Enseignants : L. Bouge, A. Bonhomme
Email : {Lub.Bouge,Alice.Bonhomme}@ens-lyon.fr

le 16/10/00

TD Programmation 3 : Les références

1 Listes chaînées

1.1 définition

Voici la définition du type `ilist` que vous utiliserez :

```
type ilist = Empty | Cons of node
and node = {item: int; next: ilist ref};;
```

1.2 Fonctions élémentaires : constructeurs, extracteurs

Munissez le type `ilist` de quelques opérations élémentaires de manière à pouvoir le manipuler comme un type abstrait :

`empty, cons, car, cdr, is_empty, rplacd`

Important ! *À partir de maintenant, les listes doivent être* exclusivement *manipulés à travers ces fonctions.*

1.3 Fonctions simples

1.3.1 Ecriture des fonctions

En utilisant les opérations élémentaires précédentes, définissez les fonctions suivantes :

`size: ilist -> int`
Retourne le nombre d'éléments d'une liste.

`generate: int -> ilist`
Génère une liste dont la taille est passée par paramètre. Les éléments de la liste auront une valeur tirée aléatoirement entre 0 et 99.

`print: ilist -> unit`
Affiche une liste.

`append_soft: ilist -> ilist -> ilist`
Renvoie une liste résultat de la concaténation logique de 2 listes.

`append_hard: ilist -> ilist -> ilist`
Renvoie une liste résultat de la concaténation physique de 2 listes.

1.3.2 Tests des fonctions

Définir deux listes l et m et évaluer :

```
#print l;;
#print m;;
#let  p1 = append_soft l m;;
#print p1;;
#size p1;;
#size l + size m;;
```

Jusqu'ici, rien de bien étonnant ou alors revoyez vos fonctions.

```
#let p2 = append_hard l m;;
#print p2;;
#p1 = p2;;
#p1 == p2;;
#l == p1;;
#l == p2;;
#print l;;
```

Que se passe-t-il ? Pourquoi ?
Essayer de prévoir le résultat des évaluations suivantes :

```
#let p = append_hard l l;;
#size p;;
```

2 Listes chaînées circulaires

On utilise la même définition de type que précédemment. Toutefois, la liste vide de sera jamais utilisée. La figure 1 montre deux exemples de listes chaînées circulaires.

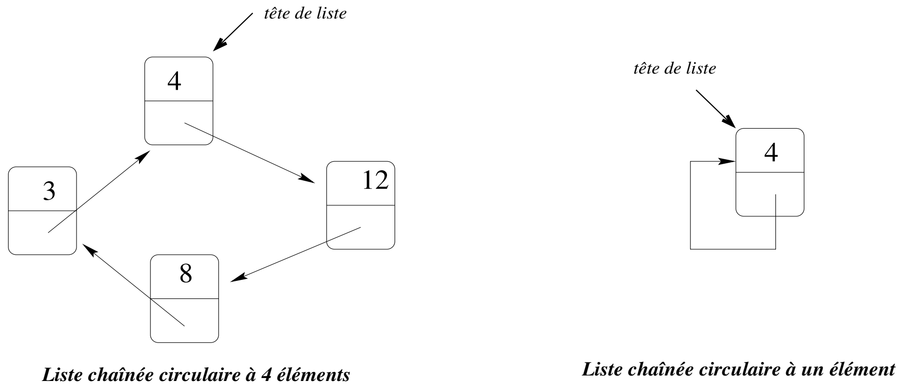

FIG. 1: Liste chaînée circulaire

Bien que la liste soit circulaire, un de ses élément sera identifié comme étant sa tête.

2.1 fonctions élémentaires

Reprendre les fonctions précédentes, `cons, car, cdr, rplacd` (sauf `empty` et `is_empty`) pour ces listes (les cas de liste vide renvoie des erreurs).

2.2 fonctions simples

Ecrire les fonctions suivantes :
`init: int -> ilist`
Crée une liste circulaire avec un seul élément.

`insert: int -> ilist -> unit`
Insère un nouvel élément après la tête de liste.
`genere: int -> ilist`
Génère une liste dont la taille est passée par paramètre. Les éléments de la liste auront une valeur tirée aléatoirement entre 0 et 99.
`forward:  ilist -> ilist`
Déplace la tête de liste vers la droite et retourne cette nouvelle valeur.
`backward:  ilist -> ilist`
Déplace la tête de liste vers la gauche et retourne cette nouvelle valeur.
`remove: ilist -> unit`
Supprime le 1er élément après la tête de liste.
`size: ilist -> int`
Retourne la taille d'une liste.
Tester vos fonctions sur les évaluations suivantes :

```
#let m = generate 5;;
#let s1 = size m;;
#insert 2 m;;
#let s2 = size m;;
#s2 = s1 + 1;;
#remove m;;
#let s2 = size m;;
#s2 = s1;;
```

Mettons votre fonction `size` à rude épreuve :

```
#let l = init 4;;
#insert 4 l;;
#insert 4 l;;
#insert 4 l;;
#size l;;
```

Ca marche toujours ? si oui, bravo, sinon modifier votre fonction en conséquence.
`print: ilist -> unit`
Affiche une liste.
`merge: ilist -> ilist -> unit`
Fusionne physiquement les deux listes. Pour ceci, on modifie ces 2 listes de manière à les lier l'une à l'autre pour ne former qu'une seule liste.
Comme précédemment, définir deux listes l et m et évaluer :

```
#merge l m;;
#size l;;
#size m;;
```

A quoi est égal `l` après `merge l l` ?

3 Nouvelle définition pour les listes chaînées circulaires

```
type cilist =  {item: int; next: cilist ref};;
```

A partir de ce type réécrire la fonction `init`.
Puis, reprendre la section 2 avec ce nouveau type.

4 Listes circulaires doublement chaînées

La figure 2 montre un exemple de liste circulaire doublement chaînée.

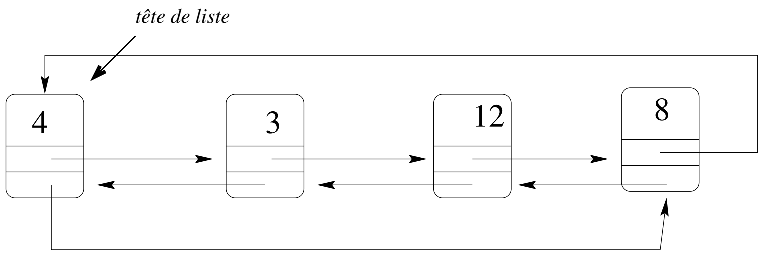

FIG. 2: Liste circulaire doublement chaînée

Définir un type de liste doublement chaînées.
Reprendre la section 2 pour ce type de listes.