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 Win32 API. Оконное приложение Win32 с минимальной достаточной функциональностью. Обработка основных оконных сообщений»

Выполнил:

студент гр. 153504

Скроба А.П.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146675523)

[2 Краткие теоретические сведения 4](#_Toc146675524)

[3 Результат выполнения программы 5](#_Toc146675525)

[Список использованных источников 9](#_Toc146675526)

[Приложение А 10](#_Toc146675527)

# 1 ЦЕЛЬ РАБОТЫ

Создание минимального оконного приложения Win32, предназначенного для чтения и редактирования текстовых документов. Основной задачей приложения является обеспечение удобства работы с текстом, включая возможность выделения и копирования текстовых фрагментов в буфер обмена.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

API (Application Programming Interface - интерфейс прикладных программ) - это множество функций, организованных, обычно, в виде DLL. Функции API позволяют организовать интерфейс между прикладной программой и средой, в которой работает эта программа. Вызов функций API позволяет программе получать доступ к ресурсам среды и управлять ее работой. Как правило, API задает стандарт взаимодействия среды и прикладной программы.

Win32 - это название интерфейса, ориентированного на 32-х разрядные приложения и реализованного на таких известных платформах как Windows 95, Windows 98, Windows NT, Windows CE. Функции, составляющие этот интерфейс, позволяют прикладной программе получать доступ к ресурсам операционной системы и управлять ее работой.

Все Windows-программы должны содержать специальную функцию, которая не используется в самой программе, но вызывается операционной системой. Эту функцию обычно называют функцией окна, или процедурой окна. Она вызывается Windows, когда системе необходимо передать сообщение в программу. Именно через нее осуществляется взаимодействие между программой и системой. Функция окна передает сообщение в своих аргументах. Согласно терминологии Windows те функции, которые вызываются системой, называются функциями обратного вызова.

Каждое окно в Windows-приложении характеризуется определенными атрибутами, называемыми классом окна. В традиционной программе класс окна должен быть определен и зарегистрирован прежде, чем будет создано окно. При регистрации необходимо сообщить Windows, какой вид должно иметь окно и какую функцию оно выполняет. В то же время регистрация класса окна еще не означает создания самого окна. Для этого требуется выполнить дополнительные действия.

В Windows поддерживаются два типа приложений: основанные на графическом интерфейсе GUI (graphical user interface) и консольные CUI (console user interface). У приложений первого типа внешний интерфейс чисто графический: создаются окна, меню, диалоговые окна и т. д. Почти все стандартные программы Windows и большинство пользовательских программ являются GUI-приложениями. Приложения консольного типа работают в текстовом режиме: они не формируют окна, не обрабатывают сообщения, но на экране тоже размещаются в окне и могут вызывать диалоговые окна.

# 3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результатом выполнения лабораторной работы является оконное приложение с интуитивно понятным пользовательским интерфейсом (рисунок 3.1). В меню присутствуют три кнопки – «Открыть», «Сохранить» и «Сохранить как…». Так же под полем для ввода и вывода текста расположена кнопка «Clear», которая полностью удаляет содержимое текста.

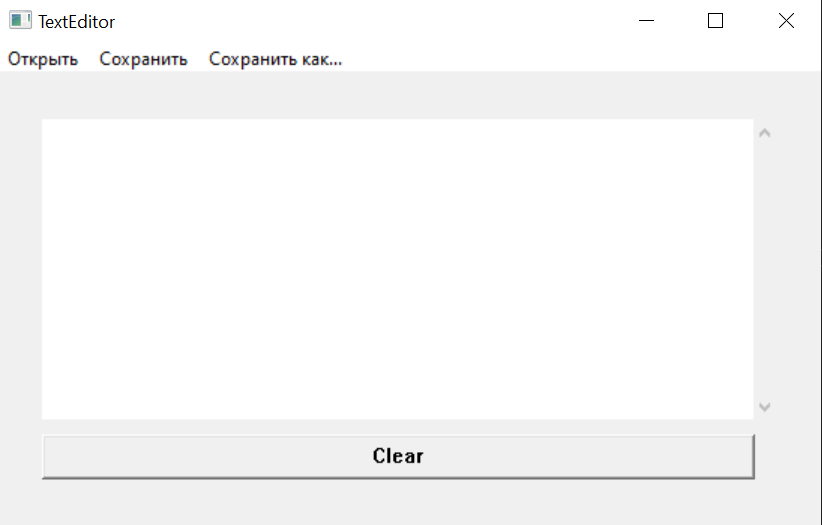
****

Рисунок 3.1 – Основной интерфейс приложения

При нажатии на кнопку «Открыть» открывается файловый менеджер, отображающий и позволяющий выбрать только файлы с расширением .txt (рисунок 3.2).

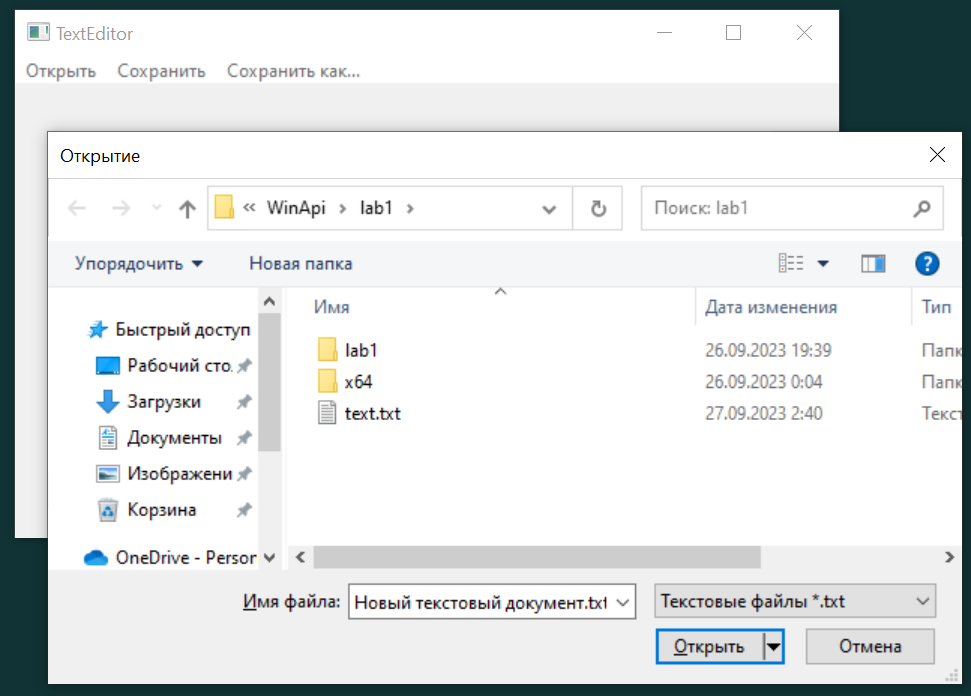


Рисунок 3.2 – Выбор файла для загрузки

После выбора файла в поле для текста появляется текст выбранного файла. В заголовке окна приложения отображается название файла и путь к выбранному файлу. Данный текст можно изменить вводом с клавиатуры или стереть все содержимое кнопкой «Clear» (рисунок 3.3).

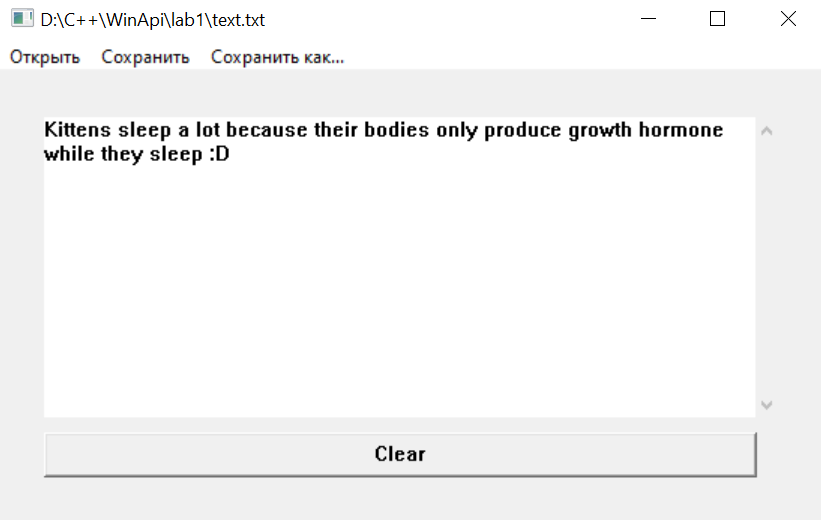
****

Рисунок 3.3 – Отображение содержимого выбранного файла

Кнопка «Сохранить» позволяет сохранить изменения в этот же файл, а кнопка «Сохранить как…» дает возможность выбрать путь и название сохраняемого файла. Стоит отметить, что если попытаться сохранить новый файл кнопкой «Сохранить», то откроется файловый менеджер, где можно будет указать имя нового файла и его расположение. Если при сохранении файла будет указано имя, которое уже существует в выбранной папке, то программа покажет диалоговое окно с предупреждением и предложит либо перезаписать файл, либо поменять название сохраняемого файла (рисунок 3.4).

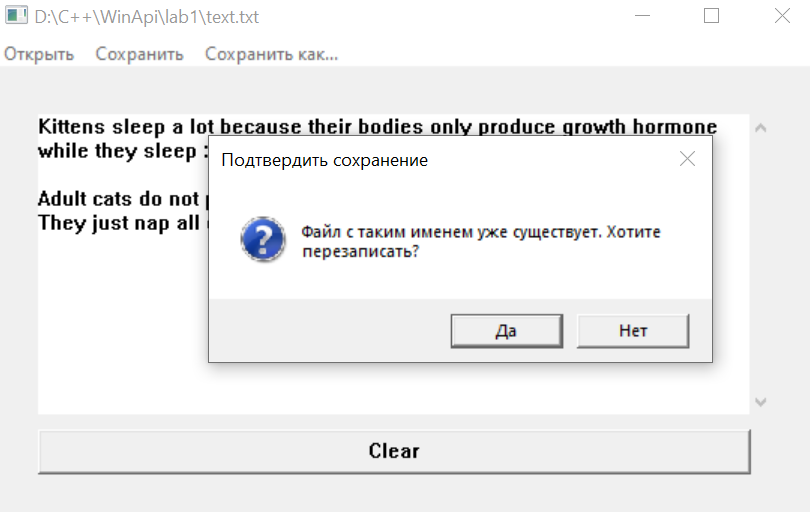


Рисунок 3.4 – Диалоговое окно с предложением перезаписи файла

После удачного сохранения файла, появится диалоговое окно с надписью об успешном сохранении файла (рисунок 3.5).

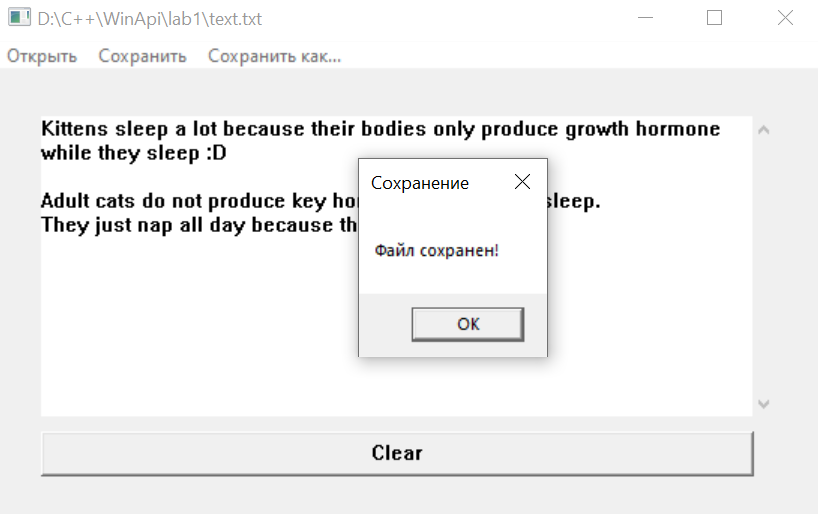
****

Рисунок 3.5 – Диалоговое окно об успешном сохранении файла

Так же в приложении присутсвует возможность выделения и копирования текста в буфер обмена (рисунок 3.6).

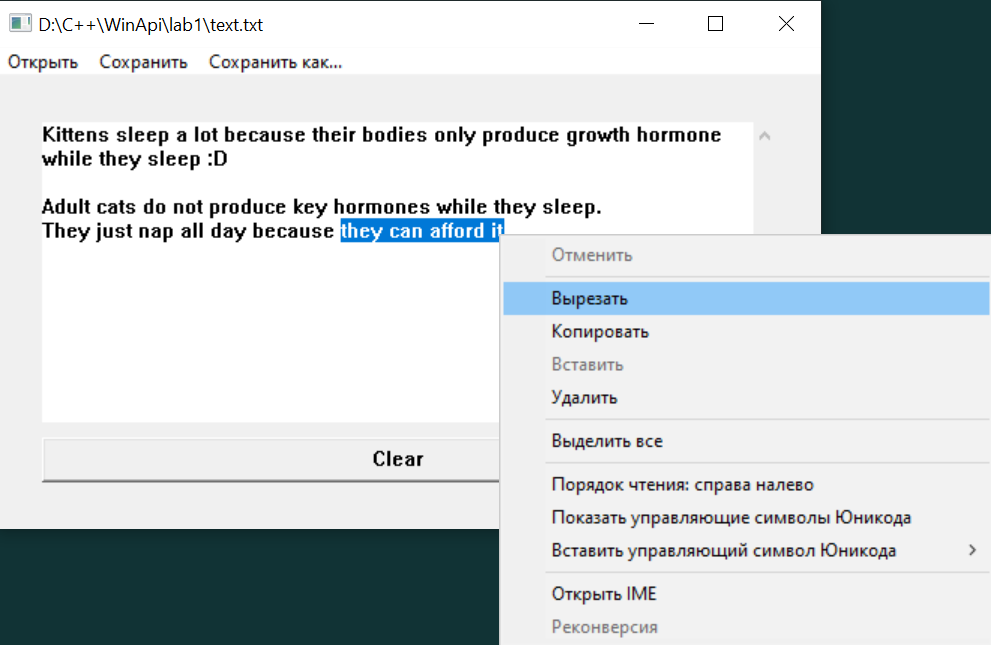
****

Рисунок 3.5 – Возможность выделения и копирования текста

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Марапулец Ю. В. Системное программирование в WIN API. Учебное пособие, 2021. – 31 с

[2] Начало работы с классическими приложениями для Windows, которые используют API Win32 [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/desktop-programming>.

[3] Сообщения окна [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/learnwin32/window-messages>.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

Листинг 1 – Файл Resource.h:

#pragma once

#define OnOpenMenuClicked 1

#define OnSaveAsMenuClicked 2

#define OnButtonClicked 3

#define OnSaveMenuClicked 4

#define TextBufferSize 256

char Buffer[TextBufferSize];

char currentFileName[260];

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

WNDCLASS WindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, LPCWSTR Name, WNDPROC wc);

void MainWndAddMenu(HWND hwnd);

void MainWndAddWidget(HWND hwnd);

void SetOpenFileParams(HWND hwnd);

void SaveData(LPCSTR path);

void LoadData(LPCSTR path);

HWND hEditConstrol;

char fileName[260];

OPENFILENAMEA ofn = { 0 };

bool isTextChanged = false;

int length = 0;

int loadFileLength;

Листинг 1 **–** Файл Source.cpp:

#include <Windows.h>

#include <string.h>

#include <shlwapi.h>

#pragma comment(lib, "shlwapi.lib")

#include "Resource.h"

int WINAPI WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ int nShowCmd) {

WNDCLASS MainClass = WindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_ARROW), hInstance, L"MainWndClass", WndProc);

if (!RegisterClassW(&MainClass)) { return -1; }

MSG MainMsg = { };

CreateWindow(L"MainWndClass", L"TextEditor", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 200, 100, 565, 360, NULL, NULL, NULL, NULL);

while (GetMessage(&MainMsg, NULL, NULL, NULL)) {

TranslateMessage(&MainMsg);

DispatchMessage(&MainMsg);

}

return 0;

}

WNDCLASS WindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, LPCWSTR Name, WNDPROC wc) {

WNDCLASS nwc = { };

nwc.hbrBackground = BGColor;

nwc.hCursor = Cursor;

nwc.hInstance = hInst;

nwc.lpszClassName = Name;

nwc.lpfnWndProc = wc;

return nwc;

}

LRESULT CALLBACK WndProc(HWND, UINT uMsg, WPARAM, LPARAM lParam) {

switch (uMsg) {

case WM\_COMMAND:

switch (wParam)

{

case OnOpenMenuClicked:

if (GetOpenFileNameA(&ofn))

{

LoadData(fileName);

lstrcpyA(currentFileName, fileName); // Сохраняем имя файла в переменной currentFileName

SetWindowTextA(hwnd, currentFileName);

loadFileLength = GetWindowTextLength(hEditConstrol);

isTextChanged = true;

}

break;

case OnSaveAsMenuClicked:

while (true)

{

if (GetSaveFileNameA(&ofn)) {

if (PathFileExistsA(fileName)) {

int result = MessageBoxW(hwnd, L"Файл с таким именем уже существует. Хотите перезаписать?", L"Подтвердить сохранение", MB\_YESNO | MB\_ICONQUESTION);

if (result == IDYES) {

SaveData(fileName);

MessageBoxW(hwnd, L"Файл сохранен!", L"Сохранение", MB\_OK);

isTextChanged = false;

break;

}

}

else {

SaveData(fileName);

isTextChanged = false;

break;

}

}

else {

break;

}

}

break;

case OnButtonClicked:

SetWindowTextW(hEditConstrol, L"");

isTextChanged = true;

break;

case OnSaveMenuClicked:

if (PathFileExistsA(fileName)) {

SaveData(fileName);

MessageBoxW(hwnd, L"Файл сохранен!", L"Сохранение", MB\_OK);

break;

}

else {

if (GetSaveFileNameA(&ofn)) {

SaveData(fileName);

MessageBoxW(hwnd, L"Файл сохранен!", L"Save", MB\_OK);

}

}

break;

default:

break;

}

break;

case WM\_CREATE:

MainWndAddMenu(hwnd);

MainWndAddWidget(hwnd);

SetOpenFileParams(hwnd);

break;

case WM\_DESTROY:

length = GetWindowTextLength(hEditConstrol);

if (length == loadFileLength) {

PostQuitMessage(0);

break;

}

if (isTextChanged || (isTextChanged == false && length > 1))

{

int result = MessageBox(hwnd, L"Хотите сохранить изменения?", L"Подтверждение", MB\_YESNO);

if (result == IDYES)

{

// Сохраняем изменения

if (PathFileExistsA(currentFileName))

{

SaveData(currentFileName);

}

else

{

if (GetSaveFileNameA(&ofn))

{

SaveData(fileName);

}

}

}

}

PostQuitMessage(0);

break;

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

}

}

void MainWndAddMenu(HWND hwnd) {

HMENU RootMenu = CreateMenu();

AppendMenu(RootMenu, MF\_STRING, OnOpenMenuClicked, L"Открыть");

AppendMenu(RootMenu, MF\_STRING, OnSaveMenuClicked, L"Сохранить");

AppendMenu(RootMenu, MF\_STRING, OnSaveAsMenuClicked, L"Сохранить как...");

SetMenu(hwnd, RootMenu);

}

void MainWndAddWidget(HWND hwnd) {

hEditConstrol = CreateWindow(L"edit", L" ", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE |WS\_VSCROLL , 30, 30, 490, 200, hwnd, NULL, NULL, NULL);

CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD, 30, 240, 475, 30, hwnd, (HMENU)OnButtonClicked, NULL, NULL);

}

void SaveData(LPCSTR path) {

HANDLE FileToSave = CreateFileA(

path,

GENERIC\_WRITE,

0,

NULL,

CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

int saveLenth = GetWindowTextLength(hEditConstrol) + 1;

char\* data = new char[saveLenth];

saveLenth = GetWindowTextA(hEditConstrol, data, saveLenth);

DWORD bytesIterated;

WriteFile(FileToSave, data, saveLenth, &bytesIterated, NULL);

CloseHandle(FileToSave);

delete[] data;

}

void LoadData(LPCSTR path) {

HANDLE FileToLoad = CreateFileA(

path,

GENERIC\_READ,

0,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

DWORD saveSize = GetFileSize(FileToLoad, NULL) + 1;

char\* data = new char[saveSize];

DWORD bytesIterated;

ReadFile(FileToLoad, data, saveSize, &bytesIterated, NULL);

data[bytesIterated] = '\0';

SetWindowTextA(hEditConstrol, data);

CloseHandle(FileToLoad);

delete[] data;

}

void SetOpenFileParams(HWND hwnd) {

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hwnd;

ofn.lpstrFile = fileName;

ofn.nMaxFile = sizeof(fileName);

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

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.lpstrInitialDir = "D:\\С++\\WinApi\\lab1";

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

}