„Programozási alapismeretek”
beadandó feladat

Készítette: Trefiman Viktor Ádám
Neptun-azonosító: DVZCBT
E-mail: dvzcbt@inf.elte.hu

Kurzuskód: IP-18PROGEG
Gyakorlatvezető neve: Csepregi-Horváth Zsófia Ágnes

2023. január 15.

Tartalom

Felhasználói dokumentáció

Feladat

Települések átlag szerinti sorrendje

A meteorológiai intézet az ország N településére adott M napos időjárás előrejelzést, az adott településen az adott napra várt legmagasabb hőmérsékletet.

Készíts programot, amely megadja a településeket átlaghőmérséklet szerint csökkenő sorrendben!

Futási környezet

IBM PC, exe futtatására alkalmas, 32-bites operációs rendszer (pl. Windows 7). Nem igényel egeret.

Használat

A program indítása

A program az `DVZCBT\bin\Release\DVZCBT.exe` néven található.

A program használata billentyűzetről való bevitel esetén

Az `DVZCBT.exe` fájl elindításával a program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

#	Adat	Magyarázat
1.	*Települések száma (n)*	1-től 1000-ig terjedő egész szám
2.	*Napok száma (m)*	1-től 1000-ig terjedő egész szám
3.	*1. település 1. nap*	-50-től 50-ig terjedő egész szám innentől
4.	*1. település 2. nap*	
…	…	
	2. település 1. nap	
	2. település 2. nap	
	…	
	n. település m. nap	

A program használata fájlból való bevitel esetén

Lehetőségünk van az adatokat **fájl**ban is megadni. Ekkor a programot *parancssorban* a következőképpen kell indítani, feltételezve, hogy a bemeneti fájlok mellette helyezkednek el:

```
DVZCBT.exe < be1.txt
```

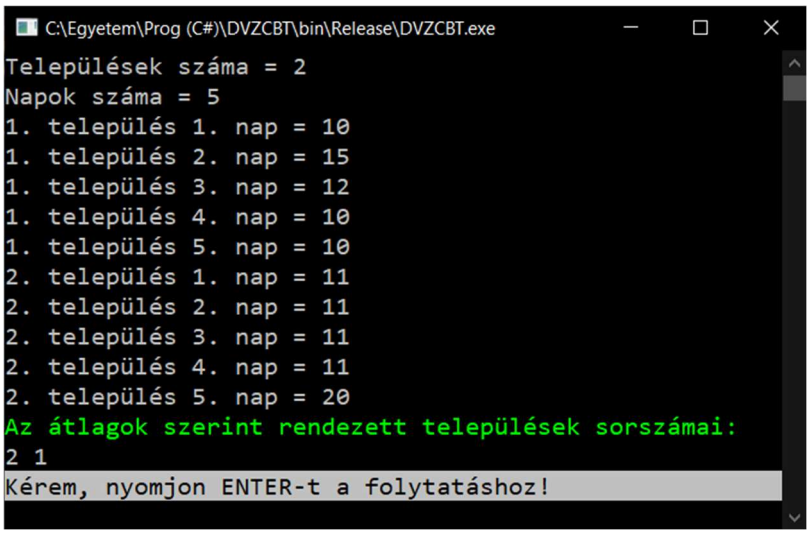
A fájl felépítésének a következő formai követelményei vannak. A fájl első sorában a települések száma (n) és a napok száma (m) van. A következő n sor mindegyikében m hőmérséklet szerepel, közülük az i-edik sorban a j-edik szám az i-edik település a j-edik sorszámú napon mért hőmérséklete.

Például:
3 5
10 15 12 10 10
11 11 11 11 20
12 16 16 16 20

A program kimenete

A program kiírja a települések sorszámát az átlaghőmérsékletek szerint csökkenő sorrendben. Az azonos átlaghőmérsékletű elemek növekvő sorrendben kerülnek kiírásra.

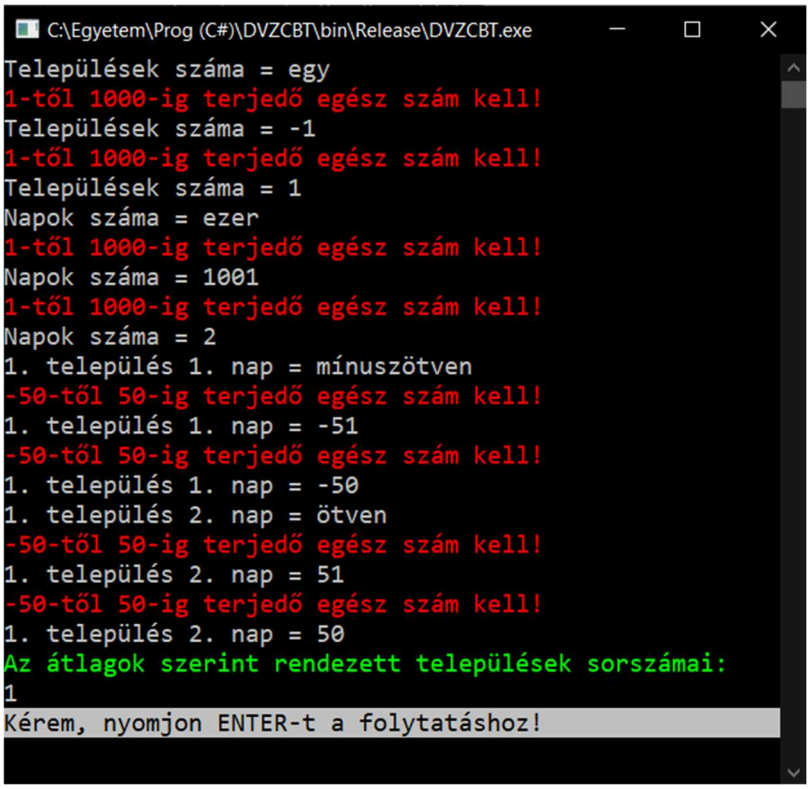
Minta bemenet és kimenet



Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha bármelyik megadandó adat nem egész szám és nincs benne a megadott intervallumban. Hiba esetén a program azzal jelzi a hibát, hogy újra kérdezi azt.

Mintafutás hibás bemeneti adatok esetén:



Fejlesztői dokumentáció

Feladat

Települések átlag szerinti sorrendje

A meteorológiai intézet az ország N településére adott M napos időjárás előrejelzést, az adott településen az adott napra várt legmagasabb hőmérsékletet.

Készíts programot, amely megadja a településeket átlaghőmérséklet szerint csökkenő sorrendben!

Tervezés

Specifikáció

Bemenet:	$N \in \mathbb{N}$, $M \in \mathbb{N}$, $falvak_{1..N,1..M} \in \mathbb{Z}^{N \times M}$
Kimenet:	$átlagok'_{1..N} \in Indexes^{N}$
Előfeltétel:	$1 \leq N \leq 1000$ és $1 \leq M \leq 1000$ és $\forall i \in [1..N]$ és $\forall j \in [1..M]: -50 \leq falvak_{i,j} \leq 50$
Utófeltétel:	$átlagok = (\overset{N}{\underset{i=1}{\forall}} Indexes(i, \sum_{j=1}^{M} falu_{i,j}))$ és $RendezettE_{\leq}(átlagok'.átlag)$ és $átlagok' \in Permutáció(átlagok)$
Definíció:	$Indexes = index \times átlag$, $index = \mathbb{N}$, $átlag = \mathbb{Z}$

Visszavezetés

Összegzés

N	~	*N*
M	~	*M*
X	~	*nap*
S	~	*összeg*

Algoritmus

```
Változók:
szum,rend,i,j,k,l: Egész
falvak: Tömb (1..N, 1..M: Egész)
Indexes: Rekord (index, átlag)
átlagok: Tömb (1..N: Indexes)
másol: Indexes
```

```
i = 1..N
    szum := 0
    j = 1..M
        szum += falvak[i][j]
    átlagok[i].index = i
    átlagok[i].átlag = szum
k := N - 1
k > 0
    rend := 0
    l = 1..k
        átlagok[l].átlag < átlagok[l + 1].átlag
        I                                        H
        másol := átlagok[l]                      -
        átlagok[l] := átlagok[l + 1]
        átlagok[l + 1] := másol
        rend := l
    k := rend
```

Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 10 Pro). Visual Studio 2022 (Version 17.3.6) fejlesztői környezet.

Forráskód

A teljes fejlesztői anyag –kicsomagolás után– az `DVZCBT` nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
`DVZCBT\bin\Release\DVZCBT.exe`	futtatható kód (a futtatáshoz szükséges fájlokkal)
`DVZCBT\obj\`	mappa fordításhoz szükséges kódokkal
`DVZCBT\Program.cs`	C# forráskód
`DVZCBT\Documets\be1.txt`	teszt-bemeneti fájl$_1$
`DVZCBT\Documets\be2.txt`	teszt-bemeneti fájl$_2$
`DVZCBT\Documets\be3.txt`	teszt-bemeneti fájl$_3$
`DVZCBT\Documets\be4.txt`	teszt-bemeneti fájl$_4$
`DVZCBT\Documets\be5.txt`	teszt-bemeneti fájl$_5$
`DVZCBT\Documets\Dokumentáció.docx`	dokumentációk (ez a fájl)

Megoldás

Programparaméterek

Típus

```
DataWithIndex = Rekord(index,avg:Egész)
```

Változó

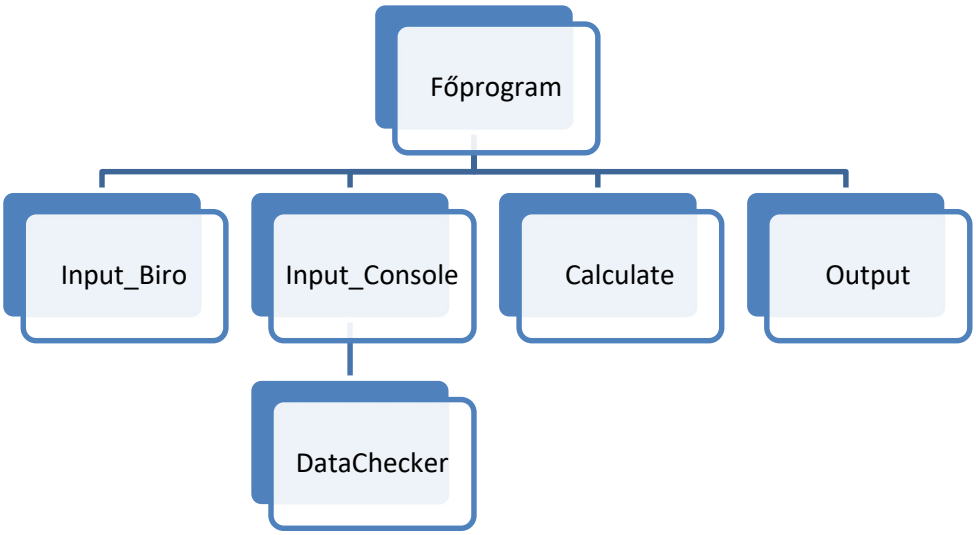
```
villages      : Tömb(1..n,1..m:Egész)
avgs          : Tömb(1..n: Egész)
```

Programfelépítés

A program által használt modulok (és helyük):

`Program.cs` – program, a forráskönyvtárban
`DVZCBT.sln` – program 'megoldás fájl', a forráskönyvtárban
`DVZCBT.csproj` – program 'projekt fájl', a forráskönyvtárban

Függvénystruktúra



A kód

A `Program.cs` fájl tartalma:

```
/*
Készítette: Trefiman Viktor Ádám
Neptun: DVZCBT
E-mail: dvzcbt@inf.elte.hu
Feladat: Települések átlag szerinti sorrendje
*/

using System;

namespace DVZCBT
{
    internal class Program
    {
        struct DataWithIndex
        {
            public int index;
            public int avg;
        }
        static void Main()
        {
            // Mátrix és méretei:
            int n = 0, m = 0;
            int[,] villages = null;

            // Input formájának eldöntése:
            if (Console.IsInputRedirected)
                Input_Biro(ref n, ref m, ref villages);
            else
                Input_Console(ref n, ref m, ref villages);

            // A feladat:
            DataWithIndex[] avgs = new DataWithIndex[n];
            Calculate(villages, n, m, ref avgs);

            // Kiírás:
            Output(avgs, n);
        }
        static void Input_Biro(ref int n, ref int m, ref int[,] villages)
        {
            // Méretek bekérése:
            string[] part = Console.ReadLine().Split();
            n = int.Parse(part[0]);
            m = int.Parse(part[1]);

            // Mátrix feltöltése:
```

```
            villages = new int[n, m];
            for (int i = 0; i < n; ++i)
            {
                part = Console.ReadLine().Split();
                for (int j = 0; j < m; ++j)
                    villages[i, j] = int.Parse(part[j]);
            }
        }
        static void DataChecker(string ask, ref int data, int min, int max)
        {
            // Adatot kér be és leellenőrzi hogy megfelel-e az intervallumnak:
            do
            {
                Console.ResetColor();
                Console.Write(ask);
                if (int.TryParse(Console.ReadLine(), out data) && data >= min && data <= max)
                    return;
                else
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.WriteLine($"{min}-től {max}-ig terjedő egész szám kell!");
                }
            } while (true);
        }
        static void Input_Console(ref int n, ref int m, ref int[,] villages)
        {
            // Méretek bekérése:
            DataChecker("Települések száma = ", ref n, 1, 1000);
            DataChecker("Napok száma = ", ref m, 1, 1000);

            // Mátrix feltöltése:
            villages = new int[n, m];
            for (int i = 0; i < n; ++i)
            {
                for (int j = 0; j < m; ++j)
                    DataChecker($"{i + 1}. település {j + 1}. nap = ", ref villages[i, j], -50, 50);
            }
        }
        static void Calculate(int[,] villages, int n, int m, ref DataWithIndex[] avgs)
        {
            // Hőmérsékletek összegzése településenként:
            for (int i = 0; i < n; ++i)
            {
                int sum = 0;
                for (int j = 0; j < m; ++j)
                    sum += villages[i, j];
                avgs[i] = new DataWithIndex { index = i + 1, avg = sum };
            }

            // Javított buborék-rendezés az összegekre:
            DataWithIndex temp;
            int k = n - 1, cut;
            while (k > 0)
            {
                cut = 0;
                for (int l = 0; l < k; ++l)
                {
                    if (avgs[l].avg < avgs[l + 1].avg)
                    {
                        temp = avgs[l];
                        avgs[l] = avgs[l + 1];
                        avgs[l + 1] = temp;
                        cut = l;
                    }
                }
                k = cut;
            }
        }
        static void Output(DataWithIndex[] avgs, int n)
        {
            // Az adatok kiírása csőből, illetve konzolról való beolvasás esetén:
            if (Console.IsInputRedirected)
            {
                for (int i = 0; i < n; ++i)
                    Console.Write($"{avgs[i].index} ");
```

```
                }
                else
                {
                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.WriteLine("Az átlagok szerint rendezett települések sorszámai:");
                    Console.ResetColor();

                    for (int i = 0; i < n; ++i)
                        Console.Write($"{avgs[i].index} ");

                    Console.ForegroundColor = ConsoleColor.Black;
                    Console.BackgroundColor = ConsoleColor.Gray;
                    Console.WriteLine("\nKérem, nyomjon ENTER-t a folytatáshoz!");
                    Console.ResetColor();
                    Console.ReadLine();
                }
            }
        }
    }
```

Tesztelés

Érvényes tesztesetek

1. teszteset: be1.txt

Bemenet – *1 település, 1 nap, 1 mérés*
1 1 1
Kimenet
1

2. teszteset: be2.txt

Bemenet – *3 település, 3 nap, 9 mérés*
3 3 3 3 3 2 2 2 1 1 1
Kimenet
1 2 3

3. teszteset: be3.txt

Bemenet – *3 település, 5 nap, 15 mérés*
3 5 10 15 12 10 10 11 11 11 11 20 12 16 16 16 20
Kimenet
3 2 1

4. teszteset: be4.txt

Bemenet – *10 település, 10 nap, 100 mérés*
10 10 -2 3 -4 6 8 5 2 -4 0 7 0 0 0 0 2 3 4 5 1 -1 -2 3 -4 6 8 5 2 -4 0 7 1 -4 3 4 -5 3 -1 7 4 6 0 2 1 3 13 8 4 19 12 22 -2 3 -4 6 8 5 2 -4 0 7 0 0 0 0 2 -3 4 5 1 -1 -2 3 -4 6 8 5 2 -4 0 7 1 4 3 4 5 3 1 7 4 6 0 -2 -1 3 13 8 4 19 12 22
Kimenet
5 10 9 1 3 6 8 4 2 7

5. teszteset: be5.txt

Bemenet – *1000 település, 1000 nap, 1000000 mérés*
Megtalálható: `DVZCBT\Documets\be5.txt`
Kimenet
Megtalálható: `DVZCBT\Documets\ki5.txt`

Érvénytelen tesztesetek

Billentyűzetes bevitel esetén

6. teszteset

Bemenet – *Szöveges adat*
Települések száma = tizenegy
Kimenet
Hibaüzenet és újrakérdezés: 1-től 1000-ig terjedő egész szám kell! Települések száma =

7. teszteset

Bemenet – *Értelmezési tartományon kívül eső szám*
Települések száma = 1 Napok száma = -1
Kimenet
Hibaüzenet és újrakérdezés: 1-től 1000-ig terjedő egész szám kell! Napok száma =

8. teszteset

Bemenet – *Értelmezési tartományon kívül eső szám*
Települések száma = 1 Napok száma = 1 1. település 1. nap = -51
Kimenet
Hibaüzenet és újrakérdezés: -50-től 50-ig terjedő egész szám kell! 1. település 1. nap =

Fejlesztési lehetőségek

1. Többszöri futtatás megszervezése
2. Települések nevének megadása
3. A napok pontos dátumának megadása
4. Grafikus visszajelzés a számolás lépéseiről