
Общее задание

Разработать класс для представления неориентированного графа и поиска цикла или пути между двумя вершинами при заданных условиях.

Должен использоваться не рекурсивный алгоритм.

Нельзя использовать никакие библиотеки, кроме встроенных в язык C++.

Разработать классическое приложение Windows в среде Visual Studio на языке C++, 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.

В программе должны использоваться меню и диалоговые окна.

Индивидуальное задание

Найти самый длинный цикл в графе, не проходящий через заданную вершину.

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» ввести дополнительные данные и, собственно, решить её.

1) Если находятся циклы (не содержащие указанную вершину), то сравниваются их длины и графически рисуется самый длинный.

2) Если цикл один и в нем нет заданной вершины, то он изображается.

3) Иначе выводится окно с сообщением «циклов в графе, не проходящих через заданную вершину - нет»

Алгоритм выполнения операций на псевдокоде

Поиск циклов ($v_n = v_k$)

ЦИКЛ ($j = 0, n-1$)

$M[j] = 0$

КОНЕЦ_ЦИКЛ

$Ks = 0$

$St[ks] = v_n$

```

M[vn] = 1
L = 0
ЦИКЛ_ПОКА (ks >= 0)
    V = St[ks]
    Pr = 0
    ЦИКЛ (j = L, n)
        ЕСЛИ (A[v, j] = 1) ТО
            ЕСЛИ (j = vk) ТО
                ЕСЛИ (t принадлежит St[])
                    Вывод (St[i], I = 0, ks), vk
                КОНЕЦ_ЕСЛИ
            ИНАЧЕ
                ЕСЛИ (M[j] = 0) ТО
                    Pr = 1
                    Прервать цикл по j
                КОНЕЦ_ЕСЛИ
            КОНЕЦ_ЕСЛИ
        КОНЕЦ_ЕСЛИ
    КОНЕЦ_ЕСЛИ
    ЕСЛИ (Pr = 1) ТО
        Ks = ks + 1
        St[ks] = j
        L = 0
        M[j] = 1
    ИНАЧЕ
        L = v + 1
        M[v] = 0
        Ks = ks - 1
    КОНЕЦ_ЕСЛИ
КОНЕЦ_ЕСЛИ

```

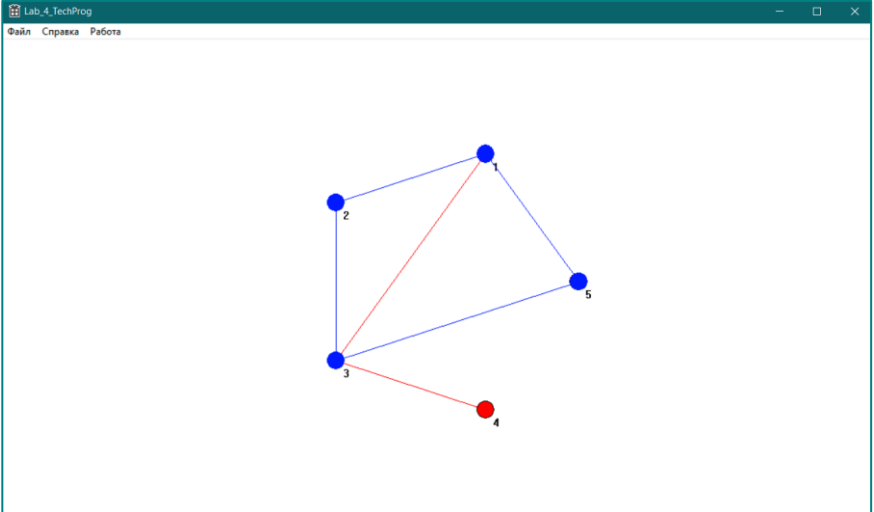
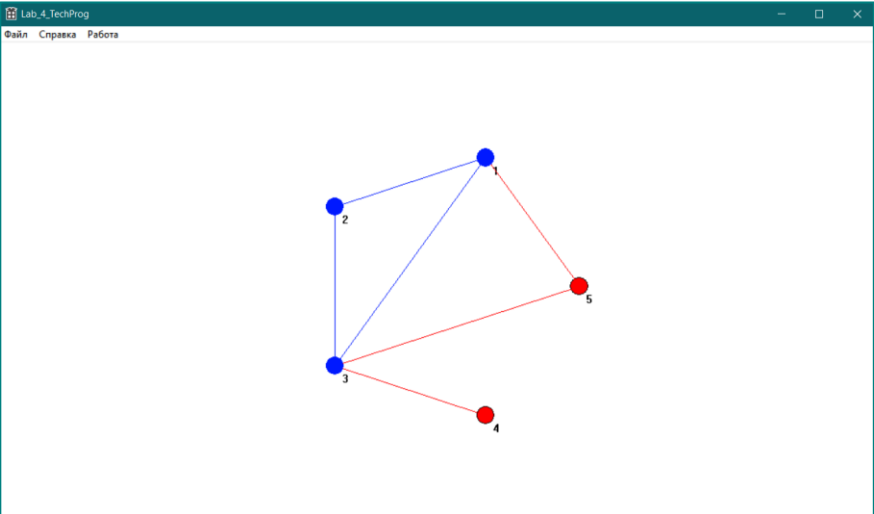
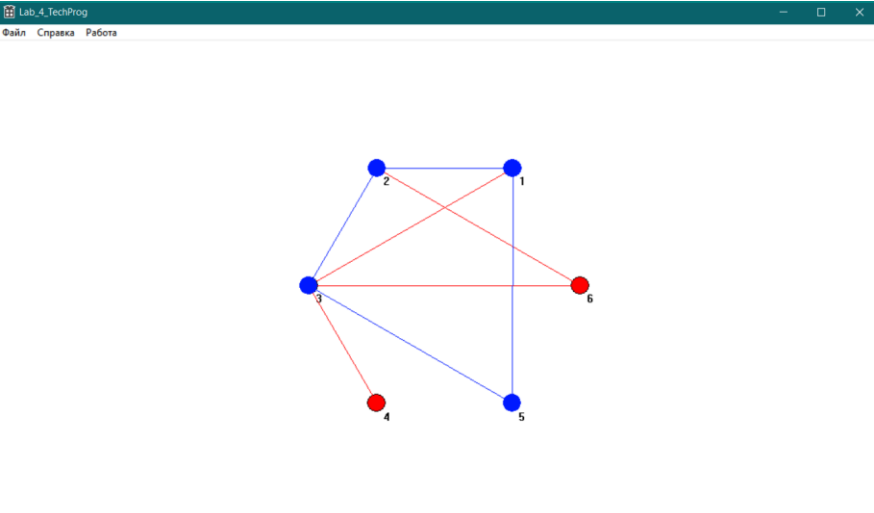
Определение самого длинного цикла:

```

max = RezLen[0]; // Длина самого длинного цикла
i = 0;
index = 0; // Индекс самого длинного цикла
ЦИКЛ_ПОКА (RezLen[i] != 0 и i < 100)
    ЕСЛИ (max <= RezLen[i]) ТО
        Max = RezLen[i];
        Index = i;
    I++;
КОНЕЦ_ЦИКЛ

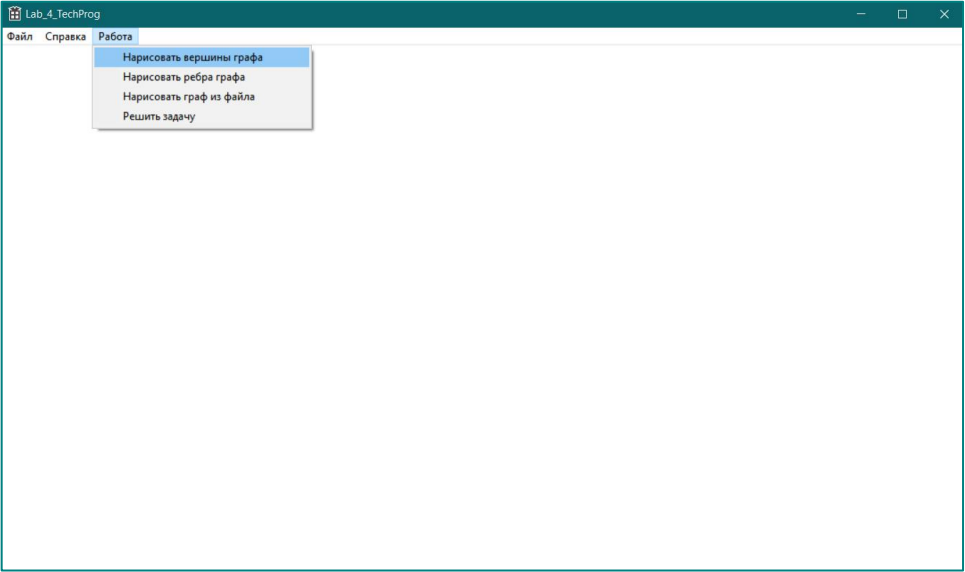
```

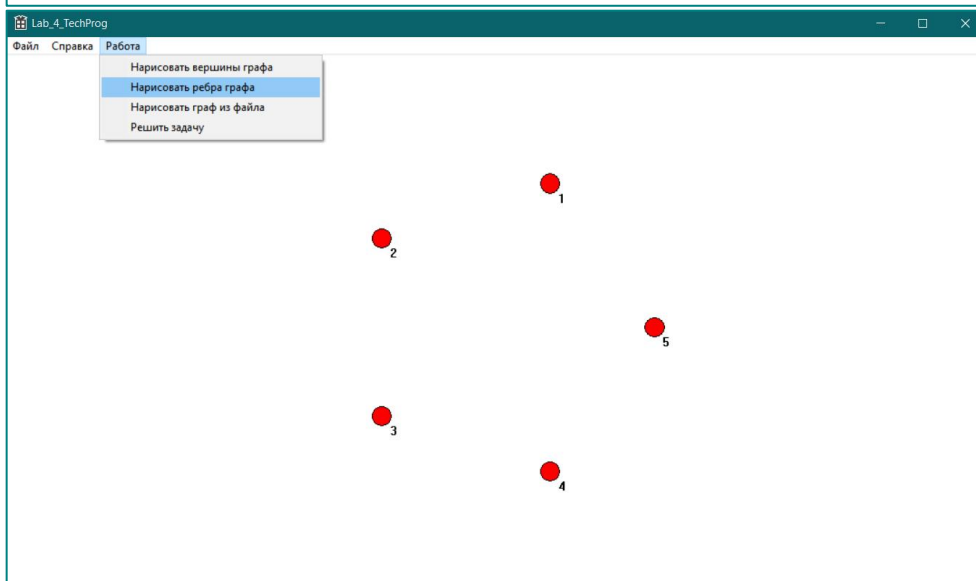
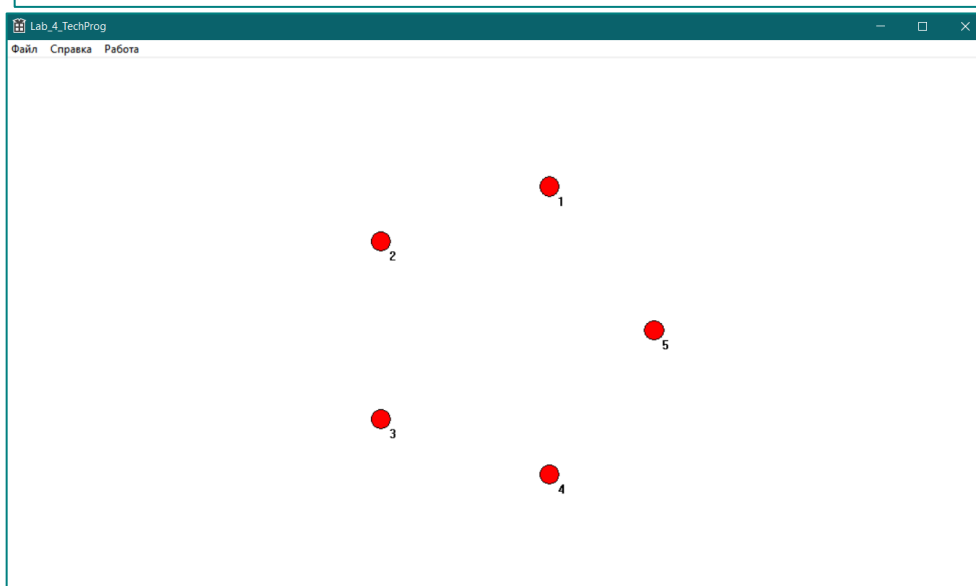
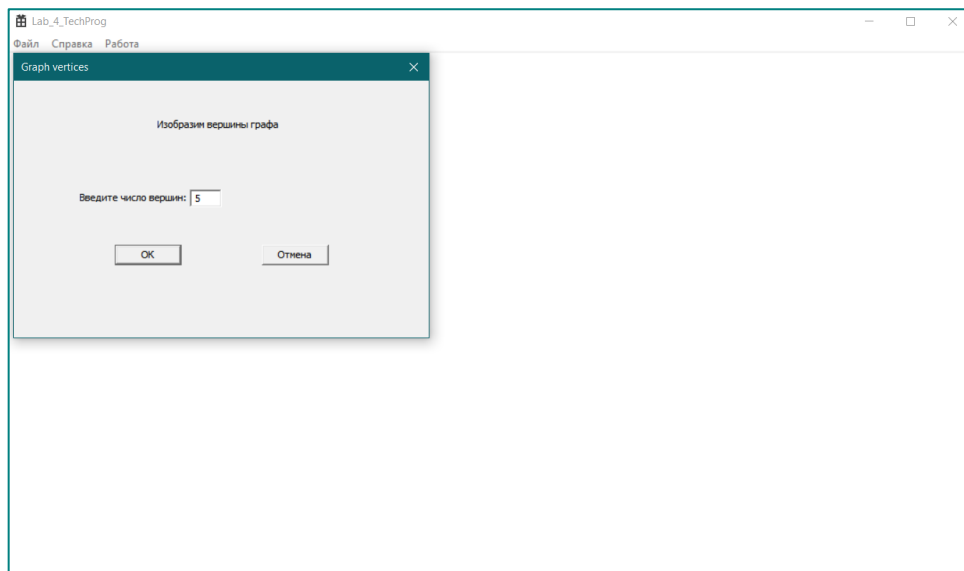
Тесты

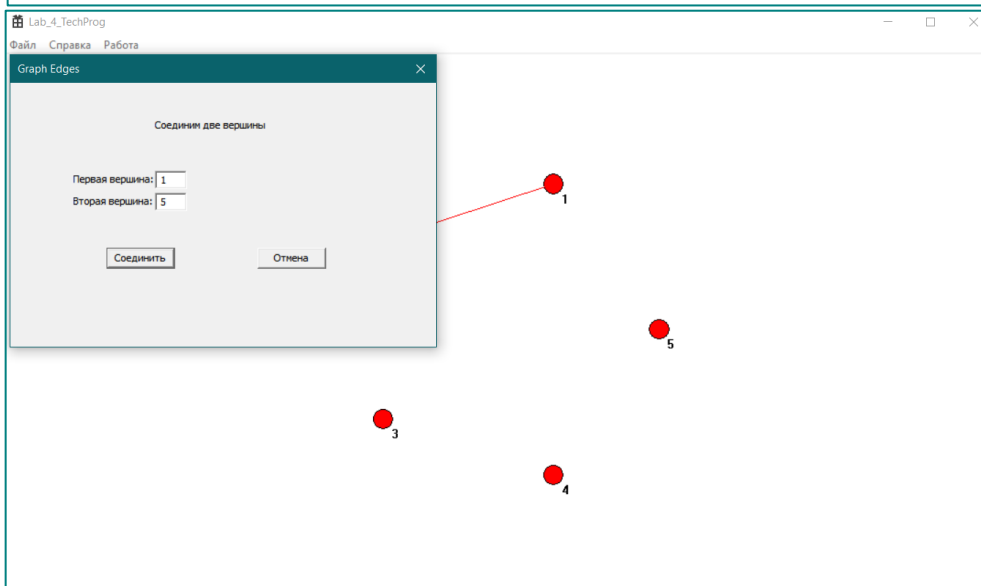
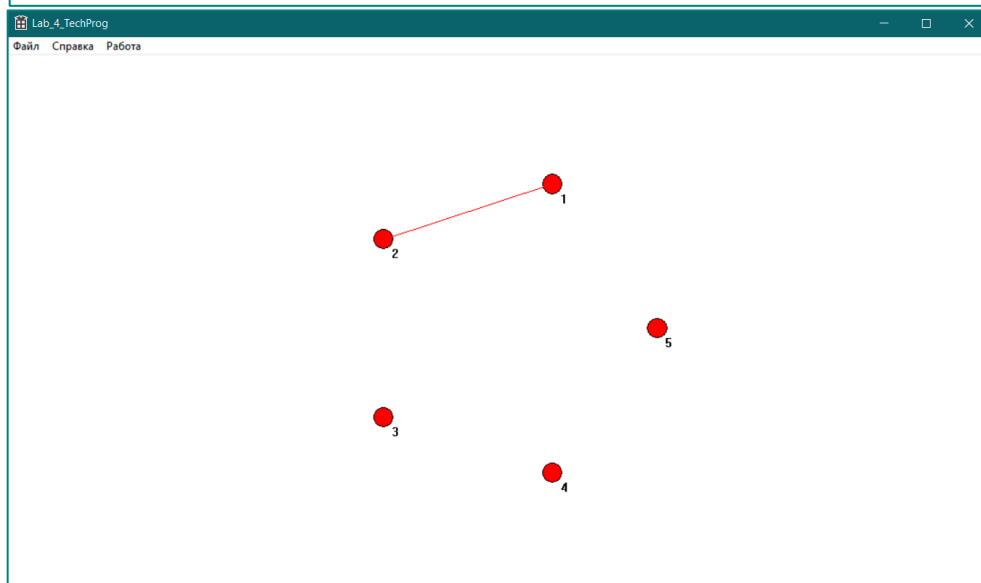
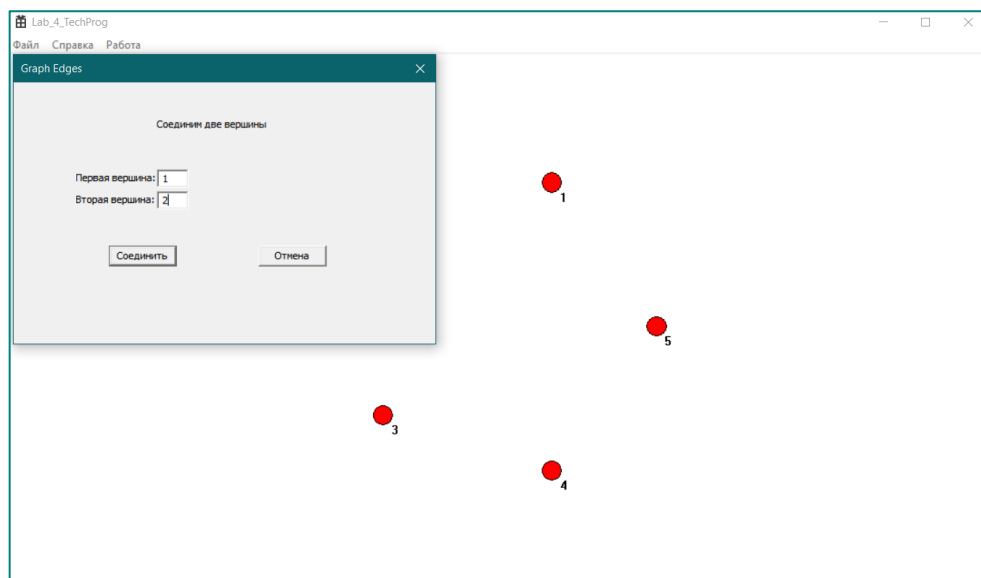
Тест №	Входные данные	Результат
1	5 //кол-во вершин 4 //заданная вершина 1 2 //ребра 2 3 3 4 1 5 3 5 1 3	
2	5 //кол-во вершин 5 //заданная вершина 1 2 //ребра 2 3 3 4 1 5 3 5 1 3	
3	6 6 1 2 2 3 3 4 1 5 3 5 1 3 2 6 3 6	

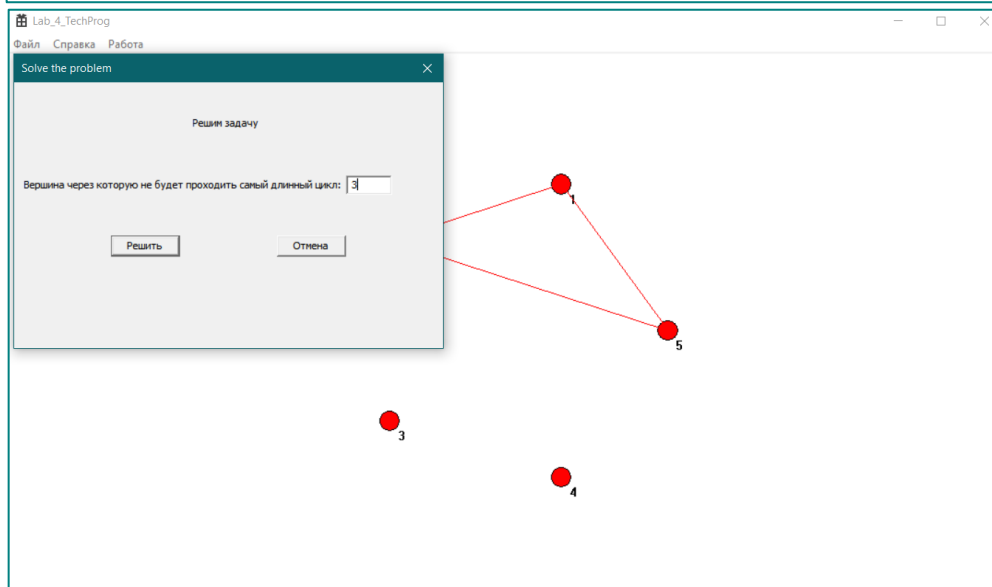
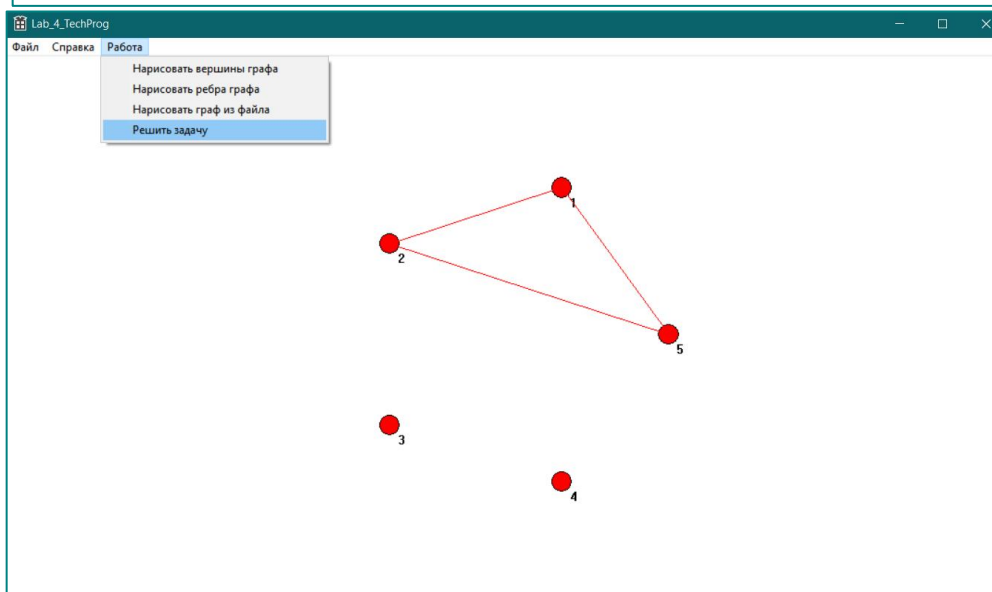
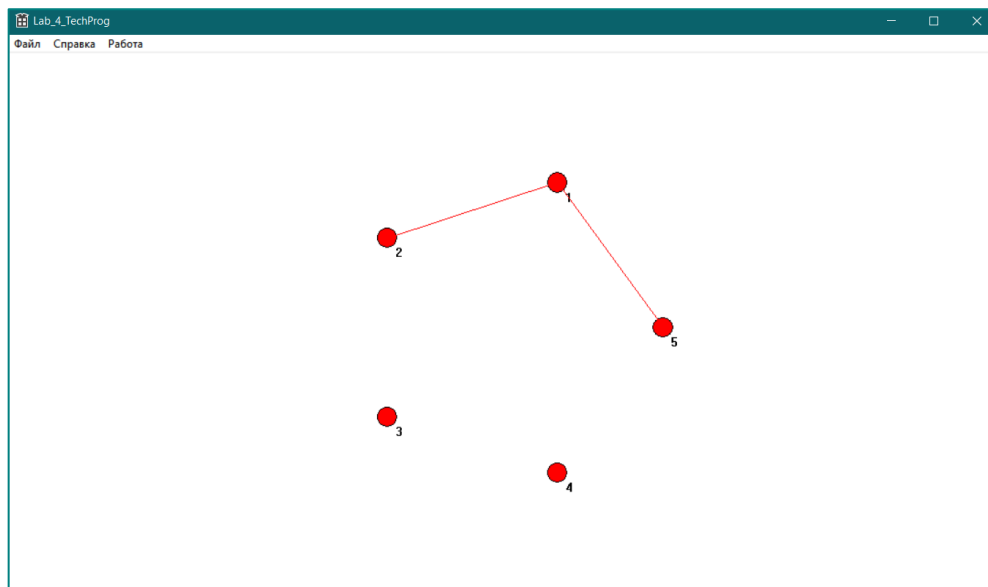
4	6 3 1 2 2 3 3 4 1 5 3 5 1 3 2 6 3 6	
5	6 2 1 2 2 3 3 4 1 5 3 5 1 3 2 6 3 6 4 5 4 6	

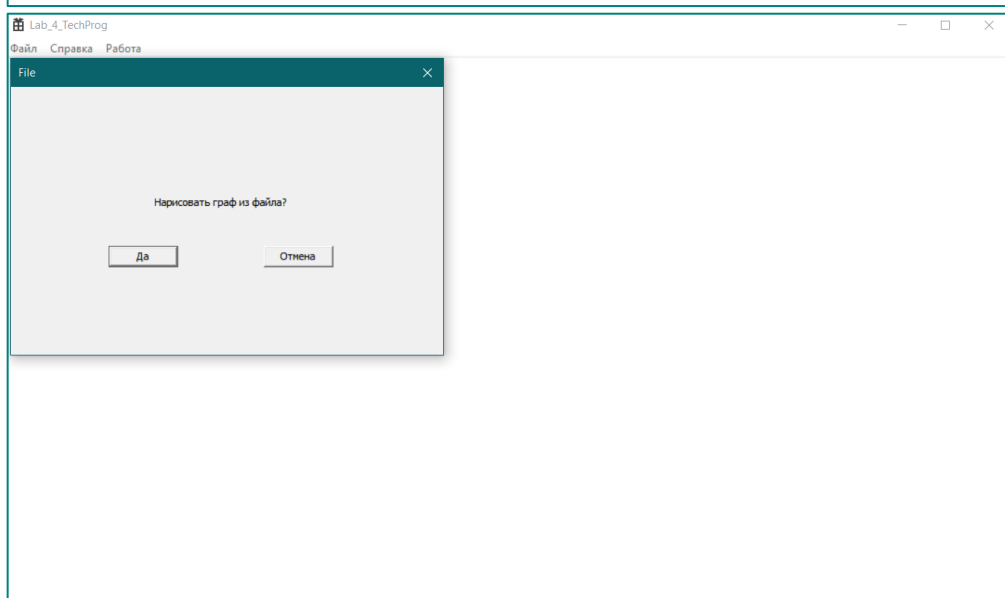
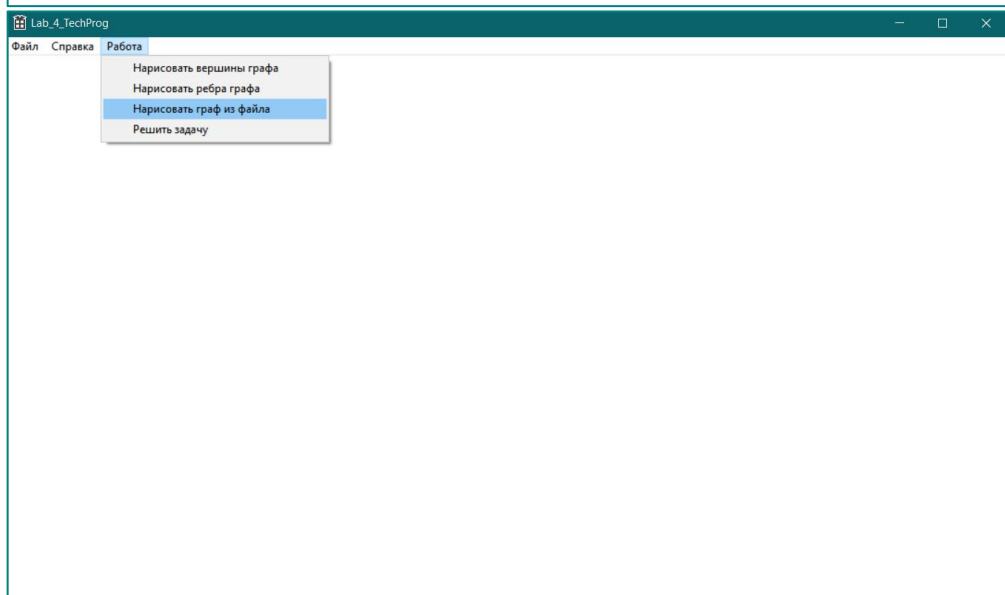
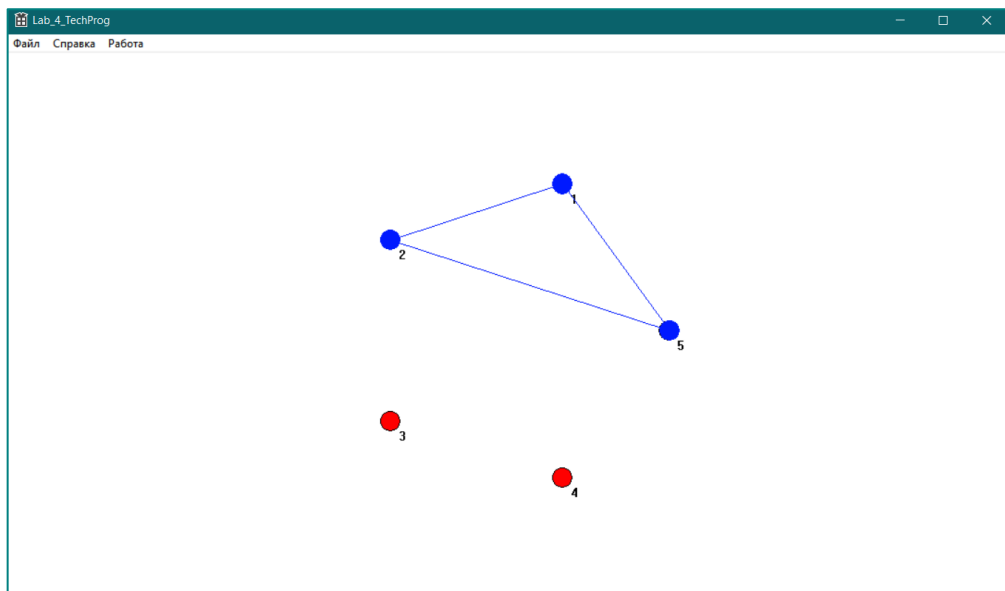
Распечатки экранов при работе программы

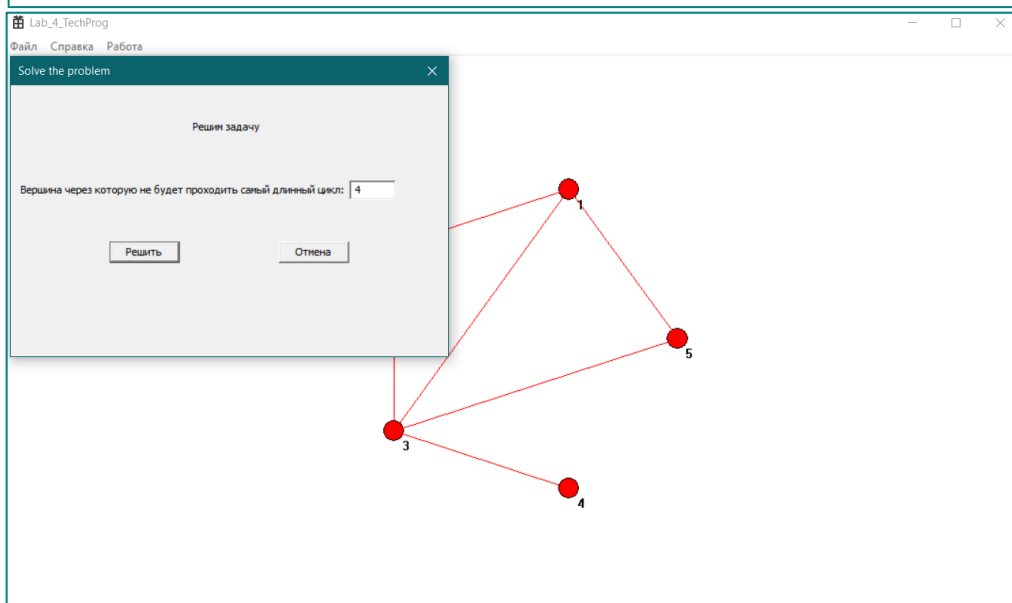
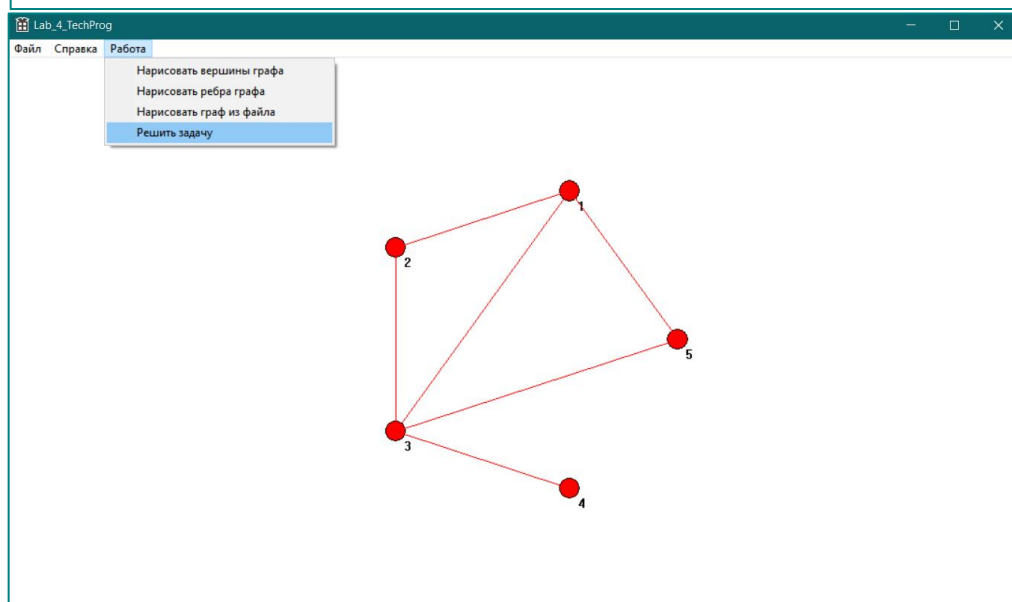
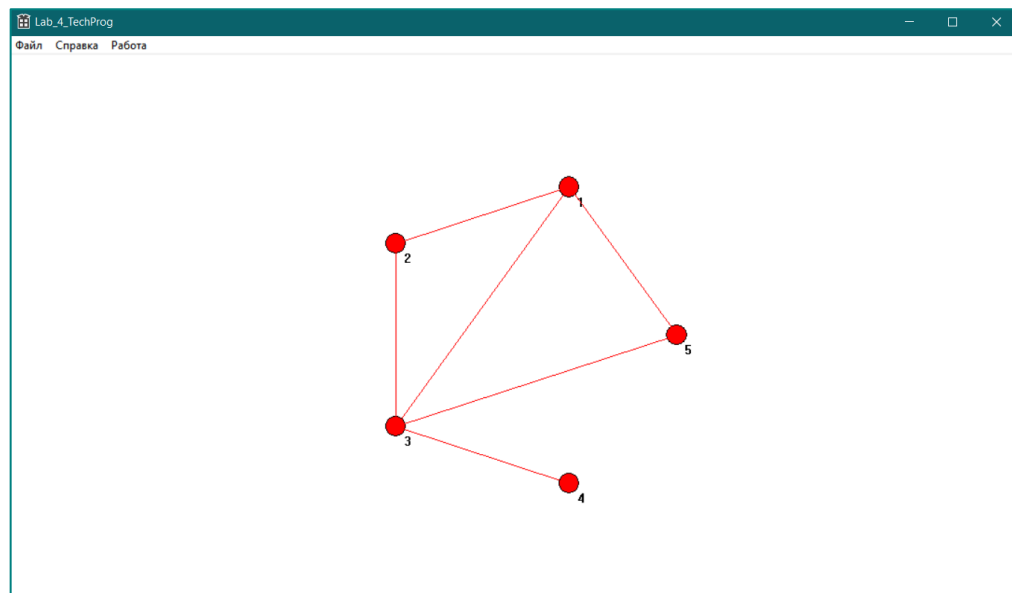


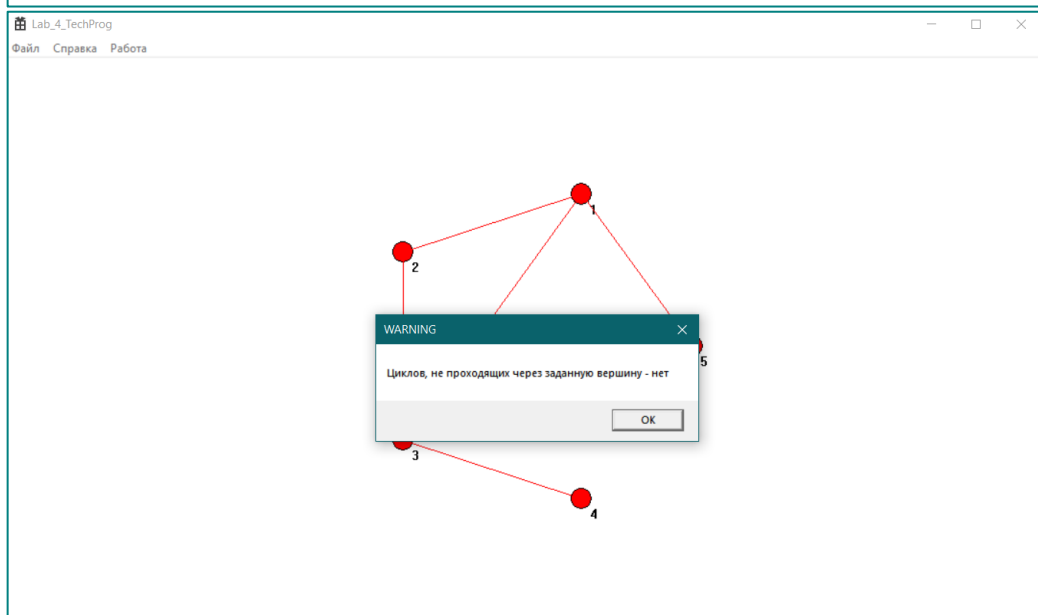
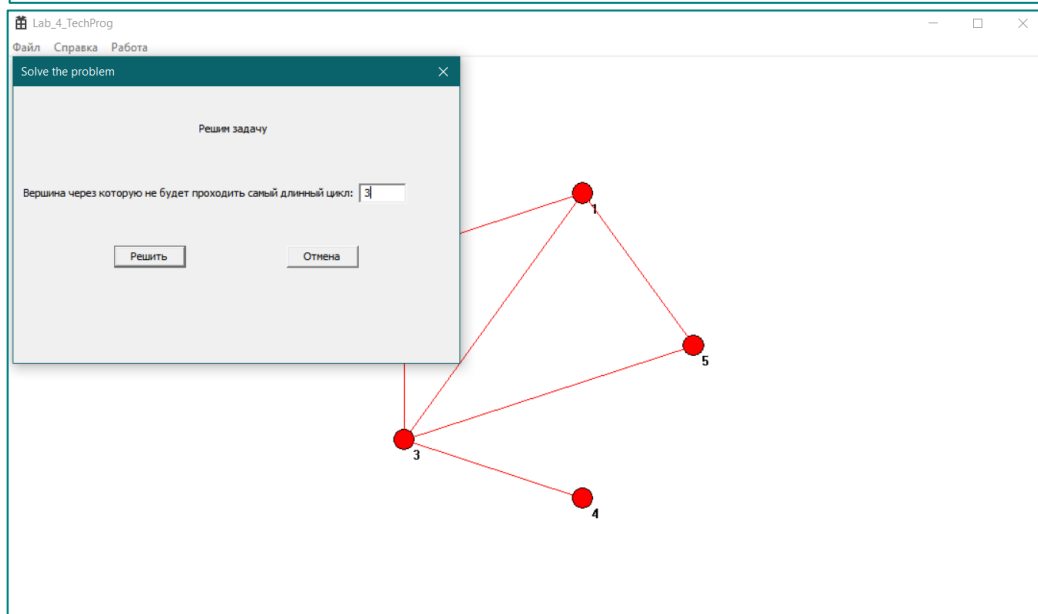
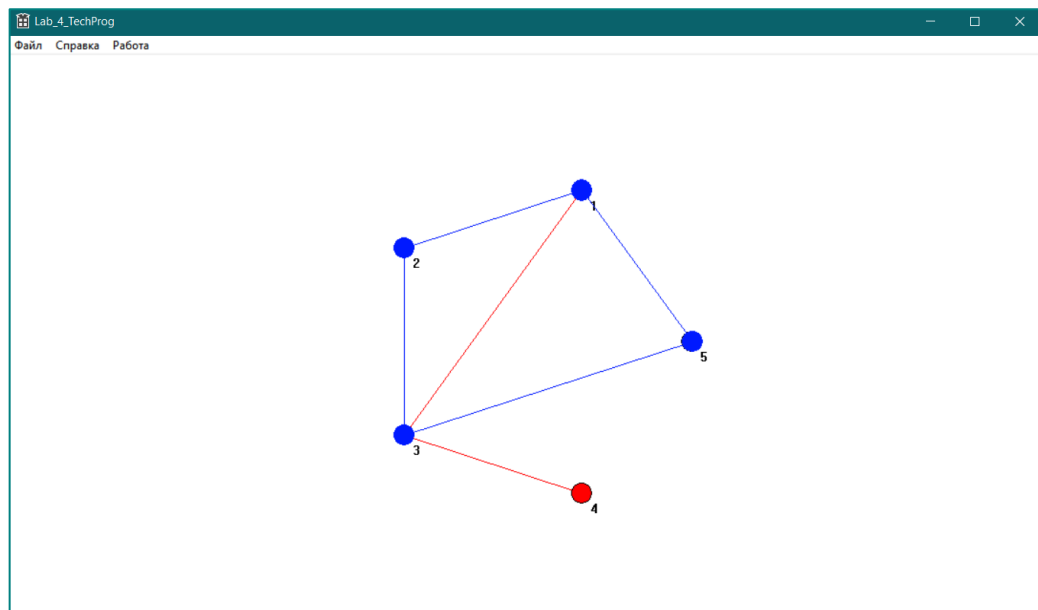












Листинг программы

```
#include "framework.h"
#include "Lab_4_TechProg.h"
#include "Graph_class.h"

#define MAX_LOADSTRING 100
#define PI 3.141592

// Глобальные переменные:
HINSTANCE hInst; // текущий экземпляр
WCHAR szTitle[MAX_LOADSTRING]; // Текст строки заголовка
WCHAR szWindowClass[MAX_LOADSTRING]; // имя класса главного окна
HBRUSH hBrushSol, hOldBrush, hBrushBlue;
//Global var:

TCHAR bufX[5];
TCHAR bufY[5];
int PrInput = 0;
int num_vertices; //Число вершин
HDC hdc;
Graph G; // Мой граф

ifstream file("TextFile1.txt");
double x_i; //Индексы i-вершины графа
double y_i;

int x_edges; //Индексы i-ребра графа
int y_edges;

const int Rad = 300; //Радиус окружности
расположения вершин
double phi = 0; //Углы между вершинами графа
TCHAR buf[3]; //Буффер для вывода номера
вершины

float *vertices_x_i = new float; //адреса вершин графа
float *vertices_y_i = new float;

int Cicle[100]; //Цикл содержащий вершины графа
int Non_vertices;

// 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_LAB4TECHPROG, szWindowClass, MAX_LOADSTRING);
MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:
if (!InitInstance (hInstance, nCmdShow))
{
    return FALSE;
}

HACCEL hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_LAB4TECHPROG));

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_LAB4TECHPROG));
    wcex.hCursor        = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH) (COLOR_WINDOW+1);
    wcex.lpszMenuName    = MAKEINTRESOURCEW(IDC_LAB4TECHPROG);
    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;
}
//My specific function:
INT_PTR CALLBACK DlgInput_vertices(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_EDIT1, bufX, 4);
                    EndDialog(hDlg, 1);
                    break;
                case IDCANCEL:
                    EndDialog(hDlg, 0);
                    break;
            }
            return (INT_PTR) TRUE;
        }
        return (INT_PTR) FALSE;
    }
}
INT_PTR CALLBACK DlgInput_file(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;
}
INT_PTR CALLBACK DlgInput_edges(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_EDIT1, 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;
}

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);
}

//
// 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_ABOUT:
DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
break;
case ID_Solve_The_Problem:
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_Solve_The_Problem), hWnd,
DlgInput_vertices);
if (PrInput)

```

```

{
Non_vertices = _wtoi(bufX);
if (G.Solve_The_Task(Cicle, Non_vertices - 1))
{
int i = 0;
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL);
SetMapMode(hdc, MM_ISOTROPIC);
hBrushSol = CreateSolidBrush(RGB(0, 25, 255));
hOldBrush = (HBRUSH)SelectObject(hdc, hBrushSol);
while ((Cicle[i+1] != 0) and (i+1 <= 100))
{
Ellipse(hdc, vertices_x_i[Cicle[i] - 1] + 20, vertices_y_i[Cicle[i]-1] + 20,
vertices_x_i[Cicle[i]-1] - 20, vertices_y_i[Cicle[i]-1] - 20);
Ellipse(hdc, vertices_x_i[Cicle[i + 1] - 1] + 20, vertices_y_i[Cicle[i + 1] -
1] + 20, vertices_x_i[Cicle[i + 1] - 1] - 20, vertices_y_i[Cicle[i + 1] - 1]
- 20);
HPEN hPen1;
hPen1 = CreatePen(PS_SOLID, 2, RGB(0, 25, 255));
SelectObject(hdc, hPen1);
Line(hdc, vertices_x_i[Cicle[i]-1], vertices_y_i[Cicle[i]-1],
vertices_x_i[Cicle[i + 1]-1], vertices_y_i[Cicle[i + 1] - 1]);
i++;
}
}
else
{
MessageBox(NULL, L"Циклов, не проходящих через заданную вершину - нет",
L"WARNING", NULL);
}
}
break;
case ID_Draw_Graph_vertices:
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_Graph_vertices), hWnd,
DlgInput_vertices);
if (PrInput)
{
num_vertices = _wtoi(bufX);
vertices_x_i = new float[num_vertices];
vertices_y_i = new float[num_vertices];
Graph G_change(num_vertices, num_vertices);
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL);
SetMapMode(hdc, MM_ISOTROPIC);
hBrushSol = CreateSolidBrush(RGB(255, 0, 0));
hOldBrush = (HBRUSH)SelectObject(hdc, hBrushSol);
for (int i = 1; i <= num_vertices; i++)
{
phi = PI * i * 2 / num_vertices;
x_i = Rad * (1 / (sqrt(1 + pow(tan(phi), 2))));
y_i = sqrt(pow(Rad, 2) - pow(x_i, 2));
if ((phi > PI/2) and (phi <= PI))
{
x_i *= -1;
}
if ((phi > PI) and (phi <= 1.5*PI))
{
x_i *= -1;
y_i *= -1;
}
if ((phi > 1.5*PI) and (phi <= 2*PI))
{
y_i *= -1;
}
}
}
}

```

```

Ellipse(hdc, x_i + 20, y_i + 20, x_i - 20, y_i - 20);
vertices_x_i[i - 1] = x_i;
vertices_y_i[i - 1] = y_i;
swprintf_s(buf, TEXT("%d"), i);
TextOut(hdc, x_i+15, y_i-15, buf, 2);
}
G.copy(G_change);
}
break;
case ID_Draw_Graph_Edges:
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_Graph_Edges), hWnd,
DlgInput_edges);
if (PrInput)
{
x_edges = _wtoi(bufX);
y_edges = _wtoi(bufY);
G.change(x_edges - 1, y_edges - 1);
G.change(y_edges - 1, x_edges - 1);
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL);
SetMapMode(hdc, MM_ISOTROPIC);
HPEN hPen1;
hPen1 = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
SelectObject(hdc, hPen1);
Line(hdc, vertices_x_i[x_edges - 1], vertices_y_i[x_edges - 1],
vertices_x_i[y_edges - 1], vertices_y_i[y_edges - 1]);
}
break;
case ID_draw_graph_from_file:
PrInput = DialogBox(hInst, MAKEINTRESOURCE(IDD_File), hWnd, DlgInput_file);
if (PrInput)
{
file.is_open();
file >> num_vertices;
vertices_x_i = new float[num_vertices];
vertices_y_i = new float[num_vertices];
Graph G_change(num_vertices, num_vertices);
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL);
SetMapMode(hdc, MM_ISOTROPIC);
hBrushSol = CreateSolidBrush(RGB(255, 0, 0));
hOldBrush = (HBRUSH)SelectObject(hdc, hBrushSol);
for (int i = 1; i <= num_vertices; i++)
{
phi = PI * i * 2 / num_vertices;
x_i = Rad * (1 / (sqrt(1 + pow(tan(phi), 2))));
y_i = sqrt(pow(Rad, 2) - pow(x_i, 2));
if ((phi > PI / 2) and (phi <= PI))
{
x_i *= -1;
}
if ((phi > PI) and (phi <= 1.5 * PI))
{
x_i *= -1;
y_i *= -1;
}
if ((phi > 1.5 * PI) and (phi <= 2 * PI))
{
y_i *= -1;
}
Ellipse(hdc, x_i + 20, y_i + 20, x_i - 20, y_i - 20);
vertices_x_i[i - 1] = x_i;
vertices_y_i[i - 1] = y_i;
swprintf_s(buf, TEXT("%d"), i);

```



```

TextOut(hdc, x_i + 15, y_i - 15, buf, 2);
}
while (not file.eof()) {
file >> x_edges >> y_edges;
G_change.change(x_edges - 1, y_edges - 1);
G_change.change(y_edges - 1, x_edges - 1);
hdc = GetDC(hWnd);
SetViewportOrgEx(hdc, 550, 300, NULL);
SetMapMode(hdc, MM_ISOTROPIC);
HPEN hPen1;
hPen1 = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
SelectObject(hdc, hPen1);
Line(hdc, vertices_x_i[x_edges - 1], vertices_y_i[x_edges - 1],
vertices_x_i[y_edges - 1], vertices_y_i[y_edges - 1]);
}
file.close();
G.copy(G_change);
}
break;
case IDM_EXIT:
DestroyWindow(hWnd);
delete[] vertices_x_i;
delete[] vertices_y_i;
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;
}
}

```