Общее задание

Разработать абстрактный класс для представления неориентированного графа и поиска всех циклов или путей между двумя заданными вершинами.
Класс должен включать объект класса "Стек" для представления стека.
Класс должен включать как минимум следующие методы:

- конструктор;
- метод для построения всех путей или циклов, удовлетворяющих некоторому условию;
- абстрактный метод для поиска еще не обработанной смежной вершины;
- абстрактный метод для проверки соответствия построенного пути или цикла заданным условиям.

Используя описанный класс как базовый разработать производный класс для решения задачи о построении всех путей или циклов, удовлетворяющих
конкретным условиям задания.
Производный класс должен содержать описание графа в формате, указанном в задании.
В производном классе должны быть переопределены виртуальные методы базового класса с учётом конкретного задания.
Необходимо разработать класс "Стек" для представления стека.
Должен использоваться не рекурсивный алгоритм поиска путей или циклов.
Нельзя использовать никакие библиотеки, кроме встроенных в язык C#.
Не должны использоваться коллекции.
Разработать программу, работающую в среде Visual Studio на основе Windows Forms и реализующую конкретное задание.
Программа должна обеспечивать ввод описания графа из файла и с клавиатуры.
Программа должна обеспечивать редактирование и сохранение описания графа.
Программа должна обеспечивать представление исходного графа и результатов в графическом виде.
Программа должна сначала найти и запомнить все пути или циклы, а затем показывать их по запросу.
В описаниях заданий используются следующие обозначения:

- стек_масс – стек на основе массива;
- стек_спис – стек на основе связанного списка;
- граф_матр – для описания графа используется матрица смежности;
- граф_масс – для описания графа используются списки в массивах;
- граф_спис – для описания графа используются связанные списки.

Индивидуальное задание

Найти все пути из одной вершины в другую, проходящие через заданное ребро в определенном направлении (стек_масс, граф_масс).

Описание работы программы

Сначала необходимо ввести граф: из файла, из клавиатуры. Далее его изобразить, чтобы было удобнее работать для формулировки условия решения задачи. После чего вводятся необходимые вершины и ребро, по которым будет пролегать наш путь.
Запуском кнопки “Выполнить” мы запускаем программу и она выводит множество путей, которые удовлетворяют заданным условиям. Опять же, для наглядности, можно их показать на графе.

Алгоритмы выполнения основным операций на псевдокоде

Алгоритм поиска путей через две вершины и заданное ребро.

```
ЦИКЛ ( j=0,n-1 )
        M[j]=0
КОНЕЦ_ЦИКЛ
ks=0
G.Push(vn);
M[vn]=1
L=0
ЦИКЛ_ПОКА ( ks>=0 )
        v = G.Get_this_position(ks);
        Pr=0
        ЦИКЛ ( j=L,n )
                ЕСЛИ (Search_vertex(v, j) ) ТО
                        ЕСЛИ ( j=vk ) ТО
                                ЕСЛИ ( a,b принадлежит стеку ) ТО
```

```
                    Вывод “Путь содержит вершины”
                    Вывод (G.Get_this_position(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_ЦИКЛ
    ЕСЛИ ( Pr=1 ) ТО
        ks=ks+1;
        S_length = ks;
        G.Push(j);
        L=0;
        M[j]=1;
    ИНАЧЕ
        L=v+1
        M[v]=0
        ks=ks-1
        G.Pop();
    КОНЕЦ_ЕСЛИ
КОНЕЦ_ЦИКЛ
```

Тесты

- X1 – путь должен проходить через вершину А = 1.
- X2 – путь должен проходить через вершину Б = 3.
- X3 – путь должен проходить через ребро Д-Е = 4-5.

Таким образом, тесты должны обеспечивать следующие ситуации:

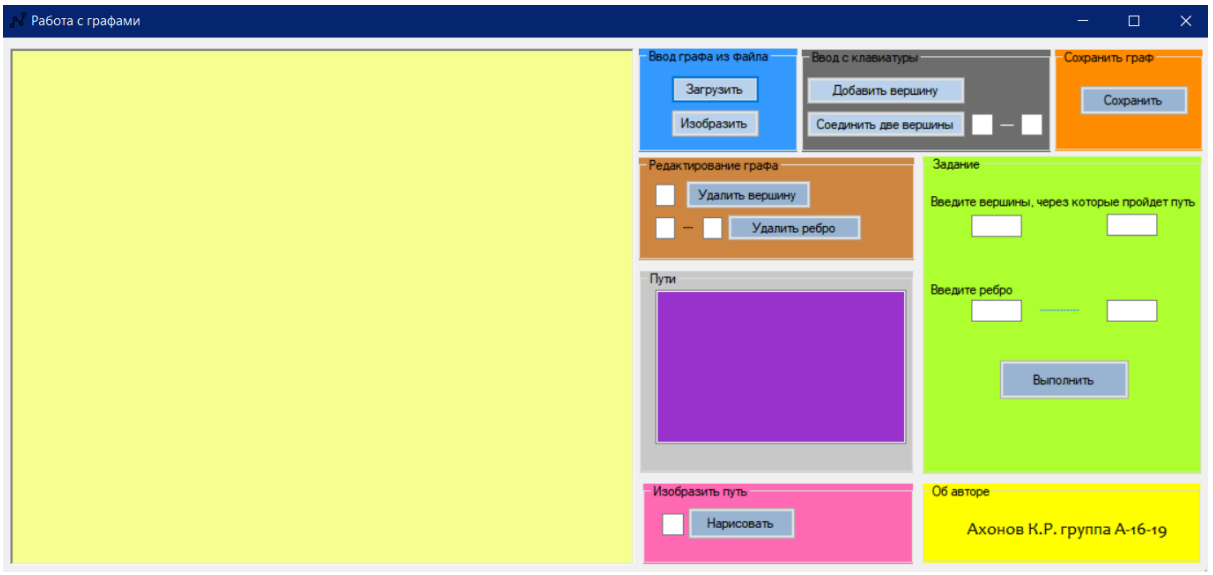
- ❖ Выполняются все три условия, и программа должна находить пути, удовлетворяющие этим условиям.
- ❖ Существует путь, для которого не выполняется условие X1, но выполняются условия X2 и X3, и программа отвергает этот путь.
- ❖ Существует путь, для которого не выполняется условие X2, но выполняются условия X1 и X3, и программа отвергает этот путь.

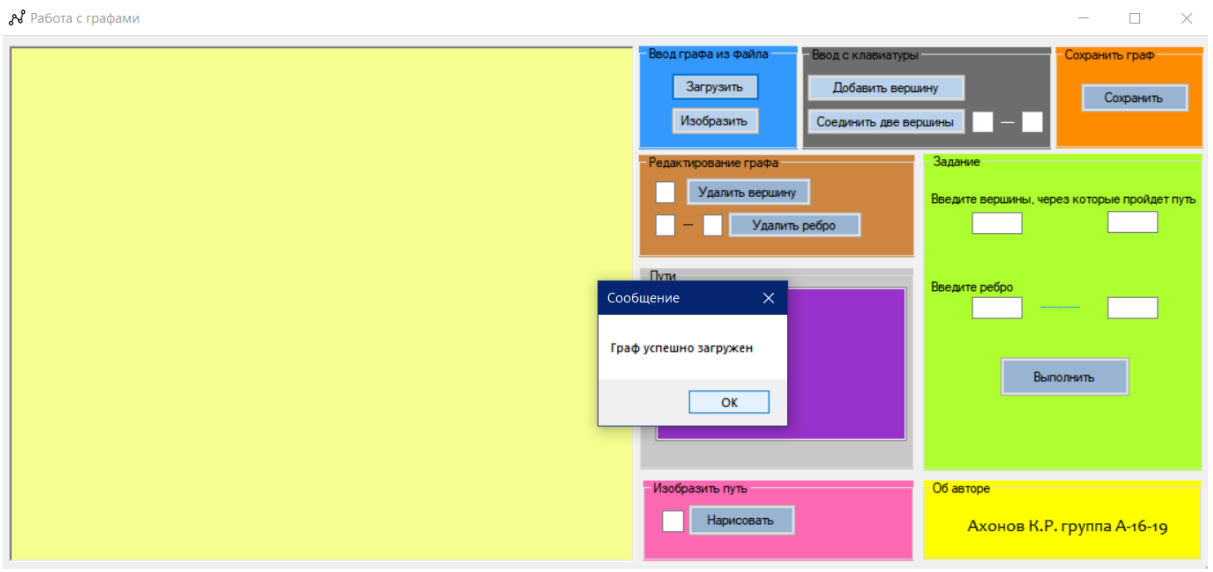
❖ Существует путь, для которого не выполняется условие X3, но выполняются условия X1 и X2, и программа отвергает этот путь.

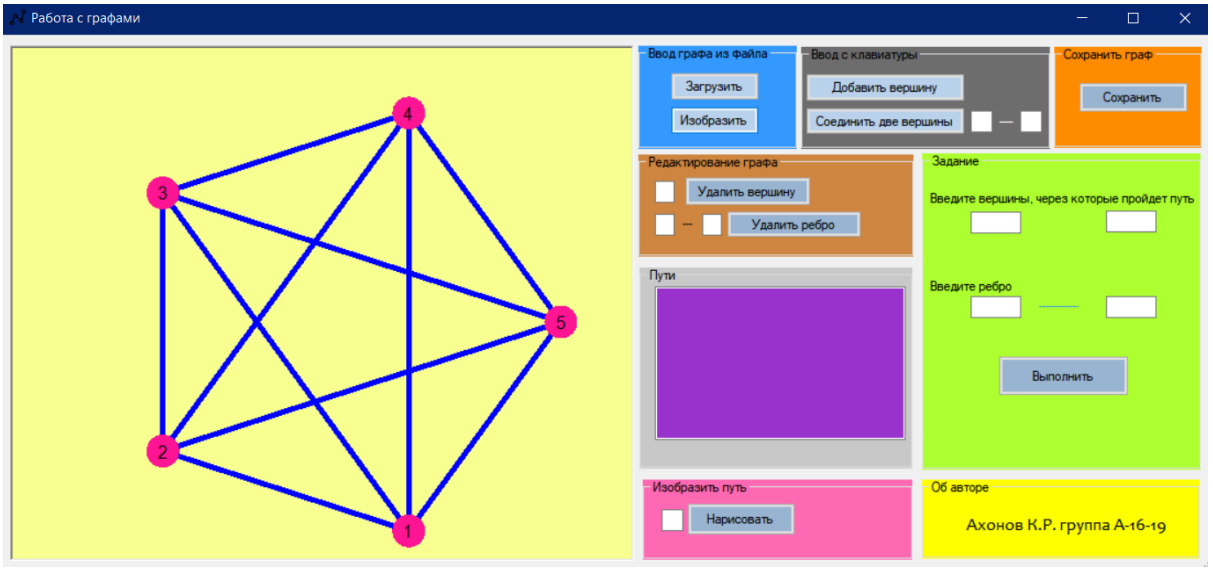
№ Теста	*Входные данные*	*Выходные данные*
1	*Граф:* *12* *23* *34* *41* *13* *15* *35* *45* *24* *25* *А = 1;* *Б = 3;* *Д-Е = 4-5;*	*1) 1->2->4->5->3* *2) 1->4->5->2->3* *3) 1->4->5->3*
2	*Граф:* *12* *23* *34* *41* *13* *15* *35* *45* *24* *25* *не выполняется это условие --- А = 1;* *Б = 3;* *Д-Е = 4-5;*	*Существует путь* *3 -> 4 -> 5 -> 2* *Но программа его отвергает:* *1) 1->2->4->5->3* *2) 1->4->5->2->3* *3) 1->4->5->3*
3	*Граф:* *12* *23* *34* *41* *13* *15*	*Существует путь* *1 -> 4 -> 5 -> 2* *Но программа его отвергает:* *1) 1->2->4->5->3* *2) 1->4->5->2->3* *3) 1->4->5->3*

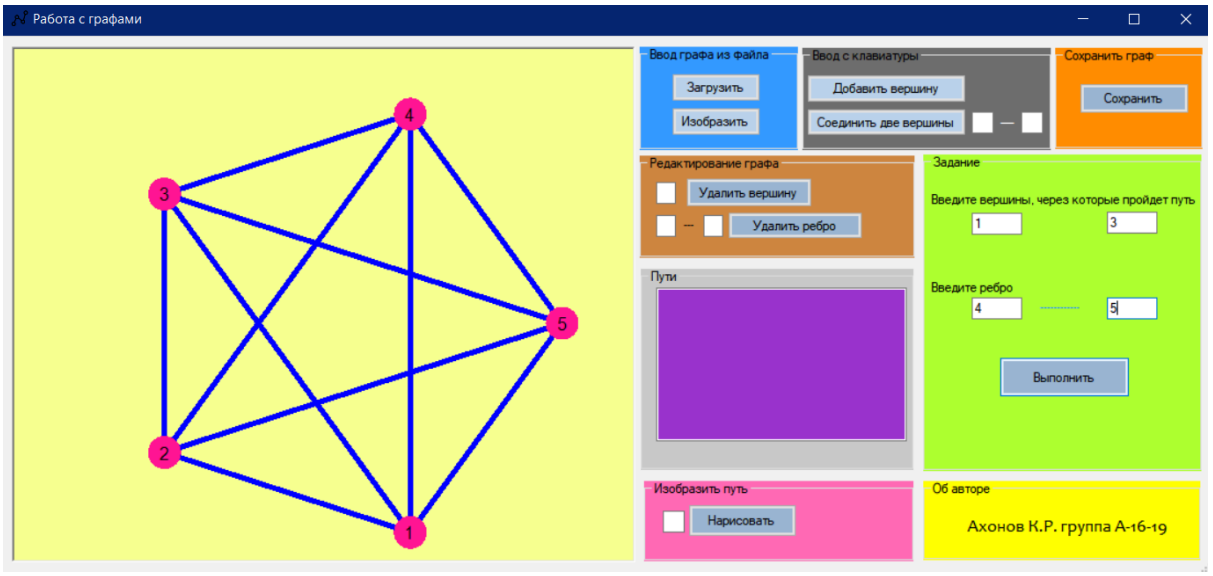
	35 *45* *24* *25* *А = 1;* *не выполняется это условие --- Б = 3;* *Д-Е = 4-5;*	
4	*Граф:* *12* *23* *34* *41* *13* *15* *35* *45* *24* *25* *А = 1;* *Б = 3;* *не выполняется это условие --- Д-Е = 4-5;*	*Существует путь* *1 -> 2 -> 3* *Но программа его отвергает:* *1) 1->2->4->5->3* *2) 1->4->5->2->3* *3) 1->4->5->3*

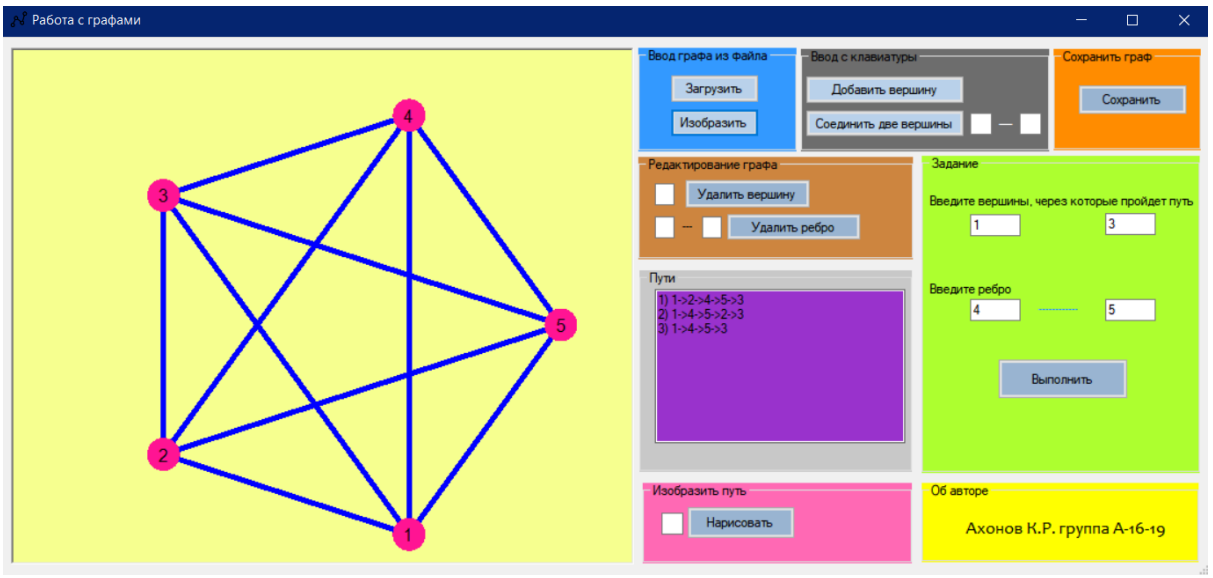
Распечатки экранов при работе программы

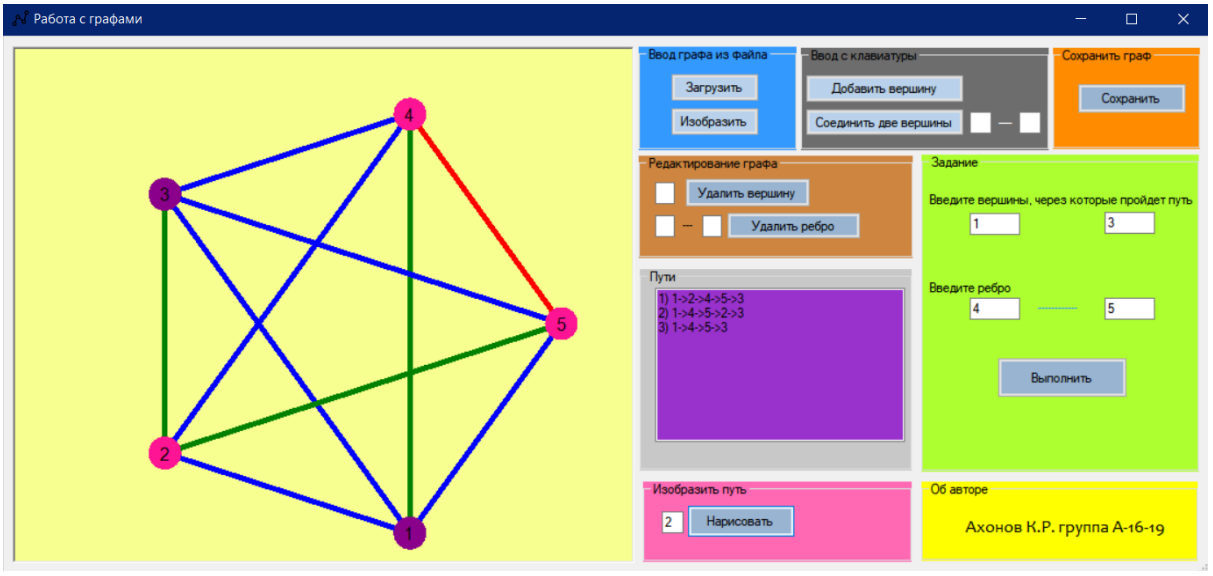


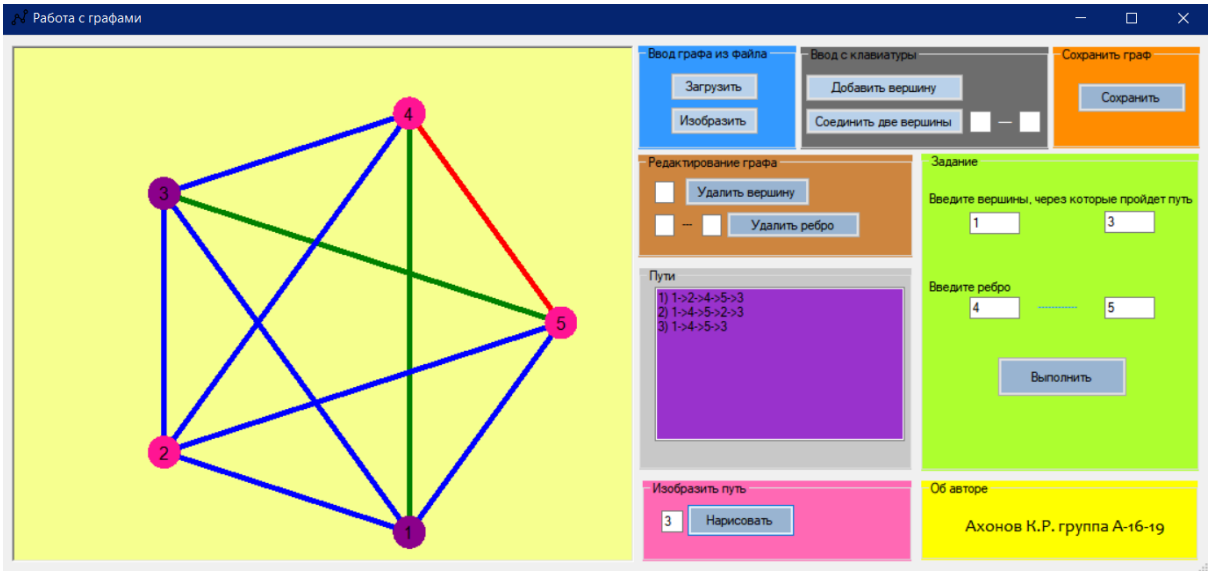

Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
Пути
Введите ребро
Выполнить
Сообщение
Граф успешно загружен
OK
Изобразить путь
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19


Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
Пути
Введите ребро
Выполнить
Изобразить путь
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19
1
2
3
4
5


Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
1
3
Пути
Введите ребро
4
5
Выполнить
Изобразить путь
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19
1
2
3
4
5


Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
1
3
Пути
1) 1->2->4->5->3
2) 1->4->5->2->3
3) 1->4->5->3
Введите ребро
4
5
Выполнить
Изобразить путь
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19
1
2
3
4
5


Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
1
3
Пути
1) 1->2->4->5->3
2) 1->4->5->2->3
3) 1->4->5->3
Введите ребро
4
5
Выполнить
Изобразить путь
2
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19
1
2
3
4
5


Работа с графами
Ввод графа из файла
Загрузить
Изобразить
Ввод с клавиатуры
Добавить вершину
Соединить две вершины
Сохранить граф
Сохранить
Редактирование графа
Удалить вершину
Удалить ребро
Задание
Введите вершины, через которые пройдет путь
1
3
Пути
1) 1->2->4->5->3
2) 1->4->5->2->3
3) 1->4->5->3
Введите ребро
4
5
Выполнить
Изобразить путь
3
Нарисовать
Об авторе
Ахонов К.Р. группа А-16-19
1
2
3
4
5

Листинг программы

form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Lw_tp_2
{
   public partial class Form1 : Form
   {
      int a, b, x, y;

      Class_graph F = new Class_graph(0);
      int[] delete_v = new int[100];
      int dv_length = 0;
      int[,] Rez_matr = new int[100, 100];

      public bool Check(int i)
      {
         bool flag = false;
         int j = 0;
         while(j < 100 && !flag)
         {
            if(delete_v[j] == i)
            {
               flag = true;
            }
            else
            {
               j++;
            }
         }
         return flag;
      }
      public Form1()
      {
         InitializeComponent();
```

```
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            int length_lines, a, b;
            string[] lines =
System.IO.File.ReadAllLines(@"C:\Users\kamil\Documents\ТП\Lw_tp_2\grap
h.txt");
            length_lines = lines.Length;
            for (int i = 0; i < length_lines; i++)
            {
                a = Convert.ToInt32(lines[i]);
                b = a / 10;
                a = a % 10;
                F.Input_graph(b, a);
            }

            MessageBox.Show("Граф успешно загружен", "Сообщение");
        }
        private void Button2_Click(object sender, EventArgs e)
        {

            double phi;
            float[] graphix_x;
            float[] graphix_y;
            double x_i, y_i, Rad;
            Rad = 200;
            string i_name;
            int num_vertices;
            Graphics g = pictureBox1.CreateGraphics();
            g.Clear(Color.FromArgb(246, 255, 143));
            SolidBrush myTrub = new SolidBrush(Color.DeepPink);
            Pen myWind = new Pen(Color.Yellow, 2);
            num_vertices = F.Num_of_vertex();
            graphix_x = new float[num_vertices];
            graphix_y = new float[num_vertices];
            for (int i = 1; i <= num_vertices; i++)
```

```
        {
            phi = Math.PI * i * 2 / num_vertices;
            x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
            y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
            if ((phi > Math.PI / 2) && (phi <= Math.PI))
            {
                x_i *= -1;
            }
            if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
            {
                x_i *= -1;
                y_i *= -1;
            }
            if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
            {
                y_i *= -1;

            }
            graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
            graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;

        }
        Pen p = new Pen(Color.Blue, 5);
        int j = 0;
        while (j < F.get_length())
        {
            g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
            j += 2;
        }
        int l = 1;
        while (l <= num_vertices)
        {
            i_name = Convert.ToString(l);
            g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15, 30,
30);
```

```
            g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
            l++;
         }
      }
      private void button3_Click(object sender, EventArgs e)
      {
         int num_vertex = F.Num_of_vertex();
         if (F.get_element(0) == 0)
         {
            F.Input_graph(1, 0);
         }
         else
         {
            F.Input_graph(num_vertex + 1, 0);
         }
         double phi;
         float[] graphix_x;
         float[] graphix_y;
         double x_i, y_i, Rad;
         Rad = 200;
         string i_name;
         int num_vertices;
         Graphics g = pictureBox1.CreateGraphics();
         g.Clear(Color.FromArgb(246, 255, 143));
         SolidBrush myTrub = new SolidBrush(Color.DeepPink);
         Pen myWind = new Pen(Color.Yellow, 2);
         num_vertices = F.Num_of_vertex();
         graphix_x = new float[num_vertices];
         graphix_y = new float[num_vertices];
         for (int i = 1; i <= num_vertices; i++)
         {
            phi = Math.PI * i * 2 / num_vertices;
            x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
            y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
            if ((phi > Math.PI / 2) && (phi <= Math.PI))
            {
```

```
            x_i *= -1;
        }
        if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
        {
            x_i *= -1;
            y_i *= -1;
        }
        if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
        {
            y_i *= -1;

        }
        graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
        graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;

    }
    Pen p = new Pen(Color.Blue, 5);
    int j = 0;
    while (j < F.get_length())
    {
        if (F.get_element(j + 1) == 0)
        {
            j += 2;
        }
        else
        {
            g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
            j += 2;
        }

    }
    int l = 1;
    while (l <= num_vertices)
    {
        i_name = Convert.ToString(l);
```

```
            g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15, 30,
30);
            g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
            l++;
         }

      }
      public void button4_Click(object sender, EventArgs e)
      {
         int num_vertex = F.Num_of_vertex();
         int v_1, v_2;
         v_1 = Convert.ToInt32(textBox1.Text);
         v_2 = Convert.ToInt32(textBox2.Text);
         F.Input_graph(v_1, v_2);
         double phi;
         float[] graphix_x;
         float[] graphix_y;
         double x_i, y_i, Rad;
         Rad = 200;
         string i_name;
         int num_vertices;
         Graphics g = pictureBox1.CreateGraphics();
         g.Clear(Color.FromArgb(246, 255, 143));
         SolidBrush myTrub = new SolidBrush(Color.DeepPink);
         Pen myWind = new Pen(Color.Yellow, 2);
         num_vertices = F.Num_of_vertex();
         graphix_x = new float[num_vertices];
         graphix_y = new float[num_vertices];
         for (int i = 1; i <= num_vertices; i++)
         {
            phi = Math.PI * i * 2 / num_vertices;
            x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
            y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
            if ((phi > Math.PI / 2) && (phi <= Math.PI))
            {
               x_i *= -1;
```

```
            }
            if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
            {
               x_i *= -1;
               y_i *= -1;
            }
            if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
            {
               y_i *= -1;

            }
            graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
            graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;

         }
         Pen p = new Pen(Color.Blue, 5);// цвет линии и ширина
         int j = 0;
         while (j < F.get_length())
         {
            if (F.get_element(j + 1) == 0)
            {
               j += 2;
            }
            else
            {
               g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
               j += 2;
            }
         }
         int l = 1;
         while (l <= num_vertices)
         {
            i_name = Convert.ToString(l);
            g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15, 30,
30);
```

```
            g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
            l++;
         }
         textBox1.Clear();
         textBox2.Clear();
      }
      public void button5_Click(object sender, EventArgs e)
      {
         int a, b, i, j;
         string[] lines = new string[F.get_length() / 2 + 5];
         string x, y;
         i = 0;
         j = 0;
         while (i < F.get_length())
         {
            a = F.get_element(i);
            b = F.get_element(i + 1);
            x = Convert.ToString(a);
            y = Convert.ToString(b);
            lines[j] = String.Concat(x, y);
            i += 2;
            j++;
         }

System.IO.File.AppendAllLines(@"C:\Users\kamil\Documents\ТП\Lw_tp_2\sa
ved_graph.txt", lines);
         MessageBox.Show("Граф успешно сохранен", "Сообщение");
      }

      private void textBox3_TextChanged(object sender, EventArgs e)
      {

      }
      public void button6_Click(object sender, EventArgs e)
      {
         int t;
```

```
t = Convert.ToInt32(textBox3.Text);
F.Delete_vertex(t);
if (delete_v[0] != 0)
{

   delete_v[dv_length] = t;
   dv_length += 1;
}
else
{
   delete_v[0] = t;
   dv_length = 1;
}
double phi;
float[] graphix_x;
float[] graphix_y;
double x_i, y_i, Rad;
Rad = 200;
string i_name;
int num_vertices;
Graphics g = pictureBox1.CreateGraphics();
g.Clear(Color.FromArgb(246, 255, 143));
SolidBrush myTrub = new SolidBrush(Color.DeepPink);
num_vertices = F.Num_of_vertex();
graphix_x = new float[num_vertices];
graphix_y = new float[num_vertices];
for (int i = 1; i <= num_vertices; i++)
{
   if (Check(i))
   {
      continue;
   }
   else
   {
      phi = Math.PI * i * 2 / num_vertices;
      x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
      y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
```

```
                if ((phi > Math.PI / 2) && (phi <= Math.PI))
                {
                    x_i *= -1;
                }
                if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
                {
                    x_i *= -1;
                    y_i *= -1;
                }
                if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
                {
                    y_i *= -1;

                }
                graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
                graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;
            }
        }
        Pen p = new Pen(Color.Blue, 5);// цвет линии и ширина
        int j = 0;
        while (j < F.get_length())
        {
            if ((graphix_x[F.get_element(j) - 1] == 0 &&
graphix_y[F.get_element(j) - 1] == 0) || (graphix_x[F.get_element(j+1) - 1] == 0
&& graphix_y[F.get_element(j+1) - 1] == 0))
            {
                j += 2;
            }
            else
            {
                g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
                j += 2;
            }
        }
        int l = 1;
```

```
            while (l <= num_vertices)
            {
               if (Check(l))
               {
                  l++;
               }
               else
               {
                  i_name = Convert.ToString(l);
                  g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15,
30, 30);
                  g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
                  l++;
               }
            }
         }
         public void button7_Click(object sender, EventArgs e)
         {
            int a, b;
            a = Convert.ToInt32(textBox4.Text);
            b = Convert.ToInt32(textBox5.Text);
            F.Delete_edge(a, b);
            double phi;
            float[] graphix_x;
            float[] graphix_y;
            double x_i, y_i, Rad;
            Rad = 200;
            string i_name;
            int num_vertices;
            Graphics g = pictureBox1.CreateGraphics();
            g.Clear(Color.FromArgb(246, 255, 143));
            SolidBrush myTrub = new SolidBrush(Color.DeepPink);
            num_vertices = F.Num_of_vertex();
            graphix_x = new float[num_vertices];
            graphix_y = new float[num_vertices];
            for (int i = 1; i <= num_vertices; i++)
```

```
        {
            phi = Math.PI * i * 2 / num_vertices;
            x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
            y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
            if ((phi > Math.PI / 2) && (phi <= Math.PI))
            {
                x_i *= -1;
            }
            if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
            {
                x_i *= -1;
                y_i *= -1;
            }
            if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
            {
                y_i *= -1;

            }
            graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
            graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;
        }
        Pen p = new Pen(Color.Blue, 5);// цвет линии и ширина
        int j = 0;
        while (j < F.get_length())
        {
            g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
            j += 2;
        }
        int l = 1;
        while (l <= num_vertices)
        {
            i_name = Convert.ToString(l);
            g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15, 30,
30);
```

```
            g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
            l++;
        }
    }
    public void button8_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < 100; i++)
        {
            for (int j = 0; j < 100; j++)
            {
                Rez_matr[i, j] = 0;
            }
        }
        //int a, b, x, y;
        int kolp = 0;
        string[] s_mas;
        string s = "";
        int h;
        x = Convert.ToInt32(textBox6.Text);
        y = Convert.ToInt32(textBox7.Text);
        a = Convert.ToInt32(textBox8.Text);
        b = Convert.ToInt32(textBox9.Text);
        F.Search_and_build_road(F.Num_of_vertex(), x, y, a, b, ref kolp, ref
Rez_matr);
        s_mas = new string[kolp];
        for(int i = 0; i < kolp; i++)
        {
            for (int j = 0; j < 15; j++)
            {
                h = Rez_matr[i, j];
                if (h == 0)
                {
                    break;
                }
                else
                {
```

```
            if(h == y)
            {
               s = String.Concat(s, Convert.ToString(h));
            }
            else
            {
               s = String.Concat(s, Convert.ToString(h), "->");
            }

         }

      }
      s_mas[i] = String.Concat(Convert.ToString(i+1),") ",s);
      s = "";
   }
   textBox10.Lines = s_mas;

}

public void button9_Click(object sender, EventArgs e)
{
   int n;
   n = Convert.ToInt32(textBox11.Text);

   double phi;
   float[] graphix_x;
   float[] graphix_y;
   double x_i, y_i, Rad;
   Rad = 200;
   string i_name;
   int num_vertices;
   Graphics g = pictureBox1.CreateGraphics();
   g.Clear(Color.FromArgb(246, 255, 143));
   SolidBrush myTrub = new SolidBrush(Color.DeepPink);
   num_vertices = F.Num_of_vertex();
   graphix_x = new float[num_vertices];
   graphix_y = new float[num_vertices];
```

```
        for (int i = 1; i <= num_vertices; i++)
        {
            phi = Math.PI * i * 2 / num_vertices;
            x_i = Rad * (1 / (Math.Sqrt(1 + Math.Pow(Math.Tan(phi), 2))));
            y_i = Math.Sqrt(Math.Pow(Rad, 2) - Math.Pow(x_i, 2));
            if ((phi > Math.PI / 2) && (phi <= Math.PI))
            {
                x_i *= -1;
            }
            if ((phi > Math.PI) && (phi <= 1.5 * Math.PI))
            {
                x_i *= -1;
                y_i *= -1;
            }
            if ((phi > 1.5 * Math.PI) && (phi <= 2 * Math.PI))
            {
                y_i *= -1;

            }
            graphix_x[i - 1] = (float)15 + (float)x_i + pictureBox1.Width / 2;
            graphix_y[i - 1] = (float)15 + (float)y_i + pictureBox1.Height / 2;

        }
        Pen p = new Pen(Color.Blue, 5);// цвет линии и ширина
        int j = 0;
        while (j < F.get_length())
        {
            p = new Pen(Color.Blue, 5);
            int m = 0;
            while(Rez_matr[n-1, m+1] != 0)
            {
                if((F.get_element(j) == a && F.get_element(j + 1) == b) ||
(F.get_element(j + 1) == a && F.get_element(j) == b))
                {
                    p = new Pen(Color.Red, 5);
                    break;
                }
```

```
                else if((F.get_element(j) == Rez_matr[n - 1, m] && F.get_element(j
+ 1) == Rez_matr[n - 1, m + 1]) || (F.get_element(j + 1) == Rez_matr[n - 1, m]
&& F.get_element(j) == Rez_matr[n - 1, m + 1]))
                {
                    p = new Pen(Color.Green,5);
                    break;
                }
                m++;
            }
            g.DrawLine(p, graphix_x[F.get_element(j) - 1],
graphix_y[F.get_element(j) - 1], graphix_x[F.get_element(j + 1) - 1],
graphix_y[F.get_element(j + 1) - 1]);
            j += 2;
        }
        int l = 1;
        while (l <= num_vertices)
        {
            myTrub = new SolidBrush(Color.DeepPink);
            if (l == x || l == y)
            {
                myTrub = new SolidBrush(Color.DarkMagenta);
            }
            i_name = Convert.ToString(l);
            g.FillEllipse(myTrub, graphix_x[l - 1] - 15, graphix_y[l - 1] - 15, 30,
30);
            g.DrawString(i_name, new Font("Arial", 12), Brushes.Black,
graphix_x[l - 1] - 7, graphix_y[l - 1] - 8);
            l++;
        }

    }
    private void textBox11_TextChanged(object sender, EventArgs e)
    {

    }
  }
}
```

abstract_class_graph.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lw_tp_2
{
    class stack
    {
        private int[] S;
        public int S_length;

        public stack(int a)
        {
            S = new int[255];
            for (int i = 0; i < 255; i++)
            {
                S[i] = 0;
            }
            S_length = a;
        }

        public void Push(int elem)
        {
            if (S_length == 0)
            {
                S[0] = elem;
                S_length = 1;
            }
            else
            {
                S[S_length] = elem;
                S_length++;
            }
        }
```

```
        public int Pop()
        {
            int i = 0;
            if (S_length == 0)
            {
                return -1;
            }
            else
            {
                i = S[S_length - 1];
                S[S_length - 1] = 0;
                S_length--;
                return i;

            }
        }
        public int Get_this_position(int t)
        {
            return S[t];
        }
    }
    abstract class abstract_class_graph
    {
        private stack G;

        public abstract_class_graph()
        {
            G = new stack(0);
        }

        abstract public bool Search_vertex(int v, int j);

        abstract public bool Check_condition(int a, int b, stack P);

        public void Search_and_build_road(int n, int vn, int vk, int a, int b, ref int
kolp, ref int[,] Rez_Matr)
        {
```

```
int[] M = new int[100];
int v, i, j, L, Pr, Prt, ks;
L = 1;
kolp = 0;
ks = 0;
G.Push(vn);
M[vn] = 1;
while (ks >= 0)
{
   Pr = 0;
   v = G.Get_this_position(ks);
   for (j = L; j <= n; j++)
   {
      if (Search_vertex(v, j))
      {
         if (j == vk)
         {
            Prt = 1;
            G.Push(j);
            if (Check_condition(a, b, G))
               Prt = 0;
            G.Pop();
            if (Prt == 0)
            {
               for (i = 0; i <= ks; i++)
                  Rez_Matr[kolp, i] = G.Get_this_position(i);
               Rez_Matr[kolp, ks + 1] = vk;
               kolp++;
            }
         }
         else
         {
            if (M[j] == 0)
            {
               Pr = 1;
               break;
            }
```

```
                }

            }
        }
        if (Pr == 1)
        {
            ks++;
            G.S_length = ks;
            G.Push(j);
            L = 1;
            M[j] = 1;
        }
        else
        {
            L = v + 1;
            M[v] = 0;
            ks--;
            G.Pop();
        }
    }

  }
 }
}
```

Class_graph.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lw_tp_2
{
    class Class_graph : abstract_class_graph
    {
        private int[] List_adjacent_vertex;
        private int n;
```

```
public int get_element(int i)
{
   return List_adjacent_vertex[i];
}
public int get_length()
{
   return n;
}

public void Input_graph(int a, int b)
{
   if (n == 0)
   {
      List_adjacent_vertex[0] = a;
      List_adjacent_vertex[1] = b;
      n = 2;
   }
   else
   {
      List_adjacent_vertex[n] = a;
      List_adjacent_vertex[n + 1] = b;
      n += 2;
   }
}
public void Delete_vertex(int k)
{
   int i = 0;
   int j = 0;
   while (j < n)
   {
      if (List_adjacent_vertex[j] == k || List_adjacent_vertex[j + 1] == k)
      {
         i = j + 2;
         while (i < n)
         {
            List_adjacent_vertex[i - 2] = List_adjacent_vertex[i];
```

```
                List_adjacent_vertex[i - 1] = List_adjacent_vertex[i + 1];
                i += 2;
            }
            List_adjacent_vertex[i - 2] = 0;
            List_adjacent_vertex[i - 1] = 0;
            n -= 2;
        }
        j += 2;
    }
}
public void Delete_edge(int a, int b)
{
    int i = 0;
    int j = 0;
    while (j < n)
    {
        if ((List_adjacent_vertex[j] == a && List_adjacent_vertex[j + 1] ==
b))
        {
            i = j + 2;
            while (i < n)
            {
                List_adjacent_vertex[i - 2] = List_adjacent_vertex[i];
                List_adjacent_vertex[i - 1] = List_adjacent_vertex[i + 1];
                i += 2;
            }
            List_adjacent_vertex[i - 2] = 0;
            List_adjacent_vertex[i - 1] = 0;
            n -= 2;
        }
        j += 2;
    }
}
public int Num_of_vertex()
{
    int max = List_adjacent_vertex[0];
    int i = 1;
```

```
        while (i < n)
        {
            if (List_adjacent_vertex[i] >= max)
            {
                max = List_adjacent_vertex[i];
                i++;
            }
            else
            {
                i++;
            }
        }
        return max;
    }
    public Class_graph(int a)
    {
        List_adjacent_vertex = new int[100];
        n = a;
    }

    public override bool Check_condition(int a, int b, stack P)
    {
        bool flag = false;
        int i = 0;
        while ((i < P.S_length) && (!flag))
        {
            if (((P.Get_this_position(i) == a) && (P.Get_this_position(i + 1) ==
b)) || ((P.Get_this_position(i + 1) == a) && (P.Get_this_position(i) == b)))
            {
                flag = true;
            }
            else
            {
                i += 1;
            }
        }
        return flag;
```

```
        }
        public override bool Search_vertex(int v, int j)
        {
            int i = 0;
            bool flag = false;
            while ((!flag) && (i < n))
            {
                if ((List_adjacent_vertex[i] == v) && (List_adjacent_vertex[i + 1] ==
j) || (List_adjacent_vertex[i + 1] == v) && (List_adjacent_vertex[i] == j))
                {
                    flag = true;
                }
                else
                {
                    i += 2;
                }
            }
            return flag;
        }
    }
}
```

program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lw_tp_2
{
    static class Data
    {
        public static string Value { get; set; }
    }

    static class Program
    {
```

```
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```