TURING 图灵计算机科学丛书

PEARSON Addison Wesley

C++教程

C++ by Dissection: The Essentials of C++ Programming

[美] Ira Pohl 著
陈朔鹰 马锐 薛静锋 吕坤 译

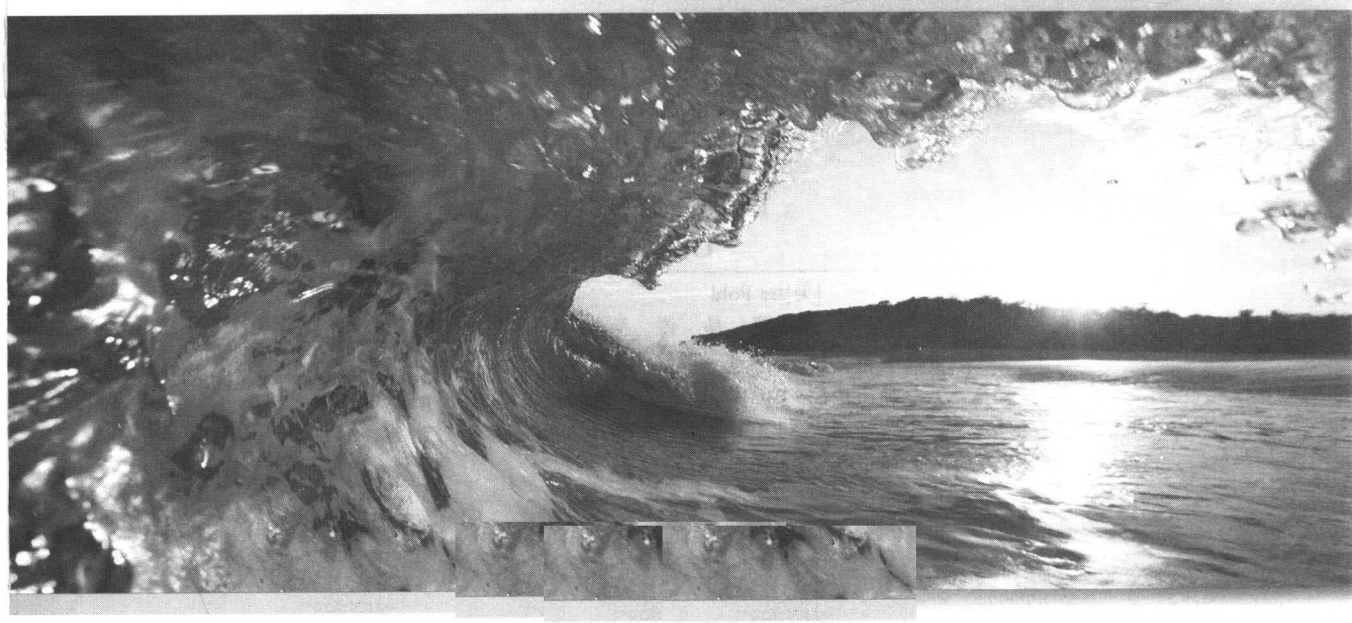
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵计算机科学丛书

C++教程

C++ by Dissection: The Essentials of C++ Programming

[美] Ira Pohl 著
陈朔鹰 马锐 薛静锋 吕坤 译



人民邮电出版社
北 京

图书在版编目（CIP）数据

C++教程 /（美）波尔（Pohl, I.）著；陈朔鹰等译.
—北京：人民邮电出版社，2007.12
（图灵计算机科学丛书）
ISBN 978-7-115-16708-8

Ⅰ.C… Ⅱ.①波…②陈… Ⅲ.C语言—程序设计—教材
Ⅳ.TP312

中国版本图书馆CIP数据核字（2007）第130774号

内 容 提 要

ANSI C++程序设计语言现在已经广泛应用于学术界和工业界，而且成为程序设计语言课程及计算机科学教学的首选语言之一。本书通过对C++程序设计语言关键特性的阐述和对具体C++程序示例的剖析，详细介绍了程序设计的过程。书中贯穿着对软件工程实践的描述，覆盖程序正确性和类型安全性，深入解释函数和指针，强调面向对象程序设计的概念，详细介绍泛型程序设计和STL，为了理解面向对象程序设计引入了UML。

本书适合作为计算机科学专业的第一门程序设计课程的教材。第一阶段指导学生如何编程，介绍C++程序设计语言，包括如何使用数组、指针及基于对象的程序设计基础；第二阶段介绍更高级的数据类型、面向对象程序设计、泛型程序设计与STL、文件处理和软件工程。此外，本书也适合没有编程经验的读者作为自学C++的参考书。

图灵计算机科学丛书

C++教程

◆ 著 [美]Ira Pohl
译 陈朔鹰 马 锐 薛静锋 吕 坤
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
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：22.75
字数：607千字 2007年12月第1版
印数：1－4 000册 2007年12月北京第1次印刷
著作权合同登记号 图字：01-2007-1968号

ISBN 978-7-115-16708-8/TP

定价：49.00元

读者服务热线：(010)88593802 印装质量热线：(010)67129223

译 者 序

C++是当今最为流行和实用的一门高级程序设计语言，它不仅保留了传统的结构化程序设计方法，而且对先进的面向对象程序设计方法也提供了完整的支持。随着ANSI和ISO C++标准相继制定完成，C++作为一种功能强大的面向对象程序设计语言，已经被全世界的程序员广泛使用。现在C++语言的身影几乎遍及所有计算机研究与应用领域。

本书通过程序示例和对C++程序设计语言关键特性的阐述，循序渐进地详细介绍了C++程序设计的过程。书中第1章~第5章介绍了C++程序设计语言的基础知识，包括C++语言的基本数据类型、语句、指针、数组以及基于对象的程序设计基础；第6章~第11章着重介绍了C++语言的面向对象特性，包括高级数据类型、面向对象程序设计、泛型程序设计和STL、文件处理以及软件工程。

本书最大的特色是采用解析方法阐述代码的关键特性，以易于理解的文字渐进式地向读者解释在代码中遇到的程序设计元素和方法，帮助读者更好地理解程序设计思想，逐步引导读者成为富有经验的程序员。另外，在本书的每一章还对C++语言和Java语言进行了比较，不仅可以帮助熟悉Java的程序员了解C++，那些已经熟悉C++，还想进一步学习Java的程序员也可以从中受益。

本书是多人共同努力的结果，参与本书翻译的人员有马锐、薛静锋、吕坤、陈朔鹰、刘瑾、江雪、彭一明、束罡和刘光宗等。全书由马锐统稿。

限于译者水平，译文难免有不当和疏漏之处，敬请读者不吝指正。

译 者

2007年5月

前　言

现在，ANSI C++程序设计语言已经广泛应用于全世界的学术界和工业界。在许多教育机构，它都是程序设计语言课程的首选语言，也是计算机科学教学的首选语言。究其原因，主要是因为C++是学习更高级课程所必需的基础课程；而且，C++拥有众多实用库，得到许多复杂集成环境的支持，它是一种能够有效地支持当今主流的面向对象程序设计方法的语言。

本书通过精心开发的可实际工作的程序来介绍程序设计过程，并以此阐明C++程序设计语言的关键特性。书中以易于理解的方式解释了程序代码，这些程序代码都在多个平台上进行了测试。书中的代码能够在大多数C++系统上使用，包括MacOS、MS-DOS、OS/2、UNIX和Windows等操作系统上的C++系统。

C++是由贝尔实验室的Bjarne Stroustrup在20世纪80年代中期发明的一种功能强大的现代语言；C++语言是C语言的后续语言，它在C语言的基础上增加了类的概念，提供了一种用户自定义类型（也叫抽象数据类型）的机制。通过自定义用户类型、继承和运行时类型绑定，C++提供了对面向对象程序设计的支持。

解析方法

本书通过精心开发的可实际工作的C++程序，利用解析（dissection）的方法，向读者清晰而详细地介绍了C++程序设计的过程。解析是作者在1984年首先开发的一种教学方法，用来阐述代码的关键特性。解析类似于对代码进行的初始结构化分析，其目的是向读者解释在代码中遇到的程序设计元素和方法。解析以易于理解的语言渐进式地解释程序和函数，可以加强读者对不同上下文中程序和函数的关键思想的理解。

无需任何程序设计背景

本书主要面向没有任何程序设计背景的读者，不熟悉C++的有编程经验的程序员也可以从中获益。本书可以作为学生学习计算机科学或程序设计的第一门课程的教材。

本书适用于计算机科学专业或其他学科的第一门程序设计课程。书中的每一章都给出了对大量程序的具体解释，以启发式方法引导学生提高程序设计技巧。本书首先介绍一个完整的程序，然后再介绍函数的编写，函数也是结构化程序设计的主要特点。函数和程序的关系类似于段落和文章的关系，编写完整的函数是程序员必须具备的能力，因此本书将其作为重点强调。书中有大量不同难度的示例和习题，教师可以从中选择合适的示例和习题进行讲解。

本书特色

本书有许多特色：

- 在网站上有本书原版的全部电子文档，也有指向有用站点的链接和本书完整的程序代码。
- 书中贯穿着对软件工程实践的描述。
- 每章都为初学者提供了作者建议的简明的编程技巧。
- 对简单的递归提前进行解释，这是计算机科学课程导论部分的内容。
- 覆盖了程序正确性和类型安全性。
- 深入解释了函数和指针，这些概念通常是初学者学习时遇到的最大障碍。
- 强调了面向对象程序设计的概念。
- 详细描述了泛型编程和STL。
- 为了理解面向对象编程引入了UML图。
- 与Java进行了比较，并配备了可选的Java习题，指出了*Java by Dissection*（与Charlie McDowell合著）一书中的对应内容。

章节特色

每章都包含了下列教学要素。

解析 通过解析的方法解释了重要示例程序的主要元素。这种对新的程序设计思想的渐进式讨论有助于第一次接触这些设计思想的读者更好地理解这些思想。

面向对象程序设计 逐步引导读者熟悉面向对象风格。第4章介绍了类：类是产生模块化程序和实现抽象数据类型的基本机制，类变量是被操纵的对象。第8章介绍了两个关键概念，即继承和虚函数。第11章讨论了面向对象程序设计原理。本书培养了程序员的面向对象程序设计观念。

编程风格和软件工程 本书通篇强调了编程风格和软件方法。本书一开始就介绍了一些重要的概念，例如结构化分支语句、嵌套控制流程、自顶向下设计方法和面向对象程序设计。本书一开始就采用了一致和合理的代码风格，并详细阐述了这种风格的重要性和原理。本书采用的代码风格也是C++专业编程人员最常用的风格。

可实际工作的代码 本书的程序代码都是可实际工作的，通过这些可执行的代码，学生能够更好地理解要讨论的程序设计思想。本书通过解析的方式解释了许多程序和函数，在习题中还经常涉及这些程序设计思想。

常见的编程错误 本书描述了许多典型的编程错误以及避免这些错误的方法。学习一门程序设计语言有障碍很大程度上是因为遇到了一些难以理解的错误，许多书籍只讨论了正确的程序代码，而让读者自己通过试错过程去发现程序设计中的漏洞。本书不但解释了C++中的典型错误是如何造成的，还解释了如何改正这些错误。

Pohl博士的建议 作者基于丰富的编程经验提出了一系列的编程技巧，对每一个技巧都简明指出了基本原理。

与Java的比较 在可选的章节中介绍了与C++实例进行对比的Java语言程序设计元素，

习题中也给出了相关内容。在大多数情况下，C++与Java具有同样的程序设计元素，本书可以帮助熟悉Java的学生学习如何从Java迁移到C++。那些熟悉C++的学生，如果想进一步学习Java，也可以从这些章节中受益。此外，想学习本书姊妹书*Java by Dissection*的读者，也可以通过本书熟悉我们讲解Java的方法。

小结 每章过后都有关于本章要点的简明列表，这可以帮助读者回顾这一章的内容，加深对这一章中提出的新观点的理解。

习题 习题用于检验学生对语言知识的掌握程度。许多习题的设计目标都是为了便于读者进行交互式学习，这样有助于读者自学。除了针对语言特性的习题之外，一些习题也针对某个主题给出了更多的细节，还有一些习题则引入了更深层次的知识。

课堂教学

本书可以作为教材使用。第一阶段指导学生如何编程，主要包括第1章~第5章的内容。这部分通过如何使用数组、指针以及基本的面向对象程序设计来介绍C++程序设计语言。第二阶段包括第6章~第11章的内容，介绍更高级的数据类型、面向对象程序设计、泛型程序设计和STL、文件处理和软件工程。如果课程是为已经有程序设计基础（不一定是C++语言程序设计基础）的学生设计的，教师可以介绍本书的所有主题。本书也可以作为需要学生使用C++的其他计算机科学课程的教材。在语言比较课中，本书可以与其关于C、Java和C#的姊妹书一同使用，它们采用同样的解析方法，用各自的语言编写了相同的示例。

交互式环境

本书是为交互式环境编写的，我们鼓励读者通过键盘和屏幕进行实验。许多PC厂商都支持交互式的C++系统，如Borland、IBM、Metroworks、微软和Symantec等。

专业人员

虽是为初学者而写，但本书同样可以成为经验丰富的程序员的良师益友。结合Al Kelley和Ira Pohl合著的*A Book on C*，计算机专业人员能够综合理解这两种语言。这两本书相辅相承，对C/C++程序设计语言进行了综合介绍，非常难得。此外，结合Ira Pohl和Charlie McDowell合著的*Java by Dissection*，学生和专业人员能够获得对面向对象语言Java的完全理解。

本书是作者讲授的许多在线专业培训课程的基础。从1986年起，作者就使用这些内容在各种各样的论坛上培训专业人员和学生。本书也是www.digitalthink.com提供的基于Web的C++培训的基础。

教辅材料

对使用本书作为教材的教师，Addison-Wesley 可以提供以下教辅材料：

- 习题解答。
- 示例程序代码。

• 含有所有插图的PowerPoint幻灯片。

要想了解更多关于教辅材料的信息，请访问www.aw.com/cssupport。

致谢

特别感谢Uwe F. Mayer、George Belotsky和Bruce Montague，他们仔细阅读了本书，提出了许多改进建议。感谢评审专家：科罗拉多州立大学的Charles Anderson，杨百翰大学的Parris Egbert，海军研究生院的Chris Eagle，路易斯安那州立大学的Nigel Gwee，佛罗里达州立大学的Stephen P. Leach和宾夕法尼亚州立大学的Steven C. Shaffer。感谢John dePillis、Debra Dolsberry和Laura Pohl，他们设计并绘制了许多卡通插图。特别要感谢Debra Dolsberry，她是本书许多材料的主要技术编辑，此外，她也负责使用FrameMaker对本书进行排版。感谢Charlie McDowell和Al Kelley编写了关于C和Java的姊妹篇。

感谢组稿编辑Maite Suarez-Rivas、项目编辑Katherine Harutunian和副总编Patty Mahtani，感谢她们的热心、支持和鼓励。感谢Argosy的Caroline Roop和Sally Boylan，感谢她们对于本书出版的悉心关注！

Ira Pohl

于加州大学圣克鲁兹分校

目　录

第1章

编写一个ANSI C++程序

本章带领读者进入ANSI C++程序设计世界。本章讨论一些基本的程序设计思想，详细地解释许多基本的程序。这里提出的基本思想是后续章节中更完整的解释的基础。本章的重点是C++的基本输入/输出函数，输入/输出信息是任何程序设计语言中都需要掌握的首要内容。

C++使用运算符<<和>>分别实现输出和输入，这里解释了这两个操作符的用法。本章中讨论的其他主题包括用于存储值的变量的用法、用于改变变量值的表达式和赋值语句的用法。

本章给出了许多示例，其中包括许多完整的程序以及对这些程序的解析，以便读者能够了解构造程序的细节。本章介绍的主题在后面各章中会有更详细且更合适的解释，这种螺旋式的学习方法向C++程序员强调了C++程序的思想和技术本质。

C++语言是C语言的超集，学会了C++，也就学会了C语言的核心内容。Al Kelley和Ira Pohl合著的*C by Dissection*（第4版）(Addison-Wesley, 2000）讲解了本书中没有介绍的C语言的其他内容。

本书大多数章节中也对C++程序和Java程序进行了比较。Java是部分基于C++的，但是，它与C++不同，某些C语言的概念并没有出现在Java中或者具有不同的含义。越来越多开始学习C++的人已经具有了Java背景。Ira Pohl和Charlie McDowell合著的*Java by Dissection*（Addison-Wesley, 1999）介绍了Java程序设计过程。现代的程序员要能够学会并区分这3种基于C的程序设计语言。

1.1 编程准备

程序的作用是指示机器执行特定的任务或解决特定的问题。逐步完成期望任务的过程叫作*算法*（algorithm），因此，程序设计就是沟通算法与计算机的一种活动。我们可以用英语给某人下达指示，然后那个人就执行这个指示，程序设计过程与此类似，只不过机器不能容忍模糊的指令，所以必须用精确的语言给出明确的指示。

程序设计过程

1. 指定任务。
2. 找出解决问题的算法。
3. 用C++为算法编写代码。
4. 测试代码。

计算机是数字电子设备，它由3个主要的部件组成：处理器、存储器和输入/输出设备。处理器也叫*中央处理器*（central processing unit）或CPU，它执行存储在存储器中的指令。与指令一样，数据也存储在存储器中。处理器根据指令按期望的方式操作数据，输入/输出设备从外界获取信息并将信息提供给外界。典型的输入设备是键盘、磁盘驱动器和磁带驱动器，

典型的输出设备是终端屏幕、打印机、磁盘驱动器和磁带驱动器。机器的物理组成是很复杂的，但是用户不需要考虑这些细节。

操作系统（operating system）是由特定程序集合组成的，它的主要目的有两个：首先，操作系统监视并协调机器的所有资源，例如，当在磁盘上创建一个文件时，操作系统需要找出一个合适的位置来存储文件，并记录该文件的名字、大小、创建日期等信息；其次，操作系统为用户提供工具，多数工具对C++程序员都很有用，其中有两种工具极为重要，即文本编辑器和C++编译器。

我们假定读者能够使用文本编辑器创建并修改包含C++代码的文件。C++代码也叫作*源代码*，包含源代码的文件叫作*源文件*。在创建了包含源代码（程序）的文件之后，需要调用C++编译器，这个过程取决于不同的系统（见1.6节）。例如，在许多UNIX系统上，可以使用如下命令调用C++编译器：

```
CC pgm.cpp
```

这里*pgm.cpp*是包含程序的文件的名字。如果*pgm.cpp*中没有错误，该命令会产生一个可执行文件，即能够运行或执行的文件。虽然我们认为这就是编译程序的过程，但其具体细节更复杂。

编译一个简单程序时，需要经历3个单独的操作步骤：首先调用预处理器，然后是编译器，最后是连接器。预处理器通过包含其他文件和进行其他操作来修改源代码的副本，编译器将预处理后的代码翻译为*目标代码*，然后连接器使用目标代码生成最后的可执行文件。包含目标代码的文件称为*目标文件*。目标文件不同于源文件，人们通常不能阅读它。当提到编译一个程序时，一般都是指调用预处理器、编译器和连接器。对于一个简单的程序来说，这些操作可以通过一个简单命令完成。

当程序员编写完一个程序后，必须对程序进行编译和测试。如果需要修改程序，必须重新编辑源代码。因此，程序设计过程就是如图1-1所示的一个循环过程。

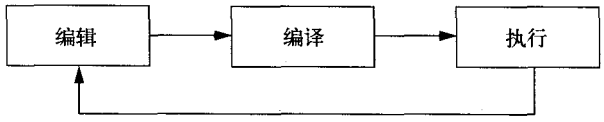


图1-1　程序设计过程

当程序员对程序的执行结果满意时，循环结束。

1.2　第一个程序

对于任何学习编程的人来说，首要的任务是在屏幕上显示一些内容。下面我们编写一个传统的C++程序，该程序在屏幕上显示一个短句Hello, world!。完整的程序如下。

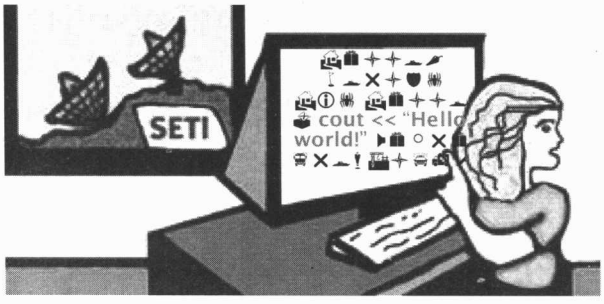
文件 hello1.cpp

```
// Hello world in C++
// by Olivia Programmer

#include <iostream>                    // I/O library
using namespace std;

int main()
{
      cout << "Hello, world!" << endl;
```

```
}
```

程序员可以使用文本编辑器将上面的代码录入一个以*.cpp*为后缀的文件中，并选择一个容易记忆的文件名，例如*hello.cpp*。当编译并执行这个程序时，下面的消息会显示在屏幕上：

```
Hello, world!
```



嗨，海蒂，比邻星对你说“Hello”，这是用C++写出来的！

*hello*程序解析

■ `// Hello world in C++`
`// by Olivia Programmer`

//是单行注释符号。程序文本可以放在页面上的任意位置，程序会忽略符号之间像空格、Tab或换行这样的空白。空白、注释和文本缩进用于建立一个良好文档化的可读程序，而不会影响程序的语义。

■ `#include <iostream>  // I/O library`

在预处理器执行了以#开始的指令后，开始编译C++程序。预处理器先于编译器将程序代码翻译为机器代码。在示例程序*hello.cpp*中的#include指令将引入所有需要的文件，通常是库定义。在这个例子中，典型的C++编译系统使用的I/O库定义在文件*iostream*中。编译器知道在哪里可以找到这个文件以及其他系统文件。

■ `using namespace std;`

在C++系统中，标准C++ I/O头文件包含在namespace std中，using声明允许程序员使用名字时不为每个名字使用std::。包含文件也可以不使用namespace和using，例如：

```
#include <iostream.h>  // I/O library
```

大部分系统提供较旧的*library_name.h*头文件，这些库不需要using namespace std;语句。

■ `int main()`
`{`

C++程序是声明和函数的集合，它从函数main()处开始执行。

■ `cout << "Hello, world!" << endl;`

标识符cout是标准输出流对象，定义在*iostream*中。在大部分C++系统中，标准输出流都与屏幕相连。标识符endl是标准操纵符（manipulator），它清空输出缓冲区，将所有字符输出并开始新的一行。运算符<<是用于输出的插入操作符，它紧跟在cout之后。

注意，如果不使用using std语句，该语句可以写成如下形式：

```
std::cout << "Hello, world!" << std::endl;
```

运算符::称为作用域解析运算符（scope resolution operator），它告诉编译器在什么范围内检

查和理解标识符cout。一个标识符的作用域（scope）是可能使用该名字的程序文本。

■ }

这个符号表示函数main()结束。在C++中，大括号是成对出现的，左大括号{表示结构开始（begin construct），右大括号}表示结构结束（end construct）。C++函数中的返回类型void表示函数没有返回值。特殊函数main()运行时向系统返回一个整数值。函数main()隐含返回0，表示程序正常终止。当然也可以在右大括号前显式书写以下代码：

```
return 0;
```

表达式“cout<<一些字符串”用于向屏幕输出数据。当读入换行符或遇到endl时，它移动到新的一行。屏幕是二维显示设备，它按照自左向右，自上向下的顺序输出。

现在重写上面的程序，使其能够使用两个cout语句。尽管该程序与上面的程序不同，但
5 它们的输出相同。

文件 hello2.cpp

```
// Hello world in C++
// by Olivia Programmer
// Version 2

#include <iostream> // I/O library
using namespace std;
int main()
{
    cout << "Hello, ";
    cout << "world!" << endl;
}
```

注意，第一条语句的字符串结尾处有一个空格字符。如果没有这个空格，输出时Hello, world!之间将没有间隔。

作为这个程序的最后一个变形，增加短句Hello, universe!，并在第2行显示。

文件 hello3.cpp

```
// Hello universe in C++
// by Olivia Programmer

#include <iostream> // I/O library
using namespace std;
int main()
{
    cout << "Hello, world!" << endl;
    cout << "Hello, universe!" << endl;
}
```

执行这个程序时，下面的内容出现在屏幕上：

```
Hello,world!
Hello,universe!
```

注意，main()函数体中的两个cout语句可以被下面的单条语句代替：

```
cout << "Hello, world!\nHello, universe!" << endl;
```

6 在这条语句中，特殊字符\n是换行符，效果与endl相同。

1.3　问题求解：烹饪法

计算机程序就是一个详细的指令列表，用来解决一个特定的任务或者一种特定类型的问

题。指令列表（也称算法）普遍应用在各种日常生活中，例如做饭、织毛衣、报名上课等。下面仔细考察一个示例，即烤肉的整个烹饪过程。

> 在要烤的肉上撒上盐和胡椒，并插入一支专用温度计，然后将它放入已预热到150°C的烤箱中开始烘烤，直至温度计指示的温度到了80~85°C。然后在烤肉上浇上肉汁卤，这个肉汁卤可以用煮肉块的原汤来调配；如果有足量的烤出来的肉汁，也可以用这种肉汁来调配。

这个烹饪方法是典型的不精确方法——“撒”是什么撒法？温度计应该插在哪个位置？烤出多少肉汁算是足量？这个烹饪方法可以更精确地被程序化成一个指令列表，可以一行一行地读取。

烤肉

1. 在要烤的肉上撒上1/8茶勺的盐和胡椒粉。
2. 将烤箱开到150°C。
3. 在肉中间插入一个肉类专用温度计。
4. 等上几分钟。
5. 如果烤箱还没有达到150°C，回到第4步。
6. 将肉放入烤箱。
7. 等上几分钟。
8. 查看温度计，如果温度还没有达到80°C，回到第7步。
9. 将肉从烤箱中取出。
10. 如果烤出来的肉汁能有半杯，直接跳到第12步。
11. 用肉的原汤准备肉汁卤，并跳到第13步。
12. 用烤出来的肉汁准备肉汁卤。
13. 把肉汁卤浇在烤肉上。

这些步骤包含了3类指令和行为：第一类涉及操作设备或者原料，第二类是检测或查看系统状态，第三类是转换到下一步。上面的第1步和第6步属于第一类，第8步中的查看温度和第10步中的测试肉汁是否足量属于第二类，第5步和第8步中的转移（回到第x步）属于第三类。

7

可以使用一些合适的图形符号来表示这些不同类别的指令和行为，图1-2中用一个简单的二维图表示这个烹饪算法。

这样的图叫作*流程图*（flowchart）。只要按照图中的箭头和每个框中的指令，就可以执行这个程序（准备烤肉）。长方形框中是操作性动作，菱形框中是条件测试，而箭头则决定了控制的转移和流动。因为流程图看上去清晰易懂，在非正式描述程序时通常用它来代替指令列表。有些烹饪书的作者甚至广泛地应用流程图。

算法——要精确

上面关于烤肉的烹饪方法不能被计算机执行是因为它的每条指令还定义得太模糊。现在考虑另外一个不是操作食物而是操作数字的例子：你用一张一美元的钞票去购买某种商品，然后找回一些一角硬币和分币。这里的问题是如何使零钱中的分币最少。大多数人在做这种每天都会发生的简单事务时都不用思考，但我们如何能精确地描述这个算法呢？