Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные системы и системное программирование

Отчёт

к лабораторной работе

на тему

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

Студент: гр.153502

Кирзнер А. П.

Проверил: Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

1 Цель работы 3

2 Теоретические сведения 4

3 Результат выполнения программы 5

Список использованных источников 9

Приложение А 10

# 1 ЦЕЛЬ РАБОТЫ

Возобновление, закрепление и развитие навыков программирования оконных приложений *Windows*: структура приложения, цикл обработки сообщений, организация взаимодействия посредством сообщений, создание и использование окон и оконных элементов управления, использование базовых средств графики *Windows*.

Построить приложение для чтения и редактирования текстовых документов с возможностью выделения и копирования текста в буфер обмена.

# 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Очевидно, что окна являются центральными для *Windows*. Элементы управления пользовательского интерфейса, такие как кнопки и поля редактирования, сами являются окнами. Основное различие между элементом управления пользовательского интерфейса и окном приложения заключается в том, что элемент управления не существует сам по себе. Вместо этого элемент управления располагается относительно окна приложения. При перетаскивании окна приложения элемент управления перемещается вместе с ним, как и ожидалось. Кроме того, элемент управления и окно приложения могут взаимодействовать друг с другом.

Поэтому, необходимо думать об окне как о программной конструкции, которая занимает определенную часть экрана, может и не отображаться в данный момент, знает, как рисовать себя, и отвечает на события пользователя или операционной системы.

Для создания окна необходимо зарегистрировать класс окна с помощью функции *RegisterClassEx* и создать окно с помощью функции *CreateWindowEx*. Окно может иметь различные свойства, такие как заголовок, размеры, стиль и обработчики сообщений.

Во время выполнения приложение получит тысячи сообщений. Кроме того, приложение может иметь несколько окон, каждое из которых имеет собственную процедуру окна. Как программа получает все эти сообщения и доставляет их в правильную процедуру окна? Приложению требуется цикл для получения сообщений и их отправки в правильные окна.

Для каждого потока, создающего окно, операционная система создает очередь для оконных сообщений. Эта очередь содержит сообщения для всех окон, созданных в этом потоке. Сама очередь скрыта от программы. Однако можно извлечь сообщение из очереди, вызвав функцию GetMessage. Эта функция удаляет первое сообщение из заголовка очереди. Если очередь пуста, функция блокируется до тех пор, пока в очередь не будет помещено другое сообщение.

Для обработки оконных сообщений необходимо определить функцию оконной процедуры (*WndProc*), которая будет вызываться системой при возникновении сообщения. В функции *WndProc* нужно обрабатывать различные типы сообщений с помощью условных операторов и выполнять соответствующие действия.

# 3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В результате выполнения лабораторной работы можно увидеть оконное приложение с простым пользовательским интерфейсом, напоминающий системное приложение «Блокнот» (рисунок 1).

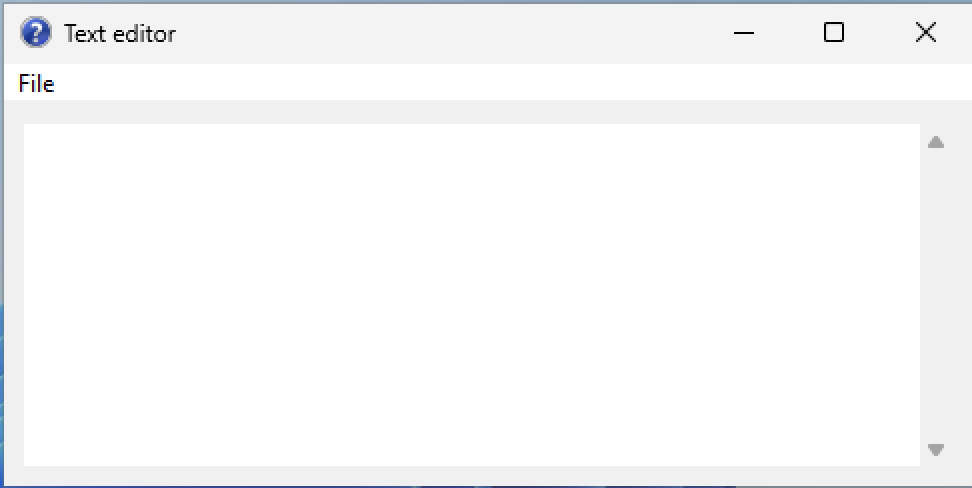


Рисунок 1 – Графический интерфейс программы

В соответствии с рисунком 2 при запуске приложения можно заметить, что поле для ввода текста пустое, никакой файл для загрузки не выбран, а также кнопка «*Save*» в меню «*File*» отключена.

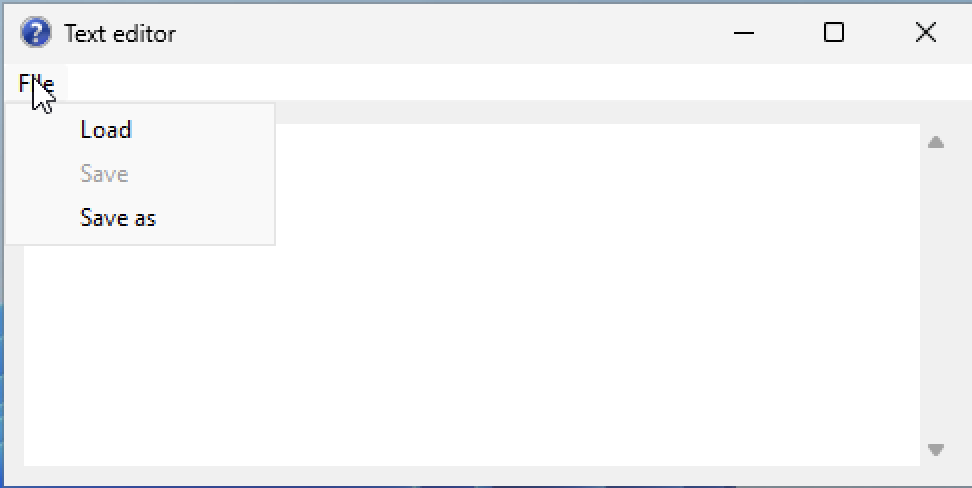


Рисунок 2 – Изначальное состояние элементов меню

При нажатии кнопки «*Load*» открывается файловый менеджер, отображающий и позволяющий выбрать только файлы с расширением *.txt* (рисунок 3).

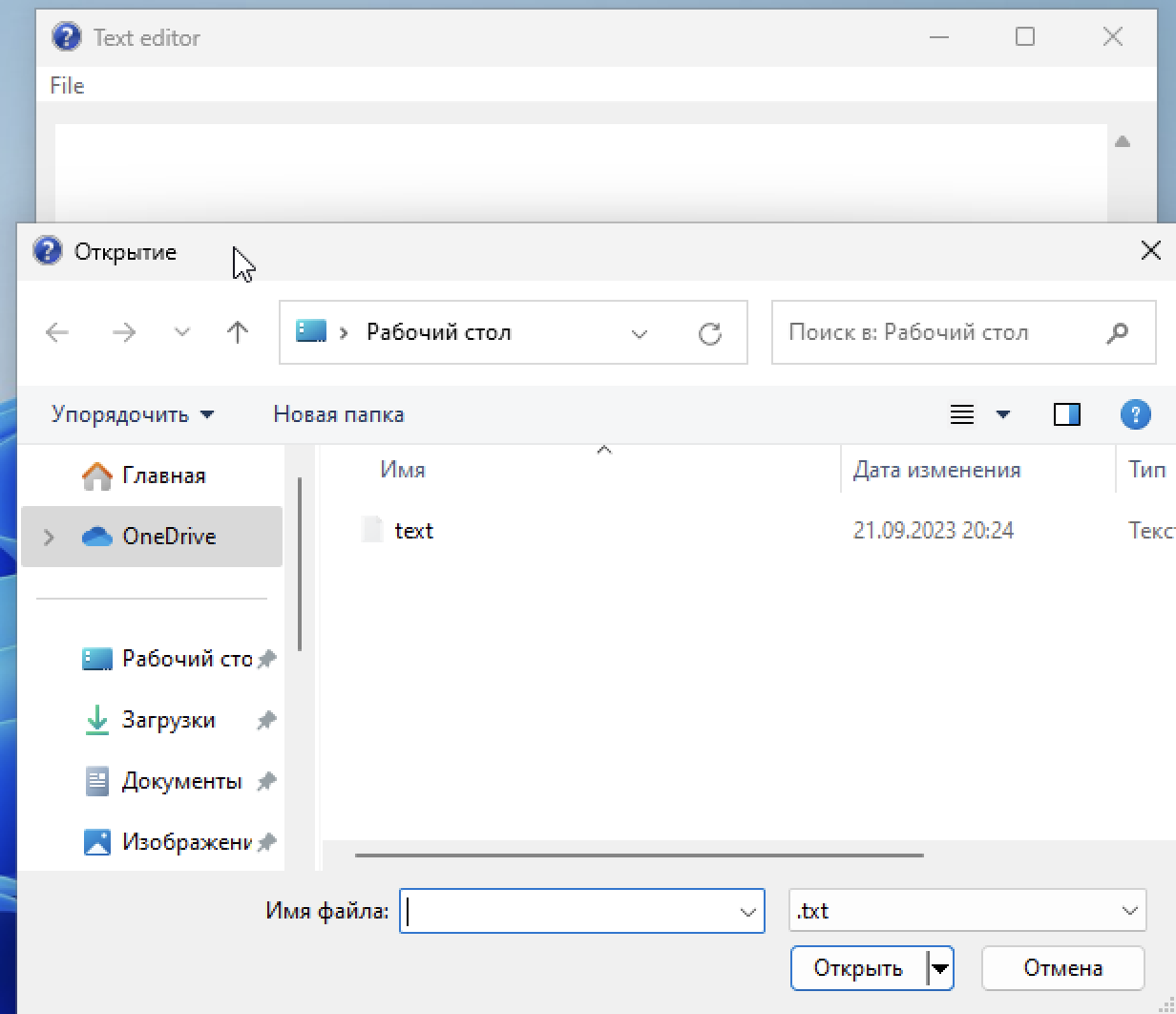


Рисунок 3 – Выбор файла для загрузки

При выборе файла в после для ввода текста появляется весь текст необходимого файла, а в заголовке окна отображается название выбранного файла (рисунок 4). Кнопка «*Save*» становится доступной.

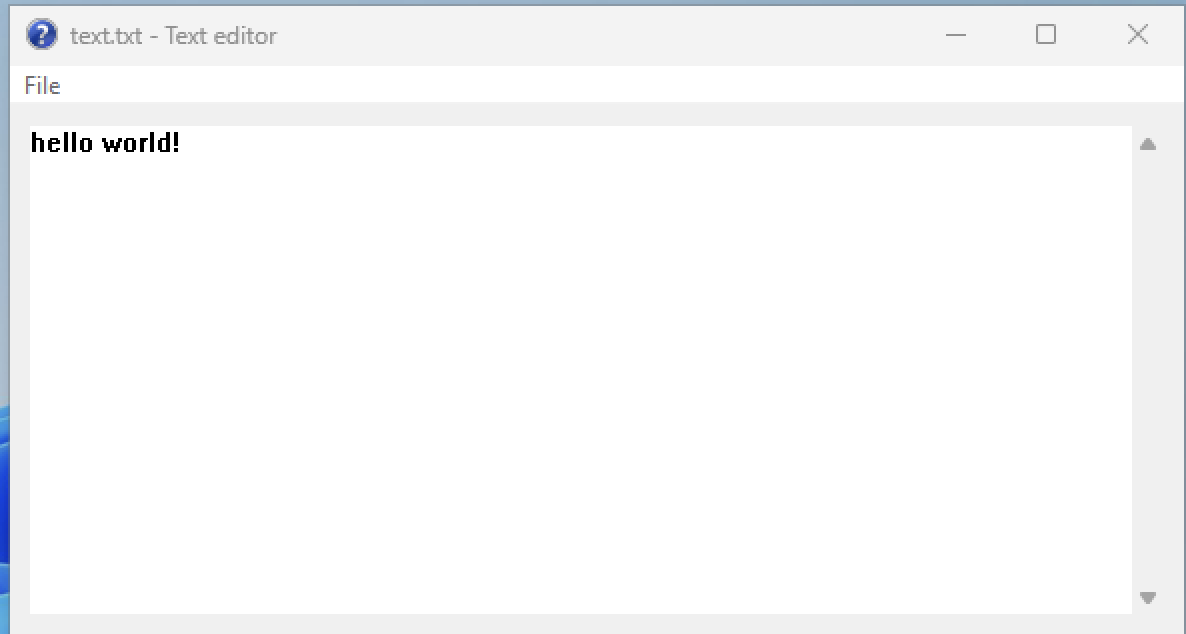


Рисунок 4 – Отображение содержимого выбранного файла

Кнопка «*Save*» позволяет сохранить изменения в загруженном файле, а кнопка «*Save as*» – сохранить текст в любом текстовом файле. Также стоит отметить, что в соответствии с рисунком 5 при выборе уже существующего файла всплывёт окно с подтверждением перезаписи содержимого файла.

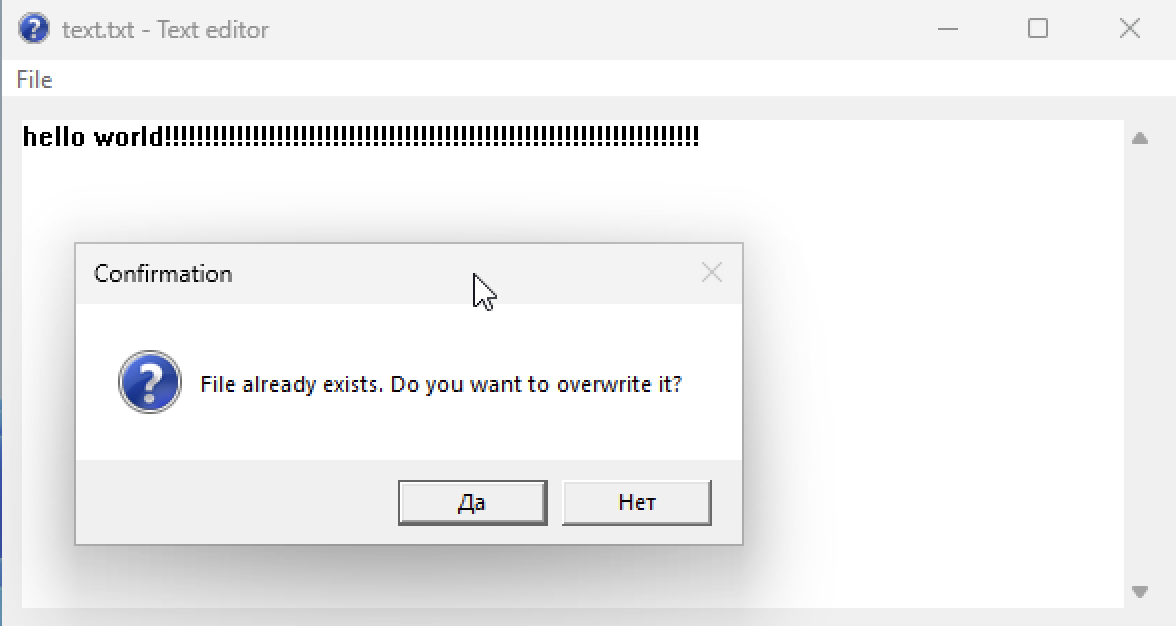


Рисунок 5 – Подтверждение перезаписи существующего файла

При загрузке очередного файла в текстовый редактор либо при выходе из приложения, если последние изменения не были сохранены, то появится вспомогательное окно, помогающее сохранить изменения при необходимости (рисунок 6).

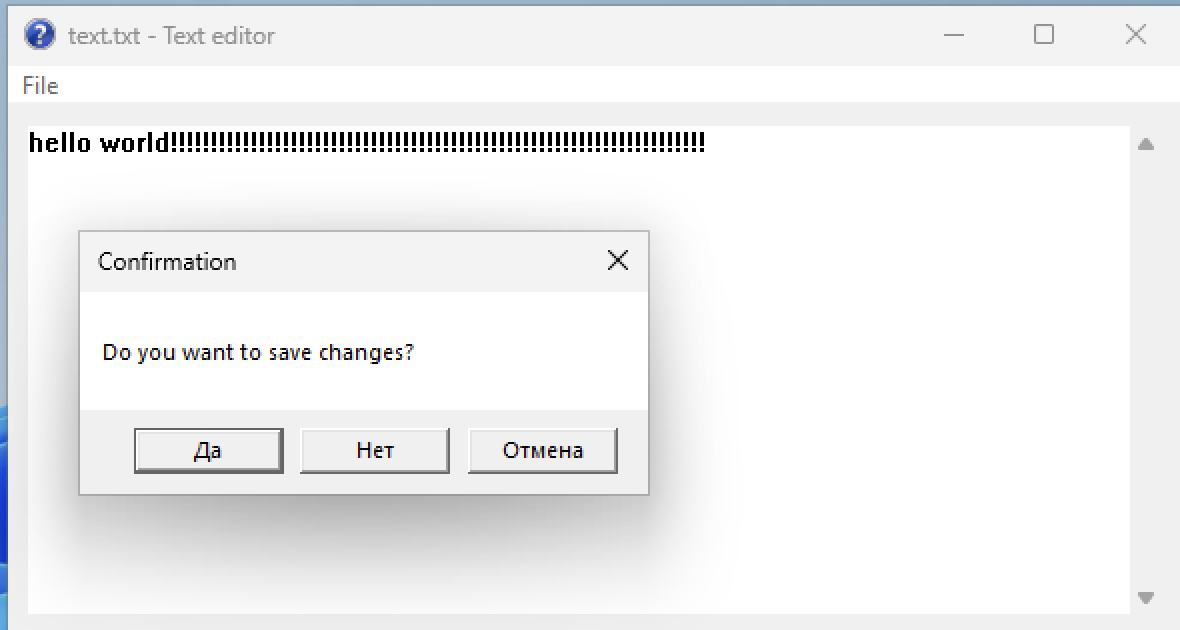


Рисунок 6 – Подтверждение сохранения изменений

Следует отметить, что в соответствии с рисунком 7 выделение текста, копирование в буфер обмена, а также вставка текста из буфера обмена осуществима.

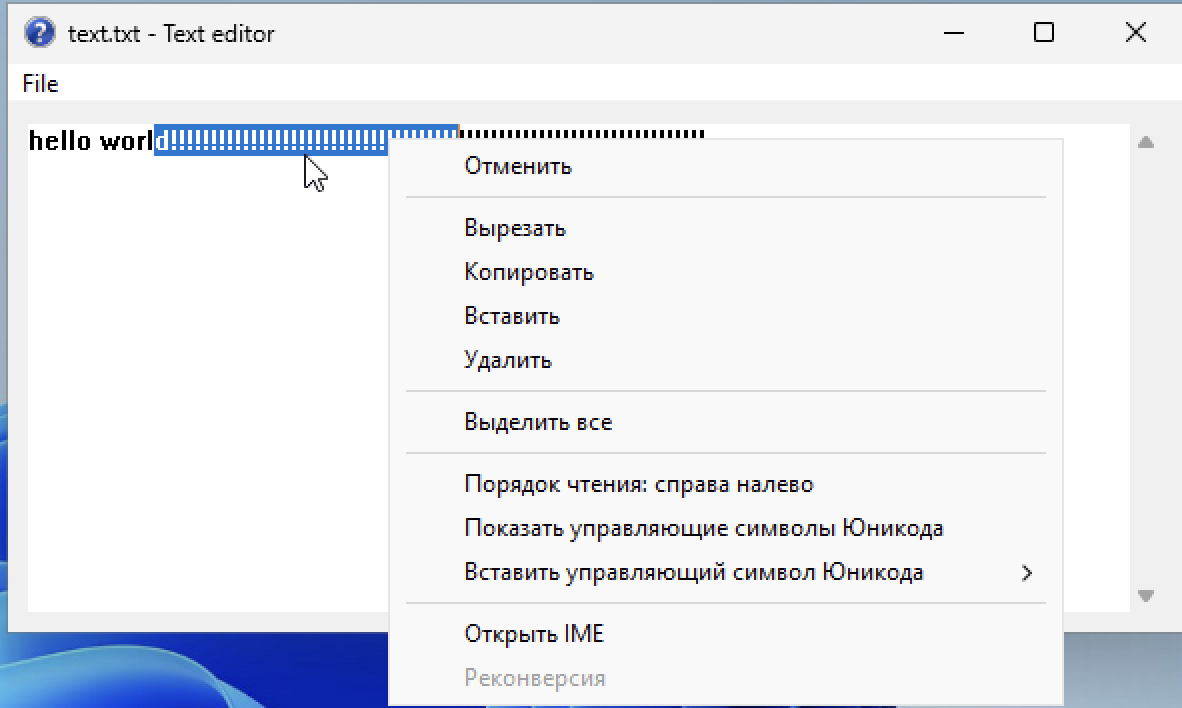


Рисунок 7 – Возможность выделения, копирования и вставки текста

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Что такое окно? [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/learnwin32/what-is-a-window>.

[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**

**Файл Lab1.h**

#pragma once

#define OnSaveFile 1

#define OnSaveFileAs 2

#define OnLoadFile 3

#define TextBufferSize 256

char Buffer[TextBufferSize];

bool isFileLoaded = false;

HWND hEditControl;

char filename[260];

OPENFILENAMEA ofn;

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndMenus(HWND hWnd);

void MainWndAddWidgets(HWND hWnd);

void SaveData(LPCSTR path);

void LoadData(LPCSTR path);

void SetOpenFileParams(HWND hWnd);

LPCSTR GetFileName(LPCSTR filepath);

bool FileExists(LPCSTR path);

bool HasTextChanged(HWND hEditControl, LPCSTR filepath);

void LoadFile(HWND hWnd);

void ChangeWindowTitle(HWND hWnd);

**Файл Lab1.cpp**

HANDLE FileToLoad = CreateFileA(

path,

GENERIC\_READ,

0,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

DWORD fileSize = GetFileSize(FileToLoad, NULL);

char\* data = new char[fileSize + 1];

DWORD bytesRead = 0;

ReadFile(FileToLoad, data, fileSize, &bytesRead, NULL);

data[bytesRead] = '\0';

CloseHandle(FileToLoad);

SetWindowTextA(hEditControl, data);

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 = ".txt\0\*.txt\0";

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.lpstrInitialDir = NULL;

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

}

LPCSTR GetFileName(LPCSTR filepath) {

LPCSTR filename = strrchr(filepath, '\\');

if (filename) {

filename++;

}

else {

filename = filepath;

}

return filename;

}

bool FileExists(LPCSTR path) {

DWORD attributes = GetFileAttributesA(path);

return (attributes != INVALID\_FILE\_ATTRIBUTES && !(attributes & FILE\_ATTRIBUTE\_DIRECTORY));

}

bool HasTextChanged(HWND hEditControl, LPCSTR filepath) {

bool result;

int editLength = GetWindowTextLength(hEditControl) + 1;

char\* editText = new char[editLength];

GetWindowTextA(hEditControl, editText, editLength);

if (FileExists(filepath)) {

HANDLE FileToCheck = CreateFileA(

filepath,

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

DWORD fileSize = GetFileSize(FileToCheck, NULL);

char\* fileData = new char[fileSize + 1];

DWORD bytesRead = 0;

ReadFile(FileToCheck, fileData, fileSize, &bytesRead, NULL);

fileData[bytesRead] = '\0';

CloseHandle(FileToCheck);

int editLength = GetWindowTextLength(hEditControl) + 1;

char\* editText = new char[editLength];

GetWindowTextA(hEditControl, editText, editLength);

result = strcmp(fileData, editText) != 0;

delete[] fileData;

delete[] editText;

return result;

}

else {

result = strcmp("", editText) != 0;

}

delete[] editText;

return result;

}

void ChangeWindowTitle(HWND hWnd) {

char newTitle[300];

strcpy\_s(newTitle, GetFileName(filename));

strcat\_s(newTitle, " - Text editor");

SetWindowTextA(hWnd, newTitle);

}

void LoadFile(HWND hWnd) {

if (GetOpenFileNameA(&ofn)) {

LoadData(filename);

ChangeWindowTitle(hWnd);

EnableMenuItem(GetMenu(hWnd), OnSaveFile, MF\_ENABLED);

isFileLoaded = true;

}

}