Prog Fonctionnelle, TD2

Prédicats

Fonctions qui retournent #t ou #f

Quelques exemples :

(= e1 e2) pour les nombres … (equal? e1 e2) autrement
(> e1 e2) (< e1 e2) (<= e1 e2) …

Attention : (= ‘’hello’’ ‘’hello’’) ne marche pas !

(even? n)
(odd? n)
(zero? n)
(negative? n)

Prédicats

Quelques fonctions déjà vues

Type	Pour vérifier le type de x
• **Nombre**	**(number? x)**
Entier	**(integer? x)**
Rationnel	**(rational? x)**
Réel	**(real? x)**
Complexe	**(complex? x)**
• **Caractère**	**(char? x)**
• **String**	**(string? x)**
• **Booléen**	**(boolean? x)**

Ces objets sont « auto-évalués » par l'interprète

Opérateurs de Boole

Rappel

(and e1 e2) (or e1 e2)
(not e1)

Exemple : (or (> (* 10 5) (/ 12 3)) (tralala pi))

Déjà #t n'est pas évaluée

Exercice : définir la fonction (xor ...)

Opérateurs de Boole

Rappel

(and e1 e2) (or e1 e2)
(not e1)

Exemple : (or (> (* 10 5) (/ 12 3)) (tralala pi))

Déjà #t n'est pas évaluée

Exercice : définir la fonction (xor ...)

```
(define (xor a b) (or (and a (not b))
                      (and b (not a)) ) )
```

Fonctions conditionnelles

if

(if test e1 e2)

Si test est #t alors e1 sinon e2 est retournée

Exemples :

```
( if (even? v) ‘’pair’’ ‘’impair’’)
( if (and (number? x) (number? y))
      (= x y)    (equal? x y))
```

Exercice : définir la fonction (max2 x y) et (max3 x y z)

Fonctions conditionnelles

if

(if test e1 e2)

Si test est #t alors e1 sinon e2 est retournée

Exemples :

```
( if (even? v) ‘’pair’’ ‘’impair’’)
( if (and (number? x) (number? y))
      (= x y)    (equal? x y))
```

Exercice : définir la fonction (max2 x y) et (max3 x y z)

```
(define (max2 x y) (if (> x y) x y)))
```

Fonctions conditionnelles

if

(if test e1 e2)

Si test est #t alors e1 sinon e2 est retournée

Exemples :

```
( if (even? v) ‘’pair’’ ‘’impair’’)
( if (and (number? x) (number? y))
      (= x y)    (equal? x y))
```

Exercice : définir la fonction (max2 x y) et (max3 x y z)

```
(define (max2 x y) (if (> x y) x y)))
(define (max3 x y z) (max2 (max2 x y) z)))
```

Fonctions conditionnelles

cond

Plus généralement :

```
( cond  (test1   e1)  Si test1 est #t  alors e1
        (test2   e2)
                ...
      [ (else    ee) ]
)
```

Exercice : définir une fonction (entredeux x y z) qui retourne la valeur qui est entre les deux autres

```
(entredeux  17 -3 45)
```

Evaluation, environnements

Expressions atomiques (valeur et symboles) :

- on retourne leur valeur

5

→ 5

Expressions non atomique :

- on évalue les arguments (l'ordre n'est pas défini)
- on applique la fonction sur les arguments déjà évalués (naturellement, récursivement)

(* 2 pi)

→ 2

→ 3.14

→ **6.28**

Evaluation d'expressions

Environnements

Environnement global :

(define …) ajoute des connaissances

Environnement local (pour limiter la porté) :

```
(let ( (ident1  expr1)
       (ident1  expr1)
        ...
       (ident1  expr1)  )    corps  )
```

Exemple :

```
(define (a_payer x)
    (let ( (taux 1.033) (frais 25) )
        (+ (* taux  x)  frais) )  )
```

Evaluation d'expressions

Fonction locale

A l'intérieur des fonctions, on peut définir des fonctions
(define …) peut contenir une séquence de définitions

Exemple :

```
(define (volume x y z)
  (define (calc1 a b ) (/ (- a b) (+ a b)))
  (+ (calc1 x y) (calc1 x z) (calc1 y z))  )
```

Fonctions récursives

Pratique : une fonction qui lit le clavier : (read)

Exemple : calculer la moyenne de n valeurs lues du clavier

La valeur moyenne des n premières valeurs :

$$M(n) = (v1 + v2 + \ldots + vn) / n$$
$$M(n) = (v1 + v2 + \ldots vn\text{-}1 + vn) / n$$
$$M(n) = (n\text{-}1) / n * (v1 + v2 + vn\text{-}1)/(n\text{-}1) + vn / n$$

$$M(n) = (n\text{-}1) / n * M(n\text{-}1) + vn / n$$

Fonctions récursives

La valeur moyenne calculée récursivement :

$$M(n) = ((n-1) * M(n-1) + vn) / n$$
$$M(1) = v1$$

```
(define (moy n)
    (if (= n 1)
        (read)
        (/ (+ (*  (moy (- n 1)) (- n 1) (read)) n) ) ) )
```

Fonctions récursives

Exercice : suite de Fibonacci

$$F(n) = F(n-1) + F(n-2)$$
$$F(0) = 0$$
$$F(1) = 1$$

1) Ecrivez la

2) Tracez les appels sous forme d’un arbre

Fonctions récursives

Exercice : suite de Fibonacci