

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**  
**INFORMATIKOS FAKULTETAS**

**OBJEKTINIS PROGRAMAVIMAS I (P175B118)**  
***Laboratorinio darbo ataskaita***

Atliko:

IFF-5/7 gr. studentas

Viktorija Ražaitė

2015 m. gruodžio 15 d.

Priėmė:

Lektorius Dr. Mindaugas Jančiukas

# TURINYS

<b>1. Objektų rinkinys .....</b>	<b>3</b>
1.1. Darbo užduotis .....	3
1.2. Programos tekstas .....	3
1.3. Pradiniai duomenys ir rezultatai .....	6
1.3.1 Pradiniai duomenys .....	6
1.3.2 Rezultatai .....	6
<b>2. Konteineris .....</b>	<b>7</b>
2.1. Darbo užduotis .....	7
2.2. Programos tekstas .....	7
2.3. Pradiniai duomenys ir rezultatai .....	13
2.3.1 Pradiniai duomenys .....	13
2.3.2 Rezultatai .....	14
<b>4. Paveldėjimas .....</b>	<b>17</b>
4.1. Darbo užduotis .....	17
4.2. Programos tekstas .....	17
4.1. Pradiniai duomenys ir rezultatai .....	26
4.1.1 Pradiniai duomenys .....	26
4.1.2 Rezultatai .....	27
<b>5. Susieti objektų rinkiniai .....</b>	<b>28</b>
5.1. Darbo užduotis .....	28
5.2. Programos tekstas .....	28
5.3. Pradiniai duomenys ir rezultatai .....	28
<b>6. Teksto analizė ir redagavimas .....</b>	<b>29</b>
6.1. Darbo užduotis .....	29
6.2. Programos tekstas .....	29
6.3. Pradiniai duomenys ir rezultatai .....	36
<b>7. Sudėtingesnis konteineris .....</b>	<b>38</b>
7.1. Darbo užduotis .....	38
7.2. Programos tekstas .....	38
7.3. Pradiniai duomenys ir rezultatai .....	44

# 1. Objektų rinkinys

## 1.1. Darbo užduotės

7. WCG turnyras. Kaune vyksta atrankinis kompiuterinio žaidimo „League of Legends“ turnyras. Žaidime dvi penkių žaidėjų komandos kovoja tarpusavyje valdydamos skirtingus čempionus, siekdamos sunaikinti priešininkų bazės gilumoje stovintį „Nexus“. Duomenų faile pateikta informacija apie pirmo rato dalyvius ir jų rezultatus: vardas, pavardė, komanda, pozicija, čempionas, sunaikinimai(K), dalyvavimai sunaikinimuose(A).

- Raskite žaidėją, pademonstravusį geriausią asmeninį rezultatą. Palyginimui naudokite vadinamąjį KA rodiklį (K+A). Ekrane atspausdinkite jo vardą, pavardę, komandos pavadinimą, poziciją bei naudotą čempioną.

- Raskite, kuris čempionas buvo naudotas „universaliausiai“ (daugiausiai skirtingų pozicijų). Ekrane atspausdinkite čempiono pavadinimą, bei kokiose pozicijose jis buvo naudotas.

- Sudarykite „Top“ pozicijoje žaidusių žaidėjų sąrašą, į failą „Top.csv“ įrašykite žaidėjų komandos pavadinimus, pavardes, vardus, naudoto čempiono pavadinimą.

## 1.2. Programos tekstas

```
Klasė
using System;

namespace _7Uzd
{
    class Turnyras
    {
        public string Vardas { get; set; }
        public string Pavarde { get; set; }
        public string Komanda { get; set; }
        public string Pozicija { get; set; }
        public string Cemp { get; set; }
        public int Sunaik { get; set; }
        public int Zuvo { get; set; }
        public int Dalyvav { get; set; }

        public Turnyras()
        {
        }

        public Turnyras(string vardas, string pavarde, string komanda, string
pozicija, string cemp, int sunaik, int zuvo, int dalyvav)
        {
            Vardas = vardas;
            Pavarde = pavarde;
            Komanda = komanda;
            Pozicija = pozicija;
            Cemp = cemp;
            Sunaik = sunaik;
            Zuvo = zuvo;
            Dalyvav = dalyvav;
        }
    }
}

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

```

namespace _7Uzd
{
    class Program
    {
        public const int MaxZaid = 50; //maksimalus zaideju skaicius

        static void Main(string[] args)
        {
            Turnyras[] turnyras;
            int turnCount = 0;
            int maxRod;
            ReadData(out turnyras, out turnCount); //nuskaitomi duomenys is
            duomenu failo
            using (StreamWriter writer = new StreamWriter(@"Duomenys.txt"))
            {
                for (int i = 0; i < turnCount; i++)
                {
                    writer.WriteLine("{0,-10} | {1,-15} | {2,-15} | {3,-10} | {4,-
10} | {5,-5} | {6,-5} | {7,-5}", turnyras[i].Vardas, turnyras[i].Pavarde,
turnyras[i].Komanda, turnyras[i].Pozicija, turnyras[i].Cemp, turnyras[i].Sunaik,
turnyras[i].Zuvo, turnyras[i].Dalyvav);
                }
                //writer.WriteLine("Komanda,Pavarde,Vardas,Cempionas");
            }

            GerAsmRez(turnyras, turnCount, out maxRod); // maxRod - didziausias
            rodiklis GerAsmRez - Geriausias asmeninis rezultatas
            Console.WriteLine(" ");
            // Lenteles paruosimas
            Console.WriteLine("-----");
            Console.WriteLine("-----Zaidejas(zaidejai)-pasieke-geriausia-
            asmenini-rezultata-----");
            Console.WriteLine("-----");
            Console.WriteLine(" Vardas, pavarde | Komanda | Pozicija |
            Cempionas ");
            Console.WriteLine("-----");
            for (int i = 0; i < turnCount; i++)
            {
                //Apskaiciuojamas KA rodiklis
                if (turnyras[i].Sunaik + turnyras[i].Dalyvav == maxRod)
            //Duomenu surasymas i faila
            {
                Console.WriteLine(" {0} {1} | {2} | {3} | {4} ",
turnyras[i].Vardas, turnyras[i].Pavarde, turnyras[i].Komanda,
turnyras[i].Pozicija, turnyras[i].Cemp);
                Console.WriteLine("-----");
            }
        }
        }
        Univers(turnyras, turnCount); // universaliausio cempiono
        ieskojimas
        TopPozicija(turnyras, turnCount); // Top pozicijos zaideju radimas

        // Randami zaidejai zaidziantys top pozicijoje
        private static void TopPozicija(Turnyras[] turnyras, int turnCount)
        {
            using (StreamWriter writer = new StreamWriter(@"Top.csv")) //duomenys
            bus rasomi i excel programa
            {
                writer.WriteLine("Komanda,Pavarde,Vardas,Cempionas");
            //Lenteles paaiskinimai
                for (int i = 0; i < turnCount; i++) // ima is masyvo
            zaideju pozicijos pavadinimus ir lygina

```

```

        { // jei pozicija lygi
zodziui 'Top', tuomet
            if (turnyras[i].Pozicija == "Top") // i excel irasomas
komandos pavadinimas, zaidejo paverde vardas ir cempionas kuriame zaidziama
                writer.WriteLine("{0},{1},{2},{3}", turnyras[i].Komanda,
turnyras[i].Pavarde, turnyras[i].Vardas, turnyras[i].Cemp);
        }
    } // Universaliausio cempiono ieskojimas, tai tas kuriame zaidzia
daugiausiai zaideju skirtingomis pozicijomis
    private static void Univers(Turnyras[] turnyras, int turnCount)
    {
        int[] PozicMas = new int[turnCount]; // sukuriamas naujas masyvas,
jame bus saugoma tam tikro cempiono skirtingu poziciju skaicius
        for (int i = 0; i < turnCount; i++)
            for (int j = i+1; j < turnCount; j++)
            { //lyginama, jei cempiono vardai tokie patys ir jei skiriasi
pozicijos
                if ((turnyras[i].Cemp == turnyras[j].Cemp) &&
(turnyras[i].Pozicija != turnyras[j].Pozicija))
                {
                    PozicMas[i] += 1; //pridedamas vienetas prie poziciju
skaiciaus
                }
            }
        int max = 0; // ieskoma didziausios reiksmes numeriukas
        for (int i = 0; i < turnCount; i++)
        {
            if (PozicMas[i] > max) //jei poziciju skaicius didesnis uz max,
kuris pradzioje yra lygus 0
                max = PozicMas[i]; // tuomet jis pakeiciamas i didesni
        }
        Console.WriteLine(); // Surasomi duomenys, kurie bus spausdinami
ekrane
        Console.WriteLine("-----");
        Console.WriteLine("Cempionas, kuris buvo naudotas universaliausiai:|
{0} ", turnyras[max-1].Cemp);
        Console.WriteLine("-----");
        Console.WriteLine(" Pozicijos, kuriose buvo zaidziama: | {0}
", turnyras[max-1].Pozicija);
        for (int i = 0; i < turnCount; i++) //Renkamos skirtingos pozicijo.
Einama per visa masyva ir ieskoma, jei
        { // cempionato, turincio daugiausiai poziciju numeriukas
sutampa, tuomet tikrinama ar poziciju pavadinimai skiriasi
            if ((turnyras[max-1].Cemp == turnyras[i].Cemp) && (turnyras[max-
1].Pozicija != turnyras[i].Pozicija))
                Console.WriteLine("{0} ", turnyras[i].Pozicija); // Ir
rezultatas spausdinamas ekrane
            }
        Console.WriteLine(""); //| lenteles uzbaigimas, jis nera cikle,
kadangi neaisku, kuris skaicius bus paskutinis cikle
        Console.WriteLine("-----");
    } // funkcija nustatanti geriausia asmenini rezultata pagal KA
rodikli
    private static void GerAsmRez(Turnyras[] turnyras, int turnCount, out int
maxRod)
    {
        maxRod = 0; //Pradzioje rodiklis prilyginamas 0
        for (int i = 0; i < turnCount; i++)
        { //jei sunaikinimu skaicius su dalyvavimo skaiciu yra
didesnis uz maxRodikli
            if (turnyras[i].Sunaik + turnyras[i].Dalyvav > maxRod)

```

```

        maxRod = turnyras[i].Sunaik+turnyras[i].Dalyvav; //tuomet jis
priskiriamas maxRodikliui
    }
}
//failo nuskaitymas
private static void ReadData(out Turnyras[] turnyras, out int turnCount)
//parsines sias reikšmes
{
    turnCount = 0; //kiek is viso yra zaideju
    turnyras = new Turnyras[MaxZaid]; //maksimalus zaideju skaicius
    using (StreamReader reader = new StreamReader("Data.csv"))
    {
        string line = null;
        while (null != (line = reader.ReadLine()))
        {
            string[] values = line.Split(','); // kaip bus skiriama eilute
            string vardas = values[0];
            string pavarde = values[1];
            string komanda = values[2];
            string pozicija = values[3];
            string cemp = values[4];
            int sunaik = int.Parse(values[5]);
            int zuvo = int.Parse(values[6]);
            int dalyvav = int.Parse(values[7]);
            Turnyras turnyr = new Turnyras(vardas, pavarde, komanda,
            pozicija, cemp, sunaik, zuvo, dalyvav);
            turnyras[turnCount++] = turnyr; //skaitliuka didina vienetu
        }
    }
}
}
}
}

```

### 1.3. Pradiniai duomenys ir rezultatai

#### 1.3.1 Pradiniai duomenys

Failas : Data.csv

```

Arnas,Sofauskas,FFBL,Top,Jarvan IV,3,0,5
Gailė,Pagalvytė,FFBL,Mid,Annie,1,1,5
Jonas,Kėdžius,FFBL,AD,Ashe,3,1,5
Radvilė,Stalėiukaitė,FFBL,Support,Taric,0,0,8
Ignas,Lovauskas,FFBL,Jungle,Amumu,1,1,6
Indrė,Langaitė,Rainbow Dash,Top,Jarvan IV,8,0,10
Diana,Paveikslaitė,Rainbow Dash,Mid,Ahri,3,4,3
Dainius,Lentauskas,Rainbow Dash,AD,Vein,6,1,10
Vytenis,Dėpiauskas,Rainbow Dash,Support,Annie,2,1,17
Raigardas,Knygius,Rainbow Dash,Jungle,Lee Sin,1,2,11
Giedrius,Palangiauskas,FuriKuri,Top,Annie,3,1,10
Audrius,Dalgiauskas,FuriKuri,Mid,Ahri,6,1,8
Gintaras,Grėbliauskas,FuriKuri,AD,Corki,3,2,8
Jomantė,Dėklaitė,FuriKuri,Support,Sonna,0,4,13
Vladimiras,Lygiauskas,FuriKuri,Jungle,Jarvan IV,2,5,8
Audronė,Grindpiūtė,Girls United,Top,Jarvan IV,3,2,8
Simona,Tinklaitė,Girls United,Mid,Annie,5,0,8
Ieva,Pjūklaitė,Girls United,AD,Vein,8,0,8
Rasa,Plaktukaitė,Girls United,Support,Soraka,1,1,14
Raminta,Foteliūtė,Girls United,Jungle,Udyr,1,1,7

```

#### 1.3.2 Rezultatai

Failas : Top.csv

Failas : cmd.exe

Failas: Duomenys.txt

## 2. Konteineris

### 2.1. Darbo užduotis

U3-7. WCG (world cyber games) turnyras. Turite trijų turnyro ratų duomenis. Keičiasi duomenų formatas. Pirmoje eilutėje rato numeris, antroje – data. Toliau informacija apie dalyvius ir jų rezultatus pateikta tokiu pačiu formatu kaip L1 užduotyje.

- Raskite žaidėją, pademonstravusį geriausią bendrą (per visus tris ratus) asmeninį rezultatą. Palyginimui naudokite vadinamąjį KDA santykį (nužudymai + dalyvavimai nužudymuose)/mirtys t.y. (K+A)/D. Ekrane atspausdinkite jo vardą, pavardę, komandos pavadinimą, poziciją bei naudotą čempioną.

- Raskite, kuris čempionas buvo naudotas „universaliausiai“ (daugiausiai skirtingų pozicijų). Ekrane atspausdinkite čempiono pavadinimą, bei kokiose pozicijose jis buvo naudotas.

- Kai kurių komandų sudėtis kito turnyro metu. Raskite, kurie žaidėjai pasitraukė, ir kas juos pakeitė. Rezultatus surašykite į failą „Pasikeitimai.csv“

- Sudarykite „Top“ pozicijoje žaidusių žaidėjų sąrašą, į failą „Top.csv“ įrašykite žaidėjų komandos pavadinimus, pavardes, vardus, naudoto čempiono pavadinimą

### 2.2. Programos tekstas

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

namespace _7Uzd
{
    class Turnyras
    {
        public string Vardas { get; set; }
        public string Pavarde { get; set; }
        public string Komanda { get; set; }
        public string Pozicija { get; set; }
        public string Cemp { get; set; }
        public int Sunaik { get; set; }
        public int Zuvo { get; set; }
        public int Dalyvav { get; set; }
        public double KDA { get; set; }
        public int PozSk { get; set; }

        public Turnyras()
        {
        }

        public Turnyras(string vardas, string pavarde, string komanda, string
pozicija, string cemp, int sunaik, int zuvo, int dalyvav)
        {
            Vardas = vardas;
            Pavarde = pavarde;
            Komanda = komanda;
            Pozicija = pozicija;
            Cemp = cemp;
            Sunaik = sunaik;
            Zuvo = zuvo;
            Dalyvav = dalyvav;
        }
    }

    class ZaidimoRatas
    {
        public const int MaxZaid = 150;

        public string Ratas { get; set; }
    }
}
```

```

public string Data { get; set; }
public Turnyras[] Zaidejai { get; set; }
public int ZaidCount { get; set; }

public ZaidimoRatas(string ratas, string data)
{
    Ratas = ratas;
    Data = data;
    Zaidejai = new Turnyras[MaxZaid];
}

public ZaidimoRatas(string ratas)
{
    Ratas = ratas;
    Zaidejai = new Turnyras[MaxZaid];
}

public void AddTurnyras(Turnyras turnyras)
{
    Zaidejai[ZaidCount] = turnyras;
    ZaidCount++;
}
}

// Pagrindine programa-----
class Program
{
    public const int RatuSkaicius = 3;
    public const int MaxZaid = 150;

    static void Main(string[] args)
    {
        ZaidimoRatas[] zaidimai = new ZaidimoRatas[3];

        zaidimai[0] = new ZaidimoRatas("1");
        zaidimai[1] = new ZaidimoRatas("2");
        zaidimai[2] = new ZaidimoRatas("3");

        string[] filePaths =
Directory.GetFiles(Directory.GetCurrentDirectory(), "*new.txt");//nuskaito visus
failus, siusdamas po viena i ReadData metoda

        foreach (string path in filePaths)
        {
            ReadData(path, zaidimai);
        }

        using (StreamWriter writer = new
StreamWriter(@"PradinaiDuomenys.txt"))
        for (int i = 0; i < 3; i++)
        {
            writer.WriteLine("-----");
Ratas-{0}-----", zaidimai[i].Ratas);
            writer.WriteLine();
            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
            {
                writer.WriteLine("{0,-10} | {1,-15} | {2,-15} | {3,-10} | {4,-
10} | {5,-5} | {6,-5} | {7,-5}", zaidimai[i].Zaidejai[j].Vardas,
zaidimai[i].Zaidejai[j].Pavarde, zaidimai[i].Zaidejai[j].Komanda,
zaidimai[i].Zaidejai[j].Pozicija, zaidimai[i].Zaidejai[j].Cemp,
zaidimai[i].Zaidejai[j].Sunaik, zaidimai[i].Zaidejai[j].Zuvo,
zaidimai[i].Zaidejai[j].Dalyvav);
            }
            writer.WriteLine();
        }
    }
}

```



```

        GerAsmRez(zaidimai);
        UniversaliausiasCemp(zaidimai);
        PakiteZaidejai(zaidimai);
        TopPozicija(zaidimai);
    }

//Patikrina kuriame rate yra zaidejas -----
private static ZaidimoRatas GetZaidimoRata(ZaidimoRatas[] zaidimai, string
ratas)
{
    for (int i = 0; i < RatuSkaicius; i++)
    {
        if (zaidimai[i].Ratas == ratas)
        {
            return zaidimai[i];
        }
    }
    return null;
}

//nuskaito pradinis duomenis -----
private static void ReadData(string file, ZaidimoRatas[] zaidimai)
{
    string ratas = null;
    string data = null;

    using (StreamReader reader = new StreamReader(@file))
    {
        string line = null;

        if ((line = reader.ReadLine()) != null)
        {
            ratas = line;
        }

        if ((line = reader.ReadLine()) != null)
        {
            data = line;
        }
        ZaidimoRatas zaidimoratas = GetZaidimoRata(zaidimai, ratas);
        while (null != (line = reader.ReadLine()))
        {
            string[] values = line.Split(',');
            string vardas = values[0];
            string pavarde = values[1];
            string komanda = values[2];
            string pozicija = values[3];
            string cemp = values[4];
            int sunaik = int.Parse(values[5]);
            int zuvo = int.Parse(values[6]);
            int dalyvav = int.Parse(values[7]);
            Turnyras turnyras = new Turnyras(vardas, pavarde, komanda,
pozicija, cemp, sunaik, zuvo, dalyvav);

            if (!zaidimoratas.Zaidejai.Contains(turnyras))
            {
                zaidimoratas.AddTurnyras(turnyras);
            }
        }
    }
}

```

```

//Iesko zaidejo, turincio geriausia asmenini rezultata, pagal KDA santyki
(nuzudymai + dalyvavimai nuzudymuose)/mirtys t.y. (K+A)/D
private static void GerAsmRez(ZaidimoRatas[] zaidimai)
{
    Turnyras[] VisiZaid = new Turnyras[MaxZaid];
    int count = 0;

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < zaidimai[i].ZaidCount; j++)
        {
            if (!VisiZaid.Contains(zaidimai[i].Zaidejai[j]))
            {
                VisiZaid[count] = zaidimai[i].Zaidejai[j];
                count++;
            }
        }
    }
    for (int i = 0; i < count; i++)
    {
        VisiZaid[i].Sunaik = 0;
        VisiZaid[i].Zuvo = 1;
        VisiZaid[i].Dalyvav = 0;
        for (int j = 0; j < 3; j++)
        {
            for (int k = 0; k < zaidimai[j].ZaidCount; k++)
            {
                if (VisiZaid[i].Pavarde ==
zaidimai[j].Zaidejai[k].Pavarde)
                {
                    VisiZaid[i].Sunaik += zaidimai[j].Zaidejai[k].Sunaik;
                    VisiZaid[i].Zuvo += zaidimai[j].Zaidejai[k].Zuvo;
                    VisiZaid[i].Dalyvav +=
zaidimai[j].Zaidejai[k].Dalyvav;
                }
            }
        }

        for (int i = 0; i < count; i++)
        {
            VisiZaid[i].KDA = (VisiZaid[i].Sunaik + VisiZaid[i].Dalyvav) /
VisiZaid[i].Zuvo;
        }

        Turnyras GeriausiasKDA = new Turnyras();
        GeriausiasKDA.KDA = 0;
        for (int i = 0; i < count; i++)
        {
            if (VisiZaid[i].KDA > GeriausiasKDA.KDA)
            {
                GeriausiasKDA = VisiZaid[i];
            }
        }

        Console.WriteLine();
        Console.WriteLine("-----");
        Console.WriteLine("-----Zaidejas-pasiekes-geriausia-asmenini-
rezultata-----");
        Console.WriteLine("-----");
        Console.WriteLine(" Vardas, pavarde | Komanda |
Pozicija | Cempionas ");
        Console.WriteLine("-----");
        Console.WriteLine("-----");
    }
}

```

```

        Console.WriteLine(" {0} {1} | {2}      | {3}          | {4}          ",
GeriausiasKDA.Vardas, GeriausiasKDA.Pavarde, GeriausiasKDA.Komanda,
GeriausiasKDA.Pozicija, GeriausiasKDA.Cemp);
        Console.WriteLine("-----");
        -----");
    }

//Iesko universaliausio cempiono, tirkina, kur buvo zaidziama daugiausia skirtingu
poziciju -----
    private static void UniversaliausiasCemp(ZaidimoRatas[] zaidimai)
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
            {
                for (int k = j+1; k < zaidimai[i].ZaidCount; k++)
                {
                    if ((zaidimai[i].Zaidejai[j].Cemp ==
zaidimai[i].Zaidejai[k].Cemp) && (zaidimai[i].Zaidejai[j].Pozicija !=
zaidimai[i].Zaidejai[k].Pozicija))
                    {
                        zaidimai[i].Zaidejai[j].PozSk += 1;
                    }
                }
            }
        }
        int max = 0;
        int ratoNr = 0;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
            {
                if (zaidimai[i].Zaidejai[j].PozSk > max)
                {
                    max = zaidimai[i].Zaidejai[j].PozSk;
                    ratoNr = i;
                }
            }
        }

        Console.WriteLine();
        Console.WriteLine("-----");
        Console.WriteLine("Cempionas, kuris buvo naudotas
universaliausiai:| {0}      ", zaidimai[ratoNr].Zaidejai[max - 1].Cemp);
        Console.WriteLine("-----");
        Console.WriteLine("-----");
        Console.WriteLine("Pozicijos, kuriose buvo zaidziama:          |
{0}", zaidimai[ratoNr].Zaidejai[max - 1].Pozicija);
        for (int i = 0; i < 1; i++)
        {
            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
            {
                if ((zaidimai[ratoNr].Zaidejai[max - 1].Cemp ==
zaidimai[i].Zaidejai[j].Cemp) && (zaidimai[ratoNr].Zaidejai[max - 1].Pozicija !=
zaidimai[i].Zaidejai[j].Pozicija))
                    Console.WriteLine(" {0}",
zaidimai[i].Zaidejai[j].Pozicija);
            }
        }
        Console.WriteLine();
        Console.WriteLine("-----");
        -----");
    }

```

```

    }

//Iesko zaideju kurie pasitrauke 2trame ir 3ciame ratuose, ir kas juos pakeite.
Rezultatai surasomi i faila „Pasikeitimai.csv“-----
private static void PakiteZaidejai(ZaidimoRatas[] zaidimai)
{
    using (StreamWriter writer = new StreamWriter(@"Pasikeitimai.csv"))
    {
        for (int i = 0; i < 2; i++)
        {
            writer.WriteLine("{0}-ame rate ivyke pasikeitimai:",i+2);
            writer.WriteLine();
            writer.WriteLine("Ka pakeite,,,Kas pakeite");
            writer.WriteLine("Vardas,Pavarde,,Vardas,Pavarde");
            writer.WriteLine();
            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
                if (zaidimai[i].Zaidejai[j].Vardas != zaidimai[i +
1].Zaidejai[j].Vardas)
                    writer.WriteLine("{0},{1},,{2},{3}",
zaidimai[i].Zaidejai[j].Vardas, zaidimai[i].Zaidejai[j].Pavarde, zaidimai[i +
1].Zaidejai[j].Vardas, zaidimai[i + 1].Zaidejai[j].Pavarde);
            writer.WriteLine();
        }
    }
}

//Iesko zaideju zaidzianciu "Top" pozicijoje -----
private static void TopPozicija(ZaidimoRatas[] zaidimai)
{
    using (StreamWriter writer = new StreamWriter(@"Top.csv"))
    {
        writer.WriteLine("RatoNr,Komanda,Pavarde,Vardas,Cempionas");
        writer.WriteLine();
        for (int i = 0; i < 3; i++)
        {
            writer.Write("{0}-as ratas",i+1);

            for (int j = 0; j < zaidimai[i].ZaidCount; j++)
            {
                if (zaidimai[i].Zaidejai[j].Pozicija == "Top")
                    if (i == 0)
                        writer.WriteLine(",{0},{1},{2},{3}",
zaidimai[i].Zaidejai[j].Komanda, zaidimai[i].Zaidejai[j].Pavarde,
zaidimai[i].Zaidejai[j].Vardas, zaidimai[i].Zaidejai[j].Cemp);
                    else if (i == 1)
                        if (zaidimai[i - 1].Zaidejai[j].Vardas !=
zaidimai[i].Zaidejai[j].Vardas)
                            writer.WriteLine(",{0},{1},{2},{3}",
zaidimai[i].Zaidejai[j].Komanda, zaidimai[i].Zaidejai[j].Pavarde,
zaidimai[i].Zaidejai[j].Vardas, zaidimai[i].Zaidejai[j].Cemp);
                        else if (zaidimai[i - 1].Zaidejai[j].Vardas !=
zaidimai[i].Zaidejai[j].Vardas)
                            writer.WriteLine(",{0},{1},{2},{3}",
zaidimai[i].Zaidejai[j].Komanda, zaidimai[i].Zaidejai[j].Pavarde,
zaidimai[i].Zaidejai[j].Vardas, zaidimai[i].Zaidejai[j].Cemp);
                    }
                writer.WriteLine();
            }
        }
    }
}
}

```

## **2.3. Pradiniai duomenys ir rezultatai**

### **2.3.1 Pradiniai duomenys**

Failas: Data1

1

2015-10-03

Arnas,Sofauskas,FFBL,Top,Jarvan IV,3,1,5  
Gaile,Pagalvyte,FFBL,Mid,Annie,1,1,5  
Jonas,Kedzius,FFBL,AD,Ashe,3,1,5  
Radvile,Stalciukaite,FFBL,Support,Taric,0,1,8  
Ignas,Lovauskas,FFBL,Jungle,Amumu,1,1,6  
Indre,Langaite,Rainbow Dash,Top,Jarvan IV,8,1,10  
Diana,Paveikslaitė,Rainbow Dash,Mid,Ahri,3,4,3  
Dainius,Lentauskas,Rainbow Dash,AD,Vein,6,1,10  
Vytenis,Deziauskas,Rainbow Dash,Support,Annie,2,1,17  
Raigardas,Knygius,Rainbow Dash,Jungle,Lee Sin,1,2,11  
Giedrius,Palangiauskas,FuriKuri,Top,Annie,3,1,10  
Audrius,Dalgiauskas,FuriKuri,Mid,Ahri,6,1,8  
Gintaras,Grebliauskas,FuriKuri,AD,Corki,3,2,8  
Jomante,Deklaite,FuriKuri,Support,Sonna,0,4,13  
Vladimiras,Lygiauskas,FuriKuri,Jungle,Jarvan IV,2,5,8  
Audrone,Grindziute,Girls United,Top,Jarvan IV,3,2,8  
Simona,Tinklaite,Girls United,Mid,Annie,5,1,8  
Ieva,Pjuklaite,Girls United,AD,Vayne,8,1,8  
Rasa,Plaktukaite,Girls United,Support,Soraka,1,1,14  
Raminta,Foteliute,Girls United,Jungle,Udyr,1,1,7  
Antanas,Stiklius,1337,Top,Soraka,0,1,8  
Jonas,Puodzius,1337,Mid,Ahri,3,1,5  
Algirdas,Kalvaitis,1337,AD,Caitlyn,3,1,5  
Jurgis,Siauciunas,1337,Support,Taric,1,1,6  
Stasys,Girnaitis,1337,Jungle,Lee Sin,1,1,5

Failas: Data2

2

2015-10-05

Arnas,Sofauskas,FFBL,Top,Jarvan IV,3,1,5  
Meile,Meilyte,FFBL,Mid,Annie,1,2,5  
Jonas,Kedzius,FFBL,AD,Ashe,3,1,5  
Radvile,Stalciukaite,FFBL,Support,Taric,0,1,8  
Ignas,Lovauskas,FFBL,Jungle,Amumu,1,1,6  
Saule,Saulute,Rainbow Dash,Top,Jarvan IV,8,1,9  
Diana,Paveikslaitė,Rainbow Dash,Mid,Ahri,3,4,3  
Dainius,Lentauskas,Rainbow Dash,AD,Vein,6,1,10  
Vytenis,Deziauskas,Rainbow Dash,Support,Annie,2,1,17  
Raigardas,Knygius,Rainbow Dash,Jungle,Lee Sin,1,2,11  
Medis,Matelis,FuriKuri,Top,Annie,3,2,10  
Audrius,Dalgiauskas,FuriKuri,Mid,Ahri,6,1,8  
Gintaras,Grebliauskas,FuriKuri,AD,Corki,3,2,8  
Jomante,Deklaite,FuriKuri,Support,Sonna,0,4,13  
Vladimiras,Lygiauskas,FuriKuri,Jungle,Jarvan IV,2,5,8  
Audrone,Grindziute,Girls United,Top,Jarvan IV,3,2,8  
Simona,Tinklaite,Girls United,Mid,Annie,5,1,8  
Ieva,Pjuklaite,Girls United,AD,Vayne,8,1,8  
Rasa,Plaktukaite,Girls United,Support,Soraka,1,1,14  
Ratas,Ratelis,Girls United,Jungle,Udyr,2,1,7  
Antanas,Stiklius,1337,Top,Soraka,0,1,8  
Jonas,Puodzius,1337,Mid,Ahri,3,1,5  
Algirdas,Kalvaitis,1337,AD,Caitlyn,3,1,5  
Jurgis,Siauciunas,1337,Support,Taric,1,1,6

Stasys,Girnaitis,1337,Jungle,Lee Sin,1,1,5

Failas: Data3

3

2015-10-07

Arnas,Sofauskas,FFBL,Top,Jarvan IV,3,2,5

Meile,Meilyte,FFBL,Mid,Annie,1,2,5

Jonas,Kedzius,FFBL,AD,Ashe,3,1,5

Radvile,Stalciukaite,FFBL,Support,Taric,2,2,8

Ignas,Lovauskas,FFBL,Jungle,Amumu,1,1,6

Saule,Saulute,Rainbow Dash,Top,Jarvan IV,8,2,9

Diana,Paveikslaitė,Rainbow Dash,Mid,Ahri,3,4,3

Dainius,Lentauskas,Rainbow Dash,AD,Vein,6,1,12

Sunelis,Sunys,Rainbow Dash,Support,Annie,2,1,14

Raigardas,Knygius,Rainbow Dash,Jungle,Lee Sin,1,2,11

Medis,Matelis,FuriKuri,Top,Annie,3,2,12

Audrius,Dalgiauskas,FuriKuri,Mid,Ahri,6,1,8

Gintaras,Grebliuskas,FuriKuri,AD,Corki,3,2,8

Jomante,Deklaite,FuriKuri,Support,Sonna,2,4,13

Aukse,Auksele,FuriKuri,Jungle,Jarvan IV,4,5,8

Audrone,Grindziute,Girls United,Top,Jarvan IV,3,2,8

Simona,Tinklaitė,Girls United,Mid,Annie,5,2,8

Ieva,Pjuklaite,Girls United,AD,Vein,8,2,8

Rasa,Plaktukaite,Girls United,Support,Soraka,1,1,14

Ratas,Ratelis,Girls United,Jungle,Udyr,2,1,7

Antanas,Stiklius,1337,Top,Soraka,2,2,8

Jonas,Puodzius,1337,Mid,Ahri,3,2,5

Algirdas,Kalvaitis,1337,AD,Caitlyn,3,1,5

Jurgis,Siauciunas,1337,Support,Taric,1,1,6

Stasys,Girnaitis,1337,Jungle,Lee Sin,1,1,5

### 2.3.2 Rezultatai

Failas: PradiniaiDuomenys.txt

-----Ratas-1-----

Arnas	Sofauskas	FFBL	Top	Jarvan IV	3	1	5
Gaile	Pagalvyte	FFBL	Mid	Annie	1	1	5
Jonas	Kedzius	FFBL	AD	Ashe	3	1	5
Radvile	Stalciukaite	FFBL	Support	Taric	0	1	8
Ignas	Lovauskas	FFBL	Jungle	Amumu	1	1	6
Indre	Langaite	Rainbow Dash	Top	Jarvan IV	8	1	10
Diana	Paveikslaitė	Rainbow Dash	Mid	Ahri	3	4	3
Dainius	Lentauskas	Rainbow Dash	AD	Vein	6	1	10
Vytenis	Deziauskas	Rainbow Dash	Support	Annie	2	1	17
Raigardas	Knygius	Rainbow Dash	Jungle	Lee Sin	1	2	11
Giedrius	Palangiauskas	FuriKuri	Top	Annie	3	1	10
Audrius	Dalgiauskas	FuriKuri	Mid	Ahri	6	1	8
Gintaras	Grebliuskas	FuriKuri	AD	Corki	3	2	8
Jomante	Deklaite	FuriKuri	Support	Sonna	0	4	13
Vladimiras	Lygiauskas	FuriKuri	Jungle	Jarvan IV	2	5	8
Audrone	Grindziute	Girls United	Top	Jarvan IV	3	2	8
Simona	Tinklaitė	Girls United	Mid	Annie	5	1	8
Ieva	Pjuklaite	Girls United	AD	Vayne	8	1	8
Rasa	Plaktukaite	Girls United	Support	Soraka	1	1	14
Raminta	Foteliute	Girls United	Jungle	Udyr	1	1	7
Antanas	Stiklius	1337	Top	Soraka	0	1	8
Jonas	Puodzius	1337	Mid	Ahri	3	1	5
Algirdas	Kalvaitis	1337	AD	Caitlyn	3	1	5
Jurgis	Siauciunas	1337	Support	Taric	1	1	6

Stasys		Girnaitis		1337		Jungle		Lee Sin		1		1		5
--------	--	-----------	--	------	--	--------	--	---------	--	---	--	---	--	---

-----Ratas-2-----

Arnas		Sofauskas		FFBL		Top		Jarvan IV		3		1		5
Meile		Meilyte		FFBL		Mid		Annie		1		2		5
Jonas		Kedzius		FFBL		AD		Ashe		3		1		5
Radvile		Stalciukaite		FFBL		Support		Taric		0		1		8
Ignas		Lovauskas		FFBL		Jungle		Amumu		1		1		6
Saule		Saulute		Rainbow Dash		Top		Jarvan IV		8		1		9
Diana		Paveikslaitė		Rainbow Dash		Mid		Ahri		3		4		3
Dainius		Lentauskas		Rainbow Dash		AD		Vein		6		1		10
Vytenis		Deziauskas		Rainbow Dash		Support		Annie		2		1		17
Raigardas		Knygius		Rainbow Dash		Jungle		Lee Sin		1		2		11
Medis		Matelis		FuriKuri		Top		Annie		3		2		10
Audrius		Dalgiauskas		FuriKuri		Mid		Ahri		6		1		8
Gintaras		Grebliuskas		FuriKuri		AD		Corki		3		2		8
Jomante		Deklaite		FuriKuri		Support		Sonna		0		4		13
Vladimiras		Lygiauskas		FuriKuri		Jungle		Jarvan IV		2		5		8
Audrone		Grindziute		Girls United		Top		Jarvan IV		3		2		8
Simona		Tinklaite		Girls United		Mid		Annie		5		1		8
Ieva		Pjuklaite		Girls United		AD		Vayne		8		1		8
Rasa		Plaktukaite		Girls United		Support		Soraka		1		1		14
Ratas		Ratelis		Girls United		Jungle		Udyr		2		1		7
Antanas		Stiklius		1337		Top		Soraka		0		1		8
Jonas		Puodzius		1337		Mid		Ahri		3		1		5
Algirdas		Kalvaitis		1337		AD		Caitlyn		3		1		5
Jurgis		Siauciunas		1337		Support		Taric		1		1		6
Stasys		Girnaitis		1337		Jungle		Lee Sin		1		1		5

-----Ratas-3-----

Arnas		Sofauskas		FFBL		Top		Jarvan IV		3		2		5
Meile		Meilyte		FFBL		Mid		Annie		1		2		5
Jonas		Kedzius		FFBL		AD		Ashe		3		1		5
Radvile		Stalciukaite		FFBL		Support		Taric		2		2		8
Ignas		Lovauskas		FFBL		Jungle		Amumu		1		1		6
Saule		Saulute		Rainbow Dash		Top		Jarvan IV		8		2		9
Diana		Paveikslaitė		Rainbow Dash		Mid		Ahri		3		4		3
Dainius		Lentauskas		Rainbow Dash		AD		Vein		6		1		12
Sunelis		Sunys		Rainbow Dash		Support		Annie		2		1		14
Raigardas		Knygius		Rainbow Dash		Jungle		Lee Sin		1		2		11
Medis		Matelis		FuriKuri		Top		Annie		3		2		12
Audrius		Dalgiauskas		FuriKuri		Mid		Ahri		6		1		8
Gintaras		Grebliuskas		FuriKuri		AD		Corki		3		2		8
Jomante		Deklaite		FuriKuri		Support		Sonna		2		4		13
Aukse		Auksele		FuriKuri		Jungle		Jarvan IV		4		5		8
Audrone		Grindziute		Girls United		Top		Jarvan IV		3		2		8
Simona		Tinklaite		Girls United		Mid		Annie		5		2		8
Ieva		Pjuklaite		Girls United		AD		Vein		8		2		8
Rasa		Plaktukaite		Girls United		Support		Soraka		1		1		14
Ratas		Ratelis		Girls United		Jungle		Udyr		2		1		7
Antanas		Stiklius		1337		Top		Soraka		2		2		8
Jonas		Puodzius		1337		Mid		Ahri		3		2		5
Algirdas		Kalvaitis		1337		AD		Caitlyn		3		1		5
Jurgis		Siauciunas		1337		Support		Taric		1		1		6
Stasys		Girnaitis		1337		Jungle		Lee Sin		1		1		5

Failas: Pasikeitimai.csv

2-ame rate ivyke pasikeitimai:

Ka pakeite,,,Kas pakeite  
Vardas,Pavarde,,Vardas,Pavarde

Gaile,Pagalvyte,,Meile,Meilyte  
Indre,Langaite,,Saulute,Saulute  
Giedrius,Palangiauskas,,Medis,Matelis  
Raminta,Foteliute,,Ratas,Ratelis

3-ame rate ivyke pasikeitimai:

Ka pakeite,,,Kas pakeite  
Vardas,Pavarde,,Vardas,Pavarde

Vytenis,Deziauskas,,Sunelis,Sunys  
Vladimiras,Lygiauskas,,Aukse,Auksele

Failas: Top.csv  
RatoNr,Komanda,Pavarde,Vardas,Cempionas

1-as ratas,FFBL,Sofauskas,Arnas,Jarvan IV  
,Rainbow Dash,Langaite,Indre,Jarvan IV  
,FuriKuri,Palangiauskas,Giedrius,Annie  
,Girls United,Grindziute,Audrone,Jarvan IV  
,1337,Stiklius,Antanas,Soraka

2-as ratas,Rainbow Dash,Saulute,Saule,Jarvan IV  
,FuriKuri,Matelis,Medis,Annie

3-as ratas

Failas: cmd.exe

```
-----Zaidejas-pasiekės-geriausia-asmenini-rezultata-----
Vardas, pavarde | Komanda | Pozicija | Cempionas
Dainius Lentauskas | Rainbow Dash | AD | Vein
Cempionas, kuris buvo naudotas universaliausiai:| Annie
Pozicijos, kuriose buvo zaidžiama: | Mid Support Top
```



## 4. Paveldėjimas

### 4.1. Darbo užduotis

U4\_7.WCG turnyras. (world cyber games).

Turite trijų turnyro ratų duomenis. Keičiasi duomenų formatas. Pirmoje eilutėje rato numeris, antroje – data. Toliau pateikta informacija apie to rato rezultatus. Turnyre varžosi kelių skirtingų žaidimų („League of Legends“ ir „Counter Strike“) žaidėjai. Sukurkite klasę „Žaidėjas“ (laukai - vardas, pavardė, komanda), kurią paveldės klasės „LOLŽaidėjas“ (papildomi laukai - pozicija, čempionas, nužudymai(K), mirtys(D), dalyvavimai nužudymuose(A)) ir „CSŽaidėjas“ (papildomi laukai - nužudymai(K), mirtys(D), mėgstamiausias ginklas).

- Raskite žaidėją, pademonstravusį geriausią bendrą (per visus tris ratus) asmeninį rezultatą. LOL žaidėjų palyginimui naudokite vadinamąjį KDA santykį (nužudymai + dalyvavimai nužudymuose)/mirtys t.y. (K+A)/D, o CS žaidėjams KD santykį (K/D). Ekrane atspausdinkite jų vardus, pavardes ir komandos pavadinimą.

- Kai kurie žaidėjai dalyvauja tiek LOL, tiek CS turnyruose. Raskite tuos žaidėjus, surikiuokite juos pagal pavardę ir vardą ir įrašykite į failą „Universalus.csv“.

- Sudarykite visų turnyre dalyvaujančių komandų sąrašą, ir atspausdinkite faile „Komandos.csv“.

### 4.2. Programos tekstas

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

namespace ConsoleApplication1
{
    /// <summary>
    /// Motinine klase
    /// </summary>

    class Zaidejas
    {
        public string vardas { get; set; }
        public string pavarde { get; set; }
        public string komanda { get; set; }

        public Zaidejas()
        {
        }

        public Zaidejas(string vardas, string pavarde, string komanda)
        {
            this.vardas = vardas;
            this.pavarde = pavarde;
            this.komanda = komanda;
        }

        public override bool Equals(object obj)
        {
            return base.Equals(obj as Zaidejas);
        }

        public bool Equals(Zaidejas zaidejas)
        {
            if (Object.ReferenceEquals(zaidejas, null))
            {
                return false;
            }
        }
    }
}
```

```

        if (this.GetType() != zaidejas.GetType())
            return false;
        return (pavarde == zaidejas.pavarde);
    }

    public override int GetHashCode()
    {
        return pavarde.GetHashCode() ^ vardas.GetHashCode();
    }
// palygina zodzius ir surikiuoja pagal abecele
    public static bool operator >=(Zaidejas pirmas, Zaidejas antras)
    {
        int ip = String.Compare(pirmas.pavarde, antras.pavarde,
StringComparison.CurrentCulture);
        return ip > 0 || ip == 0; //&& pirmas.pavarde > antras.pavarde;
    }

    public static bool operator <=(Zaidejas pirmas, Zaidejas antras)
    {
        int ip = String.Compare(pirmas.pavarde, antras.pavarde,
StringComparison.CurrentCulture);
        return ip < 0 || ip == 0; // && pirmas.pavarde < antras.pavarde;
    }
}
/// <summary>
/// LolZaidejas pavedi motinine klase Zaidejas
/// </summary>
class LOLZaidejas : Zaidejas
{
    public LOLZaidejas()
    {
    }

    public string pozicija { get; set; }
    public string cempionas { get; set; }
    public int nuzudymai { get; set; }
    public int mirtys { get; set; }
    public int dalyvNuzud { get; set; }
    public double KDA { get; set; }

    public LOLZaidejas(string vardas, string pavarde, string komanda, string
pozicija, string cempionas, int nuzudymai, int mirtys, int dalyvNuzud)
        : base(vardas, pavarde, komanda)
    {
        this.pozicija = pozicija;
        this.cempionas = cempionas;
        this.nuzudymai = nuzudymai;
        this.mirtys = mirtys;
        this.dalyvNuzud = dalyvNuzud;
    }
// palygina zodzius ir surikiuoja pagal abecele
    public static bool operator >=(LOLZaidejas pirmas, LOLZaidejas antras)
    {
        int ip = String.Compare(pirmas.pavarde, antras.pavarde,
StringComparison.CurrentCulture);
        return ip > 0 || ip == 0; //&& pirmas.pavarde > antras.pavarde;
    }

    public static bool operator <=(LOLZaidejas pirmas, LOLZaidejas antras)
    {
        int ip = String.Compare(pirmas.pavarde, antras.pavarde,
StringComparison.CurrentCulture);
        return ip < 0 || ip == 0; // && pirmas.pavarde < antras.pavarde;
    }
}
/// <summary>

```

```

/// CSZaidejas pavedi motinine klase Zaidejas
/// </summary>
class CSZaidejas : Zaidejas
{
    public CSZaidejas()
    {
    }

    public int nuzudymai { get; set; }
    public int mirtys { get; set; }
    public string megstGinklas { get; set; }
    public double KD { get; set; }

    public CSZaidejas(string vardas, string pavarde, string komanda, int
nuzudymai, int mirtys, string megstGinklas)
        : base(vardas, pavarde, komanda)
    {
        this.nuzudymai = nuzudymai;
        this.mirtys = mirtys;
        this.megstGinklas = megstGinklas;
    }
}
/// <summary>
/// Konteinerine klase
/// </summary>
class Konteinerine
{
    public const int MaxZaideju = 250;
    public const int RatuSkaicius = 3;

    public LOLZaidejas[] lolzaidejuMasyvas { get; set; }
    public CSZaidejas[] cszaidejuMasyvas { get; set; }
    public int lolzaidCount { get; private set; }
    public int cszaidCount { get; private set; }

    public string ratas { get; set; }
    public string data { get; set; }

    public Konteinerine(string ratas, string data)
    {
        this.ratas = ratas;
        this.data = data;
        lolzaidejuMasyvas = new LOLZaidejas[MaxZaideju];
        cszaidejuMasyvas = new CSZaidejas[MaxZaideju];
    }

    public Konteinerine(string ratas)
    {
        this.ratas = ratas;
        lolzaidejuMasyvas = new LOLZaidejas[MaxZaideju];
        cszaidejuMasyvas = new CSZaidejas[MaxZaideju];
    }

    public Konteinerine()
    {
        lolzaidejuMasyvas = new LOLZaidejas[MaxZaideju];
        lolzaidCount = 0;
        cszaidejuMasyvas = new CSZaidejas[MaxZaideju];
        cszaidCount = 0;
    }

    public void PridetiLOLzaideja(LOLZaidejas lolzaidejas)
    {
        lolzaidejuMasyvas[lolzaidCount] = lolzaidejas;
        lolzaidCount++;
    }
}

```

```

        public void PridetiCSzaideja(CSZaidejas cszaidejas)
        {
            cszaidejuMasyvas[cszaidCount] = cszaidejas;
            cszaidCount++;
        }
    }
    /// <summary>
    /// Pafrindine programa
    /// </summary>
    class Program
    {
        public const int MaxZaideju = 250;
        public const int RatuSkaicius = 3;

        static void Main(string[] args)
        {
            Konteinerine[] zaidimai = new Konteinerine[RatuSkaicius];
            zaidimai[0] = new Konteinerine("1");
            zaidimai[1] = new Konteinerine("2");
            zaidimai[2] = new Konteinerine("3");

            string[] filePaths =
Directory.GetFiles(Directory.GetCurrentDirectory(), "*new.txt");

            foreach (string path in filePaths)
            {
                ReadData(path, zaidimai);
            }
// visi zaidejai is skirtingu ratu surasomi i viena objekta
            LOLZaidejas[] VisiLolZaidejai = new LOLZaidejas[MaxZaideju];
            int lolcount = 0;
            SurasmasLolZaideju(zaidimai, out lolcount, ref VisiLolZaidejai);

            CSZaidejas[] VisiCsZaidejai = new CSZaidejas[MaxZaideju];
            int cscount = 0;
            SurasmasCsZaideju(zaidimai, out cscount, ref VisiCsZaidejai);

//-----
            GeriausiasAsmeninisRezultatas(zaidimai, VisiLolZaidejai, lolcount,
VisiCsZaidejai, cscount);
            UniversalusZaidejai(zaidimai, VisiLolZaidejai, lolcount,
VisiCsZaidejai, cscount);
            KomanduSarasas(zaidimai);
        }
        /// <summary>
        /// Patikrina kuriame rate zaidzia zaidejas
        /// </summary>
        /// <param name="zaidimai">Konteinerines klases objektas</param>
        /// <param name="ratas">Atsinestas rato skaicius</param>
        private static Konteinerine ZaidimoRatas(Konteinerine[] zaidimai, string
ratas)
        {
            for (int i = 0; i < RatuSkaicius; i++)
            {
                if (zaidimai[i].ratas == ratas)
                {
                    return zaidimai[i];
                }
            }
            return null;
        }

        /// <summary>
        /// Nuskaito duomenis is failu
        /// </summary>

```

```

/// <param name="zaidimai">Konteinerines klases objektas</param>
/// <param name="file">Atsinesti failu pavadinimai</param>>
public static void ReadData(string file, Konteinerine[] zaidimai)
{
    string ratas = null;
    string data = null;

    using (StreamReader reader = new StreamReader(@file))
    {
        string line = null;
        if ((line = reader.ReadLine()) != null)
        {
            ratas = line;
        }
        if ((line = reader.ReadLine()) != null)
        {
            data = line;
        }
        Konteinerine konteinerine = ZaidimoRatas(zaidimai, ratas);
        while (null != (line = reader.ReadLine()))
        {
            string[] values = line.Split(',');
            char tipas = Convert.ToChar(line[0]);
            string vardas = values[1];
            string pavarde = values[2];
            string komanda = values[3];
            switch (tipas)
            {
                case 'L':
                    string pozicija = values[4];
                    string cempionas = values[5];
                    int nuzudymai = int.Parse(values[6]);
                    int mirtys = int.Parse(values[7]);
                    int dalyvNuzud = int.Parse(values[8]);
                    LOLZaidejas lolzaidejas = new LOLZaidejas(vardas,
pavarde, komanda, pozicija, cempionas, nuzudymai, mirtys, dalyvNuzud);
                    if
(!konteinerine.lolzaidejuMasyvas.Contains(lolzaidejas))
                    {
                        konteinerine.PridetiLOLzaideja(lolzaidejas);
                    }
                    break;
                case 'C':
                    int nuzudymai1 = int.Parse(values[4]);
                    int mirtys1 = int.Parse(values[5]);
                    string megstGinklas = values[6];
                    CSZaidejas cszaidejas = new CSZaidejas(vardas,
pavarde, komanda, nuzudymai1, mirtys1, megstGinklas);
                    if
(!konteinerine.cszaidejuMasyvas.Contains(cszaidejas))
                    {
                        konteinerine.PridetiCSzaideja(cszaidejas);
                    }
                    break;
            }
        }
    }
}

/// <summary>
/// Suraso visus Lol zaidejus i viena masyva
/// </summary>
/// <param name="zaidimai">Konteinerines klases objektas</param>
/// <param name="lolcount">skaitliukas, kiek bus zaideju</param>

```

```

    /// /// <param name="VisiLolZaidejai">naujas Lol zaideju objektas</param>
    public static void SurasmasLolZaideju(Konteinerine[] zaidimai, out int
lolcount, ref LOLZaidejas[] VisiLolZaidejai)
    {
        //LOLZaidejas[] VisiLolZaidejai = new LOLZaidejas[MaxZaideju];
        lolcount = 0;
        for (int i = 0; i < RatuSkaicius; i++)
        {
            for (int j = 0; j < zaidimai[i].lolzaidCount; j++)
            {
                if
(!VisiLolZaidejai.Contains(zaidimai[i].lolzaidejuMasyvas[j]))
                {
                    VisiLolZaidejai[lolcount] =
zaidimai[i].lolzaidejuMasyvas[j];
                    lolcount++;
                }
            }
        }

        /// <summary>
        /// Suraso visus Cs zaidejus i viena masyva
        /// </summary>
        /// <param name="zaidimai">Konteinerines klases objektas</param>
        /// <param name="cscount">skaitliukas, kiek bus zaideju</param>
        /// <param name="VisiCsZaidejai">naujas Cs zaideju objektas</param>
        public static void SurasmasCsZaideju(Konteinerine[] zaidimai, out int
cscount, ref CSZaidejas[] VisiCsZaidejai)
        {
            //LOLZaidejas[] VisiLolZaidejai = new LOLZaidejas[MaxZaideju];
            cscount = 0;
            for (int i = 0; i < RatuSkaicius; i++)
            {
                for (int j = 0; j < zaidimai[i].cszaidCount; j++)
                {
                    if (!VisiCsZaidejai.Contains(zaidimai[i].cszaidejuMasyvas[j]))
                    {
                        VisiCsZaidejai[cscount] = zaidimai[i].cszaidejuMasyvas[j];
                        cscount++;
                    }
                }
            }

            /// <summary>
            /// Skaiciuoja geriausia asmenini rezultata
            /// </summary>
            /// <param name="zaidimai">Konteinerines klases objektas</param>
            /// <param name="VisiLolZaidejai">Lol zaideju objektas</param>
            /// <param name="lolcount">skaitliukas, kiek zaideju</param>
            /// <param name="VisiCsZaidejai">Cs zaideju objektas</param>
            /// <param name="cscount">skaitliukas, kiek zaideju</param>
            public static void GeriausiasAsmeninisRezultatas(Konteinerine[] zaidimai,
LOLZaidejas[] VisiLolZaidejai, int lolcount, CSZaidejas[] VisiCsZaidejai, int
cscount)
            {
                for (int i = 0; i < lolcount; i++)
                {
                    VisiLolZaidejai[i].nuzudymai = 0;
                    VisiLolZaidejai[i].mirtys = 0;
                    VisiLolZaidejai[i].dalyvNuzud = 0;
                    for (int j = 0; j < RatuSkaicius; j++)
                    {
                        for (int k = 0; k < zaidimai[j].lolzaidCount; k++)

```

```

        if (VisiLolZaidejai[i].pavarde ==
zaidimai[j].lolzaidejuMasyvas[k].pavarde)
        {
            VisiLolZaidejai[i].nuzudymai +=
zaidimai[j].lolzaidejuMasyvas[k].nuzudymai;
            VisiLolZaidejai[i].mirtys +=
zaidimai[j].lolzaidejuMasyvas[k].mirtys;
            VisiLolZaidejai[i].dalyvNuzud +=
zaidimai[j].lolzaidejuMasyvas[k].dalyvNuzud;
        }
    }
}
for (int i = 0; i < lolcount; i++)
{
    VisiLolZaidejai[i].KDA = (VisiLolZaidejai[i].nuzudymai +
VisiLolZaidejai[i].dalyvNuzud) / VisiLolZaidejai[i].mirtys;
}
LOLZaidejas GeriausiasKDA = new LOLZaidejas();
GeriausiasKDA.KDA = 0;
for (int i = 0; i < lolcount; i++)
{
    if (VisiLolZaidejai[i].KDA > GeriausiasKDA.KDA)
    {
        GeriausiasKDA = VisiLolZaidejai[i];
    }
}
Console.WriteLine("-----");
Console.WriteLine("Geriausias asmeninis rezultatas (LOL zaidejo)");
Console.WriteLine("-----");
Console.WriteLine(" Vardas      | Pavarde      | Komanda      ");
Console.WriteLine("-----");
Console.WriteLine(" {0,-11} | {1,-12} | {2,-12}", GeriausiasKDA vardas,
GeriausiasKDA.pavarde, GeriausiasKDA.komanda);
Console.WriteLine("-----");
Console.WriteLine();

for (int i = 0; i < cscount; i++)
{
    VisiCsZaidejai[i].nuzudymai = 0;
    VisiCsZaidejai[i].mirtys = 0;
    for (int j = 0; j < RatuSkaicius; j++)
    {
        for (int k = 0; k < zaidimai[j].cszaidCount; k++)
        {
            if (VisiCsZaidejai[i].pavarde ==
zaidimai[j].cszaidejuMasyvas[k].pavarde)
            {
                VisiCsZaidejai[i].nuzudymai +=
zaidimai[j].cszaidejuMasyvas[k].nuzudymai;
                VisiCsZaidejai[i].mirtys +=
zaidimai[j].cszaidejuMasyvas[k].mirtys;
            }
        }
    }
}
for (int i = 0; i < cscount; i++)
{
    VisiCsZaidejai[i].KD = (VisiCsZaidejai[i].nuzudymai /
VisiCsZaidejai[i].mirtys);
}
CSZaidejas GeriausiasKD = new CSZaidejas();
GeriausiasKD.KD = 0;
for (int i = 0; i < cscount; i++)
{

```

```

        if (VisiCsZaidejai[i].KD > GeriausiasKD.KD)
        {
            GeriausiasKD = VisiCsZaidejai[i];
        }
    }
    Console.WriteLine("-----");
    Console.WriteLine("Geriausias asmeninis rezultatas (CS zaidejo)");
    Console.WriteLine("-----");
    Console.WriteLine(" Vardas      | Pavarde      | Komanda      ");
    Console.WriteLine("-----");
    Console.WriteLine(" {0,-11} | {1,-12} | {2,-12}", GeriausiasKD.vardas,
GeriausiasKD.pavarde, GeriausiasKD.komanda);
    Console.WriteLine("-----");
}

/// <summary>
/// Randa unievršalius zaidejus, zaidziancius abiejuose turnyruose ir
surikiuoja pagal pavarde
/// </summary>
/// <param name="zaidimai">Konteinerines klases objektas</param>
/// <param name="VisiLolZaidejai">Lol zaideju objektas</param>
/// <param name="lolcount">skaitliukas, kiek zaideju</param>
/// <param name="VisiCsZaidejai">Cs zaideju objektas</param>
/// <param name="cscount">skaitliukas, kiek zaideju</param>
public static void UniversalusZaidejai(Konteinerine[] zaidimai,
LOLZaidejas[] VisiLolZaidejai, int lolcount, CSZaidejas[] VisiCsZaidejai, int
cscount)
{
    Zaidejas[] UniversalusZaidejai = new Zaidejas[MaxZaideju];
    int count = 0;

    for (int i = 0; i < lolcount; i++)
    {
        for (int j = 0; j < cscount; j++)
        {
            if ((VisiLolZaidejai[i].pavarde == VisiCsZaidejai[j].pavarde)
&& (VisiLolZaidejai[i].vardas == VisiCsZaidejai[j].vardas))
            {
                Zaidejas laikinas = VisiCsZaidejai[j];
                bool yra = false;
                for (int k = 0; k < count; k++)
                {
                    if (laikinas.Equals(UniversalusZaidejai[k]))//if
(UniversalusZaidejai[k].Equals(laikinas))
                    {
                        yra = true;
                    }
                }
                if (!yra)//(!UniversalusZaidejai.Contains(laikinas))
                {
                    UniversalusZaidejai[count] = VisiCsZaidejai[j];
                    count++;
                }
            }
        }
    }
    Zaidejas[] Keitimas = new Zaidejas[MaxZaideju];

    for (int i = 0; i < count; i++)
        for (int j = i + 1; j < count; j++)
        {
            if (UniversalusZaidejai[i] <= UniversalusZaidejai[j])
            {
                Keitimas[0] = UniversalusZaidejai[i];
            }
        }
}

```



```

        UniversalusZaidejai[i] = UniversalusZaidejai[j];
        UniversalusZaidejai[j] = Keitimas[0];
    }

    }

    using (StreamWriter writer = new StreamWriter(@"Universalus.csv"))
    {
        writer.WriteLine("Universalus zaidejai, zaide abiejuose
zaidimuose");
        for (int i = 0; i < count; i++)
        {
            writer.WriteLine("{0}, {1}", UniversalusZaidejai[i].vardas,
UniversalusZaidejai[i].pavarde);
        }
    }

    }

    /// <summary>
    /// Paraso visas komantas
    /// </summary>
    /// <param name="zaidimai">Konteinerines klases objektas</param>
    public static void KomanduSarasas(Konteinerine[] zaidimai)
    {
        string[] komanduSarasas1 = new string[MaxZaideju];
        int count1 = 0;
        string[] komanduSarasas2 = new string[MaxZaideju];
        int count2 = 0;

        for (int i = 0; i < RatuSkaicius; i++)
        {
            for (int j = 0; j < zaidimai[i].lolzaidejuCount; j++)
            {
                if
(!komanduSarasas1.Contains(zaidimai[i].lolzaidejuMasyvas[j].komanda))
                {
                    komanduSarasas1[count1++] =
zaidimai[i].lolzaidejuMasyvas[j].komanda;
                }
            }
            for (int k = 0; k < zaidimai[i].cszaidejuCount; k++)
            {
                if
(!komanduSarasas2.Contains(zaidimai[i].cszaidejuMasyvas[k].komanda))
                {
                    komanduSarasas2[count2++] =
zaidimai[i].cszaidejuMasyvas[k].komanda;
                }
            }
        }
        using (StreamWriter writer = new StreamWriter(@"Komandos.csv"))
        {
            writer.WriteLine("Lol zaideju komandos");
            for (int i = 0; i < count1; i++)
            {
                writer.WriteLine(",{0}", komanduSarasas1[i]);
            }
            writer.WriteLine();
            writer.WriteLine("Cs zaideju komandos");
            for (int i = 0; i < count2; i++)
            {
                writer.WriteLine(",{0}", komanduSarasas2[i]);
            }
        }
    }

    }
}

```

}

#### **4.1. Pradiniai duomenys ir rezultatai**

##### **4.1.1 Pradiniai duomenys**

1

2015-10-03

L, Arnas, Sofauskas, FFBL, Top, Jarvan IV, 3, 1, 5  
L, Gaile, Pagalvyte, FFBL, Mid, Annie, 1, 1, 5  
L, Jonas, Kedzius, FFBL, AD, Ashe, 3, 1, 5  
L, Radvile, Stalciukaite, FFBL, Support, Taric, 0, 1, 8  
L, Ignas, Lovauskas, FFBL, Jungle, Amumu, 1, 1, 6  
C, Indre, Langaite, Rainbow Dash, 8, 1, sakute  
C, Dainius, Lentauskas, Rainbow Dash, 10, 1, kocelas  
C, Vytenis, Deziauskas, Rainbow Dash, 2, 1, tankas  
C, Raigardas, Knygius, Rainbow Dash, 1, 2, peilis  
L, Giedrius, Palangiauskas, FuriKuri, Top, Annie, 3, 1, 10  
L, Audrius, Dalgiauskas, FuriKuri, Mid, Ahri, 6, 1, 8  
L, Gintaras, Grebliauskas, FuriKuri, AD, Corki, 3, 2, 8  
L, Jomante, Deklaite, FuriKuri, Support, Sonna, 1, 4, 13  
L, Simona, Tinklaite, Girls United, Mid, Annie, 5, 1, 8  
L, Rasa, Plaktukaite, Girls United, Support, Soraka, 1, 1, 14  
L, Raminta, Foteliute, Girls United, Jungle, Udyr, 1, 1, 7  
C, Antanas, Stiklius, 1337, 1, 5, pagalys  
C, Jonas, Puodzius, 1337, 3, 1, lankas  
C, Algirdas, Kalvaitis, 1337, 3, 1, kardas  
C, Jurgis, Siauciunas, 1337, 1, 1, dalgis

2

2015-10-05

L, Arnas, Sofauskas, FFBL, Top, Jarvan IV, 3, 1, 5  
L, Gaile, Pagalvyte, FFBL, Mid, Annie, 1, 1, 5  
L, Jonas, Kedzius, FFBL, AD, Ashe, 3, 1, 5  
L, Radvile, Stalciukaite, FFBL, Support, Taric, 2, 1, 8  
L, Ignas, Lovauskas, FFBL, Jungle, Amumu, 1, 1, 6  
C, Indre, Langaite, Rainbow Dash, 8, 1, sakute  
C, Arnas, Sofauskas, Rainbow Dash, 5, 1, automatas  
C, Dainius, Lentauskas, Rainbow Dash, 6, 1, kocelas  
C, Vytenis, Deziauskas, Rainbow Dash, 2, 1, tankas  
C, Raigardas, Knygius, Rainbow Dash, 1, 2, peilis  
L, Giedrius, Palangiauskas, FuriKuri, Top, Annie, 3, 1, 10  
L, Audrius, Dalgiauskas, FuriKuri, Mid, Ahri, 6, 1, 8  
L, Gintaras, Grebliauskas, FuriKuri, AD, Corki, 3, 2, 8  
L, Jomante, Deklaite, FuriKuri, Support, Sonna, 4, 4, 13  
L, Indre, Stiklyte, Girls United, AD, Vayne, 9, 2, 8  
L, Simona, Tinklaite, Girls United, Mid, Annie, 5, 1, 8  
L, Antanas, Stiklius, Girls United, AD, Vayne, 9, 1, 8  
L, Rasa, Plaktukaite, Girls United, Support, Soraka, 1, 1, 14  
L, Raminta, Foteliute, Girls United, Jungle, Udyr, 1, 1, 7  
C, Antanas, Stiklius, 1337, 1, 4, pagalys  
C, Jonas, Puodzius, 1337, 3, 1, lankas  
C, Algirdas, Kalvaitis, 1337, 3, 1, kardas  
C, Jurgis, Siauciunas, 1337, 1, 1, dalgis

3

2015-10-07

L, Arnas, Sofauskas, FFBL, Top, Jarvan IV, 3, 1, 5

L,Jonas,Kedzius,FFBL,AD,Ashe,3,1,5  
 L,Radvile,Stalciukaite,FFBL,Support,Taric,1,1,8  
 L,Ignas,Lovauskas,FFBL,Jungle,Amumu,1,1,6  
 C,Indre,Langaite,Rainbow Dash,8,1,sakute  
 C,Arnas,Sofauskas,Rainbow Dash,5,1,automatas  
 C,Dainius,Lentauskas,Rainbow Dash,6,1,kocelas  
 C,Vytenis,Deziauskas,Rainbow Dash,2,1,tankas  
 C,Raigardas,Knygius,Rainbow Dash,1,2,peilis  
 L,Giedrius,Palangiauskas,FuriKuri,Top,Annie,3,1,10  
 L,Audrius,Dalgiauskas,FuriKuri,Mid,Ahri,6,1,8  
 L,Gintaras,Grebliauskas,FuriKuri,AD,Corki,3,2,8  
 L,Jomante,Deklaite,FuriKuri,Support,Sonna,4,4,13  
 L,Indre,Stiklyte,Girls United,AD,Vayne,9,2,8  
 L,Simona,Tinklaite,Girls United,Mid,Annie,5,1,8  
 L,Antanas,Stiklius,Girls United,AD,Vayne,9,1,8  
 L,Rasa,Plaktukaite,Girls United,Support,Soraka,1,1,14  
 L,Raminta,Foteliute,Girls United,Jungle,Udyr,1,1,7  
 C,Antanas,Stiklius,1337,1,1,pagalys  
 C,Algirdas,Kalvaitis,1337,3,1,kardas  
 C,Jurgis,Siauciunas,1337,20,1,dalgis

#### 4.1.2 Rezultatai

Failas: cmd.exe

```

C:\windows\system32\cmd.exe
Geriausias asmeninis rezultatas <LOL zaidejo>
-----
Vardas      ! Pavarde      ! Komanda
-----
Antanas     ! Stiklius     ! Girls United
-----

Geriausias asmeninis rezultatas <CS zaidejo>
-----
Vardas      ! Pavarde      ! Komanda
-----
Jurgis      ! Siauciunas   ! 1337
-----
Press any key to continue . . .
  
```

Failas: Komandos.csv

	A	B	C
	Lol zaideju komandos		
!		FFBL	
!		FuriKuri	
!		Girls United	
i			
i	Cs zaideju komandos		
!		Rainbow Dash	
!		1337	
!			
^			

Failas: Universalus.csv

	A	B	C	D	E
1	Universalus zaidejai, zaide abiejuose zaidimuose				
2	Antanas	Stiklius			
3	Arnas	Sofauskas			
4					
5					

## **5. Susieti objektų rinkiniai**

### ***5.1. Darbo užduotis***

### ***5.2. Programos tekstas***

### ***5.3. Pradiniai duomenys ir rezultatai***

## 6. Teksto analizė ir redagavimas

### 6.1. Darbo užduotis

U5-7. Pasikartojimai

Dviejuose tekstiniuose failuose Knyga1.txt ir Knyga2.txt duotas tekstas sudarytas iš žodžių, atskirtų skyrikliais. Skyriklių aibė žinoma ir abiejuose failuose yra ta pati. Analizuojant tekstus, didžiosios ir mažosios raidės nesvarbios. Raskite, spausdinkite faile Analizė.txt ir išveskite ekrane teksto analizės rezultatus:

- ilgiausių žodžių, kurie yra tik faile Knyga1.txt, bet nėra faile Knyga2.txt, sąrašą (ne daugiau nei 10 žodžių) ir jų pasikartojimo skaičių;
- ilgiausią teksto fragmentą, sudarytą iš žodžių ir juos skiriančių skyriklių, kuris yra abiejuose failuose ir jo eilutės numerius pirmame ir antrame faile;

Spausdinkite faile ManoKnyga.txt apjungtą tekstą, sudarytą pagal tokias taisykles:

- kopijuojamas pirmojo failo tekstas tol, kol sutinkamas pirmasis antrojo failo žodis arba pasiekama failo pabaiga;
- kopijuojamas antrojo failo tekstas tol, kol sutinkamas pirmasis nenukopijuotas pirmojo failo žodis arba pasiekama failo pabaiga;
- kartojama tol, kol pasiekama abiejų failų pabaiga.

### 6.2. Programos tekstas

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

namespace ConsoleApplication1
{
    class Program
    {
        const int MaxZod = 1500000;

        static void Main(string[] args)
        {
            const string CF1d = "..\\..\\Knyga1.txt";
            const string CF2d = "..\\..\\Knyga2.txt";
            const string CFa = "..\\..\\Analyze.txt";
            const string CFm = "..\\..\\ManoKnyga.txt";

            string[] PirmoZodziai = new string[MaxZod];
            int pirmCount;
            string[] AntroZodziai = new string[MaxZod];
            int antrCount;

            Apdoroti(CF1d, CF2d, ref PirmoZodziai, out pirmCount, ref
AntroZodziai, out antrCount);

            ///<didziuju raidziu pavertimas i mazasias>
            string[] PirmMazosiom = new string[pirmCount];
            for (int i = 0; i < pirmCount; i++)
            {
                string zod = PirmoZodziai[i];
                PirmMazosiom[i] = zod.ToLower();
            }

            string[] AntrMazosiom = new string[antrCount];
            for (int i = 0; i < antrCount; i++)
            {
```

```

        string zod = Antrozodziai[i];
        AntrMazosiom[i] = zod.ToLower();
    }
    ///<----->

    ///<metodas, skirtas rasti ilgiasius zodzius, esancius tik pirmame
faile>
    string[] TikPirmoF = new string[MaxZod];///<zodziai esantys tik
pirmame faile, grazinami jau surikiuoti nuo ilgiausio>
    int[] Pasikartojimai = new int[MaxZod];
    IlgiausiZodziai(PirmMazosiom, pirmCount, AntrMazosiom, antrCount, ref
TikPirmoF, ref Pasikartojimai);

    string[] Fragmentas = new string[MaxZod];
    int countF;

    IlgiausiasFragmentas(PirmMazosiom, pirmCount, AntrMazosiom, antrCount,
ref Fragmentas, out countF);

    IsvedimasAnalyze(CF1d, CF2d, CFa, TikPirmoF, Pasikartojimai,
Fragmentas, countF);

    ManoKnyga(CFm, PirmMazosiom, pirmCount, AntrMazosiom, antrCount);
}

/// <summary>
/// Nuskaito duomenis, juos suraso i masyvus
/// </summary>
/// <param name="f1">pirmas duomenu failas</param>
/// <param name="f2">antras duomenu failas</param>
/// <param name="PirmoZodziai">Pirmo failo masyvas</param>
/// <param name="pirmCount">pirmo count</param>
/// <param name="Antrozodziai">Antro failo masyvas</param>
/// <param name="antrCount">antro count</param>
static void Apdoroti(string f1, string f2, ref string[] PirmoZodziai, out
int pirmCount, ref string[] Antrozodziai, out int antrCount)
{
    ///<zodzius skiriami tarpo skyrikliu>
    char[] skyrikliai = { ' ' };
    pirmCount = 0;
    antrCount = 0;
    ///<pirmo failo zodziai surasomi i masyva>
    string[] lines1 = File.ReadAllLines(f1, Encoding.GetEncoding(1257));

    foreach (string line1 in lines1)
    {
        if (line1.Length > 0)
        {
            string[] words = line1.Split(skyrikliai,
StringSplitOptions.RemoveEmptyEntries);
            foreach (var word in words)
            {
                PirmoZodziai[pirmCount++] = word;
            }
        }
    }
    ///<antrojo failo zodzius surasomi i masyva>
    string[] lines2 = File.ReadAllLines(f2, Encoding.GetEncoding(1257));
    foreach (string line2 in lines2)
    {
        if (line2.Length > 0)
        {
            string[] words = line2.Split(skyrikliai,
StringSplitOptions.RemoveEmptyEntries);
            foreach (var word in words)
            {

```

```

        Antrozodziai[antrCount++] = word;
    }
}

}

/// <summary>
/// Iesko ilgiausiu zodziu esanciu tik pirmame faile, juos suraso nuo
ilgiausio ir suskaiciuoja pasikartojimus
/// </summary>
/// <param name="PirmMazosiom">Pirmo failo masyvas</param>
/// <param name="pirmCount">pirmo count</param>
/// <param name="AntrMazosiom">Antro failo masyvas</param>
/// <param name="antrCount">antro count</param>
/// <param name="TikPirmoF">Masyvas zodziu esanciu tik pirmame
faile</param>
/// <param name="Pasikartojimai">kiekvieno zodzio pasikartojimu
skaicius</param>
static void IlgiausiZodziai(string[] PirmMazosiom, int pirmCount, string[]
AntrMazosiom, int antrCount, ref string[] TikPirmoF, ref int[] Pasikartojimai)
{
    ///<zodziu, kurie yra tik pirmame faile raidimas>

    int count = 0;
    string zodis1;
    string zodis2;
    for (int i = 0; i < pirmCount; i++)
    {
        if (PirmMazosiom[i].EndsWith(",") || PirmMazosiom[i].EndsWith(".")
|| PirmMazosiom[i].EndsWith(":") || PirmMazosiom[i].EndsWith(";") ||
PirmMazosiom[i].EndsWith("?") || PirmMazosiom[i].EndsWith("!"))
        {
            zodis1 =
PirmMazosiom[i].TrimEnd(PirmMazosiom[i][PirmMazosiom[i].Length - 1]);
        }
        else zodis1 = PirmMazosiom[i];
        bool tikrina = true;
        for (int j = 0; j < antrCount; j++)
        {
            if (AntrMazosiom[j].EndsWith(",") ||
AntrMazosiom[j].EndsWith(".") || AntrMazosiom[j].EndsWith(":") ||
AntrMazosiom[j].EndsWith(";") || AntrMazosiom[j].EndsWith("?") ||
AntrMazosiom[j].EndsWith("!"))
            {
                zodis2 =
AntrMazosiom[j].TrimEnd(AntrMazosiom[j][AntrMazosiom[j].Length - 1]);
            }
            else zodis2 = AntrMazosiom[j];
            if (zodis1 == zodis2)
                tikrina = false;
        }
        if (tikrina)
            TikPirmoF[count++] = zodis1;
    }

    for (int i = 0; i < count; i++)
    {///<kadangi zodis pasikartoja bent viena karta>
        Pasikartojimai[i] += 1;
    }
    ///<istrina pasikartojimus ir suskaicuoja kiek ju yra>
    for (int i = 0; i < count - 1; i++)
    {
        for (int j = i + 1; j < count; j++)
        {
            if (TikPirmoF[i] == TikPirmoF[j])
            {
                for (int k = j; k < count; k++)

```

```

        {
            TikPirmoF[k] = TikPirmoF[k + 1];
        }
        count--;
        Pasikartojimai[i] += 1;
    }
}
}
///<pereina dar karta, jei uzsiliko nesuskaiciuotu>
for (int i = 0; i < count - 1; i++)
{
    for (int j = i + 1; j < count; j++)
    {
        if (TikPirmoF[i] == TikPirmoF[j])
        {
            for (int k = j; k < count; k++)
            {
                TikPirmoF[k] = TikPirmoF[k + 1];
            }
            count--;
            Pasikartojimai[i] += 1;
        }
    }
}
}
///<sudelioti pagal ilguma>
string pagalbinis;
int skaicius;
for (int i = 0; i < count; i++)
{
    for (int j = 0; j < count; j++)
    {
        if (TikPirmoF[i].Length > TikPirmoF[j].Length)
        {
            pagalbinis = TikPirmoF[i];
            TikPirmoF[i] = TikPirmoF[j];
            TikPirmoF[j] = pagalbinis;

            skaicius = Pasikartojimai[i];
            Pasikartojimai[i] = Pasikartojimai[j];
            Pasikartojimai[j] = skaicius;
        }
    }
}
}

/// <summary>
/// Iesko ilgiausio fragmento
/// </summary>
/// <param name="PirmMazosiom">Pirmo failo masyvas</param>
/// <param name="pirmCount">pirmas count</param>
/// <param name="AntrMazosiom">Antro failo masyvas</param>
/// <param name="antrCount">antro count</param>
/// <param name="Fragmentas">Fragmento masyvas</param> <suraso zodzius su
skyrikliais kaip i masyva>
/// <param name="countF">fragmento zodziu count</param>
static void IlgiausiasFragmentas(string[] PirmMazosiom, int pirmCount,
string[] AntrMazosiom, int antrCount, ref string[] Fragmentas, out int countF)
{
    countF = 0;
    int index = 0; int indexMax = 0;
    int max = 0;
    int zodziuSk = 0;
    int nuoKurio = 0;
    bool tikrina = true;

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

```



```

        {
            for (int j = 0; j < antrCount; j++)
            {
                tikrina = true;
                if (PirmMazosiom[i] == AntrMazosiom[j])
                {
                    int k = 1;
                    index = i;

                    while ((tikrina) && (i+k < pirmCount) && (j+k <
antrCount))

                        if (PirmMazosiom[i + k] == AntrMazosiom[j + k])
                        {
                            indexMax = index + k;
                            k += 1;
                        }
                        else tikrina = false;
                    zodziuSk = indexMax - index;
                    if (zodziuSk > max)
                    {
                        nuoKurio = index;
                        max = zodziuSk;
                    }
                }
            }
            int maxis = max + nuoKurio + 1;
            for (int n = nuoKurio; n < maxis; n++)
            {
                Fragmentas[countF++] = PirmMazosiom[n];
            }
        }
        /// <summary>
        /// Isvedimas i analizes faila
        /// </summary>
        /// <param name="f1">pirmas failas</param> <skirtas surasti kurioje eilute
yra fragmentas>
        /// <param name="f2">antraas failas</param>
        /// <param name="fa">analizes failas</param>
        /// <param name="TikPirmoF">Zodziai esantys tik pirmame faile</param>
        /// <param name="Pasikartojimai">pasikartojimu skaiciuos kiekvieno
zodzio</param>
        /// <param name="Fragmentas">Ilgiausias fragmentas</param>
        /// <param name="countF">fragmento zodzio count</param>
        static void IsvedimasAnalyze(string f1, string f2, string fa, string[]
TikPirmoF, int[] Pasikartojimai, string[] Fragmentas, int countF)
        {
            using (var far = File.CreateText(fa))
            {
                far.WriteLine("Tik pirmame faile esantys zodziai ir ju
pasikartojimo skaicius");
                far.WriteLine("-----");
                far.WriteLine("      Zodis      |  Pasikartojimai");
                far.WriteLine("-----");
                for (int i = 0; i < 10; i++)
                {
                    far.WriteLine("{0,-15} | {1,-5}", TikPirmoF[i],
Pasikartojimai[i]);
                }
                far.WriteLine("-----");
                far.WriteLine();
                far.WriteLine("Ilgiausias teksto fragmentas");
                far.WriteLine("-----");
                for (int i = 0; i < countF; i++)
                {
                    far.Write("{0} ", Fragmentas[i]);

```

```

    }
    far.WriteLine();
    far.WriteLine("-----");

    char[] skyrikliai = { ' ' };
    int eil = 0;
    int count = 0;
    int[] eilute = new int[MaxZod];
    bool tinka = true;
    string[] lines1 = File.ReadAllLines(f1,
Encoding.GetEncoding(1257));

    foreach (string line1 in lines1)
    {
        eil += 1;
        if (line1.Length > 0)
        {
            string[] words = line1.Split(skyrikliai,
StringSplitOptions.RemoveEmptyEntries);
            foreach (var word in words)
            {
                for (int i = 0; i < countF; i++)
                {
                    if (Fragmentas[i] == word)
                    {
                        tinka = true;
                        for (int j = 0; j < count; j++)
                        {
                            if (eilute[j] == eil)
                                tinka = false;
                        }
                        if (tinka)
                        {
                            eilute[count++] = eil;
                        }
                    }
                }
            }
        }
    }

    far.Write("Eilutes numeris(iai) pirname faile: ");
    for (int i = 0; i < count; i++)
    {
        far.Write("{0} ", eilute[i]);
    }

    string[] lines2 = File.ReadAllLines(f2,
Encoding.GetEncoding(1257));
    eil = 0;
    int count2 = 0;
    int[] eilute2 = new int[MaxZod];
    foreach (string line2 in lines2)
    {
        eil += 1;
        if (line2.Length > 0)
        {
            string[] words = line2.Split(skyrikliai,
StringSplitOptions.RemoveEmptyEntries);
            foreach (var word in words)
            {
                for (int i = 0; i < countF; i++)
                {
                    if (Fragmentas[i] == word)
                    {
                        tinka = true;

```

```

        for (int j = 0; j < count; j++)
        {
            if (eilute2[j] == eil)
                tinka = false;
        }
        if (tinka)
        {
            eilute2[count2++] = eil;
        }
    }
}

}

}

far.WriteLine();
far.Write("Eilutes numeris(iai) antrame faile: ");
for (int i = 0; i < count2; i++)
{
    far.Write("{0} ", eilute2[i]);
}
}

}

/// <summary>
/// Sujungia du failus i viena
/// </summary>
/// <param name="fm">failas ManoKnyga</param>
/// <param name="PirmMazosiom">pirmo failo masyvas</param>
/// <param name="pirmCount">pirmo count</param>
/// <param name="AntrMazosiom">Antro failo masyvas</param>
/// <param name="antrCount">antro count</param>
static void ManoKnyga(string fm, string[] PirmMazosiom, int pirmCount,
string[] AntrMazosiom, int antrCount)
{
    int i = 0; int j = 0;
    bool tikrina = true;
    bool galas = false;
    string[] ManoKnyga = new string[MaxZod];
    int countKnyga = 0;

    while ((i < antrCount) && (j < pirmCount) && (!galas))
    {
        if (PirmMazosiom[j] != AntrMazosiom[i])
        {
            ManoKnyga[countKnyga++] = PirmMazosiom[j];
            j += 1;
        }
        else
        {
            ManoKnyga[countKnyga++] = PirmMazosiom[j];

            int k = j + 1;
            int l = i + 1; j += 1; i += 1;
            tikrina = true;

            while ((k < pirmCount) && (l < antrCount) && (tikrina) &&
(!galas))
            {
                if (PirmMazosiom[k] != AntrMazosiom[l])
                {
                    ManoKnyga[countKnyga++] = AntrMazosiom[l];

                    l += 1;
                    i += 1;
                    if (i == antrCount)
                    {

```

```

        for (int n = j; n < pirmCount; n++)
        {
            ManoKnyga[countKnyga++] = PirmMazosiom[n];
        }
    }
    if (j == pirmCount)
    {
        for (int n = i; n < antrCount; n++)
        {
            ManoKnyga[countKnyga++] = AntrMazosiom[n];
        }
    }
}
else
{
    ManoKnyga[countKnyga++] = AntrMazosiom[l];
    j += 1; i += 1;
    tikrina = false;
    if (i == antrCount)
    {
        galas = true;
        for (int n = j; n < pirmCount; n++)
        {
            ManoKnyga[countKnyga++] = PirmMazosiom[n];
        }
    }
    if (j == pirmCount)
    {
        galas = true;
        for (int n = i; n < antrCount; n++)
        {
            ManoKnyga[countKnyga++] = AntrMazosiom[n];
        }
    }
}
}

}

}

using (var far = File.CreateText(fm))
{
    for (int a = 0; a < countKnyga; a++)
    {
        far.Write("{0} ", ManoKnyga[a]);

        if ((a != 0) && (a % 10 == 0))
            far.WriteLine();
    }
}
}
}
}

```

### 6.3. Pradiniai duomenys ir rezultatai

#### 6.3.1 Pradiniai duomenys

Failas: Knyga1.txt

gyveno bobute ir diedukas. Jie  
turejo tris anukus, kurie mego valgyti braskes ir buvo pasieme

mergaite. Atskirido varna ir sulese visa derliu. Tad  
zmones liko be nieko.

Failas: Knyga2.txt

Gyveno bobute Jie ejo pasivikscioti ir turejo  
pasieme daug saldainiu, kuriuos mergaite.  
Tad ji pasaulyje atejo taika valge po  
saldaini kiekviena diena.

### 6.3.2 Rezultatai

Failas: Analize.txt

Tik pirmame faile esantys zodziai ir ju pasikartojimo skaicius

-----	
Zodis	Pasikartojimai
-----	
atskirido	1
diedukas	1
braskes	1
valgyti	1
anukus	1
sulese	1
derliu	1
zmones	1
varna	1
kurie	1
-----	

Ilgiausias teksto fragmentas

-----  
gyveno bobute  
-----

Eilutes numeris(iai) pirmame faile: 1  
Eilutes numeris(iai) antrame faile: 1

Failas: ManoKnyga.txt

gyveno bobute ir diedukas. jie ejo pasivikscioti ir turejo tris anukus,  
kurie mego valgyti braskes ir buvo pasieme daug saldainiu, kuriuos  
mergaite. atskirido varna ir sulese visa derliu. tad ji pasaulyje  
atejo taika valge po saldaini kiekviena diena. zmones liko be  
nieko.

## 7. Sudėtingesnis konteineris

### 7.1. Darbo užduotis

U6-7. Darbininkai Pirmoje failo eilutėje nurodytas darbininkų skaičius, mėnesio dirbtų dienų skaičius, vienos detalės įkainis. Tolesnėse eilutėse pateikta informacija apie darbininkus: pavardė, vardas, banko pavadinimas ir sąskaitos numeris. Žemiau pateikta, kiek kiekvieną dieną (eilutės) darbininkas (stulpeliai) pagamino detalių.

- Suskaičiuokite kiekvienam darbininkui jo atlyginimą.
- Nustatykite, kuriam darbininkui mažiausiai sekėsi dirbti.
- Nustatykite, kurią mėnesio dieną buvo pagaminta daugiausiai detalių.
- Kiekvienam bankui atskirai sudarykite pavedimų sąrašą, kur nurodysite darbininko pavardę, vardą, sąskaitos numerį ir pervedamą sumą.

Žemiau pateikta kito mėnesio rezultatai: pirmojoje eilutėje - dirbtų dienų skaičius, toliau - kiek kiekvieną dieną (eilutės) darbininkas (stulpeliai) pagamino detalių.

- Atlikite tuos pačius veiksmus, kaip ir su pirmuoju mėnesiu.
- Nustatykite, ar pirmojo mėnesio silpniausiai dirbantis darbininkas pagerino rezultatus.

### 7.2. Programos tekstas

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

namespace ConsoleApplication1
{
    class Darbininkas
    {
        public string vardas { get; set; }
        public string pavarde { get; set; }
        public string bankas { get; set; }
        public string bankSask { get; set; }
        public int detKaina { get; set; }
        public int alga { get; set; } // pirmo mėnesio alga
        public int alga2 { get; set; } // antro mėnesio alga

        public Darbininkas()
        {
        }

        public Darbininkas(string vardas, string pavarde, string bankas, string
bankSask)
        {
            this.vardas = vardas;
            this.pavarde = pavarde;
            this.bankas = bankas;
            this.bankSask = bankSask;
        }

        public Darbininkas(int detKaina)
        {
            this.detKaina = detKaina;
        }
    }

    class DetaliuKiekis
    {
        public int detaliuSk { get; set; }
        public DetaliuKiekis()
        {
        }
    }
}
```

```

        {
            detaliuSk = 0;
        }
        public DetaliuKiekis(int detaliuSk)
        {
            this.detaliuSk = detaliuSk;
        }
    }

class Konteinerine
{
    public Darbininkas[] darbininkoMasyvas { get; set; }
    public Darbininkas[] darbininkoMasyvas2 { get; set; }
    int countdarb;

    public Konteinerine(int kiek)
    {
        countdarb = 0;
        darbininkoMasyvas = new Darbininkas[kiek];
        darbininkoMasyvas2 = new Darbininkas[kiek];
    }

    public void DetiDarbInfo(Darbininkas darbininkas) ///<sudeda pavarde,
varda, banko pav ir banko sask nr>
    {
        darbininkoMasyvas2[countdarb] = darbininkas;
        darbininkoMasyvas[countdarb++] = darbininkas;
    }

    const int MaxDarbininku = 1000;
    const int MaxDetaliu = 1000;
    const int MaxDienu = 150;

    private DetaliuKiekis[,] DetaliuMasyvas;
    private DetaliuKiekis[,] DetaliuMasyvas2;

    public int dienSk { get; set; }
    public int detKaina { get; set; }
    public int darbSk { get; set; }
    public int dienuSk2 { get; set; }

    public Konteinerine()
    {
        dienSk = 0; darbSk = 0;
        DetaliuMasyvas = new DetaliuKiekis[MaxDienu, MaxDetaliu];
        DetaliuMasyvas2 = new DetaliuKiekis[MaxDienu, MaxDetaliu];
    }

    public void DetiDetaliuInfo(int i, int j, DetaliuKiekis detaliuSk)
    {
        DetaliuMasyvas[i, j] = detaliuSk;
    }
    public void DetiDetaliuInfo2(int i, int j, DetaliuKiekis detaliuSk2)
    {
        DetaliuMasyvas2[i, j] = detaliuSk2;
    }
    public DetaliuKiekis Imti(int i, int j)
    {
        return DetaliuMasyvas[i, j];
    }
    public DetaliuKiekis Imti2(int i, int j)
    {
        return DetaliuMasyvas2[i, j];
    }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        const int MaxDarbininku = 1000;

        const string CDd = "..\\..\\Duomenys.txt";
        const string CRr = "..\\..\\Rezultatai.txt";
        File.Delete(CRr);
        Konteinerine dvimatis = new Konteinerine();
        Konteinerine visiDarbininkai = new Konteinerine(MaxDarbininku);
        Skaityti(CDd, dvimatis, visiDarbininkai);

        ///<pirmo menesio info>
        bool pirmas = true;
        Atlyginimas(dvimatis, visiDarbininkai, dvimatis.dienSk, pirmas);
        int index;
        MaziausiaiSekesi(dvimatis, visiDarbininkai, out index, pirmas);
        int diena;
        DaugiausiaiPagamintaDetaliu(dvimatis, visiDarbininkai, out diena,
dvimatis.dienSk, pirmas);
        int nr = 1;
        SurasoIFaila(dvimatis, visiDarbininkai, index, diena, CRr, nr,
pirmas);

        ///<antro menesio>
        pirmas = false;
        Atlyginimas(dvimatis, visiDarbininkai, dvimatis.dienuSk2, pirmas);
        int index2;
        MaziausiaiSekesi(dvimatis, visiDarbininkai, out index2, pirmas);
        int diena2;
        DaugiausiaiPagamintaDetaliu(dvimatis, visiDarbininkai, out diena2,
dvimatis.dienuSk2, pirmas);
        nr = 2;
        SurasoIFaila(dvimatis, visiDarbininkai, index2, diena2, CRr, nr,
pirmas);

        PavedimuSarasas(dvimatis, visiDarbininkai, CRr);
        using (var writer = File.AppendText(CRr))
        {
            writer.WriteLine("Silpniausias pirmo men darbuotojas - {0} {1}",
visiDarbininkai.darbininkoMasyvas[index].vardas,
visiDarbininkai.darbininkoMasyvas[index].pavarde);
            writer.WriteLine("pirma menesi uzdirbo: {0}, o antraji: {1}",
visiDarbininkai.darbininkoMasyvas[index].alga,
visiDarbininkai.darbininkoMasyvas[index].alga2);
            if (visiDarbininkai.darbininkoMasyvas[index].alga <
visiDarbininkai.darbininkoMasyvas[index].alga2)
                writer.WriteLine("Darbuotojas pagerino praejusio menesio
rezultatus");
            else writer.WriteLine("Darbuotojas nepagerino praejusio menesio
rezultatu");
        }
        }
        /// <summary>
        /// Nuskaityto duomenis is duomeny failo
        /// </summary>
        /// <param name="f1">pradinis duomenu failas</param>
        /// <param name="dvimatis">dvimatis objektas</param>
        /// <param name="visiDarbininkai">objektas</param>
        static void Skaityti(string f1, Konteinerine dvimatis, Konteinerine
visiDarbininkai)
        {
            string line;
            using (StreamReader reader = new StreamReader(f1))
            {

```



```

line = reader.ReadLine();
string[] fd = line.Split(' ');
dvmatis.darbSk = int.Parse(fd[0]); //stulpeliu sk - darbuot sk
dvmatis.dienSk = int.Parse(fd[1]); //eiluciu sk = dienu sk
dvmatis.detKaina = int.Parse(fd[2]); //detales kaina

for (int i = 0; i < dvmatis.darbSk; i++)
{
    line = reader.ReadLine();

    string[] parts = line.Split(' ');
    string pavarde = parts[0];
    string vardas = parts[1];
    string bankas = parts[2];
    string bankSask = parts[3];
    Darbininkas darb = new Darbininkas(pavarde, vardas, bankas,
bankSask);

    visiDarbininkai.DetiDarbInfo(darb);
}
for (int i = 0; i < dvmatis.dienSk; i++)
{
    line = reader.ReadLine();
    fd = line.Split(' ');
    for (int j = 0; j < dvmatis.darbSk; j++)
    {
        int detSk = int.Parse(fd[j]); // <detaliu skaicius>
        DetaliuKiekis detaliuSk = new DetaliuKiekis(detSk);
        dvmatis.DetiDetaliuInfo(i, j, detaliuSk);
    }
}
line = reader.ReadLine();
string[] far = line.Split(' ');
dvmatis.dienuSk2 = int.Parse(far[0]);
for (int i = 0; i < dvmatis.dienuSk2; i++)
{
    line = reader.ReadLine();
    fd = line.Split(' ');
    for (int j = 0; j < dvmatis.darbSk; j++)
    {
        int detSk2 = int.Parse(fd[j]);
        DetaliuKiekis detaliuSk2 = new DetaliuKiekis(detSk2);
        dvmatis.DetiDetaliuInfo2(i, j, detaliuSk2);
    }
}
}
}
/// <summary>
/// Suskaiciuoja kiekvieno darbuotojo atlyginima
/// </summary>
/// <param name="dvmatis">dvmatis objektas</param>
/// <param name="visiDarbininkai">objektas</param>
/// <param name="count">dienu skaicius</param>
/// <param name="pirmas">jei primas menesis - true, jei antras
false</param>
static void Atlyginimas(Konteinerine dvmatis, Konteinerine
visiDarbininkai, int count, bool pirmas)
{
    for (int i = 0; i < dvmatis.darbSk; i++)
    {
        for (int j = 0; j < count; j++)
        {
            if (pirmas)
                visiDarbininkai.darbininkoMasyvas[i].alga +=
dvmatis.Imti(j, i).detaliuSk * dvmatis.detKaina;
            else visiDarbininkai.darbininkoMasyvas[i].alga2 +=
dvmatis.Imti2(j, i).detaliuSk * dvmatis.detKaina;

```

```

    }
}

/// <summary>
/// Randa darbuotoja, kuriam maziausiai sekese
/// </summary>
/// <param name="dvimatis">dvimatis objektas</param>
/// <param name="visiDarbininkai">objektas</param>
/// <param name="ind">indekstas, numeris eiles</param>
/// <param name="pirmas">jei primas menesis - true, jei antras
false</param>
static void MaziausiaiSekesi(Konteinerine dvimatis, Konteinerine
visiDarbininkai, out int ind, bool pirmas)
{
    ind = 0;
    for (int j = 1; j < dvimatis.darbSk; j++)
    {
        if (pirmas)
            if (visiDarbininkai.darbininkoMasyvas[ind].alga >
visiDarbininkai.darbininkoMasyvas[j].alga)
                ind = j;
            if ((!pirmas) && (visiDarbininkai.darbininkoMasyvas[ind].alga2 >
visiDarbininkai.darbininkoMasyvas[j].alga2))
                ind = j;
    }
}

/// <summary>
/// Randa, kuria diena buvo pagamiinta daugiausiai detaliu
/// </summary>
/// <param name="dvimatis">dvimatis objektas</param>
/// <param name="visiDarbininkai">objektas</param>
/// <param name="diena">grazina daugiausiai pelno atnesusia diena</param>
/// <param name="count">dienu skaicius</param>
/// <param name="pirmas">jei primas menesis - true, jei antras
false</param>
static void DaugiausiaiPagamintaDetaliu(Konteinerine dvimatis,
Konteinerine visiDarbininkai, out int diena, int count, bool pirmas)
{
    diena = 0;
    int maxDet = 0;
    int suma;
    for (int i = 0; i < count; i++)
    {
        suma = 0;
        for (int j = 0; j < dvimatis.darbSk; j++)
        {
            if (pirmas)
                suma += dvimatis.Imti(i, j).detaliuSk;
            else suma += dvimatis.Imti2(i, j).detaliuSk;
        }
        if (maxDet < suma)
        {
            diena = i;
            maxDet = suma;
        }
    }
}

/// <summary>
/// Suraso visus duomenis i rezultatu faila
/// </summary>
/// <param name="dvimatis">dvimatis objektas</param>
/// <param name="visiDarbininkai">objektas</param>
/// <param name="index">labiausiai nepasisekusio darbuotojo
indeksas</param>
/// <param name="diena">daugiausiai pelno atnesusi diena</param>
/// <param name="f2">rezultatu failas</param>

```

```

    /// <param name="nr">menesio numeris</param>
    /// <param name="pirmas">jei pirmas menesis - true, jei antras
false</param>
    static void SurasoIFaila(Konteinerine dvimatis, Konteinerine
visiDarbininkai, int index, int diena, string f2, int nr, bool pirmas)
    {
        using (var writer = File.AppendText(f2))
        {
            writer.WriteLine("-----");
            writer.WriteLine("| {0} MENESIS |", nr);
            writer.WriteLine("-----");
            writer.WriteLine();
            writer.WriteLine("Kiekvieno darbininko atlyginimas:");
            writer.WriteLine("-----");
            writer.WriteLine("Vardas      | Pavarde      | Alga");
            writer.WriteLine("-----");
            if (pirmas)
                for (int i = 0; i < dvimatis.darbSk; i++)
                    writer.WriteLine("{0,-10} | {1,-10} | {2,-5}",
visiDarbininkai.darbininkoMasyvas[i].vardas,
visiDarbininkai.darbininkoMasyvas[i].pavarde,
visiDarbininkai.darbininkoMasyvas[i].alga);

                else for (int i = 0; i < dvimatis.darbSk; i++)
                    writer.WriteLine("{0,-10} | {1,-10} | {2,-5}",
visiDarbininkai.darbininkoMasyvas[i].vardas,
visiDarbininkai.darbininkoMasyvas[i].pavarde,
visiDarbininkai.darbininkoMasyvas[i].alga2);
            writer.WriteLine("-----");
            writer.WriteLine("Labiausiai nesiseke dirbti darbuotojai");
            writer.WriteLine("vardu {0} {1} ",
visiDarbininkai.darbininkoMasyvas[index].vardas,
visiDarbininkai.darbininkoMasyvas[index].pavarde);
            writer.WriteLine("-----");
            writer.WriteLine("{0} diena buvo pagaminta daugiausiai detaliu",
diena + 1);

            writer.WriteLine("-----");
            writer.WriteLine();
        }
    }

    /// <summary>
    /// Paraso i rezultatu faila pavedimu i banka sarasa
    /// </summary>
    /// <param name="dvimatis">dvimatis objektas</param>
    /// <param name="visiDarbininkai">objektas</param>
    /// <param name="f2">rezultatu failas</param>
    static void PavedimuSarasas(Konteinerine dvimatis, Konteinerine
visiDarbininkai, string f2)
    {
        int bankuCount = 0;
        string[] Bankai = new string[dvimatis.darbSk];
        for (int i = 0; i < dvimatis.darbSk; i++)
        {
            if (!Bankai.Contains(visiDarbininkai.darbininkoMasyvas[i].bankas))
                Bankai[bankuCount++] =
visiDarbininkai.darbininkoMasyvas[i].bankas;
        }
        using (var writer = File.AppendText(f2))
        {
            writer.WriteLine();
            for (int i = 0; i < bankuCount; i++)
            {

                writer.WriteLine("-----
-----");
            }
        }
    }

```



Vardas	Pavarde	Alga
Juozukas	Juozas	1008
Sniegelis	Sniegas	756
Saulute	Saule	784
Onute	Ona	686
Eglute	Egle	882
Batuotas	Katinas	770
Pinokis	Buratinas	168

Labiausiai nesiseke dirbti darbuotojui  
vardu Pinokis Buratinas

11 diena buvo pagaminta daugiausiai detaliu

| 2 MENESIS |

Kiekvieno darbininko atlyginimas:

Vardas	Pavarde	Alga
Juozukas	Juozas	308
Sniegelis	Sniegas	28
Saulute	Saule	280
Onute	Ona	252
Eglute	Egle	280
Batuotas	Katinas	224
Pinokis	Buratinas	112

Labiausiai nesiseke dirbti darbuotojui  
vardu Sniegelis Sniegas

1 diena buvo pagaminta daugiausiai detaliu

Banko pavadinimas: SEB

Vardas	Pavarde	Sask nr.	Pirmo men	Antro men
Juozas	Juozukas	LT001	1008eur.	308eur.
Egle	Eglute	LT005	882eur.	280eur.
Katinas	Batuotas	LT006	770eur.	224eur.

Banko pavadinimas: SNORAS

Vardas	Pavarde	Sask nr.	Pirmo men	Antro men
Sniegas	Sniegelis	LT002	756eur.	28eur.
Buratinas	Pinokis	LT007	168eur.	112eur.

Banko pavadinimas: Swedbankas

Vardas	Pavarde	Sask nr.	Pirmo men	Antro men
Saule	Saulute	LT003	784eur.	280eur.

Banko pavadinimas: DNB

Vardas	Pavarde	Sask nr.	Pirmo men	Antro men
Ona	Onute	LT004	686eur.	252eur.

Silpniausias pirmo men darbuotojas - Pinokis Buratinas  
 pirma menesi uzdirbo: 168, o antraji: 112  
 Darbuotojas nepagerino praejusio menesio rezultatu