### МИНИСТРЕСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

### Национальный аэрокосмический университет им. Н.Е.Жуковского

### «Харьковский авиационный институт»

### Кафедра 503

### Лабораторная работа №2

### по курсу «СП»

### Выполнил: студент гр.535-А

### Курта Р. В.

### Проверил: ассистент

### Трубилко А.В.

### Харьков 2016

## Цель:

Изучение системных вызовов Win32 API работы с файлами.

Постановка задачи:

### Программа 1:

Написать программу, реализующую произвольный доступ к записям в файле двумя способами: с помощью указателя файла (file pointer).

Структура записи:

* номер записи;
* время создания записи (в формате FILETYME);
* текстовая строка заданной длины (80 символов);
* счетчик, показывающий, сколько раз запись изменялась.

Запись может быть пустая (инициализирована нулями).

В заголовке файла хранить количество непустых записей в файле и размер файла. Общее количество записей в файле задается из командной строки. Пользователь должен иметь возможность удалять и модифицировать существующие записи, обращаясь к ним по номеру. Интерфейс с пользователем реализуется на усмотрение студента.

### Программа 2:

Написать программу, реализующую функцию файлового менеджера. Программа должна выдавать приглашение на ввод команды. Поддерживаемые команды:

* Сменить директорию
* Распечатать директорию
* Скопировать файл
* Создать директорию
* Удалить файл (пустую директорию)
* Вывести подробную информацию о файле

**Задание 1**

**Код программы:**

// SP\_lab2\_part2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

struct head {

int count\_of\_structs;

unsigned int size;

};

struct body {

int num;

int count\_view;

char text[80];

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(0, "rus");

void checkError(DWORD dw);

DWORD dw;

DWORD lpDwNumOfBytes;

body\* heade;

head hed;

body\* get\_struct;

head header;

while(1==1)

{

HANDLE hFile = CreateFile(L"test.txt", GENERIC\_ALL, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, NULL, 0);

if (hFile == INVALID\_HANDLE\_VALUE)

{

HANDLE hFile = CreateFile(L"test.txt", GENERIC\_ALL, FILE\_SHARE\_READ, NULL, CREATE\_NEW, NULL, 0);

puts("ФАЙЛ УСПЕШНО СОЗДАН!");

puts("Введите число структур: ");

int count\_struct;

scanf("%d", &count\_struct);

hed.count\_of\_structs = count\_struct;

hed.size = sizeof(head) + sizeof(body)\*count\_struct;

WriteFile(hFile, &hed, sizeof(hed), &lpDwNumOfBytes, NULL);

heade = (body\*)malloc(sizeof(body) \* count\_struct);

for (int i = 0; i < count\_struct; i++) {

memset(heade[i].text, 0, sizeof(heade[i].text));

heade[i].num = i + 1;

heade[i].count\_view = 0;

strcpy(heade[i].text, "HELLO!");

WriteFile(hFile, &heade[i], sizeof(heade[i]), &lpDwNumOfBytes, NULL);

}

puts("Текущая структура:");

for (int i = 0; i < count\_struct; i++)

{

printf("%s\n", heade[i].text);

}

CloseHandle(hFile);

/\* dw = GetLastError();

checkError(dw);\*/

}

else {

ReadFile(hFile, &header, sizeof(head), &lpDwNumOfBytes, NULL);

get\_struct = (body\*)malloc(sizeof(body)\*header.count\_of\_structs);

for (int i = 0; i < header.count\_of\_structs; i++) {

if (ReadFile(hFile, &get\_struct[i], sizeof(get\_struct[i]), &lpDwNumOfBytes, NULL)) {

printf("Полученная структура номер %d\nКоличество изменений: %d\nТекст: %s\n", get\_struct[i].num, get\_struct[i].count\_view, get\_struct[i].text);

}

}

int chosen\_struct;

int case\_choise;

char text[80];

for(;;)

{

puts("Введите номер записи:");

scanf\_s("%i", &chosen\_struct);

if (chosen\_struct <= header.count\_of\_structs)

{

break;

}

else {

printf("Не больше числа структур: %d!\n", header.count\_of\_structs);

}

}

puts("Выберите операцию:\n1->изменить\n2->очистить\n3->выход");

scanf\_s("%i", &case\_choise);

DWORD offset = sizeof(head) + sizeof(body)\*(chosen\_struct - 1);

SetFilePointer(hFile, offset, 0, FILE\_BEGIN);

switch (case\_choise)

{

case 1: puts("Введите текст:");

scanf("%s", &text);

get\_struct[chosen\_struct - 1].count\_view++;

strcpy(get\_struct[chosen\_struct - 1].text, text);

WriteFile(hFile, &get\_struct[chosen\_struct - 1], sizeof(body), &lpDwNumOfBytes, NULL);

break;

case 2:

memset(get\_struct[chosen\_struct - 1].text, 0, sizeof(body));

get\_struct[chosen\_struct - 1].count\_view++;

printf("Строка структуры %d очищена...\n", chosen\_struct);

WriteFile(hFile, &get\_struct[chosen\_struct - 1], sizeof(body), &lpDwNumOfBytes, NULL);break;

case 3: return 0; break;

default:

printf("Нет такой команды!\n");

break;

}

CloseHandle(hFile);

}

}

}

void checkError(DWORD dw) {

LPVOID lpMsg;

FormatMessage(FORMAT\_MESSAGE\_ALLOCATE\_BUFFER |

FORMAT\_MESSAGE\_FROM\_SYSTEM |

FORMAT\_MESSAGE\_IGNORE\_INSERTS,

NULL,

dw,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&lpMsg,

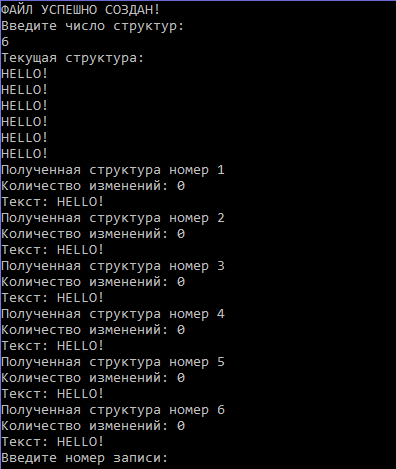
0,

NULL);

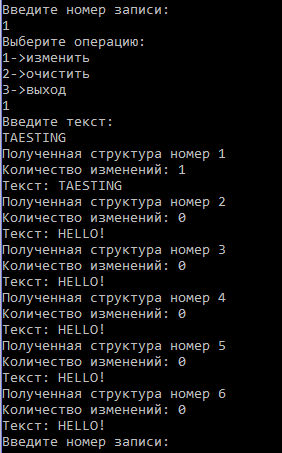
\_tprintf(TEXT("%s"), lpMsg);

}

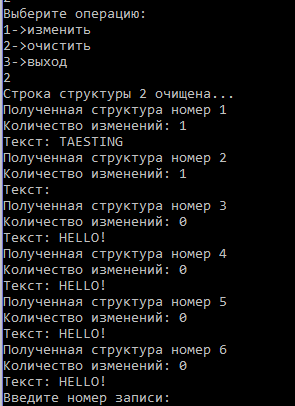
**Скриншоты:**



**Рисунок 1 ­– Автоматическое создание структуры при её отсутствии**



**Рисунок 2 – Изменение записи**



**Рисунок 3 – Очистка записи**

**Задание 2**

**Код программы:**

// SP\_lab2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <direct.h>

#define MAX\_LENGTH 100

void printDir();

void printFile();

bool changeWorkDir(char\* workDir, char\* changedDir, char\* root);

void copyFile(char\* workDir, char\* changedDir, char\* root);

void delFile(char\* workDir, char\* root);

void makeDir(char\* workDir, char\* root);

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

{

setlocale(0, "rus");

char command;

char root[MAX\_LENGTH];

GetCurrentDirectoryA(MAX\_LENGTH, root);

char workDir[MAX\_LENGTH], changedDir[MAX\_LENGTH];

strcpy(workDir, root);

puts("Выберите комманду:\n1->Сменить рабочую директорию\n2->Печать каталога\n3->Копировать файл\n4->Удалить файл\n5->Создать файл\n6->Информация о файле");

for (;;)

{

scanf("%s", &command);

switch (command)

{

case '1': changeWorkDir(workDir, changedDir, root); break;

case '2': printDir(); break;

case '3': copyFile(workDir, changedDir, root); break;

case '4': delFile(root , workDir); break;

case '5': makeDir(workDir, root); break;

case '6': printFile(); break;

case 'e': puts("Выход!"); exit(0); break;

default:

puts("Некоррентная комманда!"); break;

}

}

}

void printDir() {

wchar\_t dir\_path[MAX\_LENGTH] ;

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind;

LARGE\_INTEGER FileSize;

SYSTEMTIME st;

puts("Введите директорию: ");

\_tscanf(L"%s", &dir\_path);

hFind = FindFirstFile(dir\_path, &FindFileData);

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

\_tprintf(L"Нет такой директории!");

}

else {

puts("Содержимое каталога: ");

do

{

\_tprintf(L"%s", FindFileData.cFileName);

if (FindFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)

printf(" <=DIR=>");

else

{

printf(" <=FILE=> ");

FileSize.HighPart = FindFileData.nFileSizeHigh;

FileSize.LowPart = FindFileData.nFileSizeLow;

\_tprintf(L" <=Size: %ld=> ", FileSize);

}

FileTimeToSystemTime(&FindFileData.ftCreationTime, &st);

\_tprintf(L" <=Cteation date: %ld:%ld:%ld, %ld-%ld-%ld=>\n", st.wSecond, st.wMinute, st.wHour, st.wDay, st.wMonth, st.wYear);

}

while (FindNextFile(hFind, &FindFileData) != 0);

FindClose(hFind);

}

}

void printFile() {

wchar\_t dir\_path[MAX\_LENGTH];

wchar\_t file\_name[MAX\_LENGTH];

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind;

LARGE\_INTEGER FileSize;

SYSTEMTIME st;

puts("Введите каталог: ");

\_tscanf(L"%s", &dir\_path);

hFind = FindFirstFile(dir\_path, &FindFileData);

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

\_tprintf(L"Нет такой директории!");

}

else {

puts("Введите файл: ");

\_tscanf(L"%s", &file\_name);

bool tmp = false;

do

{

if (wcscmp(file\_name, FindFileData.cFileName) == 0)

{

tmp = true;

puts("Содержимое файла: ");

\_tprintf(L"%s", FindFileData.cFileName);

if (FindFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)

printf(" <=DIR=>");

else

{

printf(" <=FILE=> ");

FileSize.HighPart = FindFileData.nFileSizeHigh;

FileSize.LowPart = FindFileData.nFileSizeLow;

\_tprintf(L" <=Size: %ld=> ", FileSize);

}

FileTimeToSystemTime(&FindFileData.ftCreationTime, &st);

\_tprintf(L" <=Cteation date: %ld:%ld:%ld, %ld-%ld-%ld=>\n", st.wSecond, st.wMinute, st.wHour, st.wDay, st.wMonth, st.wYear);

}

} while (FindNextFile(hFind, &FindFileData) != 0);

if (!tmp) {

puts("Файл не найден.");

}

FindClose(hFind);

}

}

bool changeWorkDir(char\* workDir, char\* changedDir, char\* root) {

char str[MAX\_LENGTH];

printf("%s\n", root);

puts("Путь к новой директории: ");

scanf("%s", changedDir);

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

strcpy(str, workDir);

SetCurrentDirectoryA(changedDir);

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

if (strstr(workDir, root))

{

printf("%s\n", workDir);

return true;

}

else {

SetCurrentDirectoryA(str);

GetCurrentDirectoryA(MAX\_LENGTH, str);

printf("Нельзя выше!\n%s\n", str);

return false;

}

}

void copyFile(char\* workDir, char\* changedDir, char\* root) {

char str[MAX\_LENGTH];

printf("%s\n", root);

puts("Путь к фалу 1: ");

scanf("%s", changedDir);

char path\_one[100];

strcpy(path\_one,changedDir);

int a = strlen(changedDir);

int b = a;

int i = 0;

bool strf = true;

while ((changedDir[b] != '/') && (b >= 0))

{

if (changedDir[b - 1] == '/')

strf = false;

i++;

b--;

}

char\* splitter = "+";

char\* istr;

char\* arr[10];

if(!strf)

{

changedDir[a - i] = '+';

printf("%s\n", changedDir);

int f = 0;

istr = strtok(changedDir, splitter);

while (istr != NULL)

{

arr[f] = istr;

istr = strtok(NULL, splitter);

f++;

}

char\* f\_path = arr[0];

char\* f\_file = arr[1];

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

strcpy(str, workDir);

SetCurrentDirectoryA(f\_path);

}

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

printf("%s\n", workDir);

if (strstr(workDir, root))

{

SetCurrentDirectoryA(str); //Обратно в нашу директорию

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

puts("Путь к фалу 2: ");

scanf("%s", changedDir);

char path\_two[100];

strcpy(path\_two, changedDir);

a = strlen(changedDir);

b = a;

i = 0;

bool strf = true;

while (changedDir[b] != '/')

{

if (changedDir[b - 1] == '/')

strf = false;

i++;

b--;

}

if (!strf)

{

changedDir[a - i] = '+';

printf("%s\n", changedDir);

int f = 0;

istr = strtok(changedDir, splitter);

while (istr != NULL)

{

arr[f] = istr;

istr = strtok(NULL, splitter);

f++;

}

char\* s\_path = arr[0];

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

strcpy(str, workDir);

SetCurrentDirectoryA(s\_path);

}

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

if (strstr(workDir, root))

{

SetCurrentDirectoryA(str);

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

printf("%s\n", workDir);

printf("Это первый файл: %s\n", path\_one);

printf("Это второй файл: %s\n", path\_two);

printf("%s\n", workDir);

if (CopyFileA(path\_one, path\_two, true)) {

puts("Успешно скопировано!");

}

else {

puts("Ошибка копирования!");

}

}

else {

printf("Нельзя выше!\n");

}

}

else {

SetCurrentDirectoryA(str);

GetCurrentDirectoryA(MAX\_LENGTH, str);

printf("Нельзя выше!\n%s\n", str);

}

}

void delFile(char\* root, char\* workDir) {

char changedDir[MAX\_LENGTH];

char str[MAX\_LENGTH];

printf("%s\n", root);

puts("Путь к удаляемому фалу : ");

scanf("%s", &changedDir);

char path\_one[100];

strcpy(path\_one, changedDir);

int a = strlen(changedDir);

int b = a;

int i = 0;

bool strf = true;

while ((changedDir[b] != '/') && (b >= 0))

{

if (changedDir[b - 1] == '/')

strf = false;

i++;

b--;

}

char\* splitter = "+";

char\* istr;

char\* arr[10];

if (!strf)

{

changedDir[a - i] = '+';

printf("%s\n", changedDir);

int f = 0;

istr = strtok(changedDir, splitter);

while (istr != NULL)

{

arr[f] = istr;

istr = strtok(NULL, splitter);

f++;

}

char\* f\_path = arr[0];

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

strcpy(str, workDir);

SetCurrentDirectoryA(f\_path);

}

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

printf("%s\n", workDir);

if (strstr(workDir, root))

{

if (DeleteFileA(changedDir))

{

printf("Файл успешно удален\n");

}

else if (!rmdir(changedDir)) {

printf("Директория успешно удалена\n");

}

else {

printf("Не найдено!");

}

}

else {

printf("Вне прав\n");

}

}

void makeDir(char\* workDir, char\* root) {

char changedDir[MAX\_LENGTH], str[MAX\_LENGTH];//Путь к создаваемому файлу и временная переменная для его хранения при изменении

puts("Путь создаваемого файла: ");

scanf("%s", &changedDir);

char\* f\_name;

char\* f\_path;

int a = strlen(changedDir);

int b = a;

int i = 0;

bool strf = true;

while ((changedDir[b] != '/') && (b >= 0))//разбиение строки на путь и файл

{

if (changedDir[b - 1] == '/')

strf = false;

i++;

b--;

}

char\* splitter = "+";//разбиение строки на путь и файл

char\* istr;

char\* arr[10];

if (!strf)

{

changedDir[a - i] = '+';

printf("%s\n", changedDir);

int f = 0;

istr = strtok(changedDir, splitter);

while (istr != NULL)

{

arr[f] = istr;

istr = strtok(NULL, splitter);

f++;

}

f\_path = arr[0];

f\_name = arr[1];

GetCurrentDirectoryA(MAX\_LENGTH, workDir);//Проверка на вхождение в каталог корня

strcpy(str, workDir);

SetCurrentDirectoryA(f\_path);

}

else {

f\_name = changedDir;

}

GetCurrentDirectoryA(MAX\_LENGTH, workDir);

printf("%s\n", workDir);

if (strstr(workDir, root))

{

SetCurrentDirectoryA(str);//Перехождение обратно в каталог корня, если мы соблюдаем условие

DWORD dwTemp;

HANDLE hFile = CreateFileA(f\_name, GENERIC\_WRITE, 0, NULL,

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

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

puts("Ошибка создания.");

return;

}

}

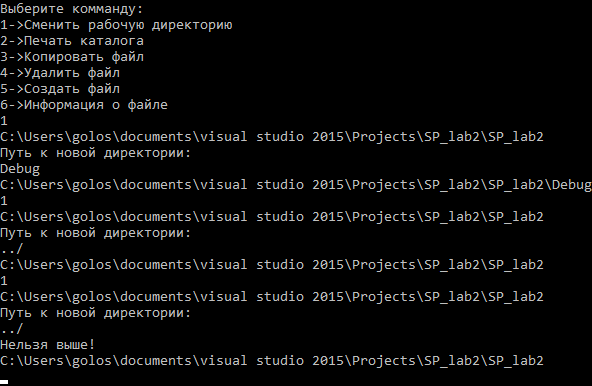
else {

puts("Вне прав.");

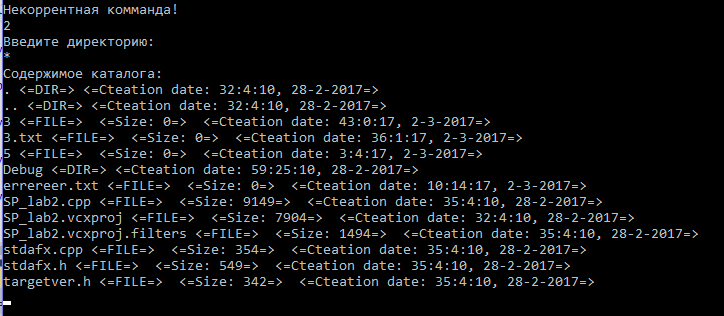
}

}

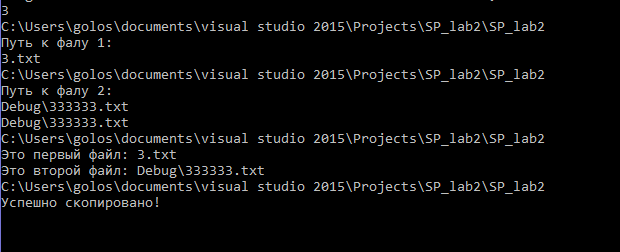
**Скриншоты:**

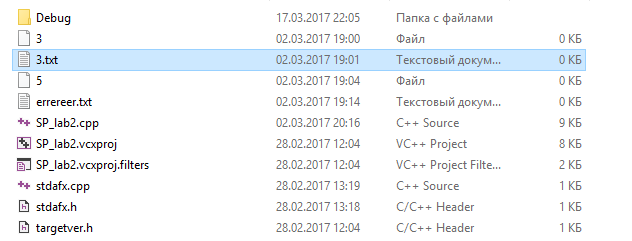


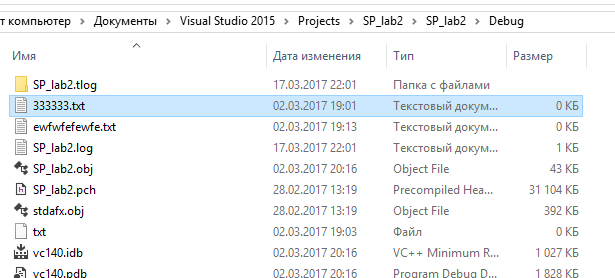
**Рисунок 4 – Cмена рабочей директории с проверкой на принадлежность корневой**



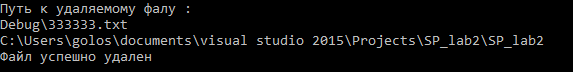
**Рисунок 5 – Вывод каталога файлов рабочей директории**



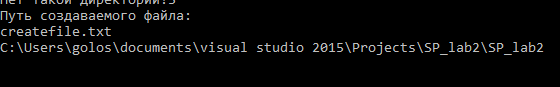
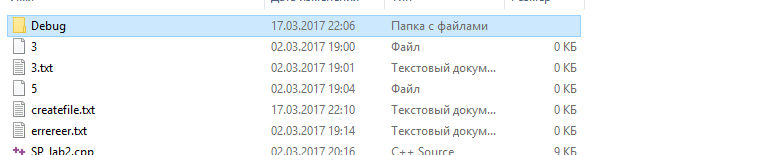




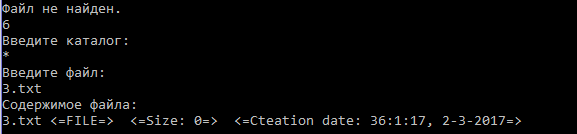
**Рисунок 6 – Копирование файла**



**Рисунок 7 – Удаление**

**Рисунок 8 – Создание**



**Рисунок 9 – Вывод информации**

**Выводы:** я освоил для себя такие новые функции, как FindFistfFile, FindNextFile, SetCurrentDirectoryA, GetCurrentDirectoryA, а также закрепил знание некоторых уже изученных, таких как CreateFileA, CreateFile, понял назначение новых типов, а также с их помощью создал полноченный файловый менеджер и файл структурю