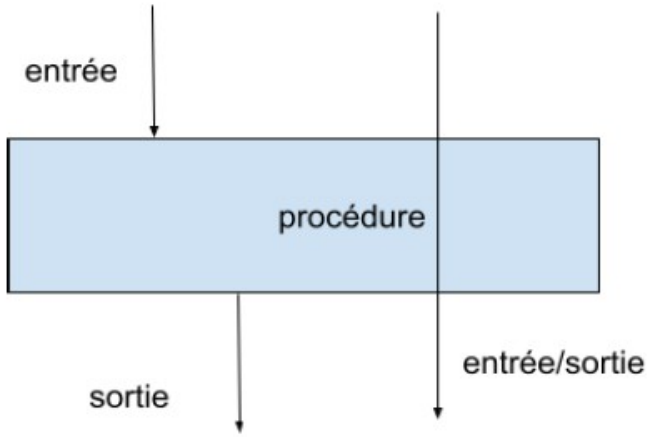
Les procédures et les fonctions

1) 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 paramètres (ou arguments) d'entrée, de sortie ou d'entrée/sortie



Déclaration de la procédure en Algo :

```
Procédure nomProcédure (liste des arguments)

Déclaration
   //variables locales
Début

   //traitement
Fin nomProcédure
```

Déclaration de la procédure en C :

```
void nomProcédure (liste d'arguments){
   //variables locales
   //traitement
}
```

Déclaration de la procédure en PHP :

```
function nomProcédure (liste d'arguments){
   //traitement
}
```

Nom de la procédure : est un identificateur comme celui des variables, composé de lettres, chiffres et le caractère "_". Il permet d'appeler la procédure en dans le programme principal.

Liste d'arguments : il y a trois types d'arguments, les paramètres d'entrée qui ont des données fournies à la procédure pour réaliser le traitement. Les paramètres de sortie qui sont les résultats

renvoyés par la procédure au programme appelant, et les paramètres d'entrée / sortie qui sont des données reçues, modifiées et renvoyées sous forme de résultats.

Variables locales : ce sont des objets de données déclarés à l'intérieur de la procédure pour réaliser le traitement. Elles sont visibles et actives exclusivement à l'intérieur de la procédure.

Traitement : dans le corps de la procédure, on peut utiliser n'importe quelle instruction comme dans un programme principal y compris d'appel d'autres procédures ou fonctions.

Le rôle des procédures : l'écriture des procédures dans la résolution des problèmes en informatique permet :

-Raccourcir les codes pour éviter les répétitions dans les programmes
-Plus de visibilité et de lisibilité dans les codes
-Optimisation des codes
-Facilité de maintenance
-C'est la base de la macro programmation et plus tard la programmation orientée objet

b- Exemples

Exemple 1 : Écrire une procédure qui affiche les nombres entre 1 et 100

```
procédure afficheNombres ( )

Déclaration
   i : entier

Début
   pour i allant de 1 à 100 faire
      afficher("Nombre : ", i )
   fin pour
fin afficheNombres
```

Traduction en C :

```
void afficheNombres ( ){
   int i ;
   for (i = 1; i <= 100 ; i++){
      printf("Nombre : %d", i ) ;
   }
}
```

Traduction en PHP :

```
function afficheNombres ( ){
   for ($i = 1; $i <= 100 ; $i++){
      printf("Nombre : %d", $i ) ;
   }
}
```

Exemple 2 : Écrire un procédure qui affiche les nombres compris entre deux limites reçues en entrée

```
procédure afficheNombres ( entrée : lim1, lim2 : entier )

Déclaration
   i : entier

Début
   pour i allant de lim1 à lim2 faire
      afficher(“Nombre : “, i )
   fin pour
fin afficheNombres
```

Traduction en C :

```
void afficheNombres ( int lim1, int lim2 ){
   int i ;
   for (i = lim1; i <= lim2 ; i++){
      printf(“Nombre : %d”, i ) ;
   }
}
```

Traduction en PHP :

```
function afficheNombres ( $lim1, $lim2){
   for ($i = $lim1; $i <= $lim2 ; $i++){
      printf(“Nombre : %d”, $i ) ;
   }
}
```

c- Appel de la procédure

Pour appeler une procédure, il suffit de mentionner son nom suivi des paramètres effectifs

Syntaxe d'appel :

```
nomProcédure(liste de paramètre effectifs)
```

Exemple complet : Écrire un programme utilisant une procédure qui reçoit deux limites et réalise la somme de tous les nombres compris entre les limites

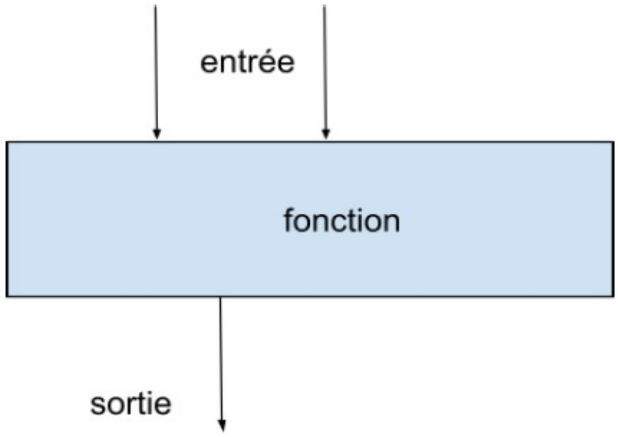
```
#include <stdio.h>

void somme (int lim1, int lim2){
   int i , s ;
   s = 0;
   for (i = lim1; i <= lim2 ; i++){
      s = s + i ;
   }
   printf(“La somme des nombres est : %d”, s);
}
```

```
int main ( ) {
   int a, b ;
   printf(“Donner la première limite :”);
   scanf(“%d”, & a);
   printf(“Donner la deuxième limite :”);
   scanf(“%d”, & b);
   //appel de la procédure somme
   somme (a, b);
   return 0;
}
```

2) Les fonctions

a- Définition et déclaration

Une fonction est entité algorithmique indépendante qui possède sa propre déclaration, réalise un traitement et peut recevoir des paramètres données et renvoie un seul résultat à son environnement.



Syntaxe en Algo :

```
fonction nomFonction (paramètres d’entrées) : type

Déclaration
   //variables locales

Début
   //traitement
   retourner expression
Fin nomFonction
```

Syntaxe en C :

```
type nomFonction (paramètres d’entrée){
   //variables locales
   //traitement
   return expression ;
}
```

Syntaxe en PHP :

```
function nomFonction (paramètres d'entrée){
   //traitement
   return expression ;
}
```

Le type de retour de la fonction est déterminé par le type de l'expression à retourner

b- Exemple

Écrire une fonction qui calcule la somme de tous les nombres compris entre deux limites

```
fonction somme (entrée : lim1, lim2 : entier) : entier

Déclaration
   i, s : entier

Début
   s ← 0
   pour i allant de lim1 à lim2 faire
      s ← s + i
   Fin pour
   retourner s
Fin somme
```

Traduction en C :

```
int somme (int lim1, int lim2 ){
   int i , s ;
   s = 0;
   for (i = lim1; i <= lim2 ; i++){
      s = s + i ;
   }
   return s;
}
```

Traduction en PHP :

```
function somme ( $lim1, $lim2 ){
   $s = 0;
   for ($i = $lim1; $i <= $lim2 ; $i++){
      $s = $s +$i ;
   }
   return $s;
}
```

c- Appel de la fonction

Pour appeler une fonction, il faut prévoir une variable qui devra recevoir le retour de la fonction

Syntaxe d'appel :

```
NomVariable ← nomFonction(paramètres effectifs)
```

Exemple complet : Écrire un programme utilisant une fonction qui reçoit deux limites et réalise la somme de tous les nombres compris entre les limites et retourne cette somme

```
#include <stdio.h>

int somme (int lim1, int lim2){
   int i , s ;
   s = 0;
   for (i = lim1; i <= lim2 ; i++){
      s = s + i ;
   }
   return s;
}

int main ( ) {
   int a, b, c ;
   printf("Donner la première limite :");
   scanf("%d", & a);
   printf("Donner la deuxième limite :");
   scanf("%d", & b);
   //appel de la fonction somme
   c = somme (a, b);
   //le résultat de retour de la fonction est stocké dans la variable c
   printf("La somme des nombres est de : %d", c);
   return 0;
}
```

3) Série d'exercices

Exercice 1 : Écrire un programme qui demande à l'utilisateur un nombre entier puis fait appel à une procédure qui affiche les diviseurs du nombre entier saisi

```
Algo : exo1

Déclaration
   procédure diviseurs (entrée : nb : entier)

Déclaration
   div : entier

Début
   pour div allant de 1 à nb faire
      si nb % div = 0
         alors afficher("Diviseur :", div)
      fin si
   fin pour
Fin diviseurs
```

```
//variables globales
   nbr : entier

Début
   afficher(“Donner votre nombre :”)
   saisir (nbr)
   //appel de la procédure : diviseurs
   diviseurs(nbr);
Fin exo1
```

Traduction en C :

```
#include <stdio.h>

//procédure diviseurs
void affiche_diviseurs( int a){
   int i;
   for(i=1; i<= a; i++){
      if( a%i==0){
         printf("%d est un diviseur de %d \n", i, a);
      }
   }
}

int main (){
   int nb;
   printf("Entrez un entier : ");
   scanf("%d", &nb);
   //Appel de la fonction affiche_diviseurs
   affiche_diviseurs(nb);
   return 0;
}
```

Traduction en PHP :

```
<!DOCTYPE html>
<html>
   <head>
      <title>Diviseurs</title>
   </head>
   <body>
      <center>
         <h1> Procédure recherche de diviseurs</h1>
         <form method="post" action="proc_diviseurs.php">
            Nombre entier : <input type="text" name="a">
            <input type="submit" name="rechercher" value="Rechercher">
         </form>
         <?php
            function affiche_diviseurs( $a){
               for($i=1; $i<= $a; $i++){
                  if( $a % $i==0){
```

```
                    printf("%d est un diviseur de %d <br/>", $i, $a);
                }
            }
        }
        if (isset($_POST['rechercher'])){
            $nb = $_POST['a'];
            affiche_diviseurs($nb);
        }
    ?>
  </center>
 </body>
</html>
```

Exercice 2 : Écrire un programme en C, PHP qui permet de demander à l'utilisateur deux limites et affiches tous les nombres pairs compris entre les deux limites en utilisant une procédure

```
#include <stdio.h>

void affiche_pairs (int lim1, int lim2){
   int i;
   i = 0;
   for (i = lim1; i <= lim2; i++){
      if (i % 2 == 0){
         printf("\n voici le nombre pair : %d", i);
      }
   }
}

int main ( ) {
   int a, b;
   printf("donner la première limite :");
   scanf("%d", & a);
   printf("donner la deuxième limite :");
   scanf("%d", & b);
   affiche_pairs (a, b);
   return 0;
}
```

Traduction en PHP :

```
<!DOCTYPE html>
<html>
 <head>
  <title>Pairs</title>
 </head>
 <body>
  <center>
   <h1> Procédure afficher les pairs </h1>
   <form method="post" action="affiche_pairs.php">
    Première Limite : <input type="text" name="lim1"> <br/>
    Deuxième Limite : <input type="text" name="lim2"><br/>
```

```
            <input type="submit" name="afficher" value="Afficher">
        </form>
        <?php
            function affiche_pairs( $lim1, $lim2){
                for($i=$lim1; $i<= $lim2; $i++){
                    if( $i % 2==0){
                        printf("%d est un nb pair<br/>", $i);
                    }
                }
            }
            if (isset($_POST['afficher'])){
                $lim1 = $_POST['lim1'];
                $lim2 = $_POST['lim2'];
                affiche_pairs( $lim1, $lim2);
            }
        ?>
    </center>
  </body>
</html>
```

Exercice 3 : Écrire une fonction qui calcule la factorielle d'un nombre entier. Ecrire un programme utilisant cette fonction

```
#include <stdio.h>

int factorielle (int n ){
   int fact = 1 ;
   int i ;
   for (i = 1 ; i <= n ; i++){
      fact = fact * i ;
   }
   return fact ;
}

int main ( ){
   int a;
   printf(“Donner un nombre entier :”);
   scanf(“%d”, & a);
   int resultat ;
   resultat = factorielle (a);
   printf(“La factorielle est de : %d”, resultat);
   return 0 ;
}
```