



Université Internationale
de Casablanca

UNIVERSITÉ RECONNUE PAR L'ÉTAT

Programmation Structurée 2 (Suite)

Pr. EL OUKKAL Sanae



NOTION DE STRUCTURE

Introduction

- La structure « tableau » permet de regrouper un certain nombre de donnée de même type, et de plus le type doit être assez simple (pour l'instant).
- Les types structurés vont nous permettre de manipuler des données structurées dont les types des éléments peuvent être différents.

Définition

- Une structure permet de rassembler sous un même nom(même éléments) des informations de types différents.
- Une structure peut contenir des données entières, flottantes, tableaux , caractères, pointeurs, etc...
- Ces données sont appelées ‘champs’ ou ‘membres de la structure’.

Déclaration d'une Structure

```
struct nom_de_la_structure{  
    typemembre 1 nommembre 1 ;  
    typemembre 2 nommembre 2 ;  
    ...  
    typemembre n nommembre n;  
};
```

N.B.:

- Les règles à respecter pour les noms des structures sont les mêmes que pour les noms de variable et de fonction.

Exemple de Déclaration

```
struct Personne{  
    char[10] Nom;  
    char[10] Prénom;  
    int age;  
};  
Personne p={"BLABLA", "blabla",18};
```

- *Remarque* : L'ordre de déclaration des structures est important

Autre Syntaxe de Déclaration d'une Structure

```
typedef struct {  
    déclarations des champs;  
}nom;
```

- Dans ce cas, *nom* devient un nouveau type.

Déclaration d'une variable de type Structure

- La déclaration d'une variable de type structure s'effectue ainsi:

```
struct nom_type nom_var;
```

Accès d'un Champ de la Structure

- L'accès à un champ d'une structure s'effectue comme suit:

`nom_var.nom_champ;`

- Restrictions sur les champs:
 - On peut mettre comme champ d'une structure tout ce dont le compilateur connaît: le type.
 - On ne peut donc pas mettre la structure elle-même. (sauf si on utilise des pointeurs).

Accès à un Champ d'une Structure

- Exemple:
 - Si `personne` est une variable de type structure:
 - `personne.taille` accède au champ `taille` de cette variable,
 - Si `pers` est un pointeur sur une variable de type structure:
 - `pers->taille` accède au champ `taille` de la variable pointée par `pers`,

Imbrication des Structures

- Un champ peut être une structure:

```
struct id{  
    char firstname[MAX];  
    char lastname[MAX];  
};  
struct people {  
    struct id identifier;  
    int age;  
};
```

Remarque 1

- `char* strcpy(char* dest, const char* src)` de la librairie `<string.h>` copie la chaîne *src* vers *dest*.

Remarque 2

- Deux structures peuvent se référencer l'une à l'autre, à condition d'utiliser des pointeurs .

```
struct state {  
    struct transition *p_t;  
    char final;  
};  
  
struct transition {  
    char letter;  
    struct state * p_dest;  
};
```

Opération sur les Structures

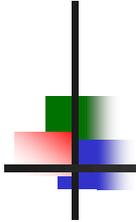
- L'affectation d'une structure fonctionne et produit la recopie des valeurs des champs.
- Syntaxe:
 - `struct z={'a',145,'y','u'};`
 - `struct t=z;`
 - La structure t contiendra toutes les informations de la structure z

Opération sur les Structures

- La comparaison directe de deux informations de type structure:

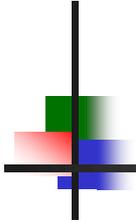
```
if ( struct1 == struct2 ) printf("C'est pareil") ;
```

n'est pas permise.
- En C, il faut essayer de comparer partie par partie nos deux structures.



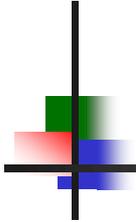
Les structures en Paramètres

- Ce sont des copies des structures qui sont passées en paramètre.
 - Elles peuvent occuper beaucoup de place en mémoire.
- Il faut transmettre leur adresse sous forme de pointeur



Les structures en Paramètres

- L'utilisation de pointeur vers les structures permet:
 - Un gain de temps,
 - Un gain d'espace mémoire,
 - La modification de la structure pointée.



TABLEAUX DE STRUCTURE

Tableau de Structure

■ Justification

- Supposons que nous avons beaucoup d'informations se rapportant à un même objet et que nous devons traiter plusieurs de ces objets.
 - **int** **marque**[MAX];
 - **int** **modele**[MAX];
 - **int** **cassette**[MAX];
 - **int** **cd**[MAX];
 - **int** **mp3**[MAX];
 - **float** **prix**[MAX];
- En utilisant un tableau pour chaque information, on obtient 6 tableaux.

Tableau de Structure

- L'utilisation de tableaux à 2 dimensions peut diminuer un peu le nombre de tableaux à gérer
 - **int** **marque**[MAX];
 - **int** **modele**[MAX];
 - **int** **options**[MAX][3];
 - **float** **prix**[MAX];
- On obtient ainsi 4 tableaux.

- Il serait intéressant de pouvoir contenir toute l'information dans un seul tableau.

Tableau de Structure

- Déclaration de la structure:

```
typedef struct{
```

```
int marque;
```

```
int modele;
```

```
int cassette;
```

```
int cd;
```

```
int mp3;
```

```
float prix;
```

```
} radio;
```

Tableau de Structure

- Déclaration d'un tableau de structures:
 - **radio tabRadio[MAX];**
 - Cette déclaration réserve de l'espace pour MAX variables de type radio.
- **Avantage:**
 - Un seul tableau à passer en paramètre à des fonctions,

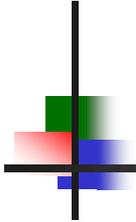


Tableau de Structure

- Désavantage:
 - On ne peut plus passer un tableau contenant une seule information.
 - Les fonctions qui doivent faire un traitement sur une information en particulier de la structure doivent gérer l'accès à celle-ci. En particulier si le traitement est le même mais pour 2 champs différents de même types.
 - Ex: une fonction qui compte et retourne le nombre de radios ayant une option donnée.

Compréhension de Base

- Exemple:

```
typedef struct
{
    char  sexe ;
        float  taille, poids ;
}Personne ;
Personne pers[MAX_PERS] ;
```

Compréhension de Base

- Avec ces déclarations :
 - pers est un tableau des personnes ;
 - pers[0], pers[1], ..., pers[99] sont 100 éléments du tableau pers. Chacun est une variable de type structure nommé "Personne" ;
 - pers[15].sexe est le sexe de la 16^{ième} personne ;

Compréhension de Base

- La déclaration : `Personne * P` ; rend valide l'affectation suivante : `P = pers` ;
- Cette affectation est équivalente à : `P = &pers[0]`.

De plus, `*(P + i)` est équivalent à `pers[i]`. Ainsi :

`*(P + i).taille` est `pers[i].taille`

Donc, `(P + i) -> taille` est `pers[i].taille`



BONNE CHANCE