中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

面向对象技术 练习

ffh

- 面向对象编程
- 从C到C++
- 类
- 继承
- 多态
- 操作符重载
- 模板与标准模板库

面向对象编程练习

- 面向对象程序设计着重于（ ）的设计。

  A. 对象 B. 类 C. 算法 D. 数据

- 面向对象程序设计中，把对象的属性和行为组织在同一个模块内的机制叫做（ ）。

  A. 抽象 B. 继承 C. 封装 D. 多态

- 在面向对象程序设计中，类通过（ ）与外界发生关系。

  A. 对象 B. 类 C. 消息 D. 接口

- 面向对象程序设计中，对象与对象之间的通信机制是（）。

  A. 对象 B. 类 C. 消息 D. 接口

- 面向对象的程序设计将数据结构与（ ）放在一起，作为一个相互依存、不可分割的整体来处理。

  A. 算法 B. 信息 C. 数据隐藏 D. 数据抽象

B C D C A

练习

- 下面（　）不是面向对象系统所包含的要素。

  A. 重载　　B. 对象　　C. 类　　D. 继承

- 下面说法正确的是（ 多选 ）。

  A. 将数据结构和算法置于同一个函数内，即为数据封装

  B. 一个类通过继承可以获得另一个类的特性

  C. 面向对象要求程序员集中于事物的本质特征，用抽象的观点看待程序

  D. 同一消息为不同的对象接受时，产生的行为是一样的，这称为一致性

- 下面说法正确的是（多选）。

  A. 对象是计算机内存中的一块区域，它可以存放代码和数据

  B. 对象实际是功能相对独立的一段程序

  C. 各个对象间的数据可以共享是对象的一大优点

  D. 在面向对象的程序中，对象之间只能通过消息相互通信

A BC AD

练习

判断题

- 在高级程序设计语言中，一般用类来实现对象，类是具有相同属性和行为的一组对象的集合，它是创建对象的模板。
- 面向对象程序设计中的消息应该包含“如何做”的信息。
- 一个消息只能产生特定的响应效果。
- 类的设计和类的继承机制实现了软件模块的可重用性。
- 学习C++语言是学习面向对象的程序设计方法的唯一途径。

- 书p3： 5、6、10、11
- p5：4、5、6、7、8、9、10
- p8：2、3
- P11：6
- P13：1

- **1.1.4 课后练习**

  5. 通过自顶向下设计法进行函数分解时，可能会出现的串联改变现象指的是什么？

  10. 如果在面向对象语言中为类Human设计“年龄”和“性别”属性，它们的表现形式是什么？

- **1.3.3 课后练习**

  2. 如果一个类或对象是一个服务器，什么可以成为客户？

  4. 客户通过什么方式请求类或对象提供服务？

- **1.5.3 课后练习**

  10. 为什么说如果一个组件没有暴露任何接口就是一个对用户无用的组件？

  11. 如果组件设计良好，使用者需要知道这个组件是使用什么语言编写的吗？

- 面向对象编程
- 从C到C++
- 类
- 继承
- 多态
- 操作符重载
- 模板与标准模板库

二

- 在整型指针变量p2、p3的定义中，错误的是（ ）。

  A. int p1，*p2，p3;　　B. int *p2，p1，*p3;

  C. int p1，*p2=&p1，*p3;　　D. int *p2，p1，*p3=&p1;

- 若有定义“double xx=3.14，*pp=&xx； ”，则*pp等价于（ ）。

  A. &xx　　B. *xx　　C. 3.14　　D. xx

- 下面对引用的描述中（ ）是错误的。

  A. 引用是某个变量或对象的别名　　B. 建立引用时，要对它初始化

  C. 对引用初始化可以使用任意类型的变量

  D. 引用与其代表的对象具有相同的地址

- 对重载的函数来说，下面叙述不正确的是（ ）。

  A. 参数的类型不同　B. 参数的顺序不同　　C. 参数的个数不同

  D. 参数的个数、类型、顺序都相同，但函数的返回值类型不同

- 下列有关设置函数参数默认值的描述中，（ ）是正确的。

  A. 对设置函数参数默认值的顺序没有任何规定

  B. 函数具有一个参数时不能设置默认值

  C. 默认参数要设置在函数的原型中，而不能设置在函数的定义语句中

  D. 设置默认参数可使用表达式，但表达式中不可用局部变量

- 下面说法正确的是（ 多选 ）。

  A. 所有的函数都可以说明为内联函数

  B. 具有循环语句、switch语句的函数不能说明为内联函数

  C. 使用内联函数，可加快程序执行的速度，但会增加程序代码的大小

  D. 使用内联函数，可以减小程序代码大小，但使程序执行的速度减慢

- 一个函数功能不太复杂，但要求被频繁调用，应选用（ ）。

  A. 内联函数　　B. 重载函数　　C. 递归函数　　D. 嵌套函数

D BC A

下面程序的输出结果为：

```
#include<iostream>
using namespace std;
int main（）
{   int x=10，&y=x;
    cout<<"x="<<x<<"，y="<<y<<endl;
    int *p=&y;  *p=100;
    cout<<"x="<<x<<"，y="<<y<<endl;
    return 0;
}
```

x=10，y=10；x=100，y=100。

下面程序的输出结果为

```
#include<iostream>
using namespace std;
int main（）
{    int num=50;  int& ref=num;
     ref=ref+10;
     cout<<"num="<<num<<endl;
     num=num+40;
     cout<<"ref="<<ref<<endl;
     return 0;
}
```

num=60ref=100

2.1.1 课后练习

3.解释错误：

```
namespace foo {
    void showdate(int,int);
    //...          }
namespace bar {
    void showdate(int);
    //...          }
using foo::showdate;
showdate(23);
```

2.4.7 课后练习

5. 以下代码段有什么错误？ (i范围)

```
void reverse_and_print( int a[ ], int size ) {
    for ( int i = 0; i < size; i++ )
        a[ i ] = 2 * i;
    int temp;
    for ( i = 0; i < size / 2; i++ ) {
        temp = a[ i ];
        a[ i ] = a[ size - 1 - i ];
        a[ size - 1 - i ] = temp;
    }
    for ( i = 0; i < size; i++ )
        cout << a[ i ] << '\n';
}
```

2.5.11 课后练习

练习4到练习35为代码正误判断，如果代码有错，说明问题所在；如果正确，写出其输出，假设用4 294 967 295代表无穷大。

5.

```
string s1="C++ is great fun";
char s2[30];
s2=s1;
cout<<s2<<endl;
```

2.8.1 课后练习

6. 解释如下代码中的错误：

```
try {
    int  *ptr = new int;   }
catch (bad_alloc)  {
    cerr<<“.....”;
    exit(EXIT_FAILURE);  }
*ptr = s;
```

- 面向对象编程
- 从C到C++
- 类
- 继承
- 多态
- 操作符重载
- 模板与标准模板库

三

- 以下不属于类访问权限的是（ ）。

  A. public　　B. static　　C. protected　　D. private

- 有关类的说法不正确的是（ ）。

  A. 类是一种用户自定义的数据类型

  B. 只有类的成员函数才能访问类的私有数据成员

  C. 在类中，如不做权限说明，所有的数据成员都是公有的

  D. 在类中，如不做权限说明，所有的数据成员都是私有的

- 在类定义的外部，可以被任意函数访问的成员有（　）。

  A. 所有类成员　　B. private或protected的类成员

  C. public的类成员　　D. public或private的类成员

B BC C

三

- 关于类和对象的说法（ ）是错误的。

  A. 对象是类的一个实例

  B. 任何一个对象只能属于一个具体的类

  C. 一个类只能有一个对象

  D. 类与对象的关系和数据类型与变量的关系相似

- 设MClass是一个类，dd是它的一个对象，pp是指向dd的指针，cc是dd的引用，则对成员的访问，对象dd可以通过（ ）进行，指针pp可以通过（ ）进行，引用cc可以通过（ ）进行。

  A. ::　　B. .　　C. &　　D. ->

- 关于成员函数的说法中不正确的是（　）。

  A. 成员函数可以无返回值　　B. 成员函数可以重载

  C. 成员函数一定是内联函数　D. 成员函数可以设定参数的默认值

三

- 下面对构造函数的不正确描述是（　）。

  A. 系统可以提供默认的构造函数

  B. 构造函数可以有参数，所以也可以有返回值

  C. 构造函数可以重载

  D. 构造函数可以设置默认参数

- 假定A是一个类，那么执行语句“A a，b（3），*p；”调用了（ ）次构造函数。

  A. 1　　B. 2　　C. 3　　D. 4

- 下面对析构函数的正确描述是（ ）。

  A. 系统可以提供默认的析构函数　　B. 析构函数必须由用户定义

  C. 析构函数没有参数　　D. 析构函数可以设置默认参数

B B AC

三

- 类的析构函数是（ ）时被调用的。

  A. 类创建　　B. 创建对象　C. 引用对象　　D. 释放对象

- 创建一个类的对象时，系统自动调用（ ）； 撤销对象时，系统自动调用（ ）。

  A. 成员函数　B. 构造函数　C. 析构函数　D. 复制构造函数

- 通常拷贝构造函数的参数是（ ）。

  A. 某个对象名　　B. 某个对象的成员名

  C. 某个对象的引用名　　D. 某个对象的指针名

- 关于this指针的说法正确的是（ ）。

  A. this指针必须显式说明

  B. 当创建一个对象后，this指针就指向该对象

  C. 成员函数拥有this指针　　D. 静态成员函数拥有this指针。

D B C C B

三

- 下列关于子对象的描述中，（ ）是错误的。

  A. 子对象是类的一种数据成员，它是另一个类的对象

  B. 子对象可以是自身类的对象

  C. 对子对象的初始化要包含在该类的构造函数中

  D. 一个类中能含有多个子对象作其成员

- 对new运算符的下列描述中，（ ）是错误的。

  A. 它可以动态创建对象和对象数组

  B. 用它创建对象数组时必须指定初始值

  C. 用它创建对象时要调用构造函数

  D. 用它创建的对象数组可以使用运算符delete来一次释放

- 对delete运算符的下列描述中，（ ）是错误的。

  A. 用它可以释放用new运算符创建的对象和对象数组

  B. 用它释放一个对象时，它作用于一个new所返回的指针

  C. 用它释放一个对象数组时，它作用的指针名前须加下标运算符［ ］

  D. 用它可一次释放用new运算符创建的多个对象

- 关于静态数据成员，下面叙述不正确的是（ ）。

  A. 使用静态数据成员，实际上是为了消除全局变量

  B. 可以使用“对象名.静态成员”或者“类名::静态成员”来访问静态数据成员

  C. 静态数据成员只能在静态成员函数中引用

  D. 所有对象的静态数据成员占用同一内存单元

- 对静态数据成员的不正确描述是（ 多选 ）。

  A. 静态成员不属于对象，是类的共享成员

  B. 静态数据成员要在类外定义和初始化

  C. 调用静态成员函数时要通过类或对象激活，所以静态成员函数拥有this指针

  D. 只有静态成员函数可以操作静态数据成员

- 下面的选项中，静态成员函数不能直接访问的是（ ）。

  A. 静态数据成员　　B. 静态成员函数

  C. 类以外的函数和数据　　D. 非静态数据成员

```
#include<iostream>
using namespace std;
class Test
{   private:        int num;
    public:         Test（）;
                    Test（int n）;
};
Test::Test（） { cout<<"Init defa"<<endl;          num=0; }
Test::Test（int n） { cout<<"Init"<<" "<<n<<endl;  num=n;        }
int main（）
{   Test  x[2];   Test y(15);    return 0;  }
```

```
#include<iostream>
using namespace std;
class Xx
{   private:        int num;
    public:         Xx（int x）{num=x;}
                    ~Xx（）{cout<<“x= ”<<num<<endl;}
};
int main（）
{   Xx  w(5);
    cout<<"Exit main"<<endl;
    return 0;
}
```

```
Exit main
x= 5
```

```
class  Book
{   public:         Book（int w）;
                    static int sumnum;
    private:        int num;
};
Book::Book（int w）{   num=w;  sumnum - = w;}
int Book::sumnum=120;
int main（）
{   Book b1（20）;        Book b2（70）;
    cout<<Book::sumnum<<endl;
    return 0;
}
```

3.1.8 课后练习

5. 如果用class作关键字进行类声明，成员在默认情况下是私有的还是公有的？
7. 是否所有的成员函数都能够在类声明之中进行定义？
8. 是否所有的成员函数都能够在类声明之外进行定义？

3.3.5 课后练习

1. 为什么通过引用方式而非传值方式来传递和返回对象（特殊情况除外）？
2. 为什么不能以引用方式返回一个auto对象？
3. 假设C是一个类，f是一个顶层函数，请解释如下f的两种声明方式有何区别：

```
void  f(c& c)  { //.....}
void  f(const  c& c)  { //.....}
```

4. 类的设计者将一个成员函数标记为const意味着什么？
6. 对一个接受字符串参数的成员函数，为什么通常为其设计两个重载版本，分别用来处理string类型参数和const char*类型参数？

3.5.12 课后练习

1. 解释下面代码中的错误：

```
class  C  {
    c ( );   //default constructor
     //...
  }
```

2. 解释下面代码中的错误：

```
class  z  {
   void   z ( );      //default constructor
    //...
}
```

3.5.12 课后练习

3. 能否对类的构造函数进行重载？

4. 类的构造函数可以是私有的吗？

5. 类的构造函数必须定义在类声明之外吗？

6. 在如下类声明中，哪个构造函数是默认构造函数？

```
class  c  {
    public:
        c ( ) ;
        c ( int ) ;
         //...
     }
```

3.5.12 课后练习

7. 解释下面代码中的错误：

```
class k {
   private:   k() ;  }
int main()  {
    k  k1;
    return 0;  }
```

10. 拷贝构造函数的作用是什么？

12. 如类的设计者没有提供拷贝构造函数，编译器是否会提供一个？

14. 在什么情况下应该为类设计一个拷贝构造函数？

15. 什么是转型构造函数？

3.7.4 课后练习

1. 对象数据成员和类数据成员之间有什么区别？

3. 解释下面代码段中的错误：

```
#include <iostream>
using namespace std;
class C {
public:
    void f() { cout << ++x << '\n'; }
private:
    static int x;
};
int main() {
  C c1;
  c1.f();
  return 0;
}
```

- 面向对象编程
- 从C到C++
- 类
- 继承
- 多态
- 操作符重载
- 模板与标准模板库

继承

- 下面对派生类的描述中，错误的是（ ）。

  A．一个派生类可以作为另外一个派生类的基类

  B．派生类至少有一个基类

  C．派生类的成员除了它自己的成员外，还包含了它的基类的成员

  D．派生类中继承的基类成员的访问权限到派生类中保持不变

- 当保护继承时，基类的（）在派生类中成为保护成员，不能通过派生类的对象来直接访问。

  A．任何成员　　B．公有成员和保护成员

  C．公有成员和私有成员　　D．私有成员

- 派生类的对象对它的基类成员中（ ）是可以访问的。

  A．公有继承的公有成员；B．公有继承的私有成员；

  C．公有继承的保护成员；D．私有继承的公有成员。

- 在公有派生情况下，有关派生类对象和基类对象的关系，不正确的叙述是（ ）。

  A．派生类的对象可以赋给基类的对象

  B．派生类的对象可以初始化基类的引用

  C．派生类的对象可以直接访问基类中的成员

  D．派生类的对象的地址可以赋给指向基类的指针

- 派生类的构造函数的成员初始化列表中，不能包含（ ）。

  A．基类的构造函数；　　B．派生类中对象成员的初始化；

  C．基类的对象成员的初始化；D．派生类中一般数据成员的初始化

- 类O定义了私有函数F1。P和Q为O的派生类，定义为class P: protected O{…}； class Q: public O{…}。 ( )可以访问Fl。

  A． O的对象　　B． P类内　　C． O类内　　D． Q类内

- 有如下类定义：

```
class XA {
            int x;
    public:  XA(int n) {x=n;}          };
class XB: public XA{
    int y;
  public:   XB(int a,int b);  };
```

在构造函数XB的下列定义中，正确的是（ ）。

A．XB::XB（int a，int b）：x(a)，y(b){ }

B．XB::XB（int a，int b）：XA(a)，y(b) { }

C．XB::XB（int a，int b）：x(a)，XB(b){ }

D．XB::XB（int a，int b）：XA(a)，XB(b){ }

有如下程序：

```
class Base{
    private:        void fun1( ) const {cout<<”fun1”;}
    protected:      void fun2( ) const {cout<<”fun2”;}
    public:         void fun3( ) const {cout<<”fun3”;}    };
class Derived : protected  Base{
    public:         void fun4( ) const {cout<<”fun4”;}    };
int main(){
    Derived obj;  obj.fun1( );  //① obj.fun2( );  //②
    obj.fun3( );  //③        obj.fun4( );  //④
}
```

其中没有语法错误的语句是（ ）。

A．①　　B．②　　C．③　　D．④

有如下类定义：

```
class MyBASE{
                    int k;
    public:         void set(int n) {k=n;}
                    int get( ) const {return k;}    };
class MyDERIVED: protected MyBASE{
    protected:      int j;
    public:         void set(int m,int n){MyBASE::set(m);j=n;}
                    int get( ) const{return MyBASE::get( )+j;}
};
```

则类MyDERIVED中保护成员个数是（ ）。

A．4　　B．3　　C．2　　D．1

```
#include<iostream>
using namespace std;
class A {  public:  A( ) {cout<<”A”;} };
class B { public:B( ) {cout<<”B”;} };
class C: public A{
            B b;
    public:          C( ) {cout<<”C”;}    };
int main( )
{  C obj;   return 0;}
```

执行后的输出结果是（）。

A．CBA　　B．BAC　　C．ACB　　D．ABC

写出程序运行结果

```
class B1{
    public:        B1(int i) {        cout<<"constructing B1 "<<i<<endl; }
                   ~B1( )   {         cout<<"destructing B1 "<<endl;   }    };
class B2 {
    public:        B2( ){    cout<<"constructing B3 *"<<endl; }
                   ~B2( ){  cout<<"destructing B3"<<endl; }  };
class C:public B2, virtual public B1 {
                   int j;
    public:        C(int a,int b,int c):B1(a),memberB1(b) ,j(c){}
    private:       B1 memberB1;  B2 memberB2; };
int main( )
{   C obj(1,2,3);  }
```

```
constructing B1 1
constructing B3 *
constructing B1 2
constructing B3 *
destructing B3
destructing B1
destructing B3
destructing B1
```

写出程序运行结果

```
class B{
    public:         void f1(){cout<<"B::f1"<<endl;}   };
class D:public B{
    public:         void f1(){cout<<"D::f1"<<endl;}    };
void f(B& rb){    rb.f1();}
int main( ){
    D d;
    B b,&rb1=b,&rb2=d;
    f(rb1);         f(rb2);
    return 0;
}
```

B::f1
B::f1