Les procédures et les fonctions

1. Les procédures

a. Définition et syntaxe

Une procédure est une entité algorithmique indépendante qui possède sa propre déclaration, réalise un traitement et échange avec son environnement un ensemble d'arguments (paramètres) d'entrée et de sortie.

Syntaxe :

```
Procédure nomProcédure (liste d'arguments)
Déclaration
        variables locales
Début
        traitement
Fin nomProcédure
```

Liste des arguments : On distingue trois types d'arguments : entrée, sortie et entrée / sortie.

- Arguments en entrée : ce sont les données reçues par la procédure pour réaliser en traitement.
- Arguments de sortie : ce sont les résultats fournis par la procédure à son environnement.
- Arguments d'entrée/sortie : des données reçues en entrée, modifiées à l'intérieur de la procédure et renvoyées en résultats à son environnement.

Variables locales : La procédure peut déclarer en son sein toute variable nécessaire à son travail. Ces variables locales sont actives et visibles exclusivement dans la procédure et ne peuvent être utilisées à l'extérieur de celle-ci.

S'agissant des variables globales, qui ont été définies dans le programme principale, ces variables peuvent être utilisées dans la procédure. Mais il faut limiter cette pratique.

Traitement : le corps de la procédure peut être constitué de toute instruction qui permet la réalisation du traitement y compris l'appel d'autres procédures et fonctions.

Syntaxe en Langage C :

```
void nomProcédure (liste des arguments )
{
        //variables locales

        //traitement
}
```

Syntaxe en Langage PHP :

```
function nomProcédure (liste des arguments )
{
        //variables locales

        //traitement
}
```

Exemple de procédure : Écrire une procédure qui permet d'afficher la somme de tous les nombres compris entre deux limites entières reçues en paramètres d'entrée.

La procédure en Algorithme :

```
procédure somme ( Entrée : lim1, lim2 : entier )
```

```
Déclaration
        s, i : entier
Début
        s <- 0
        Pour i allant de lim1 à lim2 faire
                s <- s + i
        Fin pour
        Afficher (“ La somme des nombres est de : “, s)
Fin somme
```

La procédure en langage C :

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

La procédure en langage PHP :

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

b. Appel de la procédure

Pour utiliser la procédure dans différents endroits d’un programme principal, on réalise un appel de procédure : **nomProcédure (liste des paramètres efffectifs).**

Pour appeler la procédure Somme écrite ci-dessus, on doit lui transmettre deux limites :

Appel en Algorithme :

```
Algo : utilisation_somme
Déclaration
        x, y : entier
Début
        Afficher (“Donner la première limite :”)
        saisir (x)
        Afficher (“Donner la deuxième limite :”)
        saisir (y)
        //Appel de la procédure
        somme ( x, y )
Fin utilisation_somme
```

En langage C :

```
        int main ()
        {
                int x, y ;
                printf (“Donner la première limite :”);
```

```
        scanf("%d", &x);
        printf ("Donner la deuxième limite :");
        scanf ("%d", &y) ;
        // Appel de la procédure
        somme ( x, y ) ;
        return 0;
}
```

En Langage PHP : utilisation d'un formulaire html pour saisir les deux limites

```
if (isset ($_POST['Valider'])) {
        $x = $_POST['x'] ;
        $y = $_POST['y'] ;
        //appel de la procédure
        somme ($x, $y);
}
```

1. Les fonctions

1. Définition et syntaxe

Une fonction est entité algorithmique indépendante qui possède sa propre déclaration, réalise un traitement et renvoie en sortie sous forme de résultat un seul paramètre à son environnement.

La sortie de la fonction définit le type de la fonction. Donc, une fonction est toujours typée car elle renvoie obligatoirement un résultat.

Syntaxe en Algorithme :

```
Fonction nomFonction (liste d'arguments en entrée) : type
Déclaration
        variables locales
Début
        traitement
        retourner unRésultat
Fin nomFonction
```

Syntaxe en langage C :

```
type nomFonction (liste des arguments en entrée)
{
        //variables locales
        //traitement
        return unRésultat ;
}
```

Syntaxe en langage PHP :

```
function nomFonction (liste des arguments en entrée)
{
        //variables locales
        //traitement
        return unRésultat ;
}
```

Exemple : Ecrire une fonction qui retourne la somme des nombres compris entre deux limites reçues en entrée.

Réalisation de la fonction en Algorithme :

```
fonction somme ( Entrée : lim1, lim2 : entier ) : entier
Déclaration
        s, i : entier
Début
        s <- 0
        Pour i allant de lim1 à lim2 faire
                s <- s + i
        Fin pour
        retourner s
Fin somme
```

Réalisation en Langage C :

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

Réalisation en Langage PHP :

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

b. Appel de la fonction

Pour réaliser l'appel d'une fonction, on doit prévoir une variable **de même type** que le retour de la fonction pour stocker le résultat retourné par la fonction.

Syntaxe : nomVariable <- nomFonction (liste des paramètres effectifs)

Exemple : Pour recevoir la somme des nombres compris entre les limites, on doit déclarer un entier dans le programme principal.

Appel en Algorithme :

```
Algo : utilisation_somme
Déclaration
        x, y, som  : entier
Début
        Afficher ("Donner la première limite :")
        saisir (x)
        Afficher ("Donner la deuxième limite :")
        saisir (y)
        //Appel de la fonction
        som <- somme ( x, y )
        afficher ("La somme des nombres est de : ", som)
Fin utilisation_somme
```

En langage C :

```
int main ()
{
        int x, y , som ;
        printf (“Donner la première limite :”);
        scanf(“%d”, &x);
        printf (“Donner la deuxième limite :”);
        scanf (“%d”, &y) ;
        // Appel de la procédure
        som = somme ( x, y ) ;
        printf (“La somme des nombres est de : %d “, som) ;
        return 0;
}
```

En Langage PHP : utilisation d’un formulaire html pour saisir les deux limites

```
if (isset ($_POST[‘Valider’])) {
        $x = $_POST[‘x’] ;
        $y = $_POST[‘y’] ;
        //appel de la procédure
        $som = somme ($x, $y);
        printf (“La somme des nombres est de : %d “, $som) ;
}
```

_____ FIN DU COURS _____