KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Objektinis programavimas II (P175B123)

Darbų aplankas

Atliko:

IFF-8/11 gr. studentas

Donatas Petrikauskas

2019 m. vasario 18 d.

Priėmė:

Doc. Romas Marcinkevičius

KAUNAS 2019

TURINYS

1. Rekursija (L1) 3

1.1. Darbo užduotis 3

1.2. Grafinės vartotojo sąsajos schema 3

1.3. Sąsajoje panaudotų komponentų keičiamos savybės 3

1.4. Klasių diagrama 4

1.5. Programos vartotojo vadovas 4

1.6. Programos tekstas 4

1.7. Pradiniai duomenys ir rezultatai 9

1.8. Dėstytojo pastabos 10

2. Dinaminis atminties valdymas (L2) 11

2.1. Darbo užduotis 11

2.2. Grafinės vartotojo sąsajos schema 11

2.3. Sąsajoje panaudotų komponentų keičiamos savybės 11

2.4. Klasių diagrama 11

2.5. Programos vartotojo vadovas 11

2.6. Programos tekstas 11

2.7. Pradiniai duomenys ir rezultatai 11

2.8. Dėstytojo pastabos 11

3. Bendrinės klasės ir sąsajos (L3) 12

3.1. Darbo užduotis 12

3.2. Grafinės vartotojo sąsajos schema 12

3.3. Sąsajoje panaudotų komponentų keičiamos savybės 12

3.4. Klasių diagrama 12

3.5. Programos vartotojo vadovas 12

3.6. Programos tekstas 12

3.7. Pradiniai duomenys ir rezultatai 12

3.8. Dėstytojo pastabos 12

4. Kolekcijos ir išimčių valdymas (L4) 13

4.1. Darbo užduotis 13

4.2. Grafinės vartotojo sąsajos schema 13

4.3. Sąsajoje panaudotų komponentų keičiamos savybės 13

4.4. Klasių diagrama 13

4.5. Programos vartotojo vadovas 13

4.6. Programos tekstas 13

4.7. Pradiniai duomenys ir rezultatai 13

4.8. Dėstytojo pastabos 13

5. Deklaratyvusis programavimas (L5) 14

5.1. Darbo užduotis 14

5.2. Grafinės vartotojo sąsajos schema 14

5.3. Sąsajoje panaudotų komponentų keičiamos savybės 14

5.4. Klasių diagrama 14

5.5. Programos vartotojo vadovas 14

5.6. Programos tekstas 14

5.7. Pradiniai duomenys ir rezultatai 14

5.8. Dėstytojo pastabos 14

# Rekursija (L1)

## Darbo užduotis

**LD\_20.Maršrutai.**

Turizmo agentūra organizuoja kelionę po Lietuvą. Reikia parašyti programą, kuri pasiūlytų ilgiausios kelionės

maršrutą iš nurodyto miesto. Tekstiniame faile 'U3.txt' surašyti duomenys apie kelius tarp miestų. Pirmoje failo

eilutėje yra kelių skaičius n (1 ≤ n ≤ 100), antroje eilutėje miesto pavadinimas, iš kur prasideda kelionė.

Sekančiose n failo eilučių surašyta: pirmojo miesto pavadinimas, antrojo miesto pavadinimas, kelio tarp pirmojo

ir antrojo miesto ilgis kilometrais. Miesto pavadinimas ne ilgesnis kaip 15 simbolių.

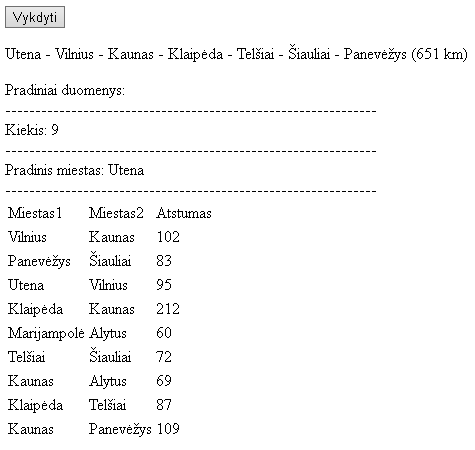
Išvykus iš vieno miesto galima nukeliauti į bet kurį kitą miestą. Tarp miestų gali būti daugiau kaip vienas

kelionės maršrutas. Kelionės metu tas pats miestas gali būti aplankytas tik vieną kartą. Maršrutas nebūtinai turi

apimti visus duotus miestus. Atspausdinkite ekrane ilgiausios kelionės maršrutą ir jo ilgį kaip parodyta

pavyzdyje.

## Grafinės vartotojo sąsajos schema



## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Button | Text | Vykdyti |
| Label1 | Text |  |
| Label2 | Text |  |

## Klasių diagrama

Marsrutas -> Marsrutai

## Programos vartotojo vadovas

Sukuriame failą pavadinimu U3.txt į jį įrašome pradinius duomenis (Pirmoj eilutėj maršrutų kiekis, Antroj eilutėj pradinis miestas, tolesnėse eilutėse rašomi maršrutai: pradinis miestas, galutinis miestas, atstumas tarp jų)). Eilutėje vienus duomenis nuo kitų skiria vienas tarpas. Užpildžius duomenų failus, galima įjungti programą. Atsidariusiame naršyklės lange matome mygtuką Vykdyti, jį nuspaudę įjungiame programą ir matome rezultatus.

## Programos tekstas

Marsrutas.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace L1

{

/// <summary>

/// Maršrutų klasė

/// </summary>

public class Marsrutas

{

public string Miestas1 { get; set; }

public string Miestas2 { get; set; }

public int Atstumas { get; set; }

public bool Pravaziuota = false;

/// <summary>

/// Konstruktorius

/// </summary>

/// <param name="miestas1">Pradinis miestas</param>

/// <param name="miestas2">Galutinis miestas</param>

/// <param name="atstumas">Atstumas tarp miestų</param>

public Marsrutas(string miestas1, string miestas2, int atstumas)

{

Miestas1 = miestas1;

Miestas2 = miestas2;

Atstumas = atstumas;

}

}

}

Marsrutai.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace L1

{

/// <summary>

/// Maršrutų konteinerinė klasė

/// </summary>

public class Marsrutai

{

private Marsrutas[] Marsrutas;

public int Kiekis { get; private set; }

/// <summary>

/// Konstruktorius

/// </summary>

/// <param name="kiekis">Objektų skaičius</param>

public Marsrutai(int kiekis)

{

Marsrutas = new Marsrutas[kiekis];

}

/// <summary>

/// Pridėjimas

/// </summary>

/// <param name="marsrutas">Maršruto objektas</param>

public void Add(Marsrutas marsrutas)

{

Marsrutas[Kiekis++] = marsrutas;

}

/// <summary>

/// Gavimas

/// </summary>

/// <param name="id">Maršruto id</param>

/// <returns>Grąžinamas maršrutas pagal nurodytą ID</returns>

public Marsrutas Get(int id)

{

return Marsrutas[id];

}

}

}

Puslapis.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Puslapis.aspx.cs" Inherits="L1.Puslapis" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Maršrutai</title>

</head>

<body>

<form id="form1" runat="server">

<asp:Button ID="Button1" runat="server" Text="Vykdyti" OnClick="Button1\_Click" />

<p>

<asp:Label ID="Label1" runat="server" Text=""></asp:Label>

</p>

Pradiniai duomenys:<br />

<asp:Label ID="Label2" runat="server" Text=""></asp:Label>

<asp:Table ID="Table1" runat="server">

</asp:Table>

</form>

</body>

</html>

Puslapis.aspx.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace L1

{

public partial class Puslapis : System.Web.UI.Page

{

private string irasymas = "U3rez.txt";

private int viso = 0;

protected void Page\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// Mygtuko paspaudimas

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void Button1\_Click(object sender, EventArgs e)

{

string aplankas = Path.Combine(HttpContext.Current.Request.PhysicalApplicationPath, "App\_Data");

Marsrutai marsrutai = new Marsrutai(100);

Skaitymas(aplankas + "\\" + "U3.txt", marsrutai, out string pMiestas);

List<string> pravaziuoti = new List<string>();

Label1.Text = pMiestas;

Vykdymas(marsrutai, pMiestas, pravaziuoti);

Label1.Text += $" ({viso} km)";

PradiniaiDuomenys("PradiniaiDuomenys.txt", marsrutai, pMiestas);

Lentele(marsrutai, pMiestas);

}

/// <summary>

/// Lentelės kūrimo klasė

/// </summary>

/// <param name="marsrutai">Maršrutų sąrašas</param>

/// <param name="pMiestas">Pradinis miestas</param>

void Lentele(Marsrutai marsrutai, string pMiestas)

{

string text = null;

text = ("--------------------------------------------------------------<br/>");

text += ($"Kiekis: {marsrutai.Kiekis}<br/>");

text += ("--------------------------------------------------------------<br/>");

text+=($"Pradinis miestas: {pMiestas}<br/>");

text+=("--------------------------------------------------------------<br/>");

Label2.Text = text;

TableRow row = new TableRow();

TableCell cell = new TableCell

{

Text = "Miestas1"

};

row.Cells.Add(cell);

TableCell cell2 = new TableCell

{

Text = "Miestas2"

};

row.Cells.Add(cell2);

TableCell cell3 = new TableCell

{

Text = "Atstumas"

};

row.Cells.Add(cell3);

Table1.Rows.Add(row);

for (int i = 0; i < marsrutai.Kiekis; i++)

{

TableRow duom = new TableRow();

TableCell m1 = new TableCell

{

Text = marsrutai.Get(i).Miestas1

};

duom.Cells.Add(m1);

TableCell m2 = new TableCell

{

Text = marsrutai.Get(i).Miestas2

};

duom.Cells.Add(m2);

TableCell a = new TableCell

{

Text = marsrutai.Get(i).Atstumas.ToString()

};

duom.Cells.Add(a);

Table1.Rows.Add(duom);

}

}

/// <summary>

/// Pradinių duomenų sudarymo metodas

/// </summary>

/// <param name="failas">Failas</param>

/// <param name="marsrutai">Maršrutų sąrašas</param>

/// <param name="pMiestas">Pradinis miestas</param>

void PradiniaiDuomenys(string failas, Marsrutai marsrutai, string pMiestas)

{

string aplankas = Path.Combine(HttpContext.Current.Request.PhysicalApplicationPath, "App\_Data");

using (StreamWriter sw = new StreamWriter(aplankas + "//" + failas))

{

sw.WriteLine("--------------------------------------------------------------");

sw.WriteLine($"Kiekis: {marsrutai.Kiekis}");

sw.WriteLine("--------------------------------------------------------------");

sw.WriteLine($"Pradinis miestas: {pMiestas}");

sw.WriteLine("--------------------------------------------------------------");

string header = string.Format("{0, -20} {1, -20} {2, -20}", "Miestas1", "Miestas2", "Atstumas");

sw.WriteLine(header);

for (int i = 0; i < marsrutai.Kiekis; i++)

{

string rasyti = string.Format("{0, -20} {1, -20} {2, -20}", marsrutai.Get(i).Miestas1, marsrutai.Get(i).Miestas2, marsrutai.Get(i).Atstumas);

sw.WriteLine(rasyti);

}

}

}

/// <summary>

/// Skaitymas iš failo

/// </summary>

/// <param name="failas">Skaitomas failas</param>

/// <param name="marsrutai">Maršrutų sąrašas</param>

/// <param name="pMiestas">Pradinis miestas</param>

void Skaitymas(string failas, Marsrutai marsrutai, out string pMiestas)

{

using (StreamReader reader = new StreamReader(failas))

{

int kiekis = int.Parse(reader.ReadLine());

pMiestas = reader.ReadLine();

for (int i = 0; i < kiekis; i++)

{

string[] duom = reader.ReadLine().Split(' ');

Marsrutas marsrutas = new Marsrutas(duom[0], duom[1], int.Parse(duom[2]));

marsrutai.Add(marsrutas);

}

}

}

/// <summary>

/// Programos vykdymo metodas

/// </summary>

/// <param name="marsrutai">Maršrutų sąrašas</param>

/// <param name="miestas">Dabartinis miestas</param>

void Vykdymas(Marsrutai marsrutai, string miestas, List<string> pravaziuoti)

{

int max = 0;

string lmiestas = null;

for(int i = 0; i < marsrutai.Kiekis; i++)

{

if (marsrutai.Get(i).Miestas1 == miestas)

{

if (!pravaziuoti.Contains(marsrutai.Get(i).Miestas2))

{

if (marsrutai.Get(i).Atstumas > max)

{

max = marsrutai.Get(i).Atstumas;

lmiestas = marsrutai.Get(i).Miestas2;

}

}

}

else if (marsrutai.Get(i).Miestas2 == miestas)

{

if (!pravaziuoti.Contains(marsrutai.Get(i).Miestas1))

{

if (marsrutai.Get(i).Atstumas > max)

{

max = marsrutai.Get(i).Atstumas;

lmiestas = marsrutai.Get(i).Miestas1;

}

}

}

}

if (lmiestas == null)

{

string aplankas = Path.Combine(HttpContext.Current.Request.PhysicalApplicationPath, "App\_Data");

using (StreamWriter sw = new StreamWriter(aplankas + "\\" +irasymas))

{

sw.WriteLine(Label1.Text);

}

return;

}

Label1.Text += " - " +lmiestas;

viso += max;

pravaziuoti.Add(lmiestas);

Vykdymas(marsrutai, lmiestas, pravaziuoti);

}

/// <summary>

/// Įrašymo funkcija

/// </summary>

void Irasymas()

{

string aplankas = Path.Combine(HttpContext.Current.Request.PhysicalApplicationPath, "App\_Data");

using (StreamWriter sw = new StreamWriter(aplankas + "\\" +irasymas))

{

sw.WriteLine(Label1.Text);

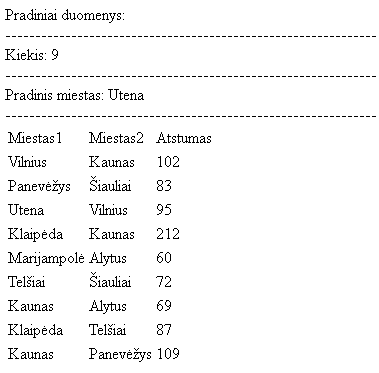
}

}

}

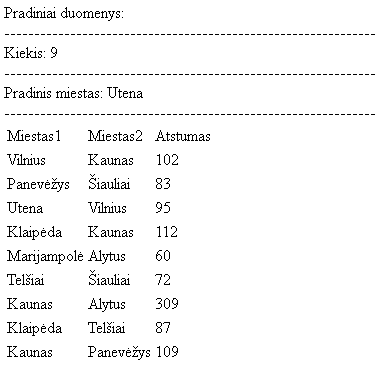
}

## Pradiniai duomenys ir rezultatai



Rezultatai:





Rezultatai:





Rezultatai:



## Dėstytojo pastabos

# Dinaminis atminties valdymas (L2)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Klasių diagrama

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

## Dėstytojo pastabos

# Bendrinės klasės ir sąsajos (L3)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Klasių diagrama

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

## Dėstytojo pastabos

# Kolekcijos ir išimčių valdymas (L4)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Klasių diagrama

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

## Dėstytojo pastabos

# Deklaratyvusis programavimas (L5)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
|  |  |  |
|  |  |  |
|  |  |  |

## Klasių diagrama

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

## Dėstytojo pastabos