения**

**DIM tabum(1 TO 9, 1 TO 9) AS INTEGER**

**REM Заполнение массива - создание таблицы умножения**

**FOR I=1 TO 9**

**FOR J=1 TO 9**

**tabum(I, J)=I\*J**

**NEXT J**

**NEXT I**

**REM Вывод массива на экран в виде таблицы**

**FOR I=1 TO 9**

**FOR J=1 TO 9**

**PRINT tabum(I,J);**

**NEXT J**

**PRINT**

**NEXT I**

**END**

**Пример:** В таблице 3x4 вычислить количество отрицательных элементов, сумму четных элементов, произведение элементов второй строки.

**REM вычислить количество...**

**DIM tabl(1 TO 3, 1 TO 4) AS INTEGER**

**REM Заполнение массива**

**FOR I=1 TO 3**

**FOR J=1 TO 4**

**INPUT "Введите элемент массива:", tabl(I, J)**

**NEXT J**

**NEXT I**

**REM Вывод массива на экран в виде таблицы**

**CLS**

**FOR I=1 TO 3**

**FOR J=1 TO 4**

**PRINT tabl(I,J);**

**NEXT J**

**PRINT**

**NEXT I**

**REM требуемые вычисления**

**k=0**

**s=0**

**p=1**

# **СИМВОЛЬНЫЕ И СТРОЧНЫЕ ПЕРЕМЕННЫЕ.**

Это переменные, значениями которых являются либо алфавитно-цифровые символы, либо несколько таких символов.

**Строки** - последовательность алфавитно-цифровых символов.

Для того, чтобы использовать такие переменные в программе необходимо их соответствующим образом объявить. Для этого используется **оператор DIM**.

```
DIM s AS STRING
```

```
s="Строка123"
```

Или добавлять справа от переменной **символ \$**.

```
s$="Тоже строка 987"
```



Строчные переменные можно склеивать и сравнивать друг с другом. Для склеивания строк (конкатенации) используют знак плюс (+).

Для сравнения строк используют операции: >, <, =, >=, <=, <>.

### Функции для работы со строками:

LEN(s\$)	Вычисляет длину строки (количество символов).
MID\$(s\$,n,k)	Выделяет из строки s\$ k символов начиная с n-го символа.
VAL(s\$)	Преобразует числовую часть начала строки в число.
STR\$(x)	Преобразует число в символьную форму.
ASC(s\$)	Вычисляет десятичный код символа.
CHR\$(x)	Преобразует код в символ.
INKEY\$	Функция опроса клавиш, нажатых на клавиатуре.

Пример: составить программу подсчитывающую, количество букв "а" в предложении.

```
REM кол-во букв "а"  
INPUT "Введите предложение", s$  
FOR I=1 TO LEN(s$)  
IF MID$(s$,I,1)="а" THEN k=k+1  
NEXT  
PRINT "Кол-во букв а =", k  
END
```

Пример: Заменить все буквы "а" в предложении на буквы "о".

```
REM замена букв  
ss$=""  
INPUT "Введите предложение", s$  
FOR I=1 TO LEN(s$)  
IF MID$(s$,I,1)="а" THEN  
ss$=ss$+"о"  
ELSE  
ss$=ss$+MID$(s$,I,1)  
END IF  
NEXT  
PRINT "Исправленная строка: ", ss$  
END
```

**Пример:** Получить предложение в обратном порядке следования символов.

**REM обратный порядок букв**

```
ss$=""
```

```
INPUT "Введите предложение", s$
```

```
FOR I=LEN(s$) TO 1 STEP -1
```

```
ss$=ss$+MID$(s$,I,1)
```

```
NEXT
```

```
PRINT "Исправленная строка: ", ss$
```

```
END
```

# ПОДПРОГРАММЫ. ПРОЦЕДУРЫ.

**Подпрограммой** называется группа операторов, к которой обращаются из основной программы несколько раз.

Принято различать два вида подпрограмм - процедуры и функции.

*Процедуры состоят из трех частей:* заголовка, тела процедуры, завершения процедуры.

**SUB имя (список параметров)**

**тело процедуры - список операторов**

**END SUB**

## Пример:

```
SUB hello (s$)  
  PRINT "Привет, ", s$, "! Как твои дела?"  
END SUB  
REM приветствие  
name1$="Саша"  
name2$="Вася"  
REM процедуру можно вызвать так  
CALL hello(name1$)  
REM а можно вызвать так  
hello(name2$)  
REM или даже так  
hello("Марина")  
END
```

*В результате выполнения программы на экране будет выведено:*

*Привет, Саша! Как твои дела?*

*Привет, Вася! Как твои дела?*

*Привет, Марина! Как твои дела?*

# ПОДПРОГРАММЫ. ФУНКЦИИ.

Функции выполняют определенные действия и возвращают вызывающей программе какое-то значение.

**FUNCTION имя (список параметров)**

**тело функции - список операторов**

**END FUNCTION**

**Пример: Функция возвращающая куб числа**

```
FUNCTION kub (x)  
kub=x*x*x  
END FUNCTION
```

```
REM Вывод кубов натуральных чисел от 1 до 10  
CLS  
FOR I=1 TO 10  
PRINT kub(I)  
NEXT  
END
```

В этой программе в цикле происходит обращение к функции **kub**, которая вычисляет куб числа.

# ГРАФИЧЕСКИЙ РЕЖИМ РАБОТЫ.

Программы могут выводит данные на экран в текстовом и графическом режиме работы. Для перехода в графический режим работы служит оператор:

**SCREEN <mode>**

<mode> - целочисленная константа, указывающая режим работы для данного экрана и адаптера.

**Пример:**

SCREEN 1

SCREEN 2

...

SCREEN 11

...



Для рисования можно использовать следующие операторы:

CLS	Очистка экрана
PSET(X,Y),C	Изобразить точку. X,Y - координаты точки, C - цвет.
PSET STEP(X,Y),C	Изобразить точку. X,Y - смещение от данной точки, C - цвет.
LINE(X1,Y1)-(X2,Y2),C	Прямая линия. X1,Y1 и X2,Y2- координаты концов линии, C - цвет.
LINE -(X2,Y2),C	Прямая линия. От текущего положения курсора до X2,Y2- координаты конца линии, C - цвет.
LINE(X1,Y1)-(X2,Y2),C,B	Прямоугольник. X1,Y1 и X2,Y2- координаты концов диагонали, C - цвет.
LINE(X1,Y1)-(X2,Y2),C,BR	Закрашенный прямоугольник. X1,Y1 и X2,Y2- координаты концов диагонали, C - цвет.
CIRCLE(X,Y),R,C	Окружность. X,Y - координаты центра, C -цвет.
CIRCLE STEP(X,Y),R,C	Окружность. X,Y - смещение от данной точки, C - цвет.
CIRCLE(X,Y),R,C,A1,A2	Дуга окружности. X,Y - координаты центра, C - цвет, A1,A2 - угловые меры начальной и конечной точки дуги.
CIRCLE(X,Y),R,C,,,K CIRCLE(X,Y),R,C,A1,A2,K	Эллипс. K - коэффициент сжатия.
PAINT(X,Y),C1,C2	Закрасить область. C1 - цвет закрашки, C2 - цвет границы.
LOCATE T1,T2	Установка курсора в данную позицию. T1, T2 - номер строки и столбца.
PRINT	Оператор вывода текста

**Пример: использования LINE**

**REM использование LINE**

**SCREEN 12**

**LINE (10, 10)-(200, 10)**

**LINE (10, 20)-(200, 40), 2, B**

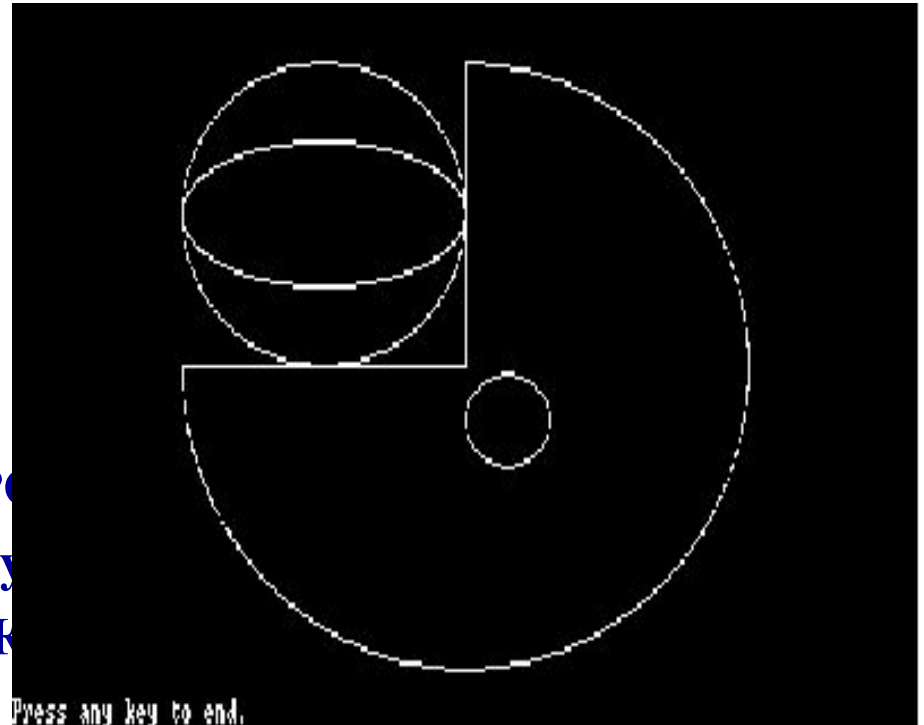
**LINE (10, 50)-(200, 70), 2, BF**

**END**

*Результат работы программы:*



```
Пример: использование CIRCLE
REM ОКРУЖНОСТЬ, ДУГА, ЭЛЛИПС
CONST PI = 3.141593
SCREEN 2
REM ОКРУЖНОСТЬ
CIRCLE (350, 115), 30
REM ДУГА ОКРУЖНОСТИ
CIRCLE (320, 100), 200, , -PI, -PI / 2
REM ОКРУЖНОСТЬ
CIRCLE STEP(-100, -42), 100
REM ЭЛЛИПС
CIRCLE STEP(0, 0), 100, , , , 5 / 25
REM ВЫВЕСТИ НАДПИСЬ В СТРОКЕ
LOCATE 25, 1: PRINT "Press any key to end."
REM ЖДЕМ НАЖАТИЯ ЛЮБОЙ КЛЮЧИКА
DO
LOOP WHILE INKEY$ = ""
```



*Результат работы программы:*

**Пример: построение окружности**

**REM окружность**

**CLS**

**INPUT "Введите координаты центра x,y: ", x,y**

**INPUT "Введите радиус окружности R: ", r**

**SCREEN 1**

**CIRCLE (x, y), r**

**END**

**Пример:** Рисование флагов.

**REM Флаги**

**SCREEN 1**

**PSET (50, 10)**

**DRAW "R20 G5 F5 L20 U10"**

**DRAW "B D20"**

**DRAW "S5 R20 G5 F5 L20 U10"**

**DRAW "BD20"**

**DRAW "S4 R20 G5 F5 L20 U10"**

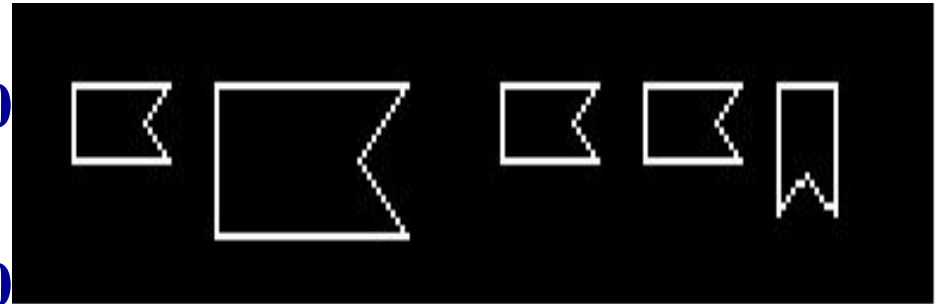
**DRAW "BD20"**

**DRAW "R20 G5 F5 L20 U10"**

**DRAW "BD40"**

**DRAW "A3 R20 G5 F5 L20 U10"**

**END**



# СОЗДАНИЕ ДВИЖУЩИХСЯ ИЗОБРАЖЕНИЙ.

1. Рисуем объект цветом отличным от цвета фона.
2. Рисуем объект цветом фона.
3. Изменяем координаты.
4. Повторяем 1-3 столько раз сколько потребуется.

**Пример: Движущийся круг.**

**REM Движущийся круг**

**SCREEN 1**

**x = 1**

**y = 1**

**REM цвет фона - 0(черный), цвет рисунка - 1**

**FOR i = 1 TO 150**

**REM Рисуем объект цветом отличным от цвета фона.**

**c = 1**

**CIRCLE (x, y), 2, c**

**REM задержка**

**FOR j = 1 TO 250000**

**NEXT j**

**Пример:** Усложним траекторию движения. Пусть шарик прыгает по поверхности, а когда поверхность закончится - упадет вниз.

**REM Прыгающий шарик**

**SCREEN 1**

**x = 1**

**y = 100**

**REM поверхность**

**LINE (0, y + 20)-(220, y + 20)**

**FOR i = 1 TO 140**

**c = 1**

**CIRCLE (x, y), 2, c**

**PAINT (x, y), c, c**

**FOR j = 1 TO 250000**

**NEXT j**

**c = 0**



## РАБОТА С ФАЙЛАМИ.

Имена файлов состоят из двух частей, разделяемых точкой: filename.ext (имя\_файла.расширение)

Имя файла может включать от 1 до 8 знаков, а соответствующее расширение - до трех знаков.

Имена файлов и расширения могут содержать следующие СИМВОЛЫ:

A-Z 0-9 () {} @ # \$ % ^ ! - \_ ' / ~

Файлы можно создавать, переименовывать, стирать; производить операции считывания и записи.

Basic предлагает три различных способа сохранения и востребования информации с диска:

последовательный, прямой и двоичный ввод/вывод файла.

## Создание последовательного файла:

1. **ОТКРЫТЬ** файл в режиме последовательного ВВОДА. Для создания файла необходимо использовать оператор **OPEN**.

В последовательных файлах есть два пути подготовки файла к выводу:

**OUTPUT (ВЫВОД)**: Если файл не существует- создается новый файл. Если файл уже существует, его содержание уничтожается, а сам файл рассматривается как новый.

**APPEND (ДОБАВИТЬ В КОНЕЦ)**: Если файл не существует- создается новый файл. Если файл уже существует, любые данные дописываются в конец этого файла.

2. Ввод данных в файл. Используйте **WRITE# PRINT#** или **PRINT#USING** для записи данных в последовательный файл.

3. **ЗАКРЫТИЕ** файла. Оператор **CLOSE** закрывает файловую переменную после завершения всех операций ввода/вывода.

## Для чтения последовательного файла:

1. **ОТКРЫТЬ** файл в режиме последовательного ВВОДА. Подготовить файл для считывания.
2. Считывать данные с файла. Использовать операторы **INPUT#**, **INPUT\$**, или **LINE INPUT#**.
3. **ЗАКРЫТЬ** файл. Оператор **CLOSE** закрывает файловую переменную после выполнения всех операций ввода/вывода.

**Недостаток** - возможен только последовательный доступ к данным.

**Пример:** Записать строку в файл, считать строку из файла.

**REM Работа с файлами.**

**REM Запись в файл**

**OPEN "file01.dat" FOR OUTPUT AS #1**

**A\$ = "Это наша текстовая строка"**

**PRINT #1, A\$**

**CLOSE #1**

**REM Чтение из файла**

**OPEN "file01.dat" FOR INPUT AS #1**

**INPUT #1, B\$**

**PRINT b\$ 'вывод на экран**

**CLOSE #1**

*В результате выполнения программы будет создан файл "file01.dat" и файл будет содержать строку Это наша текстовая строка. Затем файл будет открыт для чтения и из него будет прочитана и выведена на экран данная строка.*

## КОМБИНИРОВАННЫЕ ТИПЫ.

Для описания объекта «ученик» могут понадобиться, например, следующие характеристики:

**фамилия, имя и отчество (строки);**

**возраст (integer);**

**пол (строка);**

**класс (integer);**

**буква класса (символ);**

**и т.д.**

Описание комбинированного типа представляет собой список описаний его элементов; каждое описание похоже на описание простой переменной. Для примера, приведенного выше, описание комбинированного типа **PUPIL (ученик)** может выглядеть следующим образом:

**TYPE Pupil**

**fio AS STRING \* 20**

**age AS INTEGER**

**sex AS STRING \* 6**

**class AS INTEGER**

**classname AS STRING \* 1**

**END TYPE**

Определив собственный тип данных, можно использовать его для объявления переменных этого типа.

**DIM Schoolchildrens AS Pupil**

**DIM Group(1 TO 25) AS Pupil**

Доступ к компонентам (свойствам) переменной пользовательского типа осуществляется путем указания точки после имени переменной.

**Schoolchildrens.fio = "Иванов Иван"**

**Schoolchildrens.age = 15**

**Schoolchildrens.sex = "male"**

**Schoolchildrens.class = 10**

**Schoolchildrens.classname = "A"**

**PRINT Schoolchildrens.fio, Schoolchildrens.age, Schoolchildrens.sex,  
Schoolchildrens.class, Schoolchildrens.classname**

**Пример:**

**REM использование комбинированных типов**

**REM описание типа ученик**

**TYPE Pupil**

**fio AS STRING \* 20**

**age AS INTEGER**

**sex AS STRING \* 6**

**class AS INTEGER**

**classname AS STRING \* 1**

**END TYPE**

**REM объявление массива из 3 элементов типа ученик**

**DIM Group(1 TO 3) AS Pupil**

**Group(1).fio = "Иванов Иван"**

**Group(1).age = 15**

**Group(1).sex = "male"**

**Group(1).class = 10**

**Group(1).classname = "А"**

**Group(2).fio = "Петрова Маша"**

**Group(2).age = 14**