# Упражнения: Стриймове, файлове и директории

Можете да проверите решенията си тук: <https://judge.softuni.org/Contests/4233/10-Streams-Files-and-Directories>.

**Бележка**: За тези задачи следвайте инструкциите за необходимите методи и класове. За всеки проблем изпратете компресирана папка на вашия проект без папките **"bin"** и **"obj"** в нея.

## Четни редове

Напишете програма, която чете **текстов файл** (например input.txt) и отпечатва на конзолата всички **четни редове**. Номерата на редовете започват от **0**. Използвайте **StreamReader.**   
**Бележка**: използвайте следната структура:

|  |
| --- |
| namespace EvenLines  {  public class EvenLines  {  static void Main()  {  string inputFilePath = @"..\..\..\Files\input.txt";  Console.WriteLine(ProcessLines(inputFilePath));  }  public static string ProcessLines(string inputFilePath)  {  }  }  } |

### Примери

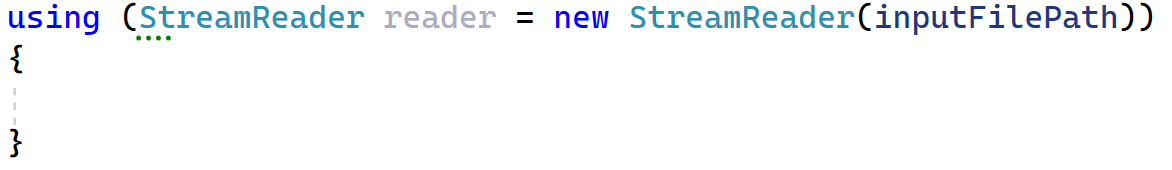
|  |  |
| --- | --- |
| **Входен файл: input.txt** | **Изход (на конзолата)** |
| -I was quick to judge him, but it wasn't his fault.  -Is this some kind of joke?! Is it?  -Quick, hide here. It is safer. | -I was quick to judge him, but it wasn't his fault.  -Quick, hide here. It is safer. |

### Насоки

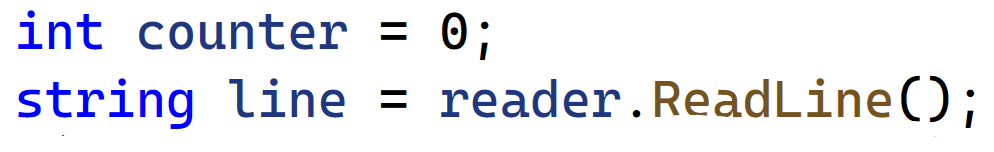
1. В метода **ProcessLines(string inputFilePath)** създайте **StringBuilder**, чрез който ще отпечатвате четните редове:



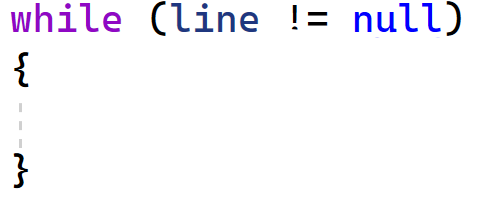
1. След това създайте **StreamReader** чрез който ще прочетете файла. Подайте му като аргумент променливата **inputFilePath**:



1. В **StreamReader-а** създайте променлива, с която ще броите редовете и още една, с която ще прочитате реда от файла:



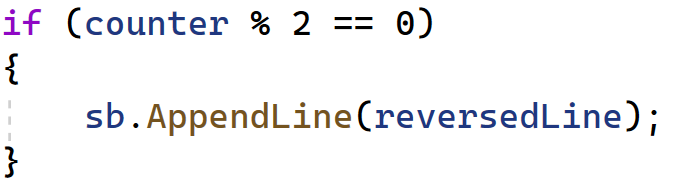
1. След това създайте **while-цикъл**, който ще итерира докато няма следващ ред във **файла**:



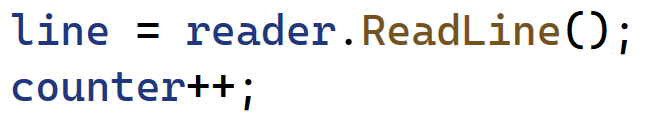
1. В **цикъла** създайте **две променливи**, които ще заменят **символите** **"."**, **","**, **"!"**, **"?"** и **"-"** и ще обръщат реда на думите чрез методите **ReverseWords()** и **ReplaceSymbols()**:



1. Създайте **проверка**, която ще проверя дали **редът е четен**. Ако е така, добавете обърнатия ред в променливата **sb**:



1. След проверката **увеличете променливата**, която брой редовете, **с 1** и прочетете **следващия** **ред**:



1. Накрая върнете **резултата**.

## Нечетни редове

Напишете програма, която чете файл (примерно input.txt) и записва всеки **нечетен** ред в друг файл. Броя на редовете **започват от 0**.

**Бележка**: използвайте следната структура:

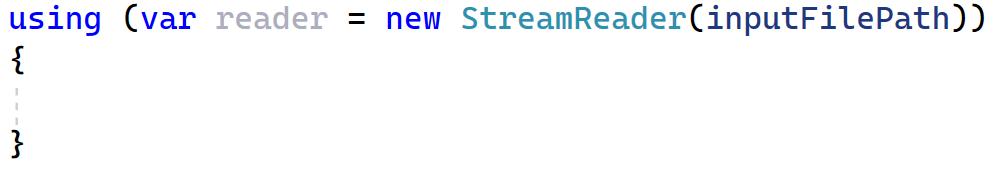
|  |
| --- |
| namespace OddLines  {  public class OddLines  {  static void Main()  {  string inputFilePath = @"..\..\..\Files\input.txt";  string outputFilePath = @"..\..\..\Files\output.txt";  ExtractOddLines(inputFilePath, outputFilePath);  }  public static void ExtractOddLines(string inputFilePath, string outputFilePath)  {  // TODO: напишете вашия код…  }  }  } |

### Примери

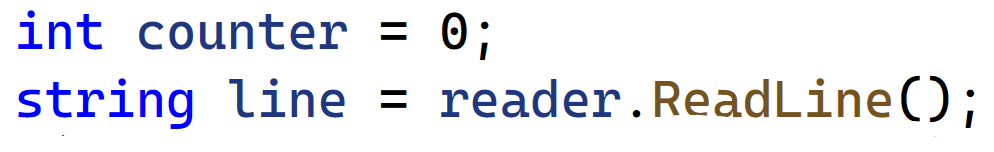
|  |  |
| --- | --- |
| **input.txt** | **output.txt** |
| Two households, both alike in dignity, In fair Verona, where we lay our scene, From ancient grudge break to new mutiny, Where civil blood makes civil hands unclean. From forth the fatal loins of these two foes A pair of star-cross'd lovers take their life; Whose misadventured piteous overthrows Do with their death bury their parents' strife. | In fair Verona, where we lay our scene,  Where civil blood makes civil hands unclean.  A pair of star-cross’d lovers take their life;  Do with their death bury their parents’ strife |

### Насоки

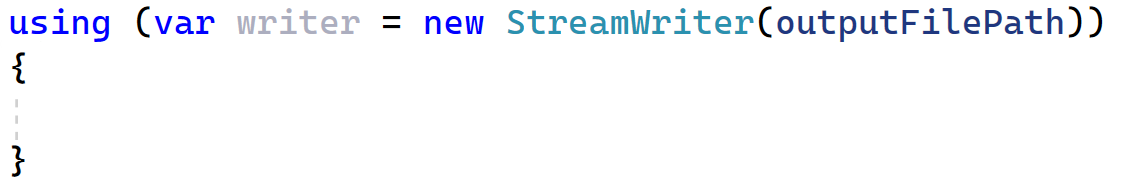
1. В метода **ExtractOddLines(string inputFilePath, string outputFilePath)** създайте **StreamReader** чрез който ще прочетете файла. Подайте му като аргумент променливата **inputFilePath**:



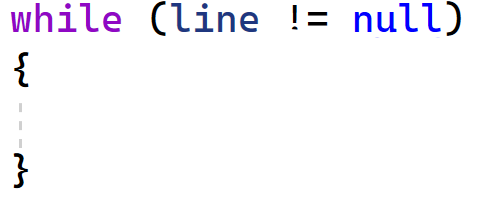
1. В **StreamReader-а** създайте променлива, с която ще броите редовете и още една, с която ще прочитате реда от файла:



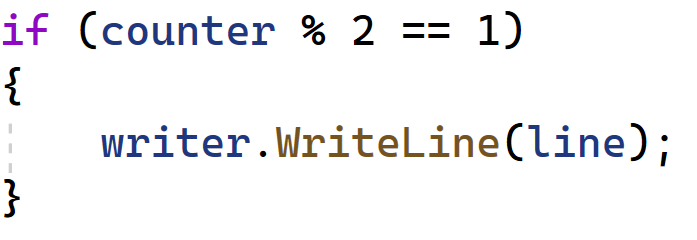
1. След това създайте **StreamWriter** чрез който ще запишете нечетните редове в друг файл. Подайте му като аргумент променливата **outputFilePath**:



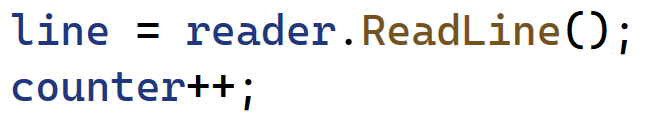
1. В променливата създайте **while-цикъл**, който ще итерира докато няма следващ ред във файла:



1. Създайте **проверка**, която ще проверя дали **редът е нечетен**. Ако е така, добавете реда в другия файл:



1. Накрая увеличете **променливата**, която брои редовете, **с 1** и прочетете **следващия ред**:



## Номера на редове

Напишете програма, която чете **текстов файл** (например input.txt) и добавя **номера на редовете**. Резултатът трябва да се напише на **друг текстов файл** (например output.txt). Използвайте статичния клас, за да **прочетете** и да **напишете** всички редове на входния и изходния файл.

**Бележка: използвайте следната структура**

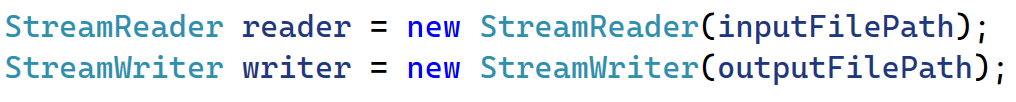
|  |
| --- |
| namespace LineNumbers  {  public class LineNumbers  {  static void Main()  {  string inputFilePath = @"..\..\..\Files\input.txt";  string outputFilePath = @"..\..\..\Files\output.txt";    RewriteFileWithLineNumbers(inputFilePath, outputFilePath);  }  public static void RewriteFileWithLineNumbers(string inputFilePath, string outputFilePath)  {  }  }  } |

### Примери

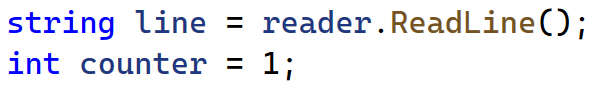
|  |  |
| --- | --- |
| **input.txt** | **output.txt** |
| -I was quick to judge him, but it wasn't his fault.  -Is this some kind of joke?! Is it?  -Quick, hide here. It is safer. | 1. -I was quick to judge him, but it wasn't his fault.  2. -Is this some kind of joke?! Is it?  3. -Quick, hide here. It is safer. |

### Насоки

1. В метода **RewriteFileWithLineNumbers(string inputFilePath, string outputFilePath)** създайте **StreamReader**, чрез който ще прочетете файла, и **StreamWriter**, чрез който да запишете изхода:



1. След като отворите **StreamReader**-а, създайте променлива, с която ще **броите редовете** и още една, с която ще прочетете **текущия ред** от файла:



1. След това използвайте **StreamWriter**-а, чрез който ще запишете **всички редове** с техния **номер** в друг файл.

A white background with blue text

Description automatically generated

1. В променливата създайте **while-цикъл**, който ще итерира докато има следващ ред във файла, и ще записва **необходимата информация**:

A close-up of a white background

Description automatically generated

## Размер на папка

Дадена ви е папка във файловата система (примерно TestFolder). Изчислите **размера** на файловете в тази папка, **включително** **подпапките**. Резултатът трябва да бъде написан в друг файл (примерно оutput.txt). Размерът трябва да бъде в **килобайти**.

**Бележка**: Използвайте следната структура:

|  |
| --- |
| namespace FolderSize  {  public class FolderSize  {  static void Main()  {  string folderPath = @"..\..\..\Files\TestFolder";  string outputFilePath = @"..\..\..\Files\output.txt";    GetFolderSize(folderPath, outputFilePath);  }  public static void GetFolderSize(string folderPath, string outputFilePath)  {  // TODO: напишете вашия код…  }  }  } |

### Пример

|  |
| --- |
| **оutput.txt** |
| 0.2421875 KB |

## Брой думи

Напишете програма, която чете списък от думи от даден файл (примерно words.txt) и намира колко пъти се повтаря всяка дума в друг файл (примерно text.txt). Съвпадението трябва да бъде **case-insensitive. Резултатът трябва да бъде написан в изходен файл (примерно output.txt).**

**Бележка**: използвайте следната структура:

|  |
| --- |
| namespace WordCount  {  public class WordCount  {  static void Main()  {  string wordsFilePath = @"..\..\..\Files\words.txt";  string textFilePath = @"..\..\..\Files\text.txt";  string outputFilePath = @"..\..\..\Files\output.txt";  CalculateWordCounts(wordsFilePath, textFilePath, outputFilePath);  }  public static void CalculateWordCounts(string wordsFilePath, string textFilePath, string outputFilePath)  {  // TODO: напишете вашия код…  }  }  } |

### Примери

|  |  |  |
| --- | --- | --- |
| **words.txt** | **text.txt** | **output.txt** |
| quick is fault | -I was quick to judge him, but it wasn't his fault.  -Is this some kind of joke?! Is it?  -Quick, hide here…It is safer. | quick – 2  is - 3  fault - 1 |

### Насоки

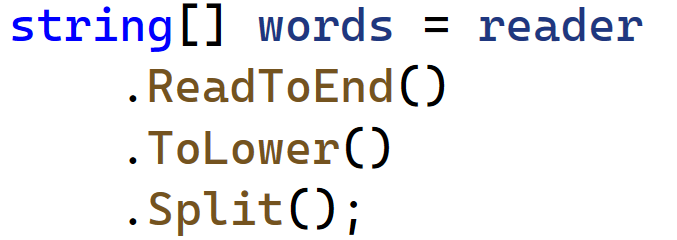
1. Създайте **речник**, който ще брой думите във файла:



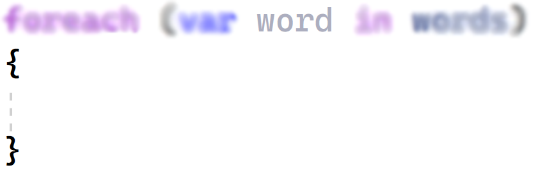
1. Създайте **StreamReader** чрез който ще прочетете файла. Подайте му като аргумент променливата **wordsFilePath**:



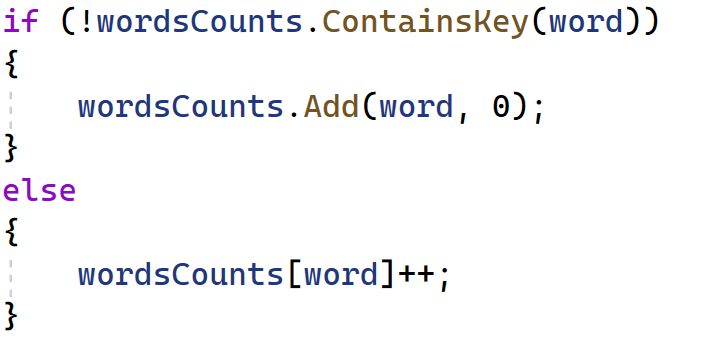
1. В него прочетете **всички** **думи** от файла:



1. След това обходете всички думи чрез **foreach-цикъл**:



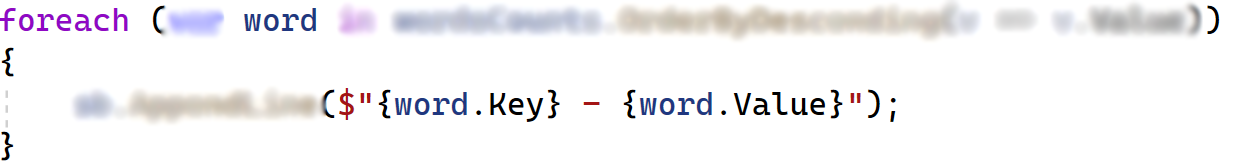
1. В него проверете дали думата **не е в речника**. Ако е така, **добавете думата** със стойност **0**. В противен случай, **увеличете** **стойността** на думата с **1**:



1. След **StreamReader-a** създайте **StringBuilder**, чрез който ще отпечатваме думите:



1. След това **добавете** всички думи в **sb**:



1. Накрая напишете целия текст във файла:



## Копиране на двоичен файл

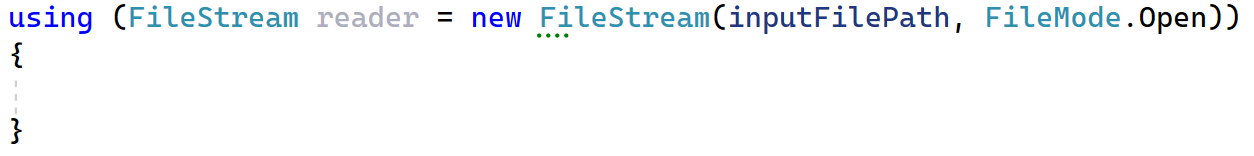
Напишете програма, която **копира** съдържанието от двоичен файл (например **copyMe.png**) до друг двоичен файл (например **copyMe-copy.png**) чрез **FileStream**. Не ви е позволено да използвате класа **File** или други подобни класове.

**Бележка:** използвайте следната структура:

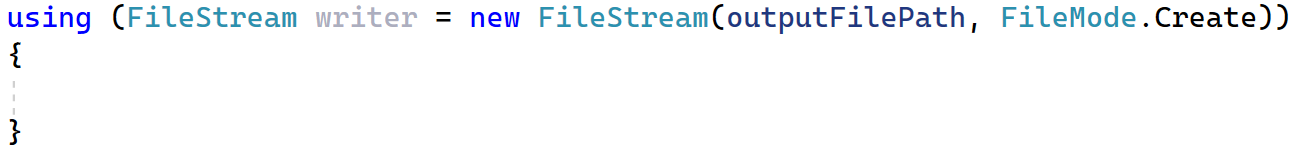
|  |
| --- |
| namespace CopyBinaryFile  {  public class CopyBinaryFile  {  static void Main()  {  string inputFilePath = @"..\..\..\Files\copyMe.png";  string outputFilePath = @"..\..\..\Files\copyMe-copy.png";  CopyFile(inputFilePath, outputFilePath);  }  public static void CopyFile(string inputFilePath, string outputFilePath)  {  }  }  } |

### Насоки

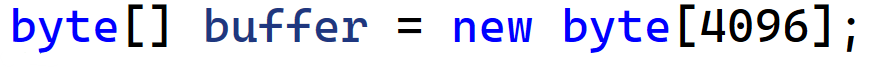
1. В метода **CopyFile(string inputFilePath, string outputFilePath)** създайте **FileStream** и подайте като аргументи променливата **outputFilePath** и **FileMode.Open**:



1. Във **FileStream-а** създайте още един **FileStream**, в който ще запишете информацията на файла. Подайте като аргументи променливата **inputFilePath** и **FileMode.Create**:



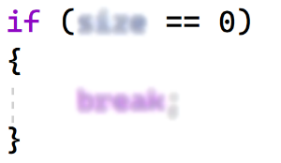
1. След това създайте **while-цикъл**.
2. В него създайте променливата **buffer**, чрез която ще **запазвате** **байтовете**, които се прочитат от **reader**:



1. Създайте **променлива**, в която ще пазите **размера на буфера**:



1. Създайте **проверка** дали **буфера има размер нула**. Ако е така, излезте от **while-цикъла**:



1. Накрая напишете съдържанието на **променливата** **buffer** във файла:



## Обхождане на директория

Напишете програма, която **преминава през дадена директория** за всички файлове с дадено **разширение**. Трябва да търси през **първото ниво** на директорията. Напишете информация за всеки файл в report.txt. Файлът трябва да бъде **запаметен на десктопа**.

Файловете трябва да бъдат **групирани по разширения**. Разширенията трябва да бъдат **сортирани** по броя на файловете в низходящ ред и по име във възходящ ред. Файловете на всяко разширение трябва да бъдат **сортирани** по техния размер. Уверете се, че пътят до директорията винаги ще бъде **валиден**.

**Бележка:** използвайте следната структура:

|  |
| --- |
| namespace DirectoryTraversal  {  public class DirectoryTraversal  {  static void Main()  {  string path = Console.ReadLine();  string reportFileName = @"\report.txt";  string reportContent = TraverseDirectory(path);    WriteReportToDesktop(reportContent, reportFileName);  }  public static string TraverseDirectory(string inputFolderPath)  {  }  public static void WriteReportToDesktop(string textContent, string reportFileName)  {  }  }  } |

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Директория** | **report.txt** |
| . | directory | .cs  --Mecanismo.cs - 0.994kb  --Program.cs - 1.108kb  --Nashmat.cs - 3.967kb  --Wedding.cs - 23.787kb  --Program - Copy.cs - 35.679kb  --Salimur.cs - 588.657kb  .txt  --backup.txt - 0.028kb  --log.txt - 6.72kb  .asm  --script.asm - 0.028kb  .config  --App.config - 0.187kb  .csproj  --01. Writing-To-Files.csproj - 2.57kb  .js  --controller.js - 1635.143kb  .php  --model.php - 0kb |

### Насоки

1. Създайте **StringBuilder**, чрез който ще отпечатвате файловете на директорията:



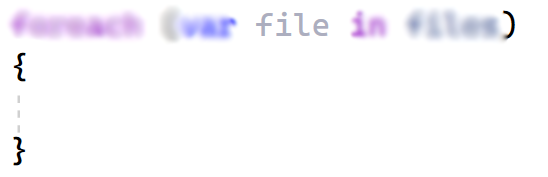
1. Вземете всички файлове чрез метода **Directory.GetFiles(inputFolderPath)**:



1. Създайте речник, в който ще пазите съдържанието на файловете:



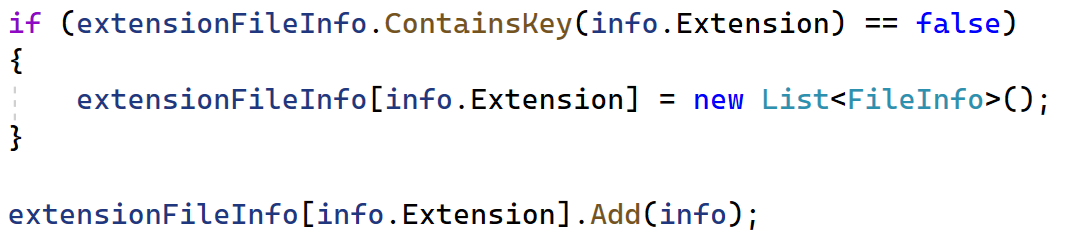
1. Създайте **foreach-цикъл**, който обхожда всички **файлове** на директорията:



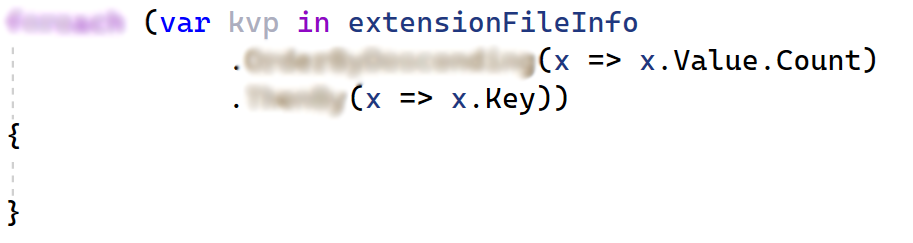
1. Създайте нова **променлива** от тип **FileInfo**, който **приема** като аргумент **file**:



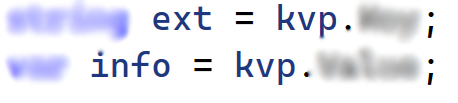
1. Създайте проверка, която проверя дали елементът, който отговаря на ключа **info.Extension**, съществува в **речника**. Ако това **не е вярно**, тогава създайте **нов елемент** в **речника** за този ключ, като му задайте **списък** с елемент като **стойност**. В противен случай, новият файл се добавя към **списъка**:



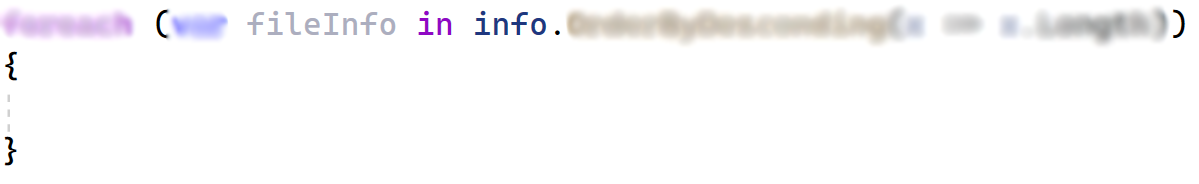
1. След него създайте още един **foreach-цикъл**, който обхожда **речника** като го сортирайте по **брой** **срещания** на файловите разширения в **низходящ ред** и след това по имената на разширенията във **възходящ ред**:



1. В него създайте две променливи за **ключа** и **стойността**:



1. След това добавете ключа в **sb**.
2. Създайте още един **foreach-цикъл**, който обхожда **всички** **файлове** на определено разширение по **големина** в **низходящ ред**:



1. Създайте променливи за **името на файла** и за **неговия размер**. Разделете **размера** на **файла** на **1024**, за да **конвертирате** размера в **килобайтове**.
2. След това добавете информацията на файла:



1. Накрая върнете стойността на **sb**.

## Копиране на съдържанието на директория

Напишете метод, който **копира директория с файлове** (без поддиректориите) в друга директория. Входната и изходната папка ще бъдат дадени като вход от конзолата. Ако изходната папка съществува, трябва да я изтриете (заедно със съдържанието в нея).

**Бележка:** използвайте следната структура:

|  |
| --- |
| namespace CopyDirectory  {  public class CopyDirectory  {  static void Main()  {  string inputFilePath = Console.ReadLine();  string outputFilePath = Console.ReadLine();  CopyAllFiles(inputFilePath, outputFilePath);  }  public static void CopyAllFiles(string inputPath, string outputPath)  {  }  }  } |

### Насоки

1. Проверете дали изходната **директория съществува**. Ако е така, я **изтрийте**.
2. Създайте **изходната** **директория**.
3. Използвайте метода **Directory.GetFiles()**, за да вземете имената на **файловете от директорията** и да ги съхраните в **списък**. Използвайте списъка, за да пренесете файловете в **изходната** **директория**.
4. Използвайте метода **File.Copy()**, за да копирате всеки файл.

## Зипване и извличане

Напишете програма, която създава **ZIP** файл (архив), като държи даден входен файл и извлича ZIP файла от архива в отделен изходен файл.

* Използвайте файла copyMe.png от ресурсите като входен файл в ZIP файл (например archive.zip)
* Извлечете файла от архива в нов файл по ваш избор (например extracted.png)

Ако вашият код е успешен, входният и изходният файл трябва да са еднакви.

**Бележка:** използвайте следната структура:

|  |
| --- |
| namespace ZipAndExtract  {  public class ZipAndExtract  {  static void Main()  {  string inputFilePath = @"..\..\..\Files\copyMe.png";  string zipArchiveFilePath = @"..\..\..\Files\archive.zip";  string extractedFilePath = @"..\..\..\Files\extracted.png";    ZipFileToArchive(inputFilePath, zipArchiveFilePath);  string fileNameOnly = Path.GetFileName(inputFilePath);  ExtractFileFromArchive(zipeArchiveFilePath, fileNameOnly, extractedFilePath);  }  public static void ZipFileToArchive(string inputFilePath, string zipArchiveFilePath)  {  }  public static void ExtractFileFromArchive(string zipArchiveFilePath, string fileName, string outputFilePath)  {  }  }  } |

### Насоки

* Използвайте класа **ZipFile**
* Записът в ZIP файла трябва да съдържа само името на файла без неговия път.

## Обединяване на текстови файлове

Напишете програма, която чете **съдържанието на два файла** (примерно input1.txt и input2.txt) и ги **съединява** в трети файл (примерно output.txt). Съединението трябва да бъде в следния формат:

* Ред 1 от input1.txt
* Ред 1 от input2.txt
* Ред 2 от input1.txt
* Ред 2 от input2.txt
* …

Ако някой от файловете има **повече редове** от другия, добавете **в края на изхода** редовете, които **не могат** да бъдат съпоставени с другия файл.

**Бележка**: използвайте следната структура:

|  |
| --- |
| namespace MergeFiles  {  public class MergeFiles  {  static void Main()  {  string firstInputFilePath = @"..\..\..\Files\input1.txt";  string secondInputFilePath = @"..\..\..\Files\input2.txt";  string outputFilePath = @"..\..\..\Files\output.txt";    MergeTextFiles(firstInputFilePath, secondInputFilePath, outputFilePath);  }  public static void MergeTextFiles(string firstInputFilePath, string secondInputFilePath, string outputFilePath)  {  // TODO: напишете вашия код…  }  }  } |

### Примери

|  |  |  |
| --- | --- | --- |
| **input1.txt** | **input2.txt** | **output.txt** |
| 1  3  5 | 2  4  6  7 | 1  2  3  4  5  6  7 |

### Насоки

1. Дефинирайте променливите **firstLine** и **secondLine**, които ще да пазят текущия ред, прочетен от съответния файл.
2. Използвайте конструкцията **using** за да създадете инстанции на **StreamReader**, които ще четат файловете с имена **firstInputFilePath** и **secondInputFilePath**.
3. Използвайте конструкцията **using** отново, за да създадете инстанция на **StreamWriter**, която да записва във файла **outputFilePath**.
4. Създайте **while-цикъл**, който ще итерира докато двата текста не свършат.

## Разделяне / Обединяване на двоични файлове

Даден ви е входен двоичен файл (примерно example.png). Напишете програма, която **разделя** файла на два с **еднакви** размери (примерно part-1.bin и part-2.bin). Когато размерът на входния файл е **нечетен**, първата част трябва да бъде с **1 байт** **повече** от втората.

След като разделите входния файл, съединете двата файла в **нов** (примерно example-joined.png). Полученият файл трябва да бъде същият като първоначалния.

**Бележка**: използвайте следната структура:

|  |
| --- |
| namespace SplitMergeBinaryFile  {  public class SplitMergeBinaryFile  {  static void Main()  {  string sourceFilePath = @"..\..\..\example.png";  string joinedFilePath = @"..\..\..\example-joined.png";  string partOnePath = @"..\..\..\part-1.bin";  string partTwoPath = @"..\..\..\part-2.bin";    SplitBinaryFile(sourceFilePath, partOnePath, partTwoPath);  MergeBinaryFiles(partOnePath, partTwoPath, joinedFilePath);  }  public static void SplitBinaryFile(string sourceFilePath, string partOneFilePath, string partTwoFilePath)  {  // TODO: напишете вашия код…  }  public static void MergeBinaryFiles(string partOneFilePath, string partTwoFilePath, string joinedFilePath)  {  // TODO: напишете вашия код…  }  }  } |

## Извличане на специални байтове

Даден ви е двоичен файл (примерно **example.png**) и текстов файл (примерно **bytes.txt**), съдържащ списък от битове в обхвата [0…255]. Напишете програма, която извлича всички **еднакви битове** в двата файла в изходен двоичен файл (примерно output.bin).

**Бележка:** използвайте следната структура:

|  |
| --- |
| namespace ExtractBytes  {  public class ExtractBytes  {  static void Main()  {  string binaryFilePath = @"..\..\..\example.png";  string bytesFilePath = @"..\..\..\bytes.txt";  string outputFilePath = @"..\..\..\output.bin";    ExtractBytesFromBinaryFile(binaryFilePath, bytesFilePath, outputFilePath);  }  public static void ExtractBytesFromBinaryFile(string binaryFilePath, string bytesFilePath, string outputFilePath)  {  // TODO: напишете вашия код…  }  }  } |

### Примери

|  |  |  |
| --- | --- | --- |
| **example.png** | **bytes.txt** | **output.bin** |
|  | 20  46  183  212 |  |