Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе №4

на тему

Работа с файлами

Студент П. С. Павлюткин

Преподаватель С. И. Сиротко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Цель работы изучить и применить на практике знания о работе с файлами на *Windows*, научиться использовать расширенные/дополнительные возможности и механизмы работы с файлами и организации ввода-вывода: неблокирующие и асинхронные операции, мультиплексирование ввода-вывода, отображение файлов в память.Для достижения цели будет создано приложение обрабатывающие содержимое файла, используя асинхронные операции.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Подсистема ввода-вывода, разновидности ввода-вывода, логические устройства. Файл и файловая система. Иерархия, виды файлов, файлы и директории (папки). Открытый файл (устройство) как объект, дескрипторный доступ, файловые дескрипторы (*Handle*). Основной набор *API* для работы с файлами: *CreateFile*(), *CloseHandle*(), *ReadFile*(), *WriteFile*(). Синхронный ввод-вывод, в т.ч. неблокирующий. Асинхронный («перекрывающийся») ввод-вывод.

В операционной системе Windows, файлы и папки организованы в файловой системе, которая предоставляет структуру для хранения и управления данными.

Для работы с файловой системой в приложениях на C++, вы можете использовать стандартную библиотеку ввода-вывода (I/O), такую как `*fstream*`, `*ifstream*`, и `*ofstream*`. Важные операции включают открытие, чтение, запись, закрытие и удаление файлов.

Для открытия файла в C++, вы можете использовать объекты `*ifstream*` (для чтения), `*ofstream*` (для записи) или `*fstream*` (для чтения и записи).

Файл можно открыть с определенным режимом, таким как в текстовом или бинарном режиме. Режим файла определяет, как данные будут интерпретироваться. Файлы должны быть закрыты после того, как к ним больше нет необходимости, чтобы освободить ресурсы. Для чтения данных из файла в C++, используйте `*ifstream*`. Вы можете читать данные в переменные вашей программы. Для записи данных в файл используйте `*ofstream*`. Вы можете записывать данные из переменных программы в файл. В текстовом режиме файлы интерпретируются как последовательности символов, в бинарном режиме - как байтов. Windows предоставляет функции для работы с файлами и директориями, такие как создание, удаление, перемещение и переименование. Для выполнения операций с директориями в *C++,* можно использовать библиотеку `<*filesystem*>`, введенную в стандарте *C++17*.

При работе с файлами важно обрабатывать ошибки. Файлы могут отсутствовать, права доступа могут быть ограничены, и другие проблемы могут возникнуть.

Используйте проверку успешности операций ввода-вывода и обработку исключений для обработки ошибок. В современных приложениях можно реализовать асинхронное чтение и запись файлов, что может повысить производительность, когда обработка больших объемов данных. Важно оценить эффективность ваших операций с файлами, особенно при работе с большими файлами и при обработке данных. Измеряйте и сравнивайте производительность различных методов. Всегда закрывайте файлы и освобождайте ресурсы, чтобы избежать утечек памяти и других проблем.

В завершении работы вашего приложения убедитесь, что все ресурсы освобождены.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создано приложение, обрабатывающие содержимое файла, используя асинхронные операции (Рисунок 1).

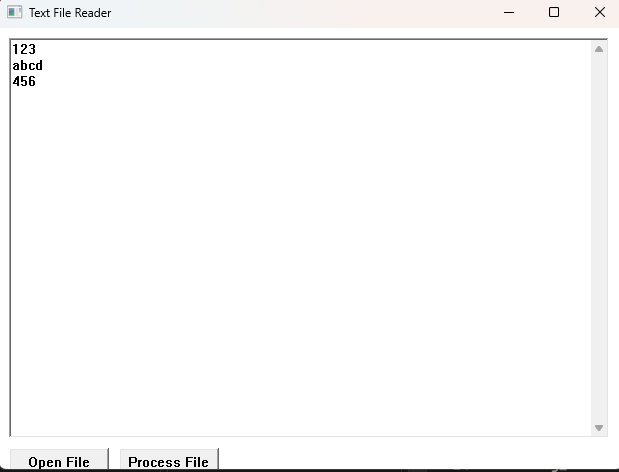


Рисунок 1 – Окно приложения с выводом текстового файла в клиентскую область окна

В нашем приложении есть две кнопки, первая, открывающая файл обычным способом и вторая кнопка, уже открывающая файл, используя асинхронные операции. Результат выполнения программы идентичен, так как это никак не влияет на сами данные (Рисунок 2).

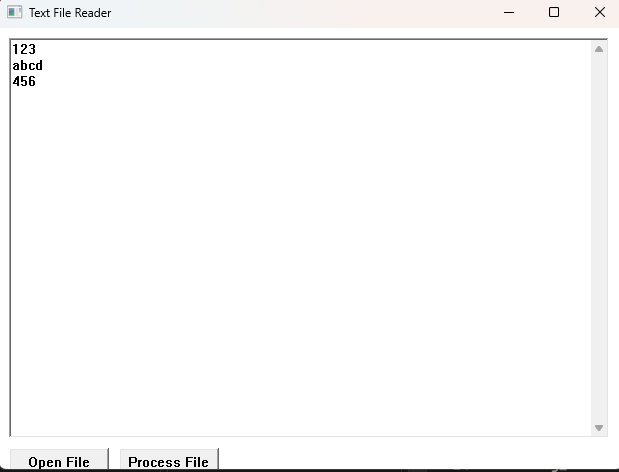


Рисунок 2– Выход за игровое поле

Наше приложение демонстрирует то, как можно читать содержимое файла, используя асинхронные операции.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были изучены и применены навыки работы с файлами и создания *Windows*-приложения с использованием *WinAPI* и языка программирования *C++.* В частности, следующие важные аспекты были рассмотрены и применены:

Была реализована функциональность открытия, чтения и обработки текстовых файлов. Это включает в себя использование диалогового окна для выбора файлов, чтение содержимого файла и вывод его в текстовом поле приложения. В лабораторной работе была продемонстрирована асинхронная обработка файлов с использованием многопоточности. Это позволило запускать операции чтения файла параллельно с выполнением других задач, не блокируя пользовательский интерфейс. В ходе работы был использован *WinAPI* для создания пользовательского интерфейса *Windows*-приложения. Это включало в себя создание окон, элементов управления и обработку сообщений. Важной частью кода было правильное управление ресурсами, включая закрытие файлов и корректное завершение работы приложения.

Данная лабораторная работа предоставила практический опыт в создании *Windows*-приложений, взаимодействии с файловой системой и реализации асинхронных операций ввода-вывода. Эти навыки могут быть полезными при разработке разнообразных приложений, обрабатывающих файлы и работающих с данными.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. *Win32 API*. Разработка приложений для *Windows*. ─ СПБ: Питер, 2008. ─ 592 с.: ип.
2. Создание классических приложений для *Windows* с использованием *API Win32* [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл Lab4.cpp

#include <windows.h>

#include <tchar.h>

#include <string>

#include <fstream>

#include <thread>

#define ID\_FILE\_OPEN 1001

#define ID\_PROCESS\_FILE 1002

/\*

Этот код представляет собой простое Windows-приложение с использованием WinAPI, которое выполняет следующие функции:

1. Открывает окно приложения с текстовым полем (многострочным редактором) и двумя кнопками: "Open File" и "Process File".

2. При нажатии на "Open File" открывается стандартное диалоговое окно для выбора текстового файла. После выбора файла его содержимое читается с использованием `std::wifstream` и выводится в текстовом поле в окне приложения.

3. При нажатии на "Process File" также открывается диалоговое окно для выбора текстового файла. Однако, в этом случае обработка файла выполняется асинхронно в новом потоке. Для этого используется `std::thread`, который запускает функцию `ProcessFileAsync`, которая читает содержимое файла и выводит его в текстовое поле.

4. Окно приложения содержит обработчик сообщений `WndProc`, который обрабатывает различные события, такие как нажатие кнопок, закрытие окна и другие.

Этот код просто демонстрирует базовую функциональность Windows-приложения для открытия и обработки текстовых файлов, как с использованием синхронного, так и с асинхронного подхода.

\*/

HWND g\_hEdit = NULL;

bool OpenFile(std::wstring& filePath) {

OPENFILENAME ofn;

TCHAR szFile[MAX\_PATH] = { 0 };

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.lpstrFilter = \_T("Text Files\0\*.txt\0All Files\0\*.\*\0");

ofn.nFilterIndex = 1;

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

if (GetOpenFileName(&ofn)) {

filePath = ofn.lpstrFile;

return true;

}

return false;

}

void ProcessFileAsync(const std::wstring& filePath) {

std::wifstream file(filePath);

if (file.is\_open()) {

std::wstring content;

std::wstring line;

while (std::getline(file, line)) {

content += line + L"\r\n";

}

file.close();

SetWindowText(g\_hEdit, content.c\_str());

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_CREATE:

g\_hEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, \_T("EDIT"), \_T(""), WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | ES\_MULTILINE | ES\_AUTOVSCROLL, 10, 10, 600, 400, hWnd, NULL, GetModuleHandle(NULL), NULL);

CreateWindow(\_T("BUTTON"), \_T("Open File"), WS\_CHILD | WS\_VISIBLE, 10, 420, 100, 30, hWnd, (HMENU)ID\_FILE\_OPEN, GetModuleHandle(NULL), NULL);

CreateWindow(\_T("BUTTON"), \_T("Process File"), WS\_CHILD | WS\_VISIBLE, 120, 420, 100, 30, hWnd, (HMENU)ID\_PROCESS\_FILE, GetModuleHandle(NULL), NULL);

return 0;

case WM\_COMMAND:

if (LOWORD(wParam) == ID\_FILE\_OPEN) {

std::wstring filePath;

if (OpenFile(filePath)) {

SetWindowText(g\_hEdit, \_T("")); // Очистим текстовое поле

ProcessFileAsync(filePath);

}

}

else if (LOWORD(wParam) == ID\_PROCESS\_FILE) {

// Вызываем асинхронную функцию для обработки файла

std::wstring filePath;

if (OpenFile(filePath)) {

std::thread(ProcessFileAsync, filePath).detach();

}

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

WNDCLASSEX wc = { 0 };

wc.cbSize = sizeof(WNDCLASSEX);

wc.style = 0;

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wc.lpszMenuName = NULL;

wc.lpszClassName = \_T("TextFileReader");

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (!RegisterClassEx(&wc)) {

MessageBox(NULL, \_T("Failed to register the window class"), \_T("Error"), MB\_ICONERROR);

return -1;

}

HWND hWnd = CreateWindow(\_T("TextFileReader"), \_T("Text File Reader"), WS\_OVERLAPPEDWINDOW, 100, 100, 640, 480, NULL, NULL, hInstance, NULL);

if (!hWnd) {

MessageBox(NULL, \_T("Failed to create the window"), \_T("Error"), MB\_ICONERROR);

return -1;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}