

# Polymorphisme et pointeurs

## Polymorphisme d'héritage

Principe de substitution de Liskov : possible de voir selon le contexte une classe ou son parent. Cette notion de voir une même chose différemment selon le contexte est appelée "polymorphisme". Ici, plus précisément, polymorphisme d'héritage.

Pour rappel, déjà vu d'autres formes de polymorphisme :

- plusieurs fonctions qui ont le même nom, mais des paramètres différents : polymorphisme ;
- plusieurs fonction de même nom et même liste de paramètres, dans une hiérarchie de classe (cf prochain chapitre) : polymorphisme;

Pour activer le polymorphisme, besoin que la classe soit passée par référence ou pointeur.

```
Base c1;  
Derived c2;  
  
Base& poly1 = c1;  
Base& poly2 = c2;
```

Dans les 2 cas, quelque soit la classe réelle, contient dans une variable de type Base.

## Pointeur unique

## Pointeur partagé

|                           |                                           |                         |
|---------------------------|-------------------------------------------|-------------------------|
| <b>Chapitre précédent</b> | <b><a href="#">Sommaire principal</a></b> | <b>Chapitre suivant</b> |
|---------------------------|-------------------------------------------|-------------------------|

[Cours, C++](#)