Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные системы и системное программирование

Отчёт
к лабораторной работе
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)

Студент: гр.153502
Александрёнок И.А.

Проверил: Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ..... 3
2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ..... 4
3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ..... 5
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ..... 8
ПРИЛОЖЕНИЕ А..... 9

1 ЦЕЛЬ РАБОТЫ

Создать приложение для визуализации и анализа данных с использованием графиков и диаграмм.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Объект Device Context – контекст графического устройства, абстракция «поверхности рисования», соответствует логическому устройству, окну или его части; описатель «экземпляра» логического устройства. Все действия над изображением в логическом устройстве выполняются через контекст. В Win GDI основной подход – векторный: изображение состоит из ряда графических примитивов, каждый из которых прорисован соответствующим инструментом (объектом) с заданными параметрами. С каждым контекстом DC одновременно связано не более одного объекта каждого вида (некоторые могут быть «пустыми»). Выбор нового осуществляется использованием функции SelectObject, которая также возвращает описатель прежнего объекта. После завершения использования желательно возвращать предыдущую установку

3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске программы пользователю предлагается выбрать режим работы приложения (рисунок 1), после предлагается выбрать файл, с которого будет произведено считывание данных для визуализации (рисунок 2). Оно происходит парами строк. Предполагается, что данные поступают в корректном формате, в противном случае, при возникновении ошибок во время чтения, оно будет остановлено и будут использованы данные по умолчанию.

В режиме работы «график» (рисунок 3) первая строка из пары является координатой x (целое неотрицательное число), вторая – y (целое число). Если приложение открыто в режиме «диаграмма» (рисунок 4), то первая строка является меткой, используемой для легенды, а вторая – непосредственно само значение (целое неотрицательное число).

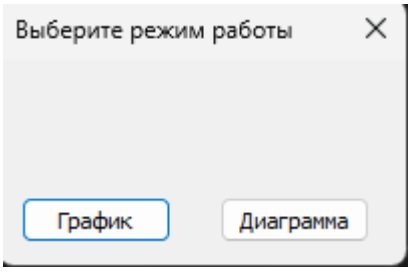

Рисунок 1 – Выбор режима работы

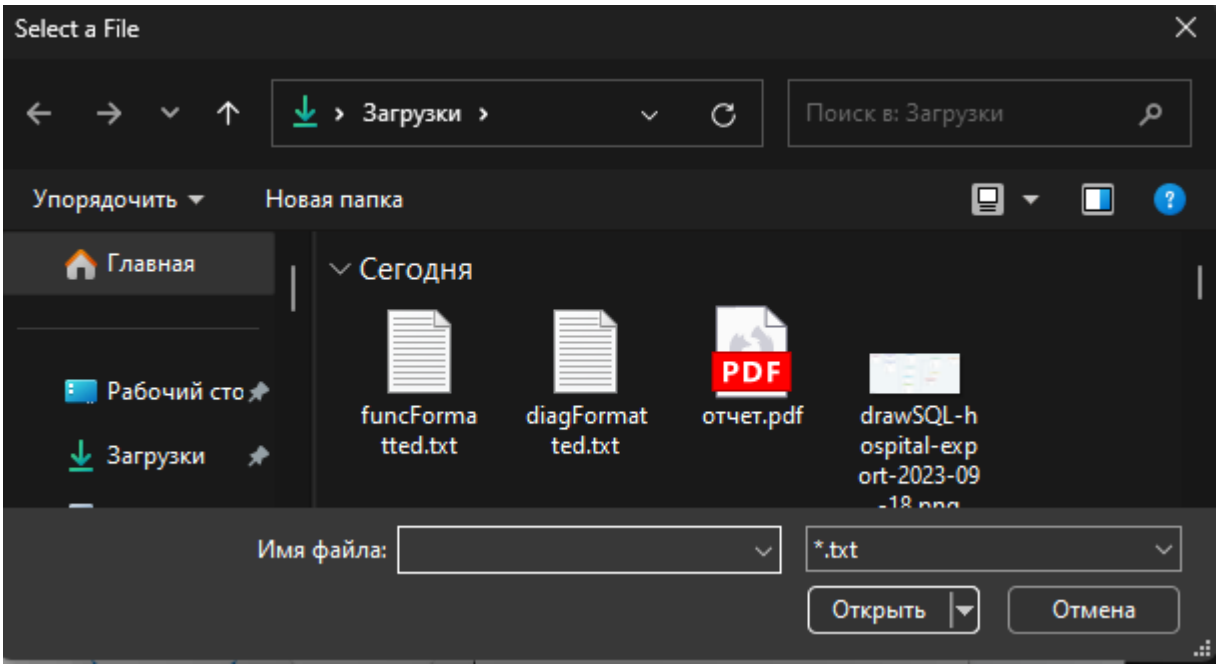

Рисунок 2 – Окно выбора файла

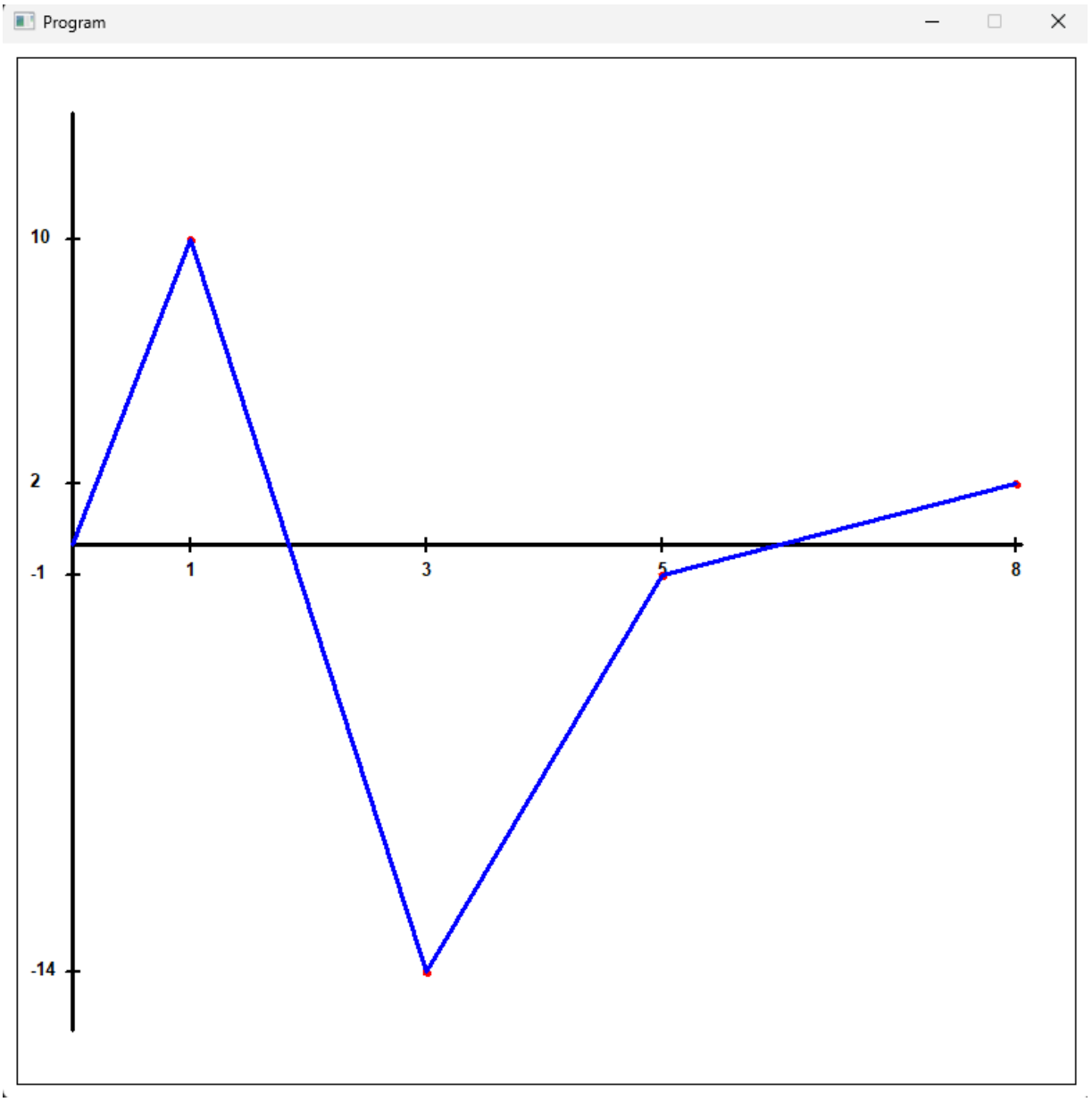

Рисунок 3 – Пример работы программы в режиме «график»

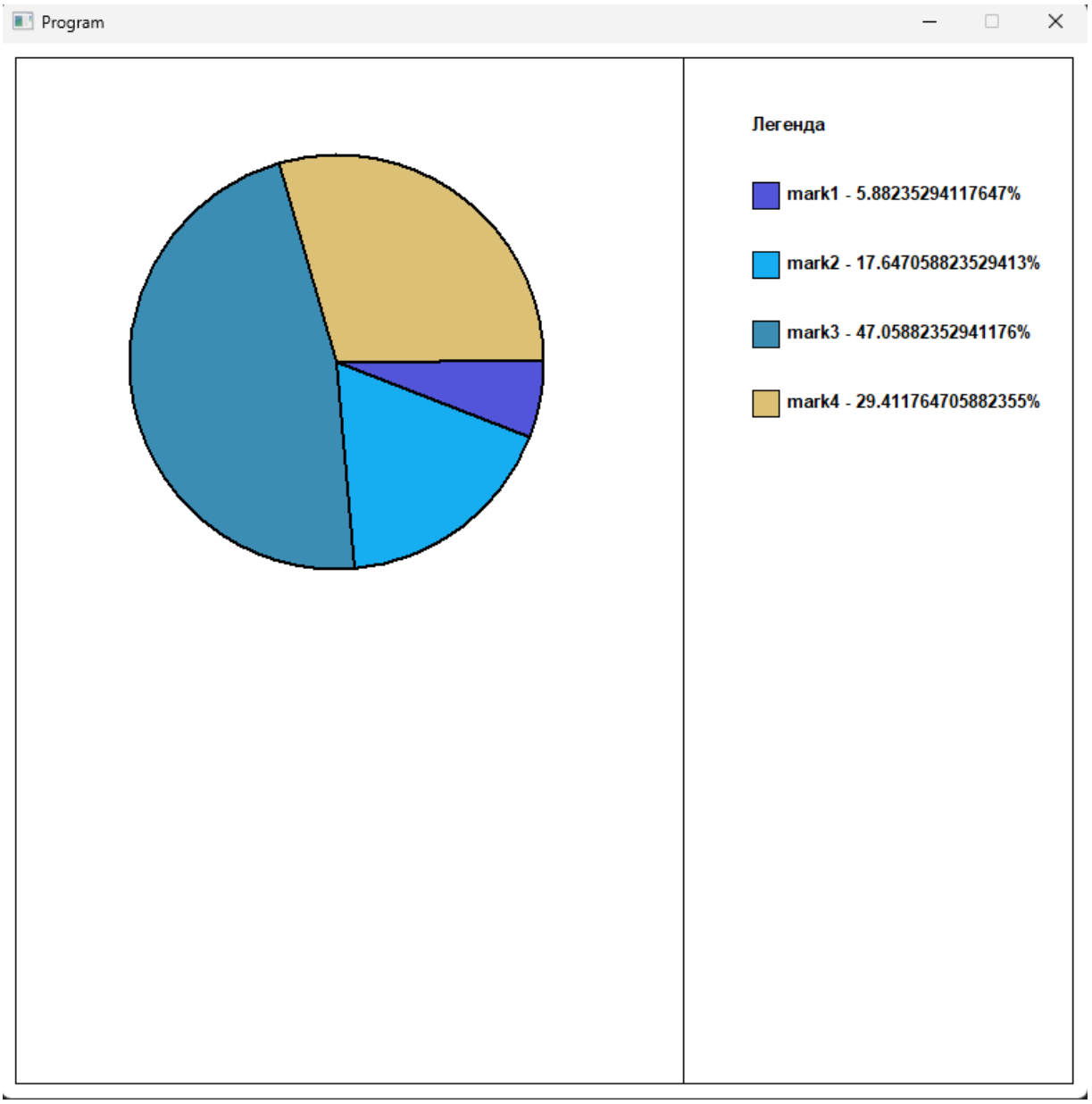

Рисунок 4 – Пример работы программы в режиме «диаграмма»

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] 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

ПРИЛОЖЕНИЕ А
Исходный код программы

Файл DiagView.h

```
#pragma once
#define _USE_MATH_DEFINES
#include<windows.h>
#include <vector>
#include <fstream>
#include<string>
#include <utility>
#include <iostream>
#include <format>
#include <math.h>
using namespace std;
namespace diag {
   RECT rt;
   int circleCenterX, circleCenterY,circleRadius=150;

   //Для диаграмм - метка + значение
   //Шаблонные данные в случае ошибки чтения пользовательских
                vector<pair<string, int>> diagramAnalysisData = {
{"First",1},{"Second",8},{"Third",3},{"Fourth",5},{"Fifth",2},{"Sixth",7} };
   int dataCount;
   int sum=0;
   double _coef;
   vector<double> calculatedCoefs;//в РАДИАНАХ!
   vector<pair<int,int>> calculatedSubs;
   vector<pair<int, int>> calculatedCoords;
   vector<COLORREF> colors;
   void ReadDataFromFile() {
    fstream new_file;
    new_file.open(filename, ios::in);
    vector<pair<string, int>> temp;
    string input;
    bool isMark = true;
    int tvalue;
    string tmark;
    try {
          while (getline(new_file, input)) {
                if (isMark) {
                      isMark = false;
                      tmark = input;
                }
                else {
```

```
                        isMark = true;
                        tvalue = stoi(input);
                        temp.push_back(make_pair(tmark, tvalue));
                    }
                }
            }
            catch (exception ex) {
            }


            if(temp.size()>1)
            diagramAnalysisData = temp;

            new_file.close();
            }



           //углы поворота
           void CalcCoefs() {

            for (int i = 0; i < dataCount; i++) {
                sum += diagramAnalysisData[i].second;
            }
            _coef = 2*M_PI / sum;

            for (int i = 0; i < dataCount; i++) {
                                calculatedCoefs.push_back(_coef                *
diagramAnalysisData[i].second);
            }


            for (int i = 1; i < dataCount; i++) {
                calculatedCoefs[i] += calculatedCoefs[i - 1];
            }


           }
           void CalcCoords() {

            for (int i = 0; i < dataCount; i++) {

                int x = circleCenterX + circleRadius * cos(calculatedCoefs[i]);
                int y = circleCenterY + circleRadius * sin(calculatedCoefs[i]);
```

```
                calculatedCoords.push_back(make_pair(x, y));

                    x = circleCenterX + circleRadius*1.2 *
cos(calculatedCoefs[i]*0.9);
                    y = circleCenterY + circleRadius*1.2 *
sin(calculatedCoefs[i]*0.9);
                calculatedSubs.push_back(make_pair(x, y));

        }
        }

        void RandColors() {
         for (int i = 0; i < dataCount; i++) {
                colors.push_back(RandomizeColor());
         }
        }

        void PrepData() {
         ReadDataFromFile();
         srand(time(0));
         circleCenterX = rt.right / 2-150;
         circleCenterY = rt.bottom / 2-150;
         dataCount = diagramAnalysisData.size();
         CalcCoefs();
         CalcCoords();
         RandColors();
        }
        void Paint(HWND hwnd) {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hwnd, &ps);

         //рамка
         Rectangle(hdc, rt.left + 10, rt.top + 10, rt.right - 10, rt.bottom - 10);

         //рамка под легенду
          Rectangle(hdc, circleCenterX+circleRadius+100, rt.top + 10, rt.right - 10,
rt.bottom - 10);
         string tempstr;
         wstring temp;
           TextOut(hdc, circleCenterX + circleRadius + 150, rt.top + 50,
L"Легенда", 7);
```

```
        //скелет диаграммы + легенда
         //DrawCircle(hdc, circleCenterX, circleCenterY, circleRadius, RGB(0, 0, 0), 4);
        DrawSector(hdc, circleCenterX, circleCenterY, circleRadius,
                        calculatedCoords[0].first, calculatedCoords[0].second, calculatedCoords[dataCount-1].first, calculatedCoords[dataCount-1].second, colors[0], 2);

        for (int i = 0; i < dataCount; i++) {

                if (i > 0) {
                        DrawSector(hdc, circleCenterX, circleCenterY, circleRadius,
                                calculatedCoords[i].first, calculatedCoords[i].second, calculatedCoords[i - 1].first, calculatedCoords[i - 1].second, colors[i], 2);
                }

                //собственно легенда
                HBRUSH newBrush = CreateSolidBrush(colors[i]);
                HGDIOBJ oldBrush = SelectObject(hdc, newBrush);
                Rectangle(hdc, circleCenterX + circleRadius + 150, (rt.top + 50) + (i + 1) * 50, circleCenterX + circleRadius + 170, (rt.top + 50) + (i + 1) * 50 + 20);
                 tempstr = diagramAnalysisData[i].first+format(" - {}%",1.0 * diagramAnalysisData[i].second / sum * 100);
                temp = wstring(tempstr.begin(), tempstr.end());
                TextOut(hdc, circleCenterX + circleRadius + 175, (rt.top + 50) + (i + 1) * 50, temp.c_str(), temp.length());



                SelectObject(hdc, oldBrush);
                DeleteObject(newBrush);
        }

        EndPaint(hwnd, &ps);
       }

    }
```

Файл Utility.h

```
#pragma once
#include <windows.h>


using namespace std;
OPENFILENAME ofn;
```

```
wstring filename;

BOOL DrawLine(HDC hdc, int xFrom, int yFrom, int xTo, int yTo, COLORREF color, int width) {
    MoveToEx(hdc, xFrom, yFrom, NULL); //сделать текущими координаты xFrom, yFrom

    HPEN newPen = CreatePen(PS_SOLID, width, color);
    HGDIOBJ oldPen = SelectObject(hdc, newPen);

    BOOL result = LineTo(hdc, xTo, yTo);
    SelectObject(hdc, oldPen);
    DeleteObject(newPen);

    return result;
}

BOOL DrawDot(HDC hdc, int x, int y, COLORREF color, int width) {
    MoveToEx(hdc, x, y, NULL); //сделать текущими координаты x,y

    HPEN newPen = CreatePen(PS_SOLID, width, color);
    HGDIOBJ oldPen = SelectObject(hdc, newPen);

    BOOL result = LineTo(hdc, x + 1, y + 1);
    SelectObject(hdc, oldPen);
    DeleteObject(newPen);

    return result;
}

//x,y- центр окружности x1,y1 - начальная точка дуги, x2,y2-конечная
BOOL DrawSector(HDC hdc, int x, int y, int radius, int x1, int y1, int x2, int y2, COLORREF color, int width) {

    HPEN newPen = CreatePen(PS_SOLID, width, RGB(0,0,0));
    HGDIOBJ oldPen = SelectObject(hdc, newPen);

    HBRUSH newBrush = CreateSolidBrush(color);
    HGDIOBJ oldBrush = SelectObject(hdc, newBrush);

    //Отрисовка сектора
    BOOL result = Pie(hdc, x - radius, y - radius, x + radius, y + radius, x1, y1, x2, y2);
```

```
    SelectObject(hdc, oldPen);
    DeleteObject(newPen);

    SelectObject(hdc, oldBrush);
    DeleteObject(newBrush);


    return result;

}
COLORREF RandomizeColor() {
    return RGB((BYTE)rand() % 255, (BYTE)rand() % 255, (BYTE)rand() %
255);
}

bool OpenFile(HWND hwnd) {

    const std::wstring title = L"Select a File";
    std::wstring src(MAX_PATH, L'\0');

    OPENFILENAMEW ofn = { };
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = NULL;
    ofn.lpstrFilter = TEXT("*.txt\0");
    ofn.lpstrFile = &src[0];  // use the std::wstring buffer directly
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrTitle = title.c_str();
    ofn.Flags = OFN_DONTADDTORECENT | OFN_FILEMUSTEXIST;

    if (GetOpenFileNameW(&ofn))
    {
        filename = src;      //<----------Save filepath in global variable
        return true;
    }
    return false;
}
```

Файл FuncView.h

```
#pragma once
#include <vector>
#include <format>
#include <windows.h>
#include <algorithm>
#include <fstream>
#include<string>
#include <utility>
```

```
#include <iostream>
#include "Utility.h"
using namespace std;

namespace func {

    RECT rt;

    //смещение оси x, чтобы видеть отрицательные значения
    int moveConst = 400;

    //Для Графиков (x,y) точки
    //Шаблонные данные в случае ошибки чтения пользовательских
    vector<pair<int, int>> funcAnalysisData = { {4,-6}, {1,1},{5,2},{10,4} };
    int dataCount;

    //установка предельных значений (в случае с y еще учет отрицательных)
    int maxDataValueX;
    int maxDataValueY;
    void SetMAXValueFuncAnalysis(bool byX) {
        int max_val = INT_MIN;
        int min_val = INT_MAX;
        if (byX) {
        for (int i = 0; i < dataCount; i++) {
        if (max_val < funcAnalysisData[i].first) {
            max_val = funcAnalysisData[i].first;
        }
        if (min_val > funcAnalysisData[i].first) {
            min_val = funcAnalysisData[i].first;
        }
        }
        maxDataValueX = max_val;
        }
        else {
        for (int i = 0; i < dataCount; i++) {
        if (max_val < funcAnalysisData[i].second) {
            max_val = funcAnalysisData[i].second;
        }
        if (min_val > funcAnalysisData[i].second) {
            min_val = funcAnalysisData[i].second;
        }
        }
        if (min_val != INT_MAX) {
        maxDataValueY = abs(min_val) > max_val ? abs(min_val) * 2+1 :
```

```
max_val * 2+1;
                }
                else {
                maxDataValueY = max_val;
                }
                }
        }

        //цена деления + координаты точек
        int xSpan, ySpan;
        vector<pair<int, int>> coords;
        void SetCoordsANDSpanData() {
                //цена деления
                ySpan = (rt.bottom - rt.top - 100) / maxDataValueY;
                xSpan = (rt.right - rt.left - 100) / maxDataValueX;

                //Разметка делений
                for (int i = 0; i < dataCount; i++) {
                int x = funcAnalysisData[i].first * xSpan;
                int y = funcAnalysisData[i].second * ySpan;
                coords.push_back({ x,y });
                }
        }

        void ReadDataFromFile() {
                fstream new_file;
                new_file.open(filename, ios::in);
                vector<pair<int, int>> temp;
                string input;
                bool isMark = true;
                int tval1, tval2;
                try {
                while (getline(new_file, input)) {
                if (isMark) {
                        isMark = false;
                        tval1 = stoi(input);
                }
                else {
                        isMark = true;
                        tval2 = stoi(input);
                        temp.push_back(make_pair(tval1, tval2));
                }
                }
                }
                catch (exception ex) {
```

```
            }

            if (temp.size() > 1)
            funcAnalysisData = temp;

            new_file.close();
        }
        //Сортировка, поиск предельных значений, цены деления и координат в пределах окна соответственно
        void PrepData() {
            ReadDataFromFile();
            dataCount = funcAnalysisData.size();
            sort(funcAnalysisData.begin(), funcAnalysisData.end(), [](auto& left, auto& right) {
            return left.first < right.first;
            });
            SetMAXValueFuncAnalysis(true);
            SetMAXValueFuncAnalysis(false);
            SetCoordsANDSpanData();
        }


        void CALLBACK Paint(HWND hwnd) {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hwnd, &ps);

            //рамка
            Rectangle(hdc, rt.left + 10, rt.top + 10, rt.right - 10, rt.bottom - 10);

            //координатные оси
            DrawLine(hdc, rt.left + 50, rt.bottom - moveConst, rt.right - 50, rt.bottom - moveConst, RGB(0, 0, 0), 3);
            DrawLine(hdc, rt.left + 50, rt.bottom - 50, rt.left + 50, rt.top + 50, RGB(0, 0, 0), 3);

            //Разметка делений
            for (int i = 0; i < dataCount; i++) {
            //ось X
            DrawLine(hdc, (rt.left + 50) + coords[i].first, rt.bottom - moveConst + 5, (rt.left + 50) + coords[i].first, rt.bottom - moveConst - 5, RGB(0, 0, 0), 2);
            string tempstr = format("{}", funcAnalysisData[i].first);
            wstring temp = std::wstring(tempstr.begin(), tempstr.end());
            TextOut(hdc, (rt.left + 47) + coords[i].first, rt.bottom - moveConst
```

```
+ 10, temp.c_str(), 3);
                //ось Y
                DrawLine(hdc, rt.left + 45, (rt.bottom - moveConst) - coords[i].second, rt.left + 55, (rt.bottom - moveConst) - coords[i].second, RGB(0, 0, 0), 2);
                tempstr = format("{}", funcAnalysisData[i].second);
                temp = std::wstring(tempstr.begin(), tempstr.end());
                TextOut(hdc, rt.left + 20, (rt.bottom - moveConst - 10) - coords[i].second, temp.c_str(), 3);
                }

                //точки
                for (int i = 0; i < dataCount; i++) {
                DrawDot(hdc, rt.left + 50 + coords[i].first, rt.bottom - moveConst - coords[i].second, RGB(255, 0, 0), 6);
                }

                //линии
                int prevX = rt.left + 50, prevY = rt.bottom - moveConst;   //начало оси
                for (int i = 0; i < dataCount; i++) {
                DrawLine(hdc, prevX, prevY, rt.left + 50 + coords[i].first, rt.bottom - moveConst - coords[i].second, RGB(0, 0, 255), 3);
                prevX = (rt.left + 50) + coords[i].first;
                prevY = (rt.bottom - moveConst) - coords[i].second;
                }
                EndPaint(hwnd, &ps);
        }
    }
```

Файл main.cpp

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

#include <windows.h>
#include "FuncView.h"
#include "DiagView.h"
#include "Utility.h"
#include "resource.h"

using namespace std;
```

```
int screenWidth = 800, screenHeight = 800;
int ID;

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);
BOOL CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM);
//точка входа
int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow)
{
  DialogBoxParam(hInstance, MAKEINTRESOURCE(IDD_DIALOG1), 0, (DLGPROC)DlgProc, 0);
    const wchar_t CLASS_NAME[] = L"Sample Window Class";
    //Регистрация окна

    WNDCLASS wc = { };

    wc.lpfnWndProc = WindowProc;        //процедура окна
    wc.hInstance = hInstance;          //дескриптор приложения
    wc.lpszClassName = CLASS_NAME;          //класс окна

    RegisterClass(&wc);

    //Создание окна

    HWND hwnd = CreateWindowEx(
        0,                      //Доп. стили (прим. прозрачные окна)
        CLASS_NAME,                     //Имя класса
        L"Program",                 //Заголовок окна
        WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX | WS_CLIPCHILDREN,           //Стиль окна

        // Координаты, ширина, высота окна
        CW_USEDEFAULT, CW_USEDEFAULT, screenWidth, screenHeight,

        NULL,
        NULL,
        hInstance,
        NULL
    );

    if (hwnd == NULL)
    {
```

```
        return 0;
    }

    ShowWindow(hwnd, nCmdShow);

    // все работает на сообщениях
    MSG msg = { };
    while (GetMessage(&msg, NULL, 0, 0) > 0)
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}

//процедура для обработки тех самых сообщений
LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{

    switch (uMsg)
    {
    case WM_CREATE:

        if (ID != IDCANCEL) {
        GetClientRect(hwnd, &func::rt);
        func::PrepData();
        }
        else {
        GetClientRect(hwnd, &diag::rt);
        diag::PrepData();
        }

        return 0;

    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;

    case WM_PAINT:
    {
        if (ID != IDCANCEL) {
        func::Paint(hwnd);
        }
```

```
            else {
            diag::Paint(hwnd);
            }
        }
        return 0;

        }
        return DefWindowProc(hwnd, uMsg, wParam, lParam);
    }
    BOOL CALLBACK DlgProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {
        switch (msg) {
        case WM_INITDIALOG:
            break;
        case WM_COMMAND:
            ID = LOWORD(wParam);

            OpenFile(hwnd);

            EndDialog(hwnd, 0);
            break;
        case WM_CLOSE:
            EndDialog(hwnd, 0);
            return 0;
        }
        return 0;
    }
```