Лекция

Подпрограммы

- *Подпрограмма* — программа, реализующая вспомогательный алгоритм.
- *Основная программа* — программа, реализующая основной алгоритм решения задачи и содержащая в себе обращения к подпрограммам.
- В VBA существуют два типа подпрограмм: подпрограммы-функции и подпрограммы-процедуры.
- Отличие функции от процедуры заключается в том, что результатом исполнения операторов, образующих тело функции, всегда является некоторое единственное значение, поэтому обращение к функции можно использовать в соответствующих выражениях наряду с переменными и константами.

- Структурным элементом программы, написанной на языке VBA, является модуль – совокупность объявлений, процедур и функций, объединенных в единое целое.
- Каждый модуль состоит из области описания и одной или нескольких процедур и функций.
- Входящие в модуль процедуры и функции объединены общей областью описания. В ней описываются данные и объекты, которые являются общепринятыми для подпрограмм модуля.

Подпрограмма-процедура имеет следующий синтаксис:

[Private | Public] [Static] Sub <имя процедуры> ([<список аргументов>])

[<Инструкции>]

[Exit Sub]

[<Инструкции>]

End Sub

Public – указывает, что процедура Sub доступна во всех других процедурах во всех модулях;

Private – указывает, что процедура Sub доступна для других процедур только того модуля, в котором она описана;

Static – указывает, что локальные переменные процедуры Sub сохраняются в промежутках времени между вызовами этой процедуры;

Sub, End Sub – служебные слова VBA;

- **<имя процедуры> –** имя процедуры, удовлетворяющее стандартным правилам именования;
- **<список аргументов> –** список переменных, представляющих аргументы, которые передаются в процедуру Sub при её вызове. Имена переменных разделяются запятой;
- **<Инструкции> –** любой набор команд VBA;
- **Exit Sub –** инструкция, выполнение которой приводит к выходу из процедуры.

Синтаксис элемента **<список аргументов>**:

- **[Optional] [ByVal | ByRef] [ParamArray] <Имя переменной> [As <Тип>] [= <По умолчанию>]**
- **Optional –** ключевое слово, указывающее, что аргумент не является обязательным. При использовании этого элемента все последующие аргументы, которые содержатся в списке аргументов, так же должны быть необязательными и описаны с помощью ключевого слова **Optional**. Все аргументы, описанные как **Optional**, должны иметь тип Variant. Не допускается использование ключевого слова **Optional** для любого из аргументов, если используется ключевое слово **ParamArray**;
- **ByVal –** ключевое слово, указывающее, что аргумент передается по значению;
- **ByRef –** ключевое слово, указывающее, что аргумент передается по ссылке. Описание **ByRef** используется в VBA по умолчанию;
- **ParamArray –** ключевое слово, которое может использоваться только в качестве последнего элемента в списке **<список аргументов>** для указания, что конечным элементом является описанный как **Optional** массив значений типа Variant. Ключевое слово **ParamArray** позволяет задавать произвольное количество аргументов, оно не может быть использовано со словами **ByVal, ByRef** или **Optional**;
- **<Имя переменной> –** имя переменной, удовлетворяющее стандартным правилам именования переменных;
- **<Тип> –** тип данных аргумента, переданного в процедуру;
- **<По умолчанию> –** любая константа или выражение, дающее константу. Используется только вместе с параметром **Optional**. Если указан тип Object, единственным значением по умолчанию может быть Nothing.

- В качестве результата процедура может возвращать в вызывающую программу множество простых или структурированных величин или не возвращать никаких значений.
- Среди параметров процедуры указываются как аргументы, так и результаты.
- Обращение к процедуре — отдельный оператор.
- Вызов процедуры из другой процедуры можно произвести несколькими способами.

Первый способ:

<Имя процедуры> <Список фактических параметров>

- **<Имя процедуры> –** имя вызываемой процедуры;
- **<Список фактических параметров> –** список аргументов, передаваемых процедуре; он должен соответствовать списку, заданному в процедуре, по количеству и типу.

Второй способ:

Call <Имя процедуры> (<Список фактических параметров>)

- **Call –** служебное слово VBA;
- **<Имя процедуры> –** имя вызываемой процедуры;
- **<Список фактических параметров> –** список аргументов, передаваемых процедуре; он должен соответствовать списку, заданному в процедуре, по количеству и типу.
- Заметим, что при втором способе вызова процедуры в отличие от первого список фактических параметров должен быть заключен в круглые скобки; в качестве разделителя в списке

- VBA позволяет вводить фактические параметры через имена аргументов в любом порядке и опускать необязательные (Optional).
- При этом после имени аргумента ставится двоеточие и знак равенства, после которого помещается значение аргумента.

Приведенный ниже пример показывает основные способы обращения к процедуре.

```
Option Explicit
Dim c As Double
'c - глобальная переменная
Function F(ByVal x As Integer) As Integer
'Функция, возводящая аргумент в квадрат
F = x ^ 2
End Function
Sub Assistant(ByVal a As Integer, ByVal b As Integer)
'Процедура, находящая сумму двух чисел и
'выводящая результат в диалоговом окне
c = a + b
MsgBox "Сумма: " & c, , "Результат"
End Sub
Sub Main()
'Главная процедура, вызывающая подпрограммы
Dim x As Double, y As Double
'Вызов процедуры с конкретными числами как фактическими параметрами
Assistant 1, 3
'Первоначальное присвоение переменным значений,
'с последующим вызовом процедуры
x = 1
y = 1
Assistant x, y + 2
'Использование функции как фактического параметра
x = 1
y = 3
Assistant F(x), y
'Вызов процедуры с указанием фактических параметров по имени
Assistant b:=3, a:=1
'Вызов процедуры с использованием ключевого слова Call
Call Assistant(x, y)
End Sub
```

Подпрограмма-функция имеет следующий синтаксис:

[Private | Public] [Static] Function
<имя функции>
[(<список аргументов>)]
[<Инструкции>]
[Exit Function]
[<Инструкции>]
End Sub

- Синтаксис инструкции **Function** содержит те же элементы, что и Sub.
  Инструкция **Exit Function** приводит к выходу из функции.
- Тип функции может быть только простым типом.
- Блок содержит локальные для функции описания и раздел операторов.
- Для возврата значения из функции следует присвоить значение имени функции.
- Обращение к функции является операндом в выражении.
- Подпрограмма-функция вызывается в выражении по своему имени, за которым следует вписок аргументов в скобках

- При обращении к подпрограмме происходит *передача* ей *аргументов по ссылке* (если формальный параметр является параметром-переменной, описан как ByRef) или *по значению* (является параметром-значением, описан как ByVal).
- Для того, чтобы понять, в каких случаях использовать тот или иной тип передачи аргументов, рассмотрим, как осуществляется замена формальных параметров на фактические в момент обращения к подпрограмме

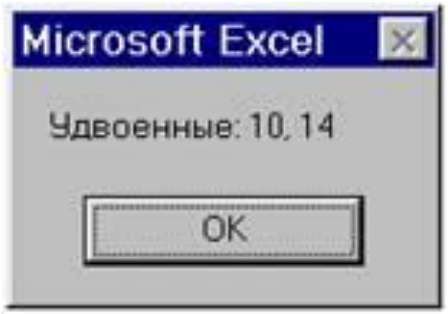
- Если параметр определен как *параметр-значение* (с помощью ключевого слова ByVal), то перед вызовом подпрограммы это значение вычисляется, полученный результат копируется во временную память и передается подпрограмме.
- Важно учесть, что даже если в качестве фактического параметра указано простейшее выражение в виде переменной или константы, все равно подпрограмме будет передана лишь копия переменной (константы).
- Любые возможные изменения в подпрограмме параметра-значения никак не воспринимаются вызывающей подпрограммой, так как в этом случае изменяется копия фактического параметра.

- Если параметр определен как *параметр-переменная* (по умолчанию или с помощью ключевого слова ByRef), то при вызове подпрограммы передается сама переменная, а не ее копия.
- Изменение параметра-переменной приводит к изменению самого фактического параметра в вызывающей подпрограмме.

- Если в качестве фактического параметра используется константа, транслятор блокирует любые присваивания константе нового значения в теле подпрограммы.
- Представленный ниже пример поясняет изложенное. В программе осуществляется ввод целых чисел, их передача процедуре Удвоение.
- Один из параметров параметр-переменная, другой – как параметр-значение.
- Значение параметров до и после вызова процедуры, а так же результат их удвоения выводятся на экран.

```
Option Explicit
Sub Удвоение(ByRef c As Integer, ByVal d As Integer)
c = c + c
d = d + d
MsgBox "Удвоенные: " & c & ", " & d
End Sub
Sub Основная()
Dim a As Integer, b  As Integer
a = Val(InputBox("Введите a", "Ввод", 5))
b = Val(InputBox("Введите b", "Ввод", 7))
MsgBox "Исходные: " & a & ", " & b
Удвоение a, b
MsgBox "Результат: " & a & ", " & b
End Sub
```

В результате прогона программы при сохранении вводимых значений переменных a и b по умолчанию будет выведено:
Как видно из примера, удвоение второго формального параметра в процедуре Удвоение не вызвало изменения фактической переменной b, так как этот параметр описан в заголовке процедуры как параметр-значение.



- Параметры-переменные используются как средство связи алгоритма, реализованного в подпрограмме, с внешним миром: с помощью этих параметров подпрограмма может передавать результаты своей работы вызывающей программе.
- В распоряжении программиста всегда есть и другой способ передачи результатов – через глобальные переменные.
- Однако злоупотребление глобальными связями делает программу, как правило, запутанной, трудной в понимании и сложной в отладке.
- В соответствии с требованиями хорошего стиля программирования рекомендуется там, где это возможно, использовать передачу результатов через фактические параметры-переменные.

- С другой стороны, описание всех формальных параметров как параметров-переменных нежелательно по двум причинам.
- Во-первых, это исключает возможность вызова подпрограммы с фактическими параметрами в виде выражений, что делает программу менее компактной.
- Во-вторых, и главных, в подпрограмме возможно случайное использование формального параметра, например, для временного хранения промежуточного результата, т.е. всегда существует опасность непреднамеренно испортить фактическую переменную.
- Вот почему параметрами-переменными следует объявлять только те, через которые подпрограмма в действительности передает результаты вызывающей программе.
- Чем меньше параметров объявлено параметрами-переменными и чем меньше в подпрограмме используется глобальных переменных, тем проще программа в понимании и отладке.
- По этой же причине не рекомендуется использовать параметры-переменные в заголовке функции: если результатом работы функции не может быть единственное значение, то логичнее использовать процедуру или нужным образом декомпозировать алгоритм на несколько подпрограмм.

1. Вычислить разность двух простых дробей: (a, b, c, d — натуральные числа). Результат получить в виде простой несократимой дроби.

- Следует вычислить числитель и знаменатель по правилам вычитания дробей, и сократить их на наибольший общий делитель (НОД).
- Вычисление НОД двух чисел оформим в виде подпрограммы, используя известный в математике алгоритм Евклида.
- Составим два варианта программы решения этой задачи: с подпрограммой-функцией и подпрограммой-процедурой.

Решение 1

```
Option Explicit
Function НОД(ByVal M As Integer, ByVal N As Integer) As Integer
'Вычисление НОД двух натуральных чисел по алгоритму Евклида
Do While M <> N
If M > N Then
M = M - N
Else
N = N - M
End If
НОД = M
Loop
End Function
Sub Способ1 ( )
Dim A As Integer, B As Integer, C As Integer, D As Integer, E As Integer, F As Integer
A = Val(InputBox("Ведите числитель первой дроби", "Ввод данных"))
B = Val(InputBox("Ведите знаменатель первой дроби", "Ввод данных"))
C = Val(InputBox("Ведите числитель второй дроби", "Ввод данных"))
D = Val(InputBox("Ведите знаменатель второй дроби", "Ввод данных"))
E = A * D - B * C                    'Вычисление числителя
F = B * D                                    'Вычисление знаменателя
If E = 0 Then
MsgBox "Разность дробей" & E, , "Результат"
Else
E = E \ НОД(Abs(E), F)               'Сокращение дроби
F = F \ НОД(Abs(E), F)
MsgBox "Разность дробей: " & E & "/" & F, , "Результат"
End If
End Sub
```

Решение 2

```
Option Explicit
Sub НОД(ByVal M As Integer, ByVal N As Integer, K As Integer)
'Вычисление НОД двух натуральных чисел по алгоритму Евклида
Do While M <> N
If M > N Then
M = M - N
Else
N = N - M
End If
K = M
Loop
End Sub
Sub Способ2 ( )
Dim A As Integer, B As Integer, C As Integer, D As Integer, E As Integer, F As Integer, K As Integer, L
      As Integer
A = Val(InputBox("Ведите числитель первой дроби", "Ввод данных"))
B = Val(InputBox("Ведите знаменатель первой дроби", "Ввод данных"))
C = Val(InputBox("Ведите числитель второй дроби", "Ввод данных"))
D = Val(InputBox("Ведите знаменатель второй дроби", "Ввод данных"))
E = A * D - B * C                              'Вычисление числителя
F = B * D                                                  'Вычисление знаменателя
If E = 0 Then
MsgBox "Разность дробей" & E, , "Результат"
Else
НОД Abs(E), F, L                'Обращение к процедуре
E = E \ L                          'Сокращение дроби
F = F \ L
MsgBox "Разность дробей: " & E & "/" & F, , "Результат"
End If
End Sub
```

2. Составить рекурсивную подпрограмму-функцию вычисления факториала целого положительного числа.

Рекурсивной называется подпрограмма, которая в своем описании содержит обращение к самой себе. Функцию N! рекурсивно можно определить, исходя из следующей формулы:

$$N! = \begin{cases} 1 & \text{при } N = 0; \\ (N-1)! * N & \text{при } N > 0. \end{cases}$$

Описание функции на VBA:

```
Function Factorial(N As Integer) As Long
If N = 0 Then
Factorial = 1
Else
Factorial = Factorial(N - 1) * N
End If
End Function
```

3. В целочисленной матрице размером 10´10 произвести сортировку чисел в строках по возрастанию значений. Первоначально заполнить матрицу целыми случайными числами в диапазоне от 0 до 100.

Для решения задачи составим процедуру Sortirovka сортировки одномерного массива по убыванию значений.
В основной програме используем эту процедуру для сортировки каждой

```
Option Explicit
Sub Sortirovka(X() As Integer, s As String)
Dim i As Integer, j As Integer, Z As Integer, s1 As String
Const N As Integer = 10
ReDim Preserve X(N) As Integer
For i = 1 To N
For j = i + 1 To N
If X(i) < X(j) Then
Z = X(j)
X(j) = X(i)
X(i) = Z
End If
Next
s = s & X(i) & "  "
Next
End Sub

Sub SortMatr()
Dim A() As Integer, B() As Integer
Dim m As Integer, k As Integer, t As Integer
Dim str As String, sort_str As String, s1 As String
Const N = 10
ReDim A(N, N) As Integer
ReDim B(N) As Integer
str = ""
sort_str = ""
'Заполнение матрицы случайными числами и
'формирование строки с элементами матрицы
For k = 1 To N
For m = 1 To N
A(k, m) = Int(100 * Rnd + 1)
str = str & A(k, m) & "  "
B(m) = A(k, m)
Next m
str = str & Chr(13)
Sortirovka X:=B, s:=sort_str    'Обращение к процедуре
sort_str = sort_str & Chr(13)   'Формирование строки с результатом
Next k
MsgBox "Исходный массив" & Chr(13) & str & Chr(13) & "Отсортированный массив" &  _
Chr(13) & sort_str
End Sub
```

Работа с подпрограммами

Visual Basic for Application позволяет осуществлять:

- обращение из одной процедуры в другую;
- передачу параметров из одной процедуры в другую.

Обращение из одной процедуры к другой

Осуществляется в теле основной программы посредством указания имени процедуры, к которой происходит обращение.

Передача параметров из одной процедуры в другую

Для задания конкретных (фактических) значений параметров при обращении к вспомогательной процедуре (подпрограмме) из основной используют оператор **Call**. Его синтаксис:

Call ИмяПроцедуры(фактические параметры)

Описание параметров в скобках после имени процедуры. Имя вспомогательной процедуры (подпрограммы) имеет синтаксис:

ИмяПроцедуры(формальные параметры)

Пример

Создать программу **Главная**, из которой осуществляется обращение к двум вспомогательным процедурам: **Исходная** и **Результат.**

Каждая из вспомогательных процедур должна выводить окно с сообщением, какая программа работает.

Программа Главная

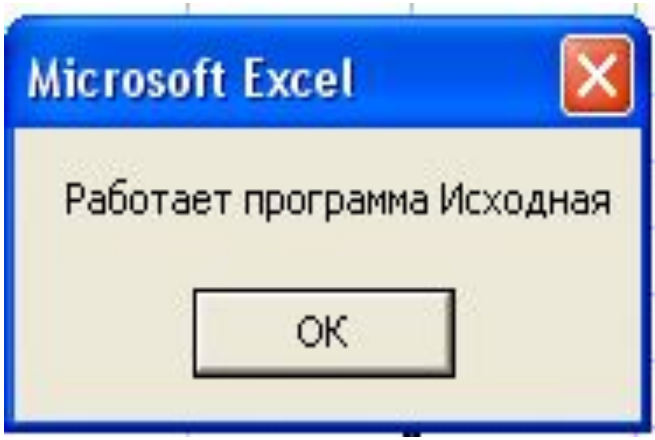
```
Public Sub Главная()
    'Обращение к процедуре Исходная
        Исходная
    'Обращение к процедуре Результат
        Результат
End Sub
```

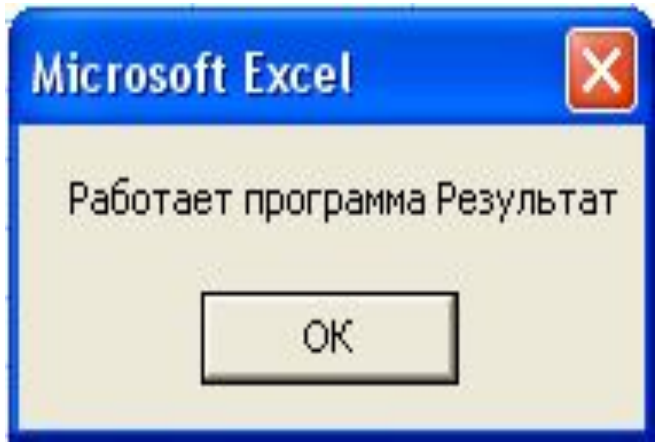
```
'Текст процедуры Исходная
    Sub Исходная()
    MsgBox ("Работает программа Исходная")
    End Sub
```

```
'Текст процедуры Результат
    Sub Результат()
    MsgBox ("Работает программа Результат")
    End Sub
```

При запуске программы (F5) курсор должен находиться в тексте основной программы.

В результате появляется первое диалоговое окно, затем
после нажатия на кнопку Ок, окно закрывается и начинает работать вторая подпрограмма.





Пример

Создать процедуру, вычисляющую площадь круга по значению радиуса R.

Осуществить несколько обращений к этой процедуре, различными способами задавая фактические параметры.

Решение

Вспомогательную процедуру, содержащую алгоритм вычисления площади круга, помещаем внизу основной программы, передающей значения фактических параметров.

```
Public Sub Обращение()
Dim Радиус As Variant

' Задание первого значения радиуса
Радиус = 25
    ' Обращение к программе площади круга
    ' вычисление для радиуса =25
       Call ПлощадьКруга(Радиус)

    ' вычисление для радиуса =18
    Call ПлощадьКруга(18)
End Sub
```

```
    'вспомогательная программа
    Sub ПлощадьКруга(R)
    Dim S  As Variant
    S = 3.14 * R ^ 2
    MsgBox ("Для R = " & R & " Площадь круга равна " & S)
    End Sub
```

Результат работы программы