НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА МАТЕМАТИЧЕСКОГО И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине: «Технологии программирования»

«Разработка классических приложений Windows»

Выполнил студент гр. А-16-20

Симаков А. М.

Принял преподаватель

А.В. Князев

Москва 2022

**Задание.**

* **Общее задание:**

Разработать классическое приложение Windows в среде Visual Studio, реализующее указанное задание.

Программа должна обеспечивать ввод исходных данных из файла и с клавиатуры.

Программа должна обеспечивать представление исходных данных и результатов в графическом виде.

Должны быть изображены оси системы координат (с центром в середине окна).

Отчёт по лабораторной работе должен содержать:

Титульный лист

Задание на работу (общее и индивидуальное)

Описание работы программы

Алгоритмы выполнения основных операций на псевдокоде

Тесты

Распечатки экранов при работе программы

Листинг программы

**Индивидуальное задание для варианта 11:**

Задана окружность и множество отрезков. Если ни один из отрезков не лежит внутри окружности, то найти отрезок наибольшей длины. В противном случае найти количество отрезков, пересекающих окружность.

* **Описание работы функций:**

**Функции классического приложения Windows**

*WinMain* – основная функция приложения

*WndProc* - функция обработки «сообщений» получаемых от Windows при возникновении «событий»

Структура *WNDCLASSEX* хранит информацию об окне: цвет фона, иконка приложения

*CreateWindowEx* – функция создания окна, возвращает handle, с помощью которого Windows отслеживает окно

Функция *ShowWindow* делает окно видимым

Для получения «сообщений» от Windows в *WinMain* используется цикл, получающий и передающий их в функцию *WndProc*.

«Сообщение» *WM\_PAINT*, вызывается при первом отображении окна приложения, а также при событиях, требующих обновления окна.

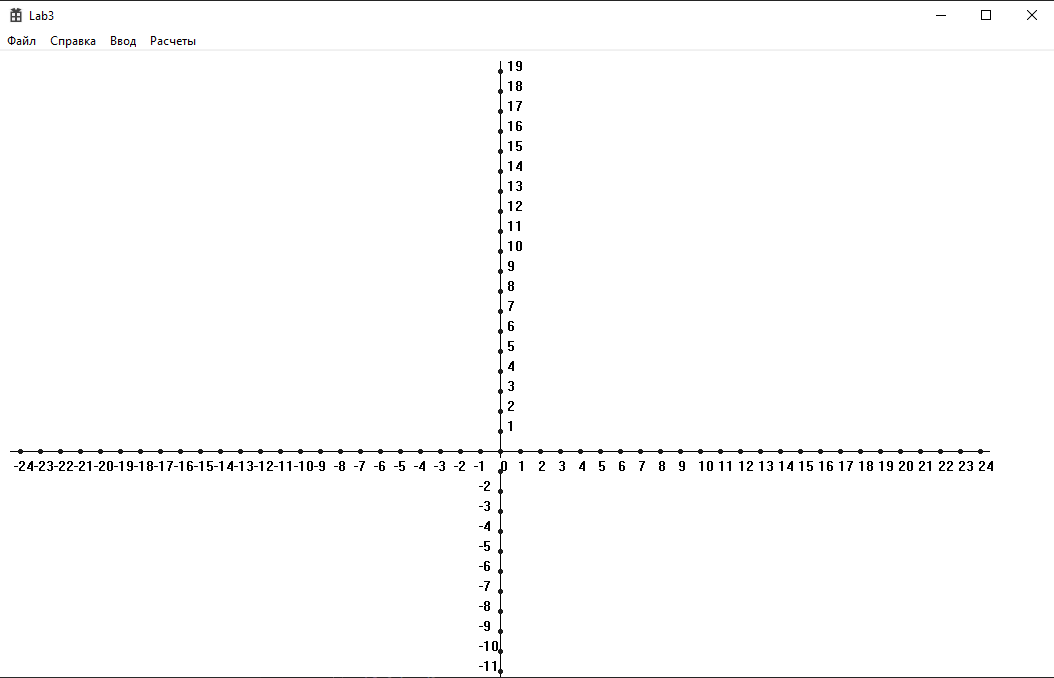
Функция *BeginPaint* подготавливает окно к отображению графической информации и заполняет информацией структуру типа *PAINTSTRUCT*

Функция *EndPaint* сообщает об окончании отрисовки графической информации в окно.

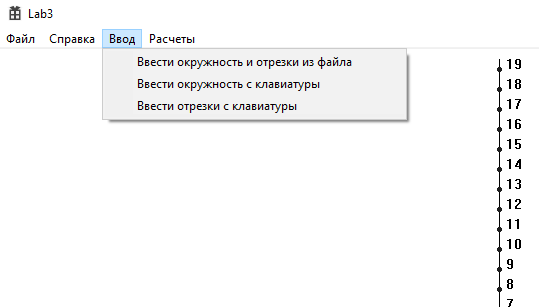
Функция *TextOut* отображает текст

**Выполнение заданий лабораторной работы**

При запуске нас встречает следующее окно с системой координат:

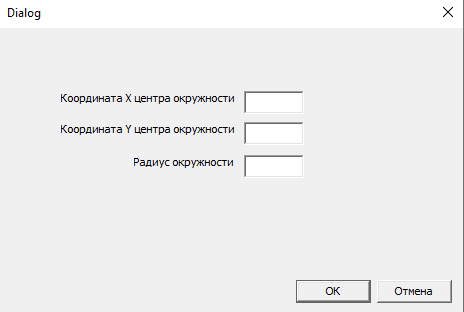


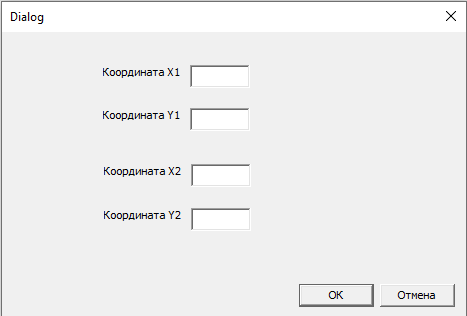
Чтобы ввести данные делаем следующее:



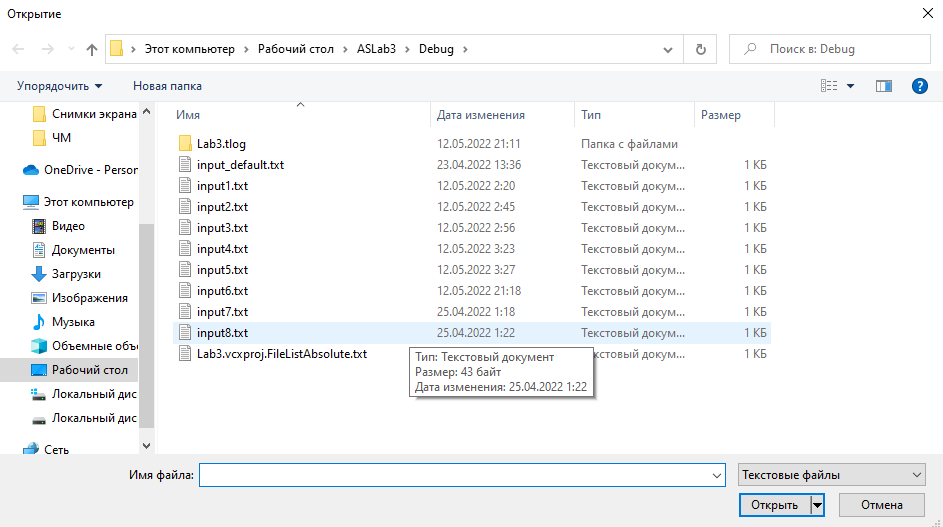
Далее выбираем тот пункт, который нас интересует

В зависимости от сделанного выбора можем лицезреть следующие окна:





Такие в случае, если данные вводятся произвольно



И такое окно с выбором файла, если используются уже готовые данные

Для выполнения расчетов 11 варианта лабораторной работы используется пункт меню «Расчеты» -> «Лаб 3 Вар 11»

Объекты *circle, segment* хранят в программе окружности и точки соответственно.

* **Алгоритмы выполнения основных операций на псевдокоде**

**Ввод окружности**

НАЧАЛО

ВВОД количество окружностей

ВВОД координат центра, радиус.

добавить окружность в массив окружностей с преобразованием координат из пользовательского представления в программное.

КОНЕЦ

**Ввод отрезков**

НАЧАЛО

ВВОД количество отрезков

ЦИКЛ от 1 до количество отрезков

НАЧАЛО

Ввод координаты концов отрезков

добавить отрезок в массив точек с преобразованием координат из пользовательского представления в программное.

КОНЕЦ

КОНЕЦ

**Поиск отрезка наибольшей длины**

НАЧАЛО

max\_l := 0.0

tmp := 0.0

ЦИКЛ от 1 до количество отрезков

НАЧАЛО

tmp := Длина отрезка

Если (tmp >= max\_l)

НАЧАЛО

max\_l = tmp

x1 = out\_x1

y1 = out\_y1

x2 = out\_x2

y2 = out\_y2

КОНЕЦ

КОНЕЦ

КОНЕЦ

**Наличие отрезка внутри окружности**

НАЧАЛО

res := false

ЦИКЛ от 1 до количество отрезков и НЕ res

НАЧАЛО

res = ((pow(sgm.out\_x1- crc.out\_x(), 2) + pow(sgm.out\_y1- crc.out\_y(), 2)) <= pow(crc.out\_r(), 2)) &&

((pow(sgm.out\_x2- crc.out\_x(), 2) + pow(sgm.out\_y2- crc.out\_y(), 2)) <= pow(crc.out\_r(), 2))

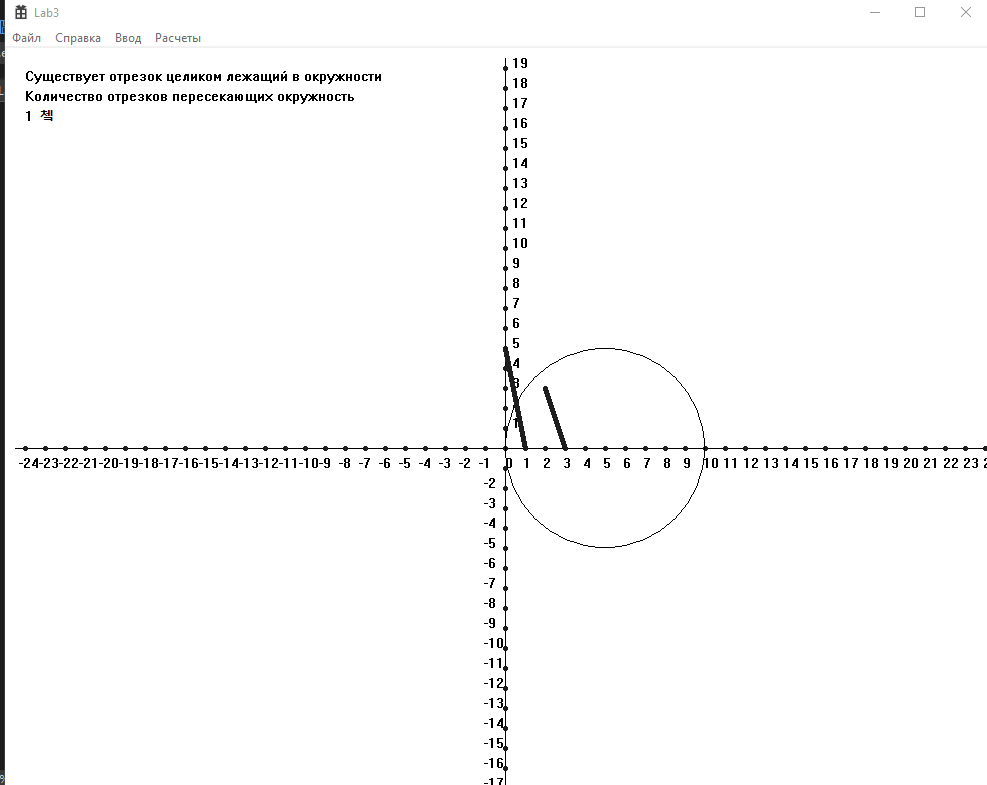
КОНЕЦ

Вернуть res

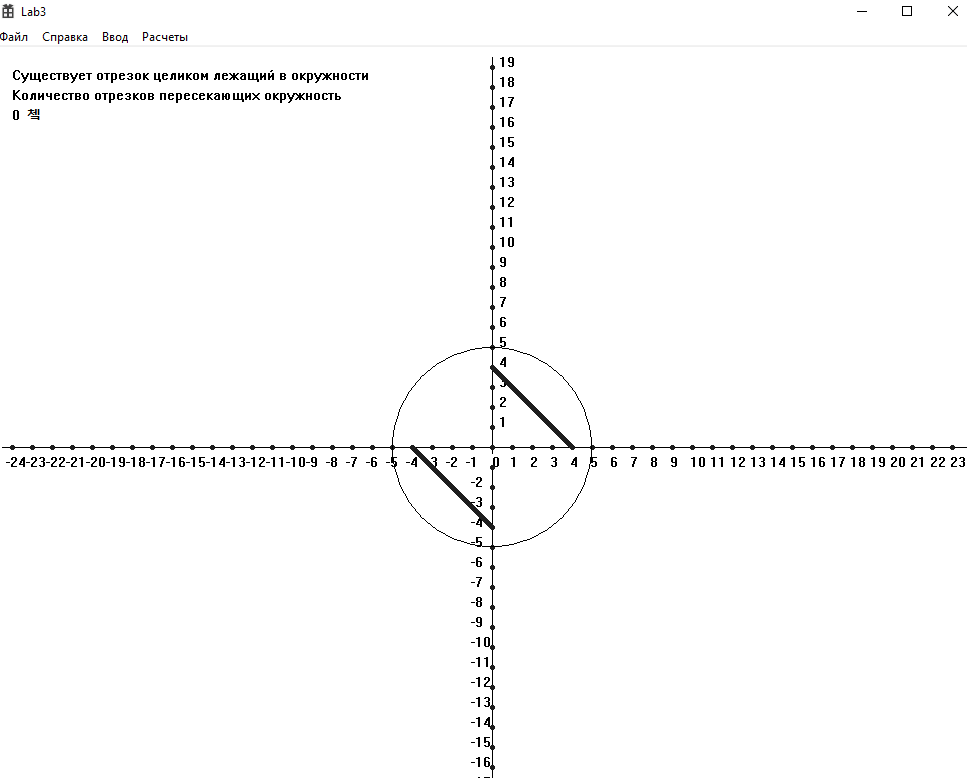
КОНЕЦ

* **Тесты**

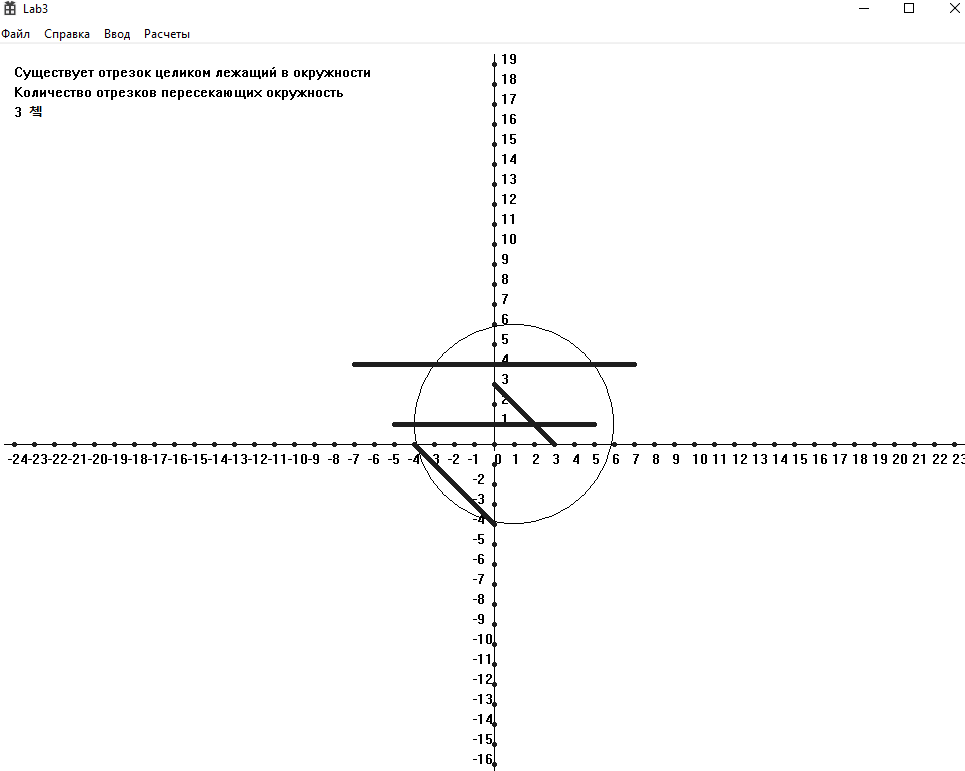
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 1 | 5  0 5 2  3 0 2  3 0 5  1 0 6  3 4 10  3  5 1  2 2  3 3 | Существует отрезок целиком лежащий в окружности  Количество отрезков пересекающих окружность 1 | Проверка корректности работы программы |



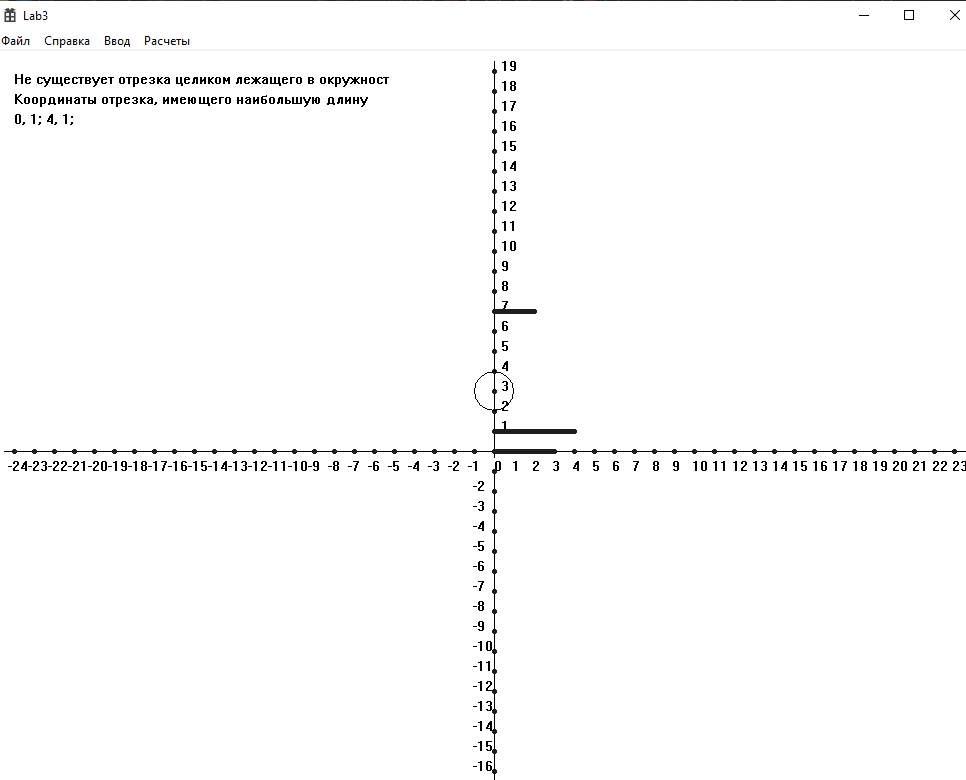
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 2 | 0 0 5  2  4 0 0 4  -4 0 0 -4 | Существует отрезок целиком лежащий в окружности  Количество отрезков пересекающих окружность 0 | Проверка корректности работы программы для случая без пересечений |



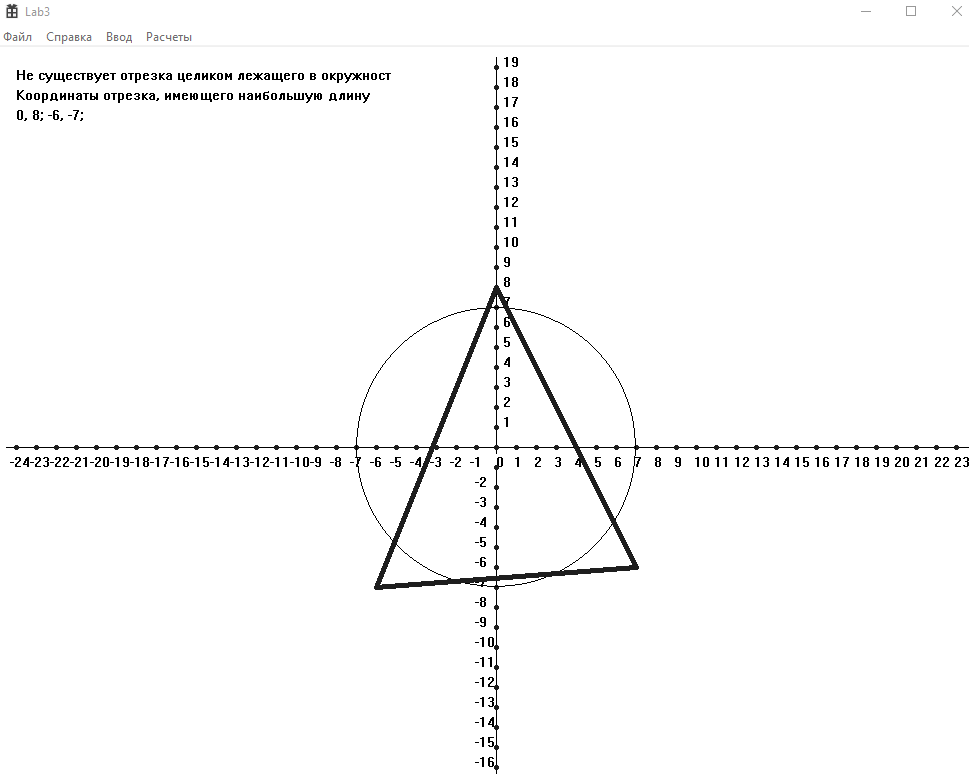
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 3 | 1 1 5  4  3 0 0 3  -4 0 0 -4  -5 1 5 1  -7 4 7 4 | Существует отрезок целиком лежащий в окружности  Количество отрезков пересекающих окружность 3 | Проверка корректности работы программы для случая, когда много отрезков, пересекающих окружность. При этом некоторые пересекают два раза |



|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 4 | 0 3 1  3  0 0 3 0  0 1 4 1  0 7 2 7 | Не существует отрезка, целиком лежащего в окружности  Координаты отрезка, имеющего наибольшую длину 0, 1; 4, 1; | Проверка корректности работы программы для случая, когда нет отрезка, полностью лежащего в окружности |



|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 5 | 0 0 7  3  0 8 -6 -7  0 8 7 -6  -6 -7 7 -6 | Не существует отрезка, целиком лежащего в окружности  Координаты отрезка, имеющего наибольшую длину 0, 8; -6, -7; | Проверка корректности работы программы для случая, когда нет отрезка, полностью лежащего в окружности. При этом все отрезки пересекают окружность |



**Листинг программы**

circlePoint.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include "circlePoint.h"

using namespace std;

// Вводит координаты окружности с преобразованием к новой системе координат.

int circle::inp(FILE\* file, const int Width, const int Heigh)

{

int num\_circles = 0;

int x = 0, y = 0, r = 0;

fscanf(file, "%d %d %d", &x, &y, &r);

add(Width / 2 + 20 \* x, Heigh / 2 - 20 \* y, 20 \* r);

return 0;

}

// Ввести отрезки с преобразованием к новой системе коорд.

int segment::inp(FILE\* file, const int Width, const int Heigh) {

int num\_points = 0;

fscanf(file, "%d", &num\_points);

int x1 = 0, y1 = 0, x2 = 0, y2 = 0;

for (int i = 0; i < num\_points; i++) {

fscanf(file, "%d %d %d %d", &x1, &y1, &x2, &y2);

add(Width / 2 + 20\*x1, Heigh / 2 - 20\*y1, Width / 2 + 20 \* x2, Heigh / 2 - 20 \* y2);

}

return 0;

}

// Вчисл. длину i-го отрезка.

double segment::length(int i) {

return (sqrt(pow(out\_x1(i) - out\_x2(i), 2) + pow(out\_y1(i) - out\_y2(i), 2)));

}

// Найти отрезок наибольшей длины.

void segment::lngest(int& x1, int& y1, int& x2, int& y2) {

double max\_l = 0.0;

double tmp = 0.0;

for (int i = 0; i < out\_num\_sgm(); i++) {

tmp = length(i);

if (tmp >= max\_l) {

max\_l = tmp;

x1 = out\_x1(i);

y1 = out\_y1(i);

x2 = out\_x2(i);

y2 = out\_y2(i);

}

}

return;

}

// true, если сущ. отрезок в окр. иначе false

bool is\_sgm\_in\_crc(circle& crc, segment& sgm) {

bool res = false;

for (int i = 0; (i < sgm.out\_num\_sgm()) && (!res); i++) {

res = ((pow(sgm.out\_x1(i) - crc.out\_x(), 2) + pow(sgm.out\_y1(i) - crc.out\_y(), 2)) <= pow(crc.out\_r(), 2)) &&

((pow(sgm.out\_x2(i) - crc.out\_x(), 2) + pow(sgm.out\_y2(i) - crc.out\_y(), 2)) <= pow(crc.out\_r(), 2));

}

return res;

}

bool inters(circle& crc, int sgm\_num, segment& sgm) {

bool res\_in = false; // У отрезка есть точка внутри окр.

bool res\_out = false; // У отрезка есть точка снаружи окр.

double max\_x = fmax(sgm.out\_x1(sgm\_num), sgm.out\_x2(sgm\_num));

double min\_x = fmin(sgm.out\_x1(sgm\_num), sgm.out\_x2(sgm\_num));

double max\_y = fmax(sgm.out\_y1(sgm\_num), sgm.out\_y2(sgm\_num));

double min\_y = fmin(sgm.out\_y1(sgm\_num), sgm.out\_y2(sgm\_num));

double x = min\_x;

double y = min\_y;

double step\_x = (max\_x - min\_x) / 10.0;

double step\_y = (max\_y - min\_y) / 10.0;

for (; (x <= max\_x) && (y <= max\_y) && (!res\_in || !res\_out); x += step\_x, y += step\_y) {

res\_in = res\_in || ((pow(x - crc.out\_x(), 2) + pow(y - crc.out\_y(), 2)) < pow(crc.out\_r(), 2));

res\_out = res\_out || ((pow(x - crc.out\_x(), 2) + pow(y - crc.out\_y(), 2)) > pow(crc.out\_r(), 2));

}

return (res\_in && res\_out);

}

int num\_inter\_sgm(circle& crc, segment& sgm) { // Возвращает количество отрезков, пересек. окр.

int num = 0;

for (int i\_sgm = 0; i\_sgm < sgm.out\_num\_sgm(); i\_sgm++) {

if (inters(crc, i\_sgm, sgm)) {

num++;

}

}

return num;

}

circlePoint.h

#pragma once

#include <cstdio>

#include <math.h>

struct circle\_data {

int x;

int y;

int r;

};

class segment;

// Класс окружности

class circle {

public:

circle() { circ = new circle\_data; }

~circle() { delete[] circ; }

void add(int x, int y, int r) { circ->x = x; circ->y = y; circ->r = r; }

int out\_x() const { return circ->x; }

int out\_y() const { return circ->y; }

int out\_r() const { return circ->r; }

friend bool is\_sgm\_in\_crc(circle& crc, segment& sgm); // true, если сущ. отрезок в окр. иначе false

friend int num\_inter\_sgm(circle& crc, segment& sgm); // Возвращает количество отрезков, пересек. окр.

friend bool inters(circle& crc, int sgm\_num, segment& sgm); // true, если sgm пересекает crc

// Вводит координаты окружности с преобразованием к новой системе координат.

int inp(FILE\* file, const int Width, const int Heigh);

private:

circle\_data\* circ;

};

struct segm\_data {

int x1;

int y1;

int x2;

int y2;

};

// Представляет массив отрезков.

class segment {

public:

segment() { num\_sgm = 0; sgm = new segm\_data[20]; }

~segment() { delete[] sgm; }

void add(int x1, int y1, int x2, int y2)

{

sgm[num\_sgm].x1 = x1; sgm[num\_sgm].y1 = y1;

sgm[num\_sgm].x2 = x2; sgm[num\_sgm].y2 = y2; num\_sgm++;

}

int out\_x1(int i) const { return sgm[i].x1; }

int out\_y1(int i) const { return sgm[i].y1; }

int out\_x2(int i) const { return sgm[i].x2; }

int out\_y2(int i) const { return sgm[i].y2; }

int out\_num\_sgm() const { return num\_sgm; }

int inp(FILE\* file, const int mainWinWidth, const int mainWinHeigh); // Ввести точки.

double length(int i); // Вчисл. длину i-го отрезка.

// Найти отрезок наибольшей длины.

void lngest(int& x1, int& y1, int& x2, int& y2);

friend bool is\_sgm\_in\_crc(circle& crc, segment& sgm); // true, если сущ. отрезок в окр. иначе false

friend int num\_inter\_sgm(circle& crc, segment& sgm); // Возвращает количество отрезков, пересек. окр.

friend bool inters(circle& crc, int sgm\_num, segment& sgm); // true, если sgm пересекает crc

private:

int num\_sgm;

segm\_data\* sgm;

};

bool is\_sgm\_in\_crc(circle& crc, segment& sgm); // true, если отрезок в окр. иначе false

int num\_inter\_sgm(circle& crc, segment& sgm); // Возвращает количество отрезков, пересек. окр.

drawing.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "drawing.h"

int draw\_os\_coord(HDC hdc, const int l\_padding, const int up\_padding, const int mainWinWidth, const int mainWinHeigh) {

HPEN hPen, hOldPen;

// Ox

MoveToEx(hdc, 10, mainWinHeigh / 2, NULL);

LineTo(hdc, mainWinWidth - 10, mainWinHeigh / 2);

// Oy

MoveToEx(hdc, mainWinWidth / 2, 10, NULL);

LineTo(hdc, mainWinWidth / 2, mainWinHeigh - 10);

hPen = CreatePen(PS\_SOLID, 5, RGB(30, 30, 30));

hOldPen = (HPEN)SelectObject(hdc, hPen);

// градуирование Ox

wchar\_t buffer[20];

for (int x = 0, y = mainWinHeigh / 2; x <= (mainWinWidth - 10) / 2; x += 20) {

if (x == 0) {

TextOut(hdc, mainWinWidth / 2 + x, mainWinHeigh / 2.0 + 7, \_ltow(x / 20, buffer, 10), 2);

} else {

TextOut(hdc, mainWinWidth / 2 + x - 2, mainWinHeigh / 2.0 + 7, \_ltow(x / 20, buffer, 10), 2);

TextOut(hdc, mainWinWidth / 2 - x - 6, mainWinHeigh / 2.0 + 7, \_ltow(-x / 20, buffer, 10), 3);

}

MoveToEx(hdc, mainWinWidth / 2 + x, mainWinHeigh / 2, NULL);

LineTo(hdc, mainWinWidth / 2 + x, mainWinHeigh / 2);

MoveToEx(hdc, mainWinWidth / 2 - x, mainWinHeigh / 2, NULL);

LineTo(hdc, mainWinWidth / 2 - x, mainWinHeigh / 2);

}

// градуирование Oy

for (int x = mainWinWidth / 2, y = 0; y <= (mainWinHeigh - 10) / 2; y += 20) {

if (y != 0) {

TextOut(hdc, mainWinWidth / 2 + 7, mainWinHeigh / 2.0 + 7 - y - 20, \_ltow(y / 20, buffer, 10), 2);

if( y != 20)

TextOut(hdc, mainWinWidth / 2 - 21, mainWinHeigh / 2.0 + 7 + y - 20, \_ltow(-y / 20, buffer, 10), 3);

} else {

//TextOut(hdc, mainWinWidth / 2 + 7, mainWinHeigh / 2.0 + 7 - y, \_ltow(3, buffer, 10), 2);

}

MoveToEx(hdc, mainWinWidth / 2, mainWinHeigh / 2 + y, NULL);

LineTo(hdc, mainWinWidth / 2, mainWinHeigh / 2 + y);

MoveToEx(hdc, mainWinWidth / 2, mainWinHeigh / 2 - y, NULL);

LineTo(hdc, mainWinWidth / 2, mainWinHeigh / 2 - y);

}

return 0;

}

// Нарисовать круг.

int draw\_circ(HWND hWnd, circle &circ) {

HPEN hPen = CreatePen(PS\_SOLID, 15, RGB(30, 100, 30));

HGDIOBJ hgBrush = GetStockObject(HOLLOW\_BRUSH);

HDC hdc;

//hgBrush = GetStockObject(HOLLOW\_BRUSH);

//hPen = CreatePen(PS\_SOLID, 15, RGB(30, 100, 30));

hdc = GetDC(hWnd);

// Создание прозрачной кисти

SelectObject(hdc, hgBrush);

Ellipse(hdc, circ.out\_x() - circ.out\_r(), circ.out\_y() - circ.out\_r(), circ.out\_x() + circ.out\_r(), circ.out\_y() + circ.out\_r());

ReleaseDC(hWnd, hdc);

DeleteObject(hPen);

DeleteObject(hgBrush);

return 0;

}

// Нарисовать отрезки.

int draw\_sgm(HWND hWnd, segment &sgm) {

HPEN hPen = CreatePen(PS\_SOLID, 15, RGB(30, 30, 30));

for (int i = 0; i < sgm.out\_num\_sgm(); i++) {

HDC hdc = GetDC(hWnd);

hPen = CreatePen(PS\_SOLID, 5, RGB(30, 30, 30));

SelectObject(hdc, hPen);

MoveToEx(hdc, sgm.out\_x1(i), sgm.out\_y1(i), NULL);

LineTo(hdc, sgm.out\_x2(i), sgm.out\_y2(i));

//MoveToEx(hdc, x, y, NULL);

//LineTo(hdc, x, y);

ReleaseDC(hWnd, hdc);

}

DeleteObject(hPen);

return 0;

}

drawing.h

#pragma once

#include <Windows.h>

#include <commdlg.h>

#include "circlePoint.h"

int draw\_os\_coord(HDC hdc, const int l\_padding, const int up\_padding, const int mainWinWidth, const int mainWinHeigh);

// Нарисовать круги.

int draw\_circ(HWND hWnd, circle &circ);

// Нарисовать точки.

int draw\_sgm(HWND hWnd, segment &sgn);

framework.h

// header.h: включаемый файл для стандартных системных включаемых файлов

// или включаемые файлы для конкретного проекта

//

#pragma once

#include "targetver.h"

#define WIN32\_LEAN\_AND\_MEAN // Исключите редко используемые компоненты из заголовков Windows

// Файлы заголовков Windows

#include <windows.h>

// Файлы заголовков среды выполнения C

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

Lab3.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

// Lab3.cpp : Определяет точку входа для приложения.

//

#include <Windows.h>

#include <commdlg.h>

#include "framework.h"

#include "drawing.h"

#include "Lab3.h"

#include <cstdio>

#include <stdlib.h>

#include <string>

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB3, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB3));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW 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(hInstance, MAKEINTRESOURCE(IDI\_LAB3));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB3);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

10, 10, mainWinWidth, mainWinHeigh, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId){

case IDM\_LAB3: {

PAINTSTRUCT ps;

HDC hdc = GetDC(hWnd);

if (is\_sgm\_in\_crc(circ, sgm)) {

TextOut(hdc, 20, 20, L"Существует отрезок целиком лежащий в окружности", 47);

// Найти количество отрезков, пересекающих окружность.

int num\_inter = num\_inter\_sgm(circ, sgm);

TextOut(hdc, 20, 40, L"Количество отрезков пересекающих окружность", 43);

std::string buffer1 = std::to\_string(num\_inter);

std::wstring stemp = std::wstring(buffer1.begin(), buffer1.end());

TextOut(hdc, 20, 60, stemp.c\_str(), 3);

} else {

TextOut(hdc, 20, 20, L"Не существует отрезка целиком лежащего в окружности", 50);

// Найти отрезок наибольшей длины.

wchar\_t bufferX1[20];

wchar\_t bufferY1[20];

wchar\_t bufferX2[20];

wchar\_t bufferY2[20];

int x1 = 0, y1 = 0, x2 = 0, y2 = 0;

sgm.lngest(x1, y1, x2, y2);

\_ltow((x1 - mainWinWidth / 2) / 20, bufferX1, 10);

\_ltow((y1 - mainWinHeigh / 2) / (-20), bufferY1, 10);

\_ltow((x2 - mainWinWidth / 2) / 20, bufferX2, 10);

\_ltow((y2 - mainWinHeigh / 2) / (-20), bufferY2, 10);

wcscat(bufferX1, L", ");

wcscat(bufferY1, L"; ");

wcscat(bufferX2, L", ");

wcscat(bufferY2, L"; ");

wcscat(bufferX1, bufferY1);

wcscat(bufferX1, bufferX2);

wcscat(bufferX1, bufferY2);

TextOut(hdc, 20, 40, L"Координаты отрезка, имеющего наибольшую длину", 45);

TextOut(hdc, 20, 60, bufferX1, wcslen(bufferX1));

}

EndPaint(hWnd, &ps);

}

break;

case IDM\_OPEN: { // Пункт "Диалоговое окно работы с файлами"

BOOL PrFileName = GetFileNameOpen();

HDC hdc = GetDC(hWnd);

PAINTSTRUCT ps;

if (PrFileName) {

FILE\* file = \_wfopen(FileName, L"r");

int x = 0, y = 0, r = 0;

if (file)

{

circ.inp(file, mainWinWidth, mainWinHeigh); // Ввести круг.

draw\_circ(hWnd, circ); // Нарисовать круги.

sgm.inp(file, mainWinWidth, mainWinHeigh); // Ввести отрезки

draw\_sgm(hWnd, sgm);// Нарисовать отрезки.

}

EndPaint(hWnd, &ps);

fclose(file);

}

}

break;

case IDM\_INP\_OKR: { // Ввод окружности.

PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD\_INP\_OKR), hWnd, DlgInputOkr);

if (PrInput) {

x = \_wtoi(bufX);

y = \_wtoi(bufY);

r = \_wtoi(bufR);

x \*= 20;

y \*= -20;

r \*= 20;

circ.add(mainWinWidth / 2 + x, mainWinHeigh / 2 + y, r);

HPEN hPen;

HBRUSH hBrush;

HGDIOBJ hgBrush;

HDC hdc = GetDC(hWnd);

hPen = CreatePen(PS\_SOLID, 15, RGB(30, 100, 30));

//hBrush = CreateSolidBrush(RGB(200, 200, 200));

// Создание прозрачной кисти

hgBrush = GetStockObject(HOLLOW\_BRUSH);

SelectObject(hdc, hgBrush);

Ellipse(hdc, mainWinWidth / 2 + x - r, mainWinHeigh / 2 + y - r, mainWinWidth / 2 + x + r, mainWinHeigh / 2 + y + r);

ReleaseDC(hWnd, hdc);

}

}

break;

case IDM\_INP\_POI: { // Ввод отрезка.

PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD\_INP\_POI), hWnd, DlgInputPoi);

if (PrInput) {

x = \_wtoi(bufX);

y = \_wtoi(bufY);

x \*= 20;

y \*= -20;

int x2 = \_wtoi(bufX2);

int y2 = \_wtoi(bufY2);

x2 \*= 20;

y2 \*= -20;

sgm.add(mainWinWidth / 2 + x, mainWinHeigh / 2 + y, mainWinWidth / 2 + x2, mainWinHeigh / 2 + y2);

HPEN hPen;

HGDIOBJ hgBrush;

HDC hdc = GetDC(hWnd);

hPen = CreatePen(PS\_SOLID, 5, RGB(30, 30, 30));

//draw\_point(hdc, hPen, mainWinWidth, mainWinHeigh);

SelectObject(hdc, hPen);

MoveToEx(hdc, mainWinWidth / 2 + x, mainWinHeigh / 2 + y, NULL);

LineTo(hdc, mainWinWidth / 2 + x2, mainWinHeigh / 2 + y2);

ReleaseDC(hWnd, hdc);

}

}

break;

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

draw\_os\_coord(hdc, 10, 10, mainWinWidth, mainWinHeigh);

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

// Функция диалогового окна для примера.

INT\_PTR CALLBACK DlgInput(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDOK:

GetDlgItemText(hDlg, IDC\_X, bufX, 4);

GetDlgItemText(hDlg, IDC\_Y, bufY, 4);

EndDialog(hDlg, 1);

break;

case IDCANCEL:

EndDialog(hDlg, 0);

break;

}

return (INT\_PTR)TRUE;

}

return (INT\_PTR)FALSE;

}

// Функция диалогового окна для ввода окружности.

INT\_PTR CALLBACK DlgInputOkr(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDOK:

GetDlgItemText(hDlg, IDC\_X, bufX, 4);

GetDlgItemText(hDlg, IDC\_Y, bufY, 4);

GetDlgItemText(hDlg, IDC\_R, bufR, 4);

EndDialog(hDlg, 1);

break;

case IDCANCEL:

EndDialog(hDlg, 0);

break;

}

return (INT\_PTR)TRUE;

}

return (INT\_PTR)FALSE;

}

// Функция диалогового окна для ввода отрезка.

INT\_PTR CALLBACK DlgInputPoi(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDOK:

GetDlgItemText(hDlg, IDC\_X, bufX, 4);

GetDlgItemText(hDlg, IDC\_Y, bufY, 4);

GetDlgItemText(hDlg, IDC\_X2, bufX2, 4);

GetDlgItemText(hDlg, IDC\_Y2, bufY2, 4);

EndDialog(hDlg, 1);

break;

case IDCANCEL:

EndDialog(hDlg, 0);

break;

}

return (INT\_PTR)TRUE;

}

return (INT\_PTR)FALSE;

}

// Функция выбора имени файла для открытия. HANDLE

BOOL GetFileNameOpen() {

OPENFILENAMEW ofn;

//HWND hwnd;

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

//ofn.hwndOwner = hwnd;

TCHAR szFilter[100] = L"Текстовые файлы\0\*.txt;\*.doc\0Двоичные файлы\0 \* .exe\0";

FileName[0] = '\0';

memset(&ofn, 0, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.lpstrFilter = szFilter;

ofn.nFilterIndex = 1;

ofn.lpstrFile = FileName;

ofn.nMaxFile = sizeof(FileName);

if (GetOpenFileName(&ofn))

return true;

else

return false;

}

Lab3.h

#pragma once

#include "resource.h"

#include "circlePoint.h"

#define MAX\_LOADSTRING 100

// Прототип функции диалогового окна.

INT\_PTR CALLBACK DlgInput(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

// Функция диалогового окна для ввода окружности.

INT\_PTR CALLBACK DlgInputOkr(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

// Функция диалогового окна для ввода точки.

INT\_PTR CALLBACK DlgInputPoi(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

// Функция выбора имени файла для открытия.

BOOL GetFileNameOpen();

TCHAR bufX[5];

TCHAR bufY[5];

TCHAR bufX2[5];

TCHAR bufY2[5];

TCHAR bufR[5];

int PrInput = 0;

int x, y, r;

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

int mainWinWidth = 1000;

int mainWinHeigh = 800;

circle circ;

segment sgm;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

// Для диалогового окна для работы с файлами.

TCHAR FileName[200];

Resource.h

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется Lab3.rc

//

#define IDC\_MYICON 2

#define IDD\_LAB3\_DIALOG 102

#define IDS\_APP\_TITLE 103

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDI\_LAB3 107

#define IDI\_SMALL 108

#define IDC\_LAB3 109

#define IDR\_MAINFRAME 128

#define IDD\_INP\_OKR 129

#define IDD\_DIALOG1 130

#define IDD\_INP\_POI 130

#define IDC\_X 1000

#define IDC\_Y 1001

#define IDC\_R 1002

#define IDC\_X2 1002

#define IDC\_EDIT1 1003

#define IDC\_Y2 1003

#define IDC\_EDIT2 1004

#define ID\_32771 32771

#define ID\_32772 32772

#define ID\_32773 32773

#define IDM\_OPEN 32774

#define IDM\_INP\_OKR 32776

#define IDM\_INP\_POI 32777

#define ID\_32778 32778

#define IDM\_LAB3 32780

#define IDC\_STATIC -1

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 1

#define \_APS\_NEXT\_RESOURCE\_VALUE 131

#define \_APS\_NEXT\_COMMAND\_VALUE 32781

#define \_APS\_NEXT\_CONTROL\_VALUE 1005

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

targetver.h

#pragma once

// // При включении SDKDDKVer.h будет задана самая новая из доступных платформ Windows.

// Если вы планируете сборку приложения для предыдущей версии платформы Windows, включите WinSDKVer.h и

// задайте желаемую платформу в макросе \_WIN32\_WINNT, прежде чем включать SDKDDKVer.h.

#include <SDKDDKVer.h>