Support de cours
Module : ASD1

Chapitre 4: Les sous-programmes

Réalisé par:
Dr. Sakka Rouis Taoufik

1

Introduction

- Pour maîtriser la complexité d'un problème, il est préférable de procéder à une conception modulaire qui consiste à le décomposer en plusieurs sous-problèmes. Ces sous-problèmes peuvent à leurs tours être décomposés jusqu'à aboutir à des traitements élémentaires simples. Ensuite, chaque sous-problème sera résolu par un sous-programme qui peut être une ***procédure*** ou une ***fonction***.
- Les sous-programmes développés peuvent être réutilisés plusieurs fois dans le même programme ou dans d'autres programmes une fois intégrés à des bibliothèques.

2

Procédure simple

- Une procédure est une structure de programme autonome qui permet d'effectuer une tâche bien précise et qui peut transmettre un certain nombre de résultats.
- La structure d'une procédure est analogue à celle d'un algorithme. Elle possède un entête, une partie déclarative et un corps.
- Pour appeler une **procédure** simple, il suffit d'écrire le nom de cette procédure.

3

Procédure simple

- Une **ressource locale** (privée) est déclarée dans une procédure et ne peut être utilisée qu'à l'intérieur de celle-ci.
- Une **ressource globale** (publique) est déclarée au début de l'algorithme. Elle peut être utilisée dans le corps principal de l'algorithme ou par les différents modules.
- Il est fortement recommandé d'utiliser autant que possible des variables locales pour rendre les modules plus autonomes et par conséquent utilisables dans n'importe quel programme.

4

Procédure simple

Exemple :

```
Algorithme Proc (* version 1 *)
Constantes
        titre = "Turbo C"
Procédure trait ( )
Variables
        i : Entier
Début
        Pour i de 1 à 7 Faire
                Ecrire("-")
        FinPour
FinProc

Début
        Ecrire(titre)
        trait ( )
Fin.
```

titre est une constante globale.

i est une variable locale à la procédure trait.

Dans ce cas, la procédure « trait » permet de souligner le titre "Turbo C".

5

Procédure paramétrée

- Lors de l'appel, il est souvent nécessaire d'échanger des informations entre la procédure et le programme ou le module appelant. C'est le rôle des *paramètres*.
- L'entête d'une procédure paramétrée comporte, en plus du nom de la procédure, une liste de **paramètres formels** (fictifs) sous la forme suivante :

  Procédure Nom_Proc (pf1 : type 1 ; pf2 : type 2 ; ...)
- Pour appeler une procédure paramétrée, il faut écrire le nom de la procédure suivi d'une liste de **paramètres effectifs** (réels) sous la forme suivante :

  Nom_Proc(pef1, pef2, ...)
- Lors de l'appel, il faut respecter le nombre, l'ordre et le type des paramètres.

6

Procédure paramétrée

```
Algorithme Proc (* version 2 *)
Variables
        titre : Chaîne
        L : Entier
Procédure Trait (n : entier)
Variables
        i : Entier
Début
        Pour i de 1 à n Faire
                Ecrire("-")
        FinPour
FinProc
Début
        Ecrire("Entrer un titre : "), Lire(titre)
        L← Long (titre)
        Ecrire(titre)
        Trait (L)
Fin.
```

Dans ce cas, la procédure « trait » permet d'afficher un trait dont la longueur effective est précisée lors de l'appel

7

Procédure paramétrée

- **Exercice**

Écrire une procédure « ReptCar » qui permet d'afficher un caractère un certain nombre de fois.

Exemple : l'appel de procédure **ReptCar("*",5)** doit afficher "*****".

8

Modes de passage de paramètres

- La substitution des paramètres effectifs aux paramètres formels s'appelle *passage de paramètres*. Elle correspond à un transfert d'informations entre le programme ou le module appelant et la procédure appelée.
- Notons d'abord que les paramètres d'une procédure peuvent être de type donnée (in), résultat (out) ou donnée-résultat (in/out).

Type de paramètre	Sens de transfert	Mode de passage
Donnée	Programme ⇒ Procédure	Par valeur
Résultat	Programme ⇐ Procédure	Par variable (par adresse)
Donnée-Résultat	Programme ⇔ Procédure	

9

Modes de passage de paramètres

- Un **paramètre donnée (in)** est une information nécessaire pour réaliser une tâche. Il ne subit aucun changement au cours de l'exécution de la procédure.
- Un **paramètre résultat (out)** est un aboutissement de l'action, sa valeur n'est pas significative avant le début de l'exécution de la procédure.
- Un **paramètre donnée-résultat (in/out)** est à la fois une donnée et un résultat, c'est à dire une information dont la valeur sera modifiée par la procédure.

Rq. En algorithmique, le mot **Var** inséré devant un paramètre formel signifie que ce paramètre est passé par variable (paramètre de type résultat ou donnée-résultat).

10

Modes de passage de paramètres

- **Exemple 1**

Écrire une procédure « puissance » qui calcule

$c = a^b = a * a * \ldots * a$ (b fois) ;

a et b étant des entiers positifs.

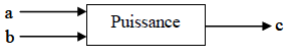
- **Exemple 2**

Écrire une procédure « permut » qui permet d'échanger les valeurs de 2 entiers a et b.

11

Modes de passage de paramètres

- **<u>Solution Exp. 1</u>**

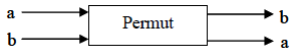


Procédure puissance (a, b : Entier ; **Var** c : Entier)
Variables
    i : Entier
Début
    C←1
    Pour i **de** 1 **à** b **Faire**
        c←c * a
    FinPour
FinProc.

12

Modes de passage de paramètres

- **Solution Exp. 2**



Procédure permut (**Var** a, b : Entier)

Variables

x : Entier

Début

x←a

a← b

b←x

FinProc.

13

Modes de passage de paramètres

- **Remarque**

Dans le mode de passage par valeur, les modifications effectuées sur le paramètre formel n'affectent pas la valeur du paramètre effectif (la procédure travaille sur une copie de la variable) alors que dans le mode de passage par variable, toute modification sur le paramètre formel affecte également le paramètre effectif correspondant (la procédure travaille sur la copie originale).

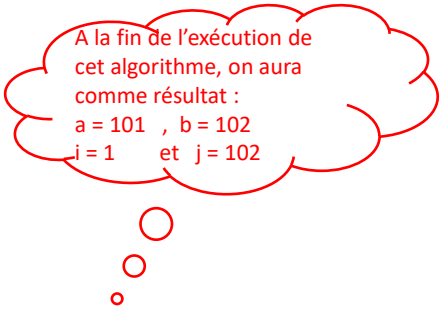
14

Modes de passage de paramètres

Exercice

Faire la trace d'exécution de l'algorithme suivant :

```
Algorithme Paramètre
Variables
        i, j : Entier
Procédure transmission (a : Entier ; Var b : Entier)
Début
        a ← a + 100
        b←b + 100
        Ecrire("a = ",a, "b = ",b)
FinProc
Début
        i←1
        J←2
        transmission(i,j)
        Ecrire("i = ",i, "j = ",j)
Fin.
```



15

Les fonctions

- Une fonction est une procédure particulière qui admet un seul paramètre résultat de type simple (entier, réel, caractère, etc.). Une fonction possède un type qui est celui de son résultat.
- Le nom de la fonction joue un double rôle, c'est à la fois l'identifiant de la fonction et une variable locale.
- Dans une fonction, le passage de tous les paramètres se fait **par valeur** (tous les paramètres sont des données).

16

Les fonctions

- Structure d'une fonction

Fonction Nom_Fonc (<paramètres formels>) : Type

    <Déclarations>

Début

    <Séquence d'instructions>

    Nom_Fonc ← résultat

FinFN

- Exemple

Écrire une fonction « min » qui retourne le minimum de 2 entiers a et b.

17

Les fonctions

- **Solution:**

Fonction min (a, b : Entier) : Entier

Variables

    c : Entier

Début

    Si (a <= b) **Alors**

        c← a

    Sinon

        c← b

    FinSi

    min←c

FinFN

18

Les fonctions

- L'appel d'une fonction se fait toujours dans une expression sous la forme :

  Nom_Fonc(<liste de paramètres effectifs>)

- **Exemples**

  x← 3

  y ← 5

  z ← min(x,y)

  z ← min(3,5)

  t ← (7 + min(x,y))/2

  Ecrire(min(x,y))

  Si (min(x,y) > 0) Alors …

- **Exercice**

Comment utiliser la fonction min pour afficher le minimum de 3 entiers x, y et z en une seule instruction et sans utiliser de variables supplémentaires ?

19

Exercices d'application

Exercice 1

Écrire une fonction PGCD_Euc qui retourne le PGCD de 2 entiers a et b en utilisant l'algorithme d'Euclide

L'algorithme d'Euclide consiste à répéter plusieurs fois le traitement :
PGCD(a,b) = PGCD(b,a Mod b) jusqu'à obtenir PGCD(x,0). Le PGCD est alors x.
Exemple : PGCD(36,16) = PGCD(16,4) = PGCD(4,0) = 4.

Exercice 2

Écrire une fonction *Triangle* qui permet de vérifier si les 3 nombres a, b et c peuvent être les mesures des côtés d'un triangle rectangle.

Remarque : D'après le théorème de Pythagore, si a, b et c sont les mesures des côtés d'un triangle rectangle, alors $a^2 = b^2 + c^2$ ou $b^2 = a^2 + c^2$ ou $c^2 = a^2 + b^2$.

20

Exercices d'application

Exercice 3

Écrire une fonction qui reçoit en arguments 2 nombres réels et un caractère et qui fournit un résultat correspondant à l'une des 4 opérations appliquées à ses deux premiers arguments, en fonction de la valeur du caractère, à savoir :

- addition pour le caractère +,

-soustraction pour –,

- multiplication pour *

-division pour /.

(Tout autre caractère que l'un des 4 cités sera interpréter comme une addition).

21

Exercices d'application

- **Exercice 4 :**

Ecrire un programme qui lit deux nombre naturel non nul *m et n* et qui détermine s'ils sont amis. Deux nombres entiers n et m sont qualifiés d'amis, si la somme des diviseurs de n est égale à m et la somme des diviseurs de m est égale à n (on ne compte pas comme diviseur le nombre lui-même et 1). Proposer une solution modulaire.

- **Exercice 5 :**

Écrire un algorithme d'une fonction qui recherche le premier nombre entier naturel dont le carré se termine par n fois le même chiffre.

Exemple : pour n = 2, le résultat est 10 car 100 se termine par 2 fois le même chiffre.

22