C++程序设计

主讲：王老师

尚德机构

学习是一种信仰

教材介绍

C++程序设计

（2008年版）

编著：刘振安

机械工业出版社

考试题型

单选题　　1分×20题 = 20分

填空题　　1分×20题 = 20分

程序改错题　4分×5题 = 20分

完成程序题　4分×5题 = 20分

程序分析题　5分×2题 = 10分

程序设计题 10分×1题 = 10分

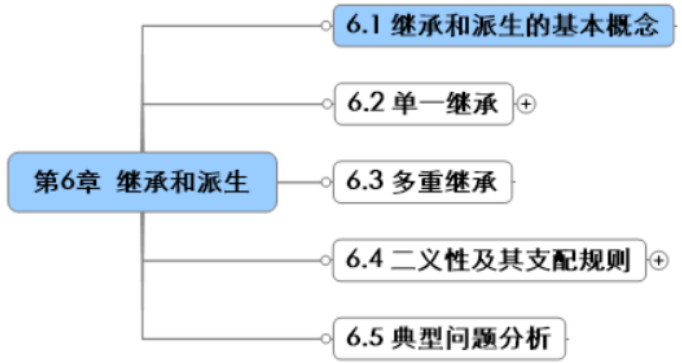
第六章　继承和派生

目录

第6章

继承和派生

§6.1 继承和派生的概念

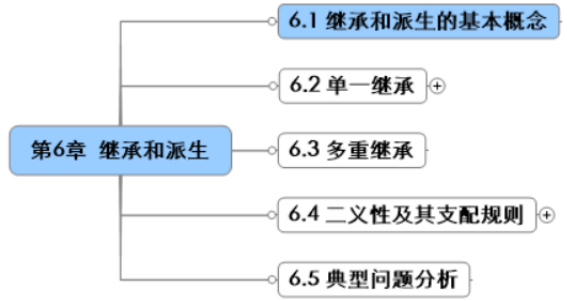


一、派生

类的派生是指从一个或多个以前定义的类产生新类的过程，原有的类称为基类，新产生的类称为派生类，**派生类继承了基类所有的数据成员和成员函数。**

派生类使用两种基本的面向对象技术：

- 一种称为性质约束，即对基类的性质进行限制；
- 另一种称为性质扩展，即增加派生类的性质。



一、派生

1、从一个或多个以前定义的类(基类)产生新类的过程称为派生，这个新类称为派生类。派生的新类同时也可以增加或者重新定义数据和操作，这就产生了类的层次性。

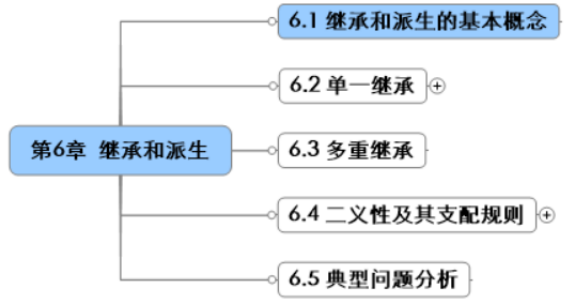
2、类的继承是指派生类继承基类的数据成员和成员函数。继承常用来表示类属关系，不能将继承理解为构成关系。

3、当从现有类中派生出新类时，派生类可以有如下几种变化：

（1）增加新的成员（数据成员或成员函数）。

（2）重新定义已有的成员函数。

（3）改变基类成员的访问权限。



二、继承

类的继承是指派生类继承基类的数据成员和成员函数为自己的成员，继承常用来表示类属关系，而不是构成关系。

C++中有两种继承方式：单一继承和多重继承，对于单一继承，派生类只能有一个基类；对于多重继承，派生类可以有多个基类。

静态成员可以被继承，这时基类对象和派生类的对象共享该静态成员。

§6.2 单一继承

一、单一继承的一般声明形式

```
class 派生类名:访问控制 基类名{
private:
    成员声明列表
protected:
    成员声明列表
public:
    成员声明列表
};
```

其中，访问控制是指如何控制基类成员在派生类中的访问属性，它是关键字private、protected和public中的一个，声明中的其余部分和类的声明类似。

二、派生类的构造函数和析构函数

1、定义派生类构造函数的一般定义形式：

派生类名::派生类名（参数表0）:基类名（参数表）{..... //函数体}

冒号后“基类名（参数表）”称为成员初始化列表，参数表给出所调用的基类构造函数所需要的实参，实参的值可来自“参数表0”，也可由表达式给出。

2、派生类析构函数的一般定义形式：

派生类名::~派生类名（）{...../函数体}

构造函数和析构函数也都可在类体内直接定义为内联函数，这时的定义形式需把上述定义式中的“派生类名::”去掉。

二、派生类的构造函数和析构函数

例1：使用默认内联函数实现单一继承

程序运行结果：

点 矩形 x=3,y=4 H=5,W=6 删除矩形 删除点

```
#include <iostream>
using namespace std;
class Point{
private:
        int x,y;
public:
        Point(int a,int b){x=a;y=b;cout<<"点"<<" ";}
        void Showxy( ){cout<<"x="<<x<<",y="<<y<<" ";}
        ~Point( ){cout<<"删除点"<<" ";}
};
class Rectangle:public Point{
private:
        int H,W;
public:
        Rectangle(int a,int b,int h,int w):Point(a,b){H=h;W=w;cout<<"矩形"<<" ";} //构造函数初始化列表
        void Show( ){cout<<"H="<<H<<",W="<<W<<" ";}
        ~Rectangle( ){cout<<"删除矩形"<<" ";}
};
void main(){
        Rectangle r1(3,4,5,6);    //生成派生类对象r1，r1先后调用基类和派生类的构造函数进行初始化
        r1.Showxy( );    //派生对象调用基类的成员函数
        r1.Show( );    //派生对象调用派生类的成员函数
}
```

点 矩形 x=3,y=4 H=5,W=6 删除矩形 删除点

3、小结

由上例可看出，构造函数和析构函数是不被继承的，所以一个派生类只能调用它的直接基类的构造函数。

当定义派生类的一个对象时，首先调用基类的构造函数，对基类成员进行初始化，然后执行派生类的构造函数，如果某个基类仍是一个派生类，则这个过程递归进行。当该对象消失时，析构函数的执行顺序和执行构造函数时的顺序正好相反。

三、类的保护成员

类的保护成员是指在类声明中以关键字protected声明的成员，保护成员具有私有成员和公有成员的双重角色：对派生类的成员函数而言，它是公有成员，可直接访问；而对其他函数而言，它是私有成员，不能直接访问，只能通过基类的对象访问。这样就解决了使用公有方式产生的派生类的成员函数虽然可直接访问基类中定义的或从另一个基类继承来的公有成员，但不能访问基类的私有成员的问题。

例2：演示使用protected成员的实例

```
#include <iostream>
using namespace std;
class Point{
protected:
        int x,y;
public:
        Point(int a,int b){x=a;y=b;}  //Point类构造函数为内联函数
        void Show( ){cout<<"x="<<x<<",y="<<y<<endl;};
};
class Rectangle:public Point{
private:
        int H,W;
public:
        Rectangle(int,int,int,int);  //声明Rectangle类的构造函数
        void Show( ){cout<<"x="<<x<<",y="<<y<<",H="<<H<<",W="<<W<<endl;}
};
Rectangle::Rectangle(int a,int b,int h,int w):Point(a,b){H=h;W=w;}
//派生类的内联构造函数定义形式：派生类名(参数表0):基类名(参数表) { }
void main( ){
        Point a(3,4);  //定义Point类对象a，并调用Point类构造函数初始化
        Rectangle r1(3,4,5,6);  //定义Rectangle类对象r1，并初始化
        a.Show( );    //基类对象a调用基类Show函数，输出“x=3,y=4”
        r1.Show( );   //派生类对象r1调用派生类Show函数，输出“x=3,y=4,H=5,W=6”
}
```

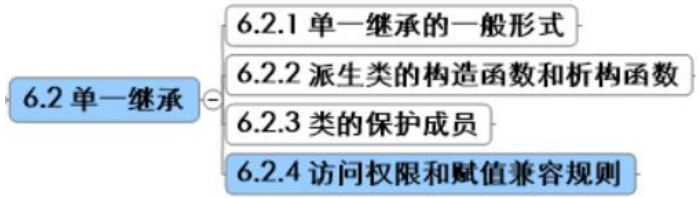
x=3,y=4
x=3,y=4,H=5,W=6

四、访问权限和赋值兼容规则

1、公有派生和赋值兼容规则

在公有派生情况下，基类成员的访问权限在派生类中保持不变。这就意味着在程序中：基类的公有成员在派生类中仍然是公有的；基类的保护成员在派生类中仍然是保护的；基类的不可访问的和私有的成员在派生类中也仍然是不可访问的。

在公有派生情况下，可以通过定义派生类自己的成员函数来访问派生类对象继承来的公有和保护成员，但不能访问继承来的私有成员和不可访问成员（基类的基类的私有成员）。所以当希望类的某些成员能够被派生类访问，而又不能被其他的外界函数访问的时候，就应当把它们定义为保护的，而千万不能把它们定义为私有的，否则在派生类中它们就会是不可访问的。



四、访问权限和赋值兼容规则

所谓**赋值兼容规则**，就是在**公有派生**情况下，通过将基类的成员都定义为公有或保护的，从而使每一个派生类的对象都可作为基类的对象来使用的情况。比如，如果将基类的某个成员定义为私有的话，则派生类中继承的该成员就不能被派生类的成员函数访问，这样派生类的对象就不能作为基类的对象使用，也就不符合赋值兼容规则。

例3：演示赋值兼容规则的实例

```
#include <iostream>
using namespace std;
class Point{
protected:int x,y;
public:     Point(int a,int b){x=a;y=b;}  //Point类构造函数为内联函数
            void Show( ){cout<<"x="<<x<<",y="<<y<<endl;};
};
class Rectangle:public Point{
private:     int H,W;
public:
            Rectangle(int,int,int,int);  //声明Rectangle类的构造函数
            void Show( ){cout<<"x="<<x<<",y="<<y<<",H="<<H<<",W="<<W<<endl;}
};
Rectangle::Rectangle(int a,int b,int h,int w):Point(a,b){H=h;W=w;}
void main( ){
            Point a(1,2);  //定义基类对象a，并调用基类构造函数进行初始化
            Rectangle b(3,4,5,6);  //定义派生类对象b，并调用构造函数初始化
            a.Show( );  //输出“x=1,y=2”
            b.Show( );  //输出“x=3,y=4,H=5,W=6”
            Point& ra=b;  //定义基类的引用ra，并用派生类对象对其初始化
            ra.Show( );    //引用ra调用基类的成员函数Show，输出“x=3,y=4”
            Point* p=&b;   //定义基类的指针p，并用派生类对象b的地址对其进行赋值
            p->Show( );    //指针p调用基类的成员函数Show，输出“x=3,y=4”
            Rectangle* pb=&b;  //定义派生类的指针pb，并用派生类对象b的地址对其进行赋值
            pb->Show( );    //指针pb调用派生类的成员函数,输出“x=3,y=4,H=5,W=6”
            a=b;    //派生类对象的属性值更新基类对象的属性值
            a.Show( );  //输出更新后的保护成员数值“x=3,y=4”}
```

x=1,y=2
x=3,y=4,H=5,W=6
x=3,y=4
x=3,y=4
x=3,y=4,H=5,W=6
x=3,y=4

2、公有继承“isa”和分层“has-a”的区别

2、公有继承“isa”和分层“has-a”的区别

所谓“isa”，就是公有继承“就是一个”的含义，是指在公有继承的赋值兼容规则下，如果类B公有继承于类A，在可以使用类A的对象的任何地方，则类B的对象同样也能使用，即每一个类B的对象“就是一个”类A的对象，但反之则不然，即如果需要一个类B的对象，则类A的对象就不行。

所谓“has-a”，就是分层时“有一个”的含义，分层也称包含、嵌入或聚合，它是一种处理过程，通过让分层的类里包含被分层的类的对象作为其数据成员，以便把一个类建立在另一些类之上。例如一个person对象有一个名字、一个地址和两个电话号码，那么person类表示的就是一种“has-a”的关系。

3、公有继承存取权限表

派生类一般都使用**公有继承**，对派生类而言，基类中的保护类型的成员介于私有和公有之间，派生类可以访问它，而类的对象、外部函数以及不属于本系类之外的类则不可访问它。

基类成员	派生类成员函数	基类和派生类对象	外部函数
private成员	不可访问	不可访问	不可访问
protected成员	protected	不可访问	不可访问
public成员	public	可访问	可访问

4、私有派生

通过私有派生，基类的私有和不可访问成员在派生类中是不可访问的，而公有和保护成员这时就成了派生类的私有成员，派生类的对象不能访问继承的基类成员，必须定义公有的成员函数作为接口。

更重要的是，虽然派生类的成员函数可通过自定义的函数访问基类的成员，但将该派生类作为基类再继续派生时，这时即使使用公有派生，原基类公有成员在新的派生类中也将是不可访问的。只能通过调用派生类自定义的函数访问原基类的公有和保护成员，这样就切断了原基类与外界的联系。私有派生的这种特点不利于进一步派生，因此实际中应用的不多。

例4：私有派生的类继续派生的例子

```
#include <iostream>
using namespace std;
class Point{
private:
        int x,y;
public:
        Point(int a,int b){x=a;y=b;}  //Point类构造函数为内联函数
        void Show( ){cout<<"x="<<x<<",y="<<y<<endl;};};
class Rectangle:private Point{   //私有继承，其公有成员将成为派生类的私有成员
private:
        int H,W;
public:
        Rectangle(int a,int b,int h,int w):Point(a,b){H=h;W=w;}
        void Show( ){Point::Show( );cout<<"H="<<H<<",W="<<W<<endl;}
};
class Test:public Rectangle{    //公有继承，仍能访问基类的公有成员
public:
        Test(int a,int b,int h,int w):Rectangle(a,b,h,w){ }
        void Show( ){Rectangle::Show( );}};
void main( ){
        Point a(1,2);  //定义基类对象a，并调用基类构造函数进行初始化
        Rectangle b(3,4,5,6);  //定义派生类对象b，并调用构造函数初始化
        Test c(7,8,9,10);  //定义再次派生类对象c，并调用构造函数初始化
         a.Show( );  //输出“x=1,y=2”
         b.Show( );  //输出“x=3,y=4” “H=5,W=6”
         c.Show( );  //输出“x=7,y=8” “H=9,W=10”
}
```

```
x=1,y=2
x=3,y=4
H=5,W=6
x=7,y=8
H=9,W=10
```

5、保护派生

派生也可使用protected定义，这种派生使原来的权限降一级使用，即private变为不可访问；protected不变；public变为protected。因为限制了数据成员和成员函数的访问权限，所以用的也很少。

它与私有派生的区别主要在于下一级的派生中，如果降上例中Rectangle类改为如下的保护继承方式：

```
class Rectangle: protected Point {  //类体 };
```

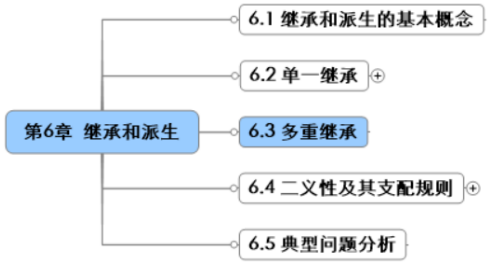
则Test类中就可以使用基类的Show函数，即下面的函数定义是正确的：

```
class Test:public Rectangle{    //公有继承，仍能访问基类的公有成员
public:
        Test(int a,int b,int h,int w):Rectangle(a,b,h,w){ }
         void Show( ){Point::Show( );Rectangle::Show( );}};
```

三种继承方式下基类成员在派生类中的访问属性

基类私有成员	继承方式	基类成员在派生类中的访问属性
Private	Public	不可直接访问
Private	Protected	不可直接访问
Private	Private	不可直接访问
Protected	Public	Protected
Protected	Protected	Protected
Protected	Private	Private
Public	Public	Public
Public	Protected	Protected
Public	Private	Private

访问属性	作用
private	只允许该类的成员函数及友元函数访问，不能被其他函数访问
protected	既允许该类的成员函数及友元函数访问，也允许其派生类的成员函数访问
public	既允许该类的成员函数访问，也允许类外部的其他函数访问



§6.3 多重继承

多重继承可视为单一继承的扩展，一个类从多个基类派生的一般形式为：

class 类名1:访问控制 类名2，访问控制 类名3，…，访问控制 类名n{ //类体}

其中，类名1继承了类名2到类名n的所有数据成员和成员函数，访问控制用于限制其派生类中的成员对基类的访问控制，规则和单一继承情况一样。

例5：演示多重继承的实例

```
#include <iostream>
using namespace std;
class A{
private:
        int a;
public:
        void setA(int x){a=x;}
        void showA( ){cout<<"a="<<a<<endl;}
};
class B{
private:
        int b;
public:
        void setB(int x){b=x;}
        void showB( ){cout<<"b="<<b<<endl;}
};
class C:public A,private B{
private:
        int c;
public:
        void setC(int x,int y){c=x;setB(y);}
        void showC( ){showB( );cout<<"c="<<c<<endl;}
};
```

```
void main( ){
        C obj;
        obj.setA(53);
        obj.showA( );    //输出信息"a = 53"
        obj.setC(55,58);
        obj.showC( );    //输出信息"b = 58 c = 55"
}
```

```
a=53
b=58
c=55
```

§6.4 二义性及其支配规则

如果一个表达式的含义能解释为可以访问多个基类中的成员，则这种对基类成员的访问就是不确定的，称这种访问具有二义性，C++规定对基类成员的访问必须是无二义性的。

对基类成员的访问必须是无二义性的，如使用一个表达式的含义能解释为可以访问多个基类中的成员，则这种对基类成员的访问就是不确定的，称这种访问具有二义性。

一、通过作用域分辨符和成员名限定消除二义性

作用域分辨操作的一般形式如下：类名::标识符，“类名”可以是任一基类或派生类名，“类标识符”是该类中声明的任一成员名。

```
#include <iostream>
using namespace std;
class A{
public:
        void func(){cout<<"a.func"<<endl;}
};
class B{
public:
        void func(){cout<<"b.func"<<endl;}
        void gunc(){cout<<"b.gunc"<<endl;}
};
class C:public A,public B{
public:
        void gunc(){cout<<"c.gunc"<<endl;}
        void hun1(){A::func();}  //使用基类A的func（）
        void hun2(){B::func();}  //使用基类B的func（）
};
```

```
void main(){
    C obj;
    obj.A::func();    //输出a.func()
    obj.B::func();    //输出b.func()
    obj.B::gunc();    //输出b.gunc()
    obj.C::gunc();    //输出c.gunc()
    obj.gunc();       //使用类C的gunc()，输出c.gunc()
    obj.hun1();       //输出a.func()
    obj.hun2();       //输出b.func()
     C c;
     C * c1=new C;
     c.C::gunc();        //使用对象c，输出c.gunc
     c1->C::gunc();      //使用指针c1，输出c.gunc
     c.B::func();        //使用对象c，输出b.func
     c1->A::func();      //使用指针c1，输出a.gunc
 }
```

```
a.func
b.func
b.gunc
c.gunc
c.gunc
a.func
b.func
c.gunc
c.gunc
b.func
a.func
```

二、派生类支配基类的同名函数

基类的成员和派生类新增的成员都具有作用域，基类在外层，派生类在内层。如果此时派生类定义了一个和基类成员函数同名的新成员函数（因参数不同属于重载，所以这里是指具有相同参数表的成员函数），派生类的新成员函数就覆盖了外层的同名成员函数，在这种情况下，直接使用成员名就只能访问派生类的成员函数，只有使用作用域分辨，才能访问基类的同名成员函数。

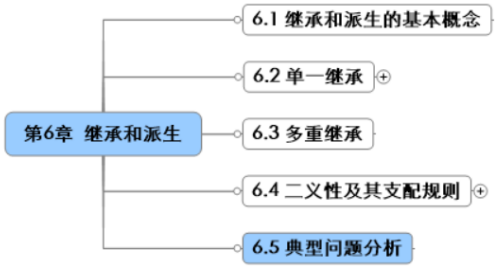
派生类D中的名字N支配基类B中同名的名字N，称为名字支配规则。如果一个名字能支配另一个名字，则二者之间就不存在二义性，当选择该名字时，使用的是支配者的名字，如果要使用被支配者的名字，则应使用成员名限定。

二、派生类支配基类的同名函数

由于二义性的原因，一个类不能从同一个类中直接继承一次以上，例如语句“class derived:public base,public base { //类体};”是错误的，如果必须这样作，可以使用一个中间类。

二义性检查是在访问权限检查之前进行的，因此成员的访问权限不能解决二义性问题。

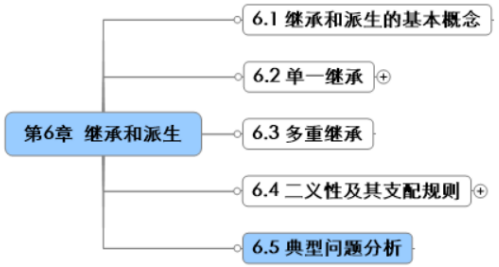
如果涉及几层继承关系，对于任一基类中可以存取的成员，都可以通过作用域分辨进行存取。一般只有派生类中使用的标识符和基类中的标识符同名时，才有必要使用作用域分辨符进行存取。



§6.5 典型问题分析

一、派生类没有使用基类的数据成员

使用派生时，爱犯的错误是忘记派生类已经继承了基类的数据成员，而又重新定义新的数据成员，例如从Point类派生Rectangle类，忘记后者继承了前者的一个点（x，y），而又重新定义了四个数据成员（x，y，H，W），其实，后者只需再定义两个数据成员H和W即可。



§6.5 典型问题分析

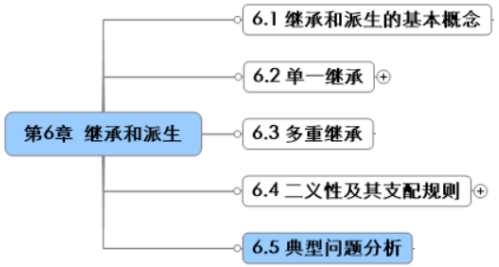
二、派生类支配基类的同名函数

当基类和派生类具有同名成员函数时，往往不知对象调用哪个成员函数，其实，只需记住：对象一定先调用自己的同名成员函数，只有使用了作用域分辨运算符，才会调用直接基类的同名成员函数。

三、二义性

程序设计过程中，一定要注意避免定义的二义性，为此可通过使用**作用域分辨运算符“::”和成员名限定来消除二义性**。

§6.5 典型问题分析

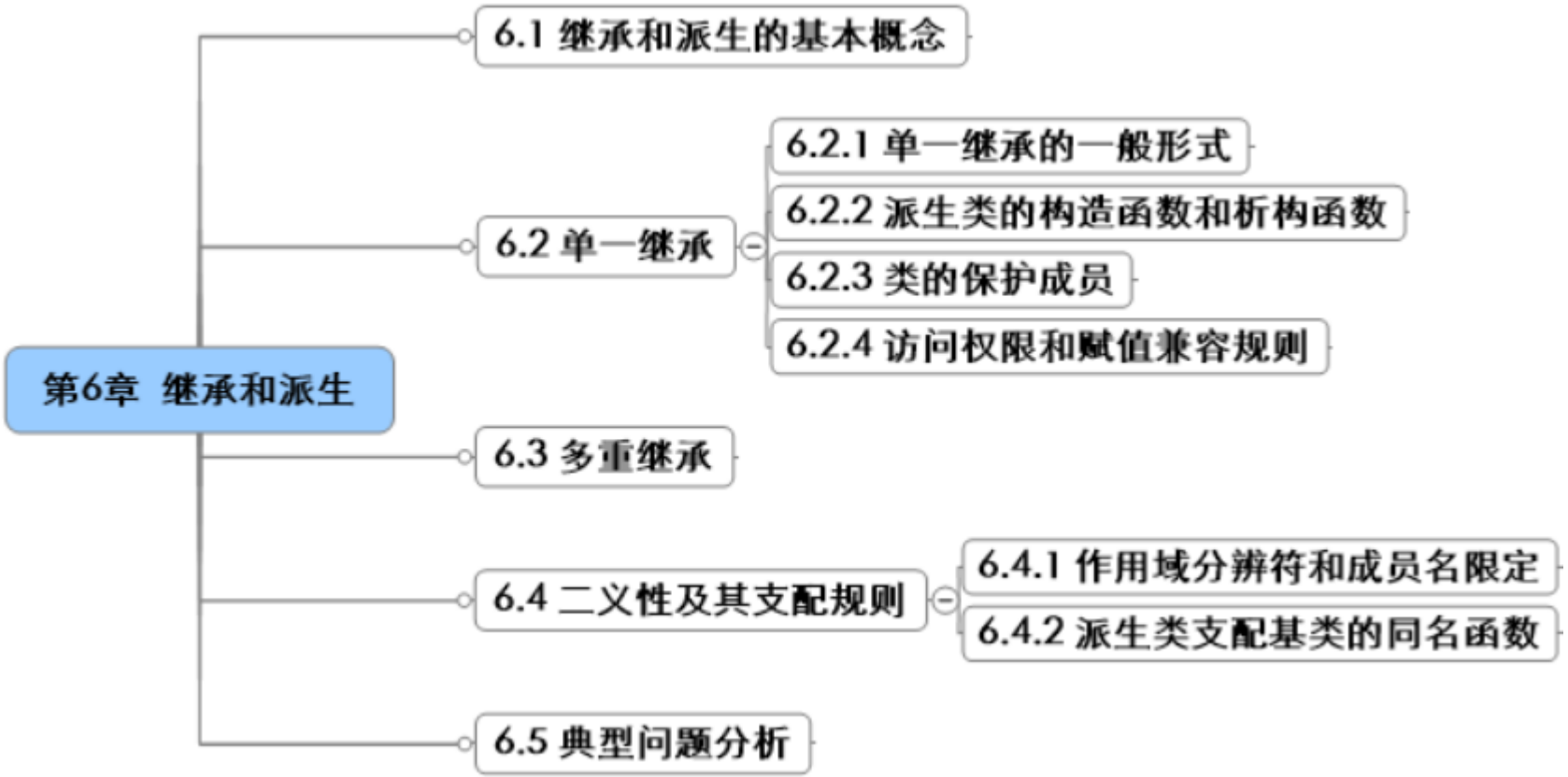


四、友元和派生类

友元声明与访问控制无关。友元声明在私有区域进行或在公有区域进行是没有太大区别的。虽然友元函数的声明可以置于任何部分，但通常置于能强调其功能的地方以使其直观。对友元函数声明的唯一限制是该函数必须出现在类声明内的某一部分。

友元关系是无等级的。友元可以访问任何一个类成员函数可以访问的对象，这比一个派生类可以访问的对象还多。友元关系不能继承。一个派生类的友元只能访问该派生类的直接基类的公有和保护成员，不能访问私有成员。当友元访问直接基类的静态保护成员时，只能使用对象名而不能使用成员名限定。

本章总结


第6章 继承和派生
6.1 继承和派生的基本概念
6.2 单一继承
6.2.1 单一继承的一般形式
6.2.2 派生类的构造函数和析构函数
6.2.3 类的保护成员
6.2.4 访问权限和赋值兼容规则
6.3 多重继承
6.4 二义性及其支配规则
6.4.1 作用域分辨符和成员名限定
6.4.2 派生类支配基类的同名函数
6.5 典型问题分析

考试真题演练

10. 下列关于构造函数的描述中，错误的是

A. 构造函数可以设置默认参数　　B. 构造函数在定义类对象时自动执行

C. 构造函数可以是内联函数　　D. 构造函数不可以重载

11. 关于成员函数特征的描述中，错误的是

A. 成员函数一定是内联函数　　B. 成员函数可以重载

C. 成员函数可以设置参数的默认值　　D. 成员函数可以是静态的

12. 下列不是函数重载所要求的条件是

A. 函数名相同　　B. 参数个数不同　　C. 参数类型不同　　D. 函数返回值类型不同

14. 用 new 运算符创建一维数组的正确形式是

A. int * p = new a[10];

B. float * p = new float[10];

C. int * p = new float[10];

D. int * p = new int[5] = {1,2,3,4,5,6};

15. 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. 类的友元函数都有 this 指针

D. 类的非静态成员函数才有 this 指针

16. 如果有 int x, * p;float y, * q;则下面操作中,正确的是

A. p = x

B. p = q

C. p = &x

D. p = &y

33. 如果要把 Student 类的返回值为 void 的成员函数 score( ),声明为类 Teacher 的友元函数,则应在类 Teacher 的定义中加入语句_____。

34. 假定 x 是一个逻辑量,则 x&&0 的值为_____。

35. 构造函数、析构函数和友元函数中,不是该类成员的是_____。

36. 使用对象的引用来初始化创建中的对象的函数是_____。

37. 定义类的动态对象数组时,系统只能够自动调用该类的_____构造函数对其进行初始化。

38. 复制构造函数使用_____作为形式参数。

7. 关于构造函数的说法，不正确的是

A. 没有定义构造函数时，系统将不会调用它

B. 其名与类名完全相同

C. 它在对象被创建时由系统自动调用

D. 没有返回值

8. 所谓数据封装就是将一组数据和与这组数据有关操作组装在一起，形成一个实体，这实体也就是

A. 类　　B. 对象　　C. 函数体　　D. 数据块

9. 下列关于类的继承描述中，错误的是

A. 基类不一定具有派生类的全部属性和方法

B. 派生类可以访问基类的所有数据成员，也能调用基类的所有成员函数

C. 继承描述类的层次关系，派生类可以具有与基类相同的属性和方法

D. 一个基类可以有多个派生类，一个派生类可以有多个基类

12. 下列关于类的权限的描述错误的是

A. 类本身的成员函数只能访问自身的私有成员

B. 类的对象只能访问该类的公有成员

C. 普通函数不能直接访问类的公有成员，必须通过对象访问

D. 一个类可以将另一个类的对象作为成员

13. 下面不能够判断字符串 S 是空串的是

A. if (S[0] ==0)

B. if(strlen(S) ==0)

C. if(strcmp(S,"") ==0)

D. if(S == '\0')

14. 下列输出字符'd'的方法中，错误的是

A. cout << put('d')

B. cout << 'd'

C. cout. put('d')

D. char a = 'd'; cout << a;

15. 关于引用，下列的说法中错误的是

A. 引用是给被引用的变量取一个别名

B. 引用主要是用来作函数的形参和函数的返回值

C. 在声明引用时，要给它另开辟内存单元

D. 在声明引用时，必须同时使它初始化

16. 下面关于 C ++ 字符数组的叙述中，错误的是

A. 字符数组可以放字符串

B. 字符数组的字符可以整体输入、输出

C. 可以在赋值语句中通过赋值运算符“ = ”对字符数组整体赋值

D. 可以用关系运算符对字符数组比较大小

17. 下列说法不正确的是

A. 主函数 main 中定义的变量在整个文件或程序中有效

B. 不同函数中，可以使用相同名字的变量

C. 形式参数是局部变量

D. 在一个函数内部，可以在复合语句中定义变量，这些变量只在复合语句中有效

18. 非数组指针或引用型变量做实参时，它和对应虚参之间的数据传递方式是

A. 地址传递　　B. 单向值传递　　C. 双向值传递　　D. 由用户指定传递方式

19. 下面叙述中错误的是

A. 预处理命令都必须以“#”开始

B. 在程序中凡是以“#”开始的语句行都是预处理命令行

C. C ++ 程序在程序执行过程中对预处理命令进行处理

D. 一行只能写一条预处理命令

20. 下列关于友元函数的描述，正确的是

A. 友元函数可以存取私有成员、公有成员和保护成员

B. 友元函数不可以是一个类

C. 友元函数的作用之一是实现数据的隐藏性

D. 在类中说明的友元函数，函数的定义不可在类体之外

21. #include < iostream. h > 命令中，include 的意义是：_____。

22. this 指针始终指向调用成员函数的 _____。

23. 假定 AB 为一个类，则执行语句 AB a[10]；时，系统自动调用该类的构造函数的次数为 _____。

30. 如果要把 A 类成员函数 f( )且返回值为 void 声明为类 B 的友元函数,则应在类 B 的定义中加入语句 _____ 。

32. 为了在对象生存期结束时释放其指针成员所指向的动态存储空间,通常为该类定义 _____。

33. 在保护派生中,基类权限为 public 的成员在派生类中为_____。

34. 在函数前面用_____保留字修饰时,则表示该函数为内联函数。

35. 面向对象的四个基本特性是多态性、继承性、封装性、_____ 。

36. 派生类的主要用途是可以定义其基类中_____。

改错题：

```
#include <iostream.h>
#include <string.h>
class Base
{public:
Base(char *s="\0"){strcpy(name,s);}
void show();
protected:
char name[20];
};
Base b;
void show()
{cout<<"name:"<<b.name<<endl;}
void main()
{Base d2("hello");
show();
}
```

1. 以下说法中不正确的是

A. C++程序中必须有一个主函数 main(),而且是从 main()的第一条语句开始执行

B. 非主函数都是在执行主函数时,通过函数调用或嵌套调用而得以执行的

C. 主函数可以在任何地方出现

D. 主函数必须出现在固定位置

2. 若有定义 int * p = new int(0),则下列说法正确的是

A. 系统用指针变量 p 来表示所指整型变量

B. 声明一个指针变量 p,指向名为 new 的存储单元

C. 系统为指针变量 p 分配一个整型数据的存储空间

D. 通过运算符 new,分配一个整型数据的存储空间,并将其内存地址赋予指针变量

3. 以下有关类与对象的叙述中,错误的是

A. 对象是类的一个实例

B. 一个类可以有多个对象

C. 任何一个对象都归属于一个具体的类

D. 只要是某个类的对象,那么该对象就可以访问这个类的所有成员

4. 以下有关构造函数的叙述中，错误的是

A. 构造函数名必须和类名一致

B. 构造函数在定义对象时自动执行

C. 在一个类中构造函数有且仅有一个

D. 构造函数可以在类体内声明在类体外实现

5. 以下叙述中正确的是

A. 类成员的定义必须放在类体内部

B. 在类中，不作特别说明的数据成员均为私有类型

C. 在类中，不作特别说明的数据成员均为公有类型

D. 类成员的定义必须是成员变量在前，成员函数在后

6. 友元函数的主要作用是

A. 提高程序的效率　　B. 加强类的封装性

C. 实现数据的隐蔽性　　D. 增加成员函数的种类

7. 在 C++ 中,字符型数据在内存中的存放形式为

A. 原码　　B. BCD 码　　C. ASCII　　D. 反码

8. 下列变量命名中,非法的是

A. A * * LONG　　B. MyCar　　C. my_car　　D. a48

9. 关于对类的描述中,错误的是

A. 类是创建对象的样板　　B. 类是具有唯一标识符的实体

C. 类就是 C 语言中的结构类型　　D. 类是具有共同行为的若干对象的统一描述体

10. 不能作为函数重载的判断依据的是

A. const　　B. 返回类型　　C. 参数个数　　D. 参数类型

11. 已知:“int a =5;char c = 'a';”则输出语句 cout <<c +1 <<a <<c;的显示结果是

A. 65a　　B. 985a　　C. 98'5'a　　D. 65'a'

13. C++语言的跳转语句中,对于 break 和 continue 说法正确的是

A. break 语句只应用于循环体中

B. continue 语句只应用于循环体中

C. break 是无条件跳转语句,continue 不是

D. break 和 continue 的跳转范围不够明确,容易产生问题

12. 下面程序的输出结果是

```
#include <iostream>
using namespace std;
int main()
{int a=1,b=-2,c=3;
if(a<b)
if(b<0)c=0;
else c++;
cout<<c<<endl;
return 0;
}
```

A. 0　　B. 2　　C. 3　　D. 4

16. 对于下面定义的类 Myclass，在函数 f( ) 中将对象成员 n 的值修改为 50 的语句应该是

```
class Myclass
{public:
    Myclass(int i){n=i;}
    void SetNum(int x){n=x;}
private:
    int n;
};
int f( )
{Myclass *p=new Myclass(45);
                                    }
```

A. p->SetNum(50)

B. SetNum(50)

C. p->n=50

D. *p->SetNum(50)

17. 下面关于对象概念的描述中错误的是

A. 任何对象都必须有继承性　　B. 对象是属性和方法的封装体

C. 对象间的通信靠消息传递　　D. 操作是对象的动态属性

18. 考虑函数原型 void pass(int x,int y=5,char z='*'),下面的函数调用中,属于不合法调用的是

A. pass (5)　　B. pass (5,8)　　C. pass (6,'#')　　D. pass (0,0,'*')

20. 有以下程序段,其输出结果是

```
#include <iostream>
using namespace std;
void main( )
{char b[ ] = "Hello,you" ;
b[5] = '!' ;
cout << b << endl;
}
```

A. Hello,you　　B. Hello　　C. Hello! you　　D. !

21. 一般 C++ 语言源程序文件的后缀是 .cpp；经过编译后，生成文件的后缀是 .obj；经过连接后，生成文件的后缀是_____。

22. C++ 语言中提供了 3 种循环语句：_____循环语句、for 循环语句和 do_while 循环语句。

23. 在“int a = 10, * p = &a;”语句中，p 的值是_____。

24. 执行 3 条语句“int a,b, * c = &a;int * p = c;p = &b;”后，c 指向_____。

25. 若有以下定义：double w[9]；则 w 数组元素下标的下限是 0，上限是_____。

26. 设 int x[3][4]；则 x 数组中含有_____个 int 类型的数组元素。

27. 将数学表达式 $ab + \frac{a^2 + b^2}{2ab}$ 写成 C++ 语言表达式为：_____。

29. 源程序文档化要求程序应加注释，注释一般分为序言性注释和_____。

30. 重载函数必须有不同的_____。

31. 在类中定义和实现的函数可以成为_____，它能够加快程序执行速度。

33. C++ 语言中，派生类继承了基类的全部数据成员和除构造函数及_____之外的全部函数。

34. 定义以下变量并假设已赋确定的值："char w;int x;foat y;double z;"，则表达式"w * x +z - y"的数据类型是_____。

35. 若有一个 Myclass 类，则执行语句"MyClass obj1,obj2[2], * p;"后，自动调用该类的构造函数_____次。

36. this 指针保证每个对象拥有自己的数据成员，又共享处理这些数据成员的_____。

37. 若有整型变量 a=1,b=-2,c=3;则表达式 a-b>b? c:a+b 的值为_____。

38. 所谓“引用”就是将一个新标识符和一块已经存在的_____相关联,通常用于函数的参数表中或者作为函数的返回值。

39. C++ 为结构动态分配内存的一般格式是“指针名 = _____;”,当不再使用这个空间时必须用“delete 指针名;”释放空间。

40. C++ 程序中使用 string 类定义存储字符串的对象时,必须在程序中包含这个类的头文件,即使用语句_____。

祝大家顺利通过考试！