Permutation de variables [bs12] - Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 13 mai 2018

Table des matières

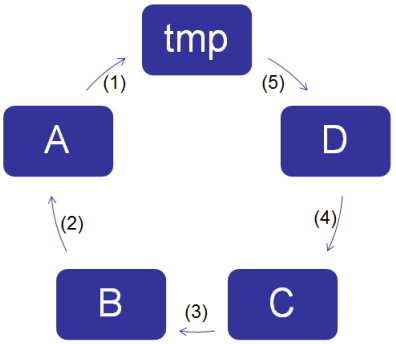
1 Permutation circulaire / pgpermuter **2**
- 1.1 Échanger deux variables . . . 2
- 1.2 Permuter trois variables (1) . . . 3
- 1.3 Permuter trois variables (2) . . . 4
- 1.4 Algorithmique : Cas général . . . 5

2 Références générales **7**

C++ - Permutation circulaire (Solution)

Mots-Clés Structures de base ■
Difficulté ●●○ (30 min) ■

Objectif
Cet exercice traite le problème **incontournable** de l'échange et de la permutation du contenu de variables, appliqué ici à des variables caractère.



...(énoncé page suivante)...

1 Permutation circulaire / pgpermuter

1.1 Échanger deux variables

Voici un exemple du résultat attendu :

```
Vos deux caracteres? z m
==> a vaut z et b vaut m
==> je permutes
==> a vaut m et b vaut z
```



Écrivez un programme qui saisit deux caractères dans `a` et `b`.
Affichez l'invite :

```
Vos deux caracteres?
```



Affichez (où `[x]` désigne le contenu de `x`) :

```
==> a vaut [a] et b vaut [b]
```



Affichez le message suivant :

```
==> je permutes
```



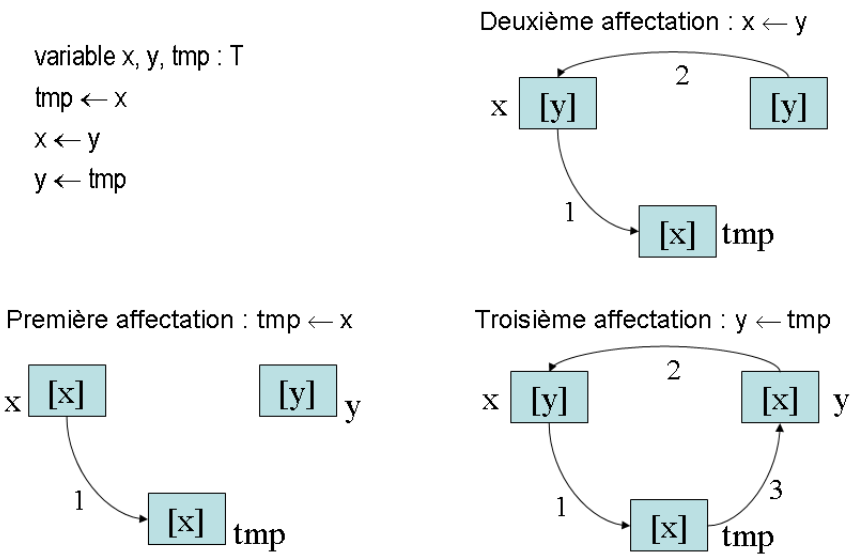
Échangez le contenu des deux variables.

Rappel de cours

Pour échanger le contenu de deux variables, il faut passer par une variable intermédiaire de même type.

Solution simple

Soit `tmp` la variable temporaire. L'écriture `[v]` désigne le contenu de `v`.



Copiez/coller l'affichage :

```
==> a vaut [a] et b vaut [b]
```

Testez.

Validez votre programme avec la solution.

Solution C++ @[pgpermuter1.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  char a, b;
  cout<<"Vos deux caracteres? ";
  cin>>a>>b;
  cout<<"==> a vaut "<<a<<" et b vaut "<<b<<endl;
  cout<<"==> je permutes"<<endl;
  char tmp;
  tmp = a;
  a = b;
  b = tmp;
  cout<<"==> a vaut "<<a<<" et b vaut "<<b<<endl;
}
```

1.2 Permuter trois variables (1)

Complétez votre programme afin qu'il saisit un troisième caractère dans `c`.
Modifiez l'invite en :

```
Vos trois caracteres?
```



Modifiez l'affichage issue de la saisie :

```
==> a vaut [a], b vaut [b] et c vaut [c]
```



Complétez l'échange en la permutation circulaire en effectuant d'abord la permutation des deux premières variables `a` et `b`, ensuite celle des deux dernières `b` et `c`. N'utilisez qu'une (et une seule) variable supplémentaire.



Modifiez l'affichage du résultat :

```
==> a vaut [a], b vaut [b] et c vaut [c]
```



Testez. Exemple d'exécution :

```
Vos trois caracteres? z m a
==> a vaut z, b vaut m et c vaut a
==> je permutes
==> a vaut m, b vaut a et c vaut z
```



Validez votre programme avec la solution.

Solution C++ @[pgpermuter2.cpp]

```cpp
#include <iostream>
using namespace std;

int main()
{
  char a, b, c;
  cout<<"Vos trois caracteres? ";
  cin>>a>>b>>c;
  cout<<"==> a vaut "<<a<<", b vaut "<<b<<" et c vaut "<<c<<endl;
  cout<<"==> je permutes"<<endl;
  char tmp;
  tmp = a;
  a = b;
  b = tmp;
  tmp = b;
  b = c;
  c = tmp;
  cout<<"==> a vaut "<<a<<", b vaut "<<b<<" et c vaut "<<c<<endl;
}
```

1.3 Permuter trois variables (2)



Modifiez astucieusement votre programme pour qu'il permute circulairement, **vers la gauche**, le contenu des trois variables de sorte que le contenu de `a` passe dans `c`, celui de `b` dans `a` et celui de `c` dans `b`. Si le premier affichage donne `fou`, le deuxième donnera `ouf`. N'utilisez que votre variable temporaire.

Solution simple

Le principe est le même que pour l'échange. Il faut que la variable temporaire garde la valeur de l'une des variables pendant que chaque variable prend la valeur de la suivante (ou de la précédente) ; plusieurs séquences d'instructions sont possibles. Il faut juste veiller à ce que l'ordre des instructions ait bien comme résultat la permutation circulaire à gauche comme demandée par l'énoncé.

Erreurs courantes

Pour s'assurer de ne pas faire d'erreur, il faut mettre à gauche de l'opérateur d'affectation, la variable qui a été sauvegardée à l'instruction précédente.



Testez.



Validez votre programme avec la solution.

Solution C++

On met en commentaire les deux instructions suivantes :

```
// b = tmp;
// tmp = b;
```

1.4 Algorithmique : Cas général



Écrivez deux **algorithmes** réalisant la permutation circulaire de quatre variables `a`, `b`, `c` et `e` de type `T` quelconques. Le premier sera une succession de transpositions, le second sera optimisé au regard du temps d'exécution.



Validez vos algorithmes avec la solution.

Solution : Premier algorithme

```
Algorithme pgpermuter4a
Variable a , b , c , e : T
Variable tmp : T
Début
  |  Saisir ( a , b , c , e )
  |  tmp <- a
  |  a <- b
  |  b <- tmp
  |
  |  tmp <- b
  |  b <- c
  |  c <- tmp
  |
  |  tmp <- c
  |  c <- e
  |  e <- tmp
```

```
  |  Afficher ( a , b , c , e )
Fin
```

Solution : Deuxième algorithme

```
Algorithme pgpermuter4b
Variable a , b , c , e : T
Variable tmp : T
Début
  |  Saisir ( a , b , c , e )
  |  tmp <- a
  |  a <- b
  |  b <- c
  |  c <- e
  |  e <- tmp
  |  Afficher ( a , b , c , e )
Fin
```

Solution commentée

Le deuxième algorithme est un perfectionnement du premier dont la trace montre que la séquence :

```
b <- tmp
tmp <- b
```

est inutile puisqu'elle réaffecte à `tmp` sa valeur : on peut donc la supprimer. De même, on peut supprimer la séquence :

```
c <- tmp
tmp <- c
```

pour obtenir finalement le deuxième algorithme.



Combien chacun comporte-t-il d'affectations ?

Solution simple

La permutation circulaire de 4 variables coûte 9 affectations par la première méthode, 5 par la seconde.



Généralisez au cas de n variables (en nombre d'affectations).

Solution simple

Dans le cas de n variables, la première coûte bien évidemment $3(n-1)$ affectations et la seconde seulement $n+1$.

2 Références générales

Comprend [Chaty-PG1-c1 :ex1], [Maysonnave-AL1 :c2 :ex6-8], [Tartier-AL1 :c3 :ex3], [Warin-PG1 :c4 :ex11] ■