Ecole Polytechnique Sousse
Département d'Informatique
NIVEAU : 3^{eme} GI, TR, IG
AU : 2020-2021

Programmation C

Les structures

1 Introduction

Nous avons déjà vu comment le tableau permettait de désigner sous un seul nom un ensemble de valeurs de même type, chacune d'entre elles étant repérée par un indice.
La structure, quant à elle, va nous permettre 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 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.
Exemple :

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

- Cette structure définie 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 définis 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 numéro ;
    int qte ;
    float prix ;
} art1 , art2 ;
```

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 3,..., valeur n } ;
Exemple : struct article art1= { 100, 285, 900} ;

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.numéro = 15 ;
$\Longrightarrow$ Affecte la valeur 15 au champ numéro de la structure art1.
— printf ("%e", art1.prix) ;
$\Longrightarrow$ Affiche, suivant le code format %e, la valeur du champ prix de la structure art1.
— scanf ("%e", &art2.prix) ;
$\Longrightarrow$ Lit, suivant le code format %e, une valeur qui sera affectée au champ prix de la structure art2.
— art1.numéro ++ ;
$\Longrightarrow$ Incrémente de 1 la valeur du champ numéro de la structure art1.

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 avantageusement :

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

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

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, le programmeur a la possibilité de créer ses 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 ;
2. typedef int vecteur [3] ;
   Les déclarations suivantes sont équivalentes :
   int v[3], w[3] ; vecteur v, w ;
3. ```
   struct article
   {
     int numero;
     int qte;
     double prix;
   };
   struct article art1, art2;
   ```
   Ce type de structure peut être redéfini de 2 manières :
   1^{ere} ***manière :***

   ```
   struct produit
   {
       int numero;
       int qte;
       double prix;
   };
   typedef struct produit prod;
   prod art1, art2 ;
   ```

   2^{eme} ***manière :***

   ```
   typedef struct produit
   {
       int numero;
       int qte;
       double prix;
   }prod;
   prod  art1, art2 ;
   ```
   ```

5 Imbrication de structures

5.1 Structure comportant des tableaux

Soit la déclaration suivante :

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

$\Longrightarrow$ Celle-ci réserve les emplacements pour 2 structures employé et courant.

- employé.heures[4]
  $\Longrightarrow$ Désigne le 5^{eme} élément du tableau heures de la structure employé.
- employé.nom[0]
  $\Longrightarrow$ Représente le 1^{er} caractère du champ nom de la structure employé.
- &courant.heures[4]
  $\Longrightarrow$ Représente l'adresse du 5^{eme} élément du tableau heures de la structure courant.
- courant.nom : Représente l'adresse du tableau nom.

5.2 Tableaux de Structures

Exemple :

```
typedef struct point
{   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
  $\Longrightarrow$ Représente le nom du point du rang i du tableau courbe.
  N.B : courbe.nom[i] n'a pas de sens
- Courbe[i].x
  $\Longrightarrow$ désigne la valeur du champ x de l'élément de rang i du tableau courbe.
- Courbe[4]
  $\Longrightarrow$ Représente la structure de type point correspondant au 5^{eme} élément du tableau courbe.
- Initialisation : point courbe[50] ={{'A',10,-4},{'M',2,5},{'P',-4,-1}} ;

5.3 Structures comportant d'autres Structures

Exemple :

```
struct date
{   int jour;
    int mois;
    int année;
};
```

```
typedef struct enreg
{   char nom[30];
    char prenom[30];
    float heures[31];
    struct date date_embauche;
    struct date date_poste;
} personne;
personne employé, courant;
```

- employé. date_embauche.année ;
  $\Longrightarrow$ Représente l'année d'embauche correspondant à la structure employé. Il s'agit d'une valeur de type int.
- Courant. date_embauche ;
  $\Longrightarrow$ Représente la date d'embauche correspondant à la structure courant.

5.4 Transmission d'une structure en valeur de retour d'une fonction

```
#include<stdio.h>
typedef struct enreg
{
    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;
}
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 struct enreg.
Résultat :
Avant appel de modif : 1 12.50
Dans modif : 0 1.00
Après appel de modif : 0 1.00