Лекция - 1 01.02.2005г

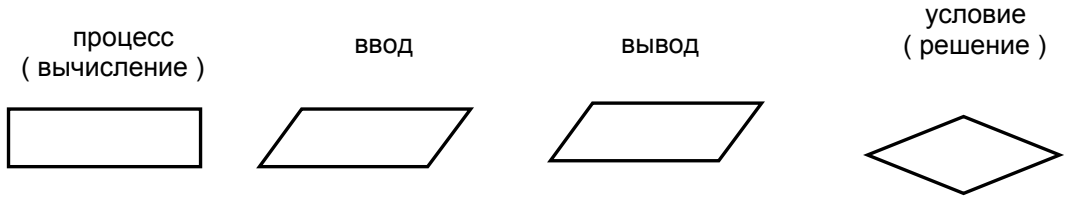
Процесс решения задачи можно разбить на следующие этапы.

1. О Б Ь Е К Т. Постановка задачи. Экспериментальное исследование физико-химического процесса или объекта и определение основных законов управляющих данным объектом или процессом.

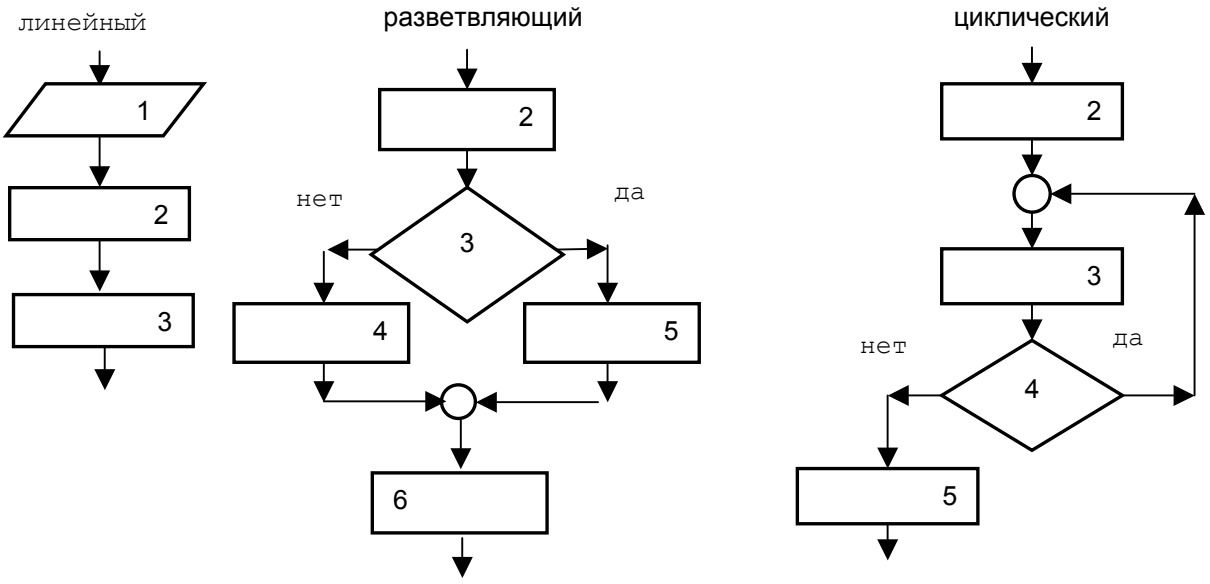
2. М О Д Е Л Ь. Построение математической модели (математическая формулировка задачи) - запись законов описывающих процесс в форме уравнения или системы уравнений (алгебраических, дифференциальных, интегральных и т.д.).

3. А Л Г О Р И Т М. Разработка численного метода и алгоритма (блок-схема). Поскольку ЭВМ не понимает постановки задачи в математической формулировке, то для решения задачи необходимо найти численный метод, позволяющий свести задачу к некоторому вычислительному алгоритму. Алгоритм можно изобразить в виде блок-схемы.

Основные Элементы Блок-схем.

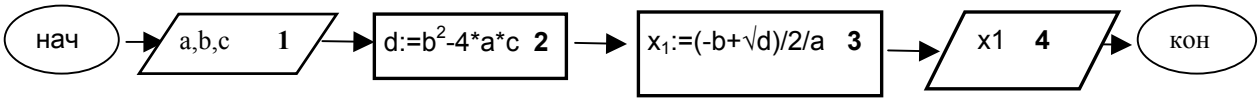


основные типы алгоритмов

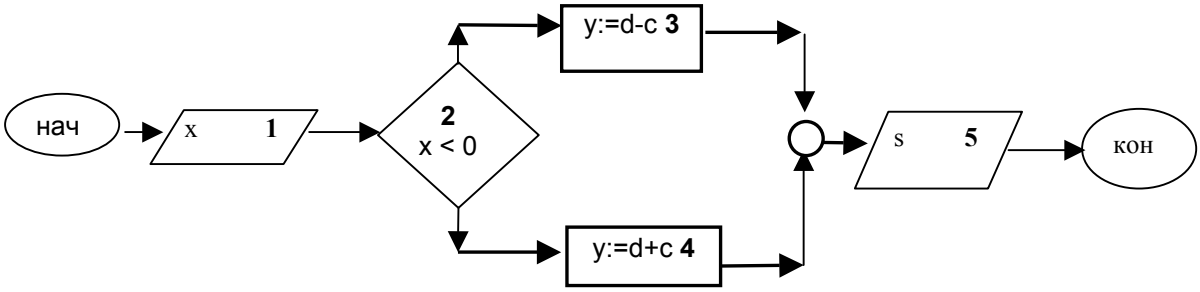


примеры алгоритмов

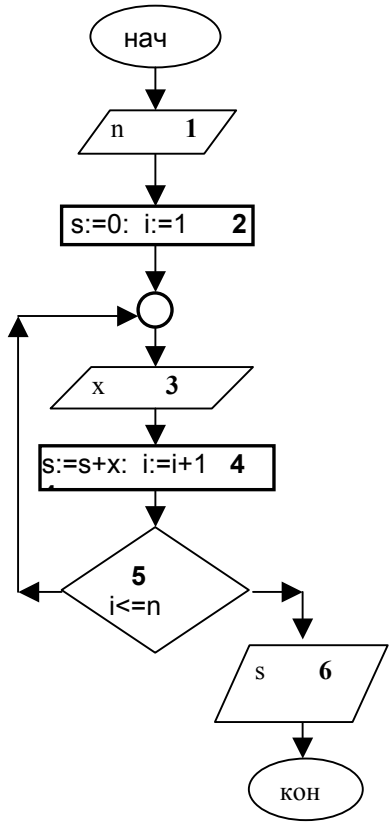
1) вычислить значения переменных $d=b^2-4ac$ и $x_1=(-b+\sqrt{d})/2/a$

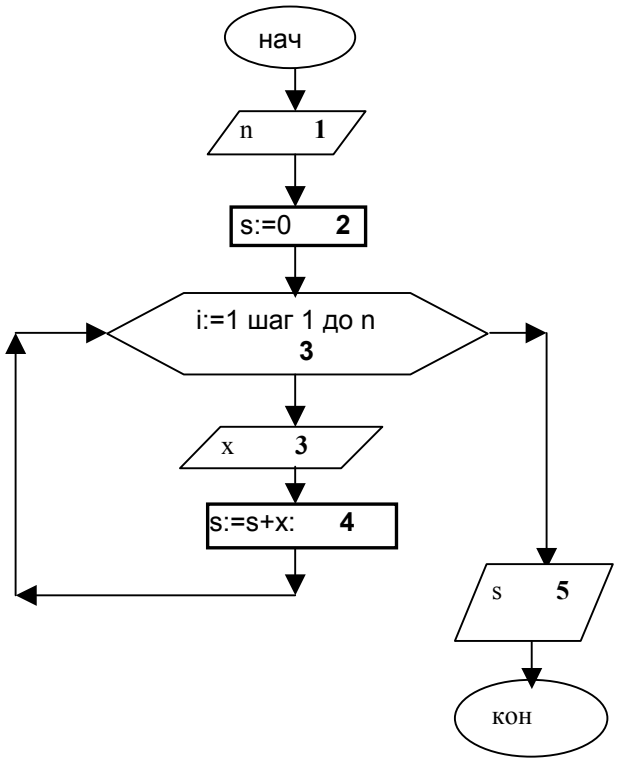


2) вычислить значение переменной $y := \begin{cases} d-c & \text{при } x < 0 \\ d+c & \text{при } x <= 0 \end{cases}$



2) вычислить значение переменной $S=x_1+x_2+x_3+..+x_n = \sum_{i=1}^{n} x_i$





4. П Р О Г Р А М М А

Алгоритм решения задачи записывается на понятном машине языке в виде точно определенной последовательности операций - программы для ЭВМ. Составление программ (программирование) обычно производится с помощью промежуточного (алгоритмического) языка.

5. Проведение вычислений и анализ результатов.

Программирование будем осуществлять на языке программирования VBA (Visual Basic for Applications). Пример программы (блок-1):

```
Private a as single, b as single, c!,d!,x1!
Sub blok1()
a=InputBox(“a=“): b=InputBox(“b=“): c=InputBox(“c=“)
d=b^2-4*a*c
x1=(-b+sqr(d))/2/a
MsgBox d
MsgBox x1
End Sub
```

Программный модуль в VBA состоит из **раздела объявлений** (**declarations**), в котором объявляются переменные и констант с указанием их типа, а так же из **процедур** (**sub**) и **функций** (**function**), в которых могут присутствовать свои объявления переменных и констант.

< раздел объявления >
sub primary()
< группа операторов>
end sub
sub secondary(<параметры>)
<группа операторов>
end sub
function bis(<параметры>) < указание типа значения функции >
<группа операторов>
end function
< описания других процедур и функций >

Переменные, константы и типы данных
Переменные и константы используются для хранения необходимых программных данных, которым даются уникальные имена (идентификаторы). Переменные получают свои значения во время выполнения программы и сохраняют их, пока им не будут присвоены новое значение. Константы применяют в случаях, когда требуется много раз использовать в программе одно и тоже значение. Обычно, перед использованием переменных и констант необходимо произвести их объявление – т.е. заранее указать их имена и типы данных, для которых они предназначены, а для констант и их значения. Основные типы данных:

Тип данных	Описание
Array	Массив переменных, для ссылке на конкретный элемент массива используется индекс. Требуемая память зависит от размера массива.
Boolean	Принимает одно из двух значений; **True** (ИСТИНА) и **FALSE** (ЛОЖЬ). Требуемая память: 2 байта.
Byte	Число без знака от 0 до 255. Требуемая память: 1 байт.
Integer	Короткие целые числовые значения. Диапазон возможных значений: от –32 768 до 32 767. Требуемая память: 2 байта. Символ определения типа по умолчанию: %
Single	Вещественное числовые значения обычной точности. Диапазон возможных значений для отрицательных чисел: от –3.402823E38 до –1.401298E-45, Диапазон возможных значений для положительных чисел: от 1.401298E-45 до 3.402823E38. Требуемая память: 4 байта. Символ определения типа по умолчанию: !

String Используется для хранения строковых значений.
Длина строки: от 0 до 64К байтов.
Требуемая память: 1 байта на символ.
Символ определения типа по умолчанию: $

Variant Может использовать для хранения любых данных. Как нетрудно догадаться, при использовании этого типа данных память расходуется не экономично, и при вычисление с данным типом требует больше времени.

При описании переменной указание типа данных может быть опущено. Тип переменной в таком случаи определяется последним символом в имени переменной: **@, #, %, !, &** или **$**. При отсутствии одного из выше перечисленных символов переменной назначается тип данных **Variant**.

Синтаксис объявления переменной:
Public/Dim/ <имя переменной> [as <имя типа>]
/ - альтернатива
[..] необязательный параметр
Объявление начинается с одного из зарезервированных слов, определяющих область действия (видимости) переменной.
Проект может включать несколько программных модулей. Каждый модуль состоит из процедур и функций. Все процедуры, функции, переменные и константы в VBA имеют свою область действия, которая зависит от того, как и где они объявлены. Имеются три различных уровня при определении области действия переменных - уровень процедуры, уровень модуля и уровень проекта

Dim – используется для объявления переменной область действия которой ограничивается процедурой, в которой она объявлена или модулем, если объявление располагается в разделе объявления модуля, и распространяется на все процедуры и функции только этого модуля.
Public – используется для объявления переменной, область действия которой распространяется на все модули проекта.

В одной строке можно объявить несколько переменных. При этом, правда следует обращать внимание на указание имени типа:
Dim a As Integer, b As Integer, c As Byte
Dim e As Integer, f, g
В первой строке объявлены две переменные типа **Integer** и одна переменная типа **Byte**. Во второй строке - три переменные, причем одна из них получит тип **Integer**, а две другие по умолчанию – тип **Variant**.
Выбирая имя переменной, следует учитывать следующее:
Имя переменной, желательно, должно отражать ее назначение
Имя должно начинаться с буквы
Имя не должно содержать точек
Имя должно быть уникальным, оно не должно совпадать с другими именами или с зарезервированными словами VBA
Имя не может содержать более 255 символов
Чтобы присвоить переменной значение, необходимо выполнить операцию присваивания, где слева от знака равенства находится имя переменной, а справа – не содержащее ошибок арифметическое или логическое выражение.
<имя переменной> = <выражение>
Под выражением может пониматься другая переменная, функция, свойство некоторого объекта, значение (числовое, строковое и т.д.) или результат операций над операндами. Тип операндов и результата должен быть совместим с типом переменной.

Синтаксис объявления массива
Массивом называют набор однотип0ных переменных, объединенных одним именем и доступных, через это имя и значения индексов. Количество индексов определяет **размерность** массива, а совокупность всех значений принимаемых индексами **размером**. Стандартно нижняя граница индекса равно нулю. С помощью оператора **Option Base 1** можно установить нижнюю границу индекса равной 1.
Public/Dim/ <имя массива> ([<список верхних границ индексов>]) [as <имя типа>]
<список верхних границ индексов> - указываются верхние значения индексов по всем измерениям, если границы индексов отсутствуют, то размерность и верхние значения индексов будут объявляться динамически при исполнении программы. Перед первым обращением к массиву, его размерность и размер должны быть определены с помощью оператора **Redim <имя массива>(<размер>**).
Пример
Dim x(10) as single, f(10,10) as double
Public y(10) as single, z() as single
Sub tabul(<список параметров>))
……
k=5
……
Redim z(k)
……
End sub
Объявлены: массив одного измерения **x** состоящий из 11 элементов вещественных чисел обычной точности; массив двух измерений **f** состоящий из 121-ого элемента вещественных чисел двойной точности; массив одного измерения **y** состоящий из 11 элементов вещественных чисел обычной точности; массив **z** как динамический, размер которого определяется в процедуре tabul.
Чтобы присвоить значение элементу массива, необходимо указать имя массива и индексы элемента.
MyArray(9) = 25
F(3,5) = 25.36

Синтаксис объявления константы (одновременно присваивается значение):
[Public] Const <имя константы> [As <имя типа>] = <значение>

Public Const pi as singl = 3.14159
Const text = “ответ”
Преобразование и совместимость типов.
Когда это возможно, преобразование одного типа в другой будет выполнено автоматически и программисту не надо об этом заботится. Следует учитывать следующее:
При преобразовании вещественного числа в целое, дробная часть округляется до ближайшего целого
При преобразовании целого числа в вещественное число, дробная часть будет равна нулю.
При преобразовании целого с большим диапазоном значений в тип с меньшим диапазоном, может возникнуть ошибка при выполнении программы , если значение числа выйдет за предел меньшего диапазона. Строковый тип нельзя преобразовать в числовой и наоборот.
Процедуры и функции.
В программе на VBA используются процедуры следующего вида:
Процедура представляет собой самостоятельную программную единицу с уникальным именем, благодаря которому она может быть вызвана и выполнена. Процедура, такого вида, не возвращает ни какого значения в ту точку, откуда она была вызвана, но ей можно передать параметры.

Процедура-обработчик события отличается от обычной процедуры тем, что вызывается автоматически при наступлении соответствующего события
Процедура-свойство. Используется для создания нового свойства объекта. Обычно необходимо создать пару процедур, которые будут автоматически вызываться каждый раз, когда свойству присваивают значение или обращаются за значением свойства
Синтаксис объявления процедуры
[Public] Sub <имя процедуры>([<список параметров>])
<группа операторов>
End Sub

Функция подобна процедуре, а отличается тем, что возвращает значение в то место, откуда она была вызвана. Поэтому имя функции выступает также в качестве переменной.
Синтаксис объявления функции.
[Public] Function <имя функции>([<список параметров>]) [As <имя типа>]
<группа операторов>
<имя функции>=<выражение>
End Function

Public - для объявления процедуры или функции доступной в пределах всех модулей.
<имя процедуры>/<имя функции> – Имя даваемое процедуре или функции, с помощью которых их можно вызвать.
<имя типа> – определят тип значения возвращаемое функцией.
<список параметров> – имена параметров с указанием их типа
Чтобы процедура или функция была исполнена, она должна быть вызвана.
Вызов процедуры осуществляется вставкой ее имени в текст программы. Сразу после имени должны следовать параметры, если таковые предусмотрены. Другой способ вызова процедуры заключается в использовании ключевого слова Call, при этом за ним должно следовать имя процедуры и заключенный в скобки список параметров.
Синтаксис вызова процедуры.
<имя процедуры> [<список параметров>]
Call <имя процедуры> [(<список параметров>)]
Чтобы функция могла вернуть значение, ее вызов должен фигурировать в правой части оператора присваивания. Арифметические или логические выражения могут также содержать вызовы функций
Синтаксис вызова функции.
<имя переменной>=<имя функции>[(<список параметров>)]
Пример:

```
Public Function My_f(x As Single) As Single
My_f = x ^ 2 - 5
End Function

Sub proc1()
Dim a As Single, b As Single, h As Single
a = (-5): b = 5: h = 0.5
Call проц2(a, b, h)
End Sub

Sub proc2(left As Single, right As Single, step As Single)
Dim i As Integer
Dim x As Single
x = left
```

```
While x <= right
Cells(i + 1, 1) = i
Cells(i + 1, 2) = x
Cells(i + 1, 3) = My_f(x)
i = i + 1: x = x + step
Wend
End Sub
```

Стандартные функции.

Математические функции:

Abs(x) – Возвращает абсолютное значение x.

Sin(x) – Возвращает синус угла x, где x – это угол, заданный в радианах

Tan(x) – Возвращает тангенс угла x, где x – это угол, заданный в радианах

Cos(x) – Возвращает косинус угла x, где x – это угол, заданный в радианах

Exp(x) – Возвращает константу e возведенную в степень x (e=2.71828…)

Int(x) – Возвращает целую часть x (отбрасывает дробную часть числа).

Log(x) – Возвращает натуральный логарифм x.

Rnd(x) – Возвращает случайное число (аргумент x является необязательным)

Sng(x) – Возвращает знак числа : -1, если x отрицательное; 1, если x положительное; 0, если x равно 0.

Sqr(x) – Возвращает корень квадратный из x.

Операции

Математические

<операнд1> + <операнд2> сложение

<операнд1> - <операнд2> вычитание

<операнд1> * <операнд2> умножение

<операнд1> / <операнд2> деление

<операнд1> ^ <операнд2> возведение в степень

<операнд1> **mo**d <операнд2> деление по модулю

<операнд1> \ <операнд2> целочисленное деление

Операции отношения

<операнд1> < <операнд2> меньше. Результат – true, если первый операнд меньше второго

<операнд1> > <операнд2> больше. Результат – true, если первый оператор больше второго.

<операнд1> <= <операнд2> Меньше или равно.

<операнд1> >= <операнд2> больше или равно.

<операнд1> = <операнд2> равно

<операнд1> <> <операнд2> не равно

Логические операции

<операнд1> AND <операнд2> Логическое «И» двух операндов

<операнд1> OR <операнд2> Логическое «ИЛИ» двух операндов

NOT <операнд> Логическое отрицание операнда

Приоритет операций

Результат вычисления значения выражения состоящего из нескольких операций зависит от последовательности их выполнения. В VBA выполняются в соответствии с их приоритетом.

Приоритет операция

вызов функции, скобки

^

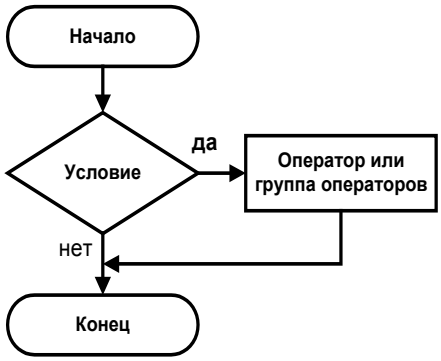
– (с одним операндом)

*,/

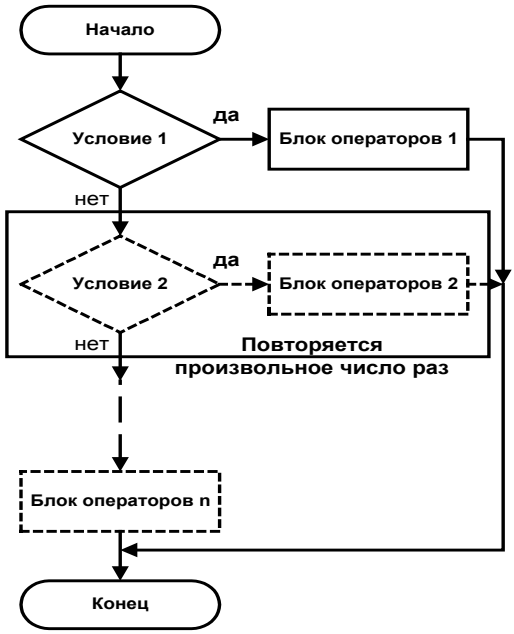
\
MOD
+,-
<,>,<=,>=,=,<>
NOT
AND
OR

Управляющие структуры
Сюда относятся операторы, которые предназначены для управления последовательностью выполнения операторов программы.

Условные операторы (Ветвление по условию)
Под ветвлением подразумевается структура, в которой часть операторов исполняется или нет в зависимости от истинности или нет определенного условия

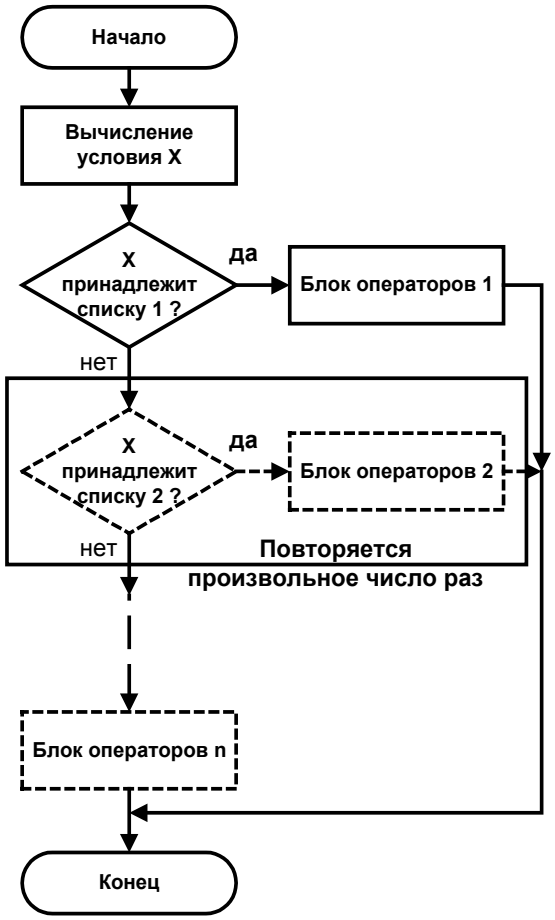


оператор
If <логическое выражение> Then <оператор>
или
If <логическое выражение> Then
<группа операторов>
End if
Если логическое выражение имеет значение True (истинно), то выполняется оператор или группа операторов иначе ни один из операторов не исполняется.
Пример
If f(a)*f(x)<0 then b=x
If f2<f3 then
x4 = x3
x2=x1+(x4-x1)/3
x3=x4-(x4-x1)/3
End if



Оператор
If <логическое выражение 1> Then
<группа операторов>
Else if <логическое выражение 2> Then
<группа операторов>
….
Else
<группа операторов>
End if
пример
if a=5 then
str=”пять”
Elseif a=4 then
Str=4
Elseif a=3 then
Srt=”три”
Else
Str=”прочие”
End if

Оператор
Select Case <анализируемое выражение>
Case <значение1>
<группа операторов>
Case <значение2>
<группа операторов>
……..
Case <значениеN>
<группа операторов>
Case Else
<группа операторов>
End Select



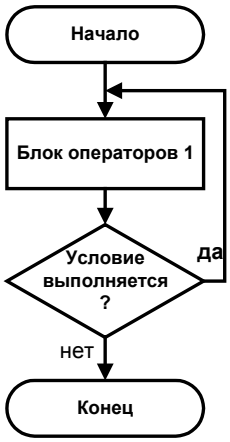
пример

```
Dim Number
Number = 8    ' инициализация переменной.
Select Case Number  ' анализируем
переменную Number.
Case 1 To 5    ' случай Number от 1 до 5
Debug.Print "Между 1 и 5"
Case 6, 7, 8    ' Number от 6 до 8
Debug.Print "Между 6 и 8"
Case 9 To 10  ' Number 9 или 10
Debug.Print "Больше чем 8"
Case Else       ' остальные значения
Debug.Print "Меньше 1 и больше 10"
End Select
```

Операторы цикла

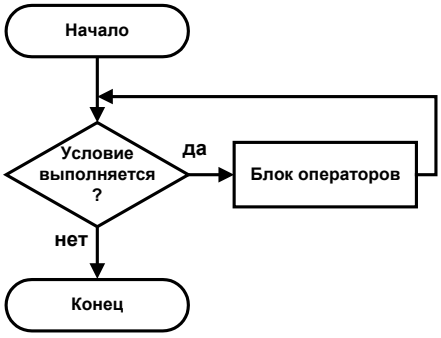
Для многократного выполнение фрагмента программы используются циклы

Оператор
While <условие выполнения>
<группа операторов>
Wend



Пока условие выполнения True (истина) выполняется группа операторов

Оператор
Do
<группа операторов>
Exit do
<группа операторов>
Loop While/Until<условие выполнения>



оператор
Do While/Until<условие выполнения>
<группа операторов>
Exit do
<группа операторов>
Loop

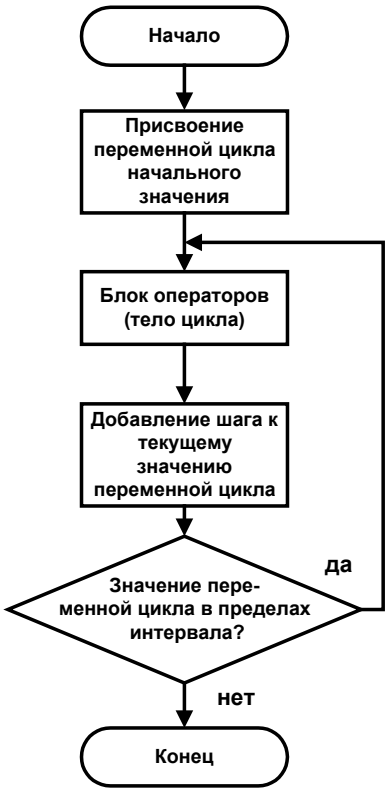
Если используется While, то группа операторов будет выполняться пока условие выполнения True (истина), а если Until, то пока условие выполнения False (ложь).

Пример

```
Sub test41()
Dim a as single, b as single, h as single
Dim x as single
a=-5: b=5: h=0.5
x=a
While x<=b
Debug.print x
Debug.print x^2-5
x=x+h
Whend
End sub
```

```
Sub test42()
 Dim a As Single, b As Single, h As Single
 Dim x As Single
  a = -5: b = 5: h = 0.5
  x = a
  Do
   Debug.Print x
   Debug.Print x ^ 2 - 5
   x = x + h
  Loop While x <= b
End Sub
```

оператор
for <переменная цикла> = <начальное значение> to <конечное значение> step <шаг>
<группа операторов>
next <переменная цикла>
<**переменная цикла**> - ранее объявленная переменная целого типа
<**начальное значение**> и <**конечное значение**> – определяют интервал, в котором будет изменяться переменная цикла с заданным приращением <**шаг**>, если <**шаг**> опушен, то по умолчанию он принимается равным.
Значение <**шаг**>может быть как положительным, так и отрицательным. Если он положителен, параметр <**начальное значение**> должен быть меньше или равен параметру <**конечное значение**>. Если <**шаг**> отрицателен, то <**начальное значение**> должен быть больше или равно <**конечному значению**>, чтобы выполнялось тело цикла <**группа операторов**>, нарушение этих соотношений приводит к невыполнению цикла.
Цикл *For* выполняется *VBA* в следующей последовательности:
1. <**переменная цикла>** устанавливается равной **<начальное значение>**.
2.VBA выполняет тело цикла**<группа операторов>** .
3.VBA увеличивает значение **<переменная цикла>** на 1 или на величину значения **<шаг>**, если он задан.
4.Значение <**переменная цикла>** сравнивается со значением **<конечное значение>**. Если <**переменная цикла>** больше, *VBA* завершает выполнение цикла. Если значение **<шаг>** отрицательно, то *VBA* завершает выполнение цикла при



условии, что значение <**переменная цикла>** меньше значения **<конечное значение>**. Если значение **<шаг>** положительно, то *VBA* завершает выполнение цикла при условии, что значение <**переменная цикла>** больше значения **<конечное значение>**. В случае завершения выполнения цикла *VBA* переходит к выполнению оператора, следующего за оператором *Next.*
VBA возвращается к шагу 2.
В следующем примере цикл For используется для вычисления суммы элементов вектора:

```
Private Sub sumx()
Dim x As Single, n As Integer, s As Single, i As Integer
n=InputBox("Количество чисел")
s=0
For i = 1 To n
x=InputBox("Введите очередное значение x")
s=s+x
Next
MsgBox"значение суммы" & s
End Sub
```

Пример 2

```
Dim I as integer
Dim vect(10) as integer
For I=0 to 10
  Vect(i)=I
Next i
```

Массив Vect примет значения: vect(0)=0; vect(1)=1; vect(2)=2;….vect(9)=9; vect(10)=10