Université A. Mira de Bejaia
Faculté de Technologie
Département de Technologie

Année universitaire 2022/2023
1ère Année Technologie
Semestre 2

TP Informatique 2

Corrigé de la série de TP N°2 – Exercices supplémentaires sur les procédures et fonctions

Exercice N°1 :

- Écrire une fonction « Max » qui calcule le maximum entre deux nombres réels X et Y.
- En utilisant la fonction « Max », écrire un programme Pascal permettant de calculer et afficher le maximum entre trois nombres réels X, Y et Z.

Solution :

Algorithme

```
Algorithme Exo_supp1;
Variables x, y, z, m2, m3: réel;
Fonction Max(x, y : reél): reél;
Début
 Si x>=y alors
   Max ← x
 Sinon
  Max ← y;
 Finsi;
Fin;

Début
 Écrire( ' Donnez trois valeurs réelles : ');
 Lire(x, y, z);
 m2 ← Max(x, y);
 m3 ← Max(m2,z);
 Écrire( ' Le maximum est :', m3:0:1);
Fin.
```

Programme Pascal

```
Program Exo_supp1;
Var x, y, z, m2, m3: real;
Function Max(x, y : real): real;
Begin
  if x>=y then
    Max:=x
  else
    Max:=y;
End;

Begin
  writeln( ' Donnez trois valeurs réelles : ');
  read(x, y, z);
  m2:=Max(x, y);
  m3:=Max(m2,z);
  writeln( ' Le maximum est :', m3:0:1);
End.
```

Exercice N°2 :

Soit la séquence d'instructions suivante qui permet de calculer la somme « S » des diviseurs d'un nombre entier « X ».

```
S :=0 ;
For i := 1 to X do
  If (X mod i) = 0 then
        S := S+i;
```

1) Écrire un sous-programme fonction qui s'appelle « somme_div » et qui calcule la somme « S » des diviseurs d'un nombre entier « X ».
2) Insérer cette fonction dans un programme Pascal qui lit un entier « X », fait appel à la fonction « somme_div » et affiche la somme de ses diviseurs « S ».
3) Réécrire le programme en transformant la fonction « somme_div » en procédure « somme_div ».
4) Modifier les instructions de la fonction pour qu'elle retourne la somme des diviseurs de « X » sauf 1 et lui-même.

Solution :

1) Écrire un sous-programme fonction qui s'appelle « somme_div » et qui calcule la somme « S » des diviseurs d'un nombre entier « X ».

Sous-programme « somme_div »

```
Function somme_div(x:integer):integer;
Var  i, s:integer;
Begin
   s:=0;
  for i:=1 to x do
    if  x mod i =0 then
       s:=s+i ;
       somme_div:=s;
End;
```

2) Insérer cette fonction dans un programme Pascal qui lit un entier « X », fait appel à la fonction « somme_div » et affiche la somme de ses diviseurs « S ».

Programme Pascal

```
Program Exo_supp2;
Var x:integer;  Sx:integer;
Function somme_div(x:integer):integer;
Var  i, s:integer;
Begin
    s:=0;
   for i:=1 to x do
     if  x mod i =0 then
        s:=s+i ;
        somme_div:=s;
End;

Begin
   writeln(' Donner la valeur de l''entier X: ');
   read(x);
   Sx:=somme_div(x);
   writeln(' La somme des diviseurs de ', x, ' est : ', Sx);
End.
```

3) Réécrire le programme en transformant la fonction « somme_div » en procédure « somme_div ».

Programme Pascal avec la fonction « somme_div »

```
Program Exo_supp2;
Var x:integer;  Sx:integer;
Function somme_div(x:integer):integer;
Var  i, s:integer;
Begin
    s:=0;
   for i:=1 to x do
     if  x mod i =0 then
        s:=s+i ;
        somme_div:=s;
End;

Begin
   writeln(' Donner la valeur de l''entier X: ');
   read(x);
   Sx:=somme_div(x);
   writeln(' La somme des diviseurs de ', x, ' est : ', Sx);
End.
```

Programme Pascal avec la procédure « somme_div »

```
Program Exo_supp2;
Var x:integer;  Sx:integer;
Procedure somme_div(x:integer;  var s:integer);
Var  i :integer;
Begin
    s:=0;
   for i:=1 to x do
     if  x mod i =0 then
        s:=s+i ;
End;

Begin
   writeln(' Donner la valeur de l''entier X: ');
   read(x);
   somme_div(x, Sx);
   writeln(' La somme des diviseurs de ', x, ' est : ', Sx);
End.
```

4) Modifier les instructions de la fonction pour qu'elle retourne la somme des diviseurs de « X » sauf 1 et lui-même.

Sous_programme « somme_div »

```
Function somme_div(x:integer):integer;
Var  i, s:integer;
Begin
    s:=0;
   for i:=2 to (x div 2) do
     if  x mod i =0 then
         s:=s+i ;
         somme_div:=s;
End;
```

Exercice N°3 :

- Écrire un programme qui lit un **tableau** T de N réels, fait appel à une **procédure** qui détermine le plus grand 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 :

Programme Pascal

```
Program Exo_supp3;
Type Tab=array[1..50] of real;
Var T: Tab;
    n, i, p : integer;
    max: real;
Procedure Max_PosMax (n:integer; T: Tab; var max:real;  var p:integer);
Var i : integer;
Begin
   max:=T[1];
   p:=1;
   for i:=2 to n do
    if T[i]> max then
    begin
      max:= T[i];
      p:= i;
    end;
End;
Begin
writeln( 'Donner la taille du tableau T :');
read(n);
writeln( 'Donner les éléments du tableau T :');
for i:=1 to n do
read(T[i]);
Max_PosMax(n, T, max, p);
writeln( 'Le maximum est :', max:0:1, ' sa position est : ', p);
End.
```

Exercice N°4 :

- Écrire une procédure « Permuter » qui réalise la permutation entre deux réels X et Y.
- En utilisant la procédure « Permuter », écrire un programme Pascal permettant de permuter les deux diagonales d'une matrice carrée A(n×n) de type réel.

Solution :

1) Le code de la procédure « Permuter » :

```
Procedure Permuter (VAR x, y: real);
Var z:real;
Begin
   Z:=x;
   X:=y;
   Y:=z;
End;
```

2) Le programme Pascal permettant de permuter les deux diagonales :

```
Program exo4 ;
Var
   A : array[1..10,1..10] Of real;
   i,j,n : integer;
Procedure Permuter (Var x, y:real);
Var
   z : real;
Begin
   Z := x;
   X := y;
   Y := z;
End;
Begin
   Write ('introduire le nombre de lignes ou le nombre de colonnes :') ;
   Read(n) ;
   Write ('introduire les ',n*n,' valeurs de la matrice A :') ;
   For i:=1 To n Do
      For j:=1 To n Do
         Read(A[i, j]) ;
{permutation entre la diagonal principale et la diagonale secondaire}
   For i:=1 To n Do
            Permuter ( A[i, i] , A[i , n-i+1] );
{affichage de la matrice A après la permutation}
   For i:=1 To n Do
      Begin
         For j:=1 To n Do
            Write ( A[i, j]:8:2);
         Writeln;
      End;
End.
```

Exercice N°5 :

- Écrire en Pascal une fonction « Nombre_Premier » qui vérifie si un nombre entier N est premier ou non.
  N.B : Un nombre est dit « premier » s'il est divisible uniquement par 1 et lui-même.
- Insérer la fonction « Nombre_Premier » dans un programme Pascal complet permettant de lire une matrice A (n×m) de type entier et de calculer et afficher le **nombre total des nombres premiers** présents dans la matrice A.
- Réécrire le programme précédent en transformant la fonction « Nombre_Premier » en une procédure du même nom.

Solution :

1) Le code de la fonction « Nombre_Premier »

```
Function Nombre_Premier(nb :integer) :Boolean;
Var Test: Boolean;
      i :integer;
Begin
Test:=true;
i:=2;
While (Test=true) and (i <= n div 2) do
Begin
  If (nb mod I = 0) then
       Test:=false;
  i:=i+1;
End;
Nombre_Premier := Test;
End;
```

2) Le programme Pascal permettant de calculer le nombre total de nombres premiers

```
Program exo5 ;
Var
   A : array[1..10,1..10] Of integer;
   i, j, n, m, Nb : integer;
Function Nombre_Premier(nb :integer) : Boolean;
Var
   Test : Boolean;
   i : integer;
Begin
   Test := true;
   i := 2;
   While (Test=true) And (i <= nb Div 2) Do
      Begin
         If (nb Mod i = 0) Then
            Test := false;
         i := i+1;
      End;
   Nombre_Premier := Test;
End;
Begin
   Write ('introduire le nombre de lignes et le nombre de colonnes :') ;
   Read(n, m) ;
   Write ('introduire les ',n*m,' valeurs de la matrice A :') ;
   For i:=1 To n Do
      For j:=1 To m Do
         Read(A[i, j]) ;
{Partie traitement : Nombre total des nombres premiers présents dans A}
   Nb := 0 ;
   For i:=1 To n Do
      For j:=1 To m Do
         If ( Nombre_Premier (A[i, j]) = true) Then
            Nb := Nb+1;
{affichage du résultat}
   Write ('Nombre total des nombres premiers présents dans A = ' , Nb);
End.
```

3) Réécrire le programme précédent en transformant la fonction « Nombre_Premier » en une procédure du même nom.

```
Program exo5_2 ;
Var
   A : array[1..10,1..10] Of integer;
   i, j, n, m, Nb : integer;
   {ajout d'une variable globale pour récupérer
 le résultat de la procédure Nombre_Premier}
   Premier : Boolean;
Procedure Nombre_Premier(nb :integer; Var Test:Boolean);
Var
   i : integer;
Begin
   Test := true;
   i := 2;
   While (Test=true) And (i <= nb Div 2) Do
      Begin
         If (nb Mod i = 0) Then
            Test := false;
         i := i+1;
      End;
End;
Begin
   Write ('introduire le nombre de lignes et le nombre de colonnes :') ;
   Read(n, m) ;
   Write ('introduire les ',n*m,' valeurs de la matrice A :') ;
   For i:=1 To n Do
      For j:=1 To m Do
         Read(A[i, j]) ;
{Partie traitement : Nombre total des nombres premiers présents dans A}
   Nb := 0 ;
   For i:=1 To n Do
      For j:=1 To m Do
         Begin
            Nombre_Premier (A[i, j], Premier);
            If ( Premier = true) Then
               Nb := Nb+1;
         End;
{affichage du résultat}
   Write ('Nombre total des nombres premiers présents dans A = ' , Nb);
End.
```