LES SOUS-PROGRAMMES

1 . Présentation

Les sous-programmes sont des parties écrites une seule fois par le programmeur mais qui sont exécutées à plusieurs reprises durant le programme.

Pour simplifier :

Un sous-programme est une partie de programme écrite une seule fois mais pouvant être exécutée plusieurs fois.

Au cours de chaque exécution, le traitement est identique mais les données peuvent être différentes. Autrement dit, le sous-programme exécute toujours la même chose sur des valeurs qui peuvent être différentes. Le programmeur doit donc fournir les valeurs qu'il veut voir traitées par le sous-programme, ce sont les **arguments**[1] du sous-programme.

2 . Les deux formes de sous-programme

Le langage du Cubloc autorise deux formes de sous-programme :

- les sous-routines
- les fonctions

Elles ont en commun :

- d'être écrites une seule fois
- de pouvoir être exécutées plusieurs fois
- de recevoir des arguments
- de fournir un résultat (bien sûr)

Elles diffèrent

- par la manière de les écrire et de les appeler
- et surtout par la manière d'utiliser le résultat.

Expliquons. Pour faciliter la compréhension, nous allons séparer le fond de la forme. Pour l'instant nous ne traitons que le fond par un texte en langue courante.

Soit à calculer la longueur de l'hypoténuse de plusieurs triangles rectangles en connaissant la longueur de leurs autres côtés.

2.1 Sous-routine

Dans le programme principal

```
' définition des variables

variable h  nb_réel   ' hypoténuse
    •
hypo (3, 4)      ' premier appel à la sous-routine, les arguments sont 3 et 4
si h > 20 alors ...
    •
hypo (5, 6)      ' deuxième appel, les arguments sont 5 et 6
```

1 Certains les appellent paramètres

```
        si h > 20 alors …
fin du programme        '  = fin du programme principal

' ======= définition de la sous-routine
Sous-routine hypo ( na  nb_réel,  nb nb_réel)
variable x  nb_réel       ' variable locale qui n'existe que dans la sous-routine
x = na * na + nb * nb
h = racine(x)
fin sous-routine
' ========== fin effective du programme
```

On remarque :

- l'emploi de variables de travail, na et nb. Ces variables prennent la valeur des arguments
- la déclaration d'une variable locale, **x**
- le résultat est inscrit dans la variable globale **h**, à l'intérieur de la sous-routine

2.2 Fonction

Dans le programme principal

```
        si hypo (3, 4)  > 20 alors …    ' premier appel à la fonction
          •
        si hypo (5, 6) > 20 alors ...       ' deuxième appel
          •
fin du programme        '  = fin du programme principal

' ======= définition de la fonction
Fonction hypo ( na  nb_réel,  nb nb_réel)  retourne  nb_réel
variable x  nb_réel  ' variable locale qui n'existe que dans la fonction,
x = na * na + nb * nb
hypo = racine(x)
fin fonction

' ========== fin effective du programme
```

On remarque

- la disparition de la variable h qui servait à exploiter le résultat
- dans le programme principal, la fonction se comporte comme une valeur, elle peut être utilisée dans des opérations au même titre qu'une variable
- la déclaration commence par le mot réservé « fonction »
- que la définition de la fonction indique le type de la valeur retournée
- l'utilisation de parenthèses.

Attention : ce texte n'indique pas la forme à adopter dans la programmation Cubloc.

3 . Exemples de programmation en langage CUBLOC

Reprenons le calcul de la longueur de l'hypoténuse d'un triangle connaissant la longueur des deux autres côtés.

3.1 Par Sub

```
'programme principal
'déclaration des variables
dim h as single      ' variable pour contenir la longueur de l'hypoténuse
        •
        •
hypo 3, 4
if h <> 5 then goto erreur
        •
        •
End                  ' fin du programme principal

' --------- début du sous-programme --------
Sub hypo(na as single, nb as single)
dim x as simple
x = na * na + nb * nb
h = sqr x
End Sub
```

3.2 Par Function

```
Dim h As Single
    If hypo(3, 4) <> 5 Then Goto erreur
       •
       •
    End              ' fin du programme principal

    ' --------- début du sous-programme --------
    Function hypo(na As Single, nb As Single) As Single
    Dim x As Single
    x = na*na + nb*nb
    hypo = Sqr x
    End Function
```

Remarquez comment se fait le passage du résultat, le retour du résultat se fait par une variable qui porte le même nom que la fonction.