

## **„Programozási alapismeretek” beadandó feladat**

*Készítette: Hegedűs Joshua  
Neptun-azonosító: YQMHWO  
E-mail: jhegedsus9@gmail.com*

*Kurzuskód: **IP-18PROGEG**  
Gyakorlatvezető neve: Purity Tamás Gábor*

***2023. január 11.***

## Tartalom

Felhasználói dokumentáció.....	3
Feladat.....	3
Futási környezet .....	3
Használat.....	3
A program indítása .....	3
A program bemenete .....	3
A program kimenete.....	3
Minta bemenet és kimenet .....	4
Hibalehetőségek .....	4
Fejlesztői dokumentáció .....	5
Feladat.....	5
Specifikáció.....	5
Fejlesztői környezet .....	6
Forráskód .....	6
Megoldás.....	7
Programparaméterek .....	7
Programfelépítés .....	7
Függvénystruktúra .....	7
Algoritmus .....	<b>Error! Bookmark not defined.</b>
A kód.....	8
Tesztelés .....	14
Érvényes tesztesetek .....	14
Érvénytelen tesztesetek .....	15
Fejlesztési lehetőségek.....	16

# Felhasználói dokumentáció

## Feladat

### Legváltozóbb települések

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

Készíts programot, amely megadja azokat a településeket, ahol az előrejelzés szerint egyik napról a másikra a lehető legnagyobb a változás!

## Futási környezet

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

Linux PC, `yqmhwo` futtatására alkalmas. Nem igényel egeret.

## Használat

### A program indítása

A program a `bin\yqmhwo.exe` illetve `bin\yqmhwo` néven található a tömörített állományban.

### A program használata billentyűzetről való bevétel esetén

Az `yqmhwo.exe` fájl elindításával a program az adatokat a **billentyűzetről** olvassa be a következő sorrendben:

#	Adat	Magyarázat
1.	<i>Települések száma = (n)</i>	1 és 1000 közti egész szám
2.	<i>Napok száma = (m)</i>	1 és 1000 közti egész szám
3.	<i>1. város 1. mérése =</i>	-50 és 50 közti egész szám
4.	<i>1. város 2. mérése =</i>	-50 és 50 közti egész szám
...	<i>1.város m. mérése =</i>	-50 és 50 közti egész szám
...	...	
...	<i>2. város 1. mérése =</i>	-50 és 50 közti egész szám
	<i>2. város 2. mérése =</i>	-50 és 50 közti egész szám
...	<i>2. város m. mérése =</i>	-50 és 50 közti egész szám
...	...	
...	<i>n. város az m. mérése =</i>	-50 és 50 közti egész szám

### A program használata fájlból való bevétel esetén

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

```
.\yqmhwo.exe bel.txt
```

Illetve linux operációs rendszeren:

```
./yqmhwo bel.txt
```

A fájl felépítésének a következő formai követelményei vannak. A fájl első sorában a települések

száma (n) és a mérések száma (m) van. A következő n sor mindegyikében m darabszám szerepel, közülük az i-edik sorban a j-edik szám az i-edik helységeben a j-edik sorszámu fajból megfigyelt mérések száma. Például:

```
3 5
10 15 12 10 10
11 11 11 11 20
25 16 16 16 20
```

### A program kimenete

A program kiírja azoknak a városoknak a darabszámát és utána azok sorszámain, ahol a legmagasabb különbség lett mérve.

### Minta bemenet és kimenet

```
[START AT]: 3:57:05
[  HOST]: HEGEDUS-PC
[  USER]: JoshH
[PLATFORM]: Win32NT

----- CREATOR -----
[ MADE BY]: YQMHW (Joshua Hegedus)
[ GITHUB]: https://github.com/joshika39/

----- EXECUTION -----

[ INPUT: 15:57:05] Number of cities = 2
[ INPUT: 15:57:06] Number of the temperatures in a city = 3
-----

[ INPUT: 15:57:07] City 1: Temperature 1 = 2
[ INPUT: 15:57:08] City 1: Temperature 2 = 4
[ INPUT: 15:57:09] City 1: Temperature 3 = 6
-----

[ INPUT: 15:57:10] City 2: Temperature 1 = 5
[ INPUT: 15:57:12] City 2: Temperature 2 = 2
[ INPUT: 15:57:13] City 2: Temperature 3 = 3
-----

[SUCCESS: 15:57:14] 1 2
[ INFO: 15:57:14] Press any key to continue...
```

### Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba esetén a program azzal jelzi a hibát, hogy újra kérdezi azt.

### Minta futás hibás bemeneti adatok esetén:

```
----- EXECUTION -----

[ INPUT: 21:25:38] Number of cities = 1
[ INPUT: 21:25:44] Number of the temperatures in a city = 0
[ ERROR: 21:25:46] Value is not in the range! (0, range: 1 -> 1000)
[ INPUT: 21:25:46] Number of the temperatures in a city = 3
-----

[ INPUT: 21:25:50] City 1: Temperature 1 = -123
[ ERROR: 21:25:53] Value is not in the range! (-123, range: -50 -> 50)
[ INPUT: 21:25:53] City 1: Temperature 1 = 23
[ INPUT: 21:25:56] City 1: Temperature 2 = 456
[ ERROR: 21:26:01] Value is not in the range! (456, range: -50 -> 50)
[ INPUT: 21:26:01] City 1: Temperature 2 = 50
[ INPUT: 21:26:05] City 1: Temperature 3 = 51
[ ERROR: 21:26:10] Value is not in the range! (51, range: -50 -> 50)
[ INPUT: 21:26:10] City 1: Temperature 3 = 4
-----

[SUCCESS: 21:26:13] 1 1
[ INFO: 21:26:13] Press any key to continue...
```

# Fejlesztői dokumentáció

## Feladat

### Legváltozóbb települések

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

Készíts programot, amely megadja azokat a településeket, ahol az előrejelzés szerint egyik napról a másikra a lehető legnagyobb a változás!

## Tervezés

### Specifikáció

Be:  $n, m \in \mathbb{N}, cities_{1..n, 1..m} \in \mathbb{N}^{n \times m}$

Ki:  $db \in \mathbb{N}, city_{1..db} \in \mathbb{N}^{db}$

Ef:  $N \in \mathbb{N}, 1 \leq N \leq 1000$

$M \in \mathbb{N}, 1 \leq M \leq 1000$

$-50 \leq Ho[M]_i \leq 50$

Uf:  $(db, helység) = Kiválogat_{i=1}^n i \text{ és } j\acute{o}(i)$

$db: \mathbb{N}$

$db = \forall_{j=1}^m (cities_{i,j} = \maxDiff)$

### ViSSzavezetés

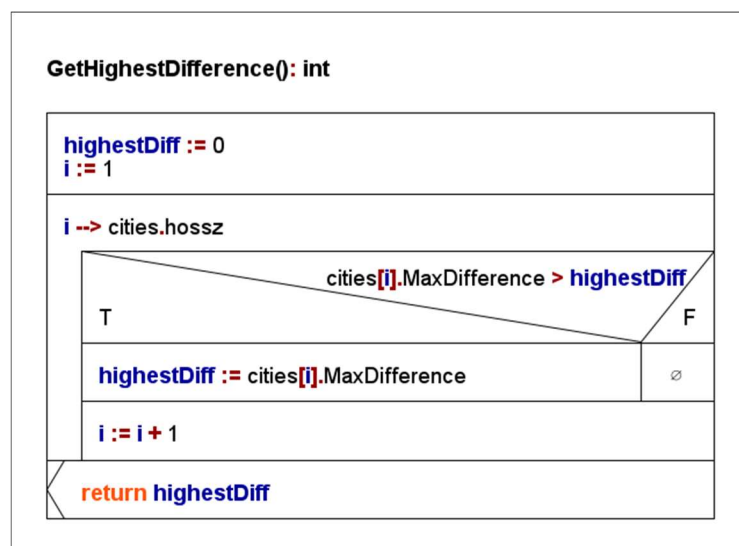
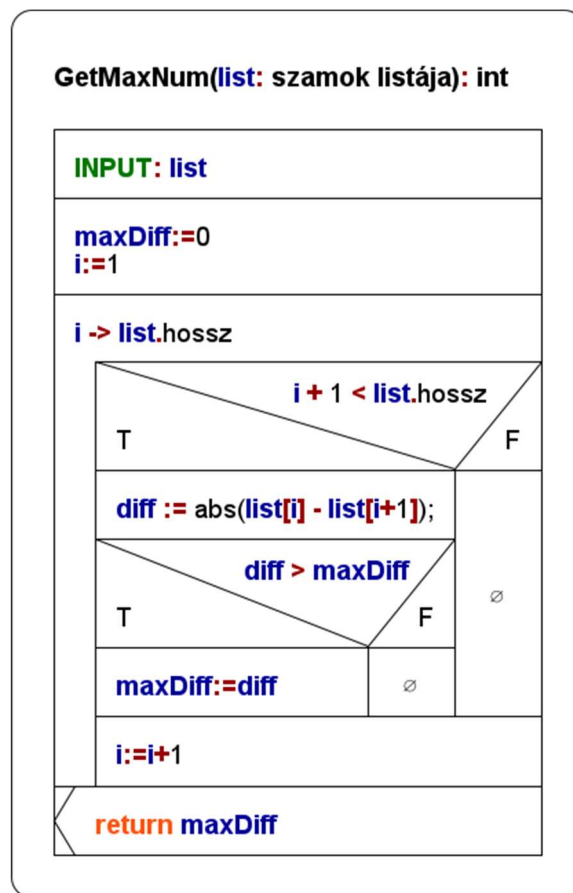
*Kiválogatás*

$y \sim city$   
 $T(x_i) \sim cities_{i,j} = \maxDiff$

*Eldöntés*

$i \sim k$   
 $T(x_i) \sim i \neq k \text{ és } cities_{k,j} =$

## Algoritmus



## Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 10 Pro). Visual Studio 2022 (Version 17.2.3) fejlesztői környezet.

## Forráskód

A teljes fejlesztői anyag –kicsomagolás után– az YQMHW0 nevű könyvtárban található meg. A fej-

lesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
YQMHW0\x64\yqmhwo.exe	Windows-on futtatható kód (a futtatáshoz szükséges fájlokkal)
YQMHW0\unix\yqmhwo	Linux-on futtatható kód
YQMHW0\src\Program.cs	C# forráskód
YQMHW0\src\yqmhwo.csproj	Projectfájl
YQMHW0\Testcases\be1.txt	teszt-bemeneti fájl <sub>1</sub>
YQMHW0\Testcases\be2.txt	teszt-bemeneti fájl <sub>2</sub>
YQMHW0\Testcases\be3.txt	teszt-bemeneti fájl <sub>3</sub>
YQMHW0\Testcases\be4.txt	teszt-bemeneti fájl <sub>4</sub>
YQMHW0\YQMHW0.docx	dokumentációk (ez a fájl)

## Megoldás

### Programparaméterek

#### Típus

ICity = **City** (bal, jobb: **Egész**)  
 ICities = **Cities** ()

#### Változó

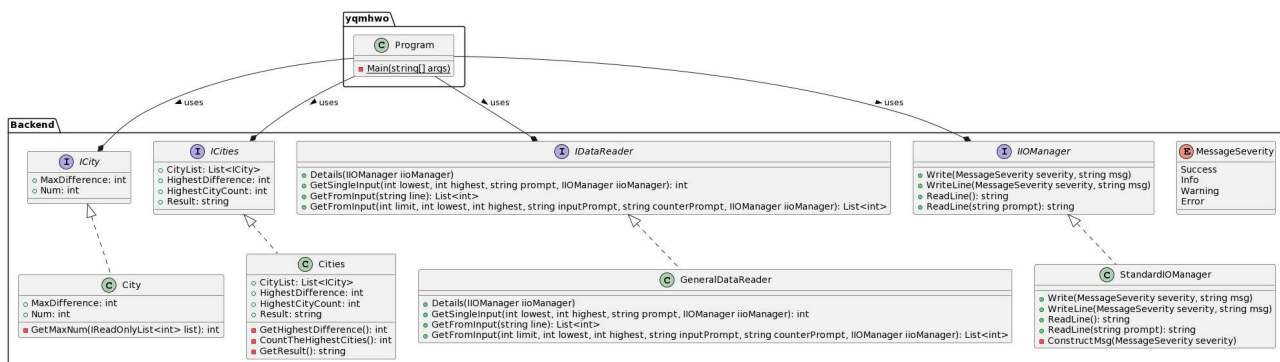
cities : **Cities** (1..n: **City**)

### Programfelépítés

A program által használt modulok (és helyük):

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

### Függvénystruktúra



Link a jobb rezolúcióhoz: [Kép](#), [Forrás](#)

## A kód

A Program.cs fájl tartalma:

```
// #define BIRO

using System;
using System.Collections.Generic;
using System.IO;
using Backend;

/*
    Készítette: Joshua Hegedus
    Neptun: YQMHW0
    E-mail: jhegedus9@gmail.com
    Feladat: Legváltozóbb települések
*/

// ReSharper disable All

namespace Backend
{
    public enum MessageSeverity
    {
        Success,
        Info,
        Warning,
        Error
    }

    #region Classes
    internal class StandardIOManager : IIOManager
    {
        public StandardIOManager()
        {
            Console.ForegroundColor = ConsoleColor.White;
        }

        public void Write(MessageSeverity severity, string msg)
        {
            #if BIRO
            Console.Write(msg);
            #else
            ConstructMsg(severity);
            Console.Write(msg);
            #endif
        }

        public void WriteLine(MessageSeverity severity, string msg)
        {
            #if BIRO
            Console.WriteLine(msg);
            #else
            ConstructMsg(severity);
            Console.WriteLine(msg);
            #endif
        }
    }
}
```



```

    #endif
}

public string ReadLine()
{
    return Console.ReadLine();
}

public string ReadLine(string prompt)
{
    #if BIRO
    return Console.ReadLine();
    #else
    var time = DateTime.Now.ToString("HH:mm:ss");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.Write($"[ INPUT: {time}] {prompt}");
    var ans = Console.ReadLine();
    Console.ForegroundColor = ConsoleColor.White;
    return ans;
    #endif
}

private void ConstructMsg(MessageSeverity severity)
{
    var time = DateTime.Now.ToString("HH:mm:ss");
    switch (severity)
    {
        case MessageSeverity.Success:
            Console.ForegroundColor = ConsoleColor.Green;
            Console.Write($"[SUCCESS: {time}] ");
            break;

        case MessageSeverity.Info:
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write($"[ INFO: {time}] ");
            break;

        case MessageSeverity.Warning:
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.Write($"[WARNING: {time}] ");
            break;

        case MessageSeverity.Error:
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Error.Write($"[ ERROR: {time}] ");
            break;
        default:
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Error.Write($"[ ERROR: {time}] Unknown Severity!{¥n}");
            Console.ForegroundColor = ConsoleColor.White;
            throw new ArgumentException("Unknown Severity!");
    }
    Console.ForegroundColor = ConsoleColor.White;
}

```

```

}

internal class GeneralDataReader : IDataReader
{
    public void Details(IIOManager iioManager)
    {
        var startedTime = DateTime.Now.ToString("h:mm:ss");
        var host = Environment.MachineName;
        var user = Environment.UserName;
        var platform = Environment.OSVersion.Platform;

        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine(" ----- HOST PC ----- ");
        Console.WriteLine($"[START AT]: {startedTime}");
        Console.WriteLine($"[  HOST]: {host}");
        Console.WriteLine($"[  USER]: {user}");
        Console.WriteLine($"[PLATFORM]: {platform}¥n");
        Console.WriteLine(" ----- CREATOR ----- ");
        Console.WriteLine($"[ MADE BY]: YQMHWQ (Joshua Hegedus)");
        Console.WriteLine($"[ GITHUB]: https://github.com/joshika39/¥n");
        Console.WriteLine(" ----- EXECUTION ----- ¥n");
        Console.ForegroundColor = ConsoleColor.White;
    }

    public int GetSingleInput(int lowest, int highest, string prompt, IIOManager iioManager)
    {
        var num = 0;
        var isCorrect = false;
        while (!isCorrect)
        {
            var numStr = iioManager.ReadLine(prompt);

            isCorrect = int.TryParse(numStr, out num);
            if (!isCorrect)
            {
                iioManager.WriteLine(MessageSeverity.Error, $"Incorrect format of values! ({numStr})");
            }
            else if (!(num >= lowest && num <= highest))
            {
                iioManager.WriteLine(MessageSeverity.Error, $"Value is not in the range! ({num}, range: {lowest} -> {highest})");
                isCorrect = false;
            }
        }

        return num;
    }

    public List<int> GetFromInput(string line)
    {
        var details = new List<int>();
        foreach (var text in line.Split(' ')) details.Add(int.Parse(text));
        return details;
    }
}

```

```

    }

    public List<int> GetFromInput(int limit, int lowest, int highest, string inputPrompt, string
counterPrompt, IIoManager iioManager)
    {
        var count = 0;
        var details = new List<int>();
        while (count != limit)
        {
            details.Add(GetSingleInput(lowest, highest, $"{inputPrompt}: {counterPrompt}
{count + 1} = ", iioManager));
            count++;
        }
        Console.WriteLine("-----¥n");
        return details;
    }
}

internal class City : ICity
{
    public City(int limit, int cityNum, IDataReader reader, IIoManager iioManager)
    {
        Num = cityNum;
        var measurements = reader.GetFromInput(limit, -50, 50, $"City {cityNum}", "Temperature",
iioManager);
        MaxDifference = GetMaxNum(measurements);
    }

    public City(int cityNum, string line, IDataReader reader)
    {
        Num = cityNum;
        var measurements = reader.GetFromInput(line);
        MaxDifference = GetMaxNum(measurements);
    }

    public int MaxDifference { get; set; }
    public int Num { get; set; }

    private int GetMaxNum(IReadOnlyList<int> list)
    {
        var maxDiff = 0;
        for (var i = 0; i < list.Count; i++)
            if (i + 1 < list.Count)
            {
                var diff = Math.Abs(list[i] - list[i + 1]);
                if (diff > maxDiff) maxDiff = diff;
            }

        return maxDiff;
    }
}

internal class Cities : ICities
{

```

```

public Cities(int cityCount, int measureCount, IDataReader reader, IIOManager iioManager)
{
    CityList = new List<ICity>();
    for (var i = 0; i < cityCount; i++)
    {
        #if BIRO
        var city = new City(i + 1, Console.ReadLine(), reader);
        #else
        var city = new City(measureCount, i + 1, reader, iioManager);
        #endif
        CityList.Add(city);
    }
}

public Cities(IDataReader reader, IReadOnlyList<string> lines)
{
    CityList = new List<ICity>();
    for (var i = 1; i < lines.Count; i++)
    {
        var city = new City(i, lines[i], reader);
        CityList.Add(city);
    }
}

public List<ICity> CityList { get; }
public int HighestDifference => GetHighestDifference();
public int HighestCityCount => CountTheHighestCities();
public string Result => GetResult();

private int GetHighestDifference()
{
    if (CityList == null) return -1;

    var highestDiff = 0;
    for (var i = 0; i < CityList.Count; i++)
        if (CityList[i].MaxDifference > highestDiff)
            highestDiff = CityList[i].MaxDifference;
    return highestDiff;
}

private int CountTheHighestCities()
{
    if (CityList == null) return -1;
    var highestCities = 0;
    foreach (var city in CityList)
        if (city.MaxDifference == HighestDifference)
            highestCities++;
    return highestCities;
}

private string GetResult()
{
    var res = $"[HighestCityCount]";
    for (var i = 0; i < CityList.Count; i++)

```

```

        if (i < CityList.Count && CityList[i].MaxDifference == HighestDifference)
            res += $" {CityList[i].Num}";

        return res;
    }
}

#endregion

#region Interfaces
public interface IDataReader
{
    void Details(IIOManager iioManager);
    int GetSingleInput(int lowest, int highest, string prompt, IIOManager iioManager);
    List<int> GetFromInput(string line);
    List<int> GetFromInput(int limit, int lowest, int highest, string inputPrompt, string
counterPrompt, IIOManager iioManager);
}

public interface ICity
{
    int MaxDifference { get; }
    int Num { get; }
}

public interface ICities
{
    List<ICity> CityList { get; }
    int HighestDifference { get; }
    int HighestCityCount { get; }
    string Result { get; }
}

public interface IIOManager
{
    void Write(MessageSeverity severity, string msg);
    void WriteLine(MessageSeverity severity, string msg);

    string ReadLine();
    string ReadLine(string prompt);
}

#endregion
}

namespace yqmhwo
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            var reader = new GeneralIDataReader();
            var outputManager = new StandardIOManager();

```

```

if (args.Length > 0)
{
    reader.Details(outputManager);
    outputManager.WriteLine(MessageSeverity.Info, $"Reading from file: {args[0]}");
    var lines = File.ReadAllLines(args[0]);
    ICities cities = new Cities(reader, lines);
    outputManager.WriteLine(MessageSeverity.Success, cities.Result);
}
else
{
    # if BIRO
    var details = reader.GetFromInput(Console.ReadLine());
    #else
    reader.Details(outputManager);
    var details = new List<int>
    {
        reader.GetSingleInput(1, 1000, "Number of cities = ", outputManager),
        reader.GetSingleInput(1, 1000, "Number of the temperatures in a city = ",
outputManager)
    };
    Console.WriteLine("-----¥n");
    #endif

    ICities cities = new Cities(details[0], details[1], reader, outputManager);
    outputManager.WriteLine(MessageSeverity.Success, cities.Result);

}

# if !BIRO
outputManager.WriteLine(MessageSeverity.Info, "Press any key to continue...");
Console.ReadKey();
#endif
}
}
}

```

## Tesztelés

### Érvényes tesztesetek

#### 1. *teszteset: bel.txt*

Bemenet – nincs település, nincs mérés
0 0
Kimenet
0

2. *teszteset: be2.txt*

Bemenet – 1 település, 1 mérés
1 1 1
Kimenet
1 1

3. *teszteset: be3.txt*

Bemenet – 3 település, 5 mérés, helyes kimenet
3 5 10 15 12 10 10 11 11 11 11 20 25 16 16 16 20
Kimenet
2 2 3

4. *teszteset: be4.txt*

Bemenet – 2 település és 3 mérés
2 3 3 5 8 -2 6 8
Kimenet
1 2

**Érvénytelen tesztesetek**

Billentyűzetes bevitel esetén

5. *teszteset*

Bemenet – szöveges adat
Number of cities = 11tizenegy
Kimenet
Incorrect format of values! (11tizenegy) Number of cities =

6. *teszteset*

Bemenet – Városok száma határokon kívüli szám $1 \leq N \leq 1000$
Number of cities = -1
Kimenet
Value is not in the range! (-1, range: 1 -> 1000) Number of cities =

## 7. *teszteset*

Bemenet – Mérések száma határokon kívüli szám $1 \leq N \leq 1000$	
Number of the temperatures in a city = 0	
Kimenet	
Value is not in the range! (0, range: 1 -> 1000)	
Number of cities =	

## Fejlesztési lehetőségek

1. Mivel osztály és interface van használva, így könnyen megoldható a fájlba való logolás
2. Illetve az IDataReader mögé akár egy SQL adatbázis implementációt is lehet rakni.
3. GUI kezelőfelület elkészítése