LANGAGE C++

LES TABLEAUX

Notion de types composés et tableaux

Un langage de programmation évolué doit fournir le moyen de **composer les types élémentaires** (**type de base**) pour construire des types plus complexes, **les types composés** (**type évolué**).

Exemples de données structurées

Tableau

Âge
20
35
26
38
22

Nom	Taille	Âge	Sexe
Dupond	1.75	41	M
Dupont	1.75	42	M
Durand	1.85	26	F
Dugenou	1.70	38	M
Pahut	1.63	22	F

Autre (plus tard)

Un tableau est une variable de type **composée** (**évoluée**).

Un tableau (variable indicée) est une collection de valeurs homogènes, constitué d'éléments qui sont tous du même type.
Ensemble de valeurs portant le même nom de variable et repérées par un nombre

Il existe deux sortes de tableaux

- Ceux dont la taille est connue à l'avance, les tableaux statiques
- Ceux dont la taille peut varier en permanence, les tableaux dynamiques.

Tableaux statiques

Tableaux de taille fixe

La taille est connue à l'avance, avant que le programme ne commence, et
- La taille ne va pas changer ; ne dépend pas du déroulement du programme.

Depuis la norme C++ 2011, il existe deux sortes de tableaux de taille fixe :
- les « anciens tableaux », « à la C »
- les array de la (nouvelle) bibliothèque standard, conçus pour être plus faciles d'utilisation

En **C++ 2011**, le type « tableau de taille fixe » est défini dans la bibliothèque **array**.

```
#include <array>
```

Les « anciens tableaux », « à la C »

```
int tableau[4];

tableau[0] = 10;
tableau[1] = 23;
tableau[2] = 505;
tableau[3] = 8;
```

Attention : un tableau commence à l'indice 0 ! Notre tableau de 4 int a donc les indices 0, 1, 2 et 3.
Il n'y a pas d'indice 4 dans un tableau de 4 cases !
C'est une source d'erreurs très courantes, souvenez-vous-en.

Tableaux statiques

Parcourir un « anciens tableaux », « à la C »

Supposons que je veuille maintenant afficher les valeurs de chaque case du tableau. Je pourrais faire autant de cout qu'il y a de cases. Mais bon, ce serait répétitif et lourd.
Le mieux est de se servir d'une boucle. La taille du tableau étant connue il est souhaitable d'utiliser une boucle for .

Les boucles for sont très pratiques pour parcourir un tableau :

```
int main()
{
        int tableau[4];
        int i = 0;
        tableau[0] = 10;
        tableau[1] = 23;
        tableau[2] = 505;
        tableau[3]  = 8;

        for(i = 0; i< 4; i++)
        {
            cout << tableau[i] << endl;
        }
        return 0;
}
```

```
Résultats : 10
            23
            505
            8
```

Attention à ne pas tenter d'afficher la valeur de tableau[4] !
Un tableau de 4 cases possède les indices 0, 1, 2 et 3, point barre. Si vous tentez d'afficher tableau[4], vous aurez soit n'importe quoi, soit une belle erreur, l'OS coupant votre programme car il aura tenté d'accéder à une adresse ne lui appartenant pas.

Tableaux statiques

Initialiser un « anciens tableaux », « à la C »

Maintenant que l'on sait parcourir un tableau, nous sommes capables d'initialiser toutes ses valeurs à 0 en faisant une boucle !

```
int main()
{
        int tableau[4], i = 0;
        // Initialisation du tableau
        for(i = 0; i< 4; i++)
        {
             tableau[i] = 0;
        }
        // Affichage de ses valeurs pour vérifier;
        for(i = 0; i< 4; i++)
        {
            cout << tableau[i] << endl;
        }
        return 0;
}
```

```
Résultats : 0
            0
            0
            0
```

```
int main()
{
        int tableau[4]= {0,0,0,0}, i = 0;

        // Affichage de ses valeurs pour vérifier;
        for(i = 0; i< 4; i++)
        {
            cout << tableau[i] << endl;
        }
        return 0;
}
```

Il faut savoir qu'il existe une autre façon d'initialiser un tableau un peu plus automatisée en C.

Elle consiste à écrire tableau[4] = {valeur1, valeur2, valeur3, valeur4}.

En clair, vous placez les valeurs une à une entre accolades, séparées par des virgules :

Tableaux statiques

Les tableaux hérités du langage C et les fonctions

Tableau en paramètre

- Un tableau statique « c » est toujours passé par **référence**. Ne pas utiliser l'esperluette (&).
  => lorsqu'on passe un tableau à une fonction, cette dernière peut le modifier.
- Nous devons passer, en plus de la référence du tableau, **sa taille**.

```
#include <iostream>
using namespace std;
void afficher(int t[],int taille);
void ecrire(int t[],int taille);
int main()
{
    int tab[] = {1,2,3};
    afficher(tab,3);
    ecrire(tab,3);
    afficher(tab,3);
    return 0;
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=3
indice=1 valeur=2
indice=2 valeur=1

Process returned 0 (0x0)   execution ti
Press any key to continue.
```

```
void afficher(int t[],int taille){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<taille;i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(int t[],int taille){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<taille;i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

Tableaux statiques

Les array (C++2011)

Syntaxe

```
array<type, taille> identificateur;
```

- **identificateur** est le nom du tableau,
- **type** correspond au type des éléments du tableau
- **taille** est le nombre d'éléments que contient le tableau.

La taille du tableau doit être **connue** au moment où on écrit le programme

```
array<double, 3> tab;
```

déclaration d'un tableau de 3 double.

Tableaux statiques

Deux alternatives pour initialiser un array

Syntaxe

```
array<int, 5> ages ( { 20, 35, 26, 38, 22 } );
        // ou :
array<int, 5> ages = { 20, 35, 26, 38, 22 } ;
```

Un array non initialisé contient « n'importe quoi ».

L'accès aux éléments d'un tableau de taille fixe se fait de la même façon que pour un tableau dynamique :

- directement : **tab[i]**
- par itération : **for(auto element : tableau)**
- itération **for « classique »**

Les tableaux de taille fixe array ont aussi une « fonction spécifique » size() qui renvoie leur taille.

```
array<int, 3> tab1 = { 1, 2, 3 } ;
array<int, 3> tab2 ;
tab2 = tab1 ; // copie de tab1 dans tab2
for(auto e : tab2){
    cout << e << " ";
}
```

```
array<int, 3> tab1 = { 1, 2, 3 } ;
array<int, 3> tab2 ;
tab2 = tab1 ; // copie de tab1 dans tab2
for(size_t i(0); i < tab2.size(); ++i) {
    cout << tab2[i] << " ";
}
```

```
1 2 3
Process returned 0 (0x0)
Press any key to continue.
```

Tableaux statiques

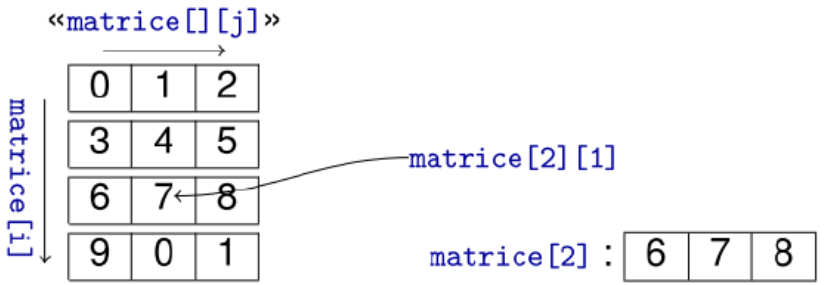
Tableau array multidimensionnel de taille fixe

On peut déclarer des tableaux de taille fixe multidimensionnels.

- On peut même faire des tableaux dynamiques de tableaux de taille fixe, des tableaux de taille fixe de tableaux dynamiques, etc.)

```
array<array<double, 2>, 2> rotation;
array<array<int, 10>, 20> statistiques;
array<array<array<double, 4>, 2>, 3> tenseur;
```

```
#include <iostream>
#include <array>
using namespace std;
int main()
{
    array<array<int, 3>, 4> matrice = {
        0, 1, 2 ,
        3, 4, 5 ,
        6, 7, 8 ,
        9, 0, 1
    };
    for(auto ligne : matrice){
        for(auto e : ligne){
            cout << e << " ";
        }
        cout << endl;
    }
    return 0;
}
```




```
0 1 2
3 4 5
6 7 8
9 0 1

Process returned 0 (0x0)
Press any key to continue.
```

Tableaux statiques

Tableau statique array et fonctions (1)

```
#include <iostream>
#include <array>
using namespace std;
void afficher(array<int,3> t);
void ecrire(array<int,3> t);
int main()
{
    array<int,3> tab = {1,2,3};
    afficher(tab);
    ecrire(tab);
    afficher(tab);
    return 0;
}
```

```
void afficher(array<int,3> t){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(array<int,3> t){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3

Process returned 0 (0x0)   execution ti
Press any key to continue.
```

Un tabeau statique array comme paramétre d'une fonctions a le même comportement qu'une variable de base (int, double, float, …)

Tableaux statiques

Tableau statique array et fonctions (2)

```
#include <iostream>
#include <array>
using namespace std;
void afficher(array<int,3> t);
void ecrire(array<int,3>& t);
int main()
{
    array<int,3> tab = {1,2,3};
    afficher(tab);
    ecrire(tab);
    afficher(tab);
    return 0;
}
```

```
void afficher(array<int,3> t){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(array<int,3>& t){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=3
indice=1 valeur=2
indice=2 valeur=1

Process returned 0 (0x0)   execution tim
Press any key to continue.
```

Tableaux dynamiques

Les vector (C++11)

Un tableau dynamique, est une collection de données homogènes, dont le nombre peut changer au cours de l'exécution du programme.

Pour les utiliser, il faut tout d'abord importer les définitions associées :

```
#include <vector>
```

Syntaxe

```
vector<type> identificateur;
```

- **identificateur** est le nom du tableau,
- **type** correspond au type des éléments du tableau.

Remarque : Le type des éléments peut être n'importe quel type C++

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> v;
    return 0;
}
```

Tableaux dynamiques

Initialisation d'un tableau dynamique

il y a cinq façons d'initialiser un tableau dynamique :

1. **vide**

```
vector<int> tableau;
```

2. **avec un ensemble de valeurs initiales**

```
vector<int> ages = { 20, 35, 26, 38, 22 };
```

3. **avec une taille initiale donnée et tous les éléments « nuls »**

```
vector<int> tab(5);
```

4. **avec une taille initiale donnée et tous les éléments à une même valeur donnée**

```
vector<int> tab1(5, 1);
```

Tableau d'entiers dont les 5 éléments de départ sont initialisés à la valeur 1

5. **avec une copie d'un autre tableau**

```
vector<int> tab2(tab1);
```

Copie de tab1 dans tab2.
Deux tableaux distinctes

Tableaux dynamiques

Accès direct aux éléments d'un tableau dynamique

Exemple: Tableau scores contenant 4 entiers

```
vector<int> scores = { 210, 126, 38, 422 };
```

Scores[0]	Scores[1]	Scores[2]	Scores[3]
210	126	38	422

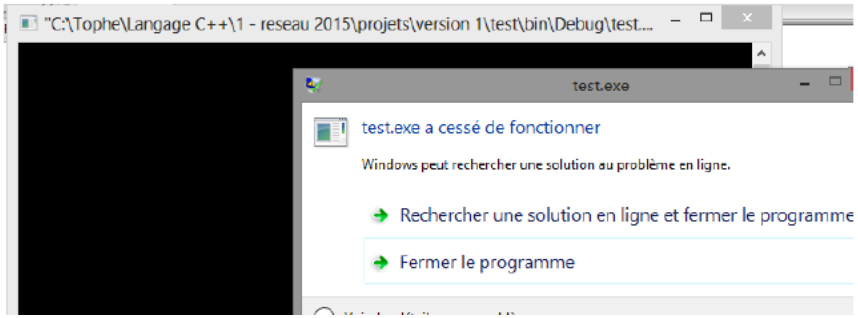
Les indices des tableaux commencent 0 et non à 1. Donc attention, tab [2] est le troisième élément du tableau tab!

Attention !

- Les indices correspondant aux éléments d'un tableau de taille T varient entre 0 et T-1
- Il n'y a pas de contrôle de débordement ! ! Il est impératif que l'élément que vous référencez existe effectivement.

Démo : Erreur accés mémoire « Segmentation Fault »

```
vector<double> v;
v[0] = 5.4;
```



Tableaux dynamiques

Accès à tous les éléments d'un tableau dynamique

Très souvent, on voudra accéder à chacun des éléments d'un tableau en effectuant une itération sur ce tableau.
Il existe en fait au moins trois façons d'itérer sur un tableau :

1. avec les itérations sur ensemble de valeurs

```
for(auto element : tableau)
```

2. avec une itération for « classique » :

```
for(size_t i(0); i < TAILLE; ++i)
```

3. [avancé] avec des itérateurs (non présenté dans ce cours)

Tableaux dynamiques

Itérations sur ensemble de valeurs

Si l'on **ne veut pas modifier** les éléments du tableau : `for(auto nom_de_variable : tableau)`

Si l'on **veut modifier** les éléments du tableau `for(auto& nom_de_variable : tableau)`

```
vector<int> tableau = { 1, 2, 3, 4 };
for(auto element : tableau){
    cout << element << " " ;
}
```

```
1 2 3 4
Process returned 0 (0x0)
Press any key to continue.
```

```
vector<int> ages(5);
for(auto& age : ages) {
    cout << "Age de l'etudiant suivant ? ";
    cin >> age;
}
cout << "Age des etudiants : " << endl;
for(auto age : ages) {
    cout << " " << age << endl;
}
```

```
Age de l'etudiant suivant ? 22
Age de l'etudiant suivant ? 28
Age de l'etudiant suivant ? 24
Age de l'etudiant suivant ? 26
Age de l'etudiant suivant ? 32
Age des etudiants :
 22
 28
 24
 26
 32

Process returned 0 (0x0)   exec
Press any key to continue.
```

Note : les itérations ne permettent pas :

- d'itérer sur plusieurs tableaux à la fois (les comparer, les mélanger, ...)
- d'accéder à plusieurs éléments
- de sauter des éléments, d'avancer de deux en deux

Tableaux dynamiques

Itération for « classique »

la fonction size() : `tab.size();`

=> retourne la taille du tableau tab.

Cette taille est de type size_t, un « int » particulier, toujours positif.

```
vector<int> ages(5);
for(size_t i(0); i < ages.size(); ++i) {
    cout << "Age de l'etudiant suivant ? ";
    cin >> ages[i];
}
cout << "Age des etudiants : " << endl;
for(size_t i(0); i < ages.size(); ++i) {
    cout << " " << ages[i] << endl;
}
```

```
Age de l'etudiant suivant ? 30
Age de l'etudiant suivant ? 22
Age de l'etudiant suivant ? 24
Age de l'etudiant suivant ? 28
Age de l'etudiant suivant ? 34
Age des etudiants :
 30
 22
 24
 28
 34
```

Si l'on veut expliciter les indices :

```
vector<int> tab = { 20, 35, 26, 38, 22 };
for(size_t i(0); i < tab.size(); ++i) {
    cout << "L'element " << i << " vaut " << tab[i] << endl;
}
```

```
L'element 0 vaut 20
L'element 1 vaut 35
L'element 2 vaut 26
L'element 3 vaut 38
L'element 4 vaut 22

Process returned 0 (0x0)
Press any key to continue.
```

Tableaux dynamiques

Exemples

Saisie au clavier des éléments du tableau

- si l'on n'a pas besoin d'expliciter les indices

```
vector<int> tab (4);
for(auto& element : tab) {
    cout << "Entrez l'element suivant :" << endl;
    cin >> element;
}
```

```
Entrez l'element suivant :
1
Entrez l'element suivant :
2
Entrez l'element suivant :
3
Entrez l'element suivant :
4

Process returned 0 (0x0)
Press any key to continue.
```

- si l'on veut expliciter les indices

```
vector<int> tab (4);
for(size_t i(0); i < tab.size(); ++i) {
    cout << "Entrez l'element " << i << ":" << endl;
    cin >> tab[i];
}
```

```
Entrez l'element 0:
1
Entrez l'element 1:
2
Entrez l'element 2:
3
Entrez l'element 3:
4

Process returned 0 (0x0)
Press any key to continue.
```

- Affectation de tous les éléments d'un tableau à la valeur 1.2

```
vector<int> tab({ 1, 2, 3 });
for(auto& e : tab) {
    e = 1.2;
}
```

```
vector<int> tab({ 1, 2, 3 });
for(size_t i(0); i < tab.size(); ++i) {
    tab[i] = 1.2;
}
```

Tableaux dynamiques

Affectation globale d'un tableau

Tout tableau (qui n'a pas été déclaré comme constant) peut être modifié par une affectation globale du tableau en tant que tel.

```
vector<int> tab1({ 1, 2, 3 });
vector<int> tab2;
tab2 = tab1 ; // copie de tout tab1 dans tab2
for(auto e : tab2) {
    cout << e << " ";
}
```

```
1 2 3
Process returned 0 (0x0)
Press any key to continue.
```

```
vector<double> tab({ 1.4, 2.0, 3.6 });
tab = vector<double>(tab.size(), 1.2);
for(auto e : tab)  {
    cout << e << " ";
}
```

Tableau anonyme

Tableaux dynamiques

Fonctions spécifiques

Un certain nombre d'opérations sont **directement attachées** au type vector.
L'utilisation de ces opérations spécifiques se fait avec la syntaxe suivante :

```
nom_de_tableau.nom_de_fonction(arg1, arg2, ...);
```

Quelques fonctions

- **tableau.size()** : renvoie la taille de tableau (type de retour : size_t)
- **tableau.front()** : renvoie une référence au 1 er élément
- **tableau.front()** est donc équivalent à tableau[0]
- **tableau.back()** : renvoie une référence au dernier élément (équivalent à tableau[tableau.size()-1])
- **tableau.empty()** : détermine si tableau est vide ou non (bool).
- **tableau.clear()** : supprime tous les éléments de tableau (tableau vide). Pas de (type de) retour.
- **tableau.pop_back()** : supprime le dernier élément de tableau. Pas de retour.
- **tableau.push_back(valeur)** : ajoute un nouvel élément de valeur valeur à la fin de tableau. Pas de retour.

```
vector<int> tab = { 20, 35, 26, 38, 22 };
size_t taille(0);
taille = tab.size();
cout << "taille = " << taille ;
```

```
taille = 5
Process returned 0 (0x0)
Press any key to continue.
```

```
vector<double> v(3, 4.5);
v.pop_back();
v.push_back(5.6);
v.push_back(6.7);
v.pop_back();
```

Tableaux dynamiques

Exercice

Ecrire une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant 0
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```

```
vector<int> tab;
saisie(tab, 5); // saisie de 5 éléments
saisie(tab); // saisie de 4 éléments
vector<int> tab2(12);
saisie(tab2, tab2.size());
```

```
void saisie(vector<int>& vect, size_t taille)
```

Tableaux dynamiques

Tableaux multidimensionnels

C'est en fait un **tableau de tableaux**...

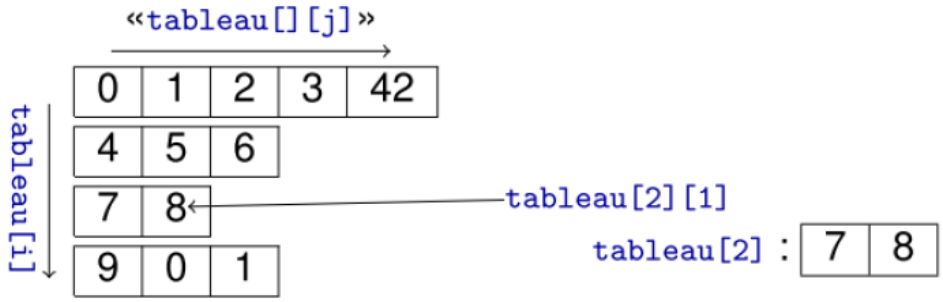
```
vector<vector<int>> tab(5, vector<int>(6));
```

correspond à la déclaration d'un tableau de 5 tableaux de 6 entiers

- **tab[i]** est donc un « **vector<int>** », c'est-à-dire un tableau dynamique d'entiers
- **tab[i][j]** sera alors le **(j +1)-ième** élément de ce tableau
- On a l'habitude de se représenter **tab[i][j]** comme l'élément de la **(i+1)ème** ligne et de la **(j+1)ème** colonne.

Mais attention !

Un vector<vector<int>> n'est pas une matrice, mais un tableau dynamique de tableaux dynamiques d'entiers (pas nécessairement tous de la même taille !).



Tableaux dynamiques

Exemple

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<vector<int>> tableau(
        { { 0, 1, 2, 3, 42 },
          { 4, 5, 6 },
          { 7, 8 },
          { 9, 0, 1 } }
        );
    for(auto ligne : tableau) {
        for(auto element : ligne) {
            cout << element << " ";
        }
        cout << endl;
    }
    for(size_t i(0); i < tableau.size(); ++i) {
        cout << "tableau[" << i <<"].size()="
        << tableau[i].size() << endl;
    }
    return 0;
}
```

```
0 1 2 3 42
4 5 6
7 8
9 0 1
tableau[0].size()=5
tableau[1].size()=3
tableau[2].size()=2
tableau[3].size()=3

Process returned 0 (0x0)
Press any key to continue.
```

Tableaux dynamiques

Les tableaux dynamiques vector et les fonctions (1)

```
#include <iostream>
#include <vector>
using namespace std;
void afficher(vector<int> t);
void ecrire(vector<int> t);
int main()
{
    vector<int> tab;
    tab.push_back(1);
    tab.push_back(2);
    tab.push_back(3);
    afficher(tab);
    ecrire(tab);
    afficher(tab);
    return 0;
}
```

```
void afficher(vector<int> t){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(vector<int> t){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3

Process returned 0 (0x0)   execution ti
Press any key to continue.
```

Tableaux dynamiques

Les tableaux dynamiques vector et les fonctions (2)

```
#include <iostream>
#include <vector>
using namespace std;
void afficher(vector<int> t);
void ecrire(vector<int>& t);
int main()
{
    vector<int> tab;
    tab.push_back(1);
    tab.push_back(2);
    tab.push_back(3);
    afficher(tab);
    ecrire(tab);
    afficher(tab);
    return 0;
}
```

```
void afficher(vector<int> t){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(vector<int>& t){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=3
indice=1 valeur=2
indice=2 valeur=1

Process returned 0 (0x0)   execution ti
Press any key to continue.
```

Tableaux dynamiques

Les tableaux dynamiques vector et les fonctions (2)

```
#include <iostream>
#include <vector>
using namespace std;
void afficher(vector<int> t);
void ecrire(vector<int>& t);
int main()
{
    vector<int> tab;
    tab.push_back(1);
    tab.push_back(2);
    tab.push_back(3);
    afficher(tab);
    ecrire(tab);
    afficher(tab);
    return 0;
}
```

```
void afficher(vector<int> t){
    cout << "--- Valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "indice=" << i << " valeur=" << t[i] << endl;
    }
}
void ecrire(vector<int>& t){
    cout << "--- Saisir les valeurs du tableaux ---" << endl;
    for(int i=0;i<t.size();i++){
        cout << "Composante d'indice=" << i << " : ";
        cin >> t[i];
    }
}
```

```
--- Valeurs du tableaux ---
indice=0 valeur=1
indice=1 valeur=2
indice=2 valeur=3
--- Saisir les valeurs du tableaux ---
Composante d'indice=0 : 3
Composante d'indice=1 : 2
Composante d'indice=2 : 1
--- Valeurs du tableaux ---
indice=0 valeur=3
indice=1 valeur=2
indice=2 valeur=1

Process returned 0 (0x0)   execution ti
Press any key to continue.
```

Les tableaux C++ 2011

Tableaux dynamiques	Tableaux statiques
`#include <vector>`	`#include <array>`
`vector<double> tab;`	`array<double, 5> tab;`
`vector<double> tab2(5);`	

```
tab[i][j]

tab.size()

for(auto element : tab)

for(auto& element : tab)
```

`tab.push_back(x);`	—
`tab.pop_back();`	—

```
vector<vector<int>> tableau(
   { { 0, 1, 2, 3, 42 },
     { 4, 5, 6        },
     { 7, 8           },
     { 9, 0, 1        } }
);
```

```
array<array<int, 3>, 4> matrice =
{
        0, 1, 2 ,
        3, 4, 5 ,
        6, 7, 8 ,
        9, 0, 1
};
```