

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Objektinis programavimas II (P175B123)
Darbų aplankas

Atliko:

IFF 8/3 gr. studentas

Domantas Greičius

2019 m. balandis 1 d.

Priėmė:

Doc. Giedrius Ziberkas

TURINYS


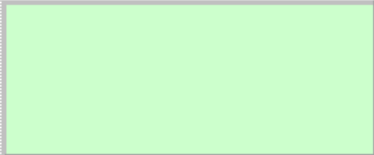
1. Rekursija (L1)	3
1.1. Darbo užduotis	3
1.2. Grafinės vartotojo sąsajos schema	3
1.3. Sąsajoje panaudotų komponentų keičiamos savybės	4
1.4. Klasių diagrama	5
1.5. Programos vartotojo vadovas	6
1.6. Programos tekstas	6
1.7. Pradiniai duomenys ir rezultatai	12
1.8. Dėstytojo pastabos	13
2. Dinaminis atminties valdymas (L2)	14
2.1. Darbo užduotis	14
2.2. Grafinės vartotojo sąsajos schema	14
2.3. Sąsajoje panaudotų komponentų keičiamos savybės	15
2.4. Klasių diagrama	16
2.5. Programos vartotojo vadovas	18
2.6. Programos tekstas	18
2.7. Pradiniai duomenys ir rezultatai	43
2.8. Dėstytojo pastabos	49
3. Bendrinės klasės ir sąsajos (L3)	50
3.1. Darbo užduotis	50
3.2. Grafinės vartotojo sąsajos schema	50
3.3. Sąsajoje panaudotų komponentų keičiamos savybės	50
3.4. Klasių diagrama	50
3.5. Programos vartotojo vadovas	50
3.6. Programos tekstas	50
3.7. Pradiniai duomenys ir rezultatai	50
3.8. Dėstytojo pastabos	50
4. Kolekcijos ir išimčių valdymas (L4)	50
4.1. Darbo užduotis	50
4.2. Grafinės vartotojo sąsajos schema	50
4.3. Sąsajoje panaudotų komponentų keičiamos savybės	51
4.4. Klasių diagrama	51
4.5. Programos vartotojo vadovas	51
4.6. Programos tekstas	51
4.7. Pradiniai duomenys ir rezultatai	51
4.8. Dėstytojo pastabos	51
5. Deklaratyvusis programavimas (L5)	51
5.1. Darbo užduotis	51
5.2. Grafinės vartotojo sąsajos schema	51
5.3. Sąsajoje panaudotų komponentų keičiamos savybės	51
5.4. Klasių diagrama	52
5.5. Programos vartotojo vadovas	52
5.6. Programos tekstas	52
5.7. Pradiniai duomenys ir rezultatai	52
5.8. Dėstytojo pastabos	52

1. Rekursija (L1)

1.1. Darbo užduotis

LD_9. Kurjeris. Studentas įsidarbina kurjeriu firmoje, turinčioje n punktų, į kuriuos jis laiku privalo pristatyti gaminius. Kurjerio algos dydis yra pastovus ir nepriklauso nuo kelionės trukmės. Norėdamas įvertinti bendrą kelionių kainą, studentas apskaičiavo kelionės kainą tarp bet kurių dviejų punktų. Užduotis – surasti tokį kelionės maršrutą, kuris praeitų tik po vieną kartą pro kiekvieną punktą ir kurio bendra punktų apvažiavimo kaina būtų mažiausia, t.y. kad studentui daugiau liktų pelno. Studento kelionės pradžia visą laiką yra pirmasis punktas. Ten jis ir turi grįžti. Duomenys surašyti tekstiniam failui 'U3.txt'. Pirmoje failo eilutėje parašytas sveikasis skaičius n ($5 \leq n \leq 50$), nurodantis kvadratinės matricos dydį. Toliau eilutėmis užrašyta matrica $K(n,n)$, kurioje išvardinta kelionių kaina tarp atskirų punktų, kur $K[i,j]=K[j,i]$ ir reiškia kelionės kaštus tarp punktų i ir j . Rezultatus išvesti į ekraną, kuriame būtų išvardinti punktai jų applanckymo eilės tvarka ir kelionės kaina.

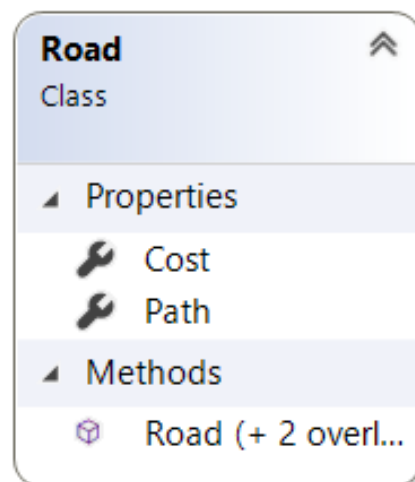
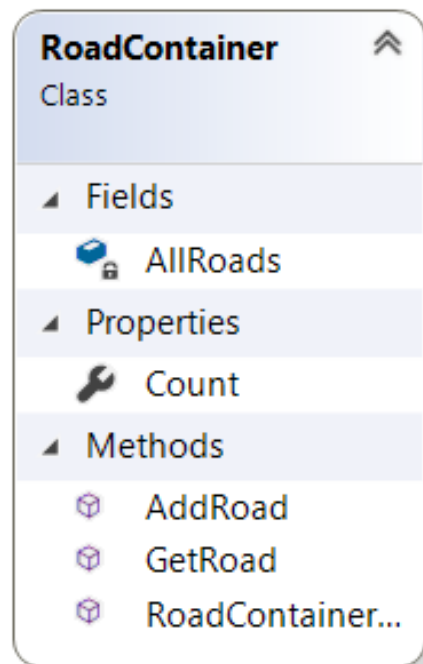
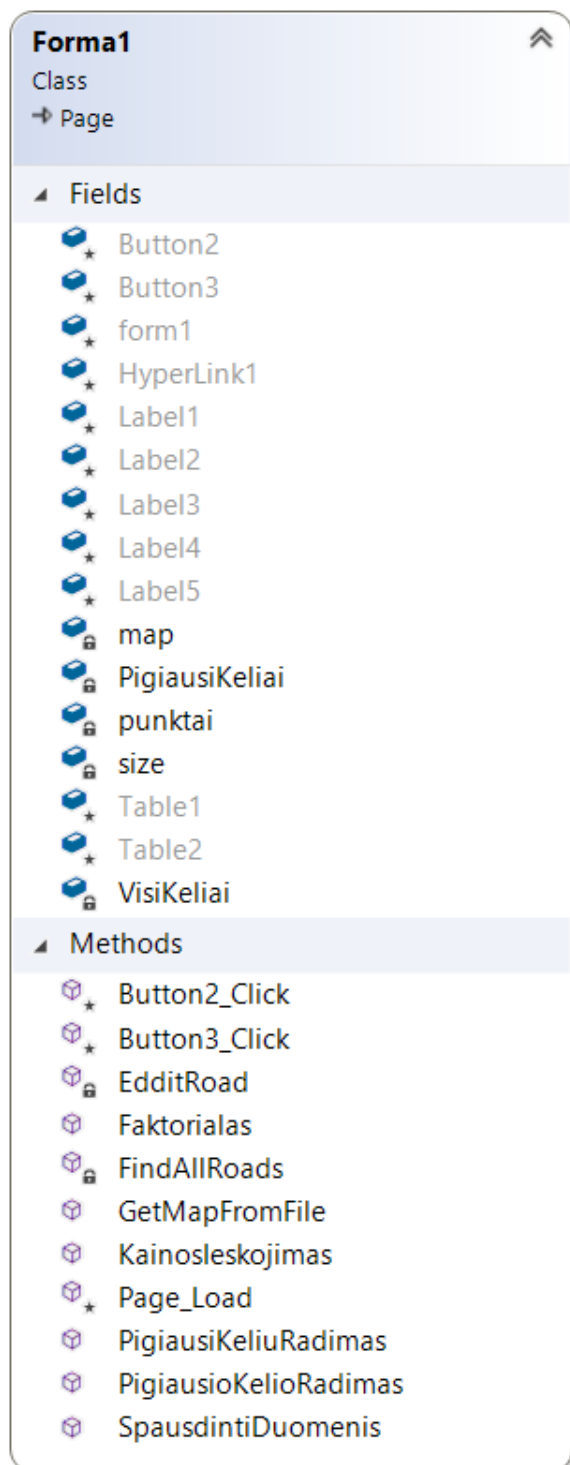
1.2. Grafinės vartotojo sąsajos schema

Domantas Greičius IFF8/3	Label14
Laboras L1-U9	Label11
Nuoroda į užduotį	HyperLink1
<input type="button" value="Nuskaityti duomenis"/>	Button2
Pradiniai duomenys(žemėlapis):	Label15
	Table1
Trumpiasi keliai:	Label13
	Table2
Pigiausio kelio/kelių kaina yra:	Label12
<input type="button" value="Ištrinti"/>	Button3

1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label	ID Text	Label4 Domantas Greičius IFF8/3
Label	ID Text	Label1 Laboras L1-U9
HyperLink	ID Text NavigationUrl	HyperLink1 Nuoroda į užduotį https://moodle.ktu.edu/pluginfile.php/7837/mod_resource/content/9/Uzduotys_rekursijai.pdf
Button	ID Text Width	Button2 Nuskaityti duomenis 206px
Label	ID Text	Label5 Pradiniai duomenys(žemėlapis):
Table	ID Height Width Gridlines BorderWidth BorderStyle BorderColor BackColor	Table1 92px 234px Both 3px Solid Black #FFFFCC
Label	ID Text	Label3 Trumpiausi keliai:
Table	ID Height Width GridLines BorderWidth BorderStyle BackColor	Table2 99px 235px Both 3px Solid #CCFFCC
Label	ID Text	Label2 Pigiausio kelio/kelių kaina yra:
Button	ID Text Width	Button3 Ištrinti 213px

1.4. Klasių diagrama



1.5. Programos vartotojo vadovas

Ši programa apskaičiuoja trumpiausią kelią, kaip reiktų aplankyti punktus, kad išvežiotojas išleistų mažiausiai pinigų ant kuro sanaudų.

Pradinių duomenų file'e „DuomenysA.txt“ pirmoje eilutėje reikia nurodyti kvadratinės matricos dydį, o tolesnėse eilutėse punkto kainą į kitus punktus. Tinkamai surašius duomenis galima paleisti programą.

Paleidę programą, spaudžiame mygtuką „Nuskaityti duomenis“. Tada ant ekrano pamatysime lentelę su nuskaityto žemėlapiu pradiniais duomenimis(kainą iš punktų į kitus punktus), o po juo bus kita lentelė, kur bus surašyta trumpiausių punktų seka, o po ja kelionės kaina. Taip pat, norėdami iš arčiau pamatyti pradinius duomenis ir rezultatus galime nueiti į programos App_Data folder'į ir atsidaryti „DuomSpausd.txt“ file'ą.

1.6. Programos tekstas

Road.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Laboras1
{
    /// <summary>
    /// Klasė skirta laikyti kelio duomenims
    /// </summary>
    public class Road
    {
        public string Path { get; set; }
        public int Cost { get; set; }

        /// <summary>
        /// Tuschias kelio konstruktorius
        /// </summary>
        public Road()
        {
            Path = String.Empty;
            Cost = 0;
        }

        /// <summary>
        /// Konstruktorius su kelio eiliškumu
        /// </summary>
        /// <param name="p">Kelio eiliškumas</param>
        public Road(string p)
        {
            Path = p;
            Cost = 0;
        }

        /// <summary>
        /// Konstruktorius su parametrais.
        /// </summary>
        /// <param name="p">Kelio eiliškumas</param>
        /// <param name="cost">Kelio kaina</param>
        public Road(string p, int cost)
        {
            Path = p;
            Cost = cost;
        }
    }
}
```

RoadContainer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Laboras1
{
    /// <summary>
    /// Kelio konteinerinė klasė
    /// </summary>
    public class RoadContainer
    {
        private Road[] AllRoads;
        public int Count { get; private set; }

        /// <summary>
        /// Tuschias konstruktorius
        /// </summary>
        public RoadContainer()
        {
            AllRoads = new Road[30];
            Count = 0;
        }

        /// <summary>
        /// Konstruktorius su dydžiu.
        /// </summary>
        /// <param name="size">Masyvo dydis</param>
        public RoadContainer(int size)
        {
            AllRoads = new Road[size];
            Count = 0;
        }

        /// <summary>
        /// Funkcija, kuri prideda kelią.
        /// </summary>
        /// <param name="r"></param>
        public void AddRoad(Road r)
        {
            AllRoads[Count] = r;
            Count++;
        }

        /// <summary>
        /// Funkcija, kuri gražina kelią.
        /// </summary>
        /// <param name="index"></param>
        /// <returns></returns>
        public Road GetRoad(int index)
        {
            return AllRoads[index];
        }
    }
}
```

Formal.aspx.cs

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

namespace Laboras1
{
    public partial class Formal : System.Web.UI.Page
    {
        static int size;
        // Žemėlapis(kvadratinės matricos) dydis
        int[][] map;
        // Žemėlapis koordinatėse esančios kainos
        int[] punktai;
        // Punktai
        RoadContainer VisiKeliai;
        // Visų kelių konteineris
        RoadContainer PigiausiKeliai;
        // Pigiausių kelių konteineris

        protected void Page_Load(object sender, EventArgs e)
        {
            // Prasidėjus programai nerodomos tuščios lentelės.
            Table1.Visible = false;
            Table2.Visible = false;
            Label2.Visible = false;
            Label3.Visible = false;
            Label5.Visible = false;
            Button3.Visible = false;
        }

        /// <summary>
        /// Funkcija, kuri skaičiuoja faktorialą(skirtas nusakyti visų įmanomų
        /// kelių kombinacijų skaičiui)
        /// </summary>
        /// <param name="number">Counter'is</param>
        /// <param name="dydis">Sandaugos dydis</param>
        /// <returns>Faktorialas</returns>
        public static int Faktorialas(int number, int dydis)
        {
            if (dydis <= 1)
            {
                return number;
            }
            return dydis * Faktorialas(number, (dydis - 1));
        }

        /// <summary>
        /// Rekursinė funkcija, kuri randa visus kelių variantus(Heap Algorithm).
        /// </summary>
        /// <param name="a">Punktai</param>
        /// <param name="puntuKiekis">Punktų kiekis</param>
        /// <param name="size">Žemėlapis ilgis</param>
        /// <param name="VisiKeliai">Kelių konteineris</param>
        /// <param name="map">Žemėlapis</param>
    }
}
```



```

static void FindAllRoads(int[] punktai, int punktuKiekis, int size,
                        RoadContainer VisiKeliai, int[][]map)
{
    // if size punktuKiekis 1 then prints the obtained
    // permutation
    if (punktuKiekis == 1)
        EdditRoad(punktai, VisiKeliai, map);

    for (int i = 0; i < punktuKiekis; i++)
    {
        FindAllRoads(punktai, punktuKiekis - 1, size, VisiKeliai, map);

        // if punktuKiekis is odd, swap first and last
        // element
        if (punktuKiekis % 2 == 1)
        {
            int temp = punktai[0];
            punktai[0] = punktai[punktuKiekis - 1];
            punktai[punktuKiekis - 1] = temp;
        }

        // If punktuKiekis is even, swap i'th and last
        // element
        else
        {
            int temp = punktai[i];
            punktai[i] = punktai[punktuKiekis - 1];
            punktai[punktuKiekis - 1] = temp;
        }
    }
}

/// <summary>
/// Funkcija, kuri papildo ir prideda kelią į konteinerį.
/// </summary>
/// <param name="a">Punktai</param>
/// <param name="VisiKeliai">Kelių konteineris</param>
/// <param name="map">Žemėlapis</param>
static void EdditRoad(int[] a, RoadContainer VisiKeliai, int[][]map)
{
    string s = "0"; // Priskiriamas pirmas punktas

    // Kuriamas kelias
    for (int i = 0; i < a.Length; i++)
    {
        s += Convert.ToString(a[i]);
    }
    s = s.Insert(s.Length, "0"); // Priskiriamas paskutinis punktas
    Road r = new Road(s);
    KainosIeskojimas(r, map);
    VisiKeliai.AddRoad(r);
}

/// <summary>
/// Funkcija, kuri randa visus pigiausias kelias.
/// </summary>
/// <param name="keliai">Kelių konteineris</param>
/// <returns>Užpildytas konteineris su pigiausiais keliais</returns>
public RoadContainer PigiausiKeliuRadimas(RoadContainer keliai)
{
    // Randa pigiausią kelią ir tikrina ar yra kelių su tokia pačia kaina.
    Road pigiausias = PigiausioKelioRadimas(keliai);
    RoadContainer pigiausiKeliai = new RoadContainer(keliai.Count);

    for (int i = 0; i < keliai.Count; i++)

```



```

</style>
</head>
<body style="height: 550px">
  <form id="form1" runat="server">
    <div>
      &nbsp;
      <asp:Label ID="Label4" runat="server" Text="Domantas Greičius
IFF8/3"></asp:Label>
      <br />
      &nbsp; <asp:Label ID="Label1" runat="server" Text="Laboras L1-U9"></asp:Label>
      &nbsp; &nbsp; <br />
      &nbsp; <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="https://moodle.ktu.edu/pluginfile.php/7837/mod_resource/content/9/Uzd
uotys_rekursijai.pdf">Nuoroda į užduotį</asp:HyperLink>
      <br />
      <br />
      &nbsp; <asp:Button ID="Button2" runat="server" OnClick="Button2_Click"
Text="Nuskaityti duomenis" Width="206px" />
      &nbsp; <br />
      <asp:Label ID="Label5" runat="server" Text="Pradiniai
duomenys (žemėlapis):" Visible="False"></asp:Label>
      <asp:Table ID="Table1" runat="server" GridLines="Both" Height="92px"
Width="234px" BackColor="#FFFFCC" BorderColor="Black" BorderStyle="Solid"
BorderWidth="3px">
      </asp:Table>
      <br />
      <asp:Label ID="Label3" runat="server" EnableViewState="False"
Text="Trumpiasi keliai:" Visible="False"></asp:Label>
      <br />
      <asp:Table ID="Table2" runat="server" GridLines="Both" Height="99px"
Width="235px" BackColor="#CCFFCC" BorderStyle="Solid" BorderWidth="3px">
      </asp:Table>
      <asp:Label ID="Label2" runat="server" Visible="False">Pigiausio
kelio/kelių kaina yra:</asp:Label>
      <br />
      <br />
    </div>
    <asp:Button ID="Button3" runat="server" OnClick="Button3_Click"
Text="Ištrinti" Width="213px" />
  </form>
</body>
</html>

```

1.7. Pradiniai duomenys ir rezultatai

Duomenys 1:

DuomenysA.txt

```
5
0 1 3 4 2
1 0 4 2 6
3 4 0 7 1
4 2 7 0 7
2 6 1 7 0
```

Rezultatai 1:

Domantas Greičius IFF8/3

Laboras L1-U9

[Nuoroda į užduotį](#)

Pradiniai duomenys(žemėlapis):

0	1	3	4	2
1	0	4	2	6
3	4	0	7	1
4	2	7	0	7
2	6	1	7	0

Trumpiausi keliai:

1	4	2	3	5	1
1	2	4	3	5	1
1	5	3	4	2	1
1	5	3	2	4	1

Pigiausio kelio/kelių kaina yra: 13

Ištrinti

Duomenys 2:

DuomenysB.txt

4

0 3 2 5

3 0 1 1

2 1 0 4

5 1 4 0

Rezultatai 2:

Domantas Greičius IFF8/3

Laboras L1-U9

[Nuoroda į užduotį](#)

Pradiniai duomenys(žemėlapis):

0	3	2	5
3	0	1	1
2	1	0	4
5	1	4	0

Trumpiausi keliai:

1	3	2	4	1
1	4	2	3	1

Pigiausio kelio/kelių kaina yra: 9

Ištrinti

1.8. Dėstytojo pastabos

Reiktų daugiau objektyškumo;

Patobulinti duomenų nuskaitymo funkciją;

1/3 Testas

5/6 Kodas

2. Dinaminis atminties valdymas (L2)

2.1. Darbo užduotis

LD_9. Prenumerata. Žmonės užsisako spaudą. Užsakymas vyksta metų ribose. Užsisakant spaudą, priskiriamas agentas, kuris pristatys užsakytą spaudą į namus. Tuo tikslu agentai gauna prenumeratorių sąrašus, kurie tikslinami kiekvieną mėnesį. Suskaičiuokite kiekvienam agentui nurodyto mėnesio (įvedamas klaviatūra) krūvį. Suformuokite sąrašą agentų, kurie dirba daugiau nei vidutinis krūvis nurodytam mėnesiui.

Duomenys:

- Tekstiniame faile U9a.txt yra tokia informacija apie prenumeratorius: prenumeratoriaus pavardė, adresas, laikotarpio pradžia (sveikasis skaičius 1..12), laikotarpio ilgis(sveikasis skaičius 1..12), leidinio kodas, leidinių kiekis, agento kodas.
- Tekstiniame faile U9b.txt yra informacija apie agentus: agento kodas, pavardė, vardas, adresas, telefonas.

Spausdinamas sąrašas turi būti surikiuotas pagal agento pavardę ir vardą abėcėlės tvarka.

Atspausdinkite kiekvienam agentui nurodyto mėnesio krūvį ir nešiojamos prenumeratos sąrašus. Jei yra agentų, kurių krūvis neviršija duotam mėnesiui nurodyto (įvedamas klaviatūra), pašalinkite juos iš sąrašo, jų krūvį paskirstydami agentams, kurių krūvis mažesnis už vidutinį. Atspausdinkite sąrašus agentams, kuriems pasikeitė krūvis.

2.2. Grafinės vartotojo sąsajos schema

Domantas Greičius IFF 8/3
L2-U9 Prenumerata

Įveskite, kurio mėnesio duomenis norite matyti:

Įveskite minimalų to mėnesio krūvį:

Vykdyti programą!

Pradiniai duomenys

Agentų sąrašas

###

Prenumeratorių sąrašas

###

[Label8]

Surikiuotas agentų sąrašas, kurie dirba daugiau nei vidutinis krūvis šitame mėnesyje:

###

Sarašas agentų su to mėnesio prenumeratoriais

###

Agentai, kuriems pasikeitė krūvis, paskirsčius pašalintų agentų krūvį:

###

Reset

Label4

TextBox1

TextBox2

Label5

Button1

Label1

Label2

Table1

Label3

Table2

Label8

Table3

Label6

Label9

Table5

Label7

Table4

Button2

2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label	ID Text	Label4 Įveskite, kurio mėnesio duomenis norite matyti:
TextBox	ID Height Width	TextBox1 16px 65px
TextBox	ID Height Width	TextBox2 16px 65px
Label	ID Text	Label5 Įveskite minimalų to mėnesio krūvį:
Button	ID Text Height Width	Button1 Vykdyti programą! 41px 235px
Label	ID Text	Label1 Pradiniai duomenys
Label	ID Text	Label2 Agentų sąrašas
Table	ID BackColor BorderStyle GridLines Height Width	Table1 White Solid Both 50px 53px
Label	ID Text	Label3 Prenumeratorių sąrašas
Table	ID BackColor BorderStyle GridLines Height Width	Table2 White Solid Both 47px 50px
Label	ID	Label8
Label	ID Text	Label6 Surikiuotas agentų sąrašas, kurie dirba daugiau nei vidutinis krūvis šitame mėnesyje:
Table	ID BackColor BorderStyle GridLines	Table3 White Solid Both
Label	ID Text	Label9 Sąrašas agentų su to mėnesio prenumeratoriais
Table	ID BackColor GridLines BorderStyle	Table5 White Both Solid

Label	ID Text	Label7 Agentai, kuriems pasikeitė krūvis, paskirsčius pašalintų agentų krūvį:
Table	ID BackColor BorderStyle GridLines	Table4 White Solid Both
Button	ID Text Height Width	Button2 Reset 47px 107px

2.4. Klasų diagrama

class Agentas
+ AgentoKodas : string + Pavarde : string + Vardas : string + Adresas : string + Telefonas : string + Kruvis : int
+ Agentas() + Agentas(in ak : string, in pav : string, in vard : string, in adresas : string, in telefonas : string) + SetKruvis(in kruvis: int) + ToString() {query} + AddKruvis() + < (in lhs : Agentas, in rhs : Agentas) : bool {query} + > (in lhs : Agentas, in rhs : Agentas) : bool {query} + == (in lhs : Agentas, in rhs : Agentas) : bool {query} + != (in lhs : Agentas, in rhs : Agentas) : bool {query}

class Prenumeratorius
+ Pavarde : string + Adresas : string + PradziaL : int + TrukmeL : int + LeidinioKodas : string + LeidiniuKiekis : int + PriklausantisAgentas : string
+ Prenumeratorius() + Prenumeratosiu(in pavarde : string, in adresas : string, in pr : int, in trukme : int, in leidKodas : string, in leidKiekis : int, in agentokodas : string) + PriskirtiAgentu (in kodas: string) + ToString() {query} + AddKruvis() + < (in lhs : Agentas, in rhs : Agentas) : bool {query} + > (in lhs : Agentas, in rhs : Agentas) : bool {query}

class Node in NodeList class
+ Data : T + Next : Node
+ Node() + Node(in data : T)

class NodeList<T>
<ul style="list-style-type: none"> - Start : Node - End : Node - Current : Node
<ul style="list-style-type: none"> + NodeList() + Delete() + NaikintiSarasa() + Rikiuoti() + FirstNode() : Node {query} + AddData(in data : T) + GautiT() : T {query} + Pradzia() + Sekantis() + ArYra() : bool {query}

class Formal
<ul style="list-style-type: none"> - DuomenysPrenumeratoriui : string - DuomenysAgentu : string - SpausdinimoFile : string - nurodytasMenesis : int - kruvis: int - vidurkis : double - PrenSarajas : NodeList<Prenumeratorius> - MenesioPren : NodeList<Prenumeratorius> - VisiAgentai : NodeList<Agentas> - MenesioAgentai : NodeList<Agentas> - AgentaiAtrinkti : NodeList<Agentas> - AgentaiPasikeite : NodeList<Agentas>
<ul style="list-style-type: none"> # Page_Load() # Button1_Click() # Button2_Click() - RodytiAgentusSuPrenumeratoriiaisTable(inout table : Table, agentai : NodeList<Agentas>, prenumeratoriai : NodeList<Prenumeratorius>) {query} - RodytiPrenumeratoriisTable(inout table : Table, prenumeratoriai : NodeList<Prenumeratorius>) {query} - RodytiAgentusTable(inout table : Table, agentai : NodeList<Agentas>) {query} - RastiPasikeitusius(in VisiAgentai : NodeList<Agentas>, in MenesioAgentai : NodeList<Agentas>, out AgentaiPasikeite : NodeList<Agentas>) - PasalintiAgentoPrenumeratoriis(inout prenumeratoriai : NodeList<Prenumeratorius>, in agentas : Agentas) - Apdoroti(inout MenesioAgentai : NodeList<Agentas>, inout pren : NodeList<Prenumeratorius>, in kruvis : int, in vidurkis : double) - AgentaiDaugiauNeiVidurkis(in agentai : NodeList<Agentas>) : NodeList<Agentas> - AgentuKruvioVidurkis(in Agentai : NodeList<Agentas>) : double - AtrinktiMenesioPrenumeratoriis(in prenumeratoriai : NodeList<Prenumeratorius>, inout prenumeratoriai : NodeList<Prenumeratorius>, innurodytasMenesis : double) - AgentuKruvioSkaiciavimas(inout agentai : NodeList<Agentas>, in prenumeratoriai : NodeList<Prenumeratorius>, in nurodytasMenesis : double) - SkaitytiPrenumeratoriis(out prenum : NodeList<Prenumeratorius>) - SkaitytiAgentus(out agentai : NodeList<Agentas>) - SpausdintiFileAgentus(in agentai : NodeList<Agentas>, in text : string) {query} - SpausdintiFilePrenumeratoriis(in prenumeratoriai : NodeList<Prenumeratorius>, in text : string) {query} - SpausdintiFileAgentusSuPrenSarajas(in agentai: NodeList<Agentas>, in prenumeratoriai : NodeList<Prenumeratorius>, in text : string) {query}

2.5. Programos vartotojo vadovas

Ši programa yra skirta stebėti agentų darbą ir dalinai jį keisti. Galima matyti visus įmonės agentų, prenumeratorių sąrašus, bei įvairiausias „statistikas“ bei rezultatus kaip: agentai, kurie dirba daugiau nei vidurkis nurodytą mėnesį, agentus su konkrečiai jiems priskirtais prenumeratoriais sąrašus...

Pradinių duomenų file'e „AgentaiA.txt“, galima rasti ir pridėti naujus įmonės agentus. Kitame pradinių duomenų file'e „Prenumeratoriai.txt“ yra prenumeratorių sąrašas su jų spaudos užsakymais, bei priskirtų agentų kodais, čia galima pridėti naujus prenumeratorius arba redaguoti esamus. Abu pradinių duomenų file'ai yra „AppData“ folder'yje.

Paleidę programą ant ekrano pamatysime teksto block'us, kur turime įvesti, kurio mėnesio duomenis mes norime matyti(nuo 1-12) ir koks turi būti minimalus to mėnesio krūvis agentams(agentus esančius žemiau minimalaus krūvio ribos pašalinsime ir jų krūvį paskirstysime kitiems agentams kurių krūvis mažiau nei bendras agentų vidurkis). Įvedus duomenis spaudžiame mygtuką „Vykdyti programą!“. Ant ekrano išvysime pradinius agentų ir prenumeratorių sąrašo duomenis iš duomenų file'ų. Toliau einant link puslapio apačios galime pamatyti agentų krūvio vidurkį, surikiuotą agentų sąrašą, kurie dirba daugiau nei vidurkis, visą agentų sąrašą su to mėnesio jiems priklausančiais prenumeratoriais, agentus, kuriems pasikeitė krūvis, paskirsčius pašalintų agentų krūvį sąrašą. Jei norime įvesti naujus duomenis, programos apačioje spaudžiame mygtuką „Reset“. Visus pradinius duomenis ir rezultatus galima rasti „Rezultatai“ folder'yje „PradiniaiDuom.txt“ file'e.

2.6. Programos tekstas

Agentas.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laboras2
{
    /// <summary>
    /// Klasė, kurioje aprašoma agento duomenys.
    /// </summary>
    class Agentas
    {
        public string AgentoKodas { get; private set; }
        public string Pavarde { get; private set; }
        public string Vardas { get; private set; }
        public string Adresas { get; private set; }
        public string Telefonas { get; private set; }

        public int Kruvis { get; private set; }

        /// <summary>
        /// Tuschias agento konstruktorius su pradinėmis reikšmėmis.
        /// </summary>
        public Agentas()
        {
            AgentoKodas = String.Empty;
            Pavarde = String.Empty;
            Vardas = String.Empty;
            Adresas = String.Empty;
            Telefonas = String.Empty;
        }

        /// <summary>
        /// Agento konstruktorius su paramentrais.
        /// </summary>
        /// <param name="ak">Agento kodas</param>
        /// <param name="pav">Pavardė</param>
        /// <param name="vard">Vardas</param>
    }
}
```

```

/// <param name="adresas">Adresas</param>
/// <param name="telefonas">Telefonas</param>
public Agentas(string ak, string pav, string vard, string adresas,
               string telefonas)
{
    AgentoKodas = ak;
    Pavarde = pav;
    Vardas = vard;
    Adresas = adresas;
    Telefonas = telefonas;
    Kruvis = 0;
}

/// <summary>
/// Funkcija, kuri nustato krūvį.
/// </summary>
/// <param name="kruvis">Krūvis</param>
public void SetKruvis(int kruvis)
{
    Kruvis = kruvis;
}

/// <summary>
/// Funkcija, kuri gražina agentą string formatu spausdinimui.
/// </summary>
/// <returns>string</returns>
public override string ToString()
{
    return String.Format("|{0, 11} | {1, 10} | {2, 10} | {3, 10} | {4, 10} |",
        AgentoKodas, Pavarde, Vardas, Adresas, Telefonas);
}

/// <summary>
/// Funkcija, kuri padidina krūvį.
/// </summary>
public void AddKruvis()
{
    Kruvis++;
}

/// <summary>
/// Palyginimo operatorius > , kuris lygina pagal
/// pavarde ir vardą.
/// </summary>
/// <param name="lhs">Agento objektas kairėje</param>
/// <param name="rhs">Agento objektas dešinėje</param>
/// <returns>true/false</returns>
public static bool operator >(Agentas lhs, Agentas rhs)
{
    if (String.Compare(lhs.Pavarde, rhs.Pavarde) == 0)
    {
        if (String.Compare(lhs.Vardas, rhs.Vardas) > 0)
        {
            return true;
        }
        return false;
    }
    if (String.Compare(lhs.Pavarde, rhs.Pavarde) > 0)
    {
        return true;
    }
    return false;
}

/// <summary>
/// Palyginimo operatorius, kuris lygina pagal

```

```

    /// pavarde ir vardas.
    /// </summary>
    /// <param name="lhs">Agento objektas kairėje</param>
    /// <param name="rhs">Agento objektas dešinėje</param>
    /// <returns>true/false</returns>
    public static bool operator <(Agentas lhs, Agentas rhs)
    {
        if (String.Compare(lhs.Pavarde, rhs.Pavarde) == 0)
        {
            if (String.Compare(lhs.Vardas, rhs.Vardas) < 0)
            {
                return true;
            }
            return false;
        }
        if (String.Compare(lhs.Pavarde, rhs.Pavarde) < 0)
        {
            return true;
        }
        return false;
    }

    /// <summary>
    /// Palyginimo operatorius
    /// </summary>
    /// <param name="lhs">Agento objektas kairėje</param>
    /// <param name="rhs">Agento objektas dešinėje</param>
    /// <returns>true/false</returns>
    public static bool operator ==(Agentas lhs, Agentas rhs)
    {
        if (lhs.Pavarde == rhs.Pavarde && lhs.Vardas == rhs.Pavarde)
        {
            return true;
        }
        return false;
    }

    /// <summary>
    /// Palyginimo operatorius
    /// </summary>
    /// <param name="lhs">Agento objektas kairėje</param>
    /// <param name="rhs">Agento objektas dešinėje</param>
    /// <returns>true/false</returns>
    public static bool operator !=(Agentas lhs, Agentas rhs)
    {
        if (lhs.Pavarde != rhs.Pavarde && lhs.Vardas != rhs.Vardas)
        {
            return true;
        }
        return false;
    }
}
}

```

Prenumeratorius.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laboras2
{
    /// <summary>
    /// Klasė skirta aprašyti prenumeratoriaus duomenims.
    /// </summary>
    class Prenumeratorius
    {
        public string Pavarde { get; private set; }
        public string Adresas { get; private set; }
        public int PradziaL { get; private set; }
        // Laikotario pradžia
        public int TrukmeL { get; private set; }
        // Laikotarpio ilgis/trukmė
        public string LeidinioKodas { get; private set; }
        public int LeidiniuKiekis { get; private set; }
        public string PriklausantisAgentas { get; private set; }

        /// <summary>
        /// Tusčias konstruktorius su pradinėmis reikšmėmis.
        /// </summary>
        public Prenumeratorius()
        {
            Pavarde = String.Empty;
            Adresas = String.Empty;
            PradziaL = 0;
            TrukmeL = 0;
            LeidinioKodas = String.Empty;
            LeidiniuKiekis = 0;
            PriklausantisAgentas = String.Empty;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="pavarde"></param>
        /// <param name="adresas"></param>
        /// <param name="pr"></param>
        /// <param name="trukme"></param>
        /// <param name="leidKodas"></param>
        /// <param name="leidKiekis"></param>
        /// <param name="agentokodas"></param>
        public Prenumeratorius(string pavarde, string adresas, int pr,
                                int trukme, string leidKodas, int leidKiekis,
                                string agentokodas)
        {
            Pavarde = pavarde;
            Adresas = adresas;
            PradziaL = pr;
            TrukmeL = trukme;
            LeidinioKodas = leidKodas;
            LeidiniuKiekis = leidKiekis;
            PriklausantisAgentas = agentokodas;
        }

        /// <summary>
        /// Funkcija, kuri pakeičia prenumeratoriaus kodą.
    }
}
```

```

    /// </summary>
    /// <param name="kodas"></param>
    public void PriskirtiAgentą(string kodas)
    {
        PriklausantisAgentas = kodas;
    }

    /// <summary>
    /// Funkcija, kuri gražina prenumeratorių string formatu spausdinimui.
    /// </summary>
    /// <returns></returns>
    public override string ToString()
    {
        return String.Format("| {0, 10} | {1, 15} |      {2, 7}      | {3, 11} | {4, 10} | {5, 9} | {6, 8} |",
            Pavarde, Adresas, PradziaL, TrukmeL, LeidinioKodas, LeidiniuKiekis,
            PriklausantisAgentas);
    }
}

```

NodeList.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laboras2
{
    /// <summary>
    /// Mazgo sąrašo klasė.
    /// </summary>
    /// <typeparam name="T">Tipas</typeparam>
    public class NodeList<T>
    {
        /// <summary>
        /// Mazgo klasė.
        /// </summary>
        public sealed class Node
        {
            public T Data;
            public Node Next;

            /// <summary>
            /// Tuschias konstruktorius
            /// </summary>
            public Node()
            {
            }

            /// <summary>
            /// Konstruktorius su parametrais.
            /// </summary>
            /// <param name="data">Objekto data</param>
            public Node(T data)
            {
                this.Data = data;
            }
        }
    }
}

```

```

private Node Start;           // Sąrašo pirmas elementas.
private Node End;             // Sąrašo paskutinis elementas.
private Node Current;         // Sąrašo tuo metu einamas elementas.

public NodeList()
{
    Start = null;
    End = null;
    Current = null;
}

/// <summary>
/// Funkcija, kuri ištrina esamą(Current) sąrašo elementą.
/// </summary>
public void Delete()
{
    if (Start == null) return;

    if (Start.Next == null)
    {
        Start = null;
        return;
    }

    // Ištrina jeigu pirmas elementas.
    if (Current == Start)
    {
        Current = Start;
        Start = Start.Next;
        return;
    }

    // Ištrina jeigu paskutinis elementas.
    if (Current == End)
    {
        Node node = Start;
        while (node.Next != End)
        {
            node = node.Next;
        }
        node.Next = null;
        End = node;
        return;
    }

    // Ištrina vidinį elementą.
    Node temp = Start;
    while (temp.Next.Next != null)
    {
        if (temp.Next == Current)
        {
            temp.Next = Current.Next;
            Current = temp;
            return;
        }
        temp = temp.Next;
    }
}

/// <summary>
/// Funkcija, kuri sunaikina sąrašą.
/// </summary>

```

```

public void NaikintiSarasa()
{
    while (Start != null)
    {
        Node d = Start;
        Start = Start.Next;
        d.Next = null;
    }
}

/// <summary>
/// Funkcija, kuri rikiuoja sąrašą(pagal pavardę ir vardą)
/// </summary>
public void Rikiuoti()
{
    Pradzia();
    for (Node first = FirstNode(); first != null; first = first.Next)
    {
        Node max = first;
        for (Node second = first; second != null; second = second.Next)
        {
            if ((second.Data as Agentas) < (max.Data as Agentas))
            {
                max = second;
            }
            T temp = first.Data;
            first.Data = max.Data;
            max.Data = temp;
        }
    }
}

/// <summary>
/// Funkcija, kuri gražina pirmą sąrašo elementą.
/// </summary>
/// <returns>Start node</returns>
public Node FirstNode()
{
    return Start;
}

/// <summary>
/// Funkcija, kuri prideda elementą į sąrašą.
/// </summary>
/// <param name="data">Objekto data</param>
public void AddData(T data)
{
    Node node = new Node(data);
    if (Start == null)
    {
        Start = node;
        End = node;
    }
    else
    {
        End.Next = node;
        End = node;
    }
}

/// <summary>
/// Funkcija, kuri gražina objektą.
/// </summary>
/// <returns>Objektas</returns>
public T GautiT()

```



```

        {
            return Current.Data;
        }

        /// <summary>
        /// Funkcija, kuri pradeda ciklą.
        /// </summary>
        public void Pradzia()
        {
            Current = Start;
        }

        /// <summary>
        /// Funkcija, kuri pereina į sekantį sąrašo elementą.
        /// </summary>
        public void Sekantis()
        {
            if (Current != null)
            {
                Current = Current.Next;
            }
        }

        /// <summary>
        /// Funkcija, kuri tikrina ar yra dar elementų sąrašė.
        /// </summary>
        /// <returns></returns>
        public bool ArYra()
        {
            return Current != null;
        }
    }
}

```

Formal.aspx.cs

```

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

namespace Laboras2
{
    public partial class Formal : System.Web.UI.Page
    {
        readonly string DuomenysPrenumeratorių = "App_Data/PrenumeratoriaiA.txt";
        // Pradinis prenumeratorių duomenų failas.
        readonly string DuomenysAgentų = "App_Data/AgentaiA.txt";
        // Pradinis agentų duomenų failas.
        readonly string SpausdinimoFailas = "Rezultatai/PradiniaiDuom.txt";
        // Pradinių duomenų ir rezultatų spausdinimo failas.

        int nurodytasMenesis;
        // Nurodytas mėnėsis pagal kurį reikės atrinkinėti.
        int kruvis;
        // Minimalus krūvis.
        double Vidurkis = 0;
        // Nurodyto mėnesio agentų krūvio vidurkis.
    }
}

```

```

        NodeList<Prenumeratorius> PrenSarasas = new NodeList<Prenumeratorius>();
// Visu prenumeratorių sąrašas.
        NodeList<Prenumeratorius> MenesioPren = new NodeList<Prenumeratorius>();
// Visu mėnesio prenumeratorių sąrašas.
        NodeList<Agentas> VisiAgentai = new NodeList<Agentas>();
// Visu agentų sąrašas.
        NodeList<Agentas> MenesioAgentai = new NodeList<Agentas>();
// Sąrašas atrinktų nurodyto mėnesio agentų.
        NodeList<Agentas> AgentaiAtrinkti = new NodeList<Agentas>();
// Sąrašas atrinktų nurodyto mėnesio agentų, kurių krūvis daugiau nei vidurkis.
        NodeList<Agentas> AgentaiPasikeite = new NodeList<Agentas>();
// Sąrašas atrinktų nurodyto mėnesio agentų, kurių krūvis pasikeitė (padidėjo)

// po krūvio paskirstymo.

    /// <summary>
    /// Puslapio užkrovimo komandos.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Visible = false;
        Label2.Visible = false;
        Label3.Visible = false;
        Label6.Visible = false;
        Label7.Visible = false;
        Label8.Visible = false;
        Label9.Visible = false;
        Table1.Visible = false;
        Table2.Visible = false;
        Table3.Visible = false;
        Table4.Visible = false;
        Button2.Visible = false;

    }

    /// <summary>
    /// Mygtuko "Reset" funkcionalumas.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button2_Click(object sender, EventArgs e)
    {
        //-----
        VisiAgentai.NaikintiSarasa();
        PrenSarasas.NaikintiSarasa();
        MenesioAgentai.NaikintiSarasa();
        AgentaiAtrinkti.NaikintiSarasa();
        AgentaiPasikeite.NaikintiSarasa();
        //-----

        Label1.Visible = false;
        Label2.Visible = false;
        Label3.Visible = false;
        Label6.Visible = false;
        Label7.Visible = false;
        Label8.Visible = false;
        Label9.Visible = false;
        Table1.Visible = false;
        Table2.Visible = false;
        Table3.Visible = false;
        Table4.Visible = false;
        Table5.Visible = false;
        Button2.Visible = false;
    }

```

```

//-----
}

/// <summary>
/// Mygtuko "Vykdyti programą!" funkcionalumas.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Button1_Click(object sender, EventArgs e)
{
    nurodytasMenesis = int.Parse(TextBox1.Text);
    kruvis = int.Parse(TextBox2.Text);

    if (File.Exists(Server.MapPath(SpausdinimoFile)))
    {
        File.Delete(Server.MapPath(SpausdinimoFile));
    }
//-----

    SkaitytiPrenumeratorius(PrenSarasas);
    SpausdintiIFilePrenumeratorius(PrenSarasas, "Pradiniai prenumeratorių
duomenys.");
    SkaitytiAgentus(VisiAgentai);
    SpausdintiIFileAgentus(VisiAgentai, "Pradiniai agentų duomenys.");
    RodytiAgentusTable(Table1, VisiAgentai);
    RodytiPrenumeratoriusTable(Table2, PrenSarasas);
    AtrinktiMenesioPrenumeratorius(PrenSarasas, MenesioPren, nurodytasMen-
esis);
//-----

    AgentuKruvioSkaiciavimas(VisiAgentai, MenesioPren, nurodytasMenesis);
    SkaitytiAgentus(MenesioAgentai);
    AgentuKruvioSkaiciavimas(MenesioAgentai, MenesioPren, nurodytasMen-
esis);
    RodytiAgentusSuPrenumeratoriaisTable(Table5, MenesioAgentai, Menesio-
Pren);
    //SpausdintiIFileAgentus(MenesioAgentai, "Agentai su to mėnesio
krūviu:");
    SpausdintiIFileAgentusSuPrenSarasais(MenesioAgentai, MenesioPren,
"Mėnesio agentai su jų prenumeratorių sąrašais.");
//-----

    AgentaiAtrinkti = AgentaiDaugiauNeiVidurkis(VisiAgentai);
    AgentaiAtrinkti.Rikiuoti();
    SpausdintiIFileAgentus(AgentaiAtrinkti, "Surikiuotas sąrašas atrinktų
nurodyto mėnesio agentų, kurių krūvis daugiau nei vidurkis.");
    Vidurkis = AgentuKruvioVidurkis(VisiAgentai);
    Label8.Text = String.Format("Agentų krūvio vidurkis yra: <b>{0}</b>",
Vidurkis);
//-----

    Apdoroti(MenesioAgentai, MenesioPren, kruvis, Vidurkis);
    //SpausdintiIFileAgentusSuPrenSarasais(MenesioAgentai, PrenSarasas,
"ekperimentas");
    RastiPasikeitusius(VisiAgentai, MenesioAgentai, AgentaiPasikeite);
    //SpausdintiIFileAgentus(AgentaiPasikeite, "Sąrašas atrinktų nurodyto
mėnesio agentų, kurių krūvis pasikeitė (padidėjo) po krūvio paskirstymo.");
    SpausdintiIFileAgentusSuPrenSarasais(AgentaiPasikeite, MenesioPren,
"Agentų sąrašas, kuriems pasikeitė krūvis, kartu su prenumeratorių sąrašais.");
//-----

    RodytiAgentusTable(Table3, AgentaiAtrinkti);
    RodytiAgentusSuPrenumeratoriaisTable(Table4, AgentaiPasikeite, PrenSa-
rasas);

```

```

Label1.Visible = true;
Label2.Visible = true;
Label3.Visible = true;
Label6.Visible = true;
Label7.Visible = true;
Label8.Visible = true;
Label9.Visible = true;
Button2.Visible = true;
//-----
}

/// <summary>
/// Funkcija, kuri lentelė parodo agentus su jiems priklausančiais pre-
numeratoriais.
/// </summary>
/// <param name="table">Table</param>
/// <param name="agentai">Agentų sąrašas</param>
/// <param name="prenumeratoriai">Prenumeratorių sąrašas</param>
void RodytiAgentusSuPrenumeratoriiais(Table table, NodeList<Agentas>
agentai, NodeList<Prenumeratorius> prenumeratoriai)
{
    if (agentai.FirstNode() != null)
    {
        for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
        {
            if (agentai.GautiT().Kruvis > 0)
            {
                TableRow row = new TableRow();
                TableCell cell = new TableCell();

//=====
                cell = new TableCell
                {
                    Text = "<b>Agento kodas</b>"
                };
                row.Cells.Add(cell);

                cell = new TableCell
                {
                    Text = "<b>Pavarde</b>"
                };
                row.Cells.Add(cell);

                cell = new TableCell
                {
                    Text = "<b>Vardas</b>"
                };
                row.Cells.Add(cell);

                cell = new TableCell
                {
                    Text = "<b>Adresas</b>"
                };
                row.Cells.Add(cell);

                cell = new TableCell
                {
                    Text = "<b>Tel nr.</b>"
                };
                row.Cells.Add(cell);

                cell = new TableCell
                {
                    Text = "<b>Kruvis</b>"

```

```

};
row.Cells.Add(cell);
//=====
table.Rows.Add(row);

//=====
row = new TableRow();
cell = new TableCell
{
    Text = agentai.GautiT().AgentoKodas
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = agentai.GautiT().Pavarde
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = agentai.GautiT().Vardas
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = agentai.GautiT().Adresas
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = agentai.GautiT().Telefonas
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = Convert.ToString(agentai.GautiT().Kruvis)
};
row.Cells.Add(cell);
//=====
table.Rows.Add(row);

row = new TableRow();

//=====
cell = new TableCell
{
    Text = "<b>Prenumeratoriaus pavarde</b>"
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = "<b>Adresas</b>"
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = "<b>Laikotarpio pradžia</b>"
};
row.Cells.Add(cell);

```

```

        cell = new TableCell
        {
            Text = "<b>Laikotarpio ilgis</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Leidinio kodas</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Leidiniu kiekis</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Priskirto agento kodas</b>"
        };
        row.Cells.Add(cell);

//=====

        table.Rows.Add(row);

        if (prenumeratoriai.FirstNode() != null)
        {
            for (prenumeratoriai.Pradzia(); prenumeratoriai.Ar-
Yra(); prenumeratoriai.Sekantis())
            {
                if (prenumeratoriai.GautiT().PriklausantisAgentas
== agentai.GautiT().AgentoKodas)
                {

//=====

                    row = new TableRow();
                    cell = new TableCell
                    {
                        Text = prenumeratoriai.GautiT().Pavarde
                    };
                    row.Cells.Add(cell);

                    cell = new TableCell
                    {
                        Text = prenumeratoriai.GautiT().Adresas
                    };
                    row.Cells.Add(cell);

                    cell = new TableCell
                    {
                        Text = Convert.ToString(
prenumeratoriai.GautiT().PradziaL)
                    };
                    row.Cells.Add(cell);

                    cell = new TableCell
                    {
                        Text = Convert.ToString(
prenumeratoriai.GautiT().TrukmeL)
                    };
                    row.Cells.Add(cell);

                    cell = new TableCell

```

```

        {
            Text = Convert.ToString(
                prenumeratoriai.GautiT().LeidinioKodas);
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = Convert.ToString(
                prenumeratoriai.GautiT().LeidiniuKiekis);
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = Convert.ToString(
                prenumeratoriai.GautiT().PriklausantisAgentas);
        };
        row.Cells.Add(cell);

//=====
        table.Rows.Add(row);
    }
}
}
table.Visible = true;
}

}

}

/// <summary>
/// Funkcija, kuri ant ekrano spausdina prenumetorių sąrašo table'a.
/// </summary>
/// <param name="table">Table'o index'as</param>
/// <param name="prenumeratoriai">Prenumeratorių sąrašas</param>
void RodytiPrenumeratoriumTable(Table table,
    NodeList<Prenumeratorius> prenumeratoriai)
{
    TableRow row = new TableRow();
    TableCell cell = new TableCell();
    if (prenumeratoriai.FirstNode() != null)
    {
        //=====
        cell = new TableCell
        {
            Text = "<b>Pavarde</b>";
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Adresas</b>";
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Laikotarpio pradžia</b>";
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Laikotarpio ilgis</b>";
        };
    }
}

```

```

};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = "<b>Leidinio kodas</b>"
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = "<b>Leidiniu kiekis</b>"
};
row.Cells.Add(cell);

cell = new TableCell
{
    Text = "<b>Agento kodas</b>"
};
row.Cells.Add(cell);

//=====
table.Rows.Add(row);

for (prenumeratoriai.Pradzia(); prenumeratoriai.ArYra();
    prenumeratoriai.Sekantis())
{
    //=====
    row = new TableRow();
    cell = new TableCell
    {
        Text = prenumeratoriai.GautiT().Pavarde
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = prenumeratoriai.GautiT().Adresas
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = Convert.ToString(prenumeratoriai.GautiT().PradziaL)
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = Convert.ToString(prenumeratoriai.GautiT().TrukmeL)
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = Convert.ToString(prenumeratoriai.GautiT().
            LeidinioKodas)
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = Convert.ToString(prenumeratoriai.GautiT().
            LeidiniuKiekis)
    };
    row.Cells.Add(cell);
}

```



```

        cell = new TableCell
        {
            Text = Convert.ToString(prenumeratoriai.GautiT().
                PriklausantisAgentas)
        };
        row.Cells.Add(cell);
        //=====
        table.Rows.Add(row);
    }

    }
    table.Visible = true;
}

/// <summary>
/// Funkcija, kuri ant ekrano spausdina agentų sąrašą table'a.
/// </summary>
/// <param name="table">Table'o index'as</param>
/// <param name="agentai">Agentų sąrašas</param>
void RodytiAgentusTable(Table table, NodeList<Agentas> agentai)
{
    TableRow row = new TableRow();
    TableCell cell = new TableCell();
    if (agentai.FirstNode() != null)
    {
        //=====
        cell = new TableCell
        {
            Text = "<b>Kodas</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Pavarde</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Vardas</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Adresas</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Tel nr.</b>"
        };
        row.Cells.Add(cell);

        cell = new TableCell
        {
            Text = "<b>Kruvis</b>"
        };
        row.Cells.Add(cell);
        //=====
        table.Rows.Add(row);
    }
}

```

```

for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
{
    //=====
    row = new TableRow();
    cell = new TableCell
    {
        Text = agentai.GautiT().AgentoKodas
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = agentai.GautiT().Pavarde
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = agentai.GautiT().Vardas
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = agentai.GautiT().Adresas
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = agentai.GautiT().Telefonas
    };
    row.Cells.Add(cell);

    cell = new TableCell
    {
        Text = Convert.ToString(agentai.GautiT().Kruvis)
    };
    row.Cells.Add(cell);
    //=====
    table.Rows.Add(row);
}
table.Visible = true;
}

/// <summary>
/// Funkcija, kuri randa pasikeitusius agentus, kurių krūvis padidėjo.
/// </summary>
/// <param name="VisiAgentai"></param>
/// <param name="MenesioAgentai"></param>
/// <param name="AgentaiPasikeite"></param>
void RastiPasikeitusius(NodeList<Agentas> VisiAgentai, NodeList<Agentas>
    MenesioAgentai, NodeList<Agentas> AgentaiPasikeite)
{
    for (VisiAgentai.Pradzia(); VisiAgentai.ArYra();
    VisiAgentai.Sekantis())
    {
        for (MenesioAgentai.Pradzia(); MenesioAgentai.ArYra();
        MenesioAgentai.Sekantis())
        {
            if (MenesioAgentai.GautiT() == VisiAgentai.GautiT())
            {
                if (MenesioAgentai.GautiT().Kruvis >
                    VisiAgentai.GautiT().Kruvis)

```

```

        {
            AgentaiPasikeite.AddData(MenesioAgentai.GautiT());
        }
    }
}

/// <summary>
/// Funkcija, kuri pašalina prenumeratoriaus kodą.
/// </summary>
/// <param name="prenumeratoriai">Prenumeratoriai</param>
/// <param name="agentas">Agentas, pagal kurio kodą pašalins</param>
void PasalintiAgentoPrenumeratorius(
    NodeList<Prenumeratorius> prenumeratoriai, Agentas agentas)
{
    for (prenumeratoriai.Pradzia(); prenumeratoriai.ArYra();
        prenumeratoriai.Sekantis())
    {
        if (prenumeratoriai.GautiT().PriklausantisAgentas ==
            agentas.AgentoKodas)
        {
            prenumeratoriai.GautiT().PriskirtiAgentą("-");
        }
    }
}

/// <summary>
/// Funkcija, kuri pašalina agentus esančius žemiau(arba lygu) nurodyto
krūvio ir paskirsto jų
/// krūvį agentams, kurių krūvis mažiau nei vidurkis.
/// </summary>
/// <param name="MenesioAgentai">Agentų sarašas</param>
/// <param name="kruvis">Nurodytas minimalus krūvis</param>
/// <param name="vidurkis">Agentų krūvio vidurkis</param>
void Apdoroti(NodeList<Agentas> MenesioAgentai, NodeList<Prenumeratorius>
    pren, int kruvis, double vidurkis)
{
    int kruvioSuma = 0;

    for (MenesioAgentai.Pradzia(); MenesioAgentai.ArYra();
        MenesioAgentai.Sekantis())
    {
        if (MenesioAgentai.GautiT().Kruvis <= kruvis)
        {
            kruvioSuma += MenesioAgentai.GautiT().Kruvis;
            PasalintiAgentoPrenumeratorius(pren, MenesioAgentai.GautiT());
            MenesioAgentai.Delete();
        }
    }

    for (int i = 0; i < kruvioSuma; i++)
    {
        for (MenesioAgentai.Pradzia(); MenesioAgentai.ArYra();
            MenesioAgentai.Sekantis())
        {
            if (MenesioAgentai.GautiT().Kruvis < vidurkis)
            {
                MenesioAgentai.GautiT().AddKruvis();
                for (pren.Pradzia(); pren.ArYra(); pren.Sekantis())
                {
                    if (pren.GautiT().PriklausantisAgentas == "-")
                    {
                        pren.GautiT().PriskirtiAgentą(
                            MenesioAgentai.GautiT().AgentoKodas);
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}

}

/// <summary>
/// Funkcija, kuri sudaro agentų sąrašą, kurių krūvis yra auksčiau nei
/// vidurkis.
/// </summary>
/// <param name="agentai">Agentų sąrašas</param>
/// <returns>Agentų sąrašas</returns>
NodeList<Agentas> AgentaiDaugiauNeiVidurkis(NodeList<Agentas> agentai)
{
    NodeList<Agentas> AgentaiDaugiau = new NodeList<Agentas>();
    double vidurkis = AgentuKruvioVidurkis(agentai);

    for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
    {
        if (agentai.GautiT().Kruvis > vidurkis)
        {
            AgentaiDaugiau.AddData(agentai.GautiT() as Agentas);
        }
    }

    return AgentaiDaugiau;
}

/// <summary>
/// Funkcija, kuri skaičiuoja agentų krūvio vidurkį.
/// </summary>
/// <param name="Agentai">Agentų sąrašas</param>
/// <returns>Vidurkis</returns>
double AgentuKruvioVidurkis(NodeList<Agentas> Agentai) // Tą menesį
{
    int suma = 0;
    int count = 0;
    for (Agentai.Pradzia(); Agentai.ArYra(); Agentai.Sekantis())
    {
        // Jei nulis, tai tada pasirinkta mėnesį, tas agentas neturi pre-
        numeratorių ir nėra įtaukiama į krūvio skaičiavimą.
        if (Agentai.GautiT().Kruvis > 0)
        {
            suma += Agentai.GautiT().Kruvis;
            count++;
        }
    }
    if (suma == 0)
    {
        return 0;
    }

    return Math.Round(((double)suma / count), 2);
}

/// <summary>
/// Funkcija, kuri atrenka nurodyto mėnesio prenumeratorius
/// </summary>
/// <param name="prenumeratoriai">Visi prenumeratoriai</param>
/// <param name="atrinkti">Atrinkti prenumeratoriai</param>
/// <param name="nurodytasMėnesis">Nurodytas atrinkimo mėnesis</param>
void AtrinktiMėnesioPrenumeratorius(NodeList<Prenumeratorius>
prenumeratoriai, NodeList<Prenumeratorius> atrinkti, int nurodytasMėnesis)

```

```

    {
        for (prenumeratoriai.Pradzia();prenumeratoriai.ArYra();
            prenumeratoriai.Sekantis())
        {
            if ((nurodytasMenesis >= prenumeratoriai.GautiT().PradziaL &&
                nurodytasMenesis <= (prenumeratoriai.GautiT().PradziaL +
                prenumeratoriai.GautiT().TrukmeL)) ||
                // Jei nurodytas mėnesis yra pvz 1, o prenumeratoriaus laiko-
                trapio pradžia pvz: 12, trukmė 3.
                (nurodytasMenesis <= prenumeratoriai.GautiT().PradziaL &&
                ((prenumeratoriai.GautiT().PradziaL + prenumeratoriai.GautiT().TrukmeL) - 12) >= nurodytasMenesis)
            )
            {
                atrinkti.AddData(prenumeratoriai.GautiT());
            }
        }
    }

    /// <summary>
    /// Funkcija, kuri skaičiuoja agentų krūvius.
    /// </summary>
    /// <param name="agentai">Agentų sąrašas</param>
    /// <param name="prenumeratoriai">Prenumeratorių sąrašas</param>
    /// <param name="nurodytasMenesis">Nurodytas atrinkimo mėnesis</param>
    void AgentuKruvioSkaiciavimas(NodeList<Agentas> agentai,
        NodeList<Prenumeratorius> prenumeratoriai, int nurodytasMenesis)
    {
        for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
        {
            agentai.GautiT().SetKruvis(0);
            for (prenumeratoriai.Pradzia(); prenumeratoriai.ArYra();
                prenumeratoriai.Sekantis())
            {
                if ((agentai.GautiT().AgentoKodas ==
                    prenumeratoriai.GautiT().PriklausantisAgentas &&
                    nurodytasMenesis >= prenumeratoriai.GautiT().PradziaL &&
                    nurodytasMenesis <= (prenumeratoriai.GautiT().PradziaL +
                    prenumeratoriai.GautiT().TrukmeL)) ||
                    // Jei nurodytas mėnesis yra pvz 1, o prenumeratoriaus
                    laikotrapio pradžia pvz: 12, trukmė 3.
                    (agentai.GautiT().AgentoKodas == prenumeratoriai.GautiT().PriklausantisAgentas &&
                    nurodytasMenesis <= prenumeratoriai.GautiT().PradziaL &&
                    ((prenumeratoriai.GautiT().PradziaL + prenumeratoriai.GautiT().TrukmeL) - 12) >= nurodytasMenesis)
                )
                {
                    agentai.GautiT().AddKruvis();
                }
            }
        }
    }

    /// <summary>
    /// Funkcija, kuri nuskaito prenumeratorių duomenis.
    /// </summary>
    /// <param name="prenumeratoriai"></param>
    void SkaitytiPrenumeratorius(NodeList<Prenumeratorius> prenum)
    {
        using (StreamReader sr = new StreamReader(Server.MapPath(
            DuomenysPrenumeratoriu), true))
        {
            string pavarde;

```

```

        string adresas;
        int lpradzia;
        int ilgisl;
        string leidkodas;
        int lkiekis;
        string agentas;

        string line = null;
        while ((line = sr.ReadLine()) != null)
        {
            string[] values = line.Split(new char[] { ';' },
                StringSplitOptions.RemoveEmptyEntries);
            pavarde = values[0];
            adresas = values[1];
            lpradzia = Convert.ToInt32(values[2]);
            ilgisl = Convert.ToInt32(values[3]);
            leidkodas = values[4];
            lkiekis = Convert.ToInt32(values[5]);
            agentas = values[6];
            Prenumeratorius p = new Prenumeratorius(pavarde, adresas,
                lpradzia, ilgisl, leidkodas, lkiekis, agentas);

            prenum.AddData(p);
        }
    }

    /// <summary>
    /// Funkcija, kuri nuskaityto agentų duomenis.
    /// </summary>
    /// <param name="agentai">Agentų sąrašas</param>
    void SkaitytiAgentus(NodeList<Agentas> agentai)
    {
        using (StreamReader sr = new StreamReader(Server.MapPath(
            DuomenysAgentu), true))
        {
            string agentoKodas;
            string pavarde;
            string vardas;
            string adresas;
            string telefonas;
            string line = null;
            while ((line = sr.ReadLine()) != null)
            {
                string[] values = line.Split(new char[] { ';' }, String
                   SplitOptions.RemoveEmptyEntries);
                agentoKodas = values[0];
                pavarde = values[1];
                vardas = values[2];
                adresas = values[3];
                telefonas = values[4];
                Agentas a = new Agentas(agentoKodas, pavarde, vardas, adresas,
                    telefonas);
                agentai.AddData(a);
            }
        }
    }

    /// <summary>
    /// Funkcija, kuri spausdina agentus į file'a.
    /// </summary>
    /// <param name="agentai">Agentų sąrašas</param>
    /// <param name="text">Tekstas skirtas nusakyti duomenims</param>
    void SpausdintiIFileAgentus(NodeList<Agentas> agentai, string text)

```

```

    {
        using (StreamWriter sr = new StreamWriter(Server.MapPath(Spausdini-
moFile), true))
        {
            sr.WriteLine("|-----|");
            sr.WriteLine("| " + text);
            sr.WriteLine("|Agento kodas| Pavardė | Vardas | Adresas
| Telefonas | Krūvis |");
            sr.WriteLine("|-----|");
            for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
            {
                string temp = String.Format(" {0, -7} |" ,agen-
tai.GautiT().Kruvis);
                if (agentai.GautiT().Kruvis > 0)
                {
                    sr.WriteLine(agentai.GautiT() + temp);
                    sr.WriteLine("|-----|");
                }
                else
                {
                    sr.WriteLine(agentai.GautiT());
                    sr.WriteLine("|-----|");
                }
            }
            sr.WriteLine();
            sr.WriteLine();
        }
    }

    /// <summary>
    /// Funkcija, kuri spausdina prenumeratorius į file'a.
    /// </summary>
    /// <param name="prenumeratoriai">Prenumeratorių sąrašas</param>
    /// <param name="text">Tekstas skirtas nusakyti duomenims</param>
    void SpausdintiIFilePrenumeratorius(NodeList<Prenumeratorius> prenum,
string text)
    {
        using (StreamWriter sr = new StreamWriter(Server.MapPath(Spausdini-
moFile), true))
        {
            sr.WriteLine("|-----|");
            sr.WriteLine("| " + text);
            sr.WriteLine("| Pavardė | Adresas | Laikotarpio pradžia
| Laikotarpio ilgis | Leidinio kodas | Leidinių kiekis | Agento kodas |");
            sr.WriteLine("|-----|");
            for (prenum.Pradzia(); prenum.ArYra(); prenum.Sekantis())
            {
                sr.WriteLine(prenum.GautiT());
                sr.WriteLine("|-----|");
            }
            sr.WriteLine();
            sr.WriteLine();
        }
    }

    /// <summary>
    /// Funkcija, kuri spausdina agentus su jų prenumeratoriais į file'a.

```

```

    /// </summary>
    /// <param name="agentai">Agentų sąrašas</param>
    /// <param name="prenumertatoriai">Prenumeratorių sąrašas</param>
    void SpausdintiIFileAgentusSuPrenSarasais(NodeList<Agentas> agentai,
    NodeList<Prenumeratorius> prenumertatoriai, string text)
    {
        using (StreamWriter sr = new StreamWriter(Server.MapPath(Spausdini-
moFile), true))
        {
            sr.WriteLine("|-----|");
            sr.WriteLine("| " + text);
            sr.WriteLine("|-----|");
            for (agentai.Pradzia(); agentai.ArYra(); agentai.Sekantis())
            {
                sr.WriteLine("|Agento kodas| Pavardė | Vardas | Adre-
sas | Telefonas | Krūvis |");
                sr.WriteLine("|-----|");
                string temp = String.Format("{0, -7} |", agen-
tai.GautiT().Kruvis);
                if (agentai.GautiT().Kruvis > 0)
                {
                    sr.WriteLine(agentai.GautiT() + temp);
                    sr.WriteLine("|-----|");
                    sr.WriteLine("| Pavardė | Adresas | Laikotarpio
pradžia | Laikotarpio ilgis | Leidinio kodas | Leidinių kiekis | Agento kodas |");
                    sr.WriteLine("|-----|");
                    for (prenumertatoriai.Pradzia(); prenumertatoriai.Ar-
Yra(); prenumertatoriai.Sekantis())
                    {
                        if (prenumertatoriai.GautiT().PriklausantisAgen-
tas==agentai.GautiT().AgentoKodas)
                        {
                            sr.WriteLine(prenumertatoriai.GautiT());
                        }
                    }
                }
                else
                {
                    sr.WriteLine(agentai.GautiT());
                    sr.WriteLine("|-----|");
                    sr.WriteLine("| Pavardė | Adresas | Laikotarpio
pradžia | Laikotarpio ilgis | Leidinio kodas | Leidinių kiekis | Agento kodas |");
                    sr.WriteLine("|-----|");
                    for (prenumertatoriai.Pradzia(); prenumertatoriai.ArYra();
prenumertatoriai.Sekantis())
                    {
                        if (prenumertatoriai.GautiT().PriklausantisAgentas ==
agentai.GautiT().AgentoKodas)
                        {
                            sr.WriteLine(prenumertatoriai.GautiT());
                        }
                    }
                }
            }
        }
    }

```



```

        sr.WriteLine(" |-----|");
        sr.WriteLine(" |-----|");
        sr.WriteLine(" |-----|");
        sr.WriteLine(" |-----|");
    }
    sr.WriteLine();
    sr.WriteLine();
}
}
}
}

```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Formal.aspx.cs" Inherits="Laboras2.Formal" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link rel="stylesheet" type="text/css" href="App_Themes/TemaA/Style.css"/>
</head>
<body>
    <form id="form1" runat="server">
        <div style="height: 732px">
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
            Domantas Greičius IFF 8/3
            <br />
            &nbsp;&nbsp;&~
            L2-U9 Prenumerata<br />
            &nbsp;&nbsp;&~
            <asp:Label ID="Label4" runat="server" Text="Įveskite, kurio mėnesio duomenis norite matyti:"></asp:Label>
            &nbsp;&nbsp;&~
            <asp:TextBox ID="TextBox1" type="number" min="1" max="12" runat="server" Height="16px" Width="65px"></asp:TextBox>
            <br />
            &nbsp;&nbsp;&~
            <asp:Label ID="Label5" runat="server" Text="Įveskite minimalų to mėnesio krūvį:"></asp:Label>
            &nbsp;&~
            <asp:TextBox ID="TextBox2" type="number" min="1" max="30" runat="server" Height="16px" Width="65px" ></asp:TextBox>
            <br />
            <br />
            &nbsp;&~<asp:Button ID="Button1" runat="server" Height="41px" OnClick="Button1_Click" Text="Vykdėti programą!" Width="235px" />
            &nbsp;&~&nbsp;&~&nbsp;&~
            <br />
            <asp:Label ID="Label1" runat="server" Text="Pradiniai duomenys"></asp:Label>
            <br />
            <asp:Label ID="Label2" runat="server" Text="Agentų sąrašas"></asp:Label>
            &nbsp;&~
            <asp:Table ID="Table1" runat="server" BorderStyle="Solid" GridLines="Both" Height="50px" Width="53px" BackColor="White">
                </asp:Table>
            <br />
            &nbsp;&~<asp:Label ID="Label3" runat="server" Text="Prenumeratorių sąrašas"></asp:Label>
```

```

        <asp:Table ID="Table2" runat="server" GridLines="Both" Height="47px"
Width="50px" BackColor="White" BorderStyle="Solid">
        </asp:Table>
        <br />
        <asp:Label ID="Label8" runat="server"></asp:Label>
        <br />
        <asp:Label ID="Label6" runat="server" Text="Surikiuotas agentų
sarašas, kurie dirba daugiau nei vidutinis krūvis šitame mėnesyje:"></asp:Label>
        <br />
        <asp:Table ID="Table3" runat="server" BackColor="White" Border-
Style="Solid" GridLines="Both">
        </asp:Table>
        <br />
        <br />
        <asp:Label ID="Label9" runat="server" Text="Sarašas agentų su to
mėnesio prenumeratoriais"></asp:Label>
        <asp:Table ID="Table5" runat="server" BackColor="White" Grid-
Lines="Both" BorderStyle="Solid">
        </asp:Table>
        <br />
        <asp:Label ID="Label7" runat="server" Text="Agentai, kuriems pasikeitė
krūvis, paskirsčius pašalintų agentų krūvį:"></asp:Label>
        <asp:Table ID="Table4" runat="server" BackColor="White" Border-
Style="Solid" GridLines="Both">
        </asp:Table>
        &nbsp;
        <asp:Button ID="Button2" runat="server" Height="47px" OnClick="But-
ton2_Click" Text="Reset" Width="107px" />
    </div>
</form>
</body>
</html>

```

Style.css

```

body {
    background-image: url('https://i.redd.it/r0ad288kma8x.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
    background-size: cover;
    height: 732px;
}

#Button2 {
margin:10px;
margin-left:50px;
}

#Label1, #Label2, #Label3, #Label6, #Label7, #Label8, #Label9 {
    margin: 50px;
    color: black;
    font-family: 'Segoe UI';
    font-weight: bold;
    font-size: 23px;
}

Table {
    margin-left: 20px;
    border-color:brown;
    border-width:4px;
    border-style:solid;
}

```

2.7. Pradiniai duomenys ir rezultatai

Duomenys 1:

AgentaiA.txt

```
7778;Agentas;Ambokas;kakz g;864444444
4542;Holmsas;Serlokas;Kkokik g.;863333333
123YY;Dzekis;Canas;Grazi g;862222222
0000;ZAgentas2;aaa;dsa g.;865555555
QWERTY;Agentas3;sss;sdsa;866666666
Cool;Agentas5;sss;ssds;867777777
SKrr;Agentas6;sssss;ffff;868888888
Noob;Agentas4;sss;sdsa;866666666
xaxa;EuEu;dasdsa;dasdsa;869999999
5115;Bondas;James;AAA g.;861111111
```

PrenumeratoriaiA.txt

```
Sakalas;Draugystes g;4;5;1111;7;5115
Petraitis;Mosedzio g;4;7;J4J4;5;5115
Jonaitis;Kazkokia g;4;2;asss;3;123YY
Hohonas;jopr g;1;10;1111;4;4542
Laimis;Gatvike g.;2;4;leidz;4;7778
Zmogus;Kauno g.;3;3;dvff;10;7778
PrenA;Kauno g.;3;3;dvff;10;7778
PrenB;Kauno g.;3;3;dvff;10;7778
PrenC;Kauno g.;3;3;dvff;10;5115
PrenD;Kauno g.;3;3;dvff;10;5115
PrenE;Kauno g.;3;3;dvff;10;0000
PrenG;Kauno g.;3;3;dvff;10;4542
PrenH;Kauno g.;3;3;dvff;10;5115
PrenI;Kauno g.;3;3;dvff;10;7778
PrenJ;Kauno g.;3;3;dvff;10;7778
PrenK;Kauno g.;3;3;dvff;10;7778
PrenK;Kauno g.;3;3;dvff;10;7778
PrenK;Kauno g.;3;3;dvff;10;5115
PrenK;Kauno g.;3;3;dvff;10;5115
PrenK;Kauno g.;3;3;dvff;10;7778
PrenK;Kauno g.;3;3;dvff;10;7778
PrenK;Kauno g.;3;3;dvff;10;SKrr
PrenK;Kauno g.;3;3;dvff;10;SKrr
PrenK;Kauno g.;3;3;dvff;10;Cool
PrenK;Kauno g.;3;3;dvff;10;4542
PrenK;Kauno g.;3;3;dvff;10;xaxa
PrenK;Kauno g.;3;3;dvff;10;QWERTY
PrenK;Kauno g.;3;3;dvff;10;QWERTY
PrenK;Kauno g.;3;3;dvff;10;QWERTY
PrenK;Kauno g.;3;3;dvff;10;QWERTY
PrenK;Kauno g.;3;3;dvff;10;0000
PrenK;Kauno g.;3;3;dvff;10;0000
PrenK;Kauno g.;3;3;dvff;10;0000
```

Rezultatai 1:

Domantas Greičius IFF 8/3

L2-U9 Prenumerata

Įveskite, kurio mėnesio duomenis norite matyti:

5

Įveskite minimalų to mėnesio krūvį:

1

Vykdyti programą!

Pradiniai duomenys Agentų sąrašas

Kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis
7778	Agentas	Ambokas	kakz g	864444444	0
4542	Holmsas	Serlokas	Kkokik g.	863333333	0
123YY	Dzekis	Canas	Grazi g	862222222	0
0000	ZAgentas2	aaa	dsa g.	865555555	0
QWERTY	Agentas3	sss	sdsa	866666666	0
Cool	Agentas5	sss	ssds	867777777	0
SKrr	Agentas6	sssss	ffff	868888888	0
Noob	Agentas4	sss	sdsa	866666666	0
xaxa	EuEu	dasdsa	dasdsa	869999999	0
5115	Bondas	James	AAA g.	861111111	0

Prenumeratorių sąrašas

Pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Agento kodas
Sakalas	Draugystes g	4	5	1111	7	5115
Petraitis	Mosedzio g	4	7	J4J4	5	5115
Jonaitis	Kazkokia g	4	2	asss	3	123YY
Hohonas	jopr g	1	10	1111	4	4542
Laimis	Gatvike g.	2	4	leidz	4	7778
Zmogus	Kauno g.	3	3	dvff	10	7778
PrenA	Kauno g.	3	3	dvff	10	7778

PrenA	Kauno g.	3	3	dvff	10	7778
PrenB	Kauno g.	3	3	dvff	10	7778
PrenC	Kauno g.	3	3	dvff	10	5115
PrenD	Kauno g.	3	3	dvff	10	5115
PrenE	Kauno g.	3	3	dvff	10	0000
PrenG	Kauno g.	3	3	dvff	10	4542
PrenH	Kauno g.	3	3	dvff	10	5115
PrenI	Kauno g.	3	3	dvff	10	7778
PrenJ	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	5115
PrenK	Kauno g.	3	3	dvff	10	5115
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	Cool
PrenK	Kauno g.	3	3	dvff	10	4542
PrenK	Kauno g.	3	3	dvff	10	xaxa
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	0000
PrenK	Kauno g.	3	3	dvff	10	0000
PrenK	Kauno g.	3	3	dvff	10	0000

Agentų krūvio vidurkis yra: 3.67

Surikiuotas agentų sąrašas, kurie dirba daugiau nei vidutinis krūvis šitame mėnesyje:

Kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis
7778	Agentas	Ambokas	kakz g	864444444	10
5115	Bondas	James	AAA g.	861111111	7
QWERTY	Agentas3	sss	sdsa	866666666	4
0000	ZAgentas2	aaa	dsa g.	865555555	4

Sarašas agentų su to mėnesio prenumeracine

Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
7778	Agentas	Ambokas	kakz g	864444444	10	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Laimis	Gatvike g.	2	4	leidz	4	7778
Zmogus	Kauno g.	3	3	dvff	10	7778
PrenA	Kauno g.	3	3	dvff	10	7778
PrenB	Kauno g.	3	3	dvff	10	7778
PrenI	Kauno g.	3	3	dvff	10	7778
PrenJ	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
PrenK	Kauno g.	3	3	dvff	10	7778
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
4542	Holmsas	Serlokas	Kkokik g.	863333333	3	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Hohonas	jopr g	1	10	1111	4	4542
PrenG	Kauno g.	3	3	dvff	10	4542
PrenK	Kauno g.	3	3	dvff	10	4542
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
123YY	Dzekis	Canas	Grazi g	862222222	1	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Jonaitis	Kazkokia g	4	2	asss	3	123YY
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
0000	ZAgentas2	aaa	dsa g.	865555555	4	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenE	Kauno g.	3	3	dvff	10	0000
PrenK	Kauno g.	3	3	dvff	10	0000
PrenK	Kauno g.	3	3	dvff	10	0000
PrenK	Kauno g.	3	3	dvff	10	0000
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
QWERTY	Agentas3	sss	sdsa	866666666	4	

PrenK	Kauno g.	5	5	dvff	10	0000
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
QWERTY	Agentas3	sss	sdsa	866666666	4	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
PrenK	Kauno g.	3	3	dvff	10	QWERTY
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
Cool	Agentas5	sss	ssds	867777777	1	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenK	Kauno g.	3	3	dvff	10	Cool
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
SKrr	Agentas6	sssss	ffff	868888888	2	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	SKrr
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
xaxa	EuEu	dasdsa	dasdsa	869999999	1	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenK	Kauno g.	3	3	dvff	10	xaxa
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
5115	Bondas	James	AAA g.	861111111	7	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Sakalas	Draugystes g	4	5	1111	7	5115
Petrailis	Mosedzio g	4	7	J4J4	5	5115
PrenC	Kauno g.	3	3	dvff	10	5115
PrenD	Kauno g.	3	3	dvff	10	5115
PrenH	Kauno g.	3	3	dvff	10	5115
PrenK	Kauno g.	3	3	dvff	10	5115
PrenK	Kauno g.	3	3	dvff	10	5115

Agentai, kuriems pasikeitė krūvis, paskirsčius pašalintų agentų krūvį:

Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
4542	Holmsas	Serlokas	Kkokik g.	863333333	4	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Jonaitis	Kazkokia g	4	2	asss	3	4542
Hohonas	jopr g	1	10	1111	4	4542
PrenG	Kauno g.	3	3	dvff	10	4542
PrenK	Kauno g.	3	3	dvff	10	4542
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
SKrr	Agentas6	sssss	ffff	868888888	4	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	SKrr
PrenK	Kauno g.	3	3	dvff	10	SKrr

Reset

Duomenys 2:

AgentaiB.txt

```
0000;Agentaxas;Zazalius;AAA g.;861111111
1111;Agentaxas;Ababa;BBB g.;862222222
2222;AgentasC;PavardenisC;CCC g.;863333333
3333;AgentasD;PavardenisD;DDD g.;864444444
```

PrenumeratoriaiB.txt

```
Prenumas1;Gatve1 g.;12;3;dvff;10;0000
Prenumas2;Gatve2 g.;11;3;dvff;10;0000
Prenumas3;Gatve3 g.;10;5;dvff;10;0000
Prenumas4;Gatve4 g.;8;3;dvff;10;0000
Prenumas5;Gatve5 g.;7;8;dvff;10;2222
Prenumas6;Gatve6 g.;12;3;dvff;10;1111
Prenumas7;Gatve7 g.;12;2;dvff;10;2222
Prenumas8;Gatve8 g.;1;5;dvff;10;1111
Prenumas9;Gatve9 g.;12;3;dvff;10;1111
Prenumas10;Gatve10 g.;12;6;dvff;10;1111
Prenumas11;Gatve10 g.;11;5;dvff;10;3333
Prenumas12;Gatve4 g.;8;3;dvff;10;3333
Prenumas13;Gatve4 g.;8;3;dvff;10;2222
Prenumas14;Gatve4 g.;8;3;dvff;10;1111
```


Rezultatai 2:

Domantas Greičius IFF 8/3

L2-U9 Prenumerata

Įveskite, kurio mėnesio duomenis norite matyti:

1

Įveskite minimalų to mėnesio krūvį:

2

Vykdyti programą!

Pradiniai duomenys

Agentų sąrašas

Kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis
0000	Agentaxas	Zazalius	AAA g.	861111111	0
1111	Agentaxas	Ababa	BBB g.	862222222	0
2222	AgentasC	PavardenisC	CCC g.	863333333	0
3333	AgentasD	PavardenisD	DDD g.	864444444	0

Prenumeratorių sąrašas

Pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Agento kodas
Prenumas1	Gatve1 g.	12	3	dvff	10	0000
Prenumas2	Gatve2 g.	11	3	dvff	10	0000
Prenumas3	Gatve3 g.	10	5	dvff	10	0000
Prenumas4	Gatve4 g.	8	3	dvff	10	0000
Prenumas5	Gatve5 g.	7	8	dvff	10	2222
Prenumas6	Gatve6 g.	12	3	dvff	10	1111
Prenumas7	Gatve7 g.	12	2	dvff	10	2222
Prenumas8	Gatve8 g.	1	5	dvff	10	1111

Prenumas8	Gatve8 g.	1	5	dvff	10	1111
Prenumas9	Gatve9 g.	12	3	dvff	10	1111
Prenumas10	Gatve10 g.	12	6	dvff	10	1111
Prenumas11	Gatve10 g.	11	5	dvff	10	3333
Prenumas12	Gatve4 g.	8	3	dvff	10	3333
Prenumas13	Gatve4 g.	8	3	dvff	10	2222
Prenumas14	Gatve4 g.	8	3	dvff	10	1111

Agentų krūvio vidurkis yra: 2.5

Surikiuotas agentų sąrašas, kurie dirba daugiau nei vidutinis krūvis šitame mėnesyje:

Kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis
1111	Agentaxas	Ababa	BBB g.	862222222	4
0000	Agentaxas	Zazalius	AAA g.	861111111	3

Sarašas agentų su to mėnesio prenumeratoriais

Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
0000	Agentaxas	Zazalius	AAA g.	861111111	3	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Prenumas1	Gatve1 g.	12	3	dvff	10	0000
Prenumas2	Gatve2 g.	11	3	dvff	10	0000
Prenumas3	Gatve3 g.	10	5	dvff	10	0000
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
1111	Agentaxas	Ababa	BBB g.	862222222	4	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Prenumas6	Gatve6 g.	12	3	dvff	10	1111
Prenumas8	Gatve8 g.	1	5	dvff	10	1111
Prenumas8	Gatve8 g.	1	5	dvff	10	1111
Prenumas9	Gatve9 g.	12	3	dvff	10	1111
Prenumas10	Gatve10 g.	12	6	dvff	10	1111
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
2222	AgentasC	PavardenisC	CCC g.	863333333	2	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Prenumas5	Gatve5 g.	7	8	dvff	10	2222
Prenumas7	Gatve7 g.	12	2	dvff	10	2222
Agento kodas	Pavarde	Vardas	Adresas	Tel nr.	Kruvis	
3333	AgentasD	PavardenisD	DDD g.	864444444	1	
Prenumeratoriaus pavarde	Adresas	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidiniu kiekis	Priskirto agento kodas
Prenumas11	Gatve10 g.	11	5	dvff	10	3333

Agentai, kuriems pasikeitė krūvis, paskirsčius pašalintų agentų krūvį:

Reset

2.8. Dėstytojo pastabos

Negalimas public FirstNode() metodas.

Daugiau naudoti css.

Geriau apsirašyti eilučių formavimą.

Testas 3/3

Kodas 6/6

3. Bendrinės klasės ir sąsajos (L3)

3.1. Darbo užduotis

3.2. Grafinės vartotojo sąsajos schema

3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

3.4. Klasių diagrama

3.5. Programos vartotojo vadovas

3.6. Programos tekstas

3.7. Pradiniai duomenys ir rezultatai

3.8. Dėstytojo pastabos

4. Kolekcijos ir išimčių valdymas (L4)

4.1. Darbo užduotis

4.2. Grafinės vartotojo sąsajos schema

4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

4.4. Klasių diagrama

4.5. Programos vartotojo vadovas

4.6. Programos tekstas

4.7. Pradiniai duomenys ir rezultatai

4.8. Dėstytojo pastabos

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

5.2. Grafinės vartotojo sąsajos schema

5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

--	--	--

5.4. *Klasių diagrama*

5.5. *Programos vartotojo vadovas*

5.6. *Programos tekstas*

5.7. *Pradiniai duomenys ir rezultatai*

5.8. *Dėstytojo pastabos*