

*Федеральное агентство по образованию  
Архангельский государственный технический университет  
Институт информационных технологий*

**А. С. Грошев**

---

# **Программирование на языке Microsoft Visual Basic Scripting Edition**

---

**Методические указания  
к выполнению лабораторных работ**

**Архангельск  
2009**

Рассмотрено и рекомендовано к изданию методической  
комиссией Института информационных технологий  
Архангельского государственного технического университета

УДК 004.65

ББК 32.973.26 - 018.2

Г 89

Грошев А.С. Программирование на языке Visual Basic Scripting Edition:  
Метод. указания к выполнению лабораторных работ. – Архангельск,  
2009. – 82 с.

Предназначена для студентов вузов, изучающих основы программирования на алгоритмических языках.

Содержит лабораторные работы по основным разделам программирования на алгоритмическом языке Visual Basic Scripting Edition.

Особенность данного учебного пособия – наряду с рассмотрением основ построения программ, рассмотрены вопросы использования языка программирования для работы с информацией текстовой, числовой, типа даты и время и с объектами WScript и OLE Automation.

Ил. 36. Табл. 11. Библиогр. 1 назв.

© А.С. Грошев, 2009

## ОГЛАВЛЕНИЕ

|                                                                                                       |    |
|-------------------------------------------------------------------------------------------------------|----|
| Лабораторная работа № 1. Создание простейшей программы. Работа с окнами сообщений и ввода данных..... | 4  |
| Лабораторная работа № 2. Типы данных. Константы. Переменные ...                                       | 13 |
| Лабораторная работа № 3. Массивы .....                                                                | 19 |
| Лабораторная работа № 4. Операторы IF и CASE.....                                                     | 23 |
| Лабораторная работа № 5. Операторы цикла Do и While.....                                              | 29 |
| Лабораторная работа № 6. Операторы цикла For и For Each .....                                         | 34 |
| Лабораторная работа № 7. Процедуры и функции пользователя .....                                       | 37 |
| Лабораторная работа № 8. Работа с числовой информацией.....                                           | 41 |
| Лабораторная работа № 9. Работа со строковой информацией .....                                        | 44 |
| Лабораторная работа № 10. Работа с информацией типа дата и время                                      | 46 |
| Лабораторная работа № 11. Работа с логическими выражениями .....                                      | 51 |
| Лабораторная работа № 12. Работа с объектами WScript.....                                             | 54 |
| Лабораторная работа № 13. Работа с информацией файловой системы .....                                 | 59 |
| Лабораторная работа № 14. Работа с информацией об ошибках.....                                        | 68 |
| Лабораторная работа № 15. Работа с информацией локальной сети ..                                      | 70 |
| Лабораторная работа № 16. Работа с объектами Windows OLE Automation (Microsoft ActiveX).....          | 71 |
| Лабораторная работа № 17. Использование скриптов на HTML-страницах .....                              | 74 |
| Приложение 1. Математические функции языка.....                                                       | 76 |
| Приложение 2. Строковые функции языка .....                                                           | 78 |
| Приложение 3. Функции работы с датой и временем .....                                                 | 79 |
| Приложение 4. Константы даты и времени.....                                                           | 80 |
| Приложение 5. Логические функции и операторы языка.....                                               | 81 |
| Приложение 6. Методы и свойства объекта ADO.Recordset.....                                            | 82 |
| Литература .....                                                                                      | 85 |

## Лабораторная работа № 1. Создание простейшей программы. Работа с окнами сообщений и ввода данных

Программа на языке *VBScript* состоит из инструкций языка (statement) в виде текстовых строк.

Несколько инструкций языка можно объединить в одну строку в текстовом файле программы с использованием разделителя строк – символа двоеточия ( : ) и наоборот, одну строку программы можно написать на нескольких строках в тексте с использованием символа подчеркивания ( \_ ).

В русском языке инструкции языка программирования обычно называют операторами языка, хотя это не совсем точно: операторами в английских первоисточниках называют символы для обозначения математических, логических и строковых операций (=, +, -, /, and, or, eqv, & и пр.). Далее будет использоваться традиционная русская терминология с использованием слова операторы для обозначения инструкций языка.

Текст программы можно написать в любом простейшем редакторе, сохраняющем файлы в кодировке ASCII или Юникод, например, в стандартных программах Windows Блокнот или WordPad.

Существуют также специализированные редакторы, предназначенные для написания в них программ.

Их преимущества:

- 1) выделение различных синтаксических конструкций языка программирования разным цветом;
- 2) наличие справочной системы по операторам, функциям и объектам языка;
- 3) возможность отладки программы путем задания точек останова, пошагового прохождения с просмотром значений переменных;
- 4) возможность запуска интерпретируемых программ непосредственно из редактора.

Всеми этими достоинствами для языка *VBScript* обладает лицензионная программа VbsEdit.

Редактор EmEditor Professional может быть настроен на выделение синтаксиса различных языков: Bat, C#, C++, CSS, HTML, Java, JavaScript, Pascal, Perl, PerlScript, PHP, Python, RHTML, Ruby, SQL, TeX, VBScript, Windows Script, x86 Assembler, XML, имеет возможность запуска скриптовых программ и открытия зарегистрированных приложений для различных типов файлов. Несколько меньшие возможности имеют редакторы Aditor Pro, TextPad и др.

Далее воспользуемся редактором Блокнот, как наиболее доступный.

Для создания простейшей программы делаем следующее:

- 1) запускаем Блокнот;
- 2) пишем в Блокноте строку: **MsgBox "Привет!"** (это имя функции с аргументом – текстовой константой между апострофами; имя **MsgBox** – сокращение от английского выражения Message Box, которое дословно можно перевести, как «коробка сообщений», в системе Windows – окно сообщений;
- 3) сохраняем текстовый файл с именем **Prg1.vbs**;
- 4) в свойствах файла в пункте *Приложение* проверяем, что для работы с ним задана программа **Microsoft Windows Based Script Host** (см. рисунок 1.1). Если этого нет, нажимаем кнопку **Изменить** и выбираем в папке `\Windows\system32\` файл **wscript.exe**;
- 5) двойным щелчком мыши открываем файл.

Результат работы этой программы – диалоговое окно в системе Windows (Windows-форма), показанное на рисунке 1.2.

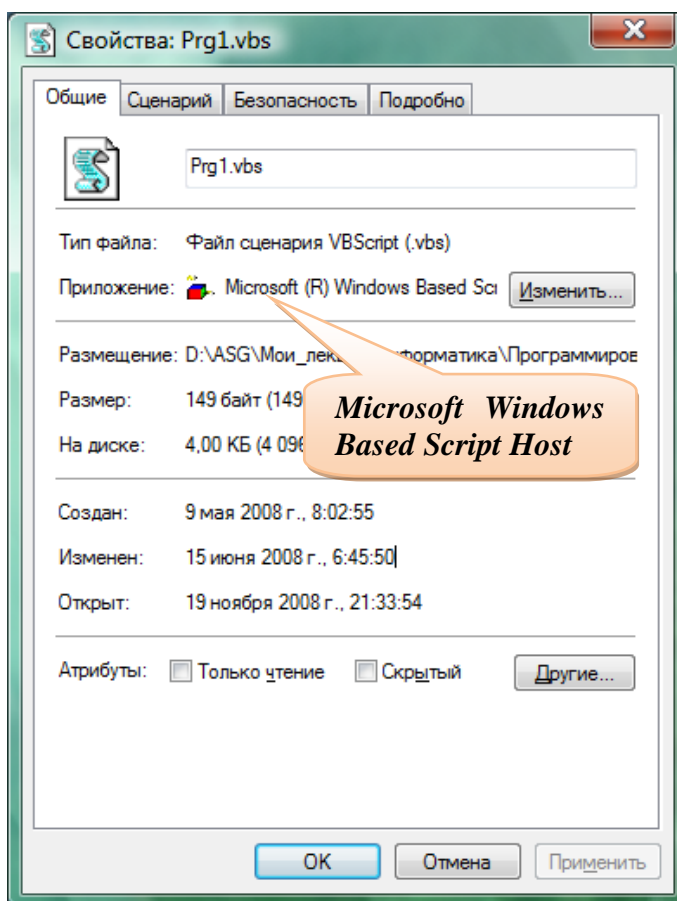


Рисунок 1.1. Окно свойств файла Prg1.vbs

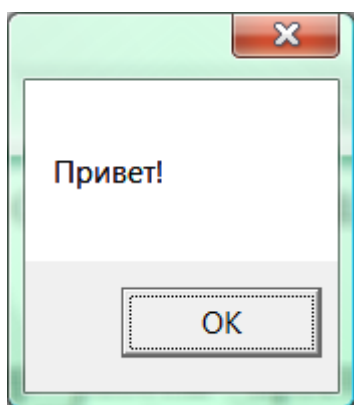


Рисунок 1.2. Пример работы простейшей программы на языке Vbscript, исполняемой системой *Windows Based Script Host* (*wscript.exe*)

При выполнении этой программы используется стандартная функция языка *VBScript* для вывода сообщений в окно Windows со следующим полным синтаксисом (здесь и далее в описании синтаксиса в квадратных скобках [ ] приводятся необязательные элементы, элементы в скобках < > должны быть заменены конкретными значениями):

```
[<р> = ] MsgBox( <Сообщение>[, <Кнопки и значок>] _  
                [, <Заголовок окна>] [, <Справка, раздел>] )
```

где назначение аргументов функции следующее:

**р** – переменная, которой присваивается код нажатой кнопки;

**Сообщение** – текст в диалоговом окне;

**Кнопки и значок** – стандартные переменные (приведены далее в таблице 1.1), определяющие кнопки, значок и номер кнопки по умолчанию в окне (например, *vbYesNoCancel* + *vbInformation* + *vbDefaultButton3* или *3+64+512*);

**Заголовок окна** – надпись на заголовке окна (например, «Мое первое окно»);

**Справка, раздел** – имя файла справки и идентификатор раздела, связанного с данным окном.

Аргументы функции следует писать в круглых скобках, если слева стоит переменная, которой присваивается значение, возвращаемое функцией, иначе аргументы следует писать за именем функции через пробел без скобок.

Дополним текст в файле *Prg1.vbs* указанными дополнительными параметрами (текст пишем в одну строку или используем знак подчеркивания \_ в конце первой строки для продолжения текста функции на следующей строке):

```
MsgBox "Привет!", vbYesNoCancel + vbInformation _  
      + vbDefaultButton3, "Мое первое окно", "tst.hlp", 1
```

Сообщение — Кнопки, значок и номер кнопки по умолчанию — Заголовок окна — Справка, ее раздел

Окно запущенной программы будет иметь вид, показанный на рисунке 1.3.

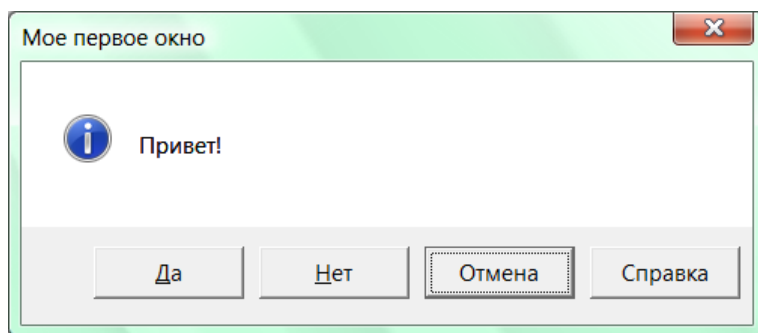


Рисунок 1.3.  
Пример  
использования  
функции MsgBox  
языка Vbscript с  
заданием набора  
кнопок и иконки

Функция **MsgBox** может возвращать значение нажатой в окне кнопки (например **6**, если нажата кнопка **Yes (Да)**), либо другие значения для кнопок **vbNo**, **vbCancel** и пр., см. далее таблицу 1.1).

Таблица 1.1. Константы диалоговых окон

| Константа                 | Значение | Описание                                                                                |
|---------------------------|----------|-----------------------------------------------------------------------------------------|
| <b>vbOKOnly</b>           | 0        | Показана только кнопка <b>ОК</b>                                                        |
| <b>vbOKCancel</b>         | 1        | Показаны кнопки <b>ОК</b> и <b>Отмена</b> (Cancel)                                      |
| <b>vbAbortRetryIgnore</b> | 2        | Показаны кнопки <b>Стоп</b> (Abort), <b>Повтор</b> (Retry) и <b>Пропустить</b> (Ignore) |
| <b>vbYesNoCancel</b>      | 3        | Показаны кнопки <b>Да</b> (Yes), <b>Нет</b> (No) и <b>Отмена</b> (Cancel)               |
| <b>vbYesNo</b>            | 4        | Показаны кнопки <b>Да</b> (Yes) и <b>Нет</b> (No)                                       |
| <b>vbRetryCancel</b>      | 5        | Показаны кнопки <b>Повтор</b> (Retry) и <b>Отмена</b> (Cancel)                          |
| <b>vbCritical</b>         | 16       | Показан значок <b>Stop Mark</b> (знак стоп)                                             |
| <b>vbQuestion</b>         | 32       | Показан значок <b>Question Mark</b> (знак вопроса)                                      |
| <b>vbExclamation</b>      | 48       | Выводится значок <b>Exclamation Mark</b> (восклицательный знак)                         |
| <b>vbInformation</b>      | 64       | Показан значок <b>Information Mark</b> (информационный знак)                            |
| <b>vbDefaultButton1</b>   | 0        | По умолчанию в окне выбрана первая кнопка                                               |
| <b>vbDefaultButton2</b>   | 256      | По умолчанию в окне выбрана вторая кнопка                                               |
| <b>vbDefaultButton3</b>   | 512      | По умолчанию в окне выбрана третья кнопка                                               |
| <b>vbDefaultButton4</b>   | 768      | По умолчанию в окне выбрана четвёртая кнопка                                            |
| <b>vbSystemModal</b>      | 4096     | Диалоговое окно выводится в модальном режиме и располагается сверху                     |



|                 |   |                                          |
|-----------------|---|------------------------------------------|
|                 |   | всех других окон                         |
| <b>vbOK</b>     | 1 | Нажата кнопка <b>ОК</b> .                |
| <b>vbCancel</b> | 2 | Нажата кнопка <b>Отмена</b> (Cancel)     |
| <b>vbAbort</b>  | 3 | Нажата кнопка <b>Смон</b> (Abort)        |
| <b>vbRetry</b>  | 4 | Нажата кнопка <b>Повтор</b> (Retry)      |
| <b>vbIgnore</b> | 5 | Нажата кнопка <b>Пропустить</b> (Ignore) |
| <b>vbYes</b>    | 6 | Нажата кнопка <b>Да</b> (Yes)            |
| <b>vbNo</b>     | 7 | Нажата кнопка <b>Нет</b> (No)            |

Для определения кода нажатой в окне **MsgBox** кнопки следует использовать следующий синтаксис функции: слева нужно написать переменную, которой будет присвоено возвращаемое функцией значение, далее следует написать символ присваивания (=) и справа от него функцию, у которой аргументы написаны в круглых скобках:

```
btn = MsgBox("Привет!", vbYesNoCancel + vbInformation _  
          + vbDefaultButton3, "Мое первое окно")
```

Диалоговое окно будет иметь тот же вид, что и раньше (только без кнопки **Справка**, см. рисунок 1.3), но после нажатия кнопки в окне переменная **btn** будет иметь значение, соответствующее нажатой кнопке.

Если Вы желаете написать в окне **Сообщение** и **Заголовок окна**, пропустив второй аргумент (**Кнопки и значок**), после первого аргумента следует поставить 2 (ДВЕ!) запятые:

```
MsgBox "Сегодня я написал свою первую программу на VBS!" _  
      , , "Окно сообщений студента Вани Иванова"
```

↑ -- [ Здесь 2 запятые, т. к. пропущен аргумент <Кнопки и значок> ]

Еще одна функция языка, позволяющая открывать окно для ввода пользователем с клавиатуры строки текста (максимальная длина строки 256 символов):

```
[<p> = ] InputBox( <Сообщение>[,<Заголовок окна>] _  
                  [,<Стр.умолч.>][,X][,Y][, <Справка, раздел>])
```

где новые параметры функции (по сравнению с **MsgBox**):

**Стр.умолч.** – строковое значение в поле ввода, которое будет показано по умолчанию при открытии окна;

**X, Y** – координаты левого верхнего угла окна в единицах **twips** (1440 twips = 1 дюйм, 567 twips = 1 см) по отношению к левому верхнему углу экрана; если координаты не указаны, окно выводится примерно в центре экрана.



Дополним программу вводом данных с клавиатуры в окне **InputBox** :

Здесь 2 запятые, т. к. пропущена  
<Стр.умолч.>

```
S = InputBox(vbLF & "Напишите строку текста:", _  
            "Окно ввода. Студент Ваня Иванов", 4000, 2000)  
Kod = MsgBox (S, vbYesNoCancel+vbInformation, _  
            "Окно сообщений")
```

Функция **InputBox** в данном примере имеет строку <Сообщение>, состоящую из двух частей: константы языка VBScript – **vbLF** – код перехода на следующую строку, и текстового значения "Напишите строку текста:"; эти две части объединены в одну строку с использованием оператора конкатенации &.

Сохраним текст, как новый файл **Prg2.vbs**, откроем его двойным щелчком мышкой и увидим окно функции **InputBox** (рисунок 1.4). В поле ввода этого окна напомним новый текст (длиной более 256 символов) и нажмем кнопку **Да**, после чего появится окно, показанное на рисунке 1.5, в сообщении которого присутствует 256 символов значения переменной **S** (если в первом окне нажать кнопку **Нет**, сообщение в окне **MsgBox** будет отсутствовать).

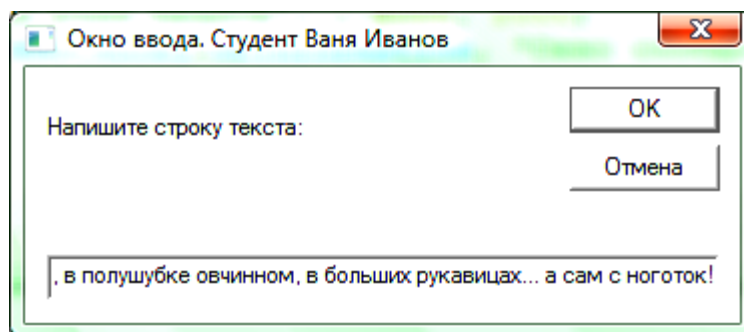


Рисунок 1.4.  
Пример использования функции **InputBox**

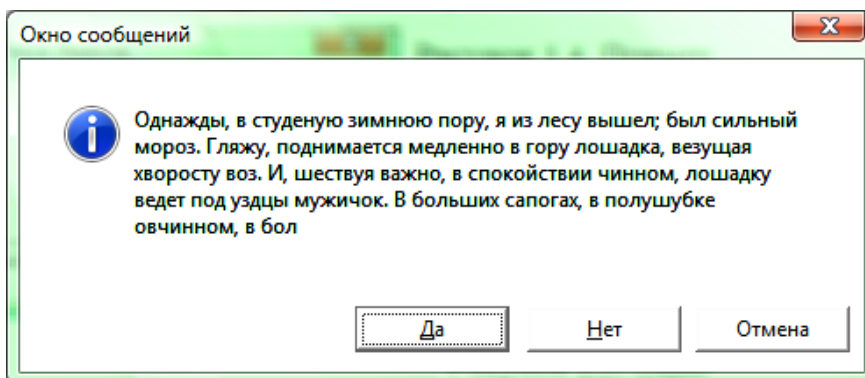


Рисунок 1.5.  
Окно функции **MsgBox** с показом текста, написанного в поле окна **InputBox**

Еще одно стандартное окно с более сложным синтаксисом – с использованием метода **Popup** объекта **WScript.Shell** – отличается от **MsgBox** тем, что для него можно задать время существования, после чего оно закроется с возвращаемым кодом -1, если пользователь не на-

жал какую-либо его кнопку (результат выполнения программы показан на рисунок 1.6):

```
Set W = CreateObject("WScript.Shell")
```

```
W.Popup "Окно закроется через 7 сек. или раньше, "  
        & vbLF & "если Вы нажмете кнопку в окне", 7,  
        "Окно Popup библиотеки WScript.Shell", vbExclamation
```

Полный синтаксис метода *Popup* объекта *WScript.Shell*:

```
[<p>=<Экземпляр объекта WScript.Shell>.Popup  
    (<Сообщение> [, <Секунды ожидания>] [  
    [, <Заголовок окна>] [, <Кнопки и значок>)]
```

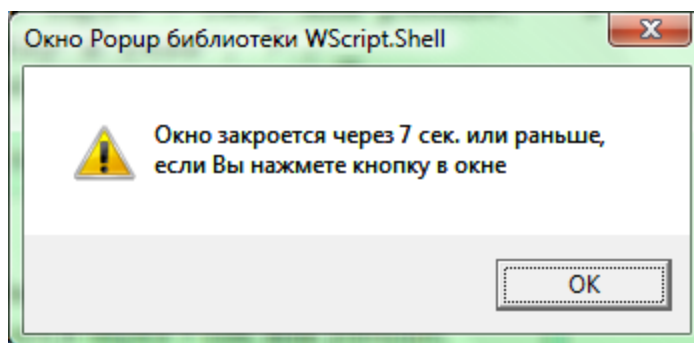


Рисунок 1.6. Использование метода *Popup* объекта *WScript.Shell*

В изложенном выше материале использовались такие основополагающие понятия языка программирования, как строковые значения (символы между апострофами), стандартные константы окон и строковая константа **vbLF** – код перехода на новую строку, переменные (**btn**, **S**, **Kod**), операции присваивания (=) и конкатенации (& – объединение двух выражений любого типа в одну строку), функции с параметрами (**MsgBox**, **InputBox**, **Now**, **CreateObject**), объекты (**WScript.Shell**) и их методы.

Последующие лабораторные работы посвящены освоению использования всех этих компонентов языка VBScript.

### Задания

Написать программу для варианта задания, соответствующего номеру Вашего компьютера. Выполнить программу, сохранить ее текст и скриншоты окон в отчете. Все окна должны иметь заголовки следующего вида: «Окно ввода (или сообщений) <Фамилия имя отчество студента>».

- 1) Вывести в левый верхний угол экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с одной кнопкой **OK** и значком Information Mark. Затем в окне *Popup* показать код нажатой кнопки при выходе из предыдущего окна.

- 2) Вывести на расстоянии 10 см по горизонтали и вертикали от левого верхнего угла экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками  и  и значком Exclamation Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 3) Вывести в центре экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками  и  и значком Stop Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 4) Вывести примерно в правом нижнем углу экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками ,  и  и значком Question Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 5) Вывести в центре экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками  и  и значком Exclamation Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 6) Вывести примерно в правом нижнем углу экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками ,  и  и значком Question Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 7) Вывести на расстоянии 20 см по горизонтали и 15 см по вертикали от левого верхнего угла экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками ,  и  и значком Information Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 8) Вывести примерно в правом верхнем углу экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками ,  и  и значком Exclamation Mark. Затем в окне **Rorip** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.
- 9) Вывести слева примерно в центре по вертикали экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками ,  и  и значком Question Mark. За-

тем в окне ***Рорир*** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.

- 10) Вывести справа примерно в центре по вертикали экрана окно для ввода текстовой строки, показать эту строку в окне сообщений с кнопками  и  и значком Information Mark. Затем в окне ***Рорир*** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.

## Лабораторная работа № 2. Типы данных. Константы. Переменные

Каждый язык программирования предназначен для обработки информации (данных) различных типов. Используемые типы данных и методы их обозначения и обработки могут несколько различаться в различных языках.

Типы данных определяют:

- формат представления данных в памяти компьютера;
- область или диапазон возможных значений;
- множество допустимых операций, применимых к данным.

В языке *Microsoft Visual Basic Scripting Edition* определен единственный тип данных – **Variant**. Это специальный тип, который может содержать в себе различные виды информации. Все функции языка также возвращают данные типа **Variant**.

Различные категории информации, которая может содержаться в типе **Variant**, называются подтипами.

В таблице 2.1 приведены подтипы данных, которые могут содержаться в типе **Variant**.

Таблица 2.1. Подтипы языка VBScript

| Подтип               | Описание                                                                                                                                                                                                       |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Byte</b>          | Целые числа в диапазоне от 0 до 255.                                                                                                                                                                           |
| <b>Boolean</b>       | Логические значения <i>True</i> или <i>False</i> .                                                                                                                                                             |
| <b>Integer</b>       | Целые числа в диапазоне от -32768 до 32767.                                                                                                                                                                    |
| <b>Long</b>          | Целые числа в диапазоне от -2 147 483 648 до 2 147 483 647.                                                                                                                                                    |
| <b>Single</b>        | Числа одинарной точности с плавающей точкой в диапазоне от -3.402823E38 до -1.401298E-45 для отрицательных значений; от 1.401298E-45 до 3.402823E38 для положительных значений.                                |
| <b>Double</b>        | Числа двойной точности с плавающей точкой в диапазоне от -1.79769313486232E308 до -4.94065645841247E-324 для отрицательных значений; 4.94065645841247E-324 до 1.79769313486232E308 для положительных значений. |
| <b>Currency</b>      | -922 337 203 685 477.5808 до 922 337 203 685 477.5807.                                                                                                                                                         |
| <b>Date / (Time)</b> | Числа, которые представляют даты и время в диапазоне между 1-01-100 0:0:0 до 31-12-9999 23:59:59.                                                                                                              |
| <b>Object</b>        | Содержит объект                                                                                                                                                                                                |
| <b>String</b>        | Строка переменной длины, которая максимально может содержать 2 миллиона символов.                                                                                                                              |
| <b>Empty</b>         | Неинициализированное значение (0 для числовых переменных, строка нулевой длины ("") для строковых переменных).                                                                                                 |
| <b>Null</b>          | Содержит неверные для подтипа данные.                                                                                                                                                                          |
| <b>Error</b>         | Содержит номер ошибки.                                                                                                                                                                                         |

Функция **VarType** возвращает информацию о том, как данные сохранены в типе **Variant**. Для преобразования одного подтипа в другой могут использоваться соответствующие функции (**Cbyte**, **Cdate**, **CSng**, **Cdbl** и др.).

Другие диалекты языка Visual Basic также имеют тип **Variant**, но наряду с ним могут определять переменные различных типов, таких же, как подтипы языка VBScript.

В тексте программы могут использоваться числа, строки текста, даты и время, которые являются константами.

**Константа** – некоторое неизменяемое значение в тексте программы. Константа может иметь имя (*идентификатор*).

Для тех констант, которые используются часто, можно задать имена. Задание имен константам делает программы легко читаемыми. Для этого в любом месте текста программы можно использовать следующее описание:

```
Const N = 1.15e-15
Const FIO = "Иванов Иван Иванович"
Const Data_r = #05-13-1988 06:30:00#
Const Time_r = #06:30:00#
Const Gorod = "Архангельск"
```

Как видно из примера, для числовых констант разделителем целой и дробной части является точка, можно использовать экспоненциальный вид чисел ( $1.15e-15 = 1.15 \times 10^{-15}$ ). Значения строковых констант следует писать между двумя кавычками ("), даты и времени – между двумя знаками решетки (#).

В языке **VBScript** существует достаточно большое количество предопределенных констант, которые сгруппированы по категориям:

1. **Date and Time Constants** – определяют константы для дат и времени для функций работы с ними (см. Приложение 4);
2. **Date Format Constants** – определяют форматы дат и времени (см. Приложение 4);
3. **MsgBox Constants** – используются в функции **MsgBox** и других диалоговых окнах (см. таблицу 1.1);
4. **String Constants** – определяют скрытые символы, используемые для манипуляции со строками:
  - **vbCr** – возврат каретки (Chr(13), переход в начало следующей строки);
  - **vbLf** – новая строка (Chr(10));
  - **vbCrLf** – новая строка (Chr(13) + Chr(10));
  - **vbNewLine** – новая строка (Chr(10) или Chr(13) + Chr(10));
  - **vbNullChar** – символ с нулевым значением (Chr(0));

- *vbNullString* – строка с нулевым значением (Chr(0));
  - *vbTab* – горизонтальная табуляция (Chr(9)) ;
  - *vbVerticalTab* – вертикальная табуляция (Chr(11)) ;
5. **VarType Constants** – определяют форматы для различных подтипов (*vbEmpty, vbNull, vbInteger, vbLong, vbSingle, vbSingle, vbCurrency, vbDate, vbString, vbObject, vbError, vbBoolean, vbVariant, vbDataObject, vbDecimal, vbByte, vbArray*);
  6. **File Attribute Constants** – атрибуты файлов (*Normal, ReadOnly, Hidden, System, Volume, Directory, Archive, Alias, Compressed*);
  7. **DriveType Constants** – типы дисковых устройств (*Unknown, Removable, Fixed, Remote, CDRom, RAMDisk*);
  8. **File Input/Output Constants** – типы ввода-вывода для файлов (*ForReading, ForWriting, ForAppending*);
  9. **Color Constants** – определяют 8 основных цветов (могут использоваться в HTML-скриптах):
    - *vbBlack, vbRed, vbGreen, vbYellow, vbBlue, vbMagenta, vbCyan, vbWhite*;
  10. **Comparison Constants** – определяют тип операции сравнения:
    - *vbBinaryCompare* – двоичное сравнение;
    - *vbTextCompare* – текстовое сравнение.

**Переменная** – имя (*идентификатор*) в программе, связанное с областью оперативной памяти компьютера, предназначенной для хранения какой-либо информации, которая может изменяться во время работы программы.

Все переменные в языке *VBScript* имеют один тип – **Variant** и во время использования могут хранить данные разных подтипов.

**Правила написания идентификаторов переменных, констант, названий процедур, функций, объектов, их методов и свойств следующие:**

- 1) идентификатор должен начинаться с латинской буквы;
- 2) может состоять из латинских строчных и прописных букв, цифр и символа подчеркивания (последний лучше не использовать, могут быть проблемы, т. к. этот же символ используется, как признак переноса строки);
- 3) длина его – не более 255 символов;
- 4) буквы в верхнем и нижнем регистре *не* различаются;
- 5) он должен быть уникален в области определения.



Для объявления переменных могут служить выражения:

```
Dim X, Y, Z  
Public A, B, C  
Private X1, X2, X3
```

Однако, переменные в языке Basic можно и не объявлять с помощью этих описаний, достаточно написать в программе новый идентификатор и присвоить ему значение, после чего транслятор будет знать, что это переменная, например:

```
Z = 1.2345  
S = "Строка текста"  
DT = #12-31-08#
```

Разница между описаниями **Dim**, **Public**, **Private** существенна при использовании внутри программы подпрограмм и функций пользователя. В этом случае различается область действия переменных и массивов.

Для описания **Dim** область действия различается в зависимости от места его расположения:

- 1) на уровне программы – переменные доступны в основной программе и во всех её подпрограммах (глобальные переменные);
- 2) описание в подпрограмме (процедуре или функции) – переменные используются только в подпрограмме, где они описаны, инициализируются и используются при исполнении этой подпрограммы, после завершения ее работы уничтожаются (локальные переменные).

Описания **Public** и **Private** используются только на уровне программы, переменные в этом случае доступны в основной программе и во всех её подпрограммах.

При отсутствии явного описания все переменные, используемые в головной программе, являются глобальными, отсутствующие в головной программе, но используемые в подпрограмме – локальные.

Если в начале программы написать строку:

```
Option Explicit
```

использование переменных без их явного описания в выражениях **Dim**, **Public** и **Private** будет запрещено. Попытка использовать необъявленную переменную вызовет сообщение об ошибке при выполнении программы.

Явное описание переменных способствует уменьшению количества ошибок при программировании, в частности, предотвращает конфликты между переменными, используемыми в основной программе и в процедурах при одинаковых именах; позволяет выявить неверно написанные имена предопределенных констант языка (иначе неверные имена просто игнорируются).

## Задания

В вариантах заданий в скобках < > задано значение переменных, которые нужно получить, в скобках ( ) – номера констант Вашего задания. Значения переменных следует формировать из констант, пробелов и знаков препинания с использованием оператора конкатенации &.

1. Задайте в программе 3 константы и присвойте им значения 1) Вашего имени, 2) отчества, 3) фамилии. Создайте 2 переменные и присвойте им значения: первой – <(3), (1), (2)>, второй – <(1) (2) (3)>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
2. Задайте в программе 3 константы и присвойте им значения Ваших 1) дня, 2) название месяца, 3) года рождения. Создайте 2 переменные и присвойте им значения: первой – <(1).(2).(3)>, второй – <Я родился (1) (2) (3) года .>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
3. Задайте в программе 4 константы и присвойте им значения данных Вашего адреса проживания: 1) город; 2) улица 3) номер дома; 4) номер квартиры. Создайте 2 переменные и присвойте им значения: первой – <(1), (2), (3), (4)>, второй – <Я живу в городе (1) на улице (2) в доме (3), квартира (4).>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
4. Задайте в программе 4 константы и присвойте им значения данных о Вашей учебе: 1) учебное заведение; 2) специальность 3) курс; 4) группа. Создайте 2 переменные и присвойте им значения: первой – <(1),(2), (3), (4)>, второй – <Я учусь в (1) на специальности (2) курс (3), группа (4).>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
5. Задайте в программе 3 константы и присвойте им значения данных о Вашей учебе в школе: 1) населенный пункт; 2) № школы; 3) любимый предмет. Создайте 2 переменные и присвойте им значения: первой – <(1), (2), (3)>, второй – <Мой любимый предмет был (3), когда я учился в (2)-й школе города (или название другого типа населенного пункта) (1).>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
6. Задайте в программе 4 константы и присвойте им значения паспортных данных (придумать близкие к возможным): 1) серия; 2) №; 3) кем выдан; 4) дата выдачи. Создайте 2 переменные и присвойте им значения: первой – <(1) – (2) , (3) (4)>, второй – <Паспортные данные: серия (1), номер (2), выдан (4) (3).>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).

7. Задайте в программе 3 константы и присвойте им значения Ваших антропометрических данных: 1) рост в см; 2) вес в кг; 3) окружность груди; 4) талии; 5) бедер, в см. Создайте 2 переменные и присвойте им значения: первой – <(1), (2), (3) – (4) – (5)>, второй – <Мой рост (1) см, вес (2), окружность груди, талии и бедер (3) – (4) – (5) см >. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
8. Задайте в программе 3 константы и присвойте им названия предметов Вашего сегодняшнего расписания: 1) 1-я пара; 2) 2-я пара 3) 3-я пара. Создайте 2 переменные и присвойте им значения: первой – <(1) – (2) – (3)>, второй – <8.20-9.05 9.10-9.55 – (1); 10.10-10.55 11.00-11.45 – (2); 12.00-12.45 12.50-13.35 – (3)>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
9. Задайте в программе 3 константы и присвойте им названия окон VBScript: 1) **MsgBox**; 2) **InputBox** 3) **Popup**. Создайте 2 переменные и присвойте им значения: первой – <Окна VBScript: (1), (2), (3)>, второй – <Их назначение и особенности: (1) – (здесь написать назначение), (2) – (здесь об этом окне), (3) – (здесь особенности этого окна)>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).
10. Задайте в программе 6 констант и присвойте им названия подтипов данных *VBScript* для работы с числами. Создайте 6 переменных и присвойте им значения: <(подтип) – диапазон данных (здесь указать диапазон)>. Покажите константы и переменные в модальном окне MsgBox (каждую в отдельной строке).

## Лабораторная работа № 3. Массивы

Переменная, которая хранит единственное значение, называется скалярной. Переменная может использоваться и для хранения серии чисел, такая переменная является массивом.

**Массив** – переменная, предназначенная для хранения пронумерованной серии значений (элементов массива).

Скалярная переменная может использоваться без явного описания с использованием ключевых слов **Dim**, **Public** или **Private**, если в начале программы не присутствует директива **Option Explicit**.

**Массив обязательно должен быть описан перед его использованием.**

Для описания массивов используются те же операторы, что и для переменных, только после имени переменной в круглых скобках указывается количество индексов и их максимальное значение у элементов массива.

Полный синтаксис этих описаний следующий:

```
Dim varname [(subscripts)] [, varname [(subscripts))] ...  
Public varname [(subscripts)] [, varname [(subscripts))] ...  
Private varname [(subscripts)] [, varname [(subscripts))] ...
```

где:

**Varname** – имя переменной;

**Subscripts** – имеет формат: **индекс1** [, **индекс2**] ... – максимальные значения индексов, минимальное значение равно 0; массив может быть одномерный, двумерный и т. д. до 60.

Пример:

```
Dim X(99) , Y(24,24) , Z(99,99,99)
```

где:

**X(99)** – одномерный массив из 99 элементов;

**Y(24,24)** – двумерный массив размерностью 24×24 элемента;

**Z(99,99,99)** – трехмерный массив размерностью 99×99×99.

Массив в языке *VBScript* после его объявления имеет тип **Variant**, поэтому его элементам можно присваивать значения различных типов. Максимальный размер массива ограничен размером свободной виртуальной памяти операционной системы.

Массив может быть динамический (изменяемой размерности), при его описании в круглых скобках размерность не указывают:

```
Dim varname()
```

Для инициализации динамического массива следует использовать оператор:

```
ReDim [Preserve] varname (subscripts) [, varname (subscripts)] . . .
```

Параметр **Preserve** используется, если выполняется повторная инициализация для изменения размера массива и необходимо сохранить значения переменных, которые им уже были присвоены.

Пример:

```
Dim X()  
ReDim X(10, 10, 10)  
. . .  
ReDim Preserve X(10, 10, 15)
```

Разница между описаниями **Dim**, **Public**, **Private** существенна при использовании внутри программы подпрограмм и функций пользователя. В этом случае различается область действия переменных и массивов.

Для описания **Dim** область действия различается в зависимости от места его расположения:

- 1) на уровне программы – переменные доступны в основной программе и во всех её подпрограммах и функциях;
- 2) описание в подпрограмме или функции – переменные используются только в подпрограмме, где они описаны, инициализируются и используются при исполнении этого модуля, после завершения его работы уничтожаются.

Описания **Public** и **Private** используются только на уровне основной программы, переменные в этом случае доступны в этой программе и во всех её подпрограммах и функциях.

Другой способ создания переменной типа **Variant**, содержащей одномерный массив, – с помощью функции **Array**:

```
A = Array(10,20,30,40)
```

В этом случае значения элементов массива будут следующими:

$A(0)=10$ ,  $A(1)=20$ ,  $A(2)=30$  и  $A(3)=40$ .

Пример программы с использованием различных описаний переменных и массивов:

```
Option Explicit  
Dim Y(99), Z(99,99,99)  
Y(0) = "Это 1-ый элемент массива Y - строка"  
Y(1) = 123.456 ' Тип второго элемента - число  
Y(99)= #12-30-2007# ' Тип 100-го элемента - дата  
Z(99,99,99)="Это элемент трехмерного массива Z "  
            & "с индексом 99,99,99"  
MsgBox Y(0) & vbCrLf & "2-ой элемент массива Y - число: " &  
            & Y(1) & vbCrLf & "100-ый элемент массива Y - дата: " &
```

```

        & Y(99) & vbLf & Z(99,99,99) , , "Окно 1"
Dim X                'Глобальная переменная X
X="Это переменная X основной программы"
MsgBox X, , "Окно 2"
Call my_proc
Sub my_proc
    Dim X            'Локальная переменная X
    X="Это переменная X подпрограммы"
    MsgBox X, , "Окно 3"
End Sub
MsgBox X, , "Окно 4 – снова глобальная X"

```

Окна, которые показывает эта программа, показаны на рисунке 3.1.

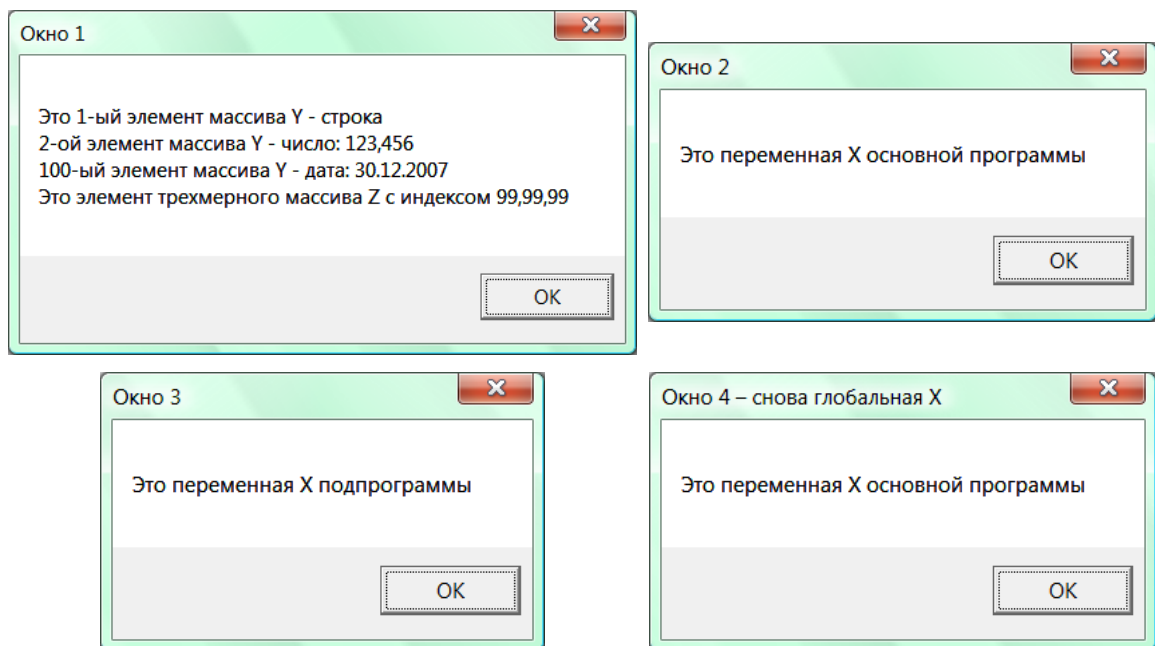


Рисунок 3.1. Использование различных описаний переменных и массивов

### Задания

- 1) Опишите в программе два одномерных массива размерностью 3 и 5 элементов, присвойте значения элементам первого массива – нечетные числа, начиная с 21, второго – буквы русского алфавита, начиная с мягкого знака. Покажите все данные в окне сообщений.
- 2) Опишите в программе динамический массив. Выполните вначале его инициализацию для размерности 3 элемента и присвойте значения элементам массива – любые числа. Покажите все данные в 1-м окне сообщений. Затем выполните повторную инициализацию для размерности 7 с сохранением значений определенных ранее элементов. Присвойте элементам с 4 по 7-й любые даты. Покажите все данные во 2-м окне сообщений.

- 3) Опишите в программе двумерный массива размерностью  $2 \times 3$  элементов и присвойте значения каждому элементу массива – время в диапазоне от 7:00 до 19:00. Покажите данные в окне сообщений в виде матрицы, в которой номер строки – первый индекс, в строке изменяется второй индекс.
- 4) Задайте с помощью функции *Array* значения 5-ти элементам массива, представляющим собой геометрическую прогрессию. Покажите все данные в окне сообщений.
- 5) Создайте с помощью функции *Array* одномерный массив, состоящий из 6-ти чисел,. Покажите данные в окне сообщений. С помощью функции *ReDim* переопределите его размерность до двумерного размерностью  $3 \times 2$ . Задайте значения всем его элементам и покажите их в окне сообщений в виде матрицы, в которой номер строки – первый индекс, в строке изменяется второй индекс.
- 6) Опишите в программе трехмерный массива размерностью  $2 \times 3 \times 4$  элементов и присвойте числовые значения элементам массива. Покажите данные в окне сообщений с указанием элемента массива и его значение (например,  $A(0,0,0)=1$  и т. д.).
- 7) Задайте элементам двумерного массива текстовые значения – каждому одно слово какого-либо четверостишия. Покажите элементы массива в окне сообщений в виде стихотворения.
- 8) Опишите в программе два одномерных массива X и Y размерностью 5 элементов, присвойте числовые значения элементам массивов. Покажите данные в окне сообщений в виде таблицы, в первой строке которой показаны имена массивов, в последующих – значения их элементов.
- 9) Опишите в программе одномерный массив из 7 элементов. Присвойте значения элементам – целые числа. Покажите элементы массива в окне сообщений в следующем порядке: 1, 7, 2, 6, 3, 5, 4.
- 10) Опишите в программе два одномерных массива размерностью 5 элементов, присвойте числовые значения элементам массивов. Покажите данные в окне сообщений: в первой строке элементы первого массива от первого до 5-го, во второй строке – элементы второго массива от 5-го до первого.



## Лабораторная работа № 4. Операторы IF и CASE

**Оператор условного перехода IF** позволяет выполнить те или иные строки программы в зависимости от логических условий.

В языке *VBScript* он может использоваться в двух различных видах (строчный и блочный синтаксисы).

Строчный синтаксис:

```
If <условие> Then <операторы1> [Else <операторы2>]
```

где:

**условие** – логическое выражение, результатом вычисления которого может быть истина (**True**), ложь (**False**) или **Null** которое приравнивается к **False**;

**операторы1** – один оператор или более (разделенных двоеточиями для строкового синтаксиса); выполняются, если условие истинно (**True**);

**операторы2** – выполняются, если **условие** не является истиной (**False**).

В логических выражениях могут использоваться следующие основные операторы сравнения и логические операции (описание использования последних см. в *Приложении 5*):

|     |                      |     |                                                                                                                 |
|-----|----------------------|-----|-----------------------------------------------------------------------------------------------------------------|
| =   | Равно                | And | Логическое «И»                                                                                                  |
| <>  | Не равно             | Or  | Логическое «ИЛИ»                                                                                                |
| <   | Меньше               | Xor | Логическое исключение<br>(E1 Xor E2 возвращает True, если только E1 = True или только E2 = True, иначе – False) |
| >   | Больше               |     |                                                                                                                 |
| <=  | Меньше или равно     | Eqv | Логическое «эквивалентно»                                                                                       |
| >=  | Больше или равно     |     |                                                                                                                 |
| Is  | Сравнение объектов   | Imp | Логическая импликация<br>(E1 Imp E2 возвращает False, если E1 = True и E2 = False, иначе – True)                |
| Not | Логическое отрицание |     |                                                                                                                 |

Для простых условных операторов следует использовать строчный синтаксис.

Пример строчного синтаксиса:

```
If A <= 9 Then A = A + 1 : B = B + A Else B = B + A
```

Блочный синтаксис является более структурированным, имеет большие возможности, легче читается и отлаживается. В одном операторе может быть выполнена проверка нескольких условий с заданием различных исполняемых фрагментов программы.

Блочный синтаксис оператора условного перехода:

```
If <условие> Then  
    [операторы]  
    [ElseIf <условие-п> Then  
        [операторы-п]] ...  
    [Else  
        [else-операторы]]  
End If
```

где:

**условие** – логическое выражение, результатом вычисления которого может быть истина (**True**), ложь (**False**) или **Null** которое приравнивается к **False**;

**операторы** – один оператор или более (разделенных двоеточиями для строкового синтаксиса), которые выполняются, если условие истинно (**True**);

**условие-п** – то же, что и **условие**;

**операторы-п** – выполняются, если **условие-п** является истинной (**True**);

**else-операторы** – один оператор или более, выполняющиеся, если предшествующие условия не были истинны.

Когда выполняется блочный **If**, проверяется **условие**, и, если оно истинно (**True**), выполняются **операторы**, следующие за **Then**. Если **условие** не является истинным (**False**), каждое **условие-п**, идущее за **ElseIf** (если они есть) проверяется. Когда истинное значение найдено, выполняются **операторы-п**, следующие за **Then** после истинного условия, после чего программа выходит за **End If** (т. е. последующие **ElseIf**, если они есть, не проверяются). Если истинных условий для **ElseIf** не найдено, выполняются **else-операторы**, следующие за **Else**.

Пример блочного синтаксиса:

```
FIO="Лютикова Лилия Максимовна"  
a=InputBox("Задайте значение переменной a", "Пример If.  
"&FIO)  
a=Eval(a)      'преобразование строки в число  
If a > 10 Then  
    b = "a > 10"  
    ElseIf a > 0 Then  
        b = "a > 0"    'будет выполнено только это при a=1!  
    ElseIf a = 1 Then  
        b = "a = 1"  
    Else  
        b = "Нет данных для заданного значения a"  
End If
```

**MsgBox "Результат выполнения IF для a = "& a & \_  
": " & b,,FIO**

Пример использования оператора условного перехода:  
расчет суммы страховых взносов в Пенсионный фонд РФ, где  
суммы перечислений (**Sp** и **Np**) зависят от налоговой базы (**S**) и года  
рождения (**G**):

| S                   | G < 1967                         |    | G ≥ 1967                         |                                  |
|---------------------|----------------------------------|----|----------------------------------|----------------------------------|
|                     | Sp                               | Np | Sp                               | Np                               |
| S ≤ 280000          | $10,3 \cdot S / 100$             | 0  | $4,3 \cdot S / 100$              | $6,0 \cdot S / 100$              |
| 280000 < S ≤ 600000 | $28840 + 5,5 (S - 280000) / 100$ | 0  | $12040 + 3,1 (S - 280000) / 100$ | $16800 + 2,4 (S - 280000) / 100$ |
| S > 600000          | 46440                            | 0  | 21960                            | 24480                            |

```

Dim Sm, G, Sp, Np
S=InputBox("Укажите налоговую базу", _
           "Расчет взносов в ПФ")
s=Eval(s) 'преобразуем строковое значение в число
G=InputBox("Укажите год рождения", _
           "Расчет взносов в ПФ")
Np = 0 : Sp = 0
If G < 1967 Then 'расчет для G < 1967
    If S < 280000 Then
        Sp = 10.3*S/100
    ElseIf S > 280000 and S <= 600000 Then
        Sp = 28840+5.5*(Sm-280000)/100
    Else
        Sp = 46440
    End If
Else 'далее расчет для G ≥ 1967
    If S < 280000 Then
        Sp = 4.3*S/100 : Np = 6.0*S/100
    ElseIf S > 280000 and S <= 600000 Then
        Sp = 12040+3.1*(S-280000)/100
        Np = 16800+2.4*(S-280000)/100
    Else
        Sp = 21960 : Np = 24480
    End If
End If
MsgBox "Np =" & Sp & " Sp =" & Np

```

**Оператор выбора Case** позволяет выполнить те или иные операторы в зависимости от множества значений заданного выражения или переменной.

Синтаксис оператора выбора:

```
Select Case <тест-выражение>
  [Case <список_выр-п>
    [<операторы-п>]] . . .
  [Case Else
    [<else-операторы-п>]]
End Select
```

где:

**тест-выражение** – любое числовое или строковое выражение;

**список\_выр-п** – список из одного или более выражений для соответствующего **Case**;

**операторы-п** – один оператор или несколько, выполняющихся, если **тест-выражение** имеет то же значение, что и значение одного из выражений **списка-п**;

**else-операторы-п** – один оператор или несколько, выполняющихся, если **тест-выражение** не совпадает ни с одним из значений **Case**-структур.

Существует ряд особенностей в выполнении структуры **Select Case**: для целых чисел условие отбора сработает и для соответствующего строкового подтипа, но для действительных чисел такое соответствие не наблюдается. как показано в следующем примере:

```
A = 1
Select Case A
  Case 1.1, 1.2, 1.3 MsgBox "A = 1.1, 1.2 или 1.3"
  Case "0.5", "1", "1.5" MsgBox "A = ""0.5"", ""1"", ""1.5""
    'будет выполнено только для Case "0.5", "1", "1.5"!
  Case 0.5, 1, 1.5 MsgBox "A = 0.5, 1.55, 1.56"
  Case Else MsgBox "Нет данных"
End Select

A = 1.55
Select Case a
  Case 1.1, 1.2, 1.3 MsgBox "A = 1.1, 1.2, 1.3"
  Case "0.5", "1.55", "1.56" MsgBox "A = ""0.5...""
  Case 0.5, 1.55, 1.56 MsgBox "A = 0.5, 1.55, 1.56"
    'будет выполнено только для Case 0.5, 1.55, 1.56!
  Case Else MsgBox "Нет данных"
End Select
```

Если же определить переменную  $a = "1.55"$  (строковое значение), в приведенном примере возникнет ошибка при выполнении с сообщением о несоответствии типов.

### Задания

1) Для Вашего варианта таблицы 4.1 задайте в окне ввода значение переменной  $X$  с учетом заданного подтипа данных.

При вводе маленьких или больших чисел с использованием буквы **e** (например,  $-1e15$ ) используйте преобразование подтипа строка в число с использованием функции **Eval**.

При вводе даты и времени используйте функцию преобразования подтипа **Cdate**.

При работе с датами учитывать, что их основные форматы #мм/дд/гггг# или #мм-дд-гг#, однако при написании года двумя цифрами и наличии на первом месте числа больше 12, формат преобразуется в #гггг-мм-дд# (см. далее Лабораторную работу № 10).

Вычислите переменную  $Y$  по одному из выражений в зависимости от значения  $X$ . Значения переменных  $X$  и  $Y$  покажите в окне сообщений.

Выполнить данное задание с использованием:

- а) строчного синтаксиса оператора условного перехода,
- б) блочного синтаксиса оператора условного перехода.

Таблица 4.1. Варианты заданий

| №   | Условие                    | Y                         | №   | Условие                               | Y                                   |
|-----|----------------------------|---------------------------|-----|---------------------------------------|-------------------------------------|
| 1.1 | $X \leq -10^{15}$          | Y = «маленькое число»     | 1.6 | X – месяц от 1 по 3                   | Y = «1-й квартал»                   |
|     | $X > -10^{15}$ и $X < 0$   | Y = «отрицательное число» |     | X – месяц от 4 по 6                   | Y = «2-й квартал»                   |
|     | $X \geq 0$ и $X < 10^{15}$ | Y = «положительное число» |     | X – месяц от 7 по 9                   | Y = «3-й квартал»                   |
|     | $X \geq 10^{15}$           | Y = «большое число»       |     | X – месяц от 10 по 12                 | Y = «4-й квартал»                   |
| 1.2 | X – символ до «Г»          | Y = 1                     | 1.7 | $X < -10^{308}$                       | Y = $-\infty$                       |
|     | X – символ от «Г» до «Ж»   | Y = 2                     |     | $X \geq -10^{308}$ и $X \leq 10^{30}$ | Y = «диапазон действительных чисел» |
|     | X – символ после «Ж»       | Y = 3                     |     | $X > 10^{308}$                        | Y = $+\infty$                       |

|     |                                                 |            |      |                                                        |                                      |
|-----|-------------------------------------------------|------------|------|--------------------------------------------------------|--------------------------------------|
| 1.3 | X – дата меньше 01.01.1900                      | Y= 19      | 1.8  | X от 0 по 255                                          | Y = «подтип Byte»                    |
|     | X – дата от 01.01.1900 до 31.12.1999            | Y= 20      |      | X от -32768 по 32767                                   | Y = «подтип Integer»                 |
|     | X – дата начиная с 01.01.2000                   | Y= 21      |      | X – целые числа другие                                 | Y = «подтип Long»                    |
| 1.4 | X – время от 0 час. 00 мин. до 6 час. 00 мин.   | Y= «ночь»  | 1.9  | X – дата и время = 1.1.2010 0:0:0                      | Y = «С Новым годом!»                 |
|     | X – время от 6 час. 01 мин. до 12 час.00 мин.   | Y= «утро»  |      | X – дата от 1 января 0000 года по 31 декабря 2099      | Y = «21 век!»                        |
|     | X – время от 12 час. 01 мин. до 18 час. 00 мин. | Y= «день»  |      | X – дата от 1 января 1900 года по 31 декабря 0099 года | Y = «20 век!»                        |
|     | X – время от 18 час. 01 мин. до 23 час. 59 мин. | Y= «вечер» |      | X – дата от 1 января 100 года по 31.12.9999            | Y = «верный диапазон дат!»           |
| 1.5 | X – месяц от 12 по 2                            | Y= «зима»  | 1.10 | X<0                                                    | Y= «X отрицательное»                 |
|     | X – месяц от 3 по 5                             | Y= «весна» |      | $X \geq 0$ и $X < 10^{-15}$                            | Y= «X маленькое положительное число» |
|     | X – месяц от 6 по 8                             | Y= «лето»  |      | $X \geq 10^{-15}$ и $X < 1$                            | Y= «X меньше 1»                      |
|     | X – месяц от 9 по 11                            | Y= «осень» |      | $X \geq 1$                                             | Y= «X не меньше 1»                   |

- 2) С использованием оператора выбора **Case** выполнить задания 1-го пункта:  
 для компьютеров с № 1 по № 5 – вариант 1.5, с № 6 по № 10 – вариант 1.6.

## Лабораторная работа № 5. Операторы цикла Do и While

**Оператор цикла** позволяет выполнить группу операторов несколько раз в соответствии с заданными условиями повтора.

Существует несколько видов оператора цикла:

- 1) *Do... Loop*
- 2) *While ... Wend*
- 3) *For ... Next*
- 4) *For Each ... Next.*

Данная лабораторная работа посвящена первым двум.

Синтаксис оператора *Do...Loop* следующий (здесь и далее в фигурных скобках {} приведены два возможных варианта, разделенных вертикальной чертой |, один из которых необходимо использовать):

- 1) первый вариант – проверка условия в начале цикла

```
Do [{While | Until} <условие>]
  [<операторы>]
  [Exit Do]
  [<операторы>]
Loop
```

- 2) второй вариант – проверка условия в конце цикла

```
Do
  [<операторы>]
  [Exit Do]
  [<операторы>]
Loop [{While | Until} <условие>]
```

где:

**условие** – логическое выражение, которое имеет значение истина (**True**) или ложь (**False**); значение условия **Null** то же, что и **False**;

для **While** (англ. *пока*) выполнение цикла продолжается, пока условие истинно,

для **Until** (англ. *до*) – выход из цикла, когда условие истинно;

**операторы** – один или несколько операторов, выполнение которых повторяется, пока условие после **while** истинно (**True**) или условие после **Until** ложно (**False**);

**Exit Do** – может использоваться, как альтернативный выход из цикла (на следующую строку программы после **Loop**); любое количество **Exit Do** может быть помещено внутри



цикла. Обычно эта команда используется с вычисляемым логическим выражением оператора *If...Then*.

Пример использования операторов циклов *Do While...Loop* и *Do Until...Loop*:

```
eps = 1e-7
a = 1
s = a
n = 2
Do While Abs(a) > eps ' цикл выполняется, пока |a| > eps
    ' или Do Until Abs(a) <= eps, что аналогично предыдущему
    a = - a * (2*n - 3) / (2*n - 1)
    s = s + a : n = n + 1
Loop
MsgBox("Расч. Pi = " & 4*S & vbCrLf & "n = " & n)
```

Результатом работы программы будет число  $\pi = 3,14159285358975$  при  $n=5000002$ , продолжительность расчета менее 30 сек. ( $\pi = 3,1415926535897932384626433832795$  на калькуляторе Windows). Если задать  $\text{eps} = 1e-8$ , то время расчета увеличится до 3 – 4 мин., результат расчета  $\pi = 3,14159267359025$  при  $n=50000002$ .

**Внимание!** При неверном написании условий окончания цикла программа может заикнуться (будет работать бесконечно долго). Чтобы прекратить выполнение заикнувшейся программы, необходимо открыть средство Windows *Диспетчер Задач* (Task Manager) с использованием сочетания клавиш Ctrl+Alt+Delete или, щелкнув правой кнопкой мыши на пустом месте панели задач, и выбрав в контекстном меню это средство, далее в разделе процессов найти и выделить *wscrip.exe* и нажать кнопку **Завершить процесс** (End Process).

Пример использования операторов циклов *Do...Loop While* и *Do...Loop Until* (результат работы – рисунок 5.1):

```
Randomize
Otv1 = vbYes
Otv2 = vbNo
Do
    Do
        MySum = FormatNumber(1000000*Rnd,2)
        'Случайное число от 0.00 до 1000000.00
        SSum = "Случайная сумма = " & MySum
        Otv2 = MsgBox(SSum & " руб." & vbCrLf & vbCrLf _
            & " Вам нравится такая сумма?", _
            vbYesNo, "Максимум - 1 000 000 руб.!")
    Loop Until Otv2 = vbYes
'выход из цикла, если нажата кнопка Да
```

```

proc = FormatNumber(MySum/1000000*100,1)
s = MySum & " руб. составляет " & proc & " % от 1
млн."
If proc>95 Then
    s = s & vbCrLf & " Редко бывает больше!"
ElseIf proc>85 Then
    s = s & vbCrLf & " Совсем неплохо!"
ElseIf proc>75 Then
    s = s & vbCrLf & " Бывает и больше...!"
Else
    s = s & vbCrLf & " Маловато!"
End If
Otvet1 = MsgBox(S & vbCrLf & vbCrLf & _
    "Повторить генерацию случайных чисел?", _
    vbYesNo, "Оценка результата... ")
Loop While Otvet1 = vbYes
    'цикл повторяется, если нажата кнопка Да

```

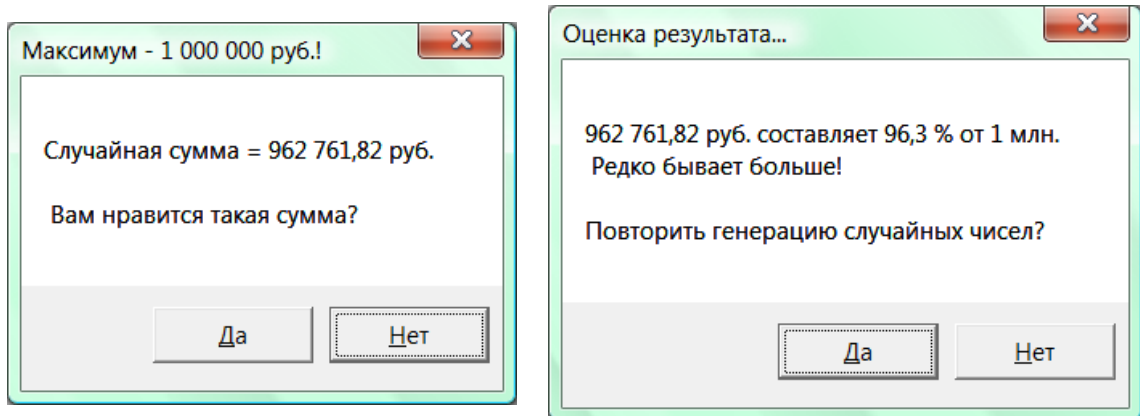


Рисунок 5.1. Пример использования циклов **Do...Loop**

Синтаксис оператора цикла **While...Wend** следующий:

```

While <условие>
    [<операторы>]
Wend

```

Выполнение операторов цикла повторяется, пока **<условие>** истинно (**True**).

Пример использования оператора **While...Wend** для расчета значения **y = arcctg(x)** с использованием итерационного ряда:

$$y = \text{arcctg}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{(2n+1)}$$

```

eps = 1e-28
k = 0 : s = 0 : a = 1 : z = 1

```

```

'x = 0.9 'вариант для предварительного тестирования
x = 0.9999999
t1=Time
Set W = CreateObject("WScript.Shell")
While Abs(a) > 1E-14
    a = z*x^(2*k+1)/(2*k+1)
    s = s + a
    k = k + 1
    z = -z
    If Int(k/1e6)=k/1e6 Then
        W.Popup "k = " & k & " Y = " & s, 1
    End If
Wend
t2=Time
MsgBox "Истинное значение "& Atn(x) & vbCrLf & _
    "Расчетное значение "& s & vbCrLf & _
    "Погрешность по a = " & FormatNumber(Abs(a),15) & _
    & vbCrLf & "Разница расч.- ист.= " & _
    FormatNumber(Abs(Atn(x)-s),15) & vbCrLf & _
    "Продолжительность расчета " & _
    FormatDateTime(t2-t1) & vbCrLf & _
    " k = " & k , vbExclamation, _
    "Y = arcctg(x). Вариант 1"

```

Результат выполнения этой программы показан на рисунке 5.2.

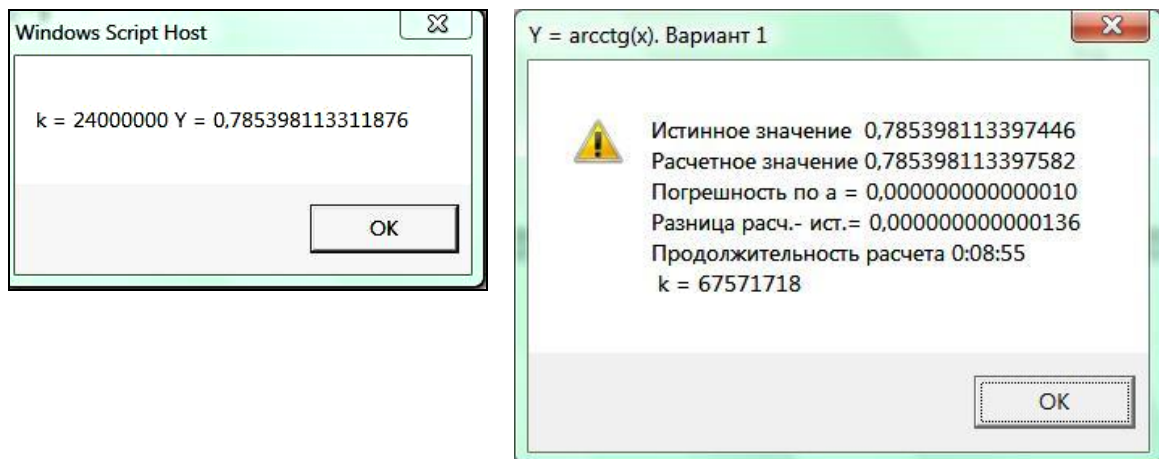


Рисунок 5.2. Результаты работы программы с циклом While . . . Wend

## Задания

Вычислите число  $\pi$  по итерационной формуле с номером, соответствующим номеру Вашего ПК, с абсолютной погрешностью вычисления от  $10^{-5}$  до  $10^{-16}$ . Найдите величину погрешности, при которой в числе  $\pi$  постоянными остаются 7 знаков после запятой. Определите программно время расчета для каждого варианта, покажите в окне со-

общений таблицу, показывающую расчетное значение  $\pi$ ,  $n$  и продолжительность расчета для различной погрешности.

Напишите пять вариантов программы для цикла **DO** с проверкой условия в начале и в конце и для цикла **WHILE**. Расчет для максимальной точности выполнить для одного варианта, т. к. его продолжительность может составлять 15 – 30 мин.

$$1) \quad \frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{4n^2}{4n^2 - 1} \qquad 2) \quad \frac{\pi}{3} = \sum_{n=0}^{\infty} (-1)^n \left( \frac{1}{6n+1} + \frac{1}{6n+5} \right)$$

$$3) \quad \frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \qquad 4) \quad \frac{1}{6} \pi^2 = \sum_{n=1}^{\infty} \frac{1}{n^2}$$

$$5) \quad \frac{1}{8} \pi^2 = \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2} \qquad 6) \quad \frac{\pi}{4} = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1}$$

$$7) \quad \frac{\pi}{4} = \sum_{n=0}^{\infty} (-1)^n \left( \frac{1}{10n+1} - \frac{1}{10n+3} + \frac{1}{10n+5} - \frac{1}{10n+7} + \frac{1}{10n+9} \right)$$

$$8) \quad \frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{4n^2}{(2n-1)(2n+1)}$$

$$9) \quad \pi = \sum_{n=0}^{\infty} \left( \frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6} \right) \left( \frac{1}{16} \right)^n$$

$$10) \quad \frac{\pi\sqrt{2}}{4} = \sum_{n=1}^{\infty} \left( \frac{(-1)^{n+1}}{4n-1} + \frac{(-1)^{n+1}}{4n-3} \right)$$

## Лабораторная работа № 6. Операторы цикла For и For Each

Синтаксис оператора цикла *For...Next* следующий:

```
For <счетчик> = <нач.знач.> To <кон.знач.> [Step <шаг>]  
    [<операторы>]  
    [Exit For]  
    [<операторы>]  
Next
```

где:

**счетчик** – числовая переменная, используемая как счетчик цикла; может быть положительной или отрицательной величиной

**нач.знач.** – начальное значение счетчика;

**кон.знач.** – конечное значение счетчика;

**шаг** – шаг изменения счетчика; на данную величину автоматически изменяется **счетчик** после каждого выполнения операторов цикла; если **шаг** не указан, значит он равен 1;

**операторы** – выполняются повторно столько раз, сколько определено значениями, заданными для **счетчика**: один раз, много раз или ни одного;

**Exit For** – может использоваться, как альтернативный выход из цикла; обычно используется с проверкой условия выхода в операторе **If...Then**; выход выполняется на строку программы, следующую за **Next**.

Пример использования цикла **For...Next** (см. рисунке 6.1):

```
S=""  
For i = 1 to 5.5 step 1/5.5  
    s = s & i & vbLF  
Next  
MsgBox S,, " For ... step 1/5.5"
```

Пример программы с альтернативным выходом (см. рисунок 6.2):

```
S = "      X          Y" & vbLF  
S = S & "-----" & vbLF  
For X = 1 to 5 step 0.11  
    Y = FormatNumber(Tan(X), 3)  
    If Abs(Y) < 0.1 Then  
        s = s & "-----" & vbLf _  
            & "Выход из цикла" & vbLf & "при |Y| < 0.1"  
        Exit For  
    End If  
    s = s & FormatNumber(X, 3) & "      " & Y & vbLf  
Next  
MsgBox S,, " For...Exit For...Next"
```

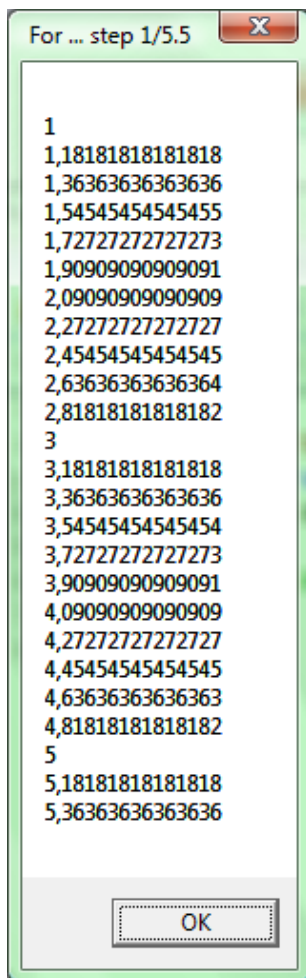


Рисунок 6.1.  
*For...Next*

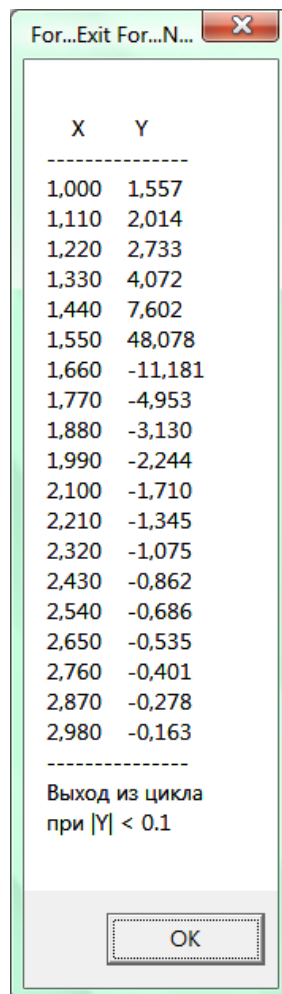


Рисунок 6.2. *For...Exit For...Next*

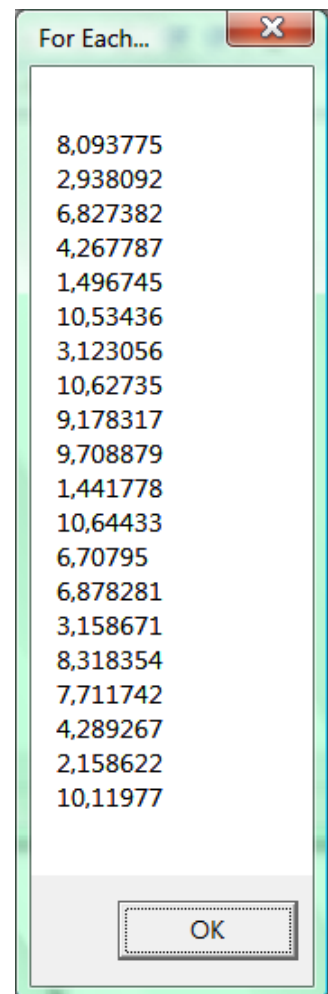


Рисунок 6.3. Цикл  
*For Each*

Синтаксис оператора цикла *For Each...Next* следующий:

```
For Each <элемент> In <группа>
    [операторы]
    [Exit For]
    [операторы]
Next [<элемент>]
```

где:

**элемент** – переменная, которая используется для перебора всех элементов коллекции или массива;

**группа** – имя коллекции объектов или массива.

Пример программы с использованием оператора *For Each* для работы с массивом (работа с коллекциями объектов будет рассмотрена далее):

```
N=19
ReDim x(19)
Randomize
For i = 0 to 19    'генерация 20-ти случайных чисел
```

```

    x(i) = 5 - 10*Rnd 'в диапазоне от -5.000 до 4.999
Next
S = ""
For Each iks in X
    S = S & iks & vbLf
Next
MsgBox S, , " For Each... "

```

В данной программе цикл **For Each** использует все значения массива *X* для формирования строки *S* (см. рисунок 6.3).

### Задания

С использованием оператора цикла **FOR** и функции **RND** сгенерировать массивы из *n* действительных чисел, необходимые для вычисления по заданной ниже формуле. Для расчета по заданной формуле использовать оператор **For Each**. Программу выполнить несколько раз для различных значений *n*. Исходные данные и результаты показать в окне сообщений.

|    |                                                                                                                   |
|----|-------------------------------------------------------------------------------------------------------------------|
| 1) | $S = \sum_{i=1}^n x_i \cdot x_j$                                                                                  |
| 3) | $c_i = a_i \cdot b_i$                                                                                             |
| 5) | $z_i = \frac{x_i}{x_{\max} - x_{\min}}$<br>$x_{\max}, x_{\min}$ - максимальное<br>и минимальное значения <i>x</i> |
| 7) | $S = \sum_{i=1}^n x_i / i$                                                                                        |
| 9) | $y_i = ax_i^2 + bx_i + c$                                                                                         |

|     |                                                               |
|-----|---------------------------------------------------------------|
| 2)  | $y_i = x_i \sum_{i=1}^n x_i$                                  |
| 4)  | $z_i = \frac{(n-i) \cdot x_i}{y_i}$                           |
| 6)  | $y_i = \frac{x_i}{x_{cp}}$<br>$x_{cp} = \sum_{i=1}^n x_i / n$ |
| 8)  | $y_i = \frac{x_i}{\sum_{i=1}^n x_i}$                          |
| 10) | $S = \sum_{i=1}^{n-1} \frac{x_i}{x_{i+1}}$                    |



## Лабораторная работа № 7. Процедуры и функции пользователя

Процедуры и функции используются для структурирования сложных программ, повышающего наглядность и читаемость программ, что облегчает процессы отладки. Процедуры и функции могут многократно использоваться при исполнении программы, для исполнения их с различными исходными данными может использоваться список аргументов.

Разница между процедурой и функцией:

имя функции после ее выполнения приобретает некоторое значение (имя функции возвращает вычисленное значение), в результате она может использоваться в составе выражений (математических, строковых и др.) программы;

имя процедуры служит только для запуска ее в работу, результатом выполнения могут быть значения переменных, массивов и пр. компонентов языка, являющихся аргументами процедуры или компонентами, общими для вызывающей программы и процедуры.

Синтаксис описания процедуры следующий:

```
[Public | Private] Sub <имя_проц.> [( <сп. арг.> )]  
    [<операторы>]  
    [Exit Sub]  
    [<операторы>]  
End Sub
```

где:

**Public** – показывает, что процедура доступна во всех других процедурах программы (является значением По умолчанию);

**Private** – показывает, что процедура доступна только в тех процедурах программы, где она объявлена;

**имя\_проц.** – имя процедуры, составленное по правилам написания идентификаторов;

**сп. арг.** – список аргументов, которые передаются процедуре, когда она вызывается для исполнения; элементы списка разделяются запятыми;

**операторы** – любая группа операторов, которая будет исполняться в процедуре.

Список аргументов имеет следующий синтаксис:

```
[ByVal | ByVal] <имя_переменной>[( )]
```

где:

**ByVal** – аргумент передается, как его значение (может быть переменной или константой при вызове процедуры, не может

возвращать из процедуры значения);

**ByRef** – аргумент передается, как ссылка; этому аргументу при обращении к процедуре должна соответствовать переменная (или массив) вызывающей программы и ее значение может изменяться после выполнения процедуры; по умолчанию параметры процедуры имеют тип **ByRef**;

**имя\_переменной** – имя переменной, являющейся аргументом процедуры, составленное по правилам написания идентификаторов..

Процедуры могут использовать локальные и глобальные переменные. Локальные переменные могут быть объявлены в процедуре с использованием структуры **Dim** <переменные>. Переменные, которые используются в процедуре, но явно не объявлены, также локальные, если они явно не объявлены на более высоком уровне за пределами процедуры.

Значения локальных переменных в процедурах не сохраняются между повторными вызовами процедур (области памяти под локальные переменные используются только во время работы процедуры).

Процедура не может быть описана внутри другой процедуры (т. е. не допускается использование вложенных процедур).

**Exit Sub** может использоваться для выхода из любого места процедуры с возвратом на следующую строку в программе после ее вызова.

Вызов процедуры необходимо выполнять с использованием структуры **[Call] <имя\_проц.> [( <сп. арг.> )]**. Можно не использовать ключевое слово **Call** при вызове процедуры, в этом случае список аргументов должен быть написан без круглых скобок сразу именем процедуры: **<имя\_проц.> <сп. арг.>**. Если используется ключевое слово **Call**, список аргументов должен быть написан в круглых скобках: **Call <имя\_проц.> ( <сп. арг.> )**.

Пример использования процедуры в программе:

```
N = InputBox("Задайте число", _
    "Вычисление факториала числа")
Call F_n(N, Fct)
MsgBox "Факториал числа " & N & " равен " _
    & FCT,, "Результат расчета"

Sub F_N(ByVal NF, FN)
    FN=1
    For i = 1 to NF
        Fn = Fn * i
    Next
    NF = 0 'только для доказательства, что N не изменится!
End Sub
```

Результат работы программы показан на рисунке 7.1.

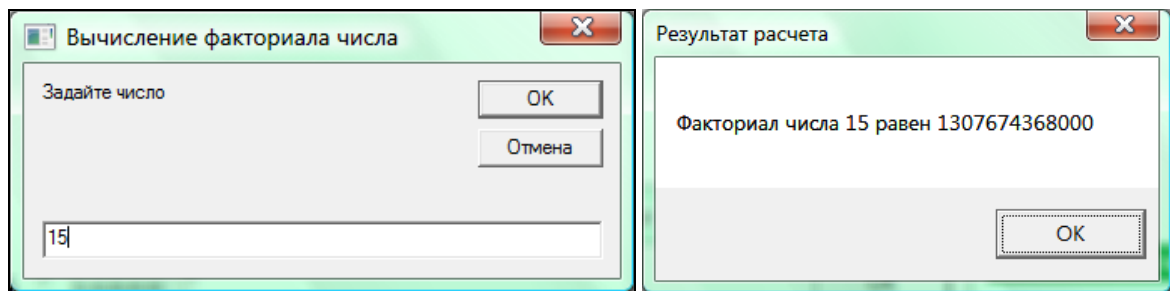


Рисунок 7.1. Результат выполнения программы с использованием процедуры Sub

Синтаксис описания функции следующий:

```
[Public | Private] Function <имя_функц.>  
[ (<сп.арг.> ) ]  
    [<операторы>]  
    [<имя_функц.> = <выражение>]  
    [Exit Function]  
    [<операторы>]  
    [<имя_функц.> = <выражение>]  
End Function
```

где:

**имя\_функц.** – имя функции, составленное по правилам написания идентификаторов;

**выражение** – вычисляемое значение, которое возвращает имя функции;

остальные обозначения те же, что и для процедуры.

Значение, возвращаемое функцией, по умолчанию имеет тип **Public**, если не указано иное (**Private**).

Аналогично процедуре, функция не может быть вложенной в другую функцию.

Пример использования функции в программе (для того, чтобы показать разницу в использовании процедур и функций, использован тот же алгоритм расчета факториала, что и в примере для процедуры):

```
N = InputBox("Задайте число", _  
    "Вычисление факториала числа")  
MsgBox "Факториал числа " & N & " равен " & F_N(N), , _  
    "Результат расчета"
```

```
Function F_N(NF)  
    F_N=1  
    For i = 1 to NF  
        F_N = F_N * i  
    Next  
End Function
```

Результат работы программы будет тот же, что и ранее (см. рисунок 7.1).

### **Задание**

- a) Выполните задания предыдущей лабораторной работы (№ 6) с заданием  $n$  в окне **InputBox** и использованием процедуры пользователя для расчета по заданной формуле. Формирование исходных данных и вывод результатов в окно **MsgBox** выполнить в головной программе. Процедура пользователя не должна использовать глобальные переменные.
- b) Выполните задания лабораторной работы № 5 с заданием погрешности вычисления в окне **InputBox** и использованием функции пользователя для расчета числа  $\pi$  по заданной формуле. Формирование исходных данных и вывод результатов в окно **MsgBox** выполнить в головной программе. Функция пользователя не должна использовать глобальные переменные.

## Лабораторная работа № 8. Работа с числовой информацией

При выполнении математических вычислений используются символы математических операций:

= присваивание, + сложение, – вычитание, / деление, \* умножение, ^ возведение в степень.

Математических функция в языке *VBScript* достаточно много:

**Abs Atn Cbool CByte CCur CDbl CInt CLng Cos CSng CStr  
Exp Fix Int FormatCurrency FormatNumber FormatPercent  
Hex \ Lbound Log Mid Mod Oct Randomize Rnd RGB Round  
Sgn Sin Tan TypeName Ubound VarType.**

Описание функций приведено в приложении 1.

Порядок выполнения операций в математических выражениях – общепринятый в математике (приоритет математических операций):

- 1) математические функции,
- 2) возведение в степень,
- 3) умножение и деление,
- 4) сложение и вычитание.

Круглые скобки ( ) могут использоваться для изменения порядка выполнения этих операций. Вначале выполняются операции в скобках, затем вне их. Если используются вложенные скобки, вначале выполняются действия во внутренних скобках, затем во внешних.

Например, для математической записи формулы:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

текст в программе на языке *VBScript* будет следующий:

$$x = (-b + (b^2 - 4*a*c)^(1/2)) / (2*a)$$

Особенность записи формул – нельзя пропускать знаки умножения, как это часто делается в математике.

Для всех тригонометрических формул (**Atn**, **Cos**, **Sin**, **Tan**) единица измерения углов – радианы, например, число  $\pi$  (пи) можно вычислить следующим образом:

$$pi = 4 * Atn(1)$$

Для сложных математических формул иногда бывает целесообразно выполнять вычисление по частям, например:

$$y = \frac{tg^2(2b + c)}{\sqrt{\left| \sqrt[3]{(a - b)^2} - \frac{e^{-b^2} \cdot \ln a}{a + \ln(2b + c)} \right|}}$$

можно вычислить следующим способом:

```

y1 = (Tan(2*b + c))^2
y2 = (a - b)^(2/3)
y3 = (Exp(-b^2)*Log(a))/(a + Log(2*b + c))
y4 = abs(y2 - y3)^(1/2)
y = y1/y4

```

При записи такой формулы в одну строку получилось бы следующее выражение:

$$y = (\text{Tan}(2*b + c))^2 / \text{abs}((a - b)^{(2/3)} - (\text{Exp}(-b^2) * \text{Log}(a)) / (a + \text{Log}(2*b + c)))^{(1/2)}$$

которое достаточно трудно проверить на правильность написания.

Для вычисления логарифмов с разными основаниями можно использовать следующую программу (в примере – вычисление десятичного логарифма, см. рисунок 8.1):

```

n = 10
x = 123.45
MsgBox " Lg(" & x & ") = " _
      & logn(x,n) , , "Lg(x) "
Function logn(xf,nf)
    logn = Log(xf) / Log(nf)
End Function

```

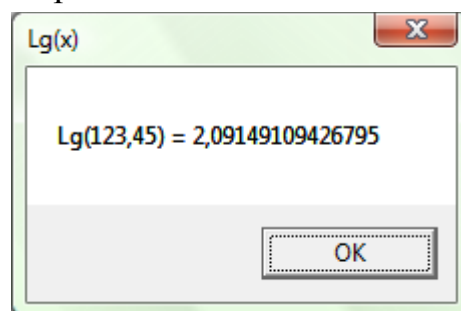


Рисунок 5.21. Вычисление логарифма  $Lg(x)$

### Задания

Напишите программу вычисления  $Y$  по заданному математическому выражению. Исходные данные и результат показать в окне сообщений.

| Вариант | Выражение                                                                 | Значение переменных |        |        | Результат |
|---------|---------------------------------------------------------------------------|---------------------|--------|--------|-----------|
|         |                                                                           | $a$                 | $b$    | $c$    |           |
| 1)      | $y = \sqrt{a(b^{1/3} + 17)} + \frac{a^5 + a^2 + \sqrt{c}}{2b}$            | 1,5                 | 10,2   | 10,034 | 6,00      |
| 2)      | $y = \sqrt{3(a+b)/(b^3 + 17)} + \frac{a^2 + \sqrt[3]{c}}{(5-a)}$          | 13,5                | 0,92   | 1,05   | -20,00    |
| 3)      | $y = \lg \frac{a + \sin(b)}{\cos 2a} + \sqrt[4]{5 + \sqrt{2+c}}$          | 0,785               | 1,5708 | 3,777  | 5,00      |
| 4)      | $y = \text{tg} \frac{a - 24\sqrt{2b+3}}{2a^{2/3} + \lg(3b)}$              | 9,01                | 7,7058 | –      | 0,05      |
| 5)      | $y = \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2+a}}}} + \sin \frac{13-b}{c^5}$ | 2,0                 | 13,0   | 0,8    | 2,00      |

|     |                                                                                       |        |        |        |         |
|-----|---------------------------------------------------------------------------------------|--------|--------|--------|---------|
| 6)  | $y = \sqrt[3]{3 + \sqrt[3]{3 + \frac{tg^2(a)}{32 + b}}} + \lg \frac{a^3}{a + b + 2c}$ | 3,1415 | 96,6   | 18,4   | 1,00    |
| 7)  | $y = \frac{tg^2(a) - tg^2(2b + c)}{\sqrt{34 - \frac{\lg(a)}{a + \lg(2b + c)}}}$       | 1,5    | 1,725  | 5,2425 | 34,00   |
| 8)  | $y = \frac{\sqrt{1 - \frac{\sin^2(a) + \cos^2(b)}{a + b + c}}}{(a + b + c)^{2/3}}$    | 3,1416 | 1,5708 | -1,884 | 0,50    |
| 9)  | $y = tg \frac{\lg^2(a + b + c) + \sin(a^2)}{\sqrt{2 + \frac{14 - b}{2c}}}$            | 0,5    | 6,385  | 4,201  | 1,00    |
| 10) | $y = \frac{b^2 - 4ac}{\sin^2(a) + \sin^2\left(\frac{2\sqrt{a}}{b - c}\right)}$        | 9,5    | -4,51  | 2,2093 | -100,00 |

## Лабораторная работа № 9. Работа со строковой информацией

Функции, которые могут использоваться при работе со строками, следующие (описание см. в Приложении 2):

**Asc Chr & InStr InStrRev Join Lcase Left Len LTrim Mid RTrim Trim Replace Right Space Split String StrComp StrConv StrReverse Tab TypeName Ucase VarType.**

Пример использования строковых функций:

- a) присвоить значение переменной **FIO**  
**FIO = "Иванов Петр Сидорович"**
- b) написать **FIO** прописными буквами  
**FIO\_p = Ucase(FIO)**
- c) разделить **FIO** на 3 переменные: фамилию, имя и отчество (без использования функции **Split**, пример с ее использованием приведен далее):

```
n1 = InStr(FIO, " ") 'n1=7, позиция первого пробела в FIO
F1 = Left(FIO, n1-1) 'F1="ИВАНОВ", 6 символов слева
n2 = InStr(n1+1, fio, " ", 1) 'm2 = 12
F2 = Mid(FIO, n1+1, n2-n1-1)
'F2 = "Петр", 4 (12-7-1) символа начиная с 8
L = Len(FIO) 'L = 21
F3 = Mid(FIO, n2+1, L-n2) 'F3 = "Сидорович"
```

- d) получить строку – инициалы и фамилия  
**F4 = Left(F2,1) & ". " & Left(F3,1) & ". " & F1**  
**'F4 = "П. С. ИВАНОВ"**

- e) использование функции **Split**:

```
f = Split(FIO) 'далее можно использовать циклы
' For Each или For с функциями Lbound и Ubound
For I = Lbound(f) to Ubound(f)
    MsgBox f(i)
Next
```

На рисунке 9.1 показаны два примера выполнения программы, содержащей описанные выше операции работы со строками.

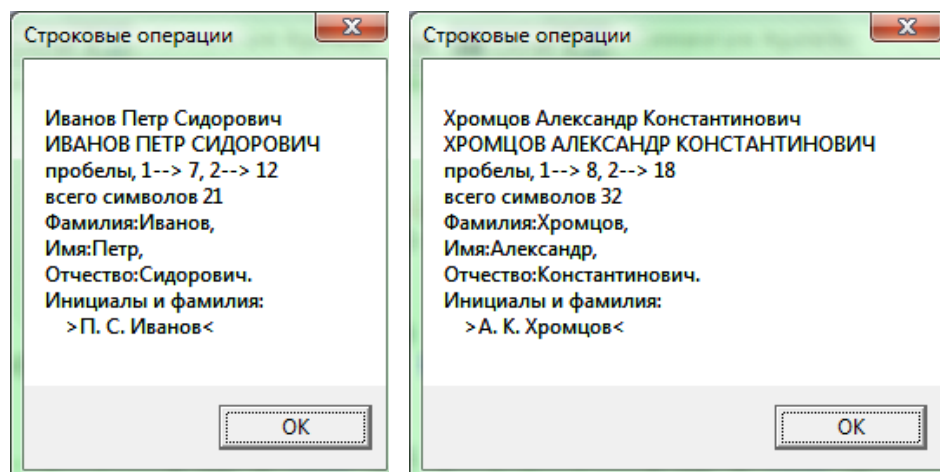


Рисунок 9.1. Использование функций для обработки строковой информации



## Задания

Напишите программу для своего варианта задания. Исходные данные и результаты показать в окне сообщений. Описание функций для работы со строками см. в Приложении 2. Программа должна сама определять количество символов или слов в исходной строке текста.

- 1) В окне **InputBox** задать строку текста и показать код каждого символа этой строки.
- 2) Показать в окне сообщений символы с кодами от 128 до 148.
- 3) Определить позиции всех пробелов в строке, состоящей из любого числа слов.
- 4) Определить позиции с конца строки всех букв «е» и «а» для строки, состоящей из любого числа слов.
- 5) Задать текстовые значения не менее пяти элементов строкового массива и создать объединением элементов массива (**Join**) одну строковую переменную.
- 6) Двумя вариантами (без использования и с использованием **Split**) разделить строку, состоящую из произвольного числа слов, на отдельные слова с присвоением полученных значений переменным.
- 7) Определить количество символов в каждом слове предложения, состоящего из любого числа слов.
- 8) Заменить в данном предложении все буквы «е» на «Е» и «о» на «О», начиная с 10-го символа.
- 9) Для данного предложения выполнить следующие операции: А) преобразовать все его символы в строчные; Б) преобразовать все символы в прописные; В) вариант «Б» преобразовать в вид исходного предложения.
- 10) Написать все слова предложения, состоящего из любого числа слов, в обратном порядке (сначала последнее слово, затем предпоследнее и т. д., кончая первым словом). Использовать функции **Split** и **Join**.

## Лабораторная работа № 10. Работа с информацией типа дата и время

Основной формат даты #мм/дд/гггг#, #мм-дд-гг# или #Mes-дд-гггг, т. е. на первом месте стоит месяц, на втором – день, на третьем – год с разделителями косая черта с правым наклоном (/) или дефис (-). Например, #12/31/2008#, #12/31/8#, #12-31-2008# (31 декабря 2008 года). Однако, при написании названия месяца (или 3 букв названия) может использоваться формат, #дд-Mes-гг#, например, #31-Дец-08#.

При написании года двумя цифрами формат даты имеет изменяемый характер. Например, #12-04-08# означает 4 декабря 2008 г., но #13-mm-08# означает 08 <мм> 2013 г., однако #13-02-29# = 13 февраля 2029 г. Если на первом месте стоит число больше 12, формат преобразуется в #гггг-мм-дд#. Если на последнем месте стоит число, больше 28 или 29 для високосного года, формат преобразуется в #дд-мм-гггг#. Если на последнем месте стоит число, больше 30 или 31 для месяцев, в которых 30 и 31 день, формат преобразуется в #дд-мм-гггг#, как показано в следующем примере:

```
MsgBox #13-01-31# '31.01.2013
MsgBox #13-02-31# '13.02.1931
MsgBox #13-03-31# '31.03.2013
MsgBox #13-04-31# '13.04.1931
MsgBox #13-05-31# '31.05.2013
MsgBox #13-06-31# '13.06.1931
MsgBox #13-07-31# '31.07.2013
MsgBox #13-08-31# '31.08.2013
MsgBox #13-09-31# '13.09.1931
MsgBox #13-10-31# '31.10.2013
MsgBox #13-11-31# '13.11.1931
MsgBox #13-12-31# '31.12.2013
```

В связи с таким сложным поведением даты для однозначного ее написания лучше пользоваться четырьмя цифрами года в дате.

Функция **CDate** конвертирует строку в дату с учетом региональных установок Windows. В русских настройках даты (Панель управления – Язык и региональные стандарты – текущий формат – русский) по умолчанию задан формат даты dd.MM.yyyy. Поэтому **Cdate("13-12-08")** = 13 декабря 2008 г., тогда как **#13-12-08#** = 8 декабря 2013 г.

При неверном задании даты возникает системная ошибка (например, дата #02/29/08# правильная – високосный год, но #02/29/07# – неверная дата!).

Функции, которые могут использоваться при работе с данными типа дата и время, следующие (описание см. в приложении 3):

**CDate DateAdd DateDiff DatePart DateSerial DateValue**

| Day     | FormatDateTime | Hour       | Minute    | Month    | MonthName | Now |
|---------|----------------|------------|-----------|----------|-----------|-----|
| Second  | Time           | TimeSerial | TimeValue | TypeName | VarType   |     |
| Weekday | WeekdayName    | Year       |           |          |           |     |

Дату и время на часах компьютера возвращают функции **Now** и **Time**.

Если заданы два значение типа дата и время, операция вычитания даст разницу между ними в днях в виде действительного числа.

Например, разница #05-02-2008 18:00# - #05-01-2008 12:00# будет равна 1.25 дня.

Если необходимо вычислить разницу в определенных единицах (годах, кварталах, месяцах, неделях, днях, часах, минутах и секундах), следует использовать функцию **DateDiff**, как показано в следующем примере:

```
Dt1 = #31-Dec-2005 12:00:00#
Dt2 = #14-Apr-2009 18:01:01#
MsgBox "Дата 1: " & Dt1 & vbCrLf & _
        "Дата 2: " & Dt2 & vbCrLf & _
        "лет (yyyy)      " & DateDiff("yyyy",dt1, dt2) & vbCrLf & _
        "кварталов (q)    " & DateDiff("q", dt1,dt2) & vbCrLf & _
        "месяцев (m)      " & DateDiff("m", dt1,dt2) & vbCrLf & _
        "недель (ww)      " & DateDiff("ww",dt1,dt2) & vbCrLf & _
        "дней года (y)     " & DateDiff("y", dt1,dt2) & vbCrLf & _
        "дней (d)          " & DateDiff("d", dt1,dt2) & vbCrLf & _
        "часов (h)         " & DateDiff("h", dt1,dt2) & vbCrLf & _
        "минут (n)         " & DateDiff("n", dt1,dt2) & vbCrLf & _
        "секунд (s)        " & DateDiff("s", dt1,dt2), , _
        "Функция DateDiff"
```

Результат работы данной программы показан на рисунке 5.23.

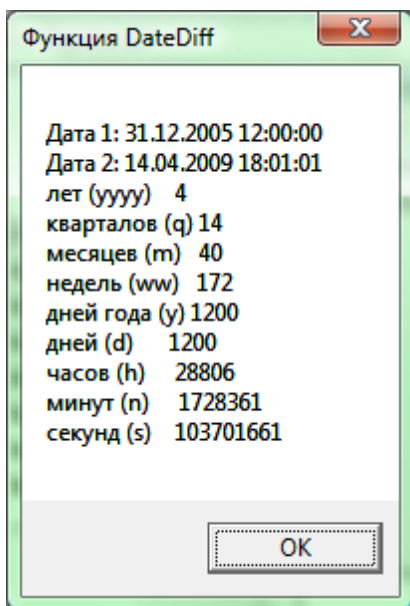


Рисунок 5.23. Использование функции **DateDiff**

Аналогичные параметры задания единицы измерения имеет функция **DateAdd**, позволяющая прибавить заданный диапазон даты и времени к начальному значению, как показано в примере (результат выполнения программы – рисунок 5.24):

```
Dt1 = #31-Dec-2005 12:00:00#
MsgBox "Дата: " & Dt1 & vbCrLf & vbCrLf & _
"+1 год (yyyy)      " & DateAdd("yyyy",1,Dt1) & vbCrLf & _
"+1 квартал (q)     " & DateAdd("q", 1,Dt1) & vbCrLf & _
"+1 месяц (m)       " & DateAdd("m", 1,Dt1) & vbCrLf & _
"+1 неделя (ww)     " & DateAdd("ww",1,Dt1) & vbCrLf & _
"+1 день года (y)   " & DateAdd("y", 1,Dt1) & vbCrLf & _
"+1 день (d)        " & DateAdd("d", 1,Dt1) & vbCrLf & _
"+1 час (h)         " & DateAdd("h", 1,Dt1) & vbCrLf & _
"+1 минута (n)      " & DateAdd("n", 1,Dt1) & vbCrLf & _
"+1 секунда (s)     " & DateAdd("s", 1,Dt1),, _
"Функция DateAdd"
```

Те же параметры задания возвращаемой части даты-времени у функции **DatePart** (результат показан на рисунке 5.25):

```
Dt1 = #31-Dec-2005 12:01:01#
MsgBox "Дата: " & Dt1 & vbCrLf & vbCrLf & _
"год (yyyy)        " & DatePart("yyyy",Dt1) & vbCrLf & _
"квартал (q)       " & DatePart("q", Dt1) & vbCrLf & _
"месяц (m)         " & DatePart("m", Dt1) & vbCrLf & _
"неделя (ww)       " & DatePart("ww",Dt1) & vbCrLf & _
"день года (y)     " & DatePart("y", Dt1) & vbCrLf & _
"день месяца (d)   " & DatePart("d", Dt1) & vbCrLf & _
"час (h)           " & DatePart("h", Dt1) & vbCrLf & _
"минута (n)        " & DatePart("n", Dt1) & vbCrLf & _
"секунда (s)       " & DatePart("s", Dt1),, _
"Функция DatePart"
```

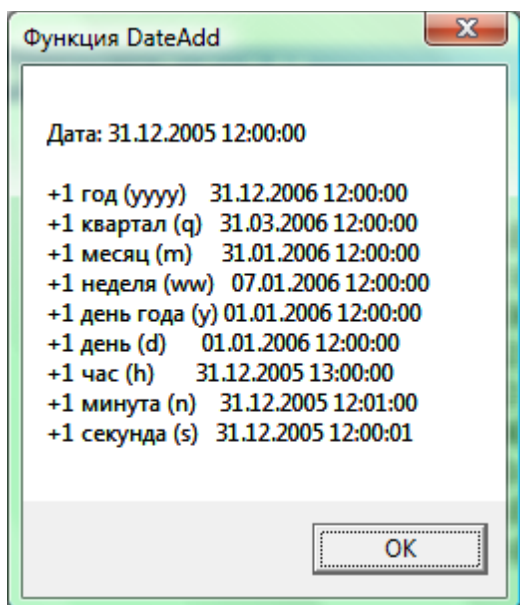


Рисунок 5.24. Использование функции **DateAdd**



Рисунок 5.25. Использование функции **DatePart**

Год, месяц, день, час, секунду для заданной даты и времени можно также определить с помощью функций **Year**, **Month**, **Day**, **Hour**, **Minute**, **Second**.

При задании в тексте программы данных подтипа дата и время можно использовать английские названия месяцев (нельзя русские). Однако, при использовании функции преобразования строки в дату-время все наоборот, если в региональных установках Windows задан русский формат дат:

```
StrDt = "1 Окт 1999"    ' Строковое значение  
Data1 = CDate(StrDt)  ' Преобразование в дату:  
                        ' Data1 будет равна #10-01-1999#
```

Тот же результат дает функция **DateValue**:

```
Date2 = DateValue(StrDt)
```

Если заменить в региональных настройках формат дат на English, система перестает понимать русские названия месяцев и признает только английские (**StrDt** = "1 Oct 1999": **Data1** = **CDate**(**StrDt**)).

Функция **Weekday**(**Data1**, **vbMonday**) покажет день недели для заданной даты. В этой функции задан первый день недели – понедельник, если опустить этот параметр, первым днем недели будет воскресенье (что соответствует английскому календарю).

Формат вывода информации подтипа дата-время можно определить с использованием функции **FormatDateTime**, в которой существует 5 форматов (**vbGeneralDate**, **vbLongDate**, **vbShortDate**, **vbLongTime**, **vbShortTime**, описание этих и других констант – в Приложении 4).

Название 7-го дня недели позволяет определить функция **WeekDayName**(7, **False**, **vbUseSystem**) – в данном случае при русских региональных настройках операционной системы вернет «воскресенье».

## **Задания**

Для приведенных ниже вариантов заданий исходные данные и результаты показать в окне сообщений.

- 1) Задайте в программе строковую переменную, значение которой равно текущей дате с написанием в ней месяца названием. Преобразуйте значение переменной в подтип «дата». Вычислите количество прожитых Вами дней.
- 2) Определите текущую дату на часах компьютера, прибавьте к ней 1 год, затем 3 месяца и 25 дней и определите название дня недели полученной даты.

- 3) Задайте в программе строковую переменную, значение которой равно текущему времени с точностью секунд. Преобразуйте значение переменной в подтип «время». Вычислите количество секунд, оставшихся до конца суток.
- 4) Определите текущую дату и время на часах компьютера, прибавьте к нему 25 часов, 30 минут и 30 секунд, и определите для полученного значения количество часов, минут и секунд, прошедших от начала суток.
- 5) Вычислите количество дней, часов, минут и секунд, прошедших с начала 21 века до текущего момента, который взять с часов компьютера.
- 6) Рассчитайте стаж работника – количество целых лет, кроме того целых месяцев и дней (например, 10 лет 1 месяц и 1 день) к текущему моменту времени, который определить по часам компьютера.
- 7) Рассчитайте количество рабочих дней при пятидневной рабочей неделе с 1.09.2010 по 30.11.2010.
- 8) Рассчитайте количество выходных дней при пятидневной и шестидневной рабочей неделе с 1.01.2010 по 31.10.2010.
- 9) Рассчитайте количество отработанных часов за период с 1.03.2010 по 31.05.2010 с учетом того, что в этом периоде один праздничный день.
- 10) Рассчитайте количество учебных часов за период с 1.09.2008 по 31.11.2010 с учетом того, что количество их по дням недели следующее: понедельник – 6, вторник – 4, среда – 8, четверг – 5, пятница – 4, суббота и воскресенье – нет занятий.

## Лабораторная работа № 11. Работа с логическими выражениями

При написании логических условий используются операции сравнения данных (подтипов число, строка, дата и время):

= равно,  
<> не равно,  
< меньше,  
<= меньше или равно,  
> больше,  
>= больше или равно.

Кроме того, используются логические операции:

**And Eqv Imp Is IsArray IsDate IsEmpty IsNull IsNumeric Not Or Xor.**

Результатом выполнения логической операции является одно из двух возможных значений:

**True** (*Истина*) или

**False** (*Ложь*).

Переменной можно присвоить логическое значение **True** или **False** (**L1 = True** или **L1 = False**), но нельзя использовать русские значения *Истина* или *Ложь* в присвоении и логических выражениях, не смотря на то, что русские версии Windows в окне сообщений показывают именно эти значения (см. рисунок 11.1).

При сравнении символьных и строковых значений учитывается регистр букв (прописные или строчные).

Приоритет при вычислении выражений, в которых присутствуют логические компоненты:

- 1) арифметические операторы;
- 2) операторы объединения (конкатенации) строковых значений (**&**, **+**);
- 3) операции сравнения данных, которые используют символы **=**, **<>**, **<**, **<=**, **>**, **>=**;
- 4) логические операции **And**, **Or**, **Not**, **Xor**, **Eqv**, **Imp**.

В операциях одного приоритета порядок вычислений слева-направо. Порядок может быть изменен при использовании круглых скобок.

Результат вычисления логического выражения может быть присвоен переменной, показан в окне **MsgBox**, использован для организации разветвляющихся алгоритмов в операторах **If...**, **Select Case...** и циклов в структурах **Do [While | Until]...**, **While**.

Пример программы с использованием логических выражений:

```
a = 1 > 10           'False
b = "я" > "Я"        'True
c = "Иванов" < "Петров" 'True
```

```

d = 1 < 10 and "я" > "Я"           'True
e = 1 > 10 or #31-12-2007# < #01-01-2008# 'True
MsgBox a & vbCrLf & b & vbCrLf & c & vbCrLf & d & vbCrLf & e

```

Результат выполнения приведенного выше текста программы показан на рисунке 11.1.

Краткое описание логических операций (здесь и в таблице 5.5 **E1** и **E2** – логические выражения.):

- **And** – логическое **И** (если выражения слева и справа от него истинны, результат **True**, иначе **False** или **Null**);
- **Or** – логическое **ИЛИ** (должно быть истинным хотя бы одно из выражений);
- **Not** – логическое отрицание (возвращает **True**, если условие ложно и наоборот);
- **Xor** – логическое исключение (выражение **E1 Xor E2** возвращает **True**, если только **E1 = True** или только **E2 = True**, иначе – **False**);
- **Eqv** – эквивалентность двух выражений, возвращает **True**, если они имеют одинаковое значение;
- **Imp** – импликация (**E1 Imp E2** возвращает **False**, если **E1 = True** и **E2 = False**, иначе – **True**).

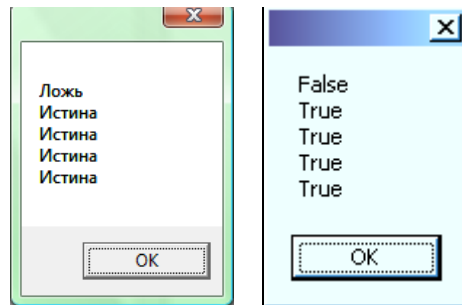


Рисунок 11.1. Результат вычисления логических функций в русской и английской версиях Windows

Полное описание результатов вычисления логических операций приведено в таблице 5.5.

Таблица 5.5. Результаты логических операций

| <b>E1</b> | <b>E2</b> | <b>E1 And E2</b> | <b>E1 Or E2</b> | <b>E1 Imp E2</b> | <b>E1 Xor E2</b> |
|-----------|-----------|------------------|-----------------|------------------|------------------|
| True      | True      | True             | True            | True             | False            |
| True      | False     | False            | True            | False            | True             |
| True      | Null      | Null             | True            | Null             |                  |
| False     | True      | False            | True            | True             | True             |
| False     | False     | False            | False           | True             | False            |
| False     | Null      | False            | Null            | True             |                  |
| Null      | True      | Null             | True            | True             |                  |
| Null      | False     | False            | Null            | Null             |                  |
| Null      | Null      | Null             | Null            | Null             |                  |

## Задания

В окне сообщений показать результаты вычисления логических значений:

- 1) **"А" > "f"** и **1e-10=1/1e10**



- 2) «Правда» > «Ложь» или True не равно False
- 3) Sin(0,5) > cos(0,5) и «Саша» > «Леши»
- 4) «эврика» = «eureka» или «привет!» = «Hi!»
- 5) 0 часов 5 мин. 1.1.2008 > 20 часов 15 мин. 31.12.2007
- 6) vbOKOnly > vbOKCancel и vbCritical < vbQuestion
- 7) vbLf=Chr(10) и vbNullChar = Chr(0)
- 8) vbSunday = 1 и vbMonday=2 и vbTuesday=3
- 9) vbBlack=&h00 и vbRed=&hFF и vbWhite = &hFFFFFF
- 10) Exp(1) = Exp(-1) или Abs(Sin(1)) = Abs(Sin(-1))

## Лабораторная работа № 12. Работа с объектами WScript

Программа на языке *VBScript* сама по себе является объектом **WScript** сервера сценариев *Windows Script Host*, (т. е. после запуска программы этот объект уже существует, не нужно давать команду о его создании). Объект **WScript** – корневой в объектной иерархии *Windows Script Host*, в которую входят три *COM*-библиотеки (*COM-Component Object Model*): *WshController*, *WshNetwork* и *WshShell* (см. рисунок 12.1 с сайта [http://msdn.microsoft.com/en-us/library/a74hyuw0\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/a74hyuw0(VS.85).aspx)).

В программах на *VBScript* могут использоваться и другие библиотеки классов *Windows* (например, объекты библиотеки **Microsoft ADO** – классы для работы с базами данных и пр.), библиотеки других систем, поддерживающих интерфейс *ActiveX* (*OLE Automation*), например, библиотеки *Microsoft Office*, *Internet Explorer* и пр.

Библиотеки классов содержат описания объектов, образующих иерархическую структуру.

Для создания в программе экземпляра объекта (не **Wscript**!) используется следующий синтаксис оператора присваивания:

```
Set <Переменная> = CreateObject("<Библиотека.Класс>")
```

С использованием этой функции создается переменная подтипа **Object**, после чего можно узнать свойства созданного объекта и пользоваться его методами (процедурами и функциями объекта).

**Свойство объекта** имеет определенное значение, которое можно узнать или изменить.

Значение свойства объекта можно использовать в операторе присваивания и в любых других выражениях с использованием синтаксиса **<Имя объекта>.<Имя свойства>** (имя объекта и имя его свойства, разделенные точкой), например:

```
<Переменная> = <Имя объекта>.<Имя свойства>
```

или

```
MsgBox <Имя объекта>.<Имя свойства>
```

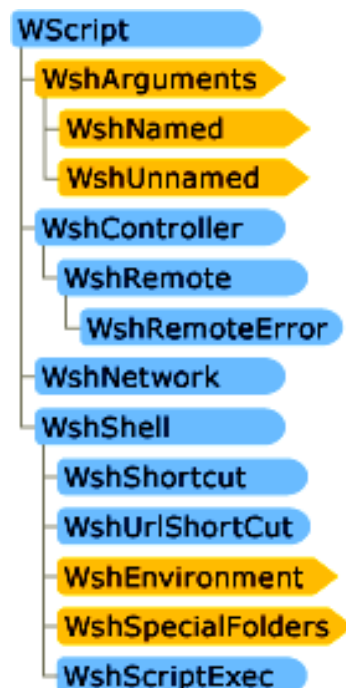


Рисунок 12.1.  
Объектная модель  
*Windows Script Host*

В некоторых случаях свойство может возвращать ссылку на коллекцию объектов или значений, в этом случае возможно только присвоение этого свойства переменной с использованием слова **Set**, как при создании объекта:

```
Set <Переменная> = <Имя объекта>. <Имя метода>
```

Чтобы изменить свойство объекта, достаточно присвоить ему какое-либо значение:

```
<Имя объекта>. <Имя свойства> = <Значение>
```

Присваиваемое значение может быть константой, выражением, свойством другого объекта, возвращаемым значением какого либо метода:

```
<Имя объекта>. <Имя свойства> = _  
    <Имя объекта2>. <Имя метода ([параметры]) >
```

**Метод объекта** – это его процедура, которая может выполнять какие-либо действия, получать и возвращать значения (параметры метода).

При использовании метода с параметрами следует использовать следующий синтаксис:

- 1) если метод не возвращает значений:

```
<Имя объекта>. <Имя метода> <п1> [, <п2>, ...] >
```

где **п1**, **п2** и т. д. – параметры;

- 2) если метод возвращает какое-либо значение (но не объект и не коллекцию объектов), обязательно использование круглых скобок:

```
<Переменная> = <Имя объекта>. <Имя метода> ([<п1> [, <п2>, ...]]) >
```

- 3) если метод возвращает ссылку на новый объект или их коллекцию – обязательно использование слова **Set**, как при создании объекта:

```
Set <Переменная> = <Имя объекта>. <Имя метода> <п1> _  
    [, <п2>, ...] >
```

При работе с объектами может использоваться структура языка **VBScript**:

```
With <Имя объекта>  
    операторы  
End With
```

Для операторов между **With** и **End With** можно использовать методы и свойства заданного объекта, которые начинаются с точки (.) без указания предшествующего имени объекта, как показано далее в примерах программ.

Объект **WScript** имеет следующие методы: *Sleep Quit Echo CreateObject ConnectObject DisconnectObject GetObject*, и свойства: *Arguments BuildVersion FullName Interactive Name Path ScriptFullName ScriptName Timeout Version StdIn StdOut StdErr* (последние 3 свойства можно использовать только для программы, запущенной с помощью *CScript.exe*, т. е. в окне *Cmd*).

Объект **Wscript** имеет подчиненный объект **Shell**, с методами: *AppActivate CreateShortcut Exec ExpandEnvironmentStrings LogEvent Popup Run* и пр.,

и свойствами: *CurrentDirectory Environment SpecialFolders*.

Описание этих объектов, их методов и свойств можно найти в справке системы Microsoft Visual Studio или на сайте корпорации Microsoft.

Методы объекта имеют один параметр (например, **Exec** *<Command>*) или несколько параметров. Данный метод используется для запуска исполняемых файлов.

Так, метод **Run** имеет следующий синтаксис:

**Run (<Command>, <WindowStyle>, <WaitOnReturn>)**

где **Command** – командная строка запуска приложения;

**WindowStyle** – необязательный параметр, число, определяет вид окна запускаемого приложения:

1 – запускает программу в окне обычного размера;

2 – запускает программу в свёрнутом окне;

3 – запускает программу в максимизированном окне;

**WaitOnReturn** – необязательный параметр, логическое значение: **C** – программа VBS возобновит работу только после завершения вызванного процесса, **False** (по умолчанию) – управление передаётся сразу на следующую строку программы.

Метод **Run** позволяет в качестве параметра **Command** указывать имя файла данных, известного для Windows типа. Путь к файлу данных следует указывать в формате 8.3 (без пробелов и длинных названий папок).

Методы объекта **WScript** позволяют не только управлять выполнением программы, но и устанавливать связь с другими объектами *ActiveX (OLE Automation)* – **Wscript.ConnectObject (<ProgID>, <Prefix>)**, создавать объекты существующих библиотек – **Wscript.CreateObject (<ProgID>, <Prefix>)**, а также создавать объект из его описания в файле – **Wscript.GetObject (<Pathname>, <ProgID>, <Prefix>)**.

Свойства данного объекта позволяют узнать параметры запущенной программы, кроме того свойством **Wscript.Timeout** можно задать максимальную продолжительность работы программы (в миллисекундах), а заданием значения свойства **WScript.Interactive = False** можно запретить вывод программой диалоговых окон.

Пример программы, сообщающей все свойства о самой себе (результат выполнения программы показан на рисунок 12.2):

```
With WScript
    .echo(.BuildVersion & vbLF & .FullName & vbLF & _
        .Interactive & vbLF & .Name & vbLF & _
        .Path & vbLF & .ScriptFullName & vbLF & _
        .ScriptName & vbLF & .Timeout & vbLF & _
        .Version)
End With
```

Пример программы, использующей метод WScript.Sleep, чтобы остановить работу программы на заданное количество миллисекунд (результат выполнения программы показан на рисунок 12.3):

```
t1 = time
WScript.Sleep 25545
t2 = time
h1 = Hour(t1): m1 = Minute(t1) : s1 = Second(t1)
h2 = Hour(t2): m2 = Minute(t2) : s2 = Second(t2)
ds = (h2-h1)*3600 + (m2-m1)*60 + (s2-s1)
MsgBox("Метод WScript.Sleep 25545" & vbLF & vbLF & _
    "Тнач = " & t1 & " Ткон = " & t2 & vbLF & vbLF & _
    "Программа проспала " & ds & " секунд"), _
    vbExclamation, "Использование метода WScript.Sleep"
```

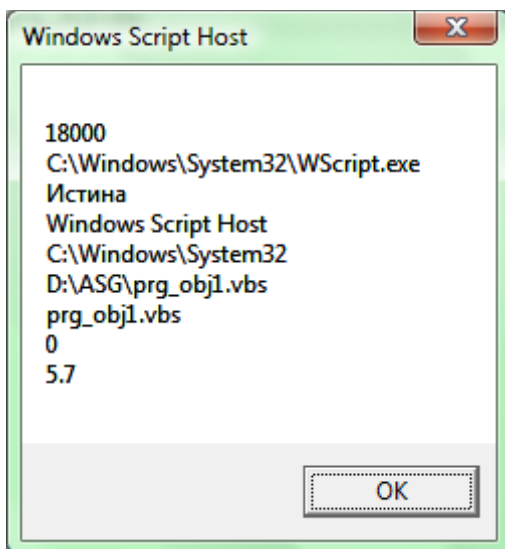


Рисунок 12.2. Свойства объекта **WScript**

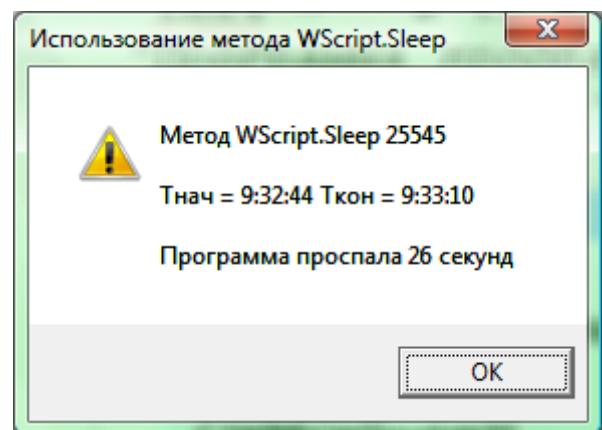


Рисунок 12.3. Использование метода **WScript.Sleep**

Использование метода **Popup** объекта **WScript.Shell** было рассмотрено ранее в лабораторной работе №1. Аналогично следует использовать методы *AppActivate*, *Exec*, *Run* и пр.

### **Задания**

- 1) С использованием метода **Exec** объекта **WScript.Shell** запустите из своей программы **Калькулятор Windows**.
- 2) С использованием метода **Exec** объекта **WScript.Shell** запустите из своей программы **Блокнот Windows**.
- 3) С использованием метода **Exec** объекта **WScript.Shell** запустите из своей программы **Explorer Windows**.
- 4) С использованием метода **Exec** объекта **WScript.Shell** запустите из своей программы **Internet Explorer Windows**.
- 5) С использованием метода **Exec** объекта **WScript.Shell** запустите из своей программы приложение **Microsoft Office Word**.
- 6) С использованием метода **Run** объекта **WScript.Shell** запустите из своей программы **Блокнот Windows** в свернутом окне
- 7) С использованием метода **Run** объекта **WScript.Shell** запустите из своей программы **Wordpad Windows** в окне, развернутом на весь экран.
- 8) Напишите программу, которая будет запускать **Блокнот Windows** и покажет после закрытия **Блокнота** сообщение о продолжительности его работы в минутах и секундах.
- 9) Напишите программу, которая с использованием метода **AppActivate** (<Имя окна приложения>) объекта **WScript.Shell** определит, запущена или нет в системе программа **Блокнот**
- 10) Напишите программу, которая с использованием метода **AppActivate** (<Имя окна приложения>) объекта **WScript.Shell** определит, запущена или нет в системе программа **Калькулятор** и, если программа не запущена, запустит ее.

## Лабораторная работа № 13. Работа с информацией файловой системы

Система Microsoft Visual Basic Scripting Edition для создания объектов, работающих с файловой системой, использует библиотеку классов Windows с именем **Scripting** (Microsoft Scripting Runtime Library – файл ...\\windows\\system32\\Scriun.dll).

Для дальнейшего понимания назначения объектов, их свойств и методов необходимо знание английской компьютерной терминологии.

Список объектов, которые могут быть созданы при работе с этой библиотекой, показан на рисунке 5.31. Полное описание объектов можно найти, например, в файле

...\\Microsoft Office\\Office12\\1049\\Vbscrip5.chm.

Главный объект библиотеки классов **Scripting** –

**Scripting.FileSystemObject**, который

имеет вложенные объекты **Drive**, **Folder**, **File** и **TextStream**.

Методы объекта **FileSystemObject**: *BuildPath CopyFile CopyFolder CreateFolder CreateTextFile DeleteFile DeleteFolder DriveExists FileExists FolderExists GetAbsolutePathName GetBaseName GetDrive GetDriveName GetExtensionName GetFile GetFileName GetFolder GetParentFolderName GetSpecialFolder GetTempName MoveFile MoveFolder OpenTextFile VarType*. Объект имеет одно свойство – *Drives*, возвращающее ссылку на коллекцию объектов **Drive** – дисковых устройств в системе.

**Коллекция** – упорядоченное множество однотипных объектов.

Любая коллекция имеет свойство *Count* (количество объектов в коллекции).

Ссылки на коллекции возвращает также метод *GetFolder*: на коллекцию **SubFolders** (объекты **Folder**) и на коллекцию **Files** (объекты **File**).

Свойства объекта **Drive**: *AvailableSpace DriveLetter DriveType FileSystem FreeSpace IsReady Path RootFolder*

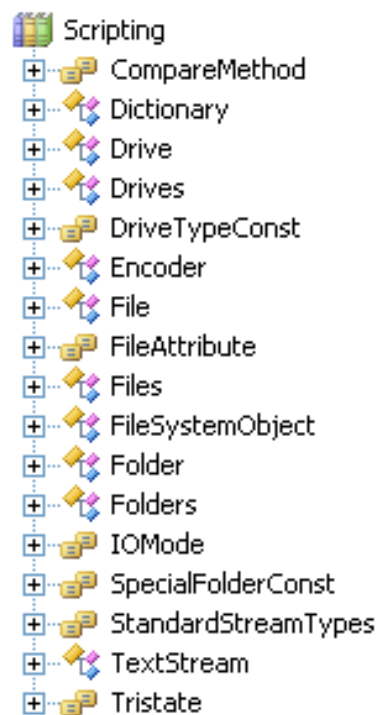


Рисунок 13.1. Библиотека классов **Scripting**



*SerialNumber ShareName TotalSize VolumeName.* Методов объект не имеет.

Методы объектов **Folder** и **File**: *Copy Delete Move OpenAsTextStream.*

Свойства объектов **Folder** и **File**: *Attributes DateCreated DateLastAccessed DateLastModified Drive Name ParentFolder Path ShortName ShortPath Size Type.*

Методы объекта **TextStream**: *Close Read ReadAll ReadLine Skip SkipLine Write WriteLine WriteBlankLines.*

Свойства объекта **TextStream**: *AtEndOfLine AtEndOfStream Column Line.*

В библиотеке классов **Scripting** присутствует объект *Dictionary* (словарь), который предназначен для сохранения множества парных значений, первый элемент каждой пары – ключевое (уникальное) значение. Этот объект имеет свойства: *CompareMode, Count, Item, Key* и методы: *Add, Exists, Items, Keys, Remove, RemoveAll.*

Некоторые примеры использования объектов приведены ниже.

**Пример 1.** Информация о дисках компьютера (результат исполнения – рисунок 13.2).

```
On Error Resume Next 'на случай, если в CD-приводе нет диска
Dim fs, dr, ndr, d, dl, drv, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set dr = fs.Drives
ndr = dr.Count
d = "Всего дисков - " & ndr & ": "
For Each dl in dr
    name_d = dl.DriveLetter & ":"
    d = d & name_d & " "
    Set drv = fs.GetDrive(fs.GetDriveName(name_d))
    s = s & vbCrLf & "Имя диска " & drv.Path & " - "
    s = s & drv.VolumeName
    Select Case drv.DriveType
        Case 0: t = "Неизвестный"
        Case 1: t = "Удаляемый (флэш и пр.) "
        Case 2: t = "Раздел винчестера"
        Case 3: t = "Сетевой"
        Case 4: t = "CD-ROM"
        Case 5: t = "RAM диск"
    End Select
    s = s & ", тип " & t & vbCrLf
    s = s & "Серийный номер диска " & _
        drv.SerialNumber & vbCrLf
    s = s & "Всего: " & _
        FormatNumber(drv.TotalSize/1024/1024, 0)
    s = s & " МБ,"
```



```

s = s & " свободно: " & _
    FormatNumber(drv.FreeSpace/1024/1024, 0)
s = s & " Мб" & vbCrLf
Next
s = d & vbCrLf & s
MsgBox s,, "Это все FileSystemObject!"

```

**Пример 2.** Информация о папках (результат исполнения – рисунок 13.3).

```

Dim fso, f, fl, fc, s
Set fso = CreateObject("Scripting.FileSystemObject")
s = "C:\inetpub"
Set f = fso.GetFolder(s)
Set fc = f.SubFolders           'коллекция объектов SubFolders
s = s & ", папки: " & vbCrLf
For Each fl in fc
    s = s & fl.name & vbCrLf   'папки в коллекции SubFolders
Next
MsgBox s,, "Коллекция SubFolders"

```

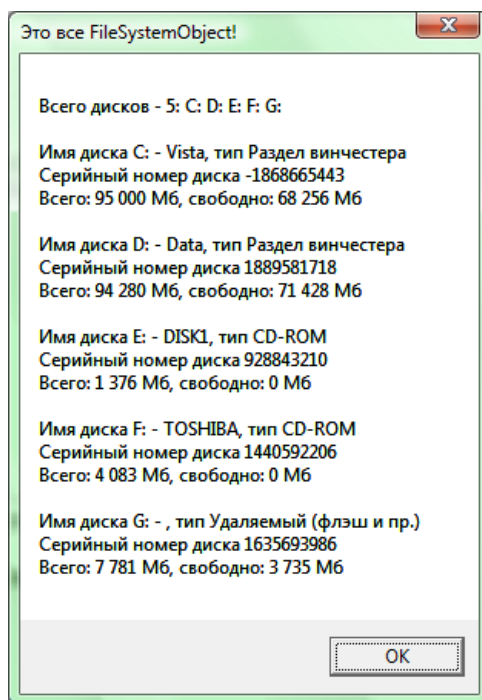


Рисунок 13.2. Пример 1

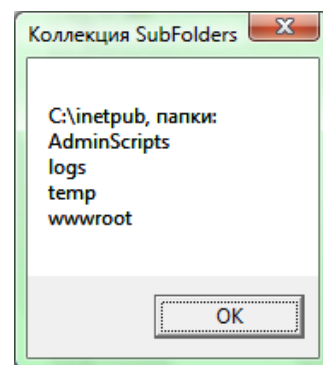


Рисунок 13.3. Пример 2

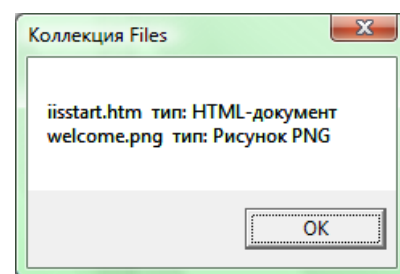


Рисунок 13.4. Пример 3

**Пример 3.** Информация о файлах и их типах в папке (результат исполнения – рисунок 13.4).

```

Dim fso, f, fl, fc, s
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFolder("C:\inetpub\wwwroot")
Set fc = f.Files           'коллекция объектов Files
For Each fl in fc
    s = s & fl.name & " тип: " & fl.Type & vbCrLf

```

```
Next
MsgBox s,, "Коллекция Files"
```

**Пример 4.** Чтение информации о файле (результат исполнения – рисунок 13.5).

```
Dim fso, fld, F_name, f, s
Set fso = CreateObject("Scripting.FileSystemObject")
F_name = "D:\ASG\Мой учебник по " & _
        "информатике\VBScript\Чтение информации о файле.vbs"
Set f = fso.GetFile(F_name)      'Создан объект File
s = "Диск - " & f.Drive.DriveLetter & vbCrLf
s = s & "Путь - " & f.Path & vbCrLf
s = s & "Путь в формате 8.3 - " & f.ShortPath & vbCrLf
s = s & "Имя - " & f.Name & vbCrLf
s = s & "Короткое имя - " & f.ShortName & vbCrLf
s = s & "Размер - " & f.Size & " байт" & vbCrLf
s = s & "Тип файла - " & f.Type & vbCrLf
s = s & "Дата создания - " & f.DateCreated & vbCrLf
s = s & "Дата последнего доступа - " & _
        f.DateLastAccessed & vbCrLf
s = s & "Дата последней модификации - " & _
        f.DateLastModified
MsgBox s,, "Свойства файла"
```

**Пример 5.** Чтение строк из файла (результат исполнения – рисунок 13.6).

```
Dim fso, F_name, f_txt, s
Set fso = CreateObject("Scripting.FileSystemObject")
F_name = "D:\VBScript\Prg_obj5.vbs"
Set f_txt = fso.OpenTextFile(F_name, 1)
Do While f_txt.AtEndOfStream <> True
    s = s + f_txt.ReadLine + vbCrLf
Loop
f_txt.Close
MsgBox s,, "Чтение строк из файла"
```

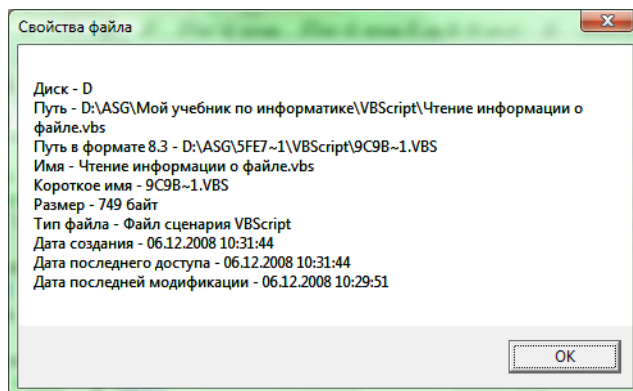


Рисунок 13.5. Пример 4

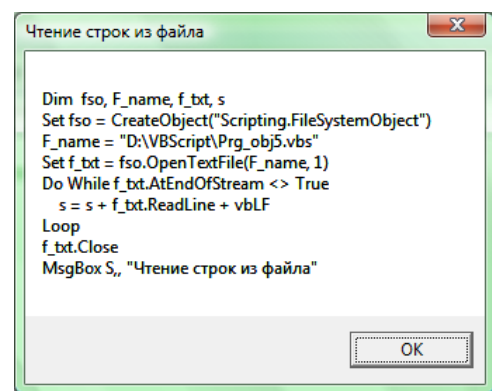


Рисунок 13.6. Пример 5

**Пример 6.** Создание папки, текстового файла и запись в файл:

```
Dim fso, fld, F_name, f_txt, ns, s
s = ""
Set fso = CreateObject("Scripting.FileSystemObject")
fld = "C:\vbscripts\"
If Not (fso.FolderExists(fld)) Then
    fso.CreateFolder(fld)
    s = "Создана папка " & fld & "!" & vbCrLf
End If
F_name = fld & "vbs_test1.txt"
If Not (fso.FileExists(F_name)) Then
    Set f_txt = fso.CreateTextFile(F_name, True)
    f_txt.WriteLine("1. Файл " & F_name & " создан!")
    s = s & "Создан файл " & F_name & "!" & vbCrLf
    s = s & " о чем записано в 1-ой строке файла!" & vbCrLf
End If
Set f_txt = fso.OpenTextFile(F_name, 1)
'Файл открыт для чтения, как объект TextStream
f_txt.ReadAll
ns = f_txt.Line 'Номер последней строки минус 1
Set f_txt = fso.OpenTextFile(F_name, 8)
'Открыт для дозаписи в конец файла
f_txt.WriteLine(ns & "-я строка текстового файла.")
f_txt.Close
s = s & "В конец файла " & F_name & vbCrLf
s = s & " добавлена " & ns & "-я строка"
s = s & vbCrLf & vbCrLf & "Всего строк в файле - " & ns
MsgBox s,, "Добавление строк в файл"
```

После запуска на исполнение этой программы первый раз (при отсутствии папки и файла) будет показано сообщение, приведенное на рисунке 13.7. После запуска на исполнение программы 6 три раза файл **vbs\_test1.txt** будет содержать текст:

1. Файл C:\vbscripts\vbs\_test1.txt создан!
- 2-я строка текстового файла.
- 3-я строка текстового файла.
- 4-я строка текстового файла.

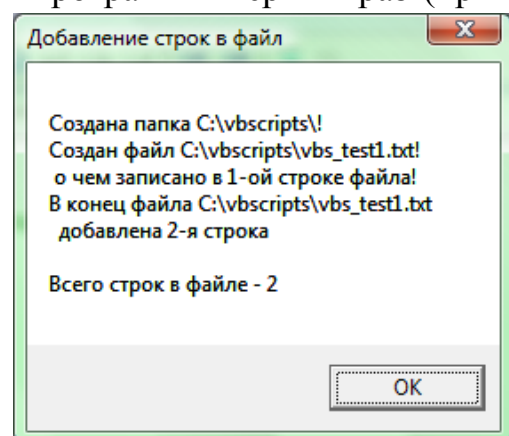


Рисунок 13.7. Пример 6

**Пример 7.** Генерация случайного массива из 10 млн. чисел (или больше...), запись чисел в текстовый файл, создаваемый программой, и открытие этого файла в Блокноте Windows:

```
Set FSO = CreateObject("Scripting.FileSystemObject")
Set Txt = FSO.CreateTextFile("D:\Test.txt")
n=10000000 : a=-10 : b=10
ReDim x(n)
Randomize
t1=Time : L = "" : xsr=0
For i=1 To n
    x(i) = b - (b-a)*Rnd
    xsr = xsr + x(i)
    L = L & FormatNumber(x(i),2) & vbTab
    If i/10 = i\10 Then Txt.WriteLine(L) : L = ""
Next
xmin=x(1) : xmax=x(1) : xsr = xsr/n
For i=2 To n
    If x(i)<xmin Then xmin=x(i)
    If x(i)>xmax Then xmax=x(i)
Next
t2=Time
Txt.Write(" Минимальное X = " & xmin & vbNewLine)
Txt.Write(" Максимальное X = " & xmax & vbNewLine)
Txt.Write(" Среднее X = " & xsr & vbNewLine)
Txt.Write(" Время работы программы = " &
FormatDateTime(t2-t1))
Txt.Close
Set rf = CreateObject("Wscript.Shell")
rf.Run("notepad D:\Test.txt")
```

В текстовый файл NTFS теоретически можно записать несколько Тбайт информации, в то время, как в окне сообщений **MsgBox** длина строки сообщений ограничена 256 символами, поэтому в нем более 50 чисел показать не удастся.

В программах на **VBScript** можно использовать вывод результатов работы программы в текстовый файл вместо окна **MsgBox**.

Конец файла Test.txt, открытого в Блокноте (последняя строка имеет номер 1000000, в строке по 10 чисел), показан на рисунке 13.8.

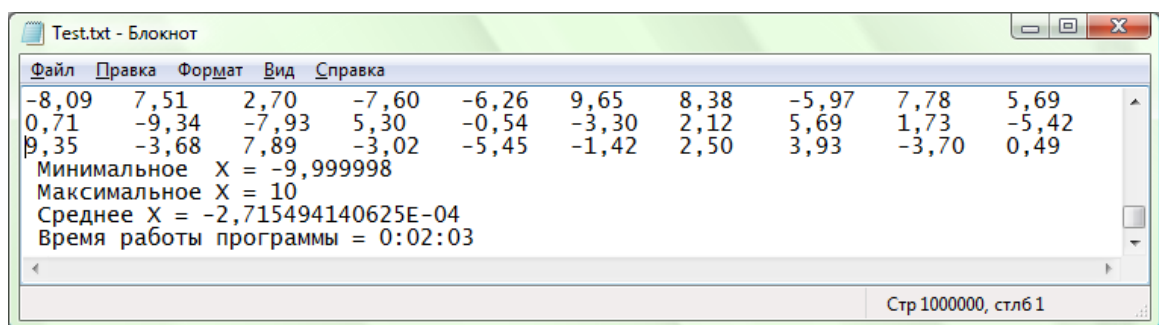


Рисунок 13.8. 10 млн. чисел, записанных программой в файл

### Пример 8. Копирование файла:

```
Const OverwriteExisting = TRUE
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CopyFile "C:\vbscripts\vbs_test1.txt", _
    "C:\vbscripts\vbs_test2.txt ", OverwriteExisting
```

Результатом работы программы будет новый файл `vbs_test2.txt`. Если запустить программу повторно, старый файл `vbs_test2.txt` будет заменен новым (т. к. параметр `OverwriteExisting = TRUE`, если задать `OverwriteExisting = FALSE` появится сообщение об ошибке «File already exist» – «Файл уже существует»).

**Пример 9.** Использование объекта *Dictionary* для чтения атрибутов файла (результаты работы – см. рисунок 13.9):

```
Set FSO = CreateObject("Scripting.FileSystemObject")
Set File = FSO.GetFile("C:\bootmgr")
Attrs = File.Attributes
Set Dict = CreateObject("Scripting.Dictionary")
Dict.Add "Обычный", "Нет"
Dict.Add "Только чтение", "Нет"
Dict.Add "Скрытый", "Нет"
    Dict.Add "Системный", "Нет"
Dict.Add "Том диска", "Нет"
Dict.Add "Папка", "Нет"
Dict.Add "Архивный", "Нет"
Dict.Add "Ярлык", "Нет"
Dict.Add "Сжатый", "Нет"
If Attrs=0 Then
    Dict.Item("Обычный") = "Да"
End If
If Attrs>=2048 Then
    Dict.Item("Сжатый") = "Да"
    Attrs = Attrs-2048
End If
If Attrs>=1024 Then
    Dict.Item("Ярлык") = "Да"
    Attrs = Attrs-1024
End If
If Attrs>=32 Then
    Dict.Item("Архивный") = "Да"
    Attrs = Attrs-32
End If
If Attrs>=16 Then
    Dict.Item("Папка") = "Да"
    Attrs = Attrs-16
End If
If Attrs>=8 Then
```

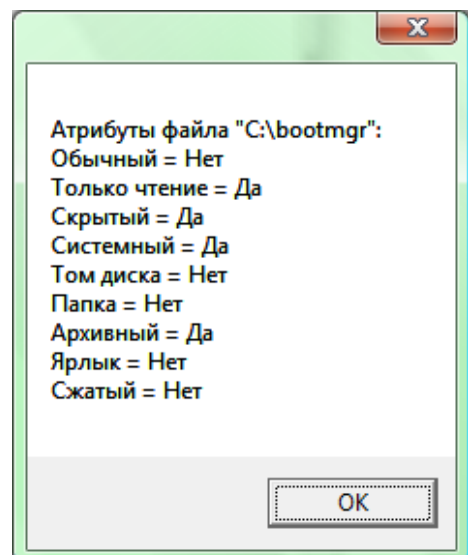


Рисунок 13.9. Пример 8

```

    Dict.Item("Том диска") = "Да"
    Attrs = Attrs-8
End If
If Attrs>=4 Then
    Dict.Item("Системный") = "Да"
    Attrs = Attrs-4
End If
If Attrs>=2 Then
    Dict.Item("Скрытый") = "Да"
    Attrs = Attrs-2
End If
If Attrs>=1 Then
    Dict.Item("Только чтение") = "Да"
    Attrs = Attrs-1
End If
Str = "Атрибуты файла " & File.Path & "": " & vbCrLf
For Each Attr In Dict
    Str = Str & Attr & " = " & Dict.Item(Attr) &
vbCrLf
Next
MsgBox Str

```

### **Задания**

- 1) Напишите программу, которая создаст в Вашей папке новый текстовый файл и пронумерует в нем строки с 1 по 50, затем создаст копию этого файла (с другим именем) в той же папке.
- 2) Напишите программу, которая прочитает текст из 5-ти файлов Ваших VBS-программ и запишет весь этот текст в новый файл. В новом файле перед текстом каждой программы написать название файла (перед названием вставить пустую строку). Откройте этот файл методом **Run** в программе по умолчанию для текстовых файлов.
- 3) Напишите программу, которая прочитает информацию о файлах Ваших программ в Вашей папке: имя файла, имя и путь, имя и путь в формате 8.3, дата создания, дата последнего изменения, размер файла и запишет эту информацию в новый текстовый файл и откроет его методом **Run** в программе по умолчанию для текстовых файлов.
- 4) Напишите программу, которая прочитает информацию о дисковых устройствах на Вашем компьютере: общее количество, имя, тип, общий объем и свободный объем каждого диска, запишет эту информацию в новый текстовый файл и откроет его методом **Run** в программе по умолчанию для текстовых файлов.
- 5) Напишите программу, которая откроет файл *Vbscrip5.chm* или анало-

гичный в предыдущих версиях Office. Предварительно найдите этот файл, скопируйте путь к нему и название в программу. Если файл не открывается методом **Exec** объекта **WScript.Shell**, попробуйте сделать это методом **Run**. Если файл не открывается, воспользуйтесь свойством **ShortPath** объекта **File** библиотеки **Scripting.FileSystemObject** для преобразования имени и пути в формат 8.3. Покажите в окне InputBox преобразованное имя и путь. Все варианты задания сохранить, сообщения системы привести в отчете.

- 6) Аналогично заданию 5 напишите программу открытия файла графического формата (\*.bmp, \*.jpg, \*.png или пр.) из папки с именем «Мои новые рисунки» в папке документов своего профиля пользователя.
- 7) Напишите программу, которая создаст новую папку «Мои новые файлы» в папке документов Вашего профиля пользователя, в новой папке создаст текстовый файл с помощью метода **CreateTextFile** библиотеки **Scripting.FileSystemObject** и откроет его методом **Run** в программе по умолчанию для текстовых файлов.
- 8) Напишите программу, которая создаст текстовый файл в папке документов Вашего профиля пользователя и запишет в него информацию о той папке, в которой находится данный файл (имя папки, имя в формате 8.3, дату создания, путь к папке, короткий путь в формате 8.3, размер папки и ее атрибуты).
- 9) Напишите программу, которая создаст текстовый файл в папке документов Вашего профиля пользователя и запишет в него информацию о всех папках в папке **C:\** на Вашем ПК и об их атрибутах.
- 10) Напишите программу, которая покажет в окне сообщений все файлы с расширением **exe** в папке **C:\Windows**.

## Лабораторная работа № 14. Работа с информацией об ошибках

Если написать в начале программы структуру

```
On Error Resume Next
```

автоматическое завершение работы программы при возникновении ошибки будет предотвращено и появится возможность выполнить некоторые дополнительные действия в программе с использованием объекта **Err** языка **VBScript**, который автоматически создается при работе программы и значение свойства *Number* которого в этом случае станет не равным нулю, что можно использовать для программной обработки ошибок.

Методы объекта **Err**: *Clear Raise*. Свойства объекта **Err**: *Description HelpContext HelpFile Number*.

Пример программы с обработкой ошибки (дополненный пример 7 предыдущей лабораторной работы):

```
On Error Resume Next  
Const OverwriteExisting = False  
Set fso = CreateObject("Scripting.FileSystemObject")  
fso.CopyFile "C:\vbscripts\vbs_test1.txt", _  
            "C:\vbscripts\vbs_test2.txt ", _  
            OverwriteExisting  
If Err.Number <> 0 Then      'файл уже существует  
    MsgBox "Файл уже существует и не может быть" & _  
        "заменен новым", vbCritical, "Сообщение об  
ошибке"  
End If
```

Пример использования для Примера 5 (см. выше):

```
On Error Resume Next  
Dim fso, F_name, f_txt, s  
Set fso = CreateObject("Scripting.FileSystemObject")  
F_name = "C:\vbscripts\vbs_test1.txt"  
Set f_txt = fso.OpenTextFile(F_name, 1)  
If Err.Number = 0 Then      'файл существует и открыт  
                            'далее – цикл чтения строк из файла  
    Do While f_txt.AtEndOfStream <> True  
        s = s + f_txt.ReadLine + vbLF  
    Loop  
    f_txt.Close  
    MsgBox S,, "Чтение строк из файла"  
Else  
    MsgBox ("Ошибка № " & CStr(Err.Number) & vbLf _  
        & Err.Description & vbLf & Err.Source) _  
    Err.Clear      'очистка свойств объекта Err
```



***End If***

Впрочем, данная ошибка может быть выявлена и другим способом: ... ***If IsObject(f\_txt) Then 'объект создан ...***

### ***Задание***

Программу, написанную Вами ранее в лабораторной работе 12 или 13, в которой выполняется открытие существующего файла, дополните фрагментом обработки ошибки открытия в случае неверного указания пути или имени файла с выводом сообщения об этом.

## Лабораторная работа № 15. Работа с информацией локальной сети

Для этой цели может использоваться объект **WScript.Network**, имеющий следующие свойства: **ComputerName** **UserName** **UserDomain** и методы: **EnumNetworkDrives** **EnumPrinterConnections** **MapNetworkDrive** **RemoveNetworkDrive** **AddPrinterConnection** **AddWindowsPrinterConnection** **RemovePrinterConnection** **SetDefaultPrinter**.

Пример использования объекта **WScript.Network**:

```
On Error Resume Next
Set Wnet = WScript.CreateObject("WScript.Network")
S = "Домен = " & Wnet.UserDomain & vbCrLf
S = S & "Имя компьютера = " & Wnet.ComputerName & vbCrLf
S = S & "Пользователь = " & Wnet.UserName
Net_f = "\\lks\asg"
Wnet.MapNetworkDrive "I:", Net_f, "True"
If Err.Number <> 0 Then 'сетевая папка не найдена
    MsgBox S & vbCrLf & "Сетевая папка " & Net_f & _
        " не найдена", vbCritical, "Ошибка..."
End If
Set NetDrives = Wnet.EnumNetworkDrives
If not isObject(NetDrives) Then
    MsgBox "Нет сетевых дисков", vbExclamation, _
        "Нет данных..."
Else
    i = 0
    S = S & vbCrLf & "Количество сетевых дисков = " & _
        NetDrives.Count/2 & vbCrLf
    While i <= NetDrives.Count-1
        S = S & NetDrives.Item(i) & " - " & _
            NetDrives.Item(i+1) & vbCrLf
        i = i+2
    Wend
    MsgBox S, vbInformation, _
        "Информация о компьютере"
End If
```

Пример работы программы показан на рисунке 15.1.

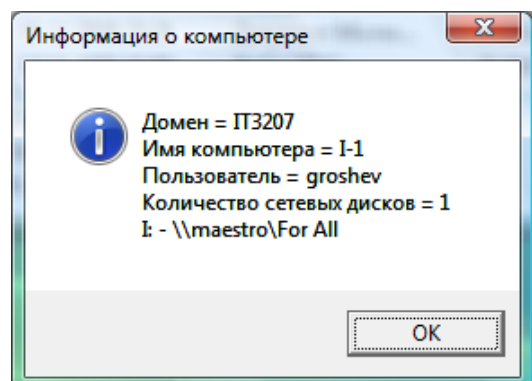


Рисунок 15.1. Пример работы с объектом **WScript.Network**

### Задание

Напишите программу, которая определит имя Вашего компьютера, имя пользователя и покажет список его логических дисков.

## Лабораторная работа № 16. Работа с объектами Windows OLE Automation (Microsoft ActiveX)

Данная возможность используется в целях администрирования серверов, компьютеров и их сетей, имеет неограниченные возможности, и здесь подробно рассматриваться не будет. Большое количество примеров использования можно найти на сайте корпорации Microsoft (<http://www.microsoft.com/technet/scriptcenter/>).

Приведем лишь примеры использования для объектов системы Microsoft Office.

Пример 1. Программа открывает существующий файл системы Microsoft Office Word (c:\vbscripts\test\_prg.doc) и добавляет в него новые строки текста:

```
Const END_OF_STORY = 6
Const MOVE_SELECTION = 0
Set objWord = CreateObject("Word.Application")
objWord.Visible = True
Set objDoc = _
    objWord.Documents.Open("c:\vbscripts\test_prg.doc")
Set objSelection = objWord.Selection
objSelection.EndKey END_OF_STORY, MOVE_SELECTION
objSelection.TypeParagraph()
objSelection.Font.Size = "14"
objSelection.TypeText "Это первая строка, которую" & _
    " VBScript добавила в документ"
objSelection.TypeParagraph()
objSelection.TypeText "Это еще одна строка, " & _
    " которую VBScript добавила в документ"
objSelection.TypeParagraph()
objSelection.TypeText "Сегодняшняя дата " & Date()
objSelection.TypeParagraph()
```

Пример работы программы показан на рисунке 16.1.

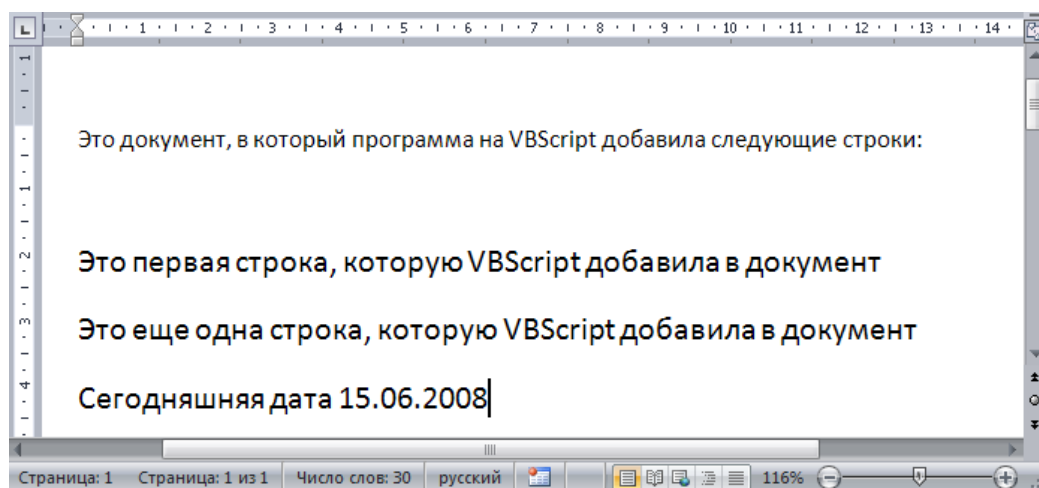
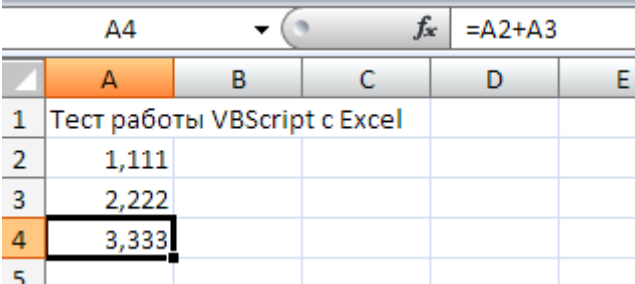


Рисунок 16.1. Пример работы с объектом *Word.Application*

Пример 2. Программа открывает новый файл системы Microsoft Office Excel и добавляет в его ячейки текст, числа и расчетную формулу:

```
Set objExcel = CreateObject("Excel.Application")
With objExcel
    .Visible = True
    .Workbooks.Add
    .Cells(1, 1).Value = "Тест работы VBScript с Excel"
    .Cells(2, 1).Value = 1.111
    .Cells(3, 1).Value = 2.222
    .Cells(4, 1).Value = "=A2+A3"
End With
```

Результат работы программы показан на рисунке 16.2.



|   | A                            | B | C | D | E |
|---|------------------------------|---|---|---|---|
| 1 | Тест работы VBScript с Excel |   |   |   |   |
| 2 | 1,111                        |   |   |   |   |
| 3 | 2,222                        |   |   |   |   |
| 4 | 3,333                        |   |   |   |   |
| 5 |                              |   |   |   |   |

Рисунок 16.2. Пример работы с объектом *Excel.Application*

Пример 3. Программа устанавливает связь с базой данных системы Microsoft Office Access (используется база данных, описанная в [1], раздел 3.3.4), выполняет запрос на отбор данных и показывает их в окне, приведенном на рисунке 16.3:

```
Dim Conn, rs, S
Set Conn = CreateObject("ADODB.Connection")
'1 вариант – использование провайдера программного ядра базы Access:
Conn.Open "PROVIDER=Microsoft.Jet.OLEDB.4.0;" & _
    "DATA SOURCE=D:\ASG\Контингент.mdb"
'2 вариант – использование провайдера Access Connectivity Engine
' Conn.Open "PROVIDER=Microsoft.ACE.OLEDB.12.0;" & _
    "DATA SOURCE=C:\AGTU\Контингент.accdb"
'3 вариант – использование источника данных' пользователя,
' описанного заранее в Администраторе источников
' данных Windows (ODBC)
'Conn.Open "DSN=Контингент"
Set rs = CreateObject("ADODB.Recordset")
Set rs = Conn.Execute("SELECT NZ, FIO FROM Список")
S=" № зач.      фамилия, имя, отчество" & vbCrLf
S = S & "-----" & vbCrLf
Do Until rs.EOF
    S = S & rs("NZ") & " " & rs("fio") & vbCrLf
    rs.MoveNext
Loop
```

```

msgbox S,,"Данные таблицы Список"
If rs.State = adStateOpen then
    rs.Close
End If
If Conn.State = adStateOpen then
    Conn.Close
End If

```

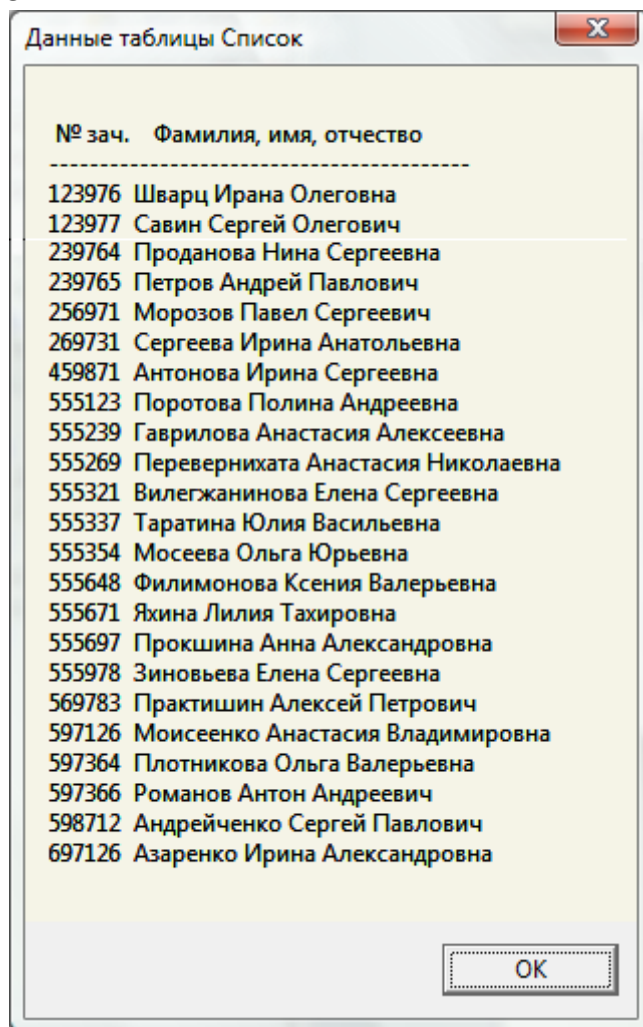


Рисунок 16.3. Пример работы с объектами *ADODB.Connection* и *ADODB.Recordset*

### Задание

Напишите программу, которая выполнит следующие действия:

- откроет любой Ваш документ Microsoft Office Word;
- откроет любой Ваш документ Microsoft Office Excel;
- прочитает записи одной из таблиц вашей базы данных Access и запишет их в текстовый файл (каждая запись – отдельная строка).

## Лабораторная работа № 17. Использование скриптов на HTML-страницах

В разделе 4.12 учебника [1] приведены некоторые сведения о языке гипертекстовой разметки HTML, который позволяет создавать сложные структурированные красиво оформленные web-страницы. Указывалось также, что на этих страницах могут использоваться программные компоненты – скрипты на языках сценариев, таких, как JavaScript, *VBScript* и TCL для создания динамических страниц и использования HTML как средства создания сетевых приложений.

Сценарий на странице должен располагаться внутри пары тэгов **<SCRIPT> и </SCRIPT>**. По умолчанию языком сценария считается JavaScript, при использовании языка *VBScript* следует использовать указание в тэге **<SCRIPT LANGUAGE="VBSCRIPT">**.

Любая программа на языке приведенная выше, может исполняться и вызовом ее через Интернет-браузер, если файл программы сохранять не с расширением \*.vbs, а как \*.htm или \*.html.

Простейший пример скрипта, который будет выполнен программой Internet Explorer, если его написать в Блокноте и сохранить с именем *My\_script1.html*:

```
<SCRIPT LANGUAGE="VBScript">  
  MsgBox ("Привет, WWW!")  
</SCRIPT>
```

Открыв файл *My\_script1.html* двойным щелчком мыши (можно также открыть его, как и файлы \*.vbs, из текстовых редакторов *EmEditor* и *Aditor*), мы увидим пустую web-страницу и созданную ей Windows-форму, показанную на рисунке 17.1. При этом необходимо разрешить исполнение скриптов на web-странице.

Сценарий может быть расположен в любом месте HTML-документа, но обычно он располагается в начале или в конце документа.

Пример HTML-страницы с математическими вычислениями на ней, выполняемыми скриптом (результат работы программы показан на рисунке 17.2):

```
<HTML>  
  <HEAD>  
    <TITLE>Работа с VBScript</TITLE>  
  </HEAD>  
  <BODY>  
    <H3>Пример использования VBScript на HTML-  
    странице</H3>
```

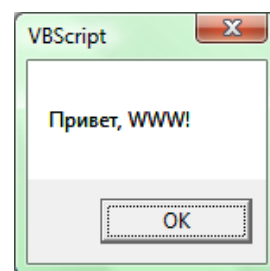


Рисунок 17.1.  
Простейший  
скрипт на  
web-странице

```

<P>Задайте количество и цену
и нажмите кнопку "Расчет стоимости" </P>
<FORM NAME="frm">
  <TABLE>
    <TR>
      <TD> Количество</TD>
      <TD> Цена</TD>
      <TD> Стоимость</TD>
    </TR>
    <TR>
      <TD><INPUT TYPE="Text" NAME="Kolic"
SIZE=12></TD>
      <TD><INPUT TYPE="Text" NAME="cena"
SIZE=12></TD>
      <TD><INPUT TYPE="Text" NAME="stoimost"
SIZE=12></TD>
    </TR>
  </TABLE>
  <BR>
  <INPUT TYPE="Button" NAME="Calc"
VALUE="Расчет стоимости">
</FORM>
</BODY>
<SCRIPT LANGUAGE="VBScript">
  Sub Calc_OnClick()
    document.frm.stoimost.value = _
    Round(document.frm.Kolic.value
    * Document.frm.cena.value,2)
  End Sub
</SCRIPT>
</HTML>

```

### Задание

Модифицируйте программу Вашей лабораторной работы № 4 с размещением VBScript-программы на web-странице. На форма должны быть поля для X, Y и кнопка «Расчет».

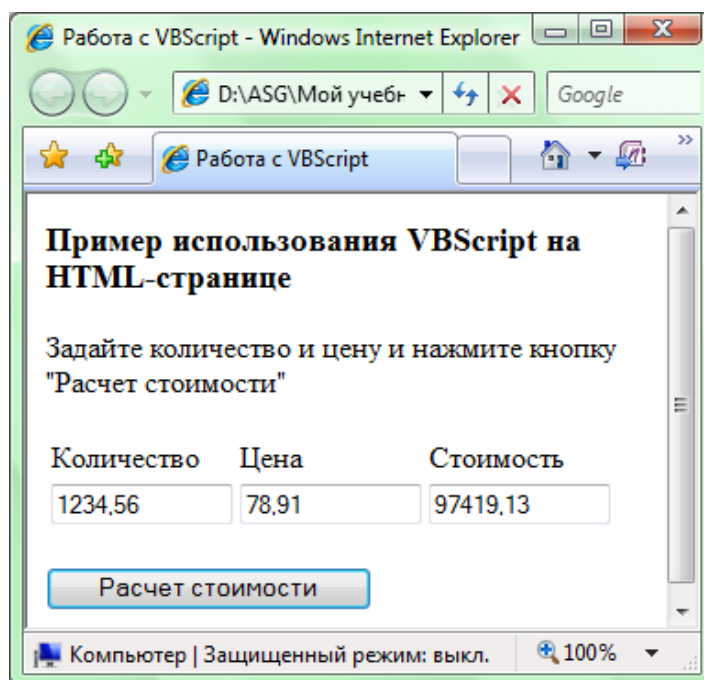


Рисунок 17.2. Использование **VBScript** на web-странице

## Приложение 1. Математические функции языка

| Синтаксис функции                                                                                   | Описание                                                                                                                                                                                                                               |
|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Abs</b> (числ.выраж.)                                                                            | Абсолютная величина числового выражения                                                                                                                                                                                                |
| <b>Array</b> (сп.эл.массива)                                                                        | Создание массива из списка                                                                                                                                                                                                             |
| <b>Atn</b> (числ.выраж.)                                                                            | Арктангенс, числовое выражение в радианах                                                                                                                                                                                              |
| <b>Cbool</b> (числ.выраж.)                                                                          | Возвращает True для ненулевого числового выражения, False для 0, ошибка для нечислового значения                                                                                                                                       |
| <b>CByte</b> (числ.выраж.)                                                                          | Преобразование в субтип <b>Byte</b>                                                                                                                                                                                                    |
| <b>CCur</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Currency</b>                                                                                                                                                                                                |
| <b>CDbl</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Double</b>                                                                                                                                                                                                  |
| <b>CInt</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Integer</b>                                                                                                                                                                                                 |
| <b>CLng</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Long</b>                                                                                                                                                                                                    |
| <b>Cos</b> (числ.выраж.)                                                                            | Косинус, числовое выражение в радианах                                                                                                                                                                                                 |
| <b>CSng</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Single</b>                                                                                                                                                                                                  |
| <b>CStr</b> (числ.выраж.)                                                                           | Преобразование в субтип <b>Строка</b>                                                                                                                                                                                                  |
| <b>Exp</b> (числ.выраж.)                                                                            | Функция экспонента $\text{Exp}(X)$ или $e^x$                                                                                                                                                                                           |
| <b>Eval</b> (выражение)                                                                             | Вычисляет выражение и возвращает его результат                                                                                                                                                                                         |
| <b>Fix</b> (числ.выраж.)                                                                            | Целая часть числа                                                                                                                                                                                                                      |
| <b>Int</b> (числ.выраж.)                                                                            | Целая часть числа                                                                                                                                                                                                                      |
| <b>FormatCurrency</b> (числ.выраж. [, дес.зн. [, вед.ноль [, исп.() для отр.чисел [, исп.разд.]]]]) | Форматирование числ.выраж. в заданный денежный формат с указанием денежной единицы, заданной в Windows                                                                                                                                 |
| <b>FormatNumber</b> (числ.выраж. [, дес.зн. [, вед.ноль [, исп.() для отр.чисел [, исп.разд.]]]])   | Форматирование числ.выраж. в заданный формат                                                                                                                                                                                           |
| <b>FormatPercent</b> (числ.выраж. [, дес.зн. [, вед.ноль [, исп.() для отр.чисел [, исп.разд.]]]])  | Форматирование числ.выраж. в заданный процентный формат (/100) с символом %                                                                                                                                                            |
| <b>Hex</b> (числ.выраж.)                                                                            | Шестнадцатеричное число десятичного числового выражения                                                                                                                                                                                |
| (числ.выраж.) \ (числ.выраж.)                                                                       | Деление двух числовых выражения, результатом которого является целое число (целочисленное деление). До деления результаты округляются до целого числа (<0,5 равно 0, >=0.5 равно 1), у результата операции отбрасывается дробная часть |
| <b>Lbound</b> (имя_массива[, размерность])                                                          | Наименьшее возможное значение индекса в массиве.. Если размерность массива не указан, он одномерный                                                                                                                                    |
| <b>Log</b> (числ.выраж.)                                                                            | Натуральный логарифм выражения. Если нужно вычислить логарифм с основанием <b>n</b> , следует использовать выражение: $\text{Logn}(x) = \text{Log}(x) / \text{Log}(n)$                                                                 |
| (числ.выраж.) <b>Mod</b> (числ.выраж.)                                                              | Остаток целочисленного деления. Перед делением результаты численных выражений                                                                                                                                                          |



|                                            |                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                            | округляются до целых чисел                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Oct</b> (число)                         | Строка, представляющая восьмеричную величину числа                                                                                                                                                                                                                                                                                                                                          |
| <b>Randomize</b>                           | Инициализация генератора случайных чисел                                                                                                                                                                                                                                                                                                                                                    |
| <b>Rnd</b> [(числ.выраж.)]                 | Случайное число большее или равное нулю, но меньшее 1.                                                                                                                                                                                                                                                                                                                                      |
| <b>RGB</b> (red, green, blue)              | Преобразование цветового значения, где каждый из трех цветов в диапазоне от 0 до 255 в одно число, рассчитанное по формуле:<br><b>CLng</b> (red + (green * 256) + (blue * 65536) )                                                                                                                                                                                                          |
| <b>Round</b> (числ.выраж. [, дес.зн.])     | Число, округленное до заданного количества десятичных знаков. Если дес.зн. не указано, его значение равно 0                                                                                                                                                                                                                                                                                 |
| <b>Sgn</b> (числ.выраж.)                   | Знак числа (>0 – 1, 0 – 0, <0 – -1)                                                                                                                                                                                                                                                                                                                                                         |
| <b>Sin</b> (числ.выраж.)                   | Синус угла, выраженного в радианах                                                                                                                                                                                                                                                                                                                                                          |
| <b>Sqr</b> (числ.выраж.)                   | Квадратный корень числа                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Tan</b> (числ.выраж.)                   | Тангенс угла, выраженного в радианах                                                                                                                                                                                                                                                                                                                                                        |
| <b>TypeName</b> (varИмя)                   | Название типа переменной (см. далее для <b>VarType</b> )                                                                                                                                                                                                                                                                                                                                    |
| <b>Ubound</b> (имя_массива[, размерность]) | Наибольшее возможное значение индекса в массиве. Если размерность массива не указан, он одномерный                                                                                                                                                                                                                                                                                          |
| <b>VarType</b> (varИмя)                    | Числовое обозначение типа переменной:<br>0 <b>vbEmpty</b><br>1 <b>vbNull</b><br>2 <b>vbInteger</b><br>3 <b>vbLong</b><br>4 <b>vbSingle</b><br>5 <b>vbDouble</b><br>6 <b>vbCurrency</b><br>7 <b>vbDate</b><br>8 <b>vbСтрока</b><br>9 <b>vbObject</b><br>10 <b>vbError</b><br>11 <b>vbBoolean</b><br>12 <b>vbVariant</b><br>13 <b>vbDataObject</b><br>17 <b>vbByte</b><br>8192 <b>vbArray</b> |

## Приложение 2. Строковые функции языка

| Синтаксис функции                                                               | Описание                                                                                                                                                                                             |
|---------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Asc</b> (строка)                                                             | ANSI-код первого символа в строке                                                                                                                                                                    |
| <b>Chr</b> (число)                                                              | Символ заданного ANSI-кода                                                                                                                                                                           |
| <b>Eval</b> (выражение)                                                         | Вычисляет выражение и возвращает его результат                                                                                                                                                       |
| <b>InStr</b> ([нач.поз., ]строка1, строка2[, тип сравн.])                       | Позиция строки2 в строке1 начиная с нач.поз. поиска для заданного типа сравнения (vbBinaryCompare или vbTextCompare, если не указано, то первый)                                                     |
| <b>InStrRev</b> (строка1, строка2[, нач.[, тип сравн.]])                        | То же, что и <b>InStr</b> , но номер позиции с конца строки                                                                                                                                          |
| <b>Join</b> (имя_массива[, разделитель])                                        | Строка, созданная из элементов массива                                                                                                                                                               |
| <b>LTrim</b> (строка), <b>RTrim</b> (строка), <b>Trim</b> (строка)              | Строка без пробелов слева ( <b>LTrim</b> ), справа ( <b>RTrim</b> ), или без тех и других ( <b>Trim</b> ).                                                                                           |
| <b>LCase</b> (строка)                                                           | Преобразует все символы строки в строчные                                                                                                                                                            |
| <b>Left</b> (строка, длина)                                                     | Возвращает заданное количество символов с начала строки                                                                                                                                              |
| <b>Len</b> (строка   имя_пер.)                                                  | Число символов в строке или строковой переменной                                                                                                                                                     |
| <b>Mid</b> (строка, нач.[, длина])                                              | Возвращает заданное количество символов с заданной позиции <b>нач.</b> в строке                                                                                                                      |
| <b>Replace</b> (исх_стр, стр_поиска, стр_замены[, нач.[, колич.[, тип сравн.]]) | Замена в исходной строке строки поиска на строку замены, начиная с позиции <b>нач.</b> , заданное количество раз                                                                                     |
| <b>Right</b> (строка, длина)                                                    | Возвращает заданное количество символов с конца строки                                                                                                                                               |
| <b>Space</b> (количество)                                                       | Строка из заданного количества пробелов                                                                                                                                                              |
| <b>Split</b> (исх_стр.[, разделитель[, количество[, тип сравн.]])               | Возвращает одномерный массив строк, полученный разбиением <i>исх_стр.</i> по <i>разделителям</i> на заданное количество частей. Если <i>разделитель</i> не указан, за него принимается знак пробела. |
| <b>StrComp</b> (строка1, строка2[,тип сравн.])                                  | Сравнение строк. Если строка1<строка2, возвращается -1, если строка1=строка2, возвращается 0, если строка1>строка2, возвращается 1.                                                                  |
| <b>String</b> (количество, символ)                                              | Создает строку из заданного количества заданных символов                                                                                                                                             |
| <b>StrReverse</b> (строка)                                                      | Переворачивает строку задом-наперед                                                                                                                                                                  |
| <b>UCase</b> (строка)                                                           | Преобразует все символы строки в прописные                                                                                                                                                           |

## Приложение 3. Функции работы с датой и временем

| Синтаксис функции                                                            | Описание                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CDate</b> (стр.выр.)                                                      | Преобразование строкового выражения в подтип Дата-время.                                                                                                                                                                                  |
| <b>DateAdd</b> (формат, количество, <исходное дата-время>)                   | Дата-время, к которому добавлен заданное количество времени для заданного формата:<br>"yyyy" Год<br>"q" Квартал<br>"m" Месяц<br>"y" День года<br>"d" День<br>"w" день недели<br>"ww" Week of year<br>"h" Час<br>"n" Минута<br>"s" Секунда |
| <b>DateDiff</b> (формат, дата1, дата2 [,первый_день_нед[, первая_нед_года]]) | Интервал между двумя временными интервалами в заданном формате                                                                                                                                                                            |
| <b>DatePart</b> (формат, дата [,первый_день_нед[, первая_нед_года]])         | Часть даты в заданном формате                                                                                                                                                                                                             |
| <b>DateSerial</b> (год, месяц, день)                                         | Дата для заданных года, месяца и дня                                                                                                                                                                                                      |
| <b>DateValue</b> ("дата-время")                                              | Подтип Дата-время для заданного строкового выражения                                                                                                                                                                                      |
| <b>Day</b> ("дата-время")                                                    | День месяца – целое число от 1 до 31                                                                                                                                                                                                      |
| <b>FormatDateTime</b> (дата[, формат)                                        | Форматирование даты в заданный формат: vbGeneralDate, vbLongDate, vbShortDate, vbLongTime, vbShortTime                                                                                                                                    |
| <b>Hour</b> (время)                                                          | Час дня, целое число от 0 до 23                                                                                                                                                                                                           |
| <b>Minute</b> (время)                                                        | Минуты часа, целое число от 0 до 59                                                                                                                                                                                                       |
| <b>Month</b> (дата)                                                          | День месяца, целое число от 1 до 12                                                                                                                                                                                                       |
| <b>MonthName</b> (месяц[, сокр_назв])                                        | Строковое обозначение месяца, сокращенное название (True) или полное (False). Если не указано, False – полное название.                                                                                                                   |
| <b>Now</b>                                                                   | Текущая дата и время системных часов                                                                                                                                                                                                      |
| <b>Second</b> (время)                                                        | Секунды в минуте, целое число от 0 до 59                                                                                                                                                                                                  |
| <b>Time</b>                                                                  | Текущее время системных часов                                                                                                                                                                                                             |
| <b>TimeSerial</b> (час, минута, секунда)                                     | Время для заданных значений часа, минуты и секунды                                                                                                                                                                                        |
| <b>TimeValue</b> ("время")                                                   | Преобразование строкового значения времени в подтип Время                                                                                                                                                                                 |
| <b>Weekday</b> (дата, [первый_день_нед.])                                    | Целое число – день недели                                                                                                                                                                                                                 |
| <b>WeekdayName</b> (день_нед., сокр., первый_день_нед.)                      | Строка, показывающая название заданного дня недели                                                                                                                                                                                        |
| <b>Year</b> (дата)                                                           | Целое число – год для заданной даты                                                                                                                                                                                                       |

## Приложение 4. Константы даты и времени

| Константа            | Значение | Описание                                                                                   |
|----------------------|----------|--------------------------------------------------------------------------------------------|
| vbSunday             | 1        | Воскресенье                                                                                |
| vbMonday             | 2        | Понедельник                                                                                |
| vbTuesday            | 3        | Вторник                                                                                    |
| vbWednesday          | 4        | Среда                                                                                      |
| vbThursday           | 5        | Четверг                                                                                    |
| vbFriday             | 6        | Пятница                                                                                    |
| vbSaturday           | 7        | Суббота                                                                                    |
| vbUseSystemDayOfWeek | 0        | Использовать для определения первого дня недели региональные настройки системы.            |
| vbFirstJan1          | 1        | Первой неделей в году считается та, в которой было 1 января.                               |
| vbFirstFourDays      | 2        | Первой неделей в году считается та, в которой было по крайней мере четыре дня нового года. |
| vbFirstFullWeek      | 3        | Первой неделей в году считается первая полная неделя.                                      |
| vbGeneralDate        | 0        | Дата и время выводятся в формате, определяемом региональными настройками системы.          |
| vbLongDate           | 1        | Выводить дату, используя полный формат.                                                    |
| vbShortDate          | 2        | Выводить дату, используя краткий формат.                                                   |
| vbLongTime           | 3        | Выводить время, используя полный формат.                                                   |
| vbShortTime          | 4        | Выводить время, используя краткий формат.                                                  |

## Приложение 5. Логические функции и операторы языка

| Синтаксис функции                            | Описание                                                                                                                                                        |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IsArray</i> (переменная)                  | <b>True</b> , если переменная = массив                                                                                                                          |
| <i>IsDate</i> (выражение)                    | <b>True</b> , если выражение может быть преобразовано в дату                                                                                                    |
| <i>IsEmpty</i> (выражение)                   | <b>True</b> , если выражение пустое                                                                                                                             |
| <i>IsNull</i> (выражение)                    | <b>True</b> , если выражение не содержит данных (значение <b>Null</b> )                                                                                         |
| <b>IsNumeric</b> (выражение)                 | <b>True</b> , если значение выражения – число                                                                                                                   |
| <b>IsObject</b> (выражение)                  | <b>True</b> , если выражение - объект                                                                                                                           |
| Результат = выражение1 <b>And</b> выражение2 | Логическая конъюнкция. Если выражения слева и справа от него истинны, результат <b>True</b> , иначе <b>False</b> или <b>Null</b>                                |
| Результат = выражение1 <b>Eqv</b> выражение2 | Проверка эквивалентности двух выражений. Возвращает <b>True</b> , если они имеют одинаковое значение                                                            |
| Результат = выражение1 <b>Imp</b> выражение2 | Логическая импликация. Выражение <b>E1 Imp E2</b> возвращает <b>False</b> , если <b>E1 = True</b> и <b>E2 = False</b> , во всех остальных случаях – <b>True</b> |
| Результат = object1 <b>Is</b> object2        | Проверка эквивалентности двух объектов                                                                                                                          |
| Результат = <b>Not</b> выражение             | Логическое отрицание. Возвращает <b>True</b> , если условие ложно и наоборот                                                                                    |
| Результат = выражение1 <b>Or</b> выражение2  | Логическая дизъюнкция. Должно быть истинным хотя бы одно из выражений                                                                                           |
| Результат = выражение1 <b>Xor</b> выражение2 | Логическое исключение. Выражение <b>E1 Xor E2</b> возвращает <b>True</b> , если только <b>E1 = True</b> или только <b>E2 = True</b> , иначе – <b>False</b>      |

## Приложение 6. Методы и свойства объекта ADO.Recordset

### Свойства/Коллекции

| Имя                          | Краткое описание                                                                                                                                                                                    |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AbsolutePage</b>          | Указывает, на какой странице текущая запись находится                                                                                                                                               |
| <b>AbsolutePosition</b>      | Указывает порядковую позицию текущей записи в объекте Recordset                                                                                                                                     |
| <b>ActiveCommand</b>         | Указывает объект <i>Command</i> , созданный связанным объектом Recordset                                                                                                                            |
| <b>ActiveConnection</b>      | Указывает, какому объекту <i>Connection</i> в настоящее время принадлежат указанная команда, Recordset, или запись                                                                                  |
| <b>BOF, EOF</b>              | <b>BOF</b> — указывает, что текущая позиция - перед первой записью в объекте Recordset.<br><b>EOF</b> — указывает, что текущая позиция - после последней записи                                     |
| <b>Bookmark</b>              | Закладка, которая однозначно определяет текущую запись в объекте Recordset или устанавливает текущую запись в объекте Recordset на запись, идентифицированную этой записью закладки <i>Bookmark</i> |
| <b>CacheSize</b>             | Указывает число записей объекта Recordset, которые кэшируются в памяти                                                                                                                              |
| <b>CursorLocation</b>        | Указывает местоположение курсора                                                                                                                                                                    |
| <b>CursorType</b>            | Указывает тип курсора, используемого в объекте Recordset                                                                                                                                            |
| <b>DataMember</b>            | Указывает Имя компонента данных, который будет найден для объекта, на который ссылается свойство DataSource                                                                                         |
| <b>DataSource</b>            | Указывает объект, который содержит данные, которые будут представлены как объект Recordset                                                                                                          |
| <b>EditMode</b>              | Указывает статус редактирования текущей записи                                                                                                                                                      |
| <b>Fields (Collection)</b>   | Коллекция полей для записи Recordset                                                                                                                                                                |
| <b>Filter</b>                | Указывает фильтр данных для Recordset.                                                                                                                                                              |
| <b>Index</b>                 | Указывает имя индекса, заданного для объекта Recordset                                                                                                                                              |
| <b>LockType</b>              | Указывает тип блокировки                                                                                                                                                                            |
| <b>MarshalOptions</b>        | Указывает, какие записи должны быть возвращены назад на сервер                                                                                                                                      |
| <b>MaxRecords</b>            | Указывает максимальное количество записей, которое возвращает Recordset для запроса                                                                                                                 |
| <b>PageCount</b>             | Указывает сколько страниц данных содержит объект Recordset                                                                                                                                          |
| <b>PageSize</b>              | Указывает из сколько записей состоит одна логическая страница данных в Recordset.                                                                                                                   |
| <b>Свойства (Collection)</b> | Коллекция свойств объекта                                                                                                                                                                           |
| <b>RecordCount</b>           | Указывает количество записей в объекте Recordset                                                                                                                                                    |
| <b>Sort</b>                  | Указывает одно имя поля или несколько по которым объект Recordset сортируется в порядке возрастания или убывания                                                                                    |
| <b>Source</b>                | Указывает источник данных для объекта Recordset                                                                                                                                                     |
| <b>State</b>                 | Состояние объекта: открыт, закрыт; соединяется, выполняется, выполняется                                                                                                                            |
| <b>Status</b>                | Статус текущей записи                                                                                                                                                                               |
| <b>StayInSync</b>            | Указывает, в иерархическом объекте Recordset, выполняются или нет изменения соответствующих дочерних записей, если позиция родительской строки изменяется                                           |

## Методы

| Имя                                                     | Краткое описание                                                                                                                                        |
|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AddNew</b>                                           | Создание новой записи в обновляемом объекте Recordset.                                                                                                  |
| <b>Cancel</b>                                           | Отменить выполнение                                                                                                                                     |
| <b>CancelBatch</b>                                      | Отменить выполнение пакетного обновления                                                                                                                |
| <b>CancelUpdate</b>                                     | Отменить обновление для объекта Recordset или коллекции полей или объекта Record до вызова метода Update                                                |
| <b>Clone</b>                                            | Создать дубликат объекта Recordset, доступный только для чтения                                                                                         |
| <b>Close</b>                                            | Закрыть объект и все подчиненные объекты                                                                                                                |
| <b>CompareBookmarks</b>                                 | Сравнивает две закладки и возвращает их относительное положение (CompareEnum)                                                                           |
| <b>Delete</b>                                           | Удаляет текущую запись или группу записей                                                                                                               |
| <b>Find</b>                                             | Поиск записи в объекте Recordset для заданных условий                                                                                                   |
| <b>GetRows</b>                                          | Переносит множество записей объекта Recordset в массив                                                                                                  |
| <b>GetString</b>                                        | Возвращает Recordset как строку с разделителями                                                                                                         |
| <b>Move</b>                                             | Перемещает текущую позицию в объекте Recordset                                                                                                          |
| <b>MoveFirst, Move-Last, MoveNext, and MovePrevious</b> | Перемещает текущую позицию в объекте Recordset в начало, в конец, на следующую или предыдущую запись                                                    |
| <b>NextRecordset</b>                                    | Очищает текущий объект Recordset и возвращает следующий Recordset при продвижении через ряд команд                                                      |
| <b>Open</b>                                             | Открыть курсор                                                                                                                                          |
| <b>Requery</b>                                          | Обновляет данные в объекте Recordset перезапуском запроса, на котором базируется этот объект                                                            |
| <b>Resync</b>                                           | Обновляет данные в текущем объекте Recordset, или коллекции полей объекта из основной базы данных                                                       |
| <b>Save</b>                                             | Сохраняет Recordset в файле, или объекте Stream                                                                                                         |
| <b>Seek</b>                                             | Ищет индекс Recordset, чтобы быстро определить запись, которая соответствует указанным значениям, и изменяет текущую позицию строки в объекте Recordset |
| <b>Supports</b>                                         | Определяет, поддерживает ли указанный объект Recordset специфический тип функциональности                                                               |
| <b>Update</b>                                           | Сохраняет любые изменения, которые Вы делаете в текущей строке объекта Recordset, или коллекции полей объекта Record                                    |
| <b>UpdateBatch</b>                                      | Записывает все пакетные обновления на диск                                                                                                              |

## События (events)

| Имя                                             | Краткое описание                                                                                                                                                                                          |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>EndOfRecordset</b>                           | Происходит, когда выполняется попытка двигаться за последнюю запись объекта Recordset                                                                                                                     |
| <b>FetchComplete</b>                            | Происходит после того, как все записи в длинной асинхронной операции были найдены в Recordset                                                                                                             |
| <b>FetchProgress</b>                            | Происходит периодически в течение длинной асинхронной операции, чтобы сообщить, сколько записей было найдено в операции выборки                                                                           |
| <b>WillChangeField and FieldChange-Complete</b> | Событие WillChangeField происходит прежде, чем изменяется значение одного или более полей в Recordset. Событие FieldChange-Complete происходит после того, как значение одного или более полей изменилось |
| <b>WillChangeRecord</b>                         | Событие WillChangeRecord происходит перед изменением одной                                                                                                                                                |

|                                                               |                                                                                                                                                             |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>and RecordChangeComplete</i></b>                        | или более записей в Recordset. Событие RecordChangeComplete происходит после одного или более изменений записей                                             |
| <b><i>WillChangeRecordset and RecordsetChangeComplete</i></b> | Событие WillChangeRecordset происходит прежде, чем изменяется Recordset. Событие RecordsetChangeComplete происходит после того, как Recordset изменился     |
| <b><i>WillMove and MoveComplete</i></b>                       | Событие WillMove происходит прежде, чем изменяется текущая позиция в Recordset. Событие MoveComplete происходит после изменения текущей позиции в Recordset |
| <b><i>WillMove and MoveComplete</i></b>                       | Событие WillMove происходит прежде, чем изменяется текущая позиция в Recordset. Событие MoveComplete происходит после изменения текущей позиции в Recordset |



## **Литература**

А. С. Грошев. Информатика: Учебник для вузов. – Архангельск, 2008. – 428 с.