

Les enregistrements

Gilles Trombettoni

IUT MPL-Sète, département info

Octobre 2019

- 1 Motivation et définition
- 2 Exemples
- 3 Des enregistrements aux objets

Motivation

- Les **enregistrements** sont des structures de données très utilisées dans les années 1970 et 1980. On les retrouve dans de nombreux langages. On parle d'enregistrements en langage Pascal ou Ada. On parle de **structures** en langage C.
On les appelle **record** ou **struct** en américain.
(On ne va pas les étudier en TD.)
- Ils vont permettre une transition vers la notion d'**objet**.

Remerciements à Alain Joubert pour son cours dont celui-ci est largement inspiré.

Définition

- Un enregistrement est une collection de plusieurs éléments qui ne sont pas obligatoirement tous de même type.
- Autrement dit, un enregistrement est une structure de données composée, qui permet de stocker sous un même identificateur (une même variable) des valeurs qui peuvent être de types différents.
- Chacun des éléments composant l'enregistrement est repéré par un **identificateur**, et non par un **indice** comme dans les tableaux.
- Un élément est généralement appelé **champ** (de l'enregistrement).

Définition d'un type enregistrement (en pseudo-code)

Un enregistrement étant composé et complexe, on définit souvent un **type enregistrement** pour déclarer plus facilement ensuite les variables enregistrement.

```
type Tenreg = enregistrement de
    champ1 : Type1;
    champ2 : Type2;
    ...
    champN : TypeN
fin enregistrement
```

où Type1, Type2, ..., TypeN peuvent être des types simples (entier, caractère, etc.), des types tableaux, des types enregistrements, etc.

On déclare ensuite une variable enregistrement comme d'habitude :
var : Tenreg

On retrouvera l'équivalent d'une déclaration de type avec une définition de **classe** en programmation à objets.

Exemple 1 : des points sur l'écran

Algo demoPoints

Types

TPointEcran = enregistrement de

x : entier; /* abscisse */

y : entier; /* ordonnée */

estVisible : booléen; /* non caché */

couleur : entier

/* 0:bleu, 1:rouge, 2:jaune, etc. */

fin enregistrement

Variables

p1, p2, p3 : TPointEcran

...

Exemple 1 : des points sur l'écran

Variables

```
p1, p2, p3 : TPointEcran
```

Début

```
p1.x <- 10
```

```
p1.y <- 20
```

```
p1.couleur <- 1 /* rouge */
```

```
p1.estVisible <- vrai
```

```
...
```

```
si p1.estVisible alors
```

```
    dessiner(p1)
```

```
finSi
```

Fin demoPoints

Exemple 2 : Enregistrement TPersonne

...

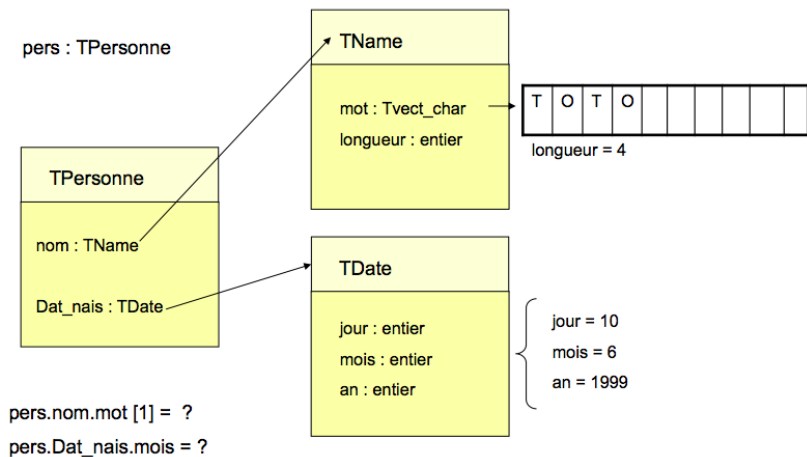
Types

```
TPersonne = enregistrement de
              nom          : TName
              date_nais    : TDate
            fin enregistrement
```

```
TDate  = enregistrement de
           jour   : entier
           mois   : entier
           an     : entier
         fin enregistrement
```

```
TName = enregistrement de
           mot      : tableau de 30 caractères
           longueur : entier
         fin enregistrement
```


Exemple 2 : variable enregistrement pers



Exemple 2 : Saisie d'une personne

```
fonction saisirPersonne () retourne TPersonne
  // Action-résultat :
  //      Création d'une variable enregistrement TPersonne
  //      Saisie de son nom et de sa date de naissance
  //      Retourne l'enregistrement initialisé
Variables
  pers : TPersonne
Début
  pers.nom <- saisirNom()
  pers.date_nais <- saisirDate()
  retourne pers
Fin saisirPersonne
```

Exemple 2 : Saisie d'une personne

```
fonction saisirDate() retourne TDate
```

```
// Résultat : une date saisie par l'utilisateur
```

```
Variables
```

```
    d : TDate
```

```
Début
```

```
    afficher ("Entrer une date sous la forme de 3 entiers")
```

```
    saisir (d.jour)
```

```
    saisir (d.mois)
```

```
    saisir (d.an)
```

```
    retourne d
```

```
Fin saisirDate
```

Exemple 2 : Deux personnes de même nom ?

```
fonction memeNom (p1: TPersonne, p2: TPersonne) retourne booléen
// Résultat : vrai ssi les deux personnes p1 et p2 ont le même nom
Variables
  i : entier
  egaux : booléen
Début
  i <- 0
  egaux <- p1.nom.longueur == p2.nom.longueur
  tantQue egaux et i < p1.nom.longueur faire
    si p1.nom.mot[i] != p2.nom.mot[i] alors
      egaux <- faux
    sinon
      i <- i + 1
  finSi
  finTantQue
  retourne egaux
Fin memeNom
```

Les classes et les objets

- Les classes généralisent les types enregistrements.
- Les objets/instances généralisent les variables enregistrements.
- Les variables enregistrements (s)ont (seulement) des données.
- Les objets (s)ont des données **et** des comportements (fonctions) !

Exemple : Des points qui bougent

Classe Point

Attributs /* Données */

x : entier

y : entier

estVisible : booléen

couleur : entier

Méthodes /* Fonctions, comportement */

fonction Point(abs: entier, ord: entier,
 coul: entier)... //Fonction d'initialisation

fonction Point()... // Fonction d'initialisation aléatoire

fonction déplacer(dx: entier, dy: entier)...

fonction dessiner()...

fonction effacer()...

Fin Point

Cette déclaration indique que toutes les variables/objets Point qui seront créés et initialisés (à partir des fonctions d'initialisation) auront la structure donnée par la classe.

Exemple : Des points qui bougent

Classe Point

Attributs

```
x : entier
y : entier
estVisible : booléen
couleur    : entier
```

Méthodes

```
fonction Point(abs: entier, ord: entier, coul: entier)
```

Début

```
  x <- abs ; y <- ord ;
  estVisible <- vrai ; couleur <- coul
  dessiner()
```

Fin Point

```
fonction deplacer(dx: entier, dy: entier)
```

Début

```
  effacer() // efface le point si estVisible et modifie estVisible
  x <- x + dx ; y <- y + dy
  dessiner() // dessine le point si estVisible
```

Fin deplacer

Exemple : Des points qui bougent

Algo dessinerPoints

Variables

p1 : Point

Début

p1 <- nouveau Point(3,8,1) // Création d'un point rouge
// visible en (3,8)

p1.effacer()

p1.deplacer(10,-5) // Le point rouge caché s'est déplacé

p1.dessiner()

Fin dessinerPoints