普通高等教育“十三五”创新型规划教材
理论+实践+数字资源一体化规划教材

紧扣教学大纲，突出重点
强化应用能力，迁移拓展
支持教学做考，立体资源

MIANXIANG DUIXIANG CHENGXUSHEJI

面向对象程序设计

主编　王瑞平

电子科技大学出版社

面向对象程序设计

主　编：王瑞平
副主编：高　原　李　源

图书在版编目(CIP)数据

面向对象程序设计 / 王瑞平主编. —成都：电子科技大学出版社，2016.3

ISBN 978-7-5647-3507-4

Ⅰ.①面… Ⅱ.①王… Ⅲ.①C语言—程序设计—高等学校—教材 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2016)第052698号

内容简介

面向对象程序设计是当前程序设计的主流，C++语言语法简洁，运行高效，具有明显的面向对象程序设计语言特征。本书是以学生已经学习过C语言为基础，主要介绍C++语言面向对象设计部分内容。本书首先介绍了C++语言相对于C语言新增的面向过程部分的内容，如C++中输入/输出、函数重载、new/delete操作符、引用等知识，然后重点介绍了C++中面向对象程序设计部分的基本知识，对象和类、类的特殊成员、继承与多态、运算符重载、I/O流和文件操作、模板及异常等知识。本书具有内容涵盖面广，项目实例丰富，便于学习的特点。

本书适用于大学计算机专业和非计算机专业的面向对象程序设计课程教学，也可供自学的读者使用。

面向对象程序设计

主　编　王瑞平

出　　版：电子科技大学出版社(成都市一环路东一段159号电子信息产业大厦　邮编：610051)

策划编辑：杜　倩

责任编辑：李述娜

主　　页：www.uestcp.com.cn

电子邮箱：uestcp@uestcp.com.cn

发　　行：新华书店经销

印　　刷：天津市蓟县宏图印务有限公司

成品尺寸：203mm×260mm　**印张** 15.5　**字数** 380千字

版　　次：2016年5月第一版

印　　次：2016年5月第一次印刷

书　　号：ISBN 978-7-5647-3507-4

定　　价：35.00元

内容简介

面向对象程序设计是当前程序设计的主流，C++语言语法简洁，运行高效，具有明显的面向对象程序设计语言特征。本书是以学生已经学习过C语言为基础，主要介绍C++语言面向对象设计部分内容。本书首先介绍了C++语言相对于C语言新增的面向过程部分的内容，如C++中输入输出、函数重载、new/delete操作符、引用等知识，然后重点介绍了C++中面向对象程序设计部分的基本知识，对象和类、类的特殊成员、继承与多态、运算符重载、I/O流和文件操作、模板及异常等知识。本书具有内容涵盖面广，项目实例丰富，便于学习的特点。

本书适用于大学计算机专业和非计算机专业的面向对象程序设计课程教学，也可供自学的读者使用。

前　言

面向对象程序设计课程是计算机类专业的一门重要的专业课，通过课程学习使学生建立面向对象程序设计的基本思想，能够灵活运用面向对象程序设计的知识进行程序设计。本教材使用C++作为面向对象程序设计的语言基础，全面介绍面向对象设计的思想和技术。本书采用知识项目内容组织全书，每个项目内容由任务贯串知识点，项目完成后提供习题练习和项目实训，并为读者提供了相关的网上链接参考内容。项目内采用“任务导入——知识储备——任务实施”模式，首先通过任务提出问题，然后介绍完成任务所需知识内容，在知识内容中除了知识讲解外还提供大量的例题，最后完成任务实施。

C++语言具有语法简洁，执行高效，支持面向对象设计的优点。本书以学生已经开设过C语言程序设计为基础，直接讲解C++语言相对于C语言新增过程设计的知识和全部的面向对象知识内容。项目1首先讲解面向对象设计的基本概念，然后介绍C++相对于C语言新增和改变的内容，如输入输出、内联函数、重载函数、new/delete操作符和引用等，这是后面项目实现的基础。项目2、项目3、项目4开始讲解面向对象的基础——类与对象，实现面向对象的第一大特征：封装。数据和操作封装在对象中，类是对象的抽象描述。其内容包括类的定义、对象使用、构造函数、析构函数、拷贝构造函数、静态成员等知识。项目5讲解继承与多态，实现面向对象的第二、三大特征：继承和多态，是面向对象设计的重要内容。项目6介绍C++中运算符重载，使用运算符重载可以提高程序的可读性。项目7介绍如何用流的概念理解输入输出。项目8介绍模板，模板是对类和函数的进一步抽象，具有更高的广泛性。运算符重载、I/O流和模板是面向对象设计抽象性的深入体现。项目9介绍了异常，使用异常处理机制可

以使程序设计更加完善，为大型程序的运行提供安全保证。全书内容丰富，知识点明确、举例适当，适合教师讲解，学生学习和掌握。

本书由王瑞平主编，共9个项目。其中项目2、项目3、项目9由王瑞平编写；项目1、项目6、项目8由李源编写；项目4、项目5、项目7由高原编写。全书由王瑞平最后审阅统稿。

由于学识水平和时间的限制，疏漏和不妥之处在所难免，敬请批评指正。

编　者

目　录

CONTENTS

项目 1　C＋＋基础知识 …… 1

1.1　从 C 到 C＋＋ …… 2
1.2　C＋＋数据的输入与输出 …… 7
1.3　函数 …… 13
1.4　const 常量 …… 20
1.5　堆内存分配 …… 22
1.6　引用 …… 24

项目 2　类与对象 …… 39

2.1　类与对象的概念 …… 40
2.2　类的定义 …… 42
2.3　对象 …… 51
2.4　构造函数 …… 54
2.5　析构函数 …… 68
2.6　const 成员 …… 72

项目 3　拷贝构造函数 …… 85

3.1　拷贝构造函数概述 …… 86
3.2　默认拷贝构造函数 …… 91
3.3　浅拷贝与深拷贝 …… 93

项目 4　静态成员和友元 …… 99

4.1　静态成员 …… 100
4.2　友元 …… 105

项目 5　继承与多态 …… 118

5.1　继承 …… 119
5.2　多态 …… 131

项目 6　运算符重载 …… 156

6.1　重载运算符 …… 157
6.2　重载单目运算符 …… 164
6.3　转换运算符重载 …… 170
6.4　重载赋值运算符 …… 172

项目 7　I/O 流 …… 180

7.1　标准 I/O …… 182
7.2　文件 I/O …… 189
7.3　串 I/O …… 194

项目 8　模板 …… 205

8.1　函数模板 …… 207
8.2　类模板 …… 209

项目 9　异常 …… 223

9.1　异常概述 …… 224
9.2　异常处理机制 …… 226
9.3　异常类 …… 229

参考文献 …… 240

项目 1　C++基础知识

学习目标

知识目标

1.了解 C++程序的组成部分、命名空间、C++中 I/O 流的使用

2.理解内联函数、函数重载、默认参数函数、new 堆内存分配及引用的概念

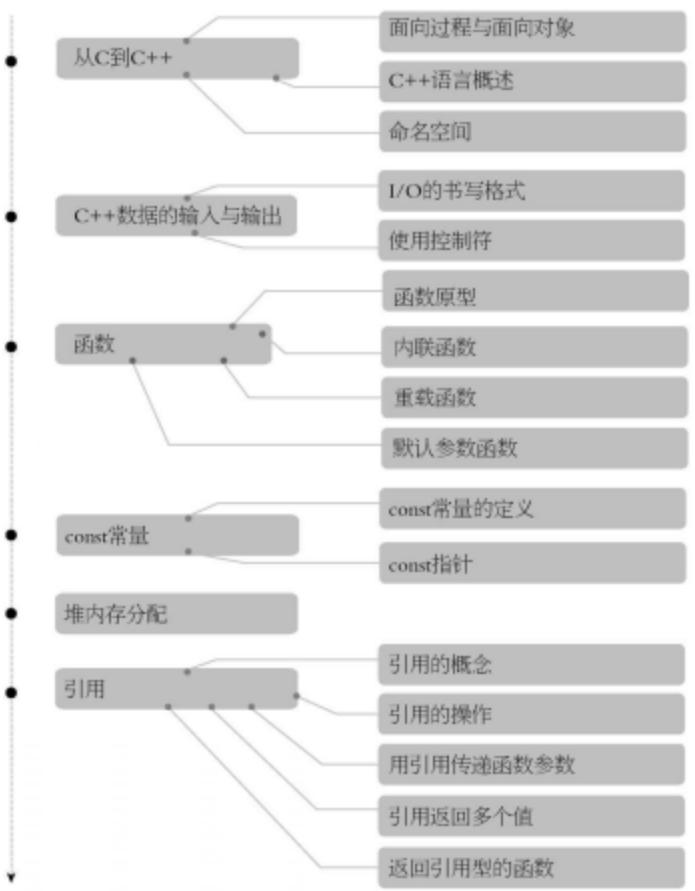
3.区别常量指针与指针常量

能力目标

1.掌握内联函数、函数重载、默认参数函数定义

2.掌握 new 动态内存分配和 delete 释放内存方法的使用

3.掌握引用及函数的引用参数和返回引用的概念和使用



C++语言不但几乎保留了C语言的全部特征，而且在C语言的基础上进行了很好的扩展。本项目将介绍从C到C++、C++数据的输入与输出、函数原型、内联函数、重载函数、默认参数函数、const常量、堆内存分配及引用等知识的概念及使用。

1.1 从C到C++

任务导入

C++语言对C语言的扩展主要体现在C++语言增加了面向对象的概念。C语言属于面向过程程序设计，C++语言属于面向对象程序设计，因此在学习C++语言前，需要了解面向对象的一些基本概念，C++语言的发展史以及C++语言中用于解决命名冲突而引入的命名空间的概念等，那么如何编写一个最简单的C++语言的程序呢？

知识储备

1.1.1 面向过程与面向对象

1.面向过程程序设计概念

面向对象出现以前，面向过程程序设计是程序设计的主流。面向过程(Procedure Oriented)是一种以过程为中心，结构化的编程思想。结构化程序设计(structured programming)是进行以模块功能和处理过程设计为主的设计方法，是过程式程序设计的一个子集，它对写入的程序使用逻辑结构，使得理解和修改更有效、更容易。

结构化程序设计概念最早由E.W.Dijkstra在1965年提出，是软件发展的一个重要的里程碑。它的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。

(1)结构化程序设计方法。结构化程序设计方法主要采用自顶向下、逐步细化，模块化设计及结构化编码。

①自顶向下、逐步细化。程序设计时，应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标。对一些任务复杂以至无法描述的问题，不要一开始就过多追求众多的细节，先从最上层总目标开始设计，然后将问题拆分为一系列较小的功能部件，逐步使问题具体化。

②模块化。一个复杂问题，肯定是由若干稍简单的问题构成。模块化是把程序要解决的总目标分解为子目标，再进一步分解为具体的小目标，每一个小目标称为一个模块。

③结构化编码。为解决的问题设计好一个结构化的算法之后，还要善于进行结构化编码，即正确地使用高级语言中顺序、选择、循环三种基本控制结构构造进行编码。

(2)结构化程序设计特点。结构化程序中的任意基本结构都具有唯一入口和唯一出口，并且程序不会出现死循环。在程序的静态形式与动态执行流程之间具有良好的对应关系。

①结构化程序设计优点。由于模块相互独立,因此在设计其中一个模块时,不会受到其它模块的牵连,因而可将原来较为复杂的问题化简为一系列简单模块的设计。模块的独立性还为扩充已有的系统、建立新系统带来了不少的方便,因为我们可以充分利用现有的模块作积木式的扩展。按照结构化程序设计的观点,任何算法功能都可以通过由程序模块组成的三种基本程序结构的组合,即顺序结构、选择结构和循环结构来实现。

结构化程序设计的基本思想是采用"自顶向下,逐步求精"的程序设计方法和"单入口单出口"的控制结构。自顶向下、逐步求精的程序设计方法从问题本身开始,经过逐步细化,将解决问题的步骤分解为由基本程序结构模块组成的结构化程序框图;"单入口单出口"的思想认为一个复杂的程序,如果它仅是由顺序、选择和循环三种基本程序结构通过组合、嵌套构成,那么这个新构造的程序一定是一个单入口单出口的程序。据此就很容易编写出结构良好、易于调试的程序来。

②结构化程序设计缺点。用户的需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;用系统开发每个阶段的成果来进行控制,不能适应事物变化的要求;系统的开发周期长。

2.面向对象程序设计

面向对象程序设计的思路和人们日常生活中处理问题的思路是相似的。在自然世界和人类社会生活中,一个复杂的事物总是由许多部分组成。例如,一部电话由听筒、机壳、键盘、送收话筒及其内部电路组成。

面向对象程序设计是一种适用于设计、开发各类软件的范型。它是将软件看成是一个由对象组成的社会。这些对象具有足够的智能,能理解从其他对象接受的信息,并以适当的行为作出响应;允许低层对象从高层对象继承属性和行为。通过这样的设计思想和方法,将所模拟的现实世界中的事物直接映射到软件系统的解空间。与传统的结构化程序设计相比,面向对象程序设计吸取了结构化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:

第一,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,避免了结构化程序由于数据和过程分离引起的弊病。

第二,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。而结构化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。

面向对象程序设计的三大特性是:封装、继承和多态。

(1)封装。封装是指将数据与具体操作的实现代码放在某个对象内部,使这些代码的实现细节不被外界发现,外界只能通过接口使用该对象,而不能通过任何形式修改对象内部实现,正是由于封装机制,程序在使用某一对象时不需要关心该对象的数据结构细节及实现操作的方法。使用封装能隐藏对象实现细节,使代码更易维护,同时因为不能直接调用、修改对象内部的私有信息,在一定程度上保证了系统安全性。

(2)继承。继承来源于现实世界,一个最简单的例子就是孩子会具有父母的一些特征,即每个孩子都会继承父亲或者母亲的某些特征,当然这只是最基本的继承关系,现实世界中还存在着更复杂的继承。面向对象的继承机制主要是用于实现代码的复用,多个类所公用的代码部分可以只在一个类中提供,而其他类只需要继承即可。

(3)多态。多态与继承联系紧密,是面向对象编程中另一个突出的特征,所谓的多态是

指在继承体系中,所有派生类都从基类继承接口,但由于每个派生类都是独立的实体,因此在接收同一消息的时候,可能会生成不同的响应。多态的作用作为隐藏代码实现细节,使得代码能够模块化;扩展代码模块,实现接口重用。简单来说:一种行为产生多种效果。

总的来说:封装可以隐藏实现细节同时包含私有成员,使得代码模块化并增加安全指数;继承可以扩展已存在的模块,目的是代码重用;多态则是为了保证类在继承和派生的时候,家谱中任何类的实例被正确调用,实现接口重用。

1.1.2 C++语言概述

1.C++的起源

C++语言是在C语言的基础上扩充形成的一种语言,而C语言的历史可追溯到1967年的BCPL语言。由剑桥大学的Martin Richards为编写操作系统软件和编译程序开发了BCPL语言(Basic Combined Programming Language)。1970年,Ken Thompson在继承BCPL语言的许多优点的基础上开发了实用的B语言。1972年,贝尔实验室的Dennis Ritchie在B语言的基础上,做了进一步的充实和完善,开发出了C语言。

C语言具有许多优点,比如语言简洁灵活,运算符和数据类型丰富,具有结构化控制语句,程序执行效率高,同时具有高级语言和汇编语言的优点等。与其他高级语言相比,C语言具有可以直接访问物理地址的优点,与汇编语言相比又具有良好的可读性和可移植性。因此,C语言得到了极为广泛的应用。

随着C语言应用的推广,C语言存在的一些缺陷或不足也开始暴露出来,并受到大家的关注。比如C语言对数据类型检查的机制比较弱,缺少支持代码重用的结构;随着软件工程规模的扩大,难以适应开发特大型程序。同时,C语言毕竟是一种面向过程的编程语言,已经不能满足运用面向对象的方法开发软件的需要。

为克服C语言本身存在的缺点,同时为支持面向对象的程序设计,1980年贝尔实验室的Bjame Stroustrup在C语言基础上创建、研制出来了一种通用的程序设计语言C++。

研制C++的一个重要目标是使C++首先是一个更好的C,所以C++根除了C中存在的问题。C++的另一个重要目标就是面向对象的程序设计,因此在C++中引入了类的机制。最初的C++被称为"带类的C",1983年正式命名为C++(C Plus Plus)。以后经过不断完善,形成了目前的C++。

当前运用得较为广泛的C++有Microsoft公司的Visual C++(简称VC++)和Borland公司的Borland C++ (简称BC++)。

C++是通往现代面向对象程序设计的途径。对于开发高性能的软件,C++是一种卓越的程序设计语言,同时,在全世界范围内,它也是程序员的首选语言。简而言之,要想成为一名专业的程序员,你应该在C++上有所造诣。C++并不只是一种流行的程序设计语言,它还为其他几种程序设计语言以及许多现代计算机思想提供了理论基础。C#和Java这两种重要的语言都来源于C++,这是有一定理论依据的。C++的语法、风格以及设计思想在很多方面都影响着现代程序设计。

C++语言是一种既可面向对象又可面向过程的混合型程序设计语言。它既适合于编写系统软件,也适合于编写应用软件。所以C++语言即全面兼容C语言,具有面向过程的

特点，又支持面向对象的程序设计方法。具体归纳如下：

(1)C++是一个更好的C，它保持了C语言的优点，大多数的C程序代码略作修改或不做修改就可在C++的集成环境下调试和运行。这对于继承和开发当前已在广泛使用的软件是非常重要的，可以节省大量的人力和物力。

(2)C++是一种面向对象的程序设计语言。它使得程序的各个模块的独立性更强，程序的可读性和可移植性更强，程序代码的结构更加合理，程序的扩充性更强。这对于设计、编制和调试一些大型的软件尤为重要。

(3)C++集成环境不仅支持C++程序的编译和调试，而且也支持C程序的编译和调试。通常，C++集成环境约定：当源程序文件的扩展名为.c时，则为C程序；而当源程序文件的扩展名为.cpp时，则为C++程序。

(4)C++的语句非常简练，对语法限止比较宽松，因此C++语法非常灵活。其优点是给用户编程带来书写上的方便。其缺点是由于编译时对语法限制比较宽松，许多逻辑上的错误不容易被发现，给用户编程增加了难度。

1.1.3　命名空间

使用命名空间的目的是对标识符的名称进行本地化，以避免命名冲突。在C++中，变量、函数和类都是大量存在的。如果没有命名空间，这些变量、函数、类的名称将都存在于全局命名空间中，会导致很多冲突。比如，如果我们在自己的程序中定义了一个函数toupper( )，这将重写标准库中的toupper( )函数，这是因为这两个函数都是位于全局命名空间中的。命名冲突还会发生在一个程序中使用两个或者更多的第三方库的情况中。此时，很有可能，其中一个库中的名称和另外一个库中的名称是相同的，这样就冲突了。这种情况会经常发生在类的名称上。比如，我们在自己的程序中定义了一个Stack类，而我们程序中使用的某个库中也可能定义了一个同名的类，此时名称就冲突了。

namespace关键字的出现就是针对这种问题的。由于这种机制对于声明于其中的名称都进行了本地化，就使得相同的名称可以在不同的上下文中使用，而不会引起名称的冲突。在命名空间出现之前，整个C++库都是定义在全局命名空间中的(这当然也是唯一的命名空间)。引入命名空间后，C++库就被定义到自己的名称空间中了，称之为std。这样就减少了名称冲突的可能性。我们也可以在自己的程序中创建自己的命名空间，这样可以对我们认为可能导致冲突的名称进行本地化。

通常来说，命名空间是唯一识别的一套名字，这样当对象来自不同的地方但是名字相同的时候就不会含糊不清了。

C++标准程序库中的所有标识符都被定义于一个名为std的namespace中。

< iostream > 和 < iostream.h > 格式不一样，前者没有后缀，实际上，在编译器include文件夹里面可以看到，二者是两个文件，打开文件就会发现，里面的代码是不一样的。后缀为.h的头文件C++标准已经明确提出不支持了，早些的实现将标准库功能定义在全局空间里，声明在带.h后缀的头文件里，C++标准为了和C区别开，也为了正确使用命名空间，规定头文件不使用后缀.h。因此，当使用 < iostream.h > 时，相当于在C中调用库函数，使用的是全局命名空间，也就是早期的C++实现；当使用 < iostream > 的时候，该头文件没有定义全局命名空间，必须使用using namespace std，这样才能正确使用cout和cin等。

由于 namespace 的概念，使用 C++标准程序库的任何标识符时，可以有三种选择：

(1)直接指定标识符。例如 std::ostream 而不是 ostream。完整语句如下：

```
std::cout << std::hex << 3.4 << std::endl;
```

(2)使用 using 关键字。using std::cout; using std::endl; using std::cin; 以上程序可以写成：

```
cout << std::hex << 3.4 << endl;
```

(3)最方便的就是使用 using namespace std; 例如，

```
using namespace std;
```

这样命名空间 std 内定义的所有标识符都有效。就好像它们被声明为全局变量一样。那么以上语句可以如下写：

```
cout << hex << 3.4 << endl;
```

因为标准库非常的庞大，所以程序员在选择的类的名称或函数名时就很有可能和标准库中的某个名字相同。所以为了避免这种情况所造成的名字冲突，就把标准库中的一切都放在名字空间 std 中。但这又产生一个新问题：无数原有的 C++代码都依赖于使用了多年的伪标准库中的功能，它们都是在全局空间下的。所以就有了 < iostream.h > 和 < iostream > 等这样的头文件，一个是为了兼容以前的 C++代码，一个是为了支持新的标准。命名空间 std 封装的是标准程序库的名称，标准程序库为了和以前的头文件区别，一般不加".h"。

任务实施

为了让学生了解 C++程序结构，下面以一个最简单的 C++程序来分析 C++的程序结构。

【例 1.1】一个最简单的 C++程序，输出"C++ is Very Good!"。

```
#include < iostream >                    //标准输入输出头文件
using namespace std;                     //使用 std 命名空间
int main()
{
    cout< < "C++ is Very Good! \n";      //C++的输出语句
    return 0;
}
```

运行结果为：

```
C++ is Very Good
```

请读者分析一下，本程序与 C 语言程序有何不同？

(1)标准 C++规定，main()函数必须声明为 int 型，因此，在 main()的最后返回一个整型 0 值传给操作系统。

(2)C++不但兼容 C 语言中"/* ……. */"形式的注释，还新增加了以"//"开头的注释。"//"是单行注释，不能跨行。

(3)C++中用 cout 进行输出代替 C 语言中 printf()函数。cout 实际上是 C++系统预定义的对象名,称为输出流对象。"<<"运算符是一个"插入运算符",与 cout 联合使用,将运算符右边的值(即右操作数)插入到输出流中。

(4)使用 cout 需要使用头文件 iostream。#include 预处理命令与 C 语言中使用类似,用于把后面的头文件包含到当前文件中。C++标准要求,由系统提供的头文件不带后缀.h,程序员自己定义的头文件可以有.h。当然,C++程序可以使用 C 语言编译系统提供的带.h 的头文件。

(5)语句"using namespace std;"的意思是"使用命名空间 std"。

1.2　C++数据的输入与输出

任务导入

C++语言中提供的 I/O 流用于数据的输入输出,比 C 语言中用于输出、输入的 printf()函数和 scanf()要方便得多。那么我们如何利用 C++语言中的 I/O 流按不同的要求控制数据输出呢?

知识储备

在 C 语言中,输入和输出的功能是通过调用 scanf 函数和 printf 函数来实现的。C++语言没有提供专门的输入/输出语句,它是通过输入/输出流(简称 I/O 流)来实现输入/输出操作的。I/O 流不是 C++语言的一部分,而是 C++库的一部分。

1.2.1　I/O 的书写格式

I/O 流是输入或输出的一系列字节,cout 是输出流对象的名字,cin 是输入流对象的名字。有关流对象 cin、cout 和流运算符的定义等信息是存放在 C++的输入输出流库中的,因此如果在程序中使用 cin、cout 和流运算符,就必须使用预处理指令把头文件 iostream 包含到本文中:

```
#include < iostream >
```

当程序需要在屏幕上输出数据时,可以使用插入操作符"<<"向 cout 输出流中插入字符,例如:

```
cout<< "Hello C++ Program.\n";
```

当程序需要接收从键盘输入的数据时,可以使用提取操作符">>"从 cin 输入流中抽取字符,例如:

```
int a;
cin>> a;
```

一条 cout 语句可以分写成若干行。如:

```
cout<< "C++ is a very easy language."<< endl;
```

可以写成：

```
cout< < "C++ is"
    < <  "a very"
    < < "easy language."
    < < endl;
```

也可以写成多个 cout 语句，即：

```
cout< < "C++ is";
cout< < " a very";
cout< < " easy language.";
cout< < endl;
```

以上 3 种情况的输出均为：

C++ is a very easy language.

在用 cout 输出数据时，用户不必通知计算机按何种类型输出，系统会自动识别输出数据的类型。如：

```
int a = 5;float b = 3.14;char c = 'A';
cout< < a< < "< < b< < "< < c< < endl;
```

输出的结果为：

5 3.14 A

cin 可以和 cout 类似的方式调整行，假设 i,j,m,m 为已定义的四个 int 型变量，则对

```
cin> > i> > j> > m> > n;
```

可以写成：

```
cin> > i
    > > j
    > > m
    > > n;
```

也可以写成：

```
cin> > i;
cin> > j;
cin> > m;
cin> > n;
```

以上三种情况都可以从键盘输入：

12 23 34 45

也可以分多行输入数据：

12

23 34

45

在用 cin 输入时，cin 也可以自动识别变量的类型从输入流中抽取相应长度的字节。如：

```
int i;float f; char c;
cin> > i> > f> > c;
```

从键盘上输入数据后，系统能够根据变量的类型，分别对i,f,c给出一个整形、浮点型和字符型的数据。

1.2.2　使用控制符

使用cout和cin的默认格式有时并不能满足人们在输入输出时的一些特殊的要求，如在输出实数时规定字段宽度，只保留两位小数，数据向左或向右对齐等。C++提供了在输入输出流中使用的控制符，控制符在头文件iomanip中定义，因此如果使用了控制符，在程序的开头除了要加iostream头文件外，还要加iomanip头文件。常用控制符如表1-1所示。

表1-1　I/O流的常用控制符

控制符	描　述
dec	设置基数为10
hex	设置基数为16
oct	设置基数为8
setfill(c)	设置填充字符为c
setprecision(n)	设置显示小数精度为n位
setw(n)	设置域宽为n个字符
setiosflags(ios::fixed)	设置浮点数以固定的小数位数显示
setiosflags(ios::scientific)	设置浮点数以指数形式显示
setiosflags(ios::left)	输出数据左对齐
setiosflags(ios::right)	输出数据右对齐
setiosflags(ios::skipws)	忽略前导的空格
setiosflags(ios::uppercase)	输出的十六进制数以大写表示
setiosflags(ios::lowercase)	输出的十六进制数以小写表示
setiosflags(ios::showpos)	输出正数时给出“+”号

任务实施

使用C++语言提供的输出控制符控制数据按不同的格数输出。

1.控制浮点数值显示

单独使用setprecision(n)可以控制输出浮点数的数字个数，setprecision(n)与setiosflags(ios::fixed)联合使用，可以控制浮点数的小数部分的精度。不设置setprecision(n)时，系统默认浮点数输出6位有效数字。

【例1.2】 分析下列输出的数据。

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    double a = 123.12345679876;
```