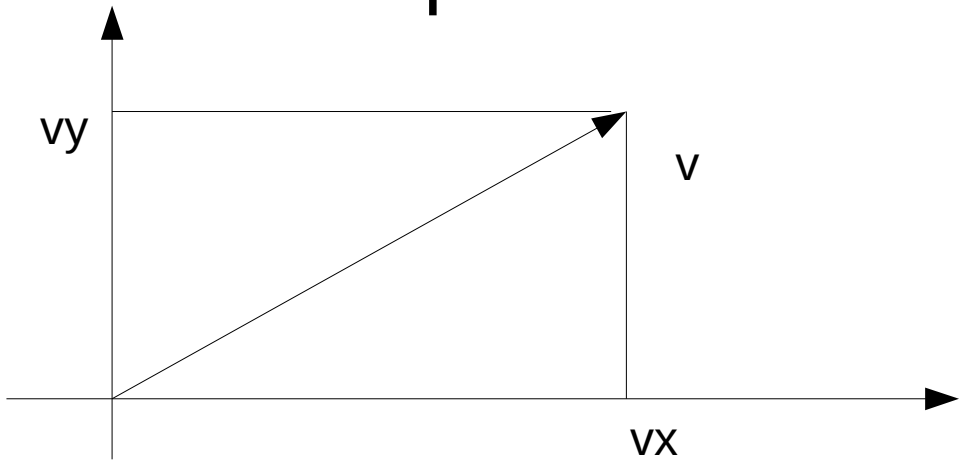
Prog Fonctionnelle, TD3

Listes et « objets »

Exemple

Vecteurs sur le plan

- Deux composants : vx et vy



Questions :

Q1) Comment les représenter / construire ?

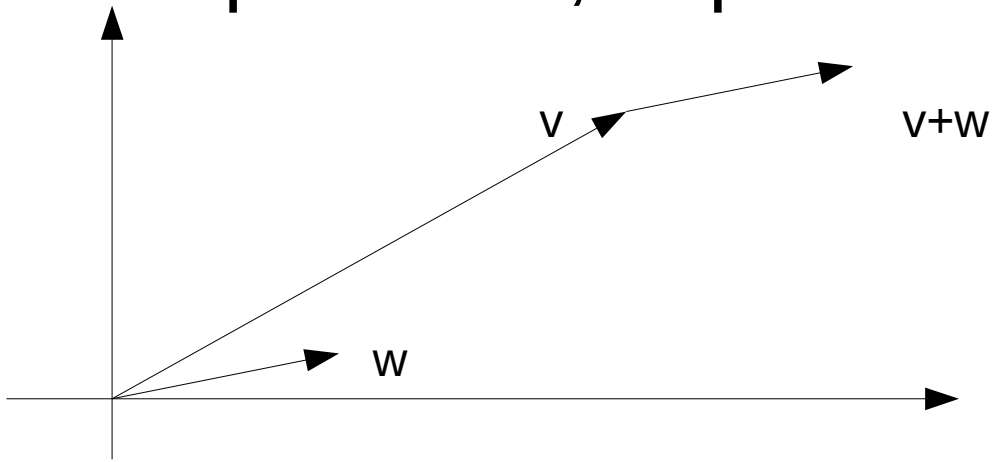
Q2) Comment les accéder ?

Q3) Comment les manipuler ?

Exemple

Vecteurs sur le plan

- Manipulations, «opérateurs»



$$Z = v + w$$

$\Rightarrow$ $zx = vx + wx$ $\Rightarrow$ puis construire l'objet à partir des composants

$zy = vy + wy$

accès

Exemple

Vecteurs sur le plan

- Manipulations, «vecteur+»

$$Z = v + w$$

⇒ zx = vx + wx ⇒ puis construire l'objet à partir des composants

zy = vy + wy

accès

```
(define (vector+ v w)
  (vecteur (+ (cx v) (cx w)) (+ (cy v) (cy w))
```

Paires

Une paire (pointée)

est un couple de S-expressions (atomiques ou composées)

- Construction d'une paire :

(cons e1 e2)

⇒ (e1 . e2)

Exemple

(cons 2 17)

⇒ (2 . 17)

Question

(cons (/ 9 3) (or #t #f))

⇒ ????

Paires

Accès aux éléments d'une paire (pointée)

Soit : (define P (cons e1 e2))

(car P) (cdr P)

Questions

(define P1 (cons (/ 9 3) (or #t #f))

(car P1) (cdr P1)

⇒ ????

(define P2 (cons 1 (cons #t #f))

(car P2) (cdr P2) (car (cdr P2))

⇒ ????

Remarque : en élément peut être une paire...

Exemple

Retours sur les vecteurs sur le plan

- Propositon :

  Q1) Comment les représenter / contruire ?

  Sous forme d'une paire, construite par (cons ..

  Q2) Comment les accéder ?

  Les composants par (car.. et (cdr..

  Q3) Comment les manipuler ?

```
(define (vecteur x y) (cons x y)
(define (cx v) (car v)
(define (cy v) (cdr v)

  (define(vector+ v w)
(vecteur (+ (cx v) (cx w)) (+ (cy v) (cy w))
```

Listes

Problème avec le produit vectoriels (vector* v w)

Le résultat est dans l’espace 3D et pas sur le plan

`Une structure plus large est nécessaire`

Listes

Une liste est une structure ordonnée d'éléments

Exemple : (1 toto #t (1 . 4))

Une construction simple

(list 3 4 8)

⇒ (3 4 8)

Une autre liste

'(toto titi tutu)

⇒ (toto titi tutu)

Une liste particulière : la liste vide ()

Listes

Accès aux éléments d'une liste, exemples

(car '(3 4 8)) le premier élément

⇒ 3

(cdr '(3 4 8)) la suite

⇒ (4 8)

Pourquoi ?

Listes et paires

Experimentation :

(cdr ‘(3 4 8))
⇒ (4 8)
(cddr ‘(3 4 8))
⇒ (8)
(cdddr ‘(3 4 8))
⇒ () la liste vide

Listes

Une liste est une paire pointée dont le deuxième élément est une liste

Exemple : (list 3 4 8)

⇒ (3 . (4 . (8 . ())))

Construction équivalente :

(cons 3 (cons 4 (cons 8 ())))

accesseur

- At

```
(define (at L n)
  (if (= n 1) (car L)
      (if (null? (cdr L)) "probleme"
          (at (cdr L) (- n 1)))))
```

Listes

Exercices : (define g (cons 2 ()))

Évaluez (tapez) g dans scheme

Quelle est la réponse ?

Listes

Exercices : Soit une liste (2 (1) 4)

Représentez cette liste sous forme de paires

Listes

Exercices : Soit une liste (2 (1) 4)

Représentez cette liste sous forme de paires

Faire progressivement :

(2 . ((1) 4))
(2 . ((1) . (4)))
(2 . ((1 . ()) . (4)))
(2 . ((1 . ()) . (4 . ())))

(cons 2 (cons (cons 1 ()) (cons 4 ())))

Listes

Exercices : (define L (list 6 (list 5 8) 3))

Représentez L sous forme de paires, puis donnez :

(car L)
(cdr L)
(cadr L)
(cdar L)
(cddr L)

Listes

Exercices : (define L (list 6 (list 5 8) 3))

Représentez L sous forme de paires, puis donnez :

(car L) 6
(cdr L) (list (list 5 8) 3)
(cadr L) (list 5 8)
(cdar L) n'existe pas
(cddr L) (list 3)

```
(6. (((5 . (8 . ()) . (3 . ())))
```

Listes

Exercices : (1 (4) 7 (2 9) 5)

Créer cette liste à l'aide de la fonction (cons…

(1 . ((4) 7 (2 9) 5))
(1 . ((4) . (7 (2 9) 5)))
(1 . ((4) . (7 . ((2 9) 5))))
(1 . ((4) . (7 . ((2 9) . (5)))))
(1 . ((4 . ()) . (7 . ((2 9) . (5)))))
(1 . ((4 . ()) . (7 . ((2 . (9 . ()) . (5)))))
(1 . ((4 . ()) . (7 . ((2 . (9 . ()) . (5 . ())))))

Listes

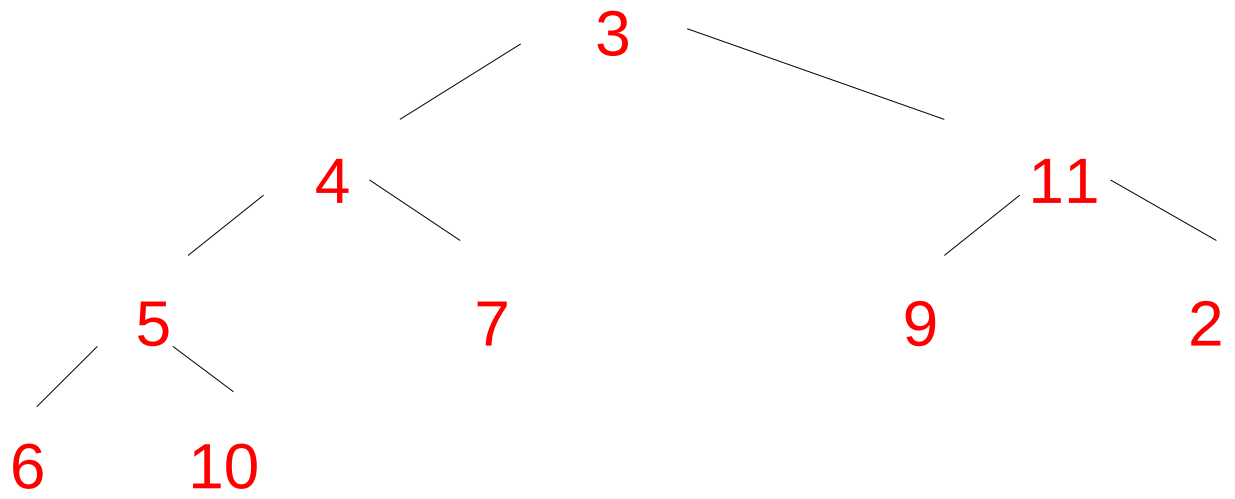
Exercices : (1 (4) 7 (2 9) 5)

Créer cette liste à l'aide de la fonction (cons...

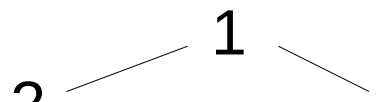
```
(cons 1
      (cons (cons 4  ())
            (cons 7
                  (cons (cons 2
                              (cons 9 ())
                        (cons 5 ())  )))))
```

Arbres

Exercice : Construire l'objet suivant :



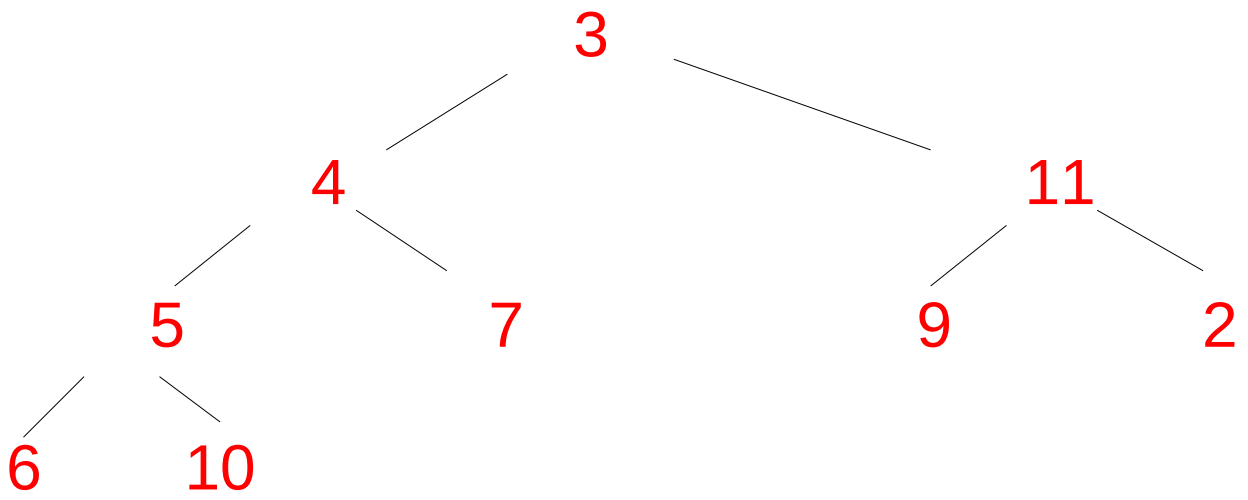
Indice : un arbre peut être une paire : (racine, liste des sous-arbres)



=> (cons 1 (list (cons 2 ()) (cons 3 ())))

Arbres

Exercice : Construire l'objet suivant :



Indice : un arbre peut être une paire : (racine, sous-arbres)

```
(cons 3 (list
    (cons 4 (list (cons 5 (list (cons 6 '()) (cons 10 '()) ) )
                  (cons 7 '()) ) )
    (cons 11 (list (cons 9 '()) (cons 2 '())) )
                   ) )
```