Excel всемогущий

13

Первый принцип удачного программиста состоит в том, чтобы никогда не писать один и тот же код дважды. Второй принцип удачного программиста требует создавать код для каждой рутинной операции.

В настоящей главе рассматриваются макросы, присланные опытными пользователями Excel со всего мира. Эти программы не только помогут сэкономить время, но и прольют свет на новые способы решения наиболее распространенных задач.

Различные программисты придерживаются различных подходов к созданию программного кода. Наглядным подтверждением этого являются приведенные в данной главе макросы.

В ЭТОЙ ГЛАВЕ

Расширение возможностей Excel с помощью VBA

VBA позволяет раскрыть возможности Excel, недоступные посредством пользовательского интерфейса. В следующих разделах рассматривается использование VBA для реализации условного форматирования с более чем тремя условиями, и расширенного фильтра с более чем двумя условиями.

Условное форматирование с более чем тремя условиями

Макрос Worksheet_Change любезно предоставлен Расселом Гауфом (Russell Hauf), проживающим в Бивертоне, штат Орегон, США.

По сути, макрос Worksheet_Change дублирует функцию условного форматирования Excel. Увеличение максимального числа условий достигается за счет наличия управляющего листа, содержащего 2 столбца — допустимых значений ячеек (столбец A) и соответствующих индексов цвета заливки (столбец B).

```
Private Sub Worksheet_Change(ByVal Target As Range)
' Этот макрос реализует условное форматирование
' с более чем 3-мя условиями.
    Dim rng As Range
' Исходный диапазон может содержать более чем 1 ячейку,
' поскольку он задается как пересечение диапазонов Target
' и D:D. Благодаря этому макрос будет выполняться корректно
' при удалении содержимого нескольких ячеек столбца D или
' вводе значений нескольких ячеек столбца D в виде массива.
    Set rng = Intersect(Target, Range("D:D"))
    If rng Is Nothing Then
        Exit Sub
    Else
        Dim cl As Range
        For Each cl In rng
            On Error Resume Next
' Если содержимое ячейки отсутствует
' в диапазоне rngColors, ее заливка удаляется.
            cl.Interior.ColorIndex = _
            Application.WorksheetFunction.VLookup(cl.Value, _
ThisWorkbook.Sheets("УФ (управляющий лист)").Range("rngColors"), _
2, False)
            If Err.Number <> 0 Then
                cl.Interior.ColorIndex = xlNone
            End If
        Next cl
    End If
End Sub
```

Расширенный фильтр с более чем двумя условиями

Макрос MultiFilter любезно предоставлен Ричи Силлсом (Richie Sills), проживающим в Вустере, Англия. Ричи работает консультантом по налогам в аудиторской компании.

Стандартный расширенный фильтр Excel позволяет задавать не более двух условий. Следующий макрос позволяет обойти это ограничение.

```
Sub MultiFilter()
' Обратите внимание, что Ричи размещает комментарии после кода.
' Другими словами, комментарий на строке 20 относится к коду
' на строке 19.
    Dim rngTarget As Range, rng1 As Range, rng2 As Range, _
rngMyRange As Range
    Const Crit1 As String = "DEF, LLC"
```

```
    Const Crit2 As String = "FGH LTD."
    Const Crit3 As String = "QRS INC."
' Критерии расширенного фильтра.

    Application.ScreenUpdating = False
    With Worksheets("Расширенный фильтр")
        .Rows(1).Insert
        .Range("A1").Value = "dummy"
' Создать заголовок "dummy".

        Set rngTarget = .Range("A1:A" & .Cells(Rows.Count, _
1).End(xlUp).Row)
' Задать исходный диапазон.

        rngTarget.AutoFilter Field:=1, Criteria1:=Crit1, _
Operator:=xlOr, Criteria2:=Crit2
' Применить стандартный расширенный фильтр с 2-мя критериями.

        Set rng1 = .AutoFilter.Range.Offset(1, 0).Resize( _
.AutoFilter.Range.Rows.Count - 1).SpecialCells(xlCellTypeVisible)
' Создать диапазон ячеек, ссылающийся
' на результат применения расширенного фильтра.

        rngTarget.AutoFilter
' Вернуться к исходным данным.

        rngTarget.AutoFilter Field:=1, Criteria1:=Crit3
' Применить стандартный расширенный фильтр с 3-м критерием.

        Set rng2 = .AutoFilter.Range.Offset(1, 0).Resize( _
.AutoFilter.Range.Rows.Count - 1).SpecialCells(xlCellTypeVisible)
' Создать диапазон ячеек, ссылающийся
' на результат применения расширенного фильтра.

        rngTarget.AutoFilter
        .Rows(1).Delete
' Удалить заголовок "dummy".

        Set rngMyRange = Union(rng1, rng2)
' Объединить диапазоны ячеек rng1 и rng2.

        rngMyRange.EntireRow.Copy Destination:=Worksheets( _
"РФ (результат)").Range("A2")
' Скопировать полученный результат в рабочий лист "РФ (результат)".
    End With
    Worksheets("РФ (результат)").Select
    Application.ScreenUpdating = True
    MsgBox "Фильтр успешно применен"
End Sub
```

Файловые операции

Макросы, приведенные в следующих разделах, используются для выполнения различных файловых операций.

Поиск файлов

Макрос `Srch` и функция `BrowseForFolderShell` любезно предоставлены Натаном П. Оливером (Nathan P. Oliver), проживающим в Миннеаполисе, штат Миннесота, США. Натан занимает должности финансового консультанта и разработчика приложений.

Следующий макрос проводит поиск файлов, в имени которых встречается заданная строка, в указанной папке и ее подпапках. По результатам поиска создается список, в котором для каждого найденного файла выводится ссылка на файл, размер файла и дата его последнего изменения.

```
Sub Srch()
    Dim i As Long, z As Long, ws As Worksheet, y As Variant
    Dim fLdr As String
    y = Application.InputBox("Пожалуйста, введите строку, _
встречающуюся в имени файла", "Ввод информации")
    If y = False And Not TypeName(y) = "String" Then Exit Sub
    Application.ScreenUpdating = False
    fLdr = BrowseForFolderShell
    With Application.FileSearch
        .NewSearch
        .LookIn = fLdr
        .SearchSubFolders = True
        .Filename = y
        Set ws = ThisWorkbook.Worksheets.Add(Sheets(1))
        On Error GoTo 1
2:      ws.Name = "Результат поиска файлов"
        On Error GoTo 0
        If .Execute() > 0 Then
            For i = 1 To .FoundFiles.Count
                If Left$(.FoundFiles(i), 1) = Left$(fLdr, 1) Then
                    If CBool(Len(Dir(.FoundFiles(i)))) Then
                        z = z + 1
                        ws.Cells(z + 1, 1).Resize(, 3) = _
                        Array(Dir(.FoundFiles(i)), _
                        FileLen(.FoundFiles(i)) \ 1000, _
                        FileDateTime(.FoundFiles(i)))
                        ws.Hyperlinks.Add Anchor:=Cells(z + 1, _
1), Address:=.FoundFiles(i)
                    End If
                End If
            Next i
        End If
    End With

    ActiveWindow.DisplayHeadings = False

    With ws
        With .[a1:c1 ]
            .Value = [{"Ссылка на файл","Размер (Kb)","Дата _
последнего изменения"}]
            .Font.Underline = xlUnderlineStyleSingle
            .EntireColumn.AutoFit
            .HorizontalAlignment = xlCenter
        End With
```

```
        .[d1:iv1 ].EntireColumn.Hidden = True
        Range(.[a65536 ].End(3)(2), _
        .[a65536 ]).EntireRow.Hidden = True
        Range(.[a2 ], .[c65536 ]).Sort [a2 ], xlAscending, _
Header:=xlNo
    End With

    Application.ScreenUpdating = True
    Exit Sub

1:  Application.DisplayAlerts = False
    Worksheets("Результат поиска файлов").Delete
    Application.DisplayAlerts = True
    GoTo 2
End Sub

Function BrowseForFolderShell() As String
    Dim objShell As Object, objFolder As Object
    Set objShell = CreateObject("Shell.Application")
' Расскомментируйте следующую строку, чтобы начать
' обзор папок с рабочего стола Windows.
    'Set objFolder =objShell.BrowseForFolder(0,"Пожалуйста, _
выберите папку", 0, 0)

' Укажите папку, с которой нужно начать обзор.
    Set objFolder = objShell.BrowseForFolder(0, "Пожалуйста, _
выберите папку", 0, "c:\")
    If (Not objFolder Is Nothing) Then
        On Error Resume Next
        If IsError(objFolder.Items.Item.Path) Then _
BrowseForFolderShell = CStr(objFolder): GoTo Here
            On Error GoTo 0
            If Len(objFolder.Items.Item.Path) > 3 Then
                BrowseForFolderShell = objFolder.Items.Item.Path _
& Application.PathSeparator
            Else
                BrowseForFolderShell = objFolder.Items.Item.Path
            End If
        Else: Application.ScreenUpdating = True: End
    End If

Here:
    Set objFolder = Nothing: Set objShell = Nothing
End Function
```

Удаление рабочей книги после определенной даты

Макрос `Workbook_Open` любезно предоставлен Томом Уртисом (Tom Urtis), проживающим в Сан-Франциско, штат Калифорния, США. Том — глава консалтинговой компании Atlas Programming Management, расположенной в районе Залива.

Приведенный ниже макрос удаляет активную рабочую книгу после 31 декабря 2004 года.

Внимание

Файл рабочей книги *не* помещается в Корзину (Recycle Bin), а навсегда удаляется с компьютера.

```
Sub Workbook_Open()
    If Date <= #12/31/2004# Then Exit Sub
    MsgBox "Сейчас рабочая книга будет удалена!"
    With ThisWorkbook
        .Saved = True
        .ChangeFileAccess xlReadOnly
        Kill .FullName
        .Close False
    End With
End Sub
```

Создание команды меню "Закрыть и удалить"

Макросы `Workbook_AddinInstall`, `Workbook_AddinnUninstall` и `CloseAndKill` любезно предоставлены Томми Майлзом (Tommy Miles), проживающем в Хьюстоне, штат Техас, США.

Одной из наиболее распространенных операций, выполняемых в Excel, является открытие временного файла, копирование из него нужной информации и закрытие этого файла. Подобные файлы "разового использования" имеют свойство накапливаться, захламляя пространство жесткого диска компьютера.

Макрос `Workbook_AddinInstall` добавляет к меню Excel Файл (File) команду Закрыть и удалить (рис. 13.1), выполнение которой приводит к закрытию активной рабочей книги и удалению ее файла. Помимо этого, имя файла активной рабочей книги удаляется из списка ранее открывавшихся файлов.

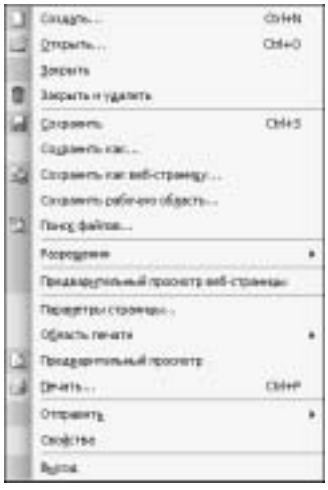


Рис. 13.1. Команда Закрыть и удалить используется для закрытия активной рабочей книги и удаления ее файла

```
Public Const CONTROLNAME As String = "Закрыть и у&далить"

Sub Workbook_AddinInstall()
    Dim cmdControl As CommandBarButton
    On Error Resume Next
    Set cmdControl =
Application.CommandBars(1).Controls("Файл").Controls(CONTROLNAME)
    If cmdControl Is Nothing Then
        Set cmdControl =
Application.CommandBars(1).Controls("Файл").Controls.Add( _
Type:=msoControlButton, Before:=Application.CommandBars( _
1).Controls("Файл").Controls("Сохранить").Index)
        With cmdControl
            .Caption = CONTROLNAME
            .FaceId = 67
            .Style = msoButtonIconAndCaption
            .DescriptionText = "Закрыть активную рабочую книгу _
и удалить ее файл"
            .OnAction = "CloseAndKill"
        End With
        Set cmdControl = Nothing
    End If
    On Error GoTo 0
    MsgBox "Команда ""Закрыть и удалить"" доступна из меню ""Файл"""
End Sub

Sub Workbook_AddinUninstall()
    On Error Resume Next
    Application.CommandBars(1).Controls("Файл").Controls( _
CONTROLNAME).Delete
    MsgBox "Команда ""Закрыть и удалить"" успешно удалена _
из меню ""Файл"""
End Sub

Sub CloseAndKill()
    Dim tmpAnswer As Variant
    If ActiveWorkbook Is Nothing Then Exit Sub

    tmpAnswer = MsgBox("Вы действительно хотите удалить _
рабочую книгу " & ActiveWorkbook.FullName & """?", _
vbYesNoCancel + vbInformation)
    If tmpAnswer = vbYes Then
        Dim tmpFileName As String
        Dim RecentFle As RecentFile
        tmpFileName = ActiveWorkbook.FullName
' Удалить имя файла активной рабочей книги из списка
' ранее открывавшихся файлов.
        For Each RecentFle In Application.RecentFiles
            If RecentFle.Path = tmpFileName Then RecentFle.Delete
        Next
        ActiveWorkbook.Close SaveChanges:=False
        On Error Resume Next
        Kill tmpFileName ' Удалить файл.
        If Err.Number <> 0 Then
            MsgBox "Невозможно удалить файл """ & tmpFileName &
""""."
        End If
    End If
End Sub
```

Импорт CSV-файлов

Макрос `OpenLargeCSVFast` любезно предоставлен Масару Каджи (Masaru Kaji), проживающим в Кобе-Сити, Япония. Масару автор Web-сайта *Colo's Excel Junk Room* (`http://www.puremis.net/excel/`), посвященного Excel и VBA.

Следующий макрос импортирует CSV-файл в Excel, а затем удаляет его.

```
Option Base 1

Sub OpenLargeCSVFast()
    Dim buf(1 To 256) As Variant
    Dim i As Long
    Const strFilePath As String = "C:\temp\Test.CSV" 'Подставьте _
сюда путь к нужному файлу.
    Dim strRenamedPath As String
    strRenamedPath = Split(strFilePath, ".")(0) & "txt"
    With Application
        .ScreenUpdating = False
        .DisplayAlerts = False
    End With

'Создание массива для параметра FieldInfo метода OpenText.
    For i = 1 To 256
        buf(i) = Array(i, 2)
    Next

    Name strFilePath As strRenamedPath
    Workbooks.OpenText Filename:=strRenamedPath, _
DataType:=xlDelimited, Comma:=True, FieldInfo:=buf
    Erase buf
    ActiveSheet.UsedRange.Copy ThisWorkbook.Sheets( _
"CSV").Range("A5")
    ActiveWorkbook.Close False
    Kill strRenamedPath
    With Application
        .ScreenUpdating = True
        .DisplayAlerts = True
    End With
End Sub
```

Считывание текстового файла в память и его последующий анализ

Макрос `ReadTxtLines` любезно предоставлен Суатом Мехметом Озгуром (Suat Mehmet Ozgur), проживающим в Стамбуле, Турция. Суат разрабатывает Excel-, Access- и Visual Basic-приложения для компаний MrExcel Consulting и TheOfficeExperts.com.

Следующий макрос реализует весьма необычный подход к считыванию содержимого текстового файла. Вместо построчного чтения, макрос загружает весь файл в память и сохраняет его в строковой переменной. Затем полученная переменная разбивается на строки. Преимущество этого метода заключа-

ется в единственном обращении к файлу на диске. Последующая обработка файла осуществляется в памяти компьютера.

```
Sub ReadTxtLines()
'Нет необходимости устанавливать библиотеку
'Microsoft Scripting Runtime, поскольку
'в этом макросе используется позднее связывание.
    Dim sht As Worksheet
    Dim fso As Object
    Dim fil As Object
    Dim txt As Object
    Dim strtxt As String
    Dim tmpLoc As Long

'Работаем с активным рабочим листом.
    Set sht = ActiveSheet

'Очистить содержимое активного рабочего листа.
    sht.UsedRange.ClearContents

'Создать объект файловой системы.
    Set fso = CreateObject("Scripting.FileSystemObject")

'Создать объект требуемого файла.
    Set fil = fso.GetFile("c:\test.csv")

'Открыть файл как текст.
    Set txt = fil.OpenAsTextStream(1)

'Считать содержимое файла в строковую переменную.
    strtxt = txt.ReadAll

'Закрыть файл.
    txt.Close

'Найти позицию 1-го символа новой строки.
    tmpLoc = InStr(1, strtxt, vbCrLf)

'Выполнять до тех пор, пока в переменной strtxt
'будут оставаться символы новой строки.
    Do Until tmpLoc = 0
'Сохранить строку в следующей пустой ячейке столбца A.
        sht.Cells(65536, 1).End(xlUp).Offset(1).Value = _
Left(strtxt, tmpLoc - 1)
' Удалить "отработанную" строку из переменной.
        strtxt = Right(strtxt, Len(strtxt) - tmpLoc - 1)
'Найти позицию следующего символа новой строки.
        tmpLoc = InStr(1, strtxt, vbCrLf)
    Loop
'Последняя строка не содержит символа новой строки.
    sht.Cells(65536, 1).End(xlUp).Offset(1).Value = strtxt
'Несмотря на то что файл уже закрыт, правила "хорошего тона"
'требуют установить значение переменной fso равным Nothing.
    Set fso = Nothing
End Sub
```

Объединение и разделение рабочих книг

Следующие 4 макроса демонстрируют возможность объединения нескольких рабочих книг в одну, а также сохранения листов рабочей книги в виде отдельных рабочих книг или документов Word.

Сохранение листов рабочей книги в виде отдельных рабочих книг

Макрос `SplitWorkbook` любезно предоставлен Томми Майлзом (Tommy Miles), проживающим в Хьюстоне, штат Техас, США.

Следующий макрос сохраняет все листы активной рабочей книги в виде отдельных рабочих книг, файлы которых носят имена соответствующих рабочих листов и размещаются в той же папке, что и исходная рабочая книга. При попытке перезаписать существующий файл выводится предупреждение.

```
Sub SplitWorkbook()
    Dim ws As Worksheet
    Dim DisplayStatusBar As Boolean
    DisplayStatusBar = Application.DisplayStatusBar
    Application.DisplayStatusBar = True
    Application.ScreenUpdating = False

    For Each ws In ThisWorkbook.Sheets
        Dim NewFileName As String
        Application.StatusBar = "Осталось рабочих листов: " & _
ThisWorkbook.Sheets.Count
        If ThisWorkbook.Sheets.Count <> 1 Then
            NewFileName = ThisWorkbook.Path & "\" & ws.Name & _
".xls"
            ws.Copy
            ActiveWorkbook.Sheets(1).Name = "Sheet1"
            ActiveWorkbook.SaveAs Filename:=NewFileName
            ActiveWorkbook.Close SaveChanges:=False
        Else
            NewFileName = ThisWorkbook.Path & "\" & ws.Name & _
".xls"
            ws.Name = "Sheet1"
            ThisWorkbook.SaveAs Filename:=NewFileName
        End If
    Next
    Application.StatusBar = False
    Application.DisplayStatusBar = DisplayStatusBar
    Application.ScreenUpdating = True
End Sub
```

Объединение нескольких рабочих книг в одну

Макрос `CombineWorkbooks` также любезно предоставлен Томми Майлзом.

Следующий макрос объединяет все рабочие книги, расположенные в заданной в папке, в одну. Рабочие листы полученной книги будут названы по именам соответствующих исходных рабочих книг.

```
Sub CombineWorkbooks()
    Dim CurFile As String

    Dim DestWB As Workbook
    Dim ws As Object 'Рабочие листы могут быть произвольного типа.

    Const DirLoc As String = "C:\Data\" 'Местоположение _
исходных файлов.

    Application.ScreenUpdating = False

    Set DestWB = Workbooks.Add(xlWorksheet)

    CurFile = Dir(DirLoc & "*.xls")

    Do While CurFile <> vbNullString
        Dim OrigWB As Workbook

        Set OrigWB = Workbooks.Open(Filename:=DirLoc & CurFile, _
ReadOnly:=True)

        CurFile = Left(Left(CurFile, Len(CurFile) - 4), 29)
                                'Получение базового имени
                                'рабочего листа путем отсечения
                                'последних 4-х символов имени
                                'исходного файла (".xls").

        For Each ws In OrigWB.Sheets
            ws.Copy After:=DestWB.Sheets(DestWB.Sheets.Count)

            If OrigWB.Sheets.Count > 1 Then
                DestWB.Sheets(DestWB.Sheets.Count).Name = _
CurFile & ws.Index
            Else
                DestWB.Sheets(DestWB.Sheets.Count).Name = CurFile
            End If
        Next

        OrigWB.Close SaveChanges:=False

        CurFile = Dir
    Loop

    Application.DisplayAlerts = False
        DestWB.Sheets(1).Delete
    Application.DisplayAlerts = True

    Application.ScreenUpdating = True

    Set DestWB = Nothing
End Sub
```

Фильтрация данных с последующим копированием полученного результата в отдельные рабочие листы

Макрос `Filter_NewSheet` любезно предоставлен Деннисом Валентайном (Dennis Wallentin), проживающим в Остерсунде, Швеция. Деннис дает советы, касающиеся использования Excel и VBA, на своем собственном Web-сайте по адресу: `www.xldennis.com`.

Следующий макрос фильтрует исходные данные (рис. 13.2) и копирует полученные результаты в отдельные рабочие листы (рис. 13.3).

	A	B	C
1	Статья	Категория	Номер
2	AA1	A	76
3	AA1	B	23
4	AA1	C	80
5	AA2	A	25
6	AA2	B	43
7	AA2	C	54
8	AA3	A	40
9	AA3	B	54
10	AA3	C	21
11	AA4	A	10
12	AA4	B	90
13	AA4	C	20
14	AA5	A	25
15	AA5	B	30
16	AA5	C	40

Рис. 13.2. Исходные данные

	A	B	C	D
1				
2	AA5	A	25	
3	AA5	B	30	
4	AA5	C	40	
5				

Рис. 13.3. Результат применения фильтра скопирован в новый рабочий лист

```
Sub Filter_NewSheet()
Dim wbBook As Workbook
Dim wsSheet As Worksheet
Dim rnStart As Range, rnData As Range
Dim i As Long

Set wbBook = ThisWorkbook
Set wsSheet = wbBook.Worksheets("Фильтр и копирование")

With wsSheet
'Убедитесь, что 1-я строка содержит заголовки столбцов.
    Set rnStart = .Range("A2")
    Set rnData = .Range(.Range("A2"), .Range("C65536").End(xlUp))
End With

Application.ScreenUpdating = True

For i = 1 To 5
'Применение расширенного фильтра.
    rnStart.AutoFilter Field:=1, Criteria1:="AA" & i
'Копирование результата фильтрации.
    rnData.SpecialCells(xlCellTypeVisible).Copy
'Добавление нового рабочего листа.
    Worksheets.Add Before:=wsSheet
'Присвоение имени новому рабочему листу.
    ActiveSheet.Name = "AA" & i
'Вставка результата фильтрации
'в новый рабочий лист.
    Range("A2").PasteSpecial xlPasteValues
Next i

'Вернуться к исходным данным.
rnStart.AutoFilter Field:=1
```

```
With Application
'Очистить буфер обмена.
    .CutCopyMode = False
    .ScreenUpdating = False
End With

End Sub
```

Экспорт данных в Word

Макрос `Export_Data_Word_Table` также любезно предоставлен Деннисом Валентайном.

Следующий макрос экспортирует данные с рабочего листа Excel в документ Word. Поскольку используется раннее связывание, необходимо добавить ссылку (команда меню редактора Visual Basic Tools⇨References (Сервис⇨Ссылки)) на библиотеку Microsoft Word Object Library.

```
Sub Export_Data_Word_Table()
Dim wdApp As Word.Application
Dim wdDoc As Word.Document
Dim wdCell As Word.Cell
Dim i As Long
Dim wbBook As Workbook
Dim wsSheet As Worksheet
Dim rnData As Range
Dim vaData As Variant

Set wbBook = ThisWorkbook
Set wsSheet = wbBook.Worksheets("Экспорт в Word")

With wsSheet
    Set rnData = .Range("A1:A10")
End With

'Поместить данные из диапазона A1:A10 в одномерный массив Variant.
vaData = rnData.Value

'Создать объект Word.
Set wdApp = New Word.Application
'Документ Test.doc должен находиться в той же папке,
'что и рабочая книга.
Set wdDoc = wdApp.Documents.Open(ThisWorkbook.Path & "\Test.doc")

'Импортировать данные в 1-й столбец 1-й таблицы.
For Each wdCell In wdDoc.Tables(1).Columns(1).Cells
    i = i + 1
    wdCell.Range.Text = vaData(i, 1)
Next wdCell

'Сохранить и закрыть документ.
With wdDoc
    .Save
    .Close
End With
```

```
'Завершить работу скрытой копии Microsoft Word.
wdApp.Quit
'Удалить внешние переменные из памяти.
Set wdDoc = Nothing
Set wdApp = Nothing

MsgBox "Данные были успешно экспортированы в документ Test.doc.", _
vbInformation

End Sub
```

Работа с примечаниями

В большинстве случаев примечания ячеек Excel используются не достаточно эффективно. Рассматриваемые в следующих разделах макросы помогут исправить это упущение.

Вывод примечаний

Макрос `ListComments` любезно предоставлен Томми Майлзом.

Excel позволяет печатать примечания, однако не позволяет выводить информацию о рабочей книге и (или) рабочем листе, к которым относится то или иное примечание. Следующий макрос помещает примечания, имена их авторов и сведения о местоположении в новую рабочую книгу для последующего просмотра, сохранения и (или) печати.

```
Sub ListComments()
    Dim wb As Workbook
    Dim ws As Worksheet

    Dim cmt As Comment

    Dim cmtCount As Long

    cmtCount = 2

    On Error Resume Next
        Set ws = ActiveSheet
            If ws Is Nothing Then Exit Sub
    On Error GoTo 0

    Application.ScreenUpdating = False

    Set wb = Workbooks.Add(xlWorksheet)

    With wb.Sheets(1)
        .Range("$A$1") = "Автор"
        .Range("$B$1") = "Рабочая книга"
        .Range("$C$1") = "Рабочий лист"
        .Range("$D$1") = "Диапазон"
        .Range("$E$1") = "Примечание"
    End With

    For Each cmt In ws.Comments
```

```
        With wb.Sheets(1)
            .Cells(cmtCount, 1) = cmt.author
            .Cells(cmtCount, 2) = cmt.Parent.Parent.Parent.Name
            .Cells(cmtCount, 3) = cmt.Parent.Parent.Name
            .Cells(cmtCount, 4) = cmt.Parent.Address
            .Cells(cmtCount, 5) = CleanComment(cmt.author, _
cmt.Text)
        End With

        cmtCount = cmtCount + 1
    Next

    wb.Sheets(1).UsedRange.WrapText = False

    Application.ScreenUpdating = True

    Set ws = Nothing
    Set wb = Nothing
End Sub

Private Function CleanComment(author As String, _
cmt As String) As String
    Dim tmp As String

    tmp = Application.WorksheetFunction.Substitute(cmt, _
author & ":", "")
    tmp = Application.WorksheetFunction.Substitute(tmp, _
Chr(10), "")

    CleanComment = tmp
End Function
```

Результат выполнения макроса `ListComments` показан на рис. 13.4.

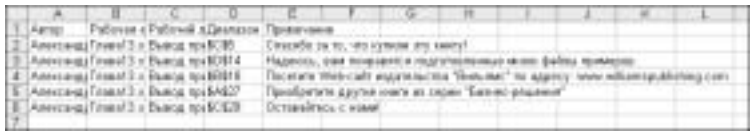


Рис. 13.4. Макрос `ListComments` позволяет получить исчерпывающую информацию о примечаниях

Изменение размера области примечания

Макрос `CommentFitter1` любезно предоставлен Томом Уртисом.

Следующий макрос изменяет размер области примечания так, чтобы она вместила в себя весь текст примечания.

```
Sub CommentFitter1()
Application.ScreenUpdating = False
Dim x As Range, y As Long

For Each x In Cells.SpecialCells(xlCellTypeComments)
    Select Case True
        Case Len(x.NoteText) <> 0
            With x.Comment
```

```
                .Shape.TextFrame.AutoSize = True
                If .Shape.Width > 250 Then
                    y = .Shape.Width * .Shape.Height
                    .Shape.Width = 150
                    .Shape.Height = (y / 200) * 1.2
                End If
            End With
    End Select
Next x
Application.ScreenUpdating = True
End Sub
```

Результат выполнения макроса `CommentFitter1` показан на рис. 13.5.

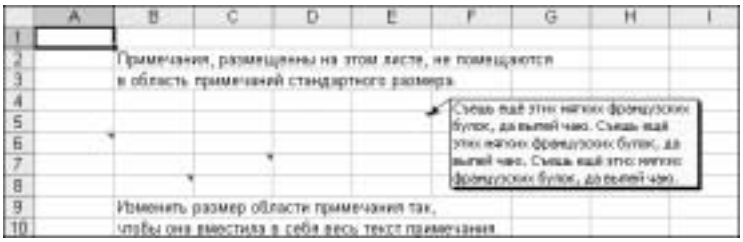


Рис. 13.5. Теперь область примечания включает в себя весь его текст

Изменение размера области примечания с помощью центрирования

Макрос `CommentFitter2` также любезно предоставлен Томом Уртисом.

Следующий макрос изменяет размер области примечания путем центрирования текста примечания.

```
Sub CommentFitter2()
Application.ScreenUpdating = False
Dim x As Range, y As Long

For Each x In Cells.SpecialCells(xlCellTypeComments)
    Select Case True
        Case Len(x.NoteText) <> 0
            With x.Comment
                .Shape.TextFrame.AutoSize = True
                If .Shape.Width > 250 Then
                    y = .Shape.Width * .Shape.Height
                    .Shape.ScaleHeight 0.9, msoFalse, _
msoScaleFromTopLeft
                    .Shape.ScaleWidth 1#, msoFalse, _
msoScaleFromTopLeft
                End If
            End With
    End Select
Next x
Application.ScreenUpdating = True
End Sub
```

Результат выполнения макроса `CommentFitter2` показан на рис. 13.6.

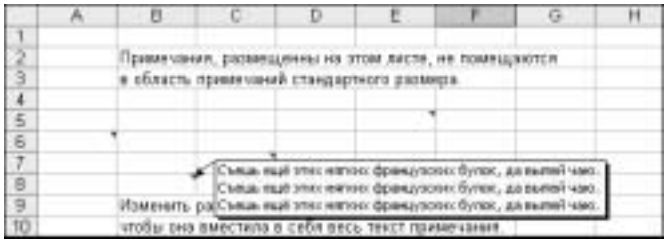


Рис. 13.6. Результат центрирования текста примечания

Размещение диаграммы в примечании

Макрос `PlaceGraph` любезно предоставлен Томом Уртисом.

Несмотря на то что Excel не позволяет разместить в примечании "настоящую" диаграмму, это можно сделать с ее изображением, как показано на рис. 13.7.

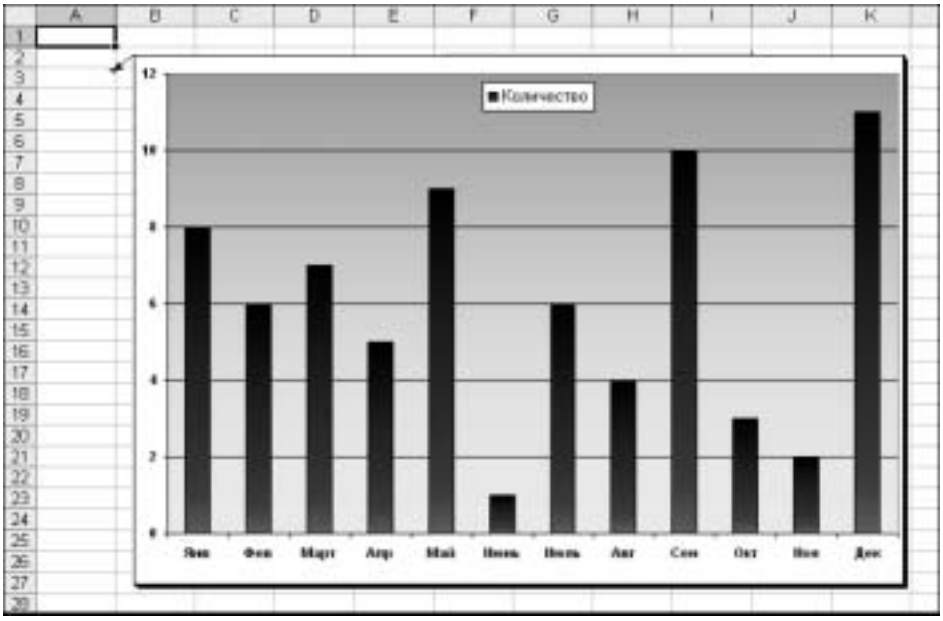


Рис. 13.7. "Диаграмма", размещенная в примечании

Чтобы поместить изображение диаграммы в примечание с помощью пользовательского интерфейса Excel, выполните следующие действия.

1. Создайте требуемое изображение.
2. Создайте примечание и выделите соответствующую ячейку.

3. Выберите команду меню Excel Вставка⇨Изменить примечание (Insert⇨ Edit Comment) или щелкните на ячейке правой кнопкой мыши и выберите команду контекстного меню Изменить примечание (Edit Comment).
4. Щелкните правой кнопкой мыши на границе области примечания и выберите команду контекстного меню Формат примечания (Format Comment).
5. Перейдите во вкладку Цвета и линии (Colors and Lines) и раскройте список Цвет (Color), расположенный в области Заливка (Fill).
6. Выберите команду Способы заливки (Fill Effects), перейдите во вкладку Рисунок (Picture) открывшегося диалогового окна Способы заливки (Fill Effects) и щелкните на кнопке Рисунок (Picture).
7. Выберите требуемое изображение и щелкните на кнопке OK для закрытия диалогового окна Способы заливки. Еще раз щелкните на кнопке OK, для того чтобы закрыть диалоговое окно Формат примечания.

Чтобы "диаграмма", помещенная в примечание описанным выше образом, всегда содержала текущие сведения, включите следующий код VBA в обработчик события SheetChange, срабатывающего при изменении исходных данных диаграммы (укажите требуемое имя файла изображения, название диаграммы, название рабочего листа, адрес ячейки и размер области примечания).

```
Sub PlaceGraph()
Dim x As String, z As Range

Application.ScreenUpdating = False
x = "C:\XWMJGraph.gif"

Set z = Worksheets("Диаграмма в примечании").Range("A3")

On Error Resume Next
z.Comment.Delete
On Error GoTo 0

ActiveSheet.ChartObjects("Chart 1").Activate
ActiveChart.Export x

With z.AddComment
    With .Shape
        .Height = 322
        .Width = 465
        .Fill.UserPicture x
    End With
End With

Range("A1").Activate
Application.ScreenUpdating = True

Set z = Nothing
End Sub
```

Замечательные возможности Excel VBA

В следующих разделах рассматриваются макросы, демонстрирующие богатые возможности Excel VBA.

Выделение ячейки с помощью условного форматирования

Макрос `Worksheet_SelectionChange` любезно предоставлен Иваном Ф. Моалой (Ivan F. Moala), проживающим в Окленде, Новая Зеландия. Иван — автор Web-сайта *The XcelFiles* (`www.xcelfiles.com`), посвященного задачам, решить которые с помощью Excel представляется невероятным или невозможным.

Следующий макрос выделяет активную ячейку с помощью условного форматирования. Следует отметить, что макрос `Worksheet_SelectionChange` удаляет любое существующее условное форматирование рабочего листа, а также очищает буфер обмена, что приводит к затруднению выполнения операций вырезания, копирования и вставки данных.

```
'////////////////////////////////////////////////////
'// Исправлено 14 февраля 2003 - с комментариями
'// Хуана Пабло Г. (Juan Pablo G.).
'// Локализованные версии Excel могут неправильно
'// интерпретировать значение TRUE.
'// Воспользуемся фактом, что TRUE можно
'// заменить любым целым числом, не равным 0.
'////////////////////////////////////////////////////

Const iInternational As Integer = Not (0)

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
Dim iColor As Integer
'// Примечание: этот макрос удаляет любое
'// существующее условное форматирование!

'// Продолжить выполнение макроса, если
'// пользователь выделит диапазон ячеек.
On Error Resume Next
iColor = Target.Interior.ColorIndex
'// Оставить действие строки On Error Resume Next
'// в силе на случай возникновения ошибок смещения строк.

If iColor < 0 Then
    iColor = 36
Else
    iColor = iColor + 1
End If

'// Проверить, совпадает ли цвет текста с цветом заливки.
If iColor = Target.Font.ColorIndex Then iColor = iColor + 1

Cells.FormatConditions.Delete

'// Выделение с помощью горизонтальной полосы.
With Range("A" & Target.Row, Target.Address) 'Rows(Target.Row)
    .FormatConditions.Add Type:=2, Formula1:=iInternational ' "TRUE"
    .FormatConditions(1).Interior.ColorIndex = iColor
```

```
End With

'// Выделение с помощью вертикальной полосы.
With Range(Target.Offset(1 - Target.Row, 0).Address & ":" & _
Target.Offset(-1, 0).Address)
    .FormatConditions.Add Type:=2, Formula1:=iInternational ' "TRUE"
    .FormatConditions(1).Interior.ColorIndex = iColor
End With

End Sub
```

Выделение ячейки без применения условного форматирования

Макрос `HighLight` и вспомогательные макросы `HighlightRight`, `HighlightLeft`, `HighlightUp`, `HighlightDown`, `DisableDelete` и `ReSet` также любезно предоставлены Иваном Ф. Моалой.

Следующий макрос выделяет ячейку, которая была сделана активной в результате использования клавиш управления курсором, без применения условного форматирования.

```
'// Альтернативный вариант макроса, выделяющего активную ячейку:
'// - не использует условное форматирование;
'// - сохраняет существующее условное форматирование рабочего листа;
'// - позволяет выполнять операции вырезания,
'//   копирования и вставки данных.
'//
'// Макрос создан Нейтом Оливером (Nate Oliver) для Альдо (Aldo).
'//
'// Альдо предложил обработку нажатия клавиши <Del>.
'//
'// Спасибо, друзья!!
'//
'// См. дополнительные комментарии в коде.

'// Размещено в стандартном модуле.

Dim strCol As String
Dim iCol As Integer
Dim dblRow As Double

Sub HighlightRight()
    HighLight 0, 1
End Sub

Sub HighlightLeft()
    HighLight 0, -1
End Sub

Sub HighlightUp()
    HighLight -1, 0, -1
End Sub

Sub HighlightDown()
    HighLight 1, 0, 1
End Sub

Sub HighLight(dblxRow As Double, iyCol As Integer, _
```

```
Optional dblZ As Double = 0)

On Error GoTo NoGo
strCol = Mid(ActiveCell.Offset(dblxRow, iyCol).Address, _
InStr(ActiveCell.Offset(dblxRow, iyCol).Address, "$") + 1, _
InStr(2, ActiveCell.Offset(dblxRow, iyCol).Address, "$") - 2)
iCol = ActiveCell.Column
dblRow = ActiveCell.Row

'// Позволяет избежать мерцания экрана.
Application.ScreenUpdating = False

With Range(strCol & ":" & strCol & "," & dblRow + dblZ & _
":" & dblRow + dblZ)
    .Select
'// Сейчас экран следует обновить.
    Application.ScreenUpdating = True
    .Item(dblRow + dblxRow).Activate
End With

NoGo:
End Sub

Sub DisableDelete()
    Cells(ActiveCell.Row, ActiveCell.Column).Select
    Application.OnKey "{DEL}"
End Sub

Sub ReSet()
    Application.OnKey "{RIGHT}"
    Application.OnKey "{LEFT}"
    Application.OnKey "{UP}"
    Application.OnKey "{DOWN}"
End Sub
```

Транспонирование данных

Макрос `TransposeData` любезно предоставлен Масару Каджи.

Следующий макрос транспонирует данные по заданному столбцу.

```
Sub TransposeData()
    Dim shOrg As Worksheet, shRes As Worksheet
    Dim rngStart As Range, rngPaste As Range
    Dim lngData As Long

    Application.ScreenUpdating = False
    On Error Resume Next
    Application.DisplayAlerts = False
    Sheets("Транспонирование (результат)").Delete
    Application.DisplayAlerts = True
    On Error GoTo 0

    On Error GoTo terminate

    Set shOrg = Sheets("Транспонирование")
    Set shRes = Sheets.Add(After:=shOrg)
    shRes.Name = "Транспонирование (результат)"
    With shOrg
'Отсортировать.
        .Cells.CurrentRegion.Sort Key1:=.[B2], Order1:=1, _
```

```
Key2:=.[C2], Order2:=1, Key3:=.[E2], Order3:=1, Header:=xlYes
'Скопировать заголовок.
        .Rows(1).Copy shRes.Rows(1)
'Задать начальный диапазон.
        Set rngStart = .[C2]
        Do Until IsEmpty(rngStart)
            Set rngPaste = shRes.[A65536].End(xlUp).Offset(1)
            lngData = GetNextRange(rngStart)
            rngStart.Offset(, -2).Resize(, 5).Copy rngPaste

'Копировать V1 в V14.
            rngStart.Offset(, 2).Resize(lngData).Copy
            rngPaste.Offset(, 5).PasteSpecial Paste:=xlAll, _
Operation:=xlNone, SkipBlanks:=False, Transpose:=True
'Копировать V1FP в V14FP.
            rngStart.Offset(, 1).Resize(lngData).Copy
            rngPaste.Offset(, 19).PasteSpecial Paste:=xlAll, _
Operation:=xlNone, SkipBlanks:=False, Transpose:=True
            Set rngStart = rngStart.Offset(lngData)
        Loop
    End With
    Application.GoTo shRes.[A1]
    shRes.Cells.Columns.AutoFit
    Application.ScreenUpdating = True
    Application.CutCopyMode = False
    If MsgBox("Удалить исходный рабочий лист?", 36) = 6 Then
        Application.DisplayAlerts = False
        Sheets("Транспонирование").Delete
        Application.DisplayAlerts = True
    End If
    Set rngPaste = Nothing
    Set rngStart = Nothing
    Set shRes = Nothing
    Exit Sub
terminate:

End Sub

Function GetNextRange(ByVal rngSt As Range) As Long
    Dim i As Long
    i = 0
    Do Until rngSt.Value <> rngSt.Offset(i).Value
        i = i + 1
    Loop
    GetNextRange = i
End Function
```

Выделение и отмена выделения несмежных ячеек

Макросы `ModifyRightClick`, `DeselectActiveCell` и `DeselectActiveArea` любезно предоставлены Томом Уртисом.

Чтобы отменить выделение ячейки или диапазона ячеек на рабочем листе, нужно щелкнуть на произвольной невыделенной ячейке. После выполнения этой операции выделение требуемых ячеек необходимо начинать заново, что весьма проблематично, если речь идет о большом количестве несмежных ячеек.

Следующий макрос добавляет в контекстное меню 2 новые команды: Отменить выделение активной ячейки и Отменить выделение активной области (рис. 13.8).

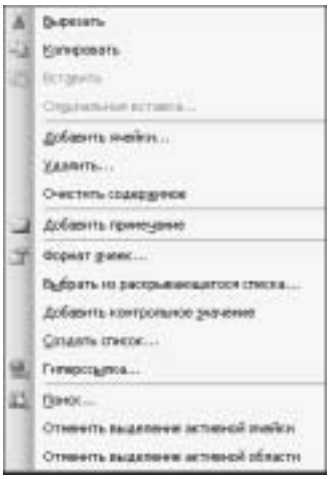


Рис. 13.8. Процедура `ModifyRightClick` создает 2 новые команды контекстного меню

Удерживая нажатой клавишу <Ctrl>, щелкните левой кнопкой мыши на ячейке несмежного диапазона, выделение которой следует отменить. Щелкните правой кнопкой мыши и выберите команду контекстного меню Отменить выделение активной ячейки (для отмены выделения только одной активной ячейки) или Отменить выделение активной области (для отмены выделения области, которой принадлежит активная ячейка).

Поместите приведенный ниже код в стандартный модуль.

```
Sub ModifyRightClick()
    Dim O1 As Object, O2 As Object
    On Error Resume Next
    With CommandBars("Cell")
        .Controls("Отменить выделение активной ячейки").Delete
        .Controls("Отменить выделение активной области").Delete
    End With
    On Error GoTo 0
    Set O1 = CommandBars("Cell").Controls.Add
    With O1
        .Caption = "Отменить выделение активной ячейки"
        .OnAction = "DeselectActiveCell"
    End With
    Set O2 = CommandBars("Cell").Controls.Add
    With O2
        .Caption = "Отменить выделение активной области"
        .OnAction = "DeselectActiveArea"
    End With
End Sub
```

```
Sub DeselectActiveCell()
    Dim x As Range, y As Range
    If Selection.Cells.Count > 1 Then
        For Each y In Selection.Cells
            If y.Address <> ActiveCell.Address Then
                If x Is Nothing Then
                    Set x = y
                Else
                    Set x = Application.Union(x, y)
                End If
            End If
        Next y
        If x.Cells.Count > 0 Then
            x.Select
        End If
    End If
End Sub

Sub DeselectActiveArea()
    Dim x As Range, y As Range
    If Selection.Areas.Count > 1 Then
        For Each y In Selection.Areas
            If Application.Intersect(ActiveCell, y) Is Nothing Then
                If x Is Nothing Then
                    Set x = y
                Else
                    Set x = Application.Union(x, y)
                End If
            End If
        Next y
        x.Select
    End If
End Sub
```

Разместите следующий код в модуле `ЭтаКнига` (ThisWorkbook).

```
Private Sub Workbook_Deactivate()
    Application.CommandBars("Cell").Reset
End Sub

Private Sub Workbook_Activate()
    ModifyRightClick
End Sub
```

VBA для профессионалов

VBA-программисты находятся в постоянном поиске более эффективных решений тех или иных задач. Размещая результаты своей работы в Internet, они оказывают неоценимую услугу всему сообществу VBA-программистов.

Установка параметров страницы

Макросы `Macro1`, `Macro1_Version2`, `Macro1_Version3` и `Macro1_Version4` любезно предоставлены Хуаном Пабло Гонсалесом (Juan Pablo

Gonzales), проживающим в Боготе, Колумбия. Хуан Пабло — разработчик программы F&I Menu Wizard, он также выполняет для Mr.Excel.com все заказы, поступающие от испаноязычных клиентов.

Следующие макросы выполняют одно и то же действие: устанавливают размеры верхнего, нижнего, левого и правого полей страницы равными 1,5 дюйма (3,8 см), а размеры полей верхнего колонтитула и нижнего колонтитула — равными 1 дюйму (2,5 см). Макрос `Macro1` создан средством записи макросов. Макросы `Macro1_Version2`, `Macro1_Version3` и `Macro1_Version4` демонстрируют возможность улучшения автоматически сгенерированного кода с целью повышения его производительности. На рис. 13.9 показана сравнительная таблица скорости выполнения всех четырех макросов.

8	Macro1	Macro1_Version2	Macro1_Version3	Macro1_Version4
9	1,6759	0,4162	0,4186	0,08
10	1,5347	0,4186	0,4207	0,08
11	1,5616	0,4155	0,4179	0,0792
12	1,5517	0,4174	0,4452	0,0794
13	1,5183	0,4169	0,4175	0,0773
14	1,5348	0,418	0,4171	0,0778
15	1,5982	0,42	0,4192	0,078
16	1,5289	0,4372	0,4183	0,0812
17	1,5277	0,4186	0,4176	0,0802
18	1,5618	0,416	0,4172	0,0796
19	1,5294	0,4162	0,4375	0,0793
20	1,5286	0,4496	0,4148	0,0793
21	1,5266	0,4177	0,4187	0,0801
22	1,5681	0,4202	0,4162	0,0791
23	1,5332	0,4173	0,4192	0,0775
24	1,5478	0,4171	0,4496	0,0781
25	1,5612	0,4241	0,4169	0,0799
26	1,5273	0,4141	0,4182	0,0786
27	1,5286	0,4149	0,4169	0,0797
28	1,5679	0,4184	0,4165	0,0808
29	**1,550631**	**0,421376923**	**0,421553846**	**0,079338462**
30	**4**	3	2	1
31				

Рис. 13.9. Сравнительная таблица скорости выполнения макросов, устанавливающих параметры страницы

```
Sub Macro1()
'
' Macro1 Macro
' Macro recorded 12/2/2003 by Juan Pablo Gonzalez
'
    With ActiveSheet.PageSetup
        .PrintTitleRows = ""
        .PrintTitleColumns = ""
    End With
    ActiveSheet.PageSetup.PrintArea = ""
    With ActiveSheet.PageSetup
        .LeftHeader = ""
        .CenterHeader = ""
        .RightHeader = ""
        .LeftFooter = ""
        .CenterFooter = ""
        .RightFooter = ""
```

```
        .LeftMargin = Application.InchesToPoints(1.5)
        .RightMargin = Application.InchesToPoints(1.5)
        .TopMargin = Application.InchesToPoints(1.5)
        .BottomMargin = Application.InchesToPoints(1.5)
        .HeaderMargin = Application.InchesToPoints(1)
        .FooterMargin = Application.InchesToPoints(1)
        .PrintHeadings = False
        .PrintGridlines = False
        .PrintComments = xlPrintNoComments
        .CenterHorizontally = False
        .CenterVertically = False
        .Orientation = xlPortrait
        .Draft = False
        .PaperSize = xlPaperLetter
        .FirstPageNumber = xlAutomatic
        .Order = xlDownThenOver
        .BlackAndWhite = False
        .Zoom = 100
    End With
End Sub
```

Как обычно, средство записи макросов создало весьма громоздкий код. Учитывая низкую скорость обновления объекта `PageSetup`, макрос `Macro1` остро нуждается в оптимизации, как показано ниже:

```
Sub Macro1_Version2()
    With ActiveSheet.PageSetup
        .LeftMargin = Application.InchesToPoints(1.5)
        .RightMargin = Application.InchesToPoints(1.5)
        .TopMargin = Application.InchesToPoints(1.5)
        .BottomMargin = Application.InchesToPoints(1.5)
        .HeaderMargin = Application.InchesToPoints(1)
        .FooterMargin = Application.InchesToPoints(1)
    End With
End Sub
```

Среднее время выполнения макроса `Macro1_Version2` более чем на 70% меньше среднего времени выполнения макроса `Macro1`, однако это еще далеко не все. Принимая во внимание низкую скорость обновления объекта `PageSetup`, воспользуемся выражением `If` для установки значения только тех параметров страницы, которые этого требуют.

В приведенном ниже макросе вызов функции `Application.InchesToPoints` был заменен фактическими значениями параметров страницы.

```
Sub Macro1_Version3()
    With ActiveSheet.PageSetup
        If .LeftMargin <> 108 Then .LeftMargin = 108
        If .RightMargin <> 108 Then .RightMargin = 108
        If .TopMargin <> 108 Then .TopMargin = 108
        If .BottomMargin <> 108 Then .BottomMargin = 108
        If .HeaderMargin <> 72 Then .HeaderMargin = 72
        If .FooterMargin <> 72 Then .FooterMargin = 72
    End With
End Sub
```

Разница во времени выполнения макросов Macro1_Version2 и Macro1_Version3 становится заметной при условии, что некоторые параметры страницы уже содержат нужные значения.

Макрос Macro1_Version4 вызывает XLM-метод PAGE.SETUP, позволяя тем самым сократить среднее время своего выполнения более чем на 95% по сравнению со средним временем выполнения макроса Macro1. Параметры метода PAGE.SETUP left, right, top, bot, head_margin и foot_margin измеряются не в точках, а в дюймах.

```
Sub Macro1_Version4()
    Dim St As String
    St = "PAGE.SETUP(, , " & _
      "1.5, 1.5, 1.5, 1.5" & _
      ", 0, False, False, False, 1, 1, True, 1, 1,False, , " & _
      "1, 1" & _
      ", False)"
    Application.ExecuteExcel4Macro St
End Sub
```

Параметрам left, right, top и bot соответствует 4-я строка приведенного выше кода, параметрам head_margin и foot_margin — 6-я строка. К сожалению, макрос Macro1_Version4 имеет два существенных недостатка. Во-первых, он основан на языке XLM, включенном в Excel для обеспечения обратной совместимости. Неизвестно, как долго Microsoft намерена поддерживать этот язык. Во-вторых, ошибка при указании параметров метода PAGE.SETUP приведет к тому, что этот метод не будет выполнен без какого-либо уведомления.

Вычисление времени выполнения кода макроса

Ниже приведен код, использующийся для вычисления времени выполнения макросов Macro1, Macro1_Version2, Macro1_Version3 и Macro1_Version4 (см. рис. 13.9).

```
Public Declare Function QueryPerformanceFrequency Lib _
    "kernel32" (lpFrequency As Currency) As Long

Public Declare Function QueryPerformanceCounter Lib _
    "kernel32.dll" (lpPerformanceCount As Currency) As Long

Sub Test()
    Dim Ar(1 To 20, 1 To 4) As Currency, WS As Worksheet
    Dim n As Currency, str As Currency, fin As Currency
    Dim y As Currency

    Dim i As Long, j As Long

    Application.ScreenUpdating = False
    For i = 1 To 4
        For j = 1 To 20
            Set WS = ThisWorkbook.Sheets.Add
            WS.Range("A1").Value = 1
```

```
            QueryPerformanceFrequency y
            QueryPerformanceCounter str
            Select Case i
            Case 1: Macro1
            Case 2: Macro1_Version2
            Case 3: Macro1_Version3
            Case 4: Macro1_Version4
            End Select
            QueryPerformanceCounter fin
            Application.DisplayAlerts = False
            WS.Delete
            Application.DisplayAlerts = True
            n = (fin - str)
            Ar(j, i) = CCur(Format(n, _
"##########.############") / y)
        Next j
    Next i
    With Range("A8").Resize(1, 4)
        .Value = Array("Macro1", "Macro1_Version2", _
"Macro1_Version3", "Macro1_Version4")
        .Font.Bold = True
    End With
    Range("A9").Resize(20, 4).Value = Ar

    With Range("A29").Resize(1, 4)
' В англоязычной версии Excel:
' .FormulaR1C1 = "=AVERAGE(R2C:R21C)"
        .FormulaR1C1Local = "=СРЗНАЧ(R2C:R21C)"
' В англоязычной версии Excel:
' .Offset(1).FormulaR1C1 = "=RANK(R22C, R22C1:R22C4, 1)"
        .Offset(1).FormulaR1C1Local = "=РАНГ(R22C; R22C1:R22C4; 1)"
        .Resize(2).Font.Bold = True
    End With
    Application.ScreenUpdating = True
End Sub
```

Запрет/разрешение выполнения операций вырезания, копирования и вставки

Макросы `EnableAllClear` и `DisAbleAllCLear` любезно предоставлены Иваном Ф. Моалой.

Иногда изменение пользователями содержимого рабочего листа крайне нежелательно. Следующий код запрещает/разрешает все способы выполнения операций вырезания, копирования и вставки данных.

```
Option Private Module

Dim ComBar As CommandBar
Dim ComBarCtrl As CommandBarControl

Sub EnableAllClear()
'// Команда Вставка (Insert)..
    EnableControl 295, True  '// ..Ячейки (Cells)
    EnableControl 296, True  '// ..Строки (Rows)
    EnableControl 297, True  '// ..Столбцы (Columns)
```

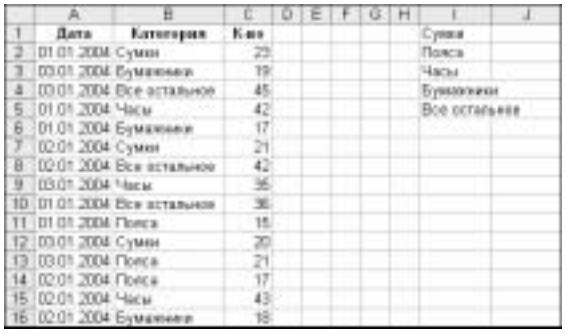
```
'// Команда...
    EnableControl 478, True  '// Правка->Удалить (Edit->Delete)
    EnableControl 292, True  '// Удалить (Delete), контекстное
                             '// меню ячейки
    EnableControl 293, True  '// Удалить строки (Delete Rows),
                             '// контекстное меню строки
    EnableControl 294, True  '// Удалить (Delete), контекстное
                             '// меню столбца
    EnableControl 847, True  '// Правка->Удалить лист
                             '// (Edit->Delete Sheet)
'// Команда...
    EnableControl 3125, True  '// Очистить содержимое (Clear
                              '// Contents), контекстное меню
    EnableControl 1964, True  '// Правка->Очистить->Все (All)
    EnableControl 872, True   '// Правка->Очистить->Форматы
                              '// (Edit->Clear->Formats)
    EnableControl 873, True   '// Правка->Очистить->Содержимое
                              '// (Edit->Clear->Contents)
    EnableControl 874, True   '// Правка->Очистить->Примечания
                              '// (Edit->Clear->Comments)
'// Команда...
    EnableControl 21, True    '// Вырезать (Cut)
    EnableControl 19, True    '// Копировать (Copy)
    EnableControl 22, True    '// Вставить (Paste)
    EnableControl 755, True   '// Специальная вставка
                              '// (Paste Special)
'// Комбинации клавиш.
    With Application
        .OnKey "^c"
        .OnKey "^v"
        .OnKey "+{DEL}"
        .OnKey "+{INSERT}"
        .CellDragAndDrop = True
        .OnDoubleClick = ""
    End With
'// Панели инструментов.
    CommandBars("ToolBar List").Enabled = True

End Sub

Sub DisAbleAllCLear()
'// Команда Вставка (Insert)..
    EnableControl 295, False  '// ..Ячейки (Cells)
    EnableControl 296, False  '// ..Строки (Rows)
    EnableControl 297, False  '// ..Столбцы (Columns)
'// Команда...
    EnableControl 478, False  '// Правка->Удалить (Edit->Delete)
    EnableControl 292, False  '// Удалить (Delete), контекстное
                              '// меню ячейки
    EnableControl 293, False  '// Удалить строки (Delete Rows),
                              '// контекстное меню строки
    EnableControl 294, False  '// Удалить (Delete), контекстное
                              '// меню столбца
    EnableControl 847, False  '// Правка->Удалить лист
                              '// (Edit->Delete Sheet)
'// Команда...
    EnableControl 21, False   '// Вырезать (Cut)
```

```
    EnableControl 19, False   '// Копировать (Copy)
    EnableControl 22, False   '// Вставить (Paste)
    EnableControl 755, False  '// Специальная вставка
                              '// (Paste Special)
'// Команда...
    EnableControl 3125, False '// Очистить содержимое (Clear
                              '// Contents), контекстное меню
    EnableControl 1964, False '// Правка->Очистить->Все (All)
    EnableControl 872, False  '// Правка->Очистить->Форматы
                              '// (Edit->Clear->Formats)
    EnableControl 873, False  '// Правка->Очистить->Содержимое
                              '// (Edit->Clear->Contents)
    EnableControl 874, False  '// Правка->Очистить->Примечания
                              '// (Edit->Clear->Comments)
'// Комбинации клавиш.
    With Application
        .OnKey "^c", "Dummy"
        .OnKey "^v", "Dummy"
        .OnKey "+{DEL}", "Dummy"
        .OnKey "+{INSERT}", "Dummy"
        .CellDragAndDrop = False
        .OnDoubleClick = "Dummy"
    End With
'// Панели инструментов.
    CommandBars("ToolBar List").Enabled = False

End Sub

Sub EnableControl(iId As Integer, blnState As Boolean)
Dim ComBar As CommandBar
Dim ComBarCtrl As CommandBarControl

On Error Resume Next
For Each ComBar In Application.CommandBars
    Set ComBarCtrl = ComBar.FindControl(Id:=iId, recursive:=True)
    If Not ComBarCtrl Is Nothing Then ComBarCtrl.Enabled = blnState
Next

End Sub

Sub Dummy()
    '// Вывод сообщения.
    MsgBox "Команда недоступна!"
End Sub
```

Определение порядка сортировки

Макрос `CustomSort` любезно предоставлен Вэем Цзянгом (Wei Jiang), проживающим в г. Шиян, Китай. Цзянг работает на должности консультанта в компании MrExcel Consulting.

Excel поддерживает сортировку списков в числовом или алфавитном порядке. Иногда этого оказывается недостаточно. Как показано на рис. 13.10, желаемый порядок сортировки списка выглядит так: “Пояса”, “Сумки”, “Часы”, “Бумажники”, “Все остальное”.



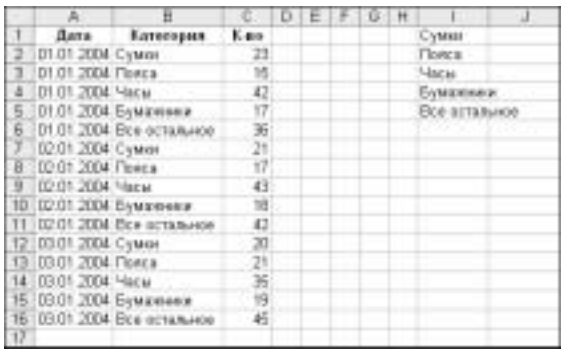
	A	B	C	I
1	Дата	Категория	Кол-во	Сумки
2	01.01.2004	Сумки	23	Пояса
3	03.01.2004	Бумажники	19	Часы
4	03.01.2004	Все остальное	45	Бумажники
5	01.01.2004	Часы	42	Все остальное
6	01.01.2004	Бумажники	17	
7	02.01.2004	Сумки	21	
8	02.01.2004	Все остальное	42	
9	03.01.2004	Часы	36	
10	01.01.2004	Все остальное	36	
11	01.01.2004	Пояса	16	
12	03.01.2004	Сумки	20	
13	03.01.2004	Пояса	21	
14	02.01.2004	Пояса	17	
15	02.01.2004	Часы	43	
16	02.01.2004	Бумажники	18	

Рис. 13.10. Желаемый порядок сортировки списка в ячейках `A2:C16` указан в столбце `I`

Следующий макрос сортирует список с учетом заданного порядка сортировки.

```
Sub CustomSort()
    ' Задать желаемый порядок сортировки.
    Application.AddCustomList ListArray:=Range("I1:I5")

' Определить номер списка, задающего порядок сортировки.
    nIndex = Application.GetCustomListNum(Range("I1:I5").Value)

' Отсортировать список, используя заданный порядок сортировки.
' Номер списка, задающего порядок сортировки, равен nIndex + 1,
' поскольку обычный порядок сортировки имеет номер 1.
    Range("A2:C16").Sort Key1:=Range("B2"), Order1:=xlAscending, _
Header:=xlNo, Orientation:=xlSortColumns, OrderCustom:=nIndex + 1
    Range("A2:C16").Sort Key1:=Range("A2"), Order1:=xlAscending, _
Header:=xlNo, Orientation:=xlSortColumns

' Удалить список, задающий порядок сортировки.
    Application.DeleteCustomList nIndex
End Sub
```

Результат выполнения макроса `CustomSort` показан на рис. 13.11.



	A	B	C	I
1	Дата	Категория	Кол-во	Сумки
2	01.01.2004	Сумки	23	Пояса
3	01.01.2004	Пояса	16	Часы
4	01.01.2004	Часы	42	Бумажники
5	01.01.2004	Бумажники	17	Все остальное
6	01.01.2004	Все остальное	36	
7	02.01.2004	Сумки	21	
8	02.01.2004	Пояса	17	
9	02.01.2004	Часы	43	
10	02.01.2004	Бумажники	18	
11	02.01.2004	Все остальное	42	
12	03.01.2004	Сумки	20	
13	03.01.2004	Пояса	21	
14	03.01.2004	Часы	36	
15	03.01.2004	Бумажники	19	
16	03.01.2004	Все остальное	45	
17				

Рис. 13.11. В результате выполнения макроса список в ячейках `A2:C16` отсортирован сперва по дате, а затем — в соответствии с порядком, заданным в столбце `I`

Создание индикатора хода процесса

Макрос `Worksheet_Change` любезно предоставлен Томом Уртисом.

Следующий макрос создает индикатор хода процесса в столбце `C`, основываясь на данных в столбцах `A` и `B` (рис. 13.12).

Рис. 13.12. Индикатор хода процесса

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column > 2 Or Target.Cells.Count > 1 Then Exit Sub
If Application.IsNumber(Target.Value) = False Then
    Application.EnableEvents = False
    Application.Undo
    Application.EnableEvents = True
    MsgBox "Введите число."
    Exit Sub
End If
Select Case Target.Column
    Case 1
        If Target.Value > Target.Offset(0, 1).Value Then
            Application.EnableEvents = False
            Application.Undo
            Application.EnableEvents = True
            MsgBox "Значение в столбце A не может быть больше _
значения в столбце B."
            Exit Sub
        End If
    Case 2
        If Target.Value < Target.Offset(0, -1).Value Then
            Application.EnableEvents = False
            Application.Undo
            Application.EnableEvents = True
            MsgBox "Значение в столбце B не может быть меньше _
значения в столбце A."
        Exit Sub
    End If
End Select
Dim x As Long
```

```
x = Target.Row
Dim z As String
z = Range("B" & x).Value - Range("A" & x).Value
With Range("C" & x)
' В англоязычной версии Excel:
' .Formula = "=IF(RC[-1]<=RC[-2],REPT(""n"",RC[-1])&REPT(""n"", _
RC[-2]-RC[-1]),REPT(""n"",RC[-2])&REPT(""o"",RC[-1]-RC[-2]))"
    .FormulaLocal = "=ЕСЛИ(RC[-1]<=RC[-2];ПОВТОР(""n""; _
RC[-1])&ПОВТОР(""n"";RC[-2]-RC[-1]);ПОВТОР(""n""; _
RC[-2])&ПОВТОР(""o"";RC[-1]-RC[-2]))"
    .Value = .Value
    .Font.Name = "Wingdings"
    .Font.ColorIndex = 1
    .Font.Size = 10
    If Len(Range("A" & x)) <> 0 Then
        .Characters(1, (.Characters.Count - z)).Font.ColorIndex = 3
        .Characters(1, (.Characters.Count - z)).Font.Size = 12
    End If
End With
End Sub
```

Создание защищенного поля для ввода пароля

Макрос `HiddenPassword` любезно предоставлен Даниелем Клэнном (Daniel Klann), проживающим в Сиднее, Австралия. Даниель знаком со множеством различных языков программирования, однако предпочитает разработку приложений на VBA для Excel и Access. Клэнн — автор собственного Web-сайта, который находится по адресу: `www.danielklann.com`.

Использование для ввода пароля обычного текстового поля крайне нежелательно, поскольку все вводимые символы отображаются на экране. Следующий макрос заменяет символы пароля знаками звездочки (*), превращая обычное текстовое поле в защищенное поле ввода пароля (рис. 13.13).

Рис. 13.13. Макрос `HiddenPassword` превращает обычное текстовое поле в защищенное поля ввода пароля

```
'Используемые функции API.
Private Declare Function CallNextHookEx Lib "user32" _
(ByVal hHook As Long, ByVal ncode As Long, _
ByVal wParam As Long, lParam As Any) As Long

Private Declare Function GetModuleHandle Lib "kernel32" Alias _
"GetModuleHandleA" (ByVal lpModuleName As String) As Long
```

```
Private Declare Function SetWindowsHookEx Lib "user32" Alias _
"SetWindowsHookExA" (ByVal idHook As Long, ByVal lpfn As Long, _
ByVal hmod As Long, ByVal dwThreadId As Long) As Long

Private Declare Function UnhookWindowsHookEx Lib "user32" _
(ByVal hHook As Long) As Long

Private Declare Function SendDlgItemMessage Lib "user32" Alias _
"SendDlgItemMessageA" (ByVal hDlg As Long, _
ByVal nIDDlgItem As Long, ByVal wMsg As Long, _
ByVal wParam As Long, ByVal lParam As Long) As Long

Private Declare Function GetClassName Lib "user32" _
Alias "GetClassNameA" (ByVal hwnd As Long, ByVal lpClassName _
As String, ByVal nMaxCount As Long) As Long

Private Declare Function GetCurrentThreadId Lib "kernel32" () _
As Long

'Константы, используемые в функциях API.
Private Const EM_SETPASSWORDCHAR = &HCC
Private Const WH_CBT = 5
Private Const HCBT_ACTIVATE = 5
Private Const HC_ACTION = 0

Private hHook As Long

Sub HiddenPassword()

    If InputBoxDK("Введите пароль", "Ввод пароля") <> _
"password" Then

        MsgBox "Неверный пароль! Доступ запрещен."

    Else

        MsgBox "Верный пароль! Добро пожаловать!"

    End If

End Sub

Public Function NewProc(ByVal lngCode As Long, _
ByVal wParam As Long, ByVal lParam As Long) As Long
    Dim RetVal
    Dim strClassName As String, lngBuffer As Long

    If lngCode < HC_ACTION Then
        NewProc = CallNextHookEx(hHook, lngCode, wParam, lParam)
        Exit Function
    End If

    strClassName = String$(256, " ")
    lngBuffer = 255

    If lngCode = HCBT_ACTIVATE Then    'Окно активировано.
```

```
        RetVal = GetClassName(wParam, strClassName, lngBuffer)

'Имя класса обычного текстового поля.
        If Left$(strClassName, RetVal) = "#32770" Then

'Следующая строка превращает обычное текстовое поле в защищенное
'поле ввода пароля. Символ "*" можно заменить на любой другой.
            SendDlgItemMessage wParam, &H1324, _
EM_SETPASSWORDCHAR, Asc("*"), &H0
        End If

    End If

'Следующая строка гарантирует корректный вызов
'всех обработчиков прерываний.
    CallNextHookEx hHook, lngCode, wParam, lParam

End Function

Public Function InputBoxDK(Prompt, Optional Title, _
Optional Default, Optional XPos, Optional YPos, _
Optional HelpFile, Optional Context) As String
    Dim lngModHwnd As Long, lngThreadID As Long

    lngThreadID = GetCurrentThreadId
    lngModHwnd = GetModuleHandle(vbNullString)

    hHook = SetWindowsHookEx(WH_CBT, AddressOf NewProc, _
lngModHwnd, lngThreadID)
    On Error Resume Next
    InputBoxDK = InputBox(Prompt, Title, Default, XPos, YPos, _
HelpFile, Context)
    UnhookWindowsHookEx hHook

End Function
```

Изменение регистра текста

Макрос `TextCaseChange` любезно предоставлен Иваном Ф. Моалой.

Word позволяет изменять регистр выделенного текста. Следующий макрос делает возможным изменение регистра текста, находящегося в выделенном диапазоне рабочего листа Excel.

```
Sub TextCaseChange()
Dim RgText As Range
Dim oCell As Range
Dim Ans As String
Dim strTest As String
Dim sCap As Integer, _
    lCap As Integer, _
    i As Integer

'// Предполагается, что перед вызовом макроса
'// пользователь выделил требуемый диапазон текста.

Again:
```

```
Ans = Application.InputBox("[С]трочные" & vbCr & "[П]рописные" _
& vbCr & "[К]ак в предложениях" & vbCr & "[Н]ачинать с прописных" _
& vbCr & "[М]алые прописные", "Введите букву", Type:=2)

If Ans = "False" Then Exit Sub
If InStr(1, "СПКНМ", UCase(Ans), vbTextCompare) = 0 Or _
Len(Ans) > 1 Then GoTo Again

On Error GoTo NoText
If Selection.Count = 1 Then
    Set RgText = Selection
Else
    Set RgText = Selection.SpecialCells(xlCellTypeConstants, 2)
End If
On Error GoTo 0

For Each oCell In RgText
    Select Case UCase(Ans)
        Case "С": oCell = LCase(oCell.Text)
        Case "П": oCell = UCase(oCell.Text)
        Case "К": oCell = UCase(Left(oCell.Text, 1)) & _
LCase(Right(oCell.Text, Len(oCell.Text) - 1))
        Case "Н": oCell = Application.WorksheetFunction.Proper( _
oCell.Text)
        Case "М"
                lCap = oCell.Characters(1, 1).Font.Size
                sCap = Int(lCap * 0.85)
'Малые прописные для всех букв.
                oCell.Font.Size = sCap
                oCell.Value = UCase(oCell.Text)
                strTest = oCell.Value
'Большие прописные для 1-х букв слов.
                strTest = Application.Proper(strTest)
                For i = 1 To Len(strTest)
                    If Mid(strTest, i, 1) = UCase(Mid(strTest, _
i, 1)) Then
                        oCell.Characters(i, 1).Font.Size = lCap
                    End If
                Next i
    End Select
Next

Exit Sub
NoText:
MsgBox "Текст в диапазоне " & Selection.Address & " отсутствует"

End Sub
```

Обработка события удаления строки или столбца

Макросы `EventHack` и `EventReset`, а также вспомогательные процедуры `AssignMacro`, `JudgeRng` и `DelExecute` любезно предоставлены Масару Каджи (Masaru Kaji).

Excel не предусматривает возможность обработки события удаления строки или столбца. Следующий макрос устраняет этот недостаток: при удалении

строки или столбца на экран выводится сообщение, содержащее номер удаленной строки или удаленного столбца.

```
Sub EventHack()
    AssignMacro "JudgeRng"
End Sub
Sub EventReset()
    AssignMacro ""
End Sub

Private Sub AssignMacro(ByVal strProc As String)
    Dim lngId As Long
    Dim CtrlCbc As CommandBarControl
    Dim CtrlCbcRet As CommandBarControls
    Dim arrIdNum As Variant

'// 293: команда Удалить строки (Delete Rows)
'//      контекстного меню строки.
'// 294: команда Удалить (Delete) контекстного
'//      меню столбца.
'// 478: команда Удалить (Delete) меню Правка (Edit).
    arrIdNum = Array(293, 294, 478)

    For lngId = LBound(arrIdNum) To UBound(arrIdNum)
        Set CtrlCbcRet = CommandBars.FindControls( _
ID:=arrIdNum(lngId))
        For Each CtrlCbc In CtrlCbcRet
            CtrlCbc.OnAction = strProc
        Next
        Set CtrlCbcRet = Nothing
    Next
End Sub

Private Sub JudgeRng()
    If Not TypeOf Selection Is Range Then Exit Sub
    With Selection
        If .Address = .EntireRow.Address Then
            Call DelExecute("строка: " & .Row, xlUp)
        ElseIf .Address = .EntireColumn.Address Then
            Call DelExecute("столбец: " & .Column, xlToLeft)
        Else
            Application.Dialogs(xlDialogEditDelete).Show
        End If
    End With
End Sub

Private Sub DelExecute(ByVal str, ByVal lngDerec As Long)
    MsgBox "Удален(а): " & str
    Selection.Delete lngDerec
End Sub
```

Поиск заданного текста с помощью свойства SpecialCells

Макросы `CheckAllCells` и `CheckSpecialCellsOnly` любезно предоставлены Иваном Ф. Моалой.

При поиске значения, текста или формулы в заданном диапазоне Excel проверяет каждую ячейку этого диапазона. Макрос `CheckAllCells` прово-

дит поиск заданного текста в диапазоне A1:Z20000, проверяя каждую его ячейку. Макрос CheckSpecialCellsOnly проводит поиск заданного текста в диапазоне A1:Z20000, проверяя только ячейки, содержащие текстовые константы. По окончании поиска оба макроса выводят сведения о результатах поиска и затраченном на это времени.

```
Sub CheckAllCells()
Dim TheRange As Range
Dim oCell As Range

    StartTime = Now
    Ctr = 0
    Set TheRange = Range("A1:Z20000")
    For Each oCell In TheRange
        If oCell.Text = "Ваш текст" Then
            Ctr = Ctr + 1
        End If
    Next oCell
    EndTime = Now

    Msg = "Проверено " & TheRange.Cells.Count & " ячеек. _
Найдено " & Ctr & " совпадений. Время поиска = " & _
Format(EndTime - StartTime, "h:mm:ss")
    MsgBox Msg

End Sub

Sub CheckSpecialCellsOnly()
Dim TheRange As Range
Dim oCell As Range

    StartTime = Now
    Ctr = 0

    Set TheRange = Range("A1:Z20000").SpecialCells( _
xlCellTypeConstants, xlTextValues)

    For Each oCell In TheRange
        If oCell.Text = "Ваш текст" Then
            Ctr = Ctr + 1
        End If
    Next oCell
    EndTime = Now

    Msg = "Проверено " & TheRange.Cells.Count & " ячеек. _
Найдено " & Ctr & " совпадений. Время поиска = " _
& Format(EndTime - StartTime, "h:mm:ss")
    MsgBox Msg

End Sub
```

Условное удаление строк

Макрос Delete_rows_with_conditions любезно предоставлен Деннисом Валентайном (Dennis Wallentin), проживающим в Остерсунде, Швеция.

Следующий макрос удаляет определенное число строк при выполнении условия "ячейка в столбце A пуста".

```
Sub Delete_rows_with_conditions()
Dim wbBook As Workbook
Dim wsSheet As Worksheet
Dim rnArea As Range
Dim lnLastRow As Long, lnMoreRows As Long
Dim i As Long, j As Long, k As Long

Set wbBook = ThisWorkbook
Set wsSheet = wbBook.Worksheets("Условное удаление строк")

Application.ScreenUpdating = False

'Определение числа строк, которые нужно удалить.
j = 5

With wsSheet
'Определение номера последней строки по столбцу A.
    lnLastRow = .Cells(Rows.Count, 1).End(xlUp).Row + 1

'Определение номера последней строки по столбцам A, B и C.
    For k = 1 To 3
        lnMoreRows = .Cells(Rows.Count, k).End(xlUp).Row + 1
        If lnLastRow < lnMoreRows Then
            lnLastRow = lnMoreRows
        End If
    Next k

'Удаление строк с пустой ячейкой в столбце A.
    For i = lnLastRow To 1 Step -1
        If IsEmpty(.Cells(i, 1)) Then
            .Cells(i, 1).Resize(j, 1).EntireRow.Delete
        End If
    Next i
End With

Application.ScreenUpdating = True

End Sub
```

Сокрытие строки формул

Макрос `Worksheet_SelectionChange` любезно предоставлен Томом Уртисом.

Если активная ячейка содержит более 50 символов, строка формул автоматически увеличивается в размере и перекрывает часть рабочего листа Excel. Следующий макрос скрывает строку формул при выделении ячейки, содержащей более 50 символов.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
If Target.Cells.Count > 1 Then Exit Sub
On Error Resume Next
If Len(Target.Text) > 50 Or Len(Target.Formula) > 50 Then
```

```
    Application.DisplayFormulaBar = False
Else
    Application.DisplayFormulaBar = True
End If
End Sub
```

Результат выполнения макроса `Worksheet_SelectionChange` показан на рис. 13.14.

Рис. 13.14. Макрос `Worksheet_SelectionChange` скрывает строку формул при выделении ячейки, содержащей более 50 символов

На закуску

В этом разделе рассматривается несколько занятных макросов, которым можно найти применение в самых различных приложениях.

Извлечение информации о курсах акций из Internet

Процедура `GetQuote` любезно предоставлена Натаном П. Оливером.

Следующая процедура извлекает из Internet информацию о курсе акций компании на заданную дату.

```
Private Sub GetQuote()
Dim ie As Object, lCharPos As Long, sHTML As String
Dim HistDate As Date, HighVal As String, LowVal As String
Dim cl As Range

Set cl = ActiveCell
HistDate = cl(, 0)

If Intersect(cl, [c2:c65366]) Is Nothing Then
    MsgBox "Выберите ячейку в столбце C."
    Exit Sub
End If

If Not CBool(Len(cl(, -1))) Or Not CBool(Len(cl(, 0))) Then
    MsgBox "Введите символ акций и требуемую дату."
    Exit Sub
End If

Set ie = CreateObject("InternetExplorer.Application")

With ie
    .Navigate "http://bigcharts.marketwatch.com/historical/ _
default.asp?detect=1&symbol=" & cl(, -1) & "&close_date=" & _
Month(HistDate) & "%2F" & Day(HistDate) & "%2F" & _
Year(HistDate) & "&x=31&y=26"
    Do While .Busy And .ReadyState <> 4
        DoEvents
    Loop
    sHTML = .Document.body.innertext
    .Quit
End With

Set ie = Nothing

lCharPos = InStr(1, sHTML, "High:", vbTextCompare)
If lCharPos Then HighVal = Mid$(sHTML, lCharPos + 5, 15)

If Not Left$(HighVal, 3) = "n/a" Then
    lCharPos = InStr(1, sHTML, "Low:", vbTextCompare)
    If lCharPos Then LowVal = Mid$(sHTML, lCharPos + 4, 15)
    cl.Value = (Val(LowVal) + Val(HighVal)) / 2
Else: lCharPos = InStr(1, sHTML, "Closing Price:", vbTextCompare)
    cl.Value = Val(Mid$(sHTML, lCharPos + 14, 15))
End If

Set cl = Nothing
End Sub
```

Результат выполнения процедуры `GetQuote` показан на рис. 13.15.

	A	B	C	D
1	Символ	Дата	Курс к закрытию	Узнать курс
2	Dell	12.01.1994	25,3125	
3	MSFT	30.01.2003	49,18	
4	VFINX	20.01.2000	133,21	
5	INNOX	06.01.2003	20,02	
6	INSTX	17.02.2004	10,94	

Рис. 13.15. Процедура `GetQuote` извлекает из Internet информацию о курсе акций компании на заданную дату

Вставка программного кода во вновь созданную рабочую книгу

Ранее в этой главе был рассмотрен макрос, создающий отчеты для региональных менеджеров в виде отдельных рабочих книг. Усложним задачу, потребовав скопировать в новые рабочие книги программный код. Для этого обратимся к объектной модели Microsoft Visual Basic for Applications Extensibility, позволяющей не только импортировать в рабочую книгу модули с программным кодом, но и создавать код непосредственно в рабочей книге.

Прежде чем выполнить какой-либо из приведенных ниже макросов, подключите библиотеку Microsoft Visual Basic for Applications Extensibility 5.3, выбрав команду меню редактора Visual Basic Tools⇨References (Сервис⇨Ссылки) и установив соответствующий флажок в открывшемся диалоговом окне.

Наиболее простой способ переноса программного кода заключается в экспорте модуля и пользовательской формы из текущей рабочей книги и их последующем импорте в новую рабочую книгу. Предположим, что региональному менеджеру необходимо передать отчет с данными о его регионе и 3 макроса, реализующие форматирование и печать отчета. Поместите все 3 макроса в модуль `modToRegion`. Предположим также, что макросы модуля `modToRegion` вызывают пользовательскую форму `frmRegion`. Следующий макрос перенесет программный код и пользовательскую форму в новую рабочую книгу.

```
Sub MoveDataAndMacro()
    Dim WSD As Worksheet
    Set WSD = Worksheets("Отчет")
' Скопировать отчет в новую рабочую книгу.
    WSD.Copy
' Сделать новую рабочую книгу активной.
' Удалить старые копии промежуточных файлов с диска.
    On Error Resume Next
    Kill ("C:\ModToRegion.bas")
    Kill ("C:\frmRegion.frm")
    On Error GoTo 0
' Экспортировать модуль и форму из исходной рабочей книги.
    ThisWorkbook.VBProject.VBComponents("ModPics").Export _
("C:\ModToRegion.bas")
    ThisWorkbook.VBProject.VBComponents("frmModPics").Export _
("C:\frmRegion.frm")
' Импортировать модуль и форму в новую рабочую книгу.
    ActiveWorkbook.VBProject.VBComponents.Import _
("C:\ModToRegion.bas")
    ActiveWorkbook.VBProject.VBComponents.Import _
("C:\frmRegion.frm")
    On Error Resume Next
    Kill ("C:\ModToRegion.bas")
    Kill ("C:\frmRegion.frm")
    On Error GoTo 0
End Sub
```

Чтобы создать программный код непосредственно в рабочей книге, следует обратить внимание на два инструмента: метод `Lines` и метод `InsertLines`. Метод `Lines` позволяет извлечь заданное число строк из указанного модуля, а метод `InsertLines` — вставить код в модуль.

> **Внимание**
>
> Код, который добавляется в рабочую книгу с помощью метода `InsertLines`, тут же компилируется. Сбой при компиляции кода может привести к ошибке общего нарушения защиты (General Protection Failure, GPF), чего нужно всячески избегать.

Следующий макрос копирует весь код модуля `ЭтаКнига` исходной рабочей книги в модуль `ЭтаКнига` новой рабочей книги.

```
Sub MoveDataAndMacro()
    Dim WSD As Worksheet
    Dim WBN As Workbook
    Set WSD = Worksheets("Отчет")
' Скопировать отчет в новую рабочую книгу.
    WSD.Copy
' Сделать новую рабочую книгу активной.
    Set WBN = ActiveWorkbook
' Скопировать обработчики событий уровня рабочей книги.
    Set WBCodeMod1 = ThisWorkbook.VBProject.VBComponents( _
"ЭтаКнига").CodeModule
    Set WBCodeMod2 = WBN.VBProject.VBComponents( _
"ЭтаКнига").CodeModule
    WBCodeMod2.InsertLines 1, WBCodeMod1.Lines(1, _
WBCodeMod1.CountOfLines)
```

Следующий шаг

В следующей главе рассматриваются Web-запросы, позволяющие импортировать данные из Internet в приложения Excel.