# **Fejlesztői dokumentáció “Település valamikor minimális hőmérséklettel”**

Készítette: Tóth Zalán  
Neptun-azonosító: cz72ym  
E-mail: [zlaval@gmail.com](mailto:zlaval@gmail.com)  
  
Kurzuskód: IP-18ePROGEG  
Gyakorlatvezető: Siegler Gábor

**2019. november 30.**

# **Feladat**

A meteorológiai intézet az ország ***settlementCount*** településére adott ***numberOfDays*** napos időjárás előrejelzést. Az adott településen az adott napra várt legmagasabb hőmérsékletet jegyezték fel.  
A program összegyűjti azon települések listáját, melyeknél legalább egy napon az adott napi hőmérséklet megegyezet az adott napra előrejelzett hőmérsékletek minimumával.

# **Specifikáció**

**Bemenet:** settlementCount∈N, numberOfDays∈N, measures∈NsettlementCount\*numberOfDays

**Kimenet:** count∈N, rowNumbers∈NsettlementCount

**Előfeltétel:** 1≤settlementCount≤1000 ∧ 1≤numberOfDays≤1000 ∧ -50≤measure≤50,  
 measure∈measures

**Utófeltétel:** count:= ; ∧

rowNumbers∈measuresdb ;∧

rowNumbersj:=∀i∈[0..numberOfDays-1], ∀j∈[0..settlementCount-1]:

measuresj,i=minimumTemperaturesi∈true measuresj,i∉rowNumbers; ∧

minimumTemperaturesi:= ∀i∈[0..numberOfDays-1], ∀j∈[0..settlementCount-1]:

min(measuresj,i);

# **Project struktúra**

|  |  |
| --- | --- |
| Állomány | Leírás |
| progalap\_complex\main.cpp | C++ forráskód |
|  |  |
| progalap\_complex\test\be1.txt | Teszt-bemeneti fájl |
| progalap\_complex\test\ki1.txt | Teszt-kimeneti fájl |
| … |  |
| progalap\_complex\test\be8.txt | Teszt-bemeneti fájl |
| progalap\_complex\test\ki8.txt | Teszt-kimeneti fájl |
|  |  |
| progalap\_complex\doc\ cz72ym\_toth\_zalan\_complex\_developer\_documentation.pdf | Fejlesztői dokumentáció |
| progalap\_complex\doc\ cz72ym\_toth\_zalan\_complex\_test\_report.pdf | Tesztjegyzőkönyv |
| progalap\_complex\doc\ cz72ym\_toth\_zalan\_complex\_user\_documentation.pdf | Felhasználói dokumentáció |

# **Fejlesztői környezet**

Lenovo notebook. Windows 10 Pro. Jetbrains CLion 2019.2 IDE. Mingw 6.3.0 fordítóprogram. Git verzió kezelő.

# **Megoldás**

## ***Programparaméterek***

***Konstans***

MAX\_COUNT : **int**(1000) [települések és napok maximális száma]  
MIN\_COUNT : **int**(1) [települések és napok minimális száma]  
MIN\_TEMPERATURE : **int**(-50) [minimum megadható hőmérséklet]  
MAX\_TEMPERATURE : **int**(50) [maximum megadható hőmérséklet]

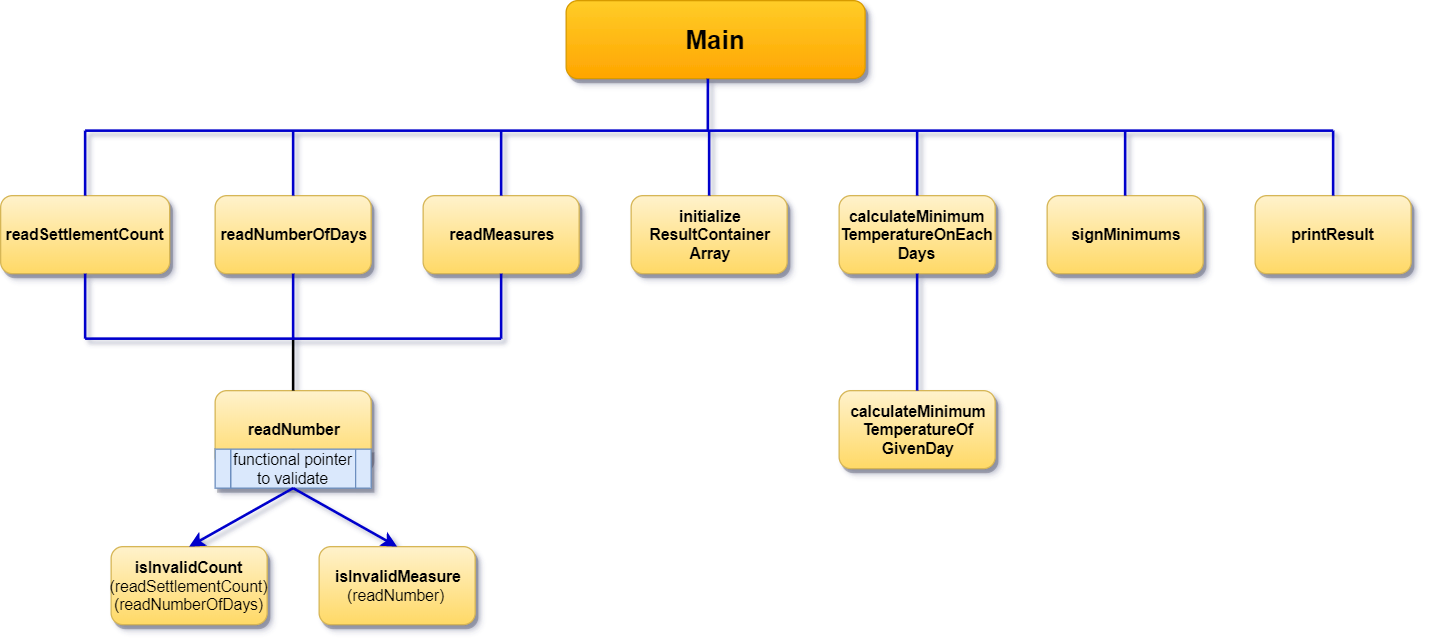
***Változó***

measures :**array**[MAX\_COUNT][MAX\_COUNT] [mérési értékek]  
settlementCount :**int**(MAX\_COUNT) [települések száma]  
numberOfDays :**int**(MAX\_COUNT) [napok száma]

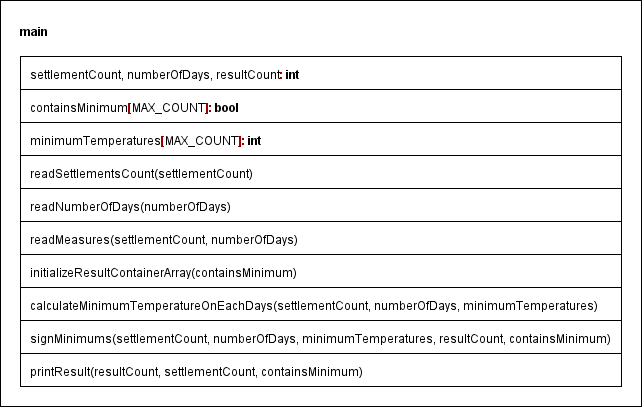
## ***Programfelépítés***

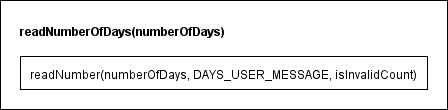
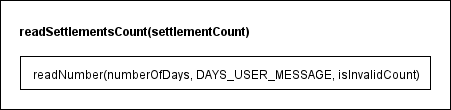
*main.cpp* [a program, a főkönyvtárban]  
*<iostream>* [képernyő és billentyűzet kezelés, a C++ része]  
*<string>* [karakterláncok kezelése, a C++ része]

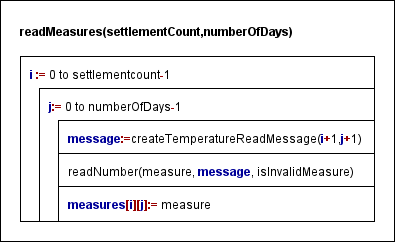
## ***Függvénystruktúra***

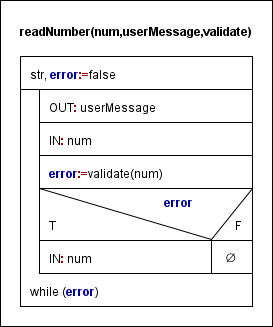


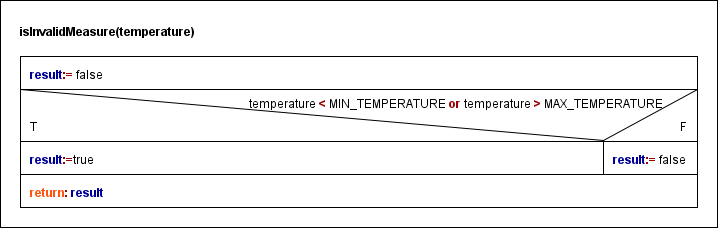
## ***Algoritmus***

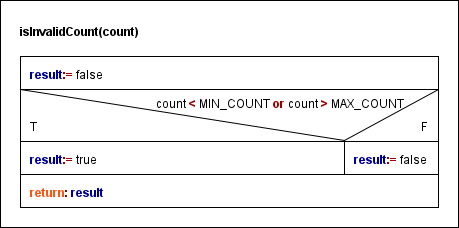


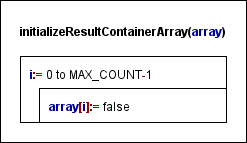


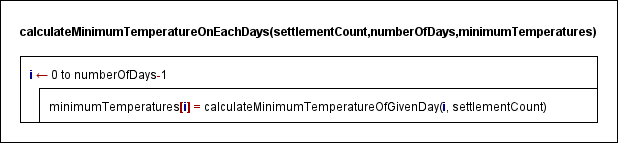


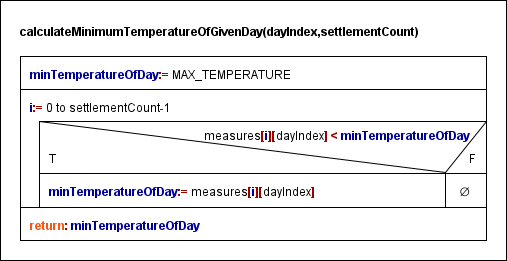


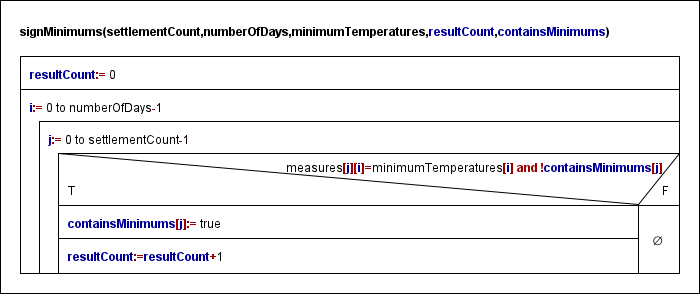


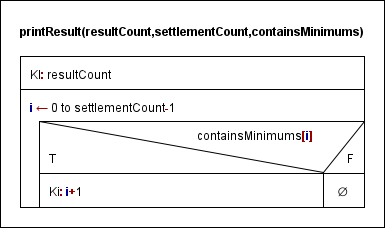












# ***Fejlesztési lehetőségek***

* Adatok fájlból is beolvashatók.
* Hibás fájl-bemenetek felismerése, valamint a hiba helyének kijelzése.
* Többszöri futtatás megszervezése.
* A városok nevének beolvasása, eredményben történő megjelenítése.
* Napok dátumként történő bekérése, napi minimumok megjelenítése.
* Városonként megjeleníthető -akár grafikusan-, mely napokon egyezett meg a hőmérséklet az adott napi minimum hőmérséklettel.

# **A kód**

### A *main.ccp* fájl tartalma:

/\* Developer: Zalan Toth  
 \* Neptun: cz72ym  
 \* Email: zlaval@gmail.com  
 \* Task: Minimal temperature on a settlement in any time  
 \*/  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
const string SETTLEMENTS\_USER\_MESSAGE = "Please enter the number of the settlements";  
const string DAYS\_USER\_MESSAGE = "Please enter the number of the days";  
  
const int MAX\_COUNT = 1000;  
const int MIN\_COUNT = 1;  
  
const int MIN\_TEMPERATURE = -50;  
const int MAX\_TEMPERATURE = +50;  
  
int measures[MAX\_COUNT][MAX\_COUNT];  
  
void readNumber(int &num, const string &userMessage, bool (\*validate)(const int &)) {  
#ifdef BIRO  
 cin >> num;  
#else  
 bool error = false;  
 string str;  
 do {  
 cout << userMessage << ": ";  
 cin >> num;  
 bool nan = cin.fail();  
 error = nan ? nan : validate(num);  
 if (error) {  
 cout << "Unexpected data: ";  
 if (nan) {  
 cout << "NAN!";  
 } else {  
 cout << num;  
 }  
 cout << ", please enter a valid number within the given range!";  
 cin.clear();  
 getline(cin, str);  
 }  
 cout << endl;  
 } while (error);  
#endif  
}  
  
string createTemperatureReadMessage(const int &row, const int &col) {  
#ifdef BIRO  
 return "";  
#else  
 string message = "Please enter the ";  
 message.append(to\_string(row));  
 message.append(". settlement's ");  
 message.append(to\_string(col));  
 message.append(". temperature");  
 return message;  
#endif  
}  
  
void printResult(const int &resultCount, const int &settlementCount, const bool \*const containsMinimums) {  
#ifdef BIRO  
 cout << resultCount << " ";  
#else  
 cout << "Number of settlements contains a daily minimum temperature: " << resultCount << endl;  
#endif  
 for (int i = 0; i < settlementCount; i++) {  
 if (containsMinimums[i]) {  
 cout << (i + 1) << " ";  
 }  
 }  
}  
  
bool isInvalidCount(const int &count) {  
 if (count < MIN\_COUNT || count > MAX\_COUNT) {  
 cout << count << " is not in the range " << MIN\_COUNT << " and " << MAX\_COUNT << endl;  
 return true;  
 }  
 return false;  
}  
  
bool isInvalidMeasure(const int &temperature) {  
 if (temperature < MIN\_TEMPERATURE || temperature > MAX\_TEMPERATURE) {  
 cout << temperature << " Celsius is not in the range " << MIN\_TEMPERATURE << " and " << MAX\_TEMPERATURE << endl;  
 return true;  
 }  
 return false;  
}  
  
void readSettlementsCount(int &settlementCount) {  
 readNumber(settlementCount, SETTLEMENTS\_USER\_MESSAGE, isInvalidCount);  
}  
  
void readNumberOfDays(int &numberOfDays) {  
 readNumber(numberOfDays, DAYS\_USER\_MESSAGE, isInvalidCount);  
}  
  
void readMeasures(const int &settlementCount, const int &numberOfDays) {  
 for (int i = 0; i < settlementCount; i++) {  
 for (int j = 0; j < numberOfDays; j++) {  
 string message = createTemperatureReadMessage(i + 1, j + 1);  
 int measure;  
 readNumber(measure, message, isInvalidMeasure);  
 measures[i][j] = measure;  
 }  
 }  
}  
  
int calculateMinimumTemperatureOfGivenDay(const int &dayIndex, const int &settlementCount) {  
 int minTemperatureOfDay = MAX\_TEMPERATURE;  
 for (int j = 0; j < settlementCount; j++) {  
 int actualTemperature = measures[j][dayIndex];  
 if (actualTemperature < minTemperatureOfDay) {  
 minTemperatureOfDay = actualTemperature;  
 }  
 }  
 return minTemperatureOfDay;  
}  
  
void calculateMinimumTemperatureOnEachDays(const int &settlementCount,  
 const int &numberOfDays,  
 int \*const minimumTemperatures) {  
 for (int i = 0; i < numberOfDays; i++) {  
 minimumTemperatures[i] = calculateMinimumTemperatureOfGivenDay(i, settlementCount);  
 }  
}  
  
void initializeResultContainerArray(bool \*const array, const int &size) {  
 for (int i = 0; i < size; i++) {  
 array[i] = false;  
 }  
}  
  
void signMinimums(const int &settlementCount,  
 const int &numberOfDays,  
 const int \*const minimumTemperatures,  
 int &resultCount,  
 bool \*const containsMinimums) {  
  
 resultCount = 0;  
 for (int i = 0; i < numberOfDays; i++) {  
 int minTempOfDay = minimumTemperatures[i];  
 for (int j = 0; j < settlementCount; j++) {  
 int actualTemperature = measures[j][i];  
 if (actualTemperature == minTempOfDay && !containsMinimums[j]) {  
 containsMinimums[j] = true;  
 resultCount++;  
 }  
 }  
 }  
}  
  
int main() {  
 int settlementCount, numberOfDays, resultCount;  
 bool containsMinimum[MAX\_COUNT];   
 int minimumTemperatures[MAX\_COUNT];  
  
 readSettlementsCount(settlementCount);  
 readNumberOfDays(numberOfDays);  
 readMeasures(settlementCount, numberOfDays);  
 initializeResultContainerArray(containsMinimum, MAX\_COUNT);  
 calculateMinimumTemperatureOnEachDays(settlementCount, numberOfDays, minimumTemperatures);  
 signMinimums(settlementCount, numberOfDays, minimumTemperatures, resultCount, containsMinimum);  
 printResult(resultCount, settlementCount, containsMinimum);  
  
 return 0;  
}