Les procédures et les fonctions

I - Les procédures :

A - Définition et Déclaration :

Une procédure est une entité algorithmique indépendante qui possède sa propre déclaration, réalise un traitement et échange avec son environnement un ensemble de paramètre d'entrées/sorties.

Synthaxe :

//////// Algo ////////

```
Procédure: nomProcedure(liste
d'argument)
Déclaration:
      | //variables locales
Début:
      | //traitement
Fin: nomProcedure
```

//////// C ////////

```
Void nomProcedure(liste d'argument)
{
      //variables locales
      //traitement
}
```

//////// PHP ////////

```
Function nomProcedure(liste d'argument)
{
      //variables locales
      //traitement
}
```

nomProcedure est un identificateur, composé de lettres,chiffres et le caractère " _ "
Liste d'arguments on distingue trois type :

- **les entrées :** données reçues par la procédure pour réaliser le traitement.
- **les sorties :** résultats fournis par la procédure à son environnement.
- **les entrées-sorties :** données reçues, modifiées par la procédure et renvoyées comme résultat de la procédure.

Les variables locales ensemble de variables visibles et actives exclusivement à l'intérieur de la procédure,utilisés pour réaliser le traitement.

Le traitement ensemble d'action utilisés pour réaliser le travail attendue de la procédure.

B - Exemples :

Exo 1 : procédure sans paramètre
ecrire une procédure en algo,c,php qui affiches les nombres entiers compris en 1 et 100.

```
Procedure : Exo1()
Déclaration :
    |i : entiers
Début :
    |pour i allant de 1  à  100
    |    |afficher(i)
    |fin pour
Fin : Exo1
```

Exo 2 : procédure avec paramètre
ecrire une procédure qui reçoit un nombre et un exposant et calcule la puissance du nombre par l'exposant en utilisant des multiplications successives.

```
Procedure : Exo2(entree nb:entierr,expo:entier)
Déclaration :
    |i,p : entiers
Début :
    |p ← 1
    |i ← 1
    |tantque i<=expo faire
    |   |p ← nb*p
    |   |i ← i+1
    |fintantque
    |afficher(p)
Fin : Exo2
```

C - Appel de la procédure :

Pour appeler une procédure, on évoque son nom suivie pour les paramètre affectifs qui doivent correspondant aux arguments de la procédure au nombre,type et ordre.

Exemple :

```
Algo : utilisation de la puissance
Déclaration :
```

```
|procedure puissance(entrée:nb,exp:entier)
|     |declaration
|     |p,i : entier
|     |debut
|     |p ← 1
|     |i ← 1
|     |tantque i<exp faire
|     |     |p ← nb*p
|     |     |i ← i+1
|     |fin tantque
|     |afficher(p)
|finpuissance
|a,b:entier //variables globales
|debut
|afficher("donner un nombre")
|saisir(a)
[afficher("saisir un exposant")
|saisir(b)
|puissance(a,b)
|fin utilisation de la puissance
```

II - Les fonctions :

A - Définition et Déclaration :

Une fonction est une entité algorithmique indépendante, qui possède sa propre declaration, réalise un traitement et retourne une seul valeur.

Le type du résultat retourné définie le type de la fonction.

Synthaxe :

//////// Algo ////////

```
Fonction: nomFonction(liste d'argument):type
Déclaration:
      | //variables locales
Début:
      | //traitement
      | retourner expression
Fin: nomFonction
```

//////// C ////////

```
Type nomFonction(liste d'argument)
{
      //variables locales
      //traitement
```

```
    return expression;
}
```

//////// PHP ////////

```
Function nomFonction(liste d'argument)
{
     //variables locales
     //traitement
      return $expression;
}
```

B - Exemples :

Exo 1 :

ecrire une fonction qui renvoie un nombre et affiche son factoriel

```
Fonction factoriel(entrée:n:entier):entier
Déclaration
   |f,c:entier
Debut
   |f ← 1
   |pour i allant de 1 à n faire
   |    |f ← f*i
   |finpour
finfonction return f
```

C - Appel d'une fonction

Pour appeler une fonction, on évoque son nom suivi de paramètre effectifs à envoyer à la fonction et qui doivent correspondre aux arguments de la fonction en nombre,type et arguments. Le retour de la fonction est reçus dans une variable de même nom.

Exemple :

ecrire une fonction qui reçoit un entier et renvoie en sortie un nombre de diviseur.

```
Algo: diviseur
Declaration:
|Fonction: nbdiviseur(entrée: nb:entier):entier
```

```
|Declaration:
|   |div,cpt:entier
|Debut
|    |cpt ← 0
|    |pour div allant de 1  à nb faire
|    |    |si nb%div=0 alors
|    |    |   |cpt → cpt+1
|    |    |finsi
|    |finpour
|return cpt
|fin nbdiviseur
|a,b:entier
|Debut
|Afficher("Donner un nb entier")
|Saisir(a)
|b ← nbdiviseur(a)
|afficher("nbr de diviseur est :",b)
|fin diviseur
```

Série D'exercice :

Exo1:

ecrire une fonction en langage C qui reçoit deux limites a et b et affiche la somme et le produit de tous les nombres compris entre les deux limites a,b.

Exo2:

réaliser la fonction suivante

- **`float moyenne(float tab[],int taille);`** permet de calculer la moyenne des éléments du tableau
- **`float maximum(float tab[],int taille);`** retourne l'élément max du tableau.
- **`float minimum(float tab[],int taille);`** retourne l'élément minimum.
- **`int occurrence(float tab[],int taille,float valeur)`** retourne le nombre d'occurrences de la valeur des tableaux.