KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Objektinio programavimo pagrindai II (P175B502)

Laboratorinio darbo ataskaita

Atliko:

IFIN-1/2 gr. studentas

Martynas Burneika

2022 m. gegužės 23 d.

Priėmė:

Dėst. Vytautas Bukšnaitis

KAUNAS 2022

TURINYS

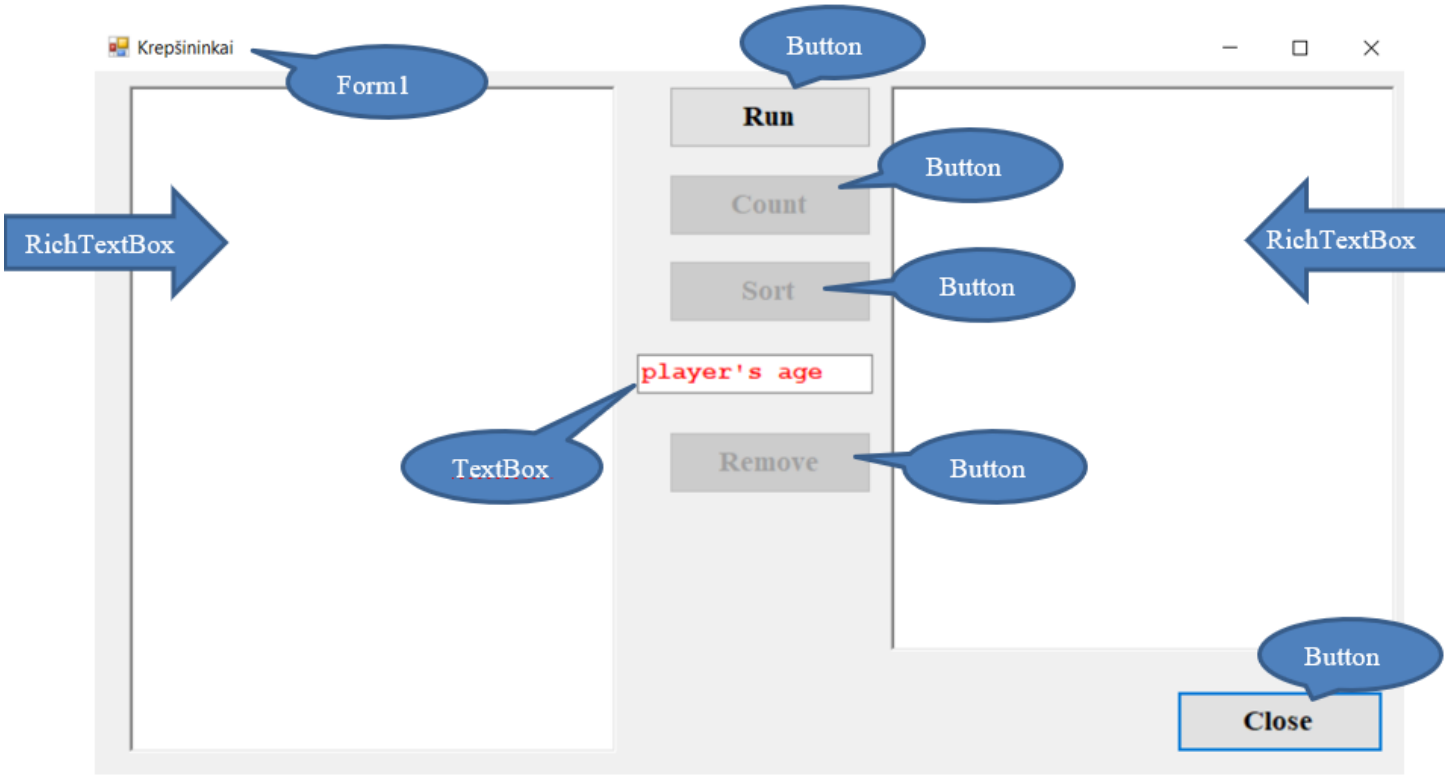
1. Grafinė vartotojo sąsaja ir algoritmų taikymas (L1)

1.1. Darbo užduotis

U1-2. Krepšinis. Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Turime dviejų mokyklų duomenis.

- Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis kiekvienoje mokykloje.
- Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.
- Surikiuokite rezultatų sąrašą amžiaus didėjimo tvarka.
- Pašalinkite iš rezultatų sąrašo krepšininkus, kurių amžius yra didesnis už nurodytą klaviatūra.

1.2. Grafinės vartotojo sąsajos schema ir paveikslas



1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Form1	Text	Krepšininkai
Run (button)	Text Font	Run Times New Roman, Bold, 14, script Baltic
Count (button)	Text Font	Count Times New Roman, Bold, 14, script Baltic
Sort (button)	Text Font	Sort Times New Roman, Bold, 14, script Baltic
Remove (button)	Text Font	Remove Times New Roman, Bold, 14, script Baltic
Type (TextBox)	Text Font	Player‘s age Courier New, Bold, 12, script Baltic

	ForeColor	Red
results (RichTextBox)	Font	Courier New, Bold, 11, script Baltic
results2 (RichTextBox)	Font	Courier New, Bold, 11, script Baltic
close (button)	Text Font	Close Times New Roman, Bold, 14, script Baltic

1.4. Programos vartotojo vadovas

- Duomenys apie komandas yra surašyti tekstiniuose failuose „Player1.txt“ ir „Player2.txt“. Pirmoje eilutėje užrašytas komandos pavadinimas. Kitose eilutėse užrašyti duomenys apie krepšininkus: *Vardas ir Pavardė, Amžius, Ūgis.*
- Programa veikia mygtukų principu. Programa paleidžiama paspaudus mygtuką „Run“, tuomet atspausdinami pradiniai duomenys apie komandas.
- Paspaudus mygtuką „Count“ suskaičiuojama ir atspausdinama abiejų komandų ūgio bei amžiaus vidurkiai.
- Paspaudus mygtuką „Sort“ sudaromas bei atspausdinamas krepšininkų sąrašas iš abiejų komandų, kurių krepšininkų ūgis didesnis už komandos ūgio vidurkį. Tas sąrašas surikiuojamas pagal amžių didėjimo tvarka ir atspausdinamas dar kartą.
- Langelyje su tekstu „player‘s age“ įvedamas amžius pagal kurį bus koreguojamas surikiuotas sąrašas. Paspaudus mygtuką „Remove“ išmetami visi krepšininkai kurių amžius didesnis už įvestą ir sąrašas atspausdinamas.

1.5. Programos tekstas

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

namespace Laboras1_2sem_
{
    /// <summary>
    /// Class to describe the data of one player
    /// </summary>
    class Player
    {
        public string NamSur { get; set; } //Player's name and surname
        public int Height { get; set; }    //player's height
        public int Age { get; set; }       //player's age

        public Player(string NamS, int age, int Hght)
        {
            NamSur = NamS;
            Height = Hght;
            Age = age;
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override string ToString()
        {
            string line;
            line = string.Format("{0, -17} {1, 2} {2,6}", NamSur, Age, Height);
            return line;
        }
```

```
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override bool Equals(object objektas)
        {
            Player plr = objektas as Player;
            return plr.NamSur == NamSur;
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override int GetHashCode()
        {
            return base.GetHashCode();
        }

        /// <summary>
        /// Student comparison method
        /// </summary>
        /// <param name="p1">player one</param>
        /// /// <param name="p2">player two</param>
        /// <returns> true if player's one age is less than age of other player or if age is
        /// equal, then compares names and surnames of both players in alphabetical order
        /// else returns false
        /// </returns>
        public static bool operator <(Player p1, Player p2)
        {
            int p = String.Compare(p1.NamSur, p2.NamSur, StringComparison.CurrentCulture);

            if ((p1.Age < p2.Age) || ((p1.Age == p2.Age) && (p < 0)))
                return true;
            else return false;
        }

        /// <summary>
        /// Student comparison method
        /// </summary>
        /// <param name="p1">player one</param>
        /// /// <param name="p2">player two</param>
        /// <returns> true if player's one age is more than age of other player or if age is
        /// equal, then compares names and surnames of both players in alphabetical order
        /// else returns false
        /// </returns>
        public static bool operator >(Player p1, Player p2)
        {
            int p = String.Compare(p1.NamSur, p2.NamSur, StringComparison.CurrentCulture);

            if ((p1.Age > p2.Age) || ((p1.Age == p2.Age) && (p > 0)))
                return true;
            else return false;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laboras1_2sem_
{
    /// <summary>
    /// Dynamic container
    /// </summary>
    class TeamCont
    {
        private Player[] Team;
```

```
public int Count { get; private set; }
public int Capacity { get; private set; }

public TeamCont(int capacity = 16)
{
    Count = 0;
    Capacity = capacity;
    Team = new Player[capacity];
}

public Player GetPlayer(int i) { return Team[i]; }

/// <summary>
/// Increases capacity of container
/// /// <param name="NewCpty">new capacity</param>
/// </summary>
public void IncreaseCapacity(int NewCpty)
{
    if (NewCpty > Capacity)
    {
        Player[] TempT = new Player[NewCpty];
        for (int i = 0; i < Count; i++)
        {
            TempT[i] = Team[i];
        }
        Team = TempT;
        Capacity = NewCpty;
    }
}

/// <summary>
/// Add's player in a container
/// /// <param name="plr">the player that will be added</param>
/// </summary>
public void SetPlayer(Player plr)
{
    if(Count == Capacity)
    {
        IncreaseCapacity(Capacity * 2);
    }
    Team[Count++] = plr;
}

/// <summary>
/// Returns team's avarage age
/// </summary>
public double AvgAge()
{
    double a = 0;
    double avg;
    for (int i = 0; i < Count; i++)
    {
        a = a + Team[i].Age;
    }
    return avg = a / Count;
}
```

```
        /// <summary>
        /// Returns team's avarage height
        /// </summary>
        public double AvgHeight()
        {
            double a = 0;
            double avg;
            for (int i = 0; i < Count; i++)
            {
                a = a + Team[i].Height;
            }
            return avg = a / Count;
        }

        /// <summary>
        /// method to sort the conatainer by age
        /// </summary>
        public void Sort()
        {
            for (int i = 0; i < Count-1; i++)
            {
                Player pl = Team[i];
                int im = i;
                for (int j = i; j < Count; j++)
                {
                    if (Team[j] < pl)
                    {
                        im = j;
                        pl = Team[j];
                    }
                }
                    Team[im] = Team[i];
                    Team[i] = pl;
            }
        }

        /// <summary>
        /// Removes a player if his is less more than specified
        /// </summary>
        /// <param name="age">the specified age</param>
        public void Remove(int age)
        {
            int m = 0;
            for (int i = 0; i < Count; i++)
            {
                if(Team[i].Age <= age)
                    Team[m++] = Team[i];
            }Count = m;
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Laboras1_2sem_
{
    public partial class Form1 : Form
    {

        const string CFd = "..\\..\\Players1.txt"; // data file
```

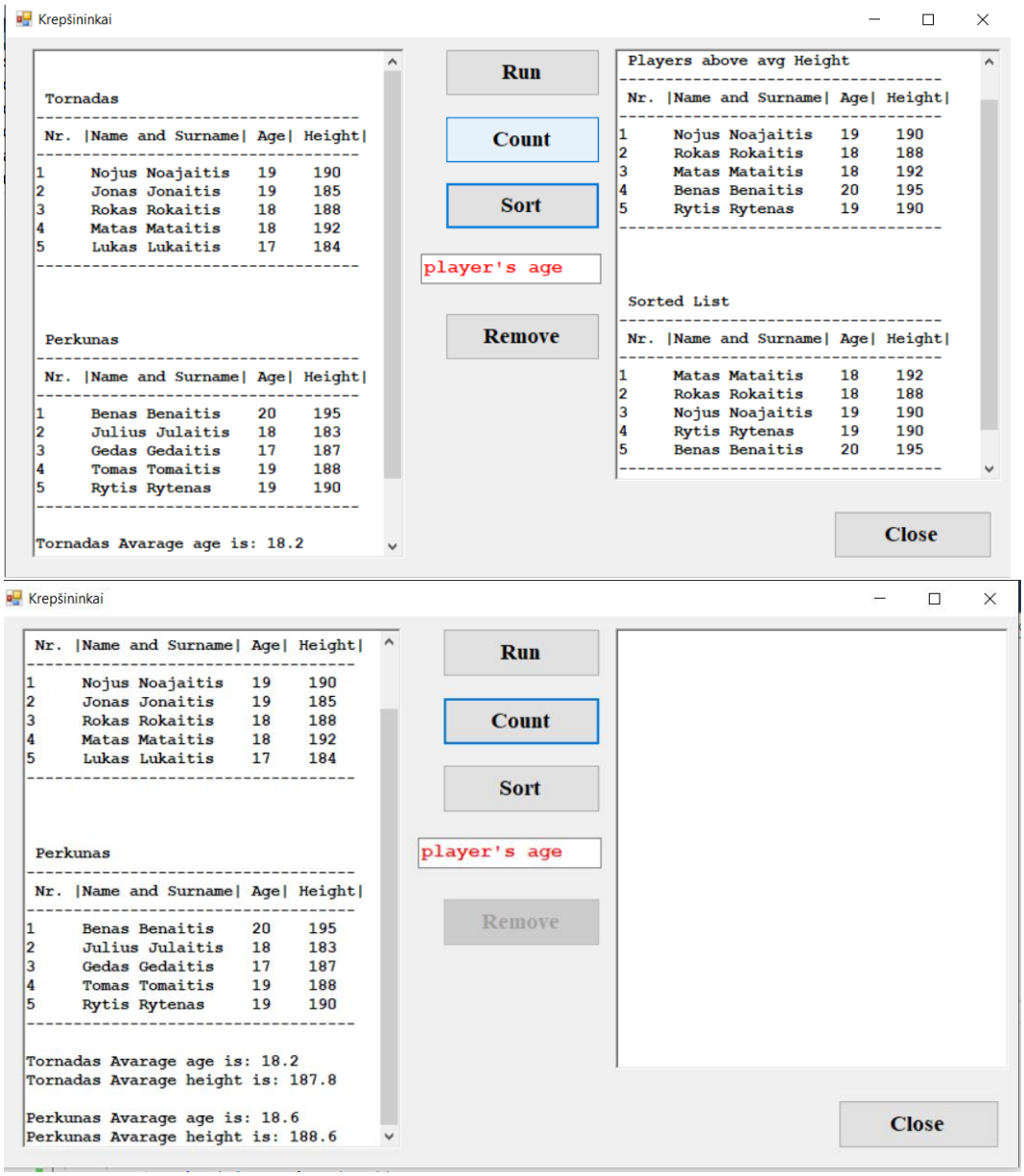
```
const string CFdm = "..\\..\\Players2.txt";// data file
const string CFr = "..\\..\\Results.txt";    // result's file

TeamCont Team;    //Team 1
TeamCont Team1;   //Team 2
TeamCont Team2;   //Team 3 (combined)
string TeamName;  //Team 1 name
string TeamName1; //Team 2 name



/// <summary>
/// Read data from file
/// </summary>
/// <param name="fn">file name</param>
/// <param name="TeamName">returns team's name</param>
static TeamCont ReadFile(string fn, out string TeamName)
{
    TeamCont Team = new TeamCont();
    using(StreamReader reader = new StreamReader(fn))
    {
        string line;
        line = reader.ReadLine();
        TeamName = line;
        while ((line = reader.ReadLine()) != null)
        {
            string[] parts = line.Split(';');
            string NamSur = parts[0];
            int Age = int.Parse(parts[1]);
            int Height = int.Parse(parts[2]);
            Player plr = new Player(NamSur, Age, Height);
            Team.SetPlayer(plr);
        }
    }
    return Team;
}
/// <summary>
/// Print's a table with a team's players
/// </summary>
/// <param name="fn">file name</param>
/// <param name="Team">Teams</param>
/// <param name="heading">heading</param>
static void Print(string fn, TeamCont Team, string heading)
{
    const string top =
            "-----------------------------------\r\n"
            + " Nr. |Name and Surname| Age| Height| \r\n"
            + "-----------------------------------";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n\n " + heading);
        fr.WriteLine(top);
        for (int i = 0; i < Team.Count; i++)
        {
            Player plr = Team.GetPlayer(i);
            fr.WriteLine("{0}     {1}",i+1, plr);
        }
        fr.WriteLine("-----------------------------------\n");
    }
}
```

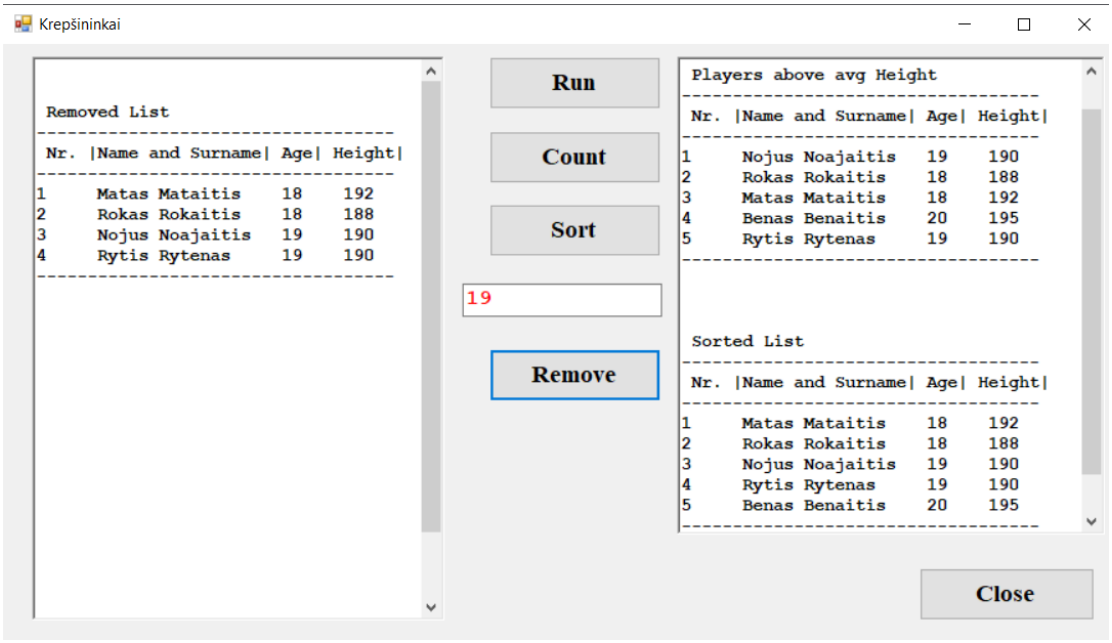
```
/// <summary>
/// Connects two Team's players into one array whose height is bigger than avarage
/// </summary>
/// <param name="T">Team 1</param>
/// /// <param name="T1">Team 2</param>
/// <returns>combined team of two team's</returns>
static TeamCont ConnectByHeight(TeamCont T, TeamCont T1)
{
    TeamCont T2 = new TeamCont();
    for (int i = 0; i < T.Count; i++)
    {
        if (T.GetPlayer(i).Height > T.AvgHeight())
        {
            Player plr = new Player(T.GetPlayer(i).NamSur,
                T.GetPlayer(i).Age, T.GetPlayer(i).Height);
            T2.SetPlayer(plr);
        }
    }
    for (int i = 0; i < T1.Count; i++)
    {
        if (T1.GetPlayer(i).Height > T1.AvgHeight())
        {
            Player plr1 = new Player(T1.GetPlayer(i).NamSur,
                T1.GetPlayer(i).Age, T1.GetPlayer(i).Height);
            T2.SetPlayer(plr1);
        }
    }return T2;
}

public Form1()
{
    InitializeComponent();

    if (File.Exists(CFr))
        File.Delete(CFr);
}

/// <summary>
/// Actions of the "close" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void close_Click(object sender, EventArgs e)
{
    Close();
}

/// <summary>
/// Actions of the "run" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void run_Click(object sender, EventArgs e)
{
    Team = ReadFile(CFd, out TeamName);
    Print(CFr, Team, TeamName);
    Team1 = ReadFile(CFdm, out TeamName1);
    Print(CFr, Team1, TeamName1);
    results.LoadFile(CFr, RichTextBoxStreamType.PlainText);
    count.Enabled = true;
    run.Enabled = false;
}
```

```
        /// <summary>
        /// Actions of the "count" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void count_Click(object sender, EventArgs e)
        {
            sort.Enabled = true;
            results.Clear();
            results.LoadFile(CFr, RichTextBoxStreamType.PlainText);
            using (var fr = File.AppendText(CFr))
            {
                fr.WriteLine(TeamName + " Avarage age is: " + Team.AvgAge());
                fr.WriteLine("\n" + TeamName + " Avarage height is: " + Team.AvgHeight());
                fr.WriteLine("\n\n" + TeamName1 + " Avarage age is: " + Team1.AvgAge());
                fr.WriteLine("\n" + TeamName1 + " Avarage height is: " + Team1.AvgHeight());
            }
            results.Text += TeamName + " Avarage age is: " + Team.AvgAge();
            results.Text += "\n" + TeamName + " Avarage height is: " + Team.AvgHeight();
            results.Text += "\n\n" + TeamName1 + " Avarage age is: " + Team1.AvgAge();
            results.Text += "\n" + TeamName1 + " Avarage height is: " + Team1.AvgHeight();
            count.Enabled = false;
        }

        /// <summary>
        /// Actions of the "sort" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void sort_Click(object sender, EventArgs e)
        {

            Team2 = ConnectByHeight(Team, Team1);
            Print(CFr, Team2, "Players above avg Height");
            Team2.Sort();
            Print(CFr, Team2, "Sorted List");
            remove.Enabled = true;
            results2.LoadFile(CFr, RichTextBoxStreamType.PlainText);
            sort.Enabled = false;

        }

        /// <summary>
        /// Actions of the "remove" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void remove_Click(object sender, EventArgs e)
        {
            //sort_Click(sender, e);
            int age = int.Parse(Įvesti.Text);
            if (age % 1 == 0)
            {
                Team2.Remove(age);
                Print(CFr, Team2, "Removed List");
                results.Clear();
                if (Team2.Count < 1)
                    results.Text += "There are no players with that age";
                else
                    results.LoadFile(CFr, RichTextBoxStreamType.PlainText);
            }
            else results.Text += "Please retype age";
            remove.Enabled = false;
        }
    }
}
```

1.6. Pradiniai duomenys ir rezultatai

Testas nr. 1



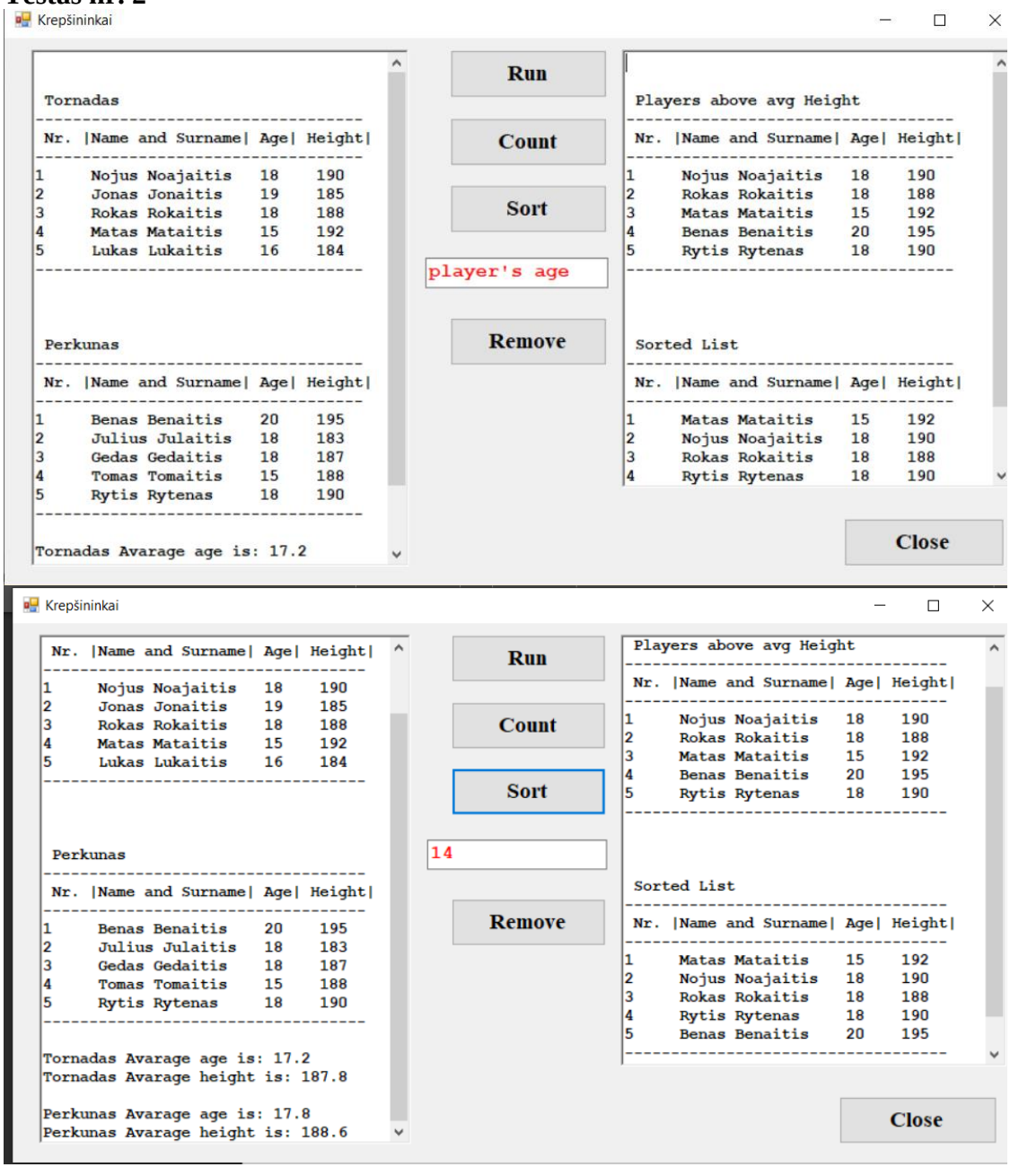


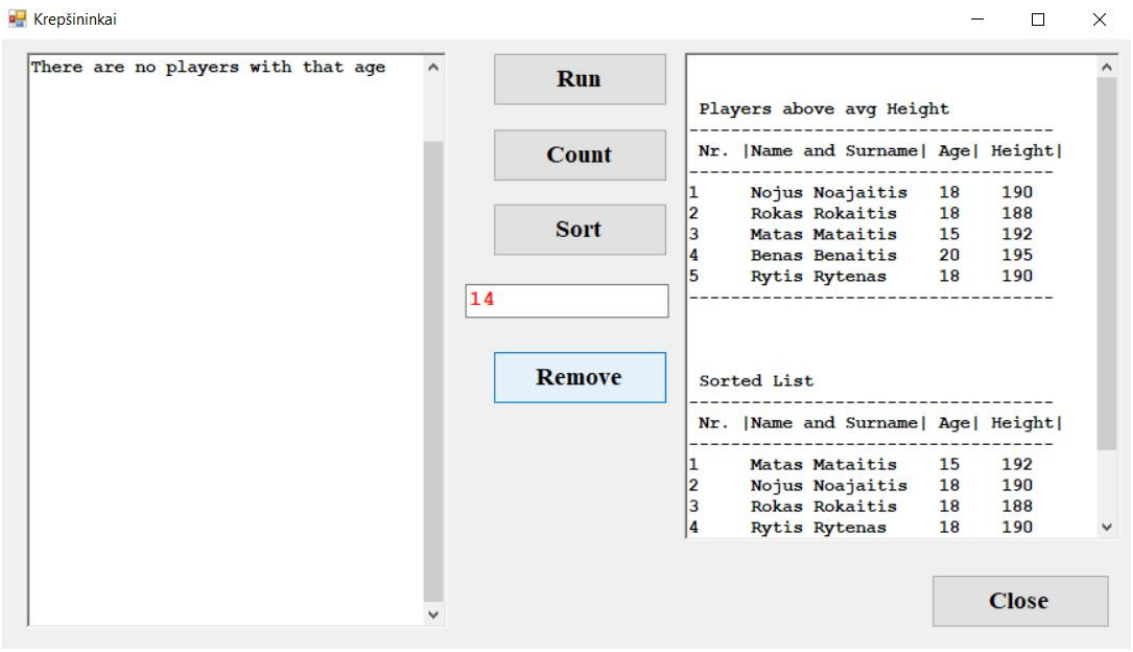
Rezultatų failas

```
Tornadas
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1      Nojus Noajaitis    19     190
2      Jonas Jonaitis     19     185
3      Rokas Rokaitis     18     188
4      Matas Mataitis     18     192
5      Lukas Lukaitis     17     184
-----------------------------------
 Perkunas
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1      Benas Benaitis     20     195
2      Julius Julaitis    18     183
3      Gedas Gedaitis     17     187
4      Tomas Tomaitis     19     188
5      Rytis Rytenas      19     190
-----------------------------------

Tornadas Avarage age is: 18.2
Tornadas Avarage height is: 187.8


Perkunas Avarage age is: 18.6
Perkunas Avarage height is: 188.6

 Players above avg Height
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1      Nojus Noajaitis    19     190
2      Rokas Rokaitis     18     188
3      Matas Mataitis     18     192
4      Benas Benaitis     20     195
5      Rytis Rytenas      19     190
-----------------------------------
```

```
Sorted List
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1      Matas Mataitis     18     192
2      Rokas Rokaitis     18     188
3      Nojus Noajaitis    19     190
4      Rytis Rytenas      19     190
5      Benas Benaitis     20     195
-----------------------------------
Removed List
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1      Matas Mataitis     18     192
2      Rokas Rokaitis     18     188
3      Nojus Noajaitis    19     190
4      Rytis Rytenas      19     190
-----------------------------------
```

Testas nr. 2





Rezultatų failas

```
Tornadas
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Nojus Noajaitis   18    190
2     Jonas Jonaitis    19    185
3     Rokas Rokaitis    18    188
4     Matas Mataitis    15    192
5     Lukas Lukaitis    16    184
-----------------------------------

 Perkunas
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Benas Benaitis    20    195
2     Julius Julaitis   18    183
3     Gedas Gedaitis    18    187
4     Tomas Tomaitis    15    188
5     Rytis Rytenas     18    190
-----------------------------------

Tornadas Avarage age is: 17.2
Tornadas Avarage height is: 187.8


Perkunas Avarage age is: 17.8
Perkunas Avarage height is: 188.6

 Players above avg Height
-----------------------------------
 Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Nojus Noajaitis   18    190
2     Rokas Rokaitis    18    188
3     Matas Mataitis    15    192
4     Benas Benaitis    20    195
5     Rytis Rytenas     18    190
-----------------------------------
```

```
 Sorted List
----------------------------------
 Nr. |Name and Surname| Age| Height|
----------------------------------
1     Matas Mataitis    15    192
2     Nojus Noajaitis   18    190
3     Rokas Rokaitis    18    188
4     Rytis Rytenas     18    190
5     Benas Benaitis    20    195
----------------------------------

 Removed List
----------------------------------
 Nr. |Name and Surname| Age| Height|
----------------------------------
----------------------------------

There are no players with that age
```

1.7. Dėstytojo pastabos

2. Dinaminis masyvas (L2)

2.1. Darbo užduotis

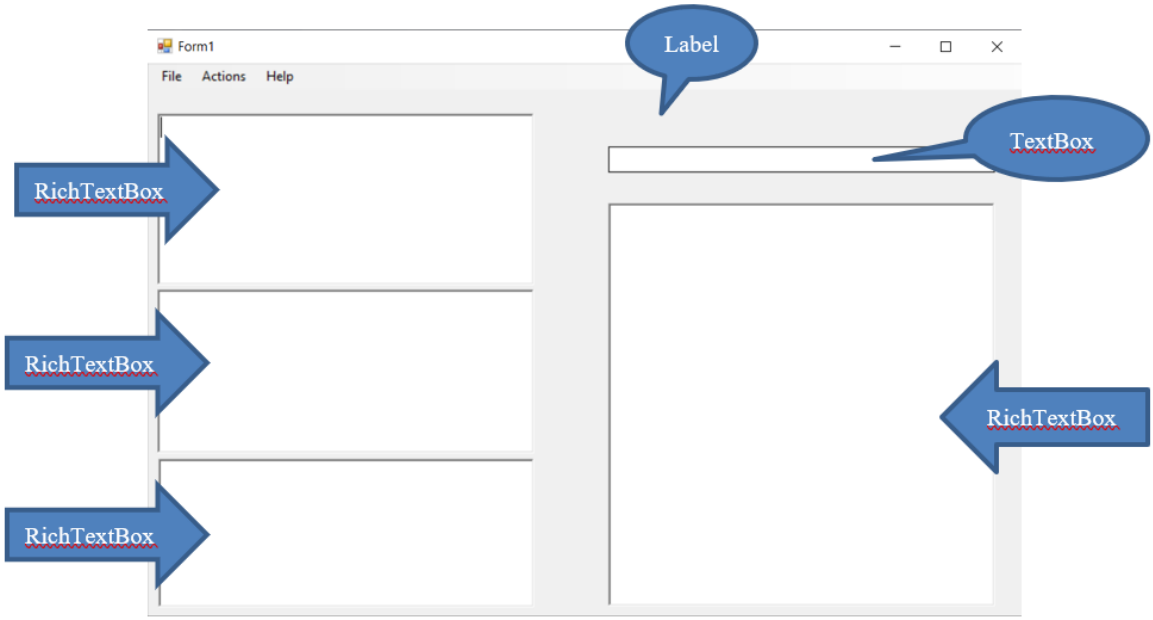
U1-2. Krepšinis.

Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Turime dviejų mokyklų duomenis.

L1+L2+L4.

- Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis kiekvienoje mokykloje.
- Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.
- Surikiuokite rezultatų sąrašą amžiaus didėjimo tvarka.
- Pašalinkite iš rezultatų sąrašo krepšininkus, kurių amžius yra didesnis už nurodytą klaviatūra. L2 papildymas.
- Papildykite surikiuotą rezultatų sąrašą naujais krepšininkais, kurių ūgis didesnis, už sudaryto sąrašo krepšininkų ūgio vidurkį. Duomenys yra atskirame faile. Pirmoje eilutėje – vadybininko vardas ir pavardė.

2.2. Grafinės vartotojo sąsajos schema ir paveikslas



2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
(Label)	Font	Times New Roman, Bold, 11, script Baltic
Įvesti (TextBox)	Font	Times New Roman, Bold, 11, script Baltic
Results1 (RichTextBox)	Font	Times New Roman, Bold, 12, script Baltic
Results2 (RichTextBox)	Font	Courier New, Bold, 11, script Baltic
Results3 (RichTextBox)	Font	Times New Roman, Bold, 12, script Baltic
Results4 (RichTextBox)	Font	Times New Roman, Bold, 12, script Baltic

2.4. Programos vartotojo vadovas

- Duomenys apie komandas yra surašyti tekstiniuose failuose „Player1.txt“ ir „Player2.txt“, „Player3.txt“. Pirmoje eilutėje užrašytas komandos pavadinimas. Kitose eilutėse užrašyti duomenys apie krepšininkus: *Vardas ir Pavardė, Amžius, Ūgis.*
- Programa veikia meniu principu. Programa paleidžiama paspaudus mygtuką „File -> Enter“ ir pasirinkus du failus, tuomet atspausdinami pradiniai duomenys apie pirmąsias dvi komandas.
- Paspaudus mygtuką „Actions -> Run“ suskaičiuojama ir atspausdinama abiejų komandų ūgio bei amžiaus vidurkiai. Sudaromas bei atspausdinamas krepšininkų sąrašas iš abiejų komandų, kurių krepšininkų ūgis didesnis už komandos ūgio vidurkį. Tas sąrašas surikiuojamas pagal amžių didėjimo tvarka ir atspausdinamas dar kartą.
- Langelyje žemiau teksto „Type player‘s age“ įvedamas amžius pagal kurį bus koreguojamas surikiuotas sąrašas. Paspaudus mygtuką „Actions -> Remove“ išmetami visi krepšininkai kurių amžius didesnis už įvestą ir sąrašas atspausdinamas.
- Paspaudus „Actions -> Add“ pasirnkame trečiąjį duomenų failą. Langelyje atspausdinami jo duomenys. Prie surikiuoto sąrašo pridedami krepšininkai, kurie yra aukštesni už surikiuoto sąrašo krepšininkų ūgio vidurkį.

2.5. Programos tekstas

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

namespace laboras2_2sem
{
    /// <summary>
    /// Class to describe the data of one player
    /// </summary>
    class Player : IComparable<Player>
    {
        public string NamSur { get; set; } //Player's name and surname
        public int Height { get; set; }    //player's height
        public int Age { get; set; }       //player's age

        public Player(string NamS, int age, int Hght)
        {
            NamSur = NamS;
            Height = Hght;
            Age = age;
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override string ToString()
        {
            string line;
            line = string.Format("{0, -17}; {1, 2}; {2,6};", NamSur, Age, Height);
            return line;
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override bool Equals(object objektas)
        {
            Player plr = objektas as Player;
            return plr.NamSur == NamSur;
        }
```

```
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        public override int GetHashCode()
        {
            return base.GetHashCode();
        }

        /// <summary>
        /// Student comparison method
        /// </summary>
        /// <param name="p1">player one</param>
        /// <returns> true if player's one age is less than age of other player or if age is
        /// equal, then compares names and surnames of both players in alphabetical order
        /// else returns false
        /// </returns>
        public int CompareTo(Player pl)
        {
            int poz = String.Compare(this.NamSur, pl.NamSur, StringComparison.CurrentCulture);
            if ((this.Age > pl.Age) || ((this.Age == pl.Age) && (poz > 0)))
                return 1;
            else return -1;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace laboras2_2sem
{
    public partial class Form1 : Form
    {

        const string CFr = "..\\..\\Results.txt";    // result's file
        const string Cffd = "..\\..\\Task.txt";
        const string Cfh = "..\\..\\user.txt";
        const string CFdd = "..\\..\\Rezultatai.csv";

        private List<Player> Team;    //Team 1
        private List<Player> Team1;   //Team 2
        private List<Player> Team2;   //Team 3 (combined Team1 with Team2)
        private List<Player> Team3;   //Team 4 (new Team)
        string TeamName;  //Team 1 name
        string TeamName1; //Team 2 name
        string TeamName2; //Team 4 (new team's agent name and surname)
```

```
/// <summary>
/// Read data from file
/// </summary>
/// <param name="fn">file name</param>
/// <param name="TeamName">returns team's name</param>
static List<Player> ReadFile(string fn, out string TeamName)
{
    List<Player> Team = new List<Player>();
    using (StreamReader reader = new StreamReader(fn))
    {
        string line;
        line = reader.ReadLine();
        TeamName = line;
        while ((line = reader.ReadLine()) != null)
        {
            string[] parts = line.Split(';');
            string NamSur = parts[0];
            int Age = int.Parse(parts[1]);
            int Height = int.Parse(parts[2]);
            Player plr = new Player(NamSur, Age, Height);
            Team.Add(plr);
        }
    }
    return Team;
}
/// <summary>
/// Print's a table with a team's players
/// </summary>
/// <param name="fn">file name</param>
/// <param name="Team">Teams</param>
/// <param name="heading">heading</param>
static void Print(string fn, List<Player> Team, string heading)
{
    const string top =
            "-----------------------------------\r\n"
            + "Nr. |Name and Surname| Age| Height| \r\n"
            + "-----------------------------------";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n" + heading);
        fr.WriteLine(top);
        for (int i = 0; i < Team.Count; i++)
        {
            Player plr = Team[i];
            fr.WriteLine("{0}     {1}", i + 1, plr);
        }
        fr.WriteLine("-----------------------------------\n");
    }
}
/// <summary>
/// Prints combined team to csv file
/// </summary>
/// <param name="fn">file name </param>
/// <param name="Team">Team</param>
/// <param name="heading">heading</param>
static void Print1(string fn, List<Player> Team, string heading)
{
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine(heading + ";");
        for (int i = 0; i < Team.Count; i++)
        {
            Player plr = Team[i];
            fr.WriteLine("{0};{1};{2};{3};", i + 1, plr.NamSur, plr.Age, plr.Height);
        }

    }
}
```

```
/// <summary>
/// Connects two Team's players into one array whose height is bigger than avarage
/// </summary>
/// <param name="T">Team 1</param>
/// <param name="T1">Team 2</param>
/// <returns>combined team of two team's</returns>
static void ConnectByHeight(List<Player> T, List<Player> T2)
{

    for (int i = 0; i < T.Count; i++)
    {
        if (T[i].Height > AvgHeight(T))
        {
            Player plr = new Player(T[i].NamSur,
                T[i].Age, T[i].Height);
            T2.Add(plr);
        }
    }
}

/// <summary>
/// Returns team's avarage age
/// </summary>
/// <param name="T">Team</param>
/// <returns>team's avarage age</returns>
static double AvgAge(List<Player> T)
{
    double a = 0;
    double avg = 0;
    for (int i = 0; i < T.Count; i++)
    {
        a = a + T[i].Age;
    }
    if (T.Count > 0)
        avg = a / T.Count;
    return avg;
}


/// <summary>
/// Returns team's avarage height
/// </summary>
/// <param name="T">Team</param>
/// <returns>team's avarage height</returns>
static double AvgHeight(List<Player> T)
{
    double a = 0;
    double avg = 0;
    for (int i = 0; i < T.Count; i++)
    {
        a = a + T[i].Height;
    }
    if(T.Count > 0)
        avg = a / T.Count;
    return avg;
}
/// <summary>
/// Finds index where to insert player
/// </summary>
/// <param name="T">Team</param>
/// <param name="name">player's name and surname</param>
/// <param name="age">player's age</param>
/// <returns></returns>
static int PlrIndex(List<Player>T, string name, int age)
{
    int ind = 0;
    for (int i = 0; i < T.Count; i++)
    {
```

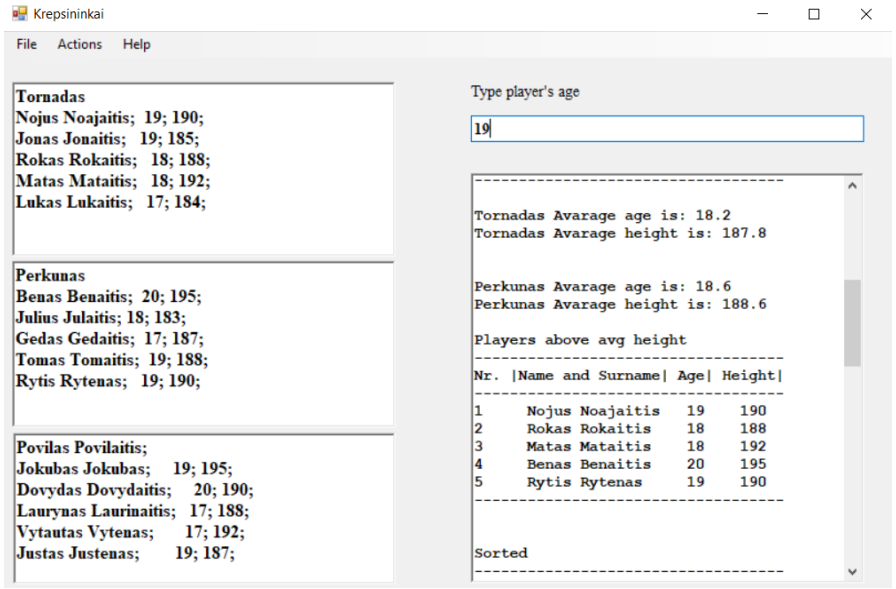
```
            int p = String.Compare(T[i].NamSur, name,StringComparison.CurrentCulture);
            if ((T[i].Age < age) || ((T[i].Age == age) && (p < 0)))
            {
                ind = i + 1;
            }
        }
        return ind;
    }
    /// <summary>
    /// Add's players whose height is bigger than avarage to the sorted list
    /// </summary>
    /// <param name="T">Team 1</param>
    /// /// <param name="T1">Team 2</param>
    /// <returns>combined team of two team's</returns>
    static void AddPlr(List<Player> T, List<Player> T1)
    {
        int ind;
        for (int i = 0; i < T.Count; i++)
        {
            if (T[i].Height > AvgHeight(T1))
            {
                ind = PlrIndex(T1, T[i].NamSur, T[i].Age);
                T1.Insert(ind, T[i]);
            }
        }
    }

    public Form1()
    {
        InitializeComponent();

        if (File.Exists(CFr))
            File.Delete(CFr);
        removeToolStripMenuItem.Enabled = false;
        runToolStripMenuItem.Enabled = false;
        addToolStripMenuItem.Enabled = false;
    }

    /// <summary>
    /// Actions of the "Enter" menu click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void enterToolStripMenuItem_Click(object sender, EventArgs e)
    {
        runToolStripMenuItem.Enabled = true;
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
        openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog1.Title = "Choose a file";
        DialogResult result = openFileDialog1.ShowDialog();
        if (result == DialogResult.OK)
        {
            string fv = openFileDialog1.FileName;
            Team = ReadFile(fv, out TeamName);
            results.LoadFile(fv, RichTextBoxStreamType.PlainText);

        }
        OpenFileDialog openFileDialog2 = new OpenFileDialog();
        openFileDialog2.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog2.Title = "Choose a file";
        DialogResult result1 = openFileDialog1.ShowDialog();
        if (result1 == DialogResult.OK)
        {
            string fv1 = openFileDialog1.FileName;
            Team1 = ReadFile(fv1, out TeamName1);
            results3.LoadFile(fv1, RichTextBoxStreamType.PlainText);

        }
```

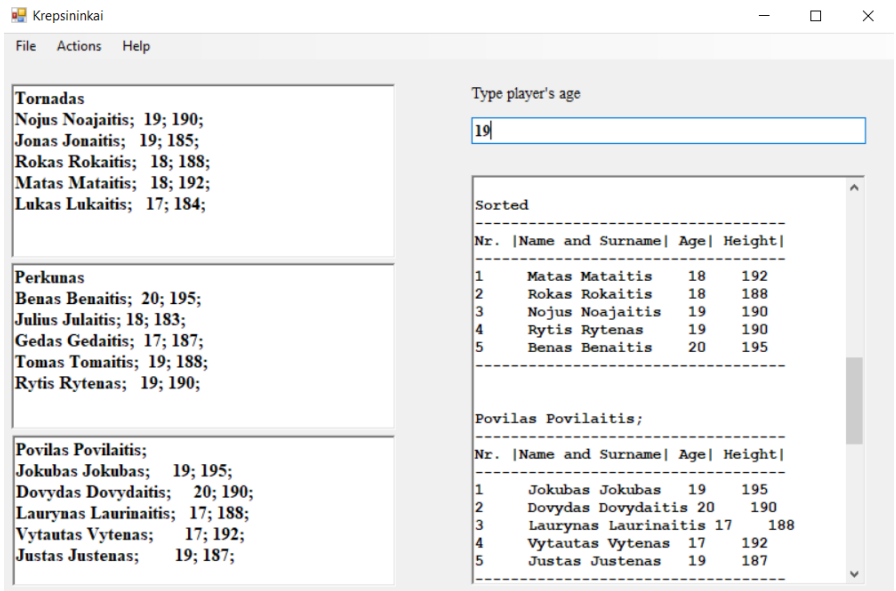
```
        Print(CFr, Team, TeamName);
        Print(CFr, Team1, TeamName1);

    }

    /// <summary>
    /// Actions of the "Close" menu click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void closeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close();
    }

    /// <summary>
    /// Actions of the "Run" menu click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void runToolStripMenuItem_Click(object sender, EventArgs e)
    {
        addToolStripMenuItem.Enabled = true;
        removeToolStripMenuItem.Enabled = false;
        runToolStripMenuItem.Enabled = false;
        using (var fr = File.AppendText(CFr))
        {
            fr.WriteLine(TeamName + " Avarage age is: " + AvgAge(Team));
            fr.WriteLine(TeamName + " Avarage height is: " + AvgHeight(Team));
            fr.WriteLine("\n\n" + TeamName1 + " Avarage age is: " + AvgAge(Team1));
            fr.WriteLine(TeamName1 + " Avarage height is: " + AvgHeight(Team1));
        }
        Team2 = new List<Player>();
        ConnectByHeight(Team, Team2);
        ConnectByHeight(Team1, Team2);
        Print(CFr, Team2, "Players above avg height");
        Team2.Sort();
        Print(CFr, Team2, "Sorted");
        results2.LoadFile(CFr, RichTextBoxStreamType.PlainText);
    }

    /// <summary>
    /// Actions of the "Remove" menu click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void removeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        addToolStripMenuItem.Enabled = false;
        runToolStripMenuItem.Enabled = false;
        removeToolStripMenuItem.Enabled = false;
        int age = int.Parse(Įvesti.Text);
        Team2.RemoveAll(item => item.Age > age);
        Print(CFr, Team2, "Removed list");
        if (Team2.Count < 1)
        {
            results2.Text += "\n\nThere are no players with that age";
            using (var fr = File.AppendText(CFr))
                fr.WriteLine("There are no players with that age");
        }
        else
            results2.LoadFile(CFr, RichTextBoxStreamType.PlainText);
    }


    /// <summary>
    /// Actions of the "Add" menu click
    /// </summary>
```

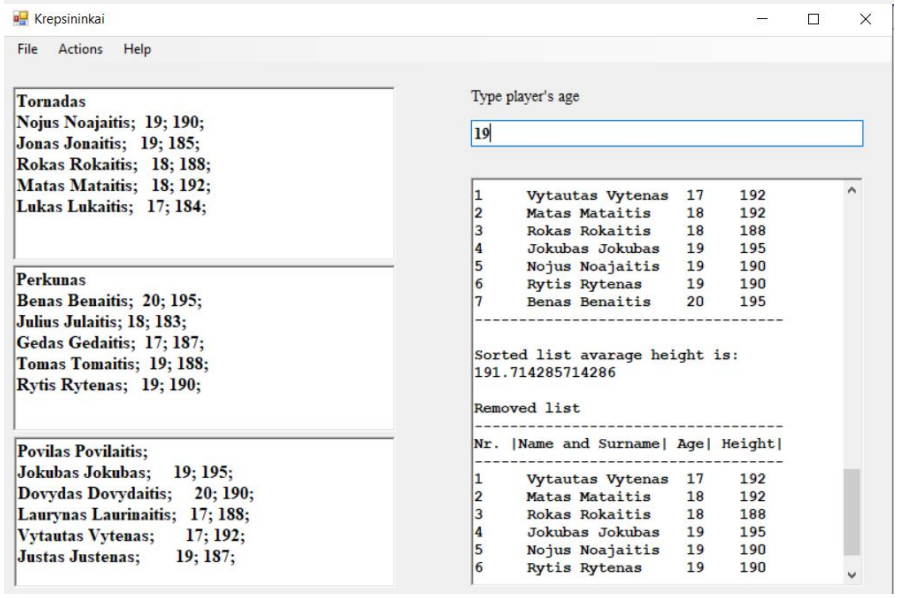
```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void addToolStripMenuItem_Click(object sender, EventArgs e)
        {
                OpenFileDialog openFileDialog2 = new OpenFileDialog();
                openFileDialog2.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
                openFileDialog2.Title = "Choose a file";
                DialogResult result1 = openFileDialog1.ShowDialog();
                if (result1 == DialogResult.OK)
                {
                    string fv1 = openFileDialog1.FileName;
                    Team3 = ReadFile(fv1, out TeamName2);
                    results4.LoadFile(fv1, RichTextBoxStreamType.PlainText);
                }
                removeToolStripMenuItem.Enabled = true;
                runToolStripMenuItem.Enabled = false;
                addToolStripMenuItem.Enabled = false;
                Print(CFr, Team3, TeamName2);
                if (File.Exists(CFdd))
                    File.Delete(CFdd);
                double vid = AvgHeight(Team2);
                AddPlr(Team3, Team2);
                Print(CFr, Team2, "Combined");
                Print1(CFdd, Team2, "Combined");
                results2.LoadFile(CFr, RichTextBoxStreamType.PlainText);
                results2.Text += "Sorted list avarage height is: " + vid;
                using (var fr = File.AppendText(CFr))
                    fr.WriteLine("Sorted list avarage height is: " + vid);
                label1.Text = "Type player's age";

        }
        /// <summary>
        /// Actions of the "Task" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void taskToolStripMenuItem_Click(object sender, EventArgs e)
        {
            results.Clear();
            results2.Clear();
            results3.Clear();
            results4.Clear();
            results2.LoadFile(Cffd, RichTextBoxStreamType.PlainText);
            removeToolStripMenuItem.Enabled = false;
            runToolStripMenuItem.Enabled = false;
            addToolStripMenuItem.Enabled = false;
            enterToolStripMenuItem.Enabled = true;
        }
        /// <summary>
        /// Actions of the "User Instructions" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userInstructionsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            results.Clear();
            results2.Clear();
            results3.Clear();
            results4.Clear();
            results2.LoadFile(Cfh, RichTextBoxStreamType.PlainText);
            removeToolStripMenuItem.Enabled = false;
            runToolStripMenuItem.Enabled = false;
            addToolStripMenuItem.Enabled = false;
            enterToolStripMenuItem.Enabled = true;
        }
    }
}
```

2.6. Pradiniai duomenys ir rezultatai

Testas nr. 1

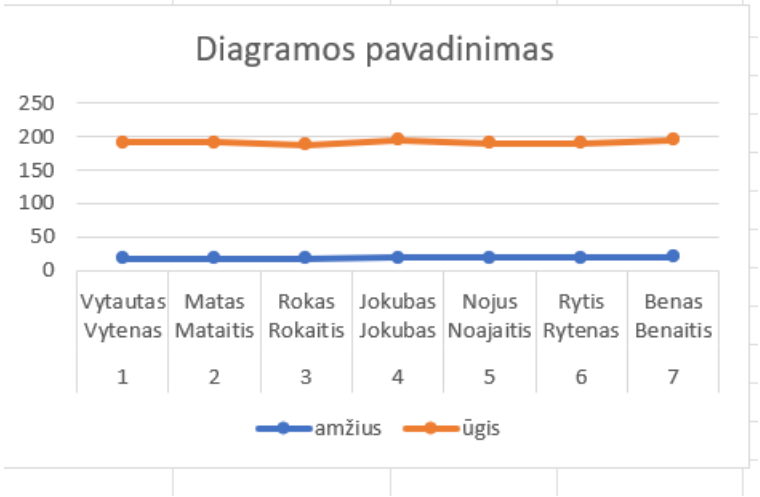




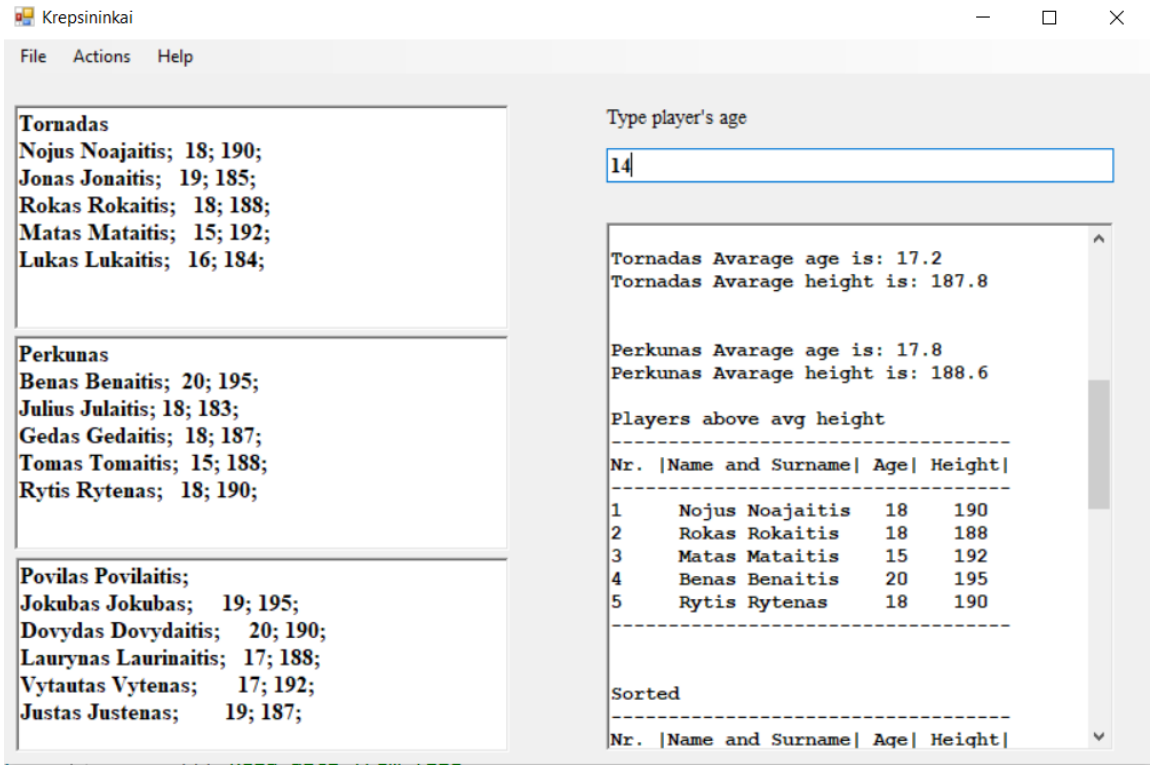


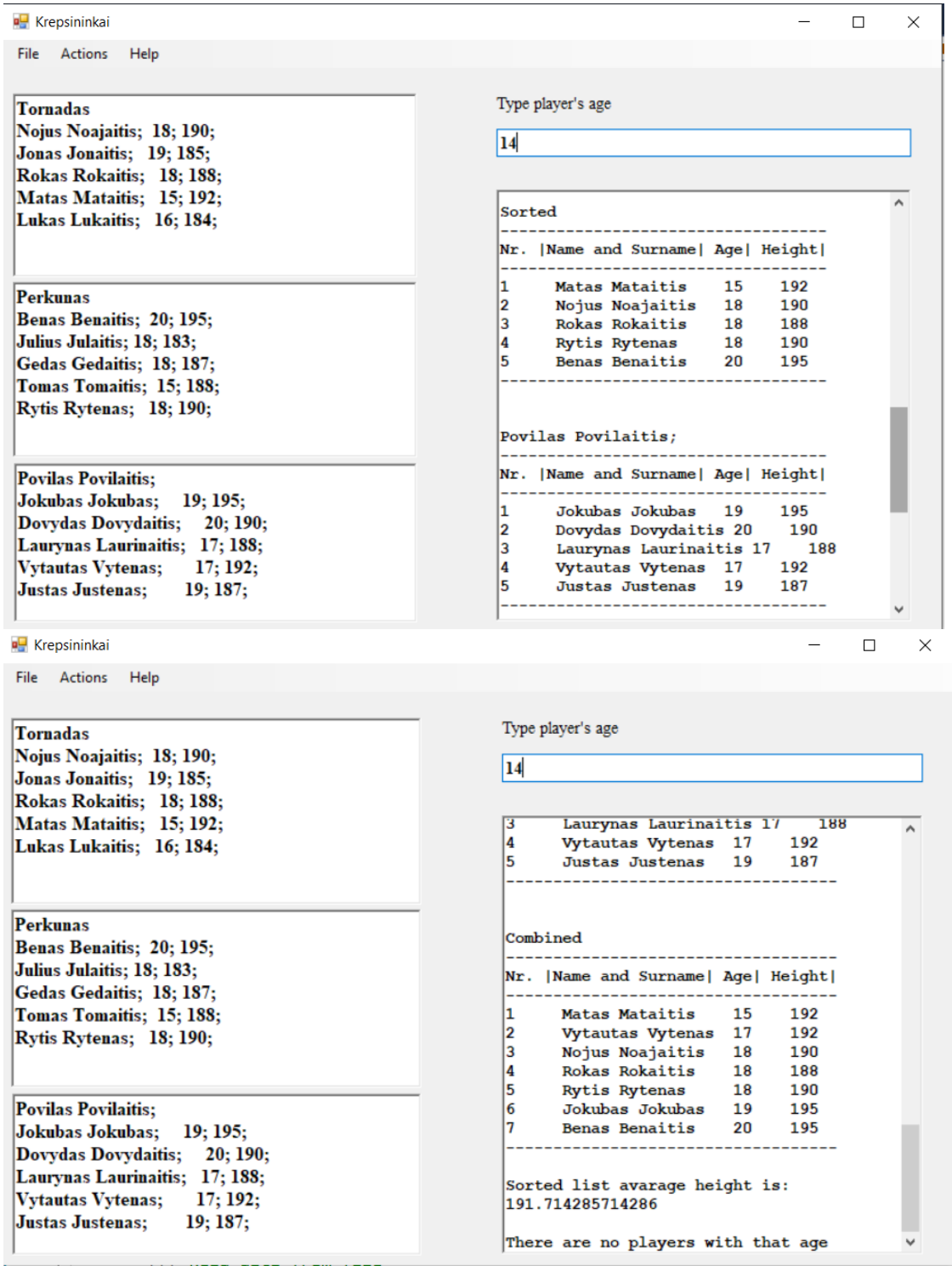
Rezultatu.xlsx failas

Combined	Column1	amžius	ūgis
1	Vytautas Vytenas	17	192
2	Matas Mataitis	18	192
3	Rokas Rokaitis	18	188
4	Jokubas Jokubas	19	195
5	Nojus Noajaitis	19	190
6	Rytis Rytenas	19	190
7	Benas Benaitis	20	195



Testas nr. 2

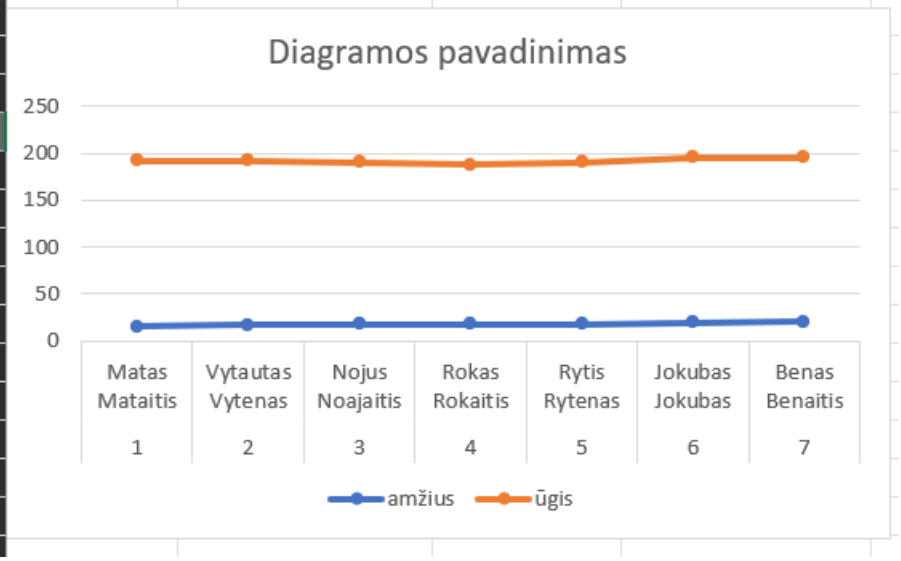



Krepsininkai
File
Actions
Help
Tornadas
Nojus Noajaitis; 18; 190;
Jonas Jonaitis; 19; 185;
Rokas Rokaitis; 18; 188;
Matas Mataitis; 15; 192;
Lukas Lukaitis; 16; 184;
Perkunas
Benas Benaitis; 20; 195;
Julius Julaitis; 18; 183;
Gedas Gedaitis; 18; 187;
Tomas Tomaitis; 15; 188;
Rytis Rytenas; 18; 190;
Povilas Povilaitis;
Jokubas Jokubas; 19; 195;
Dovydas Dovydaitis; 20; 190;
Laurynas Laurinaitis; 17; 188;
Vytautas Vytenas; 17; 192;
Justas Justenas; 19; 187;
Type player's age
14
Sorted
Nr. |Name and Surname| Age| Height|
1 Matas Mataitis 15 192
2 Nojus Noajaitis 18 190
3 Rokas Rokaitis 18 188
4 Rytis Rytenas 18 190
5 Benas Benaitis 20 195
Povilas Povilaitis;
Nr. |Name and Surname| Age| Height|
1 Jokubas Jokubas 19 195
2 Dovydas Dovydaitis 20 190
3 Laurynas Laurinaitis 17 188
4 Vytautas Vytenas 17 192
5 Justas Justenas 19 187

Krepsininkai
File
Actions
Help
Tornadas
Nojus Noajaitis; 18; 190;
Jonas Jonaitis; 19; 185;
Rokas Rokaitis; 18; 188;
Matas Mataitis; 15; 192;
Lukas Lukaitis; 16; 184;
Perkunas
Benas Benaitis; 20; 195;
Julius Julaitis; 18; 183;
Gedas Gedaitis; 18; 187;
Tomas Tomaitis; 15; 188;
Rytis Rytenas; 18; 190;
Povilas Povilaitis;
Jokubas Jokubas; 19; 195;
Dovydas Dovydaitis; 20; 190;
Laurynas Laurinaitis; 17; 188;
Vytautas Vytenas; 17; 192;
Justas Justenas; 19; 187;
Type player's age
14
3 Laurynas Laurinaitis 17 188
4 Vytautas Vytenas 17 192
5 Justas Justenas 19 187
Combined
Nr. |Name and Surname| Age| Height|
1 Matas Mataitis 15 192
2 Vytautas Vytenas 17 192
3 Nojus Noajaitis 18 190
4 Rokas Rokaitis 18 188
5 Rytis Rytenas 18 190
6 Jokubas Jokubas 19 195
7 Benas Benaitis 20 195
Sorted list avarage height is:
191.714285714286
There are no players with that age

Rezultatu.xlsx failas

	A	B	C	D
1	Combined	Column1	amžius	ūgis
2	1	Matas Mataitis	15	192
3	2	Vytautas Vytenas	17	192
4	3	Nojus Noajaitis	18	190
5	4	Rokas Rokaitis	18	188
6	5	Rytis Rytenas	18	190
7	6	Jokubas Jokubas	19	195
8	7	Benas Benaitis	20	195



2.7. Dėstytojo pastabos

3. Paveldėjimas (L3)

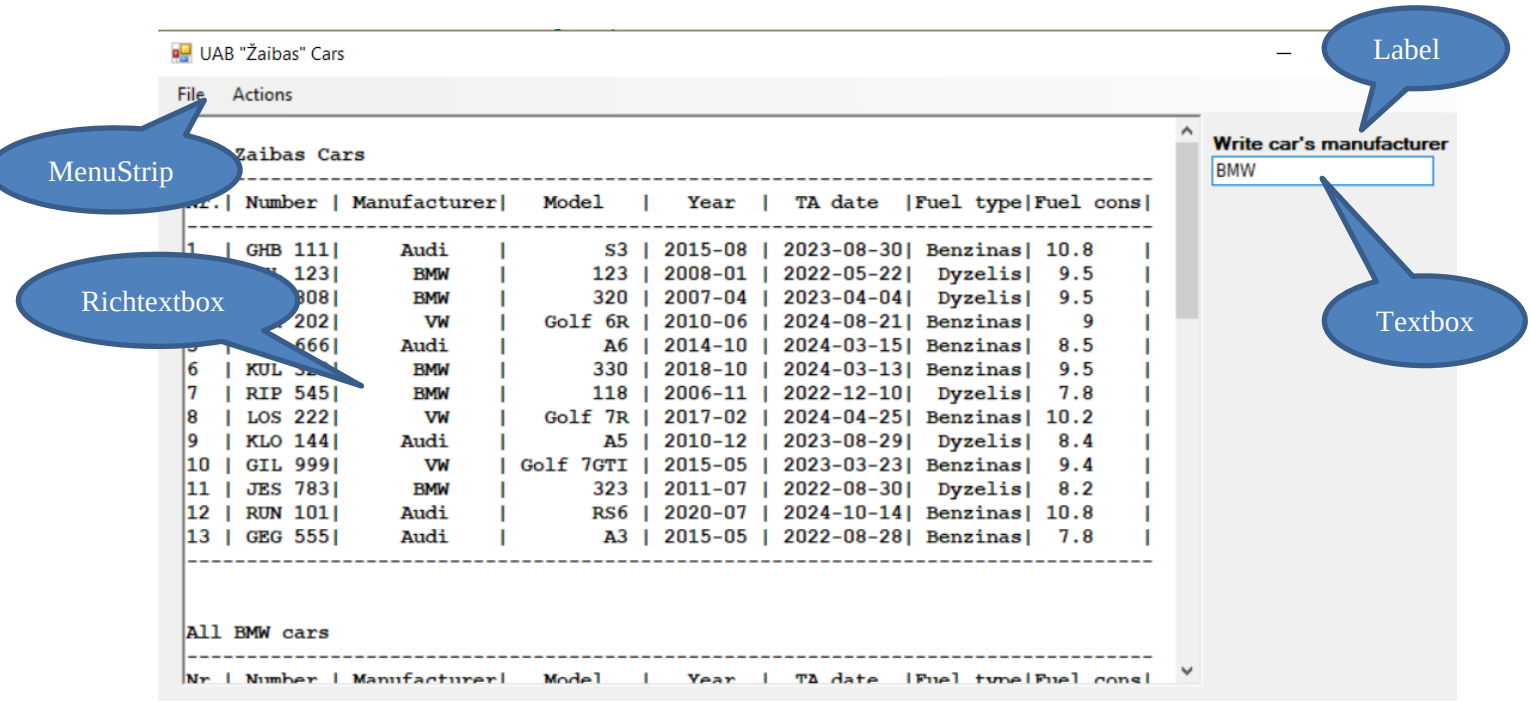
3.1. Darbo užduotis

2. Automobilių parkas.

Turite duomenis apie UAB „Žaibas“ priklausančius automobilius. Duomenų faile pateikta ši informacija: valstybinis numeris, gamintojas, modelis, pagaminimo metai ir mėnuo, techninės apžiūros galiojimo data, kuras, vidutinės kuro sąnaudos (100km).

- Raskite du naujausius automobilius (visi duomenys).
- Sudarykite visų nurodytos markės automobilių sąrašą (visi duomenys).
- Surikiuokite sudarytą sąrašą pagal pasirinktus du požymius.
- Sudarykite visų automobilių gamintojų sąrašą be pasikartojimų. Visus skaičiavimų rezultatus pateikite rezultatų faile lentelėmis.

3.2. Grafinės vartotojo sąsajos schema ir paveikslas



3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Results (RichTextBox)	Font	Courier New, Bold, 10, script Baltic
Tekstas (Label)	Font	Times New Roman, 9, script Baltic
įvesti (TextBox)	Font	Times New Roman, 9, script Baltic

3.4. Programos vartotojo vadovas

1. Paspaudus 'Enter' pasirinkite duomenų failą.
2. Langelyje užrašykite norimą auto gamintoją.
3. Paspauskite 'Save' ir pasirinkite kur norėsite saugoti duomenis.

3.5. Programos tekstas

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

namespace LAB3_sem2_
{
    /// <summary>
    /// Base class of car
    /// </summary>
    abstract class Car : Object
    {
        public string Number { get; set; }
        public string Manufacturer { get; set; }
        public string Make { get; set; }
        public DateTime Year { get; set; }
        public DateTime TAdate { get; set; }

        /// <summary>
        /// Empty constructor
        /// </summary>
        public Car()
        {

        }

        /// <summary>
        /// Construcor
        /// </summary>
        /// <param name="Nmbr">Number</param>
        /// <param name="Manuf">Manufacturer</param>
        /// <param name="Make">Make</param>
        /// <param name="Year">Year</param>
        /// <param name="TAd">TA date</param>
        public Car(string Nmbr, string Manuf, string Make, DateTime Year, DateTime TAd)
        {
            Number = Nmbr;
            Manufacturer = Manuf;
            this.Make = Make;
            this.Year = Year;
            TAdate = TAd;
        }
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            string line;
            line = string.Format(" {0,3}| {1,8}    | {2,9} | {3,2} |",
            Number, Manufacturer, Make, Year.ToString("yyyy-MM"));
            return line;
        }
    }
}
```

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

namespace LAB3_sem2_
{
    /// <summary>
    /// derivative class of car
    /// </summary>
    class Cars : Car, IComparable<Cars>, IEquatable<Cars>
    {

        public string FuelType { get; set; }
        public double FuelConsumption { get; set; }

        /// <summary>
        /// Empty constructor
        /// </summary>
        public Cars()
        {

        }
        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="Nmbr">Number</param>
        /// <param name="Manuf">Manufacturer</param>
        /// <param name="Make">Make</param>
        /// <param name="Year">Year</param>
        /// <param name="TAd">TA date</param>
        /// <param name="FuelT">Fuel type</param>
        /// <param name="FuelCon">Fuel consumption</param>
        public Cars(string Nmbr, string Manuf, string Make, DateTime Year, DateTime TAd,
string FuelT, double FuelCon):
            base(Nmbr, Manuf, Make, Year, TAd)
        {
            TAdate = TAd;
            FuelType = FuelT;
            FuelConsumption = FuelCon;
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            string line;
            line = string.Format("{0} {1,8}| {2,8}| {3,4}    |", base.ToString(),
                TAdate.ToString("yyyy-MM-dd"), FuelType, FuelConsumption);
            return line;
        }
        /// <summary>
        /// Cars comparison method
        /// </summary>
        /// <param name="car"></param>
        /// <returns>true if car's Fuel consuption is bigger than other car's or
        /// if Fuel consumption is equal then compares Cars years in a descending
order</returns>
        public int CompareTo(Cars car)
        {
            if ((this.FuelConsumption < car.FuelConsumption) ||
                (this.FuelConsumption == car.FuelConsumption) && (this.Year < car.Year))
                return 1;
            else return -1;
        }
```

```
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            return Equals(obj as Cars);
        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <param name="kitas"></param>
        /// <returns></returns>
        public bool Equals(Cars kitas)
        {
            return kitas.Number == Number &&
                kitas.Manufacturer == Manufacturer &&
                kitas.Make == Make &&
                kitas.Year == Year &&
                kitas.TAdate == TAdate &&
                kitas.FuelType == FuelType &&
                kitas.FuelConsumption == FuelConsumption;

        }

        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns></returns>
        public override int GetHashCode()
        {
            return Number.GetHashCode() ^
                Manufacturer.GetHashCode() ^
                Make.GetHashCode() ^
                Year.GetHashCode() ^
                TAdate.GetHashCode() ^
                FuelType.GetHashCode() ^
                FuelConsumption.GetHashCode();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LAB3_sem2_
{

    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        private List<Cars> Cars;  //first cars list
        private List<Cars> Cars1; //cars list of given manufacturer
```

```
private List<Cars> Cars2; //cars list without duplicates

/// <summary>
/// Reads data from file
/// </summary>
/// <param name="fn">file name</param>
static List<Cars> ReadFile(string fn)
{
    List<Cars> Cars = new List<Cars>();
    using (StreamReader reader = new StreamReader(fn))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            string[] parts = line.Split(';');
            string Nmbr = parts[0];
            string Manuf = (parts[1]);
            string Make = (parts[2]);
            DateTime Year = DateTime.Parse(parts[3]);
            DateTime TAd = DateTime.Parse(parts[4]);
            string FuelT = (parts[5]);
            double FuelCon = double.Parse(parts[6]);
            Cars plr = new Cars(Nmbr, Manuf, Make, Year, TAd, FuelT, FuelCon);
            Cars.Add(plr);
        }
    }
    return Cars;
}
/// <summary>
/// Print's a table with a team's players
/// </summary>
/// <param name="fn">file name</param>
/// <param name="C">Cars list</param>
/// <param name="heading">heading</param>
static void Print(string fn, List<Cars> C, string heading)
{
    const string top = "-----------------------------------------------------------------
        ------------------\r\n"
            + "Nr.| Number | Manufacturer|   Model   |   Year  |  TA date  |Fuel
 type|Fuel cons|\r\n"
            + "-------------------------------------------------------------------------
----------";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n" + heading);
        fr.WriteLine(top);
        for (int i = 0; i < C.Count; i++)
        {
            Cars cr = C[i];
            fr.WriteLine("{0,-3}|{1}", i + 1, cr.ToString());
        }
        fr.WriteLine("-----------------------------------------------------------------
----------------\n");
    }
}
```

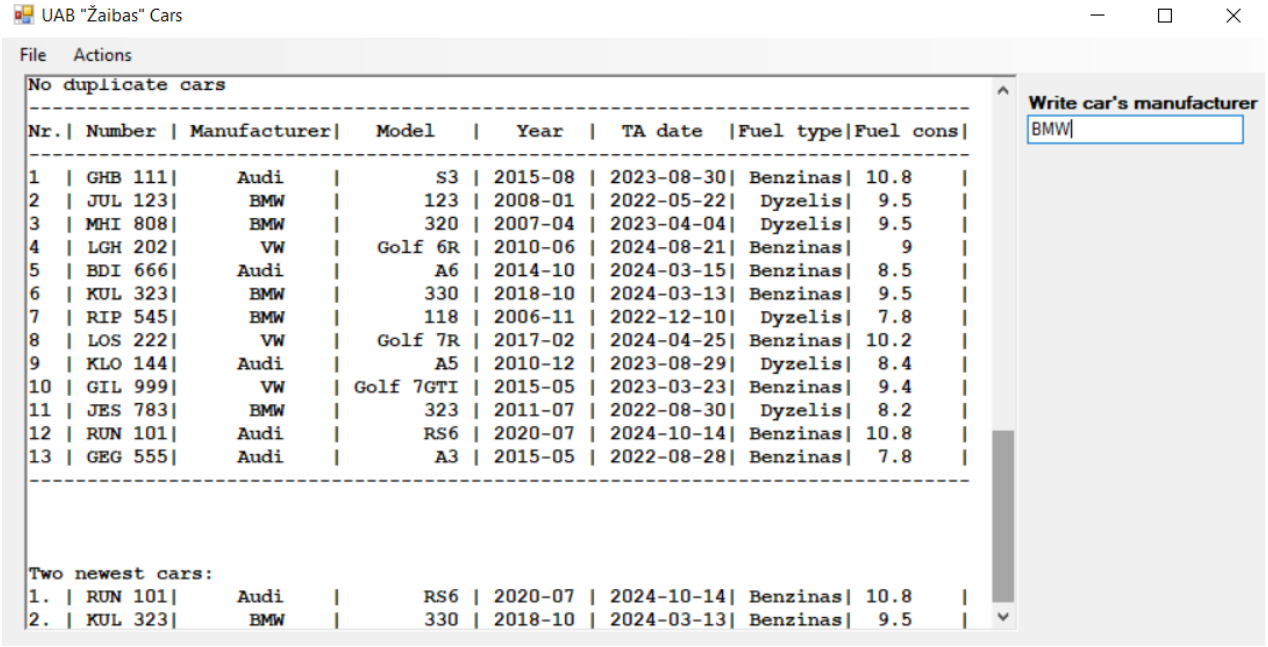
```
/// <summary>
/// returns first and second newest cars from cars list
/// </summary>
/// <param name="Cars">Cars list</param>
/// <param name="max1">first newest car</param>
/// <param name="max2">second newest car</param>
static void TwoNewestCars(List<Cars> Cars, out Cars max1, out Cars max2)
{
    //max1 = new Cars();
    //max2 = new Cars();
    //ind1 = ind2 = 0;
    if(Cars[0].Year > Cars[1].Year)
    {
        max1 = Cars[0]; max2 = Cars[1];
    }
    else
    {
        max1 = Cars[1]; max2 = Cars[0];
    }
    for (int i = 0; i < Cars.Count; i++)
    {
        if(Cars[i].Year > max1.Year)
        {
            max2 = max1;
            max1 = Cars[i];
            //ind1 = i+1;
        }
        else if(Cars[i].Year > max2.Year)
        {
            max2 = Cars[i];
            //ind2 = i+1;
        }
    }
}


/// <summary>
/// Constructs a new list with cars of the given manufacturer
/// </summary>
/// <param name="C1">Cars list</param>
/// <param name="C2">Cars1 list</param>
/// <param name="Mk">Manufacturer's name</param>
static void Construct(List<Cars> C1, List<Cars> C2, string Mk)
{

    for (int i = 0; i < C1.Count; i++)
    {
        if(C1[i].Manufacturer == Mk)
        {
            Cars cr = new Cars(C1[i].Number, C1[i].Manufacturer, C1[i].Make,
                C1[i].Year, C1[i].TAdate, C1[i].FuelType, C1[i].FuelConsumption);
            C2.Add(cr);
        }
    }
}
```

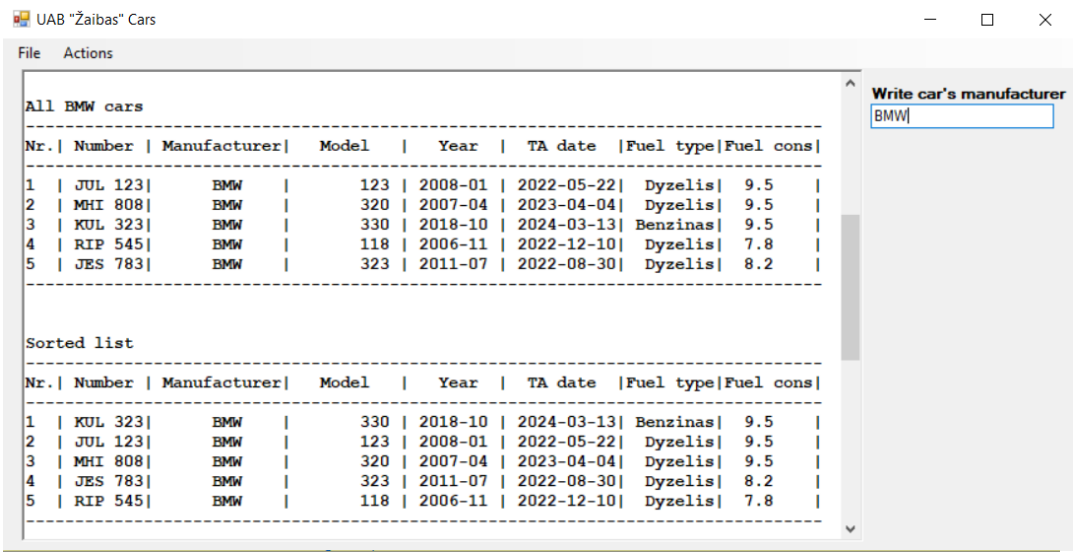
```
/// <summary>
/// Constructs a list of cars without duplicates
/// </summary>
/// <param name="C1">Cars list</param>
/// <param name="C2">Cars2 list</param>
static void NoRepeatings(List<Cars> C1, List<Cars> C2)
{
    for (int i = 0; i < C1.Count; i++)
    {
        int p = 0;
        Cars cr1 = new Cars(C1[i].Number, C1[i].Manufacturer, C1[i].Make,
            C1[i].Year, C1[i].TAdate, C1[i].FuelType, C1[i].FuelConsumption);
        for (int j = 0; j < C2.Count; j++)
        {
            if (C1[i].Equals(C2[j]))
            {
                p = 1;
            }
        }
        if (p == 0)
            {
                C2.Add(cr1);
            }
    }
}

/// <summary>
/// Actions of the "Enter" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void enterToolStripMenuItem_Click(object sender, EventArgs e)
{
    results.Clear();
    runToolStripMenuItem.Enabled = true;
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    openFileDialog1.Title = "Choose a file";
    DialogResult result = openFileDialog1.ShowDialog();
    if (result == DialogResult.OK)
    {
        string fv = openFileDialog1.FileName;
        Cars = ReadFile(fv);
        results.LoadFile(fv, RichTextBoxStreamType.PlainText);
    }
    tekstas.Text = "Write car's manufacturer";
}

/// <summary>
/// Actions of the "Run" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void runToolStripMenuItem_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    saveFileDialog1.Title = "Pasirinkite rezultatų failą";
    DialogResult result = saveFileDialog1.ShowDialog();
    if (result == DialogResult.OK)
    {
        string fv = saveFileDialog1.FileName;
        if (File.Exists(fv))
            File.Delete(fv);
        Print(fv, Cars, "UAB Zaibas Cars");
        string mak = įvesti.Text;
        Cars1 = new List<Cars>();
        Construct(Cars, Cars1, mak);
```

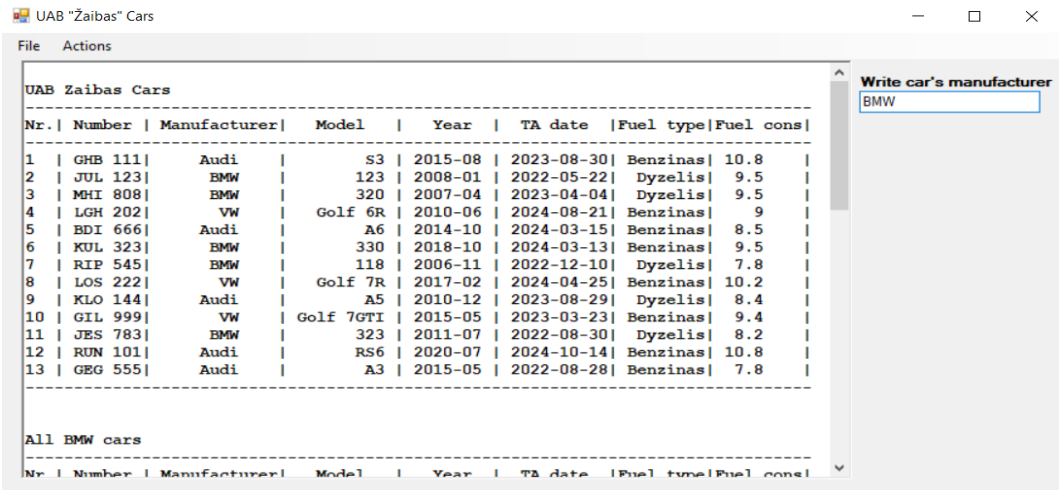
```
                Cars max1 = new Cars();
                Cars max2 = new Cars();
                //int ind1, ind2;
                TwoNewestCars(Cars, out max1, out max2);
                using (var fr = File.AppendText(fv))
                {
                    fr.WriteLine("\n\nTwo newest cars:\n1. |" + max1 + "\n2. |" + max2);
                }
                Print(fv, Cars1, "All " + mak + " cars");
                Cars1.Sort();
                Print(fv, Cars1, "Sorted list");
                Cars2 = new List<Cars>();
                NoRepeatings(Cars, Cars2);
                Print(fv, Cars2, "No duplicate cars");
                results.LoadFile(fv, RichTextBoxStreamType.PlainText);
            }
        }
        /// <summary>
        /// Actions of the "Close" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void helpToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            runToolStripMenuItem.Enabled = false;
            results.Clear();
            results.Text = "1.Paspaudus 'Enter' pasirinkite duomenų failą" +
                "\n2.Langelyje užrašykite norimą auto gamintoją" +
                "\n3.Paspauskite 'Save' ir pasirinkite kur norėsite saugoti duomenis" +
                "\n\n\n\n\nProgramą sukūrė: Martynas Burneika";
        }
    }
}
```

3.6. *Pradiniai duomenys ir rezultatai*

Testas nr. 1







Rezultatų failas:

```
UAB Zaibas Cars
-------------------------------------------------------------------------------
Nr.| Number | Manufacturer|   Model   |   Year  |  TA date  |Fuel type|Fuel cons|
-------------------------------------------------------------------------------
1  | GHB 111|     Audi    |        S3 | 2015-08 | 2023-08-30| Benzinas| 10.8    |
2  | JUL 123|      BMW    |       123 | 2008-01 | 2022-05-22|  Dyzelis|  9.5    |
3  | MHI 808|      BMW    |       320 | 2007-04 | 2023-04-04|  Dyzelis|  9.5    |
4  | LGH 202|       VW    |   Golf 6R | 2010-06 | 2024-08-21| Benzinas|    9    |
5  | BDI 666|     Audi    |        A6 | 2014-10 | 2024-03-15| Benzinas|  8.5    |
6  | KUL 323|      BMW    |       330 | 2018-10 | 2024-03-13| Benzinas|  9.5    |
7  | RIP 545|      BMW    |       118 | 2006-11 | 2022-12-10|  Dyzelis|  7.8    |
8  | LOS 222|       VW    |   Golf 7R | 2017-02 | 2024-04-25| Benzinas| 10.2    |
9  | KLO 144|     Audi    |        A5 | 2010-12 | 2023-08-29|  Dyzelis|  8.4    |
10 | GIL 999|       VW    | Golf 7GTI | 2015-05 | 2023-03-23| Benzinas|  9.4    |
11 | JES 783|      BMW    |       323 | 2011-07 | 2022-08-30|  Dyzelis|  8.2    |
12 | RUN 101|     Audi    |       RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8    |
13 | GEG 555|     Audi    |        A3 | 2015-05 | 2022-08-28| Benzinas|  7.8    |
-------------------------------------------------------------------------------


Two newest cars:
1. | RUN 101|     Audi    |       RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8    |
2. | KUL 323|      BMW    |       330 | 2018-10 | 2024-03-13| Benzinas|  9.5    |
```

All BMW cars

Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	JUL 123	BMW	123	2008-01	2022-05-22	Dyzelis	9.5
2	MHI 808	BMW	320	2007-04	2023-04-04	Dyzelis	9.5
3	KUL 323	BMW	330	2018-10	2024-03-13	Benzinas	9.5
4	RIP 545	BMW	118	2006-11	2022-12-10	Dyzelis	7.8
5	JES 783	BMW	323	2011-07	2022-08-30	Dyzelis	8.2

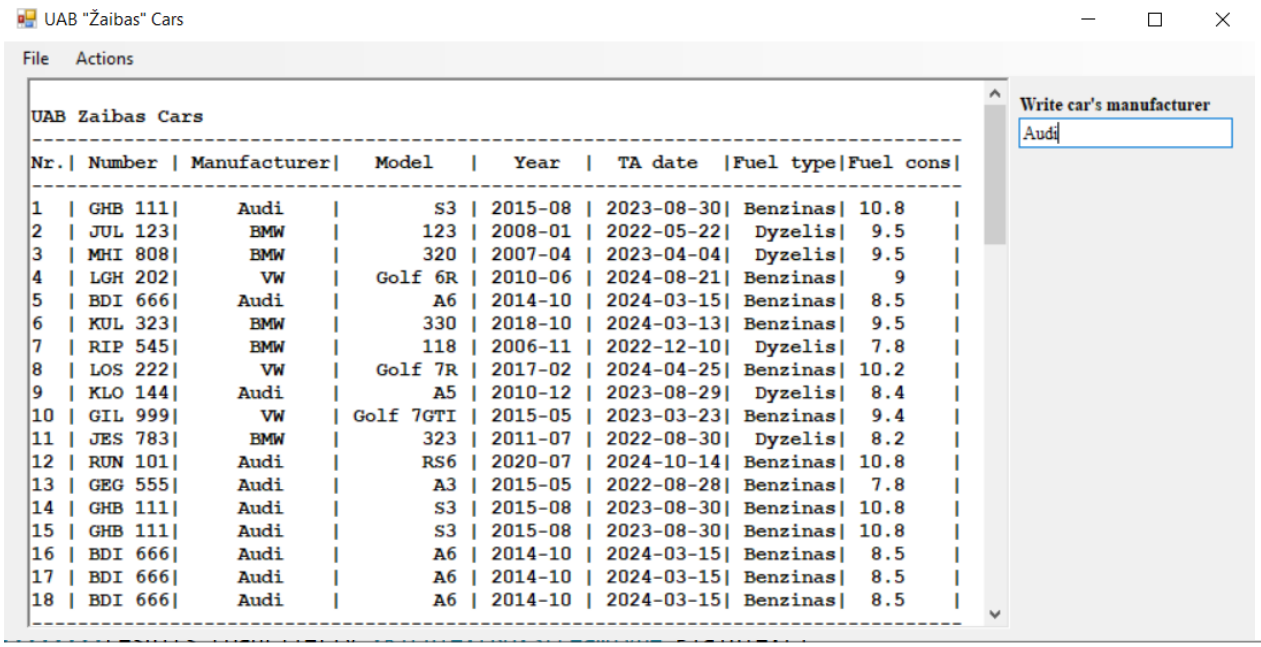
Sorted list

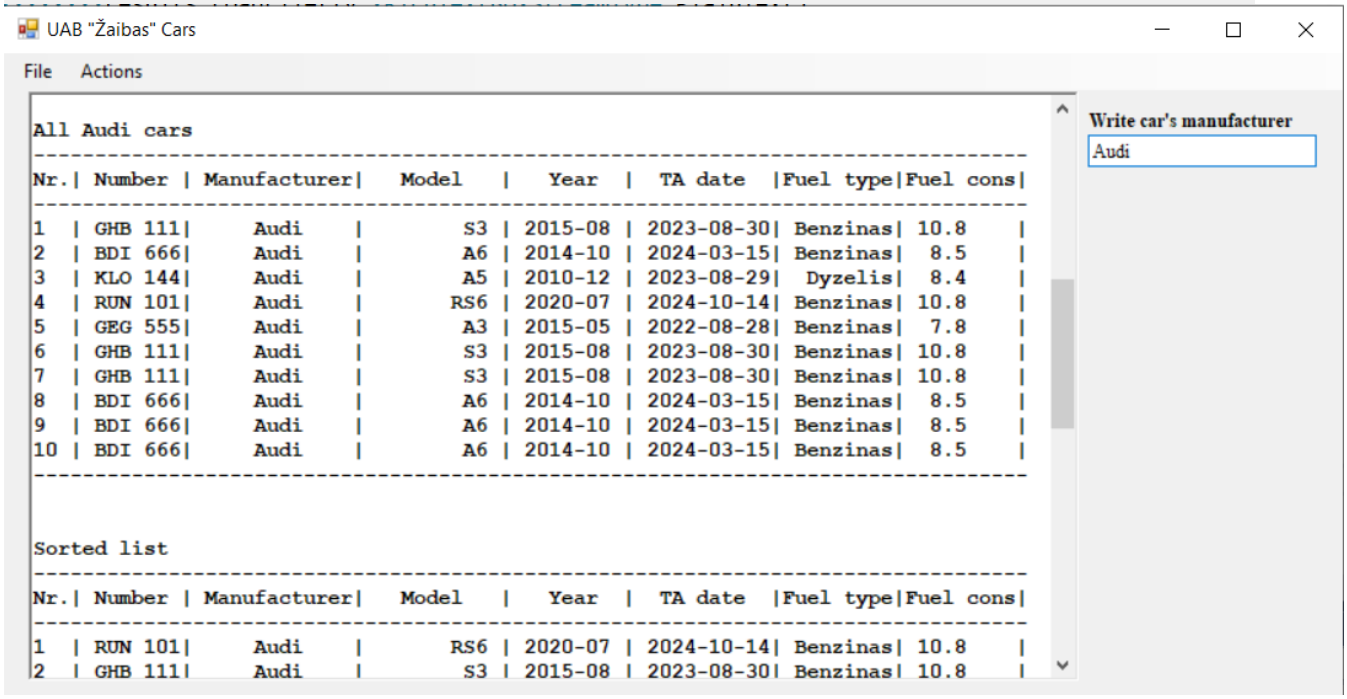
Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	KUL 323	BMW	330	2018-10	2024-03-13	Benzinas	9.5
2	JUL 123	BMW	123	2008-01	2022-05-22	Dyzelis	9.5
3	MHI 808	BMW	320	2007-04	2023-04-04	Dyzelis	9.5
4	JES 783	BMW	323	2011-07	2022-08-30	Dyzelis	8.2
5	RIP 545	BMW	118	2006-11	2022-12-10	Dyzelis	7.8

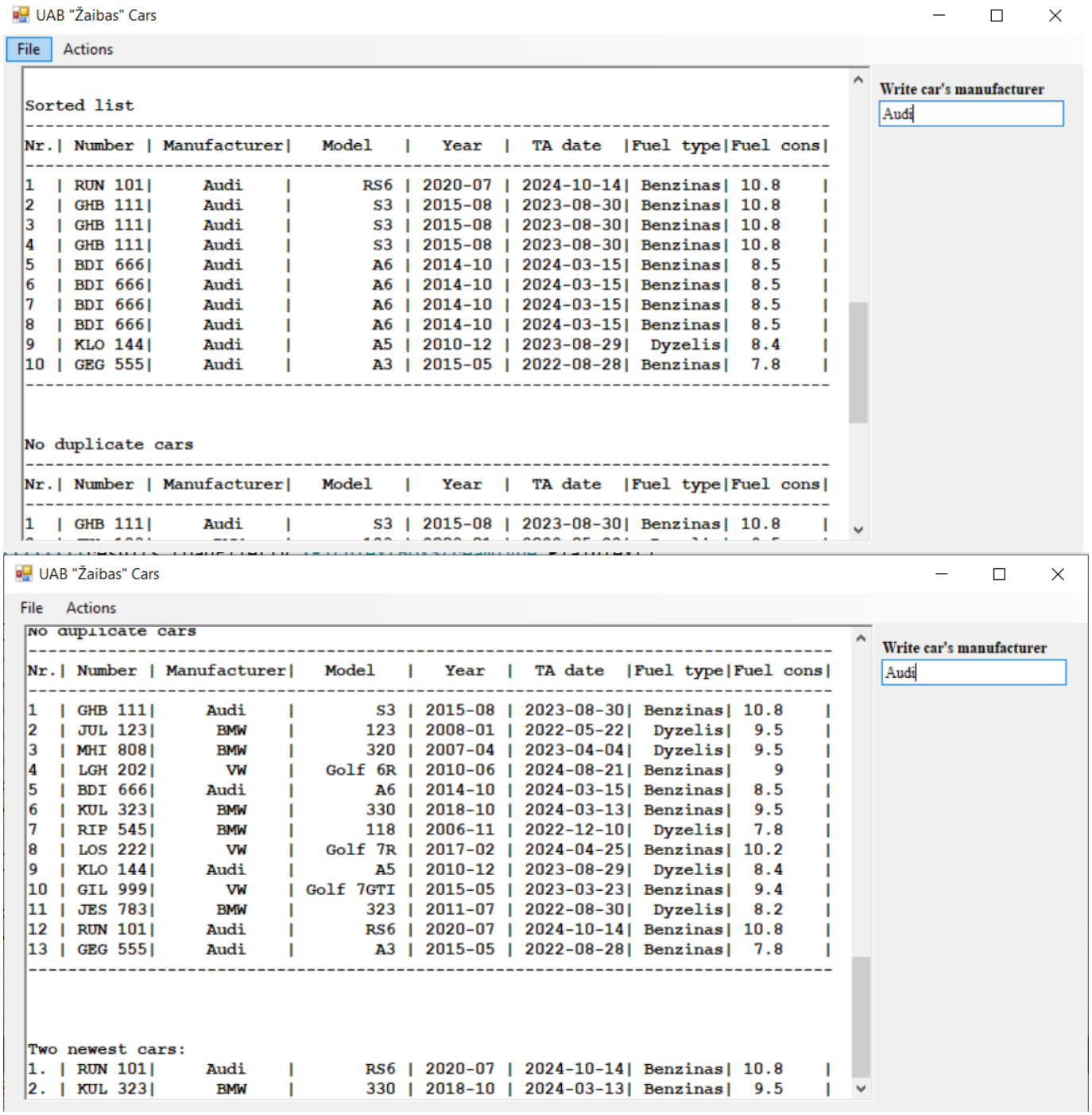
No duplicate cars

Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
2	JUL 123	BMW	123	2008-01	2022-05-22	Dyzelis	9.5
3	MHI 808	BMW	320	2007-04	2023-04-04	Dyzelis	9.5
4	LGH 202	VW	Golf 6R	2010-06	2024-08-21	Benzinas	9
5	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
6	KUL 323	BMW	330	2018-10	2024-03-13	Benzinas	9.5
7	RIP 545	BMW	118	2006-11	2022-12-10	Dyzelis	7.8
8	LOS 222	VW	Golf 7R	2017-02	2024-04-25	Benzinas	10.2
9	KLO 144	Audi	A5	2010-12	2023-08-29	Dyzelis	8.4
10	GIL 999	VW	Golf 7GTI	2015-05	2023-03-23	Benzinas	9.4
11	JES 783	BMW	323	2011-07	2022-08-30	Dyzelis	8.2
12	RUN 101	Audi	RS6	2020-07	2024-10-14	Benzinas	10.8
13	GEG 555	Audi	A3	2015-05	2022-08-28	Benzinas	7.8

Testas nr.2


UAB "Žaibas" Cars
File
Actions
UAB Zaibas Cars
Nr.| Number | Manufacturer| Model | Year | TA date |Fuel type|Fuel cons|
1 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
2 | JUL 123| BMW | 123 | 2008-01 | 2022-05-22| Dyzelis| 9.5 |
3 | MHI 808| BMW | 320 | 2007-04 | 2023-04-04| Dyzelis| 9.5 |
4 | LGH 202| VW | Golf 6R | 2010-06 | 2024-08-21| Benzinas| 9 |
5 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
6 | KUL 323| BMW | 330 | 2018-10 | 2024-03-13| Benzinas| 9.5 |
7 | RIP 545| BMW | 118 | 2006-11 | 2022-12-10| Dyzelis| 7.8 |
8 | LOS 222| VW | Golf 7R | 2017-02 | 2024-04-25| Benzinas| 10.2 |
9 | KLO 144| Audi | A5 | 2010-12 | 2023-08-29| Dyzelis| 8.4 |
10 | GIL 999| VW | Golf 7GTI | 2015-05 | 2023-03-23| Benzinas| 9.4 |
11 | JES 783| BMW | 323 | 2011-07 | 2022-08-30| Dyzelis| 8.2 |
12 | RUN 101| Audi | RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8 |
13 | GEG 555| Audi | A3 | 2015-05 | 2022-08-28| Benzinas| 7.8 |
14 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
15 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
16 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
17 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
18 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
Write car's manufacturer
Audi


UAB "Žaibas" Cars
File
Actions
All Audi cars
Nr.| Number | Manufacturer| Model | Year | TA date |Fuel type|Fuel cons|
1 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
2 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
3 | KLO 144| Audi | A5 | 2010-12 | 2023-08-29| Dyzelis| 8.4 |
4 | RUN 101| Audi | RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8 |
5 | GEG 555| Audi | A3 | 2015-05 | 2022-08-28| Benzinas| 7.8 |
6 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
7 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
8 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
9 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
10 | BDI 666| Audi | A6 | 2014-10 | 2024-03-15| Benzinas| 8.5 |
Sorted list
Nr.| Number | Manufacturer| Model | Year | TA date |Fuel type|Fuel cons|
1 | RUN 101| Audi | RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8 |
2 | GHB 111| Audi | S3 | 2015-08 | 2023-08-30| Benzinas| 10.8 |
Write car's manufacturer
Audi



Rezultatu failas:

UAB Zaibas Cars

Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
2	JUL 123	BMW	123	2008-01	2022-05-22	Dyzelis	9.5
3	MHI 808	BMW	320	2007-04	2023-04-04	Dyzelis	9.5
4	LGH 202	VW	Golf 6R	2010-06	2024-08-21	Benzinas	9
5	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
6	KUL 323	BMW	330	2018-10	2024-03-13	Benzinas	9.5
7	RIP 545	BMW	118	2006-11	2022-12-10	Dyzelis	7.8
8	LOS 222	VW	Golf 7R	2017-02	2024-04-25	Benzinas	10.2
9	KLO 144	Audi	A5	2010-12	2023-08-29	Dyzelis	8.4
10	GIL 999	VW	Golf 7GTI	2015-05	2023-03-23	Benzinas	9.4

11	JES 783	BMW	323	2011-07	2022-08-30	Dyzelis	8.2
12	RUN 101	Audi	RS6	2020-07	2024-10-14	Benzinas	10.8
13	GEG 555	Audi	A3	2015-05	2022-08-28	Benzinas	7.8
14	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
15	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
16	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
17	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
18	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5

Two newest cars:

1.	RUN 101	Audi	RS6	2020-07	2024-10-14	Benzinas	10.8
2.	KUL 323	BMW	330	2018-10	2024-03-13	Benzinas	9.5

All Audi cars

Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
2	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
3	KLO 144	Audi	A5	2010-12	2023-08-29	Dyzelis	8.4
4	RUN 101	Audi	RS6	2020-07	2024-10-14	Benzinas	10.8
5	GEG 555	Audi	A3	2015-05	2022-08-28	Benzinas	7.8
6	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
7	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
8	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
9	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
10	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5

Sorted list

Nr.	Number	Manufacturer	Model	Year	TA date	Fuel type	Fuel cons
1	RUN 101	Audi	RS6	2020-07	2024-10-14	Benzinas	10.8
2	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
3	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
4	GHB 111	Audi	S3	2015-08	2023-08-30	Benzinas	10.8
5	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
6	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
7	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
8	BDI 666	Audi	A6	2014-10	2024-03-15	Benzinas	8.5
9	KLO 144	Audi	A5	2010-12	2023-08-29	Dyzelis	8.4
10	GEG 555	Audi	A3	2015-05	2022-08-28	Benzinas	7.8

```
No duplicate cars
-------------------------------------------------------------------------------
Nr.| Number | Manufacturer|   Model   |   Year  |  TA date  |Fuel type|Fuel cons|
-------------------------------------------------------------------------------
1  | GHB 111|     Audi    |        S3 | 2015-08 | 2023-08-30| Benzinas| 10.8    |
2  | JUL 123|      BMW    |       123 | 2008-01 | 2022-05-22|  Dyzelis|  9.5    |
3  | MHI 808|      BMW    |       320 | 2007-04 | 2023-04-04|  Dyzelis|  9.5    |
4  | LGH 202|       VW    |   Golf 6R | 2010-06 | 2024-08-21| Benzinas|    9    |
5  | BDI 666|     Audi    |        A6 | 2014-10 | 2024-03-15| Benzinas|  8.5    |
6  | KUL 323|      BMW    |       330 | 2018-10 | 2024-03-13| Benzinas|  9.5    |
7  | RIP 545|      BMW    |       118 | 2006-11 | 2022-12-10|  Dyzelis|  7.8    |
8  | LOS 222|       VW    |   Golf 7R | 2017-02 | 2024-04-25| Benzinas| 10.2    |
9  | KLO 144|     Audi    |        A5 | 2010-12 | 2023-08-29|  Dyzelis|  8.4    |
10 | GIL 999|       VW    | Golf 7GTI | 2015-05 | 2023-03-23| Benzinas|  9.4    |
11 | JES 783|      BMW    |       323 | 2011-07 | 2022-08-30|  Dyzelis|  8.2    |
12 | RUN 101|     Audi    |       RS6 | 2020-07 | 2024-10-14| Benzinas| 10.8    |
13 | GEG 555|     Audi    |        A3 | 2015-05 | 2022-08-28| Benzinas|  7.8    |
-------------------------------------------------------------------------------
```

3.7. Dėstytojo pastabos

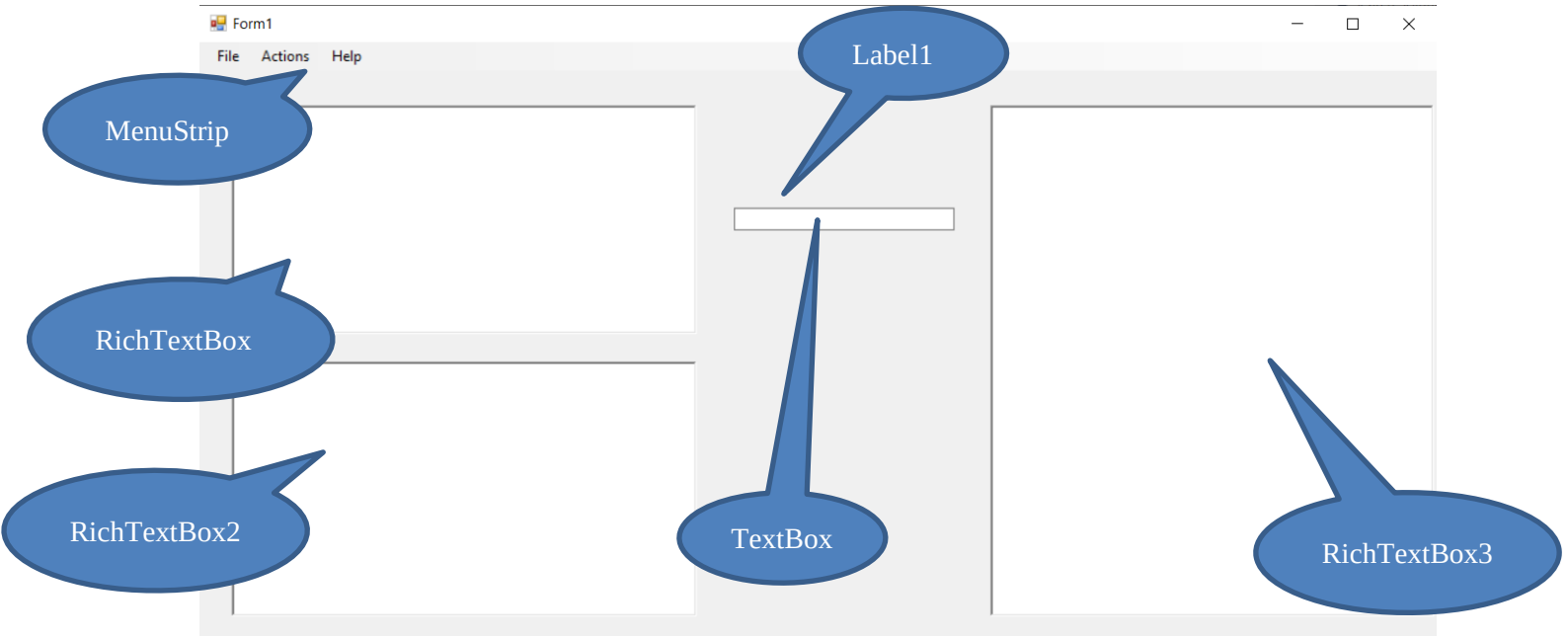
4. Susietasis sąrašas (L4)

4.1. Darbo užduotis

Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Turime dviejų mokyklų duomenis. L1+L2+L4.

- Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis kiekvienoje mokykloje.
- Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.
- Surikiuokite rezultatų sąrašą amžiaus didėjimo tvarka.
- Pašalinkite iš rezultatų sąrašo krepšininkus, kurių amžius yra didesnis už nurodytą klaviatūra.

4.2. Grafinės vartotojo sąsajos schema ir paveikslas



4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Results (RichTextBox)	Font	Times New Roman, 11, script Baltic
Results2 (RichTextBox2)	Font	Times New Roman, 11, script Baltic
Results3 (RichTextBox3)	Font	Courier New, Bold, 11, script Baltic
Įvesti (TextBox)	Font	Times New Roman, 11, script Baltic
Label1	Font	Times New Roman, 11, script Baltic

4.4. Programos vartotojo vadovas

- Duomenys apie komandas yra surašyti tekstiniuose failuose „Player1.txt“ ir „Player2.txt“. Pirmoje eilutėje užrašytas komandos pavadinimas. Kitose eilutėse užrašyti duomenys apie krepšininkus: *Vardas ir Pavardė, Amžius, Ūgis.*
- Programa veikia meniu principu. Programa paleidžiama paspaudus mygtuką „File -> Enter“ ir pasirinkus du failus, tuomet atspausdinami pradiniai duomenys apie pirmąsias dvi komandas.
- Paspaudus mygtuką „Actions -> Run“ suskaičiuojama ir atspausdinama abiejų komandų ūgio bei amžiaus vidurkiai. Sudaromas bei atspausdinamas krepšininkų sąrašas iš abiejų komandų, kurių krepšininkų ūgis didesnis už komandos ūgio vidurkį. Tas sąrašas surikiuojamas pagal amžių didėjimo tvarka ir atspausdinamas dar kartą.
- Langelyje žemiau teksto „Type player‘s age“ įvedamas amžius pagal kurį bus koreguojamas surikiuotas sąrašas. Paspaudus mygtuką „Actions -> Remove“ išmetami visi krepšininkai kurių amžius didesnis už įvestą ir sąrašas atspausdinamas.

4.5. Programos tekstas

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

namespace LAB4_sem_2_
{
    public sealed class knot
    {
        public Player Data { get; set; }
        public knot Next { get; set; }
        public knot() { }
        public knot(Player Data, knot address)
        {
            this.Data = Data;
            Next = address;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LAB4_sem_2_
{
    /// <summary>
    /// class to describe player
    /// </summary>
    public class Player
    {
        public string NamSur { get; set; } //Player's name and surname
        public int Height { get; set; }    //player's height
        public int Age { get; set; }       //player's age

        /// <summary>
        /// constructor
        /// </summary>
        /// <param name="NamS"></param>
        /// <param name="age"></param>
        /// <param name="Hght"></param>
        public Player(string NamS, int age, int Hght)
        {
            NamSur = NamS;
```

```
        Height = Hght;
        Age = age;
    }

    /// <summary>
    /// new names
    /// </summary>
    /// <param name="a"></param>
    /// <param name="b"></param>
    /// <param name="c"></param>
    void Add(string a, int b, int c)
    {
        NamSur = a;
        Height = b;
        Age = c;
    }

    /// <summary>
    /// Overriden Object class method
    /// </summary>
    /// <returns>printing format for player</returns>
    public override string ToString()
    {
        string line;
        line = string.Format("{0, -17} {1, 2} {2,6}", NamSur, Age, Height);
        return line;
    }

    /// <summary>
    /// Overriden Object class method
    /// </summary>
    /// <param name="obj">object</param>
    /// <returns>player with same name</returns>
    public override bool Equals(object obj)
    {
        Player plr = obj as Player;
        return plr.NamSur == NamSur;
    }

    /// <summary>
    /// Overriden Object class method
    /// </summary>
    /// <returns>hash code</returns>
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }


    /// <summary>
    /// compares two players by age and returns older one, if ages are equal compares
names alphabetically
    /// </summary>
    /// <param name="pl1"></param>
    /// <param name="pl2"></param>
    /// <returns></returns>
    public static bool operator >=(Player pl1, Player pl2)
    {
        int p = String.Compare(pl1.NamSur, pl2.NamSur, StringComparison.CurrentCulture);
        return ((pl1.Age > pl2.Age) || (pl1.Age == pl2.Age) && (p > 0));

    }
```

```
        /// <summary>
        /// compares two players by age and returns younger one, if ages are equal compares
       names alphabetically
        /// </summary>
        /// <param name="pl1"></param>
        /// <param name="pl2"></param>
        /// <returns></returns>
        public static bool operator <=(Player pl1, Player pl2)
        {
            int p = String.Compare(pl1.NamSur, pl2.NamSur, StringComparison.CurrentCulture);
            return ((pl1.Age < pl2.Age) || (pl1.Age == pl2.Age) && (p < 0));
        }
    }
}

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

namespace LAB4_sem_2_
{
    /// <summary>
    /// linked list container class
    /// </summary>
    public sealed class Ballers : IEnumerable
    {
        private knot pr; // start
        private knot pb; // end
        private knot ss; // link


        /// <summary>
        /// constructor
        /// </summary>
        public Ballers()
        {
            pr = null;
            pb = null;
            ss = null;
        }


        /// <summary>
        /// IEnumerator method
        /// </summary>
        /// <returns></returns>
        public IEnumerator GetEnumerator()
        {
            for (knot ss = pr; ss != null; ss = ss.Next)
            {
                yield return ss.Data;
            }
        }


        /// <summary>
        /// Adds data in reverse order
        /// </summary>
        /// <param name="pl"></param>
        public void AddDataA(Player pl)
        {
            var d = new knot(pl, null);
            d.Next = pr;
            pr = d;
        }
```

```
/// <summary>
/// Adds data in direct order
/// </summary>
/// <param name="pl"></param>
public void AddDataT(Player pl)
{
    var d = new knot(pl, null);
    if(pr != null)
    {
        pb.Next = d;
        pb = d;
    }
    else
    {
        pr = d;
        pb = d;
    }
}
//public void Papildyti(Player duom)
//{
//    Mazgas d1 = new Mazgas();
//    d1.Data = duom;
//    d1.Next = pr;
//    pr = d1;
//}


/// <summary>
/// returns data
/// </summary>
/// <returns></returns>
public Player GetData()
{
    return ss.Data;
}


/// <summary>
/// returns start
/// </summary>
public void Start()
{
    ss = pr;
}


/// <summary>
/// returns next
/// </summary>
public void Next()
{
    ss = ss.Next;
}


/// <summary>
/// return link not equal to null
/// </summary>
/// <returns></returns>
public bool Is()
{
    return ss != null;
}
```

```
        /// <summary>
        /// destroys list
        /// </summary>
        public void Destroy()
        {
            while (pr != null)
            {
                ss = pr;
                pr = pr.Next;
                ss.Next = null;
            }
            pb = ss = pr;
        }


        /// <summary>
        /// Sorts list
        /// </summary>
        public void Burbulas()
        {
            if (pr == null) { return; }
            bool kt = true;
            while (kt)
            {
                kt = false;
                knot d = pr;
                while(d.Next != null)
                {
                    if(d.Next.Data <= d.Data)
                    {
                        Player pl = d.Data;
                        d.Data = d.Next.Data;
                        d.Next.Data = pl;
                        kt = true;
                    }
                    d = d.Next;
                }
            }
        }


        /// <summary>
        /// Removes player
        /// </summary>
        /// <param name="pl"></param>
        public void RemoveV(Player pl)
        {

            knot d1 = pr;
            while (d1 != null && d1.Next != null && d1.Data != pl)
                d1 = d1.Next;
            knot v = d1;
            d1 = d1.Next;
            v.Data = null;
            v = null;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LAB4_sem_2_
{
    public partial class Form1 : Form
    {

        const string CFr = "..\\..\\Results.txt";    // result's file
        const string Cffd = "..\\..\\Task.txt";
        const string Cfh = "..\\..\\user.txt";

        Ballers A; //direct
        Ballers B; //reverse
        Ballers C; //combined
        string TeamName;  //Team 1 name
        string TeamName1; //Team 2 name


        /// <summary>
        /// Read data from file and puts in reverse order
        /// </summary>
        /// <param name="fn">file name</param>
        /// <param name="TeamName">returns team's name</param>
        static Ballers ReadFileA(string fn, out string TeamName)
        {
            var a = new Ballers();
            using (var reader = new StreamReader(fn))
            {
                string line;
                line = reader.ReadLine();
                TeamName = line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] parts = line.Split(';');
                    string NamSur = parts[0];
                    int Age = int.Parse(parts[1]);
                    int Height = int.Parse(parts[2]);
                    Player plr = new Player(NamSur, Age, Height);
                    a.AddDataA(plr);
                }
            }
            return a;
        }
        /// <summary>
        /// Read data from file and puts in direct order
        /// </summary>
        /// <param name="fn">file name</param>
        /// <param name="TeamName">returns team's name</param>
        static Ballers ReadFileT(string fn, out string TeamName)
        {
            var a = new Ballers();
            using (var reader = new StreamReader(fn))
            {
                string line;
                line = reader.ReadLine();
                TeamName = line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] parts = line.Split(';');
```

```
            string NamSur = parts[0];
            int Age = int.Parse(parts[1]);
            int Height = int.Parse(parts[2]);
            Player plr = new Player(NamSur, Age, Height);
            a.AddDataT(plr);
        }
    }
    return a;
}

/// <summary>
/// Print's a table with a team's players
/// </summary>
/// <param name="fn">file name</param>
/// <param name="Team">Teams</param>
/// <param name="heading">heading</param>
static void Print(string fn, Ballers B, string heading)
{
    const string top =
            "-----------------------------------\r\n"
            + "Nr. |Name and Surname| Age| Height| \r\n"
            + "-----------------------------------";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n" + heading);
        fr.WriteLine(top);
        int i=0;
        for (B.Start();B.Is();B.Next())
        {
            i++;
            Player plr = B.GetData();
            fr.WriteLine("{0}     {1}", i , plr);
        }
        fr.WriteLine("-----------------------------------\n");
    }
}

/// <summary>
/// Connects two Team's players into one array whose height is bigger than avarage
/// </summary>
/// <param name="T">Team 1</param>
/// <param name="T1">Team 2</param>
/// <returns>combined team of two team's</returns>
static void ConnectByHeight(Ballers T, Ballers T2)
{
    double avg = AvgHeight(T);
    for (T.Start(); T.Is(); T.Next())
    {
        Player plr = T.GetData();
        if (plr.Height > avg)
        {
            T2.AddDataT(plr);
        }
    }
}

/// <summary>
/// Returns team's avarage age
/// </summary>
/// <param name="T">Team</param>
/// <returns>team's avarage age</returns>
static double AvgAge(Ballers T)
{
    int count = 0;
    double a = 0;
    double avg = 0;
    foreach (Player pl in T)
    {
```

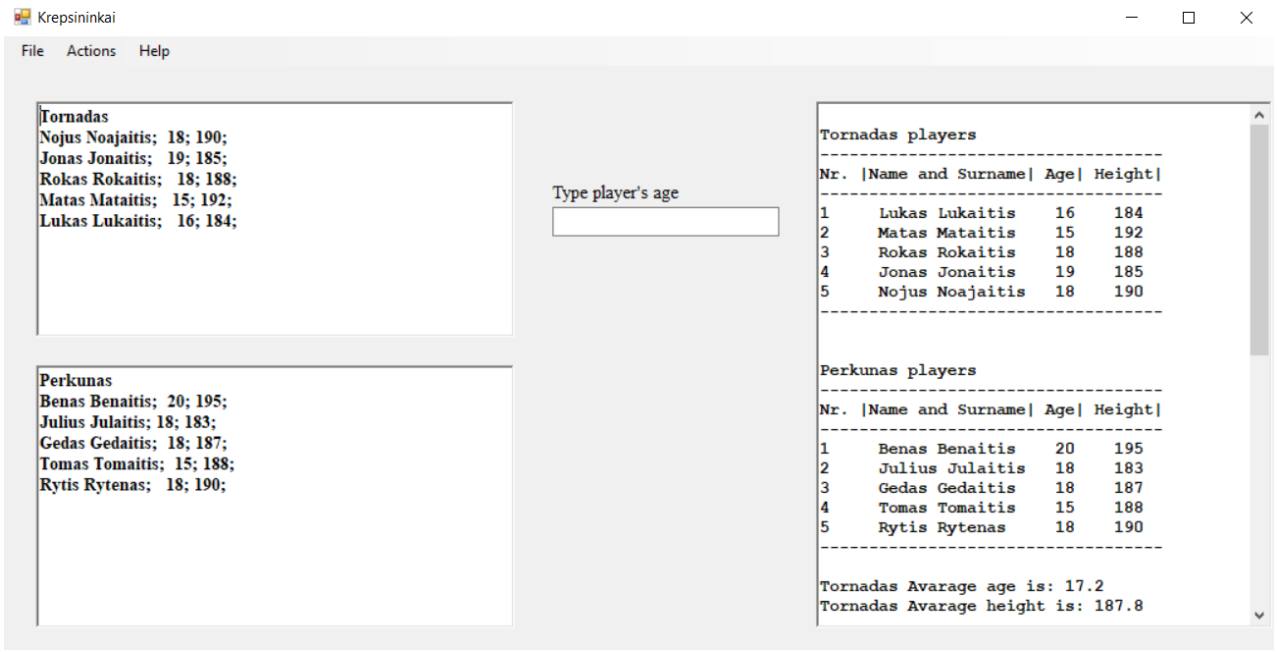
```
            count++;
            a = a + pl.Age;
        }
        if (count != 0)
            avg = a / count;
        return avg;
    }


    /// <summary>
    /// Returns team's avarage height
    /// </summary>
    /// <param name="T">Team</param>
    /// <returns>team's avarage height</returns>
    static double AvgHeight(Ballers T)
    {
        int count = 0;
        double a = 0;
        double avg = 0;
        foreach (Player pl in T)
        {
            count++;
            a = a + pl.Height;
        }
        if (count != 0)
            avg = a / count;
        return avg;
    }

    /// <summary>
    /// Removes players whose age is bigger than specified
    /// </summary>
    /// <param name="A"></param>
    /// <param name="ages"></param>
    static void Remove(Ballers A, int ages)
    {
        for (A.Start(); A.Is(); A.Next())
        {
            Player pl = A.GetData();
            if (pl.Age > ages)
            {
                A.RemoveV(pl);
            }
        }
    }

    public Form1()
    {
        InitializeComponent();

        if (File.Exists(CFr))
            File.Delete(CFr);
        removeToolStripMenuItem.Enabled = false;
        runToolStripMenuItem.Enabled = false;
    }
    /// <summary>
    /// Actions of the "Enter" menu click
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void enterToolStripMenuItem_Click(object sender, EventArgs e)
    {
        runToolStripMenuItem.Enabled = true;
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
        openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog1.Title = "Choose a file";
        DialogResult result = openFileDialog1.ShowDialog();
        if (result == DialogResult.OK)
```

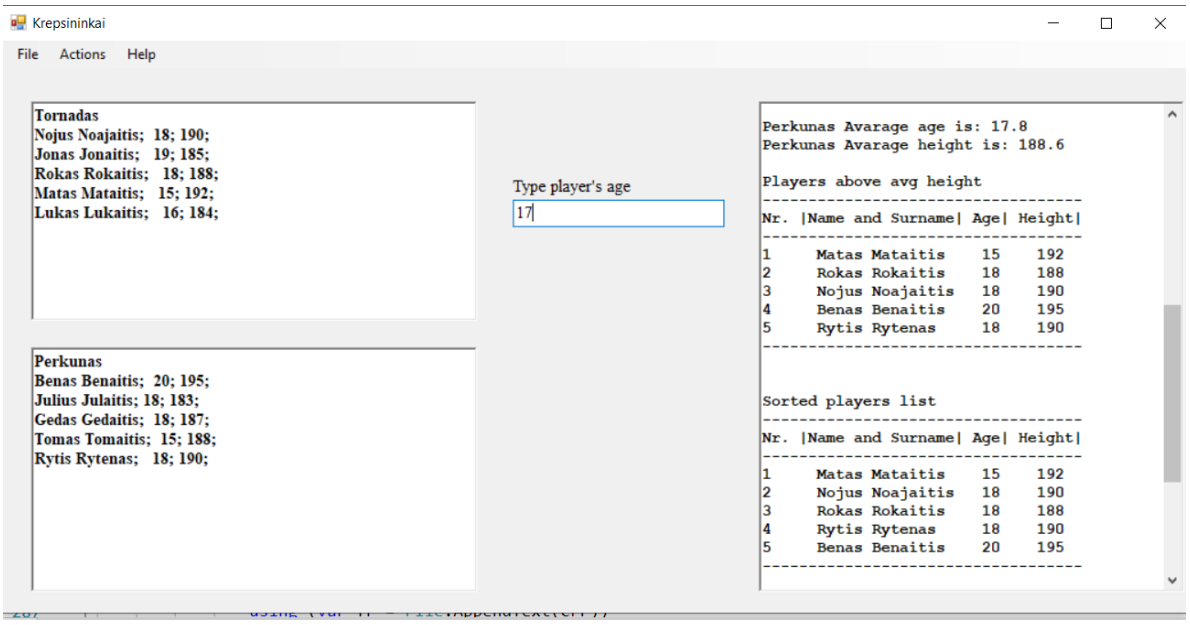
```
    {
        string fv = openFileDialog1.FileName;
        A = ReadFileA(fv, out TeamName);
        results.LoadFile(fv, RichTextBoxStreamType.PlainText);

    }
    OpenFileDialog openFileDialog2 = new OpenFileDialog();
    openFileDialog2.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    openFileDialog2.Title = "Choose a file";
    DialogResult result1 = openFileDialog1.ShowDialog();
    if (result1 == DialogResult.OK)
    {
        string fv1 = openFileDialog1.FileName;
        B = ReadFileT(fv1, out TeamName1);
        results2.LoadFile(fv1, RichTextBoxStreamType.PlainText);

    }
    Print(CFr, A, TeamName + " players");
    Print(CFr, B, TeamName1 + " players");
    label1.Text = "";
}

/// <summary>
/// Actions of the "Close" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

/// <summary>
/// Actions of the "Run" menu click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void runToolStripMenuItem_Click(object sender, EventArgs e)
{
    removeToolStripMenuItem.Enabled = true;
    runToolStripMenuItem.Enabled = false;
    using (var fr = File.AppendText(CFr))
    {
        fr.WriteLine(TeamName + " Avarage age is: " + AvgAge(A));
        fr.WriteLine(TeamName + " Avarage height is: " + AvgHeight(A));
        fr.WriteLine("\n\n" + TeamName1 + " Avarage age is: " + AvgAge(B));
        fr.WriteLine(TeamName1 + " Avarage height is: " + AvgHeight(B));

    }
    C = new Ballers();
    ConnectByHeight(A, C);
    ConnectByHeight(B, C);
    Print(CFr, C, "Players above avg height");
    C.Burbulas();
    Print(CFr, C, "Sorted players list");
    results3.LoadFile(CFr, RichTextBoxStreamType.PlainText);
    label1.Text = "Type player's age";
}
```

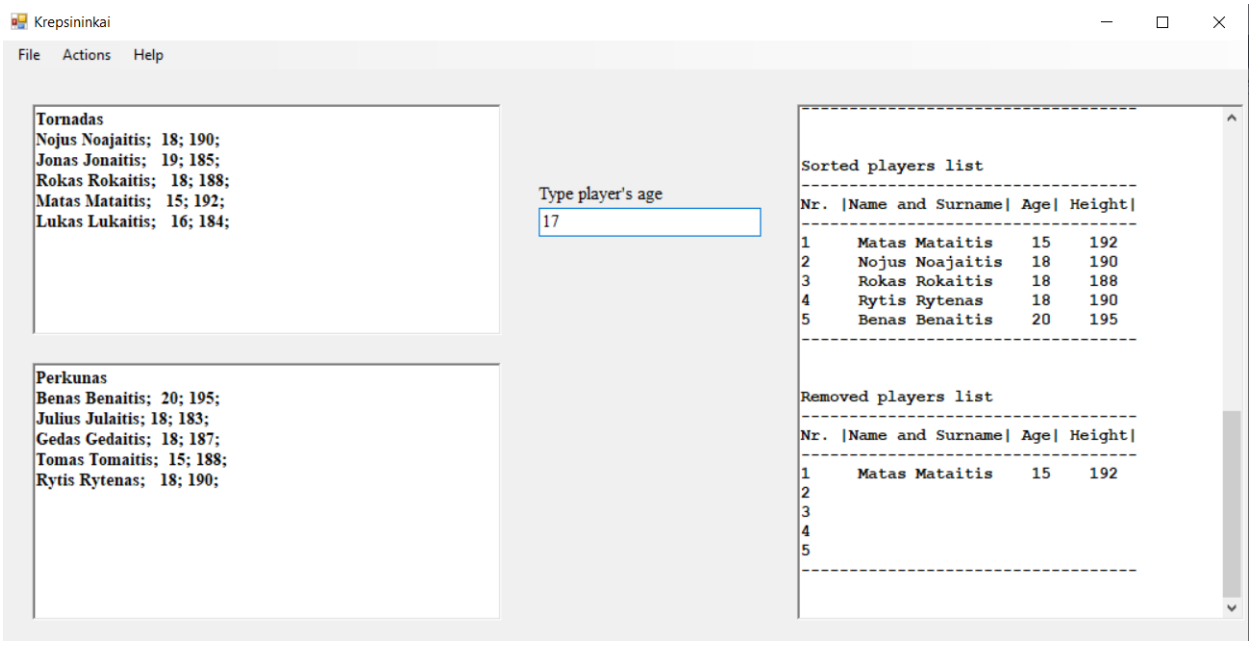
```
        /// <summary>
        /// Actions of the "Remove" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void removeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            runToolStripMenuItem.Enabled = false;
            removeToolStripMenuItem.Enabled = false;
            int age = int.Parse(Įvesti.Text);
            Remove(C, age);
            C.Start();
            if (C.GetData()!= null)
            {
                Print(CFr, C, "Removed players list");
                results3.LoadFile(CFr, RichTextBoxStreamType.PlainText);
            }
            else
            {
                results3.Text += "\n\nThere are no players with that age";
                using (var fr = File.AppendText(CFr))
                    fr.WriteLine("There are no players with that age");
            }
        }

        /// <summary>
        /// Actions of the "Task" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void taskToolStripMenuItem_Click(object sender, EventArgs e)
        {
            results.Clear();
            results2.Clear();
            results3.Clear();
            results3.LoadFile(Cffd, RichTextBoxStreamType.PlainText);
            removeToolStripMenuItem.Enabled = false;
            runToolStripMenuItem.Enabled = false;
            enterToolStripMenuItem.Enabled = true;
        }

        /// <summary>
        /// Actions of the "User Instructions" menu click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userToolStripMenuItem_Click(object sender, EventArgs e)
        {
            results.Clear();
            results2.Clear();
            results3.Clear();
            results3.LoadFile(Cfh, RichTextBoxStreamType.PlainText);
            removeToolStripMenuItem.Enabled = false;
            runToolStripMenuItem.Enabled = false;
            enterToolStripMenuItem.Enabled = true;
        }
    }
}
```

4.6. Pradiniai duomenys ir rezultatai

Testas nr. 1



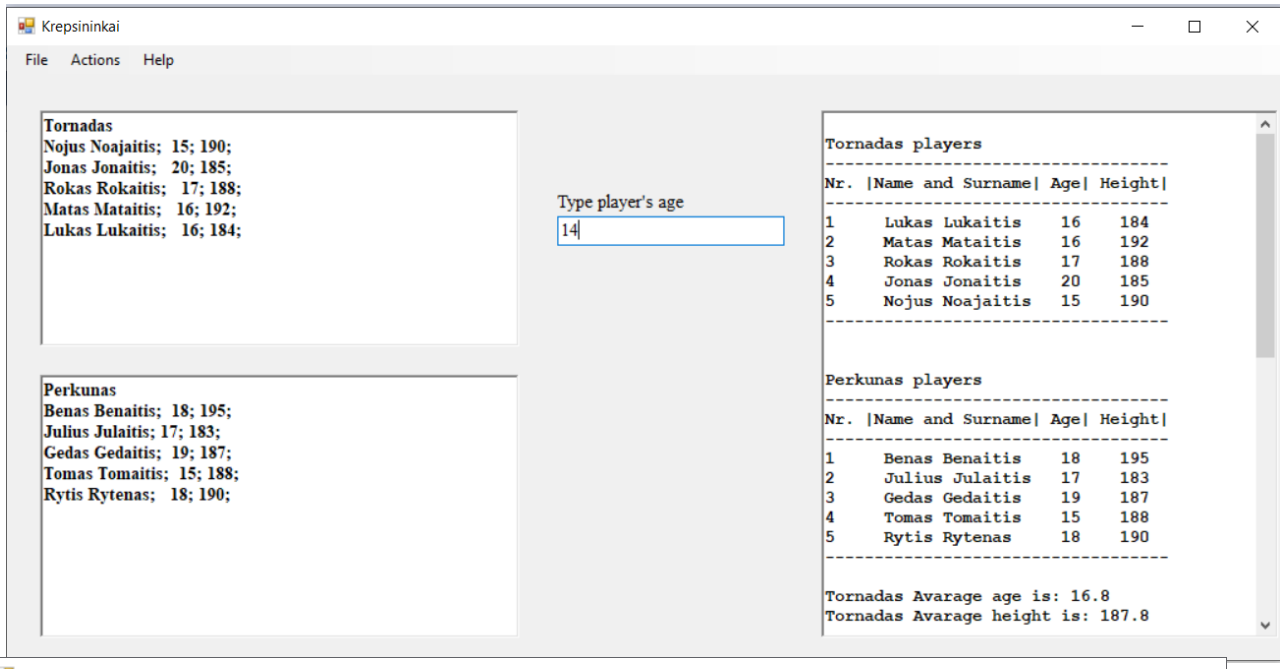


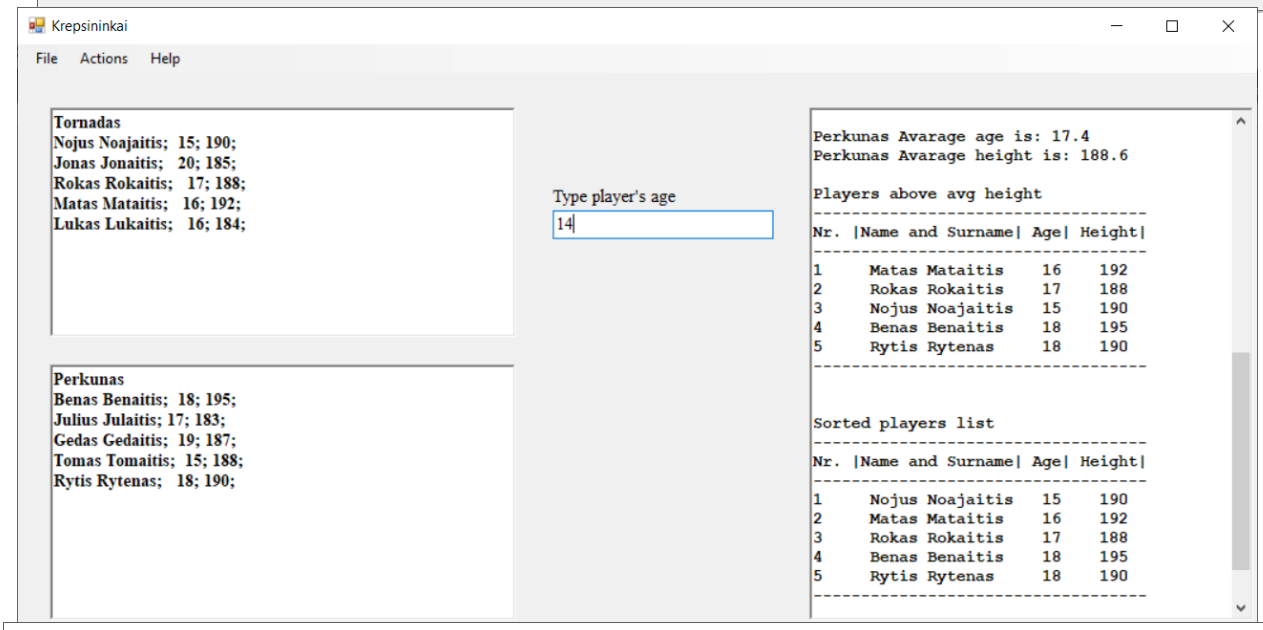


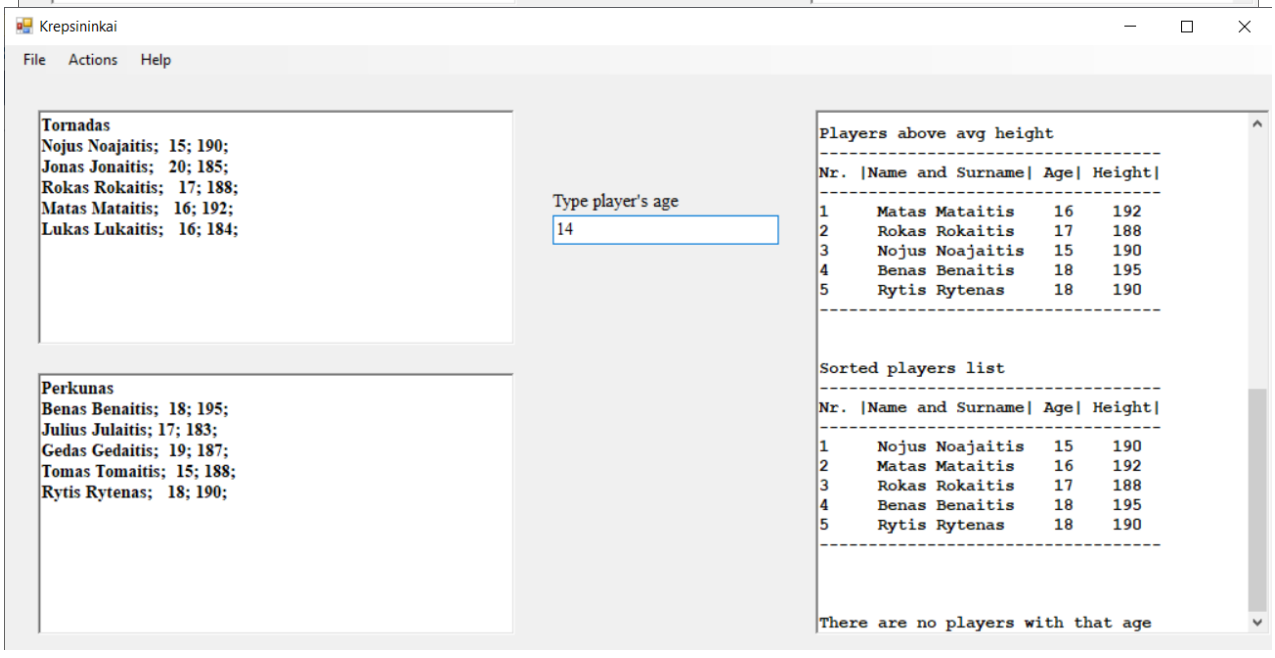
Results.txt

```
Tornadas players
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Lukas Lukaitis    16    184
2     Matas Mataitis    15    192
3     Rokas Rokaitis    18    188
4     Jonas Jonaitis    19    185
5     Nojus Noajaitis   18    190
-----------------------------------


Perkunas players
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Benas Benaitis    20    195
2     Julius Julaitis   18    183
3     Gedas Gedaitis    18    187
4     Tomas Tomaitis    15    188
5     Rytis Rytenas     18    190
-----------------------------------

Tornadas Avarage age is: 17.2
Tornadas Avarage height is: 187.8


Perkunas Avarage age is: 17.8
Perkunas Avarage height is: 188.6

Players above avg height
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Matas Mataitis    15    192
2     Rokas Rokaitis    18    188
3     Nojus Noajaitis   18    190
4     Benas Benaitis    20    195
5     Rytis Rytenas     18    190
-----------------------------------


Sorted players list
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Matas Mataitis    15    192
2     Nojus Noajaitis   18    190
3     Rokas Rokaitis    18    188
4     Rytis Rytenas     18    190
5     Benas Benaitis    20    195
-----------------------------------

Removed players list
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Matas Mataitis    15    192
2
3
4
5
-----------------------------------
```

Testas nr. 2


Krepsininkai
File Actions Help
Tornadas
Nojus Noajaitis; 15; 190;
Jonas Jonaitis; 20; 185;
Rokas Rokaitis; 17; 188;
Matas Mataitis; 16; 192;
Lukas Lukaitis; 16; 184;
Perkunas
Benas Benaitis; 18; 195;
Julius Julaitis; 17; 183;
Gedas Gedaitis; 19; 187;
Tomas Tomaitis; 15; 188;
Rytis Rytenas; 18; 190;
Type player's age
14
Tornadas players
Nr. |Name and Surname| Age| Height|
1 Lukas Lukaitis 16 184
2 Matas Mataitis 16 192
3 Rokas Rokaitis 17 188
4 Jonas Jonaitis 20 185
5 Nojus Noajaitis 15 190
Perkunas players
Nr. |Name and Surname| Age| Height|
1 Benas Benaitis 18 195
2 Julius Julaitis 17 183
3 Gedas Gedaitis 19 187
4 Tomas Tomaitis 15 188
5 Rytis Rytenas 18 190
Tornadas Avarage age is: 16.8
Tornadas Avarage height is: 187.8


Krepsininkai
File Actions Help
Tornadas
Nojus Noajaitis; 15; 190;
Jonas Jonaitis; 20; 185;
Rokas Rokaitis; 17; 188;
Matas Mataitis; 16; 192;
Lukas Lukaitis; 16; 184;
Perkunas
Benas Benaitis; 18; 195;
Julius Julaitis; 17; 183;
Gedas Gedaitis; 19; 187;
Tomas Tomaitis; 15; 188;
Rytis Rytenas; 18; 190;
Type player's age
14
Perkunas Avarage age is: 17.4
Perkunas Avarage height is: 188.6
Players above avg height
Nr. |Name and Surname| Age| Height|
1 Matas Mataitis 16 192
2 Rokas Rokaitis 17 188
3 Nojus Noajaitis 15 190
4 Benas Benaitis 18 195
5 Rytis Rytenas 18 190
Sorted players list
Nr. |Name and Surname| Age| Height|
1 Nojus Noajaitis 15 190
2 Matas Mataitis 16 192
3 Rokas Rokaitis 17 188
4 Benas Benaitis 18 195
5 Rytis Rytenas 18 190


Krepsininkai
File Actions Help
Tornadas
Nojus Noajaitis; 15; 190;
Jonas Jonaitis; 20; 185;
Rokas Rokaitis; 17; 188;
Matas Mataitis; 16; 192;
Lukas Lukaitis; 16; 184;
Perkunas
Benas Benaitis; 18; 195;
Julius Julaitis; 17; 183;
Gedas Gedaitis; 19; 187;
Tomas Tomaitis; 15; 188;
Rytis Rytenas; 18; 190;
Type player's age
14
Players above avg height
Nr. |Name and Surname| Age| Height|
1 Matas Mataitis 16 192
2 Rokas Rokaitis 17 188
3 Nojus Noajaitis 15 190
4 Benas Benaitis 18 195
5 Rytis Rytenas 18 190
Sorted players list
Nr. |Name and Surname| Age| Height|
1 Nojus Noajaitis 15 190
2 Matas Mataitis 16 192
3 Rokas Rokaitis 17 188
4 Benas Benaitis 18 195
5 Rytis Rytenas 18 190
There are no players with that age

Results.txt

```
Tornadas players
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Lukas Lukaitis    16    184
2     Matas Mataitis    16    192
3     Rokas Rokaitis    17    188
4     Jonas Jonaitis    20    185
5     Nojus Noajaitis   15    190
-----------------------------------


Perkunas players
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Benas Benaitis    18    195
2     Julius Julaitis   17    183
3     Gedas Gedaitis    19    187
4     Tomas Tomaitis    15    188
5     Rytis Rytenas     18    190
-----------------------------------

Tornadas Avarage age is: 16.8
Tornadas Avarage height is: 187.8


Perkunas Avarage age is: 17.4
Perkunas Avarage height is: 188.6

Players above avg height
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Matas Mataitis    16    192
2     Rokas Rokaitis    17    188
3     Nojus Noajaitis   15    190
4     Benas Benaitis    18    195
5     Rytis Rytenas     18    190
-----------------------------------


Sorted players list
-----------------------------------
Nr. |Name and Surname| Age| Height|
-----------------------------------
1     Nojus Noajaitis   15    190
2     Matas Mataitis    16    192
3     Rokas Rokaitis    17    188
4     Benas Benaitis    18    195
5     Rytis Rytenas     18    190
-----------------------------------

There are no players with that age
```

4.7. Dėstytojo pastabos

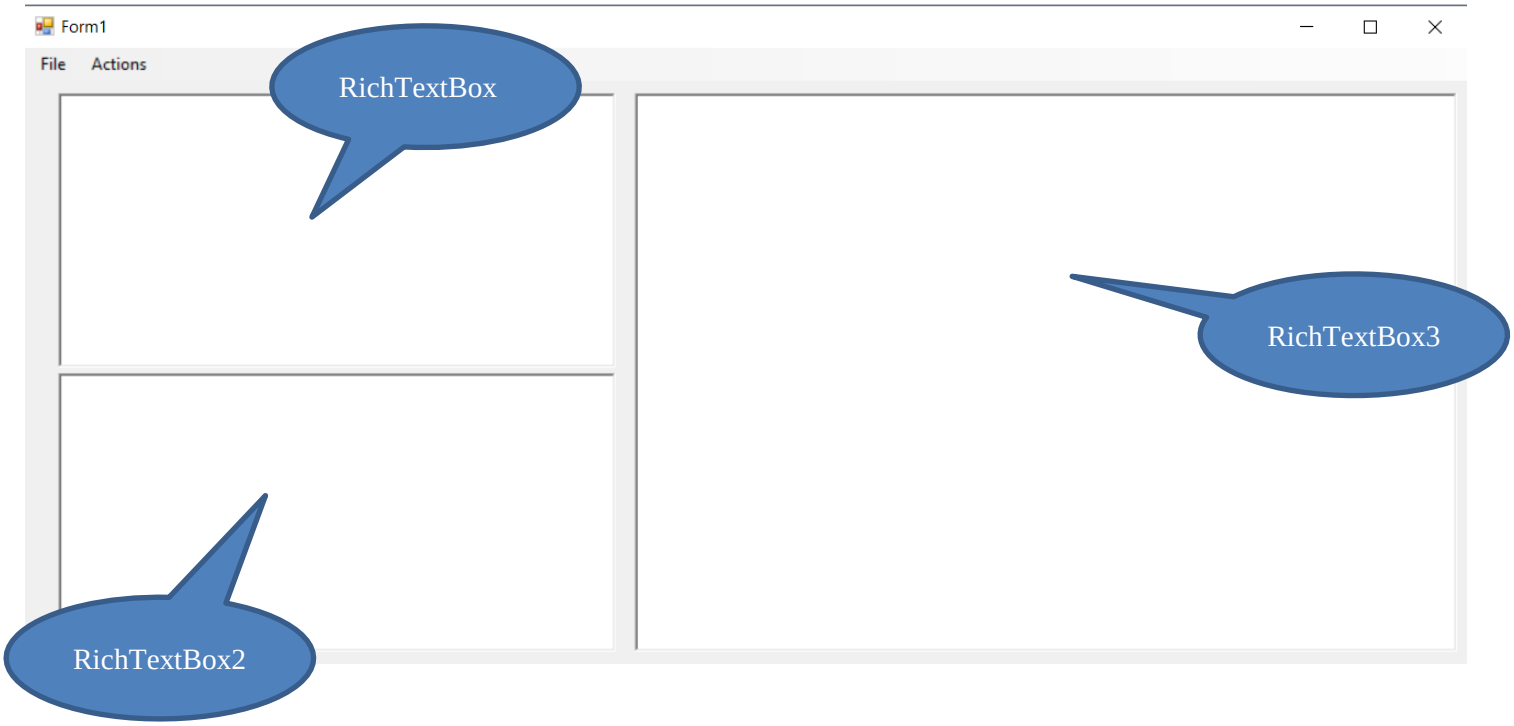
5. Bendrinės klasės (L5)

5.1. Darbo užduotis

L5_2. Autobusai. Turime duomenis apie autobusų maršrutus. Keleiviai perka bilietus. Suformuokite maršrutų, kuriais važiuos bent vienas keleivis, sąrašą. Raskite pelningiausią maršrutą. Duomenys:

- Tekstiniame faile U2a.txt yra duomenys apie autobusų maršrutus: maršruto numeris, savaitės diena, išvykimo laikas, bilieto kaina.
- Tekstiniame faile U2b.txt yra duomenys apie keleivių nupirktus bilietus: pavardė, vardas, savaitės diena, išvykimo laikas, maršruto numeris.

5.2. Grafinės vartotojo sąsajos schema ir paveikslas



5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Results(RichTextBox)	Font	Courier New, Bold, 9, script Baltic
Results2(RichTextBox2)	Font	Courier New, Bold, 9, script Baltic
Results3(RichTextBox3)	Font	Courier New, Bold, 9, script Baltic

5.4. Programos vartotojo vadovas

- Duomenys apie maršrutus ir bilietus yra surašyti tekstiniuose failuose „U2a.txt“ ir „U2b.txt“. Duomenys apie maršrutus: *maršruto numeris, savaitės diena, išvykimo laikas, bilieto kaina.* Duomenys apie bilietus: *pavardė, vardas, savaitės diena, išvykimo laikas, maršruto numeris.*
- Programa veikia meniu principu. Programa paleidžiama paspaudus mygtuką „File -> Enter“ ir pasirinkus du failus, tuomet atspausdinami pradiniai duomenys.
- Paspaudus mygtuką „Actions -> Run“ atspausdinama bilietų ir maršrutų duomenys lentelėmis. Sudaromas bei atspausdinamas maršrutų, kuriais važiavo bent vienas keleivis sąrašas.
- Paspaudus mygtuką „Actions -> Form“ Sudaromas bei atspausdinamas maršrutų, kuriais važiavo bent vienas keleivis sąrašas.
- Paspaudus mygtuką „Actions -> Find“ Surandamas ir atspausdinamas pelningiausias maršrutas.

5.5. Programos tekstas

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

namespace LAB5_sem2
{
    /// <summary>
    /// General knot class
    /// </summary>
    /// <typeparam name="Type"></typeparam>
    public sealed class Knot<Type>
    {
        public Type data { get; set; }
        public Knot<Type> Next { get; set; }


        /// <summary>
        /// constructor
        /// </summary>
        public Knot() { }


        /// <summary>
        /// constructor
        /// </summary>
        /// <param name="data"></param>
        /// <param name="address"></param>
        public Knot(Type data, Knot<Type> address)
        {
            this.data = data;
            Next = address;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LAB5_sem2
{
    /// <summary>
    /// Data class
    /// </summary>
    public class Route
    {
        public string RouteNumber { get; set; }
        public string Day { get; set; }
```

```
public DateTime DepartingTime { get; set; }
public int Price { get; set; }


/// <summary>
/// Constructor
/// </summary>
/// <param name="Rnmbr">Route number</param>
/// <param name="Day">Day</param>
/// <param name="DTime">Departing time</param>
/// <param name="Price">Price</param>
public Route(string Rnmbr, string Day, DateTime DTime, int Price)
{
    RouteNumber = Rnmbr;
    this.Day = Day;
    DepartingTime = DTime;
    this.Price = Price;
}


/// <summary>
/// Constructor
/// </summary>
public Route() { }


/// <summary>
/// new names
/// </summary>
/// <param name="a"></param>
/// <param name="b"></param>
/// <param name="c"></param>
/// <param name="d"></param>
void Add(string a, string b, int c, DateTime d)
{
    RouteNumber = b;
    Day = a;
    Price = c;
    DepartingTime = d;
}

/// <summary>
/// Overriden Object class method
/// </summary>
/// <returns>printing format for routes</returns>
public override string ToString()
{
    string line;
    line = string.Format("|  {0,-11}| {1,-14}| {2,-13}| {3,-8}|", RouteNumber, Day,
       DepartingTime.ToString("hh:mm:ss"), Price);
    return line;
}


/// <summary>
/// Overriden Object class method
/// </summary>
/// <param name="obj"></param>
/// <returns></returns>
public override bool Equals(object obj)
{
    Route rt = obj as Route;
    return rt.RouteNumber == RouteNumber;
}
```

```
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns></returns>
        public override int GetHashCode()
        {
            return base.GetHashCode();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LAB5_sem2
{
    /// <summary>
    /// Data class
    /// </summary>
    public class Tickets
    {
        public string RouteNumber { get; set; }
        public string Day { get; set; }
        public DateTime DepartingTime { get; set; }
        public string NamSur { get; set; }


        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="NamSur">Name and surname</param>
        /// <param name="Day">Day</param>
        /// <param name="DTime">Departing time</param>
        /// <param name="Rnmbr">Route number</param>
        public Tickets(string NamSur, string Day, DateTime DTime, string Rnmbr)
        {
            RouteNumber = Rnmbr;
            this.Day = Day;
            DepartingTime = DTime;
            this.NamSur = NamSur;
        }


        /// <summary>
        /// new names
        /// </summary>
        /// <param name="a"></param>
        /// <param name="b"></param>
        /// <param name="c"></param>
        /// <param name="d"></param>
        void Add(string a, string b, string c, DateTime d)
        {
            RouteNumber = b;
            Day = a;
            NamSur = c;
            DepartingTime = d;
        }
```

```
        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns>printing format for tickets</returns>
        public override string ToString()
        {
            string line;
            line = string.Format("|  {0,-18}| {1,-14}| {2,-13}| {3,-13}|",NamSur, Day,
       DepartingTime.ToString("hh:mm:ss"),RouteNumber);
            return line;
        }


        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            Route rt = obj as Route;
            return rt.RouteNumber == RouteNumber;
        }


        /// <summary>
        /// Overriden Object class method
        /// </summary>
        /// <returns></returns>
        public override int GetHashCode()
        {
            return base.GetHashCode();
        }
    }
}
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LAB5_sem2
{
    /// <summary>
    /// General linked list class
    /// </summary>
    /// <typeparam name="Type"></typeparam>
    public sealed class Routes<Type>
    {
        Knot<Type> pr;
        Knot<Type> d;
        Knot<Type> pb;

        /// <summary>
        /// constructor
        /// </summary>
        public Routes()
        {
            pr = null;
            d = null;
            pb = null;
        }
```

```
        /// <summary>
        /// returns start
        /// </summary>
        public void Start()
        {
            d = pr;
        }

        /// <summary>
        /// returns next
        /// </summary>
        public void Next()
        {
            d = d.Next;
        }

        /// <summary>
        /// returns link not equal to null
        /// </summary>
        /// <returns></returns>
        public bool Is()
        {
            return d != null;
        }

        /// <summary>
        /// returns data
        /// </summary>
        /// <returns></returns>
        public Type GetData() { return d.data; }

        /// <summary>
        /// Adds data in reverse order
        /// </summary>
        /// <param name="inf"></param>
        public void AddDataA(Type inf)
        {
            var d = new Knot<Type>(inf, null);
            d.Next = pr;
            pr = d;
        }

        /// <summary>
        /// Adds data in direct order
        /// </summary>
        /// <param name="inf"></param>
        public void AddDataT(Type inf)
        {
            var d = new Knot<Type>(inf, null);
            if (pr != null)
            {
                pb.Next = d;
                pb = d;
            }
            else
            {
                pr = d;
                pb = d;
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LAB5_sem2
{
    public partial class Form1 : Form
    {
        const string Cfd = "..\\..\\U2a.txt";
        const string Cff = "..\\..\\U2b.txt";
        const string Cfr = "..\\..\\Results.txt";

        Routes<Route> A;   //Routes List
        Routes<Tickets> B; //Tickets List
        Routes<Route> C;   //No Reapeatings List
        Routes<Route> D;   //Used Routes List

        public Form1()
        {
            InitializeComponent();

            if (File.Exists(Cfr))
                File.Delete(Cfr);

            runToolStripMenuItem.Enabled = false;
            formToolStripMenuItem.Enabled = false;
            findToolStripMenuItem.Enabled = false;
        }


        /// <summary>
        /// Reads data from tickets file
        /// </summary>
        /// <param name="fn">file name</param>
        /// <returns></returns>
        static Routes<Tickets> ReadT(string fn)
        {
            var a = new Routes<Tickets>();
            using(var reader = new StreamReader(fn))
            {
                string line;
                while((line=reader.ReadLine())!= null)
                {
                    string[] parts = line.Split(';');
                    string NamSur = parts[0];
                    string day = parts[1];
                    DateTime Dtime = DateTime.Parse(parts[2]);
                    string RNmbr = parts[3];
                    Tickets Tck = new Tickets(NamSur, day, Dtime, RNmbr);
                    a.AddDataA(Tck);
                }
            }return a;
        }
```

```
/// <summary>
/// Reads data from routes file
/// </summary>
/// <param name="fn">file name</param>
/// <returns></returns>
static Routes<Route> ReadR(string fn)
{
    var a = new Routes<Route>();
    using (var reader = new StreamReader(fn))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            string[] parts = line.Split(';');
            int Price = int.Parse(parts[3]);
            string day = parts[1];
            DateTime Dtime = DateTime.Parse(parts[2]);
            string RNmbr = parts[0];
            Route rt = new Route(RNmbr, day, Dtime, Price);
            a.AddDataA(rt);
        }
    }
    return a;
}


/// <summary>
/// Prints tickets data
/// </summary>
/// <param name="fn">file name</param>
/// <param name="B">tickets list</param>
/// <param name="heading">heading</param>
static void PrintT(string fn, Routes<Tickets> B, string heading)
{
    const string top =
          "-------------------------------------------------------------------------
      \r\n"
          + "Nr. |  Name and Surname  |      Day      |Departing Time| Route Number
      |\r\n"
          + "-------------------------------------------------------------------------
      ";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n" + heading);
        fr.WriteLine(top);
        int i = 0;
        for (B.Start(); B.Is(); B.Next())
        {
            i++;
            Tickets plr = B.GetData();
            fr.WriteLine("{0,3} {1}", i, plr);
        }
        fr.WriteLine("-------------------------------------------------------------------
      ------\n");
    }
}
```

```
/// <summary>
/// Prints routes data
/// </summary>
/// <param name="fn">file name</param>
/// <param name="B">routes list</param>
/// <param name="heading">heading</param>
static void PrintR(string fn, Routes<Route> B, string heading)
{
    const string top = "------------------------------------------------------------
      \r\n"
                     + "Nr. | Route Number|      Day      |Departing Time|  Price
      |\r\n"
           + "------------------------------------------------------------";
    using (var fr = new StreamWriter(File.Open(fn, FileMode.Append)))
    {
        fr.WriteLine("\n" + heading);
        fr.WriteLine(top);
        int i = 0;
        for (B.Start(); B.Is(); B.Next())
        {
            i++;
            Route plr = B.GetData();
            fr.WriteLine("{0,3} {1}", i, plr);
        }
        fr.WriteLine("------------------------------------------------------------\n");
    }
}


/// <summary>
/// Forms a list of used routes
/// </summary>
/// <param name="RT">Routes list</param>
/// <param name="TC">Tickets List</param>
/// <returns>Fromed list</returns>
static Routes<Route> Form(Routes<Route> RT, Routes<Tickets> TC)
{
    var A = new Routes<Route>();
    for(RT.Start(); RT.Is(); RT.Next())
    {
        Route tr = RT.GetData();
        for(TC.Start(); TC.Is(); TC.Next())
        {
            if(RT.GetData().RouteNumber == TC.GetData().RouteNumber &&
                RT.GetData().DepartingTime == TC.GetData().DepartingTime &&
RT.GetData().Day == TC.GetData().Day)
            {
                A.AddDataT(tr);
            }
        }
    }return A;
}

/// <summary>
/// Forms a list without reapeating routes
/// </summary>
/// <param name="RT">Routes list</param>
/// <param name="TC">Tickets list</param>
/// <returns>Formed list</returns>
static Routes<Route> NoReapeatings(Routes<Route> RT, Routes<Tickets> TC)
{
    Routes<Route> C = new Routes<Route>();
    Routes<Route> A = Form(RT, TC);
    for (A.Start(); A.Is(); A.Next())
    {
        int p = 0;
        Route tr = A.GetData();
        for (C.Start(); C.Is(); C.Next())
```

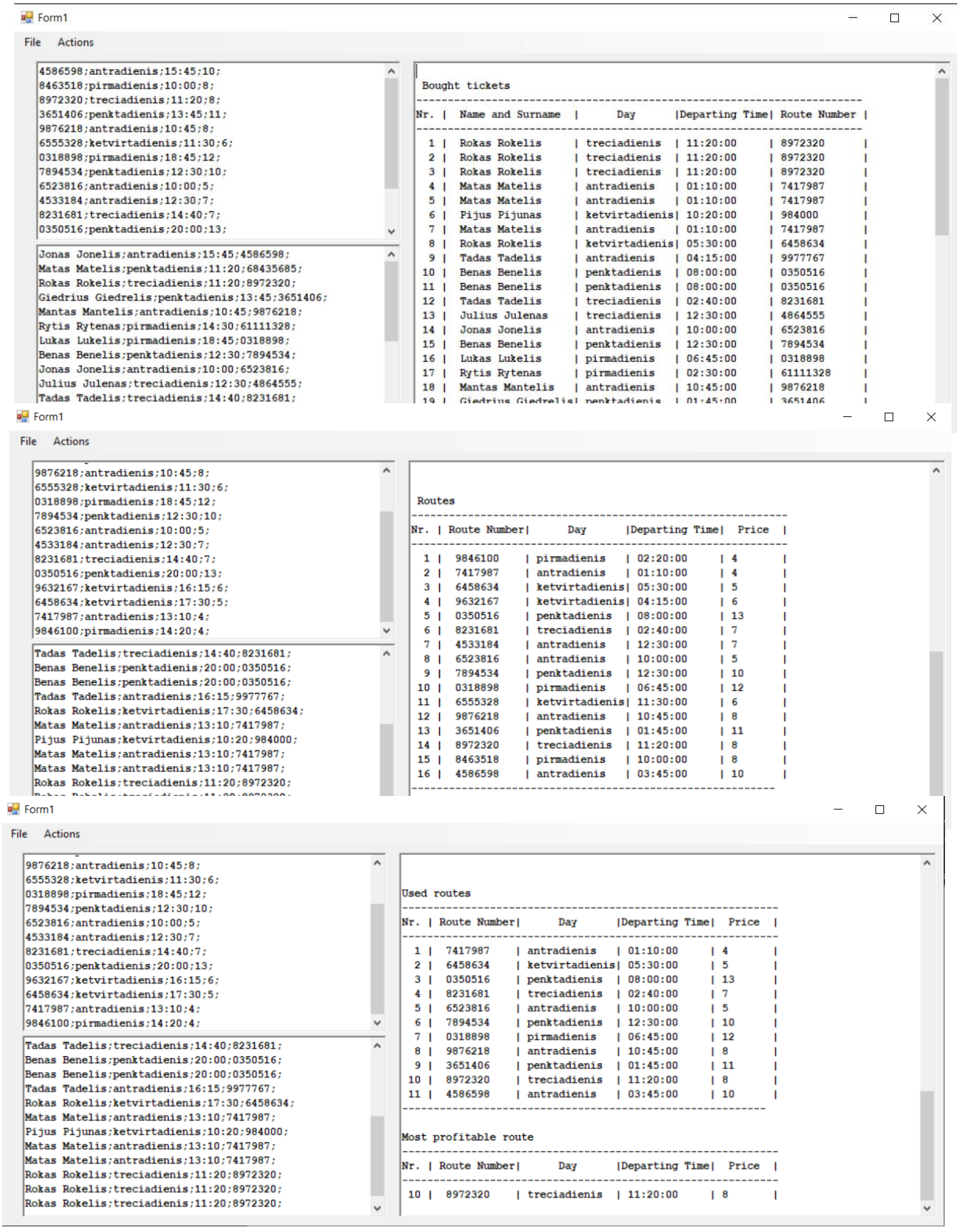
```
        {
            if (tr.RouteNumber == C.GetData().RouteNumber &&
                tr.DepartingTime == C.GetData().DepartingTime && tr.Day ==
C.GetData().Day)
            {
                p = 1;
            }
        }
        if (p == 0)
        {
            C.AddDataT(tr);
        }
    }
    return C;
}


/// <summary>
/// Finds sum of one route
/// </summary>
/// <param name="RT">Routes list</param>
/// <param name="A">Route</param>
/// <returns>sum of every route</returns>
static int Sum(Routes<Route> RT, Route A)
{

    int sum = 0;
    for (RT.Start(); RT.Is(); RT.Next())
    {

        if (RT.GetData().Day == A.Day && RT.GetData().DepartingTime == A.DepartingTime
&& RT.GetData().RouteNumber == A.RouteNumber)
        {
            sum += RT.GetData().Price;

        }
    }return sum;
}


/// <summary>
/// Finds which route was most profitable
/// </summary>
/// <param name="C">List without reapeatings</param>
/// <param name="A">List of used routes</param>
/// <param name="At">most profitable route</param>
/// <param name="ind">place in list</param>
static void Find(Routes<Route> C, Routes<Route> A, out Route At, out int ind)
{
    At = new Route();
    int max = 0; int ind1 = 0;
    ind = 0;
    using (var fr = new StreamWriter(File.Open(Cfr, FileMode.Append)))
    {
        for (C.Start(); C.Is(); C.Next())
        {
            ind1++;
            if (Sum(A, C.GetData()) > max)
            {
                max = Sum(A, C.GetData());
                At = C.GetData();
                ind = ind1;
            }
        }
    }
}
```

```
/// <summary>
/// actions of close click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}


/// <summary>
/// actions of enter click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void enterToolStripMenuItem_Click(object sender, EventArgs e)
{
    runToolStripMenuItem.Enabled = true;
    formToolStripMenuItem.Enabled = false;
    findToolStripMenuItem.Enabled = false;
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    openFileDialog1.Title = "Choose a file";
    DialogResult result = openFileDialog1.ShowDialog();
    if (result == DialogResult.OK)
    {
        string fv = openFileDialog1.FileName;
        A = ReadR(fv);
        results.LoadFile(fv, RichTextBoxStreamType.PlainText);
    }
    OpenFileDialog openFileDialog2 = new OpenFileDialog();
    openFileDialog2.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    openFileDialog2.Title = "Choose a file";
    DialogResult result1 = openFileDialog1.ShowDialog();
    if (result1 == DialogResult.OK)
    {
        string fv1 = openFileDialog1.FileName;
        B = ReadT(fv1);
        results2.LoadFile(fv1, RichTextBoxStreamType.PlainText);

    }
    PrintT(Cfr, B," Bought tickets");
    PrintR(Cfr, A," Routes");
}


/// <summary>
/// actions of run click
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void runToolStripMenuItem_Click(object sender, EventArgs e)
{
    runToolStripMenuItem.Enabled = false;
    formToolStripMenuItem.Enabled = true;
    findToolStripMenuItem.Enabled = false;
    results3.LoadFile(Cfr, RichTextBoxStreamType.PlainText);
}
```

```
        /// <summary>
        /// actions of form click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void formToolStripMenuItem_Click(object sender, EventArgs e)
        {
            runToolStripMenuItem.Enabled = false;
            formToolStripMenuItem.Enabled = false;
            findToolStripMenuItem.Enabled = true;
            D = Form(A, B);
            C = NoReapeatings(A, B);
            //PrintR(Cfr, D, "Used");
            C.Start();
            if (C.Is())
            {
                PrintR(Cfr, C, "Used routes");
            }
            else
            {
                using (var fr = File.AppendText(Cfr))
                    fr.WriteLine("There are no used routes");
            }
            results3.LoadFile(Cfr, RichTextBoxStreamType.PlainText);
        }


        /// <summary>
        /// actions of find click
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void findToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int ind = 0;
            Route At = new Route();
            Find(C, D, out At, out ind);
            if (At.RouteNumber != "" && At.Price != 0)
            {
                using(var fr = File.AppendText(Cfr))
                {
                    fr.WriteLine("Most profitable route");
                    fr.WriteLine("-------------------------------------------------------------
                     \r\n"
                        +        "Nr. | Route Number|      Day      |Departing Time|  Price
          |\r\n"
                        +        "-------------------------------------------------------------
          ");
                    fr.WriteLine("{0,3} {1}" , ind , At);
                }
            }
            else
            {
                using (var fr = File.AppendText(Cfr))
                    fr.WriteLine("There is no profitable route");
            }

            results3.LoadFile(Cfr, RichTextBoxStreamType.PlainText);
        }
    }
}
```

5.6. Pradiniai duomenys ir rezultatai

Testas nr.1



Bought tickets

Nr.	Name and Surname	Day	Departing Time	Route Number
1	Rokas Rokelis	treciadienis	11:20:00	8972320
2	Rokas Rokelis	treciadienis	11:20:00	8972320
3	Rokas Rokelis	treciadienis	11:20:00	8972320
4	Matas Matelis	antradienis	01:10:00	7417987
5	Matas Matelis	antradienis	01:10:00	7417987
6	Pijus Pijunas	ketvirtadienis	10:20:00	984000
7	Matas Matelis	antradienis	01:10:00	7417987
8	Rokas Rokelis	ketvirtadienis	05:30:00	6458634
9	Tadas Tadelis	antradienis	04:15:00	9977767
10	Benas Benelis	penktadienis	08:00:00	0350516
11	Benas Benelis	penktadienis	08:00:00	0350516
12	Tadas Tadelis	treciadienis	02:40:00	8231681
13	Julius Julenas	treciadienis	12:30:00	4864555
14	Jonas Jonelis	antradienis	10:00:00	6523816
15	Benas Benelis	penktadienis	12:30:00	7894534
16	Lukas Lukelis	pirmadienis	06:45:00	0318898
17	Rytis Rytenas	pirmadienis	02:30:00	61111328
18	Mantas Mantelis	antradienis	10:45:00	9876218
19	Giedrius Giedrelis	penktadienis	01:45:00	3651406
20	Rokas Rokelis	treciadienis	11:20:00	8972320
21	Matas Matelis	penktadienis	11:20:00	68435685
22	Jonas Jonelis	antradienis	03:45:00	4586598

Routes

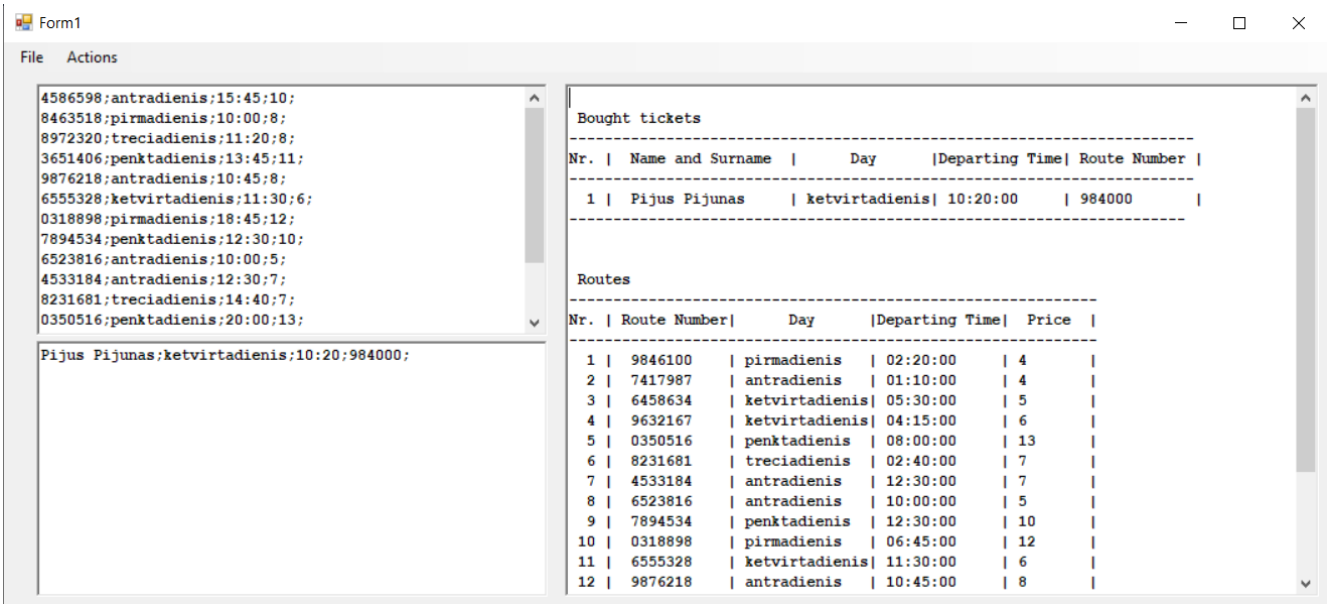
Nr.	Route Number	Day	Departing Time	Price
1	9846100	pirmadienis	02:20:00	4
2	7417987	antradienis	01:10:00	4
3	6458634	ketvirtadienis	05:30:00	5
4	9632167	ketvirtadienis	04:15:00	6
5	0350516	penktadienis	08:00:00	13
6	8231681	treciadienis	02:40:00	7
7	4533184	antradienis	12:30:00	7
8	6523816	antradienis	10:00:00	5
9	7894534	penktadienis	12:30:00	10
10	0318898	pirmadienis	06:45:00	12
11	6555328	ketvirtadienis	11:30:00	6
12	9876218	antradienis	10:45:00	8
13	3651406	penktadienis	01:45:00	11
14	8972320	treciadienis	11:20:00	8
15	8463518	pirmadienis	10:00:00	8
16	4586598	antradienis	03:45:00	10

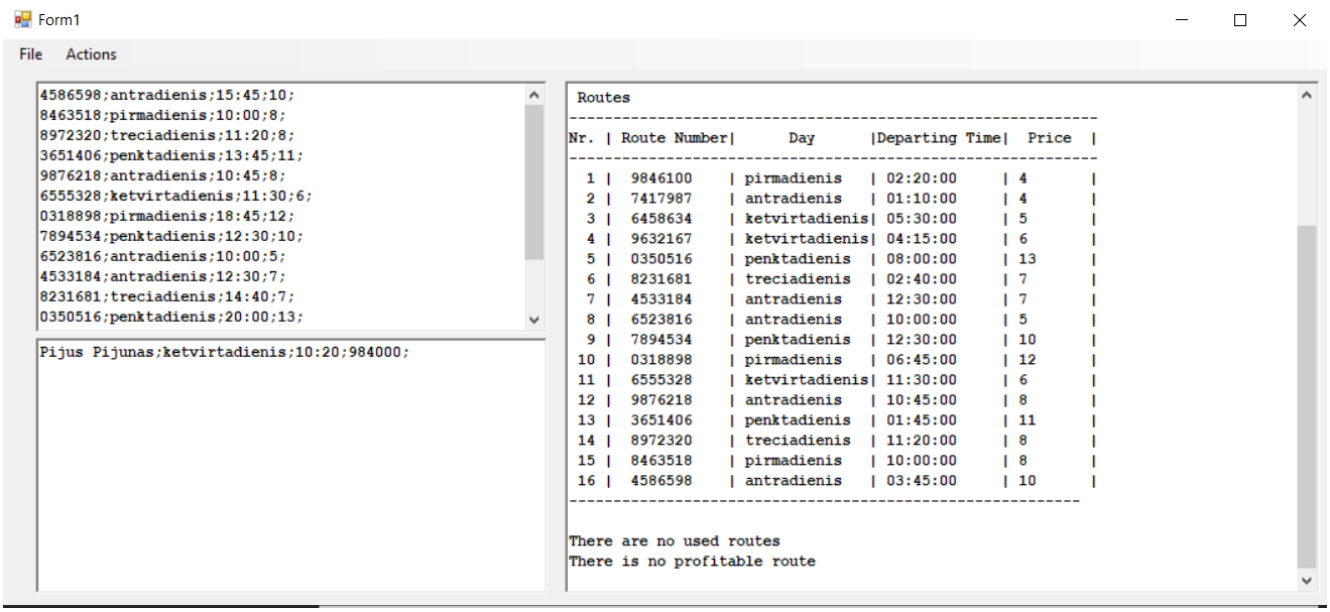
Used routes

Nr.	Route Number	Day	Departing Time	Price
1	7417987	antradienis	01:10:00	4
2	6458634	ketvirtadienis	05:30:00	5
3	0350516	penktadienis	08:00:00	13
4	8231681	treciadienis	02:40:00	7
5	6523816	antradienis	10:00:00	5
6	7894534	penktadienis	12:30:00	10
7	0318898	pirmadienis	06:45:00	12
8	9876218	antradienis	10:45:00	8
9	3651406	penktadienis	01:45:00	11

```
 10 |  8972320    | treciadienis  | 11:20:00     | 8       |
 11 |  4586598    | antradienis   | 03:45:00     | 10      |
-------------------------------------------------------------

Most profitable route
-------------------------------------------------------------
Nr. | Route Number|      Day      |Departing Time|  Price  |
-------------------------------------------------------------
 10 |  8972320    | treciadienis  | 11:20:00     | 8       |
```

Testas nr.2





Results.txt

```
Bought tickets
-------------------------------------------------------------------------
Nr. |  Name and Surname  |      Day      |Departing Time| Route Number |
-------------------------------------------------------------------------
  1 |  Pijus Pijunas     | ketvirtadienis| 10:20:00     | 984000       |
-------------------------------------------------------------------------
```

```
 Routes
-------------------------------------------------------------
Nr. | Route Number|      Day      |Departing Time|  Price  |
-------------------------------------------------------------
  1 |  9846100    | pirmadienis   | 02:20:00     | 4       |
  2 |  7417987    | antradienis   | 01:10:00     | 4       |
  3 |  6458634    | ketvirtadienis| 05:30:00     | 5       |
  4 |  9632167    | ketvirtadienis| 04:15:00     | 6       |
  5 |  0350516    | penktadienis  | 08:00:00     | 13      |
  6 |  8231681    | treciadienis  | 02:40:00     | 7       |
  7 |  4533184    | antradienis   | 12:30:00     | 7       |
  8 |  6523816    | antradienis   | 10:00:00     | 5       |
  9 |  7894534    | penktadienis  | 12:30:00     | 10      |
 10 |  0318898    | pirmadienis   | 06:45:00     | 12      |
 11 |  6555328    | ketvirtadienis| 11:30:00     | 6       |
 12 |  9876218    | antradienis   | 10:45:00     | 8       |
 13 |  3651406    | penktadienis  | 01:45:00     | 11      |
 14 |  8972320    | treciadienis  | 11:20:00     | 8       |
 15 |  8463518    | pirmadienis   | 10:00:00     | 8       |
 16 |  4586598    | antradienis   | 03:45:00     | 10      |
-------------------------------------------------------------

There are no used routes
There is no profitable route
```

5.7. Dėstytojo pastabos