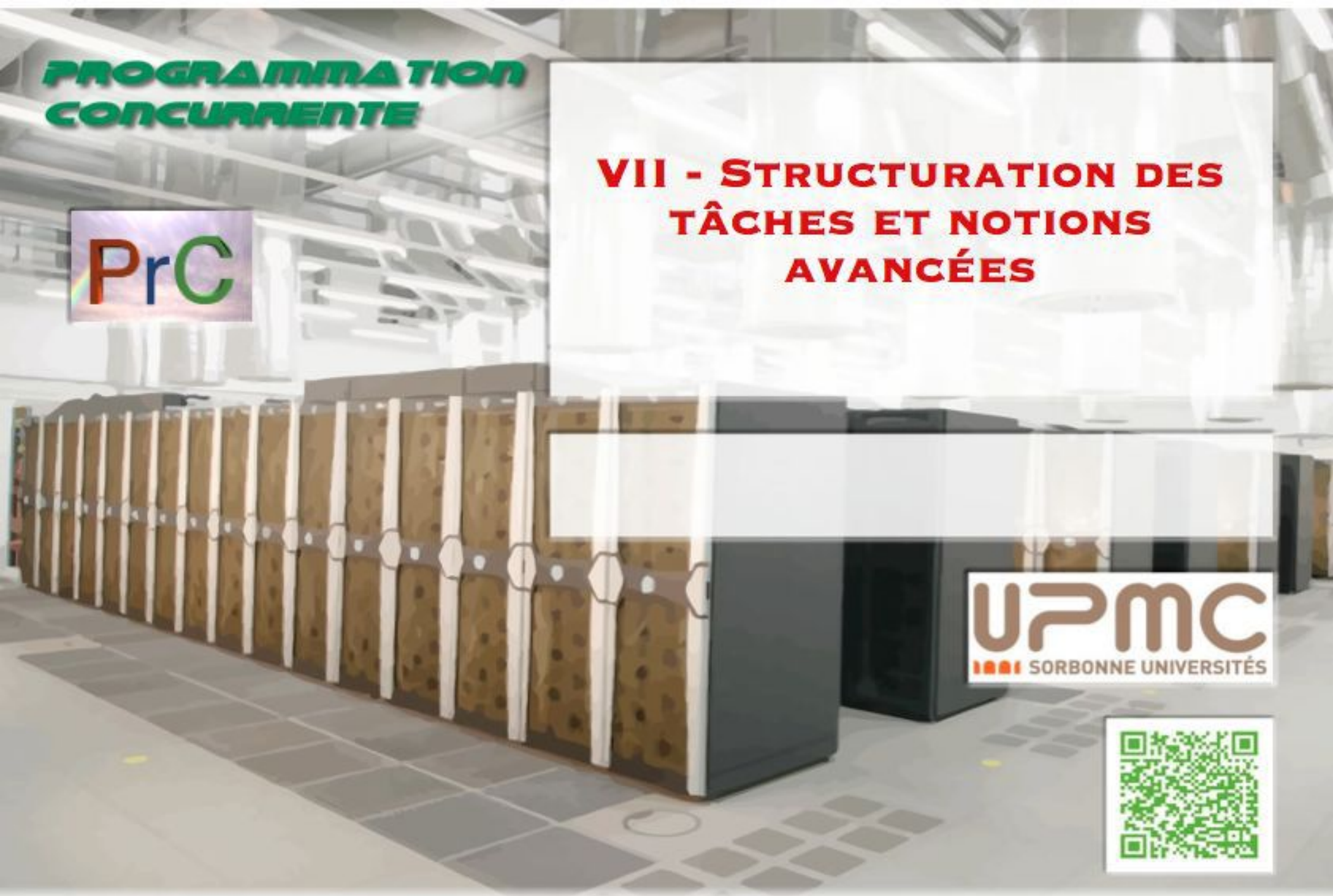

PROGRAMMATION CONCURRENTE
PrC
VII - STRUCTURATION DES TÂCHES ET NOTIONS AVANCÉES
UPMC
SORBONNE UNIVERSITÉS

27 - Récursion classique et multi-processus

Programmation Concurrente - LI330
Université P. & M. Curie - année scolaire 2013/2014

PrC

Définition

- Récursivité
  - Origine: la notion d'induction (raisonnement par récurrence)
  - Récursivité: lorsqu'un sous-programme se rappelle lui-même
  - Objectif: résoudre un calcul à partir d'une valeur inférieure
  - Nécessité: connaître le résultat du problème pour une valeur donnée auquel on peut se ramener
- Exemple: factorielle
  - n! = n x (n - 1)!
  - 0! = 1
- Un exemple de déroulement:

```
5!  = 5 *4!
      4 *3!
        3 *2!
          2 *1!
            1 *0!
```

Expression en programmation

- Invocation de la fonction dans une expression du corps de la fonction
- Appel de la procédure dans une instruction du corps de la procédure

Un exemple: calcul récursif du reste

- Principe de l'algorithme
  - 0 mod B = 0 quel que soit B
  - A mod B = (A -1 mod B) + 1
    - Attention, A mod B est une valeur comprise entre 0 et B-1
    - Si A -1 mod B = B-1, alors rendre 0
- Algorithme de mod (A, B) = R

```
SI A = 0 ALORS
   R := 0;
SINONSI mod (A - 1 , B) = B - 1 ALORS
   R := 0;
SINON
   R := mod (A - 1 , B) + 1
FSI
```

Première implémentation


```
-- Première version de la fonction modulo récursive
function Modulo (A , B : Natural) return Natural is

begin
   if A = 0 then
      return 0;
   elsif Modulo (A - 1, B) = B - 1 then
      return 0;
   else
      return Modulo (A - 1, B) + 1;
   end if;
end Modulo;
```

Seconde implémentation


```
-- Seconde (et meilleure) version de la fonction modulo récursive
function Modulo (A , B : Natural) return Natural is

   Calcul_Intermediaire : Integer;

begin
   if A = 0 then
      return 0;
   else
      Calcul_Intermediaire := Modulo (A - 1, B);
      if Calcul_Intermediaire = B - 1 then
         return 0;
      else
         return Calcul_Intermediaire + 1;
      end if;
   end if;
end Modulo;
```

Rappels: Suivi de l'exécution

Exécution de Modulo (4, 3)

```
--> Modulo (A[1] = 4, B[1] = 3)
  --> Modulo (A[2] = 3, B[2] = 3)
    --> Modulo (A[3] = 2, B[3] = 3)
      --> Modulo (A[4] = 1, B[4] = 3)
        --> Modulo (A[5] = 0, B[5] = 3)
        <-- retour de la valeur 0
        Calcul_Intermediaire[4] = 0
      <-- retour de la valeur 1
      Calcul_Intermediaire[3] = 1
    <-- retour de la valeur 2
    Calcul_Intermediaire[2] = 2
  <-- retour de la valeur 0
  Calcul_Intermediaire[1] = 0
```

```
function Modulo (A , B : Integer) return
   Integer is

   Calcul_Intermediaire : Integer;

begin
   if A = 0 then
      return 0;
   else
      Calcul_Intermediaire :=
                        Modulo (A - 1, B);
      if Calcul_Intermediaire = B - 1
      then
         return 0;
      else
         return Calcul_Intermediaire + 1;
      end if;
   end if;
end Modulo;
```

Rappels: usage de la pile

Tranche de la pile d'exécution au niveau de l'appel suivant

--> Modulo (A[1] = 4, B[1] = 3)
--> Modulo (A[2] = 3, B[2] = 3)
--> Modulo (A[3] = 2, B[3] = 3)
--> Modulo (A[4] = 1, B[4] = 3)
--> Modulo (A[5] = 0, B[5] = 3)
<-- retour de la valeur 0
Calcul_Intermediaire[4] = 0
<-- retour de la valeur 1
Calcul_Intermediaire[3] = 1
<-- retour de la valeur 2
Calcul_Intermediaire[2] = 2

Valeur de retour (1er appel)	??
Paramètre A[1]	4
Paramètre B[1]	3
Calcul_Intermediaire [1]	??

Valeur de retour (2ème appel)	??
Paramètre A[2]	3
Paramètre B[2]	3
Calcul_Intermediaire [2]	??

Valeur de retour (3ème appel)	??
Paramètre A[3]	2
Paramètre B[3]	3
Calcul_Intermediaire [3]	2

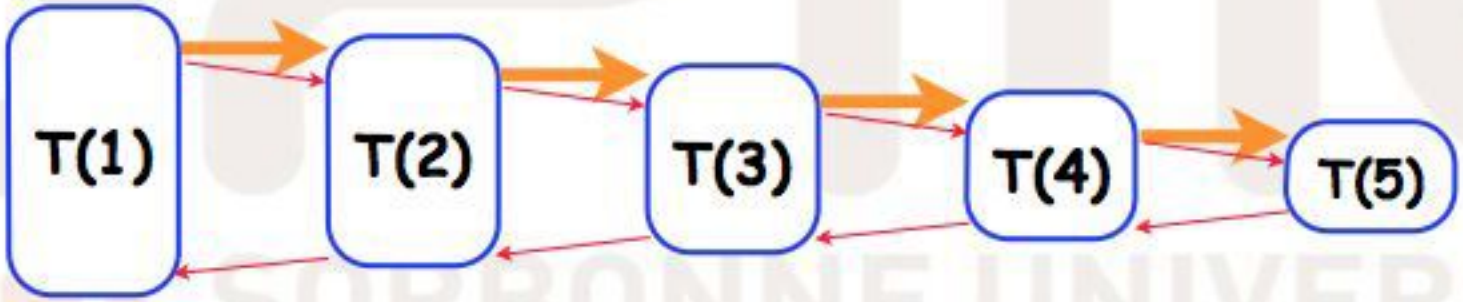
Valeur de retour (4ème appel)	1

Rappels: autres types de récursivité

- Récursivité croisée
  - A invoque B qui invoque A...
- Double récursivité
  - A invoque deux fois A (pour des valeurs différentes)
- Attention à la complexité d'exécution
  - Trop d'appels saturent la pile d'exécution qui n'est bien sûr pas infinie

Récursion et parallélisme

- Que se passe-t-il lors d'appels récursifs?
  - Empilement de contextes sur une pile d'exécution
- Peut-on «empiler» des contexte autrement?
  - Oui, avec des tâches/threads/processus
  - Chacun d'eux permet de gérer une occurrence de contexte



- Exemples classiques
  - Crible d'Eratosthène
  - Structuration des tâches en anneau