

**Slide 1**

Les fonctions

**Slide 2**

0. Résumé des épisodes précédents

Le noyau impératif : décl., affectation, séquence, test, boucle

Variables mutables et finales

$e : \text{Var} \longrightarrow \text{Ref}$

$m : \text{Ref} \longrightarrow \text{Val}$

**Slide 3** mais finalement

$e : \text{Var} \longrightarrow \text{Val} (\supseteq \text{Ref})$

$m : \text{Ref} \longrightarrow \text{Val}$

$\Theta(t, e, m) = v$

$\Sigma(p, e, m) = m'$

L'exécution d'une instruction : l'affectation

La fn  $m + (x = v)$  coïncide avec  $m$  sauf en  $x$  où elle vaut  $v$

**Slide 4** –  $\Sigma(x = t ; e, m) = (m + (e(x) = v))$   
 $v = \Theta(t, e, m)$

L'exécution d'une instruction : la séquence et le test

- $\Sigma(\{p_1 p_2\}, e, m) = \Sigma(p_2, e, \Sigma(p_1, e, m))$
- $\Sigma(\text{if } (b) p_1 \text{ else } p_2, e, m) = \Sigma(p_1, e, m)$   
si  $\Theta(b, e, m) = \text{true}$   
 $\Sigma(\text{if } (b) p_1 \text{ else } p_2, e, m) = \Sigma(p_2, e, m)$   
si  $\Theta(b, e, m) = \text{false}$

Slide 5

L'exécution d'une instruction : la boucle

`while (b) q`

Abréviation pour l'instruction infinie

```
if (b) {q if (b) {q if (b) {q if (b) ...  
                                     else skip;}  
                                     else skip;}  
                                     else skip;}  
else skip;
```

Slide 6

avec  $\Sigma(\text{skip } ;, e, m) = m$

Approximations finies

```
p0 = if (b) giveup; else skip;  
p1 = if (b) {q if (b) giveup; else skip;} else skip;  
... pn+1 = if (b) {q pn} else skip;
```

$\Sigma$  jamais définie en (giveup; , e, m)

**Slide 7**

Si  $\Sigma(p_n, e, m)$  définie, alors pour tout  $n' > n$ ,  $\Sigma(p_{n'}, e, m)$  définie et  $\Sigma(p_{n'}, e, m) = \Sigma(p_n, e, m)$

Suite  $\Sigma(p_n, e, m)$  jamais définie ou définie à partir d'un certain rang et constante sur son domaine : **limite**

-  $\Sigma(\text{while } (b) \ q, e, m) = \lim_n \Sigma(p_n, e, m)$

L'exécution d'une instruction : la déclaration

-  $\Sigma(\{T \ x = t; \ q\}, e, m) = \Sigma(q, e+(x=r), m+(r=v))$

$r$  référence quelconque qui n'apparaît pas dans  $e$  et  $m$

$v = \Theta(t, e, m)$

**Slide 8**

$\Sigma(\{\text{final } T \ x = t; \ q\}, e, m) =$

$\Sigma(q, (e+(x=v)), m)$

$v = \Theta(t, e, m)$

### Les mystères de la déclaration

```
int x = 3 ; x = x + 1 ;
```

```
int x = 3 ; q
```

Pourquoi pas `int x = 3 ;` et une séquence ?

**Slide 9**

### Les mystères de la déclaration

```
int x = 3 ; x = x + 1 ;
```

```
int x = 3 ; q
```

Pourquoi pas `int x = 3 ;` et une séquence ?

**Slide 10**

Éléments d'une séquence : **exécutés dans le même env.**

$$\Sigma(\{p_1 \ p_2\}, e, m) = \Sigma(p_2, e, \Sigma(p_1, e, m))$$

Or la déclaration modifie l'environnement de `q`

$$\Sigma(\{T \ x = t ; \ q\}, e, m) = \Sigma(q, e+(x=r), m+(r=v))$$

### Un exercice

```
p = {{int x = 3; x = x + 1;} y = 8;}
```

```
e = [y = r]
```

**Slide 11**

```
m = [r = 7]
```

$\Sigma(p, e, m)$  ?

Faites ce que je dis, pas ce que je fais

```
{
  {
    int x = 3;
    x = x + 1;
  }
  y = 8;
}
```

**Slide 12**

```
p = {{int x = 3; x = x + 1;} y = 8;}
```

Idem toujours des {} dans un while ou un if

**Slide 13**

## I. Les fonctions

**Slide 14**

```
System.out.print("Le RER en direction de ");
System.out.print("Saint-Rémy-lès-Chevreuse");
System.out.print(" partira à ");
System.out.println("8h50");
System.out.println();
System.out.println();
System.out.println();
System.out.print("Le RER en direction de ");
System.out.print("Massy-Palaiseau");
System.out.print(" partira à ");
System.out.println("8h55");
System.out.println();
System.out.println();
System.out.println();
```

**Slide 15**

```
static void sauterTroisLignes () {  
    System.out.println();  
    System.out.println();  
    System.out.println();}
```

```
System.out.print("Le RER en direction de ");  
System.out.print("Saint-Rémy-lès-Chevreuse"););  
System.out.print(" partira à ");  
System.out.println("8h50");  
sauterTroisLignes();
```

**Slide 16**

```
System.out.print("Le RER en direction de ");  
System.out.print("Massy-Palaiseau"););  
System.out.print(" partira à ");  
System.out.println("8h55");  
sauterTroisLignes();
```

### Le passage d'arguments

```
static void annoncerRer (final String d,  
                        final String h) {  
    System.out.print("Le RER en direction de ");  
    System.out.print(d);  
    System.out.print(" partira à ");  
    System.out.println(h);  
    System.out.println(); System.out.println();  
    System.out.println();}  
annoncerRer("Saint-Rémy-lès-Chevreuse", "8h50");  
annoncerRer("Massy-Palaiseau", "8h55");
```

Slide 17

### Le retour de valeur

```
a = 3;  
b = 4;  
c = 5;  
d = 12;  
u = Math.sqrt(a * a + b * b);  
v = Math.sqrt(c * c + d * d);
```

Slide 18

```
static double hypotenuse (final double x,  
                          final double y) {  
    return Math.sqrt(x * x + y * y);}
```

**Slide 19**

```
a = 3;  
b = 4;  
c = 5;  
d = 12;  
u = hypotenuse(a,b);  
v = hypotenuse(c,d);
```

### Un peu de syntaxe

Java :

```
static T f (final T1 x1, ..., Tn xn) p
```

Caml (arguments toujours finaux) :

**Slide 20**

```
let f x1 ... xn = t in p
```

```
let hypotenuse x y = sqrt(x *. x +. y *. y)
```

C :

```
T f (const T1 x1, ..., Tn xn) p
```

```
int f (const int x, int y) return x + 1;
```

## L'instruction return

Caml :

```
let hypotenuse x y = sqrt(x *. x +. y *. y)
```

### Slide 21

En Java et C la valeur à renvoyer doit être précédée de `return`

```
static double hypotenuse (final double x,  
                          final double y) {  
    return Math.sqrt(x * x + y * y);}
```

`return` interrompt le déroulement de la fonction

### Slide 22

```
static int signe (final int x) {  
    if (x < 0) return -1;  
    if (x == 0) return 0;  
    return 1;}
```

## Les fonctions et les procédures

Une fonction peut

- effectuer une action (e.g. afficher quelque chose, modifier la mémoire)
- retourner une valeur

### Slide 23

Une fonction qui ne retourne pas de valeur : une procédure

Certains langages (Pascal) : syntaxe différente

Java et C : type de retour remplacé par `void` (`void` pas un type)

Caml : une procédure retourne une valeur de type `unit`

## Les fonctions et les procédures

Appel d'une fonction : expression

```
x = hypotenuse(3,4) + 8 ;
```

### Slide 24

Appel d'une procédure : instruction

```
sauterTroisLignes() ;
```

Mais ... un appel de fonction peut aussi être une instruction

## Les variables globales

```
int x;  
x = 3;  
x = 0;
```

**Slide 25**

```
static void reset () {x = 0;}
```

```
int x;  
x = 3;  
reset();
```

```
static int x;
```

**Slide 26**

```
static void reset () {x = 0;}
```

```
x = 3;  
reset();
```

Le programme principal

```
static T1 x1 = t1 ;  
...  
static Tn xn = tn ;
```

Slide 27

```
static ... f1 (...) ...  
...  
static ... fn' (...) ...
```

p

Le programme principal

```
class Prog {  
    static T1 x1 = t1 ;  
    ...  
    static Tn xn = tn ;  
    static ... f1 (...) ...  
    ...  
    static ... fn' (...) ...  
    public static void main (String [] args) {  
        p}}}
```

Slide 28

Slide 29

## II. La fonction $\Sigma$

La dernière fois

La valeur d'une expression

$$\Theta(t, e, m) = v$$

Slide 30

Ce qui se passe quand on exécute une instruction

$$\Sigma(p, e, m) = m'$$

### Trois nouveautés

1. Outre une instruction, un environnement et une mémoire, on a besoin d'un **environnement global** pour les définitions de fonctions et les variables globales

**Slide 31**

$\Sigma(p, e, m, G) = m'$

Idem pour les expressions

2. Calculer la valeur d'une expression peut désormais modifier la mémoire :  $f(4)$

**Slide 32**

```
static int f (int x) {  
    n = n + 1;  
    return 2 * x;}
```

$\Theta(t, e, m, G) = (v, m')$

3. Il faut traduire le fait que l'instruction `return` interrompt le déroulement de la fonction

**Slide 33**

3. Il faut traduire le fait que l'instruction `return` interrompt le déroulement de la fonction

Est-il toujours vrai que

$$\Sigma(\{p_1 \ p_2\}, e, m, G) = \Sigma(p_2, e, \Sigma(p_1, e, m, G), G)$$

**Slide 34** ?

Quid de `if (x < 0) return -1; return 1;?`

3. Il faut traduire le fait que l'instruction `return` interrompt le déroulement de la fonction

Est-il toujours vrai que

$$\Sigma(\{p_1 \ p_2\}, e, m, G) = \Sigma(p_2, e, \Sigma(p_1, e, m, G), G)$$

**Slide 35** ?

Quid de `if (x < 0) return -1; return 1;`?

$\Sigma(p, e, m, G)$  est désormais ou bien `(normal, m')` ou bien `(return, v, m')`

#### L'évaluation d'une expression

- $\Theta(x, e, m, G) = (m(e(x)), m)$ , si  $x$  mutable,
- $\Theta(x, e, m, G) = (e(x), m)$ , si  $x$  finale,
- $\Theta(c, e, m, G) = (c, m)$ ,

**Slide 36** -  $\Theta(t + u, e, m, G) = (v + w, m')$

où  $(v, m') = \Theta(t, e, m, G)$

et  $(w, m'') = \Theta(u, e, m', G)$ ,

- idem pour les autres opérations

–  $\Theta(f(t_1, \dots, t_n), e, m, G)$  ?

**Slide 37**

–  $\Theta(f(t_1, \dots, t_n), e, m, G)$  ?  
 $x_1, \dots, x_n$  arguments formels et  $p$  le corps de  $f$  (dans  $G$ )  
 $e'$  environnement de variables globales de  $G$

**Slide 38**

- $\Theta(f(t_1, \dots, t_n), e, m, G)$  ?
- $x_1, \dots, x_n$  arguments formels et  $p$  le corps de  $f$  (dans  $G$ )
- $e'$  environnement de variables globales de  $G$
- $(v_1, m_1) = \Theta(t_1, e, m, G)$
- $(v_2, m_2) = \Theta(t_2, e, m_1, G)$
- ...  $(v_n, m_n) = \Theta(t_n, e, m_{n-1}, G)$

Slide 39

- $\Theta(f(t_1, \dots, t_n), e, m, G)$  ?
- $x_1, \dots, x_n$  arguments formels et  $p$  le corps de  $f$  (dans  $G$ )
- $e'$  environnement de variables globales de  $G$
- $(v_1, m_1) = \Theta(t_1, e, m, G)$
- $(v_2, m_2) = \Theta(t_2, e, m_1, G)$
- ...  $(v_n, m_n) = \Theta(t_n, e, m_{n-1}, G)$
- $e'' = e' + (x_1 = v_1) + (x_2 = v_2) + \dots + (x_n = v_n)$
- $m'' = m_n + (r_2 = v_2) + \dots + (r_n = v_n)$
- Si  $\Sigma(p, e'', m'', G)$  a la forme  $(\text{return}, v, m''')$
- alors  $(v, m''')$

Slide 40

## L'exécution des instructions

- déclaration, affectation, test, boucle : ras
- séquence : si  $\Sigma(p_1, e, m, G) = (\text{normal}, m')$   
alors  $\Sigma(\{p_1 \ p_2\}, e, m, G) = \Sigma(p_2, e, m', G)$   
et si  $\Sigma(p_1, e, m, G) = (\text{return}, v, m')$   
alors  $\Sigma(\{p_1 \ p_2\}, e, m, G) = (\text{return}, v, m')$

Slide 41

- Appel de fonction idem expressions  
si  $\Sigma(p, e'', m'', G) = (\text{normal}, m''')$  alors  
 $(\text{normal}, m''')$   
et si  $(\text{return}, v, m''')$  alors  $(\text{normal}, m''')$
- si  $\Theta(t, e, m, G) = (v, m')$ ,  
alors  $\Sigma(\text{return } t ; e, m, G) = (\text{return}, v, m')$

Slide 42

### Un exemple

```
static double hypotenuse (double x, double y) {  
    return Math.sqrt(x * x + y * y);}
```

#### Slide 43

```
public static void main (String [] args) {  
    a = 3;  
    b = 4;  
    u = hypotenuse(a,b);  
    System.out.println(u);}
```

Et si a, b, u locales à main?

Slide 44 Exercices 2.3, 2.4, 2.5.

**Slide 45**    La prochaine fois :  
Le passage par valeur et le passage par référence - la récursivité