

## Практическая работа № 11.

Тема: **Решение задач в MS Excel с использованием VBA. Организация циклов в программе.**

Цель: Научиться выполнять вычисления с использованием функций и операторов VBA.

Время: 40 мин.

Задание: Решите задачу согласно варианта тремя способами, используя операторы цикла for ... next, do ... loop, while ... wend.

Литература: 1. Симонович С.В. Информатика. Базовый курс, стр. 316 – 339  
2. [Игорь Пашенко. Excel 2007. Шаг за шагом](#)  
3. [Веденева Е.А. Функции и формулы Excel](#)  
4. [Д. М. Златопольский. 1700 заданий по Excel](#)

### Варианты задания:

1. Найти сумму чётных натуральных чисел, меньших 100.
2. Найти сумму нечётных натуральных чисел, меньших 100.
3. Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3.
4. Вычисление  $F = 10!$  выполнить каждым из трёх операторов цикла.
5. Начав тренировки, спортсмен в первый день пробежал 10 км, каждый последующий день он увеличивал дневную норму на 10% от нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?
6. Напечатать (через пробел) все буквы от 'A' до 'Z'.
7. Напечатать (через пробел) все буквы от 'a' до 'z'.
8. Кинетическая энергия движущегося тела  $W_k = mv^2 / 2$ , где  $m$  – масса тела,  $v$  – его скорость.

Составить программу для расчёта зависимости  $W_k$  от  $v$  при значениях  $v$ , изменяющихся от  $v_1$  до  $v_2$  с шагом  $\Delta v$ . Решить задачу при  $v_1 = 10.37$ ,  $v_2 = 15.44$ ,  $\Delta v = 0.23$ ,  $m = 1.004$ .

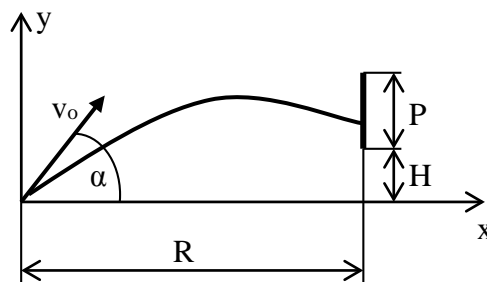
9. Траектория снаряда, вылетающего из орудия под углом  $\alpha$  с начальной скоростью  $v_0$ , описывается уравнениями

$$x = v_0 \cdot \cos \alpha \cdot t,$$

$$y = v_0 \cdot \sin \alpha \cdot t - gt^2 / 2,$$

где  $g = 9.8 \text{ м/с}^2$  – ускорение свободного падения,  $t$  – время. Вводя  $n$  заданных пар  $v_0$  и

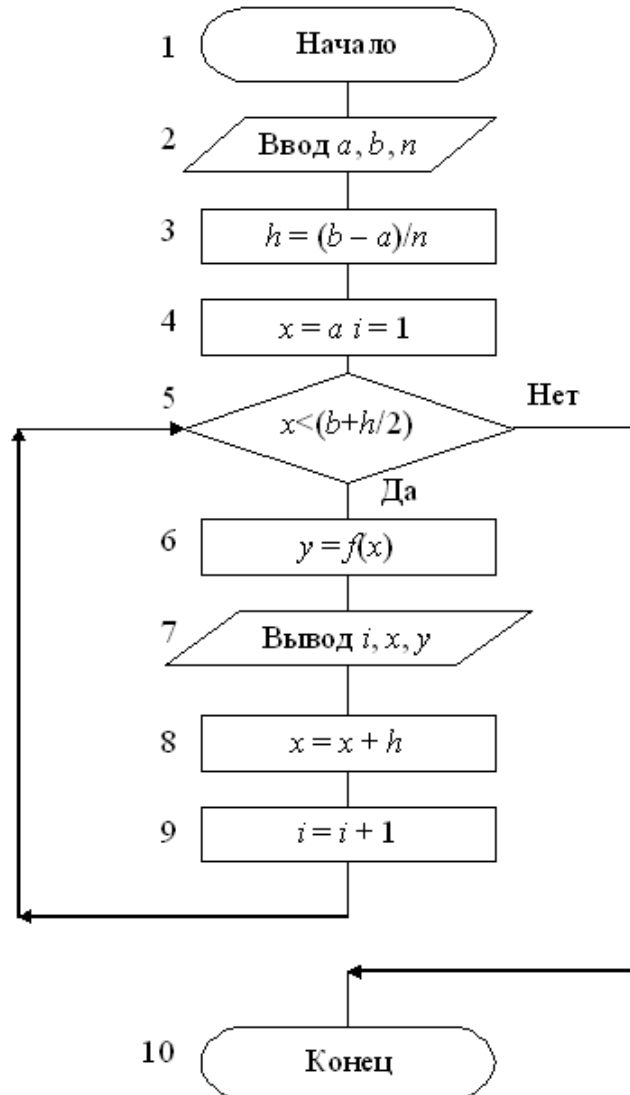
$\alpha$ , определить, сколько снарядов поразит цель высотой  $P$ , расположенную в вертикальной плоскости ствола орудия на расстоянии  $R$  на высоте  $H$  (см. рис.). Задачу решить при  $R = 5400 \text{ м}$ ;  $H = 100 \text{ м}$ ;  $P = 30 \text{ м}$  для пяти пар  $v_0$  и  $\alpha$ :  
180 м/с и 45°; 252 м/с и 30°; 280 м/с и 40°;  
233 м/с и 48°; 260 м/с и 25°.



10. Самолёт летит из пункта А в пункт Б со средней скоростью  $v$ . Найти время полёта  $t$ , если есть встречный ветер, скорость которого  $v_в$  изменяется от 0 до 15 м/с с шагом  $\Delta v_в = 0.5 \text{ м/с}$ . Расстояние между пунктами А и Б  $S = 437 \text{ км}$ ,  $v = 720 \text{ км/ч}$ . (Для тех, кто плохо знает физику:  $t = S / (v - v_в)$ )
11. Составить программу, печатающую таблицу значений градусов температуры по Цельсию и Фаренгейту. Значения градусов температуры по Цельсию изменяются от 0° до 20° с шагом 1°. ( $^{\circ}\text{F} = ^{\circ}\text{C} * 1.8 + 32$ )
12. Пользователь сберегательной кассы внёс в неё вклад  $S_0 = 3000 \text{ грн}$ . До какой суммы он возрастёт через  $N = 5$  лет, если процент годовых начислений  $P = 30 \%$  ?

13. Вскоре после рождения ребёнка родители решили внести в сберкассу вклад  $S_0$  с тем, чтобы через 16 лет, к моменту окончания школы, вклад вырос до 50000 грн. Каким должен быть вклад  $S_0$ , если процент годовых начислений  $P = 30\%$  ?
14. Через сколько лет начальный вклад  $S_0 = 2\,000$  грн. увеличится до суммы  $S = 30\,000$  грн., если процент годовых начислений  $P = 30\%$  ?
15. Вкладчик сберегательной кассы решил ежегодно вносить вклад  $S_I = 500$  грн. в течение  $N = 5$  лет. Сколько денег он получит в конце этого срока, если процент годовых начислений  $P = 30\%$  ?

**Пример блок-схемы алгоритма:**



## Методические указания.

Практически любой алгоритм содержит ряд операторов, которые нужно выполнить несколько раз подряд. Такая операция называется *циклом*. (**Циклом** называется участок программы, который выполняется многократно при различных значениях аргументов.) Операторы, которые выполняются циклически (повторяются), называются *телом цикла*. Цикл может иметь одну или несколько точек входа и обязательно один или несколько выходов. Если цикл не имеет выхода, то алгоритм составлен неправильно.

Для всех операторов цикла характерна следующая особенность: повторяющиеся вычисления записываются всего лишь один раз. Они и называются телом цикла. Вход в цикл возможен только через его начало. Переменные оператора цикла должны быть определены до входа в циклическую часть (т.е. переменным должны быть присвоены какие-либо начальные значения). Не забывайте про условие продолжения цикла. С каждым повторением операторов циклической части (тела цикла) переменная цикла должна увеличиваться (уменьшаться) на заданную величину (шаг). Выход из цикла происходит или по его естественному окончанию, или по оператору перехода *goto <метка>*. Однако применение оператора безусловного перехода является нежелательным, т.к. это нарушает структурную целостность и наглядность программы. Рекомендуется применять инструкцию **Exit**.

Если не задать приращения переменной цикла или не предусмотреть выход из цикла, то циклические вычисления будут повторяться бесконечно, произойдёт «зацикливание» программы.

Циклы широко применяются для решения самых разнообразных задач:

- табулирование функции (нахождение значения функции для аргумента, изменяющегося от начального до конечного значения с заданным шагом);
- нахождение суммы ряда;
- вычисление суммы  $n$  слагаемых;
- вычисление произведения  $n$  сомножителей (вычисление факториала);
- приближённое вычисление определённого интеграла (площади фигуры) и т.д.

Visual Basic имеет три оператора цикла.

### Оператор цикла For.

Наиболее распространенным оператором цикла является оператор FOR.

В общем виде оператор FOR выглядит следующим образом:

```
FOR счетчик = начало TO конец [STEP шаг]  
    тело цикла  
NEXT счетчик
```

где *счетчик* – целочисленная переменная, задающая количество повторений; *начало* – начальное значение переменной; *конец* – конечное значение переменной; *шаг* (необязательный параметр) – шаг приращения. Операторы, содержащиеся в теле цикла, повторяются до тех пор, пока значение счетчика не станет больше или равно значению *конец*.

В приведенном ниже примере инструкция For...Next используется для создания строки, содержащей 10 наборов по 10 цифр (от 0 до 9); каждый набор отделяется от следующего одним пробелом. Внешний цикл использует переменную-счетчик, которая уменьшается на единицу при каждом выполнении цикла.

```
Dim Words, Chars, MyString
```

```
For Words = 10 To 1 Step -1
```

```
    For Chars = 0 To 9
```

```
        MyString = MyString & Chars
```

```
    Next Chars
```

```
    MyString = MyString & " "
```

```
Next Words
```

' Цикл выполняется 10 раз.

' Цикл выполняется 10 раз

' Добавляет цифру в конец строки.

' Увеличивает счетчик

' Добавляет пробел.

## Оператор цикла с предусловием (WHILE).

Часто возникают ситуации, когда заранее неизвестно количество повторений операторов, входящих в тело цикла. В этом случае используют оператор цикла с предусловием WHILE.

В общем виде оператор WHILE выглядит следующим образом:

```
WHILE условие  
    тело цикла  
WEND
```

Операторы, содержащиеся в теле цикла, повторяются до тех пор, пока *условие* не примет значение TRUE (истина). Этот оператор называется «с предусловием», потому что вначале проверяется условие, а после этого выполняются операторы, входящие в тело цикла.

Пример:

В данном примере инструкция While...Wend используется для увеличения переменной-счетчика. Инструкции в цикле выполняются до тех пор, пока указанное условие не True.

```
Dim Counter  
Counter = 0  
While Counter < 20  
    Counter = Counter + 1  
Wend  
Debug.Print Counter
```

*' Инициализирует переменную.*  
*' Анализирует значение счетчика.*  
*' Увеличивает счетчик.*  
*' Завершает цикл While,*  
*' если Counter > 19.*  
*' Выводит 20 в окно отладки.*

## Оператор цикла DO ... LOOP.

Инструкция Do...Loop используется для выполнения наборов инструкций неопределенное число раз. Набор инструкций повторяется, пока условие имеет значение True, либо пока оно не примет значение True.

Синтаксис:

```
Do [{ While | Until } условие]  
    [инструкции]  
[Exit Do]  
    [инструкции]
```

### **Loop**

Допустим также другой синтаксис:

```
Do  
    [инструкции]  
[Exit Do]  
    [инструкции]  
Loop [{ While | Until } условие]
```

Синтаксис инструкции Do Loop содержит следующие элементы:

*Условие* - числовое выражение или строковое выражение, которое имеет значение True или False. Если условие имеет значение Null, то аргумент условие рассматривается как значение False.

*Инструкции* - одна или несколько инструкций, выполнение которых повторяется, пока условие имеет значение True или пока оно не приобретет значение True.

В любом месте управляющей структуры Do...Loop может быть размещено любое число инструкций Exit Do, обеспечивающих альтернативные возможности выхода из цикла Do...Loop. Часто используемая вместе с определением некоторого условия (например,

If...Then), инструкция Exit Do передает управление инструкции, непосредственно следующей за инструкцией Loop.

Во вложенных циклах Do...Loop инструкция Exit Do передает управление циклу охватывающего уровня по отношению к циклу, в котором она вызывается.

### **Пример:**

В данном примере показано, как можно использовать инструкции Do...Loop. Внутренний цикл Do...Loop выполняется 10 раз, затем логической переменной присваивается значение False, после чего он преждевременно завершается с помощью инструкции Exit Do. Внешний цикл завершается немедленно после проверки значения логической переменной.

```
Dim Check, Counter
Check = True: Counter = 0      ' Инициализирует переменные.
Do                             ' Внешний цикл.
    Do While Counter < 20     ' Внутренний цикл.
        Counter = Counter + 1 ' Увеличивает счетчик.
        If Counter = 10 Then  ' Если условие истинно.
            Check = False    ' Присваивает переменной
                              ' значение False.
        Exit Do              ' Завершает внутренний цикл.
    End If
Loop
Loop Until Check = False     ' Немедленно завершает внешний цикл.
```

### **Задача (пример №1).**

Найти сумму первых 15-и натуральных чисел.

Задача сводится к организации цикла по  $i$ . Для циклического накапливания сумм при составлении соответствующих алгоритмов используется предписание стандартного вида:

$$\text{Сумма} = \text{сумма} + \text{слагаемое}$$

Перед началом цикла сумма должна иметь нулевое значение. Понимать эту формулу следует так:

Пусть значение переменной «сумма» хранится в ячейке памяти № 1, а значение переменной «слагаемое» – в ячейке памяти № 2. Все выражения выполняются по правилам приоритета арифметических операций справа налево, т.е. из ячейки памяти № 2 извлекается значение переменной «слагаемое», из ячейки № 1 извлекается значение переменной «сумма», оба числа складываются (за такие операции отвечает процессор, а именно, арифметико – логическое устройство), а результат помещается в ячейку № 1 вместо старого значения переменной «сумма». Таким образом, в ячейке № 1 происходит накопление суммы. Если в качестве слагаемого используется переменная цикла, то с каждой итерацией (шагом) цикла значение этой переменной будет меняться.

Словесная запись этого алгоритма (цикл «До», с постусловием):

1.  $i = 1, S = 0$
2.  $S = S + i$
3.  $i = i + 1$
4. если  $i \leq 15$ , перейти к шагу 2
5. вывести на экран значение  $S$ .
6. конец

Блок-схема алгоритма, соответствующая этой записи, изображена на рис.1. Согласно ГОСТ 19.701-90 схему этого алгоритма можно изобразить так, как на рис. 2.

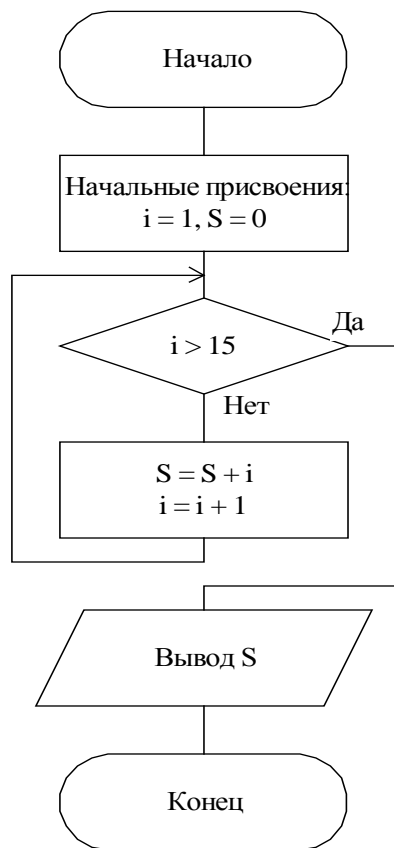


Рис. 1



Рис. 2

Текст программы:

```

Sub example()
  i = 1
  S = 0
  Do
    S = S + i
    i = i + 1
  Loop While i <= 15
  MsgBox (" S = " & S)
End Sub

```

Результат выполнения программы: S = 120.

Для решения этой задачи можно использовать и цикл с предусловием. Словесная запись этого алгоритма (цикл «Пока»):

1.  $i = 1, S = 0$
2. если  $i > 15$ , перейти к шагу 6
3.  $S = S + i$
4.  $i = i + 1$
5. вернуться к шагу 2
6. вывести на экран значение S.
7. конец

Блок-схема алгоритма, соответствующая этой записи, изображена на рис.3. Согласно ГОСТ 19.701-90 схему этого алгоритма можно изобразить так, как на рис. 4.

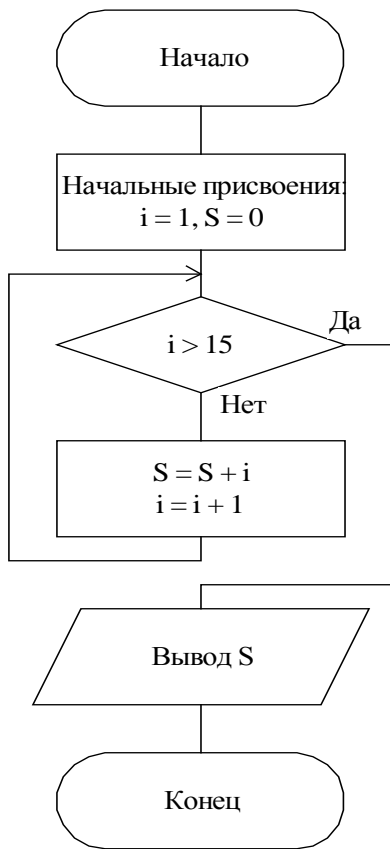


Рис.3

Текст программы:

```

Sub example()
  i = 1
  S = 0
  Do While i <= 15
    S = S + i
    i = i + 1
  Loop
  MsgBox (" S = " & S)
End Sub
  
```

Или:

```

Sub example()
  i = 1
  S = 0
  While i <= 15
    S = S + i
    i = i + 1
  Wend
  MsgBox (" S = " & S)
End Sub
  
```



Рис.4

При использовании цикла с параметром блок-схема алгоритма изображена на рис. 5.

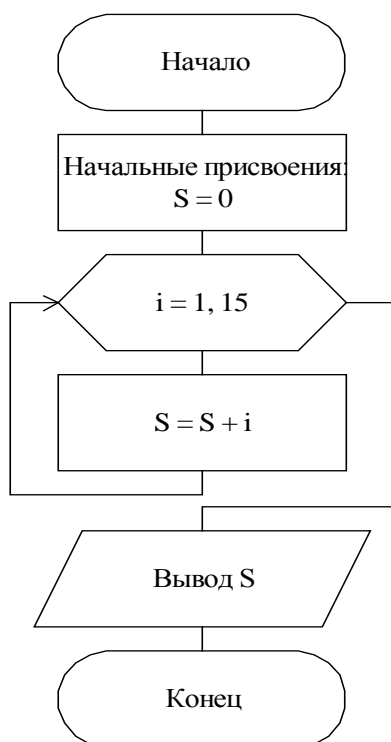


Рис. 5

Текст программы:

```
Sub Auto_Open()  
  Beep  
  a = MsgBox ("Найти сумму натуральных чисел от 0 до 15",vbYesNo,  
  "Задание")  
  if a = vbNo Then Exit Sub  
  S=0  
  For i= 1 To 15  
    S = S + i  
  Next  
  a = MsgBox ("Сумма заданных чисел = " & S, "Ответ")  
  ThisWorkbook.Sheets ("Лист1").Activate  
  Range ("a1").Select  
  i = Len ("Сумма заданных чисел = ")  
  Columns ("A:A").ColumnWidth = i  
  Range ("a1 ").Value = "Сумма заданных чисел = "  
  Range ("b1 ").Value = S  
End Sub
```

В этом примере программа несколько «приукрашена»:

- Процедура названа Auto\_Open, благодаря этому она запускается автоматически при открытии книги;
- В окне сообщения с заголовком «Задание» появляется условие задачи и две кнопки: «Yes» и «No». Если нажимается кнопка «No», задача решаться не будет, последует выход из процедуры. Если нажимается кнопка «Yes», задача решается и в окне сообщения с заголовком «Ответ» появляется ответ:  $S = 120$ .
- Затем открывается 1-й лист рабочей книги, в ячейке A1 появляется надпись «Сумма заданных чисел = », причём ширина столбца A становится равной длине этой надписи;
- В ячейке B1 появляется результат вычислений (120).



## Пример № 2.

В этом примере условие задачи оформлено на листе «Задание» в виде надписи с управляющей кнопкой Start, которая вызывает процедуру, решающую поставленную задачу:

Найти значения функции

$$Y = \frac{1 + \sin x}{1 - \cos x}$$

для всех  $X$ , изменяющихся от  $X_n$  до  $X_k$  с шагом  $dX$ , если

$$X_n = 0$$
$$X_k = 2\pi$$
$$dX = \pi/10$$

Start

Исходные данные заносятся с помощью формы UserForm1:

Текст программы:

Ввод исходных данных

$X_n =$

$X_k =$

$dX =$

OK

*Dim Xn, Xk, dX, X As Variant*

*Sub Auto\_Open()*

*Sheets("задание").Visible = True*

*Sheets("задание").Select*

*End Sub*

*Sub Start()*

*Sheets("задание").Visible = False*

*UserForm1.Show*

*Xn = Val(UserForm1.TextBox1.Value)*

*Xk = Val(UserForm1.TextBox2.Value)*

*dX = Val(UserForm1.TextBox3.Value)*

*Range("A1").Value = "Xn"*

*Range("B1").Value = "Xk"*

*Range("C1").Value = "dX"*

*Range("A1:C4").Select*

*With Selection*

*.HorizontalAlignment = xlCenter*

*.VerticalAlignment = xlBottom*

```

        .WrapText = False
        .Orientation = 0
        .ShrinkToFit = False
        .MergeCells = False
    End With
    With Selection.Font
        .Name = "Arial Cyr"
        .Size = 10
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
    Selection.Font.Bold = True
    Range("A2") = Xn
    Range("B2") = Xk
    Range("C2") = dX
    Range("b4").Select
    Range("b4").Value = "X"
    Range("c4").Value = "Y"
    Columns("C:C").ColumnWidth = 20.86
    i = 4
    For X = Xn To Xk Step dX
        If Abs(Cos(X) - 1) < 0.001 Then
            Y = "Функция не определена"
        Else
            Y = (1 + Sin(X)) / (1 - Cos(X))
        End If
        i = i + 1
        Cells(i, 2).Value = X
        Cells(i, 3).Value = Y
    Next
    Range("b4").Select
    Set tbl = ActiveCell.CurrentRegion
    tbl.Offset(0, 0).Resize(tbl.Rows.Count, tbl.Columns.Count).Select
    With Selection.Borders(xlLeft)
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlRight)
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlTop)
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlBottom)
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    Selection.BorderAround Weight:=xlThin, ColorIndex:=xlAutomatic
    Range("A4").Select
End Sub

Sub Auto_Close()

```

```

Cells.Select
Selection.Clear
Range("A1").Select
End Sub

```

Результат выполнения этой программы:

<b>Xн</b>	<b>Xк</b>	<b>dX</b>
<b>0,00</b>	<b>6,28</b>	<b>0,314</b>

<b>X</b>	<b>Y</b>
0	Функция не определена
0,314	26,76929732
0,628	8,320557201
0,942	4,391966734
1,256	2,825787841
1,57	2,001593605
1,884	1,491735644
2,198	1,140393997
2,512	0,878639707
2,826	0,671778656
3,14	0,500796644
3,454	0,354914388
3,768	0,228580251
4,082	0,120922727
4,396	0,03785636
4,71	0
5,024	0,069452838
5,338	0,457025076
5,652	2,127445718
5,966	13,79442665
6,28	Функция не определена

При закрытии книги лист с решением очищается (процедура Auto\_Close).

### **Применение операторов ветвления.**

Ветвление реализуется с помощью оператора **IF** и инструкции **Select Case**.

#### **Оператор ветвления IF.**

Блочные структуры **If** – наиболее эффективные из структур логического ветвления. Их можно применять для создания логических структур практически любой сложности.

В общем виде эта структура выглядит следующим образом:

```

IF логическое_выражение THEN
    блок операторов
END IF

```

Когда *логическое выражение* принимает значение *истина* (true), то выполняется блок операторов, заключенный между операторами **IF** и **END IF**. Если *логическое выражение* принимает значение *ложь* (false), то этот блок не выполняется и управление переходит к оператору, следующему за оператором **END IF**.

Структура **IF** может содержать оператор **ELSE**:

```

IF логическое_выражение THEN
    блок операторов
ELSE
    блок операторов

```

## END IF

Управление переходит к блоку операторов, следующему за **ELSE** в том случае, если *логическое выражение* принимает значение *ложь* (false).

Это был пример одноблочной структуры оператора **IF**. Одноблочная структура оператора **IF** преобразуется в многоблочную путем добавления оператора **ElseIf**. В этом случае структура выглядит следующим образом:

```
IF логическое_выражение_1 THEN  
    блок операторов 1  
ElseIf логическое_выражение_2 THEN  
    блок операторов 2  
ElseIf логическое_выражение_3 THEN  
    блок операторов 3  
ELSE  
    блок операторов4  
END IF
```

Когда встречается многоблочная структура **IF**, **VB** определяет значение логического выражения 1. Если оно равно true, то выполняется блок операторов 1, если false, то блок операторов 1 полностью пропускается и проверяется значение *логического выражения* 2 и т.д. Если ни одно из поставленных условий не выполняется, то управление передается блоку операторов 4.

В приведенном ниже примере показано использование как блочной, так и однострочной форм инструкции **If...Then...Else**:

```
Dim Number, Digits, MyString  
Number = 53 ' Инициализирует переменную.  
If Number < 10 Then  
    Digits = 1  
ElseIf Number < 100 Then  
    ' Условие является истинным, поэтому выполняется следующая инструкция.  
    Digits = 2  
Else  
    Digits = 3  
End If  
' Использует однострочную форму для присвоения значения.  
If Digits = 1 Then MyString = "Один" Else MyString = "Больше 1"
```

## **Инструкция Select Case.**

Выполняет одну из нескольких групп инструкций в зависимости от значения выражения.  
Синтаксис

```
Select Case выражение  
[Case список Выражений-n  
    [инструкции-n]] ...  
[Case Else  
    [инструкции_else]]  
End Select
```

В приведенном ниже примере инструкция Select Case используется для анализа значения переменной. Второе предложение Case содержит значение анализируемой переменной и следовательно выполняется только инструкция, связанная с этим предложением. Окно отладки выводится на экран командой **[Вид][Окно отладки]** (Ctrl G).

```
Sub num()  
Dim Number  
Number = 8           ' Инициализирует переменную.  
Select Case Number  ' Анализирует число.  
Case 1 To 5         ' Число между 1 и 5.  
    Debug.Print "Между 1 и 5" ' Выводит текст в окно отладки  
Case 6, 7, 8       ' Число между 6 и 8.  
                    ' Это предложение Case является единственным  
                    истинным.  
    Debug.Print "Между 6 и 8"  
Case Is > 8 And Number < 11 ' 9 или 10.  
    Debug.Print "Больше 8"  
Case Else           ' Другие значения.  
    Debug.Print "Вне интервала 1 -- 10"  
End Select  
End Sub
```