**Programozási alapismeretek**

**Komplex beadandó feladat dokumentációja**

**„Legszélsőségesebb települések”**

**Készítette:**

**Név: Kovács Martin  
Neptunkód: UBWRVY  
szakirány: Programtervező informatikus [fejlesztő]  
E-mail: kovacs.martin1011@gmail.com**

**Kurzuskód: SEK-1**

**Gyakorlatvezető: Horváth Zsófi**

**Dátum**

**Felhasználói dokumentáció**

**Feladat**

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 legszélsőségesebb településeket, azaz azokat, ahol a legkisebb és a legnagyobb várt hőmérséklet eltérése maximális!

**Bemenet**

A standard bemenet első sorában a települések száma (1≤N≤1000) és a napok száma

(1≤M≤1000) van. Az ezt követő N sorban az egyes napokra jósolt M hőmérséklet értéke található (-50≤Hi,j≤50).

**Kimenet**

A standard kimenet első sorába a legszélsőségesebb települések T számát kell kiírni! Ezt kövesse ezen települések sorszáma, növekvő sorrendben!

**Példa**

Bemenet Kimenet

3 5 2 1 2

10 15 12 6 10

11 11 11 11 20

12 16 16 16 20

**Korlátok**

Időlimit: 0.1 mp.

Memórialimit: 32 MB

**Futási környezet**

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

**Használat**

**A program indítása**

A program a Komplex\_beadando\Komplex\_beadando\bin\Debug\Komplex\_beadando.exe néven található a tömörített állományban. A Komplex\_beadando.exe fájl kiválasztásával indítható.

**A program bemenete**

A program az adatokat a billentyűzetről olvassa be a következő sorrendben:

|  |  |  |
| --- | --- | --- |
| **#** | **Adat** | **Magyarázat** |
| **1.** | tel | A települések száma (1≤tel≤1000) |
| **2.** | nap | A napok száma (1≤nap≤1000) |

Ezután ciklikusan (tel) soronként a következőket:

|  |  |  |
| --- | --- | --- |
| **#** | **Adat** | **Magyarázat** |
| **1.** | h[i,0] | Az első napra várt hőmérséklet (-50≤hi,j≤50) |
| **2.** | h[i,1] | A második napra várt hőmérséklet (-50≤hi,j≤50) |
| **…** | … | … |
| **ndb.** | h[i,ndb-1] | Az utolsó napra várt hőmérséklet (-50≤hi,j≤50) |

**A program kimenete**

A standard kimenet sorában a legszélsőségesebb települések T száma szerepel, mellette a települések sorszámai következnek, növekvő sorrendben.

**Hibalehetőségek**

Az egyes bemeneti adatokat a fenti mintának megfelelően (2. oldal) kell megadni. Hiba, ha a bemenetek nem egész számok, valamint nem a megadott alsó és felső határ között van valamelyik érték.

**Fejlesztői dokumentáció**

**Feladat**

Lásd: 2. oldal

**Specifikáció**

**Bemenet**:

*tel € N,*

*nap € N,*

*hom € R[1..tel, 1..nap]*

**Kimenet:**

*telList € N,*

*telListDb € N*

**Függvény:**

*maxKulF: N, N, N -> N*

*maxKulF(hom[,], tel, nap) =∀i ∈ [1..tel]: (∀j ∈ [1..nap]:*

*(MAX(hom[i,j]) - MIN(hom[i,j]))*

**Előfeltétel:**

*1 <= tel <= 1000 ÉS 1<= nap <= 1000 ÉS*

*∀i ∈ [1..tel]: (∀j ∈ [1..nap]: (-50 <= hom[i,j] <= 50))*

**Utófeltétel:**

*maxKul = ∀i ∈* [1..tel]: MAX(maxKulF(hom,tel,nap))

telList= KIVALOGAT(i=1..tel, maxKulF(hom,tel,nap)[i] == maxKul)

telListDb= DARAB(i=1..tel, telList)

**Fejlesztői környezet**

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 11). Visual Studio 2022 (Version 17.5.4) fejlesztői környezet.

**Megoldás**

**Programparaméterek**

***Típus***

hom kétdimenziós tömb (1..tel, 1..nap, egész)

maxKulT tömb (1..tel, egész)

telList lista (egész)

***Változók***

tel egész[1; 1000], települések száma

nap egész[1; 1000], napok száma

maxKul egész, a maximum különbségek közül a legnagyobb különbség eltárolására használatos

minHom egész, adott településen mért minimális hőmérséklet

maxHom egész, adott településen mért maximális hőmérséklet

telListDb egész, a telList listában lévő számok eltárolására

**Programfelépítés**

A program által használt modulok (és helyük):

Program.cs program, a forráskönyvtárban

Komplex\_beadando.sln program ’megoldás fájl’, a forráskönyvtárban

Komplex\_beadando.csproj program ’projekt fájl’, a forráskönyvtárban

**Függvénystruktúra**



**A teljes program algoritmusa**



**A kód**

namespace Komplex\_beadando

{

internal class Program

{

Program()

{

string[] n = Console.ReadLine().Split(' ');

int tel = int.Parse(n[0]);

int nap = int.Parse(n[1]);

if(tel < 1 || tel > 1000 || nap < 1 || nap > 1000)

{

Environment.Exit(0);

}

int[,] hom = new int[tel, nap];

for (int i = 0; i < tel; i++)

{

string[] sorok = Console.ReadLine().Split(' ');

for (int j = 0; j < nap; j++)

{

hom[i, j] = int.Parse(sorok[j]);

if (int.Parse(sorok[j]) > 50 || int.Parse(sorok[j]) < -50)

{

Environment.Exit(0);

}

}

}

int maxKul = maxKulF(hom, tel, nap).Max();

List<int> telList = new List<int>();

for(int i = 0; i < tel; i++)

{

if (maxKulF(hom, tel, nap)[i] == maxKul)

{

telList.Add(i);

}

}

int telListDb = telList.Count;

Console.Write(telListDb);

foreach(int e in telList)

{

Console.Write(" " + string.Join(" ", e + 1));

}

}

public static int[] maxKulF(int[,] hom, int tel, int nap)

{

int[] maxKulT = new int[tel];

for (int i = 0; i < tel; i++)

{

int maxHom = int.MinValue;

int minHom = int.MaxValue;

for (int j = 0; j < nap; j++)

{

maxHom = Math.Max(maxHom, hom[i, j]);

minHom = Math.Min(minHom, hom[i, j]);

}

maxKulT[i] = maxHom - minHom;

}

return maxKulT;

}

static void Main(string[] args)

{

new Program();

Console.ReadKey();

}

}

}

**Tesztelés**

**Érvényes tesztesetek**



**Fejlesztési lehetőségek:**

1. Optimalizálni a programot, futási idő miatt