

Relevés de températures [it04]

Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel  algoprog  Version 17 mai 2018

Table des matières

1 Relevés de températures / pgmeteo	2
1.1 Énoncé	2
1.2 Problème 1	3
1.3 Problème 2	3
1.4 Problème 3	4

C++ - Relevés de températures (TP)



Mots-Clés Schéma itératif ■

Requis Structures de base, Structures conditionnelles, Structures répétitives ■

Difficulté ●●○ (1 h) ■



Objectif

Cet exercice traite de relevés de températures. Il reprend des algorithmes de base étudiés dans les exercices @[Somme d'une suite] (dans le support de cours) et @[Maximum et son rang] (dans ce module). Il aborde aussi le problème du contrôle des réponses de l'utilisateur.

1 Relevés de températures / pgmeteo

1.1 Énoncé

Une station météo effectue chaque jour un relevé de températures pendant une période donnée de n jours consécutifs. Il s'agit d'écrire un programme qui :

- Saisit le nombre n de jours de la période à traiter.
- Pour chaque jour de la période, affiche le numéro du jour et saisit la température relevée ce jour.
- Calcule et affiche la température minimale et le numéro du jour où elle a été relevée.
- Calcule et affiche la température maximale et le numéro du jour où elle a été relevée.
- Calcule et affiche la température moyenne sur la période de relevés.

On demande trois versions qui correspondent aux trois questions suivantes :

1. Dans la première version, on ne fait pas d'hypothèse sur les valeurs des températures. On suppose que l'utilisateur donne un nombre de jours strictement positif.
2. Dans la deuxième version, sans le contrôler, on suppose que l'utilisateur donne un nombre de jours positifs et des températures qui appartiennent à l'intervalle $[-80, 80]$.
3. Dans la troisième version, l'algorithme oblige l'utilisateur à saisir un nombre de jours compris entre 1 et 366, ainsi qu'une température comprise en -80 et 80. L'obligation signifie que tant que l'utilisateur ne donne pas une valeur satisfaisante, l'algorithme lui redemande une valeur.

Remarque

Il peut y avoir des ex-aequo, c.-à-d. plusieurs jours correspondant à la température minimale (resp. maximale). Dans ce cas, l'algorithme n'affichera que la première occurrence du minimum (resp. maximum) rencontrée.

Exemple

Voici un exemple de dialogue correspondant à l'algorithme demandé :

```
Nombre de jours dans la periode? 7
Temperature du jour 1? 5
Temperature du jour 2? 8
Temperature du jour 3? -3
Temperature du jour 4? 4
Temperature du jour 5? 10
Temperature du jour 6? -3
Temperature du jour 7? -1
==> Temperature Mini = -3 jour 3
==> Temperature Maxi = 10 jour 5
==> Temperature Moyn = 2.9
```

1.2 Problème 1

Le problème de la recherche du maximum est analogue à la première question du problème @[Maximum et rang]. Celui du minimum l'est aussi, à condition de changer le sens des inégalités. Le nombre de valeurs à traiter correspond ici au nombre de jours de la période. L'énoncé ne le précisant pas, on décide de déclarer les températures avec le type réel.



Écrivez un programme qui déclarent les variables.



Initialisez les variables.



Écrivez la boucle du traitement.



Calculez et affichez les résultats.



Testez.

1.3 Problème 2

Dans ce problème, il existe une contrainte sur les températures qui doivent appartenir à l'intervalle $[-80, 80]$. On suppose toujours que l'utilisateur répond en cohérence avec ce qui lui est demandé. On va utiliser ces contraintes pour modifier la méthode d'initialisation. En effet, si on initialise la température minimale avec une valeur supérieure à la plus grand valeur possible (par exemple 81), elle sera forcément mise à jour lors de la première comparaison dans la boucle. Il en est de même si on initialise la température maximale avec une valeur inférieure à la plus petite valeur possible (par exemple -81).

Il n'est plus nécessaire de saisir la température du premier jour à l'extérieur de la boucle à condition que celle-ci soit exécutée dès le premier jour. Précisons aussi que les variables `jmin` et `jmax` étant nécessairement affectées lors de la première comparaison dans la boucle, il n'est pas nécessaire de les initialiser. Enfin, la somme doit être initialisée à 0 puisque la première température est traitée dans la boucle.

Avant de modifier l'algorithme, expliquons pourquoi la modification doit être faite sur la version 1.1 (utilisant des `Si` en séquence) et pourquoi elle est **fausse sur la version 1.2** (utilisant le `Si-Sinon-Si`). Il faut nécessairement que les deux températures `tmin` et `tmax` soient affectées au moins une fois dans la boucle. Il en est de même pour les numéros de jours `jmin` et `jmax`. Or la conditionnelle imbriquée pourrait ne pas le permettre. Supposons e effet que l'expression `tj < tmin` soit vraie à chaque passage dans la boucle. Cette situation peut se présenter si les températures apparaissent dans un ordre strictement décroissant qui fait que la température courante est toujours inférieure à la température de référence. Dans ce cas, la clause `Sinon` ne serait jamais activé, la variable `tmax` garderait sa valeur de départ (-81), et la variable `jmax` n'aurait pas de valeur. C'est pourquoi cette méthode d'initialisation par des valeurs extérieures à l'intervalle de validité ne peut être utilisée

qu'avec la première version de l'algorithme, celle où les deux variables `tmin` et `tmax` sont toutes deux testées par les conditionnelles consécutives.



Définissez les constantes `TPMIN=-80` et `TPMAX=80` représentant la contrainte $[-80, 80]$.



Modifiez ensuite votre programme selon cette analyse.



Testez.

1.4 Problème 3

Nous ne voulons plus faire confiance à l'utilisateur, il faut le contraindre à fournir des valeurs ayant un sens pour le problème. Dans le cas présent, l'utilisateur doit donner un nombre de jours compris entre 1 et 366, et des températures comprises entre `TPMIN=-80` et `TPMAX=80`. On va donc lui poser la question jusqu'à ce qu'il donne une réponse correcte. Comme il y a forcément une première saisie on peut envisager une boucle `Répéter`. Par exemple dans le cas des températures, on peut envisager une boucle comme suit :

```
Répéter
  Afficher("Temp ([" ,TPMIN," "," ,TPMAX," ]) jour ", nj,"? ")
  Saisir(tj)
Jusqu'à (tj >= TPMIN Et tj <= TPMAX)
```

Supposons que nous soyons au jour 5 et que l'utilisateur donne trois fois de suite une température erronée. Voici ce qu'il y aura comme dialogue :

```
Temp ([-80,80]) jour 5? 150
Temp ([-80,80]) jour 5? -110
Temp ([-80,80]) jour 5? 90
Temp ([-80,80]) jour 5? 15
Temp ([-80,80]) jour 6? ...
```

Si on veut signifier à l'utilisateur l'existence d'une erreur, il faut alors distinguer le message initial des messages d'erreur et l'utilisation d'une répétitive `Répéter` n'est plus possible. Il faut utiliser une répétitive `TantQue` dont le sens sera : tant qu'il y a une erreur donner une nouvelle valeur.

La condition d'entrée dans la boucle `TantQue` doit être la négation de la condition de sortie de la boucle `Répéter`. On aura donc :

```
Afficher("Temp ([" ,TPMIN," "," ,TPMAX," ]) jour ", nj,"? ")
Saisir(tj)
TantQue (non (tj >= TPMIN Et tj <= TPMAX)) Faire
  Afficher("Erreur Temp ([" ,TPMIN," "," ,TPMAX," ])? ")
  Saisir(tj)
FinTantQue
```

Le dialogue précédent est modifié, rendant repérables les messages correspondant à une erreur :

```
Temp ([-80,80]) jour 5? 150
Erreur Temp ([-80,80])? -110
Erreur Temp ([-80,80])? 90
Erreur Temp ([-80,80])? 15
Temp ([-80,80]) jour 6? ...
```



Complétez votre programme selon cette analyse.



Testez.