Rappel : Les tableaux en C

Un tableau est un ensemble fini d'éléments de même type, stockés en mémoire à des adresses contiguës

déclaration d'un tableau à une dimension

- *type nom-du-tableau*[*nombre-éléments*];
- Exemple

```
{
  int tab[10];
  int i;
 ...
   for (i = 0; i < 10; i++)
   {Tab[i]= i*i ;
   }
   for (i = 0; i < 10; i++)
   printf("tab[%d] = %d\n",i,tab[i]);
}
```

initialisations des tableaux

- Dans ce cas la dimension n'est pas nécessaire.

```
vecteur vect0={0,0,0};
int chiffres[]={0,1,2,3,4,5,6,7,8,9};
```

Si elle est donnée les suivantes seront mises à 0 :

```
int tableau[20]={1,2,3}; /* les 17 autres à 0 */
```

On peut également déclarer un tableau sans en donner la dimension.

Dans ce cas là le compilateur ne lui réserve pas de place, (elle aura du être réservée autre part (par exemple tableau externe ou argument formel d'une fonction).

Tableau et pointeur

- Un tableau tab[] s'identifie à un pointeur vers le premier élément du tableau tab

Sont équivalentes les deux écritures :

Tab et &Tab[0]

Tab[0] et *Tab

Tab+1= adresse de l'élément suivant du tableau (le deuxième)

Structures et unions

- Exemple de représentation de l'élément:
  Personne (nom, prenom et age)

- Contrairement aux tableaux, où tous les constituants doivent être **obligatoirement du même type**,
  Les structures sont des variables composées de plusieurs variables (ou **CHAMPS**) de types différents.

  Chaque champs n'est plus désigné par un numéro comme dans un tableau, mais par un **identificateur**.

Déclaration des structures

- **déclaration** : struct nom_type {déclaration des champs} liste_variables ;

- Pour l'exemple:

```
struct Personne {        char nom[30];
                          char prenom[20];
                          short int age;
                       } Etudiant,Professeur,Classe[50];
```

Déclaration des structures

struct nom_type {déclaration des champs} liste_variables ;

- Les champs peuvent être (scalaires, tableaux, pointeurs...), y compris une structure

- nom_type et liste_variables sont optionnels mais au moins l'un des deux doit être présent.
- Nom_type (ici Personne) est le nom d'un nouveau type, il peut être utilisé plus loin pour déclarer d'autres variables, voire d'autres types:

- struct Personne Directeur;
  struct Employé { struct Personne id; char[30] Fonction;} moi;

Les unions

- Les unions permettent de stocker un choix de plusieurs choses en une même zone mémoire.
- La définition d'une union est semblable à celle d'une structure, de même que la syntaxe d'accès.
- La différence est qu'on ne peut stocker qu'un seul élément à la fois.