Sommaire

Série 3 : Les Sous-Programmes : Procédure & Fonctions

Exercice 1 : Passage de Paramètres

Exécuter le programme suivant :

```
01 Program Exo_1;
02 Uses wincrt ;
03 var
04    a, b, c : integer;{Variables Globales du programme}
05
06 Procedure Proc1(x:integer ; y:integer ; s:integer) ;
07  Begin
08    s := x+y ;
09  End;
10
11 Procedure Proc2(x:integer ; y:integer ; var s:integer);
12  Begin
13    s := x+y;
14  End;
15
16 BEGIN {Début du Programme Principal}
17    a:=10; b:=5; c:=0;
18    Proc1(a, b, c);
19    Writeln('La somme est : ', c);
20
21    a:=10; b:=5; c:=0;
22    Proc2(a, b, c);
23    Writeln('La somme est : ', c);
24 END. {Fin du Programme Principal}
```

La partie déclaration — On a déclaré trois variables globales et deux procédures

Le Corps du Programme — La partie instructions

L'appel à la procédure Proc1, en transmettant les paramètres : a, b et c

L'appel à la procédure Proc2, en transmettant les paramètres : a, b et c

- C'est quoi la différence entre les deux procédures *Proc1* et *Proc2* ?

- Quels sont les paramètres à passage par valeur et ceux à passage par variable ?

- Quels sont les paramètres formels des deux procédures ?

- Et quels sont les paramètres effectifs ?

Solution

1- La différence entre les deux procédures *Proc1* et *Proc2* réside dans le troisième paramètre **s** . Dans Proc2, le paramètre **s** est défini en utilisant le mot clé **var**. Par contre, dans Proc1 il est défini sans utilisation du mot clé **var**. Ceci aura un effet dans l'exécution des deux procédures, c'est quoi cette effet ? (voir les autres réponses)

2- Les paramètres à passage par valeur et ceux à passage par variable :

Pour la procédure Proc1 :

- Passage par Valeur : les paramètres x, y et s
- Passage par Variable: aucun paramètre

Pour la procédure Proc2 :

- Passage par Valeur : les paramètres x et y
- Passage par Variable : les paramètres s

La définition de type de passage de paramètres (par valeur ou par variable) se fait dans la partie déclaration, où on déclare les procédures ou les fonctions. Dans le programme précédent, la partie déclaration se situe entre les lignes *03* et *14*.

3, 4- Les paramètre formels et les paramètres effectifs

Pour la procédure Proc1 :

- Paramètres formels : x, y et s
- Paramètres effectifs : a, b et c (dans l'appel de la ligne n° 18)

Pour la procédure Proc2 :

- Paramètres formels : x, y et s
- Paramètres effectifs : a, b et c (dans l'appel de la ligne n° 22)

Les paramètres formels sont les paramètres utilisés pour déclaré un sous-programme (procédure ou fonction). Donc pour chercher les paramètres formels d'un sous programme, il faut voir dans la partie déclaration.

Et pour les paramètres effectifs, il faut regarder dans la partie du corps du programme (partie instruction), pour d'éventuels appels à un sous programme. Pour chaque appel d'une procédure ou d'une fonction, on doit indiquer les paramètres effectifs. Par exemple, dans la ligne n°22, on a l'instruction : ***Proc2(a, b, c) ;*** ce qui signifie, l'appel à la procédure Proc2 avec les paramètres effectifs a, b et c. Dans ce cas là, la valeur de a sera transmise au paramètre formel x, celle de b sera transmise au paramètre formel y et la valeur de c sera transmise à s.

Remarques :

- Les paramètres effectifs sont liés à l'appel. Si une fonction ou procédure n'est pas appelée, donc elle n'a pas de paramètre effectif.

*- Si un sous programme (procédure ou fonction) est appelé plusieurs fois dans le programme principal, on spécifie les paramètres effectifs par appel (Les paramètres effectif pour le premier appel sont : …., les paramètres effectif pour le deuxième appel sont : …, **etc.**)*

5- Déroulement du programme

Dans cette exercice (et dans cette série), nous avons aussi des sous-programmes définis au sein du programme principal. Donc, il faut séparer entre les variables

globales (variable du programme principal) et les variables locales de chaque sous-programme (fonction ou procédure), comme illustré dans le tableau suivant:

Instructions	***Programme Principal***			***Procédure Proc1***			***Procédure Proc2***		
	a	*b*	*c*	*x*	*y*	*s*	*x*	*y*	*s (var)*
a:=10; b:=5; c:=0;	10	5	0						
Proc1(a, b,c) (L'appel à Proc1)	"	"	"						
=> La trasmission des paramètres s:=x+y				10	5	0 **15**			
write(c) (la valeur de **s** dans **Proc1** n'a pas été retournée à la variable globale **c**)	"	"	**0**						
a:=10; b:=5; c:=0;									
Proc2(a, b,c) (L'appel à Proc2)									
=> La trasmission des paramètres s:=x+y							10	5	0 **15**
write(c) (la valeur de **s** dans **Proc2** a été retournée à la variable globale **c**)	"	"	**15**						

La valeur de S n'est pas retournée à c (Passage par valeur)

La valeur de S est retournée à c (Passage par variable)

Remarques :

1. Dans le tableau précédant, les zones gris signifie que les **sous-programmes** ne sont pas en exécution et les zone jaune signifie que le **programme principal** a fait l'appel à un sous-programme et attends le retour d'appel (la fin du sous-programme)
2. L'appel à la procédure Proc1 – Proc1(a, b, c) – permet de transmettre les valeurs des paramètres effectif a, b et c vers les paramètres formel x, y et s.
3. L'appel à la procédure Proc2 – Proc2(a, b, c) – permet de transmettre les valeurs des paramètres effectif a, b et c vers les paramètres formel x, y et s.
4. La valeur du paramètre formel s de la procédure Proc1 n'a pas été retournée au paramètre effectif c (la variable globale c). (**La transmission par valeur**).
5. La valeur du paramètre formel s de la procédure Proc2 a été retournée au paramètre effectif c (la variable globale c). (**La transmission par variable**).
6. La transmission (le passage) des paramètres par valeur permet de transmettre les valeurs des paramètres effectif vers les paramètres formel (Paramètres d'entrée).
7. La transmission (le passage) des paramètres par variable permet de transmettre les valeurs des paramètres effectif vers les paramètres formel et, à la fin de l'exécution du sous-programme, de retourner les valeurs des paramètres formel vers les paramètre effectifs correspondant (Paramètres d'entrée et de sortie)

Exercice 2 : Appel à une fonction – Transformation Fonction->Procédure

```
01 Program CombinaisonNK;
02 Uses wincrt ;
03 var
04     n, k, c : integer; {Variables Globales du programme}
05
06 Function fact(n:integer):integer ;
07     Var
08           f, i : integer ; {Variables locales de la fonction fact}
09  Begin
10     F:=1;
11     For i:=1 to n do
12           f:=f*i;
13
14     fact:=f; {Une fonction se termine toujours par une affectation}
15  End;
16
17 BEGIN {Début du Programme Principal}
18     Writeln('Donnez la valeur de n et k :');
19     Read(n, k);
20
21     C:= fact(n) div ( fact(k) * fact(n-k) );
22
23     Writeln('La combinaison de k à partir de n = ', c);
24 END. {Fin du Programme Principal}
```

- Dérouler le programme ci-dessus pour ***n = 5*** et ***k = 3***
- Réécrire le programme en transformant le fonction ***fact*** à une procédure ***fact***.

Soit la procédure ***puissance*** définie comme suit :

```
01 Procedure puissance(x:real; n:integer ; var p:real);
02     Var
03           i : integer ; {Variables locales de la procédure puissance}
04  Begin
05     P:=1;
06     For i:=1 to n do
07           p:=p*x;
08  End;
```

- Transformer cette procédure à une fonction.

Solution

1- Le déroulement du programme pour n=5 et k=3

Instructions	***Programme Principal***			***La fonction fact***			
	n	*k*	*c*	*n*	*i*	*f*	*fact*
Read(n, k)	5	3	/				
*C:=fact(n) div (fact(k) * fact(n-k))* Trois appel à la fonction fact. En appliquant l'ordre de Priorité sur l'expression arithmétique : 1- n-k=2 2- fact(k) 3- fact(2) 4- *(fact(k) * fact(n-k))* 5- *fact(n)* 6- *div*	5	3	/				
fact(k) (l'appel à fact avec le paramètre k=3)	5	3					
=> La transmission des paramètres → n=3 *f:=1* *for i=1 f:=f*i → f=1*1 = 1* *for i=2 f:=f*i → f=1*2 = 2* *for i=3 f:=f*i → f=2*3 = 6* *fact := f → **fact = 6***				**3**	/ 1 2 3	/ 1 1 2 6	/ **6**
fact(n-k) (l'appel à fact avec le paramètre n-k=2)							
=> La transmission des paramètres → n=2 *f:=1* *for i=1 f:=f*i → f=1*1 = 1* *for i=2 f:=f*i → f=1*2 = 2* *fact := f → **fact = 2***				**2**	/ 1 2	/ 1 1 2	/ **2**
fact(n) (l'appel à fact avec le paramètre n=5)	5	3					
=> La transmission des paramètres → n=5 *f:=1* *for i=1 f:=f*i → f=01*1 = 1* *for i=2 f:=f*i → f=01*2 = 2* *for i=3 f:=f*i → f=02*3 = 6* *for i=4 f:=f*i → f=06*4 = 24* *for i=5 f:=f*i → f=24*5 = 120* *fact := f → **fact = 120***				**5**	/ 1 2 3 4 5	/ 1 1 2 6 24 120	/ **120**
*On remplace les appels des fonctions par leurs valeurs : C:= 120 div (6 * 2) → C = 120 div 12 = 10* *write (c)*			**10** **10**				

2- Réécriture du programme en transformant la fonction *fact* à une procédure

Pour transformer une fonction à une procédure nous suivons les étapes suivante :

a- Le mot clé ***Function*** devient ***Procedure*** ;

b- On supprime le type de la fonction et on crée un nouveau paramètre passage par variable du même type que la fonction ;

En appliquant les deux étapes a et b, la ligne n° 06 du programme précédent devient :

Procedure fact(n:integer ; **var** m:integer) ;

Le nouveau paramètre ici est m.

c- On remplace l'instruction *<id_fonction>:= <id_variable_resultat> ;* par l'instruction *<id_nouveau_paramètre>:= <id_variable_resultat> ;*

En appliquant l'étape c, nous la ligne N°14 du programme précédent devient :

m := f ;

d- L'appel de chaque fonction, est remplacé par l'appel à la procédure, en ajouter le paramètre supplémentaire. Par conséquent, il faut déclaré pour chaque appel de la fonction une nouvelle variable globale, du même type que la fonction, qui sera utilisée comme paramètre effectif pour l'appel à la nouvelle procédure. Pour le programme précédent, nous déclarons trois variables globales entières f1, f2 et f3 (puisque il y a trois appels à la fonction *fact*) ; on aura les nouvelle instruction suivante :

fact (n, f1) ; fact (k, f2) ; fact (n-k, f3) ;

c- L'appel à la fonction est remplacé par la variable globale qui représente le paramètre effectif correspondant au passage par variable.

```
01 Program CombinaisonNK;
02 Uses wincrt ;
03 var
04     n, k, c : integer; {Variables Globales du programme}
05
06 Procedure fact(n:integer ; var m:integer) ;
07     Var
08          f, i : integer ; {Variables locales de la fonction fact}
09  Begin
10     F:=1;
11     For i:=1 to n do
12          f:=f*i;
13
14     m:=f; {Une fonction se termine toujours par une affectation}
15  End;
16
17 BEGIN {Début du Programme Principal}
18     Writeln('Donnez la valeur de n et k :');
19     Read(n, k);
20
21     fact(n,f1);       fact(k, f2);       fact(n-k, f3);
22     c := f1 div ( f2 * f3 );
23
24     Writeln('La combinaison de k à partir de n = ', c);
25 END. {Fin du Programme Principal}
```

3- La transformation de la procédure puissance à une fonction

Pour transformer une procédure à fonction, nous suivons les étapes illustrées dans le schéma suivant :

```
01  Procedure puissance(x:real; n:integer ; var p:real);
02      Var
03            i : integer ; {Variables locales de la procédure puissance}
04  Begin
05      P:=1;
06      For i:=1 to n do
07            p:=p*x;
08  End;
```

(1) Changer le mot clé procedure par function

(2) Déplacer le paramètre avec passage par variable vers les variables locales de la fonction.

(3) Mettre le type de ce paramètre comme type de la fonction.

```
01  function puissance(x:real; n:integer) : real;
02      Var
03            i : integer ; {Variables locales de la fonction puissance}
04            P:real;
05  Begin
06      P:=1;
07      For i:=1 to n do
08            p:=p*x;
09
10      puissance : p;
11  End;
```

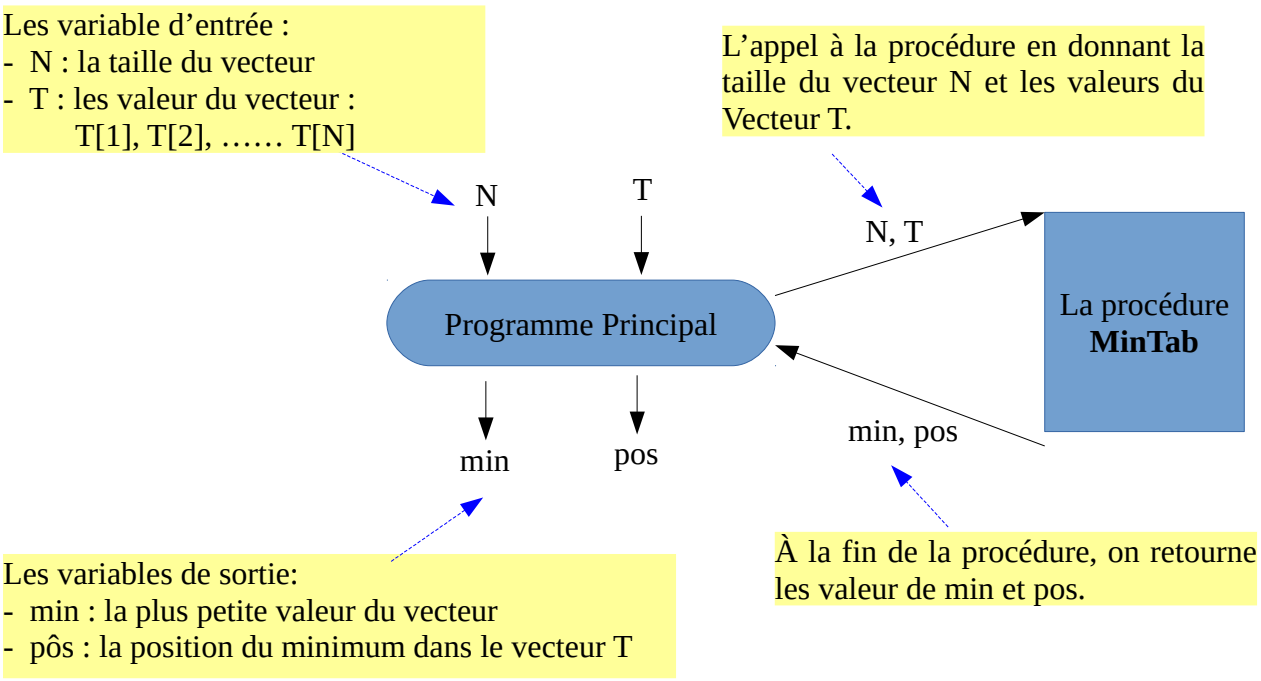
(4) Ajouter, à la fin de la fonction, l'instruction suivante :
<id_fonction> := <id_variable_resultat> ;

Exercice 3 : Minimum et sa position dans un vecteur

Écrire un programme qui lit un **tableau** T de N réels, fait appel à une **procédure** qui détermine le plus petit élément du tableau ainsi que sa position (son rang dans le tableau). Écrire cette procédure et l'insérer dans le programme. Afficher les résultats dans le programme principal.

Solution

Le programme à réaliser, ainsi que la procédure, peut être schématisé comme suit :



D'après le schéma ci-dessus, nous avons :

- Deux variables d'entrées : *N* (la taille du vecteur) et *T* les valeurs des composantes
- Deux variables de sortie : *min* (la valeur du minimum) et *pos* son rang dans *T*.
- La procédure *MinTab* contient 4 paramètres :
  - Deux paramètres d'entrée : un entier et un Tableau.
- Deux paramètres de sorties : un nombre réel et un entier.

Pour utiliser les tableaux (vecteurs ou matrices) comme paramètre dans une procédure, il faut tout d'abord créer un nouveau type, dans la partie déclaration, comme suit :

```
type TABLEAU = Array[1..100] of real;
```

La procédure MinTab sera écrite comme suit :

```
01 Procedure MinTab(m:integer; V:TABLEAU ; var min:real; var pos:integer);
02     Var {Variables locales de la procédure MinTab}
03           i : integer ;
04  Begin
05     min := V[1];
06     pos := 1;
07     for i:=1 to m do
08           if V[i]<min then
09            begin
10                min := V[i];
11                pos := i;
12            end;
13  End;
```

Le programme, avec le sous -programme ***MinTab***, sera écrit comme suit :

```
01 Program Exo3;
02 Uses wincrt ;
03
04 Type TABLEAU : Array[1..100] of real;
05
06 Var {Variables Globales du programme}
07     T : TABLEAU;
08     n,i : integer;
09
10 Procedure MinTab(m:integer; V:TABLEAU; var min:real; var pos:integer);
11     Var {Variables locales de la procédure MinTab}
12           i : integer ;
13  Begin
14     min:=V[1];
15     pos:=1;
16     for i:=1 to m do
17           if V[i]<min then
18            begin
19                min := V[i];
20                pos := i;
21            end;
22  End; {La fin de la procédure MinTab}
23
24 BEGIN {Début du Programme Principal}
25     Writeln('Donnez la taille n :');
26     Read(n);
27     for i:=1 to n do
28           read(T[i]);
29
30     MinTab(n, T, min, pos);
31
32     Writeln('Le minimum = ', min:2:2, '  et sa position = ', pos);
33 END. {Fin du Programme Principal}
```

Les paramètres formes de la procédure **MinTab**

L'appel à la procédure **MinTab** avec les paramètres effectifs : **n, T, min et pos**
Les paramètre formels sont : *m, V, min, Pos*

À l'appel :
- La valeur de **n** est transmise à *m*
- Les valeurs de des composantes du vecteur **T** sont transmis aux composantes de **V**
- la valeur **min** est transmise à *min*.
- la valeur **pos** est transmise à *pos*.

À la fin de la procédure :
- La valeur *min* est retournée à **min**.
- La valeur *pos* est retournée à **pos**.

Remarque

Les paramètres effectifs d'un sous programme peuvent avoir le même nom que les paramètres formels, cependant, aucun lien ne lie ces noms. La seule condition qu'il faut vérifier lors de l'appel à un sous programme est : le nombre de paramètres effectifs égalent le nombre de paramètres formels, et le type de chaque paramètre effectifs est le

même que le paramètre formel correspondant.

Exercice 4 : Développement Limité de l'Exponentiel.

Soit la somme suivante :

e = 1 + 1/1! + 1/2! + 1/3! + …. + 1/n!

Écrire un programme PASCAL, qui fait appel à une fonction FACT pour calculer le factoriel d'un entier n, pour calcule la somme e. Afficher le résultat dans le programme principal.

Solution

Le programme à écrire peut être schématisé comme suit :