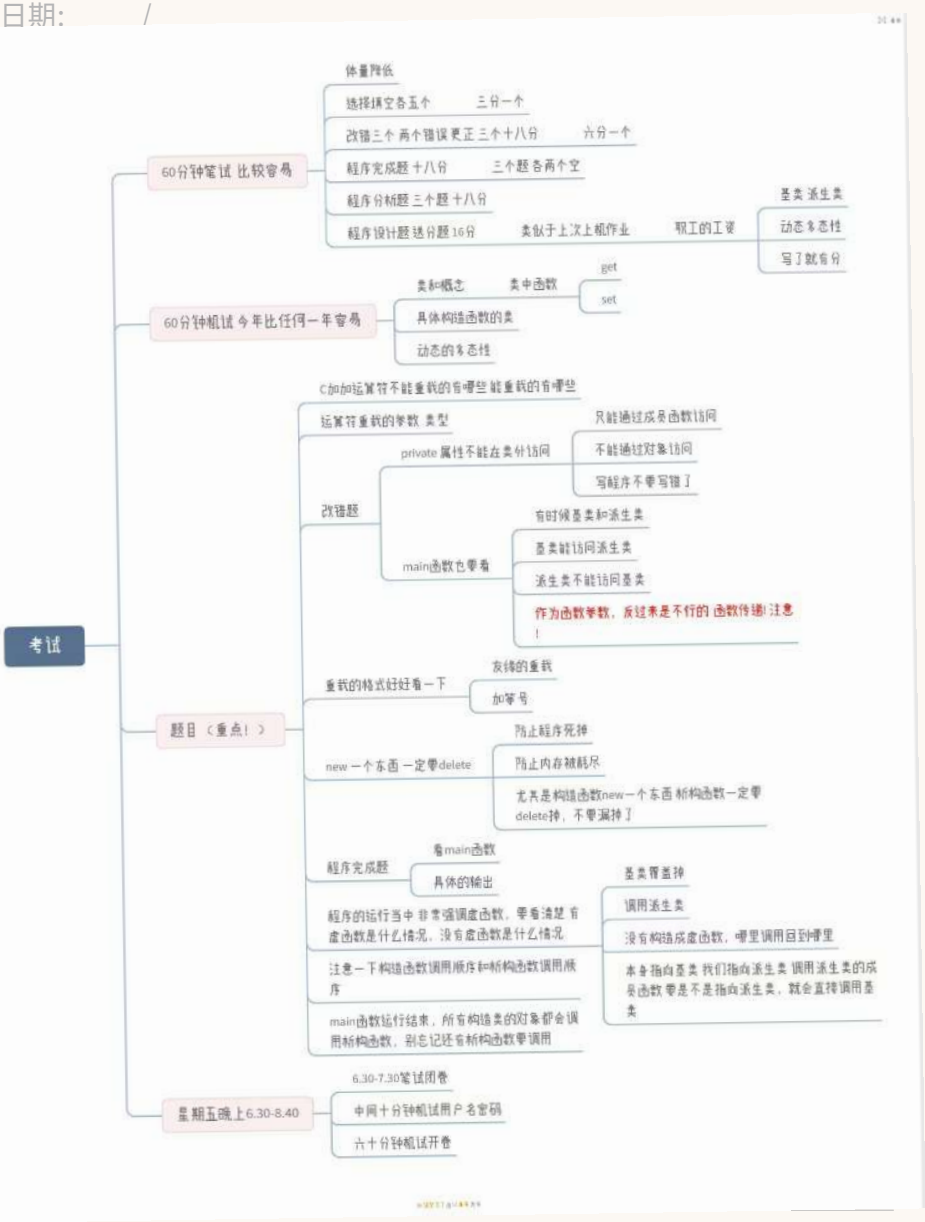
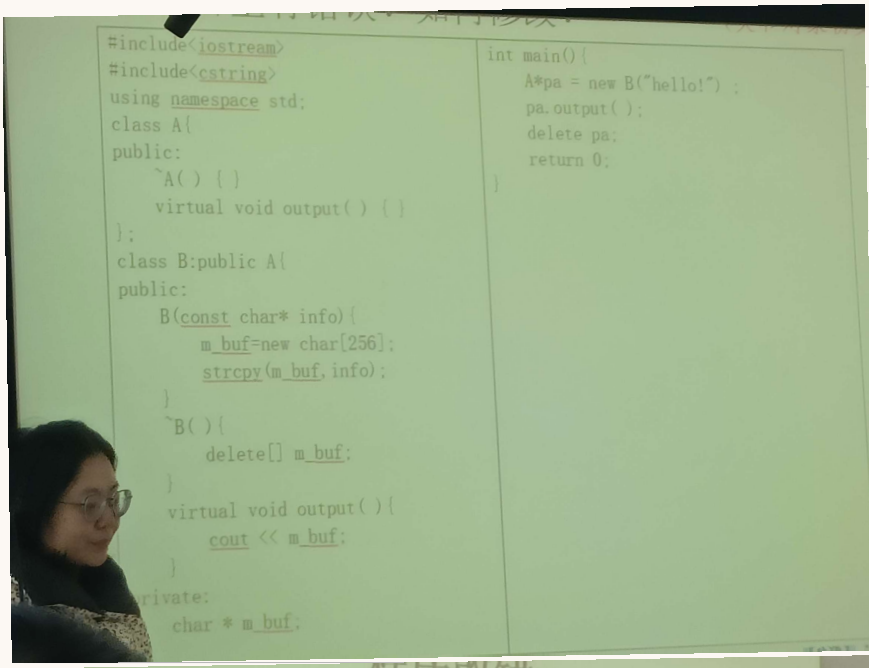
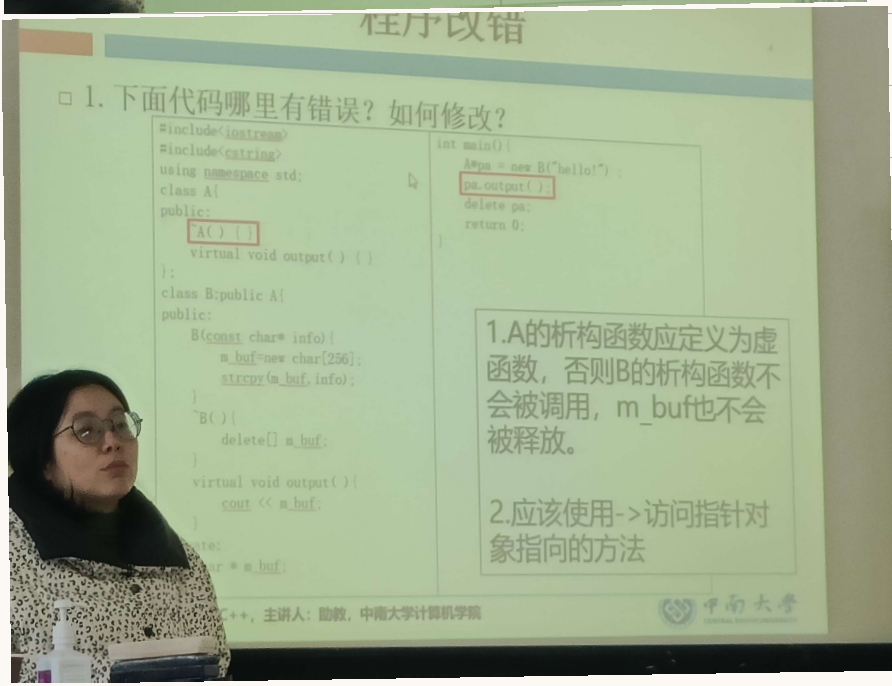
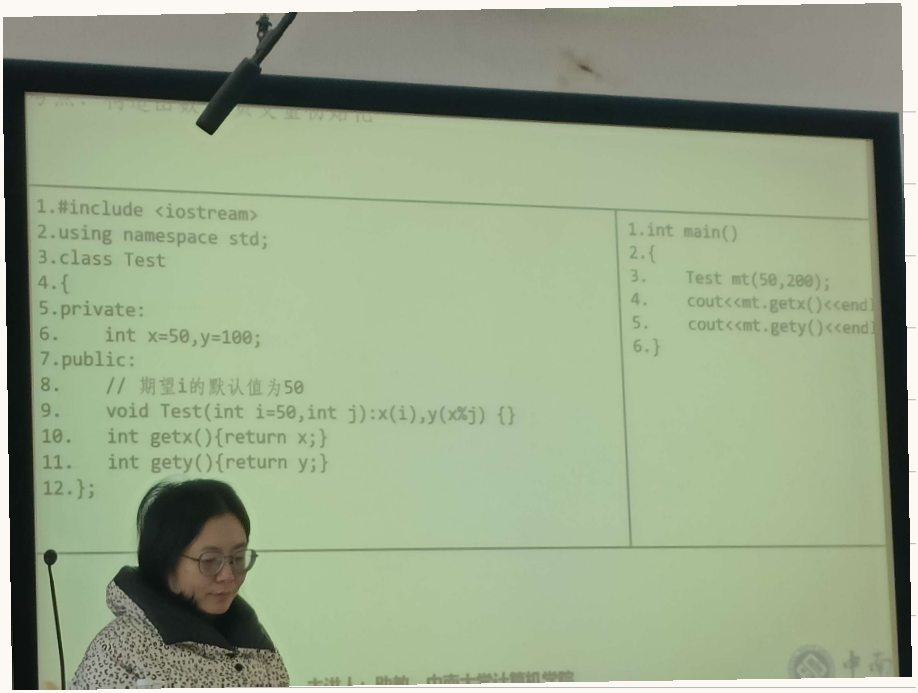
日期: /


考试
60分钟笔试 比较容易
体量降低
选择填空各五个 三分一个
改错三个 两个错误更正 三个十八分 六分一个
程序完成题 十八分 三个题各两个空
程序分析题 三个题 十八分
程序设计题 送分题16分 类似于上次上机作业 职工的工资
基类 派生类
动态多态性
写了就有分
60分钟机试 今年比任何一年容易
类和概念
类中函数
get
set
具体构造函数的类
动态的多态性
题目（重点！）
C加加运算符不能重载的有哪些 能重载的有哪些
运算符重载的参数 类型
改错题
private 属性不能在类外访问
只能通过成员函数访问
不能通过对象访问
写程序不要写错了
main函数也要看
有时候基类和派生类
基类能访问派生类
派生类不能访问基类
作为函数参数，反过来是不行的 函数传递！注意！
重载的格式好好看一下
友缔的重载
加等号
new一个东西 一定要delete
防止程序死掉
防止内存被耗尽
尤其是构造函数new一个东西 析构函数一定要delete掉，不要漏掉了
程序完成题
看main函数
具体的输出
程序的运行当中 非常强调虚函数，要看清楚 有虚函数是什么情况，没有虚函数是什么情况
基类覆盖掉
调用派生类
没有构造成虚函数，哪里调用回到哪里
注意一下构造函数调用顺序和析构函数调用顺序
本身指向基类 我们指向派生类 调用派生类的成员函数 要是不是指向派生类，就会直接调用基类
main函数运行结束，所有构造类的对象都会调用析构函数，别忘记还有析构函数要调用
星期五晚上6.30-8.40
6.30-7.30笔试闭卷
中间十分钟机试用户名密码
六十分钟机试开卷


#include<iostream>
#include<cstring>
using namespace std;
class A{
public:
~A( ) { }
virtual void output( ) { }
};
class B:public A{
public:
B(const char* info){
m_buf=new char[256];
strcpy(m_buf,info);
}
~B( ){
delete[] m_buf;
}
virtual void output( ){
cout << m_buf;
}
char * m_buf;
int main(){
A*pa = new B("hello!") ;
pa.output( );
delete pa;
return 0;
}


程序改错
1. 下面代码哪里有错误？如何修改？
1.A的析构函数应定义为虚函数，否则B的析构函数不会被调用，m_buf也不会被释放。
2.应该使用->访问指针对象指向的方法
主讲人：助教，中南大学计算机学院


1.#include <iostream>
2.using namespace std;
3.class Test
4.{
5.private:
6. int x=50,y=100;
7.public:
8. // 期望i的默认值为50
9. void Test(int i=50,int j):x(i),y(x%j) {}
10. int getx(){return x;}
11. int gety(){return y;}
12.};
1.int main()
2.{
3. Test mt(50,200);
4. cout<<mt.getx()<<end
5. cout<<mt.gety()<<end
6.}

□ 请在下面程序的横线处填上适当内容，以使程序完整,并使程序的输出为:

实部:5;虚部:3

实部:1;虚部:5

考点：重载操作符

```
#include <iostream>
using namespace std;
class Complex {
public:
  Complex(){real = image =0;}
  Complex(float a,float b) {
    real =a;
    image = b;
  }
  Complex operator+(Complex &c) {
    return Complex(①
real+c.real, image+c.image);
  }
  ②friend Complex operator-
(Complex &c1, Complex &c2) {
    return Complex(③c1.real-c2.real,c1.image-
c2.image);
  }
  float real;
  float image;
};
```

```
int main() {
  Complex c(3, 4), c2(2, -1), tmp;
  tmp = c + c2;
  cout << "实部:" << tmp.real << ";虚部
:" << tmp.image << endl;
  tmp = c - c2;
  cout << "实部:" << tmp.real << ";虚部
:" << tmp.image << endl;
}
```

实部:5;虚部:3

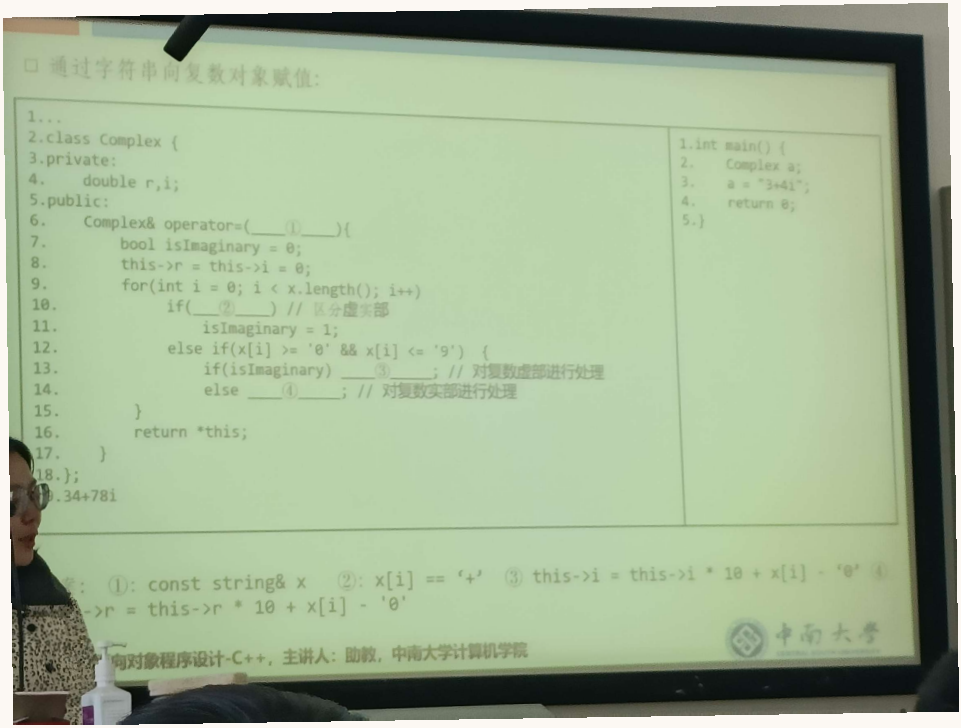
实部:1;虚部:5

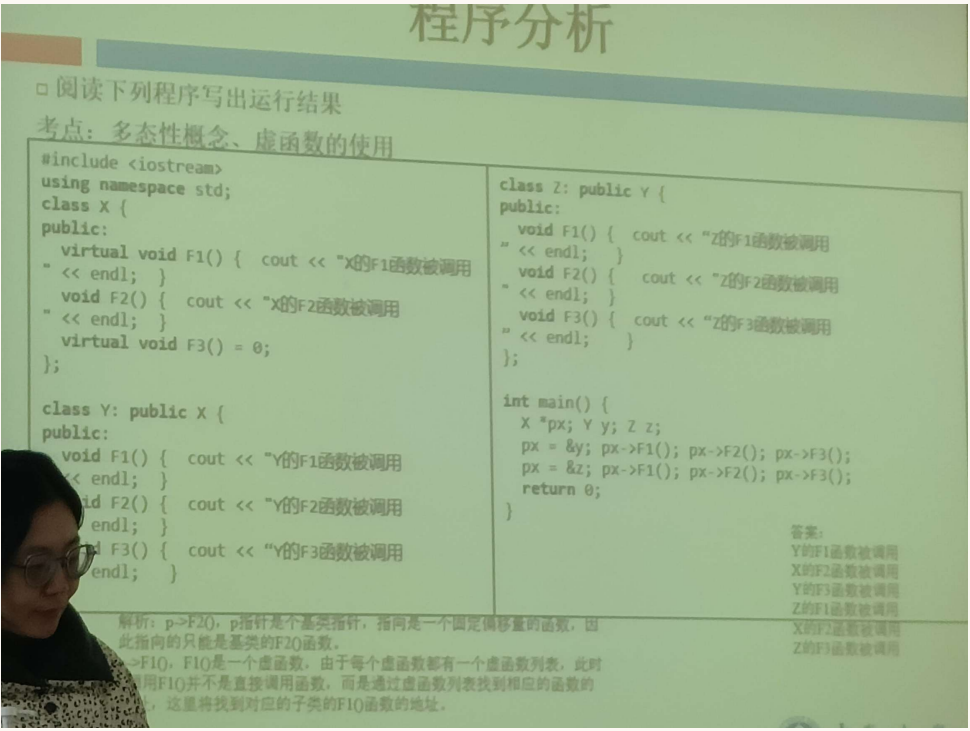
考点：重载操作符

#include <iostream>
using namespace std;
class Complex {
public:
  Complex(){real = image =0;}
  Complex(float a,float b) {
    real =a;
    image = b;
  }
  Complex operator+(Complex &c) {
    return Complex(①
real+c.real, image+c.image);
  }
  ②friend Complex operator-
(Complex &c1, Complex &c2) {
    return Complex(③c1.real-c2.real,c1.image-
c2.image);
  }
  float real;
  float image;
};

int main() {
  Complex c(3, 4), c2(2, -1), tmp;
  tmp = c + c2;
  cout << "实部:" << tmp.real << ";虚部
:" << tmp.image << endl;
  tmp = c - c2;
  cout << "实部:" << tmp.real << ";虚部
:" << tmp.image << endl;
}

助教，中南大学计算机学院


□ 通过字符串向复数对象赋值:
1...
2.class Complex {
3.private:
4. double r,i;
5.public:
6. Complex& operator=(____①____){
7. bool isImaginary = 0;
8. this->r = this->i = 0;
9. for(int i = 0; i < x.length(); i++)
10. if(____②____) // 区分虚实部
11. isImaginary = 1;
12. else if(x[i] >= '0' && x[i] <= '9') {
13. if(isImaginary) ____③____; // 对复数虚部进行处理
14. else ____④____; // 对复数实部进行处理
15. }
16. return *this;
17. }
18.};
1.int main() {
2. Complex a;
3. a = "3+4i";
4. return 0;
5.}
.34+78i
①: const string& x ②: x[i] == '+' ③ this->i = this->i * 10 + x[i] - '0' ④
->r = this->r * 10 + x[i] - '0'
向对象程序设计-C++，主讲人：助教，中南大学计算机学院
中南大学


程序分析
□ 阅读下列程序写出运行结果
考点：多态性概念、虚函数的使用
#include <iostream>
using namespace std;
class X {
public:
virtual void F1() { cout << "X的F1函数被调用
" << endl; }
void F2() { cout << "X的F2函数被调用
" << endl; }
virtual void F3() = 0;
};
class Y: public X {
public:
void F1() { cout << "Y的F1函数被调用
<< endl; }
id F2() { cout << "Y的F2函数被调用
endl; }
d F3() { cout << "Y的F3函数被调用
endl; }
class Z: public Y {
public:
void F1() { cout << "Z的F1函数被调用
" << endl; }
void F2() { cout << "Z的F2函数被调用
" << endl; }
void F3() { cout << "Z的F3函数被调用
" << endl; }
};
int main() {
X *px; Y y; Z z;
px = &y; px->F1(); px->F2(); px->F3();
px = &z; px->F1(); px->F2(); px->F3();
return 0;
}
答案：
Y的F1函数被调用
X的F2函数被调用
Y的F3函数被调用
Z的F1函数被调用
X的F2函数被调用
Z的F3函数被调用
解析：p->F2()，p指针是个基类指针，指向是一个固定偏移量的函数，因
此指向的只能是基类的F2()函数。
->F1()，F1()是一个虚函数，由于每个虚函数都有一个虚函数列表，此时
同F1()并不是直接调用函数，而是通过虚函数列表找到相应的函数的
，这里将找到对应的子类的F1()函数的地址。

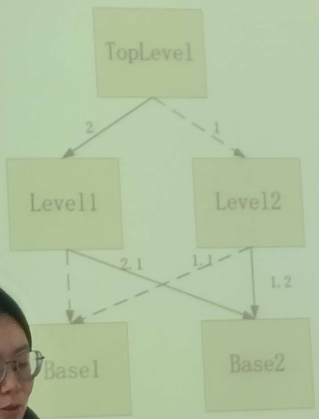
1. 阅读下列程序写出运行结果

（继承与派生）

```
#include <iostream>
using namespace std;
class Base1{
public:
    Base1(){
        cout<<"class Base1!"<<endl;
    }
};
class Base2{
public:
    Base2(){
        cout<<"class Base2!"<<endl;
    }
};
class Level1:public Base2,virtual public
Base1{
public:
    Level1(){
        cout<<"class Level1!"<<endl;
```

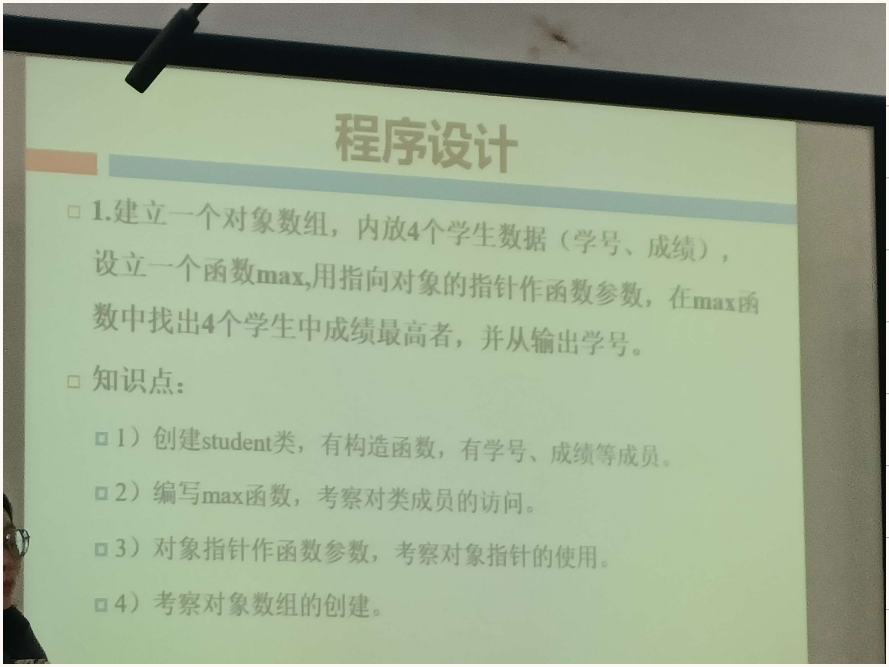
```
class Level2: public Base2,virtual public
Base1{
public:
    Level2(){
        cout<<"class Level2!"<<endl;
    }
};
class TopLevel:public Level1,virtual
public Level2{
public:
    TopLevel(){
        cout<<"class TopLevel!"<<endl;
    }
};
int main(){
    TopLevel obj;
    return 0;
}
```

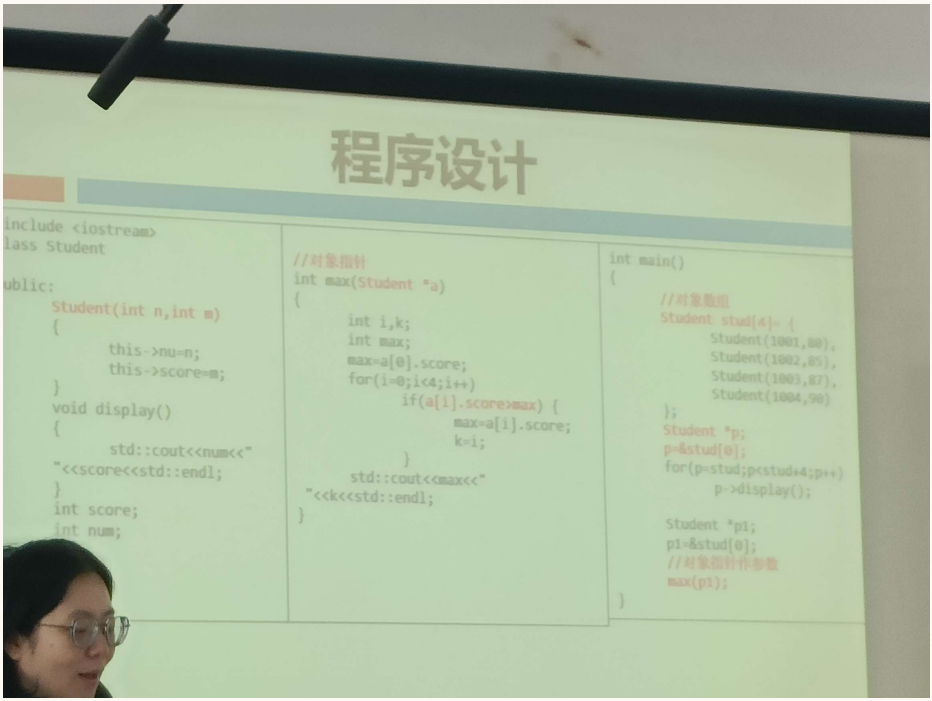
程序分析



- 虚拟继承主要是为了避免多重继承下的二义性。
- 派生类构造函数的调用次序有三个原则:
- (1)虚基类的构造函数在非虚基类之前调用;
- (2)若同一层次中包含多个虚基类,这些虚基类的构造函数按它们说明的次序调用;
- (3)若虚基类由非虚基类派生而来,则仍先调用基类构造函数,再调用派生类的构造函数.
- 继承关系如图所示，根据图可以很容易地分析出运行结果。

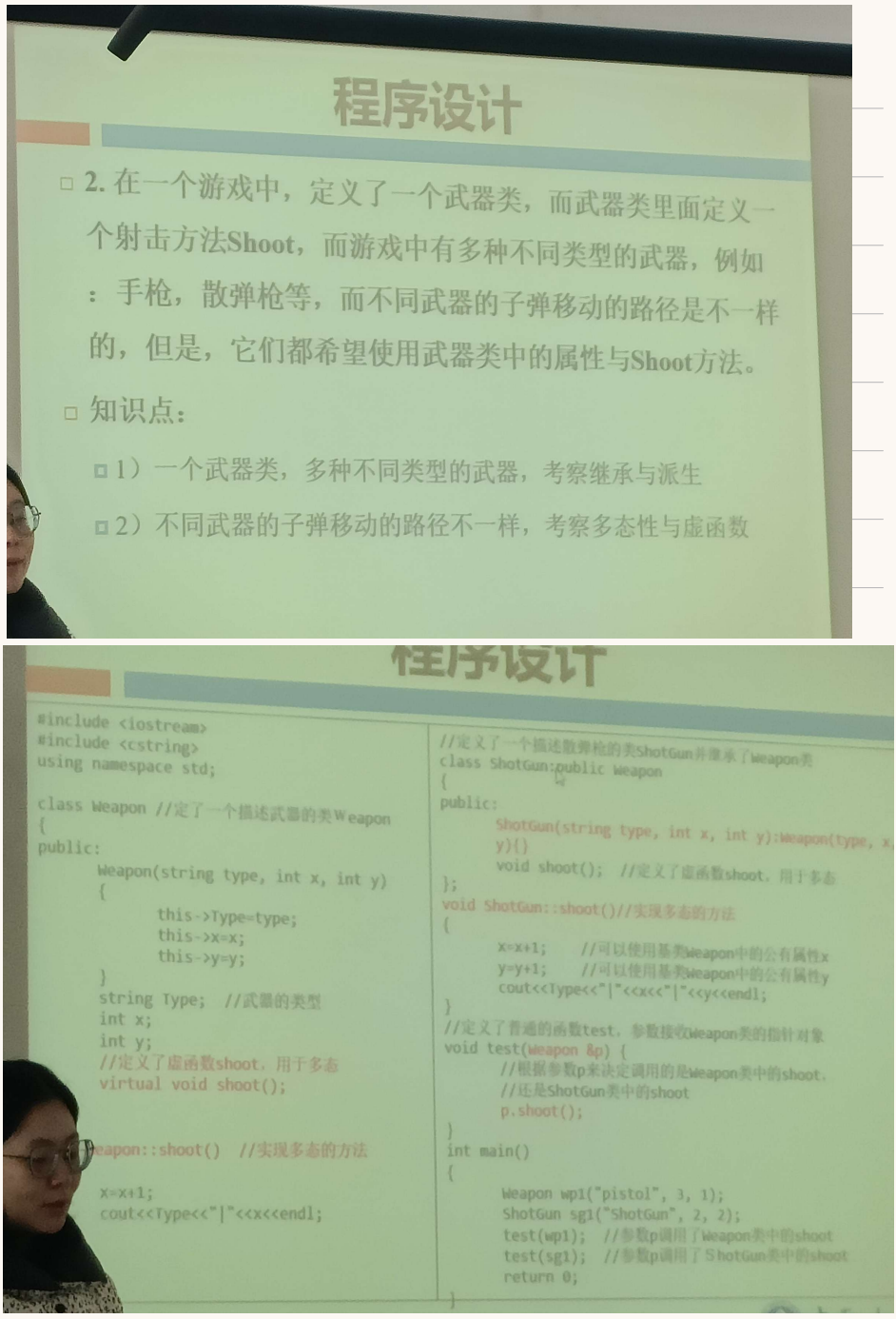
运行结果:
class Base1!
class Base2!
class Level2!
class Base2!
class Level1!
class TopLevel!

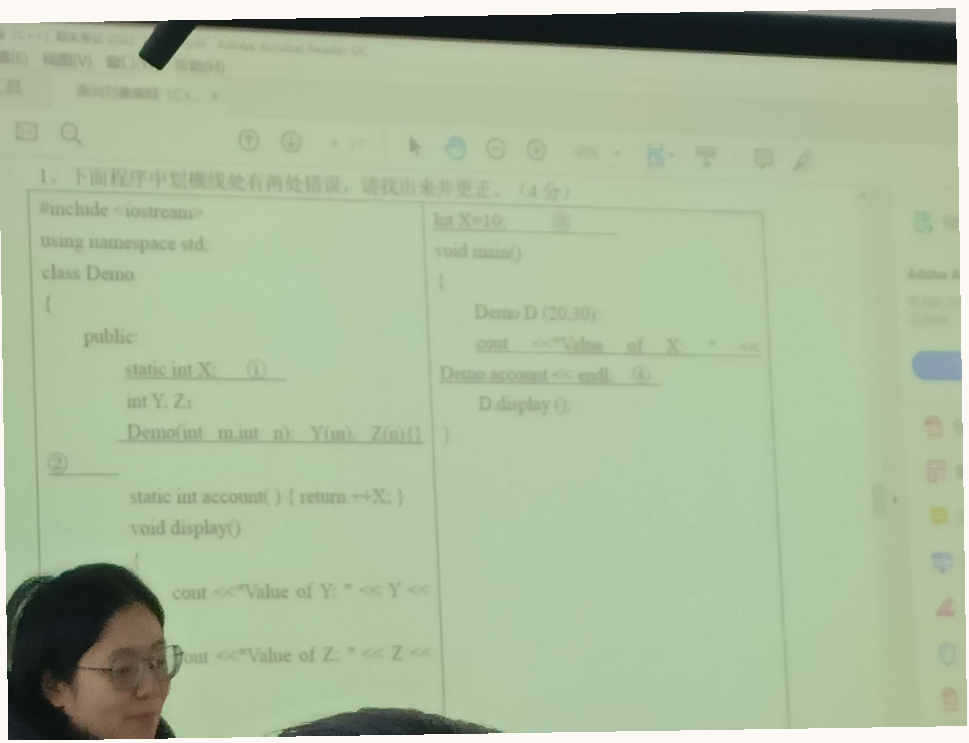

程序设计
1.建立一个对象数组，内放4个学生数据（学号、成绩），设立一个函数max,用指向对象的指针作函数参数，在max函数中找出4个学生中成绩最高者，并从输出学号。
知识点：
1）创建student类，有构造函数，有学号、成绩等成员。
2）编写max函数，考察对类成员的访问。
3）对象指针作函数参数，考察对象指针的使用。
4）考察对象数组的创建。

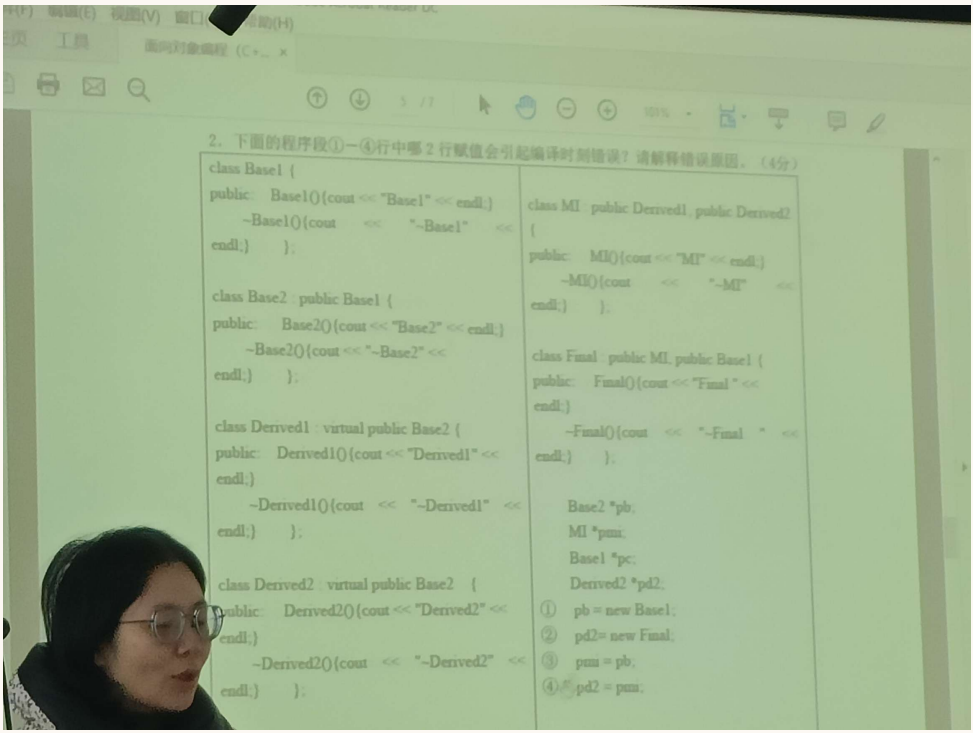

程序设计
include <iostream>
lass Student
ublic:
Student(int n,int m)
{
this->nu=n;
this->score=m;
}
void display()
{
std::cout<<num<<"
"<<score<<std::endl;
}
int score;
int num;
//对象指针
int max(Student *a)
{
int i,k;
int max;
max=a[0].score;
for(i=0;i<4;i++)
if(a[i].score>max) {
max=a[i].score;
k=i;
}
std::cout<<max<<"
"<<k<<std::endl;
}
int main()
{
//对象数组
Student stud[4]= {
Student(1001,80),
Student(1002,85),
Student(1003,87),
Student(1004,90)
};
Student *p;
p=&stud[0];
for(p=stud;p<stud+4;p++)
p->display();
Student *p1;
p1=&stud[0];
//对象指针作参数
max(p1);
}

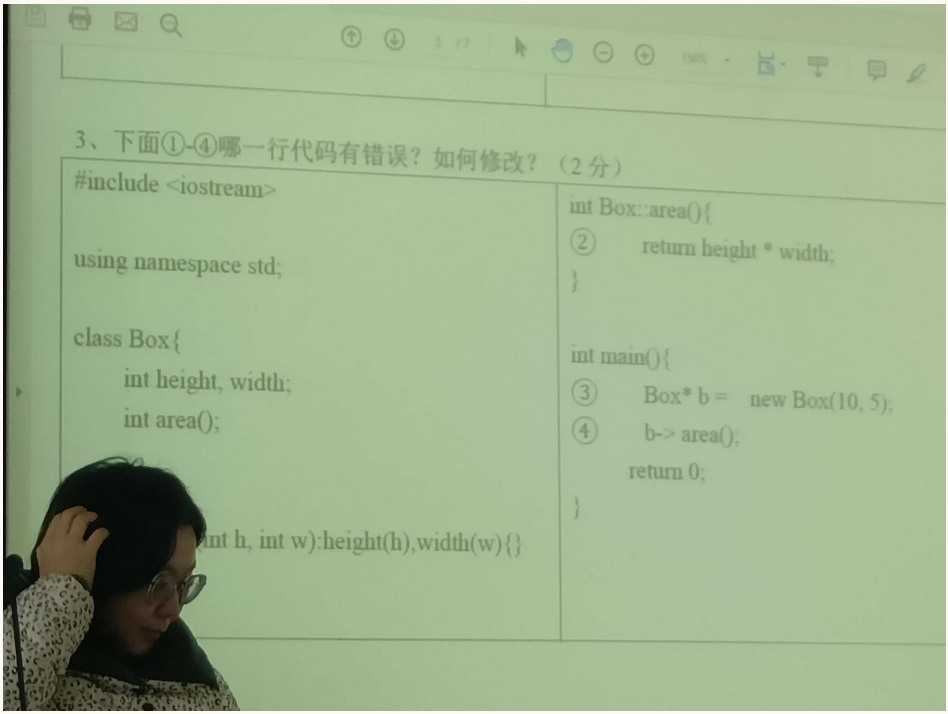
程序设计

- 2. 在一个游戏中，定义了一个武器类，而武器类里面定义一个射击方法Shoot，而游戏中有多种不同类型的武器，例如：手枪，散弹枪等，而不同武器的子弹移动的路径是不一样的，但是，它们都希望使用武器类中的属性与Shoot方法。
- 知识点：
  - 1）一个武器类，多种不同类型的武器，考察继承与派生
  - 2）不同武器的子弹移动的路径不一样，考察多态性与虚函数




1、下面程序中划横线处有两处错误，请找出来并更正。（4 分）
#include <iostream>
using namespace std;
class Demo
{
public:
static int X; ①
int Y, Z;
Demo(int m,int n): Y(m), Z(n){}
②
static int account( ) { return ++X; }
void display()
{
cout <<"Value of Y: " << Y <<
cout <<"Value of Z: " << Z <<
Int X=10; ③
void main()
{
Demo D (20,30);
cout <<"Value of X: " <<
Demo account << endl; ④
D.display ();
}


2. 下面的程序段①－④行中哪 2 行赋值会引起编译时刻错误？请解释错误原因。（4分）
class Base1 {
public: Base1(){cout << "Base1" << endl;}
~Base1(){cout << "~Base1" << endl;} };
class Base2 : public Base1 {
public: Base2(){cout << "Base2" << endl;}
~Base2(){cout << "~Base2" << endl;} };
class Derived1 : virtual public Base2 {
public: Derived1(){cout << "Derived1" << endl;}
~Derived1(){cout << "~Derived1" << endl;} };
class Derived2 : virtual public Base2 {
public: Derived2(){cout << "Derived2" << endl;}
~Derived2(){cout << "~Derived2" << endl;} };
class MI : public Derived1, public Derived2
{
public: MI(){cout << "MI" << endl;}
~MI(){cout << "~MI" << endl;} };
class Final : public MI, public Base1 {
public: Final(){cout << "Final " << endl;}
~Final(){cout << "~Final " << endl;} };
Base2 *pb;
MI *pmi;
Base1 *pc;
Derived2 *pd2;
① pb = new Base1;
② pd2= new Final;
③ pmi = pb;
④ pd2 = pmi;


3、下面①-④哪一行代码有错误？如何修改？（2分）
#include <iostream>
using namespace std;
class Box{
int height, width;
int area();
int h, int w):height(h),width(w){}
int Box::area(){
② return height * width;
}
int main(){
③ Box* b = new Box(10, 5);
④ b-> area();
return 0;
}

段落 样式

释放整个数组的空间，最适当的方法是_____。（delete [] pointer）

三、改错题

1、下面程序中划横线处有两处错误，请找出来并更正。

答案：

①错了，改为 Int Demo::X=10;

④错了，改为 cout<<"Value of X:"<<Demo::account<<endl;

2、答案：（1）行和（3）行有错误，只能用基类指针指向派生类对象，反之不可以。

3、下面①-④哪一行代码有错误？如何修改？

答案：④处有错，如果未定义类的访问属性，默认为private，不能直接通过对象访问private成员函数。修改方法，将area函数的访问属性修改为public。

四、程序分析题（本题24分，每题6分）

1、阅读下列程序写出运行结果。

五、程序完成题（12分，每空2分）

1、请在空白处补充完整代码

```
#include<iostream>
using namespace std;
class Coord
{
public:
        Coord(int i=0,int j=0){ x=i; y=j;}
        void display()
{                    ①                    }
        Coord operator ++ (int);
        int x,y;
};
Coord Coord::operator ++(int)
{
   x++;
   y++;
   ②                    ;
}
int main()
{
        Coord obj(1,1);
        obj.display();
        obj++;
        obj.display();
}
```

输出：

x=1,y=1

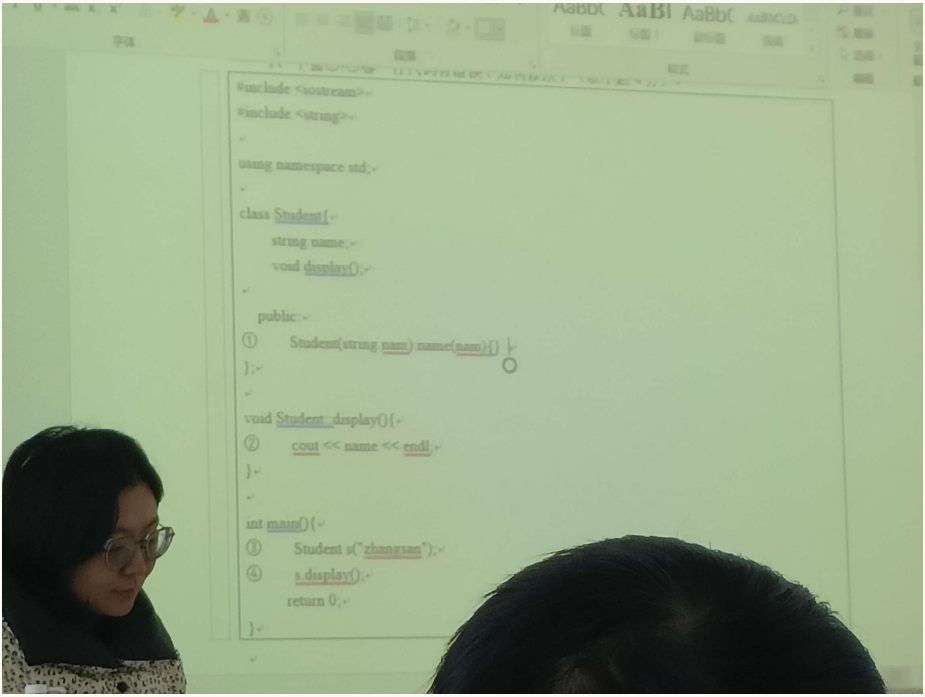
x=2,y=2

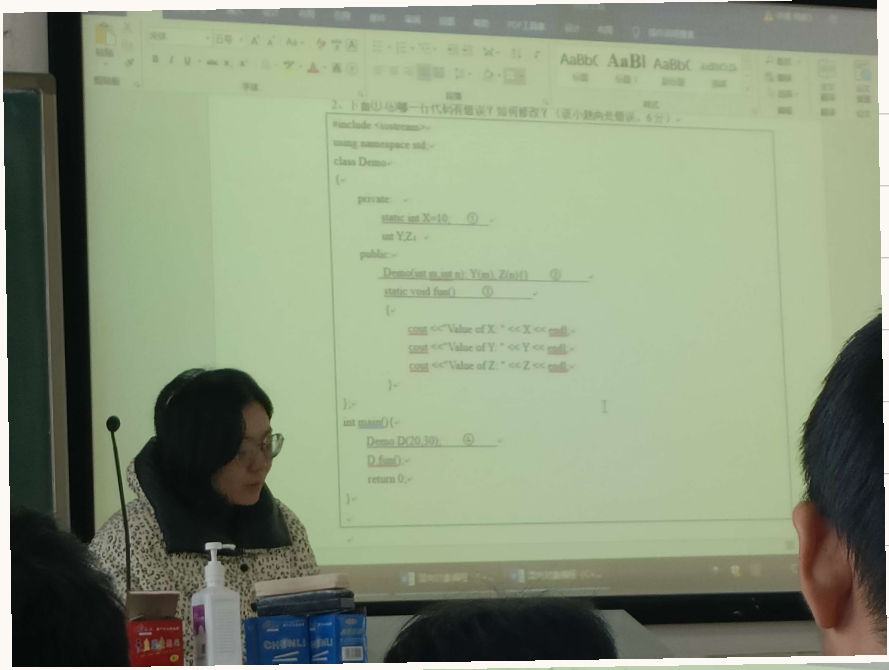
2、定义后置自增运算符：

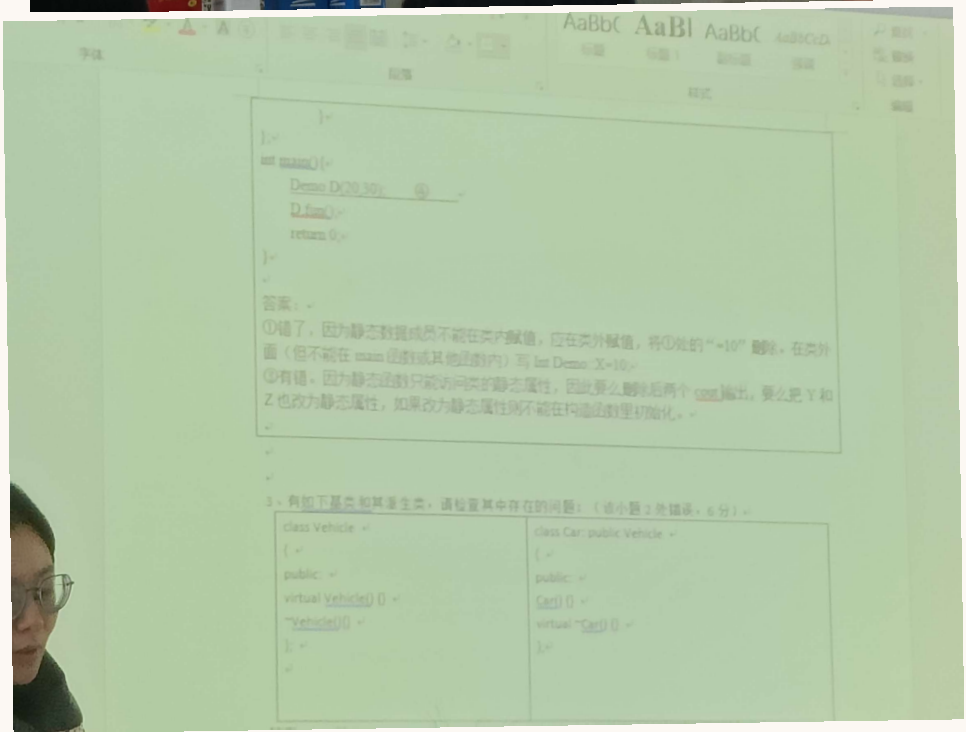
```
class Time{ public:
    Time( ){minute=0;sec=0;}
    Time operator++ (    （1）    )
  { Time temp(*this);
    sec++;
    if(sec>=60)
      {sec-=60;   ++minute; }
    return    （2）    ;
```

3、完成以下类定义，重载2个构造函数：

```
class Coord
{   int m，n;
  public:   Coord (int，int);
            Coord (Coord &);
};
Coord:: Coord (int a，int b)
{     m=a;
      （3）    ;  }
Coord:: Coord (    （4）    )
{     m=t.m;      n=t.n;   }
```


#include <iostream>
#include <string>
using namespace std;
class Student{
string name;
void display();
public:
① Student(string nam) name(nam){}
};
void Student::display(){
② cout << name << endl;
}
int main(){
③ Student s("zhangsan");
④ s.display();
return 0;
}


2、下面①-⑥哪一行代码有错误？如何修改？（请小题向关错误，6分）
#include <iostream>
using namespace std;
class Demo
{
private:
static int X=10; ①
int Y,Z;
public:
Demo(int m,int n): Y(m), Z(n){} ②
static void fun() ③
{
cout <<"Value of X: " << X << endl;
cout <<"Value of Y: " << Y << endl;
cout <<"Value of Z: " << Z << endl;
};
int main(){
Demo D(20,30); ④
D.fun(); ⑤
return 0;
}


};
int main(){
Demo D(20,30); ④
D.fun(); ⑤
return 0;
}
答案：
①错了，因为静态数据成员不能在类声赋值，应在类外赋值，将①处的“=10”删除，在类外面（但不能在 main 函数或其他函数内）写 int Demo::X=10;
③有错。因为静态函数只能访问类的静态属性，因此要么删除后两个 cout 输出，要么把 Y 和 Z 也改为静态属性，如果改为静态属性则不能在构造函数里初始化。
3、有如下基类和其派生类，请检查其中存在的问题：（请小题 2 处错误，6 分）
class Vehicle
{
public:
virtual Vehicle() {}
~Vehicle() {}
};
class Car: public Vehicle
{
public:
Car() {}
virtual ~Car() {}
};

3、有如下基类和其派生类，请检查其中存在的问题：（该小题 2 处错误，6 分）

```
class Vehicle
{
public:
virtual Vehicle() {}
~Vehicle(){}
};
```

```
class Car: public Vehicle
{
public:
Car() {}
virtual ~Car() {}
};
```

3、有如下基类和其派生类，请检查其中存在的问题：（该小题 2 处错误，6 分）


```
class Vehicle
{
public:
virtual Vehicle() {}
~Vehicle(){}
};
```

```
class Car: public Vehicle
{
public:
Car() {}
virtual ~Car() {}
};
```


答案：1）基类 Vehicle 的构造函数不应用 virtual 修饰，因为没有虚构造函数；2）基类 Vehicle 的析构函数应该用 virtual 修饰为虚析构函数，以实现对派生类对象的完全释放，而派生类 Car 的析构函数不必用 virtual 修饰。

四、程序分析题（本题 24 分，每题 6 分）

1、阅读下列程序写出运行结果

```
#include <iostream>
using namespace std;
class Coord{
```

```
int main()
{
    Coord obj(1,2);
```

日期: /