МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Озерский технологический институт — филиал НИЯУ МИФИ

Кафедра прикладной математики

Вл. Пономарев

ПРАКТИКУМ

по OLE Automation

Учебно-методическое пособие

2018 г.

УДК 681.3.06
П 56

Вл. Пономарев. Практикум по OLE Automation. Учебно-методическое пособие. Озерск: ОТИ НИЯУ МИФИ, 2018. — 25 с.

В пособии подробно излагается, как выполнять практические работы по дисциплине «Современные технологии программирования». Работы этой части включают изучение механизмов управления программными компонентами при помощи технологии OLE Automation.

В качестве основного материала при выполнении практических работ пособие предназначено для студентов, обучающихся по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения».

Рецензенты:
1.
2.

УТВЕРЖДЕНО
Редакционно-издательским
Советом ОТИ НИЯУ МИФИ

Содержание

Общие цели занятий 5
1. Работа OLE-201. Упаковка параметров 6
1.1. Рабочее пространство 6
1.2. Создание объекта автоматизации 6
1.3. Чтение свойства 7
1.4. Запись свойства 8
1.5. Вызов метода 8
1.6. Использование идентификатора локали 9
1.7. Контрольные вопросы и упражнения 9
2. Работа OLE-202. Домашняя работа по упаковке параметров 10
2.1. Варианты работ 10
2.2. Оформление отчета 10
3. Работа OLE-203. Объектная модель файловой системы 11
3.1. Рабочее пространство 11
3.2. Константы и переменные 12
3.3. Перечисление дисков 12
3.4. Перечисление каталогов 14
3.5. Перечисление файлов 15
3.6. Контрольные вопросы и упражнения 15
4. Работа OLE-204. Объектная модель Microsoft Excel 16
4.1. Рабочее пространство 16
4.2. Управление книгами 17
4.2.1. Сценарий xlQuit 17
4.2.2. Сценарий xlOpen 17
4.2.3. Сценарий xlDialogOpen 17
4.2.4. Сценарий xlOpenTabbed 17
4.2.5. Сценарий xlOpenNumbed 17
4.2.6. Сценарий xlSaveAs 18
4.2.7. Сценарий xlDialogSaveAs 18
4.2.8. Сценарий xlSaveAs97 18
4.3. Управление листами 18
4.3.1. Сценарий xlWorksheetAdd 18
4.3.2. Сценарий xlWorksheetDelete 18
4.3.3. Сценарий xlWorksheetCopy 18
4.3.4. Сценарий xlWorksheetMove 18
4.3.5. Сценарий xlWorksheetActivate1 19
4.3.6. Сценарий xlWorksheetActivate2 19
4.3.7. Сценарий xlWorksheetActivate3 19
4.4. Чтение и запись ячеек 19
4.4.1. Сценарий xlWsCellsSet 19
4.4.2. Сценарий xlWsRangeSet 19
4.4.3. Сценарий xlRangeGet 19

4.4.4. Сценарий xlSelectRow 20
4.4.5. Сценарий xlSelectCol 20
4.5. Оформление ячеек 20
4.5.1. Сценарий xlBorders 20
4.5.2. Сценарий xlInterior 20
4.5.3. Сценарий xlProtect 20
4.6. Создание формул и диаграмм 21
4.7. Контрольные вопросы и упражнения 21
5. Работа OLE-205. Объектная модель Microsoft Word 22
5.1. Рабочее пространство 22
5.2. Управление документами 22
5.2.1. Сценарий wdQuit 22
5.2.2. Сценарий wdOpen 23
5.2.3. Сценарий wdDialogOpen 23
5.2.4. Сценарий wdSaveAs 23
5.2.5. Сценарий wdDialogSaveAs 23
5.2.6. Сценарий wdSaveAs97 23
5.3. Управление текстом 23
5.3.1. Сценарий wdParaAfter 23
5.3.2. Сценарий wdParaBefore 24
5.3.3. Сценарий wdFont 24
5.4. Поиск и замена 24
5.4.1. Сценарий wdFind 24
5.4.2. Сценарий wdReplace 24
5.5. Проверка правописания 25
5.5.1. Сценарий wdGrammar 25

Общие цели занятий

В ходе практических работ предлагается изучить объектные модели продуктов компании Microsoft. В этой части работ рассматриваются следующие темы:

- управление дисками, каталогами и файлами;
- управление приложениями Microsoft Excel и Word;
- управление документами Microsoft Excel и Word;
- управление ячейками Excel;
- создание диаграмм;
- формирование текста;
- вставка в документ OLE-объектов.

Для выполнения работ используются сценарии на языке Visual Basic Scripting Edition, сценарии на языке Visual Basic for Application, программы на языке C++ в среде программирования Microsoft Visual Studio.

На выполнение и защиту работ этой части предположительно отводится 8-10 академических часов.

1. Работа OLE-201. Упаковка параметров

Цели:
- изучение интерфейса IDispatch.
Задачи:
- создание объекта автоматизации;
- получение CLSID;
- получение dispid;
- вызов свойства при помощи метода IDispatch::Invoke;
- вызов метода при помощи метода IDispatch::Invoke.
Опорные документы:

1.1. Рабочее пространство

Получим архив проекта для выполнения работы OLE-201. Извлечем из архива каталог disppars в корневой каталог диска C:. Откроем проект, убедимся, что целевой платформой является x86 или Win32, в зависимости от того, что есть. Программирование выполняется в модуле disppars.cpp, в основной функции mainf.

1.2. Создание объекта автоматизации

В качестве объекта автоматизации используем Microsoft Excel, класс Application, программный идентификатор Excel.Application. В начале модуля описываем необходимые для создания объекта структуры данных:

```
// ProgID
OLECHAR * szPROGID = (OLECHAR*)L"Excel.Application";
// свойство
OLECHAR * szVisible = (OLECHAR*)L"Visible";
// метод
OLECHAR * szEvaluate = (OLECHAR*)L"Evaluate";
LPDISPATCH pdisp = NULL;
CLSID clsid = {0};
DISPID dispid = 0;
HRESULT hr = 0;
```

Переходим в функцию mainf.

Сначала описываем завершение программы, так как при возникновении ошибки управление будет передаваться на метку complete:

```
void mainf() {
    // код завершения
complete:
    // освобождаем объекты COM
    if (pdisp) {
        pdisp->Release();
    }
    // деинициализируем OLE
    CoUninitialize();
}
```

Порядок создания объекта (в начале функции):

1. Инициализируем OLE при помощи функции CoInitialize.

2. Получаем CLSID при помощи функции CLSIDFromProgID.

3. Проверяем значение hr при помощи макроса FAILED. Если произошла ошибка, переходим на метку complete при помощи оператора goto.

Отладим эту часть программы. Выполним программу до выполнения пункта 2 включительно. Введем переменную clsid в окно Watch, убедимся, что переменная указывает на Microsoft Office Excel. Выполним программу до конца.

4. Создаем экземпляр объекта, запрашиваем IDispatch при помощи функции CoCreateInstance.

5. Проверяем hr.

Отладим эту часть программы.

Убедимся, что объект создается.

Выполняем программу до конца.

1.3. Чтение свойства

Будем читать свойство Visible. Сначала нужно получить dispid, затем вызвать Invoke, указывая множество параметров. Значение свойства возвращается в переменную типа Variant, а отсутствующие параметры упаковываются в пустую структуру DISPPARAMS.

Объявим необходимые структуры в начале модуля:

```
VARIANT vResult;
DISPPARAMS noArgs = { NULL, NULL, 0, 0 };
DISPID dispid = 0;
```

Порядок чтения свойства:

1. Инициализируем vResult при помощи функции VariantInit.

2. Получим dispid свойства при помощи метода GetIDsOfNames.

Первый параметр равен IID_NULL, в качестве массива имен параметров используем szVisible, количество имен 1, в качестве LCID используем значение 1033 (американский английский), последний параметр dispid.

Отладим эту часть программы.

Убедимся, что значение dispid равно 558.

Выполняем программу до конца.

3. Вызываем свойство Visible методом Invoke. Первый параметр dispid, второй равен IID_NULL, третий 1033, четвертый указывает на вызов свойства, он равен константе DISPATCH_PROPERTYGET, пятый указывает на структуру noArgs, шестой указывает на vResult, седьмой и восьмой параметры принимаем равными нулю.

4. Проверяем hr.

Отладим эту часть программы.

Убедимся, что значение vResult равно "BOOL = 0".

Выполняем программу до конца.

1.4. Запись свойства

Теперь запишем новое значение свойства Visible, равное истине, после чего приложение Excel должно появиться на экране.

Описываем необходимые структуры данных в начале модуля:

```
DISPPARAMS params;
DISPID dispids[1] = { DISPID_PROPERTYPUT };
VARIANTARG vararg[1];
```

Это структура для упаковки, массив dispid из одного элемента, равного константе DISPID_PROPERTYPUT, массив параметров из одного элемента.

Порядок действий:

1. Инициализируем параметр:

```
VariantInit(&vararg[0]);
```

2. Записываем массив параметров в структуру упаковки:

```
params.rgvarg = vararg;
```

3. Записываем в параметр значение свойства:

```
params.rgvarg[0].vt = VT_BOOL;
params.rgvarg[0].boolVal = -1;  // TRUE
```

4. Записываем массив именованных параметров в структуру упаковки:

```
params.rgdispidNamedArgs = dispids;
```

5. Устанавливаем количества параметров:

```
params.cArgs = 1;
params.cNamedArgs = 1;
```

6. Вызываем свойство Visible методом Invoke, учитывая, что dispid не изменился. Четвертый параметр равен константе DISPATCH_PROPERTYPUT.

7. Проверяем hr.

Отладим эту часть программы.

Убедимся, что приложение появляется.

Выполняем программу до конца.

1.5. Вызов метода

Будем вызывать метод Evaluate, dispid которого равен 1. Поскольку dispid нам известен, пропустим вызов метода GetIDsOfNames.

Сначала нужно сформировать и упаковать параметры. Будем вычислять косинус 45 градусов. Функция косинуса принимает в качестве параметра угол в радианах, поэтому вызываем функцию, которая переведет градусы в радианы, подставляя одну функцию в другую.

Порядок формирования параметров:

1. Задаем строку вычисляемого выражения:

```
params.rgvarg[0].vt = VT_BSTR;
params.rgvarg[0].bstrVal = SysAllocString(L"cos(radians(45))");
```

2. Задаем массив именованных параметров и число параметров:

```
params.rgdispidNamedArgs = NULL;
params.cArgs = 1;
params.cNamedArgs = 0;
```

3. Вызываем метод Evaluate методом Invoke. Первым параметром подставляем единицу. Параметр LCID принимаем равным 1033. Четвертый параметр равен константе DISPATCH_METHOD.

4. Проверяем hr.

Отладим эту часть программы.

Убедимся, что возвращаемый результат vResult равен "R8 = 0.707".

1.6. Использование идентификатора локали

Выясним, для чего нужно указывать идентификатор локали. Когда мы вызываем метод Evaluate, идентификатор локали задает язык, используемый для обозначения функций в выражении. Так, функция косинуса на любом языке пишется как "cos", а вот функция перевода в радианы на разных языках имеет разное написание.

В вызове метода Invoke заменим идентификатор локали на 1049 (русский, обычный). Тогда вычисляемое выражение перепишем так:

```
params.rgvarg[0].bstrVal = SysAllocString(L"cos(радианы(45))");
```

Выполним этот вызов Invoke и убедимся, что выражение вычисляется.

Теперь вернем в вызове Invoke идентификатор локали 1033.

Выполним вызов Invoke, убедимся, что теперь в vResult возвращается значение ошибки.

1.7. Контрольные вопросы и упражнения

1. Опишите параметры метода IDispatch::GetIDsOfNames.
2. Опишите параметры метода IDispatch::Invoke.
3. Опишите структуру DISPPARAMS.

2. Работа OLE-202. Домашняя работа по упаковке параметров

Тема работы:
Упаковка параметров метода Invoke.
Цели:
- изучение упаковки параметров для метода Invoke.
Задачи:
- упаковка необязательных и поименованных параметров.
Опорные документы:

2.1. Варианты работ

Работы предполагают использование сервера Microsoft Excel.

Реализуется вызов метода с двумя параметрами, первый из которых задается как позиционный, а второй как поименованный, или как позиционный, при этом часть параметров помечаются как отсутствующие.

1. Метод Application.CheckSpelling, параметр IgnoreUpperCase.
2. Метод Workbooks.Open с указанием разделителя полей Tab.
3. Метод Workbooks.OpenText с указанием разделителя полей #.
4. Метод Workbook.SaveAs с указанием параметра AddToMru.
5. Метод Worksheet.SaveAs с указанием параметра AddToMru.
6. Метод Worksheet.Protect с указанием параметра AllowXXX.
7. Метод Worksheets.Add, параметр Count.

2.2. Оформление отчета

Отчет оформляется при помощи шаблона, который можно получить на сайте преподавателя.

В отчете описывается, как была выполнена упаковка, в разделе первого уровня, название раздела «Упаковка параметров».

Программный код приводить нельзя.

Во втором разделе приводятся сведения об источниках, название раздела «Использованные источники», раздел первого уровня.

На каждый источник в основном тексте должна быть ссылка.

3. Работа OLE-203. Объектная модель файловой системы

Цели:
- изучение файловой объектной модели.
Задачи:
- управление дисками;
- управление каталогами;
- управление файлами.
Опорные документы:

3.1. Рабочее пространство

Откроем Microsoft Excel. В настройках приложения нужно разрешить макросы и доступ к VBA, включить ленту «Разработчик».

Находясь в Microsoft Excel, нажмем Alt+F11. Откроется среда Visual Basic for Application (VBA). Здесь проще всего разработать необходимый сценарий.

Выберем в меню Insert — Module.

Появится новый модуль для кода.

Впишем в модуль процедуру, внутри которой будем проводить свои эксперименты:

```
Option Explicit

Sub Drives()

End Sub
```

Конструкция Option Explicit обозначает обязательно объявление переменных. Ключевое слово Sub обозначает процедуру (ключевое слово Function обозначает функцию, возвращающую значение), название процедуры Drives. End Sub обозначает конец процедуры (конец функции обозначается End Function).

Теперь нужно подключить программный компонент, отвечающий за работу с файловой моделью посредством автоматизации.

Выберем в меню Tools — References.

Найдем в появившемся диалоге строку Microsoft Scripting Runtime и включим флажок этой строки. Закроем диалог кнопкой OK.

Откроем обозреватель объектов, Object Browser. Для этого нажмем клавишу F2, или выберем в меню View — Object Browser. Обозреватель открывается как любой другой документ в многодокументном интерфейсе. Переключаться между документами можно при помощи Ctrl+F6.

Выберем в обозревателе библиотеку Scripting в первом списке вверху.

В результате увидим все элементы библиотеки.

Упорядочим их. В левой части, где перечисляются объекты и константы, в контекстном меню выберем Group Members.

Выберем коллекцию Drives в левой части.

В правой части при этом появится перечисление свойств и методов.

Видим, что у коллекции свойство Count и свойство Item. У свойства Item есть пометка (например, голубая точка). Она обозначает свойство по умолчанию, то, название которого можно не писать в коде.

3.2. Константы и переменные

Целью работы является формирование сценариев на VBS (Visual Basic Scripting Edition). Писать такие сценарии при помощи блокнота является непростой задачей, так как в сценариях отсутствуют подсказки и проверки кода во время редактирования, а также константы. Поэтому разрабатываем код сценария в VBA, затем копируем его в соответствующий файл vbs.

При написании кода в VBA будем пользоваться подсказками и проверками, а также константами. Для этого переменные нужно объявлять с указанием их типа. При переносе кода в сценарий vbs типы переменных должны быть удалены, а все константы явно описаны.

Переменные объявляются *в начале процедуры* или *в начале модуля* при помощи ключевого слова Dim, тип указывается при помощи ключевого слова As, причем для каждой переменной в списке тип указывается отдельно, например:

```
Dim A As Single, B As Single, I As Integer, J As Integer
```

Если для какой-то переменной тип не указан, она имеет тип по умолчанию Variant, который может принимать значение любого типа.

Пишем код, не соблюдая регистр, например, все строчные. После ухода с текущей строки происходит проверка, и все идентификаторы автоматически переводятся в нужный регистр. Если какой-то идентификатор не получил правильного регистра, значит, в его написании есть ошибка, или данный идентификатор отсутствует в области видимости.

Единственный случай, когда идентификатор пишется в правильном регистре — это его объявление. Это касается переменных, констант, процедур, функций.

Нам также нужно отключить навязчивое сообщение об ошибке.

Выберем в меню Tools — Options, и *выключим* флажок Auto Syntax Check. Теперь при написании синтаксически неверной конструкции она будет выделена красным цветом.

3.3. Перечисление дисков

Сначала исследуем возможности объектной модели файловой системы в части дисковых устройств. Коллекция Drives содержит все имеющиеся диски. Элементом коллекции является объект типа Drive, а перечисление элементов выполняется при помощи цикла For Each (для каждого).

Переключимся в модуль кода Ctrl+F6.

В процедуре Drives описываем создание объекта автоматизации и цикл перебора дисков:

```
Sub Drives()
    Dim fso As FileSystemObject, D As Drive
    Set fso = CreateObject("Scripting.FileSystemObject")
    For Each D In fso.Drives
    Next
End Sub
```

В нижней части есть два окна, Immediate и Watch.

Первое окно позволяет выполнять код непосредственно, и выводить в него контрольные значения, используя метод Debug.Print.

Второе окно позволяет наблюдать значения переменных.

Если окно Immediate отсутствует, включим его Ctrl+G. Если окно Watch отсутствует, включим его, выбрав в меню View — Watch Window.

Щелкнем правой кнопкой мыши на переменную D и выберем Add Watch. Добавим эту переменную для просмотра.

Установим курсор внутри процедуры Drives (неважно где).

Выполняем код шаг за шагом при помощи F8.

Когда выполнение дойдет до оператора Next, переменная D в окне Watch отобразит сведения о первом найденном диске. Возле названия переменной появится плюсик, щелкнем его. Раскроется список свойств.

Нажимая F8 еще несколько раз, увидим все диски.

Для выполнения кода процедуры до конца (или с начала до конца) нажмем F5.

Теперь будем выводить свойства дисков в окно Immediate:

```
For Each D In fso.Drives
    Debug.Print D.AvailableSpace
Next
```

Выполняем код при помощи F8. Наблюдаем, как в окне Immediate выводятся значения. Если в компьютере есть привод компакт-диска или дискеты, появится сообщение об ошибке «Диск не готов». В этом случае программу следует остановить, выбрав в меню Run — Reset.

Есть два способа избежать сообщения об ошибке. Первый — игнорировать ошибку, используя оператор On Error Resume Next (при ошибке перейти к следующей строчке кода). Второй — не вызывать ошибку. Сейчас лучше подойдет второй способ. Для этого воспользуемся свойством диска IsReady, указывающим на готовность диска, например, так:

```
For Each D In fso.Drives
    If D.IsReady Then
        Debug.Print "AvailableSpace = " & D.AvailableSpace
    End If
Next
```

Чтобы очистить окно Immediate, введем Ctrl+G, Ctrl+A, Delete.

Установим курсор внутри процедуры Drives.

Запустим процедуру F5. Наблюдаем вывод в окно Immediate.

Выведем теперь имя диска:

```
For Each D In fso.Drives
    If D.IsReady Then
        Debug.Print "Disk " & D.DriveLetter
        Debug.Print "AvailableSpace = " & D.AvailableSpace
    End If
Next
```

Выведем почти все другие свойства: DriveType, FileSystem, FreeSpace, SerialNumber, ShareName, TotalSize, VolumeName.

После того, как это будет сделано, нужно формировать код сценария.

В сценарии не обязательно объявлять переменные, поэтому мы их туда просто не включим. Констант в этом коде нет. Однако вывод значений в сценарии производится при помощи метода WScript.Echo.

Откроем FAR.

Создадим каталог C:\fso для сценариев, клавиша F7.

Выделим код процедуры Drives, за исключением строки объявления переменных. Выделять нужно именно строки. Установим курсор в начало строки «Set fso = …», держим Shift, нажимаем клавишу «Вниз».

После выделения сдвинем код влево при помощи Shift+Tab.

Скопируем код Ctrl+C, вернем сдвиг при помощи Tab.

Перейдем в FAR.

Перейдем в каталог C:\fso.

- создадим новый файл Shift+F4,
- введем название файла drives.fso,
- вставим скопированный код,
- сохраним F2.

Установим курсор в начало файла.

- нажмем Ctrl+F7 для замены строк,
- введем Debug.Print в поле «Искать»,
- введем WScript.Echo в поле «Заменить на»,
- нажмем Enter столько раз, сколько потребуется для всех замен.

Сохраним файл F2, закроем Escape.

Запустим файл Enter.

Придется закрыть несколько сообщений.

Запустим сценарий командой cscript drives.fso.

Сценарий должен вывести информацию в консоль.

Вернемся в Excel.

Сохраним книгу и файл с макросами в каталог C:\fso, название файлов книги и макросов fso.

3.4. Перечисление каталогов

Перечисление каталогов принципиально ничем не отличается от перечисления дисков, только используются другие объекты.

Опишем в модуле кода VBA процедуру Folders.

Скопируем в нее объявление и создание объекта FileSystemObject.

Опишем переменную для объекта каталога F типа Folder.

Опишем цикл For Each:

```
For Each F In fso.GetFolder("C:\").SubFolders
Next
```

Здесь мы используем метод GetFolder для того, чтобы получить объект каталога C:\, а затем используем коллекцию SubFolders.

Дальнейшие действия выполняются примерно так же, как это происходило при перечислении дисков.

По завершении разработки кода процедуры folders создадим файл folders.vbs, скопируем в него код процедуры, внесем необходимые изменения, тестируем и отлаживаем его.

3.5. Перечисление файлов

Перечисление каталогов принципиально ничем не отличается от перечисления каталогов, только используются другие объекты.

Скопируем процедуру Folders и вставим в модуль.

Переименуем процедуру в Files.

Изменим тип переменной F на File.

Изменим коллекцию SubFolders на Files.

Выводим все свойства файла.

Формируем сценарий files.vbs, тестируем и отлаживаем его.

3.6. Контрольные вопросы и упражнения

1. Назовите ProgID библиотеки Scripting.
2. Перечислите изученные объекты и коллекции библиотеки.
3. Перечислите свойства объекта Drive.
4. Перечислите свойства объекта Folder.
5. Перечислите свойства объекта File.
6. Разработайте сценарий для вывода всех каталогов и подкаталогов заданного каталога. Название сценария subfolders.vbs.

4. Работа OLE-204. Объектная модель Microsoft Excel

Цели:
- изучение объектной модели Microsoft Excel
Задачи:
- управление книгами;
- управление расчетными листами;
- управление ячейками:
- формирование диаграмм.
Опорные документы:

4.1. Рабочее пространство

Целью работы является написание сценариев на VBS, выполняющих отдельные, в основном простые действия с объектной моделью. Писать сценарии можно при помощи редактора FAR, а разрабатывать сценарии можно в VBA приложения Excel.

Каждый сценарий начинается с получения объекта автоматизации:

```
Set EA = CreateObject("Excel.Application")
```

Сценарий показывает приложение на экране, устанавливая свойство Application.Visible:

```
EA.Visible = True
```

Чтобы во время выполнения сценария приложение не показывало диалоги предупреждений, часто нужно отключить эти диалоги:

```
EA.DisplayAlerts = False
```

Далее, обращаясь к свойствам объекта Application, получаем другие объекты, например, коллекцию книг:

```
Set Books = EA.Workbooks
```

При этом нужно обращать внимание на то, когда и какие объекты возвращают те или иные свойства объекта автоматизации и других объектов. Для этой цели следует воспользоваться обозревателем объектов, встроенным в VBA Excel, выбрав в нем соответствующую библиотеку.

По завершении сценарий закрывает приложение методом Quit, если в описании сценария не сказано иное. Перед этим часто полезно приостановить выполнение сценария с тем, чтобы посмотреть результат, пока приложение не закрыто. Это можно сделать, вызвав диалог Open, который отменяется (руками).

Названия файлов сценариев для управления Microsoft Excel должны начинаться с префикса *xl*, и располагаться в каталоге C:\XL. Все книги так же располагаются в этом каталоге.

4.2. Управление книгами

Для управления книгами используем объекты Workbooks и Workbook.

4.2.1. Сценарий xlQuit

Выполняем некоторую, продолжительную работу, например, цикл с одним миллионом итераций, после чего закрываем приложение.

4.2.2. Сценарий xlOpen

Создаем книгу при помощи приложения Microsoft Excel (руками). В первую ячейку первого листа вводим текст Open, сохраняем книгу с именем Book01. Затем пишем собственно сценарий.

При помощи метода Workbooks.Open открываем созданную книгу.

Сценарий не должен закрывать приложение.

Закрываем приложение Alt+F4.

4.2.3. Сценарий xlDialogOpen

Книга Book01 должна существовать.

Показываем диалог «Open» при помощи коллекции диалогов Dialogs, диалог номер 1, метод Show.

После появления диалога выбираем в нем руками книгу Book01.

Сценарий закрывает приложение методом Quit.

4.2.4. Сценарий xlOpenTabbed

Создадим текстовый файл tabbed.txt, введем в него текст:

1 2 3

A B C

Нужно убедиться, что между символами находятся табуляторы.

В FAR для этого нужно установить F9 — Настройки редактора — Не преобразовывать табуляцию. После создания файла нужно убедиться, что вставлены табуляторы, код 9, используя клавиши F3, F4.

Используем метод Workbooks.Open, указывая путь к текстовому файлу и разделитель полей. В результате элементы файла должны быть расположены в разных ячейках листа. Сохраняем книгу с именем BookT.

4.2.5. Сценарий xlOpenNumbed

Создадим текстовый файл numbed.txt, введем в него текст:

1#2#3

A#B#C

Используем метод Workbooks.OpenText, указывая путь к текстовому файлу и разделитель полей #. В результате элементы файла должны быть расположены в разных ячейках листа. Сохраняем книгу с именем BookN.

4.2.6. Сценарий xlSaveAs

Открываем книгу Book01, сохраняем книгу с именем Book02.

4.2.7. Сценарий xlDialogSaveAs

Открываем книгу Book01.
Показываем диалог SaveAs при помощи коллекции диалогов, номер диалога найдем в обозревателе объектов VBA, метод Show.
Сохраняем книгу с именем Book03 при помощи диалога (руками).
Сценарий закрывает приложение методом Quit.

4.2.8. Сценарий xlSaveAs97

Открываем книгу Book01.
Сохраняем книгу в формате Excel-97-2003.
Сценарий закрывает приложение методом Quit.

4.3. Управление листами

Для управления листами используем объекты Worksheets и Worksheet.

4.3.1. Сценарий xlWorksheetAdd

Добавляем книгу, добавляем в книгу лист, название листа Add1.
Сохраняем книгу с именем Book04.

4.3.2. Сценарий xlWorksheetDelete

Открываем книгу Book04.
Если в ней есть листы, кроме листа Add1, удалим их.
Сохраняем книгу с именем Book05.

4.3.3. Сценарий xlWorksheetCopy

Открываем книгу Book05.
Создаем копию листа Add1 с именем Add2, помещаем лист в начало книги. Создаем копию листа Add1 с именем Add3, помещаем лист в конец книги. Сценарий не должен создавать листы Add2 и Add3, если они существуют, но должен располагать их в порядке Add2, Add1, Add3.
Сохраняем книгу с именем Book06, затем с именем Book04.

4.3.4. Сценарий xlWorksheetMove

Открываем книгу Book04.
Сценарий должен располагать листы в порядке Add1, Add2, Add3.
Перемещаем лист Add2 за листом Add1.
Сохраняем книгу с именем Book07.

Проверяем правильность работы сценария xlWorksheetDelete после работы этого сценария.

4.3.5. Сценарий xlWorksheetActivate1

Открываем книгу Book07.
Выбираем (активируем) лист Add2.
Сохраняем книгу с именем Book072.

4.3.6. Сценарий xlWorksheetActivate2

Открываем книгу Book07.
Выбираем (активируем) лист Add3.
Сохраняем книгу с именем Book073.

4.3.7. Сценарий xlWorksheetActivate3

Открываем книгу Book07.
Выбираем (активируем) лист Add1.
Сохраняем книгу с именем Book071.

4.4. Чтение и запись ячеек

Для доступа к ячейкам используются свойства Cells и Range, возвращающие объект типа Range. Значения ячеек записываются присваиванием непосредственного значения или значения переменной V, а считываются в переменную V.

4.4.1. Сценарий xlWsCellsSet

Открываем книгу Book02. Записываем в ячейки B2, C3, E5 значения, равные их названиям, то есть B2, C3, E5, используя свойство Cells.
Сохраняем книгу.

4.4.2. Сценарий xlWsRangeSet

Открываем книгу Book02.
Записываем в диапазон ячеек A6:E8 значения от 1 до 15, значения от 1 до 5 располагаются в строке 6, от 6 до 10 в строке 7, от 11 до 15 в строке 8.
Формируем значения в массиве V и записываем в диапазон, возвращаемый свойством Range.

4.4.3. Сценарий xlRangeGet

Открываем книгу Book02.
Считываем значения из диапазона A6:E8 в массив V.
Выводим массив V при помощи Wscript.Echo в виде таблицы.

4.4.4. Сценарий xlSelectRow

Открываем книгу Book01. Выделяем строку 5.
Сохраняем книгу.

4.4.5. Сценарий xlSelectCol

Открываем книгу Book02. Выделяем столбцы 7 и 9.
Сохраняем книгу.

4.5. Оформление ячеек

Для оформления ячеек используются объекты Range, Interior, Font, коллекция Borders.

4.5.1. Сценарий xlBorders

Открываем книгу Book071.
Устанавливаем границы ячеек.
Ячейка B2 — тонкая рамка.
Ячейка D4 — толстая рамка.
Ячейка F6 — все возможные границы (всего 8).
Диапазон B8:E10 — внешняя граница толстая, внутренние тонкие.
Сохраняем книгу.

4.5.2. Сценарий xlInterior

Открываем книгу Book03.
Записываем в ячейку B2 слово Tahoma32, выбираем ее.
Задаем для ячейки шрифт Tahoma, размер 32, жирный, наклонный, подчеркнутый одинарной линией, перечеркнутый, красный.
Задаем цвет ячейки желтый.
Задаем ширину столбца «B» равной 80 единиц.
Задаем высоту строки «1» равной 80 единиц.
Задаем выравнивание по горизонтали по центру.
Задаем выравнивание по вертикали по центру.
Задаем нижнюю границу ячейки средней толщины.

4.5.3. Сценарий xlProtect

Открываем книгу Book01.
Защищаем все ячейки листа, кроме ячейки B2.
Сохраняем книгу.
Вызываем диалог Open, чтобы посмотреть результат.
Убеждаемся, что изменить можно только ячейку B2.
Отменяем диалог Open (руками).
Сценарий закрывает приложение методом Quit.

4.6. Создание формул и диаграмм

Название сценария xlChart.
Действия сценария:
- создает объект автоматизации,
- устанавливает свойство Visible,
- добавляет книгу,
- задает значения от 0 до 8 через 0,25 в диапазоне A1:A33,
- задает формулы "=sin()" в ячейках диапазона B1:B33,
- строит график на текущем листе,
- сохраняет книгу с именем Chart01,
- закрывает приложение методом Quit.

4.7. Контрольные вопросы и упражнения

1. Перечислите основные объекты Microsoft Excel.
2. Опишите методы и свойства объекта Application.
3. Опишите методы объекта Workbooks.
4. Опишите методы объекта Workbook.
5. Опишите методы объекта Worksheets.
6. Опишите методы объекта Worksheet.
7. Опишите свойства для доступа к ячейкам.

5. Работа OLE-205. Объектная модель Microsoft Word

Цели:
- изучение объектной модели Microsoft Word

Задачи:
- управление документами;
- управление текстом;
- поиск и замена;
- проверка правописания.

Опорные документы:

5.1. Рабочее пространство

Целью работы является написание сценариев на VBS, выполняющих отдельные, в основном простые действия с объектной моделью. Писать сценарии можно при помощи редактора FAR, а разрабатывать сценарии можно в VBA приложения Excel.

Каждый сценарий начинается с получения объекта автоматизации:

```
Set WA = CreateObject("Word.Application")
```

Сценарий показывает приложение на экране, устанавливая свойство Application.Visible:

```
WA.Visible = True
```

Чтобы во время выполнения сценария приложение не показывало диалоги предупреждений, часто нужно отключить эти диалоги:

```
WA.DisplayAlerts = False
```

Далее, обращаясь к свойствам объекта Application, получаем другие объекты, например, коллекцию открытых документов:

```
Set Docs = WA.Documents
```

По завершении сценарий закрывает приложение методом Quit, если в описании сценария не сказано иное.

Названия файлов сценариев для управления Microsoft Word должны начинаться с префикса *wd*, и располагаться в каталоге C:\WD.

Все создаваемые документы так же располагаются в этом каталоге.

5.2. Управление документами

5.2.1. Сценарий wdQuit

Сценарий создает объект автоматизации, показывает приложение на экране, выполняет некоторую продолжительную работу, закрывает приложение методом Quit. Следует убедиться, что приложение закрыто, при помощи диспетчера задач.

5.2.2. Сценарий wdOpen

Сначала создадим в каталоге C:\WD документ, содержащий строку текста «Hello». Название файла Hello.

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ методом Documents.Open.

Метод Quit не используется.

5.2.3. Сценарий wdDialogOpen

Сценарий создает объект автоматизации, показывает приложение на экране, вызывает диалог «Открыть», после чего пользователь выбирает в нем файл Hello.

Метод Quit не используется.

5.2.4. Сценарий wdSaveAs

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Hello, сохраняет его с именем World, используя метод Document.SaveAs.

Сценарий закрывает приложение методом Quit.

5.2.5. Сценарий wdDialogSaveAs

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Hello, вызывает диалог «Сохранить как».

Пользователь отменяет диалог.

Сценарий закрывает приложение методом Quit.

5.2.6. Сценарий wdSaveAs97

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Hello, сохраняет документ в формате Word-97.

Сценарий закрывает приложение методом Quit.

5.3. Управление текстом

5.3.1. Сценарий wdParaAfter

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Hello, добавляет параграф в конец документа, используя метод Range.InsertAfter, текст параграфа «After», сохраняет документ с именем HelloAfter.

Предварительно требуется получение объекта Range.

Сценарий закрывает приложение методом Quit.

5.3.2. Сценарий wdParaBefore

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Hello, добавляет параграф в начало документа, используя метод Range.InsertBefore, текст параграфа «Before», сохраняет документ с именем HelloBefore.

Сценарий закрывает приложение методом Quit.

5.3.3. Сценарий wdFont

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ HelloAfter, для всех букв слова Hello выбирает разные шрифты, размеры, курсивное и жирное начертание цвет.

Сценарий закрывает приложение методом Quit.

5.4. Поиск и замена

Сначала нужно создать в каталоге C:\WD новый документ Finds, содержащий 10 параграфов. Каждый параграф содержит одно слово:

Apple
Apple
Combine
Commodore
Conference
Continue
Continue
Elephant
Elephant
Resume

5.4.1. Сценарий wdFind

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Finds. Сценарий ищет следующие слова:

- Elephant;
- начинающиеся с букв Con;
- содержащие в себе букву a.

Сценарий не закрывает приложение.

5.4.2. Сценарий wdReplace

Сценарий создает объект автоматизации, показывает приложение на экране, открывает документ Finds. Сценарий делает следующие замены:

- Apple на Microsoft;
- Con на Un;
- m на X.

Сценарий не закрывает приложение.

5.5. Проверка правописания

5.5.1. Сценарий wdGrammar

Сценарий создает объект автоматизации, показывает приложение на экране. Сценарий проверяет правописание заданного слова, выводит все предположения его правильного написания. Слово задается непосредственно в тексте сценария.

Сценарий закрывает приложение методом Quit.