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 WIN32 API. ОКОННОЕ ПРИЛОЖЕНИЕ WIN32 С МИНИМАЛЬНОЙ ФУНКЦИОНАЛЬНОЙ ДОСТАТОЧНОСТЬЮ. ОБРАБОТКА ОСНОВНЫХ ОКОННЫХ СООБЩЕНИЙ

Выполнил: студент группы 153503

Сироткин Егор Алексеевич

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[3 Результаты выполнения лабораторной работы 5](#_Toc146728179)

[Выводы 7](#_Toc146728180)

[Список использованных источников 8](#_Toc146728181)

[Приложение А (обязательное) Листинг кода 9](#_Toc146728182)

## 1 ЦЕЛЬ РАБОТЫ

Цель данной лабораторной работы заключается в разработке приложения с использованием Win32 API. Это приложение будет обладать ограниченным набором функций, который поможет освоить основные навыки программирования с использованием Win32 API, включая создание оконных виджетов и обработку различных видов оконных сообщений, включая как стандартные, так и созданные пользователем. Реализовать вышеупомянутые требования на примере приложения списка дел, которое обрабатывает основные оконные сообщения через функцию WindowProc. Основные сообщения, такие как WM\_CREATE, WM\_COMMAND, и WM\_DESTROY, обрабатываются для осуществления добавления дел, использования меню, работоспособности всего приложения.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Application Programming Interface) это набор функций и подпрограмм, которые предоставляют программную связь с операционной системой Windows. Этот API обеспечивает средства для управления операционной системой, взаимодействия с ней, а также создания приложений, специально разработанных для работы в среде Windows.

Win32 API объединяет множество функций, предоставляя доступ к различным сервисам операционной системы. Приложения могут использовать эти функции для разнообразных задач, включая создание процессов и обработку оконных сообщений.

Для выполнения данной лабораторной работы, были применены следующие теоретические знания и концепции:

­– Win32 API (Application Programming Interface) - это набор функций и подпрограмм, предоставляемых операционной системой Windows для обеспечения взаимодействия с приложениями. С помощью этого программного интерфейса предоставляется доступ к разнообразным функциональным возможностям операционной системы Windows, таким как создание окон, обработка сообщений, управление файлами и многие другие.

– Обработка событий и сообщений является основным способом взаимодействия с пользователем в приложениях под Windows.

– Создание пользовательского интерфейса включает в себя использование стандартных элементов управления Windows, которые настраиваются с помощью функций Win32 API.

– Работа с окнами и элементами управления включает в себя создание окон приложения с заданными параметрами, такими как размер, положение и стиль.

– Обработка меню осуществляется через функции Win32 API, включая создание меню, добавление пунктов меню и обработку выбранных команд.

– Дизайн пользовательского интерфейса приложения реализован с учетом стандартных принципов, таких как расположение элементов и визуальное оформление текста и элементов интерфейса.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения данной лабораторной работы было разработано приложение для записи списка дел. Предоставляется возможность выбора заголовка и текста заметки (рисунок 1).



Рисунок 1 - Главное окно

Пользователь может вписывать заголовок заметки и ее содержимое (рисунок 2).

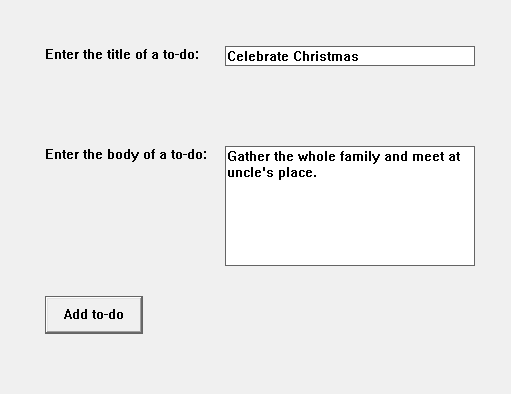


Рисунок 2 ─ Добавление заголовка и содержимого

После нажатия на кнопку “Add to-do” заметка пользователя с заголовком и содержимым попадает в поле для записей (рисунок 3).

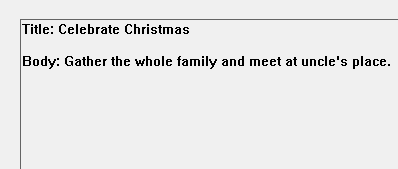


Рисунок 3 – Заметка в поле для записей

## 

## ВЫВОДЫ

В ходе данной работы было разработано оконное приложение, предназначенное для создания и управления заметками. Приложение позволяет добавлять и просматривать записи. Пользователь может вводить текст заметок и просматривать их. Процесс взаимодействия с приложением осуществляется через пользовательский интерфейс, предоставляющий удобные инструменты для работы с заметками. Таким образом, данная лабораторная работа позволила ознакомиться с основами программирования в Win32 API и создать оконное приложение, специально адаптированное для работы с заметками.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Безруков В. Win32 API. Программирование /учебное пособие. ─ СПб: СПбГУ ИТМО, 2009. ─ 90 с.

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api.

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

Листинг 1 – Файл Lab1.cpp

#include <windows.h>

#include <stdlib.h>

#define FILE\_MENU\_NEW 1

#define FILE\_MENU\_OPEN 2

#define FILE\_MENU\_EXIT 3

#define GENERATE\_BUTTON 4

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

void AddMenus(HWND);

void AddControls(HWND);

HWND hTitle, hBody, hOut;

HMENU hMenu;

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

WNDCLASS wc = { 0 };

wc.hbrBackground = (HBRUSH)COLOR\_WINDOW;

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hInstance = hInst;

wc.lpszClassName = L"Lab1";

wc.lpfnWndProc = WindowProcedure;

if (!RegisterClassW(&wc)) {

return -1;

}

CreateWindowW(L"Lab1", L"Lab1", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 1400, 700, NULL, NULL, NULL, NULL);

MSG msg = { 0 };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

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

switch (msg) {

case WM\_COMMAND:

switch (wp) {

case FILE\_MENU\_EXIT:

DestroyWindow(hWnd);

break;

case FILE\_MENU\_NEW:

MessageBeep(MB\_ICONINFORMATION);

break;

case GENERATE\_BUTTON:

wchar\_t title[31], body[200], out[250];

int titleLength = GetWindowTextLength(hTitle);

int bodyLength = GetWindowTextLength(hBody);

if (titleLength == 0 || bodyLength == 0) {

MessageBox(hWnd, L"Both title and body must be filled with non-space characters!", L"Validation Error", MB\_ICONERROR);

}

else {

GetWindowText(hTitle, title, 31);

GetWindowText(hBody, body, 200);

if (title[0] == L' ' || body[0] == L' ') {

MessageBox(hWnd, L"Title and body cannot start with a space character!", L"Validation Error", MB\_ICONERROR);

}

else {

wsprintf(out, L"Title: %s\r\n\r\nBody: %s", title, body);

SetWindowText(hOut, out);

}

}

break;

}

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProcW(hWnd, msg, wp, lp);

}

}

void AddMenus(HWND hWnd) {

hMenu = CreateMenu();

HMENU hFileMenu = CreateMenu();

HMENU hSubMenu = CreateMenu();

AppendMenu(hSubMenu, MF\_STRING, NULL, L"Submenu item");

AppendMenu(hFileMenu, MF\_STRING, FILE\_MENU\_NEW, L"New");

AppendMenu(hFileMenu, MF\_POPUP, (UINT)hSubMenu, L"Open submenu");

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

AppendMenu(hFileMenu, MF\_STRING, FILE\_MENU\_EXIT, L"Exit");

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hFileMenu, L"File");

AppendMenu(hMenu, MF\_STRING, NULL, L"Help");

SetMenu(hWnd, hMenu);

}

void AddControls(HWND hWnd) {

CreateWindow(L"static", L"Enter the title of a to-do:", WS\_VISIBLE | WS\_CHILD, 50, 50, 210, 38, hWnd, NULL, NULL, NULL);

hTitle = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE, 230, 50,250, 20, hWnd, NULL, NULL, NULL);

CreateWindow(L"static", L"Enter the body of a to-do:", WS\_VISIBLE | WS\_CHILD, 50, 150, 210, 38, hWnd, NULL, NULL, NULL);

hBody = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE, 230, 150, 250, 120, hWnd, NULL, NULL, NULL);

CreateWindow(L"button", L"Add to-do", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 50, 300, 98, 38, hWnd, (HMENU)GENERATE\_BUTTON, NULL, NULL);

hOut = CreateWindow(L"edit", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE | ES\_READONLY, 600, 50, 700, 400, hWnd, NULL, NULL, NULL);

}