KAUNO TECHNOLOGIJOS UNIVERSITETO GIMNAZIJA

OBJEKTINIO PROGRAMAVIMO PAGRINDAI I (P175B117)

Laboratorinių darbų ataskaita

Atliko:

IIINFQ klasės mokinys

Augustas Mačijauskas

2018 m. vasario 7 d.

Priėmė:

Kęstutis Simonavičius, Jūratė Pauliutė

KAUNAS 2018

TURINYS

1. Pažintis su klase 3

1.1. Darbo užduotis 3

1.2. Programos tekstas 3

1.3. Pradiniai duomenys ir rezultatai 6

2. Objektų rinkinys 7

2.1. Darbo užduotis 7

2.2. Programos tekstas 7

2.3. Pradiniai duomenys ir rezultatai 12

3. Konteinerinė klasė 16

3.1. Darbo užduotis 16

3.2. Programos tekstas 16

3.3. Pradiniai duomenys ir rezultatai 19

4. Teksto analizė ir redagavimas 22

4.1. Darbo užduotis 22

4.2. Programos tekstas 22

4.3. Pradiniai duomenys ir rezultatai 23

5. Susieti rinkiniai 25

5.1. Darbo užduotis 25

5.2. Programos tekstas 25

5.3. Pradiniai duomenys ir rezultatai 31

# Pažintis su klase

## Darbo užduotis

* Sukurkite klasę Studentas, kuri turėtų kintamuosius amžiui ir ūgiui saugoti. Trys studentai nutarė treniruotis žaisti krepšinį. Raskite, koks aukščiausio studento amžius ir koks jauniausio studento ūgis.
* Papildykite klasę Studentas kintamuoju, skirtu studento svoriui saugoti. Sukurkite klasę Liftas, kuri turėtų kintamuosius lifto keliamosios galios reikšmei ir talpai saugoti. Per kelis kartus visi studentai pakils liftu į reikiamą aukštą?
* Papildykite klasę Liftas metodais Dėti(), kurie leistų keisti lifto keliamąją galią ir talpą. Ar visi studentai vienu metu bus pakelti į reikiamą aukštį, jeigu lifto keliamoji galia bus padvigubinta? O jeigu talpa bus padvigubinta?

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Labaratorinis\_U2\_2

{

class Studentas

{

private int amzius, //amzius studento

svoris; //studento svoris

double ugis; //studento ugis

public Studentas(int amzius, double ugis, int svoris)

{

this.amzius = amzius;

this.ugis = ugis;

this.svoris = svoris;

}

public int koksAmzius() { return amzius; }

public double koksUgis() { return ugis; }

public int koksSvoris() { return svoris; }

}

class Liftas

{

private int galia,

talpa;

public Liftas(int galia, int talpa)

{

this.galia = galia;

this.talpa = talpa;

}

public int kokiaGalia() { return galia; }

public int kokiaTalpa() { return talpa; }

public void DidintiGalia(int x)

{

galia \*= x;

}

public void DidintiTalpa(int x)

{

talpa \*= x;

}

public void MazintiTalpa(int x)

{

talpa /= x;

}

}

class Program

{

static void Main(string[] args)

{

Studentas[] studentai = new Studentas[3];

studentai[0] = new Studentas(20, 1.78, 76);

studentai[1] = new Studentas(19, 1.85, 82);

studentai[2] = new Studentas(21, 1.79, 73);

Liftas l = new Liftas(150, 3);

for (int i = 0; i < studentai.Length; i++)

{

if (studentai[i].koksUgis() == auksciausioAmzius(studentai))

{

Console.WriteLine("Auksciausio studento amzius: {0,6:d}", studentai[i].koksAmzius());

}

}

for (int j = 0; j < studentai.Length; j++)

{

if (studentai[j].koksAmzius() == jauniausioUgis(studentai))

{

Console.WriteLine("Jauniausio studento ugis: {0,6:f2}", studentai[j].koksUgis());

}

}

Console.WriteLine("");

int visuStudentuSvoris = svoris(studentai);

int kiekStudentu = studentai.Length;

int kiekKartuKelsimes;

kiekKartuKelsimes = kiekKartuReiksKelti(kiekStudentu, studentai, l);

Console.WriteLine("Vidutinis studentu svoris: {0,4:d}", visuStudentuSvoris / kiekStudentu);

Console.WriteLine("");

int kiekKilimu = kiekKilsim(l, studentai, visuStudentuSvoris, kiekKartuKelsimes);

Console.WriteLine("Studentai pakils per: {0,5:d} k.", kiekKilimu);

Console.WriteLine("");

l.DidintiTalpa(2);

kiekKilimu = kiekKilsim(l, studentai, visuStudentuSvoris, kiekKartuKelsimes);

Console.WriteLine("Padidinus talpa studentai pakils per: {0,5:d} k.", kiekKilimu);

l.MazintiTalpa(2); // Graziname talpa i pradine verte, kad nepasikeistu galios skaiciavimas

l.DidintiGalia(2);

kiekKilimu = kiekKilsim(l, studentai, visuStudentuSvoris, kiekKartuKelsimes);

Console.WriteLine("Padidinus galia studentai pakils per: {0,5:d} k.", kiekKilimu);

Console.WriteLine("");

}

static int kiekKartuReiksKelti(int kiekStudentu, Studentas[] studentai, Liftas l)

{

int kiekKartuReiksKelti;

if (kiekStudentu % l.kokiaTalpa() == 0)

{

kiekKartuReiksKelti = kiekStudentu / l.kokiaTalpa();

}

else

{

kiekKartuReiksKelti = kiekStudentu / l.kokiaTalpa() + 1;

}

return kiekKartuReiksKelti;

}

static int kiekKilsim(Liftas l, Studentas[] studentai, int visuStudentuSvoris, int x)

{

int kiekKilimu = 0;

if (x \* l.kokiaGalia() > visuStudentuSvoris)

{

kiekKilimu = x;

}

else

{

kiekKilimu = x + 1;

}

return kiekKilimu;

}

static int jauniausioUgis(Studentas[] studentai)

{

int jauniausioAmzius = studentai[0].koksAmzius();

for (int i = 1; i < studentai.Length; i++)

{

if (studentai[i].koksAmzius() < jauniausioAmzius)

{

jauniausioAmzius = studentai[i].koksAmzius();

}

}

return jauniausioAmzius;

}

static int svoris(Studentas[] studentai)

{

int visuSvoris = 0;

for (int i = 0; i < studentai.Length; i++)

{

visuSvoris += studentai[i].koksSvoris();

}

return visuSvoris;

}

static double auksciausioAmzius(Studentas[] studentai)

{

double auksciausioUgis = studentai[0].koksUgis();

for (int i = 1; i < studentai.Length; i++)

{

if (studentai[i].koksUgis() > auksciausioUgis)

{

auksciausioUgis = studentai[i].koksUgis();

}

}

return auksciausioUgis;

}

}

}

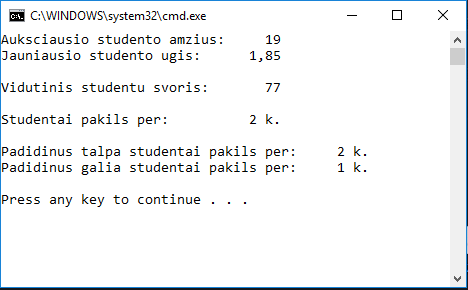
## Pradiniai duomenys ir rezultatai

Pirmi pradiniai duomenys:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Studentas1 | Studentas2 | Studentas3 |
| Amžius | 20 | 19 | 21 |
| Ūgis | 1.78 | 1.85 | 1.79 |
| Svoris | 76 | 82 | 73 |

|  |  |
| --- | --- |
|  | Liftas1 |
| Keliamoji galia | 150 |
| Talpa | 3 |

Rezultatai su pirmaisiais duomenimis:

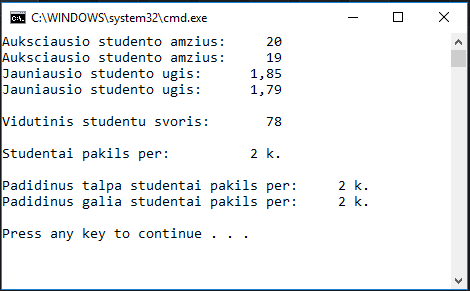


Antri pradiniai duomenys:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Studentas1 | Studentas2 | Studentas3 |
| Amžius | 20 | 19 | 19 |
| Ūgis | 1.85 | 1.85 | 1.79 |
| Svoris | 75 | 83 | 76 |

|  |  |
| --- | --- |
|  | Liftas1 |
| Keliamoji galia | 175 |
| Talpa | 2 |

Rezultatai su antraisiais duomenimis:



# Objektų rinkinys

## Darbo užduotis

**U3–2. Krepšinis**

* Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Sukurkite klasę Krepšininkas, kuri turėtų kintamuosius vardui su pavarde, amžiui bei ūgiui saugoti. Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis.
* Papildykite programą veiksmais su dviejų krepšinio mokyklų duomenimis. Kiekvienos mokyklos duomenys saugomi atskiruose failuose. Kurioje mokykloje aukščiausias sportininkas? Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace LD\_2\_Krepsinis\_Augustas\_Macijauskas

{

class Krepsininkas

{

private string vp;

private int amzius;

private double ugis;

public Krepsininkas(string vp, int amzius, double ugis)

{

this.vp = vp;

this.amzius = amzius;

this.ugis = ugis;

}

public string GautiVarda() { return vp; }

public int GautiAmziu() { return amzius; }

public double GautiUgi() { return ugis; }

}

class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

const int maxDuom = 500;

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

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

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

const string virsus = "|----------------------------------------------------------------|\r\n" +

"| | | |\r\n" +

"| Krepšininkas | Amžius | Ūgis |\r\n" +

"| | | |\r\n" +

"|----------------------------------------------------------------|";

Krepsininkas[] pirma = new Krepsininkas[maxDuom];

Krepsininkas[] antra = new Krepsininkas[maxDuom];

Krepsininkas[] auksciausi = new Krepsininkas[maxDuom];

Krepsininkas[] filtras = new Krepsininkas[maxDuom];

string pav1, pav2;

int n1, n2;

skaitymas(duom1, pirma, out pav1, out n1);

skaitymas(duom2, antra, out pav2, out n2);

if (File.Exists(rez))

File.Delete(rez);

Spausdinimas(rez, pirma, n1, pav1, virsus);

Spausdinimas(rez, antra, n2, pav2, virsus);

//Darbas su amziumi

double amziausVidurkis1, amziausVidurkis2, bendrasAmziausVidurkis;

amziausVidurkis1 = amziausVidurkis(pirma, n1);

amziausVidurkis2 = amziausVidurkis(antra, n2);

bendrasAmziausVidurkis = (amziausVidurkis1 + amziausVidurkis2) / 2;

pridetiTeksto(rez, "Krepsininku amziaus vidurkis:");

spausdintiVidurkius(rez, pav1, amziausVidurkis1, "metu(-ai)");

spausdintiVidurkius(rez, pav2, amziausVidurkis2, "metu(-ai)");

pridetiTeksto(rez, "");

spausdintiVidurkius(rez, "Bendras", bendrasAmziausVidurkis, "metu(-ai)");

pridetiTeksto(rez, "");

//Darbas su ugiais

double ugioVidurkis1, ugioVidurkis2, bendrasUgioVidurkis;

ugioVidurkis1 = ugioVidurkis(pirma, n1);

ugioVidurkis2 = ugioVidurkis(antra, n2);

bendrasUgioVidurkis = (ugioVidurkis1 + ugioVidurkis2) / 2;

pridetiTeksto(rez, "Krepsininku ugio vidurkis:");

spausdintiVidurkius(rez, pav1, ugioVidurkis1, "m");

spausdintiVidurkius(rez, pav2, ugioVidurkis2, "m");

pridetiTeksto(rez, "");

spausdintiVidurkius(rez, "Bendras", bendrasUgioVidurkis, "m");

pridetiTeksto(rez, "");

//Auksciausio ieskojimas

double auksciausias1, auksciausias2;

auksciausias1 = rastiAuksciausia(pirma, n1);

auksciausias2 = rastiAuksciausia(antra, n2);

pridetiTeksto(rez, "Auksciausias sportininkas mokosi:");

if (auksciausias1 == auksciausias2)

{

spausdintiAuksciausius(rez, pirma, n1, auksciausias1, pav1);

spausdintiAuksciausius(rez, antra, n2, auksciausias2, pav2);

}

else if (auksciausias1 > auksciausias2)

{

spausdintiAuksciausius(rez, pirma, n1, auksciausias1, pav1);

}

else

{

spausdintiAuksciausius(rez, antra, n2, auksciausias2, pav2);

}

pridetiTeksto(rez, "");

//Ugis didesnis uz vidurki

int kiekis = 0;

pridetiTeksto(rez, "Naujas rinkinys (krepsininkai, kuriu ugis didesnis uz vidurki):");

ugisDidesnisUzVidurki(pirma, auksciausi, n1, kiekis, ref kiekis, bendrasUgioVidurkis);

ugisDidesnisUzVidurki(antra, auksciausi, n2, kiekis, ref kiekis, bendrasUgioVidurkis);

Spausdinimas(rez, auksciausi, kiekis, "", virsus);

//Jauniausi krepsininkai

int jauniausias1, jauniausias2;

jauniausias1 = jauniausioAmzius(pirma, n1);

jauniausias2 = jauniausioAmzius(antra, n2);

int sk = 0;

pridetiTeksto(rez, "Jauniausi krepsininkai:");

if (jauniausias1 == jauniausias2)

{

pridetiTeksto(rez, "Krepsininkai, kuriu amzius yra " + jauniausias1 + " m.");

rastiJauniausius(pirma, n1, filtras, ref sk, jauniausias1);

Spausdinimas(rez, filtras, sk, pav1, virsus);

sk = 0;

rastiJauniausius(antra, n2, filtras, ref sk, jauniausias2);

Spausdinimas(rez, filtras, sk, pav2, virsus);

}

else if (jauniausias1 < jauniausias2)

{

pridetiTeksto(rez, "Krepsininkai, kuriu amzius yra " + jauniausias1 + " m.");

rastiJauniausius(pirma, n1, filtras, ref sk, jauniausias1);

Spausdinimas(rez, filtras, sk, pav1, virsus);

}

else if (jauniausias2 < jauniausias1)

{

pridetiTeksto(rez, "Krepsininkai, kuriu amzius yra " + jauniausias2 + " m.");

rastiJauniausius(antra, n2, filtras, ref sk, jauniausias2);

Spausdinimas(rez, filtras, sk, pav2, virsus);

}

Console.WriteLine("Programa baige darba!");

}

static void rastiJauniausius(Krepsininkas[] krepsininkai, int n, Krepsininkas[] filtras, ref int sk, int jauniausias)

{

for (int i = 0; i < n; i++) {

if (krepsininkai[i].GautiAmziu() == jauniausias) {

filtras[sk] = new Krepsininkas(krepsininkai[i].GautiVarda(), krepsininkai[i].GautiAmziu(), krepsininkai[i].GautiUgi());

sk++;

}

}

}

static void spausdintiAuksciausius(string rez, Krepsininkas[] krepsininkai, int n, double auksciausias, string pav)

{

using (var prideti = File.AppendText(rez))

{

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

{

if (krepsininkai[i].GautiUgi() == auksciausias)

prideti.WriteLine(pav);

}

}

}

static void spausdintiVidurkius(string rez, string pav, double vidurkis, string txt)

{

using (var prideti = File.AppendText(rez))

{

prideti.WriteLine("{0}: {1,5:f} {2}", pav, vidurkis, txt);

}

}

static void pridetiTeksto(string rez, string tekstas)

{

using (var prideti = File.AppendText(rez))

{

prideti.WriteLine(tekstas);

}

}

static void Spausdinimas(string rez, Krepsininkas[] krepsininkai, int n, string pav, string virsus)

{

using (var prideti = File.AppendText(rez))

{

if (n > 0) {

prideti.WriteLine(pav);

prideti.WriteLine(virsus);

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

{

prideti.WriteLine("| {0,-39}| {1} | {2} |", krepsininkai[i].GautiVarda(), krepsininkai[i].GautiAmziu(), krepsininkai[i].GautiUgi());

}

prideti.WriteLine("------------------------------------------------------------------");

prideti.WriteLine("");

}

else {

prideti.WriteLine("Sarasas tuscias");

}

}

}

static int jauniausioAmzius(Krepsininkas[] krepsininkai, int n)

{

int jauniausias = krepsininkai[0].GautiAmziu();

for (int i = 1; i < n; i++)

{

if (krepsininkai[i].GautiAmziu() < jauniausias)

{

jauniausias = krepsininkai[i].GautiAmziu();

}

}

return jauniausias;

}

static void ugisDidesnisUzVidurki(Krepsininkas[] krepsininkai, Krepsininkas[] auksciausi, int n, int ilgis, ref int kiekis, double bendrasUgioVidurkis)

{

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

{

if (krepsininkai[i].GautiUgi() > bendrasUgioVidurkis)

{

auksciausi[ilgis] = new Krepsininkas(krepsininkai[i].GautiVarda(), krepsininkai[i].GautiAmziu(), krepsininkai[i].GautiUgi());

ilgis++;

}

}

kiekis = ilgis;

}

static double rastiAuksciausia(Krepsininkas[] krepsininkai, int n)

{

double auksciausias = krepsininkai[0].GautiUgi();

for (int i = 1; i < n; i++)

{

if (krepsininkai[i].GautiUgi() > auksciausias)

auksciausias = krepsininkai[i].GautiUgi();

}

return auksciausias;

}

static double ugioVidurkis(Krepsininkas[] krepsininkai, int n)

{

double ugiuSuma = 0;

double ugioVidurkis;

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

{

ugiuSuma += krepsininkai[i].GautiUgi();

}

ugioVidurkis = ugiuSuma / n;

return ugioVidurkis;

}

static double amziausVidurkis(Krepsininkas[] krepsininkai, int n)

{

double amziuSuma = 0, amziausVidurkis;

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

{

amziuSuma += krepsininkai[i].GautiAmziu();

}

return amziausVidurkis = amziuSuma / n;

}

static void skaitymas(string duom, Krepsininkas[] krepsininkai, out string pav, out int n)

{

using (StreamReader reader = new StreamReader(duom))

{

string eilute;

string[] skaidymas;

pav = reader.ReadLine();

string vp;

int amzius;

double ugis;

int iterator = 0;

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

{

skaidymas = eilute.Split(';');

vp = skaidymas[0];

amzius = int.Parse(skaidymas[1]);

ugis = double.Parse(skaidymas[2]);

krepsininkai[iterator] = new Krepsininkas(vp, amzius, ugis);

iterator++;

}

n = iterator;

}

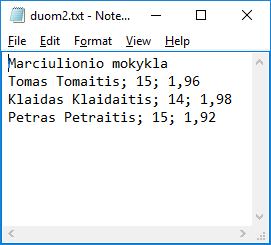
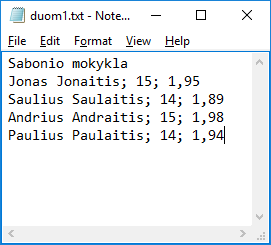
}

}

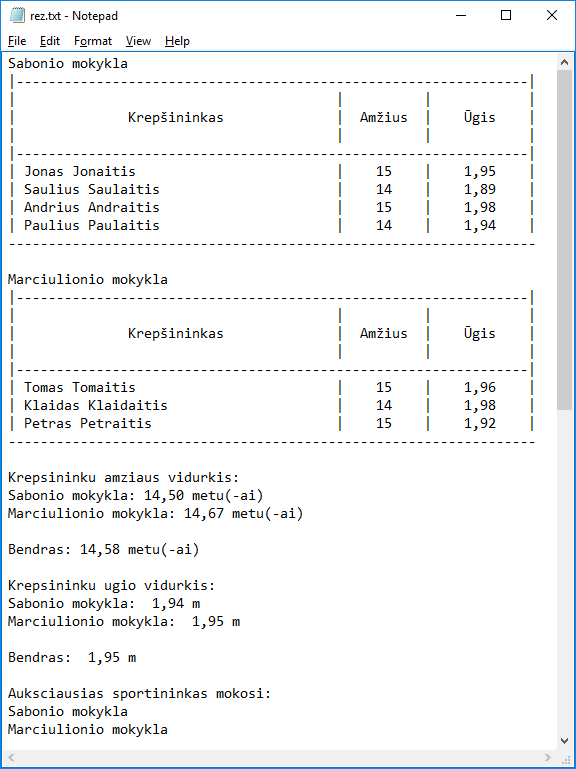
}

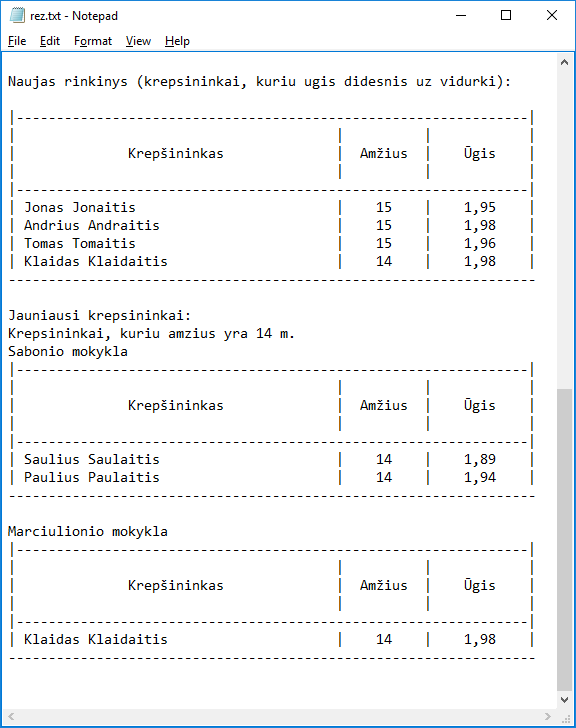
## Pradiniai duomenys ir rezultatai

Pradiniai duomenys Nr. 1:

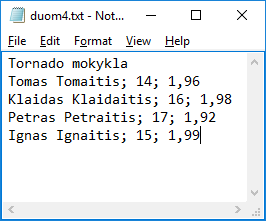
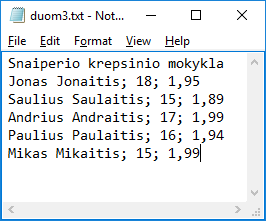


Rezultatai Nr. 1:

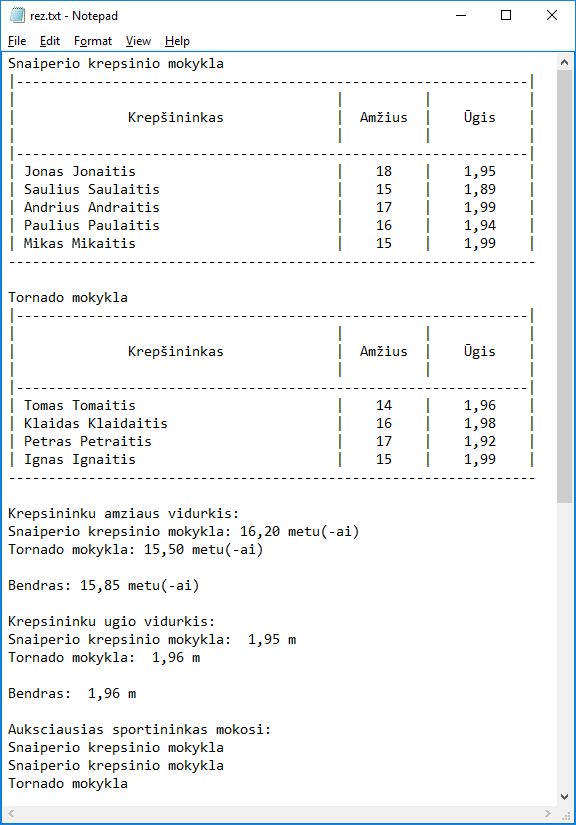


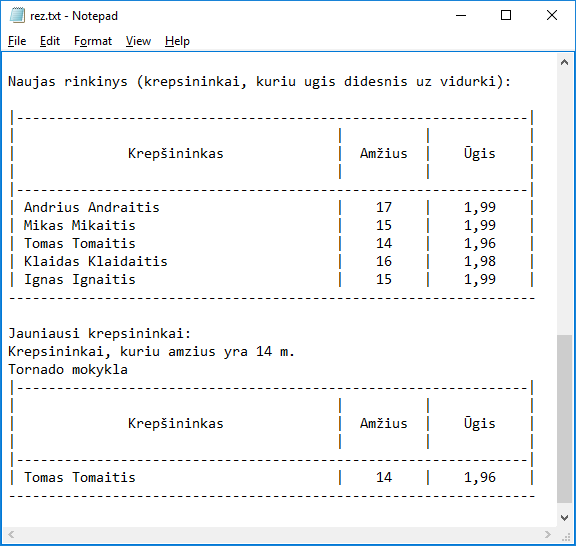


Pradiniai duomenys Nr. 2:



Rezultatai Nr. 2:





# Konteinerinė klasė

## Darbo užduotis

**U4-2. Mobiliojo ryšio kortelės**

Norėdamas palyginti mobiliojo ryšio operatorių siūlomas išankstinio mokėjimo korteles Sirvydas surinko šią informaciją į tekstinį failą. Faile eilutėmis yra kortelių duomenys: kortelės (tinklo) pavadinimas, pradinė suma kortelėje, tarifas savame tinkle, tarifas į kitus tinklus, SMS žinučių tarifas savame tinkle ir į kitus tinklus. Parašykite programą, kuri spausdintų kortelių duomenis lentele, surastų kortelę, kurios SMS žinučių tarifai į kitus tinklus mažiausi. Papildykite programą veiksmais, kurie leistų atrinkti korteles, kurios leidžia skambinti ir siųsti SMS žinutes savame tinkle nemokamai, ir šį sąrašą surikiuoti pagal pradinę sumą mažėjimo tvarka ir kortelės pavadinimą abėcėliškai.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace LD3

{

class Operatorius

{

private string pavadinimas;

private double pradSum, tarifSav, tarifKit, smsSav, smsKit;

public Operatorius(string pavadinimas, double pradSum, double tarifSav, double tarifKit, double smsSav, double smsKit)

{

this.pavadinimas = pavadinimas;

this.pradSum = pradSum;

this.tarifSav = tarifSav;

this.tarifKit = tarifKit;

this.smsSav = smsSav;

this.smsKit = smsKit;

}

public override string ToString()

{

string eilute;

eilute = string.Format(" {0, -12} {1, 6:f} {2, 6:f} {3, 6:f} {4, 6:f} {5, 6:f} ", pavadinimas, pradSum, tarifSav, tarifKit, smsSav, smsKit);

return eilute;

}

public static bool operator <=(Operatorius op1, Operatorius op2)

{

int p = String.Compare(op1.pavadinimas, op2.pavadinimas, StringComparison.CurrentCulture);

return (op1.pradSum > op2.pradSum || (op1.pradSum == op2.pradSum && p < 0));

}

public static bool operator >=(Operatorius op1, Operatorius op2)

{

int p = String.Compare(op1.pavadinimas, op2.pavadinimas, StringComparison.CurrentCulture);

return (op1.pradSum < op2.pradSum || (op1.pradSum == op2.pradSum && p > 0));

}

public string KoksPavadinimas() { return pavadinimas; }

public double KokiaPradineSuma() { return pradSum; }

public double KoksTarifasISavusTinklus() { return tarifSav; }

public double KoksTarifasIKitusTinklus() { return tarifKit; }

public double SMSiSavus() { return smsSav; }

public double SMSiKitus() { return smsKit; }

}

class Korteles

{

const int CMax = 100;

private Operatorius[] Op;

private int n;

public Korteles()

{

n = 0;

Op = new Operatorius[CMax];

}

public int Max() { return CMax; }

public int Imti() { return n; }

public Operatorius Imti(int i) { return Op[i]; }

public void Dėti(Operatorius ob) { Op[n++] = ob; }

public void Sort()

{

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

{

Operatorius min = Op[i];

int mazindex = i;

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

if (Op[j] <= min)

{

min = Op[j];

mazindex = j;

}

Op[mazindex] = Op[i];

Op[i] = min;

}

}

}

class Program

{

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

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

static void Main(string[] args)

{

Korteles korteles = new Korteles();

Skaityti(duom, ref korteles);

if (File.Exists(rez))

File.Delete(rez);

Spausdinti(rez, korteles, "Pradiniai duomenys:");

double maziausiaKaina = MaziausiosSMSKainos(korteles);

SpausdintiMaziausiosSMSKainos(rez, korteles, "SMS tarifai mažiausi:", maziausiaKaina);

Korteles naujas = new Korteles();

Nemokamai(korteles, naujas);

Spausdinti(rez, naujas, "Paliktos tik tos kortelės, su kuriomis į savus tinklus galima skambinti ir rašyti nemokamai:");

if (naujas.Imti() > 0)

{

naujas.Sort();

Spausdinti(rez, naujas, "Surikiuotos kortelės:");

}

Console.WriteLine("Programa baige darba!");

}

static void Nemokamai(Korteles korteles, Korteles naujas)

{

for (int i = 0; i < korteles.Imti(); i++)

{

if (korteles.Imti(i).KoksTarifasISavusTinklus() == 0 && korteles.Imti(i).SMSiSavus() == 0)

naujas.Dėti(korteles.Imti(i));

}

}

static void SpausdintiMaziausiosSMSKainos(string rez, Korteles korteles, string antraste, double mazK)

{

string virsus =

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

" Pavadinimas Pradinė suma Tarifas į savus Tarifas į kitus SMS į savus SMS į kitus \r\n" +

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

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

{

fr.WriteLine(antraste);

fr.WriteLine(virsus);

for (int i = 0; i < korteles.Imti(); i++)

{

if (korteles.Imti(i).SMSiKitus() == mazK)

fr.WriteLine("{0}", korteles.Imti(i).ToString());

}

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

}

}

static double MaziausiosSMSKainos(Korteles korteles)

{

double maziausiaKaina = korteles.Imti(0).SMSiKitus();

for (int i = 1; i < korteles.Imti(); i++)

{

if (korteles.Imti(i).SMSiKitus() < maziausiaKaina)

maziausiaKaina = korteles.Imti(i).SMSiKitus();

}

return maziausiaKaina;

}

static void Skaityti(string duom, ref Korteles korteles)

{

using (StreamReader reader = new StreamReader(duom))

{

string pav;

double prad, t1, t2, sms1, sms2;

string eilute;

string[] skaidymas;

while ((eilute = reader.ReadLine()) != null && korteles.Imti() < korteles.Max())

{

skaidymas = eilute.Split(';');

pav = skaidymas[0].Trim();

prad = double.Parse(skaidymas[1].Trim());

t1 = double.Parse(skaidymas[2].Trim());

t2 = double.Parse(skaidymas[3].Trim());

sms1 = double.Parse(skaidymas[4].Trim());

sms2 = double.Parse(skaidymas[5].Trim());

Operatorius op = new Operatorius(pav, prad, t1, t2, sms1, sms2);

korteles.Dėti(op);

}

}

}

static void Spausdinti(string rez, Korteles korteles, string antraste)

{

string virsus =

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

" Pavadinimas Pradinė suma Tarifas į savus Tarifas į kitus SMS į savus SMS į kitus \r\n" +

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

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

{

fr.WriteLine(antraste);

if (korteles.Imti() > 0)

{

fr.WriteLine(virsus);

for (int i = 0; i < korteles.Imti(); i++)

fr.WriteLine("{0}", korteles.Imti(i).ToString());

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

}

else

fr.WriteLine("Sąrašas tuščias");

}

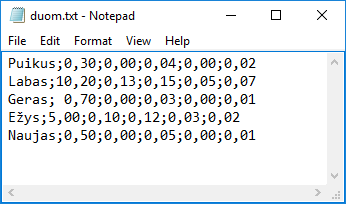
}

}

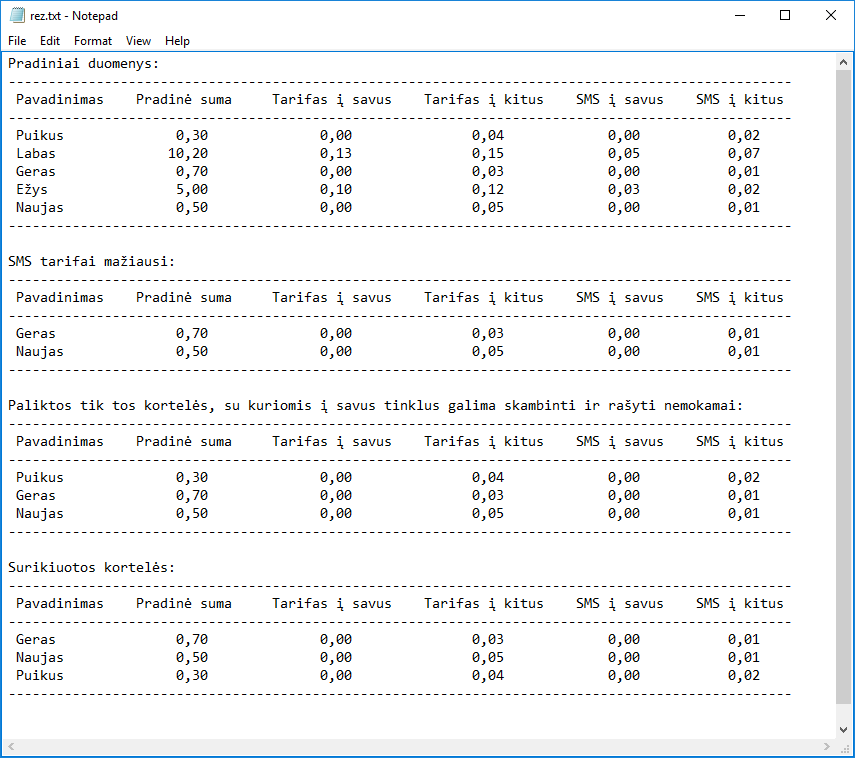
}

## Pradiniai duomenys ir rezultatai

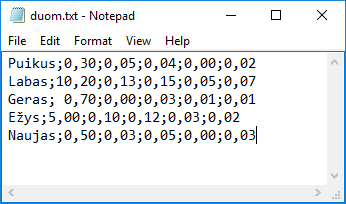
Pradiniai duomenys nr.1:



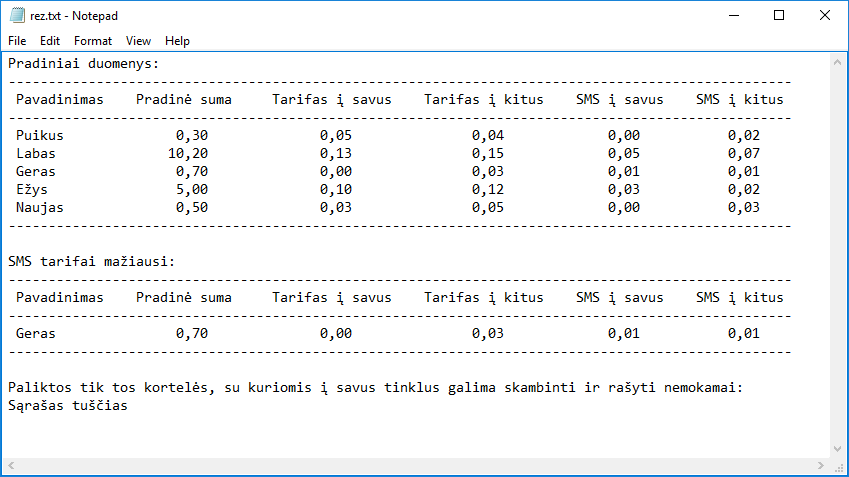
Rezultatai nr.1:



Pradiniai duomenys nr.2:



Rezultatai nr.2:



# Teksto analizė ir redagavimas

## Darbo užduotis

**U5-2. Nelyginis žodžių skaičius**

Tekstiniame faile pateikiamas tekstas. Žodžiai iš eilutės į kitą eilutę nekeliami. Žodžiai eilutėse skiriami bent vienu tarpu. Tarpai gali būti eilutės pradžioje bei gale, gali būti tuščios eilutės. Eilutėse, kuriose yra nelyginis žodžių skaičius n, n / 2 + 1 žodį pakeisti žodžiu „xxooxx“.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace LD4

{

class Program

{

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

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

static void Main(string[] args)

{

Console.InputEncoding = Encoding.Unicode;

Console.OutputEncoding = Encoding.UTF8;

if (File.Exists(rez))

File.Delete(rez);

string tustiSimboliai = "-";

Apdoroti(duom, rez, tustiSimboliai);

Console.WriteLine("\nPrograma baigė darbą!");

}

static void Apdoroti(string duom, string rez, string tustiSimboliai)

{

string[] lines = File.ReadAllLines(duom, Encoding.GetEncoding(1257));

using (var fw = File.AppendText(rez))

{

foreach(string line in lines)

{

if (line != null)

{

string prideti = "xxooxx";

StringBuilder spausdinimui = new StringBuilder();

string[] parts = line.Split(" -,.;:?!\n\t".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

if (parts.Length % 2 == 1)

{

int ind = line.IndexOf(parts[(parts.Length / 2)]);

int i;

for (i = 0; i < line.Length; i++)

{

if (i == ind)

break;

spausdinimui.Append(line[i]);

}

spausdinimui.Append(prideti);

spausdinimui.Append(line.Substring(i + parts[(parts.Length / 2)].Length));

Console.WriteLine(spausdinimui);

fw.WriteLine(spausdinimui);

}

else

{

Console.WriteLine(line);

fw.WriteLine(line);

}

}

}

}

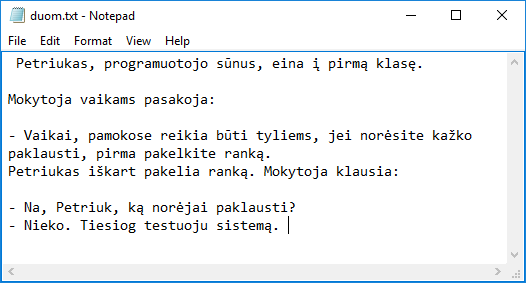
}

}

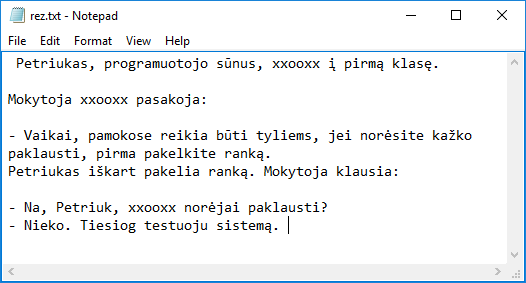
}

## Pradiniai duomenys ir rezultatai

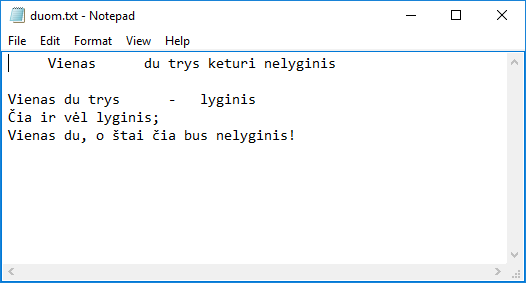
Pradiniai duomenys nr. 1:



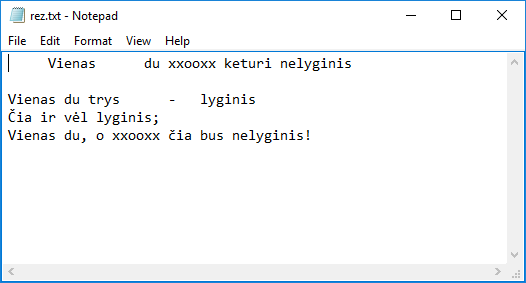
Rezultatai nr. 1:



Pradiniai duomenys nr. 2:



Rezultatai nr. 2:



# Susieti rinkiniai

## Darbo užduotis

**U6–2. Futbolas**

Pirmoje failo eilutėje nurodytas futbolo komandų skaičius. Tolesnėse eilutėse pateikta informacija apie

futbolo komandas: pavadinimas, miestas, trenerio pavardė, vardas. Žemiau pateikta I rato rezultatų

lentelė, išreikšta pelnytais įvarčiais. Suskaičiuokite kiekvienos komandos surinktų taškų skaičių, jei už

pergalę skiriami 3 taškai, o už lygiąsias – 1 taškas. Sudarykite komandų turnyrinę lentelę – surikiuokite

surinktų taškų mažėjimo tvarka. Jei komandos surinko taškų vienodai, aukščiau ta komanda, kuri turi

daugiau pergalių. Suraskite daugiausiai įvarčių pelniusią komandą. Suraskite komandas, kurios

daugiausiai rungtynių nepraleido įvarčių.

## Programos tekstas

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace LD5

{

class Komanda

{

public string pav { get; set; }

public string miestas { get; set; }

public string pavarde { get; set; }

public string vardas { get; set; }

public int pergales { get; set; }

public int pralaimejimai { get; set; }

public int lygiosios { get; set; }

public int imusta { get; set; }

public int praleista { get; set; }

public int svariosrungtynes { get; set; }

public Komanda(string pav, string miestas, string pavarde, string vardas)

{

this.pav = pav;

this.miestas = miestas;

this.pavarde = pavarde;

this.vardas = vardas;

pergales = 0;

pralaimejimai = 0;

lygiosios = 0;

imusta = 0;

praleista = 0;

svariosrungtynes = 0;

}

public override string ToString()

{

string eil = string.Format(" {0, -15} {1, -15} {2, -15} {3, -15} {4, 2:d}/{5, 2:d}/{6, 2:d} {7, 2:d}/{8, 2:d}/{9, 2:d} {10, 2:d}", pav, miestas, pavarde, vardas, pergales, pralaimejimai, lygiosios, imusta, praleista, svariosrungtynes, KiekTasku());

return eil;

}

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

{

return ((k1.KiekTasku() > k2.KiekTasku()) || ((k1.KiekTasku() == k2.KiekTasku()) && (k1.pergales > k2.pergales)));

}

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

{

return ((k1.KiekTasku() < k2.KiekTasku()) || ((k1.KiekTasku() == k2.KiekTasku()) && (k1.pergales < k2.pergales)));

}

public void SkaiciuotiTaskus(Rezultatai B, int x)

{

pergales = 0;

pralaimejimai = 0;

lygiosios = 0;

imusta = 0;

praleista = 0;

for (int j = 0; j < B.m; j++)

{

if (x != j)

{

imusta += B.ImtiReiksme(x, j);

praleista += B.ImtiReiksme(j, x);

if (B.ImtiReiksme(j, x) == 0)

{

svariosrungtynes++;

}

if (B.ImtiReiksme(x, j) == B.ImtiReiksme(j, x))

{

lygiosios++;

}

else if (B.ImtiReiksme(x, j) > B.ImtiReiksme(j, x))

{

pergales++;

}

else

{

pralaimejimai++;

}

}

}

}

public int KiekTasku()

{

return pergales \* 3 + lygiosios;

}

}

class KomanduKonteineris

{

const int Max = 30;

private Komanda[] A;

private int n;

public KomanduKonteineris()

{

n = 0;

A = new Komanda[Max];

}

public int Imti()

{

return n;

}

public void Dėti(Komanda ob)

{

A[n++] = ob;

}

public Komanda Imti(int i)

{

return A[i];

}

public void Rikiuoti(Rezultatai B)

{

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

{

Komanda pagalb = A[i];

int ind = i;

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

{

if (A[j] <= pagalb)

{

pagalb = A[j];

ind = j;

}

}

A[ind] = A[i];

A[i] = pagalb;

B.Sukeisti(i, ind);

}

}

}

class Rezultatai

{

const int MaxEil = 30;

const int MaxSt = 30;

private int[,] A;

public int n { get; set; }

public int m { get; set; }

public Rezultatai()

{

n = 0;

m = 0;

A = new int[MaxEil, MaxSt];

}

public void Dėti(int i, int j, int ivarciai)

{

A[i, j] = ivarciai;

}

public int ImtiReiksme(int i, int j)

{

return A[i, j];

}

public void Sukeisti(int k, int l)

{

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

{

int pagalb = A[k, j];

A[k, j] = A[l, j];

A[l, j] = pagalb;

}

}

}

class Program

{

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

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

static void Main(string[] args)

{

KomanduKonteineris komandos = new KomanduKonteineris();

Rezultatai rezultatai = new Rezultatai();

if (File.Exists(rez))

File.Delete(rez);

Skaityti(duom, komandos, rezultatai);

Spausdinti(rez, komandos, "Pradiniai duomenys:");

SpausdintiRezultatus(rez, rezultatai, "Turnyrinė lentelė:");

KomanduKonteineris pagalb = new KomanduKonteineris();

int iv = DaugiausiaiPelnytuIvarciu(komandos);

FormuotiIv(komandos, pagalb, iv);

Spausdinti(rez, pagalb, "Daugiausiai įvarčių:");

pagalb = new KomanduKonteineris();

int r = DaugiausiaiRungtyniuNepraleista(komandos);

if (r > 0)

{

FormuotiRung(komandos, pagalb, r);

Spausdinti(rez, pagalb, "Daugiausiai rungtynių nepraleido įvarčių:");

}

else

{

Print(rez, "Komandų, kurios nepraleido įvarčių, nebuvo\n");

}

komandos.Rikiuoti(rezultatai);

Spausdinti(rez, komandos, "Surikiuoti duomenys:");

SpausdintiRezultatus(rez, rezultatai, "Turnyrinė lentelė:");

Console.WriteLine("Programa baigė darbą!");

}

static void Print(string rez, string x)

{

using (var fw = File.AppendText(rez))

{

fw.WriteLine(x);

}

}

static void FormuotiRung(KomanduKonteineris senas, KomanduKonteineris naujas, int k)

{

for (int i = 0; i < senas.Imti(); i++)

{

if (senas.Imti(i).svariosrungtynes == k)

{

naujas.Dėti(senas.Imti(i));

}

}

}

static void FormuotiIv(KomanduKonteineris senas, KomanduKonteineris naujas, int k)

{

for (int i = 0; i < senas.Imti(); i++)

{

if (senas.Imti(i).imusta == k)

{

naujas.Dėti(senas.Imti(i));

}

}

}

static int DaugiausiaiRungtyniuNepraleista(KomanduKonteineris A)

{

int r = 0;

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

{

if (A.Imti(i).svariosrungtynes > r)

{

r = A.Imti(i).svariosrungtynes;

}

}

return r;

}

static int DaugiausiaiPelnytuIvarciu(KomanduKonteineris A)

{

int iv = 0;

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

{

if (A.Imti(i).imusta > iv)

{

iv = A.Imti(i).imusta;

}

}

return iv;

}

static void Skaityti(string duom, KomanduKonteineris A, Rezultatai B)

{

using (StreamReader reader = new StreamReader(duom))

{

string line;

string pav, miestas, pavarde, vardas;

int kiek = int.Parse(reader.ReadLine());

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

{

line = reader.ReadLine();

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

pav = parts[0];

miestas = parts[1];

pavarde = parts[2];

vardas = parts[3];

Komanda nauja = new Komanda(pav, miestas, pavarde, vardas);

A.Dėti(nauja);

}

B.n = kiek;

B.m = kiek;

for (int i = 0; i < B.n; i++)

{

line = reader.ReadLine();

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

for (int j = 0; j < B.m; j++)

{

int sk = int.Parse(parts[j]);

B.Dėti(i, j, sk);

}

}

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

{

A.Imti(i).SkaiciuotiTaskus(B, i);

}

}

}

static void SpausdintiRezultatus(string rez, Rezultatai B, string antraste)

{

using (var fw = File.AppendText(rez))

{

fw.WriteLine("Turnyrinė lentelė: ");

for (int i = 0; i < B.n; i++)

{

for (int j = 0; j < B.m; j++)

{

fw.Write(B.ImtiReiksme(i, j) + " ");

}

fw.WriteLine();

}

fw.WriteLine();

}

}

static void Spausdinti(string rez, KomanduKonteineris A, string antraste)

{

using (var fw = File.AppendText(rez))

{

string virsus = "---------------------------------------------------------------------------------------------------------------------------------------------------------------------------\r\n" +

" Nr. Komandos pavadinimas Miestas Trenerio pavardė Trenerio vardas Perg./Pral./Lyg. Įmušta/Praleista/Kiek rungtynių nepraleido Taškai \r\n" +

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

if (A.Imti() > 0)

{

fw.WriteLine(antraste);

fw.WriteLine(virsus);

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

{

fw.WriteLine(" {0,2:d} {1} ", i + 1, A.Imti(i).ToString());

}

fw.WriteLine("---------------------------------------------------------------------------------------------------------------------------------------------------------------------------");

fw.WriteLine();

}

}

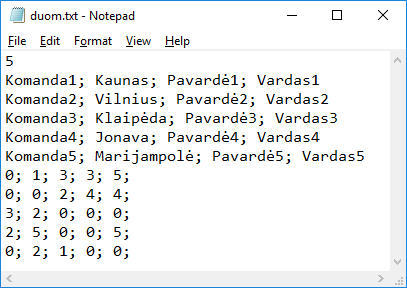
}

}

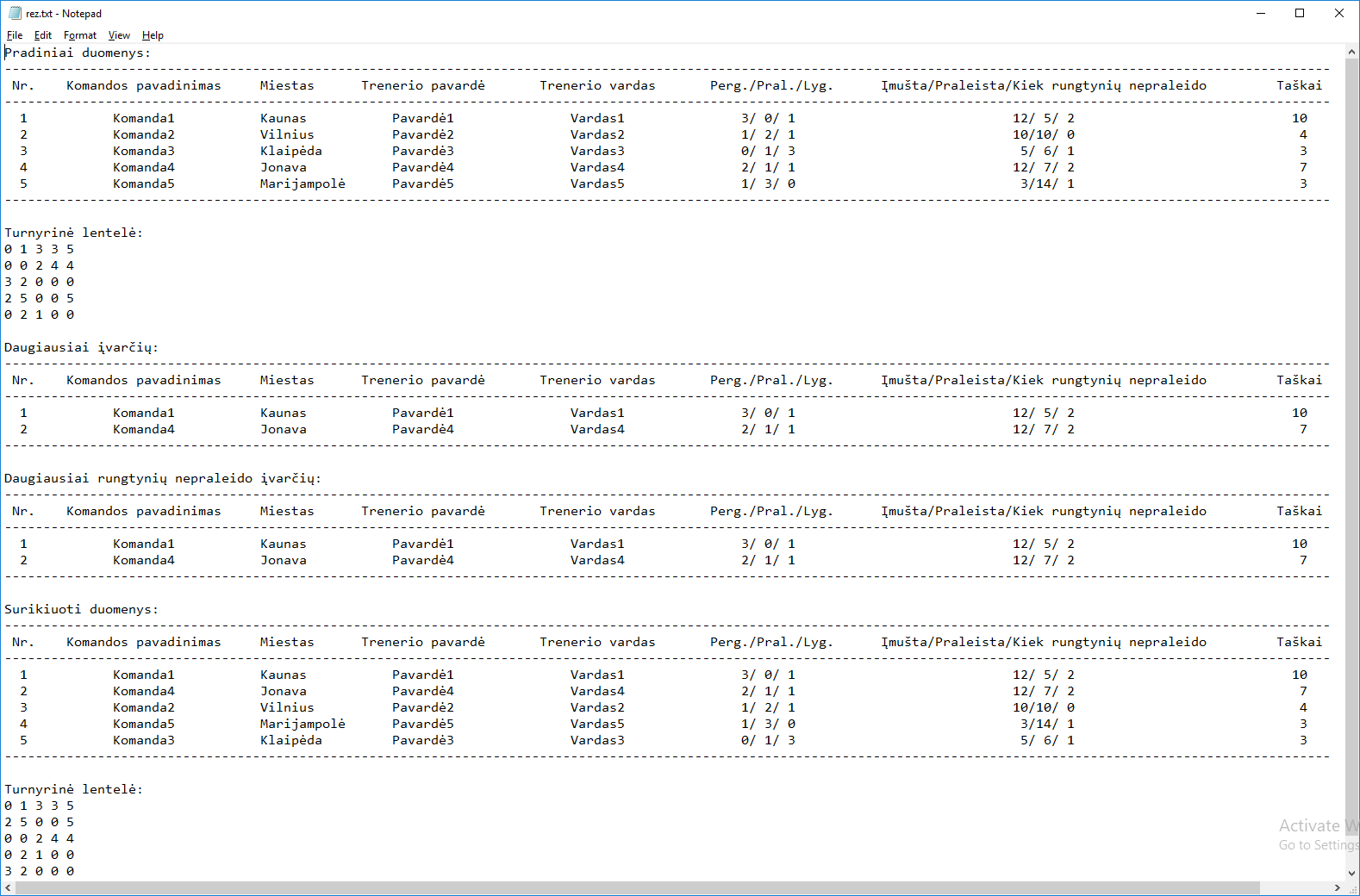
}

## Pradiniai duomenys ir rezultatai

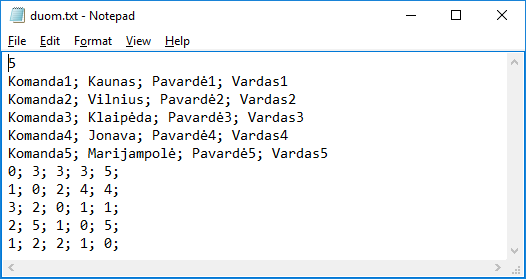
Pradiniai duomenys nr.1:



Rezultatai nr.1:



Pradiniai duomenys nr.2:



Rezultatai: nr.2:

