[file:///C:/Users/gtx28/Downloads/Telegram%20Desktop/1088.pdf](C://Users/gtx28/Downloads/Telegram%20Desktop/1088.pdf)

13.02.23

# MS Office – среда разработки.

MS Office – система приложений, объединенных единым интерфейсом и возможностью переноса данных. Также это платформа для созданий пользовательских приложений.

Каждое из приложений Office имеет 2 уровня использования:

1. Применение приложения как готового инструмента;
2. Применение приложения как инструментальной среды для разработки собственных приложений, решающих специфические задачи.

MS Office – масштабируемая среда разработки, позволяющая решать как простые, так и сложные задачи. К простым задачам относятся создание документов и создание небольших систем документов.

Сложные задачи: разработка клиентской части клиент-серверного приложения, рассчитанного на работу большого числа пользователей с базами данных (MS SQL Server, Oracle); разработка документов для совместного использования в интернете; разработка в большой системы обеспечивающая автоматизацию деятельности офиса.

MS Office содержит среду разработки – редактор Visual Basic в отдельном от основного приложения окне. Также обладает языком программирования VBA. VBA является расширением языка Visual Basic.

## VBA или язык офисного программирования.

VBA – язык визуального и ООП, позволяет создавать собственные объекты и работать с большим количеством библиотечных объектов.

Основной задачей офисного программирования является создание документов. В этом контексте под термином «документ» понимается объект в ООП, который представляет собой собрание данных разного типа и программ, обрабатывающих эти данные.

Рабочие книги Excel, документы Word, базы данных Access, и др.

Все создаваемые программные компоненты документа объединяются в одно целое, которое называют проектом. Проект является частью документа и не существует вне его. Он хранится вместе с документом. Создать независимый от документа проект на VBA невозможно.

## Система приложений MS Office

Приложения Office имеют общие возможности:

* Объектная модель всех приложений
* Использование шаблонов – файлов на базе которых можно создавать семейства близких документов
* Использование надстроек, дополнительных программ, расширяющих возможности приложения, добавлением специальных команд и новых функций
* Макро рекодер – транслятор действий пользователя на ЯП. Результатом является создание макроса (процедуры без параметров)
* Использование activeX – дополнительных инструментов, расширяющих возможности взаимодействия приложения и пользователя.
* Мастера – инструмент пошагового процесса достижения цели

## MS Excel

Возможности:

* События – основные объекты Excel рабочие книги и рабочие листы обладают событиями, использую которые можно получить контроль над объектами
* Сводные таблицы. Можно создавать и изменять программно
* Проверка данных. Можно проверять данные при вводе в том числе без использования программного кода.

Использование шаблонов. В данном случае шаблон – это книга, имеющая форматирование, может включать текст, например заголовки страниц, заголовки строк и столбцов, макросы, нестандартную панель инструментов. Созданный программный код можно хранить в шаблоне, документе или в надстройках.

## Редактор Visual Basic

Редактор Visual Basic – среда разработки, такую среду имеют основные приложения Office: Excel, Power Point, Access. Каждому объекту в проекте соответствует свое окно кода при этом объектами могут быть: рабочая книга, рабочий лист, модуль VBA, модуль класса – специальный тип модуля, позволяющий создавать новые классы объектов, форма userForm.

Проектом Office называется набор программных модулей, связанных с основным документом приложения. Он хранится вместе с документом и не может быть отделен от него.

В редакторе имеются следующие рабочие окна:

* Project explorer – окно проводника проекта, здесь можно посмотреть компоненты проекта и выполнить различные операции
* UserForm – окно формы, появляется, когда Вы редактируете пользовательскую форму
* Tool – панель инструментов управления, из нее можно добавлять инструменты в форму или документ
* Properties – здесь можно посмотреть свойства проекта или компонентов проекта и изменить их
* Код – окно программного кода
* Object browser – обозреватель объектов
* Watch – окно, контролируемых приложений, используется для отладки locals (окно локальных переменных)
* Immediate – окно для немедленного выполнения команд в ходе отладки

## Установка параметров редактора

Вкладка Editor параметры:

* Автоматическая проверка синтаксиса. По этому параметру появляется диалоговое окно, при обнаружении редактором ошибки в коде.
* Обязательное объявление переменных. При установленном параметре редактор вставляет в начало каждого модуля оператор Option Explicit в этом случае требуется явно объявить каждую переменную.
* Автоматическая вставка объектов. Предоставляет помощь при вводе кода, отображая список элементов текущего объекта.
* Отображать краткие сведения. Отображает информацию об аргументах функции, свойств и методов, которые вы вводите.
* Автоматический отступ. Определяет автоматически расположение каждой новой строки с тем же отступом, что и для предыдущей строки.

Вкладка Editor формат. Параметры определяют отображение кода, цвет, шрифт, размер и др.

Вкладка General содержит общие параметры.

Вкладка \*\*\*\*\* здесь включенный параметр означает, что окно прикреплено по отношению к границам редактора.

# Visual Basic for application

## Синтаксис

* VBA не чувствителен к регистру
* для комментания до конца строки используют REM или “”
* Символьные значения заключаются в «””»
* Максимальная длина имени 255 символов.
* Начало нового оператора – перевод на новую строку
* Нет ограничения на максимальную длину строки
* Если в строке несколько операторов, они разделются «;»
* Можно объединить несколько физических строк в одну логическую при помощи «ПРОБЕЛ» и знака подчеркивания

## Переменные, типы данных, константы

Переменная – именованная область памяти, где могут храниться различные данные, которые можно изменять во время исполнения программы. Переменная имеет 2 характеристики: имя переменной и тип данных, которые хранятся в переменной. Имя переменной должно начинаться с буквы, состоять из букв, цифр и символа подчеркивания, быть уникальным в пределах области видимости, не совпадать с ключевыми словами Visual Basic, не длиннее 255 символов.

Определение типа данных задает область возможных значений типа, структуру организации данных, операции, определенные над данными этого типа.

При объявлении переменной объявляется ее тип и область видимости. Когда переменные не имеют тип явно они имеют тип Variant. Для определения типа используют: Dim, Public, Private, Static.

ПН гр 4.6 в 11.55 + лекция, вт 9:50

Любая группа инструкций выполняемых в процедуре или функиции.

Тип значения возвращаемого функции. Возвращаемое значение функции - Optional. Указывает, что аргумент не является обязательным. При использовании этого элемента все последующие аргументы в списке должны быть описаны с помощью Optional. Недопускается использование данного ключевого слова, если используется ParamArray

27.02.23

Основные функции языка:

* Проверки типов (TypeName, IsDate, IsEmpty, IsError, IsNull, IsNumeric,
* IsObject…)
* Преобразования типов (CBool, CInt, CLng, CDbl,…, Fix, Int)
* Преобразования данных (Chr, Asc, DateValue…)
* Математические функции (Abs, Mod, Cos, Rnd…)
* Функции обработки строк (StrComp, Len, Instr, Mid, Left, Trim…)
* Функции обработки дат и времени (DateSerial, Day, Month, Year)

**Управляющие инструкции**

Инструкция - синтаксически завершенная конструкция, представляющая отдельное действие, описание или определение. Обычно, инструкция занимает отдельную строку программы, хотя допускается использование двоеточия (:) для размещения в одной строке нескольких инструкций. Допускается также использование пробела и символа продолжения строки (\_) для продолжения одной логической программной строки на нескольких физических строках.

**Основы синтаксиса**

Синтаксис VBA почти полностью совпадает с синтаксисом Visual Basic.

Основные синтаксические принципы этого языка следующие:

VBA нечувствителен к регистру;

чтобы закомментировать код до конца строки, используется одинарная

кавычка (') или команда REM;

символьные значения должны заключаться в двойные кавычки (");

максимальная длина любого имени в VBA (переменные, константы, процедуры) — 255 символов;

начало нового оператора — перевод на новую строку (точка с запятой, как в C, Java, JavaScript, для этого не используется);

ограничений на максимальную длину строки нет (хотя в редакторе в

строке помещается только 308 символов). Несколько операторов в одной

строке разделяются двоеточиями: MsgBox "Проверка 1" : MsgBox "Проверка 2"

для удобства чтения можно объединить несколько физических строк в одну логическую при помощи пробела и знака подчеркивания после него:

MsgBox "Сообщение пользователю" \_

& vUserName

*Условный оператор If...Then...Else*

Задает выполнение определенных групп инструкций в зависимости от

значения выражения. Синтаксис:

If условие Then [инструкции] [Else инструкции\_else]

Или

If условие Then

[инструкции]

[ElseIf условие-n Then

[инструкции\_elseif] ...

[Else

[инструкции\_else]]

End If

Пример.

If strUserPassword = "Triangle" Then

' [Вход пользователя в систему]

Else

' [Сообщение о неверном пароле]

End If

*Оператор выбора Select Case*

Выполняет одну из нескольких групп инструкций в зависимости от

значения выражения.

Инструкция Select Case может служить альтернативой инструкции ElseIf в

If...Then...Else при оценке одного выражения, которое имеет несколько

возможных значений. В то время как If...Then...Else для каждой инструкции

ElseIf оценивает разные выражения, инструкция Select Case оценивает

выражение только один раз, в начале управляющей структуры.

Синтаксис:

Select Case выражение

[Case списокВыражений-n

[инструкции-n]] ...

[Case Else

[инструкции\_else]]

End Select

Пример.

Sub InputExpense()

Dim strUserName As String

strUserName = InputBox("Введите свое имя")

Select Case strUserName

Case "Teresa"

MsgBox "Доступ автора"

Case "James"

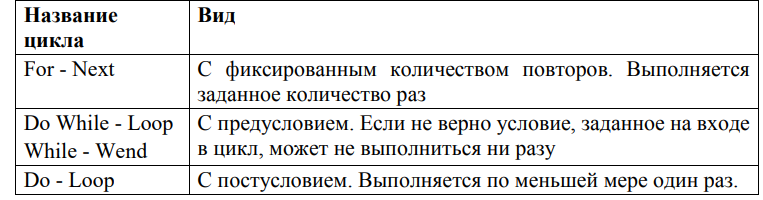
MsgBox "Доступ пользователя"

Case Else

MsgBox "В доступе отказано", vbCritical

End Select

End Sub



Обработка ошибок

Типы ошибок

Можно выделить два типа ошибок, с которыми сталкивается программист:

* ошибки, которые сопровождают создание программ,
* ошибки времени выполнения.
* Ошибки, возникающие при создании программ

Синтаксические – если неправильно будет введен оператор или ключевое слово, если не указать часть выражения. Многие синтаксические ошибки "отлавливаются" редактором кода VBA еще в процессе ввода кода. Об обнаружении других ошибок сообщается в ходе компиляции и запуска программы. При этом компилятор VBA выдает информацию о том, в какой строке кода обнаружена ошибка и в чем она заключается.

Логические – связаны с неправильными формулами расчета показателей, неверным использованием переменных и т.д. В ходе выполнения программа ведет себя не так, как планировалось. Главное здесь — найти причину неправильного поведения программы. Обычно для выявления и исправления ошибок такого типа предназначены приемы отладки.

Ошибки времени выполнения возникают в нормально работающих программах, которые прошли проверку на синтаксическую и логическую правильность.

С этими ошибками можно бороться, используя один из двух методов.

1. Разработка программы таким образом, чтобы не допустить этих ошибок, создание программных конструкций, которые предотвращают возникновение ошибок.

2. Перехват ошибок и их обработка.

При создании обработчика ошибок, Вы используете три составляющие:

• включение обработчика ошибок (перехват ошибки – установка ловушки);

• написание обработчика;

• возврат из обработчика.

**Общие принципы обработки ошибок:**

1. Перед опасным кодом (сохранение или открытие файла, возможность деления на ноль и т. п.) помещается команда:

OnErrorGoToметка\_обработчика\_ошибки

например:

Dim a As Integer, b As Integer, c As Integer

On Error GoTo ErrorHandlerDivision

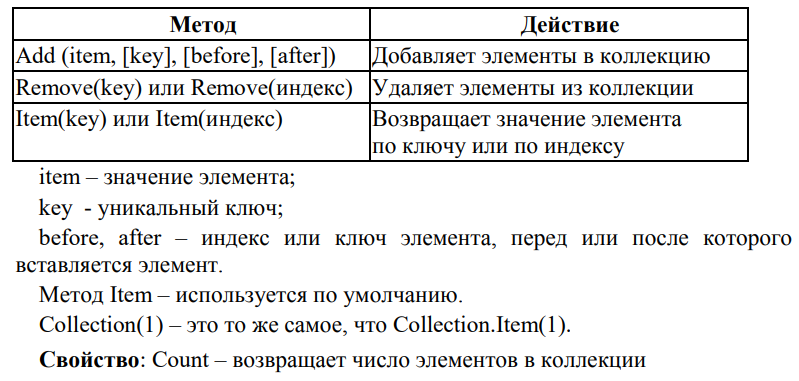
c = a / b

2. Далее в коде программы помещается метка обработчика ошибки и программный код обработки:

ErrorHandlerDivision:

MsgBox "Ошибкаприделении"

1. Поскольку в такой ситуации код обработчика ошибки будет выполняться даже в том случае, если ошибки не было, есть смысл поставить перед меткой обработчика команду Exit Sub (если это процедура) или Exit Function(если это функция)



### Использование объектной переменной

Объектная переменная используется для ссылки на объект.

Сначала описывается объектная переменная, потом указывается Set для присвоения объекта переменной.

Пример.

Dim rngTest As Excel.Range

Set rngTest = Application. Workbooks(1).Worksheets(1).Range("A1")

rngTest.Value = 5

1. Описание объектных переменных

Dim xlsSht As Excel.Worksheet - раннее связывание

Dim genObj As Object - позднее связывание

2. Установка объектной переменной

Пример.

Dim xlsWbk As Excel.Workbook

63

Set xlsWbk = Workbooks(1)

xlsWbk.PrintOut

Dim wrdDoc As Document

Set wrdDoc = Documents.Add

wrdDoc.Content.InsertAfter "Hello there."

3. Nothing

- Определение установлена ли объектная переменная

Dim xlsSht As Worksheet

...

If xlsSht Is Nothing Then

MsgBox "Переменная не указывает на объект"

Else

MsgBox "Переменная содержит ссылку на объект"

End If

Прекращение связи объектной переменной с объектом

Set xlsSht = Nothing

### Работа с объектами

Все объекты можно разделить на 3 группы

1. Объекты, чей класс определен пользователем в одном из модулей класса

Только спецификатор New позволяет создать новый объект для классов,

определенных программистом.

2. Объекты родного приложения (Excel, Word…), которому принадлежит

проект и которые доступны по умолчанию

Для создания таких объектов не используются ни спецификатор New, ни

функция CreateObject. Новые объекты, если и создаются, то

специальными методами своего класса (например, метод Add коллекции

Workbooks)

3. Active-X и ComAddIns – объекты, в частности объекты других

приложений Office при их подключении к исходному приложению.

Для создания таких объектов используются спецификатор New или

функция CreateObject.

Работая с объектами, Вы работаете с его свойствами и методами.

Синтаксис

Объект.свойство, Объект.Метод

Пример.

MsgBox Application.Workbooks.Count & " open workbooks."

**Доступ к свойствам**

Dim strDocAuthor As String

Dim wrdDoc As Document

Set wrdDoc = Documents("MyDoc")

application.UserName="Mary" '- задать свойство объекта

strDocAuthor = application.UserName ' - прочитать свойство объекта

Использование методов

Workbooks(1).PrintOut From:=2, To:=3, Copies:=1

**Использование возвращаемых значений метода**

Dim wrdDoc As Document

64

Set wrdDoc = Application.Documents.Add

wrdDoc.Content.InsertAfter "Hello"

Set d = Documents.Add(Template:="expense.dot")– метод возвращает значение

Documents.Add Template:="expense.dot" – не возвращает значение

**Использование With …End With**

Инструкция With позволяет указывать объект только один раз для последовательности инструкций. При использовании With …End With программа работает быстрее.

Например, вместо

Selection.Font.Bold = True

Selection.Font.Italic = True

Selection.Font.Underline = True

можно использовать:

With Selection.Font

.Bold = True

.Italic = True

.Underline = True

End With

### Автоматизация офисных приложений

Можно использовать автоматизацию (ранее называемую OLE-Automation) для использования объектов одного приложения в другом. Вы можете создать процедуру в WORD, которая будет работать с объектами Microsoft Excel, PowerPoint или Microsoft Access.

Две составляющие автоматизации: Клиент и Сервер. Приложения Office 2010 могут быть как клиентами, так и серверами. Ссылки на объектные библиотеки При использовании Автоматизации (использования объектов другого приложения) надо сначала установить ссылку на объектную библиотеку и ссылаться на объект с первого объекта в объектной иерархии. Объектная библиотека описывает типы объектов, имеющихся в приложении и методы и свойства объектов этого приложения.

Библиотеки, входящие в каркас документа по умолчанию:

Word (Excel…) - библиотека, задающая основу документов приложения

Word (Excel…). Здесь хранится корневой объект Application и все классы объектов, вложенных в корневой объект.

Office - библиотека объектов, общих для всех приложений Office 2010.

Stdole - библиотека классов для работы с OLE – объектами для реализации Автоматизации.

VBA - библиотека классов языка VBA. Здесь хранятся все стандартные функции, встроенные в язык, и многое другое.

Project - проект, связанный с документом. Ссылки в приложениях Использование CreateObject(Имя\_приложения.Тип\_объекта) Функция CreateObject создает новый экземпляр объекта и возвращает

ссылку на запущенный экземпляр объекта. Если объект может существовать в одном экземпляре (например, PowerPoint), функция вернет запущенный

экземпляр объекта.

Dim xlsApp As Excel.Application

Dim wrdApp As Word.Application

Dim pptApp As PowerPoint.Application

Set xlsApp = CreateObject ("Excel.Application.9")

Set wrdApp = CreateObject ("Word.Application.9")

Set pptApp = CreateObject ("PowerPoint.Application.9")

Использование GetObject([pathname] [, class]) Возвращает ссылку на объект ActiveX (Объект, который может быть открыт для других приложений и средств программирования через интерфейсы программирования), сохраненный в файле.

Pathname - Полный путь и имя файла, содержащего объект, который следует загрузить. Если аргумент pathname опущен, должен быть указан аргумент class. Class (имяПриложения.типОбъекта) - Строка, представляющая класс объекта. Если аргумент pathname опущен, GetObject возвращает текущий активный объект указанного типа. При наличии текущего экземпляра объекта, необходимо использовать функцию GetObject.

Пример.

Sub TestForApp()

On Error GoTo StartExcel

Set xlsApp = GetObject(, "Excel.Application")

Exit Sub

StartExcel:

If Err.Number = 429 Then

MsgBox "Будет запущен excel."

Set xlsApp = CreateObject("Excel.Application")

xlsApp.Visible = True

MsgBox "Запустили excel."

Else

MsgBox Err.Description

End If

End Sub

Примеры управления приложениями

Управление Word - приложением

Sub AutomateWord()

' Для тестирования кода - вставьте код в одно из приложений

' Microsoft Excel, PowerPoint или Microsoft Access

' и установите ссылку на Word object library

'Замечание: Если Вы работаете под Windows NT – вы должны

'изменить операторы описания

'Dim wrdApp As Object

'Dim wrdDoc As Object

Dim wrdApp As Word.Application

Dim wrdDoc As Word.Document

Set wrdApp = CreateObject("Word.Application")

wrdApp.Visible = True

Set wrdDoc = wrdApp.Documents.Add

wrdDoc.Content.InsertAfter "Hello World!"

wrdDoc.SaveAs ("C:\MyNewWordDoc.doc")

wrdDoc.Close

wrdApp.Quit

End Sub

Управление Excel- приложением

Sub AutomateExcel()

' Для тестирования кода - вставьте код в одно из приложений

' Word, PowerPoint или Microsoft Access

' и установите ссылку на Excel object library

' Замечание: Если Вы работаете под Windows NT – вы должны

' изменить операторы описания

' Dim xlsApp As Object

Dim xlsApp As Excel.Application

Set xlsApp = CreateObject("Excel.Application")

67

xlsApp.Visible = True

xlsApp.Workbooks.Add

xlsApp.Range("A1") = "Hello World!"

xlsApp.Workbooks(1).SaveAs \_

("C:\MyNewExcelFile.xls")

xlsApp.Quit

End Sub

Управление PowerPoint- приложением

Sub AutomatePowerPoint()

Dim pptApp As PowerPoint.Application

Dim pptPres As PowerPoint.Presentation

Dim pptSlide As PowerPoint.Slide

Set pptApp = CreateObject("PowerPoint.Application")

pptApp.Visible = True

Set pptPres = pptApp.Presentations.Add

Set pptSlide = pptApp.ActivePresentation.Slides.Add(1, ppLayoutText)

pptSlide.Shapes("Rectangle 2").TextFrame.TextRange.Text = "Hello"

pptSlide.Shapes("Rectangle 3").TextFrame.TextRange.Text = \_

"To the entire World!"

pptApp.Presentations(1).SaveAs \_

("C:\NewPowerPointPresentation.ppt")

pptApp.Quit

End Sub

Управление Microsoft Access - приложением

Sub DisplayCatalog()

Static accApp As Access.Application

Set accApp = GetObject \_

("C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb")

accApp.Visible = True

accApp.DoCmd.OpenReport "Alphabetical List of Products", View:=acNormal

accApp.Quit

End Sub

Управление Outlook - приложением

Sub SendMail()

Dim olkApp As Outlook.Application

Dim maliNewMail As MailItem

Set olkApp = CreateObject("Outlook.Application")

Set maliNewMail = olkApp.CreateItem(olMailItem)

With maliNewMail

.To = "devtrain@microsoft.com"

.Body = "Please review the proposal I sent you."

.Send

End With

End Sub

Типичные свойства и методы приложений

* Свойство Visible – видимость;
* Метод Quit – выход из приложения;
* Метод Activate – активизация приложений;
* Метод Run - запуск макроса;
* Методы Copy and Paste вложенных объектов…

В различных приложениях многие методы совпадают по именам и предназначены для решения стандартных задач, организованы по-разному. Например, метод Quit() в Excel не имеет параметров, он естественно завершает приложение. В Word у этого метода три параметра SaveChanges, Format, RouteDocument, - позволяющие в момент выхода указать, сохранять ли сделанные изменения и их формат, пересылать ли документ всем, кто работает с ним.

### Типичные свойства и методы приложений

* Свойство Visible – видимость;
* Метод Quit – выход из приложения;
* Метод Activate – активизация приложений;
* Метод Run - запуск макроса;
* Методы Copy and Paste вложенных объектов…

В различных приложениях многие методы совпадают по именам и предназначены для решения стандартных задач, организованы по-разному. Например, метод Quit() в Excel не имеет параметров, он естественно завершает приложение. В Word у этого метода три параметра SaveChanges, Format, RouteDocument, - позволяющие в момент выхода указать, сохранять ли сделанные изменения и их формат, пересылать ли документ всем, кто работает с ним.

# Использование Microsoft Excel объектов

## Объектная модель Microsoft Excel

Объекты Microsoft Excel

Каждый компонент в Microsoft Excel может быть представлен как объект

VBA.

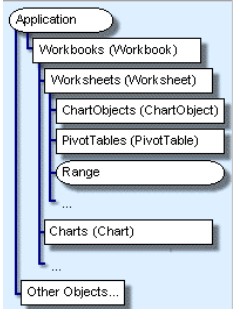
Например, Рабочая книга -> Workbook

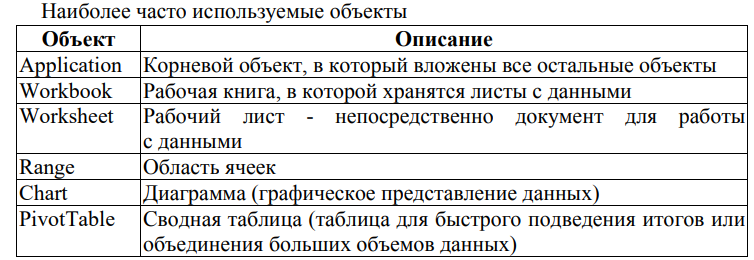
Рабочий лист -> Worksheet

Лист -> Sheet

Набор ячеек -> Range

Диаграмма -> Chart





Доступ к объектам Microsoft Excel из Excel-приложения

- полный путь:

Application.Workbooks("MyBook.xls").Worksheets("Sales").Range("A1").Select

- краткий путь:

Range("A1").Select

Можно использовать свойства объекта Application ActiveCell, ActiveChart,

ActiveSheet, ActiveWorkbook…

Доступ к объектам Microsoft Excel из других приложений:

Sub AutomateExcel()

' поставить ссылку на библиотеку

Dim xlApp as Excel.Application

Set xlApp = CreateObject("Excel.Application")

xlApp.Visible = True

End Sub

**События Workbook**

1. событие Open

Private Sub Workbook\_Open()

Windows(1).DisplayWorkbookTabs = False

End Sub

2. событие Activate

Private Sub Workbook\_Activate()

Application.CommandBars("Web").Visible = True

End Sub

3. событие Deactivate

Private Sub Workbook\_Deactivate()

Application.CommandBars("Web").Visible = False

End Sub

4. событие BeforeClose

Private Sub Workbook\_BeforeClose(Cancel As Boolean)

ActiveWorkbook.BuiltinDocumentProperties("Author") = "Jean Selva"

End Sub

5. событие BeforePrint

Private Sub Workbook\_BeforePrint (Cancel As Boolean)

If ActiveWorkbook.Saved = False Then

' Saved – true, если не было изменений после сохранения документа

MsgBox "Пожалуйста, сохраните книгу перед печатью"

Cancel = True

Exit Sub

End If

End Sub

6. событие NewSheet

Private Sub Workbook\_NewSheet(ByVal Sh As Object)

' при добавлении листа, он перемещается в конец книги

Sh.Move After:=Sheets(Sheets.Count)

End Sub

## Использование Worksheet

### Select (выделение)

Для выделения рабочего листа используется метод Activate объекта

### Worksheet

Sub ActivateSalesInfo()

' если в книге есть лист с именем "SalesInfo", сделать его активным

' в противном случае, добавить лист с этим именем

Dim sh as Worksheet

For Each sh In Worksheets

If sh.Name ="SalesInfo" Then

sh.Activate

Exit Sub

End If

Next

ActiveWorkbook.Worksheets.Add

ActiveSheet.Name = "SalesInfo"

End Sub

### Свойство Range

1) Range("A1").Value = "Треугольник"

2) Range("Profit").Formula = "=SUM(A1:A3)"

Пример.

Sub SortMyRange()

Dim rngNames As Range

Set rngNames = Range("A1", "A25")

rngNames.Sort Key1:=Range("A1"), Order1:=xlDescending

End Sub

### Свойство Offset

Свойство Offset объекта Range имеет два параметра RowOffset и

ColumnOffset – смещение по строкам и столбцам, и возвращает новый

объект, отстоящий от прежнего на заданное расстояние.

Sub ValidateRange()

Dim intCheckCells As Integer

Worksheets("Лист1").Activate

Range("A1").Activate

For intCheckCells = 1 To 10

If Not IsNumeric(ActiveCell) Then

MsgBox "Здесь не число!"

Exit Sub

End If

ActiveCell.Offset(1, 0).Activate

Next

End Sub

### Свойство Cells

Свойство Cells объекта Range возвращает объект Range. Позволяет

работать как со всеми ячейками листа, так и с единичными ячейками.

Sub MultiplicationTable()

Dim intRow As Integer, intCol As Integer

For intRow = 1 To 10

For intCol = 1 To 10

Cells(intRow, intCol).Value = intRow \* intCol

Next intCol

Next intRow

End Sub

### События Worksheet

1. событие Activate

Private Sub Worksheet\_Activate()

Range("a1:a10").Sort Key1:=Range("a1"), Order1:=xlAscending

End Sub

2. событие Calculate

Private Sub Worksheet\_Calculate()

Columns("A:F").AutoFit

End Sub

3. событие Deactivate

Private Sub Worksheet\_Deactivate()

Dim intResponse As Integer

intResponse = MsgBox("Запускать проверку?", vbYesNo)

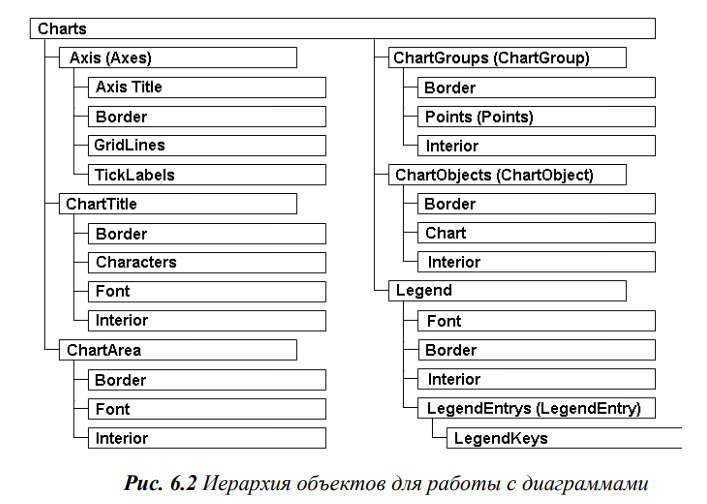
If intResponse = vbYes Then

' [запускается код проверки]

End If

End Sub

## Создание Charts (диаграмм)



Для быстрой программной реализации диаграммы можно воспользоваться методом ChartWizard объекта Chart. Его использование не требует отдельного задания свойства диаграммы.

### Коллекция Charts

Коллекция Charts содержит все листы с диаграммами конкретной рабочей

книги. Каждый лист диаграммы представляет собой объект Chart. Эта

коллекция не содержит диаграммы, внедренные в рабочий лист или лист

диалога.

Методы коллекции Charts

Метод Add

Метод Add добавляет пустой лист диаграммы. Синтаксис этого метода:

Expression.Add(Before, After, Count)

Expression - выражение, которое возвращает объект Charts.

Before - необязательный параметр типа Variant. Данный параметр

определяет лист рабочей книги, перед которым будет вставлен добавляемый

лист диаграммы.

After - необязательный параметр типа Variant. Предназначен для

определения листа рабочей книги, после которого будет вставлен

добавляемый лист диаграммы. В том случае, если оба параметра Before и

After отсутствуют, лист диаграммы будет вставлен перед активным листом.

78

Count - необязательный параметр типа Variant. Предназначен для

определения количества добавляемых листов. По умолчанию данный

параметр имеет значение равное 1.

Пример фрагмента программы, реализующей добавление в активную

рабочую книгу листа диаграммы, размещаемого после листа с именем Лист1.

ActiveWorkbook.Charts.Add After:=Worksheets("Лист1")

Метод Delete

Метод Delete позволяет удалить все листы диаграмм из коллекции Charts.

Объект Chart

Объект Chart представляет собой диаграмму, расположенную в рабочей

книге на отдельном листе диаграммы. Все объекты Chart являются

элементами коллекции Charts.

Свойства объекта Chart

Ниже рассмотрены некоторые свойства объекта Chart, позволяющие

управлять внешним видом диаграммы.

Свойство Legend возвращает ссылку на объект Legend, который

представляет собой легенду диаграммы.

Свойство ChartArea возвращает ссылку на объект ChartArea, который

предоставляет возможность работать с областью диаграммы.

Свойство ChartTitle возвращает ссылку на объект ChartTitle,

представляющий заголовок диаграммы.

Свойство ChartType определяет тип диаграммы. Значением этого

свойства может быть значение одной из констант, представленных в таблице,

например, http://www.taurion.ru/excel/pril1/32

Свойство HasAxis определяет тип осей, присутствующих на диаграмме.

Синтаксис данного свойства представлен ниже:

Expression.HasAxis(Indexl, Index2)

Expression Выражение, возвращающее объект Chart

Index1 Необязательный параметр типа Variant. Определяет тип осей и может

иметь одно из следующих значений:

XlCategory – ось категорий,

XlValue – ось значений,

XlSeriesAxis – ось рядов данных. Значение этой константы имеет смысл только для линейных диаграмм Index2 Необязательный параметр типа Variant. Определяет группу осей и может иметь одно из двух значений: xlPrimary или xlSecondary. Трехмерные

диаграммы имеют только одну группу осей

Свойство HasDataTable определяет присутствие (значение True) или отсутствие (значение False) таблицы данных на диаграмме.

Свойство HasLegend задает наличие (значение True) или отсутствие (значение False) легенды на диаграмме.

Свойство HasTitle определяет наличие (значение True) или отсутствие (значение False) заголовка и осей диаграммы.

Свойство Rotation задает угол отображения трехмерных диаграмм. Его значение измеряется в градусах в диапазоне от 0 до 360 (для трехмерной

линейчатой диаграммы это значение должно лежать в пределах от 0 до 44

По умолчанию для новой диаграммы значение свойства Rotation равно

**Методы объекта Chart**

Из существующего набора методов объекта Chart мы рассмотрим здесь

только один из них, наиболее полезный – ChartWizard. Этот метод

позволяет быстро задать параметры диаграммы без необходимости

определения каждого ее свойства в отдельности. Синтаксис этого метода

http://www.taurion.ru/excel/pril1/33.

**События объекта Chart**

Объект Chart, как и большинство других объектов, имеет набор событий,

включающих Activate, Deactivate, MouseDown, MouseUp и т. п.

**Событие Calculate**

Для объекта Chart событие Calculate происходит после того, как изменяются данные, лежащие в основе диаграммы.

**Метод Add**

Метод Add добавляет пустой лист диаграммы. Синтаксис этого метода:

Expression.Add(Before, After, Count)

Expression - выражение, которое возвращает объект Charts.

Before - необязательный параметр типа Variant. Данный параметр определяет лист рабочей книги, перед которым будет вставлен добавляемый лист диаграммы.

After - необязательный параметр типа Variant. Предназначен для определения листа рабочей книги, после которого будет вставлен добавляемый лист диаграммы. В том случае, если оба параметра Before и After отсутствуют, лист диаграммы будет вставлен перед активным листом.

Count - необязательный параметр типа Variant. Предназначен для определения количества добавляемых листов. По умолчанию данный параметр имеет значение равное 1.

Пример фрагмента программы, реализующей добавление в активную рабочую книгу листа диаграммы, размещаемого после листа с именем Лист1.

ActiveWorkbook.Charts.Add After:=Worksheets("Лист1")

**Метод Delete**

Метод Delete позволяет удалить все листы диаграмм из коллекции Charts.

**Объект Chart**

Объект Chart представляет собой диаграмму, расположенную в рабочей книге на отдельном листе диаграммы. Все объекты Chart являются элементами коллекции Charts.

Ниже рассмотрены некоторые свойства объекта Chart, позволяющие управлять внешним видом диаграммы.

Свойство Legend возвращает ссылку на объект Legend, который представляет собой легенду диаграммы.

Свойство ChartArea возвращает ссылку на объект ChartArea, который предоставляет возможность работать с областью диаграммы.

Свойство ChartTitle возвращает ссылку на объект ChartTitle,

представляющий заголовок диаграммы.

Свойство ChartType определяет тип диаграммы. Значением этого свойства может быть значение одной из констант, представленных в таблице,мнапример, http://www.taurion.ru/excel/pril1/32

Свойство HasAxis определяет тип осей, присутствующих на диаграмме.

Синтаксис данного свойства представлен ниже:

Expression.HasAxis(Indexl, Index2)

Expression Выражение, возвращающее объект Chart

Index1 Необязательный параметр типа Variant. Определяет тип осей и может иметь одно из следующих значений:

XlCategory – ось категорий,

XlValue – ось значений,

XlSeriesAxis – ось рядов данных. Значение этой константы имеет смысл только для линейных диаграмм

Index2 Необязательный параметр типа Variant. Определяет группу осей и может иметь одно из двух значений: xlPrimary или xlSecondary. Трехмерные диаграммы имеют только одну группу осей

Свойство HasDataTable определяет присутствие (значение True) или

отсутствие (значение False) таблицы данных на диаграмме.

Свойство HasLegend задает наличие (значение True) или отсутствие

(значение False) легенды на диаграмме.

Свойство HasTitle определяет наличие (значение True) или отсутствие

(значение False) заголовка и осей диаграммы.

Свойство Rotation задает угол отображения трехмерных диаграмм. Его

значение измеряется в градусах в диапазоне от 0 до 360 (для трехмерной

линейчатой диаграммы это значение должно лежать в пределах от 0 до 44).

По умолчанию для новой диаграммы значение свойства Rotation равно 20.

Из существующего набора методов объекта Chart мы рассмотрим здесь только один из них, наиболее полезный – ChartWizard. Этот метод позволяет быстро задать параметры диаграммы без необходимости определения каждого ее свойства в отдельности. Синтаксис этого метода <http://www.taurion.ru/excel/pril1/33>.

**Порядок создания сводной таблицы**

При работе со сводной таблицей необходимо:

1. Создать объект PivotCache, который будет представлять собой набор

записей, полученных с источника. Для каждого объекта PivotTable можно использовать только один объект PivotCache. Создание объекта PivotCache производится при помощи метода Add()коллекции PivotCaches:

Dim PC1 As PivotCache

Set PC1 = ActiveWorkbook.PivotCaches.Add(xlExternal)

80

2. Настроить параметры объекта PivotCache.

Главный метод объекта PivotCache — это метод CreatePivotTable(). С его помощью и производится следующий этап — создание сводной таблицы (объекта PivotTable). Этот метод принимает четыре параметра:

TableDestination— единственный обязательный параметр. Принимает объект

Range, в верхний левый угол которого будет помещена сводная таблица.

TableName— имя сводной таблицы. Если не указано, то автоматически сгенерируется имя типа "СводнаяТаблица1".

ReadData— если установить в True, то все содержимое куба будет автоматически помещено в кэш. С этим параметром нужно быть очень осторожным, поскольку неправильное его применение может резко увеличить нагрузку на клиента.

DefaultVersion— это свойство обычно не указывается. Позволяет определить версию создаваемой сводной таблицы. По умолчанию задается наиболее свежая версия. Создание сводной таблицы в первой ячейке первого листа книги может выглядеть так:

PC1.CreatePivotTableRange("A1")

Сводная таблица у нас создана, однако сразу после создания она пуста В ней предусмотрено четыре области, в которые можно размещать поля из источника:

 область столбцов — в нее помещаются те измерения ("разрез", в котором будут анализироваться данные), записей в которых меньше;

 область строк — те измерения, записей в которых больше;

 область страницы — те измерения, по которым нужно только проводить фильтрацию (например, показать данные только по такому-то региону

или только за такой-то год);

 область данных — центральная часть таблицы. Те числовые данные (например, сумма продаж), которые мы будем анализировать.

Часто требуется расположить данные в сводной таблице программным образом.

Эта операция производится при помощи объекта CubeField. Главное свойство этого объекта — Orientation, оно определяет, где будет находиться то или иное поле.

Например, помещаем измерение Customers в область столбцов:

PT1.CubeFields("[Customers]").Orientation = xlColumnField

измерение Time — в область строк:

PT1.CubeFields("[Time]").Orientation = xlRowField

измерение Product— в область страницы:

PT1.CubeFields("[Product]").Orientation = xlPageField

и, наконец, показатель UnitSales (число для анализа):

PT1.CubeFields("[Measures].[Unit Sales]").Orientation = xlDataField

Модификация созданной сводной таблицы (PivotTable)

Пример. Sub CreatePivotTable()

With Sheet1

.PivotTableWizard SourceType:=xlDatabase, SourceData:= \_ "Sales!R1C1:R541C7",

TableDestination:="", TableName:="Pivot1"

.PivotTables("Pivot1").AddFields RowFields:="Period", \_

ColumnFields:="Product",PageFields:=Array("Continent", "Category")

End With

End Sub

Группировка данных

Sub GroupQuarterly (pvtTable As PivotTable)

Dim rngGroup as Range

Set rngGroup = pvtTable.PivotFields("Date").DataRange

rngGroup.Cells(1).Group \_

periods:=Array(False,False,False,False,False,True,False)

' ARRAY (Seconds, Minutes, Hours, Days, Months, Quarters, Years)

End Sub

Обновление данных сводной таблицы pvtTable.RefreshTable