Maîtrise informatique 1999-2000 — Bordeaux I

TFP

TD 6 - COMMON LISP

Différences de base de la syntaxe Common Lisp et Scheme - Traitement de listes - Enregistrements - Fonctions et fonctionnelles

1. Il s'agit de définir un ensemble de fonctions afin de gérer des ensembles d'éléments représentés sous forme de listes (au contraire des listes, les ensembles ne peuvent contenir qu'une seule occurence d'un élément).

   Considérer tout d'abord le cas simplifié d'ensemble qui ne contient que des atomes:

   (a) Ecrire la fonction `set-emptyp` qui teste si l'ensemble est vide.
   (b) Ecrire la fonction `set-member` qui teste si un élément appartient à un ensemble.
   (c) Ecrire les fonctions d'union `set-union`, d'intersection `set-intersection`, de différence `set-difference`.
   (d) Ecrire la fonction `set-from-list` qui transforme une liste en ensemble.

2. Dans la mesure où la fonction de base `set-member` fait usage d'un test d'égalité, il serait donc intéressant d'utiliser un argument mot-clef:

   (a) Réécrire `set-member` dans ce sens.
   (b) Modifier les autres fonctions afin qu'elles permettent cette paramétrisation.

3. Application des ensembles à des structures:

   (a) Définir une structure `point` pour représenter les points du plan.
   (b) Définir la relation d'équivalence `on-the-same-line` qui consiste à identifier deux points qui représentent des vecteurs co-linéaires.
   (c) Utiliser cette relation pour des ensembles de classes de cette relation. Entre autres à partir d'une liste de points, obtenir la fonction qui ne donne qu'un représentant par classe.

4. Lisp permet facilement de construire de nouvelles fonctions puisque leurs sources ne sont que de simples listes:

   (a) Ecrire la fonction `make-power-code` qui produit le source de la fonction qui calcule la puissance n d'un entier.
   (b) Définir la fonction `square` et `cube` en l'utilisant
   (par exemple par un appel `(make-power-code 'cube 3)`)).

5. Certaines fonctions sur les listes en Common Lisp ont des effets de bord.

   (a) Le vérifier pour la fonction `mapcan`.
   (b) Ecrire une version de `mapcan` sans effets de bord.
   (c) L'utiliser pour écrire une fonction qui construit le produit cartésien de deux listes et qui permet par un argument mot-clef d'appliquer une fonction sur chacun des éléments de ce produit.
   (d) Appliquer ce mécanisme au calcul de toutes les distances entre des villes définies par leurs coordonnées (*on pourra supposer que la Terre est plate*).