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. 153503

Филипеня А.Д.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146622147)

[2 Краткие теоретические сведения 4](#_Toc146622148)

[3 Результаты выполнения лабораторной работы 6](#_Toc146622149)

[Выводы 8](#_Toc146622150)

[Список использованных источников 9](#_Toc146622151)

[Приложение А (обязательное) Листинг кода 10](#_Toc146622152)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающего минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, использования виджетов и обработки оконных сообщений. Создать текстовый редактор с графическим интерфейсом, реализовать функциональность сохранения и загрузки файлов.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений. [1]

GDI (Graphics Device Interface) — это часть Win32 API, отвечающая за графический вывод и взаимодействие с графическим оборудованием. GDI предоставляет функции для рисования графических объектов, создания изображений, управления цветами и шрифтами, а также реализации различных эффектов. Этот компонент позволяет разработчикам создавать интерфейсы пользовательских приложений, включая окна, кнопки, текст и другие графические элементы. GDI обеспечивает визуальное качество и интерактивность приложений, делая их более привлекательными и функциональными для пользователей. [1]

Совместное использование Win32 API и GDI позволяет разработчикам создавать мощные и интерактивные графические приложения под операционной системой Windows, обеспечивая широкие возможности по работе с графикой, окнами и элементами управления.

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

– Win32 API (Application Programming Interface): это набор функций и структур, предоставляемых операционной системой Windows для разработки приложений под Windows. Для создания приложения было использовано множество функций Win32 API для создания окна, обработки сообщений, управления таймерами и других задач. [1]

– Процедура обработки сообщений (SoftwareMainProcedure): это ключевой элемент приложений Windows. SoftwareMainProcedure — это функция, которая обрабатывает сообщения, отправляемые операционной системой приложению. В данном приложении SoftwareMainProcedure была реализована для обработки различных сообщений, таких как создание окна, команды меню, перерисовка и таймер.

– Меню: создано пользовательское меню с использованием функций CreateMenu и AppendMenu. Это меню позволяет пользователю сохранять или загружать файл, а также закрывать программу.

– Управление окнами: приложение управляет окном, его размерами, заголовком и стилями с использованием функций Win32 API, таких как CreateWindow и ShowWindow.

– Обработка событий: код обрабатывает различные события, такие как нажатия кнопок, изменение текста в поле ввода и закрытие окна. Это обеспечивает реакцию приложения на действия пользователя.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был разработан текстовый редактор с возможностью сохранения и загрузки файлов. При запуске программы пользователь видит базовый интерфейс текстового редактора с полем для ввода текста и кнопкой для его полного удаления. Результат работы программы показан на рисунке 3.1.



Рисунок 3.1 – Результат работы программы

Также пользователь имеет доступ к меню, при помощи которого может сохранить введенный текст в файл или загрузить файл и редактировать его содержимое. Интерфейс меню показан на рисунке 3.2.



Рисунок 3.2 – Интерфейс меню

## ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены основы программирования в Win32 API. Как результат, создано оконное приложение с минимальной достаточной функциональностью и реализована обработка основных оконных сообщений.  Разработан текстовый редактор с функциональностью сохранения и загрузки файлов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.

[2] [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/winelements/> – Дата доступа 27.09.2023

[3] [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/windows/win32/controls/button-messages> – Дата доступа 27.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**SoftwareMain.cpp**

#include <Windows.h>

#include <string>

#include "resource.h"

#include "SoftwareDefinitions.h"

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

HBRUSH backgroundColor = CreateSolidBrush(RGB(204, 255, 204));

WNDCLASS SoftwareMainClass = NewWindowClass(backgroundColor, LoadCursor(NULL, IDC\_ARROW), hInst, LoadIcon(hInst, MAKEINTRESOURCE(IDI\_ICON1)), L"MainWndClass", SoftwareMainProcedure);

CreateSolidBrush(RGB(0, 0, 255));

if (!RegisterClassW(&SoftwareMainClass)) {

return -1;

}

MSG SoftwareMainMessage = { 0 };

CreateWindow(L"MainWndClass", L"Main window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 800, 500, NULL, NULL, NULL, NULL);

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

TranslateMessage(&SoftwareMainMessage);

DispatchMessage(&SoftwareMainMessage);

}

return 0;

}

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

WNDCLASS WindowClass = { 0 };

WindowClass.hbrBackground = BdColor;

WindowClass.hCursor = Cursor;

WindowClass.hInstance = hInst;

WindowClass.hIcon = Icon;

WindowClass.lpszClassName = Name;

WindowClass.lpfnWndProc = Procedure;

return WindowClass;

}

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

switch (msg)

{

case WM\_COMMAND:

switch (wp)

{

case OnButtonClicked:

SetWindowTextA(hEditControl, "");

break;

case OnReadField:

countOfReadChars = GetWindowTextA(hEditControl, Buffer, TextBufferSize);

SetWindowTextA(hStaticControl, Buffer);

SetWindowTextA(hStaticControl, std::to\_string(countOfReadChars).c\_str());

break;

case OnSaveFile:

if (GetSaveFileNameA(&file)) {

SaveData(path);

};

break;

case OnLoadFile:

if (GetOpenFileNameA(&file)) {

LoadData(path);

};

break;

case OnExitClicked:

PostQuitMessage(0);

break;

default:

break;

}

break;

case WM\_CREATE:

AddMenu(hWnd);

AddWidgets(hWnd);

SetOpenFileParams(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:return DefWindowProc(hWnd, msg, wp, lp);

}

}

void AddMenu(HWND hWnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

AppendMenu(SubMenu, MF\_STRING, OnSaveFile, L"Save");

AppendMenu(SubMenu, MF\_STRING, OnLoadFile, L"Load");

//разделитель меню,используем флаг separator

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitClicked, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

SetMenu(hWnd, RootMenu);

}

void AddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Enter text", WS\_VISIBLE | WS\_CHILD| ES\_CENTER, 5, 5, 780, 20, hWnd, NULL, NULL, NULL);

hEditControl = CreateWindowA("edit", "", ES\_MULTILINE | WS\_VSCROLL | WS\_VISIBLE | WS\_CHILD, 5, 30, 780, 250, hWnd, NULL, NULL, NULL);

CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD, 5, 300, 120, 30, hWnd, (HMENU)OnButtonClicked, NULL, NULL);

}

void SaveData(LPCSTR path) {

HANDLE file = CreateFileA(path, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

int length = GetWindowTextLength(hEditControl)+1;

char\* data = new char[length];

length = GetWindowTextA(hEditControl, data, length);

DWORD bytesIterated; //кол-во считанных символов

WriteFile(file, data, length, &bytesIterated, NULL);

CloseHandle(file);

delete[] data;

}

void LoadData(LPCSTR path) {

HANDLE file = CreateFileA(path, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

DWORD bytesIterated;

ReadFile(file, Buffer, TextBufferSize, &bytesIterated, NULL);

SetWindowTextA(hEditControl, Buffer);

CloseHandle(file);

}

void SetOpenFileParams(HWND hWnd) {

ZeroMemory(&file, sizeof(file));

file.lStructSize = sizeof(file);

file.hwndOwner = hWnd;

file.lpstrFile = path;

file.nMaxFile = sizeof(path);

file.lpstrFilter = ".txt";

file.lpstrFileTitle = NULL;

file.nMaxFileTitle = 0;

file.lpstrInitialDir = "D:\\5 sem\\SystemProgramming\\Lab1\\example.txt";

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

}

**SoftwareMain.h**

#define OnMenuClicked 1

#define OnExitClicked 2

#define OnButtonClicked 3

#define OnReadField 4

#define OnSaveFile 5

#define OnLoadFile 6

#define OnNewWindow 7

#define TextBufferSize 256

char Buffer[TextBufferSize];

int countOfReadChars;

HWND hEditControl;

HWND hStaticControl;

char path[200];

OPENFILENAMEA file;

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

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

void AddMenu(HWND hWnd);

void AddWidgets(HWND hWnd);

void SaveData(LPCSTR path);

void LoadData(LPCSTR path);

void SetOpenFileParams(HWND hWnd);