

Langage C

Les structures

L2 Mathématique et Informatique

Université de Marne-la-Vallée

Structures en C

- ▶ Elles permettent de regrouper sous le même nom des éléments de type différent;
- ▶ on construit un type structure avec le mot clef `struct`;
- ▶ on obtient un synonyme du type `typedef`;

Exemple : fiche d'un étudiant

nom	Dupont
naissance	1987
moyenne	18.5

Définition du type `struct perso`

```
1 struct perso {  
2     char nom[16];  
3     int naissance;  
4     float moyenne;  
5 };
```

`nom`, `naissance`, `moyenne` sont les *membres* (ou champs) de la structure.

`perso` en est le *label*.

Déclaration de variables, opérations

On a défini le type struct perso.

- ▶ Déclaration

```
1 struct perso anne, jean;  
2 struct perso x = { "Dupont", 1987, 18.5};
```

- ▶ Opérations

Seules opérations possibles sur les structures :

- ▶ accès aux membres (.)

```
1 anne.naissance = 1985;  
2 strcpy (anne.nom, "Durand");
```

- ▶ affectation

```
1 jean = x;
```

une structure peut être argument d'une fonction, et être valeur de retour d'une fonction.

- ▶ prise d'adresse (&)

```
1 float bonus (struct perso * x) {  
2     return (*x).moyenne +=.5;  
3 }  
4 printf ("%f\n", bonus(&jean));
```

Définition d'un synonyme du type

Un type peut être nommé au moyen de `typedef` (accroît la lisibilité)

```
1 struct perso {  
2     char nom[16];  
3     int naissance;  
4     float moyenne;  
5 } Personne;  
6 typedef struct perso Personne;  
7 Personne jean;
```

`Personne` est un synonyme de `struct perso`. On peut définir le synonyme dès la déclaration du type:

```
1 typedef struct perso {  
2     char nom[16];  
3     int naissance;  
4     float moyenne;  
5 } Personne;  
6 Personne jean;
```

Dans cette déclaration, on aurait pu omettre le label `perso`.

Utilisable avec d'autres types

```
1 typedef unsigned int Entiernat;  
2 typedef int Tableau[MAX] ;
```

Entiernat est le type unsigned int

Tableau est le type tableau de MAX int

PRINCIPE : ce qui était une déclaration de variable sans typedef:

```
int t[MAX]; /* t est un tableau de MAX entiers*/
```

devient une déclaration de type

```
typedef int Tableau[MAX] Tableau est le type tableau de MAX  
int
```

Deux types sont compatibles s'ils ont la même spécification après développement des noms de types définis par typedef et normalisation (par exemple, remplacement de long par long int).

Espace de noms

- ▶ Un même identificateur peut être utilisé pour désigner le type d'une structure, son label et un membre;
- ▶ un même identificateur peut être utilisé pour une variable, un label et un membre;
- ▶ un type et une variable ne peuvent pas être désignés avec le même identificateur;
- ▶ deux types de structures peuvent avoir un même nom de membre (car les espaces de noms sont disjoints).

Les champs peuvent être de tout type

```
1  typedef struct {
2      int quantieme;
3      char mois[10];
4      int annee;
5  } Date
6  typedef struct {
7      char nom[16];
8      Date naissance;
9      long salaire;
10 } Personne;
11 Personne jean;
12 jean.naissance.annee=2015;
```

Tous les champs peuvent être de même type

```
1  typedef struct {
2      float reelle;
3      float imaginaire;
4  } Complexe;
```

Plus lisible que

```
1  typedef Complexe[2];
```

Utilisations

- ▶ Lorsque plusieurs informations concernent le même élément, il faut les grouper dans une structure:

```
1 typedef struct {  
2     Personne participants [MAX];  
3     int nombre;  
4 } Classe ;
```

MAX est le nombre maximum de Personne dans une Classe,
nombre est leur nombre effectif.

- ▶ Lecture avec scanf
Chaque champ doit être transmis par adresse à scanf et lu séparément avec le format correspondant.


```

1  typedef struct {
2      int quantieme;
3      char mois[10];
4      int annee;
5  } Date
6  void lireDate(Date *d){
7      printf("entrez une date\n le numero du jour :");
8      scanf("%d",&(*d).quantieme));
9      printf("le nom du mois :");
10     scanf("%s",&(*d).mois);/* c'est un tableau */
11     printf("l'annee :");
12     scanf("%d",&(*d).annee));
13 }

```

d est l'adresse de la structure;

*d est la structure;

(*d).quantieme est le champ de la structure;

&(*d).quantieme) est l'adresse du champ de la structure.