

Série d'exercices #7

IFT-2035

October 27, 2009

1 Raisonner

Soit le morceau de code suivant, où f est une fonction quelconque que l'on ne connaît pas et où le langage utilise une syntaxe de style C:

```
{
  int table[2] = {0, 1};
  int size = 2;
  int tmp = 0;

  f (table, size);
  ...
}
```

On aimerait savoir si certaines conditions sont nécessairement toujours vraies après l'appel à f . On s'intéresse plus particulièrement aux conditions suivantes:

- `table[0] == 0`
- `size == 2`
- `tmp == 0`

Indiquer lesquelles de ces trois conditions sont nécessairement vraies dans chacun des cas suivants:

1. Le langage est exactement comme C: portée statique, passage d'arguments par valeur, affectation autorisée.
2. Le langage est comme C sauf que l'affectation (autre que l'initialization) est interdite.
3. Le langage est comme C sauf que les arguments sont passés par référence.
4. Le langage est comme C mais avec portée dynamique.

2 Compteurs de référence

Soit une librairie de gestion de listes simplement chaînées en C:

```
typedef struct list list;
struct list {
    int refcount;
    void *value;
    list *next;
}
list *list_new    (void *car, list *cdr);
void *list_car    (list *l);
list *list_cdr    (list *l);
/* Maintenance des compteurs de référence. */
void list_incr    (list *l);
void list_decr    (list *l);
```

Écrire le code des fonctions proposées.

Compléter en ajoutant une opération `list_map`.

Justifiez pourquoi les incréments et décréments que vous avez judicieusement placés sont suffisants pour garantir que le comportement sera toujours correct. En extraire une convention spécifiant où doivent être ajoutés les incr/décr.

Que se passe-t-il si vous voulez manipuler des listes de listes?