МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра Вычислительной техники

ОТЧЕТ

по лабораторной работе № 1

по дисциплине «Операционные системы»

ТЕМА: «Управление файловой системой»

Студент гр. 3311	Баймухамедов Р. Р.
Преподаватель	Тимофеев А. В.

Санкт-Петербург

2024

Цель работы

Исследовать управление файловой системой с помощью Win32 API

Задание

Управление дисками, каталогами и файлами.

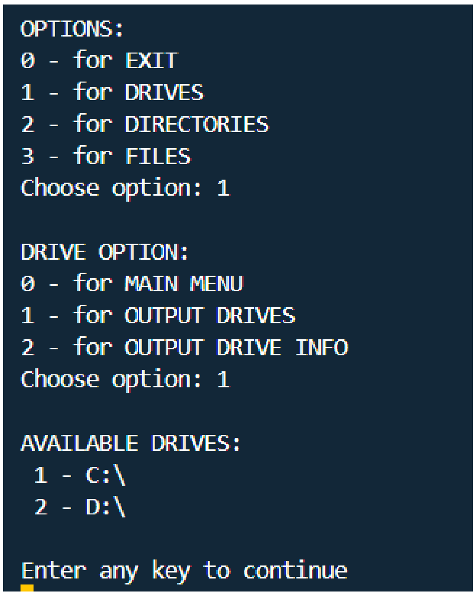
Постановка задачи и описание решения

Для выполнения данной лабораторной работы необходимо разработать консольное приложение с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которое выполняет:

- Вывод списка дисков (функции Win32 API – GetLogicalDrives / GetLogicalDrivesStrings)
- Для одного из выбранных дисков вывод информации о диске и размер свободного пространства (функции Win32 API – GetDriveType, GetVolumeInformation, GetDiskFreeSpace)
- Создание и удаление заданных каталогов (функции Win32 API – CreateDirectory, RemoveDirectory)
- Создание файлов в новых каталогах (функция Win32 API – CreateFile)
- Копирование и перемещение файлов между каталогами с возможностью выявления попытки работы с файлами, имеющими совпадающие имена (Функции Win32 API – CopyFile, MoveFile, MoveFileEx)
- Анализ и изменение атрибутов файлов (Функции Win32 API – GetFileAttributes, SetFileAttributes, GetFileInformationByHandle, GetFileTime, SetFileTime)

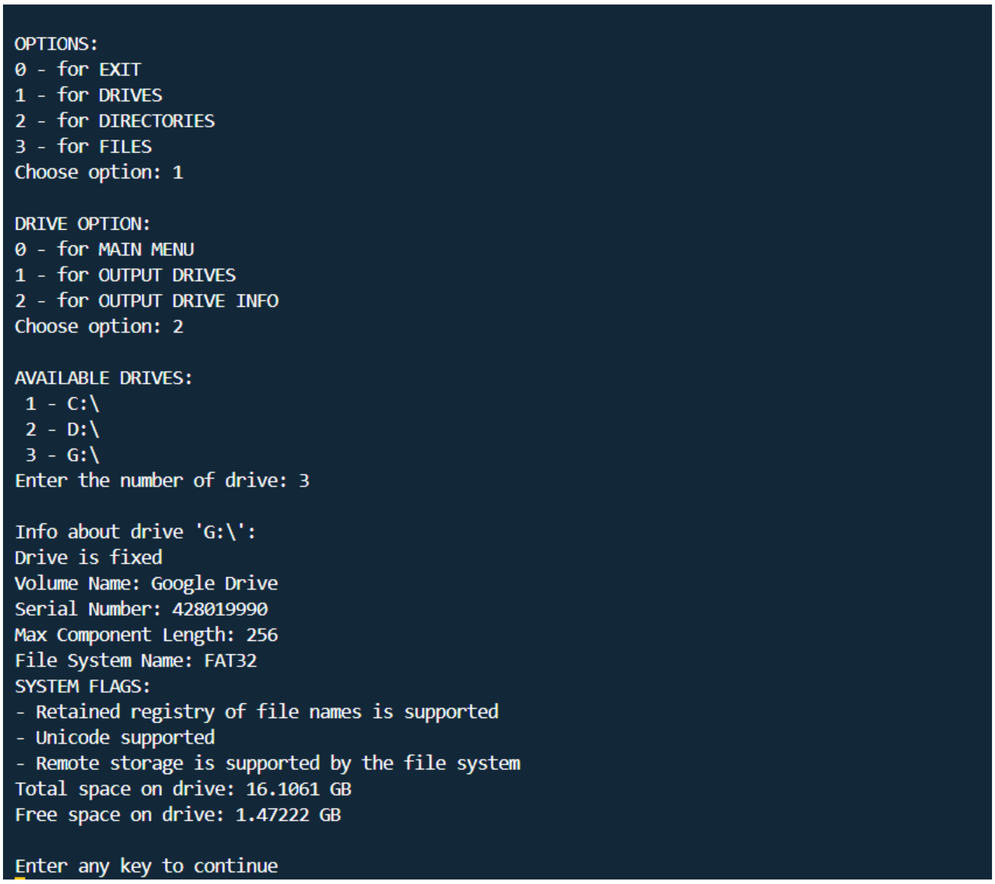
Примеры выполнения программы

Примеры работоспособности консольного приложения продемонстрированы на рис. 1-4



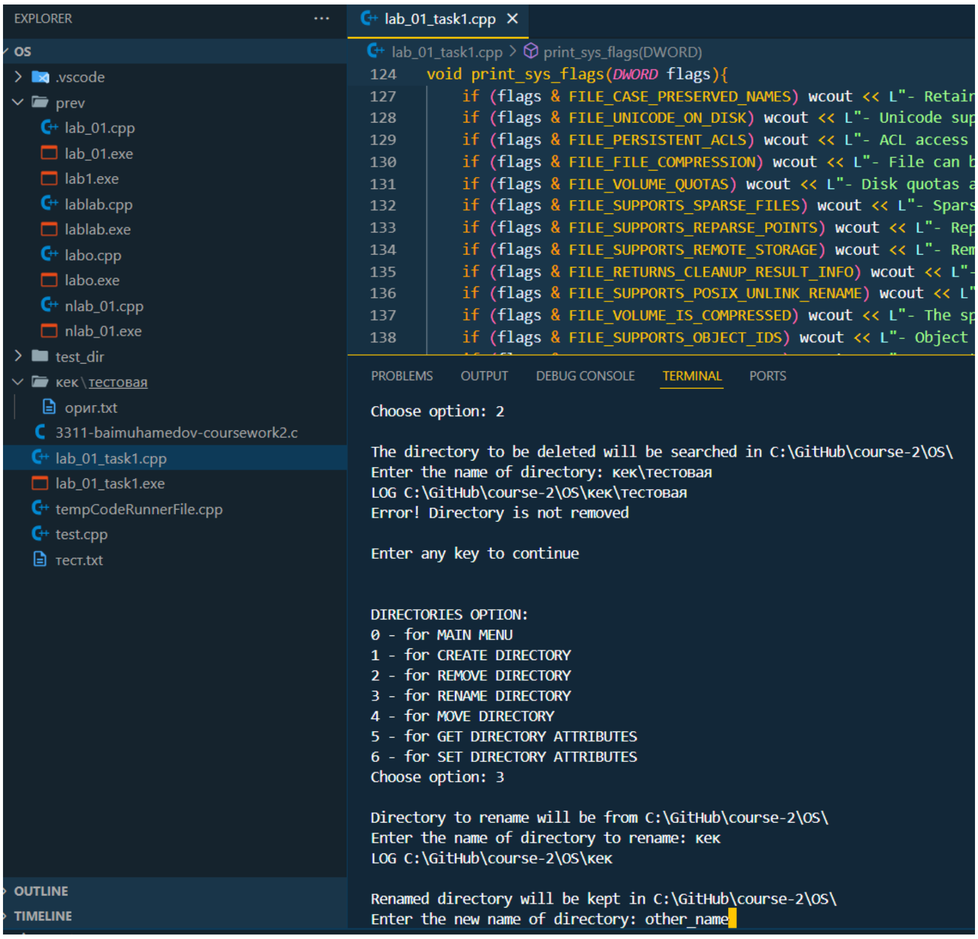
```
OPTIONS:
0 - for EXIT
1 - for DRIVES
2 - for DIRECTORIES
3 - for FILES
Choose option: 1

DRIVE OPTION:
0 - for MAIN MENU
1 - for OUTPUT DRIVES
2 - for OUTPUT DRIVE INFO
Choose option: 1

AVAILABLE DRIVES:
 1 - C:\
 2 - D:\

Enter any key to continue
```

Рисунок 1.1 (Вывод дисков)



```
OPTIONS:
0 - for EXIT
1 - for DRIVES
2 - for DIRECTORIES
3 - for FILES
Choose option: 1

DRIVE OPTION:
0 - for MAIN MENU
1 - for OUTPUT DRIVES
2 - for OUTPUT DRIVE INFO
Choose option: 2

AVAILABLE DRIVES:
 1 - C:\
 2 - D:\
 3 - G:\
Enter the number of drive: 3

Info about drive 'G:\':
Drive is fixed
Volume Name: Google Drive
Serial Number: 428019990
Max Component Length: 256
File System Name: FAT32
SYSTEM FLAGS:
- Retained registry of file names is supported
- Unicode supported
- Remote storage is supported by the file system
Total space on drive: 16.1061 GB
Free space on drive: 1.47222 GB

Enter any key to continue
```

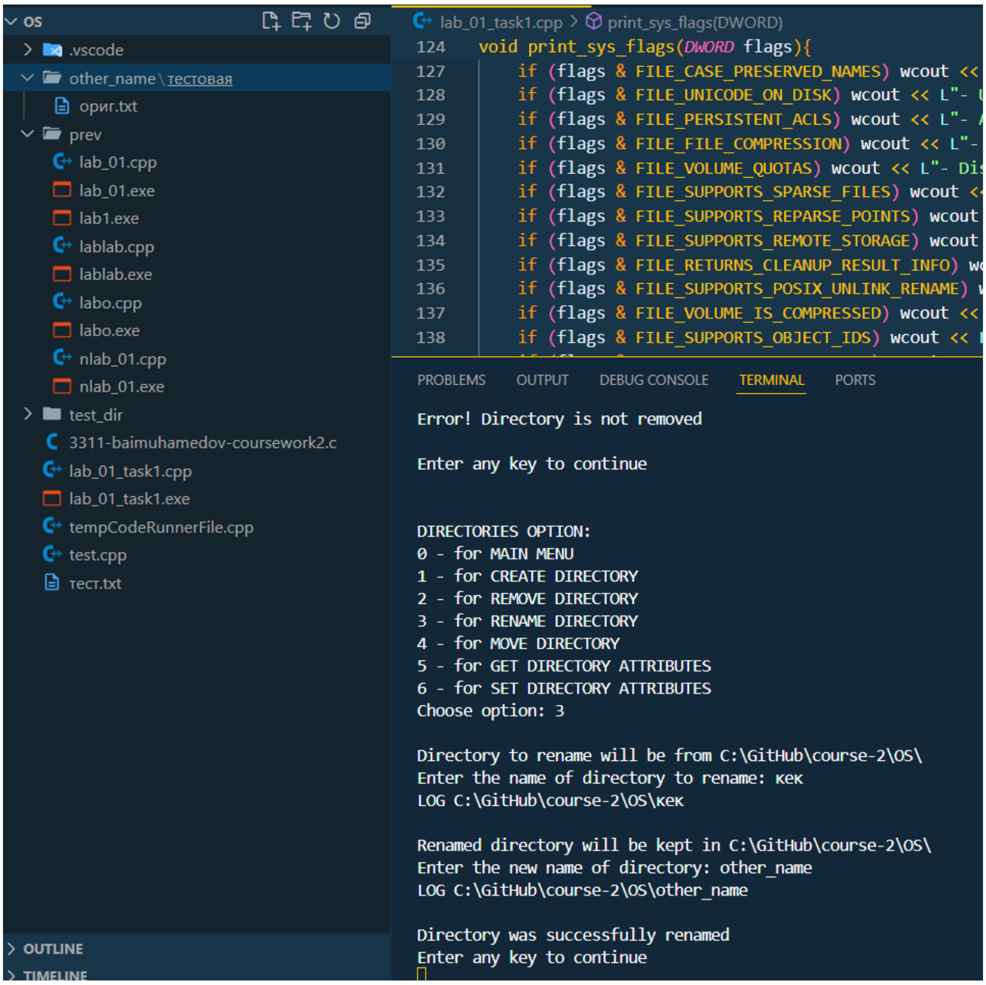
Рисунок 1.2 (Вывод информации о диске)



```
void print_sys_flags(DWORD flags){
    if (flags & FILE_CASE_PRESERVED_NAMES) wcout << L"- Retair
    if (flags & FILE_UNICODE_ON_DISK) wcout << L"- Unicode sup
    if (flags & FILE_PERSISTENT_ACLS) wcout << L"- ACL access
    if (flags & FILE_FILE_COMPRESSION) wcout << L"- File can b
    if (flags & FILE_VOLUME_QUOTAS) wcout << L"- Disk quotas a
    if (flags & FILE_SUPPORTS_SPARSE_FILES) wcout << L"- Spars
    if (flags & FILE_SUPPORTS_REPARSE_POINTS) wcout << L"- Rep
    if (flags & FILE_SUPPORTS_REMOTE_STORAGE) wcout << L"- Rem
    if (flags & FILE_RETURNS_CLEANUP_RESULT_INFO) wcout << L"-
    if (flags & FILE_SUPPORTS_POSIX_UNLINK_RENAME) wcout << L"
    if (flags & FILE_VOLUME_IS_COMPRESSED) wcout << L"- The sp
    if (flags & FILE_SUPPORTS_OBJECT_IDS) wcout << L"- Object
```

```
Choose option: 2

The directory to be deleted will be searched in C:\GitHub\course-2\OS\
Enter the name of directory: кек\тестовая
LOG C:\GitHub\course-2\OS\кек\тестовая
Error! Directory is not removed

Enter any key to continue


DIRECTORIES OPTION:
0 - for MAIN MENU
1 - for CREATE DIRECTORY
2 - for REMOVE DIRECTORY
3 - for RENAME DIRECTORY
4 - for MOVE DIRECTORY
5 - for GET DIRECTORY ATTRIBUTES
6 - for SET DIRECTORY ATTRIBUTES
Choose option: 3

Directory to rename will be from C:\GitHub\course-2\OS\
Enter the name of directory to rename: кек
LOG C:\GitHub\course-2\OS\кек

Renamed directory will be kept in C:\GitHub\course-2\OS\
Enter the new name of directory: other_name
```

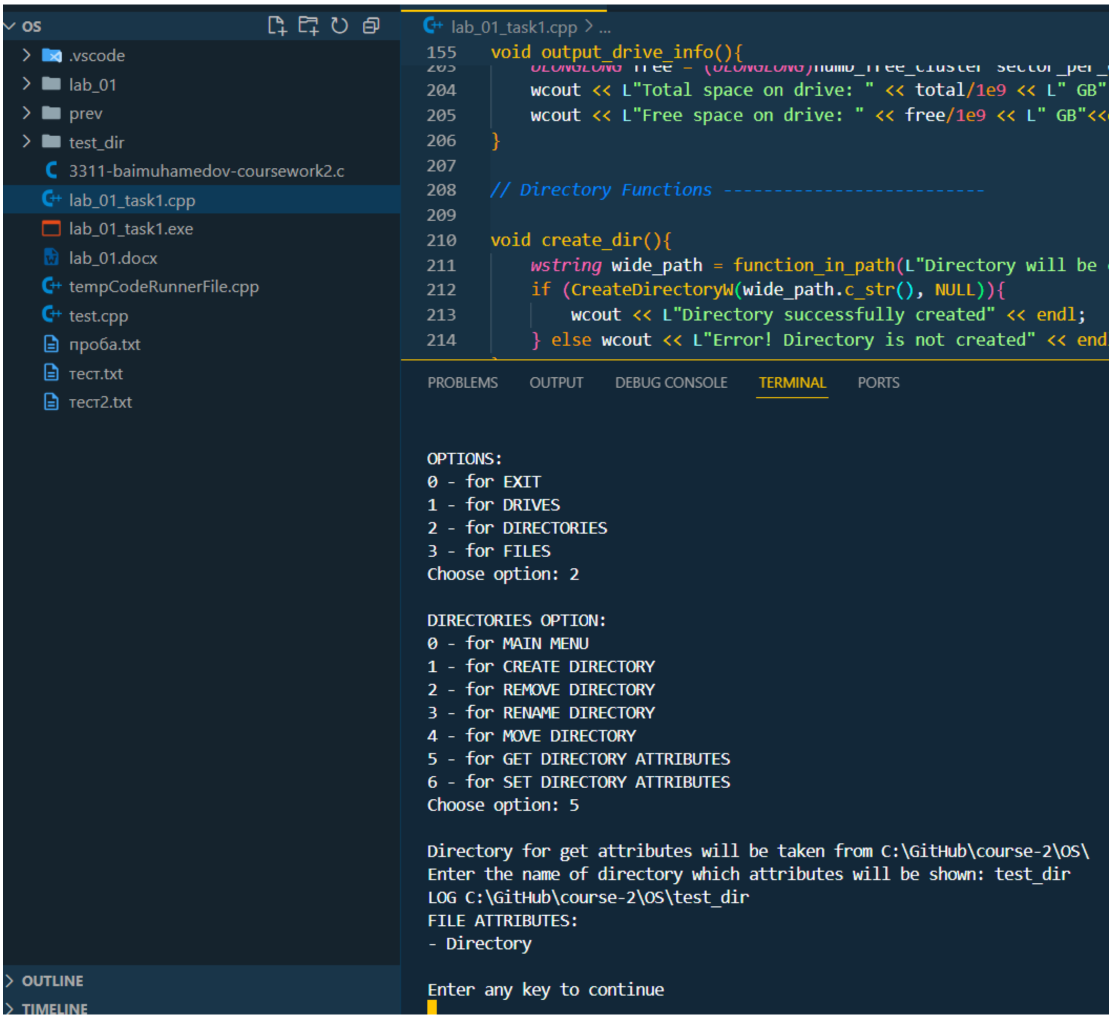
Рисунок 2.1 (до изменения названия папки)



```
void print_sys_flags(DWORD flags){
    if (flags & FILE_CASE_PRESERVED_NAMES) wcout <<
    if (flags & FILE_UNICODE_ON_DISK) wcout << L"- U
    if (flags & FILE_PERSISTENT_ACLS) wcout << L"- A
    if (flags & FILE_FILE_COMPRESSION) wcout << L"-
    if (flags & FILE_VOLUME_QUOTAS) wcout << L"- Dis
    if (flags & FILE_SUPPORTS_SPARSE_FILES) wcout <<
    if (flags & FILE_SUPPORTS_REPARSE_POINTS) wcout
    if (flags & FILE_SUPPORTS_REMOTE_STORAGE) wcout
    if (flags & FILE_RETURNS_CLEANUP_RESULT_INFO) wc
    if (flags & FILE_SUPPORTS_POSIX_UNLINK_RENAME) w
    if (flags & FILE_VOLUME_IS_COMPRESSED) wcout <<
    if (flags & FILE_SUPPORTS_OBJECT_IDS) wcout << L
```

```
Error! Directory is not removed

Enter any key to continue


DIRECTORIES OPTION:
0 - for MAIN MENU
1 - for CREATE DIRECTORY
2 - for REMOVE DIRECTORY
3 - for RENAME DIRECTORY
4 - for MOVE DIRECTORY
5 - for GET DIRECTORY ATTRIBUTES
6 - for SET DIRECTORY ATTRIBUTES
Choose option: 3

Directory to rename will be from C:\GitHub\course-2\OS\
Enter the name of directory to rename: кек
LOG C:\GitHub\course-2\OS\кек

Renamed directory will be kept in C:\GitHub\course-2\OS\
Enter the new name of directory: other_name
LOG C:\GitHub\course-2\OS\other_name

Directory was successfully renamed
Enter any key to continue
```

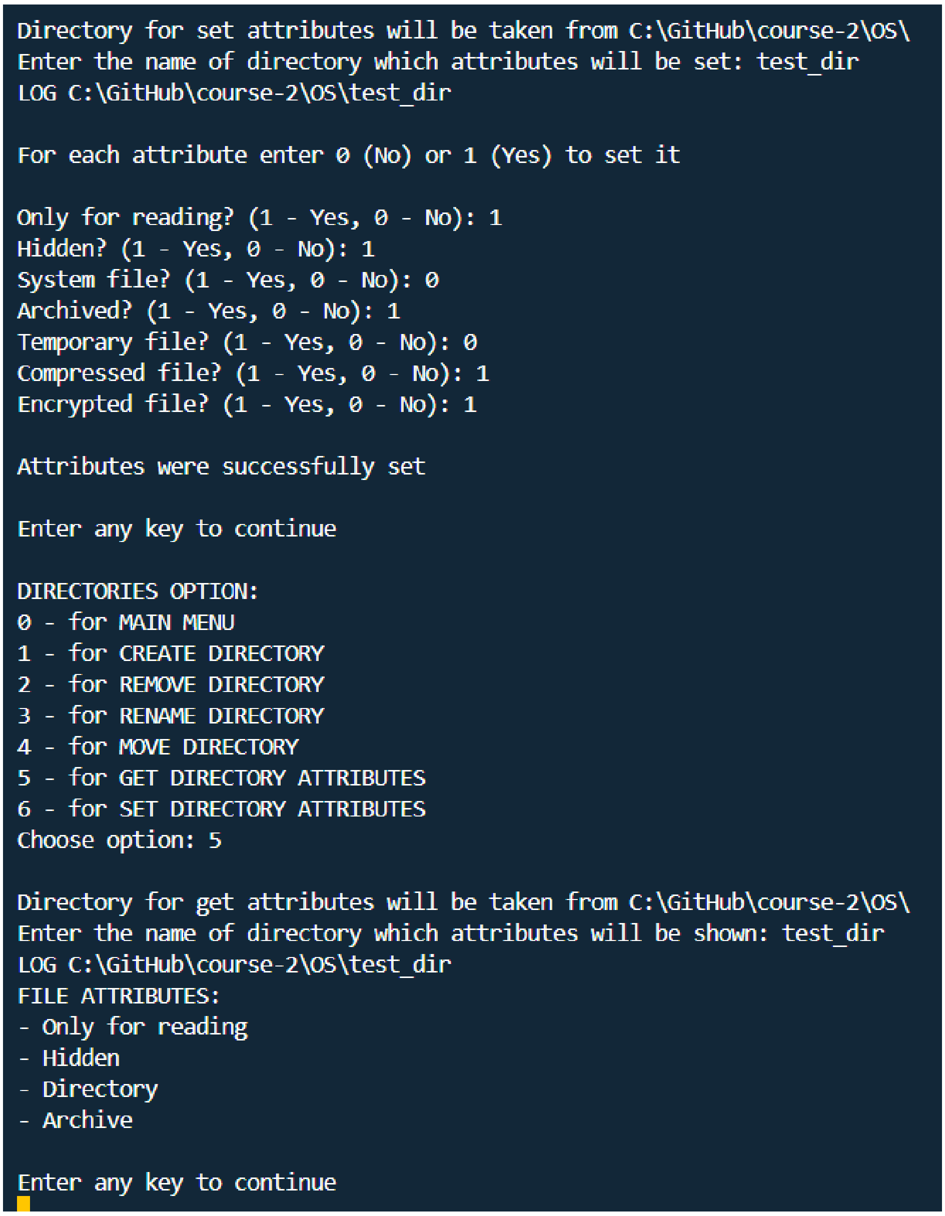
Рисунок 2.2 (после изменения названия папки)



```
void output_drive_info(){
    ULONGLONG free = (ULONGLONG)numb_free_cluster*sector_per_
    wcout << L"Total space on drive: " << total/1e9 << L" GB"
    wcout << L"Free space on drive: " << free/1e9 << L" GB"<<
}

// Directory Functions --------------------------

void create_dir(){
    wstring wide_path = function_in_path(L"Directory will be 
    if (CreateDirectoryW(wide_path.c_str(), NULL)){
        wcout << L"Directory successfully created" << endl;
    } else wcout << L"Error! Directory is not created" << end
```

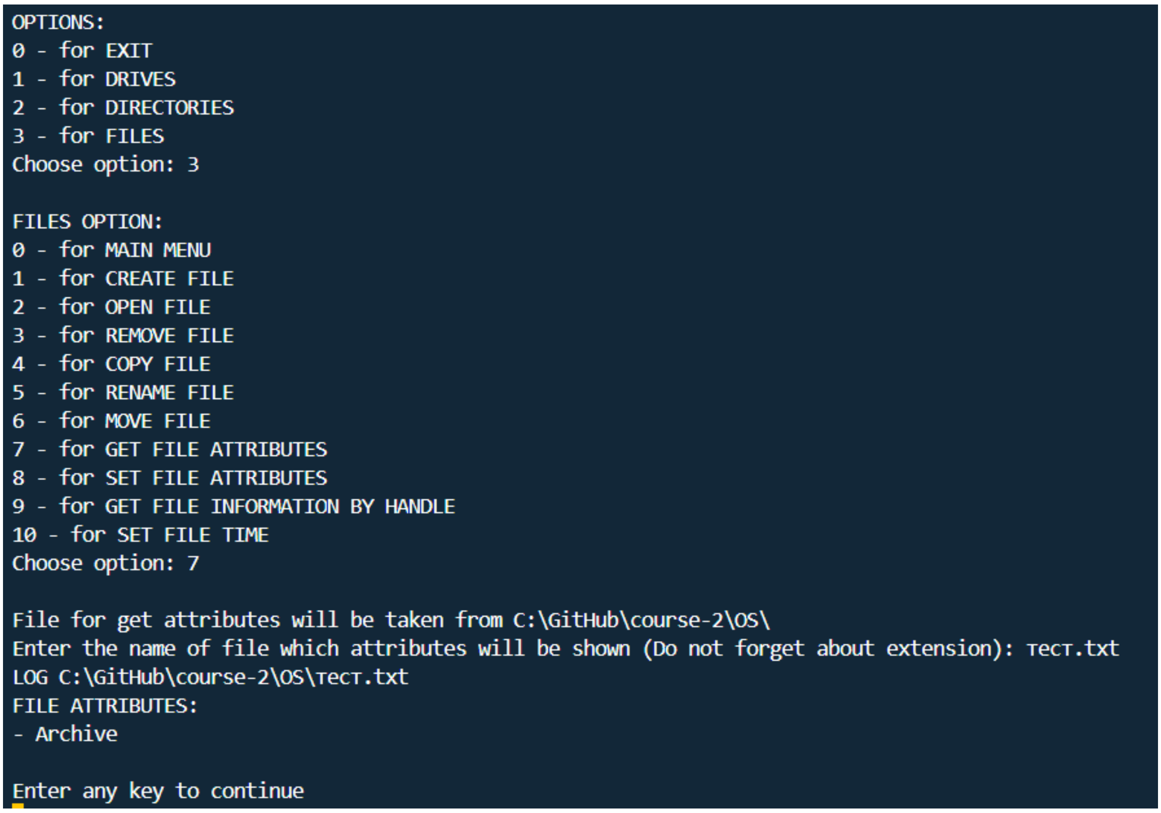
```
OPTIONS:
0 - for EXIT
1 - for DRIVES
2 - for DIRECTORIES
3 - for FILES
Choose option: 2

DIRECTORIES OPTION:
0 - for MAIN MENU
1 - for CREATE DIRECTORY
2 - for REMOVE DIRECTORY
3 - for RENAME DIRECTORY
4 - for MOVE DIRECTORY
5 - for GET DIRECTORY ATTRIBUTES
6 - for SET DIRECTORY ATTRIBUTES
Choose option: 5

Directory for get attributes will be taken from C:\GitHub\course-2\OS\
Enter the name of directory which attributes will be shown: test_dir
LOG C:\GitHub\course-2\OS\test_dir
FILE ATTRIBUTES:
- Directory

Enter any key to continue
```

Рисунок 2.3 (Атрибуты папки до изменения)



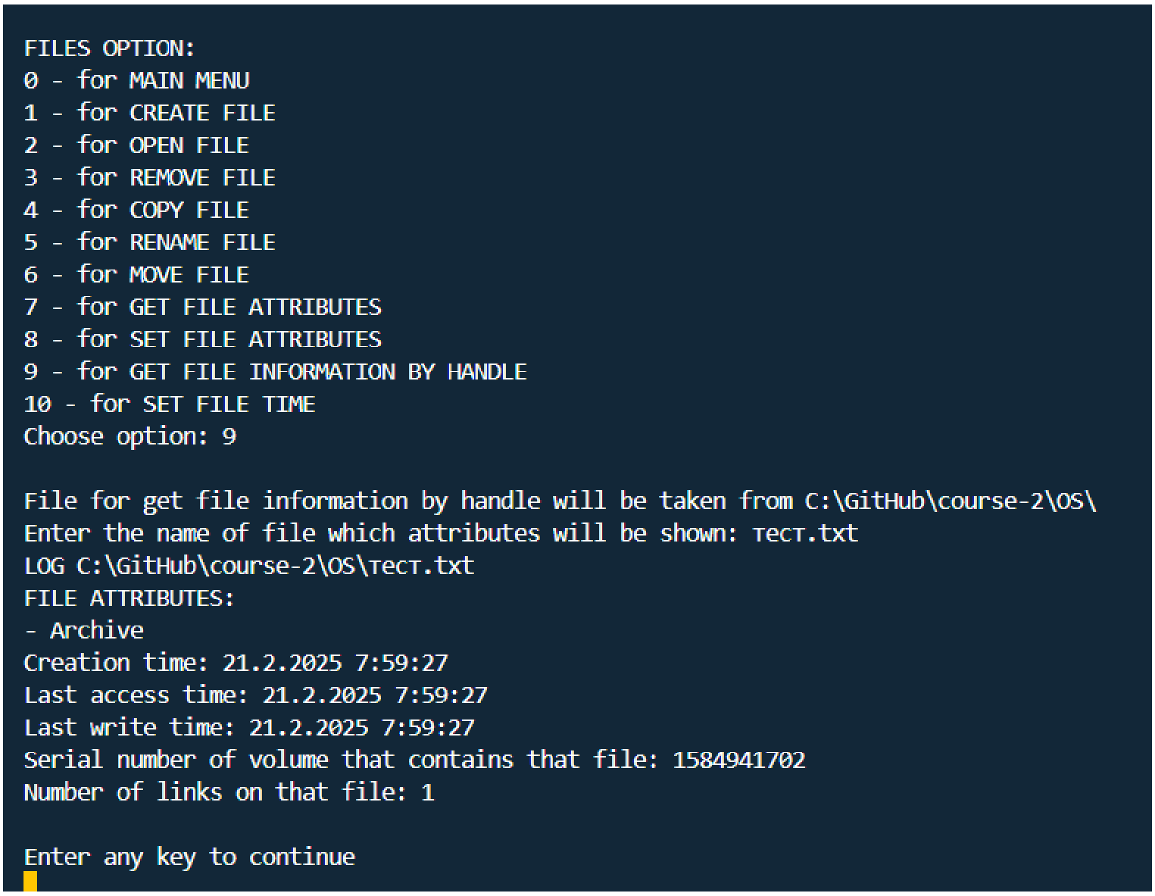
```
Directory for set attributes will be taken from C:\GitHub\course-2\OS\
Enter the name of directory which attributes will be set: test_dir
LOG C:\GitHub\course-2\OS\test_dir

For each attribute enter 0 (No) or 1 (Yes) to set it

Only for reading? (1 - Yes, 0 - No): 1
Hidden? (1 - Yes, 0 - No): 1
System file? (1 - Yes, 0 - No): 0
Archived? (1 - Yes, 0 - No): 1
Temporary file? (1 - Yes, 0 - No): 0
Compressed file? (1 - Yes, 0 - No): 1
Encrypted file? (1 - Yes, 0 - No): 1

Attributes were successfully set

Enter any key to continue

DIRECTORIES OPTION:
0 - for MAIN MENU
1 - for CREATE DIRECTORY
2 - for REMOVE DIRECTORY
3 - for RENAME DIRECTORY
4 - for MOVE DIRECTORY
5 - for GET DIRECTORY ATTRIBUTES
6 - for SET DIRECTORY ATTRIBUTES
Choose option: 5

Directory for get attributes will be taken from C:\GitHub\course-2\OS\
Enter the name of directory which attributes will be shown: test_dir
LOG C:\GitHub\course-2\OS\test_dir
FILE ATTRIBUTES:
- Only for reading
- Hidden
- Directory
- Archive

Enter any key to continue
```

Рисунок 2.4 (Изменение атрибутов папки и их просмотр)



```
OPTIONS:
0 - for EXIT
1 - for DRIVES
2 - for DIRECTORIES
3 - for FILES
Choose option: 3

FILES OPTION:
0 - for MAIN MENU
1 - for CREATE FILE
2 - for OPEN FILE
3 - for REMOVE FILE
4 - for COPY FILE
5 - for RENAME FILE
6 - for MOVE FILE
7 - for GET FILE ATTRIBUTES
8 - for SET FILE ATTRIBUTES
9 - for GET FILE INFORMATION BY HANDLE
10 - for SET FILE TIME
Choose option: 7

File for get attributes will be taken from C:\GitHub\course-2\OS\
Enter the name of file which attributes will be shown (Do not forget about extension): тест.txt
LOG C:\GitHub\course-2\OS\тест.txt
FILE ATTRIBUTES:
- Archive

Enter any key to continue
```

Рисунок 3.1 (Просмотр атрибутов файла)



```
FILES OPTION:
0 - for MAIN MENU
1 - for CREATE FILE
2 - for OPEN FILE
3 - for REMOVE FILE
4 - for COPY FILE
5 - for RENAME FILE
6 - for MOVE FILE
7 - for GET FILE ATTRIBUTES
8 - for SET FILE ATTRIBUTES
9 - for GET FILE INFORMATION BY HANDLE
10 - for SET FILE TIME
Choose option: 9

File for get file information by handle will be taken from C:\GitHub\course-2\OS\
Enter the name of file which attributes will be shown: тест.txt
LOG C:\GitHub\course-2\OS\тест.txt
FILE ATTRIBUTES:
- Archive
Creation time: 21.2.2025 7:59:27
Last access time: 21.2.2025 7:59:27
Last write time: 21.2.2025 7:59:27
Serial number of volume that contains that file: 1584941702
Number of links on that file: 1

Enter any key to continue
```

Рисунок 3.2 (Просмотр атрибутов файла при помощи дескриптора)

Win32 API – это низкоуровневый интерфейс, позволяющий точно управлять ресурсами операционной системы. В отличие от стандартных библиотек высокого уровня, он дает полный контроль над памятью, вводом-выводом, синхронизацией потоков и другими аспектами программирования.

Функции Win32 API (Windows Application Programming Interface) представляют собой фундаментальный набор инструментов, обеспечивающий взаимодействие программного обеспечения с операционной системой Windows. Этот интерфейс предоставляет разработчику доступ к управлению файлами, процессами, потоками, графической системой, сетевыми соединениями и множеством других аспектов работы системы.

Одним из значительных преимуществ Win32 API является наличие детальной документации от Microsoft. Практически каждая функция подробно описана на официальном сайте MSDN, где приведены примеры использования, возможные ошибки и рекомендации по применению.

Работа с Win32 API требует внимательности и понимания принципов функционирования операционной системы. Здесь важно учитывать такие нюансы, как работа с дескрипторами, использование правильных кодировок строк (WCHAR для широких строк), обработка многопоточности и корректное освобождение ресурсов. Ошибки при работе с API могут приводить к утечкам памяти, неожиданным сбоям и нарушению работы приложения.

Несмотря на наличие современных библиотек и фреймворков, Win32 API продолжает оставаться востребованным инструментом в системном программировании, разработке драйверов, высокопроизводительных приложений и даже в некоторых игровых движках, где требуется полный контроль над ресурсами.

Таким образом, использование Win32 API позволяет глубже понять внутренние механизмы операционной системы и дает возможность разрабатывать эффективные и производительные приложения.

Для упрощенного взаимодействия с приложением было введено “меню”, содержащее в себе 3 пункта: диски, папки, файлы. В каждом из пунктов добавлены соответствующие по заданию варианты действий.

Консольное приложение поддерживает работу с латиницей и кириллицей. Также в работе с приложением в некоторых действиях применяется относительный путь. Для относительного пути существует последовательность ../ обозначающая переход на один уровень вверх по каталогу, позволяя обращаться к родительским директориям относительно текущего расположения файла или исполняемой программы.

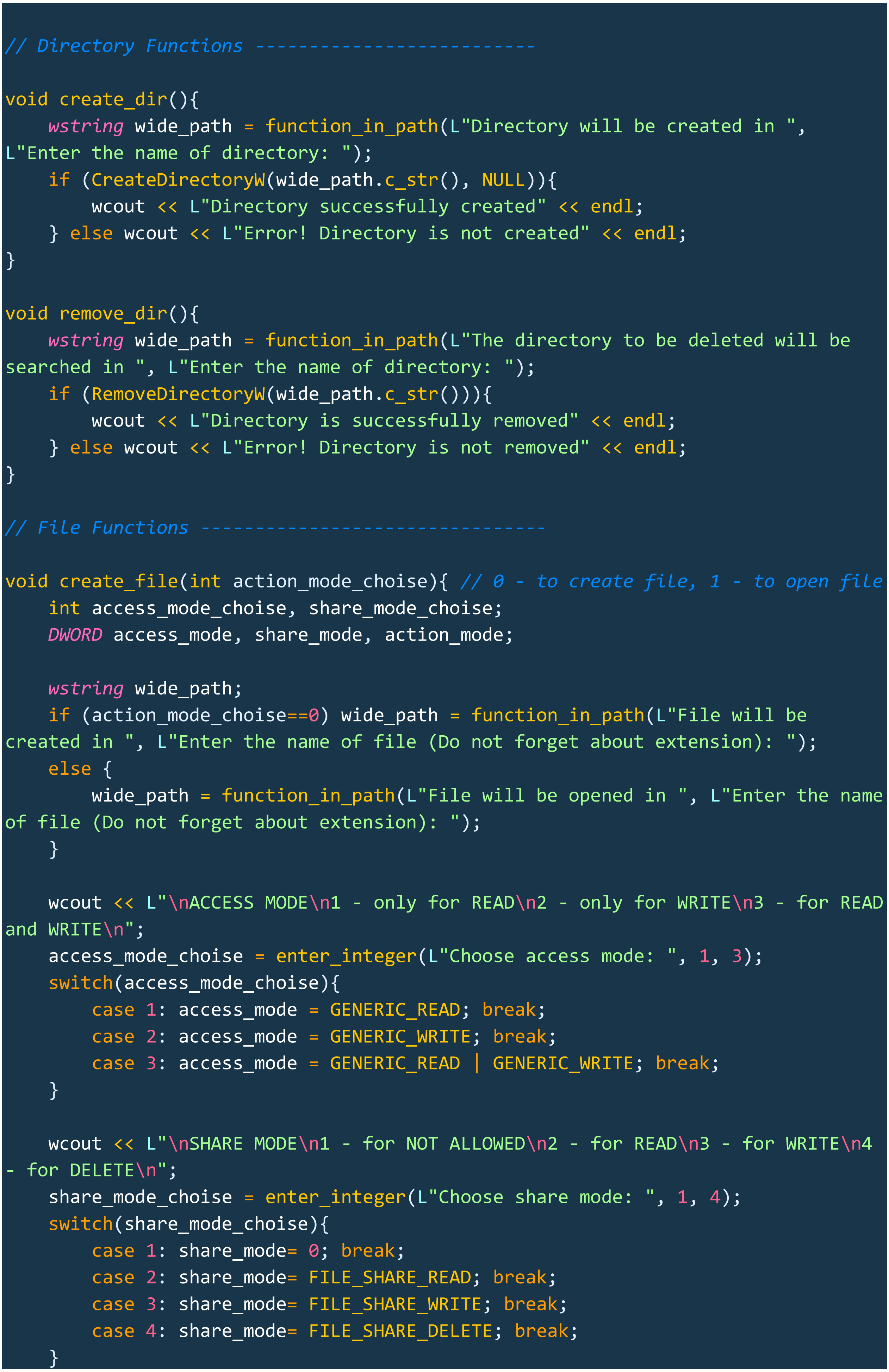
Заключение

В ходе лабораторной работы было исследовано управление файловой системой с помощью Win32 API. Win32 API предоставляет широкий набор функций для взаимодействия с операционной системой Windows на низком уровне. Оно охватывает работу с файлами, процессами, потоками, памятью, графикой и сетевыми операциями, обеспечивая точный контроль над ресурсами системы. Одним из ключевых преимуществ Win32 API является его мощность и гибкость, однако работа с ним требует внимательности: необходимо учитывать управление дескрипторами, корректное освобождение ресурсов и обработку ошибок. Благодаря обширной документации Microsoft и широкому спектру возможностей, Win32 API остается важным инструментом для системного программирования, разработки высокопроизводительных приложений и низкоуровневого управления операционной системой.

Текст программы

```cpp
#include <windows.h>
#include <iostream>
#include <string>
#include <locale>
#include <fcntl.h>
#include <io.h>

using namespace std;

// Helpful functions ---------------------------

int enter_integer(const wstring& message, int a, int b) {
    wstring input;
    int number;

    while (true) {
        wcout << message;
        getline(wcin, input);
        try {
            number = stoi(input);
            if (number >= a && number <= b) break;
            else wcout << L"Entered value is not in the range [" << a << L", " << b << L"]. Please try again!" << endl;
        } catch (...) {
            wcout << L"Error! Please try again" << endl;
        }
        wcin.clear();
    }
    return number;
}

void clear_screen() {
#if defined(_WIN32) || defined(_WIN64)
    system("cls");
#else
```

```
    system("clear");
#endif
}

int main_menu() {
    wcout << L"\nOPTIONS:" << endl;
    wcout << L"0 - for EXIT" << endl;
    wcout << L"1 - for DRIVES" << endl;
    wcout << L"2 - for DIRECTORIES" << endl;
    wcout << L"3 - for FILES" << endl;
    return enter_integer(L"Choose option: ", 0, 3);
}

int drive_menu() {
    wcout << L"\nDRIVE OPTION:" << endl;
    wcout << L"0 - for MAIN MENU" << endl;
    wcout << L"1 - for OUTPUT DRIVES" << endl;
    wcout << L"2 - for OUTPUT DRIVE INFO" << endl;
    return enter_integer(L"Choose option: ", 0, 2);
}

int dir_menu() {
    wcout << L"\nDIRECTORIES OPTION:" << endl;
    wcout << L"0 - for MAIN MENU" << endl;
    wcout << L"1 - for CREATE DIRECTORY" << endl;
    wcout << L"2 - for REMOVE DIRECTORY" << endl;
    wcout << L"3 - for RENAME DIRECTORY" << endl;
    wcout << L"4 - for MOVE DIRECTORY" << endl;
    wcout << L"5 - for GET DIRECTORY ATTRIBUTES" << endl;
    wcout << L"6 - for SET DIRECTORY ATTRIBUTES" << endl;
    return enter_integer(L"Choose option: ", 0, 6);
}

int file_menu() {
    wcout << L"\nFILES OPTION:" << endl;
    wcout << L"0 - for MAIN MENU" << endl;
    wcout << L"1 - for CREATE FILE" << endl;
    wcout << L"2 - for OPEN FILE" << endl;
    wcout << L"3 - for REMOVE FILE" << endl;
    wcout << L"4 - for COPY FILE" << endl;
    wcout << L"5 - for RENAME FILE" << endl;
    wcout << L"6 - for MOVE FILE" << endl;
    wcout << L"7 - for GET FILE ATTRIBUTES" << endl;
    wcout << L"8 - for SET FILE ATTRIBUTES" << endl;
    wcout << L"9 - for GET FILE INFORMATION BY HANDLE" << endl;
    wcout << L"10 - for SET FILE TIME" << endl;
    return enter_integer(L"Choose option: ", 0, 10);
}

void wait_reaction(){
    wcout << L"\nEnter any key to continue" << endl;
    wcin.ignore();
    // getwchar();
}

wstring get_current_dir(){
    wchar_t buffer[MAX_PATH];
    DWORD len = GetCurrentDirectoryW(MAX_PATH, buffer);
    if (len==0) return L"";
    else return wstring(buffer);
}

wstring function_in_path(const wstring& msg_to_path = L"Function will be executed in that path: ", const wstring& msg_to_enter = L"Enter the name: ") {
    wstring path = get_current_dir();
    wstring name;
    wcout << L"\n" << msg_to_path << path << L"\\" << endl;
    wcout << msg_to_enter;
    getline(wcin, name);
    wstring wide_path = path + L"\\" + name;
    wcout << L"LOG " << wide_path << endl;
    return wide_path;
}

// Drives Functions ---------------------------

DWORD get_logical_drives(){
    DWORD drives_bm = GetLogicalDrives();
    int count=1;
    wcout << L"\nAVAILABLE DRIVES:\n";
    for (int i=0; i<sizeof(DWORD)*8; i++){
        if (drives_bm & (1<<i)){
            wchar_t drive = L'A'+i;
            wcout << L" " << count << L" - " << drive << L":\\\n";
            count++;
        }
    }
    return drives_bm;
}

void print_sys_flags(DWORD flags){
    wcout << L"SYSTEM FLAGS:" << endl;
    if (flags & FILE_CASE_SENSITIVE_SEARCH) wcout << L"- File sensitive to uppercase and lowercase" << endl;
    if (flags & FILE_CASE_PRESERVED_NAMES) wcout << L"- Retained registry of file names is supported" << endl;
    if (flags & FILE_UNICODE_ON_DISK) wcout << L"- Unicode supported" << endl;
    if (flags & FILE_PERSISTENT_ACLS) wcout << L"- ACL access supported" << endl;
    if (flags & FILE_FILE_COMPRESSION) wcout << L"- File can be compressed" << endl;
    if (flags & FILE_VOLUME_QUOTAS) wcout << L"- Disk quotas are supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_SPARSE_FILES) wcout << L"- Sparse files are
supported on the volume" << endl;
    if (flags & FILE_SUPPORTS_REPARSE_POINTS) wcout << L"- Reparse points are
supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_REMOTE_STORAGE) wcout << L"- Remote storage is
supported by the file system" << endl;
    if (flags & FILE_RETURNS_CLEANUP_RESULT_INFO) wcout << L"- File system
returns cleanup result info on successful cleanup" << endl;
    if (flags & FILE_SUPPORTS_POSIX_UNLINK_RENAME) wcout << L"- POSIX-style
unlink and rename operations are supported" << endl;
    if (flags & FILE_VOLUME_IS_COMPRESSED) wcout << L"- The specified volume is a
compressed volume" << endl;
    if (flags & FILE_SUPPORTS_OBJECT_IDS) wcout << L"- Object identifiers are
supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_ENCRYPTION) wcout << L"- Encrypted file system
(EFS) is supported on the specified volume" << endl;
    if (flags & FILE_NAMED_STREAMS) wcout << L"- Named streams are supported on
the specified volume" << endl;
    if (flags & FILE_READ_ONLY_VOLUME) wcout << L"- The specified volume is read-
only" << endl;
    if (flags & FILE_SEQUENTIAL_WRITE_ONCE) wcout << L"- Sequential write-once is
supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_TRANSACTIONS) wcout << L"- Transactions are
supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_HARD_LINKS) wcout << L"- Hard links are supported
on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_EXTENDED_ATTRIBUTES) wcout << L"- Extended
attributes are supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_OPEN_BY_FILE_ID) wcout << L"- Opening by FileID is
supported by the file system" << endl;
    if (flags & FILE_SUPPORTS_USN_JOURNAL) wcout << L"- Update Sequence Number
(USN) journaling is supported on the specified volume" << endl;
    if (flags & FILE_SUPPORTS_INTEGRITY_STREAMS) wcout << L"- Integrity streams
are supported by the file system" << endl;
    if (flags & FILE_SUPPORTS_BLOCK_REFCOUNTING) wcout << L"- Logical cluster
sharing between files on the same volume is supported" << endl;
    if (flags & FILE_SUPPORTS_SPARSE_VDL) wcout << L"- Sparse valid data length
(VDL) tracking is supported by the file system" << endl;
    if (flags & FILE_DAX_VOLUME) wcout << L"- The specified volume is a Direct
Access (DAX) volume" << endl;
    if (flags & FILE_SUPPORTS_GHOSTING) wcout << L"- Ghosting is supported by the
file system" << endl;
}

void output_drive_info(){
    // Drive select
    wchar_t disk;
    wchar_t path[4];
    int option, counter=0;
    DWORD options = get_logical_drives();
    for (int i=0; i<sizeof(DWORD)*8; i++){
        if (options & 1<<i) counter++;
    }
    option = enter_integer(L"Enter the number of drive: ", 1, counter);

    counter = 1;
    for (int i=0; i<sizeof(DWORD)*8; i++){
        if (options & 1<<i){
            if (counter==option){
                disk = L'A'+i;
                break;
            } else counter++;
        }
    }

    wcout << L"\nInfo about drive '" << (wchar_t)disk << L":\\\':" << endl;
    swprintf(path, 4, L"%c:\\", disk);
    // Drive info
    int drive_type = GetDriveTypeW(path);
    switch (drive_type){
    case 0: wcout << L"Unknown type of drive" << endl; break;
    case 1: wcout << L"Incorrect root" << endl; break;
    case 2: wcout << L"Drive is removable" << endl; break;
    case 3: wcout << L"Drive is fixed" << endl; break;
    case 4: wcout << L"Drive is remotable" << endl; break;
    case 5: wcout << L"Drive is CD-ROM" << endl; break;
    case 6: wcout << L"Drive is RAM disk" << endl; break;
    }

    wchar_t volume_name[MAX_PATH], file_system_name[MAX_PATH];
    DWORD serial_number, max_length, sys_flags;
    if (GetVolumeInformationW(path, volume_name,sizeof(volume_name),
&serial_number, &max_length, &sys_flags, file_system_name,
sizeof(file_system_name))){
        wcout << L"Volume Name: " << volume_name << endl;
        wcout << L"Serial Number: " << serial_number << endl;
        wcout << L"Max Component Length: " << max_length << endl;
        wcout << L"File System Name: " << file_system_name << endl;
        print_sys_flags(sys_flags);
    } else wcout << L"Error! Please try again";

    DWORD sector_per_cluster, bytes_per_sector, numb_free_cluster,
numb_total_cluster;
    GetDiskFreeSpaceW(path, &sector_per_cluster, &bytes_per_sector,
&numb_free_cluster, &numb_total_cluster);
    ULONGLONG total =
(ULONGLONG)numb_total_cluster*sector_per_cluster*bytes_per_sector;
    ULONGLONG free =
(ULONGLONG)numb_free_cluster*sector_per_cluster*bytes_per_sector;
    wcout << L"Total space on drive: " << total/1e9 << L" GB" << endl;
    wcout << L"Free space on drive: " << free/1e9 << L" GB"<<endl;
}
```



```
// Directory Functions --------------------------

void create_dir(){
    wstring wide_path = function_in_path(L"Directory will be created in ",
L"Enter the name of directory: ");
    if (CreateDirectoryW(wide_path.c_str(), NULL)){
        wcout << L"Directory successfully created" << endl;
    } else wcout << L"Error! Directory is not created" << endl;
}

void remove_dir(){
    wstring wide_path = function_in_path(L"The directory to be deleted will be
searched in ", L"Enter the name of directory: ");
    if (RemoveDirectoryW(wide_path.c_str())){
        wcout << L"Directory is successfully removed" << endl;
    } else wcout << L"Error! Directory is not removed" << endl;
}

// File Functions --------------------------------

void create_file(int action_mode_choise){ // 0 - to create file, 1 - to open file
    int access_mode_choise, share_mode_choise;
    DWORD access_mode, share_mode, action_mode;

    wstring wide_path;
    if (action_mode_choise==0) wide_path = function_in_path(L"File will be
created in ", L"Enter the name of file (Do not forget about extension): ");
    else {
        wide_path = function_in_path(L"File will be opened in ", L"Enter the name
of file (Do not forget about extension): ");
    }

    wcout << L"\nACCESS MODE\n1 - only for READ\n2 - only for WRITE\n3 - for READ
and WRITE\n";
    access_mode_choise = enter_integer(L"Choose access mode: ", 1, 3);
    switch(access_mode_choise){
        case 1: access_mode = GENERIC_READ; break;
        case 2: access_mode = GENERIC_WRITE; break;
        case 3: access_mode = GENERIC_READ | GENERIC_WRITE; break;
    }

    wcout << L"\nSHARE MODE\n1 - for NOT ALLOWED\n2 - for READ\n3 - for WRITE\n4
- for DELETE\n";
    share_mode_choise = enter_integer(L"Choose share mode: ", 1, 4);
    switch(share_mode_choise){
        case 1: share_mode= 0; break;
        case 2: share_mode= FILE_SHARE_READ; break;
        case 3: share_mode= FILE_SHARE_WRITE; break;
        case 4: share_mode= FILE_SHARE_DELETE; break;
    }
    if (action_mode_choise==0) action_mode = CREATE_NEW;
    else action_mode = OPEN_EXISTING;

    HANDLE new_file = CreateFileW(wide_path.c_str(), access_mode, share_mode,
NULL, action_mode, FILE_ATTRIBUTE_NORMAL, NULL);
    if (new_file == INVALID_HANDLE_VALUE) {
        DWORD error = GetLastError();
        if (error == ERROR_FILE_EXISTS) {
            wcout << L"\nFile with equal name exists. Do you want to rewrite?\n1
- YES\n2 - NO\nYour choice: ";
            int rewrite_choise;
            wcin >> rewrite_choise;
            if (rewrite_choise == 1){
                if (action_mode_choise==0) action_mode = CREATE_ALWAYS;
                else action_mode = OPEN_ALWAYS;

                new_file = CreateFileW(wide_path.c_str(), access_mode,
share_mode, NULL, action_mode, FILE_ATTRIBUTE_NORMAL, NULL);
                if (new_file == INVALID_HANDLE_VALUE) {
                    wcout << L"\nError! Failed to rewrite file" << endl;
                } else {
                    wcout << L"\nFile successfully rewritten" << endl;
                    CloseHandle(new_file);
                }
            } else wcout << L"File not rewritten." << endl;
        } else {
            if (error!=0){
                wcout << L"Error! Please try again";
            }
        }
    } else {
        if (action_mode_choise==0){
            wcout << L"\nFile successfully created!" << endl;
        } else wcout << L"\nFile successfully opened!" << endl;
        CloseHandle(new_file);
    }
}

void remove_file(){
    wstring wide_path = function_in_path(L"File will be deleted in ", L"Enter the
name of file (Do not forget about extension): ");
    if(DeleteFileW(wide_path.c_str())){
        wcout << L"\nFile was successfully removed\n";
    } else wcout << L"Error! File was not removed";
}

void copy_file(){
    wstring wide_path = function_in_path(L"File will be copied from ", L"Enter
the name of file to copy (Do not forget about extension): ");
```

```
    wstring wide_repath = function_in_path(L"File will be pasted in ", L"Enter
the new name of copied file: ");

    if(CopyFileW(wide_path.c_str(), wide_repath.c_str(), TRUE)){
        wcout << L"\nFile was successfully copied\n";
    } else {
        wcout << L"\nFile with equal name exists. Do you want to rewrite?\n1 -
YES\n2 - NO\nYour choice: ";
        int rewrite_choise;
        wcin >> rewrite_choise;
        if (rewrite_choise == 1){
            if(CopyFileW(wide_path.c_str(), wide_repath.c_str(), FALSE)){
                wcout << L"File was successfully copied";
            }
            else wcout << L"Error! Please try again";
        }
    }
}

void rename(int is_dir){
    wstring wide_path, wide_repath;
    if(is_dir==0){
        wide_path = function_in_path(L"File to rename will be from ", L"Enter the
name of file to rename (Do not forget about extension): ");
        wide_repath = function_in_path(L"Renamed file will be kept in ", L"Enter
the new name of file: ");
    } else {
        wide_path = function_in_path(L"Directory to rename will be from ",
L"Enter the name of directory to rename: ");
        wide_repath = function_in_path(L"Renamed directory will be kept in ",
L"Enter the new name of directory: ");
    }

    if(MoveFileW(wide_path.c_str(), wide_repath.c_str())){
        if (is_dir==0) wcout << L"\nFile was successfully renamed";
        else wcout << L"\nDirectory was successfully renamed";
    } else {
        if (is_dir==0) wcout << L"\nFile was not renamed";
        else wcout << L"\nDirectory was not renamed";
    }
}

void move(int is_dir){
    wstring path = get_current_dir();
    wstring name, repath;
    if (is_dir==0) {
        wcout << L"\nFile will be moved according to this path: " << path <<
L"\\" << endl;
        wcout << L"Enter the name of file that will be moved (Do not forget about
extension): ";
    }
    else {
        wcout << L"\nDirectory will be moved according to this path: " << path <<
L"\\" << endl;
        wcout << L"Enter the name of directory that will be moved: ";
    }

    getline(wcin, name);
    wstring wide_path = path + L"\\" + name;
    wcout << L"Specify a relative path to move: ";
    getline(wcin, repath);
    wstring wide_repath = path + L"\\" + repath + L"\\" + name;

    if(MoveFileW(wide_path.c_str(), wide_repath.c_str())){
        wcout << L"File was successfully moved";
    } else {
        if (GetLastError() == ERROR_ALREADY_EXISTS){
            int replace = enter_integer(L"File with equal name exists. Do you
want to replace it? (0 - for NO / 1 - for YES):",0, 1);
            if (replace==0) wcout << L"Operation of moving is canceled";
            else MoveFileExW(wide_path.c_str(), wide_repath.c_str(), 0x01);
        }
    }
}

void print_file_attr(DWORD attributes) {
    wcout << L"FILE ATTRIBUTES:" << endl;

    if (attributes & FILE_ATTRIBUTE_READONLY) wcout << L"- Only for reading" <<
endl;
    if (attributes & FILE_ATTRIBUTE_HIDDEN) wcout << L"- Hidden" << endl;
    if (attributes & FILE_ATTRIBUTE_SYSTEM) wcout << L"- System file" << endl;
    if (attributes & FILE_ATTRIBUTE_DIRECTORY) wcout << L"- Directory" << endl;
    if (attributes & FILE_ATTRIBUTE_ARCHIVE) wcout << L"- Archive" << endl;
    if (attributes & FILE_ATTRIBUTE_NORMAL) wcout << L"- Default file" << endl;
    if (attributes & FILE_ATTRIBUTE_TEMPORARY) wcout << L"- Temporary file" <<
endl;
    if (attributes & FILE_ATTRIBUTE_COMPRESSED) wcout << L"- Compressed" << endl;
    if (attributes & FILE_ATTRIBUTE_ENCRYPTED) wcout << L"- Encrypted" << endl;
    if (attributes & FILE_ATTRIBUTE_VIRTUAL) wcout << L"- Virtual file" << endl;
}

void get_file_attr(int is_dir){
    wstring wide_path;
    if(is_dir==0){
        wide_path = function_in_path(L"File for get attributes will be taken from
", L"Enter the name of file which attributes will be shown (Do not forget about
extension): ");
    } else {
        wide_path = function_in_path(L"Directory for get attributes will be taken
from ", L"Enter the name of directory which attributes will be shown: ");
    }
    DWORD attributes = GetFileAttributesW(wide_path.c_str());
    if (attributes == INVALID_FILE_ATTRIBUTES) wcout << L"Error! Please try
again";
    else print_file_attr(attributes);
}

void set_file_attr(int is_dir){
    wstring wide_path;
    if (is_dir==0){
        wide_path = function_in_path(L"File for set attributes will be taken from
", L"Enter the name of file which attributes will be set (Do not forget about
extension): ");
    } else {
        wide_path = function_in_path(L"Directory for set attributes will be taken
from ", L"Enter the name of directory which attributes will be set: ");
    }

    DWORD attributes = GetFileAttributesW(wide_path.c_str());
    if (attributes == INVALID_FILE_ATTRIBUTES) {
        wcout << L"\nError! Please try again\n";
        return;
    }
    wcout << L"\nFor each attribute enter 0 (No) or 1 (Yes) to set it\n";
    int choice;
    wcout << L"\nOnly for reading? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_READONLY;
    else attributes &= ~FILE_ATTRIBUTE_READONLY;

    wcout << L"Hidden? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_HIDDEN;
    else attributes &= ~FILE_ATTRIBUTE_HIDDEN;

    wcout << L"System file? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_SYSTEM;
    else attributes &= ~FILE_ATTRIBUTE_SYSTEM;

    wcout << L"Archived? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_ARCHIVE;
    else attributes &= ~FILE_ATTRIBUTE_ARCHIVE;

    wcout << L"Temporary file? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_TEMPORARY;
    else attributes &= ~FILE_ATTRIBUTE_TEMPORARY;

    wcout << L"Compressed file? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_COMPRESSED;
    else attributes &= ~FILE_ATTRIBUTE_COMPRESSED;

    wcout << L"Encrypted file? (1 - Yes, 0 - No): ";
    wcin >> choice;
    if (choice == 1) attributes |= FILE_ATTRIBUTE_ENCRYPTED;
    else attributes &= ~FILE_ATTRIBUTE_ENCRYPTED;

    if (SetFileAttributesW(wide_path.c_str(), attributes)) {
        wcout << L"\nAttributes were successfully set\n";
    } else wcout << L"Error! Attributes were not set. Please try again";
}

void print_filetime(const FILETIME& ft) {
    SYSTEMTIME st;
    FileTimeToSystemTime(&ft, &st);
    wcout << st.wDay << L"." << st.wMonth << L"." << st.wYear << L" ";
    wcout << st.wHour << L":" << st.wMinute << L":" << st.wSecond << endl;
}

FILETIME systime_to_filetime(const SYSTEMTIME& st) {
    FILETIME ft;
    SystemTimeToFileTime(&st, &ft);
    return ft;
}

void get_file_info_by_handle(){
    wstring wide_path = function_in_path(L"File for get file information by handle will be taken from ", L"Enter the name of file which attributes will be shown: ");

    HANDLE handle_file = CreateFileW(wide_path.c_str(), GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

    if (handle_file == INVALID_HANDLE_VALUE) {
        DWORD error = GetLastError();
        if (error == ERROR_FILE_NOT_FOUND){
            wcout << L"Error! File to get info is not exist" << endl;
        }
        if (error == ERROR_ACCESS_DENIED) {
            wcout << L"Access denied. You do not have permission to open this file" << endl;
        } else if (error == ERROR_SHARING_VIOLATION) {
            wcout << L"File is being used by another process and cannot be opened" << endl;
        } else {
            wcout << L"Failed to open file" << endl;
        }
        return;
    }
    BY_HANDLE_FILE_INFORMATION file_info;
    FILETIME creation_time, laccess_time, lwrite_time;

    if (GetFileInformationByHandle(handle_file, &file_info)){
        print_file_attr(file_info.dwFileAttributes);

        if (GetFileTime(handle_file, &creation_time, &laccess_time,
&lwrite_time)){
            wcout << L"Creation time: ";
            print_filetime(creation_time);

            wcout << L"Last access time: ";
            print_filetime(laccess_time);

            wcout << L"Last write time: ";
            print_filetime(lwrite_time);
        } else {
            wcout << L"Error! Cant get file time" << endl;
        }

        wcout << L"Serial number of volume that contains that file: " <<
file_info.dwVolumeSerialNumber << endl;

        wcout << L"Number of links on that file: " << file_info.nNumberOfLinks <<
endl;
    }
    CloseHandle(handle_file);
}

void set_file_time(){
    wstring wide_path = function_in_path(L"File for set file time will be taken
from ", L"Enter the name of file which file time will be set (Do not forget about
extension): ");

    HANDLE handle_file = CreateFileW(wide_path.c_str(), FILE_WRITE_ATTRIBUTES,
FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (handle_file == INVALID_HANDLE_VALUE) {
        wcout << L"Error! Please try again" << endl;
        return;
    }

    SYSTEMTIME st;
    FILETIME ftCreation, ftLastAccess, ftLastWrite;

    int choice = enter_integer(L"Set new creation time? (1 - yes, 0 - no): ",0,
1);
    if (choice) {
        wcout << L"Enter creation date (YYYY MM DD HH MM SS): ";
        wcin >> st.wYear >> st.wMonth >> st.wDay >> st.wHour >> st.wMinute >>
st.wSecond;
        ftCreation = systime_to_filetime(st);
    } else {
        ftCreation.dwLowDateTime = 0;
        ftCreation.dwHighDateTime = 0;
    }

    wcout << L"Set new last access time? (1 - yes, 0 - no): ";
    wcin >> choice;
    if (choice) {
        wcout << L"Enter last access date (YYYY MM DD HH MM SS): ";
        wcin >> st.wYear >> st.wMonth >> st.wDay >> st.wHour >> st.wMinute >>
st.wSecond;
        ftLastAccess = systime_to_filetime(st);
    } else {
        ftLastAccess.dwLowDateTime = 0;
        ftLastAccess.dwHighDateTime = 0;
    }

    wcout << L"Set new last write time? (1 - yes, 0 - no): ";
    wcin >> choice;
    if (choice) {
        wcout << L"Enter last write date (YYYY MM DD HH MM SS): ";
        wcin >> st.wYear >> st.wMonth >> st.wDay >> st.wHour >> st.wMinute >>
st.wSecond;
        ftLastWrite = systime_to_filetime(st);
    } else {
        ftLastWrite.dwLowDateTime = 0;
        ftLastWrite.dwHighDateTime = 0;
    }

    if (SetFileTime(handle_file, (ftCreation.dwLowDateTime == 0 &&
ftCreation.dwHighDateTime == 0) ? NULL : &ftCreation, (ftLastAccess.dwLowDateTime
== 0 && ftLastAccess.dwHighDateTime == 0) ? NULL : &ftLastAccess,
(ftLastWrite.dwLowDateTime == 0 && ftLastWrite.dwHighDateTime == 0) ? NULL :
&ftLastWrite)) {
        wcout << L"File time successfully updated" << endl;
    } else wcout << L"Error! File time was not updated" << endl;

    CloseHandle(handle_file);
}

// main -----------------------------------------

int main() {
    _setmode(_fileno(stdout), _O_U16TEXT);
    _setmode(_fileno(stdin), _O_U16TEXT);

    int option;
    do {
        option = main_menu();
        switch (option) {
            case 1: {
                int drive_option;
                do {
                    drive_option = drive_menu();
                    switch(drive_option){
                    case 0: break;
                    case 1: get_logical_drives(); wait_reaction(); break;
                    case 2: output_drive_info(); wait_reaction(); break;
                    }
                } while (drive_option != 0);
                break;
            }
            case 2: {
                int dir_option;
                do {
                    dir_option = dir_menu();
                    switch(dir_option){
                    case 0: break;
                    case 1: create_dir(); wait_reaction(); break;
                    case 2: remove_dir(); wait_reaction(); break;
                    case 3: rename(1); wait_reaction(); break;
                    case 4: move(1); wait_reaction(); break;
                    case 5: get_file_attr(1); wait_reaction(); break;
                    case 6: set_file_attr(1); wait_reaction(); break;
                    }
                } while (dir_option != 0);
                break;
            }
            case 3: {
                int file_option;
                do {
                    file_option = file_menu();
                    switch (file_option){
                    case 0: break;
                    case 1: create_file(0); wait_reaction(); break;
                    case 2: create_file(1); wait_reaction(); break;
                    case 3: remove_file(); wait_reaction(); break;
                    case 4: copy_file(); wait_reaction(); break;
                    case 5: rename(0); wait_reaction(); break;
                    case 6: move(0); wait_reaction(); break;
                    case 7: get_file_attr(0); wait_reaction(); break;
                    case 8: set_file_attr(0); wait_reaction(); break;
                    case 9: get_file_info_by_handle(); wait_reaction(); break;
                    case 10: set_file_time(); wait_reaction(); break;
                    }
                } while (file_option != 0);
                break;
            }
        }
    } while (option != 0);
    wcout << L"\nGoodbye!\n" << endl;
    return 0;
```

```
}
```