Héritage en C++

Le concept d'héritage de classes est un concept général aux langages objets, mais ces langages font des choix différents dans leur manière de le mettre en œuvre.

Les attributs en C++

- peuvent avoir 3 statuts
  - `public` => visibles partout
  - `protected` => visibles dans les classes dérivées
  - `private` => visibles uniquement dans leur classe de base

La classe dérivée

- peut autoriser à ses utilisateurs l'accès aux outils ***accessibles*** de la classe de base

```
class Dérivée : public Base{...}
```

- peut interdire à ses utilisateurs l'accès aux outils ***accessibles*** de la classe de base

```
class Dérivée : private Base{...}
```

- peut limiter à ses propres classes dérivées l'accès aux outils ***accessibles*** de la classe de base

```
class Dérivée : protected Base{...}
```

Droits d'accès

```
class A{
private : int x;
protected : int y;
public : int z;};
```

```
class B : public A {
private : int a;
protected : int b;
public : int c;};
```

```
void main() {
A obj1; B obj2;
...
... obj1.x ...
... obj1.y ...
... obj1.z ...
... obj2.a ...
... obj2.b ...
... obj2.c ...
... obj2.x ...
... obj2.y ...
... obj2.z ...
... obj1.a ...}
```

Droits d'accès

```
class A{
private : int x;
protected : int y;
public : int z;};
```

```
class B : protected A {
private : int a;
protected : int b;
public : int c;};
```

```
void main() {
A obj1; B obj2;
...
... obj1.x ...
... obj1.y ...
... obj1.z ...
... obj2.a ...
... obj2.b ...
... obj2.c ...
... obj2.x ...
... obj2.y ...
... obj2.z ...
... obj1.a ...}
```

Droits d'accès

```
class A{
private : int x;
protected : int y;
public : int z;};
```

```
class B : private A {
private : int a;
protected : int b;
public : int c;};
```

```
void main() {
A obj1; B obj2;
A*pt=new B;
...
... obj1.x ..... obj1.y
... obj1.z ... obj2.a ...
... obj2.b ... obj2.c ...
... obj2.x ... obj2.y ...
... obj2.z ... obj1.a ...
…pt->c…pt->z…
}
```

La dérivation en C++

- permet de réutiliser directement tous les attributs et méthodes accessibles de la classe de base (public ou protected)
- permet d'ajouter de nouveaux composants : attributs ou méthodes
- permet de compléter les méthodes ***prévues*** (virtuelles)

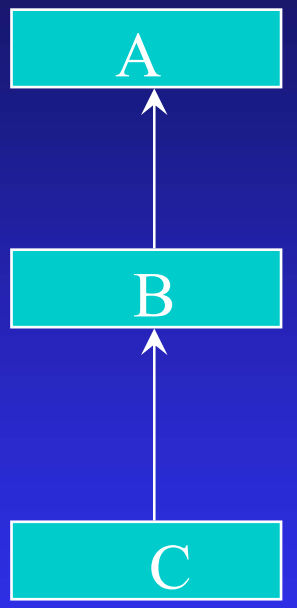
Héritage et initialisations

- L'appel du constructeur d'objet de la classe C peut se faire avec des paramètres :

```
C unC(10, 15, 20);
```

- Certains sont destinés aux constructeurs des classes de base

```
C(int arg1, int arg2, int
  arg3):B(arg1, arg2){...}
```



Les méthodes virtuelles

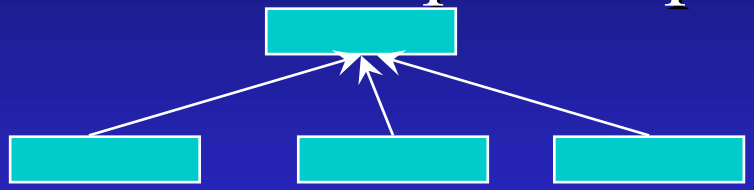
- Méthodes prévues dans la classe de base, mais implémentées dans la classe dérivée
- Exemple : la classe `ArbreBinaire` prévoit la méthode `ajout`, mais ne l'implémente pas elle-même => elle écrit :

```
virtual void ajout(const Info &)=0;
```

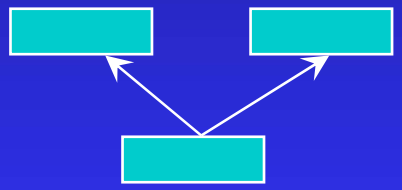
- c'est une méthode virtuelle pure.

En C++

- La dérivation multiple est possible



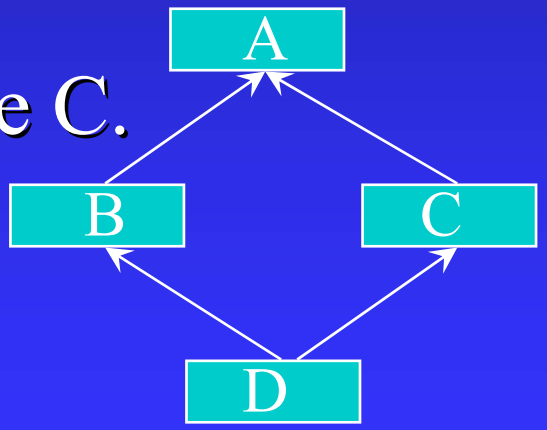
- L'héritage multiple est possible



  - une classe dérivée peut utiliser les attributs, et surtout les méthodes de plusieurs classes

L'héritage multiple peut entraîner des conflits

- La classe B dérive de A en interdisant l'accès aux éléments de A
- la classe C dérive de A en autorisant l'accès aux éléments de A
- la classe D dérive de B et de C.
- un programme utilise la classe D



Héritage virtuel

- Pour éviter que les attributs de A apparaissent 2 fois dans les objets de D

```
class B : virtual public A
class C : virtual public A
class D : public B, public C
```