

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЁТ
по лабораторной работе №1
по дисциплине «Операционные системы»
Тема: Управление файловой системой

Студент гр. 9308

Преподаватель

Соболев М.С.

Тимофеев А.В.

Санкт-Петербург,

2021

Оглавление

1. Введение	3
2. Управление дисками, каталогами и файлами	6
2.1. Вывод списка дисков.....	6
2.2. Вывод информации о диске и размер свободного пространства.....	7
2.3. Создание и удаление заданных каталогов.....	10
2.4. Создание файлов в новых каталогах	12
2.5. Копирование и перемещение файлов между каталогами.....	15
2.6. Анализ и изменение атрибутов файлов.....	18
2.7. Исходный код программы.....	24
2.8. Выводы	80
3. Копирование файла с помощью перекрывающихся операций ввода-вывода	81
3.1. Создание и запуск консольного приложения.....	81
3.2. Проверка приложения на разных размерах копируемых блоков	82
3.3. Проверка приложения на разном числе операция ввода-вывода.....	84
3.4. Исходный код программы.....	85
3.5. Выводы	94
4. Список использованных источников.....	95

1. Введение

Тема работы: Управление файловой системой.

Цель работы: Исследование управления файловой системой с помощью Win32 API.

Указания к выполнению

Задание 1.1. Управление дисками, каталогами и файлами.

1. Создайте 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, GetLogicalDriveStrings);
- 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).

2. Запустите приложение и проверьте его работоспособность на нескольких наборах вводимых данных. Запротоколируйте результаты в отчёт. Дайте свои комментарии в отчёте относительно выполнения функций Win32 API.

3. Подготовьте итоговый отчёт с развернутыми выводами по заданию.

Задание 1.2. Копирование файла с помощью операций перекрывающегося ввода-вывода. Приложение должно копировать существующий файл в новый файл, «одновременно» выполняя n перекрывающихся операций ввода-вывода (механизм APC) блоками данных кратными размеру кластера. Указания к выполнению.

1. Создайте консольное приложение, которое выполняет:

- открытие/создание файлов (функция Win32 API – `CreateFile`, обязательно использовать флаги `FILE_FLAG_NO_BUFFERING` и `FILE_FLAG_OVERLAPPED`);
- файловый ввод-вывод (функции Win32 API – `ReadFileEx`, `WriteFileEx`) блоками кратными размеру кластера;
- ожидание срабатывания вызова функции завершения (функция Win32 API – `SleepEx`);
- измерение продолжительности выполнения операции копирования файла (функция Win32 API – `TimeGetTime`).

2. Запустите приложение и проверьте его работоспособность на копировании файлов разного размера для ситуации с перекрывающимся выполнением одной операции ввода и одной операции вывода (для сравнения файлов используйте консольную команду `FC`). Выполните эксперимент для разного размера копируемых блоков, постройте график зависимости скорости копирования от размера блока данных. Определите оптимальный размер блока данных, при котором скорость копирования наибольшая. Запротоколируйте результаты в отчёт. Дайте свои комментарии в отчете относительно выполнения функций Win32 API.

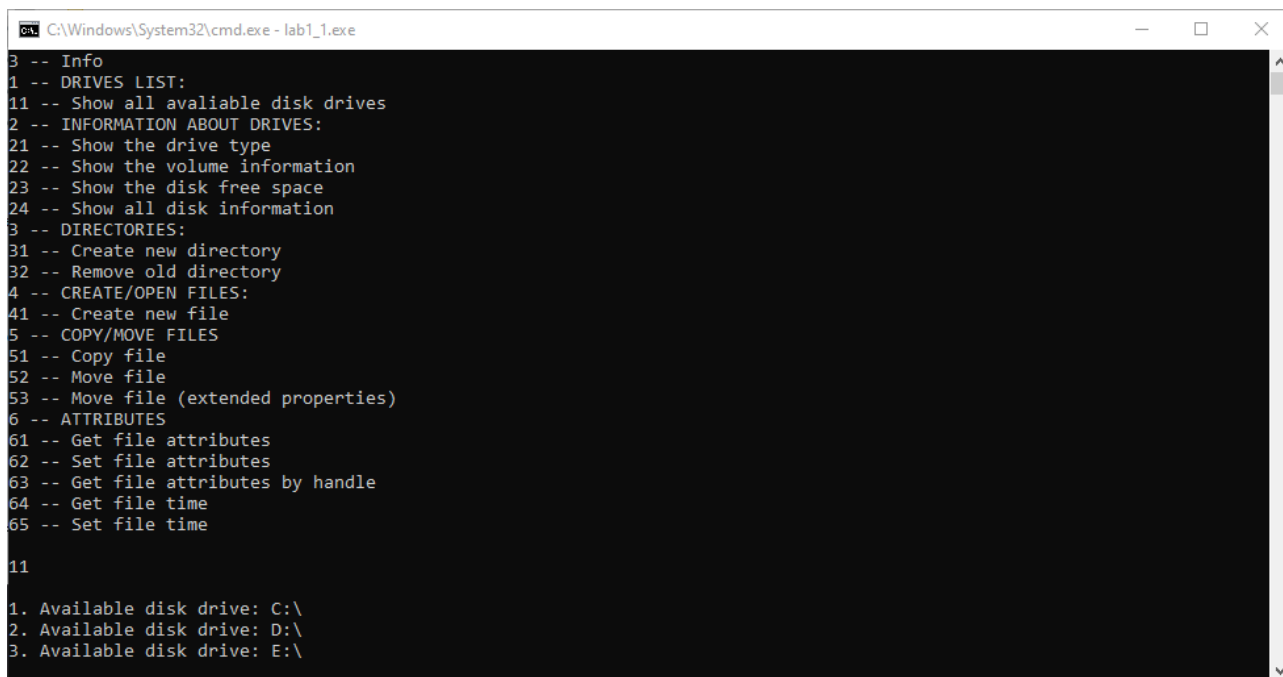
3. Произведите замеры времени выполнения приложения для разного числа перекрывающихся операций ввода и вывода (1, 2, 4, 8, 12, 16), не забывая

проверять работоспособность приложения (консольная команда FC). По результатам измерений постройте график зависимости и определите число перекрывающихся операций ввода и вывода, при котором достигается наибольшая скорость копирования файла. Запротоколируйте результаты в отчёт.

2. Управление дисками, каталогами и файлами

2.1. Вывод списка дисков

Реализация вывода списка дисков с помощью функций `GetLogicalDrives()` и `GetLogicalDriveStrings()`;



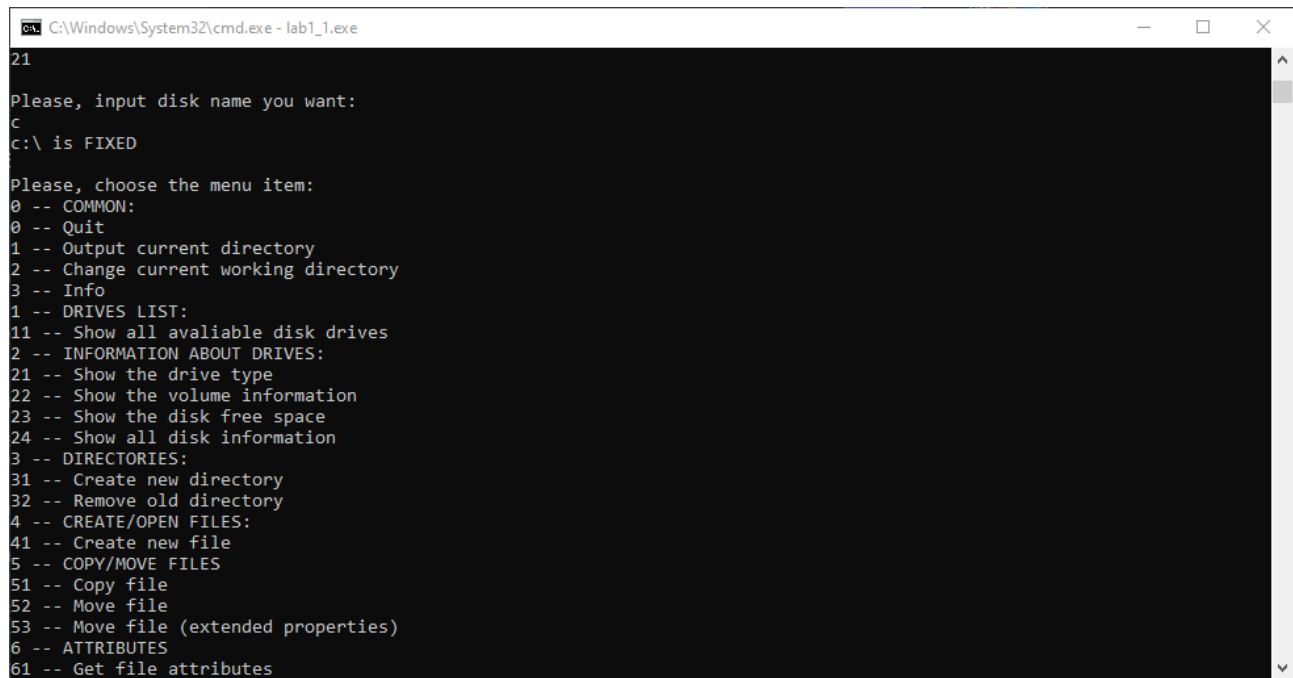
```
C:\Windows\System32\cmd.exe - lab1_1.exe
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory
32 -- Remove old directory
4 -- CREATE/OPEN FILES:
41 -- Create new file
5 -- COPY/MOVE FILES
51 -- Copy file
52 -- Move file
53 -- Move file (extended properties)
6 -- ATTRIBUTES
61 -- Get file attributes
62 -- Set file attributes
63 -- Get file attributes by handle
64 -- Get file time
65 -- Set file time

11
1. Available disk drive: C:\
2. Available disk drive: D:\
3. Available disk drive: E:\
```

Рисунок 1: Вывод списка дисков

2.2. Вывод информации о диске и размер свободного пространства

Вывод основной информации о дисках (в том числе свободное пространство, их системные данные и их системные флаги) с использованием функций `GetDriveType()`, `GetVolumeInformation()` и `GetDiskFreeSpace()`.



```
C:\Windows\System32\cmd.exe - lab1_1.exe
21
Please, input disk name you want:
c
c:\ is FIXED
Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory
32 -- Remove old directory
4 -- CREATE/OPEN FILES:
41 -- Create new file
5 -- COPY/MOVE FILES
51 -- Copy file
52 -- Move file
53 -- Move file (extended properties)
6 -- ATTRIBUTES
61 -- Get file attributes
```

Рисунок 2: Вывод типа диска

```
C:\Windows\System32\cmd.exe - lab1_1.exe

22
Please, input disk name you want:
c
Volume Name is BOOTCAMP
Volume Serial Number is 2819328019
File System is NTFS

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory
32 -- Remove old directory
4 -- CREATE/OPEN FILES:
41 -- Create new file
5 -- COPY/MOVE FILES
51 -- Copy file
52 -- Move file
```

Рисунок 3: Вывод информации о диске

```
C:\Windows\System32\cmd.exe - lab1_1.exe

23
Please, input disk name you want:
c
Disk free space information about disk c:\!
----- NET NUMBERS -----
Total Number Of Sectors per Cluster: 8
Total Number Of Bytes per Sector: 512
Total Number Of Free Clusters: 3166643
Total Number Of Clusters: 15641343
Returned value: 1
----- CALCULATED NUMBERS -----
Total Number Of Bytes per Cluster: 4096
Total Number Of Free Sectors: 25333144
Total Number Of Sectors: 125130744
Total Number Of Free Bytes: 12970569728
Total Number of Bytes: 64066940928
Total Number Of Used Clusters: 12474700
Total Number Of Used Sectors: 99797600
Total Number Of Used Bytes: 51096371200

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
```

Рисунок 4: Вывод информации о пространстве на диске


```
C:\Windows\System32\cmd.exe - lab1_1.exe
This is your new absolute file path: C:\
Commit changes? [y/n]
y
The drive "C:\" has fixed media; for example, a hard disk drive or flash drive.

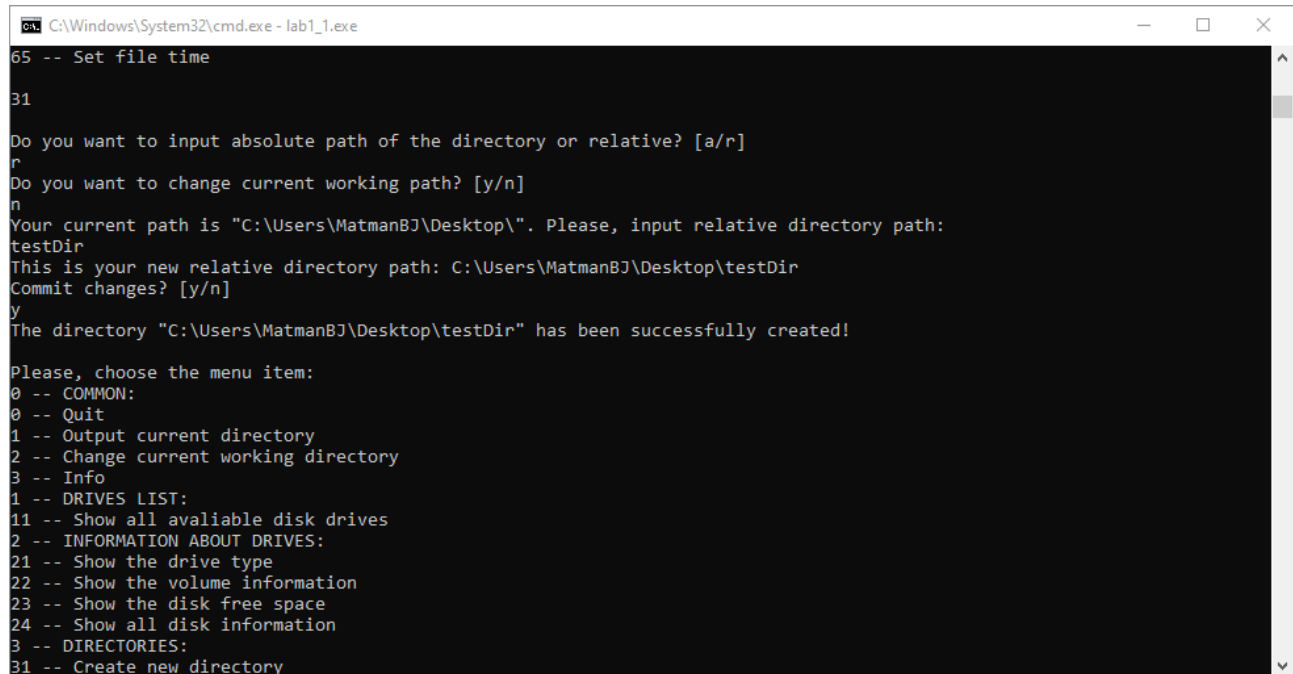
Drive name:BOOTCAMP
Type of File system:NTFS
Serial number:2819328019
System flags:
The specified volume preserved case of file names when it places a name on disk.
The specified volume supports case-sensitive file names.
The specified volume supports file-based compression.
The specified volume supports named streams.
The specified volume preserves and enforces access control lists (ACL). For example, the NTFS file system preserves and enforces ACLs, and the FAT file system does not.
The specified volume supports the Encrypted File System (EFS).
The specified volume supports extended attributes.
The specified volume supports hard links.
The specified volume supports object identifiers.
The specified volume supports open by FileID.
The specified volume supports reparse points.
The specified volume supports sparse files.
The specified volume supports transactions.
The specified volume supports update sequence number (USN) journals.
The specified volume supports Unicode in file names as they appear on disk.
The specified volume supports disk quotas.

Total drive space: 64066940928 bytes
Available drive space: 12970209280 bytes
```

Рисунок 5: Вывод всей информации о диске

2.3. Создание и удаление заданных каталогов

Реализация создания и удаления заданных каталогов с помощью функций `CreateDirectory()` и `RemoveDirectory()`.



```
C:\Windows\System32\cmd.exe - lab1_1.exe
65 -- Set file time
31
Do you want to input absolute path of the directory or relative? [a/r]
r
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative directory path:
testDir
This is your new relative directory path: C:\Users\MatmanBJ\Desktop\testDir
Commit changes? [y/n]
y
The directory "C:\Users\MatmanBJ\Desktop\testDir" has been successfully created!

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory
```

Рисунок 6: Создание каталога

GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевМС_ОС_ЛР1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_1	03.10.2021 23:47	Приложение	2 454 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "C++"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	
testDir	04.10.2021 0:00	Папка с файлами	

Рисунок 7: Директория каталога

```

C:\Windows\System32\cmd.exe - lab1_1.exe
65 -- Set file time
32
Do you want to input absolute path of the directory or relative? [a/r]
r
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative directory path:
testDir
This is your new relative directory path: C:\Users\MatmanBJ\Desktop\testDir
Commit changes? [y/n]
y
The directory "C:\Users\MatmanBJ\Desktop\testDir" has been successfully removed!

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory

```

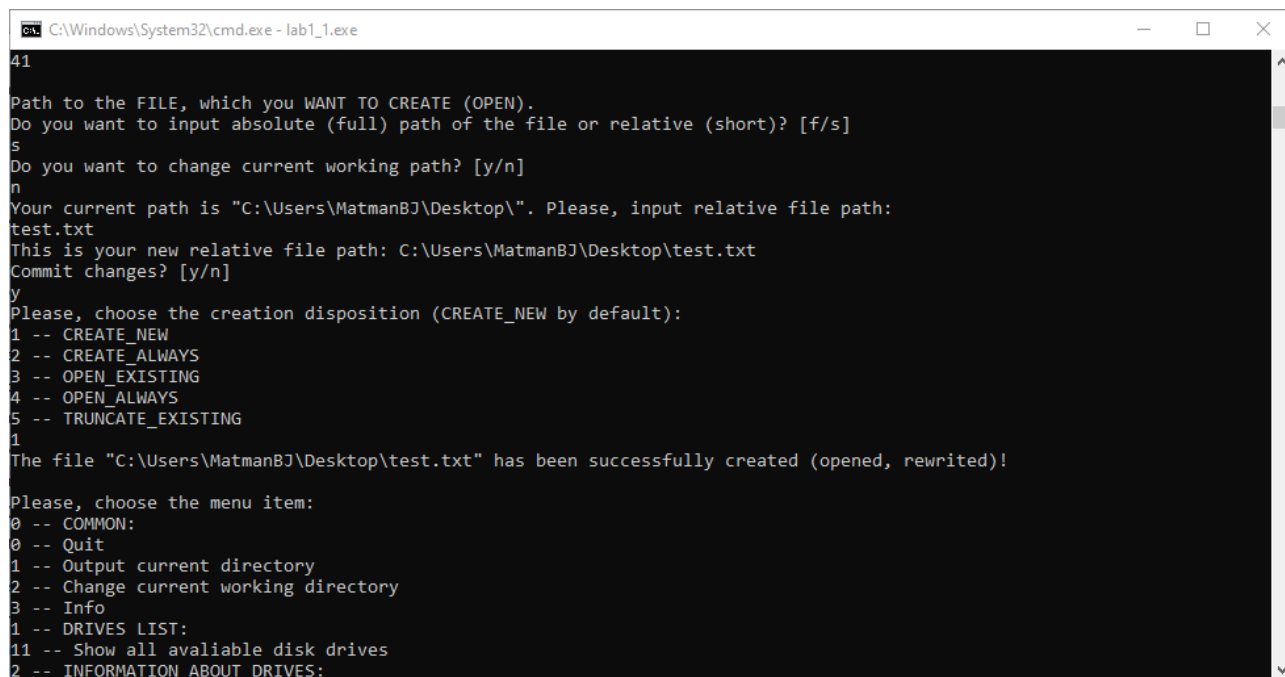
Рисунок 8: Удаление каталога

Имя	Дата изменения	Тип	Размер
GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевМС_OC_LP1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_1	03.10.2021 23:47	Приложение	2 454 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "C++"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	

Рисунок 9: Директория каталога

2.4. Создание файлов в новых каталогах

Реализация создания файлов в новых каталогах с возможностью использовать флаги для этой функции (например, для открытия) с помощью функции `CreateFile()`.



```
C:\Windows\System32\cmd.exe - lab1_1.exe
41
Path to the FILE, which you WANT TO CREATE (OPEN).
Do you want to input absolute (full) path of the file or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Please, choose the creation disposition (CREATE_NEW by default):
1 -- CREATE_NEW
2 -- CREATE_ALWAYS
3 -- OPEN_EXISTING
4 -- OPEN_ALWAYS
5 -- TRUNCATE_EXISTING
1
The file "C:\Users\MatmanBJ\Desktop\test.txt" has been successfully created (opened, rewritten)!

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
```

Рисунок 10: Создание файла

Имя	Дата изменения	Тип	Размер
GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевМС_ОС_ЛР1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_1	03.10.2021 23:47	Приложение	2 454 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "C++"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	
test	04.10.2021 0:02	Текстовый докум...	0 КБ

Рисунок 11: Директория файла

```

C:\Windows\System32\cmd.exe - lab1_1.exe
41
Path to the FILE, which you WANT TO CREATE (OPEN).
Do you want to input absolute (full) path of the file or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
y
Please, input current working directory path:
C:\Users\MatmanBJ\Desktop\
This is your new current working path: C:\Users\MatmanBJ\Desktop\
Commit changes? [y/n]
y
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Please, choose the creation disposition (CREATE_NEW by default):
1 -- CREATE_NEW
2 -- CREATE_ALWAYS
3 -- OPEN_EXISTING
4 -- OPEN_ALWAYS
5 -- TRUNCATE_EXISTING
3
The file "C:\Users\MatmanBJ\Desktop\test.txt" has been successfully created (opened, rewritten)!

Please, choose the menu item:
0 -- COMMON:
0 -- Quit

```

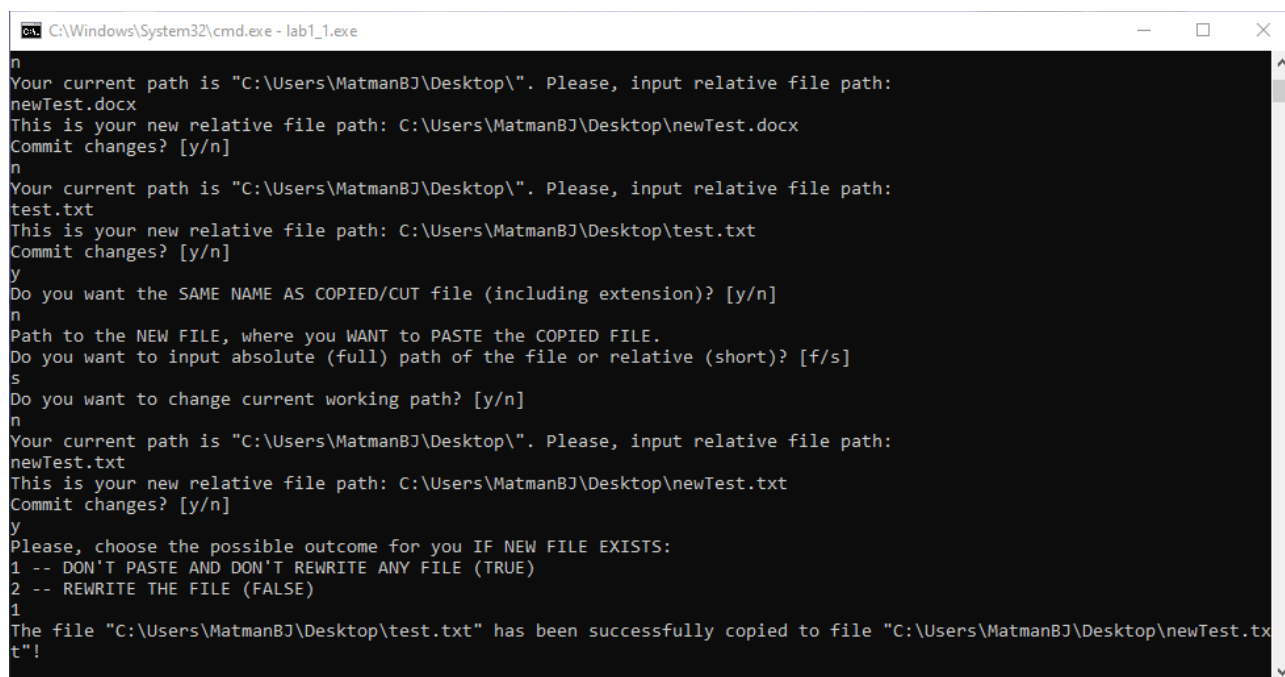
Рисунок 12: Открытие файла

Имя	Дата изменения	Тип	Размер
GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевМС_ОС_ЛР1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "C++"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	
test	04.10.2021 0:08	Текстовый докум...	1 КБ
lab1_1	04.10.2021 0:07	Приложение	2 454 КБ

Рисунок 13: Директория файла

2.5. Копирование и перемещение файлов между каталогами

Реализация копирования и перемещения файлов между каталогами (в том числе с возможностью выбрать, как поведёт себя консольное приложение при возникновении файлов с одинаковым именем) с помощью функций `CopyFile()`, `MoveFile()` и `MoveFileEx`.



```
C:\Windows\System32\cmd.exe - lab1_1.exe
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
newTest.docx
This is your new relative file path: C:\Users\MatmanBJ\Desktop\newTest.docx
Commit changes? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Do you want the SAME NAME AS COPIED/CUT file (including extension)? [y/n]
n
Path to the NEW FILE, where you WANT to PASTE the COPIED FILE.
Do you want to input absolute (full) path of the file or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
newTest.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\newTest.txt
Commit changes? [y/n]
y
Please, choose the possible outcome for you IF NEW FILE EXISTS:
1 -- DON'T PASTE AND DON'T REWRITE ANY FILE (TRUE)
2 -- REWRITE THE FILE (FALSE)
1
The file "C:\Users\MatmanBJ\Desktop\test.txt" has been successfully copied to file "C:\Users\MatmanBJ\Desktop\newTest.txt"!
```

Рисунок 14: Копирование файла

Имя	Дата изменения	Тип	Размер
GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевMC_OC_ЛР1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "CPP"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	
test	04.10.2021 0:08	Текстовый докум...	1 КБ
lab1_1	04.10.2021 0:07	Приложение	2 454 КБ
newTest	04.10.2021 0:08	Текстовый докум...	1 КБ

Рисунок 15: Директория файла

Имя	Дата изменения	Тип	Размер
GGHJ	03.10.2021 23:52	Папка с файлами	
Базы данных	30.09.2021 0:14	Папка с файлами	
Задание 1	03.10.2021 23:52	Папка с файлами	
Новая папка	03.10.2021 1:35	Папка с файлами	
Операционные системы	03.10.2021 3:47	Папка с файлами	
9308_СоболевMC_OC_ЛР1	03.10.2021 23:46	Документ Micros...	33 КБ
dd	01.01.2000 21:00	Текстовый докум...	0 КБ
Discord	22.09.2021 21:55	Ярлык	3 КБ
lab1_2	03.10.2021 23:48	Приложение	2 350 КБ
main.cpp	01.10.2021 16:24	Файл "CPP"	41 КБ
TO SORT	03.10.2021 23:59	Папка с файлами	
test	04.10.2021 0:08	Текстовый докум...	1 КБ
lab1_1	04.10.2021 0:07	Приложение	2 454 КБ
newTest	04.10.2021 0:08	Текстовый докум...	1 КБ
newDir	04.10.2021 0:12	Папка с файлами	

Рисунок 16: Директория файла до перемещения


```
C:\Windows\System32\cmd.exe - lab1_1.exe
Path to the FILE, which you WANT TO MOVE (CUT).
Do you want to input absolute (full) path of the file or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
newTest.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\newTest.txt
Commit changes? [y/n]
y
Do you want the SAME NAME AS COPIED/CUT file (including extension)? [y/n]
y
Path to the NEW DIRECTORY, where you WANT to MOVE (PASTE) the CUT FILE.
Do you want to input absolute path of the directory or relative? [a/r]
r
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative directory path:
newDir
This is your new relative directory path: C:\Users\MatmanBJ\Desktop\newDir
Commit changes? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative directory path:
newDir\
This is your new relative directory path: C:\Users\MatmanBJ\Desktop\newDir\
Commit changes? [y/n]
y
The file "C:\Users\MatmanBJ\Desktop\newTest.txt" has been successfully moved to file "C:\Users\MatmanBJ\Desktop\newDir\newTest.txt"!
```

Рисунок 17: Перемещение файла


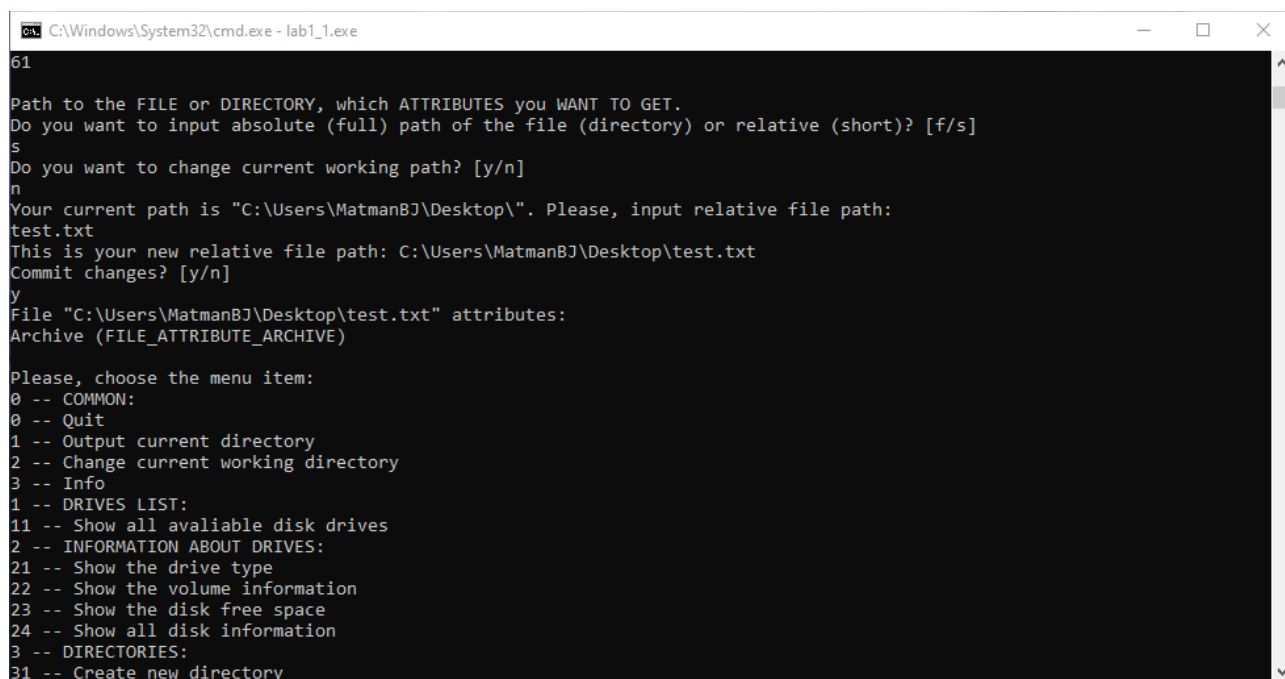
Имя	Дата изменения	Тип	Размер
 newTest	04.10.2021 0:08	Текстовый докум...	1 КБ

Рисунок 18: Директория файла после перемещения

2.6. Анализ и изменение атрибутов файлов

Реализация анализа и изменения атрибутов файлов, а также их временных данных, получения данных с помощью дескриптора с помощью функций `GetFileAttributes()`, `SetFileAttributes()`, `GetFileInformationByHandle()`, `GetFileTime()` и `SetFileTime()`.



```
C:\Windows\System32\cmd.exe - lab1_1.exe
61
Path to the FILE or DIRECTORY, which ATTRIBUTES you WANT TO GET.
Do you want to input absolute (full) path of the file (directory) or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
File "C:\Users\MatmanBJ\Desktop\test.txt" attributes:
Archive (FILE_ATTRIBUTE_ARCHIVE)

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
24 -- Show all disk information
3 -- DIRECTORIES:
31 -- Create new directory
```

Рисунок 19: Вывод атрибутов файла

```
C:\Windows\System32\cmd.exe - lab1_1.exe

62
Path to the FILE or DIRECTORY, which ATTRIBUTES you WANT TO SET.
Do you want to input absolute (full) path of the file (directory) or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Please, choose the possible attributes for the file/directory (you CAN CHOOSE MANY -- JUST SPLIT NUMBERS BY SPACE):
1 -- FILE_ATTRIBUTE_READONLY (read-only)
2 -- FILE_ATTRIBUTE_HIDDEN (hidden)
4 -- FILE_ATTRIBUTE_SYSTEM (system used)
32 -- FILE_ATTRIBUTE_ARCHIVE (archive)
128 -- FILE_ATTRIBUTE_NORMAL -- DEFAULT
256 -- FILE_ATTRIBUTE_TEMPORARY (temporary storage)
4096 -- FILE_ATTRIBUTE_OFFLINE (don't available immediatly)
8192 -- FILE_ATTRIBUTE_NOT_CONTENT_INDEXED (not indexed)
1
The file's (directory's) "C:\Users\MatmanBJ\Desktop\test.txt" attributes has been successfully changed to:
File "C:\Users\MatmanBJ\Desktop\test.txt" attributes:
Read-only (FILE_ATTRIBUTE_READONLY)

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
```

Рисунок 20: Изменение атрибутов файла

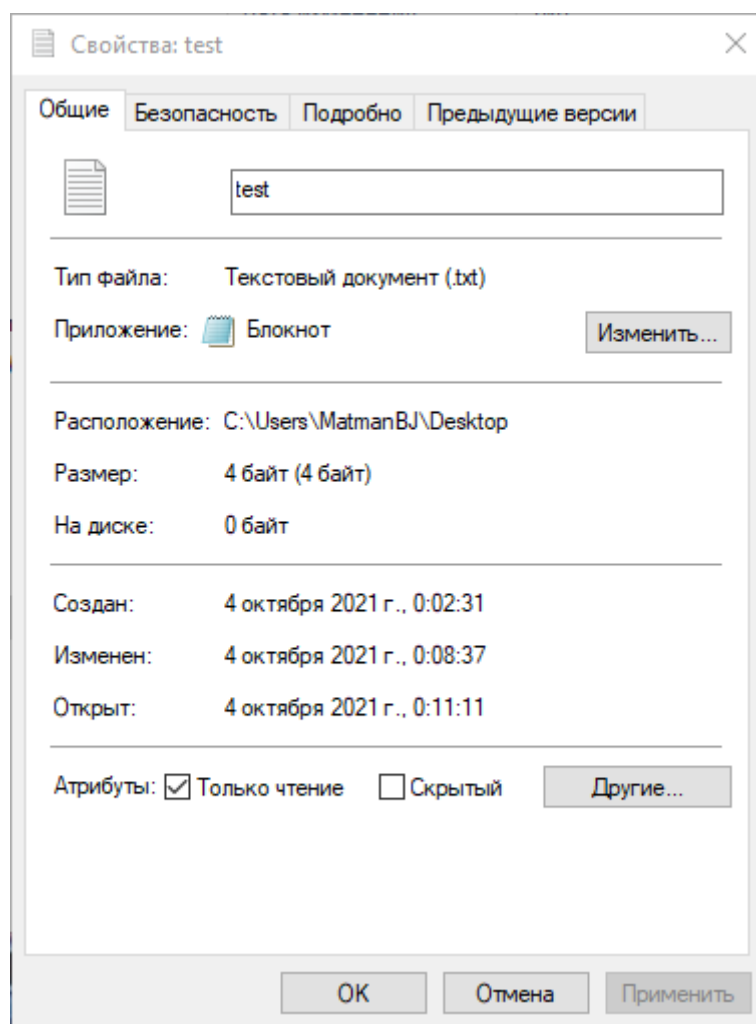


Рисунок 21: Изменённый атрибут файла в свойствах

```
C:\Windows\System32\cmd.exe - lab1_1.exe
63
Path to the FILE or DIRECTORY, which ATTRIBUTES you WANT TO GET.
Do you want to input absolute (full) path of the file (directory) or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Normal (FILE_ATTRIBUTE_NORMAL)
CREATION TIME = 2021-10-03 21:02:31.232
LAST WRITE TIME = 2021-10-03 21:08:37.696
LAST ACCESS TIME = 2021-10-03 21:16:09.947
Volume serial number: 2819328019
Local size high/low: 0 4
Number Of Links: 1
Index High/low: 1507328 46956

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
```

Рисунок 22: Вывод атрибутов файла по дескриптору

```
C:\Windows\System32\cmd.exe - lab1_1.exe
65 -- Set file time
64
Path to the FILE get time.
Do you want to input absolute (full) path of the file (directory) or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
System time = 2021-10-03 21:02:31.232
Access time = 2021-10-03 21:16:09.947
Change time = 2021-10-03 21:08:37.696

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
21 -- Show the drive type
22 -- Show the volume information
23 -- Show the disk free space
```

Рисунок 23: Вывод значений времени файла

```
C:\Windows\System32\cmd.exe - lab1_1.exe
65
Path to the FILE set time.
Do you want to input absolute (full) path of the file (directory) or relative (short)? [f/s]
s
Do you want to change current working path? [y/n]
n
Your current path is "C:\Users\MatmanBJ\Desktop\". Please, input relative file path:
test.txt
This is your new relative file path: C:\Users\MatmanBJ\Desktop\test.txt
Commit changes? [y/n]
y
Time format: 2021-10-03 18:29:40.152
Creation time:
2000-01-01 18:00:00.000
Last write time:
2000-01-01 18:00:00.000
Last access time:
2000-01-01 18:00:00.000
Time changed!

Please, choose the menu item:
0 -- COMMON:
0 -- Quit
1 -- Output current directory
2 -- Change current working directory
3 -- Info
1 -- DRIVES LIST:
11 -- Show all available disk drives
2 -- INFORMATION ABOUT DRIVES:
```

Рисунок 24: Изменение значений времени файла

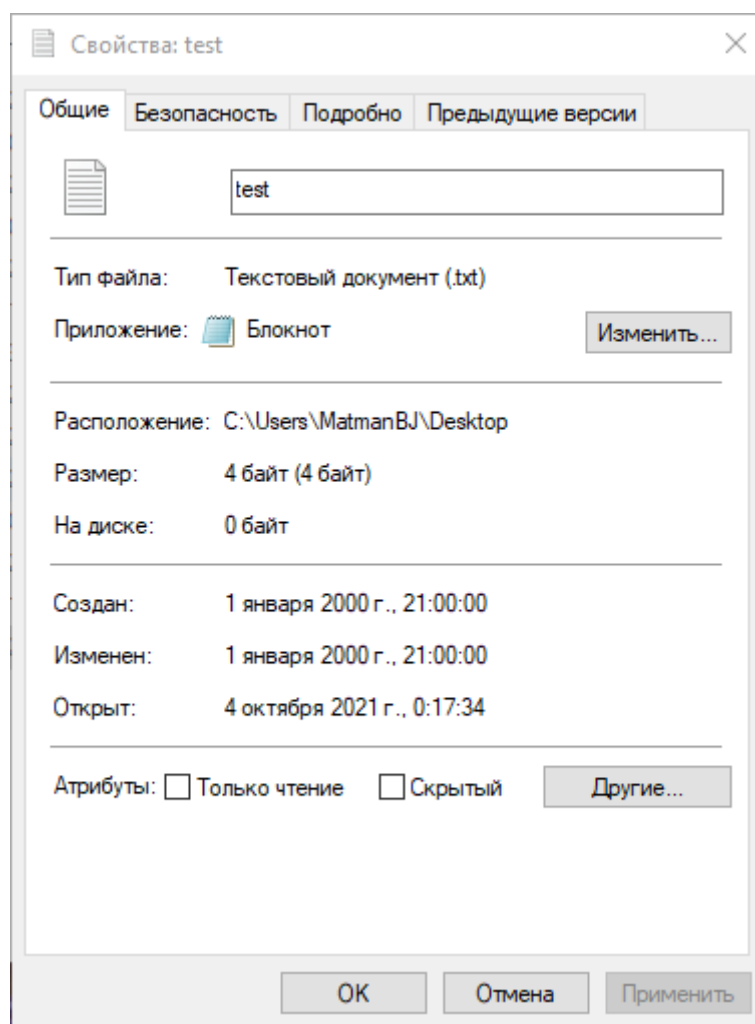


Рисунок 25: Значения времени файла в свойствах

2.7. Исходный код программы

/*

Saint Petersburg Electrotechnical University "LETI" (ETU "LETI"),
Faculty of Computer Science and Technology "FKTI",
Department of Computer Science and Engineering,
Computer Systems Engineering and Informatics (09.03.01) program.

OS labortory work 1 version 0_8 dated 2021_09_23

This software is under MIT License (X11 License).
You can see a detailed description in "LICENSE.md" file.

Copyright (c) 2021 Sobolev Matvey Sergeevich

*/

/*

Win32 API (WinAPI) is a set of functions in the library <windows.h>
API means "Application Programming Interface"

*/

```
#include <windows.h> // for WinAPI functions
#include <math.h> // for double making
#include <exception> // for exceptions
#include <iostream> // just for working
#include <string> // for the "string" type using
#include <vector> // for the "vector" type using
#include <algorithm> // for the "find" function using
```



```

using namespace std;

string currentPath = "c:\\"; // the current working path is disc "c:/" by default!

// ----- FUNCTION DECLARATION -----

string GetDiskName();
void MainMenu ();
void Info ();
void LocalGetLogicalDrives ();
void LocalGetDriveType ();
void LocalGetVolumeInformation ();
void LocalGetDiskFreeSpaceEx ();
void LocalGetDiskFreeSpace ();
void GetDiskInfo();
bool dirExists(const std::string& dirName_in);
string GetPathKernel (char localFlag);
string GetPathShell (char localFlagOne, char localFlagTwo, string localMessageOne, string
localMessageTwo);
void LocalCreateRemoveDirectory (char actionCreateRemove);
void LocalCreateFile();
void LocalCopyMoveFile(char actionCopyMove);
void LocalGetFileAttributes ();
void LocalSetFileAttributes ();
void LocalGetFileInformationByHandle ();
void GetFileTime ();
void SetFileTime ();

// ----- MAIN -----

int main (int argc, char* argv[]) // i've finally understood what it means (argc -- number of
arguments, argv -- strings of arguments (including -<word> and --<word>))

```

```

{
    // "GET CURRENT DIRECTORY", "SET CURRENT DIRECTORY"

    int flag = -1; // "-1" for incorrect input continue the program

    do
    {
        MainMenu();
        cin >> flag;
        cout << "\n";
        switch (flag)
        {
            case 0:
                cout << "Goodbye!";
                break;
            case 1:
                cout << "Your current working path is: \"" << currentPath << "\" (c:\\\
by default).\n";
                break;
            case 2:
                currentPath = GetPathKernel('c');
                break;
            case 3:
                Info();
                break;
            case 11:
                LocalGetLogicalDrives();
                break;
            case 21:
                LocalGetDriveType();
                break;
            case 22:

```

```
        LocalGetVolumeInformation();
        break;
case 23:
        LocalGetDiskFreeSpace();
        break;
case 24:
        GetDiskInfo();
        break;
case 231:
        LocalGetDiskFreeSpaceEx();
        break;
case 31:
        LocalCreateRemoveDirectory ('c');
        break;
case 32:
        LocalCreateRemoveDirectory ('r');
        break;
case 41:
        LocalCreateFile();
        break;
case 51:
        LocalCopyMoveFile('c');
        break;
case 52:
        LocalCopyMoveFile('m');
        break;
case 53:
        LocalCopyMoveFile('e');
        break;
case 61:
        LocalGetFileAttributes();
        break;
```

```

        case 62:
            LocalSetFileAttributes();
            break;
        case 63:
            LocalGetFileInformationByHandle();
            break;
        case 64:
            GetFileTime();
            break;
        case 65:
            SetFileTime();
            break;
        default:
            cout << "Incorrect input! Try again.";
            break;
    }
}
while (flag != 0);

return 0;
}

```

// ----- 0 -- GET DISK NAME -----

```

string GetDiskName ()
{
    string localDisk;

    cout << "Please, input disk name you want:\n";
    cin >> localDisk;
    localDisk = localDisk + ":\\";
}

```

```

        return localDisk;
    }

// ----- 0 -- MAIN MENU -----

void MainMenu ()
{
    cout << "\n";
    cout << "Please, choose the menu item:\n";
    cout << "0 -- COMMON:\n";
    cout << "0 -- Quit\n";
    cout << "1 -- Output current directory\n";
    cout << "2 -- Change current working directory\n";
    cout << "3 -- Info\n";
    cout << "1 -- DRIVES LIST:\n";
    cout << "11 -- Show all available disk drives\n";
    cout << "2 -- INFORMATION ABOUT DRIVES:\n";
    cout << "21 -- Show the drive type\n";
    cout << "22 -- Show the volume information\n";
    cout << "23 -- Show the disk free space\n";
    cout << "24 -- Show all disk information\n";
    cout << "3 -- DIRECTORIES:\n";
    cout << "31 -- Create new directory\n";
    cout << "32 -- Remove old directory\n";
    cout << "4 -- CREATE/OPEN FILES:\n";
    cout << "41 -- Create new file\n";
    cout << "5 -- COPY/MOVE FILES\n";
    cout << "51 -- Copy file\n";
    cout << "52 -- Move file\n";
    cout << "53 -- Move file (extended properties)\n";
    cout << "6 -- ATTRIBUTES\n";
    cout << "61 -- Get file attributes\n";
}

```

```

        cout << "62 -- Set file attributes\n";
        cout << "63 -- Get file attributes by handle\n";
        cout << "64 -- Get file time\n";
        cout << "65 -- Set file time\n";
        cout << "\n";
    }

// ----- 0 -- INFO -----

void Info ()
{
    cout << "Saint Petersburg Electrotechnical University \"LETI\" (ETU \"LETI\"),\n"
    << "Faculty of Computer Science and Technology \"FKTI\", \n"
    << "Department of Computer Science and Engineering, \n"
    << "Computer Systems Engineering and Informatics (09.03.01) program. \n\n"
    << "OS laboratory work 1 version 0_8 dated 2021_09_23\n\n"
    << "This software is under MIT License (X11 License). \n"
    << "You can see a detailed description in \"LICENSE.md\" file. \n\n"
    << "Copyright (c) 2021 Sobolev Matvey Sergeevich\n";
}

// ----- 1 -- GET LOGICAL DRIVES -----

void LocalGetLogicalDrives()
{
    int localDriveNumber = 1;
    int localDiskDetection;
    char localDriveLetter[4];
    DWORD dr = GetLogicalDrives();

    for(int i = 0; i < 26; i++)
    {

```

```

        localDiskDetection = ((dr >> i) & 0x00000001);
        if (localDiskDetection == 1)
        {
            localDriveLetter[0] = char(65 + i);
            localDriveLetter[1] = ':';
            localDriveLetter[2] = '\\';
            localDriveLetter[3] = 0;
            cout << localDriveNumber << ". Available disk drive: " << localDriveLetter
<< endl;

            localDriveNumber = localDriveNumber + 1; // next number
        }
    }
}

// ----- 2 -- GET DRIVE TYPE -----

void LocalGetDriveType()
{
    // i need to check, if i use the "uint" namespace

    int d;
    string n;

    n = GetDiskName();

    // <string variable>.c_str() means that you convert to <const char * type>, because "" isn't
    <const char *> type

    d = GetDriveType(n.c_str()); // i just want to know what's going one if i choose d or f!
    cout << n << " is";
    //cout << n + " is";

```

```

if (d == DRIVE_UNKNOWN)
{
    cout << " UNKNOWN" << endl;
}
if (d == DRIVE_NO_ROOT_DIR)
{
    cout << " DRIVE NO ROOT DIR" << endl;
}
if (d == DRIVE_REMOVABLE)
{
    cout << " REMOVABLE" << endl;
}
if (d == DRIVE_FIXED)
{
    cout << " FIXED" << endl;
}
if (d == DRIVE_REMOTE)
{
    cout << " REMOTE" << endl;
}
if (d == DRIVE_CDROM)
{
    cout << " CDROM" << endl;
}
if (d == DRIVE_RAMDISK)
{
    cout << " RAMDISK" << endl;
}
}

// ----- 2 -- GET VOLUME INFORMATION -----

```



```

void LocalGetVolumeInformation ()
{
    char VolumeNameBuffer[100];
    char FileSystemNameBuffer[100];
    string n;
    unsigned long VolumeSerialNumber;

    n = GetDiskName();

    BOOL GetVolumeInformationFlag = GetVolumeInformationA(
        n.c_str(),
        VolumeNameBuffer,
        100,
        &VolumeSerialNumber,
        NULL, //&MaximumComponentLength,
        NULL, //&FileSystemFlags,
        FileSystemNameBuffer,
        100
    );

    /*BOOL GetVolumeInformationFlag = GetVolumeInformationA(
        "d:\\",
        VolumeNameBuffer,
        100,
        &VolumeSerialNumber,
        NULL, //&MaximumComponentLength,
        NULL, //&FileSystemFlags,
        FileSystemNameBuffer,
        100
    );*/

    if (GetVolumeInformationFlag != 0)

```

```

{
    cout << " Volume Name is " << VolumeNameBuffer << endl;
    cout << " Volume Serial Number is " << VolumeSerialNumber << endl;
    cout << " File System is " << FileSystemNameBuffer << endl;
}
else
{
    cout << " Not Present (GetVolumeInformation)" << endl;
}
}

// ----- 2 -- GET DISK FREE SPACE EX -----

void LocalGetDiskFreeSpaceEx ()
{
    DWORD FreeBytesAvailable;
    DWORD TotalNumberOfBytes;
    DWORD TotalNumberOfFreeBytes;

    BOOL GetDiskFreeSpaceFlag = GetDiskFreeSpaceEx("c:\\", // directory name
(PULARGE_INTEGER)&FreeBytesAvailable, // bytes available to caller
(PULARGE_INTEGER)&TotalNumberOfBytes, // bytes on disk
(PULARGE_INTEGER)&TotalNumberOfFreeBytes // free bytes on disk
);
    if(GetDiskFreeSpaceFlag != 0)
    {
        //double d = double(unsigned long(TotalNumberOfFreeBytes))/1024/1024/1024;
        //cout << d;
        cout << " Total Number Of Free Bytes = " << (unsigned
long)TotalNumberOfFreeBytes << " ( " << (double)(((unsigned
long)(TotalNumberOfFreeBytes/1024))/1000) << " Mb )" << endl;
    }
}

```

```

        cout << " Total Number Of Bytes = " << (unsigned long)TotalNumberOfBytes << "(
" << (double)((((unsigned long)(TotalNumberOfBytes/1024))/1000) << " Mb )" << endl;
        cout << " Total Number Of Bytes = " << (unsigned long)TotalNumberOfBytes << "(
" << TotalNumberOfBytes << " Mb )" << endl;
        unsigned long tmp_1 = TotalNumberOfBytes;
        tmp_1 = tmp_1/1024;
        cout << tmp_1;
        double tmp_2 = tmp_1;
        tmp_2 = tmp_2/1000;
        cout << tmp_2;
        cout << TotalNumberOfBytes/1024000;
    }
    else
    {
        cout << " Not Present (GetDiskFreeSpace)" << endl;
    }
}

```

// ----- 2 -- GET DISK FREE SPACE -----

```

void LocalGetDiskFreeSpace ()
{
    /*long unsigned int * secPerClus;
    long unsigned int * bytePerSec;
    long unsigned int * freeClus;
    long unsigned int * totalClus;*/
    /*LPDWORD secPerClus;
    LPDWORD bytePerSec;
    LPDWORD freeClus;
    LPDWORD totalClus;*/

    string localDiskName = GetDiskName();

```

```

//const char diskNameCC[4] = {'e', ':', '\\'}; // you can do this
//string diskNameS = "e:\\"; // and you can do this

DWORD secPerClus;
DWORD bytePerSec;
DWORD freeClus;
DWORD totalClus;

//int gdfs = GetDiskFreeSpace(diskNameCC, &secPerClus, &bytePerSec, &freeClus,
&totalClus); // const char* explicitly
//int gdfs = GetDiskFreeSpace(diskNameS.c_str(), &secPerClus, &bytePerSec, &freeClus,
&totalClus); // const char* from string (with c_str() method)
int gdfs = GetDiskFreeSpace(localDiskName.c_str(), &secPerClus, &bytePerSec,
&freeClus, &totalClus);

if (gdfs != 0)
{
    cout << "Disk free space information about disk " << localDiskName << "!\n";
    cout << "----- NET NUMBERS ----- \n";
    cout << "Total Number Of Sectors per Cluster: " << (unsigned long long)secPerClus
<< "\nTotal Number Of Bytes per Sector: " << (unsigned long long)bytePerSec << "\nTotal Number
Of Free Clusters: " << (unsigned long long)freeClus << "\nTotal Number Of Clusters: " <<
(unsigned long long)totalClus << "\nReturned value: " << (unsigned long long)gdfs << "\n";
    cout << "----- CALCULATED NUMBERS ----- \n";
    cout << "Total Number Of Bytes per Cluster: " << (unsigned long
long)secPerClus*(unsigned long long)bytePerSec << "\nTotal Number Of Free Sectors: " <<
(unsigned long long)freeClus*(unsigned long long)secPerClus;
    cout << "\nTotal Number Of Sectors: " << (unsigned long long)totalClus*(unsigned
long long)secPerClus << "\nTotal Number Of Free Bytes: " << (unsigned long
long)freeClus*(unsigned long long)secPerClus*(unsigned long long)bytePerSec << "\nTotal
Number of Bytes: " << (unsigned long long)totalClus*(unsigned long long)secPerClus*(unsigned
long long)bytePerSec << "\n";
}

```

```

        cout << "Total Number Of Used Clusters: " << (unsigned long long)totalClus -
(unsigned long long)freeClus;

        cout << "\nTotal Number Of Used Sectors: " << (unsigned long
long)totalClus*(unsigned long long)secPerClus - (unsigned long long)freeClus*(unsigned long
long)secPerClus << "\nTotal Number Of Used Bytes: " << (unsigned long long)totalClus*(unsigned
long long)secPerClus*(unsigned long long)bytePerSec - (unsigned long long)freeClus*(unsigned
long long)secPerClus*(unsigned long long)bytePerSec << "\n";
    }
    else
    {
        cout << "Returned value: " << (unsigned long long)gdfs << "\nThere is no such disk
as " << localDiskName << "!\n";
    }
}

```

// ----- 2 -- GET DISK FREE SPACE -----

```
void GetDiskInfo()
```

```

{
    string drive = GetPathShell('f', 's', "Path to the DISK.\n", "Do you want to input absolute
(full) path of the file (directory) or relative (short)? [f/s]\n");
    DWORD drive_type = GetDriveTypeA(drive.c_str());
    switch (drive_type) {
        case DRIVE_UNKNOWN: cout<< "The drive type cannot be determined.\n";
            break;
        case DRIVE_NO_ROOT_DIR: cout<< "The root path is invalid; for example, there is no
volume mounted at the specified path. \n";
            break;
        case DRIVE_REMOVABLE: cout<< "The drive \""<< drive<<"\" has removable media; for
example, a floppy drive, thumb drive, or flash card reader. \n";
            break;
    }
}

```

```

    case DRIVE_FIXED: cout<< "The drive \""<< drive<<"\"has fixed media; for example, a hard
disk drive or flash drive. \n";
        break;
    case DRIVE_REMOTE: cout<< "The drive \""<< drive<<"\" is a remote (network) drive. \n";
        break;
    case DRIVE_CDROM: cout<< "The drive \""<< drive<<"\" is a CD-ROM drive\n";
        break;
    case DRIVE_RAMDISK: cout<< "The drive \""<< drive<<"\" is a RAM disk.\n";
        break;
    default: cout<< "You shouldn't see this message. Smth goes wrong";
}

```

```

char nameBuffer[100];
char sysNameBuff[100];
DWORD serialNumber,maxComponentLength,fileSystemFlags;

```

```

if(GetVolumeInformationA(drive.c_str(),nameBuffer,sizeof(nameBuffer),&serialNumber,&maxCo
mponentLength,&fileSystemFlags,sysNameBuff,sizeof(sysNameBuff))){
    cout << "\nDrive name:" << nameBuffer << endl <<
    "Type of File system:" << sysNameBuff << endl <<
    "Serial number:" << serialNumber << endl <<
    "System flags:" << endl;
    string specVol = "The specified volume";
    string specVolSup = specVol + " supports";
    if (fileSystemFlags & FILE_CASE_PRESERVED_NAMES)
        cout << specVol + " preserved case of file names when it places a name on disk.\n";
    if (fileSystemFlags & FILE_CASE_SENSITIVE_SEARCH)
        cout << specVolSup + " case-sensitive file names.\n";
    if (fileSystemFlags & FILE_FILE_COMPRESSION)
        cout << specVolSup + " file-based compression.\n";
    if (fileSystemFlags & FILE_NAMED_STREAMS)

```

```

    cout << specVolSup + " named streams.\n";
if (fileSystemFlags & FILE_PERSISTENT_ACLS)
    cout << specVol + " preserves and enforces access control lists (ACL). For example, the
NTFS file system preserves and enforces ACLs, and the FAT file system does not.\n";
if (fileSystemFlags & FILE_READ_ONLY_VOLUME)
    cout << specVol + " is read-only.\n";
if (fileSystemFlags & FILE_SEQUENTIAL_WRITE_ONCE)
    cout << specVolSup + " a single sequential write.\n";
if (fileSystemFlags & FILE_SUPPORTS_ENCRYPTION)
    cout << specVolSup + " the Encrypted File System (EFS).\n";
if (fileSystemFlags & FILE_SUPPORTS_EXTENDED_ATTRIBUTES)
    cout << specVolSup + " extended attributes.\n";
if (fileSystemFlags & FILE_SUPPORTS_HARD_LINKS)
    cout << specVolSup + " hard links. \n";
if (fileSystemFlags & FILE_SUPPORTS_OBJECT_IDS)
    cout << specVolSup + " object identifiers.\n";
if (fileSystemFlags & FILE_SUPPORTS_OPEN_BY_FILE_ID)
    cout << specVolSup + " open by FileID.\n";
if (fileSystemFlags & FILE_SUPPORTS_REPARSE_POINTS)
    cout << specVolSup + " reparse points.\n";
if (fileSystemFlags & FILE_SUPPORTS_SPARSE_FILES)
    cout << specVolSup + " sparse files.\n";
if (fileSystemFlags & FILE_SUPPORTS_TRANSACTIONS)
    cout << specVolSup + " transactions.\n";
if (fileSystemFlags & FILE_SUPPORTS_USN_JOURNAL)
    cout << specVolSup + " update sequence number (USN) journals.\n";
if (fileSystemFlags & FILE_UNICODE_ON_DISK)
    cout << specVolSup + " Unicode in file names as they appear on disk.\n";
if (fileSystemFlags & FILE_VOLUME_IS_COMPRESSED)
    cout << specVol + " is a compressed volume, for example, a DoubleSpace volume.\n";
if (fileSystemFlags & FILE_VOLUME_QUOTAS)
    cout << specVolSup + " disk quotas.\n";

```

```

}

long long FreeBytesAvailableToCaller;
long long TotalNumberOfBytes;
GetDiskFreeSpaceExA(drive.c_str(), (PULARGE_INTEGER)&FreeBytesAvailableToCaller,
(PULARGE_INTEGER)&TotalNumberOfBytes, nullptr);
cout << "\nTotal drive space: " << TotalNumberOfBytes << " bytes" << endl;
cout << "Available drive space: " << FreeBytesAvailableToCaller << " bytes" << endl;
}

// ----- 3 -- CREATE DIRECTORY -----

bool dirExists(const std::string& dirName_in)
{
    DWORD ftyp = GetFileAttributesA(dirName_in.c_str());
    if (ftyp == INVALID_FILE_ATTRIBUTES)
    {
        return false; // something is wrong with your path!
    }
    else if (ftyp & FILE_ATTRIBUTE_DIRECTORY)
    {
        return true; // this is a directory!
    }
    else
    {
        return false; // this is not a directory!
    }
}

// ----- 3 -- GET PATH KERNEL -----

// 'c' -- current (working) directory path

```



```

// 'a' -- absolute directory path
// 'r' -- relative directory path
// 'f' -- absolute (full) file path
// 's' -- relative (short) file path
string GetPathKernel (char localFlag)
{
    char localCommit = 'n';
    string localPath;
    string localOldPath = currentPath;
    while (localCommit != 'y')
    {
        if (localFlag == 'a') // 'a' means "absolute path"
        {
            cout << "Please, input absolute directory path:\n";
            cin >> localPath;
        }
        else if (localFlag == 'r') // 'r' means "relative path"
        {
            cout << "Your current path is \"" << currentPath << "\". Please, input relative
directory path:\n";
            cin >> localPath;
            localPath = currentPath + localPath;
        }
        else if (localFlag == 'f') // 'f' means "full file path"
        {
            cout << "Please, input absolute file path:\n";
            cin >> localPath;
        }
        else if (localFlag == 's') // 's' means "short file path"
        {
            cout << "Your current path is \"" << currentPath << "\". Please, input relative
file path:\n";

```

```

        cin >> localPath;
        localPath = currentPath + localPath;
    }
    else if (localFlag == 'c') // 'c' means "current"
    {
        cout << "Please, input current working directory path:\n";
        cin >> localPath;
        if (localPath.length() > 0)
        {
            if (localPath.c_str()[localPath.length() - 1] != '\\')
            {
                localPath = localPath + "\\";
            }
        }
    }
    else // current working path by default
    {
        cout << "Please, input current working directory path:\n";
        cin >> localPath;
    }
    if (localFlag != 'a' && localFlag != 'r' && localFlag != 'f' && localFlag != 's' &&
dirExists(localPath))
    {
        if (localFlag == 'c') // 'c' means "current"
        {
            cout << "This is your new current working path: " << localPath <<
"\n";
        }
        else // current working path by default
        {
            cout << "This is your new current working path: " << localPath <<
"\n";
        }
    }

```

```

    }
    cout << "Commit changes? [y/n]\n";
    cin >> localCommit;
}
else if (localFlag == 'a' || localFlag == 'r') // for directories
{
    if (localFlag == 'a') // 'a' means "absolute path"
    {
        cout << "This is your new absolute directory path: " << localPath <<
"\n";
    }
    else if (localFlag == 'r') // 'r' means "relative path"
    {
        cout << "This is your new relative directory path: " << localPath <<
"\n";
    }
    cout << "Commit changes? [y/n]\n";
    cin >> localCommit;
}
else if (localFlag == 'f' || localFlag == 's') // for files
{
    if (localFlag == 'f') // 'f' means "full file path"
    {
        cout << "This is your new absolute file path: " << localPath << "\n";
    }
    else if (localFlag == 's') // 's' means "shorth file path"
    {
        cout << "This is your new relative file path: " << localPath << "\n";
    }
    cout << "Commit changes? [y/n]\n";
    cin >> localCommit;
}
}

```

```

else
{
    cout << "Your new path \" << localPath << "\" isn't valid (exist)! Try again
(if no, old path will be returned)? [y/n]: " << "\n";
    cin >> localCommit;
    if (localCommit == 'y')
    {
        localCommit = 'n';
    }
    else if (localCommit == 'n')
    {
        localCommit = 'y';
        localPath = localOldPath;
    }
}
}
if (localFlag != 'a' && localFlag != 'r' && localFlag != 'f' && localFlag != 's')
{
    currentPath = localPath;
}
// because for setting current directory it doesn't matter, but for creating/removing it matters,
// so to unificate the program, i make local path a current only if it's not for creating or
deleting,
// but i return the local path (for every flags it will be right, only consist different things)
return localPath;
}

// ----- 3 -- GET PATH SHELL -----

string GetPathShell (char localFlagOne, char localFlagTwo, string localMessageOne, string
localMessageTwo)
{

```

```

    //"Path to the FILE, which ATTRIBUTES you WANT TO GET.\n"
    char localFunctionFlag = '!'; // just random symbol
    string localFunctionPath;
    while (localFunctionFlag != localFlagOne && localFunctionFlag != localFlagTwo)
    {
        cout << localMessageOne << localMessageTwo;
        // checking for the flag
        cin >> localFunctionFlag;
        if (localFunctionFlag != localFlagOne && localFunctionFlag != localFlagTwo)
        {
            cout << "Try again!\n";
        }
    }
    if (localFunctionFlag == localFlagOne) // full file path situation
    {
        localFunctionPath = GetPathKernel(localFunctionFlag); // set new absolute path
    }
    else if (localFunctionFlag == localFlagTwo) // short file path situation
    {
        char localChange = 'n';
        cout << "Do you want to change current working path? [y/n]\n";
        cin >> localChange;
        if (localChange == 'y')
        {
            currentPath = GetPathKernel('c'); // changing current directory
        }
        localFunctionPath = GetPathKernel(localFunctionFlag); // set new relative path
    }
    return localFunctionPath;
}

// ----- 3 -- LOCAL CREATE DIRECTORY -----

```

```

void LocalCreateRemoveDirectory (char actionCreateRemove) // if 'c' -- creating directory, if 'r' --
removing directory, creating by default
{
    char localPathFlag = 'y'; // just another letter, not 'a' or 'r', so you need input it anyway
    string localDirectory; // directory path you input

    localDirectory = GetPathShell('a', 'r', "", "Do you want to input absolute path of the directory
or relative? [a/r]\n");

    if (actionCreateRemove == 'r') // 'r'
    {
        if (RemoveDirectory(localDirectory.c_str()))
        {
            cout << "The directory \"" << localDirectory << "\" has been successfully
removed!\n";
        }
        else
        {
            cout << "Something wrong! The directory \"" << localDirectory << "\" hasn't
been removed!\n";
            cout << "Last error code is \"" << GetLastError() << "\"\n";
        }
    }
    else // 'c' and default
    {
        if (CreateDirectory(localDirectory.c_str(), NULL))
        {
            cout << "The directory \"" << localDirectory << "\" has been successfully
created!\n";
        }
        else
    }

```

```

        {
            cout << "Something wrong! The directory \"" << localDirectory << "\" hasn't
been created!\n";

            cout << "Last error code is \"" << GetLastError() << "\"\n";
        }
    }
}

```

// ----- 4 -- LOCAL CREATE FILE -----

void LocalCreateFile () // A WISE FACT: THERE IS NO "OPEN FILE" FINCTION, THERE IS
"CREATE FILE" FUNCTION WITH SPECIAL FLAG TO OPEN FILE!

```

{
    int localChoose = 0; // to start a loop
    unsigned long localCreationDisposition; // DWORD = unsigned long, localChoose =
number that has been written in specification
    char localPathFlag = 'y'; // just another letter, not 'f' or 's', so you need input it anyway
    string localFilePath; // file path you input

    localFilePath = GetPathShell('f', 's', "Path to the FILE, which you WANT TO CREATE
(OPEN).\n", "Do you want to input absolute (full) path of the file or relative (short)? [f/s]\n");

    while (localChoose < 1 || localChoose > 5)
    {
        // because "CREATE_NEW" by default (1 is number for "CREATE_NEW")
        cout << "Please, choose the creation disposition (CREATE_NEW by default):\n" <<
"1 -- CREATE_NEW\n" << "2 -- CREATE_ALWAYS\n" << "3 -- OPEN_EXISTING\n"
        << "4 -- OPEN_ALWAYS\n" << "5 -- TRUNCATE_EXISTING\n"; // DWORD =
unsigned long, localChoose = number that has been written in specification
        cin >> localChoose;
        if (localChoose < 1 || localChoose > 5)
        {

```

```

        cout << "Try again!\n";
    }
}

switch(localChoose)
{
    case 1:
        localCreationDisposition = (unsigned long)localChoose; // DWORD =
unsigned long, localChoose = number that has been written in specification
        break;
    case 2:
        localCreationDisposition = (unsigned long)localChoose;
        break;
    case 3:
        localCreationDisposition = (unsigned long)localChoose;
        break;
    case 4:
        localCreationDisposition = (unsigned long)localChoose;
        break;
    case 5:
        localCreationDisposition = (unsigned long)localChoose;
        break;
    default:
        localCreationDisposition = (unsigned long)localChoose;
        break;
}

// Open a handle to the file
HANDLE localFile = CreateFile(
    localFilePath.c_str(), // Filename (<path to the file>)
    GENERIC_WRITE, // Desired access (0/GENERIC_READ/GENERIC_WRITE)

```



```

        FILE_SHARE_READ,                //                Share                mode
(FILE_SHARE_DELETE/FILE_SHARE_READ/FILE_SHARE_WRITE)
        NULL, // Security attributes (NULL/<structure SECURITY_ATTRIBUTES adress>)
        (DWORD)localCreationDisposition, // Creates a new file, only if it doesn't already
exist
        (CREATE_ALWAYS/CREATE_NEW/OPEN_ALWAYS/OPEN_EXISTING/TRUNCATE_EXISTI
NG)
        FILE_ATTRIBUTE_NORMAL,          //                Flags                and                attributes
(FILE_ATTRIBUTE_NORMAL/FILE_ATTRIBUTE_ARCHIVE/FILE_ATTRIBUTE_ENCRYPT
ED/FILE_ATTRIBUTE_SYSTEM/
        //
FILE_ATTRIBUTE_HIDDEN/FILE_ATTRIBUTE_NOT_CONTENT_INDEXED/FILE_ATTRIB
UTE_OFFLINE/FILE_ATTRIBUTE_READONLY/FILE_ATTRIBUTE_SYSTEM/FILE_ATTRIB
UTE_TEMPORARY/
        //
FILE_FLAG_BACKUP_SEMANTICS/FILE_FLAG_DELETE_ON_CLOSE/FILE_FLAG_NO_B
UFFERING/FILE_FLAG_OPEN_NO_RECALL/FILE_FLAG_OPEN_REPARSE_POINT/FILE_F
LAG_OVERLAPPED/
        //
FILE_FLAG_POSIX_SEMANTICS/FILE_FLAG_RANDOM_ACCESS/FILE_FLAG_SEQUENT
IAL_SCAN/FILE_FLAG_WRITE_THROUGH/SECURITY_ANONYMOUS/SECURITY_CONT
EXT_TRACKING/
        //
SECURITY_DELEGATION/SECURITY_EFFECTIVE_ONLY/SECURITY_IDENTIFICATION/
SECURITY_IMPERSONATION)
        NULL); // Template file handle (NULL/<template file descriptor>)
        if (localFile != INVALID_HANDLE_VALUE)
        {
                cout << "The file \"" << localFilePath << "\" has been successfully created (opened,
rewritten)!\n";
                CloseHandle(localFile);
        }

```

```

else
{
    cout << "Something wrong! The file \"" << localFilePath << "\" hasn't been created
(opened, rewritten)!\n";
    cout << "Last error code is \"" << GetLastError() << "\"\n";
}
}

// ----- 5 -- LOCAL COPY MOVE FILE -----

void LocalCopyMoveFile (char actionCopyMove) // 'c' for copy, 'm' for moving, 'e' for extended
moving (MoveFileEx) ATTENTION: 'c' -- copy -- is default if there is other letter!
{
    // specification of "CopyFile"
    /*BOOL CopyFile(
        LPCTSTR lpExistingFileName, // current file you want to copy
        LPCTSTR lpNewFileName, // new file, where you want to copy the old one
        BOOL bFailIfExists // TRUE means STOP IF NEW FILE EXIST, FALSE means
OVERWRITE FILE ANYWAY
    );*/

    int localChoose = 0; // to start a loop
    bool localFailIfExists; // for existing file reaction
    char localPathFlag = 'y'; // just another letter, not 'f' or 's', so you need input it anyway
    char localSameNameFlag = 'a'; // random letter
    string localOldFilePath; // old (copied) file path you input
    string localNewFilePath; // new (pasted) file path you input

    // OLD FILE PATH INPUT (INCLUDING SITUATIONS "COPY" AND "MOVE")

    if (actionCopyMove == 'm' || actionCopyMove == 'e') // 'm' is for "moving", 'e' is for
"extended moving"

```

```

{
    localOldFilePath = GetPathShell('f', 's', "Path to the FILE, which you WANT TO
MOVE (CUT).\n", "Do you want to input absolute (full) path of the file or relative (short)? [f/s]\n");
}
else // 'c' is for "copy", "copy" is default
{
    localOldFilePath = GetPathShell('f', 's', "Path to the FILE, which you WANT TO
COPY.\n", "Do you want to input absolute (full) path of the file or relative (short)? [f/s]\n");
}

// SAME NAME SITUATION

while (localSameNameFlag != 'y' && localSameNameFlag != 'n')
{
    cout << "Do you want the SAME NAME AS COPIED/CUT file (including
extension)? [y/n]\n";
    // checking for the flag
    cin >> localSameNameFlag;
    if (localSameNameFlag != 'y' && localSameNameFlag != 'n') // 'n' -- no -- by default
    {
        cout << "Try again!\n";
    }
}

// NEW FILE PATH INPUT

//localPathFlag = 'y'; // random letter again to prevent ignoring if construction

if (localSameNameFlag == 'y') // same name of the file -- choosing full directory path
instead of the filename
{
    string localFilename = string(localOldFilePath);

```

```

// Remove directory if present.
// Do this before extension removal incase directory has a period character.
const size_t last_slash_idx = localFilename.find_last_of("\\/");
if (std::string::npos != last_slash_idx)
{
    localFilename.erase(0, last_slash_idx + 1);
}
// Remove extension if present.
/*const size_t period_idx = localFilename.rfind('.');
if (std::string::npos != period_idx)
{
    localFilename.erase(period_idx);
}*/

if (actionCopyMove == 'm' || actionCopyMove == 'e') // 'm' is for "moving", 'e' is for
"extended moving"
{
    localNewFilePath = GetPathShell('a', 'r', "Path to the NEW DIRECTORY,
where you WANT to MOVE (PASTE) the CUT FILE.\n", "Do you want to input absolute path of
the directory or relative? [a/r]\n") + localFilename;
}
else // 'c' is for "copy", "copy" is DEFAULT
{
    localNewFilePath = GetPathShell('a', 'r', "Path to the NEW DIRECTORY,
where you WANT to PASTE the COPIED FILE.\n", "Do you want to input absolute path of the
directory or relative? [a/r]\n") + localFilename;
}
}
else // 'n', different (as user choosed) name of the file -- choosing full file path (not the same
name is DEFAULT)
{

```

```

        if (actionCopyMove == 'm' || actionCopyMove == 'e') // 'm' is for "moving", 'e' is for
"extended moving"
        {
            localNewFilePath = GetPathShell('f', 's', "Path to the NEW FILE, where you
WANT to MOVE (PASTE) the CUT FILE.\n", "Do you want to input absolute (full) path of the file
or relative (short)? [f/s]\n");
        }
        else // 'c' is for "copy", "copy" is DEFAULT
        {
            localNewFilePath = GetPathShell('f', 's', "Path to the NEW FILE, where you
WANT to PASTE the COPIED FILE.\n", "Do you want to input absolute (full) path of the file or
relative (short)? [f/s]\n");
        }
    }

    // COPY and MOVE

    if (actionCopyMove == 'c' || (actionCopyMove != 'm' && actionCopyMove != 'e')) // copy
    {
        // IF NEW FILE EXISTS

        while (localChoose < 1 || localChoose > 2)
        {
            // because "CREATE_NEW" by default (1 is number for "CREATE_NEW")
            cout << "Please, choose the possible outcome for you IF NEW FILE
EXISTS:\n" << "1 -- DON'T PASTE AND DON'T REWRITE ANY FILE (TRUE)\n"
            << "2 -- REWRITE THE FILE (FALSE)\n";
            cin >> localChoose;
            if (localChoose < 1 || localChoose > 2)
            {
                cout << "Try again!\n";
            }
        }
    }

```

```

    }

    switch(localChoose)
    {
        case 1:
            localFailIfExists = true;
            break;

        case 2:
            localFailIfExists = false;
            break;

        default:
            localFailIfExists = true; // true -- file rewriting protection -- by default
            break;
    }

    // don't forget about c_str()!
    if (CopyFile(localOldFilePath.c_str(), localNewFilePath.c_str(), localFailIfExists)) //
copy file and watching the result immediatly
    {
        cout << "The file \"" << localOldFilePath << "\" has been successfully copied
to file \"" << localNewFilePath << "\"!\n";
    }
    else
    {
        cout << "Something wrong! The file \"" << localOldFilePath << "\" hasn't
been copied to file \"" << localNewFilePath << "\"!\n";
        cout << "Last error code is \"" << GetLastError() << "\"\n";
    }
}

else if (actionCopyMove == 'm') // move
{
    // don't forget about c_str()!

```

```

        if (MoveFile(localOldFilePath.c_str(), localNewFilePath.c_str())) // move file and
watching the result immediatly
    {
        cout << "The file \"" << localOldFilePath << "\" has been successfully
moved to file \"" << localNewFilePath << "\"!\n";
    }
    else
    {
        cout << "Something wrong! The file \"" << localOldFilePath << "\" hasn't
been moved to file \"" << localNewFilePath << "\"!\n";
        cout << "Last error code is \"" << GetLastError() << "\"\n";
    }
}
else if (actionCopyMove == 'e') // extended move
{
    string localChooseTwo = "2"; // because default
    // next parameters presented in decimal code, but can be translated at once (for
example "11111" is all of the flags (but all is uncorrect)!)
    // MOVEFILE_COPY_ALLOWED = 2 (copy, than delete old, normal work), cannot
be used with MOVEFILE_DELAY_UNTIL_REBOOT
    // MOVEFILE_CREATE_HARDLINK = 16 ("Reserved for future use", i don't know
what does it means)
    // MOVEFILE_DELAY_UNTIL_REBOOT = 4 (waiting util reboot), cannot be used
with MOVEFILE_COPY_ALLOWED
    // MOVEFILE_FAIL_IF_NOT_TRACKABLE = 32 ("function fails, if the file is the
lik source")
    // MOVEFILE_REPLACE_EXISTING = 1 (replacing file)
    // MOVEFILE_WRITE_THROUGH = 8 (doesn't return anything until it's
ACTUALLY MOVE SOMETHING!)
    unsigned long inFunctionNumber = 0;
    while (inFunctionNumber == 0)
    {

```

```

// because "CREATE_NEW" by default (1 is number for "CREATE_NEW")
cout << "Please, choose the possible flags for moving file (you CAN
CHOOSE MANY -- JUST SPLIT NUMBERS BY SPACE):\n"

<< "1 -- MOVEFILE_REPLACE_EXISTING (replacing existing, it it
exists)\n"

<< "2 -- MOVEFILE_COPY_ALLOWED (classic move) -- DEFAULT\n"
<< "4 -- MOVEFILE_DELAY_UNTIL_REBOOT (moving after rebooting
the system)\n"

<< "8 -- MOVEFILE_WRITE_THROUGH (moving file, than returning
value)\n"

<< "16 -- MOVEFILE_CREATE_HARDLINK\n"
<< "32 -- MOVEFILE_FAIL_IF_NOT_TRACKABLE\n";
fflush(stdin);
std::getline(std::cin, localChooseTwo);

// spit the string
std::string s = string(localChooseTwo);
std::string delimiter = " ";

int i = 0;
size_t pos = 0;
std::string token;
std::vector<string> v;
std::vector<int> vect{1, 2, 4, 8, 16, 32};
while ((pos = s.find(delimiter)) != std::string::npos)
{
    int tmpNumber = 0;
    token = s.substr(0, pos);
    v.push_back(token);
    tmpNumber = std::stoi(token);
    if (std::find(vect.begin(), vect.end(), tmpNumber) != vect.end())
    {

```



```

        inFunctionNumber = inFunctionNumber + tmpNumber;
        vect.erase(std::remove(vect.begin(),    vect.end(),    tmpNumber),
vect.end());

    }
    //std::cout << token << std::endl;
    s.erase(0, pos + delimiter.length());
}

int newTMPNumber = std::stoi(s);
if (std::find(vect.begin(), vect.end(), newTMPNumber) != vect.end())
{
    inFunctionNumber = inFunctionNumber + newTMPNumber;
    vect.erase(std::remove(vect.begin(),    vect.end(),    newTMPNumber),
vect.end());
}

//cout << inFunctionNumber;

//std::cout << s << std::endl;
// end split of the string

if (inFunctionNumber == 0)
{
    cout << "Try again!\n";
}
}
// don't forget about c_str()!
if      (MoveFileEx(localOldFilePath.c_str(),    localNewFilePath.c_str(),
(DWORD)inFunctionNumber)) // extended move file and watching the result immediatly
{
    cout << "The file \"" << localOldFilePath << "\" has been successfully
moved to file \"" << localNewFilePath << "\"!\n";
}

```

```

    }
    else
    {
        cout << "Something wrong! The file \"" << localOldFilePath << "\" hasn't
been moved to file \"" << localNewFilePath << "\"!\n";
        cout << "Last error code is \"" << GetLastError() << "\"\n"; // here i need to
insert last error text string
    }
}
}

```

// ----- 6 -- LOCAL GET FILE ATTRIBUTES -----

```
void LocalGetFileAttributes ()
```

```

{
    // specification of "GetFileAttributesA"
    /*DWORD GetFileAttributesA(
        LPCSTR lpFileName // file name, which i want to get the file attributes
    );*/

    DWORD localFileAttributes = 0;
    string localFilePath; // file path you input

    // FILE PATH INPUT

    localFilePath = GetPathShell('f', 's', "Path to the FILE or DIRECTORY, which
ATTRIBUTES you WANT TO GET.\n", "Do you want to input absolute (full) path of the file
(directory) or relative (short)? [f/s]\n");

    localFileAttributes = GetFileAttributes(localFilePath.c_str());

    cout << "File \"" << localFilePath << "\" attributes:\n";
}

```

```

if (localFileAttributes & FILE_ATTRIBUTE_ARCHIVE)
{
    cout << "Archive (FILE_ATTRIBUTE_ARCHIVE)\n";
}
if (localFileAttributes & FILE_ATTRIBUTE_COMPRESSED)
{
    cout << "Compressed (FILE_ATTRIBUTE_COMPRESSED)\n";
}
if (localFileAttributes & FILE_ATTRIBUTE_DEVICE)
{
    cout << "Device (FILE_ATTRIBUTE_DEVICE)\n";
}
if (localFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
{
    cout << "Directory (FILE_ATTRIBUTE_DIRECTORY)\n";
}
if (localFileAttributes & FILE_ATTRIBUTE_ENCRYPTED)
{
    cout << "Encrypted (FILE_ATTRIBUTE_ENCRYPTED)\n";
}
if (localFileAttributes & FILE_ATTRIBUTE_HIDDEN)
{
    cout << "Hidden (FILE_ATTRIBUTE_HIDDEN)\n";
}
/*if (localFileAttributes & FILE_ATTRIBUTE_INTEGRITY_STREAM)
{
    cout << "Data stream configured with integrity
(FILE_ATTRIBUTE_INTEGRITY_STREAM)\n";
}*/
if (localFileAttributes & FILE_ATTRIBUTE_NORMAL)
{
    cout << "Normal (FILE_ATTRIBUTE_NORMAL)\n";
}

```

```

    }
    if (localFileAttributes & FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)
    {
        cout << "Not indexed (FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)\n";
    }
    /*if (localFileAttributes & FILE_ATTRIBUTE_NO_SCRUB_DATA)
    {
        cout << "Data stream not to be read by the data integrity scanner
(FILE_ATTRIBUTE_NO_SCRUB_DATA)\n";
    }*/
    if (localFileAttributes & FILE_ATTRIBUTE_OFFLINE)
    {
        cout << "Don't available immediatly (FILE_ATTRIBUTE_OFFLINE)\n";
    }
    if (localFileAttributes & FILE_ATTRIBUTE_READONLY)
    {
        cout << "Read-only (FILE_ATTRIBUTE_READONLY)\n";
    }
    /*if (localFileAttributes & FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)
    {
        cout << "Data is not fully presented locally
(FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)\n";
    }
    if (localFileAttributes & FILE_ATTRIBUTE_RECALL_ON_OPEN)
    {
        cout << "Data hasn't physical representation on system
(FILE_ATTRIBUTE_RECALL_ON_OPEN)\n";
    }*/
    if (localFileAttributes & FILE_ATTRIBUTE_REPARSE_POINT)
    {
        cout << "Reparse point/representation link
(FILE_ATTRIBUTE_REPARSE_POINT)\n";
    }

```

```

    }
    if (localFileAttributes & FILE_ATTRIBUTE_SPARSE_FILE)
    {
        cout << "Sparse file (FILE_ATTRIBUTE_SPARSE_FILE)\n";
    }
    if (localFileAttributes & FILE_ATTRIBUTE_SYSTEM)
    {
        cout << "System used (FILE_ATTRIBUTE_SYSTEM)\n";
    }
    if (localFileAttributes & FILE_ATTRIBUTE_TEMPORARY)
    {
        cout << "Temporary storage (FILE_ATTRIBUTE_TEMPORARY)\n";
    }
    if (localFileAttributes & FILE_ATTRIBUTE_VIRTUAL)
    {
        cout << "Reserved for system (FILE_ATTRIBUTE_VIRTUAL)\n";
    }
}

```

// ----- 6 -- LOCAL SET FILE ATTRIBUTES -----

```

void LocalSetFileAttributes ()
{
    // specification of "SetFileAttributesA"
    /*BOOL SetFileAttributesA(
        LPCSTR lpFileName, // filename
        DWORD dwFileAttributes //attributes
    );*/

    DWORD localFileAttributes = 0;
    string localFilePath; // file path you input
}

```

```
// FILE PATH INPUT
```

```
localFilePath = GetPathShell('f', 's', "Path to the FILE or DIRECTORY, which  
ATTRIBUTES you WANT TO SET.\n", "Do you want to input absolute (full) path of the file  
(directory) or relative (short)? [f/s]\n");
```

```
localFileAttributes = GetFileAttributes(localFilePath.c_str());
```

```
string localChooseTwo = "128";
```

```
unsigned long inFunctionNumber = 0;
```

```
while (inFunctionNumber == 0)
```

```
{
```

```
    // because "CREATE_NEW" by default (1 is number for "CREATE_NEW")
```

```
    cout << "Please, choose the possible attributes for the file/directory (you CAN  
CHOOSE MANY -- JUST SPLIT NUMBERS BY SPACE):\n"
```

```
    << "1 -- FILE_ATTRIBUTE_READONLY (read-only)\n"
```

```
    << "2 -- FILE_ATTRIBUTE_HIDDEN (hidden)\n"
```

```
    << "4 -- FILE_ATTRIBUTE_SYSTEM (system used)\n"
```

```
    << "32 -- FILE_ATTRIBUTE_ARCHIVE (archive)\n"
```

```
    << "128 -- FILE_ATTRIBUTE_NORMAL -- DEFAULT\n"
```

```
    << "256 -- FILE_ATTRIBUTE_TEMPORARY (temporary storage)\n"
```

```
    << "4096 -- FILE_ATTRIBUTE_OFFLINE (don't available immediatly)\n"
```

```
    << "8192 -- FILE_ATTRIBUTE_NOT_CONTENT_INDEXED (not indexed)\n";
```

```
    fflush(stdin);
```

```
    std::getline(std::cin, localChooseTwo);
```

```
    // spit the string
```

```
    std::string s = string(localChooseTwo);
```

```
    std::string delimiter = " ";
```

```
    int i = 0;
```

```
    size_t pos = 0;
```

```

std::string token;
std::vector<string> v;
std::vector<int> vect{1, 2, 4, 32, 128, 256, 4096, 8192};
while ((pos = s.find(delimiter)) != std::string::npos)
{
    int tmpNumber = 0;
    token = s.substr(0, pos);
    v.push_back(token);
    tmpNumber = std::stoi(token);
    if (std::find(vect.begin(), vect.end(), tmpNumber) != vect.end())
    {
        inFunctionNumber = inFunctionNumber + tmpNumber;
        vect.erase(std::remove(vect.begin(), vect.end(), tmpNumber), vect.end());
    }
    //std::cout << token << std::endl;
    s.erase(0, pos + delimiter.length());
}

int newTMPNumber = std::stoi(s);
if (std::find(vect.begin(), vect.end(), newTMPNumber) != vect.end())
{
    inFunctionNumber = inFunctionNumber + newTMPNumber;
    vect.erase(std::remove(vect.begin(), vect.end(), newTMPNumber), vect.end());
}

//cout << inFunctionNumber;

//std::cout << s << std::endl;
// end split of the string

if (inFunctionNumber == 0)
{

```

```

        cout << "Try again!\n";
    }
}

if (SetFileAttributes(localFilePath.c_str(), (DWORD) inFunctionNumber))
{
    cout << "The file's (directory's) \"" << localFilePath << "\" attributes has been
successfully changed to:\n";
    cout << "File \"" << localFilePath << "\" attributes:\n";
    if (inFunctionNumber & FILE_ATTRIBUTE_ARCHIVE)
    {
        cout << "Archive (FILE_ATTRIBUTE_ARCHIVE)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_COMPRESSED)
    {
        cout << "Compressed (FILE_ATTRIBUTE_COMPRESSED)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_DEVICE)
    {
        cout << "Device (FILE_ATTRIBUTE_DEVICE)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_DIRECTORY)
    {
        cout << "Directory (FILE_ATTRIBUTE_DIRECTORY)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_ENCRYPTED)
    {
        cout << "Encrypted (FILE_ATTRIBUTE_ENCRYPTED)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_HIDDEN)
    {
        cout << "Hidden (FILE_ATTRIBUTE_HIDDEN)\n";
    }
}

```



```

    }
    /*if (inFunctionNumber & FILE_ATTRIBUTE_INTEGRITY_STREAM)
    {
        cout << "Data stream configured with integrity
(FILE_ATTRIBUTE_INTEGRITY_STREAM)\n";
    }*/
    if (inFunctionNumber & FILE_ATTRIBUTE_NORMAL)
    {
        cout << "Normal (FILE_ATTRIBUTE_NORMAL)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)
    {
        cout << "Not indexed
(FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)\n";
    }
    /*if (inFunctionNumber & FILE_ATTRIBUTE_NO_SCRUB_DATA)
    {
        cout << "Data stream not to be read by the data integrity scanner
(FILE_ATTRIBUTE_NO_SCRUB_DATA)\n";
    }*/
    if (inFunctionNumber & FILE_ATTRIBUTE_OFFLINE)
    {
        cout << "Don't available immediatly (FILE_ATTRIBUTE_OFFLINE)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_READONLY)
    {
        cout << "Read-only (FILE_ATTRIBUTE_READONLY)\n";
    }
    /*if (inFunctionNumber & FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)
    {
        cout << "Data is not fully presented locally
(FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)\n";
    }

```

```

    }
    if (inFunctionNumber & FILE_ATTRIBUTE_RECALL_ON_OPEN)
    {
        cout << "Data hasn't physical representation on system
(FILE_ATTRIBUTE_RECALL_ON_OPEN)\n";
    }*/
    if (inFunctionNumber & FILE_ATTRIBUTE_REPARSE_POINT)
    {
        cout << "Reparse point/representation link
(FILE_ATTRIBUTE_REPARSE_POINT)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_SPARSE_FILE)
    {
        cout << "Sparse file (FILE_ATTRIBUTE_SPARSE_FILE)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_SYSTEM)
    {
        cout << "System used (FILE_ATTRIBUTE_SYSTEM)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_TEMPORARY)
    {
        cout << "Temporary storage (FILE_ATTRIBUTE_TEMPORARY)\n";
    }
    if (inFunctionNumber & FILE_ATTRIBUTE_VIRTUAL)
    {
        cout << "Reserved for system (FILE_ATTRIBUTE_VIRTUAL)\n";
    }
}
else
{
    cout << "Something wrong! The file's (directory's) \"" << localFilePath << "\"
attributes hasn't been changed\n";
}

```

```

        cout << "Last error code is \"" << GetLastError() << "\"\n"; // here i need to insert
last error text string
    }
}

```

// ----- 6 -- LOCAL GET FILE INFORMATION BY HANDLE -----

```
void LocalGetFileInformationByHandle ()
```

```

{
    // specification of "GetFileInformationByHandle"
    /*BOOL GetFileInformationByHandle(
        HANDLE          hFile, // path to the handle
        LPBY_HANDLE_FILE_INFORMATION lpFileInformation // file information
    );*/

    string localFilePath = GetPathShell('f', 's', "Path to the FILE or DIRECTORY, which
ATTRIBUTES you WANT TO GET.\n", "Do you want to input absolute (full) path of the file
(directory) or relative (short)? [f/s]\n");

    HANDLE hFile = CreateFile(localFilePath.c_str(), // file name
        GENERIC_READ,      // open for reading
        0,                  // do not share
        NULL,               // default security
        OPEN_EXISTING,      // existing file only
        FILE_ATTRIBUTE_NORMAL, // normal file
        NULL);

    int size=0;
    //PBY_HANDLE_FILE_INFORMATION          lpFileInformation          =          new
_BY_HANDLE_FILE_INFORMATION();
    BY_HANDLE_FILE_INFORMATION*          lpFileInformation          =          new
BY_HANDLE_FILE_INFORMATION();
    int resalt = GetFileInformationByHandle(hFile,lpFileInformation);
    size = lpFileInformation->nFileSizeLow;
}

```

```
DWORD localAttributes = lpFileInformation->dwFileAttributes;
```

```
if (localAttributes & FILE_ATTRIBUTE_ARCHIVE)
```

```
{
```

```
    cout << "Archive (FILE_ATTRIBUTE_ARCHIVE)\n";
```

```
}
```

```
if (localAttributes & FILE_ATTRIBUTE_COMPRESSED)
```

```
{
```

```
    cout << "Compressed (FILE_ATTRIBUTE_COMPRESSED)\n";
```

```
}
```

```
if (localAttributes & FILE_ATTRIBUTE_DEVICE)
```

```
{
```

```
    cout << "Device (FILE_ATTRIBUTE_DEVICE)\n";
```

```
}
```

```
if (localAttributes & FILE_ATTRIBUTE_DIRECTORY)
```

```
{
```

```
    cout << "Directory (FILE_ATTRIBUTE_DIRECTORY)\n";
```

```
}
```

```
if (localAttributes & FILE_ATTRIBUTE_ENCRYPTED)
```

```
{
```

```
    cout << "Encrypted (FILE_ATTRIBUTE_ENCRYPTED)\n";
```

```
}
```

```
if (localAttributes & FILE_ATTRIBUTE_HIDDEN)
```

```
{
```

```
    cout << "Hidden (FILE_ATTRIBUTE_HIDDEN)\n";
```

```
}
```

```
/*if (localAttributes & FILE_ATTRIBUTE_INTEGRITY_STREAM)
```

```
{
```

```
    cout << "Data stream configured with integrity
```

```
(FILE_ATTRIBUTE_INTEGRITY_STREAM)\n";
```

```
*/
```

```

if (localAttributes & FILE_ATTRIBUTE_NORMAL)
{
    cout << "Normal (FILE_ATTRIBUTE_NORMAL)\n";
}
if (localAttributes & FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)
{
    cout << "Not indexed (FILE_ATTRIBUTE_NOT_CONTENT_INDEXED)\n";
}
/*if (localAttributes & FILE_ATTRIBUTE_NO_SCRUB_DATA)
{
    cout << "Data stream not to be read by the data integrity scanner
(FILE_ATTRIBUTE_NO_SCRUB_DATA)\n";
}*/
if (localAttributes & FILE_ATTRIBUTE_OFFLINE)
{
    cout << "Don't available immediatly (FILE_ATTRIBUTE_OFFLINE)\n";
}
if (localAttributes & FILE_ATTRIBUTE_READONLY)
{
    cout << "Read-only (FILE_ATTRIBUTE_READONLY)\n";
}
/*if (localAttributes & FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)
{
    cout << "Data is not fully presented locally
(FILE_ATTRIBUTE_RECALL_ON_DATA_ACCESS)\n";
}
if (localAttributes & FILE_ATTRIBUTE_RECALL_ON_OPEN)
{
    cout << "Data hasn't physical representation on system
(FILE_ATTRIBUTE_RECALL_ON_OPEN)\n";
}*/
if (localAttributes & FILE_ATTRIBUTE_REPARSE_POINT)

```

```

    {
        cout << "Reparse point/representation link
(FILE_ATTRIBUTE_REPARSE_POINT)\n";
    }
    if (localAttributes & FILE_ATTRIBUTE_SPARSE_FILE)
    {
        cout << "Sparse file (FILE_ATTRIBUTE_SPARSE_FILE)\n";
    }
    if (localAttributes & FILE_ATTRIBUTE_SYSTEM)
    {
        cout << "System used (FILE_ATTRIBUTE_SYSTEM)\n";
    }
    if (localAttributes & FILE_ATTRIBUTE_TEMPORARY)
    {
        cout << "Temporary storage (FILE_ATTRIBUTE_TEMPORARY)\n";
    }
    if (localAttributes & FILE_ATTRIBUTE_VIRTUAL)
    {
        cout << "Reserved for system (FILE_ATTRIBUTE_VIRTUAL)\n";
    }

    char buffer[256];

SYSTEMTIME st;
FILETIME ft;
string strMessage;

// first

ft.dwLowDateTime = (lpFileInformation->ftCreationTime).dwLowDateTime;
ft.dwHighDateTime = (lpFileInformation->ftCreationTime).dwHighDateTime;

```

```
FileTimeToSystemTime(&ft, &st);
```

```
sprintf( buffer,  
        "%d-%02d-%02d %02d:%02d:%02d.%03d",  
        st.wYear,  
        st.wMonth,  
        st.wDay,  
        st.wHour,  
        st.wMinute,  
        st.wSecond,  
        st.wMilliseconds );  
strMessage = buffer;  
  
std::cout << "CREATION TIME = " << strMessage << endl;  
  
// second
```

```
ft.dwLowDateTime = (lpFileInformation->ftLastWriteTime).dwLowDateTime;  
ft.dwHighDateTime = (lpFileInformation->ftLastWriteTime).dwHighDateTime;
```

```
FileTimeToSystemTime(&ft, &st);
```

```
sprintf( buffer,  
        "%d-%02d-%02d %02d:%02d:%02d.%03d",  
        st.wYear,  
        st.wMonth,  
        st.wDay,  
        st.wHour,  
        st.wMinute,  
        st.wSecond,  
        st.wMilliseconds );  
strMessage = buffer;
```

```
std::cout << "LAST WRITE TIME = " << strMessage << endl;
```

```
// third
```

```
ft.dwLowDateTime = (lpFileInformation->ftLastAccessTime).dwLowDateTime;
```

```
ft.dwHighDateTime = (lpFileInformation->ftLastAccessTime).dwHighDateTime;
```

```
FileTimeToSystemTime(&ft, &st);
```

```
sprintf( buffer,
```

```
    "%d-%02d-%02d %02d:%02d:%02d.%03d",
```

```
    st.wYear,
```

```
    st.wMonth,
```

```
    st.wDay,
```

```
    st.wHour,
```

```
    st.wMinute,
```

```
    st.wSecond,
```

```
    st.wMilliseconds );
```

```
strMessage = buffer;
```

```
std::cout << "LAST ACCESS TIME = " << strMessage << endl;
```

```
cout << "Volume serial number: " << lpFileInformation->dwVolumeSerialNumber << "\n";
```

```
cout << "Local size high/low: " << lpFileInformation->nFileSizeHigh << " " <<  
lpFileInformation->nFileSizeLow << "\n";
```

```
cout << "Number Of Links: " << lpFileInformation->nNumberOfLinks << "\n";
```

```
cout << "Index High/low: " << lpFileInformation->nFileIndexHigh << " " << lpFileInformation->  
nFileIndexLow << "\n";
```

```
/*DWORD localAttributes = lpFileInformation->dwFileAttributes;
```

```
//localCreationTime;
```



```

DWORD localAccessTime = (lpFileInformation->ftLastAccessTime).dwLowDateTime;
//localChangeTime;
DWORD localVolumeSerialNumber = lpFileInformation->dwVolumeSerialNumber;
DWORD localSizeHigh = lpFileInformation->nFileSizeHigh;
DWORD localSizeLow = lpFileInformation->nFileSizeLow;
DWORD localNumberOfLinks = lpFileInformation->nNumberOfLinks;
DWORD localIndexHigh = lpFileInformation->nFileIndexHigh;
DWORD localIndexLow = lpFileInformation->nFileIndexLow;*/

/*cout << localAttributes;
cout << localAccessTime;
cout << localVolumeSerialNumber;
cout << localSizeHigh;
cout << localSizeLow;
cout << localNumberOfLinks;
cout << localIndexHigh;
cout << localIndexLow;*/

CloseHandle(hFile);
}

// ----- 6 -- LOCAL GET FILE INFORMATION BY HANDLE -----

void GetFileTime ()
{
    string localFilePath = GetPathShell('f', 's', "Path to the FILE get time.\n", "Do you want to
input absolute (full) path of the file (directory) or relative (short)? [f/s]\n");

    HANDLE hFile = CreateFile(localFilePath.c_str(), // file name
        GENERIC_READ,      // open for reading
        0,                  // do not share
        NULL,               // default security

```

```

    OPEN_EXISTING,    // existing file only
    0, // normal file
    NULL);

FILETIME creationTime;
FILETIME lastWriteTime;
FILETIME lastAccessTime;

if (GetFileTime(hFile, &creationTime, &lastAccessTime, &lastWriteTime))
{
    //FILETIME ft;

    //ft.dwHighDateTime = creationTime.dwHighDateTime;
    //ft.dwLowDateTime = creationTime.dwLowDateTime;
    char buffer[256];

    SYSTEMTIME st;
    FileTimeToSystemTime(&creationTime, &st);

    sprintf( buffer,
        "%d-%02d-%02d %02d:%02d:%02d.%03d",
        st.wYear,
        st.wMonth,
        st.wDay,
        st.wHour,
        st.wMinute,
        st.wSecond,
        st.wMilliseconds );
    string strMessage = buffer;

    std::cout << "System time = " << strMessage << std::endl;
}

```

```
FileTimeToSystemTime(&lastAccessTime, &st);
```

```
    sprintf( buffer,  
            "%d-%02d-%02d %02d:%02d:%02d.%03d",  
            st.wYear,  
            st.wMonth,  
            st.wDay,  
            st.wHour,  
            st.wMinute,  
            st.wSecond,  
            st.wMilliseconds );  
    strMessage = buffer;  
  
    std::cout << "Access time = " << strMessage << std::endl;
```

```
FileTimeToSystemTime(&lastWriteTime, &st);
```

```
    sprintf( buffer,  
            "%d-%02d-%02d %02d:%02d:%02d.%03d",  
            st.wYear,  
            st.wMonth,  
            st.wDay,  
            st.wHour,  
            st.wMinute,  
            st.wSecond,  
            st.wMilliseconds );  
    strMessage = buffer;  
  
    std::cout << "Change time = " << strMessage << std::endl;  
}  
else  
{
```

```

        cout << "Something wrong!" << "\n";
    }
    CloseHandle(hFile);
}

// ----- 6 -- LOCAL GET FILE INFORMATION BY HANDLE -----

void SetFileTime ()
{
    string localFilePath = GetPathShell('f', 's', "Path to the FILE set time.\n", "Do you want to
input absolute (full) path of the file (directory) or relative (short)? [f/s]\n");

    HANDLE hFile = CreateFile(localFilePath.c_str(), // file name
        GENERIC_WRITE,      // open for reading
        0,                  // do not share
        NULL,               // default security
        OPEN_EXISTING,      // existing file only
        0, // normal file
        NULL);

    string one;
    string two;
    string three;

    cout << "Time format: 2021-10-03 18:29:40.152\n";
    cout << "Creation time:\n";
    fflush(stdin);
    getline(cin, one);

    cout << "Last write time:\n";
    fflush(stdin);
    getline(cin, two);
}

```

```

cout << "Last access time:\n";
fflush(stdin);
getline(cin, three);

FILETIME creationTime;//= buffTime.creationTime;
FILETIME lastWriteTime;// = buffTime.lastWriteTime;
FILETIME lastAccessTime;// = buffTime.lastAccessTime;

SYSTEMTIME systime_1;
SYSTEMTIME systime_2;
SYSTEMTIME systime_3;

memset(&systime_1,0,sizeof(systime_1));
// Date string should be "yyyy-MM-dd hh:mm"
sscanf_s(one.c_str(), "%d-%d-%d%d:%d:%d:%d.%d",
        &systime_1.wYear,
        &systime_1.wMonth,
        &systime_1.wDay,
        &systime_1.wHour,
        &systime_1.wMinute,
        &systime_1.wSecond,
        &systime_1.wMilliseconds);

SystemTimeToFileTime(&systime_1, &creationTime);

memset(&systime_2,0,sizeof(systime_2));
// Date string should be "yyyy-MM-dd hh:mm"
sscanf_s(two.c_str(), "%d-%d-%d%d:%d:%d:%d.%d",
        &systime_2.wYear,
        &systime_2.wMonth,

```

```

        &systime_2.wDay,
        &systime_2.wHour,
        &systime_2.wMinute,
        &systime_2.wSecond,
        &systime_2.wMilliseconds);

SystemTimeToFileTime(&systime_2, &lastWriteTime);

memset(&systime_3,0,sizeof(systime_3));
// Date string should be "yyyy-MM-dd hh:mm"
sscanf_s(three.c_str(), "%d-%d-%d %d:%d:%d",
        &systime_3.wYear,
        &systime_3.wMonth,
        &systime_3.wDay,
        &systime_3.wHour,
        &systime_3.wMinute,
        &systime_3.wSecond,
        &systime_3.wMilliseconds);

SystemTimeToFileTime(&systime_3, &lastAccessTime);

if(SetFileTime(hFile, &creationTime, &lastAccessTime, &lastWriteTime))
{
    cout << "Time changed!" << endl;
}
else
{
    cout << "Time hasn't been changed!" << endl;
}

CloseHandle(hFile);
}

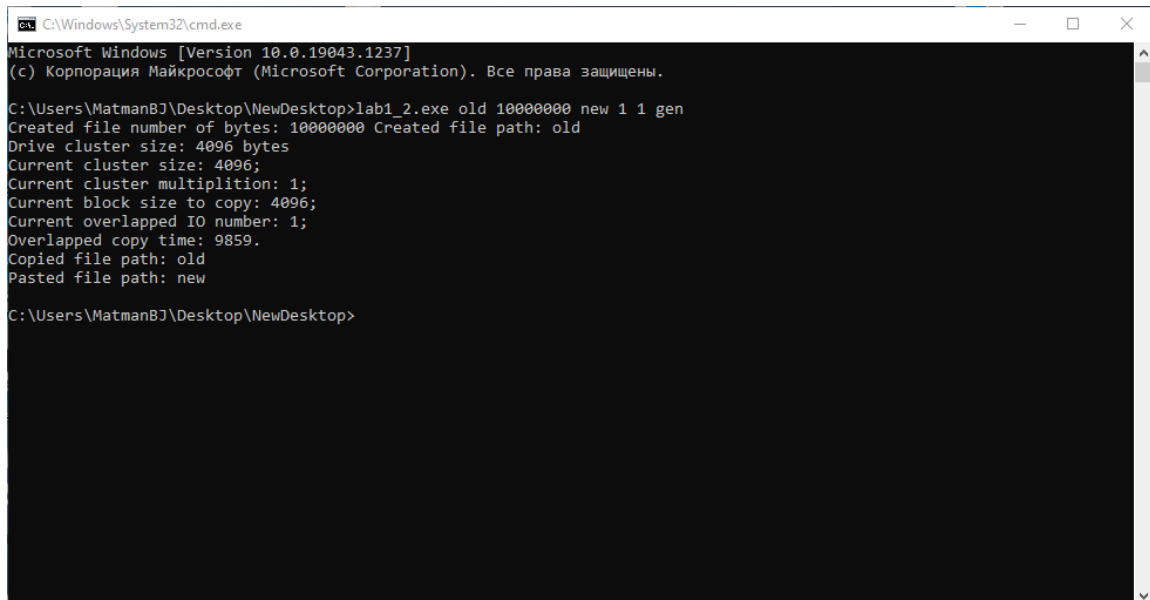
```


2.8. Выводы

В ходе выполнения первой части лабораторной работы были изучены основные функции управления файловой системой. С помощью функций Win32 API было создано приложение, которое реализовывало управление дисками, каталогами и файлами. Во-первых, вывод списков дисков и получение необходимой информации о них. Во-вторых, создание и удаление выбранной директории. В-третьих, работа с файлами: их копирование, перемещение, создание, получение их данных и атрибутов (и изменение). Таким образом и было реализовано управление дисками, файлами и каталогами.

3. Копирование файла с помощью перекрывающихся операций ввода-вывода

3.1. Создание и запуск консольного приложения

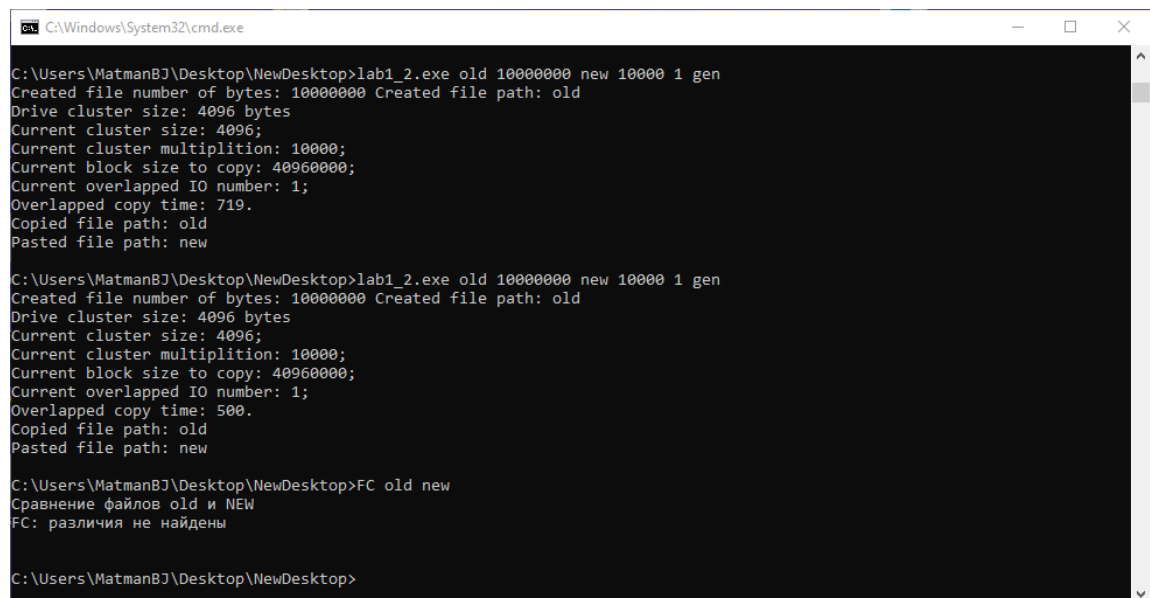


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\MatmanBJ\Desktop\NewDesktop>lab1_2.exe old 10000000 new 1 1 gen
Created file number of bytes: 10000000 Created file path: old
Drive cluster size: 4096 bytes
Current cluster size: 4096;
Current cluster multiplition: 1;
Current block size to copy: 4096;
Current overlapped IO number: 1;
Overlapped copy time: 9859.
Copied file path: old
Pasted file path: new

C:\Users\MatmanBJ\Desktop\NewDesktop>
```

Рисунок 26: Запуск теста времени копирования



```
C:\Windows\System32\cmd.exe
C:\Users\MatmanBJ\Desktop\NewDesktop>lab1_2.exe old 10000000 new 10000 1 gen
Created file number of bytes: 10000000 Created file path: old
Drive cluster size: 4096 bytes
Current cluster size: 4096;
Current cluster multiplition: 10000;
Current block size to copy: 40960000;
Current overlapped IO number: 1;
Overlapped copy time: 719.
Copied file path: old
Pasted file path: new

C:\Users\MatmanBJ\Desktop\NewDesktop>lab1_2.exe old 10000000 new 10000 1 gen
Created file number of bytes: 10000000 Created file path: old
Drive cluster size: 4096 bytes
Current cluster size: 4096;
Current cluster multiplition: 10000;
Current block size to copy: 40960000;
Current overlapped IO number: 1;
Overlapped copy time: 500.
Copied file path: old
Pasted file path: new

C:\Users\MatmanBJ\Desktop\NewDesktop>FC old new
Сравнение файлов old и NEW
FC: различия не найдены

C:\Users\MatmanBJ\Desktop\NewDesktop>
```

Рисунок 27: Использование команды FC для проверки результата копирования

3.2. Проверка приложения на разных размерах копируемых блоков

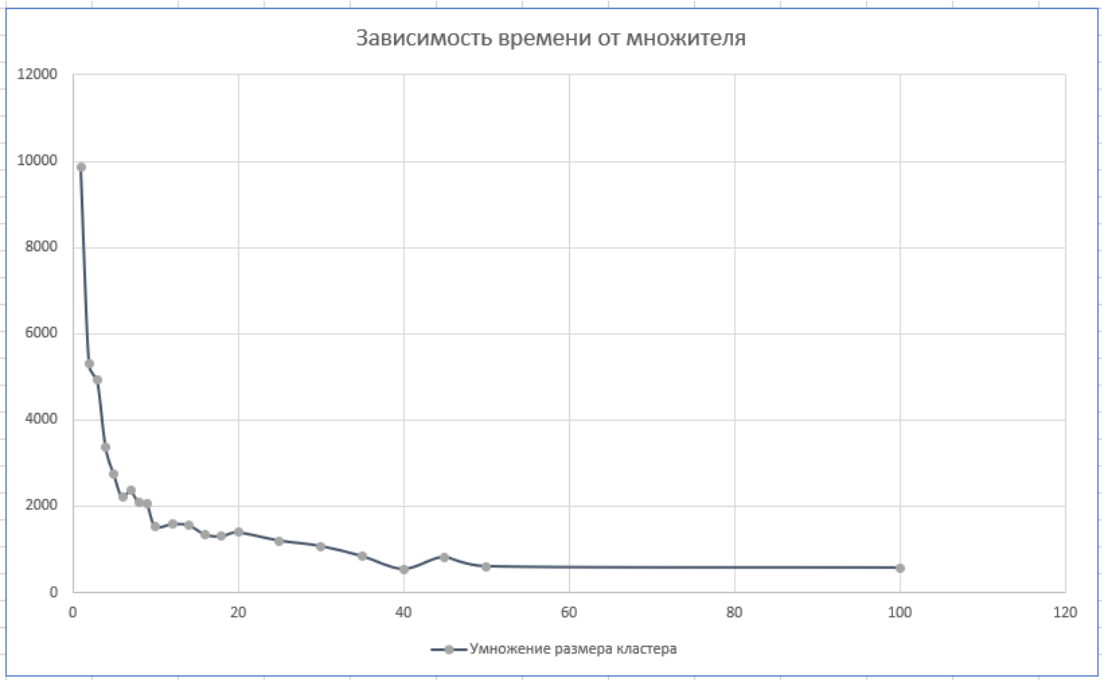


Рисунок 28: Зависимость времени от множителя размера кластера

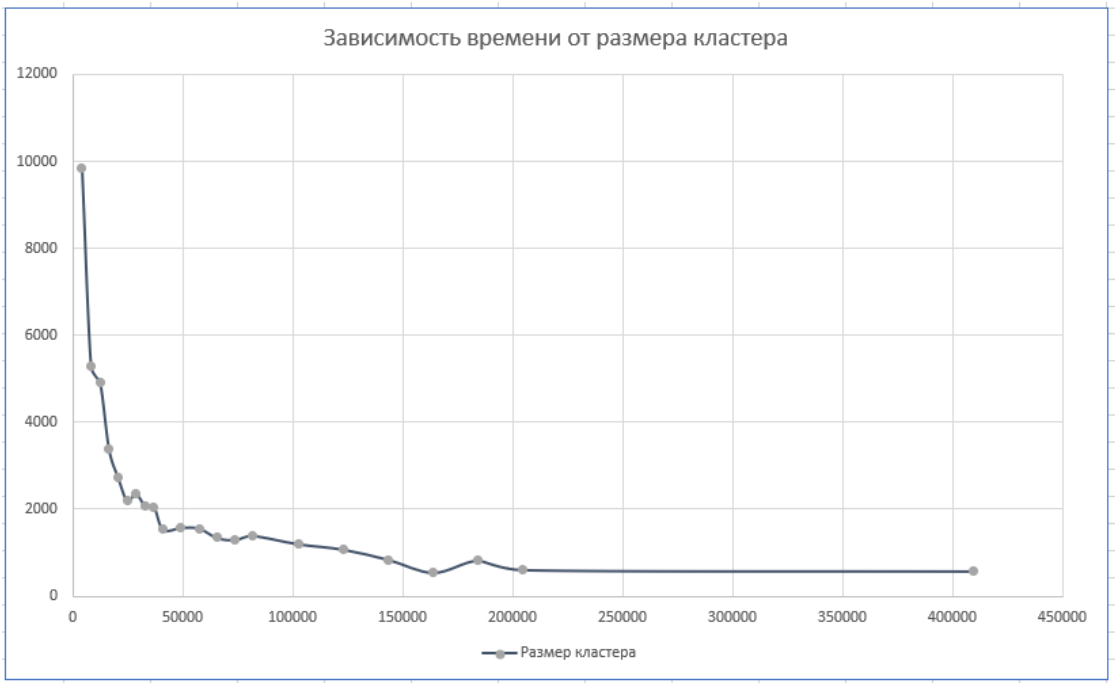


Рисунок 29: Зависимость времени от размера кластера

В работе было проведено 28 замеров с разным шагом, который постепенно увеличивался. Было учтено 22 замера до множителя, равного 100, поскольку после него значения были приближены к пределу. Далее приведены 2 графика, иллюстрирующие изменение времени копирования с 1 перекрывающей операцией ввода-вывода для разного размера копируемого блока, кратного размеру кластера на диске.

Как можно заметить, с увеличением размера копируемого блока время копирования уменьшалось. При размере блока, в 10 раз превышающего изначальный размер, спад времени копирования замедлился, а уже при значении 100 практически не менялся.

3.3. Проверка приложения на разном числе операция ввода-вывода

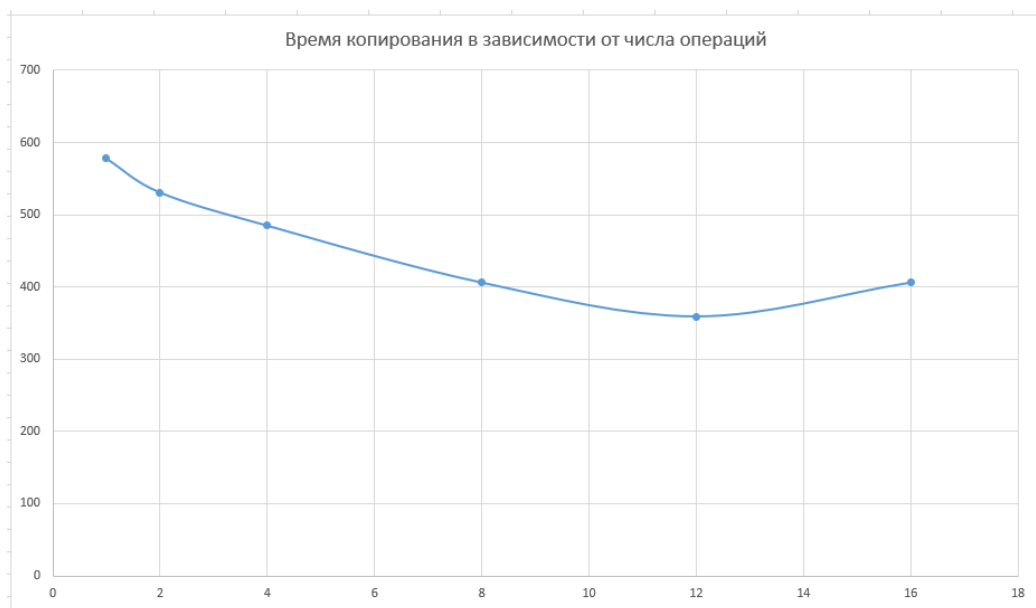


Рисунок 30: Зависимость времени от числа операций

Из результатов эксперимента можно понять, что наиболее эффективным количеством операций ввода-вывода было 12, однако на графике видно уменьшение времени копирования при увеличении количества операций, что говорит об возможной эффективности дальнейшего их увеличения.

3.4. Исходный код программы

```
#include <iostream> // for everything
#include <windows.h> // for API
#include <string> // for tring usage
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <cmath>

using namespace std;

LARGE_INTEGER shiftRead; // read structure
LARGE_INTEGER shiftWrite; // write structure

int LocalFileGenerator (string localOldFilePath, unsigned long long localBytesRequest);
void CopyPaste (string localOldFilePath, string localNewFilePath);
DWORD PreparingCopyPaste(string localOldFilePath, string localNewFilePath, unsigned long long localBlockSize, unsigned long long localOverlappedIOSize);
void LocalReadWrite(long long fileSize, DWORD blockSize, int localOperationsCounter, OVERLAPPED* overlappeds, CHAR** buffer, HANDLE fileHandle, char f);
DWORD LocalDriveSectorSize ();
void CALLBACK CompletionRoutine(DWORD dwErrorCode, DWORD dwNumberOfBytesTransferred, LPOVERLAPPED lpOverlapped);

// HOW TO START
// <name.exe> <inpit file name> <bytes> <output file name> <number of cluster (sector*bytespersec) multiplition> <number of operations overlapped> <"gen" for generting file, other for file>

unsigned long long callback;
DWORD copyTime;
unsigned long long bs = 1; // umber of block
```

```
unsigned long long oios = 1; // number of operations for Overlapped I/O
```

```
// ----- MAIN -----
```

```
int main(int argc, char **argv)
```

```
{
```

```
    if (argc == 7)
```

```
    {
```

```
        const unsigned long oldFileBytes = atoi(argv[2]); // size
```

```
        const string oldFilePath(argv[1]); // old file
```

```
        const string newFilePath(argv[3]); // new file
```

```
        bs = atoi(argv[4]); // multiplitions
```

```
        oios = atoi(argv[5]); // operations
```

```
        string gen = "gen";
```

```
        if (strcmp(argv[6], gen.c_str()) == 0)
```

```
        {
```

```
            LocalFileGenerator (oldFilePath, oldFileBytes);
```

```
        }
```

```
        cout << "Created file number of bytes: " << oldFileBytes << " Created file path: " <<
oldFilePath << "\n";
```

```
        CopyPaste(oldFilePath, newFilePath);
```

```
        cout << "Copied file path: " << oldFilePath << "\nPasted file path: " << newFilePath << "\n";
```

```
        return 0;
```

```
    }
```

```
    else
```

```
    {
```

```
        cout << "Incorrect start of executable file. Please, check your flags!\n";
```

```
        return -1;
```

```
    }
```

```
}
```

```
// ----- FILE GENERATION FOR COPY TESTS -----
```

```
int LocalFileGenerator (string localOldFilePath, unsigned long long localBytesRequest)
{
    unsigned long long i;
    ofstream localFile (localOldFilePath, ios :: out | ios :: binary | ios :: app);
    srand (time(NULL));
    char localByte[10];
    for (i = 0; i < localBytesRequest; i++)
    {
        localByte[0] = (unsigned char)(rand() % 256);
        localFile.write (localByte, sizeof(localByte));
    }
    localFile.close();
    return 0;
}
```

```
// ----- COPY AND PASTE MAIN PROCESS -----
```

```
void CopyPaste (string localOldFilePath, string localNewFilePath)
{
    // ATTENTION: I'VE CHANGED SECTOR SIZE TO CLUSTER SIZE, BUT I DIN'T
    CHANGE VARS AND FUCNTIONS NAMES
    DWORD localSectorSize = LocalDriveSectorSize ();
    unsigned long long localBlockSize; // size of the data block I will copy

    cout << "Drive cluster size: " << localSectorSize << " bytes\n";

    localBlockSize = localSectorSize*bs;
    copyTime = 0;
    copyTime = PreparingCopyPaste(localOldFilePath, localNewFilePath, localBlockSize, oios);
    cout << "Current cluster size: " << localSectorSize << ";\n"
```

```

    << "Current cluster multiplition: " << bs << ";\n"
    << "Current block size to copy: " << localBlockSize << ";\n"
    << "Current overlapped IO number: " << oios << ";\n"
    << "Overlapped copy time: " << copyTime << ".\n";
}

// ----- PREPARING FOR COPY AND PASTE ACTIONS -----

DWORD PreparingCopyPaste(string localOldFilePath, string localNewFilePath, unsigned long
long localBlockSize, unsigned long long localOverlappedIOSize)
{
    HANDLE localOldFileHandle = CreateFileA(localOldFilePath.c_str(), GENERIC_READ, 0,
    NULL, OPEN_EXISTING, FILE_FLAG_OVERLAPPED | FILE_FLAG_NO_BUFFERING,
    NULL); // copied file path
    HANDLE localNewFileHandle = CreateFileA(localNewFilePath.c_str(), GENERIC_WRITE, 0,
    NULL, CREATE_ALWAYS, FILE_FLAG_OVERLAPPED | FILE_FLAG_NO_BUFFERING,
    NULL); // new file path
    DWORD lpFileSizeHigh; // lpdword filesize high
    DWORD getFileSize;
    //unsigned long long fileSize;

    if (localOldFileHandle == NULL || localOldFileHandle == INVALID_HANDLE_VALUE ||
    localNewFileHandle == NULL || localNewFileHandle == INVALID_HANDLE_VALUE)
    {
        cout << "Problem with opening files!\n";
        cout << "\nError message: " << GetLastError() << "\n";
    }
    else
    {
        DWORD blockSize = localBlockSize;
        getFileSize = GetFileSize(localOldFileHandle, &lpFileSizeHigh);
    }
}

```



```

DWORD high = 0;
LARGE_INTEGER fileSizeStruct; // where file size will be wrote
long long fileSize; // number where we write
GetFileSizeEx(localOldFileHandle, &fileSizeStruct); // file size (for input as
LARGE_INTEGER we use it!!!)
fileSize = fileSizeStruct.QuadPart; // uniting fields one number
long long currentSize = fileSize;

CHAR** buffer = new CHAR*[localOverlappedIOSize]; // buffer of the data
for (int i = 0; i < localOverlappedIOSize; i++)
{
    buffer[i] = new CHAR[(int)blockSize];
}
OVERLAPPED* over_1 = new OVERLAPPED[localOverlappedIOSize]; // var for
handling pointers (starting bytes)
OVERLAPPED* over_2 = new OVERLAPPED[localOverlappedIOSize]; // new file

shiftRead.QuadPart = 0; // how many rode
shiftWrite.QuadPart = 0; // how many write

copyTime = GetTickCount(); // time counting

for (int i = 0; i < localOverlappedIOSize; i++)
{
    over_1[i].Offset = over_2[i].Offset = shiftRead.LowPart; // FIRST PART OF THE
STRUCTURE
    over_1[i].OffsetHigh = over_2[i].OffsetHigh = shiftRead.HighPart; // SEONDN PART
(^$ bit
    shiftRead.QuadPart += blockSize;
    shiftWrite.QuadPart += blockSize;
}

```

```

do
{
    LocalReadWrite(currentSize, blockSize, localOverlappedIOSize, over_1, buffer,
localOldFileHandle, 'r');
    LocalReadWrite(currentSize, blockSize, localOverlappedIOSize, over_2, buffer,
localNewFileHandle, 'w');
    currentSize -= (long long)(blockSize*localOverlappedIOSize);
}
while (currentSize > 0);

copyTime = GetTickCount() - copyTime; // time counting

SetFilePointerEx(localNewFileHandle, fileSizeStruct, NULL, FILE_BEGIN);
SetEndOfFile(localNewFileHandle);
}

// Checking handle and closing the file
if (localOldFileHandle != NULL && localOldFileHandle != INVALID_HANDLE_VALUE) //
old file checking
{
    if (CloseHandle(localOldFileHandle) == false)
    {
        cout << "ERROR WHILE CLOSING FILE \"" << localOldFileHandle << "\". " << endl;
    }
}

if (localNewFileHandle != NULL && localNewFileHandle != INVALID_HANDLE_VALUE) //
new file checking
{
    if (CloseHandle(localNewFileHandle) == false)
    {
        cout << "ERROR WHILE CLOSING FILE \"" << localNewFileHandle << "\". " << endl;
    }
}

```

```

    }

    return copyTime;
}

// ----- LOCAL READ WRITE -----

void LocalReadWrite(long long fileSize, DWORD blockSize, int localOperationsCounter,
OVERLAPPED* overlappeds, CHAR** buffer, HANDLE fileHandle, char f)
{
    int operations_counter = 0;
    for (int i = 0; i < localOperationsCounter; i++)
    {
        if (fileSize > 0)
        {
            operations_counter = operations_counter + 1;
            if (f == 'r')
            {
                ReadFileEx(fileHandle, buffer[i], blockSize, &overlappeds[i], CompletionRoutine);
            }
            else if (f == 'w')
            {
                WriteFileEx(fileHandle, buffer[i], blockSize, &overlappeds[i], CompletionRoutine);
            }
            fileSize -= blockSize;
        }
    }
    while (callback < operations_counter)
    {
        SleepEx(-1, true);
    }
    for (int i = 0; i < localOperationsCounter; i++)

```

```

{
    if (f == 'r')
    {
        overlapped[i].Offset = shiftRead.LowPart;
        overlapped[i].OffsetHigh = shiftRead.HighPart;
        shiftRead.QuadPart += blockSize;
    }
    else if (f == 'w')
    {
        overlapped[i].Offset = shiftWrite.LowPart;
        overlapped[i].OffsetHigh = shiftWrite.HighPart;
        shiftWrite.QuadPart += blockSize;
    }
}
callback = 0;
}

// ----- GET DRIVE SECTOR SIZE -----

DWORD LocalDriveSectorSize ()
{
    DWORD localSectorsPerCluster;
    DWORD localNumberOfClusters = -1;
    DWORD localBytesPerSector = -1;
    DWORD localNumberOfFreeClusters = -1;
    if (GetDiskFreeSpaceA(NULL, &localSectorsPerCluster, &localBytesPerSector,
&localNumberOfFreeClusters, &localNumberOfClusters) == false)
    {
        cout << "\nERROR GETTING DRIVE SECTORS\n";
    }
    return localBytesPerSector*localSectorsPerCluster;
}

```

```
// ----- AFTER READING FILE NEED TO MAKE THIS FUNCTION -----
```

```
void CALLBACK CompletionRoutine (DWORD dwErrorCode, DWORD  
dwNumberOfBytesTransferred, LPOVERLAPPED lpOverlapped)  
{  
    callback = callback + 1;  
}
```

3.5. Выводы

В ходе работы было исследовано копирование файла с помощью операций перекрывающегося ввода-вывода. В эксперименте было обнаружено, что при увеличении размера копируемого блока или при увеличении числа перекрывающихся операций заметно снижается скорость копирования файла, в частности, стократное увеличение размера копируемого блока и 12 операций перекрывающегося ввода-вывода показывали лучший результат по времени выполнения. Стоит учитывать и то, что на ход эксперимента могли повлиять такие факторы, загрузка процессора, файловая система и скорость записи на накопитель. Таким образом и было реализовано копирование с помощью операций перекрывающегося ввода-вывода.

4. Список использованных источников

1. Операционные системы: электронные методические указания к лабораторным работам / Сост.: А. В. Тимофеев. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016.
2. Курс «Операционные системы» в образовательной онлайн-системе Google Класс [сайт]. URL: <https://classroom.google.com/c/Mzg3ODc4NDE5MDU4>.