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 № 1

на тему «Основы программирования в Win 32 API. Оконное приложение Win 32 с минимальной достаточной функциональностью. Обработка основных оконных сообщений»

Выполнил:

студент гр. 153504

Михалевич М.П.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цели работы 3](#_Toc146655820)

[2 Краткие теоретические сведения 4](#_Toc146655821)

[3 Результаты выполнения лабораторной работы 5](#_Toc146655822)

[ВЫВОД 7](#_Toc146655823)

[ПРИЛОЖЕНИЕ А 8](#_Toc146655824)

## 1 Цели работы

Целью данной работы является освоение основ программирования с использованием Win32 API и создание оконного приложения с минимальной, но достаточной функциональностью. Это включает в себя изучение обработки основных оконных сообщений, создание графического интерфейса и разработку приложения, предназначенного для чтения и редактирования текстовых документов, а также способного осуществлять выделение и копирование текста в буфер обмена.

## 2 Краткие теоретические сведения

Win32 API представляет собой набор функций и процедур, предоставляемых операционной системой Windows для создания приложений. Этот интерфейс обеспечивает взаимодействие приложения с операционной системой и аппаратным обеспечением компьютера. Он позволяет разработчикам создавать оконные приложения для Windows, обрабатывать события, взаимодействовать с ресурсами системы (например, файлами, памятью) и многие другие задачи.

В Win32 API, одной из фундаментальных концепций является регистрация класса окна. Это действие определяет, как окно будет выглядеть и какие свойства оно будет иметь.

Другой важной частью программы является создание окна. С использованием функции CreateWindow, в программе создается окно с заданными параметрами, такими как стиль, размер и положение на экране. Это окно становится основным элементом интерфейса приложения.

Для обработки событий и взаимодействия с пользователем код содержит функцию WndProc, которая обрабатывает различные сообщения, поступающие от операционной системы.

Для взаимодействия с пользователем и обеспечения функциональности приложения используются элементы управления, такие как кнопки и текстовые поля, создаваемые с помощью функции CreateWindow и добавляемые в окно.

Дополнительно, код включает функции для работы с файлами, включая открытие, сохранение и редактирование текстовых файлов. Это позволяет пользователям выполнять операции с файлами непосредственно из приложения.

## 3 Результаты выполнения лабораторной работы

В ходе выполнения лабораторной работы было разработано приложение для чтения и редактирования текстовых документов. При запуске программы пользователь может воспользоваться меню, при помощи которого может открыть и сохранить текстовый файл каждое, а также выйти из программы. Меню показано на рисунке 3.1.

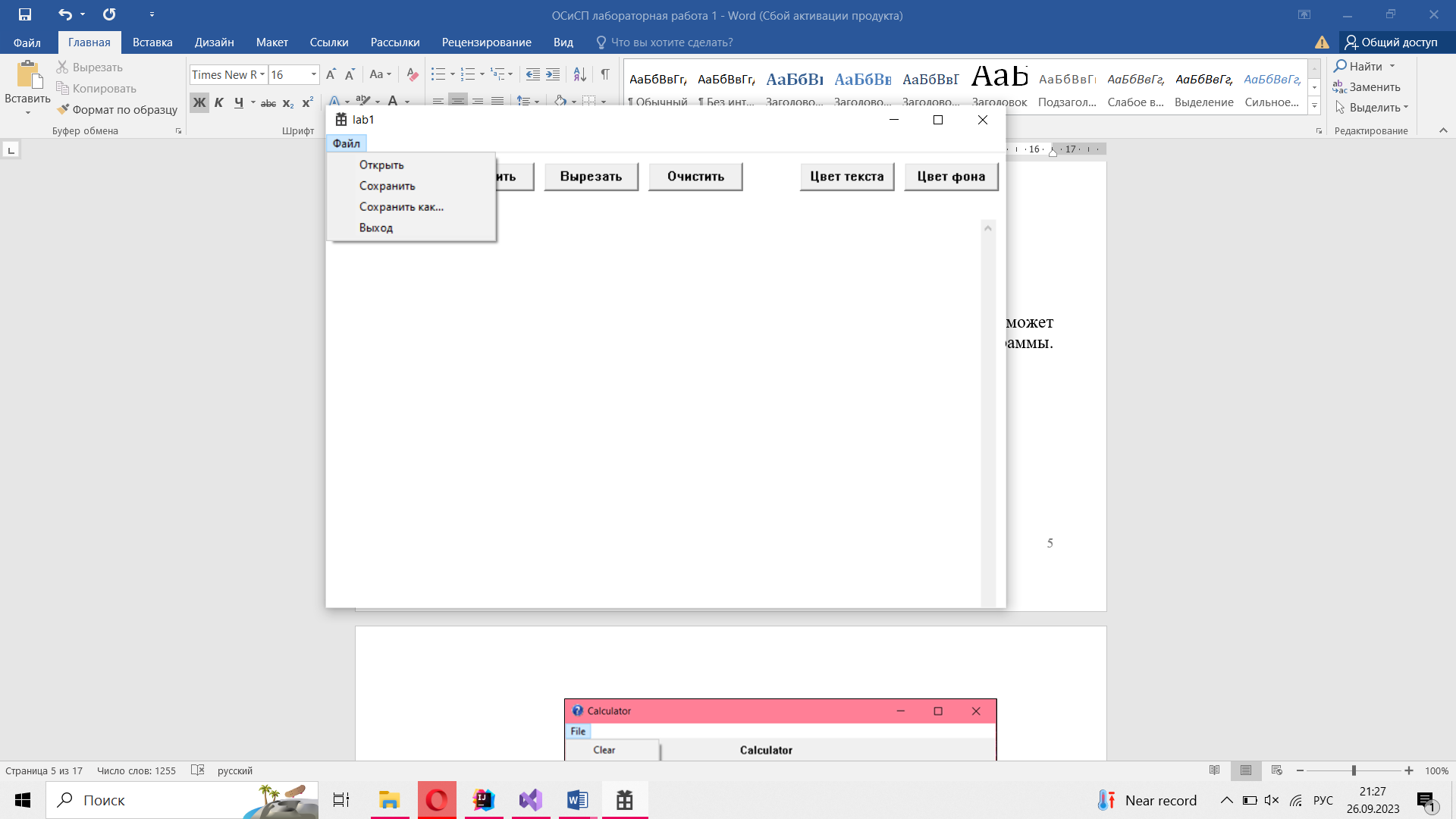


Рисунок 3.1 – Меню

Дополнительно, пользователь может выделять необходимый текст, копировать его в буфер обмена для последующей вставки в другие приложения, а также выполнять вырезание выделенного текста и очистку всего текстового поля при необходимости. Работа программы показана на рисунке 3.2.

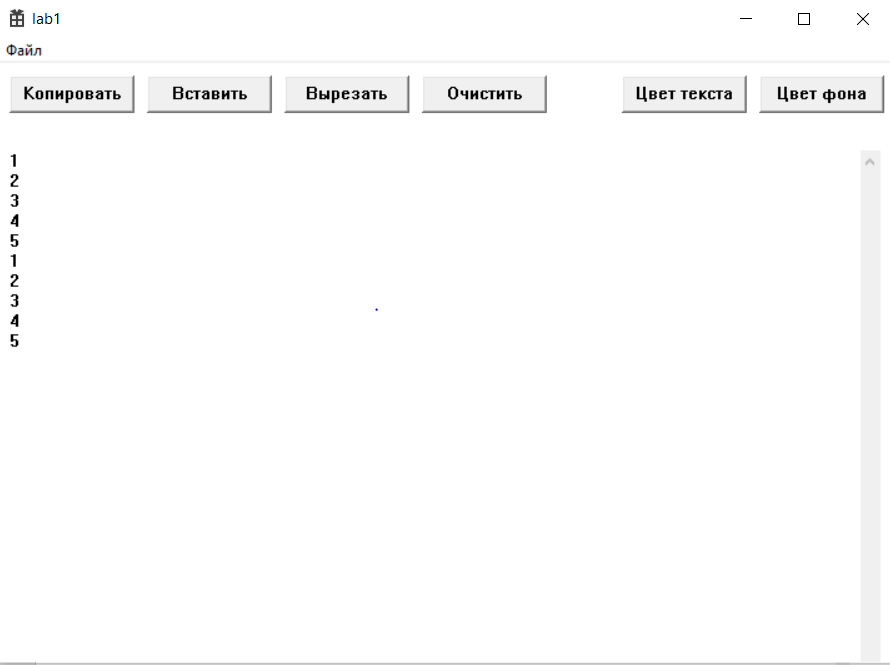


Рисунок 3.2 – Результат работы программы

## ВЫВОД

В результате выполнения лабораторной работы по изучению основ программирования в Win32 API были освоены ключевые понятия и компоненты этого интерфейса. Созданное оконное приложение, обладающее минимальной, но достаточной функциональностью и способное обрабатывать основные оконные сообщения. Было разработано приложение для чтения и редактирования текстовых документов с возможностью выделения и копирования текста в буфер обмена.

## ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг кода**

#include "framework.h"

#include "helpers.h"

#include "lab1.h"

#include <commdlg.h>

#include <fstream>

#define MAX\_LOADSTRING 100

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

HWND hEdit; //

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB1));

MSG msg;

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

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

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB1));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB1);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

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

{

static WCHAR fileName[MAX\_PATH] = L"";

static COLORREF rgbCurrentText = RGB(0, 0, 0);

switch (message)

{

case WM\_CREATE: // Сообщение о создании окна

{

CreateWindow(L"BUTTON", L"Копировать", WS\_CHILD | WS\_VISIBLE, 10, 10, 100, 30, hWnd, (HMENU)COPY\_BUTTON, hInst, NULL);

CreateWindow(L"BUTTON", L"Вставить", WS\_CHILD | WS\_VISIBLE, 120, 10, 100, 30, hWnd, (HMENU)PASTE\_BUTTON, hInst, NULL);

CreateWindow(L"BUTTON", L"Вырезать", WS\_CHILD | WS\_VISIBLE, 230, 10, 100, 30, hWnd, (HMENU)CUT\_BUTTON, hInst, NULL);

CreateWindow(L"BUTTON", L"Очистить", WS\_CHILD | WS\_VISIBLE, 340, 10, 100, 30, hWnd, (HMENU)CLEAN\_BUTTON, hInst, NULL);

hEdit = CreateWindow(

L"EDIT",

L"",

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

10,

30,

1000,

500,

hWnd,

NULL,

hInst,

NULL

);

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

EndPaint(hWnd, &ps);

}

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case EXIT:

DestroyWindow(hWnd);

break;

case OPEN\_BUTTON:

{

std::wstring openedFileName = HandleOpen(hWnd, hEdit);

wcscpy\_s(fileName, openedFileName.c\_str());

break;

}

case SAVE\_BUTTON:

{

if (wcslen(fileName) > 0)

{

WCHAR szNewContent[10000];

GetWindowText(hEdit, szNewContent, 10000);

std::wstring FilteredContent = FilterContent(szNewContent);

SaveToFile(hWnd, fileName, FilteredContent.c\_str());

}

else

{

HandleSaveAs(hWnd, hEdit);

}

}

break;

case SAVE\_AS\_BUTTON:

HandleSaveAs(hWnd, hEdit);

break;

case COPY\_BUTTON:

SendMessage(hEdit, WM\_COPY, 0, 0);

break;

case PASTE\_BUTTON:

SendMessage(hEdit, WM\_PASTE, 0, 0);

break;

case CUT\_BUTTON:

SendMessage(hEdit, WM\_CUT, 0, 0);

break;

case CLEAN\_BUTTON:

SetWindowText(hEdit, L"");

break;

default:

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

}

}

break;

case WM\_KEYDOWN:

{

if (GetKeyState(VK\_CONTROL) & 0x8000)

{

switch (wParam)

{

case 'S': // Ctrl + S

PostMessage(hWnd, WM\_COMMAND, MAKEWPARAM(SAVE\_BUTTON, 0), 0);

return 0;

case 'C': // Ctrl + C

SendMessage(hEdit, WM\_COPY, 0, 0);

return 0;

case 'V': // Ctrl + V

SendMessage(hEdit, WM\_PASTE, 0, 0);

return 0;

case 'X': // Ctrl + X

SendMessage(hEdit, WM\_CUT, 0, 0);

return 0;

}

}

}

break;

case WM\_SIZE:

{

RECT rc;

GetClientRect(hWnd, &rc);

int windowWidth = rc.right - rc.left;

int windowHeight = rc.bottom - rc.top;

int editHeight = windowHeight - 50;

SetWindowPos(

hEdit,

NULL,

10,

70,

windowWidth - 20,

editHeight,

SWP\_NOZORDER

);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

#include "helpers.h"

void SaveToFile(HWND hWnd, const WCHAR\* fileName, const WCHAR\* content)

{

std::wofstream file(fileName);

file << content;

file.close();

MessageBox(hWnd, L"Файл успешно сохранен!", L"Уведомление", MB\_OK);

}

std::wstring FilterContent(const WCHAR\* content)

{

std::wstring FilteredContent;

for (int i = 0; content[i] != L'\0'; i++) {

if (content[i] != L'\n') {

FilteredContent += content[i];

}

}

return FilteredContent;

}

void HandleSaveAs(HWND hWnd, HWND hEdit)

{

WCHAR fileName[MAX\_PATH] = L"";

WCHAR szNewContent[10000];

GetWindowText(hEdit, szNewContent, sizeof(szNewContent) / sizeof(WCHAR));

std::wstring FilteredContent = FilterContent(szNewContent);

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFilter = L"Текстовые файлы (\*.txt)\0\*.txt\0";

ofn.lpstrFile = fileName;

ofn.nMaxFile = sizeof(fileName) / sizeof(WCHAR);

ofn.Flags = OFN\_EXPLORER | OFN\_PATHMUSTEXIST | OFN\_HIDEREADONLY;

if (GetSaveFileName(&ofn))

{

SaveToFile(hWnd, fileName, FilteredContent.c\_str());

}

}

std::wstring HandleOpen(HWND hWnd, HWND hEdit)

{

WCHAR fileName[MAX\_PATH] = L"";

WCHAR fileContent[10000] = L"";

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFilter = L"Текстовые файлы (\*.txt)\0\*.txt\0";

ofn.lpstrFile = fileName;

ofn.nMaxFile = sizeof(fileName) / sizeof(WCHAR);

ofn.Flags = OFN\_EXPLORER | OFN\_FILEMUSTEXIST | OFN\_HIDEREADONLY;

if (GetOpenFileName(&ofn))

{

std::wifstream file(fileName);

if (file.is\_open())

{

file.get(fileContent, sizeof(fileContent) / sizeof(WCHAR), L'\0');

file.close();

WCHAR modifiedContent[10000] = L"";

int j = 0;

for (int i = 0; fileContent[i] != L'\0'; i++) {

if (fileContent[i] == L'\n') {

modifiedContent[j++] = '\r';

}

modifiedContent[j++] = fileContent[i];

if (fileContent[i] == L'\r') {

modifiedContent[j++] = '\n';

}

}

SetWindowText(hEdit, modifiedContent);

return fileName;

}

else

{

MessageBox(hWnd, L"Не удалось открыть файл!", L"Ошибка", MB\_OK);

}

}

}