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 №2

на тему

Расширенное использование оконного интерфейса Win32 и GDI.

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 создании графического приложения с применением Win32 API и GDI, обладающего расширенными возможностями для взаимодействия с оконным интерфейсом. В процессе выполнения лабораторной работы будет разработано графическое приложение, способное записывать заметки пользователя, предоставлять пользователю возможность помечать заметки как выполненные, а также изменять цветовое оформление приложения.

## 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.

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.

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 и других задач.

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

– GDI (Graphics Device Interface): GDI — это подсистема Windows для рисования графики на экране и на печать. В данном приложении функции GDI, такие как CreateSolidBrush, использовались для изменения цвета фона окна.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения данной лабораторной работы был добавлен дополнительный функционал к приложению для записи списка дел. Возможно отображение нескольких заметок одновременно (Рисунок 1).

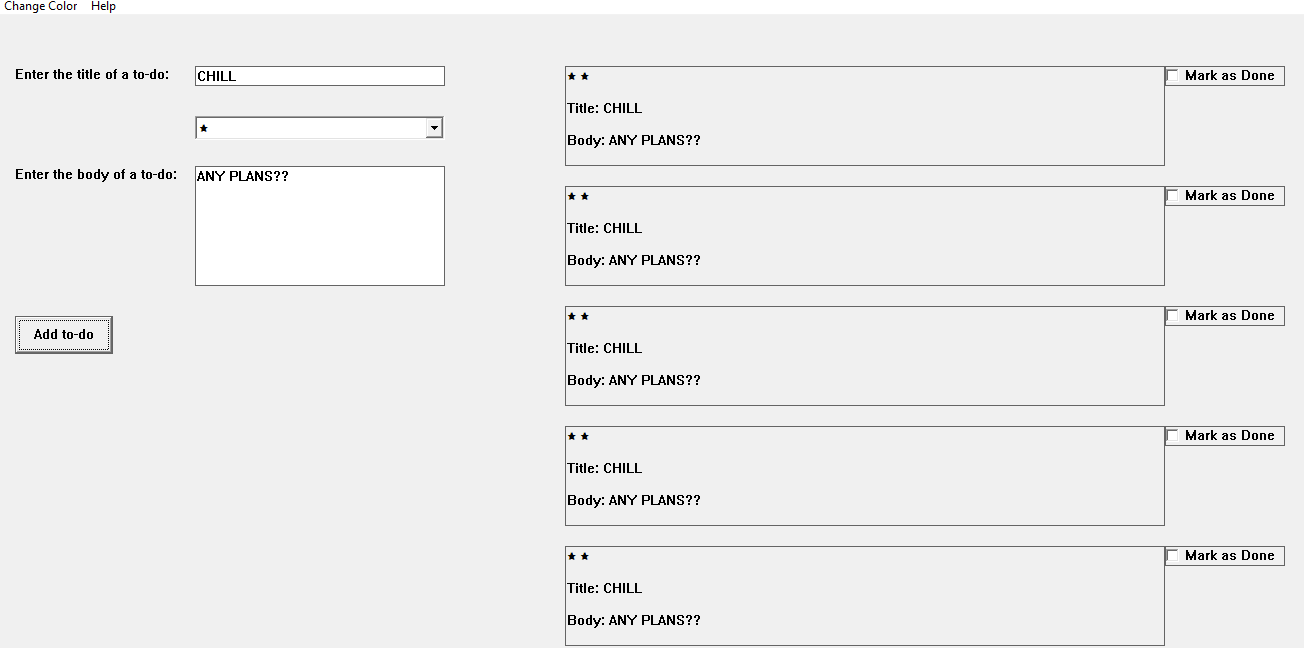
****

Рисунок 1 - Главное окно

Добавлена возможность изменять фоновой цвет приложения из предложенных (Рисунок 2).

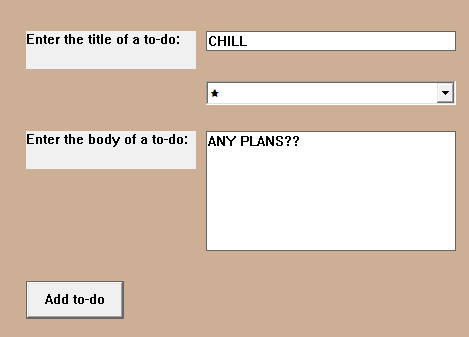


Рисунок 2 ─ Измененное цветовое оформление

Появилась опция отметить заметку как выполненную, нажав на соответствующий элемент. (Рисунок 3).

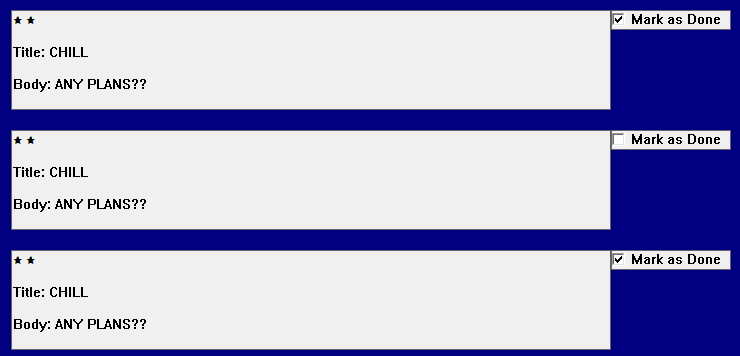


Рисунок 3 – Заметка в поле для записей

Выход из приложения доступен из контекстного меню (Рисунок 4).

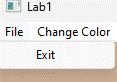


Рисунок 4 – Выход в контекстном меню

## ВЫВОДЫ

В ходе данной работы было разработано оконное приложение с использованием Win32 API, а также использованы библиотеки GDI и механизм перехвата сообщений (winhook). Приложение создано с целью управления заметками, позволяя пользователям добавлять, просматривать и редактировать записи. Пользовательский интерфейс предоставляет удобные инструменты для взаимодействия с заметками, включая ввод и просмотр текста. Таким образом, данная лабораторная работа не только позволила ознакомиться с основами программирования в Win32 API и создать оконное приложение, специально адаптированное для работы с заметками, но также включала в себя расширенное использование библиотеки GDI для формирования графического интерфейса и механизм перехвата сообщений (winhook) для обработки различных видов оконных сообщений и расширения функциональности приложения.

## 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 ─ Дата доступа 24.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

Листинг 1 – Файл Lab2.cpp

#include <windows.h>

#include <stdlib.h>

#define FILE\_MENU\_NEW 1

#define FILE\_MENU\_OPEN 2

#define FILE\_MENU\_EXIT 3

#define CHANGE\_COLOR\_MENU 4

#define GENERATE\_BUTTON 5

#define CHECKBOX\_BASE\_ID 1000

#define COMBO\_BOX\_ID 6

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

LRESULT CALLBACK KeyboardHook(int nCode, WPARAM wParam, LPARAM lParam);

void AddMenus(HWND);

void AddControls(HWND);

void ChangeBackgroundColor(HWND, COLORREF);

void SendToDoWithEmoji(HWND);

void SimulateButtonClick(HWND hWnd);

HWND hTitle, hBody, hAddButton, hComboBox;

HMENU hMenu;

int yPos = 50;

int checkboxID = CHECKBOX\_BASE\_ID;

COLORREF backgroundColor = RGB(255, 255, 255);

HHOOK hHook = NULL;

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 CHANGE\_COLOR\_MENU:

if (backgroundColor == RGB(255, 255, 255)) {

backgroundColor = RGB(205, 175, 149);

}

else if (backgroundColor == RGB(205, 175, 149)) {

backgroundColor = RGB(0, 0, 128);

}

else {

backgroundColor = RGB(255, 255, 255);

}

ChangeBackgroundColor(hWnd, backgroundColor);

break;

case GENERATE\_BUTTON:

SendToDoWithEmoji(hWnd);

break;

}

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

// Install the keyboard hook

hHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardHook, GetModuleHandle(NULL), 0);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

LRESULT CALLBACK KeyboardHook(int nCode, WPARAM wParam, LPARAM lParam) {

if (nCode >= 0) {

if (wParam == WM\_KEYDOWN) {

KBDLLHOOKSTRUCT\* keyInfo = (KBDLLHOOKSTRUCT\*)lParam;

if (keyInfo->vkCode == VK\_RETURN) {

// Simulate a button click when Enter is pressed

SimulateButtonClick(GetActiveWindow());

}

}

}

return CallNextHookEx(hHook, nCode, wParam, lParam);

}

void AddMenus(HWND hWnd) {

hMenu = CreateMenu();

HMENU hFileMenu = CreateMenu();

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

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

AppendMenu(hMenu, MF\_STRING, CHANGE\_COLOR\_MENU, L"Change Color");

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, 170, 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, 170, 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);

hComboBox = CreateWindow(L"combobox", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | CBS\_DROPDOWNLIST, 230, 100, 250, 200, hWnd, (HMENU)COMBO\_BOX\_ID, NULL, NULL);

SendMessage(hComboBox, CB\_ADDSTRING, 0, (LPARAM)L"★");

SendMessage(hComboBox, CB\_ADDSTRING, 0, (LPARAM)L"©");

SendMessage(hComboBox, CB\_ADDSTRING, 0, (LPARAM)L"⚠");

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

}

void ChangeBackgroundColor(HWND hWnd, COLORREF color) {

SetClassLongPtr(hWnd, GCLP\_HBRBACKGROUND, (LONG\_PTR)CreateSolidBrush(color));

RedrawWindow(hWnd, NULL, NULL, RDW\_ERASE | RDW\_INVALIDATE);

}

void SendToDoWithEmoji(HWND hWnd) {

wchar\_t title[31], body[250], out[300];

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, 250);

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 {

int selectedIndex = SendMessage(hComboBox, CB\_GETCURSEL, 0, 0);

if (selectedIndex != CB\_ERR) {

wchar\_t emoji[2] = { 0 };

SendMessage(hComboBox, CB\_GETLBTEXT, selectedIndex, (LPARAM)emoji);

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

}

else {

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

}

HWND hOut = CreateWindow(L"edit", out, WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_MULTILINE | ES\_READONLY, 600, yPos, 600, 100, hWnd, NULL, NULL, NULL);

CreateWindow(L"button", L"Mark as Done", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | BS\_AUTOCHECKBOX, 1200, yPos, 120, 20, hWnd, (HMENU)checkboxID, NULL, NULL);

yPos += 120;

checkboxID++;

}

}

}

void SimulateButtonClick(HWND hWnd) {

SendMessage(hWnd, WM\_COMMAND, MAKEWPARAM(GENERATE\_BUTTON, BN\_CLICKED), (LPARAM)hAddButton);

}