TABLEAUX

TABLEAU

Un tableau est une liste d'éléments ayant le même type et désignés sous le même nom et accessibles par indices (commençant par 1) .

- TABLEAU UNIDIMENSIONNEL
  - Il est déclarer comme suit:

    nomTableau: tableau[taille] de type
  - **Exemple**: Notes : tableau[10] de réels
- TABLEAU BIDIMENSIONNEL (on peut avoir un tableau multidimensionnel)
  - Il est déclarer comme suit:

    nomTableau: tableau[ligne][colonne] de type
  - **Exemple**: matrice: tableau[2][3] de réels
- **Lecture**: écrire(nomTableau[i][j][k]…[n])
- **Ecriture**: lire(nomTableau[i][j][k]…[n])
- **Affectation**: nomTableau[i][j][k]…[n] ← Valeur

TABLEAU EN C++

- TABLEAU UNIDIMENSIONNEL
  - Il est déclaré comme suit:

    Type nom [taille] ;
  - **Exemple**: double notes[10] ;
- TABLEAU BIDIMENSIONNEL (on peut avoir un tableau multidimensionnel)
  - Il est déclaré comme suit:

    type nom [ligne][colonne] ;
  - **Exemple**: double matrice[2][3] ;
- **Lecture**: cout << nomTableau[i][j][k]…[n] << endl;
- **Ecriture**: cin >> nomTableau[i][j][k]…[n];
- **Affectation**: nomTableau[i][j][k]…[n] = Valeur;

DÉCLARATION TYPEDEF

- Elle permet d'attribuer un nom à un type. Sa forme générale est: **typedef** <*déclaration*>
- C'est très pratique pour nommer certains types de tableaux :

```
typedef int Matrice[2][3];   // définit un type Matrice
```

- Et de l'utiliser pour déclarer ensuite, son équivalent :

```
Matrice M;
```

TABLEAUX DYNAMIQUES - VECTOR

Un tableau dynamique est un tableau dont la taille peut varier.

- Syntaxe: **vector**<TYPE> nom(TAILLE); // Il va falloir inclure la bibliothèque <vector>
- Quelques fonctions utiles:
  - push_back (): Ajout à la fin du vecteur
  - pop_back (): retire de la fin du vecteur
  - size() : retourne le nombre d'éléments du vecteur
  - erase() : supprime un éléments ou un intervalle d'un vecteur et déplace les éléments suivants.
- **Exemples**:
  - Créer un tableau vide: vector <double> tab;
  - Créer un tableau de 10 éléments: vector <int> tab(10);
  - Créer un tableau de 10 éléments initialisés à 0:
    - vector <int> tab(10, 0);
    - tab.push_back(3); // ajout du 11ème case au tableau de valeur 3;
    - tab.pop_back(); // suppression de la dernière case du tableau;
    - int const taille(tab.size()); // variable contenant la taille du tableau;
    - tab.erase(tab.begin()+5); // suppression du 6iéme élément;
    - tab.erase(tab.begin(), tab.begin()+5); // suppression des 5 premiers éléments;

TABLEAUX DYNAMIQUES - VECTOR

Il est également possible de créer des tableaux multidimensionnels de taille variable en utilisant les vector.

- Syntaxe pour 2D: **vector**<**vector**<TYPE>> nom; // nous avons plus tôt un tableau de ligne.

  Exemples:

  - vector<vector<int>> mat;
  - mat.push_back(vector(3)); //ajout d'une ligne de 3 cases;
  - mat[0].push_back(4); //ajout d'une case contenant 4 à la 1ière ligne du tableau;
  - mat[0][2] = 6; // change la valeur de la cellule (0, 2) du tableau;

STRING ET TABLEAUX

Une chaine de caractère est en réalité un tableau de caractères. Ce qui veut dire qu'il a beaucoup de point communs avec les vector.

- Pour pouvoir utiliser la classe standard, il faut rajouter la bibliothèque <string>;
  - Elle embarque toutes les opérations de base sur les chaînes:
    - Déclaration: string s1; string s2="Hello";
    - Saisie et Affichage: cin>>s1; cout<<s2;
    - Concaténation: string s3=s1+s2;
      - s3.size(); // pour connaitre le nombre de lettres;
      - s3.push_back("!"); // pour ajouter des lettres à la fin;
      - s3.at(i); // pour récupérer le i-ème caractère;
      - getline(cin, s4); // pour saisir une chaîne de caractères en utilisant le passage à la ligne comme séparateur (notre chaîne de caractères peut alors comporter des espaces);

STRING ET TABLEAUX

Exemple:

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

int main()
{
	string prenom("Masamba");
	cout << "Je suis" << prenom << "et toi ?" <<endl;
	prenom[2] = ‘d';
	prenom[3] = ‘e';
	cout << "moi c’est" << prenom << "!" << endl;

	return 0;
}
```

EXERCICES D'APPLICATIONS

Application 10:

Ecrivez un programme qui lit au clavier une suite de nombres entiers positifs ou nuls et qui les affiche dans l'ordre inverse de leur lecture. La frappe d'un nombre négatif indique la fin de la série. Nous avons des raisons de penser qu'il n'y aura pas plus de 100 nombres.

Application 11 :

On considère un tableau tab de N entiers. Ecrire un programme permettant:

a) de compter le nombre d'éléments nuls de tab
b) de chercher la position et la valeur du premier élément non nul de tab
c) de remplacer les éléments positifs par leur carré

Application 12 :

Ecrire un programme qui permet de saisir des nombres entiers dans un tableau à deux dimensions TAB [10][20] et de calculer les totaux par ligne et par colonne dans des tableaux TOTLIG[10] et TOTCOL[20].

Application 13 :

Ecrire un programme qui permet de chercher une valeur x dans un tableau à deux dimensions t[m][n]. Le programme doit aussi afficher les indices ligne et colonne si x a été trouvé.