

# Лекция 5.

## Ввод и вывод информации

РХТУ им. Д.И. Менделеева

Каф. ИКТ

Курс создал: ст. преп. А.М. Васецкий

2013 г

## Встроенные диалоговые окна

В проектах VBA встречаются две разновидности диалоговых окон: **окна сообщений (MsgBox)** и **окна ввода (InputBox)**.

Они встроены в VBA, и если их возможностей достаточно, то можно обойтись без проектирования диалоговых окон. Окно сообщений выводит простейшие сообщения для пользователя, а окно ввода обеспечивает ввод информации.

# Функция InputBox

Выводит на экран диалоговое окно, содержащее сообщение и поле ввода, устанавливает режим ожидания ввода текста пользователем или нажатия кнопки, а затем возвращает значение типа **string**, содержащее текст, введенный в поле.

Синтаксис:

**InputBox (prompt [, title] [, default] [, xpos][, ypos] [, helpfile, context])**

# Аргументы функции InputBox

**prompt** – строковое выражение, отображаемое как сообщение в диалоговом окне. Строковое значение **prompt** может содержать несколько строк. Для разделения строк допускается использование символов **chr(10)**, **chr(13)** и их комбинации

**title** – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения

**default** – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым

**xpos** – числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана. Если этот аргумент опущен, диалоговое окно выравнивается по центру экрана по горизонтали

**ypos** – числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана. Если этот аргумент опущен, диалоговое окно помещается по вертикали примерно на одну треть высоты экрана

# Аргументы функции InputBox (продолжение)

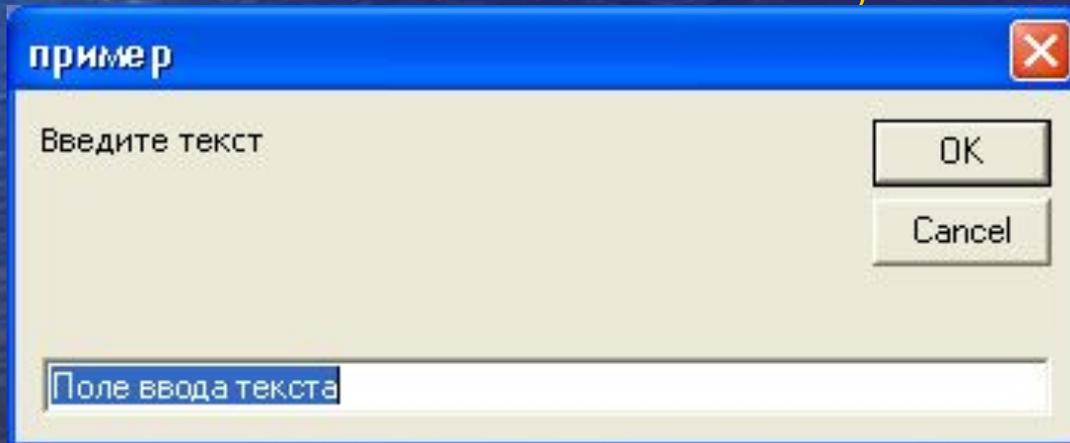
**helpfile** – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо наличие также аргумента **context**

**context** – числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот аргумент указан, необходимо наличие также аргумента **helpfile**

# Пример InputBox

```
10: str = InputBox("Введите текст", "пример", "Поле ввода текста")
```

```
If Len(str) > 0 Then  
    MsgBox "Введено:" & vbCrLf & str, vbExclamation +  
vbOKOnly  
Else  
    i = MsgBox("Текст не введен!", vbCritical +  
vbRetryCancel)  
    If i = vbRetry Then  
        GoTo 10  
    Else  
        MsgBox "Отменено пользователем", vbExclamation +  
vbOKOnly  
    End If  
End If
```



## Процедура MsgBox

Выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем, а затем возвращает значение типа **Integer**, указывающее, какая кнопка была нажата.

Синтаксис:

**MsgBox (prompt [, buttons] [, title] [, helpfile, context])**

# Аргументы функции MsgBox

**prompt** – строковое выражение, отображаемое как сообщение в диалоговом окне

**buttons** – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию этого аргумента равняется 0. Значения констант, определяющих число и тип кнопок используемого значка, приведены на следующих слайдах

**title** – строковое выражение, отображаемое в строке, заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения

**helpfile** – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо указать также аргумент **context**

**context** – числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот аргумент указан, необходимо указать также аргумент **helpfile**

# Значения аргумента `buttons`, определяющие кнопки

<b><code>vbOKOnly</code></b>	<b>0</b>	
<b><code>vbOKCancel</code></b>	<b>1</b>	
<b><code>vbAbortRetryIgnore</code></b>	<b>2</b>	
<b><code>vbYesNoCancel</code></b>	<b>3</b>	
<b><code>vbYesNo</code></b>	<b>4</b>	
<b><code>vbRetryCancel</code></b>	<b>5</b>	

# Значения аргумента `buttons`, определяющие отображаемые информационные значки

<b><code>vbCritical</code></b>	<b>16</b>	MsgBox "Пример", vbOKOnly + vbCritical
<b><code>vbQuestion</code></b>	<b>32</b>	MsgBox "Пример", vbOKOnly + vbQuestion
<b><code>vbExclamation</code></b>	<b>48</b>	MsgBox "Пример", vbOKOnly + vbExclamation
<b><code>vbInformation</code></b>	<b>64</b>	MsgBox "Пример", vbOKOnly + vbInformation 

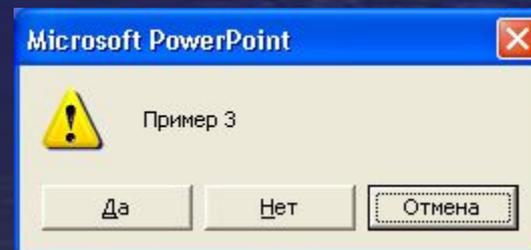
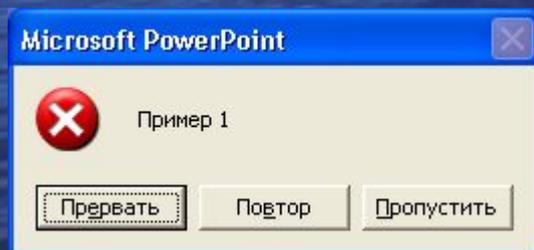


# Значения аргумента buttons определяющие основную кнопку

Константа	Значение	Номер основной кнопки
<b>vbDefaultButton1</b>	<b>0</b>	<b>1</b>
<b>vbDefaultButton2</b>	<b>256</b>	<b>2</b>
<b>vbDefaultButton3</b>	<b>512</b>	<b>3</b>
<b>vbDefaultButton4</b>	<b>768</b>	<b>4</b>

## Примеры:

1. MsgBox "Пример 1", vbAbortRetryIgnore + vbCritical + vbDefaultButton1
2. MsgBox "Пример 2", vbRetryCancel + vbQuestion + vbDefaultButton2
3. MsgBox "Пример 3", vbYesNoCancel + vbExclamation + vbDefaultButton3



# Возвращаемые значения окна MsgBox

Константа	Значение	Нажатая кнопка
<b>vbOK</b>	<b>1</b>	<b>ОК</b>
<b>vbCancel</b>	<b>2</b>	<b>Отмена (Cancel)</b>
<b>vbAbort</b>	<b>3</b>	<b>Прервать (Abort)</b>
<b>vbRetry</b>	<b>4</b>	<b>Повторить (Retry)</b>
<b>vbIgnore</b>	<b>5</b>	<b>Пропустить (Ignore)</b>
<b>vbYes</b>	<b>6</b>	<b>Да (Yes)</b>
<b>vbNo</b>	<b>7</b>	<b>Нет (No)</b>

## Пример 1.

```
i = MsgBox("Нажмите одну из клавиш",  
vbYesNoCancel + vbInformation + vbDefaultButton1)
```

## Пример 2.

```
i = MsgBox("Нажмите одну из клавиш",  
vbAbortRetryIgnore + vbCritical + vbDefaultButton2)
```

# Типы файлов в VBA

## Файл последовательного доступа

Рассматривается как последовательность строк произвольной длины, разделенных специальными символами. Чтение и запись в файл производится построчно

## Файл произвольного доступа

Состоит из записей фиксированной длины и размер записи указывается при его открытии. Это позволяет локализовать любую запись в файле по ее номеру

## Бинарный файл

Является частным случаем файла произвольного доступа. Размер записи в бинарном файле считается равным 1 байту

# Открытие файла

## Open

Открывает файл для работы с ним.

Синтаксис: **Open Путь For Режим [Access Доступ] [Блокировка] As [ # ] НомерФайла [Len=Длина]**

**Путь** – строковое выражение, указывающее имя файла

**Режим** – устанавливает режим работы с файлом.

Допустимые значения: **Append, Binary, Input, Output** или **Random**

**Доступ** – устанавливает операции, разрешенные с файлом.

Допустимые значения: **Read, Write** или **Read Write**

**Блокировка** – устанавливает операции, разрешенные с открытым файлом другим процессам.

Допустимые значения: **Shared, Lock Read, Lock Write** и **Lock Read Write**

**номерФайла** – допустимый номер файла (1-255). Параметру

**НомерФайла** предшествует символ **#**. Значение **номерФайла** нельзя изменять, пока файл открыт.

**длина** – число,  $\leq 32767$  (байт). Для файлов, открытых в режиме **Random**, это значение является длиной записи. Для файлов с последовательным доступом это значение является числом буферизуемых символов

# Заккрытие файла

## Close

Завершает операции ввода/вывода с файлом, открытым с помощью инструкции **open**. Эта инструкция очищает буфер и указывает операционной системе обновить таблицу размещения файлов.

**Каждый файл по завершении работы с ним должен быть закрыт, иначе это может привести к частичной потере информации.**

Синтаксис: **Close [СписокНомеровФайлов]**

Аргумент **СписокНомеровФайлов** может представлять один или несколько номеров файлов. При этом используется следующий синтаксис, где **номерФайла** представляет любой допустимый номер файла: **[[#] номерФайла] [, [#] номерФайла]...**

Если опустить список аргументов, то будут закрыты все открытые файлы

# Прочие функции для работы с файлами

## Reset

Закрывает все активные файлы, открытые с помощью инструкции **open**, и записывает содержимое всех буферов файлов на диск, открытых с помощью инструкции **Open**

## FreeFile

Функция возвращает доступный номер, который может использоваться в инструкции **Open**

```
Sub FilOpen()
```

```
' Открывает файл для ввода данных из него.
```

```
' Файл должен существовать
```

```
Open "File.inp" For Input As #1
```

```
' Открывает файл для бинарного вывода
```

```
Open "Filebin.txt" For Binary Access Write As #2
```

```
' Открывает файл для вывода
```

```
' Другим процессам разрешено пользоваться этим файлом для
```

```
' операций ввода/вывода
```

```
Open "FileOutput.txt" For Output Shared As #3
```

```
Close
```

```
End Sub
```

# File System Objects (FSO)

FSO позволяет открывать файлы как текстовой поток и читать и писать в них **object.OpenTextFile(filename[, iomode[, create[, format]])**

□ **filename** – полный путь к файлу

□ **iomode** – режим ввода/вывода

□ **create** – Логическое значение, показывающее будет ли создан новый файл, если файл **filename** не найден (**False** – по умолчанию)

□ **format** – формат открываемого файла. По умолчанию - ASCII

# Значения констант для открытия файлов через FSO

*iomode* принимает следующие значения:  
**ForReading = 1** – открывает файл для чтения  
**ForAppending = 8** – открывает файл для добавления

*format* принимает следующие значения:  
**TristateUseDefault = -2** для открытия файла используются системные установки  
**TristateTrue = -1** – открывает файл в кодировке Unicode.  
**TristateFalse = 0** – открывает файл в кодировке ASCII

# Открытие файла, как объекта FSO

## Метод **OpenTextFile**

```
Sub OpenTextFileTest()  
Const ForReading = 1, ForWriting = 2  
Const ForAppending = 8  
Const TristateFalse = 0  
Dim fs, f  
Set fs=CreateObject("Scripting.FileSystemObject")  
Set f = fs.OpenTextFile("c:\testfile.txt",  
ForAppending,TristateFalse)  
    f.Write "Hello world!"  
    f.Close  
End Sub
```

## OpenAsTextStream метод

Метод открывает файл и делает возможным чтение/запись в него

*object.OpenAsTextStream([iomode, [format]])*

□ *object* – файловый объект

□ *iomode* – режим записи/чтения

□ *format* – формат файла

# Значения констант OpenAsTextStream

*iomode* принимает следующие значения:

**ForReading = 1** – открывает файл для чтения

**ForWriting = 2** – открывает файл для записи.

Если файл существовал, то его содержимое предварительно стирается

**ForAppending = 8** – открывает файл для добавления

*format* принимает следующие значения:

**TristateUseDefault = -2** для открытия файла

используются системные установки

**TristateTrue = -1** – открывает файл в кодировке Unicode.

**TristateFalse = 0** – открывает файл в кодировке ASCII

# Пример

```
Sub TextStreamTest()  
    Const ForReading = 1, ForWriting = 2, ForAppending = 3  
    Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0  
    Dim fs, f, ts, s  
    Set fs = CreateObject("Scripting.FileSystemObject")  
    fs.CreateTextFile "test1.txt"           'Create a file  
    Set f = fs.GetFile("test1.txt")  
    Set ts = f.OpenAsTextStream(ForWriting, TristateUseDefault)  
    ts.Write "Hello World"  
    ts.Close  
    Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)  
    s = ts.ReadLine  
    MsgBox s  
    ts.Close  
End Sub
```

# FSo. Список папок

```
Private Sub GetFolders()  
Dim FSO As Object, Drive As Object, Folders As Object  
Dim Folder As Object  
    ' формируем ссылку на объект FileSystemObject  
Set FSO = CreateObject("Scripting.FileSystemObject")  
    ' получаем доступ к диску D:\  
Set Drive = FSO.GetFolder("D:\")  
    ' формируем коллекцию каталогов  
Set Folders = Drive.SubFolders  
With ListBox1 'устанавливаем свойства списка  
    .ColumnHeads = False  
    .MultiSelect = fmMultiSelectSingle  
End With  
For Each Folder In Folders      'Начинаем перебор коллекции  
    ListBox1.AddItem Folder.Name  
Next  
End Sub
```

# Функции по работе с файлами

<b>ChDir</b>	Изменяет текущую папку. Синтаксис: <b>ChDir</b> путь
<b>ChDrive</b>	Изменяет текущий диск. Синтаксис: <b>ChDrive</b> диск. Например, <b>ChDrive "D"</b>
<b>CurDir</b>	Функция возвращает текущую папку
<b>Dir</b>	<p>Возвращает строку, представляющую имена файла и папки, соответствующие определённому критерию или метку диска</p> <p>Синтаксис: <b>Dir</b>[(Путь[, атрибуты])]</p> <ul style="list-style-type: none"><li>□ <b>Путь</b> – путь к файлу/папке</li><li>□ <b>Атрибуты</b> – атрибуты файла/папки (См. ниже)</li></ul>
<b>FileAttr</b>	<p>Возвращает значение типа <b>Long</b>, представляющее режим файла, открытого с помощью инструкции <b>open</b>.</p> <p>Возвращаемые значения: <b>1</b> (для режима <b>input</b>), <b>2</b> (<b>output</b>), <b>4</b> (<b>Random</b>), <b>8</b> (<b>Append</b>) и <b>32</b> (<b>Binary</b>).</p> <p>Синтаксис: <b>FileAttr</b> (НомерФайла, Тип)</p> <ul style="list-style-type: none"><li>□ <b>НомерФайла</b> – допустимый номер файла</li><li>□ <b>Тип</b> – число, указывающее характер возвращаемых данных. Если тип установлен равным 1, то функция <b>FileAttr</b> возвращает значение, указывающее режим работы файла</li></ul>

# Функции по работе с файлами (продолжение)

<b>GetAttr</b>	Возвращает значение типа <b>integer</b> , определяющее атрибуты файла, каталога или папки. Значение, возвращаемое функцией <b>GetAttr</b> , является суммой констант, приведенных ниже. Синтаксис: <b>GetAttr (путь)</b>
<b>SetAttr</b>	Устанавливает атрибуты файла. Синтаксис: <b>SetAttr pathname, attributes</b> Атрибуты в аргументе <b>attributes</b> определяются как сумма констант из таблицы ниже.

## Константы атрибутов файла

Константа	Значение	Описание
<b>vbNormal</b>	<b>0</b>	Обычный
<b>vbReadOnly</b>	<b>1</b>	Только чтение
<b>vbHidden</b>	<b>2</b>	Скрытый
<b>vbSystem</b>	<b>4</b>	Системный
<b>vbDirectory</b>	<b>16</b>	Каталог или папка
<b>vbArchive</b>	<b>32</b>	Файл был изменен после последнего резервирования

# Функции по работе с файлами (продолжение)

## FileCopy

Копирует файл.

Синтаксис: **FileCopy** source, destination

Аргументы:

- **source** – строковое выражение, указывающее имя копируемого файла
- **destination** – строковое выражение, указывающее имя результирующего файла. Аргумент **destination** может содержать имя каталога или папки и диска

## FileDateTime

Функция возвращает дату и время последнего изменения файла.

Синтаксис: **FileDateTime**(путь )

# Функции по работе с файлами (продолжение)

<b>Kill</b>	<p>Удаляет существующий файл. Синтаксис: <b>Kill</b> путь</p> <p>В аргументе путь допустимо использование символов (*) и (?) для удаления нескольких файлов по маске.</p>
<b>MkDir</b>	<p>Создает новую папку. Синтаксис: <b>MkDir</b> путь</p>
<b>Rmdir</b>	<p>Удаляет существующую папку. Синтаксис: <b>Rmdir</b> путь</p>

# Примеры. Функция DIR

```
Sub Get_All_File_from_Folder()  
    Dim sFolder As String, sFiles As String  
    'диалог запроса выбора папки с файлами  
    With Application.FileDialog(msoFileDialogFolderPicker)  
        If .Show = False Then Exit Sub  
        sFolder = .SelectedItems(1)  
    End With  
    sFolder = sFolder & IIf(Right(sFolder, 1) = Application.PathSeparator, _  
        "", Application.PathSeparator)  
    'отключаем обновление экрана, чтобы наши действия не мелькали  
    Application.ScreenUpdating = False  
    sFiles = Dir(sFolder & "*.xls*")  
    Do While sFiles <> ""  
  
        Workbooks.Open sFolder & sFiles 'открываем книгу  
        'действия с файлом  
        'Запишем на первый лист книги в ячейку A1 - test  
        'Правда здесь есть вероятность ошибки, если первый лист имеет тип Chart  
        ActiveWorkbook.Sheets(1).Range("A1").Value = "test"  
        'Закрываем книгу с сохранением изменений  
        ActiveWorkbook.Close True 'если поставить False - книга будет  
        'закрыта без сохранения  
        sFiles = Dir 'Ищем следующий файл  
    Loop  
    'возвращаем ранее отключенное обновление экрана  
    Application.ScreenUpdating = True  
End Sub
```

# Ввод данных в файл последовательного доступа.

## Оператор Write

Записывает неформатированные данные в файл последовательного доступа. В отличие от инструкции **Print**, инструкция **Write** вставляет запятые между элементами и заключает строки в кавычки по мере записи их в файл.

Синтаксис:

**Write #НомерФайла, [СписокВывода]**

- **НомерФайла** – номер файла
- **СписокВывода** – выражение или список выражений, записываемых в файл.

Данные, записанные с помощью инструкции **Write**, обычно считываются из файла с помощью инструкции **Input**

# Ввод данных в файл последовательного доступа.

## Оператор Print

Записывает форматированные данные в файл последовательного доступа. Синтаксис:

**Print #НомерФайла, [СписокВывода]**

- **номерФайла** – номер файла
- **списокВывода** – выражение (или список выражений), записываемое в файл.

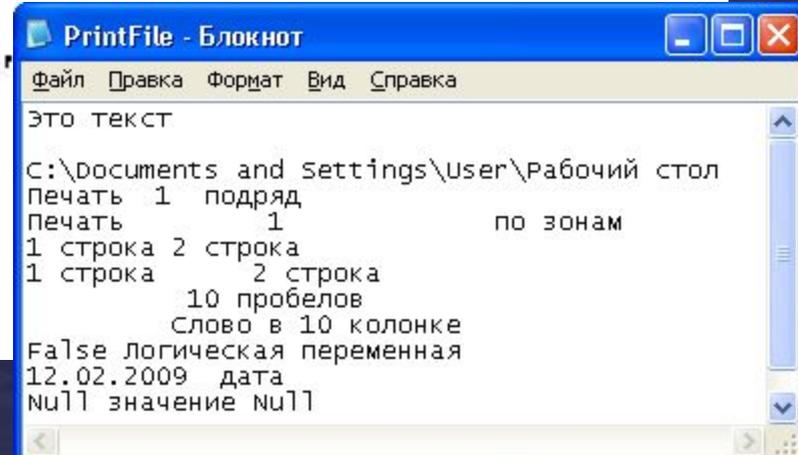
В аргументе **СписокВывода** разделителем списка выводимых выражений является ";" (данные выводятся подряд) или "," (данные выводятся по зонам). Кроме того, в аргументе **СписокВывода** допускается использование функций **Spc** и **Tab**:

**Spc(n)** – используется для вставки **n** пробелов в файл

**Tab(n)** – устанавливает курсор в столбец с номером **n**

# Пример использования оператора Print

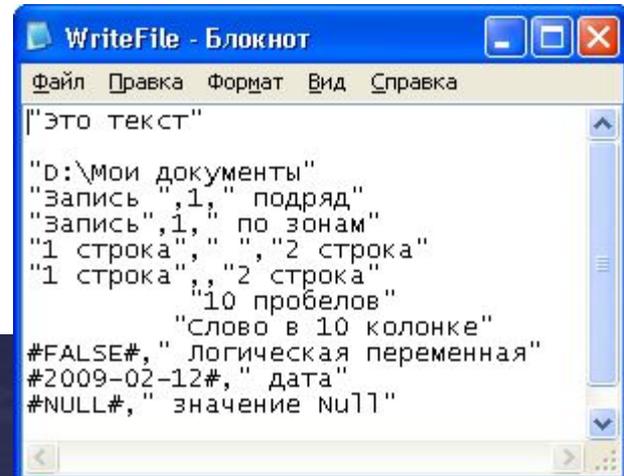
```
Sub PrintFun()  
' Открываем файл для записи  
Open "PrintFile.txt" For Output As #1  
' Вывод в файл текстовой строки  
Print #1, "Это текст"  
Print #1, ' пустая строка.  
Print #1, CurDir ' Текущая папка (по умолчанию папка исходного файла)  
Print #1, "Печать "; 1; " подряд" ' Печать подряд  
Print #1, "Печать", 1, " по зонам" ' Печать в 2 зоны  
Print #1, "1 строка"; " "; "2 строка" ' Строки с пробелом  
Print #1, "1 строка"; Tab; "2 строка" ' Строки с табулятором  
Print #1, Spc(10); "10 пробелов"  
Print #1, Tab(10); "Слово в 10 колонке"  
  
Dim MyBool, MyDate, MyNull, MyError  
MyBool = False: MyDate = #2/12/2009#: MyNull = Null  
  
Print #1, MyBool; " Логическая переменная"  
Print #1, MyDate; " дата"  
Print #1, MyNull; " значение Null"  
  
Close #1  
End Sub
```



```
PrintFile - Блокнот  
Файл Правка Формат Вид Справка  
Это текст  
C:\Documents and Settings\User\Рабочий стол  
Печать 1 подряд  
Печать 1 по зонам  
1 строка 2 строка  
1 строка 2 строка  
10 пробелов  
Слово в 10 колонке  
False логическая переменная  
12.02.2009 дата  
Null значение Null
```

# Пример применения оператора Write

```
Sub WriteFun()  
'Открываем файл для записи  
Open "WriteFile.txt" For Output As #1  
'Вывод в файл текстовой строки  
Write #1, "Это текст"  
Write #1, 'пустая строка.  
Write #1, CurDir 'Текущая папка (по умолчанию папка исходного файла)  
Write #1, "Запись "; 1; " подряд" ' Запись подряд  
Write #1, "Запись", 1, " по зонам" ' Запись в 2 зоны  
Write #1, "1 строка"; " "; "2 строка" 'Строки с пробелом  
Write #1, "1 строка"; Tab; "2 строка" 'Строки с табулятором  
Write #1, Spc(10); "10 пробелов"  
Write #1, Tab(10); "Слово в 10 колонке"  
  
Dim MyBool, MyDate, MyNull, MyError  
MyBool = False: MyDate = #2/12/2009#: MyNull = Null  
  
Write #1, MyBool; " Логическая переменная"  
Write #1, MyDate; " дата"  
Write #1, MyNull; " значение Null"  
  
Close #1  
End Sub
```



```
WriteFile - Блокнот  
Файл Правка Формат Вид Справка  
|"Это текст"  
"D:\Мои документы"  
"Запись 1, подряд"  
"Запись 1, по зонам"  
"1 строка", " ", "2 строка"  
"1 строка", "2 строка"  
"10 пробелов"  
"Слово в 10 колонке"  
#FALSE#, " Логическая переменная"  
#2009-02-12#, " дата"  
#NULL#, " значение Null"
```

# Ввод данных из файла последовательного доступа. Оператор Input

<b>Input #</b>	<p>Считывает данные из открытого файла последовательного доступа и присваивает их переменным. Данные, считываемые с помощью инструкции <b>input #</b>, обычно записываются в файл с помощью инструкции <b>Write #</b>.</p> <p>Синтаксис: <b>Input #НомерФайла, СписокПеременных</b></p> <ul style="list-style-type: none"><li>□ <b>номерФайла</b> – номер файла</li><li>□ <b>СписокПеременных</b> – список переменных, которым следует присвоить значения, считанные из файла. Переменные в списке разделяются запятыми</li></ul>
<b>Input</b>	<p>Возвращает значение типа <b>string</b>, содержащее символы из файла, открытого в режиме <b>input</b> или <b>Binary</b>.</p> <p>Функция <b>input</b> считывает данные, записываемые в файл с помощью инструкции <b>Print #</b> или <b>Put</b>.</p> <p>Синтаксис: <b>Input (Число, [#] НомерФайла)</b></p> <ul style="list-style-type: none"><li>□ <b>Число</b> задает число возвращаемых символов. Если аргумент <b>Число</b> равен <b>1</b>, то производится посимвольное считывание данных.</li></ul>

# Ввод данных из файла последовательного доступа

<b>Line Input #</b>	<p>Считывает строку из открытого файла последовательного доступа и присваивает ее переменной типа <b>string</b>. Данные, считываемые с помощью инструкции <b>Line input #</b>, как правило, записываются в файл с помощью инструкции <b>Print #</b>.</p> <p>Синтаксис: <b>Line Input #НомерФайла, ИмяПеременной</b></p> <ul style="list-style-type: none"><li>□ <b>НомерФайла</b> – номер файла</li><li>□ <b>ИмяПеременной</b> – имя переменной типа <b>Variant</b> или <b>String</b></li></ul>
<b>EOF</b>	<p>Функция возвращает значение <b>True</b> при достижении конца файла.</p> <p>Синтаксис: <b>EOF (НомерФайла)</b></p> <p>При последовательном считывании информации из файла часто используется следующий цикл:</p> <p><b>Do While Not EOF(1) Loop</b></p> <p>или, эквивалентный цикл:</p> <p><b>While Not EOF (1) Wend</b></p> <p><b>Прим.</b> Для бинарных файлов вместо этой функции используется <b>LOF</b></p>

## Пример использования оператора Input

```
Dim MyChar
```

```
Open "FILE.tst" For Input As #1
```

‘ Цикл до конца файла

```
Do While Not EOF(1)
```

‘ Считываем 1 символ

```
MyChar = Input(1, #1)
```

‘ Вывод в окно Immediate

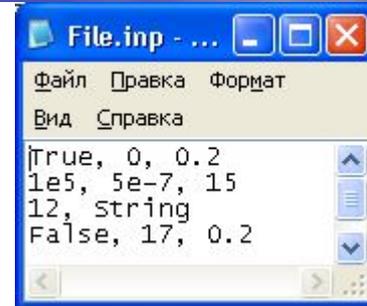
```
Debug.Print MyChar
```

```
Loop
```

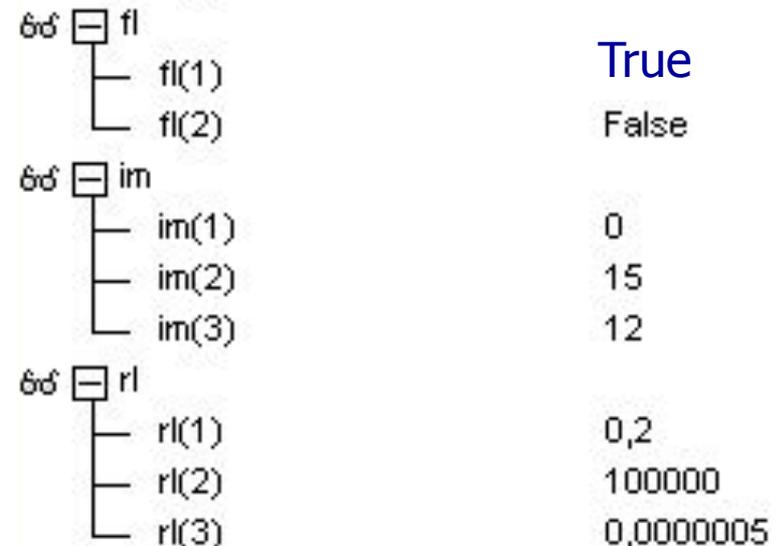
```
Close #1
```

# Пример

```
Sub FileInp()  
Dim IM(1 To 3) As Integer  
Dim RL(1 To 3) As Double  
Dim FL(1 To 2) As Boolean  
Dim St(1 To 5) As String
```



```
St(3) = "D:\Мои Документы-AMVAS\AMVAS-4_Work\Educate-Programming\Samples"  
ChDir (St(3)) 'изменяем путь  
St(4) = CurDir  
Open "File.inp" For Input Access Read As #1  
St(2) = EOF(1)  
'Do While Not EOF(1)  
    Input #1, FL(1), IM(1), RL(1)  
    Input #1, RL(2), RL(3), IM(2)  
    Input #1, IM(3), St(1)  
    Line Input #1, St(2)  
'Loop  
Close #1  
End Sub
```



St		
St(1)	"String"	
St(2)	"False, 17, 0.2"	
St(3)	"D:\Мои Документы-AMVAS\AMVAS-4_Work\Educate-Programming\Samples"	
St(4)	"D:\Мои Документы-AMVAS\AMVAS-4_Work\Educate-Programming\Samples"	

# Пример. Вывод массива в файл

```
Sub ArrOutput()  
Const M As Byte = 2, N As Byte = 5  
Const cDir = "D:\Мои Документы-AMVAS\AMVAS-4_Work\Educate-Programming\Samples"  
Dim Arr(M, N) As Integer  
Dim i As Byte, j As Byte  
Dim str As String  
  
ChDir (cDir)  
Open "file.out" For Output As #1  
Print #1, String(20, "*")  
Print #1, "Пример файла вывода массива"  
Print #1, String(20, "*")  
Print #1, "Размерности массива"  
Print #1, M; ", "; N  
Print #1, "<BEGIN>"  
For i = 0 To M  
    str = ""  
    For j = 0 To N  
        Arr(i, j) = i * 10 + j  
        str = str & CStr(Arr(i, j)) 'Заполнение строки  
        If j < N Then str = str & "," 'добавление разделителя  
    Next j  
    Print #1, str  
Next i  
Print #1, "<END>"  
Close #1  
End Sub
```

file - Блокнот

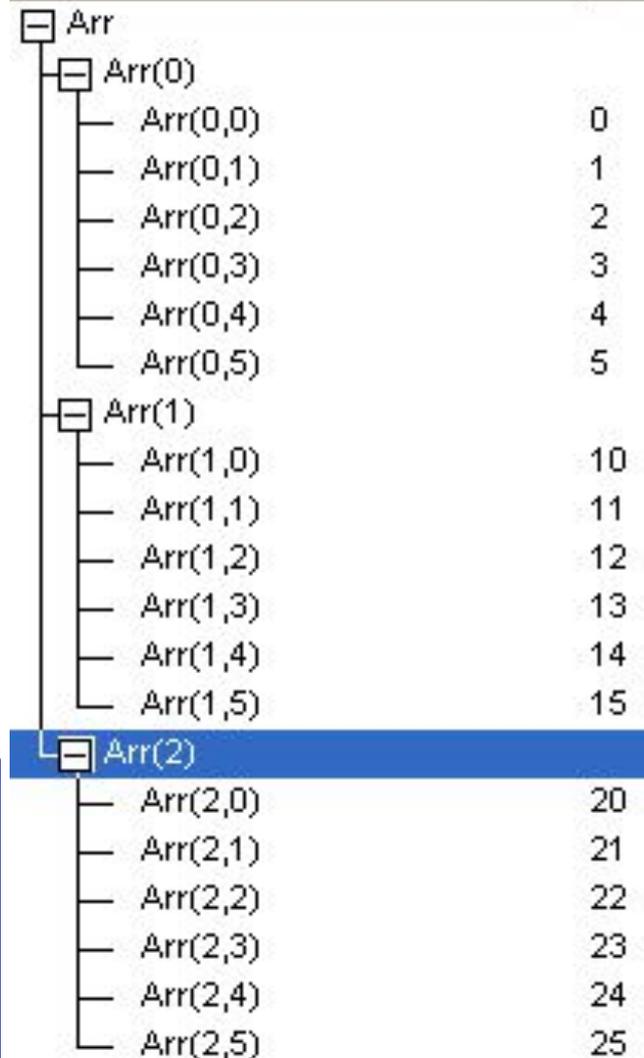
Файл Правка Формат Вид Справка

```
*****  
Пример файла вывода массива  
*****  
Размерности массива  
2 , 5  
<BEGIN>  
0,1,2,3,4,5  
10,11,12,13,14,15  
20,21,22,23,24,25
```

# Пример. Ввод массива из файла

```
Sub ArrInput ()  
  
Const cDir = "D:\Мои Документы-AMVAS\AMVAS-4_Work\Educate-Programming\Samples"  
Const stStart = "<BEGIN>"  
Dim M As Byte, N As Byte  
Dim i As Byte, j As Byte  
Dim Arr() As Integer  
Dim str As String * 10  
  
ChDir (cDir)  
Open "file.out" For Input As #1  
  
For i = 1 To 4  
    Input #1, str ' считывание строк комментариев  
Next i  
Input #1, M, N ' считывание размерностей  
ReDim Arr (M, N)  
  
Do  
    Input #1, str  
    str = Mid(str, 1, 7)  
Loop While Trim(str) <> stStart  
  
For i = 0 To M  
    For j = 0 To N  
        Input #1, Arr(i, j)  
    Next j  
Next i  
Close #1  
End Sub
```

```
file - Блокнот  
Файл  Правка  Формат  Вид  Справка  
*****  
Пример файла вывода массива  
*****  
Размерности массива  
2, 5  
<BEGIN>  
0,1,2,3,4,5  
10,11,12,13,14,15  
20,21,22,23,24,25
```



# Работа с файлом произвольного доступа

## Put

Записывает содержимое переменной в файл произвольного доступа.

Синтаксис: **Put [#] НомерФайла, [НомерЗаписи] ,  
ИмяПеременной**

- **НомерФайла** – номер файла
- **НомерЗаписи** – номер записи (режим **Random**) или номер байта (режим **Binary**), с которого следует начать запись. Если аргумент **НомерЗаписи** опущен, то записывается на то место, где был установлен указатель после выполнения последней инструкции **Get** или **Put**, либо куда он переведен с помощью функции **Seek**
- **ИмяПеременной** – имя переменной, содержащей данные, которые следует записать в файл

## Get

Читает данные из открытого файла произвольного доступа в переменную. Синтаксис:

**Get [#] НомерФайла, [НомерЗаписи], ИмяПеременной**

- **НомерФайла** – номер файла
- **НомерЗаписи** – номер записи (режим **Random**) или номер байта (режим **Binary**), с которого следует начать чтение
- **ИмяПеременной** – имя переменной, в которую следует поместить считанные данные

# Работа с файлом произвольного доступа

<b>Seek</b>	<p>Функция возвращает значение типа <b>Long</b>, определяющее текущее положение указателя чтения/записи внутри файла, открытого с помощью инструкции <b>Open</b>.</p> <p>Синтаксис: <b>Seek(НомерФайла)</b></p>
<b>LOF</b>	<p>Функция возвращает значение типа <b>Long</b>, представляющее размер файла в байтах, открытого с помощью инструкции <b>Open</b>. Для определения размера закрытого файла следует использовать функции <b>FileLen</b>.</p> <p>Синтаксис: <b>LOF(НомерФайла)</b></p>
<b>FileLen</b>	<p>Возвращает значение типа <b>Long</b>, содержащее размер файла в байтах.</p> <p>Синтаксис: <b>FileLen(Путь)</b></p>
<b>Loc</b>	<p>Возвращает текущую позицию указателя в файле.</p> <p>Синтаксис: <b>Loc(НомерФайла)</b></p>

# Пример записи

```
Type Record ' Пользовательский тип.  
  ID As Integer  
  Name As String * 20  
End Type
```

```
Dim MyRecord As Record, RecordNumber  
' Открываем для прямой записи  
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)  
For RecordNumber = 1 To 5 ' 5 оборотов цикла  
  MyRecord.ID = RecordNumber ' Заполняем поля  
  MyRecord.Name = "My Name" & RecordNumber  
' Записываем записи в файл  
  Put #1, RecordNumber, MyRecord  
Next RecordNumber  
Close #1
```

## Пример чтение из файла

‘Открываем тестовый файл

```
Open "TESTFILE" For Random As #1
```

```
Len = Len(MyRecord)
```

```
Position = 3 ' номер записи.
```

‘ читаем третью запись

```
Get #1, Position, MyRecord
```

```
Close #1
```

## Выбор файла пользователем

Если приложению необходимо получить от пользователя имя файла, то это можно сделать несколькими путями.

Например через `GetOpenFilename` объекта `Application`.

Более общим приёмом является использование метода `FileDialog`.

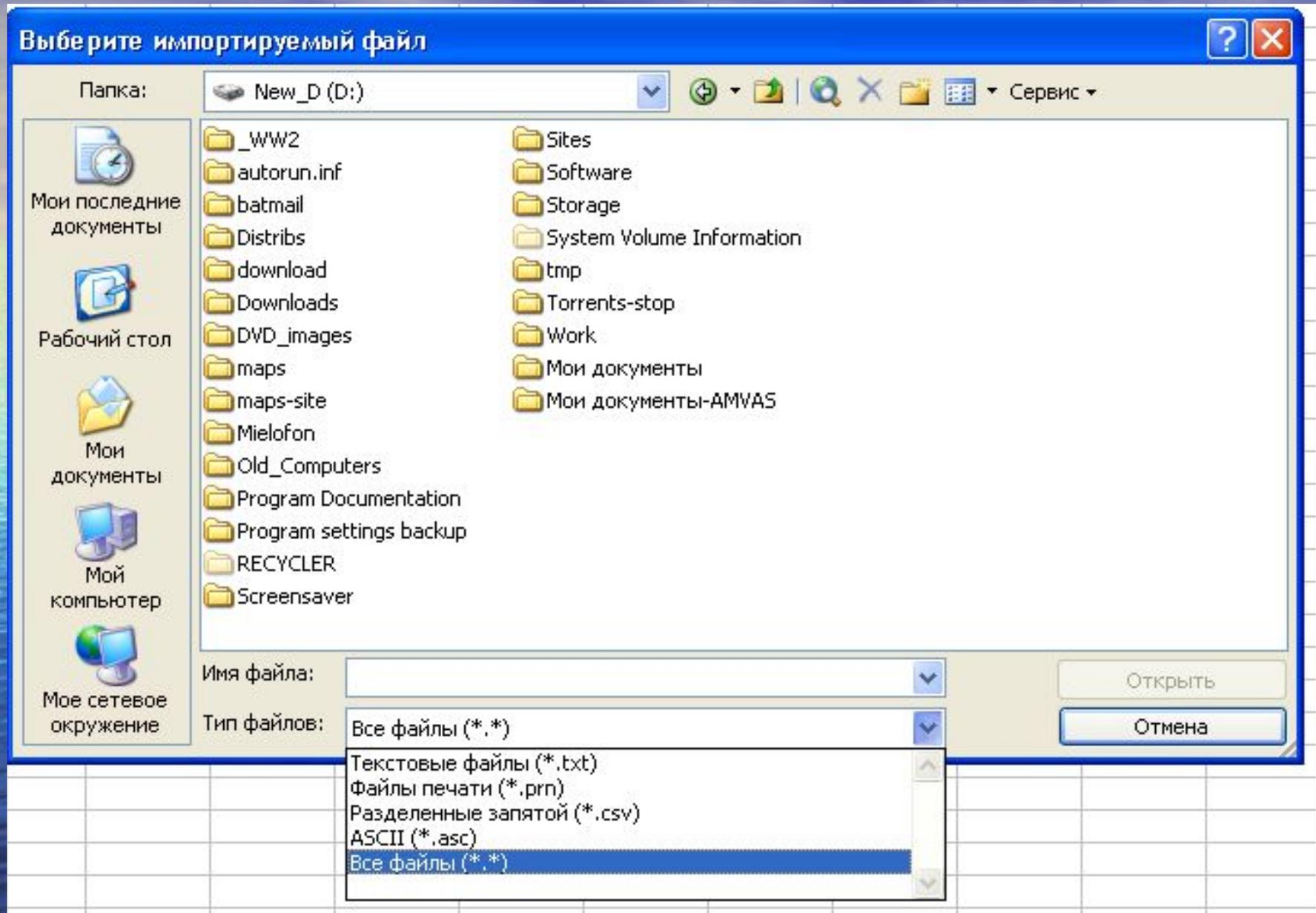
Использование `InputBox` возможно, но не рекомендуется в силу большой вероятности ошибок и неудобства для пользователя.

# Пример кода GetOpenFileName

Option Explicit

```
Sub GetImportFileName()  
Dim Filt As String  
Dim FilterIndex As Integer  
Dim FileName As Variant  
Dim Title As String  
' Настройка списка фильтров  
    Filt = "Текстовые файлы (*.txt), *.txt," & _  
        "файлы печати (*.prn), *.prn," & _  
        "Разделенные запятой (*.csv), *.csv," & _  
        "ASCII (*.asc), *.asc," & _  
        "Все файлы (*.*), *.*"  
' По умолчанию используется фильтр *.*  
    FilterIndex = 5  
' Заголовок окна  
    Title = "Выберите импортируемый файл"  
' Получение имени файла  
    FileName = Application.GetOpenFilename _  
        (FileFilter:=Filt, FilterIndex:=FilterIndex, Title:=Title)  
' При отмене выйти из окна  
    If FileName = False Then  
        MsgBox "файл не выбран"  
        Exit Sub  
    End If  
' Отображение полного имени и пути  
    MsgBox "Вы выбрали " & FileName  
End Sub
```

# Диалоговое окно



# FileDialog

```
Sub ShowFileDialog()  
Dim Fd As FileDialog, i As Integer, btCancel As Integer  
Dim FileNames() As String  
'Задаём тип диалогового окна  
Set Fd = Application.FileDialog(msoFileDialogOpen)  
'Получение имени файла  
With Fd  
    With .Filters  
        'Добавляем фильтры  
        .Clear  
        'Очищаем старые фильтры  
        .Add "Текстовые файлы", "*.txt"  
        .Add "файлы печати", "*.prn"  
        .Add "Все файлы", "*.*"  
    End With  
    .AllowMultiSelect = True  
    'Разрешаем выбор нескольких файлов  
    .Title = "Выберите файлы"  
    'Назначаем заголовок окна  
    .ButtonName = "ВЫБОР"  
    'Надпись на кнопке  
    .FilterIndex = .Filters.Count  
    'По умолчанию выбраны "все файлы"  
    .InitialView = msoFileDialogViewDetails  
    'Задан вид по умолчанию  
    btCancel = .Show  
    'Показываем форму пользователю  
    If btCancel <> 0 Then  
        ReDim FileNames(1 To .SelectedItems.Count) 'Переразмериваем  
        For i = 1 To .SelectedItems.Count  
            'Здесь содержится полные пути к выбранным файлам  
            FileNames(i) = .SelectedItems.Item(i)  
        Next i  
    Else  
        ReDim FileNames(1 To 1)  
        FileNames(1) = ""  
        'Переразмериваем  
        'Пользователь выбрал "Отмена"  
    End If  
    For i = 1 To UBound(FileNames)  
        MsgBox "Выбран файл: " & FileNames(i)  
    Next i  
End With  
End Sub
```

# Выбор папки

Когда требуется выбрать папку, то  
МОЖНО ИСПОЛЬЗОВАТЬ МЕТОД

**BrowseForFolder** объекта **Application**

```
Sub FolderOpen()  
    Dim objShell, objFolder, objFolderItem  
    Dim strPath As String  
  
    On Error GoTo Problem  
    Set objShell = CreateObject("Shell.Application")  
    Set objFolder = objShell.BrowseForFolder(0, "Выберите папку:", &H4000, "")  
    Set objFolderItem = objFolder.Self  
    strPath = objFolderItem.Path  
    MsgBox strPath  
    Exit Sub  
Problem:  
    MsgBox "Выбор отменен"  
End Sub
```

# Выбор папки при помощи FileDialog

```
Sub PickFolderDialog()  
  ' Выбор папки  
  Dim Fd As FileDialog, i As Integer, btCancel As Integer  
  Dim Folder As String  
  
  ' Задаём тип диалогового окна  
  Set Fd = Application.FileDialog(msoFileDialogFolderPicker)  
  ' Получение имени папки  
  With Fd  
    .AllowMultiSelect = False           ' Запрещаем выбор нескольких папок  
    .Title = "Выберите папку"          ' Назначаем заголовок окна  
    .ButtonName = "ВЫБОР"              ' Надпись на кнопке  
    .InitialView = msoFileDialogViewDetails ' Задан вид по умолчанию  
    btCancel = .Show                   ' Показываем форму пользователю  
    Folder = ""  
    If btCancel <> 0 Then Folder = .SelectedItems.Item(1) ' Выбранный путь  
  End With  
  
  MsgBox "Выбрана папка: " & Folder  
End Sub
```

# Процедуры обработки ошибок

При составлении приложений важно предусмотреть, чтобы программа анализировала возможные ошибки, возникающие при ее выполнении по вине пользователя, и информировала. При этом возможно два подхода:

- Предотвращение ошибок: программно анализировать вводимые или вычисляемые данные и в случае, если они могут приводить к ошибке, обеспечить, чтобы программа информировала пользователя о необходимости корректного задания данных.
- Обработка ошибок: в случае появления ошибки, перехватить ее, обработать и программно откликнуться на возникшую ошибку.

При создании приложений надо сочетать оба подхода, применяя в каждом конкретном случае и для каждой возможной ошибки тот, который кажется разработчику наиболее эффективным.

# Функция перехвата ошибок On Error

**On Error** производит перехват ошибки. Устанавливает, что программа должна делать в случае появления ошибки.

Допустимы следующие синтаксисы.

## □ Синтаксис 1: **On Error GoTo строка**

Активизирует подпрограмму обработки ошибок, начало которой определяется обязательным аргументом строка, значением которого может быть либо метка строки, либо номер строки.

## □ Синтаксис 2: **On Error Resume Next**

Указывает, что при возникновении ошибки происходит передача управления на инструкцию, непосредственно следующую за инструкцией, где возникла ошибка.

## □ Синтаксис 3: **On Error GoTo 0**

Отключает любой активизированный обработчик ошибок в текущей процедуре.

# Инструкция **Resume**

Обеспечивает процедуре возможность продолжить работу после обработки ошибки. Допустимы следующие синтаксисы.

## □ Синтаксис 1: **Resume**

После обработки ошибки управление передается той инструкции, в которой произошла ошибка.

## □ Синтаксис 2: **Resume строка**

После обработки ошибки управление передается инструкции, определенной аргументом строка. Значением этого аргумента может быть любая метка строки или номер строки.

## □ Синтаксис 3: **Resume Next**

После обработки ошибки управление передается инструкции, следующей за инструкцией, в которой произошла ошибка.

# Инструкция Exit

Инструкция **Exit** останавливает выполнение процедуры.

Допустимые синтаксисы:

- **Exit Sub**
- **Exit Function**
- **Exit Property**

## Свойства объекта Err

<b>Number</b>	Возвращает код ошибки
<b>Source</b>	Имя текущего проекта VBA
<b>Description</b>	Возвращает строковое выражение, содержащее текст сообщения об ошибке
<b>HelpFile</b>	Полное имя (включая диск и путь) файла справки VBA
<b>HelpContext</b>	Контекстный идентификатор файла справки VBA, соответствующий ошибке с кодом, указанным в свойстве <b>Number</b>
<b>LastDLLError</b>	Содержит системный код ошибки для последнего вызова библиотеки динамической компоновки (DLL)

# Пример перехвата ошибок

```
Option Explicit
```

```
Sub ErrHandle()
```

```
Dim str As String
```

```
Dim R(1 To 2) As Double
```

```
' подпрограмма демонстрирует перехват ошибок
```

```
On Error GoTo ErrHandler
```

```
Print #1, "string"
```

```
Open "file" For Input As #1
```

```
R(5) = 5 / 0
```

```
Err.Raise 68 'генерируется ошибка с кодом 68
```

```
10:
```

```
On Error Resume Next
```

```
R(5) = 5 / 0
```

```
On Error GoTo ErrHandler2
```

```
str = "Всё нормально"
```

```
MsgBox str, vbInformation
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox "Обнаружена ошибка " & Err.Number & _
```

```
Chr(13) & Err.Description, vbCritical
```

```
Resume Next 'управление передаётся следующему оператору
```

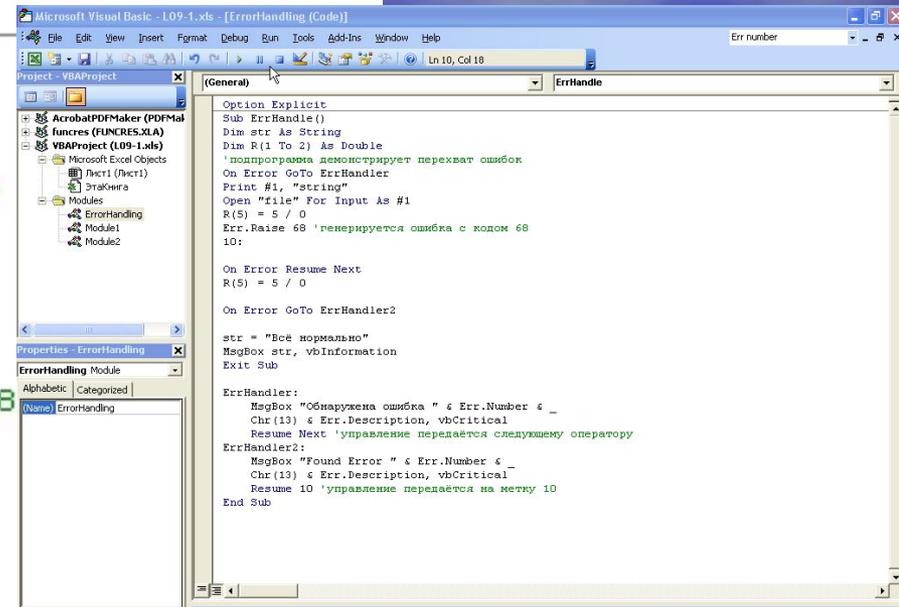
```
ErrorHandler2:
```

```
MsgBox "Found Error " & Err.Number & _
```

```
Chr(13) & Err.Description, vbCritical
```

```
Resume 10 'управление передаётся на метку 10
```

```
End Sub
```



# Коды ошибок

Код	Сообщение
3	Return без вызова процедуры
5	Аргумент превышает допустимое значение
6	Переполнение
7	Не хватает памяти
9	Индекс за пределами диапазона
10	Массив недоступен для переразмеривания.
11	Деление на 0
13	Несоответствие типов
14	Слишком большая строка

# Коды ошибок

Код	Сообщение
16	Слишком сложное выражение
17	Невозможно выполнить операцию
18	Прерывание, вызванное пользователем
20	Оператор <b>Resume</b> без наличия ошибки
28	Недостаточно стека
35	Функция, подпрограмма или свойство не определены
52	Неправильное имя файла или идентификатор.
53	Файл не найден

# Коды ошибок

<b>Код</b>	<b>Сообщение</b>
<b>54</b>	Неверный режим работы с файлом
<b>55</b>	Файл уже открыт
<b>57</b>	Ошибка ввода-вывода
<b>58</b>	Файл уже существует
<b>59</b>	Некорректная длина записи
<b>61</b>	Диск заполнен
<b>62</b>	Попытка чтения из файла, после достижения его конца
<b>63</b>	Некорректный номер записи
<b>67</b>	Слишком много файлов

# Коды ошибок

Код	Сообщение
68	Устройство недоступно
70	Доступ к ресурсу закрыт
71	Диск не готов
74	Попытка перемещения файла на другой диск при помощи оператора <b>Name</b>
75	Ошибка файла/пути
76	Путь не найден
91	Объектная переменная или <b>With</b> блок не заданы
92	Вход внутрь цикла <b>For</b> извне

# Коды ошибок

Код	Сообщение
93	Некорректно задана строка в операторе Like
94	Некорректное использование Null
320	Некорректные символы в имени файла
321	Некорректный формат файла
322	Невозможно создать временный файл
325	Некорректный формат файла ресурсов
335	Невозможен доступ к системным ресурсам

**СПАСИБО ЗА ВНИМАНИЕ!**