„Programozási alapismeretek”  
beadandó feladat

Készítette: *Tran Bach Khoa*Neptun-azonosító: *SAOF2T*E-mail: [tbkhoa98@gmail.com](mailto:tbkhoa98@gmail.com)

Kurzuskód: IP-08PAEDGyakorlatvezető neve: *???*

2019. december 11.

Tartalom

[Felhasználói dokumentáció 3](#_Toc410323328)

[Feladat 3](#_Toc410323329)

[Futási környezet 3](#_Toc410323330)

[Használat 3](#_Toc410323331)

[A program indítása 3](#_Toc410323332)

[A program bemenete 3](#_Toc410323333)

[A program kimenete 3](#_Toc410323334)

[Minta bemenet és kimenet 4](#_Toc410323335)

[Hibalehetőségek 4](#_Toc410323336)

[Fejlesztői dokumentáció 5](#_Toc410323337)

[Feladat 5](#_Toc410323338)

[Specifikáció 5](#_Toc410323339)

[Fejlesztői környezet 5](#_Toc410323340)

[Forráskód 6](#_Toc410323341)

[Megoldás 6](#_Toc410323342)

[Programparaméterek 6](#_Toc410323343)

[Programfelépítés 6](#_Toc410323344)

[Függvénystruktúra 6](#_Toc410323345)

[Algoritmus 7](#_Toc410323346)

[A kód 7](#_Toc410323347)

[Tesztelés 9](#_Toc410323348)

[Érvényes tesztesetek 9](#_Toc410323349)

[Érvénytelen tesztesetek 10](#_Toc410323350)

[Fejlesztési lehetőségek 10](#_Toc410323351)

2. Felhasználói dokumentáció
   1. 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 azokat a településeket, ahol az átlagtól a maximum jobban eltér, mint a minimum!

* 1. Futási környezet

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

* 1. Használat
     1. A program indítása

A program az SAOF2T\SAOF2T.exe néven található a tömörített állományban. A SAOF2T.exe fájl kiválasztásával indítható.

* + 1. A program bemenete

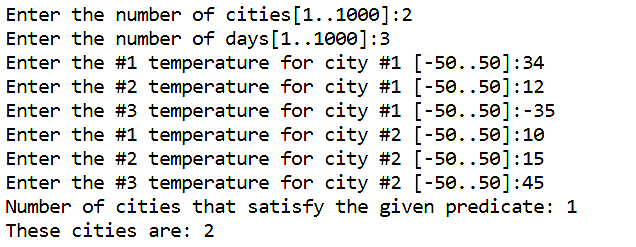
A program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

|  |  |  |
| --- | --- | --- |
| # | Adat | Magyarázat |
| **1.** | *N* | A települések száma (1≤N≤1000). |
| **2.** | *M* | A napok száma (1≤M≤1000). |
| **3.** | H*1* | Az első napra jósolt hőmérséklet értéke (-50≤Hi,j≤50). |
| **4.** | H*2* | A második napra jósolt hőmérséklet értéke (-50≤Hi,j≤50). |
| **...** | *…* |  |
| **N+2.** | H *N* | Az N-edik napra jósolt hőmérséklet értéke (-50≤Hi,j≤50). |

* + 1. A program kimenete

A standard kimenet első sorába azon települések T számát írja ki a program, ahol az átlagtól a maximum jobban eltér, mint a minimum, ezt követően pedig ezen települések sorszámát, növekvő sorrendben.

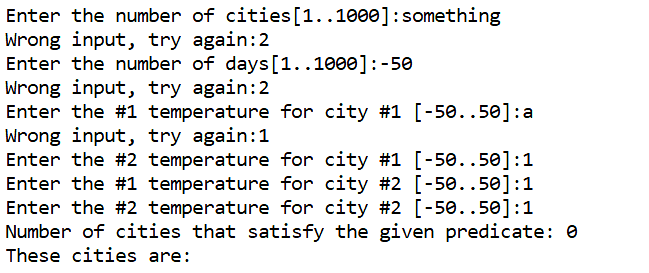
* + 1. Minta bemenet és kimenet



* + 1. Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha a települések száma nem egész szám, vagy nem esik a **1..1000** intervallumba; vagy ha a napok száma nem egész szám, vagy nem esik a **1..1000** intervallumba; vagy valamely hőmérséklet értéke nem egész szám, vagy nem esik a **-50..50** intervallumba. Hiba esetén a program azzal jelzi a hibát, hogy újra kérdezi azt.

* + - 1. Mintafutás hibás bemeneti adatok esetén:



1. Fejlesztői dokumentáció
   1. 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 azokat a településeket, ahol az átlagtól a maximum jobban eltér, mint a minimum!

* 1. Specifikáció

**Bemenet**: N∈N, M∈N, H∈NNxM

**Kimenet**: T∈N, napok∈NT

**Előfeltétel**: N=Hossz(H) ∧ N∈[1..1000] ∧ M∈[1..1000] ∧ ∀i∈[1..N]:M = Hossz(Hi) ∧ ∀i∈[1..N]:∀j∈[1..M]:Hi,j∈[-50..50]

**Utófeltétel**: T=hossz(napok) ∧

∀i∈[1..T]:napoki=abs(átlagoknapok.i-minoknapok.i) < abs(átlagoknapok.i-maxoknapok.i)

∀i∈[1..N]:abs(avgArray(Hi)-minArray(Hi)) < abs(avgArray(Hi)-maxArray(Hi)) ↔

∃j∈[1..T]:napokj=i

**Definíció**: sumArray: NN→N  
sumArray (List):=

avgArray: NN →N  
avgArray (List):= sumArray(List) / N

maxArray: NN →N  
maxArray (List):= ∀i∈[1..N]:Listi <= maxArray (List)

minArray: NN →N  
minArray (List):= ∀i∈[1..N]:Listi >= maxArray (List)

* 1. Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 7). mingw32-g++.exe c++ fordítóprogram (v4.7), Code::Blocks (v13.12) fejlesztői környezet.

* 1. Forráskód

A teljes fejlesztői anyag –kicsomagolás után– az SAOF2T nevű könyvtárban található meg. A fej­lesztés során használt könyvtár-struktúra:

|  |  |
| --- | --- |
| Állomány | Magyarázat |
| SAOF2T\SAOF2T.exe | futtatható kód |
|  |  |
| Biro version\saof2t.cpp | C++ forráskód, biró változat |
| Full version\saof2t.cpp | C++ forráskód, teljes változat |
|  |  |
| Test files\be1 - ver 1.txt | teszt-bemeneti fájl1.1 |
| Test files\be1 - ver 2.txt | teszt-bemeneti fájl1.2 |
| Test files\be1 - ver 3.txt | teszt-bemeneti fájl1.3 |
| Test files\be2.txt | teszt-bemeneti fájl2 |
| Test files\be3.txt | teszt-bemeneti fájl3 |
| Test files\be4.txt | teszt-bemeneti fájl4 |
| Test files\be5.txt | teszt-bemeneti fájl5 |
|  |  |
| documentation.docx | dokumentációk (ez a fájl) |

* 1. Megoldás
     1. Programparaméterek
        1. Konstans

1. MaxN : **Egész**(1000) [a települések maximális száma]  
   MaxM : **Egész**(1000) [a hőmérsékletek maximális száma]
   * + 1. Változó

N : **Egész**

M : **Egész**  
H : **Tömb**[1..MaxN:**Tömb**[1..MaxM:**Egész**]]  
napok : **Vector**<**Egész**>

* + 1. Programfelépítés

A program által használt modulok (és helyük):

saof2t.cpp – program, a forráskönyvtárban  
iostream – képernyő-, és billentyűkezelés, a C++ rendszer része

vector – C++ vectorjához tartozó könyvtár  
stdlib.h – általános rutinok, a C++ rendszer része

* + 1. Függvénystruktúra

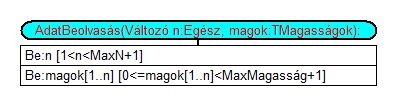


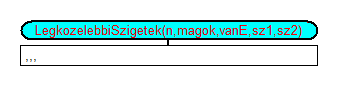
* + 1. A teljes program algoritmusa

Főprogram:

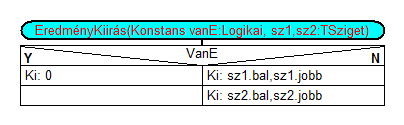


Alprogramok:





**Változó**  
 db:**Egész**  
 …



* + 1. A kód

A SAOF2T.cpp fájl tartalma:

1. /\*
2. Created by: Tran Bach Khoa
3. Neptun: SAOF2T
4. E-mail: tbkhoa98@gmail.com
5. Exercise: Időjárás előrejelzés - Az átlag kisebb a minimum és maximum átlagánál
6. \*/
7. #include <iostream>
8. #include <vector>
9. #include <stdio.h>
11. **using** **namespace** std;
13. **const** **int** MAXN = 1000;
14. **const** **int** MAXM = 1000;
15. **const** **int** MINT = -50;
16. **const** **int** MAXT = 50;
17. //Input:
18. **int** n = 0;
19. **int** m = 0;
20. **int** temps[MAXN][MAXM] = {0};
21. //Output:
22. vector <**int**> goodDays;
24. **void** readIn(**int** &n, **int** &m, **int** (&temps)[MAXN][MAXM]);
25. **int** readNumber(**int** min, **int** max);
26. **void** process(**int** &n, **int** &m, **int** (&temps)[MAXN][MAXM], vector<**int**> &goodDays);
27. **double** sumArray(**const** **int** list[], **int** n);
28. **double** avgArray(**const** **int** list[], **int** n);
29. **int** maxArray(**const** **int** list[], **int** n);
30. **int** minArray(**const** **int** list[], **int** n);
31. **void** printDays(**const** vector<**int**> &days);
33. **int** main(){
34. readIn(n, m, temps);
35. process(n, m, temps, goodDays);
36. printDays(goodDays);
37. **return** 0;
38. }
40. **void** readIn(**int** &n, **int** &m, **int** (&temps)[MAXN][MAXM]){
41. cout << "Enter the number of cities[1.." << MAXN << "]:";
42. n = readNumber(1, MAXN);
43. cout << "Enter the number of cities[1.." << MAXN << "]:";
44. m = readNumber(1, MAXM);
46. **for** (**int** i = 0; i < n; i++){
47. **for** (**int** j = 0; j < m; j++){
48. cout << "Enter the #" << j + 1 << " temperature for city #" << i + 1 << " [" <<   MINT << ".." << MAXT << "]:";
49. temps[i][j] = readNumber(MINT, MAXT);
50. }
51. cin.ignore(1000, '\n');
52. }
53. }
55. // Continuously tries to read in a single number from standard input.
56. // @min The minimal accepted number (included).
57. // @max The maximal accepted number (included).
58. **int** readNumber(**int** minVal, **int** maxVal){
59. **int** x;
60. **bool** inputCorrect = **false**;
61. **while**(!inputCorrect) {
62. cin >> x;
63. inputCorrect = !(x < minVal || maxVal < x || cin.fail());
64. **if**(!inputCorrect) {
65. cin.clear();
66. cin.ignore(1000, '\n');
67. cout << "Wrong input, try again:";
68. }
69. }
70. **return** x;
71. }
73. **void** process(**int** &n, **int** &m, **int** (&temps)[MAXN][MAXM], vector <**int**> &goodDays){
74. **for** (**int** i = 0; i < n; i++){
75. **double** avgT = avgArray(temps[i], m);
76. **int** maxT = maxArray(temps[i], m);
77. **int** minT = minArray(temps[i], m);
78. **if**(maxT-avgT > avgT-minT){
79. goodDays.push\_back(i+1);
80. }
81. }
82. }
84. //Calculates the sum of decimal values inside of an array of n length.
85. **double** sumArray(**const** **int**\* list, **int** n){
86. **double** sum = 0;
87. **for**(**int** i = 0; i < n; i++){
88. sum += (**double**)list[i];
89. }
90. **return** sum;
91. }
93. //Calculates the average of decimal values inside of an array of n length.
94. **double** avgArray(**const** **int**\* list, **int** n){
95. **double** avg = sumArray(list, n) / (**double**)n;
96. **return** avg;
97. }
99. //Takes the maximum value found in an n length array.
100. **int** maxArray(**const** **int**\* list, **int** n){
101. **int** maximum = list[0];
102. **for**(**int** i = 1; i < n; i++){
103. maximum = maximum < list[i] ? list[i] : maximum;
104. }
105. **return** maximum;
106. }
108. //Takes the minimum value found in an n length array.
109. **int** minArray(**const** **int**\* list, **int** n){
110. **int** minimum = list[0];
111. **for**(**int** i = 1; i < n; i++){
112. minimum = list[i] < minimum ? list[i] : minimum;
113. }
114. **return** minimum;
115. }
117. **void** printDays(**const** vector<**int**> &days){
118. cout << "Number of cities that satisfy the given predicate: " << days.size() < endl;
119. cout << "These cities are:";
120. **for**(**int** i = 0; i < days.size(); i++){
121. cout << " " << days[i];
122. }
123. }
     1. Tesztelés
        1. Érvényes tesztesetek
           1. teszteset: be1.txt

|  |
| --- |
| Bemenet – *1 város, 1 nap és bármilyen hőmérséklet* |
| N = 1  M = 1  H1,1= 1 vagy 23 vagy -13 |
| Kimenet |
| 0 |

* + - 1. teszteset: be2.txt

|  |
| --- |
| Bemenet – *1 város, 3 nap és a hőmérsékletek maxa jobban eltér* |
| N = 1  M = 3  H1,1= 1  H1,2= 2  H1,3= 5 |
| Kimenet |
| 1 1 |

* + - 1. teszteset: be3.txt

|  |
| --- |
| Bemenet – *1 város, 3 nap és a hőmérsékletek minimuma jobban eltér* |
| N = 1  M = 3  H1,1= 1  H1,2= 4  H1,3= 5 |
| Kimenet |
| 0 |

* + - 1. teszteset: be4.txt

|  |
| --- |
| Bemenet – *1 város, 3 nap és a hőmérsékletek minimuma és maxa egyenlően térnek el* |
| N = 1  M = 3  H1,1= 1  H1,2= 3  H1,3= 5 |
| Kimenet |
| 0 |

* + - 1. teszteset: be5.txt

|  |
| --- |
| Bemenet – *Nagyobb méretű teszteset* |
| N = 5  M = 5 |
| H1,1= -1  H1,2= -1  H1,3= -1  H1,4= -1  H1,5= 1 |
| H2,1= 1  H2,2= 2  H2,3= 3  H2,4= 4  H2,5= 5 |
| H3,1= 1  H3,2= 1  H3,3= 1  H3,4= 1  H3,5= 5 |
| H4,1= -42  H4,2= -23  H4,3= 5  H4,4= 15  H4,5= 35 |
| H5,1= -1  H5,2= 5  H5,3= 5  H5,4= 5  H5,5= 5 |
| Kimenet |
| 2 1 3 |

* + 1. Érvénytelen tesztesetek
       1. teszteset

|  |
| --- |
| Bemenet – *típushiba* |
| N = 54word |
| Kimenet |
| Wrong input, try again: N = |

* + - 1. teszteset

|  |
| --- |
| Bemenet – *Rossz magasság* |
| N = 5  M = -54 |
| Kimenet |
| Wrong input, try again: M= |

* 1. Fejlesztési lehetőségek

1. Adatok –a felhasználó igénye szerint– akár fájlból is fogadása.
2. Hibás fájl-bemenetek felismerése, és a hiba helyének (sor sorszámának) kiírása.
3. Többszöri futtatás megszervezése
4. A bemeneti sorozat grafikus megjelenítése és az eredmény városok elütő színű kijelzése.