Les procédures et les fonctions

Déclaration d'une procédure

```
procédure nomproc (arguments)
Déclaration
  | variables locales
Début
  | traitement
Fin nomproc
```

Appel de la procédure

```
nomproc(liste de paramètres)
```

Déclaration d'une fonction

```
fonction nomFct (arguments) : type
Déclaration
  | variables locales
Début
  | traiement
  | retourner expression
fin nomFct
```

Appel de la fonction

```
nomVariable <- nomFct(liste de parametres)
```

Exercices

Exercice 1

ecrire une procédure en algo, c, php qui reçoit un entier et affiche ses diviseurs

Algo :

```
procédure diviseurs (nb: entier
Déclaration
  | i: entier
Début
  | pour i allant de 1 a nb faire
  | | Si nb % i = 0
  | | | Alors afficher ("diviseur : ", i)
  | | Fin si
  | Fin pour
Fin diviseur
```

C :

```c
#include <stdio.h>

int main() {
  void diviseurs(int nb) {
    int i;
    for(i=1; i<=nb; i++) {
      if(nb % i == 0) {
```

```
        printf("diviseur %d", i);
      }
    }
  }
}
```

PHP :

```
<?php
  function diviseur($nb) {
    for($i-1; $i < $nb; i++) {
      if($nb % $i == 0) {
        echo "diviseur : ".$i;
      }
    }
  }
?>
```

Exercice 2

ecrire une fonction en algo, c, php qui reçoit deux limites et retourne la somme de tous les nombres comprises entre elles.

Algo :

```
  fonction somme(nb1, nb2 : entier) : entier
  Déclaration
    | s, i : entier
  Début
    | s <- 0
    | pour i allant de nb1 à nb2 faire
    |  | s <- s + i
    |  fin pour
    retourner s
  fin somme
```

C:

```
#include <stdio.h>

int main() {
  int somme(nb1, nb2) {
    s = 0;
    for(int i = nb1; i <= nb2; i++) {
      s = s + 1;
    }
    return s;
  }
  return 0;
}
```

PHP :

```
<?
  function somme ($nb1, $nb2) {
    $s = 0;
    for($i = nb1; $i <= nb2; $i++) {
      $s = $s + 1;
    }
    return s;
  }
?>
```