第一章

1. 什么是面向对象程序设计?它与传统的结构化程序设计有什么不同？

面向对象程序设计是软件系统设计与实现的新方法，这种方法是通过增加软件的可扩充性和可重用性来提高程序员的生产能力，控制软件的复杂性，降低软件维护的开销。

与传统的结构式程序设计相比，面向对象程序设计吸取了结构式程序设计的一切优点

（自顶向下、逐步求精的设计原则）。而二者之间的最大差别表现在：

（1）面向对象程序采用数据抽象和信息隐藏技术使组成类的数据和操作是不可分割的，避免了结构式程序由于数据和过程分离引起的弊病。

（2）面向对象程序是由类定义、对象（类实例）和对象之间的动态联系组成的。而结构式程序是由结构化的数据、过程的定义以及调用过程处理相应的数据组成的

1. 在面向对象程序设计中，什么是对象？什么是类？对象与类的关系是什么？

对象：在面向对象系统中，对象是系统中用来描述客观事物的一个实体，他是构成系统的一个基本单位。

类：类是对象的抽象及描述，是具有共同属性和操作的多个对象的相似特性的统一描述体。

对象与类的关系是什么：对象的共性抽象为类，类的实例化就是对象。

1. 现实世界中的对象有哪些特征？
2. 具有唯一标识名。可以区别于其他对象。
3. 具有一个状态，由与其相关联的属性值集合所表征。
4. 具有一组操作方法即服务，每个操作决定对象的一种行为。
5. 一个对象的成员仍可以是一个对象。
6. 模块独立性
7. 动态连接性
8. 易维护性
9. 什么是消息？消息具体有什么性质？

消息：消息是面向对象系统中实现对象间的的通信和请求任务的操作，是要求某个对象执行其中某个功能操作的规格说明

性质：

1. 同一个对象可接受不同形式的多个消息，产生不同的响应。
2. 相同形式的消息可以发给不同对象，所做出的响应可以是截然不同的。
3. 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以对消息不予理会，对消息的响应并不是必需的。
4. 什么是方法？在C++中它是通过什么来实现的？

方法：方法是对象所能执行的操作

在C++语言中方法是通过函数来实现的，成为成员函数。

1. 什么是抽象和封装？什么是继承？请举列说明。

抽象：抽象是一种从一般的观点看待事物的方法，他要求程序员集中于事物的本质特征，而不是具体细节或具体实现。 封装：所谓的数据封装是指一组数据和与这组数据有关的操作集合组装在一起，形成一个能动的实体，也就是对象。

例：比如 一个封装好的类 我们只要知道他能做什么 而不用去考虑他的内部是如何做的 比如 一辆汽车 他做为一个实体 我们 可以将他抽象成一个类 而这个汽车类 又是红色跑车这个对象的抽象 同时 我们红色跑车是封装好的 你只要会开车 他就能跑起来 而你 可以忽略它的内部原理 即 如何跑起来的

继承：从已有的对象类型出发建立一种新的对象类型，使它继承原有对象的特点和功能。

例：继承分为单继承和多继承，子类对单个直接父类的继承称为单继承，子类对于多余一个直接父类的继承称为多继承。父类也称为基类或超类，子类也称为派生类。

1. 传统程序设计方法的局限性主要有哪些？
2. 用户要求难以在系统分析阶段准确定义，致使系统在交付使用时产生许多问题。
3. 用系统开发每个阶段的成果来进行控制，不能适应书无变化的要求。
4. 系统的开发周期长。

1. 面向对象程序设计的优点主要有那些？
2. 易维护  
   采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是在局部模块，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，在以前的项目的领域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  
   在软件开发时，根据设计的需要对现实世界的事物进行抽象，产生类。使用这样的方法解决问题，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、多态的特性，自然设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

第二章

1．分析下列程序的执行结果：

输出随机数

2. 分析下列程序的执行结果：

i=0

3. C++语言对C语言在结构化程序设计方面进行了哪些扩充？ 主要在以下方面进行了扩充：

文件扩展名、注释符、名字空间、输入输出、变量的定义、强制类型转换、动态内存的分配与释放、作用域运算符::、引用、const修饰符、字符串、函数

4. 下述C++程序有若干处错误，试找出并纠正之。

正确程序为：

#include

const float PAI=3.14159265；

float square(float r)

{return PAI\*r\*r;}

float square(float high,float length=0 )

{return high\*length;}

float (\*fs)(float,float=0);

void main()

{

fs=&square;

cout<<"The circle's square is "<<fs(1.0)<<'\n'; }

5. 引用类型与指针类型有什么区别？

指针的内容或值是某一变量的内存单元地址，而引用则与初始化它的变量具有相同的内存单元地址。指针是个变量，可以把它再赋值成其它的地址，然而，建立引用时必须进行初始化并且决不会再指向其它不同的变量。

C++没有提供访问引用本身地址的方法，因为它与指针或其它变量的地址不同，它没有任何意义。引用总是作为变量的别名使用，引用的地址也就是变量的地址。引用一旦初始化，就不会与初始化它的变量分开。

6．函数、内联函数以及宏的区别。

程序的模块在C++中通过函数来实现，函数由函数说明和函数体2部分组成。 内联函数是C++语言特有的一种函数附加类别，是通过在函数声明之前插入“inline”关键字实现的。编译器会将编译后的全部内联函数的目的机器代码复制到程序内所有的引用位置并把往返传送的数据也都溶合进引用位置的计算当中，用来避免函数调用机制所带来的开销，提高程序的执行效率。显然这是以增加程序代码空间为代价换来的。

宏定义是编译预处理命令，分为带参数的和不带参数的宏定义。在编译之前进行预处理时，用宏定义中的字符串替换程序中所有出现的宏名。

7. 函数重载有什么好处？

函数重载使函数方便使用，便于记忆，也使程序设计更加灵活，增加程序的可读性。例如，求两个数中最大值的函数max，不管其参数是整数类型、实数类型、字符串，都可以使用同名函数来实现，调用时只需使用max就可以了，编译器将根据实参的类型判断应该调用哪一个函数。

第三章

1． 为什么要引入构造函数和析构函数？

对象的初始化是指对象数据成员的初始化，在使用对象前，一定要初始化。由于数据成员一般为私有的(private)，所以不能直接赋值。

析构函数的功能是当对象被撤消时，释放该对象占用的内存空间。析构函数的作用与构造函数正好相反，一般情况下，析构函数执行构造函数的逆操作。在对象消亡时，系统将自动调用析构函数，执行一些在对象撤消前必须执行的清理任务。

2. 类的公有、私有和保护成员之间的区别是什么？

① 私有成员private: 私有成员是在类中被隐藏的部分，它往往是用来描述该类对象属性的一些数据成员，私有成员只能由本类的成员函数或某些特殊说明的函数访问，而类的外部根本就无法访问，实现了访问权限的有效控制.

② 公有成员public：公有成员对外是完全开放的，公有成员一般是成员函数，它提供了外部程序与类的接口功能，用户通过公有成员访问该类对象中的数据。

③ 保护成员protected: 只能由该类的成员函数，友元，公有派生类成员函数访问的成员。保护成员与私有成员在一般情况下含义相同，它们的区别体现在类的继承中对产生的新类的影响不同。

3. 什么是拷贝构造函数，它何时被调用？

拷贝构造函数的功能是用一个已有的对象来初始化一个被创建的同类对象，是一种特殊的构造函数，具有一般构造函数的所有特性，当创建一个新对象时系统自动调用它；其形参是本类对象的引用，它的特殊功能是将参数代表的对象逐域拷贝到新创建的对象中。

在以下四种情况下系统会自动调用拷贝构造函数：

① 用类的一个对象去初始化另一个对象

cat cat1;

cat cat2(cat1); // 创建 cat2时系统自动调用拷贝构造函数，

② 用类的一个对象去初始化另一个对象时的另外一种形式

Cat cat2=cat1;

③ 对象作为函数参数传递时，调用拷贝构造函数。

f(cat a){ }

cat b;

④ 如果函数的返回值是类的对象，函数调用返回时，调用拷贝构造函数。

4. 设计一个计数器类，当建立该类的对象时其初始状态为0，考虑为计数器定义哪些成员？

// counter.h

#ifndef counter\_h

#define counter\_h

class counter{

private:

int count;

public:

counter(); void setCount(int i); int getCount(); void displayCount(); void incrementCount(); void decrementCount(); ~counter(){}

#endif

// counter.cpp

#include "counter.h"

#include

counter::counter(){

} count = 0;

void counter::displayCount(){

}

cout << count << endl;

int counter::getCount(){

return count;

}

void coun

count = i;

}

void counter::incrementCount(){

count++;

}

void counter::decrementCount(){

}

// MAIN.CPP

#include "counter.h"

#include

void main(){

counter c1; count--;

}

c1.setCount(4); c1.displayCount(); for(int i=0; i<=10;i++){ c1.incrementCount(); c1.displayCount(); }

5. 定义一个时间类，能提供和设置由时、分、秒组成的时间，并编写出应用程序，定义时间对象，设置时间，输出该对象提供的时间。

#include <iostream>

class Time{

int hour,minute,second;

public:

,

Time(int h,int m int s )

{hour = h;

minute = m;

second = s;}

void display()

{ cout << hour << ": " << minute << ": " << second }

};

void main()

{

Time t(12,12,2); t.display();

}

6．设计一个学生类student，它具有的私有数据成员是：注册号、姓名、数学、英语、计算机成绩；具有的公有成员函数是：求三门课总成绩的函数sum；求三门课平均成绩的函数average；显示学生数据信息的函数print；获取学生注册号的函数get\_reg\_num;设置学生数据信息的函数set\_stu\_inf。

编制主函数，说明一个student类对象的数组并进行全班学生信息的输入与设置，而后求出每一学生的总成绩、平均成绩、全班学生总成绩最高分、全班学生总平均分，并在输入一个注册号后，输出该学生有关的全部数据信息。

#include

#include

using name

class Student{

private:

int num;

char name[10];

float math;

float english;

float computer;

public:

void set\_stu\_inf(int n,char \*ch,float m,float e,float c) {

num=n; strcpy(name,ch); math=m; english=e; computer=c;

}

float sum()

{ } float average() { } int get\_reg\_num() { return num; return (math+english+computer)/3; return (math+english+computer);

};

} void print() { cout<<"学号："<

void main()

{

Student stu[50];

int i,q,a,z,x,max=0,aver=0; //i为循环变量，q：学号；a：数学

//成绩； z：英语成绩； x：计算机成绩

int count = 0; //表示学生人数

char\*we=new char[10];

// 输入学生信息 for(;;) {

cout<<"请输入学生的学号、姓名、数学成绩、英语成绩、计算机成绩：(若输入的学号为0则表示退出)" << endl;

} aver+=(a+z+x); stu[count++].set\_stu\_inf(q,we,a,z,x); if(max>a+z+x); else max=a+z+x; cin>>q>>we>>a>>z>>x; if (q ==0 ) break;

// 输出所有学生信息

cout<<"学生信息为："<

for( i = 0; i < count; i++){

stu[i].print();

cout<

}

cout<<"全班学生总成绩最高分为"<

<<"全班学生总平均分为"<

cout<<"请输入要查的学生的学号:"<

cin>>q;

for( i = 0; i < count; i++){

} if (q==stu[i].get\_reg\_num()) { } cout<<"此学生信息为："<

if (i==count)

cout<<"查无此人"<

}

7. 模拟栈模型的操作，考虑顺序栈和链栈两种形式。

链栈:

#include

class Stack //定义堆栈类

{

struct Node

{

int content;

Node \*next;

} \*top;

public:

Stack() { top = NULL; } // 构造函数的定义

bool push(int i); // 压栈成员函数的声明

bool pop(int& i); // 弹栈成员函数的声明

};

bool Stack::push(int i) // 压栈成员函数的定义

{

Node \*p=new Node;

if (p == NULL)

{

cout << "Stack is overflow.\n";

return false;

}

else

{

p->content = i;

p->next = top;

top = p;

return true;

}

}

bool Stack::pop(int& i) // 弹栈成员函数的定义 {

if (top == NULL)

{

cout << "Stack is empty.\n";

return false;

}

else

{

Node \*p=top;

top = top->next;

i= p->content;

delete p;

return true;

}

}

void main()

{

Stack st1,st2; // 定义对象st1和st2 int x;

for(int i=1;i<=5;i++)

{

st1.push(i); // 压栈成员函数的调用 st2.push(i); // 压栈成员函数的调用 }

cout<<"stack1:"<

for(i=1;i<=3;i++)

{

st1.pop(x); // 弹栈成员函数的调用 cout<

}

st1.push(20);

for(i=1;i<=4;i++)

{

if(st1.pop(x))

cout<

else

break;

}

cout<<"stack2:"<

while(st2.pop(x))

cout<

}

顺序栈采用一维数组来实现。（略）

8. 写出下列程序的运行结果：

Constructing 22 11

Constructing 20 10

display:22 11

display:20 10

Destructing20 10

Destructing22 11

第四章

1.友元的作用是什么？

友元提供了在不同类的成员函数之间、类的成员函数与一般函数之间进行数据共享的机制，

通过友元，一个普通函数或另一个类中的成员函数可以访问类中的私有成员和保护成员。

2.友元概念的引入虽然方便了类之间的数据共享，但是否削弱了对象的封装性？

C++中的友元为封装隐藏这堵不透明的墙开了一个小孔，外界可以通过这个小孔窥视内部的秘密。

友元的正确使用能提高程序的运行效率，但破坏了类的封装性和数据的隐蔽性，导致程序可能护性变差，

因此一定要谨慎使用。

1. 友元的作用是？  
   友元提供了在不同类的成员函数之间，类的成员函数与一般函数之间进行数据共享的机制。（一个普通函数或者另一个类中的成员函数可以访问类中的私有成员保护成员）  
     
   4.友元的三种实现形式是？  
   1.是类本身的友元函数  
   2.A类中的某个函数是B类的友元函数  
   3.声明A类为B类的友元类，那么A中所有的函数都可以调用B类中的成员

5.设计并测试个矩形类(Retangle), 属性为矩形的左下与右上角的坐标，矩形水平放置。操作为计算矩形周长与面积。类对象的数据由友元函数来进行访问，并对比友元函数.成员函数和普通函数使用上的不同。

解:这里的矩形的4边分别与x轴y轴平行,为最简单的情况。注意参数有缺省值的函数的声明和定义格式。

#include < iostream: >

#include < cmath>

using namespace std;

class Rectangle {

double left, top;

double right, bottom;

public:

Rectangle(double I=0, double t=0, double r=0, double b=0);

~ Rectangle(){}; //析构函数，在此函数体为空

void Assign(double l,double t,double r,double b);

double getLeft(){ return left;} //以下四个函数皆为内联成员函数double getRight(){ return right:}double getTop0{return top;}

double getBottom({return bottom;}

void Show();

double Area();

double Perimeter();

};

//构造函数，带缺省参数，缺省值为全0，在声明中指定

Rectangle:Rectangle(double I, double t, double r, double b) {

left= l;top= t;

right= r; bottom= b;

}

voidRetcange:Assis(double l, double t, double r, double b){//赋值

left= l;top= t;

right = r; bottom = b;

}

void Rectangle::Show(){//成员函数直接使用私有的数据成员

cout<<"left-top pointis("<<left<<","<<top<<")"<<'\n';

cout<<"right-bottom point is (" <<right<<"," <<bottom<<")"<<'\n';

}

double Rectangle::Area(){

return fabs((right-left)\*(bottom-top));

}

double Rectangle::Perimeter(){

return 2\*(fabs(right-left)+fabs(bottom-top));

}

int main(){

Rectangle rect;

rect.Show();

rect.Assign(100,200,300,400);

rect.Show();

Rectangle rect1(0,0,200,200);

rect1.Show();

Rectangle rect2(rect1);

rect2.Show();

cout<<"面积"<<rect.Area()<<'\t'<<"周长"<<rect.Perimeter()<<endl;

return 0;

}

第五章

1. 什么是类的继承与派生？

继承性是面向对象程序设计的第二个重要特性，通过继承实现了数据抽象基础

上的代码重用。继承是对许多问题中分层特性的一种自然描述，因而也是类的具体

和被重新利用的一种手段，它所表达的就是一种对象类之间的相交关系。它使得某类

对象可以继承另外一类对象的特征和能力。继承所具的作用有两个方面：一方面可以减少

代码冗余；另一方面可以通过协调来减少相互之间的接口和界面。通过继承方式定义的

子类也称为派生类。

2.类的三种继承方式之间的区别是什么？

类的继承方式有 public(公有)继承、 protected(保护)继承和 private(私有)

继承三种。对于不同的继承方式,会导致基类成员原来的访问属性在派生类中有所变化

(1)基类的私有成员在派生类中均是不可访问的,它只能由基类的成员访问。

2)在公有继承方式下,基类中的公有成员和保护成员在派生类中的访问属性不变。

(3)在保护继承方式下,基类中的公有成员和保护成员在派生类中均为保护的

(4)在私有继承方式下,基类中的公有成员和保护成员在派生类中均为私有的

需要注意的是

保护成员与私有成员唯一的不同是当发生派生后,处在基类 protected区的成员可被

派生类直接访问,而私有成员在派生类中是不可访问的。在同一类中私有成员和保护成员

的用法完全一样

3.派生类能否直接访问基类的私有成员?若否,应如何实现?

派生类不能直接访问基类的私有成员。

具体实现方式(1)在类定义体中增加保护段

为了便于派生类的访问,可以将基类私有成员中需提供给派生类访问的部分定义为保护段成员。保护段成员可以被它的派生类访问,但是对于外界是隐藏起来的。这样,既方便了派生类的访问,又禁止外界对它的派生类访问。

这种方式的缺点是在公有派生的情况下,如果把成员设为保护访问控制,则为外界访

问基类的保护段成员提供了机会,而三种派生方式,我们经常使用的是公有派生。

(2)将需访问基类私有成员的派生类成员函数声明为基类的友元

这样派生类中的其它成员函数均无权访问它,外界不可能通过派生新类来达到访问基

类私有成员的目的。

4.派生类构造函数和析构函数的执行顺序是怎样的?在多继承中,派生类

构造函数和析构函数的执行顺序又是怎样的?

构造函数的执行顺序:先祖先(基类)、再客人(对象成员),后自己(派生类本身)

析构函数的执行顺序和构造函数正好严格相反:先自己(派生类本身),再客人(对象成

),后祖先(基类)。

在多个基类之间严格按照派生类定义时从左到右的顺序来排列先后。而析构函数的调

用顺序刚好与构造函数的相反。

5.派生类的构造函数和析构函数的作用是什么?

在下面两种情况下,必须定义派生类的构造函数:派生类本身需要构造函数:在定义

生类对象时,其相应的基类对象需调用带有参数的构造函数。

派生类对象的初始化也是通过派生类的构造函数实现的。具体来说,就是对该类的数据成

员赋初值

派生类析构函数的功能与没有继承关系的类中析构函数的功能一样,也是在对象消亡

之前进行一些必要的清理工作。

6.多继承一般应用在哪些场合?

对于多重继承,派生类可以有多个直接基类。这时的派生类同时得到了多个己有类的

特征。

7.在类的派生中为何引入虚基类?在含有虚基类的派生类中,当创建它的

对象时,构造函数的执行顺序如何?

当在多条继承路径上有一个公共的基类,在这些路径中的某几条路径汇合处,这个公

共的基类就会产生多个实例(或多个副本),若想只保存这个基类的一个实例,可以将这个

公共基类说明为虚基类。从基类派生新类时,使用关键字 virtual可以将基类说明成虚基

在多个基类之间严格按照派生类定义时从左到右的顺序来排列先后。而析构函数的调

用顺序刚好与构造函数的相反。如果基类中有虚基类,则构造函数的调用顺序采用下列规

(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,这些虛基类的构造函数按照他们说明的次序调用

(3)若虚基类由非虚基类派生而来,则仍然先调用基类构造函数,再调用派生类的构

造函数

特别需要注意,当一个派生类同时有多个基类时,对于所有需要给予参数进行初始化

的基类,都要显式给出基类名和参数表。对于使用默认构造函数的基类,可以不给出类名

同样,对于对象成员,如果是使用默认构造函数,也不需要写出对象名和参数表。而对于

单继承,就只需要写一个基类名就可以了

8.  
设计一个大学的类系统，学校中有学生、教师，每种人员都有自己的特性，他们之间有相同的地方。利用继承机制定义这个

系统中的各个类及类上必须操作。

#include

using namespace std;

class Person

{

protected:

string name;

long num;

public:

virtual void setin()=0;

virtual void setout()=0;

};

class Teacher:public Person

{

string title;

float salary;

public:

void setin()

{

cout<<"输入老师的信息:"<

cin>>name>>num>>title>>salary;

}

void setout()

{

cout<<"教师姓名:"<

cout<<"教师编号:"<

cout<<"教师职称:"<

}; class Student:public Person

{

string major;

float averscore;

public:

void setin()

{

cout<<"输入学生的信息:"<

cin>>name>>num>>major>>averscore; }

void setout()

{

cout<<"学生姓名:"<

cout<<"专业:"<

绩:"<

}

};

int main()

{

Teacher teacher1;

Student student1;

teacher1.setin();

student1.setin();

teacher1.setout();

student1.setout();

system("PAUSE");

retur  
  
9.假定车可以分为货车和客车，客车又可以分为轿车，面包车，和公共汽车，请设计相应的类的层次结构  
#include  
using namespace std;  
class vehicle // 定义基类vehicle  
{  
public: // 公有函数成员  
vehicle(int in\_wheels,float in\_weight); // 给数据成员初始化  
int get\_wheels(); // 获取车轮数  
float get\_weight(); // 获取汽车重量  
void setWeels(int wls);  
void setWeight(float wt);  
void display(){  
cout<<"车轮数:"<  
<<"汽车重量:"<  
}  
private: // 私有数据成员  
int wheels; // 车轮数  
float weight; // 表示汽车承重  
};  
vehicle::vehicle(int in\_wheels,float in\_weight){  
wheels = in\_wheels;  
weight = in\_weight;  
}  
float vehicle::get\_weight(){  
return weight;  
}  
int vehicle::get\_wheels(){  
return wheels;  
}  
void vehicle::setWeels(int wls){  
wheels = wls;  
}

10.设计一个能细分为矩形、三角形、圆形和椭圆形的“图形”类。使用继承将这些图形分类，找出能作为基类部分的共同特征(如宽、高、中心点等)和方法(如初始化、求面积等),并看看这些图形是否能进一步划分为子类。

#include< i ostream>

using namespace std;

class Figure //定义基类图形类

{

public: //公有函数成员

Figure(int wid){width = wid;}

float area(){ }

int getWidthO{retumn width;}

private: //私有数据成员

int width; //宽度或半径

};  
class Rectangle:public Figure{ //定义矩形类  
 int height;

public:  
 Rectangle(int wid, int hei):Figure(wid){

height = hei;

}  
float area(){return getWidth() \* height;}

};  
  
class Triangle:public Figure{ //定义三角形类  
 int height;

public:  
  
 Triangle(int wid, int hei):Figure(wid){

height = hei;

}

float area(){return 1.0/2\* getWidthO()\* height;}

};

class Circle:public Figure{//定义圆类

public:

Circle(int wid):Figure(wid){

}

float area(){return 3.14 \* getWidth() \* getWidth();}

};

void main(){

Rectangle rect(5 ,4);

Triangle tri(5,4);

Circle cir(5);

cout<<"矩形的面积是 "<<rect.area<< endl

<<"三角形的面积是: "<<tri.area()<< endl

<<"圆的面积是: "<cir.area()<<endl;

}

**程序的运行结果为:**

矩形的面积是: 20

三角形的面积是: 10

圆的面积是: 78.5

11.考虑大学的学生情况,试利用单继承来实现学生和毕业生两个类,设计

相关的数据成员及函数,编程对继承情况进行测试,

参考程序:

#include<iostream.h>

#include<string.h)

class Student

{

public:

Student (int n,char\*na. int g): number (n), grade(g){

strcpy(name, na);

}

int getNumber(){

return number;

}

char \* getName(){

return name;

}

int getGrade(){

return grade;

}

void display(){

cout<<"学号:"<<number<<endl

<<"姓名:"<<name<<end

<<"年级:"<<grade<<endl;

}

private:

int number;

char name[20];

int grade;

}

class Graduate:public Student{

char designSubject[20];

public:

Graduate(int n,char \*na,char \* deSub,int g=4):Student(n,na,g)

{strcpy(designSubject,deSub);}

void display(){

Student::display();

cout<<"设计题目:"<<designSubject<<endl;

}

};

void main(){

Student li(2,"Li Ming",3);

Graduate zhang(3,"Zhang Gang","学生成绩管理系统")；

Li.display();

cout<<endl<<"---------------------------------"<<endl;

zhang.display();

}

程序的运行结果：

学号：2

姓名：Li Ming

年级：3

-------------------------

学号：3

姓名：Zhang Gang

年级：4

设计题目：学生成绩管理系统

12.定义一个哺乳动物类,再由此派生出人类、狗类和猫类,这些类中均

有 speak()函数,观察在调用过程中,到底使用了谁的 speak()函数。

参考程序:

#include<iostream.h>

#include<string.h>

class Animal{

float weight;

public:

void speak(){}

void setWeight (float wt){weight = wt;}

float getWeight() {return weight;}

};

class Human: public Animal{

public:

void speak(){

cout<<"说话”<<endl;

}

};

class Cat:public Animal{

pubilc:

void speak(){

cout<<"喵喵"<<endl;

}

};

class Dog: public Animal{

public:

void speak(){

cout<<"汪汪"<<endl;

}

};

void main(){

Human hm;

Cat cat;

Dog dog;

cout<<"人:";

hm.speak();

cout<<"猫:";

cat.speak();

cout<<"狗:";

dog.speak();

}

程序的运行结果:

人:说话

猫:喵喵

狗:汪汪

13.通过多重继承定义研究生类,研究生既有学生的属性,又有教师的属性。

参考程序:

#include <iostream.h>

#include <string.h>

class Persor{

protected:

char m\_strName[10];

int m\_nSex;

int m\_nAge;

public:

Person(char \*name, int age, char sex){

strcpy(m\_strName, name);

m\_nSex=(sex=='m'?0:1);

m\_nAge= age；

}

void setName(char \*name){

strcpy(m\_strName, name);

}

void setSex(int sex){

m\_nSex=(sex=='m'?0:1);

}

void stAge(int age){

m\_nAge= age；

}

char\* getName(){

return m\_strName;

}

int getAge(){

return m\_nAge;

}

int getSex(){

return m\_nSex;

}

void ShowMe(){

cout<<”姓名:”<<m\_strName<<endl;

cout<<”性别:”<<(m\_nSex==0?””:””)<<endl;

cout<<”年龄:”<<m\_nAge<<endl;

}

};

classTeacher:public Person{

Char m\_strDept[20];

Int m\_fSalary;

public:

　　　Teacher(char\*name,int age,charl sex,chear\*dept,int salary)

　　　:person(name,agel,sex){

　　　Strcpy(m\_strDept,dept);

　　　M\_fSalary = salary;

　　　}

　　　void ShowMe(){

　　　Person::ShowMe();

　　　cout<<”工作单位:”<<m\_strDept<<endl;

　　　cout<<”月薪:”<<m\_fSalary<<endl;

　　　}

　　　void setSalary(int salary){

　　　m\_fSalary=salary;

　　　}

　　　char\*getDept(){

　　　return m\_strDept;

　　　}

　　　int getSalary(){

　　　Return M\_fSalary;

　　　}

};

Class Student:pulaic Person{

　　　char m\_srtID[12];

　　　char m\_srtClass[12];

Public:

　　　Student(char\*name,int age,char sex,char\*ID,char\*Class)

　　　:person(name,age,sex){

　　　Strcpy(m\_strID,ID);

　　　Strcpy(m\_strIClass,Class);

　　　}

Void ShowMe(){

　　　Cout<<”学号:”<<m\_strID<<endl;

　　　Person::ShowMe();

　　　Cout<<”班级:”<<m\_strIClass<<”\n”;

}

Void setID(char\*ID){

　　　Strcpy(m\_strID,ID));

}

Void setID(char\*ID){

　　　Strcpy(m\_strIClass,Class);

}

Char\* getID(){

　　　Return m\_strID;

}

Char\* getID(){

　　　Return m\_strIClass;

}

};

Class Master:puvlic Srudent,public Teacher{

Pucblic:

　　　Master(char \*name,int age,char sex,char \*ID,char \*Class,char \*dept,int salary)

　　　:Student(name,age,sex,ID,Class),Teacher(name,lage,sex,dept,salary){

　　　}

Void ShowMe(){

　　　Student::ShowMe();

　　　Cout<<”工作单位：”<<getDept()<<endl;

　　　Cout<<”月薪：”<<getSalary()<<endl;

　　　}

};

Void main(){

　　　Teachet teachet1(“刘馨”,38,’m’,”计算机系”,3800);

　　　Student std1(“刘丽”,20,’f’,”030106003”,”计算机03”);

　　　Master master(“张鑫”,25,’f’,”003”,”机械052”,”机械系”,1000);

　　　Teacher1.ShowMe();

　　　Cout<<”........................”<<endl;

　　　Std1.ShowMe();

　　　Cout<<”........................”<<endl;

　　　Master.ShowMe();

}

程序的运行结果为：

姓名：刘馨

性别：男

年龄：38

工作单位：计算机系

月薪：3800

....................

学号：030106003

姓名：刘丽

性别：女

年龄：20

班级：计算机03

....................

学号：003

姓名：张鑫

性别：女

年龄：25

班级：机械052

工作单位：机械系

月薪：1000