
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).

Индивидуальное задание

Задана окружность и множество точек. Если более 4 точек лежат внутри окружности, то найти количество точек, лежащих в первой четверти. В противном случае найти сумму расстояний точек до начала координат.

Описание работы программы

Осуществляется ввод исходных данных – с помощью диалоговых окон, либо же из файла. Данные изображаются. Затем пользователь может кликнуть на раздел меню «решить задачу» и, собственно, решить её.

1) Если больше 4-х точек лежат внутри окружности, то в окне выводится количество точек, лежащих в первой четверти.

2) Иначе выводится сумма расстояний от точек до начала координат.

Алгоритм выполнения операций на псевдокоде

Определение количества точек, принадлежащих окружности:

```
n-число точек
x_0, y_0 - массивы точек
l - число точек принадлежащих окружности
ЦИКЛ (i = 1, n)
    ЕСЛИ((x_0[i] - x)^2 + (y_0[i] - y)^2 < r^2) ТО
        ++l;
КОНЕЦ_ЦИКЛ
ВЫВЕСТИ l
```

Определение количества точек в первой четверти:

```
l - число точек принадлежащих первой четверти
n-число точек
```

```

x_0, y_0 - массивы точек
ЦИКЛ (I = 1, n)
    ЕСЛИ(x_0[i]> и y_0[i]>0) ТО
        ++1;
КОНЕЦ_ЦИКЛ
ВЫВЕСТИ 1;

```

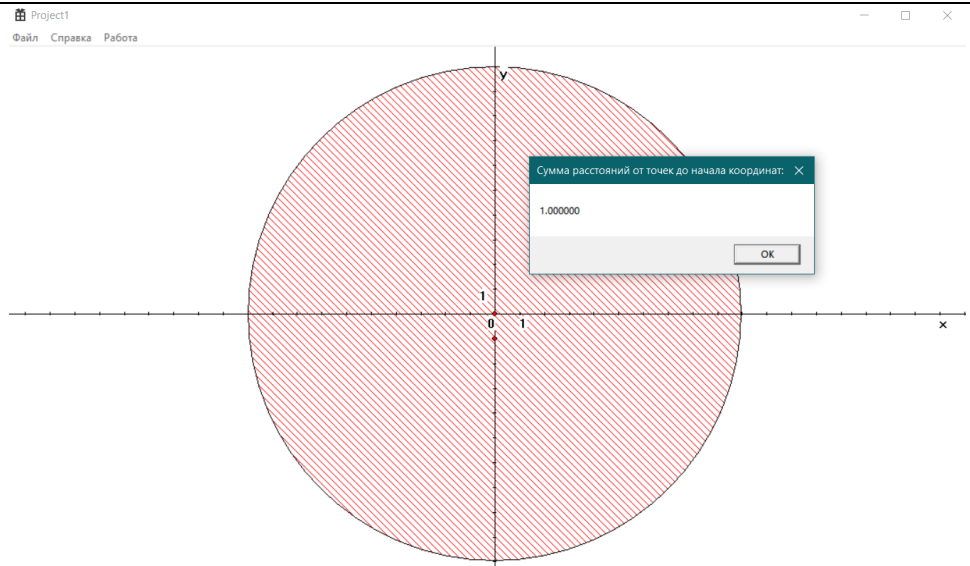
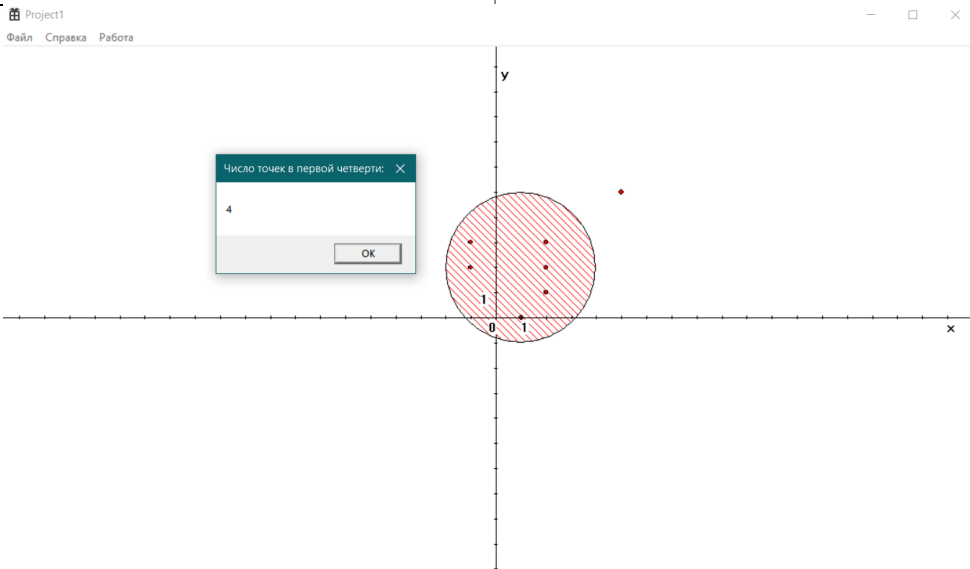
Определение суммы расстояний от точек до начала координат:

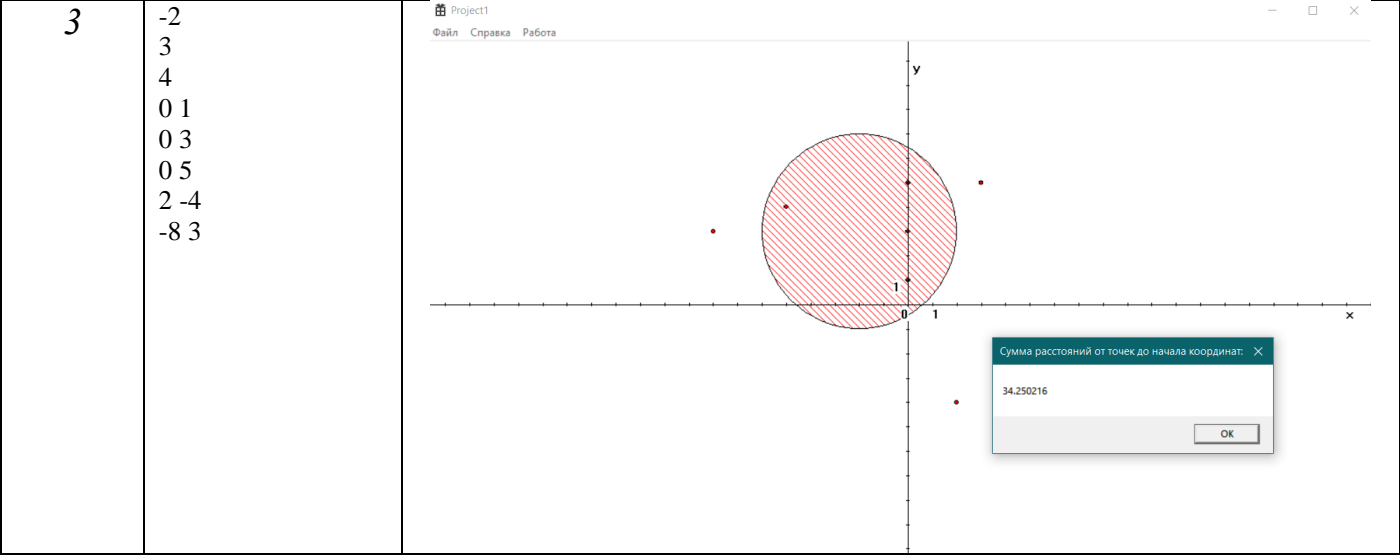
```

Sum - сумма расстояний
x_0, y_0 - массивы точек
n-число точек
ЦИКЛ (I = 1, n)
    SUM += sqrt(x_0[i]^2 + y_0[i]^2)
КОНЕЦ_ЦИКЛ
ВЫВЕСТИ SUM

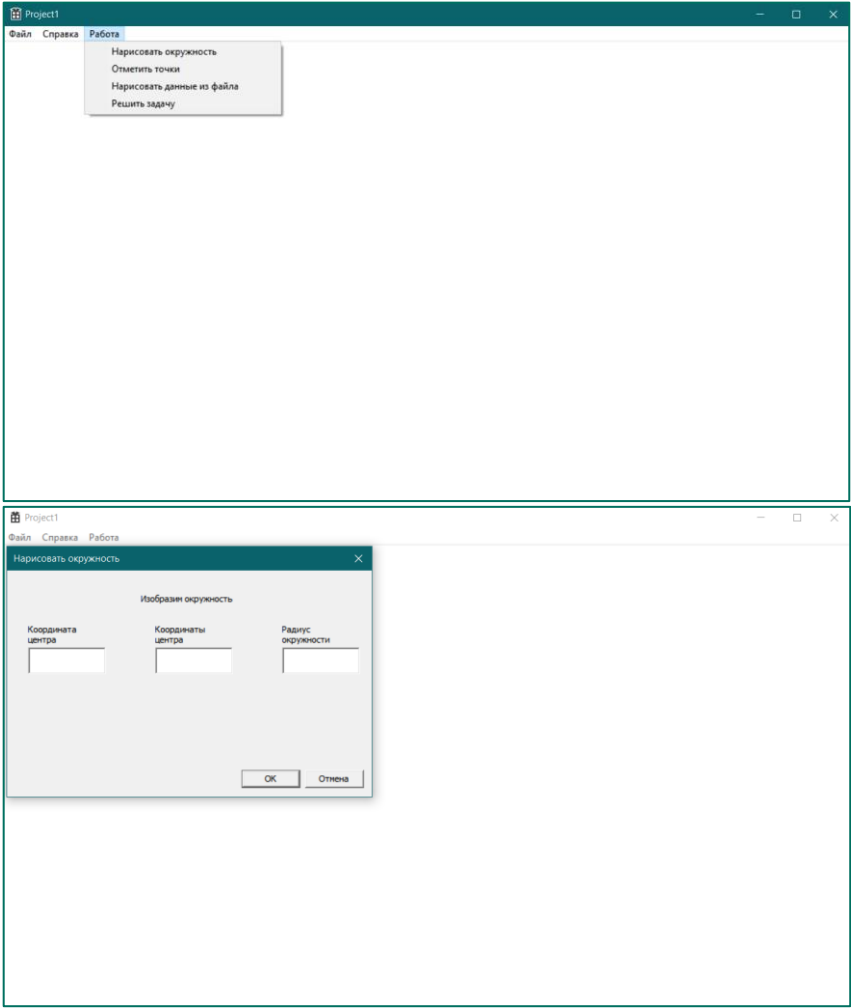
```

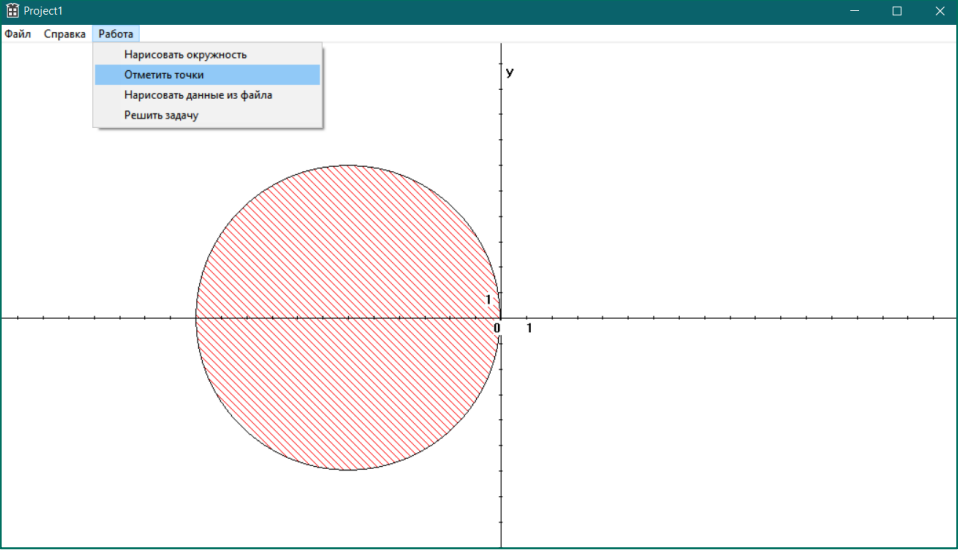
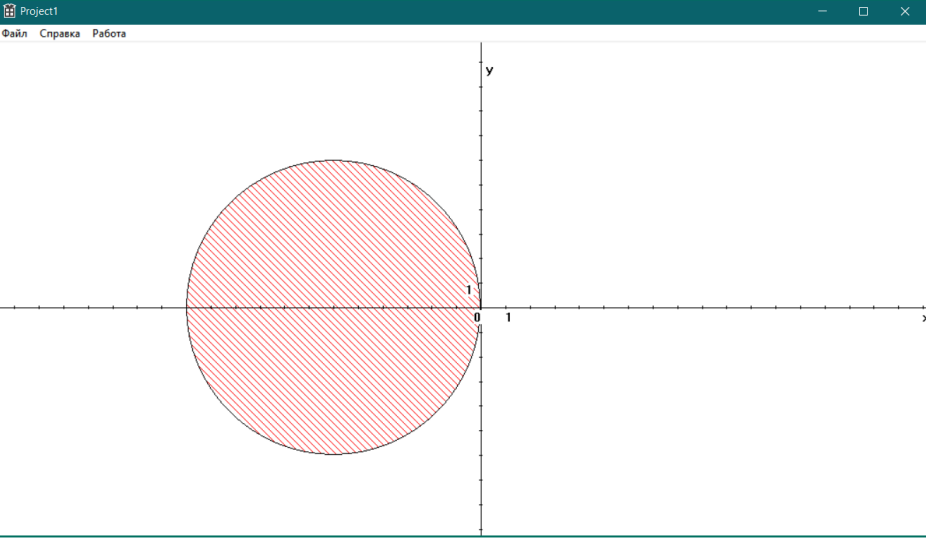
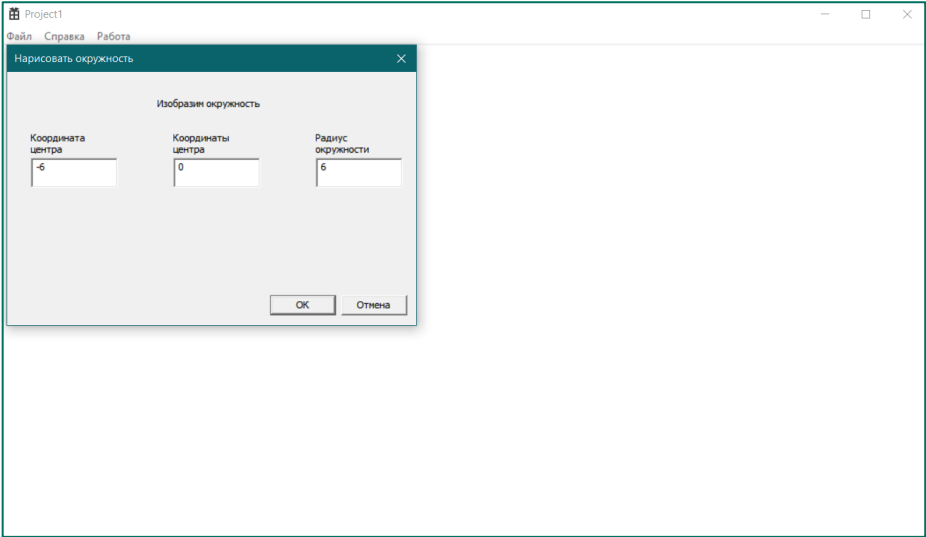
Тесты

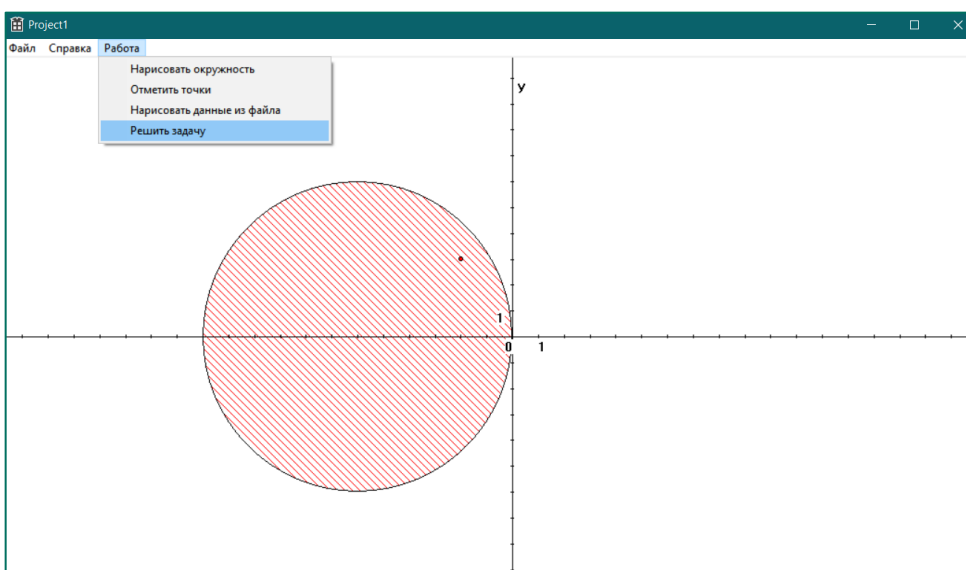
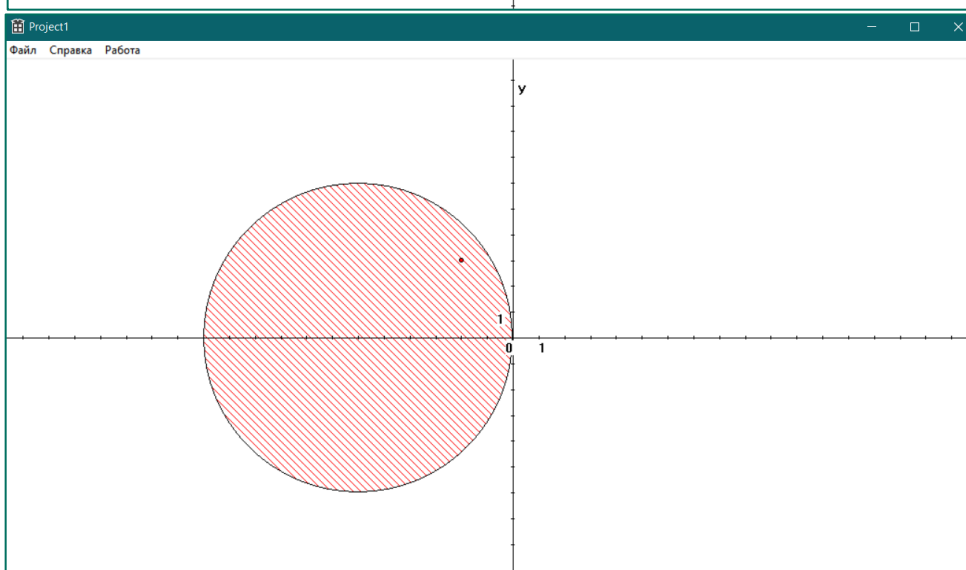
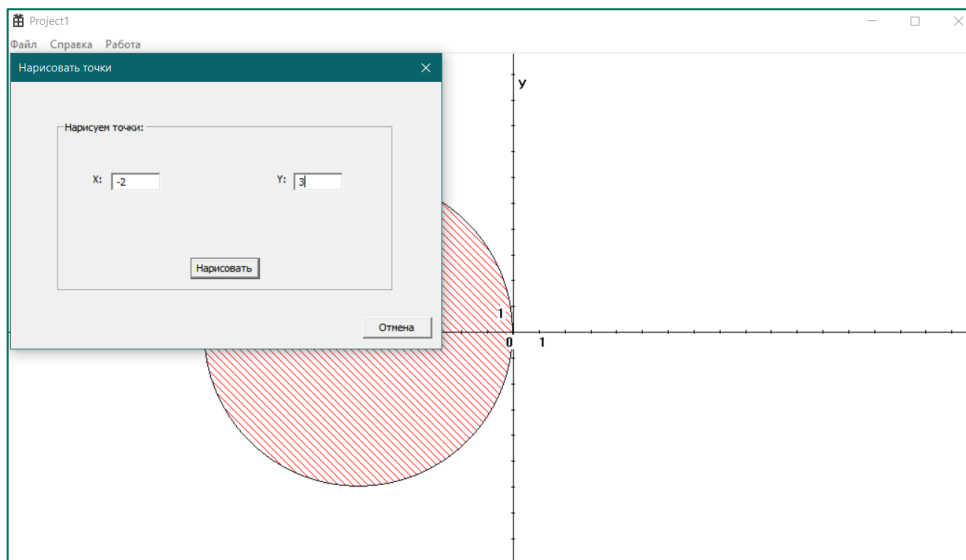
Тест №	Входные данные	Результат
1	0//x – координата центра 0//y – координата центра 10//радиус 0 0 //точки 0 -1 //точки	
2	1 2 3 1 0 2 1 2 2 2 3 -1 2 -1 3 5 5	

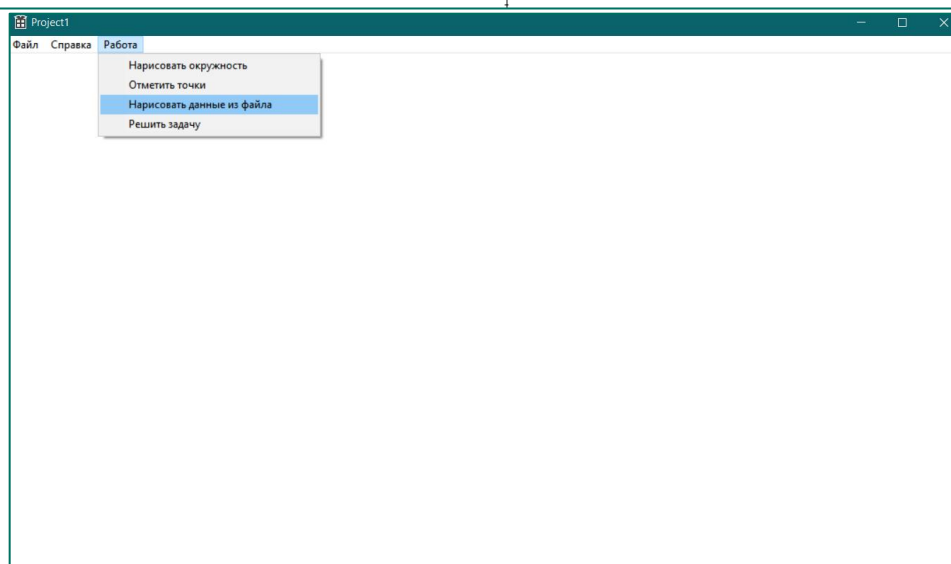
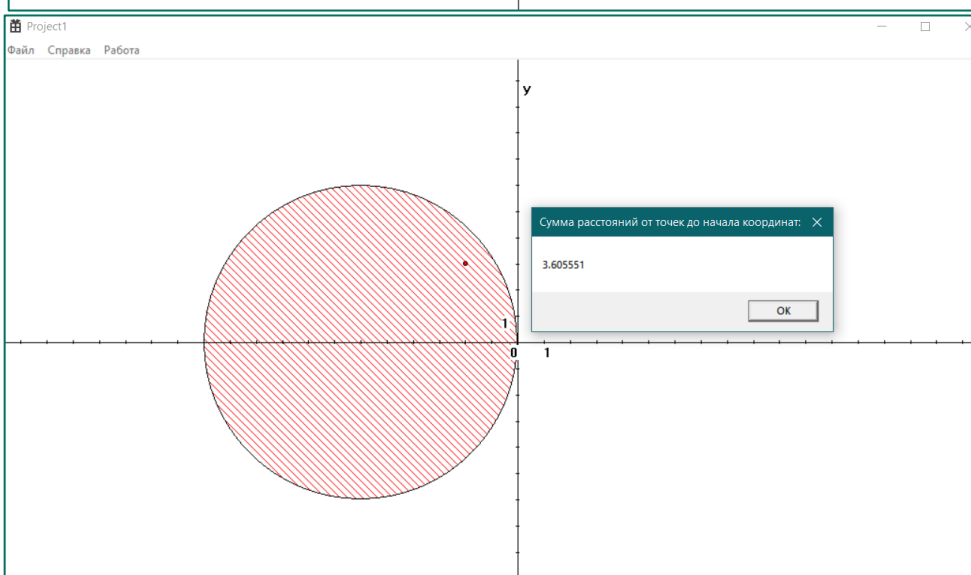
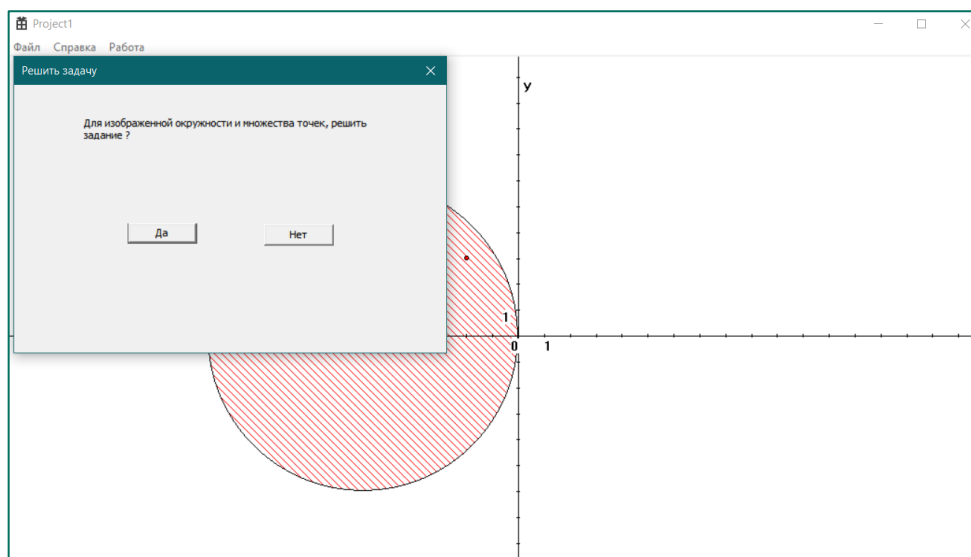


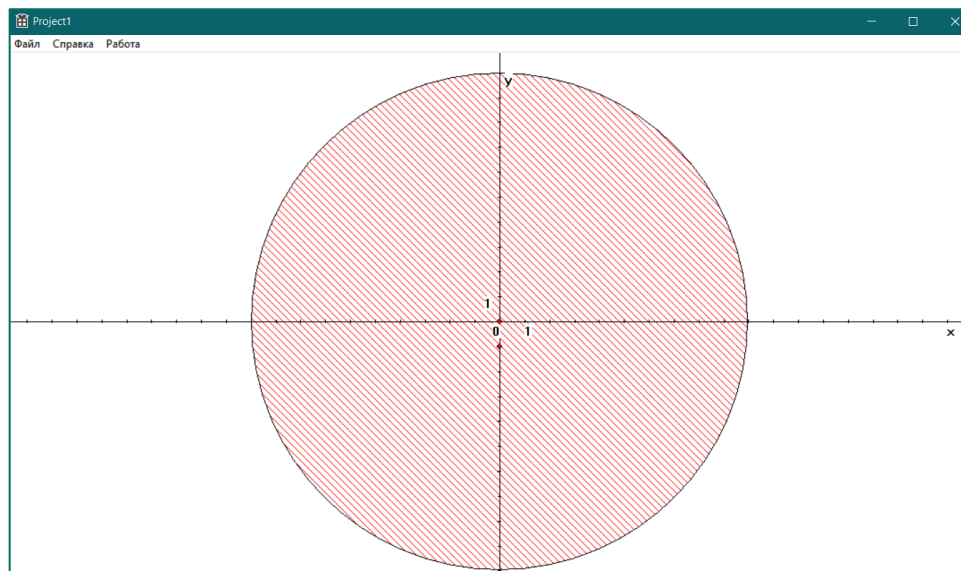
Распечатки экранов при работе программы











Листинг программы

```
#include "framework.h"
#include "Project1.h"
#include "iostream"
#include <vector>
#include <string>
#include <fstream>
using namespace std;
#define MAX_LOADSTRING 100

// Глобальные переменные:
HINSTANCE hInst;
WCHAR szTitle[MAX_LOADSTRING];
WCHAR szWindowClass[MAX_LOADSTRING];
TCHAR bufX[5];
TCHAR buf[32] = { 0 };
TCHAR buff[100];
TCHAR bufY[5];
double s;
char Buf[2];
char f[50];
TCHAR bufR[5];
ifstream file("TextFile1.txt");
int PrInput = 0;
vector <int> x_points;
vector <int> y_points;
int x, y, r;
int x_p, y_p;
HDC hdc;
int xView, yView;

typedef struct tagOFN
{
    DWORD lStructSize;
    LPCTSTR lpstrFilter;
    WORD nFilterIndex; // Должен быть равен 1
    LPTSTR lpstrFile;
    DWORD nMaxFile;
```

```
// текущий экземпляр
// Текст строки заголовка
// имя класса главного окна
```

```

        LPCTSTR lpstrInitialDir;
    } OPENFILENAME, * LOPENFILENAME;
    BOOL GetOpenFileName(LOPENFILENAME lpofn);
    HBRUSH hBrushSol, hOldBrush, hBrushBlue;
    bool check_4_points(int x, int y, int r, vector<int> x_points, vector<int>
y_points) {
        int l = 0;
        for (int i = 0; i < x_points.size(); i++) {
            if (pow(x_points[i] - x, 2) + pow(y_points[i] - y, 2) <= pow(r, 2)) {
                ++l;
            }
        }
        if (l > 4)
            return true;
        else
            return false;
    }
    int number_points_in_1_part(vector<int> x_points, vector<int> y_points) {
        int l = 0;
        for (int i = 0; i < x_points.size(); i++) {
            if ((x_points[i] > 0) and (y_points[i] > 0)) {
                ++l;
            }
        }
        return l;
    }
    double sum_length_to_center(vector<int> x_points, vector<int> y_points) {
        double sum = 0;
        for (int i = 0; i < x_points.size(); i++) {
            sum += sqrt(pow(x_points[i]/50, 2) + pow(y_points[i]/50, 2));
        }
        return sum;
    }
}

BOOL Line(HDC hdc, int x1, int y1, int x2, int y2)
{
    MoveToEx(hdc, x1, y1, NULL); //сделать текущими координаты x1, y1
    return LineTo(hdc, x2, y2);
}

// 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);

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_PROJECT1, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Выполнить инициализацию приложения:
    if (!InitInstance (hInstance, nCmdShow))

```



```

{
return FALSE;
}

HACCEL hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_PROJECT1));

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_PROJECT1));
wcex.hCursor        = LoadCursor(nullptr, IDC_ARROW);
wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
wcex.lpszMenuName    = MAKEINTRESOURCEW(IDC_PROJECT1);
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,
                                40, 0, 1100, 650, 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
    //
    //

    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_CIRCLE, bufX, 4);
                        GetDlgItemText(hDlg, IDC_Y_CIRCLE, bufY, 4);
                        GetDlgItemText(hDlg, IDC_R_CIRCLE, bufR, 4);
                        EndDialog(hDlg, 1);
                        break;
                    case IDCANCEL:
                        EndDialog(hDlg, 0);
                        break;
                }
                return (INT_PTR) TRUE;
        }
        return (INT_PTR) FALSE;
    }

    INT_PTR CALLBACK DlgInputP(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_POINTS, bufX, 4);
GetDlgItemText(hDlg, IDC_EDIT2, bufY, 4);
EndDialog(hDlg, 1);
break;
case IDCANCEL:
EndDialog(hDlg, 0);
break;
}
return (INT_PTR) TRUE;
}
return (INT_PTR) FALSE;
}
INT_PTR CALLBACK DlgInputTA(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_POINTS, bufX, 4);

GetDlgItemText(hDlg, IDC_EDIT2, bufY, 4);
EndDialog(hDlg, 1);
break;
case IDCANCEL:
EndDialog(hDlg, 0);
break;
}
return (INT_PTR) TRUE;
}
return (INT_PTR) FALSE;
}
INT_PTR CALLBACK DlgInputT(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:
EndDialog(hDlg, 1);
break;
case IDCANCEL:
EndDialog(hDlg, 0);
break;
}
return (INT_PTR) TRUE;
}
return (INT_PTR) FALSE;
}
}

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)
{
break;
case IDD_CIRCLE: // НОВЫЙ пункт меню
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_CIRCLE), hWnd, DlgInput);
if (PrInput)
{
x = _wtoi(bufX) * 50;
y = _wtoi(bufY) * 50;
r = _wtoi(bufR) * 50;
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL); //Начало координат
SetMapMode(hdc, MM_ISOTROPIC); //логические единицы отображаем, как  $\phi$ 
HBRUSH hBrush;
hBrush = CreateHatchBrush(HS_FDIAGONAL, RGB(255, 0, 0));
SelectObject(hdc, hBrush);
Ellipse(hdc, x+r, y+r, x-r, y-r);
Line(hdc, 0, 1000, 0, -1000); //ось Y
Line(hdc, -1000, 0, 1500, 0); //ось X
MoveToEx(hdc, 0, 0, NULL);
for (int i = -1000; i < 1000; i += 50)
{
Line(hdc, i, 3, i, -3);
Line(hdc, -3, i, 3, i);
}
TextOut(hdc, -15, -5, L"0", 1);
TextOut(hdc, 50, -5, L"1", 1);
TextOut(hdc, -30, 50, L"1", 1);
TextOut(hdc, 900, -5, L"x", 1);
TextOut(hdc, 10, 500, L"y", 1);
ReleaseDC(hWnd, hdc);
}
break;
case IDD_POINTS: // Изменённый пункт меню
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_POINTS), hWnd, DlgInputP);
if (PrInput)
{
x_p = _wtoi(bufX) * 50;
x_points.push_back(x_p);
y_p = _wtoi(bufY) * 50;
y_points.push_back(y_p);
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL); //Начало координат
SetMapMode(hdc, MM_ISOTROPIC); //логические единицы отображаем, как
физические
hBrushSol = CreateSolidBrush(RGB(255, 0, 0));
hOldBrush = (HBRUSH) SelectObject(hdc, hBrushSol);
Ellipse(hdc, x_p + 5, y_p + 5, x_p - 5, y_p - 5);
Line(hdc, 0, 1000, 0, -500); //ось Y
Line(hdc, -1000, 0, 1500, 0); //ось X
MoveToEx(hdc, 0, 0, NULL);
for (int i = -1000; i < 1000; i += 50)
{
Line(hdc, i, 3, i, -3);
Line(hdc, -3, i, 3, i);
}
TextOut(hdc, -15, -5, L"0", 1);

```

```

TextOut(hdc, 50, -5, L"1", 1);
TextOut(hdc, -30, 50, L"1", 1);
TextOut(hdc, 900, -5, L"x", 1);
TextOut(hdc, 10, 500, L"y", 1);

ReleaseDC(hWnd, hdc);
}
break;
case IDD_TASK:

PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_TASK), hWnd, DlgInputT);
if (PrInput) {
if (check_4_points(x, y, r, x_points, y_points)) {
int val = number_points_in_1_part(x_points, y_points);
swprintf_s(buff, TEXT("%d"), val);
MessageBox(NULL, buff, L"Число точек в первой четверти:", NULL);
}
else {
s = sum_length_to_center(x_points, y_points);
swprintf_s(buff, TEXT("%f"), s);
MessageBox(NULL, buff, L"Сумма расстояний от точек до начала координат:",
NULL);
}
}
break;
case ID_FILE_INP:
file.is_open();
file >> x;
file >> y;
file >> r;
x *= 50;
y *= 50;
r *= 50;
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL); //Начало координат
SetMapMode(hdc, MM_ISOTROPIC); //логические единицы отображаем, как ф
HBRUSH hBrush;
hBrush = CreateHatchBrush(HS_FDIAGONAL, RGB(255, 0, 0));
SelectObject(hdc, hBrush);
Ellipse(hdc, x + r, y + r, x - r, y - r);
Line(hdc, 0, 1000, 0, -1000); //ось Y
Line(hdc, -1000, 0, 1500, 0); //ось X
MoveToEx(hdc, 0, 0, NULL);
for (int i = -1000; i < 1000; i += 50)
{
Line(hdc, i, 3, i, -3);
Line(hdc, -3, i, 3, i);
}
TextOut(hdc, -15, -5, L"0", 1);
TextOut(hdc, 50, -5, L"1", 1);
TextOut(hdc, -30, 50, L"1", 1);
TextOut(hdc, 900, -5, L"x", 1);
TextOut(hdc, 10, 500, L"y", 1);

while (not file.eof()) {
file >> x_p >> y_p;
x_p *= 50;
x_points.push_back(x_p);
y_p *= 50;
y_points.push_back(y_p);
hBrushSol = CreateSolidBrush(RGB(255, 0, 0));
hOldBrush = (HBRUSH)SelectObject(hdc, hBrushSol);
Ellipse(hdc, x_p + 5, y_p + 5, x_p - 5, y_p - 5);
}

```

```

file.close();
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...
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;
}

```