DAUPHINE UNIVERSITÉ PARIS | TUNIS

Le langage C++

Master Actuariat

Fatma CHAKER KHARRAT
chakerfatma@yahoo.fr

Année universitaire 2014/2015

DAUPHINE UNIVERSITÉ PARIS | TUNIS

Le langage C++
Master Actuariat

Séance 2 :
Les tableaux

Les Tableaux à la « C »: Déclaration

Type nom[taille];

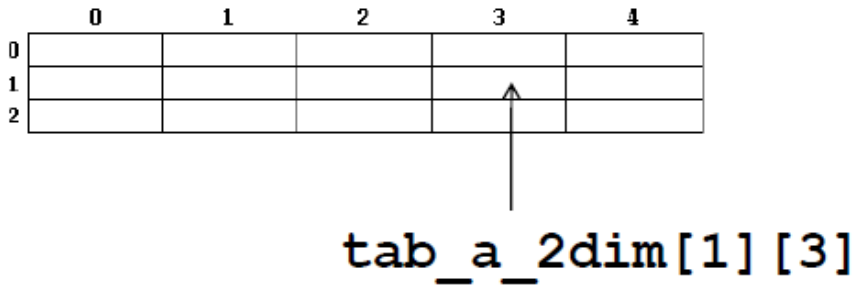
Exemple :

```
int Tab[32];
float salaire[32];
```

- Le type est le même pour chaque élément du tableau
- La taille du tableau doit être exprimé avec un chiffre pour pouvoir compiler

```
#define MAX_ELEM 32     // ou encore const int MAX_ELEM=32;
float salaire[MAX_ELEM];
```

int tab_a_2dim[3][5];



Les Tableaux à la « C »

- Le compilateur va réserver ***taille de tableau * taille du type*** espaces contiguës en mémoire
- Le **nom du tableau** est une **constante** qui vaut **l'adresse du premier élément**
- Les **indices d'un tableau** vont de **0 à MAX_ELEM – 1**
- Pour accéder à un élément en particulier, suffit de spécifier son indice.

  Exemple : **Tab[3]** correspond au **4ième élément** du tableau Tab

Les Tableaux : Initialisation

- Un tableau peut être initialisé **à la déclaration**

```
int tab2 [] = {1,2,3} ; // Déclaration et initialisation d'un tableau
                        //de 3 entier. Les indices commencent à zéro
```

- L'initialisation peut aussi avoir lieu **durant l'exécution du programme**

```
for(i = 0; i < 8; i++)
  Tab[i] = i*2;
```

- autre exemple

```
for(i = 0; i < 8; i++)
   cin>>Tab[i];
```

- Les deux déclarations suivantes sont équivalentes

```
int tab[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int tab[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
```

La classe vector du C++

Pour utiliser la classe `vector`, il faut placer en tête du fichier :

```
# include <vector>
```

Un tableau est typé:

```
vector<int> Tab(100,5) ;
vector<int> Tab(50) ;
vector<double> T ;
```

Structure générale: `vector<` *type* `>` *Nom*`(`*n*`,`*v*`) ;`

- `vector<` *type* `>` *Nom1* `=` *Nom2* `;`
  → les valeurs de *Nom2* sont alors recopiées dans *Nom1*.
- `T.size()` correspond à la taille de `T`.
  NB: `size()` renvoie en fait un entier non signé, son type exact est `vector<type>::size_type`

La classe vector du C++

- `T[i]` désigne le i-ème élément avec i = 0, ...`T.size()-1`.
- `vector<vector<int> > T` définit un tableau à deux dimensions.

  Pour l'initialiser, on peut utiliser l'instruction suivante :

```
vector<vector<int> >
T2(100,vector<int>(50,1)) ;
```

  ...on initialise chacune des 100 cases de T1 avec un tableau de taille 50 rempli de 1.

Exemple

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
// initialier une matrice Mat(3,4) avec la valeur 5
vector<vector<int> >Mat(3,vector<int>(4,5));
for(int i=0;i<3;i++)
            for(int j=0;j<4;j++)
            {cout<<" "<<Mat[i][j];
            if (j==3)
            cout<<endl;
            }
}
```

Exécution :

5 5 5 5
5 5 5 5
5 5 5 5

Tableaux comme paramètre de fonction

- Passer un tableau “C” en paramètre d’une fonction revient à passer **l’adresse de la première case** ➔ **passage par référence.**

- On peut utiliser indifféremment le formalisme « **tableau** » (avec ou sans la dimension effective du tableau) ou le formalisme **pointeur**.

Tableaux comme paramètre de fonction

Formalisme « tableau »

Les déclarations suivantes sont équivalentes :

- **Void fct(int T[])**
- **void fct(int T[5])**

A l'appel : fct(T);

Le prototype : void fct(int []);

Formalisme « pointeur »

La déclaration d'un tableau en utilisant le formalisme pointeur :

void fct(int *T)

Le prototype : void fct(int *);

A l'appel : fct(T);

Exemple :

Exemple 1 : Formalisme Tableau

```
//proptotype
void saisir(int [ ]);
int main()
{
int Tab[30];
int N;
//appel
saisir(Tab);
…..

}
//définition de la fonction
void saisir(int T[ ]) // ou T[30]
{
….
}
```

Exemple 2 : Formalisme Pointeur

```
//proptotype
void saisir(int *);
int main()
{
int Tab[30];
int N;
//appel
saisir(Tab);
…..

}
//définition de la fonction
void saisir(int *T)
{
….
}
```

Exercice : Tableaux

Ecrire un programme qui fait appel à 2 fonctions :

- Une fonction **remplir** qui ne renvoie **aucune valeur** et qui saisi les éléments du tableau. On prévoira 2 arguments : **le tableau et sa dimension.**

- Une fonction **afficher** qui ne renvoie **aucune valeur** et qui affiche les éléments du tableau. On prévoira 2 arguments : **le tableau et sa dimension.**

- Une fonction **chercher** qui renvoie un **booléen** qui teste la présence d'une valeur dans un tableau et d'afficher le nombre d'occurrence de cette valeur. On prévoira 4 arguments : **le tableau, sa dimension, la valeur et le nombre d'occurrence.**

Ecrire un programme d'essai.