Elements d'algorithmique

TP 5 : les tableaux (2)

1 Parcours d'un tableau

Un tableau est souvent manipulé élément par élément. On peut commencer par le premier, continuer avec le second, ... jusqu'au dernier. Pour parcourir un tableau, on peut procéder en utilisant une boucle *FOR*.

Exemple :

```
PROGRAM max;
VAR tab : ARRAY [1..10] OF INTEGER;
    i, tab_max : INTEGER;
BEGIN
FOR i := 1 TO 10 DO
    readln(tab[i]);
tab_max := tab[1];
FOR i := 1 TO 10 DO
    IF (tab[i] > tab_max) THEN
        tab_max := tab[i];
writeln('Max: ', tab_max);
readln;
END.
```

2 Passage d'un tableau en paramètre

Les deux procédures *exemple*1 et *exemple*2 ci-dessous illustrent les deux moyens de passer un tableau en paramètre selon si sa taille est fixée ou inconnue. Il faut remarquer que dans le prototype de la procédure *exemple*1, la taille de la variable tableau *tab* est spécifié à l'aide de la notation [1..20] alors que dans le prototype de la procédure *exemple*2, la taille du tableau doit être passée en second paramètre n. Attention ! lorsque la notation pour le passage de tableau de taille inconnue est employée, le premier élément n'est plus *tab*[1] mais *tab*[0] et ainsi de suite.

```
PROCEDURE exemple1(VAR tab : ARRAY[1..20] OF INTEGER);
BEGIN
    tab[1] := 100 (* accès au premier élément *)
    tab[2] := 100 (* accès au deuxième élément *)
    tab[20] := 100 (* accès au dernier élément *)
END;

PROCEDURE exemple2(VAR tab : ARRAY OF INTEGER; n : INTEGER);
BEGIN
    tab[0] := 100 (* accès au premier élément *)
    tab[1] := 100 (* accès au deuxième élément *)
```

```
    tab[n - 1] := 100 (* accès au dernier élément *)
END;

VAR tab1 : ARRAY[1..20] OF INTEGER;
    tab2 : ARRAY[1..300] OF INTEGER;
BEGIN (* programme principal *)
    exemple1(tab1);
    exemple2(tab1, 20);
    exemple2(tab2, 300);
END.
```

3 Exercices

3.1 Exercice 1

En algorithmique, la dichotomie (du grec "couper en deux") est un processus de recherche où à chaque étape l'espace de recherche est restreint à l'une de deux parties. Voici un exemple simple pour illustrer le mécanisme de recherche par dichotomie : Pierre propose à Paul le jeu suivant : "choisis en secret un nombre compris entre 0 et 100 ; je vais essayer de le deviner le plus rapidement possible, mais tu ne dois répondre à mes questions que par oui ou par non". Paul choisit 65 et attend les questions de Pierre :

```
* est-ce que le nombre est plus grand que 50? (100 divisé par 2)
* oui
* est-ce que le nombre est plus grand que 75? ((50 + 100) / 2)
* non
* est-ce que le nombre est plus grand que 63? ((50 + 75 + 1) / 2)
* oui
```

Pierre réitère ses questions jusqu'à trouver 65. Par cette méthode itérative, Pierre est sûr de trouver beaucoup plus rapidement le nombre qu'en posant des questions du type "est-ce que le nombre est égal à 30 ?".

Écrire une fonction *rechercheDicho* qui accepte pour paramètre un tableau **trié** *tab* et sa taille n ainsi qu'un élément *elt* à rechercher dans le tableau. La fonction renverra l'indice de l'élément trouvé dans le tableau ou 0 si l'élément n'existe pas. Par exemple, l'indice de l'élément 7 dans $1, 3, 3, 4, 7$ est 5. Le tableau étant trié, on mettra en oeuvre la recherche par dichotomie.

3.2 Exercice 2

Écrire une fonction *existeElt* qui accepte pour paramètre un tableau *tab* et sa taille n ainsi qu'un élément *elt* à rechercher dans le tableau. La fonction renverra un booléen *true* si *elt* existe dans le tableau, $false$ sinon.

3.3 Exercice 3

Écrire une procédure ajoutant un entier k donné en paramètre dans un tableau trié *tab* fourni lui aussi en paramètre. Exemple : si $tab = 1, 5, 12, 13$ et $k = 10$ alors à la fin de la procédure *tab* contiendra $1, 5, 10, 12, 13$

3.4 Exercice 4

Écrire une procédure *init* qui initialise un tableau *tab* avec N réels générés aléatoirement compris dans l'intervalle $[-1, 1]$.