Les tableaux

Ch. Lasou, N.E. Oussous, E. Wegrzynowski

Licence ST-A, USTL - API1

29 janvier 2007

Motivation

Nombre de problèmes nécessitent de travailler avec des ensembles de données homogènes

Motivation

Nombre de problèmes nécessitent de travailler avec des ensembles de données homogènes

- séries statistiques

Motivation

Nombre de problèmes nécessitent de travailler avec des ensembles de données homogènes

- séries statistiques
- vecteurs

Motivation

Nombre de problèmes nécessitent de travailler avec des ensembles de données homogènes

- séries statistiques
- vecteurs
- tables de fonctions

Motivation

Nombre de problèmes nécessitent de travailler avec des ensembles de données homogènes

- séries statistiques
- vecteurs
- tables de fonctions
- etc ...

Objectifs

- pouvoir désigner et manipuler un ensemble de données d'un même type

Objectifs

- pouvoir désigner et manipuler un ensemble de données d'un même type
- tout en ayant un accès direct à chacun de ses éléments.

Définition intuitive

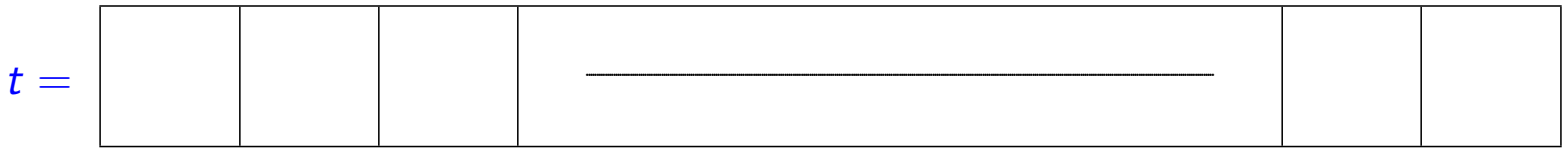
Un tableau t est

$t =$

Définition intuitive

Un tableau t est

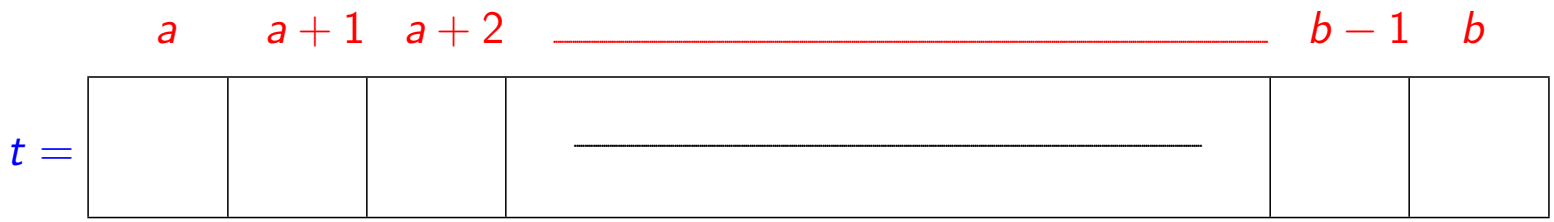
- une collection de cases (mémoires)



Définition intuitive

Un tableau t est

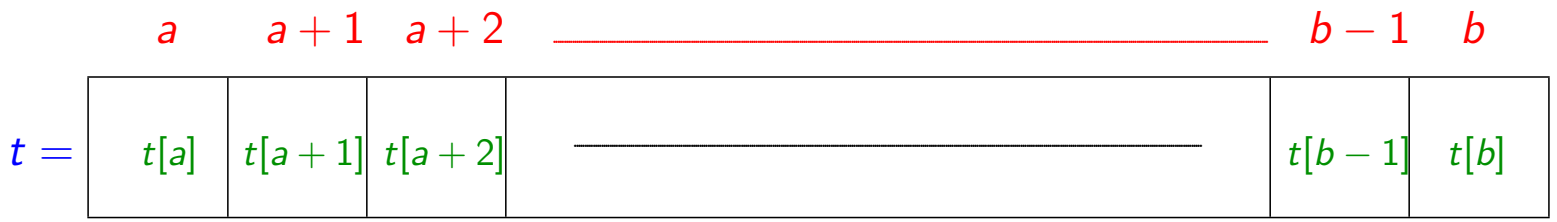
- une collection de cases (mémoires)
- désignées par des indices : a, $a+1$, ..., b



Définition intuitive

Un tableau t est

- une collection de cases (mémoires)
- désignées par des indices : a, $a+1$, ..., b
- contenant des valeurs, toutes du même type : $t[a]$, ..., $t[b]$



Définition formelle d'un tableau

Définition

Soient $I = [\![a, b]\!]$ un intervalle et E un type de données quelconque. Un tableau d'éléments de type E est une donnée composée d'éléments de type E, chacun d'entre eux étant directement accessible par un indice $k \in [\![a, b]\!]$.

Notations

1. élément d'indice k d'un tableau t noté $t[k]$. C'est un élément de E. Il n'est défini que si $k \in [\![a, b]\!]$.

Notations

1. élément d'indice k d'un tableau t noté $t[k]$. C'est un élément de E. Il n'est défini que si $k \in [\![a, b]\!]$.
2. Si $c, d \in [\![a, b]\!]$ et $c \leq d$, $t[c..d]$ = sous-tableau des éléments de t d'indices compris entre c et d.

Notations

1. élément d'indice k d'un tableau t noté $t[k]$. C'est un élément de E. Il n'est défini que si $k \in [\![a, b]\!]$.
2. Si $c, d \in [\![a, b]\!]$ et $c \leq d$, $t[c..d]$ = sous-tableau des éléments de t d'indices compris entre c et d.
3. Si $x \in E$, $x \in t$ signifie $\exists k \in [\![a, b]\!]\ x = t[k]$

Notations

1. élément d'indice k d'un tableau t noté $t[k]$. C'est un élément de E. Il n'est défini que si $k \in [\![a, b]\!]$.
2. Si $c, d \in [\![a, b]\!]$ et $c \leq d$, $t[c..d]$ = sous-tableau des éléments de t d'indices compris entre c et d.
3. Si $x \in E$, $x \in t$ signifie $\exists k \in [\![a, b]\!]\ x = t[k]$
4. Si $x \in E$, $x \notin t$ signifie $\forall k \in [\![a, b]\!]\ x \neq t[k]$

Notations

1. élément d'indice k d'un tableau t noté $t[k]$. C'est un élément de E. Il n'est défini que si $k \in [\![a, b]\!]$.
2. Si $c, d \in [\![a, b]\!]$ et $c \leq d$, $t[c..d]$ = sous-tableau des éléments de t d'indices compris entre c et d.
3. Si $x \in E$, $x \in t$ signifie $\exists k \in [\![a, b]\!]\ x = t[k]$
4. Si $x \in E$, $x \notin t$ signifie $\forall k \in [\![a, b]\!]\ x \neq t[k]$
5. S'il existe une relation d'ordre sur E, notée $\leq$, et si $x \in E$, $x \leq t$ signifie $\forall k \in [\![a, b]\!]\ x \leq t[k]$

Avec Free Pascal, possibilité de définir deux sortes de tableaux

1. les tableaux statiques : la *taille* (ou *longueur*) du tableau est fixée une fois pour toute lors de l'écriture du programme, et le compilateur se charge de réserver l'emplacement mémoire pour chaque variable de ce type ;

Avec Free Pascal, possibilité de définir deux sortes de tableaux

1. les tableaux statiques : la *taille* (ou *longueur*) du tableau est fixée une fois pour toute lors de l'écriture du programme, et le compilateur se charge de réserver l'emplacement mémoire pour chaque variable de ce type ;
2. les tableaux dynamiques : seul le type des éléments est fixé lors de l'écriture du programme. La taille est déterminée à l'exécution du programme et peut varier durant celle-ci.

Avec Free Pascal, possibilité de définir deux sortes de tableaux

1. les tableaux statiques : la *taille* (ou *longueur*) du tableau est fixée une fois pour toute lors de l'écriture du programme, et le compilateur se charge de réserver l'emplacement mémoire pour chaque variable de ce type ;
2. les tableaux dynamiques : seul le type des éléments est fixé lors de l'écriture du programme. La taille est déterminée à l'exécution du programme et peut varier durant celle-ci.

Seuls les tableaux statiques seront présentés et utilisés.

Les tableaux statiques

- possibilité de définir des tableaux d'éléments de tout type

Les tableaux statiques

- possibilité de définir des tableaux d'éléments de tout type
- les indices sont définis par un intervalle d'un type ordinal

Le mot-clé **array**

Syntaxe de la déclaration en PASCAL

Un tableau est déclaré à l'aide des mots-clés **array** et **of**

array[<indice>] **of** <element>

Le mot-clé array

Syntaxe de la déclaration en PASCAL

Un tableau est déclaré à l'aide des mots-clés **array** et **of**

```
array[<indice>] of <element>
```

Exemples

1. Tableau de 12 entiers

```
type   MOIS = 1..12 ;
       NB_JOURS = array[MOIS] of CARDINAL ;
```

Le mot-clé array

Syntaxe de la déclaration en PASCAL

Un tableau est déclaré à l'aide des mots-clés **array** et **of**

```
array[<indice>] of <element>
```

Exemples

1. Tableau de 12 entiers

```
type   MOIS = 1..12 ;
       NB_JOURS = array[MOIS] of CARDINAL ;
```

2. Tableau de 26 chaînes de caractères

```
type LETTRES = 'A'..'Z';
     T_STRING = array[LETTRES] of STRING ;
```

La fonction low

Présentation

Si *t* est une variable d'un type tableau, la fonction **low** donne la valeur de l'indice du premier élément du tableau.

La fonction low

Présentation

Si t est une variable d'un type tableau, la fonction **low** donne la valeur de l'indice du premier élément du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

$$\textbf{low}(t) = 1$$

La fonction low

Présentation

Si t est une variable d'un type tableau, la fonction **low** donne la valeur de l'indice du premier élément du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

   $$\textbf{low}(t) = 1$$

2. Si t est de type T_STRING, alors

   $$\textbf{low}(t) = \text{'A'}$$

La fonction high

Présentation

Si t est une variable d'un type tableau, la fonction **high** donne la valeur de l'indice du dernier élément du tableau.

La fonction high

Présentation

Si t est une variable d'un type tableau, la fonction **high** donne la valeur de l'indice du dernier élément du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

$$\mathbf{high}(t) = 12$$

La fonction high

Présentation

Si t est une variable d'un type tableau, la fonction **high** donne la valeur de l'indice du dernier élément du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

   $$\textbf{high}(t) = 12$$

2. Si t est de type T_STRING, alors

   $$\textbf{high}(t) = \text{'Z'}$$

La fonction **length**

Présentation

Si *t* est une variable d'un type tableau, la fonction **length** donne la longueur (taille, nombre de cases) du tableau.

La fonction **length**

Présentation

Si t est une variable d'un type tableau, la fonction **length** donne la longueur (taille, nombre de cases) du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

$$\textbf{length}(t) = 12$$

La fonction length

Présentation

Si t est une variable d'un type tableau, la fonction **length** donne la longueur (taille, nombre de cases) du tableau.

Exemples

1. Si t est de type NB_JOURS, alors

   $$\textbf{length}(t) = 12$$

2. Si t est de type T_STRING, alors

   $$\textbf{length}(t) = 26$$

Notation indicielle

Présentation

Si t est une variable d'un type tableau indexé par l'intervalle $[\![a, b]\!]$, et $i \in [\![a, b]\!]$, l'élément d'indice i de t est désigné en PASCAL par `t[i]`. `t[i]` peut être considéré comme une variable du type des éléments de t.

Notation indicielle

Présentation

Si t est une variable d'un type tableau indexé par l'intervalle $[\![a, b]\!]$, et $i \in [\![a, b]\!]$, l'élément d'indice i de t est désigné en PASCAL par t[i]. t[i] peut être considéré comme une variable du type des éléments de t.

Exemples

1. Si t est de type NB_JOURS, et $i \in [\![1, 12]\!]$ alors t[i] est de type CARDINAL.

Notation indicielle

Présentation

Si t est une variable d'un type tableau indexé par l'intervalle $[\![a, b]\!]$, et $i \in [\![a, b]\!]$, l'élément d'indice i de t est désigné en PASCAL par t[i]. t[i] peut être considéré comme une variable du type des éléments de t.

Exemples

1. Si t est de type NB_JOURS, et $i \in [\![1, 12]\!]$ alors t[i] est de type CARDINAL.
2. Si t est de type T_STRING, et $i \in [\![' A', ' Z']\!]$ alors t[i] est de type **STRING**.

Remarque

Toute tentative d'accès à un élément d'un tableau à l'aide d'un indice dont la valeur est en dehors de l'intervalle provoque une erreur

Remarque

Toute tentative d'accès à un élément d'un tableau à l'aide d'un indice dont la valeur est en dehors de l'intervalle provoque une erreur

1. à la compilation (si le compilateur est en mesure de détecter cette tentative). Le message d'erreur est alors

```
Error : range check error while evaluating constants
```

Remarque

Toute tentative d'accès à un élément d'un tableau à l'aide d'un indice dont la valeur est en dehors de l'intervalle provoque une erreur

1. à la compilation (si le compilateur est en mesure de détecter cette tentative). Le message d'erreur est alors

```
Error : range check error while evaluating constants
```

2. ou à l'exécution. Le message est alors

```
Runtime error 201 at ...
```

Attribution d'une valeur à un élément

Exemple

Schéma classique d'initialisation d'un tableau

```
for i := low(t) to high(t) do
begin
   donner une valeur à t[i]
end {for};
```

où « donner une valeur à t[i] » peut être

Attribution d'une valeur à un élément

Exemple

Schéma classique d'initialisation d'un tableau

```
for i := low(t) to high(t) do
begin
   donner une valeur à t[i]
end {for};
```

où « donner une valeur à t[i] » peut être

- t[i] := <expression>

Attribution d'une valeur à un élément

Exemple

Schéma classique d'initialisation d'un tableau

```
for i := low(t) to high(t) do
begin
   donner une valeur à t[i]
end {for};
```

où « donner une valeur à t[i] » peut être

- t[i] := <expression>
- ou lire (t[i]), si lire est une procédure de lecture de valeurs du type des éléments de t.

Constante typée

Possibilité de définir des constantes de type tableau : constantes typées.
Les valeurs sont énumérées entre deux parenthèses

Constante typée

Possibilité de définir des constantes de type tableau : constantes typées.
Les valeurs sont énumérées entre deux parenthèses

Exemple

```
const
    C_NB_JOURS : NB_JOURS = (31,28,31,30,31,30,
                             31,31,30,31,30,31);
```

Affectation de tableaux

Affectation de tableaux

Il est possible d'affecter à une variable de type tableau, toute valeur produite par une expression du même type tableau.

Affectation de tableaux

Affectation de tableaux

Il est possible d'affecter à une variable de type tableau, toute valeur produite par une expression du même type tableau.

Exemple

Si $t1$ et $t2$ sont deux tableaux du même type,

```
t1 := t2 ;
```

est une instruction valide.

Remarque

La comparaison de deux tableaux à l'aide de l'opérateur = n'est pas autorisée par le compilateur.

Remarque

La comparaison de deux tableaux à l'aide de l'opérateur = n'est pas autorisée par le compilateur.
Par exemple, si $t1$ et $t2$ sont deux variables du même type tableau, alors la condition dans l'instruction

```
if t1=t2 then ...
```

provoque l'arrêt de la compilation du programme avec le message

Error : Operator is not overloaded

Remarque

La comparaison de deux tableaux à l'aide de l'opérateur = n'est pas autorisée par le compilateur.
Par exemple, si $t1$ et $t2$ sont deux variables du même type tableau, alors la condition dans l'instruction

```
if t1=t2 then ...
```

provoque l'arrêt de la compilation du programme avec le message

Error : Operator is not overloaded

Si on veut tester l'égalité de deux tableaux, il faut le programmer !

Trois utilisations classiques des tableaux

Trois utilisations classiques des tableaux

- stockage de données ;

Trois utilisations classiques des tableaux

- stockage de données ;
- tabulation de fonctions ;

Trois utilisations classiques des tableaux

- stockage de données ;
- tabulation de fonctions ;
- représentation de données structurées.

Stockage de données

Utilisation des tableaux pour mémoriser un ensemble de données de même type :

Stockage de données

Utilisation des tableaux pour mémoriser un ensemble de données de même type :

- nombre d'occurrences des lettres dans un texte ;

Stockage de données

Utilisation des tableaux pour mémoriser un ensemble de données de même type :

- nombre d'occurrences des lettres dans un texte ;
- notes obtenues à un examen par un groupe d'étudiants

Stockage de données

Utilisation des tableaux pour mémoriser un ensemble de données de même type :

- nombre d'occurrences des lettres dans un texte ;
- notes obtenues à un examen par un groupe d'étudiants
- etc. . .

Notes

Déclaration d'un type pour la gestion des notes d'un groupe de 20 étudiants.

Notes

Déclaration d'un type pour la gestion des notes d'un groupe de 20 étudiants.

```
const N = 20;
type NOTES = array[1..N] of REAL;
```

Exemple de traitement

Saisie des notes depuis le clavier

```
var
    mesnotes : NOTES;
    i : CARDINAL;

...

for i := 1 to N do
begin
  readln(mesnotes[i]);
end {for};
```

Calcul de la moyenne

Moyenne des notes

```
var
    mesnotes : NOTES;
    i : CARDINAL;
    s, moyenne : REAL;

    ...

s := 0;
for i := 1 to N do
begin
  s := s + mesnotes[i];
end {for};
moyenne := s / N;
```

Tabulation

Utilisation des tableaux pour tabuler des fonctions, i.e. utiliser des tables de valeurs de fonctions

Tabulation

Utilisation des tableaux pour tabuler des fonctions, i.e. utiliser des tables de valeurs de fonctions

- pour des fonctions prédéfinies (afin d'économiser des calculs par exemple)

Tabulation

Utilisation des tableaux pour tabuler des fonctions, i.e. utiliser des tables de valeurs de fonctions

- pour des fonctions prédéfinies (afin d'économiser des calculs par exemple)
- pour de nouvelles fonctions

Tables de sinus

Déclaration d'une table pour les sinus de multiples de $\frac{\pi}{N}$

```
const N = 20;
type
  TABLE = array[0..2*N-1] of REAL;
var
  tabsin : TABLE;
```

Construction de la table des sinus

Calcul de $\sin(k\pi/N)$ pour $k \in [\![0, 2N-1]\!]$

```
for k := 0 to 2*N-1 do
begin
    tabsin[k] := sin(k*Pi/N);
end {for};
```

Nom d'un jour

Une fonction de conversion de valeurs de type JOUR en valeurs de type **STRING**

```
type
  JOUR = (LUNDI, MARDI, MERCREDI, JEUDI,
          VENDREDI, SAMEDI, DIMANCHE);


  function nomDuJour(j : JOUR) : STRING;
  const
    NOMS : array[JOUR] of STRING =
      ('LUNDI', 'MARDI', 'MERCREDI', 'JEUDI',
       'VENDREDI', 'SAMEDI', 'DIMANCHE');
  begin
    nomDuJour := noms[j];
  end {nomDuJour};
```

Représentation de données

Utilisation de tableaux pour représenter des données

Représentation de données

Utilisation de tableaux pour représenter des données

- par exemple des vecteurs,

Vecteurs de $\mathbb{R}^3$

Déclaration d'un type pour représenter des vecteurs de $\mathbb{R}^3$

```
const DIM = 3;
type
    VECTEUR = array[1..DIM] of REAL;
```

Exemple de traitement sur les vecteurs

Calcul de la somme de deux vecteurs

```
function somme(v1,v2 : VECTEUR) : VECTEUR;
var
   v : VECTEUR;
   i : 1..DIM;
begin
   for i := 1 to DIM do
   begin
      v[i] := v1[i] + v2[i];
   end {for};
   somme := v;
end {somme};
```