Département Génie Électrique et Informatique Industrielle

Période 3 Les tableaux

1/ Saisie et affichage d'un tableau ☼

Écrire la fonction **`entrerTab`** qui permet de saisir au clavier **`n`** valeurs pour les ranger dans un tableau **`t`** : **`void entrerTab( int *t, int n ) ;`**

Écrire la fonction **`void afficheTab(const int * t , int taille) ;`**
qui affiche les éléments d'un tableau d'entiers **`t`** de **`taille`** éléments.

On utilisera ces fonctions par la suite pour saisir ou afficher des tableaux d'entier.

Modifier ces fonctions pour créer les fonctions **`entrerTabf et afficheTabf`** pour saisir ou afficher des tableaux de réels.

2/ Recopie de certains éléments d'un tableau ☼

Écrire la fonction

`void copiePos(int * dest, const int * source, int n) ;`

qui recopie dans **`dest`** au plus **`n`** éléments strictement positifs de **`source`**. Les valeurs recopiées dans le tableau dest seront "regroupées au début".

Essayer vos fonctions avec les tableaux ci-dessous :

```
int  tabInt[10]={65,21,-9,23,-1,32,5,-69,12,-3};
int  tabPos[10]={0} ;
```

2/ Calculs et recherches sur un tableau

Écrire la fonction **`moyTab`** qui retourne la moyenne des n premières valeurs d'un tableau t.

`float moyTab(const int *t, int n ) ;`☼

Écrire les fonctions **`maxTab`** et **`minTab`** qui retournent respectivement la plus grande et la plus petite des n premières valeurs d'un tableau t.

`int maxTab(const int * t , int n ) ;`☼
`int minTab(const int * t , int n ) ;`☼

Écrire la fonction **`indexMaxTab`** qui retourne l'indice de la plus grande valeur des n premières valeurs d'un tableau t :

`int  indexMaxTab(const int * t , int n ) ;` ☼☼

Écrire la fonction **`rechercheTab`** qui retourne 1 si la valeur **`x`** se trouve dans les **`n`** premières cases d'un tableau **`t`** et 0 sinon :

`int rechercheTab( const int * t, int x , int n) ;` ☼☼

Écrire les prototypes des fonctions identiques aux précédentes pour lesquelles le tableau est un tableau de réel.

Écrire la fonction **interval** qui retourne le nombre de valeurs comprises entre **min** et **max** dans le tableau **t** qui comporte **n** valeurs :
int interval (const float *t, float min, float max, int n) ; ☼☼

Écrire la fonction **normalize** qui normalise dans un tableau **tNorm** un tableau **t** de **n** éléments dans l'intervalle d'arrivée **min**, **max**.

```
     norm = (orig - MIN) * (max - min) / (MAX - MIN) + min
[MIN,MAX] : intervalle d'origine des valeurs de t, utiliser les
fonctions maxTab et minTab
[min,max] : intervalle cible
orig : valeur dans l'intervalle d'origine (du tableau t)
norm : valeur normalisée dans l'intervalle cible (du tableau
tNorm)
```

void normalize(const int*t, float *tNorm, float min, float max,int n);
☼☼

4/ Le loto ☼☼☼

Le principe du loto Flash (avant 2008) est un tirage aléatoire de 6 entiers compris entre 1 et 49.
(cf http://fr.wikipedia.org/wiki/Loto) Attention un tirage ne doit pas comporter de doublon.

Écrire la fonction **loto** qui remplit le tableau **t** avec **n** nombres aléatoires compris entre 1 et 49 :

```
void loto(int *t, int n ) ;
```

Écrire la fonction **nbBons** qui retourne le nombre de valeurs identiques entre deux tableaux sans doublons de même taille :

```
int nbBons(const int * t1 ,const int * t2 , int taille) ;
```

En utilisant les fonctions précédentes écrire un programme qui "joue" au loto jusqu'à avoir les 6 bons numéros. Après avoir "gagné", votre programme indiquera : le nombre de tirages total et le nombre de fois qu'il a trouvé 0,1,2,3,4,5 ou 6 bons numéros.

NB : Bien tester les deux fonctions avant d'écrire le programme complet. Commencer par essayer d'avoir seulement 3 ou 4 bons numéros car la durée pour avoir les 6 numéros peut être assez importante.

5/ Insertion, suppression d'éléments ☼☼☼

Écrire la fonction **inser** qui, dans le tableau **t**, insère à la position **index** l'entier **val**.

```
int inser(int *t, int index, int val, int n) ;
```

Les éléments suivants sont décalés d'une position le dernier est perdu.
Le tableau comporte **n** éléments. La fonction retourne 0 en cas de succès -1 sinon.

Écrire la fonction **suppr** qui, dans le tableau **t**, supprime l'élément à la position **index**. Les éléments suivants sont décalés d'une position le dernier est recopié. Le tableau comporte **n** éléments. La fonction retourne 0 en cas de succès -1 sinon.

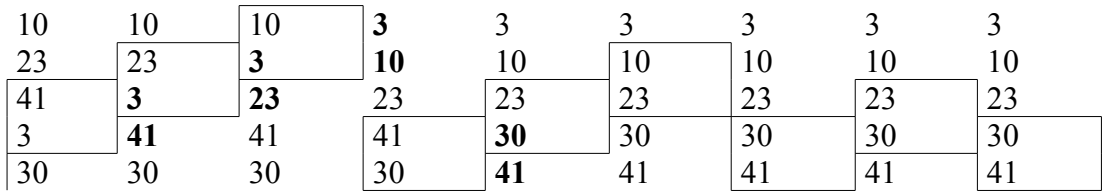
```
int suppr(int *t, int index, int n) ;
```

6/ Tri d'un tableau d'entiers ☼☼☼

On désire mettre en œuvre l'algorithme du tri à bulles sur un tableau d'entiers. Cette méthode consiste à comparer les éléments du tableau deux à deux, à partir du bas, et à les permuter si l'élément du bas est inférieur à l'autre. A la fin, le tableau sera classé par ordre croissant.

ex: Soit le tableau correspondant à l'initialisation suivante:

```
int TableauEntier [5]={10,23,41,3,30};
```



Ce qui donne l'algorithme suivant:

```
Entier i,j, TableauEntier [5]
N=5
Pour j variant de 1 à N-1
  Pour i variant de N-1 à j
        Si TableauEntier [i] < TableauEntier [i-1]
              Permuter TableauEntier [i] avec TableauEntier  [i-1]
        Fsi
  Finpour
Finpour
```

On remarque que cet algorithme peut être amélioré car, si au cours d'un parcours du tableau aucune valeur n'est permutée le tableau est trié, il n'est donc plus nécessaire de le parcourir une nouvelle fois.

Écrire la fonction **`trie`** qui trie un tableau d'entier **`t`** de taille **`n`** et qui retourne le nombre de permutations effectuées :

```
int trie( int * t , int n ) ;
```