Filière IGE – 2017/2018
Algorithmique & Programmation
Leçon 8 : Structures

Prof. *A. DARGHAM*

Structures

- **Définition**
  - Une **structure** est une **collection de plusieurs éléments** de **noms et de types différents formant une unité unique**.
  - Les **éléments** d'une **structure** s'appellent des **champs** (ou **membres**).
  - Chaque **champ** est **identifié** par un **nom** et il est d'un certain **type**.
  - Une **structure** est un **regroupement** de ses **champs**.

Structures

- **Définition**
  - Par exemple, les **informations relatives** à un **livre** sont en particulier :
    - **L'auteur** (une **chaîne de caractères**);
    - Le **titre** (une **chaîne de caractères**);
    - **L'année** (un **nombre entier**);
    - Le nombre **de pages** (un **nombre entier**);
    - Le **prix** (un **nombre réel**).

Structures

- **Définition**
  - Au lieu de définir 5 **variables dispersées** :

```
char auteur[50];
char titre[250];
int annee;
int nb_pages;
float prix;
```

  - Il est **très bénéfique** de les **regrouper au sein d'une seule unité** : une **structure** « livre ».

Structures

- **Déclaration**
  - La **structure** « livre » **se déclare** en C de la manière suivante :

```
struct
{
    char auteur[50];
    char titre[250];
    int annee;
    int nb_pages;
    float prix;
} Livre;
```

Champs de la structure

Variable structure

Structures

- **Déclaration**
  - **Le nom** « `Livre` » désigne une **structure** : c'est une **variable composée** de 5 **variables simples** qui constituent ses **champs**.
  - Pour **référencer** le **champ** « `auteur` » de la **structure** « `Livre` », on écrit :

```
Livre.auteur
```

Structures

- **Initialisation d'une structure :**
  - Il est possible **d'initialiser** une **structure** au **moment de sa déclaration** en **introduisant les valeurs de ses champs** (dans l'ordre) entre deux accolades { }, comme dans un tableau.
  - Par exemple, pour créer un livre d'auteur « **Cay S. Horstmann** », de titre « **C++ pour les programmeurs C et PASCAL** », d'année d'édition **1993**, de nombre de pages **336**, et de prix **700.00** DH, on écrit :

Structures

- **Exemple :**

```
struct
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Livre = {"Cay S. Horstmann", "C++ pour les
  programmeurs C et PASCAL", 1993, 336, 700.00};
```

Structures

- **Accès aux champs d'une structure**
  - Pour accéder à un champ d'une structure, on utilise la syntaxe suivante :

```
Nom_structure.Nom_champ
```

  - Par exemple, si l'on veut changer l'année de l'édition du livre à 2000, on écrira l'instruction suivante :

```
Livre.annee = 2000;
```

Structures

- **Accès aux champs d'une structure**
  - Les **champs d'une structure** sont des **variables ordinaires**, on peut :
    - Lire leurs valeurs;
    - Les afficher;
    - Leur appliquer des opérations (selon leurs types).
  - Le programme suivant, lit les informations sur un livre en les conservant dans une structure, puis affiche ces informations :

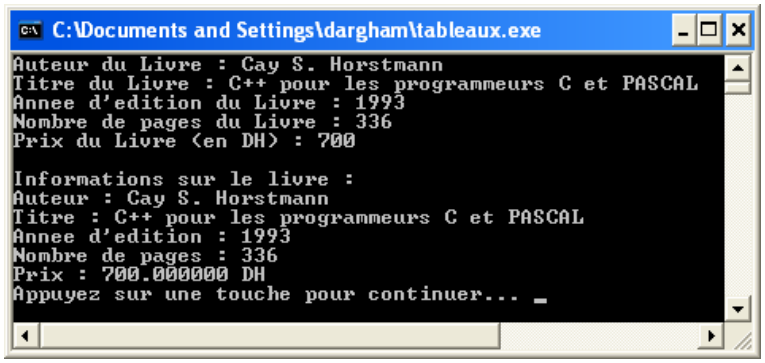
Structures

```
#include <stdio.h>
struct
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Livre;
main()
{
   printf("Auteur du Livre : ");
   gets(Livre.auteur);
   printf("Titre du Livre : ");
   gets(Livre.titre);
```

Structures

```
   printf("Annee d'edition du Livre : ");
   scanf("%d", &Livre.annee);
   printf("Nombre de pages du Livre : ");
   scanf("%d", &Livre.nb_pages);
   printf("Prix du Livre (en DH) : ");
   scanf("%f", &Livre.prix);
   printf("Informations sur le livre : \n");
   printf("Auteur : %s\n", Livre.auteur);
   printf("Titre : %s\n", Livre.titre);
   printf("Annee d'edition : %d\n", Livre.annee);
   printf("Nombre de pages : %d\n", Livre.nb_pages);
   printf("Prix : %f DH\n", Livre.prix);
   system("pause");
}
```

Structures

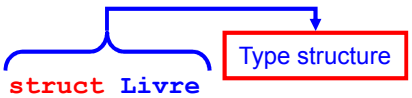


Structures

- **Type structure :**
  - Le fait de déclarer une **structure** selon la **syntaxe précédente ne permet pas de définir** un **nouveau type de données**.
  - Dans l'exemple précédent, le nom `Livre` désigne une **variable** et **non pas un type**.
  - Pour transformer l'identifiant Livre à un nom de type, il faut l'écrire juste après le mot-clé `struct` :

Structures

- **Exemple :**




```
struct Livre
{
    char auteur[50];
    char titre[250];
    int annee;
    int nb_pages;
    float prix;
};
```

Structures

- **Type structure :**
  - Maintenant, **l'identificateur `Livre` désigne un type structure** (ce n'est plus une variable).
  - La **première syntaxe** que l'on a vu crée ce qu'on appelle une **variable structure anonyme**.
  - Avec la **deuxième syntaxe**, il est aussi **possible de déclarer des variables structures** ayant comme **type la structure** « `Livre` » **au moment de la déclaration** ou **en différé** :

Structures

- **Exemple :**

```
struct Livre
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
}L1, L2;

struct L3, L4;
```

Type structure

Variables structures

Structures

- **Type structure :**
  - La **présence du mot-clé** « **struct** » est **obligatoire** dans cette forme de déclaration.
  - **L'initialisation d'une variable structure** est **également possible** avec cette syntaxe :

```
struct Livre L = {"Linton Stone", "New Lower
  Cambridge English", 1971, 286, 50.0};
```

Structures

- **Type structure introduit par « typedef » :**
  - Une autre façon pour **déclarer un type structure** est d'utiliser le **mot-clé** « **typedef** » :

```
typedef struct
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Livre;
```

Type structure nommé

Structures

- **Type structure introduit par « typedef » :**
  - **L'identificateur** `Livre` désigne un **type structure** qui remplace « `struct Livre` » qui **n'est pas maintenant autorisée**.
  - Pour déclarer des variables de ce type, on écrira par exemple :

```
Livre L = {"Linton Stone", "New Lower
  Cambridge English", 1971, 286, 50.0};
```

Structures

- **Type structure introduit par « typedef » :**
  - On peut également **combiner les deux variantes :**

```
typedef struct Livre
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Book;
```

Le type « struct Livre » est renommé « Book »

Structures

- **Type structure introduit par « typedef » :**
  - Avec cette forme il y a **deux manières** pour **déclarer une variable structure**.
  - Par exemple, on peut déclarer une variable désignant un livre en utilisant soit « **struct Livre** » ou « **Book** » :

```
struct Livre L1 = {"Nino Silverio", "Langage
  C++", 1998, 613, 650.00};
Book L2 = {"GUY L. STEELE JR.", "COMMON LISP",
  1990, 1029, 1200.00};
```

Structures

- **Imbrication de structures :**
  - Le **champ d'une structure** peut lui aussi être de **type structure** : c'est **l'imbrication de structures**.
  - Par exemple, imaginons un employé identifié par un **nom**, un **prénom**, une **date de naissance** (**jour** / **mois** / **année**) et un **salaire**.
  - Le **champ date de naissance** est déjà une **variable composée** : c'est une **structure**.

Structures

```
typedef struct
{
   int jour;
   int mois;
   int annee;
} Date;
typedef struct
{
   char nom[50];
   char prenom[50];
   Date date_naissance; /* Imbrication de structures */
   float salaire;
} Employe;
```

Structures

- **Imbrication de structures :**
  - Voici un exemple de déclaration, d'initialisation et de manipulation de telles structures :

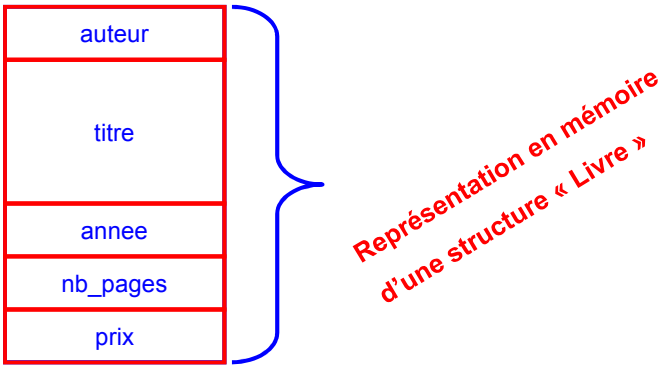
```
Employe e = {"Jean", "Pascal", {15, 9, 1975},
  9000};
printf("Mois de naissance : %d\n",
  e.date_naissance.mois);
```

Structures

- **Représentation en mémoire d'une structure :**
  - Les **champs d'une structure** sont **stockés en mémoire** dans un **espace contigu**, l'un après l'autre et dans l'ordre de leur déclaration.
  - La **taille d'une structure** est (au moins) égale à la **somme des tailles de ses champs**.
  - Par exemple, une structure de type Livre déclaré dans les sections précédentes occupera une **zone mémoire continue** d'au moins :

    50 + 250 + 4 + 4 + 4 = 312 octets.

Structures



Structures

- **Représentation en mémoire d'une structure :**
  - **L'adresse mémoire** d'une **structure** est **l'adresse du premier octet de son premier champ**.
  - Ainsi, l'adresse d'une structure « `L1` » de type « `Livre` » est égale à l'adresse de « `L1.auteur` », c'est-à-dire :

```
&L1 = &L1.auteur
```

Structures

- **Affectation globale de structures :**
  - **Contrairement aux tableaux**, Il est possible **d'affecter** une **structure à une autre**.
  - Ceci réalise une **copie de tous les champs** de la **structure** (**copie membre à membre**).

Structures

- **Exemple :**

```
typedef struct
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Livre;
Livre L1 = {"New Lower Cambridge English",
  "Linton Stone", 1971, 286, 350.00};
Livre L2 = L1;
/* Affectation globale : copie membre à membre */
```

Structures

- **Tableaux de structures :**
  - Il est possible de déclarer un tableau dont les éléments sont des structures, par exemple :

```
Livre Mes_Livres[100];
/* Mes_Livres est un tableau de 100 Livres */
puts(Mes_Livres[0].auteur);
/* affiche l'auteur du premier livre */
gets(Mes_Livres[99].titre);
/* lit le titre du dernier livre */
float Total = 0.0 , Moyenne;
for(i = 0; i < 100; i++)
   Total += Mes_Livres[i].prix;
Moyenne = Total / 100.0;
/* Moyenne des prix des 100 livres */
```

Structures

- **Exemple :**
  - Le morceau de code suivant qui affiche les titres de tous les livres dont le nombre de pages n'excède pas 100 pages :

```
Livre Mes_Livres[100];
int i;

for(i = 0; i < 100; i++)
{
   if(Mes_Livres[i].nb_pages <= 100)
     puts(Mes_Livres[i].titre);
}
```

Structures

- **Pointeurs sur structures :**
  - Un **pointeur sur une structure stocke l'adresse d'une variable structure** (c'est-à-dire **l'adresse du premier octet de son premier champ**).
  - L'opérateur "**->**" permet **d'accéder au champ d'une structure** via **un pointeur pointant vers cette structure**.
  - Par exemple, le programme suivant déclare un pointeur sur une structure de type « Livre » :

Structures

```
#include <stdio.h>

typedef struct
{
   char auteur[50];
   char titre[250];
   int annee;
   int nb_pages;
   float prix;
} Livre;

main()
{
   Livre L = {"Nino Silverio", "Langage C++", 1998,
   613, 700.00};
```

Structures

```
    Livre * pL;  /* pointeur sur une structure Livre */

    pL = &L; /* pL pointe sur L */
    printf("Titre du Livre : %s\n", L.titre);
    printf("Titre du Livre : %s\n", (*pL).titre);
    printf("Titre du Livre : %s\n", pL->titre);
    system("pause");
}
```

Structures