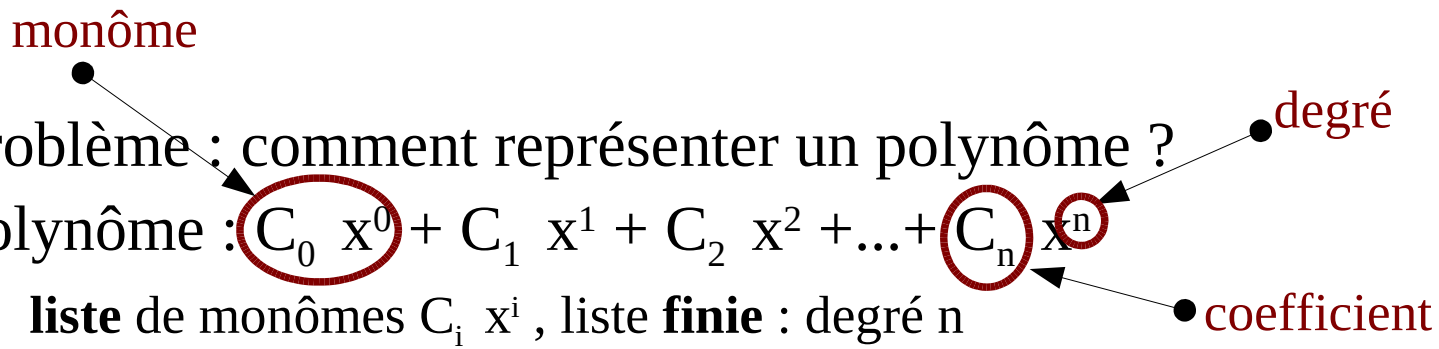
5 Types abstraits

© J. Morinet Lambert , M. Cadot UHP, L. Pierron

version : 17/11/04

Application : le type polynôme



- Problème : comment représenter un polynôme ?
- Polynôme : $C_0\ x^0 + C_1\ x^1 + C_2\ x^2 + \ldots + C_n\ x^n$
  - **liste** de monômes $C_i\ x^i$, liste **finie** : degré n
  - informations d'un monôme : coefficient, degré
- Choix de représentation
  - liste finie **n connu et information semblable** => **tableau** de n *monômes*;

    $\{C_0, C_1, C_2, \ldots, C_n\}$
  - comment représenter un monôme :
    - association de deux informations coefficient (réel), degré (entier)
    - remarque : indice du tableau est identique au degré
    - utilise l'indice d'accès du tableau comme degré
      P[i] = C_i contient le coefficient du monôme de degré i

Déclaration de polynôme

- Déclarations

```
#define taille 21                  /* nombre total de monômes */
#define degmin 0                   /*degré du  terme constant */
#define degmax 20                  /* degré maximum compatible */
 typedef  float  poly [taille] ;  /* tableau des coefficients*/
```

- Utilisation

```
poly P1,P2;              /* déclaration de variables */
P1[3] = 5.3;             /* coefficient de degré 3 */
```

- Cas particulier : liste incomplète **(ici n < degmax)**
  - solution 1 : gère le nombre d'éléments effectifs de la liste (cf. structure)
  - solution 2 : on **IMPOSE** la gestion de tous les coefficients présents dans le tableau (notre choix)

Définir un type polynôme

- comment le représenter ? cf. diapo précédente
- quelles fonctions définir pour
  - créer : les **générateurs**
  - observer : quel est son degré ? Quelle sa valeur pour x=7 ?
  - modifier : comment faire la somme de deux polynômes ?
- comment s'affranchir de la représentation interne (tableau)?
  - créer un **type abstrait de données** : réfléchir en terme de polynôme et pas en terme de tableau
  - de quels outils élémentaires doit-on disposer ?

Procédures de création

- Place réservée lors de la déclaration
- Initialiser :
  - à quelle valeur ? 0 dans une procédure **`initpoly(poly *P)`**
  - penser à initialiser avant toute utilisation
- Ranger les coefficients :
  - procédure **`fixcoef(poly *P, int degre, float coeff)`**

Ces procédures (**`initpoly, fixcoef`**) sont des "**générateurs**"

Procédures d'observation

- comment accéder (lire) un coefficient ?
  - procédure **`lirecoef(poly P, int degre)`**
- comment connaître le degré du polynôme ?
  - algorithme : degré du 1er terme de coefficient non nul quand on parcourt les monômes en partant du dernier
  - procédure **`degre(poly P)`**

Ces procédures (**`lirecoef,degre`**) sont des "**observateurs**"

Autres procédures

- afficher un polynôme ? procédure **`affichepoly(poly P)`**
  - affichage brut : on affiche tous les coefficients
  - affichage soigné : on affiche selon les règles mathématiques
    - ne pas afficher si le coefficient est nul
    - afficher + devant termes de coefficient positif sauf premier
    - mais attention on ne peut écrire des indices ou des exposants en mode texte standard
- faire la somme de deux polynômes en effectuant la somme des coefficients :

  `addpoly(poly *P1, poly P2)` => P1 = P1+P2

Avantages types abstraits

- Le **type abstrait** est défini par :
  - un nom de type : **`poly`**
  - la liste des procédures générateurs, modifieurs et observeurs : **`initpoly, fixcoef, lirecoef, degre, affichepoly`**
- Seul l'**auteur** du type abstrait doit connaître la représentation interne pour définir les procédures précédentes. Dans l'exemple : *tableau*.
- L'**utilisateur** du type abstrait **s'abstrait** de la représentation interne :
  - Déclare les variables utiles avec le type **`poly`**
  - Variables manipulées **uniquement** avec les procédures associées au type
  - Fonctionnalité manquante => demande à l'auteur ajout nouvelle procédure