

Nom, prénom : \_\_\_\_\_ Groupe : \_\_\_\_\_

## TP 9 : Tableaux à 1 dimension

### *Préparation*

1. Ecrire un programme qui déclare et initialise un tableau de dix flottants avant d'en afficher la moyenne.

2. Ecrire un programme qui remplit un tableau de SizeTab valeurs avec des chiffres tirés au hasard entre 1 et 10, SizeTab étant une constante symbolique.

3. Pour décoder un message présenté sous forme de chiffres compris entre 1 et 26 nous devons effectuer deux opérations :

- convertir le chiffre en lettre ( $1 \rightarrow a, 2 \rightarrow b, 3 \rightarrow c \dots$ )
- effectuer un triple décalage à droite des lettres de l'alphabet. ( $a \rightarrow d, b \rightarrow e, \dots$ )

Voici la correspondance des lettres :

$0 \rightarrow \_ \quad 1 \rightarrow d \quad 2 \rightarrow e \quad 3 \rightarrow f \quad \dots \quad 24 \rightarrow a \quad 25 \rightarrow b \quad 26 \rightarrow c$

Ecrire une fonction qui reçoit un entier en argument. Si cet entier est nul la fonction renvoie le caractère espace, sinon la fonction renvoie la lettre décodée. Par exemple `printf ("%c", decode (25))` ; affiche la lettre b. Il est formellement interdit de tester les 27 cas un par un .

## ***Travail pratique***

Exercice 1 : Saisir et tester le premier programme de la préparation avec les valeurs suivantes : 12.3, 13.5, 15.5, 12.7 ,14.6, 9.8, 5.6, 9, 12.8 et 18.9.

Exercice 2 : Saisir le second programme de la préparation. A la fin du programme principal, ajouter une nouvelle boucle permettant d'afficher les valeurs.

```
void main()
{
    // Déclaration des variables
    ...
    // Remplissage du tableau
    ...
    // Affichage
    ...
}
```

Modifier le programme de façon à calculer la moyenne de 10000 valeurs tirées au hasard entre 21 et 71 inclus. Vous veillerez à ne pas initialiser le générateur pseudo aléatoire.

### Exercice 3 :

a. Tester le programme suivant avec la fonction decode de la préparation.

```
int i,Message[]={25,15,24,19,12,0,26,24,0,10,24,15,26,5,2};
for (i=0;i<15;i++)
    printf ("%c",decode(Message[i]));
```

Quel est le message ?

b. Lorsque la fonction est opérationnelle, saisir le programme principal suivant et relever la durée du décodage.

```
void main ()
{
    int i,j,a;
    time_t t;

    // Zone à compléter à la question c.

    t = time(NULL);
    for (i=0;i<100;i++)
    {
        gotoxy(1,1);
        printf ("%d%",i);
        for (j=0;j<2000000;j++)
            a=decode(random(27));
    }
    printf ("\nDurée du décodage : %d secondes",time(NULL)-t);
    getch() ;
}
```

c. Nous allons maintenant utiliser une Look Up Table (LUT) pour remplacer la fonction decode. Il s'agit d'un tableau comportant 27 valeurs de façon à ce que LUT[0]=' ', LUT[1]='d', LUT[2]='e' ... et LUT[26]='c'.

Dans le programme principal, ajouter le tableau aux déclarations des variables sans l'initialiser.

Dans la partie (// Zone à compléter à la question c.) ajouter le remplissage du tableau en utilisant la fonction decode.

Dans les boucles imbriquées, remplacer la ligne `a=decode(random(27));` par `a=LookUpTable[random(27)]`.

Relever la nouvelle durée du décodage et expliquer ce résultat.

Exercice supplémentaire : Ecrire un programme qui tire 100 valeurs au hasard et les mémorise dans un tableau avant de les trier dans l'ordre croissant. Reporter le code sur le document réponse.

## ***Compte-rendu à compléter***

Exercice 1 : Moyenne : \_\_\_\_\_

Exercice 2 : Moyenne : \_\_\_\_\_

Exercice 3 :

a. Message: \_\_\_\_\_

b. Durée du décodage : \_\_\_\_\_ secondes

c. Déclaration du tableau :

Remplissage du tableau à l'aide de la fonction decode :

Durée du décodage : \_\_\_\_\_ secondes

Explication :

Exercice supplémentaire :