**C++实验报告四： 对象传递使用方法**

**刘振宇 自18G2 201830312021**

思想感悟：

1进一步加深对类和对象的理解

2对象传递使用方法：

普通对象传递

引用对象传递

指针对象传递

3指针和引用的区别

4静态成员概念和使用

静态数据和函数都是属于类的不属于对象

**对象传递的使用方法和区别：**

普通对象传递：

对象可以作为参数传递给函数，方法和传递基本类型的变量相同。向函数传递对象时，是通过“传值调用”传递给函数的，即单向传递，只有实参传给形参，而不能由形参传回来给实参。因此函数中对对象的任何修改均均不影响调用该函数的对象（实参）本身。但是，如果将对象的地址传给函数，函数中对形参对象的修改将会影响该函数的实参对象本身。

引用对象传递：

在实际程序应用中，因为引用对象传递给函数参数不仅拥有只针对象的优点，还能使引用对象的传递更加简捷直接，所以大部分程序员喜欢使用对象引用取代对象指针作为函数参数，使用对象引用就变得较为普遍。

指针对象传递：

对象指针也可以用来作为函数的参数，使用对象传递可以实现“传址调用”，即实参和形参对象指针变量指向同一内存地址，函数调用过程中如果对形参对象指针所指的对象值进行修改，实参对象的值也会随之改变。

**指针和引用的区别：**

指针和引用的区别：

  1.指针是一个实体，而引用仅是个别名；

  2.引用必须被初始化，指针不必；

  3.引用只能在定义时被初始化一次，之后不可变；指针可以改变所指的对象；

  4.可以有const指针，但是没有const引用；

  5.不存在指向空值的引用，但是存在指向空值的指针，即引用不能为空，指针可以为空；

  6.“size of 引用”得到的是所指向的变量（对象）的大小，而“size of 指针”得到的是指针本身（所指向的变量或对象的地址）的大小；

  7.指针和引用的自增（++）运算意义不一样； 指针自增指向下一个地址，而引用是对变量本身的值的增加。

  8.程序为指针变量分配内存区域，而引用不需要分配内存区域；

  9.指针可以有多级，但是引用只能是一级，例如int \*\*p是合法的，而 int &&a是不合法的；

  10.指针和引用作为函数参数进行传递时也不同。用指针传递参数，可以实现对实参进行改变的目的；在将引用作为函数参数进行传递时，实质上传递的是实参本身，而不是实参的一个拷贝，因此对形参的修改其实是对实参的修改。

  总而言之，指针与引用的一个重要的不同是指针可以被重新赋值以指向另一个不同的对象。但是引用则总是指向在初始化时被指定的对象，之后便不能随意改变。

**静态成员概念和使用：**

对象与对象之间的成员变量是相互独立的。要想共用数据，则需要使用静态成员。

　　只要在类中声明静态成员变量，即使不定义对象，也可以为静态成员变量分配空间，进而可以使用静态成员变量。（因为静态成员变量在对象创建之前就已经被分配了内存空间）

　　静态成员变量虽然在类中，但它并不是随对象的建立而分配空间的，也不是随对象的撤销而释放（一般的成员在对象建立时会分配空间，在对象撤销时会释放）。静态成员变量是在程序编译时分配空间，而在程序结束时释放空间。

静态成员的定义和声明要加个关键**static**。静态成员可以通过双冒号来使用，

即**<类名>::<静态成员名>**。

初始化静态成员变量要在类的外面进行。初始化的格式如下：

**数据类型 类名::静态成员变量名 = 初值；**

不能用参数初始化表对静态成员变量进行初始化。

既可以通过类名对静态成员变量进行引用，也可以通过对象名来对静态成员变量进行引用。

**普通成员函数和静态成员函数的区别**：普通成员函数在参数传递时编译器会隐藏地传递一个this指针.通过this指针来确定调用类产生的哪个对象;但是静态成员函数没有this指针,不知道应该访问哪个对象中的数据,所以在程序中不可以用静态成员函数访问类中的普通变量.

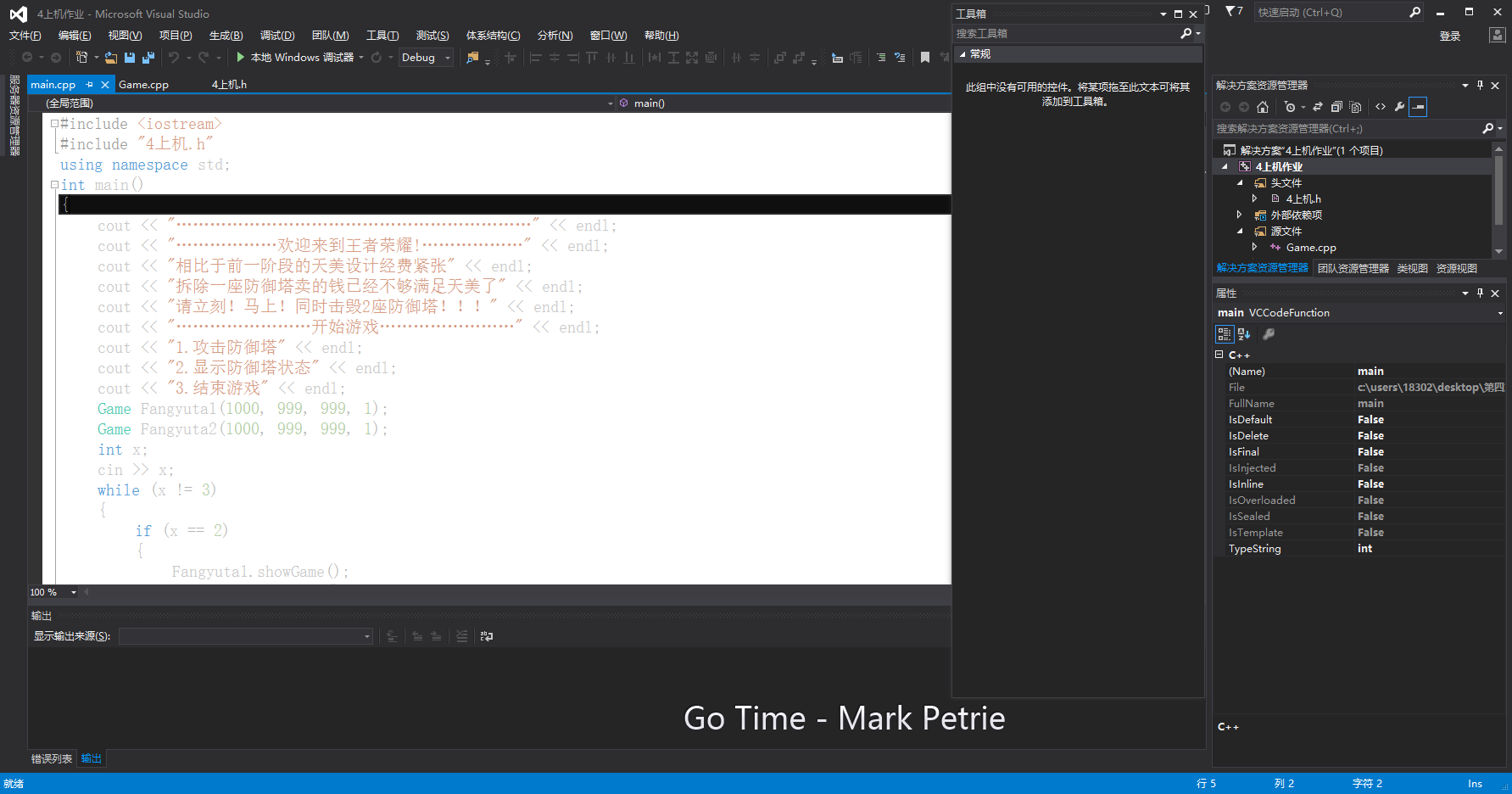
**上机作业：**

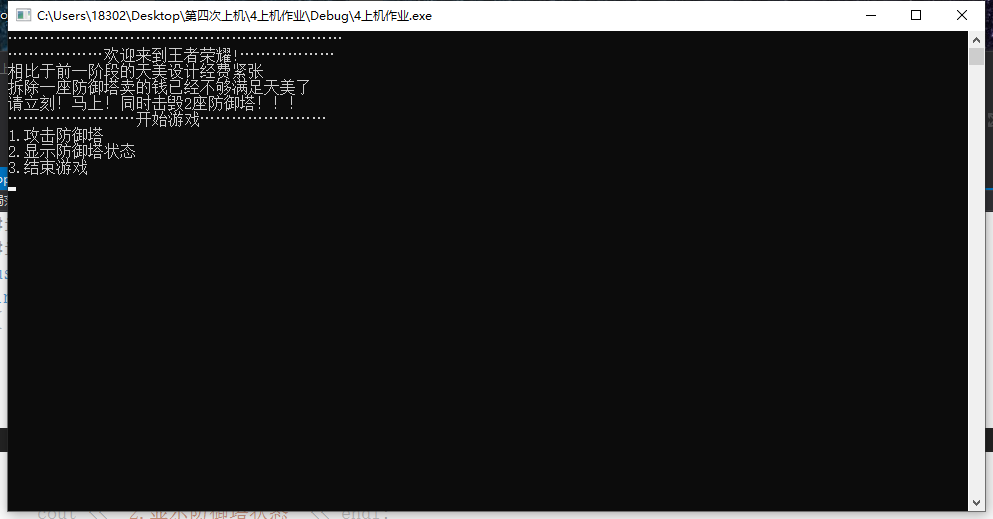
实现一个技能对两个目标的伤害分摊（见附件）

我决定用实验三的程序做一下修改：

姜子牙大招同时刮到两个防御塔（独爱老头，法力无边）

开始界面：





最后拆塔成功的截图（qwq）

