

第七周作业

一、简答题

1、（考察继承的基本概念，2分）

Q：派生类从基类那里继承了什么？派生类不能从基类那里继承什么？

2、（考察虚函数，1分）

Q：（1）观察下述代码

```
class Corporation
{
    ...
public:
    Corporation();
    virtual void head(){ cout << "Corporation's head\n"};
    ...
}
class PublicCorporation: public Corporation
{
    ...
public:
    PublicCorporation();
    void head(){ cout << "PublicCorporation's head\n"};
    ...
}
int main(){
    Corporation* ph = new PublicCorporation();
    ph->head();
}
```

此时ph->head()将被如何解释？输出是？

（2）观察下述代码

```
class Corporation
{
    ...
public:
    Corporation();
    void head(){ cout << "Corporation's head\n"};
    ...
}
class PublicCorporation: public Corporation
```

```
{  
    ...  
public:  
    PublicCorporation();  
    void head(){ cout << "PublicCorporation's head\n"};  
    ...  
}  
int main(){  
    Corporation* ph = new PublicCorporation();  
    ph->head();  
}
```

此时ph->head()将被如何解释？输出是？

3、（考察静态动态绑定，2分）

Q：写出下面程序的运行结果，并作出简要解释：

```
1  #include <iostream>  
2  using namespace std;  
3  class A  
4  {  
5      int m;  
6  public:  
7      A() { cout << "in A's default constructor\n"; }  
8      A(const A&) {cout << "in A's copy constructor\n"; }  
9      ~A() { cout << "in A's destructor\n"; }  
10 };  
11 class B  
12 {  
13     int x,y;  
14 public:  
15     B() { cout << "in B's default constructor\n"; }  
16     B(const A&) {cout << "in B's copy constructor\n"; }  
17     ~B() { cout << "in B's destructor\n"; }  
18 };  
19 class C: public B  
20 {  
21     int z;  
22     A a;  
23 public:  
24     C() { cout << "in C's default constructor\n"; }  
25     C(const A&) {cout << "in C's copy constructor\n"; }  
26     ~C() { cout << "in C's destructor\n"; }  
27 };  
28  
29 void func1(C x)  
30 {  
31     cout << "In func1\n";  
32 }  
33 void func2(C &x)  
34 {  
35     cout << "In func2\n";  
36 }  
37 int main()  
38 {  
39     cout << "-----Section 1-----\n";  
40     C c;  
41     cout << "-----Section 2-----\n";  
42     func1(c);  
43     cout << "-----Section 3-----\n";  
44     func2(c);  
45     cout << "-----Section 4-----\n";  
46     return 0;  
47 }
```

4、（考察静态绑定，1分）

Q：下面的程序输出结果是：

A::Fun

A::Do

A::Fun

C::Do

请完成该程序：

```
#include <iostream>
using namespace std;
class A {
    private:
        int nVal;
    public:
        void Fun()
        { cout << "A::Fun" << endl; }
        virtual void Do()
        { cout << "A::Do" << endl; }
};
class B:public A {
    public:
        virtual void Do()
        { cout << "B::Do" << endl; }
};
class C:public B {
    public:
        void Do( )
        { cout << "C::Do" << endl; }
        void Fun()
        { cout << "C::Fun" << endl; }
};
void Call(
// 在此处补充你的代码
    ) {
    p->Fun(); p->Do();
}
int main() {
    Call( new A );
    Call( new C );
    return 0;
}
```

```
}
```

二、编程题（4分）

1、（1分）Q：某滴出行有3种计费方式：

A. 按照时间计费：即自司机到达乘车点起开始计时，至目的地计时结束，每分钟需支付1.2元，不足一分钟按一分钟计算；

• 按照距离计费：

B. 第一种——自司机到达乘车点开始计算路程，至目的地时结束，每公里需支付2.7元，不足一公里按一公里计算；

C. 第二种——自司机达到乘车点开始计算路程，前2公里需支付11元，不足2公里按11元计费，之后每公里需支付2元，不足一公里按一公里计算。

已知在出行前，打车软件会提供预估时间和行驶距离。请使用使用抽象类和动态绑定使得查询最便宜的计费方式。

命令行交互示例：

请输入预计行驶时间和路程：

32 16.5

最便宜的出行方式为：

A方式，需支付38.4元

本作业需提交源码和readme文件。要求采用面向对象设计方法，不设定具体接口，可自行设计。除源码外，另需提交一份readme，说明设计思路，如包含哪些类、每个类的成员变量、成员函数以及关键函数的实现算法、数据结构等。

2、（3分）Q：The Benevolent Order of Programmers用来维护瓶装葡萄酒箱。为了描述它，BOP Portmaster设置了一个Port类，其声明如下：

```
#include <iostream> using namespace std; class Port
{
private:
    char * brand;
    char style[20]; // i.e., tawny, ruby, vintage
    int bottles;
public:
```

```

    Port(const char * br = "none", constchar * st = "none", int b = 0);
    Port(const Port & p);                                // copy constructor
    virtual ~Port() { delete [] brand; }
    Port & operator=(const Port & p);
    Port & operator+=(int b);                            // adds b to bottles
    Port & operator-=(int b); available // subtracts b from bottles, if available
    int BottleCount() const { return bottles; }
    virtual void Show() const;
    friend ostream & operator<<(ostream & os, const Port & p);
};

```

Show()方法按下面的格式显示信息：

Brand: Gallo

Kind: tawny

Bottles: 20

operator<<() 函数按下面的格式显示信息（末尾没有换行）：

Gallo, tawny, 20

Portmaster完成了Port类的方法后派生了VintagePort类，然后被解职——因为一不小心将一瓶45度的Cockburn泼到了正在准备烤肉调料的人身上，VintagePort类如下所示：

```

class VintagePort : public Port                        // style necessarily = "vintage"
{
private:
    char * nickname;                                // i.e., "The Noble" or "Old Velvet", etc.
    int year;                                        // vintage year
public:
    VintagePort();
    VintagePort(const char * br, int b, const char * nn, int y);
    VintagePort(const VintagePort & vp);
    ~VintagePort() { delete [] nickname; }
    VintagePort & operator=(const VintagePort & vp);
    void Show() const;
    friend ostream & operator<<(ostream & os, const VintagePort & vp);
};

```

你被指定负责完成VintagePort：

- 重新创建Port方法定义，因为前任被开除时销毁了方法的定义
- 解释为什么有的方法重新定义了，而有些没有重新定义
- 解释为什么没有将operator=()和operator<<()声明为虚拟的

d. 提供VintagePort中各个方法的定义

参考：《程序设计教程——用C++语言编程》（第三版）

《C++ Prime Plus》（第五版）