TP-Initiation à la Programmation

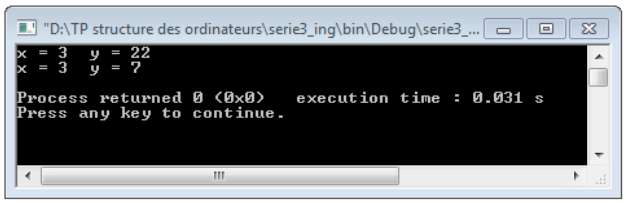
Corrigé de la série de TP N°3 – Sous programmes : Fonctions et Procédures

Exercice N°1 :

1) Exécution du programme :

```c
#include <stdio.h>

int x=3, y=7 ;
void  Proc1 (int A, int *B)
{
    A = A + 1;   *B = 22;
}
void Proc2 (int A, int B)
{
    A = A + 1;   B = 22;
}
int main()
{
    x=3, y=7 ;
    Proc1(x, &y);
    printf("x = %d  y = %d\n", x, y);
    x=3, y=7 ;
    Proc2(x, y);
    printf("x = %d  y = %d\n", x, y);
    return 0;
}
```



2) La différence entre les deux procédures Proc1 et Proc2 :

La différence entre **Proc1** et **Proc2** est que **Proc1** reçoit **y** par adresse (pointeur int *B), donc elle peut modifier réellement **y**, par contre **Proc2** reçoit tous ses paramètres par valeur (copies), donc aucune variable du programme principal n'est modifiée.

3) Les paramètres à passage par valeur et ceux à passage par variable :

Paramètre / Procédure	Proc1	Proc2
Paramètres à passage par valeur	A	A et B
Paramètres à passage par variable	B	Aucun paramètre

4) Les paramètres formels et effectifs des deux procédures :

Type de paramètres / Procédure	Proc1	Proc2
Paramètres formels	A et B	A et B
Paramètres effectifs	x et y	x et y

5) Les variables globales et locales dans le programme :

variables globales	x, y
variables locales	A, B dans Proc1 et Proc2

Remarque:

- ✓ Le **passage par valeur** consiste à transmettre à une fonction une **copie** de la variable, ce qui signifie que toute modification effectuée dans la fonction ne change pas la variable originale. En revanche, le **passage par variable (ou par adresse)** consiste à transmettre l'**adresse mémoire** de la variable à l'aide d'un pointeur, ce qui permet à la fonction d'accéder directement à cette variable et de la modifier réellement. Ainsi, le passage par valeur protège la variable d'origine, tandis que le passage par variable permet sa modification directe.

- ✓ 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.

- ✓ Les **variables globales** sont déclarées en dehors de toute fonction, ce qui signifie qu'elles peuvent être accessibles dans toutes les fonctions du programme, y compris la fonction main, et qu'elles conservent leur valeur pendant toute la durée d'exécution du programme. Les **variables locales** sont déclarées à l'intérieur d'une fonction et ne sont accessibles que dans cette fonction. Elles ne sont ni visibles ni modifiables par les autres fonctions du programme et n'existent que pendant l'exécution de la fonction dans laquelle elles sont déclarées.

6) Les appels Proc1(3,7) et Proc2(3,7) sont-ils corrects, expliquer pourquoi :

- Pour Proc1(3,7) : L'appel est incorrect car le paramètre formel B est en entrée / Sortie : il doit lui correspondre une variable, et non pas une constante.

- Pour Proc2(3,7) : L'appel est correct car les paramètres formels A et B sont des paramètres d'entrée seulement. Par conséquent, ils peuvent leur correspondre des variables ou des constantes.

7) Déroulement du programme :

Instructions	**Variables programme principale**		**Variables procédure Proc1**		**Variables procédure Proc2**		**Affichage**
	x	**y**	**A**	**B**	**A**	**B**	
x=3 ; y=7 ;	3	7					
Proc1(x, **&**y); *Appel à la procédure Proc1 et transmission des paramètres*	3	7 →	3	7			
A = A + 1 ; A = 3 + 1 ; A = 4 ;			**4**	**7**			
*B = 22 ;				**22**			
Proc1 a calculé la valeur de A et B, la valeur de B est retournée à la variable globale y (car B est passé par variable)		**22**					
printf ("x = %d y = %d\n", x, y);	3	**22**	4	22			x= 3 y=22
x=3 ; y=7 ;	3	7					
Proc2 (x, y); *Appel à la procédure Proc2 et transmission des paramètres*	3	7 →			3	7	
A = A + 1; A = 3 + 1; A=4 ;					**4**	7	
B = 22;						**22**	
Proc2 a calculé la valeur de A et B, mais contrairement à Proc1, la valeur de B n'est pas retournée à la variable globale y (car B est passé par valeur)							
printf ("x = %d y = %d\n", x, y);	3	7			3	22	x= 3 y=7

8) Exécuter le programme en donnant le type « réel » à la variable y. Que se passe-t-il ? Pourquoi ?

Nous obtenons **une erreur de compilation** indiquant qu'un appel à la procédure Proc1 est effectué avec un incompatible type de paramètre. Ceci revient au passage d'un paramètre de **type float** à une procédure qui attend un paramètre de **type int***. (Les types mémoire sont incompatibles entre le paramètre effectif (y) et son paramètre formel correspondant (B), donc le compilateur refuse la conversion et génère une erreur.)

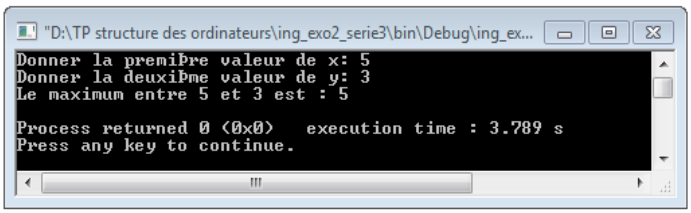
Exercice N°2 :

Programme en langage C

```
#include<stdio.h>
int maximum(int a, int b) {
    if (a>b)
    return a ;
    else
    return b ; }
int main( ) {
    int x, y, results ;
    printf ("Donner la première valeur de x: ") ;
    scanf ("%d", &x) ;
    printf ("Donner la deuxième valeur de y: ") ;
    scanf("%d", &y) ;
results= maximum (x, y) ;
printf("Le maximum entre %d et %d est : %d \n ", x, y, results) ;
return 0 ; }
```

1) Exécution du programme pour x = 5 et y = 3 :

```
#include<stdio.h>
int maximum(int a, int b)
{
      if (a>b)
      return a ;
      else
      return b ;
      }
int main( )
{
      int x, y, results ;
      printf ("Donner la première valeur de x: ") ;
      scanf ("%d", &x) ;
      printf ("Donner la deuxième valeur de y: ") ;
      scanf("%d", &y) ;
results= maximum (x, y) ;
printf("Le maximum entre %d et %d est : %d \n ", x, y, results) ;
return 0 ;
}
```



2) Déroulement du programme pour x = 5 et y = 3 :

Instructions	Variables programme principale			Variables fonction maximum			Affichage
	x	y	results	a	b	maximum	
printf ("Donner la première valeur de x: ") ;							Donner la première valeur de x:
scanf ("%d", &x) ;	5						
printf ("Donner la deuxième valeur de y: ") ;							Donner la deuxième valeur de y:

scanf ("%d", &y) ;	5	3					
results= maximum (**x**, **y**) ; *Appel à la fonction maximum et transmission des paramètres*	5	3	?	5	3		
if (a>b) ⟶ **if** (5>3) True **return** a ; *la valeur de maximum est retournée à la variable globale results*						5	
results= maximum (**x**, **y**) ; results= **5**			5				
printf("Le maximum entre %d et %d est : %d \n ", **x**, **y**, **results**) ;	5	3	5				Le maximum entre **5** et **3** est : **5**

3) les paramètres formels da la fonction : Les paramètres formels de la fonction sont: **a** et **b**

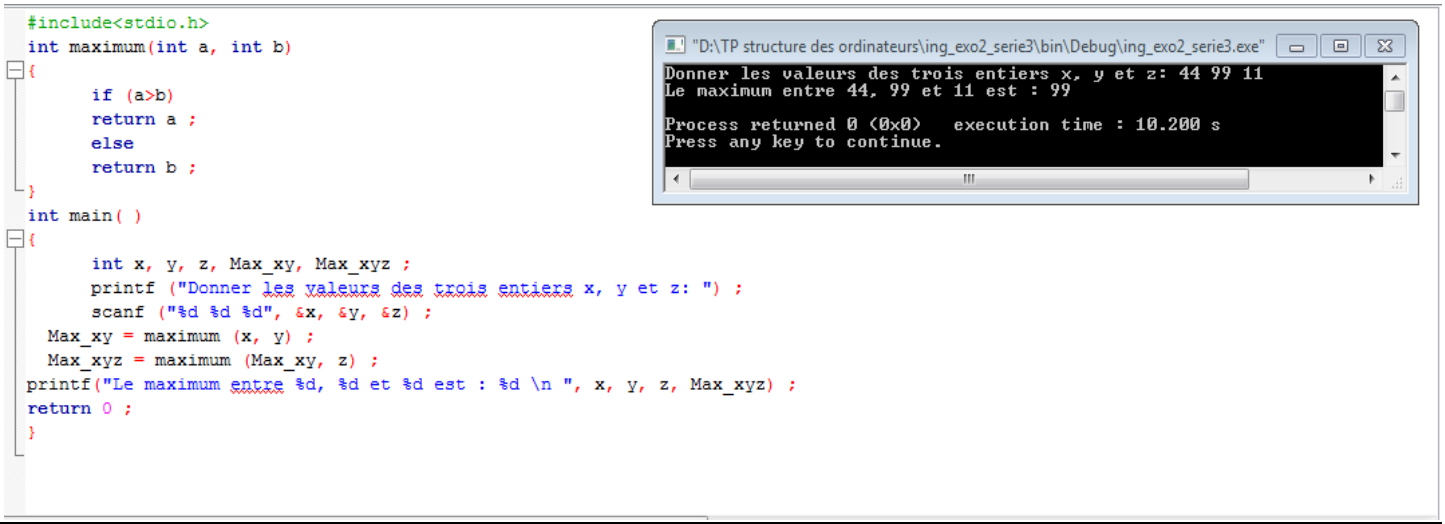
4) les paramètres effectifs: Les paramètres effectifs sont: x, y et results

5) Réécrire le programme pour déterminer le maximum entre trois nombres entiers x, y et z :

Programme en langage C

```
#include<stdio.h>
int maximum(int a, int b)
{
    if (a>b)
    return a ;
    else
    return b ;
}
int main( )
{
    int x, y, z, Max_xy, Max_xyz ;
    printf ("Donner les valeurs des trois entiers x, y et z: ") ;
    scanf ("%d %d  %d ", &x, &y, &z) ;
  Max_xy = maximum (x, y) ;
  Max_xyz = maximum (Max_xy, z) ;
printf("Le maximum entre %d, %d et %d est : %d \n ", x, y, z, Max_xyz) ;
return 0 ;
}
```

Compilation et exécution du programme :



6) Réécrire le programme en remplaçant la fonction par une procédure :

Avec deux nombres entiers x et y

```
#include <stdio.h>

void maximum(int a, int b, int *res)
{
   if (a > b)
      *res = a;
   else
      *res = b;
}
int main()
{
   int x, y, result;

   printf("Donner la première valeur de x : ");
   scanf("%d", &x);
   printf("Donner la deuxième valeur de y : ");
   scanf("%d", &y);

   maximum (x, y, &result);
   printf("Le maximum entre %d et %d est : %d\n",
x, y, result);
   return 0;
}
```

Avec trois nombres entiers x, y et z

```
#include <stdio.h>
void maximum(int a, int b, int *res)
{
   if (a > b)
      *res = a;
   else
      *res = b;
}
int main()
{
   int x, y, z, Max_xy, Max_xyz;
   printf ("Donner les valeurs des trois entiers x, y et z : ");
   scanf ("%d %d %d", &x, &y, &z);
   maximum (x, y, &Max_xy);
   maximum (Max_xy, z, &Max_xyz);
   printf("Le maximum entre %d, %d et %d est : %d\n",
x, y, z, Max_xyz);
   return 0;
}
```

Compilation et exécution du programme pour deux entiers

```
#include <stdio.h>
void maximum(int a, int b, int *res)
{
    if (a > b)
        *res = a;
    else
        *res = b;
}
int main()
{
    int x, y, result;

    printf("Donner la première valeur de x : ");
    scanf("%d", &x);
    printf("Donner la deuxième valeur de y : ");
    scanf("%d", &y);

    maximum(x, y, &result);
    printf("Le maximum entre %d et %d est : %d\n", x, y, result);
    return 0;
}
```

```
"D:\TP structure des ordinateurs\ing_exo2_serie3\bin\Debug\ing_exo2_seri...
Donner la premiPre valeur de x : 123
Donner la deuxiPme valeur de y : 999
Le maximum entre 123 et 999 est : 999

Process returned 0 (0x0)   execution time : 14.106 s
Press any key to continue.
```

Compilation et exécution du programme pour trois entiers

```
#include <stdio.h>
void maximum(int a, int b, int *res)
{
    if (a > b)
        *res = a;
    else
        *res = b;
}
int main()
{
    int x, y, z, Max_xy, Max_xyz;

    printf("Donner les valeurs des trois entiers x, y et z : ");
    scanf("%d %d %d", &x, &y, &z);

    maximum(x, y, &Max_xy);
    maximum(Max_xy, z, &Max_xyz);

    printf("Le maximum entre %d, %d et %d est : %d\n", x, y, z, Max_xyz);
    return 0;
}
```

```
"D:\TP structure des ordinateurs\ing_exo2_serie3\bin\Debug\ing_exo2_serie3.exe"
Donner les valeurs des trois entiers x, y et z : 21 43 122
Le maximum entre 21, 43 et 122 est : 122

Process returned 0 (0x0)   execution time : 14.842 s
Press any key to continue.
```

Exercice N°3 :

Écrire un programme en **C** qui permet de lire un tableau **T de N réels**. Le programme doit faire appel à une procédure qui détermine le plus grand élément du **tableau T** ainsi que sa position.

Programme en langage C

```
#include <stdio.h>

   int N, i;
   float T[100];   // Paramètres globales du programme
   float max;
   int pos;

void MaxTableau (float T [ ], int N, float *max, int *pos)
{
   int i;
   *max = T [0];   // initialisations
   *pos = 0;

   for (i = 1; i < N; i++)
   {
      if (T[i] > *max)
      {
         *max = T[i];
         *pos = i;
      }
   }
}
int main ()   //Début du programme principal
{
   printf ("Donner la taille du tableau N : ");
   scanf ("%d", &N);

   for (i = 0; i < N; i++)
   {                              // Lecture du tableau
      printf ("T[%d] = ", i);
      scanf ("%f", &T[i]);
   }
   MaxTableau (T, N, &max, &pos);   //Appel de la procédure

   printf (" Le plus grand element est : %.2f\n", max);  // Affichage des résultats
   printf ("Sa position est : %d\n", pos);

   return 0;
}  //Fin du programme principal
```

Compilation et exécution du programme

```
#include <stdio.h>
    int N, i, pos; float T[100], max;
void MaxTableau(float T[], int N, float *max, int *pos)
{    int i;
    *max = T[0];
    *pos = 0;
    for(i = 1; i < N; i++)
    {
        if(T[i] > *max)
        {   *max = T[i];
            *pos = i;}}}
int main()
{   printf("Donner la taille du tableau N : ");
    scanf("%d", &N);
    for(i = 0; i < N; i++)
    {   printf("T[%d] = ", i);
        scanf("%f", &T[i]);}
    MaxTableau(T, N, &max, &pos);
    printf("Le plus grand element est : %.2f\n", max);
    printf("Sa position est : %d\n", pos);
    return 0;}
```

```
"D:\TP structure des ordinateurs\ing_exo3_Serie3\bin\Debug\ing_ex...
Donner la taille du tableau N : 5
T[0] = 23
T[1] = 5
T[2] = 99
T[3] = 41
T[4] = 33
Le plus grand element est : 99.00
Sa position est : 2

Process returned 0 (0x0)   execution time : 35.184 s
Press any key to continue.
```

Exercice N°4 :

Écrire un programme en **C** qui lit les valeurs d'une matrice **M (N ×N)** de type **entier**, comportant une fonction qui permet de retrouver le nombre de fois qu'un nombre (choisi par l'utilisateur) apparait ainsi que ses positions dans la matrice.

Programme en langage C

```
#include <stdio.h>
int RepPos (int M [] [100], int N, int x)
{
   int i, j, cpt = 0; //initialisation du compteur
   printf ("Positions du nombre %d :\n", x);
   for (i = 0; i < N; i++)
   {
      for (j = 0; j < N; j++)
      {
         if (M[i][j] == x)   //Recherche de la variable x dans la matrice M
         {
            printf (" - Position : (ligne %d , colonne %d)\n", i, j);  //affichage de la position
            cpt=cpt+1;    //incrémentation du compteur a chaque fois x trouvé en M
         }
      }
   }
   return cpt;
}
int main ()     //début du programme principal
{
   int N, i, j, x, nb, M[100][100];
   printf ("Donner la dimension N de la matrice NxN : ");
   scanf ("%d", &N) ;
   for (i = 0; i < N; i++)
   {
      for (j = 0; j < N; j++)
      {
         printf ("M[%d][%d] = ", i, j);
         scanf ("%d", &M[i][j]);   //Lecture de la matrice
      }
   }
   printf ("Donner le nombre a chercher : ");
   scanf ("%d", &x);
   nb = RepPos (M, N, x);   //Appel de la fonction
   printf ("Le nombre %d apparait %d fois dans la matrice.\n", x, nb); //affichage nb d'apparition de x
   return 0;
}    //Fin du programme principal
```

Compilation et exécution du programme

```c
#include <stdio.h>
int RepPos(int M[][100], int N, int x)
{ int i, j, cpt = 0;
    printf("\nPositions du nombre %d :\n", x);
    for(i = 0; i < N; i++)
    {  for(j = 0; j < N; j++)
        {  if(M[i][j] == x)
            {  printf(" - Position : (ligne %d , colonne %d)\n", i, j);
            cpt=cpt+1;}}}    return cpt;}
int main()
{   int N, i, j, x, nb, M[100][100];
    printf("Donner la dimension N de la matrice NxN : ");
    scanf("%d", &N);
    for(i = 0; i < N; i++)
    {   for(j = 0; j < N; j++)
        { printf("M[%d][%d] = ", i, j);
          scanf("%d", &M[i][j]);}}
    printf("\nDonner le nombre a chercher : ");
    scanf("%d", &x);       nb = RepPos(M, N, x);
    printf("\nLe nombre %d apparait %d fois dans la matrice.\n", x, nb);
    return 0;}
```

```
"D:\TP structure des ordinateurs\ing_exo4_serie3\bin\Deb...
Donner la dimension N de la matrice NxN : 3
M[0][0] = 2
M[0][1] = 4
M[0][2] = 9
M[1][0] = 6
M[1][1] = 2
M[1][2] = 55
M[2][0] = 3
M[2][1] = 7
M[2][2] = 2

Donner le nombre a chercher : 2

Positions du nombre 2 :
 - Position : (ligne 0 , colonne 0)
 - Position : (ligne 1 , colonne 1)
 - Position : (ligne 2 , colonne 2)

Le nombre 2 apparait 3 fois dans la matrice.
```