Type accès

Chapitre 10

Objets statiques

Ils sont **déclarés**

Un **nom** est associé à leur valeur

Leur **durée de vie** est déterminée par leur bloc de déclaration

L'allocation de l'espace mémoire nécessaire à leur représentation est effectuée par le compilateur

Objets dynamiques

Ils ne sont pas déclarés

Il n’y a pas d’association nom-valeur

Leur durée de vie est indépendante du bloc de déclaration

L’allocation mémoire est réalisée dynamiquement

Type **access**

Intérêt

- Les objets de type `access` permettent d'accéder des objets construits dynamiquement
- Ils peuvent être considérés comme des **pointeurs**

Déclaration

```
<déclaration_type_access>::=
     type <ident_type> is access <def_type>;
```

Valeurs d'un type `access`

- La déclaration d'un type `access` crée un ensemble d'adresses permettant de référencer des objets d'un type donné

Type access : exemple

```
type Ref_int is access Integer;
X:Ref_int;
```

- L'ensemble des valeurs du type `Ref_int` est un ensemble d'adresses ne pouvant référencer que des objets de type `Integer` ou d'un sous-type d'`Integer` .

- Les valeurs de la variable (statique) `X` sont ces adresses.

- La valeur `NULL` est une valeur d'adresse particulière. Elle ne référence aucun objet.

Création dynamique d'objet

Le constructeur **new**

<création_objet_dynamique>::=
new *<type_obj_référencé>'(<exp>)*

Sémantique

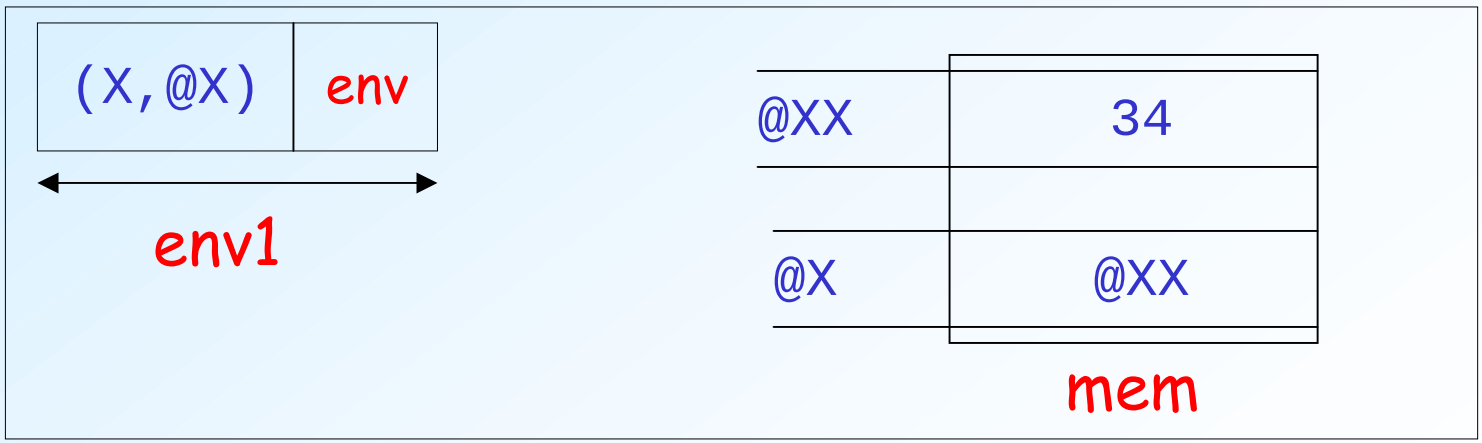
- **Crée** un objet du type référencé (objet anonyme) => allocation de la ressource mémoire nécessaire
- **Initialise** cet objet avec la valeur de l'expression
- **Retourne** l'*adresse* de cet objet

Exemple (1/2)

Soit la déclaration dans l'environnement (env, mem)

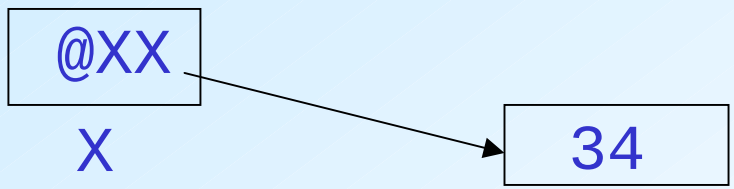
```
X:Ref_int:= new Integer'(34);
```

Etat après la déclaration



Exemple (2/2)

Autre représentation



2 objets ont été créés

- Une variable de type `Ref_int` (pointeur sur un objet de type `Integer`)
- Une variable anonyme de type `Integer` et de valeur `34`
- La portée de l'objet anonyme (crée dynamiquement) est liée à la portée de la variable du type **`access`**.

Le constructeur `all`

Comment à partir d'une variable de type **`access`**
accéder à l'objet référencé ?

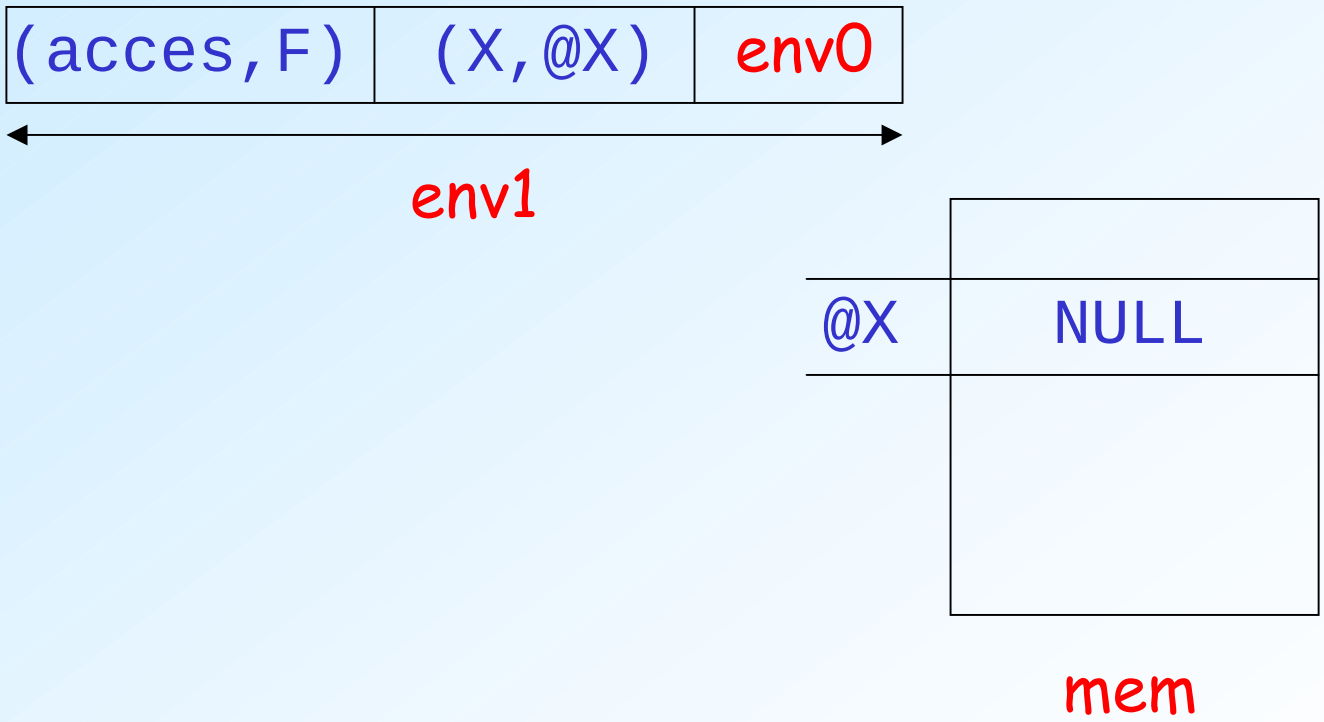
```
X.all -- désigne l'objet référencé par X
```

Pour accéder à un champ d'un objet créé dynamiquement

```
type Point is record x,y:Float; end record;
type Ref_Point is access Point;
P:Point:= new Point'(3.62,7.98);
put(P.all.x);put(P.all.y);
```

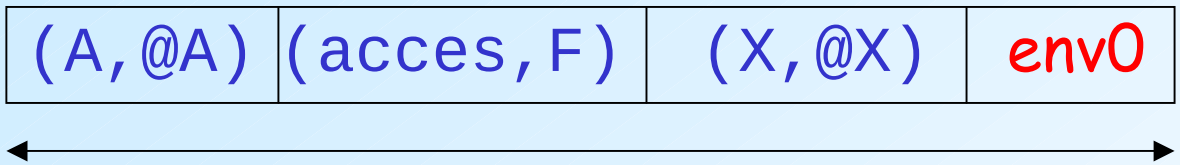
```
declare
   type Ref_int is access Integer;
   X:Ref_int:= NULL;
   procedure acces(A:out Ref_int) is
   begin
   -- état 2 (état initial d'exécution)
      A:= new Integer'(100);
   -- état 3 (état final de l'exécution)
   end acces;
   -- état 1 (après les déclarations)
begin
   acces(X);
   -- état 4 (état de retour après l'appel de la
   -- procédure
   put(X.all);
end;
```

Etat 1 (après les déclarations)



F=<<A→Corps de acces, env1>>

Etat 2 (état initial d'exécution)



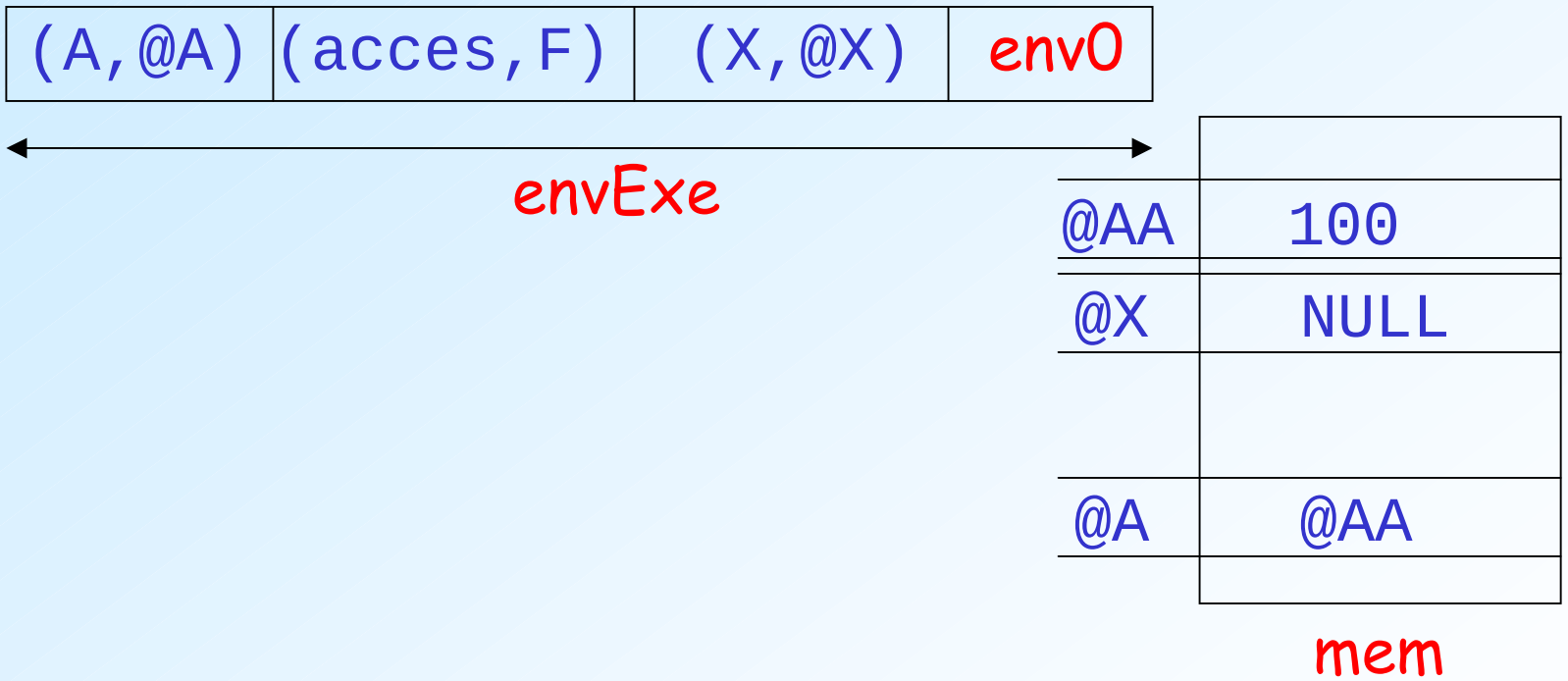
envExe

@X	NULL
@A	??

mem

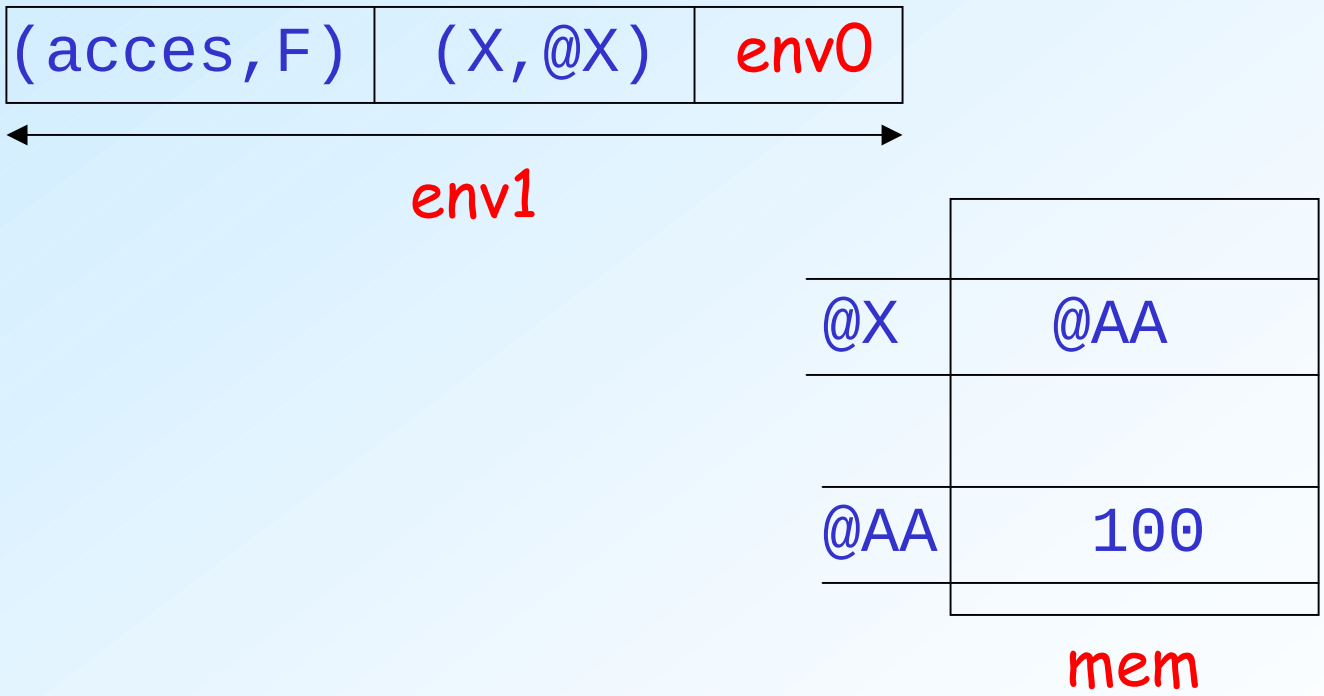
F=<<A→Corps de acces, env1>>

Etat 3 (avant transmission du paramètre en retour)



F=<<A→Corps de acces, env1>>

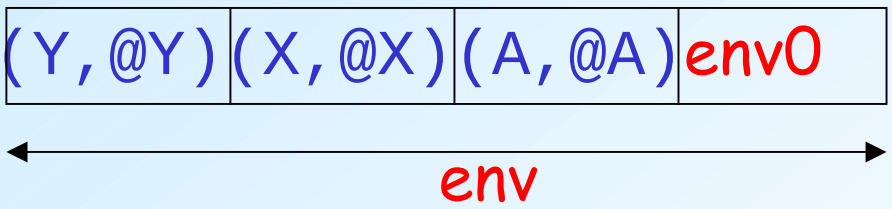
Etat 4 (après retour vers l'appelant)

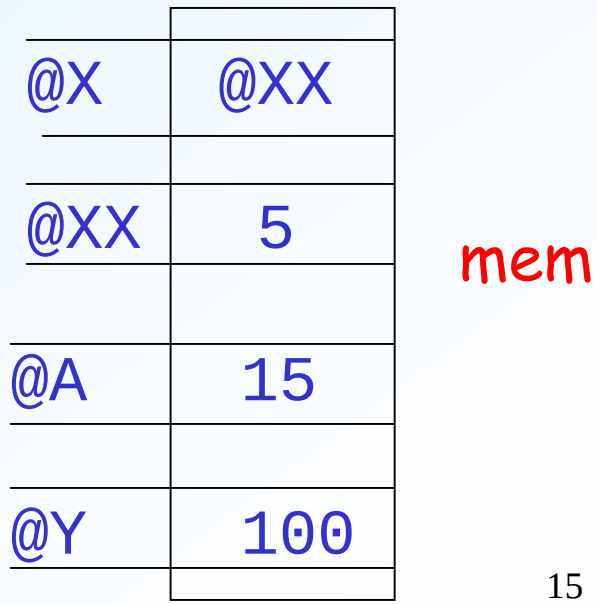


F=<<A→Corps de acces, env1>>

Pointeur constant

```
A:Integer:= 15;
X:Ref_int:= new Integer'(5);
Y:constant Ref_int:= new Integer'(100);
```





```
with Ada.Integer_Text_io;use Ada.Integer_Text_io;
with Ada.Text_io;use Ada.Text_io;
procedure acces1 is
   type Ref_int is access Integer;
   A:Integer:= 15;
   X:Ref_int:= new Integer'(A);
   Y:constant Ref_int:= new Integer'(100);
   Z:constant Ref_int:= X;
begin
   put("X=");put(X.all);new_line;-- affiche 15
   put("Y=");put(Y.all);new_line;-- affiche 100
   put("Z=");put(Z.all);new_line;-- affiche 15
   X:= new Integer'(50);
  put("X=");put(X.all);new_line;-- affiche 50
  put("Z=");put(Z.all);new_line;-- affiche 15
end acces1;
```

```
with Ada.Integer_Text_io;use Ada.Integer_Text_io;
with Ada.Text_io;use Ada.Text_io;
procedure acces2 is
   type Ref_int is access Integer;
   A:Integer:= 15;
   X:Ref_int:= new Integer'(A);
   Y:constant Ref_int:= new Integer'(100);
   Z:constant Ref_int:= X;
begin
   put("X=");put(X.all);new_line;-- affiche 15
   put("Y=");put(Y.all);new_line;-- affiche 100
   put("Z=");put(Z.all);new_line;-- affiche 15
   X.all:= 50;
   put("X=");put(X.all);new_line;-- affiche 50
   put("Z=");put(Z.all);new_line;-- affiche 50
end acces2;
```

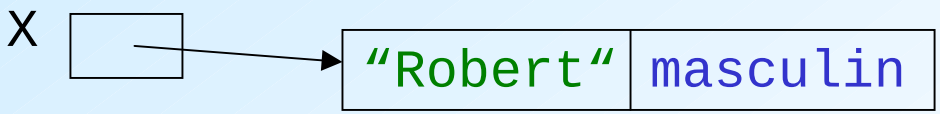
```
with Ada.Integer_Text_io;use Ada.Integer_Text_io;
with Ada.Text_io;use Ada.Text_io;
procedure acces3 is
   type Ref_int is access Integer;
   A:Integer:= 15;
   X:Ref_int:= new Integer'(A);
   Y:constant Ref_int:= new Integer'(100);
   Z:constant Ref_int:= X;
begin
   Y:=X; -- affectation à la constante Y=>interdit
   X:=Y; -- objet référencé par X devient inaccessible
   X:=A; -- incompatibilité de types
   Y.all:=A;
end acces3;
```

Exemple (1/2)

```
declare
 type Genre is (masculin,feminin);
 type Invite is
  record
   nom :String(1..6);
   sexe:Genre;
  end record;
 type Reveillon is access Invite;
 X,Y:Reveillon;
 I:Invite;
```

Exemple (2/2)

```
begin
 X:= new Invite'(nom=>"Robert",sexe=>masculin);
```




```
 I:= X.all;
 I.nom:= X.all.nom; -- ou X.nom
 Y:= X; -- pointent-ils sur le même invité ?
 -- Y.all et X.all sont ils égaux ?
 Y.nom:= "Marcel";
 -- X.nom et Y.nom sont ils égaux ?
 Y:= new Invite'( "Denise" ,feminin);
 end;
```