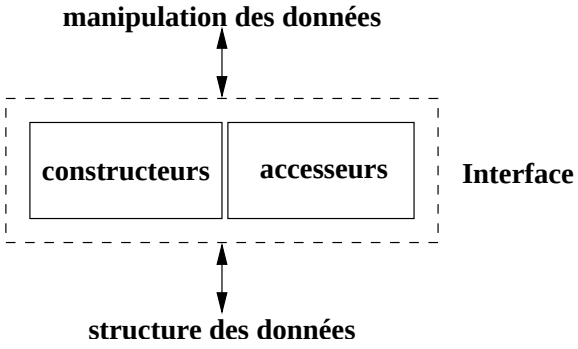
Programmation applicative 2 – L2
TD 7 : Abstraire avec des données

G.Artignan {artignan@lirmm.fr} M.Lafourcade {lafourcade@lirmm.fr}
S.Daudé {sylvain.daude@univ-montp2.fr} A.Chateau {chateau@lirmm.fr}
B.Paiva Lima da Silva {bplsilva@gmail.com} C.Dony {dony@lirmm.fr}

1 Utiliser une structure abstraite de données



Cela consiste à isoler [1] la partie qui définit comment des données sont structurées de la partie qui indique comment les utiliser. Il s'agit alors de fournir une **interface** (à l'aide de **constructeurs** et d'**accesseurs**). Pour manipuler des structures de données, il suffit de connaître l'interface associée, sans nécessairement connaître la structure.

Exemple : La structure de données `pair` :

constructeur	accesseurs
`cons` : $exp \times exp \rightarrow pair$ $e_1, e_2 \mapsto$ `(`$\overline{\texttt{e}}_1$ `.` $\overline{\texttt{e}}_2$`)`	`car` : $pair \rightarrow exp$ `(`$\texttt{e}_1$ `.` $\texttt{e}_2$`)` $\mapsto \texttt{e}_1$ `cdr` : $pair \rightarrow exp$ `(`$\texttt{e}_1$ `.` $\texttt{e}_2$`)` $\mapsto \texttt{e}_2$

2 Implantation des nombres complexes

Le nombre complexe $a + i.b$ sera ici implanté par une liste `( a b )`.

Exercice 1 *Écrire les fonctions d'accès, de construction et de test liées aux nombres complexes :*
- *Construire le complexe $a + ib$ par* `(faire_complexe a b )`
- *Accéder aux parties réelles et imaginaires par les fonctions* `partie_reel` *et* `partie_imag`.
- *Tester si un nombre complexe c est un réel pur par l'appel* `(reel? C)`*, s'il est imaginaire pur par* `(imag? C )`.

Exercice 2 *Écrire les quatre fonctions* `somme_complexe`, `produit_complexe`, `inverse_complexe` *et* `divise_complexe`.

Exercice 3 *Écrire la fonction qui élève un nombre complexe à une puissance entière (éventuellement négative).*

1. On parle de barrière d'abstraction.

3 Implantation des polynômes

Un polynôme à une variable est un objet mathématique que l'on définit normalement comme une somme de termes $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_kx^k$ chaque terme étant de la forme a_kx^k où x est la variable du polynôme, k le degré du terme et a_k son coefficient. On appelle ordre d'un polynôme le plus grand degré de ses termes. Une manière classique de représenter ces types de polynômes en Scheme est de les représenter sous la forme d'une liste formée de la suite $a_0, .., a_n$ de tous les coefficients. Un polynôme de degré n est donc représenté ainsi : `(a0 a1 ... an)`

Si un terme est nul, le coefficient vaut 0. Par la suite, toutes les opérations sur des polynômes sont supposées s'appliquer sur des polynômes de même variable.

Exercice 4 Fonctions élémentaires. *Définir*

1. *la fonction* `monome` *telle que* `(monome deg coef)` *retourne le monôme de degré* `deg` *et de coefficient* `coef`.
2. *la fonction* `addpoly` *telle que* `(addpoly p1 p2)` *retourne la somme des deux polynômes* `p1` *et* `p2`. *Exemple :*

```
> (addpoly '( 4  5  0  3  8 )   '( 1  2 ))
> ( 5  3  0  3  8 )
```

3. *la fonction* `multconst` *telle que* `(multconst p c)` *retourne le produit du polynôme* `p` *par la constante* `c`. *Exemple :*

```
> (multconst '( 4  5  0  3  8 )    5  )
> ( 20  25  0  15  40 )
```

4. *la fonction* `multvar` *qui multiplie le polynôme* `p` *en argument par le monôme* `x`. *Ainsi le polynôme* $x^2 + 3x + 4$ *donne* $x^3 + 3x^2 + 4x$.
5. *la fonction* `evalpoly` *telle que* `(evalpoly p v)` *calcule la valeur du polynôme* `p` *en* `v`. *Exemples :*

```
> (evalpoly  '( 4  5 )  2)
> 14
> (evalpoly  '( 3  2  1 )  1)
> 6
```

Exercice 5 Affichage d'un polynôme. *Définir*

1. *la fonction* `monome->string` *telle que* `(monome->string ak k)` *rende en valeur la chaîne de caractères correspondant au monôme* a_kx^k *. Exemples :*

```
> (monome->string  3  7 )
> "3x^7"
> (monome->string  3  1 )
> "3x"
```

2. *la fonction* `polynome->string` *telle que* `(polynome->string p )` *rende en valeur la chaîne de caractères correspondant au polynôme* `p`.

Exercice 6 Multiplication de polynômes. *Définir la fonction* `multpoly` *qui fait le produit de ses deux polynômes passés en argument.*

Notation creuse Dans cet exercice, il est utilisé une nouvelle implantation des polynômes, dite en « notation creuse » : le polynôme $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_kx^k$ est représenté par la liste des monômes qui composent le polynôme, où un monôme a_kx^k est représenté par un couple `( k . ak )`. L'ordre des monômes est quelconque dans la liste, mais il ne doit pas y avoir deux monômes de même degré. Par exemple, les deux listes `(( 0 . a0 ) ( 1 . a1 )... ( n . an ))` et `( ( n . an ) ... ( 1 . a1 ) ( 0 . a0 ) )` représentent le même polynôme $p(x)$. L'intérêt de cette représentation tient à l'inutilité de mettre les monômes de coefficient nul. Par exemple, le polynôme $x^{27} + 1$ sera représenté par exemple par la liste : `( (0 . 1) (27 . 1))`.

Exercice 7 *Définir*

1. *la fonction* `addmonome` *qui prend en argument un monôme et un polynôme et qui en fait la somme.*
2. *la multiplication de deux polynômes pour cette notation.*
3. *les deux fonctions de conversion* `pleine->creuse` *et* `creuse->pleine` *qui transforment un polynôme en argument d'une notation vers l'autre.*