Chapitre IV : *LES TYPE STRUCTURES OU TYPES COMPOSES*

PROBLEMATIQUE

Ecrire un programme qui permet de saisir les données de 3 étudiants. Un étudiant est caractérisé par son nom, prénom, matricule, telephone, sa moyenne, son âge et son adresse.
Le programme détermine et affiche l'étudiant qui a la plus grande moyenne et l'étudiant qui a la plus petite moyenne.
Pour les caractéristiques de l'étudiant, il faudra déclarer plusieurs variables. Ce qui équivaut à faire plusieurs Aller-Retour au niveau de la mémoire et faire plusieurs réservations d'espace.
Pour plus d'organisation et d'optimisation, il serait préférable d'utiliser les structures pour regrouper les variables en bloc.

INTRODUCTION

Les objets (variables) de type structure sont comme des tableaux constitués d'un ensemble de valeurs. Mais à la différence des tableaux ces valeurs ne sont pas nécessairement de même type. ***L'accès à une valeur de la structure ne se fait pas avec l'aide d'indice mais grâce à son nom****.*

SYNTAXE *:*

```
struct nomEnregistrement
{
        type1 champ1 ;
        type2 champ2 ;
        …
        typen champn ;
} ;
struct nomEnregistrement nomVariableEnregistrement ;
```

EXEMPLE 1

*Pour définir un type structure '**PERSONNE'** composée de 3 champs : nom, prenom et age, il faut :*

```
struct PERSONNE{
        char nom[25] ;
        char prenom[50] ;
        int age ;
} ;
 struct PERSONNE x1, x2 ;
```

*Dans cette déclaration '**PERSONNE'** est le nom du type créé alors que nom, prénom et Age représente des champs de la structure. Ces champs peuvent être de n'importe quel type de base, des tableaux, des pointeurs, des structures.*

Il existe trois méthodes pour déclarer des variables de types structures :

1e méthode

```
struct PERSONNE{
char nom[25] ;
char prenom[50] ;
int age ;
} ;
 struct PERSONNE x1, x2 ;
```

2e méthode

```
struct PERSONNE{
char nom[25] ;
char prenom[50] ;
int age ;
}P1,P2 ;
struct PERSONNE X3;
```

3e méthode

```
struct {
 char nom[25] ;
char prenom[50] ;
 int age ;
}P1,P2 ;
```

Il existe deux façons d'initialiser :

Pendant la Déclaration :

```
struct PERSONNE P1= { "GAYE", "Abdoulaye", 30 };
```

Après la Déclaration :

```
struct PERSONNE P2;
 strcpy(P2.nom,"GAYE");
strcpy(P2.prenom,"Abdoulaye");
P2.age=30;
```

EXEMPLE 2

Déclarer et initialiser un tableau de trois étudiants. Un étudiant est caractérisé par matricule, nom, prénom et age.

METHODE 1

```
struct ETUDIANT {
        char nom[25];
        char prenom[50];
        char mat[10];
        int age;
};
struct ETUDIANT T[3]={
        {"GAYE", "Abdoulaye", "Et001", 19},
        {"DIOP", "Fatou", "Et002", 28},
        {"SECK", "Doudou","Et003", 21}
 };
```

METHODE 2

```
struct ETUDIANT {
        char nom[25];
        char prenom[50];
        char mat[10];
        int age;
};
struct ETUDIANT T[3];
strcpy(T[0].nom,"GAYE");
strcpy(T[0].prenom,"Abdoulaye");
strcpy(T[0].mat,"Et001");
T[0].age=19;
……..
```

<u>Remarque</u> *:*

*Pour accéder aux champs d'une variable de type structure P, il faut faire suivre du nom un point puis le nom du champ. (**nomVariableEnregistrement.champ**)*

Application 1
Écrire un programme qui permet de saisir les données de 3 étudiants. Chaque étudiant est caractérisé par nom, prénom, âge et moyenne. Le programme affiche les données de chaque étudiant puis détermine et affiche la moyenne la plus grande.

Application 2 *(2 methodes)*
Refaire l'exo 1 mais à la place de l'age mettre la date de Naissance caractérisée par jour, mois et année.

NB : Il existe une autre méthode plus simple pour définir une structure :
L'utilisation d'ALIAS ou TYPEDEF.
Syntaxe
typedef struct {

} Personne ;

Personne P ;

EXEMPLE

```
typedef struct {
        char nom[25];
        char prenom[50];
        char mat[10];
        int age;
} ETUDIANT;
ETUDIANT E ;
```