KAUNO TECHNOLOGIJOS UNIVERSITETO GIMNAZIJA

Objektinio programavimo pagrindai II (P175B502)

Laboratorinių darbų ataskaitos

Atliko:

III NFQ klasės mokinys

Augustas Mačijauskas

2018 m. gegužės 31 d.

Priėmė:

Kęstutis Simonavičius, Jūratė Pauliutė

KAUNAS 2018

TURINYS

1. Grafinė vartotojo sąsaja ir algoritmų taikymas (L1) 3

1.1. Darbo užduotis 3

1.2. Grafinės vartotojo sąsajos schema ir paveikslas 3

1.3. Sąsajoje panaudotų komponentų keičiamos savybės 4

1.4. Programos vartotojo vadovas 4

1.5. Programos tekstas 5

1.6. Pradiniai duomenys ir rezultatai 11

2. Dinaminis masyvas (L2) 14

2.1. Darbo užduotis 14

2.2. Grafinės vartotojo sąsajos schema ir paveikslas 14

2.3. Sąsajoje panaudotų komponentų keičiamos savybės 16

2.4. Programos vartotojo vadovas 17

2.5. Programos tekstas 18

2.6. Pradiniai duomenys ir rezultatai 26

3. Paveldėjimas (L3) 30

3.1. Darbo užduotis 30

3.2. Grafinės vartotojo sąsajos schema ir paveikslas 30

3.3. Sąsajoje panaudotų komponentų keičiamos savybės 33

3.4. Programos vartotojo vadovas 34

3.5. Programos tekstas 34

3.6. Pradiniai duomenys ir rezultatai 40

4. Susietasis sąrašas (L4) 44

4.1. Darbo užduotis 44

4.2. Grafinės vartotojo sąsajos schema ir paveikslas 44

4.3. Sąsajoje panaudotų komponentų keičiamos savybės 44

4.4. Programos vartotojo vadovas 44

4.5. Programos tekstas 44

4.6. Pradiniai duomenys ir rezultatai 44

5. Bendrinės klasės (L5) 45

5.1. Darbo užduotis 45

5.2. Grafinės vartotojo sąsajos schema ir paveikslas 45

5.3. Sąsajoje panaudotų komponentų keičiamos savybės 45

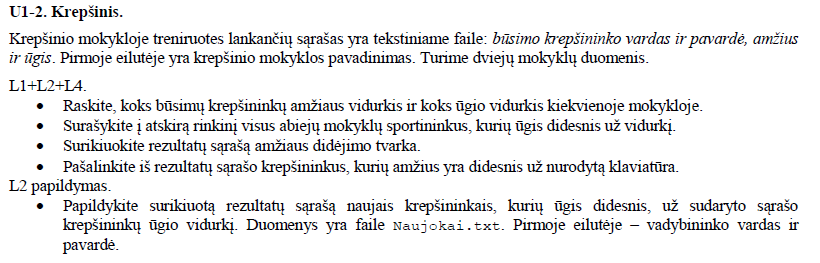
5.4. Programos vartotojo vadovas 45

5.5. Programos tekstas 45

5.6. Pradiniai duomenys ir rezultatai 45

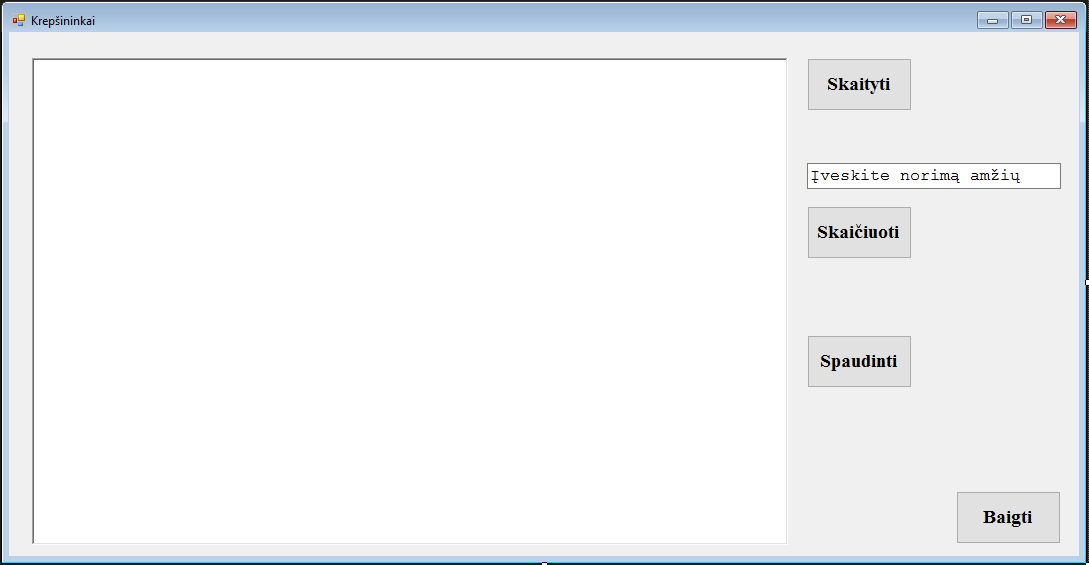
# Grafinė vartotojo sąsaja ir algoritmų taikymas (L1)

## Darbo užduotis



## Grafinės vartotojo sąsajos schema ir paveikslas

Schema:



button1

richTextBox1

Form1

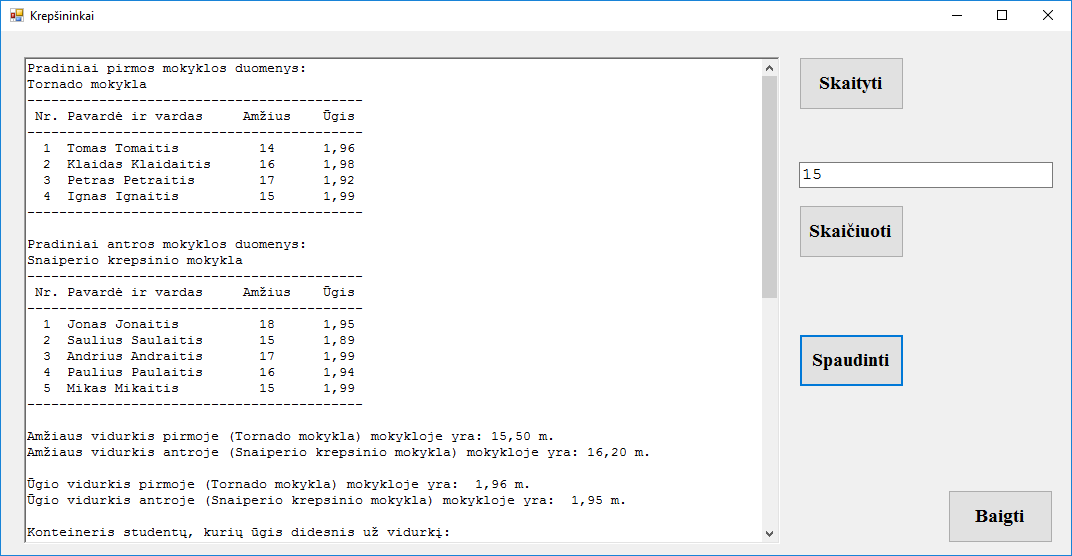
textBox1

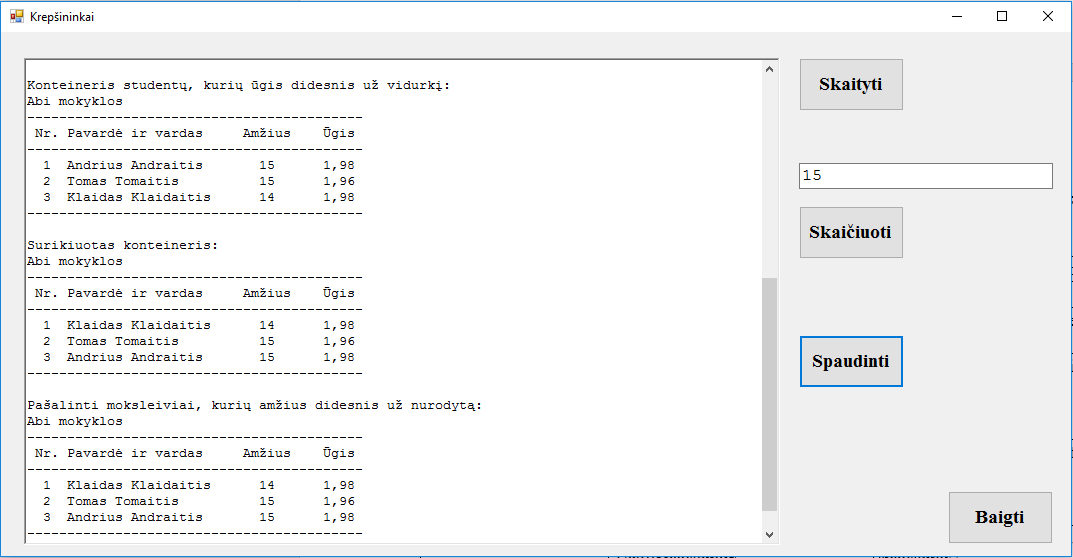
button4

button3

button2

Paveikslas:





## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Form1 | Text: Krepšininkai | Programos langas |
| richTextBox1 (tipas RichTextBox) | Font: Courier New, Regular, 10;  Script: Baltic; | Atvaizduoti pradinių duomenų ir skaičiavimų turiniui |
| Button1 (tipas Button) | Name: read  Text: Skaityti  Font: Times New Roman, Bold, 14;  Script: Baltic; | Mygtukas duomenims iš failo nuskaityti |
| Button2 (tipas Button) | Name: skaiciuoti  Text: Skaičiuoti  Font: Times New Roman, Bold, 14;  Script: Baltic; | Mygtukas skaičiavimams atlikti |
| Button3 (tipas Button) | Name: print  Text: Spausdinti  Font: Times New Roman, Bold, 14;  Script: Baltic; | Mygtukas skaičiavimų rezultatams spausdinti |
| Button4 (tipas Button) | Name: baigti  Text: Baigti  Font: Times New Roman, Bold, 14;  Script: Baltic; | Mygtukas programos lango uždarymui |
| TextBox1 (tipas TextBox) | Text: Įveskite norimą amžių  Font: Courier New, Regular, 12;  Script: Baltic; | Teksto laukas norimam amžiui užrašyti |

## Programos vartotojo vadovas

Programos naudojimo eiga:

1. Paleidę programa, spauskite mygtuką „Skaityti“, kuris nuskaitys krepšininkų duomenis iš failo.
2. Kai duomenys bus nuskaityti, į žemiau esantį teksto lauką (su tekstu „Įveskite norimą amžių“) įveskite amžių, už kurį vyresnius krepšininkus norite pašalinti.
3. Įvedę amžių, spauskite mygtuką „Skaičiuoti“, kuris atliks visus reikiamus skaičiavimus.
4. Spauskite mygtuką „Spausdinti“, kuris atspausdins skaičiavimų rezultatus ekrane.
5. Spauskite mygtuką „Baigti“, jei norite baigti darbą su programa.

## Programos tekstas

**Klasė Krepsininkas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L1

{

class Krepsininkas

{

public string VP { get; set; }

public int Amžius { get; set; }

public double Ūgis { get; set; }

public Krepsininkas(string vrdpav, int amz, double ug)

{

VP = vrdpav;

Amžius = amz;

Ūgis = ug;

}

public override string ToString()

{

string eilute;

eilute = string.Format(" {0, -20} {1, 2} {2, 5:f}", VP, Amžius, Ūgis);

return eilute;

}

public static bool operator<=(Krepsininkas k1, Krepsininkas k2)

{

return k1.Amžius < k2.Amžius;

}

public static bool operator>=(Krepsininkas k1, Krepsininkas k2)

{

return k1.Amžius > k2.Amžius;

}

}

}

**Klasė Krepsininkai:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L1

{

/// <summary>

///

/// </summary>

class Krepsininkai

{

const int Max = 500;

private Krepsininkas[] Krep;

public int Kiek { get; set; }

public Krepsininkai()

{

Kiek = 0;

Krep = new Krepsininkas[Max];

}

/// <summary>

/// Grąžina atitinkamo indekso vietoje esantį krepšininko objektą;

/// </summary>

/// <returns>Krepsin</returns>

public Krepsininkas ImtiKrepsininka(int i)

{

return Krep[i];

}

/// <summary>

/// Papildo krepšininkų masyvą

/// </summary>

public void DetiStudenta(Krepsininkas ob)

{

if (Kiek + 1 < Max)

Krep[Kiek++] = ob;

}

public void Rikiuoti()

{

for (int i = 0; i < Kiek - 1; i++)

{

Krepsininkas pagalb = Krep[i];

int ind = i;

for (int j = i + 1; j < Kiek; j++)

{

if (Krep[j] <= pagalb)

{

pagalb = Krep[j];

ind = j;

}

}

Krep[ind] = Krep[i];

Krep[i] = pagalb;

}

}

public void Šalinti(int amz)

{

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

{

if (Krep[i].Amžius > amz)

{

for (int j = i; j < Kiek - 1; j++)

{

Krep[j] = Krep[j + 1];

}

Kiek--;

i--;

}

}

}

}

}

**Programos tekstas:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace L1

{

public partial class Form1 : Form

{

const string duom1 = "..\\..\\duom1\_2.txt";

const string duom2 = "..\\..\\duom2\_2.txt";

const string rez = "..\\..\\rez.txt";

string mokPav1;

string mokPav2;

Krepsininkai mokykla1;

Krepsininkai mokykla2;

Krepsininkai naujasKonteineris;

public Form1()

{

InitializeComponent();

print.Enabled = false;

skaiciuoti.Enabled = false;

if (File.Exists(rez))

File.Delete(rez);

}

/// <summary>

/// Atlieka duomenų nuskaitymą ir surašymą į konteinerius.

/// </summary>

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

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

private void read\_Click(object sender, EventArgs e)

{

mokykla1 = Skaityti(duom1, out mokPav1);

mokykla2 = Skaityti(duom2, out mokPav2);

Spausdinti(rez, mokykla1, mokPav1, "Pradiniai pirmos mokyklos duomenys:");

Spausdinti(rez, mokykla2, mokPav2, "Pradiniai antros mokyklos duomenys:");

naujasKonteineris = new Krepsininkai();

string x = File.ReadAllText(rez);

richTextBox1.Text = x;

skaiciuoti.Enabled = true;

}

/// <summary>

/// Atliekami reikiami skaičiavimai, jų rezultatai įvedami į duomenų failą.

/// </summary>

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

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

private void skaiciuoti\_Click(object sender, EventArgs e)

{

double amzVid1 = AmziausVidurkis(mokykla1);

SpausdintiTeksta(rez, string.Format("Amžiaus vidurkis pirmoje ({0}) mokykloje yra: {1, 5:f} m.", mokPav1, amzVid1));

double amzVid2 = AmziausVidurkis(mokykla2);

SpausdintiTeksta(rez, string.Format("Amžiaus vidurkis antroje ({0}) mokykloje yra: {1, 5:f} m.\r\n", mokPav2, amzVid2));

double ugioVid1 = ŪgioVidurkis(mokykla1);

SpausdintiTeksta(rez, string.Format("Ūgio vidurkis pirmoje ({0}) mokykloje yra: {1, 5:f} m.", mokPav1, ugioVid1));

double ugioVid2 = ŪgioVidurkis(mokykla2);

SpausdintiTeksta(rez, string.Format("Ūgio vidurkis antroje ({0}) mokykloje yra: {1, 5:f} m.\r\n", mokPav2, ugioVid2));

double vidur = Math.Round(((ugioVid1 + ugioVid2) / 2), 2);

UgisDidesnisUzVidurki(mokykla1, naujasKonteineris, vidur);

UgisDidesnisUzVidurki(mokykla2, naujasKonteineris, vidur);

Spausdinti(rez, naujasKonteineris, "Abi mokyklos", "Konteineris studentų, kurių ūgis didesnis už vidurkį:");

//mokykla1.Rikiuoti();

//Spausdinti(rez, mokykla1, mokPav1, "Surikiuotas konteineris:");

//mokykla2.Rikiuoti();

//Spausdinti(rez, mokykla2, mokPav2, "Surikiuotas konteineris:");

naujasKonteineris.Rikiuoti();

Spausdinti(rez, naujasKonteineris, "Abi mokyklos", "Surikiuotas konteineris:");

int amz = int.Parse(textBox1.Text);

//mokykla1.Šalinti(amz);

//Spausdinti(rez, mokykla1, mokPav1, "Pašalinti moksleiviai, kurių amžius didesnis už nurodytą:");

//mokykla2.Šalinti(amz);

//Spausdinti(rez, mokykla2, mokPav2, "Pašalinti moksleiviai, kurių amžius didesnis už nurodytą:");

naujasKonteineris.Šalinti(amz);

Spausdinti(rez, naujasKonteineris, "Abi mokyklos", "Pašalinti moksleiviai, kurių amžius didesnis už nurodytą:");

print.Enabled = true;

}

/// <summary>

/// Spaudina tekstą iš duomenų failo į programos langą.

/// </summary>

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

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

private void print\_Click(object sender, EventArgs e)

{

string x = File.ReadAllText(rez);

richTextBox1.Text = x;

}

/// <summary>

/// Užėjus ant teksto lauko, jį išvalo.

/// </summary>

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

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

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

textBox1.Text = "";

}

/// <summary>

/// Baigia programos veikimą.

/// </summary>

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

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

private void baigti\_Click(object sender, EventArgs e)

{

Close();

}

/// <summary>

/// Formuoja naują konteinerį iš krepšininkų, kurių ūgis didesnis už vidurkį.

/// </summary>

/// <param name="kont1">Pirmas konteineris</param>

/// <param name="kont2">Antras kont</param>

/// <param name="naujas">Trečias kont</param>

private void UgisDidesnisUzVidurki(Krepsininkai kont, Krepsininkai naujas, double ugis)

{

for (int i = 0; i < kont.Kiek; i++)

{

if (kont.ImtiKrepsininka(i).Ūgis > ugis)

{

naujas.DetiStudenta(kont.ImtiKrepsininka(i));

}

}

}

/// <summary>

/// Randa krepšininkų amžiaus vidurkį.

/// </summary>

/// <param name="kont">Konteineris</param>

/// <returns>Amžiaus vidurkis</returns>

static double AmziausVidurkis(Krepsininkai kont)

{

double suma = 0;

for (int i = 0; i < kont.Kiek; i++)

{

suma += kont.ImtiKrepsininka(i).Amžius;

}

return suma / kont.Kiek;

}

/// <summary>

/// Randa krepšininkų ūgio vidurkį.

/// </summary>

/// <param name="kont">Konteineris</param>

/// <returns>Ūgio vidurkis</returns>

static double ŪgioVidurkis(Krepsininkai kont)

{

double suma = 0;

for (int i = 0; i < kont.Kiek; i++)

{

suma += kont.ImtiKrepsininka(i).Ūgis;

}

return suma / kont.Kiek;

}

/// <summary>

/// Nuskaito duomenų failus.

/// </summary>

/// <param name="fr"></param>

/// <param name="pav"></param>

/// <returns></returns>

private Krepsininkai Skaityti(string fr, out string pav)

{

Krepsininkai konteineris = new Krepsininkai();

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

{

string line;

line = reader.ReadLine();

pav = line;

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

{

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

string pavVrd = parts[0];

int amz = int.Parse(parts[1]);

double ug = double.Parse(parts[2]);

Krepsininkas krep = new Krepsininkas(pavVrd, amz, ug);

konteineris.DetiStudenta(krep);

}

return konteineris;

}

}

/// <summary>

/// Spausdina duomenis į failą lentele.

/// </summary>

/// <param name="fw"></param>

/// <param name="kont"></param>

/// <param name="pav"></param>

/// <param name="antraste"></param>

private void Spausdinti(string fw, Krepsininkai kont, string pav, string antraste)

{

const string virsus =

"------------------------------------------\r\n" +

" Nr. Pavardė ir vardas Amžius Ūgis \r\n" +

"------------------------------------------";

using (var fr = new StreamWriter(File.Open(fw, FileMode.Append)))

{

if (kont.Kiek > 0)

{

fr.WriteLine(antraste);

fr.WriteLine(pav);

fr.WriteLine(virsus);

for (int i = 0; i < kont.Kiek; i++)

{

Krepsininkas krep = kont.ImtiKrepsininka(i);

fr.WriteLine("{0, 3} {1}", i + 1, krep);

}

fr.WriteLine("------------------------------------------\r\n");

}

else

{

fr.WriteLine("Studentų konteineris tuščias!\r\n");

}

}

}

/// <summary>

/// Spausdina tekstą į failą.

/// </summary>

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

/// <param name="x"></param>

private void SpausdintiTeksta(string rez, string x)

{

using (StreamWriter fw = new StreamWriter(File.Open(rez, FileMode.Append)))

{

fw.WriteLine(x);

}

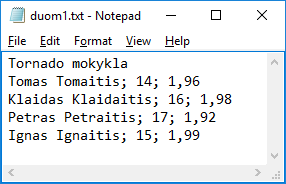
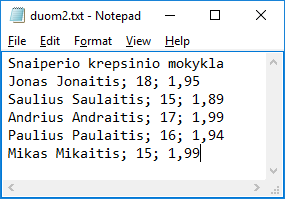
}

}

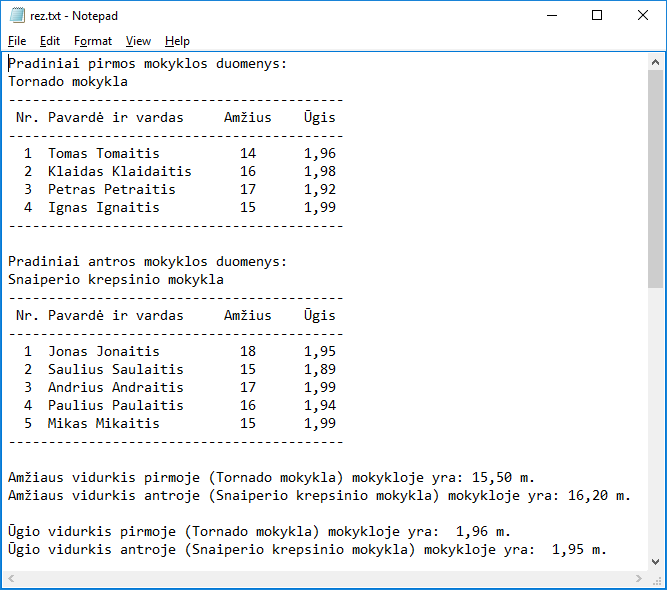
}

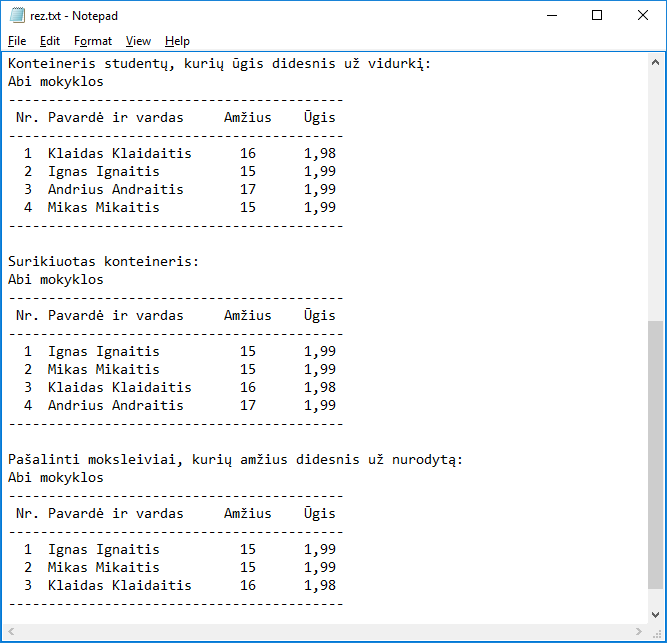
## Pradiniai duomenys ir rezultatai

Pradiniai duomenys nr. 1:

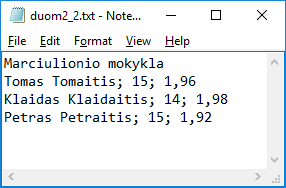
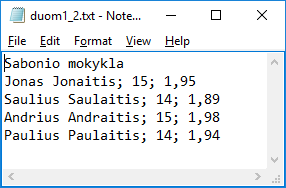
 

Rezultatai nr. 1:

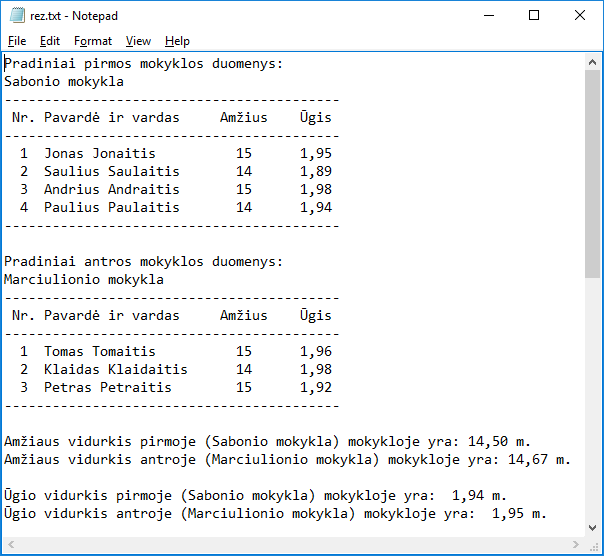


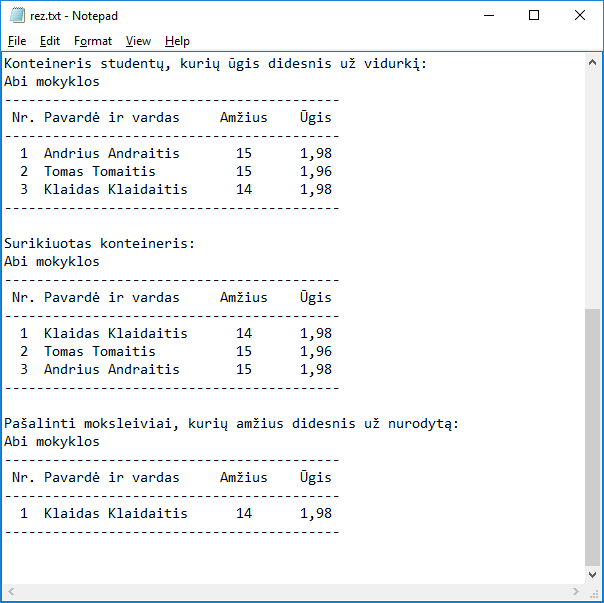


Pradiniai duomenys nr. 2:



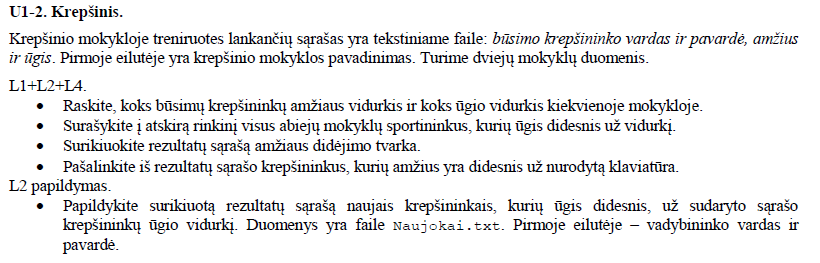
Rezultatai nr. 2:





# Dinaminis masyvas (L2)

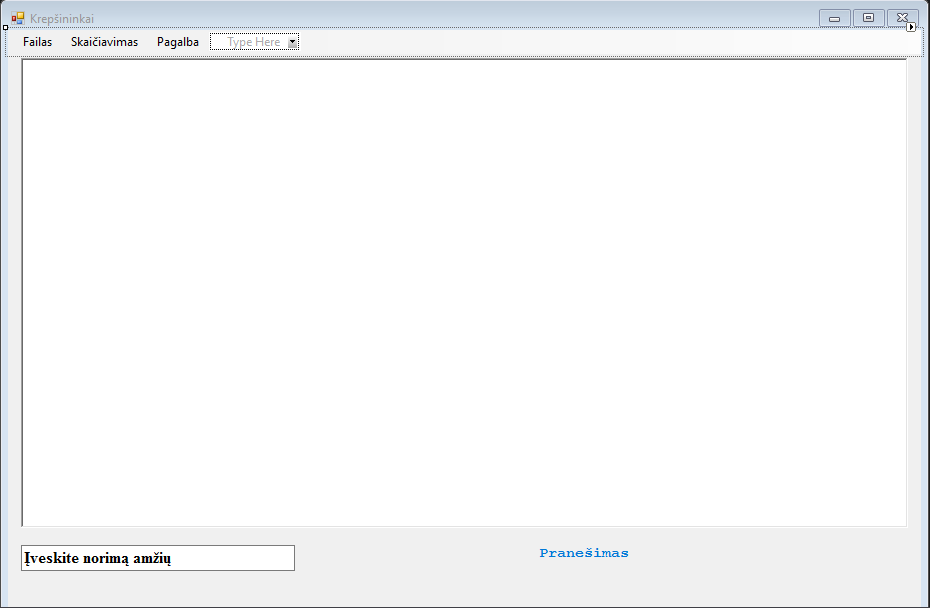
## Darbo užduotis



## Grafinės vartotojo sąsajos schema ir paveikslas

Schema:

Form1



menuStrip1

richTextBox1

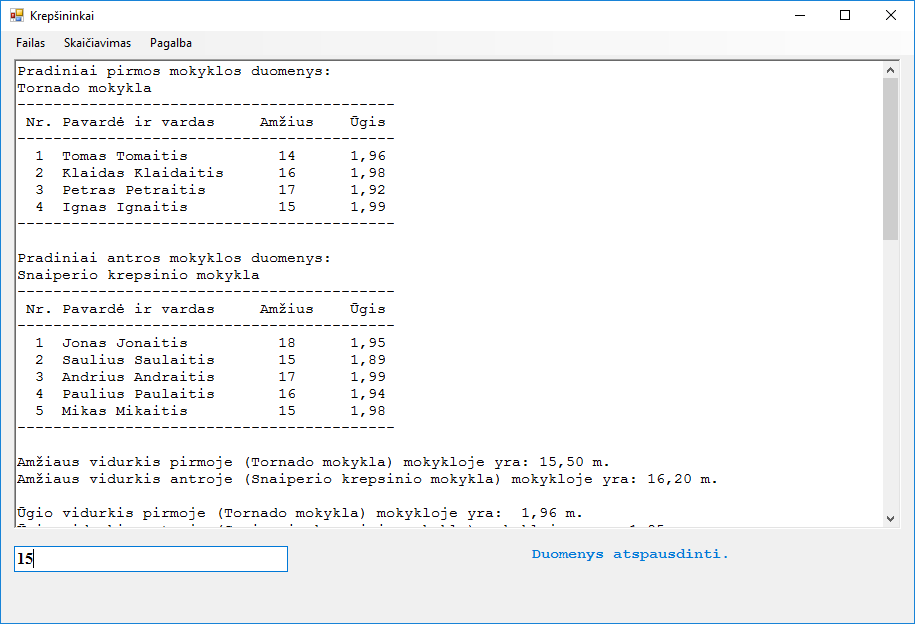
textBox1

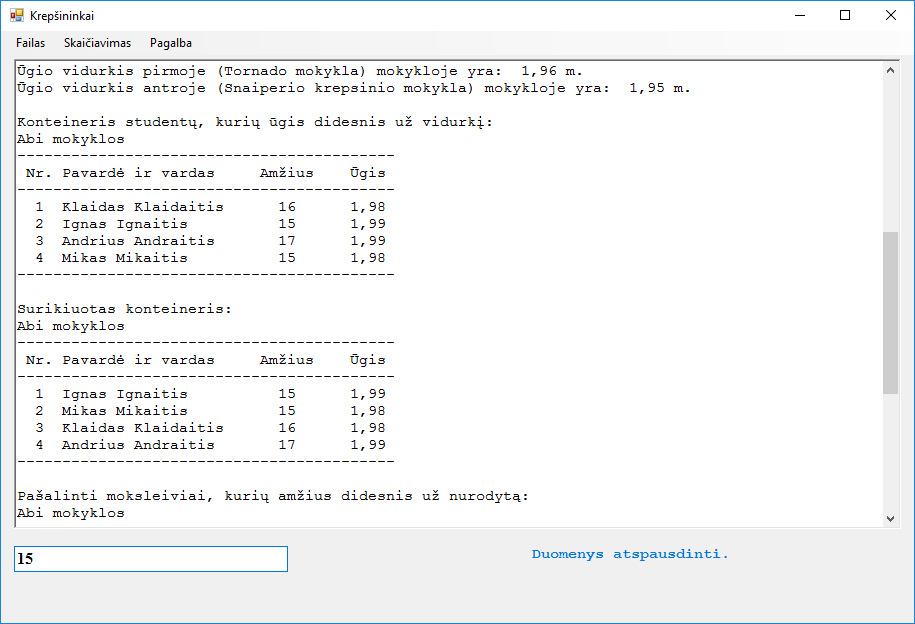
label1

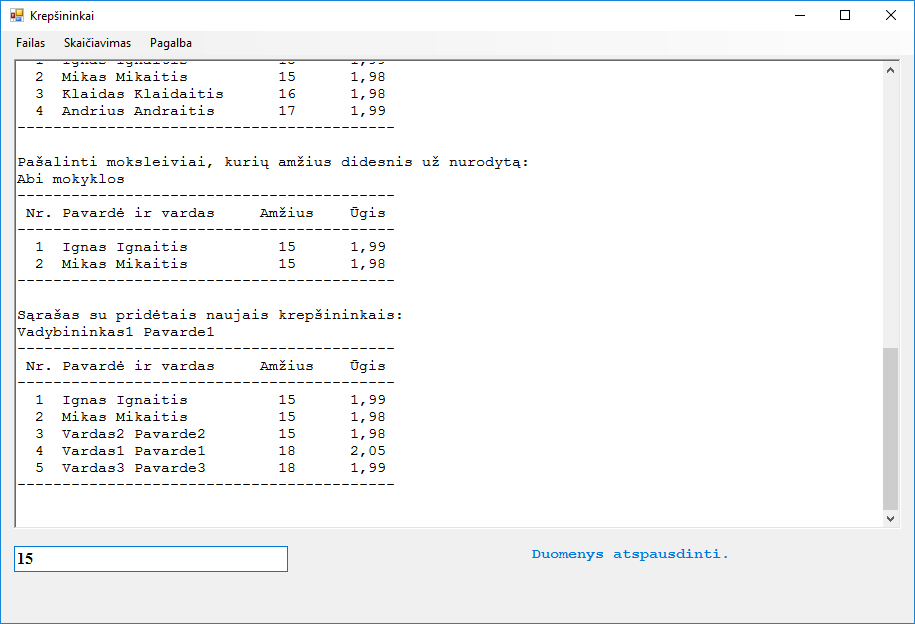
Meniu punktai:

|  |  |  |
| --- | --- | --- |
| Failas | Skaičiavimas | Pagalba |
| Įvesti | Skaičiuoti | Nurodymai vartotojui |
| Baigti | Spausdinti | Naudojimo sąlygos |
|  | Nauji žaidėjai |  |
| Išsaugoti |  |

Paveikslas:







## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Form1 | Text: Krepšininkai | Programos langas |
| richTextBox1  (tipas RichTextBox) | Font: Courier New, Regular, 11;  Script: Baltic; | Atvaizduoti pradinių duomenų ir skaičiavimų turiniui |
| textBox1  (tipas TextBox) | Text: Įveskite norimą amžių  Font: Times New Roman, Bold, 12;  Script: Baltic; | Norimo amžiaus įvedimui |
| label1  (tipas Label) | Text: Pranešimas  Font: Courier New, Bold, 11;  Script: Baltic;  ForeColor: Highlight | Spausdina pranešimus apie programos veikimą. |
| menuStrip1  (tipas MenuStrip) | Font: Standartiniai nustatymai | Naudojamas meniu punktų, kurie bus naudojami programos veikimo metu, išdėstymui. |
| failasToolStripMenuItem  (tipas ToolStripMenu) | Text: Failas  Font: Standartiniai nustatymai | Meniu punktas, kuriame yra mygtukai duomenų nuskaitymui ir programos darbo baigimui |
| ivestiToolStripMenuItem  (tipas ToolStripMenu) | Name: nuskaityti  Text: Įvesti  Font: Standartiniai nustatymai | Meniu punktas duomenims iš failo nuskaityti |
| baigtiToolStripMenuItem  (tipas ToolStripMenu) | Name: baigti  Text: Baigti  Font: Standartiniai nustatymai | Meniu punktas, baigiantis darbą su programa |
| skaičiavimasToolStripMenuItem  (tipas ToolStripMenu) | Text: Skaičiavimas  Font: Standartiniai nustatymai | Meniu punktas, kuriame yra mygtukai skaičiavimams, spausdinimui, naujų žaidėjų pridėjimui ir saugojimui .csv failą atlikti. |
| skaiciuotiToolStripMenuItem  (tipas ToolStripMenu) | Name: skaiciuoti  Text: Skaičiuoti  Font: Standartiniai nustatymai | Meniu punktas skaičiavimams atlikti |
| spausdintiToolStripMenuItem  (tipas ToolStripMenu) | Name: print  Text: Spausdinti  Font: Standartiniai nustatymai | Meniu punktas gautiems rezultatams atspausdinti |
| naujiZaidejaiToolStripMenuItem  (tipas ToolStripMenu) | Name: pridėtiNaujų  Text: Nauji žaidėjai  Font: Standartiniai nustatymai | Meniu punktas, pridedantis naujus žaidėjus iš failo |
| issaugotiToolStripMenuItem  (tipas ToolStripMenu) | Name: išsaugoti  Text: Išsaugoti  Font: Standartiniai nustatymai | Meniu punktas, leidžiantis išsaugoti rezultatus į .csv failą. |
| pagalbaToolStripMenuItem  (tipas ToolStripMenu) | Text: Pagalba  Font: Standartiniai nustatymai | Meniu punktas, turintis mygtukus, kurie atvaizduoja nurodymus vartotojui ir naudojimo sąlygas. |
| nurodymaiVartotojuiToolStripMenuItem (tipas ToolStripMenu) | Text: Nurodymai vartotojui  Font: Standartiniai nustatymai | Meniu punktas, spausdinantis nurodymus vartotojui. |
| naudojimoSalygosToolStripMenuItem (tipas ToolStripMenu) | Text: Naudojimo sąlygos  Font: Standartiniai nustatymai | Meniu punktas, spausdinantis naudojimo sąlygas. |

## Programos vartotojo vadovas

1. Atsidarę programą, spauskite Failas - > Įvesti. Pasirinkite failus, iš kurių norite nuskaityti duomenis.
2. Į teksto lauką (pavadinimu „Įveskite norimą amžių“) įveskite amžių, už kurį vyresnius krepšininkus norite pašalinti.
3. Spauskite Skaičiavimas -> Skaičiuoti. Bus atlikti skaičiavimai su pradiniais duomenimis.
4. Spauskite Skaičiavimas -> Spausdinti. Ekrane bus atvaizduoti skaičiavimų rezultatai.
5. Spauskite Skaičiavimas -> Nauji žaidėjai. Prie gautų duomenų bus pridėti nauji žaidėjai, išlaikant rikiavimo tvarką. Ekrane nesimatys jokių pakitimų.
6. Vėl spauskite Skaičiavimas -> Spausdinti. Masyvas su naujais žaidėjais bus atspausdintas ekrane.
7. Spauskite Skaičiavimas -> Išsaugoti. Pasirinkite arba sukurkite naują .csv failą ir rezultatų masyvas bus išsaugotas jame.
8. Jei norite gauti nurodymus vartotojui, spauskite Pagalba -> Nurodymai vartotojui. Naujame lange matysite nuorodas kaip naudotis programa.
9. Norėdami peržiūrėti naudojimo sąlygas, spauskite Pagalba -> Naudojimo sąlygos.

## Programos tekstas

**Klasė Krepsininkas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L2

{

class Krepsininkas

{

public string VP { get; set; }

public int Amžius { get; set; }

public double Ūgis { get; set; }

public Krepsininkas(string vrdpav, int amz, double ug)

{

VP = vrdpav;

Amžius = amz;

Ūgis = ug;

}

public override string ToString()

{

string eilute;

eilute = string.Format(" {0, -20} {1, 2} {2, 5:f}", VP, Amžius, Ūgis);

return eilute;

}

public static bool operator <=(Krepsininkas k1, Krepsininkas k2)

{

int p = String.Compare(k1.VP, k2.VP, StringComparison.CurrentCulture);

return ((k1.Amžius < k2.Amžius) || (k1.Amžius == k2.Amžius && p < 0));

}

public static bool operator >=(Krepsininkas k1, Krepsininkas k2)

{

int p = String.Compare(k1.VP, k2.VP, StringComparison.CurrentCulture);

return ((k1.Amžius > k2.Amžius) || (k1.Amžius == k2.Amžius && p > 0));

}

public static bool operator ==(Krepsininkas k1, Krepsininkas k2)

{

int p = String.Compare(k1.VP, k2.VP, StringComparison.CurrentCulture);

return (p == 0 && k1.Amžius == k2.Amžius && k1.Ūgis == k2.Ūgis);

}

public static bool operator !=(Krepsininkas k1, Krepsininkas k2)

{

int p = String.Compare(k1.VP, k2.VP, StringComparison.CurrentCulture);

return (p != 0 || k1.Amžius != k2.Amžius || k1.Ūgis != k2.Ūgis);

}

}

}

**Programos tekstas:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace L2

{

public partial class Form1 : Form

{

const string rez = "..\\..\\rez.txt";

const string naujokai = "..\\..\\Naujokai.txt";

const string nurodymai = "..\\..\\Nurodymai.txt";

const string salygos = "..\\..\\NaudojimoSąlygos.txt";

List<Krepsininkas> mokykla1;

List<Krepsininkas> mokykla2;

List<Krepsininkas> naujasKonteineris;

List<Krepsininkas> naujiKrepsininkai;

string mokPav1;

string mokPav2;

string Vadybininkas;

public Form1()

{

InitializeComponent();

print.Enabled = false;

skaiciuoti.Enabled = false;

pridėtiNaujų.Enabled = false;

išsaugoti.Enabled = false;

if (File.Exists(rez))

File.Delete(rez);

}

/// <summary>

/// Vykdo failų nuskaitymą.

/// </summary>

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

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

private void nuskaityti\_Click(object sender, EventArgs e)

{

label1.Text = "Vyksta duomenų įvedimas. Laukite.";

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog1.Title = "Pasirinkite pirmąjį duomenų failą";

DialogResult result = openFileDialog1.ShowDialog();

if (result == DialogResult.OK)

{

string fv = openFileDialog1.FileName;

mokykla1 = Skaityti(fv, out mokPav1);

}

OpenFileDialog openFileDialog2 = new OpenFileDialog();

openFileDialog2.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog2.Title = "Pasirinkite antrąjį duomenų failą";

DialogResult result2 = openFileDialog2.ShowDialog();

if (result2 == DialogResult.OK)

{

string fv = openFileDialog2.FileName;

mokykla2 = Skaityti(fv, out mokPav2);

}

Spausdinti(rez, mokykla1, mokPav1, "Pradiniai pirmos mokyklos duomenys:");

Spausdinti(rez, mokykla2, mokPav2, "Pradiniai antros mokyklos duomenys:");

naujasKonteineris = new List<Krepsininkas>();

string x = File.ReadAllText(rez);

richTextBox1.Text = x;

skaiciuoti.Enabled = true;

nuskaityti.Enabled = false;

label1.Text = "Duomenys įvesti ir atspausdinti,\ngalima skaičiuoti.";

}

/// <summary>

/// Vykdo skaičiavimus

/// </summary>

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

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

private void skaiciuoti\_Click(object sender, EventArgs e)

{

label1.Text = "Vykdomi skaičiavimai";

double amzVid1 = AmziausVidurkis(mokykla1);

SpausdintiTeksta(rez, string.Format("Amžiaus vidurkis pirmoje ({0})" +

" mokykloje yra: {1, 5:f} m.", mokPav1, amzVid1));

double amzVid2 = AmziausVidurkis(mokykla2);

SpausdintiTeksta(rez, string.Format("Amžiaus vidurkis antroje ({0})" +

" mokykloje yra: {1, 5:f} m.\n", mokPav2, amzVid2));

double ugioVid1 = ŪgioVidurkis(mokykla1);

SpausdintiTeksta(rez, string.Format("Ūgio vidurkis pirmoje ({0})" +

" mokykloje yra: {1, 5:f} m.", mokPav1, ugioVid1));

double ugioVid2 = ŪgioVidurkis(mokykla2);

SpausdintiTeksta(rez, string.Format("Ūgio vidurkis antroje ({0})" +

" mokykloje yra: {1, 5:f} m.\n", mokPav2, ugioVid2));

double vidur = Math.Round(((ugioVid1 + ugioVid2) / 2), 2);

UgisDidesnisUzVidurki(mokykla1, naujasKonteineris, vidur);

UgisDidesnisUzVidurki(mokykla2, naujasKonteineris, vidur);

Spausdinti(rez, naujasKonteineris, "Abi mokyklos",

"Konteineris studentų, kurių ūgis didesnis už vidurkį:");

Rikiuoti(naujasKonteineris);

Spausdinti(rez, naujasKonteineris, "Abi mokyklos",

"Surikiuotas konteineris:");

int amz = int.Parse(textBox1.Text);

Šalinti(naujasKonteineris, amz);

Spausdinti(rez, naujasKonteineris, "Abi mokyklos",

"Pašalinti moksleiviai, kurių amžius didesnis už nurodytą:");

print.Enabled = true;

skaiciuoti.Enabled = false;

label1.Text = "Skaičiavimai atlikti,\ngalima spausdinti.";

pridėtiNaujų.Enabled = true;

}

/// <summary>

/// Spausdina

/// </summary>

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

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

private void print\_Click(object sender, EventArgs e)

{

string x = File.ReadAllText(rez);

richTextBox1.Text = x;

label1.Text = "Duomenys atspausdinti.";

}

/// <summary>

/// Prideda naujus žaidėjus

/// </summary>

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

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

private void pridėtiNaujų\_Click(object sender, EventArgs e)

{

label1.Text = "Pridedami nauji krepšininkai.";

naujiKrepsininkai = Skaityti(naujokai, out Vadybininkas);

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

{

Krepsininkas krep = naujiKrepsininkai[i];

int ind = RastiIndeksą(naujasKonteineris, krep);

naujasKonteineris.Insert(ind, krep);

}

Spausdinti(rez, naujasKonteineris, Vadybininkas, "Sąrašas su" +

" pridėtais naujais krepšininkais:");

label1.Text = "Duomenys pridėti,\ngalima spausdinti.";

išsaugoti.Enabled = true;

pridėtiNaujų.Enabled = false;

}

/// <summary>

/// Leidžia saugoti rezultatus į .csv failą

/// </summary>

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

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

private void išsaugoti\_Click(object sender, EventArgs e)

{

label1.Text = "Krepšininkai spausdinami į .csv failą.";

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "CSV|\*.csv";

saveFileDialog1.Title = "Pasirinkite .csv failą, į kurį norite" +

" spausdinti";

DialogResult result = saveFileDialog1.ShowDialog();

if (result == DialogResult.OK)

{

string fv = saveFileDialog1.FileName;

if (File.Exists(fv))

File.Delete(fv);

SpausdintiĮCSVFailą(fv, naujasKonteineris, "Abi mokyklos",

"Spausdinimas į .csv failą:");

}

label1.Text = "Krepšininkai atspaudinti į .csv failą.";

}

/// <summary>

/// Spausdina nurodymus vartotojui

/// </summary>

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

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

private void nurodymaiVartotojuiToolStripMenuItem\_Click(object sender, EventArgs e)

{

string x = File.ReadAllText(nurodymai);

MessageBox.Show(x);

}

/// <summary>

/// Spausdina naudojimo sąlygas

/// </summary>

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

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

private void naudojimoSąlygosToolStripMenuItem\_Click(object sender, EventArgs e)

{

string x = File.ReadAllText(salygos);

MessageBox.Show(x);

}

/// <summary>

/// Išvalo teksto lauką ant jo paspaudus

/// </summary>

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

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

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

textBox1.Text = "";

}

/// <summary>

/// Baigia programos darbą

/// </summary>

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

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

private void baigti\_Click(object sender, EventArgs e)

{

Close();

}

/// <summary>

/// Rikiuoja duomenis

/// </summary>

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

private void Rikiuoti(List<Krepsininkas> A)

{

int i = 0;

bool bk = true;

while (bk)

{

bk = false;

for (int j = A.Count - 1; j > i; j--)

{

if (A[j] <= A[j - 1])

{

bk = true;

Krepsininkas krep = A[j];

A[j] = A[j - 1];

A[j - 1] = krep;

}

}

i++;

}

}

/// <summary>

/// Šalina duomenis

/// </summary>

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

/// <param name="amz"></param>

private void Šalinti(List<Krepsininkas> A, int amz)

{

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

{

if (A[i].Amžius > amz)

{

A.Remove(A[i]);

i--;

}

}

}

/// <summary>

/// Spausdina į .csv failą

/// </summary>

/// <param name="fw"></param>

/// <param name="kont"></param>

/// <param name="pav"></param>

/// <param name="antraste"></param>

private void SpausdintiĮCSVFailą(string fw, List<Krepsininkas> kont, string pav,

string antraste)

{

using (var fr = new StreamWriter(File.Open(fw, FileMode.Append)))

{

if (kont.Count > 0)

{

fr.WriteLine(";" + antraste);

fr.WriteLine(";" + pav);

fr.WriteLine(" {0, -5} ; {1, -20} ; {2, -8} ; {3, -6} ", "Nr.",

"Vardas Pavardė", "Amžius", "Ūgis");

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

{

Krepsininkas krep = kont[i];

fr.WriteLine(" {0, -3} ; {1, -20} ; {2, 3:d} ; {3, 5:f} ",

(i + 1).ToString(), krep.VP, krep.Amžius, krep.Ūgis);

}

}

else

{

fr.WriteLine("Studentų konteineris tuščias!\n");

}

}

}

/// <summary>

/// Randa įterpiamų krepšininkų indeksus

/// </summary>

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

/// <param name="krep"></param>

/// <returns></returns>

private int RastiIndeksą(List<Krepsininkas> A, Krepsininkas krep)

{

int ind = A.Count;

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

{

if (krep <= A[i])

{

ind = i;

}

}

return ind;

}

/// <summary>

/// Formuoja naują konteinerį iš krepšininkų, kurių ūgis didesnis už vidurkį

/// </summary>

/// <param name="kont1">Pirmas konteineris</param>

/// <param name="kont2">Antras kont</param>

/// <param name="naujas">Trečias kont</param>

private void UgisDidesnisUzVidurki(List<Krepsininkas> kont, List<Krepsininkas> naujas,

double ugis)

{

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

{

if (kont[i].Ūgis > ugis)

{

naujas.Add(kont[i]);

}

}

}

/// <summary>

/// Randa krepšininkų amžiaus vidurkį

/// </summary>

/// <param name="kont">Konteineris</param>

/// <returns>Amžiaus vidurkis</returns>

static double AmziausVidurkis(List<Krepsininkas> kont)

{

double suma = 0;

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

{

suma += kont[i].Amžius;

}

return suma / kont.Count;

}

/// <summary>

/// Randa krepšininkų ūgio vidurkį

/// </summary>

/// <param name="kont">Konteineris</param>

/// <returns>Ūgio vidurkis</returns>

static double ŪgioVidurkis(List<Krepsininkas> kont)

{

double suma = 0;

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

{

suma += kont[i].Ūgis;

}

return suma / kont.Count;

}

/// <summary>

/// Failų nuskaitymas

/// </summary>

/// <param name="fr"></param>

/// <param name="pav"></param>

/// <returns></returns>

private List<Krepsininkas> Skaityti(string fr, out string pav)

{

List<Krepsininkas> konteineris = new List<Krepsininkas>();

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

{

string line;

line = reader.ReadLine();

pav = line;

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

{

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

string pavVrd = parts[0];

int amz = int.Parse(parts[1]);

double ug = double.Parse(parts[2]);

Krepsininkas krep = new Krepsininkas(pavVrd, amz, ug);

konteineris.Add(krep);

}

return konteineris;

}

}

/// <summary>

/// Spausdina rezultatus į failus

/// </summary>

/// <param name="fw"></param>

/// <param name="kont"></param>

/// <param name="pav"></param>

/// <param name="antraste"></param>

private void Spausdinti(string fw, List<Krepsininkas> kont, string pav,

string antraste)

{

const string virsus =

"------------------------------------------\r\n" +

" Nr. Pavardė ir vardas Amžius Ūgis \r\n" +

"------------------------------------------";

using (var fr = new StreamWriter(File.Open(fw, FileMode.Append)))

{

if (kont.Count > 0)

{

fr.WriteLine(antraste);

fr.WriteLine(pav);

fr.WriteLine(virsus);

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

{

Krepsininkas krep = kont[i];

fr.WriteLine("{0, 3} {1}", i + 1, krep.ToString());

}

fr.WriteLine("------------------------------------------\n");

}

else

{

fr.WriteLine("Studentų konteineris tuščias!\n");

}

}

}

/// <summary>

/// Spausdina tekstą į failus

/// </summary>

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

/// <param name="x"></param>

private void SpausdintiTeksta(string rez, string x)

{

using (StreamWriter fw = new StreamWriter(File.Open(rez, FileMode.Append)))

{

fw.WriteLine(x);

}

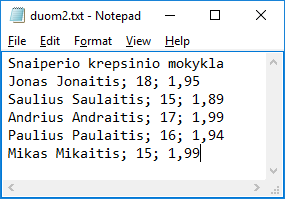
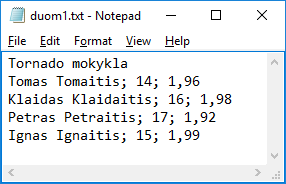
}

}

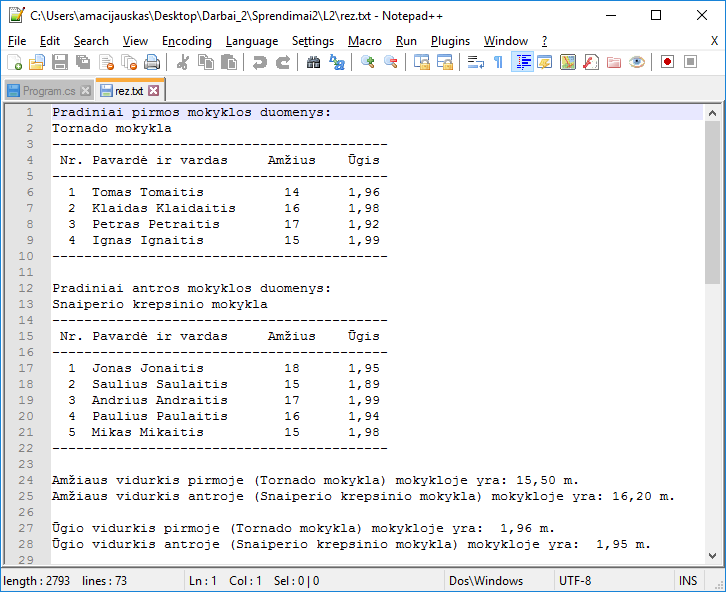
}

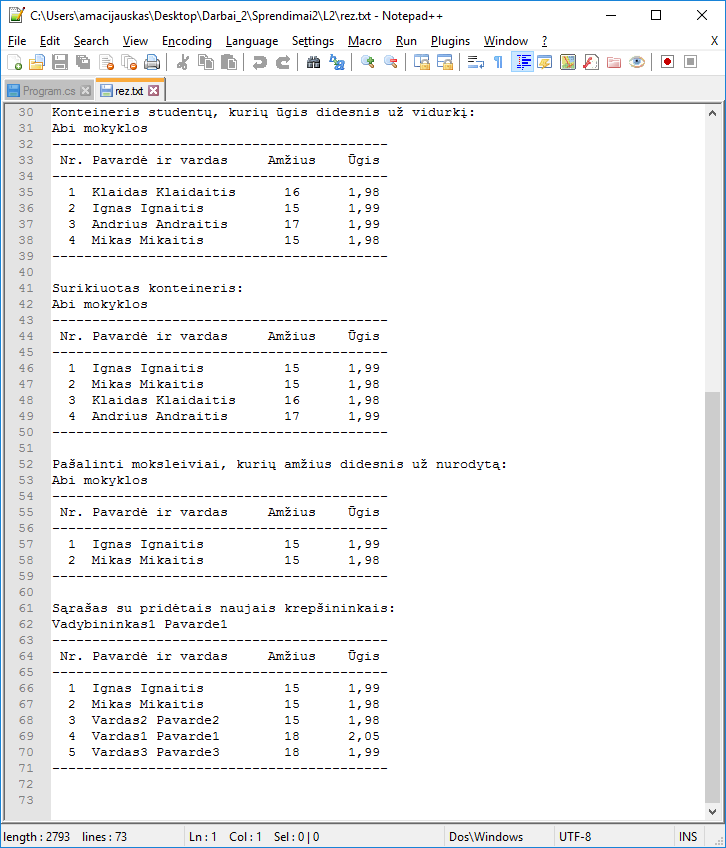
## Pradiniai duomenys ir rezultatai

Pradiniai duomenys nr. 1:

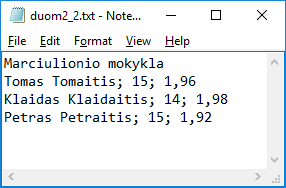
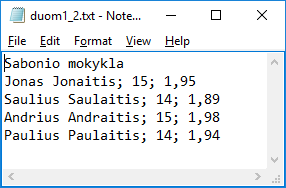


Rezultatai nr. 1:

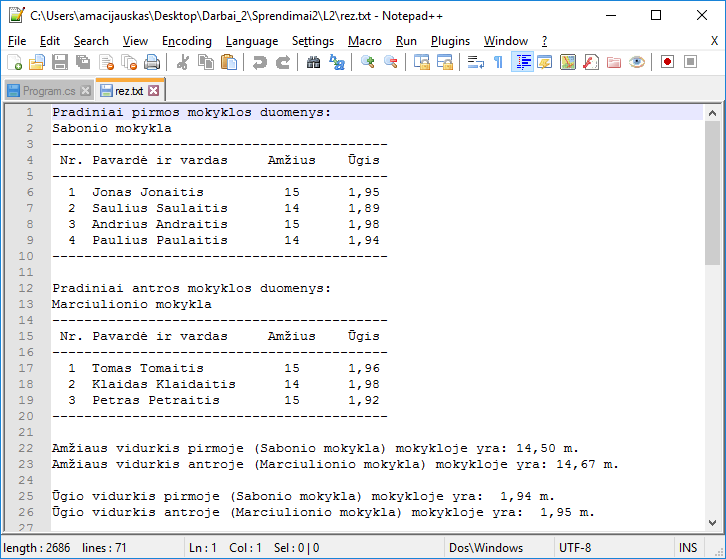


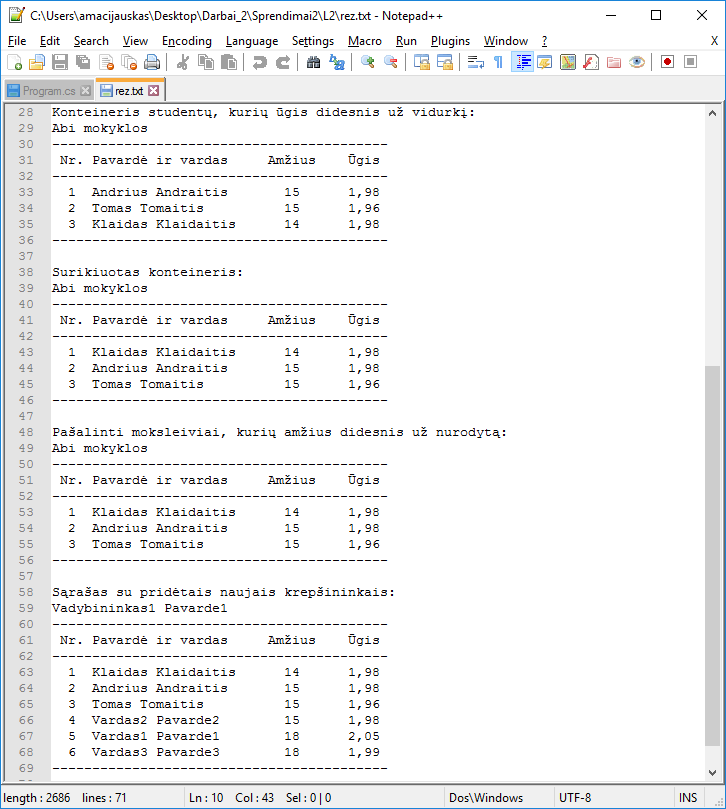


Pradiniai duomenys nr. 2:



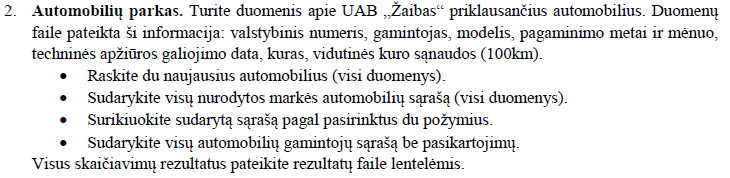
Rezultatai nr. 2:





# Paveldėjimas (L3)

## Darbo užduotis



## Grafinės vartotojo sąsajos schema ir paveikslas

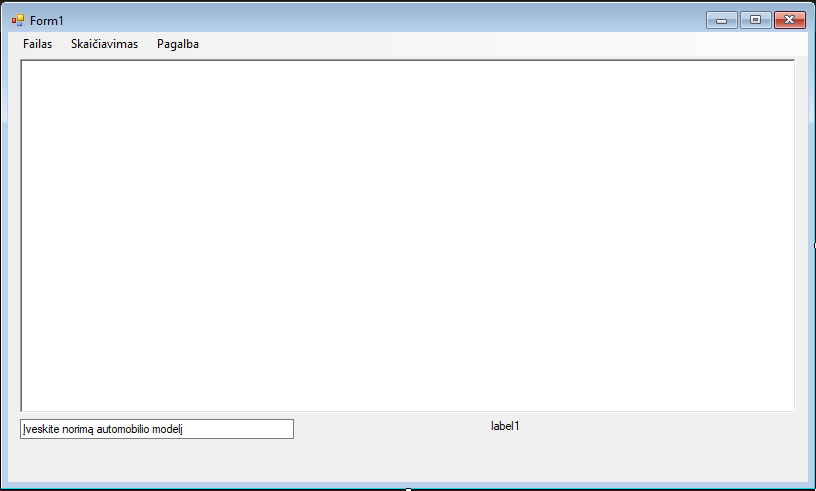
Schema:

Form1

label1

richTextBox1

menuStrip1

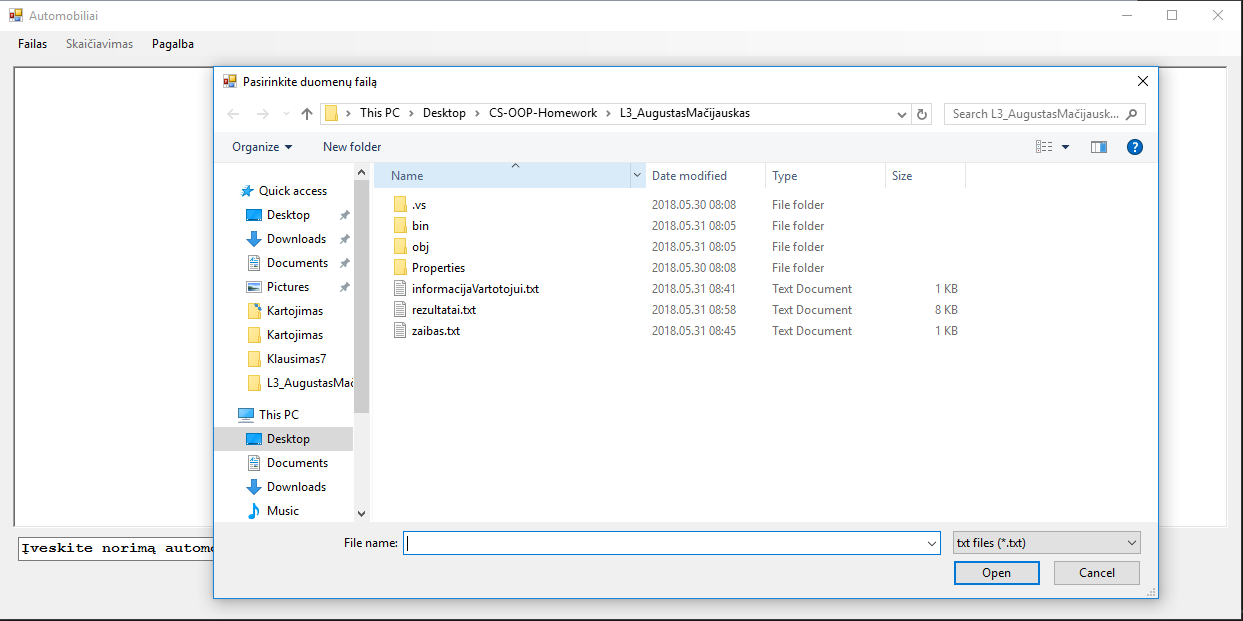


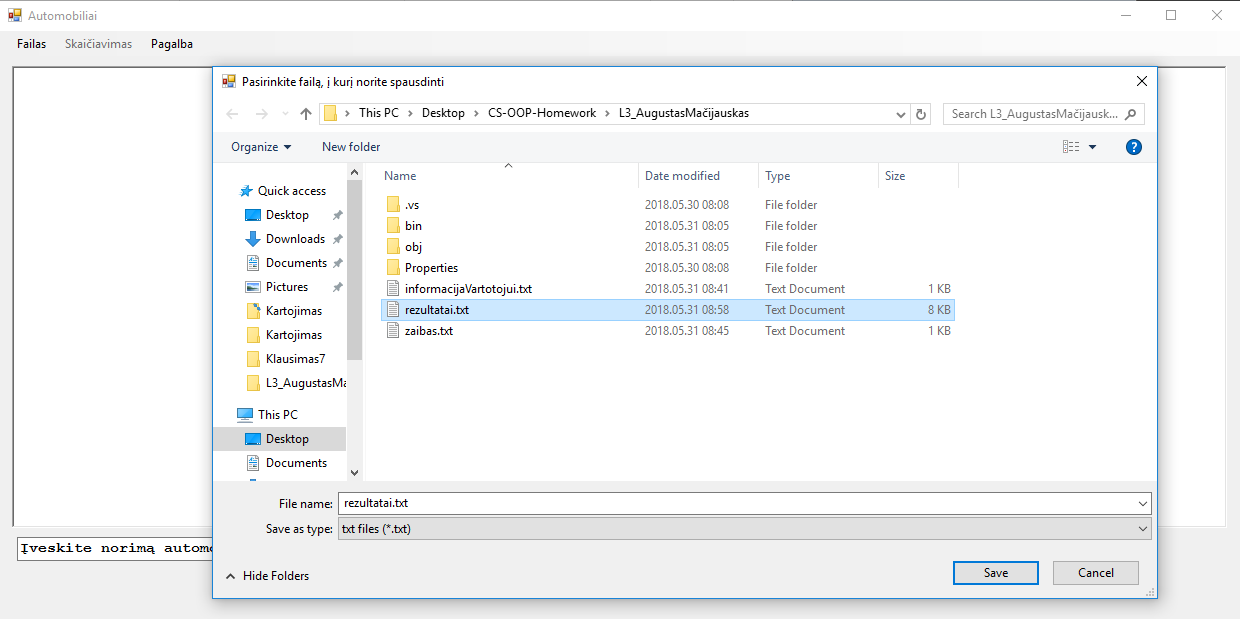
textBox1

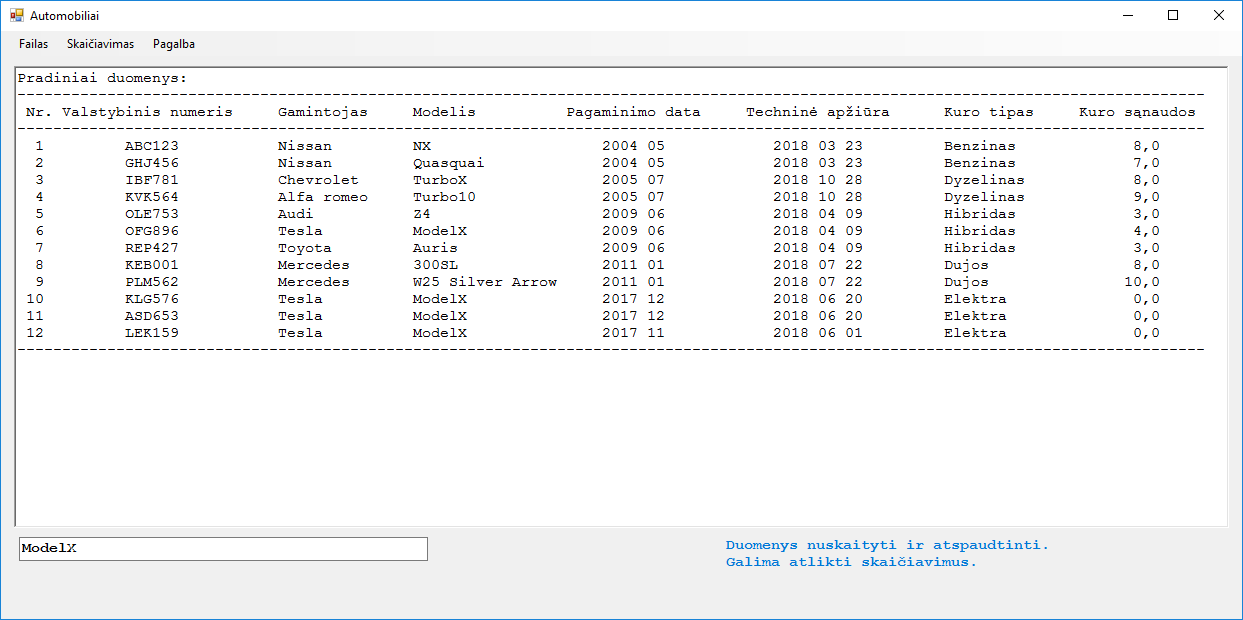
Meniu punktai:

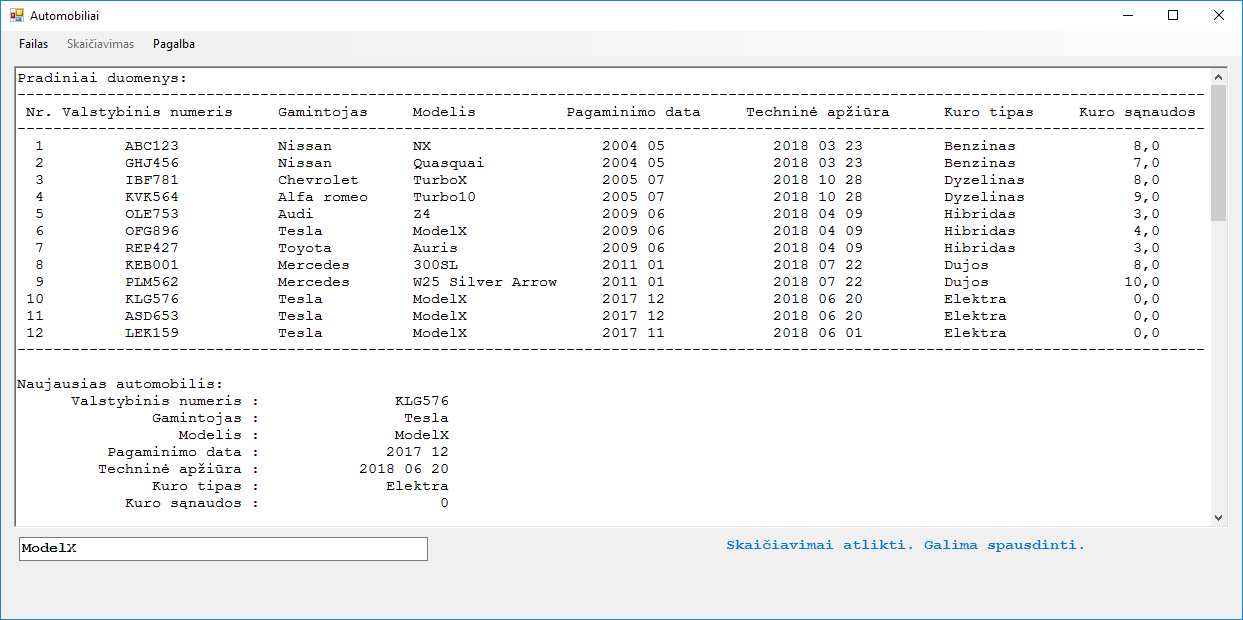
|  |  |  |
| --- | --- | --- |
| Failas | Skaičiavimas | Pagalba |
| Įvesti | Skaičiuoti | Informacija |
| Baigti | Spausdinti |  |

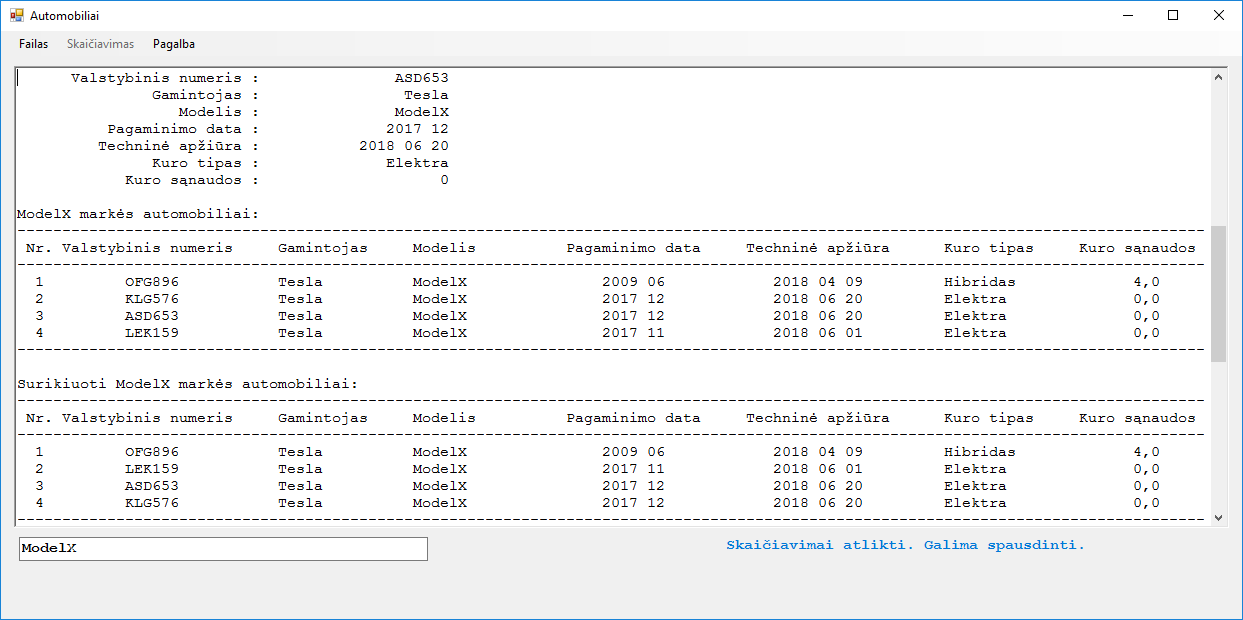
Paveikslas:

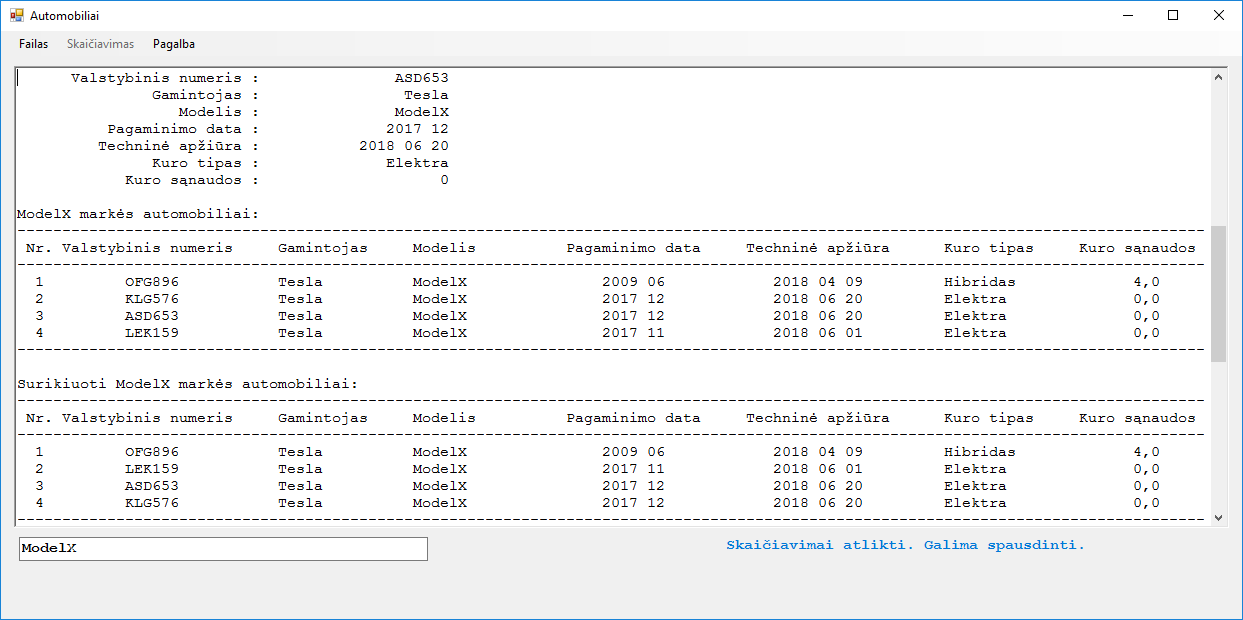


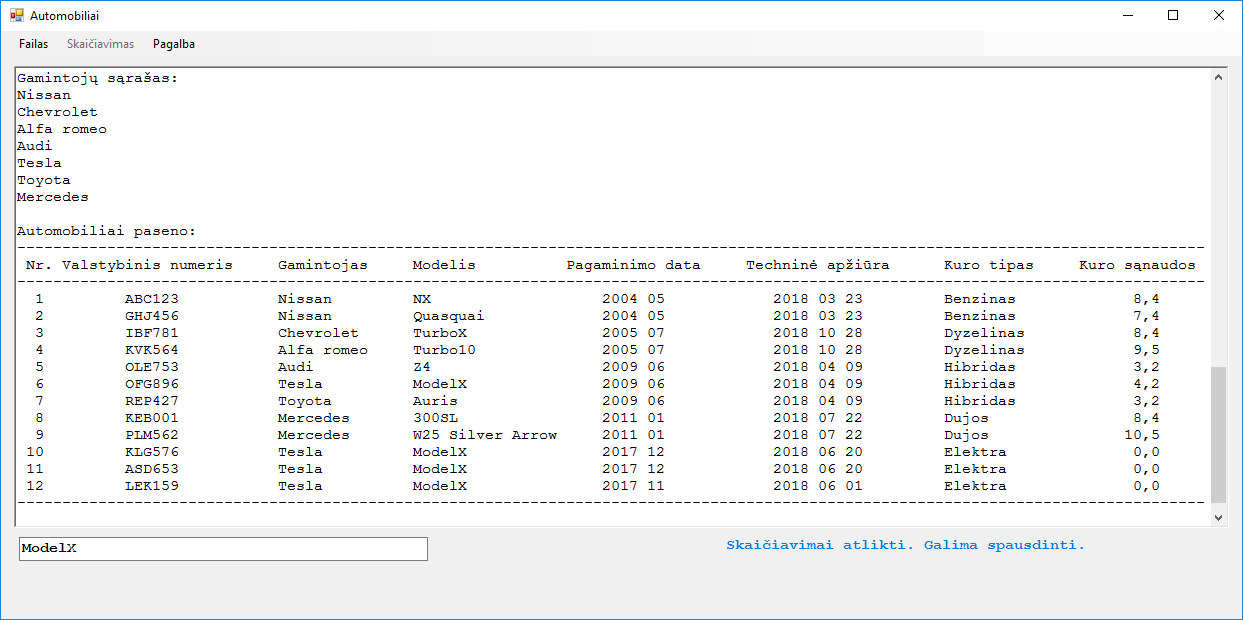












## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Form1 | Text: Automobiliai  Name: Automobiliai | Programos langas |
| richTextBox1  (tipas RichTextBox) | Font: Courier New, Regular, 11;  Script: Baltic; | Atvaizduoti pradinių duomenų ir skaičiavimų turiniui |
| textBox1  (tipas TextBox) | Text: Įveskite norimą amžių  Font: Courier New, Bold, 11;  Script: Baltic; | Norimo automobilio modelio įvedimui |
| label1  (tipas Label) | Text: Pranešimas  Font: Courier New, Bold, 11;  Script: Baltic;  ForeColor: Highlight | Spausdina pranešimus apie programos veikimą. |
| menuStrip1  (tipas MenuStrip) | Font: Standartiniai nustatymai | Naudojamas meniu punktų, kurie bus naudojami programos veikimo metu, išdėstymui. |
| failasToolStripMenuItem  (tipas ToolStripMenu) | Text: Failas  Font: Standartiniai nustatymai | Meniu punktas, kuriame yra mygtukai duomenų nuskaitymui ir programos darbo baigimui |
| ivestiToolStripMenuItem  (tipas ToolStripMenu) | Name: Ivesti  Text: Įvesti  Font: Standartiniai nustatymai | Meniu punktas duomenims iš failo nuskaityti ir rezultatų failui pasirinkti |
| baigtiToolStripMenuItem  (tipas ToolStripMenu) | Name: Baigti  Text: Baigti  Font: Standartiniai nustatymai | Meniu punktas, baigiantis darbą su programa |
| skaičiavimasToolStripMenuItem  (tipas ToolStripMenu) | Text: Skaičiavimas  Font: Standartiniai nustatymai | Meniu punktas, kuriame yra mygtukai skaičiavimams ir spausdinimui atlikti |
| skaiciuotiToolStripMenuItem  (tipas ToolStripMenu) | Name: Skaiciuoti  Text: Skaičiuoti  Font: Standartiniai nustatymai | Meniu punktas skaičiavimams atlikti |
| spausdintiToolStripMenuItem  (tipas ToolStripMenu) | Name: Spausdinti  Text: Spausdinti  Font: Standartiniai nustatymai | Meniu punktas gautiems rezultatams atspausdinti |
| pagalbaToolStripMenuItem  (tipas ToolStripMenu) | Text: Pagalba  Font: Standartiniai nustatymai | Meniu punktas, turintis mygtukus, kurie atvaizduoja nurodymus vartotojui ir naudojimo sąlygas. |
| informacijaToolStripMenuItem (tipas ToolStripMenu) | Text: Informacija  Font: Standartiniai nustatymai | Meniu punktas, spausdinantis pagalbą vartotojui. |

## Programos vartotojo vadovas

PASTABA: norėdami peržiūrėti pagalbą vartotojui, spauskite Pagalba -> Informacija. Norėdami baigti darbą su programa, spauskite Failas -> Baigti.

1. Atidarę programą, spauskite Failas -> Įvesti.
2. Pasirinkite duomenų failą, iš kurio norite nuskaityti.
3. Pasirinkite rezultatų failą, į kurį norite spausdinti.
4. Į teksto laukelį „Įveskite norimą automobilio modelį“ įveskite savo norimą automobilio modelį.
5. Spauskite Skaičiavimas -> Skaičiuoti. Bus atlikti skaičiavimai.
6. Kai skaičiavimai bus atlikti, spauskite Skaičiavimas -> Spausdinti.
7. Programos veikimas baigtas. Galite peržiūrėti gautus rezultatus ir uždaryti programą.

## Programos tekstas

**Abstrakti klasė AutomobilioBlueprint:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace L3\_AugustasMačijauskas

{

abstract class AutomobilioBlueprint

{

public string Gamintojas { get; set; }

public string Modelis { get; set; }

public DateTime PagaminimoData { get; set; }

public string Kuras { get; set; }

public AutomobilioBlueprint(string gam, string mod, DateTime pag, string kur)

{

this.Gamintojas = gam;

this.Modelis = mod;

this.PagaminimoData = pag;

this.Kuras = kur;

}

public override string ToString()

{

return string.Format(" {0, -10} {1, -7} {2:yyyy MM} {3, -15} ", Gamintojas, Modelis, PagaminimoData, Kuras);

}

public abstract void Nusidėvėjimas();

}

}

**AutomobilioBlueprint klasę paveldinti išvestinė klasė:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace L3\_AugustasMačijauskas

{

class Automobilis : AutomobilioBlueprint

{

public string ValstybinisNumeris { get; set; }

public DateTime TechninėApžiūra { get; set; }

public double VidutinėsSąnaudos { get; set; }

public Automobilis(string valstNr, string gam, string mod, DateTime pag, DateTime tech, string kur, double vidSan) : base (gam, mod, pag, kur)

{

this.ValstybinisNumeris = valstNr;

this.TechninėApžiūra = tech;

this.VidutinėsSąnaudos = vidSan;

}

public static bool operator <(Automobilis a1, Automobilis a2)

{

int poz = string.Compare(a1.ValstybinisNumeris, a2.ValstybinisNumeris, StringComparison.CurrentCulture);

return ((a1.TechninėApžiūra < a2.TechninėApžiūra) || ((a1.TechninėApžiūra == a2.TechninėApžiūra) && (poz < 0)));

}

public static bool operator >(Automobilis a1, Automobilis a2)

{

int poz = string.Compare(a1.ValstybinisNumeris, a2.ValstybinisNumeris, StringComparison.CurrentCulture);

return ((a1.TechninėApžiūra > a2.TechninėApžiūra) || ((a1.TechninėApžiūra == a2.TechninėApžiūra) && (poz > 0)));

}

public static bool operator <=(Automobilis a1, Automobilis a2)

{

return a1.PagaminimoData < a2.PagaminimoData;

}

public static bool operator >=(Automobilis a1, Automobilis a2)

{

return a1.PagaminimoData > a2.PagaminimoData;

}

public override string ToString()

{

return string.Format(" {0, 6} {1, -10} {2, -20} {3:yyyy MM} {4:yyyy MM dd} {5, -15} {6, 5:f1}", ValstybinisNumeris, Gamintojas, Modelis, PagaminimoData, TechninėApžiūra, Kuras, VidutinėsSąnaudos);

}

public override void Nusidėvėjimas()

{

VidutinėsSąnaudos \*= 1.05;

}

}

}

**Programos tekstas:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace L3\_AugustasMačijauskas

{

public partial class Autumobiliai : Form

{

List<Automobilis> automobiliai;

List<Automobilis> markes;

List<string> gamintojai;

string rez = "";

const string info = "..\\..\\informacijaVartotojui.txt";

public Autumobiliai()

{

InitializeComponent();

Skaiciuoti.Enabled = false;

Spausdinti.Enabled = false;

skaičiavimasToolStripMenuItem.Enabled = false;

}

private void įvestiToolStripMenuItem\_Click(object sender, EventArgs e)

{

label1.Text = "Vyksta duomenų įvedimas. Prašome palaukti.";

OpenFileDialog openFile = new OpenFileDialog();

openFile.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFile.Title = "Pasirinkite duomenų failą";

DialogResult result = openFile.ShowDialog();

if (result == DialogResult.OK)

{

string fn = openFile.FileName;

automobiliai = Read(fn);

}

label1.Text = "Duomenys nuskaityti. Pasirinkite failą,\nį kurį norite spausdinti.";

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFile.Title = "Pasirinkite failą, į kurį norite spausdinti";

result = saveFile.ShowDialog();

if (result == DialogResult.OK)

{

rez = saveFile.FileName;

if (File.Exists(rez))

File.Delete(rez);

Write(rez, automobiliai, "Pradiniai duomenys:");

}

string x = File.ReadAllText(rez);

richTextBox1.Text = x;

label1.Text = "Duomenys nuskaityti ir atspaudtinti.\nGalima atlikti skaičiavimus.";

Skaiciuoti.Enabled = true;

Ivesti.Enabled = false;

skaičiavimasToolStripMenuItem.Enabled = true;

}

private List<Automobilis> Read(string fn)

{

List<Automobilis> ret = new List<Automobilis>();

using (StreamReader reader = new StreamReader(fn))

{

string line;

string[] parts;

string vn, gam, mod, kur;

DateTime pag, tech;

double san;

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

{

parts = line.Split(';');

vn = parts[0].Trim();

gam = parts[1].Trim();

mod = parts[2].Trim();

pag = DateTime.Parse(parts[3].Trim());

tech = DateTime.Parse(parts[4].Trim());

kur = parts[5].Trim();

san = double.Parse(parts[6].Trim());

Automobilis naujas = new Automobilis(vn, gam, mod, pag, tech, kur, san);

ret.Add(naujas);

}

}

return ret;

}

private void Write(string fn, List<Automobilis> kont, string antraste)

{

const string virsus =

"------------------------------------------------------------------------------------------------------------------------------------\r\n" +

" Nr. Valstybinis numeris Gamintojas Modelis Pagaminimo data Techninė apžiūra Kuro tipas Kuro sąnaudos \r\n" +

"------------------------------------------------------------------------------------------------------------------------------------";

using (var writer = new StreamWriter(File.Open(fn, FileMode.Append)))

{

if (kont.Count < 0)

{

writer.WriteLine("Studentų konteineris tuščias!\n");

}

else

{

writer.WriteLine(antraste);

writer.WriteLine(virsus);

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

{

Automobilis krep = kont[i];

writer.WriteLine("{0, 3} {1}", i + 1, krep.ToString());

}

writer.WriteLine("------------------------------------------------------------------------------------------------------------------------------------\n");

}

}

}

private void Write(string fn, List<string> gamintojai, string antraste)

{

using (var writer = new StreamWriter(File.Open(fn, FileMode.Append)))

{

writer.WriteLine(antraste);

foreach (string gamintojas in gamintojai)

{

writer.WriteLine(gamintojas);

}

writer.WriteLine();

}

}

private void Write(string fn, string text)

{

using (var writer = new StreamWriter(File.Open(fn, FileMode.Append)))

{

writer.WriteLine(text);

}

}

private void Baigti\_Click(object sender, EventArgs e)

{

Close();

}

private void Skaiciuoti\_Click(object sender, EventArgs e)

{

label1.Text = "Atliekami skaičiavimai. Prašome palaukti.";

Automobilis naujausias1;

Automobilis naujausias2;

RastiDuNaujausius(automobiliai, out naujausias1, out naujausias2);

Write(rez, "Naujausias automobilis:\n" +

$"{"Valstybinis numeris",25} : {naujausias1.ValstybinisNumeris,20}\n" +

$"{"Gamintojas",25} : {naujausias1.Gamintojas,20}\n" +

$"{"Modelis",25} : {naujausias1.Modelis,20}\n" +

$"{"Pagaminimo data",25} : {naujausias1.PagaminimoData.ToString("yyyy MM"),20}\n" +

$"{"Techninė apžiūra",25} : {naujausias1.TechninėApžiūra.ToString("yyyy MM dd"),20}\n" +

$"{"Kuro tipas",25} : {naujausias1.Kuras,20}\n" +

$"{"Kuro sąnaudos",25} : {naujausias1.VidutinėsSąnaudos,20}\n");

Write(rez, "Antras naujausias automobilis:\n" +

$"{"Valstybinis numeris",25} : {naujausias2.ValstybinisNumeris,20}\n" +

$"{"Gamintojas",25} : {naujausias2.Gamintojas,20}\n" +

$"{"Modelis",25} : {naujausias2.Modelis,20}\n" +

$"{"Pagaminimo data",25} : {naujausias2.PagaminimoData.ToString("yyyy MM"),20}\n" +

$"{"Techninė apžiūra",25} : {naujausias2.TechninėApžiūra.ToString("yyyy MM dd"),20}\n" +

$"{"Kuro tipas",25} : {naujausias2.Kuras,20}\n" +

$"{"Kuro sąnaudos",25} : {naujausias2.VidutinėsSąnaudos,20}\n");

string marke = textBox1.Text;

markes = AtrinktiPagalMarke(automobiliai, marke);

if (markes.Count > 0)

{

Write(rez, markes, $"{marke} markės automobiliai:");

Rikiuoti(markes);

Write(rez, markes, $"Surikiuoti {marke} markės automobiliai:");

}

else

{

Write(rez, "Modelių sąrašas tuščias!\n");

}

gamintojai = AtrinktiGamintojus(automobiliai);

Write(rez, gamintojai, "Gamintojų sąrašas:");

Paseno(automobiliai);

Write(rez, automobiliai, "Automobiliai paseno:");

Skaiciuoti.Enabled = false;

Spausdinti.Enabled = true;

label1.Text = "Skaičiavimai atlikti. Galima spausdinti.";

}

private void Paseno(List<Automobilis> automobiliai)

{

// automobiliai.ForEach(x => x.Nusidėvėjimas());

foreach (Automobilis auto in automobiliai)

{

auto.Nusidėvėjimas();

}

}

private List<string> AtrinktiGamintojus(List<Automobilis> automobiliai)

{

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

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

{

if (!ret.Contains(automobiliai[i].Gamintojas))

{

ret.Add(automobiliai[i].Gamintojas);

}

}

return ret;

}

private void Rikiuoti(List<Automobilis> automobiliai)

{

int minIndex;

for (int i = 0; i < automobiliai.Count - 1; i++)

{

minIndex = i;

for (int j = i + 1; j < automobiliai.Count; j++)

{

if (automobiliai[j] < automobiliai[minIndex])

{

minIndex = j;

}

Automobilis temp = automobiliai[i];

automobiliai[i] = automobiliai[minIndex];

automobiliai[minIndex] = temp;

}

}

}

private List<Automobilis> AtrinktiPagalMarke(List<Automobilis> automobiliai, string marke)

{

List<Automobilis> ret = new List<Automobilis>();

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

{

if (automobiliai[i].Modelis == marke)

{

Automobilis naujas = new Automobilis(automobiliai[i].ValstybinisNumeris,

automobiliai[i].Gamintojas, automobiliai[i].Modelis,

automobiliai[i].PagaminimoData, automobiliai[i].TechninėApžiūra,

automobiliai[i].Kuras, automobiliai[i].VidutinėsSąnaudos);

ret.Add(naujas);

}

}

return ret;

}

private void RastiDuNaujausius(List<Automobilis> automobiliai, out Automobilis a1, out Automobilis a2)

{

a1 = automobiliai[0];

a2 = automobiliai[1];

if (a2 >= a1)

{

Automobilis temp = a1;

a1 = a2;

a2 = temp;

}

for (int i = 2; i < automobiliai.Count; i++)

{

Automobilis temp = automobiliai[i];

if (temp >= a1 && temp >= a2)

{

a2 = a1;

a1 = temp;

}

else if (temp >= a2)

{

a2 = temp;

}

}

}

private void Spausdinti\_Click(object sender, EventArgs e)

{

string txt = File.ReadAllText(rez);

richTextBox1.Text = txt;

skaičiavimasToolStripMenuItem.Enabled = false;

}

private void Informacija\_Click(object sender, EventArgs e)

{

label1.Text = "Peržiūrima informacija vartotojui.";

string x = File.ReadAllText(info);

MessageBox.Show(x);

}

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

textBox1.Text = "";

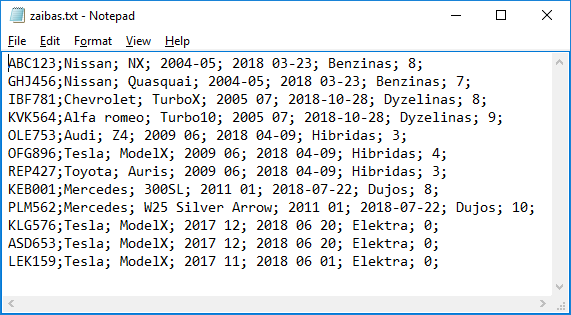
}

}

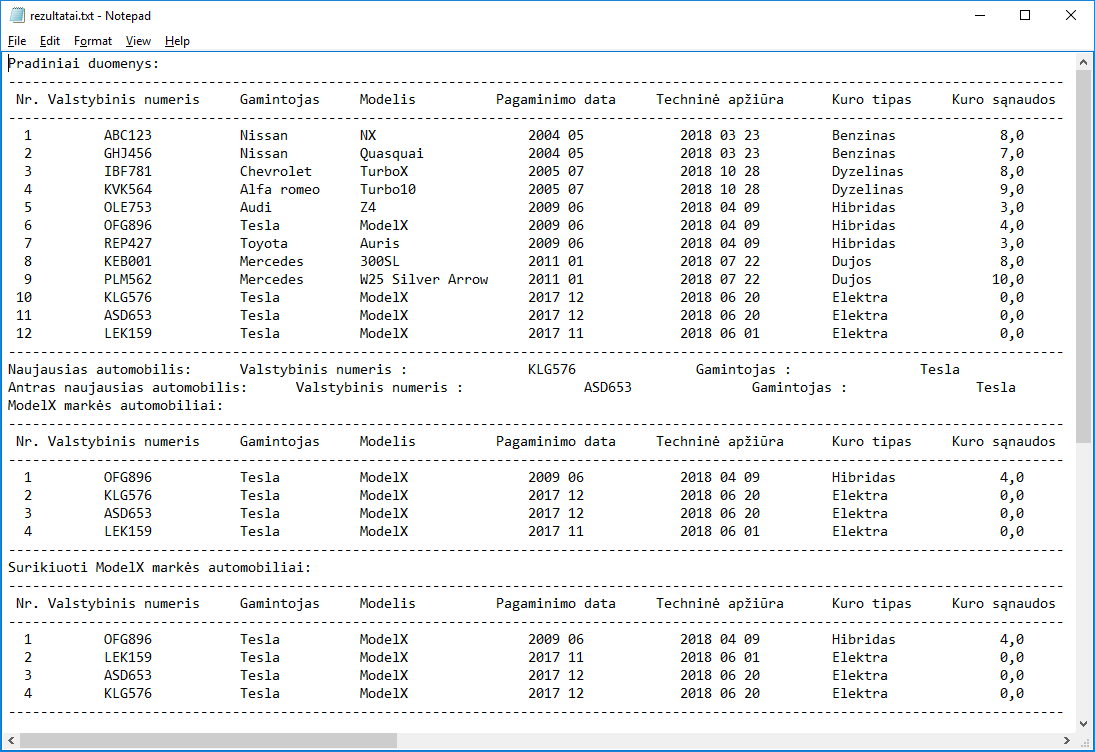
}

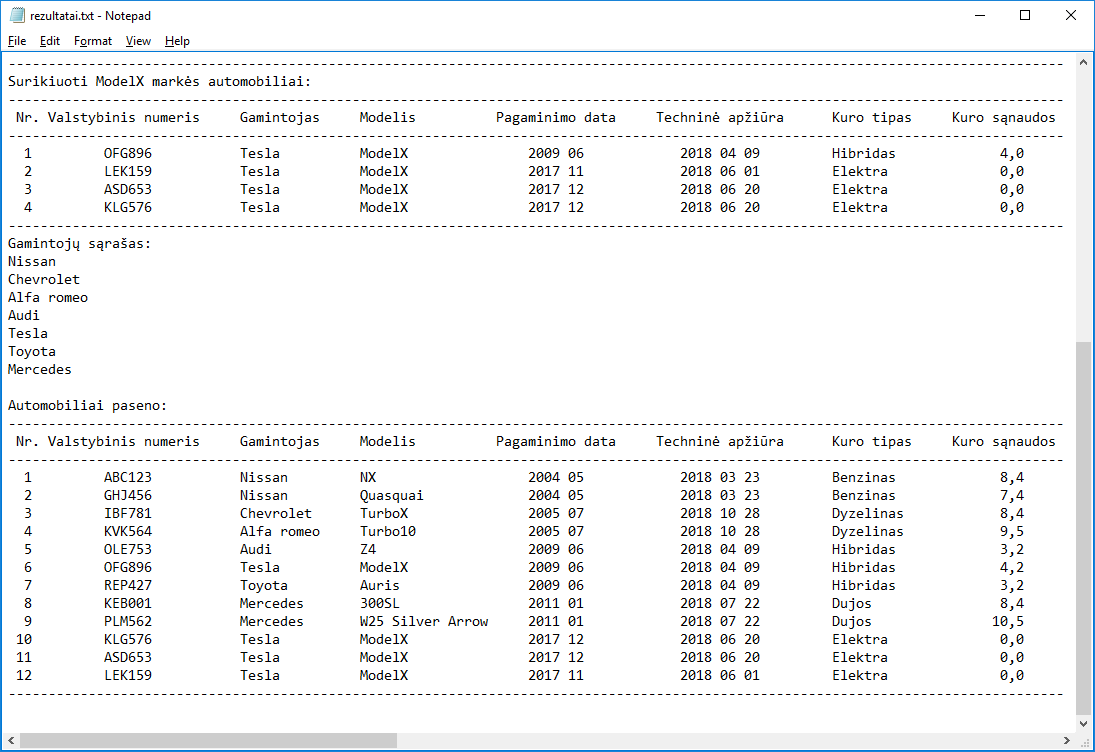
## Pradiniai duomenys ir rezultatai

Pradiniai duomenys nr. 1:

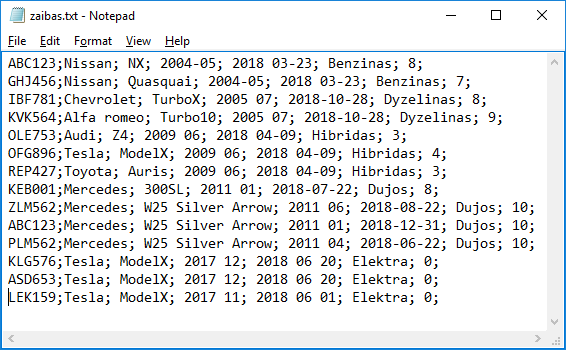


Rezultatai nr. 1:

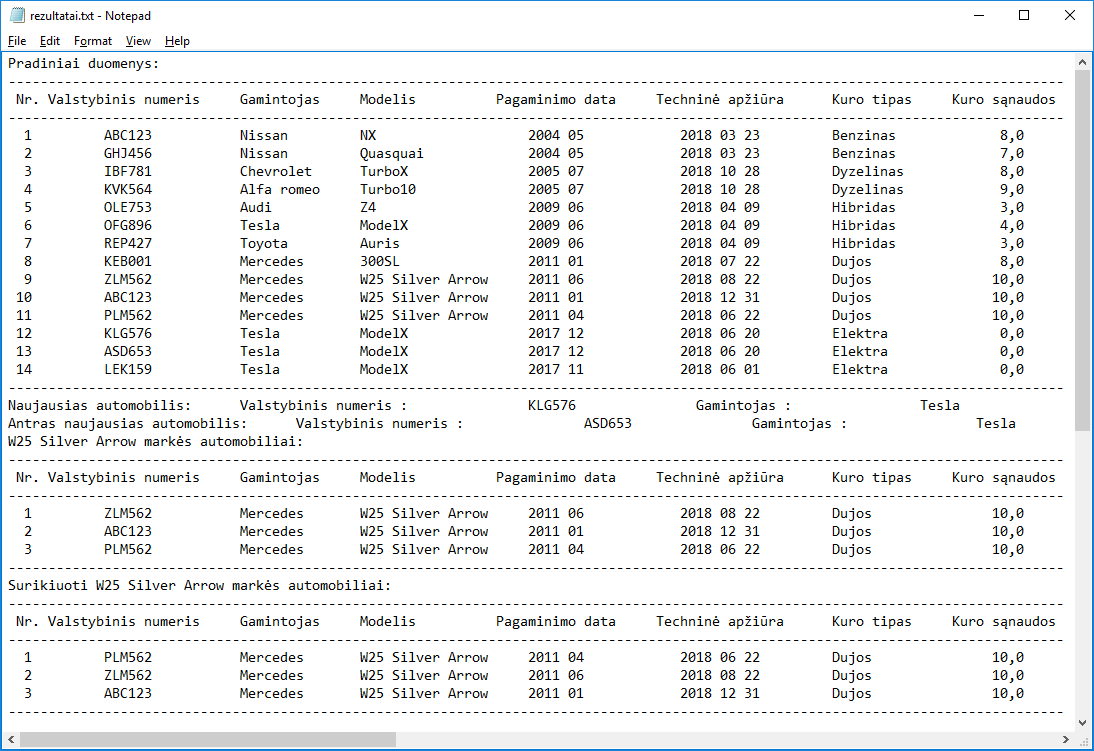


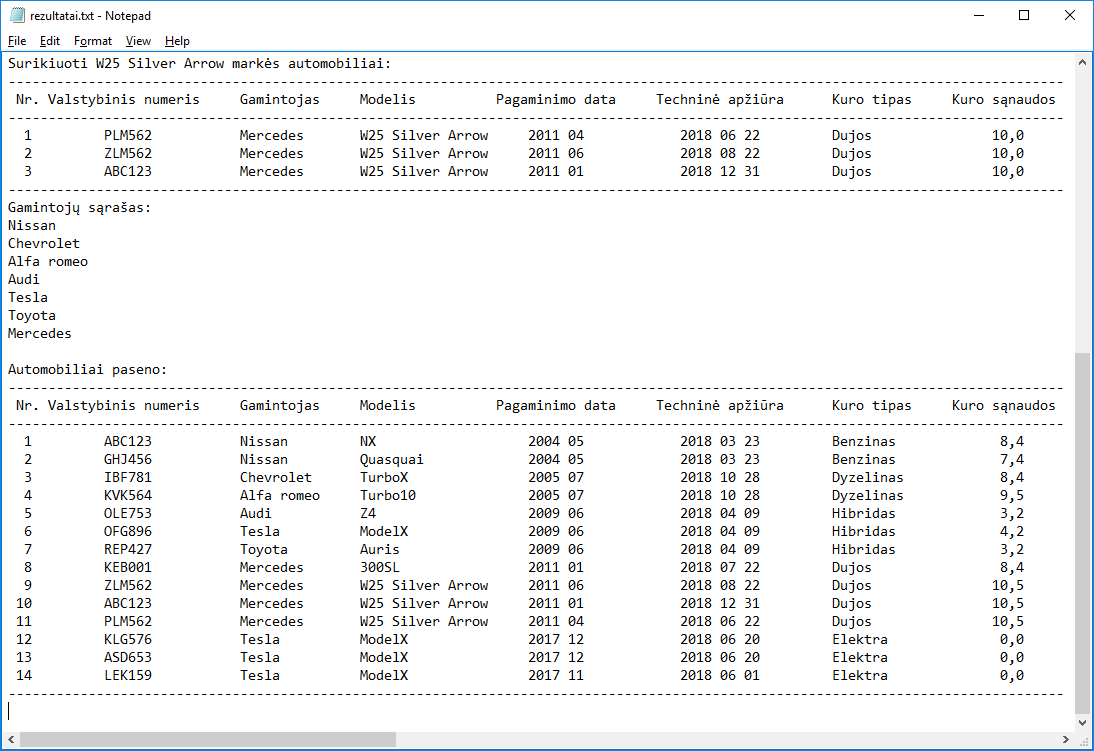


Pradiniai duomenys nr. 2:



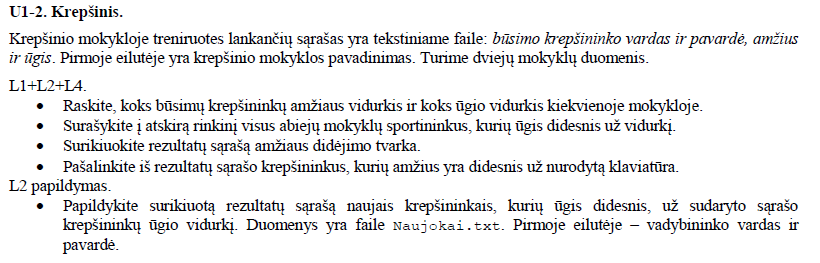
Rezultatai nr. 2:





# Susietasis sąrašas (L4)

## Darbo užduotis



## Grafinės vartotojo sąsajos schema ir paveikslas

## Sąsajoje panaudotų komponentų keičiamos savybės

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

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai

# Bendrinės klasės (L5)

## Darbo užduotis

## Grafinės vartotojo sąsajos schema ir paveikslas

## Sąsajoje panaudotų komponentų keičiamos savybės

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

## Programos vartotojo vadovas

## Programos tekstas

## Pradiniai duomenys ir rezultatai