HZ BOOKS

PEARSON Addison Wesley

Modern C++ Design C++ 设计新思维

Absolute C++中文版

（原书第2版）

完美的C++教程

Absolute C++ *Second Edition*

（美） Walter Savitch 著
江山 等译

机械工业出版社
China Machine Press

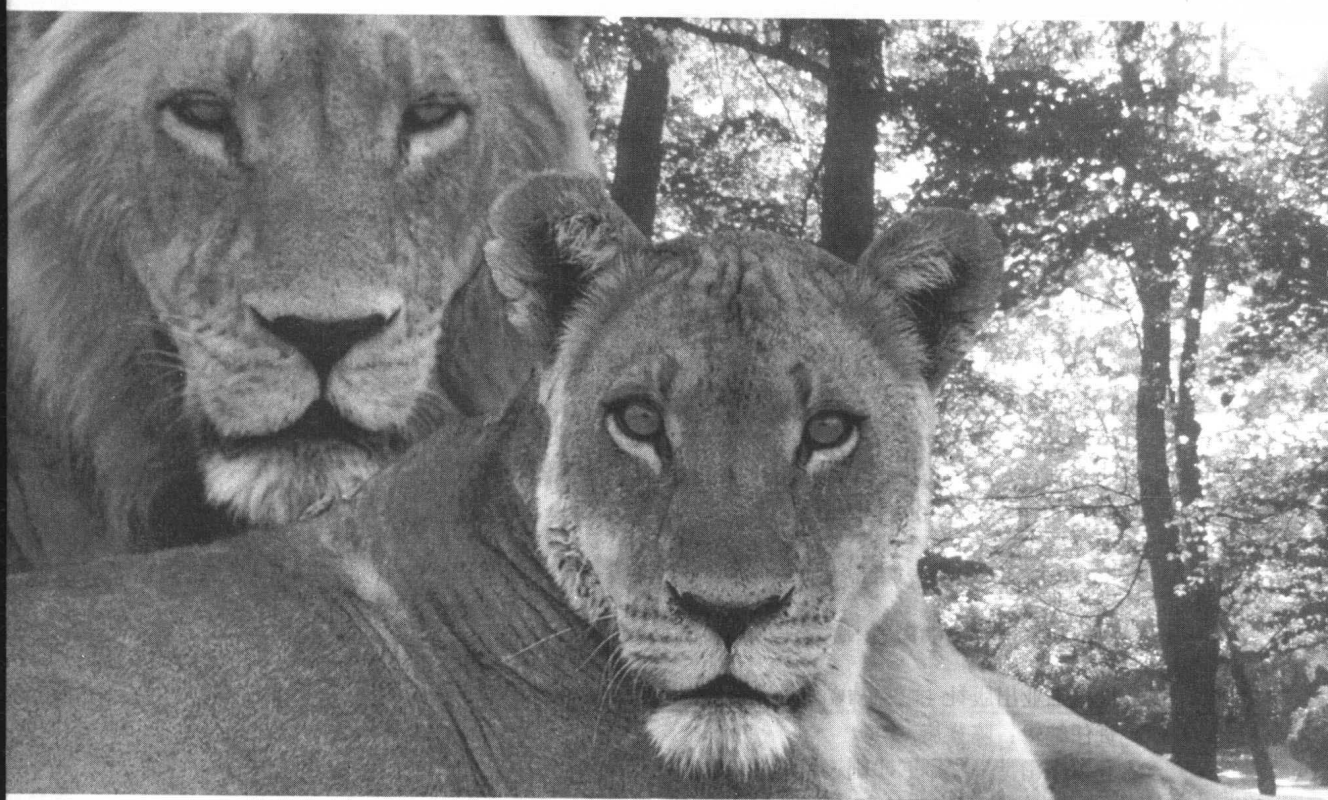
C++ 设计新思维

Absolute C++ 中文版

（原书第2版）

完美的C++教程

Absolute C++ *Second Edition*



（美） Walter Savitch 著
江山 等译

机械工业出版社
China Machine Press

本书是讲解C++语言程序设计的优秀教程。全书围绕C++语言的结构来组织，开始章节介绍编程的普通概念，接下来详细介绍C++中的继承、多态、异常处理以及标准模板库(STL)，同时还包含模式和UML的介绍。本书内容系统、全面，给出了大量代码示例、自测练习、编程提示和编程练习，并且提供了练习的解答。本书有利于初学者尽快掌握C++的编程知识，养成良好的编程习惯；具备相当编程经验的人，也可以从本书了解到使用C++的更有效的方法。

本书适合高等院校师生、程序设计专业人员和程序设计爱好者参考。

本书版权登记号：图字：01-2006-3143

图书在版编目（CIP）数据

Absolute C++中文版（原书第2版）/（美）萨维奇（Savitch, W.）；江山等译. －北京：机械工业出版社，2007.4

（C++设计新思维）

书名原文：Absolute C++, Second Edition

ISBN 978-7-111-20945-4

Ⅰ. A…　Ⅱ. ①萨…②江…　Ⅲ. C语言－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2007）第024585号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：陈冀康

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2007年4月第1版第1次印刷

186mm × 240mm · 39.5印张

定价：75.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线（010）68326294

“C++设计新思维”丛书前言

自C++诞生尤其是ISO/ANSI C++标准问世以来，以Bjarne Stroustrup为首的C++社群领袖一直不遗余力地倡导采用“新风格”教学和使用C++。事实证明，除了兼容于C的低阶特性外，C++提供的高级特性以及在此基础上发展的各种惯用法可以让我们编写出更加简洁、优雅、高效、健壮的程序。

这些高级特性和惯用法包括精致且高效的标准库和各种“准标准库”，与效率、健壮性、异常安全等主题有关的各种惯用法，以及在C++的未来占据更重要地位的模板和泛型程序设计技术等。它们发展于力量强大的C++社群，并被这个社群中最负声望的专家提炼、升华成一本本精彩的著作。毫无疑问，这些学术成果必将促进C++社群创造出更多的实践成果。

我个人认为，包括操作系统、设备驱动、编译器、系统工具、图像处理、数据库系统以及通用办公软件等在内的基础软件更能够代表一个国家的软件产业发展质量，迄今为止，此类基础性的软件恰好是C++所擅长开发的，因此，可以感性地说，C++的应用水平在一定程度上可以折射出一个国家的软件产业发展水平和健康程度。

前些年国内曾引进出版了一大批优秀的C++书籍，它们拓宽了中国C++程序员的视野，并在很大程度上纠正了长期以来存在于C++的教育、学习和使用方面的种种误解，对C++相关的产业发展起到了一定的促进作用。然而在过去的两年中，随着.NET、Java技术吸引越来越多的注意力，中国软件产业业务化、项目化的状况愈发加剧，擅长于“系统编程”的C++语言的应用领域似有进一步缩减的趋势，这也导致人们对C++的出版教育工作失去了应有的重视。

机械工业出版社华章分社不断推进中国C++“现代化”教育，从2006年起将陆续推出一套“C++设计新思维”丛书。这套丛书秉持精品、高端的理念，其作译者为包括Herb Sutter在内的国内外知名C++技术专家和研究者、教育者，议题紧密围绕现代C++特性，以实用性为主，兼顾实验性和探索性，形式上则是原版影印、中文译著和原创兼收并蓄。每一本书相对独立且交叉引用，篇幅短小却内容深入。作为这套丛书的特邀技术编辑，我衷心希望它们所展示的技术、技巧和理念能够为中国C++社群注入新的活力。

荣　耀

2005年12月

南京师范大学

www.royaloo.com

译者序

说起程序设计语言，C++可谓大名鼎鼎，不仅因为其内容博大精深，更因其在业界的广为使用。当然，这也导致了市场上C++书籍的泛滥。实际上很多C++书籍都相当一般，没有特色或不易研读。好书不是没有，较早的如C++之父Bjarne Stroustrup所著的《C++程序设计语言》，较近的有《C++ Primer》（现在是第4版）和《Effective C++》，这些都是精品，但却稀有。反观本书，它没有上面这几本书那样显赫或具有特色，但却十分朴实。全书循序渐进，深入浅出地介绍了C++的各个主题，可以很好地帮助读者成为一名优秀的C++程序员，而且不要求读者事先对C++有太多了解。

本书的好用、实用离不开作者Walter Savitch教授的精心组织，这位加州大学圣迭戈分校的教授除了在计算领域做出重要的学术贡献外，还著有一系列编程语言教材（国内大都已经引进），受到普遍欢迎，畅销不衰。Savitch教授是一位真正难得的教材作者，读他的书你会发现，书中其实没有什么惊人的观点（从目录上就能看出来），但他却能充分考虑读者和学生的学习方式，将内容安排得井井有条，深浅有序，极具可读性。就本书而言，除了系统、全面的内容介绍外，还给出了大量代码示例、自测练习、编程提示和编程练习，并且提供了练习的解答。编程练习是这本书的一个亮点，并不是每本书都能提供这方面的编程项目，而且本书还在相关网站提供解决方案，这对我们的实际编程尤其具有指导意义。尽快地掌握这些编程经验，将有助于编程初学者养成良好的编程习惯，即使是具备相当编程经验的人员，也可以从本书了解到使用C++的更有效方法。

本书有几个特点值得强调：

- 内容全面。本书所包含的C++内容非常全面，充分考虑了不同水平的读者需要。特别是本书详细介绍C++中的继承、多态、异常处理和标准模板库（STL），同时还包含了模式和UML的介绍。
- 要点总结。每个主要的知识点都在文中单独进行总结，这既是内容的总结，也是快速的语言参考。贯穿每一章，还有“陷阱”、“编程技巧”、“提示”等经验小结，有利于读者深入理解知识点。
- 强调编程实践。这一点上面已经提到，除了一般意义上的教学示例外，本书在每一章末尾提供了大量编程练习，实际做一下这些练习题具有深远的意义。
- 代码示例关注细节。本书为代码段配有极为丰富的注释和说明文字，而不是简单地堆砌代码，这十分难得，使得书中的代码非常易于阅读和理解。

总之，本书无论是作为正式教材还是自学用书，都非常适合。如果您是教师，想找一本合适的C++语言教材，或者您是一位自学者，想从零起步成为一名专业的C++程序员，我们都郑重向您推荐本书。

本书的翻译和统稿工作主要由江山完成，另外，刘洞宾、方亮、李璐等人参与了部分章节的翻译工作。由于时间仓促，加之译者的水平有限，在翻译过程中难免会出现错误和疏漏，恳请广大读者批评指正。

前　言

本书是关于C++语言程序设计的教程和参考书。本书主要是围绕C++语言的特点来组织的，虽然包括一些编程技术，但不是针对编程技术的详细课程。本书的目标读者是那些并没有太多C++语言编程经验的本科生，因此，本书适合作为使用C++语言的计算机课程的教材和参考书，适用于不同层次的读者。开始的章节是为入门级读者准备的，其中许多小节向有经验的程序员介绍了基本的C++语法。后续的章节对初学者也是易于理解的，但主要是为那些希望掌握更高级主题的学生准备的（对于那些需要包含更多教学材料以及更多基本编程技术的入门级教材的读者，我推荐我编写的另一本书：*Problem Solving with C++: The Object of Programming*，第5版，Addison-Wesley）。本书也适合任何自学C++语言的读者。

本书所包含的C++内容非常全面，超出了初学者应该掌握的范围。特别是，本书详细介绍了C++中的继承、多态、异常处理以及标准模板库（STL），同时还包含模式和统一建模语言（UML）的介绍。

第2版的变化

本书第2版的内容和介绍的顺序与第1版并无二致。如果教师已经在教学中使用本书第1版，可以继续教授而无需做出改变。第2版在行文语言和代码的组织上更有条理，但内容变化不大。

本书第2版显著扩充并改进了每一章结尾的编程练习，全书共有超过50个新的编程练习。同时，部分编程练习还全面整合到CodeMate（Addison-Wesley的在线教学和课后作业资源）中。

ANSI/ISO C++标准

本书的内容完全符合最新的ANSI/ISO C++标准。

标准模板库

标准模板库（STL）是一个包含编程前数据结构类和算法的扩展库。STL本身就是一个和C++语言同样大的主题，因此本书非常充分地介绍了STL，分别用一章的篇幅介绍了一般模板和STL的特色，同时也提供了其他与STL相关的资料。

面向对象的编程

本书是围绕C++语言的结构来组织的，同其他高级编程语言一样，本书的开始章节是针对编程的普通概念，而不是专门针对面向对象的编程（OOP）。这使本书可以作为参考书和学习第二编程语言的教材。但是，考虑到C++是一门面向对象的编程语言，因此，如果读者要用C++而不是C语言编程，就必须利用C++语言的面向对象编程特性。本书覆盖了大量的C++语言实现的封装、继承和多态，最后一章是关于模式和UML的，也包括与OOP相关的材料。

灵活安排的主题

本书允许指导教师灵活地重新安排教学材料，如果将本书作为参考书，这一点很重要。这也符合我

的写作初衷——使内容适应教师的风格，而不是将教师限制在我个人的内容安排偏好上。在这种思想的指导下，我在每个章节前面的介绍中概述了本章所包含的内容。

便于学生理解接受

对一本书来说，仅仅按照正确的顺序罗列恰当的主题是不够的，甚至仅有内容的清楚和正确也是不够的。书的内容必须以一种初学者易接受的方式来表达。和其他我编写的受到学生们欢迎的教材一样，本书也是按照适合学生阅读和理解的方式来编写的。

总结框

每个主要的知识点都在一个方框中进行了总结。每章都有这些方框，既是内容的总结，也是快速的语言参考，同时还可以让那些通晓一般编程规则，但需要知道C++语言特点的读者快速学习C++语法。

自测练习

每一章都包含有大量的自测练习，完整的答案附在每一章的最后。

其他特色

书中还穿插了许多易犯错误小节、编程技巧小节以及包含输入/输出的示例程序。每一章的最后都有一个总结，以及适合学生练手的编程练习。

CodeMate在线教学资源

CodeMate是一个在线资源，它能提供教学帮助并对学生的编程练习实践进行评估。本书的代码演示以及部分编程练习已经完全整合到CodeMate中。使用CodeMate，学生可以获得与编程练习相关的提示，编写并编译程序，获取如何解决编译器错误消息的反馈，而所有这些通过一台可访问因特网的计算机即可完成。教师可以跟踪每位学生在编程练习上的进展。有关CodeMate的更多信息，请访问

http://www.aw-bc.com/codemate

支持材料

有些支持材料面向所有读者开放，有些材料则只有授权的教师才可获得。

所有读者可获得的材料

- 自测题集
- 书中程序的源代码
- PowerPoint幻灯片

要获得这些学生支持材料，请访问

http://www.aw-bc.com/savitch

授权的教师可获得的资料

以下是经过授权的教师才可获得的资料。要获悉如何访问这些教师资料，请联系当地的Addison-

Wesley销售代表或发送电子邮件到aw.cse@aw.com ⊖。

- 访问Addison-Wesley的CodeMate。
- 教师资源手册——包括每一章的教学提示、测试题和解答以及众多编程练习的解决方案。
- 测试题库和测试生成器。
- PowerPoint教案，包括书中的程序和图片。

电子邮件交流

我非常希望听到读者对本书的任何评论，这样我就有动力去改善本书来进一步满足大家的需要。请将评论发至wsavitch@ucsd.edu。

我想知道的是读者对本书的评价和修改意见，不过，我不打算向学生提供电子邮件咨询或教学服务，因为我收到的此类邮件实在是太多了。特别是，我无法对本书中的练习提供解答，或对你的老师提供的练习进行解答。作为对期待此类帮助的读者的一些补偿，本书对所有的“自测练习”都提供了解答。教师手册提供了一些“编程练习”的解答，但这些材料只向采用本书作为教材的教师提供，而不能分发给学生。

致谢

为了本书的出版，许多人提供了宝贵的帮助和支持。Addison-Wesley公司的Frank Ruggirello和Susan Hartman最先提出了本书的写作构想并促成了第1版的面世，我对他们及本书第2版的编辑Matt Goldstein充满了感激之情。我还要感谢Michelle Brown、Katherine Harutunian、Joyce Wells以及Addison-Wesley公司的所有其他工作人员，感谢他们无私的支持和鼓励。

在此要特别感谢Addison-Wesley公司的Patty Mahtani，是她的大力支持和鼓励使得本书得以顺利付印。还有Daniel Rausch、Meghan James以及Argosy 出版公司的工作人员，感谢他们在本书的录入、排版等生产环节上做了大量重要的工作。

我要特别地感谢Kenrick Mock帮忙编写了许多精彩的新的编程练习，并更新了教师手册。我尤其感谢David Teague，他对本书第1版做了细致的审查和研究工作。同时，感谢我的好朋友Mario Lopez，他和我进行了大量有益的关于C++的交流。

以下人员对本书第2版提出了建议，在此感谢他们的努力工作和有益的意见：Victoria Rayskin，Central Los Angeles大学；Jerry K. Bilbrey, Jr，Francis Marion大学；Albert M. K. Cheng，休斯顿大学；Tim Lin，加利福尼亚工艺学院，Pomona；Ron DiNapoli，康奈尔大学；R. M. Lowe，Clemson大学；Martin Dulberg，北卡罗来那州立大学；Jeffrey L. Popyack，Drexel大学。

以下人员对本书第1版提出了勘误和建议，在此感谢他们的努力工作和有益的意见：Kenrick Mock，阿拉斯加大学，安克雷奇；Richard Albright，特拉华大学；H. E. Dunsmore，普度大学；Christopher E. Cramer；Drue Coles，波士顿大学；Evan Golub，马里兰大学；Stephen Corbesero，Moravian学院；Fredrick H. Colclough，科罗拉多理工大学；Joel Weinstein，东北大学；Stephen P. Leach，佛罗里达州立大学；Alvin S. Lim，厄本大学；Martin Dulberg，北卡罗来那州立大学。

W. S.
http://www-cse.ucsd.edu/users/savitch/
wsavitch@ucsd.edu

⊖ 中国读者可填写书后的“教学支持说明”，然后与出版社联系。——编辑注

目　录

第1章 C++基础

概述

本章简要介绍C++语言。仔细阅读本章之后，你将掌握一些基本的C++程序，包括表达式、赋值语句以及控制台输入/输出（console I/O）。有关赋值语句和表达式的部分与其他大多数高级语言类似，但各种语言都有各自不同的控制台I/O的语法，因此如果你对C++不甚熟悉的话，那么读到这部分的时候会觉得比较生疏。

1.1 C++简介

本节概要介绍C++程序设计语言。

1.1.1 C++语言的起源

从分类上讲，C++程序设计语言可以被看做是更具现代特色的C语言。C语言是由AT&T贝尔实验室的Dennis Ritchie在20世纪70年代创建的。它首先用于编写及维护UNIX操作系统（在此之前，UNIX系统编程用的是汇编语言或B语言——UNIX的创始人Ken Thompson所发明的一种语言）。C语言的用途非常广泛，可用于编写任何程序，但它的成功及广受欢迎是与UNIX操作系统密切相关的。如果要维护UNIX系统，就需要用C语言。C语言与UNIX配合得如此之好，以至于在短时期内，不只是系统程序，几乎所有在UNIX环境下运行的商业程序都采用C语言进行编写。C语言十分流行，一时间产生了许多可运用于其他常见操作系统的不同版本，C语言的运用也不再局限于使用UNIX的计算机了。尽管C语言广受欢迎，但C语言并不是没有缺陷的。

C语言的独特之处在于它是一种拥有众多低级语言特性的高级语言，从某种程度上来说，C语言介于严格的高级语言与低级语言之间，兼具两者的优缺点。正如（低级的）汇编语言一样，C语言程序可以直接操作计算机的内存。另一方面，C语言又像其他高级语言一样比汇编语言更易于读写，这使C语言成为编写系统程序的最佳选择。然而，对其他程序而言（某种程度上甚至对系统程序也如此），C语言不如其他语言易于理解，也不像其他高级语言那样拥有许多自动检测功能。

为了克服上述及其他一些缺陷，20世纪80年代早期，AT&T贝尔实验室的另一个人Bjarne Stroustrup开发了C++语言。Stroustrup意图使C++成为更好的C语言，因此C语言的大部分内容都包含在C++中，所以绝大多数C语言程序也是C++程序（反之并不成立，许多C++程序并不是C程序）。与C不同的是，C++有许多创建类的工具，因此它可用于面向对象的程序设计。

1.1.2 C++与面向对象的程序设计

面向对象的程序设计（object-oriented programming，OOP）是当今流行的一种功能强大的编程技术。OOP的主要特点是：封装、继承与多态。封装是一种信息隐藏或抽象的方式；继承可用于编写可重复使

用的代码；多态指的是在继承的内容中可以让一个名字具有多个意义的方法。话虽如此，我们必须承认这些对于从未听说过OOP的读者而言意义不大，我们会在以后的章节中对上述问题进行详尽的介绍。C++通过类来支持OOP，类是一种结合了数据及算法的数据类型。但是C++并不像某些作者所说的那样是“纯OOP语言”，C++的OOP特性中融入了对有效性及实用性的考虑。这种结合使C++成为当前应用最为广泛的OOP语言，虽然在实际应用中并非所有的操作都严格遵守OOP的准则。

1.1.3 C++的特点

C++有类的概念，因此它是面向对象的语言。它允许函数及操作符的重载（所有这些术语的含义都将会有进一步的解释，如果你一时不能完全理解也没有关系）。C++与C语言的联系使它较之新的面向对象的语言更为传统，但是它比现在流行的其他语言具有更有力的抽象机制。C++的模板工具可以实现完全而直接的算法抽象。C++模板允许使用类型参数进行编码，最新的C++标准及大多数C++编译器允许存在多重名字空间，以便提供更多的类名和函数名的重复使用。C++的异常处理工具与其他程序设计语言相似，而C++的内存管理则与C语言相似，程序员必须分配他们自己使用的内存，并做相应的回收工作。由于C语言是C++的基本子集，大多数编译器允许在C++程序中使用C语言风格的内存管理。但C++也有其自身风格的内存管理的语法。建议在C++中编码时采用C++风格的内存管理。本书将只采用C++风格的内存管理。

1.1.4 C++术语

在C++中所有过程式的实体都称做*函数*（function）。在其他语言中称为*过程*（procedure）、*方法*（method）、*函数*（function）或*子程序*（subprogram）的在C++中都叫做*函数*。在下一小节里我们将提及**C++程序**（program）。一个C++程序实际上就是一个叫做main的函数，运行一个程序时，运行时系统会自动调用名为main的函数。其他C++术语与大多数程序设计语言的术语非常相似，本书会对用到的每一个概念都详加解释。

1.1.5 C++程序示例

示例1.1中包含了一个简单的C++程序，程序后面附有运行该程序后可能会出现的两种运行结果。一个C++程序实质上就是一个main函数的函数定义。当这个程序运行时，main函数被调用。main函数的函数体包含在大括号{}内。当程序运行时，将执行大括号中的语句。

示例1.1 一个简单的C++程序

```
1  #include <iostream>
2  using namespace std;

3  int main( )
4  {
5      int numberOfLanguages;

6      cout << "Hello reader.\n"
7           << "Welcome to C++.\n";

8      cout << "How many programming languages have you used? ";
9      cin >> numberOfLanguages;
```

```
10      if (numberOfLanguages < 1)
11          cout << "Read the preface. You may prefer\n"
12               << "a more elementary book by the same author.\n";
13      else
14          cout << "Enjoy the book.\n";

15      return 0;
16  }
```

运行结果示例1：

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 0
Read the preface. You may prefer
a more elementary book by the same author.
```

用户从键盘输入“0”

运行结果示例2：

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 1
Enjoy the book
```

用户从键盘输入“1”

下面两行语句包含了在程序中使用的控制台输入和输出工具的库，有关这两行的详细内容将在1.3节以及第9章、第11章和第12章中讲解：

```
#include <iostream>
using namespace std;
```

下面一行表明main是一个没有参数、返回值为int（整数）的函数：

```
int main()
```

某些编译器允许省略上面的int或用void来代替，这意味着该函数没有返回值。但上面的形式是在C++程序中开始一个main函数最常用的方法。

执行以下语句时程序终止：

```
return 0;
```

该语句终止函数main的运行，并使函数值返回0。按照ANSI/ISO C++标准，该语句并不是必需的，但目前许多编译器仍然需要保留该语句。C++函数的所有具体内容将在第3章中做详细介绍。

在C++中声明变量的方法与在其他程序设计语言中相似。示例1.1中下面的这行语句声明了变量numberOfLanguages：

```
int numberOfLanguages;
```

C++中的int类型是一种数字（整数）类型。

如果你从未使用过C++编写程序，那么在运用cin及cout进行控制台输入/输出的时候就会觉得比较陌生。这一点稍后将加以解释。但我们不妨先从示例1.1这个简单的程序中获得一点大致的印象。例如，请看下面两行：

```
cout<<"How many programming languages have you used?";
cin>>numberOfLanguages;
```

第一行在屏幕上输出引号里面的原文。第二行要求用户在键盘上输入一个数字，并将变量

numberOfLanguages的值设置为该数字。

再看这几行：

```
cout << "Read the preface. You may prefer\n"
     <<"a more elementary book by the same author.\n";
```

以上两行输出的是两个字符串而不是一个字符串。具体细节将在1.3节中加以说明。通过上面简短的介绍，你可以在阅读1.3节之前对cin及cout的使用有一个初步了解。符号\n是换行符，它告诉计算机在输出时要从新的一行开始。

也许你还不是非常确定如何书写每一个语句，但对于if-else语句，你也许可以猜出它的含义。具体的细节将在下一章中加以阐述。

顺便说一句，如果你没有任何使用程序设计语言的经验，建议你仔细阅读本书的前言部分，看看就上面讨论的内容你是否需要补充一些更基础的知识。阅读本书并不要求有使用C++的经历，但一些基本的程序设计基础还是必不可少的。

1.2 变量、表达式及赋值语句

C++中的变量、表达式以及赋值语句与其他一般用途的语言类似。

1.2.1 标识符

变量（或在程序中定义的其他项）的名字称做**标识符**（identifier）。一个C++标识符需以字母或下划线开始，其余的字符应该是字母、数字或下划线。例如，下面的标识符都是有效的：

```
x  x1  x_1  _abc  ABC123z7  sum  RATE  count  data2  bigBonus
```

上面这些名字都是合法的，可以被编译器接受。但前5个用做标识符并不合适，因为它们没有描述出标识符的具体用途。下面这些是非法的标识符，会被编译器拒绝：

```
12  3x  %change  data-1  myfirst.c  PROG.CPP
```

前3个不正确的原因是因为没有以字母或下划线开始，后3个则是由于包括了除字母、数字和下划线之外的其他符号。

虽然以下划线开头的标识符是合法的，但最好避免出现这样的情况，因为一般来讲，以下划线开始的标识符是为系统标识符和标准库保留的。

C++是一种**区分大小写**（case-sensitive）的语言，也就是说，它可以区分标识符中字母的大小写。因此，下面是3个不同的标识符，可以用来命名3个不同的变量：

```
rate  RATE  Rate
```

不过为了避免混淆，在同一程序中最好不要使用这样的标识符。另外，虽然没有明文规定，但C++中的变量首字母一般都要求小写。预定义的标识符（如main、cin、cout等）则必须全都是小写。这种习惯目前在OOP中逐步演化成大小写字母（及数字）混用书写变量名，一般以小写字母开始，以大写字母区分“词界”。如下所示：

```
topSpeed, bankRate1, bankRate2, timeOfArrival
```