UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Scheme

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Fonction

- Une fonction prend des arguments et retourne un résultat
- Arguments et résultat peuvent être de n'importe quel type :
  - Nombre
  - Booléen
  - Caractère
  - Chaîne de caractères
  - Liste
  - Fonction

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Appel à une fonction

- Parenthèse ouvrante
- Nom de la fonction
- Espace
- Premier argument
- Espace
- Deuxième argument
- …
- Parenthèse fermante

```
(nom_fonction Arg1 Arg2 ... Argn)
```

- Il faut donner à la fonction le bon nombre d'arguments, et du bon type

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemples

```
(+ 3 5)
```

-> retourne 8

```
(- 15 x)
```

-> retourne la différence entre 15 et x si x a une valeur numérique, une erreur sinon

```
(+ (* 3 4) (+ 5 1))
```

-> retourne 18

```
(* 5) ou (+ 3)
```

-> ne sont pas corrects

```
(* 5 "toto")
```

-> n'est pas correct non plus

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Évaluation d'une fonction

- Lorsqu'on lui fournit un appel de fonction, Scheme :
  - Évalue chacun des arguments
  - Regarde s'il connaît la fonction, sinon affiche un message d'erreur
  - Applique la fonction aux résultats de l'évaluation des arguments
  - Affiche le résultat
- C'est un processus récursif

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les variables

- L'affectation revient à attribuer une valeur à un symbole. On utilise pour cela la forme spéciale *define*

- Exemples :

```
(define a 5)
(define b 4)
(+ a b)
```

-> retourne 9

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les fonctions

- Une fonction en Scheme est un objet qui, à partir de données, retourne un seul et unique résultat.
- Les données sont appelées paramètres ou arguments.
- Le nombre de paramètres est l'arité de la fonction.
- Une fonction à résultat booléen est appelée un prédicat. Par convention, le nom d'un prédicat se terminera par ? (integer?, real?, ...)

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemples

```
(define (double x)
    (* 2 x))
(define (carre x)
    (* x x))
```

```
(double 4)
```

- retourne 8

```
(carre 4)
```

- retourne 16

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les tests

```
(if test expr_si_vrai expr_si_faux)
```

- Deux constantes : *#t* (vrai) et *#f* (faux)
- *test* est une expression booléenne
- si sa valeur est vraie (*#t*), seule *si_vrai* est évaluée et retournée
- si sa valeur est fausse (*#f*), seule *si_faux* est évaluée et retournée

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
(define (valeur_absolue x)
   (if (> x 0) x (- x)))
```

```
(valeur_absolue 8)
```

- retourne 8

```
(valeur_absolue -8)
```

- retourne 8 également

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Conditions

- L'expression *cond* :

```
(cond
  (condition1 expr11 expr12 ... expr1N)
  (condition2 expr21 expr22 ... expr2N)
  ......
  (conditionN exprN1 exprN2 ... exprNN)
  (else expr1 expr2 ... exprN)
)
```

- Les conditions sont évaluées séquentiellement.
- Dès qu'une condition est vraie (*#t*), les expressions associées sont successivement évaluées et la valeur de la dernière expression est retournée.
- Si toutes les conditions sont fausses (*#f*), les expressions du *else* sont évaluées et sa valeur (celle de la dernière expression) est retournée

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Récursivité

- Une fonction est dite récursive si sa définition contient un appel à elle-même (en réalité c'est l'algorithme qui est récursif)

- Il faut s'assurer de l'arrêt de l'algorithme :

  - il faut toujours mettre une condition d'arrêt (sinon le programme tourne à l'infini) grâce à une expression conditionnelle dont au moins l'une des alternatives ne contient pas d'appel récursif

  - la condition d'arrêt doit devenir vraie après un nombre fini d'appels récursifs

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Exemple

```
(define (factorielle n)
  (if (= 0 n)
      1
      (* n (factorielle (- n 1)))))
```

Processus d'évaluation :

```
(factorielle 5)
(* 5 (factorielle 4))
(* 5 (* 4 (factorielle 3)))
(* 5 (* 4 (* 3 (factorielle 2))))
(* 5 (* 4 (* 3 (* 2 (factorielle 1)))))
(* 5 (* 4 (* 3 (* 2 (* 1 (factorielle 0))))))
(* 5 (* 4 (* 3 (* 2 (* 1 1)))))
(* 5 (* 4 (* 3 (* 2 1))))
(* 5 (* 4 (* 3 2)))
(* 5 (* 4 6))
(* 5 24)
120
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

La paire pointée

- Une paire pointée correspond à la notion de couple.
- Le type *Paire* est défini par l'ensemble des couples d'objets quelconques, les opérateurs permettant de manipuler des paires sont :
    - l'opérateur de construction, *cons* : prend en argument deux objets quelconques et renvoie la paire correspondante;
    - l'opérateur d'accès au premier élément, *car.*
    - l'opérateur d'accès au deuxième élément, *cdr.*
    - le prédicat de type, *pair?* : renvoie vrai (*#t*), si l'objet qu'on lui donne en argument est de type *Paire*.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Les listes

- Une liste est une collection ordonnée d'objets quelconques en nombre fini.
- À la différence d'un ensemble, un objet peut apparaître plusieurs fois dans un même liste et l'ordre est important. ((1, 2, 3) ≠ (2,1,3))
- La plus petite liste est celle qui ne contient aucun objet, on l'appelle la *liste vide*.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Opérateurs sur les listes

- *cons* : prend en argument un objet quelconque et une liste et renvoie la liste obtenue en ajoutant l'objet au début de la liste ;
- *car* : prend en argument une liste non vide et renvoie son premier élément;
- *cdr* : prend en argument une liste non vide et renvoie le reste de cette liste (c'est à dire la liste sans son premier élément) ;
- *null?* : renvoie *#t* si son argument est la liste vide;
- *list?* : renvoie *#t* si l'objet donné en argument est une liste.