

Programmation et Structure de Données

Structure de données de Base
Les Pointeurs

Notion de pointeurs

1. Définition : Un Pointeur permet de manipuler les adresses

Déclaration de pointeurs :

n : entier; Variable simple

*ad : entier; **Variable pointeur**

(en C++ : int *ad;)

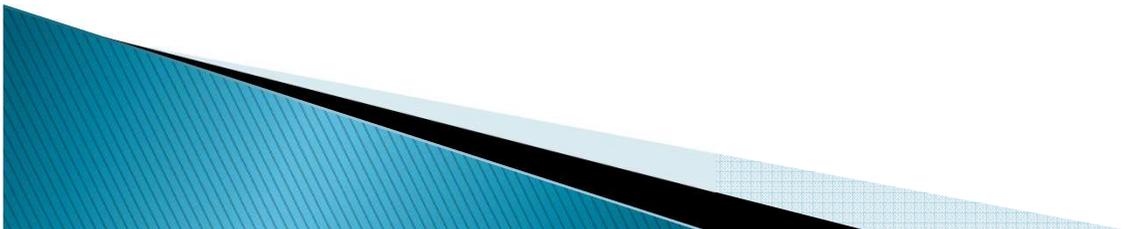
ad ← &n; (en C++ : ad = &n;)

Cette écriture permet d'affecter à ad la valeur de référence de la variable n en mémoire (lier la variable n avec l'adresse ad)

Si on fait n ← 20; alors l'instruction : Écrire (*ad) va afficher 20;

Et si on fait *ad ← 60; alors l'instruction : Écrire (n) va afficher 60;

C'est pour ça qu'il y a une différence entre le passage par valeur et par adresse dans les sous programmes



Pointeurs et Tableaux

2. Tableaux et pointeur en C++ : un tableau est un pointeur constant

Les deux déclarations sont similaires : `int T[10];` et `int *T;`
Ainsi les deux écritures sont identiques : `T[i] = 12;` `*(T+i) = 12;`

Ex : initialiser à 0 les éléments d'un tableau T d'entier :

```
const int dim = 10;  
int i, T[dim];  
for (i=0 ; i<dim ; i++)  
    *(T+i) = 0 ; // équivalent à T[i] = 0;
```

Tableaux à deux dimensions :

La déclaration `int M[3] [4];` veut dire que M désigne un tableau de 3 éléments, chacun de ces éléments étant lui-même un tableau de 4 entiers

Exemple : afficher les éléments d'une matrice Mat de taille NxM:

```
for(i=0;i<N;i++)  
{  
    for(j=0; j<M; j++) cout << *(Mat[i]+j) << "  ";  
    cout << "\n";  
}
```

Gestion Dynamique

3. La gestion dynamique : les opérateurs *new* et *delete*:

new permet l'allocation dynamique d'un emplacement en mémoire

delete permet de libérer l'emplacement qui a été réservé par *new* précédemment.

Tableau à une dimension : ex : un prg qui remplit un tableau de N entiers puis affiche ses éléments :

```
int N,i;
cout << "Donner un nombre positif non nul :";
cin >> N;
int *Tab ;
Tab = new int[N];
for(i=0; i<N; i++)
    { cout << "Donner l'element "<<i+1<<" : ";
      cin >> Tab[i]; //ou *(Tab+i);
    }
for(i=0; i<N; i++) cout << Tab[i]<<" "; //ou *(Tab+i);
delete Tab;
```

On peut réutiliser la variable Tab :

```
cin >> N;
Tab = new int[N];
```

Gestion Dynamique

Tableau à deux dimensions: ex : un prg qui qui remplit une Matrice NxM d'entiers puis affiche ses éléments sous forme de tableau à 2 dimensions

```
int N,M,i,j;
cout << "Donner un nombre de lignes (positif non nul) :"; cin >> N;
cout << "Donner un nombre colonnes (positif non nul) :"; cin >> M;
int **Mat = new int*[N];
    for(i=0;i<N;i++)
        Mat[i] = new int[M];
for(i=0;i<N;i++)
    for(j=0;j<M;j++)
        { cout << "\nDonner l'element "<<i<<" - "<<j<<" : ";
          cin >> Mat[i][j]; //ou *(Mat[i]+j);
        }
for(i=0;i<N;i++)
    { for(j=0;j<M;j++) cout << Mat[i][j]<<setw(5);
      cout << "\n";
    }
delete Mat;
```

Les tableaux comme arguments d'une fonction

4. Les tableaux comme arguments d'une fonction :

Tab à une dimension : La déclaration T[] est remplacée par *T

Ex : fonction qui retourne la somme des éléments d'un tableau passé en argument : int somme (int *T, int N)

```
{
    int i, S=0;
    for(i=0; i<N; i++) S += T[i];
    return S;
}
```

Tab à deux dimensions : La déclaration Mat[][col] est remplacée par **Mat

Ex : fonction qui retourne la somme des éléments du diagonal d'une matrice carrée passée en argument :

```
int som_diag (int **Mat, int N)
{
    int i, S=0;
    for(i=0; i<N; i++) S += Mat[i][i];
    return S;
}
```