Chapitre 9

UNIONS ET STRUCTURES

Objectif:

① Définir ce qu'est une structure et une union.

② Connaître les différentes manières de manipulation des structures et des unions.

Pré requis:

Les Chapitres précédents de ce cours de langage C

PLAN

Chapitre 9

UNIONS ET STRUCTURES

I/ DEFINITION

Une structure représente une collection de données liées dont les types peuvent êtres différente. Une structure peut aussi être simultanément de données entières, en virgule flottantes ou caractères ... etc.

II/ DECLARATION D'UNE STRUCTURE

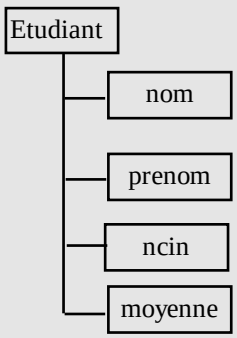
Syntaxe:

```
struct   nom{
type1   membre1;
type2   membre2;
.
.
typen   membren;
};
```

Les membres peuvent êtres des variables ordinaires, des pointeurs, des tableaux, ou d'autres structures.

Exemple:

```
struct etudiant{
char    nom[20];
char    prenom[20];
int  ncin;
float  moyenne;
};
```



Disposant de ce type de structure, les variables structures etudiants_ii2 et etudiants_ig2 peuvent êtres définis de la façon suivante:

```
struct etudiant  etudiant_ii2,etudiant_ig2;
```

Dans ce cas , etudiant_ii2 et etudiant_ig2 sont déclarées comme des variable de type etudiant.

Il est possible de condenser en une seule écriture la déclaration de variable de type structure et la composition de ce type de structure.

```
Struct   etudiant
{
char nom[80];
char prenom[80];
long int ncn;
char classe[10];
float moyenne;
}  etudiant_ii2, etudiant_ig2
```

d'ou la syntaxe générale :

```
struct  nom
{
type_1  nombre_1;
type_2  nombre_2;
. . .
type_n  nombre_n;
}   variable1, variable2, ... , variablek
```

- Comme les tableaux, une variable structure peut être initialisé directement lors de sa déclaration :

syntaxe :

```
struct  nom variable={valeur1; valeur2; ... , valeurn};
```

Exemple :

```
struct  etudiant  majeur={"xyz","uhf",5577501,"ii2,19.0};
```

- Une variable peut être aussi un tableau de structure

Exemple

```
struct  etudiant  II2[19];
```

dans ce cas II2 est un tableau de 19 étudiants.

III - MANIPULATION DES STRUCTURES

Les membres d'une structures peuvent être manipulés individuellement comme entités distinctes. Il faut donc disposer d'un moyen d'accès direct à ces membres, ce qui se fait de la façon suivant :

```
variable.membre
```

variable fait référence à la structure traitée, et membre à un membre précis de cette structure, variable et membre doivent être séparés par un point (.).

Exemple

```
etudiantii2.nom
```

Exemple

```
# include <studio.h>
struct  etudiant
{
char nom[80];
char prenom[80];
long int ncn;
char classe[10];
float moyenne;
}
main ()
{
struct etudiant  etudiant_ii2={"xyz","uhf",5577501,"ii2",19.0};
printf("\n nom=%S",etudiantii2.nom);
printf("\n prenom=%S",etudiantii2.prenom);
printf("\n NCIN=%Ld",etudiantii2.NCIN);
printf("\n classe=%S",etudiantii2.classe);
printf("\n moyenne=%f",etudiantii2.Moyenne);
}
```

IV - LES TYPES DE DONNEES PERSONNALISES (TYPEDEF).

Il est possible de créer des types de données personnalisés

syntaxe :

```
typedef  type nouveau_type;
```

Exemple

```
typedef int age;
typedef struct etudiant
        {
        .
        .
        } ;
etudiant  etudiant_ii2, etudiant_ig2;
age  age1, age2;
```

V/ LES POINTEURS SUR STRUCTURE ET ACCES AU DONNEES

soit le type suivant:

```
typedef   struct  etudiant {
                        char  nom[30];
                        char prenom[30];
                        int NCIN;
                        char classe[10];
                        float moyenne;
                    };
etudiant    *terminal
```

Ici terminal est un pointeur vers une structure de type etudiant. Pour accéder à l'élément terminal, on utilise la notation suivante:

```
nom_de_la_variable->membre;        ou        (*nom_de_la_variable).membre
```

Dans notre cas on aura donc:

```
terminal->nom;              ou          (*terminal).nom;
```

Exemple:

```
#include<stdio.h>
```

```
#include<string.h>
main()
{
int n=333;
char  t='N';
float  b=99.99;
typedef  struct  date{
                        int  mois;
                        int  jour;
                        int année;
                        };
struct  {
        int   *no_compte;
        char  *etat_compte;
        char   nom[30];
        char   *solde;
        date   der_versement;
        }client, *pc=&client;

client.no_compte=&n;
client.etat_compte=&t;
strcpy(client.nom,"ali");
client.solde=&b;
printf("%d  %c  %s   %f\n",*client.no_compte, *client.etat_compte, client.nom, *client.solde);
printf("%d  %c  %s   %f\n",*pc->.no_compte, *pc->etat_compte,pc->nom, *pc->solde);
}
```

Résultat : La même information est afficher deux fois.

VI/ LES STRUCTURES AUTOREFERENTIELLES

Parmi les membres d'une structure, on peut trouver un ou plusieurs pointeurs sur des structures de même type:

Syntaxe

```
struct  etiquette  {
        type1           membre1;
        type2           membre2;
        .               .
        .               .
        .               .
        struct  etiquette       *nom;
                };
```

Dans ce cas la structure de type etiquette contient donc un membre pointant sur une autre structure ayant le type etiquette. Ce type de structure est dite autoréférentielle. Avec ce type de déclaration on peut implanter tout les algorithmes de listes chaînées. (Vue en algorithmique).

VII/ LES UNIONS

Les union contiennent des membres de type différents mais qui partagent la même zone mémoire. Elles permettent d'économiser la mémoire utilisée, et présentent un intérêt pour les applications mettant en oeuvre des membres dont les valeurs n'ont pas à être affectés au même moment.

La syntaxe générale d'une union est la suivante:

```
union  etiquette  {
        type1   membre1;
        type2   membre2;
        .       .
        .       .
        .       .
        typek   membrek;
        };
```

La déclaration de variables de type union se fait de la même manière que pour les structures.

Exemple de déclaration:

```
union  ref  {
                char   couleur[12];
                int  taille;
```

```
                };
        struct  vetement  {
                        char fabricant[25];
                        union ref  description;
                        }chemise,blouse;
```

L'accès au membres d'une union se fait de façon analogue à ceux des structures.

ISET DE SFAX
II N2

TP N°6
PROGRAMMATION C

A.U 1997/1998 Sem 1
Cours de Mr: TAYARI Lassaad

Les enregistrements

EX N°1 Ecrire un programme en C qui permet de lire puis d'afficher un enregistrement (type struct) contenant nom, prénom et classe.

EX N°2 Soit la déclaration suivante:

```
typedef struct date {
        short int   jour;
        short int   mois;
        short int   annee;
                };
```

```
typedef struct enregistrement   {
                int valide;
                int   numéro;
                char  nom[20];
                char prenom[20];
                date  date_nais;
                long int   NCIN;
                char  classe[10];
                        };
```

on se propose de créer un programme de gestion d'enregistrements. Ce programme doit permettre d'ajouter, de consulter, de modifier et de supprimer des enregistrements.

Pour ce faire il va falloir créer un tableau de N enregistrements. Au début vous devez initialiser le champ valide de tous les enregistrements à 0 (zéro). Un enregistrement qui a le champ valide égal zéro est donc supprimé, un enregistrement qui a le champ valide égal a 1 est un enregistrement existant.

Le champ numéro est unique pour chaque enregistrement, donc il ne doit pas exister deux enregistrements ou plus qui on le même numéro (clé).

Le programme doit afficher le menu suivant:

M E N U

1- Ajouter un enregistrement
2- Consulter un enregistrement
3- Modifier un enregistrement
4- Supprimer un enregistrement
5- Supprimer tous les enregistrements
6- Fin de travail

Bon Travail