

《C++面向对象程序设计》试题 (A 卷)

(满分: 100 分 时间: 150 分钟 日期: 2005.1.10)

一、(共 20 分, 每题 2 分) 单项选择

1. 已定义了一个类 A 并有语句 `A* pa=new A(5);` 那么以下说法正确的是:
 - A. 该语句会创建 A 类的一个对象, 并将此对象的地址赋给指针 pa;
 - B. 该语句会创建 A 类的 5 个对象, 且它们的地址是连续的;
 - C. 当指针变量 pa 超出其作用域时, pa 和为对象分配的空间都将被自动释放;
 - D. 类 A 中一定有一个显式定义的构造函数和一个显式定义的虚析构函数;
2. 定义类 A 的非静态成员函数 `A& f(A& one)` 时, 需有语句 `return exp;` 则 exp 不能是:
 - A. 类 A 中类型为 A 的静态数据成员
 - B. f 中用语句 `A a = one;` 定义的量 a
 - C. one
 - D. *this
3. 在一个类中, 下列哪些成员可以是多个?
 - A. 无参构造函数
 - B. 析构函数
 - C. 原型相同的非静态成员函数
 - D. 静态成员函数
4. 对类 T 中的下列成员, 若不考虑代码优化, 一般来说它们中的哪个执行起来最慢?
 - A. 缺省的无参构造函数
 - B. 缺省的拷贝构造函数
 - C. `T f() { T t; return t; }`
 - D. 不能判定 A、B、C 哪个最慢
5. 对函数原型为 `int& min(int& a, int& b);` 的全局函数, 下列语句块哪个是错误的:
 - A. `int x=10,y=5; min(x,y)=100;`
 - B. `int x=10,y=5, z; z=min(x,y+=x);`
 - C. `int x=10,y=5; min(x,10)=y;`
 - D. `int x=10,y=5; x=min(x,y);`
6. 关于拷贝构造函数错误的说法有:
 - A. 拷贝构造函数是一种特殊的构造函数, 且不能在类中进行函数重载。
 - B. 若自定义派生类的拷贝构造函数, 那么也必须自定义基类的拷贝构造函数。
 - C. 拷贝构造函数只能有一个参数, 且为常量引用或非常量引用。
 - D. 拷贝构造函数不允许在函数体中使用 return 语句。
7. 下列对虚基类声明正确的是:
 - A. `class virtual B: public A`
 - B. `class B: virtual public A`
 - C. `class B: public A virtual`
 - D. `virtual class B : public A`
8. 若有语句 `A*const p=new B;`, 其中类 B 是从类 A 直接派生得到的, 那么下列说法错误的是:
 - A. 若要执行语句 `p->f();`, 那么类 A 中一定有一个与 f() 函数匹配的函数声明

明。

B. 类 B 一定是通过 public 继承方式从类 A 派生的。

C. 类 B 的构造函数中, 至少有一个无参的或提供全部缺省参数的构造函数。

D. 通过指针 p, 只能访问类中的常量成员函数或静态成员函数。

9. 下面说明的哪个数据不能作为类 T 的数据成员定义。

A. T* pT

B. class Q{ } q1,q2;

C. static T t;

D. const T t;

10. 下列哪种运算符可以被重载:

A. sizeof

B. ::

C. ? :

D. []

二、(共 15 分, 每题 3 分) 回答下列各题, 可以举例说明。

1. 说明基类中的 public、protected 和 private 成员分别在 public、protected 和 private 继承方式下在派生类中的可访问性。

2. 若类 A 有私有数据成员 int n, 哪些情况下, 在一个函数的函数体中能够直接访问 A 类的数据成员 n。(至少三种)

如: a) void A::f() { int k=n; }

b) void A::g(A& one) { n+=one.n; }

c) A 的友员函数 int h(A& one) {return one.n; }

3. 哪些情况下, 定义构造函数时必须使用初始化列表。(至少三种)

4. 将一个基类的析构函数定义为虚函数的作用和目的是什么?

5. 为什么在类中不能用成员函数 void f(int&) 重载(overload)本类的成员函数 int f(int)?

三、(共 10 分, 每题 1 分) 判断正误, 只需回答“正确”或“错误”。

1. this 指针既可以是指向常量对象的非常量指针, 也可以是指向非常量对象的常量指针。

2. 即使不创建类的实例对象, 也可以在 main 函数中访问该类的公有静态成员函数。

3. 在一个函数体内不允许定义其它函数, 同样在一个类中也不允许定义其它类。

4. 若想在 C++ 文件中使用一个模板类, 那么在预编译后, 此文件必须含有该模板类的全部声明和定义。

5. 在捕获异常时, 异常会进行自动类型转换并按照异常类型的同一性进行匹配。

6. 一个抽象类的后裔类, 既可以是抽象类, 也可以是具体类。

7. 类 A 中的转换函数可定义为: int operator A() const {return 0; }.

8. 类的多个实例对象可共享本类的虚函数表。

9. 用 const 修饰的成员函数只能通过常量对象调用。

10. 类 B 是类 A 的派生类, 类 C 是类 A 的友员类, 那么类 C 也是类 B 的友员类。

四、(25 分, 每小题 5 分) 指明下列各程序中存在的 C++ 语言规范之处(每题一处), 说明你的理由或依据。解答方式可参考下面例子。

<p>例子程序:</p> <pre>#include <iostream.h> class A {</pre>	<p>对左边例子程序可以解答如下:</p> <p>成员函数 f 被说明成 protected,</p>
---	---

<pre>protected: void f() { cout << "A::f()" << endl;} }; void main() { A a; a.f(); }</pre>	<p>这与 main 函数中对它的使用: a.f() 语句不符。理由是: 一个类中 protected 访问权限下的成员除了可用于实现该类、实现该类的派生类或后裔类、实现友员外, 不能被使用。</p>
<pre>(1) #include <iostream.h> class Complex { public: Complex(float r=0.0,float i=0.0) : rPart(r), iPart(i) { } private: float rPart; float iPart; };</pre>	<pre>void main() { Complex c1; Complex c2(3.14); Complex c3(c2); c1 = c2; cout << c1 << c2 <<c3 << endl; }</pre>
<pre>(2) #include <iostream.h> class B; class A { public: A(int anInt=0):i(anInt){ } A(const B& aB); int getI() const { return i; } private: int i; }; class B { public: B(int int1=0, int int2=0): i(int1),j(int2) { } int getI() const { return i; }</pre>	<pre>(3) class A { public: A* f() { delete pA; pA = new A; return pA;} ~A() { delete pA; } static A* g() { return this; } private: int i; static A* pA; }; A* A::pA = 0; void main() { A a; A* p = a.f(); A* q = a.g(); }</pre>

<pre>int getJ() const { return j; } operator A(){ int k = (i<j)?i:j; return A(k); } private: int i; int j; }; A::A(const B& aB) { if (aB.getI()> aB.getJ()) i = aB.getI(); else i = aB.getJ(); } void main() { B b(10,20); A a(b); cout << a.getI() << endl; }</pre>	<pre>(4) #include <iostream.h> class B { public: B(int int1,int int2) : j(int2), i(int1) { } virtual void f() const { cout << "B::f()" << endl; } private: int i; int j; }; class D: public B { public: D(int anInt) : i(anInt) { } void f() const { cout << "D::f()" << endl; } private: int i; }; void main() { D d; d.f(); }</pre>
--	--

<pre>(5) #include <iostream.h> class A { public: A(int anInt=0): a(anInt) { } void f() { cout << "A::f()" << endl; } void g() { if (i>0) a.f(); else cout << "A::g()" << endl; } private: int i; A a; };</pre>	<pre>void main() { A a1(100); a1.g(); A a2(-10); a2.g(); }</pre>
--	---

五、（5 分）写出下面程序的运行结果

<pre>#include <iostream.h> class A { public: A() { cout<<"<1> A::A()" "<<endl;} virtual ~A() { cout<<"<2> A::~~A()"<<endl;} virtual void g() { cout<<"<3> A::g()"<<endl;} void h() { cout<<"<4> A::h()"<<endl;} virtual void f() { g(); h(); } };</pre>	<pre>void main() { { B b; b.k(); } }</pre>
--	--

<pre> class B:public A { public: B() {cout<<"<5> B::B() "<<endl; } virtual ~B() {cout<<"<6> B::~~B()"<<endl;} virtual void g() {cout<<"<7> B::g()"<<endl; } void h() {cout<<"<8> B::h()"<<endl; } virtual void k() { f(); g(); h(); } }; </pre>	
---	--

六、（5 分）写出下面程序的运行结果

<pre> #include <iostream.h> class A { public: A() {cout<<1<<endl; } virtual ~A() {cout<<2<<endl;} virtual int Add(int n){return 0;} }; </pre>	<pre> class C: public A { public: C(A& obj,int n):a(obj),num(n) { cout<<3<<endl; } virtual ~C() { cout<<4<<endl; } virtual int Add(int n) { return a.Add(n+num); } private: A& a; int num; }; </pre>
<pre> class B:public A { public: B(int n):num(n) { } virtual ~B() {} virtual int Add(int n) {num+=n;return num;} private: int num; }; </pre>	<pre> void main () { { B b(100); C c1(b,1), c2(c1,2); cout<<c2.Add(50)<<endl; } } </pre>

七、（共 10 分）某程序中关于类 A 和类 B 的部分定义如下：

<pre>#include <iostream.h> class A { public: A(int num):n(num) { } void Show() const {cout<<n<<" ";} private: int n; };</pre>	<pre>class B { public: B(int n1,int n2) {array[0]=new A(n1); array[1]=new A(n2); } ~B() { delete array[0]; delete array[1]; } void Show() const { array[0]->Show(); array[1]->Show(); } private: A* array[2]; };</pre>
--	--

- 1) (5 分)定义并实现类 B 的赋值函数，使得类 B 对象间能够进行深赋值。
- 2) (5 分)定义并实现类 B 的完成深拷贝的拷贝构造函数。

八、(共 10 分)小王编写一个程序时，定义了类 B 和全局函数 f，部分代码如下：

<pre>class B { public: B(int n):data(n) {} int Data() const {return data;} void g1(); void g2(); void g3(); private: const int data; };</pre>	<pre>void f(B& b) { int condition= b.Data(); if(condition ==1) { b.g1();} else if(condition ==5) { b.g2();} else if(condition == 9) { b.g3();} }</pre>
---	--

当把此程序交给用户试用时，针对函数 f，用户提出了一项新的要求：当 condition 为 100 时，依次执行 b 的成员函数 g1()和 g2()。经过进一步了解，小王获悉：以后可能还要增加处理 condition 的值是其它数值时的情况，但这些需要分别处理的不同条件值的个数肯定不多。小王希望他写出的代码既能满足上述要求，又不用每次都改写 f 的代码。请你帮小王重新设计，使得新设计能够满足小王的愿望。简要说明你的设计思想，给出实现代码。