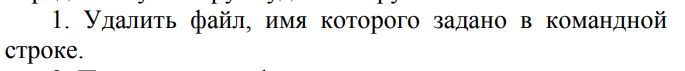
**Лабораторная работа 4**

**Студент:** Смирнов Константин ПКС-303

**Учитель:** Хусточка Алёна Витальевна



#include <stdio.h>

#include <Windows.h>

#include <iostream>

#include <string.h>

#include <filesystem>

#include <locale.h>

int wmain(int argc, wchar\_t\* argv[]) {

setlocale(LC\_ALL, "Rus");

DWORD bytesWritten;

char readBuffer[256];

DWORD bytesRead;

char buffer[256] = "new\_file.txt";

LPCWSTR path = argv[1];

printf("Файл: %s\n", buffer);

HANDLE file = CreateFile(path, GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, OPEN\_ALWAYS, NULL, NULL);

if (file == INVALID\_HANDLE\_VALUE) {

std::cout << "File not opened";

exit(EXIT\_FAILURE);

}

else {

printf("new\_file.txt was created.\n");

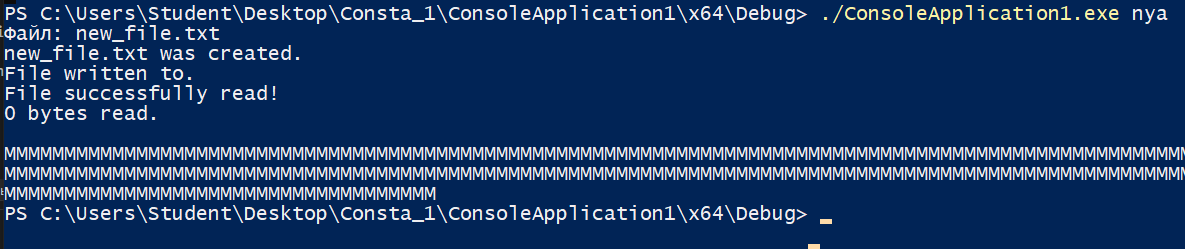
}

LPCWSTR path = argv[1];

DeleteFileW(path);

CloseHandle(file);

}



2)



#include <stdio.h>

#include <Windows.h>

#include <iostream>

#include <string.h>

#include <filesystem>

#include <locale.h>

int wmain(int argc, wchar\_t\* argv[]) {

setlocale(LC\_ALL, "Rus");

DWORD bytesWritten;

char readBuffer[256];

DWORD bytesRead;

char buffer[256] = "new\_file.txt";

LPCWSTR path = argv[1];

LPCWSTR path\_1 = argv[2];

// printf("Файл: %s\n", buffer);

HANDLE file = CreateFile(path, GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE, NULL, OPEN\_ALWAYS, NULL, NULL);

if (file == INVALID\_HANDLE\_VALUE) {

std::cout << "File not opened";

exit(EXIT\_FAILURE);

}

else {

std::cout << "file was created" << std::endl;

}

// Sleep(2000);

//\_wrename(argv[1], argv[2]);

if (MoveFileW(argv[1], argv[2]) == true)

{

std::cout << "file was renemed" << std::endl;

goto end;

}

else

{

std::cout << "file was not renemed" << std::endl;

goto end;

}

// Sleep(2000);

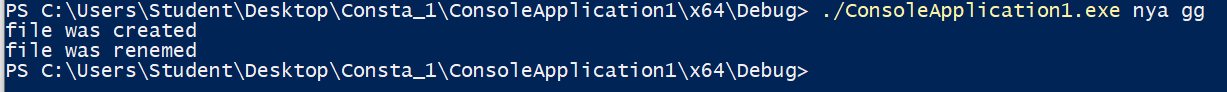
end:

DeleteFileW(path);

DeleteFileW(path\_1);

CloseHandle(file);

}



3)



#include <stdio.h>

#include <Windows.h>

#include <iostream>

#include <string.h>

#include <filesystem>

#include <locale.h>

#include <iostream>

int wmain(int argc, wchar\_t\* argv[]) {

setlocale(LC\_ALL, "Rus");

DWORD bytesWritten;

char readBuffer[256];

DWORD bytesRead;

char buffer[256] = "new\_file.txt";

LPCWSTR path = argv[1];

LPCWSTR path\_1 = argv[2]; //C:\Users\Student\Desktop\Consta\_1\ConsoleApplication1\x64\Debug

LPCWSTR path\_1 = argv[3]; // C:\Users\Student\Desktop\Consta\_1

// printf("Файл: %s\n", buffer);

HANDLE file = CreateFile(path, GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE, NULL, OPEN\_ALWAYS, NULL, NULL);

if (file == INVALID\_HANDLE\_VALUE) {

std::cout << "File not opened";

exit(EXIT\_FAILURE);

}

else {

std::cout << "file was created" << std::endl;

}

// Sleep(2000);

//\_wrename(argv[1], argv[2]);

if (CopyFile(argv[1], argv[2], true) == true)

{

std::cout << "file was copyed" << std::endl;

goto end;

}

else

{

std::cout << "file was not copyed" << std::endl;

goto end;

}

// Sleep(2000);

end:

DeleteFileW(path);

DeleteFileW(path\_1);

CloseHandle(file);

}

4. Вывести на экран размер файла, имя которого задано в командной строке.

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <filesystem>

#include <iostream>

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

HANDLE file =CreateFile(path,GENERIC\_READ,0,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL, NULL);

int size = GetFileSize(file, NULL);

std::cout<< "File size: " << size <<std::endl;

DWORD attr = GetFileAttributes(path);

printf("\nFile attributes: %u\n", attr);

CloseHandle(file);

}

5.Вывести на экран атрибуты файла, имя которого задано в командной строке.

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <filesystem>

#include <iostream>

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

HANDLE file = CreateFile(path, GENERIC\_READ,0,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL, NULL);

DWORD attr = GetFileAttributes(path);

std::cout<< "File attributes: " << attr <<std::endl;

CloseHandle(file);

}

6.Установить атрибут «только для чтения» файлу, имя которого задано в командной строке.

#include <Windows.h>

#include <stdio.h>

#include <filesystem>

#include <iostream>

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

SetFileAttributes(path, FILE\_ATTRIBUTE\_READONLY);

DWORD roattr = GetFileAttributes(path);

std::cout<< "\nFile attributes: “ << roattr <<std::endl;

CloseHandle(file);

}

7.Установить атрибут «скрытый» файлу, имя которого задано в командной строке.

#include <Windows.h>

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <filesystem>

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

SetFileAttributes(path, FILE\_ATTRIBUTE\_HIDDEN);

DWORD hiattr = GetFileAttributes(path);

std::cout << "File attributes: ", << hiattr <<std::endl;

}

8.Вывести на экран содержимое папки, имя которой задано в командной строке. Выводить только имена (и типы) файлов и папок.

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <filesystem

#include <iostream>

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

WIN32\_FIND\_DATAW wfd;

HANDLE const hFind = FindFirstFile(fpath, &wfd);

do

{

std::cout << &wfd.cFileName[0] <<std::endl;

}

while (NULL != FindNextFile(hFind, &wfd));

FindClose(hFind);}

9.Создать заданную папку в текущей. Имя папки задается в командной строке.

#include <Windows.h>

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <filesystem

int wmain(int argc, wchar\_t\* argv[]) {

LPCWSTR path = argv[1];

if (CreateDirectory(path, NULL))

std::cout <<"Make " << std::endl;

else{

std::cout "\Papka ne sozdalas” << std::endl;

}

}

10.Переместить файл из одной папки в другую. Имена файла и папок задаются в командной строке.

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <filesystem>

#include <iostream>

int wmain(int argc, wchar\_t\* argv[])

{

LPCWSTR path = argv[1];

LPCWSTR path1 = argv[2];

if (MoveFile(path, path1))

{

DeleteFile(path);

DeleteFile(path1);

std::cout << "File movied" <<std::endl;

}

else std::cout << "File not moved\n" << std::endl;

}

Отсортировать по возрастанию содержимое числового файла:

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

int main() {

HANDLE myFile = CreateFile(L"C://Users/Consta\_1/Desktop/new\_file.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, 0);

printf("Descriptor = %d", myFile);

DWORD myFileSize = GetFileSize(myFile, NULL);

printf("\nSize = %u", myFileSize);

HANDLE myFileMapping = CreateFileMapping(myFile, NULL, PAGE\_READWRITE, 0, 0, L"C://Users/Consta\_1/Desktop/MyFileMapping.txt");

printf("\nFile mapping = %d", myFileMapping);

LPVOID myFileMapView = MapViewOfFile(myFileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

printf("\nMap view = %d", myFileMapView);

printf("\n");

char\* myFileMemory = new char[myFileSize];

CopyMemory(myFileMemory, myFileMapView, myFileSize);

int i;

int j;

int temp;

for (i = 0; i < myFileSize-1; i++)

for (j = 1; j < myFileSize; j++)

{

if (myFileMemory[i] > myFileMemory[j])

{

temp = myFileMemory[i];

myFileMemory[i] = myFileMemory[j];

myFileMemory[j] = temp;

}

}

CopyMemory(myFileMapView, myFileMemory, myFileSize);

if (!UnmapViewOfFile(myFileMapView))

printf("\nError, memory not cleared");

else

printf("\nMemory cleared");

if (!CloseHandle(myFile))

printf("\nHandle not closed");

else

printf("\nHandle closed");

getch();

}

Упорядочить буквы по алфавиту наоборот:

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

int main() {

HANDLE myFile = CreateFile(L"C://Users/Consta\_1/Desktop/new\_file.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, 0);

printf("Descriptor = %d", myFile);

DWORD myFileSize = GetFileSize(myFile, NULL);

printf("\nSize = %u", myFileSize);

HANDLE myFileMapping = CreateFileMapping(myFile, NULL, PAGE\_READWRITE, 0, 0, L"C://Users/Consta\_1/Desktop/MyFileMapping.txt");

printf("\nFile mapping = %d", myFileMapping);

LPVOID myFileMapView = MapViewOfFile(myFileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

printf("\nMap view = %d", myFileMapView);

printf("\n");

char\* myFileMemory = new char[myFileSize];

CopyMemory(myFileMemory, myFileMapView, myFileSize);

char letter = myFileMemory[0];

for (int i = 0; i < myFileSize; i++)

for (int j = 1; j < myFileSize-1; j++)

{

if (myFileMemory[i] > myFileMemory[j])

{

letter = myFileMemory[i];

myFileMemory[i] = myFileMemory[j];

myFileMemory[j] = letter;

}

}

CopyMemory(myFileMapView, myFileMemory, myFileSize);

if (!UnmapViewOfFile(myFileMapView))

printf("\nError, memory not cleared");

else

printf("\nMemory cleared");

if (!CloseHandle(myFile))

printf("\nHandle not closed");

else

printf("\nHandle closed");

getch();

}

Посчитать количество цифр в файле:

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

int main() {

HANDLE myFile = CreateFile(L"C://Users/Consta\_1/Desktop/new\_file.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, 0);

printf("Descriptor = %d", myFile);

DWORD myFileSize = GetFileSize(myFile, NULL);

printf("\nSize = %u", myFileSize);

HANDLE myFileMapping = CreateFileMapping(myFile, NULL, PAGE\_READWRITE, 0, 0, L"C://Users/Consta\_1/Desktop/MyFileMapping.txt");

printf("\nFile mapping = %d", myFileMapping);

LPVOID myFileMapView = MapViewOfFile(myFileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

printf("\nMap view = %d", myFileMapView);

printf("\n");

char\* myFileMemory = new char[myFileSize];

CopyMemory(myFileMemory, myFileMapView, myFileSize);

int count = 0;

for (int i = 0; i < myFileSize; i++)

{

if ((myFileMemory[i] == '0') || (myFileMemory[i] == '1') || (myFileMemory[i] == '2') || (myFileMemory[i] == '3') || (myFileMemory[i] == '4') || (myFileMemory[i] == '5') || (myFileMemory[i] == '6') || (myFileMemory[i] == '7') || (myFileMemory[i] == '8') || (myFileMemory[i] == '9'))

count++;

}

printf("%d", count);

CopyMemory(myFileMapView, myFileMemory, myFileSize);

if (!UnmapViewOfFile(myFileMapView))

printf("\nError, memory not cleared");

else

printf("\nMemory cleared");

if (!CloseHandle(myFile))

printf("\nHandle not closed");

else

printf("\nHandle closed");

\_getch();

}

Удалить буквы из текста:

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

int main() {

HANDLE myFile = CreateFile(L"C://Users/Consta\_1/Desktop/new\_file.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, 0);

printf("Descriptor = %d", myFile);

DWORD myFileSize = GetFileSize(myFile, NULL);

printf("\nSize = %u", myFileSize);

HANDLE myFileMapping = CreateFileMapping(myFile, NULL, PAGE\_READWRITE, 0, 0, L"C://Users/Consta\_1/Desktop/MyFileMapping.txt");

printf("\nFile mapping = %d", myFileMapping);

LPVOID myFileMapView = MapViewOfFile(myFileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

printf("\nMap view = %d", myFileMapView);

printf("\n");

char\* myFileMemory = new char[myFileSize];

CopyMemory(myFileMemory, myFileMapView, myFileSize);

int count = 0;

char b = ' ';

for (int i = 0; i < myFileSize; i++)

{

if (myFileMemory[i] >= 'a' && myFileMemory[i] <= 'z')

myFileMemory[i] = b;

}

CopyMemory(myFileMapView, myFileMemory, myFileSize);

if (!UnmapViewOfFile(myFileMapView))

printf("\nError, memory not cleared");

else

printf("\nMemory cleared");

if (!CloseHandle(myFile))

printf("\nHandle not closed");

else

printf("\nHandle closed");

\_getch();

}

#include <stdio.h>

#include <Windows.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

int main() {

HANDLE myFile = CreateFile(L"C://Users/Consta\_1/Desktop/new\_file.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, 0);

printf("Descriptor = %d", myFile);

DWORD myFileSize = GetFileSize(myFile, NULL);

printf("\nSize = %u", myFileSize);

HANDLE myFileMapping = CreateFileMapping(myFile, NULL, PAGE\_READWRITE, 0, 0, L"C://Users/Consta\_1/Desktop/MyFileMapping.txt");

printf("\nFile mapping = %d", myFileMapping);

LPVOID myFileMapView = MapViewOfFile(myFileMapping, FILE\_MAP\_WRITE, 0, 0, 0);

printf("\nMap view = %d", myFileMapView);

printf("\n");

char\* myFileMemory = (char\*)myFileMapView;

int i;

int j;

int temp;

for (i = 0; i < myFileSize - 1; i++)

for (j = 1; j < myFileSize; j++)

{

if (myFileMemory[i] < myFileMemory[j])

{

temp = myFileMemory[i];

myFileMemory[i] = myFileMemory[j];

myFileMemory[j] = temp;

}

}

if (!UnmapViewOfFile(myFileMapView))

printf("\nError, memory not cleared");

else

printf("\nMemory cleared");

if (!CloseHandle(myFile))

printf("\nHandle not closed");

else

printf("\nHandle closed");

getch();

}

/