全国高等教育自学考试系列辅导教材

面向对象程序设计复习与考试指导

计算机及应用专业（独立本科段）

主编　刘振安

编者　刘振安　葛愿　李佳　蒋里

高等教育出版社

全国高等教育自学考试系列辅导教材

面向对象程序设计复习与考试指导

计算机及应用专业（独立本科段）

主编　刘振安
编者　刘振安　葛愿　李佳　蒋里

高等教育出版社

内 容 提 要

本书是全国高等教育自学考试“计算机及应用专业”系列辅导教材之一，是独立本科段“面向对象程序设计”课程的复习与考试指导教材。它依据考试大纲“识记、领会、识别、应用”的四个层次，紧密结合自考考生特点，以典型例题的形式对每一章节的内容进行透彻的分析与解答，每一章后配有自测试题供考生及时检验学习效果；同时本书还提供了四套模拟试卷，全面涵盖本课程的考核知识点，以利考生参悉考试过程，从而进入良好的考试状态并顺利通过考试。

本辅导教材除适用于“计算机及应用”专业的考生外，还适用于“计算机信息管理”“计算机通信工程”和“计算机应用及教育”等专业相同课程的考生，也可以作为工程技术人员、社会读者的学习参考用书。

图书在版编目(CIP)数据

面向对象程序设计复习与考试指导 刘振安主编
—北京：高等教育出版社，2001.12
本科自考 成人教育计算机及应用和相关专业用书
ISBN 7-04-010356-7

Ⅰ.面… Ⅱ.刘… Ⅲ.面向对象语言-程序设计-高等教育-自学考试-自学参考资料 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2001)第 072057 号

面向对象程序设计复习与考试指导
刘振安 主编

出版发行	高等教育出版社		
社 址	北京市东城区沙滩后街 55 号	邮政编码	100009
电 话	010-64054588	传 真	010-64014048
网 址	http://www.hep.edu.cn http://www.hep.com.cn		
经 销	新华书店北京发行所		
印 刷	国防工业出版社印刷厂		
开 本	787×1092 1/16	版 次	2001 年 12 月第 1 版
印 张	14.25	印 次	2001 年 12 月第 1 次印刷
字 数	340 000	定 价	14.00 元

序

高等教育自学考试正式实施已经 20 周年了，20 年来，每年都有数以百万计的人参加考试，目前全国自考在籍考生已达 1600 多万人，显示了其在高等教育总体格局中的地位。近年来，随着计算机技术及应用水平的不断提高，报考“计算机及应用”专业的考生逐年增加。

为了满足自学考生的需求，高等教育出版社根据全国高等教育自学考试指导委员会 1999 年重新修订的“计算机及应用专业教学大纲”的教学基本要求和相应新教材的出版，组织编写了“高等教育自学考试计算机及应用专业系列辅导教材”。在这一系列辅导教材中，包括“计算机及应用”专业专科和独立本科段开考的共 15 门课程的辅导教材。

这套辅导教材是根据教材的编写思想和教学大纲的基本要求，从自考生的学习水平及自考生以自学为主的特点出发，为应试考生提供既包含自考教材中的考核知识点，又包含各知识点相互关系的讲解；既指出课程中的学习重点，又进行课程难点的分析；既提供多种典型例题，又详细地进行分析解答；各章后面附有大量的自测试题，以帮助考生测试各章内容的掌握情况，全书后面包含有多套模拟试卷，综合、全面测试考生对本课程的理解和掌握情况；使考生既学到了知识，又逐渐进入考试的实战状态，达到考试中应答自如，顺利通过考试的目的。本系列辅导教材并不单纯以解答教材中的例题为目的，而是通过作者在自考辅导过程中积累的典型例题，逐层分析、解答，从而使考生全面掌握本课程的知识。由此，也构成了本系列辅导教材例题丰富、实用性强的特点。

在组织编写这套系列辅导教材时，高等教育出版社选择的作者中，有些是自学考试教材的编写者，有些是多年来一直从事自学考试辅导教学工作的出色教师，他们非常了解自学考生的特点，富有自学考试的辅导经验。

本系列辅导教材适用于“计算机及应用”专业专科和独立本科段考生的自学及教师上课辅导，也适用于“计算机信息管理”专业、“计算机通信工程”专业、“计算机应用及教育”专业相同课程的考生，也可以作为工程技术人员、社会读者的学习用书。

全国高等教育自学考试指导委员会
电子电工与信息类专业委员会副主任
陈国良
2001、7

前　言

由于高等教育自学考试具有试卷上各道试题难易程度变化不大，但知识覆盖面广的特点，所以考生只要充分复习所学的课程内容，就能正确答出考题。也正因为考试的知识面广，所以常碰到不会的试题也是正常现象。不过要注意的是，自学考试与高考是有区别的，自学考试不受录取名额限制，只要考生复习充分并保持良好的考试心态，就可以通过考试。为了帮助考生更好地掌握教学内容，理解考试大纲的要求及规定的考试题型，我们编写了本书。

本书章节结构和辅导内容均与教材一致，重点是结合考试大纲的要求，先介绍知识点，再针对自学难点和容易混淆的概念，通过精选典型例题进行重点与难点分析，最后又增加一章总复习，从另一个角度指导考生系统地复习所学内容。

考试大纲给出七种题型，但同一个问题可以采取不同的题型，四个能力层次的试题都可以反映在这七种题型中，尤其程序分析、改错和完成程序的灵活性更大，所以不要死记题型，应掌握同一问题可使用不同题型的变化方法。附录给出的自测模拟试题的目的就是想通过演示题型的变化，帮助考生进一步掌握书本的内容，提高应试能力。每套试题的份量较大，试卷中的不同能力层次的试题所占的比例也不相等，有个别试题也较难，目的是进行能力训练并提高应变能力。

学习 C++的关键是必须从原来已经比较熟悉的，面向过程的设计方法中转变思想，接受面向对象的编程思想。这种思想方法的转变是需要花大力气的，尤其需要下定决心抛弃原来的编程习惯和思考方法，必须牢固掌握 C++语言的基本结构和主要特征，而 C++语言的独有特征及其与 C 语言的区别则是学习过程中的重点和难点。在自学时要注意与教材同步，注意每一章的考核知识点以及各种题型的解题能力，不仅应掌握辅导教材给出的题型变化，还要掌握分析问题的方法。只要按照要求吃透并掌握各种题型的分析解题能力，就能具有应变能力，顺利通过考试。

我们精心组织例题和习题并对程序进行仔细验证。参加本书编写的主要有刘振安、葛愿、李佳、蒋里、孙忱、尹雷、彭程、潘剑峰、徐菁、张巍、章守信、田钢、颜廷荣、叶刚、刘胜璞、马杰华、陈志雄和叶刚等。

由于这门课程以前没有考过，而且又是新开设的课程，所以更要求考生脚踏实地准备应试。相信本辅导教材能够帮助考生充分准备考试，从而考出好的成绩。

刘振安

2001.5.8 于合肥

目　录

第一章　面向对象及C++基础知识

本章主要的要求是能初步理解面向对象程序设计及C++语言中的新思想，目的是为学习C++的关打下坚固的基础。

1.1　必考知识点及相互关系

必考知识点中的“领会”层次包括编译指令的几种形式及其作用，并能在程序中应用编译指令。对面向对象程序设计思想，C++语言与C语言的关系，要求达到“识记”层次，其是要先建立对 const 修饰符的认识，了解它的作用和使用方法。对于基本程序的组成，要求达到应用“层次”。

本章对应用层次的要求较多，应该通过一些典型C++程序，并通过对比C语言，认识C++的新特点。这包括注意利用已有知识学习C++，例如可以给标准的C程序增加C++的新特点来理解这些新内容。这些新内容包括程序的基本结构和作用；注释、续行、标准输入及输出语句；头文件及常量的定义；函数原型的作用、意义及其使用方法；对新引入的 new 和 delete 内联函数及引用等，都要建立正确的认识。

为了给以后的学习打下基础，本章将重点介绍一下引用和 const 的基本使用知识，并对使用中的疑难问题，通过实例加以解释。这些内容虽然超出教材第一章的范围，但考试大纲要求范围之内，所以希望能熟练掌握它们。另外，把面向对象的特点放在第二章，结合类来阐明它们，以增进感性认识。

1.2　重点与难点分析

因为本章仅仅是初步建立C++程序，new 和 delete、内联函数及引用等概念，所以本章重点放在对新概念的理解和简单应用上，尤其是正确理解 const 和引用的概念，这有以后的学习很有帮助。只有在以后的学习中不断加深理解并扩大应用的深度，才能熟练地掌握和正确地使用它们。

【例1.1】 编写一个程序，用来演示C++的基本程序结构。

为了演示C++的基本程序结构，设计一个输入半径，输出其面积的示例程序如下：

```
#include <iostream.h>                          系统头文件
const double pi=3.14159;
void main()
```

```
{
  float r;
  cout<<"r=";                                  输出提示信息
  cin>>r;                                      通过键盘给半径赋值
  float s = pi*r*r;
  cout <<"\nThe area is: "<< s<<endl;
}
```

假设输入 5.6，则运行过程及结果如下：

```
r=5.6
The area is: 98.5203
```

这个程序首先演示了如何使用系统头文件 iostream.h 及使用类型修饰符 const 修饰 pi，编译器将 pi 视为一个常量，不再为它分配内存，并且每当在程序中遇到它时，都用在说明时所给出的初始值 3.14159 取代它。

“cout<<”是把后面的内容送到标准输出设备（这里是显示器），而“cin>>”则是把标准输入设备（键盘）接收到的数据，存入后面的变量中。如果把“<<”和“>>”看作表示方向的符号，就会发现数据确实是在按照一定的方向流动，这就是“流”这个名称的由来。C++是自带输入和输出的，并可根据数据的类型，自动地使用合适的输出方式。

C++提供一种新的注释方式：从“//”开始，直到行尾，都被计算机当作注释。例如：

```
i--;                                  // i=i-1
```

C 风格的多行注释“/*……*/”，在 C++中仍然可以使用。一般情况下，在需要短的注释的场合，常常使用行注释方式“//”。

C++使用 endl 换行，可以在一条语句中多次使用换行符。例如：

```
cout<<endl<<"How about it ?"<<endl<<endl;
```

上述语句在“How about it ?”的前后各产生一个空行。

注意：在 C++程序中，可以在需要变量时临时声明，如程序中的变量 s。

【例 1.2】 分析对比引用与 const 的作用。

下面的程序没有使用 const

```
#include <iostream.h>
    void main()
    {
      //const int num=405;              // 用来对比的语句
      int num=405;
      int & x=num;
      int & y=num;
      int &z=y;
      ++x;
      cout<<"x="<<x<<",num="<<num<<", y="<<y<<",z="<<z<<endl;
      ++y;
```

```
    ++x;
    cout<<"x="<<x<<",num="<<num<<",y="<<y<<",z="<<z<<endl;
    z=255;
    cout<<"x="<<x<<",num="<<num<<",y="<<y<<",z="<<z<<endl;
    cout<<"Addr of x is:"<<&x<<endl;
    cout<<"Addr of y is:"<<&y<<endl;
    cout<<"Addr of z is:"<<&z<<endl;
    cout<<"Addr of num is:"<<&num<<endl;
}
```

它的输出结果如下：

```
x=406,num=406, y=406,z=406
x=408,num=408 ,y=408,z=408
x=255,num=255 ,y=255,z=255
Addr of x is:0xfff4
Addr of y is:0xfff4
Addr of z is:0xfff4
Addr of num is:0xfff4
```

由此可见，引用类型是 C++在 C 语言的基础上新增加的一种类型。用引用类型声明的引用，不同于一般的类型所声明的对象。引用不是对象，只是用来标识对象，你可以把它看成是对象的别名。

从上面的程序中明显看到，当改变任意一个别名的值时，引用对象本身及其所有别名会同步变化，因为它们的存储地址是相同的。

如果将语句

```
int num=405;
```

改为如下语句：

```
const int num=405;
```

并将最后一条输出语句注释掉，则程序输出如下：

```
x=406,num=405, y=405,z=405
x=407,num=405 ,y=406,z=406
x=407,num=405 ,y=255,z=255
Addr of x is:0xfff2
Addr of y is:0xfff0
Addr of z is:0xfff0
```

为引用提供的初始值必须是一个也具有初始值的变量。如果是一个常量或是一个使用 const 修饰的变量，则编译器首先建立一个临时变量，然后将该常量的值置入临时变量中，对引用的操作就是对该临时变量的操作。所以 num 不变，而 x 和 y 在第一次取得 num 的值之后，就分道扬镳，各自为政去了，它们各自具有自己的地址，不再存在内部联系。这相当于使用 num 初始化 x 和 y。因为 z 引用了 y，所以它们两个具有相同的地址并同步变化。

应该从现在开始就养成使用 const 和引用的习惯，尤其是定义常数上，更应该马上抛弃

使用 define 定义常数的习惯，而改用 const 定义它们。

面向对象的程序设计方法和传统的程序设计方法相比，面向对象的程序设计具有抽象、封装、继承和多态性等特征。但要真正认识这些特征的内在含义，则要随着知识的展开而逐步了解。

注意：引用常量问题比较复杂，这里所说的复杂，不是说它难于理解，而是指不同的编译器大相庭径。例如，早先的教科书大都是以 Borland C++为例，BC 可以引用常量，而现在流行的 Microsoft Visual C++则不能引用，所以在实际使用时，一定要注意使用的具体环境。

【例 1.3】 结合下面的程序，简要说明 Swap( )函数声明、调用和定义的方法。

```
#include<iostream.h>
void Swap(int& a, int& b);                    // 用原型声明 Swap 函数
int main()
{
  int x(125), y(119);
  cout<<"x="<<x<<", y="<<y<<endl;
  Swap(x,y);                                  // 调用 Swap 函数
  cout<<"x="<<x<<", y="<<y<<endl;
  return 0;
}

void Swap(int& a, int& b)                     // 定义 Swap 函数
{
  int temp;
  temp=a;
  a=b;
  b=temp;
}
```

程序输出如下：

```
x=125, y=119
x=119, y=125
```

函数定义的一般形式如下：

```
类型 函数名（参数表
{
函数体
}
```

函数 Swap(int &x, int &y)的定义在参数表中进行变量说明时，每个变量必须分别指定类型和名字。下述说明方法是不正确的：

```
void Swap (int &x, &y)                        // 错误
```

函数原型标识一个函数的返回类型，同时也标识该函数参数的个数和类型。可以在程序

中使用函数说明语句来说明一个函数的原型。函数说明语句的一般形式为：

类型 函数名（参数类型说明列表）；

其中“列表”是用逗号隔开的一个个类型说明，其个数和指定的类型必须和函数定义中的参数的个数和对应类型一致，例如：

```
void Swap (int &, int &) ;
```

函数原型的重要作用是可以使编译器检查一个函数调用表达式中可能存在的问题。

函数调用是由函数名和函数调用运算符（ ）组成的一个表达式，其一般形式为：

函数名（实参表）

实参表是用逗号隔开的一个表达式列表，其中的每个表达式的值称为实参。在函数调用时，实参的值传给形参；在实参表中，实参的个数必须和形参的个数相同，实参的类型必须和对应的形参的类型一致。函数调用表达式的类型为在函数定义中为该函数指定的类型，如果其类型不为 void 的话，这个表达式值就是函数定义中的 return 语句所返回的值，否则它表示一个无值表达式。

当在函数定义中没有指定形参时，调用表达式中的实参表为空，表明函数不需要从调用者那里接收数据。虽然参数表可以为空，但函数名后的一对圆括号不能省略。

缺省参数就是不要求设定该参数，而由编译器在需要时给该参数赋予预先设定的值。

任何一个 C++程序都必须定义一个 main 函数，它的返回类型总是 int 类型。这个函数由操作系统来调用，在 main 函数执行完以后，程序随之终止。main 也可以使用 return 向操作系统返回一个值。可使用操作系统的命令检查 main 的返回值。一般约定在 main 返回 0 时，表示程序运行过程中没有出现错误，其它非零值表示程序出现异常情况。如果没有为 main 指定返回值，则返回值是任意的。对于不返回值的函数，函数返回类型指定为 void，这时，return 语句中不能带有表达式，但可以在其后加一个分号，形成一个调用表达式语句或将这一条 return 语句省略。

如果将引用作为实参，在执行主函数调用中的调用语句时，系统自动用实参来初始化形参。这样，形参就成为实参的一个别名，对形参的任何操作也就直接作用于实参。

【例 1.4】 比较值调用和引用的实例。

```
#include <iostream.h>
void comp(int, int&);
int main()
{
    int count = 8, index = 15;
    cout << "The values are ";
    cout<<count<<","<<index<<endl;
    comp(count, index);
    cout << "The values are ";
    cout<<count<<","<<index<<endl;
    return 0;
}
```

```
void comp(int in1, int &in2)
{
    in1 = in1 + 100;
    in2 = in2 + in1;
    cout << "The values are ";
    cout<<in1<<","<<in2<<endl;
}
```

comp 函数的第 1 个参数 in1 是普通的 int 型，被调用时传递的是实参 count 的值．第 2 个参数 in2 是引用，被调用时由实参 index 初始化后成为 index 的一个别名。在函数 comp 里面对参数 in1 的改变不会影响实参，但对形参 in2 的改变的实质上是对主函数中变量 index 的改变。所以当它返回主函数后，count 的值没有变化．但 index 的值发生了变化。

```
The values are 8,15
The values are 108,123
The values are 8,123
```

1.3　自 测 试 题

1．分析下面程序的输出结果。

```
#include <iostream.h>
void fun ( )
{
    static int i = 48 ;
    i++ ;
    cout << "i=" << i << endl ;
}

void main ( )
{
    for (int j = 0 ; j < 2 ; j++)
    fun () ;
}
```

2．分析下面程序的功能及函数 abort()和 exit()的区别。

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
void main ()
{
```

```
    float root;
    float input( );
    root=input( );
    cout << "The square root = " << root <<endl;
}

float input ( )
{
    float x;
    cout <<"Enter a real number:";
    cin >> x;
    if  ( x < 0.0) {
    cout << "Input < 0 " << endl;
    exit(1);
    }
    return sqrt(x);
}
```

3. 下面是使用内联函数 CalArea 计算圆的面积，请补上所缺内容。

```
double CalArea(radius)
{return 3.14;}
```

4. 分析使用引用程序的输出。

```
#include <iostream.h>
void addsub(int&, int&);
void main( )
{
    int a=15, b=20;
    addsub(a,b);
    cout << "a="<<a<<", b=" <<b;
}
void addsub(int& m, int& n)
{
    int temp=m;
    m=m+n;
    n=temp-n;
}
```

自测试题答案

1. 静态全局变量的初始化是在程序开始执行主函数 main()之前完成，静态局部变量的

初始化则在程序运行中第一次经过它时进行。在上述程序中，i 的初始化是在程序运行中第二次经过它时进行的，即 i=48,所以程序的输出是：

i = 49

i = 50

2．这是一个计算平方根的程序，函数 input 用来请求用户输入一个浮点数，然后使用函数 sqrt( )求这个数的平方根，并将其值返回到调用者那里。该函数同时检测用户是否输入了一个负数，若是，显示一条错误信息，并使用函数 exit()立即终止程序的执行。

函数 abort()和 exit()的作用一样，不同的是 exit()函数在程序终止之前要做一些清理工作，例如，关闭该程序中已打开的文件，并调用有关的退出函数或析构函数；而函数 abort()却不做这些工作，它仅有的作用是终止程序的运行。调用函数 abort()时不需要提供参数。它在 stdlib.h 中的说明为：

void abort();

3．分析：内联函数使用关键字 inline 定义函数，函数 CalArea 的参数类型为 double

```
inline double CalArea(double radius)          // 内联函数，计算圆的面积
{return 3.14*radius*radius;}
```

4．当程序中调用函数 addsub 时，实参 a 和 b 分别初始化引用 m 和 n，所以在函数 addsub 中，m 和 n 分别引用 a 和 b，对 m 和 n 的访问就是对 a 和 b 的访问，所以函数 addsub 改变了 main 函数中变量 a 和 b 的值。通过使用引用参数，一个函数可以修改另一个函数内的变量，程序的输出为：a = 35，b = -5

第二章　类 和 对 象

在开始学习本章时，对熟悉 C 程序过程设计方法的读者而言，要特别注意体会类的特点及其真正含义，做好转变思想的工作。本章是全书的主角，概念多，容易与 C 程序混淆的地方也不少，需要建立的新概念也具有一定的抽象性，正确建立类的概念，是学好面向对象程序设计的关键。一定要注意通过习题进一步加深理解，为后续的学习打下基础。

2.1　必考知识点及相互关系

对本章的考核内容要求较高，除了结构和联合的内容只要求达到“识记”层次之外，其他的均要求达到应用层次。其中要求达到“简单应用”层次的有成员函数重载的作用和重载方法，空类、嵌套类和 this 指针。而要求达到“综合应用”层次的很多，包括：

（1）类的作用域、对象的存取及类的实例化；

（2）说明类的方法、类的标识符、作用域运算符及成员函数的实现；

（3）说明对象的方法和访问类对象成员的方法；

（4）封装的实现方法和类的存取控制方法；

（5）内联成员函数的作用及其使用方法。

学习中要注意体会理解并掌握 C++语言中类的基础知识及其使用，能够熟练利用类的概念进行编程。建议通过上机调试程序，并变换不同的语句来实现同一功能以加深理解其中的奥妙。注意通过例题掌握类的存取权限，体会类的封装性及其提供接口的方法。

2.2　重点与难点分析

本章的重点是学会初步建立类并使用类的对象；难点是理解 C++语言中的 this 指针，类的抽象和封装的概念并真正认识私有成员的保护性。因此，本节将把重点放在类的组成及其存取权限上，以便为学习类的兼容性规则打下基础。

本章不介绍使用另一个类的对象成员组成新类的方法，而把它放到第三章集中介绍。

【例 2.1】 分析下面程序存在的问题及输出结果。

```
#include <iostream.h>
class Location
{
    public:
```

```
        int x,y;
        void Init(int a,int b){x=a;y=b;};
        int Getx(){return x;}
        int Gety(){return y;}
        int Setx(int a){x=a;}
        int Sety(int b){y=b;}
};

void main()
{
  Location A;
  A.Setx(5);
  A.y=89;
  cout<<"x="<<A.x<<endl;
  cout<<"y="<<A.y<<endl;
}
```

在 BC31 中，这个程序中的 Setx(int)和 Sety(int)函数虽然可以通过编译，但会给出警告，因为它们均没有返回值，也就是没有 return 语句。可以让它返回 0，也可以将函数定义为 void 类型。

```
    int Setx(int a){x=a; return 0 ;}
    void Setx(int a){x=a;}
```

一般采用后者。

程序输出如下：

```
    x=5
    y=89
```

如果使用 VC，则上述程序不能通过编译，修改方法同上。

如果把上面的程序改为用 struct 定义。则有：

```
#include <iostream.h>
struct Location
{
    public:
        int x,y;
        void Init(int a,int b){x=a;y=b;};
        int Getx(){return x;}
        int Gety(){return y;}
        void Setx(int a){x=a;}
        void Sety(int b){y=b;}
};
```