Université Internationale de Casablanca

Cours Algorithmique: Procédures & Fonctions

Sanae EL OUKKAL

Introduction

- Dès qu'on commence à écrire des programmes importants, il devient difficile d'avoir une vision globale sur son fonctionnement et de traquer les erreurs.

→ ***D'où l'intérêt d'utiliser des Fonctions et des Procédures***

Fonction/Procédure: Définition

- Un sous-algorithme est un bloc faisant partie d'un algorithme. Il est déclaré dans la partie entête (avant le début de l'algorithme) puis appelé dans le corps de l'algorithme.

- Étant donné qu'il s'agit d'un bloc à part entière, il possède éventuellement un en-tête, une série de traitements, et une gestion des résultats tout comme l'algorithme qui le contient.

But

- Découper l'algorithme (action) en sous-algorithmes (sous-actions) plus simples,
- Mettre en commun les parties qui se répètent,
- Simplifier et faciliter la maintenance du code,
- Structurer et donner une meilleure lisibilité des programmes,
- Réutiliser dans d'autres algorithmes/programmes,
- …

Type de sous-algorithme

- Un sous-algorithme peut se présenter sous forme de *fonction* ou de *procédure*.

- Une ***fonction*** est un sous-algorithme qui, à partir de donnée(s), calcule et rend à l'algorithme Un et Un seul résultat alors qu'en général, une ***procédure*** affiche le(s) résultat(s) demandé(s).

Fonction

- Une fonction est un bloc d'instructions qui retourne obligatoirement une et une seule valeur résultat à l'algorithme appelant.

- Une fonction n'affiche jamais la réponse à l'écran car elle la renvoie simplement à l'algorithme appelant.

- Donc, Le rôle d'une fonction est similaire à celui d'une fonction en mathématique : elle ***retourne*** un résultat à partir des valeurs des paramètres.

Fonction

- Une fonction s'écrit en dehors du programme principal sous la forme :

```
Fonction nom_fonction (paramètres et leurs types) : type_fonction
        Instructions;
        retourne… ;
FinFonction
```

Fonction

- Pour le choix d'un nom de fonction il faut respecter les mêmes règles que celles pour les noms de variables,

- Type_fonction est le type du résultat retourné,

- L'instruction ***retourne*** sert à retourner la valeur du résultat.

Fonction

- Ecrire la fonction SommeCarre, qui calcule la somme des carrées de deux réels x et y?

- **Fonction** SommeCarre (x : réel, y: réel) : réel

  variable z : réel;

  z←x^2+y^2;

  retourne(z);

  FinFonction

Fonction

- Ecrire la fonction Pair qui détermine si un nombre est pair?

- **Fonction** Pair (n : entier) : booléen

  retourne(n%2=0);

  FinFonction

Appel de la Fonction

- L'utilisation d'une fonction se fera par simple écriture de son nom dans le programme principale.

- Le résultat étant une valeur, devra être affecté ou être utilisé dans une expression, une écriture, ...

Exemple (appel des fonctions précédentes)

```
Variables z : réel; b : booléen ;
Fonction SommeCarre (x : réel, y: réel ) : réel
        variable z : réel;
        z←x^2+y^2;
        retourne(z);
FinFonction
Fonction Pair (n : entier ) : booléen
        retourne(n%2=0);
FinFonction
Début
        b ←Pair(3);
        z←5*SommeCarre(7,2)+1;
        écrire("SommeCarre(3,5)= ", SommeCarre(3,5));
Fin
```

Procédure

- Dans certains cas, on peut avoir besoin de répéter une tâche dans plusieurs endroits du programme, mais que dans cette tâche on ne calcule pas de résultats.

- Dans ces cas, on ne peut pas utiliser une fonction, on utilise plutôt une ***procédure*****.**

Procédure

- Une *procédure* est un bloc d'instructions nommé et déclaré dans l'entête de l'algorithme et appelé dans son corps à chaque fois que le programmeur en a besoin.

- C'est un sous-programme semblable à une fonction mais qui ne retourne rien.

Procédure

- Une procédure s'écrit en dehors du programme principal sous la forme :

Procédure nom_procédure (paramètres et leurs types)

Instructions;

FinProcédure

- *Remarque :*
  - Une procédure peut ne pas avoir de paramètres.

Appel d’une Procédure

- L'appel d'une procédure, se fait dans le programme principale ou dans une autre procédure par une instruction indiquant le nom de la procédure :

Procédure exemple_proc (…)

…;

FinProcédure

Algorithme exepmleAppelProcédure

Début

exemple_proc (…) ;

…

Fin

Appel d’une Procédure

- *Remarque :*
  - Contrairement à l'appel d'une fonction, on ne peut pas affecter la procédure appelée ou l'utiliser dans une expression.
  - L'appel d'une procédure est une instruction autonome.

Variable Locale/Globale

- On peut manipuler 2 types de variables dans un module (procédure ou fonction) :
  - des ***variables locales*** et
  - des ***variables globales***.

- Elles se distinguent par ce qu'on appelle leur portée (leur "champ de définition", leur "durée de vie").

Variable Locale/Globale

- Une *variable locale* n'est connue qu'à l'intérieur du module ou elle a été définie. Elle est créée à l'appel du module et détruite à la fin de son exécution

- Une *variable globale* est connue par l'ensemble des modules et le programme principale. Elle est définie durant toute l'application et peut être utilisée et modifiée par les différents modules du programme

Variable Locale/Globale

- La manière de distinguer la déclaration des variables locales et globales diffère selon le langage.

- En général, les variables déclarées à l'intérieur d'une fonction ou procédure sont considérées comme variables locales.

- En pseudo-code, on va adopter cette règle pour les variables locales et on déclarera les variables globales dans le programme principale.

- **Conseil :**
  - Il faut utiliser autant que possible des variables locales plutôt que des variables globales. Ceci permet d'économiser la mémoire et d'assurer l'indépendance de la procédure ou de la fonction.

Exemples

1- Ecrire une fonction qui calcule le cube d'une valeur .

2- Ecrire une procédure qui calcule la somme des n premiers entiers.

– sur l'écran apparaît le message :

entrez la valeur de n :

si on entre 3, on obtient somme=6

Paramètres

- Soit l'exemple suivant:
  - *Définition de la Fonction:*

Fonction Pair (***n*** : entier) : booléen

retourne(n%2=0);

FinFonction

  - *Appel de la Fonction:*

b ←Pair(***3***);

- Lors de l'appel de la Fonction Pair(3); le paramètre ***formel n*** est remplacé par le paramètre ***effectif 3***.

Paramètres

- Les paramètres servent à échanger des données entre le programme principale (ou la procédure/fonction appelante) et la procédure/fonction appelée.

- Les paramètres placés dans la déclaration d'une procédure/fonction sont appelés ***paramètres formels***.

- Ces paramètres peuvent prendre toutes les valeurs possibles mais ils sont abstraits (n'existent pas réellement).

Paramètres

- Les paramètres placés dans l'appel d'une procédure/fonction sont appelés ***paramètres effectifs***.

- Ils contiennent les valeurs pour effectuer le traitement.

- Le nombre de paramètres effectifs doit être égal au nombre de paramètres formels. L'ordre et le type des paramètres doivent aussi correspondre.

À Suivre ….