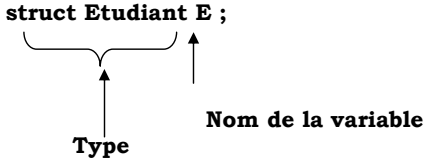
1.Définition

Une structure est un nouveau type de variable permettant de regrouper différents *champs* de types différents ou pas.

Exemple :



1.Définition de structure

En langage C, on créera d'abord un nouveau type de variables structures auquel on donnera un nom ("date", "etudiant", ...) et pour lequel on spécifiera la liste des champs (nom et type de chaque champ).

Syntaxe :

	Ou bien
struct <nom_structure>	
{	**typedef struct**
<type_champ1>	**<nom_structure>**{
<nom_champ1> ;	<type_champ1>
<type_champ1>	<nom_champ1> ;
<nom_champ1> ;	<type_champ1>
.	<nom_champ1> ;
.	.
<type_champN>	.
<nom_champN> ;	<type_champN>
} ;	<nom_champN> ;
	} **<alias>**;

Exemple1 :

struct complexe{	**typedef**	**Typedef struct**
float reel;	**struct {**	**_complexe{**
float imag;	**float reel;**	**float reel;**
};	**float imag;**	**float imag; }**
	} complexe;	**complexe;**

Struct _complexe est équivalent à **complexe.**

Exemple2 :

```
typedef struct date
{
int jour;
char mois[20];
int annee;
} date;
```

Une structure peut contenir une autre structure, qui doit avoir été déclarée avant :

```
typedef struct {
char nom[32];
char prenom[32];
date
date_naissance;
} etudiant;
```

Les champs peuvent donc être de n'importe quel type connu : types de bases, tableaux, pointeurs ou autre structure.

3- Déclaration d'une variable structure

Après avoir déclaré un type structure, on eut l'utiliser pour déclarer des variables de ce type dans nos fonctions C. Le nouveau type s'utilise comme un type de base.

Exemple :

	Ou bien
struct complexe	typedef struct
{	{
…	…
} ;	}complexe ;
void main() {	void main() {
int n;	int n;
struct complexe	**complexe**
C1,*C2;	**C1,*C2;**
…	…
}	}

Lors de la déclaration de la variable, on peut initialiser les champs, avec une notation semblable à celle utilisée pour les tableaux :

complexe Z = { 1, 0.5 }; /* Z = 1 + 0.5 i */

4- Accès aux champs d'une structure

- Si **Z** est une variable de type structure (déclarée comme **complexe Z**), **Z.reel** représente le champ **reel** de la structure **Z**.

- Si on a un pointeur sur une structure, comme par exemple complexe ***pZ**, on peut accéder aux champs de deux façons :

  - **(*pZ).reel** : car ***pZ** représente la structure pointée par **pZ** (autrement dit, dont **l'adresse est pZ**).

  - **pZ->reel** : notation spécialisée équivalente, qui est un raccourci très pratique. La flèche « -> » indique que l'on suit le pointeur pour arriver au champ.

Exemple :

```
typedef struct
{
     float reel,imag ;
 } complexe ;

void main()
{
complexe z;
complexe *pz;
pz= &z;
//saisie d'un nombre complexe
printf("partie reelle: ");
scanf("%f",&z.reel) ;//scanf("%f",&pz->reel) ;
printf("partie imaginaire: ");
scanf("%f",&z.imag) ;
//affichage du nombre complexe et de son module
printf("z=%.2f+i*%.2f\n|z|=%.2f",z.reel,z.imag,sqrt(z.reel*z.reel+z.imag*z.im
ag));
/*printf("z=%.2f+i*%.2f\n|z|=%.2f",pz->reel,pz->imag, sqrt (pz->reel *
pz->reel+ pz->imag*pz->imag));*/
}
```

5- Affectation de structures

On peut affecter une structure à une variable structure de même type :

```
complexe c1, c2 ;
.....
c1=c2 ;
```


Remarque :
Aucune comparaison n'est possible sur les structures, même pas les opérateurs == et !=.

6- Passage d'une structure en paramètre

Une fonction peut prendre une variable structure en paramètre.

On peut passer par valeur
void Affiche(comlexe c);
ou par adresse :
**void Affiche(complexe *pc
);**

On préférera toujours la deuxième solution, qui évite la duplication de la structure sur la pile (opération qui peut être coûteuse, voire impossible si la structure occupe une taille mémoire importante). Dans la fonction, on utilise alors la notation ->.

Exemple (La somme de deux complexes) :
1ère Méthode

```
//définition de la structure
typedef struct
{float reel,imag ;}complexe ;
//fonction somme de deux complexes
complexe somme(complexe *c1,complexe *c2)
{complexe s ;
s.reel=c1->reel+c2->reel ;
s.imag=c1->imag+c2->imag ;
return s;
}
//fonction principale
void main()
{complexe a={1,1},b={2,2} ,s1;
//appel de la fonction somme
s1=somme(&a,&b) ;
printf("(%.2f+i*%.2f)+(%.2f+i*%.2f)=%.2f+i*%.2f",a.reel,a.imag,
b.reel,b.imag,s1.reel,s1.imag)) ;
}
```

2ème Méthode

```
typedef struct
{float reel,imag ;}complexe,*COMP ;
//fonction somme de deux
complexes
complexe somme(COMP c1,COMP
c2)
{complexe s ;
s.reel=c1->reel+c2->reel ;
s.imag=c1->imag+c2->imag ;
return s;
}
//fonction principale
void main()
{COMP a,b;
a=(COMP)malloc(sizeof(complexe));
b=(COMP)malloc(sizeof(complexe));
complexe s1;

printf("reel(a)=");
scanf("%f",&a->reel);
printf("imag(a)=");
scanf("%f",&a->imag);
printf("reel(b)=");
scanf("%f",&b->reel);
printf("imag(b)=");
scanf("%f",&b->imag);
//appel de la fonction somme
s1=somme(a,b) ;
printf("(%.2f+i*%.2f)+(%.2f+i*%.2f)=%.2f+i*%.2f",a->reel,a->imag, b->
reel ,b->imag,s1.reel,s1.imag) ;}
```

7- Tableaux de structures

Une déclaration de tableau de structures se fait selon le même modèle que la déclaration d'un tableau dont les éléments sont de type simple.

```
complexe T[10];
```

ou bien

```
Complexe *pT ;
pT=(complexe*)malloc(10*sizeof(complexe)) ;
```