Методы программирования на VBA

Настоящая заметка продолжает знакомство с VBA, она иллюстрируют часто используемые приемы VBA, которые можно применять в собственных проектах.[1]

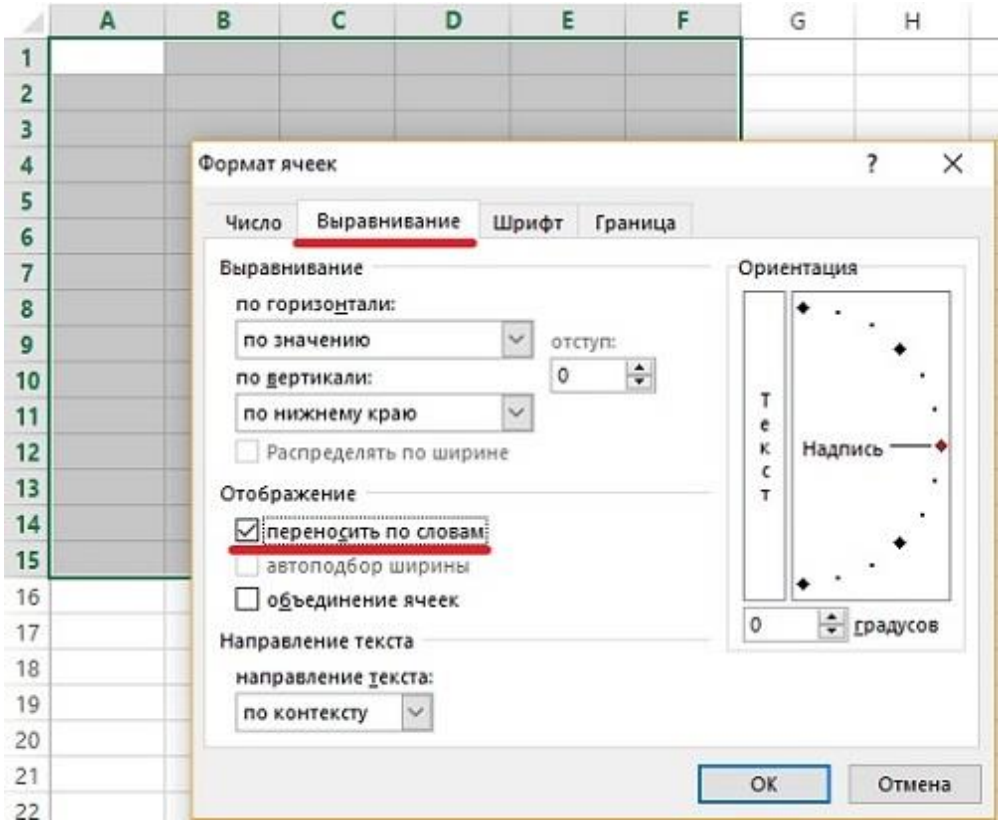

Рис. 1. Включение опции *Перенос по словам*

Переключение значения булева свойства

Булево свойство — это логическое свойство, принимающее одно из двух значений: True (ИСТИНА) или False (ЛОЖЬ). Самый простой способ изменить булево свойство — использовать оператор `Not`. В следующем примере изменяется свойство переноса по словам `WrapText` в выделенном диапазоне ячеек.

```
Sub ToggleWrapText()
' Управляет переносом слов в выделенных ячейках
    If TypeName(Selection) = "Range" Then
        Selection.WrapText = Not ActiveCell.WrapText
    End If
End Sub
```

Обратите внимание, что за основу взята активная ячейка. Когда диапазон выделен и значения свойств в разных ячейках неодинаковы (например, в одних ячейках шрифт полужирный, а в других — нет), то диапазон считается смешанным, и Excel использует в качестве базового значение свойства активной ячейки. Если, например, активная ячейка имеет полужирный шрифт, то начертание текста в выделенных ячейках после щелчка на кнопке *Полужирный* панели инструментов будет обычным. Эта простая процедура имитирует поведение элемента интерфейса Excel. Процедура использует функцию `TypeName`, чтобы проверить, является ли выделенный объект диапазоном. Если это не так, то ничего не происходит.

Выделите диапазон A1:F15 (рис. 1), кликните правой кнопкой мыши, выберите *Формат ячеек*, перейдите на закладку *Выравнивание*, и поставьте галочку *переносить по словам*, нажмите *ОК*. Запустите процедуру `ToggleWrapText()` из меню, или нажав Ctrl+Shift+W. Повторно откройте окно *Формат ячеек*, и убедитесь, что галочка снята.

---

[1] По материалам книги Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA. – М: Диалектика, 2013. – С. 352–358.

Определение количества страниц для печати

Если нужно определить количество страниц для печати, можно использовать команду Excel *Предварительный просмотр*, а затем подсчитать количество отображающихся на экране страниц. Этот процесс поддается автоматизации с помощью следующей процедуры VBA, которая вычисляет количество страниц для печати на активном листе путем подсчета горизонтальных и вертикальных разрывов страницы (рис. 2).

```
Sub PageCount()
MsgBox (ActiveSheet.HPageBreaks.Count + 1) * _
     ActiveSheet.VPageBreaks.Count + 1) & " страниц"
End Sub
```

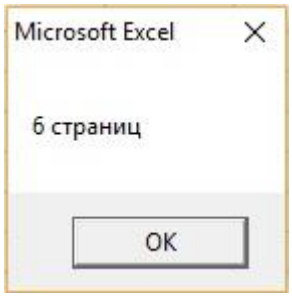

Рис. 2. Процедура VBA подсчитывает число печатных страниц на активном листе

Следующая процедура циклически просматривает все листы в активной рабочей книге и отображает общее количество страниц для печати.[2]

```
Sub ShowPageCount()
    Dim PageCount As Integer
    Dim sht As Worksheet
    PageCount = 0
    For Each sht In Worksheets
       PageCount = PageCount + (sht.HPageBreaks.Count + 1) * _
        (sht.VPageBreaks.Count + 1)
    Next sht
    MsgBox "Число страниц = " & PageCount
End Sub
```

Отображение даты и времени

Процедура DateAndTime отображает окно сообщения с текущими датой и временем (рис. 3). В строке заголовка окна сообщения представлен пользовательский текст. Процедура использует в качестве аргумента функции `Format` функцию `Date`. В результате строка с датой будет представлена в удобном для восприятия формате. Тот же прием применяется для задания формата времени.

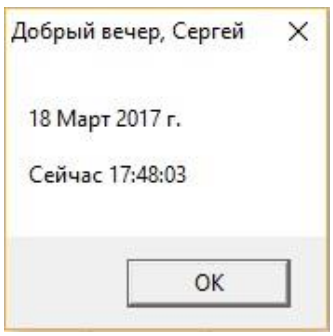

Рис. 3. Приветствие, текущие дата и время

```
Sub DateAndTime()
'   Отображение текущей даты и времени
    Dim TheDate As String, TheTime As String
```

[2] Обе процедуры дают неверное число листов печати, если граница в точности проходит по последней строке/столбцу диапазона.

```
    Dim Greeting As String
    Dim FullName As String, FirstName As String
    Dim SpaceInName As Long
    TheDate = Format(Date, "Long Date")
    TheTime = Format(Time, "Long Time")
'   Формирование строки приветствия на основе текущего времени
    Select Case Time
        Case Is < 0.5:     Greeting = "Доброе утро, "
        Case Is >= 0.7083: Greeting = "Добрый вечер, "
        Case Else:         Greeting = "Добрый день, "
    End Select
'   Включение в приветствие имени пользователя
    FullName = Application.UserName
    SpaceInName = InStr(1, FullName, " ", 1)
'   Обработка ситуации, когда имя не включает пробелы
    If SpaceInName = 0 Then SpaceInName = Len(FullName)
    FirstName = Left(FullName, SpaceInName)
    Greeting = Greeting & FirstName
'   Отображение сообщения
    MsgBox TheDate & vbCrLf & vbCrLf & _
     "Сейчас " & TheTime, vbOKOnly, Greeting
End Sub
```

В данном примере использованы именованные форматы ("Long Date" и "Long Time") для обеспечения работоспособности макроса независимо от региональных настроек компьютера пользователя. Однако вы можете обратиться к другим форматам. Например, чтобы отобразить дату в формате мм/дд/гг, воспользуйтесь следующим оператором: TheDate = Format(Date, "мм/дд/гг")

Чтобы отобразить в зависимости от времени суток приветствие в строке заголовка, используется конструкция `Select Case`. Значения времени задаются в VBA так же, как в Excel. Если время меньше 0,5 (полдень), то это утро. Если время больше 0,7083 (5 часов вечера), то это вечер. Все остальное время — это день. Мы выбрали простой способ и использовали функцию VBA `Time`, которая возвращает значение времени из строки.

Следующие операторы определяют имя пользователя, указанное на вкладке *Общие* диалогового окна *Параметры Excel*. Для нахождения первого пробела в имени пользователя использована функция VBA `InStr`. Если полное имя не имеет пробелов, оно будет взято целиком. Если поле для введения имени пользователя не заполнено, то Excel всегда использует значение `User`.

Функция `MsgBox` объединяет дату и время, но использует встроенную константу `vbCrLf` для вставки между ними разрыва строки. `vbOKOnly` — предопределенная константа, возвращающая 0; в результате окно сообщения содержит только кнопку ОК. Последний аргумент — приветствие `Greeting`, составленное ранее в процедуре.

Отображение списка шрифтов

Если вам необходимо познакомиться со списком всех установленных шрифтов, помните, что в Excel нет прямого способа получить эту информацию. Описанная методика основана на том, что в Excel для обеспечения совместимости поддерживаются свойства и методы объекта `CommandBar`. Эти свойства и методы использовались для работы с панелями инструментов и меню в версиях Excel, предшествующих Excel 2007.

Макрос `ShowInstalledFonts` отображает список установленных шрифтов в столбце A активного рабочего листа. При этом создается временная панель инструментов (объект `CommandBar`), добавляется элемент управления `Font`, а также считываются шрифты из этого элемента управления. Затем временная панель инструментов удаляется.

```
Sub ShowInstalledFonts()
    Dim FontList As CommandBarControl
    Dim TempBar As CommandBar
    Dim i As Long
```

```
'    Создание временной панели CommandBar
     Set TempBar = Application.CommandBars.Add
     Set FontList = TempBar.Controls.Add(ID:=1728)
'    Помещение шрифтов в столбец A
     Range("A:A").ClearContents
     For i = 0 To FontList.ListCount - 1
       Cells(i + 1, 1) = FontList.List(i + 1)
       Cells(i + 1, 1).Font.Name = FontList.List(i + 1)
     Next i
'    Удаление временной панели CommandBar
     TempBar.Delete
End Sub
```

Название каждого шрифта отображается одновременно с его начертанием (рис. 4). Для этого в состав цикла `For-Next` добавлен оператор:

```
Cells(i + 1, 1).Font.Name = FontList.List(i + 1)
```

Будьте осторожны, поскольку использование большого количества шрифтов в рабочей книге поглотит ресурсы вашей системы и сможет даже вызвать ее крах.

Рис. 4. Отображение названий и начертаний шрифтов

Сортировка массива

Несмотря на то что в Excel существует встроенная команда сортировки ячеек, в VBA метод сортировки массивов не представлен. Один возможный, но достаточно неудобный способ решения этой задачи — перенести массив в диапазон ячеек на рабочем листе, отсортировать данные с помощью команд Excel и занести результат обратно в массив. Однако если в вашей программе имеет большое значение скорость выполнения операции, то лучше написать на VBA процедуру сортировки. Ниже рассматривается несколько методов сортировки.

- *Сортировка на рабочем листе.* Массив переносится на рабочий лист Excel; диапазон на рабочем листе сортируется и переносится обратно в массив. Единственным аргументом этой процедуры является массив.
- *Пузырьковый метод.* Его несложно запрограммировать, однако алгоритм сортировки не самый эффективный, особенно для большого количества элементов в массиве.
- *Быстрая сортировка.* Намного более быстрая процедура, чем пузырьковый алгоритм, но, чтобы в ней разобраться, потребуется время. Работает исключительно с типами данных Integer и Long.

- *Метод пересчета.* Работает очень быстро, однако для его улучшения также потребуются время и определенные усилия. Как и быстрая сортировка, работает с типами данных Integer и Long.

На рис. 5 показано диалоговое окно для этого проекта. При тестировании вы можете задать размер массива (не более 10 тысяч элементов); элементами массивов выступали произвольные числа (типа Long). Код VBA можно изучить в приложенном Excel-файле.

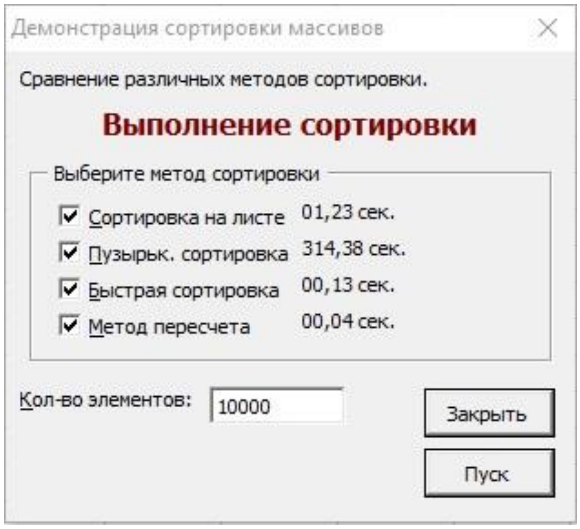

Рис. 5. Сравнение времени необходимого для сортировки 100 000 элементов массива

Обработка последовательности файлов

Одной из главных причин использования макросов является многократное повторение определенной операции. Пример из этого раздела показывает, как выполнить макрос в нескольких разных файлах, сохраненных на диске. Этот пример, который призван помочь вам написать собственную программу выполнения задачи, запрашивает у пользователя сведения о файле и обрабатывает соответствующие запросу рабочие книги. В рассматриваемом случае обработка предусматривает импорт файла и ввод ряда формул суммирования, описывающих данные в файле.

```
Sub BatchProcess()
    Dim FileSpec As String
    Dim i As Integer
    Dim FileName As String
    Dim FileList() As String
    Dim FoundFiles As Integer
'   Указание пути и спецификаций файла
    FileSpec = ThisWorkbook.Path & "\" & "text??.txt"
    FileName = Dir(FileSpec)
'   Найден ли файл?
    If FileName <> "" Then
        FoundFiles = 1
        ReDim Preserve FileList(1 To FoundFiles)
        FileList(FoundFiles) = FileName
    Else
        MsgBox "Не найдены файлы, которые соответствуют " & FileSpec
        Exit Sub
    End If
'   Получение других имен файлов
    Do
        FileName = Dir
        If FileName = "" Then Exit Do
        FoundFiles = FoundFiles + 1
        ReDim Preserve FileList(1 To FoundFiles)
        FileList(FoundFiles) = FileName & "*"
    Loop
```

```
'   Циклический обход и обработка файлов
    For i = 1 To FoundFiles
        Call ProcessFiles(FileList(i))
    Next i
End Sub

Sub ProcessFiles(FileName As String)
'   Импорт файла
    Workbooks.OpenText FileName:=FileName, _
        Origin:=xlWindows, _
        StartRow:=1, _
        DataType:=xlFixedWidth, _
        FieldInfo:= _
        Array(Array(0, 1), Array(3, 1), Array(12, 1))
'   Ввод суммарных формул
    Range("D1").Value = "A"
    Range("D2").Value = "B"
    Range("D3").Value = "C"
    Range("E1:E3").Formula = "=COUNTIF(B:B,D1)"
    Range("F1:F3").Formula = "=SUMIF(B:B,D1,C:C)"
End Sub
```

Файл рассмотренного выше примера, находится в отдельной папке. В ней можно найти дополнительные файлы text01.txt, text02.txt и text03.txt.

Если нужно импортировать другие текстовые файлы, процедуру придется немного изменить. Соответствующие заданному критерию файлы находятся в массиве `Found-Files`, а процедура использует для обработки файлов цикл `For-Next`. В цикле обработка выполняется отдельной процедурой `ProcessFiles`. Эта процедура использует метод `OpenText` для импорта файла и вставки в него пяти формул.