



Chapitre 8:

Les Enregistrements

(Les structures)

Les Structures

1. Introduction

- ❑ Un tableau permet de désigner sous un seul nom un ensemble de valeurs de même type, chacune d'entre elles étant repérée par un indice.
- ❑ Une structure, permet de désigner sous un seul nom un ensemble de valeurs pouvant être de types différents.
- ❑ L'accès à chaque élément de la structure (nommé champ) se fera, cette fois, non plus par une indication de position, mais par son nom au sein de la structure.

2. Définition et déclaration d'une structure

Syntaxe :

```
struct <nom>
{
    <type_champ1><nom_champ1>;
    <type_champ2><nom_champ2>;
    ...
    <type_champn><nom_champn>;
};
```

Les champs peuvent être des variables ordinaires, des pointeurs, des tableaux, ou autres structures.

Les Structures

Exemple :

```
struct article
{
    int numero, qte;
    float prix;
};
```

- ❑ Cette structure définit un modèle de structure mais ne réserve pas de variables correspondantes à cette structure.
- ❑ Ce modèle s'appelle "**article**" et il précise le nom et le type de chacun des champs.
- ❑ Disposant de ce type de structure, les variables structures art1 et art2 peuvent être définies de la façon suivante : **struct article art1, art2 ;**
- ❑ Dans ce cas art1 et art2 sont déclarés comme des variables de type "**article**".
- ❑ Il est possible de condenser en une seule écriture la déclaration de variable de type structure et la composition de ce type de structure.

```
struct article
{
    int numero;
    int qte;
    float prix;
} art1, art2;
```

Les Structures

D'où la syntaxe générale :

```
struct nom
{
    <type_champ1><nom_champ1>;
    <type_champ2><nom_champ2>;
    ...
    <type_champn><nom_champn>;
} variable_1, variable_2, ... , variable_n ;
```

❑ Comme les tableaux, une variable structure peut être initialisée directement lors de sa déclaration :

Syntaxe:

```
struct nom variable = {valeur_1, valeur_2, ..., valeur_n};
```

Exemple:

```
struct article art1= {100, 285, 900} ;
```

Les Structures

3. Utilisation d'une structure

En C, il est possible d'utiliser une structure de deux manières :

- ❑ en travaillant individuellement sur chacun de ses champs,
- ❑ en travaillant de manière "globale" sur l'ensemble de la structure.

3.1. Utilisation des champs d'une structure

- ❑ `art1.numero = 15;`
→ Affecte la valeur 15 au champ numéro de la structure **art1**.
- ❑ `printf("%f", art1.prix) ;`
→ Affiche la valeur du champ prix de la **structure art1**.
- ❑ `scanf("%f", &art2.prix) ;`
→ Lit une valeur qui sera affectée au champ prix de la **structure art2**.
- ❑ `art1.numero++;`
→ Incrémente de 1 la valeur du champ numéro de la **structure art1**.

Les Structures

3.2. Utilisation globale d'une structure

- ❑ Il est possible d'affecter à une structure le contenu d'une structure défini à partir du même modèle.

Exemple :

Si art1 et art2 ont été déclarés suivant le modèle "**article**" défini précédemment :
`art1=art2;` Une telle affectation globale remplace :

```
art1.numero = art2.numero;  
art1.qte = art2.qte;  
art1.prix = art2.prix;
```

→ Cette affectation est possible que si les structures ont été définies avec le même modèle.

Les Structures

4. Déclaration de types synonymes

- ❑ Ce type de déclaration peut être simplifié par la déclaration ***typedef*** qui permet de définir des types synonymes aux types utilisés.
- ❑ Donc possibilité de créer des propres types : Il suffit de les déclarer en début de programme (après les déclarations des bibliothèques et les "***define***").

Syntaxe :

```
typedef <type> <nom_synonyme_1> <nom_synonyme_2> ... <nom_synonyme_n> ;
```

Exemple 1 :

```
typedef int entier;
```

→ Signifie que ***entier*** est synonyme de ***int*** de telle sorte que les déclarations suivantes sont équivalentes.

Les déclarations suivantes sont équivalentes :

```
int n, p;
entier n, p;
```

Exemple 2 :

```
typedef int vecteur [3];
```

Les déclarations suivantes sont équivalentes :

```
int v[3], w[3];
vecteur v, w;
```

Les Structures

Exemple 3 :

```
struct article
{
    int numero ;
    int qte ;
    float prix ;
};
struct article art1, art2;
```

Ce type de structure peut être redéfini de deux manières :

1^{ère} **manière** :

```
struct article
{
    int numero;
    int qte;
    float prix;
};
typedef struct article art;
art art1, art2;
```

2^{ème} **manière** :

```
typedef struct
{
    int numero;
    int qte;
    float prix;
} article;
article art1, art2;
```


Les Structures

5. Structure comportant des tableaux

Soit la déclaration suivante :

```
struct personne
{
    char nom[30] ;
    char prenom[20] ;
    float heures[31] ;
} employe, directeur;
```

Celle-ci réserve les emplacements pour 2 structures ***employe*** et ***directeur***.

❑ **`employe.heures[4]`**

→ Désigne le 5^{ème} élément du tableau heures de la structure ***employe***.

❑ **`employe.nom[0]`**

→ Représente le 1^{er} caractère du champ nom de la structure ***employe***.

❑ **`&directeur.heures[4]`**

→ Représente l'adresse du 5^{ème} élément du tableau heures de la structure ***directeur***.

❑ **`directeur.nom`**

→ Représente l'adresse du tableau ***nom***.

Les Structures

6. Tableaux de Structures

Exemple :

```
typedef struct
{
    char nom ;
    int x,y ;
} point;
point courbe[50] ; /*Déclaration*/
```

❑ La structure **point** pourrait, par exemple, servir à représenter un point d'un plan, **point** qui serait défini par son nom et ses coordonnées.

❑ La structure courbe représente un ensemble de 50 points.

❑ **Courbe[i].nom** → Représente le nom du point du rang *i* du tableau **courbe**.

N.B : **courbe.nom[i]** n'a pas de sens

❑ **Courbe[i].x** → désigne la valeur du champ x de l'élément de rang *i* du tableau courbe.

❑ **courbe[4]** → Représente la structure de type point correspondant au 5^{ème} élément du tableau courbe.

❑ Initialisation :

```
point courbe[50] ={{ 'A' ,10,-4},{ 'M' ,2,5},{ 'P' ,-4,-1}} ; 10
```

Les Structures

7. Structures comportant d'autres Structures

Exemple :

```
typedef struct
{
    int jour;
    int mois;
    int annee;
} date;
typedef struct
{
    char nom[30];
    char prenom[30];
    float heures[31];
    date date_embauche;
    date date_poste;
} personne;
personne employe, directeur;
```

- ❑ `employe.date_embauche.annee ;`
→ Représente l'année d'embauche correspondant à la structure **employé**. Il s'agit d'une valeur de type int.
- ❑ `directeur.date_embauche ;`
→ Représente la date d'embauche correspondant à la structure **directeur**.

Les Structures

8. Transmission d'une structure en valeur de retour d'une fonction

```
#include<stdio.h>
typedef struct
{
    int a;
    float b;
}point;

point modif()
{
    point s;
    s.a = 0;
    s.b = 1;
    printf("Dans fct : %d\t %.2f\n", s.a, s.b);
    return s;
}
```

Les Structures

```
void main()
{
    point x;
    x.a = 1;
    x.b = 12.5;
    printf("Avant appel de fct : %d\t %.2f\n", x.a, x.b );
    x = modif();
    printf("Après appel de fct : %d\t %.2f\n", x.a, x.b );
}
```

La fonction "modif" retourne une variable de type "point".

Résultat :

Avant appel de modif :	1	12.50
Dans modif :	0	1.00
Après appel de modif :	0	1.00

Les Structures

9. Passage d'une structure en paramètre pour une fonction

```
#include<stdio.h>

typedef struct
{
    int a;
    float b;
}point;

void modif(point s)
{
    s.a = 0;
    s.b = 1;
    printf("Dans fct : %d\t %.2f\n", s.a, s.b);
}
```

Les Structures

9. Passage d'une structure en paramètre pour une fonction

```
void main()
{
    point x;
    x.a = 1;
    x.b = 12.5;
    printf("Avant appel de fct : %d\t %.2f\n", x.a,
x.b );
    modif(x);
    printf("Après appel de fct : %d\t %.2f\n", x.a,
x.b );
}
```

Résultat :

Avant appel de fct : 1 12.50
Dans fct : 0 1.00
Après appel de fct : 1 12.50