„Programming”
Big Project – Challenge Day

Made by: Bakhtishod Umurzokov
Neptun code: Y62UF3
E-mail: y62uf3@inf.elte.hu

Course code: IP-18fPROGEG
Teacher’s name: Zsuzsa Pluhár, Vincze Dorottya

2024. January 14.

Content

User documentation

Task

The Day of Challenge is one of the favorite social sport events of people since 1991. In Hungary, 1591 settlements have taken part in the challenge so far. Last year, the populations of the villages and towns taking part did 48 million minutes of sport during a single day. To apply for the challenge, the name of the settlement, the population, and the number of people willing to take part is needed. When processing the data, the settlements are categorized according to this: I. category: less than 700 people as population; II. category: 700-1499 people; III. category: 1500-2999 people; IV. category: 3000-7999 people; V. category: 8000-24999 people; VI. category: 25000-69999 people; VII. category: more than 70000 people.
Write a program that gives the number of participants for each category and the settlement which had the highest amount of participants.

Runtime environment

An IBM PC that is capable of running exe files, 32-bit operating system (eg. Windows 7). No mouse needed..

Usage

Starting the program

The program can be found in the archived file by the name `Y62UF3\Y62UF3\bin\Debug\net6.0\Y62UF3.exe`. You can start the program by clicking the `Y62UF3.exe` file.

Program input

The program reads the input data from the keyboard in the following order:

#	Data	Explanation
1.	*dataLength*	The count of settlements (1≤ *dataLength* ≤100).
2.	$settlement_1$	The name of the settlement.
3.	$population_1$	The population of the settlement (1≤P≤200 000).
4.	$participants_1$	The count of those taking part (1≤R≤10 000)
5.	$settlement_2$	The name of the settlement.
6.	$population_2$	The population of the settlement (1≤P≤200 000).
7.	$participants_2$	The count of those taking part (1≤R≤10 000)
...	...	
3*N+1.	$settlement_N$	The name of the settlement.
3*N+2	$population_N$	The population of the settlement (1≤P≤200 000).
3*N+3	$participants_N$	The count of those taking part (1≤R≤10 000)

Program output

The program writes out the count of participants for each category (in increasing order of categories) in the first line. In the second line it writes out the name of the settlement which had the highest amount of participants. If there are more than one solution, writes out one on the smallest index.

Sample input and output

```
Run    Y62UF3
"C:\Program Files\JetBrains\JetBrains Rider 2023.3.2\plugins\dpa\DotFiles\JetBrains.DPA.Runner.exe" --handle=9992 --backend-pid=13704 --etw-collect-flags=67108622 --detach-event-name=dpa.detach.9992 C:/Users/User/RiderProjects/Y62UF3/Y62UF3/bin/Debug/net6.0/Y62UF3.exe
4
Miskolc
166823
1234
Aszod
5772
234
Zirc
7454
333
Boly
3939
282
0 0 0 849 0 0 1234
Miskolc

Process finished with exit code 0.
Y62UF3 > Y62UF3 > C# Program.cs    12:1  CRLF  UTF-8  4 spaces
```

Possible errors

The input should be given according to the sample. If the count of settlements is not a whole number, or it is not in the range 1..100, it will cause a problem. If the name of settlement is not a string, or the population of settlement is not a number in the range 1..200000, or the count of participants is not a number in the range 1..10000, it also will cause a problem. In the case of an error, the program displays an error message, or asks for the repetition of the input.

Sample of running in the case of invalid data:

```
Run    Y62UF3
"C:\Program Files\JetBrains\JetBrains Rider 2023.3.2\plugins\dpa\DotFiles\JetBrains.DPA.Runner.exe" --handle=9192 --backend-pid=13704 --etw-collect-flags=67108622 --detach-event-name=dpa.detach.9192 C:/Users/User/RiderProjects/Y62UF3/Y62UF3/bin/Debug/net6.0/Y62UF3.exe
Hi
Unhandled exception. System.FormatException: Input string was not in a correct format.
   at System.Number.ThrowOverflowOrFormatException(ParsingStatus status, TypeCode type)
   at System.Convert.ToInt32(String value)
   at B3Assignment.Main(String[] args) in C:\Users\User\RiderProjects\Y62UF3\Y62UF3\Program.cs:line 7
Y62UF3 > Y62UF3 > C# Program.cs    12:1  CRLF  UTF-8  4 spaces
```

Developer documentation

Task

The Day of Challenge is one of the favorite social sport events of people since 1991. In Hungary, 1591 settlements have taken part in the challenge so far. Last year, the populations of the villages and towns taking part did 48 million minutes of sport during a single day. To apply for the challenge, the name of the settlement, the population, and the number of people willing to take part is needed. When processing the data, the settlements are categorized according to this: I. category: less than 700 people as population; II. category: 700-1499 people; III. category: 1500-2999 people; IV. category: 3000-7999 people; V. category: 8000-24999 people; VI. category: 25000-69999 people; VII. category: more than 70000 people.
Write a program that gives the number of participants for each category and the settlement which had the highest amount of participants.

Specification

Input: $\texttt{dataLength}\in\mathbb{N}$, $\texttt{settlements[1..dataLength]}\in T^{\texttt{dataLength}}$, $\texttt{populations[1..dataLength]}\in N^{\texttt{dataLength}}$, $\texttt{participants[1..dataLength]}\in N^{\texttt{dataLength}}$

Output: $\texttt{categories[1..7]}\in N$, $\texttt{highestParticipantsSettlement}\in T$

Precondition: $\texttt{dataLength}\in\texttt{[1..100]} \wedge \forall i\in\texttt{[1..dataLength]}:\texttt{populations[i]}\in\texttt{[1..200000]} \wedge \forall i\in\texttt{[1..dataLength]}:\texttt{participants[i]}\in\texttt{[1..10000]}$

Postcondition:

$\forall i\in\texttt{[1..dataLength]}:(\texttt{condition}):\texttt{categories[category]}=\texttt{categories[category]}+\texttt{participants[i]} \wedge$

$$\texttt{maxIndex} = \underset{i=1}{\overset{\texttt{dataLength}}{\texttt{MAXINDEX}}}(\texttt{participants[i]}) \wedge$$

$\texttt{highestParticipantsSettlement} = \texttt{settlements[maxIndex]}$

Definitions:

$\texttt{condition}=(\texttt{populations[i]}<700):\texttt{category}=1$

$\texttt{condition}=(\texttt{populations[i]}\leq 1499):\texttt{category}=2$

$\texttt{condition}=(\texttt{populations[i]}\leq 2999):\texttt{category}=3$

$\texttt{condition}=(\texttt{populations[i]}\leq 7999):\texttt{category}=4$

$\texttt{condition}=(\texttt{populations[i]}\leq 24999):\texttt{category}=5$

$\texttt{condition}=(\texttt{populations[i]}\leq 69999):\texttt{category}=6$

$\texttt{condition}=(\texttt{populations[i]}\geq 70000):\texttt{category}=7$

Developer environment

IBM PC, an operating system capable of running exe files (eg. Windows 7), .NET Framework or .NET Core SDK, Visual Studio.

Source code

All the sources can be found in the *Y62UF3* folder (after extraction). The folder structure used for development:

File	Explanation
Y62UF3\Y62UF3\bin\Debug\net6.0\Y62UF3.exe	Executable code
Y62UF3\Y62UF3\Program.cs	C# source code
Y62UF3\test1.txt	input test file$_1$
Y62UF3\test2.txt	input test file$_2$
Y62UF3\test3.txt	input test file$_3$
Y62UF3\test4.txt	input test file$_4$
Y62UF3\docs\Y62UF3.docx	documentation (this file)

Solution

Program parameters

Contants

```
MaxPopulation  : Integer(200000)   [the max number of population]
MaxParticipants  : Integer(10000)  [the max number of participants]
```

Types

```
Settlements = Array(1..:String)
Populations = Array(1..MaxPopulation:Integer)
Participants = Array(1..MaxParticipants:Integer)
```

Variables

```
dataLength     : Integer
settlements    : Settlements
populations    : Populations
participants   : Participants
```

The structure of the program

The modules used by the program, and their locations:

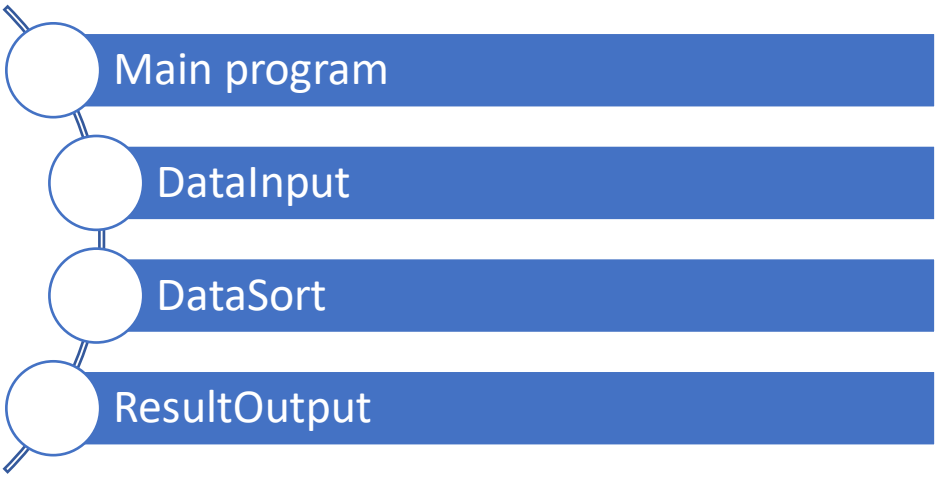
`Program.cs` – the program, in the source folder

`Namespaces` – code organizer, contains classes and other types

`Classes` – program organizer, contain methods, properties, fields, and other types

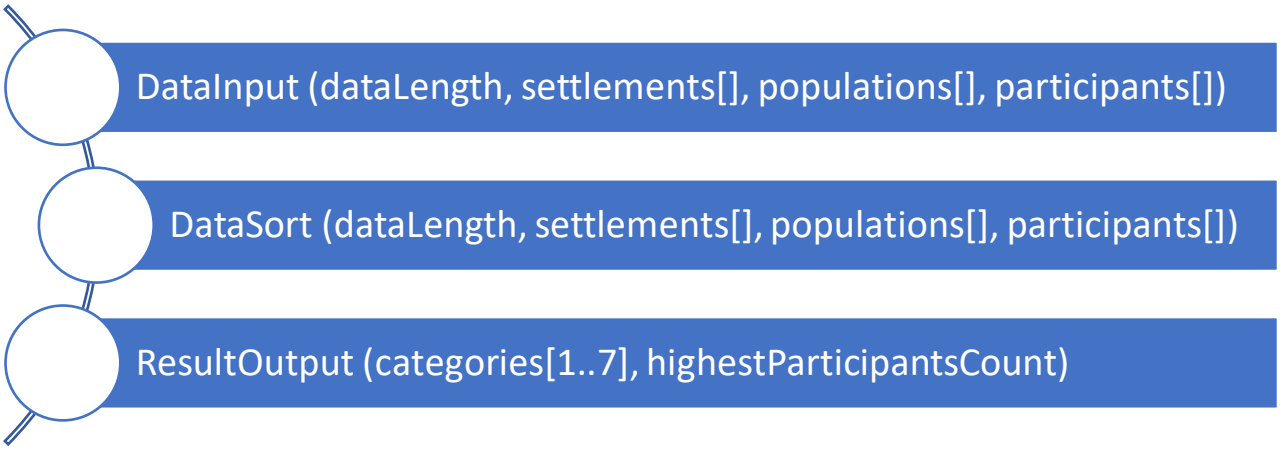
`Main method` – entry point of C# application

Structure of functions



The algorithm of the program

Main program:



Subprograms:

DataInput (dataLength, settlements[], populations[], participants[])	
In: dataLength [1<dataLength<100]	
In: settlements[1..dataLength]	
In: populations[1..dataLength][1≤populations[1..dataLength]≤200000]	
In: participants[1..dataLength][1≤participants[1..dataLength]≤10000]	

DataSort (dataLength, settlements[], populations[], participants[])	
categories[1..7] highestParticipantsSettlement:="" highestParticipantsCount:=0	

<table>
<tr><td rowspan="7">i=1..dataLength
category:=1</td><td colspan="3">T \ populations[i]<700 / F</td></tr>
<tr><td rowspan="2">category:=1</td><td colspan="2">T \ populations[i]≤1499 / F</td></tr>
<tr><td>category:=2</td><td>…..</td></tr>
<tr><td colspan="3">…..</td></tr>
<tr><td colspan="3">categories[category]=categories[category]+participants[i]</td></tr>
<tr><td colspan="3">T \ participants[i]>highestParticipantsCount / F</td></tr>
<tr><td colspan="3">highestParticipantsCount:=participants[i]
highestParticipantsSettlement:=settlements[i]</td></tr>
</table>

The code

The content of the `Program.cs` file:

```
/*
  Created by: Bakhtishod Umurzokov
  Neptun: Y62UF3
  E-mail: y62uf3@inf.elte.hu
  Task: „Big Project” - Challenge Day
*/
using System;

class B3Assignment
{
    static void Main(string[] args)
    {
//Input:
        int dataLength = Convert.ToInt32(Console.ReadLine());

        string[] settlements = new string[dataLength];
        int[] populations = new int[dataLength];
        int[] participants = new int[dataLength];

        int[] categories = new int[7];
        string highestParticipantsSettlement = "";
        int highestParticipantsCount = 0;

        for (int i = 0; i < dataLength; i++)
        {
            string settlementName = Console.ReadLine();
            settlements[i] = settlementName;
            int populationCount= Convert.ToInt32(Console.ReadLine());
            populations[i]=populationCount;
            int participantCount = Convert.ToInt32(Console.ReadLine());
            participants[i]=participantCount;

            int category=0;

            if (populations[i] < 700)
            {
                category = 0;
```

```
            }
            else if (populations[i] <= 1499)
            {
                category = 1;
            }
            else if (populations[i] <= 2999)
            {
                category = 2;
            }
            else if (populations[i] <= 7999)
            {
                category = 3;
            }
            else if (populations[i] <= 24999)
            {
                category = 4;
            }
            else if (populations[i] <= 69999)
            {
                category = 5;
            }
            else
            {
                category = 6;
            }

            categories[category] += participants[i];

            if (participants[i] > highestParticipantsCount)
            {
                highestParticipantsCount = participants[i];
                highestParticipantsSettlement = settlements[i];
            }
        }

//Output:

        for(int i = 0;i<7;i++)
        {
            if (i == 6)
            {
                Console.WriteLine($"{categories[i]}");
            }
            else
            {
                Console.Write($"{categories[i]} ");
            }
        }
        Console.WriteLine(highestParticipantsSettlement);
    }
}
```

Testing

Valid test cases

1. test case: test1.txt

Input
4 Miskolc 166823 1234 Aszod 5772 234 Zirc 7454 333 Boly 3939 282
Output
0 0 0 849 0 0 1234 Miskolc

2. test case: test2.txt

Input – *starts with continent, there are at least 2 islands*
3 Samarkand 500000 750 Tashkent 1000000 3000 Kashkadarya 800000 500
Output
0 0 0 0 0 0 4250 Tashkent
…

Invalid test cases

3. test case

Input – *wrong length*
0 …
Output
Asking again:

4. test case

Intput – wrong *height*
2 Kashkadarya 800000 110000
Output
Asking again:

…

8. test case

…

Further development options

1. Data to be read from file or keyboard
2. Editing the data inputs with id number of data
3. Capability to run multiple times after each other