30 Visibility in C++

是OOP中的概念，它是指一个类中的成员或者方法是否可见，可见的意思就是谁能够访问他们，谁能够调用它们，还有谁能够使用它们。

可见性并不会影响你程序的实际运行状况，也不会对程序性能等方面有影响，它只单纯的是语言层面的概念，帮助你写出更好的代码，更好地组织代码

1. 三个基础可见修饰符

1. private

*struct和public*的区别：struct默认公开，class默认私有，见19课

```C++
class Entity
{
private:    // 只有*这个Entity类可以访问到这些变量
    int X, Y;
};
```

只有*：其实C++中还有叫*friend*(友元)的关键字，可以标记其它类或者函数为当前类的友元，允许你访问这个类的**私有**成员。

```C++
class Entity
{
private:
    int X, Y;
    void Print() {}
public:
    Entity()
    {
        X = 0;
        Print();
    }
};

class Player : public Entity
{
public:
    Player()
    {
        X = 2;
        Print();  // 无法访问
    }
};

int main()
{
    Entity e;
    e.Print();    // 无法访问
    e.X = 2;      // 无法访问

    std::cin.get();
}
```

2. protected

*protected*的可见性比*private*更高，但是低于*public*。
意思是这个类以及它的所有的派生类都可以访问到这些成员。

```C++
protected:  // 派生类均可访问
    int X, Y;

    void Print() {}
......

class Player : public Entity
{
public:
    Player()
    {
        X = 2;
        Print();   // 可以访问
    }
};

int main()
{
    Entity e;
    e.Print();     // 不是子类，无法访问
    e.X = 2;       // 无法访问

    std::cin.get();
}
```

3. public

```C++
class Entity
{
public:
    int X, Y;

    void Print() {}
public:
    Entity()
    {
        X = 0;
        Print();
    }
};

class Player : public Entity
{
public:
    Player()
    {
        X = 2;
        Print();   // 可以访问
    }
};

int main()
{
    Entity e;
    e.Print();     // 可以访问
    e.X = 2;       // 可以访问

    std::cin.get();
}
```

2. 为什么要设可见性

首先，对开发者来说一切都设置为*public*绝对是一个很糟糕的想法。

指定可见性可以确保别人不会调用他们不应该接触的代码，造成一些破坏

也让提供更好的可读性，以免自己忘记。