Université Montpellier-II - UFR - DEUG MIAS

17 Séquences et effets de bords en programmation récursive

Effet de bord : modification explicite de l'état de l'ordinateur (de la mémoire ou de l'affichage).
Exemple : instruction d'affectation.

```
(define x 1)
```

Il y a un sens à écrire des séquences d'instruction réalisant des effets de bord.

```
(display
```

Fonctions récursives avec effets de bord.

```
(define (display-elements l)
    (if (not (null? l))
        (begin
            (display (car l)) ;; effet de bord
            (display `` '')   ;; effet de bord
            (display-elements (cdr l)))))
```

Affichage en profondeur d'abord des données contenues dans un arbre binaire.

```
(define (affichage a)
  (or (arbre-vide? a)
      (begin (affichage (fg a))
             (display (val-noeud a))
             (affichage (fd a)))))
```

Dessins Récursifs

```
(define (feuille long angle)
  (if (> long 1)
      (begin
        (draw (/ long 2))
        (turn (- angle))
        (feuille (/ long 2) angle)
        (turn (* angle 2))
        (feuille (/ long 2) angle)
        (turn (- angle))
        (draw (/ long 2))
        (turn (- angle))
        (feuille (/ long 2) angle)
        (turn angle)
        ;(feuille (/ long 2) angle)
        (turn angle)
        (feuille (/ long 2) angle)
        (turn (- angle))
        (move (- long)))))

(define (naperon long angle)
  (let ((n (/ 360 angle)))
    (define (boucle long angle times)
      (if (> times 0)
          (begin
            (feuille long angle)
            (turn angle)
            (boucle long angle (- times 1)))))
```

```
     (boucle long angle n)))

(turtles)
(turn 90)
(naperon 200 60)
```

17.1 Modification physique de la mémoire pour optimisation

Les procédures set-car ! et set-cdr !
La fonction append destructrice.
Les listes circulaires.

```
(define (d-append l1 l2)
  (cond ((null? l1) l2)
        ((null? (cdr l1)) (set-cdr! l1 l2) l1)
        (#t (d-append (cdr l1) l2))))

(define heures (list 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24))

(d-append heures heures)
```

17.1.1 Un arbre binaire impératif

A jout de nouvelles fonctions d'interface pour pouvoir modifier le fils gauche et le fils droit.
(define (set-fg ! a a2) (set-car ! (cdr a) a2))
(define (set-fd ! a a2) (set-car ! (cddr a) a2))

```
(define (p-insere n a)
  ;; version n'acceptant pas les insertions successives
  (display a) (display n) (display "\n")
  (if (arbre-vide? a)
      (make-feuille n))
      (let ((valeur (val-noeud a)))
        (cond ((< n valeur)
               (if (arbre-vide? (fg a))
                   (set-fg! a (make-feuille n))
                   (p-insere n (fg a))))
              ((> n valeur)
               (if (arbre-vide? (fd a))
                   (set-fd! a (make-feuille n))
                   (p-insere n (fd a))))
              ((= n valeur) a))))


;;;-----------------------------------------
(define l2 (make-feuille 5))
(p-insere 4 l2)
(p-insere 7 l2)
(p-insere 8 l2)
(p-insere 1 l2)
(p-insere 2 l2)
```

Exercice : Une version de la fonction p-insere qui autorise l'écriture de (set ! l (insere 2 (insere 1 (insere 8 (insere 7 (insere 4 l))))))