



مدرسة علوم المعلومات  
+٤1٢٨ | +٢٥.٥٥.٥٤ | ٤١٢٤٤٥١  
ECOLE DES SCIENCES  
DE L'INFORMATION  
[www.esi.ac.ma](http://www.esi.ac.ma)

# Algorithmique

## 5. Les tableaux

# Sommaire

---

- I. Tableaux statiques
- II. Tableaux dynamiques
- III. Tri dans un tableau
- IV. Recherche dans un tableau

**Rappel**

# Rappel

---

- ▶ Algorithme informatique

- ▶ Suite d'instructions ordonnée qui décrit de façon exhaustive les différentes étapes à suivre par un processeur pour résoudre un problème donné en un temps fini

# Rappel

---

## ► Pseudo-code algorithmique

**ALGORITHME** *nom\_de\_l'algorithme*

<partie des déclarations>

**DEBUT**

<partie des instructions>

**//commentaire**

**FIN**

# Rappel

---

- ▶ Un programme n'est pas purement séquentiel ➔  
nécessité d'avoir des **structures de contrôle**
  1. Les structures alternatives (tests)
  2. Les structures itératives (boucles)

# Rappel

---

- ▶ Un programme n'est pas purement séquentiel →  
nécessité d'avoir des **structures de contrôle**

1. **Les structures alternatives (tests)**
2. **Les structures itératives (boucles)**

# Rappel

---

**ALGORITHME** *nom\_de\_l'algorithme*

<partie des déclarations>

**DEBUT**

*séquence 1*

**SI** *condition 1* **ALORS**

*séquence2*

**FINSI**

*séquence3*

**FIN**



# Rappel

---

**ALGORITHME** *nom\_de\_l'algorithme*

<partie des déclarations>

**DEBUT**

*séquence1*

**SI** *condition1* **ALORS**

*séquence2*

**SINON**

*séquence3*

**FINSI**

*séquence4*

**FIN**

# Rappel

---

- ▶ Un test imbriqué est exprimé comme suit

**SI** *condition 1* **ALORS**

*séquence 1*

**SINON**

**SI** *condition2* **ALORS**

*séquence2*

**SINON**

*séquence3*

**FINSI**

**FINSI**

# Rappel

---

- Pour alléger l'écriture et améliorer la lisibilité, on peut fusionner SINON et SI en **SINONSI** → un seul bloc de test

**SI** *condition 1* **ALORS**

*séquence 1*

**SINONSI** *condition 2* **ALORS**

*séquence 2*

**SINON**

*séquence 3*

**FINSI**

# Rappel

---

## ► Test « SUIVANT ... CAS »

**SUIVANT** *variable* **FAIRE**

**CAS** *valeur\_1* : *sequence\_1*

**CAS** *valeur\_2* : *sequence\_2*

...

**CAS** *valeur\_n* : *sequence\_n*

**AUTRES CAS** : *sequence\_autre*

**FINSUIVANT**

# Rappel

---

## ► Test « SELONQUE ... CAS ... »

### **SELONQUE**

*condition\_1 : sequence\_1*

*condition\_2 : sequence\_2*

...

*condition\_n : sequence\_n*

**SINON:** *sequence\_sinon*

### **FINSELONQUE**

# Rappel

---

- ▶ Un programme n'est pas purement séquentiel ➔  
nécessité d'avoir des **structures de contrôle**
  1. Les structures alternatives (tests)
  2. **Les structures itératives (boucles)**

# Rappel

---

- ▶ Une **boucle** est une structure de contrôle de type itératif (ou répétitif)
- ▶ Elle permet de répéter, plusieurs fois, une instruction ou un ensemble d'instructions
- ▶ La répétition est soumise à une condition

# Rappel

---

- ▶ On utilise trois types de boucles
  - ▶ TANTQUE ... FAIRE
  - ▶ POUR
  - ▶ RÉPÉTER ... JUSQU'À



# Rappel

---

- ▶ Boucle « TANTQUE ... FAIRE »
  - ▶ Permet de répéter une instruction tant qu'une condition est vraie

**TANTQUE** *condition* **FAIRE**

*instructions*

**FIN TANTQUE**

# Rappel

---

- ▶ Boucle « POUR »
  - ▶ Permet de répéter une instruction un nombre déterminé de fois
    - ▶ i.e. le nombre d'itérations est connu à l'avance
  - ▶ Utilise un **compteur** qui est incrémenté après chaque exécution du bloc d'instructions de la boucle

# Rappel

---

## ► Boucle « POUR »

**POUR** *compteur*  $\leftarrow$  *valeur\_initiale* **à** *valeur\_finale* **pas** *valeur\_pas*  
*instructions*  
*compteur* **SUIVANT**

# Rappel

---

- ▶ Boucle « RÉPÉTER ... JUSQU'À »
  - ▶ Permet de répéter une instruction jusqu'à ce que la condition d'arrêt soit vraie

**RÉPÉTER**

*instructions*

**JUSQU'À** *condition\_arrêt*

# Problématique

---

- ▶ Les boucles permettent à l'utilisateur de saisir **n** nombres → on déclare une seule variable mais on écrase sa valeur à chaque nouvelle saisie
- ▶ Si on veut garder les **n** nombres pour les utiliser par la suite → on déclare **n** variables différentes
  - ▶ Quand **n** est petit → Lourd mais gérable
  - ▶ Quand **n** est grand ?

# Problématique

---

- ▶ Les boucles permettent à l'utilisateur de saisir **n** nombres → on déclare une seule variable mais on écrase sa valeur à chaque nouvelle saisie
- ▶ Si on veut garder les **n** nombres pour les utiliser par la suite → on déclare **n** variables différentes
- ▶ Idéalement, tout stocker dans une seule variable → **tableau**

# Tableaux statiques

# Définition

---

- ▶ Tableau statique

- ▶ Séquence de données de **même type**, chacune référencée par un nombre appelé **indice** (ou index)

- ▶ Désigné par

- ▶ Son nom

- ▶ Le type de ses éléments

- ▶ Sa taille (i.e. le nombre de ses éléments)



# Déclaration de tableaux statiques

---

## ► Syntaxe

**VAR** nomTableau[N] :Type

- N : taille du tableau
- Premier indice : 0
- Dernier indice :  $N - 1$

# Déclaration de tableaux statiques

---

## ► Syntaxe

**VAR** nomTableau[N] :Type

## ► Exemple :

**VAR** nombres[5] : entier

**VAR** lettres[26] : caractère

**VAR** mots[100] : chaîne

**VAR** absents[107] : booléen

# Utilisation de tableaux statiques

---

- ▶ Pour désigner un élément du tableau, on utilise son nom avec, entre parenthèses, l'indice de l'élément concerné

- ▶ Exemple :

nombres[0]

nombres[1]

nombres[2]

nombres[3]

nombres[4]

# Utilisation de tableaux statiques

---

- ▶ Pour affecter une valeur à une case du tableau, on utilise son nom avec, entre parenthèses, l'indice concerné puis le signe d'affectation et la valeur

- ▶ Exemple :

nombres[0]  $\leftarrow$  5

nombres[1]  $\leftarrow$  60

nombres[2]  $\leftarrow$  10

nombres[3]  $\leftarrow$  0

nombres[4]  $\leftarrow$  90

# Utilisation de tableaux statiques

---

nombres[0]  $\leftarrow$  5

nombres[1]  $\leftarrow$  60

nombres[2]  $\leftarrow$  10

nombres[3]  $\leftarrow$  0

nombres[4]  $\leftarrow$  90

| 0        | 1         | 2         | 3        | 4         |
|----------|-----------|-----------|----------|-----------|
| <b>5</b> | <b>60</b> | <b>10</b> | <b>0</b> | <b>90</b> |

# Utilisation de tableaux statiques

---

- ▶ L'indice peut être un nombre, une variable ou une expression calculée

- ▶ Exemple

nombres[0]

nombres[i]

nombres[2\*i+5]

# Utilisation de tableaux statiques

---

- ▶ Exemple : demander à l'utilisateur de saisir 10 nombres réels

# Utilisation de tableaux statiques

---

- ▶ Exemple : demander à l'utilisateur de saisir 10 nombres réels

```
ALGORITHME tableau_réels  
VAR nombres[10] : réel  
DEBUT  
    POUR i ← 0 à 10  
        Afficher(" Saisir le nombre numéro ", i+1)  
        Lire(nombres[i])  
    i SUIVANT  
FIN
```



# Exercice

---

- ▶ Que fait cet algorithme ?

```
ALGORITHME exemple_tableau  
VAR tableau[10] , i : entier  
DEBUT  
    POUR i  $\leftarrow$  0 à 10  
        tableau[i]  $\leftarrow$  i  
    i SUIVANT  
    POUR i  $\leftarrow$  0 à 10  
        Afficher(tableau[i])  
    i SUIVANT  
FIN
```

# Exercice

---

- ▶ Que fait cet algorithme ?

**ALGORITHME** *exemple\_tableau*

**VAR** tableau[10] , i, j : **entier**

**DEBUT**

tableau[0]  $\leftarrow$  0

**POUR** i  $\leftarrow$  1 à 10

tableau[i]  $\leftarrow$  tableau[i - 1] + i

i **SUIVANT**

**POUR** i  $\leftarrow$  0 à 10

Afficher(tableau[i])

i **SUIVANT**

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir 20 nombres et puis qui les affiche

# Exercice

---

**ALGORITHME** *moyenne*

**VAR** nombres[20] : **réel**

**DEBUT**

somme  $\leftarrow$  0

**POUR**  $i \leftarrow 0$  à 20

Afficher("Saisir le nombre numéro ",  $i+1$ )

Lire(notes[ $i$ ])

$i$  **SUIVANT**

**POUR**  $i \leftarrow 0$  à 20

Afficher("Le ",  $i$ , "ème nombre est : ", nombres[ $i$ ])

$i$  **SUIVANT**

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir 20 nombres et puis qui les affiche à partir du dernier saisi

# Exercice

---

**ALGORITHME** *moyenne*

**VAR** nombres[20] : **réel**

**DEBUT**

somme  $\leftarrow$  0

**POUR**  $i \leftarrow 0$  à 20

Afficher("Saisir le nombre numéro ",  $i+1$ )

Lire(notes[ $i$ ])

$i$  **SUIVANT**

**POUR**  $i \leftarrow 20$  à  $-1$  pas  $-1$

Afficher("Le ",  $i$ , "ème nombre est : ", nombres[ $i - 1$ ])

$i$  **SUIVANT**

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir 20 notes et lui affiche leur moyenne

# Exercice

---

**ALGORITHME** *moyenne*

**VAR** notes[20], somme : **réel**

**DEBUT**

somme  $\leftarrow$  0

**POUR** i  $\leftarrow$  0 à 20

Afficher(" Saisir la note numéro ", i)

Lire(notes[i])

somme  $\leftarrow$  somme + notes[i]

i **SUIVANT**

Afficher("La moyenne est : ", somme/20)

**FIN**



# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir 20 notes et lui affiche leur moyenne, la note minimale et la note maximale

**ALGORITHME** *moyenne\_min\_max*

**VAR** notes[20], somme, min, max : **réel**

**DEBUT**

Afficher("Saisir la 1<sup>ère</sup> note")

Lire(notes[0])

somme, max, min  $\leftarrow$  notes[0]

**POUR** i  $\leftarrow$  1 à 20

Afficher(" Saisir la note numéro ", i)

Lire(notes[i])

somme  $\leftarrow$  somme + notes[i]

**SI** notes[i] < min **ALORS**

min  $\leftarrow$  notes[i]

**SINONSI** notes[i] > max **ALORS**

max  $\leftarrow$  notes[i]

**FINSI**

i **SUIVANT**

Afficher("Moyenne = ", somme/20, "Min = ", min, "Max = ", max)

**FIN**

# Problématique

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir des notes et lui affiche leur moyenne, la note minimale et la note maximale
- ▶ Le nombre de notes à saisir n'est pas connu à l'avance

# Problématique

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir des notes et lui affiche leur moyenne, la note minimale et la note maximale
- ▶ Le nombre de notes à saisir n'est pas connu à l'avance

**➔ Tableau dynamique**

# **Tableaux dynamiques**

# Définition

---

- ▶ Tableau dynamique

- ▶ Séquence de données de **même type**, chacune référencée par un nombre appelé **indice** (ou index), dont la **taille est variable** et peut changer lors de l'exécution

# Déclaration de tableaux dynamiques

---

## ► Syntaxe

**VAR** nomTableau[ ] : Type

## ► Exemple :

**VAR** nombres[ ] : entier

**VAR** lettres[ ] : caractère

**VAR** mots[ ] : chaîne

**VAR** absents[ ] : booléen

# Redimensionnement

---

- ▶ L'opération de redimensionnement permet de fixer la taille du tableau dynamique
- ▶ Elle intervient dans le corps de l'algorithme
- ▶ Elle est effectuée par l'instruction **Redim**



# Redimensionnement

**ALGORITHME** *tableau\_dynamique*

**VAR** notes[ ] : **réel**

**VAR** n : **entier**

**DEBUT**

Afficher("Quelle est le nombre de notes à saisir ?")

Lire(n)

**Redim** notes[n]

**POUR**  $i \leftarrow 0$  à n

Afficher(" Saisir la note numéro ",  $i + 1$ )

Lire(notes[i])

i **SUIVANT**

**FIN**

# Redimensionnement

---

- ▶ L'opération de redimensionnement permet de fixer la taille du tableau dynamique
- ▶ Elle intervient dans le corps de l'algorithme
- ▶ Elle est effectuée par l'instruction **Redim**
  - ▶ À tout moment, on peut ajouter / supprimer des cases

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir des notes et lui affiche leur moyenne, la note minimale et la note maximale
- ▶ Demander à l'utilisateur combien de notes il veut saisir

**ALGORITHME** *moyenne*

**VAR** notes[ ], somme, min, max : **réel**

**VAR** n : **entier**

**DEBUT**

somme  $\leftarrow$  0

Afficher("Combien de notes voulez-vous saisir ?")

Lire(n)

**Redim** notes[n]

Afficher("Saisir la 1<sup>ère</sup> note")

Lire(notes[0])

somme, max, min  $\leftarrow$  notes[0]

**POUR** i  $\leftarrow$  1 à n

Afficher(" Saisir la note numéro ", i)

Lire(notes[i])

somme  $\leftarrow$  somme + notes[i]

**SI** notes[i] < min **ALORS**

min  $\leftarrow$  notes[i]

**SINONSI** notes[i] > max **ALORS**

max  $\leftarrow$  notes[i]

**FINSI**

i **SUIVANT**

Afficher("Moyenne = ", somme/20, "Min = ", min, "Max = ", max)

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de saisir des nombres réels puis qui lui affiche le nombre des positifs, le nombre des négatifs, le nombre minimal et le nombre maximal
- ▶ Demander à l'utilisateur combien de nombres il veut saisir

**ALGORITHME** *exercice\_tableau\_dynamique*

**VAR** nombres[ ], min, max : **réel**

**VAR** n, nbPositifs : **entier**

**DEBUT**

nbPositifs  $\leftarrow$  0

Afficher("Combien de nombre voulez-vous saisir ?")

Lire(n)

**Redim** nombres[n]

**POUR** i  $\leftarrow$  0 à n

Afficher("Saisir le nombre numéro ", i)

Lire(nombres[i])

**SI** i = 0 **ALORS**

min  $\leftarrow$  nombres[i]

max  $\leftarrow$  nombres[i]

**SINONSI** nombres[i] < min **ALORS**

min  $\leftarrow$  nombres[i]

**SINONSI** nombres[i] > max **ALORS**

max  $\leftarrow$  nombres[i]

**FINSI**

**SI** nombres[i] >= 0 **ALORS**

nbPositifs  $\leftarrow$  nbPositifs + 1

**FINSI**

i **SUIVANT**

Afficher("Min = ", min, "Max = ", max, "Nb + = ", nbPositifs, "Nb - ", n - nbPositifs)

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de remplir deux tableaux (de même taille) puis qui affiche le tableau contenant le produit des éléments des deux

- ▶ Demander à l'utilisateur la taille des tableaux

|                |    |     |     |    |    |
|----------------|----|-----|-----|----|----|
| Tableau 1      | 5  | 2   | 10  | 15 | 1  |
| ✕<br>Tableau 2 | 5  | 60  | 10  | 0  | 90 |
| =<br>Tableau 3 | 25 | 120 | 100 | 0  | 90 |

## **ALGORITHME** *exercice\_tableau\_dynamique*

**VAR** tab1[ ], tab2[ ], tab3[ ] : **réel**

**VAR** n : **entier**

**DEBUT**

Afficher("Combien de nombre voulez-vous saisir ?")

Lire(n)

**Redim** tab1[n]

**Redim** tab2[n]

**Redim** tab3[n]

Afficher("Remplissage du 1<sup>er</sup> tableau")

**POUR** i  $\leftarrow$  0 à n

Afficher("Saisir le nombre numéro ", i)

Lire(tab1[i])

i **SUIVANT**

Afficher("Remplissage du 2<sup>e</sup> tableau")

**POUR** i  $\leftarrow$  0 à n

Afficher("Saisir le nombre numéro ", i)

Lire(tab2[i])

i **SUIVANT**

**POUR** i  $\leftarrow$  0 à n

tab3[i]  $\leftarrow$  tab1[i]\*tab2[i]

Afficher(tab1[i], "\*", tab2[i], "=", tab3[i])

i **SUIVANT**

**FIN**



# Problématique

---

- ▶ Besoin : pour une classe de 10 étudiants, nous voulons stocker, pour chaque étudiant, ses notes dans 2 matières
  - ▶ Un tableau de 10 éléments pour la note de la première matière; et un autre tableau pour la note de la deuxième

# Problématique

---

- ▶ Besoin : pour une classe de 10 étudiants, nous voulons stocker, pour chaque étudiant, ses notes dans 2 matières
  - ▶ Un tableau de 10 éléments pour la note de la première matière; et un autre tableau pour la note de la deuxième
  - ▶ Matrice de  $10 \times 2$  avec, dans chaque ligne  $i$ , les deux notes de l'étudiant numéro  $i$

# Problématique

---

- ▶ Besoin : pour une classe de 10 étudiants, nous voulons stocker, pour chaque étudiant, ses notes dans 2 matières
  - ▶ Un tableau de 10 éléments pour la note de la première matière; et un autre tableau pour la note de la deuxième
  - ▶ Matrice de 10 x 2 avec, dans chaque ligne  $i$ , les deux notes de l'étudiant numéro  $i$  → **tableau multidimensionnel**

# **Tableaux multidimensionnels**

# Définition

---

- ▶ Tableau à 2 dimensions

- ▶ Séquence de données de **même type**, chacune référencée par **deux indices**

- ▶ Désigné par

- ▶ Son nom

- ▶ Le type de ses éléments

- ▶ Sa taille  $n \times m$

- Peut être assimilé à une matrice de  $n$  lignes et  $m$  colonnes

# Déclaration de tableaux à 2 dimensions

---

## ► Syntaxe

**VAR** nomTableau[N][M] :Type

## ► Exemple :

**VAR** nombres[5][5] : entier

**VAR** lettres[26][2] : caractère

**VAR** mots[1][10] : chaîne

**VAR** absents[107][30] : booléen

# Exemple de tableaux à 2 dimensions

**ALGORITHME** *tableau\_2d*

**VAR** notes[20][2] : **réel**

**DEBUT**

**POUR**  $i \leftarrow 0$  à 20

Afficher("Saisir la 1<sup>ère</sup> note de l'étudiant N° ",  $i+1$ )

Lire(notes[i][0])

Afficher("Saisir la 2<sup>e</sup> note de l'étudiant N° ",  $i+1$ )

Lire(notes[i][1])

$i$  **SUIVANT**

**FIN**

# Exercice

---

- ▶ Que font les algorithmes suivants ?



**ALGORITHME** *exemple\_tab*

**VAR** tab[5][10] : entier

**VAR** i, j, k: entier

**DEBUT**

k  $\leftarrow$  1

**POUR** i  $\leftarrow$  0 à 5

**POUR** j  $\leftarrow$  0 à 10

tab[i][j]  $\leftarrow$  k

k  $\leftarrow$  k + 1

j **SUIVANT**

i **SUIVANT**

**FIN**

**ALGORITHME** *exemple\_tab*

**VAR** tab[5][10] : entier

**VAR** i, j : entier

**DEBUT**

**POUR** i  $\leftarrow$  0 à 5

**POUR** j  $\leftarrow$  0 à 10

            tab[i][j]  $\leftarrow$  i + j

        j **SUIVANT**

    i **SUIVANT**

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui génère un tableau de booléens de  $5 \times 10$  où toutes les cases contiennent la valeur VRAI

# Exercice

---

**ALGORITHME** *tab\_bool*

**VAR** *tab*[5][10] : **booléen**

**DEBUT**

**POUR** *i*  $\leftarrow$  0 à 5

**POUR** *j*  $\leftarrow$  0 à 10

*tab*[*i*][*j*]  $\leftarrow$  VRAI

*j* **SUIVANT**

*i* **SUIVANT**

**FIN**

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de remplir une matrice de  $10 \times 10$  et vérifie si elle est symétrique ou non

# Exercice

---

- ▶ Écrire un algorithme qui demande à l'utilisateur de remplir une matrice de  $10 \times 10$  et vérifie si elle est symétrique ou non

|          |          |          |          |
|----------|----------|----------|----------|
| <b>x</b> | <b>a</b> | <b>b</b> | <b>c</b> |
| <b>a</b> | <b>x</b> | <b>d</b> | <b>e</b> |
| <b>b</b> | <b>d</b> | <b>x</b> | <b>f</b> |
| <b>c</b> | <b>e</b> | <b>f</b> | <b>x</b> |

# Exercice

**ALGORITHME** *symétrie\_matrice*

**VAR** matrice[10][10] : **réel**

**VAR** sym : **booléen**

**DEBUT**

sym  $\leftarrow$  VRAI

**POUR** i  $\leftarrow$  0 à 10

**POUR** j  $\leftarrow$  0 à 10

Lire(matrice[i][j])

j **SUIVANT**

i **SUIVANT**

i  $\leftarrow$  0

j  $\leftarrow$  0

**TANTQUE** sym ET i < 10

**TANTQUE** j < 10

**SI** j <> i et matrice[i][j] <> matrice[j][i] **ALORS**  
sym = FALSE

**FINSI**

j  $\leftarrow$  j + 1

**FINTANTQUE**

i  $\leftarrow$  i + 1

**FINTANTQUE**

**SI** sym **ALORS**

Afficher("La matrice saisie est symétrique")

**SINON**

Afficher("La matrice saisie n'est pas symétrique")

**FINSI**

**FIN**



# Exercice

---

- ▶ Écrire un algorithme qui cherche, dans un tableau d'entiers de  $10 \times 10$ , le nombre minimal et le nombre maximal



**ALGORITHME** *tab\_bool*

**VAR** tab[10][10], min, max : **réel**

**VAR** i, j : **entier**

**DEBUT**

*//instructions de remplissage du tableau omises*

min, max  $\leftarrow$  tab[0][0]

**POUR** i  $\leftarrow$  0 à 10

**POUR** j  $\leftarrow$  0 à 10

**SI** tab[i][j] < min **ALORS**

min  $\leftarrow$  tab[i][j]

**SINONSI** tab[i][j] > max **ALORS**

max  $\leftarrow$  tab

**FINSI**

j **SUIVANT**

i **SUIVANT**

Afficher("Max = ", max, " et Min = ", min)

**FIN**

# **Recherche dans un tableau**

# Recherche dans un tableau

---

## ► Proposition ?

# Recherche séquentielle

---

- ▶ Vérifier si une valeur existe dans un tableau → utiliser un flag
- ▶ Parcourir le tableau
  - ▶ Si élément courant = valeur → flag = VRAI
- ▶ Vérifier le flag à la fin de l'exécution
  - ▶ Si flag = VRAI → l'élément existe

# Recherche séquentielle

---

- ▶ Pseudo-code
  - ▶ Recherche de la valeur *val* dans le tableau `tab[N]`

# Recherche séquentielle

existe  $\leftarrow$  FAUX

**POUR**  $i \leftarrow 0$  à  $N$

**SI**  $\text{tab}[i] = \text{val}$  **ALORS**

    existe  $\leftarrow$  VRAI

**FINSI**

**SI** existe **ALORS**

    Afficher("La valeur recherchée a été trouvée")

**SINON**

    Afficher("La valeur recherchée n'a pas été trouvée")

**FINSI**

$i$  **SUIVANT**

# Recherche séquentielle

---

- ▶ Pseudo-code
  - ▶ Recherche de la valeur *val* dans le tableau `tab[N]`
  - ▶ Amélioration possible : ne pas continuer les comparaisons si l'élément a été trouvé
  - ▶ Encore mieux : **recherche dichotomique**

# Recherche dichotomique

---

## ▶ Principe

- ▶ Comparer la valeur à trouver avec l'élément au milieu du tableau
  - ▶ Si inférieure, refaire la même comparaison avec la première moitié du tableau
  - ▶ Si supérieure, refaire la même comparaison avec la deuxième moitié du tableau
- ▶ Nécessite que le tableau soit trié



# Recherche dichotomique

---

- ▶ Pseudo-code
  - ▶ Recherche de la valeur *val* dans le tableau `tab[N]`

# Recherche dichotomique

existe  $\leftarrow$  FAUX

debut  $\leftarrow$  0

fin  $\leftarrow$  N

**TANTQUE** existe = FAUX ET debut  $\leq$  fin

milieu  $\leftarrow$  (debut + fin) DIV 2

**SI** tab[milieu] = val **ALORS**

existe  $\leftarrow$  VRAI

**SINONSI** tab[milieu] < val **ALORS**

debut  $\leftarrow$  milieu + 1

**SINON**

fin  $\leftarrow$  milieu - 1

**FINSI**

**FINTANTQUE**



## 5. Les tableaux