UNIVERSITE VIRTUELLE DU SENEGAL

foo nekk foofu la

Algorithme et structure de données

Dr Ndeye Massata NDIAYE

Séquence 2 : Les tableaux

Les tableaux

Objectifs spécifiques

- créer un tableau
- manipuler un tableau
- utiliser un tableau dans un algorithme

1 Notion de tableau

Soit un entier n positif.

Supposons qu'on veuille saisir n valeurs réelles afin de calculer leur moyenne, ou de trouver leur minimum, ou de les afficher par ordre croissant.

Pour des valeurs petites de n, on peut déclarer n variables réelles pour résoudre le problème. Mais si n est assez grand, on se rend compte que cela devient impropre, fastidieux, voire impossible.

Il faudrait, dans ce cas, utiliser une variable permettant de représenter les n valeurs. Le type de données de cette variable serait le type tableau.

Définition :

Un tableau est une collection séquentielle d'éléments de même type, où chaque élément peut être identifié par sa position dans la collection. Cette position est appelée indice et doit être de type scalaire.

Déclaration :

Pour déclarer un tableau, il faut donner :

- son nom (identificateur de la variable)
- ses bornes : la borne inférieure correspondant à l'indice minimal et la borne supérieure correspondant à l'indice maximal.
- le type des éléments le composant.

Syntaxe :

type

nom = **tableau[**<indice minimum> **..** <indice maximum>**] de** <type des composants>

ou

Variable

nom : **tableau[**<indice minimum> **..** <indice maximum>**] de** <type des composants>

Exemple :

variable t : tableau[1..10] de réels

Schématiquement, on va représenter la variable t comme suit:

1	2	3	4	5	6	7	8	9	10
8.4	3.5	12	20	10	13.34	50	100	30.1	60.9

Dans la mémoire centrale, les éléments d'un tableau sont stockés de façon linéaire, dans des zones contiguës.

Le tableau ci-dessus est de dimension 1, nous verrons un peu plus loin que l'on peut représenter des tableaux à 2 dimensions, voire même plus.

L'élément n° I sera représenté par l'expression t[I].

Dans notre exemple, t[I] peut être traité comme une variable réelle. On dit que le tableau t est de taille 10.

2 Création d'un tableau

La création d'un tableau consiste au remplissage des cases le composant. Cela peut se faire par saisie, ou par affectation.

Par exemple, pour remplir le tableau t précédent, on peut faire :

t[1] <- 8.4 ; t[2] <- 3.5 ; … t[10] <- 60.9 ;

Si on devait saisir les valeurs, il faudrait écrire :

Pour i <- 1 à 10 faire lire(t[i]);

3 Affichage d'un tableau

Afficher un tableau revient à afficher les différents éléments qui le composent. Pour cela, on le parcourt (généralement à l'aide d'une boucle avec compteur) et on affiche les éléments un à un.

Exercice d'application

Ecrire un programme qui permet de créer un tableau d'entiers t1 de taille 20 par saisie, et un tableau t2 de même taille en mettant dans t2[i] le double de t1[i], i ∈ {1, .., 20 }.

```
Type tab = tableau[1..20] d'entier;
variables t1, t2 : tab ;
        i : entier;

debut
        { saisie de t1 }
        pour i <- 1 à 20 faire
        debut
                ecrire('donner t1[',i,']:'); lire(t1[i]);
        fin;
        { création de t2 }
        pour i <-1 à 20 faire
        t2[i] <- 2*t1[i];
        { affichage de t2 }
        pour i <-1 à 20 faire
        ecrire('t2[',i,']=', t2[i]);
fin.
```

4 Traitement d'un tableau

Après avoir créé un tableau, on peut y effectuer plusieurs opérations comme le calcul de la somme ou de la moyenne des éléments, la recherche du plus petit ou du grand élément du tableau, le test d'appartenance d'un objet au tableau, …

Pour la suite, on considère un tableau d'entiers t déclaré comme suit :

```
variable t : array[1 .. n] of entier ;
```

4.1 Somme des éléments d'un tableau

On effectue la somme des éléments du tableau t, le résultat est dans la variable S :

```
S <- 0 ;
pour i <-1 à n faire
        S <- S + t[i];
ecrireln('la somme des éléments de t est:', S);
```

4.2 Minimum d'un tableau

On cherche le plus petit élément du tableau t, le résultat est dans la variable min :

```
min <- t[1] ;
pour i <-2 à 10 faire
    si t[i] < min alors  min <- t[i];
ecrire('Le minimum des elements de t est:', min) ;
```

4.3 Test d'appartenance

On cherche si l'entier x appartient à t, le résultat est mis dans la variable booléenne appartient :

```
appartient <- faux
pour i<-1 à n faire
        si t[i]=x alors   appartient <- true;
si appartient alors
        ecrire('x appartient à t')
sinon  ecrire('x n''appartient pas à t');
```

On remarque que l'on peut arrêter les itérations (la recherche) si l'on rencontre l'élément x dans t. Pour cela, il faut utiliser une boucle *Tant Que* ou *Repeat*:

```
i <- 1;
```

```
tant que (t[i]<>x) and (i<=n) faire
    i<-i+1;
```

```
si i>n alors
      ecrire(‘x n’’appartient pas à t’)
sinon
      ecrire(‘x appartient à t’);
```

```
i <- 0 ;
Repetr
          i<-i+1;
Jusqu’à (t[i]=x) ou (i>n);
si i>n alors
      ecrire(‘x n’’appartient pas à t’)
sinon
      ecrire(‘x appartient à t’);
```

Dans les deux cas, si x appartient à t, la valeur de i est l'indice de la case qui le contient.

4.4 Ajout d'un élément

On suppose que le tableau t est « rempli » et qu'il reste une case non occupée à la fin (la $n^{ième}$ case).

Si on veut alors ajouter un entier x à la fin du tableau, il suffira d'écrire

```
t[n] <- x ;
```

Mais, si on veut ajouter x dans une case dont l'indice k est différent de n, il faudra décaler les éléments t[k], t[k+1], …, t[n-1] vers la droite pour libérer la case d'indice k :

```
pour i <- n à k+1 faire
                    t[i] <- t[i-1];
t[k] <- x;
```

4.5 Suppression d'un élément

Pour supprimer l'élément se trouvant à la position k de t, on l'écrase en faisant décaler les éléments placés après lui vers la gauche :

```
pour i <- k à n faire
                    t[i] <- t[i+1];
```

Il faut noter qu'après une telle opération, l'élément se trouvant à la dernière position (n) n'est plus significatif.

5 Les tableaux à deux dimensions

Pour traiter les notes obtenues par un étudiant à 10 épreuves on peut utiliser un tableau de 10 réels. Pour traiter les notes obtenues par 5 étudiants, on pourrait utiliser 5 tableaux de 10 réels chacun.

Mais puisqu'on va effectuer très probablement les mêmes traitements sur ces tableaux, il est préférable de les regrouper dans une seule variable qui sera un tableau de 5 lignes et 10 colonnes.

Chaque élément de ce tableau multidimensionnel sera identifié par deux indices : la position indiquant la ligne et la position indiquant la colonne.

Déclaration :

```
variable t : tableau[1..5, 1..10] de reels ;
```

Pour accéder à l'élément se trouvant sur la ligne i et la colonne j, on utilise le terme **t[i,j]**.

Exercice d'application

Ecrire un programme qui permet de créer (par saisie) et d'afficher un tableau t à deux dimensions d'entiers de taille 3x5.

```
Type TAB_3_5 = array[1..3, 1..5] d'entier ;
variable t : TAB_3_5 ;
        i,j : entier ;
debut
    (* saisie de t *)
    pour i <-1 à 3 faire
        pour j <-1 à 5 faire
        debut
            ecrire('fairenner t[',i,',',j,']:'); lire(t[i]);
        fin;
    (* affichage de t *)
    pour i <-1 à 3 faire
    debut
        pour j <-1 à 5 faire
            ecrire(t[i,j],'  ');
        ecrire;
    fin;
fin.
```