МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 10**

**по дисциплине «Операционные системы»**

**Тема: «Работа с файлами»**

Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

Москва, 2025г

**Работа с файлами**

1. Разработать приложение на С++, которое позволяет:

- Вывести содержимое файла

- Копировать содержимое файла в другой файл

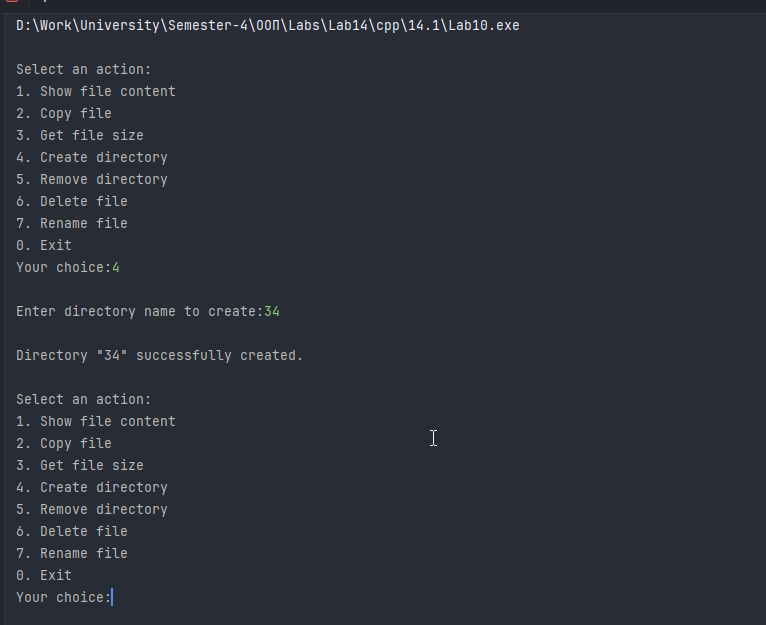
- Определить размер файла

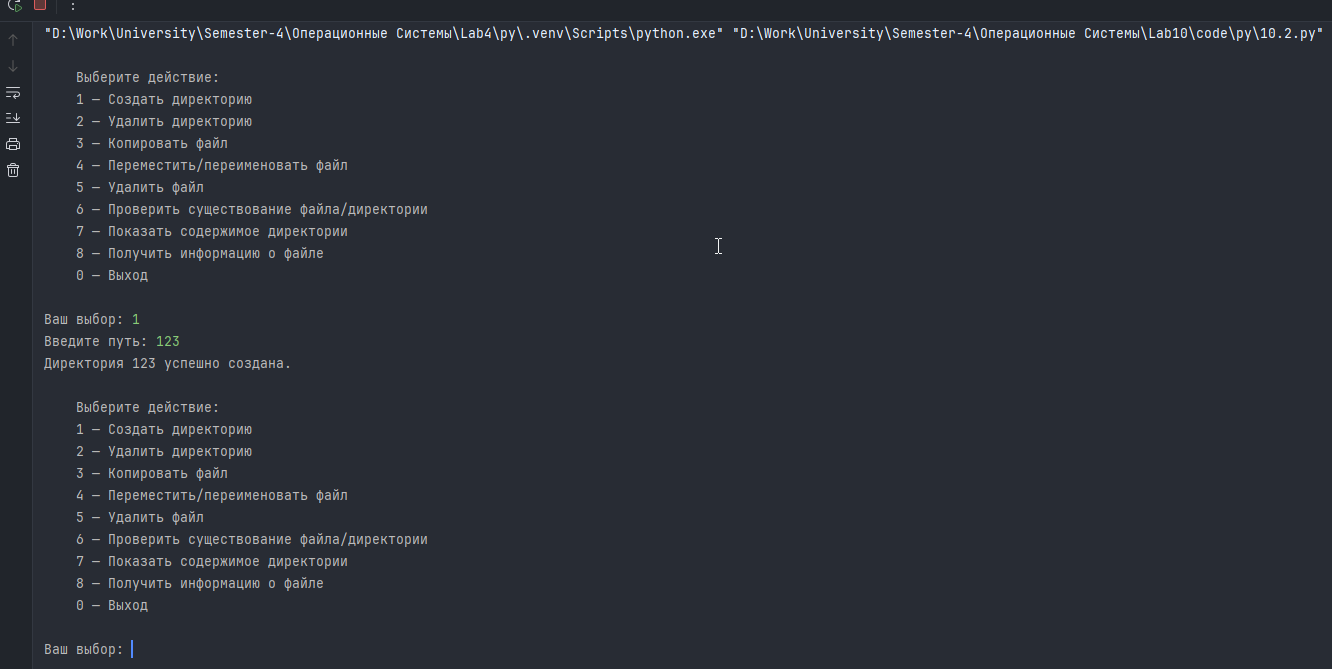
- Создать папку

- Удалить папку

- Удалить файл

2. Реализовать следующие примеры приложений на Python, которые демонстрируют различные операции с файлами и директориями, по необходимости дополнить или исправить программный код.

  
*Рисунок 1. Работающее приложение*

  
*Рисунок 2. Пример работающего приложений на питоне*

**Листинг кода**

**1.cpp**##include <iostream>

#include <fstream>

#include <string>

#include <filesystem>

using namespace std;

namespace fs = filesystem;

void printFileContent(const string&);

void copyFile(const string&, const string&);

void getFileSize(const string&);

void createDirectory(const string&);

void removeDirectory(const string&);

void removeFile(const string&);

void renameFile(const string&, const string&);

int main() {

unsigned short choice;

string file1, file2;

do {

cout << "\nSelect an action:\n";

cout << "1. Show file content\n";

cout << "2. Copy file\n";

cout << "3. Get file size\n";

cout << "4. Create directory\n";

cout << "5. Remove directory\n";

cout << "6. Delete file\n";

cout << "7. Rename file\n";

cout << "0. Exit\n";

cout << "Your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter file name: ";

cin >> file1;

printFileContent(file1);

break;

case 2:

cout << "Enter source file name: ";

cin >> file1;

cout << "Enter destination file name: ";

cin >> file2;

copyFile(file1, file2);

break;

case 3:

cout << "Enter file name: ";

cin >> file1;

getFileSize(file1);

break;

case 4:

cout << "Enter directory name to create: ";

cin >> file1;

createDirectory(file1);

break;

case 5:

cout << "Enter directory name to remove: ";

cin >> file1;

removeDirectory(file1);

break;

case 6:

cout << "Enter file name to delete: ";

cin >> file1;

removeFile(file1);

break;

case 7:

cout << "Enter current file name: ";

cin >> file1;

cout << "Enter new file name: ";

cin >> file2;

renameFile(file1, file2);

break;

case 0:

break;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 0);

system("pause");

return 0;

}

// Print file content

void printFileContent(const string& fileName) {

ifstream file(fileName);

if (!file) {

cerr << "Failed to open file: " << fileName << endl;

return;

}

string line;

cout << "Content of file \"" << fileName << "\":" << endl;

while (getline(file, line)) {

cout << line << endl;

}

file.close();

}

// Copy file content to another file

void copyFile(const string& source, const string& destination) {

ifstream src(source, ios::binary);

ofstream dest(destination, ios::binary);

if (!src || !dest) {

cerr << "Error opening files." << endl;

return;

}

dest << src.rdbuf();

cout << "File successfully copied from \"" << source << "\" to \"" << destination << "\"" << endl;

src.close();

dest.close();

}

// Determine file size

void getFileSize(const string& fileName) {

if (!fs::exists(fileName)) {

cerr << "File does not exist: " << fileName << endl;

return;

}

auto size = fs::file\_size(fileName);

cout << "Size of file \"" << fileName << "\": " << size << " bytes" << endl;

}

// Create directory

void createDirectory(const string& dirName) {

if (fs::exists(dirName)) {

cerr << "Directory \"" << dirName << "\" already exists." << endl;

return;

}

try {

if (fs::create\_directory(dirName)) {

cout << "Directory \"" << dirName << "\" successfully created." << endl;

} else {

cerr <<"Failed to create directory \"" << dirName << "\"." << endl;

}

} catch (const fs::filesystem\_error& e) {

cerr << "Error creating directory: " << e.what() << endl;

}

}

// Remove directory

void removeDirectory(const string& dirName) {

if (!fs::exists(dirName)) {

cerr << "Directory \"" << dirName << "\" does not exist." << endl;

return;

}

if (!fs::is\_directory(dirName)) {

cerr << "\"" << dirName << "\" is not a directory." << endl;

return;

}

try {

uintmax\_t removedCount = fs::remove\_all(dirName);

cout << "Directory \"" << dirName << "\" successfully removed with " << removedCount << " elements." << endl;

} catch (const fs::filesystem\_error& e) {

cerr << "Error removing directory: " << e.what() << endl;

}

}

// Delete file

void removeFile(const string& fileName) {

if (!fs::exists(fileName)) {

cerr << "File \"" << fileName << "\" does not exist." << endl;

return;

}

if (fs::is\_directory(fileName)) {

cerr << "\"" << fileName << "\" is a directory, not a file." << endl;

return;

}

try {

if (fs::remove(fileName)) {

cout << "File \"" << fileName << "\" successfully deleted." << endl;

} else {

cerr << "Failed to delete file \"" << fileName << "\"." << endl;

}

} catch (const fs::filesystem\_error& e) {

cerr << "Error deleting file: " << e.what() << endl;

}

}

// Rename file

void renameFile(const string& oldName, const string& newName) {

if (!fs::exists(oldName)) {

cerr << "File \"" << oldName << "\" does not exist." << endl;

return;

}

if (fs::exists(newName)) {

cerr << "File with name \"" << newName << "\" already exists." << endl;

return;

}

try {

fs::rename(oldName, newName);

cout << "File \"" << oldName << "\" successfully renamed to \"" << newName << "\"." << endl;

} catch (const fs::filesystem\_error& e) {

cerr << "Error renaming file: " << e.what() << endl;

}

}

**2.py**

// import os

import shutil

import time

#---------------------1------------------------------

def create\_directory(path):

try:

os.makedirs(path)

print(f"Директория {path} успешно создана.")

except FileExistsError:

print(f"Директория {path} уже существует!")

except Exception as e:

print(f"Ошибка при создании директории: {e}")

#---------------------2------------------------------

def delete\_directory(path):

if not os.path.exists(path):

print(f"Директория {path} не найдена!")

return

if os.listdir(path):

print(f"Директория {path} не пуста!")

confirm = input("Уверены, что хотите удалить папку?(y/n): ").lower()

if confirm != 'y':

print("Отмена удаления.")

return

try:

shutil.rmtree(path)

print(f"Директория {path} удалена.")

except Exception as e:

print(f"Ошибка при удалении: {e}")

#---------------------3------------------------------

def copy\_file(src, dst):

try:

shutil.copy2(src, dst)

print(f"Файл скопирован из '{src}' в '{dst}'")

except Exception as e:

print(f"Ошибка при копировании: {e}")

#---------------------4------------------------------

def move\_or\_rename\_file(src, dst):

try:

shutil.move(src, dst)

print(f"Файл перемещён или переименован в '{dst}'")

except Exception as e:

print(f"Ошибка при перемещении/переименовании: {e}")

#---------------------5------------------------------

def delete\_file(path):

try:

os.remove(path)

print(f"Файл '{path}' удалён.")

except FileNotFoundError:

print("Файл не найден.")

except Exception as e:

print(f"Ошибка при удалении файла: {e}")

#---------------------6------------------------------

def check\_existence(path):

if os.path.exists(path):

if os.path.isdir(path):

print(f"'{path}' — директория.")

elif os.path.isfile(path):

print(f"'{path}' — файл.")

else:

print(f"'{path}' существует, но это не файл и не директория.")

else:

print(f"'{path}' не существует.")

#---------------------7------------------------------

def list\_directory(path):

if not os.path.isdir(path):

print(f"Путь '{path}' не является директорией.")

return

print(f"Содержимое директории '{path}':")

try:

for item in os.listdir(path):

print("•", item)

except Exception as e:

print(f"Ошибка при чтении директории: {e}")

#---------------------8------------------------------

def file\_info(path):

if not os.path.isfile(path):

print(f"'{path}' не является файлом.")

return

try:

size = os.path.getsize(path)

modified\_time = os.path.getmtime(path)

print(f"Информация о файле '{path}':")

print(f"Размер: {size} байт")

print(f"Последнее изменение: {time.ctime(modified\_time)}")

except Exception as e:

print(f"Ошибка при получении информации: {e}")

#---------------------------------------------------

def main():

actions\_with\_path = {

1: create\_directory,

2: delete\_directory,

5: delete\_file,

6: check\_existence,

7: list\_directory,

8: file\_info

}

while True:

print("""

Выберите действие:

1 — Создать директорию

2 — Удалить директорию

3 — Копировать файл

4 — Переместить/переименовать файл

5 — Удалить файл

6 — Проверить существование файла/директории

7 — Показать содержимое директории

8 — Получить информацию о файле

0 — Выход

""")

try:

choice = int(input("Ваш выбор: "))

except ValueError:

print("Ошибка: введите номер действия (0-8).")

continue

if choice in actions\_with\_path:

path = input("Введите путь: ").strip()

actions\_with\_path[choice](path)

continue

match choice:

case 3:

src = input("Путь к исходному файлу: ").strip()

dst = input("Путь назначения: ").strip()

copy\_file(src, dst)

case 4:

src = input("Путь к исходному файлу: ").strip()

dst = input("Новый путь или новое имя: ").strip()

move\_or\_rename\_file(src, dst)

case 0:

print("Выход из программы.")

break

case \_:

print("Неверный выбор. Попробуйте снова.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Вывод:** В ходе выполнения лабораторной работы было разработано консольное приложение на C++, предоставляющее функциональность для выполнения базовых операций с файлами и директориями: просмотр содержимого, копирование, получение размера, создание и удаление папок, удаление и переименование файлов. Приложение реализовано с использованием библиотеки filesystem из стандарта C++17, что обеспечивает кроссплатформенность и надежную обработку ошибок. В программе предусмотрено корректное закрытие файловых потоков для предотвращения ошибок удержания процесса, а также добавлены проверки существования файлов и директорий перед выполнением операций. Дополнительно в рамках лабораторной работы были изучены и реализованы примеры различных операций с файлами на языке Python, демонстрирующие работу с файловой системой с использованием модулей os и shutil.