**KAUNO TECHNOLOGIJOS UNIVERSITETO GIMNAZIJA**

**OBJEKTINIO PROGRAMAVIMO PAGRINDAI I (P175B117)**

***Laboratorinių darbų ataskaita***

Atliko:

3 Bentley klasės mokinys

Arminas Tomaševičius

2016 m. rugsėjo 29 d.

Priėmė:

Kęstutis Simonavičius

**KAUNAS 2016**

**TURINYS**

**1.** **Pažintis su klase 3**

1.1. Darbo užduotis 3

1.2. Programos tekstas 3

1.3. Pradiniai duomenys ir rezultatai 3

**2.** **Objektų rinkinys 4**

2.1. Darbo užduotis 4

2.2. Programos tekstas 4

2.3. Pradiniai duomenys ir rezultatai 4

**3.** **Konteinerinė klasė 4**

3.1. Darbo užduotis 4

3.2. Programos tekstas 4

3.3. Pradiniai duomenys ir rezultatai 4

**4.** **Teksto analizė ir redagavimas 4**

4.1. Darbo užduotis 4

4.2. Programos tekstas 4

4.3. Pradiniai duomenys ir rezultatai 4

**5.** **Susieti rinkiniai 4**

5.1. Darbo užduotis 4

5.2. Programos tekstas 4

5.3. Pradiniai duomenys ir rezultatai 4

* **Pažintis su klase**
* ***Darbo užduotis***

U2–23. Kepykla.

* Sukurkite klasę Duona, kuri turėtų kintamuosius duonos miltų pavadinimui, kepaliuko svoriui ir kainai saugoti. Kepykla kepa duoną iš 3 skirtingų rūšių miltų. Raskite, kuri kepama duona mažiausiai sveria ir kurios duonos kaina didžiausia.
* Papildykite klasę Duona kintamuoju, skirtu duonos kepaliuko užimamam ant lentynos plotui saugoti. Sukurkite klasę Kepykla, kuri turėtų kintamąjį kepyklos pavadinimui saugoti ir 3 kintamuosius (kiekvienai duonos miltų rūšiai), skirtus saugoti per pamainą reikalingam iškepti duonos kepaliukų kiekiui. Suskaičiuokite, kiek ploto lentynų kiekvienos rūšies duonai sudėti reikės.
* Papildykite klasę Kepykla kintamuoju, kuriame būtų saugoma informacija apie tai, kiek kg duonos kepyklos automobilis gali vežti. Kiek kartų reikės važiuoti kepyklos automobiliui, kad išvežti į parduotuves visą per pamainą iškeptą duoną?
* ***Programos tekstas***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace U\_23

{

class Duona

{

private string name;

private double mass;

private double cost;

private double area;

public Duona(string name, double mass, double cost, double area)

{

this.name = name;

this.mass = mass;

this.cost = cost;

this.area = area;

}

public string Name

{

set { name = value; }

get { return name; }

}

public double Mass

{

set { mass = value; }

get { return mass; }

}

public double Cost

{

set { cost = value; }

get { return cost; }

}

public double Area

{

set { area = value; }

get { return area; }

}

}

class Kepykla

{

private string name;

private double kiekis\_1;

private double kiekis\_2;

private double kiekis\_3;

private double trans;

public Kepykla(string name, double kiekis\_1, double kiekis\_2, double kiekis\_3, double trans)

{

this.name = name;

this.kiekis\_1 = kiekis\_1;

this.kiekis\_2 = kiekis\_2;

this.kiekis\_3 = kiekis\_3;

this.trans = trans;

}

public string Name

{

set { name = value; }

get { return name; }

}

public double Kiekis\_1

{

set { kiekis\_1 = value; }

get { return kiekis\_1; }

}

public double Kiekis\_2

{

set { kiekis\_2 = value; }

get { return kiekis\_2; }

}

public double Kiekis\_3

{

set { kiekis\_3 = value; }

get { return kiekis\_3; }

}

public double Trans

{

set { trans = value; }

get { return trans; }

}

}

class Program

{

static void Main(string[] args)

{

string name;

string kname;

double mass, cost, area;

int kiekis\_1;

int kiekis\_2;

int kiekis\_3;

int trans;

int[] kiekis = new int[100];

List<Duona> duona = new List<Duona>();

Console.WriteLine("Įveskite 1 duonos miltų pavadinimą: ");

name = (string)Console.ReadLine();

Console.WriteLine("Įveskite 1 duonos masę(g): ");

mass = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 1 duonos kainą: ");

cost = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 1 duonos plotą(cm2): ");

area = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 1 duonos kepaliukų kiekį gaminamą kepyklos per 1 d: ");

kiekis\_1 = int.Parse(Console.ReadLine());

Duona d1 = new Duona(name, mass, cost, area);

duona.Add(d1);

kiekis[0] = kiekis\_1;

Console.WriteLine("Įveskite 2 duonos miltų pavadinimą: ");

name = (string)Console.ReadLine();

Console.WriteLine("Įveskite 2 duonos masę(g): ");

mass = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 2 duonos kainą: ");

cost = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 2 duonos plotą(cm2): ");

area = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 2 duonos kepaliukų kiekį gaminamą kepyklos per 1 d: ");

kiekis\_2 = int.Parse(Console.ReadLine());

Duona d2 = new Duona(name, mass, cost, area);

duona.Add(d2);

kiekis[1] = kiekis\_2;

Console.WriteLine("Įveskite 3 duonos miltų pavadinimą: ");

name = (string)Console.ReadLine();

Console.WriteLine("Įveskite 3 duonos masę(g): ");

mass = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 3 duonos kainą: ");

cost = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 3 duonos plotą(cm2): ");

area = double.Parse(Console.ReadLine());

Console.WriteLine("Įveskite 3 duonos kepaliukų kiekį gaminamą kepyklos per 1 d: ");

kiekis\_3 = int.Parse(Console.ReadLine());

Duona d3 = new Duona(name, mass, cost, area);

duona.Add(d3);

kiekis[2] = kiekis\_3;

Console.WriteLine("Įveskite kepyklos pavadinimą: ");

kname = (string)Console.ReadLine();

Console.WriteLine("Įveskite kepyklos transporto galimą vežti kepaliukų kiekį: ");

trans = int.Parse(Console.ReadLine());

Kepykla k1 = new Kepykla(kname, kiekis\_1, kiekis\_2, kiekis\_3, trans);

Min\_mass(duona);

Max\_cost(duona);

Area(duona, kiekis);

Transportation(k1);

Vid\_mass(duona);

}

public static void Min\_mass(List<Duona> duona)

{

double min\_mass = Double.MaxValue;

foreach (Duona d in duona)

{

if (min\_mass >= d.Mass)

{

min\_mass = d.Mass;

}

}

foreach (Duona d in duona)

{

if (min\_mass == d.Mass)

{

Console.WriteLine("Mažiausiai sveria: {0} {1} g", d.Name, d.Mass);

}

}

}

public static void Max\_cost(List<Duona> duona)

{

double max\_cost = Double.MinValue;

foreach (Duona d in duona)

{

if (max\_cost <= d.Cost)

{

max\_cost = d.Cost;

}

}

foreach (Duona d in duona)

{

if (max\_cost == d.Cost)

{

Console.WriteLine("Didžiausia kaina: {0} {1}", d.Name, d.Cost);

}

}

}

public static void Area(List<Duona> duona, int[] kiekis)

{

int i = 0;

double sum\_area = 0;

foreach (Duona d in duona)

{

double area = 0;

area += d.Area \* kiekis[i];

i++;

Console.WriteLine("{0} rūšies: {1} cm2", i, area);

sum\_area += area;

}

Console.WriteLine("Visi duonos kepaliukai užims: {0} cm2", sum\_area);

}

public static void Transportation(Kepykla k)

{

int times = 0;

if ((k.Kiekis\_1 + k.Kiekis\_2 + k.Kiekis\_3) % k.Trans > 0)

{

times = (int)((k.Kiekis\_1 + k.Kiekis\_2 + k.Kiekis\_3) / k.Trans + 1);

}

else

{

times = (int)((k.Kiekis\_1 + k.Kiekis\_2 + k.Kiekis\_3) / k.Trans);

}

Console.WriteLine("Kepyklos automobiliui reikės važiuoti: {0} kartus", times);

}

public static void Vid\_mass(List<Duona>duona)

{

double vid\_mass = 0;

double sum = 0;

foreach(Duona d in duona)

{

sum += d.Mass;

}

vid\_mass = sum / duona.Count;

Console.WriteLine("Vidutinis kepaliuko svoris: {0}", vid\_mass);

}

}

}

* ***Pradiniai duomenys ir rezultatai***







* **Objektų rinkinys**
* ***Darbo užduotis***

U3–22. Kaliausė

* Duomenys apie ūkininkų pagamintas kaliauses yra faile: tipas, aprėdų skaičius, svoris ir kaina. Pirmoje eilutėje yra ūkininko pavardė ir vardas. Sukurkite klasę Kaliausė vienos kaliausės duomenims saugoti. Kiek aprėdų turi sunkiausia kaliausė? Kokia vidutinė kaliausės kaina?
* Papildykite programą veiksmais su dviejų ūkininkų kaliausių rinkiniais. Kiekvieno rinkinio duomenys saugomi atskiruose failuose. Kurio ūkininko sunkiausia kaliausė daugiau aprėdyta? Kurio ūkininko vidutinė kaliausės kaina mažesnė? Surašykite į atskirą rinkinį visus abiejų ūkininkų nurodyto tipo kaliausių duomenis.
* ***Programos tekstas***

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace U3\_22

{

class Kaliausė

{

private string type;

private int count;

private double mass;

private double cost;

public Kaliausė(string type, int count, double mass, double cost)

{

this.type = type;

this.count = count;

this.mass = mass;

this.cost = cost;

}

public string Type

{

set { type = value; }

get { return type; }

}

public int Count

{

set { count = value; }

get { return count; }

}

public double Mass

{

set { mass = value; }

get { return mass; }

}

public double Cost

{

set { cost = value; }

get { return cost; }

}

}

class Program

{

static void Main(string[] args)

{

string name1, name2;

int kiek1, kiek2;

const string CFd1 = "...[\\...\\Duom1.txt](file://\\...\\Duom1.txt)";

const string CFd2 = "...[\\...\\Duom2.txt](file://\\...\\Duom2.txt)";

const string Rez = "...[\\...\\Rez.txt](file://\\...\\Rez.txt)";

Kaliausė[] K1 = new Kaliausė[100];

Kaliausė[] K2 = new Kaliausė[100];

Kaliausė[] R = new Kaliausė[100];

int[] count1 = new int[100];

int[] count2 = new int[100];

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

Skaityti(CFd1, K1, out kiek1, out name1);

SpausdintiPradiniusDuomenis(Rez, K1, kiek1, name1);

Skaityti(CFd2, K2, out kiek2, out name2);

SpausdintiPradiniusDuomenis(Rez, K2, kiek2, name2);

Count\_HighestMass(K1, kiek1, name1);

Count\_HighestMass(K2, kiek2, name2);

SpausdintiDuomenis(Rez, Vid\_Cost(K1, kiek1), name1);

SpausdintiDuomenis(Rez, Vid\_Cost(K2, kiek2), name2);

count1 = Count\_HighestMass\_Compare(K1, kiek1, count1);

count2 = Count\_HighestMass\_Compare(K2, kiek2, count2);

SpausdintiMax(Rez, count1, count2, name1, name2);

Vid\_Cost\_Lowest(K1, K2, kiek1, kiek2, name1, name2);

Console.WriteLine("Įveskite kaliausių rinkinio tipą");

string x = Console.ReadLine();

int check = 0;

check = Make\_Type\_List(K1, R, kiek1, x, check);

Make\_Type\_List(K2, R, kiek2, x, check);

SpausdintiRinki(Rez, R, check);

}

static void Count\_HighestMass(Kaliausė[] K, int kiek, string name)

{

using (StreamWriter writer = new StreamWriter("...\\...\\Rez.txt", true))

{

writer.WriteLine("Ūkininko {0} sunkiausios kaliausės aprėdų skaičius:", name);

double Max\_mass = 0;

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

{

if (K[i].Mass > Max\_mass)

{

Max\_mass = K[i].Mass;

}

}

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

{

if (K[i].Mass == Max\_mass)

{

writer.WriteLine("Kaliausės tipas {0}, aprėdų skaičius {1}, masė {2}, kaina {3}", K[i].Type, K[i].Count, K[i].Mass, K[i].Cost);

}

}

}

}

static double Vid\_Cost(Kaliausė[] K, int kiek)

{

double sum = 0;

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

{

sum += K[i].Cost;

}

double vid\_cost = sum / kiek;

return vid\_cost;

}

static int[] Count\_HighestMass\_Compare(Kaliausė[] K, int kiek, int[] count)

{

double Max\_mass = 0;

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

{

if (K[i].Mass > Max\_mass)

{

Max\_mass = K[i].Mass;

}

}

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

{

if (K[i].Mass == Max\_mass)

{

count[i] = K[i].Count;

}

}

return count;

}

static void Vid\_Cost\_Lowest(Kaliausė[] K1, Kaliausė[] K2, int kiek1, int kiek2, string name1, string name2)

{

using (StreamWriter writer = new StreamWriter("...\\...\\Rez.txt", true))

{

if (Vid\_Cost(K1, kiek1) < Vid\_Cost(K2, kiek2))

{

writer.WriteLine("Ūkininko {0} vidutinė kaliausės kaina mažesnė: {1:f2}", name1, Vid\_Cost(K1, kiek1));

}

else if (Vid\_Cost(K1, kiek1) > Vid\_Cost(K2, kiek2))

{

writer.WriteLine("Ūkininko {0} vidutinė kaliausės kaina mažesnė: {1:f2}", name2, Vid\_Cost(K2, kiek2));

}

else

{

writer.WriteLine("Abiejų ūkininkų vidutinės kaliausių kainos lygios: {0:f2}", Vid\_Cost(K1, kiek1));

}

}

}

static int Make\_Type\_List(Kaliausė[] K, Kaliausė[] R, int kiek, string x, int check0)

{

int check = check0;

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

{

if (x == K[i].Type)

{

R[check] = new Kaliausė(K[i].Type, K[i].Count, K[i].Mass, K[i].Cost);

check++;

}

}

return check;

}

static void Skaityti(string CFd, Kaliausė[] K, out int kiek, out string name)

{

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

{

string type;

int count;

double mass, cost;

string line;

line = reader.ReadLine();

string[] parts;

name = (line);

line = reader.ReadLine();

kiek = int.Parse(line);

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

{

line = reader.ReadLine();

parts = line.Split(' ');

type = parts[0];

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

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

cost = double.Parse(parts[3]);

K[i] = new Kaliausė(type, count, mass, cost);

}

}

}

static void SpausdintiDuomenis(string fv, double vid\_cost, string name)

{

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

{

fr.WriteLine("Vidutinė {0} kaliausės kaina: {1:f2}", name, vid\_cost);

}

}

static void SpausdintiPradiniusDuomenis(string fv, Kaliausė[] K, int kiek, string name)

{

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

{

fr.WriteLine(name);

fr.WriteLine();

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

"| | | | |\r\n" +

"| Tipas | Kiekis | Masė | Kaina |\r\n" +

"| | | | |\r\n" +

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

fr.WriteLine(virsus);

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

{

fr.WriteLine("|{0,-48}|{1,13}|{2,9}|{3,8}|", K[i].Type, K[i].Count, K[i].Mass, K[i].Cost);

}

fr.WriteLine();

fr.WriteLine();

}

}

static void SpausdintiRinki(string fv, Kaliausė[] R, int check)

{

using (StreamWriter fr = new StreamWriter(fv, true))

{

if (check == 0)

{

Console.WriteLine("Nurodyto tipo kaliausių nėra");

fr.WriteLine("Nurodyto tipo kaliausių nėra");

}

else

{

fr.WriteLine();

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

"| | | | |\r\n" +

"| Tipas | Kiekis | Masė | Kaina |\r\n" +

"| | | | |\r\n" +

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

fr.WriteLine(virsus);

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

{

fr.WriteLine("|{0,-48}|{1,13}|{2,9}|{3,8}|", R[i].Type, R[i].Count, R[i].Mass, R[i].Cost);

}

}

}

}

static void SpausdintiMax(string fv, int[] count1, int[] count2, string name1, string name2)

{

using (StreamWriter fr = new StreamWriter(fv, true))

{

if (count1.Max() == count2.Max())

{

fr.WriteLine("Abiejų ūkininkų sunkiausios kaliausės yra vienodai aprėdytos: {0}", count1.Max());

}

else if (count1.Max() < count2.Max())

{

fr.WriteLine("Ūkininko {0} sunkiausia kaliausė yra daugiausiai aprėdyta: {1}", name2, count2.Max());

}

else

{

fr.WriteLine("Ūkininko {0} sunkiausia kaliausė yra daugiausiai aprėdyta: {1}", name1, count1.Max());

}

}

}

}

}

* ***Pradiniai duomenys ir rezultatai***

1)

Duom1:

Jonas

2

baisioji 5 1 2

bjaurioji 5 1 2

Duom2:

Antanas

2

baisioji 5 1 2

bjaurioji 5 1 2

Rez:





2)

Duom1:

Jonas

5

baisioji 5 1 2

bjaurioji 5 5 7

baisioji 8 5 8

kitokia 8 4 3

bjaurioji 10 5 4

Duom2:

Antanas

3

baisioji 5 1 2

bjaurioji 5 1 2

keistoji 8 5 7

Rez:





3)

Duom1:

Jonas

5

baisioji 5 1 2

bjaurioji 5 5 7

baisioji 8 5 8

kitokia 8 4 3

bjaurioji 10 5 4

Duom2:

Antanas

3

baisioji 5 1 2

bjaurioji 5 1 2

keistoji 8 5 7

Rez:





* **Konteinerinė klasė**
* ***Darbo užduotis***
* ***Programos tekstas***
* ***Pradiniai duomenys ir rezultatai***
* **Teksto analizė ir redagavimas**
* ***Darbo užduotis***
* ***Programos tekstas***
* ***Pradiniai duomenys ir rezultatai***
* **Susieti rinkiniai**
* ***Darbo užduotis***
* ***Programos tekstas***
* ***Pradiniai duomenys ir rezultatai***