上海市精品课程

TURING 高等院校计算机教材系列

# C++程序设计
## 思想与方法

翁惠玉 编著

人民邮电出版社
POSTS & TELECOM PRESS

**图书在版编目（CIP）数据**

C++程序设计：思想与方法 / 翁惠玉编著 . —北京：人民邮电出版社，2008.8
（高等院校计算机教材系列）
ISBN 978-7-115-18309-5

Ⅰ. C… Ⅱ. 翁… Ⅲ. C语言－程序设计 . 高等学校－教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第086430号

## 内 容 提 要

本书以C++为语言环境，重点讲授程序设计的思想和方法，涉及过程化程序设计和面向对象程序设计。本书分为两大部分：第一部分主要介绍一些基本的程序设计思想、概念、技术、良好的程序设计风格以及过程化程序设计，包括数据类型、控制结构、数组、指针、数据封装、过程封装以及各种常用的算法；第二部分重点介绍面向对象的思想，包括类的设计与使用、运算符的重载、继承、多态性、输入/输出、异常处理、容器和迭代器等。

本书旨在使读者通过学习，并经过一定的训练和实践，能够掌握程序设计的方法，并具备良好的程序设计风格。本书可作为各大专院校计算机专业程序设计课程的教材，也可供从事计算机软件开发的科研人员作为参考资料。

高等院校计算机教材系列

**C++程序设计：思想与方法**

◆ 编　　著　翁惠玉
责任编辑　杨海玲

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京铭成印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：19.5
字数：461千字　2008年8月第1版
印数：1～4000册　2008年8月北京第1次印刷

ISBN 978-7-115-18309-5 /TP

定价：35.00元

读者服务热线：(010)88593802　印装质量热线：(010)67129223
反盗版热线：(010)67171154

# 前　言

程序设计是计算机专业十分重要的一门课程，是实践性非常强的一门课程，也应该是一门非常有趣、让学生很有成就感的课程。但在教学过程中，很多学生的反应是：听懂了，但不会做，以至于最后丧失了兴趣。我认为主要的问题是教学过程中过分重视程序设计语言本身，过分强调理解语言的语法，而没有把思路放在解决问题的方法上面。

本书是作者根据多年来在上海交通大学计算机系讲授“程序设计”课程的经验，参考了近年来国内外主要的程序设计教材编写而成的。本书全面介绍结构化程序设计和面向对象程序设计两方面内容，书中秉承以程序设计方法为主、程序设计语言为辅的思想，采用以问题求解引出知识点的方法，在介绍语言要素的同时，更多地强调编程思想。本书的目标是让学生先了解学习的目的，提高学习兴趣，最后能利用学到的知识解决某一应用领域的问题。

C++是业界非常流行的语言，它既支持过程化的程序设计，又支持面向对象的程序设计，可以很好地体现程序设计的思想和方法。因为本书的主旨是强调程序设计的思想，所以选择C++语言作为教学语言恰好服务于这个目标。但是，因为本书以介绍基本的程序设计思想、概念和方法为基础，强调算法、抽象等重要的程序设计技术，所以对于C++的某些特殊成分和技巧将不予重点介绍。

本书内容大体分为两大部分：第1章到第9章为第一部分，它们主要介绍一些基本的程序设计思想、概念、技术、良好的程序设计风格（书中涉及程序设计风格的内容均对其加了波浪线做突出处理）以及过程化程序设计，包括数据类型、控制结构、数据封装、过程封装以及各种常用的算法等；第10章到第16章为第二部分，重点介绍面向对象的思想，包括如何设计及实现一个类，如何利用组合和继承实现代码的重用，如何利用多态性使程序更加灵活，如何利用抽象类制定一些工具的规范，最后为了更好地与数据结构课程衔接，介绍了容器和迭代器的概念。

本书得以顺利地编写和出版首先要感谢上海交通大学计算机系的俞勇教授，是他的鼓励使我有勇气把我的这些经验提供给大家参考；我还要感谢上海交通大学电信学院程序设计课程组的各位老师，与他们经常在一起讨论使我不断加深了对程序设计的理解；我还要感谢可爱的学生们，是他们与我在课上和课后的互动使我了解他们的困惑，清楚他们学习的难点。

由于作者水平有限，本书可能存在很多不足，敬请读者批评指正。

# 目　录

试读结束：需要全本请在线购买： www.ertongbook.com

# 第 1 章 绪　论

自从第一台计算机问世以来，计算机技术发展得非常迅速，功能不断扩展，性能突飞猛进。特别是微型计算机的出现，使得计算机的应用从早期单纯的数学计算发展到处理各种媒体的信息。计算机本身也从象牙塔进入了千家万户。

计算机系统由硬件和软件两部分组成。硬件是计算机的物理构成，是计算机的物质基础；软件是计算机程序及相关文档，是计算机的灵魂。

## 1.1　计算机硬件

经典的计算机硬件结构是由计算机的鼻祖冯·诺依曼提出的，因此被称为冯·诺依曼体系结构。冯·诺依曼体系结构主要包括以下3个方面内容。

(1) 计算机的硬件由5大部分组成，即运算器、控制器、存储器、输入设备和输出设备，这些部分通过总线互相连接，如图1-1所示。在现代计算机系统中，运算器和控制器通常集成在一块称为CPU的芯片上。

(2) 数据的存储与运算采用二进制表示。

(3) 程序和数据一样，存放在存储器中。

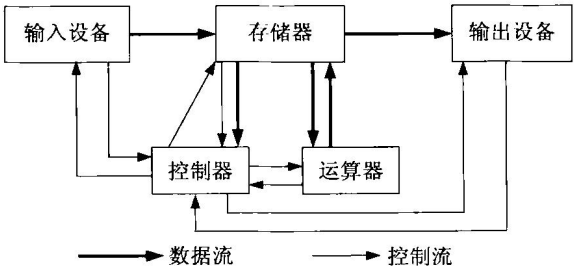


图1-1　计算机硬件系统的组成

运算器是真正执行计算的组件。它在控制器的控制下执行程序中的指令，完成算术运算、逻辑运算和移位运算等。不同厂商生产的机器，由于运算器的设计不同，能够完成的指令也不完全一样。每台计算机能完成的指令集合称为这台计算机的指令系统或机器语言。运算器由算术逻辑

单元（ALU）和寄存器组成。ALU完成相应的运算，寄存器用来暂存参加运算的数据和中间结果。

控制器用于协调机器其余部分的工作，是计算机的“神经中枢”。控制器依次读入程序的每条指令，分析指令，命令各其他部分共同完成指令要求的任务。控制器由程序计数器（PC）、指令寄存器（IR）、指令译码器（ID）、时序控制电路及微操作控制电路等组成。程序计数器用来对程序中的指令进行计数，使控制器能依次读取指令；指令寄存器暂存正在执行的指令；指令译码器用来识别指令的功能，分析指令的操作要求；时序控制电路用来生成时序信号，以协调在指令执行周期中各部件的工作；控制电路用来控制各种操作命令。

存储器用来存储数据和程序。存储器可分为主存储器和外存储器。主存储器又称为内存，用来存放正在运行的程序和数据，具有存取速度快，可直接与运算器、控制器交换信息等特点，但其容量一般不大，而且一旦断电，信息将全部丢失。外存储器（包括硬盘、光盘等）用来存放长期保存的数据，其特点是存储容量大、成本低，但它不能直接与运算器、控制器交换信息，需要时可成批地与内存交换信息。

存储器内最小的存储单元是比特（bit）。它可以存放二进制数的一位，即一个0或一个1。通常8个比特组成一个字节（byte，即B）。字节是大部分计算机分配存储器时的最小单位。存储器的容量也是用字节来表示的，如某台计算机的内存为1GB，则表明这台计算机的内存是1G字节。

输入/输出设备又称外围设备，它是外部和计算机交换信息的渠道。输入设备用于输入程序、数据、操作命令、图形、图像和声音等信息。常用的输入设备有键盘、鼠标、扫描仪、光笔及语音输入装置等。输出设备用于显示或打印程序、运算结果、文字、图形、图像等，也可以播放声音和视频等信息。常用的输出设备有显示器、打印机、绘图仪及声音播放装置等。

事实上，计算机的工作过程与我们日常生活中的工作过程完全一致。当要求你做一道四则运算题时，你会通过某个途径获取这道题并记录下来。例如，你会把这道题目抄到自己的本子上，那么本子就是主存储器，你的笔就是输入装置。要计算这道题目，你会根据先乘除后加减的原则找出里面的乘除部分，在草稿纸上计算，结果写回本子上，再执行加减得到最后结果写回本子。在此过程中，你的大脑就是CPU，先做乘除后做加减的过程就是程序，草稿纸就是ALU中的寄存器，把答案交给老师的过程就是输出过程。

## 1.2 计算机软件

计算机硬件是有形的实体，可以从商店里买到。但如果计算机只有硬件，那么，它只能成为一个装饰品。计算机之所以有魅力是因为它有七十二般变化，可以根据我们的要求变换不同的角色——一会儿是计算器，一会儿是字典，一会儿是CD播放机，一会儿又成了一架照相机。要做到这些，必须有各种软件的支持。硬件相当于计算机的“躯体”，而软件相当于计算机的“思想”和处理问题的能力。一个人可能面临各种要处理的问题，他必须学习相关的知识；计算机需要解决各种问题，它需要安装各种软件。

软件可以分为系统软件和应用软件。系统软件居于计算机系统中最靠近硬件的部分，它将计算机的用户与硬件隔离。系统软件与具体的应用无关，但其他软件都要通过系统软件才能发挥作

用。操作系统就是典型的系统软件。应用软件是为了支持某一应用而开发的软件，如字处理软件、财务软件等。

## 1.3 算法与程序设计

要让计算机能够完成某个任务，必须有相应的软件，而软件中最主要的部分就是程序。

程序设计就是教会计算机去完成某一特定的任务，即设计出完成某个任务的程序。它包括两个过程：第一步是设想计算机是如何一步一步地完成这个任务的，第二步是用计算机认识的语言描述这个完成任务的过程。前者称为*算法设计*，后者称为*编码*。计算机认识的语言就是*程序设计语言*，如C++语言。

算法设计就是要设计一个使用计算机提供的基本动作来解决某一问题的方案，是程序设计的灵魂。解决问题的方案要成为一个算法，必须用清楚的、明确的形式来表达，以使人们能够理解其中的每一个步骤，无二义性。算法中的每一个步骤必须有效，以使人们在实践中能够执行它们。例如，若某一算法包含“用π的确切值与$r$相乘”这样的操作，则这个方案就不是有效的，因为无法算出π的确切值。而且，算法不能无休止地运行下去，必须在有限的时间内给出一个答案。综上所述，算法必须具有以下3个特点。

(1) 表述清楚、明确，无二义性。

(2) 有效性，即每一个步骤都切实可行。

(3) 有限性，即可在有限步骤后得到结果。

有些问题非常简单，一下子就可以想到相应的算法，没有多大的麻烦就可写一个解决该问题的程序；而当问题变得很复杂时，就需要更多的思考才能想出解决它的算法。与所要解决的问题一样，各种算法的复杂性也千差万别。大多数情况下，一个特定的问题可以有多个不同的解决方案（即算法），在编写程序之前需要考虑许多潜在的解决方案，最终选择一个合适的方案。

算法可以用不同的方法表示。常用的有自然语言、传统的流程图、结构化流程图、伪代码和PAD图等方法。本书主要采用伪代码的方法。所谓的*伪代码*就是介于自然语言和程序设计语言之间的一种表示方法，通常采用程序设计语言的控制结构，用自然语言表示基本的处理。例如，要设计一个算法打印1到100之间的数的平方表，用伪代码表示如下：

```
for ( i=1; i<=100; ++i)
    输出i和i的平方;
```

一旦设计了一个算法，就可以用程序设计语言来描述它，这就是*编码*。为了让程序员在设计程序时更注重解决问题的算法，而不是某一台计算机上的指令系统，人们希望程序设计语言能独立于计算机系统，更贴近日常的表示。因此，程序设计语言通常都设计得类似于英语，且具有较强的数学计算能力。

## 1.4 程序的编译和调试

为了让用高级语言编写的程序能够在不同的计算机系统上运行，首先必须将程序翻译成该计

算机特有的机器语言。例如，若为Macintosh机器写了一个C++的程序，这将需要运行一个特殊的程序，该程序将C++语言写的程序转换成Macintosh的机器语言；如果在IBM的PC机上运行该程序，则需要使用另一种翻译程序。在高级语言和机器语言之间执行这种翻译任务的程序叫作编译器。

在大部分计算机系统上，运行程序之前需要先输入程序并将其保存在一个文件中。文件是存储在计算机外存中的信息集合的统称。每个文件都必须有一个文件名，通常用句点将文件名分成两部分，如myprog.cpp。前一部分由文件的创建者指定，通常选择一个能反映文件内容的字符串；后一部分称为扩展名，表示文件的类型，如扩展名“.c”表示文件的内容是C语言编写的程序，“.cpp”表示文件的内容是C++语言编写的程序。包含程序文本的文件称为源文件。

输入文件或修改文件内容的过程称为文件的编辑。各个计算机系统的编辑过程差异很大，不可能用一种统一的方式来描述，因此在编辑源文件之前，必须先熟悉所用的机器上的编辑方法。

一旦有了源文件，下一步就是使用编译器将源文件翻译成计算机可直接读懂的形式。这个过程也因机器而异，但在大多数情况下，编译器将源文件翻译成中间文件，这种中间文件称为目标文件，其中包含适用于特定计算机系统的实际指令，这个目标文件和其他目标文件一起组成可在系统上运行的可执行文件。这里所谓的其他目标文件常常是一些称为库的预定义的目标文件，库中含有许多完成常用操作的指令。将所有独立的目标文件组合成一个可执行文件的过程称为连接，这个过程见图1-2。

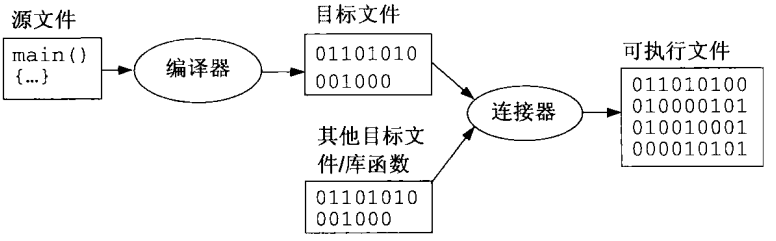


图1-2 编译过程

在编译过程中，编译器会找出源文件中的语法错误和词法错误。用户可根据编译器输出的出错信息来修改源文件，直到编译器生成了正确的目标代码。

语法错误还不是最令人沮丧的。往往程序运行失败不是因为编写的程序包含语法错误，而是程序合乎语法却给出了不正确的答案或者根本没给出答案。检查程序便会发现程序中有些逻辑错误，程序员称这种错误为bug。找出并改正这种逻辑错误的过程称为调试（debug），它是程序设计过程中一个重要的环节。调试一般需要运行程序，通过观察程序的阶段性结果来找出错误的位置和原因。

逻辑错误非常难以察觉。有时程序员非常确信程序的算法是正确的，但随后却发现它不能正确处理以前忽略了的一些情况；或者也许在程序的某个地方做了一个特殊的假定，但随后却忘记了；又或者可能犯了一个非常愚蠢的错误。

程序的调试及测试只能发现程序中的错误，而不能证明程序是正确的。因此，在程序的使用过程中可能会不断发现程序中的错误。在使用时发现错误并改正错误的过程称为程序的维护。

试读结束：需要全本请在线购买： www.ertongbook.com

## 小结

本章主要介绍了程序设计所需的下列基础知识和基本概念。

- 计算机系统包括软件和硬件：硬件是计算机的躯壳，软件是计算机的灵魂。
- 计算机硬件主要有5大部分组成：运算器、控制器、存储器、输入设备和输出设备。
- 程序设计包括算法设计、编码、编译、调试及运行维护5个阶段。

## 习题

1. 简述冯·诺依曼计算机的组成及工作过程。
2. 简述寄存器、主存储器和外存储器的异同点。
3. 所有的计算机能够执行的指令都是相同的吗？
4. 投入正式运行的程序就是完全正确的程序吗？
5. 为什么需要编译器？
6. 调试的作用是什么？

# 第 2 章

# 通过例子学习

## 2.1 第一个程序：输出 Hello world.

我们先从一个简单的程序开始。

代码清单2-1给出了一个完整的程序的结构。该程序主要由3个部分组成：程序注释、编译预处理命令和主程序。尽管这个程序结构非常简单，但它却是所有程序的典型，可以将它作为C++语言程序组织的范例。

**代码清单2-1** 输出Hello,world.的C++程序

```
//文件名: hello.cpp                                    } 程序注释
//在屏幕上显示 "Hello world."

#include <iostream>                                    } 编译预处理命令

int main()
{
     std::cout << "Hello world." << std::endl;         } 主程序
        return 0;
}
```

### 2.1.1 注释

hello.cpp程序的第一部分为一段注释，描述该程序用来做什么。在C++语言中，注释是从//开始到本行结束。在C++程序中也可以用C语言风格的注释，即在/*与*/之间所有的文字都是注释，可以是连续的几行。

注释是写给人看的，而不是写给计算机的。它们向其他程序员传递该程序的有关信息。C++语言编译器将程序转换为可由机器执行的形式时，注释被完全忽略。

一般来说，每个程序都以一个专门的、从整体描述程序操作过程的注释开头，称为*程序注释*。它包括源程序文件的名称和一些与程序操作有关的信息。程序注释还可以描述程序中特别复杂的部分，指出可能的使用者，给出如何改变程序行为的一些建议，等等。注释也可以出现在主程序中间，解释主程序中一些比较难理解的部分。

因为注释并不是真正可执行的部分，所以很多程序员往往不愿意写，但注释对将来程序的维护非常重要。给程序添加注释是良好的编程风格。

### 2.1.2 编译预处理

C++的编译分成两个阶段：预编译和编译。先执行预编译，再执行编译。预编译处理程序中的预编译命令,就是那些以#开头的指令。如代码清单2-1中的 #include <iostream>。常用的编译预处理命令主要有库包含和宏定义。

#### 1. 库包含

库包含表示程序使用了某个库。库是实用工具的集合，这些实用工具是由其他程序员编写的，能够完成特定的功能。iostream是C++提供的标准输入/输出库。程序中所有数据的输入/输出都由该库提供的功能完成。本书的每个程序都会用到这个库。

C++系统提供了许多标准库，完成各种常用的功能。程序员编写程序时，可以使用这些库提供的实用工具，从而省去自己编写这些实用工具的麻烦。库对于程序设计来说是十分重要的，当你开始编写一些较复杂的程序时，马上将会依赖一些重要的库。

要使用一个库就必须在程序中给出足够的信息，以便使编译器知道这个库里有哪些工具可用，这些工具又是如何使用的。大多数情况下，这些信息以头文件的形式提供。每个库都要提供一个头文件，这种文件为编译器提供了对库所提供的工具的描述，以便在程序中用到这些库的功能时编译器可以检查程序中的用法是否正确。#include命令的意思就是把iostream头文件加入到现在正在编写的程序中。iostream是一个头文件的名称，它主要定义了输入流cin和输出流cout对象。

#include有以下两种格式：

```
#include <文件名>
#include "文件名"
```

用尖括号标记的是系统的标准库，可以通过以下语句包含标准库iostream：

```
#include <iostream>
```

个人编写的库用引号标记。例如，某个程序员自己写了一个库user，于是#include行被写为

```
#include "user"
```

#### 2. 宏定义

宏定义用#define实现。宏包括不带参数的宏和带参数的宏。不带参数的宏通常用来定义符号常量。所谓的常量是指在程序执行过程中不变的值，如hello.cpp中的"Hello world"，而符号常量就是用符号表示的常量。带参数的宏用来定义一些较为复杂的操作。

不带参数的宏的定义格式如下：

```
#define  标识符   替换文本
```

当文件中出现这一行时，以后出现的所有该标识符都会在编译预处理时自动用替换文本取代。例如：

```
#define  PI  3.14159
```

在预编译时会用3.14159取代程序中出现的所有符号常量PI。可以用符号常量为某个常量建立一个名字，然后在整个程序中使用这个名字。如果需要修改整个程序中用到的该常量，可以在#define指令中做一次性的修改，然后再重新编译程序就会自动修改出现在程序中的所有这个常量。

在进行宏定义时，可以引用已经定义过的宏名。例如：

```
#define RADIUS 5
#define PI 3.14159
#define AREA  PI*RADIUS*RADIUS
```

经过编译预处理后，程序中的RADIUS全部替换成了5，PI全部替换成了3.14159，AREA替换成3.14159*5*5。

在C++中，符号常量一般用大写字母表示。

带参数的宏的处理方式较为复杂。不仅需要进行简单的字符串替换，还要进行参数的替换。带参数的宏的定义格式为

```
#define  宏名(参数表)  替换文本
```

在编译预处理时，首先用替换文本取代程序中的宏名，然后再进行参数的替换。例如，有一个宏定义为

```
#define  CIRCLE_AREA(x)    (PI* (x) * (x))
```

当程序中出现语句area = CIRCLE_AREA(4)，就会被替换成

```
area = (3.14159 * (4)* (4))
```

用#define定义符号常量和宏是C语言常用的方法。在C++中，这些功能有更好的解决方法。

### 2.1.3 主程序

代码清单2-1所示的文件hello.cpp的最后一部分是程序主体，由以下几行组成：

```
int main()
{
    std::cout << "Hello, world" << std::endl;
    return 0;
}
```

这5行是一个C++语言中函数的例子。函数由一系列独立的程序步骤组成，这些程序步骤集合在一起完成某一功能。每个函数有一个名字。代码清单2-1中的函数名为main。可以把函数看成数学中的函数。数学中的函数有函数名、自变量、函数表达式和函数结果值，C++中的函数也是如此。一个C++程序可以由一个或多个函数组成。在hello.cpp中只有一个函数，名字为main，该函数的执行结果值为一个整型数。main后面的圆括号中的内容是参数，即对应于数学函数中的自变量。空括号表示没有参数。函数表达式（即为所执行的步骤）列于花括号中，称为语句。这些语句共同组成函数的主体，称为函数体。hello.cpp中的main函数的函数体只有两条语句，但大多数函数都有几条连续执行的语句。

名字为main的函数在C++中有独特的作用。每个C++程序至少有一个函数。在组成程序的所

有函数中必须有一个名为main的函数。main函数是程序执行的入口。

当运行C++程序时，计算机从main函数主体开始执行语句，从第一条语句执行到最后一条语句。在main函数的语句中可能调用到其他函数。在hello.cpp中，main的主体由下面两条语句组成：

```
std::cout << "Hello, world" << std::endl;
return 0;
```

cout是标准的输出流对象，它是输入/输出流库的一部分。与cout相关联的设备是显示器。<<称为流插入运算符，表示将其后的数据插入到该流中。std::cout << "Hello, world" << std::endl;表示把Hello, world显示在显示器上，endl表示换行。双引号括起来的Hello, world被称为字符串常量。std是cout和endl所属的名字空间。::称为作用域限定符。return 0表示把0作为函数的执行结果，通常表示函数正常结束。

### 2.1.4 名字空间

大型的程序通常由很多源文件组成，每个源文件可能由不同的开发人员开发。开发人员可以自由地命名自己的源文件中的实体，如变量名、函数名等。这样很可能造成不同的源文件中有同样的名字。当这些源文件连接起来形成一个可执行文件时，就会造成重名。为了避免这种情况，C++引入了名字空间的概念，把一组程序实体组合在一起，构成一个作用域，称为名字空间。同一个名字空间中不能有重名，不同的名字空间中可以定义相同的实体名。引用某个实体时，需要加上名字空间的限定。

C++的标准库中的实体都是定义在名字空间std中的，因此引用标准库中的名字都要指出是名字空间std中的实体。例如，引用cout必须写成std::cout。

但这种表示方法非常繁琐，为此C++引入了一个使用名字空间的指令using namespace，它的格式如下：

```
using namespace 名字空间名;
```

一旦用了使用名字空间的指令，该名字空间中的所有实体在引用时就不需要再加名字空间的限定了。例如，代码清单2-1所示的程序也可以写成如下形式：

```
//文件名: hello.cpp
//该程序在屏幕上显示 "Hello world."

#include <iostream>
using namespace std;

int main()
{   cout << "Hello world." << endl;
    return 0;
}
```

## 2.2 程序示例：计算圆的面积和周长

为了使读者对C++程序的工作过程有更好的了解，下面来看一个复杂一点的例子，计算圆的

面积和周长。程序见代码清单2-2。

**代码清单2-2 计算圆的面积和周长的程序**

```
//文件名：circle.cpp
//计算圆的面积和周长

#define  PI 3.14159 //定义符号常量
#include <iostream>
using namespace std;

int main()
{   double radius, area, circum;          } 变量定义

    cout << "请输入圆的半径：" ;           } 输入阶段
    cin >> radius;

    area = PI * radius * radius;          } 计算阶段
    circum = 2 *  PI *  radius;

    cout << endl;
    cout << "圆的面积为：" << area << endl;   } 输出阶段
    cout << "圆的周长为：" << circum << endl;

    return 0;
}
```

circle.cpp用到了一些hello.cpp中没有的概念。除了在编译预处理部分增加了一个符号常量外，主要的区别是在main函数中。

在编译预处理命令中定义了常量PI。经过预编译后，main函数中的PI都被3.14159替换。如果该程序要进一步提高计算精度，可以将π的值取得更精确些。这只要修改一个地方就可以了，就是把原有的#define指令改成

```
#define  PI  3.1415926
```

这样，在计算圆的面积和周长时用的π都变成了3.1415926。

circle.cpp的main函数包含了C++函数体的完整内容。函数体分为变量定义部分和语句部分。语句部分又可分为输入阶段、计算阶段和输出阶段。一般各部分之间用一个空行隔开，以便于阅读。

变量（也称为对象）是一些在程序编写时值尚未确定的数据的存放处。例如，在编写求圆的面积和周长的程序时，用户要计算的圆的半径尚未可知，程序运行时用户才输入这个半径值。为了在程序中指明这些目前尚未确定值的对象，可创建一个变量来保存这些需要明确的值，每个变量都有自己的名字。一旦要用到它包含的值，就可使用变量名。变量的名字要精心选定，这样将来阅读程序的程序员就很容易分辨出每个变量的作用。在程序circle.cpp中，变量radius代表圆的半径，变量area和circum分别代表面积和周长。

在C++语言中，变量使用之前必须先定义。定义一个变量就是告知C++编译器创建了一个新的变量名，并具体指定了该变量可以保存的数据类型。例如，在程序circle.cpp中，

```
double radius, area, circum;
```

试读结束：需要全本请在线购买： www.ertongbook.com