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 Win 32 и GDI. Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений (winhook)

Выполнил: студент группы 153503

Бобко Илья Витальевич

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[2 Результаты выполнения лабораторной работы 6](#_Toc146728179)

[Список использованных источников 8](#_Toc146728181)

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

## 1 ЦЕЛЬ РАБОТЫ

Цель работы ­ изучить и применить на практике знания о расширенном использовании Win32 и GDI для создания приложений на Windows, научиться формировать сложные изображения, обрабатывать различные сообщения, а также изучить механизм перехвата сообщений с использованием winhook. Для этого мы дополним нашу имеющийся калькулятор корней квадратного уравнения, добавив кнопку выбора фона а также альтернативу кнопке - нажатие кнопки “b”.

## 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, управления таймерами 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, перерисовка и таймер.

– GDI (Graphics Device Interface): GDI — это подсистема Windows для рисования графики на экране и на печать. В данном приложении функции GDI, такие как Ellipse и Rectangle, использовались для рисования объектов на окне.

– Таймеры: для обновления позиции объекта и перерисовки окна с определенной частотой были использованы таймеры с помощью функций SetTimer и KillTimer. Это обеспечило анимацию в приложении.

– Меню: создано пользовательское меню с использованием функций CreateMenu и AppendMenu. Это меню позволяет пользователю выбирать тип фигуры, направление движения и цвет фигуры, что делает его важным элементом графического интерфейса пользователя (GUI).

– Выбор параметров: пользователь имеет возможность настраивать различные параметры анимации, такие как тип траектории, скорость, тип фигуры и цвет. Это демонстрирует концепцию интерактивности в приложении.

– 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 было разработано дополнение к оконному приложению, вычисляющему корни квадратного уравнения. (рисунок 1).

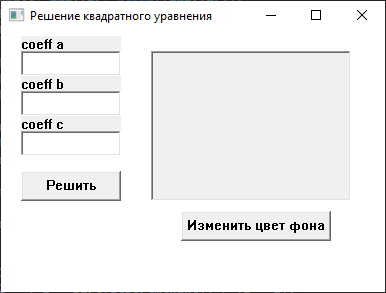


Рисунок 1 - Главный экран

Пользователь может изменить цвет фона на любой другой цвет из палитры по средствам нажатия соответствующей кнопке “Изменить цвет фона” или по нажатию кнопки “b” (рисунок 2).

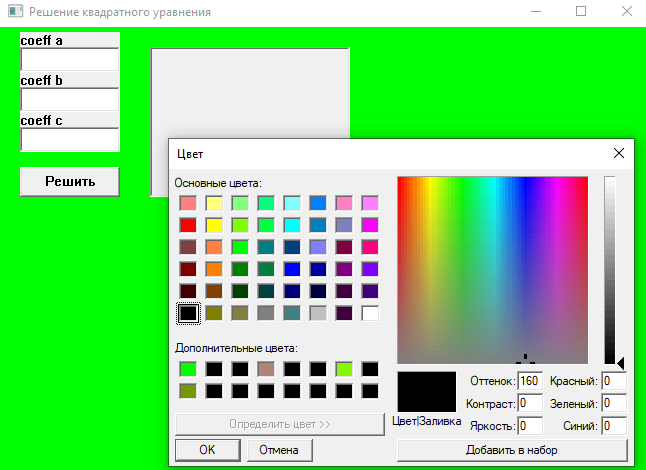


Рисунок 2 ─ Изменение фона

## ВЫВОДЫ

В результате выполнения работы было создано оконное приложение, способное открывать, редактировать и сохранять текстовые документы, а также осуществлять выделение и копирование текста в буфер обмена. В ходе обработки основных оконных сообщений были реализованы необходимые функции для работы с текстовыми документами. Была создана область для отображения текста с возможностью прокрутки, а также добавлены элементы управления для выполнения операций над текстовыми файлами. Таким образом, лабораторная работа позволила ознакомиться с основами программирования в Win32 API и создать оконное приложение с базовой функциональностью, отражающей основные принципы работы с окнами и обработкой сообщений.

## 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)

## Листинг кода

Листинг 1 – Файл source.cpp

#include <windows.h>

#include <cmath>

#include <string>

HWND hAEdit, hBEdit, hCEdit, hResultEdit, hButtonSolve, hButtonChangeColor, hButton;

HINSTANCE hInstance;

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

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

HHOOK hKeyboardHook = NULL;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

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

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"QuadraticEquationSolver";

wcex.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (!RegisterClassEx(&wcex)) {

MessageBox(NULL, L"Не удалось зарегистрировать класс окна.", L"Ошибка", MB\_ICONERROR);

return 1;

}

HWND hWnd = CreateWindow(L"QuadraticEquationSolver", L"Решение квадратного уравнения",

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 400, 300, NULL, NULL, hInstance, NULL);

if (!hWnd) {

MessageBox(NULL, L"Не удалось создать окно.", L"Ошибка", MB\_ICONERROR);

return 1;

}

CreateWindow(L"static", L"coeff a", WS\_VISIBLE | WS\_CHILD, 20, 5, 100, 25, hWnd, NULL, NULL, NULL);

CreateWindow(L"static", L"coeff b", WS\_VISIBLE | WS\_CHILD, 20, 45, 100, 25, hWnd, NULL, NULL, NULL);

CreateWindow(L"static", L"coeff c", WS\_VISIBLE | WS\_CHILD, 20, 85, 100, 25, hWnd, NULL, NULL, NULL);

hAEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_NUMBER,

20, 20, 100, 25, hWnd, NULL, hInstance, NULL);

hBEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_NUMBER,

20, 60, 100, 25, hWnd, NULL, hInstance, NULL);

hCEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_NUMBER,

20, 100, 100, 25, hWnd, NULL, hInstance, NULL);

hButton = CreateWindow(L"BUTTON", L"Решить", WS\_CHILD | WS\_VISIBLE,

20, 140, 100, 30, hWnd, (HMENU)1, hInstance, NULL);

hResultEdit = CreateWindowEx(WS\_EX\_CLIENTEDGE, L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_READONLY,

150, 20, 200, 150, hWnd, NULL, hInstance, NULL);

hButtonSolve = CreateWindow(L"BUTTON", L"Решить", WS\_CHILD | WS\_VISIBLE,

20, 140, 100, 30, hWnd, (HMENU)1, hInstance, NULL);

hButtonChangeColor = CreateWindow(L"BUTTON", L"Изменить цвет фона", WS\_CHILD | WS\_VISIBLE,

180, 180, 150, 30, hWnd, (HMENU)2, hInstance, NULL);

hInstance = hInst;

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

hKeyboardHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardHook, hInstance, 0);

if (!hKeyboardHook) {

MessageBox(NULL, L"Не удалось установить глобальный хук клавиш.", L"Ошибка", MB\_ICONERROR);

return 1;

}

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(hKeyboardHook);

return (int)msg.wParam;

}

void ChangeWindowBackgroundColor(HWND hWnd) {

static COLORREF customColor = RGB(255, 255, 255);

CHOOSECOLOR cc = { sizeof(CHOOSECOLOR) };

cc.hwndOwner = hWnd;

cc.lpCustColors = &customColor;

cc.Flags = CC\_FULLOPEN | CC\_RGBINIT;

if (ChooseColor(&cc)) {

customColor = cc.rgbResult;

HBRUSH hBrush = CreateSolidBrush(customColor);

SetClassLongPtr(hWnd, GCLP\_HBRBACKGROUND, (LONG\_PTR)hBrush);

InvalidateRect(hWnd, NULL, TRUE);

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

}

}

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

if (nCode >= 0 && wParam == WM\_KEYDOWN) {

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

if (kbStruct->vkCode == 'B') {

ChangeWindowBackgroundColor(GetForegroundWindow());

}

}

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

}

void SolveQuadraticEquation(HWND hWnd) {

wchar\_t aBuffer[100], bBuffer[100], cBuffer[100];

GetWindowText(hAEdit, aBuffer, sizeof(aBuffer));

GetWindowText(hBEdit, bBuffer, sizeof(bBuffer));

GetWindowText(hCEdit, cBuffer, sizeof(cBuffer));

double a = \_wtof(aBuffer);

double b = \_wtof(bBuffer);

double c = \_wtof(cBuffer);

double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0) {

SetWindowText(hResultEdit, L"Нет действительных корней");

}

else if (discriminant == 0) {

double root = -b / (2 \* a);

SetWindowText(hResultEdit, std::to\_wstring(root).c\_str());

}

else {

double root1 = (-b + sqrt(discriminant)) / (2 \* a);

double root2 = (-b - sqrt(discriminant)) / (2 \* a);

SetWindowText(hResultEdit, (std::to\_wstring(root1) + L", " + std::to\_wstring(root2)).c\_str());

}

}

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

switch (message) {

case WM\_COMMAND:

if (LOWORD(wParam) == 1) {

SolveQuadraticEquation(hWnd);

}

else if (LOWORD(wParam) == 2) {

ChangeWindowBackgroundColor(hWnd);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

break;

}

return 0;

}