A sunset scene over the ocean. A large, bright sun is low on the horizon, casting a warm orange glow. In the center, a lighthouse silhouette stands on a small island. To the left, a dolphin is leaping from the water. Several birds are flying in the sky. The water reflects the sun and the lighthouse.

# **Базовые алгоритмические структуры языка Basic**

*(с примерами)*

**Автор: Кутузова Е. С. 11 А класс**

# Содержание:

1. Происхождение понятия «алгоритм»
2. Что такое алгоритм?
3. Основные виды алгоритмов
4. Линейный алгоритм
5. Разветвляющийся алгоритм
6. Циклический алгоритм
7. Вспомогательный алгоритм
8. Блок-схемы
9. Блок-схема для линейных алгоритмов
10. Блок-схемы для алгоритмов с ветвлением
11. Блок-схемы для циклических алгоритмов
12. Блок-схема для вспомогательных алгоритмов



# Содержание:

## 13. Microsoft Quick BASIC

- Среда разработчика Quick BASIC
- Основные операторы
- Другие операторы
- Линейные алгоритмические структуры
- Пример линейной программы
- Алгоритмические структуры с ветвлением
- Пример программы с ветвлением (однострочная форма)
- Пример программы с ветвлением (многострочная форма)
- Циклические алгоритмические структуры
- Пример циклической программы, реализованной с помощью цикла с параметром



# Содержание:

## 24. Циклы ДО и ПОКА

### 25. Пример циклической программы, реализованной с помощью цикла ДО

- Пример циклической программы, реализованной с помощью цикла ДО
- Пример циклической программы, реализованной с помощью цикла ПОКА
- Пример циклической программы, реализованной с помощью цикла ПОКА
- Алгоритмическая структура для программ с подпрограммами
- Пример программы с подпрограммой
- Вместо заключения

# Происхождение понятия «алгоритм».

- Слово **«алгоритм»** происходит от имени выдающегося математика средневекового Востока Мухаммеда аль-Хорезми. Им впервые были предложены приемы выполнения математических вычислений с многозначными числами. Позже в Европе эти приемы стали называть **алгоритмами** от «Algorithmi»-латинского написания имени аль-Хорезми. В наше время понятие алгоритма понимается шире, не ограничиваясь арифметическими вычислениями.

# Что такое алгоритм?

- **Алгоритм** - понятное и очень точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.
- *Для алгоритма строго не определяется форма его представления.* Алгоритм можно изобразить графически, можно записать специальными символами, но программа должна быть записана на языке исполнителя (для ЭВМ это язык программирования).

# Основные виды алгоритмов

- Алгоритмы бывают четырех основных видов: **линейные** (самые простые), **с ветвлением** (разветвляющиеся), **циклические** и **вспомогательные**. А теперь рассмотрим подробнее каждый вид алгоритмов.



# Линейный алгоритм:

1. Линейный или последовательный алгоритм - описание таких действий, которые выполняются однократно в заданном порядке.

*Пример:* алгоритм решения задачи (от записи данных до получения ответа), алгоритм открывания двери (вставить ключ, повернуть ключ, открыть дверь) и т.д.



# Разветвляющийся алгоритм:

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

*Пример:* алгоритм покупки билетов (спрашиваем в кассе, есть ли билеты, если билеты есть, *то* подаем деньги, получаем билеты) и т.д.



# Циклический алгоритм:

3. Циклический алгоритм – описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие. *Перечень повторяющихся действий называют телом цикла.*

*Пример:* алгоритм нахождения значений  $y$  при заданных или задающихся значениях  $x$  для построения графика функции.



# Вспомогательный алгоритм:

4. Вспомогательный алгоритм — это алгоритм, который можно использовать в других алгоритмах, указав только его имя. *Вспомогательному алгоритму должно быть присвоено имя.*

*Пример:* алгоритм для определения корней любого квадратного уравнения (алгоритм нахождения дискриминанта является как бы **вспомогательным алгоритмом** и находится внутри основного).

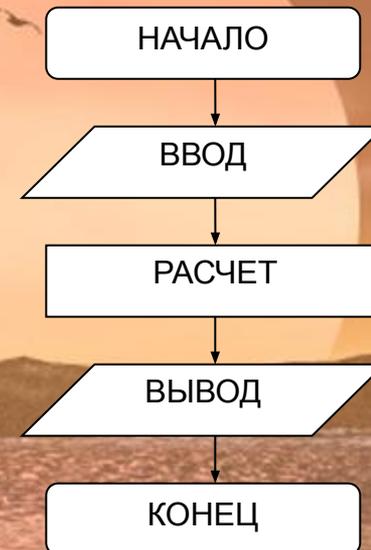


# Блок-схемы:

- Любой алгоритм очень удобно представлять в виде блок-схемы, так как это не только самый наглядный и простой способ представления алгоритма, но и лучший способ представления алгоритма для перевода его на любой язык программирования.

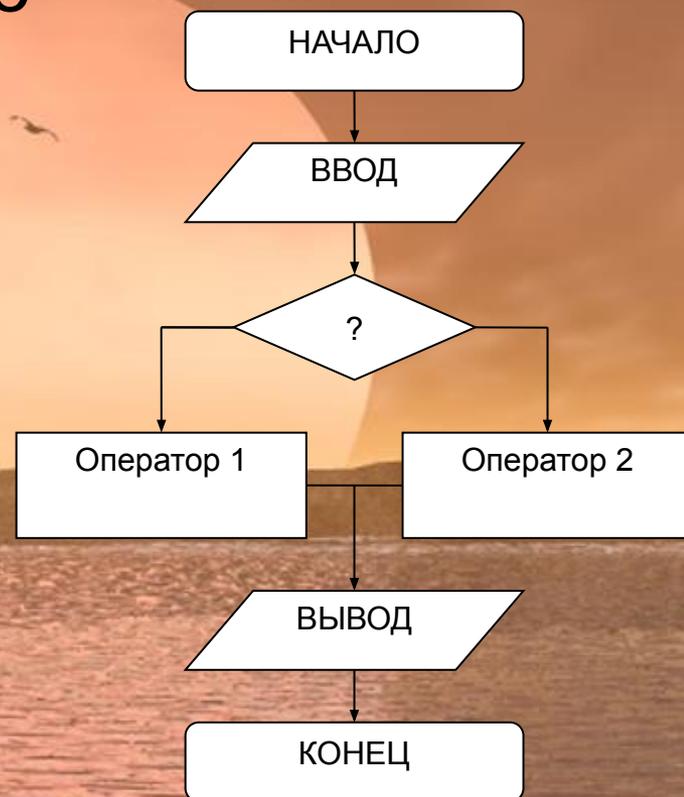
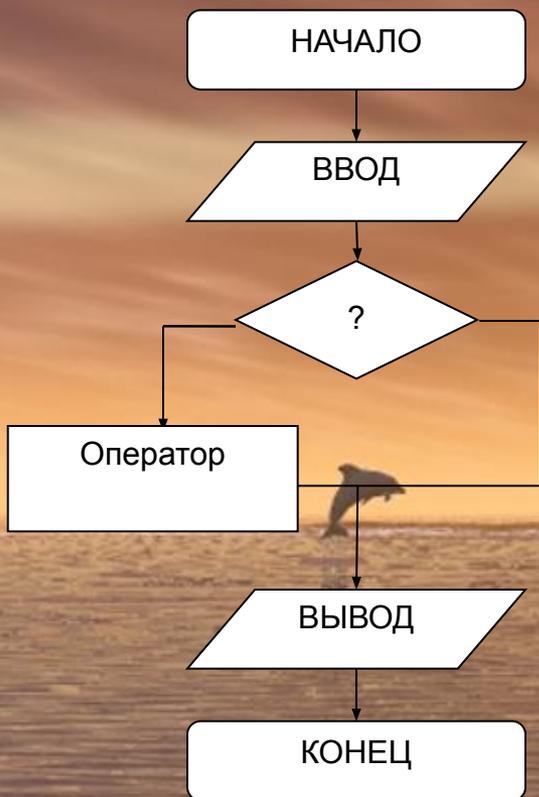
# Блок-схема для линейных алгоритмов

- Так в общем виде выглядит блок-схема для линейных алгоритмов:



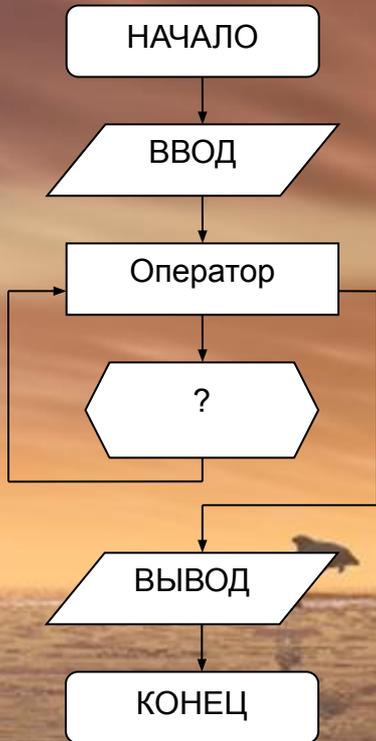
# Блок-схемы для алгоритмов с ветвлением

- Так в общем виде выглядят блок-схемы для алгоритмов с ветвлением:

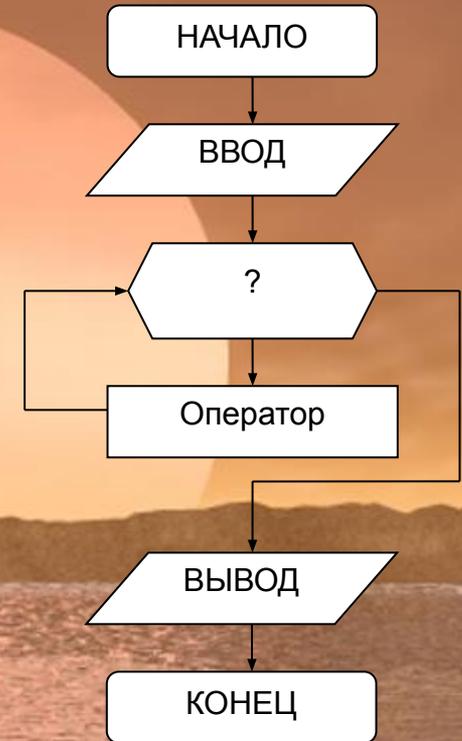


# Блок-схемы для циклических алгоритмов

- Блок схемы для циклических алгоритмов бывают двух видов: **цикл ДО** (на рис. слева) и **цикл ПОКА** (на рис. справа)



Цикл До



Цикл ПОКА

# Блок-схема для вспомогательных алгоритмов

- Так в общем виде выглядит блок-схема для вспомогательных алгоритмов:



# Microsoft Quick BASIC

- **Microsoft Quick BASIC** – это один из самых известных, простых и понятных языков программирования. Основам программирования на этом языке посвящено очень много книг и учебных пособий. Далее мы познакомимся со средой Microsoft Quick BASIC и разберем примеры программ на этом языке.



# Среда разработчика *Quick BASIC*

- Так выглядит среда Microsoft Quick Basic когда мы запускаем файл QBASIC.exe:



# Основные операторы:

- Оператор - это ключевое слово в строке программы.
1. **Let** – оператор присваивания (с англ. Пусть).
  2. **Print** – оператор вывода (с англ. Печать).
  3. **Input** – оператор ввода (с англ. Ввод).
  4. **End** – оператор конца программы (с англ. Конец).
  5. **If then else** – оператор условного перехода.
  6. **For to step - next While, Repeat, Until, Loop** – операторы цикла.
  7. **Rem** – пустой оператор пояснения.
  8. **Gosub** и **return** – операторы подпрограммы.

## Другие операторы:

- Нужно заметить, что выше мной были перечислены далеко не все операторы. Существует ещё оператор **cls** (оператор очистки экрана), операторы графики, такие как **pset**, **line**, **circle**, **paint**, **draw** и др., оператор массива **dim** и ещё много других. Но их полное изучение заняло бы очень много времени. А теперь мы переходим к примерам программ, составленных на языке Quick Basic.

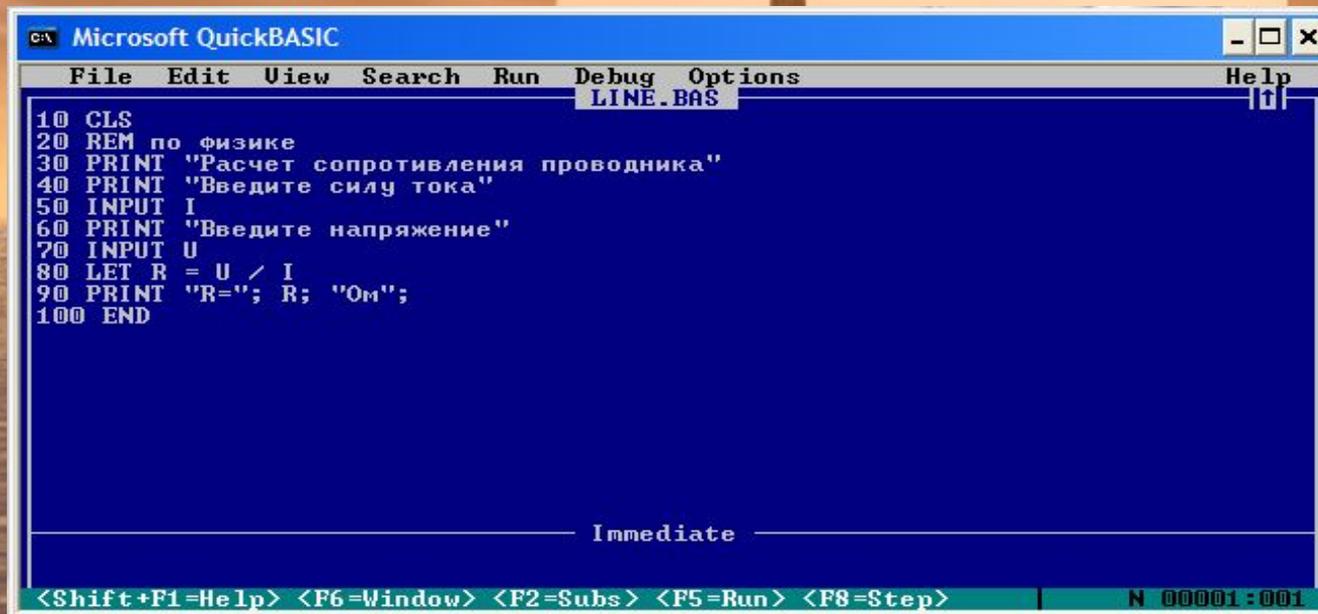
# Линейная алгоритмическая структура

- Самый простой вид алгоритмической структуры – **линейная**. Она выглядит так:

1. **Ввод** (оператор **INPUT**),
2. **Расчет** по формуле,
3. **Вывод** (оператор **PRINT**).

# Пример линейной программы:

- Типичный пример линейной программы – это программа расчета по физическим или математическим формулам. Здесь: расчет сопротивления проводника, если известна сила тока и напряжение.



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
LINE.BAS
10 CLS
20 REM по физике
30 PRINT "Расчет сопротивления проводника"
40 PRINT "Введите силу тока"
50 INPUT I
60 PRINT "Введите напряжение"
70 INPUT U
80 LET R = U / I
90 PRINT "R="; R; "Ом";
100 END

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 000001:001
```

# Алгоритмические структуры с ветвлением

- Алгоритмическая структура с ветвлением может быть записана двумя способами:

- **Многострочная форма**

1. **If** *Условие* **Then**

2. *Серия 1*

3. **Else**

4. *Серия 2*

5. **End If**

- **Однострочная форма**

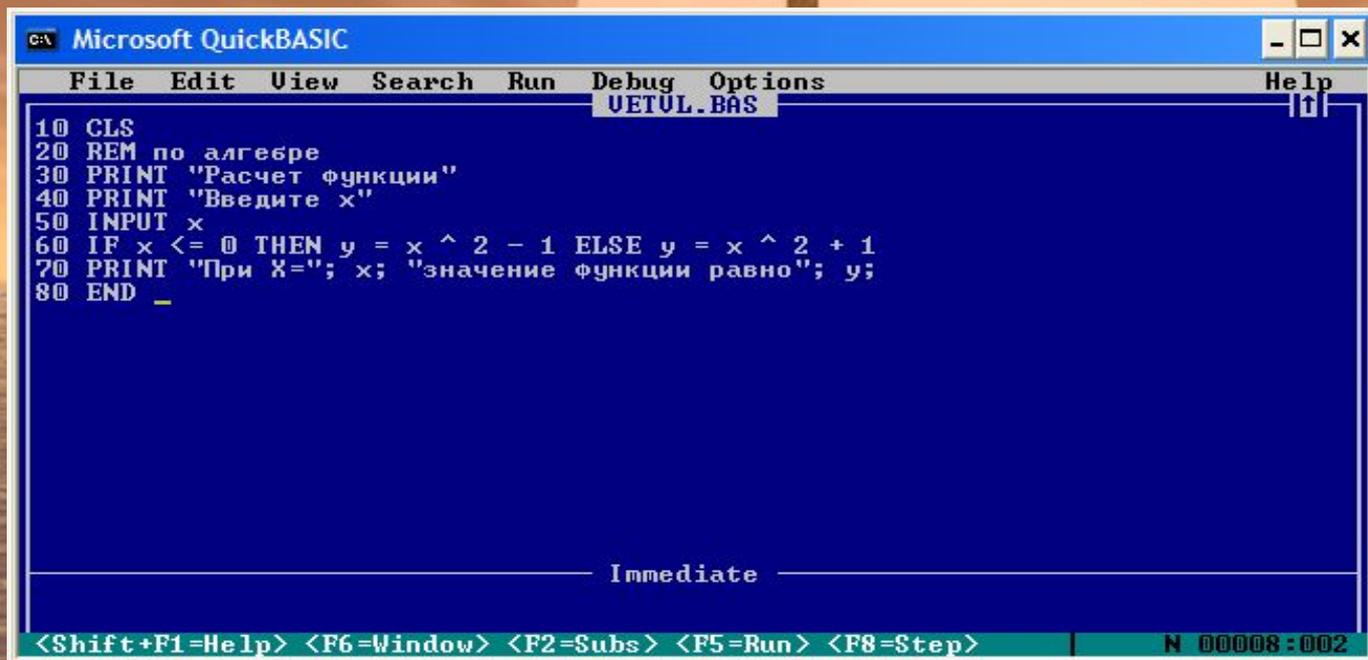
1. **If** *Условие*

2. **Then** *Серия 1a*

3. **Else** *Серия 2a*

# Пример программы с ветвлением (однострочная форма):

- **Здесь:** расчет функции  $y=x^2+1$ , если  $x>0$  или  $y=x^2-1$ , если  $x\leq 0$ .



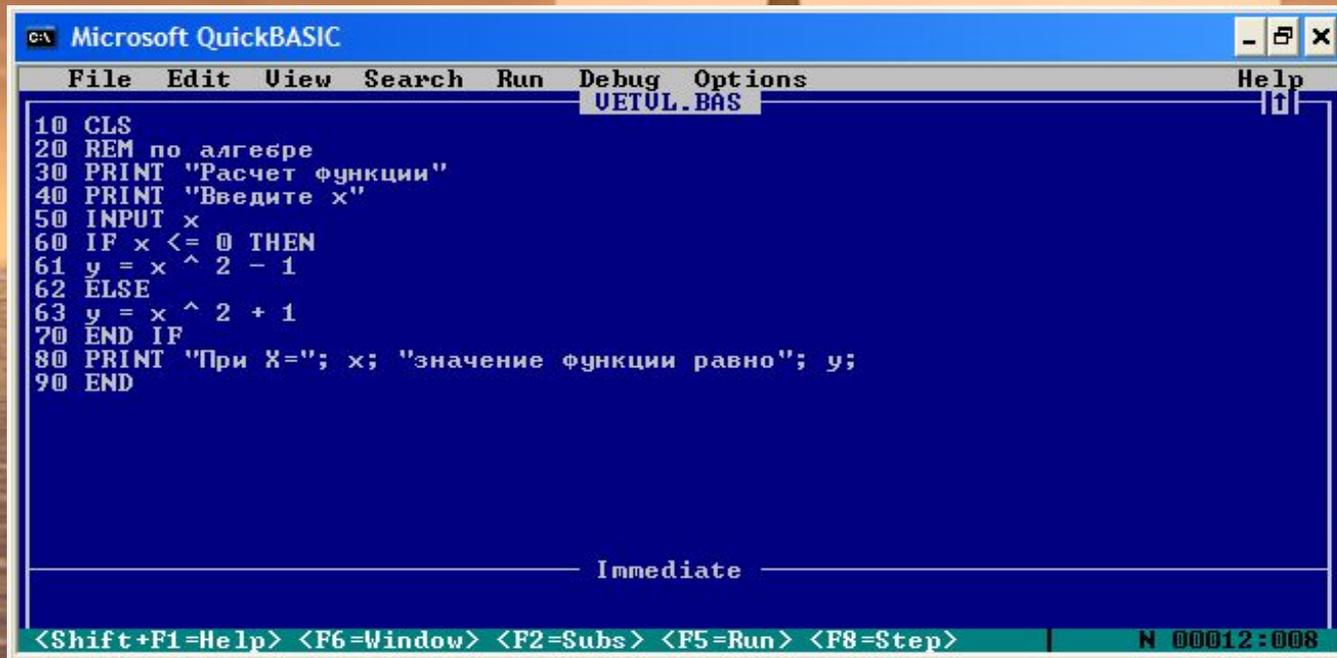
```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
UETUL.BAS
10 CLS
20 REM по алгебре
30 PRINT "Расчет функции"
40 PRINT "Введите x"
50 INPUT x
60 IF x <= 0 THEN y = x ^ 2 - 1 ELSE y = x ^ 2 + 1
70 PRINT "При X="; x; " значение функции равно"; y
80 END _

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00008:002
```

# Пример программы с ветвлением (многострочная форма):

- Типичный пример программы с ветвлением – это программа расчета функции не определенной на всей числовой прямой. Здесь: расчет функции  $y=x^2+1$ , если  $x>0$  или  $y=x^2-1$ , если  $x\leq 0$ .



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
VETUL.BAS
10 CLS
20 REM по алгебре
30 PRINT "Расчет функции"
40 PRINT "Введите x"
50 INPUT x
60 IF x <= 0 THEN
61 y = x ^ 2 - 1
62 ELSE
63 y = x ^ 2 + 1
70 END IF
80 PRINT "При X="; x; "значение функции равно"; y;
90 END

Immediate

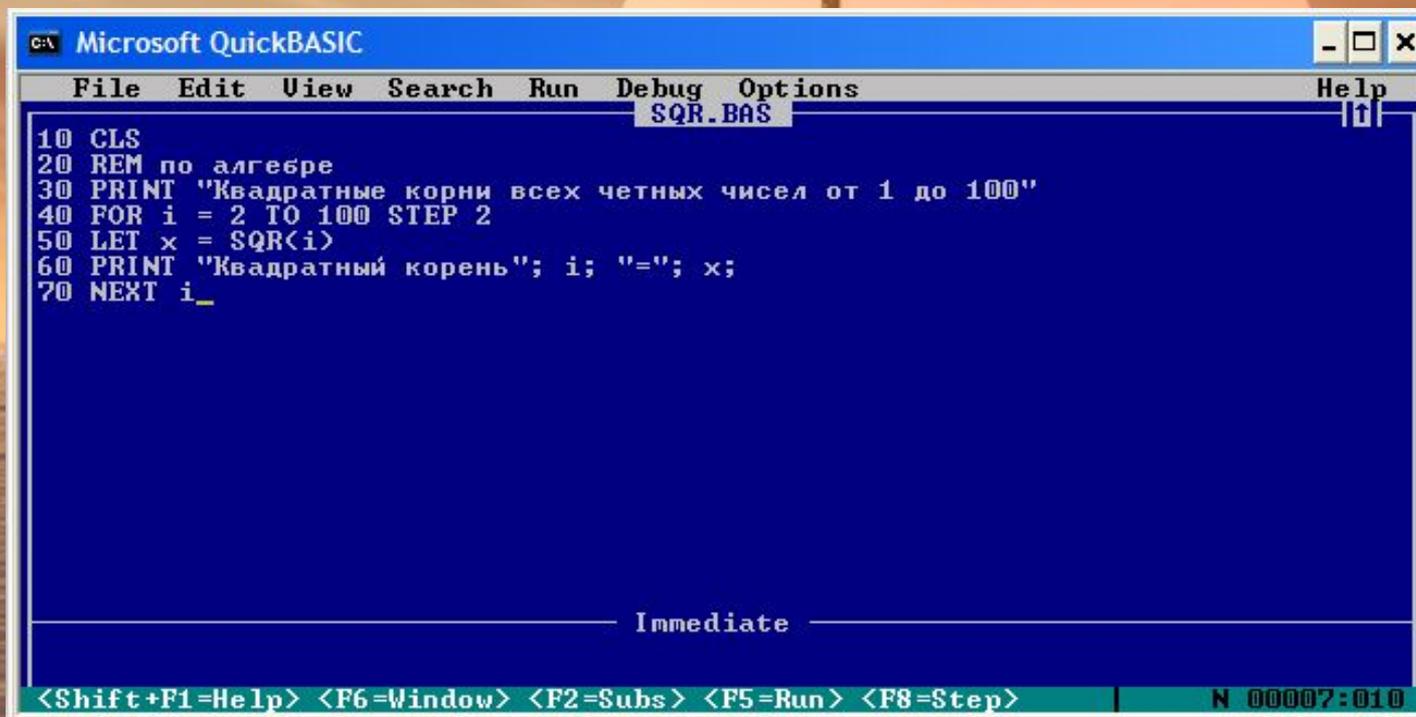
<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00012:008
```

# Циклические алгоритмические структуры

- Циклические алгоритмические структуры бывают трех видов: цикл с параметром, цикл с предусловием(цикл **ДО**) и цикл с последующим условием(цикл **ПОКА**). Цикл с параметром осуществляется за счет оператора **FOR TO STEP – NEXT**, а циклы **ДО** и **ПОКА** за счет операторов **DO...LOOP** и ключевых слов **WHILE** и **UNTIL**.

# Пример циклической программы, реализованной с помощью цикла с параметром:

- Программа вывода квадратных корней всех четных чисел от 1 до 100:



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
SQR.BAS
10 CLS
20 REM по алгебре
30 PRINT "Квадратные корни всех четных чисел от 1 до 100"
40 FOR i = 2 TO 100 STEP 2
50 LET x = SQR(i)
60 PRINT "Квадратный корень"; i; "="; x;
70 NEXT i_

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00007:010
```

# Циклы **ДО** и **ПОКА**:

**Do ... Loop** – оператор цикла с условием. Существуют 2 ключевых слова, которые передают противоположный смысл:

**While** – пока выполняется условие,

**Until** – пока не выполняется условие.

- Цикл с предусловием

**Do While** *Условие*

*Тело Цикла*

**Loop**

ИЛИ

**Do Until** *Условие*

*Тело цикла*

**Loop**

- Цикл с постусловием.

**Do**

*Тело цикла*

**Loop While** *Условие*

ИЛИ

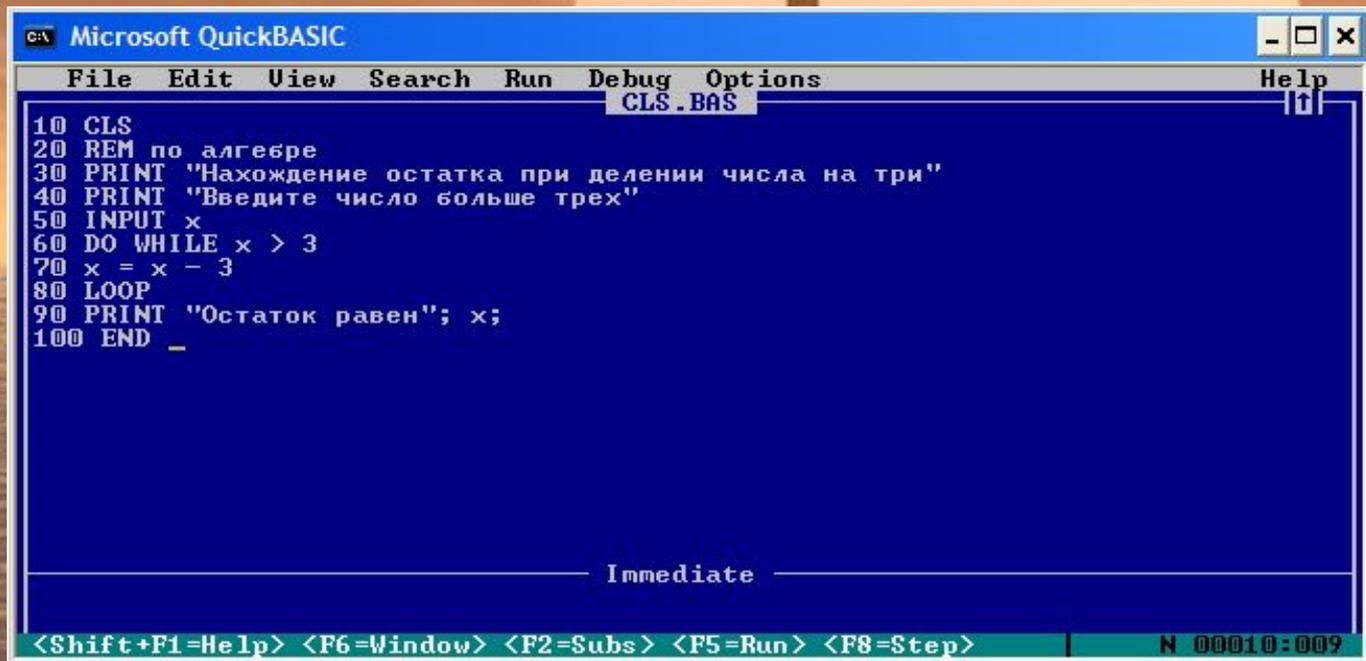
**Do**

*Тело цикла*

**Loop Until** *Условие*

# Пример циклической программы, реализованной с помощью цикла ДО:

- Используем ключевое слово While.  
**Здесь:** программа на нахождение остатка при делении какого-либо числа на три.



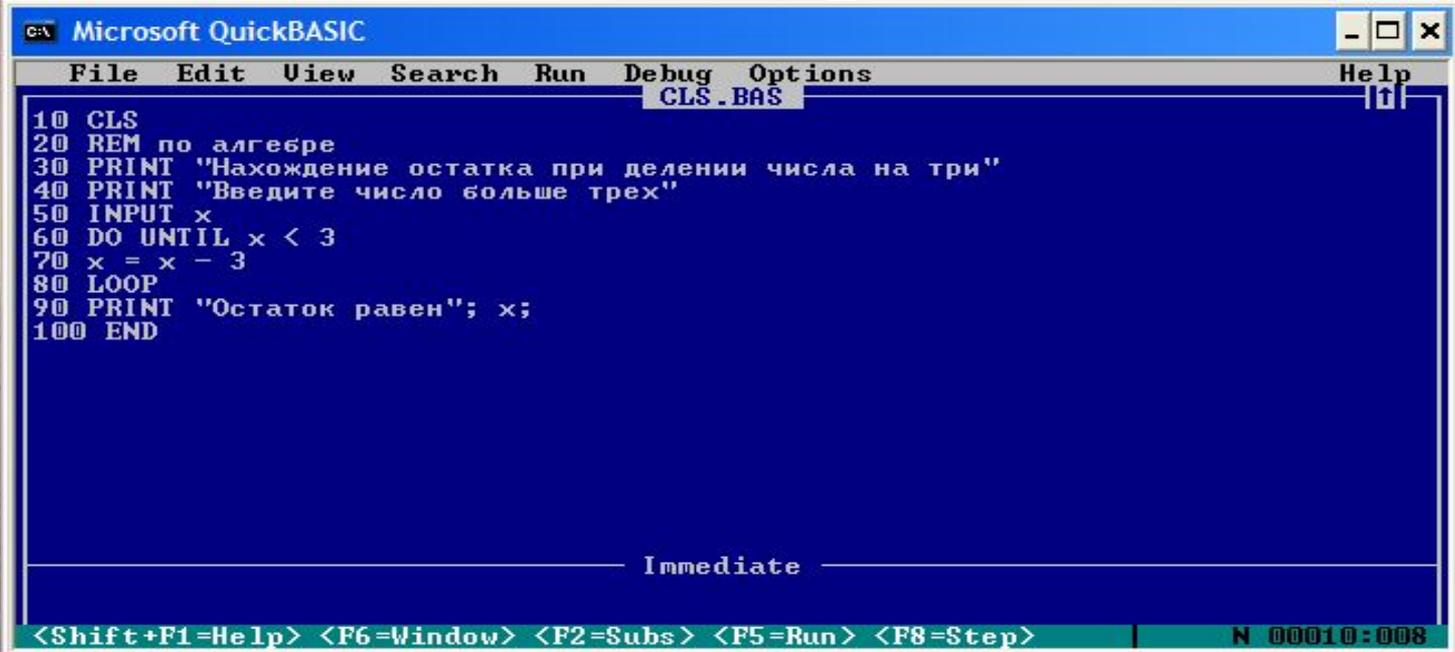
```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
CLS.BAS
10 CLS
20 REM по алгебре
30 PRINT "Нахождение остатка при делении числа на три"
40 PRINT "Введите число больше трех"
50 INPUT x
60 DO WHILE x > 3
70 x = x - 3
80 LOOP
90 PRINT "Остаток равен"; x;
100 END _

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00010:009
```

# Пример циклической программы, реализованной с помощью цикла ДО:

- Используем ключевое слово Until.  
**Здесь:** та же самая задача, что и на предыдущем слайде, но с использованием ключевого слова Until.



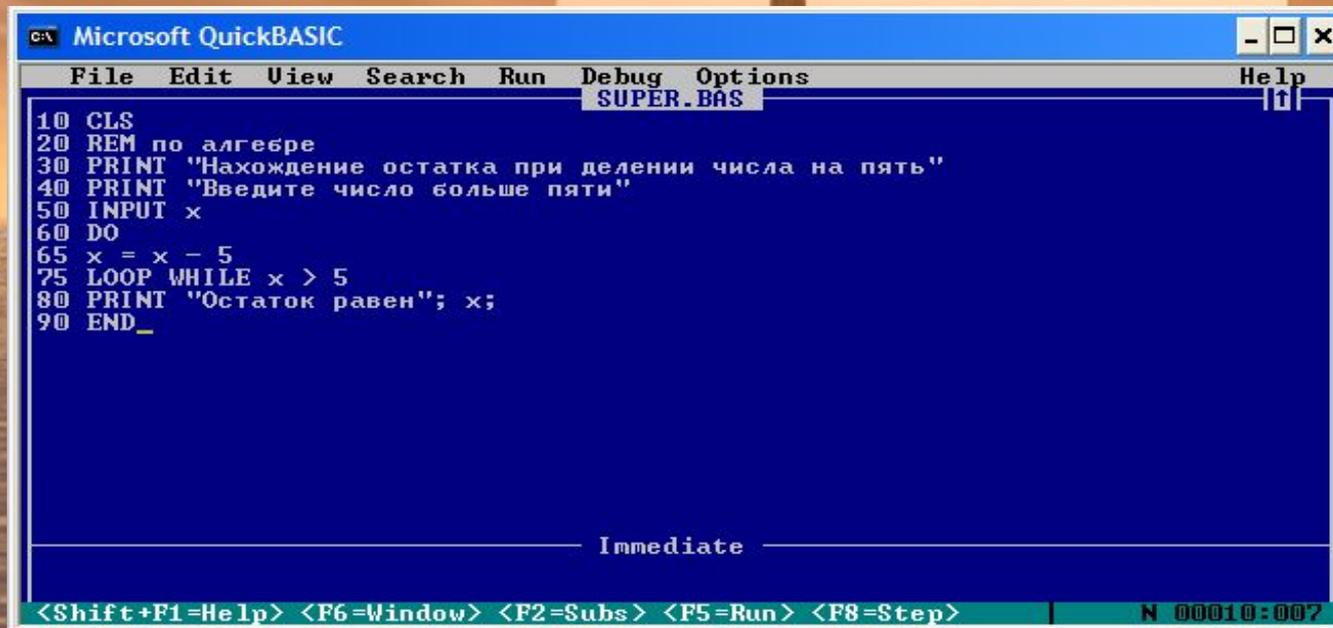
```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
CLS.BAS
10 CLS
20 REM по алгебре
30 PRINT "Нахождение остатка при делении числа на три"
40 PRINT "Введите число больше трех"
50 INPUT x
60 DO UNTIL x < 3
70 x = x - 3
80 LOOP
90 PRINT "Остаток равен"; x;
100 END

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00010:008
```

## Пример циклической программы, реализованной с помощью цикла ПОКА:

- Используем ключевое слово While.  
**Здесь:** программа на нахождение остатка при делении какого-либо числа на пять.



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
SUPER.BAS
10 CLS
20 REM по алгебре
30 PRINT "Нахождение остатка при делении числа на пять"
40 PRINT "Введите число больше пяти"
50 INPUT x
60 DO
65 x = x - 5
75 LOOP WHILE x > 5
80 PRINT "Остаток равен"; x;
90 END_

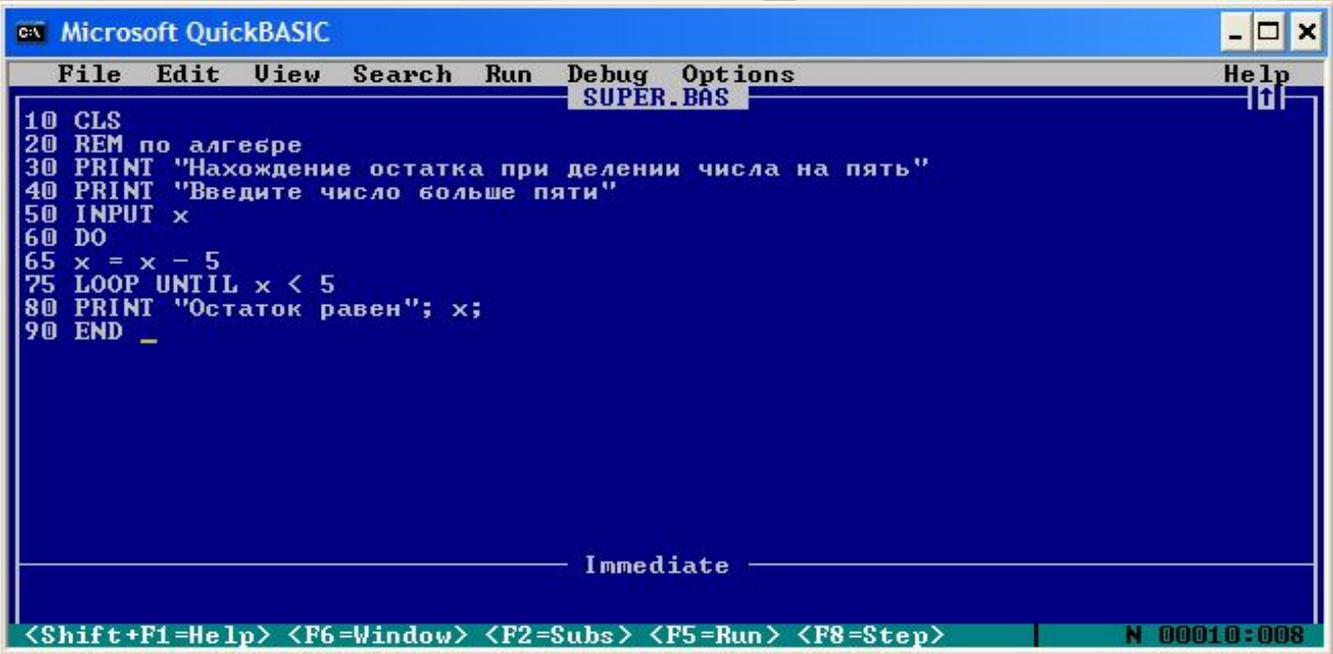
Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00010:007
```



## Пример циклической программы, реализованной с помощью цикла ПОКА:

- Используем ключевое слово Until.  
**Здесь:** та же самая задача, что и на предыдущем слайде, но с использованием ключевого слова Until.



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
SUPER.BAS
10 CLS
20 REM по алгебре
30 PRINT "Нахождение остатка при делении числа на пять"
40 PRINT "Введите число больше пяти"
50 INPUT x
60 DO
65 x = x - 5
75 LOOP UNTIL x < 5
80 PRINT "Остаток равен"; x;
90 END _

Immediate

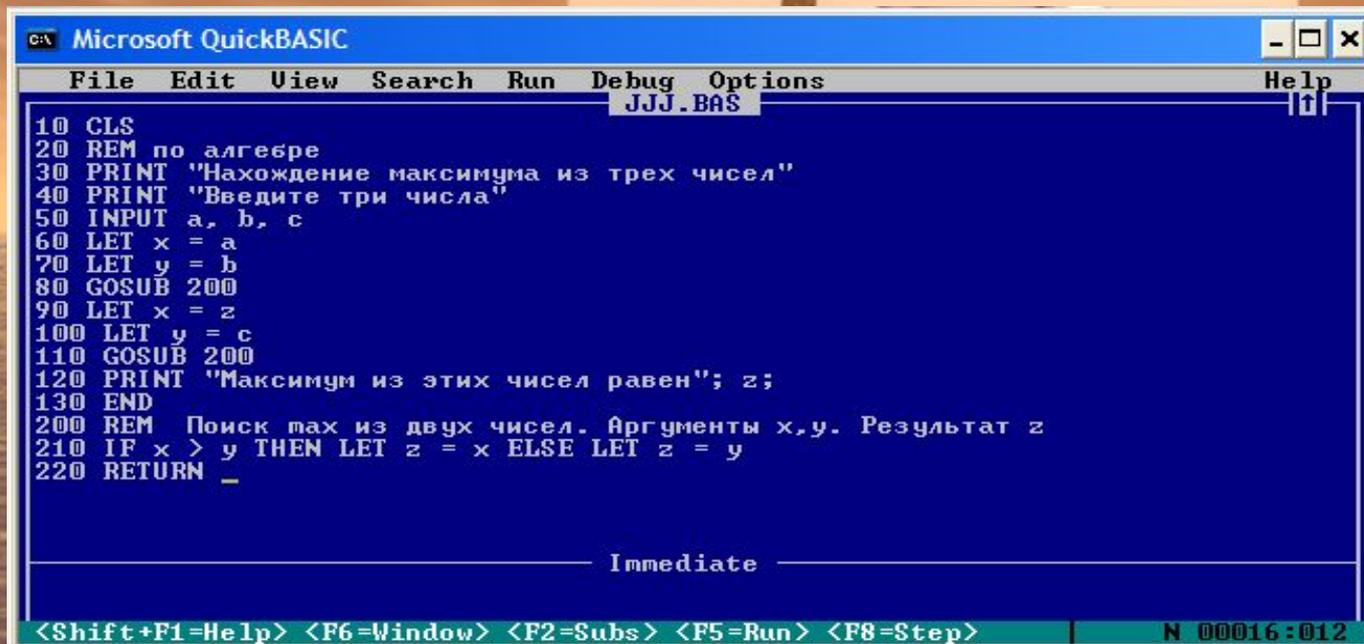
<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00010:008
```

# Алгоритмическая структура для программ с подпрограммами

- Алгоритмическая структура для программ с подпрограммами выглядит так:
  1. **Ввод** (оператор **INPUT**)
  2. **Вычисления** по формуле и обращение к подпрограмме (операторы **GOSUB** и **RETURN**).
  3. **Вывод** (оператор **PRINT**).
  4. **Подпрограмма** (описание подпрограммы).
  5. **Конец** программы (оператор **END**)

# Пример программы с подпрограммой:

- Оператор **GOSUB** осуществляет переход на подпрограмму, а **RETURN** возвращает обратно, т.е. вместе осуществляют обращение к подпрограмме. **Здесь:** программа нахождения максимума из трех чисел.



```
Microsoft QuickBASIC
File Edit View Search Run Debug Options Help
JJJ.BAS
10 CLS
20 REM по алгебре
30 PRINT "Нахождение максимума из трех чисел"
40 PRINT "Введите три числа"
50 INPUT a, b, c
60 LET x = a
70 LET y = b
80 GOSUB 200
90 LET x = z
100 LET y = c
110 GOSUB 200
120 PRINT "Максимум из этих чисел равен"; z;
130 END
200 REM Поиск max из двух чисел. Аргументы x,y. Результат z
210 IF x > y THEN LET z = x ELSE LET z = y
220 RETURN _

Immediate

<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step> | N 00016:012
```

# Вместо заключения

- Мы познакомились с основными алгоритмическими структурами языка программирования Quick Basic. Хочется заметить, что в данной презентации описаны далеко не все операторы и типы программ. Здесь не затронуты программы с использованием графики, работа с массивами и др., так как эти темы настолько обширны, что для них потребовалось бы создать отдельную презентацию. Я надеюсь, что эта презентация будет полезна при изучении языка Quick Basic.

КОНЕЦ

