

Struct et typedef Travaux Pratiques – Séance n° 9

typedef

Le mot clé **typedef** sert à donner un nom différent à un type. Sa syntaxe est **typedef type₁ type₂** ; où type₂ est le nouveau nom du type, et type₁ un type existant.

Par exemple :

```
typedef struct complexe Complexe;  
Complexe c;  
  
typedef double polynome[DEGREMAX];  
polynome p;
```

Exercices sur les dates

```
typedef struct {  
    int jour, mois, annee;  
    char nomDuMois[10];  
} Date;  
  
Date d1 = { 27, 1, 2011, "janvier"};  
Date d2 = { 33, 13, 2011, "mars"};
```

À partir des déclarations précédentes, écrivez une fonction `dateValide` qui teste si une date passée en paramètre est valide ou pas.

Exercices sur les complexes

```
/* le type Complexe représente les nombres mathématiques complexes */  
typedef struct {  
    double reel;  
    double img;  
} Complexe;
```

- 1) Écrivez la fonction `ecrireComplexe` qui écrit sur la sortie standard, sous la forme (r,i) , un nombre complexe passé en paramètre.
- 2) Écrivez la fonction `initComplexe` qui retourne un nombre complexe initialisé aux valeurs des parties réelle et imaginaire passées en paramètre.
- 3) Dans la fonction `main`, testez vos deux fonctions précédentes.
- 4) Déclarez la constante `Complexe I` qui représente le complexe $(0,1)$.
- 5) Définissez les fonctions `partieReelle` et `partieImaginaire` qui retournent les valeurs des parties réelle et imaginaire d'un complexe passé en paramètre.
- 6) Complétez la fonction `main` pour tester ces deux fonctions.

On veut pouvoir réaliser les opérations mathématiques standard sur les complexes. Commençons par programmer les fonctions de conversion de la représentation polaire vers la représentation cartésienne.

Rappel : Tout complexe z admet une représentation cartésienne $x + iy$ et polaire $\rho e^{i\theta}$ où ρ est le module et θ l'argument de z . L'argument n'est défini que si $z \neq 0$. Le passage d'un système de coordonnées à l'autre se fait à l'aide des formules de conversion :

coordonnées polaires	coordonnées cartésiennes
$\rho = \sqrt{x^2 + y^2}$	$x = \rho \cos(\theta)$
$\theta = \text{atan}(y/x)$	$y = \rho \sin(\theta)$

7) Écrivez la fonction **rho** qui calcule le module d'un complexe passé en paramètre.

8) Écrivez la fonction **theta** qui calcule l'argument d'un complexe. Attention, la fonction **atan** est définie de \mathbb{R} vers $]-\frac{\pi}{2}, \frac{\pi}{2}[$. Utilisez La fonction **atan2** qui règle ce problème.

9) Testez les deux fonctions **rho** et **theta**.

10) Écrivez la fonction **polComplexe** qui possède deux paramètres de type réel représentant le module et l'argument d'un complexe polaire, et qui renvoie un objet de type complexe en coordonnées cartésiennes. Cette fonction possède l'en-tête suivant :

```
Complexe polComplexe(double rho, double theta)
```

11) Ajoutez les fonctions qui effectuent la somme, la soustraction et le produit de deux nombres complexes. Ces fonctions ont les en-têtes suivants :

```
Complexe plus(Complexe a, Complexe b)
Complexe moins(Complexe a, Complexe b)
Complexe mult(Complexe a, Complexe b)
```

Notez que le produit de deux complexes est plus simple à écrire en utilisant les coordonnées polaires :

$$\begin{aligned}\rho(z1 \times z2) &= \rho(z1) \times \rho(z2) \\ \theta(z1 \times z2) &= \theta(z1) + \theta(z2)\end{aligned}$$

Pensez à utiliser la fonction **polComplexe** pour le produit.

12) Définissez les fonctions booléennes **egal** et **différent** qui testent l'égalité et la différence entre deux complexes transmis en paramètre. On prendra soin de traiter le problème posé par l'opérateur **==** sur les réels.

13) Testez toutes les fonctions précédentes dans votre fonction **main**.