Kauno technologijos universitetas
Informatikos fakultetas

Objektinis programavimas I (P175B118)

Laboratorinių darbų ataskaita

Lukas Kuzmickas IFF-1/6

Studentas

Doc. Dr. Svajūnas Sajavičius

Dėstytojas

Kaunas 2022

TURINYS

1. Duomenų klasė

1.1. Darbo užduotis

Proto mūšis. Studentų atstovybė organizuoja žaidimą „Protų mūšis“. Turite žaidimui paruoštus klausimus. Duomenų faile pateikiama ši informacija: tema, sudėtingumas, klausimo autorius, klausimo tekstas, 4 atsakymo variantai, teisingas atsakymas, balai.

- Raskite, kiek yra I, II ir III sudėtingumo lygio klausimų, rezultatus atspausdinkite ekrane.
- Sudarykite visų klausimų temų sąrašą, surašykite temas į failą „Temos.csv“.
- Sudarykite klausimų rinkinį, kuris turėtų tik skirtingų temų klausimus. Į rinkinį įtraukite ne daugiau kaip 4 klausimus. Į failą „Klausimai.csv“ įrašykite klausimų temas, tekstus ir balų skaičių.

1.2. Programos tekstas

QuestionData.cs

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

namespace L1_U17
{
    /// <summary>
    /// Data class for question data
    /// </summary>
    public class QuestionData
    {
        public string themeOfTheQuestion { get; set; }
        public string questionLevel { get; set; }
        public string questionAuthor { get; set; }
        public string Question { get; set; }
        public string[] allAnswers { get; set; }
        public string correctAnswer { get; set; }
        public int markOfQuestion { get; set; }
        public string firstAnswer { get; set; }
        public string secondAnswer { get; set; }
        public string thirdAnswer { get; set; }
        public string fourthAnswer { get; set; }



        /// <summary>
        /// QuestionData constructor
        /// </summary>
        /// <param name="theme">Question theme</param>
        /// <param name="level">Question level</param>
        /// <param name="author">Question author</param>
        /// <param name="question">Question itself</param>
        /// <param name="allAnswers">All question answers</param>
        /// <param name="fAnswer">First question answer</param>
        /// <param name="sAnswer">Second question answer</param>
```

```
        /// <param name="tAnswer">Third question answer</param>
        /// <param name="frAnswer">Fourth question answer</param>
        /// <param name="correctA">Correct answer</param>
        /// <param name="mark">Point mark for question</param>
        public QuestionData(string theme, string level, string author, string
question, string[] allAnswers, string fAnswer, string sAnswer, string tAnswer,
string frAnswer, string correctA, int mark)
        {
            this.themeOfTheQuestion = theme;
            this.questionLevel = level;
            this.questionAuthor = author;
            this.Question = question;
            this.allAnswers = allAnswers;
            this.firstAnswer = fAnswer;
            this.secondAnswer = sAnswer;
            this.thirdAnswer = tAnswer;
            this.fourthAnswer = frAnswer;
            this.correctAnswer = correctA;
            this.markOfQuestion = mark;
        }
    }
}
```

TaskUtils.cs

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

namespace L1_U17
{
    /// <summary>
    /// TaskUtils class for operations with data
    /// </summary>
    static class TaskUtils
    {
        /// <summary>
        /// Method for finding count of first difficulty questions
        /// </summary>
        /// <param name="list">List with all questions data</param>
        /// <returns>Count of all first difficulty questions</returns>
        public static int FindFirstDifficulty(List<QuestionData>list)
        {
            int questionCount = 0;
            foreach(var item in list)
            {
                if(item.questionLevel == "I")
                {
                    questionCount++;
                }
            }
            return questionCount;
```

```
        }

        /// <summary>
        /// Method for finding the count of second difficulty questions
        /// </summary>
        /// <param name="list">List with all questions data</param>
        /// <returns>Count of all second difficulty question data</returns>
        public static int FindSecondDifficulty(List<QuestionData> list)
        {
            int questionCount = 0;
            foreach (var item in list)
            {
                if (item.questionLevel == "II")
                {
                    questionCount++;
                }
            }
            return questionCount;
        }



        /// <summary>
        /// Method for finding the count of third level difficulty questions
        /// </summary>
        /// <param name="list">List with all question data</param>
        /// <returns>Count of third difficulty questions</returns>
        public static int FindThirdDifficulty(List<QuestionData> list)
        {
            int questionCount = 0;
            foreach (var item in list)
            {
                if (item.questionLevel == "III")
                {
                    questionCount++;
                }
            }
            return questionCount;
        }

        /// <summary>
        /// Method for finding all question themes and putting it to a list of
strings
        /// </summary>
        /// <param name="list">List with question data</param>
        /// <returns>String list of all themes</returns>
        public static List<string> FindAllThemes(List<QuestionData>list)
        {
            List<string> allThemes = new List<string>();
            foreach (var item in list)
            {
                allThemes.Add(item.themeOfTheQuestion);
            }
            return allThemes;

        }

        /// <summary>
        /// Method for finding all 4 different question combination which doesn't
include same themes
        /// </summary>
        /// <param name="list">list of all question data</param>
        /// <param name="alteredData">list of max 4 elements with different
question themes</param>
        /// <returns>List of different theme strings</returns>
```

```
        public static List<string>allDifferentThemes(List<QuestionData>list, ref
List<QuestionData>alteredData)
        {
            List<string> allDifferentThemes = new List<string>();
            int count = 0;
            foreach(var element in list)
            {
                if(!allDifferentThemes.Contains(element.themeOfTheQuestion) &&
count < 4)
                {
                    count++;
                    allDifferentThemes.Add(element.themeOfTheQuestion);
                    alteredData.Add(element);
                }
            }
            return allDifferentThemes;
        }


    }
}
```

InOut.cs

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

namespace L1_U17
{
    /// <summary>
    /// InOut class for data input and output operations
    /// </summary>
    static class InOut
    {

        /// <summary>
        /// Method for reading data from a file and putting it on QuestionData
list
        /// </summary>
        /// <param name="fileName">file with data</param>
        /// <returns>List with all data</returns>
        public static List<QuestionData> ReadFile(string fileName)
        {
            List<QuestionData> list = new List<QuestionData>();
            string[] lines = File.ReadAllLines(fileName);
            foreach(var line in lines)
            {
                string[] values = line.Split(';');
                string theme = values[0];
                string level = values[1];
                string author = values[2];
                string question = values[3];
                string [] allAnswers = values[4].Split(',');
                string firstAnswer = allAnswers[0];
                string secondAnswer = allAnswers[1];
                string thirdAnswer = allAnswers[2];
```

```
                string fourthAnswer = allAnswers[3];
                string correctAnswer = values[5];
                int markQuestion = int.Parse(values[6]);

                QuestionData data = new QuestionData(theme, level, author,
question, allAnswers ,firstAnswer, secondAnswer, thirdAnswer, fourthAnswer,
correctAnswer, markQuestion);
                list.Add(data);
            }
            return list;
        }


        /// <summary>
        /// Method for printing all question data to Console output
        /// </summary>
        /// <param name="data">QuestionData list with all data</param>
        public static void PrintDataToConsole(List<QuestionData>data)
        {
            string lines = new string('-', 198);
            Console.WriteLine(lines);
            Console.WriteLine("|{0,-15}|{1,-30}|{2,-30}|{3,-40}|{4,-35}|{5,-
20}|{6,-20}|", "Klausimo tema", "Klausimo lygis", "Klausimo autorius", "Klausimas"
                , "Galimi atsakymai", "Teisingas atsakymas", "Klausimo balas");
            Console.WriteLine(lines);
            foreach (var element in data)
            {
                Console.WriteLine("|{0,-15}|{1,-30}|{2,-30}|{3,-40}|{4,-35}|{5,-
20}|{6,-20}|", element.themeOfTheQuestion, element.questionLevel,
                    element.questionAuthor, element.Question, " " + "a)" +
element.firstAnswer + " " + "b)" + element
                    .secondAnswer + " " + "c)" + element.thirdAnswer +" " + "d)" +
element.fourthAnswer, element.correctAnswer, element.markOfQuestion);
            }
            Console.WriteLine(lines);
            Console.WriteLine("");
        }


        /// <summary>
        /// Method for printing all question data from a list to file
        /// </summary>
        /// <param name="data">List with all question data</param>
        /// <param name="fileName">File to create data</param>
        public static void PrintDataToFile(List<QuestionData> data, string
fileName)
        {
            using(StreamWriter sw = File.CreateText(fileName))
            {
                string lines = new string('-', 198);
                sw.WriteLine(lines);
                sw.WriteLine("|{0,-15}|{1,-30}|{2,-30}|{3,-40}|{4,-35}|{5,-
20}|{6,-20}|", "Klausimo tema", "Klausimo lygis", "Klausimo autorius", "Klausimas"
                , "Galimi atsakymai", "Teisingas atsakymas", "Klausimo balas");
                sw.WriteLine(lines);
                foreach (var element in data)
                {
                    sw.WriteLine("|{0,-15}|{1,-30}|{2,-30}|{3,-40}|{4,-35}|{5,-
20}|{6,-20}|", element.themeOfTheQuestion, element.questionLevel,
                    element.questionAuthor, element.Question, " " + "a)" +
element.firstAnswer + " " + "b)" + element
                    .secondAnswer + " " + "c)" + element.thirdAnswer + " " + "d)"
+ element.fourthAnswer, element.correctAnswer, element.markOfQuestion);
                }
                sw.WriteLine(lines);
```

```
                sw.WriteLine("");
            }

        }

        /// <summary>
        /// Method for printing all Question themes to a CSV file
        /// </summary>
        /// <param name="allThemes">A list of strings with all themes</param>
        /// <param name="fileName">A csv file to write all themes</param>
        public static void PrintAllThemesToCsv(List<string>allThemes,string
fileName)
        {
            using(StreamWriter sw = File.CreateText(fileName))
            {
                sw.WriteLine("Visos temos");
                foreach(var item in allThemes)
                {
                    sw.WriteLine(item) ;
                }
            }
        }

        /// <summary>
        /// Method for printing QuestionData to CSV file
        /// </summary>
        /// <param name="list">List with all Question data</param>
        /// <param name="fileName">CSV file to write data</param>
        public static void PrintQuestionDataToCsv(List<QuestionData>list, string
fileName)
        {
            using (StreamWriter sw = File.CreateText(fileName))
            {
                foreach (var item in list)
                {
                    sw.WriteLine("{0};{1};{2}", item.themeOfTheQuestion,
item.Question, item.markOfQuestion);
                }
            }
        }

    }
}
```

Program.cs

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

namespace L1_U17
{
    public class Program
    {
        static void Main(string[] args)
        {
            const string mainQuestionData = @"Duomenys.txt";
            const string themeData = @"Temos.csv";
```

```
            const string randomQuestionData = @"Klausimai.csv";
            const string resultFile = @"Rez.txt";
            int firstLevelQuestionCount = 0;
            int secondLevelQuestionCount = 0;
            int thirdLevelQuestionCount = 0;



            List<QuestionData> allData = InOut.ReadFile(mainQuestionData);
            InOut.PrintDataToConsole(allData);
            InOut.PrintDataToFile(allData, resultFile);

            firstLevelQuestionCount = TaskUtils.FindFirstDifficulty(allData);
            Console.WriteLine("I lygio klausimų yra: {0}",
firstLevelQuestionCount);
            secondLevelQuestionCount = TaskUtils.FindSecondDifficulty(allData);
            Console.WriteLine("II lygio klausimų yra: {0}",
secondLevelQuestionCount);
            thirdLevelQuestionCount = TaskUtils.FindThirdDifficulty(allData);
            Console.WriteLine("III lygio klausimų yra: {0}",
thirdLevelQuestionCount);



            List<string> allQuestionThemes = new List<string>();
            allQuestionThemes = TaskUtils.FindAllThemes(allData);
            InOut.PrintAllThemesToCsv(allQuestionThemes, themeData);



            List<QuestionData> randomQuestionSet = new List<QuestionData>();
            List<string> allDifferentThemes = new List<string>();
            allDifferentThemes = TaskUtils.allDifferentThemes(allData, ref
randomQuestionSet);
            InOut.PrintQuestionDataToCsv(randomQuestionSet, randomQuestionData);



        }
    }
}
```

1.3. Pradiniai duomenys ir rezultatai

Pirmieji pradiniai duomenys

```
Matematika;III;Dovydas;5x+6=10;1410,26,1210,1110;26;5
Istorija;I;Deividas;Zalgirio musis ivyko;1410,1310,1210,1110;1410;7
Fizika;III;Lukas;2kg = xg;2000,1310,1210,1110;2000;5
Istorija;II;Daugardas;LT nepriklausomybe;1918,1310,1210,1110;1918;5
Fizika;II;Arvydas;F=ma;pagreitis,1310,1210,1110;pagreitis;5
Astronomija;I;Jonas;musu planeta;zeme,menulis,1210,1110;zeme;5
Matematika;I;Petras;25-2=?;1410,1310,1210,23;23;5
```

Rez.txt

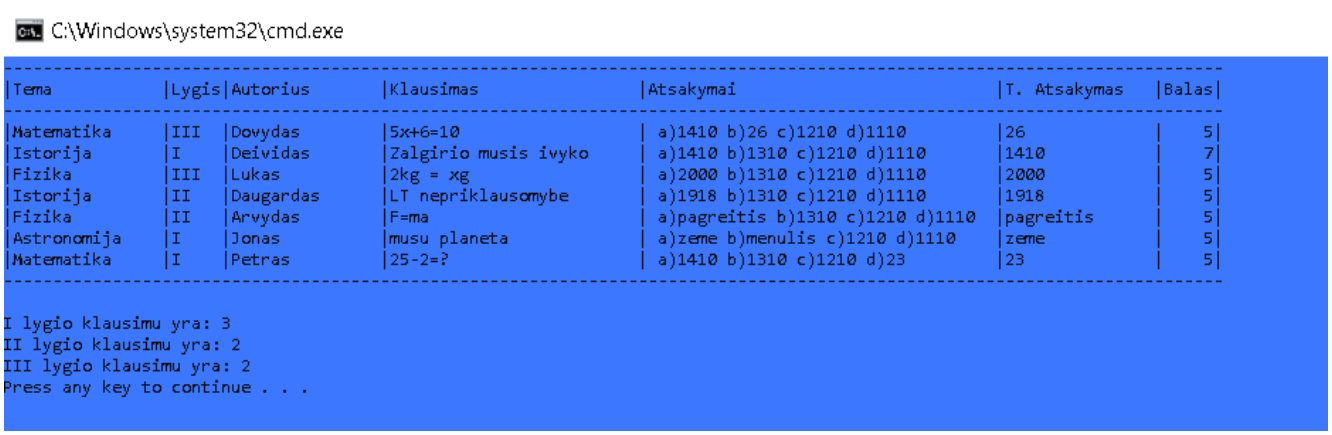
Tema	Lygis	Autorius	Klausimas	Atsakymai	T. Atsakymas	Balas
Matematika	III	Dovydas	5x+6=10	a)1410 b)26 c)1210 d)1110	26	5
Istorija	I	Deividas	Zalgirio musis ivyko	a)1410 b)1310 c)1210 d)1110	1410	7
Fizika	III	Lukas	2kg = xg	a)2000 b)1310 c)1210 d)1110	2000	5
Istorija	II	Daugardas	LT nepriklausomybe	a)1918 b)1310 c)1210 d)1110	1918	5
Fizika	II	Arvydas	F=ma	a)pagreitis b)1310 c)1210 d)1110	pagreitis	5
Astronomija	I	Jonas	musu planeta	a)zeme b)menulis c)1210 d)1110	zeme	5
Matematika	I	Petras	25-2=?	a)1410 b)1310 c)1210 d)23	23	5

Klausimai.csv

```
Matematika;5x+6=10;5

Istorija;Zalgirio musis ivyko;7

Fizika;2kg = xg;5

Astronomija;musu planeta;5
```

Temos.csv

```
Visos temos

Matematika

Istorija

Fizika

Istorija

Fizika

Astronomija

Matematika
```



Konsolinės sąsajos nuotrauka 1pav.

Antrieji pradiniai duomenys

```
Matematika;III;Dovydas;5x+6=10;1410,26,1210,1110;26;5
Istorija;I;Deividas;Zalgirio musis ivyko;1410,1310,1210,1110;1410;7
Fizika;III;Lukas;2kg = xg;2000,1310,1210,1110;2000;5
Istorija;II;Daugardas;LT nepriklausomybe;1918,1310,1210,1110;1918;5
```

Rez.txt

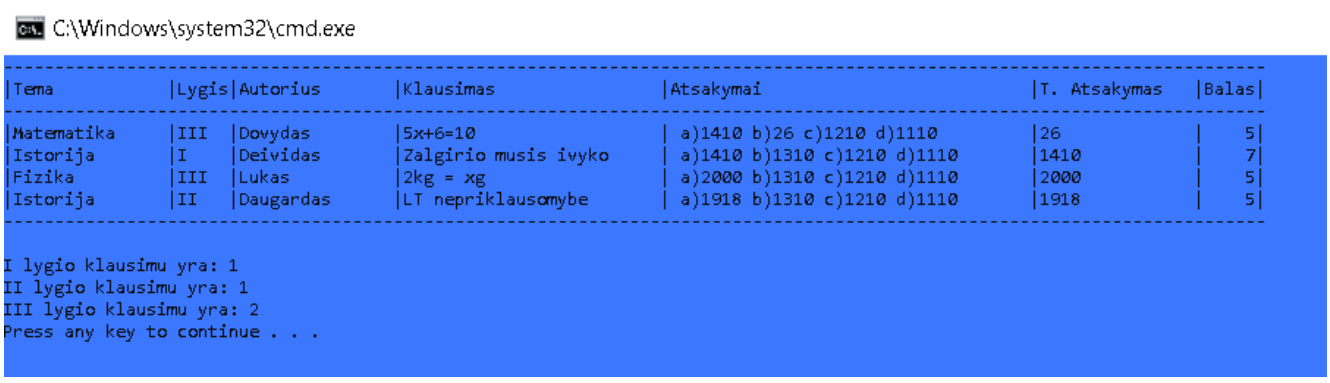
```
-------------------------------------------------------------------------------------------------------------------------------
|Tema           |Lygis|Autorius       |Klausimas              |Atsakymai                        |T. Atsakymas   |Balas|
-------------------------------------------------------------------------------------------------------------------------------
|Matematika     |III  |Dovydas        |5x+6=10                | a)1410 b)26 c)1210 d)1110       |26             |    5|
|Istorija       |I    |Deividas       |Zalgirio musis ivyko   | a)1410 b)1310 c)1210 d)1110     |1410           |    7|
|Fizika         |III  |Lukas          |2kg = xg               | a)2000 b)1310 c)1210 d)1110     |2000           |    5|
|Istorija       |II   |Daugardas      |LT nepriklausomybe     | a)1918 b)1310 c)1210 d)1110     |1918           |    5|
-------------------------------------------------------------------------------------------------------------------------------
```

Klausimai.csv

```
Matematika;5x+6=10;5

Istorija;Zalgirio musis ivyko;7

Fizika;2kg = xg;5
```

Temos.csv

```
Visos temos
Matematika
Istorija
Fizika
Istorija
```



Konsolinės sąsajos nuotrauka 2 pav.

1.4. Dėstytojo pastabos

2. Skaičiavimų klasė

2.1. Darbo užduotis

2.2. Programos tekstas

2.3. Pradiniai duomenys ir rezultatai

2.4. Dėstytojo pastabos

3. Konteineris

3.1. Darbo užduotis

3.2. Programos tekstas

3.3. Pradiniai duomenys ir rezultatai

3.4. Dėstytojo pastabos

4. Teksto analizė ir redagavimas

4.1. Darbo užduotis

U4L-17. Ilgiausias sakinys Dviejuose tekstiniuose failuose Knyga1.txt ir Knyga2.txt duotas tekstas sudarytas iš žodžių, atskirtų skyrikliais. Skyriklių aibė žinoma ir abejuose failuose yra ta pati. Raskite ir spausdinkite faile Rodikliai.txt:

• žodžių, kurie yra tik faile Knyga2.txt, bet nėra faile Knyga1.txt, skaičių ir tokių žodžių sąrašą (ne daugiau nei 10 žodžių), surikiuotą pagal pasikartojimo skaičių mažėjimo tvarka, o kai pasikartojimų skaičius sutampa – pagal abėcėlę;

• ilgiausią sakinį (didžiausias žodžių kiekis), jo ilgį (simboliais ir žodžiais) ir vietą (sakinio pradžios eilutės numerį) pirmame ir antrame faile.

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

• kopijuojamas pirmojo failo tekstas tol, kol sutinkamas pirmasis nenukopijuotas antrojo failo žodis arba pasiekiama failo pabaiga;

• kopijuojamas antrojo failo tekstas tol, kol sutinkamas pirmasis nenukopijuotas pirmojo failo žodis arba pasiekiama failo pabaiga;

• kartojama tol, kol pasiekiama abiejų failų pabaiga.

4.2. Programos tekstas

Words.cs

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

namespace L4_17
{
    /// <summary>
    /// Data class for words
    /// </summary>
    public class Words
    {
        public int Count { get; set; }
        public string Word { get; set; }
        public string Sentence { get; set; }
        public int SentenceCharCount { get; set; }
        public int SentenceIndex { get; set; }

        /// <summary>
        /// Data constructor for words
```

```
        /// </summary>
        public Words()
        {
            this.Count = 1;
            this.Word = "";
        }

        /// <summary>
        /// Data constructor for sentences
        /// </summary>
        /// <param name="sentence">Sentence</param>
        /// <param name="sentenceCharCount">Length of the sentence</param>
        /// <param name="sentenceIndex">Starting index</param>
        public Words(string sentence, int sentenceCharCount, int sentenceIndex)
        {
            Sentence = sentence;
            SentenceCharCount = sentenceCharCount;
            SentenceIndex = sentenceIndex;
        }

        /// <summary>
        /// Method for comparing two words if they are equal
        /// </summary>
        /// <param name="obj">Other object</param>
        /// <returns>If words are equal</returns>
        public override bool Equals(object obj)
        {
            return this.Word == ((Words)obj).Word;
        }

        /// <summary>
        /// HashCode method for comparing words
        /// </summary>
        /// <returns>Hashcode</returns>
        public override int GetHashCode()
        {
            return this.Word.GetHashCode();
        }

        /// <summary>
        /// CompareTo method for comparing two objects
        /// </summary>
        /// <param name="word">word object to compare</param>
        /// <returns>Compare int</returns>
        public int CompareTo(Words word)
        {
            if (this.Count.CompareTo(word.Count) > 0)
                return -1;
            else if (this.Count.CompareTo(word.Count) < 0)
                return 1;
            else
                return this.Word.CompareTo(word.Word);
        }
    }
}
```

TaskUtils.cs

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

```
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace L4_17
{
    /// <summary>
    /// Class for operations with data
    /// </summary>
    internal class TaskUtils
    {
        /// <summary>
        /// Method for finding words that are in SecondBookFile, but not in the
FirstBookFile
        /// </summary>
        /// <param name="fileName1">firstBookData</param>
        /// <param name="fileName2">secondBookData</param>
        /// <param name="punctuation">punctuation</param>
        /// <returns>List of words that are in SecondBookFile only</returns>
        public static List<Words> FindNotRepeatingWords(string fileName1, string
fileName2, char[] punctuation)
        {
            List<Words> RepWords = new List<Words>();
            using (StreamReader reader = new StreamReader(fileName2))
            {
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] parts = line.Split(punctuation,
                    StringSplitOptions.RemoveEmptyEntries);
                    foreach (string part in parts)
                    {
                        Words repeating = new Words();
                        repeating.Word = part;
                        if (FoundInFile(fileName1, punctuation, part) == 0 &&
                        !RepWords.Contains(repeating))
                        {
                            repeating.Count = FoundInFile(fileName2, punctuation,
part);
                            RepWords.Add(repeating);
                        }
                    }
                }
                return RepWords;
            }
        }

        /// <summary>
        /// Additional method for FindNotRepeatingWords for finding count of words
repeated
        /// </summary>
        /// <param name="fileName2">filename</param>
        /// <param name="punctuation">punctuation</param>
        /// <param name="word">word</param>
        /// <returns>int count of words</returns>
        static int FoundInFile(string fileName2, char[] punctuation, string word)
        {
            int count = 0;
            using (StreamReader reader2 = new StreamReader(fileName2))
            {
                string line2;
                while ((line2 = reader2.ReadLine()) != null)
                {
                    string[] parts2 = line2.Split(punctuation,
                    StringSplitOptions.RemoveEmptyEntries);
```

```
                    foreach (string part in parts2)
                    {
                        if (part == word)
                        {
                            count++;
                        }
                    }
                }
                return count;
            }
        }
        /// <summary>
        /// Method for sorting List<Words> elements
        /// </summary>
        /// <param name="repeatings">List with words and repeat counts</param>
        public static void Sort(List<Words> repeatings)
        {
            bool flag = true;
            while (flag)
            {
                flag = false;
                for (int i = 0; i < repeatings.Count - 1; i++)
                {
                    Words a = repeatings[i];
                    Words b = repeatings[i + 1];
                    if (a.CompareTo(b) > 0)
                    {
                        repeatings[i] = b;
                        repeatings[i + 1] = a;
                        flag = true;
                    }
                }
            }
        }

        /// <summary>
        /// Method for finding the length of a string
        /// </summary>
        /// <param name="sentence">string</param>
        /// <returns>length of a string</returns>

        public static int NumberOfCharsInASentence(string sentence)
        {
            int count = 0;
            foreach(char ch in sentence)
            {
                count++;
            }
            return count;
        }


        /// <summary>
        /// Method for finding the index of the longest sentence
        /// </summary>
        /// <param name="fileName">fileName with all text</param>
        /// <param name="longestSentence">longest sentence from fileName</param>
        /// <param name="patern">regex pattern for splitting sentences</param>
        /// <returns>index of longest sentence in fileName</returns>
        public static int FindStartingIndexOfLargestSentence(string fileName,
string longestSentence, string patern)
        {
            int startingIndex = 0;

            string allText = File.ReadAllText(fileName);
```

```
            string[] sentences = Regex.Split(allText, patern);

            foreach(string sentence in sentences)
            {

                if(sentence.Equals(longestSentence))
                {
                    startingIndex = allText.IndexOf(sentence);
                }
            }
            return startingIndex;
        }



            /// <summary>
            /// Method for merging two files together to a string
            /// </summary>
            /// <param name="file1contents">first file string</param>
            /// <param name="file2contents">second file string</param>
            /// <returns>merged file</returns>
            public static string WriteMergedFile(string file1contents, string
file2contents)
            {
            string main = file1contents;
            string aditional = file2contents;
            string result = "";
            while (main != "")
            {
                string word = "empty";
                word = Regex.Match(aditional, @"\w+",
RegexOptions.IgnoreCase).Value;
                int index = main.IndexOf(word);
                if (index == -1)
                {
                    result += main + " ";
                    break;
                }
                else
                {
                    result += main.Substring(0, index);
                    main = main.Remove(0, index + word.Length);
                    int newWordIndex = Regex.Match(main, @"\w").Index;
                    main = main.Remove(0, newWordIndex);
                }
                string temp = main;
                main = aditional;
                aditional = temp;
            }
            result += aditional;
            return result;
        }


        /// <summary>
        /// Regex for splitting text into sentences
        /// </summary>
        /// <param name="fileName">filename</param>
        /// <param name="patern">regex pattern</param>
        public static void SplitIntoSentences(string fileName, string patern)
        {
            string allText = File.ReadAllText(fileName);
            string[] sentences = Regex.Split(allText, patern);
            foreach(var item in sentences)
```

```
            {
                Console.WriteLine(item);
                Console.WriteLine("");
            }
        }
    }
}
```

InOut.cs

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

namespace L4_17
{
    /// <summary>
    /// Class for data input/output
    /// </summary>
    internal class InOut
    {
        /// <summary>
        /// Method for reading filedata to a single string
        /// </summary>
        /// <param name="filename"></param>
        /// <returns></returns>
        public static string ReadFile(string filename)
        {
            return File.ReadAllText(filename, Encoding.UTF8);
        }


        /// <summary>
        /// Method for printing longest sentence data to a file
        /// </summary>
        /// <param name="fileName">result file</param>
        /// <param name="longestSentence">longest sentence string</param>
        /// <param name="length">longest sentence length</param>
        /// <param name="index">longest sentence starting index</param>
        /// <param name="header">additional information</param>
        public static void PrintLongestSentence(string fileName, string
longestSentence,int length,int index, string header)
        {
            using(var writer = File.AppendText(fileName))
            {
                writer.WriteLine(header);
                writer.WriteLine("");
                writer.WriteLine("|Sakinys: {0}|", longestSentence);
                writer.WriteLine("|Ilgis: {0}|", length);
                writer.WriteLine("|Indeksas: {0}|", index);
                writer.WriteLine("");
            }
        }

        /// <summary>
        /// Method for printing words that repeat in one file but not the other
        /// </summary>
        /// <param name="file">result file</param>
        /// <param name="nRepWords">repeating words</param>
```

```
        /// <param name="header">additional information</param>
        public static void PrintRepeating(string file, List<Words> nRepWords,
string header)
        {
            using (var writer = File.AppendText(file))
            {
                writer.WriteLine("");
                writer.WriteLine("");
                writer.WriteLine("{0}", header);
                writer.WriteLine("");
            if (nRepWords.Count == 0)
                {
                    writer.WriteLine("Nera tokiu zodzių");
                }
                if (nRepWords.Count != 0)
                {
                    if (nRepWords.Count > 10)
                    {
                        for (int i = 0; i < 10; i++)
                        {
                            writer.WriteLine("|Zodis: ({0})| Pasikartojimo
skaicius: ({1})|",
                            nRepWords[i].Word, nRepWords[i].Count);
                        }
                    }
                    else
                    {
                        foreach (Words word in nRepWords)
                        {
                            writer.WriteLine("|Zodis: ({0})| Pasikartojimo
skaicius: ({1})|",
                            word.Word, word.Count);
                        }
                    writer.WriteLine("");
                }
            }
        }
        /// <summary>
        /// Method from printing one file to another
        /// </summary>
        /// <param name="filename">path</param>
        /// <param name="contents">string content</param>
        public static void PrintFile(string filename, string contents)
        {
            File.AppendAllText(filename, contents);
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;

namespace L4_17
```

```
{
    internal class Program
    {
        static void Main(string[] args)
        {
            const string firstBookData = "Knyga1.txt";
            const string secondBookData = "Knyga2.txt";
            const string resultFile = "Rodikliai.txt";
            const string rez = "Rez.txt";
            string mergedFile = "ManoKnyga.txt";
            int firstBookLongestSentenceIndex = 0;
            int secondBookLongestSentenceIndex = 0;
            int firstBookSentenceLength = 0;
            int secondBookSentenceLength = 0;
            if(File.Exists(resultFile))
            {
                File.Delete(resultFile);
            }

            if(File.Exists(rez))
            {
                File.Delete(rez);
            }
            char[] punctuation = { ' ', '.', ',', '!', '?', ':', ';', '(', ')',
'\t'};
            string firstFileData = InOut.ReadFile(firstBookData);
            string secondFileData = InOut.ReadFile(secondBookData);
            File.AppendAllText(rez, String.Format("Pirmojo {0} failo pradiniai
duomenys:\n", firstBookData));
            InOut.PrintFile(rez, firstFileData);
            InOut.PrintFile(rez, "\n");
            File.AppendAllText(rez, String.Format("Antrojo {0} failo pradiniai
duomenys:\n", secondBookData));
            InOut.PrintFile(rez, secondFileData);
            List<Words>repeatingWordsInSecond = new List<Words>();
            repeatingWordsInSecond =
TaskUtils.FindNotRepeatingWords(firstBookData, secondBookData, punctuation);
            TaskUtils.Sort(repeatingWordsInSecond);
            InOut.PrintRepeating(resultFile, repeatingWordsInSecond, "Zodziai,
kurie yra tik faile Knyga2.txt, bet nera Knyga1.txt (surikiuoti pagal
pasikartojimo sk. arba abeceles tvarka)\n");
            List<Words> allSentences = new List<Words>();
            List<Words> longestSentence = new List<Words>();
            Words longestSentenceDataFirstFile = new Words();
            Words longestSentenceDataSecondFile = new Words();
            string pattern = @"[.!?]+\s*(?=\p{Lu}|$)"; //regex pattern splitinti
sakiniam
            List<string> allSentencesInFirstDataFile = Regex.Split(firstFileData,
pattern).ToList();
            List<string> longestSentencesInFirstDataFile =
                allSentencesInFirstDataFile.Where(s => s.Length ==
allSentencesInFirstDataFile.Max(l => l.Length)).ToList();
            string longFirstFile = "";
            foreach(var word in longestSentencesInFirstDataFile)
            {
                longFirstFile = word;
            }
            firstBookLongestSentenceIndex =
TaskUtils.FindStartingIndexOfLargestSentence(firstBookData, longFirstFile,
pattern);
            firstBookSentenceLength =
TaskUtils.NumberOfCharsInASentence(longFirstFile);
```

```
            InOut.PrintLongestSentence(resultFile, longFirstFile,
firstBookSentenceLength, firstBookLongestSentenceIndex, "Knyga1.txt ilgiausias
sakinys");
            List<string> allSentencesInSecondDataFile =
Regex.Split(secondFileData, pattern).ToList();
            List<string> longestSentencesInSecondDataFile =
                allSentencesInSecondDataFile.Where(s => s.Length ==
allSentencesInSecondDataFile.Max(l => l.Length)).ToList();
            string longSecondFile = "";
            foreach (var word in longestSentencesInSecondDataFile)
            {
                longSecondFile = word;
            }
            secondBookLongestSentenceIndex =
TaskUtils.FindStartingIndexOfLargestSentence(secondBookData, longSecondFile,
pattern);
            secondBookSentenceLength =
TaskUtils.NumberOfCharsInASentence(longSecondFile);
            InOut.PrintLongestSentence(resultFile, longSecondFile,
secondBookSentenceLength, secondBookLongestSentenceIndex, "Knyga2.txt ilgiausias
sakinys");
            string resultFileData = InOut.ReadFile(resultFile);
            InOut.PrintFile(rez, resultFileData);

            string merged = TaskUtils.WriteMergedFile(firstFileData,
secondFileData);
            File.WriteAllText(mergedFile, merged);



            //TaskUtils.SplitIntoSentences(firstBookData, pattern);
            //TaskUtils.SplitIntoSentences(secondBookData, pattern);
        }
    }
}
```

4.3. Pradiniai duomenys ir rezultatai

Pirmieji pradiniai duomenys

Knyga1.txt

```
Lorem ipsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.
```

Knyga2.txt

```
brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.
```

Rez.txt

```
Pirmojo Knyga1.txt failo pradiniai duomenys:

Lorem ipsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

Antrojo Knyga2.txt failo pradiniai duomenys:
```

brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

Zodziai, kurie yra tik faile Knyga2.txt, bet nera Knyga1.txt (surikiuoti pagal pasikartojimo sk.)

|Zodis: (apsum)| Pasikartojimo skaicius: (1)|

|Zodis: (brem)| Pasikartojimo skaicius: (1)|

Knyga1.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 150|

Knyga2.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 149|

Rodikliai.txt

Zodziai, kurie yra tik faile Knyga2.txt, bet nera Knyga1.txt (surikiuoti pagal pasikartojimo sk.)

|Zodis: (apsum)| Pasikartojimo skaicius: (1)|
|Zodis: (brem)| Pasikartojimo skaicius: (1)|

Knyga1.txt ilgiausias sakinys

```
|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit
saepe et alias debitis quo libero nulla ut fugiat adipisci|
|Ilgis: 126|
|Indeksas: 150|

Knyga2.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit
saepe et alias debitis quo libero nulla ut fugiat adipisci|
|Ilgis: 126|
|Indeksas: 149|
```

ManoKnyga.txt

```
Lorem ipsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione
quidem!
At dolores culpa et impedit architecto ut corrupti culpa.
In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias
debitis quo libero nulla ut fugiat adipisci.
Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil
aut saepe placeat.
 brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione
quidem!
At dolores culpa et impedit architecto ut corrupti culpa.
In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias
debitis quo libero nulla ut fugiat adipisci.
Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil
aut saepe placeat.
```

Antrieji pradiniai duomenys

Knyga1.txt

```
omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias
debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut
saepe placeat.
```

Knyga2.txt

```
brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione
quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias
debitis quo libero nulla ut fugiat adipisci.
```

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

Rez.txt

Pirmojo Knyga1.txt failo pradiniai duomenys:

omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

Antrojo Knyga2.txt failo pradiniai duomenys:

brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

Zodziai, kurie yra tik faile Knyga2.txt, bet nera Knyga1.txt (surikiuoti pagal pasikartojimo sk.)

|Zodis: (amet)| Pasikartojimo skaicius: (1)|

|Zodis: (apsum)| Pasikartojimo skaicius: (1)|

|Zodis: (autem)| Pasikartojimo skaicius: (1)|

|Zodis: (brem)| Pasikartojimo skaicius: (1)|

|Zodis: (doloribus)| Pasikartojimo skaicius: (1)|

|Zodis: (Nam)| Pasikartojimo skaicius: (1)|

|Zodis: (perspiciatis)| Pasikartojimo skaicius: (1)|

|Zodis: (sit)| Pasikartojimo skaicius: (1)|

Knyga1.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 86|

Knyga2.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 149|

Rodikliai.txt

Zodziai, kurie yra tik faile Knyga2.txt, bet nera Knyga1.txt (surikiuoti pagal pasikartojimo sk.)

|Zodis: (amet)| Pasikartojimo skaicius: (1)|

|Zodis: (apsum)| Pasikartojimo skaicius: (1)|

|Zodis: (autem)| Pasikartojimo skaicius: (1)|

|Zodis: (brem)| Pasikartojimo skaicius: (1)|

|Zodis: (doloribus)| Pasikartojimo skaicius: (1)|

|Zodis: (Nam)| Pasikartojimo skaicius: (1)|

|Zodis: (perspiciatis)| Pasikartojimo skaicius: (1)|

|Zodis: (sit)| Pasikartojimo skaicius: (1)|

Knyga1.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 86|

Knyga2.txt ilgiausias sakinys

|Sakinys: In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci|

|Ilgis: 126|

|Indeksas: 149|

ManoKnyga.txt

omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

brem apsum dolor sit amet. Nam autem doloribus ut perspiciatis omnis est ratione quidem!

At dolores culpa et impedit architecto ut corrupti culpa.

In labore distinctio ut maiores ratione hic ullam dicta in suscipit saepe et alias debitis quo libero nulla ut fugiat adipisci.

Ad expedita soluta qui molestiae blanditiis et eligendi dolor et internos nihil aut saepe placeat.

4.4. Dėstytojo pastabos

5. Paveldėjimas

5.1. Darbo užduotis

5.2. Programos tekstas

5.3. Pradiniai duomenys ir rezultatai

5.4. Dėstytojo pastabos