Licence Informatique — **UE INF101**
Licence Mathématiques — **Initiation à la programmation**
Université Bordeaux 1 — **Travaux Dirigés**
2006-2007 — **feuille 4**

Tableaux

Exercice 1. Ecrire un programme `tableau` comportant les fonctions suivantes

1. `void affiche_tableau(int t[], int n)` qui affiche les `n` premiers éléments du tableau `t` passé en paramètre.
2. `void saisir(int t[], int n)` qui stocke dans le tableau `t` passé en paramètre les `n` valeurs entières saisies.
3. `int maximum(int t[], int n)` qui retourne le maximum des `n` premiers éléments du tableau `t` passé en paramètre.
4. `int indice_minimum(int t[], int n)` qui retourne l'indice du minimum des `n` premiers éléments du tableau `t` passé en paramètre.
5. `int amplitude_maximum(int t[], int n)` qui retourne la différence entre le plus grand et le plus petit des `n` premiers éléments du tableau `t` passé en paramètre.

Note : Pour chaque fonction, le tableau `t` doit avoir au moins `n` cases allouées avant l'appel à la fonction.

Le programme `tableau` permet de tester les différentes fonctions sur le tableau d'entiers dont la taille (i.e. le nombre d'éléments) et les valeurs sont saisies.

Exercice 2. Ecrire une fonction `void doubler(int t[], int n)` qui modifie le tableau `t` en doublant chacun de ses termes.
Exemple : si avant l'appel le tableau `t` a pour éléments 1, -3, 12, -183, après l'appel le tableau contiendra 2, -6, 24, -366.

Exercice 3. Ecrire une fonction `void cumuler(int t[], int n)` qui modifie le tableau `t` en plaçant dans chaque case la somme des valeurs des termes précédents.
Exemple : si avant l'appel le tableau `t` a pour éléments 1, -3, 12, -183, après l'appel le tableau contient 1, -2, 10, -173.

Exercice Complémentaire 1. Soit un tableau `t` de `n` entiers. Pour les éléments `t[i]` de ce tableau, $0 < i < n-1$, on définit un voisinage `N(t[i],r)` de rayon `r`,`r>0`, et de centre l'élément `t[i]` comme l'ensemble `N(t[i],r)` :

$$N(t[i], r) = \{t[j], i - r \leq j \leq i + r\}$$

On dit que $t[i]$ est un `minimum local` dans un rayon r, $r > 0$, ssi $t[i]$ est le plus petit élément de $N(t[i], r)$.

Écrire les fonctions suivantes :

- La fonction `void afficher_voisinage(int t[], int n, int i, int rayon)`.
  Par exemple pour le tableau d'entiers $t = \{11, 5, 10, 3, 5, 10, 11\}$, `afficher_voisinage(t, 7, 3, 1)` affiche $N(t[3], 1) = \{10, 3, 5\}$.
- La fonction `bool est_minimum_local(int t[], int n, int i, int r)` qui renvoie `true` si $t[i]$ est un `minimum local` dans un rayon r ou `false` sinon.
  Par exemple `est_minimum_local(t, 7, 3, 2)` renvoie `true`, et `est_minimum_local(t, 7, 2, 1)` renvoie `false`.
- La fonction `int rechercher_voisinage_maximal(int t[], int n, int i)` qui renvoie le rayon maximal du voisinage pour lequel $t[i]$ est minimum local s'il existe, et 0 sinon.
  Par exemple `rechercher_voisinage_maximal(t, 7, 3)` renvoie `3`, et `rechercher_voisinage_maximal(t, 7, 2)` renvoie `0`.

Écrire un programme `tester-voisinage` qui permet de tester les fonctions précédentes.