**Основы программирования на VBA**

Для начала в качестве примера рассмотрим простую процедуру VBA типа Sub. Она хранится в модуле VBA и вычисляет сумму первых ста положительных целых чисел. По окончании вычислений процедура отображает сообщение с результатом.[[1]](#footnote-1)

Sub VBA\_Demo()

‘Пример простой процедуры VBA

Dim Total As Long, i As Long

Total = 0

For i = 1 To 100

Total = Total + i

Next i

MsgBox Total

End Sub

В этой процедуре применяются некоторые популярные элементы языка:

* комментарий (строка, начинающаяся апострофом);
* оператор объявления переменной (строка, начинающаяся ключевым словом Dim);
* две переменные (Total и i);
* два оператора присваивания (Total = 0 и Total = Total + i);
* циклическая структура (For–Next);
* функция VBA (MsgBox).

### Комментарии

Вы можете использовать для комментария новую строку либо вставить комментарий после инструкции в той же строке.

В некоторых случаях нужно проверить процедуру, не вставляя в нее инструкцию либо даже целый набор инструкций. Вместо удаления соответствующей инструкции достаточно превратить ее в комментарий путем добавления апострофа в начале строки. После этого VBA проигнорирует инструкцию (или инструкции) при выполнении процедуры. Для преобразования комментария в инструкцию достаточно удалить знак апострофа.

### Переменные, типы данных и константы

*Переменная* представляет собой именованное место хранения данных в памяти компьютера. Переменные могут содержать данные разных типов. Вы можете использовать в названиях буквы, числа и некоторые знаки препинания, но первой в имени переменной всегда должна вводиться буква. VBA не различает регистры в названии переменных. Нельзя использовать в именах пробелы или точки, но можно разделять слова нижним подчеркиванием.

В VBA используется очень много зарезервированных слов, которые не допускается применять в качестве названий переменных или процедур.

Интерпретатор VBA облегчает жизнь программистам, автоматически обрабатывая любые [типы данных](http://on-line-teaching.com/vba/lsn007.html). Однако это чревато негативными последствиями — медленным выполнением операций и менее эффективным использованием памяти. В результате, позволяя VBA самостоятельно определять типы данных, вы можете столкнуться с проблемами выполнения больших или сложных приложений.

Рекомендуется выбирать тот тип данных, в котором используется минимальное количество байтов для хранения значений. Для проведения математических вычислений в рабочих листах Excel использует тип данных Double. Его рекомендуется применять и в процессе обработки чисел в VBA для обеспечения той же точности вычислений.

Если вы не объявили тип данных для переменной, используемой в процедуре VBA, по умолчанию будет задан тип данных Variant. Для определения типа данных переменной используется функция VBA TypeName. Чтобы обеспечить обязательное объявление всех используемых переменных, необходимо включить следующую строку в качестве первой инструкции в модуле VBA: Option Explicit. Чтобы оператор Option Explicit автоматически включался при вставке нового модуля VBA, пройдите в редакторе VBE по меню Tools –> Options и на вкладке Editor выберите параметр Require Variable Declaration (Обязательное объявление переменных; подробнее см. [Настройка среды Visual Basic Editor](http://baguzin.ru/wp/?p=16244)).

Существуют три типы областей действия переменных:



### Работа с константами

Иногда необходимо использовать именованное значение или строку, которая никогда не изменяется, — константу. Если процедура несколько раз ссылается на определенное значение, например, на процентную ставку, следует объявить это значение как константу и использовать в выражениях имя константы, а не ее значение. Такой прием не только делает программу более удобной для восприятия, но и облегчает последующее изменение кода — достаточно изменить только одну инструкцию, а не несколько.

Для объявления констант используется оператор Const. Например,

Const NumQuarters as Integer = 4

Как и переменные, константы имеют область действия. При попытке изменения значения константы в коде VBA вы получите сообщение об ошибке (чего и следовало ожидать). Константа — это постоянное значение, а не переменная. В Excel и VBA существует целый ряд предопределенных констант, которые можно использовать без объявления. Вам даже необязательно знать значение этих констант для их применения. При записи макросов обычно используются константы, а не значения. В следующей процедуре для изменения ориентации страницы активного листа на альбомную применена встроенная константа:

Sub SetToLandscape ()

ActiveSheet.PageSetup.Orientation = xlLandscape

End Sub

Фактическое значение переменной xlLandscape равно 2. Окно Object Browser содержит список всех констант Excel и VBA. Чтобы открыть Object Browser в VBE, нажмите клавишу <F2>.

В VBA дата и время определяются как значения, заключенные между знаками #

Const FirstDay As Date = #1/1/2007#

Const Noon = #12:00:00#

Даты всегда определяются в формате “месяц/день/год”, даже если система настроена на отображение данных в другом формате.

### Операторы присваивания

Оператор присваивания — это инструкция VBA, выполняющая математическое вычисление и присваивающая результат переменной или объекту. В справочной системе Excel выражение определяется как комбинация ключевых слов, операторов, переменных и констант. Эта комбинация возвращает в результате строку, число или объект. Выражение может выполнять вычисление, обрабатывать символы или тестировать данные.

Большое количество операций, выполняемых в VBA, связано с разработкой (и отладкой) выражений. Если вы знаете, как создавать формулы в Excel, то у вас не будет возникать проблем с созданием выражений в VBA. В формуле рабочего листа Excel результат отображается в ячейке. С другой стороны, выражение VBA может присваивать значение переменной или использоваться как значение свойства. В VBA оператором присваивания выступает знак равенства (=).

### Массивы

Массив — это группа элементов одного типа, которые имеют общее имя; на конкретный элемент массива ссылаются, используя имя массива и индекс. Например, можно определить массив из 12 строк так, чтобы каждая переменная соответствовала названию месяца. Если вы назовете массив MonthNames, то можете обратиться к первому элементу массива как MonthNames (0), ко второму — как MonthNames (1) и так до MonthNames (11).

Объявить массив, содержащий ровно 100 целых чисел, можно следующим образом:

Dim MyArray(1 То 100) As Integer

По умолчанию в массивах VBA в качестве первого элемента используется нуль. Если вы хотите, чтобы в качестве первого индекса всех массивов использовалась единица, то перед первой процедурой модуля нужно сделать следующее объявление: Option Base 1

Динамический массив не имеет предопределенного количества элементов. Он объявляется с незаполненными значениями в скобках: Dim MyArray() As Integer. Тем не менее, прежде чем динамический массив можно будет использовать в программе, необходимо обратиться к оператору ReDim, указывающему VBA, сколько элементов находится в массиве. Для этого часто применяется переменная, значение которой неизвестно до тех пор, пока процедура не будет запущена на выполнение. Например, если переменной х присвоено число, размер массива определяется с помощью следующего оператора: ReDim MyArray (1 to х).

### Объектные переменные

*Объектая переменная* — это переменная, представляющая целый объект, например, диапазон или рабочий лист: Dim InputArea As Range. Для присваивания объекта переменной воспользуйтесь ключевым словом Set: Set InputArea = Range("С16:Е16").

### Встроенные функции

В VBA есть ряд встроенных функций, упрощающих вычисления и операции. Например, функция VBA UCase, преобразующая строку в верхний регистр, эквивалентна функции Excel ПРОПИСН. Чтобы использовать функцию Excel в операторе VBA, перед названием функции введите следующее выражение:

Application.WorksheetFunction

Важно понимать, что вы не можете использовать функции Excel, для которых в VBA представлены эквивалентные функции. Например, VBA не позволяет получить доступ к функции Excel КОРЕНЬ (SQRT), так как в VBA имеется собственная версия этой функции: Sqr. Таким образом, следующий оператор выдает ошибку:

MsgBox Application.WorksheetFunction.Sqrt(123)

Функция MsgBox — одна из самых полезных в VBA. Кроме прочего, это превосходный инструмент отладки, поскольку вы можете в любое время вставить функцию MsgBox, чтобы приостановить программу и отобразить результат вычисления или присваивания. Функция MsgBox не только возвращает значение, но и отображает диалоговое окно, в котором пользователь может выполнить определенные действия. Значение, возвращаемое функцией MsgBox, является ответом пользователя на отображенный запрос. Функция MsgBox может применяться даже в том случае, когда ответ пользователя не требуется, а нужно отобразить сообщение. Синтаксис функции MsgBox:

MsgBox(сообщение[, кнопки] [, заголовок] [, файл\_справки, контекст])

* Сообщение (обязательный аргумент) — сообщение, которое отображается в диалоговом окне.
* Кнопки (необязательный аргумент) — значение, определяющее, какие кнопки и пиктограммы (если нужно) отображаются в окне сообщения. Применяйте встроенные константы (например, vbYesNo).
* Заголовок (необязательный аргумент) — текст, который отображается в строке заголовка окна сообщения. По умолчанию отображается текст Microsoft Excel.
* Файл\_справки (необязательный аргумент) — название файла справки, соответствующего окну сообщения.
* Контекст (необязательный аргумент) — контекстный идентификатор раздела справки. Представляет конкретный раздел справки для отображения. Если используется аргумент контекст, следует также задействовать аргумент файл\_справки.

Вы можете присвоить полученное значение переменной либо использовать функцию без оператора присваивания. В приведенном ниже примере результат присваивается переменной Ans.

Ans = MsgBox("Продолжить?", vbYesNo + vbQuestion, "Сообщи")

If Ans = vbNo Then Exit Sub

Обратите внимание, что в качестве значения аргумента кнопки используется сумма двух встроенных констант (vbYesNo + vbQuestion). Благодаря константе vbYesNo в окне сообщения отображаются две кнопки: одна с меткой Yes, а вторая — с меткой No. Добавление vbQuestion в состав аргумента также приведет к отображению значка вопроса. Как только будет выполнен первый оператор, переменная Ans получит одно из двух значений, представленных константами vbYes и vbNo. В этом примере процедура завершает свою работу после щелчка на кнопке No.

### Управление объектами и коллекциями

VBA предлагает две конструкции, которые помогут вам упростить управление объектами и коллекциями. Конструкция With - End With позволяет выполнять несколько операций над одним объектом. Чтобы понять, как она работает, проанализируйте следующую процедуру, которая изменяет шесть свойств выделенного объекта (подразумевается, что выделен объект Range).

Sub ChangeFontl()

Selection.Font.Name = "Cambria"

Selection.Font.Bold = True Selection.Font.Italic = True

Selection.Font.Size = 12

Selection.Font.Underline = xlUnderlineStyleSingle

Selection.Font.ThemeColor = xlThemeColorAccentl

End Sub

Эту процедуру можно переписать с помощью конструкции With - End With. Процедура, показанная ниже, работает точно так же, как и предыдущая.

Sub ChangeFont2 ()

With Selection.Font

.Name = "Cambria"

.Bold = True

.Italic = True

.Size = 12

.Underline = xlUnderlineStyleSingle

.ThemeColor = xlThemeColorAccentl

End With

End Sub

Предположим, вы решили выполнить действие над всеми объектами коллекции или вам необходимо оценить все объекты коллекции и совершить действие при выполнении определенных условий. Это идеальная ситуация для применения конструкции For Each - Next. Синтаксис конструкции:

For Each элемент In коллекция

[инструкции]

[Exit For]

[инструкции]

Next [элемент]

Например:

Sub CountSheets()

Dim Item as Worksheet

For Each Item In ActiveWorkbook.Worksheets

MsgBox Item.Name

Next Item

End Sub

В следующем примере закрываются все окна, за исключением активного:

Sub Closelnactive()

Dim Book as Workbook

For Each Book In Workbooks

If Book.Name <> ActiveWorkbook.Name Then Book.Close

Next Book

End Sub

### Контроль за выполнением кода

Некоторые процедуры VBA начинают выполняться с первых строк кода. Однако иногда необходимо контролировать последовательность операций, пропуская отдельные операторы, повторно выполняя некоторые команды и проверяя условия для определения следующего действия, выполняемого процедурой.

Оператор GoTo перенаправляет ход выполнения программы на новую инструкцию, которая помечена специальным образом (текстовая строка, заканчивающаяся двоеточием, или число, заканчивающееся пробелом, указанные перед инструкцией). В приведенной ниже процедуре применена функция VBA InputBox для получения имени пользователя. Если имя пользователя отличается от Ховард, то процедура переходит к метке WrongName, на чем заканчивает свою работу. В противном случае процедура выполняет дополнительные операции. Оператор Exit Sub заканчивает выполнение процедуры.

Sub GoToDemo()

UserName = InputBox("Введите свое имя:")

If UserName <> "Ховард" Then GoTo WrongName

MsgBox ("Привет, Ховард...")

‘ - [Здесь вводится дополнительный код] -

Exit Sub

WrongName:

MsgBox "Извините, эту процедуру может запускать только Ховард."

End Sub

В действительности оператор GoTo необходим только для перехват ошибок (код выше является неудачным примером, который не следует использовать).

Вероятно, конструкция If-Then чаще остальных используется для группирования инструкций VBA:

If *условие* Then *инструкции\_истина* [Else *инструкции\_ложь*]

Например

Sub GreetMe()

If Time < 0.5 Then

MsgBox "Доброе утро"

Elself Time >= 0.5 And Time < 0.75 Then

MsgBox "Добрый день"

Else

MsgBox "Добрый вечер"

End If

End Sub

VBA использует систему дат и времени, похожую на задействованную в Excel. Время дня выражается дробным числом, например, полдень представлен как 0.5. Значение 0.75 представляет время 18:00 — три четверти суток и тот момент, когда день переходит в вечер. Вложенные структуры If-Then достаточно громоздкие. Поэтому рекомендуется использовать их только для принятия простых бинарных решений. Если же необходимо выбрать между тремя и более вариантами, то целесообразно обратиться к конструкции Select Case.

В следующей процедуре используется функция VBA WeekDay, с помощью которой определяется, является ли текущий день субботой либо воскресеньем (функция Weekday возвращает значение 1 либо 7). Затем отображается соответствующее сообщение.

Sub GreetUserlO

Select Case Weekday(Now)

Case 1, 7

MsgBox "Это выходные"

Case Else

MsgBox "Это не выходные"

End Select

End Sub

Интерпретатор VBA осуществляет выход из конструкции Select Case, как только найдено условие True. Следовательно, для максимальной эффективности, в первую очередь, следует выполнить проверку наиболее вероятного случая.

*Цикл* — это процесс повторения набора инструкций. Возможно, вы заранее знаете, сколько раз должен повториться цикл, или это значение определяется переменными в программе. Простейший пример хорошего цикла — For-Next:

For *счетчик* = *начало* То *конец* [Step *шаг*]

[*инструкции*]

[Exit For]

[*инструкции*]

Next [*счетчик*]

Следующая процедура суммирует квадратные корни первых 100 целых чисел:

Sub SumSquareRoots()

Dim Sum As Double

Dim Count As Integer

Sum = 0

For Count = 1 To 100

Sum = Sum + Sqr(Count)

Next Count

MsgBox Sum

End Sub

Значение переменной Step в цикле For-Next может быть отрицательным. Приведенная ниже процедура удаляет строки 2, 4, 6, 8 и 10 в активном листе:

Sub DeleteRows ()

Dim RowNum As Long

For RowNum = 10 To 2 Step -2

Rows(RowNum).Delete

Next RowNum

End Sub

Циклы For-Next могут также содержать один или более операторов Exit For. Когда программа встречает этот оператор, то сразу же выходит из цикла:

Sub ExitForDemo()

Dim MaxVal As Double

Dim Row As Long

MaxVal = Application.WorksheetFunction.Max(Range("A:A"))

For Row = 1 To 1048576

If Cells(Row, 1).Value = MaxVal Then

Exit For

End If

Next Row

MsgBox "Максимальное значение в строке " & Row

Cells(Row, 1).Activate

End Sub

Максимальное значение в столбце вычисляется с помощью функции Excel МАХ. Затем это значение присваивается переменной MaxVal. Цикл For-Next проверяет каждую ячейку в столбце. Если определенная ячейка равна MaxVal, оператор Exit For заканчивает процедуру. Однако перед выходом из цикла процедура сообщает пользователю о расположении искомой ячейки и активизирует ее.

Цикл Do While выполняется до тех пор, пока удовлетворяется заданное условие. Цикл Do While может иметь один из двух представленных ниже синтаксисов.

Do [While *условие*]

[*инструкции*]

[Exit Do]

[*инструкции*]

Loop

или

Do

[*инструкции*]

[Exit Do]

[*инструкции*]

Loop [While *условие*]

Процедура EnterDates1 вводит даты текущего месяца в столбец рабочего листа, начиная с активной ячейки:

Sub EnterDatesl ()

‘ цикл Do While, условие проверяется в начале

Dim TheDate As Date

TheDate = DateSerial(Year(Date), Month(Date), 1)

Do While Month(TheDate) = Month(Date)

ActiveCell = TheDate

TheDate = TheDate + 1

ActiveCell.Offset(1, 0).Activate

Loop

End Sub

В этой процедуре используется переменная TheDate, которая хранит даты, записанные в рабочем листе. Для инициализации переменной используется первый день текущего месяца. В процессе выполнения цикла значение переменной TheDate было введено в активную ячейку, затем это значение было увеличено на единицу, после чего активизируется следующая ячейка. Цикл выполняется до тех пор, пока значение месяца, присвоенное переменной TheDate, совпадет со значением месяца текущей даты.

Циклы Do While также могут включать один или более операторов Exit Do. По достижении оператора Exit Do цикл завершается, а управление передается оператору, следующему за оператором Loop.

Структура цикла Do Until имеет много общего с конструкцией Do While. Разница заключается лишь в том, как проверяется условие цикла. В варианте Do While цикл выполняется до тех пор, пока выполняется условие. В цикле Do Until цикл выполняется, пока условие не станет выполняться. Структура Do Until также может быть представлена двумя видами синтаксиса.

1. По материалам книги [Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA](http://baguzin.ru/wp/?p=16255). – М: Диалектика, 2013. – С. 211–251. [↑](#footnote-ref-1)