KAUNO TECHNOLOGIJOS UNIVERSITETAS

OBJEKTINIS PROGRAMAVIMAS II (P175B123)

Laboratorinio darbo ataskaita

Atliko:

IFF-5/1 gr. studentas

Lukas Gužauskas

2016 m. balandžio 21 d.

Priėmė:

Docentas Giedrius Ziberkas

KAUNAS 2016

TURINYS

1. Rekursija 4

1.1. Darbo užduotis 4

1.2. Programos tekstas 4

1.3. Pradiniai duomenys ir rezultatai 9

1.4. Grafinės vartotojo sąsajos schema 10

1.5. Grafinės vartotojo sąsajos elementų pakeistos savybės 10

1.6. Klasių diagramos 11

1.7. Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus 11

2. Susietasis sąrašas 12

2.1. Darbo užduotis 12

2.2. Programos tekstas 12

2.3. Pradiniai duomenys ir rezultatai 21

2.4. Grafinės vartotojo sąsajos schema 25

2.5. Grafinės vartotojo sąsajos elementų pakeistos savybės 26

2.6. Klasių diagramos 26

2.7. Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus 27

3. Bendrinis susietasis sąrašas 28

3.1. Darbo užduotis 28

3.2. Programos tekstas 28

3.3. Pradiniai duomenys ir rezultatai 40

3.4. Grafinės vartotojo sąsajos schema 43

3.5. Grafinės vartotojo sąsajos elementų pakeistos savybės 43

3.6. Klasių diagramos 44

3.7. Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus 45

4. Bendrinės kolekcijos 46

4.1. Darbo užduotis 46

4.2. Programos tekstas 46

4.3. Pradiniai duomenys ir rezultatai 52

4.4. Grafinės vartotojo sąsajos schema 53

4.5. Grafinės vartotojo sąsajos elementų pakeistos savybės 53

4.6. Klasių diagramos 54

4.7. Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus 54

5. Deklaratyvusis programavimas 55

5.1. Darbo užduotis 55

5.2. Programos tekstas 55

5.3. Pradiniai duomenys ir rezultatai 59

5.4. Grafinės vartotojo sąsajos schema 60

5.5. Grafinės vartotojo sąsajos elementų pakeistos savybės 61

5.6. Klasių diagramos 61

5.7. Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus 61

# Rekursija

## Darbo užduotis

**U13 Skaičių magija.**

Duotas skaičių rinkinys, kurį sudaro 24 skaičiai. Šis rinkinys suformuotas tokiu būdu: sveikieji skaičiai nuo 1 iki 10 įtraukiami du kartus ir keturi skaičiai: 25, 50, 75, 100. Atsitiktiniu būdu iš jų išrenkami šeši skaičiai.

Atsitiktiniu būdu išrenkamas skaičius nuo 100 iki 999 (ribos įtraukiamos). Panaudodami kiekvieną iš išrinktų skaičių ne daugiau kaip vieną kartą ir sujungdami juos aritmetinėmis operacijomis: +, \*, -, / (galima dalinti tik tuomet, kai dalinasi be liekanos), gaukite išrinktą triženklį skaičių. Numatykite variantą, kad skaičius negali būti gautas. Sudarykite šio uždavinio sprendimui programą. Programa turi spręsti šį uždavinį ne daugiau kaip 5 sekundes.

Duomenys. Tekstiniame faile U3.txt užrašyti 7 skaičiai, pirmieji 6 skaičiai – tai atsitiktinai išrinkti duomenys iš duoto skaičių rinkinio, ir jie užrašyti pirmoje eilutėje. 7 - asis skaičius – tai atsitiktiniu būdu išrinktas rezultato skaičius, kurį reikia gauti, ir šis skaičius yra antroje eilutėje. Rezultatai. Ekrane eilutėmis spausdinkite atskirus veiksmus ir jų rezultatus.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

using System.Text;

**MagicSet.cs**

public class MagicSet

{

const int Cn = 100;

public int i;

public int[] Skaiciai { get; set;}

public int tikslas { get; set; }

public int[] Veiksmai { get; set; }

public bool Sprendinys { get; set; }

public MagicSet()

{

tikslas = 0;

Skaiciai = new int[Cn];

Veiksmai = new int[Cn];

}

}

**Ld1.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Ld1.aspx.cs" Inherits="Ld1" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Skaičių magija</title>

<style type="text/css">

.newStyle1 {

text-align: center;

}

</style>

</head>

<body>

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

<div style="text-align: center">

<asp:Label ID="Label1" runat="server" Text="Skaičių magija" Font-Bold="True" Font-Names="Times New Roman" Font-Size="XX-Large" ForeColor="#3399FF" CssClass="newStyle1"></asp:Label>

<br />

<asp:Label ID="Label4" runat="server" Font-Bold="True" Font-Size="X-Large" ForeColor="#6699FF" Text="Rezultatai"></asp:Label>

<br />

<asp:Label ID="Label5" runat="server" Font-Bold="True" Font-Size="X-Large" Text="Skaičiai"></asp:Label>

<br />

<asp:Label ID="Label3" runat="server" Text="Galutinio tikslas" Font-Bold="True" Font-Size="X-Large"></asp:Label>

<br />

<asp:TextBox ID="TextBox1" runat="server" Height="172px" TextMode="MultiLine" Width="257px" Font-Size="X-Large" ForeColor="#336699" Enabled="False" EnableTheming="True"></asp:TextBox>

<br />

<asp:Button ID="Button1" runat="server" Text="Skaičiuoti!" OnClick="Button1\_Click" BackColor="#6699FF" BorderStyle="Dashed" Font-Size="X-Large" ForeColor="Black" />

</div>

</form>

</body>

</html>

**Ld1.aspx.cs**

public partial class Ld1 : System.Web.UI.Page

{

const string fv = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld1\Duomenys.txt";

const string fr = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld1\Rezultatai.txt";

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

{

int kiek;

string line = "";

MagicSet Ms = new MagicSet();

Skaityti(fv, out kiek, Ms);

Label4.Visible = false;

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

line = line + " " + Ms.Skaiciai[i].ToString();

Label5.Text = "Skaičiai" + line;

Label3.Text = "galutinis tikslas: " + Ms.tikslas.ToString();

TextBox1.Visible = false;

}

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

{

int kiek;

MagicSet Ms = new MagicSet();

Skaityti(fv, out kiek, Ms);

Label4.Visible = true;

TextBox1.Visible = true;

if (File.Exists(fr))

File.Delete(fr);

Kombinancija(ref Ms, 0, kiek, fr);

if (Ms.Sprendinys)

{

Label4.Text = "Rezultatai";

Button1.Visible = false;

}

else

{

Label4.Text = "Nėra sprendinio";

TextBox1.Visible = false;

Button1.Visible = false;

}

}

/// <summary>

/// Skaitomi duomenys iš failo

/// </summary>

/// <param name="fv">duomenų failo vardas</param>

/// <param name="kiek">masyvo dydis</param>

/// <param name="Sk">Masyvo vardas</param>

/// <param name="rez">galutinis tikslas</param>

/// <returns>iš failo nuskaitytų skaičių masyvą</returns>

static void Skaityti(string fv, out int kiek, MagicSet Ms)

{

const int Cn = 100;

int[] Sk = new int[Cn];

using (StreamReader reader = new StreamReader(fv))

{

string line;

line = reader.ReadLine();

string[] parts = line.Split(' ');

kiek = parts.Length;

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

{

Sk[i] = int.Parse(parts[i]);

}

line = reader.ReadLine();

Ms.tikslas = int.Parse(line);

}

Ms.Skaiciai = Sk;

}

/// <summary>

/// surandami aritmetiniai veiksmai tikslui pasiekti

/// </summary>

/// <param name="ms">Objekto vardas</param>

/// <param name="i">Skaičių, su kuriais atliekami veiksmai, numeris masyve</param>

/// <param name="kiek">Magijos skaičius</param>

/// <param name="suma">Einamoji suma patikrinti</param>

/// <param name="fr">Spausdinamo failo vardas</param>

void Magija(MagicSet ms, int i, int kiek, int suma, string fr)

{

int[] Sk = ms.Skaiciai;

if (i != kiek)

{

int temp;

string zenklas;

i++;

for (int j = 0; j < 4; j++)

{

ms.Veiksmai[i] = j;

temp = suma;

suma = Veiksmas(suma, Sk[i], j, out zenklas);

if (zenklas != "!")

{

if (suma == ms.tikslas)

{

int rez = ms.Skaiciai[0];

Print(ms, 0, i, rez, fr);

ms.Sprendinys = true;

return;

}

Magija(ms, i, kiek, suma, fr);

if (ms.Sprendinys) { return; }

suma = temp;

}

}

}

}

/// <summary>

/// Išspausdimas

/// </summary>

/// <param name="ms">Objekto vardas</param>

/// <param name="j">Spausdinamų aritmetinių veiksmų eilutės numeris</param>

/// <param name="i">Visų spausdinamų eilučių skaičius</param>

/// <param name="suma">Tarpinės sumos</param>

/// <param name="fr">Spausdinamo failo vardas</param>

void Print(MagicSet ms, int j, int i, int suma, string fr)

{

if (j != i)

{

string zenklas;

int tarp = suma;

suma = Veiksmas(suma, ms.Skaiciai[j + 1], ms.Veiksmai[j + 1], out zenklas);

using (var frr = File.AppendText(fr))

{

frr.WriteLine(tarp + " " + zenklas + " " + ms.Skaiciai[j + 1] + " = " + suma);

}

TextBox1.Text = TextBox1.Text + "\r\n "

+ tarp.ToString() + " " + zenklas + " " + ms.Skaiciai[j + 1].ToString() + " = " + suma.ToString();

j++;

Print(ms, j, i, suma, fr);

}

}

/// <summary>

/// Atliekamos aritmetinės operacijos ir grąžina operacijos rezultatą: +, \*, -, /

/// </summary>

/// <param name="suma">operacijos rezultatas</param>

/// <param name="demuo">operacijos dėmuo</param>

/// <param name="i">operacijos numeris</param>

/// <param name="zenklas">grąžinama aritmetinės operacijos</param>

/// <returns>Gražina aritmetinės operacijos rezultatas</returns>

int Veiksmas(int suma, int demuo, int i, out string zenklas)

{

if (i == 0)

{

zenklas = "+";

return (suma + demuo);

}

else

if (i == 1)

{

zenklas = "-";

return (suma - demuo);

}

else

if (i == 2)

{

zenklas = "\*";

return (suma \* demuo);

}

else

{

int dalyba = suma / demuo;

if (dalyba \* demuo == suma)

{

zenklas = "/";

return dalyba;

}

else

{

zenklas = "!";

return suma;

}

}

}

/// <summary>

/// Skaičių kombinancija po kiekvieno skaičiaus sukeitimo

/// </summary>

/// <param name="ms">Objekto vardas</param>

/// <param name="i">Skaičių, su kuriais atliekami veiksmai, numeris masyve</param>

/// <param name="kiek">Magijos skaičius</param>

/// <param name="fr">Spausdinamo failo vardas</param>

void Kombinancija(ref MagicSet ms, int i, int kiek, string fr)

{

if (i == kiek)

{

int suma = ms.Skaiciai[0];

Magija(ms, 0, kiek - 1, suma, fr);

}

for (int j = i; j < kiek; j++)

{

Swap(ref ms.Skaiciai[i], ref ms.Skaiciai[j]);

Kombinancija(ref ms, i + 1, kiek, fr);

Swap(ref ms.Skaiciai[i], ref ms.Skaiciai[j]);

}

}

/// <summary>

/// Skaičių sukeitimas

/// </summary>

/// <param name="x">Skaičius iš X</param>

/// <param name="y">X skaičius į Y</param>

void Swap(ref int x, ref int y)

{

int temp;

temp = x;

x = y;

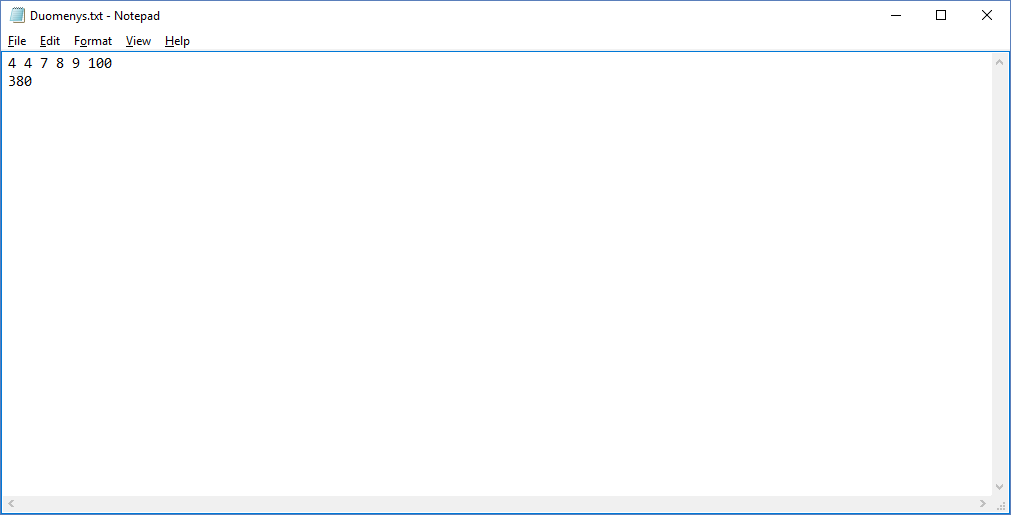
y = temp;

}

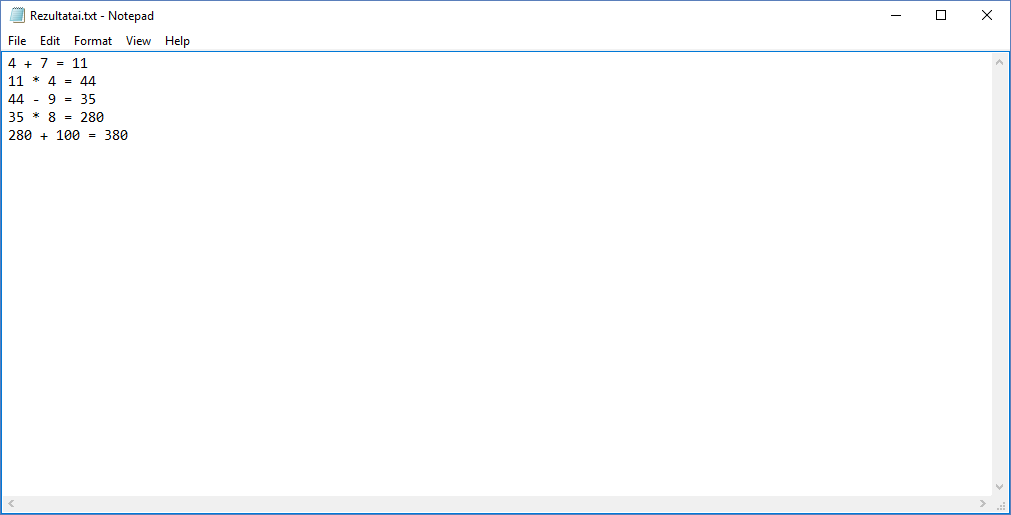
}

## Pradiniai duomenys ir rezultatai

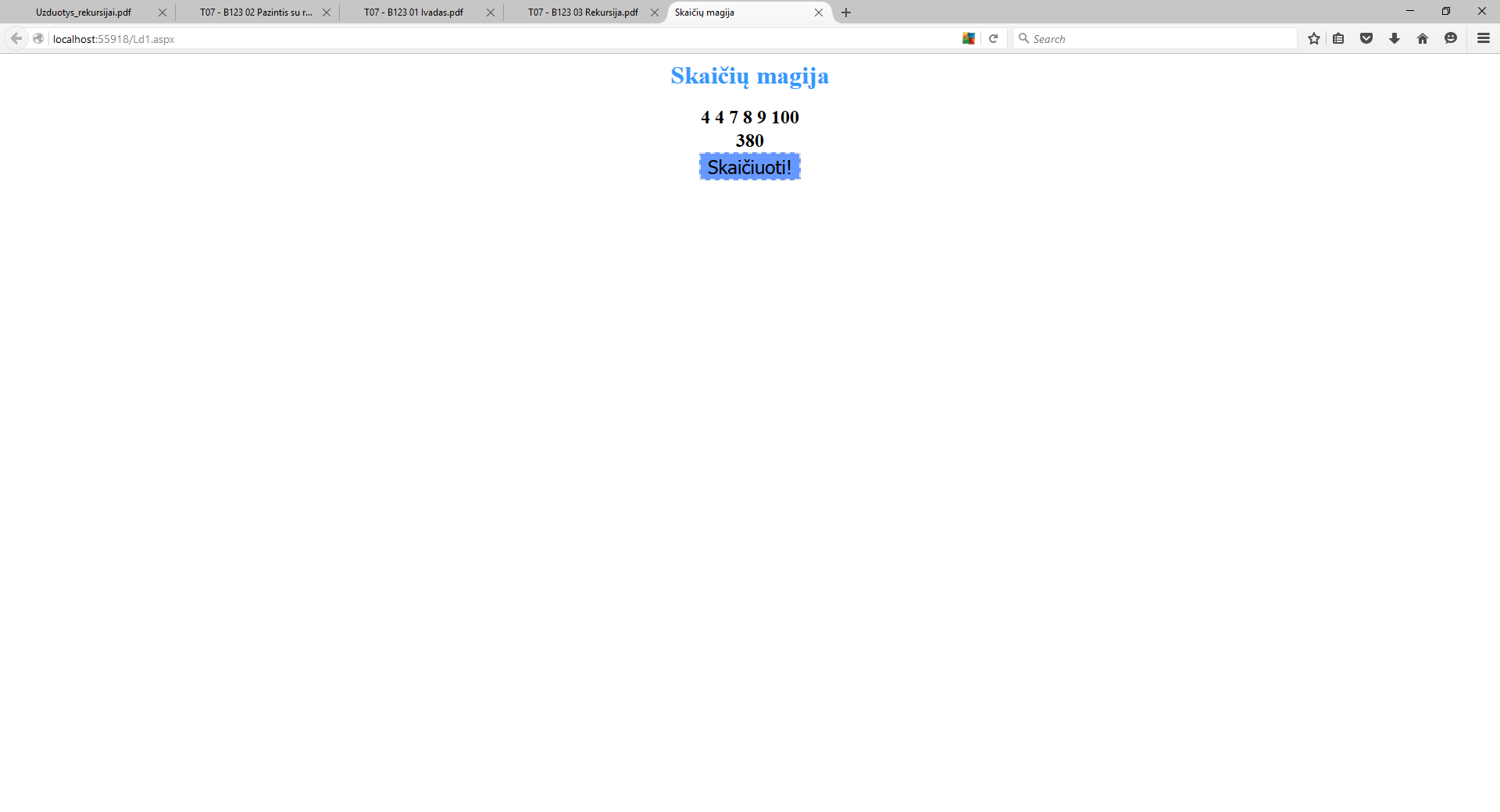
Pradiniai duomenys:

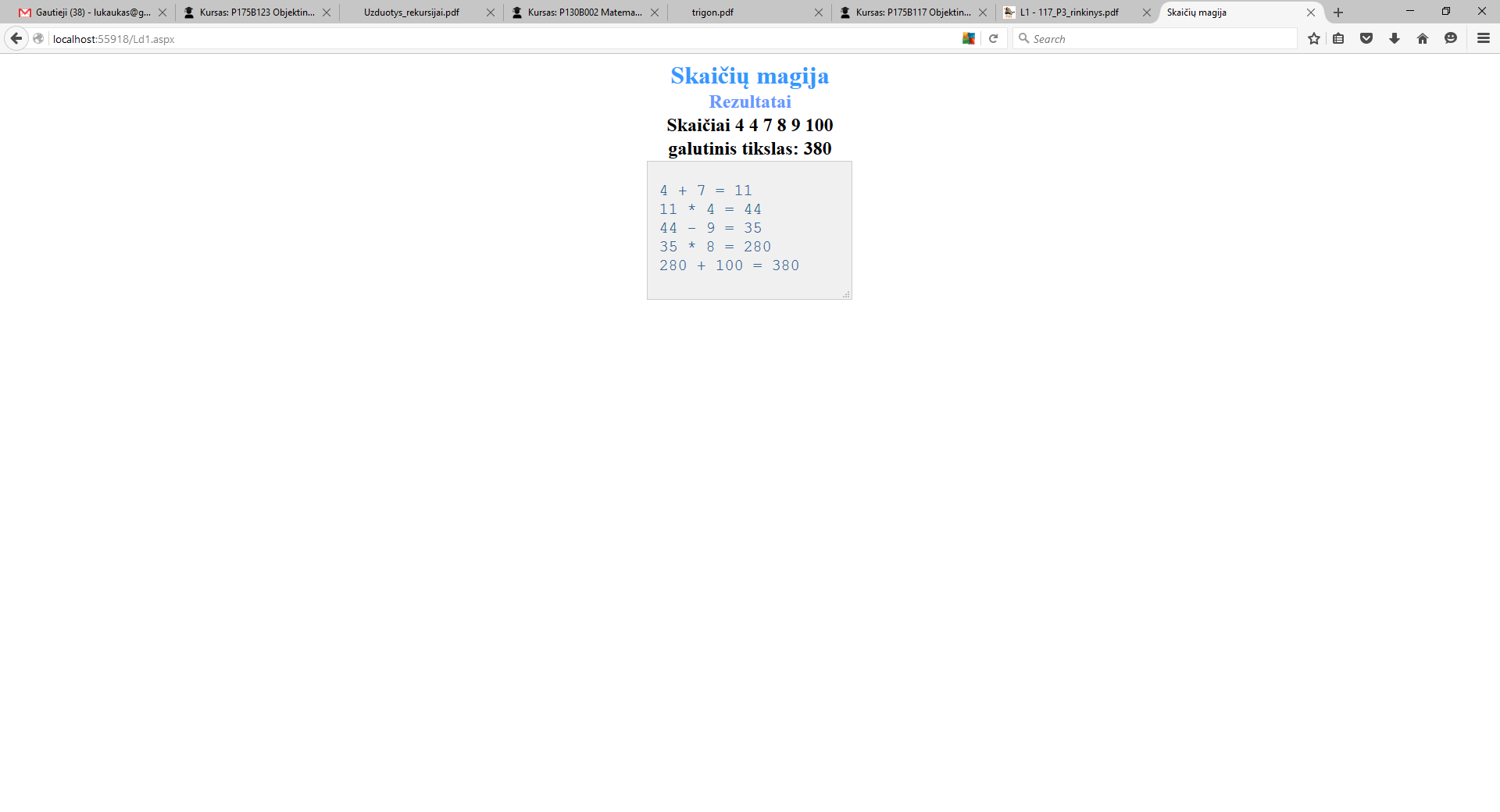


Rezultatai:



## Grafinės vartotojo sąsajos schema





## Grafinės vartotojo sąsajos elementų pakeistos savybės

|  |  |  |  |
| --- | --- | --- | --- |
| Komponentas | ID | Text | OnClick |
| Label | Label1 | Skaičių magija |  |
| Label | Label3 | Galutinio tikslas |  |
| Label | Label4 | Rezultatai |  |
| Label | Label5 | Skaičiai |  |
| Button | Button1 | Skaičiuoti! | Button\_Click |

## Klasių diagramos

|  |
| --- |
| MagicSet |
| + Skaiciai[] : integer  + Tikslas : integer  + Veiksmai[] : integer  + Sprendinys : boolean |
| + MagicSet() {query} |

## Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus

Programos darbiniame aplanke sukuriami tekstiniai duomenų failai Duomenys.txt. Tekstiniame faile Duomenys.txt įrašomas skaičių rinkinys. Tekstiniame faile Duomenys.txt įrašomi 7 skaičiai, pirmieji 6 skaičiai – tai atsitiktinai išrinkti duomenys iš duoto skaičių rinkinio, ir jie užrašyti pirmoje eilutėje. Galimų skaičių rinkinį sudaro skaičiai nuo 1 iki 10, išrinkti skaičiai gali kartotis daugiausiai 2 kartus, ir skaičiai 25, 50, 75, 100.

7 -asis skaičius – tai atsitiktiniu būdu iš intervalo 100-999 parenkamas rezultato skaičius, kurį reikia gauti, ir šis skaičius yra antroje eilutėje.

Užpildžius duomenų failą ir išsaugojus, galima įjungti programą. Įjungę programą pirmiausiai užkraunami duomenys iš failo ir monitoriuje matomi vienoje eilutėje 6 skaičiai, iš kurių aritmetinių veiksmų pagalba turi būti gaunamas 7 skaičius, kuris matomas sekančioje eilutėje. Žemiau mygtukas „Skaičiuoti“, kurį paspaudus atlieka kompiuteris skaičiavimus ir gauti rezultatai išvedami monitoriuje eilutėmis atskiri veiksmai ir jų rezultatai tekstiniame lange. Taip pat rezultatai įvedami tekstiniame faile Rezultatai.txt. Programai atlikus savo darbą, galima išeiti iš programos.

# Susietasis sąrašas

## Darbo užduotis

**U13 Žaidėjai.**

Krepšininkai pagal žaidimo poziciją komandoje skirstomi į 3 pozicijas: centrą, puolėją, gynėją. Krepšinio komandos tarpusavy žaidžia rungtynes. Vienos komandos laimi, o kitos pralaimi, lygiųjų nebūna. Sudarykite kiekvienos pozicijos geriausių žaidėjų sąrašus (taškai, klaidos).

Suraskite, kurios komandos žaidėjai pelnė daugiausiai taškų ir atspausdinkite šios komandos užimamą vietą.

Duomenys:

* Tekstiniame faile U13a.txt duota informacija apie krepšininkus: komanda, vardas, pavardė, gimimo metai, ūgis, žaidimo pozicija, įmestų taškų skaičius, padarytų klaidų skaičius.
* Tekstiniame faile U13b.txt duota informacija apie komandas: komanda, žaistų rungtynių skaičius, laimėtų rungtynių skaičius.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka. Iš žaidėjų sąrašo pašalinkite nurodytos komandos (įvedama klaviatūra) žaidėjus.

## Programos tekstas

**Krepšininkas.cs**

public class Krepšininkas

{

public string team { get; set; }

public string var { get; set; }

public string pav { get; set; }

public int gim { get; set; }

public int ugis { get; set; }

public string poz { get; set; }

public int taskas { get; set; }

public int klaida { get; set; }

public Krepšininkas() { }

public Krepšininkas(string team1, string vardas, string pavarde, int gimi, int ugiss, string pozicija, int point, int klaid)

{

this.team = team1;

this.pav = pavarde;

this.var = vardas;

this.gim = gimi;

this.ugis = ugiss;

this.poz = pozicija;

this.taskas = point;

this.klaida = klaid;

}

public override string ToString()

{

string eilute;

eilute = string.Format("{0,-20}{1,-12}{2,-18}{3,-12}{4,10}{5, -9}{6,10}{7,5}", team, var, pav, gim, ugis, poz, taskas, klaida);

return eilute;

}

static public bool operator >(Krepšininkas pirmas, Krepšininkas antras)

{

int poz = String.Compare(pirmas.pav, antras.pav, StringComparison.CurrentCulture);

return poz > 0;

}

static public bool operator <(Krepšininkas pirmas, Krepšininkas antras)

{

int poz = String.Compare(pirmas.pav, antras.pav, StringComparison.CurrentCulture);

return poz < 0;

}

}

**Komanda.cs**

public class Komanda

{

public string team { get; set; }

public int rung { get; set; }

public int win { get; set; }

public Komanda() { }

public Komanda(string komanda, int rungtynes, int laime)

{

this.team = komanda;

this.rung = rungtynes;

this.win = laime;

}

public override string ToString()

{

string eilute = string.Format("{0,-20}{1,7}{2,30}", team, rung, win);

return eilute;

}

static public bool operator >(Komanda pirmas, Komanda antras)

{

return pirmas.win > antras.win;

}

static public bool operator <(Komanda pirmas, Komanda antras)

{

return pirmas.win < antras.win;

}

}

**Mazgas.cs**

public sealed class Mazgas

{

public Krepšininkas Duom { get; set; }

public Mazgas Kitas { get; set; }

public Mazgas(Krepšininkas reiksme, Mazgas adr)

{

Duom = reiksme;

Kitas = adr;

}

}

**MazgasKom.cs**

public sealed class MazgasKom

{

public Komanda Duom { get; set; }

public MazgasKom Kitas { get; set; }

public MazgasKom(Komanda reiksme, MazgasKom adr)

{

Duom = reiksme;

Kitas = adr;

}

}

**Sąrašas.cs**

public sealed class Sąrašas

{

private Mazgas pr;

private Mazgas pb;

private Mazgas d;

public Sąrašas()

{

pr = null;

pb = null;

d = null;

}

public void DetiDuomenisA(Krepšininkas naujas)

{

pr = new Mazgas(naujas, pr);

}

public void Pradzia() { d = pr; }

public void Kitas() { d = d.Kitas; }

public bool Yra() { return d != null; }

public Krepšininkas ImtisDuomenis() { return d.Duom; }

public void Rikiuoti()

{

for (Mazgas d1 = pr; d1 != null; d1 = d1.Kitas)

{

Mazgas maxv = d1;

for (Mazgas d2 = d1; d2 != null; d2 = d2.Kitas)

if (d2.Duom < maxv.Duom)

maxv = d2;

Krepšininkas Kre = d1.Duom;

d1.Duom = maxv.Duom;

maxv.Duom = Kre;

}

}

/// <summary>

/// Surasta ir pašalinimas, kai įveskia nurodytos komandos žaidėjus.

/// </summary>

/// <param name="nickPlayer">Įvesti vardą</param>

/// <param name="lastPlayer">Įvesti pavardę</param>

/// <param name="available">Taip ar ne</param>

public void Pasalinti(string team, ref bool available)

{

Mazgas dd = pr;

while (dd != null && dd.Duom.team != team)

{

dd = dd.Kitas;

}

if (dd != null) { available = true; }

Mazgas s = dd;

while (s != null && s.Duom.team == team)

{

s = s.Kitas;

}

if (s == null)

{

Mazgas pb = dd;

for (Mazgas dt = pr; dt != null; dt = dt.Kitas)

{

if (dt.Kitas == pb)

{

dt.Kitas = null;

}

}

}

else

{

dd.Duom = s.Duom;

dd.Kitas = s.Kitas;

s = null;

Pasalinti(team, ref available);

}

}

}

**KitaSar.cs**

class KitasSar

{

private MazgasKom prk;

private MazgasKom pbk;

private MazgasKom dk;

public KitasSar()

{

prk = null;

pbk = null;

dk = null;

}

public void DetiDuomenisK(Komanda nauja)

{

prk = new MazgasKom(nauja, prk);

}

public void PradziaK() { dk = prk; }

public void KitasK() { dk = dk.Kitas; }

public bool YraK() { return dk != null; }

public Komanda ImtisDuomenisK() { return dk.Duom; }

public void Rikiuoti()

{

for (MazgasKom d1 = prk; d1 != null; d1 = d1.Kitas)

{

MazgasKom maxv = d1;

for (MazgasKom d2 = d1; d2 != null; d2 = d2.Kitas)

if (d2.Duom > maxv.Duom)

maxv = d2;

Komanda Kom = d1.Duom;

d1.Duom = maxv.Duom;

maxv.Duom = Kom;

}

}

}

**LD2.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LD2.aspx.cs" Inherits="LD2" %>

<!DOCTYPE html>

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

<head runat="server">

<title>LD2</title>

<style type="text/css">

html, body

{

background-image: url('7018311-3d-black-background.jpg');

}

.newTable1 {

height: 25%;

width: 38%;

}

.newTable2 {

height: 30%;

width: 60%;

}

.newButton1 {

height:5%;

width: 7%;

position: absolute;

}

</style>

</head>

<body>

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

<div>

<p>

<asp:Label ID="Label1" runat="server" Text="Grandų mūšis" Style="left: 39%; position: absolute; top: 15px; width: 201px;" Font-Bold="True" Font-Size="XX-Large" ForeColor="White" ></asp:Label>

</p>

<asp:Label ID="Label2" runat="server" Style="top: 10%; position: absolute;" Text="Pradiniai duomenys U13a.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Style="top: 15%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:TextBox ID="TextBox2" runat="server" Style="top: 15%; left: 55%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:Label ID="Label3" runat="server" Style="top: 10%; left: 55%; position: absolute;" Text="Pradiniai duomenys U13b.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:Button ID="Button1" runat="server" Style="top: 20%; left: 40%;" Text="Spausdinti" OnClick="Button1\_Click" CssClass="newButton1"/>

<asp:Label ID="Label4" runat="server" Style="top: 45%; left: 20%; position: absolute;" Text="Rezultatai:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox3" runat="server" Style="top: 50%; left: 20%; position: absolute;" TextMode="MultiLine" CssClass="newTable2" Enabled="False"></asp:TextBox>

<asp:Label ID="Label5" runat="server" Style="top: 85%; left: 30%; position: absolute" Text="Įveskite žodžius, bet kuris pašalintas komandos žaidėjus" Font-Bold="True" ForeColor="White" Font-Size="X-Large"></asp:Label>

<asp:TextBox ID="TextBox4" runat="server" Style="top: 90%; left: 40%; height: 30px; width: 200px; position: absolute;"></asp:TextBox>

<asp:Button ID="Button2" runat="server" Style="top: 95%; left: 40%; position:absolute;" Text="OK" CssClass="newButton1" OnClick="Button2\_Click"/>

</div>

</form>

</body>

</html>

**LD2.aspx.cs**

public partial class LD2 : System.Web.UI.Page

{

const string fv1 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld2\U13a.txt";

const string fv2 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld2\U13b.txt";

const string fr = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld2\Rezultatai.txt";

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

{

TextBox1.Text = File.ReadAllText(fv1, Encoding.GetEncoding(1257));

TextBox2.Text = File.ReadAllText(fv2, Encoding.GetEncoding(1257));

Label4.Visible = false;

TextBox3.Visible = false;

Label5.Visible = false;

TextBox4.Visible = false;

Button2.Visible = false;

}

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

{

File.WriteAllText(fr, string.Empty);

Sąrašas A = new Sąrašas();

var K = new KitasSar();

var gynejasTaskas = new Sąrašas();

var puolejasTaskas = new Sąrašas();

var centrasTaskas = new Sąrašas();

var gynejasKlaida = new Sąrašas();

var puolejasKlaida = new Sąrašas();

var centrasKlaida = new Sąrašas();

Label4.Visible = true;

TextBox3.Visible = true;

Label5.Visible = true;

TextBox4.Visible = true;

Button2.Visible = true;

Button1.Visible = false;

IvestiA(fv1, A);

Spausdinti(fr, A, "Atvirkštinis žaidėjų sąrašas");

IvestiK(fv2, K);

SpausdintiK(fr, K, "Turyno lentelė");

Taskas(A, K, 2, ref gynejasTaskas, ref puolejasTaskas, ref centrasTaskas);

Spausdinti(fr, gynejasTaskas, "Daugiausiai taškų pelnęs gynėjas:");

Spausdinti(fr, puolejasTaskas, "Daugiausiai taškų pelnęs puolėjas:");

Spausdinti(fr, centrasTaskas, "Daugiausiai taškų pelnęs centras:");

Klaida(A, K, 2, ref gynejasKlaida, ref puolejasKlaida, ref centrasKlaida);

Spausdinti(fr, gynejasKlaida, "Daugiausiai klaidų padaręs gynėjas:");

Spausdinti(fr, puolejasKlaida, "Daugiausiai klaidų padaręs puolėjas:");

Spausdinti(fr, centrasKlaida, "Daugiausiai klaidų padaręs centras:");

K.Rikiuoti();

KomandosPoint(A, K, fr);

A.Rikiuoti();

Spausdinti(fr, A, "Rikiuotas krepšininkų sąrašas");

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Sąrašas A = new Sąrašas();

var B = new KitasSar();

Label4.Visible = true;

TextBox3.Visible = true;

IvestiA(fv1, A);

IvestiK(fv2, B);

string team = TextBox4.Text;

if (team != "")

{

bool available = false;

A.Pasalinti(team, ref available);

if (available)

{

A.Rikiuoti();

Spausdinti(fr, A, "Sąrašas po žaidėjo pašalinimo:");

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

else

{

Label5.Visible = true;

Label5.Text = "Tokio žaidėjo nėra sąraše !!!";

}

}

else

{

Label5.Visible = true;

Label5.Text = "Jūs neįveskite nė žodžio !!!";

}

}

/// <summary>

/// Skaitomi krepšininkų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

static void IvestiA(string fv, Sąrašas A)

{

string team, pav, vard, poz;

int gim, ugis, point, kl;

string line;

using (StreamReader reader = new StreamReader(fv, Encoding.GetEncoding(1257)))

while ((line = reader.ReadLine()) != null)

{

var Split = line.Split(';');

team = Split[0];

vard = Split[1];

pav = Split[2];

gim = int.Parse(Split[3]);

ugis = int.Parse(Split[4]);

poz = Split[5];

point = int.Parse(Split[6]);

kl = int.Parse(Split[7]);

var kre = new Krepšininkas(team, vard, pav, gim, ugis, poz, point, kl);

A.DetiDuomenisA(kre);

}

}

/// <summary>

/// Skaitomi komandos krepšininkų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

static void IvestiK(string fv, KitasSar A)

{

string team;

int rung, win;

string line;

using (StreamReader reader = new StreamReader(fv, Encoding.GetEncoding(1257)))

while ((line = reader.ReadLine()) != null)

{

string[] Split = line.Split(';');

team = Split[0];

rung = int.Parse(Split[1]);

win = int.Parse(Split[2]);

var kom = new Komanda(team, rung, win);

A.DetiDuomenisK(kom);

}

}

/// <summary>

/// Spausdinami krepšinikų duomenys į failą

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="eil">Paraštė</param>

static void Spausdinti(string fv, Sąrašas A, string eil)

{

using (var frr = new StreamWriter(fv, true))

{

string bruk = new string('-', 105);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Komanda Vardas Pavardė Gimimo metai Ūgis(cm) Pozicija Taškai Klaidos");

frr.WriteLine(bruk);

for (A.Pradzia(); A.Yra(); A.Kitas())

frr.WriteLine(A.ImtisDuomenis());

frr.WriteLine(bruk);

frr.WriteLine();

}

}

/// <summary>

/// Spausdinami komandos krepšinikų duomenys į failą

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="eil"></param>

static void SpausdintiK(string fv, KitasSar A, string eil)

{

using (var frr = new StreamWriter(fv, true))

{

string bruk = new string('-', 70);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Komanda Žaistų rungtynių skaičius Laimėtų rungtynių skaičius");

frr.WriteLine(bruk);

for (A.PradziaK(); A.YraK(); A.KitasK())

frr.WriteLine(A.ImtisDuomenisK());

frr.WriteLine(bruk);

frr.WriteLine();

}

}

/// <summary>

/// Paieška ir saraso formavimas, kuris kiekvienoje pozicijoje žaidėjas daugiausiai pelnė taškų.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="B">Konteinerinės komandos vardas</param>

/// <param name="i">Masyvo indeksas</param>

/// <param name="G">Konteinerinės gynejo vardas</param>

/// <param name="P">Konteinerinės puolejo vardas</param>

/// <param name="C">Konteinerinės centro vardas</param>

static void Taskas(Sąrašas A, KitasSar B, int i, ref Sąrašas G, ref Sąrašas P, ref Sąrašas C)

{

Krepšininkas kre = new Krepšininkas();

string[] pozicija = { " gynėjas", " puolėjas", " centras" };

if (i == -1)

return;

for (B.PradziaK(); B.YraK(); B.KitasK())

{

int suma = 0;

for (A.Pradzia(); A.Yra(); A.Kitas())

if (A.ImtisDuomenis().poz == pozicija[i] && A.ImtisDuomenis().team == B.ImtisDuomenisK().team)

if (A.ImtisDuomenis().taskas > suma)

{

suma = A.ImtisDuomenis().taskas;

kre = A.ImtisDuomenis();

}

if (pozicija[i] == pozicija[2])

C.DetiDuomenisA(kre);

else

if (pozicija[i] == pozicija[1])

P.DetiDuomenisA(kre);

else

G.DetiDuomenisA(kre);

}

Taskas(A, B, i - 1, ref G, ref P, ref C);

}

/// <summary>

/// Paieška ir suformuoti sarasas, kuris kiekvienoje pozicijoje žaidėjas daugiausiai padaryti klaidu.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="B">Konteinerinės komandos vardas</param>

/// <param name="i">Masyvo indeksas</param>

/// <param name="G">Konteinerinės gynejo vardas</param>

/// <param name="P">Konteinerinės puolejo vardas</param>

/// <param name="C">Konteinerinės centro vardas</param>

static void Klaida(Sąrašas A, KitasSar B, int i, ref Sąrašas G, ref Sąrašas P, ref Sąrašas C)

{

Krepšininkas kre = new Krepšininkas();

string[] pozicija = { " gynėjas", " puolėjas", " centras" };

if (i == -1)

return;

for (B.PradziaK(); B.YraK(); B.KitasK())

{

int suma = 0;

for (A.Pradzia(); A.Yra(); A.Kitas())

if (A.ImtisDuomenis().poz == pozicija[i] && A.ImtisDuomenis().team == B.ImtisDuomenisK().team)

if (A.ImtisDuomenis().klaida > suma)

{

suma = A.ImtisDuomenis().klaida;

kre = A.ImtisDuomenis();

}

if (pozicija[i] == pozicija[2])

C.DetiDuomenisA(kre);

else

if (pozicija[i] == pozicija[1])

P.DetiDuomenisA(kre);

else

G.DetiDuomenisA(kre);

}

Klaida(A, B, i - 1, ref G, ref P, ref C);

}

/// <summary>

/// Paieška ir spausdinimas, kurios komandos žaidėjai pelnė daugiausiai taškų ir atspausdinama šios komandos užimta vieta.

/// </summary>

/// <param name="A">Krepšininko konteinerinės vardas</param>

/// <param name="B">Komandos konteinerinės vardas</param>

/// <param name="fv">Duomenų failo vardas</param>

static void KomandosPoint(Sąrašas A, KitasSar B, string fv)

{

int max = 0;

int k = 0;

string team = "";

for (B.PradziaK(); B.YraK(); B.KitasK())

{

string komanda = B.ImtisDuomenisK().team;

int win = B.ImtisDuomenisK().win;

Max(A, komanda, win, ref max, ref team);

}

for (B.PradziaK(); B.YraK(); B.KitasK())

{

k++;

using (var frr = new StreamWriter(fv, true))

{

if (B.ImtisDuomenisK().team == team)

{

frr.WriteLine("Komanda, kurios žaidėjai pelnė daugiausiai taškų:");

frr.WriteLine("{0, -20}{1,2} taškų/ai ir užimta{2,2} vieta.", team, max, k);

frr.WriteLine();

}

}

}

}

/// <summary>

/// Suskaičiuoja ir grąžina, kurios komandos žaidėjai pelnė daugiausiai taškų ir palyginima kas daugiausiai pelnė taškų.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="komanda">Komandos pavadinimas</param>

/// <param name="win">Komandos laimeti</param>

/// <param name="max">Daugiausiai taškų</param>

/// <param name="team">Komandos pavadinimas, kuris daugiausiai taškų</param>

static void Max(Sąrašas A, string komanda, int win, ref int max, ref string team)

{

int suma = 0;

int win1 = 0;

for (A.Pradzia(); A.Yra(); A.Kitas())

{

if (komanda == A.ImtisDuomenis().team)

suma = suma + A.ImtisDuomenis().taskas;

}

if (suma > max)

{

team = komanda;

max = suma;

win1 = win;

}

else

if (suma == max)

if (win > win1)

{

win1 = win;

team = komanda;

max = suma;

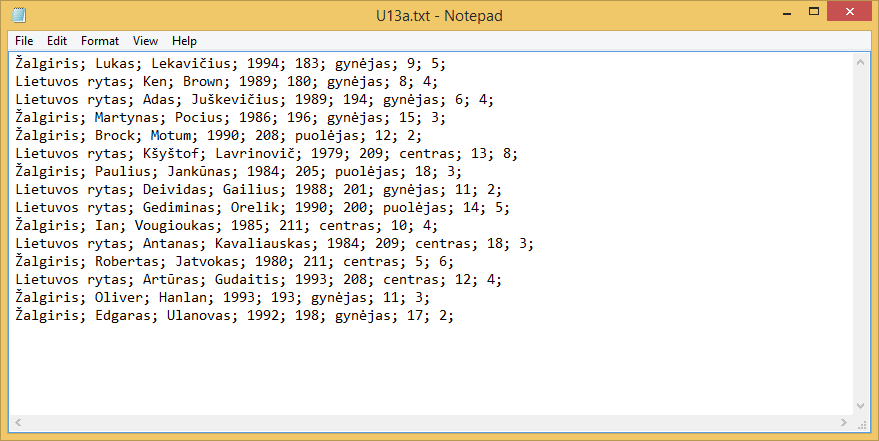
}

}

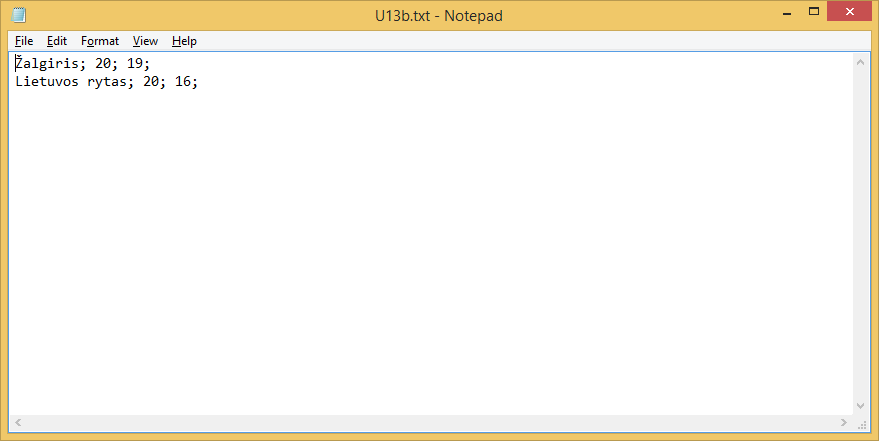
}

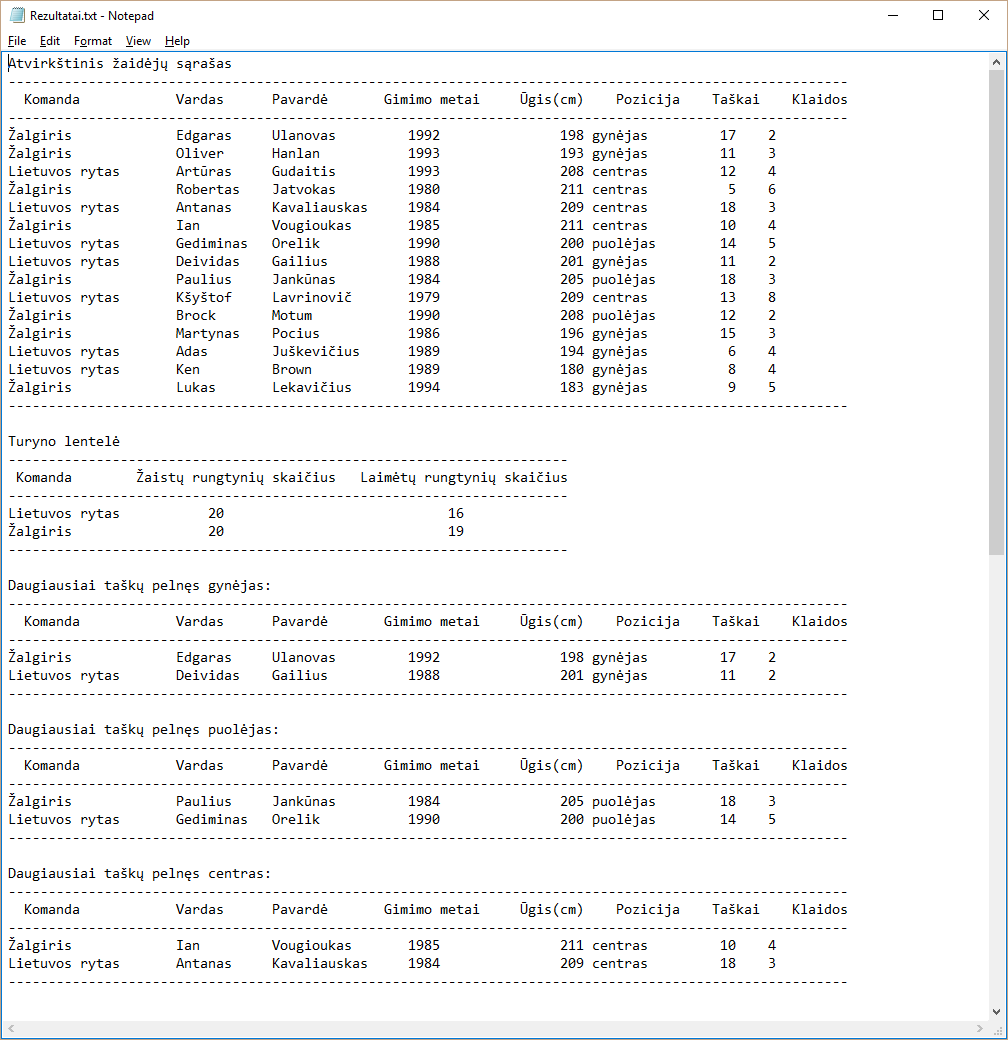
## Pradiniai duomenys ir rezultatai

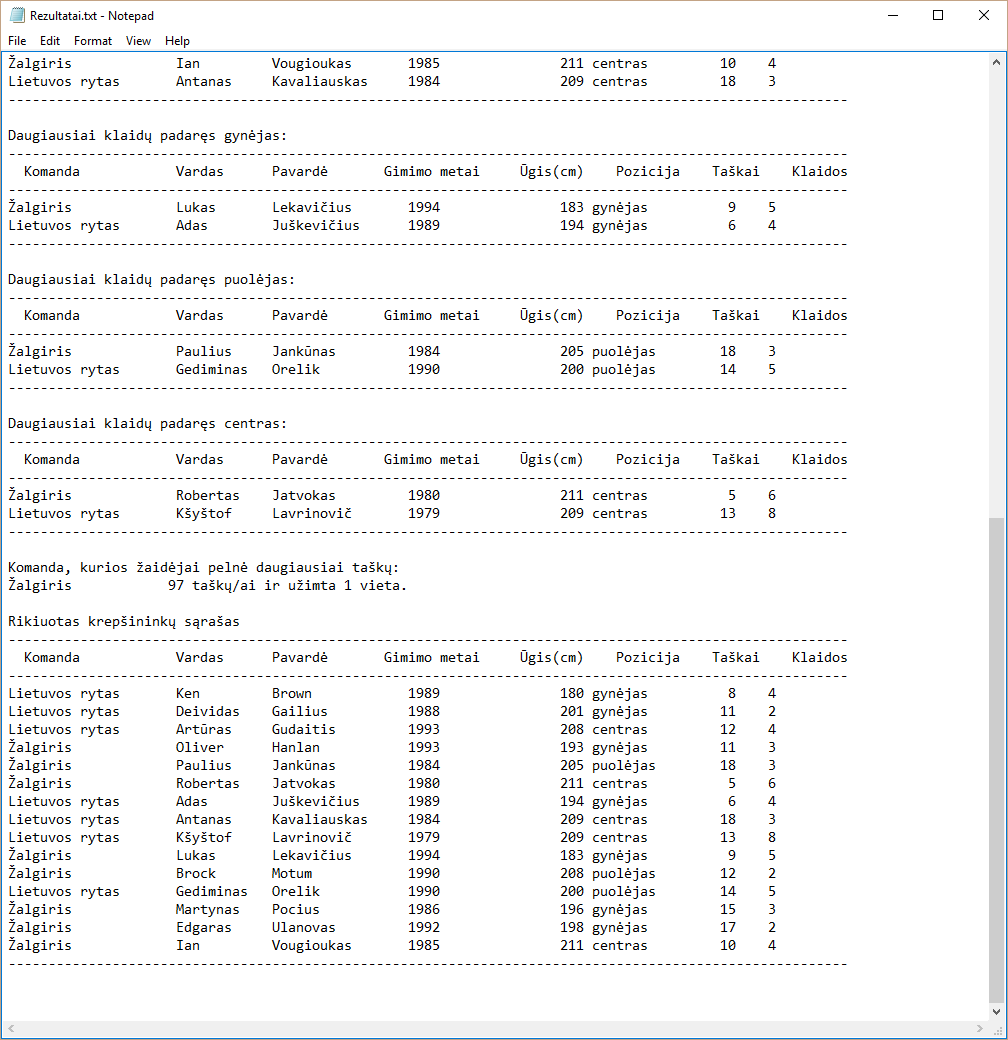
Pradiniai U13a.txt duomenys:

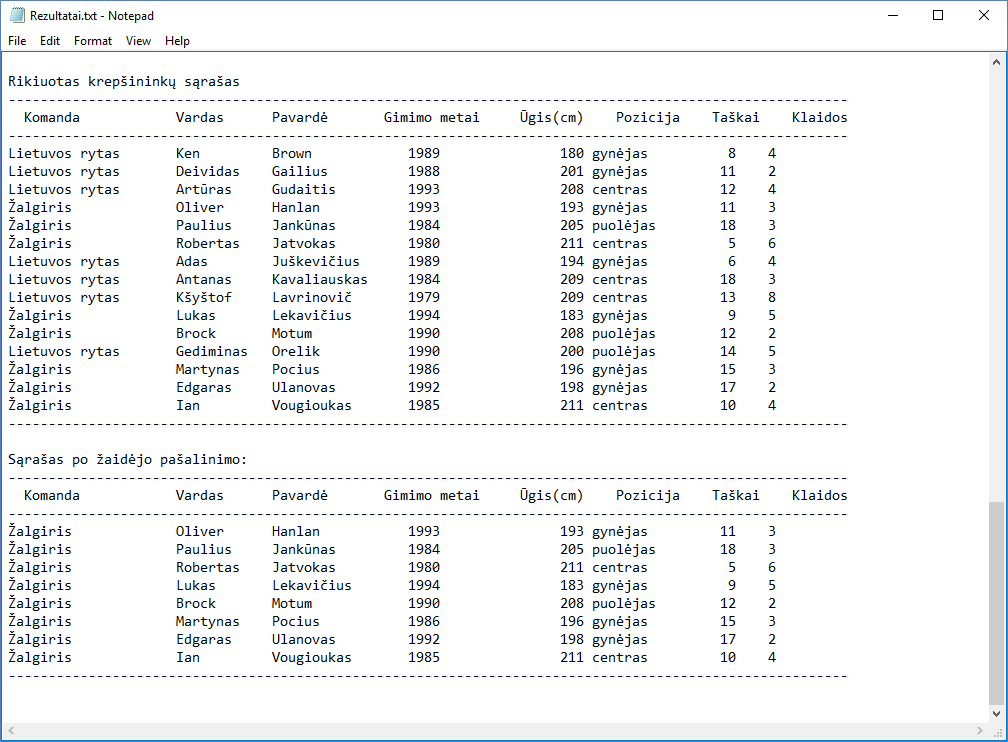


Pradiniai U13b.txt duomenys:

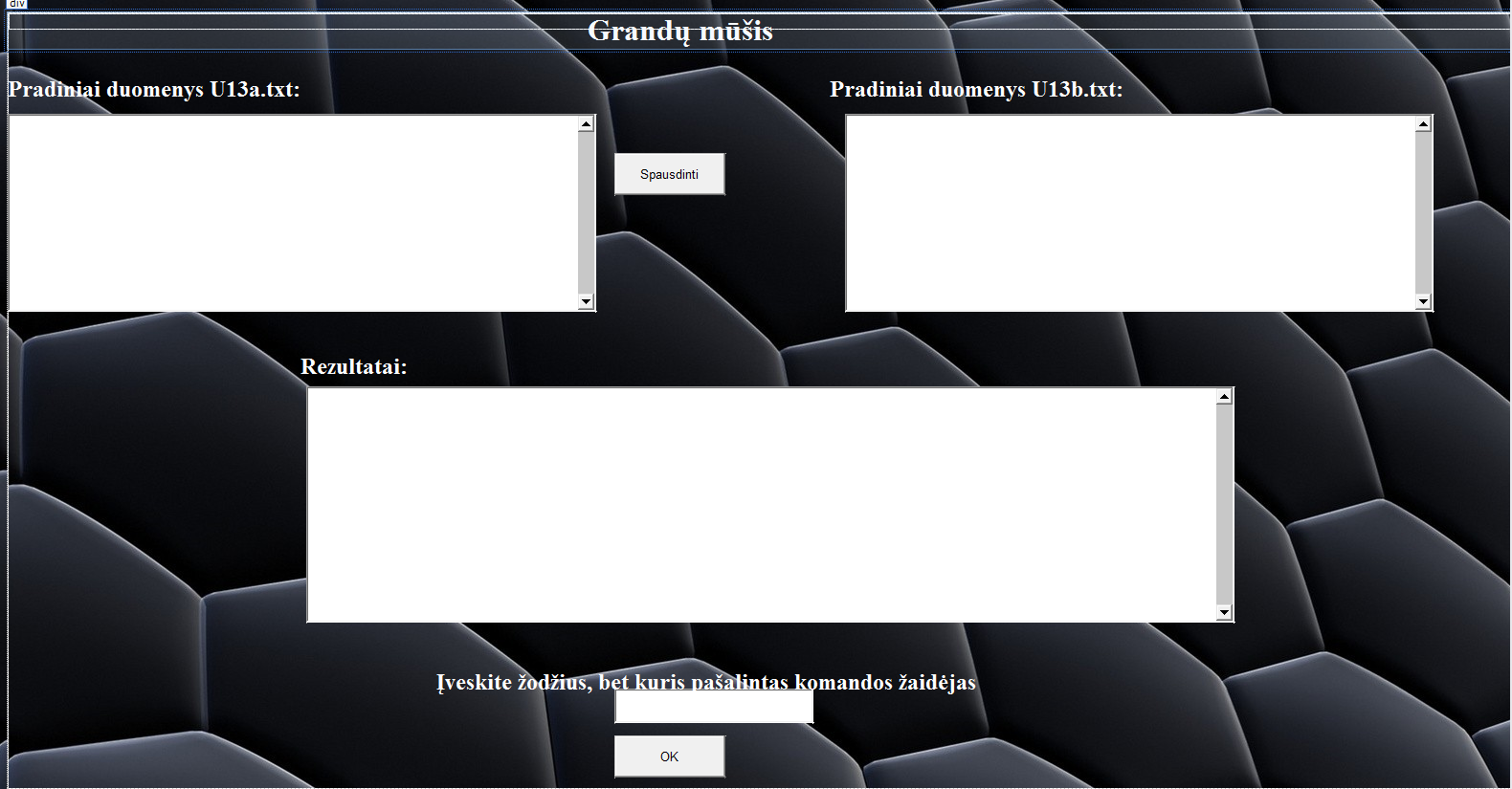


Rezultatai:





## Grafinės vartotojo sąsajos schema



## Grafinės vartotojo sąsajos elementų pakeistos savybės

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Komponentas | ID | Text | On\_Click | Font - Blod | Font - Size | ForeColor | TextMode |
| Label | **Label1** | **Grandų mūšis** |  | **True** | **XX-Large** | **White** |  |
| Label | **Label2** | **Pradiniai duomenys U13a.txt** |  | **True** | **X-Large** | **White** |  |
| Label | **Label3** | **Pradiniai duomenys U13b.txt** |  | **True** | **X - Large** | **White** |  |
| Label | **Label4** | **Rezultatai:** |  | **True** | **X - Large** | **White** |  |
| Label | **Label5** | **Įveskite žodžius, kuris pašalintas komandos žaidėjai** |  | **True** | **X - Large** | **White** |  |
| TextBox | **TextBox1** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox2** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox3** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox4** |  |  |  |  |  |  |
| Button | **Button1** | **Spausdinti** | **Button1\_Click** |  |  |  |  |
| Button | **Button2** | **OK** | **Button2\_Click** |  |  |  |  |

## Klasių diagramos

|  |
| --- |
| **Krepšininkas** |
| **+ team: string**  **+ var: string**  **+ pav: string**  **+ gim: integer**  **+ ugis: integer**  **+ poz: string**  **+ taskas: integer**  **+ klaida: integer** |
| **+ Krepšininkas()**  **+ Krepšininkas(in team1: string, in vardas: string, in gimi: integer, in ugiss: integer, in pozicija: string, in point: integer, in klaid: integer)**  **+ ToString() {query}**  **+ >(in pirmas: Krepšininkas, in antras: Krepšininkas)**  **+ <(in pirmas: Krepšininkas, in antras: Krepšininkas)** |

|  |
| --- |
| **Komanda** |
| **+ team: string**  **+ rung: integer**  **+ win: integer** |
| **+ Komanda()**  **+ Komanda(in komanda: string, in rungtynes: integer, in laime: integer)**  **+ ToString() {query}**  **+ >(in pirmas: Komanda, in antras: Komanda)**  **+ <(in pirmas: Komanda, in antras: Komanda)** |

|  |
| --- |
| **Mazgas** |
| **+ Duom: Krepšininkas**  **+ Kitas: Mazgas** |
| **+ Mazgas(in reiksme: Krepšininkas, in adr: Mazgas)** |

|  |
| --- |
| **MazgasKom** |
| **+ Duom: Komanda**  **+ Kitas: MazgasKom** |
| **+ MazgasKom(in reiksme: Komanda, in adr: MazgasKom)** |

|  |
| --- |
| **Sąrašas** |
| **- pr: Mazgas**  **- pb: Mazgas**  **- d: Mazgas** |
| **+ Sąrašas()**  **+ DetiDuomenisA(in naujas: Krepšininkas)**  **+ Pradzia() {query}**  **+ Kitas(){query}**  **+ Yra(){query}**  **+ ImtisDuomenis(): Krepšininkas {query}**  **+ Rikiuoti() {query}**  **+ Pasalinti(in team: string, inout available: boolean)** |

|  |
| --- |
| **KitaSar** |
| **- prk: Mazgas**  **- pbk: Mazgas**  **- dk: Mazgas** |
| **+ KitaSar()**  **+ DetiDuomenisK(in naujas: Komanda)**  **+ PradziaK() {query}**  **+ KitasK(){query}**  **+ YraK(){query}**  **+ ImtisDuomenis(): Komanda {query}**  **+ Rikiuoti() {query}** |

## Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus

Programos darbiniame aplanke sukuriami tekstiniai duomenų failai U13a.txt ir U13b.txt.

Tekstiniame faile U13a.txt užpildoma informacija apie krepšininkus: komanda, vardas, pavardė, gimimo metai, ūgis, žaidimo pozicija, įmestų taškų skaičius, padarytų klaidų skaičius. Tekstiniame faile U13b.txt užpildoma informacija apie komandas: komanda, žaistų rungtynių skaičius, laimėtų rungtynių skaičius. Kiekviena komanda ir krepšininkas yra atskiroje eilutėje. Eilutėje vienus duomenis nuo kitų skiria vienas tarpas. Užpildžius duomenų failus, galima įjungti programą. Įjungus programą, užsikrauna duomenys, juos matome ekrane. Tarp duomenų tekstinių laukų matomas ir mygtukas „Spausdinti. Paspaudus mygtuką, programa atlieka skaičiavimus, rezultatai matomi naujame tekstiniame lauke: atvirkštinis žaidėjų sąrašas, turnyro lentelė, daugiausiai taškų pelnę žaidėjai, daugiausiai klaidų padarę žaidėjai, rikiuotas krepšininkų sąrašas. Žemiau laukelis „Pašalinti komandos žaidėjus“. Norint, kad programa pašalintų pageidaujamos komandos žaidėjus, reikia laukelyje surinkti komandos pavadinimą ir paspausti žemiau esantį mygtuką „OK“. Programai atlikus skaičiavimus, likusios komandos žaidėjų sąrašą galima pamatyti jau minėtame tekstiniame lauke „Rezultatai“.

# Bendrinis susietasis sąrašas

## Darbo užduotis

**U13 Žaidėjai.**

Krepšininkai pagal žaidimo poziciją komandoje skirstomi į 3 pozicijas: centrą, puolėją, gynėją. Krepšinio komandos tarpusavy žaidžia rungtynes. Vienos komandos laimi, o kitos pralaimi, lygiųjų nebūna. Sudarykite kiekvienos pozicijos geriausių žaidėjų sąrašus (taškai, klaidos).

Suraskite, kurios komandos žaidėjai pelnė daugiausiai taškų ir atspausdinkite šios komandos užimamą vietą.

Duomenys:

* Tekstiniame faile U13a.txt duota informacija apie krepšininkus: komanda, vardas, pavardė, gimimo metai, ūgis, žaidimo pozicija, įmestų taškų skaičius, padarytų klaidų skaičius.
* Tekstiniame faile U13b.txt duota informacija apie komandas: komanda, žaistų rungtynių skaičius, laimėtų rungtynių skaičius.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka. Iš žaidėjų sąrašo pašalinkite nurodytos komandos (įvedama klaviatūra) žaidėjus.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

using System.Collections;

using System.Text;

**Player.cs**

/// <summary>

/// Klasė krepšininko duomenims saugoti

/// </summary>

/// <class name="Player"></class>

public class Player : IComparable<Player>, IEquatable<Player>

{

public string team { get; set; } // Komanda

public string firstName { get; set; } // Vardas

public string lastName { get; set; } // Pavardė

public int birth { get; set; } // Gimimo metai

public int height { get; set; } // Ūgis

public string postion { set; get; } // Pozicija

public int point { get; set; } // taškas

public int turnovers { get; set; } // klaida

/// <summary>

/// Pradiniai krepšininko duomenys

/// </summary>

public Player() { }

/// <summary>

/// Krepšininko duomenų įrašymas

/// </summary>

/// <param name="tea">komanda</param>

/// <param name="first">vardas</param>

/// <param name="last">pavardė</param>

/// <param name="birt">gimimo metai</param>

/// <param name="hei">ūgis</param>

/// <param name="pos">pozicija</param>

/// <param name="poi">taškai</param>

/// <param name="tur">klaida</param>

public Player(string tea, string first, string last, int birt, int hei, string pos, int poi, int tur)

{

this.team = tea;

this.firstName = first;

this.lastName = last;

this.birth = birt;

this.height = hei;

this.postion = pos;

this.point = poi;

this.turnovers = tur;

}

/// <summary>

/// Spausdinimo metodas

/// </summary>

/// <returns>eilutė</returns>

public override string ToString()

{

string eilute;

eilute = string.Format("{0,-20}{1,-12}{2,-18}{3,-12}{4,10}{5, -9}{6,10}{7,5}", team, firstName, lastName, birth, height, postion, point, turnovers);

return eilute;

}

/// <summary>

/// Palyginimo sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns></returns>

public int CompareTo(Player other)

{

if (other == null) return 1;

if (lastName.CompareTo(other.lastName) != 0)

return lastName.CompareTo(other.lastName);

else

return 0;

}

/// <summary>

/// Palyginimo užkloto operatorius

/// </summary>

/// <param name="first">Krepšininkas pirmas</param>

/// <param name="second">Krepšininkas antras</param>

/// <returns>1</returns>

static public bool operator >(Player first, Player second)

{

return first.CompareTo(second) == 1;

}

/// <summary>

/// Palyginimo užkloto operatorius

/// </summary>

/// <param name="first">krepšininkas pirmas</param>

/// <param name="second">krepšininkas antras</param>

/// <returns>-1</returns>

static public bool operator <(Player first, Player second)

{

return first.CompareTo(second) == -1;

}

/// <summary>

/// Vienodų sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns>true ar false</returns>

public bool Equals(Player other)

{

if (other == null)

return false;

if (this.team == other.team)

return true;

else

return false;

}

/// <summary>

/// Vienodų sąsajos užkloto metodas

/// </summary>

/// <param name="obj">objektas</param>

/// <returns>true ar false</returns>

public override bool Equals(Object obj)

{

if (obj == null)

return false;

Player play = obj as Player;

if (play == null)

return false;

else

return Equals(play);

}

/// <summary>

/// Užklotas metodas

/// </summary>

/// <returns>gethashcode</returns>

public override int GetHashCode()

{

return base.GetHashCode();

}

}

**Team.cs**

/// <summary>

/// Klasė krepšininkų komandos duomenims saugoti

/// </summary>

/// <class name="Team"></class>

public class Team : IComparable<Team>, IEquatable<Team>

{

public string team { get; set; } // komandos pavadinimas

public int match { get; set; } // rungtynės

public int win { get; set; } // laimėjimai

/// <summary>

/// Pradiniai komandos duomenys

/// </summary>

public Team() { }

/// <summary>

/// Komandų duomenų įrašymas

/// </summary>

/// <param name="komanda">komandos pavadinimas</param>

/// <param name="rungtynes">rungtynės</param>

/// <param name="laime">laimėjimai</param>

public Team(string komanda, int rungtynes, int laime)

{

this.team = komanda;

this.match = rungtynes;

this.win = laime;

}

/// <summary>

/// Spausdinimo metodas

/// </summary>

/// <returns>eilute</returns>

public override string ToString()

{

string eilute = string.Format("{0,-20}{1,7}{2,30}", team, match, win);

return eilute;

}

/// <summary>

/// Palyginimo sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns></returns>

public int CompareTo(Team other)

{

if (other == null)

return 1;

else

return win.CompareTo(other.win);

}

/// <summary>

/// Palyginimo užkloto metodas

/// </summary>

/// <param name="first">komanda pirma</param>

/// <param name="second">komanda antra</param>

/// <returns>1</returns>

static public bool operator >(Team first, Team second)

{

return first.CompareTo(second) == 1;

}

/// <summary>

/// Palyginimo užkloto metodas

/// </summary>

/// <param name="first">komanda pirma</param>

/// <param name="second">komanda antra</param>

/// <returns>-1</returns>

static public bool operator <(Team first, Team second)

{

return first.CompareTo(second) == -1;

}

/// <summary>

/// Vienodų sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns>true ar false</returns>

public bool Equals(Team other)

{

if (other == null)

return false;

if (this.win == other.win)

return true;

else

return false;

}

/// <summary>

/// Vienodų sąsajos užkloto metodas

/// </summary>

/// <param name="obj">objektas</param>

/// <returns>true ar false</returns>

public override bool Equals(Object obj)

{

if (obj == null)

return false;

Team team = obj as Team;

if (team == null)

return false;

else

return Equals(team);

}

/// <summary>

/// Užklotas metodas GetHashCode

/// </summary>

/// <returns>GetHashCode</returns>

public override int GetHashCode()

{

return base.GetHashCode();

}

}

**Node.cs**

/// <summary>

/// Klasė Mazgo duomenims saugoti

/// </summary>

/// <class name="Node"></class>

public sealed class Node<type> where type : IComparable<type>, IEquatable<type>

{

public type Data { get; set; } // tipo duomenys

public Node<type> Next { get; set; } // tipo kito mazgo adresas

/// <summary>

/// Mazgo duomenų įrašymas

/// </summary>

/// <param name="value">reikšmė</param>

/// <param name="adr">adresas</param>

public Node(type value, Node<type> adr)

{

Data = value;

Next = adr;

}

}

**Sarasas.cs**

/// <summary>

/// Klasė susietojo sąrašo duomenims saugoti

/// </summary>

/// <class name="Sarasas"></class>

/// <typeparam name="tipo klasės"></typeparam>

public sealed class Sarasas<type> : IEnumerable<type> where type : IComparable<type>, IEquatable<type>

{

private Node<type> st; // tipo mazgo pradžia

private Node<type> end; // tipo mazgo pabaiga

/// <summary>

/// Pradiniai sąrašo duomenys

/// </summary>

public Sarasas()

{

this.st = null;

this.end = null;

}

/// <summary>

/// Grąžina tipo klasės objektą

/// </summary>

/// <returns>objektas</returns>

public type ImtiDuomenis() { return st.Data; }

/// <summary>

/// Padeda į tipo objektų sąrašą naują tipo objektą

/// </summary>

/// <param name="news">tipo objektas</param>

public void DetiDuomenisA(type news)

{

st = new Node<type>(news, st);

}

/// <summary>

/// Grąžina sąsajos iteratorių

/// </summary>

/// <returns></returns>

public IEnumerator<type> GetEnumerator()

{

for (Node<type> dd = st; dd != null; dd = dd.Next)

{

yield return dd.Data;

}

}

IEnumerator IEnumerable.GetEnumerator()

{

throw new NotImplementedException();

}

/// <summary>

/// Surikiuoja krepšininkų masyvą išrinkimo metodu pagal abėcėlę.

/// </summary>

public void Sort()

{

for (Node<type> d1 = st; d1 != null; d1 = d1.Next)

{

Node<type> maxv = d1;

for (Node<type> d2 = d1; d2 != null; d2 = d2.Next)

if (d2.Data.CompareTo(maxv.Data) < 0)

maxv = d2;

type Pl = d1.Data;

d1.Data = maxv.Data;

maxv.Data = Pl;

}

}

/// <summary>

/// Surikiuoja komandų masyvą išrinkimo metodu pagal laimėjimus.

/// </summary>

public void Sort1()

{

for (Node<type> d1 = st; d1 != null; d1 = d1.Next)

{

Node<type> maxv = d1;

for (Node<type> d2 = d1; d2 != null; d2 = d2.Next)

if (d2.Data.CompareTo(maxv.Data) > 0)

maxv = d2;

type team = d1.Data;

d1.Data = maxv.Data;

maxv.Data = team;

}

}

/// <summary>

/// Suranda ir pašalina nurodytos komandos žaidėjus.

/// </summary>

/// <param name="one">krepšininko objektas</param>

/// <param name="available">taip ar ne</param>

public void Remove1(Player one, ref bool available)

{

Node<type> dd = st;

while (dd != null && dd.Data.Equals(one) == false)

dd = dd.Next;

if (dd != null) { available = true; }

Node<type> s = dd.Next;

while (s != null && s.Data.Equals(one) == true)

s = s.Next;

if (s == null)

{

Node<type> pb = dd;

for (Node<type> dt = st; dt != null; dt = dt.Next)

if (dt.Next == pb)

dt.Next = null;

}

else

{

dd.Data = s.Data;

dd.Next = s.Next;

}

s = null;

}

}

**LD3.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LD3.aspx.cs" Inherits="LD3" %>

<!DOCTYPE html>

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

<head runat="server">

<title>LD3</title>

<style type="text/css">

html, body

{

background-image: url('7018311-3d-black-background.jpg');

}

.newTable1 {

height: 25%;

width: 38%;

}

.newTable2 {

height: 30%;

width: 60%;

}

.newButton1 {

height:5%;

width: 7%;

position: absolute;

}

#form1 {

height: 722px;

width: 1610px;

}

</style>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Text="Grandų mūšis" Style="left: 39%; position: absolute; top: 15px; width: 201px;" Font-Bold="True" Font-Size="XX-Large" ForeColor="White" ></asp:Label>

<asp:Label ID="Label2" runat="server" Style="top: 10%; position: absolute;" Text="Pradiniai duomenys U13a.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Style="top: 15%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:TextBox ID="TextBox2" runat="server" Style="top: 15%; left: 55%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:Label ID="Label3" runat="server" Style="top: 10%; left: 55%; position: absolute;" Text="Pradiniai duomenys U13b.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:Button ID="Button1" runat="server" Style="top: 24%; left: 43%;" Text="Spausdinti" OnClick="Button1\_Click" CssClass="newButton1"/>

<asp:Label ID="Label4" runat="server" Style="top: 45%; left: 20%; position: absolute;" Text="Rezultatai:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox3" runat="server" Style="top: 50%; left: 20%; position: absolute;" TextMode="MultiLine" CssClass="newTable2" Enabled="False"></asp:TextBox>

<asp:TextBox ID="TextBox4" runat="server" Style="top: 89%; left: 40%; height: 30px; width: 200px; position: absolute;"></asp:TextBox>

<asp:Label ID="Label5" runat="server" Style="top: 85%; left: 29%; position: absolute" Text="Įveskite žodžius, bet kuris pašalintas komandos žaidėjai" Font-Bold="True" ForeColor="White" Font-Size="X-Large"></asp:Label>

<asp:Button ID="Button2" runat="server" Style="top: 95%; left: 42%; position:absolute;" Text="OK" CssClass="newButton1" OnClick="Button2\_Click"/>

</div>

</form>

</body>

</html>

**Program.apsx.cs**

public partial class LD3 : System.Web.UI.Page

{

const string fv1 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld3\U13a.txt";

const string fv2 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld3\U13b.txt";

const string fr = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld3\Rezultatai.txt";

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

{

TextBox1.Text = File.ReadAllText(fv1, Encoding.GetEncoding(1257));

TextBox2.Text = File.ReadAllText(fv2, Encoding.GetEncoding(1257));

Label4.Visible = false;

TextBox3.Visible = false;

Label5.Visible = false;

TextBox4.Visible = false;

Button2.Visible = false;

}

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

{

File.WriteAllText(fr, string.Empty);

Label4.Visible = true;

TextBox3.Visible = true;

Label5.Visible = true;

TextBox4.Visible = true;

Button2.Visible = true;

Button1.Visible = false;

var A = new Sarasas<Player>();

var K = new Sarasas<Team>();

var guardPoint = new Sarasas<Player>();

var forwardPoint = new Sarasas<Player>();

var centerPoint = new Sarasas<Player>();

var guardT = new Sarasas<Player>();

var forwardT = new Sarasas<Player>();

var centerT = new Sarasas<Player>();

readData1(fv1, A);

printData1(fr, A, "Atvirkštinis žaidėjų sąrašas");

readData2(fv2, K);

printData2(fr, K, "Turyno lentelė");

Point(A, K, 2, ref guardPoint, ref forwardPoint, ref centerPoint);

printData1(fr, guardPoint, "Daugiausiai taškų pelnęs gynėjas:");

printData1(fr, forwardPoint, "Daugiausiai taškų pelnęs puolėjas:");

printData1(fr, centerPoint, "Daugiausiai taškų pelnęs centras:");

Turnovers(A, K, 2, ref guardT, ref forwardT, ref centerT);

printData1(fr, guardT, "Daugiausiai klaidų padaręs gynėjas:");

printData1(fr, forwardT, "Daugiausiai klaidų padaręs puolėjas:");

printData1(fr, centerT, "Daugiausiai klaidų padaręs centras:");

K.Sort1();

TeamPoint(A, K, fr);

A.Sort();

printData1(fr, A, "Rikiuotas krepšininkų sąrašas");

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Label4.Visible = true;

TextBox3.Visible = true;

var A = new Sarasas<Player>();

var K = new Sarasas<Team>();

readData1(fv1, A);

readData2(fv2, K);

string team = TextBox4.Text;

bool available = false;

if (team != "")

{

foreach (Player one in A)

if (one.team == team)

A.Remove1(one, ref available);

if (available)

{

A.Sort();

printData1(fr, A, "Sąrašas po žaidėjo pašalinimo:");

}

else

{

Label5.Visible = true;

Label5.Text = "Tokio žaidėjo nėra sąraše !!!";

}

}

else

{

Label5.Visible = true;

Label5.Text = "Jūs neįveskite nė žodžio !!!";

}

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

/// <summary>

/// Skaitomi krepšininkų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

static void readData1(string fv, Sarasas<Player> A)

{

string team, pav, vard, poz;

int gim, ugis, point, kl;

string line;

using (StreamReader reader = new StreamReader(fv, Encoding.GetEncoding(1257)))

while ((line = reader.ReadLine()) != null)

{

var Split = line.Split(';');

team = Split[0];

vard = Split[1];

pav = Split[2];

gim = int.Parse(Split[3]);

ugis = int.Parse(Split[4]);

poz = Split[5];

point = int.Parse(Split[6]);

kl = int.Parse(Split[7]);

var kre = new Player(team, vard, pav, gim, ugis, poz, point, kl);

A.DetiDuomenisA(kre);

}

}

/// <summary>

/// Skaitomi komandos krepšininkų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

static void readData2(string fv, Sarasas<Team> A)

{

string team;

int rung, win;

string line;

using (StreamReader reader = new StreamReader(fv, Encoding.GetEncoding(1257)))

while ((line = reader.ReadLine()) != null)

{

string[] Split = line.Split(';');

team = Split[0];

rung = int.Parse(Split[1]);

win = int.Parse(Split[2]);

var kom = new Team(team, rung, win);

A.DetiDuomenisA(kom);

}

}

/// <summary>

/// Spausdinami krepšinikų duomenys į failą

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="eil">Paraštė</param>

static void printData1<type>(string fv, IEnumerable<type> A, string eil) where type : IComparable<type>, IEquatable<type>

{

using (var frr = new StreamWriter(fv, true))

{

string bruk = new string('-', 105);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Komanda Vardas Pavardė Gimimo metai Ūgis(cm) Pozicija Taškai Klaidos");

frr.WriteLine(bruk);

foreach (type one in A)

frr.WriteLine(one);

frr.WriteLine(bruk);

frr.WriteLine();

}

}

/// <summary>

/// Spausdinami komandos krepšinikų duomenys į failą

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="eil"></param>

static void printData2<type>(string fv, IEnumerable<type> A, string eil) where type : IComparable<type>, IEquatable<type>

{

using (var frr = new StreamWriter(fv, true))

{

string bruk = new string('-', 70);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Komanda Žaistų rungtynių skaičius Laimėtų rungtynių skaičius");

frr.WriteLine(bruk);

foreach (type one in A)

frr.WriteLine(one);

frr.WriteLine(bruk);

frr.WriteLine();

}

}

/// <summary>

/// Paieška ir saraso formavimas, kuris kiekvienoje pozicijoje žaidėjas daugiausiai pelnė taškų.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="B">Konteinerinės komandos vardas</param>

/// <param name="i">Masyvo indeksas</param>

/// <param name="G">Konteinerinės gynejo vardas</param>

/// <param name="P">Konteinerinės puolejo vardas</param>

/// <param name="C">Konteinerinės centro vardas</param>

static void Point(Sarasas<Player> A, Sarasas<Team> B, int i, ref Sarasas<Player> G, ref Sarasas<Player> P, ref Sarasas<Player> C)

{

Player pla = new Player();

string[] pozicija = { " gynėjas", " puolėjas", " centras" };

if (i == -1)

return;

foreach (Team one in B)

{

int suma = 0;

foreach (Player value in A)

if (value.postion == pozicija[i] && value.team == one.team)

if (value.point > suma)

{

suma = value.point;

pla = value;

}

if (pozicija[i] == pozicija[2])

C.DetiDuomenisA(pla);

else

if (pozicija[i] == pozicija[1])

P.DetiDuomenisA(pla);

else

G.DetiDuomenisA(pla);

}

Point(A, B, i - 1, ref G, ref P, ref C);

}

/// <summary>

/// Paieška ir saraso formavimas, kuris kiekvienoje pozicijoje žaidėjas daugiausiai padarė klaidu.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="B">Konteinerinės komandos vardas</param>

/// <param name="i">Masyvo indeksas</param>

/// <param name="G">Konteinerinės gynejo vardas</param>

/// <param name="P">Konteinerinės puolejo vardas</param>

/// <param name="C">Konteinerinės centro vardas</param>

static void Turnovers(Sarasas<Player> A, Sarasas<Team> B, int i, ref Sarasas<Player> G, ref Sarasas<Player> P, ref Sarasas<Player> C)

{

Player pla = new Player();

string[] pozicija = { " gynėjas", " puolėjas", " centras" };

if (i == -1)

return;

foreach (Team one in B)

{

int suma = 0;

foreach (Player value in A)

if (value.postion == pozicija[i] && value.team == one.team)

if (value.turnovers > suma)

{

suma = value.turnovers;

pla = value;

}

if (pozicija[i] == pozicija[2])

C.DetiDuomenisA(pla);

else

if (pozicija[i] == pozicija[1])

P.DetiDuomenisA(pla);

else

G.DetiDuomenisA(pla);

}

Turnovers(A, B, i - 1, ref G, ref P, ref C);

}

/// <summary>

/// Paieška ir spausdinimas, kurios komandos žaidėjai pelnė daugiausiai taškų ir atspausdinama šios komandos užimta vieta.

/// </summary>

/// <param name="A">Krepšininko konteinerinės vardas</param>

/// <param name="B">Komandos konteinerinės vardas</param>

/// <param name="fv">Duomenų failo vardas</param>

static void TeamPoint(Sarasas<Player> A, Sarasas<Team> B, string fv)

{

int max = 0;

int k = 0;

string team = "";

foreach (Team one in B)

{

string komanda = one.team;

int win = one.win;

Max(A, komanda, win, ref max, ref team);

}

foreach (Team one in B)

{

k++;

using (var frr = new StreamWriter(fv, true))

{

if (one.team == team)

{

frr.WriteLine("Komanda, kurios žaidėjai pelnė daugiausiai taškų:");

frr.WriteLine("{0, -20}{1,2} taškų/ai ir užimta{2,2} vieta.", team, max, k);

frr.WriteLine();

}

}

}

}

/// <summary>

/// Suskaičiuoja ir grąžina, kurios komandos žaidėjai pelnė daugiausiai taškų ir palyginima kas daugiausiai pelnė taškų.

/// </summary>

/// <param name="A">Konteinerinės vardas</param>

/// <param name="komanda">Komandos pavadinimas</param>

/// <param name="win">Komandos laimeti</param>

/// <param name="max">Daugiausiai taškų</param>

/// <param name="team">Komandos pavadinimas, kuris daugiausiai taškų</param>

static void Max(Sarasas<Player> A, string komanda, int win, ref int max, ref string team)

{

int suma = 0;

int win1 = 0;

foreach (Player one in A)

{

if (komanda == one.team)

suma = suma + one.point;

}

if (suma > max)

{

team = komanda;

max = suma;

win1 = win;

}

else

if (suma == max)

if (win > win1)

{

win1 = win;

team = komanda;

max = suma;

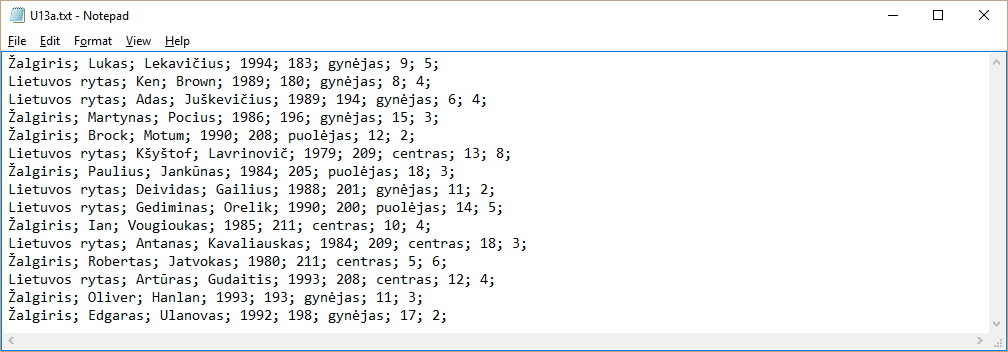
}

}

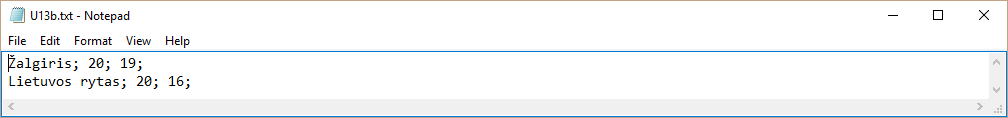
}

## Pradiniai duomenys ir rezultatai

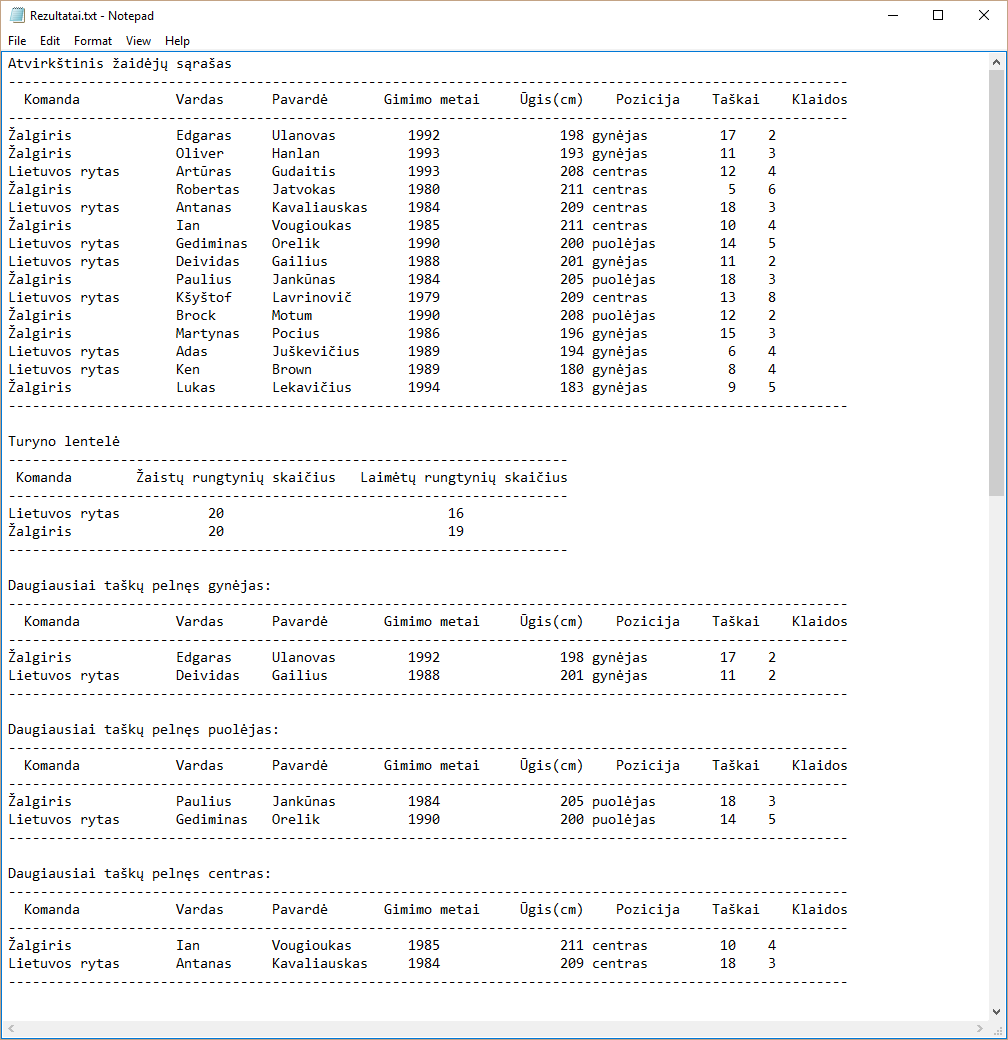
Pradiniai U13a.txt duomenys:

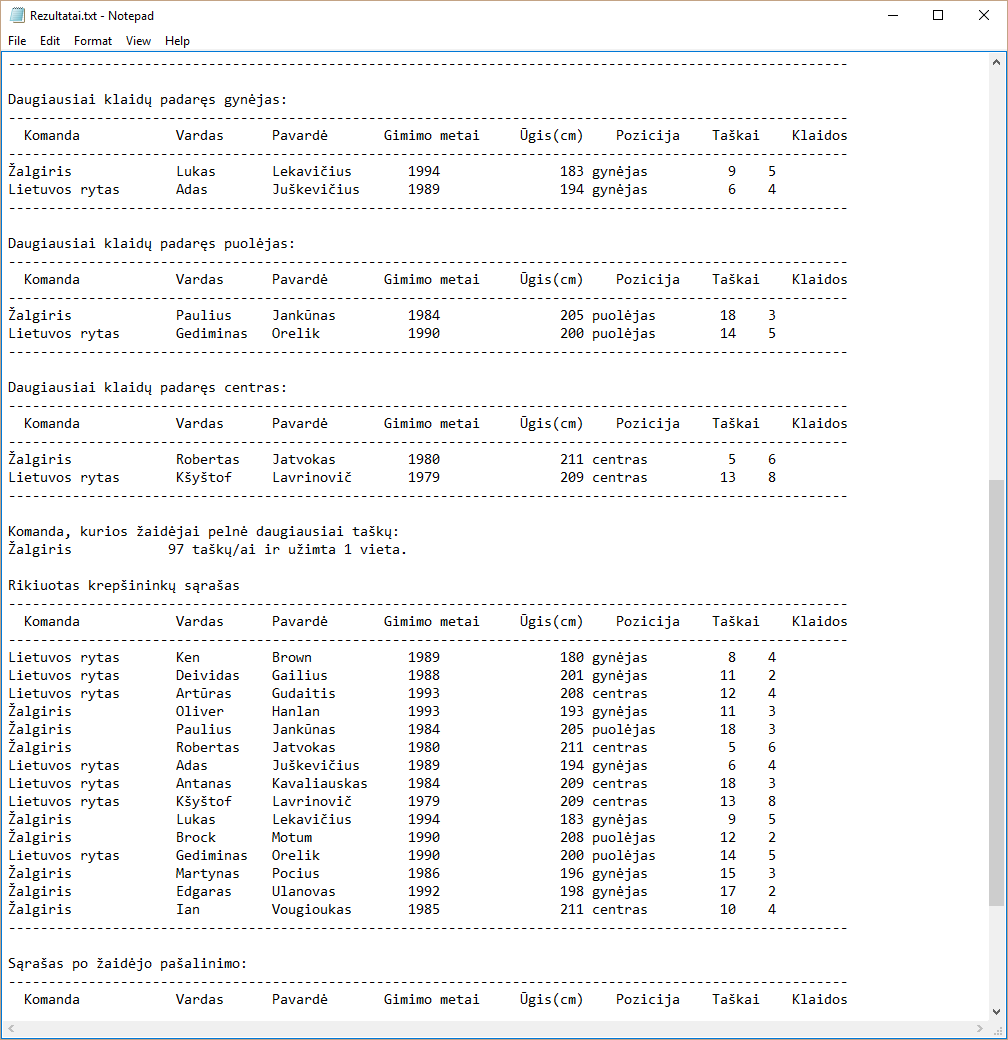


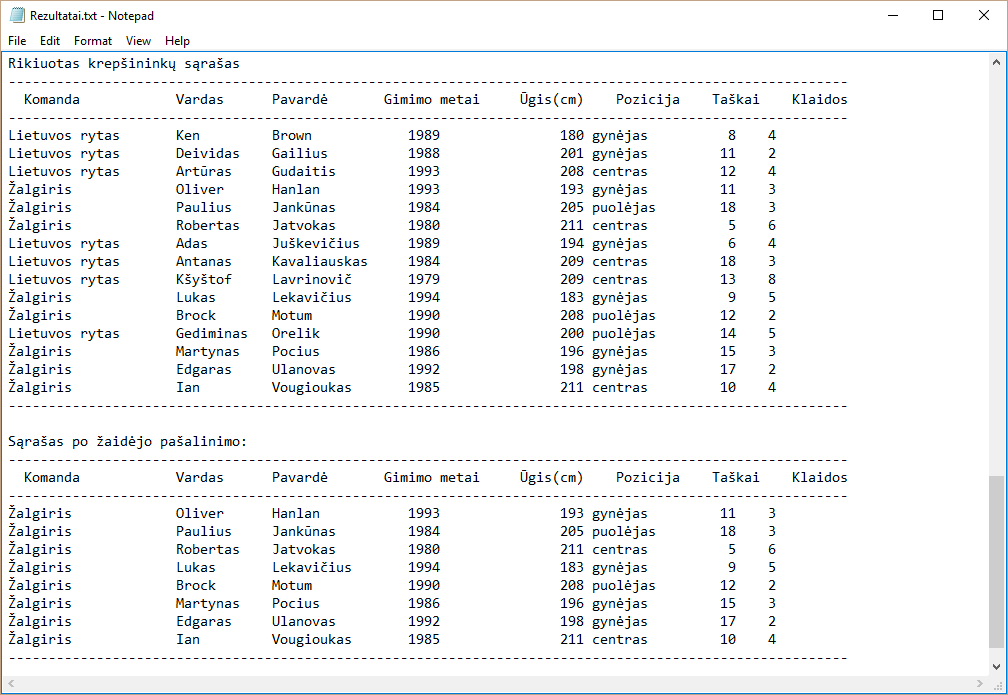
Pradiniai U13b.txt duomenys:



Rezultatai:







## Grafinės vartotojo sąsajos schema



Button2

TextBox4

Label5

TextBox3

Button1

TextBox2

TextBox1

Label4

Label3

Label2

Label1

## Grafinės vartotojo sąsajos elementų pakeistos savybės

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Komponentas | ID | Text | On\_Click | Font - Blod | Font - Size | ForeColor | TextMode |
| Label | **Label1** | **Grandų mūšis** |  | **True** | **XX-Large** | **White** |  |
| Label | **Label2** | **Pradiniai duomenys U13a.txt** |  | **True** | **X-Large** | **White** |  |
| Label | **Label3** | **Pradiniai duomenys U13b.txt** |  | **True** | **X - Large** | **White** |  |
| Label | **Label4** | **Rezultatai:** |  | **True** | **X - Large** | **White** |  |
| Label | **Label5** | **Įveskite žodžius, kuris pašalintas komandos žaidėjai** |  | **True** | **X - Large** | **White** |  |
| TextBox | **TextBox1** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox2** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox3** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox4** |  |  |  |  |  |  |
| Button | **Button1** | **Spausdinti** | **Button1\_Click** |  |  |  |  |
| Button | **Button2** | **OK** | **Button2\_Click** |  |  |  |  |

## Klasių diagramos

|  |
| --- |
| **Player** |
| **+ team: string**  **+ firstName: string**  **+ lastName: string**  **+ birth: integer**  **+ height: integer**  **+ postion: string**  **+ point: integer**  **+ turnovers: integer** |
| **+ Player()**  **+ Player(in tea: string, in first: string, in last: string, in birt: integer, in hei: integer, in pos: string, in poi: integer, in tur: integer)**  **+ ToString() {query}**  **+ CompareTo(in other: Player)**  **+ >(in first: Player, in second: Player)**  **+ <(in first: Player, in second: Player)**  **+ Equals(in other: Player)**  **+ Equals(in obj: Object)**  **+ GetHashCode() {query}** |

|  |
| --- |
| **Team** |
| **+ team: string**  **+ match: integer**  **+ win: integer** |
| **+ Team()**  **+ Team(in komanda: string, in rungtynes: integer, in laime: integer)**  **+ ToString() {query}**  **+ CompareTo(in other: Team)**  **+ >(in first: Team, in second: Team)**  **+ <(in first: Team, in second: Team)**  **+ Equals(in other: Team)**  **+ Equals(in obj: Object)**  **+ GetHashCode() {query}** |

|  |
| --- |
| **Node** |
| **+ Data: type**  **+ Next: Node<type>** |
| **+ Node(in value: type, in adr: Node<type>)** |

|  |
| --- |
| **Sarasas** |
| **- st: Node<type>**  **- end: Node<type>** |
| **+ Sąrašas()**  **+ ImtisDuomenis(): type**  **+ DetiDuomenisA(in news: type)**  **+ GetEnumerator(): IEnumerator<type>**  **+ IEnumerable.GetEnumerator(): IEnumerator**  **+ Sort() {query}**  **+ Sort() {query}**  **+ Remove(in one: Player, inout available: boolean)** |

## Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus

Programos darbiniame aplanke sukuriami tekstiniai duomenų failai U13a.txt ir U13b.txt.

Tekstiniame faile U13a.txt užpildoma informacija apie krepšininkus: komanda, vardas, pavardė, gimimo metai, ūgis, žaidimo pozicija, įmestų taškų skaičius, padarytų klaidų skaičius. Tekstiniame faile U13b.txt užpildoma informacija apie komandas: komanda, žaistų rungtynių skaičius, laimėtų rungtynių skaičius. Kiekviena komanda ir krepšininkas yra atskiroje eilutėje. Eilutėje vienus duomenis nuo kitų skiria vienas tarpas. Užpildžius duomenų failus, galima įjungti programą. Įjungus programą, užsikrauna duomenys, juos matome ekrane. Tarp duomenų tekstinių laukų matomas ir mygtukas „Spausdinti. Paspaudus mygtuką, programa atlieka skaičiavimus, rezultatai matomi naujame tekstiniame lauke: atvirkštinis žaidėjų sąrašas, turnyro lentelė, daugiausiai taškų pelnę žaidėjai, daugiausiai klaidų padarę žaidėjai, rikiuotas krepšininkų sąrašas. Žemiau laukelis „Pašalinti komandos žaidėjus“. Norint, kad programa pašalintų pageidaujamos komandos žaidėjus, reikia laukelyje surinkti komandos pavadinimą ir paspausti žemiau esantį mygtuką „OK“. Programai atlikus skaičiavimus, likusios komandos žaidėjų sąrašą galima pamatyti jau minėtame tekstiniame lauke „Rezultatai“.

# Bendrinės kolekcijos

## Darbo užduotis

**U13 Telefonų abonentai.**

Dokumento 1-oje eilutėje yra gatvės pavadinimas. Kiekvienoje kitoje eilutėje yra: pavardė, namo nr., telefono nr. Atskirame faile yra keičiamų telefono numerių sąrašas (sąraše: senas nr., naujas nr.). Sudaryti sąrašą abonentų, kurių telefonų numeriai pasikeitė. Sąraše: pavardė, gatvė, namo nr., naujas telefono nr. Rikiavimo raktas: namo nr., pavardė. Kurioje gatvėje pakeista daugiausia telefonų?

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Text;

using System.IO;

using System.Collections;

**Resident.cs**

/// <summary>

/// Klasė Gyventojaus duomenis saugoti

/// </summary>

/// <class name="Resident"></class>

public class Resident : IComparable<Resident>, IEquatable<Resident>

{

public string street { get; set; } // gatvė

public string name { get; set; } // pavardė

public int homeNR { get; set; } // namo nr.

public int telNr { get; set; } // tel. nr.

/// <summary>

/// Pradiniai gyventojaus duomenys

/// </summary>

public Resident() { }

/// <summary>

/// Spausdinimo metodas

/// </summary>

/// <returns>formatas</returns>

public override string ToString()

{

return string.Format("{0, -12}{1, -13}{2, 8}{3, 20}", name, street, homeNR, telNr);

}

/// <summary>

/// Palyginimo sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns></returns>

public int CompareTo(Resident other)

{

if (other == null) return 1;

if (homeNR.CompareTo(other.homeNR) != 0)

return homeNR.CompareTo(other.homeNR);

else

return name.CompareTo(other.name);

}

/// <summary>

/// Vienodų sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns>true ar false</returns>

public bool Equals(Resident other)

{

if (other == null)

return false;

if (this.street == other.street)

return true;

else

return false;

}

/// <summary>

/// Vienodų sąsajos užkloto metodas

/// </summary>

/// <param name="obj">objektas</param>

/// <returns>true ar false</returns>

public override bool Equals(Object obj)

{

if (obj == null)

return false;

Resident res = obj as Resident;

if (res == null)

return false;

else

return Equals(res);

}

/// <summary>

/// Užklotas metodas

/// </summary>

/// <returns>gethascode</returns>

public override int GetHashCode()

{

return base.GetHashCode();

}

}

**TelNumber.cs**

/// <summary>

/// Klasė telefonų numerio duomenims saugoti

/// </summary>

/// <class name="TelNumber"></class>

public class TelNumber : IComparable<TelNumber>, IEquatable<TelNumber>

{

public int oldTelNr { get; set; } // senas telefono numeris

public int newTelNr { get; set; } // naujas telefono numeris

/// <summary>

/// Pradiniai telefonų numerio duomenys

/// </summary>

public TelNumber() { }

/// <summary>

/// Palyginimo sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns></returns>

public int CompareTo(TelNumber other)

{

if (other == null) return 1;

if (oldTelNr.CompareTo(other.oldTelNr) != 0)

return oldTelNr.CompareTo(other.oldTelNr);

else

return -1;

}

/// <summary>

/// Vienodų sąsajos metodas

/// </summary>

/// <param name="other">objektas</param>

/// <returns>true ar false</returns>

public bool Equals(TelNumber other)

{

if (other == null)

return false;

if (this.oldTelNr == other.oldTelNr)

return true;

else

return false;

}

/// <summary>

/// Vienodų sąsajos užkloto metodas

/// </summary>

/// <param name="obj">objektas</param>

/// <returns>true ar false</returns>

public override bool Equals(Object obj)

{

if (obj == null)

return false;

TelNumber res = obj as TelNumber;

if (res == null)

return false;

else

return Equals(res);

}

/// <summary>

/// Užklotas metodas

/// </summary>

/// <returns>gethashcode</returns>

public override int GetHashCode()

{

return base.GetHashCode();

}

}

**LD4.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LD4.aspx.cs" Inherits="LD4" %>

<!DOCTYPE html>

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

<head runat="server">

<title>LD4</title>

<style type="text/css">

html, body

{

background-image: url('7018311-3d-black-background.jpg');

}

.newTable1 {

height: 25%;

width: 32%;

}

.newTable2 {

height: 30%;

width: 60%;

}

.newButton1 {

height:5%;

width: 11%;

position: absolute;

}

#form1 {

height: 684px;

width: 1610px;

}

</style>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Text="Telefonų abonentai" Style="left: 40%; position: absolute; top: 15px; width: 282px;" Font-Bold="True" Font-Size="XX-Large" ForeColor="White" ></asp:Label>

<asp:Label ID="Label2" runat="server" Style="top: 10%; position: absolute;" Text="Pradiniai duomenys U13.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Style="top: 15%; position: absolute; left: 10px;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:Button ID="Button1" runat="server" Style="top: 25%; left: 42%; right: 802px;" Text="Spausdinti" OnClick="Button1\_Click" CssClass="newButton1"/>

<asp:TextBox ID="TextBox2" runat="server" Style="top: 15%; left: 65%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:TextBox ID="TextBox3" runat="server" Style="top: 52%; left: 18%; position: absolute;" TextMode="MultiLine" CssClass="newTable2" Enabled="False"></asp:TextBox>

<asp:Label ID="Label4" runat="server" Style="top: 10%; left: 65%; position: absolute;" Text="Telefonų numerių sąrašas" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:Label ID="Label3" runat="server" Style="top: 45%; left: 18%; position: absolute;" Text="Rezultatai:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

</div>

</form>

</body>

</html>

**LD4.aspx.cs**

public partial class LD4 : System.Web.UI.Page

{

const string fv1 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld4\Failas";

const string fvtel = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld4\TelefonoNr.txt";

const string fr = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld4\Rezultatai.txt";

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

{

TextBox3.Visible = false;

Label3.Visible = false;

string[] files = Directory.GetFiles(fv1);

TextBox1.Text = File.ReadAllText(files[0], Encoding.GetEncoding(1257));

TextBox1.Text = TextBox1.Text + "\n" + File.ReadAllText(files[1], Encoding.GetEncoding(1257));

TextBox1.Text = TextBox1.Text + "\n" + File.ReadAllText(files[2], Encoding.GetEncoding(1257));

TextBox2.Text = File.ReadAllText(fvtel, Encoding.GetEncoding(1257));

}

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

{

File.WriteAllText(fr, string.Empty);

Button1.Visible = false;

TextBox3.Visible = true;

Label3.Visible = true;

var A = new LinkedList<Resident>();

var Tel = new LinkedList<TelNumber>();

var NewList = new LinkedList<Resident>();

ReadData(fv1, A);

PrintData(fr, A, " Pradiniai duomenis");

ReadDataTel(fvtel, Tel);

NewLists(A, Tel, NewList);

PrintData(fr, NewList, "Abonentų sąrašas");

List<Resident> Cr;

Cr = NewList.ToList();

Cr.Sort();

PrintData(fr, Cr, "Rikiuotas");

Max(NewList, fr);

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

/// <summary>

/// Skaitomi gyventojų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

static void ReadData(string fv, LinkedList<Resident> A)

{

string line;

string[] files = Directory.GetFiles(fv);

foreach (string file in files)

{

using (var reader = new StreamReader(file, Encoding.GetEncoding(1257)))

{

line = reader.ReadLine();

string[] values = line.Split(';');

string street = values[0];

while ((line = reader.ReadLine()) != null)

{

values = line.Split(';');

var Res = new Resident();

Res.name = values[0];

Res.homeNR = Convert.ToInt32(values[1]);

Res.telNr = Convert.ToInt32(values[2]);

Res.street = street;

A.AddLast(Res);

}

}

}

}

/// <summary>

/// Skaitomi telefonų numerių duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

static void ReadDataTel(string fv, LinkedList<TelNumber> A)

{

string line;

using (var reader = new StreamReader(fv))

{

while ((line = reader.ReadLine()) != null)

{

string[] values = line.Split(';');

var Res = new TelNumber();

Res.oldTelNr = Convert.ToInt32(values[0]);

Res.newTelNr = Convert.ToInt32(values[1]);

A.AddLast(Res);

}

}

}

/// <summary>

/// Spausdinami gyventojų duomenys į failą

/// </summary>

/// <typeparam name="type">typas</typeparam>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

/// <param name="eil">Paraštė</param>

static void PrintData<type>(string fv, IEnumerable<type> A, string eil) where type : IComparable<type>, IEquatable<type>

{

using (var frr = new StreamWriter(fv, true))

{

//string rs = "";

string bruk = new string('-', 55);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Pavardė Gatvė Namas Nr. Tel.Nr.");

frr.WriteLine(bruk);

foreach (type one in A)

frr.WriteLine(one);

frr.WriteLine(bruk);

frr.WriteLine();

}

}

/// <summary>

/// Suranda ir sudaro sąrašą abonentų, kurių telefonų numeriai pasikeitė

/// </summary>

/// <param name="A">Gyventojų kolekcijos vardas</param>

/// <param name="TelNr">Telefonų numerių kolekcijos vardas</param>

/// <param name="New">Naujas sąrašas</param>

static void NewLists(LinkedList<Resident> A, LinkedList<TelNumber> TelNr, LinkedList<Resident> New)

{

foreach (Resident one in A)

{

foreach (TelNumber tel in TelNr)

{

if (one.telNr == tel.oldTelNr)

{

one.telNr = tel.newTelNr;

New.AddLast(one);

break;

}

}

}

}

/// <summary>

/// Suranda ir išspausdina, kurioje gatvėje pasikeičia daugiausiai telefonų numerių.

/// </summary>

/// <param name="A">Kolekcijos vardas</param>

/// <param name="fv">Duomenų failo vardas</param>

static void Max(LinkedList<Resident> A, string fv)

{

int max = 0;

IEnumerable<string> D = A.Select(m => m.street);

int number = 0;

string street = "";

string str = "";

foreach (string one in D)

{

if (one != street)

{

number = A.Count(l => l.street == one);

if (number > max)

{

max = number;

str = one;

}

}

street = one;

}

using (var frr = new StreamWriter(fv, true))

{

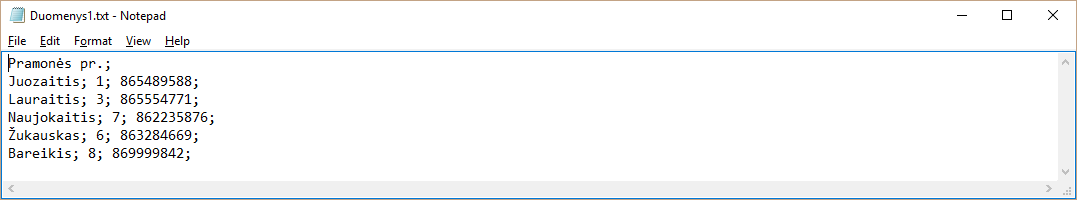
frr.WriteLine("{0} gatvėje pakeista daugiausia telefonų", str);

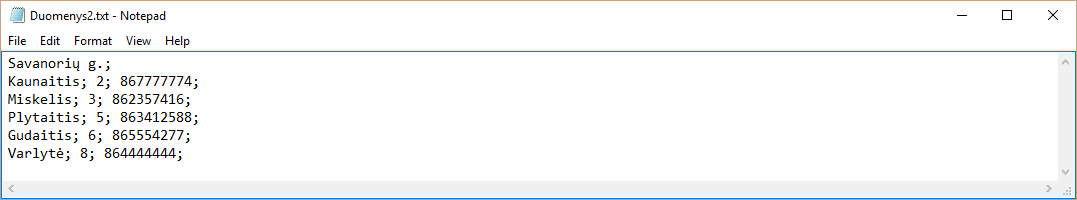
}

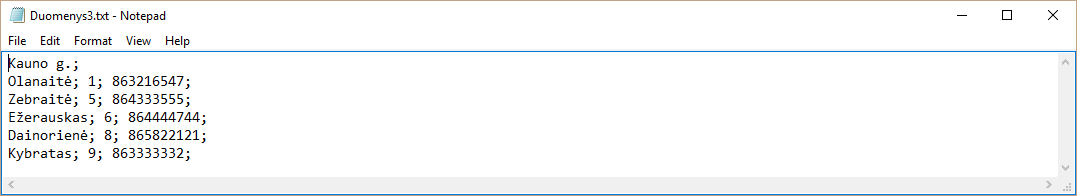
}

## Pradiniai duomenys ir rezultatai

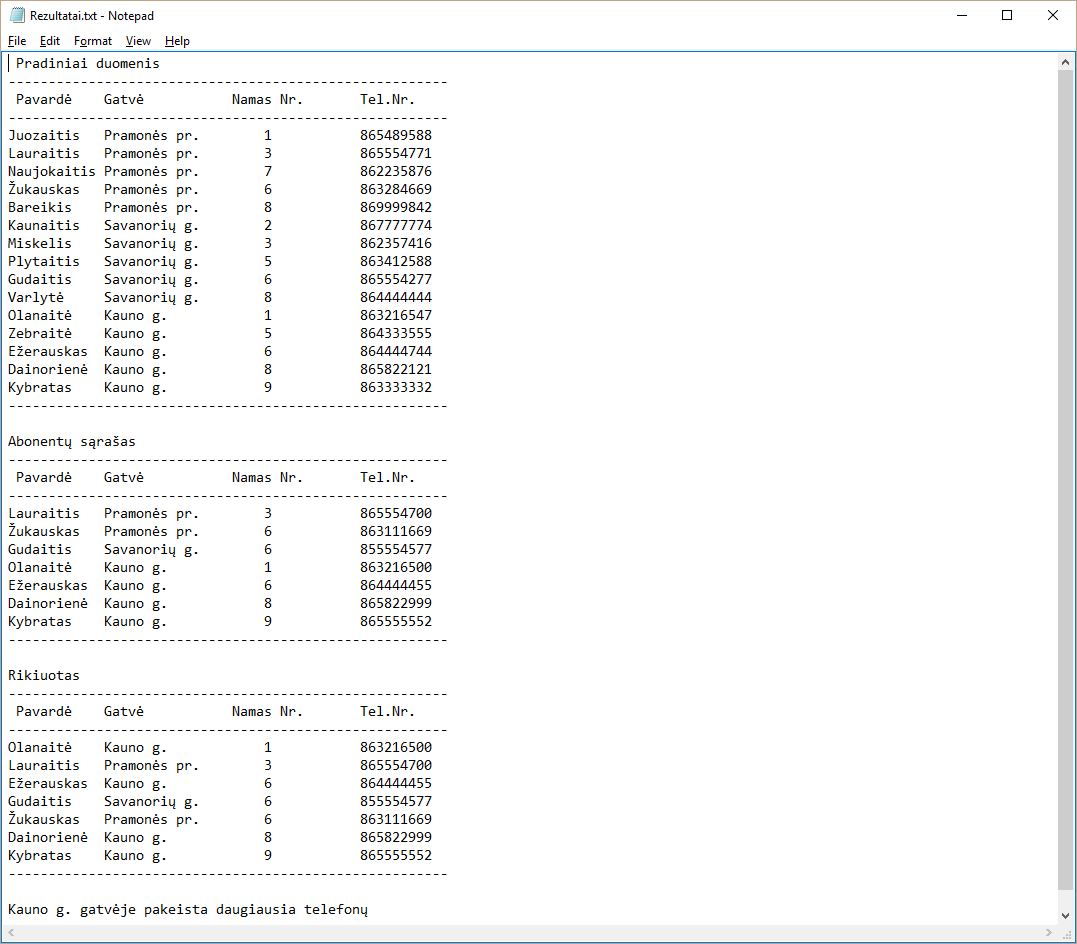
Pradiniai failo duomenys:



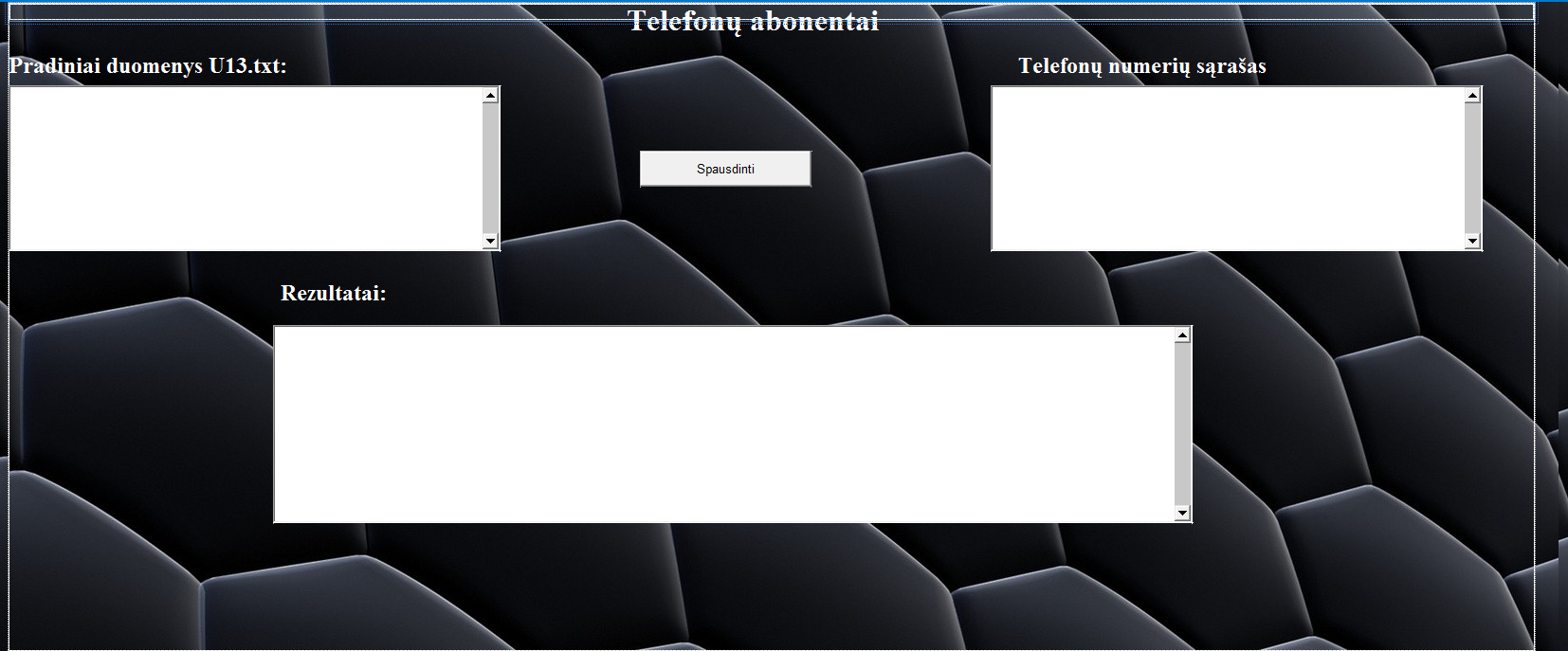




Rezultatai:



## Grafinės vartotojo sąsajos schema



Button1

Label3

TextBox3

TextBox2

TextBox1

Label1

Label4

Label2

## Grafinės vartotojo sąsajos elementų pakeistos savybės

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Komponentas | ID | Text | On\_Click | Font - Blod | Font - Size | ForeColor | TextMode |
| Label | **Label1** | **Telefonų abonentai** |  | **True** | **XX-Large** | **White** |  |
| Label | **Label2** | **Pradiniai failo duomenys** |  | **True** | **X-Large** | **White** |  |
| Label | **Label3** | **Rezultatai:** |  | **True** | **X - Large** | **White** |  |
| Label | **Label4** | **Telefonų numeris.txt** |  | **True** | **X - Large** | **White** |  |
| TextBox | **TextBox1** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox2** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox3** |  |  |  |  |  | **MultLine** |
| Button | **Button1** | **Spausdinti** | **Button1\_Click** |  |  |  |  |

## Klasių diagramos

|  |
| --- |
| **Resident** |
| **+ street: string**  **+ name: string**  **+ homeNr: integer**  **+ telNr: integer** |
| **+ Resident()**  **+ ToString() {query}**  **+ CompareTo(in other: Resident)**  **+ Equals(in other: Resident)**  **+ Equals(in obj: Object)**  **+ GetHashCode() {query}** |

|  |
| --- |
| **TelNumber** |
| **+ oldTelNr: integer**  **+ newTelNr: integer** |
| **+ TelNumber()**  **+ ToString() {query}**  **+ CompareTo(in other: TelNumber)**  **+ Equals(in other: TelNumber)**  **+ Equals(in obj: Object)**  **+ GetHashCode() {query}** |

## Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus

Programos darbiniame aplanke sukuriamas failo katalogas ir tekstinis duomenens failas TelefonoNr.txt. Failo katalogo viduje yra skirtingi 3 tekstiniai duomenų failai užpildyti informacija apie gyventojus: 1 – oje eilutėje yra gatvės pavadinimas, kiekvienoje kitoje eilutėje – pavardė, namo nr., telefono nr. Tekstiniame faile TelefonoNr.txt užpildyta informacija apie keičiamus telefono numerius: senas nr., naujas nr.. Eilutėje vienus duomenis nuo kitų skiria vienas tarpas. Užpildžius duomenų failus, galima įjungti programą. Įjungus programą, užsikrauna duomenys, juos matome ekrane. Tarp duomenų tekstinių laukų matomas ir mygtukas „Spausdinti“. Paspaudus mygtuką, programa atlieka skaičiavimus, rezultatai matomi naujame tekstiniame lauke: atvirkštinis gyventojų sąrašas, abonentų sąrašas, rikiuotas abonentų sąrašas, kiek gatvėje pakeista daugiausia telefonų.

# Deklaratyvusis programavimas

## Darbo užduotis

**U13 Telefonų abonentai.**

Dokumento 1-oje eilutėje yra gatvės pavadinimas. Kiekvienoje kitoje eilutėje yra: pavardė, namo nr., telefono nr. Atskirame faile yra keičiamų telefono numerių sąrašas (sąraše: senas nr., naujas nr.). Sudaryti sąrašą abonentų, kurių telefonų numeriai pasikeitė. Sąraše: pavardė, gatvė, namo nr., naujas telefono nr. Rikiavimo raktas: namo nr., pavardė. Kurioje gatvėje pakeista daugiausia telefonų?

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Text;

using System.IO;

using System.Collections;

**Resident.cs**

/// <summary>

/// Klasė Gyventojaus duomenis saugoti

/// </summary>

/// <class name="Resident"></class>

public class Resident

{

public string street { get; set; } // gatvė

public string name { get; set; } // pavardė

public int homeNR { get; set; } // namo nr.

public int telNr { get; set; } // tel. nr.

/// <summary>

/// Pradiniai gyventojaus duomenys

/// </summary>

public Resident() { }

/// <summary>

/// Spausdinimo metodas

/// </summary>

/// <returns>formatas</returns>

public override string ToString()

{

return string.Format("{0, -12}{1, -13}{2, 8}{3, 20}", name, street, homeNR, telNr);

}

}

**TelNumber.cs**

/// <summary>

/// Klasė telefonų numerio duomenims saugoti

/// </summary>

/// <class name="TelNumber"></class>

public class TelNumber

{

public int oldTelNr { get; set; } // senas telefono numeris

public int newTelNr { get; set; } // naujas telefono numeris

/// <summary>

/// Pradiniai telefonų numerio duomenys

/// </summary>

public TelNumber() { }

}

**LD5.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="LD5.aspx.cs" Inherits="LD5" %>

<!DOCTYPE html>

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

<head runat="server">

<title>LD5</title>

<style type="text/css">

html, body

{

background-image: url('7018311-3d-black-background.jpg');

}

.newTable1 {

height: 25%;

width: 32%;

}

.newTable2 {

height: 30%;

width: 60%;

}

.newButton1 {

height:5%;

width: 11%;

position: absolute;

}

#form1 {

height: 684px;

width: 1610px;

}

</style>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Text="Telefonų abonentai" Style="left: 40%; position: absolute; top: 15px; width: 282px;" Font-Bold="True" Font-Size="XX-Large" ForeColor="White" ></asp:Label>

<asp:Label ID="Label2" runat="server" Style="top: 10%; position: absolute;" Text="Pradiniai duomenys U13.txt:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Style="top: 15%; position: absolute; left: 10px;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:Button ID="Button1" runat="server" Style="top: 25%; left: 42%; right: 802px;" Text="Spausdinti" OnClick="Button1\_Click" CssClass="newButton1"/>

<asp:TextBox ID="TextBox2" runat="server" Style="top: 15%; left: 65%; position: absolute;" CssClass="newTable1" TextMode="MultiLine"></asp:TextBox>

<asp:TextBox ID="TextBox3" runat="server" Style="top: 52%; left: 18%; position: absolute;" TextMode="MultiLine" CssClass="newTable2" Enabled="False"></asp:TextBox>

<asp:Label ID="Label4" runat="server" Style="top: 10%; left: 65%; position: absolute;" Text="Telefonų numerių sąrašas" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

<asp:Label ID="Label3" runat="server" Style="top: 45%; left: 18%; position: absolute;" Text="Rezultatai:" Font-Bold="True" Font-Size="X-Large" ForeColor="White"></asp:Label>

</div>

</form>

</body>

</html>

**LD5.aspx.cs**

public partial class LD5 : System.Web.UI.Page

{

const string fv1 = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld5\Failas";

const string fvtel = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld5\TelefonoNr.txt";

const string fr = @"C:\Users\Z585\Documents\Visual Studio 2012\WebSites\Ld5\Rezultatai.txt";

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

{

TextBox3.Visible = false;

Label3.Visible = false;

string[] files = Directory.GetFiles(fv1);

foreach (var file in files)

TextBox1.Text = TextBox1.Text + "\n" + File.ReadAllText(file, Encoding.GetEncoding(1257));

TextBox2.Text = File.ReadAllText(fvtel, Encoding.GetEncoding(1257));

}

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

{

File.WriteAllText(fr, string.Empty);

Button1.Visible = false;

TextBox3.Visible = true;

Label3.Visible = true;

var A = new LinkedList<Resident>();

var Tel = new LinkedList<TelNumber>();

ReadData(fv1, A);

ReadDataTel(fvtel, Tel);

PrintData(fr, A, "Pradiniai duomenis");

List<Resident> NewList = (from a in A from b in Tel where a.telNr == b.oldTelNr select new Resident{ name = a.name, street = a.street, homeNR = a.homeNR, telNr = b.newTelNr}).ToList<Resident>();

if (NewList.Any())

PrintData(fr, NewList, "Abonentų sąrašas");

List<Resident> Sort = (from a in NewList orderby a.homeNR, a.name select a).ToList<Resident>();

PrintData(fr, Sort, "Rikiuota");

var B = NewList.ToList<Resident>();

var kok = from a in B select new { kieks = NewList.Count(m => m.street == a.street), pav = a.street};

var k = from a in kok where kok.Max(l => l.kieks) == a.kieks select a.pav;

using (var frr = new StreamWriter(fr, true))

{

string pav = k.First();

frr.WriteLine(pav + " " + "gatvėje pakeista daugiausia telefonų");

}

TextBox3.Text = File.ReadAllText(AppDomain.CurrentDomain.BaseDirectory + "Rezultatai.txt");

}

/// <summary>

/// Skaitomi gyventojų duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

static void ReadData(string fv, LinkedList<Resident> A)

{

string line;

string[] files = Directory.GetFiles(fv);

foreach (string file in files)

{

using (var reader = new StreamReader(file, Encoding.GetEncoding(1257)))

{

line = reader.ReadLine();

string[] values = line.Split(';');

string street = values[0];

while ((line = reader.ReadLine()) != null)

{

values = line.Split(';');

var Res = new Resident();

Res.name = values[0];

Res.homeNR = Convert.ToInt32(values[1]);

Res.telNr = Convert.ToInt32(values[2]);

Res.street = street;

A.AddLast(Res);

}

}

}

}

/// <summary>

/// Skaitomi telefonų numerių duomenys iš failo

/// </summary>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

static void ReadDataTel(string fv, LinkedList<TelNumber> A)

{

string line;

using (var reader = new StreamReader(fv))

{

while ((line = reader.ReadLine()) != null)

{

string[] values = line.Split(';');

var Res = new TelNumber();

Res.oldTelNr = Convert.ToInt32(values[0]);

Res.newTelNr = Convert.ToInt32(values[1]);

A.AddLast(Res);

}

}

}

/// <summary>

/// Spausdinami gyventojų duomenys į failą

/// </summary>

/// <typeparam name="type">typas</typeparam>

/// <param name="fv">Duomenų failo vardas</param>

/// <param name="A">Kolekcijos vardas</param>

/// <param name="eil">Paraštė</param>

static void PrintData(string fv, IEnumerable A, string eil)

{

using (var frr = new StreamWriter(fv, true))

{

string bruk = new string('-', 55);

frr.WriteLine(eil);

frr.WriteLine(bruk);

frr.WriteLine(" Pavardė Gatvė Namas Nr. Tel.Nr.");

frr.WriteLine(bruk);

foreach (var one in A)

frr.WriteLine(one);

frr.WriteLine(bruk);

frr.WriteLine();

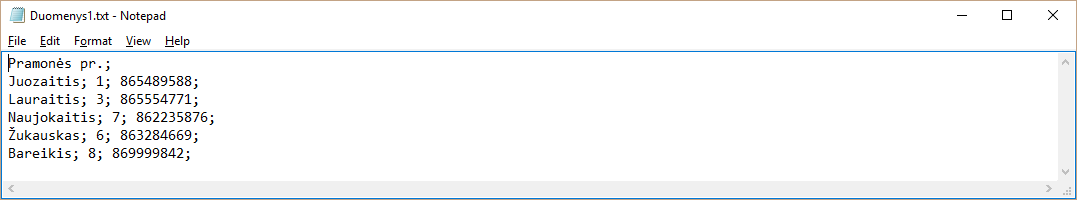
}

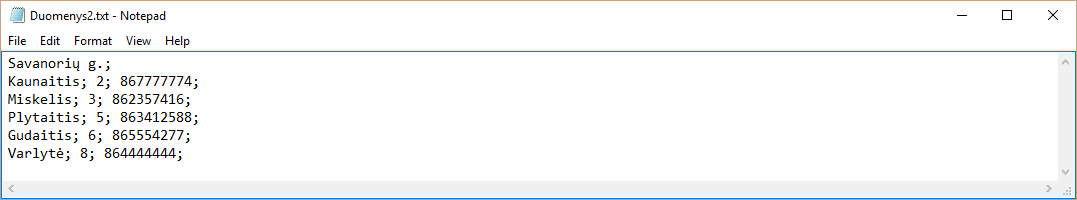
}

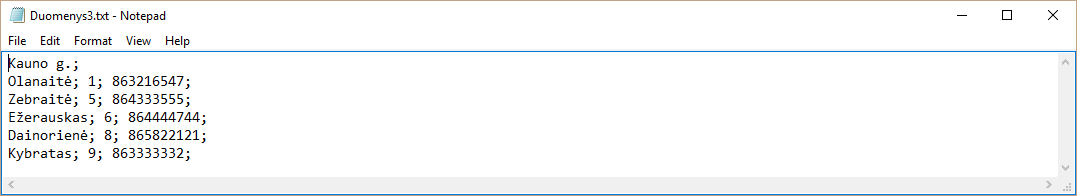
}

## Pradiniai duomenys ir rezultatai

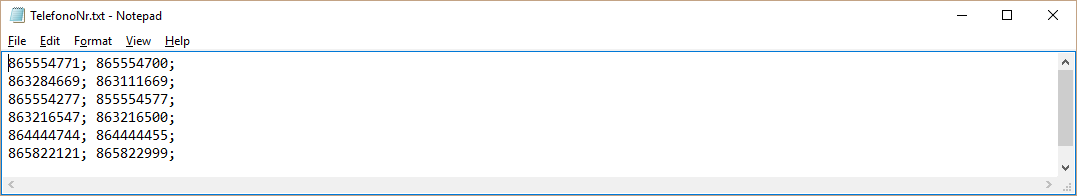
Pradiniai failo duomenys:



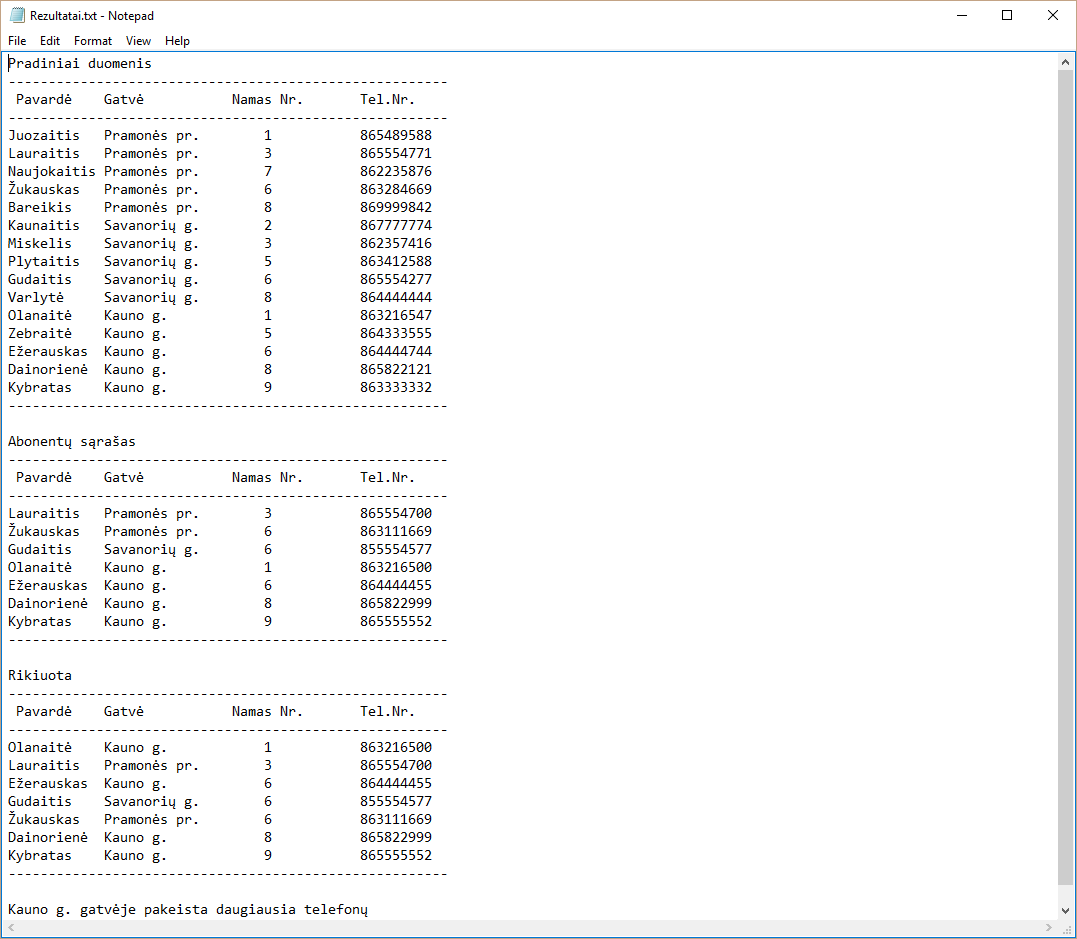




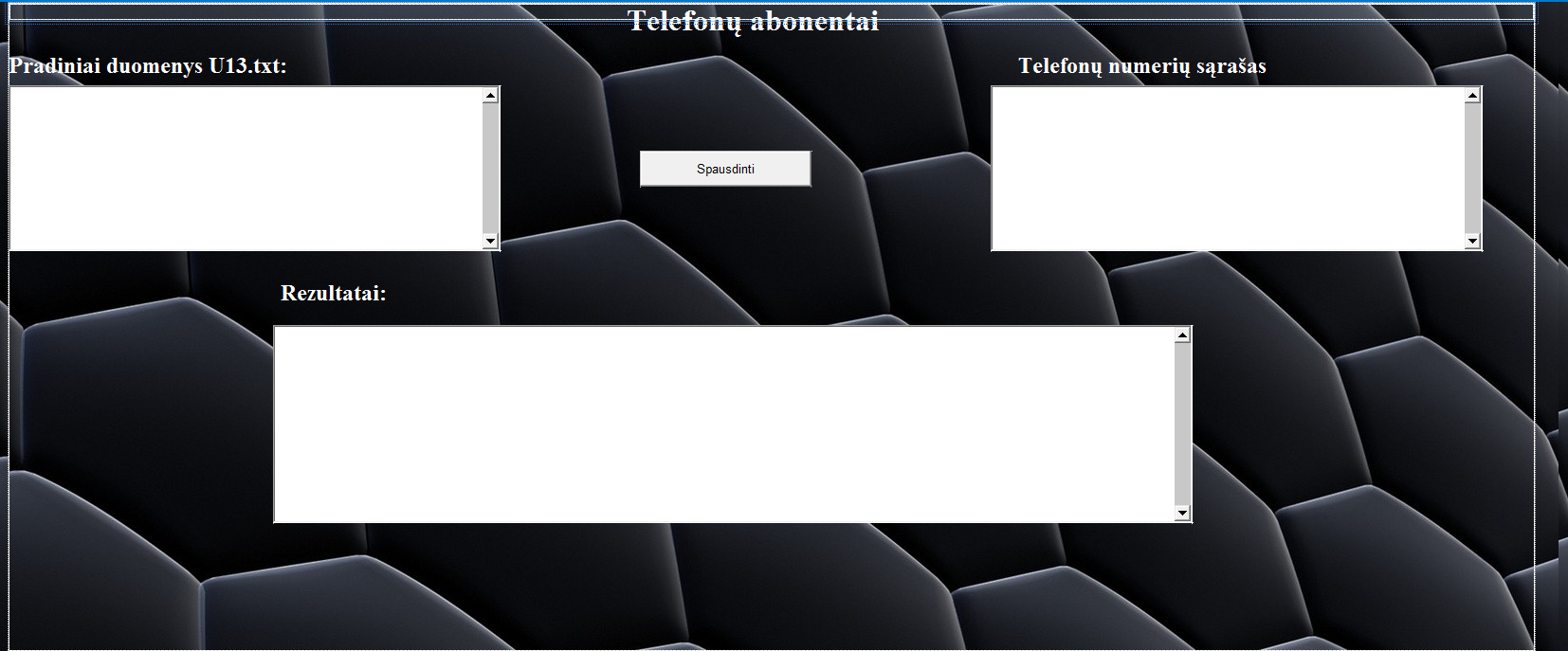
Pradiniai TelefonoNr.txt duomenys:



Rezultatai.txt



## Grafinės vartotojo sąsajos schema



Label4

Label1

Label2

Button1

TextBox1

TextBox3

Label3

TextBox2

## Grafinės vartotojo sąsajos elementų pakeistos savybės

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Komponentas | ID | Text | On\_Click | Font - Blod | Font - Size | ForeColor | TextMode |
| Label | **Label1** | **Telefonų abonentai** |  | **True** | **XX-Large** | **White** |  |
| Label | **Label2** | **Pradiniai failo duomenys** |  | **True** | **X-Large** | **White** |  |
| Label | **Label3** | **Rezultatai:** |  | **True** | **X - Large** | **White** |  |
| Label | **Label4** | **Telefonų numeris.txt** |  | **True** | **X - Large** | **White** |  |
| TextBox | **TextBox1** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox2** |  |  |  |  |  | **MultLine** |
| TextBox | **TextBox3** |  |  |  |  |  | **MultLine** |
| Button | **Button1** | **Spausdinti** | **Button1\_Click** |  |  |  |  |

## Klasių diagramos

|  |
| --- |
| **Resident** |
| **+ street: string**  **+ name: string**  **+ homeNr: integer**  **+ telNr: integer** |
| **+ Resident()**  **+ ToString() {query}** |

|  |
| --- |
| **TelNumber** |
| **+ oldTelNr: integer**  **+ newTelNr: integer** |
| **+ TelNumber()**  **+ ToString() {query}** |

## Programos vartotojo vadovas, kur pateikiamas tipinis darbo scenarijus

Programos darbiniame aplanke sukuriamas failo katalogas ir tekstinis duomenens failas TelefonoNr.txt. Failo katalogo viduje yra skirtingi 3 tekstiniai duomenų failai užpildyti informacija apie gyventojus: 1 – oje eilutėje yra gatvės pavadinimas, kiekvienoje kitoje eilutėje – pavardė, namo nr., telefono nr. Tekstiniame faile TelefonoNr.txt užpildyta informacija apie keičiamus telefono numerius: senas nr., naujas nr.. Eilutėje vienus duomenis nuo kitų skiria vienas tarpas. Užpildžius duomenų failus, galima įjungti programą. Įjungus programą, užsikrauna duomenys, juos matome ekrane. Tarp duomenų tekstinių laukų matomas ir mygtukas „Spausdinti“. Paspaudus mygtuką, programa atlieka skaičiavimus, rezultatai matomi naujame tekstiniame lauke: atvirkštinis gyventojų sąrašas, abonentų sąrašas, rikiuotas abonentų sąrašas, kiek gatvėje pakeista daugiausia telefonų.