Assignment 8

191220022 丁一凡

一、 概念题

1. 简述多继承的含义；在多继承中，什么情况下会出现二义性？C++是怎么消除二义性的？

多继承是指一个类可以有两个或两个以上的直接基类

在多继承中，当多个基类中包含同名的成员时，它们在派生类中就会出现命名冲突，如下所示

```
class A
{   ......
public:
    void f();
    void g();
};
class B
{   ......
public:
    void f();
    void h();
};
class C: public A, public B
{   ......
public:
    void func()
    {
        f();
    }
}
```

在类C中访问函数 `f()` 时就会出现二义性

C++通过采用基类名受限访问来消除二义性

```
class C: public A, public B
{   ......
public:
    void func()
    {
        A::f();
        B::f();
    }
}
```

2. 继承和组合相较彼此有什么优缺点？你觉得它们各自适用于什么样的场景？

继承机制能更容易实现类之间的子类型关系，但是存在继承与封装的矛盾

组合可以避免继承与封装的矛盾，但是难以实现类之间的子类型关系

当两者为部分与整体的关系时，适用组合，比如发动机于汽车的关系

在两者在概念上为层次关系时，适用继承，如矩形和正方形

二、 编程题

1.

类的构造顺序：

先按基类的声明顺序构造基类，之后构造成员类，最后完成自己的类的构造

这一顺序可以递归完成

所以总体顺序为

`Object`、`Base`、`Derived1`、`Object`、`Base`、`Object`、`Derived2`、`Mid`;

`Object`;

`Object`、`Base`、`Derived1`;

`Object`、`Base`、`Object`、`Derived2`;

`Final`

析构顺序即为其倒序

2.

```
int double_compare(const void *p1, const void *p2){
    if(*(double *)p1 < *(int*)p2)
        return -1;
    else if(*(double *)p1 > *(bouble*)p2)
        return 1;
    else
        return 0;
}


void merge_sort(void *base, unsigned int count, unsigned int element_size,
```

```
 int (*cmp)(const void *, const void *)){
    if(count == 0)
        return ;
    if(count == 1)
        return ;
    int mid = count / 2;
    merge_sort(base, mid, element_size, cmp);
    merge_sort(base + mid * element_size, count - mid, element_size, cmp);
    void* result = (void*) malloc(count * element_size);
    int i = 0, j = mid, k = 0;
    while(i < mid && j < count)
    {
        if((*cmp)((char*)base + i * element_size, (char*)base + j *
element_size))
            (char*) result + k * element_size = (char*) base + i * element_size;
        else
            (char*)result + k * element_size = (char*) base + j * element_size;
    }
    if(i<count/2)
    {
        for(; i < count/2; i++, k++)
            (char*) result + k * element_size = (char*) base + i * element_size;
    }
    else if(j < count)
    {
        for(; j < count; j++, k++)
            (char*)result + k * element_size = (char*) base + j * element_size;
    }
    free base;
    base = result;
}
```