



Cours de Langage C

Les structures



Les structures

- Une **structure** est un exemple de type composite, c'est-à-dire un type de données formé de **plusieurs variables** pouvant être de **types différents**
- Cela permet au développeur de définir de nouveaux types en utilisant l'instruction **typedef**
- Cela permet de définir un type adapté au problème à traiter



Les structures

- *Exemple :*

```
struct complexe      // Définition
{
    double re ;      //-> struct est un mot réservé du langage
    double im ;      //-> complexe est le nom de la structure
};
```

- Le nouveau type de données est « struct complexe »
- re et im sont des **champs** de la structure
- On accède aux champs via l'opérateur .

- *Exemple :*

```
struct complexe z ;    // Déclaration d'un objet z de type struct complexe
z.re=1 ; z.im=2 ;      // Initialisation des « champs » de cette structure
```

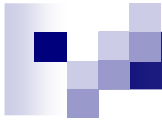


Structures et types internes

- Les éléments d'une même structure peuvent avoir des **types différents** :

```
typedef struct client
{
char        nom[100]        ;    // nom du client
char        prenom[100]     ;    // prénom du client
int         numero          ;    // le numéro du client
double      montant         ;    // somme sur son compte
double      ancien_montant[10]; // ses 10 anciens montants
} CLIENT ;
```

nom, prenom, etc. sont des **champs** de la structure



Structures et types internes

- *Exemple d'accès aux différents champs de la structure, en utilisant l'opérateur .*

client.numero = 12 ;

client.montant = 2222 ;

strcpy(client.nom , "MARTIN") ; // Fonction spécifique
// aux chaînes de caractères



Type synonyme

- On peut utiliser un type synonyme pour clarifier l'écriture :

Exemple :

```
struct complexe           // Définition DANS LE HEADER, EN TETE
{                          //-> struct est un mot réservé du langage
    double re ;           //-> complexe est le nom de la structure
    double im ;
};

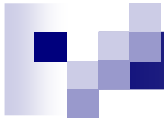
typedef struct complexe COMPLEXE ;    // par convention en
                                      MAJUSCULE
```

- *Syntaxe plus compacte :*

```
typedef struct complexe {double re ; double im ; } COMPLEXE ;
```

- void main()

```
{ COMPLEXE z ;           ... }    // Déclaration de la variable dans le main
```



Structures et fonctions

- Ce nouveau type de variable peut être utilisé exactement de la même manière que les autres types standards du langage (int, double ...)
- Il permet de typer des variables, des fonctions, des tableaux.
- On peut définir des pointeurs sur des structures (et aussi sur des tableaux de structure)
- On peut créer des structures comprenant des structures
- Exemple : Fonction **somme** de complexes

```
COMPLEXE somme (COMPLEXE z1, COMPLEXE z2)
{
    COMPLEXE z
    z.re = z1.re + z2.re;
    z.im = z1.im + z2.im;
    return z
}
```

A noter : en retournant une seule variable de type structure, on retourne en fait 2 valeurs



Structures et pointeurs

- On peut définir un pointeur sur une structure

```
COMPLEXE *z1;           // déclaration du pointeur
```

```
z1 = malloc(sizeof(COMPLEXE)); /* allocation dynamique de mémoire pour  
                                un COMPLEXE et initialisation du pointeur */
```

- Pour accéder aux différents champs de la structure via le pointeur z1, il faut utiliser l'opérateur « -> »

```
z1->re
```

```
z1->im
```




Structures et tableaux

- On peut **définir** des tableaux statiques de structures :
`// déclaration d'un tableau de 10 structures COMPLEXE`
`COMPLEXE tab[10] ;`
- Exemple **d'utilisation** : Pour accéder aux différents champs du tableau de structures on peut utiliser l'opérateur `.` Avec la notation `Tab[i]`
`tab[2].re = 5 ;`
`tab[2].im = 10 ;`
- On peut définir aussi des tableaux dynamiques de structures :
`COMPLEXE *tab;`
`tab = malloc(10*sizeof(COMPLEXE)) ;`
`// allocation mémoire pour un tableau dynamique de 10 structures complexes`
- Exemple d'utilisations avec les 2 syntaxes :

la syntaxe `.` (orientée variable)

`tab[2].re = 5 ;`

`tab[2].im = 10 ;`

⇔

⇔

syntaxe `->` (orienté pointeur)

`(tab+2)->re = 5;`

`(tab+2)->im = 10 ;`



Conclusion :

Vous savez désormais :

- Définir une structure avec différentes syntaxes
- Manipuler les structures et accéder à leurs champs
- Combiner les structures avec les fonctions ou les tableaux
- Déclarer de manière statique ou dynamique des tableaux de structures
- Utiliser des pointeurs sur des structures et accéder aux champs de ces structures avec diverses syntaxes