TP – Informatique 2

Corrigé de la Série de TP N°3 – Sous-Programmes : Procédures et Fonctions

Exercice N°1 :

1) Exécuter le programme pour k=3 et m=3:

```
1  Program CalculPuissance;
2  Var k: real;     { La base }
3      m: integer;  { L'exposant }
4      result: real; { Le résultat de k^m }
5  Procedure Calc_Puissance(k: real; m: integer; Var P: real);
6    Var i: integer;
7   Begin
8      If m = 0 then
9        P := 1  { Tout nombre élevé à la puissance 0 donne 1 }
10     Else
11    Begin
12       P := 1;
13       For i := 1 to m do
14       P := P * k;  { On multiplie P par k, m fois }
15    End;
16  End;
17 Procedure Affich_Resultat(k: real; m: integer; result: real);
18   Begin
19     WriteLn('Le resultat de ', k:0:2, '^', m, ' est: ', result:0:2);
20   End;
21 Begin
22   Write('Entrez la base (k): ');
23   ReadLn(k);
24   Write('Entrez l''exposant (m): ');
25   ReadLn(m);
26   Calc_Puissance (k, m, result);  { Calculer la puissance }
27   Affich_Resultat (k, m, result); { Afficher le résultat }
28 End.
29
```

```
MyPascal V1.20.5 (Exécution) C...
Entrez la base (k): 3
Entrez l'exposant (m): 3
Le resultat de 3.00^3 est: 27.00
```

Exécution du programme

2) Quels sont les paramètres formels et effectifs.

Paramètres formels	Paramètres effectifs
Les **paramètres formels** sont ceux qui apparaissent dans la signature des procedures ou dans les fonctions: ***Les paramètres formels sont :*** 1. **Procédure Calc_Puissance** : ➢ ***Procedure*** *Calc_Puissance(k: real; m: integer; Var P: real);* – k: real : la base (de type réel). – m: integer : l'exposant (de type entier). – P: real : le résultat de la puissance (de type réel) qui est passé par référence (indiqué par le **mot-clé Var**). 2. **Procédure Affich_Resultat** : ➢ ***Procedure*** *Affich_Resultat(k: real; m: integer; result: real);* – k: real : la base (de type réel). – m: integer : l'exposant (de type entier). – result: real : le résultat de la puissance (de type réel).	Les **paramètres effectifs** sont les valeurs ou variables passées aux procédures ou aux fonctions lors de leur appel dans le programme principal: ***Les paramètres effectifs sont :*** 1. **Procédure Calc_Puissance** : ➢ ***Calc_Puissance****(k, m, result);* – k : la variable contenant la base (réel). – m : la variable contenant l'exposant (entier). – result : la variable dans laquelle le résultat de la puissance sera stocké. Cette variable est passée par référence 2. **Procédure Affich_Resultat** : ➢ ***Affich_Resultat****(k, m, result);* – k : la variable contenant la base (réel). – m : la variable contenant l'exposant (entier). – result : la variable contenant le résultat de la puissance (réel).

3) **Quels sont les paramètres à passage par valeur et ceux à passage par variable de la fonction puissance?**

Les paramètres à passage par valeur et ceux à passage par variable sont:

- Les paramètres passés par valeur sont ceux qui sont simplement copiés lorsqu'ils sont passés à une procédure ou une fonction.
  - ✓ **k: real** : La base (un réel).
  - ✓ **m: integer** : L'exposant (un entier).

- Les paramètres passés par ***variable (ou par référence***) sont ceux qui permettent à la ***procédure ou fonction*** de modifier la valeur de la variable d'origine dans le programme appelant. Cela se fait grâce au **mot-clé Var** dans ***la déclaration du paramètre.***
  - ✓ **P: real** dans la procédure ***Calc_Puissance.***

4) **Déroulement du programme pour k=3 et m=3.**

Instructions	Paramètres / Variables					Affichage
	k	m	i	p	result	
Write('Entrez la base (k): ');	/	/	/	/	/	Entrez la base (k):
ReadLn(k);	**3**	/	/	/	/	/
Write('Entrez l''exposant (m): ');	**3**	/	/	/	/	Entrez l''exposant (m):
ReadLn(m);	**3**	**3**	/	/	/	/
(l'appel à la procédure Calc_Puissance (k, m, result);	**3**	**3**	/	/	/	/

If m = 0 then False
P := 1
Else
Begin
P := 1;
For i := 1 to m do
	3	**3**	/	/	1	/

For i := 1 P := P * k; P := 1 * 3; P:=3;	**3**	**3**	**1**	3	**3**	/
For i := 2 P := P * k; P := 3 * 3; P:=9;	**3**	**3**	**2**	9	**9**	/
For i := 3 P := P * k; P := 9 * 3; P:=27;	**3**	**3**	**3**	9	**27**	/
(l'appel à la procédure Affich_Resultat (k, m, result);	**3**	**3**	**3**	9	**27**	/

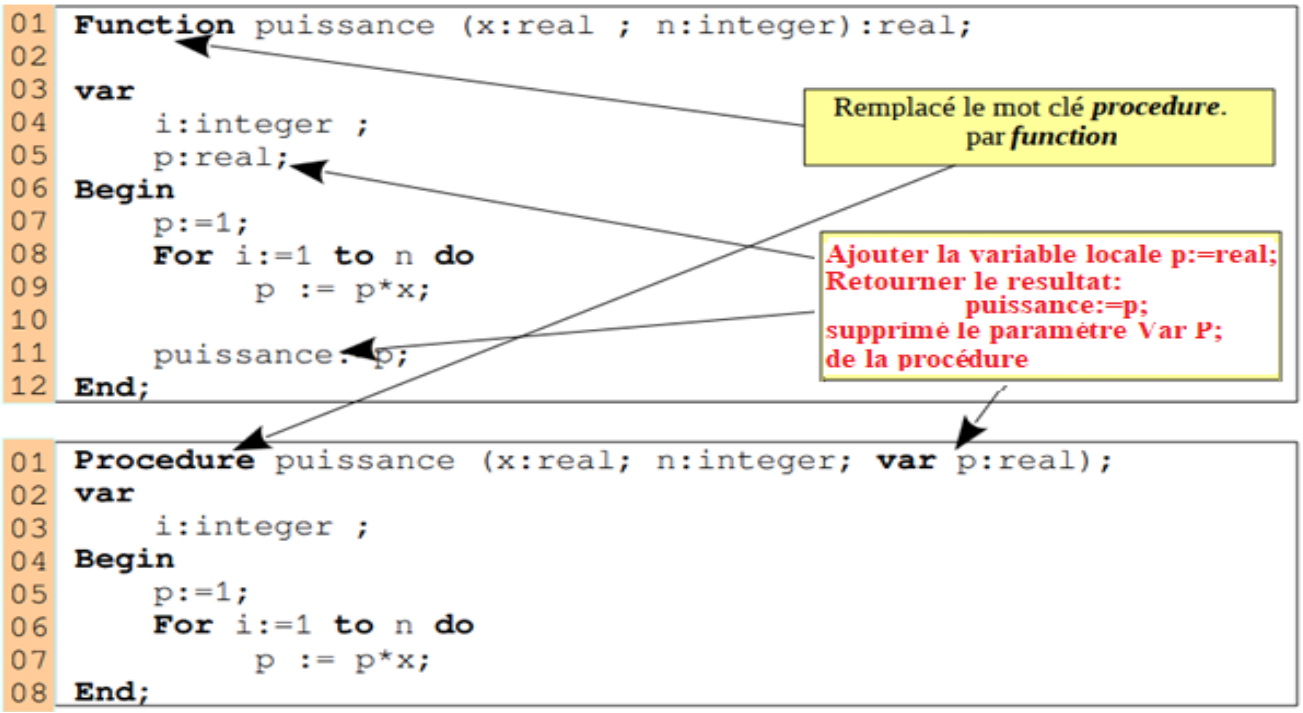
Begin
WriteLn('Le resultat de ', k:0:2, '^', m, ' est: ', result:0:2);
End;
	3	**3**	**3**	9	**27**	**Le resultat de 3.00^3 est: 27.00**

5) Transformer la procédure Calc_puissance en une fonction Calc_puissance :

1) Pour transformer la procédure en une fonction :

Lors de la transformation d'une procédure en une fonction, il faut **retirer** l'utilisation du paramètre passé par **VAR** (qui est habituellement utilisé pour la sortie des résultats dans la procédure) et **retourner le résultat directement** dans la fonction. La fonction aura donc **le même nombre de paramètres** que la procédure, mais **aucun paramètre ne sera passé par VAR**.

Au lieu de modifier un paramètre via **VAR**, la fonction renverra une **valeur** qui sera **le résultat calculé**.




```
Function puissance (x:real ; n:integer):real;

var
    i:integer ;
    p:real;
Begin
    p:=1;
    For i:=1 to n do
        p := p*x;

    puissance:=p;
End;
```

```
Procedure puissance (x:real; n:integer; var p:real);
var
    i:integer ;
Begin
    p:=1;
    For i:=1 to n do
        p := p*x;
End;
```

Dans notre cas la fonction est :

```
Function Calc_Puissance(k: real; m: integer ): real;
  Var  i: integer;
       P: real;
 Begin
    If m = 0 then
      P := 1  { Tout nombre élevé à la puissance 0 donne 1 }
    Else
   Begin
     P := 1;
     For i := 1 to m do
     P := P * k;  { On multiplie P par k, m fois }
   End;
    Calc_Puissance := P;  { Retourner le résultat calculé }
 End;
```

Pour insérer maintenant cette FONCTION dans le programme, trois étapes sont nécessaires :

1. Retirer le paramètre VAR :

- Dans la procédure Calc_Puissance, le paramètre de sortie P était passé par VAR. Pour transformer cette procédure en fonction, il faut retirer le mot-clé VAR et retourner le résultat directement depuis la fonction.

2. Utiliser Calc_Puissance := P pour retourner le résultat :

- Dans une fonction, au lieu de modifier un paramètre passé par VAR, il faut retourner le résultat de la fonction en utilisant Calc_Puissance := P à la fin de la fonction, où P est la variable qui contient le calcul de la puissance.

3. Modifier l'appel de la procédure en appel de fonction :

- Dans le programme principal, remplacer l'appel de la **procédure** par l'appel de la **fonction**. Au lieu de ***Calc_Puissance(k, m, result);,*** on utilise maintenant ***result := Calc_Puissance(k, m);,*** car la fonction renvoie directement une valeur et elle est assignée à la variable result.

En appliquant ces étapes, nous obtenons le programme suivant :

Programme complet en utilisant la procédure

```
Program CalculPuissance;
Var  k: real;      { La base }
       m: integer;   { L'exposant }
       result: real; { Le résultat de k^m }
Function Calc_Puissance(k: real; m: integer ): real;
   Var  i: integer;
         P: real;
  Begin
       If m = 0 then
          P := 1  { Tout nombre élevé à la puissance 0 donne 1 }
       Else
      Begin
           P := 1;
         For i := 1 to m do
            P := P * k;  { On multiplie P par k, m fois }
      End;
      Calc_Puissance := P;  { Retourner le résultat calculé }
  End;
    Procedure Affich_Resultat(k: real; m: integer; result: real);
        Begin
            WriteLn('Le resultat de ', k:0:2, '^', m, ' est: ', result:0:2);
        End;
  Begin
      Write('Entrez la base (k): ');
      ReadLn(k);
      Write('Entrez l''exposant (m): ');
      ReadLn(m);
  result := Calc_Puissance(k, m); { Appel de la fonction calc_puissance }
  Affich_Resultat (k, m, result); { Appel de la procédure Affich_Résultat }
End.
```

Compilation et exécution du programme

```pascal
Program CalculPuissance;
Var k: real;     { La base }
    m: integer;  { L'exposant }
    result: real; { Le résultat de k^m }
Function Calc_Puissance(k: real; m: integer ): real;
  Var i: integer;
      P: real;
 Begin
    If m = 0 then
      P := 1  { Tout nombre élevé à la puissance 0 donne 1 }
    Else
   Begin
     P := 1;
     For i := 1 to m do
     P := P * k;  { On multiplie P par k, m fois }
   End;
    Calc_Puissance := P;  { Retourner le résultat calculé }
 End;
Procedure Affich_Resultat(k: real; m: integer; result: real);
   Begin
    WriteLn('Le resultat de ', k:0:2, '^', m, ' est: ', result:0:2);
   End;
Begin
 Write('Entrez la base (k): ');
 ReadLn(k);
 Write('Entrez l''exposant (m): ');
 ReadLn(m);
 result := Calc_Puissance(k, m); { Appel de la fonction calc_puissance }
 Affich_Resultat (k, m, result); { Appel de la procédure Affich_Résultat }
End.
```

```
MyPascal V1.20.5 (Exé...
Entrez la base (k): 3
Entrez l'exposant (m): 3
Le resultat de 3.00^3 est: 27.00
```

Exercice 2 :

1) Exécuter le programme:

Programme	Exécution du programme

```pascal
Program Somme;
        { Déclaration des procédures }
Procedure Sous_Prog1(x, y: real; s: real);
  Begin
    s := x + y;
  End;
Procedure Sous_Prog2(x, y: real; var s: real);
  Begin
    s := x + y;
  End;
              { Programme principal }
Var
  a, b, c: real;
Begin
  a := 10; b := 5;c := 0;
  Sous_Prog1(a, b, c);
  WriteLn('La somme S-Prog1 est : ', c:6:2);
  a := 10;b := 5;c := 0;
  Sous_Prog2(a, b, c);
  WriteLn('La somme S-Prog2 : ', c:6:2);
End.
```

```pascal
Program Somme;
        { Déclaration des procédures }
Procedure Sous_Prog1(x, y: real; s: real);
  Begin
    s := x + y;
  End;
Procedure Sous_Prog2(x, y: real; var s: real);
  Begin
    s := x + y;
  End;
              { Programme principal }
Var
  a, b, c: real;
Begin
  a := 10; b := 5;c := 0;
  Sous_Prog1(a, b, c);
  WriteLn('La somme S-Prog1 est : ', c:5:2);
  a := 10;b := 5;c := 0;
  Sous_Prog2(a, b, c);
  WriteLn('La somme S-Prog2 : ', c:5:2);
End.
```

```
MyPascal V1.20.5 (...
La somme S-Prog1 est :  0.00
La somme S-Prog2 : 15.00
```

Le programme comporte deux sous - programmes **sous_prog1** et **sous_prog2** qui ont pour objectif de calculer la somme de deux nombres de **type réels** passés en paramètres et de stocker le résultat dans une variable **S** également de **type réels** . Le programme principal utilise ces **deux sous - programmes** pour calculer la somme de deux nombres ***a et b*** et stocker le résultat dans une variable **C** d'abord **sous_prog1** , puis **sous_prog2** .

2) Quelle est la différence entre les deux procédures sous_prog1 et sous_prog2 ?

La différence entre **Sous_Prog1** et **Sous_Prog2** réside dans la manière dont le paramètre **s** est passé à chaque procédure.

- **Sous_Prog1** utilise un paramètre de type real passé **par valeur**, ce qui signifie que toute modification de **s** à l'intérieur de la procédure n'affecte pas la variable **c** dans le programme principal. En effet, une copie locale de **s** est créée dans la procédure, et toute modification de cette copie reste locale à la procédure.
- **Sous_Prog2**, en revanche, utilise un paramètre de type real passé **par référence** (**grâce au mot-clé var**), ce qui permet à la procédure de modifier directement la valeur de la variable **s** dans le programme principal. En d'autres termes, **s** fait référence à la variable **c** dans main, et toute modification de **s** affecte directement c.

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

Paramètre/Procédure	Procédure 1	Procédure 2
Paramètres à passage par valeur	`x`, `y`, `s`	`x`, `y`
Paramètres à passage par variable	Aucun paramètres	`s`

4) Quels sont les paramètres formels des deux procédures ?
Les **paramètres formels** des deux procédures :

Paramètre/Procédure	Procédure 1	Procédure 2
Paramètres formels	`x`, `y`, `s`	`x`, `y`, `s`

5) Quels sont les paramètres effectifs ?

Les **paramètres effectifs** des deux procédures :

Paramètre/Procédure	Procédure 1	Procédure 2
Paramètres effectifs	`a`, `b`, `c`	`a`, `b`, `c`

Rappel :

Les **paramètres formels** sont les paramètres utilisés dans la **déclaration des procédures** et **fonctions**. Ce sont les variables qui apparaissent dans la signature de la procédure ou de la fonction, et qui servent à définir leur structure interne.

En revanche, les **paramètres effectifs** sont les valeurs ou les variables réelles passées lors de l'appel de ces procédures et fonctions. Ces valeurs sont transmises aux paramètres formels pour permettre l'exécution de la procédure ou de la fonction.

Exercice 03:

Programme en pascal (comparer la somme de deux vecteurs)

```
Program ComparerVecteurs;
Var
  V1, V2: array[1..100] of Integer;
  sommeV1, sommeV2, N, i: Integer;
Function SommeVecteur(V: array of Integer; N: Integer): Integer;
Var
  i, somme: Integer;
Begin
somme := 0;
  For i := 0 to N do
    somme := somme + V[i];
    SommeVecteur := somme;
End;
Procedure ComparerSommes(sommeV1, sommeV2: Integer);
  Begin
     If sommeV1 > sommeV2 then
        WriteLn('La somme de V1 est plus grande que celle de V2.')
     else if sommeV1 < sommeV2 then
        WriteLn('La somme de V2 est plus grande que celle de V1.')
     else
        WriteLn('Les sommes de V1 et V2 sont égales.');
End;
Begin
  Write('Entrer la taille des vecteurs (maximum 100) : ');
  ReadLn(N);
  WriteLn('Entrer les éléments du vecteur V1 :');
  For i := 1 to N do
  Begin
    ReadLn(V1[i]);
  End;

  WriteLn('Entrer les éléments du vecteur V2 :');
  For i := 1 to N do
  Begin
    ReadLn(V2[i]);
  End;
  sommeV1 := SommeVecteur(V1, N);  { Calculer la somme de V1 }
  sommeV2 := SommeVecteur(V2, N);  { Calculer la somme de V2 }
  WriteLn('Somme des éléments de V1 : ', sommeV1);
  WriteLn('Somme des éléments de V2 : ', sommeV2);
    ComparerSommes(sommeV1, sommeV2);  { Comparer les sommes des deux vecteurs }
End.
```

A) Compilation et Exécution du programme

```pascal
Program ComparerVecteurs;
Var
 V1, V2: array[1..100] of Integer;
 sommeV1, sommeV2, N, i: Integer;
Function SommeVecteur(V: array of Integer; N: Integer): Integer;
  Var
   i, somme: Integer;
  Begin
   somme := 0;
    For i := 0 to N do
     somme := somme + V[i];
     SommeVecteur := somme;
   End;
{ Sous-programme pour comparer les sommes des vecteurs V1 et V2 }
  Procedure ComparerSommes(sommeV1, sommeV2: Integer);
   Begin
    If sommeV1 > sommeV2 then
      WriteLn('La somme de V1 est plus grande que celle de V2.')
    else if sommeV1 < sommeV2 then
      WriteLn('La somme de V2 est plus grande que celle de V1.')
    else
      WriteLn('Les sommes de V1 et V2 sont égales.');
   End;
  Begin
    Write('Entrer la taille des vecteurs (maximum 100) : ');
    ReadLn(N);
    WriteLn('Entrer les éléments du vecteur V1 :');
   For i := 1 to N do
    Begin
     ReadLn(V1[i]);
    End;
    WriteLn('Entrer les éléments du vecteur V2 :');
    For i := 1 to N do
     Begin
     ReadLn(V2[i]);
   End;
 { Calcul des sommes des vecteurs }
 sommeV1 := SommeVecteur(V1, N); { Calculer la somme de V1 }
 sommeV2 := SommeVecteur(V2, N); { Calculer la somme de V2 }
 WriteLn('Somme des éléments de V1 : ', sommeV1);
 WriteLn('Somme des éléments de V2 : ', sommeV2);
 { Comparer les deux sommes }
 ComparerSommes(sommeV1, sommeV2);
End.
```

```
MyPascal V1.20.5 (Exécution) C:\Users\M...
Entrer la taille des vecteurs (maximum 100) : 3
Entrer les éléments du vecteur V1 :
3
2
6
Entrer les éléments du vecteur V2 :
1
2
4
Somme des éléments de V1 : 11
Somme des éléments de V2 : 7
La somme de V1 est plus grande que celle de V2.
```

B) Quelle est la différence entre une procédure et une fonction?

Une fonction est généralement utilisée pour effectuer un calcul et retourner une valeur. Dans ce cas, nous utiliserons des fonctions pour calculer les sommes des vecteurs.

En revanche la procédure est utilisée pour effectuer des actions sans retourner de valeur. Ici, elle sera employée pour comparer les deux sommes et afficher le résultat de la comparaison.

Exercice 04:

En utilisant la fonction **Puissance** de l'exo1 et une autre fonction **fact** qui calcule le factoriel d'un entier n tel que $n! = 1 \times 2 \times\times 4 \times \ldots . \times n$„ Ecrire un programme Pascal pour calculer la valeur approximative de $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$.

PROGRAMME EN PASCAL : Calcul de fonction exponentiel

```
Program Calcul_Exponentiel;
Var i, n: integer; {Variables globales du programme principal}
X, S: real;
FUNCTION Fact (m: integer): integer;
var j, f: integer; {Variables locales de la fonction Fact}
begin
f:=1;
For j:=1 to m do
f:= f*j;
Fact := f;
end;
FUNCTION Puissance (k: real; m: integer) : real;
Var j: integer; {Variables locales de la fonction Puiss}
P:real;
Begin
P:=1;
For j:=1 to m do
P:=P*k;
Puissance := P; End;
BEGIN {Début du programme principal}
write('Introduire les valeurs de n et x : ');
read(n, x);
S:=1;
for i:=1 to n do
S:=S+ Puissance(x, i) / Fact(i) ; {Calcul de S en appelant
les deux fonctions Fact et Puiss}
write('La valeur approximative de Exp (x) = ', S:6:2);
END. {Fin du programme principal}
```

```
MyPascal V1.20.5 (Exécution) C:\...
Introduire les valeurs de n et x : 3
3
La valeur approximative de Exp (x) =  13.00
```