ESATIC ECOLE SUPERIEURE AFRICAINE DES TIC	Année scolaire 2014 - 2015 PROF : KONAN HYACINTHE LICENCE 1

INTRODUCTION A L'ALGORITHMIQUE

CHAPITRE 7 :
FONCTIONS

1. Appel de fonctions prédéfinies

Les algorithmes font souvent appel à des fonctions prédéfinies :

- fonctions mathématiques : racine, fonctions trigonométriques, etc.
  exemple : pour calculer l'hypoténuse d'un triangle rectangle
  ```
  LIRE (a, b)
  c ← racine((a * a) + (b * b))
  ```
- fonction sur les chaînes de caractères : longueur, sous_chaine
  exemple : pour afficher les trois premiers caractères d'une chaîne
  ```
  LIRE (chaine)
  SI nbcar >= 3 ALORS
      ECRIRE ('Début : ', sous_chaine (chaine, 1, 3))
  FINSI
  ```

Dans un langage de programmation, les FONCTIONS PREDEFINIIES constituent une véritable boîte à outils, souvent organisée en BIBLIOTHEQUES DE FONCTIONS qui regroupent les fonctions par domaine (math, chaînes de caractères, système, etc.).
Si de nouvelles fonctions sont nécessaires, il est possible de les définir, c'est à dire de leur donner un identificateur et de programmer les calculs qui conduisent à ce qu'elles doivent renvoyer.

2. Rôle d'une fonction

Une fonction prend des données par l'intermédiaire de ses PARAMATRES, et renvoie un résultat conforme à sa SPECIFICATION, à condition que les conditions d'appels soient respectées.
Par exemple un algorithme ne doit pas appeler une fonction racine avec un paramètre de valeur négative, si cela arrive, c'est l'algorithme qui est faux, pas la fonction.

3. Exemple

Les deux algorithmes **Maxquatre** calculent le maximum de quatre nombres entiers. Dans le premier algorithme, la même instruction conditionnelle est répétée trois fois, aux noms des variables près Dans le deuxième algorithme, les trois instructions conditionnelles analogues sont remplacées par des appels à une même fonction qui s'applique à des valeurs différentes. Ceci rend plus compréhensible ce que fait l'algorithme en évitant de dupliquer des instructions identiques.

```
ALGORITHME Maxquatre_1
/* demande deux nombre à l'utilisateur, calcule et affiche le plus grand des deux. */
VARIABLE
        a, b, c, d : ENTIER /* nombres d'étude */
        mab, mcd  : ENTIER /* maxima intermédiaires */
        maxi    : ENTIER /* maximum des 4 nombres */

DEBUT
        LIRE (a, b, c, d)
        SI a > b ALORS
                mab ← a
        SINON
                 mab ← b
        FINSI
        SI c > d ALORS
                mab ← c
        SINON
                 mab ← d
        FINSI
        SI mab > mcd ALORS
                maxi ← mab
        SINON
                 maxi ← mcdb
        FINSI
        ECRIRE (maxi)
Fin

ALGORITHME Maxquatre_2
/* idem avec appels de fonctions */
VARIABLES
        idem ci-dessus
DEBUT
        LIRE(a, b, c, d)
        mab ← maxdeux(a, b)
        mcd ← maxdeux(c, d)
        maxi ← maxdeux(mab, mcd)
        ECRIRE(maxi)
FIN
```

Définition de la fonction maxdeux

```
FONCTION maxdeux(x, y : ENTIER) : ENTIER
/* renvoie le maximum des deux entiers passés en paramètres */

VARIABLE
        mxy  : ENTIER  /* maximum de x et de y */

DEBUT
        SI x > y ALORS
                mxy ← x
        SINON
                mxy ← y
        FINSI
        RETOURNER (mxy)
FIN
```

4. Définition, syntaxe et mécanisme de passage de paramètres

Une fonction est un bloc d'instructions

- qui porte un nom, son IDENTIFICATEUR
- qui prend des valeurs en entrée par l'intermédiaire de PARAMETRES
- qui calcule et renvoie un résultat conforme à sa spécification et aux conditions de son appel

Le PROFIL (ou SIGNATURE) d'une fonction est constitué de la liste de types des valeurs entrées et du type de la valeur renvoyée en résultat

liste des types des valeurs d'entrée → type du résultat

exemples :
maxdeux : (entier, entier) → entier
sous-chaine : (chaine de caractères, entier, entier) → chaine de caractères

4.1. Définition de la fonction

Une fonction peut être définie n'importe où, à l'extérieur d'un algorithme. La définition de la fonction se compose de :

- son en-tête qui comprend :
  - son identificateur
  - la liste de ses paramètres formels et de leur type
  - le type de la valeur qu'elle renvoie
- des déclarations locales (constantes ou variables)
- les instructions qui calculent son résultat
- au moins une instruction RETOURNER qui renvoie la valeur résultat

```
FONCTION <ident-fonction>
    (<ident-paramètre> : <type> <role>
     ...
     <ident-paramètre> : <type> <role>) : <type>

/* spécification et conditions d'appels */

VARIABLES locales
  <ident-variable> : <type> <rôle>
  ....
  <ident-variable> : <type> <rôle>

DEBUT
   <instructions>
   RETOURNER <expression>
FIN
```

NB : <role> correspond à du commentaire explicitant le rôle d'un paramètre ou d'une variable.

Attention : Il est indispensable de spécifier chaque fonction, et de préciser les conditions dans lesquelles elle doit être appelée. En effet si le programmeur qui intègre une fonction à un de ses programmes ne respecte pas les conditions d'appels, il n'a aucune garantie sur ce que fera la fonction dans son programme.

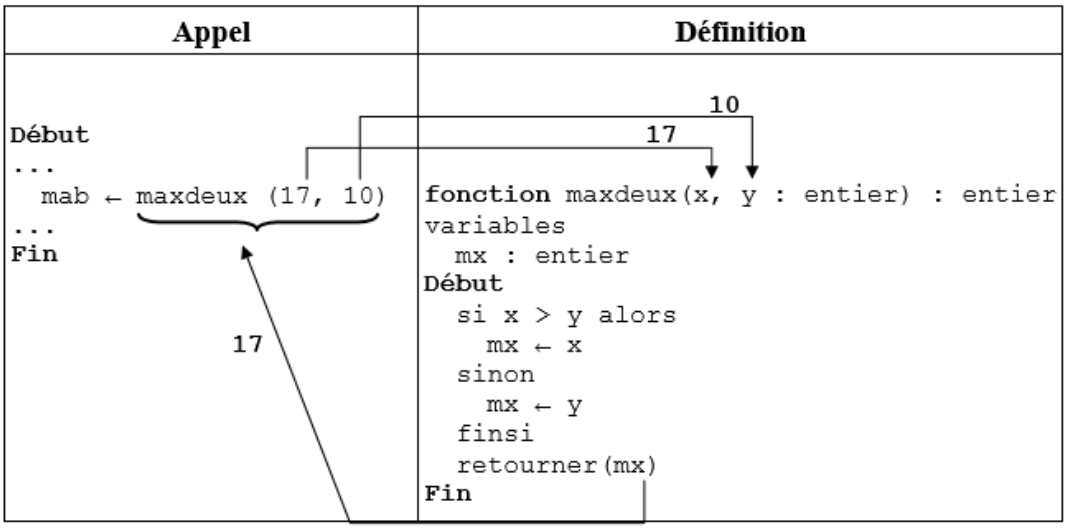
4.2. Appel de la fonction

La fonction est appelée depuis un algorithme principal, ou depuis une autre fonction (ou procédure que nous verrons plus loin), dans l'expression où le résultat qu'elle renvoie doit être utilisé.

<ident-fonction>(<ident-paramètre>, ..., <ident-paramètre>)

4.3. Passage de paramètres

Dans la définition de la fonction : PARAMETRES FORMELS
Dans l'appel de la fonction : PARAMETRES EFFECTIFS (ou réel3)
Il doit y avoir une correspondance **biunivoque** entre les paramètres formels et les paramètres effectifs, la correspondance étant régie par **l'ordre d'écriture**. À l'appel de la fonction, le premier paramètre effectif passe sa valeur au premier paramètre formel, le deuxième paramètre effectif passe sa valeur au deuxième paramètre formel, et ainsi de suite. La fonction exécute son code puis renvoie son résultat. Les paramètres qui se correspondent doivent avoir des types compatibles.



5. Intérêt des fonctions

- Ne pas dupliquer du code
- Offrir une meilleure lisibilité car le lecteur peut comprendre ce que fait une fonction, uniquement à la lecture de son nom, sans avoir à déchiffrer du code.
- Partager le développement entre différentes équipes qui se spécialisent.
- Construire des bibliothèques de fonction pour réutiliser ce qui a déjà été fait.

FIN DU CHAPITRE 7