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

ОТЧЁТ

к лабораторной работе №1

на тему

Основы программирования в Win32 API. Оконное приложение Win32 с минимальной достаточной функциональностью. Обработка основных оконных сообщений

Выполнил: студент группы 153503

Степанов Владислав Николаевич

Проверил: Гриценко Никита Юрьевич

Минск 2023

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_gjdgxs)

[2 Краткие теоретические сведения 4](#_30j0zll)

3 [Результаты выполнения лабораторной работы 6](#_1fob9te)

[Выводы 8](#_3znysh7)

[Список использованных источников 9](#_2et92p0)

[Приложение А (обязательное) Листинг кода 10](#_tyjcwt)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, использования виджетов и обработки оконных сообщений. Реализовать вышеупомянутые требования на примере текстового редактора, которой обрабатывает основные оконные сообщения через функцию WndProc. Основные сообщения, такие как WM\_CREATE, WM\_SIZE, WM\_DESRTOY, WM\_COMMAND, и WM\_CLOSE, обрабатываются для обеспечения работоспособности текстового редактора, открытия/закрытия файлов, использования меню.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Application Programming Interface) представляет собой мощный набор инструментов и функций, предоставляемых операционной системой Windows. Этот интерфейс позволяет разработчикам создавать приложения, взаимодействовать с операционной системой и использовать ее возможности для разнообразных задач.

Win32 API включает в себя обширный набор функций, предоставляющих доступ к различным аспектам операционной системы Windows. С их помощью можно создавать оконные приложения, работать с файлами, управлять ресурсами компьютера и обрабатывать события. На практике это означает, что разработчики могут создавать приложения, которые взаимодействуют с оконным интерфейсом, выполняют сложные вычисления и манипулируют данными.

В рамках данной лабораторной работы мы использовали следующие теоретические аспекты и концепции:

- Этот интерфейс предоставляет набор функций и инструментов, с помощью которых разработчики могут создавать приложения для операционной системы Windows. Он предоставляет доступ к различным функциональным возможностям Windows.

- В Windows-приложениях обработка событий и сообщений играет ключевую роль. Функции обработки сообщений, такие как SoftwareMainProcedure, позволяют реагировать на события, происходящие в приложении, такие как нажатия кнопок и выбор пунктов меню.

- Для создания графического интерфейса приложения мы использовали стандартные элементы управления Windows, такие как окна, кнопки и текстовые поля. Win32 API функции позволили создать и настроить эти элементы.

- Для выполнения математических операций, таких как вычисления и работы с числами, мы использовали функции стандартной библиотеки и операции с числами с плавающей точкой.

- С помощью функции CreateWindow мы создали оконное приложение с заданными характеристиками. Кроме того, мы использовали функции для работы с элементами управления, такими как кнопки и текстовые поля.

- Для создания и обработки меню мы воспользовались функцией CreateMenu и связанными функциями для добавления пунктов меню и обработки выбора команд.

- Для обработки ошибок и вывода сообщений об ошибках использовались функции, такие как MessageBoxA.

- Графический интерфейс приложения разрабатывался с учетом стандартных принципов пользовательского интерфейса, включая расположение элементов и читаемость текста.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был разработан текстовый редактор. Он предоставляет возможность пользователю открывать редактор. (Рисунок 1).

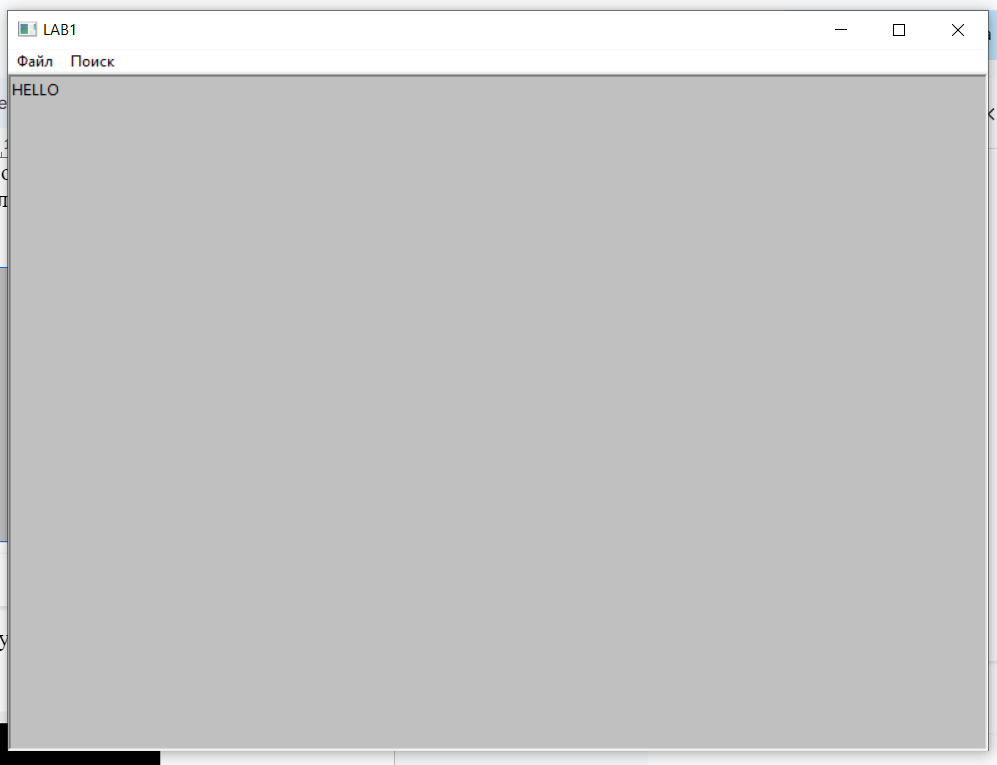


Рисунок 1 - Главное окно

Пользователь может писать текст (Рисунок 2).

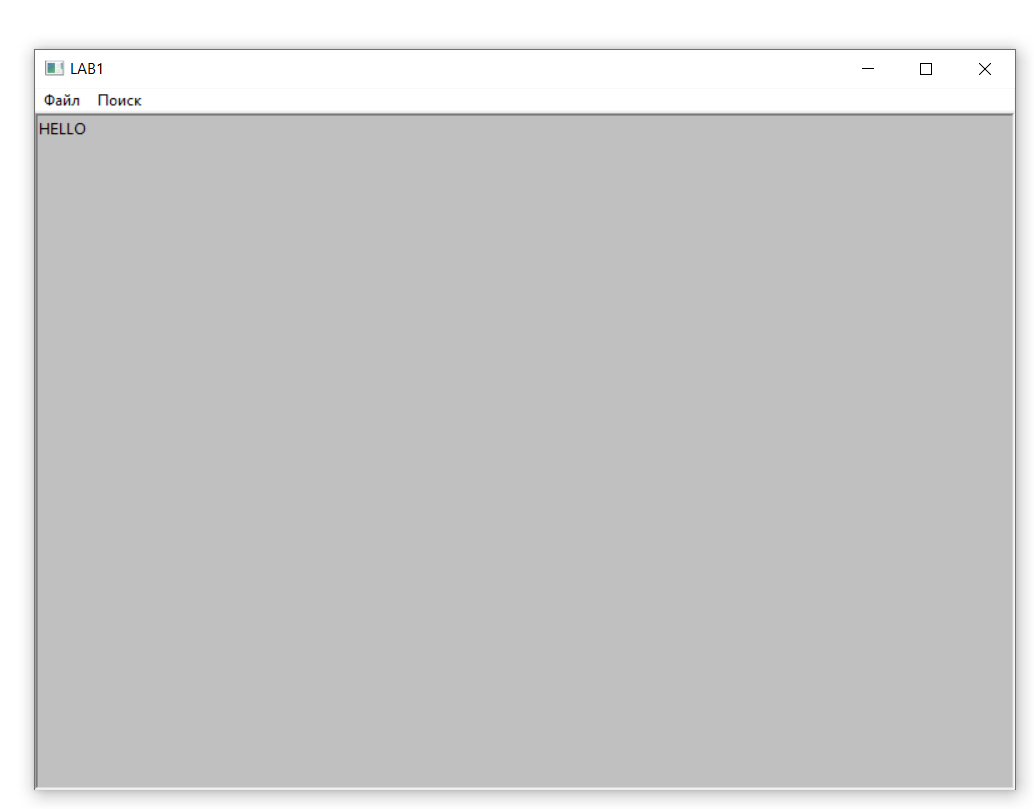


Рисунок 2 ─ Редактирование текста

Также есть система пользовательского меню с возможностью открывать/сохранять файлы, в том числе создавать новые, (Рисунок 3).

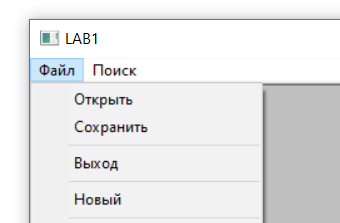


Рисунок 3 – Меню

## ВЫВОДЫ

В процессе выполнения данной лабораторной работы был создан текстовый редактор с использованием Win32 API. Проект включает в себя разработку графического интерфейса приложения, обработку основных оконных сообщений, а также добавление дополнительных функций, таких как возможность изменения цветовой схемы и шрифта. Получившееся приложение предоставляет удобный инструмент для редактирования текстовых файлов, и оно может быть использовано как начинающими пользователями, так и более опытными пользователями для управления и редактирования текстовой информации.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Учимся работать с файлами [Электронный ресурс]. ─ Режим доступа: https://eax.me/winapi-files.

[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 Source.cpp

#include <windows.h>

#include <richedit.h>

#include <string>

#include "Richedit.h"

#include "commctrl.h"

#include "resource.h"

#define RICHEDIT\_CLASS L"RICHEDIT50W"

#define IDM\_OPEN 1001

#define IDM\_SAVE 1002

#define IDM\_EXIT 1003

#define IDM\_NEW 1004

#define IDC\_CLOSE\_BUTTON 1005

#define IDM\_SEARCH 1006

#define IDM\_SEARCH\_IN\_SEARCH\_WINDOW 1008

#define IDM\_CHANGE\_THEME 1009

#define IDM\_CHANGE\_FONT 1010

#define HOTKEY\_ID 1

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

LRESULT CALLBACK SearchWindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

static HWND hEdit = nullptr;

static HWND hSearchEdit = nullptr;

static int searchStartPosition = 0;

static HWND hSearchWindow = nullptr;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR lpCmdLine, int nCmdShow)

{

if (!RegisterHotKey(NULL, HOTKEY\_ID, MOD\_CONTROL, 'F'))

{

MessageBox(NULL, L"Не удалось зарегистрировать горячую клавишу!", L"Error", MB\_ICONERROR | MB\_OK);

return 1;

}

MSG msg{};

HWND hWnd{};

WNDCLASS wc{ sizeof(WNDCLASS) };

wc.lpfnWndProc = WndProc;

wc.hInstance = hInstance;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = reinterpret\_cast<HBRUSH>(GetStockObject(LTGRAY\_BRUSH));

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

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

wc.hInstance = hInstance;

wc.lpszClassName = L"LAB1!";

wc.lpszMenuName = nullptr;

wc.style = CS\_VREDRAW | CS\_HREDRAW;

WNDCLASS wcSearchWindow{ sizeof(WNDCLASS) };

wcSearchWindow.lpfnWndProc = SearchWindowProc;

wcSearchWindow.hInstance = hInstance;

wcSearchWindow.hbrBackground = reinterpret\_cast<HBRUSH>(GetStockObject(LTGRAY\_BRUSH));

wcSearchWindow.lpszClassName = L"SearchWindowClass";

if (!RegisterClass(&wcSearchWindow) || !RegisterClassW(&wc)) {

return EXIT\_FAILURE;

}

if (hWnd = CreateWindow(wc.lpszClassName, L"LAB1", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, nullptr, nullptr, wc.hInstance, nullptr)) {

HMENU hMenu = CreateMenu();

HMENU hFileMenu = CreateMenu();

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hFileMenu, L"Файл");

AppendMenu(hFileMenu, MF\_STRING, IDM\_OPEN, L"Открыть");

AppendMenu(hFileMenu, MF\_STRING, IDM\_SAVE, L"Сохранить");

AppendMenu(hFileMenu, MF\_SEPARATOR, 0, NULL);

AppendMenu(hFileMenu, MF\_STRING, IDM\_EXIT, L"Выход");

AppendMenu(hFileMenu, MF\_SEPARATOR, 0, NULL);

AppendMenu(hFileMenu, MF\_STRING, IDM\_NEW, L"Новый");

AppendMenu(hFileMenu, MF\_SEPARATOR, 0, NULL);

AppendMenu(hFileMenu, MF\_STRING, IDM\_CHANGE\_THEME, L"Поменять тему");

AppendMenu(hFileMenu, MF\_STRING, IDM\_CHANGE\_FONT, L"Изменить шрифт");

AppendMenu(hMenu, MF\_STRING, IDM\_SEARCH, L"Поиск");

SetMenu(hWnd, hMenu);

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return EXIT\_SUCCESS;

}

void ResizeEditControl(HWND hWnd, HWND hEdit)

{

RECT clientRect;

GetClientRect(hWnd, &clientRect);

SetWindowPos(hEdit, nullptr, 0, 0, clientRect.right, clientRect.bottom, SWP\_NOZORDER);

}

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

{

static wchar\_t currentFileName[MAX\_PATH] = L"";

switch (message)

{

case WM\_CREATE:

{

INITCOMMONCONTROLSEX icex;

icex.dwSize = sizeof(INITCOMMONCONTROLSEX);

icex.dwICC = ICC\_STANDARD\_CLASSES;

InitCommonControlsEx(&icex);

LoadLibrary(TEXT("Msftedit.dll"));

hEdit = CreateWindowEx(

WS\_EX\_CLIENTEDGE,

RICHEDIT\_CLASS,

L"",

WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | WS\_HSCROLL | ES\_MULTILINE | ES\_AUTOVSCROLL | ES\_AUTOHSCROLL | ES\_NOHIDESEL,

0, 0, 800, 600,

hWnd,

nullptr,

GetModuleHandle(nullptr),

nullptr

);

if (hEdit == nullptr)

{

MessageBox(hWnd, L"Не удалось создать элемент управления Edit.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

break;

case WM\_SIZE:

ResizeEditControl(hWnd, hEdit);

break

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDM\_CHANGE\_THEME:

OpenColorDialog();

break;

case IDM\_CHANGE\_FONT:

{

CHOOSEFONT cf = { sizeof(CHOOSEFONT) };

LOGFONT lf = { 0 };

cf.hwndOwner = hWnd;

cf.lpLogFont = &lf;

cf.Flags = CF\_SCREENFONTS | CF\_EFFECTS | CF\_INITTOLOGFONTSTRUCT;

if (ChooseFont(&cf))

{

HFONT hFont = CreateFontIndirect(&lf);

SendMessage(hEdit, WM\_SETFONT, (WPARAM)hFont, TRUE);

}

break;

}

case IDM\_OPEN:

{

OPENFILENAME ofn;

wchar\_t szFile[MAX\_PATH] = L"";

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFilter = L"C++ файлы (\*.cpp;\*.txt)\0\*.cpp;\*.txt\0Все файлы (\*.\*)\0\*.\*\0";

ofn.lpstrFile = szFile;

ofn.nMaxFile = MAX\_PATH;

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

if (GetOpenFileName(&ofn))

{

wcscpy\_s(currentFileName, MAX\_PATH, ofn.lpstrFile);

HANDLE hFile = CreateFile(ofn.lpstrFile, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, 0, NULL);

if (hFile != INVALID\_HANDLE\_VALUE)

{

DWORD dwFileSize = GetFileSize(hFile, NULL);

wchar\_t\* fileContent = (wchar\_t\*)malloc((dwFileSize + 1) \* sizeof(wchar\_t));

if (fileContent)

{

DWORD bytesRead;

ReadFile(hFile, fileContent, dwFileSize \* sizeof(wchar\_t), &bytesRead, NULL);

fileContent[dwFileSize / sizeof(wchar\_t)] = L'\0';

SetWindowText(hEdit, fileContent);

free(fileContent);

}

else

{

MessageBox(hWnd, L"Не удалось выделить память для содержимого файла.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

CloseHandle(hFile);

}

else

{

MessageBox(hWnd, L"Не удалось открыть выбранный файл.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

break;

}

case IDM\_SAVE:

{

HANDLE hFile = INVALID\_HANDLE\_VALUE;

if (currentFileName[0] != L'\0')

{

hFile = CreateFile(currentFileName, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

}

if (hFile == INVALID\_HANDLE\_VALUE)

{

OPENFILENAME ofn;

wchar\_t szFile[MAX\_PATH] = L"";

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFilter = L"C++ files (\*.cpp;\*.txt)\0\*.cpp;\*.txt\0All files (\*.\*)\0\*.\*\0";

ofn.lpstrFile = szFile;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_OVERWRITEPROMPT;

if (GetSaveFileName(&ofn))

{

hFile = CreateFile(ofn.lpstrFile, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile != INVALID\_HANDLE\_VALUE)

{

wcscpy\_s(currentFileName, MAX\_PATH, ofn.lpstrFile);

}

else

{

MessageBox(hWnd, L"Не удалось создать выбранный файл.", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

}

if (hFile != INVALID\_HANDLE\_VALUE)

{

int textLength = GetWindowTextLength(hEdit);

wchar\_t\* textBuffer = (wchar\_t\*)malloc((textLength + 1) \* sizeof(wchar\_t));

if (textBuffer)

{

GetWindowText(hEdit, textBuffer, textLength + 1);

DWORD bytesWritten;

WriteFile(hFile, textBuffer, textLength \* sizeof(wchar\_t), &bytesWritten, NULL);

free(textBuffer);

CloseHandle(hFile);

}

else

{

MessageBox(hWnd, L"Не удалось выделить память для текста.", L"Ошибка", MB\_OK | MB\_ICONERROR);

CloseHandle(hFile);

}

}

break;

}

case IDM\_NEW:

{

if (MessageBox(hWnd, L"Вы уверены, что хотите создать новый документ? Все несохраненные изменения будут потеряны.", L"Подтвердите действие", MB\_YESNO | MB\_ICONQUESTION) == IDYES)

{

SetWindowText(hEdit, L"");

currentFileName[0] = L'\0';

}

break;

}

case WM\_CLOSE:

CloseWindow(hWnd);

break;

case IDM\_EXIT:

PostQuitMessage(0);

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}