Programmation C++ (débutant)/Les tableaux statiques

Le cours du chapitre 6 : les tableaux statiques

Les tableaux

Une variable entière de type int ne peut contenir qu'une seule valeur. Si on veut stocker en mémoire un ensemble de valeurs, il faut utiliser une structure de données appelée tableau.
Dans ce chapitre, les tableaux seront statiques : leur taille sera fixée une fois pour toute. Il s'agit d'une structure de donnée absolument fondamentale pour stocker une liste d'éléments.

Déclaration d'un tableau statique

Syntaxe : `type identificateur[taille];`
Un tableau sera constitué d'un ensemble de cases. Chaque case comportera un élément dont le type sera type. Le nom du tableau sera identificateur. Le nombre total de cases du tableau sera taille. Cette variable sera obligatoirement une constante.

Un exemple de tableau

Déclaration d'un tableau : int a[10];
a est un tableau de 10 cases. Chaque case contient un entier (type int). La première case s'appelle a[0]. La deuxième case s'appelle a[1] et la dixième case a[9].
Remarque : la case a[10] n'existe pas ! Car la première case possède l'indice 0. Si vous essayez d'accéder à une case dont l'indice n'est pas valide, le programme "plantera".
Sur chaque case, on peut effectuer les opérations habituelles : addition, affectation…etc…

Exemple 1 : utilisation d'un tableau

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

int main()
{
    int t[10], i;
    for(i=0; i<10; i++)
    {
        cout << "Tapez la valeur numero " << i << " : ";
        cin >> t[i];
    }
    for(i=0; i<10; i++) t[i] = t[i]+1;
    for(i=0; i<10; i++) cout << "La valeur numero " << i <<" est : "<< t[i] <<endl;
    return 0;
}
```

- **Explications**
  - Dans ce programme, nous allons tout d'abord saisir une à une le contenu des 10 cases d'un tableau t.

- Ensuite nous allons effectuer un traitement simple sur ce tableau : nous allons incrémenter de 1 le contenu de chaque case.
- Finalement, nous afficherons le contenu final de chaque case du tableau.
- Dans ce programme, nous commençons par définir un tableau t de 10 cases de type entier. La première case de ce tableau sera t[0],… et la dernière t[9].
- La première boucle for permet de saisir une à une les cases du tableau : for(i=0;i<10;i++) { cout<<"Tapez la valeur numero "<<i<<" : "; cin>>t[i]; } remarque : la première valeur de i pour laquelle le corps du for sera effectué sera i=0, la dernière i=9
- for(i=0;i<10;i++)t[i]=t[i]+1;
  Dans cette boucle for, on augmente de 1 le contenu de chaque case du tableau.
- for(i=0;i<10;i++) cout<<"La valeur numero "<<i<<" est : " <<t[i]<<endl;
  On affiche une à une le contenu des cases du tableau.

- **Exécution**

Tapez la valeur numero 0 : 5
Tapez la valeur numero 1 : 2
Tapez la valeur numero 2 : 50
Tapez la valeur numero 3 : 10
Tapez la valeur numero 4 : 20
Tapez la valeur numero 5 : 60
Tapez la valeur numero 6 : 80
Tapez la valeur numero 7 : 90
Tapez la valeur numero 8 : 10
Tapez la valeur numero 9 : 10
La valeur numero 0 est 6
La valeur numero 1 est 3
La valeur numero 2 est 51
La valeur numero 3 est 11
La valeur numero 4 est 21
La valeur numero 5 est 61
La valeur numero 6 est 81
La valeur numero 7 est 91
La valeur numero 8 est 11
La valeur numero 9 est 11

La taille d'un tableau est une CONSTANTE

La taille d'un tel tableau est obligatoirement constante et connue à la compilation du programme. On parle de tableau statique.
Interdiction de taper : int i; cin>>i; int t[i];
Lorsque nous aurons étudié les pointeurs et le mot clé new (dans quelques chapitres), nous pourrons créer des tableaux dont la taille est variable. Patience !

Déclaration et initialisation d'un tableau

On peut déclarer et initialiser un tableau de la manière suivante : int t[]={8,7,6,4,8};
Le tableau sera toujours un tableau de taille fixe, ici un tableau à 5 cases. Le tableau t n'est pas un tableau de taille variable.
Cette syntaxe évite d'écrire : int t[5]; t[0]=8; t[1]=7; t[2]=6; t[3]=4; t[4]=8;

Exemple 2 : un tableau de double

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

int main()
{
    double t[4];
    int i;
    for(i=0; i<4; i++)
        t[i] = 1.0 / (i+1);
    for(i=0; i<4; i++) cout<<"La valeur numéro "<<i<<" est : " << t[i] <<endl;
    return 0;
}
```

- **Explications**
  - On peut définir un tableau de n'importe quel type : ici un tableau de double. Dans ce programme, nous définissons un tableau t de 4 double.
  - Nous remplissons ensuite ce tableau en mettant 1/(i+1) dans la case numéro i.
  - Nous affichons ensuite une à une le contenu de chaque case.
- **Exécution**

La valeur numero 0 est 1
La valeur numero 1 est 0.5
La valeur numero 2 est 0.33333
La valeur numero 3 est 0.25

Indice d'un élément dans un tableau

Le contenu d'un tableau peut être un int, un double … Ce type est défini lors de la déclaration du tableau et ne peut pas changer. L'indice d'un élément d'un tableau est lui obligatoirement de type entier.
Par exemple, il est interdit d'écrire :

```
double a;
double b[10];
a=8;
b[a]=123;
```

Traitements sur les tableaux

Nous allons maintenant étudier différents algorithmes standards qu'il faut savoir effectuer sur des tableaux : calcul de la moyenne, recherche du plus petit élément,…
Ces traitements et recherche en tout genre sont des algorithmes indispensables que doit connaître tout programmeur.

Calcul de la moyenne

- On suppose que des éléments sont stockés dans un tableau contenant 4 cases : on veut calculer la moyenne des éléments de ce tableau.
- **Algorithme utilisé :**

On initialise une variable s à 0. Il faudra parcourir le tableau et ajouter à chaque étape l'élément courrant du tableau à s. On divisera ensuite s par le nombre d'éléments du tableau. Il faut maintenant formaliser cet algorithme.

Exemple 3 : calcul de la moyenne

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

int main()
{
    int t[4], i;
    double s=0;
    for(i=0; i<4; i++)
    {
        cout << "Tapez la valeur numéro " << i << " : ";
        cin >> t[i];
       s = s + t[i];
    }

    s = s/4;
    cout << "La moyenne est : " << s << endl;
    return 0;
}
```

- **Explications**
  - Dans cet exemple, on déclare un tableau t de 4 cases (la première sera t[0], la dernière t[3]).
  - Grâce à une boucle for, on saisit une à une le contenu des 4 cases du tableau.
    for(i=0;i<4;i++) { cout<<"Tapez la valeur numéro "<<i<<" : "; cin>>t[i]; }
  - Pour calculer la somme des éléments d'un tableau, on initialise s à 0 et à chaque étape on ajoute t[i] à s grâce à la boucle for suivante :

for(i=0;i<4;i++)s=s+t[i];
A la fin de cette boucle, la variable s contient la somme des éléments du tableau.
Remarque 1 : un bug courant dans le calcul d'une somme est d'oublier d'initialiser s à 0 avant la boucle.
Remarque 2 : un autre bug courant est de se tromper dans les bornes de la boucle for.

- **Exécution**

Tapez la valeur numero 0 : 5
Tapez la valeur numero 1 : 2
Tapez la valeur numero 2 : 50

Tapez la valeur numero 3 : 10
La moyenne est 16.75

Recherche dans un tableau

On veut écrire un programme qui recherche le plus petit élément dans un tableau contenant 4 cases.
Algorithme utilisé : on va stocker notre plus petit élément dans une variable ppt. On commence par initialiser ppt au premier élément du tableau (d'indice zéro). On parcourt alors tous les autres éléments du tableau en comparant l'élément courant à ppt et en mettant éventuellement à jour la valeur de ppt. Il faut maintenant formaliser cet algorithme.

Exemple 4

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

int main()
{
    int t[4], i, ppt;
    for(i=0; i<4; i++)
    {
        cout << "Tapez la valeur numéro " << i << " : ";
        cin >> t[i];
    }
    ppt = t[0];
    for(i=1; i<4; i++) if(ppt>t[i]) ppt=t[i];
    cout << "La plus petite valeur est "<< ppt <<endl;
    return 0;
}
```

- **Explications**
  - Pour calculer le plus petit élement d'un tableau, on initialise ppt à t[0].
  - On parcourt ensuite le tableau de la case 1 à la dernière case d'indice 3 en comparant t[i] à ppt. Si t[i] est plus petit que le plus petit courant ppt alors on copie t[i] dans ppt (t[i] devient alors le nouveau plus petit courant). Dans le cas contraire, on ne modifie pas la valeur de ppt.
  - On affiche finalement la valeur de ppt en dehors de la boucle.
- **Exécution**

Tapez la valeur numero 0 : 9
Tapez la valeur numero 1 : 7
Tapez la valeur numero 2 : 11
Tapez la valeur numero 3 : 15
La plus petite valeur est 7

Exemple 5 : inverser l'ordre des éléments d'un tableau

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

int main()
{
    int t[6], i, a;

    for(i=0; i<6; i++)
    {
        cout << "Tapez la valeur numéro " << i << " : ";
        cin >> t[i];
    }

    for(i=0; i<3; i++)
    {
        a = t[i];
        t[i] = t[5-i];
        t[5-i] = a;
    }

    for(i=0; i<6; i++)
        cout << "La valeur numéro " << i << " est " << t[i] << endl;

    return 0;
}
```

- **Explications**
  - Pour inverser le contenu du tableau t à 6 cases il y a 3 étapes :

Étape 0 : on échange t[0] et t[5]
Étape 1 : on échange t[1] et t[4]
Étape 2 : on échange t[2] et t[3]

- A l'étape i, on échange t[i] et t[5-i].
- Chaque étape sera numérotée de i=0 à i=2.
- Attention de ne pas échanger 2 fois le contenu de chaque case en écrivant par erreur :

for(i=0; i<6; i++){ a=t[i]; t[i]=t[5-i]; t[5-i]=a;}

- **Exécution**

Tapez la valeur numero 0 : **9**
Tapez la valeur numero 1 : **7**
Tapez la valeur numero 2 : **11**
Tapez la valeur numero 3 : **15**
Tapez la valeur numero 4 : **16**
Tapez la valeur numero 5 : **4**
La valeur numero 0 est 4
La valeur numero 1 est 16
La valeur numero 2 est 15

La valeur numero 3 est 11
La valeur numero 4 est 7
La valeur numero 5 est 9

Suppression et tassement

On veut supprimer toutes les valeurs valant 9 dans un tableau de 6 cases en décalant tous les élément vers la gauche et en remplaçant ces éléments par des 0 placés à la fin du tableau.
Exemple :
Valeur initiale du tableau : 9, 8, 9, 9, 9, 6
Valeur finale du tableau : 8, 6, 0, 0, 0, 0
Ce problème paraît simple mais on trouve tellement de solutions fausses à ce problème que, désespérés, nous avons fini par le mettre dans le cours ! Par exemple, il est complètement inutile d'écrire 2 boucles imbriquées.

Exemple 6 : suppression et tassement

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

int main()
{
    int t[6], i, j = 0;

    for (i=0 ; i<6 ; i++)
    {
        cout << "Tapez la valeur numéro " << i << " : ";
        cin >> t[i];
    }

    for(i=0 ; i<6 ; i++) if (t[i] != 9) { t[j] = t[i]; j++; }
    for(i=j ; i<6 ; i++) t[i] = 0;

    for(i=0 ; i<6 ; i++) cout << "La valeur numéro " << i << " est " << t[i] << endl;
    return 0;
}
```

Explications

1. Dans cet exemple, on commence par saisir chacun des 6 éléments du tableau.

```
for(i=0;i<6;i++) { cout<<"Tapez la valeur numéro "<<i<<" : "; cin>>t[i]; }
```

2. On va utiliser un indice j qui sera l'indice dans le tableau où doit être mis le prochain élément différent de 9. Cet élément est initialisé à 0.
3. On parcourt une seule fois le tableau et on place les valeurs différentes de 9 dans la case numéro j du tableau en incrémentant à chaque fois j de 1.

```
for(i=0 ; i<6 ; i++) if (t[i] != 9) { t[j] = t[i]; j++; }
```

4. On rajoute ensuite des 0 à partir de la case numéro j jusqu'à la fin du tableau.

```
for(i=j ; i<6 ; i++) t[i] = 0;
```

Les constantes

Imaginons que dans un programme, on utilise un tableau de 100 cases et que l'on effectue des traitements en tout genre sur ce tableau. Il y aura des tonnes d'endroits dans le programme (au niveau des boucles notamment) où il y aura des valeurs valant 100 (ou 99). Si on décide de changer la taille de tableau et de la faire passer à 200 : il faudra faire des tonnes de modifications dans notre programme notamment au niveau des indices des boucles, avec à chaque fois un risque d'erreur non négligeable.

Une solution : les constantes

On va définir une constante N qui vaudra 100 et on va écrire tout notre programme, notamment tous nos indices de boucles en fonction de N.
Si on veut changer la taille de tableau, il suffira de changer la valeur de N à un seul endroit dans le programme.

Deux syntaxes pour les constantes

Il y a 2 syntaxes pour définir des constantes : l'une, la plus ancienne est d'utiliser la directive de compilation #define, l'autre est de définir une variable globale constante. On préférera utiliser cette deuxième solution mais les deux sont à connaître.
Première syntaxe (version C)
#define nom valeur
Deuxième syntaxe (version C++)
const type identificateur=valeur_constante;

Exemple 7 : les constantes

```
#include <iostream>
using namespace std;
#define N 10

int main()
{
    int t[N], i;
    for (i=0; i<N; i++) t[i] = i*i;
    for (i=0; i<N; i++) cout<< t[i] <<endl;
    return 0;
}
```

- **Explications**
  - On définit une macro N valant toujours 10.
  - Avant la compilation, toutes les occurrences de N seront remplacées par sa valeur c'est-à-dire 10.
  - Tous nos indices de boucles seront calculés en fonction de N.
  - Dans ce programme, on déclare un tableau d'entiers à N cases (N est une constante valant 10).
  - On met ensuite dans la case i du tableau la valeur i*i et on affiche finalement toutes les cases du tableau.
- **Exécution**

0
1
4

```
9
16
25
36
49
64
81
```

Exemple 8 : les constantes

```
#include <iostream>
using namespace std;
const int N=10;

int main()
{
    int t[N], i;
    for(i=0; i<N; i++) t[i] = i*i;
    for(i=0; i<N; i++) cout<< t[i] <<endl;
    return 0;
}
```

- **Explications**
  - Cet exemple est absolument identique au précédent à la différence que N est cette fois-ci une variable globale constante de type entier valant 10. On préférera utiliser cette solution que d'utiliser #define.
  - Dans un programme en C++, il est déconseillé d'utiliser des variables globales qui ne sont pas constantes : cela nuit en général gravement à la structuration de l'application.

Tableau à 2 dimensions

Si on veut déclarer un tableau de 5 lignes et 4 colonnes, il faut déclarer :
int a[5][4];
On accède alors à l'élément ligne i colonne j de la manière suivante : a[i][j]=99;
Dans cet exemple, i doit être compris entre 0 et 4 (bornes incluses) et j entre 0 et 3 (bornes incluses).

Exemple 9 : un tableau à 2 dimensions

```
#include <iostream>
using namespace std;
const int N = 2;
const int M = 3;

int main()
{
    int i, j;
    int t[N][M];

    for(i=0; i<N; i++)
      for(j=0; j<M; j++)
      {
```

```
        cout<<"Tapez t["<< i <<"]["<< j <<"] :";
        cin >> t[i][j];
      }

    cout<<"Voici le tableau :"<<endl;
    for(i=0; i<N; i++)
    {
      for(j=0; j<M; j++) cout<< t[i][j] <<" ";
      cout<<endl;
    }
    return 0;
}
```

- **Explications**
  - Dans cet exemple, on déclare un tableau t d'entiers comportant 2 lignes et 3 colonnes.
  - Par 2 boucles imbriquées, on saisit un à un les 6 éléments du tableau.
  - On affiche ensuite le contenu du tableau.
- **Exécution**

Tapez t[0][0] : 3
Tapez t[0][1] : 4
Tapez t[0][2] : 5
Tapez t[1][0] : 6
Tapez t[1][1] : 8
Tapez t[1][2] : 4
Voici le tableau :
3 4 5
6 8 4

Et maintenant …

Nous avons étudié les types de base, les structures de contrôle et les tableaux, il faut maintenant effectuer des exercices pour s'entraîner à manipuler des tableaux. Ces manipulations pourront être l'occasion de mettre en oeuvre des algorithmes classiques de recherche, de suppression de tri …etc…Nous en avons étudié quelques-uns en cours et il en reste des tonnes à étudier. A vous de jouer !

Exercices

EXERCICE 1

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit afficher le nombre d'entiers supérieurs ou égaux à 10.

EXERCICE 2

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et afficher "V se trouve dans le tableau" ou "V ne se trouve pas dans le tableau".

EXERCICE 3

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit ensuite afficher l'indice du plus grand élément.

EXERCICE 4

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et doit supprimer la première occurence de V en décalant d'une case vers la gauche les éléments suivants et en rajoutant un 0 à la fin du tableau. Le programme doit ensuite afficher le tableau final.

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V et un entier i compris entre 0 et 9. Le programme doit décaler d'une case vers la droite tous les élements à partir de l'indice i (en supprimant le dernier élément du tableau) et doit mettre la valeur V dans le tableau à l'indice i. Le programme doit ensuite afficher le tableau final.

EXERCICE 6

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit ensuite afficher soit "le tableau est croissant", soit "le tableau est décroissant", soit "le tableau est constant", soit "le tableau est quelconque".

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.

Algorithme suggéré :

On cherche l'indice du plus petit élément parmi les indices de 0 à 9 et on échange cet élément avec t[0].
On cherche l'indice du plus petit élément parmi les indices de 1 à 9 et on échange cet élément avec t[1].
On cherche l'indice du plus petit élément parmi les indices de 2 à 9 et on échange cet élément avec t[2].
On cherche l'indice du plus petit élément parmi les indices de 8 à 9 et on échange cet élément avec t[8].

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.

Algorithme suggéré (tri bulle) :

On parcourt le tableau en comparant t[0] et t[1] et en échangeant ces éléments s'ils ne sont pas dans le bon ordre.
on recommence le processus en comparant t[1] et t[2],... et ainsi de suite jusqu'à t[8] et t[9].
On compte lors de ce parcours le nombre d'échanges effectués.
On fait autant de parcours que nécessaire jusqu'à ce que le nombre d'échanges soit nul : le tableau sera alors trié.

EXERCICE 9

Ecrire un programme qui saisit 2 tableaux de 10 entiers a et b. c est un tableau de 20 entiers. Le programme doit mettre dans c la fusion des tableaux a et b. On copiera dans les 10 premières cases de c le tableau a, dans les dix dernières le tableau b. Le programme affiche ensuite le tableau c.

EXERCICE 10

Ecrire un programme qui saisit 2 tableaux de 10 entiers a et b qui doivent être triés dans l'ordre croissant. Le programme devra tout d'abord vérifier que les deux tableaux sont triés. Le tableau c est un tableau de 20 entiers. Le programme doit mettre dans c la fusion des tableaux a et b. Le tableau c devra contenir les éléments de a et ceux de b et devra être trié. Le programme affiche ensuite le tableau c.

EXERCICE 11

Ecrire un programme qui gère une liste d'entiers grâce au menu suivant :
1. Ajouter un entier
2. Afficher la liste des entiers
3. Supprimer le dernier entier de la liste.
4. Afficher la dernière note tapée
5. Quitter

Il y aura au maximum 10 entiers. Lorsqu'on rajoute un entier, il sera rajouté à la fin de la liste.

EXERCICE 12

Ecrire un programme qui gère une liste d'entiers grâce au menu suivant :
1. Ajouter un entier
2. Afficher la liste des entiers
3. Supprimer le premier entier ayant une valeur donnée.
4. Supprimer tous les entiers ayant une valeur donnée
5. Quitter

Il y aura au maximum 10 entiers. La liste devra être en permanence triée : lorqu'on rajoute un entier, il sera inséré au bon endroit dans la liste pour que celle-ci reste triée.

EXERCICE 13

Ecrire un programme qui demande à l'utilisateur de taper un entier N<=20 et qui affiche la N-ième ligne du triangle de pascal.
ligne 1 : 1 1
ligne 2 : 1 2 1
ligne 3 : 1 3 3 1
ligne 4 : 1 4 6 4 1
et ainsi de suite ...

EXERCICE 14

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers compris entre 0 et 20 qui seront stockés dans un tableau et qui affiche le nombre de fois qu'on a tapé un 0, le nombre de 1, le nombre de 2, ..., le nombre de 20.

EXERCICE 15

Ecrire un programme qui demande à l'utilisateur de taper le contenu d'un tableau de réels de 3 lignes et 3 colonnes et qui affiche ce tableau mais en affichant la moyenne des éléments de chaque ligne, de chaque colonne et la moyenne globale.