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. 153504
Прескурел Я.Ю.

Проверил:
Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

1 Цели работы 3
2 Краткие теоретические сведения 4
3 Полученные результаты 5
Выводы 6
Список использованных источников 7
Приложение А 8

1 ЦЕЛИ РАБОТЫ

Изучить основы программирования в Win 32 API. Создать оконное приложение Win 32 с минимальной достаточной функциональностью. Реализовать обработку основных оконных сообщений. Разработать оконное приложение, которое позволяет пользователю рисовать и редактировать графические фигуры (круги, прямоугольники) с помощью мыши и клавиш клавиатуры.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Оконное приложение Win32 API – это приложение Windows, которое использует библиотеки Win32 API для создания и управления окнами и элементами пользовательского интерфейса. Минимально достаточное оконное приложение на Win32 API обычно состоит из окна, которое может быть открыто, закрыто и перерисовано.

Для создания окна в Win32 API, программист должен зарегистрировать класс окна и создать экземпляр этого класса. Зарегистрированный класс содержит информацию о том, как окно должно выглядеть и какие обработчики событий должны быть вызваны для обработки сообщений, отправленных в окно.

Обработка основных оконных сообщений включает в себя обработку сообщений, таких как WM_CREATE, WM_PAINT, WM_COMMAND, WM_RBUTTONDOWN, WM_LBUTTONDOWN, WM_KEYDOWN WM_CLOSE и WM_DESTROY. Сообщение WM_CREATE отправляется системой в окно при создании окна, сообщение WM_PAINT отправляется при необходимости перерисовки окна, сообщение WM_COMMAND отправляется при действиях с элементами управления(кнопки, меню), сообщение WM_RBUTTONDOWN(WM_LBUTTONDOWN) отправляется при нажатии правой(левой) кнопки мыши, сообщение WM_KEYDOWN отправляется при нажатии клавиши на клавиатуре, сообщение WM_CLOSE отправляется, когда пользователь закрывает окно, а сообщение WM_DESTROY отправляется, когда окно должно быть уничтожено.

Реализация обработки этих сообщений в приложении Win32 API обычно осуществляется через обработчики сообщений оконной процедуры, которые определены программистом. Оконная процедура приложения может быть определена как статическая функция в коде приложения, которая будет вызвана при каждом получении сообщения окном.

Также возможно использование других функций Win32 API, таких как CreateWindow, ShowWindow и UpdateWindow, чтобы создавать, отображать и обновлять окна приложения.

Результатом использования Win32 API являются интуитивно понятные и функциональные приложения, которые позволяют пользователям взаимодействовать с компьютером посредством элементов управления, таких как текст, графика, кнопки и поля ввода. Тем не менее, следует учитывать, что в различных контекстах некоторые термины в документации Windows могут иметь разные значения, например, слово "служба" может относиться к вызываемой подпрограмме, драйверу устройства или к обслуживающему процессу.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В ходе выполнения лабораторной работы было реализовано оконное приложение, которое позволяет пользователю рисовать и редактировать графические фигуры (круги, прямоугольники) с помощью мыши и клавиш клавиатуры. Результат работы программы показан на рисунке 3.1.

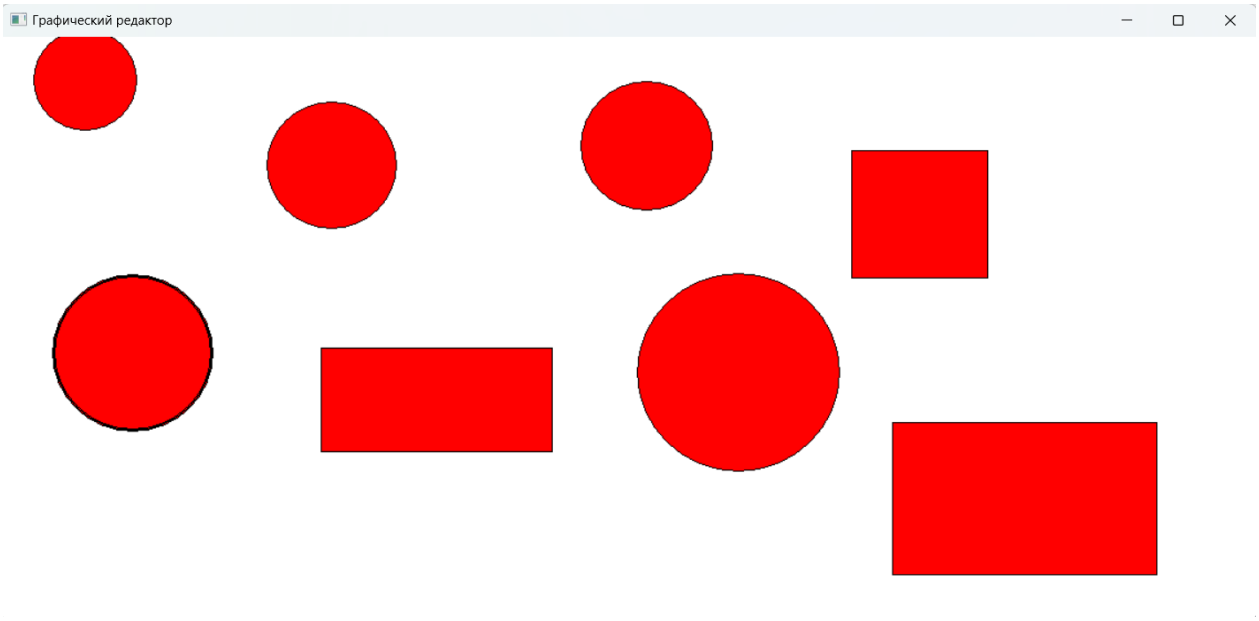

Рисунок 3.1 – Результат работы программы

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены основы программирования с использованием Win32 API, что позволило создать оконное приложение с необходимой функциональностью. Были применены функции, классы и методы Win32 API, такие как RegisterClass, CreateWindow, GetMessage, DefWindowProc, которые обеспечивают взаимодействие с операционной системой Windows и позволяют эффективно управлять окном и обрабатывать события. В результате было разработано приложение, которое успешно обрабатывает основные оконные сообщения и позволяет пользователю работать с графическими фигурами, такими как круги и прямоугольники, с помощью устройств ввода мыши и клавиатуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Марапулец Ю. В. Системное программирование в WIN API. Учебное пособие, 2021. – 31 с

[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

Lab1.cpp

```
#include <windows.h>
#include <windowsx.h>
#include <cmath>

const wchar_t CLASS_NAME[] = L"Графический редактор";

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);

#define SHAPE_NONE 0
#define SHAPE_CIRCLE 1
#define SHAPE_RECTANGLE 2

int shape = SHAPE_NONE;
bool drawing = false;
int startX = 0;
int startY = 0;
int endX = 0;
int endY = 0;

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR
pCmdLine, int nCmdShow)
{
    WNDCLASS wc = {};
    wc.lpfnWndProc = WindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = CLASS_NAME;
    RegisterClass(&wc);

    HWND hwnd = CreateWindowEx(
        0,
        CLASS_NAME,
        L"Графический редактор",
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        NULL
    );

    if (hwnd == NULL)
    {
        return 0;
    }

    ShowWindow(hwnd, nCmdShow);

    MSG msg = {};
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
```

```
        DispatchMessage(&msg);
    }

    return 0;
}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    switch (uMsg)
    {
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;

    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hwnd, &ps);

        if (drawing)
        {
            HBRUSH hBrush = CreateSolidBrush(RGB(255, 0, 0));
            HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

            switch (shape)
            {
            case SHAPE_CIRCLE:
            {
                int radius = (int)sqrt(pow(endX - startX, 2) + pow(endY -
startY, 2));
                int centerX = startX;
                int centerY = startY;

                Ellipse(hdc, centerX - radius, centerY - radius, centerX +
radius, centerY + radius);
            }
            break;

            case SHAPE_RECTANGLE:
            {
                Rectangle(hdc, startX, startY, endX, endY);
            }
            break;
            }

            SelectObject(hdc, hOldBrush);
            DeleteObject(hBrush);
        }

        EndPaint(hwnd, &ps);
    }
    return 0;

    case WM_LBUTTONDOWN:
    {
        startX = GET_X_LPARAM(lParam);
        startY = GET_Y_LPARAM(lParam);

        drawing = true;
```

```
    }
    return 0;

    case WM_MOUSEMOVE:
    {
        if (drawing)
        {
            endX = GET_X_LPARAM(lParam);
            endY = GET_Y_LPARAM(lParam);

            InvalidateRect(hwnd, NULL, TRUE);
        }
    }
    return 0;

    case WM_LBUTTONUP:
    {
        if (drawing)
        {
            endX = GET_X_LPARAM(lParam);
            endY = GET_Y_LPARAM(lParam);

            drawing = false;

            InvalidateRect(hwnd, NULL, TRUE);
        }
    }
    return 0;

    case WM_KEYDOWN:
    {
        switch (wParam)
        {
        case 'C':
            shape = SHAPE_CIRCLE;
            break;

        case 'R':
            shape = SHAPE_RECTANGLE;
            break;

        case VK_ESCAPE:
            shape = SHAPE_NONE;
            InvalidateRect(hwnd, NULL, TRUE);
            break;
        }
    }
    return 0;
    }

    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}
```