Pointeurs

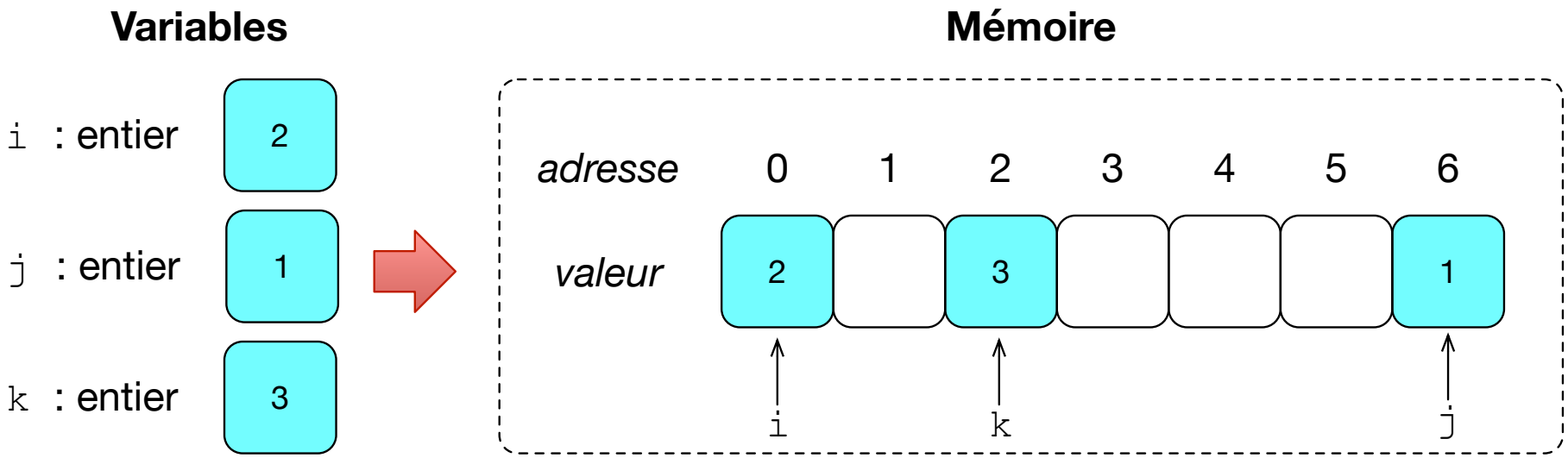
Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 14 octobre 2025

IUT Valence
Université Grenoble Alpes

Variables et mémoire



- Une variable occupe un **emplacement mémoire**, désigne cet emplacement dans le cas d'une affectation (écriture) et la valeur qu'il contient dans le cas d'une expression (lecture)
  - N.B. : *dans la suite, pour faire simple, on schématise une mémoire dont les cases peuvent indifféremment contenir n'importe quel type de données (entiers, booléens, flottants, ...)*
  - cf. R1.03 pour une vision plus fidèle à l'architecture matérielle

IUT Valence
Université Grenoble Alpes

Passage de paramètres

- On considère les 2 fonctions suivantes, et on suppose que l'on appelle la fonction `principale`

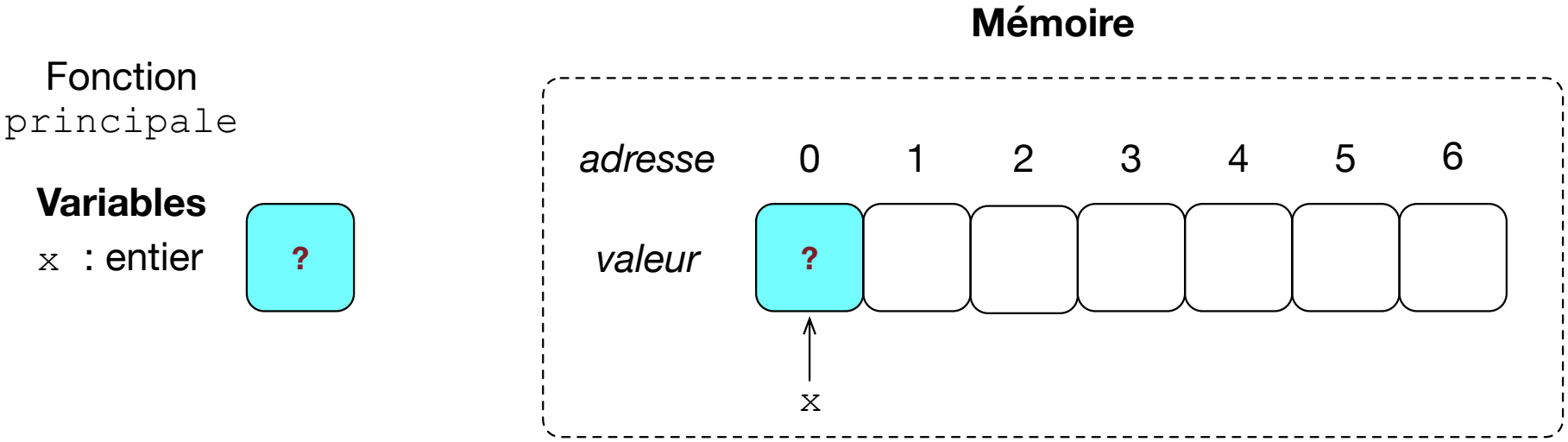
```
FONCTION autre_fonction(n : entier) : entier
 n ← n + 1
 RETOURNER n
FIN FONCTION
```

```
FONCTION principale() : (aucun)
 VARIABLE x : entier

 x ← 10
 afficher_entier(autre_fonction(x))
 afficher_entier(x)
FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Allocation de variable

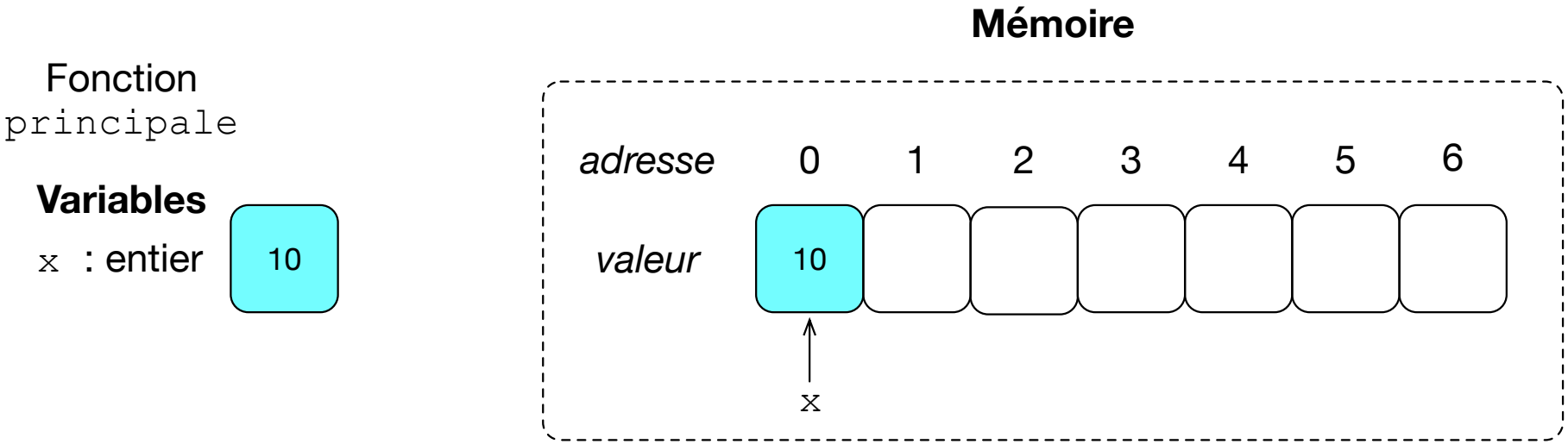
- A l'appel d'une fonction, les **variables qu'elle déclare sont allouées en mémoire** (les emplacements sont réservés)



- Les variables ne sont pas initialisées, les emplacements **ne doivent pas être lus**

Initialisation / affectation de variable

```
x ← 10
```

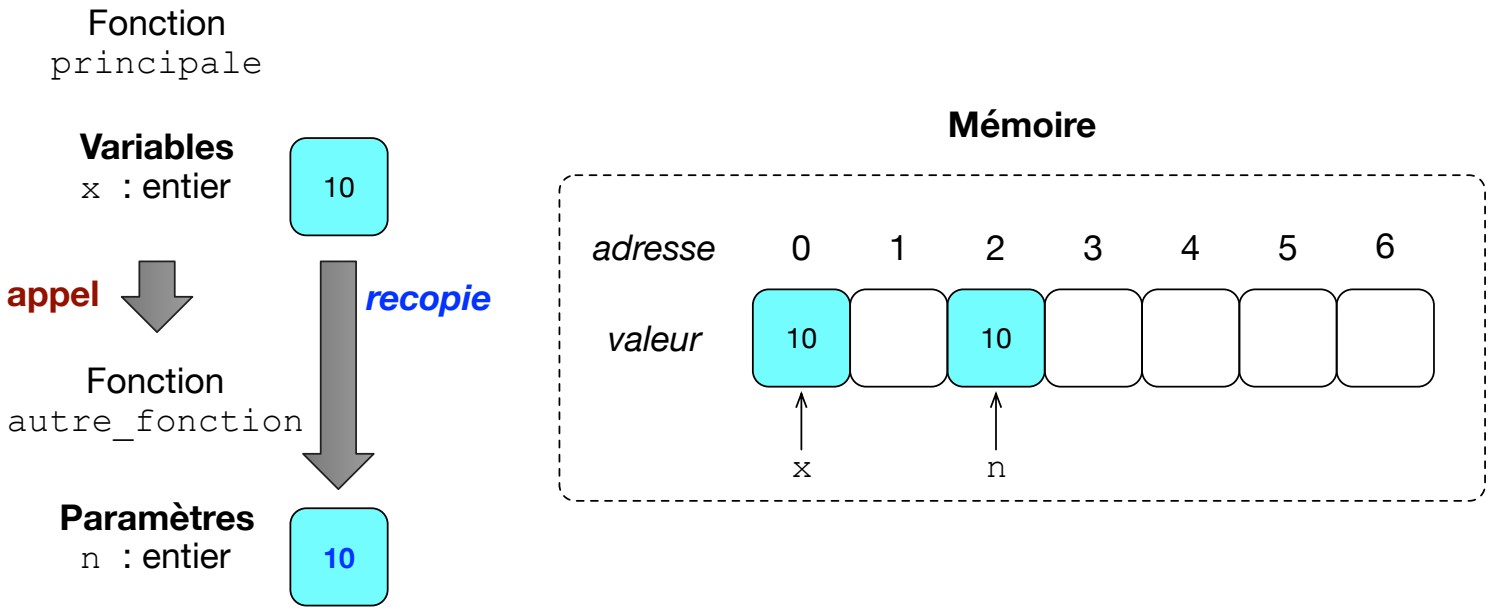


- L'affectation d'une valeur à une variable **modifie le contenu de l'emplacement désigné par la variable**
- La première affectation (initialisation) rend valide la lecture

IUT Valence Université Grenoble Alpes

Passage de paramètres

```
afficher_entier(autre_fonction( x ) )
```



- Dans notre *pseudo code*, on considère que les entiers, flottants et booléens sont **passés par valeur**
  - N.B. : On parle aussi de passage par copie car la **valeur passée en paramètre** par la fonction appelante est **recopiée dans un autre emplacement mémoire** qui pourra être désigné par le nom du paramètre dans la fonction appelée

IUT Valence Université Grenoble Alpes

Passage de paramètres

```
n ← n + 1
```

Fonction
`principale`

Variables

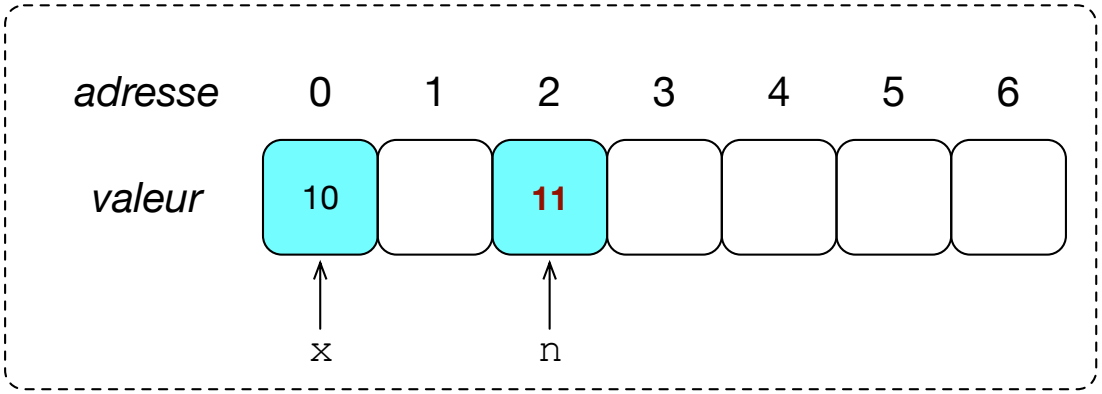
`x` : entier — 10

Fonction
`autre_fonction`

Paramètres

`n` : entier — 11

Mémoire

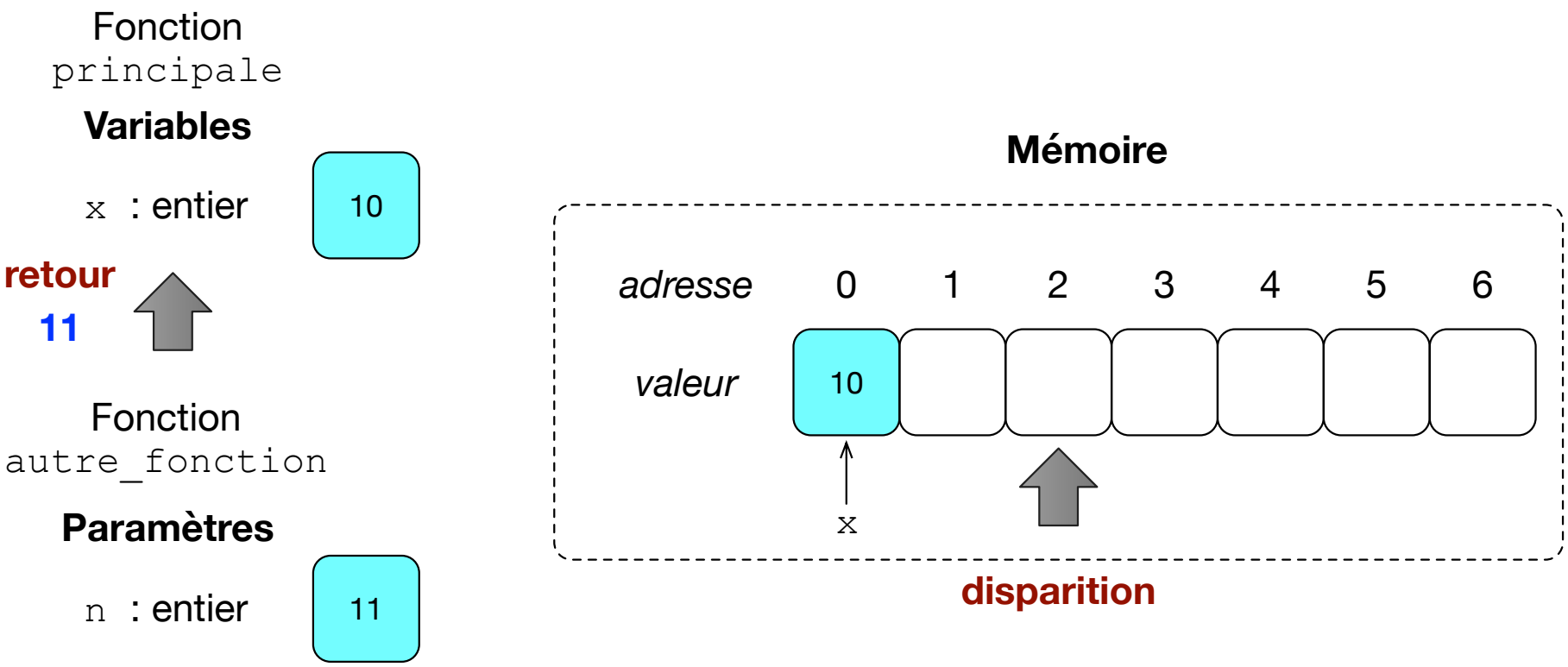


- La modification de la valeur du paramètre dans la fonction **s'effectue dans l'emplacement mémoire du paramètre** et non dans celui de la variable dont la valeur a été passée en paramètre

IUT Valence Université Grenoble Alpes

Passage de paramètres

```
RETOURNER n
```



- A la sortie d'une fonction, les **emplacements mémoire occupés par les variables et paramètres sont désalloués**
- N.B. : les emplacements désalloués ne doivent plus être lus

IUT Valence Université Grenoble Alpes

Passage de paramètres

- Dans l'exemple, l'exécution de la fonction `principale` produit donc l'affichage de 11 et 10

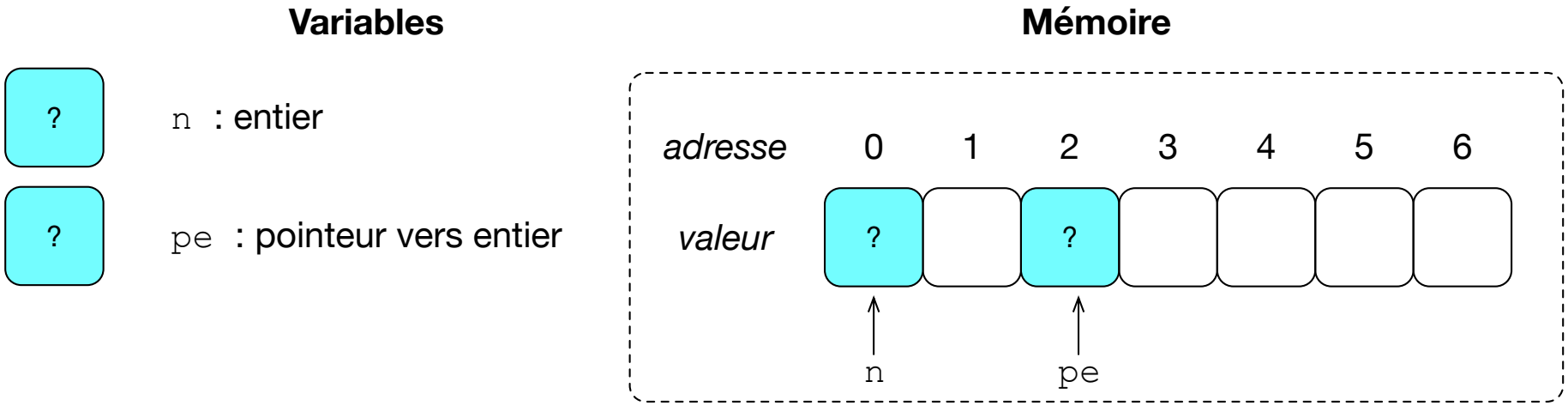
```
FONCTION autre_fonction(n : entier) : entier
 n ← n + 1
 RETOURNER n
FIN FONCTION

FONCTION principale() : (aucun)
 VARIABLE x : entier

 x ← 10
 afficher_entier(autre_fonction(x))
 afficher_entier(x)
FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Variable de type pointeur

- Une **variable de type *pointeur vers un type T*** est une **variable dont la valeur est l'adresse d'un emplacement mémoire** où est stockée une valeur de type T

```
VARIABLE pe : pointeur vers entier
```



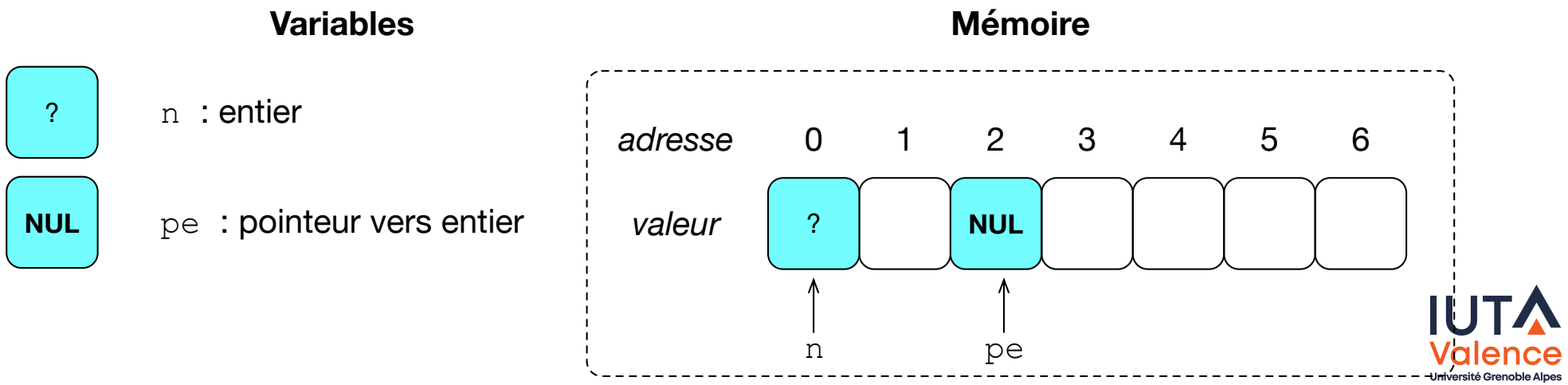
IUT Valence Université Grenoble Alpes

Initialisation d'une variable de type pointeur

- Une variable de type *pointeur vers un type T* ne doit pas être lue si elle n'est pas initialisée
- On peut **initialiser une variable de type pointeur** avec la valeur `NUL` pour indiquer qu'elle ne désigne pas encore d'emplacement
  - Elle peut alors être lue, on saura qu'elle ne désigne pas encore d'emplacement

---

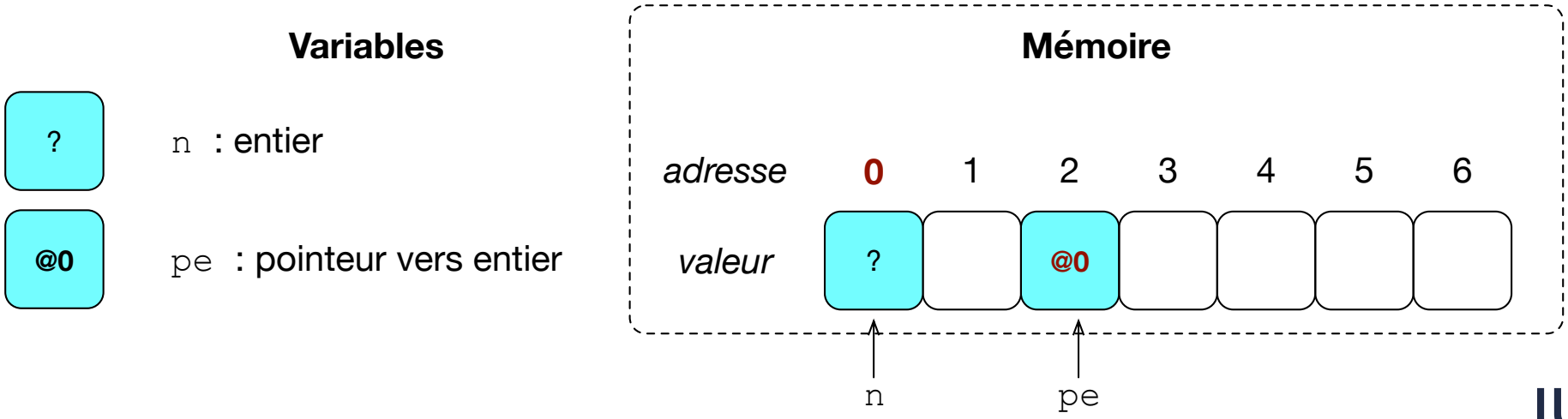
`pe` ← `NUL`



Initialisation d'une variable de type pointeur

- On ne peut pas « fabriquer d'adresse », **on ne peut affecter comme valeur à une variable de type pointeur que l'adresse d'une variable (ou d'un paramètre) du type attendu**
- l'opérateur & désigne l'**adresse d'une variable**

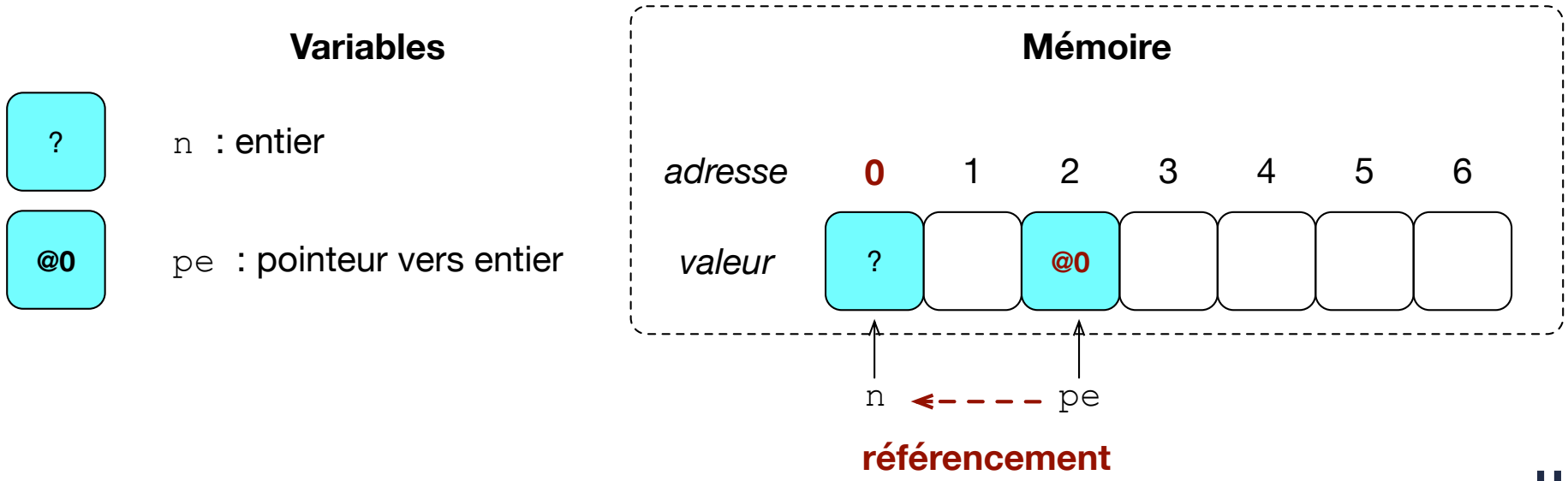
```
pe ← &n
```



Initialisation d'une variable de type pointeur

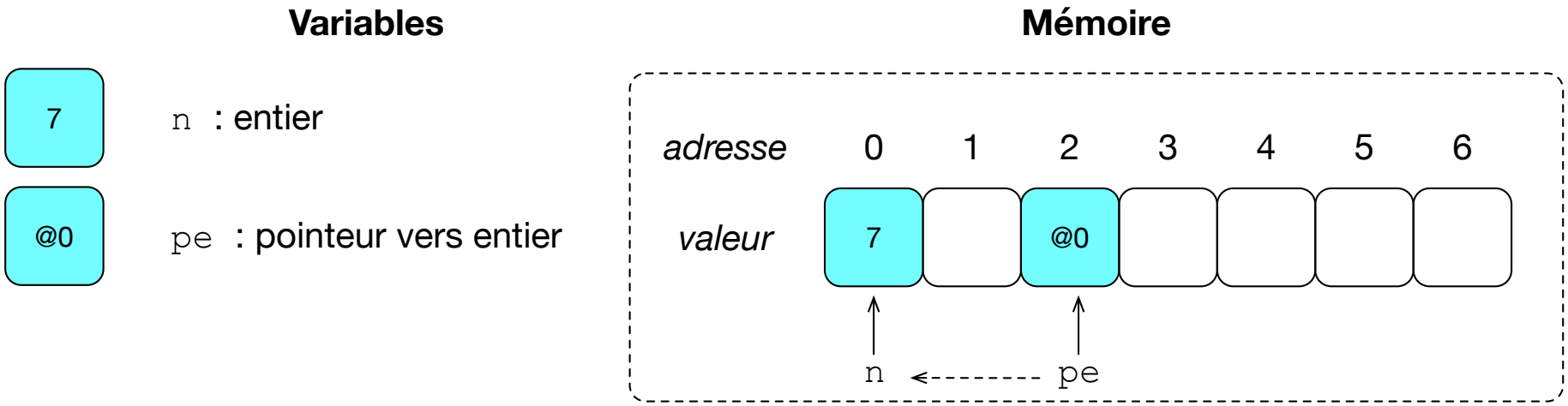
- La valeur d'une variable de type pointeur vers le type `T` est l'**adresse de l'emplacement mémoire d'une valeur de type `T`**
  - On dit que le pointeur **référence** la variable

```
pe ← &n
```



Déréferencement d'une variable de type pointeur

- L'opérateur ↑ exprime le **déréférencement**, c'est à dire la **désignation de l'emplacement mémoire dont l'adresse est contenue dans le pointeur**



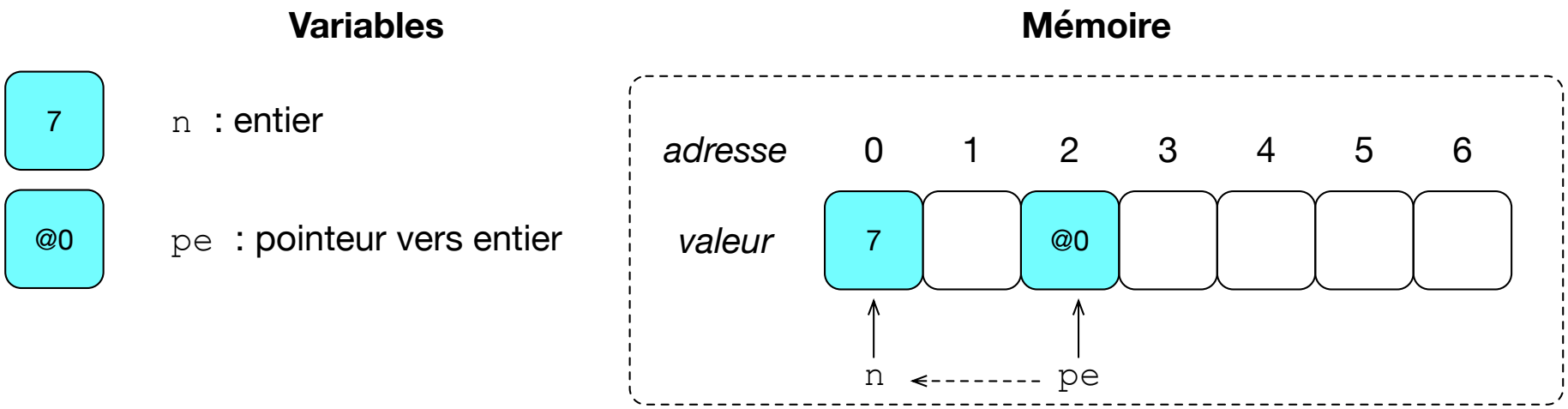
- Un pointeur dont la valeur est `NUL` ou dont la valeur désigne un emplacement non initialisé ne doit pas être déréférencé

IUT Valence Université Grenoble Alpes

Déréférencement d'une variable de type pointeur

- L'opérateur ↑ peut être utilisé dans une expression
  - Dans ce cas l'**expression vaut la valeur contenue dans l'emplacement dont l'adresse est contenue dans le pointeur**

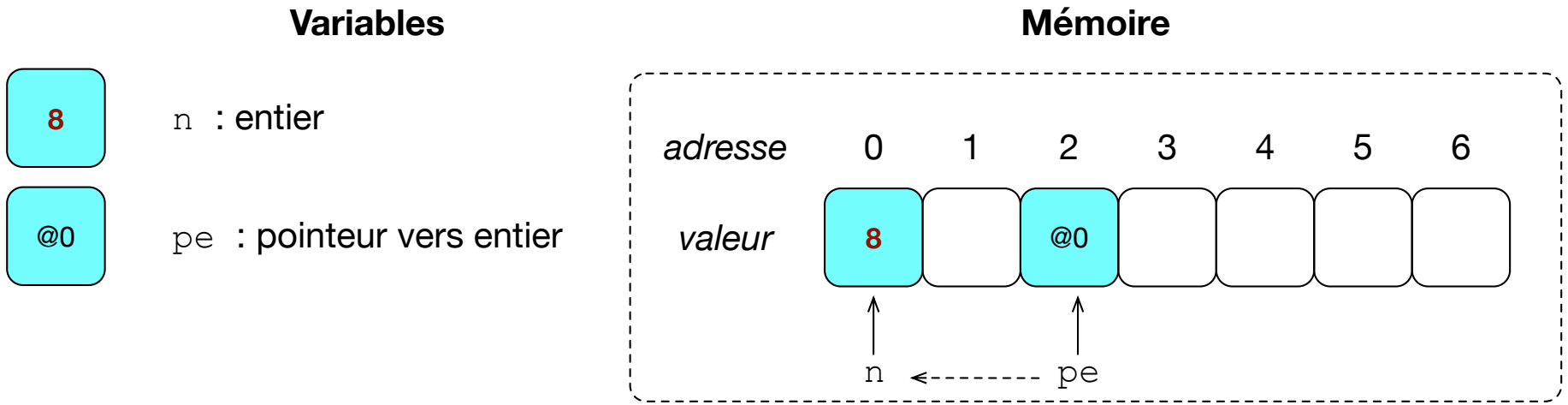
```
afficher_entier(pe↑)                    // affiche 7
```



Déréférencement d'une variable de type pointeur

- L'opérateur ↑ peut être utilisé dans une affectation
  - Dans ce cas l'**emplacement dont l'adresse est contenue dans le pointeur est réaffecté avec la nouvelle valeur**

`pe↑ ← 8`



`pe↑ ← pe↑ +1` `// ?`

IUT Valence Université Grenoble Alpes

Passage de pointeur en paramètre d'une fonction

- Les pointeurs sont passés en paramètre **par valeur**
  - La réaffectation (dans la fonction) de la valeur d'un pointeur passé en paramètre n'a **pas d'effet au retour** sur la valeur que l'appelant a passé en paramètre
  - La réaffectation (dans la fonction) de la valeur de l'emplacement mémoire dont l'adresse est contenue dans un pointeur passé en paramètre **persiste lors du retour**

```
FONCTION ma_fonction(n : entier,
                     pe : pointeur vers entier) :
                         booléen
  ...
FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Passage de pointeur en paramètre d'une fonction

```
FONCTION ma_fonction(n : entier,
                     pe : pointeur vers entier)
                     : booléen
 VARIABLE i : entier

 i ← n - 1

 SI n > 42 ALORS
  pe↑ ← pe↑ +1
  RETOURNER VRAI
 FIN SI

 pe ← &i
 pe↑ ← pe↑ −1
 RETOURNER FAUX

FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Passage de pointeur en paramètre d'une fonction

```
FONCTION principale(x : entier) : (aucun)

 VARIABLE px : pointeur vers entier

 afficher_entier(x)
 px ← &x

 afficher_booleen(ma_fonction(x, px))
 afficher_entier(x)

FIN FONCTION
```

- Quel est l'affichage de `principale(43)` ?
- Quel est l'affichage de `principale(41)` ?

IUT Valence
Université Grenoble Alpes

Exercice

Enoncé du problème

On veut permuter les valeurs de 2 entiers.

Spécification du problème

Signature de la fonction

IUT Valence
Université Grenoble Alpes

Exercice 2

Spécification du problème

- **Donnée d'entrée** : *pe1*, pointeur vers un entier (premier)
- **Donnée d'entrée** : *pe2*, pointeur vers un entier (second)
- **Donnée de sortie** : (aucune)
- **Pré-condition** : $pe1 \neq NUL$ ET $pe2 \neq NUL$
- **Post-condition** : les 2 valeurs sont permutées.

Signature de la fonction

- **permute** (pe1 : pointeur vers entier, pe2 : pointeur vers entier) : (aucun)

IUT Valence
Université Grenoble Alpes

Exercice

```
FONCTION permute (pe1 : pointeur vers entier,
                  pe2 : pointeur vers entier) : (aucun)

  VARIABLE temp : entier

  temp ← pe1↑
  pe1↑ ← pe2↑
  pe2↑ ← temp

FIN FONCTION
```

Au final, ça sert à quoi une variable de type pointeur ?

- Manipuler des données que l'**on ne sait pas désigner directement**
- Passer des paramètres qui servent aussi à **retourner des résultats** (paramètres *in Vs out*)
- Pouvoir gérer dynamiquement la mémoire

IUT Valence
Université Grenoble Alpes

Fin !

IUT Valence
Université Grenoble Alpes