27 Inheritance in C++

1. 继承

面向对象是一个非常非常大的编程范式，类的继承是其中最基本的一个概念，是我们可以利用的最强大的特性之一。继承使类之间有了相互关联的层级关系，换句话说，它使我们拥有一个包含通用功能的积累，然后从这个最开始的父类中可以创建出很多的派生类。

这就是为什么继承非常有用，以为它可以帮助我们避免写很多重复的代码
*code duplication*是指我们必须多次地写相同的代码，或者代码略微不同但是实际上是在做相同的事。

为了避免一次次地重复，我们可以把所有通用的*functions*放在一个父类中，然后很容易地从基类(*base class*)中创建派生类,稍微改变一些功能或者引入全新的功能。

—— 继承给我们提供了这样一种方式：把一系列类的所有通用的代码(功能)放到基类中，这样我们就不用像写模板那样不断重复了。

案例

C++

```
class Entity
{
public:
    float X, Y;

    void Move(float xa,float ya)
    {
        X += xa;
        Y += ya;
    }
};

class Player //要做的和Entity类很像，多了个Name和功能
{
public:
    const char* Name;
    float X, Y;

    void Move(float xa, float ya)
    {
        X += xa;
        Y += ya;
    }

void PrintName()
    {
        std::cout << Name << std::endl;
    }
};
```

那么，问题来了：

2. 怎么继承?

```C++
class Entity   // sizeof(Entity) = 8 (2个float)
{
public:
    float X, Y;

    void Move(float xa,float ya)
    {
        X += xa;
        Y += ya;
    }
};

class Player : public Entity // 此时这个Player类同时是Entity和Player类
{                            // sizeof(Player) = 12 (2个float,1个char)
public:
    const char* Name;

    void PrintName()
    {
        std::cout << Name << std::endl;
    }
};

int main()
{
    Player player;
    player.PrintName();
    player.Move(5,5);
    std::cin.get();
}
```

3. 多态

多态*polymorphism*,基本上来说就是使用一个单一的符号来表示多个不同的类型