Plan du cours

1) Éléments de base
2) Présentation du formalisme algorithmique
3) Éléments de base du langage PASCAL
4) Modularité
5) Structures de données statiques

Enregistrements (Objets composés)

Généralités

- **1. Définition**
- Une variable de type **enregistrement** est une variable structurée avec plusieurs champs.
- Les **champs** sont les attributs ou caractéristiques de l'enregistrement. Ils sont parfois appelés rubriques ou propriétés.

- **2. Déclaration**
- Syntaxe

```
type
dentificateur = record <liste de champs> ;
end;
```

- Syntaxe : Spécification d'un champ dans la déclaration

```
identificateur : type_champ ;
identificateur1,  identificateur2,  ..........  :
type_champ ;
```

Enregistrements (Objets composés)

- Exemple

```
type
        personne = record
                nom     :string[40];
                prenom  :string[50];
                age     :integer;
        end;
        voiture = record
                marque : string ;
                cylindree : real ;
                couleur : string;
                nom : string ;
                prix : integer ;
        end ;
        une_couleur = (trefle, carreau, coeur, pique);
        une_valeur = (sept, huit, ..., dame, roi, as);
        carte = record
                couleur : une_couleur;
                valeur : une_valeur;
        end;
```

Enregistrements (Objets composés)

- **3.Accès aux champs**
- Il faut ensuite déclarer les variables associées. Soit v une variable de type enregistrement. Pour **accéder** a un champ de cet enregistrement, il suffit simplement d'écrire :

```
v.identificateur_du_champ
```

- **Exemple**
- soient les déclarations suivantes :

```
type
        voiture = record
                marque : string ;
                cylindree : real ;
                couleur : string;
                nom : string ;
                prix : integer ;
        end ;
var auto : voiture;
```

- Méthode
- Pour afficher la marque et le prix de la variable auto, on pourra écrire :

```
writeln(auto.marque, auto.prix...);
```

Enregistrements (Objets composés)

Ecriture dans un enregistrement

- Syntaxe : Affectation globale (comme pour les tableaux)
- Si les deux enregistrements sont exactement de même type, on peut faire une affectation globale, comme pour un tableau :

```
enreg1:= enreg2;
```

- Syntaxe : Affectation directe sur un champ

```
enreg.champ:= valeur;
```

- Syntaxe : Par une instruction de lecture

```
read(enreg.champ);
```

Enregistrements (Objets composés)

Ecriture dans un enregistrement

- Exemple : Création de la "307 HDI«
- Il suffit d'écrire les instructions suivantes :

```
auto.marque := 'Peugeot';
auto.cylindree := 2.0;
auto.couleur := 'gris';
auto.nom := '307 HDI';
auto.prix := 18000;
```

Enregistrements (Objets composés)

Instruction with

L'écriture devient lourde et fastidieuse a cause de la répétition de l'identificateur de l'enregistrement. L'instruction **with** permet d'alléger l'écriture en ≪ factorisant ≫ l'identificateur

- Syntaxe

```
with <enregistrement> do
      begin
            <Bloc d'instructions>
      end;
```

Enregistrements (Objets composés)

Instruction with

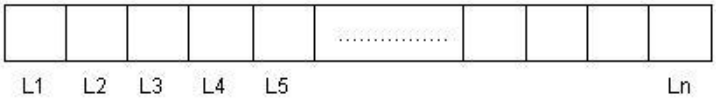
- **Exemple**

```
with auto do
      begin
      marque := 'Peugeot' ;
      cylindree := 2.0 ;
      couleur := 'gris' ;
      nom := '307 HDI' ;
      prix := 18000 ;
end ;
```

- Remarque: Cela évite d'avoir a écrire plusieurs fois le préfixe auto : (auto.marque, auto.couleur, auto.nom, auto.prix...).

Les tableaux

- Définition
- Un **tableau (Vecteur)** est une collection ordonnée d'éléments ayant tous le même type. On accède a chacun de ces éléments individuellement a l'aide d'un indice.
- Un tableau a une dimension est parfois appelé vecteur .
- Il peut être représente sous la forme suivante :

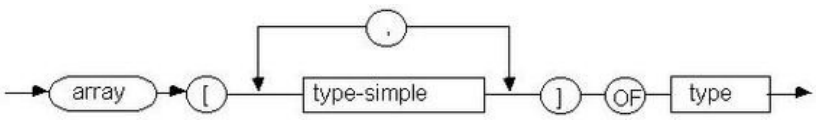


- Dimension du tableau : 1
- Taille du tableau : n
- Les Li (i = 1 ... n) **doivent être de même type**

Les tableaux

- Déclaration d'un type tableau

- On décrit un tableau comme suit :
  - VAR nomdutableau :TABLEAU [1..N] DE type
  - TYPE nomdutableau :TABLEAU [1..N] DE type
- Syntaxe Pascal
  - VAR identificateur = array[type-index] of type-elements;
  - TYPE identificateur = array[type-index] of type-elements;



Les tableaux

Remarque

- L'indice doit être **de type ordinal**, c'est a dire qu'il doit prendre ses valeurs dans un ensemble fini et ordonne (l'indice ne peut donc pas être de type réel).
- Les éléments doivent tous être de même type. Tout type est autorise.
- Quand on ne connait pas exactement le nombre d'éléments a l'avance, il faut majorer ce nombre, quitte a n'utiliser qu'une partie du tableau.
- Il est impossible de mettre une variable dans l'intervalle de définition du tableau. Il est conseille en revanche d'utiliser des constantes définies au préalable dans la section **const**.

Les tableaux

- **Exemples**

- Liste = array [1..100] of real;
- Chaine = array [1..80] of char ;
- Symptome = (FIEVRE, DELIRE, NAUSEE) ;
- Malade = array [symptome] of boolean ;
- Code = 0..99 ; Codage = array [1..N_VILLE] of code ;

Les tableaux

Déclaration d'une variable de type tableau

- Pour déclarer une variable de type tableau, il est préférable que le type correspondant ait été déclare auparavant.

- Exemple

```
const NMAX=100 ;
type Vecteur=array[1..NMAX] of real ;
var v : Vecteur;
```

- Conseil
- La déclaration suivante est autorisée, mais déconseillée :

```
var v : array[1..100] of integer
```

Les tableaux

Ecriture dans un tableau

On peut écrire de plusieurs façons dans un tableau :

- par affectation directe case par case : T[I] := ... ;
- par lecture : read (T[I]);

Les tableaux

Exemple : Le programme suivant permet de "remplir" un tableau a l'aide d'une boucle repeat.

```
program Mon_tableau;
Const Taille_max=10;
Type TAB=array[1..Taille_max] of integer;
Var Tableau:TAB;
indice: integer;
begin
for indice:=1 to Taille_max do
        Tableau[indice]:=0;
indice:=1;
repeat
        write('entrez l''element N° ',indice,':');
        readln(Tableau[indice]);
        indice:=indice+1;
until indice > Taille_max;
end.
```

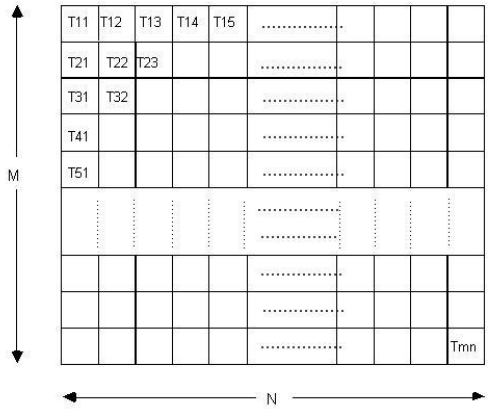
Les tableaux

Exemple: Le programme suivant calcule le produit scalaire de deux vecteurs entres par l'utilisateur.

```
program PRODUIT-SCALAIRE ;
Type Coordonnee = (X1, X2, X3) ;
Vecteur = array [Coordonnee] of real ;
Var u, v : vecteur ;  resultat : real ; c : Coordonnee ;
begin
resultat := 0 ;
for C := X1 to X3 do
begin
        read (u[c]) ;
        readln (v[c]) ;
        resultat := resultat + u[c] * v[c] ;
end ;
writeln ('le produit scalaire est : ', resultat) ;
end.
```

Les tableaux à plusieurs dimensions

- 1. Tableau à 2 dimensions
- Un tableau de dimension 2 est parfois également appelé "MATRICE". Un tel tableau peut être représenté sous la forme suivante :



Les tableaux à plusieurs dimensions

- Syntaxe : Déclaration du type

```
type identificateur = array[1..M, 1..N] of
type-elements;
```

- Méthode : Accès a un élément d'un tableau T

```
T[i][j] ou T[i,j]
```

- **Exemples**

```
Type Tableau = array [1..10,1..20] of integer ;
Point = array [1..50,1..50,1..50] of real ;
Matrice = array [1..M,1..N] of real ;
Carte = array [1..M] of array [1..N] of real ;
Var
lieu : Carte;
Méthode : accès a un élément
lieu[i][j] ou lieu[i,j]
```

Les tableaux à plusieurs dimensions

Exemple : Initialisation d'une matrice unité de dimension 10. Il s'agit donc d'une matrice a 10 colonnes et 10 lignes, ne comportant que des 0, sauf sur sa diagonale ou il n'y a que des 1.

```
program SOMME_MATRICE ;
Const l_max = 10 ; c_max = 10 ;
Type Matrice = array [1..l_max,1..c_max] of integer ;
Var i, j : integer; mat : Matrice;
begin
for i := 1 to l_max do
begin
          for j := 1 to c_max do
          begin
                    if i = j then
                              mat [i, j] := 1
                    else
                              mat [i, j] := 0 ;
                    write (mat [i, j]);
          end ;
          writeln ;
end ;
end.
```