课 程 名 称：面向对象与C++程序设计 试 卷： A 是否开卷 否

授课院 (系)： 软件学院 考试日期：2017 年 5 月 11日 试卷共16页

**一 、单项选择题（30分, 每题2分）**

1. 关于类对象，以下理解正确的为（ ）。

【A】一个类的对象只有一个，一个对象必属于某个类

【B】一个类可以实例化很多个对象，一个对象可以属于多个类

【C】同一个类的所有对象可调用的成员函数都是相同的

【D】同一个类的所有对象的数据成员都是相同的，可使用的成员函数不同

1. 以下关于类理解的不正确的为（ ）。

【A】类是一种抽象数据类型，它将数据与相关操作进行封装

【B】类中成员的访问权限有public、private和protected

【C】类的成员函数可以直接访问本类的任何数据成员

【D】类的成员函数只能直接访问本类的public数据成员

1. 对const的使用，以下描述错误的为（ ）。

【A】可以用const修饰对象、数据成员或成员函数

【B】常对象只能调用类的常成员函

【C】常数据成员只能由构造函数通过初始化列表进行初始化

【D】常成员函数不能直接修改类的数据成员，但能调用该类的非常成员函数

const成员函数不能调用非const的成员函数，因为有可能修改成员数据

1. 在日期类Date中重载后置自增运算符，以下声明最合适的为（ ）。

【A】void operator ++ (Date ) ;

【B】Date& operator ++ ( ) ; 后置自增运算符

返回的是操作之前的值 int参数没有意义，只是用来区分前置和后置、

返回值为对象的引用，表示返回自增之前的对象

【C】void operator ++ ( int ) ;

【D】Date operator ++ ( int ) ;

1. 关于C++中的重载机制，以下描述正确为（ ）。

【A】派生类中可以定义与基类中同名的函数，这也是一种函数重载x

【B】函数名与参数列表相同，但是返回值不同，这是合法的函数重载x

【C】所有的运算符都可以通过成员函数或友元函数的形式进行重载

【D】类中的析构函数不能被重载

1. 关于友元，以下理解正确的为（）。
2. 类的友元函数可以直接访问类中的public与protected成员，

但不能直接访问private成员

1. 若类A为类B的友元类，则A中所有成员函数都是类B的友元函数
2. 若类A为类B的友元类，则B中所有成员函数都是类A的友元函数
3. 类的友元函数能够访问本类中的所有成员，也能访问派生类的所有成员
4. 关于基类与派生类，以下理解正确的为（ ）。
5. 派生类从基类继承全部成员，也可以直接访问基类所有成员x
6. 派生类是对基类的扩展，可以新增数据成员与成员函数
7. 派生类从基类继承部分成员，可以直接访问这些基类成员x
8. 派生类可以使用基类的成员，基类也可以使用派生类成员
9. 有如下三个类，若实例化对象D d，则输出结果为（ ）。

class A

{ public: A(){cout<<” A ”;}

};

class B

{ A a;

public: B(){cout<<”B ”;}

};

class D：public B

{ public: D( ){cout<<”D ”}

};

1. DBA 【B】ABD 【C】DAB 【D】BAD
2. 对于Animal类的理解不正确的为（ ）。
   * 1. Animal类为抽象类，类中所有成员函数均为纯虚函数

class Animal

{ char name[10];

public:

Animal ( char\* );

virtual void move( Food& ) =0;

virtual void eat( Food& );

void sleep ();

};

* + 1. Animal类不能实例化对象, 但能定义指针Animal\* p;
    2. Animal类为抽象类，可以声明函数void fun(Animal\*);
    3. Animal类为抽象类，可以声明函数void fun(Animal&);

1. 关于C++的多态性，以下理不正确的是（ ）。

【A】不可以通过函数重载实现C++的多态性

【B】可以通过虚函数实现C++的多态性

【C】封装、继承、多态是面向对象程序设计的三大特征

【D】在公有继承体系下，用基类指针或引用对象调用虚函数能实现多态

1. 对于如下函数模板描述正确的为（ ）。

【A】该函数模板能够比较任何类型的三个对象，并返回其中最大的对象

【B】执行max (3, 4.5f, 6.7); 函数模板实例化为函数max<double>

【C】执行max(‘3’,’5’,’7’); 函数模板实例化为函数max<int>

【D】若要比较三个char\*的字符串，并返回其中最大的字符串，需要特化该函数模板

template <class T>

T max(T &x, T &y, T& z)

{

T temp= x>y ? x : y ;

temp= temp>z ? temp : z;

return temp;

}

1. 关于C++异常处理机制，以下描述错误的为（ ）。

【A】通常将需要检测异常的代码放在try语句块中

【B】throw语句可以抛出任何类型的异常对象，包括基本数据类型

【C】引发异常的throw语句和捕获异常的catch语句必须在同一函数中

【D】检测异常的try语句和捕获异常的catch语句必须在同一函数中

1. 关于流的理解正确的为（ ）。

【A】输出流将数据从内存传送到某个设备中，读磁盘文件文件为输出操作

【B】输入流将数据从某个设备送到内存缓冲区，写磁盘文件文件为输入操作

【C】将优盘中的文件打开并通过投影仪播放，进行的为输入操作

【D】将优盘中的文件打开并通过投影仪播放，即进行输入操作又有输出操作

1. 以下标准流对象中属于istream类为（ ）。

【A】cin 【B】cerr 【C】clog 【D】cout

1. 在如下继承层次中，当定义派生类Drived对象时，调用构造函数的顺序为（ ）。

class B{…};

class B1: virtual B{…};

class B2: virtual B{…};

class D : public B2, public B1{…};

【A】 B(), B1(), B(), B2(), D()

【B】 B (), B2(), B(), B1(), D()

【C】 B(), B2(), B1(), D()

【D】 B(), B1(), B2(), D()

**二、程序阅读题（20分, 共4题）**

写出下面程序的运行结果

1. （5分）

#include <iostream>

#include <cstring>

using namespace std;

class Goods//商品类

{

protected:

char\* name; //名称

double price; //价格

public:

Goods( );

~Goods( );

Goods( char\* n, double d);

void setPrice(double);

void setName(char\*);

void show()const;

};

Goods::Goods( ):price(0)

{

name = new char [10];

strcpy(name,"NULL");

cout<<"default construct"<<endl;

}

Goods::Goods(char\* n, double d)

{

name = new char [strlen(n)+1];

if(name) strcpy(name,n);

price = d>0?d:0;

cout<<"construct"<<name<<endl;

}

Goods::~Goods( )

{

cout<<"destruct "<<name<<endl;

if(name) delete[]name;

}

void Goods::setPrice(double d)

{

price = d>0?d:0;

}

void Goods::setName(char\* n)

{

if(name) delete[]name;

name = new char [strlen(n)+1];

if(name) strcpy(name,n);

}

void Goods::show()const

{

cout<<"商品名称："<<name<<",商品价格:"<<price<<"RMB"<<endl;

}

int main()

{

Goods dress("T-shirt",129),food;

food.setName("Milk");

food.setPrice(2.99);

dress.show();

food.show();

return 0;

}

1. （5分）

include <iostream>

#include <cstring>

using namespace std;

//图书类型

class Books

{

protected:

char\* name; //名称

double price; //价格

public:

Books( char\* n, double d=0);

Books( Books& );

~Books( );

void show()const;

double compare(Books);

};

Books:: Books(char\* n, double d)

{

name = new char [strlen(n)+1];

if(name) strcpy(name,n);

price = d>0?d:0;

}

Books:: Books (Books & g):price(g.price)

{

name = new char [strlen(g.name)+1];

if(name) strcpy(name,g.name);

cout<<"copy construct"<<endl;

}

Books::~ Books ( )

{

cout<<"destruct "<<name<<endl;

if(name) delete[]name;

}

double Books:compare(Books g)

{

return price-g.price;

}

void Books::show()const

{

cout<<"图书名称："<<name<<",图书价格:"<<price<<"RMB"<<endl;

}

int main()

{

Books b1("Math",129),b2("English",99);

cout<< "价格相差"<<b1.compare(b2)<<endl;

return 0;

}

1. （5分）

#include <iostream>

#include <cstring>

using namespace std;

class Student

{

public:

Student(int = 1 , char\* ="学生") ;

Student(const Student& s) ;

virtual void print()const;

protected:

char name[10]; //姓名

int grade; //年级

};

Student::Student(int no , char\* n )

{

grade = no>0?no:0;

if(n) strcpy(name,n);

cout<<"construct Student"<<endl;

}

void Student::print()const

{

cout<<"学生姓名："<<name<<" 年级：" <<grade<<endl;

}

class CollegeStudent:public Student

{

public:

CollegeStudent(int = 1, char\* = "大学生", double score=0);

double fee() const ;

void print() const;

protected:

double gpa ;

};

CollegeStudent:: CollegeStudent(int g, char \* n, double s):Student(g, n),gpa(s)

{cout<<"costruct College"<<endl;}

double CollegeStudent::fee()const

{

if(grade==1||grade==2) return 6000;

if(grade==3||grade==4) return 12000;

if(grade>4) return 10000;

}

void CollegeStudent::print() const

{

Student::print();

cout<<" 平均分 = "<< gpa<<" 学费"<< fee()<<endl;

}

int main()

{

Student\* pS [3];

pS[0] = new Student;

pS[1] = new CollegeStudent;

pS[2] = new CollegeStudent(2," 明明",3.5);

for(int i=0;i<3;i++)

pS[i]->print();

return 0;

}

4. （5分）

#include <iostream>

using namespace std;

class MyException { };

void testFun(int test) ;

int main()

{

try

{

for(int i=0; i<3; i++)

testFun(i);

cout << "in try"<<endl;

}

catch(MyException&)

{

cout<<"catch MyException "<<endl;

}

return 0;

}

void testFun(int test)

{

try

{

if(test==2)

throw MyException();

if(test==1)

throw "Error!";

cout<<test<<endl;

}

catch(const char\*)

{ cout<<"catch char\* "<<endl;

}

cout<<"End of testFun()\n";

}

**三、分析程序，回答问题（20分, 每题5分，共4题）**

1.（5分）定义数组类模板，在横线处给出相关代码。

#include<iostream>

using namespace std;

const int MAX=100; //数组容量

template< typename T >

class Array

{

public :

Array ( int s=0 ) ;

T& operator[](int i);

friend ostream& operator<<(ostream&, const Array<T>&);

protected :

T a[MAX+1] ; //下标范围1~MAX

int size ; //数组中存储元素数目

} ;

//类模板成员函数

template<typename T>

Array<T>::Array(int s)

{

if( )

size=s;

else size=0;

}

template < typename T >

//重载[]

{

if( )

{ i = 0;

cout<<"Out of range!";

}

return a[i];

}

//声明模板

ostream& operator<<(ostream& out, const Array<T>& a)

{

for( int i=1;i<=a.size;i++)

out<< a[i]<<" ";

//返回

}

2. （5分）设计账户类，在横线处给出相关代码

#include<iostream>

#include<cstring>

using namespace std;

class Account //账户类

{

protected:

char id[6]; //账户号

double balance; //结余

static double rate; //月利率

public:

Account( char\* ="10001", double=0);

double withdraw( double cost); //存款

double deposit( double cost); //取款

double settle( );//计算利息

double getBalace( )const{return balance;}

static double getRate( ){return rate;}

};

//初始化rate=0.005;

Account::Account( char\* n, double b)

{

if(n) strcpy(id,n);

balance=b>0?b:0;

}

double Account::deposit( double cost )

{

if(cost>0)

{ //结余增加cost

return cost;

}

else

return 0;

}

double Account::settle( )

{

//定义并计算利息interest

balance += interest;

return interest; }

int main()

{

cout<<"月利率:"<< ; //输出 月利率:0.005

Account a1, a2("10003",1000);

cout<<"存款:"<< a1.deposit(200)<<endl; //输出 存款:200

cout<<"利息:"<< ; //输出a2的利息:5

return 0;

}

3．已知如下商品类，在横线处给出相关代码，完成将多个商品对象按照格式化的方式写入文件中。

//商品类

class Goods

{

protected:

char\* name; //名称

double price; //价格

public:

Goods( char\* n=NULL, double d=0);

~Goods( ){ if(name) delete[]name;};

double getPrice()const{return price;}

const char\* getName()const{return name;}

void show()const;

};

Goods::Goods(char\* n, double d)

{

if(n) {

name = new char [strlen(n)+1];

strcpy(name,n);

}

else {

name = new char[10];

strcpy(name,"Goods");

}

price = d>0?d:0;

}

void Goods::show()const //向屏幕输出商品信息

{

cout<<"商品名称："<<name <<",商品价格:"<<price<<"RMB"<<endl;

}

void outGoods( Goods\* gg, int n, ofstream& out) //向文件输出商品信息

{

for(int i=0;i<n;i++) //格式化写文件

}

int main()

{

Goods g[3] = {Goods("T-shirt",129),Goods("Milk",2.99)};

for(int i=0;i<3;i++)

//向屏幕输出商品信息

ofstream outFile;

//打开文件outFile.txt

//向文件outFile.txt 中写入 数组goods的商品信息

//关闭文件outFile.txt

return 0;

}

程序输出（在屏幕上显示与写入文件的格式相同）

商品名称： T-shirt, 商品价格: 129RMB

商品名称： Milk, 商品价格: 2.99RMB

商品名称： Goods, 商品价格: 0RMB

4. 分析代码，回答以下问题。

#include <iostream>

#include <cstring>

using namespace std;

class Student

{ char name[10]; //姓名

int grade; //年级

public:

Student(int = 1 , char\* ="学生") ;

virtual void print()const

{ cout<<"\n学生姓名："<<name<<" 年级：" <<grade; }

};

Student::Student(int no , char\* n )

{

grade = no>0?no:0;

if(n) strcpy(name,n);

}

class CollegeStudent: public Student

{

public:

CollegeStudent(int = 1, char\* = "大学生", double score=0);

double fee() const ;

void print() const;

protected:

double gpa ;

};

CollegeStudent:: CollegeStudent(int g, char \* n, double s)

{

grade = no>0?no:0;

if(n) strcpy(name,n);

gpa = s>0?s:0;

}

double CollegeStudent::fee()const

{

if(grade==1||grade==2) return 6000;

if(grade==3||grade==4) return 12000;

if(grade>4) return 10000;

}

void CollegeStudent::print() const

{

Student::print();

cout<<" 平均分 = "<< gpa<<" 学费"<< fee()<<endl;

}

* 1. CollegeStudent类中包含哪些数据成员，它们的访问权限分别是什么？（3分）
  2. 分析CollegeStudent类的构造函数存在语法错误，给出正确的定义。（2分）

**四、编程题（20分，共3题）**

1. （6分）已知如下日期类Date的声明，给出相关成员函数的定义，使之能够满足相关的功能。

#include<iostream>

using namespace std;

class Date

{

public:

Date(int y=2000, int m=1, int d=1);

bool isDate( int y, int m, int d); //判断日期合法性

void showDate() const;

bool isLeapYear() const

{ return ( year%4 == 0 && year%100 != 0 )||( year%400 == 0); }

private:

int month;

int day;

int year;

static int Maxdays[13];//每月最大日期

};

int Date::Maxdays[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

int main()

{

Date d1, d2(2012,7,31);

d1.showDate(); //输出 1/1/2000

d2.showDate(); //输出 7/31/2012

Date d3(2012,2,29);

d3.showDate();//输出无效日

return 0;

}

//类成员函数实现

bool Date :: isDate(int y, int m, int d )

{

if(m<1 || m>12)

{

cout<<"无效月"<<endl;

return false;

}

if(m==2&&isLeapYear()&&day==29)

return true;

if(d<1 || d>Maxdays[m])

{

cout<<"无效日"<<endl;

return false;

}

return true;

}

2. （8分）利用上题中的日期类Date，设计一个会员类Member，属性包括会员号、积分、入会时间及会员折扣discount（假设所有会员折扣率为0.9），结算成员函数根据消费金额cost计算, 假设每消费100元积1分，返回打折后的实际消费金额类声明形式如下，请给出该类的完整定义。

class Member

{

protected:

char id[6]; //会员号

int count; //积分

Date roll; //入会时间

static double discount; //折扣率

public:

Member( char\*, Date&, int=0);

double settle( double cost); //结算

int getCount()const;

void show()const;

static double getDiscount(){return discount;}

};

double Member::discount=0.9;

int main()

{

Date d1, d2(2017,5,11);

Member Kate("10002",d1), Gates("20001",d2,10);

cout<<"实收金额"<<Kate.settle(2000)<<endl;

Gates.show();

Kate.show();

return 0;

}

输出结果为：

实收金额1800

No.20001,积分：10

注册日期：5/11/2017

No.10002,积分：20

注册日期：1/1/2000

3. （6分）设计一个函数模板void reverse(T a[], int size); 将长度为size的数组T a[ ]中的元素翻转存储。在main函数中使用数组double d[]= {0.1,2.3,3.4,4.5,5.6}测试该模板，翻转数组后正向遍历该数组输出结果为5.6 4.5 3.4 2.3 0.1。给出函数模板的完整定义。

#include<iostream>

using namespace std;

int main()

{

double d[ ] = {0.1,2.3,3.4,4.5,5.6};

int n = sizeof(d)/sizeof(double);

reverse( d,n);

for(int i =0; i<n;i++)

cout<<d[i]<<" ";

return 0;

}

**五、面向对象程序设计（10分，共1题）**

利用面向对象思想为某银行设计一个管理用户的程序框架。用户（User）可以开设不同类型的帐户（Account）， 每个帐户要有账号id与余额balance，可分为储蓄帐户（SavingAccount）与信用卡帐户（CreditAccount），以后可能会分出新的帐户类型（如基金帐户，股票帐户等）。不同帐户都可以进行取款withdraw( )、存款deposit( )、结算settle()与输出print(), 但操作方式不同，例如：储蓄帐户可以进行定期与活期存款，而信用卡可以根据信用额度进行透支。

用户（User）的属性包括：姓名name与管理账户的数组account 以及目前个数num，每个用户最多开设10个帐户。用户可以按账号id查询某个帐户的结余find( ), 可以计算所有账户的余额总和total( )，也可显示所有帐户的相关信息print()。

要求：设计出所有的类，只需声明出类的主要属性和成员函数，给出类的声明（无需类的实现），注意合理设计类的关系及其虚函数。给出User类的声明以及输出所有账户信息的show方法的实现，要求调用每种帐户的print()输出其各自的信息。

课 程 名 称：面向对象与C++程序设计 试 卷： A 是否开卷 否

授课院 (系)： 软件学院 考试日期：2015 年 5 月 10 日 试卷共16页

**一 、单项选择题（30分, 每题2分）**

1. 关于类的理解，以下不正确的为（ ）。

【A】类是一种抽象的数据类型，它将数据与相关操作进行封装

【B】类中成员的访问权限有public、private和protected

【C】类的成员函数可以直接访问本类的任何数据成员

【D】类的成员函数只能直接访问本类的public数据成员

1. 关于对象，以下理解正确的为（ ）。

【A】一个类的对象只有一个，一个对象必属于某个类

【B】一个类可以实例化很多个对象，一个对象可以属于多个类

【C】同一个类的所有对象可调用的成员函数都是相同的

【D】同一个类的对象的数据成员都是相同的，但可使用的成员函数不同

1. 在日期类Date中重载前置自增运算符，最合适的声明形式为（ ）。

【A】void operator ++ (Date ) ;

【B】Date& operator ++ ( ) ;

【C】Date operator ++ ( int ) ;

【D】void operator ++ ( int ) ;

1. 以下函数可以实现重载的为（ ）。

【A】void sort( int\* p, int n ); void sort( int a[], int n );

【B】double max( double, double); int max(int, int);

【C】char fun( ) const; int fun( ) const;

【D】float ave1( int, int, int); float ave2(int, int, int);

1. 对const的使用，以下描述错误的为（ ）。

【A】可以用const修饰对象、数据成员或成员函数

【B】常对象只能调用类的常成员函

【C】常数据成员只能由构造函数通过初始化列表进行初始化

【D】常成员函数不能直接修改类的数据成员，但能调用该类的非常成员函数

1. 以下关于友元的理解，不正确的为（ ）。
2. 类的友元函数可以直接访问类中的所有成员
3. 类的友元函数便于访问类中私有成员，但破坏类的封装性
4. 若类A为类B的友元类，则B中所有成员函数都是类A的友元函数
5. 若A类的成员函数fun( )为B类的友元函数，则A::fun()中可访问B的私有成员
6. 关于基类与派生类，以下理解正确的为（ ）。
7. 派生类是对基类的扩展，可以新增数据成员与成员函数
8. 派生类从基类继承所有成员，派生类中可以直接使用基类的任何成员
9. 派生类中可以使用基类的任何成员，但不能对成员函数进行重写
10. 派生类可以使用基类的成员，基类也可以使用派生类成员
11. 有如下三个类，若实例化D类对象d，则输出结果为（ ）。

class B1

{ public: B1(){cout<<” B1 ”;}

};

class B2

{ public: B2(){cout<<”B2 ”;}

};

class D：public B2，public B1

{ public:

D( ){cout<<”D ”}

};

1. B2 B1 D
2. B1 B2 D
3. D B1 B2
4. D B2 B1
5. 对于下面两个类，描述正确的为（ ）。

//鸟类

class Bird

{ char name[10];

public:

Bird ( char\* );

virtual void fly( )=0;

virtual void eat( );

void sleep () ;

};

//鸵鸟类

class Ostrich: Bird

{

public:

Ostrich ( char\*) ;

virtual void eat( ) ;

void sleep( ) ;

void run( );

};

【A】Brid类为抽象类，它是Ostrich类为公有基类

【B】Brid类为抽象类，所有的成员函数均为虚函数

【C】Ostric类为具体类，不包含任何纯虚函数

【D】Ostric类为抽象类，不能用其实例化对象

1. 关于C++的多态性，以下理不正确的是（ ）。

【A】多态性分为编译时的静态多态性和运行时的动态多态性

【B】函数重载与运算符重载属于静态多态

【C】虚函数实现的重载属于动态多态

【D】只要有虚函数就能实现动态多态

1. 对于如下函数模板描述正确的为（ ）。

template <class T>

void Swap(T &x, T &y)

{

T temp= x;

x = y;

y = temp;

}

【A】该函数模板能够交换任何类型的对象

【B】该函数模板不能交换两个数组中的内容

【C】执行Swap(‘3’,’5’); 函数模板实例化为函数Swap<int>

【D】执行Swap(3, 4.5); 函数模板实例化为函数Swap<double>

1. 关于C++异常处理机制，以下描述错误的为（ ）。

【A】try、throw和catch语句用于实现异常处理

【B】throw语句可以抛出任何类型的异常对象

【C】检测异常的try语句和捕获异常的catch语句必须在同一函数中

【D】引发异常的throw语句和捕获异常的catch语句必须在同一函数中

1. 关于流的理解正确的为（ ）。

【A】C++中把数据之间的传输操作称作流，流中传输的是若干字节组成的字节序列

【B】输入流将数据从某个设备送到内存缓冲区，写磁盘文件文件为输入操作

【C】输出流将数据从内存传送到某个设备中，读磁盘文件文件为输出操作

【D】将U盘中的文件打开并通过投影仪播放为输出操作

1. istream类中不能实现字符串输入的成员函数为（ ）。

【A】get() 【B】getline() 【C】read() 【D】<<

1. 若派生类Dev公有的继承基类Base，则下述正确的为（ ）。

【A】Base\* pBase = new Dev();

【B】Base b; Dev\* pDev = &b;

【C】Base b; Dev d; d = b;

【D】Base b; Dev& d = b;

**二、写出下面程序的运行结果。（20分, 共4题）**

1. （5分）

#include<iostream>

#include<cstring>

using namespace std;

class Student

{

int age;

char \*name;

public:

Student ( ):age(18)

{

name = new char[10];

strcpy(name,"noName");

cout<<"default construct "<<endl;

}

Student (int a , char \*str)

{

name = new char[strlen(str)+1];

strcpy(name,str);

age= a>0 ? a: 0;

cout<<"construct "<<name<<endl;

}

~ Student ()

{

cout<<"destruct "<<name<<endl;

if(name )delete[] name;

}

};

int main()

{

Student \* pp = new Student (18,"Zhang");

Student p;

delete pp;

return 0;

}

1. （5分）

#include<iostream>

using namespace std;

class Student

{

int age;

char name[10];

public:

Student (int a =18, char \*str = "noName")

{

strcpy(name,str);

age= a>0 ? a: 0;

}

Student(Student& s):age(s.age)

{

strcpy(name,s.name);

cout<<"copy construct "<<endl;

}

~ Student ()

{

cout<<"destruct "<<name<<endl;

}

bool compare( Student s)const

{

return age>s.age;

}

const char\* getName()const

{

return name; }

};

int main()

{

Student p1, p2(22,"Linda");

if(p1.compare(p2))

cout<< p1.getName()<<" is older."<<endl;

else

cout<< p2.getName()<<" is older."<<endl;

return 0;

}

1. （5分）

#include <iostream>

#include <cstring>

using namespace std;

class Student

{

public:

Student(int = 1 , char\* ="学生") ;

Student(const Student& s) ;

virtual void print()const;

protected:

char name[10];

int grade;

};

Student::Student(int no , char\* n )

{

grade = no>0?no:0;

if(n) strcpy(name,n);

}

void Student::print()const

{

cout<<"\n学生姓名："<<name<<" 年级：" <<grade;

}

class CollegeStudent:public Student

{

public:

CollegeStudent(int = 1, char\* = "大学生", double score=0);

double fee() const ;

void print() const;

protected:

double gpa ;

};

CollegeStudent:: CollegeStudent(int g, char \* n, double s):Student(g, n)

{

gpa = s>0 ?s:0; }

double CollegeStudent::fee()const

{

if(grade==1||grade==2) return 6000;

if(grade==3||grade==4) return 12000;

return 10000;

}

void CollegeStudent::print() const

{

Student::print();

cout<<" 平均分 = "<< gpa<<" 学费"<< fee()<<endl;

}

int main()

{

Student\* pS [3];

pS[0]= new Student;

pS[1] = new CollegeStudent;

pS[2] = new CollegeStudent(3," 明明",3.5);

for(int i=0;i<3;i++)

pS[i]->print();

return 0;

}

4. （5分）

#include <iostream>

using namespace std;

template< typename T>

T Div(T a, T b)

{

if(b==0) throw b;

cout<<"End of Div"<<endl;

return a/b;

}

int main()

{

try

{ cout<<"1/2="<< Div(1,2)<<endl;

cout<<"6.0/0.0="<< Div(6.0,0.0)<<endl;

cout<<"End of try"<<endl;

}

catch(int)

{ cout<<"Error2: /0 "<<endl;

}

catch( double )

{ cout<<"Error1: /0.0 "<<endl;

}

cout<<"End of main"<<endl;

return 0;

}

**三、编程题（20分，共3题）**

1. （5分）已知如下商品类Goods的声明，请给出相关成员函数的定义，使之能够满足相关的功能。

class Goods

{

int id; //商品编号

char name[10]; //商品名称

double price; //商品价格

public:

Goods(int i=0, char\* n="NULL", double d=0);

void setPrice(double);

double getPrice()const{return price;}

const char\* getName()const{return name;}

void show() const;

};

// 定义商品类，成员函数实现

// ……

int main()

{

Goods dress(201522,"T-shirt",129);

dress.setPrice(99); // 特价99元

dress.show( ); //输出结果 商品编号:201522,商品名称：T-shirt,商品价格:99RMB

return 0;

}

2. （7分）利用上题中的商品类Goods，设计一个商品记录类Record，属性包括商品goods，购买商品实际消费金额cost，以及商品折扣discount（假设所有商品设置统一的折扣率为0.8），类声明形式如下，请给出该类的完整定义，并在main()函数中进行测试。

#include <iostream>

using namespace std;

class Goods;

class Record // 商品记录类

{

protected:

Goods goods; //商品

double cost; //实际消费，初值为0

static double discount; //折扣率，值为0.8

public:

Record(){ cost=0;}

Record( Goods& ); //构造函数

double pay( ); //计算实际消费金额

static double getCount(){return discount ;}

friend ostream& operator<<(ostream&, Record&);

};

// 设置折扣率discount

//成员函数定义

Record:: Record( Goods& g )

{

}

//计算返回实际消费金额 = 商品实际价格\*折扣率

double Record::pay( )

{

}

//输出记录信息：商品编号,商品名称,商品价格（RMB）以及消费金额（RMB）

ostream& operator<<(ostream& out, Record& r)

{

}

int main()

{

Goods food(1001,"Bread",2.5);

Goods dress(2012,"T-shirt",129);

Goods toy(3067,"HelloKity",45.9);

Record r[3] = { Record(food), Record(dress),Record(toy)};

//测试代码，补充完整，使其符合下面的执行结果

return 0;

}

输出结果为：

折扣率：0.8

商品编号:1001,商品名称：Bread,商品价格:2.5RMB 消费金额：2RMB

商品编号:2012,商品名称：T-shirt,商品价格:129RMB 消费金额：103.2RMB

商品编号:3067,商品名称：HelloKity,商品价格:45.9RMB 消费金额：36.72RMB

3. （8分）利用上题中的商品记录类Record，设计一个账单类Bill，账单中包括的账单号、多条消费记录、消费数量以及消费总额，给出相关成员函数的定义，使之能够满足相关的功能，通过main()函数中的测试代码。

#include<iostream>

#include<fstream>

#include<cstring>

using namespace std;

class Goods{……};

class Record{……};

const int MAX=20; //账单中最大记录数

//账单类

class Bill

{

string id; //账单号

Record records[MAX]; //消费记录

int size; //消费数量

double totalCost; //消费总额

public:

Bill(string i):id(i){ size=0;}

int getSize()const{return size;}

void addRecord(Record);

double getCost();

void printBill( ostream& );

};

//成员函数实现

//……

//测试代码

int main()

{

Goods food(1001,"Bread",2.5);

Goods dress(2012,"T-shirt",129);

Goods toy(3067,"HelloKity",45.9);

Record r[3] = { Record(food), Record(dress),Record(toy)};

Bill b("000101");

for(int i=0;i<3;i++)

b.addRecord(r[i]); //向账单中增加记录

b.printBill(cout); //将账单信息输出到屏幕上

ofstream out("file.txt");

b.printBill(out); //将账单信息输出到文件file中

return 0;

}

输出结果为：

账单号0001

No.1 商品编号:1001,商品名称：Bread,商品价格:2.5RMB 消费金额：2RMB

No.2 商品编号:2012,商品名称：T-shirt,商品价格:129RMB 消费金额：103.2RMB

No.3 商品编号:3067,商品名称：HelloKity,商品价格:45.9RMB 消费金额：36.72RMB

消费总额 = 141.92RMB

**四、阅读程序，回答问题（20分, 每题5分，共4题）**

1.（5分）设计堆栈类模板，在横线处给出相关代码

#include <iostream.h>

#include <process.h>

const int MAX = 10; //堆栈最大容量

//类模板声明

template<typename T>

class Stack

{ private:

T st[MAX];

int top; //栈顶

public:

Stack() { top = -1; }

void push( const T& number) ; //入栈

T pop(); //出栈

};

//类模板定义

template<typename T>

//入栈成员

{ if( ) // 判断栈已满

{ cerr<<"Stack is Full!"<<endl;

abort(); //结束程序

}

//入栈操作

top++;

}

//声明模板

T Stack<T>::pop( ) //出栈成员

{

if(top<0) // 判断栈为空

{ cerr<<"Stack is Empty!"<<endl;

abort(); //结束程序

}

return //返回栈顶元素

}

2. （5分）设计有理数类，在横线处给出相关代码

#include<iostream>

//标准命名空间

class Rational

{

public:

Rational(int nn=1,int mm=1);

void print() const;

Rational operator+(Rational & A); //加法

Rational operator-( );// 取反

private:

int m; // 分母

int n; // 分子

};

Rational::Rational(int nn, int mm):n(nn)

{

//设置分母

}

Rational Rational::operator+(Rational & A)

{

Rational C;

//计算分子

c.m= m\*A.m; //计算分母

return C;

}

Rational Rational::operator-()

{

return // 分子取反

}

//输出成员

{

if(m<0)

cout<< -n <<'/'<<-m<<endl;

else

cout<< n <<'/'<<m<<endl;

}

3．分析以下代码，找出其中的语法错误与逻辑错误并改正。

#include<iostream>

using namespace std;

class Point

{

public:

Point( int a = 0, int b = 0 ):x(a), y(b){}

void print() const

{ cout << '[' << x << "," << y << "]" ;}

private:

int x, y;

};

class Circle : public Point

{

public:

Circle(int xx=0, int yy=0, double r=1.0) ;

double area( ) const;

void print() const;

private:

double radius;

static double Pi;

};

Circle ::Circle(int xx=0, int yy=0, double r=1.0){

x = xx ; y=yy ;

r = radius;

}

double area( ) const

{ return Pi\* radius \* radius ;}

void Circle ::print()

{

cout << '[' << x << "," << y << "]" ;

cout << "r = "<< radius;

}

4. 分析以下代码中，找出并改正main()中的错误，使其能够实现多态并输出正确的结果。

#include <iostream>

#include <cstring>

using namespace std;

class Student

{

public:

Student(int = 1 , char\* ="学生") ;

Student(const Student& s) ;

virtual void print()const;

virtual double fee()const =0;

protected:

char name[10]; //姓名

int grade; //年级

};

Student::Student(int no , char\* n )

{

grade = no>0?no:0;

if(n) strcpy(name,n);

}

void Student::print()const

{

cout<<"学生姓名："<<name<<" 年级：" <<grade;

}

class CollegeStudent:public Student

{

public:

CollegeStudent(int = 1, char\* = "大学生", double score=0);

double fee() const ;

void print() const;

protected:

double gpa ;

};

CollegeStudent:: CollegeStudent(int g, char \* n, double s)

:Student(g, n),gpa(s){}

double CollegeStudent::fee()const

{

if(grade==1||grade==2) return 6000;

if(grade==3||grade==4) return 12000;

if(grade>4) return 10000;

}

void CollegeStudent::print() const

{

Student::print();

cout<<" 平均分 = "<< gpa;

}

class GraduateStudent : public Student

{

public:

GraduateStudent()

{ field = new char[10];

strcpy(field,"Software");

}

GraduateStudent(int no, char \* n ,char\* f)

: Student(no, n)

{ field = new char[strlen(f)+1];

strcpy(field, f);

}

~GraduateStudent()

{ delete[] field; }

void print() const

{

Student::print();

cout<<" 研究领域:"<<field;

}

double fee()const{ return 5000;}

private:

char \*field;

};

int main()

{

Student pS [3] = { CollegeStudent,

CollegeStudent(3,"敏敏",3.5),

GraduateStudent(2,"Gates","IT")};

for(int i=0;i<3;i++)

{

pS[i].print();

cout<<" 学费"<<pS[i].fee()<<endl;

}

return 0;

}

正确输出结果

学生姓名：大学生 年级：1 平均分 = 0 学费6000

学生姓名：敏敏 年级：3 平均分 = 3.5 学费12000

学生姓名：Gates 年级：2 研究领域:IT 学费5000

**五、面向对象程序设计（10分，共1题）**

利用**面向对象思想**为某银行设计一个用户管理的程序框架。用户（User）可以开设不同类型的账户（Account）， 账户类型包括储蓄账户（SavingAccount）与信用卡账户（CreditAccount），以后可能会增添新的账户分类（如基金账户，股票账户等）。每个账户有账号id与余额balance，信用卡账户还包括透支额度credit。不同账户都可以进行取款withdraw( )、存款deposit( )与计算余额的settle()操作, 但操作方式不同(例如：储蓄账户可以进行定期与活期存款，而信用卡可以根据信用额度进行透支)，但都可以用print()输出该账户的信息。

用户（User）的属性包括：姓名name与管理账户的数组account，每个用户最多开设10个账户，当前账户的个数为num。 用户可以按账号id查询某个账户的结余find( ), 可以计算所有账户的余额总和total()，也可显示show()所有账户的相关信息。

要求设计出所有的类，只需声明出类的主要属性和成员函数，给出类的声明（无需类的具体实现），注意合理设计类的关系及其虚函数。并给出User类中输出用户所有账户信息的show( )的定义。

课程名称：面向对象方法与C++程序设计 试 卷： A 是否开卷 否

授课院 (系)： 软件学院 考试日期：2016 年 5 月 8 日 试卷共14页

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总分 |
| 标准分 | 30 | 20 | 20 | 20 | 10 |  |  |  |  |  | 100 |
| 得 分 |  |  |  |  |  |  |  |  |  |  |  |

**一 、单项选择题（30分, 每题2分）**

1. 在类定义的外部，可以被访问的成员有（ ）

【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

1. 类clase C1的说明如下，错误的语句是（ ）

class C1{

int a; //(A)

void //(B)

public:

C1(int val); //(C)

~C1( ); //(D)

};

1. 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】析构函数可以有参数

【D】析构函数可以设置默认参数

1. 下面对友元的错误描述是（ ）

【A】 关键字friend用于声明友元

【B】 一个类的成员函数可以是另一个类的友元函数

【C】 友元函数访问类对象的成员不受访问特性影响

【D】 友元函数通过this指针访问对象成员

1. 以下关于this指针的叙述中正确的是（ ）

【A】任何与类相关的函数都有this指针

【B】类的成员函数都有this指针

【C】类的友员函数都有this指针

【D】类的非静态成员函数才有this指针

1. 下列函数中，不能重载运算符的函数是（ ）

【A】成员函数 【B】构造函数

【C】普通函数 【D】友员函数

1. 关于C++中的重载机制，以下描述正确为（ ）。

【A】可以对普通函数进行重载，也可以重载类的成员函数

【B】运算符只可以重载为成员函数

【C】类的构造函数与析构函数都可以被重载

【D】普通类和类模板都可以进行重载

1. 派生类采用（ ）方式可以使基类中的保护数据成员成为自己的私有数据成员。

【A】 私有继承　　　 【B】保护继承

【C】 公有继承　　　 【D】私有、保护、公有均可

1. 在如下继承层次下，实例化派生类Derived的对象时，调用构造函数的顺序为（ ）。

class Base{…};

class Base1: virtual Base{…};

class Base2: virtual Base{…};

class Derived : public Base2, public Base1{…};

【A】 Base(), Base1(), Base(), Base2(), Drived()

【B】 Base(), Base2(), Base(), Base1(), Drived()

【C】 Base(), Base1(), Base2(), Drived()

【D】 Base(), Base2(), Base1(), Drived()

1. 关于多重继承二义性的描述，错误的是（ ）。

【A】派生类的多个基类中存在同名成员时，派生类对这个成员访问可能出现二义性

【B】一个派生类是从具有共同的间接基类的两个直接基类派生来的，派生类对该公共

基类的访问可能出现二义性

【C】解决二义性最常用的方法是作用域运算符对成员进行限定

【D】派生类和它的基类中出现同名函数时，将可能出现二义性

1. 下面描述中，正确的是（ ）

【A】 虚函数是没有实现的函数

【B】 纯虚函数不必在派生类中实现

【C】 抽象类是没有纯虚函数的类

【D】 抽象类指针可以指向不同的派生类对象

1. 在C++中，容器是一种( )。

【A】标准类 【B】标准对象

【C】标准函数 【D】标准类模板

1. 有如下函数模板定义：

template <class T>T fun(T x,T y){return x\*x+y\*y;}

在下列对fun的调用中，错误的是（ ）

【A】fun(2,8) 【B】fun(2.0,8.2) 【C】fun(2.3,8) 【D】fun('2','8')

1. 关于异常描述不正确的是（ ）。

【A】c++的异常处理机制通过3个保留字try、catch、throw实现

【B】任何需要检测的语句必须放在try语句中，并trow语句抛出异常

【C】throw语句抛出异常后，catch语句利用数据类型匹配进行异常捕获

【D】一旦catch捕获异常后，不能将异常用throw语句再次抛出

1. istream类中不能实现输入字符串的成员函数为（ ）。

【A】get() 【B】getline()

【C】<< 【D】read()

**二、写出下面程序的运行结果。（20分, 每题5分，共4题）**

1. （5分）

#include <iostream>

using namespace std;

class CExample {

private:

int a;

public:

CExample(int b){

a=b;

cout<<"构造函数"<<endl;

}

CExample(const CExample& C){

a=C.a;

cout<<"拷贝构造函数"<<endl;

}

void Show ()

{

cout<<a<<endl;

}

CExample add(){

a++;

return \*this;

}

};

int main()

{

CExample A(100);

CExample B=A;

B.Show();

B=A.add();

B.Show();

return 0;

}

1. （5分）

#include<iostream>

using namespace std;

class Internet

{

public:

Internet(char \*name,char \*address) {

strcpy(Internet::name,name);

strcpy(Internet::address,address);

count++;

}

static void Internet::Sc() {

cout<<count<<endl;

}

Internet &Rq();

public:

char name[20];

char address[20];

static int count;

};

Internet& Internet::Rq(){

return \*this;

}

int Internet::count = 0;

void vist() {

Internet a1("abc","www.cndev-lab.com");

Internet a2("123","www.cndev-lab.com");

}

void fn(Internet s) {

cout<<s.Rq().count<<endl;

}

void main() {

cout<<Internet::count<<endl;

vist();

Internet::Sc();

Internet b("sina","www.cndev-lab.com");

Internet::Sc();

fn(b);

fn(b);

}

1. （5分）

#include<iostream>

using namespace std;

class BASE {

public :

void get (int I, int K ){

a=I;x=K;

}

void print(){cout<<"a="<<a<<'\t'<<"x="<<x<<endl;

}

int a ;

protected:

int x;

};

class A :public BASE{

public:

void get (int I, int k){

BASE obj3,obj4;

obj3.get(50,60);

obj3.print();

BASE::a=I;x=k;

a=BASE::a+obj3.a;

obj4=\*this;

obj4.print();

}

void print(){

cout<<"a="<<a<<endl;

BASE::print();

}

private:

int a;

};

void main(){

BASE Obj1;

A Obj2;

Obj1.get(10,20);

Obj2.get(30,40);

Obj1.print ();

Obj2.print ();

}

1. （5分）

#include<iostream>

using namespace std;

void XHandler(int test){

try {

if(test) throw test;

}

catch(int i){

cout << "Caught exception #: " << i << endl;

}

}

void main(void){

cout << "Start: " << endl;

XHandler(1);

XHandler(2);

XHandler(0);

XHandler(3);

cout << "End";

}

**三、程序填空（20分, 每题5分，共4题）**

1. 每个横线处填写一条语句，使得程序输出如下结果。

constructor

3 abc

copystructor

3 abc

destructor

destructor

#include <iostream>

#include <string>

using namespace std;

class student

{

public:

student(char \*na);

~student();

student(const student &);

void print();

private:

//记载字符个数

char \*name; //字符起始地址

};

void student::print()

{

cout<<num<<" "<<name<<endl;

}

student::student(char \*na)

{

num=strlen(na);

strcpy(name,na);

cout<<"constructor\n";

}

student::~student()

{

cout<<"destructor\n";

}

student::student(const student &s)

{

//深度复制，使得每个对象有各自的堆空间

num=s.num;

name=new char[num+1];

cout<<"copystructor\n";

}

void main()

{

student s1("abc");

s1.print();

s2.print();

}

1. 程序填空，使得输出为：1243

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

using namespace std;

class A{

private:

int X;

protected:

int Y;

public:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A(int a, int b, int c) {

X=a; Y=b; Z=c;

}

int \_\_\_\_\_\_\_\_\_\_\_\_\_{

return X;

}

};

class B :public A{

private:

int K;

public:

B(int a, int b, int c, int d):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_{

K=d;

}

void Show( ) {

cout << GetX() << Y << K;

}

};

void main( ){

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b.Show();

cout<<b.Z;

}

#include <iostream>

using namespace std;

template <class T>

class Array1D{

public:

Array1D(int size =0);

//深复制

Array1D(const Array1D<T>& v){

size = v.size;

element = new T[size];

for(int i = 0;i <size;i++)

element[i]=v.element[i];

}

~Array1D(){

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

element =0; }

T& operator[](int i)const;

int Size(){return size;}

private:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_//数组的大小

T\* element;//一维数组的起始地址

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Array1D<T>::Array1D(int sz ){

if(sz <0) {

cout << "argument error" << endl;

exit(0);

}

size = sz;

element = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

template <class T>

T& Array1D<T>::operator[](int i )const{

if(i<0 || i>=size) i = 0;

return element[i];

}

int main(){

Array1D<int> array1(10);//实例化成整型一维数组

int i;

for(i =0; i< 10;i++)

array1[i] = i;//调用[]函数

Array1D<int> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

for(i =0; i< 10;i++)

cout << array2[i] << ' '; //输出0 1 2 3 4 5 6 7 8 9

cout << endl;

return 0;

}

#include<iostream>

#include<fstream>

using namespace std;

class Student

{

public:

char id[10];

char name[10];

float score;

};

void main()

{

//键盘输入写到stud1.dat

Student s[5];

int i;

for(i=0;i<=4;i++)

{

cin>>s[i].id>>s[i].name>>s[i].score;

}

ofstream outf(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, ios::binary|ios::out) ;

outf.write(\_\_\_\_\_\_\_\_\_\_\_ \_\_\_, sizeof(Student)\*5 ) ; //写入五条记录

outf.close();

//读出每人成绩加10

ifstream inf( "stud1.dat",\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) ;

inf.\_\_\_\_\_\_\_\_\_ ( (char \*) s, sizeof(Student)\*5 ) ; //读出五条记录

for(i=0;i<=4;i++)

s[i].score+=10;

inf.close();

//再次写入文件stud1.dat

outf.\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( "stud1.dat",ios::binary|ios::out) ;

outf.write( (char \*) s, sizeof(Student)\*5 ) ;

outf.close();

}

**四、编程题（20分，共3题）**

1. （8分）设计一个线段基类（Line），包括的属性为长度，当创建无参数对象时要求用户输入长度，当创建有参数对象时无需用户输入长度。其派生的矩形类在产生对象时要求输入两个相邻边（其中一条是从线段继承来的）的长度。完成线段类和矩形的设计与实现，使得下面程序运行结果如下：

输入线段的长度：

3

输入另一条相邻边线段长度：

4

矩形，相邻边分别为：A: 3, B: 4

#include < iostream >

#include < cmath >

using namespace std;

class Line{ //线段基类

};

class Rectangle : public Line{ //矩形类

};

void main(){

Rectangle \*t = new Rectangle();

t->printSize();

}

1. （6分）完成下面类成员函数运算符重载+和-（取反）的声明与实现，使得运行结果为：

c1 + c2 = (-2, -6)

- c1 = (3, -4)

#include <iostream>

using namespace std ;

class Complex //定义Complex类

{

public:

Complex( ){dReal=0;dImag=0; } //默认构造函数

Complex(double r, double i){ dReal=r; dImag=i; } //重载构造函数

//重载运算符+

//重载运算符-

void print( )const;

private:

double dReal; //实部

double dImag; //虚部

};

// 重载加号

// 重载取反符号

//输出复数

void Complex::print( )const {

cout << '(' << dReal << ", " << dImag << ')' << endl;

}

// 测试函数

int main( ){

Complex c0,c1(-3,4),c2(1,-10);

c0 = c1 + c2;

cout << "c1 + c2 = ";

c0.print( );

c0 = -c1 ;

cout << "- c1 = ";

c0.print( );

return 0;

}

1. **（6分）**设计一个函数模板max，返回给定数组中数据的最大值，并用int型和double型数组在main函数中使用模板。

template < class T>

T max (T arr[], int n) //arr为数据数组，n为数组中数据个数。

{

}

void main()

{

}

1. **面向对象程序设计（10分）**

利用面向对象思想，描述一个班级（ClassRoom）学生运动的程序框架（类结构）。班级学生包括男生（Male）、女生（Female）。男生运动为俯卧撑，女生运动为仰卧起坐。

要求：

（1）设计出所有的类，只需要写出类的声明（只列出属性和方法）。

（2）必须有一个班级类，并且该类有一个sports方法，当运行班级类的sports方法时，输出所有学生运动的信息，对班级类的sports方法写出实现。