

L

Table des matières

1. LA FORME CANONIQUE DE COPLIEN	4
1.1 CONSTRUCTEUR PAR DÉFAUT	4
1.2 CONSTRUCTEUR DE RECOPIE	5
1.2.1 BUT DU CO96(C)JTJ1EUR PDU CIESIE	


```
    tab[0]=0;  
}
```

1.6

2. De la surcharge des opérateurs et de la bonne utilisation des références en général

Ce chapitre traite plus spécialement de la surcharge des opérateurs mais plus généralement de la bonne utilisation des références en C++. En effet, la plupart des

(Notez au passage l'utilisation des valeurs directes des booléens et non pas de références : `sizeof(bool) < sizeof(@) !`)

... et souvenez vous ! les arguments d'une fonction sont toujours évalués

Nous pourrions répondre :

« Il est nécessaire de renvoyer une référence lorsque l'opérateur (ou une méthode quelconque) doit renvoyer l'objet cible (i.e. `*this`) de manière à ce que celui-ci soit modifiable. C'est typiquement le cas des opérateurs dont le résultat *lvalue*, c'est


```
gauche.deno()*droite.deno());
```


LC co-mpo-rtmmt s-mmuabl s-o-us- po-uvz abs-lumnt pas-l changr

3.4

Rappelons rapidement ce qu'est un *downcast*

4.2.3.3 L'opérateur `dynamic_cast`

Les *downcast* posent un problème particulier car une vérification n'est possible qu'à l'exécution. Aussi, contrairement à `static_cast` et `const_cast` qui ne sont que des informations à l'adresse du compilateur, `dynamic_cast`

Sortons un objet du conteneur. Décidons qu'il s'agit d'un `Cercle` et essayons de lui appliquer la méthode `setRayon`

Dans le même ordre idée, la plupart des bibliothèques orientées objet (dont OWL et VCL d'Inprise) encapsulant l'API Windows proposent une conversion implicite de leur classe `Fenêtre` vers le type `HANDLE` de Windows.

Ce genre de bug non détecté par le compilateur est particulièrement pernicieux à détecter et doit être évité autant que possible

4.3.1.3 Parade

Le membre de gauche de la $(uc)1l4b1(a(r)11ais)-101l4be$