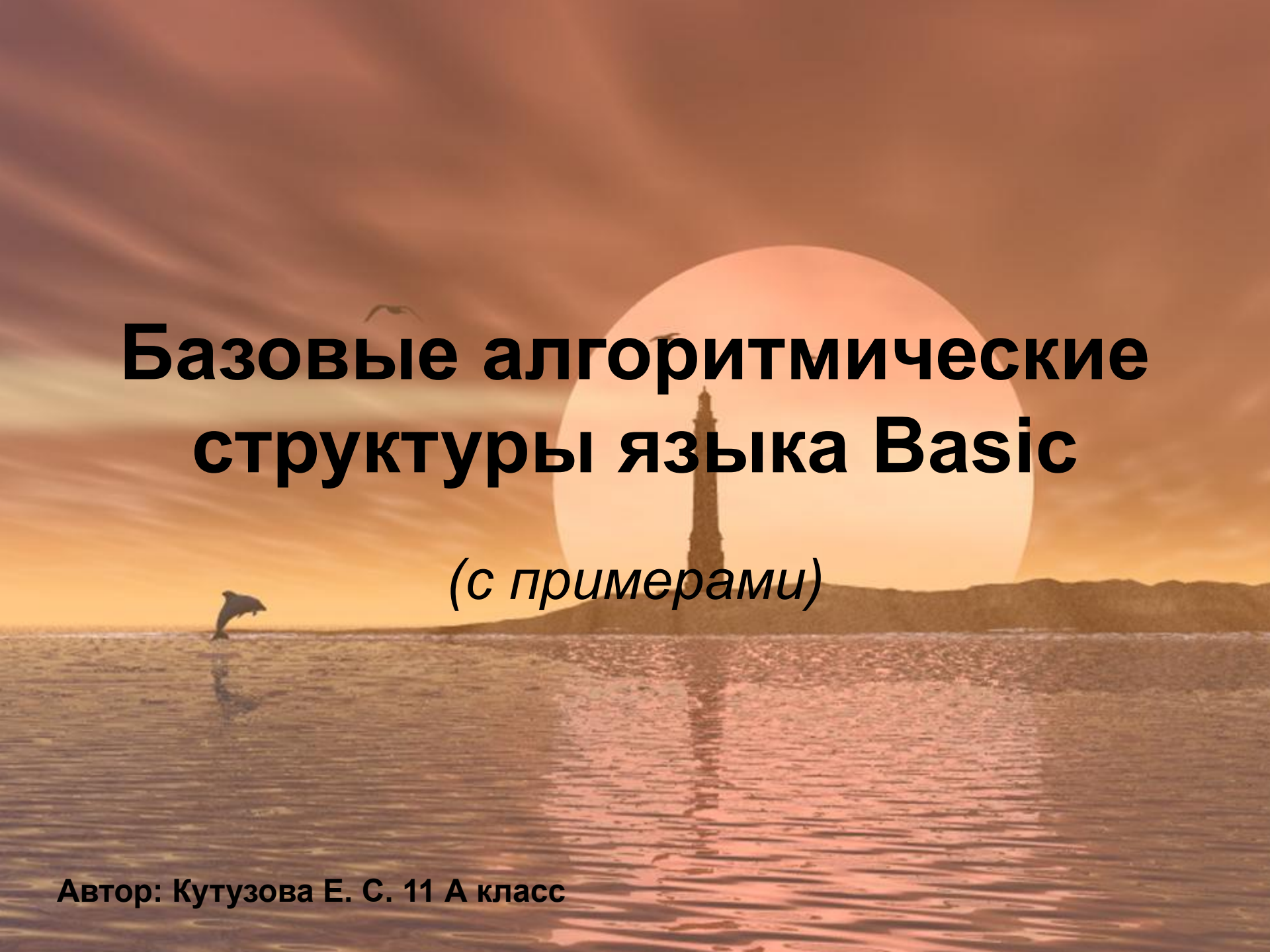


Базовые алгоритмические структуры языка Basic

(с примерами)

Автор: Кутузова Е. С. 11 А класс

Содержание:

Содержание:

Содержание:

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.*

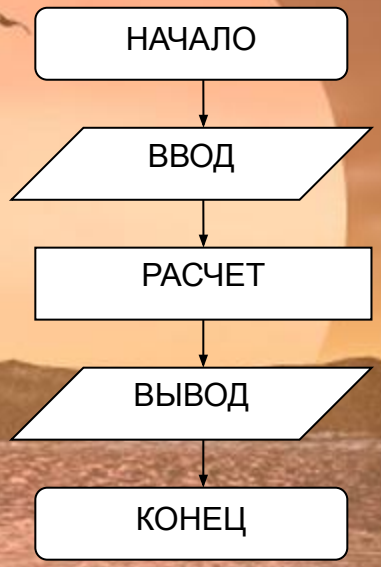
Пример: алгоритм для определения корней любого квадратного уравнения (алгоритм нахождения дискриминанта является как бы вспомогательным алгоритмом и находится внутри основного.

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.**

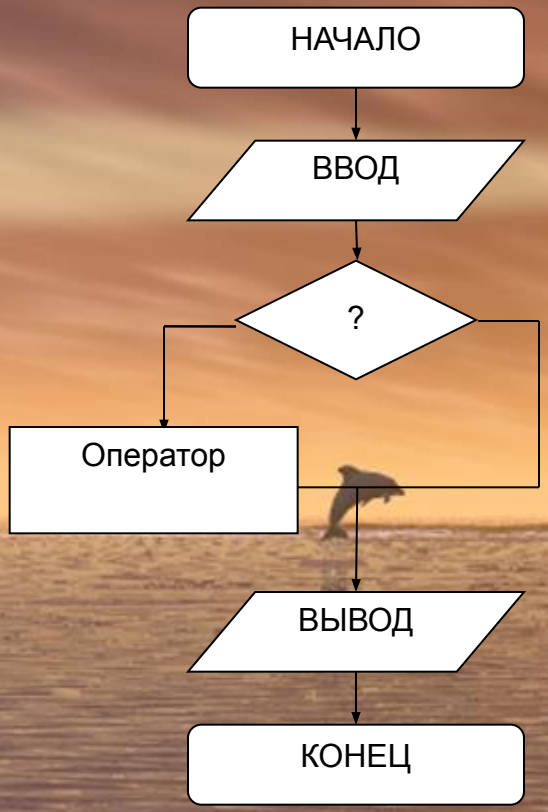
Блок-схема для линейных алгоритмов

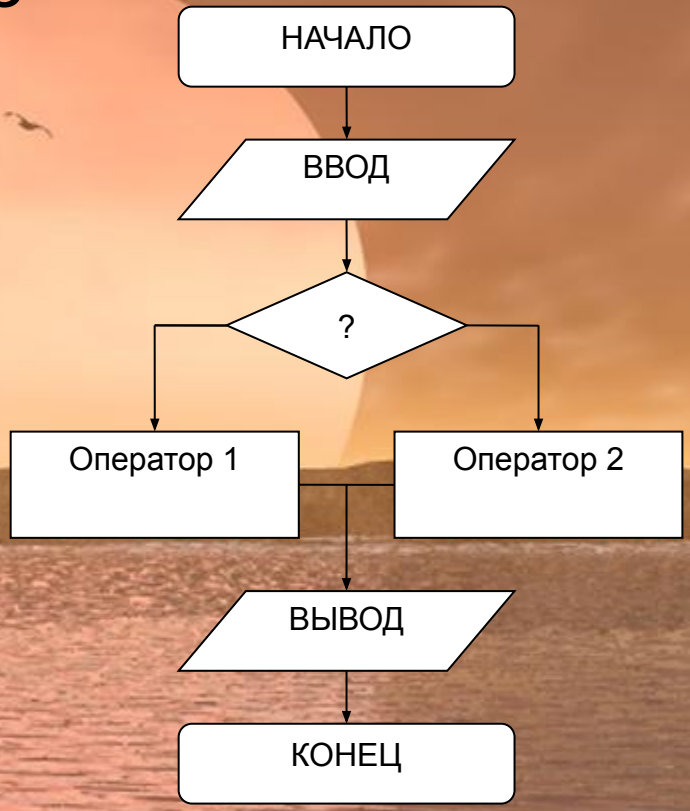
- Так в общем виде выглядит блок-схема для линейных алгоритмов:



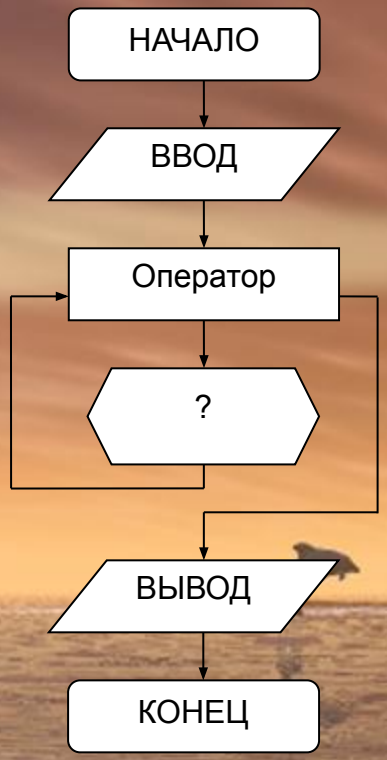
Блок-схемы для алгоритмов с ветвлением

- 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.

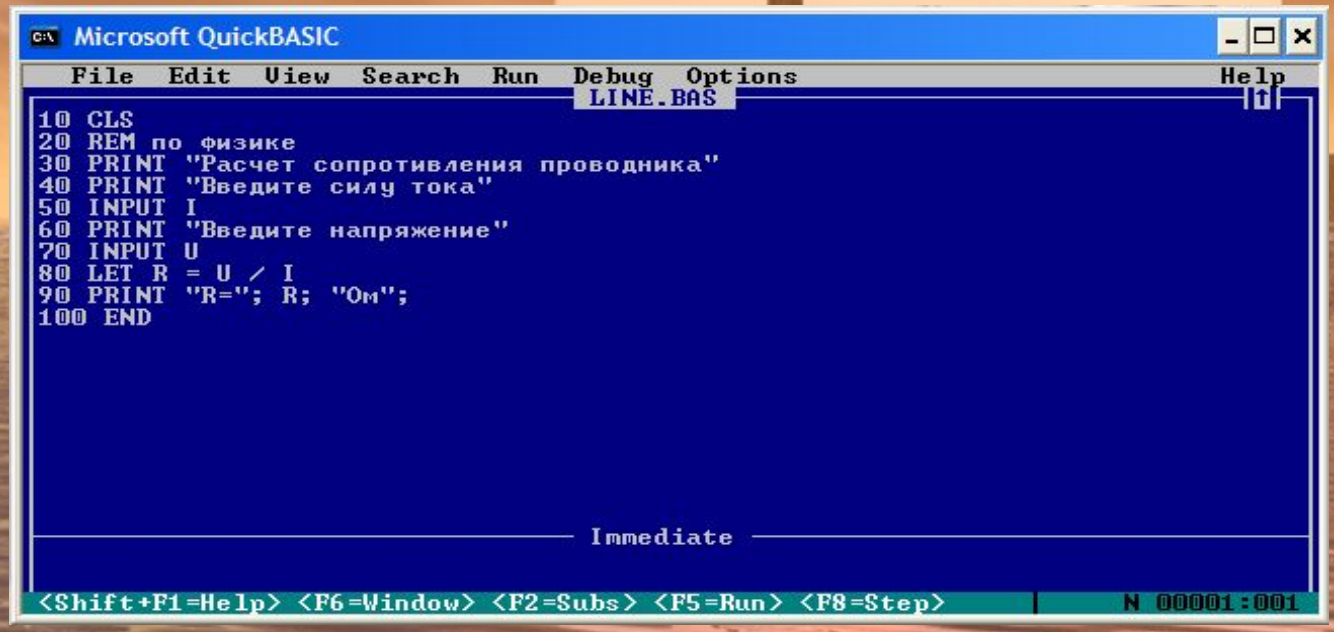
Линейная алгоритмическая структура

- 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.*



Алгоритмические структуры с ветвлением

- Алгоритмическая структура с ветвлением может быть записана двумя способами:
- **Многострочная форма**

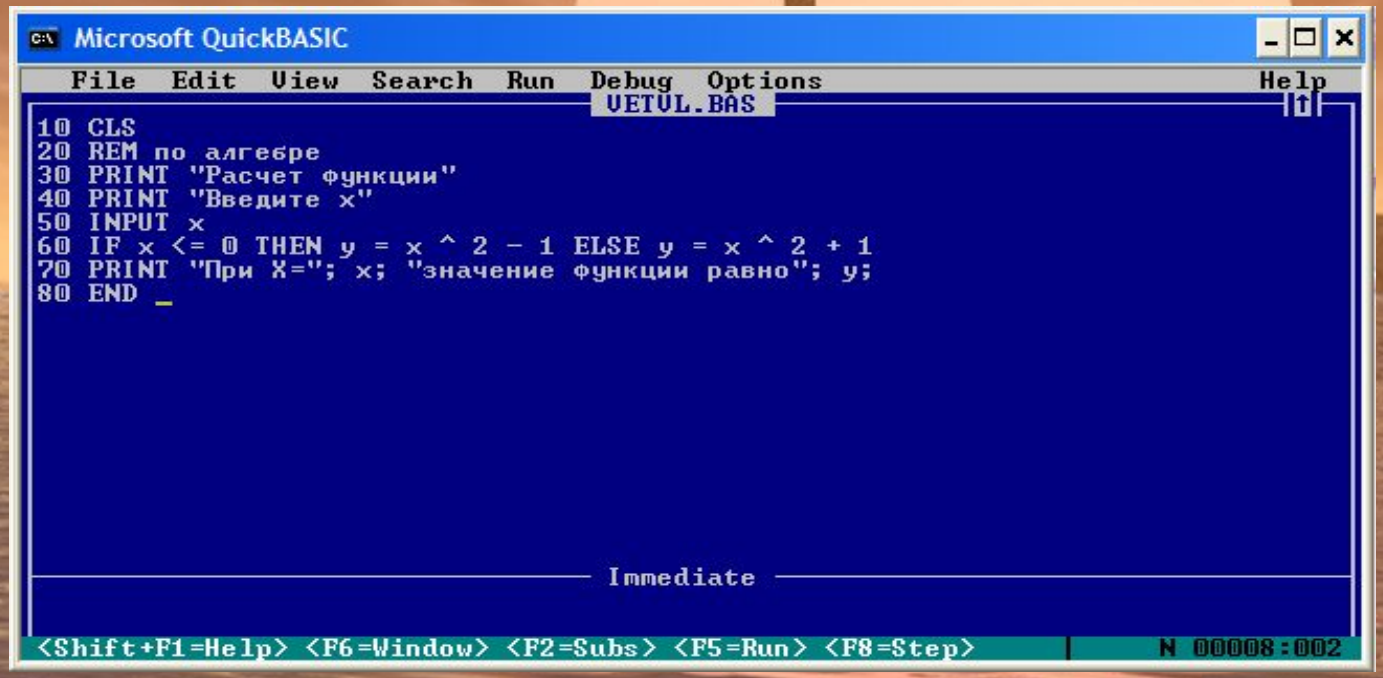
1. **If** *Условие* **Then**
2. *Серия 1*
3. **Else**
4. *Серия 2*
5. **End If**

- **Однострочная форма**

1. **If** *Условие*
2. **Then** *Серия 1а*
3. **Else** *Серия 2а*

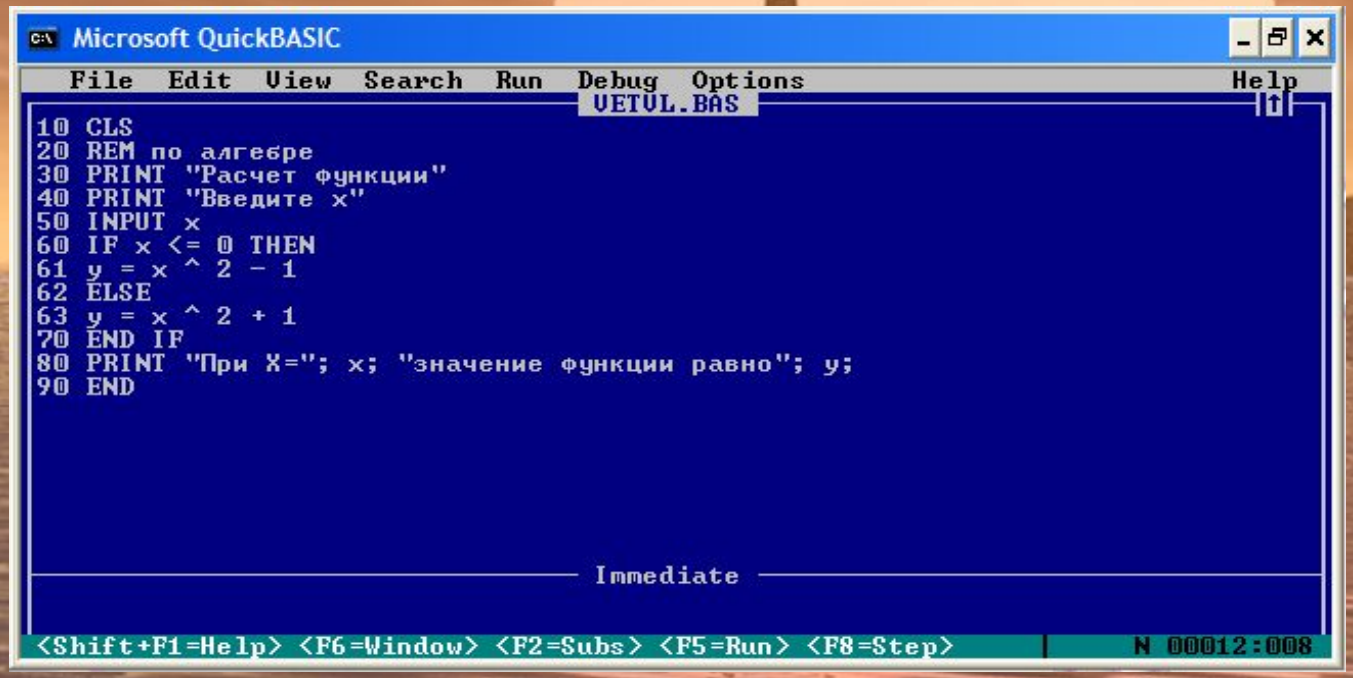
Пример программы с ветвлением (однострочная форма):

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



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≤0.*

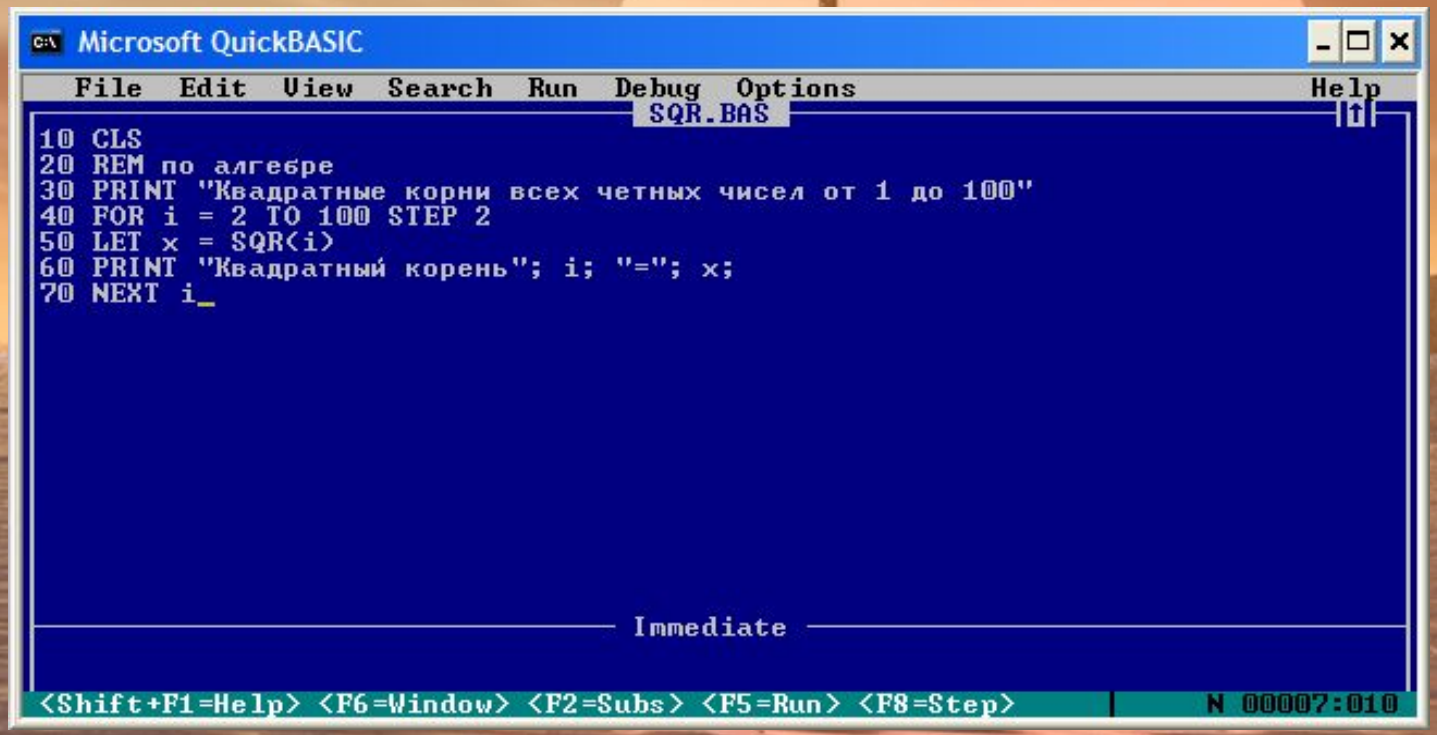


Циклические алгоритмические структуры

- 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 SPEP – 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:*



Циклы ДО и ПОКА:

Do … Loop – оператор цикла с условием. Существуют 2 ключевых слова, которые предают 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

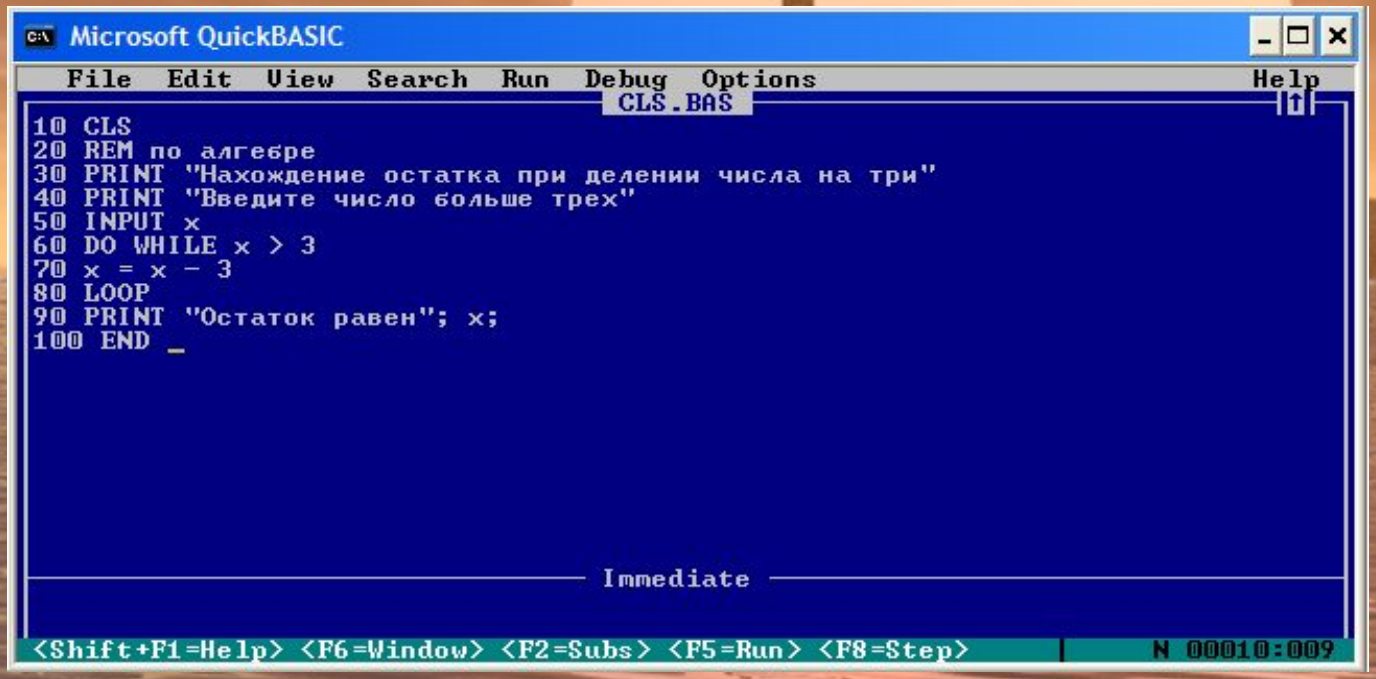
- **Цикл с постусловием.**

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.

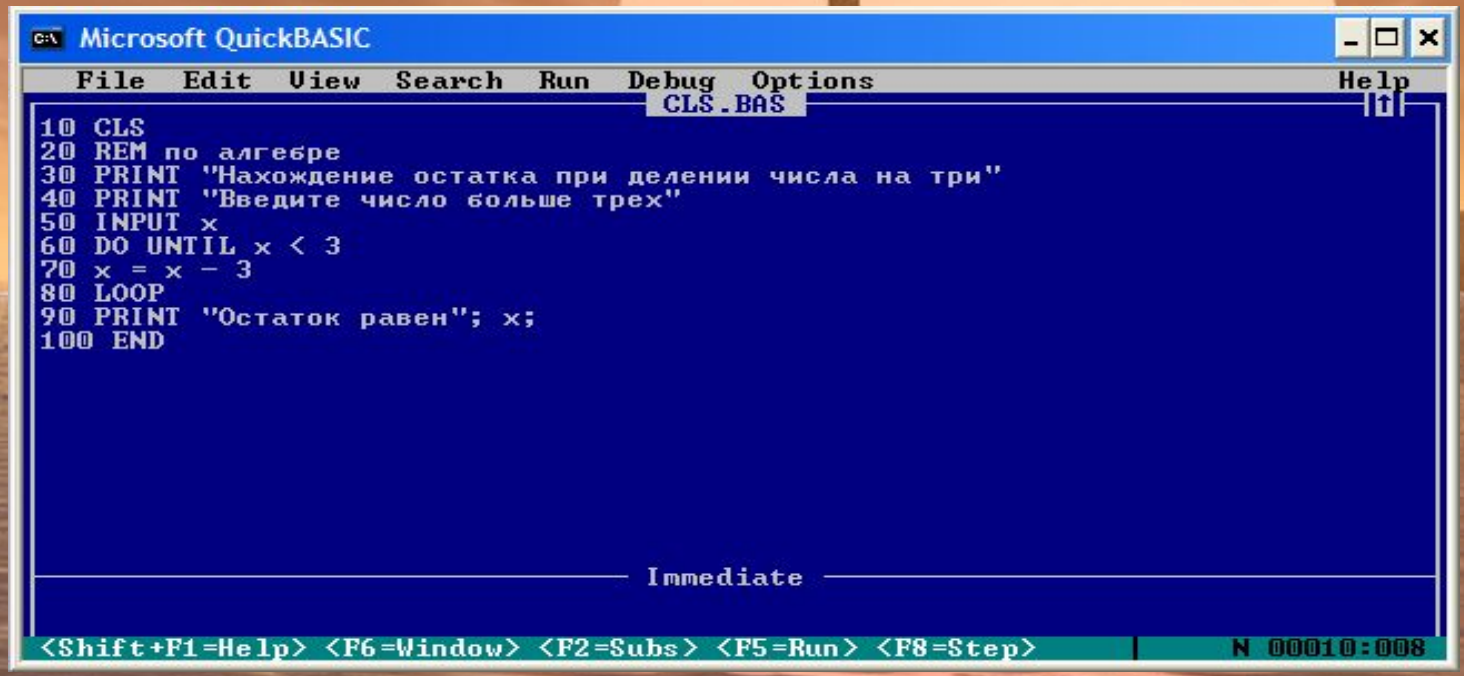
  Здесь: *программа на нахождение остатка при делении какого-либо числа на три.*



Пример циклической программы, реализованной с помощью цикла ДО:

- Используем ключевое слово Until.

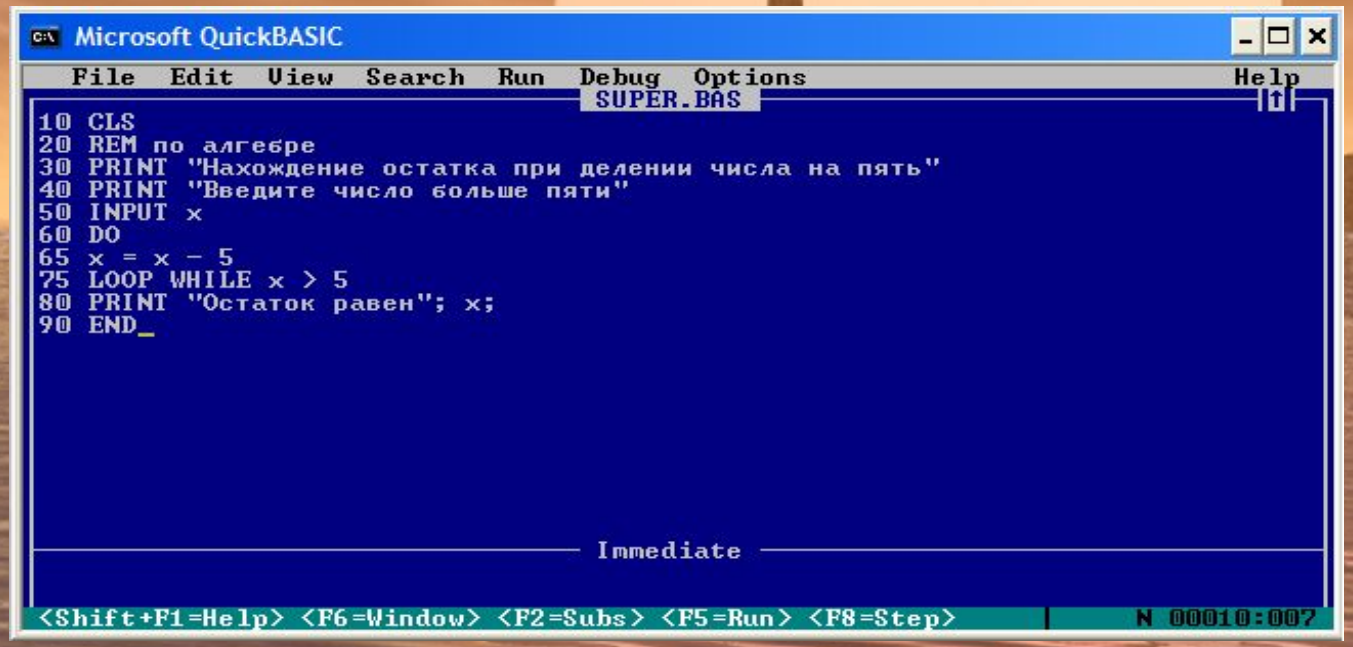
  Здесь: *та же самая задача, что и на предыдущем слайде, но с использованием ключевого слова Until.*



Пример циклической программы, реализованной с помощью цикла ПОКА:

- Используем ключевое слово While.

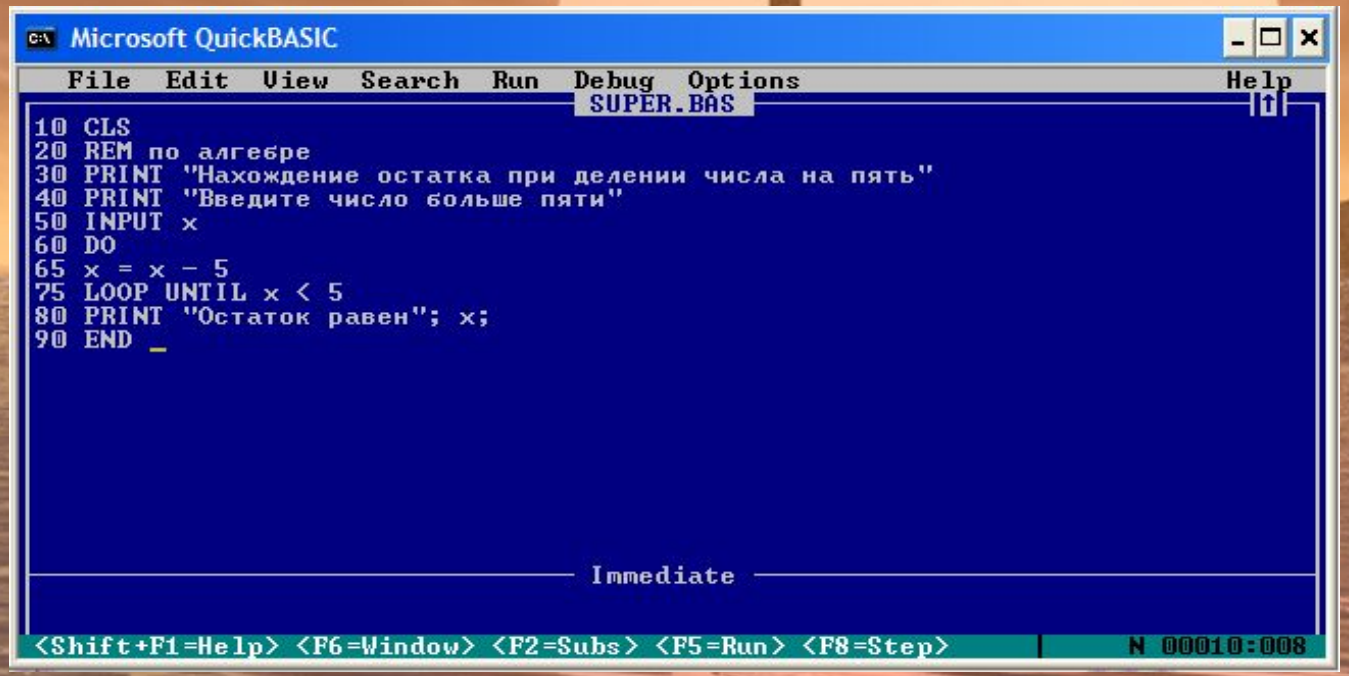
  Здесь: *программа на нахождение остатка при делении какого-либо числа на пять.*



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.*



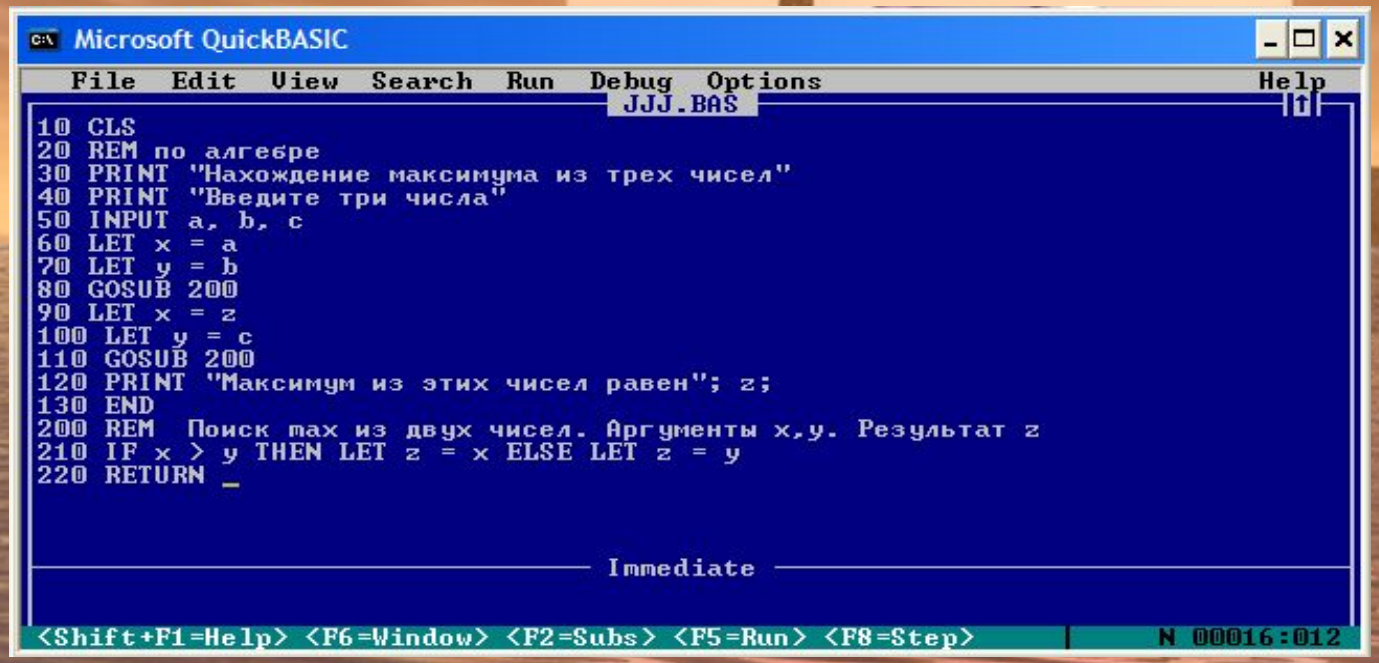
Алгоритмическая структура для программ с подпрограммами

- 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.*



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