СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| ВВЕДЕНИЕ  1 Разработка программы  1.1 Разработка подпрограммы построения графического представления графа  1.2 Разработка интерфейса  2 Тестирование программы  ЗАКЛЮЧЕНИЕ  СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ  ПРИЛОЖЕНИЕ А - Исходный код программы | 3  4  4  7  8  10  11  12 |

ВВЕДЕНИЕ

В задачах теории графов небольшой размерности удобно иметь графическое представление графа, которое включает в себя вершины и соединяющие их ребра.

В рамках данной работы рассматривается разработка программы для графического отображения графа по матрице смежности. Целью работы является изучение особенностей разработки программ и проектирования экранных форм на языке C#. С этой целью были выделены 3 задачи:

1) разработать программу для построения графического представления с использованием элементов экранной формы;

2) спроектировать интерфейс экранной формы.

**1 Разработка программы**

**1.1 Разработка подпрограммы построения графического представления графа**

Для построения графического представления графа используем компонент Windows.Forms Graphics, позволяющий рисовать линии и фигуры на элементах формы. В качестве такового примем элемент типа Panel.

Для составления представления графа введём ряд структур для сохранения информации о его графическом представлении (представлены на рисунке 1):

1) struct Arrow – содержит координаты 3-х точек, составляющих графическое представление стрелки;

2) struct Line – содержит координаты 2-х точек, необходимых для построения отрезка и компоненту Arrow;

3) struct GraphImage – содержит информацию о числе вершин, их координатах в графическом представлении, радиусе, а также массив компонент типа Line, являющиеся представлением рёбер графа.

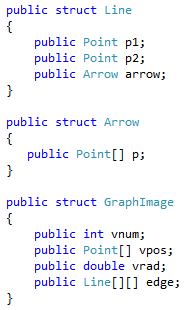


Рисунок 1 – Структуры для графического представления графа

Для построения графического представления графа и заполнения информации в вышеперечисленных структурах введём метод GraphImage generateGraphImage(bool[][] adjMatrix, double vrad, int width, int height), который генерирует представление графа по матрице смежности adjMatrix с вершинами радиуса vrad ширины width и высоты height. Метод устроен следующим образом, вначале случайным образом генерируются координаты точек в пределах заданного размера с учётом радиуса вершин, а затем смежные вершины соединяются линиями со стрелками, также с учётом радиуса вершин. Реализация метода представлена на рисунке 2.

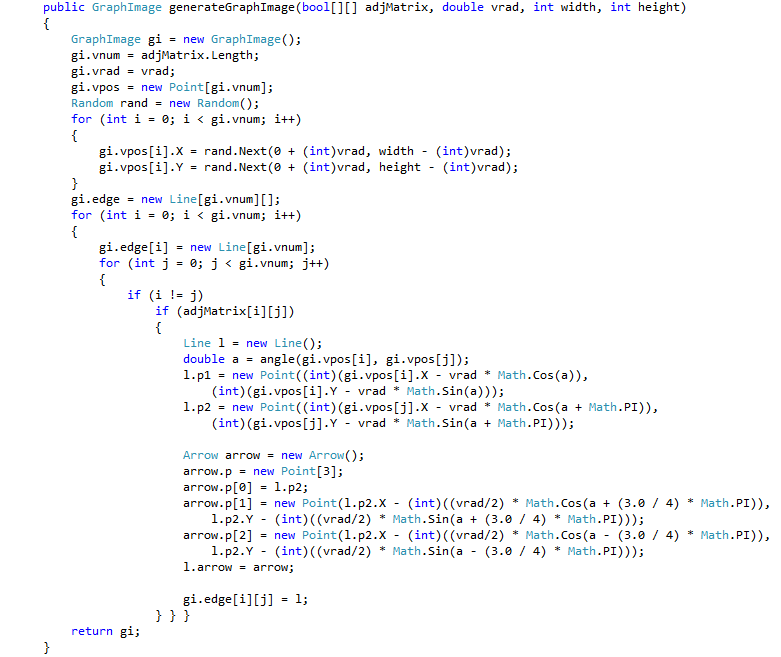


Рисунок 2 – Метод generateGraphImage(…)

После генерации графического представления графа его можно отрисовать. Введём для этого метод drawGraph(GraphImage gi), который рисует графическое представление gi. Данный метод рисует вершины с подписями и соединяющие их рёбра. Для рисования используются методы Graphics: DrawEllipse(…), DrawString(…), DrawLine(…) и DrawPolygon(…). Реализация метода представлена на рисунках 3.

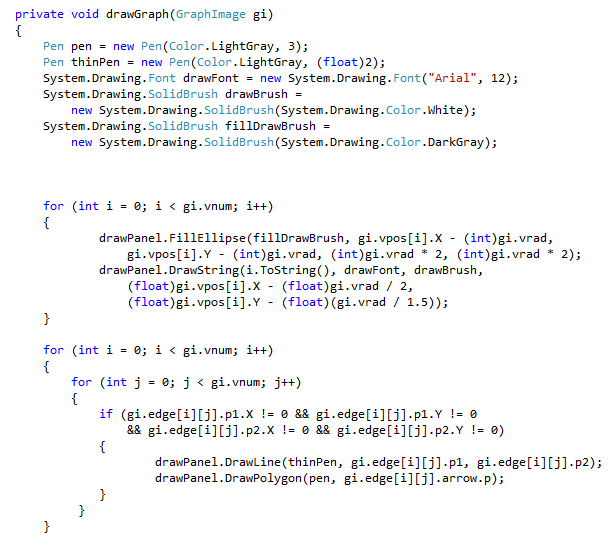


Рисунок 3 – Метод drawGraph(…)

Для примера рассмотрим результат работы методов generateGraphImage(…) и drawGraph(…) для следующей матрицы смежности (рисунок 4) с выделением пути [0, 3, 2] . Граф представлен на рисунке 5.

C:\Users\ПК\Desktop\3.PNG

Рисунок 4 – Матрица смежности отображаемого графа

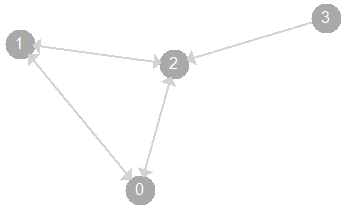


Рисунок 5 – Пример работы методов отображения графа

**1.3 Разработка интерфейса**

Рассмотрим интерфейс разработанной программы (рисунок 6). Форма разделена на три части: в верхней части вводятся данные графа, по нажатию кнопки «Нарисовать» на панели Panel рисуется соответствующий граф. В нижней части расположена панель графического представления графа.

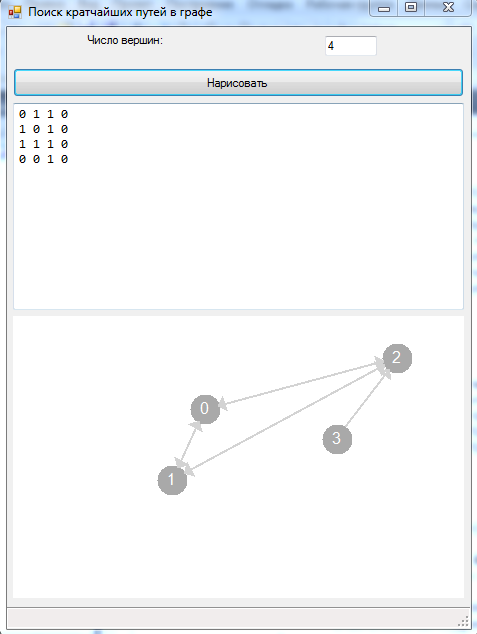


Рисунок 6 – Интерфейс программы

Также в нижней части формы имеется строка подсказок, в которой выводится информация об ошибках ввода.

**2. Тестирование программы**

Рассмотрим пример работы программы (Рисунок 7): сгенерированное представление графа соответствует заданной матрице смежности, что говорит о правильности работы программы.

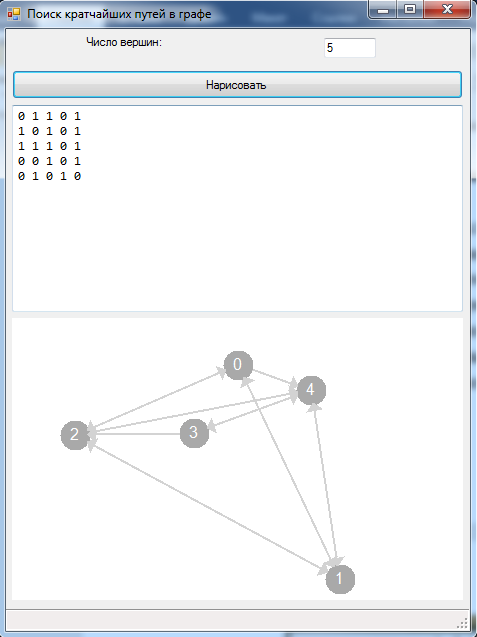


Рисунок 7 – Тестовый пример

ЗАКЛЮЧЕНИЕ

В результате выполнения этой работы была разработана программа для графического представления графа по матрице смежности. Также был спроектирован интерфейс экранной формы. Были получены навыки разработки программ и проектирования форм на языке C#. Поставленные задачи выполнены. Цель достигнута.

СПИСОК ИСПОЛЗОВАНЫХ ИСТОЧНИКОВ

1) Воронина, В.В. Типовые алгоритмы и их реализации на языке C# [Текст] / В.В. Воронина. – Ульяновск: УлГТУ, 2013. – 35 с.

2) Технология разработки объектно-ориентированных программна языке C# в среде Visual Studio.Net [Текст]: метод. указ. к лаб. раб. / сост. И.А. Васюткина, реценз. С.Г. Юн. – Новосибирск: НГТУ, 2010. – 85 с.

ПРИЛОЖЕНИЕ А

**Исходный код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Graph\_SearchPath

{

public partial class Form1 : Form

{

bool[][] adjMatrix;

Graphics drawPanel;

GraphImage graphImage;

public struct Line

{

public Point p1;

public Point p2;

public Arrow arrow;

}

public struct Arrow

{

public Point[] p;

}

public struct GraphImage

{

public int vnum;

public Point[] vpos;

public double vrad;

public Line[][] edge;

}

public GraphImage generateGraphImage(bool[][] adjMatrix, double vrad, int width, int height)

{

GraphImage gi = new GraphImage();

gi.vnum = adjMatrix.Length;

gi.vrad = vrad;

gi.vpos = new Point[gi.vnum];

Random rand = new Random();

for (int i = 0; i < gi.vnum; i++)

{

gi.vpos[i].X = rand.Next(0 + (int)vrad, width - (int)vrad);

gi.vpos[i].Y = rand.Next(0 + (int)vrad, height - (int)vrad);

}

gi.edge = new Line[gi.vnum][];

for (int i = 0; i < gi.vnum; i++)

{

gi.edge[i] = new Line[gi.vnum];

for (int j = 0; j < gi.vnum; j++)

{

if (i != j)

if (adjMatrix[i][j])

{

Line l = new Line();

double a = angle(gi.vpos[i], gi.vpos[j]);

l.p1 = new Point((int)(gi.vpos[i].X - vrad \* Math.Cos(a)),

(int)(gi.vpos[i].Y - vrad \* Math.Sin(a)));

l.p2 = new Point((int)(gi.vpos[j].X - vrad \* Math.Cos(a + Math.PI)),

(int)(gi.vpos[j].Y - vrad \* Math.Sin(a + Math.PI)));

Arrow arrow = new Arrow();

arrow.p = new Point[3];

arrow.p[0] = l.p2;

arrow.p[1] = new Point(l.p2.X - (int)((vrad/2) \* Math.Cos(a + (3.0 / 4) \* Math.PI)),

l.p2.Y - (int)((vrad/2) \* Math.Sin(a + (3.0 / 4) \* Math.PI)));

arrow.p[2] = new Point(l.p2.X - (int)((vrad/2) \* Math.Cos(a - (3.0 / 4) \* Math.PI)),

l.p2.Y - (int)((vrad/2) \* Math.Sin(a - (3.0 / 4) \* Math.PI)));

l.arrow = arrow;

gi.edge[i][j] = l;

} } }

return gi;

}

public Form1()

{

InitializeComponent();

drawPanel = panelGraphics.CreateGraphics();

}

double length(Point p1, Point p2)

{

return Math.Sqrt(Math.Pow(p2.X - p1.X, 2) + Math.Pow(p2.Y - p1.Y, 2));

}

double angle(Point p1, Point p2)

{

double A = Math.Atan2(p1.Y - p2.Y, p1.X - p2.X) /\*/ Math.PI \* 180\*/;

//A = (A < 0) ? A + 360 : A;

return A;

}

int digitNum(int number)

{

int del = number;

for (int i = 1; ; i++)

{

del /= 10;

if (del == 0) return i;

}

}

private void drawGraph(GraphImage gi)

{

Pen pen = new Pen(Color.LightGray, 3);

Pen thinPen = new Pen(Color.LightGray, (float)2);

System.Drawing.Font drawFont = new System.Drawing.Font("Arial", 12);

System.Drawing.SolidBrush drawBrush =

new System.Drawing.SolidBrush(System.Drawing.Color.White);

System.Drawing.SolidBrush fillDrawBrush =

new System.Drawing.SolidBrush(System.Drawing.Color.DarkGray);

for (int i = 0; i < gi.vnum; i++)

{

drawPanel.FillEllipse(fillDrawBrush, gi.vpos[i].X - (int)gi.vrad,

gi.vpos[i].Y - (int)gi.vrad, (int)gi.vrad \* 2, (int)gi.vrad \* 2);

drawPanel.DrawString(i.ToString(), drawFont, drawBrush,

(float)gi.vpos[i].X - (float)gi.vrad / 2,

(float)gi.vpos[i].Y - (float)(gi.vrad / 1.5));

}

for (int i = 0; i < gi.vnum; i++)

{

for (int j = 0; j < gi.vnum; j++)

{

if (gi.edge[i][j].p1.X != 0 && gi.edge[i][j].p1.Y != 0

&& gi.edge[i][j].p2.X != 0 && gi.edge[i][j].p2.Y != 0)

{

drawPanel.DrawLine(thinPen, gi.edge[i][j].p1, gi.edge[i][j].p2);

drawPanel.DrawPolygon(pen, gi.edge[i][j].arrow.p);

}

}

}

}

private void clearGraphics()

{

drawPanel.Clear(Color.White);

}

private void btnOk\_Click(object sender, EventArgs e)

{

try

{

int vertexNum = 0;

try

{

vertexNum = int.Parse(tbVertexNum.Text);

}

catch (FormatException fe) { labelError.Text = "Число вершин введено неверно!"; throw fe; }

//

//Проверки на корректность данных

//

try

{

if (vertexNum < 0) throw new FormatException();

}

catch (FormatException fe) { labelError.Text = "Число вершин графа не может быть отрицательным!"; throw fe; }

//

//

String[] matrixStr = tbMatrix.Text.Split('\n');

try

{

if (matrixStr.Length != vertexNum) throw new FormatException();

}

catch (FormatException fe) { labelError.Text = "Число строк матрицы не соответсвует числу вершин!"; throw fe; }

adjMatrix = new bool[matrixStr.Length][];

try

{

for (int i = 0; i < matrixStr.Length; i++)

{

String[] matrixStrEl = matrixStr[i].Split(' ');

try

{

if (matrixStrEl.Length != vertexNum) throw new FormatException();

}

catch (FormatException fe) { labelError.Text = "Число элементов матрицы в " + i + " строке не соответсвует числу вершин!"; throw fe; }

adjMatrix[i] = new bool[matrixStrEl.Length];

for (int j = 0; j < matrixStrEl.Length; j++)

{

adjMatrix[i][j] = int.Parse(matrixStrEl[j]) == 1? true : false;

}

}

}

catch (FormatException fe) { labelError.Text = "Матрица смежности введена неверно!"; throw fe; }

clearGraphics();

graphImage = generateGraphImage(adjMatrix, 15.0 , panelGraphics.Width, panelGraphics.Height);

drawGraph(graphImage);

labelError.Text = " ";

}

catch (FormatException fe) { }

}

private void panelGraphics\_Resize(object sender, EventArgs e)

{

drawPanel = panelGraphics.CreateGraphics();

clearGraphics();

drawGraph(graphImage);

}

private void btnRedrawGraph\_Click(object sender, EventArgs e)

{

if (adjMatrix != null)

{

clearGraphics();

graphImage = generateGraphImage(adjMatrix, 15.0, panelGraphics.Width, panelGraphics.Height);

drawGraph(graphImage);

}

}

private void lbPaths\_SelectedIndexChanged(object sender, EventArgs e)

{

clearGraphics();

drawGraph(graphImage);

}

private void panelGraphics\_Paint(object sender, PaintEventArgs e)

{

}

}

}