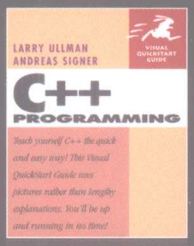
TURING

Peachpit Press

写给大家看的C++书

[美] Larry Ullman Andreas Signer 著
杨 涛 王建桥 杨晓云 等译



- 有大师指导，人人都能成为编程高手
- 学习 C++ 就这么简单
- 图文并茂，丰富的实战代码

人民邮电出版社
POSTS & TELECOM PRESS

TURING

写给大家看的C++书

[美] Larry Ullman Andreas Signer 著
杨 涛 王建桥 杨晓云 等译

人民邮电出版社
北 京

图书在版编目（CIP）数据

写给大家看的C++书 /（美）厄尔曼（Ullman, L.），（美）赛纳（Signer, A.）著；杨涛等译．—北京：人民邮电出版社，2009.7
书名原文：C++ Programming
ISBN 978-7-115-19518-0

Ⅰ.写… Ⅱ.①厄…②赛…③杨… Ⅲ.C语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2008）第201268号

内 容 提 要

本书从实际出发，基于示例循序渐进地讲述了基本的C++编程技术。作者首先教你如何创建一个基本的程序，然后从简单的变量开始逐一讲解了数据类型、操作符、输入输出文件、函数、对象、调试和动态内存管理、模板等内容。书中所有示例均在Windows、Unix和Mac OS X操作系统上测试通过，其流畅的叙述方式可以指导读者为任何平台开发C++应用程序。

本书简单易懂，适用于C++初学者，也可作为高等院校计算机专业的教材使用。

写给大家看的C++书

◆ 著 [美] Larry Ullman Andreas Signer
译 杨 涛 王建桥 杨晓云 等
责任编辑 傅志红 刘艳娟

◆ 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
北京顺义振华印刷厂印刷

◆ 开本：800×1000 1/16
印张：26.5
字数：729千字 2009年7月第1版
印数：1－3 000册 2009年7月北京第1次印刷

著作权合同登记号 图字：01-2008-5815号
ISBN 978-7-115-19518-0/TP

定价：49.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

谨以此书献给Peachpit Press出版公司的员工们，尤其是勤劳敬业的Rebecca Gulick编辑。感谢她的耐心和支持，感谢她为了让本书尽善尽美而做的一切。

——Larry Ullman

谨以此书献给Claudia，感谢她付出耐心，给予我支持和理解。

谨以此书献给我的母亲，为了这么多年的养育之恩。

——Andreas Signer

致　谢

Larry向以下诸位表示最衷心的问候和感谢。

合作者Andi Signer。写书是一个大工程，如果没有他作为合作伙伴，我是不可能写出这本书的。感谢他的出色工作和全身心的投入。

Peachpit Press出版公司的全体员工，尤其是Nancy Ruenzel、Nancy Davis和Marjorie Baer；营销团队，是他们把这本书介绍给了书店；Lisi Baldwin，她把翻译好的版本送来给我；还有Peachpit和Pearson公司里的其他人，虽然我不知道他们的名字和职务，但没有他们就没有这本书。

编辑Rebecca Gulick时常为我打气鼓劲，并担当协调者和指路人，而且总是使一切柳暗花明。

文字编辑Bob Campbell，感谢他对细节的注意、他的建议和他的反馈意见。

我还要感谢Karin Arrigoni为本书编写了索引，感谢Pat Christenson、Owen Wolfson和Amy Hassos为本书的印刷和排版所做的一切。

技术编辑Brent Knigge，感谢他诚实的观点、严格的把关和独到的市场眼光。还要感谢他一直以来在本书的支持论坛上为读者们提供的帮助。

感谢本书的读者，你们的兴趣给了我工作的快乐。

最后，感谢Jessica。一切尽在不言中。

Andi向以下诸位表示感谢。

Larry Ullamn。谢谢他同意和一位初出茅庐的作者合写这本书，感谢他在写作方面给我提出的宝贵建议。

我的同事。他们对C++的优点和缺点的讨论引人入胜，给了我很多启发。

我的朋友们。感谢他们对书中示例代码的讨论以及提供的宝贵意见，也感谢他们把我从计算机前拉走去清醒一下头脑。

引　言

我们认为，程序员选用C++语言来编写程序的理由不外乎两种，由此可以把C++程序员大致划分为两类：第一类是那些一开始就学C++语言的人们（或者是在学校，或者是工作以后）；第二类是那些对编写软件有兴趣，并认为C++最适合完成其工作的人们。不管你属于哪类，我们都不会让你失望——无论是对C++语言，还是对这本书。

C++已经有很多年的历史了，虽然在它之后又出现了Java和C#之类的新语言，但它至今仍是人们开发软件时的最佳选择之一。那些巨头中的巨头，如微软、Adobe、Sun、英特尔、亚马逊、Google、苹果、诺基亚等公司，都在使用C++。这门语言相对比较容易使用（选用本书作为入门教材就更是如此了☺），而且具有十分强大的威力。现在，你可以用它编写出一些基本的程序，而几个月后就会写出很不错的程序了。

虽然C++是一种专业开发人员的程序设计语言，但那些没有经过正规培训或者只具备普通计算机水平的人们也可以掌握它。我们是按照“无需任何预备知识”和“读者想要知道的都在这里”这两条标准来编写这本C++入门级教程的。你们不需要具备任何程序设计经验（包括C语言方面的经验），只要按照书里给出的示例程序和解释来学习，就可以迅速掌握许多实实在在的真本事。

什么是 C++

了解C++必须从C语言开始。C语言出现于20世纪70年代，它向程序员提供了一种全新的、宝贵的工具（C语言又起源于B语言，但我们没必要追溯到那么遥远）。C语言的两大主要优点是性能好和可移植性强。与其他程序设计语言相比，用C语言编写出的程序往往更简洁、更快，而绝大多数C语言代码可以轻而易举地在许多操作系统上使用。

C语言是一种过程式语言（procedure language），其意思是计算机命令是按照顺序执行的。这本身并没有什么不好，但随着程序数量的增加和软件规模的扩大，依靠“过程”来编写程序会让软件开发工作的效率变得越来越低。

C++语言是美国贝尔实验室的工程师Bjarne Stroustrup在20世纪80年代创建的。作为C语言的增强版本，C++在保留了C语言的全部优点（执行效率高，可移植性强，能够在众多低档计算机上运行）的同时，还增加了如下优点：

- 支持对象和OOP（参阅下页“什么是OOP”）；
- 能够显著提高程序员的工作效率；
- 解决了C语言中的常见问题。

但所有这些并不意味着C语言没必要存在了，只是作为其升级版本的C++更完善而已。这把我们

带到了下一个话题……

虽说C++脱胎于C语言，但在学习本书时并不需要你了解C语言。如果你已经掌握了C语言，那当然没有什么坏处，但你很快就会发现，只要能够在C++里找到更好的解决方案，我们就会抛弃那些既容易出问题又陈旧过时的C语言技巧（比如使用C++字符串来取代C语言中的字符串）。C语言仍是C++的一个子集，这意味着绝大多数用C语言编写出来的代码在C++环境里仍是合法的，但本书的重点将是如何正确地使用C++来编写程序。

什么是OOP

C语言和C++语言之间的一个主要区别，是能否支持OOP（object-oriented programming，面向对象编程）技术。在20世纪70年代末到80年代初，出现了一些新的面向对象的程序设计语言，比如Smalltalk。OOP关注的焦点是数据而不是逻辑。虽然OOP本质上与过程式程序设计技术只是一个事物的两面，但它对软件开发方式的影响却是十分深远的。

OOP要求程序员先找出需要解决的问题，再把问题表述为一个“类”（class）。在具体解决某个特定的问题时，程序员需要创建一个相应的类的“实例”（instance），这个实例就叫做“对象”（object）。对象是一种特殊的变量类型，同时包含某种数据和操纵这种数据的方式。OOP的内涵当然远不止这些，在处理复杂问题的时候更能体现其优势，其中最重要的是能让你的代码更容易地重用。

OOP继而引出了“泛型编程”（generic programming），这也是C++所支持的。泛型编程技术能够让程序员编写出与数据类型无关的解决方案。我们将在第13章对此做深入探讨。

使用 C++编程的步骤

使用C++来开发应用程序是一个多步骤的过程。首先，你必须了解最终结果是什么：你的应用程序应该完成什么样的工作。只有明确了目标，你才可以更好地确定自己需要声明哪些变量、需要完成哪些功能，等等。本书里每一个应用程序示例的开头都增加了一段简明的注释，来说明它的具体用途。

对初学者而言，下一步将是开始编写有关的C++源代码，这是一个普通的文本文件，如代码清单i-1所示。很明显，本书的重点是教会你需要敲入哪些代码才能创建出你所想要的应用程序。

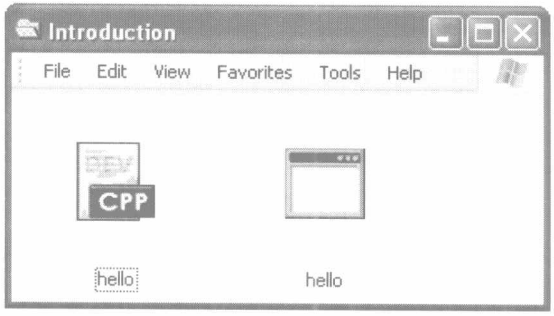
代码清单i-1 最基本的C++源代码文件的格式

```
// hello.cpp - Script i.1
// This is a sample
// C++ file.

#include <iostream>

int main() {

   // Say hello.
   std::cout << "Hello, World!";

   return 0;

}
```

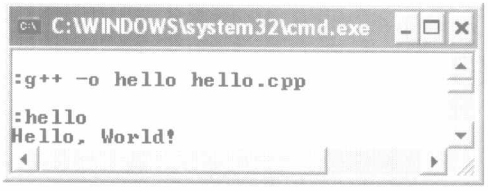
在编写好源代码之后，你将需要对它进行编译。编译是把一个包含着C++代码的普通文本文件转换为一个能够让计算机识别和执行的指令集合的过程。如果编译成功，其结果是一个可执行的程序文件，而编译步骤的目的就是为了让这个程序文件能够在那台计算机上执行（如图i-1所示）。

如果编译步骤未能成功完成，则需要对你的代码进行调试、调试、再调试。出现问题的原因可能是有拼写错误、遗漏了某个特定的字符或者误用了某个函数。可以告慰大家的是，本书里的示例代码都进行了调试。换句话说，只要你准确地按照书中的指示去做，代码就能正确运行。

有了编译好的应用程序之后，你就可以像对待其他的应用程序那样，双击其图标去运行它了。本书里的所有示例程序都将在一个控制台或终端窗口里运行（如图i-2所示）。



图i-1　hello程序（右图）是hello.cpp文件（左图）的编译结果



图i-2　在Windows系统上编译和运行hello程序

需要准备些什么

使用C++是免费的，对机器的要求也很低。对初学者来说，只要有一台计算机就行，你可能早就知道这一点了。你的电脑是什么样的、它运行的是哪一种操作系统、它的内存和可用的硬盘空间有多少都不重要。如果你的电脑能运行（比如说）Mozilla Firefox，你就可以用C++创建应用程序。

最重要的要求，事实上也是唯一的要求，是你的电脑里必须有一个合格的C++编译器。它可以是简单的g++（gcc的一部分），可以免费获得，在绝大多数操作系统上都可以运行，绝大多数派生自Unix的操作系统都包含了它。除了C++编译器，你还需要有一个文本编辑器和一个用来运行C++编译器的命令行界面。这两个工具在所有的操作系统里都有。

虽然可以用文本编辑器和编译器来创建应用程序，但最好还是使用一个完备的IDE（integrated development environment，集成开发环境），比如Windows环境下的Dev-C++或是Mac OS X环境下的Xcode，这两种工具都可以免费获得。使用IDE，你可在同一个界面下完成编写、编译、调试和执行C++代码的工作。附录A介绍了这两种工具的安装。在第1章里，你们将了解这两种工具的基本用法。我们强烈建议大家根据自己的操作系统选用Dev-C++或Xcode。本书里的每一个应用程序示例都遵循最新的C++标准（参阅注解），并已经在上述两种IDE环境里进行过测试。如果你使用的是Dev-C++或Xcode，那么试用本书的应用程序示例就会顺利许多。

至于你，本书的读者，除了学习C++的兴趣和愿望，你不需要再准备任何东西。

C++标准

有些技术，比如C++，是由一个专门的机构来管理的，该机构负责决定哪些功能和语法可以或不可以接受。通过这样的做法，这些机构就可以在应用户要求而增加新功能的同时保证该技术的稳定性（以便让那些现有的应用程序仍可以使用）。

因为相对比较“年轻”，关于C++的标准只有两个（不像C语言那样有一大批各种各样的标准）。第一个正式的C++标准是在1998年定义的。第二个C++标准是在2003年推出的，但它的大部分内容是对第一个C++标准中的已知错误进行修正。不管怎么说，我们在本书里遵守这个最新的标准，这意味着你们用过的任何符合这个标准的工具都可以毫无问题地用来运行本书里的代码。

因为C++是C语言的一个扩展，还因为有许多C++代码要用到各种C语言函数库，所以我们就必须考虑关于C语言的标准。最新的C语言标准叫做C99，它是在1999年发布的。

C++管理委员会已经在研究下一代C++标准了，新标准的代号是C++0X，预计在2009年前后推出。[①]

关于本书

因为C++脱胎于C语言，所以有许多关于C++程序设计的教科书都要求其读者具备一定程度的C语言知识。[②]本书没有这样的要求。我们将在本书的前几章对C和C++的基本知识进行介绍，但是从C++方面出发的。你们从本书里不会学到C语言的全部细节，但可以学到足够的C语言的基本知识以使用C++来开发各种有实际用途的应用程序。

与大多数C++程序设计语言的教科书一样，本书的结构有时也会有点儿奇怪。因为C++的全部目的是为了让软件开发工作变得更容易，所以一些看起来非常复杂的记号其实不难懂。因此，你们偶尔会在前面的章节里遇到一些我们还没有详细介绍的新概念——因为它们很容易实现。我们在此提及，让读者心里有这样一个底：虽然我现在还不能明白其中的奥妙，但再过一段时间这些疑难就会迎刃而解。这虽然略显怪异，但与其为了减少概念上的混乱而在刚开始采取不太准确的讨论，还不如这么讨论C++更好一些。

我们希望通过本书把最基本的C++编程技术介绍给大家，但不打算过于深入细节或是用那些不太常用的技术难点把读者搞得头晕脑胀。本书使用了以下一些体例。

首先，我们会一步一步地告诉你们应该敲入哪些代码或是可以采取哪些别的步骤。需要你们从键盘敲入的文本是用如下字体印刷的：

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

因为本书的宽度比常用的文本编辑器或IDE的窗口要小，某些步骤里的代码行在书中被分成了两行甚至更多行，但它们在编辑器里是不应该分断开的。我们在书中会用一个小箭头来表明这种分断，如下所示：

```
std::cout << "Hello, world! How are you
→doing on this fine Sunday afternoon?";
```

① C++0X标准草案的制订已于2008年10月完成，进入最后的标准审批和投票阶段，预计将于2009年或2010年正式发布。——编者注

② 事实上，包括Stroustrup在内的许多C++专家都建议直接学习C++，而不先学C。——编者注

对于这样的代码，你们应该把它们连续输入到同一行里，否则可能会发生错误。

我们将每个程序的完整C++代码单独保存为一个文件，并编了行号以方便大家查阅（参见代码清单i-1）。那些行号用不着你亲自输入，因为那反而会让你的代码无法使用。绝大多数好的文本编辑器和IDE都可以替你完成这个编号的工作。在代码清单里，与正在介绍的新概念有关的节用黑体字突出显示，以提醒大家注意。

书中会有许多窗口截图，它们或者是某个程序的运行结果（如图i-2所示），或者是需要输入的命令（图i-3所示），或者是某程序的某个特定部分。所有这些截图都取材于Windows或Mac OS X系统（Mac OS X系统的截图和操作步骤也适用于绝大多数Unix和Linux系统）。同一个程序在不同的计算机上运行时，其窗口画面很难做到完全相同，但其主要内容应该差不多，而其功能应该是完全相同的。

最后，我们要告诉大家本书没有哪些东西：本书没有像其他一些编程教科书那样，在每一章的末尾为读者准备一些思考题或练习题。这套系列丛书不采用这种体例。但你们将看到一些关于如何改进或应用有关技巧的建议，更重要的是，你们将从实际出发循序渐进地学会C++。

图i-3　在Mac OS X系统的终端窗口里调用g++编译器

提问的智慧

无论你是想在本书的支持论坛上发布消息、给笔者发电子邮件，还是在某个新闻组里提问题，要知道怎样最有效地提出你的问题，才能较快地收获高质量的回信。如果你想在最短的时间里获得最好的答案，请按以下步骤进行：

1. 搜索因特网，查阅随机手册，浏览与你的问题有关的各种文档。
2. 选择最恰当的论坛（新闻组、邮件列表等）上提出你的问题。
3. 给你的求助信加上一个简明扼要的标题。
4. 详细描述你的问题，完整地给出有关的代码，说清楚是什么地方出了问题，包括你正在运行的操作系统和你正在使用的开发环境（IDE、编译器等）。

如果你想得到更多的提示和启发，请到www.catb.org/~esr/faqs/smart-questions.html上好好读读由Eric Steven Raymond编写的“How to Ask Questions the Smart Way”（提问的智慧）。你在那里花费10分钟，可以让你在以后省下几小时的时间！

如何获得帮助

虽然本书是以最务实、最基本和最容易上手的原则编写的，但你们在学习过程中难免会遇到一些问题，需要一些帮助。下面是一些求助手段，我们按响应速度的快慢排列（速度快的列在前面）。

- 搜索因特网。

  如果你的问题与某个特定的函数、头文件或概念有关，Google往往能让你立刻找到答案。
- 使用C++新闻组或论坛。

  附录B列出了一些可以去求助的地方。如果你提出问题的方式足够聪明（参阅注解），就应该能在比较短的时间里得到想要的答案。
- 访问本书的支持网站。

  本书的官方Web站点可以在www.DMCInsights.com/cppvqs处找到。其中可以查到本书里的所有代码清单、到其他资源的链接以及一份勘误表。
- 访问本书的支持论坛。

  在本书的支持网站上，可以找到一个支持论坛。读者可以在那里提出问题、获得答案、看其他人在做什么，等等。这个论坛由作者本人负责管理，只要没有特殊情况，我们就一定会回答你们提出的问题。
- 给作者发电子邮件。

  如果其他办法都没效果，我们欢迎大家发送电子邮件到cppvqs@DMCInsights.com。但必须声明在先：我们不可能替你完成你的工作，不会替你调试你熬夜编写出来的200行代码，而且我们可能需要几天的时间才能给你们回信。不过，只要你给我们发来邮件，我们就一定会回复的，我们将尽最大的努力来帮助你。

目　录

第1章 创建基本的程序

用C++编写程序需要多个步骤。首先需要明确这个程序的用途和需求。只有在搞清楚那个程序要做些什么之后，才可以开始编写代码。完成编程工作之后，还需要把那些C++代码编译成可执行程序。如果编译成功，就可以运行这个可执行程序，查看你辛勤劳动的成果了。当然，在实际工作中，这个过程往往会穿插着调试、调试、再调试以及再多点儿调试等步骤。

这一章将讲解如何编写、编译和运行C++程序。本章所提供的信息和所描述的技巧是本书所有后续内容的基础，也是你的C++程序员职业生涯的基础。

本章先介绍创建一个简单的C++源文件所需要的基本语法和指令，然后讲解如何编译和执行这个程序。接下来，介绍如何用C++来输出文本消息和如何暂停某个程序的执行。最后两个编程技巧是如何在代码里使用各种空白字符和注释。掌握了所有这些基础知识之后，我们将向大家介绍如何使用两种流行且免费的开发工具来创建基本的C++程序。

1.1 C++的基本语法

如果你从未编写过程序，那么可能会感叹创建一个简单的应用程序如此轻而易举。C++代码是可以在任何一种文本编辑器中编写的普通文本。这种文本文件通常使用.cpp作为扩展名，对这种文本文件进行编译（类似将在下一节看到的那样）就能创建出一个可执行程序。

C++源文件的基本语法如下所示：

```
int main() {
    return 0;
}
```

这三行代码定义了一个名为main的函数。当C++程序开始执行时，系统将自动调用这个函数。需要特别注意的是，C++通常是区分字母大小写的，所以必须严格按照书里的样子输入代码。如果输入的是Int而不是int，或者输入的是Main而不是main，再或者输入的是RETURN而不是return，那么程序将无法运行（在编译程序时就会看到错误）。

函数的内容（将在程序执行时发生的事情）必须放在左、右花括号之间，这个区域叫做*函数体*。作为一种约定俗成的格式，函数体通常会整体缩进一些，这可以让它与函数的关系更清晰明显。

这个例子里，main()函数只做了一件事：返回数值0。在后面的章节里，你将学到更多关于让函数返回各种值的知识，但就目前而言，只要记住以下两件事就行了。

1. 函数所返回的值（无论是数值、字符还是任何其他东西）的类型必须与函数的定义相吻合。在

这个例子里，main()前面的int表明该函数将返回一个整数。

2. 作为一种实践中的标准做法，让main()函数返回一个0值意味着没有发生任何错误。

为了让大家熟悉编写和编译C++文件的过程，本书的第一个示例程序将只包含这些少到不能再少的代码。在动手实践之前，请参见“C++工具”注解里对各种C++开发工具的介绍。

C++工具

创建C++程序需要文本编辑器（用来录入C++代码）和编译器（用来把C++代码转换为可执行程序）。你只需准备这两样东西就足够了，好在每一种操作系统上都有许多种免费的文本编辑器和编译器可供选用。

话虽如此，大部分专业的C++程序员都会使用某种IDE来编程。这类软件可以提供所有必要的工具（文本编辑器和编译器）以及其他一些便利的辅助工具，比如内建的调试器等。使用IDE的另一个好处是，可以通过图形化用户界面（graphical user interface，GUI）来编译和运行程序。

了解如何使用标准化（或者说老式的）工具和一站式IDE是很有用的，所以本章将先向大家介绍前者，然后再介绍后者。在前者的例子里，我们将使用文本编辑器和命令行编译器来创建、编译和运行C++程序。对于后者，你将看到如何使用免费的Dev-C++（适用于Windows操作系统）软件和Xcode（适用于Mac OS X操作系统）软件来做同样的事情。在后面的各个章节里，选用哪种工具将无关紧要，因为我们的讨论重点将全部集中在C++代码上。

创建C++源文件

1. 打开文本编辑器。

C++程序是从以普通文本为内容的源文件开始的。这类文件可以用任何一种允许把普通文本保存为文件的文本编辑器（例如Windows上的“记事本”）来编写。很明显，有些编辑器（例如，Mac上的BBEdit程序，Unix上的vi或emacs程序）要比其他的更好用。有些文本编辑器（例如TextEdit或WordPad）会把文件默认保存为Rich Text Format（.rtf）格式或是给文件加上别的扩展名。正因如此，应该避免使用这些编辑器。

2. 创建一个空白的新文本文档。

3. 开始定义main()函数（代码清单1-1）：

```
int main() {
```

代码清单1-1　这三条语句是编写一个C++程序所需要的最少量的代码。它实际上是一个什么事情都做不了的程序

```
1   int main() {
2     return 0;
3   }
```

这行代码开始定义一个名为main的用户定义函数，它将返回一个整数值。跟在main()函数后面的左花括号标志着函数体的开始。

4. 给这个函数增加一条return语句：

```
return 0;
```

C++中的许多函数都会返回一个值，即使它只是数值0。这个值所代表的含义是“没有发生任何错误”。注意，这行代码通常会缩进4个空格（或一个制表位），以表明它是这个函数的一部分。

5. 结束main()函数：

```
}
```

千万不要忘记这个右花括号！这标志着main()函数定义的结束。在键入这个右花括号之后，可能还需要再按一下键盘上的回车键——如果不这样做，在（某些编译环境下）编译这个程序时将会看到一条警告消息。

6. 把这个文件保存为first.cpp。

记住，C++源文件使用.cpp作为扩展名。我们之所以把它命名为first，是因为它是本书中编写的第一个程序。

为让计算机里的内容井井有条、便于维护，你应该创建一个专用的文件夹来存放所有的C++代码。还可以在这个文件夹里再创建一些子文件夹，分别对应本书的每一章。不管怎么做，千万要记住first.cpp文件的存放位置，因为在后面的步骤里还要用到这个位置。

✔提示

- 你也许见过一些使用其他扩展名的C++源代码文件。在Unix系统上，C++源文件可能会使用.C作为扩展名（小写的.c代表着C文件，Unix操作系统是区分字母大小写的）。你也许还见过.cc或.cxx之类的扩展名。我们在本书里将一直使用.cpp。
- 不同的操作系统对文件名的长度有不同的限制。MS-DOS把文件的基本名（比如说，不带扩展名.cpp的first）限制为不超过8个字符；一些非常老的Unix版本把它限制为不超过14个字符，但Windows、Mac OS X和绝大多数现代的操作系统都支持长文件名（长度当然要合理）。
- Windows和Mac OS X都不区分文件名里的大小写字母，但Unix区分。按照区分字母大小写的情况来命名文件是一种良好的编程习惯，它可以让代码有更好的可移植性。具体到这个例子，你应该把文件命名为first.cpp，而不是first.CPP或First.cpp。
- 在C++里，完全可以省略main()函数里的return 0语句，因为如果没有另行指定，C++会假设这个函数将默认地返回一个0值。不过，作为一种良好的编程习惯，应该编写return语句。（我们之所以提到这一点，是因为你可能在其他资料里看到过省略它的情况。）
- 为了明确地表明某个函数不带输入参数，有些程序员会像下面这样来定义函数：

  ```
  int main(void) { ...
  ```

  输入参数到底是什么会随着学习的深入而逐渐明朗，现在只要知道int main()和int main(void)都是对的就行了。

1.2 编译 C++程序

简单地说，编译器的基本用途是把C++源代码转换为一个可执行程序。它可以获取高级指令并生成计算机能理解的低级机器代码。每个C++源代码文件都必须经过编译才能执行，只要对某个源代码文件进行了修改，就必须重新编译它。

有两种办法可以用来编译和运行C++代码：

- 使用命令行提示符和一个独立的编译器；