Cours 3

M.Adam – JF.Kamp – S.Letellier – F.Pouit

1er juillet 2016

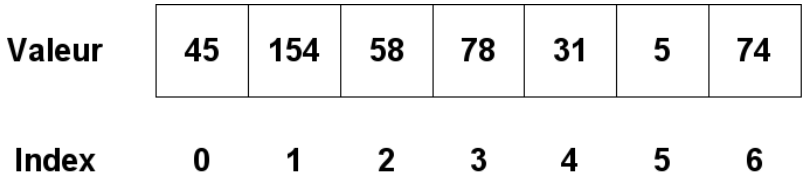
Table des matières

1 Les tableaux

1.1 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/`



1.2 Les particularités de TestAlgo

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

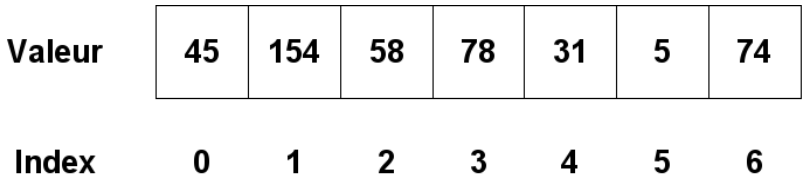
1.3 Déclaration d'un tableau

```
tableau_de type monTab [TAILLE];
```

avec

- `type` vaut `entier`, `reel`, `booleen`, `chaine` ou `caractere`,
- `TAILLE` est un entier supérieur à 0.

1.3.1 Exemple de déclaration

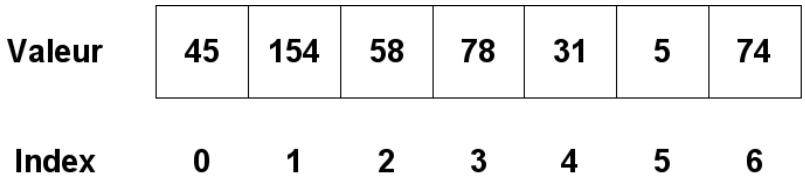



```
tableau_de entier valeurs[7];
```

1.4 Affectation

```
monTab[INDICE] := une_valeur;
saisir("Valeur du tableau",@monTab[INDICE]);
```

1.4.1 Exemple d'affectation




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

1.5 Lecture

```
monTab[INDICE]
```

1.5.1 Exemple de lecture

```
i := 0;
tantque (i <> 7)
    afficherln(valeurs[i]);
    i := i+1;
fintantque
```

1.6 Exemple complet

```
##
# Création d'un tableau d'entier et
# affichage de son contenu
# @author M.Adam
##
algo Tableau
principal
var
```

```
    entier i;
    tableau_de entier valeurs[7];
debut
    valeurs[0] := 45;
    valeurs[1] := 154;
    valeurs[2] := 58;
    valeurs[3] := 78;
    valeurs[4] := 31;
    valeurs[5] := 5;
    valeurs[6] := 74;
    i := 0;
    tantque (i <> 7)
      afficherln("valeurs["+i+"] = "+valeurs[i]);
      i := i+1;
    fintantque
fin
```

L'exécution produit :

```
TestAlgo - Interprétation engagée.
valeurs[0] = 45
valeurs[1] = 154
valeurs[2] = 58
valeurs[3] = 78
valeurs[4] = 31
valeurs[5] = 5
valeurs[6] = 74
TestAlgo - Fin de l'interprétation.
```

1.7 Les limitations de TestAlgo

- Tous les éléments d'un tableau sont de même type,
- il n'est pas possible d'utiliser des tableaux de tableau,

2 Construction méthodique des boucles

2.1 Rôle de la boucle

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

```
tableau_de entier tab1[10];
tableau_de entier tab2[10];
```

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

- même éléments au même indice.

2.2 Construction méthodique

La méthode consiste à construire la boucle `tantque` 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

2.3 Principe

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

2.3.1 Exemple de principe

- Les deux tableaux sont parcourus en parallèle.
- Une variable booléenne `idem` si les deux tableaux sont identiques.

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
	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	20	50	74	10	89	10
	0	1	2	3	4	5	6	7	8	9

2.4 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** : à chaque tour de boucle, le code de la boucle doit s'exécuter sans erreur,
- **Progression** : à chaque tour de boucle, il faut diminuer la "distance" par rapport à la solution.

2.4.1 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.

```
si (tab1[i] <> tab2[i]) alors
    idem := faux;
finsi
i := i+1;
```

Cette dernière instruction assure la progression.

2.5 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 facultatives, et permettent juste de rendre l'algorithme plus rapide.

Ces conditions sont disjonctives (ou).

2.5.1 Exemple de condition de sortie

- Dépassement de la taille du tableau : `i >= 10`
- Valeurs différentes pour le dernier indice : `non idem`

La condition de sortie est donc :

```
i >= 10 ou non idem
```

2.6 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.

2.6.1 Exemple de condition de sortie

```
non (i >= 10 ou non idem)
```

qui est logiquement équivalent à

```
i < 10 et idem
```

2.7 Initialisation

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

2.7.1 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 := vrai;
```

2.8 La terminaison

Le code à exécuter après le dernier tour de boucle.

2.8.1 Exemple de terminaison

```
si (idem) alors
    afficherln("Les deux tableaux sont identiques");
sinon
    afficherln("Les deux tableaux sont différents");
finsi
```

2.9 Code complet

Le code complet est constitué de la manière suivante :

```
initialisation;
tantque (condition de continuation)
    corps de boucle;
fintantque
terminaison;
```

2.9.1 Exemple du code complet

```
entier i;
booleen idem;
...
i := 0;
idem := vrai;
tantque (i < 10 et idem)
    si (tab1[i] <> tab2[i]) alors
        idem := faux;
    finsi
    i := i+1;
fintantque

si (idem) alors
    afficherln("Les deux tableaux sont identiques");
sinon
    afficherln("Les deux tableaux sont différents");
finsi
```

2.10 Remarque

Évidemment cette manière de construire une boucle peut paraitre longue et fastidieuse, mais elle est une meilleure garantie de la correction du code.

Nous pourrons utiliser, par la suite, une forme plus rapide pour écrire le code.

2.11 Une question

Que serait l'algorithme de l'exemple si nous n'avions que la première condition de sortie `i <= 10` ?

3 Et maintenant ?

3.1 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 ?

3.2 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.