

# Module 3

# Tableaux

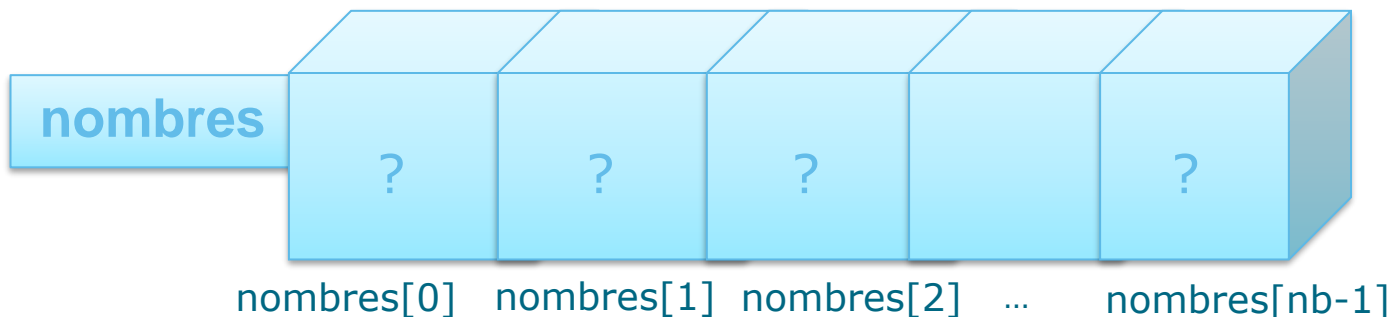
*DA & IA – Bloc 1*

# Plan du module

- Qu'est-ce qu'un tableau?
- Cellule d'un tableau
- Tableau à simple indice
  - Manipulation d'un tableau
  - Utilité d'un tableau
- Tableau à double indice

# Qu'est-ce qu'un tableau?

- Un tableau est un ensemble de valeurs de **même type**, portant le **même nom de variable**.
- Chaque valeur est repérée par un nombre.
- Le nombre qui, au sein d'un tableau, sert à repérer chaque valeur s'appelle l'**indice**.
- Un tableau comprenant **nb** éléments verra ainsi son indice aller de **0** à **nb-1**.



# Qu'est-ce qu'un tableau?

## Structure de données (ou collection)

- **homogène**
  - tous les éléments sont de même type
- **séquentielle (linéaire)**
  - tous les éléments sont consécutifs en mémoire
- **indicée**
  - accès direct à un élément via un indice
- de **taille fixe**

# Cellule d'un tableau

Les cellules d'un tableau peuvent contenir des valeurs :

- de type **simple** : nombres entiers/réels ou caractères
- de type **complexe** : des regroupements de données dans une structure  
= tableau de structures
- de type **tableau**

# Cellule d'un tableau

## Exemples:

- **nombres** { cellule (nb\*) } (chaque cellule contient un nombre)
- **achats** { cellule (nbAchats\*) } { date  
nomClient  
montant }
- **étudiants** { cellule (100\*) } { nom  
section  
année  
**cotes** { cellule (5\*) } { intituléCours  
points }

# Tableau à simple indice

- La forme de tableau la plus simple est le tableau à une dimension (ou simple indice) appelé aussi **vecteur**.
- On accède à chaque élément du tableau (cellule) via son **indice**. Plus précisément le nom du tableau, suivi de l'indice de la cellule entre crochets : **nombres [indice]**.
- L'espace mémoire est alloué de façon **statique** car la longueur du tableau est fixe.

# Manipulation d'un tableau

## Exemple 1

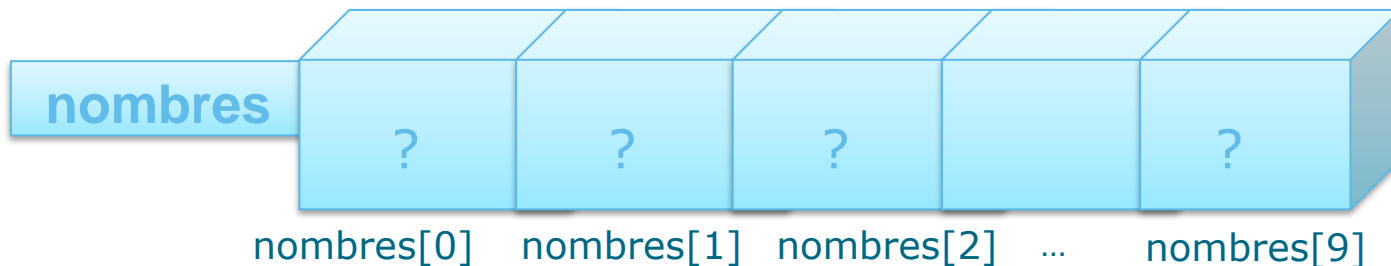
### Remplir un tableau avec des nombres obtenus au clavier

On demande à l'utilisateur d'introduire 10 nombres au clavier et de remplir un tableau avec ces 10 nombres.

IN : 10 nombres                      OUT : /

Tableau créé :

**nombres** { cellule (chacune contient un nombre)  
(10 \*)



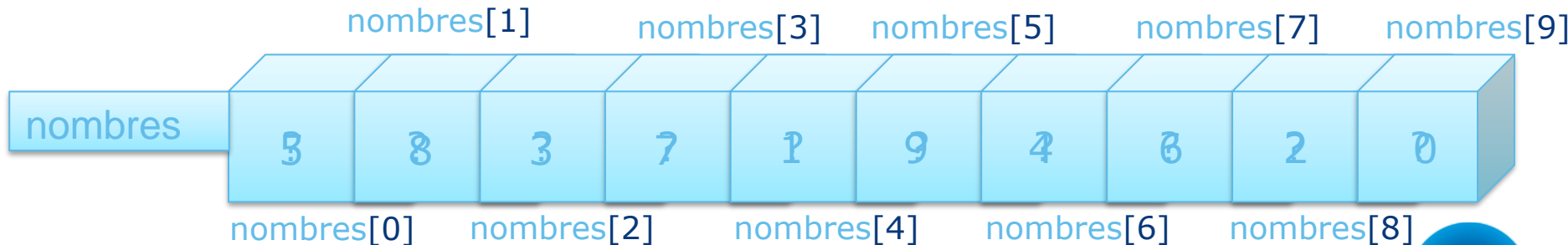


# Manipulation d'un tableau

0 ————— 0  
| tableauGarni |  
0 ————— 0 ↓ nombres

```
*  
// nombres = ARRAY[10]  
iNombre = 0  
while (iNombre < 10)  
  obtenir nombres[iNombre]  
  iNombre ++
```

## ■ Représentation graphique :



# Manipulation d'un tableau

## Exemple 2

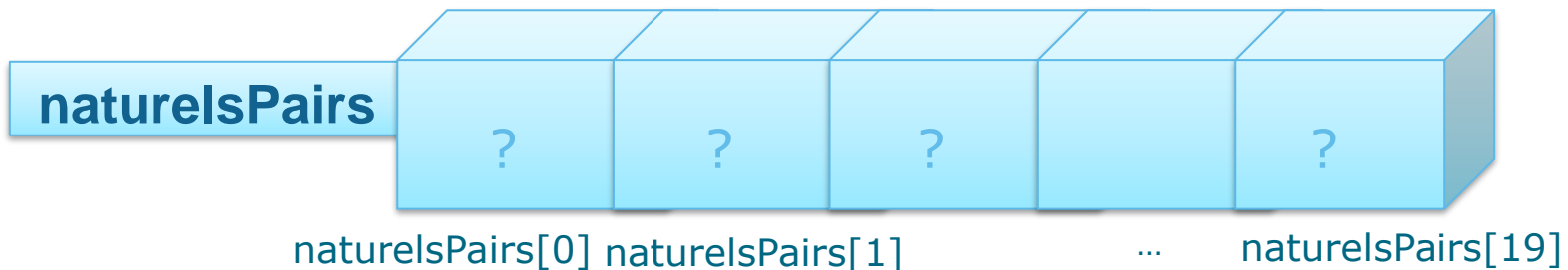
Garnir un tableau de 20 cellules avec les naturels pairs successifs en commençant par une valeur reçue en paramètre

*Pré condition: le nombre reçu en paramètre est un naturel pair*

IN: /    OUT: /

Tableau créé :

**naturelsPairs** { cellule (chacune contient un naturel pair)  
(20 \*)



# Manipulation d'un tableau

o ————— o ↓ nombrePair  
| naturelsPairsGarni |  
o ————— o ↓ naturelsPairs

\*

```
// naturelsPairs = ARRAY[20]
iNombre = 0
while (iNombre < 20 )
    naturelsPairs[iNombre] = nombrePair
    nombrePair += 2
    iNombre ++
```

# Manipulation d'un tableau

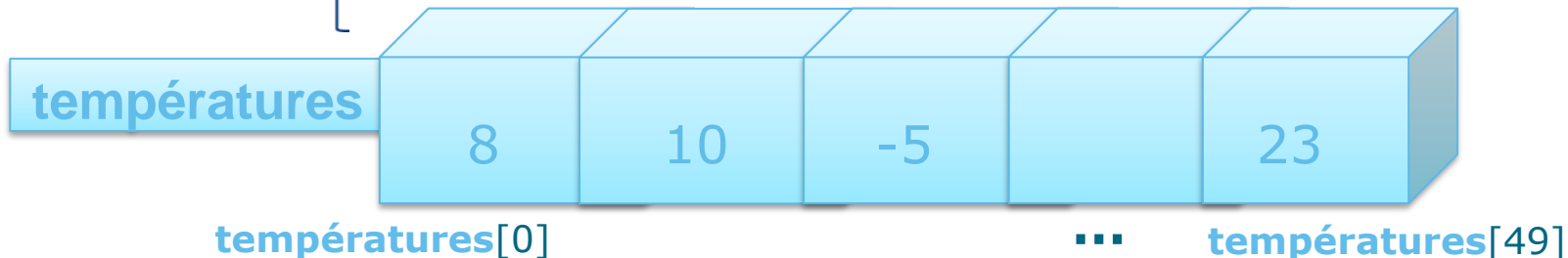
## Exemple 3

### Afficher le contenu d'un tableau dans l'ordre inverse des cellules

Ecrire le module qui reçoit en paramètre un tableau **températures** de 50 cellules entièrement garni. On demande d'afficher son contenu de la dernière à la première cellule.

#### Description du tableau

**températures** { cellule (chacune contient une température)  
(50 \*)



# Manipulation d'un tableau

o ————— o ↓ températures  
| afficherOrdreInverse |  
o ————— o

```
    *  
    iTemp = 49  
    while(iTemp ≥ 0)  
        sortir temperatures[iTemp]  
        iTemp --
```

# Manipulation d'un tableau

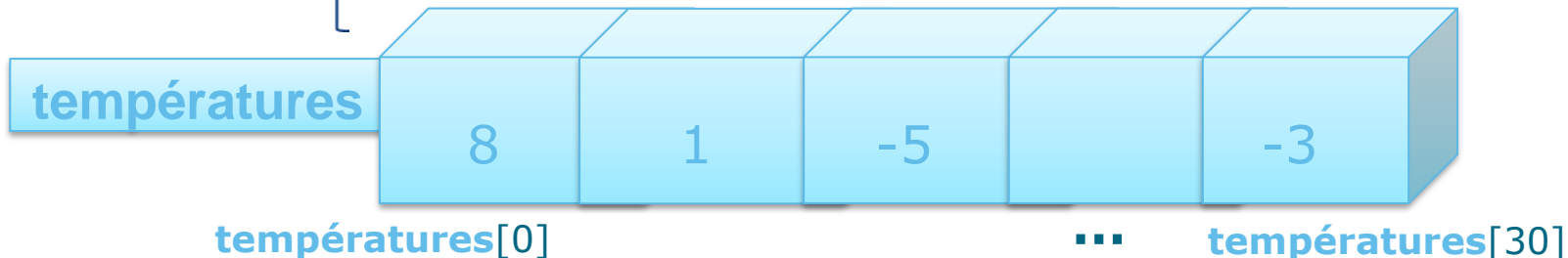
## Exemple 4

### Rechercher un élément dans un tableau.

Le tableau **températures** contient, dans l'ordre chronologique, les températures relevées chaque jour de janvier 2023 à 8h30 à Namur. Ecrire le module qui **reçoit** ce tableau en paramètre et qui **renvoie** le numéro du premier jour du mois où on a relevé une température négative.

### Description du tableau

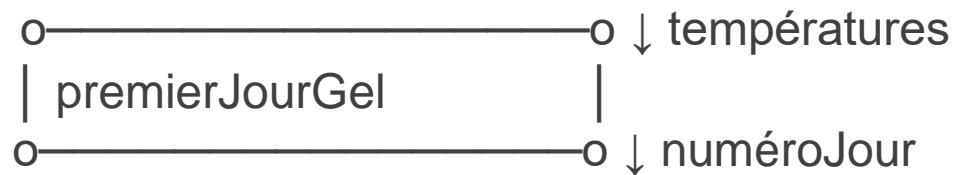
**températures** { cellule (chacune contient une température)  
(31 \*)



# Manipulation d'un tableau

## Post condition

Le module renvoie un numéro de jour égal à -1 s'il n'y a aucune température négative



Obligatoirement la première condition

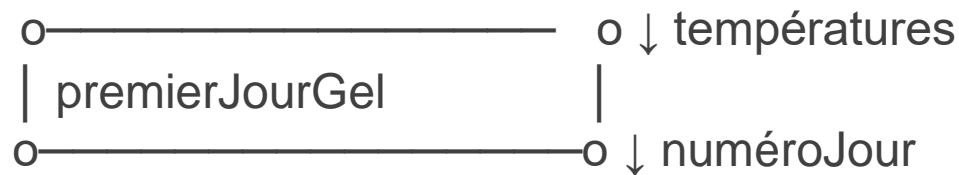
```
*
iJour = 0
while (iJour < 31 and temperatures[iJour] ≥ 0)
    iJour ++
numéroJour = iJour + 1
```

Que se passe-t-il s'il n'a jamais gelé pendant le mois de janvier?

# Manipulation d'un tableau

## Post condition

Le module renvoie un numéro de jour égal à -1 s'il n'y a aucune température négative



Obligatoirement la première condition

```
*  
iJour = 0  
while (iJour < 31 and températures[iJour] ≥ 0)  
    iJour ++  
  
if (iJour == 31)  
    numéroJour = -1  
else  
    numéroJour = iJour + 1
```

Que se passe-t-il s'il n'a jamais gelé pendant le mois de janvier?



# Utilité d'un tableau

- Le tableau permet de rassembler plusieurs variables de même type en une seule. Chacune sera désignée au sein du tableau par un numéro.
- On aura ainsi recours à un tableau lorsque, dans un programme, on a besoin de **conserver simultanément en mémoire** un certain nombre de valeurs de même type **de manière à les utiliser ultérieurement**.

Ne pas utiliser de tableau si ce n'est pas nécessaire!  
Risque d'augmenter inutilement la complexité de l'algorithme.

# Utilité d'un tableau

## Exercice 1 (résolu en cours)

On souhaite écrire le DA qui , à partir de 10 cotes (différentes) obtenues de l'utilisateur au clavier, calcule la cote moyenne pour ensuite afficher la cote qui se rapproche le plus de cette moyenne.

**IN** : 10 cotes

**OUT**: cote plus proche de la moyenne

**Tableau créé :**

**cotes**  $\left[ \begin{array}{l} \text{cellule} \\ (10^*) \end{array} \right]$  chaque cellule contient une cote

# Exercice 1 (solution)

```
0-----0
| afficherCoteProche |
0-----0

*
// cotes = ARRAY[10]
moyenne = 0
iCote = 0
while (iCote < 10)
  obtenir cotes[ iCote]
  moyenne += cote
  iCote ++

moyenne /= 10
écartMin = 21
iCote = 0
while (iCote < 10)
  écart = moyenne - cotes[iCote]
  if( écart < 0)
    écart = -écart
  if( écart < écartMin)
    écartMin = écart
    coteProche = cotes[iCote]
  iCote ++

sortir coteProche
```

# Exercice 1 (solution avec des modules)

```
0-----0
| afficherCoteProche |
0-----0

*
0-----0
| tableauGarniMoyenne |
0-----0 ↓ cotes, moyenne
0-----0 ↓ cotes, moyenne
| cotePlusProche |
0-----0 ↓ cotePlusProche
sortir cotePlusProche
```

```
0-----0
| tableauGarniMoyenne |
0-----0 ↓ cotes, moyenne

*
// cotes = ARRAY[10]
moyenne = 0
iCote = 0
while (iCote < 10)
  obtenir cotes[ iCote]
  moyenne += cote
  iCote ++
moyenne /= 10
```

```
0-----0 ↓ cotes, moyenne
| cotePlusProche |
0-----0 ↓ coteProche

*
écartMin = 21
iCote = 0
while (iCote < 10)
  écart = moyenne - cotes[iCote]
  if( écart < 0)
    écart = -écart
  if( écart < écartMin)
    écartMin = écart
    coteProche = cotes[iCote]
  iCote ++
```

# Utilité d'un tableau

## Exercice 2 (variante de l'ex 1) (résolu en cours)

On souhaite écrire le DA qui , à partir de cotes (différentes) obtenues de l'utilisateur au clavier( -1 pour terminer), permet de calculer la cote moyenne pour ensuite afficher la cote qui se rapproche le plus de cette moyenne.

**IN :** cote ( ou -1)  
( ? \*)

**OUT:** cote plus proche de la moyenne

**Tableau créé :**

**cotes** { cellule { (chaque cellule contient une cote)  
(**tailleMax**\*) ← Taille physique du tableau  
(**nbCotes**\*) ← Taille logique du tableau

# Exercice 2 (solution)

```
0-----0
| afficherCoteProche |
0-----0

*
// cotes = ARRAY[tailleMax]
nbCotes = 0
moyenne = 0
obtenir cote
while (cote ≠ -1)
  cotes[ nbCotes ] = cote
  moyenne += cote
  nbCotes ++
  obtenir cote

if(nbCotes ≠ 0)
  moyenne /= nbCotes

écartMin = 21
iCote = 0
while (iCote < nbCotes)
  écart = moyenne - cotes[iCote]
  if( écart < 0)
    écart = -écart

  if( écart < écartMin)
    écartMin = écart
    coteProche = cotes[iCote]

  iCote ++

sortir coteProche
```

# Tableau de structures: introduction

## Exemple 5

Le tableau **températures** contient 100 relevés de température effectués en 2023 à 8h30 à Namur. Ecrire le module qui **reçoit** ce tableau en paramètre et qui **renvoie** la date du jour (MMJJ) où on a relevé la plus haute température.

Description du tableau

**températures**  $\left\{ \begin{array}{l} \text{cellule} \\ (100 *) \end{array} \right\}$  (chacune contient une température)

On a également besoin de mémoriser les 100 dates => 2<sup>e</sup> tableau qui contient les 100 dates ???

Nécessité de gérer deux tableaux : source de complication et d'erreurs !

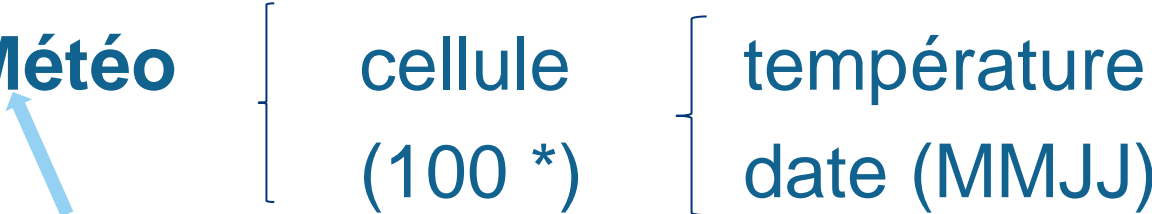
**Solution:** un seul tableau dont chaque cellule est composée de 2 champs

# Tableau de structures

Le contenu d'une cellule est composé de plusieurs champs.

Exemple

**infosMétéo** { cellule { température  
(100 \*) date (MMJJ)



Respecter le clean code dans le nom du tableau

L'accès à un champ d'une cellule se fera comme dans l'exemple qui suit:

`infosMétéo[5].température`

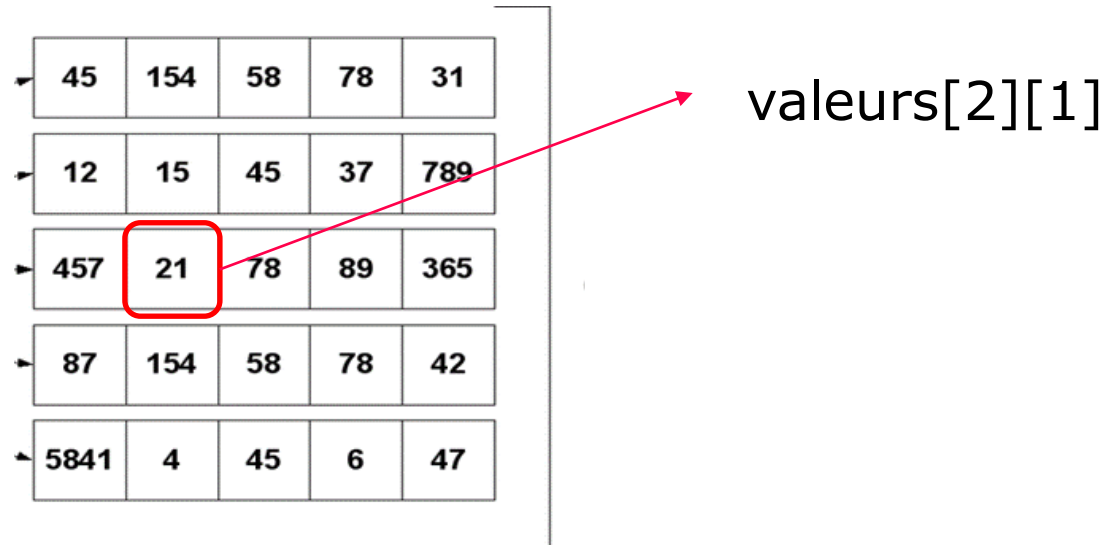
`infosMétéo[35].date`

Une cellule n'est plus de **type simple** mais de type complexe, on parle de **STRUCTURE**



# Tableau à double indice

Un tableau à double indice est un tableau à deux dimensions (aussi appelé matrice).



→ 45	154	58	78	31
→ 12	15	45	37	789
→ 457	21	78	89	365
→ 87	154	58	78	42
→ 5841	4	45	6	47

valeurs[2][1]