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 WIN 32 API. ОКОННОЕ ПРИЛОЖЕНИЕ WIN 32 С МИНИМАЛЬНОЙ ДОСТАТОЧНОЙ ФУНКЦИОНАЛЬНОСТЬЮ. ОБРАБОТКА ОСНОВНЫХ ОКОННЫХ СООБЩЕНИЙ.

Выполнил студент гр.153502 Богомолов М.А.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc147500218)

[2 Теоретические сведения 4](#_Toc147500219)

[3 Описание функций программы 5](#_Toc147500220)

[3.1 Открытие файла, сохранение файла и создание файла 5](#_Toc147500221)

[3.2 Копирование и вставка текста 7](#_Toc147500222)

[Список использованных источников 9](#_Toc147500223)

[Приложение А (обязательное) Листинг кода 10](#_Toc147500224)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, таких как обработка оконных сообщений.

В качестве задачи необходимо построить приложение для чтения и редактирования текстовых документов с возможностью выделения и копирования текста в буфер обмена.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

*Windows API* (англ. *application programming interfaces*) – общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств *Microsoft Windows* корпорации Майкрософт. Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой *Windows*. Для создания программ, использующих *Windows API*, корпорация Майкрософт выпускает комплект разработчика программного обеспечения, который называется *Platform SDK* и содержит документацию, набор библиотек, утилит и других инструментальных средств для разработки.

*Windows API* спроектирован для использования в языке Си для написания прикладных программ, предназначенных для работы под управлением операционной системы *MS Windows*. Работа через *Windows API* – это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ. Более низкий уровень доступа, необходимый только для драйверов устройств, в текущих версиях *Windows* предоставляется через *Windows Driver Model*.

*Windows API* представляет собой множество функций, структур данных и числовых констант, следующих соглашениям языка Си. В то же время конвенция вызова функций отличается от *cdecl*, принятой для языка C: *Windows API* использует *stdcall* (*winapi*). Все языки программирования, способные вызывать такие функции и оперировать такими типами данных в программах, исполняемых в среде *Windows*, могут пользоваться этим *API*. В частности, это языки *C*++, *C*#, *Pascal*, *Visual Basic* и многие другие [1].

В графическом приложении *windows* окно — это прямоугольная область экрана, в которой приложение отображает выходные данные и получает входные данные от пользователя. Поэтому одной из первых задач графического приложения на основе *Windows* является создание окна.

Окно использует экран совместно с другими окнами, в том числе из других приложений. Только одно окно за раз может получать входные данные от пользователя. Пользователь может использовать мышь, клавиатуру или другое устройство ввода для взаимодействия с этим окном и приложением, которому оно принадлежит [2].

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

Согласно формулировке задачи, были спроектированы следующие функции программы:

– открытие текстового файла;

– создание нового текстового файла;

– редактирование текстового файла;

– копирование выделенного текста в буфер обмена;

– вставка текста из буфера обмена;

– сохранение файла.

## **3.1** **Открытие файла, сохранение файла и создание файла**

Для открытия текстового файла необходимо в меню нажать *File*-*Open* и в открывшемся диалоговом окне выбрать нужный файл (Рисунок 1).

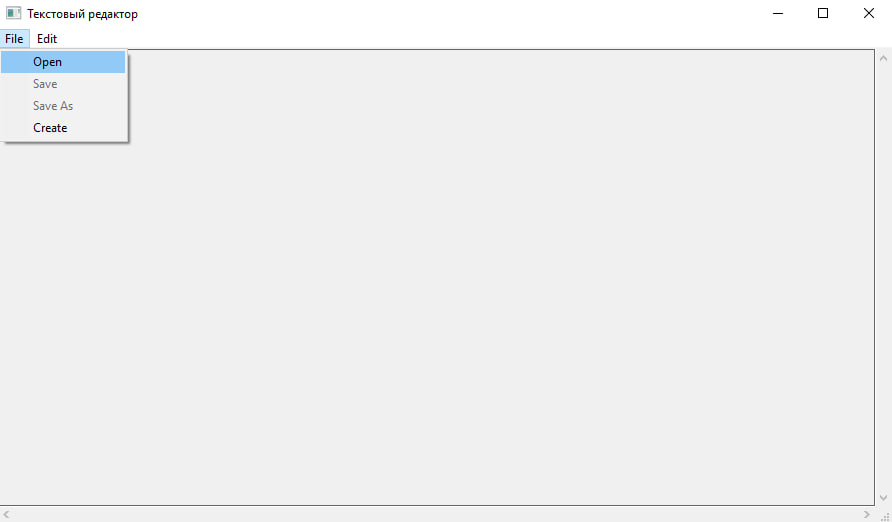


Рисунок 1 – Открытие файла

Для сохранения изменений в файле в открытой копии необходимо нажать в меню *File*-*Save*. В свою очередь, для сохранения изменений в файле в новой копии требуется нажать в меню *File*-*Save As* и в диалоговом окне выбрать нужную директорию и указать имя новой копии редактируемого файла (Рисунок 2).

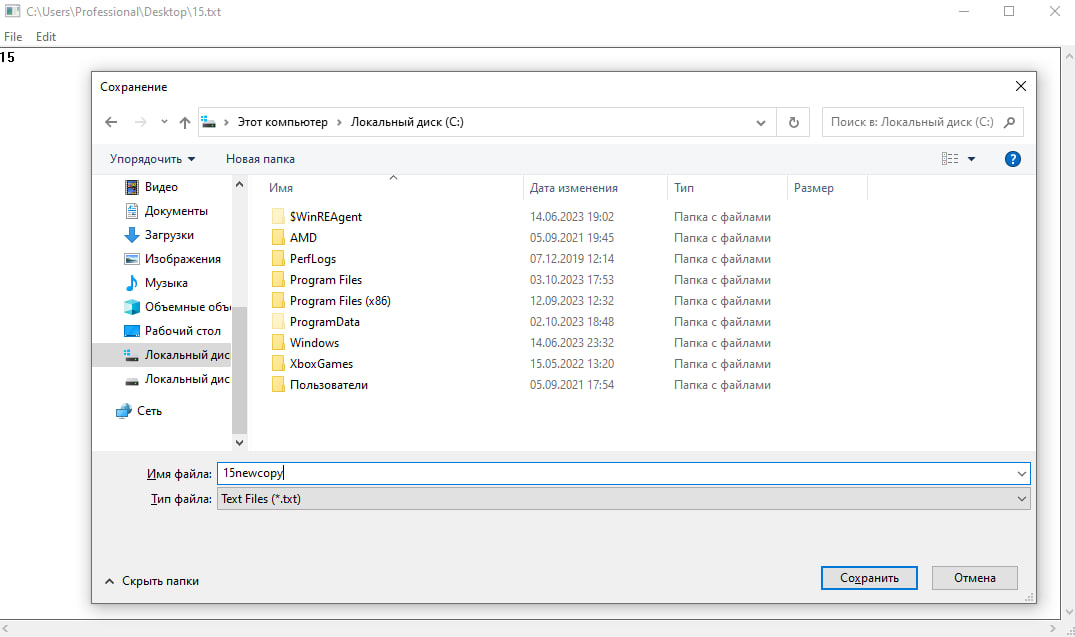


Рисунок 2 – Сохранение файла в новой копии

Для создания нового файла и открытия его в текстовом редакторе (по умолчанию он пуст) необходимо нажать в меню *File*-*Create*, после чего в диалоговом окне установить его название и директорию создания (Рисунок 3).

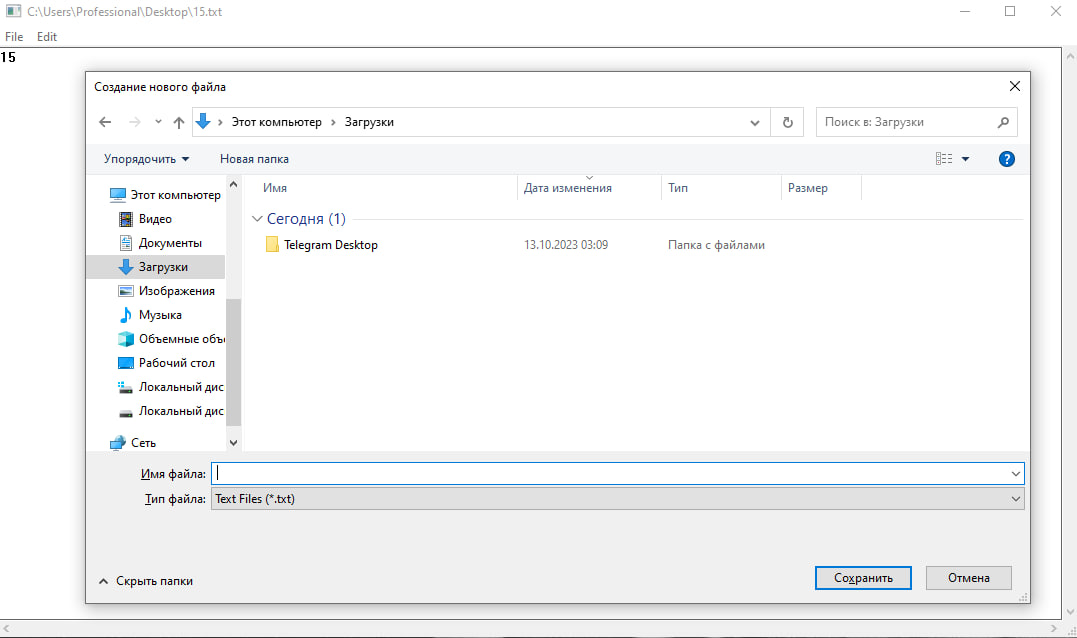


Рисунок 3 – Создание нового файла

## **3.2** **Копирование и вставка текста**

После открытия или создания файла поле редактирования текста в открытом или созданном файле становится активным. Помимо поля редактирования, активными становятся элементы меню *Edit*. Чтобы копировать текст из области редактирования требуется выделить его курсором и нажать в меню *Edit*-*Copy* (Рисунок 4).

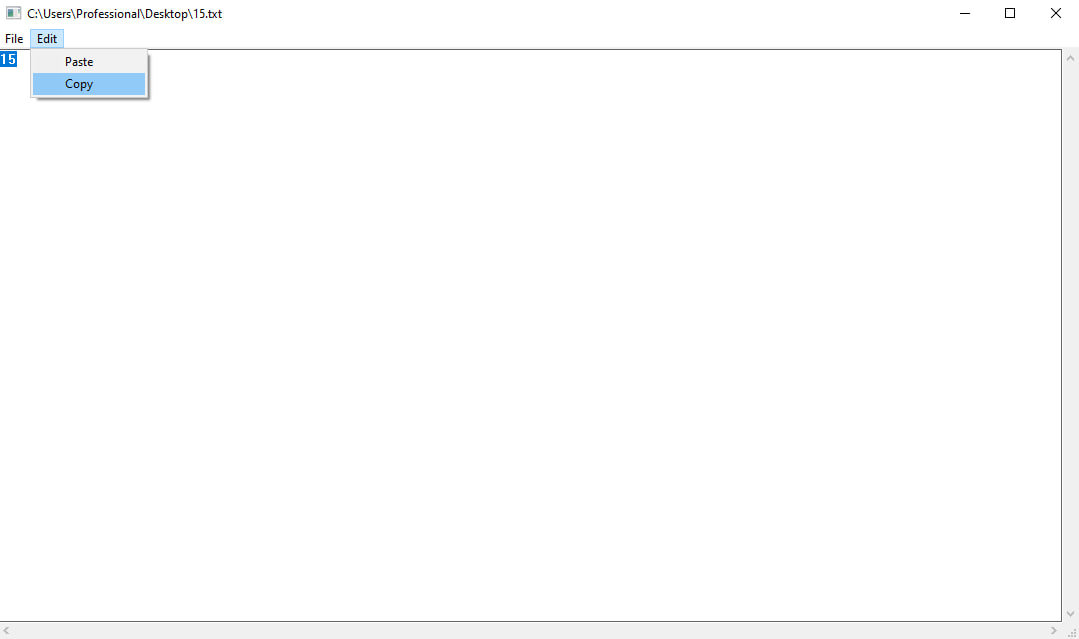


Рисунок 4 – Копирование выделенного текста

Для вставки скопированного текста в текущее местоположение курсора нужно нажать в меню *Edit*-*Paste* (Рисунок 5).



Рисунок 5 – Вставка скопированного текста

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Windows API [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Windows_API> – Дата доступа: 09.10.2023.

[2] Windows (окна и сообщения) [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/winmsg/windows> – Дата доступа: 10.10.2023.

ПРИЛОЖЕНИЕ А  
(обязательное)   
Листинг кода

Листинг 1 – Файл OSISP\_1.cpp

#include "framework.h"

#include "OSISP\_1.h"

#include <windows.h>

#include <commdlg.h>

#define MAX\_CONTENT\_SIZE 4096

HINSTANCE hInst;

HWND hwndEdit;

TCHAR currentFileName[MAX\_PATH] = \_T("");

#define ID\_FILE\_OPEN 1

#define ID\_FILE\_SAVE 2

#define ID\_FILE\_CREATE 3

#define ID\_FILE\_SAVEAS 4

#define ID\_EDIT\_COPY 5

#define ID\_EDIT\_PASTE 6

bool isEditingEnabled = false;

void OpenFile(HWND hwnd);

void SaveFile(HWND hwnd);

void CreateFile(HWND hwnd);

void SaveFileAs(HWND hwnd);

bool IsTextModified(HWND hwndEdit);

void UpdateWindowTitle(HWND hwnd);

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

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

{

hInst = hInstance;

WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS\_HREDRAW | CS\_VREDRAW, WndProc, 0L, 0L, GetModuleHandle(NULL), NULL, NULL, NULL, NULL, \_T("Sample"), NULL };

RegisterClassEx(&wc);

HWND hwnd = CreateWindow(\_T("Sample"), \_T("Текстовый редактор"), WS\_OVERLAPPEDWINDOW & ~WS\_THICKFRAME, CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, NULL, NULL, hInstance, NULL);

if (!hwnd) return 0;

HMENU hMenu = CreateMenu();

HMENU hSubMenu = CreatePopupMenu();

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

AppendMenu(hSubMenu, MF\_STRING | MF\_GRAYED, ID\_FILE\_SAVE, \_T("Save"));

AppendMenu(hSubMenu, MF\_STRING | MF\_GRAYED, ID\_FILE\_SAVEAS, \_T("Save As"));

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

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hSubMenu, \_T("File"));

HMENU hEditMenu = CreateMenu();

AppendMenu(hEditMenu, MF\_STRING | MF\_GRAYED, ID\_EDIT\_PASTE, \_T("Paste"));

AppendMenu(hEditMenu, MF\_STRING | MF\_GRAYED, ID\_EDIT\_COPY, \_T("Copy"));

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hEditMenu, \_T("Edit"));

SetMenu(hwnd, hMenu);

hwndEdit = CreateWindow(\_T("EDIT"), NULL, WS\_CHILD | WS\_VISIBLE | WS\_HSCROLL | WS\_VSCROLL | WS\_BORDER | ES\_MULTILINE | ES\_AUTOHSCROLL | ES\_AUTOVSCROLL, 0, 30, 800, 570, hwnd, NULL, hInstance, NULL);

EnableWindow(hwndEdit, FALSE);

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

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

{

switch (message)

{

case WM\_SIZE:

RECT rcClient;

GetClientRect(hwnd, &rcClient);

MoveWindow(hwndEdit, rcClient.left, rcClient.top, rcClient.right, rcClient.bottom, TRUE);

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case ID\_FILE\_OPEN:

if (IsTextModified(hwndEdit)) {

int result = MessageBox(hwnd, \_T("Сохранить изменения перед открытием нового файла?"), \_T("Подтверждение"), MB\_YESNOCANCEL | MB\_ICONQUESTION);

if (result == IDYES) {

SaveFile(hwnd);

}

else if (result == IDCANCEL) {

return 0;

}

}

OpenFile(hwnd);

SetFocus(hwndEdit);

break;

case ID\_FILE\_SAVE:

SaveFile(hwnd);

break;

case ID\_EDIT\_COPY:

if (OpenClipboard(hwnd)) {

EmptyClipboard();

CloseClipboard();

}

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

break;

case ID\_EDIT\_PASTE:

if (OpenClipboard(hwnd)) {

if (IsClipboardFormatAvailable(CF\_TEXT)) {

HANDLE hClipboardData = GetClipboardData(CF\_TEXT);

if (hClipboardData) {

char\* clipboardText = static\_cast<char\*>(GlobalLock(hClipboardData));

if (clipboardText) {

int textLength = MultiByteToWideChar(CP\_ACP, 0, clipboardText, -1, NULL, 0);

if (textLength > 0) {

wchar\_t\* utf8Text = new wchar\_t[textLength];

MultiByteToWideChar(CP\_ACP, 0, clipboardText, -1, utf8Text, textLength);

int cursorPos = SendMessage(hwndEdit, EM\_GETSEL, 0, 0);

SendMessage(hwndEdit, EM\_REPLACESEL, TRUE, (LPARAM)utf8Text);

delete[] utf8Text;

}

}

GlobalUnlock(hClipboardData);

}

}

CloseClipboard();

}

break;

case ID\_FILE\_SAVEAS:

SaveFileAs(hwnd);

break;

case ID\_FILE\_CREATE:

if (IsTextModified(hwndEdit)) {

int result = MessageBox(hwnd, \_T("Сохранить изменения перед созданием нового файла?"), \_T("Подтверждение"), MB\_YESNOCANCEL | MB\_ICONQUESTION);

if (result == IDYES) {

SaveFile(hwnd);

}

else if (result == IDCANCEL) {

return 0;

}

}

CreateFile(hwnd);

SetFocus(hwndEdit);

break;

}

EnableMenuItem(GetMenu(hwnd), ID\_FILE\_SAVE, IsWindowEnabled(hwndEdit) ? MF\_ENABLED : MF\_GRAYED);

EnableMenuItem(GetMenu(hwnd), ID\_FILE\_SAVEAS, IsWindowEnabled(hwndEdit) ? MF\_ENABLED : MF\_GRAYED);

EnableMenuItem(GetMenu(hwnd), ID\_EDIT\_COPY, IsWindowEnabled(hwndEdit) ? MF\_ENABLED : MF\_GRAYED);

EnableMenuItem(GetMenu(hwnd), ID\_EDIT\_PASTE, IsWindowEnabled(hwndEdit) ? MF\_ENABLED : MF\_GRAYED);

DrawMenuBar(hwnd);

break;

case WM\_CLOSE:

if (IsTextModified(hwndEdit)) {

int result = MessageBox(hwnd, \_T("Сохранить изменения перед закрытием?"), \_T("Сохранение"), MB\_YESNOCANCEL | MB\_ICONQUESTION);

if (result == IDYES) {

SaveFile(hwnd);

}

else if (result == IDCANCEL) {

return 0;

}

}

DestroyWindow(hwnd);

break;

case WM\_DROPFILES:

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

void OpenFile(HWND hwnd)

{

OPENFILENAME ofn;

TCHAR szFileName[MAX\_PATH] = \_T("");

ZeroMemory(&ofn, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hwnd;

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

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

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

if (GetOpenFileName(&ofn)) {

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

if (hFile != INVALID\_HANDLE\_VALUE) {

DWORD dwFileSize = GetFileSize(hFile, NULL);

if (dwFileSize != INVALID\_FILE\_SIZE) {

char\* buffer = new char[dwFileSize + 1];

DWORD dwRead;

if (ReadFile(hFile, buffer, dwFileSize, &dwRead, NULL)) {

buffer[dwRead] = '\0';

SendMessage(hwndEdit, WM\_SETTEXT, 0, (LPARAM)\_T(""));

SetWindowTextA(hwndEdit, buffer);

}

delete[] buffer;

}

CloseHandle(hFile);

lstrcpy(currentFileName, szFileName);

EnableWindow(hwndEdit, TRUE);

isEditingEnabled = true;

UpdateWindowTitle(hwnd);

}

}

}

void SaveFile(HWND hwnd)

{

if (lstrlen(currentFileName) == 0) {

OPENFILENAME ofn;

TCHAR szFileName[MAX\_PATH] = \_T("");

ZeroMemory(&ofn, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hwnd;

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

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_OVERWRITEPROMPT;

ofn.lpstrDefExt = \_T("txt");

if (!GetSaveFileName(&ofn)) {

return;

}

lstrcpy(currentFileName, szFileName);

UpdateWindowTitle(hwnd);

}

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

if (hFile != INVALID\_HANDLE\_VALUE) {

int textLength = GetWindowTextLength(hwndEdit);

if (textLength > 0) {

char\* buffer = new char[textLength + 1];

GetWindowTextA(hwndEdit, buffer, textLength + 1);

DWORD dwWritten;

WriteFile(hFile, buffer, textLength, &dwWritten, NULL);

delete[] buffer;

}

CloseHandle(hFile);

}

}

void CreateFile(HWND hwnd)

{

OPENFILENAME ofn;

TCHAR szFileName[MAX\_PATH] = \_T("");

ZeroMemory(&ofn, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hwnd;

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

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

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

ofn.lpstrDefExt = \_T("txt");

ofn.lpstrTitle = \_T("Создание нового файла");

if (GetSaveFileName(&ofn)) {

lstrcpy(currentFileName, szFileName);

UpdateWindowTitle(hwnd);

SetWindowText(hwndEdit, \_T(""));

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

if (hFile != INVALID\_HANDLE\_VALUE) {

CloseHandle(hFile);

EnableWindow(hwndEdit, TRUE);

isEditingEnabled = true;

}

}

}

void SaveFileAs(HWND hwnd) {

OPENFILENAME ofn;

TCHAR szFileName[MAX\_PATH] = \_T("");

ZeroMemory(&ofn, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hwnd;

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

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_OVERWRITEPROMPT;

ofn.lpstrDefExt = \_T("txt");

if (GetSaveFileName(&ofn)) {

lstrcpy(currentFileName, szFileName);

UpdateWindowTitle(hwnd);

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

if (hFile != INVALID\_HANDLE\_VALUE) {

int textLength = GetWindowTextLength(hwndEdit);

if (textLength > 0) {

char\* buffer = new char[textLength + 1];

GetWindowTextA(hwndEdit, buffer, textLength + 1);

DWORD dwWritten;

WriteFile(hFile, buffer, textLength, &dwWritten, NULL);

delete[] buffer;

}

CloseHandle(hFile);

}

}

}

bool IsTextModified(HWND hwndEdit) {

return IsWindowEnabled(hwndEdit);

}

void UpdateWindowTitle(HWND hwnd) {

if (lstrlen(currentFileName) > 0) {

SetWindowText(hwnd, currentFileName);

}

else {

SetWindowText(hwnd, \_T("Текстовый редактор"));

}

}