Constructeur par défaut ?

A-t-on déjà un constructeur par défaut ?

Peut-on écrire

```
Complexe z2;
```

?

☞ **NON !** Car le constructeur par défaut par défaut n'est plus fourni, vu que nous avons déclaré un autre constructeur !

$\Longrightarrow$ on doit donc ajouter un constructeur par défaut si l'on en veut un.

Veut-on un constructeur par défaut ?

☞ Oui, cela fait sens pour les complexes : typiquement 0

Jean-Cédric Chappelier
& Jamila Sam

Constructeur par défaut (1/3)

Nous avons 3 façons de le faire :

1. soit séparément, version 1 :

```
class Complexe {
public:
  // constructeurs
  Complexe(double abscisse, double ordonnee)
  : x_(abscisse),  y_(ordonnee)
  {}
  Complexe()
  : x_(0.0),  y_(0.0)
  {}
  //...
};
```

Constructeur par défaut (2/3)

Nous avons 3 façons de le faire :

2. soit séparément, version 2 (C++11 uniquement) :

```
class Complexe {
public:
  // constructeurs
  Complexe(double abscisse, double ordonnee)
  : x_(abscisse),  y_(ordonnee)
  {}
  Complexe()
  : Complexe(0.0, 0.0)
  {}
  //...
};
```


Jean-Cédric Chappelier
& Jamila Sam

Constructeur par défaut (3/3)

Nous avons 3 façons de le faire :

3. soit regroupé avec notre constructeur précédent, en donnant des valeurs par défaut aux paramètres :

```
class Complexe {
public:
  // constructeurs
  Complexe(double abscisse = 0.0, double ordonnee = 0.0)
  : x_(abscisse),  y_(ordonnee)
  {}
  //...
};
```

Note : nous avons alors dans ce cas *trois* constructeurs (écrits en un seul) !

Jean-Cédric Chappelier
& Jamila Sam

Plongement des réels ?

Comment faire pour « plonger $\mathbb{R}$ dans $\mathbb{C}$ » ?

```
Complexe z3(1.0);
```

Avec les solutions 1 et 2 précédentes :
écrire encore un troisième constructeur.

Avec la solution 3 : rien à faire, c'est déjà fait !

Constructeur de copie ?

Peut-on (déjà) construire des complexes comme ceci :

```
Complexe z4(z3);
```

Réponse : **OUI !**
Nous avons un constructeur de copie fourni par défaut.

Est-ce qu'il nous suffit (copie de surface) ?

☞ Oui ! (donc rien à faire !)

Jean-Cédric Chappelier
& Jamila Sam

Constructeur en coordonnées polaires ?

Peut-on construire des complexes comme ceci (pour $z_5 = e^{i\pi}$) :

```
Complexe z5(1.0, M_PI);
```

Question subsidiaire : si l'on décide de changer la *représentation interne* de nos nombres complexes (e.g. en polaires), comment construire un nombre par coordonnées cartésiennes ?

Réponse à la 2e question est : **exactement comme avant !** (*encapsulation*)

Réponse à la 1re question : oui, mais cela ne construit pas $e^{i\pi}$, mais bien $1 + \pi i$!

Constructeur en coordonnées polaires ? (1/2)

Comment faire un constructeur en coordonnées polaires ?

1. ne pas en faire, mais fournir une méthode :
   cf le manipulateur `polaires()` de la semaine passée
   (☞ mais cela oblige alors une *construction* en cartésiennes :-(!)

2. changer le prototype (la « *signature* ») :
   - version simple :

```
Complexe(double module, double argument, bool inutile)
: x_(module * cos(argument)),  y_(module * sin(argument))
{}
```

   Utilisation :

```
Complexe z5(1.0, M_PI, true);
```


Jean-Cédric Chappelier
& Jamila Sam

Constructeur en coordonnées polaires ? (2/2)

3. changer le prototype (la « *signature* ») :
   - version plus avancée :

```
enum Systeme { cartesiennes, polaires };

Complexe(double prems, double deuz, Systeme mode = cartesiennes)
{
  if (mode == cartesiennes) {
    x_ = prems;   y_ = deuz;
  } else {
    x_= prems * cos(deuz);  y_ = prems * sin(deuz);
  }
}
```

Utilisation : `Complexe z6(1.0, M_PI, polaires);`

4.  offrir une « *factory* » : méthode *de classe* construisant un `Complexe` avec les arguments voulus ; p.ex. :

```
static Complexe genere_polaires(double module, double argument)
{ return Complexe(module * cos(argument), module * sin(argument)); }
```