KAUNO TECHNOLOGIJOS UNIVERSITETAS INFORMATIKOS FAKULTETAS

OBJEKTINIS PROGRAMAVIMAS II (P175B123)

Laboratorinio darbo ataskaita

Atliko:

IFF-5/7 gr. Studentė

Viktorija Ražaitė 2016 m. vasario 24 d.

Priėmė:

Jurgis Pralgauskis

KAUNAS 2016

TURINYS

1. Rekursija

1.1. Darbo užduotis

LD_7. Žodžių paieška.

Parašykite programą, kuri perskaitytų iš tekstinio failo "Trecias.txt" tekstą po vieną simbolį (1 < simbolių kiekis <= 2000) ir jais užpildytų masyvą A[n,n]. Į masyvą nerašomi eilutės pabaigos, naujos eilutės ir failo pabaigos simboliai. n parenkamas toks mažiausias, kad tekstas tilptų į kvadratinę matricą. Jei paskutinei eilutei trūksta simbolių, užpildote tarpais. Po to programa iš tekstinio failo "Zodziai.txt", kuriame kiekvienas žodis yra atskiroje eilutėje ir prasideda nuo 1 pozicijos, perskaito eilinį žodį, kurio ilgis k <= n/2. Reikia nustatyti, kiek kartų šis žodis kartojasi lentelėje A[n, n]. Paieška turėtų būti atliekama horizontaliai iš kairės į dešinę, vertikaliai iš viršaus žemyn ir pagal dešinę diagonalę žemyn ir į dešinę (ne tik pagrindinė diagonalė). Žodžiai iš eilutės (stulpelio) į eilutę (stulpelį) nekeliami. Žodžiai nepersidengia, bet trumpesnis žodis gali būti ilgesniojo žodžio dalimi.

Ekrane atspausdinkite parinktą n reikšmę ir kiekvieno žodžio pasikartojimų skaičių.

Trecias.txt	Matrica								
Berzas,sula;;sula;;klevu saldial lapasula	B	e	r	z	a	s	,	s	u
a aula, ar suart zemes vaikai du	l	a	;	;	s	u	l	a	;
Zodziai.txt	;	k	l	e	v	u		s	a
Sula	l	d	i	a	l		l	a	p
Alus	a	s	u	l	a			a	
Atsakymas			a	u	l	a	,		a
n = 9	r			s	u	a	r	t	
sula 3	z	e	m	e	s		v	a	i
alus 2	k	a	i		d	u			

1.2. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.IO; using
System.Linq; using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

publicpartialclassFormL1 : System.Web.UI.Page
{
publicclassSimboliai
    {
publicchar s { get; set; }

public Simboliai(char s)
        {
this.s = s;
        }
}
publicclassKonteinerine
    {
constint Max = 45; // max stulpeliu ir
eiluciuprivateSimboliai[,] Simb;          public
Konteinerine()
        {
            Simb = newSimboliai[Max, Max];
        }           publicvoid Deti(int i, int j,
Simboliai a)          {
            Simb[i, j] = a;
        }           publicSimboliai Imti(int
i, int j)
        {
return Simb[i, j];
```

```
        }
}
constint MaxZodziu = 2000;
constint MaxMatrica = 45;
privateconststring f1 = @"F:\2tras\objektinis\L1\Trecias.txt";
privateconststring f2 = @"F:\2tras\objektinis\L1\Zodziai.txt";
privateconststring f3 = @"F:\2tras\objektinis\L1\Rezultatai.txt";
//Konteinerine simbol = new Konteinerine();//dvimatis
//string[,] Masyvas = new string[MaxMatrica, MaxMatrica];

protectedvoid Page_Load(object sender, EventArgs e)
    {

}
protectedvoid TextBox1_TextChanged(object sender, EventArgs e)     {

    }
///<summary>
/// Skaiciuoja, kiek zodziu yra vertikaliai
///</summary>
///<param name="zodziuC">kiek yra duotuju zodziu</param>
///<param name="zodziuMasyvas">duotuju zodziu masyvas</param>
///<param name="simboliai">simboliu konteineris</param>
///<param name="n">matricos dydis n*n</param>
///<param name="ZodziuSkaicius">Rastu zodziu skaicius</param>
///<param name="i">parametras, kuris nurodo, kuris zodis bus
ieskomas</param>staticvoid RecursionHORIZONTALE(int zodziuC, string[]
zodziuMasyvas,
Konteinerine simboliai, int n, refint[] ZodziuSkaicius, int i)
    {          char[] zodRaides =
zodziuMasyvas[i].ToCharArray();         for (int k = 0; k
< n; k++) //stulpeliai ||
        {
bool tikrinaH = true;             int
nuoKurio = 0;

for (int l = 0; l < n; l++) //eilutes --
            {
if (zodRaides.Length <= (n - l)) // patikrina ar tas zodis tilps iki eilutes
pabaigos
if ((char.ToLower(zodRaides[0])) == char.ToLower(simboliai.Imti(l, k).s))
                    {
                        tikrinaH = true;
for (int m1 = 1; m1 < zodziuMasyvas[i].Length; m1++)
                        {
if ((m1 + l <= n) && (tikrinaH))
                            {
if ((char.ToLower(zodRaides[m1])) == char.ToLower(simboliai.Imti(l +
m1, k).s))                                          nuoKurio = l + m1;
else tikrinaH = false;
                            }
}

if (tikrinaH) ZodziuSkaicius[i] += 1;
                    }
            }
        }
if ((i + 1) < zodziuC)
            RecursionHORIZONTALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, i + 1);

    }
```

```
///<summary>
/// Skaiciuoja, kiek zodziu yra horizontaliai
///</summary>
///<param name="zodziuC">kiek yra duotuju zodziu</param>
///<param name="zodziuMasyvas">duotuju zodziu masyvas</param>
///<param name="simboliai">simboliu konteineris</param>
///<param name="n">matricos dydis n*n</param>
///<param name="ZodziuSkaicius">Rastu zodziu skaicius</param>
///<param name="i">parametras, kuris nurodo, kuris zodis bus
ieskomas</param>staticvoid RecursionVERTIKALE(int zodziuC, string[] zodziuMasyvas,
Konteinerine simboliai, int n, refint[] ZodziuSkaicius, int i)
    {
bool tikrinaV = true;         int
nuoKurioV = 0;
char[] zodRaides = zodziuMasyvas[i].ToCharArray();
for (int l = 0; l < n; l++) //stulpeliai ||
        {
for (int k = 0; k < n; k++) //eilutes --
            {
if (zodRaides.Length <= (n - k)) // patikrina ar tas zodis tilps iki stulpelio
pabaigos
if ((char.ToLower(zodRaides[0])) == char.ToLower(simboliai.Imti(l, k).s))
                    {
                        tikrinaV = true;
for (int m1 = 1; m1 < zodziuMasyvas[i].Length; m1++)
                        {
if ((m1 + k <= n) && (tikrinaV))
                            {
if ((char.ToLower(zodRaides[m1])) == char.ToLower(simboliai.Imti(l, k + m1).s))
                                    nuoKurioV = k + m1;
else tikrinaV = false;
                            }
}

if (tikrinaV) ZodziuSkaicius[i] += 1;
                    }
            }
}
if ((i + 1) < zodziuC)
            RecursionVERTIKALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, i + 1);

    }
///<summary>
/// Skaiciuoja, kiek zodziu yra pagal diagonale
///</summary>
///<param name="zodziuC">kiek yra duotuju zodziu</param>
///<param name="zodziuMasyvas">duotuju zodziu masyvas</param>
///<param name="simboliai">simboliu konteineris</param>
///<param name="n">matricos dydis n*n</param>
///<param name="ZodziuSkaicius">Rastu zodziu skaicius</param>
///<param name="i">parametras, kuris nurodo, kuris zodis bus ieskomas</param>
staticvoid RecursionDIAGONALE(int zodziuC, string[] zodziuMasyvas,
Konteinerine simboliai, int n, refint[] ZodziuSkaicius, int i)
    {
bool tikrinaD = true;         int
nuoKurioD = 0;
char[] zodRaides = zodziuMasyvas[i].ToCharArray();
for (int k = 0; k < n; k++) //stulpeliai ||         {
for (int l = 0; l < n; l++) //eilutes --
            {
if ((zodRaides.Length <= (n - l)) && (zodRaides.Length <= (n -
```

```
k))) // patikrina ar tas zodis tilps iki eilutes ir stulpelio pabaigosif
((char.ToLower(zodRaides[0])) == char.ToLower(simboliai.Imti(l, k).s))
                    {
                        tikrinaD = true;
for (int m1 = 1; m1 < zodziuMasyvas[i].Length; m1++)
                        {
if ((m1 + l <= n) && (tikrinaD) && (m1 + k <= n))
                            {
if ((char.ToLower(zodRaides[m1])) == char.ToLower(simboliai.Imti(l +
m1, k + m1).s))                                     nuoKurioD = l +
m1;                                 else tikrinaD = false;
                            }
}

if (tikrinaD) ZodziuSkaicius[i] += 1;
                    }
            }
}
if ((i + 1) < zodziuC)
            RecursionDIAGONALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, i + 1);

    }
///<summary>
/// Paspaudus mygtuka "start" bus atliekami veiksmai nurodyti siame void'e
///</summary>
protectedvoid Button1_Click(object sender, EventArgs e)
    {
        TextBox7.Text = " tarpas ";
Konteinerine simboliai = newKonteinerine();
string[] zodziuMasyvas = newstring[MaxZodziu];
int zodziuC;          int n;
        Skaityti(simboliai, zodziuMasyvas, out zodziuC, out n);
//-------surasys, kiek kartu buvo rastas kiekvienas zodisint[]
ZodziuSkaicius = newint[zodziuC];          for (int i = 0; i <
zodziuC; i++)
        {
            ZodziuSkaicius[i] = 0;
        }

//-----HORIZONTALIAIint iHoriz =
0;
        RecursionHORIZONTALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, iHoriz);



//--------------VERTIKALIAIint iVertik
= 0;
        RecursionVERTIKALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, iVertik);

//--------------PAGAL DIAGONALEint iDiogon
= 0;
        RecursionDIAGONALE(zodziuC, zodziuMasyvas, simboliai, n, ref
ZodziuSkaicius, iDiogon);

string eill = "kvadratine matrica: " + n + " x " + n + "\r\n";
for (int i = 0; i < zodziuC; i++)          {
            eill += "zodis '" + zodziuMasyvas[i] + "' pasikartoja " +
ZodziuSkaicius[i] + " kartus";
eill += "\r\n";
```

```
}
using (var writer = File.AppendText(f3))
        {
            writer.WriteLine();
            writer.WriteLine("Gauti rezultatai: ");
writer.WriteLine();             writer.WriteLine(eill);
        }
        TextBox6.Text = "Rezultatai:";
TextBox3.Text = eill;
    }

///<summary>
/// Skaito duomenis is duom failu
///</summary>
///<param name="simboliai">simboliu konteineris</param>
///<param name="zodziuMasyvas">duotuju zodziu masyvas</param>
///<param name="n">matricos dydis n*n</param>
///<param name="zodziuC">kiek yra duotuju zodziu</param>
publicvoid Skaityti(Konteinerine simboliai, string[] zodziuMasyvas, outint
zodziuC, outint nn)
    {
int MaxEil = 50;
string line;         int
simboliuSk = 0;
string[] eilutes = newstring[MaxEil];
int teksEil = 0; // kiek duotam faile yra eiluciu

using (StreamReader reader = newStreamReader(f1))
        {
while ((line = reader.ReadLine()) != null)
            {
                simboliuSk += line.Length;
eilutes[teksEil++] = line;
            }
        }
//------------randa kvadratines matricos n x ndouble n
= Math.Sqrt(simboliuSk);         n = Math.Round(n, 1);
if (n % Math.Round(n, 0) == 0) n = Math.Round(n, 0);
else         {
if (n % Math.Round(n, 0) >= 0.5) n = Math.Round(n, 0);             else
n = n = Math.Round(n, 0) + 1;
        }
//------------sudeda teksta i viena eilute
nn = Convert.ToInt32(n);         string failas =
"";
for (int i = 0; i < teksEil; i++)
        {
            failas = failas + eilutes[i];
        }
//------------slpit'ina kas simboli ir deda i kvadratine matricastring[] sym =
newstring[simboliuSk];

char[] symbol = failas.ToCharArray();

int count = 0; char a;         for (int jj
= 0; jj < n; jj++)             for (int ii
= 0; ii < n; ii++)
            {
if (count < failas.Length)
                    a = symbol[count];
else                     a = ' ';
```

```
Simboliai ss = newSimboliai(a);
simboliai.Deti(ii, jj, ss);                    count++;
            }

//-------------skaito antro failo zodzius          zodziuC
= 0;
string line1; string eilZod = "";
using (StreamReader reader = newStreamReader(f2))
        {
while ((line1 = reader.ReadLine()) != null)
            {
if (line1.Length <= n / 2)
zodziuMasyvas[zodziuC++] = line1;                  eilZod +=
line1 + "\r\n";
            }
        }          if
(File.Exists(f3))
File.Delete(f3);
//------------suraso pradinius duomenis i rezultatu failastring eil =
"";
using (var writer = File.AppendText(f3))
        {
            writer.WriteLine("Duotas tekstas: ");
writer.WriteLine();
for (int i = 0; i < teksEil; i++)
            {
                writer.WriteLine(eilutes[i]);
            }
            writer.WriteLine("-----------------------------------");
writer.WriteLine();
            writer.WriteLine("Zodziu      matrica
");                              writer.WriteLine();
for (int i = 0; i < n; i++)
            {
for (int j = 0; j < n; j++)
                {
                    writer.Write(" " + simboliai.Imti(j, i).s);
eil += " " + simboliai.Imti(j, i).s;
                }
                writer.WriteLine();
eil += "\r\n";
            }
            writer.WriteLine("-----------------------------------");
writer.WriteLine();
            writer.WriteLine("Duoti zodziai:");
writer.WriteLine();
for (int i = 0; i < zodziuC; i++)
            {
                writer.WriteLine(zodziuMasyvas[i]);
            }
            writer.WriteLine("-----------------------------------");
            TextBox1.Text = eil;
            TextBox2.Text = "Pradiniai duomenys";
            TextBox5.Text = "Duoti zodziai";
            TextBox4.Text = eilZod;
        }
    }
}
```

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="FormL1.aspx.cs"
Inherits="FormL1" %>
```

```
<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
</head>
<bodystyle="background-color: #ADDFFF;">
<formid="form1"runat="server">
<div>
<asp:ButtonID="Button1"runat="server"OnClick="Button1_Click"
Text="START"/>
</div>
<h1>LD1_7 ŽODŽIŲ PAIEŠKA</h1>
<p>

<asp:TextBoxID="TextBox2"runat="server"></asp:TextBox>

            &nbs
p;         
<asp:TextBoxID="TextBox5"runat="server"></asp:TextBox>
</p>
<p>
<asp:TextBoxID="TextBox1"runat="server"
OnTextChanged="TextBox1_TextChanged"Height="146px"TextMode="MultiLine"
Width="201px"></asp:TextBox>
            
<asp:TextBoxID="TextBox4"runat="server"
OnTextChanged="TextBox1_TextChanged"Height="146px"TextMode="MultiLine"
Width="193px"></asp:TextBox>
</p>
<p>
<asp:TextBoxID="TextBox6"runat="server"></asp:TextBox>
</p>
<p>
 <asp:TextBoxID="TextBox3"runat="server"Height="168px"
TextMode="MultiLine"Width="466px"></asp:TextBox>

            &nbs
p;    
<asp:TextBoxID="TextBox7"runat="server"></asp:TextBox></p>
<p>
 </p><p>
 </p>
</form>
</body>
</html>
```

1.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys

```
Failas: Trecias.txt

Berzas,sula;;sula;;klevu saldial lapasula
a   aula, ar  suart zemes vaikai du

Failas:  Zodziai.txt
```

```
Sula
Alus
```

Rezultatai

```
Failas: Rezultatai.txt

Duotas tekstas:

Berzas,sula;;sula;;klevu saldial lapasula
a   aula, ar  suart zemes vaikai du --------
--------------------------

Zodziu matrica

 B e r z a s , s u
l a ; ; s u l a ;  ;
k l e v u   s a  l d
i a l   l a p  a s u
l a     a       a u
l a ,   a  r     s u
a r t    z e m e s
v a i  k a i   d u
---------------------------------

Duoti zodziai:

Sula
Alus
---------------------------------

Gauti rezultatai:
 kvadratine matrica: 9 x 9 zodis
'Sula' pasikartoja 3 kartus zodis
'Alus' pasikartoja 2 kartus
```

1.4. Grafinės vartotojo sąsajos schema

1.5. Grafinės vartotojo sąsajos elementų pakeistos savybės

Komponentas	ID	text	Back Color	OnClick	TextMode
Button	Button1	START	#ADDFFF	Button1_Click	
TextBox	TextBox1		#FFCCFF		MultiLine
TextBox	TextBox2				
TextBox	TextBox3				
TextBox	TextBox4				
TextBox	TextBox5		#FFCCFF		MultiLine
TextBox	TextBox6				

1.6. Klasių diagramos

Veiksmai
+RecursionHORIZONTALE(in zodziuC:integer, in zodziuMasyvas:string[], in simboliai:Konteinerine, in n:integer, inout ZodziuSkaicius:integer[], in i:integer) +RecursionVERTIKALE(in zodziuC:integer, in zodziuMasyvas:string[], in simboliai:Konteinerine, in n:integer, inout ZodziuSkaicius:integer[], in i:integer) +RecursionDIAGONALE(in zodziuC:integer, in zodziuMasyvas:string[], in simboliai:Konteinerine, in n:integer, inout ZodziuSkaicius:integer[], in i:integer) +Page_Load(in sender:object, in e:EventArgs) +Page_Init(in sender:object, in e: EventArgs) +TextBox1_TextChanged(in sender:object, in e: EventArgs) +TextBox2_TextChanged(in sender:object, in e: EventArgs) +Button1_Click(in sender:object, in e: EventArgs)

Skaityti
+ Skaityti(){query}

1.7. Programos vartotojo vadovas

Sukuriamas tekstinis duomenų failas Trecias.txt ir paršomas tekstas. Tekstas surašomas į nxn matricą, n parenkamos toks mažiausias, kad tekstas tilptų į kvadratinę matricą, jei trūksta simbolių, užpildoma tarpais. Užpildžius duomenų failą, galima įjungti programą. Įjungę programą paspaudžiame mygtuką „START", kuris į ekraną išveda pradinius duomenis skiltyje „Pradiniai duomenys" ir išveda rezultatus skiltyje „Rezultatai". Baigę savo darbą, galime išeiti iš programos spausdami raudoną x mygtuką dešiniam viršutiniame programos kampe. Įvykdžius programą, taip pat sukuriamas tekstinis rezultatų failas Rezultatai.txt.

2. Susietasis sąrašas

2.1. Darbo užduotis

LD_7. Moduliai.
Studentai renkasi modulius. Už modulius yra atsakingi dėstytojai. Dėstytojas gali būti atsakingas už keletą modulių.

Suraskite, kuris dėstytojas turi daugiausiai pasirinktų modulių.
Nustatykite, ar visų grupių studentai pasirinko šio dėstytojo modulius.
Atspausdinkite grupes, kurių studentai nepasirinko šio dėstytojo modulių.

Duomenys:

- Tekstiniame faile U7a.txt duota informacija apie studentų pasirenkamus modulius: modulio pavadinimas, studento pavardė, vardas, grupė.
- Tekstiniame faile U7b.txt duota informacija apie modulius: modulio pavadinimas, atsakingo dėstytojo pavardė, vardas, kreditų kiekis.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka.
Sudarykite nurodyto modulio (įvedamas klaviatūra) pasirinkusių studentų sąrašą.

2.2. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

publicpartialclassL2 : System.Web.UI.Page
{
publicclassStudentas
    {
publicstring modulis { get; set; }
publicstring pavarde { get; set; }
publicstring vardas { get; set; }
publicstring grupe { get; set; }
public Studentas(string modulis, string pavarde, string vardas, string grupe)
        {
this.modulis = modulis;
this.pavarde = pavarde;
this.vardas = vardas;
this.grupe = grupe;
        }
staticpublicbooloperator>(Studentas pir, Studentas ant)
        {
int ip = String.Compare(pir.grupe, ant.grupe, StringComparison.CurrentCulture);
int ip1 = String.Compare(pir.pavarde, ant.pavarde,
StringComparison.CurrentCulture);
int ip2 = String.Compare(pir.vardas, ant.vardas, StringComparison.CurrentCulture);
return ip > 0 || ip == 0 && ip1 > 0 || ip1 == 0 && ip2 > 0;
        }
staticpublicbooloperator<(Studentas pir, Studentas ant)
        {
int ip = String.Compare(pir.grupe, ant.grupe, StringComparison.CurrentCulture);
int ip1 = String.Compare(pir.pavarde, ant.pavarde,
StringComparison.CurrentCulture);
int ip2 = String.Compare(pir.vardas, ant.vardas, StringComparison.CurrentCulture);
return ip < 0 || ip == 0 && ip1 < 0 || ip1 == 0 && ip2 < 0;
        }
publicoverridestring ToString()
```

```
        {
string eilute;
            eilute = string.Format("{0,-15} {1,-20} {2, -15} {3,-10}", modulis,
pavarde, vardas, grupe);
return eilute;
        }
    }

publicclassDestytojas
    {
publicstring modulis { get; set; }
publicstring pavarde { get; set; }
publicstring vardas { get; set; }
publicint kreditai { get; set; }
public Destytojas(string modulis, string pavarde, string vardas, int kreditai)
        {
this.modulis = modulis;
this.pavarde = pavarde;
this.vardas = vardas;
this.kreditai = kreditai;
        }
publicstring ToString1()
        {
string eilute;
            eilute = string.Format("{0,-15} {1,-15} {2, -15} {3,-10}", modulis,
pavarde, vardas, kreditai);
return eilute;
        }
    }

publicsealedclassMazgas1
    {
publicStudentas DuomStud { get; set; }
publicMazgas1 Kitas { get; set; }
public Mazgas1(Studentas st, Mazgas1 adr)
        {
            DuomStud = st;
            Kitas = adr;
        }
    }

publicsealedclassMazgas2
    {
publicDestytojas DuomDest { get; set; }
publicMazgas2 Kitas { get; set; }
public Mazgas2(Destytojas ds, Mazgas2 adr)
        {
            DuomDest = ds;
            Kitas = adr;
        }
    }

publicsealedclassSarasas1//Studentu sarasas
    {
privateMazgas1 pr; //saraso pradzia
privateMazgas1 pb; //saraso pabaiga
privateMazgas1 d; //saraso sasajai
public Sarasas1()
        {
this.pr = null;
this.pb = null;
this.d = null;
        }
publicStudentas ImtiDuomenis()
        {
return d.DuomStud;
```

```
        }
publicMazgas1 GautiPirma()
        {
return pr;
        }
publicMazgas1 GautiPaskutini()
        {
return pb;
        }
publicvoid DetiDuomenisT(Studentas naujas)
        {
var dd = newMazgas1(naujas, null);
if (pr != null)
            {
                pb.Kitas = dd;
                pb = dd;
            }
else
            {
                pr = dd;
                pb = dd;
            }
        }
    }

publicsealedclassSarasas2//Destytoju sarasas
    {
privateMazgas2 pr; //saraso pradzia
privateMazgas2 pb; //saraso pabaiga
privateMazgas2 d; //saraso sasajai
public Sarasas2()
        {
this.pr = null;
this.pb = null;
this.d = null;
        }
publicDestytojas ImtiDuomenis()
        {
return d.DuomDest;
        }
publicMazgas2 GautiPirma()
        {
return pr;
        }
publicMazgas2 GautiPaskutini()
        {
return pb;
        }
publicvoid DetiDuomenis2T(Destytojas naujas)
        {
var dd = newMazgas2(naujas, null);
if (pr != null)
            {
                pb.Kitas = dd;
                pb = dd;
            }
else
            {
                pr = dd;
                pb = dd;
            }
        }
    }

protectedvoid Page_Load(object sender, EventArgs e)
    { }
```

```
Sarasas1 Studentai = newSarasas1();
Sarasas2 Destytojai = newSarasas2();

protectedvoid Button1_Click(object sender, EventArgs e)
    {
string CD1 = @"F:\\2tras\\objektinis\\L2\\U7a.txt"; //studentu duomenu failas
string CD2 = @"F:\\2tras\\objektinis\\L2\\U7b.txt"; //destytoju duomenu failas
string CR = @"F:\\2tras\\objektinis\\L2\\REZ.txt"; //rezultatu duomenu failas
if (File.Exists(CR))
File.Delete(CR);
        SkaitytiStud(CD1, Studentai);
        SkaitytiDest(CD2, Destytojai);
string destytojasMaxVardas; string destytojasMaxPavarde;//destytojas turintis
daugiausiai moduliu
        DestytojasMaxMod(Studentai, Destytojai, out destytojasMaxVardas, out
destytojasMaxPavarde);
        SpausdintiRezultatus(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde, CR);
        GrupesKurNepasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde, CR);

    }
///<summary>
/// Nuskaito studentu duomenu faila
///</summary>
///<param name="fv">studentu failas</param>
///<param name="Studentai">Studentu sarasas</param>
staticvoid SkaitytiStud(string fv, Sarasas1 Studentai)
    {
string line;
using (var reader = newStreamReader(fv))
        {
while ((line = reader.ReadLine()) != null)
            {
var v = line.Split(' ');
var student = newStudentas(v[0], v[1], v[2], v[3]);
                Studentai.DetiDuomenisT(student);
            }
        }
    }
///<summary>
/// Nuskaito destytoju duomenu faila
///</summary>
///<param name="fv">destytoju failas</param>
///<param name="Destytojai">Destytoju sarasas</param>
staticvoid SkaitytiDest(string fv, Sarasas2 Destytojai)
    {
string line;
using (var reader = newStreamReader(fv))
        {
while ((line = reader.ReadLine()) != null)
            {
var v = line.Split(' ');
var destyt = newDestytojas(v[0], v[1], v[2], int.Parse(v[3]));
                Destytojai.DetiDuomenis2T(destyt);
            }
        }
    }
///<summary>
/// suranda destytoja, turinti daugiausiai moduliu
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
```

```
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
staticvoid DestytojasMaxMod(Sarasas1 Studentai, Sarasas2 Destytojai, outstring
destytojasMaxVardas, outstring destytojasMaxPavarde)
    {
int max = 0;
        destytojasMaxVardas = null;
        destytojasMaxPavarde = null;
for (Mazgas2 g = Destytojai.GautiPirma(); g != null; g = g.Kitas)
        {
int laik = 0;
for (Mazgas1 d = Studentai.GautiPirma(); d != null; d = d.Kitas)
            {
if (g.DuomDest.modulis == d.DuomStud.modulis)
                {
                    laik += 1;
                }
            }
if (max < laik)
            {
                destytojasMaxVardas = g.DuomDest.vardas;
                destytojasMaxPavarde = g.DuomDest.pavarde;
                max = laik;
            }
        }
    }
///<summary>
/// Suranda ar visu grupiu studentai pasirinko sio destytojo modulius
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<returns></returns>
staticbool ArVisuPasirinko(Sarasas1 Studentai, Sarasas2 Destytojai, string
destytojasMaxVardas, string destytojasMaxPavarde)
    {
for (Mazgas2 g = Destytojai.GautiPirma(); g != null; g = g.Kitas)
        {
if ((destytojasMaxVardas == g.DuomDest.vardas) && (destytojasMaxPavarde ==
g.DuomDest.pavarde))
            {
string modulis = g.DuomDest.modulis;
for (Mazgas1 d = Studentai.GautiPirma(); d != null; d = d.Kitas)
                {
if (modulis == d.DuomStud.modulis)
                    {
int visi = 0; int ne = 0;
for (Mazgas1 d2 = d; d2 != null; d2 = d2.Kitas)
                        {
if (d.DuomStud.grupe == d2.DuomStud.grupe)
                            {
                                visi += 1;
if (d2.DuomStud.modulis != modulis)
                                    ne += 1;
                            }
                        }
if (ne == visi)
returnfalse;
                    }
                }
            }
        }
returntrue;
```

```
    }
///<summary>
/// suranda grupes, kuriu bent vienas studentas nepasirinko sio destytojo modulio
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<param name="fr">rezultatu failas</param>
publicvoid GrupesKurNepasirinko(Sarasas1 Studentai, Sarasas2 Destytojai, string
destytojasMaxVardas, string destytojasMaxPavarde, string fr)
    {
Sarasas1 StudentaiGR = newSarasas1();
        StudentaiGR = Studentai;
//rusiuoja pagal grupes, pavardes ir vardus

//jei grupej yra nors vienas nepasirinkes modulio, atspauszinti
//jei visa grupe pasirinkusi ta moduli, pasalinti grupe

for (Mazgas2 g = Destytojai.GautiPirma(); g != null; g = g.Kitas)
        {
if ((destytojasMaxVardas == g.DuomDest.vardas) && (destytojasMaxPavarde ==
g.DuomDest.pavarde))
            {
string modulis = g.DuomDest.modulis;
for (Mazgas1 d = Studentai.GautiPirma(); d != null; d = d.Kitas)
                {
if (modulis == d.DuomStud.modulis)
                    {
int visi = 0; int ne = 0; string grupe = "";
for (Mazgas1 d2 = d; d2 != null; d2 = d2.Kitas)
                        {
if (d.DuomStud.grupe == d2.DuomStud.grupe)
                            {
                                visi += 1;
if (d2.DuomStud.modulis == modulis)
                                {
                                    ne += 1;
                                    grupe = d.DuomStud.grupe;
                                }


                            }
                        }
if (ne == visi)
                        {
                            SalintiGrupe(StudentaiGR, grupe);
                        }
                    }
                }
            }
        }
        Rikiuoti(StudentaiGR);
        SpausdintiGrupes(StudentaiGR, fr);
    }
///<summary>
/// Surikiuoja nauja studentu sarasas pagal grupe, pavarde ir varda
///</summary>
///<param name="StudentaiGR">naujas studentu sarasas</param>
staticvoid Rikiuoti(Sarasas1 StudentaiGR)
    {
for (Mazgas1 d1 = StudentaiGR.GautiPirma(); d1 != null; d1 = d1.Kitas)
        {
Mazgas1 max = d1;
```

```
for (Mazgas1 d2 = d1; d2 != null; d2 = d2.Kitas)
            {
if (d2.DuomStud < max.DuomStud)
                    max = d2;
Studentas stud = d1.DuomStud;
                d1.DuomStud = max.DuomStud;
                max.DuomStud = stud;
            }
        }
    }
///<summary>
/// pasalina grupes, kuriu visi studentai pasirinke sio destytojo modulius
///</summary>
///<param name="StudentaiGR">Surikiuotu studentu sarasas</param>
///<param name="grupe">grupe, kuria reikia pasalinti</param>
staticvoid SalintiGrupe(Sarasas1 StudentaiGR, string grupe)
    {
for (Mazgas1 d = StudentaiGR.GautiPirma(); d != null; d = d.Kitas)
        {
if (d.DuomStud.grupe == grupe)
            {
if (d.Kitas != null)
                {
Mazgas1 s = d.Kitas;
                    d.DuomStud = s.DuomStud;
                    d.Kitas = s.Kitas;
                }
else
                    PasalintiPaskutini(StudentaiGR);
            }
        }
    }
///<summary>
/// pasalina paskutini elemanta
///</summary>
///<param name="StudentaiGR">surikiuotu studentu sarasas</param>
staticvoid PasalintiPaskutini(Sarasas1 StudentaiGR)
    {
Mazgas1 k = StudentaiGR.GautiPirma();
for (Mazgas1 d = StudentaiGR.GautiPirma(); d != null; d = d.Kitas)
        {
            k = d.Kitas;
if(k.Kitas == null)
            {
                d.Kitas = null;
                StudentaiGR.DetiDuomenisT(d.DuomStud);
            }
        }
    }
///<summary>
/// spausdinti rezultatus
///</summary>
///<param name="StudentaiGR">surikiuotu studentu sarasas</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiGrupes(Sarasas1 StudentaiGR, string fr)
    {
string line = "";
using (var writer = File.AppendText(fr))
        {

            writer.WriteLine();
            writer.WriteLine("Grupes, kuriu studentai nepasirinko sio destytojo
moduliu:");
            Label5.Text = "Grupes, kuriu studentai nepasirinko sio destytojo
moduliu:";
```

```
            line += "Modulis         Pavarde              Vardas          Grupe" +
"\r\n" + "\r\n";
for (Mazgas1 d = StudentaiGR.GautiPirma(); d != null; d= d.Kitas)
            {
                writer.WriteLine(d.DuomStud.ToString());
                line += d.DuomStud.ToString() + "\r\n";
            }
        }
        TextBox3.Text = line;
    }
protectedvoid Button2_Click(object sender, EventArgs e)
    {
string CD1 = @"F:\\2tras\\objektinis\\L2\\U7a.txt"; //studentu duomenu failas
string CR = @"F:\\2tras\\objektinis\\L2\\REZ.txt"; //rezultatu duomenu failas
        SkaitytiStud(CD1, Studentai);
        Label4.Text = "Iveskite modulio pavadinima ir dar karta paspauskite
antraji 'START' mygtuka";
string modulisVIP = TextBox1.Text;
Sarasas1 NewStudentai = newSarasas1();
        SudarytiStudentuSarasa(NewStudentai, modulisVIP, Studentai);
if(modulisVIP != "")
        SpausdintiRez2(NewStudentai, modulisVIP, CR);
    }
///<summary>
/// sudaromas studentu sarasas pagal ivesta klaviatura moduli
///</summary>
///<param name="NewStudentai">naujas studentu sarasas</param>
///<param name="modulisVIP">Ivestas modulis</param>
///<param name="Studentai">studentu sarasas</param>
publicvoid SudarytiStudentuSarasa(Sarasas1 NewStudentai, string modulisVIP,
Sarasas1 Studentai)
    {
string eil = "";
if (modulisVIP != "")
        {
for (Mazgas1 d = Studentai.GautiPirma(); d != null; d = d.Kitas)
            {
if (d.DuomStud.modulis == modulisVIP)
                {
                    NewStudentai.DetiDuomenisT(d.DuomStud);
                    eil += d.DuomStud + "\r\n";
                }
            }
        }
else
            TextBox2.Text = "Neteisingas modulio pavadinimas";
        TextBox2.Text = eil;
    }
///<summary>
/// spausdina gautus rezultatus
///</summary>
///<param name="Studentai">studentu sarasas</param>
///<param name="Destytojai">destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiRezultatus(Sarasas1 Studentai, Sarasas2 Destytojai, string
destytojasMaxVardas, string destytojasMaxPavarde, string fr)
    {
using (var writer = File.AppendText(fr))
        {
//--- pradiniu doumenu spausdinimas
            writer.WriteLine("--------------PRADINIAI-DUOMENYS-----------------
");
```

```
            writer.WriteLine();
            writer.WriteLine("---------Studentu-duomenu-failas--------");
for (Mazgas1 d = Studentai.GautiPirma(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.DuomStud.ToString());
            }
            writer.WriteLine();
            writer.WriteLine("---------Destytoju-duomenu-failas--------");
for (Mazgas2 d = Destytojai.GautiPirma(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.DuomDest.ToString1());
            }
            writer.WriteLine("-----------------------------------------------");
            writer.WriteLine();
//---------
            writer.WriteLine();
            writer.WriteLine("Daugiausiai pasirinktu moduliu turi destytojas - {0}
{1}", destytojasMaxVardas, destytojasMaxPavarde);
            Label1.Text = "Daugiausiai pasirinktu moduliu turi destytojas - " +
destytojasMaxVardas + " " + destytojasMaxPavarde;
            writer.WriteLine();
if (ArVisuPasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde))
            {
                writer.WriteLine("Nevisu grupiu studentai pasirinko sio destytojo
modulius.");
                Label2.Text = "Visu grupiu studentai pasirinko sio destytojo
modulius.";
            }
if (!ArVisuPasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde))
            {
                writer.WriteLine("Ne visu grupiu studentai pasirinko sio destytojo
modulius.");
                Label2.Text = "Ne visu grupiu studentai pasirinko sio destytojo
modulius.";
            }
        }
    }
///<summary>
/// spausdinami rezultatai po ivedamo modulio klaviatura
///</summary>
///<param name="NewStudentai">naujas studentu sarasas</param>
///<param name="modulisVIP">Ivestas modulis</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiRez2(Sarasas1 NewStudentai, string modulisVIP, string fr)
    {
using (var writer = File.AppendText(fr))
        {
            writer.WriteLine();
            writer.WriteLine("-----------------------------------------------");
            writer.WriteLine("Ivestas modulio pavadinimas: {0}", modulisVIP);
            writer.WriteLine("Sio modulio studentu sarasas:");
for (Mazgas1 d = NewStudentai.GautiPirma(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.DuomStud.ToString());
            }
        }
    }
}
```

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="L2.aspx.cs"Inherits="L2"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
```

```
<title></title>
<styletype="text/css">
#form1 {
height: 816px;
        }
</style>
</head>
<bodystyle="background-color:#ADDFFF;">
<formid="form1"runat="server">
<div>

</div>
<asp:ButtonID="Button1"runat="server"OnClick="Button1_Click"Text="START"BackColor=
"Blue"BorderColor="White"/>
<br/>
<asp:LabelID="Label1"runat="server"Text="Label"></asp:Label>
 
<br/>
<asp:LabelID="Label2"runat="server"Text="Label"></asp:Label>
 
<br/>
<br/>
<asp:LabelID="Label5"runat="server"Text="Label"></asp:Label>
<br/>
<asp:TextBoxID="TextBox3"runat="server"Height="231px"Width="556px"BackColor="#FFCC
FF"TextMode="MultiLine"></asp:TextBox>
<br/>
<br/>
<asp:ButtonID="Button2"runat="server"OnClick="Button2_Click"Text="START"BackColor=
"Blue"/>
<br/>
<asp:LabelID="Label4"runat="server"Text="Label"></asp:Label>
 <p>
<asp:TextBoxID="TextBox1"runat="server"Height="30px"Width="187px"BorderColor="#FFC
CFF"TextMode="MultiLine"></asp:TextBox>
</p>
<p>
<asp:TextBoxID="TextBox2"runat="server"Height="144px"Width="563px"BackColor="#FFCC
FF"TextMode="MultiLine"></asp:TextBox>
</p>
</form>
</body>
</html>
```

2.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys

```
Failas: U7a.txt

Programavimas Tomaitis Tomas IFF-5/4
Matematika Pjuklaite Ieva IFF-5/2
Psichologija Puodzius Jonas IFF-5/3
Programavimas Lygiauskas Vladimiras IFF-5/1
Programavimas Aiseviciute Aiste IFF-5/4
Medijos Pukutis Pukis IFF-5/5
Programavimas Stasys Girnaitis IFF-5/4
Programavimas Gaile Pagalvyte IFF-5/2
Medijos Ignas Lovauskas IFF-5/2
Programavimas Diana Paveikslaite IFF-5/1
Matematika Dainius Lentauskas IFF-5/2
Programavimas Vytenis Deziauskas IFF-5/3
Psichologija Raigardas Knygius IFF-5/1
Programavimas Giedrius Palangiauskas IFF-5/4
```

```
Psichologija Audrius Dalgiauskas IFF-5/3
Programavimas Jomante Deklaityte IFF-5/4
Matematika Antanas Stiklius IFF-5/3

Failas: U7b.txt

Programavimas Jonas Jonaitis 6
Medijos Arnas Sofauskas 5
Informacines Radvile Stalciukaite 7
Matematika Gintaras Grebliauskas 8
Skaitmenine Arnas Sofauskas 4
Psichologija Radvile Stalciukaite 7
```

Rezultatai

```
Failas: REZ.txt

--------------PRADINIAI-DUOMENYS-----------------

---------Studentu-duomenu-failas--------
Programavimas   Tomaitis            Tomas           IFF-5/4
Matematika      Pjuklaite           Ieva            IFF-5/2
Psichologija    Puodzius            Jonas           IFF-5/3
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Programavimas   Aiseviciute         Aiste           IFF-5/4
Medijos         Pukutis             Pukis           IFF-5/5
Programavimas   Stasys              Girnaitis       IFF-5/4
Programavimas   Gaile               Pagalvyte       IFF-5/2
Medijos         Ignas               Lovauskas       IFF-5/2
Programavimas   Diana               Paveikslaite    IFF-5/1
Matematika      Dainius             Lentauskas      IFF-5/2
Programavimas   Vytenis             Deziauskas      IFF-5/3
Psichologija    Raigardas           Knygius         IFF-5/1
Programavimas   Giedrius            Palangiauskas   IFF-5/4
Psichologija    Audrius             Dalgiauskas     IFF-5/3
Programavimas   Jomante             Deklaityte      IFF-5/4
Matematika      Antanas             Stiklius        IFF-5/3

---------Destytoju-duomenu-failas--------
Programavimas   Jonas           Jonaitis        6
Medijos         Arnas           Sofauskas       5
Informacines    Radvile         Stalciukaite    7
Matematika      Gintaras        Grebliauskas    8
Skaitmenine     Arnas           Sofauskas       4
Psichologija    Radvile         Stalciukaite    7
-------------------------------------------------


Daugiausiai pasirinktu moduliu turi destytojas - Jonaitis Jonas

Nevisu grupiu studentai pasirinko sio destytojo modulius.

Grupes, kuriu studentai nepasirinko sio destytojo moduliu:
Psichologija    Raigardas           Knygius         IFF-5/1
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Matematika      Dainius             Lentauskas      IFF-5/2
Programavimas   Gaile               Pagalvyte       IFF-5/2
Medijos         Ignas               Lovauskas       IFF-5/2
Matematika      Pjuklaite           Ieva            IFF-5/2
Matematika      Antanas             Stiklius        IFF-5/3
```

```
Psichologija    Audrius             Dalgiauskas     IFF-5/3
Programavimas   Vytenis             Deziauskas      IFF-5/3
Psichologija    Puodzius            Jonas           IFF-5/3
Medijos         Pukutis             Pukis           IFF-5/5

-----------------------------------------------
Ivestas modulio pavadinimas: Programavimas
Sio modulio studentu sarasas:
Programavimas   Tomaitis            Tomas           IFF-5/4
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Programavimas   Aiseviciute         Aiste           IFF-5/4
Programavimas   Stasys              Girnaitis       IFF-5/4
Programavimas   Gaile               Pagalvyte       IFF-5/2
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Vytenis             Deziauskas      IFF-5/3
Programavimas   Giedrius            Palangiauskas   IFF-5/4
Programavimas   Jomante             Deklaityte      IFF-5/4
```

2.4. Grafinės vartotojo sąsajos schema

START

Daugiausiai pasirinktu moduliu turi destytojas - Jonaitis Jonas
Visu grupiu studentai pasirinko sio destytojo modulius.

Grupes, kuriu studentai nepasirinko sio destytojo moduliu:

```
Psichologija    Raigardas           Knygius         IFF-5/1
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Matematika      Dainius             Lentauskas      IFF-5/2
Programavimas   Gaile               Pagalvyte       IFF-5/2
Medijos         Ignas               Lovauskas       IFF-5/2
Matematika      Pjuklaite           Ieva            IFF-5/2
Matematika      Antanas             Stiklius        IFF-5/3
Psichologija    Audrius             Dalgiauskas     IFF-5/3
Programavimas   Vytenis             Deziauskas      IFF-5/3
Psichologija    Puodzius            Jonas           IFF-5/3
Medijos         Pukutis             Pukis           IFF-5/5
```

START

Iveskite modulio pavadinima ir dar karta paspauskite antraji 'START' mygtuka

```
Programavimas
```

```
Programavimas   Tomaitis            Tomas           IFF-5/4
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Programavimas   Aiseviciute         Aiste           IFF-5/4
Programavimas   Stasys              Girnaitis       IFF-5/4
Programavimas   Gaile               Pagalvyte       IFF-5/2
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Vytenis             Deziauskas      IFF-5/3
Programavimas   Giedrius            Palangiauskas   IFF-5/4
Programavimas   Jomante             Deklaityte      IFF-5/4
```

2.5. Grafinės vartotojo sąsajos elementų pakeistos savybės

Komponentas	ID	text	Back Color	OnClick	TextMode
Button	Button1	START	Blue	Button1_Click	
Label	Label1				
Label	Label2				
Label	Label5				
TextBox	TextBox3		#FFCCFF		MultiLine
Button	Button2	START	Blue	Button2_Click	
Label	Label4				
TextBox	TextBox1		#FFCCFF		MultiLine
TextBox	TextBox2		#FFCCFF		MultiLine

2.6. Klasių diagramos

1.6. Klasių diagramos

Veiksmai
+Page_Load(in sender:object, in e:EventArgs) +Button1_Click(in sender:object, in e: EventArgs) +DestytojasMaxMod(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string) +ArVisuPasirinko(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string) +GrupesKurNepasirinko(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string, fr: string) +Rikiuoti(StudentaiGR:string()) +SalintiGrupe(StudentaiGR:string(), grupe:string) +PasalintiPaskutini(StudentaiGR:string()) +SpausdintiGrupes(StudentaiGR:string(), fr:string) +SpausdintiRezultatus(Studentai:string(),Destytojai:string(),destytojasMaxVardas:string, destytojasMaxPavarde:string, fr:string) +Button2_Click(in sender:object, in e: EventArgs) +SudarytiStudentuSarasa(NewStudenai:string(), modulisVIP:string, Studentai:string()) +SpausdintiRez2(NewStudentai:string(), modulisVIP:string, fr:string)

Skaityti
+ SkaitytiStud(){query} + SkaitytiDest(){query}

2.7. Programos vartotojo vadovas

Tekstiniame faile duota informacija apie studentų pasirenkamus modulius. Kitame faile duota informacija apie modulius. Paleidę programą spaudžiame pirmąjį „Start" mygtuką, tuomet ekrane parodoma informacija kuris dėstytojas turi daugiausia pasirinktų modulių, ar visų grupių studentai pasirinko to dėstytojo modulius. Taip pat, atspausdintą lentelę su grupėmis, kurių studentai nepasirinko šio dėtytojo modulių. Toliau spaudžiame antrąjį „Start" mygtuką. Atsiranda užrašas, prašantis įvesti modulio pavadinimą ir dar kartą paspaustiantrąjį „Start" mygtuką. Tuomet atsiranda jūsų įvesto modulio studentų sąrašas.

3. Bendrinis susietasis sąrašas

3.1. Darbo užduotis

LD_7. Moduliai.
Studentai renkasi modulius. Už modulius yra atsakingi dėstytojai. Dėstytojas gali būti atsakingas už keletą modulių.

Suraskite, kuris dėstytojas turi daugiausiai pasirinktų modulių.
Nustatykite, ar visų grupių studentai pasirinko šio dėstytojo modulius.
Atspausdinkite grupes, kurių studentai nepasirinko šio dėstytojo modulių.

Duomenys:

- Tekstiniame faile U7a.txt duota informacija apie studentų pasirenkamus modulius: modulio pavadinimas, studento pavardė, vardas, grupė.
- Tekstiniame faile U7b.txt duota informacija apie modulius: modulio pavadinimas, atsakingo dėstytojo pavardė, vardas, kreditų kiekis.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka.
Sudarykite nurodyto modulio (įvedamas klaviatūra) pasirinkusių studentų sąrašą.

3.2. Programos tekstas

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

publicpartialclassL3 : System.Web.UI.Page
{
publicclassStudentas : IComparable<Studentas>, IEquatable<Studentas>
    {
publicstring modulis { get; set; }
publicstring pavarde { get; set; }
publicstring vardas { get; set; }
publicstring grupe { get; set; }

//public Studentas() { }

public Studentas(string modulis, string pavarde, string vardas, string grupe)
        {
this.modulis = modulis;
this.pavarde = pavarde;
this.vardas = vardas;
this.grupe = grupe;
        }
publicint CompareTo(Studentas other)
        {
if (other == null) return 1;
if (modulis.CompareTo(other.modulis) != 0)
return modulis.CompareTo(other.modulis);
else
return pavarde.CompareTo(other.pavarde);
        }
staticpublicbooloperator>(Studentas pir, Studentas ant)
        {
return pir.CompareTo(ant) == 1;
```

```
        }
staticpublicbooloperator<(Studentas pir, Studentas ant)
        {
return pir.CompareTo(ant) == -1;
        }
publicoverridestring ToString()
        {
string eilute;
            eilute = string.Format("{0,-15} {1,-20} {2, -15} {3,-10}", modulis,
pavarde, vardas, grupe);
return eilute;
        }
publicbool Equals(Studentas other)
        {
thrownewNotImplementedException();
        }
    }

publicclassDestytojas : IComparable<Destytojas>, IEquatable<Destytojas>
    {
publicstring modulis { get; set; }
publicstring pavarde { get; set; }
publicstring vardas { get; set; }
publicint kreditai { get; set; }
public Destytojas(string modulis, string pavarde, string vardas, int kreditai)
        {
this.modulis = modulis;
this.pavarde = pavarde;
this.vardas = vardas;
this.kreditai = kreditai;
        }
publicstring ToString1()
        {
string eilute;
            eilute = string.Format("{0,-15} {1,-15} {2, -15} {3,-10}", modulis,
pavarde, vardas, kreditai);
return eilute;
        }
publicint CompareTo(Destytojas other)
        {
if (other == null) return 1;
if (modulis.CompareTo(other.modulis) != 0)
return modulis.CompareTo(other.modulis);
else
return pavarde.CompareTo(other.pavarde);
        }
staticpublicbooloperator>(Destytojas pir, Destytojas ant)
        {
return pir.CompareTo(ant) == 1;
        }
staticpublicbooloperator<(Destytojas pir, Destytojas ant)
        {
return pir.CompareTo(ant) == -1;
        }
publicbool Equals(Destytojas other)
        {
thrownewNotImplementedException();
        }
    }
publicsealedclassMazgas<t>wheret : IComparable<t>, IEquatable<t>
    {
publict Duom { get; set; }
publicMazgas<t> Kitas { get; set; }
public Mazgas(t st, Mazgas<t> adr)
        {
            Duom = st;
```

```
            Kitas = adr;
        }
    }
publicsealedclassSarasas<t> : IEnumerablewheret : IComparable<t>, IEquatable<t>
    {
privateMazgas<t> pr; //saraso pradzia
privateMazgas<t> pb; //saraso pabaiga
privateMazgas<t> d; //saraso sasajai
public Sarasas()
        {
this.pr = null;
this.pb = null;
this.d = null;
        }
publicMazgas<t> Pradzia()
        {
return pr;
        }
publicvoid Kitas()
        {
            d = d.Kitas;
        }
publicbool Yra()
        {
return d != null;
        }
publict ImtiDuomenis()
        {
return d.Duom;
        }
publicvoid DetiDuomenisT(t naujas)
        {
var dd = newMazgas<t>(naujas, null);
if (pr != null)
            {
                pb.Kitas = dd;
                pb = dd;
            }
else
            {
                pr = dd;
                pb = dd;
            }
        }
publicIEnumerator GetEnumerator()
        {
for (Mazgas<t> dd = pr; dd != null; dd = dd.Kitas)
            {
yieldreturn dd.Duom;
            }
        }
//būtina aprašyti, nes IEnumerable<T> paveldi iš IEnumerable
IEnumeratorIEnumerable.GetEnumerator()
        {
thrownewNotImplementedException();
        }
publicvoid Rikiuoti()
        {
for (Mazgas<t> d1 = pr; d1 != null; d1 = d1.Kitas)
            {
Mazgas<t> max = d1;
for (Mazgas<t> d2 = d1; d2 != null; d2 = d2.Kitas)
if (d2.Duom.CompareTo(max.Duom) < 0)
                        max = d2;
t laik = d1.Duom;
                d1.Duom = max.Duom;
```

```
                max.Duom = laik;
            }
        }
    }

protectedvoid Page_Load(object sender, EventArgs e)
    { }
protectedvoid Button1_Click(object sender, EventArgs e)
    {
var Studentai = newSarasas<Studentas>();
var Destytojai = newSarasas<Destytojas>();
string CD1 = @"E:\\2tras\\objektinis\\L2\\U7a.txt"; //studentu duomenu failas
string CD2 = @"E:\\2tras\\objektinis\\L2\\U7b.txt"; //destytoju duomenu failas
string CR = @"E:\\2tras\\objektinis\\L2\\REZ.txt"; //rezultatu duomenu failas
if (File.Exists(CR))
File.Delete(CR);
        SkaitytiStud(CD1, Studentai);
        SkaitytiDest(CD2, Destytojai);
string destytojasMaxVardas; string destytojasMaxPavarde;//destytojas turintis
daugiausiai moduliu
        DestytojasMaxMod(Studentai, Destytojai, out destytojasMaxVardas, out
destytojasMaxPavarde);
        SpausdintiRezultatus(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde, CR);
        GrupesKurNepasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde, CR);

    }
///<summary>
/// Nuskaito studentu duomenu faila
///</summary>
///<param name="fv">studentu failas</param>
///<param name="Studentai">Studentu sarasas</param>
staticvoid SkaitytiStud(string fv, Sarasas<Studentas> Studentai)
    {
string line;
using (var reader = newStreamReader(fv))
        {
while ((line = reader.ReadLine()) != null)
            {
var v = line.Split(' ');
var student = newStudentas(v[0], v[1], v[2], v[3]);
                Studentai.DetiDuomenisT(student);
            }
        }
    }
///<summary>
/// Nuskaito destytoju duomenu faila
///</summary>
///<param name="fv">destytoju failas</param>
///<param name="Destytojai">Destytoju sarasas</param>
staticvoid SkaitytiDest(string fv, Sarasas<Destytojas> Destytojai)
    {
string line;
using (var reader = newStreamReader(fv))
        {
while ((line = reader.ReadLine()) != null)
            {
var v = line.Split(' ');
var destyt = newDestytojas(v[0], v[1], v[2], int.Parse(v[3]));
                Destytojai.DetiDuomenisT(destyt);
            }
        }
    }
///<summary>
/// suranda destytoja, turinti daugiausiai moduliu
```

```
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
staticvoid DestytojasMaxMod(Sarasas<Studentas> Studentai, Sarasas<Destytojas>
Destytojai, outstring destytojasMaxVardas, outstring destytojasMaxPavarde)
    {
int max = 0;
        destytojasMaxVardas = null;
        destytojasMaxPavarde = null;
foreach (Destytojas dest in Destytojai)
        {
int laik = 0;
foreach (Studentas stud in Studentai)
            {
if (dest.modulis == stud.modulis)
                    laik += 1;
            }
if (max < laik)
            {
                destytojasMaxVardas = dest.vardas;
                destytojasMaxPavarde = dest.pavarde;
                max = laik;
            }
        }
    }
///<summary>
/// Suranda ar visu grupiu studentai pasirinko sio destytojo modulius
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<returns></returns>
staticbool ArVisuPasirinko(Sarasas<Studentas> Studentai, Sarasas<Destytojas>
Destytojai, string destytojasMaxVardas, string destytojasMaxPavarde)
    {
foreach (Destytojas dest in Destytojai)
        {
if ((destytojasMaxVardas == dest.vardas) && (destytojasMaxPavarde ==
dest.pavarde))
            {
string modulis = dest.modulis;
foreach (Studentas stud in Studentai)
                {
if (modulis == stud.modulis)
                    {
int visi = 0; int ne = 0;
foreach (Studentas stud1 in Studentai)
                        {
if (stud1.grupe == stud.grupe)
                            {
                                visi += 1;
if (stud.modulis != modulis)
                                    ne += 1;
                            }
                        }
if (ne == visi)
returnfalse;
                    }
                }
```

```
            }
        }
returntrue;
    }
publicvoid SalintiGrupe(Sarasas<Studentas> StudentaiGR, string grupe)
    {
for (Mazgas<Studentas> d = StudentaiGR.Pradzia(); d != null; d = d.Kitas)
        {
if (d.Duom.grupe == grupe)
            {
if (d.Kitas != null)
                {
Mazgas<Studentas> laik = d.Kitas;
                    d.Duom = laik.Duom;
                    d.Kitas = laik.Kitas;
                }
else
                    PasalintiPaskutini(StudentaiGR);
            }
        }

    }
///<summary>
/// suranda grupes, kuriu bent vienas studentas nepasirinko sio destytojo modulio
///</summary>
///<param name="Studentai">Studentu sarasas</param>
///<param name="Destytojai">Destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<param name="fr">rezultatu failas</param>
publicvoid GrupesKurNepasirinko(Sarasas<Studentas> Studentai, Sarasas<Destytojas>
Destytojai, string destytojasMaxVardas, string destytojasMaxPavarde, string fr)
    {
var StudentaiGR = newSarasas<Studentas>();
        StudentaiGR = Studentai;
//rusiuoja pagal grupes, pavardes ir vardus

//jei grupej yra nors vienas nepasirinkes modulio, atspauszinti
//jei visa grupe pasirinkusi ta moduli, pasalinti grupe
foreach (Destytojas dest in Destytojai)
        {
if (destytojasMaxVardas == dest.vardas && destytojasMaxPavarde == dest.pavarde)
            {
string modulis = dest.modulis;
foreach (Studentas stud in Studentai)
                {
if (modulis == stud.modulis)
                    {
int visi = 0; int ne = 0; string grupe = "";
foreach (Studentas stud1 in Studentai)
                        {
if (stud.grupe == stud1.grupe)
                            {
                                visi += 1;
if (stud1.modulis == modulis)
                                {
                                    ne += 1;
                                    grupe = stud1.grupe;
                                }
                            }
                        }
if (ne == visi)
                        {
                            SalintiGrupe(StudentaiGR, grupe);
```

```
                        }
                    }
                }
            }
        }
        StudentaiGR.Rikiuoti();
        SpausdintiGrupes(StudentaiGR, fr);
    }
///<summary>
/// pasalina paskutini elemanta
///</summary>
///<param name="StudentaiGR">surikiuotu studentu sarasas</param>
staticvoid PasalintiPaskutini(Sarasas<Studentas> StudentaiGR)
    {
Mazgas<Studentas> k = StudentaiGR.Pradzia();
for (Mazgas<Studentas> d = StudentaiGR.Pradzia(); d != null; d = d.Kitas)
        {
            k = d.Kitas;
if (k.Kitas == null)
            {
                d.Kitas = null;
                StudentaiGR.DetiDuomenisT(d.Duom);
            }
        }
    }
///<summary>
/// spausdinti rezultatus
///</summary>
///<param name="StudentaiGR">surikiuotu studentu sarasas</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiGrupes(Sarasas<Studentas> StudentaiGR, string fr)
    {
string line = "";
using (var writer = File.AppendText(fr))
        {

            writer.WriteLine();
            writer.WriteLine("Grupes, kuriu studentai nepasirinko sio destytojo
moduliu:");
            Label5.Text = "Grupes, kuriu studentai nepasirinko sio destytojo
moduliu:";
            line += "Modulis         Pavarde              Vardas          Grupe" +
"\r\n" + "\r\n";
for (Mazgas<Studentas> d = StudentaiGR.Pradzia(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.Duom.ToString());
                line += d.Duom.ToString() + "\r\n";
            }
        }
        TextBox3.Text = line;
    }
protectedvoid Button2_Click(object sender, EventArgs e)
    {
string CD1 = @"E:\\2tras\\objektinis\\L2\\U7a.txt"; //studentu duomenu failas
string CR = @"E:\\2tras\\objektinis\\L2\\REZ.txt"; //rezultatu duomenu failas
var Studentai = newSarasas<Studentas>();
        SkaitytiStud(CD1, Studentai);
        Label4.Text = "Iveskite modulio pavadinima ir dar karta paspauskite
antraji 'START' mygtuka";
string modulisVIP = TextBox1.Text;
var NewStudentai = newSarasas<Studentas>();
        SudarytiStudentuSarasa(NewStudentai, modulisVIP, Studentai);
if (modulisVIP != "")
            SpausdintiRez2(NewStudentai, modulisVIP, CR);
    }
///<summary>
```

```
/// sudaromas studentu sarasas pagal ivesta klaviatura moduli
///</summary>
///<param name="NewStudentai">naujas studentu sarasas</param>
///<param name="modulisVIP">Ivestas modulis</param>
///<param name="Studentai">studentu sarasas</param>
publicvoid SudarytiStudentuSarasa(Sarasas<Studentas> NewStudentai, string
modulisVIP, Sarasas<Studentas> Studentai)
    {
string eil = "";
if (modulisVIP != "")
        {
for (Mazgas<Studentas> d = Studentai.Pradzia(); d != null; d = d.Kitas)
            {
if (d.Duom.modulis == modulisVIP)
                {
                    NewStudentai.DetiDuomenisT(d.Duom);
                    eil += d.Duom + "\r\n";
                }
            }
        }
else
            eil = "Neteisingas modulio pavadinimas";
        TextBox2.Text = eil;
    }
///<summary>
/// spausdina gautus rezultatus
///</summary>
///<param name="Studentai">studentu sarasas</param>
///<param name="Destytojai">destytoju sarasas</param>
///<param name="destytojasMaxVardas">daugiausiai moduliu turincio destytojo
vardas</param>
///<param name="destytojasMaxPavarde">daugiausiai moduliu turincio destytojo
pavarde</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiRezultatus(Sarasas<Studentas> Studentai, Sarasas<Destytojas>
Destytojai, string destytojasMaxVardas, string destytojasMaxPavarde, string fr)
    {
using (var writer = File.AppendText(fr))
        {
//--- pradiniu doumenu spausdinimas
            writer.WriteLine("--------------PRADINIAI-DUOMENYS-----------------
");
            writer.WriteLine();
            writer.WriteLine("---------Studentu-duomenu-failas--------");
for (Mazgas<Studentas> d = Studentai.Pradzia(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.Duom.ToString());
            }
            writer.WriteLine();
            writer.WriteLine("---------Destytoju-duomenu-failas--------");
for (Mazgas<Destytojas> d = Destytojai.Pradzia(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.Duom.ToString1());
            }
            writer.WriteLine("------------------------------------------------");
            writer.WriteLine();
//---------
            writer.WriteLine();
            writer.WriteLine("Daugiausiai pasirinktu moduliu turi destytojas - {0}
{1}", destytojasMaxVardas, destytojasMaxPavarde);
            Label1.Text = "Daugiausiai pasirinktu moduliu turi destytojas - " +
destytojasMaxVardas + " " + destytojasMaxPavarde;
            writer.WriteLine();
if (ArVisuPasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde))
            {
```

```
                writer.WriteLine("Nevisu grupiu studentai pasirinko sio destytojo
modulius.");
                Label2.Text = "Visu grupiu studentai pasirinko sio destytojo
modulius.";
            }
if (!ArVisuPasirinko(Studentai, Destytojai, destytojasMaxVardas,
destytojasMaxPavarde))
            {
                writer.WriteLine("Ne visu grupiu studentai pasirinko sio destytojo
modulius.");
                Label2.Text = "Ne visu grupiu studentai pasirinko sio destytojo
modulius.";
            }
        }
    }
///<summary>
/// spausdinami rezultatai po ivedamo modulio klaviatura
///</summary>
///<param name="NewStudentai">naujas studentu sarasas</param>
///<param name="modulisVIP">Ivestas modulis</param>
///<param name="fr">rezultatu failas</param>
publicvoid SpausdintiRez2(Sarasas<Studentas> NewStudentai, string modulisVIP,
string fr)
    {
using (var writer = File.AppendText(fr))
        {
            writer.WriteLine();
            writer.WriteLine("-----------------------------------------------");
            writer.WriteLine("Ivestas modulio pavadinimas: {0}", modulisVIP);
            writer.WriteLine("Sio modulio studentu sarasas:");
for (Mazgas<Studentas> d = NewStudentai.Pradzia(); d != null; d = d.Kitas)
            {
                writer.WriteLine(d.Duom.ToString());
            }
        }
    }
}
```

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="L3.aspx.cs"Inherits="L3"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
</head>
<bodystyle="background-color:#A4A4A4">
<formid="form1"runat="server">
<div>

</div>
<asp:ButtonID="Button1"runat="server"OnClick="Button1_Click"Text="Button"BackColor
="Blue"BorderColor="White"/>
<br/>
<asp:LabelID="Label1"runat="server"Text="Label"></asp:Label>
<br/>
<asp:LabelID="Label2"runat="server"Text="Label"></asp:Label>
<br/>
<br/>
<asp:LabelID="Label5"runat="server"Text="Label"></asp:Label>
<br/>
<asp:TextBoxID="TextBox3"runat="server"Height="254px"BordeColor="#FFCCFF"TextMode=
"MultiLine"Width="653px"></asp:TextBox>
<br/>
<br/>
```

```
<asp:ButtonID="Button2"runat="server"OnClick="Button2_Click"Text="Button"BackColor
="Blue"BorderColor="White"/>
<br/>
<asp:LabelID="Label4"runat="server"Text="Label"></asp:Label>
<br/>
<br/>
<asp:TextBoxID="TextBox1"runat="server"Height="18px"BordeColor="#58D3F7"TextMode="
MultiLine"Width="222px"></asp:TextBox>
<br/>
<br/>
<asp:TextBoxID="TextBox2"runat="server"Height="175px"BordeColor="#58D3F7"TextMode=
"MultiLine"Width="658px"></asp:TextBox>
<p>
 </p>
<p>
 </p>
<p>
 </p>
<p>
 </p>
<br/>
<br/>
</form>
</body>
</html>
```

3.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys

Failas: U7a.txt

```
Programavimas Tomaitis Tomas IFF-5/4
Matematika Pjuklaite Ieva IFF-5/2
Psichologija Puodzius Jonas IFF-5/3
Programavimas Lygiauskas Vladimiras IFF-5/1
Programavimas Aiseviciute Aiste IFF-5/4
Medijos Pukutis Pukis IFF-5/5
Programavimas Stasys Girnaitis IFF-5/4
Programavimas Gaile Pagalvyte IFF-5/2
Medijos Ignas Lovauskas IFF-5/2
Programavimas Diana Paveikslaite IFF-5/1
Matematika Dainius Lentauskas IFF-5/2
Programavimas Vytenis Deziauskas IFF-5/3
Psichologija Raigardas Knygius IFF-5/1
Programavimas Giedrius Palangiauskas IFF-5/4
Psichologija Audrius Dalgiauskas IFF-5/3
Programavimas Jomante Deklaityte IFF-5/4
Matematika Antanas Stiklius IFF-5/3
```

Failas: U7b.txt

```
Programavimas Jonas Jonaitis 6
Medijos Arnas Sofauskas 5
Informacines Radvile Stalciukaite 7
Matematika Gintaras Grebliauskas 8
Skaitmenine Arnas Sofauskas 4
Psichologija Radvile Stalciukaite 7
```

Rezultatai

Failas: REZ.txt

```
--------------PRADINIAI-DUOMENYS------------------

---------Studentu-duomenu-failas--------
Programavimas   Tomaitis            Tomas           IFF-5/4
Matematika      Pjuklaite           Ieva            IFF-5/2
Psichologija    Puodzius            Jonas           IFF-5/3
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Programavimas   Aiseviciute         Aiste           IFF-5/4
Medijos         Pukutis             Pukis           IFF-5/5
Programavimas   Stasys              Girnaitis       IFF-5/4
Programavimas   Gaile               Pagalvyte       IFF-5/2
Medijos         Ignas               Lovauskas       IFF-5/2
Programavimas   Diana               Paveikslaite    IFF-5/1
Matematika      Dainius             Lentauskas      IFF-5/2
Programavimas   Vytenis             Deziauskas      IFF-5/3
Psichologija    Raigardas           Knygius         IFF-5/1
Programavimas   Giedrius            Palangiauskas   IFF-5/4
Psichologija    Audrius             Dalgiauskas     IFF-5/3
Programavimas   Jomante             Deklaityte      IFF-5/4
Matematika      Antanas             Stiklius        IFF-5/3

---------Destytoju-duomenu-failas--------
Programavimas   Jonas           Jonaitis        6
Medijos         Arnas           Sofauskas       5
Informacines    Radvile         Stalciukaite    7
Matematika      Gintaras        Grebliauskas    8
Skaitmenine     Arnas           Sofauskas       4
Psichologija    Radvile         Stalciukaite    7
-------------------------------------------------


Daugiausiai pasirinktu moduliu turi destytojas - Jonaitis Jonas

Nevisu grupiu studentai pasirinko sio destytojo modulius.

Grupes, kuriu studentai nepasirinko sio destytojo moduliu:
Psichologija    Raigardas           Knygius         IFF-5/1
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Matematika      Dainius             Lentauskas      IFF-5/2
Programavimas   Gaile               Pagalvyte       IFF-5/2
Medijos         Ignas               Lovauskas       IFF-5/2
Matematika      Pjuklaite           Ieva            IFF-5/2
Matematika      Antanas             Stiklius        IFF-5/3
Psichologija    Audrius             Dalgiauskas     IFF-5/3
Programavimas   Vytenis             Deziauskas      IFF-5/3
Psichologija    Puodzius            Jonas           IFF-5/3
Medijos         Pukutis             Pukis           IFF-5/5

-----------------------------------------------
Ivestas modulio pavadinimas: Programavimas
Sio modulio studentu sarasas:
Programavimas   Tomaitis            Tomas           IFF-5/4
Programavimas   Lygiauskas          Vladimiras      IFF-5/1
Programavimas   Aiseviciute         Aiste           IFF-5/4
Programavimas   Stasys              Girnaitis       IFF-5/4
Programavimas   Gaile               Pagalvyte       IFF-5/2
Programavimas   Diana               Paveikslaite    IFF-5/1
Programavimas   Vytenis             Deziauskas      IFF-5/3
Programavimas   Giedrius            Palangiauskas   IFF-5/4
```

3.4. Grafinės vartotojo sąsajos schema

3.5. Grafinės vartotojo sąsajos elementų pakeistos savybės

Komponentas	ID	text	Back Color	OnClick	TextMode
Button	Button1	START	Blue	Button1_Click	
Label	Label1				
Label	Label2				
Label	Label5				
TextBox	TextBox3		#58D3F7		MultiLine
Button	Button2	START	Blue	Button2_Click	
Label	Label4				
TextBox	TextBox1		#58D3F7		MultiLine
TextBox	TextBox2		#58D3F7		MultiLine

3.6.Klasių diagramos

3.6. Klasių diagramos

Studetas
+ modulis : string + pavarde : string + vardas : string + grupe : string
+ Studentas(in modulis : string; in pavarde : string; in vardas : string; in grupe: string) + CompareTo(in other:Studentas) + ToString() + Equals(in other: Studentas)

Destytojas
+ modulis : string + pavarde : string + vardas : string + kreditai : integer
+ Destytojas(in modulis : string; in pavarde : string; in vardas : string; in kreditai: integer) + ToString() + CompareTo(in other: Destytojas)

Mazgas
+ Duom : tipas + Kitas : tipas
+ Mazgas(in st:tipas; in adr : tipas)

Sarasas<tipas>
-pr : Mazgas -pb: Mazgas

-d: Mazgas
+Pradzia() +Kitas() +ImtiDuomenis() +DetiDuomenisT(<tipas>) +GetEnumerator() +Rikiuoti

Veiksmai
+Page_Load(in sender:object, in e:EventArgs) +Button1_Click(in sender:object, in e: EventArgs) +DestytojasMaxMod(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string) +ArVisuPasirinko(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string) +GrupesKurNepasirinko(Studentai:string(), Destytojai:string(), out destytojasMaxVardas : string, out destytojasMaxPavarde : string, fr: string) +SalintiGrupe(StudentaiGR:string(), grupe:string) +PasalintiPaskutini(StudentaiGR:string()) +SpausdintiGrupes(StudentaiGR:string(), fr:string) +SpausdintiRezultatus(Studentai:string(),Destytojai:string(),destytojasMaxVardas: string, destytojasMaxPavarde:string, fr:string) +Button2_Click(in sender:object, in e: EventArgs) +SudarytiStudentuSarasa(NewStudenai:string(), modulisVIP:string, Studentai:string()) +SpausdintiRez2(NewStudentai:string(), modulisVIP:string, fr:string) + SkaitytiStud(){query} + SkaitytiDest(){query}

3.7. *Programos vartotojo vadovas*

Tekstiniame faile duota informacija apie studentų pasirenkamus modulius. Kitame faile duota informacija apie modulius. Paleidę programą spaudžiame pirmąjį „Start" mygtuką, tuomet ekrane parodoma informacija kuris dėstytojas turi daugiausia pasirinktų modulių, ar visų grupių studentai pasirinko to dėstytojo modulius. Taip pat, atspausdintą lentelę su grupėmis, kurių studentai nepasirinko šio dėtytojo modulių. Toliau spaudžiame antrąjį „Start" mygtuką. Atsiranda užrašas, prašantis įvesti modulio pavadinimą ir dar kartą paspaustiantrąjį „Start" mygtuką. Tuomet atsiranda jūsų įvesto modulio studentų sąrašas.

4. Bendrinės kolekcijos

4.1. Darbo užduotis

LDD_7. Išrinktieji. Darbo valandomis seimo nariai priiminėja rinkėjus seimo priimamajame. Apie tai yra kaupiama informacija kiekvieną dieną. Pirmoje failo eilutėje yra data. Toliau faile yra lankytojų pavardės, vardai, atvykimo laikas, konsultacijos trukmė. Atskirame faile yra seimo narių budėjimo grafikas: seimo nario pavardė ir vardas, dienos data, budėjimo priimamajame laikas (pradžia, pabaiga).

Sudaryti seimo narių sąrašą, kuriame būtų informacija apie vidutinę vienos konsultacijos trukmę, vidutinį apsilankančių rinkėjų skaičių per dieną.

Surūšiuoti pagal trukmę ir apsilankiusių rinkėjų skaičių.

4.2. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : Page
{
    public class Diena : IComparable<Diena>, IEquatable<Diena>
    {
        public string data { get; set; }
        public string pavarde { get; set; }
        public string vardas { get; set; }
        public DateTime atvykLaik { get; set; }
        public DateTime konsultacTruk { get; set; }
        public Diena(string data, string pavarde, string vardas, DateTime
atvykLaik, DateTime konsultacTruk)
        {
            this.data = data;
            this.pavarde = pavarde;
            this.vardas = vardas;
            this.atvykLaik = atvykLaik;
            this.konsultacTruk = konsultacTruk;
        }
        public override string ToString()
        {
            return string.Format("{0,-15} {1,-15} {2,-15} {3,-5:HH:mm} {4,-
5:HH:mm}", data, pavarde, vardas, atvykLaik, konsultacTruk);
        }
        public int CompareTo(Diena other)
        {
            throw new NotImplementedException();
        }

        public bool Equals(Diena other)
        {
            throw new NotImplementedException();
        }
    }
    public class Nauja : IComparable<Nauja>, IEquatable<Nauja>
    {
        public string pavarde { get; set; }
        public string vardas { get; set; }
        public double trukme { get; set; }
        public int rinkejai { get; set; }
        public Nauja(string pavarde, string vardas, double trukme, int rinkejai)
        {
            this.pavarde = pavarde;
```

```
            this.vardas = vardas;
            this.trukme = trukme;
            this.rinkejai = rinkejai;
        }
        public override string ToString()
        {
            return string.Format("{0,-15} {1,-15} {2,-10} {3,-5}", pavarde,
vardas, trukme, rinkejai);
        }
        public int CompareTo(Nauja other)
        {
            if (other == null) return 1;
            if (trukme != other.trukme)
            {
                return trukme.CompareTo(other.trukme);
            }
            else if (rinkejai != other.rinkejai)
            {
                return rinkejai.CompareTo(other.rinkejai);
            }
            else return 1;
        }
        static public bool operator <(Nauja pir, Nauja ant)
        {
            return pir.CompareTo(ant) == 1;
        }
        static public bool operator >(Nauja pir, Nauja ant)
        {
            return pir.CompareTo(ant) == -1;
        }
        //public bool Equals(Nauja other)
        //{
        //    throw new NotImplementedException();
        //}
        public override bool Equals(object obj)
        {
            return base.Equals(obj as Nauja);
        }
        public bool Equals(Nauja narys)
        {
            if (Object.ReferenceEquals(narys, null))
            {
                return false;
            }
            if (this.GetType() != narys.GetType())
                return false;
            return (pavarde == narys.pavarde);
        }
        public override int GetHashCode()
        {
            return pavarde.GetHashCode() ^ vardas.GetHashCode();
        }
    }

    public class Seimas : IComparable<Seimas>, IEquatable<Seimas>
    {
        public string pavardeS { get; set; }
        public string vardasS { get; set; }
        public string dienosData { get; set; }
        public DateTime budejimasPR { get; set; }
        public DateTime budejimasPB { get; set; }
        public Seimas() { }
        public Seimas(string pavardeS, string vardasS, string dienosData, DateTime
budejimasPR, DateTime budejimasPB)
        {
            this.pavardeS = pavardeS;
```

```
            this.vardasS = vardasS;
            this.dienosData = dienosData;
            this.budejimasPR = budejimasPR;
            this.budejimasPB = budejimasPB;
        }
        public override string ToString()
        {
            return string.Format("{0,-15} {1,-15} {2,-15} {3,-5:HH:mm} {4,-
5:HH:mm}", pavardeS, vardasS, dienosData, budejimasPR, budejimasPB);
        }
        public int CompareTo(Seimas other)
        {
            throw new NotImplementedException();
        }
        //public bool Equals(Nauja other)
        //{
        //    throw new NotImplementedException();
        //}
        public override bool Equals(object obj)
        {
            return base.Equals(obj as Seimas);
        }
        public bool Equals(Seimas seim)
        {
            if (Object.ReferenceEquals(seim, null))
            {
                return false;
            }
            if (this.GetType() != seim.GetType())
                return false;
            return (pavardeS == seim.pavardeS);
        }
        public override int GetHashCode()
        {
            return pavardeS.GetHashCode() ^ vardasS.GetHashCode();
        }
    }
    protected void Page_Load(object sender, EventArgs e)
    { }
    static void SkaitytiDienas(LinkedList<Diena> DienosInfo)
    {
        string[] files = System.IO.Directory.GetFiles(@"F:\\2tras\\objektinis\\L4
new", "Diena*.txt");
        string data = null; string line = null;
        foreach (var file in files)
        {
            using (var failas = new System.IO.StreamReader(file,
Encoding.GetEncoding(1257)))
            {

                line = failas.ReadLine();
                if (line != null)
                {
                    data = line;
                }

                while ((line = failas.ReadLine()) != null)
                {
                    var value = line.Split(' ');
                    var dien = new Diena(data, value[0], value[1],
Convert.ToDateTime(value[2]), Convert.ToDateTime(value[3]));
                    DienosInfo.AddLast(dien);
                }
            }
        }
    }
```

```
    static void SkaitytiSeima(LinkedList<Seimas> SeimoInfo, string fd)
    {
        string line;
        using (var failas = new System.IO.StreamReader(fd,
Encoding.GetEncoding(1257)))
        {
            while ((line = failas.ReadLine()) != null)
            {
                var value = line.Split(' ');
                var seim = new Seimas(value[0], value[1], value[2],
Convert.ToDateTime(value[3]), Convert.ToDateTime(value[4]));
                SeimoInfo.AddLast(seim);
            }
        }
    }

    static void SarasoSukurimas(LinkedList<Diena> DienosInfo, LinkedList<Seimas>
SeimoInfo, LinkedList<Nauja> Sarasas)
    {
        foreach (Seimas seim in SeimoInfo)
        {
           // bool laik = true;
           //todo  Sarasas.contains

            //foreach (Nauja nauj in Sarasas)
            //{
            if (!Sarasas.Contains(new Nauja(seim.pavardeS, "", 0, 0)))
                //if(!Sarasas.Contains(nauj))
                //if (seim.pavardeS == nauj.pavarde)
                    //laik = false;
            //}
            //if (laik)
            {
                string pavarde = seim.pavardeS;
                string vardas = seim.vardasS;
                int sk=0; double trukme=0;
                VidutineKonsultacijosTrukme(DienosInfo, SeimoInfo, seim, ref sk,
ref trukme);
                double trukm = trukme;
                int rinkejai = sk;
                var naujas = new Nauja(pavarde, vardas, trukme, rinkejai);
                Sarasas.AddLast(naujas);
            }
        }
    }
    static void VidutineKonsultacijosTrukme(LinkedList<Diena> DienosInfo,
LinkedList<Seimas> SeimoInfo, Seimas seim, ref int sk, ref double trukme)
    {
        sk = 0; int laik = 0; trukme = 0;
        DateTime trukm = new DateTime();

        foreach (Seimas seimas in SeimoInfo)
        {
            foreach (Diena dien in DienosInfo)
            {

                if (seim.vardasS == seimas.vardasS && seim.pavardeS ==
seimas.pavardeS)
                {
                    double hour = dien.konsultacTruk.Hour;
                    double min = dien.konsultacTruk.Minute;
                    double dienPab = (dien.atvykLaik.Hour + hour) * 60 +
dien.atvykLaik.Minute + min;
                    double seimolaik = seimas.budejimasPB.Hour * 60 +
seimas.budejimasPB.Minute;
```

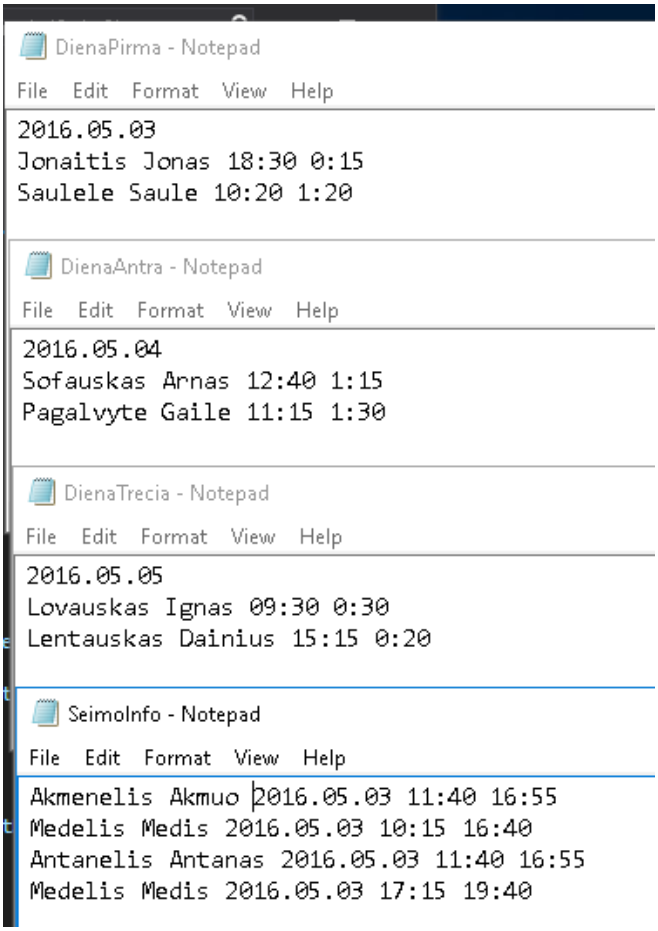
```
                    if ((seimas.dienosData == dien.data) && (seimas.budejimasPR <=
dien.atvykLaik) && (seimolaik >= dienPab))
                    {
                        sk++; laik++;
                        trukm = trukm.AddHours(hour).AddMinutes(min);
                    }
                }
            }
        }

        if (laik == 0)
            trukme = 0;
        else
        {
            trukme = 60 * trukm.Hour + trukm.Minute;
            trukme = trukme / sk;
        }
    }
    public void Rikiuoti(LinkedList<Nauja> Sarasas)
    {
        List<Nauja> Temp = new List<Nauja>();
        foreach (Nauja item in Sarasas)
        {
            Temp.Add(item);
        }
        for (int i = 0; i < Temp.Count; i++)
        {
            for (int j = i; j < Temp.Count - 1; j++)
            {
                if (Temp[i] > Temp[j])
                {
                    Nauja tempor = Temp[i];
                    Temp[i] = Temp[j];
                    Temp[j] = tempor;
                }
            }
        }
        Sarasas.Clear();
        foreach (Nauja item in Temp)
        {
            Sarasas.AddLast(item);
        }
    }
    public void Spausdina(LinkedList<Diena> DienosInfo, LinkedList<Seimas>
SeimoInfo, LinkedList<Nauja> Sarasas, string fr)
    {
        using (var wr = new StreamWriter(fr, true))
        {
            wr.WriteLine("PRADINIAI DUOMENYS");
            wr.WriteLine();
            string eil = "Dienu failu informacija" + "\r\n";
            foreach (Diena d in DienosInfo)
            {
                eil += d.ToString() + "\r\n";
            }
            wr.WriteLine(eil);
            wr.WriteLine();
            TextBox2.Text = eil;
            eil = "Seimo failo informacija" + "\r\n";
            foreach (Seimas one in SeimoInfo)
            {
                eil += one.ToString() + "\r\n";
            }
            wr.WriteLine(eil);
            wr.WriteLine(); wr.WriteLine();
            TextBox1.Text = eil;
```

```
            wr.WriteLine("Naujas sarasas");
            wr.WriteLine();
            eil = "\r\n";
            foreach (Nauja n in Sarasas)
            {
                eil += n.ToString() + "\r\n";
            }
            wr.WriteLine(eil);
            TextBox3.Text = eil;
        }
    }
    protected void TextBox1_TextChanged(object sender, EventArgs e)
    { }
    protected void Button1_Click1(object sender, EventArgs e)
    {
        const string CDS = @"F:\\2tras\\objektinis\\L4 new\\SeimoInfo.txt";
        const string REZ = @"F:\\2tras\\objektinis\\L4 new\\Rezultatai.txt";
        if (File.Exists(REZ))
        {
            File.Delete(REZ);
        }

        LinkedList<Diena> DienosInfo = new LinkedList<Diena>();
        LinkedList<Seimas> SeimoInfo = new LinkedList<Seimas>();

        SkaitytiDienas(DienosInfo);
        SkaitytiSeima(SeimoInfo, CDS);

        LinkedList<Nauja> Sarasas = new LinkedList<Nauja>();
        SarasoSukurimas(DienosInfo, SeimoInfo, Sarasas);

        Rikiuoti(Sarasas);
        Spausdina(DienosInfo, SeimoInfo, Sarasas, REZ);
    }
}
```

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click1"
style="height: 26px" Text="Button" />
        <br />
        <asp:TextBox ID="TextBox1" runat="server" Height="180px"
OnTextChanged="TextBox1_TextChanged" TextMode="MultiLine"
Width="661px"></asp:TextBox>
        <asp:TextBox ID="TextBox2" runat="server" Height="160px"
TextMode="MultiLine" Width="655px"></asp:TextBox>
        <br />
        <br />
        <asp:TextBox ID="TextBox3" runat="server" Height="147px"
TextMode="MultiLine" Width="662px"></asp:TextBox>
    </form>
</body>
</html>
```

4.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys:



Rezultatai:

Rezultatai - Notepad

File Edit Format View Help

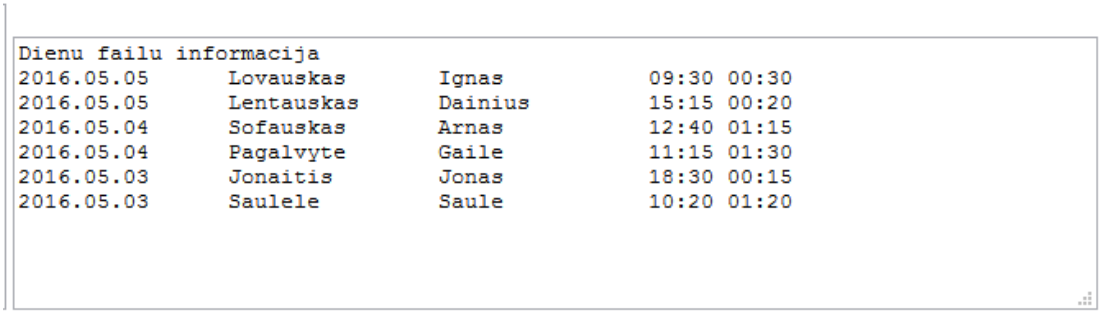
```
PRADINIAI DUOMENYS

Dienu failu informacija
2016.05.05      Lovauskas       Ignas           09:30 00:30
2016.05.05      Lentauskas      Dainius         15:15 00:20
2016.05.04      Sofauskas       Arnas           12:40 01:15
2016.05.04      Pagalvyte       Gaile           11:15 01:30
2016.05.03      Jonaitis        Jonas           18:30 00:15
2016.05.03      Saulele         Saule           10:20 01:20


Seimo failo informacija
Akmenelis       Akmuo           2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      10:15 16:40
Antanelis       Antanas         2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      17:15 19:40



Naujas sarasas


Akmenelis       Akmuo           0         0
Antanelis       Antanas         0         0
Medelis         Medis           47,5      2
```

4.4. Grafinės vartotojo sąsajos schema

Button

```
Seimo failo informacija
Akmenelis       Akmuo           2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      10:15 16:40
Antanelis       Antanas         2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      17:15 19:40
```

```
Medelis         Medis           47,5      2
Akmenelis       Akmuo           0         0
Antanelis       Antanas         0         0
```



```
Dienu failu informacija
2016.05.05      Lovauskas       Ignas           09:30 00:30
2016.05.05      Lentauskas      Dainius         15:15 00:20
2016.05.04      Sofauskas       Arnas           12:40 01:15
2016.05.04      Pagalvyte       Gaile           11:15 01:30
2016.05.03      Jonaitis        Jonas           18:30 00:15
2016.05.03      Saulele         Saule           10:20 01:20
```

4.5. Grafinės vartotojo sąsajos elementų pakeistos savybės

Komponentas	ID	text	Back Color	OnClick	TextMode
Button	Button1	START		Button1_Click	
TextBox	TextBox3				MultiLine
TextBox	TextBox1				MultiLine
TextBox	TextBox2				MultiLine

4.6. Klasių diagramos

Diena
+data : string +pavarde : string +vardas : string + atvykLaik : DateTime + konsultacTrukm : DateTime
+ Diena(int data : string, in pavarde: string, in vardas: string, in atvykLaik : DateTime, in konsultTrukm : DateTime) + ToString() + CompareTo(in other : Diena) + Equals(in other : Diena)

Nauja
+pavarde : string +vardas : string +trukme : double + int : rinkejai
+ Nauja(, in pavarde: string, in vardas: string, in trukme:double, int rinkejai : int) + ToString() + CompareTo(in other : Nauja) + Equals(in other : Diena)

Seimas
+pavarde : string +vardas : string +dienosData:string +budejimasPR:DateTime +budejimasPB:DateTIme
+Seimas(in pavarde : string, in vardas : string, in dienosData:string, in budejimasPR:DateTime, in budejimasPB:DateTime)

+ ToString() + CompareTo(in other : Diena) + Equals(in other : Diena)

4.7. Programos vartotojo vadovas

Viename faile duota informacija apie seimo narių budėjimo grafiką: seimo nario pavardė ir vardas, dienos data, budėjimo priimamajame laikas(pradžia, pabaiga). Kituose failuose informacija apie lankytojus: pirmoje eilutėje data, tolimesnėse informacija apie lankytoją: pavardės, vardai, atvykimo laikas, konsultacijos trukmė.

Paspaudus „SART" mygtuką, programa surašo pradinius duomenis ir sukuria naują saršą, kuriame yra seimo nario vardas, pavardė, vidutinė konsultacijos trukmė ir vidutinis apsilankančių rinkėjų skaičius per dieną.

Sarašas surikiuotas pagal trukmę ir apsilankiusių lankytojų skaičių.

5. Deklaratyvusis programavimas

5.1. Darbo užduotis

LDD_7. Išrinktieji. Darbo valandomis seimo nariai priiminėja rinkėjus seimo priimamajame. Apie tai yra kaupiama informacija kiekvieną dieną. Pirmoje failo eilutėje yra data. Toliau faile yra lankytojų pavardės, vardai, atvykimo laikas, konsultacijos trukmė. Atskirame faile yra seimo narių budėjimo grafikas: seimo nario pavardė ir vardas, dienos data, budėjimo priimamajame laikas (pradžia, pabaiga).

Sudaryti seimo narių sąrašą, kuriame būtų informacija apie vidutinę vienos konsultacijos trukmę, vidutinį apsilankančių rinkėjų skaičių per dieną.

Surūšiuoti pagal trukmę ir apsilankiusių rinkėjų skaičių.

5.2. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : Page
{
    public class Diena
    {
        public string data { get; set; }
        public string pavarde { get; set; }
        public string vardas { get; set; }
        public DateTime atvykLaik { get; set; }
        public DateTime konsultacTruk { get; set; }
        public Diena(string data, string pavarde, string vardas, DateTime
atvykLaik, DateTime konsultacTruk)
        {
            this.data = data;
            this.pavarde = pavarde;
            this.vardas = vardas;
            this.atvykLaik = atvykLaik;
            this.konsultacTruk = konsultacTruk;
        }
        public override string ToString()
        {
            return string.Format("{0,-15} {1,-15} {2,-15} {3,-5:HH:mm} {4,-
5:HH:mm}", data, pavarde, vardas, atvykLaik, konsultacTruk);
        }
    }

    public class Seimas : IEquatable<Seimas>
    {
        public string pavardeS { get; set; }
        public string vardasS { get; set; }
        public string dienosData { get; set; }
        public DateTime budejimasPR { get; set; }
        public DateTime budejimasPB { get; set; }

        public double trukme { get; set; }
        public int rinkejai { get; set; }

        public Seimas() { }
        public Seimas(string pavardeS, string vardasS, string dienosData, DateTime
budejimasPR, DateTime budejimasPB, double trukme, int rinkejai)
        {
```

```
            this.pavardeS = pavardeS;
            this.vardasS = vardasS;
            this.dienosData = dienosData;
            this.budejimasPR = budejimasPR;
            this.budejimasPB = budejimasPB;
            this.trukme = trukme;
            this.rinkejai = rinkejai;
        }
        public override string ToString()
        {
            return string.Format("{0,-15} {1,-15} {2,-15} {3,-5:HH:mm} {4,-
5:HH:mm}", pavardeS, vardasS, dienosData, budejimasPR, budejimasPB);
        }
        public string ToString2()
        {
            return string.Format("{0,-15} {1,-15} {2,-10} {3,-5}", pavardeS,
vardasS, trukme, rinkejai);
        }

        public bool Equals(Seimas other)
        {
            return pavardeS.Equals(other.pavardeS);
        }
    }
    protected void Page_Load(object sender, EventArgs e)
    { }
    static void SkaitytiDienas(List<Diena> DienosInfo)
    {
        string[] files =
System.IO.Directory.GetFiles(@"E:\\2tras\\objektinis\\L5", "Diena*.txt");
        string data = null; string line = null;
        foreach (var file in files)
        {
            using (var failas = new System.IO.StreamReader(file,
Encoding.GetEncoding(1257)))
            {

                line = failas.ReadLine();
                if (line != null)
                {
                    data = line;
                }

                while ((line = failas.ReadLine()) != null)
                {
                    var value = line.Split(' ');
                    var dien = new Diena(data, value[0], value[1],
Convert.ToDateTime(value[2]), Convert.ToDateTime(value[3]));
                    DienosInfo.Add(dien);
                }
            }
        }
    }

    static void SkaitytiSeima(List<Seimas> SeimoInfo, string fd)
    {
        string line;
        using (var failas = new System.IO.StreamReader(fd,
Encoding.GetEncoding(1257)))
        {
            while ((line = failas.ReadLine()) != null)
            {
                var value = line.Split(' ');
                var seim = new Seimas(value[0], value[1], value[2],
Convert.ToDateTime(value[3]), Convert.ToDateTime(value[4]), 0, 0);
                SeimoInfo.Add(seim);
```

```
            }
        }
    }

    static void SarasoSukurimas(List<Diena> DienosInfo, List<Seimas> SeimoInfo,
List<Seimas> NaujasSarasas)
    {
        foreach (Seimas seim in SeimoInfo)
        {
            string pavarde = seim.pavardeS;
            string vardas = seim.vardasS;
            int sk = 0; double trukme = 0;
            VidutineKonsultacijosTrukme(DienosInfo, SeimoInfo, seim, ref sk, ref
trukme);
            double trukm = trukme;
            int rinkejai = sk;
            var naujas = new Seimas(pavarde, vardas, seim.dienosData,
seim.budejimasPR, seim.budejimasPB, trukme, rinkejai);
            // if (!NaujasSarasas.Contains(naujas))

            //bool laik = false;
            //foreach (Seimas nauj in NaujasSarasas)
            //    if (nauj.pavardeS.Contains(seim.pavardeS))
            //        laik = true;
            //if(!laik)

            NaujasSarasas.Add(naujas);

        }
        NaujasSarasas.Distinct();
    }
    static void VidutineKonsultacijosTrukme(List<Diena> DienosInfo, List<Seimas>
SeimoInfo, Seimas seim, ref int sk, ref double trukme)
    {
        sk = 0; int laik = 0; trukme = 0; int skaicius = 0;
        DateTime trukm = new DateTime();
        SeimoInfo.Where(seimas => seimas.Equals(seim)).ToList().ForEach(seimas =>
        {
            DienosInfo.ForEach(dien =>
            {
                double hour = dien.konsultacTruk.Hour;
                double min = dien.konsultacTruk.Minute;
                double dienPab = (dien.atvykLaik.Hour + hour) * 60 +
dien.atvykLaik.Minute + min;
                double seimolaik = seimas.budejimasPB.Hour * 60 +
seimas.budejimasPB.Minute;
                if ((seimas.dienosData == dien.data) && (seimas.budejimasPR <=
dien.atvykLaik) && (seimolaik >= dienPab))
                {
                    skaicius++; laik++;
                    trukm = trukm.AddHours(hour).AddMinutes(min);
                }
            });
        });
        if (laik == 0)
            trukme = 0;
        else
        {
            trukme = 60 * trukm.Hour + trukm.Minute;
            trukme = trukme / sk;
        }
        sk = skaicius;

    }
    public void Spausdina(List<Diena> DienosInfo, List<Seimas> SeimoInfo,
List<Seimas> NaujasSarasas, string fr)
```

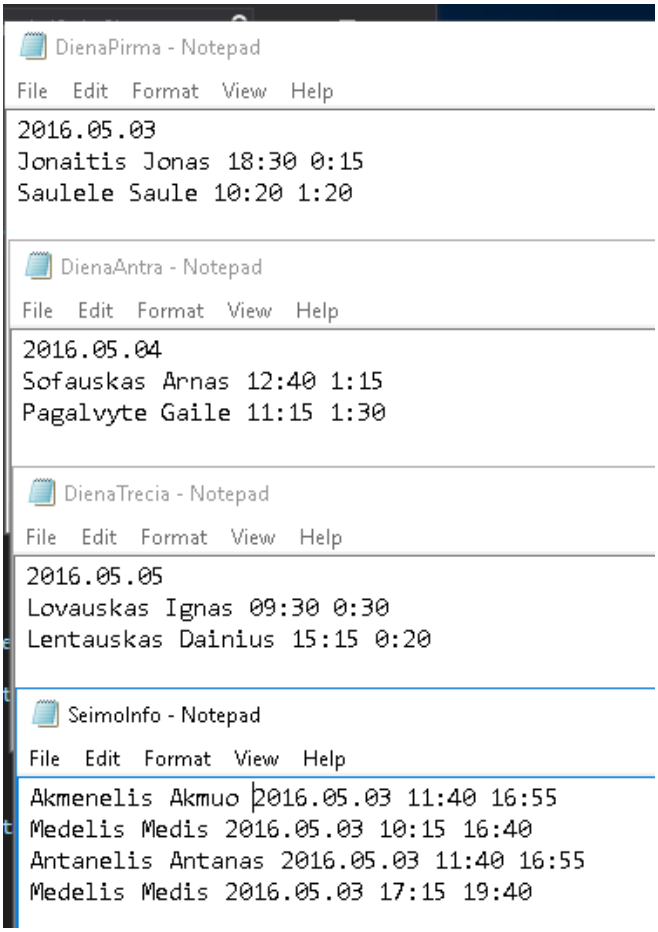
```
    {
        using (var wr = new StreamWriter(fr, true))
        {
            wr.WriteLine("PRADINIAI DUOMENYS");
            wr.WriteLine();
            string eil = "Dienu failu informacija" + "\r\n";
            foreach (Diena d in DienosInfo)
            {
                eil += d.ToString() + "\r\n";
            }
            wr.WriteLine(eil);
            wr.WriteLine();
            TextBox2.Text = eil;
            eil = "Seimo failo informacija" + "\r\n";
            foreach (Seimas one in SeimoInfo)
            {
                eil += one.ToString() + "\r\n";
            }
            wr.WriteLine(eil);
            wr.WriteLine(); wr.WriteLine();
            TextBox1.Text = eil;
            wr.WriteLine("Naujas sarasas");
            wr.WriteLine();
            eil = "\r\n";
            foreach (Seimas n in NaujasSarasas)
            {
                eil += n.ToString2() + "\r\n";
            }
            wr.WriteLine(eil);
            TextBox3.Text = eil;
        }
    }
    protected void TextBox1_TextChanged(object sender, EventArgs e)
    { }
    protected void Button1_Click1(object sender, EventArgs e)
    {
        const string CDS = @"E:\\2tras\\objektinis\\L5\\SeimoInfo.txt";
        const string REZ = @"E:\\2tras\\objektinis\\L5\\Rezultatai.txt";
        if (File.Exists(REZ))
        {
            File.Delete(REZ);
        }

        List<Diena> DienosInfo = new List<Diena>();
        List<Seimas> SeimoInfo = new List<Seimas>();

        SkaitytiDienas(DienosInfo);
        SkaitytiSeima(SeimoInfo, CDS);

        List<Seimas> NaujasSarasas = new List<Seimas>();
        SarasoSukurimas(DienosInfo, SeimoInfo, NaujasSarasas);

        NaujasSarasas = NaujasSarasas.OrderBy(nn => nn.trukme).ThenBy(nn =>
nn.rinkejai.ToString()).ToList();
        Spausdina(DienosInfo, SeimoInfo, NaujasSarasas, REZ);
    }
}
```

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
```

```
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click1"
style="height: 26px" Text="Button" />
        <br />
        <asp:TextBox ID="TextBox1" runat="server" Height="180px"
OnTextChanged="TextBox1_TextChanged" TextMode="MultiLine"
Width="661px"></asp:TextBox>
        <asp:TextBox ID="TextBox2" runat="server" Height="160px"
TextMode="MultiLine" Width="655px"></asp:TextBox>
        <br />
        <br />
        <asp:TextBox ID="TextBox3" runat="server" Height="147px"
TextMode="MultiLine" Width="662px"></asp:TextBox>
    </form>
</body>
</html>
```

5.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys:



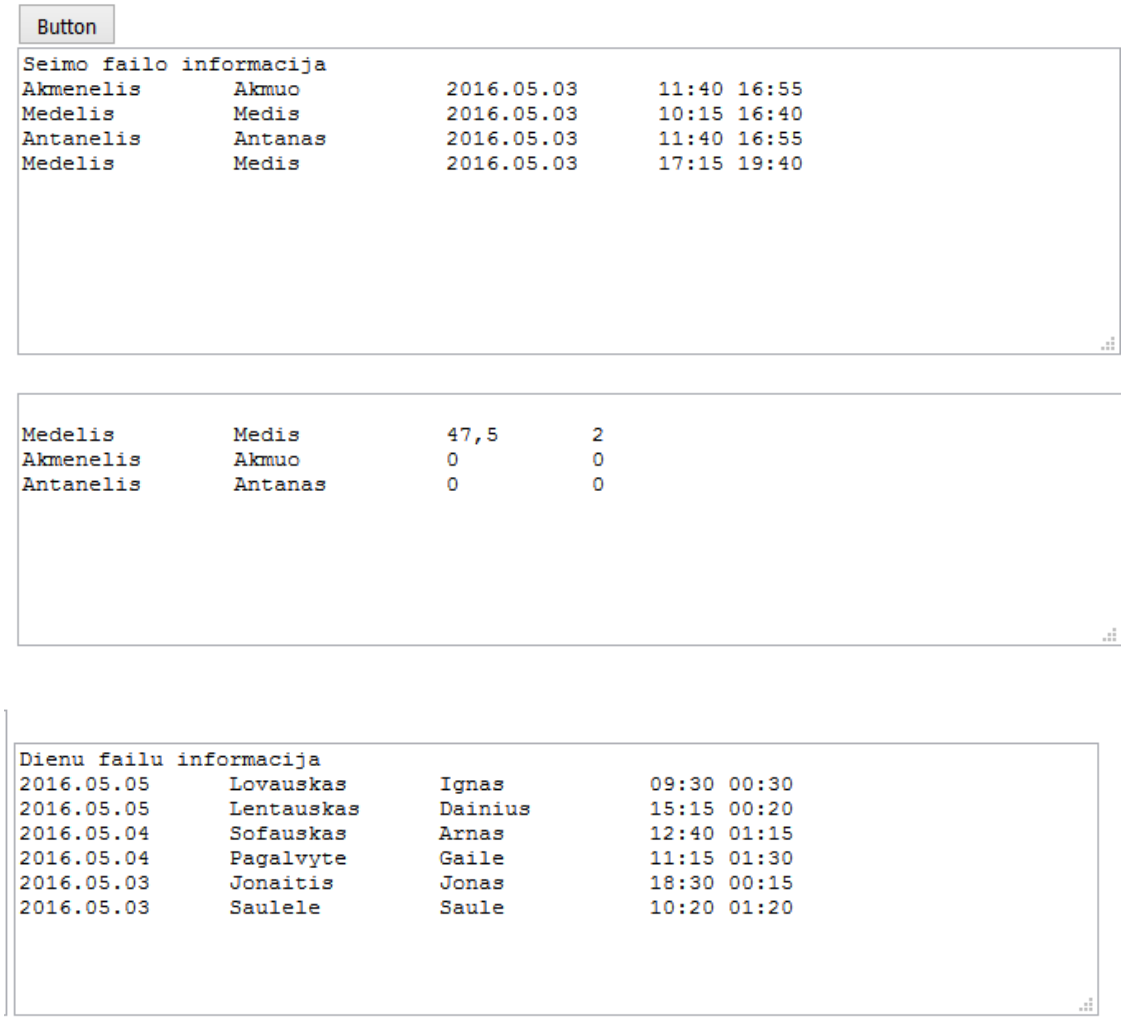
Rezultatai:

Rezultatai - Notepad

File Edit Format View Help

```
PRADINIAI DUOMENYS

Dienu failu informacija
2016.05.05      Lovauskas       Ignas           09:30 00:30
2016.05.05      Lentauskas      Dainius         15:15 00:20
2016.05.04      Sofauskas       Arnas           12:40 01:15
2016.05.04      Pagalvyte       Gaile           11:15 01:30
2016.05.03      Jonaitis        Jonas           18:30 00:15
2016.05.03      Saulele         Saule           10:20 01:20


Seimo failo informacija
Akmenelis       Akmuo           2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      10:15 16:40
Antanelis       Antanas         2016.05.03      11:40 16:55
Medelis         Medis           2016.05.03      17:15 19:40



Naujas sarasas


Akmenelis       Akmuo           0         0
Antanelis       Antanas         0         0
Medelis         Medis           47,5      2
```

5.4. Grafinės vartotojo sąsajos schema



5.5. Grafinės vartotojo sąsajos elementų pakeistos savybės

Komponentas	ID	text	Back Color	OnClick	TextMode
Button	Button1	START		Button1_Click	
TextBox	TextBox3				MultiLine
TextBox	TextBox1				MultiLine
TextBox	TextBox2				MultiLine

5.6. Klasių diagramos

Diena
+data : string +pavarde : string +vardas : string + atvykLaik : DateTime + konsultacTrukm : DateTime
+ Diena(int data : string, in pavarde: string, in vardas: string, in atvykLaik : DateTime, in konsultTrukm : DateTime) + ToString()

Seimas
+pavarde : string +vardas : string +dienosData:string +budejimasPR:DateTime +budejimasPB:DateTIme +tukme:double +rinkejai:integer
+Seimas(in pavarde : string, in vardas : string, in dienosData:string, in budejimasPR:DateTime, in budejimasPB:DateTime) + ToString() + CompareTo(in other : Diena) + Equals(in other : Diena) +Seimas(in pavarde : string, in vardas : string, in dienosData:string, in budejimasPR:DateTime, in budejimasPB:DateTime) + ToString() +Equals(in other : Seimas)

5.7. Programos vartotojo vadovas

Viename faile duota informacija apie seimo narių budėjimo grafiką: seimo nario pavardė ir vardas, dienos data, budėjimo priimamajame laikas(pradžia, pabaiga). Kituose failuose informacija apie lankytojus: pirmoje eilutėje data, tolimesnėse informacija apie lankytoją: pavardės, vardai, atvykimo laikas, konsultacijos trukmė.

Paspaudus „SART" mygtuką, programa surašo pradinius duomenis ir sukuria naują saršą, kuriame yra seimo nario vardas, pavardė, vidutinė konsultacijos trukmė ir vidutinis apsilankančių rinkėjų skaičius per dieną.

Sarašas surikiuotas pagal trukmę ir apsilankiusių lankytojų skaičių.