R1.01 : Initiation au développement Cours 3

M.Adam, N.Delomez, JF.Kamp, L.Naert

IUT de Vannes

7 août 2022

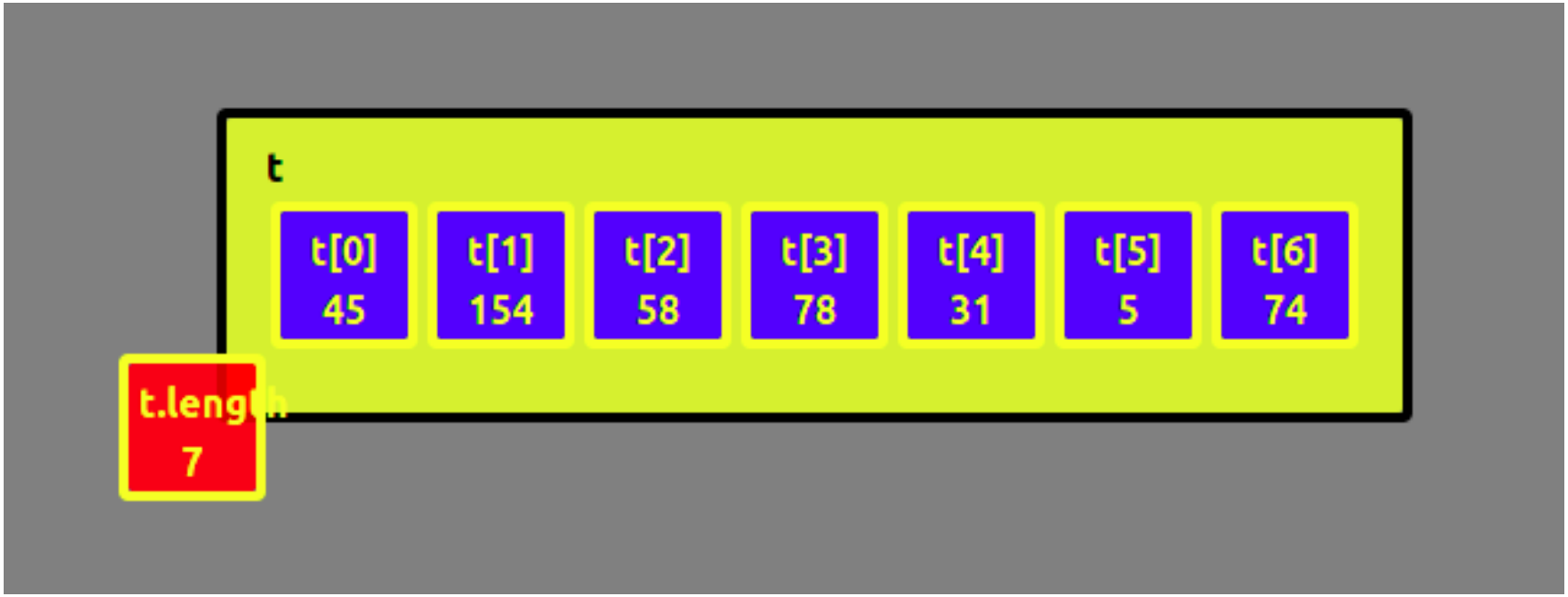
Sommaire

Définition

En informatique, un tableau (array en anglais) est une structure de données de base qui est un ensemble d'éléments (des variables ou autres entités contenant des données), auquel on a accès à travers un numéro d'index (ou indice).
Source `http://fr.wikipedia.org/`

Valeur	**45**	**154**	**58**	**78**	**31**	**5**	**74**
Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**

Avec AlgoTouch



Les particularités

En java,

- le premier indice est 0,
- un tableau est une suite d'éléments de **même type**.

Attention : un tableau peut avoir autant d'indices, de dimensions que nécessaire.

Déclaration d'un tableau

```
type[] tab;
```

Attention en Java, il ne s'agit que de la création d'une référence à un tableau.

Création d'un tableau

Un tableau peut être créé à la déclaration :

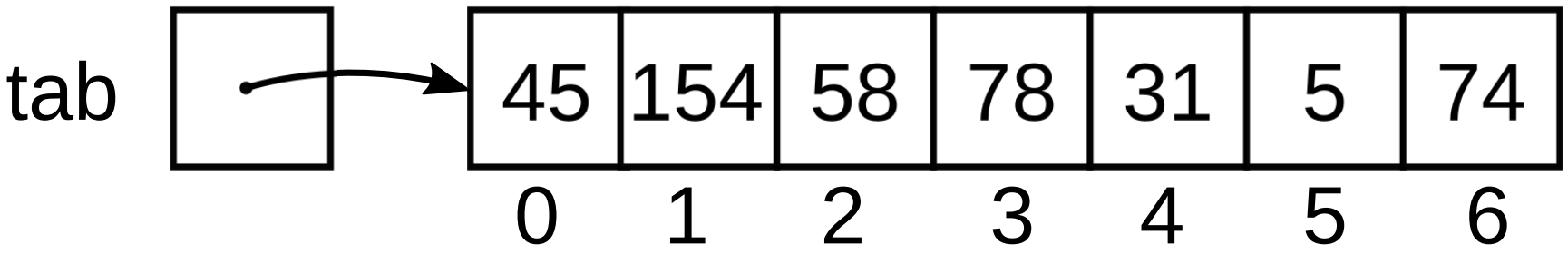
```
int[] tab = {45, 154, 58, 78, 31, 5, 74}
```

ou après la déclaration :

```
int[] tab;
...
tab = new int[7];
```

Dans ce cas tous les éléments du tableau sont à 0.

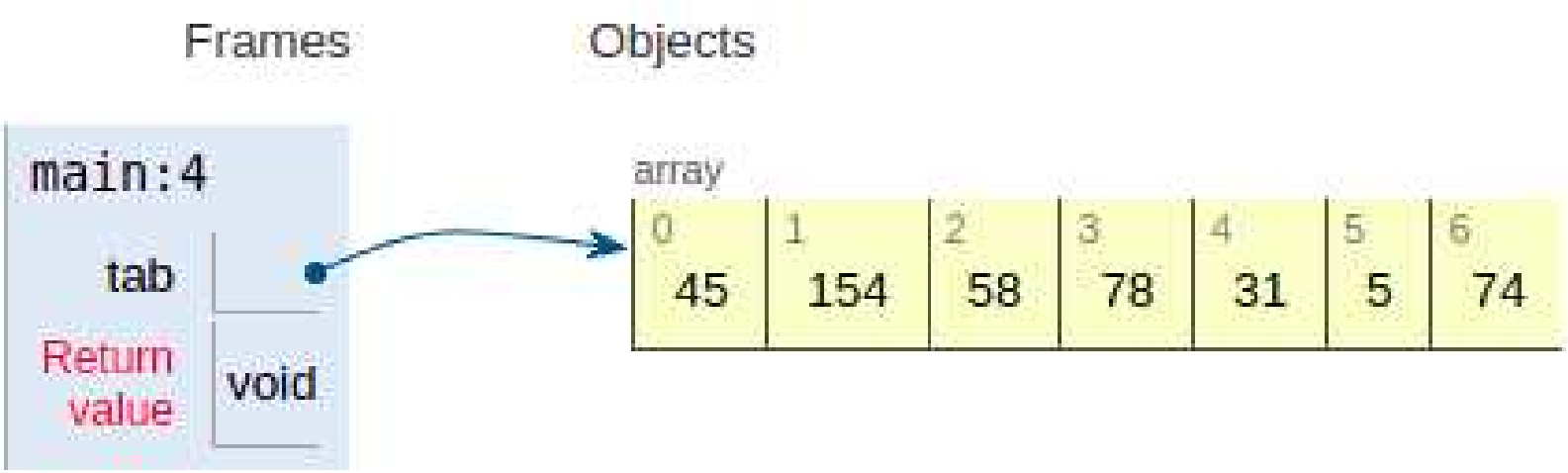
Exemple de déclaration




```
int[] tab = {45, 154, 58, 78, 31, 5, 74};
```

Exemple sur JavaTutor https://tinyurl.com/C03JT01

Avec AlgoTouch



Longueur d'un tableau

La longueur d'un tableau est son nombre d'éléments.
il est accessible par `tab.length` pour un tableau `tab`.

```
/**
 * Affichage de la taille d'un tableau
 * @author M.Adam
 **/

class TailleTab {
    void principal () {
        int[] tab;
        tab = new int[7];
        System.out.println ();
        System.out.println ("tab.length = " + tab.length);
    }
}
```

```
$

tab.length = 7

$
```

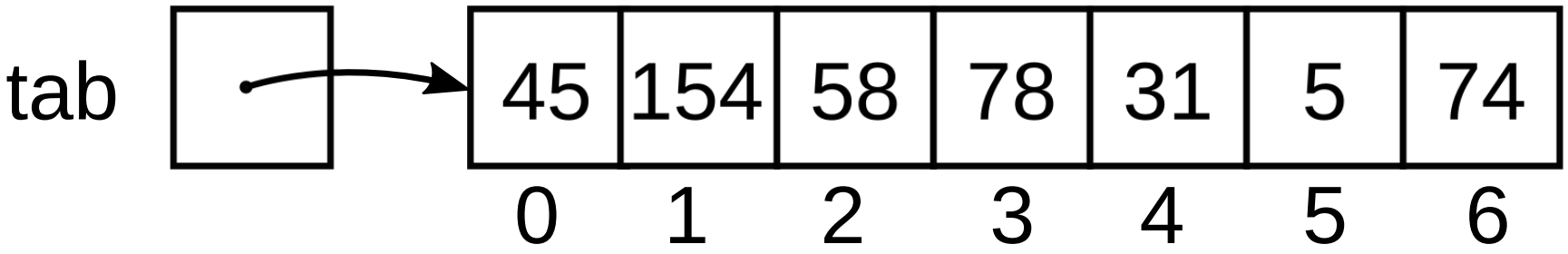
- Exemple sur JavaTutor : https://tinyurl.com/C03JT03
- Exemple sur AlgoTouch : https://tinyurl.com/C03AGT03

Affectation

```
int[] t = new int[7];
int indice = 5;


tab[indice] = 31;
tab[5] = SimpleInput.getInt("Valeur du tableau");
```

Exemple d'affectation




```
tab[0] = 45;
tab[1] = 154;
tab[2] = 58;
tab[3] = 78;
tab[4] = 31;
tab[5] = 5;
tab[6] = 74;
```

Lecture du contenu d'un tableau

```
int[] tab = {45, 154, 58, 78, 31, 5, 74};
int i = 5;

System.out.println (tab[5]);
System.out.println (tab[i]);
```

- Exemple sur JavaTutor : https://tinyurl.com/C03JT04
- Exemple sur AlgoTouch : https://tinyurl.com/C03AGT04

Exemple de l'affichage du contenu d'un tableau

```
i = 0;
while (i < tab.length) {
    System.out.println ("tab["+i+"] = " + tab[i]);
    i = i + 1;
}
```

Exemple complet

```
/**
* Création d'un tableau d'entiers et
* affichage de son contenu
* @author M.Adam
**/
class Tableau {
    void principal () {
        int i;
        int[] tab = {45, 154, 58, 78, 31, 5, 74};

        i = 0;
        while (i < tab.length) {
            System.out.println ("tab["+i+"] = " + tab[i]);
            i = i + 1;
        }
    }
}
```

L'exécution produit :

```
$
tab[0] = 45
tab[1] = 154
tab[2] = 58
tab[3] = 78
tab[4] = 31
tab[5] = 5
tab[6] = 74

$
```

- Exemple sur JavaTutor : https://tinyurl.com/C03JT05
- Exemple sur AlgoTouch : https://tinyurl.com/C03AGT05

Sommaire

Exemple du cours

Soient deux tableaux, `tab1` et `tab2`, contenant 10 entiers :

```
final int LG_TAB = 10;
int[] tab1 = new int[LG_TAB];
int[] tab2 = new int[LG_TAB];
```

L'algorithme doit comparer les deux tableaux et dire si les deux tableaux sont identiques :

- mêmes éléments au même indice.

Construction méthodique

La méthode consiste à construire la boucle `while` dans l'ordre suivant :

1. Principe : explication du principe général de la boucle
2. Corps de la boucle
3. Les conditions de sortie
4. La condition de continuation
5. L'initialisation
6. La terminaison
7. L'écriture complète du code

Principe

L'explication en français ou avec des schémas du principe du calcul à effectuer pour obtenir le résultat.

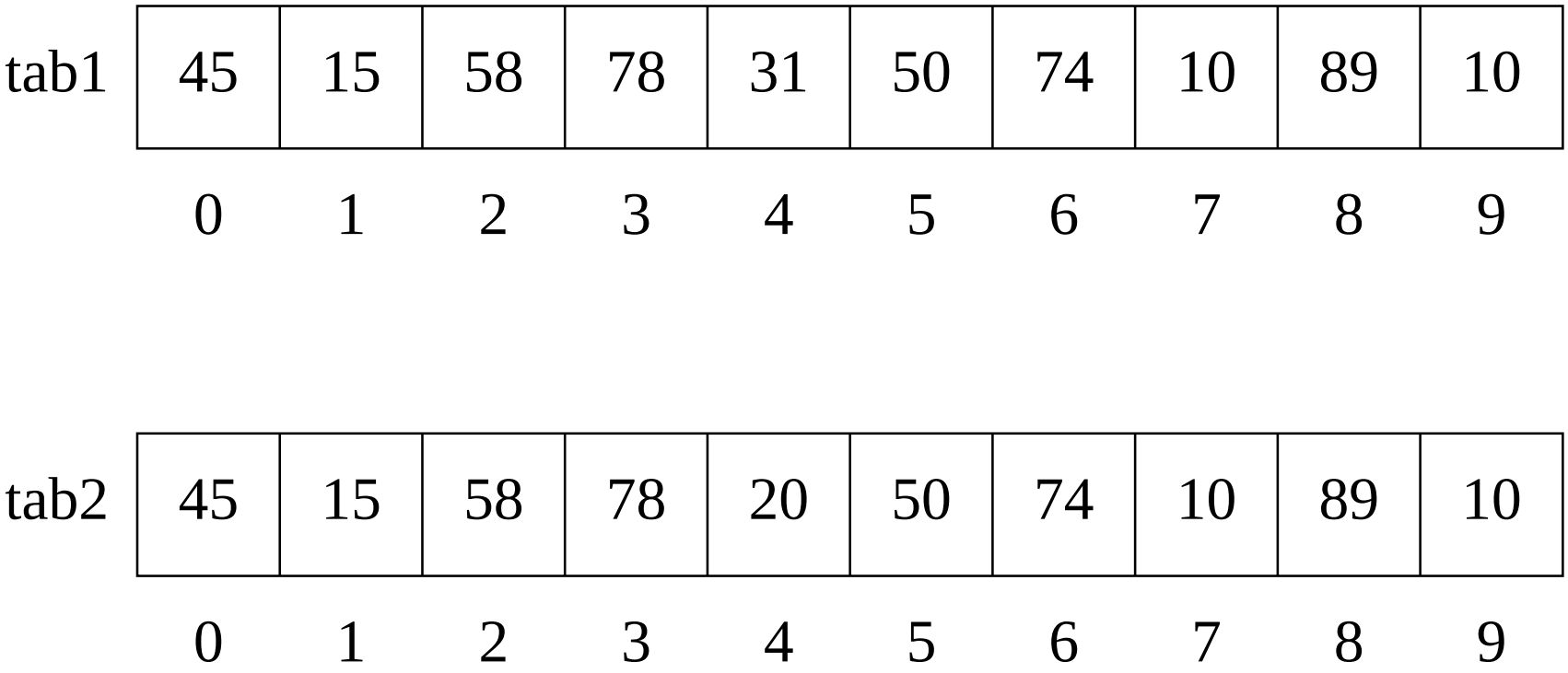
Exemple de principe

- Les deux tableaux sont parcourus en parallèle.
- Une variable booléenne `idem` est à vrai tantque les deux tableaux sont identiques.

	0	1	2	3	4	5	6	7	8	9
tab1	45	15	58	78	31	50	74	10	89	10

	0	1	2	3	4	5	6	7	8	9
tab2	45	15	58	78	31	50	74	10	89	10

Exemple avec AlgoTouch


tab1
45 15 58 78 31 50 74 10 89 10
0 1 2 3 4 5 6 7 8 9
tab2
45 15 58 78 20 50 74 10 89 10
0 1 2 3 4 5 6 7 8 9

Corps de boucle

- Écrire le code du corps de la boucle,
- Spécifier les variables nécessaires.

Même si nous n'insisterons pas, deux points doivent être vérifiés :

- **Invariant** : propriété vraie avant et après l'exécution du corps de boucle,
- **Progression** : à chaque tour de boucle, il faut diminuer la "distance" par rapport à la solution.

Exemple de corps de boucle

Deux variables sont utilisées :

- `idem` : booléen vrai ssi les deux tableaux sont identiques,
- `i` : indice de parcours des deux tableaux.

```
if (tab1[i] != tab2[i]) {
    idem = false;
}
i = i + 1;
```

Cette dernière instruction assure la progression.

Conditions de sortie

Pour quelles raisons ne pas exécuter à nouveau le corps de boucle ?
Ces conditions peuvent être classées en deux :

- celles qui font que l'exécution du corps de boucle provoquera une erreur. Elles sont obligatoires,
- celles qui indiquent que la solution est trouvée. Elles sont le plus souvent facultatives, et permettent de rendre l'algorithme plus rapide.

Ces conditions sont disjonctives (ou, ||).

Exemple de condition de sortie

- Dépassement de la taille du tableau : `i >= tab1.length`
- Valeurs différentes à un indice : `!idem`

La condition de sortie est donc :

```
i >= tab1.length || !idem
```

qui peut aussi s'écrire :

```
!idem || i >= tab1.length
```

Condition de continuation

Quelle est la condition pour continuer à exécuter le corps de la boucle ?
Il s'agit de la **négation logique** de celle de la condition de sortie.

Exemple de condition continuation

```
!(i >= tab1.length || !idem)
```

qui est logiquement équivalent à

```
i < tab1.length && idem
```

Il est possible de conserver la première écriture.

Initialisation

C'est le code à exécuter avant le premier tour de boucle.

Exemple d'initialisation

Il est nécessaire de commencer au début du tableau.
Au début, le tableau `tab1` est vu comme identique `tab2`.

```
i = 0;
idem = true;
```

La terminaison

- Le code à exécuter après le dernier tour de boucle.
- Parfois (souvent) cette partie est vide.

Exemple de terminaison

```
if (idem) {
    System.out.println
        ("Les deux tableaux sont identiques");
} else {
    System.out.println
        ("Les deux tableaux sont différents");
}
```

Code complet

Le code complet est constitué de la manière suivante :

```
initialisation;
while (condition de continuation) {
    corps de boucle;
}
terminaison;
```

Exemple du code complet

```
int i;
boolean idem;
...
i = 0;
idem = true;

while (i < tab1.length && idem) {
    if (tab1[i] != tab2[i]) {
        idem = false;
    }
    i = i + 1;
}

if (idem)  {
    System.out.println("Les deux tableaux sont identiques");
} else {
    System.out.println("Les deux tableaux sont différents");
}
```

Avec AlgoTouch

- En vidéo : `https://tinyurl.com/C03AGTYT08`
- En AlgoTouch : `https://tinyurl.com/C03AGT07`

Remarque

Évidemment cette manière de construire une boucle peut paraitre longue et fastidieuse, mais elle est une bonne garantie de la correction du code. Nous pourrons utiliser, par la suite, une forme plus rapide pour écrire le code.

Questions

- Que serait le programme de l'exemple si nous n'avions que la première condition de sortie `i >= tab1.length` ?
- Que serait le programme de l'exemple si nous n'avions pas la deuxième condition `idem` ?

Petit exercice

L'objectif est d'écrire un programme qui détermine la longueur de la plus grande suite croissante d'un tableau d'entiers.

Sommaire

Autres formes de boucles

- Comment écrire une boucle quand il faut au moins une fois exécuter le corps ?
- Comment écrire une boucle quand le nombre de tours est connu à l'avance ?

Les boucles imbriquées

Comment appliquer la méthode de construction des boucles quand il y a imbrication ?
Une idée : diviser le gros problème en petits problèmes.