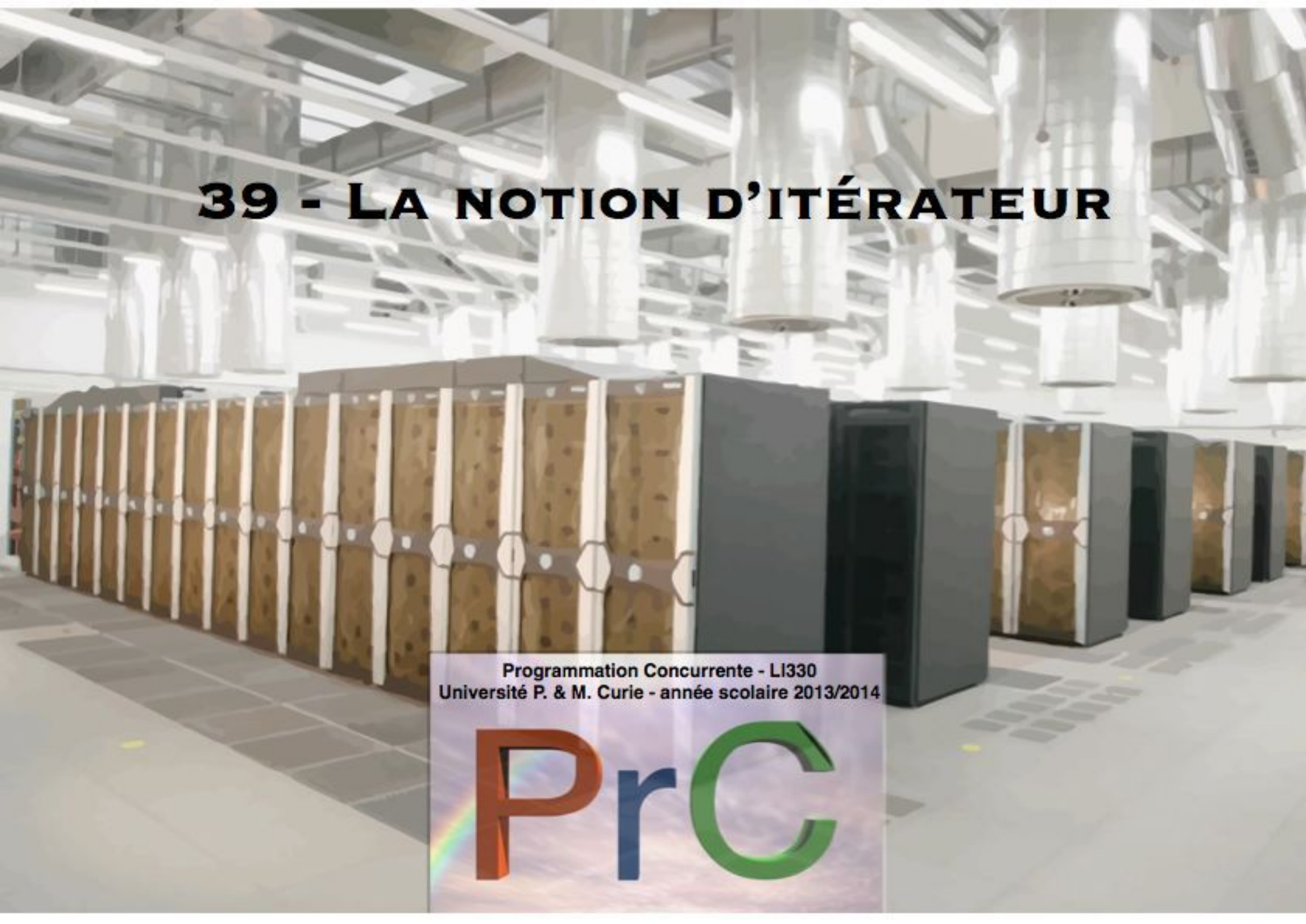

39 - La notion d'itérateur

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

PrC

Définition de la notion d'itérateur

- Primitive de parcours d'une structure permettant d'appliquer une opération dans cette structure
- Exemples typiques
  - Parcours de liste
  - Parcours d'arbres
- Application sur un exemple: le parcours préfixe dans l'unité de gestion d'arbres binaires génériques
- Itérateur différent de primitives de parcours

Exemple 1: itérateur simple

- Modification de l'unité Arbres_Binaires_Gen
- Insérer un itérateur de parcours postfixe...
- Utiliser la généricité pour permettre à l'utilisateur de spécifier le traitement à appliquer

Exemple 1: spécification de Arbres_Binaires_Gen2 (1)

arbres_binaires_gen2.ads

```
-- Gestionnaire d'arbres binaires générique
generic
   type Un_Contenu is limited private;
   with procedure Copie (A : in Un_Contenu;
                         B : out Un_Contenu);
   with function "=" (A, B : Un_Contenu) return Boolean is <>;
package Arbres_Binaires_Gen2 is
   -- ...
   -- Un type pour définir si on est à gauche ou à droite de l'arbre
   type Direction is (Gauche, Droite);
   -- le type arbre binaire (limité-privé)
   type Un_Arbre_Binaire is limited private;
   -- ...
   -- Se déplacer dans l'arbre.
   -- raise Arbre_Vide_Error
   procedure Aller_Racine (A : in out Un_Arbre_Binaire);
   -- raise Element_Courant_Null_Error
   procedure Descendre (A : in out Un_Arbre_Binaire;
                        D : in Direction);
```

Fonctions de parcours

Exemple 1: spécification de Arbres_Binaires_Gen2 (2)

- Procédure générique pour permettre d'associer un traitement particulier
- Instanciation à deux niveaux
- En C, on passerait par des pointeurs sur des fonctions

```
   -- ...
   -- Itérateur implémentant un parcours postfixe.
   generic
      with procedure J_Applique (Etiquette_Noeud : in out Un_Contenu;
                                 Profondeur : in Natural;
                                 Est_Feuille : in Boolean);
   procedure Parcours_Postfixe (A : in out Un_Arbre_Binaire);

   -- ...
end Arbres_Binaires_Gen2;
```

Itérateur, on passe en paramètre la structure sur laquelle on appliquera le parcours

Exemple 1: corps de Parcours_Postfixe


```
procedure Parcours_Postfixe (A : in out Un_Arbre_Binaire) is

   procedure Effectue_Parcours (N : in A_Noeud;
                                P : in Natural := 0) is

   begin
      if N /= null then
         Effectue_Parcours (N.Descendance (Gauche), P + 1);
         Effectue_Parcours (N.Descendance (Droite), P + 1);
         J_Applique (Etiquette_Noeud => N.all.Contenu,
                     Profondeur => P,
                     Est_Feuille => N.Descendance(Gauche)=null and
                                    N.Descendance(Droite)=null);
      end if;
   end Effectue_Parcours;
begin
   Effectue_Parcours (A.Racine);
end Parcours_Postfixe;
```

dans
arbres_binaires_gen2.adb

Exemple 1: instanciation de l'unité

- L'instanciation de Arbres_Binaires_Gen2 se fait dans les mêmes conditions que Arbres_Binaires_Gen

```
-- Gestionnaire d'arbres binaires d'entiers obtenu par instanciation
with Arbres_Binaires_Gen2, Copie_Entiers;

package Arbres_Binaires_D_Entiers2 is new Arbres_Binaires_Gen2
                (Un_Contenu => Integer,
                 "=" => "=",
                 Copie => Copie_Entiers);
```

- Cette instanciation permet d'accéder à l'itérateur pour des arbres binaires d'entiers.

arbres_binaires_d_entiers2.ads

Exemple 1: instanciations de l'itérateur (1)


```
with Ada.Text_IO, Arbres_Binaires_D_Entiers2, Ada.exceptions;
use Ada.Text_IO, Arbres_Binaires_D_Entiers2, Ada.exceptions;

procedure T_Arbres_Binaires_Entiers2 is

   -- Seule ligne modifiée afin de ne pas avoir à changer les déclarations qui suivent
   subtype Un_Arbre_Binaire_D_Entiers is
       Arbres_Binaires_D_Entiers2.Un_Arbre_Binaire;

   procedure Affiche_Un_Noeud (Val : in out Integer;
                               Prof : in Natural;
                               Est_Feuille : in Boolean) is
      Typ_Noeud : Character := 'N';
   begin
      if Est_Feuille then
         Typ_Noeud := 'F';
      end if;
      Put_Line ((1.. Prof * 3 => ' ') & Integer'Image (Val) &
                " -> " & Typ_Noeud);
   end Affiche_Un_Noeud;
```

T_arbres_binaires_entiers2.ads

Exemple 1: instanciations de l'itérateur (2)


```
procedure Ecrase_Une_Feuille (Val : in out Integer;
                              Prof : in Natural;
                              Est_Feuille : in Boolean) is

begin
   if Est_Feuille then
      Val := Val + Prof;
   end if;
end Ecrase_Une_Feuille;


-- Les instanciations pour: 1) affichage écran, 2) modification de l'arbre...
procedure Affiche_Arbre_Postfixe is
   new Parcours_Postfixe (Affiche_Un_Noeud);
procedure Modifie_Arbre_Postfixe is
   new Parcours_Postfixe (Ecrase_Une_Feuille);
```

Exemple 1: instanciations de l'itérateur (3)


```
   A:Un_Arbre_Binaire_D_Entiers;

begin -- T_Arbres_Binaires_Entiers2
   Ajoute_Noeud (1, Gauche, A);
   Ajoute_Noeud (2, Gauche, A);
   Ajoute_Noeud (3, Droite, A);
   Descendre (A, Droite);
   Ajoute_Noeud (4, Gauche, A);
   Ajoute_Noeud (5, Droite, A);
   Descendre (A, Droite);
   Ajoute_Noeud (6, Droite, A);
   Descendre (A, Droite);
   Ajoute_Noeud (7, Droite, A);
   Aller_Racine (A);
   Descendre (A, Gauche);
   Ajoute_Noeud (8, Gauche, A);
   Descendre (A, Gauche);
   Ajoute_Noeud (9, Gauche, A);
   Descendre (A, Gauche);
   Ajoute_Noeud (10, Gauche, A);
   Descendre (A, Gauche);
   Ajoute_Noeud (11, Gauche, A);
   Put_Line ((1 .. 40 => '-'));
   Affiche_Arbre_Postfixe (A);
   Put_Line ((1 .. 40 => '-'));
   Modifie_Arbre_Postfixe (A);
   Affiche_Arbre_Postfixe (A);
   Put_Line ((1 .. 40 => '-'));
end T_Arbres_Binaires_Entiers2;
```

Exemple 1: Exécution


```
$ t_arbres_binaires_entiers2
----------------------------------------
                    11 -> F
                10 -> N
            9 -> N
        8 -> N
    2 -> N
        4 -> F
                7 -> F
            6 -> N
        5 -> N
    3 -> N
1 -> N
----------------------------------------
                    16 -> F
                10 -> N
            9 -> N
        8 -> N
    2 -> N
        6 -> F
                11 -> F
            6 -> N
        5 -> N
    3 -> N
1 -> N
----------------------------------------
```

Exemple 2: itérateur avec échappement

- Modification de l'unité Arbres_Binaires_Gen2
- Insérer un itérateur de parcours postfixe...
- Utiliser la généricité pour permettre à l'utilisateur de spécifier le traitement à appliquer
- Permettre à l'utilisateur, par le biais du traitement qu'il applique, de terminer prématurément le parcours

Exemple 2: spécification de Arbres_Binaires_Gen3


```
-- Gestionnaire d'arbres binaires générique
generic
   type Un_Contenu is limited private;
   with procedure Copie (A : in Un_Contenu;
                         B : out Un_Contenu);
   with function "=" (A, B : Un_Contenu) return Boolean is <>;
package Arbres_Binaires_Gen3 is
   -- ...
   -- Itérateur implémentant un parcours postfixe.
   generic
      -- Ici, la valeur du booléen rendu dans J_Arrete indique
      -- s'il faut continuer (valeur vraie) ou arrêter (valeur fausse)
      with procedure J_Applique (Etiquette_Noeud : in out Un_Contenu;
                                 Profondeur : in Natural;
                                 Est_Feuille : in Boolean;
                                 J_Arrete : out Boolean);
   procedure Parcours_Postfixe_Stopable
                              (A : in out Un_Arbre_Binaire);


   -- ...
end Arbres_Binaires_Gen3;
```

Exemple 2: corps de Parcours_Postfixe_Stoppable


```
procedure Parcours_Postfixe_Stopable (A : in out Un_Arbre_Binaire) is
   Stop : exception;
   procedure Effectue_Parcours (N : in A_Noeud;
                                P : in Natural := 0) is
      Halte : Boolean := False;
   begin
      if N /= null then
         Effectue_Parcours (N.Descendance (Gauche), P + 1);
         Effectue_Parcours (N.Descendance (Droite), P + 1);
         J_Applique (Etiquette_Noeud => N. Contenu, Profondeur => P,
                     Est_Feuille => N.Descendance(Gauche)=null and
                     N.Descendance (Droite)=null, Halte);
         if Halte then raise Stop; end if;
      end if;
   end Effectue_Parcours;
begin
   Effectue_Parcours (A.Racine);
exception
   when Stop =>
      null;
end Parcours_Postfixe_Stopable;
```

Exemple 2: instanciation de l'itérateur (1)


```
with Ada.Text_IO, Arbres_Binaires_D_Entiers3, Ada.exceptions;
use Ada.Text_IO, Arbres_Binaires_D_Entiers3, Ada.exceptions;
procedure T_Arbres_Binaires_Entiers3 is
   subtype Un_Arbre_Binaire_D_Entiers is
               Arbres_Binaires_D_Entiers3.Un_Arbre_Binaire;

   procedure Affiche_Et_Bloque_Si_5 (Val : in Integer;
                                     Prof : in Natural;
                                     Est_Feuille : in Boolean;
                                     Arret : out Boolean) is

      Typ_Noeud : Character := 'N';

   begin
      if Est_Feuille then
         Typ_Noeud := 'F';
      end if;
      Put_Line ((1.. Prof * 3 => ' ') & Integer'Image (Val) &
                  " -> " & Typ_Noeud);
      Arret := Val = 5;
   end Affiche_Et_Bloque_Si_5;
```

Exemple 2: instanciation de l'itérateur (2)


```
   procedure Affiche_Jusqu_A_5 is new
      Parcours_Postfixe_Stopable (Affiche_Et_Bloque_Si_5);

   A : Un_Arbre_Binaire_D_Entiers;

begin -- T_Arbres_Binaires_Entiers3
   Ajoute_Noeud (1, Gauche, A); Ajoute_Noeud (2, Gauche, A);
   Ajoute_Noeud (3, Droite, A); Descendre (A, Droite);
   Ajoute_Noeud (4, Gauche, A); Ajoute_Noeud (5, Droite, A);
   Descendre (A, Droite); Ajoute_Noeud (6, Droite, A);
   Descendre (A, Droite); Ajoute_Noeud (7, Droite, A);
   Aller_Racine (A); Descendre (A, Gauche);
   Ajoute_Noeud (8, Gauche, A); Descendre (A, Gauche);
   Ajoute_Noeud (9, Gauche, A);Descendre (A, Gauche);
   Ajoute_Noeud (10, Gauche, A); Descendre (A, Gauche);
   Ajoute_Noeud (11, Gauche, A);

   Affiche_Jusqu_A_5 (A);
end T_Arbres_Binaires_Entiers3;
```

Exemple 2: exécution

```
$ t_arbres_binaires_entiers3
                    11 -> F
                  10 -> N
              9 -> N
          8 -> N
      2 -> N
          4 -> F
                  7 -> F
              6 -> N
          5 -> N
```