设计过程中主要考虑以下几个因素：

1、Monster、Hero以及Boss类有诸多共同属性，例如Attack、Defence以及Health，同时又许多共同方法，例如健康状态的输出等，因此他们应该继承自同一父类。

2、相比于Monster，Boss仅仅多了一个Special Attack属性，因此可以是Boss继承自Monster。

3、与Hero交互的有Monster和Boss两个类，Hero应该同时具有对Monster和Boss的兼容性。

实现过程：

1、主要采用的技术有：类的继承，方法的覆盖，以及虚函数的应用，主要设计了4个类，Common\_类作为主要的父类，Monster和Hero分别继承于Common\_，Boss继承于Monster类。为了实现较好的界面，采用的cls等系统调用。

2、Hero和Monster继承自Common\_类，Boss继承自Monster类。为了简单起见，采用了public继承方法。相比于Common\_类，Boss新增了SpecialAttack类，并且对状态输出函数进行了覆盖。同样，相比于Monster类，Boss类实现了Special Attack，并且对战斗准备函数实现了覆盖。

3、Monster和Boss类关于战斗过程的方法采用虚函数，传入Hero类的是一个指针。因此Hero可以根据传入指针所指向的具体对象调用不同的方法，能够兼容对Monster和Boss的操作。

下一步的改进：

1、可以将fight过程单独抽象出来做一个对象，从而实现fight与对象实体的分离。

The following factors are mainly considered in the design process:

1. The Monster, Hero, and Boss classes have many common attributes, such as Attack, Defence, and Health, as well as many common methods, such as health state output, so they should inherit from the same parent class.

2. Compared to Monster, Boss only has one Special Attack property, so Boss can inherit from Monster.

3. There are two classes of Monster and Boss that interact with Hero. Hero should have compatibility with Monster and Boss at the same time.

Implementation process:

1, the main techniques used are: class inheritance, method coverage, and virtual function application, mainly designed four classes, Common\_ class as the main parent class, Monster and Hero respectively inherit from Common\_, Boss inherits from Monster class. In order to achieve a better interface, system calls such as cls are used.

2. Hero and Monster inherit from the Common\_ class, and Boss inherits from the Monster class. For the sake of simplicity, the public inheritance method is used. Compared to the Common\_ class, Boss added the SpecialAttack class and overridden the state output function. Similarly, compared to the Monster class, the Boss class implements Special Attack and overrides the combat preparation function.

3, Monster and Boss class on the battle process using virtual functions, the Hero class is a pointer. Therefore, Hero can call different methods based on the specific object pointed to by the incoming pointer, and is compatible with Monster and Boss operations.

Next improvement:

1. The fight process can be abstracted separately to make an object, thus achieving the separation of the fight from the object entity.