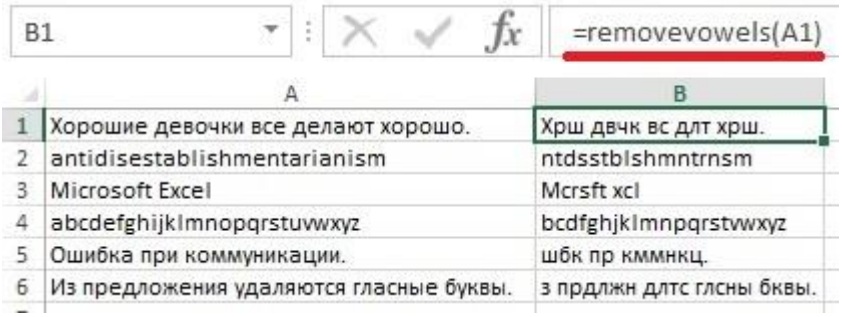
Пользовательские функции VBA

Ранее были рассмотрены процедуры VBA. В настоящей заметке рассмотрены функции VBA.[1] *Функция* — это процедура VBA, которая выполняет вычисления и возвращает значение. Функции можно использовать в коде VBA или в формулах Excel. Процедуру можно рассматривать как команду, которая выполняется пользователем или другой процедурой. С другой стороны, функция обычно возвращает отдельное значение (или массив) подобно функциям рабочих листов Excel и встроенным функциям VBA.



	A	B
1	Хорошие девочки все делают хорошо.	Хрш двчк вс длт хрш.
2	antidisestablishmentarianism	ntdsstblshmntrnsm
3	Microsoft Excel	Mcrsft xcl
4	abcdefghijklmnopqrstuvwxyz	bcdfghjklmnpqrstvwxyz
5	Ошибка при коммуникации.	шбк пр кммнкц.
6	Из предложения удаляются гласные буквы.	з прдлжн длтс глсны бквы.

Рис. 1. Применение пользовательской функции в формуле рабочего листа

Excel содержит более 400 встроенных функций. Если этого количества недостаточно, можно создавать пользовательские функции с помощью VBA. Однако следует отметить, что функции VBA, используемые в формулах, обычно выполняются медленнее, чем встроенные функции Excel. Пользовательские функции отображаются в диалоговом окне *Мастер функций* наряду со встроенными функциями Excel.

Пример пользовательской функции

Начнем с примера – функции *RemoveVowels (УдалитьГласные)*, которая принимает текстовый аргумент, удаляет все гласные буквы и возвращает текст, состоящий только из согласных.

```
Function RemoveVowels(txt) As String
' Удаляет все гласные звуки из аргумента Txt
   Dim i As Long
   RemoveVowels = ""
   For i = 1 To Len(txt)
      If Not ucase(Mid(txt, i, 1)) Like "[АЕIOUАЕИОУЮЭЯ]" Then
         RemoveVowels = RemoveVowels & Mid(txt, i, 1)
      End If
   Next i
End Function
```

> Код пользовательских функций, которые используются в формуле рабочего листа, вводите в обычном модуле VBA. Если вы поместите пользовательские функции в модуле *Лист*, в *Пользовательской форме* или в модуле *ЭтаКнига*, они не будут выполняться в формулах.

Функцию *RemoveVowels* можно использовать, например, в формуле в ячейке B1 (рис. 1) =RemoveVowels (A1). Вы также можете создавать вложенные пользовательские функции и сочетать их в формулах с обычными функциями Excel. Например, =ПРОПИСН(RemoveVowels(A1))

Пользовательские функции можно применять не только в формулах рабочего листа, но и в процедурах VBA. Например, процедура *ZapTheVowels()* сначала отображает окно для ввода текста пользователем, затем обрабатывает этот текст функцией *RemoveVowels*, и наконец использует встроенную функцию VBA *MsgBox* для отображения результатов (рис. 2). Первоначальные данные отображаются в заголовке окна сообщения.

```
Sub ZapTheVowels()
    Dim UserInput As String
```

[1] По материалам книги Джон Уокенбах. Excel 2010. Профессиональное программирование на VBA. – М: Диалектика, 2013. – С. 287–323.

```
    UserInput = InputBox("Введите текст:")
    MsgBox RemoveVowels(UserInput), vbInformation, UserInput
End Sub
```

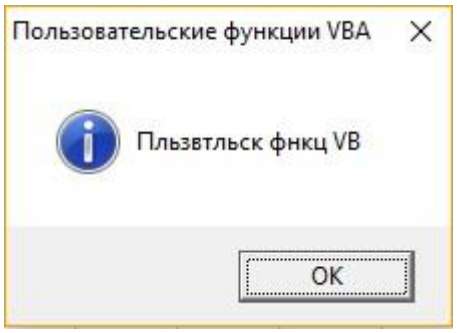


Рис. 2. Применение пользовательской функции в процедуре VBA

Помните, что функции, используемые в формулах рабочего листа, — «пассивные». Они **не могут** изменять содержимое рабочего листа. Например, нельзя написать функцию, которая будет изменять цвет текста в ячейке в зависимости от значения этой ячейки. Функция возвращает значение, но не может выполнять операции над объектами.

Из этого правила имеется одно исключение. Вы можете изменить текст комментария ячейки с помощью пользовательской функции VBA:

```
Function ModifyComment(Cell As Range, Cmt As String)
        Cell.Comment.Text Cmt
End Function
```

Например, можно ввести в ячейку B1 формулу =ModifyComment(A1,"Комментарий был изменен"). Функция не работает, если в ячейке A1 отсутствует комментарий.

Рассмотрим код функции *RemoveVowels* подробнее. Функция начинается с ключевого слова Function, а не Sub, после которого указывается название функции (RemoveVowels). Эта специальная функция использует только один аргумент (Txt), заключенный в скобки. Ключевое слово As String определяет тип данных значения, которое возвращает функция. (Excel по умолчанию использует тип данных Variant, если тип данных не определен.)

Вторая строка — простой комментарий (необязательный), который описывает выполняемые функцией действия. После комментария приведен оператор Dim, который объявляет переменную (i), применяемую в функции. Тип этой переменной — Long. Далее в качестве переменной используется имя функции. Как только функция завершает свое выполнение, возвращается текущее значение переменной, которое соответствует названию функции.

Следующие пять инструкций образуют цикл For-Next. Процедура циклически просматривает каждый символ введенного текста, создавая на их основе строку. Первая инструкция в цикле использует функцию VBA Mid, которая возвращает единственный символ строки ввода, а также преобразует этот символ в символ верхнего регистра. Затем этот символ сравнивается со списком символов с помощью оператора VBA Like (подробнее см. Оператор Like). Другими словами, значение выражения If будет True, если символ отличен от символов A, E, I, O, U, А, Е, И, О, У, Ы, Э, Ю и Я. В подобных случаях символ добавляется к переменной *RemoveVowels*.

По завершении цикла из строки ввода удаляются все гласные буквы. Эта строка и является значением, возвращаемым функцией RemoveVowels. Процедура завершается оператором End Function. (Альтернативный код – *RemoveVowels2*, выполняющий ту же задачу приведен в модуле VBA приложенного Excel-файла.)

Синтаксис функции

Для объявления функции применяется следующий синтаксис (элементы аналогичны обычной процедуре; подробнее см. Работа с процедурами VBA).

```
[Public | Private][Static] Function имя ([список_аргументов])[As тип]
      [инструкции]
```

```
    [имя = выражение]
    [Exit Function]
    [инструкции]
    [имя = выражение]
End Function
```

Значение всегда присваивается названию функции минимум один раз и, как правило, тогда, когда функция завершила выполнение. Создание пользовательской функции начните с создания модуля VBA (можно также использовать существующий модуль). Введите ключевое слово `Function`, после которого укажите название функции и список ее аргументов (если они есть) в скобках. Вы также можете объявить тип данных значения, которое возвращает функция, используя ключевое слово `As` (это делать необязательно, но рекомендуется). Вставьте код VBA, выполняющий требуемые действия, и убедитесь, что необходимое значение присваивается переменной процедуры, соответствующей названию функции, минимум один раз в теле функции. Функция заканчивается оператором `End Function`.

Имена функций подчиняются тем же правилам, что и имена переменных. Если вы планируете использовать функцию в формуле рабочего листа, убедитесь, что название не имеет форму адреса ячейки. Также не присваивайте функциям имена, которые соответствуют названиям встроенных функций Excel. Если область действия функции не задана, то по умолчанию подразумевается `Public`. Функции, объявленные как `Private`, не отображаются в диалоговом окне *Мастер функций*.

Функцию можно вызвать одним из следующих способов:

- вызвать ее из другой процедуры;
- включить ее в формулу рабочего листа;
- включить в формулу условного форматирования;
- вызвать ее в окне отладки VBE (*Immediate*). Этот метод обычно применяется на этапе тестирования (рис. 3).

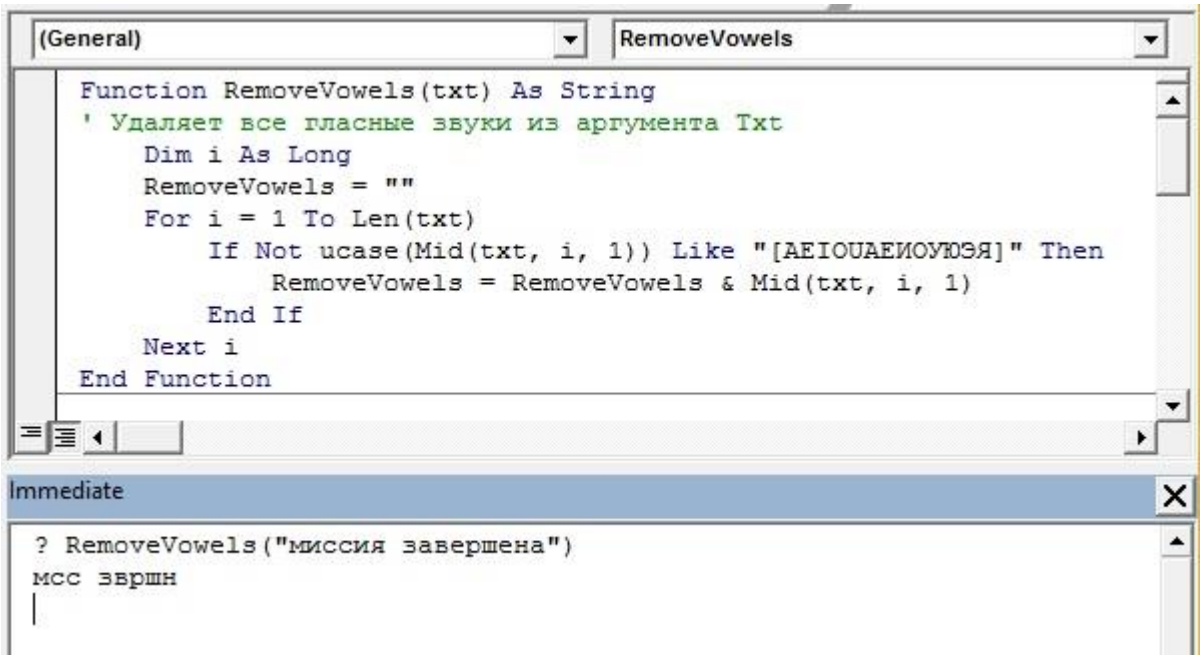


Рис. 3. Вызов функции в окне отладки

В отличие от процедур, функции не отображаются в диалоговом окне *Макрос* (меню *Разработчик –> Код –> Макросы*; или Alt+F8).

Аргументы функций

Аргументы могут представляться переменными (в том числе массивами), константами, символьными данными или выражениями. Некоторые функции не имеют аргументов. Функции имеют как обязательные, так и необязательные аргументы.

Функции без аргументов

В Excel есть несколько встроенных функций, не имеющих аргументов, например, СЛЧИС, СЕГОДНЯ, ТДАТА. Несложно создать аналогичные пользовательские функции. Например:

```
Function User()
' Возвращает имя пользователя
    User = Application.UserName
End Function
```

При вводе формулы =User() ячейка возвращает имя текущего пользователя (рис. 4). Обратите внимание: при использовании функции без аргумента в формуле рабочего листа необходимо указать пустые скобки.

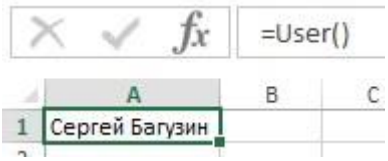


Рис. 4. Формула =User() возвращает имя текущего пользователя

Пользовательские функции ведут себя подобно встроенным функциям Excel. Обычно пользовательская функция пересчитывается тогда, когда это нужно, т.е. в случае изменения одного из аргументов функции. Однако вы можете выполнять пересчет функций чаще. Функция пересчитывается при изменении любой ячейки, если в процедуру добавлен оператор

```
Application.Volatile True
```

Метод `Volatile` объекта `Application` имеет один аргумент (`True` или `False`). Если функция выделена как *volatile* (изменяемая), она пересчитывается всякий раз, когда изменяется любая ячейка листа. При использовании аргумента `False` метода `Volatile` функция пересчитывается только тогда, когда в результате пересчета изменяется один из ее аргументов.

В Excel есть встроенная функция СЛЧИС. Но мне не слишком понравилось, что случайные числа изменяются при каждом пересчете рабочего листа. Поэтому я разработал функцию, которая возвращает случайные числа, не изменяющиеся при пересчете формул. Для этого была использована встроенная функция VBA `Rnd`:

```
Function StaticRand()
' Возвращает случайное число, не изменяемое при пересчете формул
    StaticRand = Rnd()
End Function
```

Значения, полученные с помощью этой формулы, никогда не изменяются. Но у пользователя остается возможность принудительного пересчета формулы с помощью комбинации клавиш <Ctrl+Alt+F9>.

Функция с одним аргументом

Допустим вам нужно подсчитать комиссионные, зависящие от объема продаж. Вычисления основываются на следующей таблице значений:

	A	B
1	Объем продаж за месяц, $	Ставка комиссионных, %
2	0-9999	8,0%
3	10000-19999	10,5%
4	20000-39999	12,0%
5	Более 40000	14,0%

Рис. 5. Таблица комиссионных

Существует несколько способов вычислить комиссионные. Например, с помощью следующей формулы (если объем продаж поместить в ячейку D1):

=ЕСЛИ(И(D1>=0;D1<=9999,99);D1*0,08;ЕСЛИ(И(D1>=10000;D1<=19999,99);D1*0,105;ЕСЛИ(И(D1>=20000;D1<=39999,99);D1*0,12;ЕСЛИ(D1>=40000;D1*0,14))))

Эта формула неудачна по нескольким причинам. Во-первых, она сложна, ее нелегко набрать, и в дальнейшем редактировать. Во-вторых, значения строго определены в формуле, из-за чего ее сложно изменять. Гораздо лучше использовать ВПР (рис. 6).

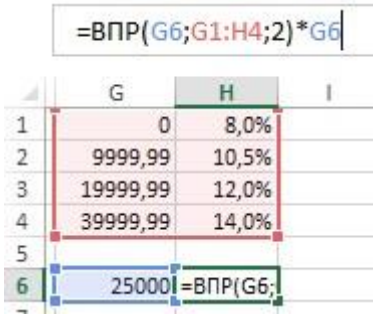


Рис. 6. Использование функции ВПР для вычисления комиссионных

Еще лучше (тогда не нужно использовать таблицу соответствия) создать пользовательскую функцию:

```
Function Commission(Sales)
    Const Tier1 = 0.08
    Const Tier2 = 0.105
    Const Tier3 = 0.12
    Const Tier4 = 0.14
'   Вычисление комиссионных с продаж
    Select Case Sales
       Case 0 To 9999.99: Commission = Sales * Tier1
       Case 10000 To 19999.99: Commission = Sales * Tier2
       Case 20000 To 39999.99: Commission = Sales * Tier3
       Case Is >= 40000: Commission = Sales * Tier4
    End Select
End Function
```

После ввода в модуль VBA эту функцию можно использовать в формуле на рабочем листе или вызвать из других процедур VBA. При вводе в ячейку следующей формулы будет получен результат 3000:

=Commission(B2)

Используйте аргументы, а не ссылки на ячейки. Все применяемые в пользовательской функции диапазоны должны передаваться в качестве аргументов. Рассмотрим функцию, которая возвращает значение в ячейке A1, умноженное на 2.

```
Function DoubleCell()
     DoubleCell = Range("A1") * 2
End Function
```

Хотя эта функция работает, в некоторых случаях она выдает неправильный результат. Причина в том, что вычислительный механизм Excel не учитывает диапазоны, которые не передаются в качестве аргументов. Вследствие этого иногда перед возвратом функцией значения, не вычисляются все связанные величины. Следует также написать функцию `DoubleCell`, в качестве аргумента которой передается значение ячейки A1.

```
Function DoubleCell(cell)
     DoubleCell = cell * 2
End Function
```

Функция с двумя аргументами

Представим, что менеджер, о котором речь шла выше, внедряет новую политику, разработанную для уменьшения текучести кадров: общая сумма комиссионных, подлежащих выплате, увеличивается на 1% за каждый год, который служащий проработал в компании. Изменим пользовательскую функцию Commission так, чтобы она принимала два аргумента. Новый аргумент

представляет количество лет, отработанных сотрудником в компании. Назовем эту новую функцию Commission2:

```
Function Commission2(Sales, Years) As Single
'    Вычисление комиссионных с продаж на основе
'    длительности стажа
     Commission2 = Commission(Sales) + _
      (Commission(Sales) * Years / 100)
End Function
```

Функция с аргументом в виде массива

В качестве аргументов функции могут принимать один или несколько массивов, обрабатывать этот массив (массивы) и возвращать единственное значение. Функция, представленная ниже, принимает в качестве аргумента массив и возвращает сумму его элементов.

```
Function SumArray(List) As Double
    Dim Item As Variant
    SumArray = 0
    For Each Item In List
        If WorksheetFunction.IsNumber(Item) Then _
            SumArray = SumArray + Item
    Next Item
End Function
```

Функция Excel ЕЧИСЛО проверяет, является ли каждый элемент числом, прежде чем добавить его к общему целому. Добавление этого простого оператора проверки данных устраняет ошибки несоответствия типов при попытке выполнить арифметическую операцию над строкой.

Функция с необязательными аргументами

Многие встроенные функции Excel имеют необязательные аргументы. Пример — функция ЛЕВСИМВ, возвращающая символы с левого края строки. Она имеет следующий синтаксис:

ЛЕВСИМВ(*текст, кол_символов*)

Первый аргумент — обязательный, в отличие от второго. Если не указан второй аргумент, Excel предполагает значение 1.

Пользовательские функции, разработанные в VBA, также могут иметь необязательные аргументы. Необязательный аргумент вы зададите, если введете перед именем аргумента ключевое слово `Optional`. В списке аргументов необязательные аргументы определяются после всех обязательных. Например:

```
Function User2(Optional Uppercase As Variant)
    If IsMissing(Uppercase) Then Uppercase = False
    User2 = Application.UserName
    If Uppercase Then User2 = UCase(User2)
End Function
```

Если аргумент равен `False` или опущен, то имя пользователя возвращается без каких-либо изменений. Если же аргумент функции `True`, то имя пользователя возвращается в символах верхнего регистра (с помощью VBA-функции `Ucase`). Обратите внимание на первый оператор функции — он содержит VBA-функцию `IsMissing`, которая определяет наличие аргумента. Если аргумент отсутствует, оператор присваивает переменной `Uppercase` значение `False` (задано по умолчанию).

Функция VBA, возвращающая массив

VBA содержит весьма полезную функцию с названием `Array`. Она возвращает значение с типом данных `Variant`, которое содержит массив (т.е. несколько значений). Если вы не знакомы с формулами массивов в Excel, предлагаю начать с Excel. Введение в формулы массива. Формула массива вводится в ячейку после нажатия <Ctrl+Shift+Entei>. Excel добавляет вокруг формулы скобки, чтобы указать, что это формула массива.

Функция `MonthNames` — простой пример применения функции `Array` в пользовательской функции.

```
Function MonthNames()
    MonthNames = Array("Январь", "Февраль", "Март", _
      "Апрель", "Май", "Июнь", "Июль", "Август", _
      "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"
End Function
```

Функция `MonthNames` возвращает горизонтальный массив названий месяцев. На рабочем листе выделите 12 ячеек, введите формулу =MonthNames() и нажмите <Ctrl+Shift+Enter>. Если необходимо сгенерировать вертикальный массив названий месяцев, выделите вертикальный диапазон, введите формулу =ТРАНСП(MonthNames()) и нажмите <Ctrl+Shift+Enter>.

Функция, возвращающая значение ошибки

В VBA содержатся встроенные константы для обозначения ошибок, которые должна возвращать пользовательская функция (эти значения — ошибки выполнения формул Excel, а не ошибки выполнения кода VBA):

- xlErrDivO (для ошибки #ДЕЛ/ 0!);
- xlErrNA (для ошибки #Н/Д);
- xlErrName (для ошибки #ИМЯ?);
- xlErrNull (для ошибки #ПУСТО!);
- xlErrNum (для ошибки #ЧИСЛО!);
- xlErrRef (для ошибки #ССЫЛ!);
- xlErrValue (для ошибки #ЗНАЧ!).

Ниже приведена преобразованная функция RemoveVowels (см. пример в начале). Конструкция `If-Then` применяется для выполнения альтернативного действия в случае, когда аргумент не является текстовым. Эта функция вызывает функцию Excel ЕТЕКСТ, которая определяет, содержит ли аргумент текст. Если ячейка содержит текст, то функция возвращает нормальный результат. Если же ячейка содержит не текст (или пуста), то функция возвращает ошибку #ЗНАЧ!

```
Function RemoveVowels3(txt) As Variant
' Удаляет все гласные буквы из аргумента Txt
' Возвращает ошибку #ЗНАЧ!, если аргумент - не строка
    Dim i As Long
    RemoveVowels3 = ""
    If Application.WorksheetFunction.IsText(txt) Then
        For i = 1 To Len(txt)
            If Not UCase(Mid(txt, i, 1)) Like "[AEIOUAЕИОУЮЭЯ]" Then
                RemoveVowels3 = RemoveVowels3 & Mid(txt, i, 1)
            End If
        Next i
    Else
        RemoveVowels3 = CVErr(xlErrValue)
    End If
End Function
```

Обратите внимание, что был изменен тип данных для возвращаемого функцией значения. Поскольку функция может возвращать что-то еще, кроме строки, тип данных был изменен на `Variant`.

Функция с неопределенным количеством аргументов

Существует возможность создавать пользовательские функции, имеющие неопределенное количество аргументов. Примените в качестве последнего (или единственного) аргумента массив и добавьте перед ним ключевое слово `ParamArray` (`ParamArray` относится только к последнему аргументу в списке аргументов процедуры. Он всегда имеет тип данных `Variant` и всегда является необязательным аргументом). Следующая функция возвращает сумму всех аргументов, в качестве которых может выступать, как одно значение (ячейка), так и диапазон.

```
Function SimpleSum(ParamArray arglist() As Variant) As Double
```

```
    Dim cell As Range
    Dim arg As Variant
    For Each arg In arglist
        For Each cell In arg
            SimpleSum = SimpleSum + cell
        Next cell
    Next arg
End Function
```

Отладка функций

При использовании формулы на рабочем листе для тестирования функции происходящие в процессе выполнения ошибки не отображаются в знакомом диалоговом окне сообщений. Формула просто возвращает значение ошибки (#ЗНАЧ!). К счастью, это не представляет большой проблемы при отладке функций, так как всегда существует несколько обходных путей.

- Поместите в важных местах функцию `MsgBox`, чтобы контролировать значения отдельных переменных.
- Протестируйте функцию, вызвав ее из процедуры, а не в формуле рабочего листа. Ошибки в процессе выполнения отображаются обычным образом.
- Определите точку остановки в функции и просмотрите функцию пошагово. При этом можно воспользоваться всеми стандартными инструментами отладки. Чтобы добавить точку остановки, поместите курсор в операторе, в котором вы решили приостановить выполнение, и выберите команду *Debug –> Toggle Breakpoint* (*Отладка –> Точка остановки*) или нажмите <F9>.
- Используйте в программе один или несколько временных операторов `Debug.Print` (Отладка, Печать), чтобы отобразить значения в окне *Immediate* редактора VBA. Например, чтобы проконтролировать циклически изменяемое значение, используйте следующий метод:

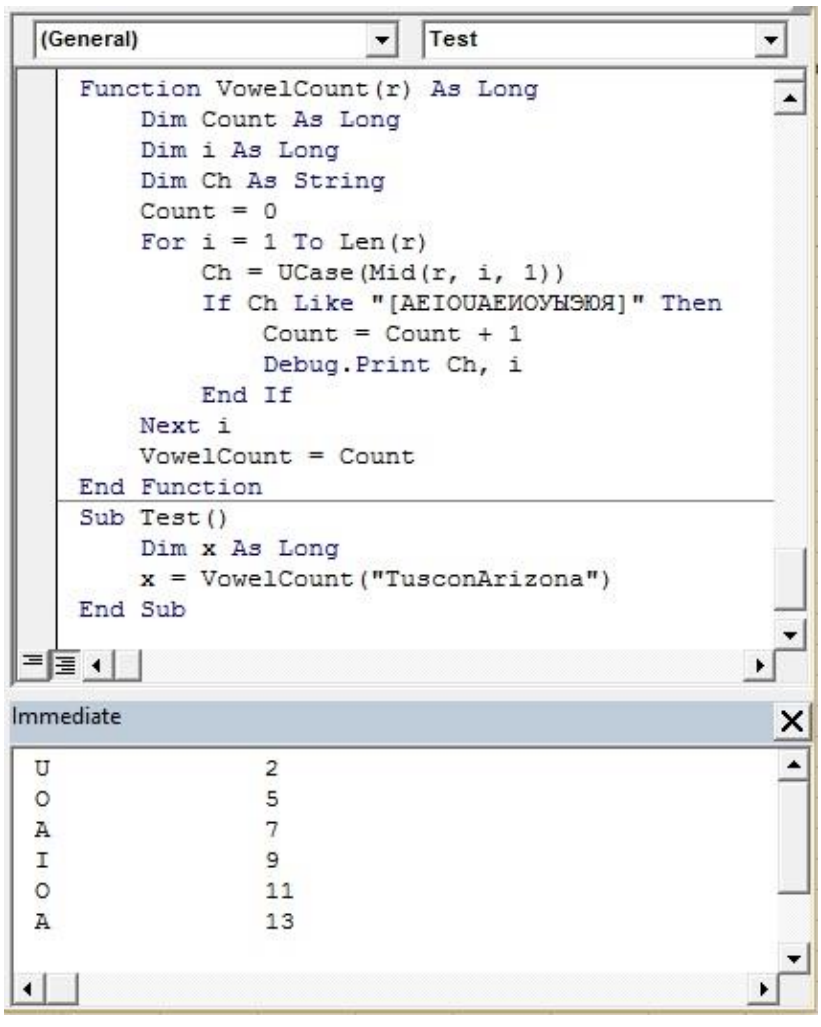


Рис. 7. Используйте окно отладки для отображения результатов при выполнении функции

В данном случае значения двух переменных, Ch и i, выводятся в окне отладки (*Immediate*) всякий раз, когда в программе встречается оператор `Debug.Print`. Встаньте курсором в любое место процедуры `Test()` и нажмите F5. На рис. 7 показан результат для случая, когда функция принимает аргумент `TusconArizona`.

Использование метода MacroOptions

Можно воспользоваться методом `MacroOptions` объекта `Application`, который позволяет включить в состав встроенных функций Excel разработанные вами функции. Этот метод позволяет:

- добавить описание функции (начиная с версии Excel 2010;
- указать категорию функции;
- добавить описание аргументов функции.

```
Sub DescribeFunction()
    Dim FuncName As String
    Dim FuncDesc As String
    Dim FuncCat As Long
    Dim Arg1Desc As String, Arg2Desc As String
    FuncName = "Draw"
    FuncDesc = "Содержимое случайной ячейки диапазона"
    FuncCat = 5 'Ссылки и массивы
    Arg1Desc = "Диапазон, который содержит значения"
    Arg2Desc = "(не обязательный) Если False или отсутствует, _
      функция Rnd не пересчитывается. "
    Arg2Desc = Arg2Desc & "Если True, функция Rnd пересчитывается "
    Arg2Desc = Arg2Desc & "при любом изменении на листе."
    Application.MacroOptions _
        Macro:=FuncName, _
        Description:=FuncDesc, _
        Category:=FuncCat, _
        ArgumentDescriptions:=Array(Arg1Desc, Arg2Desc)
End Sub
```

На рис. 8 показаны диалоговые окна *Мастер функций* и *Аргументы функции* после выполнения процедуры `DescribeFunction()`.

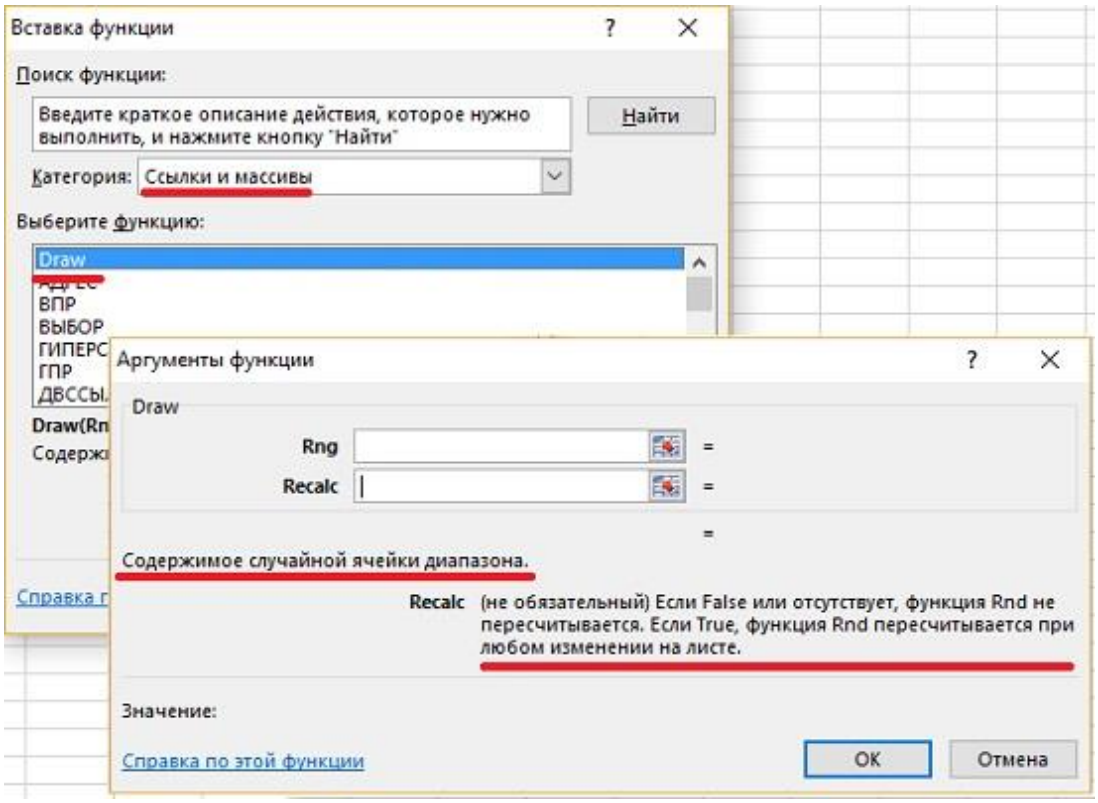


Рис. 8. Вид диалоговых окон *Мастер функций* и *Аргументы функции* для пользовательской функции

Процедуру `DescribeFunction()` следует вызывать только один раз. После ее вызова информация, связанная с функцией, сохраняется в рабочей книге. Но если вы модифицировали процедуру, повторите ее вызов.

Если вы не укажете категорию функции с помощью метода `MacroOptions`, пользовательская функция рабочего листа появится в категории *Определенные пользователем* диалогового окна *Мастер функций*. В таблице (рис. 9) перечислены номера категорий, которые можно использовать в качестве значений аргумента `Category` метода `MacroOptions`. Обратите внимание, что некоторые из этих категорий (от 10 до 13) обычно не отображаются в диалоговом окне *Мастер функций*. Если же отнести одну из пользовательских функций в подобную категорию, она появится в диалоговом окне.

Номер категории	Название категории
0	Полный алфавитный перечень (All)
1	Финансовые (Financial)
2	Дата и время (Date & Time)
3	Математические (Math & Trig)
4	Статистические (Statistical)
5	Ссылки и массивы (Lookup & Reference)
6	Работа с базой данных (Database)
7	Текстовые (Text)
8	Логические (Logical)
9	Проверка свойств и значений (Information)
12	Управление макросами (Macro Control)
13	Динамический обмен данными/Внешние (DDE/External)
14	Определенные пользователем (User Defined)
15	Инженерные (Enhineering)
16	Аналитические (Cube)
17	Совместимость (Compatibility)*

Рис. 9. Номера категорий функций

Использование надстроек для хранения пользовательских функций

При желании можно сохранить часто используемые пользовательские функции в файле надстройки. Основное преимущество такого подхода заключается в следующем: функции могут быть применены в формулах без спецификатора имени файла. Предположим, у вас есть пользовательская функция ZapSpaces; она хранится в файле Myfuncs.xlsm. Чтобы применить ее в формуле другой рабочей книги (отличной от Myfuncs.xlsm), необходимо ввести следующую формулу: =Myfuncs.xlsm!ZapSpaces(A1:C12).

Если вы создадите надстройку на основе файла Myfuncs.xlsm и эта надстройка будет загружена в текущем сеансе работы Excel, то ссылку на файл можно пропустить, введя следующую формулу: =ZapSpaces(A1:C12). Создание надстроек будет рассмотрено отдельно.

> *Потенциальная проблема, которая может возникнуть из-за использования надстроек для хранения пользовательских функций, связана с зависимостью рабочей книги от файла надстроек. Если вы передаете рабочую книгу сотруднику, не забудьте также передать копию надстройки, которая содержит требуемые функции.*

Использование функций Windows API

VBA может заимствовать методы из других файлов, которые не имеют ничего общего с Excel или VBA, например, файлы DLL (Dynamic Link Library — динамически подключаемая библиотека), которые используются Windows и другими программами. В результате в VBA появляется возможность выполнять операции, которые без заимствованных методов находятся за пределами возможностей языка.

Windows API (Application Programming Interface — интерфейс прикладного программирования) представляет собой набор функций, доступных программистам в среде Windows. При вызове функции Windows из VBA вы обращаетесь к Windows API. Многие ресурсы Windows, используемые программистами Windows, можно получить из файлов DLL, в которых хранятся программы и функции, подсоединяемые в процессе выполнения программы, а не во время компиляции.

Прежде чем использовать функцию Windows API, ее необходимо объявить вверху программного модуля. Если программный модуль — это не стандартный модуль VBA (т.е. модуль для *UserForm*, *Лист* или *ЭтаКнига*), то API-функцию необходимо объявить, как `Private`.

Объявление API-функции имеет некоторую сложность — функция должна объявляться максимально точно. Оператор объявления указывает VBA следующее:

- какую API-функцию вы используете;
- в какой библиотеке расположена API-функция;
- аргументы API-функции.

После объявления API-функцию можно использовать в программе VBA.

Рассмотрим пример API-функции, которая отображает имя папки Windows (с помощью стандартных операторов VBA эту задачу порой выполнить невозможно). Для начала объявим API-функцию:

```
Declare PtrSafe Function GetWindowsDirectoryA Lib "kernel32" _
      (ByVal lpBuffer As String, ByVal nSize As Long) As Long
```

Эта функция, имеющая два аргумента, возвращает название папки, в которой установлена операционная система Windows. После вызова этой функции путь к папке Windows будет храниться в переменной `lpBuffer`, а длина строки пути — в переменной `nSize`.

Следующий пример отображает результат в окне сообщения:

```
Sub ShowWindowsDir()
    Dim WinPath As String * 255
    Dim WinDir As String
    WinPath = Space(255)
    WinDir = Left(WinPath, GetWindowsDirectoryA _
      (WinPath, Len(WinPath)))
    MsgBox WinDir, vbInformation, "Windows Directory"
End Sub
```

В процессе выполнения процедуры ShowWindowsDir отображается окно сообщения с указанием расположения папки Windows.

Иногда требуется создать оболочку (wrapper) для API-функций. Другими словами, вы создадите собственную функцию, использующую API-функцию. Такой подход существенно упрощает использование API-функции. Ниже приведен пример такой функции VBA:

```
Function WindowsDir() As String
'   Название папки Windows
    Dim WinPath As String * 255
    WinPath = Space(255)
    WindowsDir = Left(WinPath, GetWindowsDirectoryA _
       (WinPath, Len(WinPath)))
End Function
```

После объявления этой функции можно вызвать ее из другой процедуры: `MsgBox WindowsDir()`. Можно также использовать эту функцию в формуле рабочего листа: =WindowsDir().

Внимание! Не удивляйтесь сбоям в системе при использовании в VBA функций Windows API. Заранее сохраните свою работу перед тестированием.

Определение состояния клавиши <Shift>

Предположим, вы написали макрос VBA, который будет выполняться с помощью кнопки на панели инструментов. Необходимо, чтобы этот макрос выполнялся по-другому, если пользователь после щелчка на кнопке удерживает клавишу <Shift>. Чтобы узнать о нажатии клавиши <Shift>, можно использовать API-функцию `GetKeyState`. Функция `GetKeyState` сообщает о том, нажата ли конкретная клавиша. Функция имеет один аргумент, `nVirtKey`, который представляет код интересующей вас клавиши.

Ниже приведена программа, которая выявляет, что при выполнении процедуры обработки события `Button_Click` была нажата клавиша <Shift>. Обратите внимание, что для определения состояния клавиши <Shift> используется константа (принимающая шестнадцатеричное значение), которая затем применяется как аргумент функции `GetKeyState`. Если `GetKeyState` возвращает значение меньше 0, это означает, что клавиша <Shift> нажата; в противном случае клавиша <Shift> не нажата. Аналогичную проверку можно устроить для клавиш Ctrl и Alt (рис. 10).

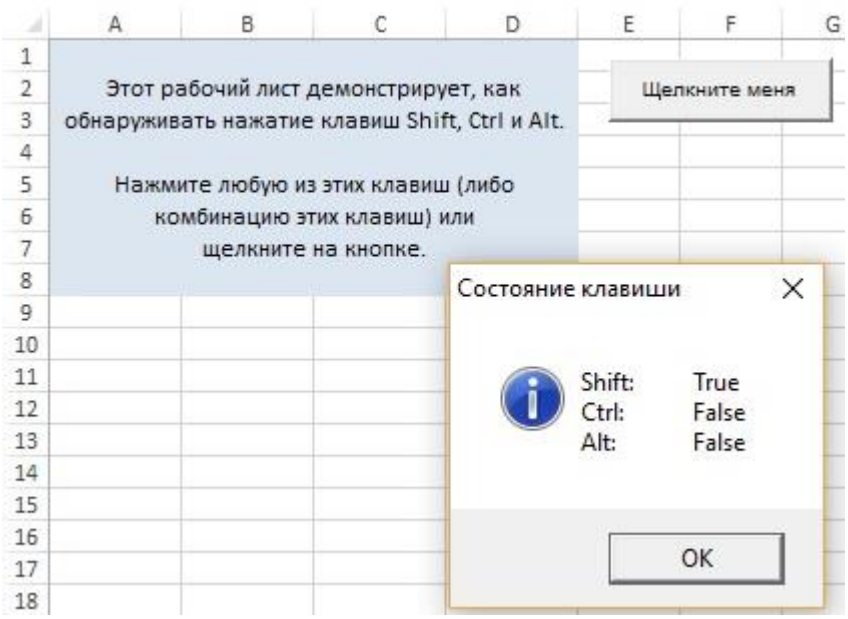


Рис. 10. Проверка нажатия клавиш Shift, Ctrl и Alt

Код функции VBA можно найти в приложенном Excel-файле

Работа с функциями Windows API может быть довольно сложной. Во многих книгах по программированию перечислены операторы объявления API-функций с соответствующими примерами. Как правило, можно просто скопировать выражения объявления и использовать функции, не вникая в их суть. Большинство VBA-программистов в Excel рассматривают API-функции как панацею для решения большинства задач. В Интернете вы найдете сотни вполне надежных примеров, которые можно скопировать и вставить в собственную программу.

> *В текстовом файле содержатся объявления и константы Windows API. Можно открыть этот файл в текстовом редакторе и скопировать соответствующие объявления в модуль VBA.*