

Kauno technologijos universitetas
Informatikos fakultetas

Objektinis programavimas I (P175B118)

Laboratorinių darbų ataskaita

Martynas Kuliešius IFF-1/9

Studentas

Prof. Vacius Jusas

Dėstytojas

TURINYS

1. Duomenų klasė.....	3
1.1. Darbo užduotis	3
1.2. Programos tekstas.....	3
1.3. Pradiniai duomenys ir rezultatai.....	14
1.4. Dėstytojo pastabos.....	22
2. Skaičiavimų klasė	23
2.1. Darbo užduotis	23
2.2. Programos tekstas.....	23
2.3. Pradiniai duomenys ir rezultatai.....	35
2.4. Dėstytojo pastabos.....	40
3. Konteineris.....	41
3.1. Darbo užduotis	41
3.2. Programos tekstas.....	41
3.3. Pradiniai duomenys ir rezultatai.....	59
3.4. Dėstytojo pastabos.....	64
4. Teksto analizė ir redagavimas	65
4.1. Darbo užduotis	65
4.2. Programos tekstas.....	65
4.3. Pradiniai duomenys ir rezultatai.....	70
4.4. Dėstytojo pastabos.....	71
5. Paveldėjimas	72
5.1. Darbo užduotis	72
5.2. Programos tekstas.....	73
5.3. Pradiniai duomenys ir rezultatai.....	91
5.4. Dėstytojo pastabos.....	97

1. Duomenų klasė

1.1. Darbo užduotis

U1-8. **Turistų informacijos centras.** Turite turistų informacijos centro pateiktus duomenis apie Lietuvoje veikiančius muziejus. Duomenų faile pateikta ši informacija: pavadinimas, miestas, tipas, 7 savaitės dienos (1 – darbo, 0 – nedarbo), bilieto kaina, požymis „turi gidą“.

- Suskaičiuokite, kiek muziejų Kaune turi gidus. Atspausdinkite ekrane suskaičiuotą kiekį ir pilną informaciją apie šiuos muziejus.
- Raskite, kokio tipo muziejus galima aplankyti Vilniuje trečiadieniais, ir atspausdinkite muziejų tipus ekrane.
- Sudarykite Kauno muziejų, kurie dirba ne mažiau, kaip 3 dienas per savaitę, sąrašą. Pilną informaciją apie šiuos muziejus įrašykite į failą „Kaunas.csv“.

1.2. Programos tekstas

Failas **Museum.cs**:

```
namespace Laboratorinis1
{
    /// <summary>
    /// Museum class that stores data about the museum object
    /// </summary>
    internal class Museum
    {
        public string Name { get; set; }
        public string City { get; set; }
        public string Type { get; set; }
        public int Mon { get; set; }
        public int Tues { get; set; }
        public int Wednes { get; set; }
        public int Thurs { get; set; }
        public int Fri { get; set; }
        public int Sat { get; set; }
        public int Sun { get; set; }
        public double Price { get; set; }
        public string Guided { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name"> name of musem </param>
        /// <param name="city"> city that is the museum in </param>
        /// <param name="type"> type of the museum </param>
        /// <param name="mon"> monday data </param>
        /// <param name="tues"> tuesday data </param>
        /// <param name="wednes"> wednesday data </param>
        /// <param name="thurs"> thursday data </param>
        /// <param name="fri"> friday data </param>
        /// <param name="sat"> saturday data </param>
        /// <param name="sun"> sunday data </param>
        /// <param name="price"> price of the ticket </param>
        /// <param name="guided"> does the museum have a guide </param>
        /// If the museum works on that day it is written as 1, if it doesnt work
        - as 0
        public Museum(string name, string city, string type,
```

```
int mon, int tues, int wednes, int thurs,  
int fri, int sat, int sun, double price, string guided)  
{  
    this.Name = name;  
    this.City = city;  
    this.Type = type;  
    this.Mon = mon;  
    this.Tues = tues;  
    this.Wednes = wednes;  
    this.Thurs = thurs;  
    this.Fri = fri;  
    this.Sat = sat;  
    this.Sun = sun;  
    this.Price = price;  
    this.Guided = guided;  
}  
}
```

Failas InOutUtils.cs:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace Laboratorinis1
{
    internal class InOutUtils
    {
        /// <summary>
        /// Read from file method, reads from file Museums.csv
        /// </summary>
        /// <param name="fileName"> the file name of the file that has the museum
data </param>
        /// <returns></returns>
        public static List<Museum> ReadMuseums(string fileName)
        {
            List<Museum> Museums = new List<Museum>();
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);
            foreach (string line in Lines)
            {
                string[] Bits = line.Split(';'); // splits read line at ; and
assigns to string array for separation

                string name = Bits[0];
                string city = Bits[1];
                string type = Bits[2];
                int mon = int.Parse(Bits[3]);
                int tues = int.Parse(Bits[4]);
                int wednes = int.Parse(Bits[5]);
                int thurs = int.Parse(Bits[6]);
                int fri = int.Parse(Bits[7]);
                int sat = int.Parse(Bits[8]);
                int sun = int.Parse(Bits[9]);
                double price = double.Parse(Bits[10]);
                string guided = Bits[11];

                Museum museum = new Museum(name, city, type, mon, tues,
creation wednes, thurs, fri, sat, sun, price, guided); // new museum object

                Museums.Add(museum); // adds a new museum object to list Museums
            }
            return Museums; // returns museum list
        }

        /// <summary>
        /// Method that prints museum list to file
        /// </summary>
        /// <param name="Museums"> List of museums</param>
        public static void PrintMuseums(List<Museum> Museums)
        {
            Console.WriteLine(new string('-', 142));
            Console.WriteLine("| {0,-30} | {1,-10} | {2,-10} | {3,-6} | {4,-6} |
{5,-7} |" +
                " {6,-7}| {7,-5} | {8,-4} | {9,-6} | {10,-5}| {11,-11} |",
                "Pavadinimas", "Miestas", "Tipas", "Pirmad", "Antrad", "Treciad",
                "Ketvirt", "Penkt",
                "Sest", "Sekmad", "Kaina", "Turi gida");
            Console.WriteLine(new string('-', 142));
            foreach (Museum museum in Museums)
```

```

        {
            Console.WriteLine("| {0,-30} | {1,-10} | {2,-10} | {3,6} | {4,6} | {5,7} |" +
                " {6,7}| {7,5} | {8,4} | {9,6} | {10,5}| {11,-11} |",
                museum.Name, museum.City, museum.Type, museum.Mon, museum.Tues,
                museum.Wednes, museum.Thurs, museum.Fri,
                museum.Sat, museum.Sun, museum.Price, museum.Guided);
        }
        Console.WriteLine(new string('-', 142));
    }

    /// <summary>
    /// Prints out explanation on whether museum working meaning. Extra
    function, not necessary
    /// </summary>
    public static void PrintHelp()
    {
        Console.WriteLine("");
        Console.WriteLine("Muziejų darbo paaiškinimas:");
        Console.WriteLine("1 - dirba, 0 - nedirba");
        Console.WriteLine("");
    }

    /// <summary>
    /// Method that outputs a count of filtered museums and museum information
    to console
    /// </summary>
    /// <param name="Museums"> List of museums</param>
    /// <param name="countedMuseums"> Ammount of museums that are within
    parameters </param>
    public static void PrintFilteredAndCounted(List<Museum> Museums, int
        countedMuseums)
    {
        string dashes = new string('-', 142);

        Console.WriteLine("Visas muziejų skaičius, kurie atitinka filtrą:
        {0}", countedMuseums);

        Console.WriteLine(dashes);
        Console.WriteLine("| {0,-30} | {1,-10} | {2,-10} | {3,-6} | {4,-6} |
        {5,-7} |" +
            " {6,-7}| {7,-5} | {8,-4} | {9,-6} | {10,-5}| {11,-11} |",
            "Pavadinimas", "Miestas", "Tipas", "Pirmad", "Antrad", "Treciad",
            "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gidą");

        Console.WriteLine(dashes);

        foreach (Museum museum in Museums)
        {
            Console.WriteLine("| {0,-30} | {1,-10} | {2,-10} | {3,6} | {4,6} |
            {5,7} |" +
                " {6,7}| {7,5} | {8,4} | {9,6} | {10,5}| {11,-11} |",
                museum.Name, museum.City, museum.Type, museum.Mon, museum.Tues,
                museum.Wednes, museum.Thurs, museum.Fri,
                museum.Sat, museum.Sun, museum.Price, museum.Guided);
        }
        Console.WriteLine(dashes);
    }

    /// <summary>
    /// A method that prints the starting data to a .txt file
    /// </summary>
    /// <param name="fileName"> FileName to where to print the starting
    data</param>

```

```

    /// <param name="Museums"> A list of museums </param>
    public static void PrintStartingResources(string fileName, List<Museum>
Museums)
    {
        string dashes = new string('-', 142); // a string for table borders

        string[] lines = new string[Museums.Count + 5];
        lines[0] = String.Format("Pradiniai duomenys:");
        lines[1] = String.Format(dashes); // top border of table
        lines[2] = String.Format("| {0,-30} | {1,-10} | {2,-10} | {3,-6} |
{4,-6} | {5,-7} |" +
            " {6,-7}| {7,-5} | {8,-4} | {9,-6} | {10,-5}| {11,-11} |",
            "Pavadinimas", "Miestas", "Tipas", "Pirmadi", "Antradi", "Treciadi",
            "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida"); // information
        explanations

        lines[3] = String.Format(dashes); // lower border of the table
        for (int i = 0; i < Museums.Count; i++)
        {
            lines[i + 4] = String.Format("| {0,-30} | {1,-10} | {2,-10} |
{3,6} | {4,6} | {5,7} |" +
                " {6,7}| {7,5} | {8,4} | {9,6} | {10,5}| {11,-11} |",
                Museums[i].Name, Museums[i].City, Museums[i].Type, Museums[i].Mon,
                Museums[i].Tues, Museums[i].Wednes, Museums[i].Thurs, Museums[i].Fri,
                Museums[i].Sat, Museums[i].Sun, Museums[i].Price,
                Museums[i].Guided); // assings an element of list to string array
        }
        lines[Museums.Count + 4] = String.Format(dashes); // bottom border of
        table

        File.WriteAllLines(fileName, lines, Encoding.UTF8); // prints string
        array to.txt file
    }

    /// <summary>
    /// A method that prints out the types of museums that were found working
    in that day
    /// </summary>
    /// <param name="Types"> list of types</param>
    public static void PrintTypes(List<string> Types)
    {
        if (Types != null) // checks if list is not empty
        {
            Console.WriteLine("Šiuos muziejų tipus galite apžvelgti savo
pasirinktame mieste, pasirinkta diena: ");
            foreach (string type in Types) // takes one part of the list and
            outputs to console
            {
                Console.WriteLine(type);
            }
        }
        else
        {
            Console.WriteLine("Jūsų pasirinkta diena, muziejai nedirbo");
            Console.WriteLine("");
        }
    }

    /// <summary>
    /// Method that prints task information to a .csv file
    /// </summary>
    /// <param name="fileName"> file name</param>
    /// <param name="Museums"> list of museums that work</param>
    /// <param name="workdays"> the ammount of days the museums work</param>
    public static void PrintWorkingMuseums(string fileName, List<Museum>

```

```

Museums, int workdays)
{
    Console.WriteLine("Pagal parametrus atrinktų muziejų informacija yra
faile {0}", fileName);
    string[] lines = new string[Museums.Count + 5];
    lines[0] = String.Format("Muziejų, kurie dirba bent {0} dienas/u
pasirinktame mieste informacija: ", workdays);

    lines[1] = String.Format("{0},{1},{2},{3},{4},{5}," +
        "{6},{7},{8},{9},{10},{11}",
        "Pavadinimas", "Miestas", "Tipas", "Pirmad", "Antrad", "Treciad",
        "Ketvirt", "Penkt", "Sest", "Sekmad", "Kaina", "Turi gida"); //
information explanations
    for (int i = 0; i < Museums.Count; i++)
    {
        lines[i + 2] = String.Format("{0},{1},{2},{3},{4},{5}," +
            "{6},{7},{8},{9},{10},{11}",
            Museums[i].Name, Museums[i].City, Museums[i].Type,
Museums[i].Mon,
            Museums[i].Tues, Museums[i].Wednes, Museums[i].Thurs,
Museums[i].Fri,
            Museums[i].Sat, Museums[i].Sun, Museums[i].Price,
            Museums[i].Guided); // assigns an element of list to
string array
    }
    File.WriteAllLines(fileName, lines, Encoding.UTF8); // prints string
array to .csv file
}
}
}

```


Failas TaskUtils.cs:

```
using System;
using System.Collections.Generic;

namespace Laboratorinis1
{
    internal class TaskUtils
    {
        /// <summary>
        /// List filtering with user selected filters
        /// </summary>
        /// <param name="Museums"> List of museums</param>
        /// <param name="isGuided"> user defined parameter for having a guide
        </param>
        /// <param name="selectedCity"> user defined parameter for a specific city
        </param>
        /// <returns> returns a List of Museums that are within user selected
        parameters </returns>
        public static List<Museum> FilterCityGuide(List<Museum> Museums, string
isGuided, string selectedCity)
        {
            string guider;
            if (isGuided.ToLower() == "taip")
            {
                guider = "Turi gida";
            }
            else
            {
                guider = "Neturi gido";
            }
            List<Museum> Filtered = new List<Museum>();
            foreach (Museum museum in Museums)
            {
                if (museum.City.Equals(selectedCity) &&
                    museum.Guided.Equals(guider))
                {
                    Filtered.Add(museum);
                }
            }
            return Filtered;
        }

        /// <summary>
        /// Counts the ammount of museums that are within selected parameters
        /// </summary>
        /// <param name="Museums"> List of museums</param>
        /// <param name="selectedCity"> User-selected City parameter </param>
        /// <returns> a number of museums in the selected city </returns>
        public static int CountSelectedCity(List<Museum> Museums, string
selectedCity)
        {
            int count = 0;

            foreach (Museum museum in Museums)
            {
                if (museum.City.Equals(selectedCity))
                {
                    count++;
                }
            }
            return count;
        }
    }
}
```

```

/// <summary>
/// Filters museums by user selected city
/// </summary>
/// <param name="Museums"></param>
/// <param name="selectedCity"></param>
/// <returns> A filtered list of museums </returns>
public static List<Museum> FilterCity(List<Museum> Museums, string
selectedCity)
{
    List<Museum> Filtered = new List<Museum>();
    foreach (Museum museum in Museums)
    {
        if (museum.City.Equals(selectedCity))
        {
            Filtered.Add(museum);
        }
    }
    return Filtered;
}

/// <summary>
/// Finds different types of museums that you can visit on selected day
/// </summary>
/// <param name="Museums"> List of Museums </param>
/// <param name="selectedDay"> Selected day </param>
/// <returns> Returns a string list of the museum types</returns>
public static List<string> FindTypes(List<Museum> Museums, string
selectedDay)
{
    List<string> Types = new List<string>(); // string list that contains
types of museums

    switch (selectedDay)
    {
        case "Pirmadienī":
            foreach (Museum museum in Museums)
            {
                if (museum.Mon == 1)
                {
                    if (!Types.Contains(museum.Type))
                    {
                        Types.Add(museum.Type);
                    }
                }
            }
            break;
        case "Antradienī":
            foreach (Museum museum in Museums)
            {
                if (museum.Tues == 1)
                {
                    if (!Types.Contains(museum.Type))
                    {
                        Types.Add(museum.Type);
                    }
                }
            }
            break;
        case "Trečdienī":
            foreach (Museum museum in Museums)
            {
                if (museum.Wednes == 1)
                {
                    if (!Types.Contains(museum.Type))
                    {

```

```

        Types.Add(museum.Type);
    }
}
break;
case "Ketvirtadienį":
    foreach (Museum museum in Museums)
    {
        if (museum.Thurs == 1)
        {
            if (!Types.Contains(museum.Type))
            {
                Types.Add(museum.Type);
            }
        }
    }
    break;
case "Penktadienį":
    foreach (Museum museum in Museums)
    {
        if (museum.Fri == 1)
        {
            if (!Types.Contains(museum.Type))
            {
                Types.Add(museum.Type);
            }
        }
    }
    break;
case "Šeštadienį":
    foreach (Museum museum in Museums)
    {
        if (museum.Sat == 1)
        {
            if (!Types.Contains(museum.Type))
            {
                Types.Add(museum.Type);
            }
        }
    }
    break;
case "Sekmadienį":
    foreach (Museum museum in Museums)
    {
        if (museum.Sun == 1)
        {
            if (!Types.Contains(museum.Type))
            {
                Types.Add(museum.Type);
            }
        }
    }
    break;
default:
    Console.WriteLine("Neteisingai įvesta diena arba tokios
dienos nėra!");
    break;
}
return Types;
}

/// <summary>
/// Creates a list of museums that work a user defined ammount of days in
a week.
/// </summary>

```

```

        /// <param name="Museums"> List of museums </param>
        /// <param name="workdays"> Ammount of days the user wants the museum to
work </param>
        /// <returns> a List of museums that work more than the user defined
ammount of days </returns>
        public static List<Museum> WorkingDays(List<Museum> Museums, int workdays)
        {
            List<Museum> Working = new List<Museum>();
            int works;

            foreach (Museum museum in Museums)
            {
                works = museum.Mon + museum.Tues + museum.Wednes + museum.Thurs +
                    museum.Fri + museum.Sat + museum.Sun;

                if (works >= workdays)
                {
                    Working.Add(museum);
                }
                works = 0;
            }
            return Working;
        }
    }
}

```

Failas Program.cs:

```

///
/// Martynas Kuliešius IFF-1/9 L1_U8
///

using System;
using System.Collections.Generic;

namespace Laboratorinis1
{
    internal class Program
    {
        /// <summary>
        /// Main function of program
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            //Test 1
            List<Museum> allMuseums = InOutUtils.ReadMuseums(@"Data1.csv"); //
reads information from .csv file and adds to list

            //Test 2
            //List<Museum> allMuseums = InOutUtils.ReadMuseums(@"Data2.csv"); //
reads information from .csv file and adds to list

            Console.WriteLine("Muziejų informacija"); // information of the table
            InOutUtils.PrintMuseums(allMuseums); // prints to console table of
information
            InOutUtils.PrintHelp(); // prints to console a helper for museum work

            InOutUtils.PrintStartingResources("Museums.txt", allMuseums); //
prints starting information to Museums.txt file.

            // First objective of individual task

```

```

        Console.WriteLine("Pasirinkite miesta, kurio muziejuose norite
apsilankyti: ");
        string selectedCity = Console.ReadLine();
        Console.WriteLine("Ar reikia gido? ");
        string sGuided = Console.ReadLine();

        Console.WriteLine("");

        List<Museum> FilteredByCityGuide =
TaskUtils.FilterCityGuide(allMuseums, sGuided, selectedCity);
        int countedMuseums = TaskUtils.CountSelectedCity(FilteredByCityGuide,
selectedCity);
        InOutUtils.PrintFilteredAndCounted(FilteredByCityGuide,
countedMuseums);
        Console.WriteLine("");

        // Second objective of individual task

        Console.WriteLine("Pasirinkite miesta, kurio muziejuose norite
apsilankyti: ");
        selectedCity = Console.ReadLine();
        List<Museum> SelectedMuseums = TaskUtils.FilterCity(allMuseums,
selectedCity);
        Console.WriteLine("Kelintadienį norite apsilankyti?");
        string selectedDay = Console.ReadLine();
        Console.WriteLine("");
        List<string> Types = TaskUtils.FindTypes(SelectedMuseums,
selectedDay);
        InOutUtils.PrintTypes(Types);
        Console.WriteLine("");

        // Third objective of individual task

        Console.WriteLine("Pasirinkite miesta, kurio muziejus norite
matyti.");
        selectedCity = Console.ReadLine();
        List<Museum> SelectedMuseum = TaskUtils.FilterCity(allMuseums,
selectedCity);
        Console.WriteLine("Bent kiek dienų turi dirbti šie muziejai?");
        int workdays = int.Parse(Console.ReadLine());
        Console.WriteLine("");
        List<Museum> WorkingMuseums = TaskUtils.WorkingDays(SelectedMuseum,
workdays);
        InOutUtils.PrintWorkingMuseums(selectedCity + ".csv", WorkingMuseums,
workdays);

        Console.ReadKey();
    }
}

```

1.3. Pradiniai duomenys ir rezultatai

Testas Nr. 1:

7 muziejai Kaune ir Vilniuje. Skirtingi muziejų tipai ir skirtingos darbo dienos. Skiriasi gido turėjimo žymeklis.

Pradiniai duomenys iš Data1.csv failo

```
Kauno Karo Muziejus;Kaunas;Muziejus;0;0;1;0;0;1;0;3.3;Turi gida;  
Vilniaus Karo Muziejus;Vilnius;Muziejus;1;1;0;0;1;1;0;3.2;Neturi gido;  
Kauno Paveikslų galerija;Kaunas;Galerija;0;1;1;0;0;0;1;3.4;Neturi gido;  
Kauno Sakralinis Muziejus;Kaunas;Muziejus;1;1;1;0;0;1;1;5.09;Turi gida;  
Kauno Miesto Muziejus;Kaunas;Muziejus;1;1;0;0;0;1;1;1.7;Turi gida;  
Vilniaus Muziejus;Vilnius;Muziejus;0;0;1;0;0;1;0;8.62;Turi gida;  
Vilniaus MO muziejus;Vilnius;Muziejus;0;1;1;1;0;0;1;2.22;Neturi gido
```

Pradiniai duomenys, išvesti į Museums.txt failą:

***Įkėlimas į Word sugadina išdėstymą ***

Pradiniai duomenys:

```
-----  
-----  
| Pavadinimas          | Miestas   | Tipas      | Pirmad | Antrad |  
Treciad | Ketvirt | Penkt  | Sest  | Sekmad | Kaina| Turi gida  |  
-----  
-----  
| Kauno Karo Muziejus   | Kaunas    | Muziejus   | 0 | 0 |  
1 | 0 | 0 | 1 | 0 | 3.3| Turi gida  |  
| Vilniaus Karo Muziejus | Vilnius    | Muziejus   | 1 | 1 |  
0 | 0 | 1 | 1 | 0 | 3.2| Neturi gido |  
| Kauno Paveikslų galerija | Kaunas    | Galerija   | 0 | 1 |  
1 | 0 | 0 | 0 | 1 | 3.4| Neturi gido |  
| Kauno Sakralinis Muziejus | Kaunas    | Muziejus   | 1 | 1 |  
1 | 0 | 0 | 1 | 1 | 5.09| Turi gida  |  
| Kauno Miesto Muziejus   | Kaunas    | Muziejus   | 1 | 1 |  
0 | 0 | 0 | 1 | 1 | 1.7| Turi gida  |  
| Vilniaus Muziejus      | Vilnius    | Muziejus   | 0 | 0 |  
1 | 0 | 0 | 1 | 0 | 8.62| Turi gida  |  
| Vilniaus MO muziejus    | Vilnius    | Muziejus   | 0 | 1 |  
1 | 1 | 0 | 0 | 1 | 2.22| Neturi gido |  
-----  
-----
```

Pradiniai duomenys:

Pavadinimas	Miestas	Tipas	Pirmad	Antrad	Treciad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą
Kauno Karo Muziejus	Kaunas	Muziejus	0	0	1	0	0	1	0	3.3	Turi gidą
Vilniaus Karo Muziejus	Vilnius	Muziejus	1	1	0	0	1	1	0	3.2	Neturi gido
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	1	0	0	0	1	3.4	Neturi gido
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	0	1	1	5.09	Turi gidą
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	0	1	1	1.7	Turi gidą
Vilniaus Muziejus	Vilnius	Muziejus	0	0	1	0	0	1	0	8.62	Turi gidą
Vilniaus MO muziejus	Vilnius	Muziejus	0	1	1	1	0	0	1	2.22	Neturi gido

Rezultatai konsolėje:

Savarankiško darbo užduotis: U1-8

Muziejų informacija

Pavadinimas	Miestas	Tipas	Pirmad	Antrad	Treciad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą
Kauno Karo Muziejus	Kaunas	Muziejus	0	0	1	0	0	1	0	3.3	Turi gidą
Vilniaus Karo Muziejus	Vilnius	Muziejus	1	1	0	0	1	1	0	3.2	Neturi gido
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	1	0	0	0	1	3.4	Neturi gido
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	0	1	1	5.09	Turi gidą
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	0	1	1	1.7	Turi gidą
Vilniaus Muziejus	Vilnius	Muziejus	0	0	1	0	0	1	0	8.62	Turi gidą
Vilniaus MO muziejus	Vilnius	Muziejus	0	1	1	1	0	0	1	2.22	Neturi gido

Muziejų darbo paaiškinimas:

1 - dirba, 0 - nedirba

Pasirinkite miestą, kurio muziejuose norite apsilankyti:

Kaunas

Ar reikia gido?

Taip

Visas muziejų skaičius, kurie atitinka filtrą: 3

Pavadinimas	Miestas	Tipas	Pirmad	Antrad	Trečiad	Ketvirt	Penkt	Sest	
Sekmad	Kaina	Turi gidą							

Kauno Karo Muziejus	Kaunas	Muziejus	0	0	1	0	0	1	0
Turi gidą									3.3
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	0	1	1
5.09	Turi gidą								
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	0	1	1
Turi gidą									1.7

Pasirinkite miestą, kurio muziejuose norite apsilankyti:									
Vilnius									
Kelintadienį norite apsilankyti?									
Trečiadienį									
Šiuos muziejų tipus galite aplankyti savo pasirinktame mieste, pasirinktą dieną:									
Muziejus									
Pasirinkite miestą, kurio muziejus norite matyti.									
Kaunas									
Bent kiek dienų turi dirbti šie muziejai?									
3									
Pagal parametrus atrinktų muziejų informacija yra faile Kaunas.csv									
Press any key to continue . . .									

Konsolės vaizdas:

```
Savarankiško darbo užduotis: U1-8

Muziejų informacija
-----
| Pavadinimas | Miestas | Tipas | Pirmad | Antradi | Trečiadi | Ketvirt | Penkt | Šešt | Sekmadi | Kaina | Turi gidą |
-----
| Kauno Karo Muziejus | Kaunas | Muziejus | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3.3 | Turi gidą |
| Vilniaus Karo Muziejus | Vilnius | Muziejus | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 3.2 | Neturi gido |
| Kauno Paveikslų galerija | Kaunas | Galerija | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 3.4 | Neturi gido |
| Kauno Sakralinis Muziejus | Kaunas | Muziejus | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 5.09 | Turi gidą |
| Kauno Miesto Muziejus | Kaunas | Muziejus | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1.7 | Turi gidą |
| Vilniaus Muziejus | Vilnius | Muziejus | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 8.62 | Turi gidą |
| Vilniaus MO muziejus | Vilnius | Muziejus | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 2.22 | Neturi gido |
-----

Muziejų darbo paaiškinimas:
1 - dirba, 0 - nedirba

Pasirinkite miestą, kurio muziejuose norite apsilankyti:
Kaunas
Ar reikia gido?
Taip

Visas muziejų skaičius, kurie atitinka filtrą: 3
-----
| Pavadinimas | Miestas | Tipas | Pirmad | Antradi | Trečiadi | Ketvirt | Penkt | Šešt | Sekmadi | Kaina | Turi gidą |
-----
| Kauno Karo Muziejus | Kaunas | Muziejus | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3.3 | Turi gidą |
| Kauno Sakralinis Muziejus | Kaunas | Muziejus | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 5.09 | Turi gidą |
| Kauno Miesto Muziejus | Kaunas | Muziejus | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1.7 | Turi gidą |
-----

Pasirinkite miestą, kurio muziejuose norite apsilankyti:
Vilnius
Kelintadienį norite apsilankyti?
Trečiadienį

Šiuos muziejų tipus galite aplankyti savo pasirinktame mieste, pasirinktą dieną:
Muziejus

Pasirinkite miestą, kurio muziejus norite matyti.
Kaunas
Bent kiek dienų turi dirbti šie muziejai?
3

Pagal parametrus atrinktų muziejų informacija yra faile Kaunas.csv
Press any key to continue . . .
```

Rezultatai Kaunas.csv faile:

Muziejų kurie dirba bent 3 dienas/ų pasirinktame mieste informacija:											
Pavadinimas	Miestas	Tipas	Pirmadi	Antradi	Trečiadi	Ketvirt	Penkt	Šešt	Sekmadi	Kaina	Turi gidą
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	1	0	0	0	1	3.4	Neturi gido
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	0	1	1	5.09	Turi gidą
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	0	1	1	1.7	Turi gidą

Testas Nr. 2

Pradiniai duomenys Data2.csv faile:

7 muziejai Kaune ir Vilniuje. Skirtingi muziejų tipai ir skirtingos darbo dienos. Skiriasi gido turėjimo žymeklis.

```
Vilniaus MO muziejus;Vilnius;Muziejus;1;1;0;1;0;0;1;2.22;Neturi gido;
Kauno Paveikslų galerija;Kaunas;Galerija;0;1;0;0;0;0;3.4;Neturi gido;
Vilniaus Muziejus;Vilnius;Muziejus;0;0;1;0;0;1;1;8.62;Turi gidą;
Vilniaus Karo Muziejus;Vilnius;Muziejus;1;1;0;0;1;1;0;3.2;Turi gidą;
Kauno Karo Muziejus;Kaunas;Muziejus;1;0;1;0;1;0;0;3.7;Neturi gido;
Kauno Miesto Muziejus;Kaunas;Muziejus;1;1;0;0;1;0;0;1.7;Turi gidą;
Kauno Sakralinis Muziejus;Kaunas;Muziejus;1;1;1;0;1;1;1;5.09;Neturi gido;
```

Pradiniai duomenys, išvesti į Museums.txt failą:

*Įkėlimas į Word sugadina išdėstymą *

Pradiniai duomenys:

```
-----
-----
| Pavadinimas      | Miestas | Tipas   | Pirmad | Antradiadi | Trečiadiadi | Ketvirtadiadi | Penktadiadi | Šeštadiadi | Sėkmadiadi | Turi gidą |
-----
-----
| Vilniaus MO muziejus      | Vilnius | Muziejus | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 2.22 | Neturi gido |
| Kauno Paveikslų galerija  | Kaunas  | Galerija | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3.4 | Neturi gido |
| Vilniaus Muziejus        | Vilnius | Muziejus | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 8.62 | Turi gidą |
| Vilniaus Karo Muziejus    | Vilnius | Muziejus | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 3.2 | Turi gidą |
| Kauno Karo Muziejus      | Kaunas  | Muziejus | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 3.7 | Neturi gido |
| Kauno Miesto Muziejus    | Kaunas  | Muziejus | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1.7 | Turi gidą |
| Kauno Sakralinis Muziejus | Kaunas  | Muziejus | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 5.09 | Neturi gido |
| Vilniaus Petro galerija   | Vilnius | Galerija | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6.9 | Turi gidą |
| Vilniaus Aleksėjaus parkas | Vilnius | Parkas   | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0.5 | Neturi gido |
-----
-----
```

Pradiniai duomenys:

Pavadinimas	Miestas	Tipas	Pirmad	Antrad	Treciad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą
Vilniaus MO muziejus	Vilnius	Muziejus	1	1	0	1	0	0	1	2.22	Neturi gidą
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	0	0	0	0	0	3.4	Neturi gidą
Vilniaus Muziejus	Vilnius	Muziejus	0	0	1	0	0	1	1	8.62	Turi gidą
Vilniaus Karo Muziejus	Vilnius	Muziejus	1	1	0	0	1	1	0	3.2	Turi gidą
Kauno Karo Muziejus	Kaunas	Muziejus	1	0	1	0	1	0	0	3.7	Neturi gidą
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	1	0	0	1.7	Turi gidą
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	1	1	1	5.09	Neturi gidą
Vilniaus Petro galerija	Vilnius	Galerija	1	1	1	1	1	1	1	6.9	Turi gidą
Vilniaus Aleksėjaus parkas	Vilnius	Parkas	0	0	1	0	0	0	0	0.5	Neturi gidą

Rezultatai konsolėje:

Savarankiško darbo užduotis: U1-8

Muziejų informacija

Pavadinimas	Miestas	Tipas	Pirmad	Antrad	Treciad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą
-------------	---------	-------	--------	--------	---------	---------	-------	------	--------	-------	-----------

Vilniaus MO muziejus	Vilnius	Muziejus	1	1	0	1	0	0	1	2.22	Neturi gidą
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	0	0	0	0	0	3.4	Neturi gidą
Vilniaus Muziejus	Vilnius	Muziejus	0	0	1	0	0	1	1	8.62	Turi gidą
Vilniaus Karo Muziejus	Vilnius	Muziejus	1	1	0	0	1	1	0	3.2	Turi gidą
Kauno Karo Muziejus	Kaunas	Muziejus	1	0	1	0	1	0	0	3.7	Neturi gidą
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	1	0	0	1.7	Turi gidą
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	1	1	1	5.09	Neturi gidą
Vilniaus Petro galerija	Vilnius	Galerija	1	1	1	1	1	1	1	6.9	Turi gidą
Vilniaus Aleksėjaus parkas	Vilnius	Parkas	0	0	1	0	0	0	0	0.5	Neturi gidą

Muziejų darbo paaiškinimas:

1 - dirba, 0 - nedirba

Pasirinkite miestą, kurio muziejuose norite apsilankyti:

Kaunas

Ar reikia gido?

Taip

Visas muziejų skaičius, kurie atitinka filtrą: 1

| Pavadinimas | Miestas | Tipas | Pirmad | Antrad | Treciad | Ketvirt | Penkt | Sest |
Sekmad | Kaina | Turi gidą |

| Kauno Miesto Muziejus | Kaunas | Muziejus | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1.7 |
Turi gidą |

Pasirinkite miestą, kurio muziejuose norite apsilankyti:

Vilnius

Kelintadienį norite apsilankyti?

Trečiadienį

Šiuos muziejų tipus galite aplankyti savo pasirinktame mieste, pasirinktą dieną:

Muziejus

Galerija

Parkas

Pasirinkite miestą, kurio muziejus norite matyti.

Kaunas

Bent kiek dienų turi dirbti šie muziejai?

3

Pagal parametrus atrinktų muziejų informacija yra faile Kaunas.csv

Press any key to continue . . .

Konsolės vaizdas:

Savarankiško darbo užduotis: U1-8

Muziejų informacija

Pavadinimas	Miestas	Tipas	Pirmad	Antradi	Trečiadi	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gido
Vilniaus MO muziejus	Vilnius	Muziejus	1	1	0	1	0	0	1	2.22	Neturi gido
Kauno Paveikslų galerija	Kaunas	Galerija	0	1	0	0	0	0	0	3.4	Neturi gido
Vilniaus Muziejus	Vilnius	Muziejus	0	0	1	0	0	1	1	8.62	Turi gido
Vilniaus Karo Muziejus	Vilnius	Muziejus	1	1	0	0	1	1	0	3.2	Turi gido
Kauno Karo Muziejus	Kaunas	Muziejus	1	0	1	0	1	0	0	3.7	Neturi gido
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	1	0	0	1.7	Turi gido
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	1	1	1	5.09	Neturi gido
Vilniaus Petro galerija	Vilnius	Galerija	1	1	1	1	1	1	1	6.9	Turi gido
Vilniaus Aleksėjaus parkas	Vilnius	Parkas	0	0	1	0	0	0	0	0.5	Neturi gido

Muziejų darbo paaiškinimas:

1 - dirba, 0 - nedirba

Pasirinkite miestą, kurio muziejuose norite apsilankyti:

Kaunas

Ar reikia gido?

Taip

Visas muziejų skaičius, kurie atitinka filtrą: 1

Pavadinimas	Miestas	Tipas	Pirmad	Antradi	Trečiadi	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gido
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	1	0	0	1.7	Turi gido

Pasirinkite miestą, kurio muziejuose norite apsilankyti:

Vilnius

Kelintadienį norite apsilankyti?

Trečiadienį

Šiuos muziejų tipus galite apžvelgti savo pasirinktame mieste, pasirinktą dieną:

Muziejus

Galerija

Parkas

Pasirinkite miestą, kurio muziejus norite matyti.

Kaunas

Bent kiek dienų turi dirbti šie muziejai?

3

Pagal parametrus atrinktų muziejų informacija yra faile Kaunas.csv

Press any key to continue . . .

Rezultatai Kaunas.csv faile:

Muziejų	kurie dirba bent 3 dienas/ų pasirinktame mieste informacija:										
Pavadinimas	Miestas	Tipas	Pirmadi	Antradi	Trečiadi	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gido
Kauno Karo Muziejus	Kaunas	Muziejus	1	0	1	0	1	0	0	3.7	Neturi gido
Kauno Miesto Muziejus	Kaunas	Muziejus	1	1	0	0	1	0	0	1.7	Turi gido
Kauno Sakralinis Muziejus	Kaunas	Muziejus	1	1	1	0	1	1	1	5.09	Neturi gido

1.4. Dėstytojo pastabos

InOutUtils.cs turi būti pirmiau negu TaskUtils.cs

2. Skaičiavimų klasė

2.1. Darbo užduotis

U2-8. Turistų informacijos centras. Turizmo informacijos centre perorganizuoti ir atskirai surašyti duomenys apie dvejuose miestuose veikiančius muziejus. Keičiasi duomenų formatas. Pirmoje eilutėje – miestas, antroje – atsakingo asmens vardas ir pavardė. Toliau informacija apie muziejus pateikta tokiu pačiu formatu kaip L1 užduotyje, tik nebėra miesto stulpelio.

- Raskite muziejų, kuris dirba daugiausia savaitės dienų. Atspausdinkite ekrane visus jo duomenis. Jei yra keli, spausdinkite visus.
- Suskaičiuokite, kuriame mieste yra daugiau muziejų, turinčių gidus, rezultatą atspausdinkite ekrane.
- Sudarykite kiekvieno miesto muziejų, kuriuos galima aplankyti nemokamai, sąrašą, į failus „Nemokami_miestas.csv“ įrašykite muziejaus tipą ir pavadinimą. Jei muziejus dirba tik šeštadieniais ir sekmadieniais, atitinkamoje eilutėje įrašykite „TIK SAVAITGALIAIS“.

2.2. Programos tekstas

Museum.cs failas:

```
using System;

namespace U2_8
{
    class Museum
    {
        public string Name { get; set; }
        public string Type { get; set; }
        public int Mon { get; set; }
        public int Tues { get; set; }
        public int Wednes { get; set; }
        public int Thurs { get; set; }
        public int Fri { get; set; }
        public int Sat { get; set; }
        public int Sun { get; set; }
        public double Price { get; set; }
        public string Guided { get; set; }
        public int WorkingDays { get; set; }
        public string WeekEnders { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name"> name of museum</param>
        /// <param name="type"> type of museum </param>
        /// <param name="mon"> monday </param>
        /// <param name="tues"> tuesday </param>
        /// <param name="wednes"> wednesday </param>
        /// <param name="thurs"> thursday </param>
        /// <param name="fri"> friday </param>
        /// <param name="sat"> saturday </param>
        /// <param name="sun"> sunday </param>
        /// <param name="price"> price of ticket</param>
        /// <param name="guided"> does museum have a guide </param>
    }
}
```

```

    /// <param name="WorkingDays"> how many days does museum work </param>
    /// <param name="WeekEnders"> string for checking if museum only works on
weekends </param>

    public Museum(string name, string type,
        int mon, int tues, int wednes, int thurs,
        int fri, int sat, int sun, double price, string guided)
    {
        this.Name = name;
        this.Type = type;
        this.Mon = mon;
        this.Tues = tues;
        this.Wednes = wednes;
        this.Thurs = thurs;
        this.Fri = fri;
        this.Sat = sat;
        this.Sun = sun;
        this.Price = price;
        this.Guided = guided;
    }

    /// <summary>
    /// Calculates how many days a week does a museum work
    /// </summary>
    public void CalculateWorkingDays()
    {
        int works = 0;
        works = Mon + Tues + Thurs + Wednes + Fri + Sat + Sun;

        WorkingDays = works;
    }

    /// <summary>
    /// Assigns value to weekend string if museum only works
    /// </summary>
    public void CalculateWeekenders()
    {
        string weekend;

        if ((Mon + Tues + Thurs + Wednes + Fri) == 0 & (Sat + Sun > 0))
        {
            weekend = "Tik Savaitgaliaais!";
        }
        else
        {
            weekend = "";
        }

        WeekEnders = weekend;
    }

    /// <summary>
    /// Overrides the method ToString() to return a string in a required
format
    /// </summary>
    /// <returns>A string of information in a required format</returns>
    public override string ToString()
    {
        string line;
        line = string.Format(String.Format("| {0,-30} | {1,-10} | {2,6} |
{3,6} | {4,7} |" +
            " {5,7}| {6,5} | {7,4} | {8,6} | {9,5}| {10,-11} |", Name, Type,
Mon, Tues,
            Wednes, Thurs, Fri, Sat, Sun, Price, Guided));
    }

```



```

        return line;
    }

    public static bool operator <(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays < museum2.WorkingDays;
    }
    public static bool operator >(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays > museum2.WorkingDays;
    }

    public static bool operator ==(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays == museum2.WorkingDays;
    }

    public static bool operator !=(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays != museum2.WorkingDays;
    }

    public static bool operator ==(Museum museum, int num)
    {
        return museum.Price == num;
    }

    public static bool operator !=(Museum museum, int num)
    {
        return museum.WorkingDays != num;
    }

    /// <summary>
    /// Overrides the method Equals() to compare
    /// </summary>
    public override bool Equals(object obj)
    {
        return this.Name == ((Museum)obj).Name;
    }

    /// <summary>
    /// Security function
    /// </summary>
    public override int GetHashCode()
    {
        return this.Name.GetHashCode();
    }
}
}

```

MuseumsRegister.cs failas:

```
using System;
using System.Collections.Generic;

namespace U2_8
{
    class MuseumsRegister
    {
        private List<Museum> Museums;
        public string Manager { get; set; } // pirmos eilutes saugomi duomenys
        public string City { get; set; } // antros eilutes saugomi duomenys

        /// <summary>
        /// creates new list
        /// </summary>
        public MuseumsRegister()
        {
            Museums = new List<Museum>();
        }

        /// <summary>
        /// Adds new objects to list
        /// </summary>
        /// <param name="Museums"></param>
        public MuseumsRegister(List<Museum> Museums)
        {
            Museums = new List<Museum>();

            foreach (Museum museum in Museums)
            {
                this.Museums.Add(museum);
            }
        }

        /// <summary>
        /// Add a museum to register
        /// </summary>
        /// <param name="museum"> Add museum object to register</param>
        public void Add(Museum museum)
        {
            Museums.Add(museum);
        }

        /// <summary>
        /// Finds number of museums
        /// </summary>
        /// <returns></returns>
        public int MuseumsCount()
        {
            return this.Museums.Count;
        }

        /// <summary>
        /// Returns a museum in the register by its index
        /// </summary>
        public Museum GetByIndex(int index)
        {
            return Museums[index];
        }
    }
}
```

```

/// <summary>
/// Finds museum that works the most
/// </summary>
/// <returns> index of museum that works the most days of the week </returns>
public int MostWorking()
{
    int ID=0;

    for (int i = 0; i < Museums.Count; i++)
    {
        if (Museums[i] > Museums[ID])
        {
            ID = i; // max day integer
        }
    }

    return ID;
}

/// <summary>
/// Bool to check if there's a duplicate in the register
/// </summary>
/// <param name="museum"></param>
/// <returns></returns>
public bool Contains(Museum museum)
{
    return Museums.Contains(museum);
}

/// <summary>
/// Calculates how many museums have guided tours in one city
/// </summary>
/// <returns> Returns ammount of museums that have a guide </returns>
public int CalculateGuides()
{
    int guideNum = 0;

    foreach (Museum museum in Museums)
    {
        if (museum.Guided.Equals("Turi gida"))
        {
            guideNum++;
        }
    }

    return guideNum;
}

/// <summary>
/// Creates list of museums that work the most
/// </summary>
/// <param name="MostWorkingMuseum"> Museum object</param>
/// <param name="Filtered"> a list for filtered museums to be stored in</param>
/// <param name="compared"> museum comparison register</param>
/// <returns></returns>
public List<Museum> ComparedByWorking(Museum MostWorkingMuseum, List<Museum>
Filtered, MuseumsRegister compared)
{
    for (int i = 0; i < compared.Museums.Count; i++)
    {
        if (MostWorkingMuseum == compared.Museums[i])
        {
            if (!Filtered.Contains(compared.Museums[i]))

```

```

        {
            Filtered.Add(compared.Museums[i]);
        }
    }
}

return Filtered;
}

/// <summary>
/// Makes a list of free to enter museums
/// </summary>
/// <param name="Freebies"> List of museums that are free to enter </param>
/// <param name="register"> the main storage register </param>
/// <returns></returns>
public List<Museum> FreeMuseums(List<Museum> Freebies, MuseumsRegister register)
{
    for (int i = 0; i < register.Museums.Count; i++)
    {
        if (register.Museums[i] == 0)
        {
            if (!Freebies.Contains(register.Museums[i]))
            {
                Freebies.Add(register.Museums[i]);
            }
        }
    }

    return Freebies;
}
}
}

```

InOutUtils.cs failas:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace U2_8
{
    class InOutUtils
    {
        /// <summary>
        /// Read function, reads from file Cityn.csv
        /// </summary>
        public static MuseumsRegister ReadMuseums(string fileName)
        {
            MuseumsRegister Museums = new MuseumsRegister();
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

            string line;
            Museums.City = Lines[0];
            Museums.Manager = Lines[1];

            for (int i = 2; i < Lines.Length; i++)
            {
                line = Lines[i];
                string[] Bits = line.Split(';'); // splits read line at ; and
                assigns to string array for separation
                string name = Bits[0];
                string type = Bits[1];
                int mon = int.Parse(Bits[2]);
                int tues = int.Parse(Bits[3]);
                int wednes = int.Parse(Bits[4]);
                int thurs = int.Parse(Bits[5]);
                int fri = int.Parse(Bits[6]);
                int sat = int.Parse(Bits[7]);
                int sun = int.Parse(Bits[8]);
                double price = double.Parse(Bits[9]);
                string guided = Bits[10];

                Museum museum = new Museum(name, type, mon, tues,
                wednes, thurs, fri, sat, sun, price, guided); // new museum
                creation

                museum.CalculateWorkingDays(); // calculates how many days of the
                week certain museum works
                museum.CalculateWeekenders(); // checks if the museum only works
                on weekends

                Museums.Add(museum); // adds new museum to museum register
            }
            return Museums; // returns museum register
        }

        /// <summary>
        /// Extra method to print out a header for table
        /// </summary>
        public static void TableHeader()
        {
            string dashes = new string('-', 169);

            Console.WriteLine(dashes);
            Console.WriteLine("{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6} |
{5,6} | {6,7} |" +
                " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
```

```

        "Atsakingas asmuo", "Pavadinimas", "Miestas", "Tipas", "Pirmad",
        "Antrad", "Treciad", "Ketvirt", "Penkt",
        "Sest", "Sekmad", "Kaina", "Turi gida");
        Console.WriteLine(dashes);
    }
    /// <summary>
    /// Fixed PrintMostWorkingMuseums method.
    /// </summary>
    /// <param name="City">list of the museums in the city</param>
    public static void PMWM(List<Museum> City, MuseumsRegister Register)
    {
        string dashes = new string('-', 169);

        foreach (Museum museum in City)
        {
            Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
                " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
Register.Manager, museum.Name, Register.City, museum.Type, museum.Mon,
museum.Tues,
                museum.Wednes, museum.Thurs, museum.Fri, museum.Sat, museum.Sun,
museum.Price, museum.Guided);
        }
        Console.WriteLine(dashes);
    }

    /*
    /// <summary>
    /// Prints list of museums to console || first task
    /// </summary>
    public static void PrintMostWorkingMuseums(Museum mostWorkingMuseum,
List<Museum> City1, List<Museum> City2, MuseumsRegister register1, MuseumsRegister
register2) // prints museums in a table in console
    {
        string dashes = new string('-', 169);
        City1 = register1.ComparedByWorking(mostWorkingMuseum, City1,
register1);

        Console.WriteLine(dashes);
        Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6} |
{5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
            "Atsakingas asmuo", "Pavadinimas", "Miestas", "Tipas", "Pirmad",
            "Antrad", "Treciad", "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida");
        Console.WriteLine(dashes);

        foreach (Museum museum in City1)
        {
            Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
                " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
register1.Manager, museum.Name, register1.City, museum.Type, museum.Mon,
museum.Tues,
                museum.Wednes, museum.Thurs, museum.Fri, museum.Sat, museum.Sun,
museum.Price, museum.Guided);
        }
        Console.WriteLine(dashes); Console.WriteLine(dashes);
        InOutUtils.PrintMostWorkingMuseums(mostWorkingMuseum, City2,
register2);

        Console.WriteLine(dashes);
    }

    /// <summary>

```

```

    /// Prints list of museums to console || first task second part
    /// </summary>
    public static void PrintMostWorkingMuseums(Museum mostWorkingMuseum,
List<Museum> City2, MuseumsRegister register2) // prints museums in a table in
console
    {
        City2 = register2.ComparedByWorking(mostWorkingMuseum, City2,
register2);

        foreach (Museum museum in City2)
        {
            Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
                " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
register2.Manager, museum.Name, register2.City, museum.Type, museum.Mon,
museum.Tues,
                museum.Wednes, museum.Thurs, museum.Fri, museum.Sat, museum.Sun,
museum.Price, museum.Guided);
        }
    }
    */

    /// <summary>
    /// Appends a register to an already existing non-empty file
    /// </summary>
    public static void AppendMuseumsToTxt(string fileName, MuseumsRegister
register)
    {
        string[] lines = new string[register.MuseumsCount() + 7];
        lines[0] = register.City;
        lines[1] = register.Manager;
        lines[2] = String.Format(new string('-', 128));
        lines[3] = String.Format("| {0,-30} | {1,-10} | {2,6} | {3,6} | {4,7}
|" +
                " {5,7}| {6,5} | {7,4} | {8,6} | {9,5}| {10,-11} |"
                , "Pavadinimas", "Tipas", "Pirmad", "Antrad", "Treciad", "Ketvirt",
"Penkt",
                "Sest", "Sekmad", "Kaina", "Turi gida");
        lines[4] = String.Format(new string('-', 128));
        for (int i = 0; i < register.MuseumsCount(); i++)
        {
            Museum museum = register.GetByIndex(i);
            lines[i + 6] = museum.ToString();
        }
        lines[register.MuseumsCount() + 6] = String.Format(new string('-',
128));

        File.AppendAllLines(fileName, lines, Encoding.UTF8);
    }

    /// <summary>
    /// Prints free to enter museum list to a specified .txt file
    /// </summary>
    /// <param name="txt"></param> .txt file name
    /// <param name="register"></param> free to enter museum list
    public static void PrintFreebies(string fileName, List<Museum>
Freebies, MuseumsRegister register) // prints free to enter museums to .txt file
    {
        Console.WriteLine("Muziejų, kurie yra nemokami sarašas yra faile {0}",
fileName);

        string[] lines = new string[Freebies.Count + 3];

        lines[0] = String.Format("Nemokami muziejai mieste
{0}", register.City); ;
    }

```

```

        lines[1] = String.Format("Tipas, Pavadinimas");

        for (int i = 0; i < Freebies.Count; i++)
        {
            lines[i + 2] = string.Join(", ",
                Freebies[i].Type, Freebies[i].Name, Freebies[i].WeekEnd);
        }

        File.WriteAllLines(fileName, lines, Encoding.UTF8);
    }
}

```

Program.cs failas:

```

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;

namespace U2_8
{
    class Program
    {
        static void Main(string[] args)
        {
            // Clearing old test files
            if (File.Exists("startingData.txt"))
            {
                File.Delete("startingData.txt");
            }
            if (File.Exists("Nemokami_Kaunas.txt"))
            {
                File.Delete("Nemokami_Kaunas.txt");
            }
            if (File.Exists("Nemokami_Vilnius.txt"))
            {
                File.Delete("Nemokami_Vilnius.txt");
            }

            Console.OutputEncoding = Encoding.UTF8;

            MuseumsRegister register1= InOutUtils.ReadMuseums(@"City1.csv"); //
            reads information from .csv file and adds to register
            InOutUtils.AppendMuseumsToTxt("startingData.txt", register1); //
            prints out starting data

            MuseumsRegister register2 = InOutUtils.ReadMuseums(@"City2.csv"); //
            reads information from .csv file and adds to register
            InOutUtils.AppendMuseumsToTxt("startingData.txt", register2); //
            prints out starting data

            Console.WriteLine("");

            // First task

            // These lists are going to be used later
            List<Museum> City1 = new List<Museum>();
            List<Museum> City2 = new List<Museum>();

```



```

        Museum MostWorkingCity1 =
register1.GetList()[register1.MostWorking()]; // pertvarkyta su GetList() funkcija
|| Finds the most working museum from first city
        Museum MostWorkingCity2 =
register2.GetList()[register2.MostWorking()]; // pertvarkyta su GetList() funkcija
|| Finds the most working museum from second city

        City1 = register1.ComparedByWorking(MostWorkingCity1, City1,
register1);
        City2 = register2.ComparedByWorking(MostWorkingCity2, City2,
register2);

        if (MostWorkingCity1 > MostWorkingCity2)
        {
            Console.WriteLine("Muziejų, kurie dirba daugiausiai per savaitę
buvo mieste {0}, jų sąrašas:", register1.City);

            InOutUtils.TableHeader();
            InOutUtils.PMWM(City1, register1);
            Console.WriteLine("");
        }
        else if (MostWorkingCity1 < MostWorkingCity2)
        {
            Console.WriteLine("Muziejų, kurie dirba daugiausiai per savaitę
buvo mieste {0}, jų sąrašas:", register2.City);
            InOutUtils.TableHeader();
            InOutUtils.PMWM(City2, register2);
            Console.WriteLine("");
        }
        else
        {
            Console.WriteLine("Abiejuose miestuose daugiausiai dirbančių
muziejų skaičius buvo vienodas, jų sąrašas:");
            InOutUtils.TableHeader();
            InOutUtils.PMWM(City1, register1);
            InOutUtils.PMWM(City2, register2);
            Console.WriteLine("");
        }

        // second task

        int guidedCity1 = register1.CalculateGuides(); // calculates the
ammount of museums that have guides
        int guidedCity2 = register2.CalculateGuides(); // calculates the
ammount of museums that have guides

        if (guidedCity1 > guidedCity2)
        {
            Console.WriteLine("Mieste {0} yra daugiau muziejų su gidais nei
mieste {1}. Muziejų su gidais kiekis: {2}", register1.City, register2.City,
guidedCity1);
            Console.WriteLine("");
        }
        else if (guidedCity1 < guidedCity2)
        {
            Console.WriteLine("Mieste {0} yra daugiau muziejų su gidais nei
mieste {1}. Muziejų su gidais kiekis: {2}", register2.City, register1.City,
guidedCity2);
            Console.WriteLine("");
        }
        else
        {
            Console.WriteLine("Abiejuose miestuose yra vienodas kiekis muziejų
su gidais. Muziejų su gidais skaičius: {0}", guidedCity1);

```

```

        Console.WriteLine("");
    }

    // third task

    City1.Clear();
    City2.Clear();

    City1 = register1.FreeMuseums(City1, register1); // Adds museums that
are free to enter to list
    City2 = register2.FreeMuseums(City2, register2); // Adds museums that
are free to enter to list

    string FreebieFileName; // Printing file for free to enter museums

    if (City1.Count > 0)
    {
        FreebieFileName = "Nemokami_" + register1.City + ".txt";
        InOutUtils.PrintFreebies(FreebieFileName, City1, register1);
    }
    else
    {
        Console.WriteLine("Mieste {0} nėra muziejų, kuriuos galima
aplankyti nemokamai", register1.City);
    }

    if (City2.Count > 0)
    {
        FreebieFileName = "Nemokami_" + register2.City + ".txt";
        InOutUtils.PrintFreebies(FreebieFileName, City2, register2);
        Console.WriteLine("");
    }
    else
    {
        Console.WriteLine("Mieste {0} nėra muziejų, kuriuos galima
aplankyti nemokamai", register2.City);
        Console.WriteLine("");
    }

    Console.ReadKey();
}
}
}

```

2.3. Pradiniai duomenys ir rezultatai

Testas nr. 1:

Vilniuje yra daugiau muziejų su gida. Tiek Kaune, tiek Vilniuje yra muziejai, kurie dirba kasdien. Kaune yra muziejų, kurie yra nemokami, Vilniuje nėra.

City1.csv failas:

```
Kaunas
Petras Petrauskas
Kauno Karo Muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gidą;
Kauno Meno galerija;Galerija;0;0;0;0;0;1;1;0;Neturi gido;
Kauno Mokslo muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gidą;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido;
Kauno Zoologijos sodas;Parkas;1;1;1;1;1;1;1;5;Neturi gido;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;1;1;3.3;Turi gidą;
```

City2.csv failas:

```
Vilnius
Antanas Antanas
Vilniaus Meno galerija;Galerija;1;0;1;0;1;0;1;3.3;Neturi gido;
Vilniaus Muzikos Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gidą;
Vilniaus Karo Muziejus;Muziejus;0;0;1;0;1;0;1;3.3;Turi gidą;
Vilniaus Antano Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gidą;
Vilniaus Memorialinis parkas;Parkas;1;0;0;0;1;0;1;3.3;Neturi gido;
Vilniaus Uno Muziejus;Muziejus;1;1;1;1;1;1;1;3.3;Turi gidą;
Vilniaus Petro Muziejus;Muziejus;1;0;1;0;1;0;0;3.3;Neturi gido;
Vilniaus Gedimino Muziejus;Muziejus;0;0;0;0;1;0;1;3.3;Turi gidą;
```

Spausdinti pradiniai duomenys data.txt faile:

```
Kaunas
Petras Petrauskas
-----
| Pavadinimas          | Tipas      | Pirmadi | Antradi | Trečiadi |
Ketvirt | Penkt | Šešt | Sekmadi | Kaina | Turi gidą |
-----
| Kauno Karo Muziejus   | Muziejus   | 1 | 0 | 1 |
0 | 1 | 0 | 1 | 0 | Turi gidą |
| Kauno Meno galerija   | Galerija   | 0 | 0 | 0 |
0 | 0 | 1 | 1 | 0 | Neturi gido |
| Kauno Mokslo muziejus | Muziejus   | 1 | 0 | 1 |
0 | 1 | 0 | 1 | 0 | Turi gidą |
| Kauno Karo Muziejus   | Muziejus   | 0 | 0 | 0 |
0 | 0 | 0 | 1 | 3.3 | Neturi gido |
```

Kauno Zoologijos sodas	Parkas	1	1	1
1 1 1 1 5	Neturi gido			
Kauno Karo Muziejus	Muziejus	0	0	0
0 0 1 1 3.3	Turi gida			

Vilnius				
Antanas Antanaskas				

Pavadinimas	Tipas	Pirmad	Antrad	Treciad
Ketvirt Penkt Sest Sekmad Kaina	Turi gida			

Vilniaus Meno galerija	Galerija	1	0	1
0 1 0 1 3.3	Neturi gido			
Vilniaus Muzikos Muziejus	Muziejus	1	0	1
0 1 0 1 3.3	Turi gida			
Vilniaus Karo Muziejus	Muziejus	0	0	1
0 1 0 1 3.3	Turi gida			
Vilniaus Antano Muziejus	Muziejus	1	0	1
0 1 0 1 3.3	Turi gida			
Vilniaus Memorialinis parkas	Parkas	1	0	0
0 1 0 1 3.3	Neturi gido			
Vilniaus Uno Muziejus	Muziejus	1	1	1
1 1 1 1 3.3	Turi gida			
Vilniaus Petro Muziejus	Muziejus	1	0	1
0 1 0 0 3.3	Neturi gido			
Vilniaus Gedimino Muziejus	Muziejus	0	0	0
0 1 0 1 3.3	Turi gida			

Ekrane spausdinami rezultatai:

```
Savarankiško darbo užduotis: U2-8
Pradiniai duomenys yra atspausdinti startingData.txt faile
Muziejų, kurie dirba daugiausiai per savaitę sąrašas:
-----
|Atsakingas asmuo | Pavadinimas | Miestas | Tipas | Pirmad | Antradi | Trečiadi | Ketvirt | Penkt | Šeštadi | Sekmadi | Kaina | Turi gidą |
-----
|Petras Petrauskas | Kauno Zoologijos sodas | Kaunas | Parkas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | Neturi gido |
|Antanas Antanas | Vilniaus Uno Muziejus | Vilnius | Muziejus | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3.3 | Turi gidą |
-----
Mieste Vilnius yra daugiau muziejų su gidais nei mieste Kaunas. Muziejų su gidais kiekis: 5
Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Kaunas.txt
Mieste Vilnius nėra muziejų, kuriuos galima aplankyti nemokamai
```

Į tekstinius failus spausdinami rezultatai:

Nemokami_Kaunas.txt:

Nemokami muziejai mieste Kaunas
Tipas, Pavadinimas
Muziejus, Kauno Karo Muziejus,
Galerija, Kauno Meno galerija, Tik Savaitgaliais!
Muziejus, Kauno Mokslo muziejus,

Testas nr. 2:

Kaune yra daugiau muziejų su gidais. Tik Vilniuje yra muziejai, kurie dirba kasdien. Tiek Kaune, tiek Vilniuje yra muziejų, kurie yra nemokami.

City1.csv failas:

```
Kaunas
Petras Petrauskas
Kauno Karo Muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gidą;
Kauno Meno galerija;Galerija;0;0;0;0;0;1;1;0;Turi gidą;
Kauno Mokslo muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gidą;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido;
Kauno Zoologijos sodas;Parkas;1;1;1;1;1;1;1;5;Turi gidą;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;1;1;0;Turi gidą;
Kauno Šventas sodas;Parkas;1;1;1;1;1;1;1;0;Turi gidą;
Kauno Miesto Muziejus;Muziejus;0;0;0;1;0;1;1;2.5;Turi gidą;
```

City2.csv failas:

```
Vilnius
Antanas Antanas
Vilniaus Meno galerija;Galerija;1;0;1;0;1;0;1;3.3;Neturi gido;
Vilniaus Muzikos Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gidą;
Vilniaus Karo Muziejus;Muziejus;0;0;1;0;1;0;1;0;Turi gidą;
Vilniaus Antano Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Neturi gido;
Vilniaus Memorialinis parkas;Parkas;1;0;0;0;1;0;1;0;Neturi gido;
Vilniaus Uno Muziejus;Muziejus;1;1;1;1;1;1;1;3.3;Neturi gido;
Vilniaus Petro Muziejus;Muziejus;1;0;1;0;1;0;0;3.3;Neturi gido;
Vilniaus Gedimino Muziejus;Muziejus;0;0;0;0;1;0;1;0;Neturi gido;
Vilniaus MO Muziejus;Muziejus;1;1;1;1;1;1;1;6.9;Turi gidą;
```

Vilniaus Žirgų Muziejus;Muziejus;1;1;1;1;1;1;1;6.9;Neturi gido;
Vilniaus Miesto Muziejus;Muziejus;0;0;0;0;0;1;1;0;Neturi gido;

Spausdinti pradiniai duomenys data.txt faile:

Kaunas

Petras Petrauskas

```
-----
| Pavadinimas          | Tipas          | Pirmad | Antrad | Treciad |
Ketvirt| Penkt | Sest  | Sekmad | Kaina| Turi gida  |
-----
| Kauno Karo Muziejus   | Muziejus      | 1 | 0 | 1 |
0| 1 | 0 | 1 | 0| Turi gida  |
| Kauno Meno galerija   | Galerija      | 0 | 0 | 0 |
0| 0 | 1 | 1 | 0| Turi gida  |
| Kauno Mokslo muziejus | Muziejus      | 1 | 0 | 1 |
0| 1 | 0 | 1 | 0| Turi gida  |
| Kauno Karo Muziejus   | Muziejus      | 0 | 0 | 0 |
0| 0 | 0 | 1 | 3.3| Neturi gido |
| Kauno Zoologijos sodas | Parkas        | 1 | 1 | 1 |
1| 1 | 1 | 1 | 5| Turi gida  |
| Kauno Karo Muziejus   | Muziejus      | 0 | 0 | 0 |
0| 0 | 1 | 1 | 0| Turi gida  |
| Kauno Šventas sodas   | Parkas        | 1 | 1 | 1 |
1| 1 | 1 | 1 | 0| Turi gida  |
| Kauno Miesto Muziejus | Muziejus      | 0 | 0 | 0 |
1| 0 | 1 | 1 | 2.5| Turi gida  |
-----
```

Vilnius

Antanas Antanauskas

```
-----
| Pavadinimas          | Tipas          | Pirmad | Antrad | Treciad |
Ketvirt| Penkt | Sest  | Sekmad | Kaina| Turi gida  |
-----
| Vilniaus Meno galerija | Galerija      | 1 | 0 | 1 |
0| 1 | 0 | 1 | 3.3| Neturi gido |
| Vilniaus Muzikos Muziejus | Muziejus      | 1 | 0 | 1 |
0| 1 | 0 | 1 | 3.3| Turi gida  |
| Vilniaus Karo Muziejus   | Muziejus      | 0 | 0 | 1 |
0| 1 | 0 | 1 | 0| Turi gida  |
| Vilniaus Antano Muziejus | Muziejus      | 1 | 0 | 1 |
0| 1 | 0 | 1 | 3.3| Neturi gido |
| Vilniaus Memorialinis parkas | Parkas        | 1 | 0 | 0 |
0| 1 | 0 | 1 | 0| Neturi gido |
| Vilniaus Uno Muziejus   | Muziejus      | 1 | 1 | 1 |
1| 1 | 1 | 1 | 3.3| Neturi gido |
| Vilniaus Petro Muziejus | Muziejus      | 1 | 0 | 1 |
0| 1 | 0 | 0 | 3.3| Neturi gido |
-----
```

Vilniaus Gedimino Muziejus	Muziejus	0	0	0	
0 1	0	1	0	Neturi gido	
Vilniaus MO Muziejus	Muziejus	1	1	1	
1 1	1	1	6.9	Turi gidą	
Vilniaus Žirgų Muziejus	Muziejus	1	1	1	
1 1	1	1	6.9	Neturi gido	
Vilniaus Miesto Muziejus	Muziejus	0	0	0	
0 0	1	1	0	Neturi gido	

Ekrane spausdinami rezultai:

```
Savarankiško darbo užduotis: U2-8

Pradiniai duomenys yra atspausdinti startingData.txt faile

Muziejų, kurie dirba daugiausiai per savaitę sąrašas:
-----
|Atsakingas asmuo|Pavadinimas|Miestas|Tipas|Pirmad|Antradi|Treciad|Ketvirt|Penkt|Sest|Sekmad|Kaina|Turi gidą|
-----
|Petras Petrauskas|Kauno Zoologijos sodas|Kaunas|Parkas|1|1|1|1|1|1|1|5|Turi gidą|
|Petras Petrauskas|Kauno Šventas sodas|Kaunas|Parkas|1|1|1|1|1|1|1|0|Turi gidą|
|Antanas Antanasuskas|Vilniaus Uno Muziejus|Vilnius|Muziejus|1|1|1|1|1|1|1|3.3|Neturi gido|
|Antanas Antanasuskas|Vilniaus MO Muziejus|Vilnius|Muziejus|1|1|1|1|1|1|1|6.9|Turi gidą|
|Antanas Antanasuskas|Vilniaus Žirgų Muziejus|Vilnius|Muziejus|1|1|1|1|1|1|1|6.9|Neturi gido|
-----

Mieste Kaunas yra daugiau muziejų su gidais nei mieste Vilnius. Muziejų su gidais kiekis: 7

Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Kaunas.txt
Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Vilnius.txt

Press any key to continue . . .
```

Į tekstinius failus spausdinami rezultatai:

Nemokami_Kaunas.txt:

Nemokami muziejai mieste Kaunas
Tipas, Pavadinimas
Muziejus, Kauno Karo Muziejus,
Galerija, Kauno Meno galerija, Tik Savaitgaliais!
Muziejus, Kauno Mokslo muziejus,
Parkas, Kauno Šventas sodas,

Nemokami_Vilnius.txt:

Nemokami muziejai mieste Vilnius
Tipas, Pavadinimas
Muziejus, Vilniaus Karo Muziejus,
Parkas, Vilniaus Memorialinis parkas,
Muziejus, Vilniaus Gedimino Muziejus,
Muziejus, Vilniaus Miesto Muziejus, Tik Savaitgaliais!

2.4. Dėstytojo pastabos

InOutUtils.cs ne iš eilės įdėtas buvo *Pataisyta*

GetList() metodas negalimas *Pataisyta*

Gautas Pažymys: 7

3. Konteineris

3.1. Darbo užduotis

U3_8. Turistų informacijos centras. Turizmo informacijos centre perorganizuoti ir atskirai surašyti duomenys apie dvejuose miestuose veikiančius muziejus. Keičiasi duomenų formatas. Pirmoje eilutėje – miestas, antroje – atsakingo asmens vardas ir pavardė. Toliau informacija apie muziejus pateikta tokiu pačiu formatu kaip L1 užduotyje, tik nebėra miesto stulpelio.

- Raskite muziejų, kuris dirba daugiausia savaitės dienų. Atspausdinkite ekrane visus jo duomenis. Jei yra keli, spausdinkite visus.
- Raskite, kokio tipo muziejus galima aplankyti kiekviename mieste trečiadieniais, ir atspausdinkite muziejų tipus ekrane.
- Sudarykite nurodytų miestų muziejų, kurių pavadinimai sutampa, sąrašą ir įrašykite jų duomenis į failą „Sutampa.csv“.
- Sudarykite kiekvieno miesto muziejų, kuriuos galima aplankyti nemokamai, sąrašą, į failus „Nemokami_miestas.csv“ įrašykite muziejaus tipą ir pavadinimą. Jei muziejus dirba tik šeštadieniais ir sekmadieniais, atitinkamoje eilutėje įrašykite „TIK SAVAITGALIAIS“. Surikiuokite muziejus pagal tipus ir pavadinimus.

3.2. Programos tekstas

Museum.cs failas:

```
using System;

namespace U3_8
{
    class Museum
    {
        public string Name { get; set; }
        public string Type { get; set; }
        public int Mon { get; set; }
        public int Tues { get; set; }
        public int Wednes { get; set; }
        public int Thurs { get; set; }
        public int Fri { get; set; }
        public int Sat { get; set; }
        public int Sun { get; set; }
        public double Price { get; set; }
        public string Guided { get; set; }
        public int WorkingDays { get; set; }
        public string WeekEnder { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name"> name of musem</param>
        /// <param name="type"> type of museum </param>
        /// <param name="mon"> monday </param>
        /// <param name="tues"> tuesday </param>
        /// <param name="wednes"> wednesday </param>
        /// <param name="thurs"> thursday </param>
        /// <param name="fri"> friday </param>
    }
}
```

```

    /// <param name="sat"> saturday </param>
    /// <param name="sun"> sunday </param>
    /// <param name="price"> price of ticket</param>
    /// <param name="guided"> does museum have a guide </param>
    /// <param name="WorkingDays"> how many days does museum work </param>
    /// <param name="WeekENDER"> string for checking if museum only works on
weekends </param>

    public Museum(string name, string type,
        int mon, int tues, int wednes, int thurs,
        int fri, int sat, int sun, double price, string guided)
    {
        this.Name = name;
        this.Type = type;
        this.Mon = mon;
        this.Tues = tues;
        this.Wednes = wednes;
        this.Thurs = thurs;
        this.Fri = fri;
        this.Sat = sat;
        this.Sun = sun;
        this.Price = price;
        this.Guided = guided;
    }

    /// <summary>
    /// Calculates how many days a week does a museum work
    /// </summary>
    public void CalculateWorkingDays()
    {
        int works = 0;
        works = Mon + Tues + Thurs + Wednes + Fri + Sat + Sun;

        WorkingDays = works;
    }

    /// <summary>
    /// Assigns value to weekend string if museum only works
    /// </summary>
    public void CalculateWeekenders()
    {
        string weekend;

        if ((Mon + Tues + Thurs + Wednes + Fri) == 0 & (Sat + Sun > 0))
        {
            weekend = "Tik Savaitgaliais!";
        }
        else
        {
            weekend = "";
        }

        WeekENDER = weekend;
    }

    /// <summary>
    /// Overrides the method ToString() to return a string in a required
format
    /// </summary>
    /// <returns>A string of information in a required format</returns>
    public override string ToString()
    {
        string line;
        line = string.Format(String.Format("| {0,-30} | {1,-10} | {2,6} |
{3,6} | {4,7} |" +

```

```

        " {5,7}| {6,5} | {7,4} | {8,6} | {9,5}| {10,-11} |", Name, Type,
Mon, Tues,
        Wednes, Thurs, Fri, Sat, Sun, Price, Guided));

    return line;
}

/// <summary>
/// Compares working day ammounts of most working museums of two cities
/// </summary>
public static bool operator <(Museum museum1, Museum museum2)
{
    return museum1.WorkingDays < museum2.WorkingDays;
}

/// <summary>
/// Compares working day ammounts of most working museums of two cities
/// </summary>
public static bool operator >(Museum museum1, Museum museum2)
{
    return museum1.WorkingDays > museum2.WorkingDays;
}

/// <summary>
/// Compares working day ammounts of most working museums of two cities
/// </summary>
public static bool operator ==(Museum museum1, Museum museum2)
{
    return museum1.WorkingDays == museum2.WorkingDays;
}

/// <summary>
/// Compares working day ammounts of most working museums of two cities
/// </summary>
public static bool operator !=(Museum museum1, Museum museum2)
{
    return museum1.WorkingDays != museum2.WorkingDays;
}

/// <summary>
/// Checks if museums ticket price is equal to specified ammount
/// </summary>
public static bool operator ==(Museum museum, int num)
{
    return museum.Price == num;
}

/// <summary>
/// Checks if museums ticket price is equal to specified ammount
/// </summary>
public static bool operator !=(Museum museum, int num)
{
    return museum.Price != num;
}

/// <summary>
/// Security function
/// </summary>
public override int GetHashCode()
{
    return this.Name.GetHashCode();
}

/// <summary>
/// Overrides the method Equals() to compare names of museums

```

```

    /// </summary>
    public override bool Equals(object obj)
    {
        return this.Name == ((Museum)obj).Name;
    }

    /// <summary>
    /// Compares two museum objects by name
    /// </summary>
    public int CompareTo(Museum other)
    {
        if (this.Type.CompareTo(other.Type) < 0)
        {
            return -1;
        }
        else if (this.Type.CompareTo(other.Type) == 0)
        {
            if (this.Name.CompareTo(other.Name) < 0)
            {
                return -1;
            }
            else if (this.Name.CompareTo(other.Name) == 0)
            {
                return 0;
            }
            else
            {
                return 1;
            }
        }
        else
        {
            return 1;
        }

        //return this.Name.CompareTo(other.Name);
    }
}

```

MuseumsContainer.cs failas:

```
using System;

namespace U3_8
{
    class MuseumsContainer
    {
        private Museum[] museums;
        private int Capacity;
        public int Count { get; private set; }

        public MuseumsContainer( int capacity=16)
        {
            this.Capacity = capacity; // int capacity = 16 is the default capacity
of container
            this.museums = new Museum[capacity];

            /// <summary>
            /// makes room for extra museums if current container capacity inst enough
            /// </summary>
            private void EnsureCapacity(int minimumCapacity)
            {
                if (minimumCapacity > this.Capacity)
                {
                    Museum[] temp = new Museum[minimumCapacity];
                    for (int i = 0; i < this.Count; i++)
                    {
                        temp[i] = this.museums[i];
                    }
                    this.Capacity = minimumCapacity;
                    this.museums = temp;
                }
            }

            /// <summary>
            /// adds a new instance of museum to container
            /// </summary>
            public void Add(Museum museum)
            {
                if (this.Count == this.Capacity) // container is full
                {
                    EnsureCapacity(this.Capacity + 2);
                }
                this.museums[this.Count++] = museum;
            }

            /// <summary>
            /// returns a certain museums information according to index
            /// </summary>
            public Museum Get(int index)
            {
                return this.museums[index];
            }

            /// <summary>
            /// checks if museum in question exists in container
            /// </summary>
            public bool Contains(Museum museum)
            {
                for (int i = 0; i < this.Count; i++)
                {
```

```

        if (this.museums[i].Equals(museum))
        {
            return true;
        }
    }
    return false;
}

/// <summary>
/// Sorts the container by height (lowest to highest), surname and name
/// </summary>
///
public void Sort()
{
    int n = this.Count;
    if (n > 0)
    {
        for (int i = 0; i < n - 1; i++)
        {
            int min = i;
            for (int j = i + 1; j < n; j++)
            {
                if (museums[j].Type.CompareTo(museums[min].Type) < 0)
                {
                    min = j;
                }
            }

            Museum temp = museums[i];
            museums[i] = museums[min];
            museums[min] = temp;
        }
    }
}

/// <summary>
/// Puts new museum object in index location
/// </summary>
public void Put(Museum newMuseum, int index)
{
    if (index >= 0 && index < Count)
    {
        this.museums[index] = newMuseum;
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// inserts new museum object in index location
/// </summary>
public void Insert(Museum newMuseum, int index)
{
    if (index >= 0 && index <= Count)
    {
        Count++;
        Museum temp = newMuseum;
        for (int i = index; i < Count; i++)
        {
            EnsureCapacity(Count);
            Museum removed = museums[i];
            this.museums[i] = temp;
            temp = removed;
        }
    }
}

```

```

        }
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// removes museum object at index location
/// </summary>
public void RemoveAt(int index)
{
    if (index >= 0 && index < Count)
    {
        for (int i = index; i < Count-1; i++)
        {
            this.museums[i] = museums[i + 1];
        }
        Count--;
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// removes museum object by name while using RemoveAt() method
/// </summary>
///
public void Remove(string name)
{
    bool flag = false;
    for (int i = 0; i < Count; i++)
    {
        if (this.museums[i].Name.Equals(name))
        {
            flag = true;
            RemoveAt(i);
            break;
        }
    }
    if (!flag)
    {
        Console.WriteLine("Muziejaus nurodytu pavadinimu ({0}) saraše
nėra", name);
    }
}
}
}

```

MuseumsRegister.cs failas:

```

using System.Collections.Generic;

namespace U3_8
{
    class MuseumsRegister
    {
        private MuseumsContainer AllMuseums;
    }
}

```

```

public string Manager { get; set; } // pirmos eilutes saugomi duomenys
public string City { get; set; } // antros eilutes saugomi duomenys

/// <summary>
/// creates new list
/// </summary>
public MuseumsRegister()
{
    AllMuseums = new MuseumsContainer();
}

/// <summary>
/// Adds new objects to container
/// </summary>
///
public MuseumsRegister(MuseumsContainer Museums)
{
    AllMuseums = new MuseumsContainer();
    for (int i = 0; i < Museums.Count; i++)
    {
        AllMuseums.Add(Museums.Get(i));
    }
}

/// <summary>
/// gives ammount of museums in container
/// </summary>
public int MuseumsCount()
{
    return AllMuseums.Count;
}

/// <summary>
/// Adds a new object to container
/// </summary>
public void Add(Museum museum)
{
    AllMuseums.Add(museum);
}

/// <summary>
/// returns museum object that is in the indexed place on the container
/// </summary>
public Museum Get(int index)
{
    return AllMuseums.Get(index);
}

/// <summary>
/// Finds museum that works the most
/// </summary>
/// <returns>fuhggvhyghgkiuhkyjkuy</returns>
public int MostWorking()
{
    int ID = 0;

    for (int i = 0; i < AllMuseums.Count; i++)
    {
        if (AllMuseums.Get(i) > AllMuseums.Get(ID))
        {
            ID = i; // index of most working museum
        }
    }

    return ID;
}

```



```

    }

    /// <summary>
    /// Creates list of museums that work the most
    /// </summary>
    public MuseumsContainer ComparedByWorking(Museum MostWorkingMuseum,
MuseumsContainer Filtered, MuseumsRegister compared)
    {
        for (int i = 0; i < compared.AllMuseums.Count; i++)
        {
            if (MostWorkingMuseum == AllMuseums.Get(i))
            {
                if (!Filtered.Contains(AllMuseums.Get(i)))
                {
                    Filtered.Add(AllMuseums.Get(i));
                }
            }
        }

        return Filtered;
    }

    /// <summary>
    /// Creates list of museums that work on Wednesday
    /// </summary>
    public List<string> FindWorkingOnWednesday(List<string> City,
MuseumsRegister compared)
    {
        for (int i = 0; i < compared.AllMuseums.Count; i++)
        {
            if (AllMuseums.Get(i).Wednes == 1)
            {
                City.Add(AllMuseums.Get(i).Type);
            }
        }
        return City;
    }

    /// <summary>
    /// removes duplicate museum types with the help of HashSet(no duplicate
entries are allowed in this type of list)
    /// </summary>
    public void RemoveDuplicateMuseumTypes(ref List<string> CityTypes)
    {
        List<string> NoDuplicates = new List<string>();
        HashSet<string> hash = new HashSet<string>();

        foreach (string line in CityTypes)
        {
            if (hash.Add(line))
            {
                NoDuplicates.Add(line);
            }
        }

        CityTypes = NoDuplicates;
    }

    /// <summary>
    /// Adds to container museum information that shares the same names
    /// </summary>
    public MuseumsContainer DuplicateNames(MuseumsContainer SameNames,
MuseumsRegister register1, MuseumsRegister register2)
    {
        for (int i = 0; i < register1.AllMuseums.Count; i++)

```

```

        {
            for (int j = 0; j < register2.AllMuseums.Count; j++)
            {
                if
(register1.AllMuseums.Get(i).Equals(register2.AllMuseums.Get(j)))
                {
                    SameNames.Add(register1.AllMuseums.Get(i));
                    SameNames.Add(register2.AllMuseums.Get(j));
                }
            }
        }

        return SameNames;
    }

    /// <summary>
    /// Makes a container of free to enter museums
    /// </summary>
    public MuseumsContainer FreeMuseums(MuseumsContainer Freebies,
MuseumsRegister register)
    {
        for (int i = 0; i < register.MuseumsCount(); i++)
        {
            if (register.Get(i) == 0)
            {
                if (!Freebies.Contains(register.Get(i)))
                {
                    Freebies.Add(register.Get(i));
                }
            }
        }

        return Freebies;
    }
}
}

```

InOutUtils.cs failas:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace U3_8
{
    class InOutUtils
    {
        /// <summary>
        /// Read function, reads from specified file
        /// </summary>
        public static MuseumsRegister ReadMuseums(string fileName)
        {
            MuseumsRegister Museums = new MuseumsRegister();
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

            string line;
            Museums.City = Lines[0];
            Museums.Manager = Lines[1];

            for (int i = 2; i < Lines.Length; i++)
            {
                line = Lines[i];
                string[] Bits = line.Split(';'); // splits read line at ; and
assigns to string array for separation
                string name = Bits[0];
                string type = Bits[1];
                int mon = int.Parse(Bits[2]);
                int tues = int.Parse(Bits[3]);
                int wednes = int.Parse(Bits[4]);
                int thurs = int.Parse(Bits[5]);
                int fri = int.Parse(Bits[6]);
                int sat = int.Parse(Bits[7]);
                int sun = int.Parse(Bits[8]);
                double price = double.Parse(Bits[9]);
                string guided = Bits[10];

                Museum museum = new Museum(name, type, mon, tues,
creation                wednes, thurs, fri, sat, sun, price, guided); // new museum

                museum.CalculateWorkingDays(); // calculates how many days of the
week certain museum works
                museum.CalculateWeekenders(); // checks if the museum only works
on weekends
                Museums.Add(museum); // adds new museum to museum register
            }
            return Museums; // returns museum register
        }

        /// <summary>
        /// Prints museum information of container to screen
        /// </summary>
        public static void PrintMuseumsToScreen(MuseumsRegister register, string
header)
        {
            Console.WriteLine(header);
            Console.WriteLine();
            Console.WriteLine(register.City);
            Console.WriteLine(register.Manager);
        }
    }
}
```

```

        Console.WriteLine(new string('-', 128));
        Console.WriteLine(String.Format("| {0,-30} | {1,-10} | {2,6} | {3,6} |
{4,7} |" +
        " {5,7}| {6,5} | {7,4} | {8,6} | {9,5}| {10,-11} |"
        , "Pavadinimas", "Tipas", "Pirmad", "Antrad", "Treciad", "Ketvirt",
        "Penkt",
        "Sest", "Sekmad", "Kaina", "Turi gida"));
        Console.WriteLine(new string('-', 128));
        for (int i = 0; i < register.MuseumsCount(); i++)
        {
            Museum museum = register.Get(i);
            Console.WriteLine(museum.ToString());
        }
        Console.WriteLine(new string('-', 128));

        Console.WriteLine();

    }

    /// <summary>
    /// Prints museum information of container to a specified .txt file
    /// </summary>
    public static void PrintMuseumsToTxt(string fileName, MuseumsRegister
register, string header)
    {
        using (var fn = File.AppendText(fileName))
        {
            fn.WriteLine(header);
            fn.WriteLine();
            fn.WriteLine(register.City);
            fn.WriteLine(register.Manager);
            fn.WriteLine(new string('-', 128));
            fn.WriteLine(String.Format("| {0,-30} | {1,-10} | {2,6} | {3,6} |
{4,7} |" +
            " {5,7}| {6,5} | {7,4} | {8,6} | {9,5}| {10,-11} |"
            , "Pavadinimas", "Tipas", "Pirmad", "Antrad", "Treciad", "Ketvirt",
            "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida"));
            fn.WriteLine(new string('-', 128));
            for (int i = 0; i < register.MuseumsCount(); i++)
            {
                Museum museum = register.Get(i);
                fn.WriteLine(museum.ToString());
            }
            fn.WriteLine(new string('-', 128));

            fn.WriteLine();
        }
    }

    /// <summary>
    /// Prints list of museums to console register method
    /// </summary>
    public static void PrintMostWorkingMuseums(MuseumsRegister register,
MuseumsContainer container) // prints museums in a table in console
    {
        //foreach (Museum museum in container)
        for (int i = 0; i < container.Count; i++)
        {
            Museum museum = container.Get(i);
            Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
            register.Manager, museum.Name, register.City, museum.Type, museum.Mon,
            museum.Tues,

```

```

        museum.Wednes, museum.Thurs, museum.Fri, museum.Sat, museum.Sun,
        museum.Price, museum.Guided);
    }

    Console.WriteLine(new string('-', 169));
}

/// <summary>
/// Prints types of museums of a list
/// </summary>
public static void PrintMuseumTypes(List<string> CityTypes,
MuseumsRegister register) // prints museums in a table in console
{
    Console.WriteLine("Mieste {0} trečiadieniais galima aplankyti šiuos
muziejų tipus:", register.City);
    foreach (string line in CityTypes)
    {
        Console.WriteLine("{0,-10} ", line);
    }

    Console.WriteLine(new string('-', 70));
}

/// <summary>
/// Prints information of museums that have the same name to .csv file
/// </summary>
public static void PrintSameNamesToCSV(string fileName, MuseumsContainer
SameNames, string header)
{
    File.WriteAllText(fileName, header+"\n");

    for (int i = 0; i < SameNames.Count; i++)
    {
        Museum output = SameNames.Get(i);

        string csv =
string.Format("{0};{1};{2};{3};{4};{5};{6};{7};{8};{9};{10};\n", output.Name,
output.Type, output.Mon,
        output.Tues, output.Wednes, output.Thurs, output.Fri,
output.Sat, output.Sun, output.Price, output.Guided);
        File.AppendAllText(fileName, csv);
    }
}

/// <summary>
/// Prints free to enter museum list to a specified .txt file
/// </summary>
public static void PrintFreebies(string fileName, MuseumsContainer
Freebies, MuseumsRegister register) // prints free to enter museums to .txt file
{
    Console.WriteLine("Muziejų, kurie yra nemokami sarašas yra faile {0}",
fileName);

    string[] lines = new string[Freebies.Count + 3];

    lines[0] = String.Format("Nemokami muziejai mieste {0}",
register.City); ;
    lines[1] = String.Format("Tipas, Pavadinimas");

    for (int i = 0; i < Freebies.Count; i++)
    {
        lines[i + 2] = string.Join(", ",
            Freebies.Get(i).Type, Freebies.Get(i).Name,
Freebies.Get(i).WeekEnd);
    }
}

```

```
        File.WriteAllLines(fileName, lines, Encoding.UTF8);  
    }  
}
```

Program.cs failas:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;

namespace U3_8
{
    class Program
    {
        static void Main(string[] args)
        {
            // deletes old result files to reset results
            if (File.Exists(@"startingData.txt"))
            {
                File.Delete(@"startingData.txt");
            }

            if (File.Exists(@"Sutampa.csv"))
            {
                File.Delete(@"Sutampa.csv");
            }

            Console.OutputEncoding = Encoding.UTF8;

            MuseumsRegister register1= InOutUtils.ReadMuseums(@"City1.csv"); //
            reads information from .csv file and adds to register
            //MuseumsRegister register1 = InOutUtils.ReadMuseums(@"City3.csv"); //
            reads information from .csv file and adds to register
            InOutUtils.PrintMuseumsToTxt("startingData.txt", register1, "Pirmojo
            miesto pradiniai duomenys:"); // prints out starting data
            // InOutUtils.PrintMuseumsToScreen(register1,"Pirmojo miesto pradiniai
            duomenys:"); // prints out starting data

            MuseumsRegister register2 = InOutUtils.ReadMuseums(@"City2.csv"); //
            reads information from .csv file and adds to register
            //MuseumsRegister register2 = InOutUtils.ReadMuseums(@"City4.csv"); //
            reads information from .csv file and adds to register
            InOutUtils.PrintMuseumsToTxt("startingData.txt", register2, "Antrojo
            miesto pradiniai duomenys"); // prints out starting data
            // InOutUtils.PrintMuseumsToScreen(register2, "Antrojo miesto
            pradiniai duomenys:"); // prints out starting data

            Console.WriteLine("");

            // First task

            Museum MostWorkingCity1 = register1.Get(register1.MostWorking()); //
            pertvarkyta su GetList() funkcija
            Museum MostWorkingCity2 = register2.Get(register2.MostWorking()); //
            pertvarkyta su GetList() funkcija

            MuseumsContainer City1 = new MuseumsContainer();
            MuseumsContainer City2 = new MuseumsContainer();

            register1.ComparedByWorking(MostWorkingCity1, City1, register1);
            register2.ComparedByWorking(MostWorkingCity2, City2, register2);

            if (MostWorkingCity1 > MostWorkingCity2)
            {
```

```

        Console.WriteLine("Mieste {0} yra darbščiausi muziejai, jų
sarašas:", register1.City);
        Console.WriteLine(new string('-', 169));
        Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
            "Atsakingas asmuo", "Pavadinimas", "Miestas", "Tipas",
"Pirmad", "Antrad", "Treciad", "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida");
        Console.WriteLine(new string('-', 169));
        InOutUtils.PrintMostWorkingMuseums(register1, City1);
        Console.WriteLine("");
    }
    else if (MostWorkingCity1 < MostWorkingCity2)
    {
        Console.WriteLine("Mieste {0} yra darbščiausi muziejai, jų
sarašas:", register2.City);
        Console.WriteLine(new string('-', 169));
        Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
            "Atsakingas asmuo", "Pavadinimas", "Miestas", "Tipas",
"Pirmad", "Antrad", "Treciad", "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida");
        Console.WriteLine(new string('-', 169));
        InOutUtils.PrintMostWorkingMuseums(register2, City2);
        Console.WriteLine("");
    }
    else
    {
        Console.WriteLine("Abiejuose miestuose darbščiausi muziejai dirbo
vienodą dienų skaičių:");
        Console.WriteLine(new string('-', 169));
        Console.WriteLine("|{0,-25} | {1,-30} | {2,-10} | {3,-10} | {4,6}
| {5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |",
            "Atsakingas asmuo", "Pavadinimas", "Miestas", "Tipas",
"Pirmad", "Antrad", "Treciad", "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida");
        Console.WriteLine(new string('-', 169));
        InOutUtils.PrintMostWorkingMuseums(register1, City1);
        InOutUtils.PrintMostWorkingMuseums(register2, City2);
        Console.WriteLine("");
    }

    // second task

    /*
    City1.Clear(); // clears list
    City2.Clear(); // clears list
    */

    List<string> City1Types = new List<string>();
    List<string> City2Types = new List<string>();

    register1.FindWorkingOnWednesday(City1Types, register1); // adds
museums that work on wednesdays to the list
    register2.FindWorkingOnWednesday(City2Types, register2); // adds
museums that work on wednesdays to the list

    register1.RemoveDuplicateMuseumTypes(ref City1Types); // removes
duplicate museum types
    register2.RemoveDuplicateMuseumTypes(ref City2Types); // removes
duplicate museum types

```



```

        InOutUtils.PrintMuseumTypes(City1Types, register1); // prints out
museum types of that city
        InOutUtils.PrintMuseumTypes(City2Types, register2); // prints out
museum types of that city
        Console.WriteLine("");

        // third task

        MuseumsContainer SameNames = new MuseumsContainer();

        register1.DuplicateNames(SameNames, register1, register2); // finds
and adds museums with the same names to container

        if (SameNames.Count != 0)
        {
            Console.WriteLine("Muziejų, kurių pavadinimai sutampa, sąrašas yra
faile Sutampa.csv.");
            InOutUtils.PrintSameNamesToCSV("Sutampa.csv", SameNames, "Muziejų,
kurių vardai sutampa sąrašas: ");
            Console.WriteLine("");
        }
        else
        {
            Console.WriteLine("Muziejų, kurių pavadinimai sutampa nėra.");
            Console.WriteLine("");
        }

        // fourth task

        MuseumsContainer FreebiesCity1 = new MuseumsContainer(); // List
containing museums that are free to enter from first city
        MuseumsContainer FreebiesCity2 = new MuseumsContainer(); // List
containing museums that are free to enter from second city

        FreebiesCity1 = register1.FreeMuseums(FreebiesCity1, register1); //
Adds museums that are free to enter to list
        FreebiesCity2 = register1.FreeMuseums(FreebiesCity2, register2); //
Adds museums that are free to enter to list

        FreebiesCity1.Sort();
        FreebiesCity2.Sort();

        string FreebieFileName; // file name for free to enter museums

        if (FreebiesCity1.Count > 0)
        {
            FreebieFileName = "Nemokami_" + register1.City + ".txt";
            InOutUtils.PrintFreebies(FreebieFileName, FreebiesCity1,
register1);
        }
        else
        {
            Console.WriteLine("Mieste {0} nėra muziejų, kuriuos galima
aplankyti nemokamai", register1.City);
        }

        if (FreebiesCity2.Count > 0)
        {
            FreebieFileName = "Nemokami_" + register2.City + ".txt";
            InOutUtils.PrintFreebies(FreebieFileName, FreebiesCity2,
register2);
            Console.WriteLine("");
        }
        else
        {

```

```
        Console.WriteLine("Mieste {0} nėra muziejų, kuriuos galima  
aplankyti nemokamai", register2.City);  
        Console.WriteLine("");  
    }  
  
    Console.ReadKey();  
}  
  
}
```

3.3. Pradiniai duomenys ir rezultatai

Pirmas testas:

Tiek Kaune tiek Vilniuje yra muziejus su vienodu pavadinimu, Vilniuje nėra nemokamų muziejų.

City1.csv failas:

```
Kaunas
Petras Petrauskas
Kauno Karo Muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Kauno Meno galerija;Galerija;0;0;0;0;0;1;1;0;Neturi gido
Kauno Mokslo muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido
Kauno Zoologijos sodas;Parkas;1;1;1;1;1;1;1;5;Neturi gido
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;1;1;3.3;Turi gida
Vienybės Parkas;Parkas;0;1;1;1;1;1;0;0.69;Neturi gido
```

City2.csv failas:

```
Vilnius
Antanas Antanaukas
Vilniaus Meno galerija;Galerija;1;0;1;0;1;0;1;3.3;Neturi gido
Vilniaus Muzikos Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gida
Vilniaus Karo Muziejus;Muziejus;0;0;1;0;1;0;1;3.3;Turi gida
Vilniaus Antano Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gida
Vilniaus Memorialinis parkas;Parkas;1;1;1;1;1;1;1;3.3;Neturi gido
Vilniaus Uno Muziejus;Muziejus;1;1;1;1;1;1;1;3.3;Turi gida
Vilniaus Petro Muziejus;Muziejus;1;0;1;0;1;0;0;3.3;Neturi gido
Vilniaus Gedimino Muziejus;Muziejus;0;0;0;0;1;0;1;3.3;Turi gida
Vienybės Parkas;Parkas;0;1;1;1;1;0;0;0.69;Neturi gido
```

Programos vaizdas:

```
Savarankiško darbo užduotis: U3-8

Abiejuose miestuose darbščiausi muziejai dirbo vienodą dienų skaičių:
-----
|Atsakingas asmuo | Pavadinimas | Miestas | Tipas | Pirmad | Antradi | Trečiadi | Ketvirt | Penkt | Šešt | Sekmad | Kaina | Turi gida |
-----
|Petras Petrauskas | Kauno Zoologijos sodas | Kaunas | Parkas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | Neturi gido |
-----
|Antanas Antanaukas | Vilniaus Memorialinis parkas | Vilnius | Parkas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3.3 | Neturi gido |
|Antanas Antanaukas | Vilniaus Uno Muziejus | Vilnius | Muziejus | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3.3 | Turi gida |
-----

Mieste Kaunas trečiadieniais galima aplankyti šiuos muziejų tipus:
Muziejus
Parkas
-----
Mieste Vilnius trečiadieniais galima aplankyti šiuos muziejų tipus:
Galerija
Muziejus
Parkas
-----

Muziejų, kurių pavadinimai sutampa, sąrašas yra faile Sutampa.csv.
Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Kaunas.txt
Mieste Vilnius nėra muziejų, kuriuos galima aplankyti nemokamai

Press any key to continue . . .
```

startingData.txt failas:

Pirmojo miesto pradiniai duomenys:

Kaunas

Petras Petrauskas

Pavadinimas					Tipas		Pirmad	Antrad	Trečiad
Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gida				
Kauno Karo Muziejus					Muziejus		1	0	1
0	1	0	1	0	Turi gida				
Kauno Meno galerija					Galerija		0	0	0
0	0	1	1	0	Neturi gido				
Kauno Mokslo muziejus					Muziejus		1	0	1
0	1	0	1	0	Turi gida				
Kauno Karo Muziejus					Muziejus		0	0	0
0	0	0	1	3.3	Neturi gido				
Kauno Zoologijos sodas					Parkas		1	1	1
1	1	1	1	5	Neturi gido				
Kauno Karo Muziejus					Muziejus		0	0	0
0	0	1	1	3.3	Turi gida				
Vienybės Parkas					Parkas		0	1	1
1	1	1	0	0.69	Neturi gido				

Antrojo miesto pradiniai duomenys

Vilnius

Antanas Antanauskas

Pavadinimas					Tipas		Pirmad	Antrad	Trečiad
Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gida				
Vilniaus Meno galerija					Galerija		1	0	1
0	1	0	1	3.3	Neturi gido				
Vilniaus Muzikos Muziejus					Muziejus		1	0	1
0	1	0	1	3.3	Turi gida				
Vilniaus Karo Muziejus					Muziejus		0	0	1
0	1	0	1	3.3	Turi gida				
Vilniaus Antano Muziejus					Muziejus		1	0	1
0	1	0	1	3.3	Turi gida				
Vilniaus Memorialinis parkas					Parkas		1	1	1
1	1	1	1	3.3	Neturi gido				
Vilniaus Uno Muziejus					Muziejus		1	1	1
1	1	1	1	3.3	Turi gida				
Vilniaus Petro Muziejus					Muziejus		1	0	1
0	1	0	0	3.3	Neturi gido				
Vilniaus Gedimino Muziejus					Muziejus		0	0	0
0	1	0	1	3.3	Turi gida				
Vienybės Parkas					Parkas		0	1	1
1	1	0	0	0.69	Neturi gido				

Sutampa.csv failas:

Muziejų, kurių vardai sutampa sąrašas:
Vienybės Parkas;Parkas;0;1;1;1;1;1;0;0.69;Neturi gido;
Vienybės Parkas;Parkas;0;1;1;1;1;0;0.69;Neturi gido;

Nemokami_Kaunas.txt failas:

Nemokami muziejai mieste Kaunas
Tipas, Pavadinimas
Galerija, Kauno Meno galerija, Tik Savaitgaliais!
Muziejus, Kauno Karo Muziejus,
Muziejus, Kauno Mokslo muziejus,

Antras testas:

Sutampančių pavadinimų nėra, Vilniuje ir Kaune buvo nemokamų muziejų.

City3.csv failas:

Kaunas
Petras Petrauskas
Kauno Karo Muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gida;
Kauno Meno galerija;Galerija;0;0;0;0;0;1;1;0;Turi gida;
Kauno Mokslo muziejus;Muziejus;1;0;1;0;1;0;1;0;Turi gida;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;1;3.3;Neturi gido;
Kauno Zoologijos sodas;Parkas;1;1;1;1;1;1;1;5;Turi gida;
Kauno Karo Muziejus;Muziejus;0;0;0;0;0;1;1;0;Turi gida;
Kauno Šventas sodas;Parkas;1;1;1;1;1;1;1;0;Turi gida;
Kauno Miesto Muziejus;Muziejus;0;0;0;1;0;1;1;2.5;Turi gida;

City4.csv failas:

Vilnius
Antanas Antanuskas
Vilniaus Meno galerija;Galerija;1;0;1;0;1;0;1;3.3;Neturi gido;
Vilniaus Muzikos Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Turi gida;
Vilniaus Karo Muziejus;Muziejus;0;0;1;0;1;0;1;0;Turi gida;
Vilniaus Antano Muziejus;Muziejus;1;0;1;0;1;0;1;3.3;Neturi gido;
Vilniaus Memorialinis parkas;Parkas;1;0;0;0;1;0;1;0;Neturi gido;
Vilniaus Uno Muziejus;Muziejus;1;1;1;1;1;1;1;3.3;Neturi gido;
Vilniaus Petro Muziejus;Muziejus;1;0;1;0;1;0;0;3.3;Neturi gido;
Vilniaus Gedimino Muziejus;Muziejus;0;0;0;0;1;0;1;0;Neturi gido;
Vilniaus MO Muziejus;Muziejus;1;1;1;1;1;1;1;6.9;Turi gida;
Vilniaus Žirgų Muziejus;Muziejus;1;1;1;1;1;1;1;6.9;Neturi gido;
Vilniaus Miesto Muziejus;Muziejus;0;0;0;0;0;1;1;0;Neturi gido;

Programos vaizdas:

```
Savarankiško darbo užduotis: U3-8

Abiejuose miestuose darbščiausi muziejai dirbo vienodą dienų skaičių:
-----
|Atsakingas asmuo | Pavadinimas | Miestas | Tipas | Pirmad | Antrad | Treciad | Ketvirt | Penkt | Sest | Sekmad | Kaina | Turi gidą |
-----
|Petras Petrauskas | Kauno Zoologijos sodas | Kaunas | Parkas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | Turi gidą |
|Petras Petrauskas | Kauno Šventas sodas | Kaunas | Parkas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | Turi gidą |
-----
|Antanas Antanasuskas | Vilniaus Uno Muziejus | Vilnius | Muziejus | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3.3 | Neturi gido |
|Antanas Antanasuskas | Vilniaus MO Muziejus | Vilnius | Muziejus | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6.9 | Turi gidą |
|Antanas Antanasuskas | Vilniaus Žirgų Muziejus | Vilnius | Muziejus | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6.9 | Neturi gido |
-----

Mieste Kaunas trečiadieniais galima aplankyti šiuos muziejų tipus:
Muziejus
Parkas
-----
Mieste Vilnius trečiadieniais galima aplankyti šiuos muziejų tipus:
Galerija
Muziejus
-----

Muziejų, kurių pavadinimai sutampa nėra.
Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Kaunas.txt
Muziejų, kurie yra nemokami sąrašas yra faile Nemokami_Vilnius.txt

Press any key to continue . . .
```

startingData.txt failas:

```
Pirmojo miesto pradiniai duomenys:

Kaunas
Petras Petrauskas
-----

| Pavadinimas | Tipas | Pirmad | Antrad | Treciad |
Ketvirt | Penkt | Sest | Sekmad | Kaina | Turi gidą |
-----

| Kauno Karo Muziejus | Muziejus | 1 | 0 | 1 |
0 | 1 | 0 | 1 | 0 | Turi gidą |
| Kauno Meno galerija | Galerija | 0 | 0 | 0 |
0 | 0 | 1 | 1 | 0 | Turi gidą |
| Kauno Mokslo muziejus | Muziejus | 1 | 0 | 1 |
0 | 1 | 0 | 1 | 0 | Turi gidą |
| Kauno Karo Muziejus | Muziejus | 0 | 0 | 0 |
0 | 0 | 0 | 1 | 3.3 | Neturi gido |
| Kauno Zoologijos sodas | Parkas | 1 | 1 | 1 |
1 | 1 | 1 | 1 | 5 | Turi gidą |
| Kauno Karo Muziejus | Muziejus | 0 | 0 | 0 |
0 | 0 | 1 | 1 | 0 | Turi gidą |
| Kauno Šventas sodas | Parkas | 1 | 1 | 1 |
1 | 1 | 1 | 1 | 0 | Turi gidą |
| Kauno Miesto Muziejus | Muziejus | 0 | 0 | 0 |
1 | 0 | 1 | 1 | 2.5 | Turi gidą |
-----

Antrojo miesto pradiniai duomenys

Vilnius
Antanas Antanasuskas
-----

| Pavadinimas | Tipas | Pirmad | Antrad | Treciad |
Ketvirt | Penkt | Sest | Sekmad | Kaina | Turi gidą |
```

Vilniaus Meno galerija	Galerija		1		0		1	
0 1	0	1	3.3	Neturi gido				
Vilniaus Muzikos Muziejus	Muziejus		1		0		1	
0 1	0	1	3.3	Turi gida				
Vilniaus Karo Muziejus	Muziejus		0		0		1	
0 1	0	1	0	Turi gida				
Vilniaus Antano Muziejus	Muziejus		1		0		1	
0 1	0	1	3.3	Neturi gido				
Vilniaus Memorialinis parkas	Parkas		1		0		0	
0 1	0	1	0	Neturi gido				
Vilniaus Uno Muziejus	Muziejus		1		1		1	
1 1	1	1	3.3	Neturi gido				
Vilniaus Petro Muziejus	Muziejus		1		0		1	
0 1	0	0	3.3	Neturi gido				
Vilniaus Gedimino Muziejus	Muziejus		0		0		0	
0 1	0	1	0	Neturi gido				
Vilniaus MO Muziejus	Muziejus		1		1		1	
1 1	1	1	6.9	Turi gida				
Vilniaus Žirgų Muziejus	Muziejus		1		1		1	
1 1	1	1	6.9	Neturi gido				
Vilniaus Miesto Muziejus	Muziejus		0		0		0	
0 0	1	1	0	Neturi gido				


Nemokami_Kaunas.txt failas:

Nemokami muziejai mieste Kaunas
 Tipas, Pavadinimas
 Galerija, Kauno Meno galerija, Tik Savaitgaliais!
 Muziejus, Kauno Karo Muziejus,
 Muziejus, Kauno Mokslo muziejus,
 Parkas, Kauno Šventas sodas,

Nemokami_Vilnius.txt failas:

Nemokami muziejai mieste Vilnius
 Tipas, Pavadinimas
 Muziejus, Vilniaus Gedimino Muziejus,
 Muziejus, Vilniaus Karo Muziejus,
 Muziejus, Vilniaus Miesto Muziejus, Tik Savaitgaliais!
 Parkas, Vilniaus Memorialinis parkas,

3.4. Dėstytojo pastabos

 Vacius Jusas - Pr, 7 lapkr. 2022, 10:09
1. LD2 pastabose nėra pažymių.

 Vacius Jusas - Pr, 7 lapkr. 2022, 10:11
2. Nepasiruošta rikiavimui pagal 2 laukus.

 Vacius Jusas - Pr, 7 lapkr. 2022, 10:18
3. Netinkama klasių išdėstymo seka, konteineris privalo būti aukščiau.

 Vacius Jusas - Pr, 7 lapkr. 2022, 10:19
4. Beprasmiškas if:
else if (museums[j].Type.CompareTo(museums[min].Type) < 0
&& museums[j].CompareTo(museums[min]) < 0)
{
min = j;
}

 Vacius Jusas - Pr, 7 lapkr. 2022, 10:21
for (int i = index; i < Count; i++)
{
this.museums[i] = museums[i + 1]
Kur klaida?

1. Sutvarkyta
2. Sutvarkyta
3. Sutvarkyta
4. Sutvarkyta
5. Klaida: turi būti Count-1 | Sutvarkyta

Gautas Balas - **7**

4. Teksto analizė ir redagavimas

4.1. Darbo užduotis

U4L-8. Skaičių suma

Tekstiniame faile `Knyga.txt` duotas tekstas sudarytas iš žodžių, atskirtų skyrikliais. Skyriklių aibė žinoma. Raskite ir spausdinkite faile `Rodikliai.txt`:

- ilgiausią (didžiausias žodžių kiekis) teksto fragmentą, sudarytą iš žodžių, kur žodžio paskutinė raidė sutampa su kito žodžio pirmąja raide (tarp didžiųjų ir mažųjų raidžių skirtumo nedaryti) ir juos skiriančių skyriklių, bei jo eilutės numerius;
- Žodžių, kuriuos sudaro tik skaitmenys, kiekį. Suskaičiuokite tokių skaičių bendrą sumą.

4.2. Programos tekstas

TaskUtils.cs failas:

```
using System;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;

namespace U4L_8
{
    class TaskUtils
    {
        /// <summary>
        /// Runs input method and calls 'Analise' method do all the work
        /// </summary>
        public static void Process(string fd, string fr)
        {
            // deletes old result file
            if (File.Exists(fr))
            {
                File.Delete(fr);
            }

            string[] lines = File.ReadAllLines(fd, Encoding.UTF8);

            Analise(lines, fr);
        }

        /// <summary>
        /// Removes unnecessary punctuation that repeats itself.
        /// </summary>
        public static void RemoveUnnecessaryPunctuation(ref string[] lines)
        {
            MatchEvaluator evaluator = new MatchEvaluator(ReturnFirstCharacter);
            for (int i = 0; i < lines.Length; i++)
            {
                lines[i] = Regex.Replace(lines[i], @"(\W)\1{1,}", evaluator);
                if (lines[i].Length == 1)
                    lines[i] = "";
            }
        }
    }
    /// <summary>
```

```

/// Returns first character from a regex match.
/// </summary>
public static string ReturnFirstCharacter(Match match)
{
    return Convert.ToString(match.Value[0]);
}

/// <summary>
/// Finds the longest text fragment that
/// is made from words where the last
/// letter is the same as the next
/// word first letter, line numbers.
/// Counts the number of words that are
/// only made from numbers and finds their sum.
/// </summary>
public static void Analise(string[] lines, string fr)
{
    // antra uzduotis
    //-----
--
    int numbersCount = 0; // stores the count of number type words
    int numbersSum = 0; // stores the sum of number type words
    ProcessNumbers(lines, ref numbersCount, ref numbersSum);
    //-----
--

    string allText = String.Join("\n", lines); // Joins all lines to a
single string

    int largestWordCountInFragment = 0; // Stores word count in fragment
    int longestFragmentLineStart = 0; // Stores fragment starting line
index
    int longestFragmentLineEnd = 0; // Stores fragment ending line index

    int longestFragmentStartIndex = 0; // Stores the index where the
fragment begins.
    int longestFragmentEndIndex = 0; // Stores the index where the
fragment ends.

    int currentLine = 0;
    int endLine = 0;

    string pattern = "\\w+|\\n\\s\\W+";
    MatchCollection matches = Regex.Matches(allText, pattern);
    for (int i = 0; i < matches.Count; i++)
    {
        Match match = matches[i];
        CheckIfMatchContainsNewLines(match, ref currentLine);
        if (IsAWord(match))
        {
            string firstWord = match.Value;
            int secondWordIndex = FindNextWord(matches, i);

            if (secondWordIndex > 0 &&
                WordsStartEndMatch(firstWord,
matches[secondWordIndex].Value))
            {
                string secondWord = matches[secondWordIndex].Value;

                int wordsInFragment = 2;

                int lastMatchingIndex = secondWordIndex;

```

```

        FindFragmentEnd(matches, ref lastMatchingIndex, ref
wordsInFragment);
        endLine = currentLine + CountNewLines(matches, i,
lastMatchingIndex);

        if (wordsInFragment > largestWordCountInFragment)
        {
            largestWordCountInFragment = wordsInFragment;
            longestFragmentLineStart = currentLine;
            longestFragmentLineEnd = endLine;
            longestFragmentStartIndex = i;
            longestFragmentEndIndex = lastMatchingIndex;
        }

        currentLine = endLine;
        i = lastMatchingIndex;
    }
}

using (StreamWriter writer = File.CreateText(fr))
{
    if (largestWordCountInFragment > 0)
    {
        // Prints the longest fragment.
        for (int i = longestFragmentStartIndex; i <=
longestFragmentEndIndex; i++)
        {
            writer.Write(matches[i].Value);
        }

        writer.WriteLine();

        if (longestFragmentLineEnd - longestFragmentLineStart > 0)
        {
            writer.WriteLine("Žodžių fragmente: {0}\nEilutės: {1}-
{2}",
                largestWordCountInFragment, longestFragmentLineStart +
1,
                longestFragmentLineEnd + 1);
        }
        else
        {
            writer.WriteLine("Žodžių fragmente: {0}\nEilutė: {1}",
                largestWordCountInFragment, longestFragmentLineStart +
1);
        }

        writer.WriteLine();
    }
    writer.WriteLine("Žodžių, kuriuos sudaro " +
        "tik skaitmenys, kiekis: {0}", numbersCount);
    writer.WriteLine("Žodžių, kuriuos sudaro " +
        "tik skaitmenys, bendra suma: {0}", numbersSum);
}

/// <summary>
/// Counts the amount of new line characters in a fragment.
/// </summary>
public static int CountNewLines(MatchCollection matches, int
fragmentStartIndex, int fragmentEndIndex)
{
    int count = 0;
    for (int i = fragmentStartIndex;
        i < fragmentEndIndex; i++)

```

```

        {
            CheckIfMatchContainsNewLines(matches[i], ref count);
        }
        return count;
    }

    /// <summary>
    /// Checks if a match contains new lines.
    /// </summary>
    public static void CheckIfMatchContainsNewLines(Match match, ref int
count)
    {
        // a match may contain multiple new Lines
        string m = match.Value;

        for (int i = 0; i < m.Length; i++)
        {
            if (m[i] == '\n')
                count++;
        }
    }

    /// <summary>
    /// Checks if the two given words in parameters ending and start match.
    /// </summary>
    public static bool WordsStartEndMatch(string word1, string word2)
    {
        int currentLastIndex = word1.Length - 1; // stores last characters
index

        return Char.ToLower(Convert.ToChar(word1[currentLastIndex])) ==
Char.ToLower(Convert.ToChar(word2[0]));
    }

    /// <summary>
    /// Checks if the regex match is a word. not a number
    /// </summary>
    public static bool IsAWord(Match match)
    {
        return Regex.IsMatch(match.Value, @"^\w+");
    }

    /// <summary>
    /// Finds the next word in a collection of regex matches.
    /// </summary>
    public static int FindNextWord(MatchCollection matches, int currentIndex)
    {
        int nextIndex = -1;
        for (int i = currentIndex + 1; i < matches.Count; i++)
        {
            if (IsAWord(matches[i]))
            {
                return i;
            }
        }
        return nextIndex;
    }

    /// <summary>
    /// Finds the text fragment ending index
    /// and the word count in fragment through references.
    /// </summary>
    public static bool FindFragmentEnd(MatchCollection matches, ref int
matchIndex, ref int wordCount)
    {

```

```

        Match match = matches[matchIndex];
        string firstWord = match.Value;
        int secondWordIndex = FindNextWord(matches, matchIndex);

        // if there is a words after the first word,
        // and if it matches
        if (secondWordIndex > 0 &&
            WordsStartEndMatch(firstWord, matches[secondWordIndex].Value))
        {
            string secondWord = matches[secondWordIndex].Value;
            wordCount++;
            // recursively find the fragment ending.
            FindFragmentEnd(matches, ref secondWordIndex, ref wordCount);
            matchIndex = secondWordIndex;
            return true;
        }
        else
            return false;
    }

    /// <summary>
    /// Finds the count and sum of number words in book.
    /// </summary>
    /// Skaiciuoja visus skaicius, ne tik atskirus skaitinius zodzcius
    public static void ProcessNumbers
        (string[] lines, ref int count, ref int sum)
    {
        for (int i = 0; i < lines.Length; i++)
        {
            string line = lines[i];
            MatchCollection numbers = Regex.Matches(line, @"[-]?\d+");
            foreach (Match number in numbers)
            {
                count++; // counts ammount of number words
                sum += Convert.ToInt32(number.Value); // adds values together
            }
        }
    }
}

```

Program.cs failas:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

/// <summary>
/// Uzduotis
/// </summary>
namespace U4L_8
{
    class Program
    {
        static void Main(string[] args)
        {
            const string CFd = "Knyga.txt"; // duom failas
            const string CFr = "Rodikliai.txt"; // rez failas

            TaskUtils.Process(CFd, CFr);

            Console.ReadKey();
        }
    }
}

```

```
}  
}
```

4.3. Pradiniai duomenys ir rezultatai

Testas 1:

Console vaizdas:



```
C:\WINDOWS\system32\cmd.exe  
Savarankiško darbo užduotis: U4-8L  
Press any key to continue . . .
```

Pradiniai duomenys Knyga.txt:

```
.,!?:;()\'  
Avis smėlis, sausra kompiuteris aviacija, aušra asilas sesė ėriukas saulė  
dog good deaf fall like evil dog love everyone erode engine emergency you ultra  
core  
654645 nasdasffdsf sd dasdas das 444 hhh555
```

Rezultatai Rodikliai.txt:

```
love everyone erode engine emergency you ultra  
Žodžių fragmente: 7  
Eilutė: 3  
  
Žodžių, kuriuos sudaro tik skaitmenys, kiekis: 3  
Žodžių, kuriuos sudaro tik skaitmenys, bendra suma: 655644
```

Testas 2:

Console vaizdas:



```
C:\WINDOWS\system32\cmd.exe
Savarankiško darbo užduotis: U4-8L
Press any key to continue . . .
```

Pradiniai duomenys Knyga1.txt:

```
.,!?:;()\'
Avis smėlis, sausra kompiuteris aviacija, aušra asilas sesė ėriukas saulė
dog good deaf fall like evil dog asd25lol love everyone no erode engine emergency
Y Y Y Y Y Y Y.
you ultra core 90
45 nasdasffdsf sd dasdas das
```

Rezultatai Rodikliai.txt:

```
erode engine emergency
Y Y Y Y Y Y Y.
you ultra
Žodžių fragmente: 12
Eilutės: 3-5

Žodžių, kuriuos sudaro tik skaitmenys, kiekis: 3
Žodžių, kuriuos sudaro tik skaitmenys, bendra suma: 160
```

4.4. Dėstytojo pastabos

Gauta 5

5. Paveldėjimas

5.1. Darbo užduotis

U5_8. Turistų informacijos centras. Turizmo informacijos centre perorganizuoti ir atskirai surašyti duomenys apie trijuose miestuose veikiančius muziejus. Pirmoje eilutėje – miestas, antroje – atsakingo

asmens vardas ir pavardė. Turizmo informacijos centras teikia informaciją apie lankytinas vietas – muziejus, paminklus ir kita. Sukurkite klasę „Place“ (savybės - pavadinimas, adresas, įkūrimo ar pastatymo metai), kurią paveldės klasės „Museum“ (savybės – tipas, 7 savaitės dienos (1 – darbo, 0 – nedarbo), požymis „turi gidą“, bilieto kaina) ir „Statue“ (savybė – autorius, kam skirtas).

- Suskaičiuokite, kiek muziejų turi gidus, rezultatą atspausdinkite ekrane.
- Raskite seniausią lankytiną vietą, visą informaciją apie ją atspausdinkite ekrane.
- Sudarykite visų lankytinų vietų sąrašą, išrikiuokite pagal metus ir pavadinimą, įrašykite visų lankytinų vietų pavadinimus į failą „VisosVietos.csv“.
- Sudarykite lankytinų vietų, kurios buvo pastatytos po Lietuvos nepriklausomybės paskelbimo (po 1990 m.) sąrašą. Į failą „Po1990.csv“ įrašykite visus lankytinos vietos duomenis.

5.2. Programos tekstas

Place.cs failas:

```
namespace U5_8
{
    class Place
    {
        public string Name { get; set; } // name of place
        public string Address { get; set; } // address of place
        public int DoB { get; set; } // date of birth or when was the place
        created

        public Place(string name, string address, int dob)
        {
            this.Name = name;
            this.Address = address;
            this.DoB = dob;
        }
        public Place() { }

        public override bool Equals(object other)
        {
            return this.DoB == ((Place)other).DoB;
        }
        public override int GetHashCode()
        {
            return this.DoB.GetHashCode();
        }

        public int CompareTo(Place other)
        {
            int result = this.DoB.CompareTo(other.DoB);
            if (result == 0)
            {
                return this.Name.CompareTo(other.Name);
            }
            return result;
        }
    }
}

/// <summary>
/// Overrides the method ToString() to return a string in a required
format
/// </summary>
/// <returns>A string of information in a required format</returns>
public override string ToString()
{
    string line;
    line = string.Format(String.Format("| {0,-30} | {1,-30} | {2,8} |",
Name, Address, DoB));

    return line;
}
}
```

Museum.cs failas:

```
using System;

namespace U5_8
{
    class Museum : Place
    {
        public string Type { get; set; }
        public int Mon { get; set; }
        public int Tues { get; set; }
        public int Wednes { get; set; }
        public int Thurs { get; set; }
        public int Fri { get; set; }
        public int Sat { get; set; }
        public int Sun { get; set; }
        public double Price { get; set; }
        public string Guided { get; set; }
        public int WorkingDays { get; set; }
        public string WeekEnders { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name"> name of museum</param>
        /// <param name="type"> type of museum</param>
        /// <param name="mon"> monday</param>
        /// <param name="tues"> tuesday</param>
        /// <param name="wednes"> wednesday</param>
        /// <param name="thurs"> thursday</param>
        /// <param name="fri"> friday</param>
        /// <param name="sat"> saturday</param>
        /// <param name="sun"> sunday</param>
        /// <param name="price"> price of ticket</param>
        /// <param name="guided"> does museum have a guide</param>
        /// <param name="WorkingDays"> how many days does museum work</param>
        /// <param name="WeekEnders"> string for checking if museum only works on
weekends</param>

        public Museum(string name, string adress, int dob, string type,
            int mon, int tues, int wednes, int thurs,
            int fri, int sat, int sun, double price, string
guided):base(name,adress,dob)
        {
            this.Type = type;
            this.Mon = mon;
            this.Tues = tues;
            this.Wednes = wednes;
            this.Thurs = thurs;
            this.Fri = fri;
            this.Sat = sat;
            this.Sun = sun;
            this.Price = price;
            this.Guided = guided;
        }

        /// <summary>
        /// Calculates how many days a week does a museum work
        /// </summary>
        public void CalculateWorkingDays()
        {
            int works = 0;
            works = Mon + Tues + Thurs + Wednes + Fri + Sat + Sun;
        }
    }
}
```

```

        WorkingDays = works;
    }

    /// <summary>
    /// Assigns value to weekend string if museum only works
    /// </summary>
    public void CalculateWeekenders()
    {
        string weekend;

        if ((Mon + Tues + Thurs + Wednes + Fri) == 0 & (Sat + Sun > 0))
        {
            weekend = "Tik Savaitgaliais!";
        }
        else
        {
            weekend = "";
        }

        WeekEnd = weekend;
    }

    /// <summary>
    /// Overrides the method ToString() to return a string in a required
format
    /// </summary>
    /// <returns>A string of information in a required format</returns>
    public override string ToString()
    {
        string line;
        line = string.Format(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10} | {4,6} | {5,6} | {6,7} |" +
            " {7,7}| {8,5} | {9,4} | {10,6} | {11,5}| {12,-11} |", Name,
Adress, DoB, Type, Mon, Tues,
            Wednes, Thurs, Fri, Sat, Sun, Price, Guided));

        return line;
    }

    /// <summary>
    /// Compares working day ammounts of most working museums of two cities
    /// </summary>
    public static bool operator <(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays < museum2.WorkingDays;
    }

    /// <summary>
    /// Compares working day ammounts of most working museums of two cities
    /// </summary>
    public static bool operator >(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays > museum2.WorkingDays;
    }

    /// <summary>
    /// Compares working day ammounts of most working museums of two cities
    /// </summary>
    public static bool operator ==(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays == museum2.WorkingDays;
    }

    /// <summary>
    /// Compares working day ammounts of most working museums of two cities

```

```

    /// </summary>
    public static bool operator !=(Museum museum1, Museum museum2)
    {
        return museum1.WorkingDays != museum2.WorkingDays;
    }

    /// <summary>
    /// Checks if museums ticket price is equal to specified ammount
    /// </summary>
    public static bool operator ==(Museum museum, int num)
    {
        return museum.Price == num;
    }

    /// <summary>
    /// Checks if museums ticket price is equal to specified ammount
    /// </summary>
    public static bool operator !=(Museum museum, int num)
    {
        return museum.Price != num;
    }

    /// <summary>
    /// Security function
    /// </summary>
    public override int GetHashCode()
    {
        return this.Name.GetHashCode();
    }

    /// <summary>
    /// Overrides the method Equals() to compare names of museums
    /// </summary>
    public override bool Equals(object obj)
    {
        return this.Name == ((Museum)obj).Name;
    }

    /// <summary>
    /// Compares two museum objects by name
    /// </summary>
    public int CompareTo(Museum other)
    {
        return this.Name.CompareTo(other.Name);
    }
}
}

```

Statue.cs failas:

```
using System;

namespace U5_8
{
    class Statue : Place
    {
        public string Author { get; set; }
        public string Recipient { get; set; }

        public Statue(string name, string adress, int dob, string author,
            string recipient) : base(name, adress, dob)
        {
            this.Author = author;
            this.Recipient = recipient;
        }
        public override string ToString()
        {
            string line;
            line = string.Format(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-15} | {4,-15} |", Name, Adress, DoB, Author, Recipient));

            return line;
        }
    }
}
```

PlaceContainer.cs failas:

```
using System;

namespace U5_8
{
    class PlaceContainer
    {
        private Place[] places;
        private int Capacity;
        public int Count { get; private set; }

        public PlaceContainer(int capacity = 16)
        {
            this.Capacity = capacity; // int capacity = 16 is the default capacity
of container
            this.places = new Place[capacity];
        }

        /// <summary>
        /// makes room for extra places if current container capacity inst enough
        /// </summary>
        private void EnsureCapacity(int minimumCapacity)
        {
            if (minimumCapacity > this.Capacity)
            {
                Place[] temp = new Place[minimumCapacity];
                for (int i = 0; i < this.Count; i++)
                {
                    temp[i] = this.places[i];
                }
                this.Capacity = minimumCapacity;
                this.places = temp;
            }
        }

        /// <summary>
        /// adds a new instance of place to container
        /// </summary>
        public void Add(Place place)
        {
            if (this.Count == this.Capacity) // container is full
            {
                EnsureCapacity(this.Capacity + 2);
            }
            this.places[this.Count++] = place;
        }

        /// <summary>
        /// returns a certain places information according to index
        /// </summary>
        public Place Get(int index)
        {
            return this.places[index];
        }

        /// <summary>
        /// checks if place in question exists in container
        /// </summary>
        public bool Contains(Place place)
        {
            for (int i = 0; i < this.Count; i++)
            {
                if (this.places[i].Equals(place))
                {

```

```

        return true;
    }
    return false;
}

/// <summary>
/// Adds all register information to one register
/// </summary>
public void CombineAll(ref PlaceContainer EveryPlace, PlaceContainer
register)
{
    for (int i = 0; i < register.Count; i++)
    {
        EveryPlace.Add(register.Get(i));
    }
}

/// <summary>
/// Sorts the container by DoB(date of birth/ creation date) and then by
name
/// </summary>
public void Sort()
{
    int n = this.Count;

    for (int i = 0; i < n-1; i++)
    {
        for (int j = 0; j < n-i-1; j++)
        {
            if (places[j].CompareTo(places[j + 1]) > 0)
            {
                Place temp = places[j];
                places[j] = places[j+1];
                places[j+1] = temp;
            }
            if (places[j].CompareTo(places[j + 1]) == 0)
            {
                Place temp = places[j];
                places[j] = places[j + 1];
                places[j + 1] = temp;
            }
        }
    }
}

/// <summary>
/// Puts new place object in index location
/// </summary>
public void Put(Place newPlace, int index)
{
    if (index >= 0 && index < Count)
    {
        this.places[index] = newPlace;
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// inserts new place object in index location
/// </summary>
public void Insert(Place newPlace, int index)

```

```

{
    if (index >= 0 && index <= Count)
    {
        Count++;
        Place temp = newPlace;
        for (int i = index; i < Count; i++)
        {
            Place removed = places[i];
            this.places[i] = temp;
            temp = removed;
        }
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// removes place object at index location
/// </summary>
public void RemoveAt(int index)
{
    if (index >= 0 && index < Count)
    {
        for (int i = index; i < Count; i++)
        {
            this.places[i] = places[i + 1];
        }
        Count--;
    }
    else
    {
        Console.WriteLine("Nurodytas netinkamas indeksas");
    }
}

/// <summary>
/// removes place object by name while using RemoveAt() method
/// </summary>
public void Remove(string name)
{
    bool flag = false;
    for (int i = 0; i < Count; i++)
    {
        if (this.places[i].Name.Equals(name))
        {
            flag = true;
            RemoveAt(i);
        }
    }
    if (!flag)
    {
        Console.WriteLine("Vietos nurodytu pavadinimu ({0}) sąrašė nėra",
name);
    }
}
}
}

```


PlacesComparator.cs failas:

```
namespace U5_8
{
    class PlacesComparator
    {
        public virtual int Compare(Place a, Place b)
        {
            return a.CompareTo(b);
        }
    }
}
```

Register.cs failas:

```
namespace U5_8
{
    class Register
    {
        private PlaceContainer AllPlaces;
        public string Manager { get; set; } // pirmos eilutes saugomi duomenys
        public string City { get; set; } // antros eilutes saugomi duomenys

        /// <summary>
        /// creates new list
        /// </summary>
        public Register()
        {
            AllPlaces = new PlaceContainer();
        }

        /// <summary>
        /// Adds new objects to container
        /// </summary>
        ///
        public Register(PlaceContainer Places)
        {
            AllPlaces = new PlaceContainer();
            for (int i = 0; i < Places.Count; i++)
            {
                AllPlaces.Add(Places.Get(i));
            }
        }

        /// <summary>
        /// gives ammount of museums in container
        /// </summary>
        public int PlacesCount()
        {
            return AllPlaces.Count;
        }

        /// <summary>
        /// gives ammount of museums in container
        /// </summary>
        public PlaceContainer GetContainer()
        {
            return AllPlaces;
        }

        /// <summary>
        /// Adds a new object to container
        /// </summary>
        public void Add(Place place)
        {
            AllPlaces.Add(place);
        }

        /// <summary>
        /// returns museum object that is in the indexed place on the container
        /// </summary>
        public Place Get(int index)
        {
            return AllPlaces.Get(index);
        }

        /// <summary>
        /// Finds museum that works the most
    }
}
```

```

/// </summary>
public int MostWorking()
{
    int ID = 0;

    for (int i = 0; i < AllPlaces.Count; i++)
    {
        Place max = AllPlaces.Get(ID);
        Place checker = AllPlaces.Get(i);
        if (max is Museum && checker is Museum)
        {
            if ((max as Museum).WorkingDays > (checker as
Museum).WorkingDays)
            {
                ID = i; // index of most working museum
            }
        }
        else
        {
            ID++;
        }
    }

    return ID;
}

/// <summary>
/// counts number of museum guides in register
/// </summary>
public void MuseumGuideCount(ref int count)
{
    for (int i = 0; i < PlacesCount(); i++)
    {
        Place place = Get(i);
        if (place is Museum)
        {
            if ((place as Museum).Guided == "Turi gida")
            {
                count++;
            }
        }
    }
}

/// <summary>
/// finds oldest place in register
/// </summary>
public void OldestPlace(ref int OldDate, ref Place oldestPlace)
{
    for (int i = 0; i < PlacesCount(); i++)
    {
        if (AllPlaces.Get(i).DoB < OldDate)
        {
            OldDate = AllPlaces.Get(i).DoB;
            oldestPlace = AllPlaces.Get(i);
        }
    }
}

/// <summary>
/// Adds places that were created after 1990s to a register
/// </summary>
public void After1990s(ref Register After90s, Register register)

```

```
{
    for (int i = 0; i < register.PlacesCount(); i++)
    {
        if (register.Get(i).DoB > 1990)
        {
            After90s.Add(register.Get(i));
        }
    }
}
}
```

InOut.cs failas:

```
using System;
using System.IO;
using System.Text;

namespace U5_8
{
    class InOut
    {
        /// <summary>
        /// Read function, reads from specified file
        /// </summary>
        public static Register ReadPlaces(string fileName)
        {
            Register Places = new Register();
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

            string line;
            Places.City = Lines[0];
            Places.Manager = Lines[1];

            for (int i = 2; i < Lines.Length; i++)
            {
                line = Lines[i];
                string[] Bits = line.Split(';'); // splits read line at ; and
                assigns to string array for separation
                string name = Bits[0];
                string adress = Bits[1];
                int dob = int.Parse(Bits[2]);

                int range = Bits.Length / 13; // 1-muziejus 0-statula

                switch (range)
                {
                    case 0:
                    {
                        string author = Bits[3];
                        string recipient = Bits[4];

                        Statue statue = new Statue(name, adress, dob, author,
recipient);
                        Places.Add(statue);

                        break;
                    }
                    case 1:
                    {
                        string type = Bits[3];
                        int mon = int.Parse(Bits[4]);
                        int tues = int.Parse(Bits[5]);
                        int wednes = int.Parse(Bits[6]);
                        int thurs = int.Parse(Bits[7]);
                        int fri = int.Parse(Bits[8]);
                        int sat = int.Parse(Bits[9]);
                        int sun = int.Parse(Bits[10]);
                        double price = double.Parse(Bits[11]);
                        string guided = Bits[12];

                        Museum museum = new Museum(name, adress, dob, type,
mon, tues,
                            wednes, thurs, fri, sat, sun, price, guided);

                        museum.CalculateWorkingDays(); // calculates how many
days of the week certain museum works
                    }
                }
            }
        }
    }
}
```

```

        museum.CalculateWeekenders(); // checks if the museum
only works on weekends

        Places.Add(museum);

        break;
    }
}
return Places; // returns museum register
}

/// <summary>
/// Prints museum information of container to screen
/// </summary>
public static void PrintPlacesToScreen(Register register, string header)
{
    Console.WriteLine(header);
    Console.WriteLine();
    Console.WriteLine(register.City);
    Console.WriteLine(register.Manager);
    Console.WriteLine(new string('-', 171));
    Console.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10}
| {4,2} | {5,2} | {6,2} |" +
        " {7,2}| {8,2} | {9,2} | {10,6} | {11,5}| {12,-11} |"
        , "Pavadinimas", "Adresas", "Įkurta", "Tipas", "Pirmad", "Antrad",
"Treciad", "Ketvirt", "Penkt",
        "Sest", "Sekmad", "Kaina", "Turi gida"));
    Console.WriteLine(new string('-', 171));
    for (int i = 0; i < register.PlacesCount(); i++)
    {
        Place place = register.Get(i);

        if(place is Museum)
        {
            Console.WriteLine((place as Museum).ToString());
        }
        else
        {
            Console.WriteLine((place as Statue).ToString() + " *
Kūrėjas/Autorius | Kam skirta statula" + new string(' ', 16)+"|");
        }

    }
    Console.WriteLine(new string('-', 171));

    Console.WriteLine();
}

/// <summary>
/// Prints place information of container to a specified .txt file
/// </summary>
public static void PrintMuseumsToTxt(string fileName, Register register,
string header)
{
    using (var fn = File.AppendText(fileName))
    {
        fn.WriteLine(header);
        fn.WriteLine();
        fn.WriteLine(register.City);
        fn.WriteLine(register.Manager);
        fn.WriteLine(new string('-', 171));
        fn.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10}
| {4,2} | {5,2} | {6,2} |" +
            " {7,2}| {8,2} | {9,2} | {10,6} | {11,5}| {12,-11} |"

```

```

        , "Pavadinimas", "Adresas", "Įkurta", "Tipas", "Pirmad", "Antrad",
"Treciad", "Ketvirt", "Penkt",
        "Sest", "Sekmad", "Kaina", "Turi gida"));
        fn.WriteLine(new string('-', 171));

        for (int i = 0; i < register.PlacesCount(); i++)
        {
            Place place = register.Get(i);
            if (place is Museum)
            {
                fn.WriteLine((place as Museum).ToString());
            }
            else
            {
                fn.WriteLine((place as Statue).ToString() + "      *
Kūrėjas/Autorius | Kam skirta statula" + new string(' ', 16) + "|");
            }
        }
        fn.WriteLine(new string('-', 171));

        fn.WriteLine();
    }
}

/// <summary>
/// Prints places info that were created after 1990s
/// </summary>
public static void PrintAfter1990s(string fileName, Register After90s,
string header)
{
    using (var fn = File.AppendText(fileName))
    {
        fn.WriteLine(header);
        fn.WriteLine();

        fn.WriteLine(new string('-', 171));
        fn.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10}
| {4,2} | {5,2} | {6,2} |" +
            " {7,2}| {8,2} | {9,2} | {10,6} | {11,5}| {12,-11} |"
            , "Pavadinimas", "Adresas", "Įkurta", "Tipas", "Pirmad", "Antrad",
"Treciad", "Ketvirt", "Penkt",
            "Sest", "Sekmad", "Kaina", "Turi gida"));
        fn.WriteLine(new string('-', 171));

        for (int i = 0; i < After90s.PlacesCount(); i++)
        {
            Place place = After90s.Get(i);
            if (place is Museum)
            {
                fn.WriteLine((place as Museum).ToString());
            }
            else
            {
                fn.WriteLine((place as Statue).ToString() + "      *
Kūrėjas/Autorius | Kam skirta statula" + new string(' ', 16) + "|");
            }
        }
        fn.WriteLine(new string('-', 171));

        fn.WriteLine();
    }
}

/// <summary>
/// Prints container information

```

```

    /// </summary>
    public static void PrintAll(string fileName, PlaceContainer AllPlace,
string header)
    {
        using (var fn = File.AppendText(fileName))
        {
            fn.WriteLine(header);
            fn.WriteLine();

            fn.WriteLine(new string('-', 171));
            fn.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10}
| {4,2} | {5,2} | {6,2} |" +
                " {7,2}| {8,2} | {9,2} | {10,6} | {11,5}| {12,-11} |"
                , "Pavadinimas", "Adresas", "Įkurta", "Tipas", "Pirmad", "Antrad",
"Treciad", "Ketvirt", "Penkt",
                "Sest", "Sekmad", "Kaina", "Turi gida"));
            fn.WriteLine(new string('-', 171));

            for (int i = 0; i < AllPlace.Count; i++)
            {
                Place place = AllPlace.Get(i);
                if (place is Museum)
                {
                    fn.WriteLine((place as Museum).ToString());
                }
                else
                {
                    fn.WriteLine((place as Statue).ToString() + " *
Kūrėjas/Autorius | Kam skirta statula" + new string(' ', 16) + "|");
                }
            }
            fn.WriteLine(new string('-', 171));
            fn.WriteLine();
        }
    }
}

```


Program.cs failas:

```
using System;
using System.IO;
using System.Text;

namespace U5_8
{
    class Program
    {
        static void Main(string[] args)
        {
            // deletes old result files to reset results
            if (File.Exists(@"startingData.txt"))
            {
                File.Delete(@"startingData.txt");
            }
            if (File.Exists(@"Po1990.csv"))
            {
                File.Delete(@"Po1990.csv");
            }
            if (File.Exists(@"VisosVietos.csv"))
            {
                File.Delete(@"VisosVietos.csv");
            }

            Console.OutputEncoding = Encoding.UTF8;

            Register register1 = InOut.ReadPlaces(@"City1.csv"); // reads
information from .csv file and adds to register
            InOut.PrintPlacesToScreen(register1, "Pirmojo miesto pradiniai
duomenys:"); // prints out starting data to screen
            InOut.PrintMuseumsToTxt("startingData.txt", register1, "Pirmojo miesto
pradiniai duomenys:"); // prints starting data to csv file

            Register register2 = InOut.ReadPlaces(@"City2.csv"); // reads
information from .csv file and adds to register
            InOut.PrintPlacesToScreen(register2, "Antrojo miesto pradiniai
duomenys:"); // prints out starting data
            InOut.PrintMuseumsToTxt("startingData.txt", register2, "Antrojo miesto
pradiniai duomenys:"); // prints starting data to csv file

            Register register3 = InOut.ReadPlaces(@"City3.csv"); // reads
information from .csv file and adds to register
            InOut.PrintPlacesToScreen(register3, "Treciojo miesto pradiniai
duomenys:"); // prints out starting data
            InOut.PrintMuseumsToTxt("startingData.txt", register3, "Treciojo
miesto pradiniai duomenys:"); // prints starting data to csv file

            Console.WriteLine("");

            // first task
            // Museum count task

            int GuideCount = 0;

            register1.MuseumGuideCount(ref GuideCount);
            register2.MuseumGuideCount(ref GuideCount);
            register3.MuseumGuideCount(ref GuideCount);

            Console.WriteLine("Bendras gidų kiekis miestų muziejuose: {0}",
GuideCount);
```

```

// Second task
// Oldest place

int oldDate = 3000;
Place oldestPlace= new Place();

register1.OldestPlace(ref oldDate, ref oldestPlace);
register2.OldestPlace(ref oldDate, ref oldestPlace);
register3.OldestPlace(ref oldDate, ref oldestPlace);

Console.WriteLine("");
Console.WriteLine("Seniausios lankytinos vietos informacija: ");
if (oldestPlace is Museum)
{
    Console.WriteLine(new string('-', 171));
    Console.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-10} | {4,2} | {5,2} | {6,2} |" +
        " {7,2}| {8,2} | {9,2} | {10,6} | {11,5}| {12,-11} |"
        , "Pavadinimas", "Adresas", "Įkurta", "Tipas", "Pirmad", "Antrad",
        "Treciad", "Ketvirt", "Penkt",
        "Sest", "Sekmad", "Kaina", "Turi gida"));
    Console.WriteLine(new string('-', 171));
    Console.WriteLine(oldestPlace as Museum).ToString();
    Console.WriteLine(new string('-', 171));
}
else
{
    Console.WriteLine(new string('-', 112));
    Console.WriteLine(String.Format("| {0,-30} | {1,-30} | {2,6} | {3,-15} | {4,-15} |"
        , "Pavadinimas", "Adresas", "Įkurta", "Autorius", "Kam
        skirta"));
    Console.WriteLine(new string('-', 112));
    Console.WriteLine(oldestPlace as Statue).ToString();
    Console.WriteLine(new string('-', 112));
}

Console.WriteLine();

// Third task

PlaceContainer AllPlaces = new PlaceContainer();

AllPlaces.CombineAll(ref AllPlaces, register1.GetContainer());
AllPlaces.CombineAll(ref AllPlaces, register2.GetContainer());
AllPlaces.CombineAll(ref AllPlaces, register3.GetContainer());

AllPlaces.Sort();

InOut.PrintAll("VisosVietos.csv", AllPlaces, "Visos lankomos vietos
surikiuotos pagal įkūrimo datą ir pavadinimą: ");
Console.WriteLine("Visų lankytinų vietų surikiuotas sąrašas yra faile
'VisosVietos.csv' ");
Console.WriteLine();

// Fourth task
// Places after 1990s

Register After90s = new Register();

After90s.After1990s(ref After90s, register1);
After90s.After1990s(ref After90s, register2);

```

```

        After90s.After1990s(ref After90s, register3);

        Console.WriteLine("Lankytinų vietų, įkurtų po 1990-ųjų metų sąrašas  
yra faile 'Po1990.csv' ");

        InOut.PrintAfter1990s("Po1990.csv", After90s, "Lankytinų vietų, kurios  
buvo sukurtos po 1990-ųjų metų sąrašas:");
        Console.WriteLine();

        Console.ReadKey();

    }
}

```

5.3. Pradiniai duomenys ir rezultatai

Testas 1:

City1.csv failas:

```

Kaunas
Petras Petrauskas
Kauno Karo Muziejus;Gatvė 312;1918;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Kauno Meno galerija;Gatvė 332;1938;Galerija;0;0;0;0;0;1;1;0;Neturi gido
Kauno Mokslo muziejus;Gatvė 22;1928;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Kauno Karo Muziejus;Gatvė 32;1998;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido
Žmogus;gatvė 25;2020;Petras Mazūras;Žmonėm
Pegasas;gatvė 55;1011;Petras Mazūras;Augintiniam
Arklys;gatvė 45;2010;Petras Mazūras;Arkliam

```

City2.csv failas:

```

Vilnius
Vilniaus Meras
Vilniaus Karo Muziejus;Gatvė 312;1918;Muziejus;1;1;1;1;1;0;1;0;Turi gida
Vilniaus Meno galerija;Gatvė 332;1938;Galerija;0;0;0;0;0;1;1;0;Turi gida
Vilniaus Mokslo muziejus;Gatvė 652;1928;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Vilniaus Karo Muziejus;Gatvė 442;1998;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido
Gyvūnas;gatvė 235;2020;Petras Mazūras;Žmonėm
Vienaragis;gatvė 15;2011;Rikas Sarbievijus;Augintiniam
Mašina;gatvė 325;2010;Antanas Kulka;Arkliam

```

City3.csv failas:

```

Jonava
Jonavos Meras
Jonavos Karo Muziejus;Gatvė 312;1928;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Jonavos Meno galerija;Gatvė 332;1968;Galerija;0;1;1;1;0;1;1;0;Neturi gido
Jonavos Mokslo muziejus;Gatvė 232;1928;Muziejus;1;0;1;0;1;0;1;0;Turi gida
Jonavos Karo Muziejus;Gatvė 352;1928;Muziejus;0;0;0;0;0;0;1;3.3;Neturi gido
Vanduo;gatvė 245;2020;Antanas Mazūras;Žmonėm
Medis;gatvė 525;2011;Petras Mazūras;Augintiniam
Vežimas;gatvė 415;2010;Jonas Mazūras;Arkliam

```

startingData.txt:

Pirmojo miesto pradiniai duomenys:

Kaunas

Petras Petrauskas

| Pavadinimas | Adresas |
Įkurta | Tipas | Pirmad | Antrad | Treciad | Ketvirt | Penkt |
Sest | Sekmad | Kaina | Turi gida |

| Kauno Karo Muziejus | Gatvė 312 |
1918 | Muziejus | 1 | 0 | 1 | 0 | 1 |
0 | 1 | 0 | Turi gida |
| Kauno Meno galerija | Gatvė 332 |
1938 | Galerija | 0 | 0 | 0 | 0 | 0 |
1 | 1 | 0 | Neturi gido |
| Kauno Mokslo muziejus | Gatvė 22 |
1928 | Muziejus | 1 | 0 | 1 | 0 | 1 |
0 | 1 | 0 | Turi gida |
| Kauno Karo Muziejus | Gatvė 32 |
1998 | Muziejus | 0 | 0 | 0 | 0 | 0 |
0 | 1 | 3.3 | Neturi gido |
| Žmogus | gatvė 25 |
2020 | Petras Mazūras | Žmonėm | * Kūrėjas/Autorius |
Kam skirta statula |
| Pegasas | gatvė 55 |
1011 | Petras Mazūras | Augintiniam | * Kūrėjas/Autorius |
Kam skirta statula |
| Arklys | gatvė 45 |
2010 | Petras Mazūras | Arkliam | * Kūrėjas/Autorius |
Kam skirta statula

Antrojo miesto pradiniai duomenys:

Vilnius

Vilniaus Meras

| Pavadinimas | Adresas |
Įkurta | Tipas | Pirmad | Antrad | Treciad | Ketvirt | Penkt |
Sest | Sekmad | Kaina | Turi gida |

Vilniaus Karo Muziejus Gatvė 312									
1918	Muziejus		1		1		1		1
0		1		0	Turi gida				
Vilniaus Meno galerija Gatvė 332									
1938	Galerija		0		0		0		0
1		1		0	Turi gida				
Vilniaus Mokslo muziejus Gatvė 652									
1928	Muziejus		1		0		1		0
0		1		0	Turi gida				1
Vilniaus Karo Muziejus Gatvė 442									
1998	Muziejus		0		0		0		0
0		1		3.3	Neturi gido				
Gyvūnas gatvė 235									
2020	Petras Mazūras		Žmonėm				*	Kūrėjas/Autorius	
Kam skirta statula									
Vienaragis gatvė 15									
2011	Rikas Sarbievijus		Augintiniam				*	Kūrėjas/Autorius	
Kam skirta statula									
Mašina gatvė 325									
2010	Antanas Kulka		Arkliam				*	Kūrėjas/Autorius	
Kam skirta statula									

Treciojo miesto pradiniai duomenys:

Jonava

Jonavos Meras

Pavadinimas Adresas									
Įkurta	Tipas		Pirmad		Antrad		Trečiad		Ketvirt
Sest	Sekmad		Kaina		Turi gida				Penkt

Jonavos Karo Muziejus Gatvė 312									
1928	Muziejus		1		0		1		0
0		1		0	Turi gida				1
Jonavos Meno galerija Gatvė 332									
1968	Galerija		0		1		1		1
1		1		0	Neturi gido				0

Po1990.csv failas:

Lankytinų vietų, kurios buvo sukurtos po 1990-ųjų metų sąrašas:

94

Vanduo		gatvė 245		2020
Antans Mazūras	Žmonėm	* Kūrėjas/Autorius	Kam skirta statula	
Medis		gatvė 525		2011
Petras Mazūras	Augintiniam	* Kūrėjas/Autorius	Kam skirta statula	
Vežimas		gatvė 415		2010 Jonas
Mazūras	Arkliam	* Kūrėjas/Autorius	Kam skirta statula	

VisosVietos.csv failas:

Visos lankomos vietos surikiuotos pagal ikūrimo datą ir pavadinimą:

Pavadinimas			Adresas					
Įkurta	Tipas	Pirmad	Antrad	Treciad	Ketvirt	Penkt		
Sest	Sekmad	Kaina	Turi gida					

Pegasas			gatvė 55					
1011	Petras Mazūras	Augintiniam	* Kūrėjas/Autorius					
Kam skirta statula								
Kauno Karo Muziejus			Gatvė 312					
1918	Muziejus	1	0	1	0	1		
0	1	0	Turi gida					
Vilniaus Karo Muziejus			Gatvė 312					
1918	Muziejus	1	1	1	1	1		
0	1	0	Turi gida					
Jonavos Karo Muziejus			Gatvė 312					
1928	Muziejus	1	0	1	0	1		
0	1	0	Turi gida					
Jonavos Karo Muziejus			Gatvė 352					
1928	Muziejus	0	0	0	0	0		
0	1	3.3	Neturi gido					
Jonavos Mokslo muziejus			Gatvė 232					
1928	Muziejus	1	0	1	0	1		
0	1	0	Turi gida					
Kauno Mokslo muziejus			Gatvė 22					
1928	Muziejus	1	0	1	0	1		
0	1	0	Turi gida					
Vilniaus Mokslo muziejus			Gatvė 652					
1928	Muziejus	1	0	1	0	1		
0	1	0	Turi gida					

Kauno Meno galerija		Gatvė 332	
1938 Galerija	0	0 0	0 0
1 1	0	Neturi gido	
Vilniaus Meno galerija		Gatvė 332	
1938 Galerija	0	0 0	0 0
1 1	0	Turi gida	
Jonavos Meno galerija		Gatvė 332	
1968 Galerija	0	1 1	1 0
1 1	0	Neturi gido	
Kauno Karo Muziejus		Gatvė 32	
1998 Muziejus	0	0 0	0 0
0 1	3.3	Neturi gido	
Vilniaus Karo Muziejus		Gatvė 442	
1998 Muziejus	0	0 0	0 0
0 1	3.3	Neturi gido	
Arklys		gatvė 45	
2010 Petras Mazūras	Arkliam	* Kūrėjas/Autorius	
Kam skirta statula			
Mašina		gatvė 325	
2010 Antanas Kulka	Arkliam	* Kūrėjas/Autorius	
Kam skirta statula			
Vežimas		gatvė 415	
2010 Jonas Mazūras	Arkliam	* Kūrėjas/Autorius	
Kam skirta statula			
Medis		gatvė 525	
2011 Petras Mazūras	Augintiniam	* Kūrėjas/Autorius	
Kam skirta statula			
Vienaragis		gatvė 15	
2011 Rikas Sarbievijus	Augintiniam	* Kūrėjas/Autorius	
Kam skirta statula			
Gyvūnas		gatvė 235	
2020 Petras Mazūras	Žmonėm	* Kūrėjas/Autorius	
Kam skirta statula			
Vanduo		gatvė 245	
2020 Antans Mazūras	Žmonėm	* Kūrėjas/Autorius	
Kam skirta statula			
Žmogus		gatvė 25	
2020 Petras Mazūras	Žmonėm	* Kūrėjas/Autorius	
Kam skirta statula			

Console vaizdas:

Savarankiško darbo užduotis: U5-8

Pirmojo miesto pradiniai duomenys:

Kaunas

Petras Petrauskas

Pavadinimas	Adresas	Įkurta	Tipas	Pirmad	Antrad	Trečiad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą	
Kauno Karo Muziejus	Gatvė 312	1918	Muziejus	1	0	1	0	1	0	1	0	Turi gidą	
Kauno Meno galerija	Gatvė 332	1938	Galerija	0	0	0	0	0	1	1	0	Neturi gido	
Kauno Mokslo muziejus	Gatvė 22	1928	Muziejus	1	0	1	0	0	1	0	1	Turi gidą	
Kauno Karo Muziejus	Gatvė 32	1998	Muziejus	0	0	0	0	0	0	0	1	3.3	Neturi gido
Zmogus	gatvė 25	2020	Petras Mazūras	Zmonėm								Kam skirta statula	
Pegasas	gatvė 55	1011	Petras Mazūras	Augintiniam								* Kūrėjas/Autorius	Kam skirta statula
Arklys	gatvė 45	2010	Petras Mazūras	Arkliam								* Kūrėjas/Autorius	Kam skirta statula

Antrojo miesto pradiniai duomenys

Vilnius

Vilniaus Meras

Pavadinimas	Adresas	Įkurta	Tipas	Pirmad	Antrad	Trečiad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą	
Vilniaus Karo Muziejus	Gatvė 312	1918	Muziejus	1	1	1	1	1	0	1	0	Turi gidą	
Vilniaus Meno galerija	Gatvė 332	1938	Galerija	0	0	0	0	0	1	1	0	Turi gidą	
Vilniaus Mokslo muziejus	Gatvė 652	1928	Muziejus	1	0	1	0	1	0	1	0	Turi gidą	
Vilniaus Karo Muziejus	Gatvė 442	1998	Muziejus	0	0	0	0	0	0	1	3.3	Neturi gido	
Gyvūnas	gatvė 235	2020	Petras Mazūras	Zmonėm								* Kūrėjas/Autorius	Kam skirta statula
Vienaragis	gatvė 15	2011	Rikas Šarbievičius	Augintiniam								* Kūrėjas/Autorius	Kam skirta statula
Mašina	gatvė 325	2010	Antanas Kulka	Arkliam								* Kūrėjas/Autorius	Kam skirta statula

Trečiojo miesto pradiniai duomenys

Jonava

Jonavos Meras

Pavadinimas	Adresas	Įkurta	Tipas	Pirmad	Antrad	Trečiad	Ketvirt	Penkt	Sest	Sekmad	Kaina	Turi gidą	
Jonavos Karo Muziejus	Gatvė 312	1928	Muziejus	1	0	1	0	1	0	1	0	Turi gidą	
Jonavos Meno galerija	Gatvė 332	1968	Galerija	0	1	1	1	0	1	1	0	Neturi gido	
Jonavos Mokslo muziejus	Gatvė 232	1928	Muziejus	1	0	1	0	1	0	1	0	Turi gidą	
Jonavos Karo Muziejus	Gatvė 352	1928	Muziejus	0	0	0	0	0	0	1	3.3	Neturi gido	
Vanduo	gatvė 245	2020	Antanas Mazūras	Zmonėm								* Kūrėjas/Autorius	Kam skirta statula
Medis	gatvė 525	2011	Petras Mazūras	Augintiniam								* Kūrėjas/Autorius	Kam skirta statula
Vežimas	gatvė 415	2010	Jonas Mazūras	Arkliam								* Kūrėjas/Autorius	Kam skirta statula

Bendras gidų kiekis miestų muziejuose: 7

Seniausios lankytinos vietos informacija:


Pavadinimas	Adresas	Įkurta	Autorius	Kam skirta
Pegasas	gatvė 55	1011	Petras Mazūras	Augintiniam

Visų lankytinų vietų surikiuotas sąrašas yra faile 'VisosVietos.csv'


Lankytinų vietų, įkurta po 1990-ųjų metų sąrašas yra faile 'Po1990.csv'

5.4. Dėstytojo pastabos


► Komentarai (3)



Vacius Jusas - Tr, 14 gruod. 2022, 08:47
1. Netinkamas klasių išdėstymas.



Vacius Jusas - Tr, 14 gruod. 2022, 08:49
2. ToString() neužklotas.



Vacius Jusas - Tr, 14 gruod. 2022, 08:51
3. /// Sorts the container by DoB(date of birth/ creation date) and then by name
Ar tai tiesa?

1. Pertvarkyta
2. Užklojimas padarytas
3. Netiesa, rikiuoja net ir jei DoB ir name yra vienodi.

Gautas pažymys - 7