普通高等教育“十三五”规划教材

C++ 程序设计

刘丽华　刘宏妮　主编

C++ CHENGXU SHEJI

化学工业出版社

普通高等教育“十三五”规划教材

C++程序设计

刘丽华　刘宏妮　主编

化学工业出版社
·北京·

在程序设计方法方面，C++既支持传统的面向过程的程序设计方法，也支持新的面向对象的程序设计方法。因为C++是一种混合语言，所以就使得它保持了与C语言的兼容，C程序员仅需学习C++语言的特征，就可很快地运用C++类编写程序。

全书共分8章。第1章为C++初步知识；第2章是类与对象；第3章是继承和多态；第4章介绍特殊成员函数；第5章介绍运算符重载；第6章是I/O流；第7章是模板；第8章介绍了异常处理。各章均附有与内容相对应的习题。

本书概念清楚，重点突出，使学生能对使用C++进行面向对象编程有一个完整的整体认识，并初步掌握实用程序的编制方法及大程序的设计方法，为课程设计打下基础。

可作为计算机及相关专业学生的教材，同时也适合作为社会上各种培训班的教材，并可供广大计算机工作者自学之用。

图书在版编目（CIP）数据

C++程序设计/刘丽华，刘宏妮主编. —北京：化学工业出版社，2016.6

普通高等教育“十三五”规划教材

ISBN 978-7-122-26901-0

Ⅰ.①C… Ⅱ. ①刘… ②刘… 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085998号

责任编辑：廉 静　　装帧设计：王晓宇

责任校对：宋 夏

出版发行：化学工业出版社（北京市东城区青年湖南街13号 邮政编码100011）

印　　刷：北京永鑫印刷有限责任公司

装　　订：三河市宇新装订厂

787mm×1092mm 1/16 印张10¼ 字数266千字 2016年8月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899

网　　址：http://www.cip.com.cn

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：28.00元

前言

C++语言是为了适应 20 世纪 90 年代开发和维护复杂的应用软件的需要而研制的。它的目标是为程序员的程序开发提供优良的程序设计环境，以便能产生模块化程度高、重用性和可维护性好的程序。同时，C++语言非常强调代码的有效性和紧凑性，它是程序员的语言，允许程序员决定如何实现特定的操作。因此，C++语言已经在各个领域得到了广泛应用，尤其适用于中型和大型的程序开发项目。许多事实已经证明，C++应用于 C 语言曾经使用过的所有场合，其效果比 C 语言要好得多，从开发时间、开发费用到形成软件的可重用性、可扩充性、可维护性和可靠性等方面，都显示出 C++的优越性。在程序设计方法方面，C++既支持传统的面向过程的程序设计方法，也支持新的面向对象的程序设计方法，因此，C++是一种混合语言。由于 C++的这种特性，就使得 C++保持与 C 语言兼容，从而使许多 C 语言代码不经修改就可以为 C++所用，用 C 语言编写的众多的库函数和实用软件也可以用于 C++中，从而方便了 C 语言用户向 C++的过渡。不过，用 C++编写的程序的可读性更好，代码结构更为合理，可以直接在程序中映射问题空间的结构。

本书的重点是强调面向对象的程序设计方法，涉及少量 C 语言的知识，所以也可以作为直接学习 C++的教材。第 1 章是 C++初步知识，重点是介绍面向对象的基本概念，并从 C++的观点出发，介绍许多 C 语言所没有的概念。第 2 章是类和对象，重点是介绍面向对象的程序设计知识及定义和使用类的方法。第 3 章是继承和多态，介绍单一继承、多重继承和虚基类，C++的多态性、虚函数、虚函数的多态性及虚析构函数。第 4 章是特殊成员函数，介绍各种常用成员函数的特征。第 5 章是运算符重载，介绍类运算符、友元运算符、重载。第 6 章是 I/O 流，介绍流类库及流应用。第 7 章是模板，模板是将来的发展趋势，所以本书也介绍了模板的基本概念，简要介绍函数模板、类模板、模板与继承的关系。第 8 章是异常处理，介绍了流的错误和处理。

本书的对象是计算机及相关专业的学生，注重培养独立解决问题的能力，概念清楚，重点突出，使学生能对使用 C++编程有一个完整的认识，并初步掌握实用程序的编制方法及大程序的设计方法，为课程设计打下基础。

各章除了附有精心挑选的按题型分类的习题之外，还给出了多选题及编程题，以便于概念的理解和编程能力的训练。

本书第 2、3、5、8 章由刘丽华老师编写，第 1、4、7 章由刘宏妮老师编写，第 6 章由关蕊老师编写，最后由刘丽华老师统稿。

由于水平有限，不妥之处在所难免，希望同行及读者指正。

编　者

2015 年 5 月于本溪

目录

第1章 C++初步知识

计算机软件开发一直被两大难题所困扰：一是如何超越程序复杂性障碍，二是如何在计算机系统中自然地表示客观世界，即对象模型。用 C++语言编写的面向对象程序设计是软件工程学中的结构化程序设计、模块化、数据抽象、信息隐藏、知识表示、并行处理等各种概念的积累与发展，在 20 世纪 90 年代是解决上述两大难题最有希望、最有前途的方法。

面向对象程序设计是软件开发方法的一场革命，它代表了新颖的计算机程序设计的思维方法。该方法与通常结构程序设计十分不同，它支持一种概念，即旨在使计算机问题的求解更接近人的思维活动，人们能够利用 C++语言充分挖掘硬件潜在能力，在减少开销的前提下，提供更强有力的软件开发工具。

面向对象程序设计是软件系统的设计与实现的新方法。这种新方法是通过增加软件可扩充性和可重用性，来改善并提高程序员的生产能力的，并能控制维护软件的复杂性和软件维护的开销。当使用面向对象程序设计方法时，软件开发的设计阶段更加紧密地与实现阶段相联系。在软件设计与实现中，当今有许许多多方法，面向对象方法是在实践中超越其他许多方法的潜在的大有前途的方法，并且在各个应用领域中面向对象程序设计都获得了巨大成功。

从目前现状来看，C++和面向对象程序设计，不仅在尖端技术应用领域（如金融和通信）已立稳脚跟，而且 C++已逐渐地为企业开发人员所接受。C++产品可以给用户一个很好的起跳点，用户可在此基础上按照接近人的思维活动，按不同的对象类向前进展，C++和面向对象程序设计在企业信息系统中占领一席之地是迟早之事。

面向对象方法包含了分析、设计和实现的面向对象方法，这部分是当今软件开发最薄弱的部分，面向对象方法对软件系统开发起着关键作用。本书不仅要在面向对象方法，而且还要在面向对象模型和设计方面加以介绍。面向对象模型和设计更好地加速了对问题需求的了解，使设计更加简洁清晰。特别是分析和设计过程中产生的高质量的产品，能极大地减少在开发后期发现的错误，并能显著地改善系统质量。

C++是目前最流行的面向对象程序设计语言。它在 C 语言的基础上进行了改进和扩充，增加了面向对象程序设计的功能，更适合编制复杂的大型软件系统。这一章将引入面向对象的概念，并通过一个简单的 C++程序来加深用户对面向对象程序设计方法的理解。

1.1 C++语言的起源和特点

正如从名字上可以猜测到的一样，C++语言是从C语言继承来的，但这种继承主要只是表现在语句形式、模块化程序设计等方面。如果从更重要的方面——概念和思想方面来看，C++源于早期的SIMULA语言，因为C++语言的最大特征是支持“面向对象的程序设计”（面向对象的程序设计的概念见1.2节）。SIMULA语言被广泛地用于系统仿真，设计它的主要目的是模仿现实世界的真实个体，而使用的主要手段是构造计算机领域的对象来表述现实的客体。由于SIMULA语言的应用领域并不十分广阔，更重要的一点是它缺乏强有力的开发工具的支持，它并没有得到很大的重视。随后推出的另外一种面向对象的语言SMALLTALK也没有取得太大的成功，很多人认为它没有提供给自己足够的灵活性和如同C或BASIC语言那样丰富的功能，原因关键还在于它和人们早已得心应手的语言并不兼容。比如说，一个C程序员可能会对它的新特性退避三舍，因为C的特性对他是十分熟悉和亲切的，同时C的确是功能强大的，大多数人不愿放弃这些。

C++的产生正是为了解开这样的一个“情结”。面对越来越大，越来越复杂的系统，使用C语言已经感到力不从心了，但C语言作为应用最为广泛的程序设计语言之一，又不能轻易放弃。必须有一种面向对象的程序设计语言，它对C语言有很高的兼容性，使得C程序员只需在原有的知识上进行一定的扩充，就能够方便地进行面向对象的程序设计。

1980年起，Bell实验室的Bjarne strouctrup博士及其同事开始为这个目标对C语言进行改进和扩充。由于这种被扩充和改进的C语言的大量特性与类（class）相关，它最初被开发者称为“带类的C”。但很快人们就认识到这个称呼太片面了，这个“扩展了的C”不仅以标准ANSI C作为子集保留了C语言的全部精华，同时又吸收了SIMULA 67，ALGOL 68和BCPL语言的许多特性，它已远远超过了C语言。随着这个语言的广泛应用和在各个领域取得成果的增多，它给程序设计带来了全新概念和表现出来远大的前景，它的开发者因此将C++这一名字赋予它。

C++是面向对象的程序设计语言，它与过去的面向过程的程序设计语言比较，C++的最大特征在于它是面向对象的程序设计语言。所谓对象是现实世界中的实体，例如桌子、电视接收机、张三等。具有共同行为和特征的实体的集合，可以被归纳成一类，因此每个对象都是属于某个类的对象，例如，人是一个类，而每个具体的人则是人这个类中的一个对象。面向对象的程序设计是程序设计的一种新思想，该思想认为程序是相互联系的离散对象的集合。面向对象的程序设计语言即是支持这种思想的程序设计语言。

1.2 什么是面向对象

在面向对象的程序设计方法出现之前，占据主流的是结构化程序设计方法。对于复杂的问题，结构化程序设计采用模块化、自顶向下逐步求精的设计原则，因此结构化的程序往往清晰、易读。典型的结构化程序设计语言有C语言、PASCAL语言等，著名的UNIX操作系统的大部分代码就是用C语言编写的。

随着软件技术的发展，需要开发的系统越来越复杂。人们逐渐发现，对于大型软件系统来说，如果采用结构化的设计方法，设计、编程、测试和维护等工作都非常困难，而且有许

多问题是结构化设计自身无法解决的。在这种背景下，产生了面向对象的方法，而面向对象的程序设计语言（Object-Oriented Programming Language，简称 OOPL）也应运而生。

那么，什么是面向对象的方法呢?

面向对象方法的出发点和基本原则，就是使分析、设计和实现一个系统的方法尽可能地接近我们认识一个系统的方法。形象一点来说，就是使得描述问题的问题空间和解决问题的方法空间在结构上尽可能地一致。这样说可能太抽象，但随着对 C++语言学习的深入，会逐步体会到这一点。

下面介绍面向对象方法的几个重要概念，作为后面学习 C++语言的基础。

对象（Object）：是由信息和对它进行处理的描述所组成的包，其结构如图 1-1 所示。

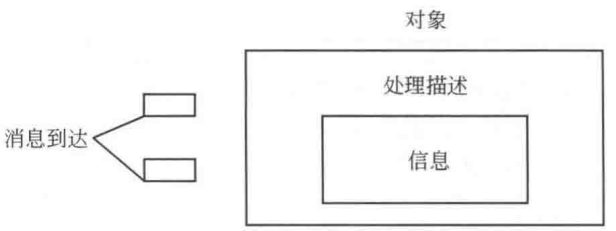

图 1-1　对象的图解

消息（Message）：是对某个对象进行处理的说明方法（Method）：是类似于过程的一个实体，是对对象接受了某一消息后所采取的一系列操作的描述。

类（Class）：是对具有共同特征的对象的描述。

类的封装（Encapsulation）：封装是把一类对象的状态（用数据表示）和方法（用函数来表示）封闭起来，装入对象体中，形成一个能动的实体。OOPL 的封装机制模仿现实生活中的封装技术，把一类对象的数据和函数封闭起来，并提供访问它们的机制。外界只有调用对象的共有成员函数才能和对象交换信息，这样就达到了封装的目的。

类的继承（Inheritance）：是指新的类继承原有类的全部数据、函数和访问机制，并可以增添新的数据、函数和访问机制。这样产生的新类叫子类或派生类，原来的类叫父类或基类，这种产生新类的方法叫类的派生，也叫类的继承。如“汽车”是一个类，“轿车”就是它的子类，而某辆实际的小轿车就是这个子类的一个对象。

多态性：是指相似而实质不同的操作可以有相同的名称。例如，“和”的操作，可以是“整数和”也可以是“矢量和”，在 C++中，这两种和的操作都可以简单地称为“和”。C++的多态性使得 C++与人的思维习惯更趋一致。用 C++编制的程序也更方便人的阅读。面向对象的方法还有许多特征和概念，如虚函数等，会在后面具体介绍。

1.3　C++对面向对象程序设计的支持

C 语言产生于 1972 年，最早用来编写 UNIX 操作系统，经过完善与改进后，它发展成为一种通用的计算机语言。1983 年进行标准化（ANSI C）后，其发展更为迅速，几乎所有操作系统都提供对 C 语言的支持。

1985 年，AT&T 的 Bell 实验室在 C 语言的基础上，吸收了 OOPL 的特点，形成了面向对象的程序设计语言 C++。

C++是一种灵活高效、可移植的面向对象程序设计语言。C++诞生以后，发展极其迅速，很多公司都研制了自己的 C++版本，并推出了多种 C++的集成开发环境。

C++支持基本的面向对象的概念，如对象、类、方法、消息、子类和继承性以及多态性等。表 1-1 给出了 C++对这些概念的命名约定。

表 1-1　C++的概念及命名约定

面向对象的概念	C++的命名的约定
对象	对象
类	类
方法	成员函数
实例变量	成员
消息	函数调用
子类	派生类/子类
继承	派生/继承

C++程序设计语言在 C 语言的基础上扩充了类、内联函数、运算符重载、变量类型、引用和自由存储管理运算符等语法。我们在后面通过简单的例子来了解 C++语言的特点，在第二章、第三章再系统地学习 C++语言的语法与编程方法。

1.4　C++语言与 C 语言的关系

C 语言也诞生在 ATLT 的 Bell 实验室，1972 年由 Dennis Ritchie 在 B 语言的基础上开发出来的一个高级语言，今天 C 语言的使用已遍及计算机的各个领域。

C 语言有以下几个显著的特点：

首先，它是一种结构化语言，要求一个程序由众多的函数组成，程序的逻辑结构由顺序、选择和循环三种基本结构组成，适宜于大型程序的模块化设计。

其次，它可以部分取代汇编语言，同时具有很高的可移植性，这使得 C 语言程序在保证支持不同硬件环境的前提下，具有较高的代码效率。

再次，它提供了丰富的数据类型和运算，具有较强的数据表达能力，因而在许多不同的场合广泛应用。

总之，C 语言反映了设计者追求高效、灵活的设计思想，它支持模块化程序设计，从而支持大规模软件开发的愿望。

C++语言保留了 C 语言设计者的良好愿望，并使得 C 语言语句成为 C++语言的一个子集。一般，用 C 语言编写的程序可直接在 C++编译器下编译。

1.4.1　C++语言与 C 语言的主要区别

首先，C++提出了类（class）的概念。类是数据和函数的集合，数据用来描述类所属对象的状态，函数用来描述此类对象的行为。例如，大学生代表在大学读书的一类人，即大学生是一个类，每个具体的大学生都是这个类中的对象。大学生这个类中的数据可以是学生的姓名、性别、年龄、学校、专业、入学时间等，描述此类对象的行为可以是入学、改换专业、毕业等。

C 语言中的结构只是数据的集合，这种结构也可在 C++语言中使用。不同的是 C++语言将 C 语言中的“结构”概念扩充成近似于上述“类”的概念，即 C++语言中的结构既可以有数据，也可以有函数。

C++语言沿用了 C 语言中的结构，概念上没有变化。下列关键字是 C++语言新增的：

class，private，protected，public，this，new，delete，friend，inline，virtual，volatile。

1.4.2 C++语言与C语言的细小区别

为较完整地讲述C++语言与C语言的关系，在此介绍它们之间的细微差别。

（1）C++语言在保留C语言原有注释方式的同时，增加了行注释。以“//”起始的，以换行符结束的部分是行注释。

（2）const关键字，用这个关键字修饰的标识符为常量。它的引入可以替代C语言中的符号常量定义。比较下面两个语句：

```
define Number 1
const Number=1
```

它们的功能相同，但后一语句在编译时进行类型检查。

（3）说明结构、联合和枚举变量时union和enum的使用，例如：

```
/*C 语言的说明*/
struct  semployee a0, b0;
union  uemployee a1, b1;
enum  eemployee a2, b2;
//C++语言的说明
semployee a0, b0;
uemployee a1, b1;
eemployee a2, b2;
```

不必在结构类型名、联合类型名和枚举类型名前加关键字struct，union，enum。

（4）变量的说明可以放在程序的任一位置上，例如：

```
for(int I =0; I <100; I++)
```

（5）提供了作用域运算符“::”，当其中一个变量被一个局部变量阻挡时，运用作用域运算符仍可以操作该全局变量。

（6）标准输入输出一般不再使用C语言的printf和scanf，而使用三个标准I/O流，它们是：cout（与标准输出设备相联）及cin（与标准输入设备相联）和cerr（与标准错误输出设备相联）。在微机上，各设备一般分别为显示器、键盘和显示器。“<<”和“>>”分别是定义为流的插入和提取操作，例如：

```
cout<<"welcome";
cin>>a;
cerr<<"There is an error"
//向显示器输出"welcome"
//从键盘取得数据至 a
//在显示器上立即显示错误信息
```

自从1983年AT&T的Bell实验室推出了它的C++标准之后，仅仅经历了9～10年，C++的应用已广泛地深入到计算机技术的各个领域，并且取得了很大的成功。面向对象的概念越来越多地被人们所应用，例如，新一代具有革命性的微机和工作站操作系统WINDOWS NT就大量运用了这一概念。

面对愈来愈复杂的系统，愈来愈大型的软件，面向对象的程序设计从一个新的角度使得问题空间和解题空间保持更为有效的一致性，使得计算机软件更加易于理解。作为一种成功的面向对象程序设计的语言，C++成为人们对于客观世界进行概括和准确描述的有效手段，C++使得软件开发者能方便地仿真客观世界。所以可以相信，今天的C++正如当年的C语言

一样，会给软件设计带来一次巨大进步，而它的应用会迅速地普及到计算机技术的各个领域，成为新的更为有效工具。

需要指出的是，C++只是作为面向对象的程序设计方法的一种实现，在当今还有其他的程序设计语言也能支持面向对象程序设计，例如面向对象的 PASCAL 语言及前面已经提到的 SMALL TALK 语言等。因为面向对象方法学的提出是程序设计的革命，很多的专家学者都在为使这种新思想的更完美实现而努力着，不仅仅是软件产生巨大的变革，计算机硬件也受到这个思想的影响而发生了变化。从对 C++语言的使用和与相似的语言比较中得到的体会，我们认为它将在面向对象的程序设计语言中成为主流。

1.5 输入/输出的认识

1.5.1 I/O 的书写格式

输入和输出并不是 C++语言的组成部分，它们是由 iostream.h 库支持的。在本节中，只介绍在开始编写程序时所必须了解的知识。输入是由终端接收信息，在标准库中用 cin、scanf( ) 表示。输出是向终端发送信息，在标准库中用 cout、printf( )表示。输出符“<<”用于将一个量指向标准输出。例如：

```
cout<<"The sum of 7+3="
cout<<7+3;
cout<<"\n";
```

双字符序列\n 表示换行符。当它被送向终端后，其后的信息将在屏幕的下一行从第一列开始显示。输出符“<<”是可以连写的，例如：

```
cout<<"The sum of 7+3="<<7+3<<"\n";
```

其中每一个输出符都将被顺序执行。为了提高程序的可读性，往往将一个紧缩了的输出语句分在几行书写。下例中的三行文本是一条输出语句。

```
cout<<"the sum of"
    <<v1<<"+"
    <<v2<<"="<<v1+v2<<"\n";
```

在程序中定义了这些变量。同样，变量也必须先声明后使用。有关变量的详细内容将在第 2 章的前一部分进行介绍。

同样，输出符由终端给出一串信息。例如，下面的程序将读入两个整数，比较其大小，并输出最大值。

程序 1-1 比较两个整数大小并输出最大值。

```
//**          LT1-1.cpp                    **
#include <iostream.h>
  void main( )
{
  int val1, val2, val;
  cout <<"请输入两个整型数：";
  cin>>val1>>val2;
  if (val1>val2)
   val=val1;
```

```
else
  val=val2;
cout <<val<<"是最大数";
}
```

当编译并执行后，程序输出如下（假定输入的数值为 17 和 124)：

```
请输入两个整型数：17 124
124 是最大数
```

输入的两个数据 17 和 124 之间有一个空格。空格符、制表符、换行符都被 C++认为是空白符。语句：

```
cin>>val11>>val2;
```

将正确地读入这两个数据，因为输入符“>>”会自动地忽略掉输入中包含的空白符。

cin 和 cout 的前置声明包含在 iostream.h 中。如果程序员忘记引用 iostream.h，则编译器将在引用 cin 和 cout 的每一处产生一个类型错误。由此可见，存放前置声明是头文件的一个基本用途。

cerr 是 iostream.h 中第三个被预先定义的函数。它处理基本错误信息，用于程序执行时出现的意外情况。下面的程序片断将防止程序员进行除零操作：

```
if(v2==0)
{
cerr  <<"\n error: attempt to divide by  zero";
return;
}
v3=v1/v2;
```

下面的程序将从标准设备依次输入字符，直到文件结束。它同时统计字符个数和行数。它的输出如下：

程序 1-2 输入字符并统计个数和行数

```
lineCount(行数)  characterCount(字符数)
```

程序的实现如下：

```
//**          LT1-2.cpp                  **
#include <iostream.h>
void main( )
{
char ch;
int lineCnt=0, charcnt=0;
while (cin.get(ch))
{
switch(ch)
{case '\t':
case 'U': break;
case '\n': ++lineCnt; break;
default:   ++charCnt; break;
}
}
cout<<lineCnt<<" "<<charcnt<<"\n";
}
```

get( )是 iostream.h 中的一个函数，它每次读入一个字符并将其赋给参数表中的参数（本例中是 ch)。两字符序列\t 表示制表字符。

switch 语句提供了一种对量进行条件测试的方法。若所测试的量与某一 case 语句后显示给出的量相等，则执行此 case 后的语句，否则执行 default 后的语句。每当读到一个换行符时，lineCnt 值加 1，读到一个不是空白、制表及换行的字符时 charcnt 值加 1。while 语句是一种构成循环的语句。当指定条件为真时，它重复执行一组语句。在本例中，switch 语句在 get( ) 函数未碰到文件尾标志时被重复执行。

1.5.2 控制符的使用

流的默认格式输出有时不能满足特殊的要求，如：

```
double average =9.40067;
cout <<average<<endl;
```

希望显示的是 9.40，即保留两位小数，可是却显示了 9.40067，默认显示了 6 位有效位。

用控制符（manipulators）可以对 I/O 流进行控制。控制符是在头文件 iomanip.h 中定义的对象。可以直接将控制符插入流中。常见的控制符如表 1-2 所示。

表 1-2 I/O 流的常用控制符

控制符	描述
dec	置基数为 10
hex	置基数为 16
oct	置基数为 8
setfill(c)	设填充字符为 c
setprecision(n)	设显示小数位数为 n 位
setw(n)	设域宽为 n 个字符
setiosflags(ios:: fixed)	固定的浮点显示
setiosflags(ios:: scientific)	指数表示
setiosflags(ios:: left)	左对齐
setiosflags(ios:: right)	右对齐
setiosflags(ios:: skipws)	忽略前导空白
setiosflags(ios:: uppercase)	16 进制数大写输出
setiosflags(ios:: lowercase)	16 进制数小写输出

使用控制符时，要在程序开始加入文件 iomanip.h。

（1）控制浮点数值显示

使用 setprecision(n)可以控制输出流显示浮点数的数字个数。C++默认的流输出数值有效位是 6。

如果 setprecision(n)和 setiosflags(ios:: fixed)合用，可以控制小数点右边的数字个数。setiosflags(ios:: fixed)是用定点方式表示实数的。

如果与 setiosflags(ios:: scientific) 合用，可以控制指数表示法的小数位数。setiosflags(ios:: scientific)是用指数方式表示实数的。

程序 1-3 下面的代码分别用浮点数、定点数和指数的方式表示一个实数。

```
//**          LT1-3.cpp                    **
#include<iostream.h>
#include <iomanip.h>
void main( )
```

```
{
double amount =22.0/7;
cout<<amount<<endl;
cout<<setprecision(0)<<amount<<endl
<<setprecision(1)<<amount<<endl
<<setprecision(2)<<amount<<endl
<<setprecision(3)<<amount<<endl
<<setprecision(4)<<amount<<endl;
cout<< setiosflags(ios:: fixed) <<endl;
cout<<setprecision(8)<<amount<<endl;
cout<< setiosflags(ios:: scientific)<<amount<<endl;
cout<<setprecision(6);
}
```

程序运行结果：

```
3.14286
3
3
3.1
3.14
3.143
3.14285714
3.14285714e+00
```

在用浮点数表示的输出中，setprecision(n)表示有效位数。

第一行输出数值之前没有设置有效位数，所以使用流的有效位数默认值 6。第二行设置了有效位数为 0，C++的最小有效位数是 1，所以作为有效位数为 1 来处理。第三到第六行按设置的有效位数输出。

在定点数表示的输出中，setprecision(n)表示小数位数。

第七行的输出是与 setiosflags(ios:: fixed)合用的。所以 setprecision(8)设置的是小数点后面的位数，而非全部数字个数。

在用指数形式输出时，setprecision(n)表示小数位数。

第八行的输出用 setiosflags(ios:: scientific)来表示指数的输出形式。其有效位沿用上次的设置值 8。

小数位数截断显示时，进行四舍五入处理。

（2）设置值的输出宽度

除了使用空格来强行控制输出间隔外，还可以用 setw(n)控制符。如果一个值需要比 set(n)确定的字符数更多的字符，则该值将使用它所需的所有字符。例如：

```
float amount=3.14159;
cout<<set(w)<<amount<<endl;
```

其运行结果为：3.14159。它并不按 4 为宽度进行输出，而是按实际宽度输出。

如果一个字符数比 setw(n)确定的字符个数更少，则在数字字符前显示空白。不同于其他控制符，setw(n)仅仅影响下一个数值输出，换句话说，使用 setw(n)设置的间隔方式并不保存其效力。例如：

```
cout <<setw(8)
     <<10
```

```
    <<20<<end1;
```

运行效果为：

```
    1020
```

运行结果，会在1020前面添加6个空格。整数20并没按照宽度8输出。setw( )的默认值为宽度0，即setw(0)，意思是按输出数值的表示宽度输出，所以20就紧挨10了，若要每一个数值都有宽度8，则每一个值都要设置：

```
cout<<setw(8)<<10
    <<setw(8)<<20<<end1;
```

（3）输出八进制和十六进制数

三个常用的控制符是hex，oct和dec。它们分别对应十六进制、八进制和十进制的显示。这三个控制符在iostream.h头文件中定义。

程序1-4 显示数字的十六、八、十进制。

```
//**          LT1-4.cpp                    **
#include <iostream.h>
void main( )
{
int number=1001;
cout<<"十进制数表示："<<dec<<number<<end1;
cout<<"十六进制数表示："<<hex<<number<<end1;
cout<<"八进制数表示："<<oct<<number<<end1;
}
```

运行结果为：

十进制数表示：1001

十六进制数表示：3e9

八进制数表示：1751

1001是一个十进制的数，不能把它理解为十六进制或八进制，因为它不是以0x或0开头的。但在输出时，流根据控制符进行过滤，使其按一定的进制来显示。

用头文件iomanip.h中的setiosflags(ios:: uppercase)可以控制十六进制数按大写输出。例如，上例中增加一个头文件，对十六进制进行大写控制，代码如下：

```
#include <iostream.h>
#include <iomanip.h>
void mian( )
{
int number=1001;
cout<<"十进制数表示："<<dec<<number<<end1;
cout<<"十六进制数表示："<<hex
    <<setiosflags(ios:: uppercase)<<number<<end1;
cout<<"八进制数表示："<<oct<<number<<end1;
}
```

运行结果为：

十进制数表示：1001

十六进制数表示：3E9

八进制数表示：1751