Université Internationale de Casablanca

Cours Algorithmique: Procédures & Fonctions (Suite) Tableaux

Sanae EL OUKKAL

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.

Transmission des Paramètres

- Il existe deux modes de transmission de paramètres dans les langages de programmation :
  - ***La transmission par valeur*** : les valeurs des paramètres effectifs sont affectées aux paramètres formels correspondants au moment de l'appel de la procédure. Dans ce mode le paramètre effectif ne subit aucune modification
  - **La transmission par adresse (ou par référence) :** les adresses des paramètres effectifs sont transmises à la procédure appelante. Dans ce mode, le paramètre effectif subit les mêmes modifications que le paramètre formel lors de l'exécution de la procédure

Transmission des Paramètres

- ***Remarque :***
  - Le paramètre effectif doit être une variable (et non une valeur) lorsqu'il s'agit d'une transmission par adresse.

- En pseudo-code, on va préciser explicitement le mode de transmission dans la déclaration de la procédure.

Exemple

```
Variables a, b : entier;
Procédure SomCar (x, y: entier)
        variable som:entier;
        x ← x*x ;
        y ← y*y ;
        som ← x + y ;
         Ecrire (x, " + ", y, " = ", som) ;
Fin Procédure

Début
        a ← 3 ;
        b ← 4 ;
         SomCar(a, b) ;
Fin
```

- Après exécution, la valeur de a est 3 et la valeur de b est 4.

Exemple

```
Variables a, b : entier;
Procédure SomCar (x, y: entier par référence)
        variable som:entier;
        x ← x*x ;
        y ← y*y ;
        som ← x + y ;
         Ecrire (x, " + ", y, " = ", som) ;
Fin Procédure

Début
        a ← 3 ;
        b ← 4 ;
         SomCar(a, b) ;
        ecrire (a);
        ecrire (b);
Fin
```

- Après exécution, la valeur de a est 9 et la valeur de b est 16.

TABLEAUX

Exemple Introductif

- Supposons qu'on veut conserver les notes d'une classe de 30 étudiants pour extraire quelques informations. Par exemple : calcul du nombre d'étudiants ayant une note supérieure à 10.

- Le seul moyen dont nous disposons actuellement consiste à déclarer 30 variables, par exemple **N1, …, N30.**

- Après 30 instructions lire, on doit écrire 30 instructions Si pour faire le calcul:

Exemple Introductif

```
nbre ← 0;
Si (N1 >10) alors
    nbre ←nbre+1;
FinSi
….
Si (N30>10) alors
    nbre ←nbre+1;
FinSi
```

c'est lourd à écrire

- Heureusement, les langages de programmation offrent la possibilité de rassembler toutes ces variables dans une seule structure de donnée appelée ***tableau*****.**

Tableaux

- Un tableau est un ensemble d'éléments de même type désignés par un identificateur unique.

- Une variable entière nommée ***indice*** permet d'indiquer la position d'un élément donné au sein du tableau et de déterminer sa valeur.

- Un tableau se définit en indiquant ***son nom***, le ***type*** des éléments stockés dans le tableau, ainsi que leur ***nombre***, écrit entre crochets. Ce nombre désigne la taille maximale du tableau.

Tableaux

- *En pseudo code:*

  Variable Tableau identificateur**[dimension]** : **type;**

- ***Exemple:***

  Variable Tableau notes[**30**] : **réel;**

- On peut définir des tableaux de tous types: tableaux d'entiers, de réels, de caractères, de booléens, de chaînes de caractères, …

Tableau: Remarque

- L'accès à un élément du tableau se fait au moyen de l'indice. Par exemple, ***notes[i]*** donne la valeur de l'élément ***i*** du tableau notes.

- Selon les langages, le premier indice du tableau est soit 0, soit 1. Le plus souvent c'est 0 (c'est ce qu'on va adopter en pseudo-code). Dans ce cas, ***notes[i]*** désigne l'élément ***i+1*** du tableau notes.

- Il est possible de déclarer un tableau sans préciser au départ sa dimension. Cette précision est faite ultérieurement.

Tableaux et Les Boucles

- Les boucles sont extrêmement utiles pour les algorithmes associés aux tableaux.
- En effet, de nombreux algorithmes relatifs au tableau nécessitent de parcourir les éléments du tableau dans un certain ordre.
- Le traitement de chacun des éléments étant souvent le même, seule la valeur de l'indice est amenée à changer.
- Une boucle est donc parfaitement adaptée à ce genre de traitements.

Exemple

- Pour le calcul du nombre d'étudiants ayant une note supérieure à 10 avec les tableaux, on peut écrire ?

```
Variables i ,nbre : entier ;
            Tableau notes[30] : réel;
Début
      nbre ← 0;
      Pour i allant de 0 à 29
          Si (notes[i]>10) alors
                nbre ←nbre+1 ;
          FinSi
      FinPour
      écrire ("le nombre de notes supérieures à 10 est : ", nbre);
Fin
```

Saisir Un Tableau

- Instructions qui permettent de saisir les éléments d'un tableau :

```
Variable i : entier;
Pour i allant de 0 à n-1
    écrire ("Saisie de l'élément ", i + 1);
    lire (T[i]);
FinPour
```

Afficher Un Tableau

- Instructions qui permettent de saisir les éléments d'un tableau :

```
Variable i : entier;
Pour i allant de 0 à n-1
    écrire ("T[",i, "] =", T[i]);
FinPour
```

Tableau:
Remarque

- On peut par conséquent utiliser des procédure qui permettent de saisir et afficher un tableau.

- Quand on déclare un tableau comme paramètre d'une procédure, on peut ne préciser sa dimension qu'au moment de l'appel.

- En tous cas, un tableau est inutilisable tant qu’on n’a pas précisé le nombre de ses éléments.

- Un grand avantage des tableaux est qu'on peut traiter les données qui y sont stockées de façon simple en utilisant des boucles.

Les Procédures

Procédure SaisieTab (n : entier par valeur, tableau T : réel par référence)

Variable i : entier;

Pour i allant de 0 à n-1

écrire ("Saisie de l'élément ", i + 1);

lire (T[i]);

FinPour

Fin Procédure

Procédure AfficheTab (n : entier par valeur, tableau T : réel par valeur)

Variable i : entier;

Pour i allant de 0 à n-1

écrire ("T[",i, "] =", T[i]);

FinPour

Fin Procédure

Appel des Procédures

- Le corps de l'algorithme où on fait l'appel des procédures SaisieTab et AfficheTab :

```
Variable p : entier;
            Tableau A[10]: réel;
Début
    p ← 10;
    SaisieTab(p, A);
    AfficheTab(10,A);
Fin
```

Tableaux : 2 Problèmes Classiques

- **Tri d'un tableau**
  - Tri par sélection
  - Tri par insertion
  - …..

- **Recherche d'un élément dans un tableau**
  - Recherche séquentielle
  - Recherche dichotomique

Tri d'un tableau

- Le tri consiste à ordonner les éléments du tableau dans l'ordre croissant ou décroissant.

- Il existe plusieurs algorithmes connus pour trier les éléments d'un tableau :
  - Le tri par sélection
  - Le tri par insertion
  - Le tri à bulles
  - …

Tri par Sélection

- *Principe:*
  - à l'étape i, on sélectionne le plus petit élément parmi les (n -i +1) éléments du tableau les plus à droite.
  - On l'échange ensuite avec l'élément i du tableau.

Tri par Insertion

- Le tri par insertion est le tri le plus connu.

- C'est celui que les gens utilisent intuitivement quand ils doivent trier une liste d'objets, par exemple quand on joue aux cartes.

Tri par Insertion

- *Principe:*
  - On fait comme si les éléments à trier étaient donnés un par un, le premier élément constituant, à lui tout seul, une liste triée de longueur 1.

  - On range ensuite le second élément pour constituer une liste triée de longueur 2, puis on range le troisième élément pour avoir une liste triée de longueur 3 et ainsi de suite...

Tri à Bulles

- *Principe:*
  - Il consiste à comparer répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils sont mal triés.
  - Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

Recherche Séquentielle/Par Balayage/Simple/Linéaire

- Le r*echerche séquentielle* ou *recherche linéaire* est un algorithme pour trouver une valeur dans une liste.

- Elle consiste simplement à considérer les éléments de la liste les uns après les autres, jusqu'à ce que l'élément soit trouvé, ou que toutes les cases aient été lues. Elle est aussi appelée *recherche par balayage*.

Recherche Dichotomique

- Regardez le documents pdf : Tri & Recherche.

À Suivre ….