

Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Иркутский государственный аграрный университет имени А.А. Ежевского
Институт экономики, управления и прикладной информатики
Кафедра информатики и математического моделирования

Асалханов П.Г.

Операционные системы

Методические указания к лабораторным работам для студентов направления
09.03.03 Прикладная информатика

Иркутск 2020

Лабораторная работа №1. РАБОТА В СРЕДЕ MS-DOS (ВНУТРЕННИЕ И ВНЕШНИЕ КОМАНДЫ)

✓ Для работы в среде операционной системы используем режим

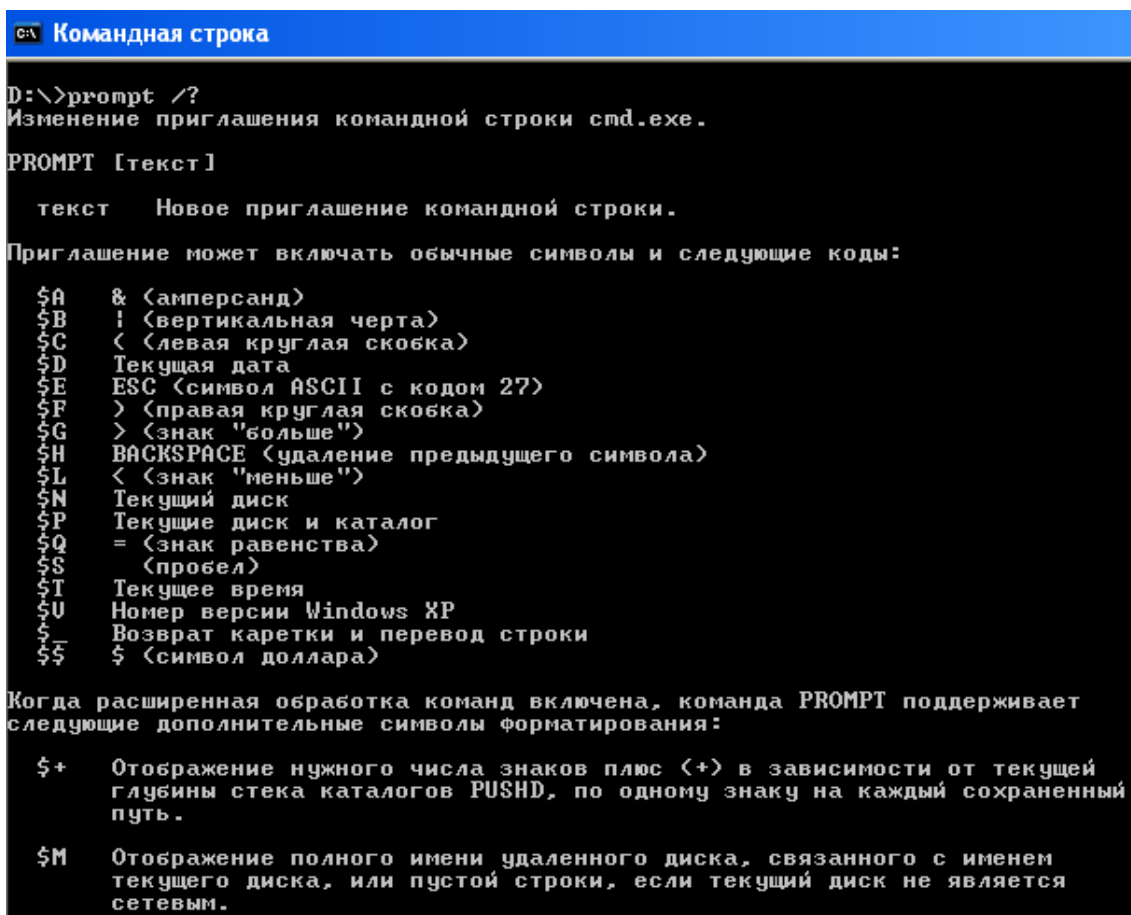
командной строки Windows – Командная строка  Командная строка .

Задание 1.1 Настроить вид приглашения ОС при помощи внутренней команды *prompt*.

Порядок работы:

1. Для получения справки о ключах и параметрах выполнить следующую команду (рис.1.1):

PROMPT /? ↵



```
D:\>prompt /?
Изменение приглашения командной строки cmd.exe.

PROMPT [текст]

    текст    Новое приглашение командной строки.

Приглашение может включать обычные символы и следующие коды:

$A    & (амперсанд)
$B    | (вертикальная черта)
$C    < (левая круглая скобка)
$D    Текущая дата
$E    ESC (символ ASCII с кодом 27)
$F    > (правая круглая скобка)
$G    > (знак "больше")
$H    BACKSPACE (удаление предыдущего символа)
$L    < (знак "меньше")
$N    Текущий диск
$P    Текущие диск и каталог
$Q    = (знак равенства)
$S    (пробел)
$T    Текущее время
$U    Номер версии Windows XP
$_    Возврат каретки и перевод строки
$$    $ (символ доллара)

Когда расширенная обработка команд включена, команда PROMPT поддерживает
следующие дополнительные символы форматирования:

$+    Отображение нужного числа знаков плюс (+) в зависимости от текущей
      глубины стека каталогов PUSHD, по одному знаку на каждый сохраненный
      путь.

$M    Отображение полного имени удаленного диска, связанного с именем
      текущего диска, или пустой строки, если текущий диск не является
      сетевым.
```

Рисунок 1.1 Результат выполнения команды PROMPT /?

2. Установить приглашение следующего вида: (D) (рис.1.2).

PROMPT \$C\$N\$F ↵

```
D:\>prompt $C$N$F
(D) _
```

Рисунок 1.2 Настройка вида приглашения ОС

3. Установить стандартный вид приглашения: D:\>

PROMPT \$P\$G↵

4. Настроить приглашение вида:

&Текущая_дата&Текущее_время&Версия ОС= (рис.1.3)

Выполнить самостоятельно.



```
27.02.201210:19:02,72Microsoft Windows XP [Версия 5.1.2600] _
```

Рисунок 1.3 Результат выполнения текущего задания

5. Вернуть стандартный вид приглашения ОС. Выполнить самостоятельно.

Задание 1.2 Просмотреть и при необходимости настроить системные дату и время при помощи внутренних команд ОС: DATE и TIME. Выполнить самостоятельно.

Задание 1.3 Создать дерево заданной структуры (рис. 1.4):

Порядок работы:

✓ *Следите за изменением вида приглашения к работе операционной системы*

1. Переход на рабочий диск:

D: ↵

2. Просмотр оглавления диска:

D:\>**DIR**↵

3. Создание каталога PORTFEL:

D:\>**MD _PORTFEL**↵

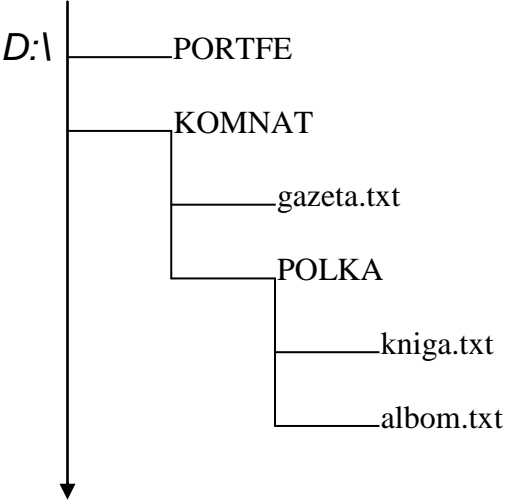
Структура:	Полные имена
 <pre> D:\ ├── PORTFE │ └── KOMNATA │ ├── gazeta.txt │ └── POLKA │ ├── kniga.txt │ └── albom.txt </pre>	D:\PORTFEL D:\KOMNATA D:\KOMNATA\gazeta.txt D:\KOMNATA\POLKA D:\KOMNATA\POLKA\kniga.txt D:\KOMNATA\POLKA\albom.txt

Рисунок 1.4 Структура к заданию 1.3

4. Просмотр оглавления диска:

D:\>**DIR**↵

5. Создание каталога KOMNATA:

D:\>**MD _KOMNATA**↵

6. Просмотр оглавления диска:

D:\>**DIR**↵

7. Открытие каталога KOMNATA:

D:\>**CD _KOMNATA**↵

8. Просмотр оглавления каталога:

D:\KOMNATA>**DIR**↵

9. Создание файла gazeta.txt:

D:\KOMNATA>**COPY_CON_gazeta.txt**↵

✓ *Обратить внимание на отсутствие приглашения к работе операционной системы в режиме ввода текста!*

- переключиться на русский алфавит и набрать текст:

«Нет ни в чем вам благодати; ↵

С счастием у вас разлад: ↵

И прекрасны вы некстати, ↵

И умны вы невпопад.» ↵

А.С. Пушкин↵

- завершить создание файла, для чего необходимо нажать функциональную клавишу <F6> - в результате появится признак конца файла ^Z, ↵ (рис.1.5).

```
D:\komnata>copy con gazeta.txt
"Нет ни в чем вам благодати;
С счастьем у вас разлад:
И прекрасны вы некстати,
И умны вы невпопад."
А.С. Пушкин
^Z
Скопировано файлов:      1.
D:\komnata>_
```

Рисунок 1.5 Создание текстового файла

10. Просмотр оглавления каталога KOMNATA

D:\KOMNATA>**DIR**↵

11. Создание каталога POLKA

D:\KOMNATA>**MD POLKA**↵

12. Просмотр оглавления каталога KOMNATA (рис.1.6).

D:\KOMNATA>**DIR**↵

13. Открытие каталога POLKA (рис.1.6)

D:\KOMNATA>**CD POLKA**↵

✓ *Обратить внимание на изменение вида приглашения к работе операционной системы!*

```
D:\komnata>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 08EC-CC18

Содержимое папки D:\komnata
27.02.2012 10:32 <DIR>      .
27.02.2012 10:32 <DIR>      ..
27.02.2012 10:31          117 gazeta.txt
27.02.2012 10:32 <DIR>      polka
                1 файлов          117 байт
                3 папок 17 305 399 296 байт свободно

D:\komnata>cd polka
D:\komnata\polka>
```

Рисунок 1.6 Просмотр каталога и смена текущего каталога

14. Просмотр оглавления каталога POLKA

D:\KOMNATA\POLKA>**DIR**↵

15. Создание файла kniga.txt

D:\КОМНАТА\POLKA>**COPY_CON_kniga.txt**↵

✓ *Обратить внимание на отсутствие приглашения к работе операционной системы в режиме ввода текста!*

- переключиться на русский алфавит и набрать текст:

«Полу-милорд, полу-купец, ↵

Полу-мудрец, полу-невежда, ↵

Полу-подлец, но есть надежда, ↵

Что будет полным наконец.» ↵

А.С. Пушкин↵

- завершить создание файла, для чего необходимо нажать функциональную клавишу <F6>

- в результате появится признак конца файла ^Z, ↵

16. Просмотр оглавления каталога POLKA:

D:\КОМНАТА\POLKA>**DIR**↵

17. Создание файла albom.txt:

D:\КОМНАТА\POLKA>**COPY_CON_albom.txt**↵

✓ *Обратить внимание на отсутствие приглашения к работе операционной системы в режиме ввода текста!*

- переключиться на русский алфавит и набрать текст:

«Судьба свои дары явить желала в нем, ↵

В счастливом баловне соединив ошибкой↵

Богатство, знатный род – с возвышенным умом↵

И простодушие с язвительной улыбкой.» ↵

А.С. Пушкин↵

- завершить создание файла, для чего необходимо нажать функциональную клавишу

<F6> - в результате появится признак конца файла ^Z, ↵

18. Просмотр оглавления каталога POLKA:

D:\КОМНАТА\POLKA>**DIR**↵

✓ *Создание структуры завершено!*

Задание 1.4 Скопировать файл `gazeta.txt` в каталог POLKA с тем же именем.

Порядок работы:

1. Заккрытие каталога POLKA и переход в родительский для него каталог KOMNATA:

```
D:\KOMNATA\POLKA>CD.. ↵
```

2. Копирование файла `gazeta.txt`:

```
D:\KOMNATA>COPY_gazeta.txt_POLKA↵
```

✓ *Использование собственно имен файла (`gazeta.txt`) и каталога (POLKA) возможно благодаря тому, что рабочим каталогом является каталог KOMNATA*

✓ *Имя файла-приемника не указывается, так как требуется копии дать имя файла-источника*

3. Просмотр результатов копирования:

-открытие каталога POLKA

```
D:\KOMNATA>CD_POLKA↵
```

-просмотр оглавления каталога POLKA

```
D:\KOMNATA\POLKA>DIR↵
```

В результате копирования в каталоге должен появиться файл `gazeta.txt`.

Задание 1.5 Скопировать файл `albom.txt` в каталог KOMNATA с именем `portret.txt`.

Порядок работы:

1. D:\KOMNATA\POLKA>COPY_albom.txt_\KOMNATA\portret.txt↵

- для просмотра результатов копирования выйти из каталога POLKA в родительский для него каталог KOMNATA и вывести оглавление каталога на экран

2. D:\KOMNATA\POLKA>CD..↵

3. D:\KOMNATA>DIR↵

✓ Для копирования на другое устройство **всегда** указывается **полное имя файла-приемника!**

Задание 1.6 Переместить файл `gazeta.txt` из каталога `KOMNATA` в каталог `PORTFEL` с тем же именем.

Порядок работы:

✓ Команды перемещения в среде операционной системы нет, поэтому операция перемещения выполняется в два этапа: копирование, а затем удаление файла-источника!

1. Копирование файла `gazeta.txt` из каталога `KOMNATA` в каталог `PORTFEL` с тем же именем:

```
D:\KOMNATA>COPY_gazeta.txt_\PORTFEL↵
```

2. Проверка результатов копирования:

```
D:\KOMNATA>DIR_\PORTFEL↵
```

3. Удаление файла-источника:

```
D:\KOMNATA>DEL_gazeta.txt↵
```

4. Проверка результатов удаления:

```
D:\KOMNATA>DIR↵
```

Задание 1.7 Переместить файл `portret.txt` из каталога `KOMNATA` в каталог `POLKA` с именем `foto.txt`.

Порядок работы:

1. Открытие каталога `POLKA`:

```
D:\KOMNATA>CD_POLKA↵
```

2. Просмотр оглавления каталога `POLKA`:

```
D:\KOMNATA\POLKA>DIR↵
```

3. Копирование файла `portret.txt` из каталога `KOMNATA` в каталог `POLKA` с именем `foto.txt`:

```
D:\KOMNATA\POLKA>COPY_\KOMNATA\portret.txt_foto.txt↵
```

4. Просмотр результатов копирования:

```
D:\KOMNATA\POLKA>DIR↵
```

5. Удаление файла-источника:


```
D:\KOMNATA\POLKA>DEL _\KOMNATA\portret.txt
```

6. Проверка удаления файла:

```
D:\KOMNATA\POLKA>DIR _\KOMNATA
```

Задание 1.8 Переименовать файлы в каталоге POLKA по шаблону, заменив вторую букву имени на #.

Порядок работы:

1. Переименование файлов в каталоге POLKA по шаблону, с заменой второй буквы имени на символ #:

```
D:\KOMNATA\POLKA>REN *.* ?#*.*
```

2. Просмотр результатов переименования (рис.1.7):

```
D:\KOMNATA\POLKA>DIR
```

```
D:\komnata\polka>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 08EC-CC18

Содержимое папки D:\komnata\polka
27.02.2012 10:41 <DIR> .
27.02.2012 10:41 <DIR> ..
27.02.2012 10:38 178 albom.txt
27.02.2012 10:39 119 gazeta.txt
27.02.2012 10:40 130 kniga.txt
          3 файлов          427 байт
          2 папок 17 305 391 104 байт свободно

D:\komnata\polka>ren *.* ?#*.*

D:\komnata\polka>dir
Том в устройстве D не имеет метки.
Серийный номер тома: 08EC-CC18

Содержимое папки D:\komnata\polka
27.02.2012 10:41 <DIR> .
27.02.2012 10:41 <DIR> ..
27.02.2012 10:38 178 a#bom.txt
27.02.2012 10:39 119 g#zeta.txt
27.02.2012 10:40 130 k#iga.txt
          3 файлов          427 байт
          2 папок 17 305 391 104 байт свободно

D:\komnata\polka>
```

Рисунок 1.7 Просмотр результатов переименования

Задание 1.9 Слить файлы a#bom.txt, k#iga.txt и g#zeta.txt в каталоге POLKA, результат поместить в каталог PORTFEL с именем vse.txt.

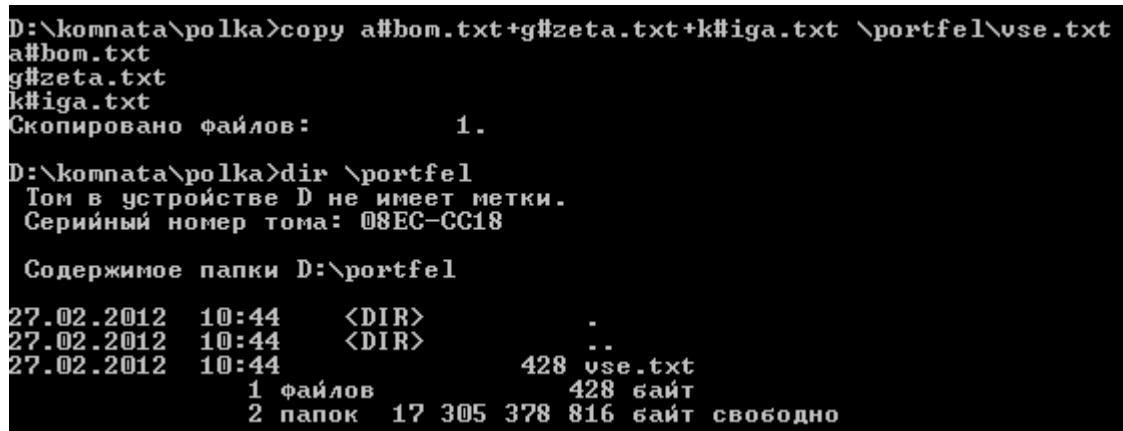
Порядок работы:

1. Слияние файлов a#bom.txt, k#iga.txt и g#zeta.txt в каталоге POLKA, результат требуется поместить в каталог PORTFEL с именем vse.txt:

```
D:\KOMNATA\POLKA>COPY_a#bom.txt+k#iga.txt+g#zeta.txt_
_\PORTFEL\vse.txt
```

2. Просмотр результата слияния (рис.1.8):

```
D:\KOMNATA\POLKA>DIR_\PORTFEL
```



```
D:\komnata\polka>copy a#bom.txt+g#zeta.txt+k#iga.txt \portfel\vse.txt
a#bom.txt
g#zeta.txt
k#iga.txt
Скопировано файлов:          1.

D:\komnata\polka>dir \portfel
Том в устройстве D не имеет метки.
Серийный номер тома: 08EC-CC18

Содержимое папки D:\portfel

27.02.2012  10:44    <DIR>          .
27.02.2012  10:44    <DIR>          ..
27.02.2012  10:44                428 vse.txt
                1 файлов        428 байт
                2 папок   17 305 378 816 байт свободно
```

Рисунок 1.8 Просмотр результата слияния

Задание 1.10 Просмотреть содержимое результирующего файла vse.txt.

Порядок работы:

1. Переход в корневой каталог:

```
D:\KOMNATA\POLKA>CD\
```

2. Открытие каталога PORTFEL:

```
D:\>CD_PORTFEL
```

3. Просмотр оглавления каталога PORTFEL:

```
D:\PORTFEL>DIR
```

4. Просмотр содержимого файла vse.txt:

```
D:\PORTFEL>TYPE_vse.txt
```

Задание 1.11 Скопировать все файлы из каталога POLKA в каталог PORTFEL, используя шаблон, объединяющий все текстовые файлы в каталоге.

Порядок работы:

1. D:\PORTFEL>COPY_\KOMNATA\POLKA*.TXT

✓ Если при копировании не указывать каталог-приемник и имя копии, то результат копирования будет помещен в текущий каталог с именем файла-источника!

2. Просмотреть оглавление каталога.

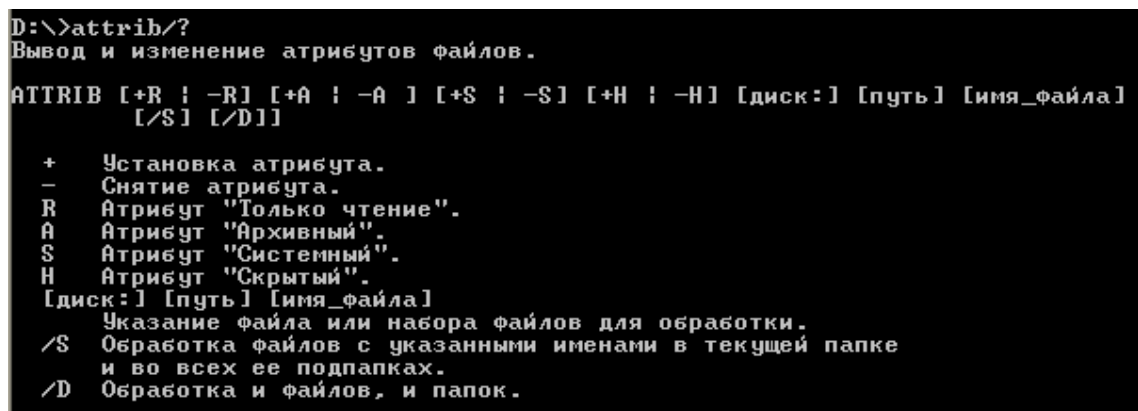
D:\PORTFEL>**DIR**↵

Задание 1.12 Установить файлу vse.txt из каталога PORTFEL атрибут скрытый.

Порядок работы:

1. Вывод и изменение атрибутов файлов в среде операционной системы выполняет команда ATTRIB. Для получения справки о команде в командной строке набрать и исполнить (рис.1.9):

ATTRIB/?↵



```
D:\>attrib/?
Вывод и изменение атрибутов файлов.

ATTRIB [+R | -R] [+A | -A ] [+S | -S] [+H | -H] [диск:] [путь] [имя_файла]
[/S] [/D]

+ Установка атрибута.
- Снятие атрибута.
R Атрибут "Только чтение".
A Атрибут "Архивный".
S Атрибут "Системный".
H Атрибут "Скрытый".
[диск:] [путь] [имя_файла]
    Указание файла или набора файлов для обработки.
/S Обработка файлов с указанными именами в текущей папке
и во всех ее подпапках.
/D Обработка и файлов, и папок.
```

Рисунок 1.9 Справка о команде ATTRIB

2. Открыть каталог PORTFEL и просмотреть его оглавление.

Выполнить самостоятельно.

3. Для установки атрибута в командной строке набрать команду:

ATTRIB_+H _vse.txt↵

4. Просмотреть оглавление каталога PORTFEL. Выполнить самостоятельно.

5. Просмотреть оглавление текущего каталога с установленными объектам атрибутами, включая скрытые. Выполнить самостоятельно.

6. Отменить атрибут «скрытый» файла vse.txt:

ATTRIB_-H _vse.txt↵

Задание 1.13 Построить полученное дерево папок и файлов с помощью внешней команды операционной системы TREE.

1. Перейти в корневой каталог диска D:

2. Выполнить команду:

TREE /F

Результат показать преподавателю.

Задание 1.14 Удалить полученную структуру с диска D:

✓ *При удалении следует помнить о том, что нельзя удалить каталог до тех пор, пока он не пуст!*

Порядок работы:

1. Удаление каталога PORTFEL:

- удаление содержимого каталога:

```
D:\PORTFEL>DEL_*.txt ↵
```

-просмотр результатов удаления:

```
D:\PORTFEL>DIR↵
```

- выход из каталога в корневой каталог:

```
D:\PORTFEL>CD\↵
```

-удаление каталога PORTFEL:

```
D:\>RD_PORTFEL↵
```

-просмотр результатов удаления:

```
D:\>DIR↵
```

2. Удаление каталога KOMNATA:

✓ *При удалении следует помнить о том, что нельзя удалить каталог до тех пор, пока он не пуст!*

-откроем каталог KOMNATA:

```
D:\>CD_KOMNATA↵
```

-удаление файлов в каталоге POLKA:

```
D:\KOMNATA>DEL_POLKA↵
```

На экране появится запрос на удаление всех файлов в каталоге, для удаления нажать на клавиатуре «Y». В результате каталог будет очищен.

-просмотр результатов удаления файлов:

```
D:\KOMNATA>DIR_POLKA↵
```

-удаление каталога POLKA:

```
D:\KOMNATA>RD_POLKA↵
```

-просмотр результатов удаления:

```
D:\KOMNATA>DIR↵
```

-удаление файлов из каталога KOMNATA:

```
D:\KOMNATA>DEL *.*↵
```

-просмотр результатов удаления:

```
D:\KOMNATA>DIR↵
```

-закрывать каталог KOMNATA:

```
D:\KOMNATA>CD\↵
```

-удалить каталог KOMNATA:

```
D:\>RD KOMNATA↵
```

-просмотр результатов удаления:

```
D:\>DIR↵
```

Задание 1.15 Выполнить любые три варианта из заданий для самостоятельной работы 1, используя команды операционной системы.

Выполнить самостоятельно.

Задание 1.16 Сформировать BAT-файл для создания структуры из задания 1.3. Сохранить этот файл в свой каталог.

✓ BAT-файл – это текстовый файл, содержанием которого является набор команд. BAT-файлы называются командными или пакетными файлами. Создаются BAT-файлы с помощью любого текстового редактора. В среде операционной системы MS-DOS можно воспользоваться командой создания текстового файла.

Порядок работы:

1. Создание файла (текущий каталог – свой рабочий):

```
COPY CON _sozдание.bat↵
```

2. Набрать текст файла (каждая команда в отдельной строке):

```
D: ↵
```

```
CD\↵
```

```
MD _PORTFEL↵
```

```
MD _KOMNATA↵
```

CD_KOMNATA↵
COPY_CON_gazeta.txt↵
MD_POLKA↵
CD_POLKA↵
COPY_CON_kniga.txt↵
COPY_CON_albom.txt↵
CD. ↵
COPY_gazeta.txt_POLKA↵
CD_POLKA↵
COPY_albom.txt_KOMNATA\portret.txt↵
CD.↵
COPY_gazeta.txt_PORTFEL↵
DEL_gazeta.txt↵
CD_POLKA↵
COPY_KOMNATA\portret.txt_foto.txt↵
DEL_KOMNATA\portret.txt↵
REN_. *_?#*. ** ↵
COPY_a#bom.txt+k#iga.txt+g#zeta.txt_PORTFEL\vse.txt↵
DIR_PORTFEL↵
CD↵
CD_PORTFEL↵
TYPE_vse.txt↵
COPY_KOMNATA\POLKA.TXT*↵

3. Завершить создание файла. Выполнить самостоятельно.

4. Запустить BAT-файл и проверить результат его работы. При необходимости исправить ошибки. Для редактирования файла можно использовать БЛОКНОТ.

✓ При выполнении команд создания текстовых файлов будут появляться запросы на ввод содержимого. Необходимо набирать текст и завершать создание файлов.

5. Продемонстрировать результат преподавателю.

Задание 1.17 Создать BAT-файл для удаления полученной структуры с именем `udalenie.bat` в своем каталоге. Выполнить самостоятельно. Продемонстрировать результат работы файла преподавателю.

Задание 1.18 Создать BAT-файл для выполнения любого варианта из заданий для самостоятельной работы 1. Работу файла показать преподавателю.

Задание 1.19 Создать BAT-файл для удаления структуры из Задания 1.18. Результат показать преподавателю.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 1

Вариант 1

1. Сформировать дерево заданной структуры (рис.1.10).

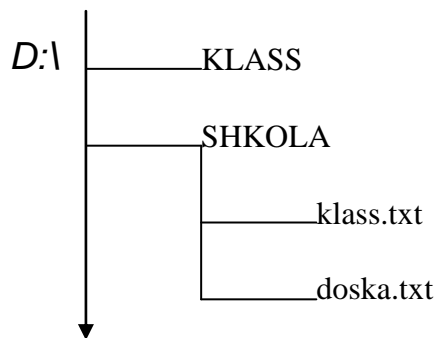


Рисунок 1.10 Структура к варианту 1

2. Скопировать из каталога SHKOLA файлы в каталог KLASS с теми же именами.
3. Слить в каталоге KLASS файлы в один, результат записать в каталог SHKOLA с именем `zavuch.txt`.
4. Просмотреть содержимое файла `zavuch.txt`.
5. Установить файлу `zavuch.txt` атрибут «только чтение».
6. Получить графическое представление полученной структуры.

Вариант 2

1. Сформировать дерево заданной структуры (рис.1.11).
2. В каталоге PRIMER создать файл `text.txt`.
3. Скопировать из каталога KONTR оба файла в каталог PRIMER с теми же именами.

4. В каталоге PRIMER слить файлы в один, результат поместить в этот же каталог с именем prim.txt.

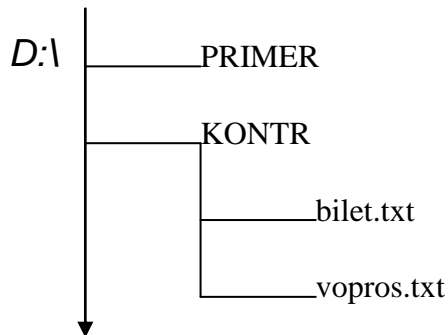


Рисунок 1.11 Структура к варианту 2

5. Просмотреть содержимое каталога PRIMER.
6. Просмотреть содержимое слитого файла prim.txt.
7. Установить файлу prim.txt атрибут «скрытый».
8. Получить графическое представление полученной структуры.

Вариант 3

1. Сформировать дерево заданной структуры (рис.1.12).

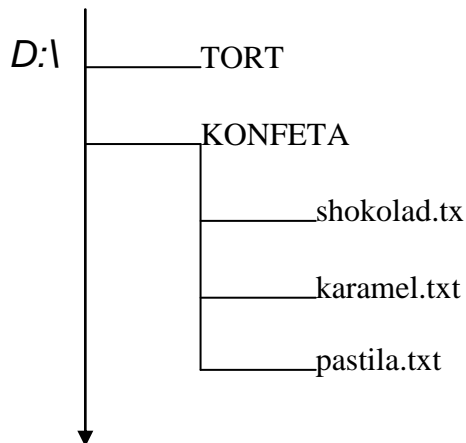


Рисунок 1.12 Структура к варианту 3

2. В каталоге KONFETA слить файлы shokolad.txt и pastila.txt в один, результат записать в каталог TORT с именем marmelad.txt.
3. Файл karamel.txt скопировать в каталог TORT с тем же именем.
4. В каталоге TORT просмотреть содержимое обоих файлов.
5. Просмотреть оглавление обоих каталогов.
6. Установить файлу marmelad.txt атрибут «скрытый».

7. Получить графическое представление полученной структуры.

Вариант 4

1. Сформировать дерево заданной структуры (рис.1.13).

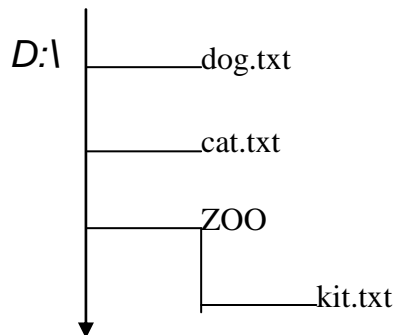


Рисунок 1.13 Структура к варианту 4

2. Скопировать файлы dog.txt и cat.txt в каталог ZOO.
3. Слить в каталоге ZOO файлы в один, результат записать в этот же каталог с именем tigr.txt.
4. Просмотреть содержимое слитого файла tigr.txt.
5. Просмотреть оглавление каталога ZOO.
6. Установить файлу tigr.txt атрибут «только чтение».
7. Получить графическое представление полученной структуры.

Вариант 5

1. Сформировать дерево заданной структуры (рис.1.14).

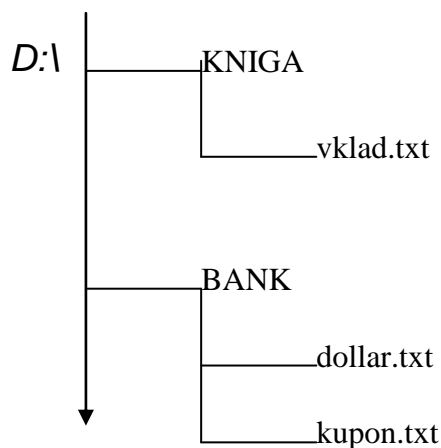


Рисунок 1.14 Структура к варианту 5

2. Скопировать из каталога BANK файл dollar.txt в каталог KNIGA с тем же именем.

3. Слить в каталоге KNIGA файлы vklad.txt, dollar.txt и kupon.txt из каталога BANK в один, результат записать в каталог BANK с именем rubl.txt.
4. Просмотреть содержимое файла rubl.txt.
5. Установить файлу rubl.txt атрибут «только чтение».
6. Получить графическое представление полученной структуры.

Вариант 6

1. Сформировать дерево заданной структуры (рис.1.15).

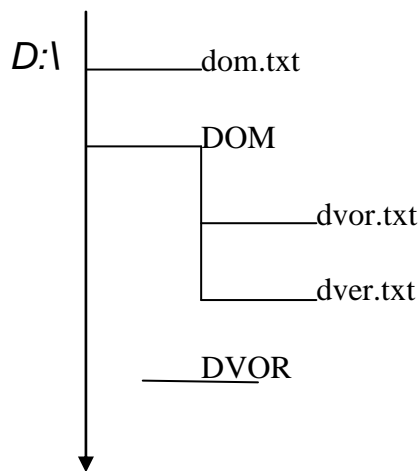


Рисунок 1.15 Структура к варианту 6

2. Скопировать файл dom.txt в каталог DOM с тем же именем.
3. Слить в каталоге DOM файлы в один, результат записать в каталог DVOR с именем zabor.txt.
4. Просмотреть содержимое файла zabor.txt.
5. Установить файлу zabor.txt атрибут «скрытый».
6. Получить графическое представление полученной структуры.

Вариант 7

1. Сформировать дерево заданной структуры (рис.1.16).

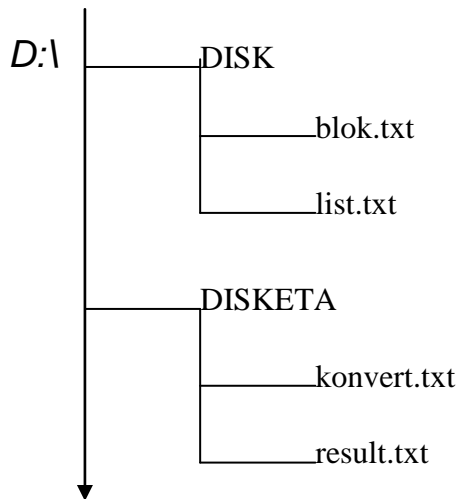


Рисунок 1.16 Структура к варианту 7

2. Слить в каталоге DISK файлы, результат поместить в каталог DISKETA с именем blesk.txt.
3. Слить в каталоге DISKETA файлы, результат записать в каталог DISK с именем reka.txt.
4. Просмотреть содержимое файлов reka.txt, blesk.txt.
5. Установить файлу reka.txt атрибут «скрытый».
6. Получить графическое представление полученной структуры.

Вариант 8

1. Сформировать дерево заданной структуры (рис.1.17).

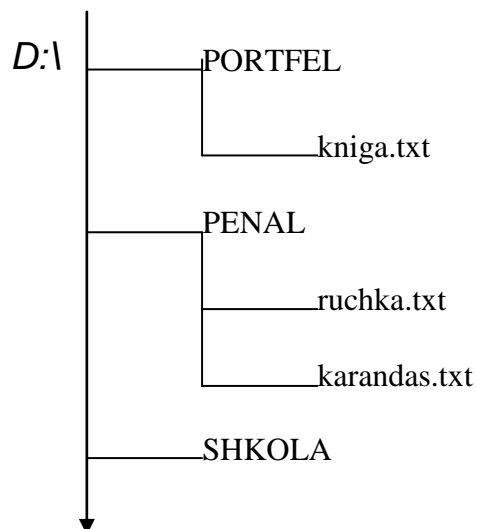


Рисунок 1.17 Структура к варианту 8

2. Слить в каталоге PENAL файлы, результат записать в каталог PORTFEL с именем rezinka.txt.

3. Просмотреть содержимое файла rezinka.txt.
4. Из каталога PORTFEL скопировать файлы в каталог SHKOLA с заменой расширения на .007.
5. Слить в каталоге SHKOLA файлы, результат записать в каталог PORTFEL с именем первого суммируемого файла.
6. Установить файлу rezinka.txt атрибут «только чтение».
7. Получить графическое представление полученной структуры.

Вариант 9

1. Сформировать дерево заданной структуры (рис.1.18).

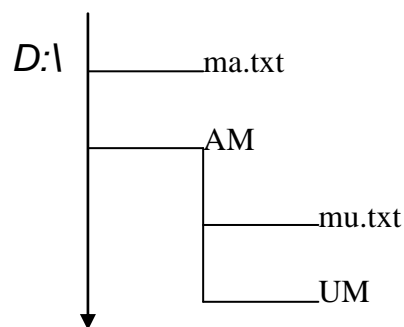


Рисунок 1.18 Структура к варианту 9

2. Скопировать файл ma.txt в каталог AM с тем же именем.
3. Слить в каталоге AM файлы в один, результат записать в этот же каталог с именем ammu.txt.
4. Просмотреть содержимое файла ammu.txt.
5. Скопировать файл ammu.txt в каталог UM с именем maum.txt.
6. Установить файлу maum.txt атрибут «только чтение».
7. Получить графическое представление полученной структуры.

Вариант 10

1. Сформировать дерево заданной структуры (рис.1.19).

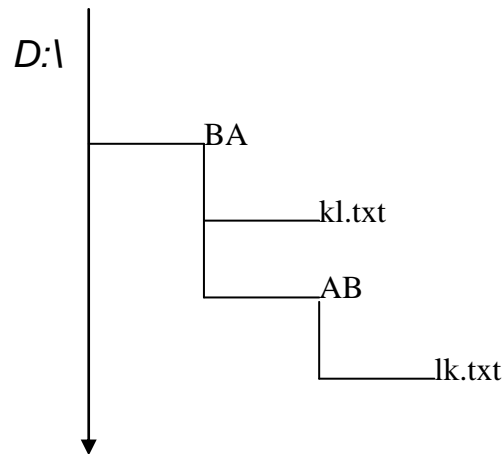


Рисунок 1.19 Структура к варианту 10

2. Скопировать файл kl.txt в каталог АВ с тем же именем.
3. Слить в каталоге АВ файлы в один, результат записать в каталог ВА с именем kllk.txt.
4. Просмотреть содержимое файла kllk.txt.
5. Скопировать файл kllk.txt в каталог АВ с именем lkkl.txt.
6. Установить файлу lkkl.txt атрибут «скрытый».
7. Получить графическое представление полученной структуры.

Вариант 11

1. Сформировать дерево заданной структуры (рис.1.20).

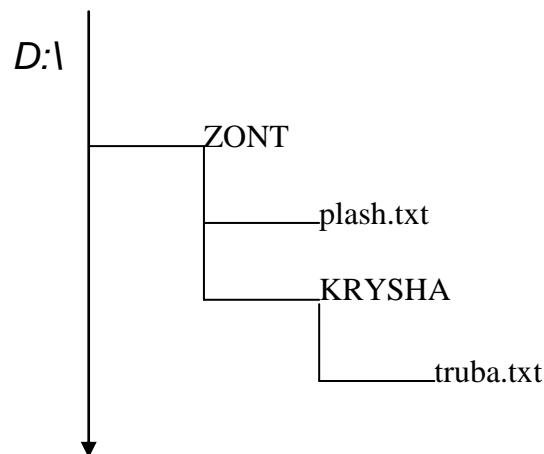


Рисунок 1.20 Структура к варианту 11

2. Скопировать файл splash.txt в каталог KRYSHA с тем же именем.
3. Слить в каталоге KRYSHA файлы в один, результат записать в каталог ZONT с именем voda.txt.

4. Просмотреть содержимое файла voda.txt.
5. Установить файлу voda.txt атрибут «скрытый».
6. Просмотреть оглавление каталога ZONT.
7. Отменить атрибут «скрытый».
8. Получить графическое представление полученной структуры.

Вариант 12

1. Сформировать дерево заданной структуры (рис.1.21).
2. В каталоге NOCH слить файлы в один, результат поместить в каталог ОБЛАКО с именем tucha.txt.
3. Просмотреть содержимое файла tucha.txt.
4. Файл svet.txt скопировать в каталог ОБЛАКО с тем же именем.
5. В каталоге ОБЛАКО слить файлы в один, результат поместить в каталог DEN с именем zvezda.txt.
6. Установить файлу zvezda.txt атрибут «только чтение».

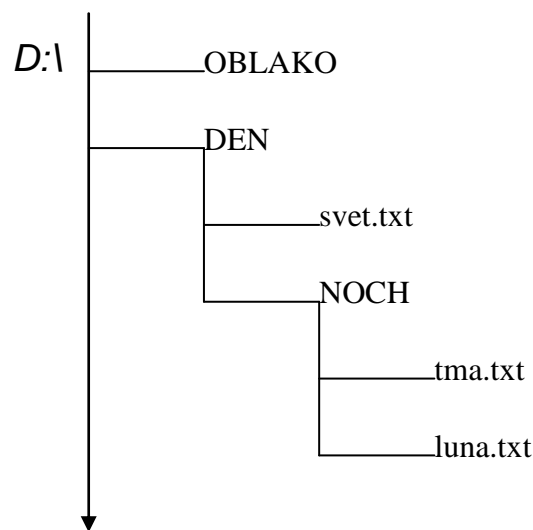
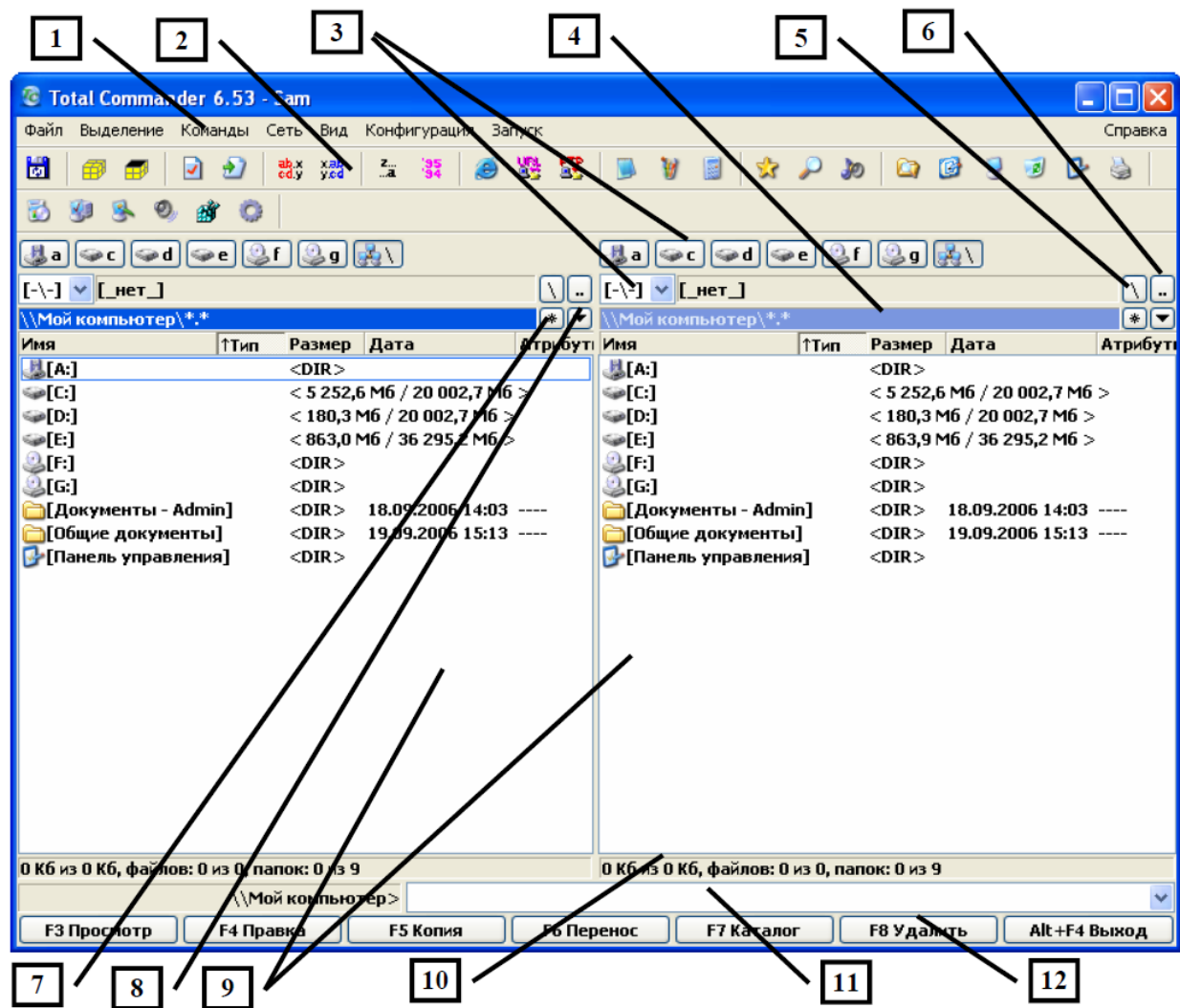


Рисунок 1.21 Структура к варианту 12

7. Просмотреть оглавление каталога DEN.
8. Просмотреть содержимое файла zvezda.txt.
9. Установить файлу zvezda.txt атрибут «скрытый».
10. Получить графическое представление построенной структуры.

Лабораторная работа №2. Основы работы с файловым менеджером Total Commander

Файловый менеджер **Total Commander (TC)** представляет удобный доступ к файлам и папкам, позволяет осуществлять все операции, используемые при работе с файлами и папками. Внешний вид файлового менеджера показан на рисунке 1.



- 1 – главное меню; 6 – переход на один уровень вверх; 11 – командная строка;
2 – панель инструментов; 7 – избранные каталоги; 12 – панель
3 – кнопки выбора дисков; 8 – история; функциональных клавиш.
4 – текущий путь; 9 – файловые панели;
5 – переход в корневой каталог; 10 – информационная строка;

Рисунок 1 – Файловый менеджер **Total Commander**

Основные операции с файлами:

1) Выделение файлов

Чтобы выделить файлы или каталоги, просто щёлкните по ним мышью или переместитесь на них с помощью клавиш курсора и нажмите **INSERT**.

Если вы выбираете каталог, используя клавишу **ПРОБЕЛ**, показывается размер дискового пространства, занятого этим каталогом. Все описанные ниже операции доступны из меню **Выделение**.

2) Выделение нескольких последовательных объектов

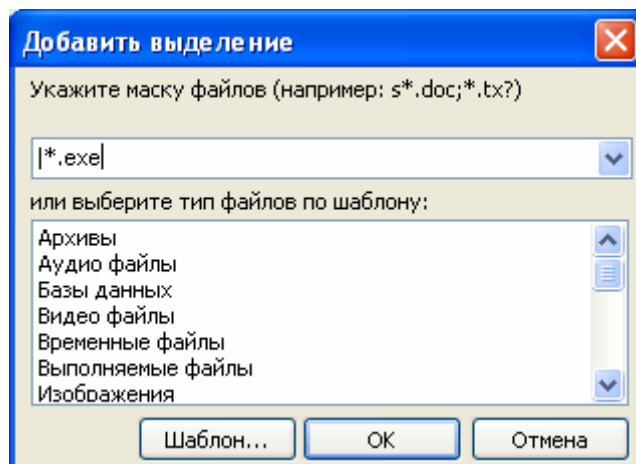
Щёлкните по первому файлу или каталогу, предназначенному для выделения. После этого нажмите клавишу **SHIFT** и, удерживая её, щёлкните левой кнопкой мыши на последнем объекте, который хотите выделить.

3) Выделение нескольких несмежных объектов

Выделяйте левой кнопкой мыши любые несмежные файлы или каталоги, держа при этом нажатой клавишу **CTRL** (снятие выделения с отдельного файла/каталога выполняется точно так же).

4) Выделение и отмена выделения определённых типов файлов

Нажмите клавишу **Num+** (или **Num-**) или выберите одну из команд выделения **Выделить группу** → **Снять выделение группы** в меню **Выделение**. Затем в появившемся диалоге введите нужный вам тип файла (например, *.txt). Вы можете также указать несколько типов файлов, и даже те типы файлов, которые не должны быть выделены. Их следует отделить символом вертикальной черты "|".



Пример 1: w*.*|*.bak *.old Выделить все файлы, которые начинаются с w и не заканчиваются .bak или .old.

Пример 2: |*.exe Выделить все файлы, кроме программ.

5) Выделить всё / Снять всё выделение

Нажмите сочетание клавиш **CTRL+Num+** (или, соответственно, **CTRL+Num-**) или выберите команду **Выделить всё / Снять всё выделение** в меню **Выделение**. Выделить всё содержимое файловой панели можно также при помощи комбинации клавиш **CTRL+A**.

6) Выделить по расширению / Снять выделение по расширению

Выбрав файл с нужным вам расширением, нажмите сочетание клавиш ALT+Num+ (или ALT+Num–), чтобы выделить все файлы с таким же расширением в текущей панели или, соответственно, снять выделение с этих файлов.

7) Инвертировать выделение

Эта команда отметит все файлы в исходном каталоге, которые не были отмечены, и снимет выделение у ранее отмеченных файлов. Для вызова команды нажмите клавишу Num* (умножение).

8) Изучить самостоятельно пункты:

Сохранить выделение, Восстановить выделение, Сохранить выделение в файл, Загрузить выделение из файла. Определить горячие клавиши соответствующих команд.

Обновление содержимого панели:

Нажмите CTRL+R. При этом обновится текущая панель. Эту операцию следует выполнить, например, после смены гибкого диска, чтобы обновить содержимое файловой панели.

Просмотр содержимого файлов:

Выберите файлы, которые хотите просмотреть, и нажмите F3. Встроенная программа просмотра файлов **Lister** показывает содержимое файла под курсором.

Интегрированный просмотрщик файлов позволяет просматривать файлы в текстовом, двоичном или шестнадцатеричном формате, Unicode-файлы и HTML-страницы, файлы растровой графики, мультимедиа и файлы RTF. Быстрый просмотр выделенного файла можно выполнить при помощи нажатия **Ctrl + Q**.

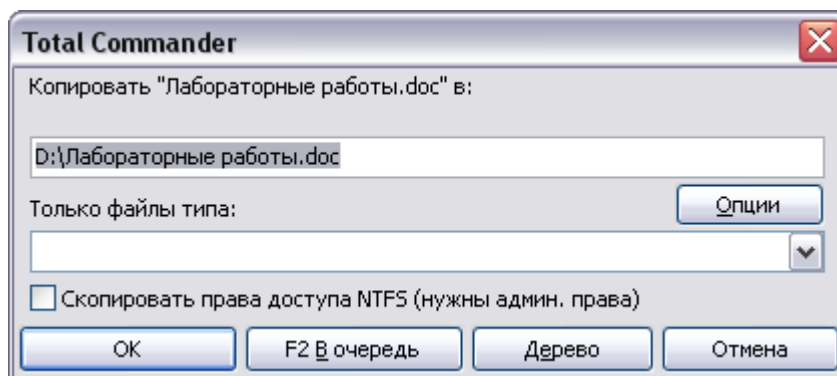
Правка (F4)

Поместите курсор на файл, который вы хотите редактировать, и нажмите F4. При этом запускается выбранный вами в диалоге настройки редактор, а в него загружается выбранный файл. По умолчанию запускается стандартный Блокнот Windows. Он может работать только с текстовыми файлами ограниченного размера.

Если вы хотите редактировать файлы других типов, просто дважды щёлкните на файле или нажмите **ENTER**. Запустится программа, ассоциированная с файлом.

Копирование (F5) и Перемещение (F6)

Эта команда копирует файлы и целые каталоги из исходного каталога в каталог на другой панели. Выделите файлы, которые вы хотите скопировать, и нажмите **F5**. При этом откроется диалоговое окно, в котором вы можете ввести каталог назначения и маску файлов.



Чтобы скопировать файл в тот же самый каталог (под другим именем), нажмите **SHIFT+F5**. Ярлык для файла можно создать комбинацией клавиш **CTRL+SHIFT+F5**.

Кнопкой **Дерево** вы можете выбрать каталог назначения из дерева каталогов.

При нажатии кнопки **F2 В очередь** выбранные файлы будут добавлены в список последнего открытого диспетчера фоновой пересылки. Это полезно при копировании нескольких больших файлов друг за другом, что более эффективно, чем параллельное копирование их всех в фоновом режиме.

Кнопка **Опции** позволяет установить параметры для автоматического копирования. По умолчанию ТС выводит запрос о перезаписи файлов. Эта кнопка позволяет, например, установить по умолчанию "Заменить все", "Пропустить все" или "Заменить все старые".


В поле "**Только файлы типа**", вы можете указать, какие файлы копировать, причём это распространяется также на файлы из подкаталогов.


Примеры:

- | | |
|-------------------|---|
| *.txt *.doc | Будут копироваться только файлы .doc и .txt. |
| *.* *.bak *.old | Будет копироваться всё, кроме файлов .bak и .old. |

Для перемещения файлов воспользуйтесь клавишей **F6**.

Архивы

Если вы хотите создать новый архив и упаковать в него выделенные файлы, просто нажмите кнопку  на панели инструментов или сочетание **ALT+F5**. Откроется диалоговое окно упаковки файлов. При использовании сочетания **ALT+SHIFT+F5** файлы после упаковки будут удалены.

Если вы хотите распаковать архив под курсором (или выделенные архивы), нажмите кнопку  на панели инструментов или **ALT+F9**. После указания каталога назначения (и при необходимости – маски файлов), все файлы из архива будут распакованы.

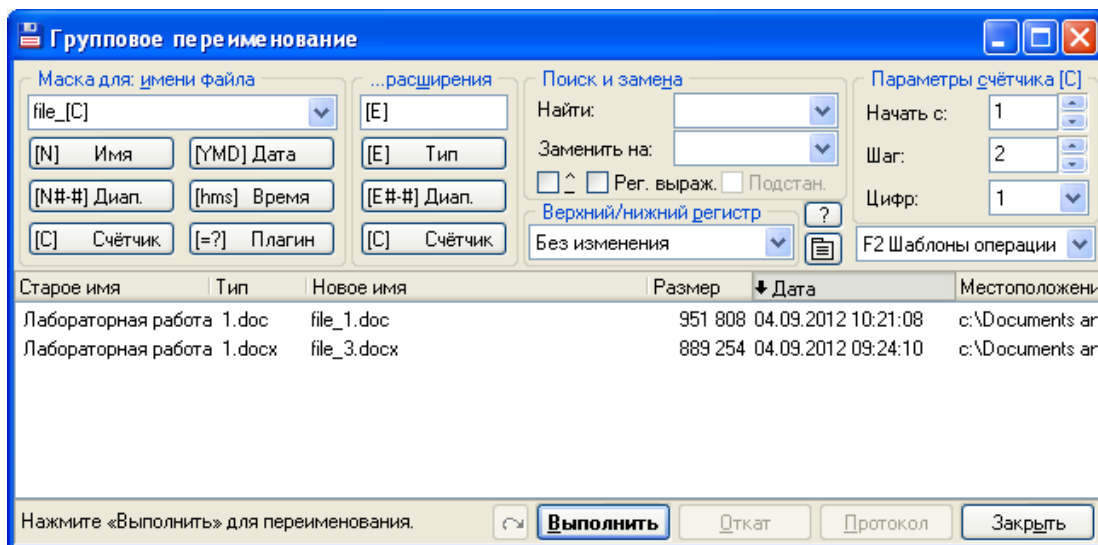
Также для выполнения данных операций можно пользоваться пунктами главного меню **Файл** → **Упаковать** и **Файл** → **Распаковать** или контекстным меню группы выделенных файлов (**Добавить в архив..**) или архива (**Извлечь файлы...**)

Переименование файлов и каталогов

Для переименования файла или каталога под курсором нажмите правую кнопку мыши и выберите пункт **Переименовать**. Также можно воспользоваться сочетанием **SHIFT+F6**.

Групповое переименование файлов

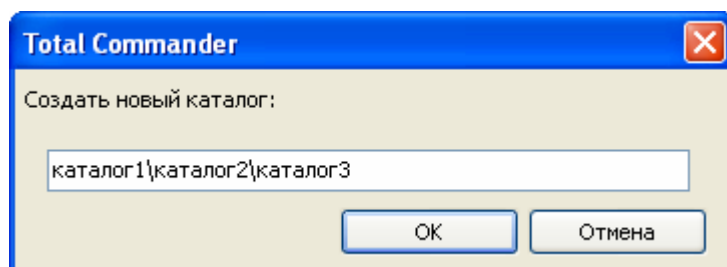
Выделите несколько файлов и выберите пункт **Файл** → **Групповое переименование**. Также можно воспользоваться сочетанием **CTRL+M**. В открывшемся диалоговом окне можно настроить формат имени группы файлов. Например:



После нажатия на клавишу «выполнить» файлы будут переименованы в соответствии с заданным форматом.

Создание каталога (F7)

Эта команда создаёт новый подкаталог в исходном каталоге. После нажатия **F7** просто введите желаемое имя каталога. Можно также создавать и несколько подкаталогов за одну операцию. Просто отделите подкаталоги обратной косой чертой (обратный слэш), например:



Проверьте, что произойдет, если в поле имени каталога ввести **каталог1\каталог2\каталог3**.

Удаление (F8)

Выделите файлы и/или каталоги, которые хотите удалить, и нажмите F8. После подтверждения файлы удаляются. Процесс может быть прерван в любой момент кнопкой '**Отмена**'. Для каждого непустого каталога будет запрашиваться подтверждение в дополнительном диалоговом окне. Предупреждение: все файлы И ПОДКАТАЛОГИ в этом каталоге будут удалены.

Поиск файлов

Используйте пункт меню **Инструменты** → **Поиск файлов** или сочетание клавиш **ALT+F7**.

В открывшемся диалоговом окне можно задать маску для поиска файлов. Имена, содержащие пробелы, **ДОЛЖНЫ** быть помещены в двойные кавычки, например, "Письмо к Иванову.doc", иначе Total Commander искал бы каждую часть имени по отдельности. Для удаления ненужных записей из истории поиска, вы можете использовать сочетание **Shift+Del**.

Примеры:

*.ini находит, например, **win.ini**

Иванов находит "Письмо к Иванову.doc"

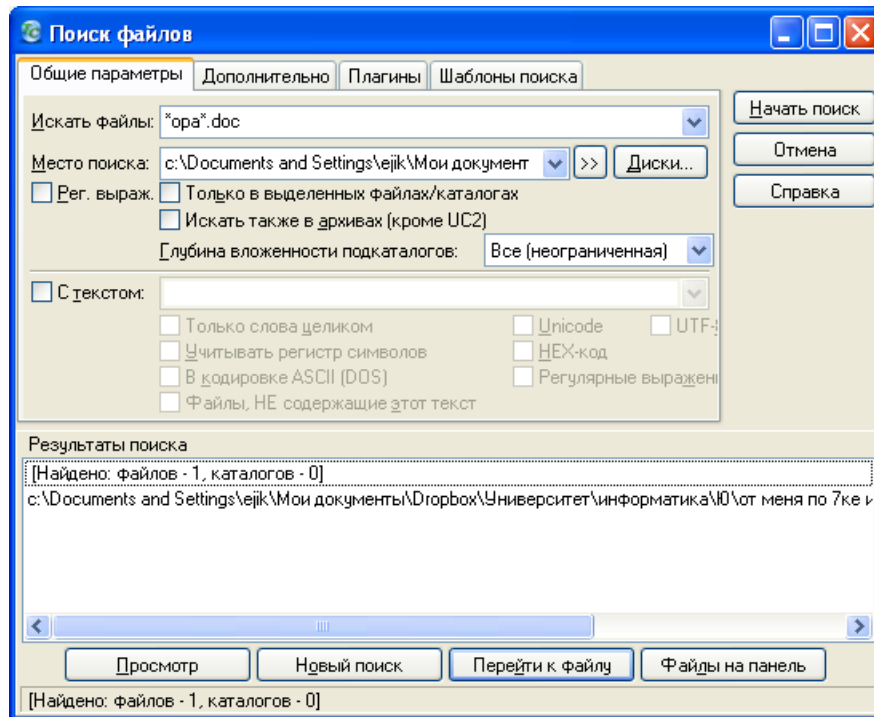
*.bak *.sik *.old находит все файлы резервных копий с этими расширениями

*n.ini теперь находит имена, которые **обязательно** содержат 'n' перед точкой.

*n?.ini находит имена, которые **обязательно** содержат 'n' и ещё *ровно* один символ перед точкой. Например, *Wcmd_eng.ini*.

w*.*.bak *.old находит файлы, которые начинаются с w и не заканчиваются на .bak или .old

На вкладке «Дополнительно» можно указать такие параметры поиска, как размер файла, дата создания. На вкладке «Шаблоны поиска» можно сохранить созданный вами запрос или открыть уже введенный шаблон.



Результаты поиска

Содержит список всех найденных файлов. Подробности (дата и время модификации, размер) показаны под списком из-за недостатка свободного пространства. Дважды щёлкните на файле, чтобы перейти в каталог, в котором этот файл находится. Чтобы скопировать весь список файлов в буфер обмена, щёлкните в этом окне и нажмите **Ctrl+C**. Щелчок правой кнопкой на найденном файле открывает системное контекстное меню для него.

Просмотр Загружает выделенный файл во внутреннюю программу просмотра (Lister).

Новый поиск Закрывает нижнюю часть диалога для нового поиска.

Перейти к файлу Если вы выбрали файл в списке, вы можете перейти в каталог, где находится этот файл, нажав "Перейти к файлу". Удерживая нажатой клавишу **Shift**, вы откроете этот каталог в новой вкладке.

Файлы на панель Передаёт найденные файлы в исходную файловую панель, где они могут быть скопированы, перемещены или удалены. Это работает, только когда поиск внутри архивов НЕ включён! Удерживая нажатой клавишу **Shift**, вы откроете результаты поиска в новой вкладке. С помощью **F2** или **Ctrl+R** вы можете вернуться к обычному режиму отображения файлов.

Сравнение файлов по содержимому

Выделите файлы, которые нужно сравнить. Используйте пункт меню **Файлы** → **Сравнить по содержимому**..

Задания для самостоятельной работы:

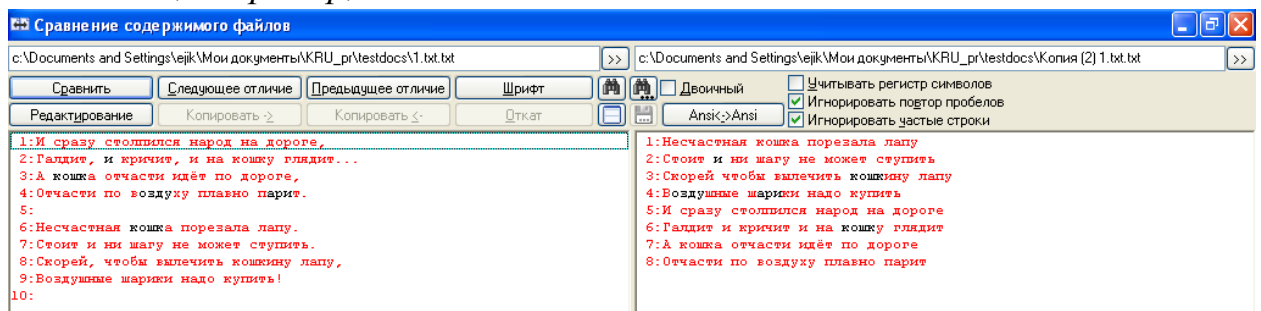
1. Создайте в своем каталоге папку **Лабораторная_ТС**.
2. За **одно** нажатие клавиши **F7** в папке **Лабораторная_ТС** создайте каталог **Учеба**, с подкаталогами **Информатика** и **Физика**.

*В отчёт включите текст команды, которую вы использовали. Например, при выполнении команды **папка1/папка2|папка3** в папке **Лабораторная_ТС** будут созданы каталоги **папка1** и **папка3**, находящиеся внутри **Лабораторная_ТС**, и **папка2**, находящийся внутри каталога **папка1** (Проверьте).*

3. Создайте в каталоге **Информатика** 3 текстовых файла **file01.txt**, **file02.txt**, **file03.txt**. Введите в них текст (не менее 30 символов) при помощи Блокнота. Создайте 2 изображения **img1.bmp**, **img2.bmp**.

4. Скопируйте файл **file03.txt** в каталог **Физика**, переместите файл **file02.txt** в эту же папку.

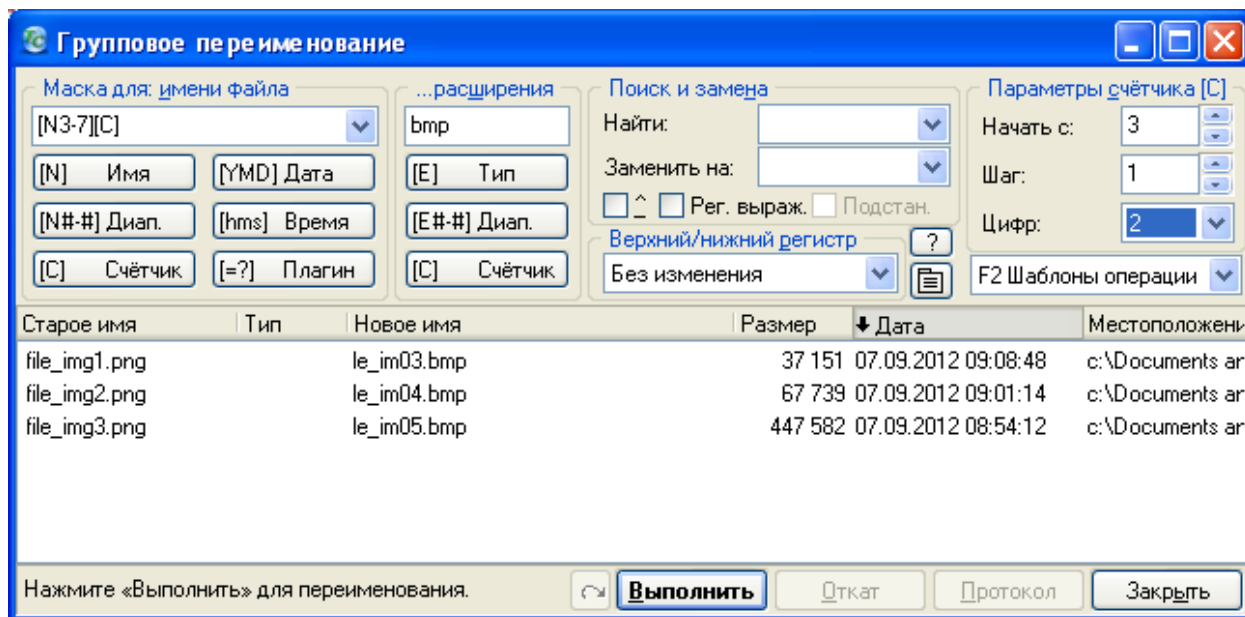
5. Выполните сравнение файлов **file03.txt** и **file02.txt** по содержимому. *В отчёт включите скриншот с результатами сравнения. Он может выглядеть, например, так:*



6. Переименуйте файлы из папки **Физика** в: **new_file001.txt**, **new_file003.txt**, используя групповое переименование файлов.

В отчёт включите скриншот диалогового окна с настройками для переименования файлов. Например, настройки для переименования группы

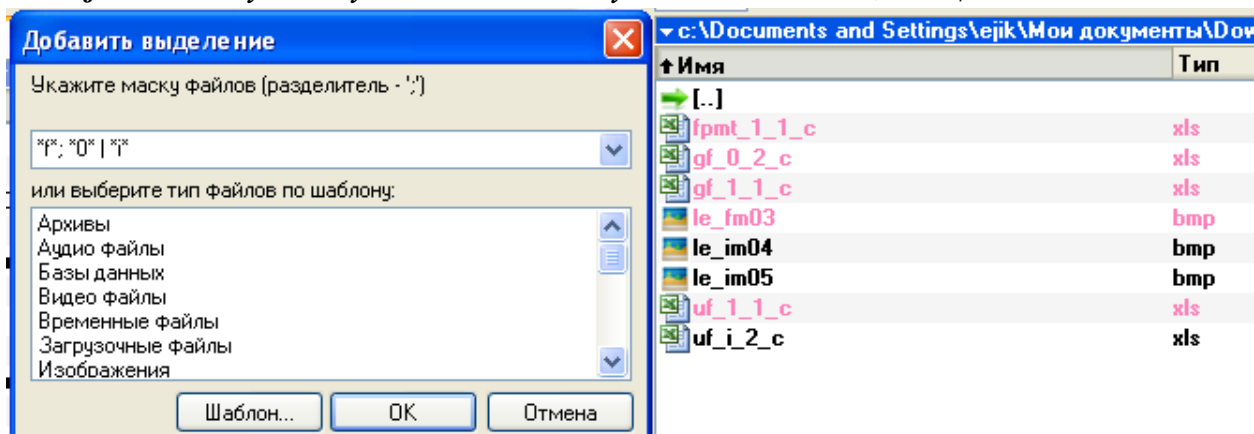
файлов *file_img1.png*, *file_img2.png*, *file_img3.png* в *le_im03.bmp*, *le_im04.bmp*, *le_im05.bmp* могут выглядеть так:



В заданиях 7-9 в отчёт включите текст маски и скриншот с результатами её применения для выделения файлов.

7. Создайте маску для выделения изображений.
8. Создайте маску для выделения файлов с именем, начинающимся с символов "fil".
9. Создайте маску для выделения файлов с расширением .txt , в названии которых присутствует сочетание "img", но отсутствует сочетание "fil".

Например, чтобы выделить файлы, в названии которых присутствует "0" или "f", но отсутствует "i", используется маска *f*; *0* | *i*



В заданиях 10-14 в отчёт включите скриншот окна с настройками и результатами поиска.

10. Найдите все файлы на диске Z, созданные в течение трех дней.

11. Найдите все файлы с расширением **.txt**, имена которых начинаются на **“fi”** на диске **Z**.

12. Найдите все файлы с расширением **.txt**, имена которых начинаются на **“fi”** или содержат в имени файла сочетание **“img”** на диске **Z**.

13. Найдите на диске **Z** все изображения, используя шаблоны поиска.

14. Найдите на диске **Z** все файлы, содержащие в названии строку **“file”**, после которой находится ровно 2 символа, а затем следует расширение.

*Подсказка: например, найдётся файл **new_file01.txt**.*

15. Заархивируйте папку **Лабораторная_ТС**. Создайте **.zip** и **.rar** – архивы; самораспаковывающийся архив (какое у него расширение?); архив, защищённый паролем. *Результаты продемонстрируйте преподавателю. В отчёт внесите заполненную таблицу.*

Размер несжатой папки, Кб	Размер zip-архива, Кб	Размер rar-архива, Кб	Размер самораспаковывающегося архива, Кб	Размер zip-архива, защищённого паролем, Кб

16. Пропредмонстрируйте отчёт и результаты выполнения лабораторной работы преподавателю.

Лабораторная работа № 3 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 меньше возможности имеет 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.

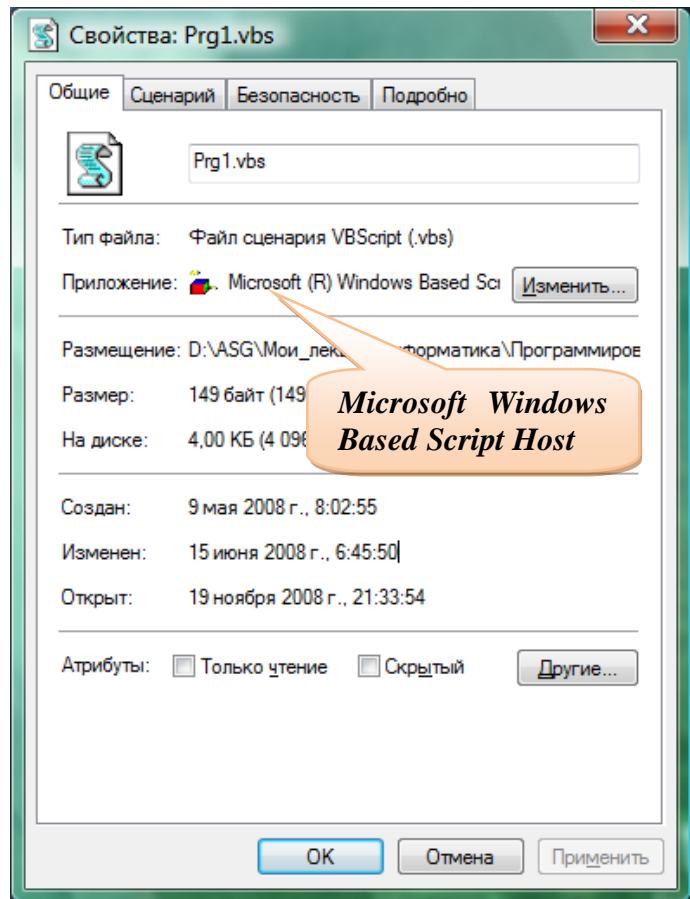


Рисунок 1.1. Окно свойств файла Prg1.vbs

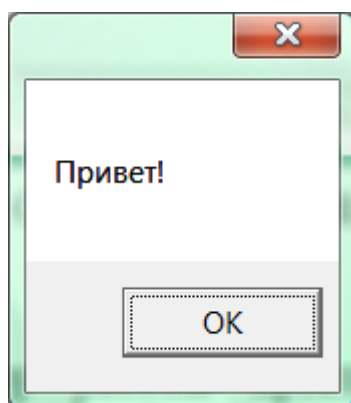


Рисунок 1.2. Пример работы простейшей программы на языке Vbscript, исполняемой системой *Windows Based Script Host* (*wscript.exe*)

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
```

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

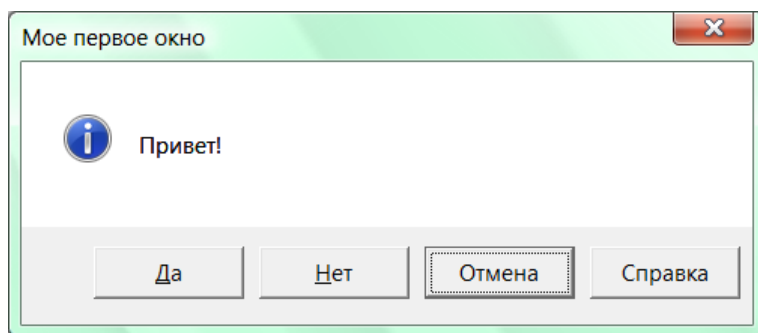


Рисунок 1.3.
Пример использования функции MsgBox языка Vbscript с заданием набора кнопок и иконки

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

		всех других окон
vbOK	1	Нажата кнопка ОК .
vbCancel	2	Нажата кнопка Отмена (Cancel)
vbAbort	3	Нажата кнопка Стоп (Abort)
vbRetry	4	Нажата кнопка Повтор (Retry)
vbIgnore	5	Нажата кнопка Пропустить (Ignore)
vbYes	6	Нажата кнопка Да (Yes)
vbNo	7	Нажата кнопка Нет (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* :

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

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

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 напишем 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).

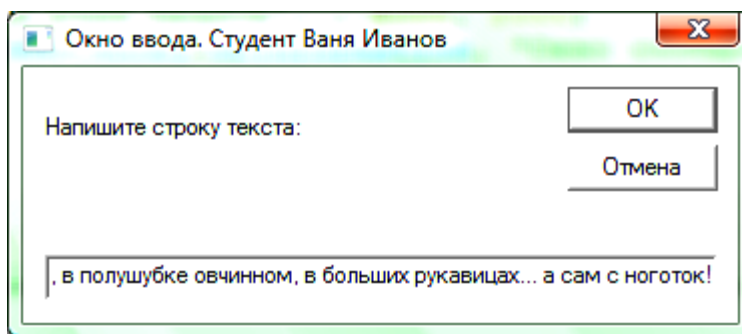


Рисунок 1.4.
Пример использования функции **InputBox**

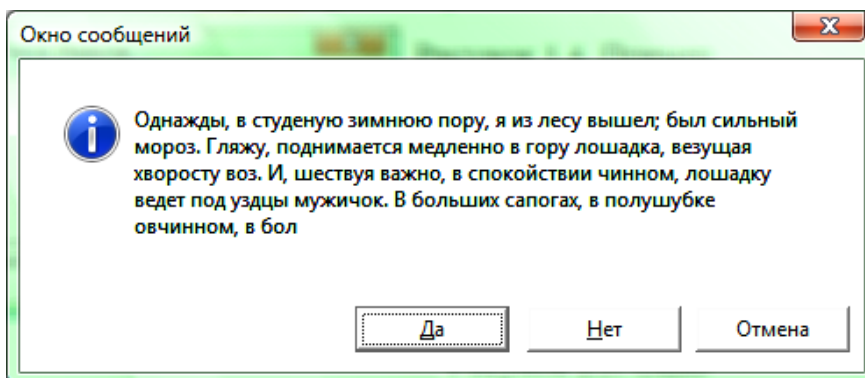


Рисунок 1.5.
Окно функции **MsgBox** с показом текста, написанного в поле окна **InputBox**

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>])]
```

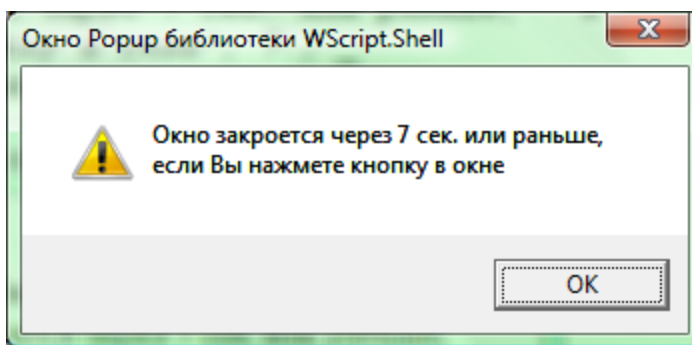


Рисунок 1.6. Использование метода *Popup* объекта *WScript.Shell*

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 **PopUp** 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 **PopUp** 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 **PopUp** 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 **PopUp** 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 **PopUp** 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 **PopUp** 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 **PopUp** 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 **PopUp** 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 Да и Нет и значком Information Mark. Затем в окне **PopUp** показать код нажатой кнопки при выходе из предыдущего окна. Определить коды нажатия для всех кнопок.

Лабораторная работа № 4. 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
Byte	Целые числа в диапазоне от 0 до 255.
Boolean	Логические значения <i>True</i> или <i>False</i> .
Integer	Целые числа в диапазоне от -32768 до 32767.
Long	Целые числа в диапазоне от -2 147 483 648 до 2 147 483 647.
Single	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.
Double	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.
Currency	-922 337 203 685 477.5808 до 922 337 203 685 477.5807.
Date / (Time)	Числа, которые представляют даты и время в диапазоне между 1-01-100 0:0:0 до 31-12-9999 23:59:59.
Object	Содержит объект
String	Строка переменной длины, которая максимально может содержать 2 миллиона символов.
Empty	Неинициализированное значение (0 для числовых переменных, строка нулевой длины ("") для строковых переменных).
Null	Содержит неверные для подтипа данные.
Error	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 predefined констант, которые сгруппированы по категориям:

1. **Date and Time Constants** – определяют константы для дат и времени для функций работы с ними;
2. **Date Format Constants** – определяют форматы дат и времени;
3. **MsgBox Constants** – используются в функции **MsgBox** и других диалоговых окнах;
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, vbDouble, 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 № 5. 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) – двухмерный 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) & vbLf & "2-ой элемент массива Y - число: " &  
& Y(1) & vbLf & "100-ый элемент массива Y - дата: " &
```

```

    & Y(99) & vbCrLf & 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.

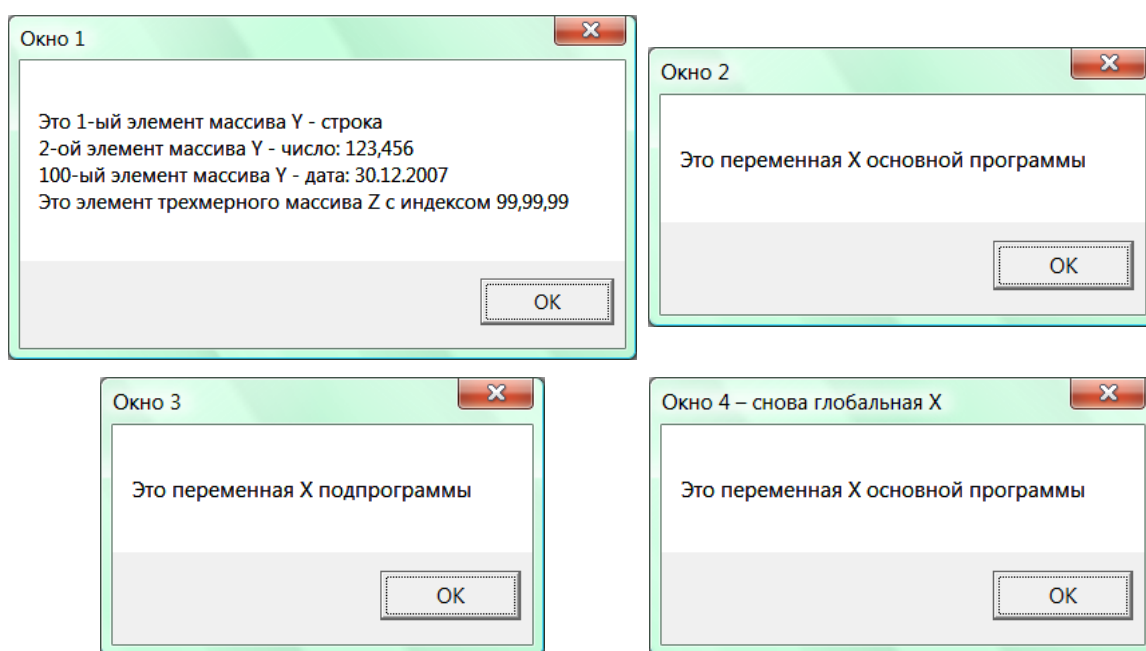


Рисунок 3.1. Использование различных описаний переменных и массивов

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 2×3 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 3×2 . 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 № 6. 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*):

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

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-n> Then  
        [операторы-n]] ...  
    [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-n – то же, что и **условие**;

операторы-n – выполняются, если **условие-n** 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-n**, идущее за **ElseIf** (если они есть) проверяется. Когда истинное значение найдено, выполняются **операторы-n**, 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 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 $\#гггг-мм-дд\#$.

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:

- строчного синтаксиса оператора условного перехода,
- блочного синтаксиса оператора условного перехода.

Таблица 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 начина- ющая 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 Но- вым годом!»
	X – время от 6 час. 01 мин. до 12 час.00 мин.	Y= «утро»		X – дата от 1 января 0000 года по 31 де- кабря 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 де- кабря 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 отри- цательное»
	X – месяц от 3 по 5	Y= «весна»		X \geq 0 и X<10 ⁻¹⁵	Y= «X ма- ленькое по- ложительное число»
	X – месяц от 6 по 8	Y= «лето»		X \geq 10 ⁻¹⁵ и X<1	Y= «X мень- ше 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.

Приложение. Логические функции и операторы языка

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

Лабораторная работа № 7. 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 & vbLf & "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 *wscript.exe* и нажать кнопку **Завершить процесс** (End Process).

Пример использования операторов циклов *Do...Loop While* и *Do...Loop Until* (результат работы – рисунок 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." & vbLf & vbLf _
& " 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 & vbLf & " Редко бывает больше!"
ElseIf proc>85 Then
    s = s & vbLf & " Совсем неплохо!"
ElseIf proc>75 Then
    s = s & vbLf & " Бывает и больше...!"
Else
    s = s & vbLf & " Маловато!"
End If
Otvet1 = MsgBox(S & vbLF & vbLF & _
    "Повторить генерацию случайных чисел?", _
    vbYesNo, "Оценка результата... ")
Loop While Otvet1 = vbYes
    'цикл повторяется, если нажата кнопка Да

```

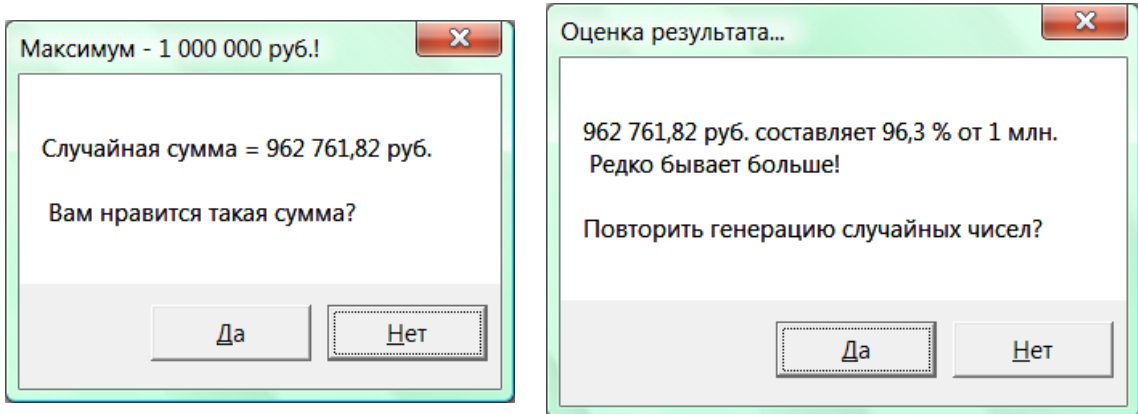


Рисунок 1. Пример использования циклов *Do...Loop*

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 $y = \text{arcctg}(x)$ 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) & vbLf & _
    "Расчетное значение "& s & vbLf & _
    "Погрешность по a = " & FormatNumber(Abs(a),15) & _
    & vbLf & "Разница расч.- ист.= " & _
    FormatNumber(Abs(Atn(x)-s),15) & vbLf & _
    "Продолжительность расчета " & _
    FormatDateTime(t2-t1) & vbLf & _
    " k = " & k , vbExclamation, _
    "Y = arcctg(x). Вариант 1"

```

Результат выполнения этой программы показан на рисунке 2.

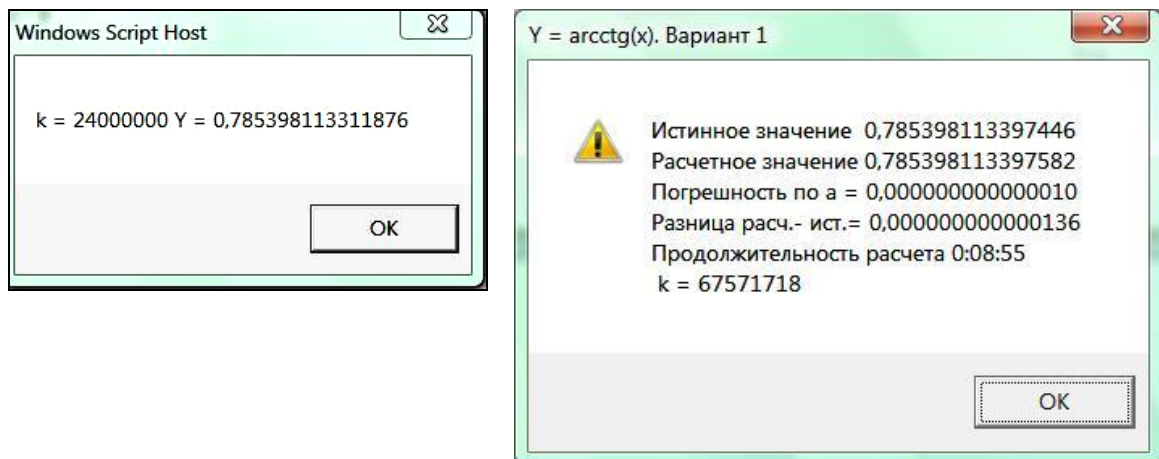


Рисунок 2. Результаты работы программы с циклом While . . . Wend

Задания

Вычислите число π 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 π постоянными остаются 7 знаков после запятой. Определите программно время расчета для каждого варианта, покажите в окне со-

общений таблицу, показывающую расчетное значение π , 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.

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

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

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

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

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

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

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

Лабораторная работа № 8. 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 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 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"
```

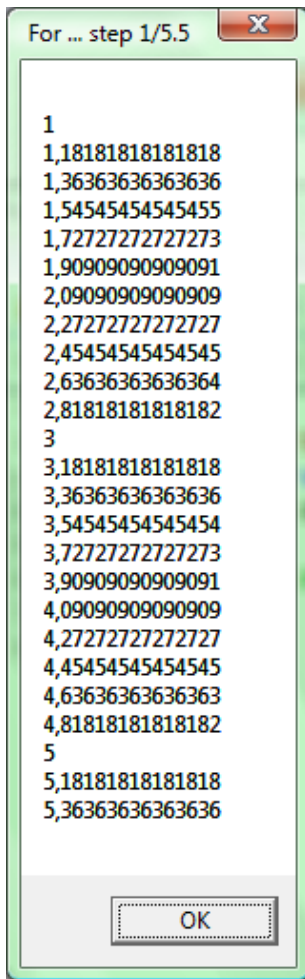



Рисунок 1.
For...Next

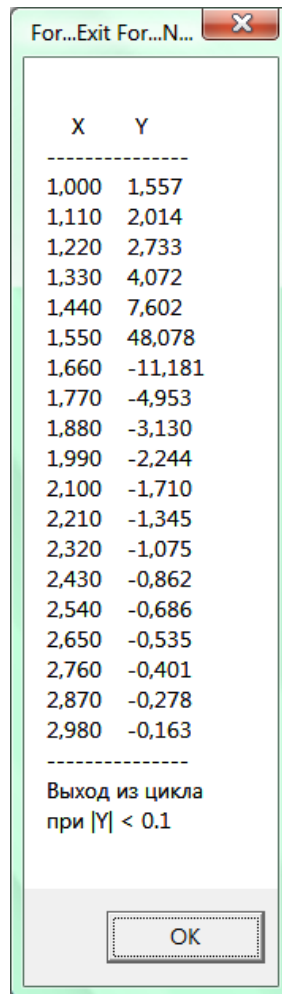


Рисунок 2. *For...Exit For...Next*

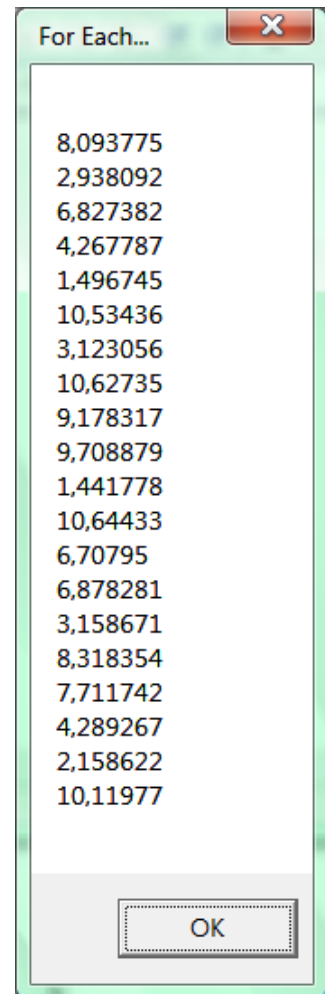


Рисунок 3. Цикл
For Each

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):

```

Dim x(19)
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 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)	$S = \sum_{i=1}^n x_i \cdot x_j$	2)	$y_i = x_i \sum_{i=1}^n x_i$
3)	$c_i = a_i \cdot b_i$	4)	$z_i = \frac{(n-i) \cdot x_i}{y_i}$
5)	$z_i = \frac{x_i}{x_{\max} - x_{\min}}$ x_{\max}, x_{\min} - максимальное и минимальное значения x	6)	$y_i = \frac{x_i}{x_{cp}}$ $x_{cp} = \sum_{i=1}^n x_i / n$
7)	$S = \sum_{i=1}^n x_i / i$	8)	$y_i = \frac{x_i}{\sum_{i=1}^n x_i}$
9)	$y_i = ax_i^2 + bx_i + c$	10)	$S = \sum_{i=1}^{n-1} \frac{x_i}{x_{i+1}}$

Лабораторная работа № 9. 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 1.

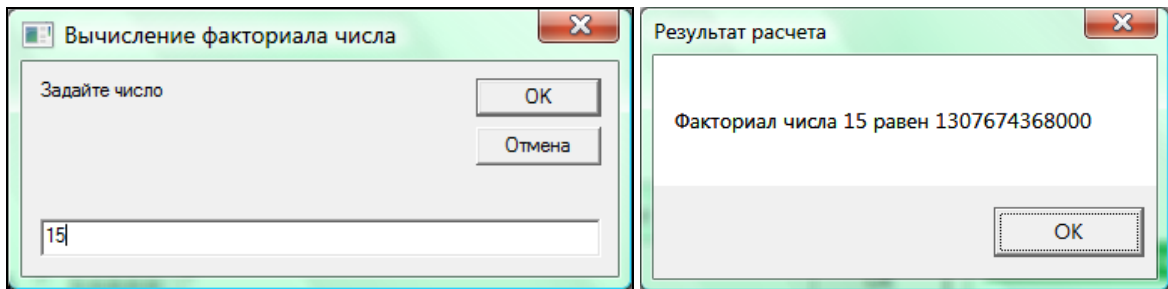


Рисунок 1. Результат выполнения программы с использованием процедуры Sub

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 1).

Задание

- a) Выполните задания предыдущей лабораторной работы (№ 8) 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 № 7 с заданием погрешности вычисления в окне **InputBox** и использованием функции пользователя для расчета числа π 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 № 10. Работа с числовой информацией

При выполнении математических вычислений используются символы математических операций:

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

Математических функций 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.

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 π (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):

```

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

```

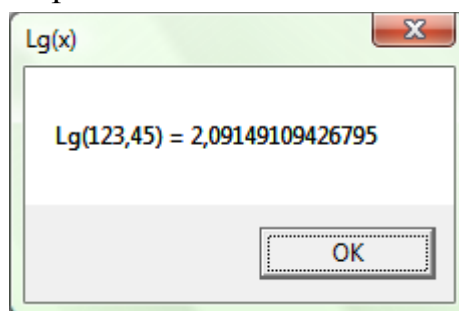


Рисунок. Вычисление логарифма $Lg(x)$

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)	$y = \sqrt[3]{3 + \sqrt[3]{3 + \frac{\operatorname{tg}^2(a)}{32 + b}}} + \operatorname{lg} \frac{a^3}{a + b + 2c}$	3,1415	96,6	18,4	1,00
7)	$y = \frac{\operatorname{tg}^2(a) - \operatorname{tg}^2(2b + c)}{\sqrt{34 - \frac{\operatorname{lg}(a)}{a + \operatorname{lg}(2b + c)}}}$	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)	$y = \operatorname{tg} \frac{\operatorname{lg}^2(a + b + c) + \sin(a^2)}{\sqrt{2 + \frac{14 - b}{2c}}}$	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

Приложение. Математические функции языка

Синтаксис функции	Описание
Abs (числ.выраж.)	Абсолютная величина числового выражения
Array (сп.эл.массива)	Создание массива из списка
Atn (числ.выраж.)	Арктангенс, числовое выражение в радианах
Cbool (числ.выраж.)	Возвращает True для ненулевого числового выражения, False для 0, ошибка для нечислового значения
CByte (числ.выраж.)	Преобразование в субтип Byte
CCur (числ.выраж.)	Преобразование в субтип Currency
CDBl (числ.выраж.)	Преобразование в субтип Double
CInt (числ.выраж.)	Преобразование в субтип Integer
CLng (числ.выраж.)	Преобразование в субтип Long
Cos (числ.выраж.)	Косинус, числовое выражение в радианах
CSng (числ.выраж.)	Преобразование в субтип Single
CStr (числ.выраж.)	Преобразование в субтип Строка
Exp (числ.выраж.)	Функция экспонента $\text{Exp}(X)$ или e^x
Eval (выражение)	Вычисляет выражение и возвращает его результат
Fix (числ.выраж.)	Целая часть числа
Int (числ.выраж.)	Целая часть числа
FormatCurrency (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
FormatNumber (числ.выраж. [, дес.зн. [, вед.ноль [, исп.() для отр.чисел [, исп.разд.]]])	Форматирование числ.выраж. в заданный формат
FormatPercent (числ.выраж. [, дес.зн. [, вед.ноль [, исп.() для отр.чисел [, исп.разд.]]])	Форматирование числ.выраж. в заданный процентный формат (/100) с символом %
Hex (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
Lbound (имя_массива[, размерность])	Наименьшее возможное значение индекса в массиве.. Если размерность массива не указан, он одномерный
Log (числ.выраж.)	Натуральный логарифм выражения. Если нужно вычислить логарифм с основанием n , следует использовать выражение: $\text{Log}_n(x) = \text{Log}(x) / \text{Log}(n)$
(числ.выраж.) Mod (числ.выраж.)	Остаток целочисленного деления. Перед делением результаты численных выражений

	округляются до целых чисел
Oct (число)	Строка, представляющая восьмеричную величину числа
Randomize	Инициализация генератора случайных чисел
Rnd [(числ.выраж.)]	Случайное число большее или равное нулю, но меньшее 1.
RGB (red, green, blue)	Преобразование цветового значения, где каждый из трех цветов в диапазоне от 0 до 255 в одно число, рассчитанное по формуле: $CLng(red + (green * 256) + (blue * 65536))$
Round (числ.выраж. [, дес.зн.])	Число, округленное до заданного количества десятичных знаков. Если дес.зн. не указано, его значение равно 0
Sgn (числ.выраж.)	Знак числа (>0 – 1, 0 – 0, <0 – -1)
Sin (числ.выраж.)	Синус угла, выраженного в радианах
Sqr (числ.выраж.)	Квадратный корень числа
Tan (числ.выраж.)	Тангенс угла, выраженного в радианах
TypeName (varИмя)	Название типа переменной (см. далее для VarType)
Ubound (имя_массива[, размерность])	Наибольшее возможное значение индекса в массиве. Если размерность массива не указан, он одномерный
VarType (varИмя)	Числовое обозначение типа переменной: 0 vbEmpty 1 vbNull 2 vbInteger 3 vbLong 4 vbSingle 5 vbDouble 6 vbCurrency 7 vbDate 8 vbСтрока 9 vbObject 10 vbError 11 vbBoolean 12 vbVariant 13 vbDataObject 17 vbByte 19 vbArray

Лабораторная работа № 11. Работа со строковой информацией

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

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:

- присвоить значение переменной **FIO**
FIO = "Иванов Петр Сидорович"
- написать **FIO** прописными буквами
FIO_p = Ucase(FIO)
- 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"
- получить строку – инициалы и фамилия
F4 = Left(F2, 1) & ". " & Left(F3, 1) & ". " & F1
'F4 = "П. С. Иванов"
- 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 1 показаны два примера выполнения программы, содержащей описанные выше операции работы со строками.

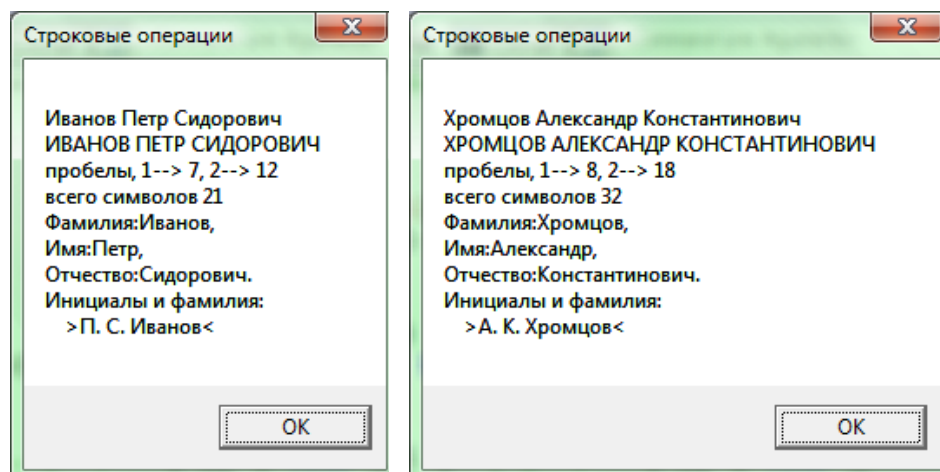


Рисунок 1. Использование функций для обработки строковой информации

Задания

Напишите программу для своего варианта задания. Исходные данные и результаты показать в окне сообщений. Описание функций для работы со строками см. в Приложении. 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 «e» и «a» 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 «e» на «E» и «o» на «O», 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**.

Приложение. Строковые функции языка

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

Лабораторная работа № 12. 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-мм-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 1):

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 & vbLf & _  
      "Дата 2: " & Dt2 & vbLf & _  
      "лет (yyyy) " & DateDiff("yyyy",dt1, dt2) & vbLf & _  
      "кварталов (q) " & DateDiff("q", dt1,dt2) & vbLf & _  
      "месяцев (m) " & DateDiff("m", dt1,dt2) & vbLf & _  
      "недель (ww) " & DateDiff("ww",dt1,dt2) & vbLf & _  
      "дней года (y) " & DateDiff("y", dt1,dt2) & vbLf & _  
      "дней (d) " & DateDiff("d", dt1,dt2) & vbLf & _  
      "часов (h) " & DateDiff("h", dt1,dt2) & vbLf & _  
      "минут (n) " & DateDiff("n", dt1,dt2) & vbLf & _  
      "секунд (s) " & DateDiff("s", dt1,dt2),, _  
      "Функция DateDiff"
```

Результат работы данной программы показан на рисунке 1.

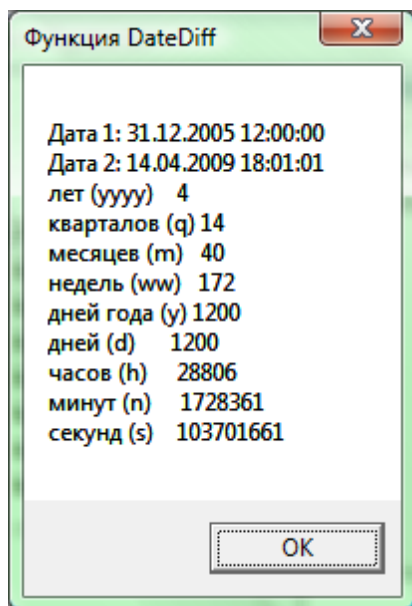


Рисунок 1. Использование функции **DateDiff**

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 2):

```
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 3):

```
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"
```

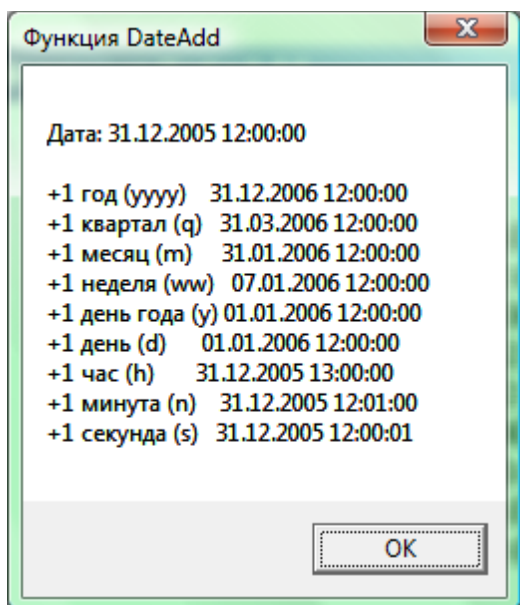


Рисунок 2. Использование функции *DateAdd*



Рисунок 3. 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 2).

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.

Приложение 1. Функции работы с датой и временем

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

Приложение 2. 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.

Лабораторная работа № 13. 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** ($L1 = \text{True}$ или $L1 = \text{False}$), 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 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 1.

Краткое описание логических операций (здесь и в таблице 1 **E1** и **E2** – логические выражения.):

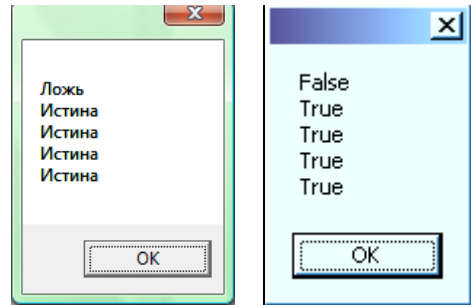


Рисунок 1. Результат вычисления логических функций в русской и английской версиях Windows

- **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**).

Полное описание результатов вычисления логических операций приведено в таблице 1.

Таблица 1. Результаты логических операций

E1	E2	E1 And E2	E1 Or E2	E1 Imp E2	E1 Xor E2
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) **"A" > "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))

Приложение. Логические функции и операторы языка

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