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 ФУНКЦИОНАЛЬНОЙ ДОСТАТОЧНОСТЬЮ. ОБРАБОТКА ОСНОВНЫХ ОКОННЫХ СООБЩЕНИЙ**

Выполнил:

студент группы 153503

Скворцов А. В.

Проверил:

Гриценко Н. Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[3 Результаты выполнения лабораторной работы 5](#_Toc146728179)

[Выводы 7](#_Toc146728180)

[Список использованных источников 8](#_Toc146728181)

[Приложение А (обязательное) Листинг кода 9](#_Toc146728182)

## 1 ЦЕЛЬ РАБОТЫ

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 (как базовых, так и пользовательских). Реализовать вышеупомянутые требования на примере приложения для чтения и редактирования текстовых документов с возможностью выделения и копирования текста в буфер обмена, которое обрабатывает основные оконные сообщения через функцию WindowProc. Основные сообщения, такие как WM\_PAINT, WM\_CREATE, WM\_SIZE, WM\_COMMAND, и WM\_CLOSE, обрабатываются для управления отрисовкой приложения, обработки основных событий (открытие, сохранение файла), закрытия окна.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Интерфейс прикладного программирования Win32 API представляет собой набор функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы Windows. Этот API предоставляет средства для управления и взаимодействия с операционной системой, а также для создания приложений, которые работают в среде Windows.

Win32 API включает в себя более 3000 функций, которые позволяют реализовать различные сервисы операционной системы. Эти функции могут быть вызваны из приложений для выполнения разнообразных задач, таких как создание процессов и обработка оконных сообщений.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

­– Win32 API (Application Programming Interface): Win32 API — это набор функций и подпрограмм, предоставляемых операционной системой Windows для взаимодействия с приложениями. Он предоставляет доступ к различным функциональным возможностям Windows, таким как создание окон, обработка сообщений, работа с файлами и др.

– Обработка событий и сообщений: в Windows-приложениях взаимодействие с пользователем осуществляется через обработку событий и сообщений.

– Создание пользовательского интерфейса: для создания графического интерфейса приложения были использованы стандартные элементы управления Windows. Эти элементы были созданы и настроены с использованием Win32 API функций.

– Работа с окнами и элементами управления: для создания оконного приложения была использована функция CreateWindowEx, которая создает окно с указанными характеристиками, такими как размер, положение и стиль.

– Обработка меню: для создания меню и обработки команд меню использовалась функция CreateMenu и связанные с ней функции для добавления пунктов меню и обработки выбора команд.

– Интерфейс пользователя: графический интерфейс приложения разработан с учетом стандартных принципов пользовательского интерфейса, таких как расположение элементов и размещение текста.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был разработан текстовый редактор, напоминающий классическое приложение «Блокнот». Приложение позволяет пользователю создать текстовый файл и печатать текст с клавиатуры (рисунок 1).

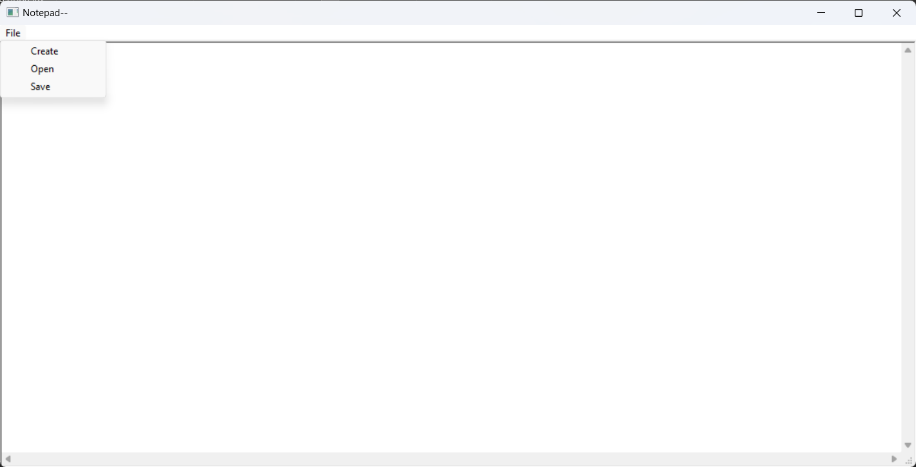


Рисунок 1 - Главный экран

Пользователь может открыть файл, чтобы просмотреть его содержимое и при необходимости отредактировать (рисунок 2).

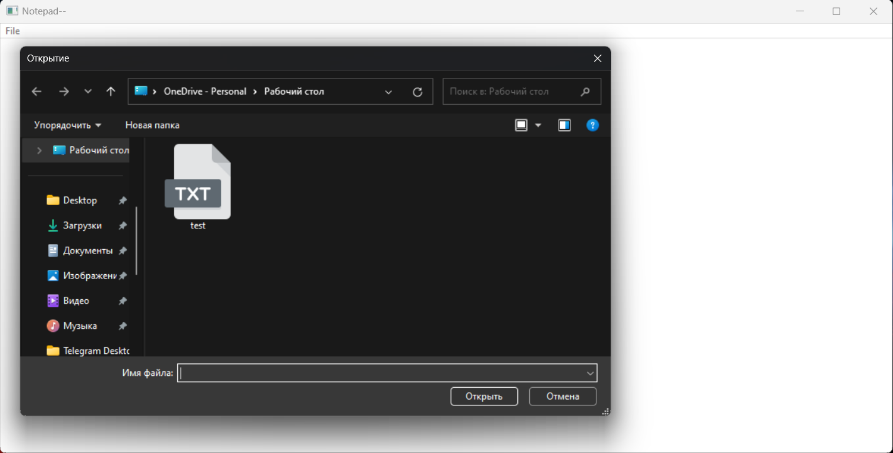


Рисунок 2 ─ Открытие файла

Также пользователь имеет возможность выделить весь текст с помощью сочетания клавиш Ctrl + A, скопировать в буфер обмена, вставить (рисунок 3).

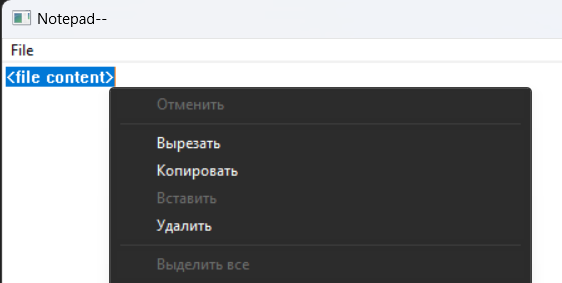


Рисунок 3 – Копирование и вставка

Пользователю предоставляется возможность сохранить свой файл (рисунок 4).

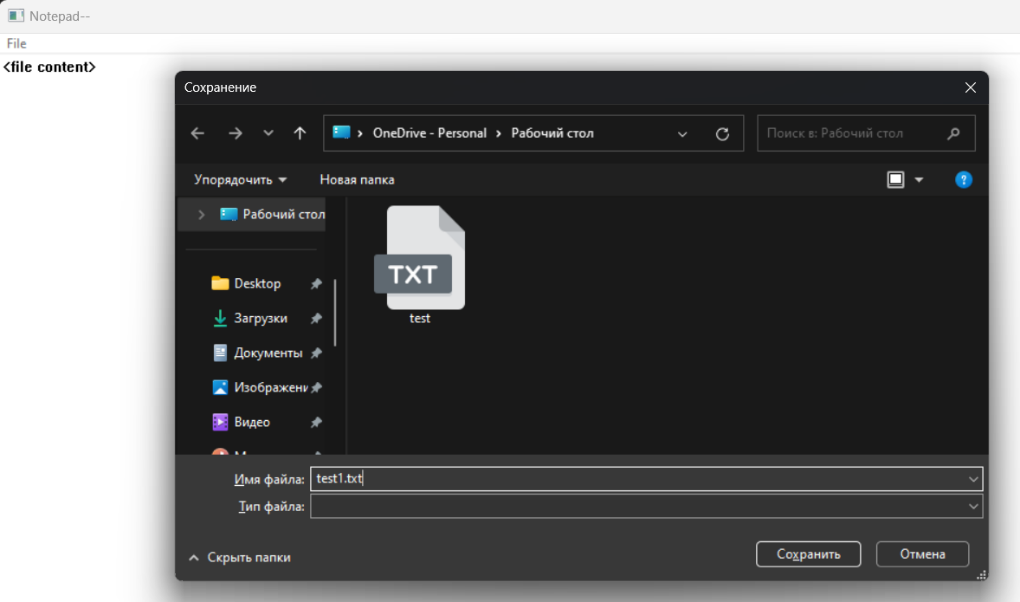


Рисунок 4 – Сохранение файла

## ВЫВОДЫ

В результате выполнения работы было создано оконное приложение, способное открывать, редактировать и сохранять текстовые документы, а также осуществлять выделение и копирование текста в буфер обмена. В ходе обработки основных оконных сообщений были реализованы необходимые функции для работы с текстовыми документами. Была создана область для отображения текста с возможностью прокрутки, а также добавлены элементы управления для выполнения операций над текстовыми файлами. Таким образом, лабораторная работа позволила ознакомиться с основами программирования в Win32 API и создать оконное приложение с базовой функциональностью, отражающей основные принципы работы с окнами и обработкой сообщений.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. Win32 API. Разработка приложений для Windows. ─ СПб: Питер, 2008. ─ 592 с.: ил.
2. Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 19.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

Листинг 1 – Файл main.cpp

#ifndef UNICODE

#define UNICODE

#endif

#define ID\_FILE\_CREATE 9001

#define ID\_FILE\_OPEN 9002

#define ID\_FILE\_SAVE 9003

#define VK\_A 0x41

// Макросы для распознавания конкретной команды из MenuBar'а

#include <windows.h>

#include <shobjidl.h>

#include <iostream>

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

//LRESULT CALLBACK HotKeyProc(int nCode, WPARAM wParam, LPARAM lParam);

void OpenFile(HWND hwnd);

void SaveFile(HWND hwnd);

HWND hWndEdit = NULL;

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow) {

const wchar\_t CLASS\_NAME[] = L"Sample Window Class";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

HWND hwnd = CreateWindowEx(

0,

CLASS\_NAME,

L"Notepad--",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL,

NULL,

hInstance,

NULL

);

if (hwnd == NULL) {

return 0;

}

ShowWindow(hwnd, nCmdShow);

MSG msg = { };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

switch (uMsg) {

case WM\_DESTROY:

{

PostQuitMessage(0);

return 0;

}

case WM\_CREATE:

{

HRESULT hr = CoInitializeEx(NULL, COINIT\_APARTMENTTHREADED | COINIT\_DISABLE\_OLE1DDE);

if (!SUCCEEDED(hr)) {

DestroyWindow(hwnd);

return 0;

}

HMENU hMenu, hSubMenu;

// Создание MenuBar

hMenu = CreateMenu();

hSubMenu = CreatePopupMenu();

AppendMenu(hSubMenu, MF\_STRING, ID\_FILE\_CREATE, L"&Create");

AppendMenu(hSubMenu, MF\_STRING, ID\_FILE\_OPEN, L"&Open");

AppendMenu(hSubMenu, MF\_STRING, ID\_FILE\_SAVE, L"&Save");

AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hSubMenu, L"&File");

SetMenu(hwnd, hMenu);

hWndEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, TEXT("Edit"), NULL,

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

0, 0, 780, 560, hwnd, NULL,

(HINSTANCE)GetWindowLong(hwnd, GWL\_HINSTANCE),

NULL);

RegisterHotKey(hwnd, 1, MOD\_CONTROL, VK\_A);

return 0;

}

case WM\_COMMAND:

{

switch (LOWORD(wParam)) {

// Обработка сообщений из нажатых кнопок в меню

case ID\_FILE\_CREATE:

SetWindowTextA(hWndEdit, "");

break;

case ID\_FILE\_OPEN:

OpenFile(hwnd);

break;

case ID\_FILE\_SAVE:

SaveFile(hwnd);

break;

default:

break;

}

return 0;

}

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

FillRect(hdc, &ps.rcPaint, (HBRUSH)(COLOR\_WINDOW + 1));

EndPaint(hwnd, &ps);

return 0;

}

case WM\_SIZE:

{

// Растягивание окна текстового редактора совместно с растягиванием основного окна

MoveWindow(hWndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), false);

return 0;

}

case WM\_HOTKEY:

{

if (wParam == 1) {

SendMessage(hWndEdit, EM\_SETSEL, 0, GetWindowTextLength(hWndEdit));

}

break;

}

case WM\_CLOSE:

{

CoUninitialize();

DestroyWindow(hwnd);

return 0;

}

return 0;

}

return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

void OpenFile(HWND hwnd) {

IFileOpenDialog\* pFileOpen;

HRESULT hr = CoCreateInstance(CLSID\_FileOpenDialog, NULL, CLSCTX\_ALL,

IID\_IFileOpenDialog, reinterpret\_cast<void\*\*>(&pFileOpen));

if (SUCCEEDED(hr)) {

hr = pFileOpen->Show(NULL);

if (SUCCEEDED(hr)) {

IShellItem\* pItem;

hr = pFileOpen->GetResult(&pItem);

if (SUCCEEDED(hr)) {

PWSTR pszFilePath;

hr = pItem->GetDisplayName(SIGDN\_FILESYSPATH, &pszFilePath);

if (SUCCEEDED(hr)) {

HANDLE hFile = CreateFile(pszFilePath, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

DWORD fileSize = GetFileSize(hFile, NULL);

LPSTR buffer = (LPSTR)GlobalAlloc(GPTR, fileSize + 1);

DWORD read;

if (ReadFile(hFile, buffer, fileSize, &read, NULL)) {

SetWindowTextA(hWndEdit, buffer);

} else {

MessageBoxA(hwnd, "Cannot read file", "Error", MB\_OK);

}

GlobalFree((HGLOBAL)buffer);

ShowWindow(hWndEdit, SW\_SHOW);

SetFocus(hWndEdit);

CloseHandle(hFile);

}

pItem->Release();

}

}

pFileOpen->Release();

}

}

void SaveFile(HWND hwnd) {

IFileSaveDialog\* pFileSave;

HRESULT hr = CoCreateInstance(CLSID\_FileSaveDialog, NULL, CLSCTX\_ALL,

IID\_IFileSaveDialog, reinterpret\_cast<void\*\*>(&pFileSave));

if (SUCCEEDED(hr)) {

hr = pFileSave->Show(NULL);

if (SUCCEEDED(hr)) {

IShellItem\* pItem;

hr = pFileSave->GetResult(&pItem);

if (SUCCEEDED(hr)) {

PWSTR pszFilePath;

hr = pItem->GetDisplayName(SIGDN\_FILESYSPATH, &pszFilePath);

if (SUCCEEDED(hr)) {

HANDLE hFile = CreateFile(pszFilePath, GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_NEW, FILE\_ATTRIBUTE\_NORMAL, NULL);

DWORD fileSize = GetWindowTextLength(hWndEdit) + 1;

LPSTR buffer = (LPSTR)GlobalAlloc(GPTR, fileSize);;

GetWindowTextA(hWndEdit, buffer, fileSize);

DWORD wroted;

if (WriteFile(hFile, (void\*)buffer, fileSize, &wroted, NULL)) {

MessageBox(hwnd, L"File successfully saved", L"Saved", MB\_OK);

CloseHandle(hFile);

}

GlobalFree((HGLOBAL)buffer);

}

}

}

}

}