Parcours imbriqué [tb07] – Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 19 mai 2018

Table des matières

C - Parcours imbriqué (Solution)

Mots-Clés Tableau unidimensionnel ■
Utilise Définitions et notations, Tableaux et paramètres, Parcours de tableaux ■
Difficulté • ○ ○ (30 min) ■

1 Parcours imbriqué / pgimbrique

Objectif

Cet exercice détermine le nombre de successeurs supérieurs pour chaque case d'un tableau d'entiers.

Définitions C

```c
enum { TMAX = ... };
typedef int ITableau[TMAX];
```

1.1 Procédure calculerNSuperieurs (nombre de supérieurs)

Ce problème calcule, pour chaque case d'un `ITableau`, le nombre de cases suivantes qui contiennent un élément strictement supérieur. Exemple :

```
[45, 54, 1, -56, 22, 134, 49, 12, 90, -27]
==> [ 4 2 5 6 3 0 1 1 0 0 ]
```

Écrivez une fonction `nsuperieursTab(t,k,n,valeur)` qui calcule et renvoie le nombre d'éléments de `t[k..n]` supérieur à `valeur` (entier), où `t` est un `ITableau`.

Déduisez une procédure `calculerNSuperieurs(t,n,trs)` qui, dans un `ITableau trs`, calcule le nombre de valeurs successeurs supérieures à `t[ix]` pour chaque case `ix` des `n` éléments d'un `ITableau t`.

Validez votre fonction et votre procédure avec la solution.

Solution C @[pgimbrique.c]

```c
int nsuperieursTab(const Tableau t,int k,int n,int valeur)
{
  int rs = 0;
  int ix;
  for (ix=k ; ix<n ; ++ix)
  {
    if (t[ix] > valeur)
    {
      rs += 1;
    }
  }
  return rs;
}
```

```c
void calculerNSuperieurs(const Tableau t,int n,Tableau trs)
{
  int ix;
  for (ix=0 ; ix<n ; ++ix)
  {
    trs[ix] = nsuperieursTab(t, ix+1, n, t[ix]);
  }
}
```

1.2 Procédure afficherTab (affichage d'un tableau)

Écrivez le **profil** d'une procédure `afficherTab(t,n)` qui affiche les `n` premières valeurs d'un `ITableau t`.

Affichage des valeurs
L'affichage de `t` est un parcours complet des `n` valeurs.
Il est donc piloté par une boucle `Pour`.

Écrivez le corps de la procédure. Affichez les valeurs à la queue-leu-leu séparés par un espace, le tout entre crochet. Exemple :

```
[45 54... -27]
```

Validez votre procédure avec la solution.

Solution C @[UtilsTB.c]

```c
void afficherTab(const Tableau t,int n)
{
  printf("[");
  int ix;
  for (ix=0; ix<n; ++ix)
  {
    printf("%d ",t[ix]);
  }
  printf("]\n");
}
```

1.3 Programme de test

Téléchargez le fichier suivant et mettez-le dans votre dossier.
C @[UtilsTB.c]

Copiez/collez ensuite les lignes suivantes :
C **Au début** de votre programme :

```c
#include "UtilsTB.c"
```

Soit la fonction `saisirTab(t)` qui effectue la saisie contrôlée du nombre de valeurs (entier compris entre 1 et `TMAX`), saisit les valeurs entières dans un `ITableau t` puis renvoie l'entier du nombre de valeurs saisies.

C @[saisirTab] (dans UtilsTB)

Écrivez un programme qui calcule, pour chaque case d'un `ITableau` de `nelems` valeurs, le nombre de cases suivantes qui contiennent un élément strictement supérieur. Les résultats seront placés dans un autre `ITableau` puis ce dernier sera affiché.

Testez. Exemple d'exécution :

```
Nombre d'éléments dans [1..50]? 10
t[0]? 45
t[1]? 54
t[2]? 1
t[3]? -56
t[4]? 22
1t[5]? 134
t[6]? 49
t[7]? 12
t[8]? 90
t[9]? -27
[ 4 2 5 6 3 0 1 1 0 0 ]
```

Validez votre programme avec la solution.

Solution C @[pgimbrique.c]

```c
int main(void)
{
  Tableau tab;
  Tableau trs;
  int nelems = saisirTab(tab);
  calculerNSuperieurs(tab,nelems,trs);
  afficherTab(trs,nelems);
}
```

1.4 Fonction ninferieursTab (nombre d'inférieurs)

En partant de la fonction et procédure écrites ci-dessus,

Écrivez une fonction `ninferieursTab(t,k,n,valeur)` qui calcule et renvoie le nombre de valeurs inférieures à une valeur `valeur` (entier) dans les `t[k..n]` valeurs d'un `ITableau t`.

Déduisez une procédure `calculerNInferieurs(t,n,trs)` qui calcule dans un `ITableau trs` le nombre de valeurs successeurs inférieures à `t[ix]` pour chaque case `ix` des `n` éléments d'un `ITableau t` d'entiers.

Validez votre fonction et votre procédure avec la solution.

Solution C @[pgimbrique.c]

```c
int ninferieursTab(const Tableau t,int k,int n,int valeur)
{
  int rs = 0;
  int ix;
  for (ix=k ; ix<n ; ++ix)
  {
    if (t[ix] < valeur)
    {
      rs += 1;
    }
  }
  return rs;
}
```

```c
void calculerNInferieurs(const Tableau t,int n,Tableau trs)
{
  int ix;
  for (ix=0 ; ix<n ; ++ix)
  {
    trs[ix] = ninferieursTab(t, ix+1, n, t[ix]);
  }
}
```

2 Références générales

Comprend ■