МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Ижевский государственный технический университет

имени М. Т. Калашникова»

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Курсовая работа

по дисциплине «Алгоритмы и структуры данных»

На тему «Построение оптимального маршрута»

Выполнил

студент группы Б19-191-2з А.Р.Закиров

Руководитель

Ст. преп. Кафедры ПО М. О.Еланцев

Рецензия

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ижевск 2021 СОДЕРЖАНИЕ

Введение3

1. Описание задания4

2. Анализ предметной области6

3. Система классов7

4. Выбор и обоснование языка программирования8

5. Контрольный пример9

6. Код программы12

Заключение29

ВВЕДЕНИЕ

Задача построения оптимального маршрута, впервые поднятая в 1832 году в книге «Коммивояжер – как он должен вести себя и что должен делать для того, чтобы доставлять товар и иметь успех в своих делах – советы старого курьера», является актуальной и на сегодняшний день – разрабатываются новые методы решения задачи, реализуются программы, которые позволяют работать с количеством узлов близким к миллиону за приемлемое время. Неугасаемый интерес к этой задачи обусловлен разнообразием применений ее на практике. Поиск оптимального пути широко используется во всех задачах транспортной логистики, на производстве – в виде задач минимизации времени переналадки и при работе дыропробивных прессов.

Вопросы оптимизации маршрута поднимаются в трудах таких известных деятелей математики как Уильям Роуэн Гамильтон, Джордж Данциг, Ричард Мэннинг Карп, Дэвид Аплгейт, Герхард Райнельт.

Несмотря на обширную теоретическую базу, к сожалению, представлено не так много приложений, позволяющих людям, далеким от математики и программирования, использовать существующую разработки для решения практических задач. Пользователь, который желает обойти определённое количество мест за минимальное время, используя такие популярные картографические сервисы как «Яндекс.Карты» и «Google Карты», не может решить поставленную задачу, так как маршрут в приложении строится по заранее заданному в определённом порядке списку мест.

1. ОПИСАНИЕ ЗАДАНИЯ

Задача данной курсовой работы - на карте размещено несколько городов, связанных в виде графа. Каждое ребро графа обозначает допустимость пути из одного города в другой и содержит число – расстояние между городами. Оператор вводит начальный город и список городов адресатов.

Необходимо:

1. Составить оптимальный маршрут (с минимальной длиной пути), стартующий от начального города и проходящий по всем городам-адресатам.
2. Маршрут должен быть представлен в виде списка городов в порядке их посещения, каждому из которых добавлен один из следующих признаков: начальный город, город-адресат, транзитный город (не является адресатом, либо повторный въезд в город-адресат).
3. Город может несколько раз присутствовать в списке, если этого требует оптимальный маршрут.
4. Карта городов должна читаться из файла.

На рисунке №1 изображён граф представляющий карту, для которого будет выстраиваться алгоритм программы.

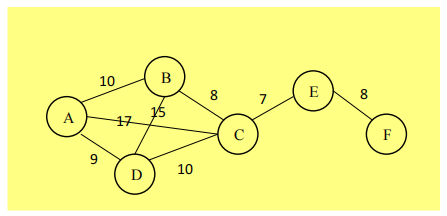


Рис. № 1

На рисунке №2 отображён вид информации о вершинах графа и весе рёбер между представленной в файле, из которого будет получена информация о графе

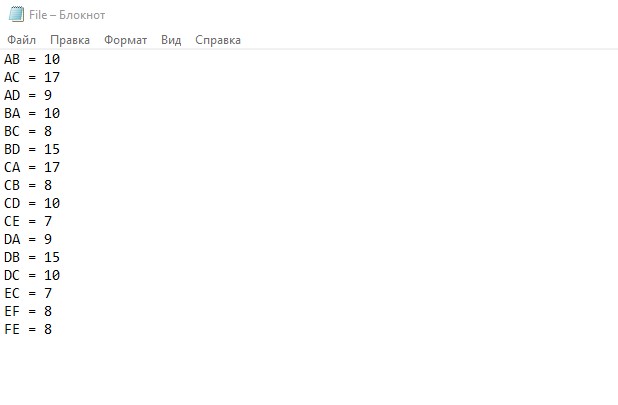


Рис. № 2

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Популярные картографические сервисы вроде Яндекс.Карты и Google Maps не предоставляют пользователям возможность поиска оптимально пути. При вводе нескольких координат сервис выстраивает маршрут в том порядке, в котором данные были введены. Пользователи могут выбирать средства передвижения (на машине, пешком, на наземном транспорте), но все эти настройки влияют исключительно на варианты построения маршрута между его фиксированными точками.

Анализ, проведенный путем сравнением десятков как русскоязычных, так и зарубежных картографических сервисов показывает, что среди самых популярных вариантов только у одного доступна функция построения оптимально пути, и далеко не всегда она работает корректно.

Дла решения поставленной задачи мною был выбран алгоритм Де́йкстры. Это [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Английский язык) на [графах](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "), изобретённый нидерландским учёным [Эдсгером Дейкстрой](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5" \o "Граф (математика)) - голландский учёный, труды которого оказали влияние на развитие [информатики](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0" \o "Информатика) и [информационных технологий](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8" \o "Информационные технологии); один из разработчиков концепции [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Структурное программирование), исследователь [формальной верификации](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D1%80%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F" \o ") и [распределённых вычислений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F" \o "Распределённые вычисления). [Тьюринговский лауреат](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BC%D0%B8%D1%8F_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0" \o "Премия Тьюринга) (1972).

Данный алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без [рёбер](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B1%D1%80%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)" \o ") отрицательного [веса](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2" \l "%D0%92" \o "). Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации [OSPF](https://ru.wikipedia.org/wiki/OSPF" \o "1959 год) и [IS-IS](https://ru.wikipedia.org/wiki/IS-IS" \o ").

3. СИСТЕМА КЛАССОВ

В данном разделе представлена структура классов проекта и их взаимодействие.

Program главный класс запускающий программу и осуществляющий чтение информации из файла

Form1 Класс отвечающий за функциональность интерфеса программы

Dijkstra Класс описывающий алгоритм Дейкстры, с помошью которого осуществляются вычисления оптимального маршрута

Graph

Класс описывающий граф(карту) в виде списка вершин

GraphVertexInfo

Класс описывающий дополнительную информацию о вершине

GraphVertex

Класс описывающий вершину графа (город)

GraphEdge

Класс описывающий ребра вершины

1. ВЫБОР И ОБОСНОВАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ

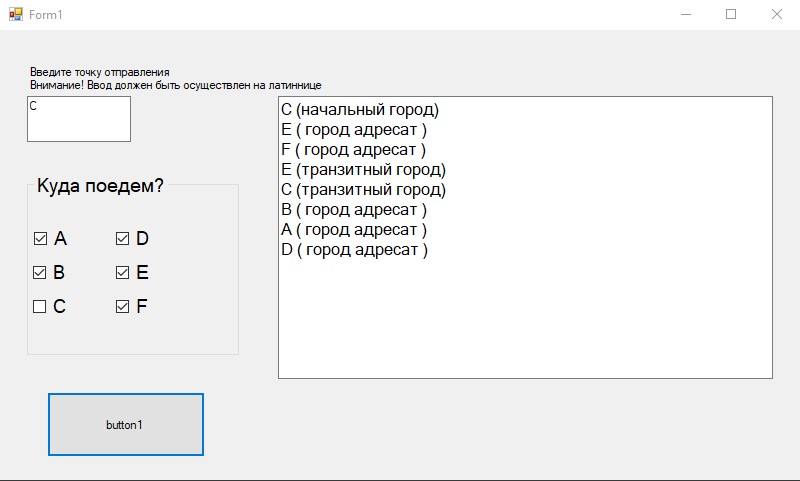
Для выполнения данной работы мною был выбран язык программирования C# (си шарп), так как на мой взгляд на сегодняшний день язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

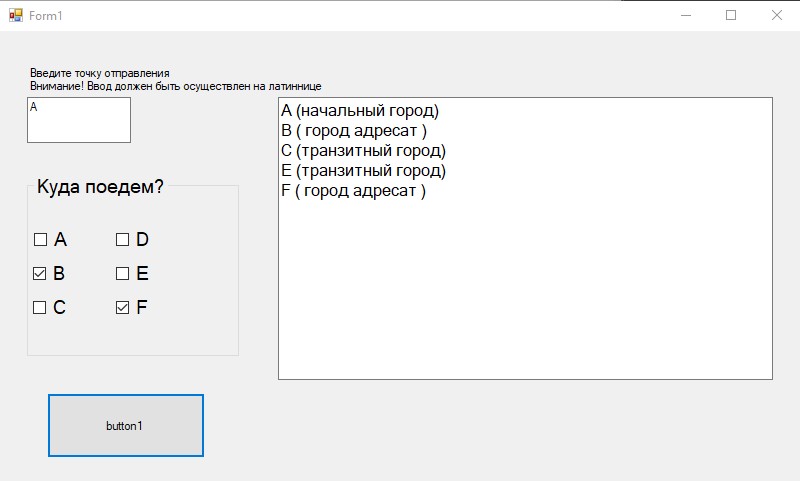
C# уже не молодой язык и как и вся платформа .NET уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 9.0, которая вышла 10 ноября 2020 года вместе с релизом .NET 5.

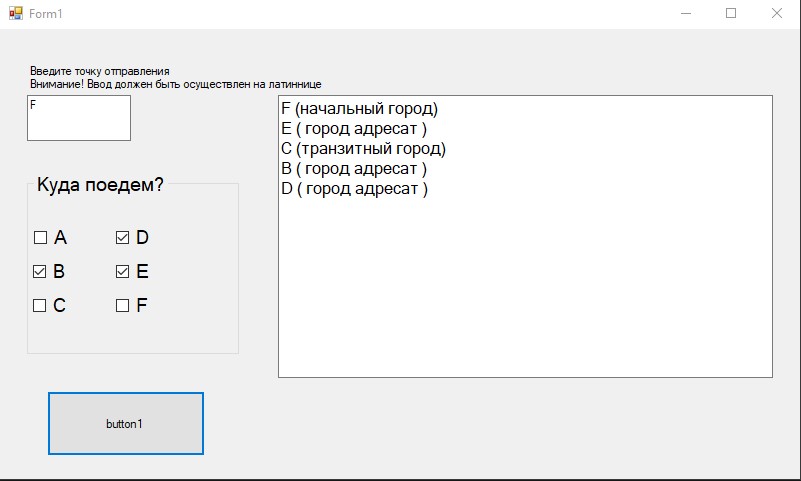
C# является объектно-ориентированным и поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

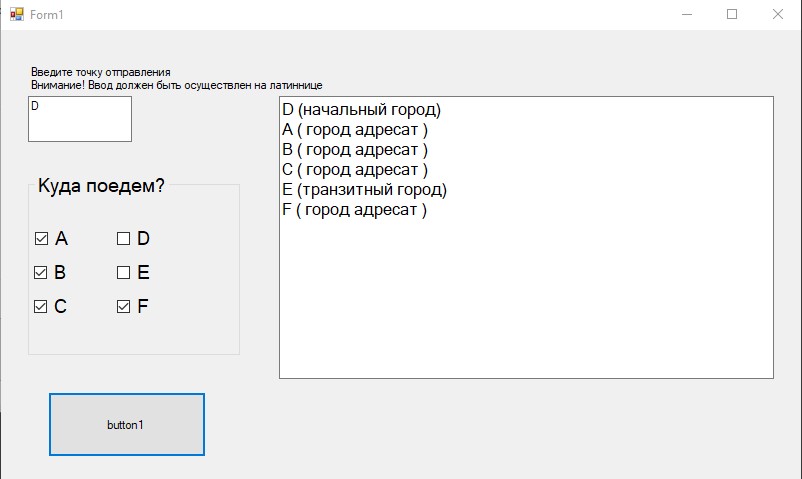
Задание было решено выполнить в виде проекта Windows Forms. Windows Forms — [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9" \o "Интерфейс программирования приложений) (API), отвечающий за [графический интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F" \o "Графический интерфейс пользователя) и являющийся частью [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework). Данный интерфейс упрощает доступ к элементам интерфейса [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows) за счет создания обёртки для существующего [Win32 API](https://ru.wikipedia.org/wiki/Windows_API" \o "Windows API) в [управляемом коде](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4" \o "Управляемый код).

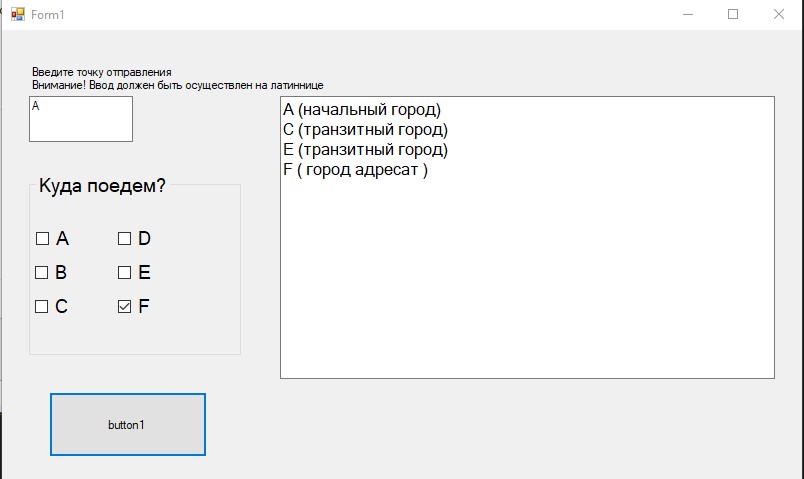
5. КОНТРОЛЬНЫЙ ПРИМЕР











6. КОД ПРОГРАММЫ

Класс Program

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ShortWayApp

{

public static class Program

{

public static string path;

public static StringBuilder str1;

public static StringBuilder str2;

public static string[] citysArray;

public static Dijkstra dijkstra;

public static Graph g;

[STAThread]

static async Task Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

str1 = new StringBuilder();

str2 = new StringBuilder();

string file = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"File.txt");

citysArray = new string[6];

try

{

using (StreamReader sr = new StreamReader(file))

{

str1.Append(await sr.ReadToEndAsync());

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

citysArray = CreateCitysArray(str1, citysArray);

g = new Graph();

foreach(var point in citysArray)

{

g.AddVertex(point);

}

GraphCreate(g, str1);

dijkstra = new Dijkstra(g);

dijkstra.InitInfo();

Application.Run(new Form1());

}

public static string[] CreateCitysArray(StringBuilder str1, string[] citysArray)

{

for (int i = 0, j = 0; i < str1.Length; i++)

{

if (char.IsLetter(str1[i]))

{

if (citysArray[0] == null)

{

citysArray[0] = str1[i].ToString();

}

else

{

int count = 0;

foreach (var city in citysArray)

{

if (str1[i].ToString() != city)

{

count++;

}

}

if (count + 1 != citysArray.Length)

{

citysArray[j] = str1[i].ToString();

j++;

}

}

}

}

return citysArray;

}

public static void GraphCreate(Graph g, StringBuilder str1)

{

for (int i = 0; i < str1.Length; i++)

{

if (char.IsLetter(str1[i]) && char.IsLetter(str1[i + 1]))

{

string name = str1[i] + "" + str1[i + 1];

i += 5;

if (char.IsNumber(str1[i]))

{

StringBuilder st = new StringBuilder();

while (char.IsNumber(str1[i]))

{

st.Append(str1[i]);

if (i + 1 == str1.Length) break;

else i++;

}

g.AddEdge(name[0].ToString(), name[1].ToString(), int.Parse(st.ToString()));

}

}

}

}

}

}

Класс Form 1

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace ShortWayApp

{

public partial class Form1 : Form

{

public static string[] checkedCitys;

public Form1()

{

InitializeComponent();

groupBox1.Controls.Add(checkBox1);

groupBox1.Controls.Add(checkBox2);

groupBox1.Controls.Add(checkBox3);

groupBox1.Controls.Add(checkBox4);

groupBox1.Controls.Add(checkBox5);

groupBox1.Controls.Add(checkBox6);

}

private void button1\_Click(object sender, EventArgs e)

{

textBox2.Clear();

checkedCitys = GetCheckedCity();

string result = "";

string startVert = textBox1.Text.ToUpper();

string st = textBox1.Text.ToUpper();

string previousVert = "";

if(checkedCitys.Length < 2)

{

st = Program.dijkstra.FindShortestPath(startVert, checkedCitys[0]);

}

else

{

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

{

if(startVert.Length > 1)

{

startVert = startVert[startVert.Length - 1].ToString();

}

result = Program.dijkstra.FindShortestPath(startVert, checkedCitys, previousVert);

st += result;

previousVert = startVert;

startVert = result;

}

}

string[] returnedArr = st.ToCharArray().Select(i => (i.ToString())).ToArray();

returnedArr[0] += " (начальный город)";

for (int i = 1; i < returnedArr.Length; i++)

{

int indexF = st.IndexOf(returnedArr[i][0].ToString());

int indexL = st.LastIndexOf(returnedArr[i][0].ToString());

int indexOfArr = Array.IndexOf(checkedCitys, returnedArr[i][0].ToString());

if (indexOfArr < 0)

{

returnedArr[i] += " (транзитный город)";

}

else if (indexF >= 0 && indexF == indexL)

{

returnedArr[i] += " ( город адресат )";

}

else if (indexF >= 0 && indexL >= 0 && indexF != indexL)

{

returnedArr[indexF] += " ( город адресат )";

returnedArr[indexL] += " (транзитный город)";

}

}

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

{

if(returnedArr[i].Length >= 20)

{

returnedArr[i] = returnedArr[i].Substring(0, 20);

}

textBox2.Text += returnedArr[i] + Environment.NewLine;

}

}

private string[] GetCheckedCity()

{

int count = 0;

foreach(CheckBox chb in groupBox1.Controls)

{

if (chb.Checked)

{

var elem = Program.g.FindVertex(chb.Text);

if(elem != null)

{

count++;

}

}

}

string[] arr = new string[count];

arr[0] = textBox1.Text.ToUpper();

if (count == 0)

{

MessageBox.Show("Не выбран пункт назначения");

return arr;

}

else

{

int i = 0;

foreach (CheckBox chb in groupBox1.Controls)

{

if (chb.Checked)

{

var elem = Program.g.FindVertex(chb.Text);

if (elem != null)

{

arr[i] = chb.Text;

i++;

}

}

}

return arr;

}

}

}

}

Класс GraphVertex

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShortWayApp

{

public class GraphVertex

{

public string Name { get; }

public List<GraphEdge> Edges { get; }

public GraphVertex(string vertexName)

{

Name = vertexName;

Edges = new List<GraphEdge>();

}

public void AddEdge(GraphEdge newEdge)

{

Edges.Add(newEdge);

}

public void AddEdge(GraphVertex vertex, int edgeWeight)

{

AddEdge(new GraphEdge(vertex, edgeWeight));

}

public override string ToString() => Name;

}

Класс GraphEdge

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShortWayApp

{

public class GraphEdge

{

public GraphVertex ConnectedVertex { get; }

public int EdgeWeight { get; }

public GraphEdge(GraphVertex connectedVertex, int weight)

{

ConnectedVertex = connectedVertex;

EdgeWeight = weight;

}

}

}

Класс GraphVertexInfo

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShortWayApp

{

public class GraphVertexInfo

{

public GraphVertex Vertex { get; set; }

public bool IsUnvisited { get; set; }

public int EdgesWeightSum { get; set; }

public GraphVertex PreviousVertex { get; set; }

public GraphVertexInfo(GraphVertex vertex)

{

Vertex = vertex;

IsUnvisited = true;

EdgesWeightSum = int.MaxValue;

PreviousVertex = null;

}

}

}

Класс Graph

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShortWayApp

{

public class Graph

{

public List<GraphVertex> Vertices { get; }

public Graph()

{

Vertices = new List<GraphVertex>();

}

public void AddVertex(string vertexName)

{

Vertices.Add(new GraphVertex(vertexName));

}

public GraphVertex FindVertex(string vertexName)

{

foreach (var v in Vertices)

{

if (v.Name.Equals(vertexName))

{

return v;

}

}

return null;

}

public GraphVertex[] FindVertex(string[] vertexName)

{

int count = 0;

foreach (var v in Vertices)

{

int index = Array.IndexOf(vertexName, v.Name);

if (index > -1)

{

count++;

}

}

GraphVertex[] arr = new GraphVertex[count];

count = 0;

foreach (var v in Vertices)

{

int index = Array.IndexOf(vertexName, v.Name);

if (index > -1)

{

arr[count] = v;

count++;

}

}

return arr;

}

public void AddEdge(string firstName, string secondName, int weight)

{

var v1 = FindVertex(firstName);

var v2 = FindVertex(secondName);

if (v2 != null && v1 != null)

{

v1.AddEdge(v2, weight);

v2.AddEdge(v1, weight);

}

}

}

}

Класс RemoveForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace HotelApp

{

public partial class RemoveForm : Form

{

public RemoveForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int roomNum;

if(textBox1 == null)

{

throw new NullReferenceException();

}

bool isNum = int.TryParse(textBox1.Text,out roomNum);

if(isNum && roomNum > 0 && roomNum <= Form1.db.rooms.Length && Form1.db.rooms[roomNum -1].GetRoomStatus() != 0 )

{

MessageBox.Show(Form1.db.rooms[roomNum - 1].client.firstName + " " +

Form1.db.rooms[roomNum - 1].client.lastName + ".\n" +

"К оплате " + Form1.db.rooms[roomNum - 1].GetChek() + " рублей");

Form1.db.RemoveClient(roomNum);

this.Close();

}

else

{

MessageBox.Show("Номер введен не корректно, либо в данном номере ни кто не проживает");

}

}

}

}

Класс Dijkstra

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShortWayApp

{

public class Dijkstra

{

Graph graph;

List<GraphVertexInfo> infos;

public Dijkstra(Graph graph)

{

this.graph = graph;

}

public void InitInfo()

{

infos = new List<GraphVertexInfo>();

foreach (var v in graph.Vertices)

{

infos.Add(new GraphVertexInfo(v));

}

}

// Получение информации о вершине графа

public GraphVertexInfo GetVertexInfo(GraphVertex v)

{

foreach (var i in infos)

{

if (i.Vertex.Equals(v))

{

return i;

}

}

return null;

}

// Поиск непосещенной вершины с минимальным значением суммы

public GraphVertexInfo FindUnvisitedVertexWithMinSum()

{

var minValue = int.MaxValue;

GraphVertexInfo minVertexInfo = null;

foreach (var i in infos)

{

if (i.IsUnvisited && i.EdgesWeightSum < minValue)

{

minVertexInfo = i;

minValue = i.EdgesWeightSum;

}

}

return minVertexInfo;

}

// Поиск кратчайшего пути по названиям вершин

public string FindShortestPath(string startName, string finishName)

{

return FindShortestPath(graph.FindVertex(startName), graph.FindVertex(finishName));

}

public string FindShortestPath(string startName, string[] finishName, string previousVert)

{

return FindShortestPath(graph.FindVertex(startName), graph.FindVertex(finishName), graph.FindVertex(previousVert));

}

// Поиск кратчайшего пути по вершинам

public string FindShortestPath(GraphVertex startVertex, GraphVertex finishVertex)

{

InitInfo();

var first = GetVertexInfo(startVertex);

first.EdgesWeightSum = 0;

while (true)

{

var current = FindUnvisitedVertexWithMinSum();

if (current == null)

{

break;

}

SetSumToNextVertex(current);

}

return GetPath(startVertex, finishVertex);

}

public string FindShortestPath(GraphVertex startVertex, GraphVertex[] finishVertex, GraphVertex previousVert)

{

InitInfo();

var first = GetVertexInfo(startVertex);

first.EdgesWeightSum = 0;

int minVal = int.MaxValue;

GraphVertexInfo v = null;

while (true)

{

var current = FindUnvisitedVertexWithMinSum();

if (current == null)

{

break;

}

else

{

SetSumToNextVertex(current);

int index = -1;

foreach (var t in finishVertex)

{

if (t.Name == current.Vertex.Name)

{

index = 0;

}

}

if(minVal > current.EdgesWeightSum && index == 0 && current.Vertex != startVertex && current.IsUnvisited == false && current.Vertex != previousVert)

{

minVal = current.EdgesWeightSum;

v = current;

}

}

}

if (v.PreviousVertex != previousVert && previousVert != null)

{

return FindShortestPath(startVertex, v.Vertex).Substring(1);

}

else if(v.PreviousVertex == previousVert)

{

Array.Resize(ref Form1.checkedCitys, Form1.checkedCitys.Length + 1);

return previousVert.Name;

}

return v.Vertex.Name;

}

public void SetSumToNextVertex(GraphVertexInfo info)

{

info.IsUnvisited = false; // Вершина не посещена

foreach (var e in info.Vertex.Edges) // перебор ребер вершины

{

var nextInfo = GetVertexInfo(e.ConnectedVertex);

var sum = info.EdgesWeightSum + e.EdgeWeight;

if (sum < nextInfo.EdgesWeightSum)

{

nextInfo.EdgesWeightSum = sum;

nextInfo.PreviousVertex = info.Vertex;

}

}

}

string GetPath(GraphVertex startVertex, GraphVertex endVertex)

{

var path = endVertex.ToString();

while (startVertex != endVertex)

{

endVertex = GetVertexInfo(endVertex).PreviousVertex;

path = endVertex.ToString() + path;

}

return path;

}

}

}

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был проведен анализ предметной области, который выявил ряд проблем, с которыми сталкиваются пользователи программ для построения маршрутов.

Так же проведен анализ алгоритмов построения оптимального маршрута, по результатам которого мною был выбран, для решения поставленных задач, алгоритм Дейкстры находящий кратчайшие пути от одной из вершин графа до всех остальных.

На основе алгоритма было разработано оконное приложение, с интерфейсом реализованным с помощью Windows Forms и языка программирования C#. Приложение ShortWayApp прокладывает кратчайший маршрут между вершинами графа которые необходимо посетить, при этом отмечая вершину как город-адресат или транзитный город.

В ходе контрольных испытаний программа выполнила все поставленные перед курсовым проектом условия и требования в полном объёме, поэтому можно считать задачу выполненной.