

## TP I1101 n°6

### Exercice 1

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Calculer et afficher ensuite la somme des éléments du tableau.

### Exercice 2

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher ensuite le tableau résultant.

### Exercice 3

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.

**Idée:** Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu

### Exercice 4

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Copiez ensuite toutes les composantes strictement positives dans un deuxième tableau TPOS et toutes les valeurs strictement négatives dans un troisième tableau TNEG. Afficher les tableaux TPOS et TNEG.

### Exercice 5

Calculer pour une valeur X donnée du type **float** la valeur numérique d'un polynôme de degré n:  $P(X) = A_n X^n + A_{n-1} X^{n-1} + \dots + A_1 X + A_0$

Les valeurs des coefficients  $A_n, \dots, A_0$  seront entrées au clavier et mémorisées dans un tableau A de type **float** et de dimension n+1.

- Utilisez la fonction **pow()** pour le calcul.
- Utilisez le *schéma de Horner* qui évite les opérations d'exponentiation :

$$\begin{array}{c} A_n \\ \underbrace{\quad} \\ * X + A_{n-1} \\ \underbrace{\quad} \\ * X + A_{n-2} \\ \underbrace{\quad} \\ \dots \\ * X + A_0 \end{array}$$

## TP I1101 n°6

### Exercise 1

Write a program that reads the dimension  $N$  of an array of **int**  $T$  (maximum size: 50 components), fills the array with keyboard input values and displays the array. Then calculate and display the sum of the array elements.

### Exercise 2

Write a program that reads the dimension  $N$  of an array of **int**  $T$  (maximum size: 50 components), fills the array with keyboard input values and displays the table. Then delete all instances of the value 0 in the table  $T$ , pack the remaining elements and print the resulting table.

### Exercise 3

Write a program that reads the dimension  $N$  of an array of **int**  $T$  (maximum size: 50 components), fills the array with keyboard input values and displays the table. Arrange then the array elements  $T$  in reverse order without using aid table, and display the resulting table.

**Hint:** Interchange the array elements using two indexes that run through the table starting respectively at the beginning and end of the table and meet in the middle

### Exercise 4

Write a program that reads the dimension  $N$  of an array of **int**  $T$  (maximum size: 50 components), fills the array with keyboard input values and displays the table. Then copy all strictly positive elements in a second table  $TPOS$  and all strictly negative values in a third table  $TNEG$ . Display tables  $TPOS$  and  $TNEG$ .

### Exercise 5

Calculate, for a given **float** value  $X$ , the numeric value of a polynomial of degree  $n$ :

$$P(X) = A_n X^n + A_{n-1} X^{n-1} + \dots + A_1 X + A_0$$

The values of coefficients  $A_n, \dots, A_0$  will be entered by keyboard and stored in a **float** array  $A$  of dimension  $n + 1$ .

- Use the **pow ()** function to calculate.
- Use the *Horner scheme* that avoids the exponentiation operations:

$$\begin{array}{c} A_n \\ \underbrace{\quad} \\ * X + A_{n-1} \\ \underbrace{\quad} \\ * X + A_{n-2} \\ \underbrace{\quad} \\ \dots \\ * X + A_0 \end{array}$$

# Solution

## Exercice 1

```
#include <stdio.h>
#define N 50
void main()
{
    int n, i, T[N], somme = 0;

    do
    {
        printf("entrer le nombre d'element du tableau : ");
        scanf("%d", &n);
    } while (n <= 0 || n > N);

    printf("remplir le tableau : \n");
```

Saisir d'un nombre valide de taille du tableau qui est entre 0 et le maximum N.

```
    for (i = 0; i < n; i++)
        scanf("%d", &T[i]);

    for (i = 0; i < n; i++)
        somme += T[i];

    for (i = 0; i < n; i++)
        printf("T[%d]=%d ", i, T[i]);

    printf("somme=%d", somme);
}
```

La plupart des opérations sur le tableau sont effectués dans des boucles tel que :

- La saisie de ses éléments ;
- Une Opération demandé ;
- Affichage clair de ses éléments.

Exemple d'exécution :

```
entrer le nombre d'element du tableau : 5
remplir le tableau :
3
2
10
6
1
T[0]=3 T[1]=2 T[2]=10 T[3]=6 T[4]=1 somme=22
```

## Exercice 2

### Méthode 1 :

```
#include <stdio.h>
#define N 50
void main()
{
```

```
    int n, T[N], i, j;
```

```
    do
    {
        printf("entrer le nombre d'element du tableau: ");

        scanf("%d", &n);

    } while (n <= 0 || n>N);
```

Saisir d'un nombre valide de taille du tableau qui est entre 0 et le maximum N.

```
    printf("remplir le tableau :\n");
    for (i = 0; i < n; i++)
    {
        printf("T[%d]= ? = ", i);
        scanf("%d", &T[i]);
    }
```

Remplir les éléments du tableau en les demandant avec ses indices correspondants.

```
    printf("\ntu as remplis un tableau de le forme\n");
    for (i = 0; i < n; i++)
        printf("T[%d]=%d \n", i, T[i]);
```

Affichage de chaque élément avec son indice dans le tableau.

```
    for (i = 0; i < n; i++)
    {
        if (T[i] == 0)
        {
            for (j = i; j < n - 1; j++)
            {
                T[j] = T[j + 1];
            }
            n--;
            i--;
        }
    }
```

- ❖ Effectuant un test sur chaque élément du tableau si elle égale a 0 ;
- ✓ Si oui, tasser le tableau c.à.d. affectons chaque élément par le suivant commençant de i jusqu'à « n-1 » ;
- ✓ A ce fait on perd la valeur de T[i] alors il est indispensable de diminuer le nombre d'élément du tableau ;
- ✓ Mais de même puisque T[i] contient une nouvelle valeur celle qui était T[i+1], un nouveau test sur i est indispensable >> d'où diminuer i par 1.

```

printf("\n apres elimination ton tableau devient: \n");

    for (i = 0; i < n; i++)
    {
        printf("T[%d]=%d \n", i, T[i]);
    }
}

```

Affichage nouveau  
tableau ou le  
résultat.

Exemple d'exécution :

```

entrer le nombre d'element du tableau: 10
remplir le tableau :
T[0]= ? = 3
T[1]= ? = 4
T[2]= ? = 5
T[3]= ? = 0
T[4]= ? = 0
T[5]= ? = 4
T[6]= ? = 0
T[7]= ? = 94
T[8]= ? = 1
T[9]= ? = 8

```

tu as remplis un tableau de le forme

```

T[0]=3
T[1]=4
T[2]=5
T[3]=0
T[4]=0
T[5]=4
T[6]=0
T[7]=94
T[8]=1
T[9]=8

```

apres elimination ton tableau devient:

```

T[0]=3
T[1]=4
T[2]=5
T[3]=4
T[4]=94
T[5]=1
T[6]=8

```

## Méthode 2 (expert)

```
#include <stdio.h>
#define N 50
void main()
{
    int n, T[N], i, k = 0;

    do
    {
        printf("Entrer le nombre d'element du tableau: ");
        scanf("%d", &n);
    } while (n <= 0 || n > N);

    printf("Remplir le tableau :\n");
    for (i = 0; i < n; i++)
        scanf("%d", &T[i]);

    for (i = 0; i < n; i++)
        if (T[i] != 0)
            T[k++] = T[i];

    printf("\n Apres elimination ton tableau
deviant : \n");
    for (i = 0; i < k; i++)
        printf("T[%d]=%d \n", i, T[i]);
}
```

Exemple d'exécution :

Au lieu d'éliminer tout 0 dans le tableau puis son tassement comme la méthode 1 >> remplir le tableau de nouveau par ses éléments différents que 0 en utilisant un variable k tel qu'il est :

- Initialisé par 0
- Incrémenté par 1 après affectation d'un élément différent que 0.

Entrer le nombre d'element du tableau: 10

Remplir le tableau :

3  
4  
5  
0  
0  
4  
0  
4  
94  
1

Apres elimination ton tableau devient :

T[0]=3  
T[1]=4  
T[2]=5  
T[3]=4  
T[4]=4  
T[5]=94  
T[6]=1

## Exercice 3

### Méthode 1

```
#include <stdio.h>
```

```
#define N 50
```

```
void main()
```

```
{
```

```
    int n, i, j, T[N], temp;
```

```
    do
```

```
    {
```

```
        printf("entrer le nombre d'element du tableau : ");
```

```
        scanf("%d", &n);
```

```
    } while (n <= 0 || n > N);
```

```
    printf("remplir le tableau :\n");
```

```
    for (i = 0; i < n; i++)
```

```
    {
```

```
        printf("T[%d]= ? = ", i, T[i]);
```

```
        scanf("%d", &T[i]);
```

```
    }
```

```
    for (i = 0, j = n - 1; i < n / 2; i++, j--)
```

```
    {
```

```
        temp = T[i];
```

```
        T[i] = T[j];
```

```
        T[j] = temp;
```

```
    }
```

```
    printf("\n apres reverse ton tableau devient: \n");
```

```
    for (i = 0; i < n; i++)
```

```
        printf("T[%d]=%d \n", i, T[i]);
```

```
}
```

Boucles  
usuelles du  
tableau.

"swap" les  
elements du  
tableau

### Méthode 2 : (même idée en utilisant un seul indice)

```
#include <stdio.h>
```

```
#define N 50
```

```
void main()
```

```
{
```

```
    int n, i, T[N], temp;
```

```
    do
```

```
    {
```

```
        printf("entrer le nombre d'element du tableau : ");
```

```
        scanf("%d", &n);
```

```
    } while (n <= 0 || n > N);
```

```
    printf("remplir le tableau :\n");
```

```

for (i = 0; i < n; i++)
{
    printf("T[%d]= ? = ", i, T[i]);
    scanf("%d", &T[i]);
}

for (i = 0; i < n / 2; i++)
{
    temp = T[i];
    T[i] = T[n - i - 1];
    T[n - i - 1] = temp;
}
printf("\n apres reverse ton tableau devient: \n");
for (i = 0; i < n; i++)
    printf("T[%d]=%d \n", i, T[i]);
}

```

C'est à trouver alors la relation entre les deux éléments à swaper.

Exemple d'exécution :

```

entrer le nombre d'element du tableau : 10
remplir le tableau :
T[0]= ? = 1
T[1]= ? = 2
T[2]= ? = 3
T[3]= ? = 4
T[4]= ? = 5
T[5]= ? = 6
T[6]= ? = 7
T[7]= ? = 8
T[8]= ? = 9
T[9]= ? = 10

apres reverse ton tableau devient:
T[0]=10
T[1]=9
T[2]=8
T[3]=7
T[4]=6
T[5]=5
T[6]=4
T[7]=3
T[8]=2
T[9]=1

```

---



## Exercice 4

```
#include <stdio.h>
#define N 50
void main()
{
    int n, i, T[N], TPOS[N], TNEG[N], P = 0, Ne = 0; } (*Voir note↓)

    do
    {
        printf("Entrer le nombre d'element du tableau : ");
        scanf("%d", &n);
    } while (n <= 0 || n>N);

    printf("Remplir le tableau :\n");

    for (i = 0; i < n; i++)
    {
        scanf("%d", &T[i]);
        if (T[i] != 0)
            if (T[i] > 0)
                TPOS[P++] = T[i];
            else
                TNEG[Ne++] = T[i];
    }

    printf("\nTu as remplis un tableau de le forme\n");
    for (i = 0; i < n; i++)
        printf("T[%d]=%d \n", i, T[i]);

    printf("\n Le tableau strict positive deviant :\n");
    for (i = 0; i < P; i++)
        printf("T[%d]=%d \n", i, TPOS[i]);

    printf("\n Le tableau strict negative deviant :\n");
    for (i = 0; i < Ne; i++)
        printf("T[%d]=%d \n", i, TNEG[i]);
}
```

Parfois on est capable d'effectuer notre opération demande en même temps lors de saisir des éléments comme dans ce cas.

Affichage des tableaux formées.

### ➡ \*Note :

- ❖ Pour remplir un nouveau tableau qui a un indice indépendant des autres tableaux, à ne pas oublier que cette indice doit être initialisé et souvent par 0.
- ❖ Puis après l'affectation attendue ou demandé, à ne pas oublier d'incrémenter cet variable d'indice (variable++) ;
- ❖ Exemple ici on a « P » indice de remplissage pour « TPOS » :
  - TPOS[P++] = T[i] ; (même que TPOS[P] = T[i] ; P = P + 1 ;)

Exemple d'exécution :

Entrer le nombre d'element du tableau : 10

Remplir le tableau :

1034

34

0

5

-9

0

5

-4

-10

16

Tu as remplis un tableau de le forme

T[0]=1034

T[1]=34

T[2]=0

T[3]=5

T[4]=-9

T[5]=0

T[6]=5

T[7]=-4

T[8]=-10

T[9]=16

Le tableau strict positive devient :

T[0]=1034

T[1]=34

T[2]=5

T[3]=5

T[4]=16

Le tableau strict negative devient :

T[0]=-9

T[1]=-4

T[2]=-10

## Exercice 5

```
#include <stdio.h>
#include <math.h>
#define N 50
```

### Partie a:

```
void main()
{
    int n, i;
    float x, T[N], resultat = 0;

    do
    {
        printf("donner ton degre n \n"); scanf("%d", &n);
    } while (n <= 0 || n > N);
    n = n + 1;
    printf("donner les elements (n+1) inconnus coef croissant d'ordre 0 >> n \n");
    for (i = 0; i < n; i++)
    {
        printf("coefficient A%d:", i);
        scanf("%f", &T[i]);
    }

    printf(" give me x for calculation !!\n"); scanf("%f", &x);
    for (i = 0; i < n; i++)
    {
        resultat = resultat + T[i] * powf(x, i);
    }

    printf("le finale resultat=%f", resultat);
}
```

Calculant le résultat initialisé par 0 en y ajoutant la valeur de chaque  $A_i(x)^i$ .

Exemple d'exécution :

```
donner ton degre n
5
donner les elements (n+1) inconnus coef croissant d'ordre 0 >> n
coefficient A0:13
coefficient A1:2
coefficient A2:3
coefficient A3:4
coefficient A4:5
coefficient A5:6
 give me x for calculation !!
3
le finale resultat=2017.000000
```

### **Parite b :**

```
#include <stdio.h>
#define N 50

void main()
{
    int n, i;
    float x, T[N], resultat = 0, temp = 0;
    do
    {
        printf("donner ton degre n \n"); scanf("%d", &n);
    } while (n <= 0 || n > N);
    n = n + 1;
    printf("donner les elements (n+1) inconnus coef croissant d'ordre 0 >> n \n");
    for (i = 0; i < n; i++)
    {
        printf("coefficient A%d:", i);
        scanf("%f", &T[i]);
    }

    printf(" give me x for calculation !!\n"); scanf("%f", &x);
    for (i = n - 1; i >= 0; i--)
    {
        resultat = resultat*x + T[i];
    }
    printf("le finale resultat=%f",resultat);
}
```

Exemple d'exécution :

```
donner ton degre n
5
donner les elements (n+1) inconnus coef croissant d'ordre 0 >> n
coefficient A0:13
coefficient A1:2
coefficient A2:3
coefficient A3:4
coefficient A4:5
coefficient A5:6
 give me x for calculation !!
3
le finale resultat=2017.000000
```

---

---