COURS

Structures de Données

Chapitre :

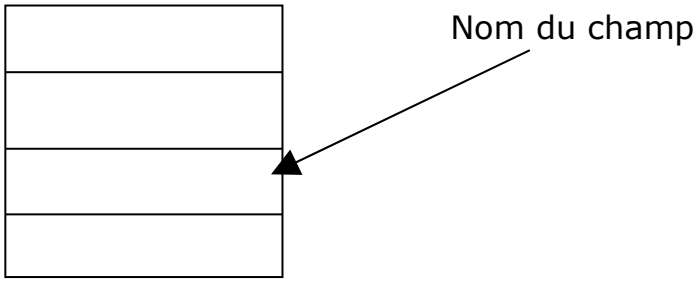
Le Type Enregistrement

1 Le type enregistrement

1.1 Définition

Le **tableau** permet de désigner sous un seul nom un ensemble de valeurs de **même type**, repérées par des indices.

Un enregistrement, appelé **structure** En langage C, est une variable complexe qui permet de désigner sous un seul nom un ensemble de valeurs pouvant être de **type différent.**



- ✓ Chaque élément de la structure est nommé **champ**

- ✓ L'accès à un champ se fait par son nom dans la structure

1.2 Déclaration d'une structure

Lors de la déclaration de la structure, on définit un modèle où on indique les champs de la structure, c'est-à-dire le type et le nom des variables qui la composent :

Syntaxe:

```
struct  <Nom_Structure> {
                    <type_champ1> <Nom_Champ1>;
                    <type_champ2> <Nom_Champ2>;
                    <type_champ3> <Nom_Champ3>;
                        ...
                    };
```

Exemple:

```
struct  Etudiant   {
            char Nom[30];
            int Age;
            float MoyenneScolaire;
            };
```

REMARQUES :

- ✓ Le nom des champs répond aux critères des noms de variables
- ✓ Deux champs ne peuvent avoir le même nom
- ✓ Les données peuvent être de n'importe quel type hormis le type de la structure dans laquelle elles se trouvent
- ✓ La déclaration d'une structure ne fait que donner l'allure de la structure, c'est-à-dire en quelque sorte une définition d'un type de variable complexe.
  La déclaration ne réserve donc pas d'espace mémoire pour une variable structurée (variable de type structure).

Il faut donc définir une (ou plusieurs) variable(s) structurée(s) après avoir déclarée la structure...

1.3 Définition d'une variable de type structure

La définition d'une variable structurée est une opération qui consiste à créer une variable ayant comme type celui d'une structure que l'on a précédemment déclaré, c'est-à-dire la nommer et lui réserver un emplacement en mémoire.

Une variable structurée doit être définie comme suit :

struct <Nom_Structure> <Var_Structure>;

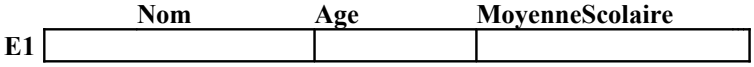
Exemples:

```
struct Etudiant {
        char Nom[12];
        int Age;
        float MoyenneScolaire;
        };
struct Etudiant E1, E2;
```

Ou bien

```
struct Etudiant {
        char Nom[12];
        int Age;
        char Sexe;
        float MoyenneScolaire;
        }E1, E2;
```

En mémoire, en déclarant une variable E1, l'espace réservé aura l'allure suivante :



1.4 Initialisation d'une structure

Lors de sa déclaration on peut initialiser une structure en indiquant entre {} la liste des valeurs séparées par des virgules.

Chaque valeur étant une constante ayant le type du champ correspondant.

Exemples:

```
struct Etudiant {
        int niveau;
        float moyenne;
    };
struct  Etudiant  E1 = {2, 10.9};
struct  Etudiant  E2 =  {2, 12.5};
```

1.5 Utilisation d'une structure

Chaque variable de type structure possède des champs repérés avec des noms uniques. Toutefois le nom des champs ne suffit pas pour y accéder étant donné qu'ils n'ont de contexte qu'au sein de la variable structurée...

Pour accéder aux champs d'une structure on utilise l'opérateur de champ (un simple point) placé entre le nom de la variable structurée et le nom du champ.

Syntaxe:

<Var_Structure>.<Nom_Champi>;

Ainsi, pour affecter des valeurs à la variable **E1,** on pourra écrire:

```
E1.niveau = 2;

scanf ("%f", &E1.moyenne) ;

printf ("Moyenne = %f", E1.moyenne) ;
```

Il est clair, qu'on peut effectuer des affectations entre deux variables de même type de structure :

- ✓ soit d'une façon individuelle (champs par champs), sur chacun de leurs champs,
- ✓ soit d'une manière globale sur toute la structure.

Exemple:

Affectation individuelle :

```
Etudiant1. niveau =  Etudiant2.niveau;
Etudiant1.moyenne =  Etudiant2.moyenne;
```

Affectation Globale :

```
Etudiant1 = Etudiant2;
```

Exercice d'application 1:

Définir une fiche Etudiant. Saisir dans le programme les informations d'un étudiant et les affichées.

Informations: Nom, prénom, age, moyenne.

2 Structure comportant des tableaux

Comme on a vu précédemment, une structure peut contenir des champs de type chaîne de caractères ou bien de type tableau.

Exemple:

```
struct UnEtudiant {
        char nom [30];
        char prenom [30];
        float  notes [4];
        int niveau;
        float moyenne;
        };
struct  UnEtudiant  Etudiant1,  Etudiant2;
```

Notation:

Etudiant1.nom[0] : désigne le premier caractère du champ **nom** de l'étudiant **Etudiant1**

Etudiant2.notes[3] : désigne la quatrième note du tableau **notes** de l'étudiant **Etudiant2**

Initialisation:

```
struct UnEtudiant Etudiant1 = {"Ben Salem", "Ali", {10.0, 12.5,
13.6,  7.5}, 2, 10.9};
 struct UnEtudiant Etudiant2 = {"Selmi ", "Ahmed", {12.0, 12.5,
14.0,  11.5}, 2, 12.5};
```

Utilisation:

```
strcpy (Etudiant1.nom, "Ben Salem");
Etudiant1.notes[2] = 13.6;
printf ("%f ",  Etudiant2.notes[i]);
```

3 Structure comportant d'autres structures

une structure peut contenir aussi des champs de type structure. Cette dernière doit être différente de la première.

Exemple:

```
struct Date {
        int jour;
        int mois;
        int annee;
        };
struct UnEtudiant  {
                char nom [30] ;
                char prenom [30]  ;
                struct  Date  date_naissance ;
                };
```

Initialisation:

```
struct UnEtudiant Eudiant1 = {"Ali", "Ben Salem", {1, 1, 2000}};
```

Utilisation:

```
Etudiant1.date.jour = 1;
scanf ("%d %d %d", &Etudiant1.date.jour, &Etudiant1.date.mois,
&Etudiant1.date.annee);
```

Exercice 2:

Reprendre Exercice de cours 1 en ajoutant aux informations: date naissance et les notes de 3 matières: Structures de Données, Math et Anglais.

4 Tableaux de structures

Étant donné qu'une structure est composée d'éléments de taille fixes, il est possible de créer un tableau ne contenant que des éléments du type d'une structure donnée.

Il suffit de créer un tableau dont le type est celui de la structure et de le repérer par un nom de variable:

Déclaration:

```
struct  <Nom_Structure>
      {
            <type_champ1> <Nom_Champ1>;
            …
            <type_champN> <Nom_ChampN>;

      };

struct  <Nom_Structure>  <Tab_Struct>[N];
```

Accès:

A un élément du tableau :

<Tab_Struct>[i] (structure N° i)

A un élément d'une structure :

<TabStruct>[i].<NomChampj>

Exemple:

Soit la déclaration suivante :

```
struct Etudiant {
        char nom [30] ;
        char prenom [30] ;
        float  notes     [4] ;
        } ;

struct  Etudiant  tab [50];
```

On peut écrire les instructions suivantes :

```
for (i = 0; i < 50; i++)
{
    printf ("Entrez le nom de l 'étudiant n°%d ", i+1);
    scanf ("%s", tab[i].nom);
    printf ("Entrez le prénom de l'étudiant n°%d ", i+1);
    scanf ("%s", tab[i].prenom);
    for (j =0; j < 4; j++)
      {
        printf ("Entrez la note %d de l'étudiant n°%d ", j+1, i+1);
        scanf ("%f", &tab[i].notes[j]);
      }
}
```

5 Utilisation de « typedef »

La principale utilité des ***typedef*** est de faciliter l'écriture des programmes.

Lorsqu'on donne un nom à un type structure par ***typedef***, l'utilisation est beaucoup plus aisée. En effet, si on déclare :

```
struct Pers
  {
  ...
  };
```

les déclarations de variables se feront par :

```
struct Pers P1,P2;
```

alors que si on déclare :

```
struct  Pers
 {
  ...
 };
typedef stuct Pers Personne;
```

les déclarations de variables se feront par :

```
struct Pers P1,P2;
```

ou bien par:

```
Personne P1,P2;
```

On voit que la seconde méthode permet d'éviter la répétition du mot clé **`struct`**.

Remarque:

on peut aussi faire la déclaration comme suit:

```
typedef stuct
 {
  ...
 } Personne;
```

dans ce cas, les déclarations de variables se feront **uniquement** par :

```
Personne P1,P2;
```

6 Pointeurs sur structures

Il est possible de déclarer un pointeur sur une variable de type structure et de l'utiliser comme tout autre pointeur:

```
struct DateDef {
        int day;
        int month;
        int year;
    };
typedef struct DateDef Date;
Date  *Date_pointer;
Date  DateTaravail;
    Date_pointer = &DateTravail;

    (*Date_pointer).day = 21;
    (*Date_pointer).month = 07;
  (*Date_pointer).year = 1968;

    ++(*Date_pointer).day;
  scanf(« %d »,&(*Date_pointer).day);
    if((*Date_pointer).month == 08 )
        ......
```

Le langage C a prévu un nouvel opérateur, noté ->:

```
(*x).y
```

sera exprimé plus clairement par:

```
x->y
```

l'instruction if du programme ci-dessus sera:

```
if( Date_pointer->month == 08 )
        .....
```

Exemple:

```
#include <stdio.h>
#include <stdlib.h>
   struct DateDef {
         int day;
         int month;
         int year;
      };
   typedef struct DateDef Date;

    int main()
      {
   Date  *Date_pointer;
   Date_pointer = (struct Date*)malloc(sizeof(struct Date));

      Date_ptr->month = 9;
      Date_ptr->day = 25;
      Date_ptr->year = 1983;

      printf("Todays date is %d/%d/%d.\n", Date_ptr->month, \
                  Date_ptr->day, Date_ptr->year % 100);
      return 0;
      }
```

Exercice 1 :

Créer une structure **fiche** contenant les informations suivantes :

- ✓ Nom sur 20 caractères
- ✓ Prénom sur 20 caractères
- ✓ Age de type entier
- ✓ Note de type réel

Saisir une fiche et puis l'afficher.

```
#include <stdio.h>
struct fiche
{
	char nom[21];
	char prenom[21];
	int age;
	float note;
};
int main()
{
	fiche f;
	printf("SAISIE D'UNE FICHE \n");
	printf("NOM: ");gets(f.nom);
	printf("PRENOM: ");gets(f.prenom);
	printf("AGE: ");scanf("%d",&f.age);
	printf("NOTE: ");scanf("%f",&f.note);
	printf("\n\nLECTURE DE LA FICHE:\n");
	printf("NOM: %s PRENOM: %s AGE: %2d NOTE: %4.1f",
					f.nom,f.prenom,f.age,f.note);
	printf("\n\nPOUR SORTIR FRAPPER UNE TOUCHE ");
	return 0;
}
```

Exercice 2 :

Créer une structure **point** contenant les informations suivantes :

- ✓ num de type entier
- ✓ x de type réel
- ✓ y de type réel

Saisir 4 points les ranger dans un tableau et puis les afficher.

```
#include <stdio.h>
struct point {
        int num ;
        float x ;
        float y ;
};

int main()
{
        point p[4];    /* tableau de points */
        int i;
        float xx,yy;
        /* saisie */
        printf("SAISIE DES POINTS\n\n");
        for(i=0;i<4;i++)
                {
                printf("\n Point Nº%1d\n",i);
                p[i].num = i;
                printf("X= "); scanf("%f",&xx);
                printf("Y= "); scanf("%f",&yy);
                p[i].x = xx; p[i].y = yy;
                }

        /* relecture */
        printf("\n\nAFFICHAGE DES POINTS\n\n");
        for(i=0;i<4;i++)
                {
                printf("\nPOINT Nº%1d",p[i].num);
```

```
            printf("\nX= %f",p[i].x);
            printf("\nY= %f\n",p[i].y);
            }
        printf("\n\nPOUR SORTIR FRAPPER UNE TOUCHE ");
        return 0;
}
```