# Chapitre II

# Tableaux, Pointeurs, Fonctions, et Structures en C++

## I.1. Introduction

Dans ce chapitre, nous allons étudier les tableaux, les pointeurs, et les fonctions en C++. La première partie de ce chapitre sera consacrée aux différents types de tableaux statiques et dynamiques. Dans la deuxième partie, nous allons étudier les pointeurs et leur avantage d'utilisation avec les tableaux et pour l'allocation dynamique de la mémoire. Des exemples avec solutions seront présentés après chaque nouvelle notion étudiée pour faciliter la compréhension et acquérir rapidement les compétences visée.

## I.2. Tableaux statiques unidimensionnels

Un tableau uni-dimensionnel est une suite limitée de case mémoire contenant des éléments simples et de même type (int, char, float, double...). Ces éléments sont appelés composants du tableau et leur nombre représente la taille du tableau.

### I.2.1 Syntaxe de la déclaration d'un tableau statique unidimensionnel

La syntaxe de déclaration des tableaux unidimensionnels est la suivante :

type nom [taille]= {valeurs initiales} ;

*Exemple :*

```
int age [5]={15, 20, 70, 11,18} ; // tableau appelé 'age' de type 'int' et de taille 5. Les cinq
                                  // composantes sont initialisées respectivement par 15, 20…18.
float moy [5]; //tableau appelé 'moy' de type 'float', de taille 5, et des composantes non initialisées.
```

Pour accéder aux composantes du tableau, il suffit d'indiquer le nom du tableau suivi de l'indice de la composante, par exemple :

```
age[0] ;    // pour accéder à la première composantes (15).
age[4] ; // pour accéder à la dernière composantes (18).
```

*Remarque importante :*

- Le nom d'un tableau représente l'adresse du premier élément du tableau.
- l'accès à la première composante du tableau se fait par nom [0] et l'accès à la dernière composante se fait par nom [taille -1].
- Le tableau occupe un espace mémoire égale : taille du tableau*nombre d'octets constituant chaque composante. Ainsi le tableau '*age*' occupe 10 octets et le tableau '*moy*' occupe 20 octets.

### I.2.2 Parcourir un tableau unidimensionnel

L'un des avantages les plus importants des tableaux, c'est qu'on peut utiliser les boucles pour lire, afficher, initialiser ou modifier chaque composante séparément des autres.

*Exemple N°1*

Le programme suivant affiche les composantes d'un tableau en utilisant la boucle for.

```
#include <iostream>
using namespace std;

int main()
{
 int const taille (5);//taille du tableau,il est très recommandé
                      //de la déclarer comme constante

 int age [taille]={15,20,70,11,18}; /* Déclaration d'un tableau
 /    unidimensionnel de type int contient cinq valeurs entières*/
 for(int i=0;i<taille;i++)
 {
  cout<<"age["<<i<<"]=\t"<<age[i]<<endl;/* affichage des composants
  du tableau 'age' l'un après l'autre*/
 }
return 0;
}
```

L'exécution du programme ci-dessus donne le résultat suivant :

```
age[0]= 15
age[1]= 20
age[2]= 70
age[3]= 11
age[4]= 18

Process returned 0 (0x0)   execution time : 0.093 s
Press any key to continue.
```

*Exemple N°2*

Le programme suivant lit 05 valeurs entières et les stocker dans un tableau.

```
#include <iostream>
using namespace std;

int main()
{
 int const taille (5);//taille du tableau

 int tab [taille];// Déclaration d'un tableau unidimensionnel
                  //  de cinq (05) composants (éléments)
 int i(0);//indice
 int size_tab (taille);/*Stocker la taille de tab dans une variable
                     pour qu'on puisse l'incémenter ou la décrémenter*/
 do
 {
  cout<<" Entrer la valeur du tab["<<i<<"]= ?\t ";
  cin >> tab[i];
  i++;
  size_tab--;
  }
  while (size_tab>0);
return 0;
}
```

*Exemple d'exécution*:

```
"C:\Users\Malik\Desktop\Programmation OrientÚe Objet\Exemples_chapitre2\ex02\bin\Debug\ex02.exe"
Entrer la valeur du tab[0]= ?   12
Entrer la valeur du tab[1]= ?   24
Entrer la valeur du tab[2]= ?   85
Entrer la valeur du tab[3]= ?   32
Entrer la valeur du tab[4]= ?   47

Process returned 0 (0x0)   execution time : 12.216 s
Press any key to continue.
```

*Exercice* :

Ecrire un programme qui remplit les (composants) d'un tableau de taille sept (07) par des valeurs entrées au clavier et affiche ensuite:

- Le tableau rempli ;
- La somme des composantes du tableau ;
- L'indice et la valeur de la composante la plus grande ;
- L'indice et la valeur de la composante la plus petite ;

*Solution :*

```
#include <iostream>
using namespace std;
int main ()
{
 int const taille (4);//taille du tableau
 int max_tab(0),idice_max(0), min_tab(0),idice_min(0);
 long int som_Tab(0);// variable à utiliser pour stocker
                     // la somme des composants du tableau

 float Tab[taille]; //Déclaration du tableau

 //Remplissage du tableau
 cout<<"Entrer la valeur du Tab[0]= ?\t";
 cin >> Tab[0];
 max_tab=Tab[0]; idice_max=0;
 min_tab=Tab[0]; idice_min=0;
 som_Tab+=Tab[0];
 for(int i=1;i<taille;i++)
 {
  cout<<"Entrer la valeur du Tab["<<i<<"]= ?\t";
  cin >> Tab[i];
  som_Tab +=Tab[i];//Calcul la somme des composants du tableau
  //chercher la composante dont la valeur la plus grande et son indice
  if (Tab[i]>max_tab)
  {
   max_tab=Tab[i];
   idice_max=i;
  }
  //chercher la composante dont la valeur la plus petite et son indice
  if (Tab[i]<min_tab)
  {
   min_tab=Tab[i];
   idice_min=i;
  }
 }
  //Affichage du tbleau
  for(int i=0;i<taille;i++)
```

```
38      {
39       cout<<"Tab["<<i<<"]=\t"<<Tab[i]<<endl;
40      }
41      cout<<"la somme des composantes du tableau= "<<som_Tab<<endl;
42      cout<<"la valeur la plus grande= Tab["<<idice_max<<"]= "<<max_tab<<endl;
43      cout<<"la valeur la plus faible= Tab["<<idice_min<<"]= "<<min_tab<<endl;
44        return 0;
45  }
```

*Exemple d'exécution*:

```
Entrer la valeur du Tab[0]= ?   14
Entrer la valeur du Tab[1]= ?   -9
Entrer la valeur du Tab[2]= ?   84
Entrer la valeur du Tab[3]= ?   20
Tab[0]= 14
Tab[1]= -9
Tab[2]= 84
Tab[3]= 20
la somme des composantes du tableau= 109
la valeur la plus grande= Tab[2]= 84
la valeur la plus faible= Tab[1]= -9
```

## I.3. Tableaux multidimensionnels

Un tableau multidimensionnel est un tableau de tableau ; c.à.d. un tableau **bidimensionnel** est un tableau unidimensionnel d'un tableau unidimensionnel et un tableau **tridimensionnel** est un tableau unidimensionnel d'un tableau bidimensionnel… etc.

### I.3.1 Syntaxe de la déclaration des tableaux multidimensionnels

La syntaxe de la déclaration des tableaux multidimensionnels est similaire à celle des tableaux unidimensionnel sauf qu'on ajout la taille pour chaque dimension.

*Exemple :*

```
float tab1[10][5] ; // déclaration d'un tableau bidimensionnel (10 x 5= 50 composants)
double tab2[10][5][7] ; // déclaration d'un tableau tridimensionnel (10x5x7= 350 composants)
```

Pour accéder aux composantes du tableau multidimensionnel, il suffit d'indiquer le nom du tableau suivi des indices du composant. Par exemple:

```
tab1[1][2] ;        // Pour accéder au composant identifié par les indices (1,2)
tab2[0][1][3] ; // Pour accéder au composant identifié par les indices (0,1,3).
tab2[1][5][0]=2018 ;// Affecter la valeur 2018 au composant identifié par les indices (1,5,0)
```

*Remarque:*

Le tableau de n dimensions occupe un espace mémoire égale : taille de la 1ère dimension * taille de la 2ème dimension*….. *taille de la nième dimension *nombre d'octets constituant chaque composante. Ainsi, si la variable du type float occupe 4 octets et le type double occupe 8 octets le tableau *tab1* occupe 200 octets et le tableau *tab2* occupe 2800 octets.

### I.3.2 Parcourir un tableau multidimensionnel

Pour parcourir le contenu d'un tableau de *n* dimensions, on a besoin à *n* indices et une boucle imbriquée *n* fois.

*Exemple N°1*

Le programme suivant affiche les composantes d'un tableau à deux dimensions ainsi que leur somme.

```
#include <iostream>
using namespace std;

int main()
{

    long int somme(0);
//Déclaration et initialisation d'un tableau à deux dimensions
    int Tabl[4][4]={
                    {1,2,3,4},    // première ligne
                    {2,4,6,8},    // deuxième ligne
                    {4,8,12,18}, // troisième ligne
                    {8,16,24,36} // quatrième ligne
                    };
//Affichage du tableau
for(int i=0; i<4;i++)//pour parcourir les 4 lignes
{
    for(int j=0; j<4;j++)//pour parcourir les 4 colonnes
    {
     cout <<Tabl[i][j]<<"\t";
     somme+=Tabl[i][j];//calcul de la somme des composantes du tableau
    }
    cout <<"\n";
}
    cout <<"la somme de toutes les composante du tableau= "<<somme<<endl;
    return 0;
}
```

L'exécution du programme ci-dessus donne le résultat suivant :

```
"C:\Users\Malik\Desktop\Programmation OrientÚe Objet\Exemple_Prog\Exemples_chapitre2\Tamblau_2dimensions"
1       2       3       4
2       4       6       8
4       8       12      18
8       16      24      36
la somme de toutes les composante du tableau= 156
```

*Exemple N°2*

Le programme suivant :

- rempli d'un tableau bidimensionnel par des valeurs entrées au clavier ;
- affiche le tableau et la somme de chaque ligne.

```
#include <iostream>
using namespace std;

int main()
{
 long int somme(0);
 int const L(3),C(3);//dimensions du tableau
 int  i(0),j(0);//indices
 int Tab[L][C];//Déclaration du tableau bidimensionnel
```

```
10
11  //Lecture des composantes du tableau 'Tab'
12  for(int i=0; i<L; i++)//pour parcourir les 4 lignes
13  {
14   for(int j=0; j<C;j++)//pour parcourir les 4 colonne
15   {
16    cout <<"Tab["<<i<<"]["<<j<<"]= ?"<<"\t";
17    cin>> Tab[i][j];
18   }
19  }
20
21  //Affichage du tableau la somme de chaque ligne
22  for(i=0; i<L; i++)
23  {
24   somme=0;
25   for(j=0; j<C;j++)
26   {
27    cout <<Tab[i][j]<<"\t";
28    somme +=Tab[i][j]; //calcul de la somme de ligne i
29   }
30   cout << " La somme des composantes de la ligne " <<i+1<< "=\t"<< somme<·
31  }
32  return 0;
33 }
```

*Exemple d'exécution*:

```
Tab[0][0]= ?    14
Tab[0][1]= ?    2
Tab[0][2]= ?    1
Tab[1][0]= ?    12
Tab[1][1]= ?    -2
Tab[1][2]= ?    4
Tab[2][0]= ?    0
Tab[2][1]= ?    1
Tab[2][2]= ?    15
14      2       1       La somme des composantes de la ligne 1=        17
12      -2      4       La somme des composantes de la ligne 2=        14
0       1       15      La somme des composantes de la ligne 3=        16
```

## I.4. Tableaux dynamiques

Un tableau dynamique est une suite variable de case mémoire (taille variable), i.e. on peut varier la taille du tableau en ajoutant ou en supprimant des cases mémoires.

### I.4.1 Syntaxe de la déclaration d'un tableau dynamique

La déclaration des tableaux dynamiques est un peu différente à celle des tableaux statiques. Par exemple, on doit écrire tous d'abord le mot '**vector'** suivi ensuite par le **type entre deux chevrons**, ensuite **le nom du tableau** et enfin la **taille du tableau <u>entre deux parenthèses</u>**

Syntaxe: **vector** <**type**> **Nom** (**Taille**).

*Exemples :*

vector <int> age (5) ; // Déclaration d'un tableau dynamique unidimensionnel de type *int* de
// taille égale à 5. Le tableau 'age' est non initialisé.

vector <int> annee (5,2018) ; // annee est un tableau dynamique dont toutes ses composantes sont
//initialisées par 2019, i.e. annee contient {2018, 2018, 2018, 2018, 2018}

vector <string> mois (3, "Novembre") ; /* mois est un tableau dynamique de type *string* dont toutes