Procédures et Paramètres formels [ss]
Exercices de cours

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 16 mai 2018

Table des matières

Java - Exercices de cours (Solution)

Mots-Clés Algorithmes paramétrés ■
Difficulté ●●○ (1 h) ■

1 Appréhender le cours

1.1 Salutation téléphonique / pgproc

Écrivez une procédure `un` qui affiche « Allo ».

Écrivez une procédure `deux` qui :

1. Appelle la procédure `un`.
2. Puis affiche « Bonjour ».

Écrivez un programme qui appelle les procédures `deux` puis `un` puis `deux`.

Testez. Résultat d'exécution :

```
Allo
Bonjour
Allo
Allo
Bonjour
```

Validez votre programme avec la solution.

Solution Java @[pgproc.java]

```java
class PGProc {

/**
  Procédure
*/

public static void un()
{
  System.out.println("Allo");
}

/**
  Procédure
*/

public static void deux()
{
  un();
  System.out.println("Bonjour");
}

public static void main(String[] args)
{
  deux();
```

```
  un();
  deux();
}

}
```

1.2 Saisie d'un horaire / pgxsaisir

Quel est le problème de la procédure suivante dont le but est de réaliser la saisie d'un moment (heures-minutes-secondes) ?

```
public static void saisirHMS(int hr, int mm, int ss)
{
  Scanner input = new Scanner(System.in);
  System.out.print("hr? ");
  hr = input.nextInt();
  System.out.print("mn? ");
  mm = input.nextInt();
  System.out.print("ss? ");
  ss = input.nextInt();
}
```

Solution simple

Le problème vient du passage de paramètre : ici c'est un passage par valeur alors qu'il faut faire un passage par référence.

Écrivez une opération `saisirhms(...)` qui résout le problème.

Validez votre opération avec la solution.

Solution Java @[pgxsaisir.java]

```
/**
  Procédure
  @param[out] hr - un entier
  @param[out] mm - un entier
  @param[out] ss - un entier
*/

public static void saisirhms(int[] hr, int[] mm, int[] ss)
{
  Scanner input = new Scanner(System.in);
  System.out.print("hr mn ss? ");
  hr[0] = input.nextInt();
  mm[0] = input.nextInt();
  ss[0] = input.nextInt();
}
```

Écrivez alors un programme qui teste votre procédure.

Testez.

Validez votre programme avec la solution.

Solution Java @[pgxsaisir.java]

```java
public static void main(String[] args)
{
  int[] h = {-1};
  int[] m = {-1};
  int[] s = {-1};
  saisirHMS(h[0],m[0],s[0]);
  System.out.println(h[0] + " " + m[0] + " " + s[0]);

  saisirhms(h,m,s);
  System.out.println(h[0] + " " + m[0] + " " + s[0]);
}
```

2 Appliquer le cours

2.1 Autour de la procédure permuter2i / pgpermuter2i

Écrivez le **profil** d'une procédure `permuter2i(a,b)` qui échange les contenus de deux entiers `a` et `b`.

Orientation

Les paramètres formels `a` et `b` sont des paramètres mixtes **Donnée/Résultat**. En effet, ils ont des valeurs **avant** l'appel et ils seront **modifiés** lors de l'échange.

Solution Paramètres

Modifiés : Les entiers `a` et `b`

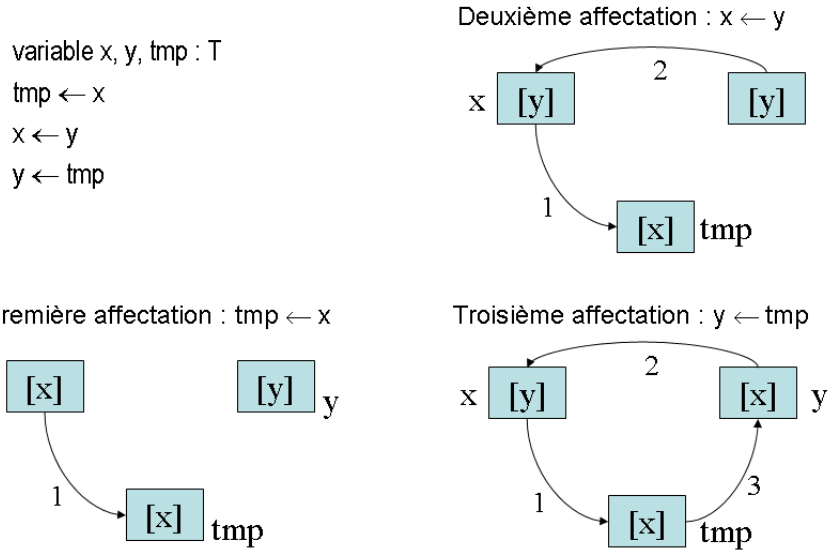
Écrivez le corps de la procédure.

Rappel de cours

Pour permuter deux variables, il faut passer par une variable intermédiaire.

Solution simple

On a :



Validez votre procédure avec la solution.

Solution Java

```
/**
  Permute les valeurs de deux entiers
  @param[in,out] a - un entier
  @param[in,out] b - un entier
*/

public static void permuter2i(int[] a, int[] b)
{
  int tmp = a[0];
  a[0] = b[0];
  b[0] = tmp;
}
```

Écrivez une procédure `decroitre2i(a,b)` qui classe **deux** entiers `a` et `b` par ordre **décroissant**, c.-à-d. qu'à l'issue de la procédure, `a` doit contenir le plus grand entier et `b` le plus petit de `(a,b)`.

Solution Paramètres

Modifiés : Les entiers `a` et `b`

Validez votre procédure avec la solution.

Solution Java

```
/**
  Ordonne deux entiers par ordre décroissant
  @param[in,out] a - un entier
  @param[in,out] b - un entier
  @post a >= b
*/

public static void decroitre2i(int[] a, int[] b)
{
  if (a[0] < b[0])
  {
    permuter2i(a,b);
  }
}
```

Déduisez une procédure `decroitre3i(a,b,c)` qui classe **trois** entiers `a`, `b` et `c` par ordre décroissant, en appelant **trois** fois la procédure `decroitre2i` :

- Classez `a` et `b` en ordre décroissant.
- Puis classez `b` et `c` en ordre décroissant.
- Puis classez `a` et `b` en ordre décroissant.

Solution simple

En effet après les deux premiers classements, `c` contiendra le plus petit de `(a,b,c)`, et à l'issue du dernier classement, `a` contiendra le plus grand de `(a,b,c)`. Par conséquent, `b` contiendra le médian de `(a,b,c)`. **Attention,** ici aussi les trois paramètres formels `a`, `b` et `c` sont des paramètres mixtes.

Validez votre procédure avec la solution.

Solution Java

```
/**
  Ordonne trois entiers par ordre décroissant
  @param[in,out] a - un entier
  @param[in,out] b - un entier
  @param[in,out] c - un entier
  @post a >= b >= c
*/

public static void decroitre3i(int[] a, int[] b, int[] c)
{
  decroitre2i(a,b);
  decroitre2i(b,c);
  decroitre2i(a,b);
}
```

Écrivez un programme qui teste vos procédures comme dans cet exemple d'exécution :

```
Trois entiers? 2 -9 6
==> Apres decroitre2i(n1,n2) 2 -9
==> Apres decroitre3i(n1,n2,n3) 6 2 -9
```

Testez.

Validez votre programme avec la solution.

Solution Java @[pgpermuter2i.java]

```
public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  int[] n1 = new int[1], n2 = new int[1], n3 = new int[1];
  System.out.print("Trois entiers? ");
  n1[0] = input.nextInt();
  n2[0] = input.nextInt();
  n3[0] = input.nextInt();

  decroitre2i(n1,n2);
  System.out.println("==> Apres decroitre2i(n1,n2) " + n1[0] + " " + n2[0]);

  decroitre3i(n1,n2,n3);
```

```
  System.out.println("==> Apres decroitre3i(n1,n2,n3) " + n1[0] + " " + n2[0] + " " +
    n3[0]);
}
```

Écrivez le **profil** d'une procédure `echanger3i(a,b,c)` qui permute circulairement, vers la gauche, les contenus des entiers `a`, `b` et `c`.

Orientation Paramètres

Modifiés : Les entiers `a`, `b` et `c`

Écrivez le corps de la procédure de sorte que le contenu de `a` va dans `c`, celui de `b` dans `a` et celui de `c` dans `b`) en utilisant **deux** fois la procédure `permuter2i`.
(Faites attention à la position des paramètres !)

Aide simple

Il y a plusieurs solutions.

Validez votre procédure avec la solution.

Solution Java

```
/**
  Permute circulairement les valeurs de trois entiers via permuter2i
  @param[in,out] a - un entier
  @param[in,out] b - un entier
  @param[in,out] c - un entier
*/

public static void echanger3i(int[] a, int[] b, int[] c)
{
  permuter2i(a,c);
  permuter2i(a,b);
}
```

2.2 Procédure quorest / pgquorest

Écrivez le **profil** d'une procédure `quorest(a,b,qt,rt)` qui calcule le quotient dans `qt` (entier) et le reste dans `rt` (entier) de la division entière d'un entier `a` par un entier `b`.

Solution Paramètres

Entrants : Les entiers `a` et `b`

Sortants : Les entiers `qt` et `rt`

Propriété

La **division euclidienne** de a par b est définie par :

$$\forall a, b \in \mathbb{N} : a = q \times b + r \text{ et } 0 \leq r < b$$

q est le quotient et r le reste de la division entière.

Écrivez le corps de la procédure.

Validez votre procédure avec la solution.

Solution Java

```
/**
  Quotient et reste de la division entière
  @param[in] a - un entier
  @param[in] b - un entier
  @param[out] qt - quotient de a par b
  @param[out] rt - reste de a par b
*/

public static void quorest(int a, int b, int[] qt, int[] rt)
{
  qt[0] = a / b;
  rt[0] = a % b;
}
```

Écrivez un programme qui saisit deux entiers puis teste votre procédure comme dans l'exemple d'exécution :

```
Deux entiers? 13 3
13 = 3 * 4 + 1
```

Testez.

Validez votre programme avec la solution.

Solution Java @[pgquorest.java]

```java
public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  int n1, n2;
  System.out.print("Deux entiers? ");
  n1 = input.nextInt();
  n2 = input.nextInt();
  int[] qt = new int[1], rt = new int[1];
  quorest(n1,n2,qt,rt);
  System.out.println(n1 + " = " + n2 + " * " + qt[0] + " + " + rt[0]);
}
```