**Работа с диапазонами в VBA**

Настоящая заметка продолжает знакомство с VBA, в ней приводятся примеры управления диапазонами на рабочих листах Excel с помощью VBA.[[1]](#footnote-1)

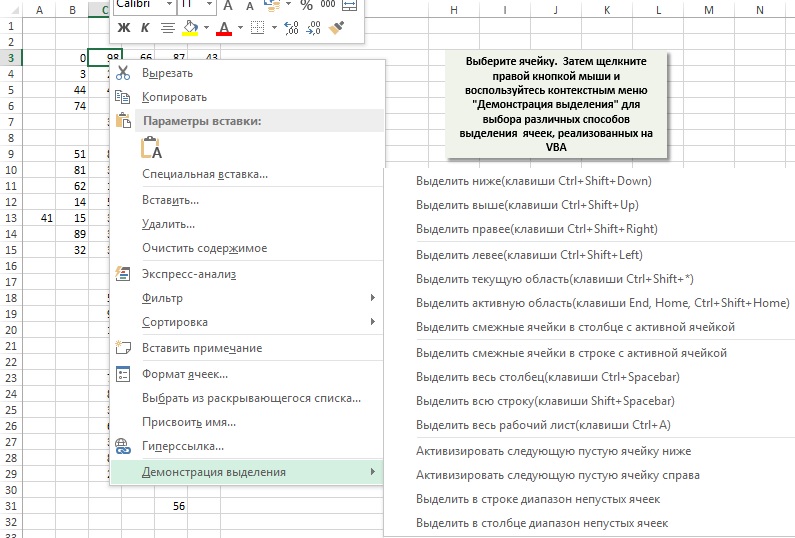


Рис. 1. Пример, демонстрирующий, как выделять диапазоны различной формы в VBA

### Копирование диапазона

Функция записи макросов Excel используется не столько для создания хорошего кода, сколько для поиска названий необходимых объектов, методов и свойств. Например, при записи операции копирования и вставки можно получить код:

Sub Макрос()

Range("A1").Select

Selection.Copy

Range("B1").Select

ActiveSheet.Paste

End Sub

Обратите внимание, что данная программа выделяет ячейки. Однако в VBA для работы с объектом не обязательно его выделять. Данную процедуру можно заменить значительно более простой — применить метод Сору, который использует аргумент, представляющий адрес места вставки копируемого диапазона.

Sub CopyRange()

Range("А1").Copy Range("В1")

End Sub

Предполагается, что рабочий лист является активным и операция выполняется на активном рабочем листе. Чтобы скопировать диапазон на другой рабочий лист или в другую книгу, необходимо задать ссылку:

Sub CopyRange2()

Workbooks("Filel.xlsx").Sheets("Лист1").Range("Al").Copy \_

Workbooks("File2.xlsx").Sheets("Лист2").Range("Al")

End Sub

Еще одним подходом к решению этой задачи является использование для представления диапазонов объектных переменных:

Sub CopyRange3()

Dim Rngl As Range, Rng2 As Range

Set Rngl = Workbooks("Filel.xlsx").Sheets("Лист1").Range("Al")

Set Rng2 = Workbooks("File2.xlsx").Sheets("Лист2").Range("Al")

Rngl.Copy Rng2 End Sub

Можно копировать большой диапазон. Адрес места вставки определяется единственной ячейкой (представляющей верхний левый угол вставляемого диапазона):

Sub CopyRange4 ()

Range("А1:С800").Copy Range("Dl")

End Sub

Для перемещения диапазона ячеек вместо метода Сору используется метод Cut.

Если размер копируемого диапазона не известен используется свойство CurrentRegion, возвращающее объект Range, который соответствует прямоугольнику ячеек вокруг заданной ячейки:

Sub CopyCurrentRegion2()

Range("Al").CurrentRegion.Copy Sheets("Лист2").Range("Al")

End Sub

Метод End имеет один аргумент, определяющий направление, в котором увеличивается выделение ячеек. Следующий оператор выделяет диапазон от активной ячейки до последней непустой ячейки внизу:

Range (ActiveCell, ActiveCell.End(xlDown)).Select

Три остальные константы имитируют комбинации клавиш при выделении в других направлениях: xlUp (вверх), xlToLeft (влево) и xlToRight (вправо).

В прилагаемом Excel-файле определено несколько распространенных типов выделения ячеек (см. рис. 1). Код любопытен тем, что является также примером создания контекстного меню.

### Запрос значения ячейки

Следующая процедура запрашивает значение у пользователя и вставляет его в ячейку А1:

Sub GetValuel()

Range("Al").Value = InputBox("Введите значение")

End Sub

Однако при выполнении этой процедуры возникает проблема. Если пользователь щелкнет на кнопке *Отмена* в окне ввода данных, то процедура удалит данные, которые находились в текущей ячейке. Модифицированная версия процедуры адекватно реагирует на щелчок на кнопке *Отмена* и не выполняет при этом никаких действий:

Sub GetValue2()

Dim UserEntry As Variant

UserEntry = InputBox("Введите значение")

If UserEntry <> "" Then Range("Al").Value = UserEntry

End Sub

Во многих случаях следует проверить правильность данных, введенных пользователем. Например, необходимо обеспечить введение только чисел в диапазоне от 1 до 12 (рис. 2). Это можно сделать при помощи процедуры GetValue3(), код которой приведен в *Модуле1* приложенного Excel-файла. Некорректные данные игнорируются, и окно запроса значения отображается снова. Этот цикл будет повторяться, пока пользователь не введет правильное значение или не щелкнет на кнопке *Отмена*.

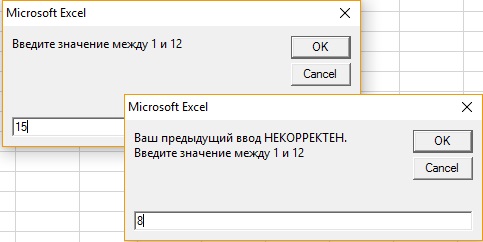


Рис. 2. Проверка данных, введенных пользователем

### Ввод значения в следующую пустую ячейку

Если требуется ввести значение в следующую пустую ячейку столбца или строки, используйте код (рис. 3):

Sub GetData()

Dim NextRow As Long

Dim Entry1 As String, Entry2 As String

Do

' Определение следующей пустой строки

NextRow = Cells(Rows.Count, 1).End(xlUp).Row + 1

' Запрос данных

Entry1 = InputBox("Введите имя")

If Entry1 = "" Then Exit Sub

Entry2 = InputBox("Введите сумму")

If Entry2 = "" Then Exit Sub

' Запись данных

Cells(NextRow, 1) = Entry1

Cells(NextRow, 2) = Entry2

Loop

End Sub

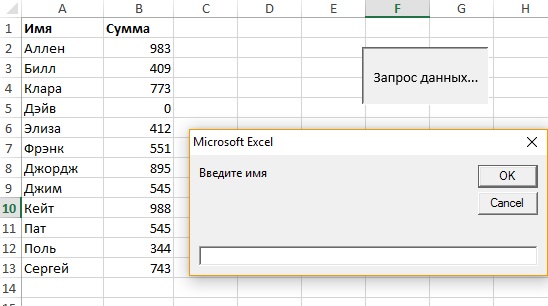


Рис. 3. Макрос вставляет данные в следующую пустую строку рабочего листа

Это бесконечный цикл. Для выхода из него (щелкните на кнопке *Cancel*) использовались операторы Exit Sub. Обратите внимание строку, в который определяется значение переменной NextRow. Если вам трудно ее понять, проанализируйте содержимое ячейки: перейдите в последнюю ячейку столбца А и нажмите <End> и <↑>. После этого будет выделена последняя непустая ячейка в столбце А. Свойство Row возвращает номер этой строки; чтобы получить расположенную под ней строку (следующую пустую строку), к этому номеру прибавляется 1.

### Приостановка работы макроса для определения диапазона пользователем

В некоторых ситуациях макрос должен взаимодействовать с пользователем. Например, можно создать макрос, который приостанавливается, когда пользователь указывает диапазон ячеек. Для этого воспользуйтесь функцией Excel *InputBox*. Не путайте метод Excel *InputBox* с функцией VBA InputBox. Несмотря на идентичность названий, это далеко не одно и то же.

Процедура, представленная ниже, демонстрирует, как приостановить макрос и разрешить пользователю выбрать ячейку. Затем автоматически формула вставляется в каждую ячейку выделенного диапазона.

Sub GetUserRange()

Dim UserRange As Range

Prompt = "Выберите диапазон для случайных чисел."

Title = "Выбор диапазона"

' Отображение поля ввода

On Error Resume Next

Set UserRange = Application.InputBox( \_

Prompt:=Prompt, \_

Title:=Title, \_

Default:=ActiveCell.Address, \_

Type:=8) 'Выделение диапазона

On Error GoTo 0

' Отменено ли отображене поля ввода?

If UserRange Is Nothing Then

MsgBox "Отменено."

Else

UserRange.Formula = "=RAND()"

End If

End Sub

Окно ввода данных показано на рис. 4. Важный момент в этой процедуре – определение аргумента *Туре* равным 8 (в этом случае *InputBox* вернет диапазон; подробнее см. [Application.InputBox Method](https://msdn.microsoft.com/ru-ru/library/ff839468.aspx)).

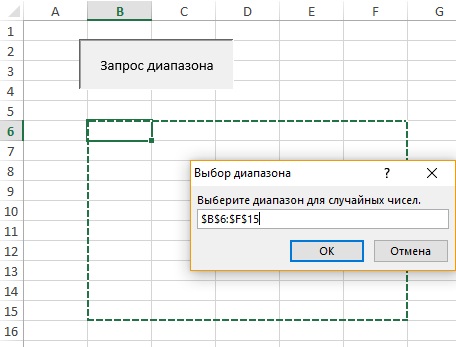


Рис. 4. Использование окна ввода данных с целью приостановки выполнения макроса

Оператор On Error Resume Next игнорирует ошибку, если пользователь не выберет диапазон, а щелкает *Отмена*. В таком случае объектная переменная UserRange не получает значения. В этом случае отобразится окно сообщения с текстом «Отменено». Если же пользователь щелкнет на кнопке OK, то макрос продолжит выполняться. Строка On Error Go То указывает на переход к стандартной обработке ошибки. Проверка корректного выделения диапазона необязательна. Excel позаботится об этом вместо вас.

Обязательно проверьте, включено ли обновление экрана при использовании метода *InputBox* для выделения диапазона. Если обновление экрана отключено, вы не сможете выделить рабочий лист. Чтобы проконтролировать обновление экрана, в процессе выполнения макроса используйте свойство ScreenUpdating объекта Application.

### Подсчет выделенных ячеек

Работая с макросом, который обрабатывает выделенный диапазон ячеек, можно использовать свойство Count, чтобы определить, сколько ячеек содержится в выделенном (или любом другом) диапазоне. Например, оператор MsgBox Selection.Count демонстрирует окно сообщения, которое отображает количество ячеек в текущем выделенном диапазоне. Свойство Count использует тип данных Long, поэтому наибольшее значение, которое может храниться в нем, равно 2 147 483 647. Если выделить лист целиком, то ячеек будет больше, и свойство Count сгенерирует ошибку. Используйте свойство CountLarge, которое не имеет таких ограничений.

Если активный лист содержит диапазон data, то следующий оператор присваивает количество ячеек в диапазоне data переменной с названием CellCount:

CellCount = Range("data").Count

Вы можете также определить, сколько строк или столбцов содержится в диапазоне. Следующее выражение вычисляет количество столбцов в выделенном диапазоне:

Selection.Columns.Count

Следующий оператор пересчитывает количество строк в диапазоне с названием data и присваивает это количество переменной RowCount.

RowCount = Range ("data" ). Rows . Count

### Просмотр выделенного диапазона

Вы можете столкнуться с трудностями при создании макроса, который оценивает каждую ячейку в диапазоне и выполняет операцию, определенную заданному критерию. Если выделен целый столбец или строка, то работа макроса может занять много времени. Процедура ColorNegative устанавливает красный цвет для ячеек, которые содержат отрицательные значения. Цвет фона для других ячеек не определяется. Код процедуры можно найти в *Модуле4* приложенного Excel-файла.

Усовершенствованная процедура ColorNegative2, создает объектную переменную WorkRange типа Range, которая представляет собой пересечение выделенного диапазона и диапазона рабочего листа (рис. 5). Если выделить столбец F (1048576 ячеек), то его пересечение с рабочим диапазоном В2:I16) даст область F2:F16, которая намного меньше исходного выделенного диапазона. Время, затрачиваемое на обработку 15 ячеек, намного меньше времени, уходящего на обработку миллиона ячеек.

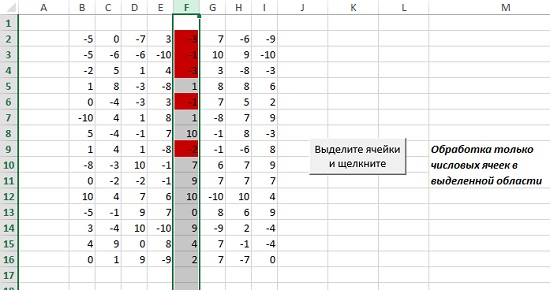


Рис. 5. В результате пересечения используемого диапазона и выделенного диапазона рабочего листа уменьшается количество обрабатываемых ячеек

И всё же процедура ColorNegative2 недостаточно эффективна, поскольку обрабатывает все ячейки в диапазоне. Поэтому предлагается процедура ColorNegative3. В ней используется метод SpecialCells, с помощью которого генерируются два поднабора выделенной области: один поднабор (ConstantCells) включает ячейки, которые содержат исключительно числовые константы; второй поднабор (FormulaCells) включает ячейки, содержащие числовые формулы. Обработка ячеек в этих поднаборах осуществляется с помощью двух конструкций For Each-Next. Благодаря тому, что исключается обработка пустых и нетекстовых ячеек, скорость выполнения макроса существенно увеличивается.

Sub ColorNegative3()

' Окрашивание ячеек с отрицательными значениями в красный цвет

Dim FormulaCells As Range, ConstantCells As Range

Dim cell As Range

If TypeName(Selection) <> "Range" Then Exit Sub

Application.ScreenUpdating = False

' Создание поднаборов исходной выделенной области

On Error Resume Next

Set FormulaCells = Selection.SpecialCells(xlFormulas, xlNumbers)

Set ConstantCells = Selection.SpecialCells(xlConstants, xlNumbers)

On Error GoTo 0

' Обработка ячеек с формулами

If Not FormulaCells Is Nothing Then

For Each cell In FormulaCells

If cell.Value < 0 Then

cell.Interior.Color = RGB(255, 0, 0)

Else

cell.Interior.Color = xlNone

End If

Next cell

End If

' Обработка ячеек с константами

If Not ConstantCells Is Nothing Then

For Each cell In ConstantCells

If cell.Value < 0 Then

cell.Interior.Color = RGB(255, 0, 0)

Else

cell.Interior.Color = xlNone

End If

Next cell

End If

End Sub

Оператор On Error необходим, поскольку метод SpecialCells генерирует ошибку, если не находит в диапазоне ячеек указанного типа.

### Удаление всех пустых строк

Следующая процедура удаляет все пустые строки в активном рабочем листе. Она достаточно эффективна, так как не проверяет все без исключения строки, а просматривает только строки в так называемом «используемом диапазоне», определяемом с помощью свойства UsedRange объекта Worksheet.

Sub DeleteEmptyRows()

Dim LastRow As Long

Dim r As Long

Dim Counter As Long

Application.ScreenUpdating = False

LastRow = ActiveSheet.UsedRange.Rows.Count + \_

ActiveSheet.UsedRange.Rows(1).Row - 1

For r = LastRow To 1 Step -1

If Application.WorksheetFunction.CountA(Rows(r)) = 0 Then

Rows(r).Delete

Counter = Counter + 1

End If

Next r

Application.ScreenUpdating = True

MsgBox Counter & " Пустые строки удалены."

End Sub

Первый шаг — определить последнюю используемую строку и присвоить этот номер строки переменной LastRow. Это не так просто, как можно ожидать, поскольку текущий диапазон необязательно начинается со строки 1. Следовательно, значение LastRow вычисляется таким образом: к найденному количеству строк используемого диапазона прибавляется номер первой строки текущего диапазона и вычитается 1.

В процедуре применена функция Excel СЧЁТЗ, определяющая, является ли строка пустой. Если данная функция для конкретной строки возвращает 0, то эта строка пустая. Обратите внимание, что процедура просматривает строки снизу вверх и использует отрицательное значение шага в цикле For-Next. Это необходимо, поскольку при удалении все последующие строки перемещаются «вверх» в рабочем листе. Если бы в цикле просмотр выполнялся сверху вниз, то значение счетчика цикла после удаления строки оказалось бы неправильным.

В макросе используется еще одна переменная, Counter, с помощью которой подсчитывается количество удаленных строк. Эта величина отображается в окне сообщения по завершении процедуры.

### Дублирование строк

Пример, рассматриваемый в этом разделе, демонстрирует использование возможностей VBA для создания дубликатов строк. На рис. 6 показан пример рабочего листа, используемого организаторами лотереи. В столбце *А* вводится имя. В столбце *В* содержится количество лотерейных билетов, приобретенных одним покупателем. В столбце *С* находится случайное число сгенерированное с помощью функции СЛЧИС. Победитель определяется путем сортировки данных в третьем столбце (выигрыш соответствует наибольшему случайному числу).

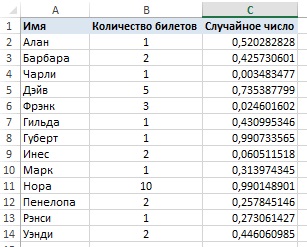


Рис. 6. Дублирование строк на основе значений в столбце В

А теперь нужно продублировать строки, в результате чего количество строк для каждого участника лотереи будут соответствовать количеству купленных им билетов. Например, если Барбара приобрела два билета, для нее создаются две строки. Ниже показана процедура, выполняющая вставку новых строк.

Sub DupeRows()

Dim cell As Range

' 1-я ячейка, содержащая сведения о количестве билетов

Set cell = Range("B2")

Do While Not IsEmpty(cell)

If cell > 1 Then

Range(cell.Offset(1, 0), cell.Offset(cell.Value \_

- 1,0)).EntireRow.Insert

Range(cell, cell.Offset(cell.Value - 1, - 1)). \_ EntireRow.FillDown

End If

Set cell = cell.Offset(cell.Value, 0)

Loop

End Sub

Объектная переменная cell была инициализирована ячейкой В2, первой ячейкой, в которой находится числовая величина. Вставка новых строк осуществляется в цикле, а их копирование происходит с помощью метода FillDown. Значение переменной cell увеличивается на единицу, после чего выбирается следующий участник лотереи, Цикл выполняется до тех пор, пока не встретится пустая ячейка. На рис. 7 показан рабочий лист после выполнения этой процедуры.

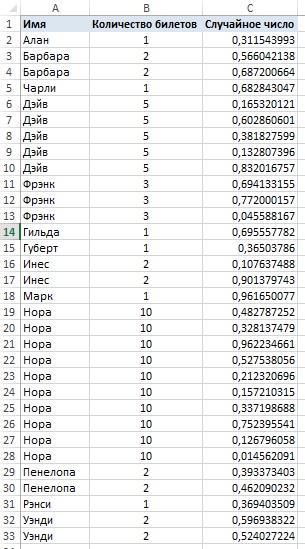


Рис. 7. В соответствии со значением в столбце В добавлены новые строки

### Определение диапазона, находящегося в другом диапазоне

Функция InRange имеет два аргумента, оба — объекты Range. Функция возвращает значение True (Истина), если первый диапазон содержится во втором.

Function InRange(rng1, rng2) As Boolean

' Возвращает True, если rng1 является подмножеством rng2

InRange = False

If rng1.Parent.Parent.Name = rng2.Parent.Parent.Name Then

If rng1.Parent.Name = rng2.Parent.Name Then

If Union(rng1, rng2).Address = rng2.Address Then

InRange = True

End If

End If

End If

End Function

Возможно, функция InRange кажется сложнее, чем того требует ситуация, поскольку в коде должна быть реализована проверка принадлежности двух диапазонов одной и той же книге и рабочему листу. Обратите внимание, что в процедуре используется свойство Parent, которое возвращает объект-контейнер заданного объекта. Например, следующее выражение возвращает название листа для объекта rng1:

rng1.Parent.Name

Следующее выражение возвращает название рабочей книги rng1:

rng1.Parent.Parent.Name

Функция VBA Union возвращает объект Range, который представляет собой объединение двух объектов типа Range. Объединение содержит все ячейки, относящиеся к исходным диапазонам. Если адрес объединения двух диапазонов совпадает с адресом второго диапазона, первый диапазон входит в состав второго диапазона.

### Определение типа данных ячейки

В состав Excel входит ряд встроенных функций, которые могут помочь определить тип данных, содержащихся в ячейке. Это функции ЕНЕТЕКСТ, ЕЛОГИЧ и ЕОШИБКА. Кроме того, VBA поддерживает функции IsEmpty, IsDate и IsNumeric.

Ниже описана функция CellType, которая принимает аргумент-диапазон и возвращает строку, описывающую тип данных левой верхней ячейки этого диапазона (рис. 8). Такую функцию можно использовать в формуле рабочего листа или вызвать из другой процедуры VBA.

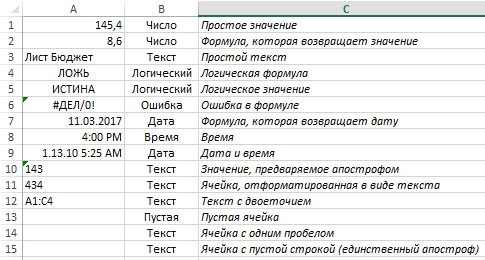


Рис. 8. Функция CellType, возвращающая тип данных ячейки

Function CellType(Rng)

' Возвращает тип ячейки, находящейся в левом верхнем углу диапазона

Dim TheCell As Range

Set TheCell = Rng.Range("A1")

Select Case True

Case IsEmpty(TheCell)

CellType = "Пустая"

Case TheCell.NumberFormat = "@"

CellType = "Текст"

Case Application.IsText(TheCell)

CellType = "Текст"

Case Application.IsLogical(TheCell)

CellType = "Логический"

Case Application.IsErr(TheCell)

CellType = "Ошибка"

Case IsDate(TheCell)

CellType = "Дата"

Case InStr(1, TheCell.Text, ":") <> 0

CellType = "Время"

Case IsNumeric(TheCell)

CellType = "Число"

End Select

End Function

Обратите внимание на использование оператора SetTheCell. Функция CellType получает аргумент-диапазон произвольного размера, но этот оператор указывает, что функция оперирует только левой верхней ячейкой диапазона (представленной переменной TheCell).

1. По материалам книги [Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA](http://baguzin.ru/wp/?p=16255). – М: Диалектика, 2013. – С. 325–342. [↑](#footnote-ref-1)