# „Programozási alapismeretek”
# beadandó feladat

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

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

***2023. január 21.***

# Tartalom

# 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 bevitel esetén*

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

| # | Adat | Magyarázat |
|---|---|---|
| **1.** | *Települések száma = (n)* | 1 és 1000 közti egész szám |
| **2.** | *Napok száma = (m)* | 1 és 1000 közti egész szám |
| **3.** | *1. város 1. mérése =* | -50 és 50 közti egész szám |
| **4.** | *1. város 2. mérése =* | -50 és 50 közti egész szám |
| **…** | *1.város m. mérése =* | -50 és 50 közti egész szám |
| **…** | … | |
| **…** | *2. város 1. mérése =* | -50 és 50 közti egész szám |
| | *2. város 2. mérése =* | -50 és 50 közti egész szám |
| **…** | *2. város m. mérése =* | -50 és 50 közti egész szám |
| **…** | … | |
| **…** | *n. város az m. mérése =* | -50 és 50 közti egész szám |

### *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:

```
.\yqmhwo.exe be1.txt
```

Illetve linux operációs rendszeren:

```
./yqmhwo be1.txt
```

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

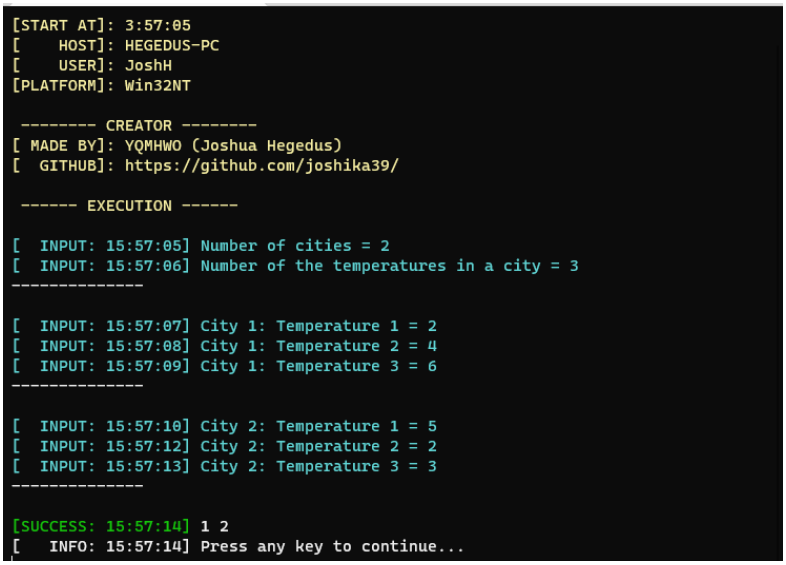
száma (n) és a mérések száma (m) van. A következő n sor mindegyikében m darabszám szerepel, közülük az i-edik sorban a j-edik szám az i-edik helységben a j-edik sorszámú fajból megfigyelt mérések száma. Például:

```
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 sorszámait, ahol a legmagasabb különbség lett mérve.

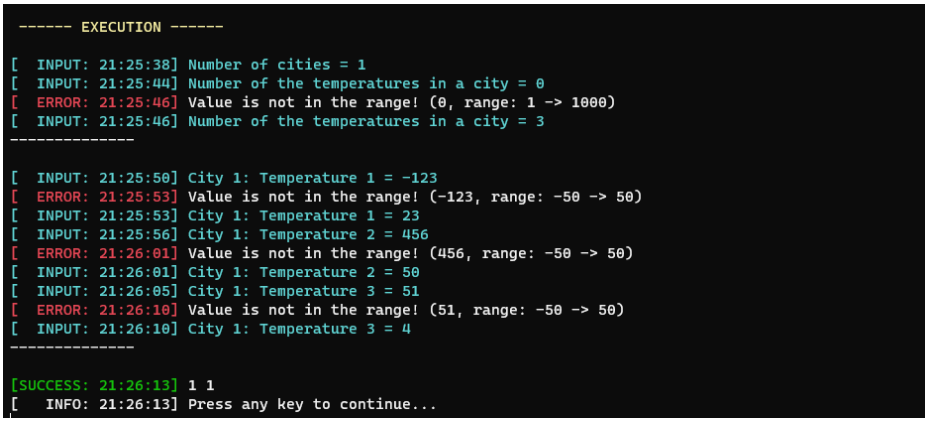
## *Minta bemenet és kimenet*




## *Hibalehetőségek*

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

### *Mintafutás hibás bemeneti adatok esetén:*



# 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: $n, m \in \mathbb{N}, cities_{1..n,1..m} \in \mathbb{N}^{nxm}$
Ki: $db \in \mathbb{N}, city_{1..db} \in \mathbb{N}^{db}$
Ef: $\mathrm{N} \in \mathbb{N}, 1 \leq \mathrm{N} \leq 1000$
$\mathrm{M} \in \mathbb{N},\ 1 \leq \mathrm{M} \leq 1000$
$-50 \leq \mathrm{Ho[M]_i} \leq 50$
Uf: $(db, helység) = Kiválogat_{\substack{i=1 \\ jó(i)}}^{n}\ i$ és
$db: \mathbb{N}$
$db = \forall_{j=1}^{m}(cities_{i,j} = maxDiff)$

### Visszavezetés

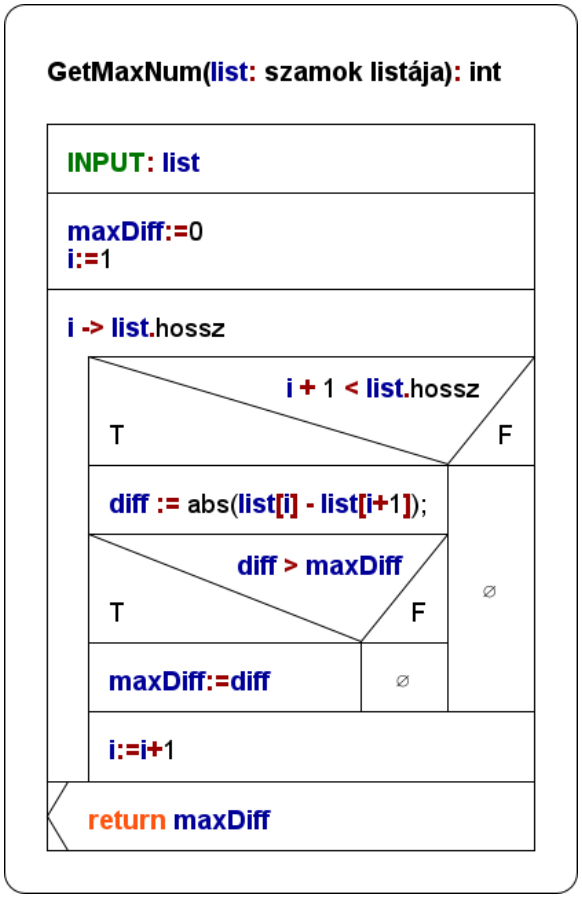
*Kiválogatás*

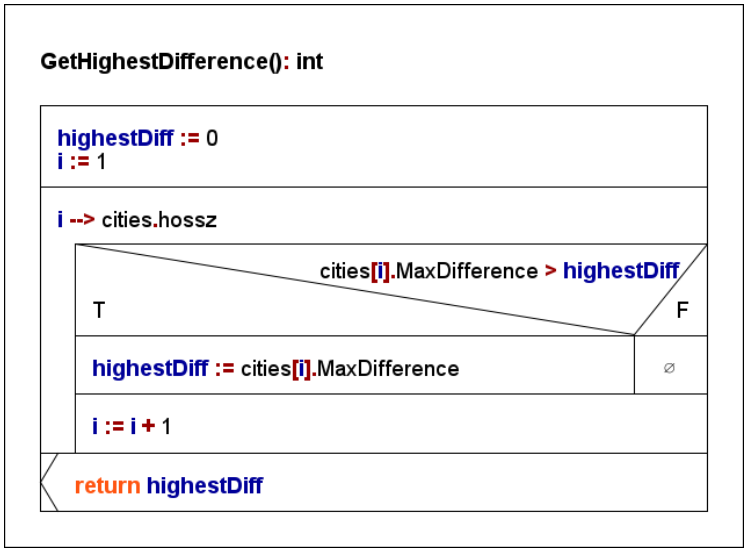
| | | |
|---|---|---|
| $y$ | ~ | $city$ |
| $T(x_i)$ | ~ | $cities_{i,j} = maxDiff$ |

*Eldöntés*

| | | |
|---|---|---|
| $i$ | ~ | $k$ |
| $T(x_i)$ | ~ | $i \neq k$ 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 `YQMHWO` nevű könyvtárban található meg. A fej-

lesztés során használt könyvtár-struktúra:

| Állomány | Magyarázat |
|---|---|
| `YQMHWO\x64\yqmhwo.exe` | Windows-on futtatható kód (a futtatáshoz szükséges fájlokkal) |
| `YQMHWO\unix\yqmhwo` | Linux-on futtatható kód |
| | |
| `YQMHWO\src\Program.cs` | C# forráskód |
| `YQMHWO\src\yqmhwo.csproj` | Projectfájl |
| `YQMHWO\Testcases\be1.txt` | teszt-bemeneti fájl$_1$ |
| `YQMHWO\Testcases\be2.txt` | teszt-bemeneti fájl$_2$ |
| `YQMHWO\Testcases\be3.txt` | teszt-bemeneti fájl$_3$ |
| `YQMHWO\Testcases\be4.txt` | teszt-bemeneti fájl$_4$ |
| | |
| `YQMHWO\YQMHWO.docx` | dokumentációk (ez a fájl) |
| | |

# Megoldás

## *Programparaméterek*

### *Típus*

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

### *Változó*

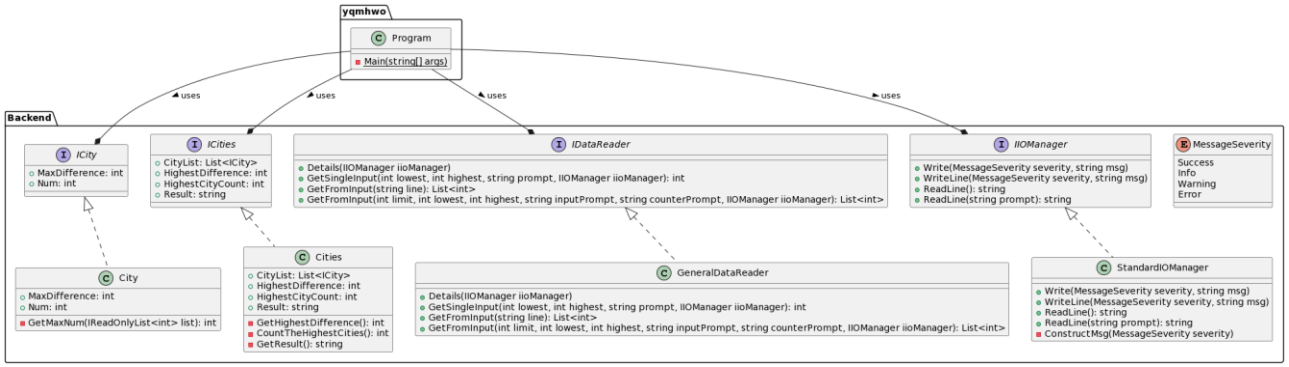
```
cities         : Cities(1..n: City)
```

## *Programfelépítés*

A program által használt modulok (és helyük):

`Program.cs` – program, a forráskönyvtárban
`yqhmwo.sln` – program 'megoldás fájl', a forráskönyvtárban
`yqhmwo.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: YQMHWO
  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
}

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]: YQMHWO (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 GeneralDataReader();
            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: be1.txt*

| **Bemenet** – *nincs település, nincs mérés* |
|---|
| 0 0 |
| **Kimenet** |
| 0 |

### *2. teszteset: be2.txt*

| **Bemenet** – *1 település, 1 mérés* |
|---|
| 1 1<br>1 |
| **Kimenet** |
| 1 1 |

### *3. teszteset: be3.txt*

| **Bemenet** – *3 település, 5 mérés, helyes kimenet* |
|---|
| 3 5 |
| 10 15 12 10 10<br>11 11 11 11 20<br>25 16 16 16 20 |
| **Kimenet** |
| 2 2 3 |

### *4. teszteset: be4.txt*

| **Bemenet** – *2 település és 3 mérés* |
|---|
| 2 3 |
| 3 5 8<br>-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)<br>Number of cities = |

### *6. teszteset*

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

### *7. teszteset*

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

## Fejlesztési lehetőségek

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