Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Программирование в среде WINDOWS»

Отчет по лабораторным работам № 1-4

Выполнил:
студент группы ИУ5-43
Терентьев В.О.

Проверил:
преподаватель
Аксенова М.В.

Москва, 2020 г.

1. Лабораторная работа № 1

1.1. Постановка задачи

Составьте программу, в которой в главное окно бледно-розового цвета выводятся три концентрических эллипса с размерами 350x250, 250x150 и 150x50 пикселов. Внешний эллипс нарисуйте толстым (6-8 пиксела) черным пером, средний – синим, а внутренний – желтым. Эллипсы должны быть прозрачными. Нажатие левой клавиши – изменение толщины пера у эллипса, над которым находится курсор.

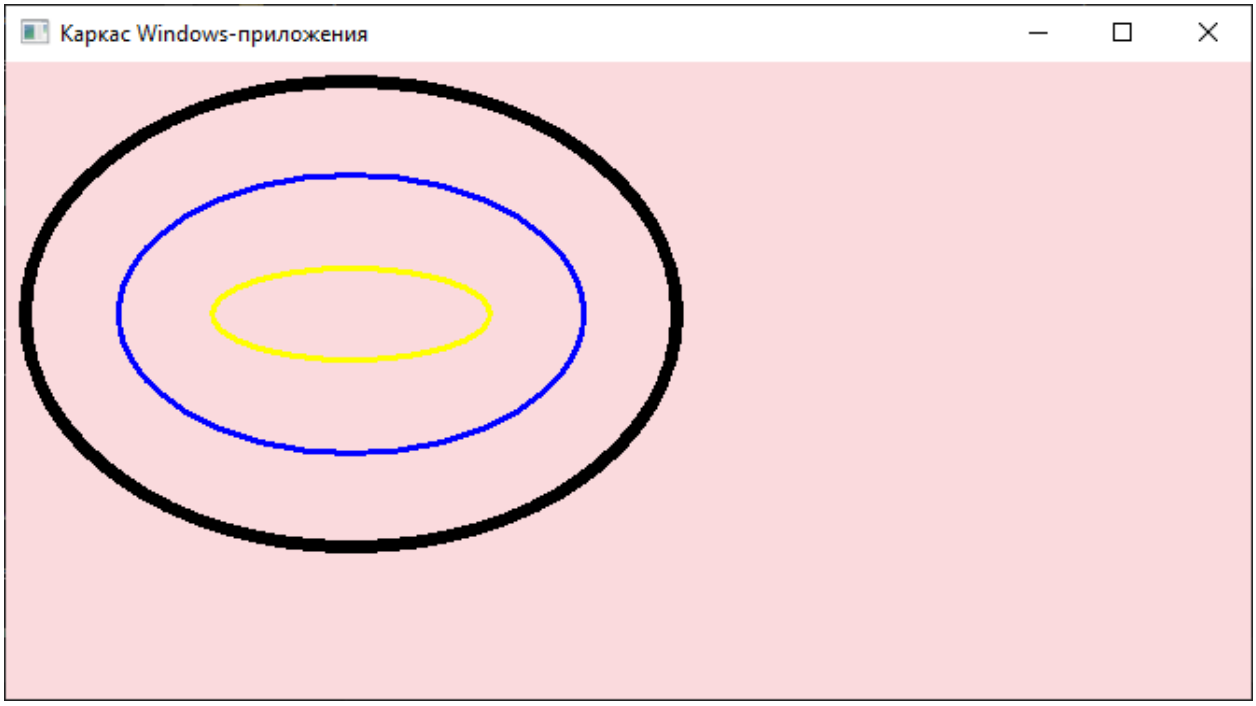
1.2. Текст программы

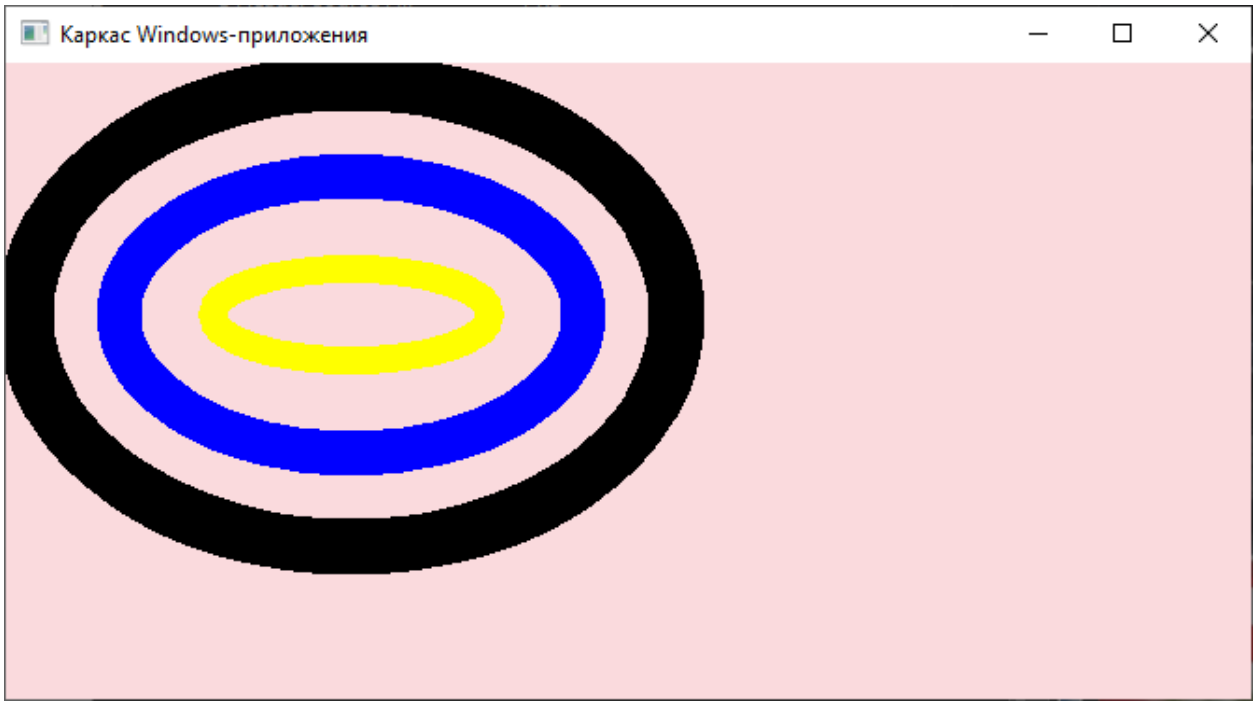
```
#include <Windows.h>
#include <tchar.h>
#include <math.h>

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

TCHAR WinName[] = _T("MainFrame");

HDC hdc;

int WINAPI _tWinMain(HINSTANCE This,         // Дескриптор текущего приложения
    HINSTANCE Prev,     // В современных системах всегда 0
    LPTSTR cmd,         // Командная строка
    int mode)       // Режим отображения окна
{
    HWND hWnd;      // Дескриптор главного окна программы
    MSG msg;        // Структура для хранения сообщения
    WNDCLASS wc;    // Класс окна

    // Определение класса окна
    wc.hInstance = This;
    wc.lpszClassName = WinName;                 // Имя класса окна
    wc.lpfnWndProc = WndProc;                   // Функция окна
    wc.style = CS_HREDRAW | CS_VREDRAW;             // Стиль окна
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);         // Стандартная иконка
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Стандартный курсор
    wc.lpszMenuName = NULL;                     // Нет меню
    wc.cbClsExtra = 0;                      // Нет дополнительных данных класса
    wc.cbWndExtra = 0;                      // Нет дополнительных данных окна
    //wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);    // Заполнение окна белым
цветом
    wc.hbrBackground = CreateSolidBrush(RGB(250, 218, 221));


    // Регистрация класса окна
    if (!RegisterClass(&wc)) return 0;

    // Создание окна
```

```
    hWnd = CreateWindow(WinName,            // Имя класса окна
        _T("Каркас Windows-приложения"),        // Заголовок окна
        WS_OVERLAPPEDWINDOW,        // Стиль окна
        CW_USEDEFAULT,              // x
        CW_USEDEFAULT,              // y     Размеры окна
        CW_USEDEFAULT,              // width
        CW_USEDEFAULT,              // Height
        HWND_DESKTOP,               // Дескриптор родительского окна
        NULL,                       // Нет меню
        This,                       // Дескриптор приложения
        NULL);                  // Дополнительной информации нет

    ShowWindow(hWnd, mode);                 // Показать окно

    // Цикл обработки сообщений
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);         // Функция трансляции кодов нажатой клави
ши
        DispatchMessage(&msg);      // Посылает сообщение функции WndProc()
    }
    return 0;
}

// Оконная функция вызывается операционной системой
// и получает сообщения из очереди для данного приложения

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

    static int a=7, b=3, c = 3, x, y;

    switch (message)         // Обработчик сообщений
    {

    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        hdc = BeginPaint(hWnd, &ps);
        HPEN hPen1 = CreatePen(PS_SOLID, a, RGB(0, 0, 0));
        SelectObject(hdc, hPen1);
        HBRUSH hBrush = (HBRUSH)GetStockObject(HOLLOW_BRUSH);
        SelectObject(hdc, hBrush);
        Ellipse(hdc, 10, 10, 360, 260);

        HPEN hPen2 = CreatePen(PS_SOLID, b, RGB(0, 0, 255));
        SelectObject(hdc, hPen2);
        Ellipse(hdc, 60, 60, 310, 210);

        HPEN hPen3 = CreatePen(PS_SOLID, c, RGB(255, 255, 0));
        SelectObject(hdc, hPen3);
```

```
            Ellipse(hdc, 110, 110, 260, 160);


            EndPaint(hWnd, &ps);
            break;
        }

        case WM_LBUTTONUP:
        {
            x = LOWORD(lParam) - 185;
            y = -HIWORD(lParam) + 135;
            if ((((x * x) / (75.0 * 75)) + ((y * y) / (25.0 * 25))) <= 1)
                c++;
            else if (((x * x) / (125.0 * 125) + (y * y) / (75.0 * 75)) <= 1)
                b++;
            else if (((x * x) / (175.0 * 175) + (y * y) / (125.0 * 125)) <= 1)
                a++;
            InvalidateRect(hWnd, NULL, true);
            break;
        }
        case WM_DESTROY:
            PostQuitMessage(0);
            break;          // Завершение программы
        default:            // Обработка сообщения по умолчанию
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
    }
```

1.3. Анализ результатов





2. Лабораторная работа № 2

2.1. Постановка задачи

Сформировать два приложения, которые открывают по одному окну. В окне 1 по щелчку левой клавиши мыши: при помощи FindWindow() найти дескриптор окна 2. Выдать сообщение об этом. Если операция неудачная, то выдать сообщение об этом. При помощи функции SendMessage() и поля WPARAM передать свой дескриптор второму окну. Выдать сообщение об этом. В окне 2: при получении сообщения WM_USER+1 (левая клавиша) нарисовать прямоуголиник (при повторном получении сообщения WM_USER+1 цвет, размер, координаты изменяются). При получении сообщения WM_USER+2 (правая клавиша) закрыть приложение 1.

2.2. Текст программы

Приложение 1

```
#include <Windows.h>
#include <tchar.h>
#include <string>

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

TCHAR WinName[] = _T("MainFrame");

int WINAPI _tWinMain(HINSTANCE This,         // Дескриптор текущего приложения
    HINSTANCE Prev,     // В современных системах всегда 0
    LPTSTR cmd,         // Командная строка
    int mode)       // Режим отображения окна
{
    HWND hWnd;      // Дескриптор главного окна программы
    MSG msg;        // Структура для хранения сообщения
    WNDCLASS wc;    // Класс окна
    // Определение класса окна
    wc.hInstance = This;
    wc.lpszClassName = WinName;                 // Имя класса окна
    wc.lpfnWndProc = WndProc;                   // Функция окна
    wc.style = CS_HREDRAW | CS_VREDRAW;             // Стиль окна
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);         // Стандартная иконка
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Стандартный курсор
    wc.lpszMenuName = NULL;                     // Нет меню
    wc.cbClsExtra = 0;                      // Нет дополнительных данных класса
    wc.cbWndExtra = 0;                      // Нет дополнительных данных окна
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);  // Заполнение окна белым цвет
ом


    // Регистрация класса окна
    if (!RegisterClass(&wc)) return 0;
```

```
    // Создание окна
    hWnd = CreateWindow(WinName,            // Имя класса окна
        _T("Каркас Windows-приложения"),        // Заголовок окна
        WS_OVERLAPPEDWINDOW,        // Стиль окна
        CW_USEDEFAULT,              // x
        CW_USEDEFAULT,              // y     Размеры окна
        CW_USEDEFAULT,              // width
        CW_USEDEFAULT,              // Height
        HWND_DESKTOP,               // Дескриптор родительского окна
        NULL,                       // Нет меню
        This,                       // Дескриптор приложения
        NULL);                  // Дополнительной информации нет

    ShowWindow(hWnd, mode);                 // Показать окно

    // Цикл обработки сообщений
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);         // Функция трансляции кодов нажатой клави
ши
        DispatchMessage(&msg);      // Посылает сообщение функции WndProc()
    }
    return 0;
}

// Оконная функция вызывается операционной системой
// и получает сообщения из очереди для данного приложения

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HWND a;
    switch (message)         // Обработчик сообщений
    {
    case WM_LBUTTONUP:
    {
        if (!a) {
            a = FindWindow(NULL, TEXT("Окно 2"));
            if (!a)
                MessageBox(hWnd, TEXT("Окно не найдено"), TEXT("Error"), MB_OK);
            else
            {
                MessageBox(hWnd, TEXT("Окно найдено"), TEXT("Message"), MB_OK);
            }
        }
        else {
            LRESULT b = SendMessage(a, WM_USER + 1, (WPARAM)hWnd, lParam);
            //MessageBox(hWnd, TEXT("Сообщение отправлено"), TEXT("Message"), MB_
OK);
        }
    }
    break;
```

```
    case WM_RBUTTONUP:
    {
        LRESULT b = SendMessage(a, WM_USER + 2, (WPARAM)hWnd, lParam);
    }
    break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;          // Завершение программы
    default:            // Обработка сообщения по умолчанию
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

Приложение 2

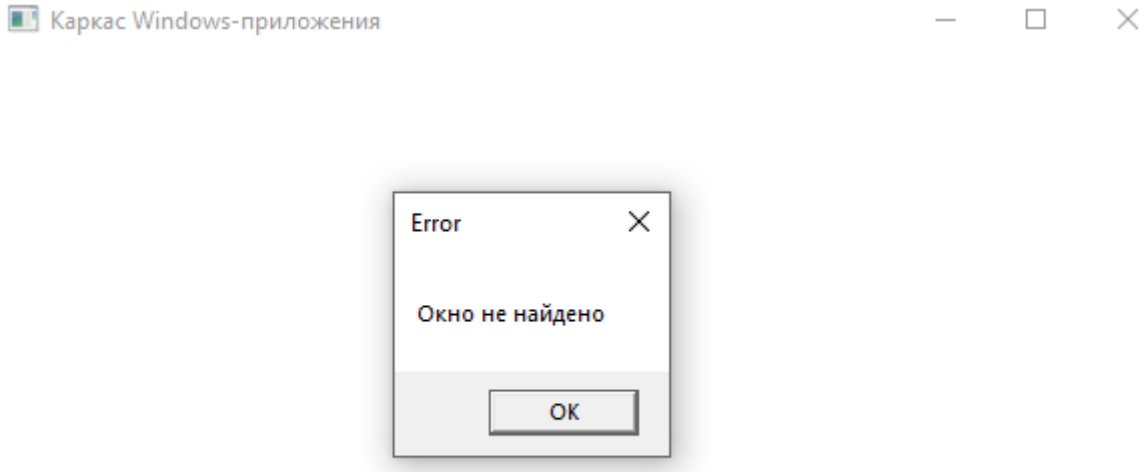
```
#include <Windows.h>
#include <tchar.h>

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

TCHAR WinName[] = _T("MainFrame");

int WINAPI _tWinMain(HINSTANCE This,         // Дескриптор текущего приложения
    HINSTANCE Prev,     // В современных системах всегда 0
    LPTSTR cmd,         // Командная строка
    int mode)       // Режим отображения окна
{
    HWND hWnd;      // Дескриптор главного окна программы
    MSG msg;        // Структура для хранения сообщения
    WNDCLASS wc;    // Класс окна
    // Определение класса окна
    wc.hInstance = This;
    wc.lpszClassName = WinName;                 // Имя класса окна
    wc.lpfnWndProc = WndProc;                   // Функция окна
    wc.style = CS_HREDRAW | CS_VREDRAW;             // Стиль окна
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);         // Стандартная иконка
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Стандартный курсор
    wc.lpszMenuName = NULL;                     // Нет меню
    wc.cbClsExtra = 0;                      // Нет дополнительных данных класса
    wc.cbWndExtra = 0;                      // Нет дополнительных данных окна
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);  // Заполнение окна белым цвет
ом


    // Регистрация класса окна
    if (!RegisterClass(&wc)) return 0;

    // Создание окна
    hWnd = CreateWindow(WinName,            // Имя класса окна
```

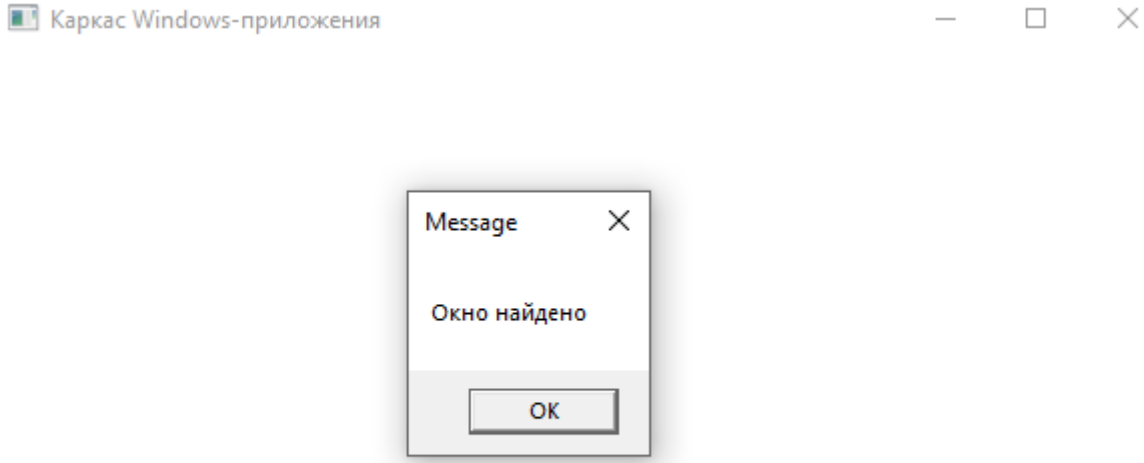
```
        _T("Окно 2"),       // Заголовок окна
        WS_OVERLAPPEDWINDOW,        // Стиль окна
        CW_USEDEFAULT,              // x
        CW_USEDEFAULT,              // y     Размеры окна
        CW_USEDEFAULT,              // width
        CW_USEDEFAULT,              // Height
        HWND_DESKTOP,               // Дескриптор родительского окна
        NULL,                       // Нет меню
        This,                       // Дескриптор приложения
        NULL);                  // Дополнительной информации нет

    ShowWindow(hWnd, mode);                 // Показать окно

    // Цикл обработки сообщений
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);         // Функция трансляции кодов нажатой клави
ши
        DispatchMessage(&msg);      // Посылает сообщение функции WndProc()
    }
    return 0;
}

// Оконная функция вызывается операционной системой
// и получает сообщения из очереди для данного приложения

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int x = 0, r1, r2, r3, r4, r5, r6, r7;
    PAINTSTRUCT ps;
    static HDC hdc;

    switch (message)         // Обработчик сообщений
    {
    case WM_PAINT:
    {
        hdc = BeginPaint(hWnd, &ps);
        if (x == 1) {
            HPEN hPen1 = CreatePen(PS_SOLID, 7, RGB(0, 0, 0));
            SelectObject(hdc, hPen1);
            HBRUSH hBrush = (HBRUSH)GetStockObject(HOLLOW_BRUSH);
            SelectObject(hdc, hBrush);
            Rectangle(hdc, 10, 10, 360, 260);
        }
       else if (x > 1) {
            HPEN hPen1 = CreatePen(PS_SOLID, 7, RGB(r1 % 255, r2 % 255, r3 % 255)
);
            SelectObject(hdc, hPen1);
            HBRUSH hBrush = (HBRUSH)GetStockObject(HOLLOW_BRUSH);
            SelectObject(hdc, hBrush);
            Rectangle(hdc, r4 % 600, r5 % 600, r6 % 600, r7 % 600);
```

```
        }
        EndPaint(hWnd, &ps);
    }
    break;
    case WM_USER+1:
    {
        r1 = rand(); r2 = rand(); r3 = rand(); r4 = rand(); r5 = rand(); r6 = ran
d(); r7 = rand();
        x++;
        InvalidateRect(hWnd, NULL, false);
    }
    break;
    case WM_USER + 2:
    {
        SendMessage((HWND)wParam, WM_SYSCOMMAND, SC_CLOSE, lParam);
        //CloseWindow((HWND)wParam);
    }
    break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;          // Завершение программы
    default:            // Обработка сообщения по умолчанию
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

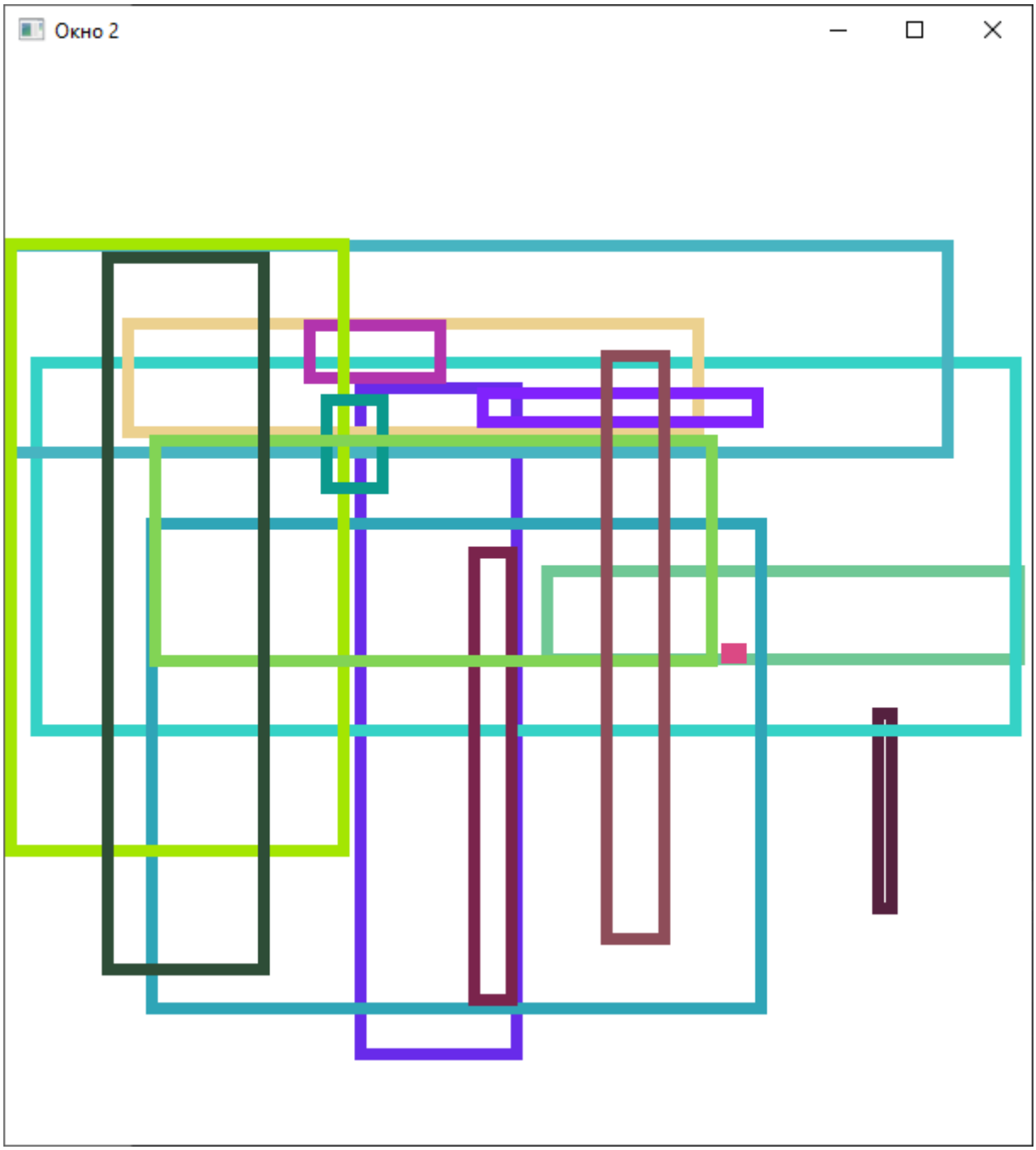
2.3. Анализ результатов

Приложение 1




Каркас Windows-приложения
Message
Окно найдено
OK

Приложение 2



3. Лабораторная работа № 3

3.1. Постановка задачи

Составьте программу "Калькулятор 2".

3.2. Текст программы

```
#pragma comment(linker,"\"/manifestdependency:type='win32' \
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \
processorArchitecture='*' publicKeyToken='6595b64144ccf1df' language='*'\"")

#include <Windows.h>
#include <tchar.h>
#include <string>
#include <atlstr.h>

#define windowWidth 324
#define windowHeight 380
#define buttonWidth 78
#define buttonHeight 50

using namespace std;

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

TCHAR WinName[] = _T("MainFrame");

int WINAPI _tWinMain(HINSTANCE This,         // Дескриптор текущего приложения
    HINSTANCE Prev,     // В современных системах всегда 0
    LPTSTR cmd,         // Командная строка
    int mode)       // Режим отображения окна
{
    HWND hWnd;      // Дескриптор главного окна программы
    MSG msg;        // Структура для хранения сообщения
    WNDCLASS wc;    // Класс окна
    // Определение класса окна
    wc.hInstance = This;
    wc.lpszClassName = WinName;                 // Имя класса окна
    wc.lpfnWndProc = WndProc;                   // Функция окна
    wc.style = CS_HREDRAW | CS_VREDRAW;             // Стиль окна
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);         // Стандартная иконка
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Стандартный курсор
    wc.lpszMenuName = NULL;                     // Нет меню
    wc.cbClsExtra = 0;                      // Нет дополнительных данных класса
    wc.cbWndExtra = 0;                      // Нет дополнительных данных окна
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);  // Заполнение окна белым цвет
ом


    // Регистрация класса окна
```

```
    if (!RegisterClass(&wc)) return 0;

    // Создание окна
    hWnd = CreateWindow(WinName,            // Имя класса окна
        _T("Калькулятор"),      // Заголовок окна
        WS_OVERLAPPEDWINDOW,        // Стиль окна
        CW_USEDEFAULT,              // x
        CW_USEDEFAULT,              // y     Размеры окна
        windowWidth + 16,               // width
        windowHeight + 40,              // Height
        HWND_DESKTOP,               // Дескриптор родительского окна
        NULL,                       // Нет меню
        This,                       // Дескриптор приложения
        NULL);                  // Дополнительной информации нет

    ShowWindow(hWnd, mode);                 // Показать окно

    // Цикл обработки сообщений
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);         // Функция трансляции кодов нажатой клави
ши
        DispatchMessage(&msg);      // Посылает сообщение функции WndProc()
    }
    return 0;
}

// Оконная функция вызывается операционной системой
// и получает сообщения из очереди для данного приложения

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HWND sign, zero, point, equals, one, two, three, plus, four, five, six
, minus, seven, eight, nine, multiplication, division, backspace, clearEntry, cle
ar, hEdit, hStatic;
    static TCHAR buf[256];
    static TCHAR ttt[256];
    //memset(buf, 0, 256);
    static bool refresh = false;
    static int operation[2] = { 0 };
    static double leftOperand = NULL;

    switch (message)         // Обработчик сообщений
    {
    case WM_CREATE:
    {
        sign = CreateWindow(
            _T("button"),
            _T("+/-"),
            WS_CHILD | WS_VISIBLE,
            3,
```

```
    windowHeight - buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)110,
    NULL,
    NULL
);

zero = CreateWindow(
    _T("button"),
    _T("0"),
    WS_CHILD | WS_VISIBLE,
    3 + buttonWidth + 2,
    windowHeight - buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)100,
    NULL,
    NULL
);

point = CreateWindow(
    _T("button"),
    _T(","),
    WS_CHILD | WS_VISIBLE,
    3 + 2 * buttonWidth + 2 * 2,
    windowHeight - buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)111,
    NULL,
    NULL
);

equals = CreateWindow(
    _T("button"),
    _T("="),
    WS_CHILD | WS_VISIBLE,
    3 + 3 * buttonWidth + 3 * 2,
    windowHeight - buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)112,
    NULL,
    NULL
);
```

```
one = CreateWindow(
    _T("button"),
    _T("1"),
    WS_CHILD | WS_VISIBLE,
    3,
    windowHeight - 2 - 2 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)101,
    NULL,
    NULL
);

two = CreateWindow(
    _T("button"),
    _T("2"),
    WS_CHILD | WS_VISIBLE,
    3 + buttonWidth + 2,
    windowHeight - 2 - 2 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)102,
    NULL,
    NULL
);

three = CreateWindow(
    _T("button"),
    _T("3"),
    WS_CHILD | WS_VISIBLE,
    3 + 2 * buttonWidth + 2 * 2,
    windowHeight - 2 - 2 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)103,
    NULL,
    NULL
);

plus = CreateWindow(
    _T("button"),
    _T("+"),
    WS_CHILD | WS_VISIBLE,
    3 + 3 * buttonWidth + 3 * 2,
    windowHeight - 2 - 2 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
```

```
    (HMENU)113,
    NULL,
    NULL
);

four = CreateWindow(
    _T("button"),
    _T("4"),
    WS_CHILD | WS_VISIBLE,
    3,
    windowHeight - 2 * 2 - 3 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)104,
    NULL,
    NULL
);

five = CreateWindow(
    _T("button"),
    _T("5"),
    WS_CHILD | WS_VISIBLE,
    3 + buttonWidth + 2,
    windowHeight - 2 * 2 - 3 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)105,
    NULL,
    NULL
);

six = CreateWindow(
    _T("button"),
    _T("6"),
    WS_CHILD | WS_VISIBLE,
    3 + 2 * buttonWidth + 2 * 2,
    windowHeight - 2 * 2 - 3 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)106,
    NULL,
    NULL
);

minus = CreateWindow(
    _T("button"),
    _T("-"),
    WS_CHILD | WS_VISIBLE,
```

```
        3 + 3 * buttonWidth + 3 * 2,
        windowHeight - 2 * 2 - 3 * buttonHeight - 3,
        buttonWidth,
        buttonHeight,
        hWnd,
        (HMENU)114,
        NULL,
        NULL
    );

    seven = CreateWindow(
        _T("button"),
        _T("7"),
        WS_CHILD | WS_VISIBLE,
        3,
        windowHeight - 3 * 2 - 4 * buttonHeight - 3,
        buttonWidth,
        buttonHeight,
        hWnd,
        (HMENU)107,
        NULL,
        NULL
    );

    eight = CreateWindow(
        _T("button"),
        _T("8"),
        WS_CHILD | WS_VISIBLE,
        3 + buttonWidth + 2,
        windowHeight - 3 * 2 - 4 * buttonHeight - 3,
        buttonWidth,
        buttonHeight,
        hWnd,
        (HMENU)108,
        NULL,
        NULL
    );

    nine = CreateWindow(
        _T("button"),
        _T("9"),
        WS_CHILD | WS_VISIBLE,
        3 + 2 * buttonWidth + 2 * 2,
        windowHeight - 3 * 2 - 4 * buttonHeight - 3,
        buttonWidth,
        buttonHeight,
        hWnd,
        (HMENU)109,
        NULL,
        NULL
    );
```

```
multiplication = CreateWindow(
    _T("button"),
    _T("x"),
    WS_CHILD | WS_VISIBLE,
    3 + 3 * buttonWidth + 3 * 2,
    windowHeight - 3 * 2 - 4 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)115,
    NULL,
    NULL
);

backspace = CreateWindow(
    _T("button"),
    _T("←"),
    WS_CHILD | WS_VISIBLE,
    3,
    windowHeight - 4 * 2 - 5 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)116,
    NULL,
    NULL
);

clearEntry = CreateWindow(
    _T("button"),
    _T("CE"),
    WS_CHILD | WS_VISIBLE,
    3 + 1 * buttonWidth + 1 * 2,
    windowHeight - 4 * 2 - 5 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)117,
    NULL,
    NULL
);

clear = CreateWindow(
    _T("button"),
    _T("C"),
    WS_CHILD | WS_VISIBLE,
    3 + 2 * buttonWidth + 2 * 2,
    windowHeight - 4 * 2 - 5 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
```

```
    hWnd,
    (HMENU)118,
    NULL,
    NULL
);

division = CreateWindow(
    _T("button"),
    _T("÷"),
    WS_CHILD | WS_VISIBLE,
    3 + 3 * buttonWidth + 3 * 2,
    windowHeight - 4 * 2 - 5 * buttonHeight - 3,
    buttonWidth,
    buttonHeight,
    hWnd,
    (HMENU)119,
    NULL,
    NULL
);

hEdit = CreateWindow(
    _T("edit"),
    _T("0"),
    WS_CHILD | WS_VISIBLE | ES_RIGHT | ES_NUMBER | ES_NOHIDESEL,
    0,
    windowHeight - 5 * 2 - 6 * buttonHeight - 3 - 50,
    windowWidth,
    100,
    hWnd,
    NULL,
    NULL,
    NULL
);
SendMessage(hEdit, EM_SETLIMITTEXT, 10, 0);

hStatic = CreateWindow(
    _T("static"),
    _T("0"),
    WS_CHILD | WS_VISIBLE | SS_RIGHT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    windowWidth - 10,
    15,
    hWnd,
    NULL,
    NULL,
    NULL
);

HMENU hMenu1 = CreateMenu();
HMENU hMenu2 = CreateMenu();
```

```
    HMENU hMenubar1 = CreateMenu();
    AppendMenu(hMenu1, MF_STRING, 201, L"Copy");
    AppendMenu(hMenu1, MF_STRING, 202, L"Paste");
    AppendMenuW(hMenubar1, MF_POPUP, (UINT_PTR)hMenu1, L"Edit");
    AppendMenu(hMenu2, MF_STRING, 211, L"About Calculator");
    AppendMenuW(hMenubar1, MF_POPUP, (UINT_PTR)hMenu2, L"Help");
    SetMenu(hWnd, hMenubar1);

    LOGFONT lf;
    memset(&lf, 0, sizeof(LOGFONT));
    lstrcpy(lf.lfFaceName, _T("Calibri")); // Имя шрифта.
    lf.lfHeight = 80; // По высоте.
    HFONT hFont = CreateFontIndirect(&lf);
    SendMessage(hEdit, WM_SETFONT, (WPARAM)hFont, 0L);
}
break;
case WM_GETMINMAXINFO:
{
    MINMAXINFO* MMI = (MINMAXINFO*)lParam;
    MMI->ptMinTrackSize.x = windowWidth + 16;
    MMI->ptMaxTrackSize.x = windowWidth + 16;
    MMI->ptMinTrackSize.y = windowHeight + 60;
    MMI->ptMaxTrackSize.y = windowHeight + 60;
}
break;
case WM_COMMAND:
{
    //if (refresh2) SetWindowText(hStatic, ttt);
    switch (LOWORD(wParam))
    {
    case 100:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
        if (lstrlen(buf) <= 7) {
            StrCat(buf, TEXT("0"));
            SetWindowText(hEdit, buf);
        }
    }
    break;

    case 101:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
        if (lstrlen(buf) <= 7) {
            StrCat(buf, TEXT("1"));
            SetWindowText(hEdit, buf);
        }
    }
    break;
```

```
case 102:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
    if (lstrlen(buf) <= 7) {
        StrCat(buf, TEXT("2"));
        SetWindowText(hEdit, buf);
    }
}
break;

case 103:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
    if (lstrlen(buf) <= 7) {
        StrCat(buf, TEXT("3"));
        SetWindowText(hEdit, buf);
    }
}
break;

case 104:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
    if (lstrlen(buf) <= 7) {
        StrCat(buf, TEXT("4"));
        SetWindowText(hEdit, buf);
    }
}
break;

case 105:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
    if (lstrlen(buf) <= 7) {
        StrCat(buf, TEXT("5"));
        SetWindowText(hEdit, buf);
    }
}
break;

case 106:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
    if (lstrlen(buf) <= 7) {
        StrCat(buf, TEXT("6"));
```

```
            SetWindowText(hEdit, buf);
        }
    }
    break;

    case 107:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
        if (lstrlen(buf) <= 7) {
            StrCat(buf, TEXT("7"));
            SetWindowText(hEdit, buf);
        }
    }
    break;

    case 108:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
        if (lstrlen(buf) <= 7) {
            StrCat(buf, TEXT("8"));
            SetWindowText(hEdit, buf);
        }
    }
    break;

    case 109:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0' || refresh) { *buf = 0; refresh = false; }
        if (lstrlen(buf) <= 7) {
            StrCat(buf, TEXT("9"));
            SetWindowText(hEdit, buf);
        }
    }
    break;

    case 110:
    {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf != '0' && *buf != 0) {
            if (*buf == '-') StrCpy(buf, &buf[1]);
            else {
                TCHAR tmp[256] = { '-' };
                StrCat(tmp, buf);
                StrCpy(buf, tmp);
            }
            SetWindowText(hEdit, buf);
        }
    }
```

```
break;

case 111:
{
    GetWindowText(hEdit, buf, sizeof(buf));
    if (StrChr(buf, ',') == NULL) {
        StrCat(buf, TEXT(","));
        SetWindowText(hEdit, buf);
    }
}
break;

case 112:
{
    operation[0] = operation[1];
    operation[1] = 1;
    refresh = true;
    if (operation[0] == 0) leftOperand = _wtof(buf);
    StrCat(ttt, buf);
    StrCat(ttt, TEXT(" = "));
    SetWindowText(hStatic, ttt);
    memset(&ttt, 0, lstrlen(buf));
}
break;

case 113:
{
    operation[0] = operation[1];
    operation[1] = 2;
    refresh = true;
    if (operation[0] == 0) leftOperand = _wtof(buf);
    StrCat(ttt, buf);
    StrCat(ttt, TEXT(" + "));
    SetWindowText(hStatic, ttt);
}
break;

case 114:
{
    operation[0] = operation[1];
    operation[1] = 3;
    refresh = true;
    if (operation[0] == 0) leftOperand = _wtof(buf);
    StrCat(ttt, buf);
    StrCat(ttt, TEXT(" - "));
    SetWindowText(hStatic, ttt);
}
break;

case 115:
{
```

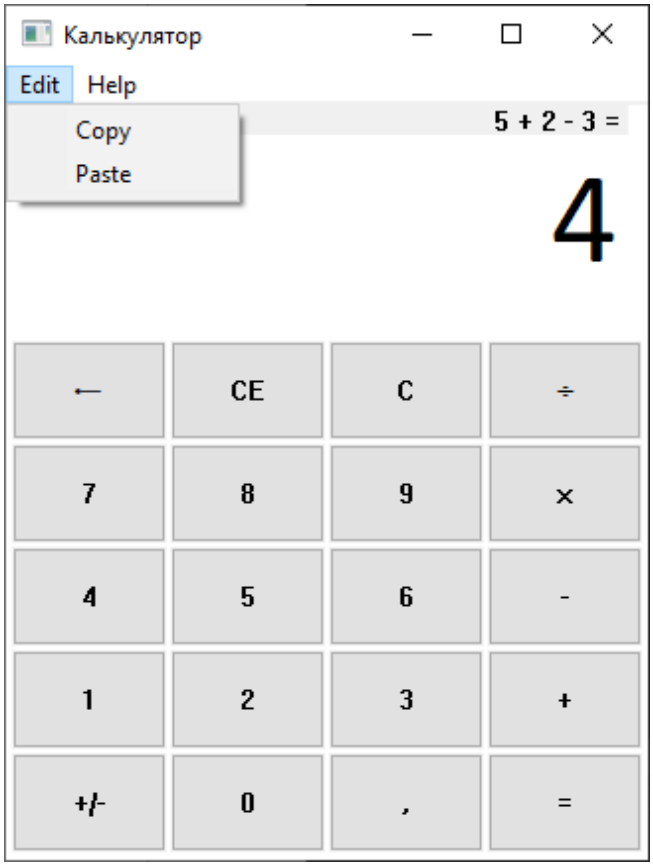
```
    operation[0] = operation[1];
    operation[1] = 4;
    refresh = true;
    if (operation[0] == 0) leftOperand = _wtof(buf);
    StrCat(ttt, buf);
    StrCat(ttt, TEXT(" * "));
    SetWindowText(hStatic, ttt);
}
break;

case 116:
{
    if (!refresh) {
        GetWindowText(hEdit, buf, sizeof(buf));
        if (*buf == '0') *buf = 0;
        int a = lstrlen(buf);
        if (a == 0 || a == 1) buf[0] = '0';
        else StrNCpy(buf, buf, a);
        SetWindowText(hEdit, buf);
    }
}
break;

case 117:
{
    memset(&buf[1], 0, lstrlen(buf));
    buf[0] = '0';
    SetWindowText(hEdit, buf);
}
break;

case 118:
{
    refresh = false;
    operation[0] = 0;
    operation[1] = 0;
    leftOperand = NULL;
    memset(&buf[1], 0, lstrlen(buf));
    buf[0] = '0';
    SetWindowText(hEdit, buf);
    SetWindowText(hStatic, buf);
}
break;

case 119:
{
    operation[0] = operation[1];
    operation[1] = 5;
    refresh = true;
    if (operation[0] == 0) leftOperand = _wtof(buf);
    StrCat(ttt, buf);
```

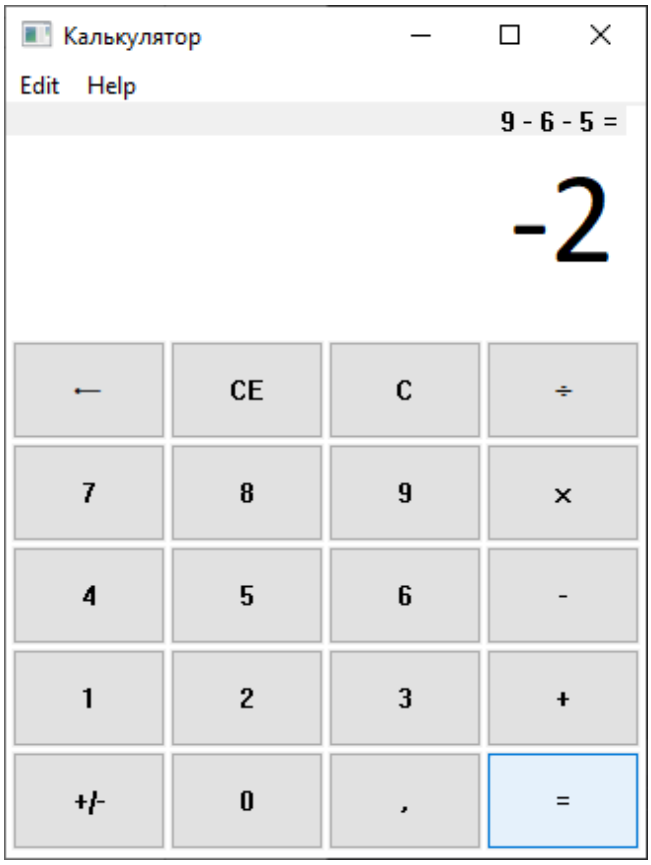
```
            StrCat(ttt, TEXT(" / "));
            SetWindowText(hStatic, ttt);
        }
        break;


        case 201:
        {
            OpenClipboard(0);
            EmptyClipboard();
            HGLOBAL hStrMem = GlobalAlloc(GMEM_MOVEABLE, sizeof(buf));
            void* pStrMem = GlobalLock(hStrMem);
            memcpy(pStrMem, buf, sizeof(buf));
            GlobalUnlock(pStrMem);
            SetClipboardData(CF_UNICODETEXT, hStrMem);
            CloseClipboard();
            //SendMessage(hEdit, WM_COPY, 0, 0);
        }
        break;

        case 202:
        {
            /*HGLOBAL hglb = GetClipboardData(CF_TEXT);
            if (hglb != NULL)
            {
                LPCSTR* lptstr = GlobalLock(hglb);
                if (lptstr != NULL)
                {
                    // Обращаемся к определяемой прикладной программой функции
                    // ReplaceSelection, чтобы вставить текст и перерисовать окно
.

                    SetWindowText(hStatic, lptstr);
                    GlobalUnlock(hglb);
                }
            }
            CloseClipboard();
            memset(&buf[1], 0, lstrlen(buf));
            buf[0] = '0';
            SetWindowText(hStatic, buf);*/
            SendMessage(hEdit, WM_PASTE, 0, 0);
        }
        break;


        case 211:
        {
            MessageBox(hWnd, TEXT("Терентьев В.О. ИУ5-
43"), TEXT("Message"), MB_OK);
        }
        break;
```

```
        }
    }
    break;
    /*case WM_SIZE:
    {
        RECT rc;
        GetClientRect(hwndDlg, &rc);
        SetWindowPos(hEdit, 0, rc.left + 4, rc.top + 4, rc.right - rc.left - 8, r
c.bottom - rc.top - 34, SWP_NOZORDER);
        SetWindowPos(hOk, 0, rc.left + 4, rc.bottom - 26, rc.right - rc.left - 11
2, 22, SWP_NOZORDER);
        SetWindowPos(hCancel, 0, rc.right - 104, rc.bottom - 26, 100, 22, SWP_NOZ
ORDER);
    }
    break;*/
    case WM_DESTROY:
        PostQuitMessage(0);
        break;          // Завершение программы
    default:            // Обработка сообщения по умолчанию
        return DefWindowProc(hWnd, message, wParam, lParam);
    }

    switch (operation[0]) {
    case 0:
        break;
    case 1:
    {
        operation[0] = 0;
        string klj = to_string(leftOperand);
        copy(klj.begin(), klj.begin() + 8, buf);
        SetWindowText(hEdit, buf);
    }
    break;
    case 2:
    {

        operation[0] = 0;
        PWSTR qwe = StrChr(buf, ',');
        if (qwe != NULL) {
            *qwe = '.';
        }
        leftOperand = leftOperand + _wtof(buf);
        string klj = to_string(leftOperand);
        copy(klj.begin(), klj.begin() + 8, buf);
        //*(double*)buf = leftOperand;
        SetWindowText(hEdit, buf);
    }
    break;

    case 3:
    {
```

```
            operation[0] = 0;
            PWSTR qwe = StrChr(buf, ',');
            if (qwe != NULL) {
                *qwe = '.';
            }
            leftOperand = leftOperand - _wtof(buf);
            string klj = to_string(leftOperand);
            copy(klj.begin(), klj.begin() + 8, buf);
            SetWindowText(hEdit, buf);
        }
        break;

        case 4:
        {
            operation[0] = 0;
            PWSTR qwe = StrChr(buf, ',');
            if (qwe != NULL) {
                *qwe = '.';
            }
            leftOperand = leftOperand * _wtof(buf);
            string klj = to_string(leftOperand);
            copy(klj.begin(), klj.begin() + 8, buf);
            SetWindowText(hEdit, buf);
        }
        break;

        case 5:
        {
            operation[0] = 0;
            PWSTR qwe = StrChr(buf, ',');
            if (qwe != NULL) {
                *qwe = '.';
            }
            leftOperand = leftOperand / _wtof(buf);
            string klj = to_string(leftOperand);
            string::iterator it = klj.end();
            copy(klj.begin(), klj.begin() + 8, buf);
            SetWindowText(hEdit, buf);
        }
        break;
        }

        return 0;
}
```

3.3. Анализ результатов





4. Лабораторная работа № 4

4.1. Постановка задачи

Первый поток выводит в верхнюю половину окрашенного окна большой круг, кисть для заливки которого выбирается случайным образом из трех созданных заранее кистей разных цветов. Для того чтобы наглядно фиксировать моменты смены кисти, в центр круга выводится номер такта этого потока.

Второй поток выводит в нижнюю половину окна цветными символами текущее время (часы, минуты и секунды), получаемое с помощью функции GetLocalTime().

4.2. Текст программы

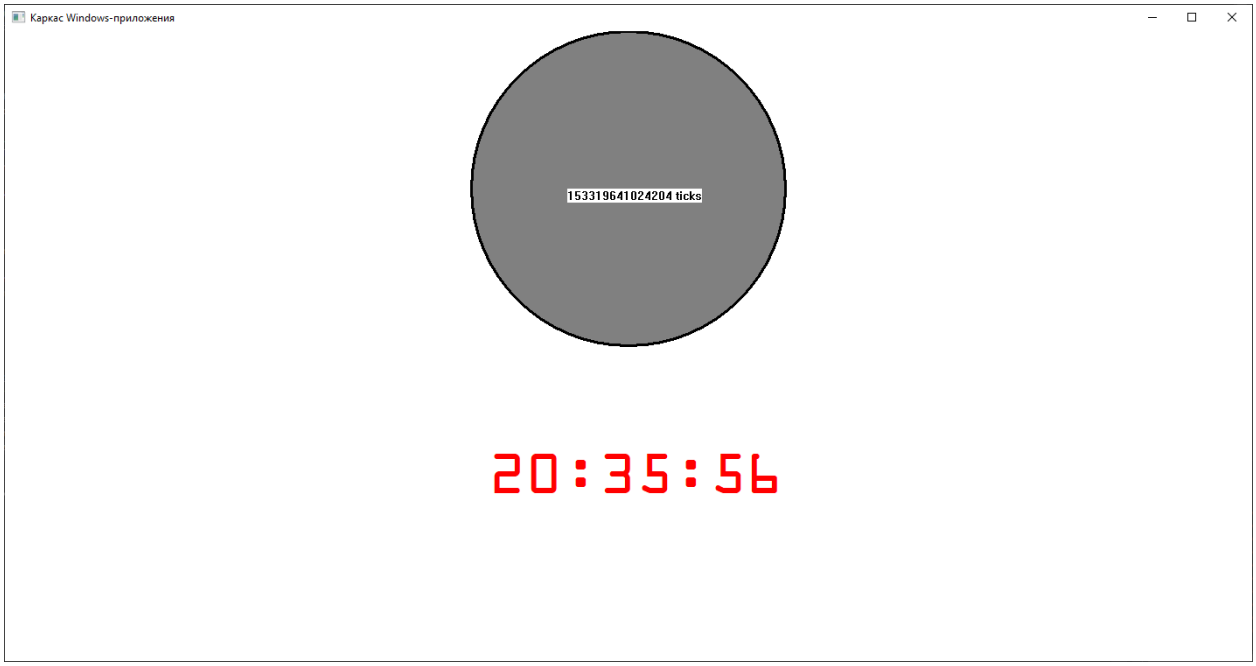
```
#include <Windows.h>
#include <tchar.h>
#include <string>
#include <atlstr.h>

#include <mmsystem.h>
#pragma comment(lib, "winmm.lib")
#pragma intrinsic(__rdtsc)


using namespace std;

static int widht, hight, r;
static HWND hWnd;
static HDC hdc;
static PAINTSTRUCT ps;
static HANDLE hThread2;
static BOOL bFin = true;

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

TCHAR WinName[] = _T("MainFrame");

DWORD WINAPI Thread2(LPVOID);

int WINAPI _tWinMain(HINSTANCE This,         // Дескриптор текущего приложения
    HINSTANCE Prev,     // В современных системах всегда 0
    LPTSTR cmd,         // Командная строка
    int mode)       // Режим отображения окна
{
    HWND hWnd;      // Дескриптор главного окна программы
    MSG msg;        // Структура для хранения сообщения
    WNDCLASS wc;    // Класс окна
    // Определение класса окна
    wc.hInstance = This;
    wc.lpszClassName = WinName;                 // Имя класса окна
```

```
    wc.lpfnWndProc = WndProc;                    // Функция окна
    wc.style = CS_HREDRAW | CS_VREDRAW;             // Стиль окна
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);          // Стандартная иконка
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);       // Стандартный курсор
    wc.lpszMenuName = NULL;                      // Нет меню
    wc.cbClsExtra = 0;                       // Нет дополнительных данных класса
    wc.cbWndExtra = 0;                       // Нет дополнительных данных окна
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);  // Заполнение окна белым цвет
ом


    // Регистрация класса окна
    if (!RegisterClass(&wc)) return 0;

    // Создание окна
    hWnd = CreateWindow(WinName,             // Имя класса окна
        _T("Каркас Windows-приложения"),         // Заголовок окна
        WS_OVERLAPPEDWINDOW,         // Стиль окна
        CW_USEDEFAULT,               // x
        CW_USEDEFAULT,               // y     Размеры окна
        CW_USEDEFAULT,               // width
        CW_USEDEFAULT,               // Height
        HWND_DESKTOP,                // Дескриптор родительского окна
        NULL,                        // Нет меню
        This,                        // Дескриптор приложения
        NULL);                   // Дополнительной информации нет

    ShowWindow(hWnd, mode);                  // Показать окно

    // Цикл обработки сообщений
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);          // Функция трансляции кодов нажатой клави
ши
        DispatchMessage(&msg);       // Посылает сообщение функции WndProc()
    }
    return 0;
}

// Оконная функция вызывается операционной системой
// и получает сообщения из очереди для данного приложения

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    DWORD dwIDThread;
    switch (message)          // Обработчик сообщений
    {

    case WM_CREATE:
    {
        bFin = true;
```

```
            hdc = GetDC(hWnd);
            hThread2 = CreateThread(NULL, 0, Thread2, (void*)hdc, 0, &dwIDThread);
            RECT rc;
            GetClientRect(hWnd, &rc);
            hight = rc.bottom;
            widht = rc.right;
        }
        break;

        case WM_SIZE:
        {
            RECT rc;
            GetClientRect(hWnd, &rc);
            hight = rc.bottom;
            widht = rc.right;
        }
        break;

        case WM_PAINT:
        {

            hdc = BeginPaint(hWnd, &ps);

            HPEN hPen1 = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));
            SelectObject(hdc, hPen1);
            HBRUSH hBrush1 = (HBRUSH)GetStockObject(BLACK_BRUSH);
            HBRUSH hBrush2 = (HBRUSH)GetStockObject(GRAY_BRUSH);
            HBRUSH hBrush3 = (HBRUSH)GetStockObject(WHITE_BRUSH);
            r = rand() % 3;

            if (r == 0) {
                SelectObject(hdc, hBrush1);
            }
            else if (r == 1) {
                SelectObject(hdc, hBrush2);
            }
            else {
                SelectObject(hdc, hBrush3);
            }
            Ellipse(hdc, (widht / 2) - (hight / 4), 0, (widht / 2) + (hight / 4), hig
ht / 2);

            /*
            DWORD start;
            start = timeGetTime();
            wstring b = to_wstring(start);
            TextOutW(hdc, (widht / 2) - 30, hight / 4, b.c_str(), 9);
            */


            unsigned __int64 i;
```

```
        i = __rdtsc();
        char q[100];
        snprintf(q, 100, "%I64d ticks\n", i);
        wstring b(&q[0], &q[99]);
        TextOutW(hdc, (widht / 2) - 70, hight / 4, b.c_str(), 21);


        /*
        wstring b = to_wstring(GetCurrentProcessorNumber());
        TextOutW(hdc, (widht / 2) - 60, hight / 4, b.c_str(), 15);
        */

        EndPaint(hWnd, &ps);
    }
    break;

    case WM_DESTROY:
        bFin = false;
        PostQuitMessage(0);
        break;          // Завершение программы
    default:            // Обработка сообщения по умолчанию
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

DWORD WINAPI Thread2(LPVOID hdcc) {
    HDC hdc = (HDC)hdcc;
    SetTextColor(hdc, RGB(255, 0, 0));
    SetTextAlign(hdc, TA_BOTTOM);
    SYSTEMTIME lpSystemTime;
    LOGFONT lgf;
    HFONT fn;
    memset(&lgf, 0, sizeof(LOGFONT));
    lgf.lfHeight = 72;
    StrCpy(lgf.lfFaceName, _T("OCR A Extended"));
    fn = CreateFontIndirect(&lgf);
    SelectObject(hdc, fn);
    while (bFin) {
        GetLocalTime(&lpSystemTime);
        wstring a = to_wstring(lpSystemTime.wHour) + L":" + to_wstring(lpSystemTi
me.wMinute) + L":" + to_wstring(lpSystemTime.wSecond);
        TextOutW(hdc, widht / 2 - 160, hight - (hight / 4), a.c_str(), 8);
        Sleep(10);
    }
    return 0;
}
```

4.3. Анализ результатов



5. Выводы

Больше всего понравилось делать 3 лабораторную работу, потому что получилось почти готовое собственное приложение, аналог используемого и распространенного калькулятора. Следовательно, можно сравнить и понять, что ты уже научился и умеешь делать по сравнению с действующими разработчиками, что еще не умеешь, где ты можешь сделать лучше, а где другие сделали лучше. 2 и 4 лабораторные работы понравились с технической точки зрения (взаимодействие с окнами приложений и многопоточность). В 1 и 2 лабораторных работах было не понятно, как дальше то, что ты сделал, можно было бы использовать или применить (хотя 1 лабораторная работа была скорее всего начальная, подготовительная и с другими целями).

6. Ссылка на репозиторий

https://github.com/iYroglif/WinApi