МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки

**Лабораторна робота**

з Системного програмування

(назва дисципліни)

на тему: «Вивчення системних викликів Win32 API роботи з файлами»

Виконав: студент 3-го курсу групи № 535а

напряму підготовки (спеціальності)

123-«Комп’ютерна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва напряму підготовки (спеціальності))

Скринник Є. І.

(прізвище й ініціали студента)

Прийняв: асистент каф.503

Мозговий М.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2020

**Цель работы:**

Изучение системных вызовов Win32 API работы с файлами.

**Постановка задачи:**

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.

1. 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

**Код первой программы:**

#include <stdio.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <vector>

#include <locale.h>

using namespace std;

const char recordsFileName[] = "records.recordbase";

struct Record {

UINT number;

FILETIME creationTime;

TCHAR data[81];

UINT changesCounter;

};

struct Header {

UINT notNullRecordsCount;

UINT FileSize;

};

// Cчитывает headerInfo из файла

Header ReadHeader();

// Cчитывает все записи, идущие после структуры headerInfo

vector<Record> ReadRecs();

// Выводите все записи

void PrintRecords();

// Добавляет запись в конец

void AddRecord();

// Изменяет запись по номеру

void ChangeRecord();

// Удаляет запись по номеру

void DeleteRecord();

// Сохранение записей

void SaveRecords(Header headerInfo, vector <Record> records);

int main()

{

setlocale(LC\_ALL, "Russian");

int choice;

for (;;) {

printf("Лабораторная работа № 3\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nПрограмма для работы с записями\n\n");

printf("Меню:\n");

printf("1 - Просмотр всех записей\n");

printf("2 - Добавления записи\n");

printf("3 - Изменение записи\n");

printf("4 - Удаления записи\n");

printf("5 - Выход\n");

printf("Введите пункт меню:");

scanf("%i", &choice);

system("cls");

switch (choice) {

case 1: PrintRecords(); break;

case 2: AddRecord(); break;

case 3: ChangeRecord(); break;

case 4: DeleteRecord(); break;

case 5: return 0; break;

default:printf("Выбран неверный пункт меню попробуйте еще раз!\n");

}

system("pause");

system("cls");

}

}

Header ReadHeader() {

Header headerInfo;

DWORD readedBytes;

HANDLE recordsFile = CreateFile(recordsFileName, GENERIC\_READ, NULL, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

SetFilePointer(recordsFile, NULL, NULL, FILE\_BEGIN);

ReadFile(recordsFile, &headerInfo, sizeof(Header), &readedBytes, NULL);

if (readedBytes == 0) {

headerInfo.notNullRecordsCount = 0;

headerInfo.FileSize = 0;

}

CloseHandle(recordsFile);

return headerInfo;

}

vector<Record> ReadRecs() {

HANDLE recordsFile = CreateFile(recordsFileName, GENERIC\_READ, NULL, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

SetFilePointer(recordsFile, sizeof(Header), NULL, FILE\_BEGIN);

vector<Record> records(0);

Record i;

DWORD readedBytes = 0;

for (;;) {

bool k = ReadFile(recordsFile, &i, sizeof(Record), &readedBytes, NULL);

if (readedBytes == 0) break;

records.push\_back(i);

}

CloseHandle(recordsFile);

return records;

}

void PrintRecords() {

SYSTEMTIME SysTime;

vector <Record> records = ReadRecs();

Header count = ReadHeader();

printf("Размер файла: %u байтов\nКоличество непустых записей: %u\n", count.FileSize,count.notNullRecordsCount);

for (int i = 0; i < records.size(); i++) {

FileTimeToSystemTime(&records[i].creationTime, &SysTime);

printf("#%i ",records[i].number);

printf("%2u.%2u.%u %u:%u ",SysTime.wDay,SysTime.wMonth,SysTime.wYear,SysTime.wHour,SysTime.wMinute);

printf("%5u ", records[i].changesCounter);

printf("%s;\n", records[i].data);

}

}

void AddRecord() {

Header headerInfo = ReadHeader();

vector <Record> records = ReadRecs();

SYSTEMTIME k, s;

GetLocalTime(&k);

FILETIME fileTime;

Record record;

//Перевод времени из системного в файловое

SystemTimeToFileTime(&k, &fileTime);

FileTimeToSystemTime(&fileTime, &s);

printf("Введите данные размером до 80 символов, не используя пробел или введите \* чтобы создать пустую запись:\n");

scanf("%s", record.data);

if (strcmp(record.data, "\*") == 0) {

record.number = records.size();

record.creationTime = fileTime;

strcpy(record.data, "");

record.changesCounter = 0;

}

else {

record.number = records.size();

record.creationTime = fileTime;

record.changesCounter = 0;

headerInfo.notNullRecordsCount++;

}

records.push\_back(record);

headerInfo.FileSize = sizeof(headerInfo) + sizeof(Record)\*records.size();

SaveRecords(headerInfo, records);

}

void ChangeRecord() {

SYSTEMTIME sysTime;

Header headerInfo = ReadHeader();

vector <Record> records = ReadRecs();

UINT recordNumber;

printf("Введите номер записи которую вы хотите изменить:");

scanf("%u",&recordNumber);

try {

FileTimeToSystemTime(&records.at(recordNumber).creationTime, &sysTime);

printf("\nВыбранная запись:\n");

printf("Номер записи: %u\n", records[recordNumber].number);

printf("Время создания записи: %u.%u.%u %u:%u\n", sysTime.wDay, sysTime.wMonth, sysTime.wYear, sysTime.wHour, sysTime.wMinute);

printf("Данные записи: %s\n", records[recordNumber].data);

printf("Количество изменений: %u\n", records[recordNumber].changesCounter);

printf("\nВведите новые данные(Без пробелов и меньше 80 символов)\nДля обнуления записи введите \*\nВвод:");

TCHAR str[81];

strcpy(str, records[recordNumber].data);

scanf("%s", records[recordNumber].data);

if (strcmp(records[recordNumber].data, "\*") == 0) strcpy(records[recordNumber].data, "");

if (strcmp(str, "") == 0 && strcmp(records[recordNumber].data, "") != 0) headerInfo.notNullRecordsCount++;

if (strcmp(str, "") != 0 && strcmp(records[recordNumber].data, "") == 0) headerInfo.notNullRecordsCount--;

records[recordNumber].changesCounter++;

}

catch(const std::out\_of\_range& e)

{

printf("\*\*\* Произошла ошибка изменении записи \*\*\*\n");

}

SaveRecords(headerInfo, records);

}

void DeleteRecord() {

Header headerInfo = ReadHeader();

vector <Record> records = ReadRecs();

try {

int i;

printf("Введите номер записи которую хотите удалить: ");

scanf("%i", &i);

if (strcmp(records.at(i).data, "") != 0) headerInfo.notNullRecordsCount--;

records.erase(records.begin() + i);

vector<Record>(records).swap(records);

for (int k = 0; k < records.size(); k++) {

records[k].number = k;

}

}

catch (const std::out\_of\_range& e)

{

printf("\n\*\*\* Произошла ошибка при удалении записи \*\*\*");

}

SaveRecords(headerInfo, records);

}

void SaveRecords(Header headerInfo, vector <Record> records)

{

HANDLE recordsFile = CreateFile(recordsFileName, GENERIC\_WRITE, NULL, NULL, TRUNCATE\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

SetFilePointer(recordsFile, NULL, NULL, FILE\_BEGIN);

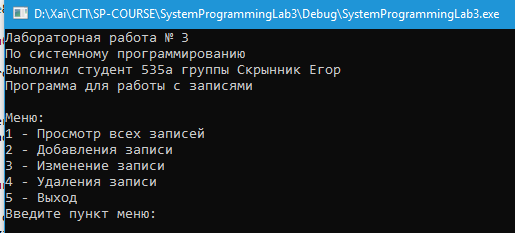
WriteFile(recordsFile, &headerInfo, sizeof(Header), NULL, NULL);

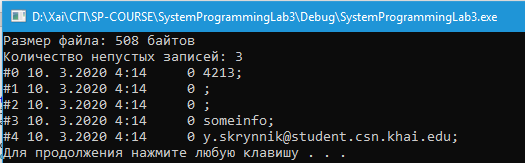
SetFilePointer(recordsFile, NULL, NULL, FILE\_END);

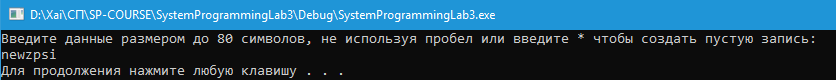
for (int i = 0; i < records.size(); i++) WriteFile(recordsFile, &records[i], sizeof(Record), NULL, NULL);

CloseHandle(recordsFile);

}

**Результат работы:**

Рисунок 1 – Главное меню приложения.

Рисунок 2 – Просмотр всех записей.  
Рисунок 3 – Добавление записи.

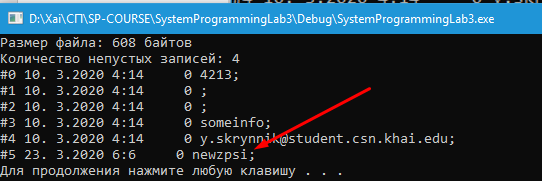
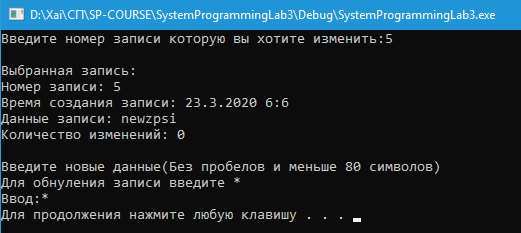
Рисунок 4 – Добавленная запись.

Рисунок 5 – Интерфейс изменения записи.

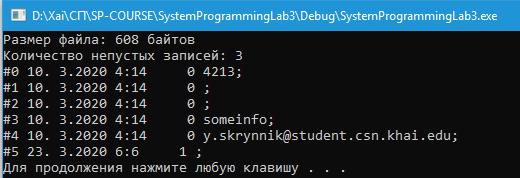
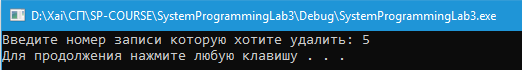


Рисунок 6 – Увеличение счетчика изменений записи, и сама измененная запись.



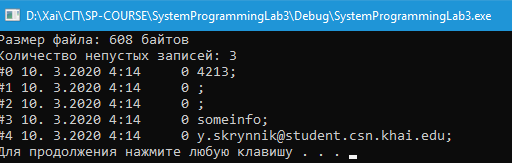
Рисунок 7 – Удаление записи с номером 5.

Рисунок 8 – Результат удаления.

**Код второй программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>;

#include <iostream>

#include <locale.h>

using namespace std;

// Выбор директории

void SetDirectory(TCHAR \*buff);

// Вывод файлов выбранной директории

void PrintDirectory(TCHAR \*buff);

// Копирование файла

void FileCopy(TCHAR \*buff);

// Создание директории

void MakeDirectory(TCHAR \*buff);

// Подробная информация о файле

void DetailedInfoAboutFile(TCHAR \*buff);

// Удаление файла или пустой директории

void DeleteFileOrDirectory(TCHAR \*buff);

int main() {

setlocale(LC\_ALL, "rus");

int choice;

for (;;) {

TCHAR buff[MAX\_PATH];

GetCurrentDirectory(sizeof(buff), buff);

system("cls");

printf("Лабораторная работа № 3\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nФайловый менеджер\n\n");

printf("Текущая директория:%s\nМеню:\n",buff);

printf("1 - Выбор директории\n");

printf("2 - Печать файлов выбранной директории\n");

printf("3 - Копировать файл\n");

printf("4 - Создание директории\n");

printf("5 - Получить подробную информацию о файле\n");

printf("6 - Удаление файла или пустой директории\n");

printf("7 - Выход\n");

printf("Введите пункт меню:");

scanf("%i", &choice);

system("cls");

switch (choice) {

case 1: SetDirectory(buff); break;

case 2: PrintDirectory(buff); break;

case 3: FileCopy(buff); break;

case 4: MakeDirectory(buff); break;

case 5: DetailedInfoAboutFile(buff); break;

case 6: DeleteFileOrDirectory(buff); break;

case 7: return 0; break;

default:printf("Выбран неверный пункт меню попробуйте еще раз!\n");

}

system("pause");

system("cls");

}

}

void SetDirectory(TCHAR \*buff) {

for (;;) {

printf("Введите полный путь к директории\n");

printf("Например: C://folder\n>>");

TCHAR path[MAX\_PATH];

getchar();

fgets(path, MAX\_PATH - 1,stdin);

path[strlen(path) - 1] = '\0';

if (!SetCurrentDirectory(path))

printf("\n\*\*\*Ошибка! Неверный путь к директории. \*\*\*\n");

else {

GetCurrentDirectory(sizeof(path), path);

printf("Ваша текущая директория: %s\n",path);

strcpy(buff, path);

return;

}

system("pause");

}

}

void PrintDirectory(TCHAR \*buff) {

printf("Файлы в текущей директории: \n");

WIN32\_FIND\_DATA FindFileData;

TCHAR find[MAX\_PATH];

strcpy(find, buff);

strcat(find, "\\\*");

HANDLE hf = FindFirstFile(find, &FindFileData);

if (hf != INVALID\_HANDLE\_VALUE)

{

do{

SYSTEMTIME systime;

FILETIME localfiletime;

FileTimeToLocalFileTime(&FindFileData.ftCreationTime, &localfiletime);

FileTimeToSystemTime(&localfiletime, &systime);

LONGLONG size = (FindFileData.nFileSizeHigh \* (MAXDWORD + 1)) + FindFileData.nFileSizeLow;

printf("%-40s %20lu bytes ",FindFileData.cFileName,size);

printf("%2u.%u.%u %u:%u\n", systime.wDay, systime.wMonth, systime.wYear, systime.wHour, systime.wMinute);

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

FindClose(hf);

}

}

void FileCopy(TCHAR \*buff) {

for (;;) {

printf("Введите имя файла в текущей директории например lab1.txt\n>>");

TCHAR find[MAX\_PATH], s[MAX\_PATH], newDirectory[MAX\_PATH];

strcpy(find, buff);

getchar();

fgets(s, MAX\_PATH - 1, stdin);

s[strlen(s) - 1] = '\0';

strcat(find, "\\");

strcat(find, s);

printf("\nВведите путь куда следует копировать файл например(D://...)\n>>");

fgets(newDirectory, MAX\_PATH - 1, stdin);

newDirectory[strlen(newDirectory) - 1] = '\0';

strcat(newDirectory, "\\");

strcat(newDirectory, s);

if (CopyFile(find, newDirectory, false)) return;

else cout << "\*\*\*Ошибка! Неккоректный путь или имя файла \*\*\*\n";

}

}

void MakeDirectory(TCHAR \*buff) {

TCHAR newDir[MAX\_PATH], way[MAX\_PATH];

printf("\nВведите название новой директории\n>>");

getchar();

fgets(newDir, MAX\_PATH - 1, stdin);

newDir[strlen(newDir) - 1] = '\0';

strcpy(way, buff);

strcat(way, "\\");

strcat(way, newDir);

if (CreateDirectory(way, NULL))

printf("Директория успешно создана\n");

else

printf("\*\*\*Ошибка! Ну удалось создать директорию \*\*\*\n");

}

void DetailedInfoAboutFile(TCHAR \*buff) {

TCHAR file[MAX\_PATH], way[MAX\_PATH];

printf("Введите имя файла:\n>>");

getchar();

fgets(file, MAX\_PATH - 1, stdin);

file[strlen(file) - 1] = '\0';

strcpy(way, buff);

strcat(way, "\\");

strcat(way, file);

HANDLE hFile1 = CreateFile(way, GENERIC\_READ, NULL, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

BY\_HANDLE\_FILE\_INFORMATION information;

BOOL f = GetFileInformationByHandle(hFile1, &information);

CloseHandle(hFile1);

if (!f) {

printf("\*\*\* Ошибка неккоректное имя файла \*\*\*\n");

}

else {

printf("Файл ");

if ((information.dwFileAttributes & FILE\_ATTRIBUTE\_HIDDEN)!=0)

printf("скрытый ");

if ((information.dwFileAttributes & FILE\_ATTRIBUTE\_SYSTEM)!=0)

printf("системный ");

if ((information.dwFileAttributes & FILE\_ATTRIBUTE\_ARCHIVE)!=0)

printf("архив ");

if ((information.dwFileAttributes & FILE\_ATTRIBUTE\_COMPRESSED)!=0)

printf("сжатый ");

if ((information.dwFileAttributes & FILE\_ATTRIBUTE\_NORMAL )!=0)

printf("нормальный ");

cout << ".\n";

SYSTEMTIME systime;

FILETIME localfiletime;

FileTimeToLocalFileTime(&information.ftCreationTime, &localfiletime);

FileTimeToSystemTime(&localfiletime, &systime);

printf("Время создания - %2u.%u.%u %u:%u\n", systime.wDay, systime.wMonth, systime.wYear, systime.wHour, systime.wMinute);

FileTimeToLocalFileTime(&information.ftLastWriteTime, &localfiletime);

FileTimeToSystemTime(&localfiletime, &systime);

printf("Время изменения - %2u.%u.%u %u:%u\n", systime.wDay, systime.wMonth, systime.wYear, systime.wHour, systime.wMinute);

}

}

void DeleteFileOrDirectory(TCHAR \*buff) {

TCHAR file[MAX\_PATH], way[MAX\_PATH];

printf("Введите имя файла или директории которую вы хотите удалить!\n>>");

getchar();

fgets(file, MAX\_PATH - 1, stdin);

file[strlen(file) - 1] = '\0';

strcpy(way, buff);

strcat(way, "\\");

strcat(way, file);

BOOL g = DeleteFile(way), k;

if (!g) {

k = RemoveDirectory(way);

}

if (!g && !k) {

DWORD lastErrorCode = GetLastError();

void\* cstr; //В эту переменную будет записано сообщение

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

GetLastError(),

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&cstr,

0,

NULL

);

// Преобразование сообщения в читабельный вид

PTCHAR res((char\*)cstr);

printf("%s", res);

}

}

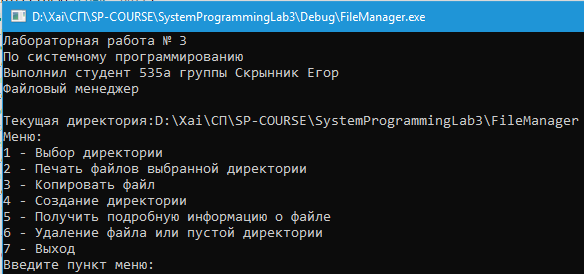
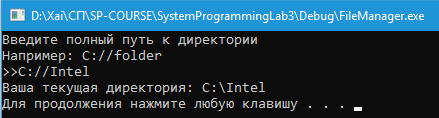
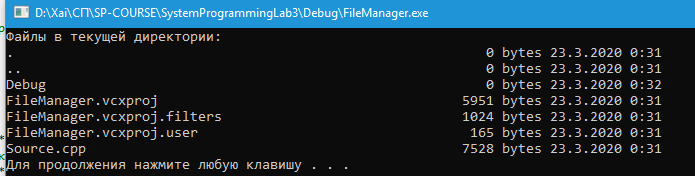
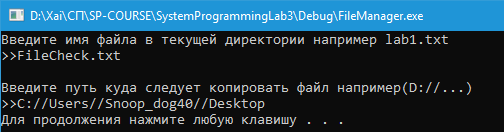
**Результат работы:**

Рисунок 9 – Главное меню второй программы.



Рисунок 10 – Смена директории.

Рисунок 11 – Печать содержимого директории.

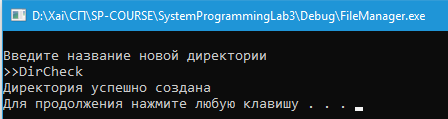
Рисунок 12 – Копирование файла.

Рисунок 13 – Создание директории.

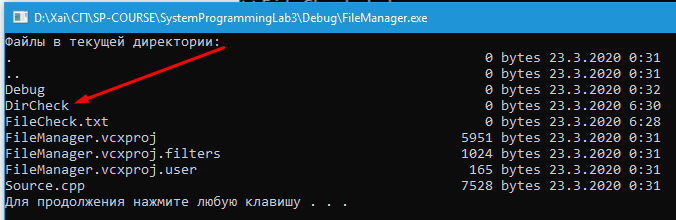
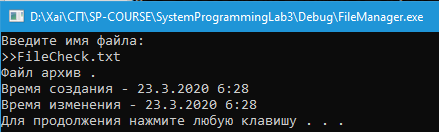
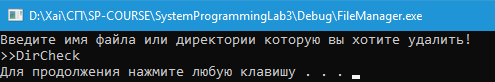


Рисунок 14 – Созданная директория.

Рисунок 15 – Получение подробной информации о файле.

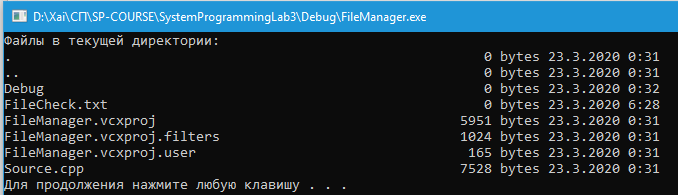
Рисунок 16 – Удаление папки DirCheck.

Рисунок 17 – Просмотр директории после удаления папки.