ГБПОУ «Дзержинский педагогический колледж»

Рекомендации по выполнению

лабораторных работ по МДК 01.01

«Системное программирование»

Дзержинск - 2014

|  |  |
| --- | --- |
| Одобрен ПЦК специальности «Информатика»  Протокол № \_\_\_\_\_\_\_\_\_\_\_  от «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_201 г.  Председатель ПЦК\_\_\_\_\_\_\_\_\_\_\_\_Семенова Н.В. |  |
| Составители:  Руденко Н.А., преподаватель информатики  ГБПОУ ДПК  Семенова Н.В., преподаватель информатики  ГБПОУ ДПК |  |

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
|  | Стр. |
| Лабораторная работа №1……………………………………………………………………… | 4 |
| Лабораторная работа №2……………………………………………………………………… | 5 |
| Лабораторная работа №3……………………………………………………………………… | 6 |
| Лабораторная работа №4……………………………………………………………………… | 10 |
| Лабораторная работа №5……………………………………………………………………… | 14 |

***Лабораторная работа №1***

***«ПРИЕМЫ РАБОТЫ С КОМАНДНОЙ СТРОКОЙ И  
 ПЕРЕМЕННЫМИ СРЕДЫ WINDOWS» (2ч.)***

***Цель работы:*** *познакомиться с параметрами командной строки и переменными среды Windows.*

Стандарты языка C++ пакета С++Builder позволяют создавать программное обеспечение не только прикладного, но и системного уровня. Например, возможна обработка параметров командной строки.

Число параметров командной строки может быть прочитано из переменной **ParamCount**, причем доступ к i-тому параметру командной строки осуществляется как **ParamStr(i)**, где **i** изменяется от **0** до **ParamCount-1**.

Нижеследующий фрагмент кода заносит в компонент Memo1 содержимое параметров командной строки, а в компонент Memo2 содержимое переменных среды Windows:

Memo1->Clear(); // очистка Memo1

// заполнение списка Memo1 параметрами командной строки

for (i=0; i <= ParamCount(); i++)

Memo1->Lines->Add(ParamStr(i));

Memo2->Clear();// очистка Memo2

// заполнение списка Memo2 переменными среды Windows

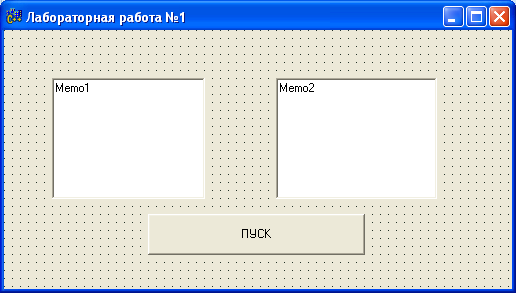
i=0;

while (\_environ[i])

Memo2->Lines->Add(\_environ[i++]);

**Задание.**

1. **Программным путем получите информацию о параметрах командной строки и переменных среды Windows. Для этого в среде С++Builder создайте приложение следующего вида:**

****

1. **Раскройте значение переменных среды Windows.**

Требования к оформлению работы

По каждой лабораторной работе составляется отчет, который должен содержать:

- титульный лист;

- название и цель работы;

- лабораторное задание;

- распечатку формы до запуска приложения и с результатами выполнения приложения;

- программный код с пояснениями;

- выводы, которые должны доказывать или оценивать правильность составленной программы или объяснять допущенные ошибки.

**Лабораторная работа №2. Работа с операционной системой MS-DOS**

Цель: научиться работать с командной строкой

|  |  |
| --- | --- |
| Задание | Выполнение |
| Запустить командную строку | Пуск \ Программы \ Стандартные \ Коммандная строка |
| Перейти на диск D | D: |
| Выполнить команду просмотра логического устройства | dir |
| В корневом каталоге диска создайте директорию и назовите своей фамилией | md IVANOV |
| Проверьте наличие созданного каталога | Dir |
| Сделать каталог с фамилией текущим | cd IVANOV |
| Создать в каталоге с фамилией каталог с номером лабораторной работы | md LAB1 |
| Сделать каталог LAB1 текущим | cd LAB1 |
| Создать в текущем каталоге файл с расширением .txt и именем, соответствующим вашему реальному. Ввести фамилию, дату выполнения работы. | copy con IVAN.txt  IVANOV 1.09.2006  Нажать F6 для выхода из режима редактирования. |
| Создать в текущем каталоге файл text1.txt. Ввести «Я изучаю DOS» | copy con text1.txt  Я изучаю DOS F6 |
| Создать в текущем каталоге каталог с именем COPYTEST | md COPYTEST |
| Копировать файл text1.txt в каталог COPYTEST | copy text1.txt D:\IVANOV\ LAB1\COPYTEST |
| Копировать файл с именем в каталог COPYTEST с переименованием на PETR.txt | copy IVAN.txt D:\IVANOV  \LAB1\COPYTEST\PETR.txt |
| Переместиться на уровень выше | cd.. |
| Выполнить команду просмотра каталога | Dir |
|  |  |
| Переместить файл text1.txt в каталог с фамилией | move text1.txt D:\IVANOV |
| Переместить файл с именем в каталог с фамилией с переименованием на MARIA.txt | move IVAN.txt D:\IVANOV\  MARIA.txt |
| Проверить наличие файла в каталоге IVANOV при помощи команды dir | |

**Лабораторная работа №3. Использование функций API**

API (Application Programming Interface) – это набор функций, которые операционная система предоставляет программисту. Эти функции находятся в стандартных динамически компонуемых библиотеках (DLL-файлы), которые содержатся в системном каталоге. Использование функций API позволяет достаточно просто выполнять различного рода задачи.

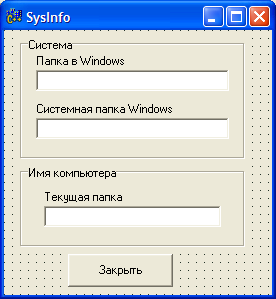
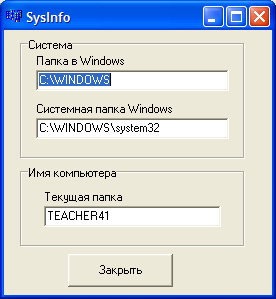
**Информация о системе**

Создайте приложение, в котором будет выводиться информация о системе, используя следующие функции:

* GetWindowsDirectory — возвращает путь к каталогу Windows;
* GetSystemDirectory — возвращает путь к системному каталогу Windows;
* GetComputerName — возвращает имя компьютера.

Разместите на форме следующие объекты и задайте им соответствующие свойства:

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Firm1 | Caption | SysInfo |
| GroupBox1 | Caption | Система |
| GroupBox2 | Caption | Имя компьютера |
| LabeledEdit1 (вкладка Additional) | Text |  |
| EditLabel-> Caption | Папка в Windows |
| LabeledEdit2 | Text |  |
| EditLabel-> Caption | Системная папка Windows |
| LabeledEdit3 | Text |  |
| EditLabel-> Caption | Текущая папка |

Программный код:

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

char WindowsDirectory[MAX\_PATH];GetWindowsDirectory(WindowsDirectory, MAX\_PATH);

LabeledEdit1->Text=WindowsDirectory;char SystemDirectory[MAX\_PATH];

GetSystemDirectory(SystemDirectory, MAX\_PATH);LabeledEdit2->Text=SystemDirectory;

unsigned long Size = MAX\_COMPUTERNAME\_LENGTH + 1;char \*Buffer = new char[Size];

GetComputerName(Buffer, &Size);LabeledEdit3->Text=Buffer;

delete [] Buffer;}

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{Close();}

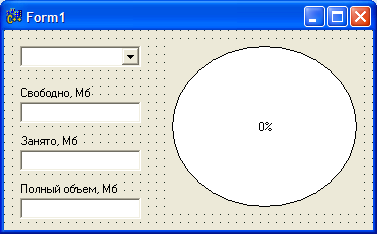
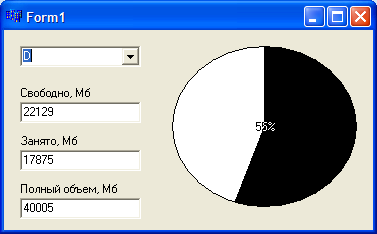
**Информация о дисках**

Создайте приложение, в котором будет выводиться системная информация о дисках, используя следующие функции:

* DiskSize — возвращает размер диска в байтах. В качестве входного параметра используется целое число, которое означает: 0 — текущий диск, 1 — А, 2 — В и т
* DiskFree — возвращает размер в байтах свободного пространства на диске.

Разместите на форме следующие объекты и задайте им соответствующие свойства:

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Form1 | Caption | hddInfo |
| ComboBox1 | Text |  |
| Items | A B C D E (ввести через Enter) |
| LabeledEdit1 (вкладка Additional) | Text |  |
| EditLabel-> Caption | Свободно, Мб |
| LabeledEdit2 | Text |  |
| EditLabel-> Caption | Занято, Мб |
| LabeledEdit3 | Text |  |
| EditLabel-> Caption | Полный объем, Мб |
| CGauge1 (вкладка Samples) | Kind | gkPie |
| BorderStyle | bsNone |

Программный код:

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#pragma package(smart\_init)

#pragma link "CGAUGES"

#pragma resource "\*.dfm"

TForm1 \*Form1;

\_\_int64 Size;

\_\_int64 FreeSpace;

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{ }

void \_\_fastcall TForm1::ComboBox1Change(TObject \*Sender)

{

if (ComboBox1->ItemIndex>-1)

{

Size=DiskSize(ComboBox1->ItemIndex+1);

FreeSpace = DiskFree(ComboBox1->ItemIndex+1);

if (Size>0)

{ LabeledEdit3->Text=IntToStr(Size/1024/1024);

LabeledEdit1->Text=IntToStr(FreeSpace/1024/1024);

LabeledEdit2->Text=IntToStr((Size-FreeSpace)/1024/1024);

CGauge1->Progress= FreeSpace\*100/Size;

} else

{

ShowMessage("Диска нет! \n Вставьте другой диск.");} } }

**Дополнительное задание**

Измените приложение так, чтобы информация о дисках выводилась в Гигабайтах

**Лабораторная работа №4**

**Копирование файлов с использованием Win32**

Последовательная обработка файлов – самая простая, наиболее обычная и наиболее необходимая функция любой файловой системы. Копирование файлов, часто с обновлением, и объединением отсортированных файлов – распространенные формы последовательной обработки. Простое копирование позволяет определить сильные и слабые стороны разных систем и перейти к Win32.

Цель работы – изучить копирование файлов с использованием стандартной библиотеки на языке С, с использованием интерфейса программирования приложений (API) Win32, а также с использованием функции-полуфабриката Win32.

**1. Копирование файлов с использованием библиотеки С. Программа Срс**.

Обращение к программе производится из командной строки: **Срс файл1, файл2**

#include <vcl.h>

#pragma hdrstop

#pragma argsused

#include <stdio.h>

#include <errno.h>

#define BUF\_SIZE 256

int main (int argc, char \*argv [])

{

FILE \*in\_file, \*out\_file; char rec

[BUF\_SIZE]; size\_t bytes\_in,

bytes\_out;

if (argc != 3)

{printf("Использование: Cрc file1 File2\n");return 1;}

in\_file = fopen (argv [1],"rb");

if (in\_file == NULL)

{perror (argv [1] ) ;return 2;}

out\_file = fopen (argv [2], "wb");

if (out\_file == NULL)

{perror (argv [2] ) ;return 3;}

/\* Обрабатываем входной файл по одной записи. \*/

while ( (bytes\_in = fread (rec, 1, BUF\_SIZE, in\_file) ) > 0)

{bytes\_out = fwrite (rec, 1, bytes\_in, out\_file) ;

if (bytes\_out != bytes\_in) {perror ("Неисправимая ошибка записи.");return 4;}}

fclose (in\_file);

fclose (out\_file);}

**2. Копирование файлов с использованием API Win32.** **Программа Сpw.**

Обращение к программе производится из командной строки: **Cpw файл1, файл2**

#include <vcl.h>

#pragma hdrstop

#pragma argsused

#include <windows.h>

#include <stdio.h>

#define BUF\_SIZE 256

int main (int argc, LPTSTR argv [])

{

HANDLE hIn, hOut;

DWORD nIn, nOut;

CHAR Buffer [BUF\_SIZE];

if (argc != 3)

{printf ("Использование: CpW file1 File2\n");

return 1;}

hIn = CreateFile (argv [1], GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

if (hIn==INVALID\_HANDLE\_VALUE)

{printf ("Нельзя открыть входной файл. Ошибка: %х\n", GetLastError ());return 2; }

hOut = CreateFile (argv [2], GENERIC\_WRITE, 0, NULL,

CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut==INVALID\_HANDLE\_VALUE)

{printf ("Нельзя открыть выходной файл. Ошибка: %х\n", GetLastError () ) ;return 3;}

while (ReadFile (hIn, Buffer, BUF\_SIZE, &nIn, NULL) && nIn > 0)

{WriteFile (hOut, Buffer, nIn, &nOut, NULL);

if (nIn != nOut)

{printf ("Неисправимая ошибка записи: %x\n", GetLastError ());

return 4;}}

CloseHandle (hIn);

CloseHandle (hOut);return 0;}

# Копирование файлов с использованием функции полуфабриката Win32. Программа cpСF.

Win32 содержит множество функций-полуфабрикатов, которые объединяют несколько функций для выполнения часто встречающихся задач. В некоторых случаях эти функции-полуфабрикаты могут улучшить быстродействие. Например, CopyFile значительно упрощает программу. В частности, здесь совершенно не нужно искать оптимальный размер буфера, который в двух предыдущих программах был произвольно задан в 256 байт.

Обращение производится из командной строки: сpCF файл1 файл2

#include <vcl.h>

#pragma hdrstop

#pragma argsused

#include <windows.h>

#include <stdio.h>

int main (int argc, LPTSTR argv [])

{

if (argc != 3)

{printf ("Использование: cpCF file1 file2\n");return 1; }

if (!CopyFile (argv [1], argv [2], FALSE))

{printf ("Ошибка CopyFile: %x\n", GetLastError ());return 2; } return 0;}

Приведенные программы копирования файлов демонстрируют многочисленные различия между библиотекой С и Win32. Примеры с использованием Win32 демонстрируют стиль программирования и соглашения Win32, но дают лишь первое представление о функциональных возможностях, доступных в Win32.

**4.Содержание работы**

1. Изучить программы копирования файлов.
2. Последовательно набрать и отладить программы копирования файлов в среде C++ Builder6.
3. Выполнить задание по копированию файла 1 в файл 2, выбрав произвольные имена файлов.

Программы запустить из командной строки.

1. Подготовить отчет по выполненной работе. Листинги программ записать в тетрадь.
2. В отчете написать название, цель работы и ответить на контрольные вопросы.

**5.Контрольные вопросы**

1. Что такое API Win32?

2. Какие операционные системы обслуживает API Win 32?

3.Какие особенности имеет API Win 32?

4. Какие преимущества программирования дает API 32?

5. Какой основной тип переменных используется в Win 32?

6. Для управления каких систем могут быть написаны программы с использованием Win32?

7. Что означает строка int main (int argc, LPTSTR argv [])?

8. Поясните, какую функцию выполняет данный оператор: hIn = CreateFile (argv [1], GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

9. Поясните, какую функцию выполняет данный оператор: hOut = CreateFile (argv [2], GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

10.Поясните, какую функцию выполняет данный оператор: while (ReadFile (hIn, Buffer, BUF\_SIZE, &nIn, NULL) && nIn > 0)

11. Поясните, какую функцию выполняет данный оператор: WriteFile (hOut, Buffer, nIn, &nOut, NULL);

12. Поясните, какую функцию выполняет данный оператор: if (!СopyFile (argv [1], argv [2], FALSE)).

**Лабораторная работа № 5**. ***Работа с накопителями информации.***

***Цель работы:*** Ознакомление с возможностями для работы с накопителями информации в среде Borland C++ Builder 6.

Основные функции для работы с накопителями информации:

***GetLogicalDrives: DWORD; – ф***ункция возвращает битовую маску, в которой хранится информация о доступных накопителях: наличие или отсутствие накопителя. То есть каждый бит отвечает за присутствие определённого логического диска. Либо, если бит не установлен, то это указывает на отсутствие логического диска. DWORD имеет размер 32 бита, этого вполне достаточно, чтобы уместить весь английский алфавит, который содержит 26 букв. Например: нулевой бит это информация о накопителе 'A', первый бит – информация о накопителе 'B', второй бит это информация о накопителе 'C' и.т.д. В случае ошибки функция возвращает нуль.

***GetLogicalDriveStrings(nBufferLength: DWORD; lpBuffer: PAnsiChar): DWORD; – ф***ункция записывает специальную переменную ***lpBuffer*** имена присутствующих в системе накопителей.

Параметры:

***nBufferLength:*** определяет максимальный размер в символах, буфера указанного в парметре ***lpBuffer***. Этот размер не включает завершающего нулевого символа.

***lpBuffer***: Указатель на буфер, в которой записываются имена присутсвующих в системе накопителей в следующем формате: 'C:\nullD:\null<null>'. Т.е. на каждый диск отводится по 4 байта.

Возвращаемые значения

Если функция выполнилась без ошибок, то она возвращает длину в символах буфера, не включая завершающего нулевого символа. В случае ошибки возвращает нуль.

***GetDriveType(lpRootPathName: PChar): UINT;*** – функция возвращает тип носителя: сменный, фиксированный, CD-ROM, диск RAM, или сетевой диск.

Параметры:

***lpRootPathName:*** строка определяющая корневой каталог устройства, тип которого надо определить(например 'A:\'– для дисковода). Если ***lpRootPathName*** равен ***NULL***, то возвращается тип текущего накопителя.

Возвращаемые значения

Возвращаемое значение определяет тип накопителя. Может быть:

**0** – Тип накопителя не определен.

**1** – Корневой директорий не существует.

***DRIVE\_REMOVABLE*** – Накопитель может удаляться с накопителя.

***DRIVE\_FIXED*** – Фиксированный диск(не может быть удален).

***DRIVE\_REMOTE*** – Удаленный накопитель(сетевой диск).

***DRIVE\_CDROM*** – диск CD-ROM.

***DRIVE\_RAMDISK*** – Накопитель является виртуальным RAM-диском.

***GetDiskFreeSpaceEx(lpDirectoryName: PChar; &lpFreeBytesAvailableToCaller, &lpTotalNumberOfBytes, &lpTotalNumberOfFreeBytes: ULARGE\_INTEGER): BOOL;*** – функция получает информацию о дисковом пространстве доступного на накопителе: общий объем пространства, общий объем свободного пространства, и общий объем свободного пространства доступного для пользователя.

Параметры:

***lpDirectoryName***: строка определяющая корневой каталог устройства, размер которого надо определить(например 'A:\'– для дисковода). Если ***lpRootPathName*** равен ***NULL***, то выбирается текущий накопитель.

***lpFreeBytesAvailableToCaller***: общий объём свободного пространства доступного для пользователя.

***lpTotalNumberOfBytes***: общее число байтов на диске.

***lpTotalNumberOfFreeBytes***: общее число свободный байтов на диске.

Возвращаемые значения

Если функция выполнилась без ошибки, то она вернёт не нулевое значение.

***GetVolumeInformation(lpRootPathName: PChar; lpVolumeNameBuffer: PChar; nVolumeNameSize: DWORD; lpVolumeSerialNumber: PDWORD; &lpMaximumComponentLength, &lpFileSystemFlags: DWORD; &lpFileSystemNameBuffer: PChar; &nFileSystemNameSize: DWORD): BOOL; –*** возвращает информацию о файловой системе и о накопители информации.

Параметры:

l***pRootPathName***: строка определяющая корневой каталог устройства (например 'A:\'– для дисковода). Если ***lpRootPathName*** равен ***NULL***, то выбирается текущий накопитель.

***lpVolumeNameBuffer***: имя метки тома.

***nVolumeNameSize:*** Содержит длину в символах имени метки тома.

***lpVolumeSerialNumber:*** Указатель на переменную, которая содержит серийный номер тома.

***lpMaximumComponentLength:*** Указатель на переменную, которая возвращает максимальную длину допустимого имени файла на диске.

lpFileSystemFlags: Указатель на переменную, которая возвращает флаги связанные с текущей файловой системой. Этот параметр может быть комбинацией любых флагов кроме ***FS\_FILE\_COMPRESSION*** и ***FS\_VOL\_IS\_COMPRESSED***, которые взаимно исключены:

***FS\_CASE\_IS\_PRESERVED*** –файловая система сохраняет имя файла на диске.

***FS\_CASE\_SENSITIVE*** –файловая система поддерживает контекстно-зависимые имена файлов.

***FS\_UNICODE\_STORED\_ON\_DISK*** –файловая система поддерживает имена файлов в Unicode.

***FS\_PERSISTENT\_ACLS*** –файловая система сохраняет и осуществляет ACL. Например, NTFS сохраняет и осуществляет ACL,а FAT нет.

***FS\_FILE\_COMPRESSION*** – Файловая система поддерживает сжатие файлов.

***FS\_VOL\_IS\_COMPRESSED*** – Том является сжатым; например DoubleSpace.

***lpFileSystemNameBuffer:*** Указатель на переменную, которая возвращает имя файловой системы (как например, FAT или NTFS).

***nFileSystemNameSize:*** Параметр определяет длину переменной имени файловой системы в символах.Возвращаемые значения: если функция выполнилась без ошибки, то она вернёт не нулевое значение.

**Практическая часть**

Запустите *C++Builder 6*. Сохраните проект в новой отдельной папке.

Присвойте основной форме заголовок, в качестве имени которого используйте название дисциплины, номер работы и варианта и свою фамилию.

Установите для формы ограничения изменения размеров максимум 800х600, минимум 600х400. Для этого используйте свойства **MaxHeight**, **MaxWidth** и **MinHeight**, **MinWidth**. Свойства находятся в свойстве **Constraints,** для доступа к ним нажмите на знак плюса напротив свойства **Constraints.**

Установите позицию показа формы по центру экрана (свойство **Position** =*poDesktopCenter*).

Выведите список всех логических дисков используемых системой, сначала с помощью функции ***GetLogicalDrives()***, а затем с помощью – ***GetLogicalDriveStrings()***. Сравните полученные результаты.

Для каждого логического диска определите:

его тип с помощью функции ***GetDriveType***;

размер диска, количество занятого пространства и количество свободного места на диске, количество места доступного пользователю, с помощью функции ***GetDiskFreeSpaceEx***;

тип файловой системы, максимальный размер имени, метку тома, атрибуты файловой системы диска с помощью функции ***GetVolumeInformation***.

Добавьте на форму текстовой поле и кнопку для выполнения индивидуального задания, согласно номеру варианта

|  |  |
| --- | --- |
| №п/п | Вариант задания |
| 1 | При нажатии кнопки должен создаваться каталог путь, к которому указан в текстовом поле. |
| 2 | При нажатии кнопки должен создаваться файл путь, к которому указан в текстовом поле. |
| 3 | Определить дату создания файла путь, к которому указан в текстовом поле. |
| 4 | Определить дату последнего изменения файла путь, к которому указан в текстовом поле. |
| 5 | Определить дату последнего открытия файла путь, к которому указан в текстовом поле. |
| 6 | При нажатии кнопки вывести в текстовое поле текущий каталог |
| 7 | При нажатии кнопки должен удаляться каталог путь, к которому указан в текстовом поле. |
| 8 | При нажатии кнопки должен удаляться файл путь, к которому указан в текстовом поле. |
| 9 | Определить размер файла путь, к которому указан в текстовом поле. |
| 10 | Определить архивный ли файл путь, к которому указан в текстовом поле. |
| 11 | Определить путь к системному каталогу Windows |
| 12 | Определить используется ли файл только для чтения, к которому указан в текстовом поле. |
| 13 | Установить для файла путь, к которому указан в текстовом поле атрибут только для чтения. |

Отчет сдайте преподавателю.