Aller voir www.isima.fr/~loic/cpp

STL

vector: puchBack, back ,...
list:
map: multimap, set, fonctionne par paire

Parcour par les iterator, cons_iterato, ostream iterator, begin , rbegin, end, rend, reverseiterator

copy:
transform: pour les map
generate

Autre

- revoir const
- operateur

```
ostream & operator << (ostream &str, const A& a){ str<<a; return str;}
```

- forme de coplien

```
class T
 {
        public:
                T (); // Constructeur par défaut
                T (const T&); // Constructeur de recopie
                ~T (); // Destructeur éventuellement virtuel
                T &operator=(const T&); // Operator d'affectation
 };
```

- héritage

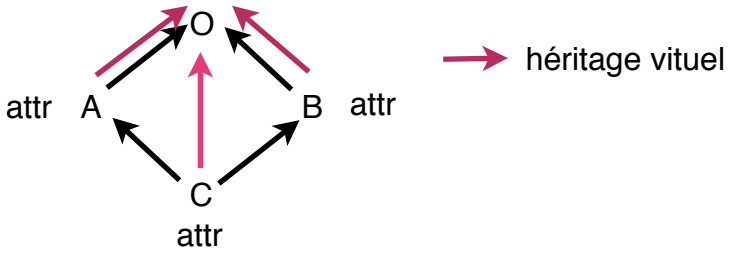
La classe F est un class M mais la classe M n'est pas une classe F

```
Class M{ m1(); virtual m2();}
Class F{ m1(); virtual m2();}

M *m = new F();
m->m1(); // appel de m1 de M
m->m2(); // appel de m2 de F
m->m3(); // probléme, pas de fct m3 dans M
```

Aller voir www.isima.fr/~loic/cpp

- héritage en diamant



pas de problème A et B n'ont pas les mêmes attributs ou méthodes, sinon, il faut lever l'ambiguité. Pour ce faire, Héritage virtuel.

C: public virtual O, public A, public B {};
A: public virtual O {};
B: public virtual O {};

Ou lever l'ambiguité dans la classe C par exemple pour l'attribut attr
dans la class C on va écrire A::attr ou B::attr pour choisir

- virtual (http://cpp.developpez.com/faq/cpp/?page=virtual)

/ ! \ List d'initialisation / ! \

- Foncteur:
classe avec redef de () qui se comporte comme une fonction

```
class F{
        X operator () (){}
}
```