Лекция 3.

Функции VBA (Часть 1)

РХТУ им. Д.И. Менделеева
Каф. ИКТ
Курс создал: ст. преп. А.М. Васецкий

2013 г

Пользовательские процедуры

Процедура является самостоятельной частью кода, которая имеет имя и может содержать аргументы, выполнять последовательность инструкций и изменять значения своих аргументов.

Синтаксис:

```
[Private | Public][Static] Sub Имя[(СписокАргументов)]
    [Инструкции]
    [Exit Sub]
    [Инструкции]
End Sub
```

Инструкция Exit Sub приводит к немедленному выходу из процедуры

Элементы описания процедуры

Public	Указывает, что процедура Sub доступна для всех других процедур 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
Имя	Имя процедуры Sub, удовлетворяющее стандартным правилам именования переменных
СписокАргументов	Список переменных, представляющий аргументы, которые передаются в процедуру Sub при ее вызове. Имена переменных разделяются запятой
Инструкции	Любая группа инструкций, выполняемых в процедуре Sub

Синтаксис элемента СписокАргументов

[Optional] [ByVal | ByRef] [ParamArray] имяПеременной[( )] [As тип] [= поУмолчанию]

Optional	Ключевое слово, указывающее, что аргумент не является обязательным. При использовании этого элемента все последующие аргументы, которые содержатся в списке СписокАргументов, также 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	Указывает, что этот аргумент передается по значению
ByRef	Указывает, что этот аргумент передается по ссылке. Описание ByRef используется в VBA по умолчанию

Синтаксис элемента СписокАргументов

ParamArray	Используется только в качестве последнего элемента в списке СписокАргументов для указания, что конечным аргументом 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
имяПеременной	Имя переменной, удовлетворяющее стандартным правилам именования переменных
тип	Тип данных аргумента, переданного в процедуру; поддерживаются типы Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String (только строки переменной длины), object, variant. Если отсутствует ключевое слово Optional, могут быть также указаны определяемый пользователем тип или объектный тип
поУмолчанию	Любая константа или выражение, дающее константу. Используется только вместе с параметром optional. Если указан тип object, единственным значением по умолчанию может быть значение Nothing

Примеры

```
Sub Main()
    Calc 99, 43 ‘один из вариантов вызова
    Call Calc(38, 49) ‘второй вариант вызова
End Sub

Sub Calc(x As Single, y As Single)
If x * y < 0 Then
    MsgBox "Корень локализован"
Else
    MsgBox "Корень не локализован"
End If
End Sub
```

ParamArray

```
Sub AnyNumberArgs(strName As String,
ParamArray intScores() As Variant)

Dim intI As Integer
Debug.Print strName; " Scores"
For intI = 0 To UBound(intScores())
    Debug.Print "     "; intScores(intI)
Next intI
End Sub
```

Примеры вызова:

```
AnyNumberArgs "Jamie", 10, 26, 32, 15, 22,
24, 16
AnyNumberArgs "Kelly", "High", "Low",
"Average", "High"
```

Конфликты имён

Если в двух модулях находятся процедуры или функции с одинаковыми именами, то во избежание конфликтов их рекомендуется вызывать с использованием имени модуля:

```
Sub Main()
    Module1.MyProcedure
End Sub
```

Примеры

```
Option Explicit
'Процедура СторонаТреугольника позволяет найти длину недостающей стороны
'прямоугольного треугольника, где переменные A и B отведены под длины катетов,
'а переменная c - под гипотенузу.
'При работе с необязательными переменными необходимо использовать функцию isMissing,
'возвращающую значение True, если соответствующий аргумент не был передан в процедуру,
'и False в противном случае.
```

```
Sub Main()
Dim A1 As Double, B1 As Double, C1 As Double, Triangle(1 To 3) As Double
A1 = 1
B1 = 2
C1 = 3
Triangle(1) = TriangleSide(A1, B1)
Triangle(2) = TriangleSide(A1, , C1)
Triangle(3) = TriangleSide(, B1, C1)

MsgBox Triangle(1) & Chr(13) & Triangle(2) & Chr(13) _
& Triangle(3)

End Sub
```

```
Function TriangleSide(Optional A As Variant, Optional B As Variant, _
Optional C As Variant)
    If Not (IsMissing(A)) And Not (IsMissing(B)) Then
        TriangleSide = Sqr(A ^ 2 + B ^ 2)
    End If

    If Not (IsMissing(A)) And Not (IsMissing(C)) Then
        TriangleSide = Sqr(C ^ 2 - A ^ 2)
End If
    If Not (IsMissing(B)) And Not (IsMissing(C)) Then
    TriangleSide = Sqr(C ^ 2 - B ^ 2)
End If
End Function
```

Пользовательские функции

Синтаксис инструкции Function содержит те же элементы, что и sub. Инструкция Exit Function приводит к немедленному выходу из процедуры Function.

Синтаксис:

```
[Public | Private] [Static] Function Имя [(СписокАргументов)] [As Тип]
        [Инструкции]
        [Имя = Выражение]
        [Exit Function]
        [Инструкции]
        [Имя = Выражение]
End Function
```

Подобно процедуре Sub, процедура Function является самостоятельной процедурой, которая может получать аргументы, выполнять последовательность инструкций и изменять значения своих аргументов. В отличие от процедуры sub, когда требуется использовать возвращаемое функцией значение, Function может применяться в правой части выражения, как и любая другая встроенная функция, например, cos.

Процедура Function вызывается в выражении по своему имени, за которым следует список аргументов в скобках. Для возврата значения из функции следует присвоить значение имени функции. Любое число таких инструкций присвоения может находиться в любом месте процедуры.

Рекомендации по компоновке функции

При вызове функций, помимо значения функции бывает полезно получить какую-либо информацию о ходе вычислений (достигнутая точность, возникшие ошибки и т.п.). Поэтому помимо входных переменных целесообразно заложить в заголовок одну или больше выходных
переменных.

Пример организации такой функции:

```
Function SQRT(X As Double, flErr As Boolean) As Double
'Вычисление квадратного корня (модификация)
flErr = (X < 0) 'возвращаем флаг ошибки
SQRT = 0 'Задаём значение по умолчанию
If Not flErr Then SQRT = Sqr(X)
End Function
```

Передача массивов, как аргументы функции

Синтаксис: (ByVal | ByRef) Arrayname() **As** type

- Arrayname – имя передаваемого массива
- type – представляет любой допустимый тип VBA или определенный пользователем тип.
- Круглые скобки являются обязательными.

Прим. Лучше передавать массивы через ByRef (по ссылке), чтобы не увеличивать объёмы задействованной памяти.

Пример:

```
Function Fun(ByVal A() as Byte) as Boolean
```

Операторы перехода и выбора

GoTo	Оператор безусловного перехода. Синтаксис: GoTo Строка Задает безусловный переход на указанную строку внутри процедуры. Обязательный аргумент строка может быть любой меткой строки или номером строки
If Then Else	Оператор условного перехода. Синтаксис: If Условие Then [Инструкции] [Else Инструкции_else] Если Условие принимает значение True, то выполняется инструкция (или инструкции) после Then, если False, то выполняется инструкция (или инструкции) после Else. Ветвь Else является необязательной. Допускается также использование формы синтаксиса в виде блока: If Условие Then [Инструкции] [Elself Условие-n Then [Инструкции elseif] . . . [Else [Инструкции_else]] End If

Операторы перехода и выбора

Select Case	Оператор выбора. Синтаксис: **Select Case выражение** **[Case списокВыражений-1 [инструкции- 1]]** **…** **[Case списокВыражений-n [инструкции-n]]** **[Case Else [инструкции_else]]** **End Select**
Choose	Возвращает значение, выбранное из списка аргументов. Синтаксис: **Choose(индекс, вариант-1[, вариант-2, ... [, вариант-n]])** индекс – числовое выражение или поле, значением которого является число, лежащее между 1 и числом элементов в списке вариант – выражение типа Variant, содержащее один из элементов списка Действие функции Choose: если индекс равняется 1 , возвращается первый элемент списка, если индекс равняется 2, возвращается второй элемент списка и т. д. Функцию Choose можно использовать для выбора одного из возможных значений, представленных в виде списка

Операторы перехода и выбора

Iif	Возвращает одну из двух альтернатив. Синтаксис: **Iif(expr, truepart, falsepart)** expr – проверяемое выражение truepart – значение или выражение, возвращаемое, если expr имеет значение True falsepart – значение или выражение, возвращаемое, если expr имеет значение False Прим. Побочным эффектом является то, что вне зависимости от результата вычисляются и truepart и falsepart. Иногда это может привести к ошибкам выполнения.
Switch	Возвращается значение, соответствующее первому истинному выражению в списке. Синтаксис: **Switch (выражение-1, значение-1, выражение-2, значение-2 ... [, выражение-n, значение-n]])** выражение – выражение типа Variant, подлежащее вычислению значение – возвращаемое значение или выражение, если соответствующее выражение принимает значение True

IF. Примеры

Использование If в качестве переключателя логической переменной можно избежать.

Обычное решение:

```
Dim fl as Boolean, x as Double
If x>0 then
   fl = true
else
   fl = false
Endif
```

Короткая запись: fl = (x>0)

Примеры

В зависимости от значения **Cityname** возвращает язык

Matchup = Switch(CityName = "London", "English", CityName = "Rome", "Italian", CityName = "Paris", "French")

Если Test>10, то много, иначе, мало

IIf(Test > 10, "много", "мало")

Choose(1, "Верх", "Низ", "Бок")

Вернёт "Верх"

Select Case

```
Select Case sDayOfWeek
Case "Понедельник"
   MsgBox "Пн"
Case "Вторник"
   MsgBox "Вт"
…
Case Else
   MsgBox "Нет такого"
End Select
```

Операторы повтора
For - Next

Синтаксис:

For Счётчик = Начало To Конец [Step Шаг]
[Инструкции]
[Exit For]
[Инструкции]
Next [Счётчик]

Повторяет выполнение группы инструкций, пока Счётчик изменяется от начального значения до конечного с указанным шагом. Если Шаг не указан, то он полагается равным 1.

Альтернативный способ выхода из цикла предоставляет инструкция Exit For

Пример:

```
For i=1 to 10 Step 2
  j=j+1
Next i
```

Операторы повтора
For Each - Next

Синтаксис:

For Each Элемент In Группа
[Инструкции]
[Exit For]
[Инструкции]
Next [Элемент]

```
For Each Object in Collection
' Здесь идет блок кода
Next object
```

```
For i = 1 to Collection.Count
' Здесь идет блок кода
Next i
```

Повторяет выполнение группы инструкций для каждого элемента массива (array) или семейства (collection). Альтернативный способ выхода из цикла предоставляет инструкция Exit For.

Прим. Для коллекций работает быстрее, чем оператор For - Next

Пример

```
Sub TestForEachNextRange()
For Each iCell In Range("A1:C5")
    i = i + 1
    iCell.Value = 10*Int(Rnd*10) & "_" & i
Next
MsgBox "Число ячеек : " & i
End Sub
```

iCell - переменная, которой присваиваются значения элементов группы (массива или семейства) Для работы с элементами массива переменная должна принадлежать к типу Variant.

Пример цикла For Each - Next

80_1	80_2	50_3
90_4	90_5	20_6
60_7	90_8	20_9
50_10	10_11	90_12
60_13	0_14	50_15

Примечание: Пример в презентации сделан с использованием внедрённого объекта MS Excel

Операторы повтора. Do While | Until - Loop

Синтаксис:

Do [While | Until Условие]
[Инструкции]
[Exit Do]
[Инструкции]
Loop

Повторяет выполнение набора инструкций, пока условие имеет значение True | False. Если условие ложно (т.е. False | True соответственно) уже при входе в цикл, то следует передача управления команде, следующей за Loop

Альтернативный способ выхода из цикла предоставляет инструкция Exit Do

Операторы повтора.
Do Loop - While | Until

Синтаксис:

Do

[Инструкции]

[Exit Do]

[Инструкции]

Loop [While | Until Условие]

Повторяет выполнение набора инструкций, пока условие имеет значение True | False.

Цикл выполнится по крайней мере 1 раз.

Альтернативный способ выхода из цикла предоставляет инструкция Exit Do

Пример Цикла Do – Loop While

```
Sub DoWhile()
Const N = 1000
Dim A(N) As String, i As Integer
i = 0
Do
    i = i + 1
    A(i) = InputBox(i & " Value?")
Loop While A(i) > 0
MsgBox "Read " & i & " values"
End Sub
```

Операторы повтора. While - Wend

While Условие
[Инструкции]
Wend

Выполняет последовательность инструкций пока условие истинно.

В отличие от других циклов не имеет альтернативного выхода.

По возможности следует избегать его применения

Встроенные функции VBA

В VBA имеется большой набор встроенных функций и процедур, использование которых существенно упрощает программирование. Эти функции можно разделить на следующие основные категории:

- **Математические функции**
- **Функции проверки типов**
- **Функции преобразования форматов**
- **Функции обработки строк**
- **Функции времени и даты**
- **Прочие функции**

Математические функции

Abs(x)	Модуль числа x
Atn(x)	Арктангенс числа x
Cos(x)	Косинус числа x
Exp(x)	Экспонента числа x
Log(x)	Натуральный логарифм числа x
Sgn(x)	Знак числа x
Sin(x)	Синус числа x
Sqr(x)	Квадратный корень числа x
Tan(x)	Тангенс числа x

Математические функции

RND(x)	Случайное число из интервала [0,1] Если x<0, то функция каждый раз возвращает одно и то же число Если x>0 или отсутствует, то следующее случайное число x=0 случайное число, возвращенное при предыдущем вызове. Функция Randomize(x) используется для инициализации генератора. x здесь базовое число для генерации
Oct(x)	Переводит в восьмеричную систему. Возвращаемый тип - Variant(String)
Hex(x)	Переводит в шестнадцатеричную систему. Возвращаемый тип - String

Математические функции

Round	Округление Round(*expression* [,*numdecplaces*]) *Expression* – выражение *Numdecplaces* – число разрядов до которых ведётся округление
Fix(x) **Int(x)**	Обе функции, Int и Fix, отбрасывают дробную часть числа и возвращают целое значение. Различие между ними состоит в том, что для отрицательного значения аргумента число функция Int возвращает ближайшее отрицательное целое число <= указанному Fix – ближайшее отрицательное целое число >= указанному

Некорректные математические операции

Есть минимум две математические операции, которые VBA, как и многие другие языки выполняют некорректно.

1. Возведение 0 в 0 степень с математической точки зрения не определено. Однако:

```
Sub Pow()
Dim x As Double, i As Integer
i = 0
x = 0 ^ i
MsgBox x
End Sub
```

Некорректные математические операции

2. Корни нечётных степеней от отрицательных чисел в математике определены. Однако, из-за того, что они вычисляются при помощи соответствующих дробных степеней, реализованных, как и остальные степени в машинной арифметике через логарифмы, это вызывает ошибку.

```
Sub SQ()
Dim x As Double
x = -9
x = x ^ (1 / 3) ‘даст ошибку
x = Sgn(x)*Abs(x) ^ (1 / 3) ‘не даст ошибку

MsgBox x
End Sub
```

Производные функции
Тригонометрические функции

Sec(X) = 1/Cos(X)
Cosec(X) = 1/Sin(X)
Cotan(X) = 1/Tan(X)
Arcsin(X) = Atn(X/Sqr(-X * X + 1))
Arccos(X) = Atn(-X/Sqr(-X * X + 1)) +
+ 2 * Atn(1)
Arcsec(X) = Atn(X/Sqr(X * X – 1)) +
Sgn((X) – 1) * (2 * Atn(1))
Arccosec(X) = Atn(X/Sqr(X * X - 1)) +
+ (Sgn(X) – 1) * (2 * Atn(1))
Arccotan(X) = Atn(X) + 2 * Atn(1)

Гиперболические функции

Sinh(X) = (Exp(X) – Exp(-X)) / 2
Cosh(X) = (Exp(X) + Exp(-X)) / 2
Tanh(X) = (Exp(X) – Exp(-X)) / (Exp(X) + Exp(-X))
Sech(X) = 2 / (Exp(X) + Exp(-X))
Cosech(X) = 2 / (Exp(X) – Exp(-X))
Cotanh(X) = (Exp(X) + Exp(-X)) / (Exp(X) – Exp(-X))
Arcsinh(X) = Log(X + Sqr(X * X + 1))
Arccosh(X) = Log(X + Sqr(X * X – 1))
Arctanh(X) = Log((1 + X) / (1 – X)) / 2
Arcsech(X) = Log((Sqr(-X * X + 1) + 1) / X)
Arccosech(X) = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Arccotanh(X) = Log((X + 1) / (X – 1)) / 2
LogN(X) = Log(X) / Log(N) ‘Логарифм по основанию N

Функции даты и времени

Date	Возвращает значение типа Variant (Date), содержащее текущую системную дату
DatePart	Возвращает значение типа variant (Integer) , содержащее указанный компонент даты. Синтаксис: DatePart (interval, date [, firstdayofweek[, firstweekofyear]])
Now	Возвращает значение типа Variant (Date) , содержащее текущую дату и время по системному календарю и часам компьютера

Функции даты и времени

Time	Возвращает значение типа Variant (Date), с текущем временем по системным часам компьютера
Hour, Minute, Second	Возвращают значения типа variant (integer), содержащее целое число, которое представляет часы, минуты и секунды в значении времени. Синтаксис: Hour (время) Minute (время) Second (время) • время – значение времени или выражение, его определяющее
TimeValue	Преобразует строку в формат времени
Timer	Возвращает значение типа Single, представляющее число секунд, прошедших после полуночи.
TimeSerial	Возвращает значение типа Variant (Date), содержащее значение времени, соответствующее указанным часу, минуте и секунде. Синтаксис: TimeSerial (hour, minute, second) Аргументы: hour, minute и second – значения типа Variant (Integer)

Функции даты и времени

DateSerial	Возвращает значение типа Variant (Date), соответствующее указанному году, месяцу и дню. Синтаксис: DateSerial (year, month, day) Аргументы: year, month и day – значения типа Integer
Day, Month, Year	Возвращает значение типа Variant (Integer), содержащее целое число, которое представляет день, месяц, год в значении даты. Синтаксис: Month (дата) Year (дата) дата –значение даты или выражение, её определяющее
Weekday	Возвращает значение типа Variant (integer), содержащее целое число, представляющее день недели. Синтаксис: Weekday (date, [firstdayofweek]) • date – выражение, представляющее дату • firstdayofweek – указывает первый день недели. Если этот аргумент опущен, подразумевается vbSunday (Вс)

Функции даты и времени (продолжение)

DateAdd

Возвращает значение типа Variant (Date), содержащее дату, к которой добавлен указанный временной интервал.
Синтаксис: DateAdd(interval, number, date)
Аргументы:

- interval – строковое выражение, указывающее тип добавляемого временного интервала
- number – числовое выражение, указывающее число временных интервалов, которое следует добавить. Оно может быть положительным (для получения более поздних дат) или отрицательным (для получения более ранних дат).
- date – значение типа Variant (Date) или литерал даты, представляющий дату, к которой добавляется указанный временной интервал

Функции даты и времени (продолжение)

DateDiff

Возвращает значение типа Variant (Long) , указывающее число временных интервалов между двумя датами.

Синтаксис: DateDiff (interval, date1, date2[, firstdayofweek [, firstweekofyear]])

Аргументы:

- Interval – строковое выражение, указывающее тип временного интервала, который следует использовать при вычислении разности между датами date1 и date2. Допустимые значения: yyyy (год), q (квартал), m (месяц), y (день года), d (день месяца), w (день недели), ww (неделя), h (часы), m (минуты), s (секунды)
- date1, date2 – значения типа Variant (Date). Две даты, разность между которыми следует вычислить
- firstdayofweek – постоянная, указывающая первый день недели
- firstweekofyear– постоянная, указывающая первую неделю года

СПАСИБО ЗА ВНИМАНИЕ!