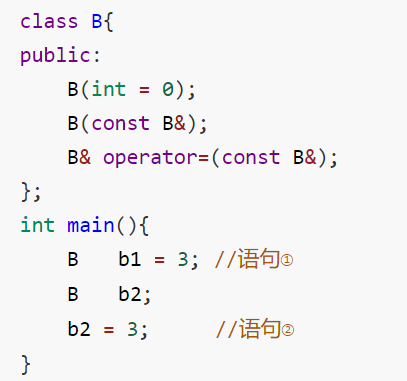
**中国地质大学（武汉）课程考核结课考试试卷** 教务处制 版本 ：2014.12



**课程名称： 面向对象程序设计（C++） 学时： 48 考 试 时 长 ： 120** 分 钟 **卷 面 总 分 ： 100** 分**考试方式：**闭卷笔试■ 开卷笔试□ 口试□ 其它  **辅助工具：**可用□ 工具名称： 不可用□

**试题内容：**

一、单项选择题（15 题，每题 2 分，共 30 分）

1．已定义字符串 char str[5],则下列表达式中不能表示 str[1]的地址的是（ B ）。A. str+1 B. str++ C. &str[0]+1 D. &str[1]

2．已知：int a[]={1,2,3,4,5,6},\*p=a,x；下面语句中 x 的值为 5 的是（ D ）。

A. p+=3; x =( p *++*); B. p *+=*5; x *=*p++ ; C. p+=4; x=*++* p; D. p *+=*4; x*=*p++;

1. 设有如下定义，下面关于 ptr 正确叙述是（ C ）。

int (\*ptr)();

* 1. ptr 是指向一维数组的指针变量。
  2. ptr 是指向 int 型数据的指针变量。
  3. ptr 是指向函数的指针，该函数返回一个 int 型数据。
  4. ptr 是一个函数名，该函数的返回值是指向 int 型数据的指针。

1. 对于定义 int \*f()中，标识符 f 代表的是（ D ）。

A．一个指向函数的指针 B．一个指针型函数，该函数返回值为指针C．一个指向整型数据的指针 D．一个指向数组的指针

1. 下列说法正确的是（ B ）。
   1. 类的构造函数不能重载;
   2. 类的析构函数不能带有参数;
   3. 基类的私有成员变量可以被派生类中的成员函数访问;
   4. 将函数的形参定义为类的引用，将触发类的拷贝构造函数的调用;
2. 有关 main 函数中的语句①和语句②的说法正确的是（ D ）。

A．语句①和语句②中“=”的语义相同B．语句①会调用 operator=

1. 语句②只会调用 operator=
2. 语句②会调用拷贝构造函数和 operator=

# 试卷类别

**A ■ B □**

**使用学期**

2019 年

装

春■ 秋□

**命题人签字**

**审题人签字**

订

**审定人签字**

线

# 考生学号

**考生姓名**

**所在班级**

1. 在下列哪种情况下**不会**调用拷贝构造函数（ A ）。
   1. 用一个对象去初始化本类的另一个对象时。
   2. 函数的形参是类的对象，在进行形参和实参的结合时。C．函数的返回值是类的对象，函数执行完返回时。

D．将类的一个对象赋值给另一个本类的对象时。8．下列关于友元的描述错误的是（ B ）。

1. 友元关系是单向的且不可传递。
2. 在友元函数中可以通过 this 指针直接引用对象的私有成员。C．在类的成员函数中可以直接访问同类其他对象的私有成员。D．通过友元可以实现在类的外部对类的私有成员的访问。
3. 下列说法正确的是（ C ）。
   1. 所有类都必须定义构造函数。
   2. 构造函数必须初始化类的所有数据成员。
   3. 子类定义了构造函数，可以不再调用父类的构造函数。
   4. 构造函数可以访问类的非静态成员。
4. 类 B 从类 A 以公有继承的方式派生，下面错误的语句为（ B ）。

A．A \*a=new B; B．B b[10];A &a=\*b;

C．A a; B b; a=b; D．B b(A()); 11．下列关于继承的说法哪个是正确的（ B ）。

1. 子类只继承父类的 public 成员。
2. 子类继承父类的非私有成员。
3. 子类只继承父类的成员，而不继承父类的父类的成员。D．子类将继承父类的所有成员。
4. 已知派生类 DerivedClass 共有继承基类 BaseClass，且有如下定义： BaseClass b,\*pb;

DerivedClass d,\*pd;

下面不符合类型兼容规则的是（ D ）。

* 1. b=d;
  2. BaseClass &bb=d;
  3. pd=&b;
  4. pb=&d;

1. 在派生类中重新定义虚函数时，除了（ D ），其它方面都必须与基类中相应的虚函数保持一致。

A．参数个数B．参数类型C．函数名称D．函数体

## 

1. 下面关于 C++流的叙述中，正确的是（ A ）。

# 

装

# 

订

# 

线

# 考生学号

**考生姓名**

**所在班级**

1. cin 是一个输入流对象。
2. 可以用 ofstream 定义一个输入流对象。
3. get()对文件进行写操作。
4. 执行语句序列 char x[80]; cin.getline(x,80);时，若键入 Happy new year，则 x 中的字符串是"Happy"。
5. 下列说法错误的是（ D ）。
   1. 如果 try 块中没有抛出异常，则 try 块执行完后忽略该 try 块的异常处理器 catch 块，程序在最后一个 catch 块后恢复执行。
   2. 如果在 try 块以外抛出异常，程序将被终止。
   3. try 块抛出异常后，从对应的 try 块开始到异常被抛出之间所构造的所有自动对象将被析构。
   4. 抛出异常和异常处理必须放在同一个函数中。

## 二、填空与简答题（填空每空 1 分，简答每题 3 分，共 30 分）

1. 用指向对象的指针引用对象成员使用操作符 【1】->\* 。
2. 当一个对象生成以后，系统就为这个对象定义了一个 【2】this指针 ，它指向这个对象的地址。
3. 非成员函数应声明为类的 【3】友元 函数才能访问这个类的 private 成员。
4. 对于常量数据成员和引用数据成员的初始化只能通过 【4】括号赋值 来 完 成 。
5. 调用语句 int\* pa = new int[10];动态分配内存pa，释放 pa 的语句为 【5】delete pa 。

21．引用 【6】不是 （填是或者不是）一种数据类型，所以没有引用的指针。

1. 【7】纯虚函数 是一种特别的虚函数, 拥有 【8】纯虚函数 的类就是 【9】虚基类 , 这种类不能实例化。
2. 含有继承关系和子对象的类构造对象时，析构函数的调用顺序是 【10】先子类后父类 。
3. C++支持四种类型的多态，分别是强制多态、 【11】参数多态 、 【12】过载多态 、 【13】包含多态 。
4. 类的构造函数 【14】不可以 (可以/不可以)是虚函数，类的析构函数 【15】可以 (可以/不可以)是虚函数。
5. 多继承时，多个基类中的同名的成员在派生类中由于标识符不唯一而出现二义性。在派生类中可以采用 【16】虚基类 或 【17】作用域分辨符 来消除该问题。
6. 为类而非对象定义数据成员的关键字是 【18】static ，其访问使用 【19】this 操作符。
7. throw;语句的作用是 【20】用来抛出异常 ，catch(…)语句的作用是 【21】匹配异常类型，对指定的异常进行处理 。
8. 列举拷贝构造函数调用的时机。

函数的形参是类的对象，在进行形参和实参的结合时，函数的返回值是类的对象，函数执行完返回时。

将类的一个对象赋值给另一个本类的对象时。

1. 编译时的多态性与运行时的多态性有何不同？分别怎样实现？

编译时的[多态性](http://www.so.com/s?q=%E5%A4%9A%E6%80%81%E6%80%A7&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank):就是在程序编译的时候，也就是生成解决方案的时候就决定要实现什么操作。而[运行时](http://www.so.com/s?q=%E8%BF%90%E8%A1%8C%E6%97%B6&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)的多态性:就是指直到系统运行时,才根据实际情况决定实现何种操作。

1. 简述面向对象程序设计的优点和不足。

1、易维护  
采用[面向对象思想](http://www.so.com/s?q=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E6%80%9D%E6%83%B3&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)设计的结构，[可读性](http://www.so.com/s?q=%E5%8F%AF%E8%AF%BB%E6%80%A7&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)高，由于继承的存在，即使改变[需求](http://www.so.com/s?q=%E9%9C%80%E6%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)，那么维护也只是在局部[模块](http://www.so.com/s?q=%E6%A8%A1%E5%9D%97&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)，所以维护起来是非常方便和较低成本的。  
2、质量高  
在设计时，可重用现有的，在以前的[项目](http://www.so.com/s?q=%E9%A1%B9%E7%9B%AE&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)的[领域](http://www.so.com/s?q=%E9%A2%86%E5%9F%9F&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)中已被测试过的类使系统满足业务需求并具有较高的质量。  
3、效率高  
在[软件开发](http://www.so.com/s?q=%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)时，根据设计的需要对[现实世界](http://www.so.com/s?q=%E7%8E%B0%E5%AE%9E%E4%B8%96%E7%95%8C&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)的[事物](http://www.so.com/s?q=%E4%BA%8B%E7%89%A9&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)进行抽象，产生类。使用这样的方法[解决问题](http://www.so.com/s?q=%E8%A7%A3%E5%86%B3%E9%97%AE%E9%A2%98&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)，接近于日常生活和自然的思考方式，势必提高软件开发的效率和质量。  
4、易扩展  
由于继承、封装、[多态](http://www.so.com/s?q=%E5%A4%9A%E6%80%81&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)的[特性](http://www.so.com/s?q=%E7%89%B9%E6%80%A7&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)，自然设计出[高内聚](http://www.so.com/s?q=%E9%AB%98%E5%86%85%E8%81%9A&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)、低耦合的[系统结构](http://www.so.com/s?q=%E7%B3%BB%E7%BB%9F%E7%BB%93%E6%9E%84&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)，使得系统更灵活、更容易扩展，而且成本较低。

## 三、看程序写结果（每题 4 分，共 16 分）

|  |  |
| --- | --- |
| 1. 阅读程序，请写出程序的运行结果。#include<iostream>   using namespace std; class A {  public:  int n;  };  class B: virtual public A {}; class C: virtual public A {}; class D: public A {};  class E:public B, public C,public D{ public:  int getn(){return B::n;}  };  void main() { E d;  d.B::n=10;  d.C::n=20;  d.D::n=30;  cout<<d.B::n<<endl; cout<<d.C::n<<endl; cout<<d.D::n<<endl; cout<<d.getn()<<endl;  }  20 20 30 20   1. 阅读程序，请写出程序的运行结果。#include <iostream>   using namespace std; class A{  static int gInstCount; public:  A() {cout << "Instance " << ++gInstCount << " is constructed." << endl;}  ~A() {gInstCount--;}  };  int A::gInstCount = 0; int main(){  A\* pa = new A(); delete pa;  A a[2];  return 0;  }  Instance 1 is constructed.  Instance 1 is constructed.  Instance 2 is constructed. | 34．阅读程序，请写出程序的运行结果。#include <iostream>  using namespace std; class A {  const int i; int &j;  public:  A(int& var):i(var++), j(++var){} void show() {  cout<<"i:" << i << endl  <<"j:" << j << endl;  }  };  int main(){  int x=10; A a1(x);  a1.show(); return 0;  }  i:10  J:12 |

|  |  |
| --- | --- |
| 35．阅读程序，请写出程序的运行结果。 |  |
| #include <iostream> using namespace std; class B{  public:  B();  B(int i);  ~B();  virtual void Print() ; private:  int b;  };  B::B(){ b=0;  cout<<"B's default constructor called."<<endl;} B::B(int i){  b=i;  cout<<"B's constructor called." <<endl;  }  B::~B(){  cout<<"B's destructor called."<<endl;  }  void B::Print(){ cout<<b<<endl; }  class C:public B{ public:  C();  C(int i,int j); | ~C();  void Print() ; private:  int c; B b;  };  C::C()  { c=0;  cout<<"C's default constructor called." <<endl;  }  C::C(int i,int j):B(i)  { c=j;  cout<<"C's constructor called."<<endl;  } C::~C()  { cout<<"C's destructor called."<<endl; } void C::Print()  { B::Print(); cout<<c<<endl;  }  void main()  {  C obj(5,6);  obj.Print();  }  B's constructor called.  B's default constructor called.  C's constructor called.  5  6  C's destructor called.  B's destructor called.  B's destructor called. |

装

订

四、综合题（2 题，每题 4 分，共 8 分）

36. 指出下面程序类 B 成员函数中的两处错误，并提出改正方法。

# 

**考生学号**

线

**考生姓名**

**所在班级**

1. #include <iostream>
2. using namespace std;
3. class A{
4. public:
5. A(int i,int j){a=i;b=j;}
6. void Move(int x,int y){a+=x;b+=y;}

1.指

7 void Show( ){cout<<"("<<a<<","<<b<<")"<<endl;}

出下面程序类 B 成员函数中的两处错误，并提出改

8 private: int a,b;

9 };

1. class B:private A{
2. public:
3. B(int i,int j,int k,int l):(i,j){x=k;y=l;} //构造函数
4. void Show( ){cout<<x<<","<<y<<endl;}//显示 xy 值
5. void fun( ){Move(3,5);}
6. void f1( ){Show( );} //显示 ab 值
7. private:
8. int x,y;

18 };

37.class Trapezoid : public Shape  
{  
private:  
double top;  
double bottom;  
double height;  
public:  
Trapezoid(double t, double b, double h)  
{  
top = t;  
bottom = b;  
height = h;  
}  
  
double Area()  
{  
return (top + bottom) \* height / 2;  
}  
};  
  
#define PI 3.1415926  
class Circle : public Shape  
{  
private:  
double r;  
public:  
Circle(double r)  
{  
this->r = r;  
}  
  
double Area()  
{  
return PI \* r \* r;  
}  
};  
  
class Triangle : public Shape  
{  
private:  
double bottom;  
double height;  
public:  
Triangle(double b, double h)  
{  
bottom = b;  
height = h;  
}  
  
double Area()  
{  
return bottom \* height;;  
}  
}

38.#include<iostream>

using namespace std;

class Complex {

private:

double real;

double image;

public:

Complex()

{

real = 0;

image = 0;

}

Complex(double a, double b)

{

real = a;

image = b;

}

Complex operator+(const Complex &c)

{

/\*Complex tmp;

tmp.real = real + c.real;

tmp.image = image + c.image;

return tmp;\*/

real = real + c.real;

image = image + c.image;

return \*this;

}

Complex operator-(const Complex &c)const

{

Complex tmp;

tmp.real = real - c.real;

tmp.image = image - c.image;

return tmp;

}

Complex operator\*(const Complex &c)const

{

Complex tmp;

tmp.real = real\*c.real - image\*c.image;

tmp.image = image\*c.real + real\*c.image;

return tmp;

}

void show()

{

cout << real << "+" << image << "i" << endl;

}

};

39.class Line//Line类的声明

{

public:

 Line (Point xp1, Point xp2);

 Line (Line &);

 double GetLen()

private: //私有数据成员

 Point p1,p2; //Point类的对象p1,p2

 double len;

};

//组合类的构造函数

Line:: Line (Point xp1, Point xp2)

:p1(xp1),p2(xp2)

{

 cout<<"Line构造函数被调用"<<endl;

 double x=double(p1.GetX()-p2.GetX());

 double y=double(p1.GetY()-p2.GetY());

 len=sqrt(x\*x+y\*y);

}

//组合类的拷贝构造函数

Line:: Line (Line &Seg): p1(Seg.p1), p2(Seg.p2)

{

 cout<<"Line拷贝构造函数被调用"<<endl;

 len=Seg.len;

}