Apress

# C语言入门经典

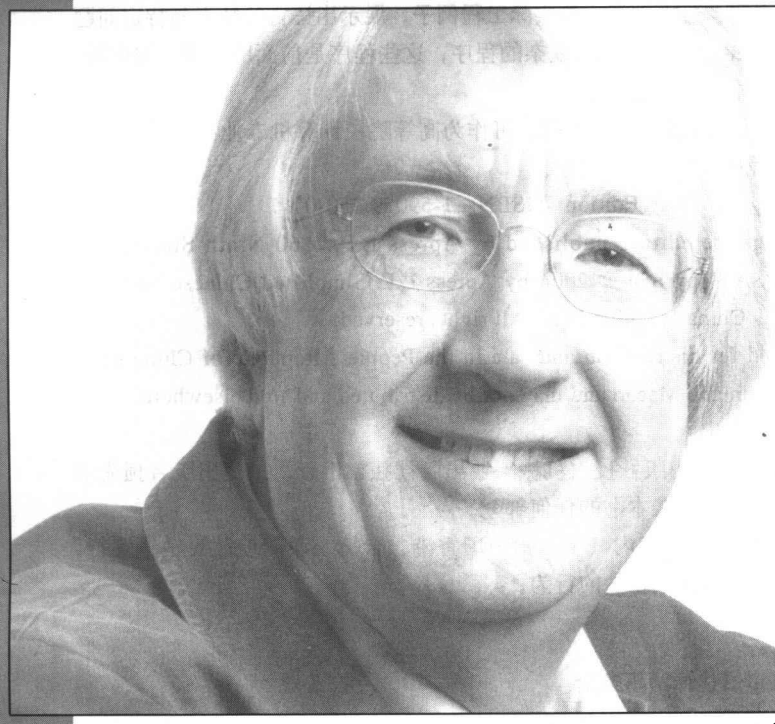
（原书第3版）

*Beginning C, Third Edition*

（美） Ivor Horton 著

张 欣 等译

机械工业出版社
China Machine Press





# C语言入门经典

（原书第3版）

*Beginning C, Third Edition*

（美） Ivor Horton 著

张 欣 等译

机械工业出版社
China Machine Press

本书集综合性、实用性为一体，系统介绍C语言及程序设计方法。书中不仅讲解了编程的入门知识和C语言基础内容，还提供了一些实际工程例子，展示出语言的特点与特定问题的关系；在每一章的最后都给出了一个稍微复杂的程序，这些程序是前面所学例子的应用，有助于读者领会在实际编程中C语言的应用。

本书内容深入浅出，循序渐进，讲解透彻。可作为高等院校计算机专业相关课程的教材或参考书。也适合广大程序设计爱好者自学。

Ivor Horton: Beginning C, Third Edition (ISBN: 1-59059-253-0).



**本书版权登记号：图字：01-2004-6886**

**图书在版编目（CIP）数据**

C语言入门经典（原书第3版）/（美）霍顿（Horton, I.）著；张欣等译. -北京：机械工业出版社，2007.5

书名原文：Beginning C, Third Edition

ISBN 978-7-111-21162-4

Ⅰ.C…　Ⅱ.①霍…　②张…　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035747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：李东震

北京京北制版印刷厂印刷 · 新华书店北京发行所发行

2007年5月第1版第1次印刷

186mm × 240mm · 28.25印张

定价：55.00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

# 译者序

有关C语言编程的书籍国内已有许多（包括国外教材的中译本），且侧重点各不相同。C语言入门并不难，但是，正如许多人认为的那样，掌握程序设计语言的最困难之处是用其灵活高效地开发实际软件系统，这需要大量的实践和学习，而C语言因其丰富的功能和复杂的特点更是如此。

Ivor Horton先生一直致力于研究C、C++、Java等一系列计算机编程语言，并著有多本畅销教材。作者以切身经验为基础，在本书中详谈了如何学好C语言编程。通过学习本书，程序员可以解决好以下几个问题：掌握遍布C程序设计语言与环境中的相关术语；弄清C语言元素的用法而不仅仅是了解它们；熟悉如何在实际的应用场合合理地使用C语言。本书中目标是，让初学者深入了解复合的和规模较大的程序例子是如何工作的，而不仅仅介绍只有几行代码的小程序。因此作者经常会在跨越几章节的范围内，构造实用的程序例子。通过这种强度的训练，初学者可以掌握管理代码的方式以及如何综合运用编程语言的特性等内容。最后，作者告诫读者，通过钻研一本编程语言的书籍去掌握该程序设计语言的过程是相当艰难和曲折的。因此，初学者必须在充满信心的同时，练习编写程序，以享受编程所带来的乐趣。

本书从C语言的基本概念出发，深入浅出地讲述了程序设计及开发的思想与方法，对每一部分的知识点、概念、难点，都力求以较为细腻的方式和浅显易懂的实例进行讲解。同时，对每一个知识点都配备了相应的趣味实例，最终指引读者全面掌握C语言编程思路与开发技巧。

本书由张欣组织翻译，参与本书翻译的还有白佳、卞雨桂、陈洁、成洁、杜鲲、黄璜、李才应、刘天成、刘吟、明卫军、潘秀燕、钱金蕾、王华红。

由于译者水平有限，纰漏在所难免，恳请读者批评指正。

# 前　言

欢迎使用本书。有了本书，你就会成为称职的C语言程序员。从许多方面来说，C语言都是学习程序设计的理想语言。它很简洁，因此无须学习大量的语法，就能够开始编写真正的应用程序。除了简明易学外，它还是一种功能非常强大的语言，至今仍被专业人士广为使用。C语言的强大之处体现在，它能够进行各种层次的程序设计，从硬件驱动程序和操作系统组件到大规模的应用程序，无所不包。事实上，任何计算机都可以使用C语言编译器，因此，一旦学会了C语言，就可以在任何环境下进行程序设计。最后一点，掌握了C语言，就为理解面向对象的C++语言奠定了良好的基础。

积极热情的程序员都必将面对三大障碍，即掌握适用于所有程序设计语言的术语，理解如何使用语言的元素（而不仅仅只知道它们是什么）以及领会如何在实际环境中应用程序设计语言，本书的目的就是将这些障碍降到最低。

术语是在专业人士与优秀的业余人士之间进行交流时必不可少的，因此掌握这些术语是必需的。我的方法是让你理解这些术语，并习惯在各种背景下使用它们。这样你才能更有效地使用大多数程序设计产品附带的文档，而且能自如地阅读和学习大多数程序设计语言的相关文献。

显然，理解语言元素的语法和作用是学习一门语言的关键，不过认识语言的特性如何发挥作用和如何应用它们，也同等的重要。在说明每种语言特性与特定问题的关系时，我通常采用实际应用的程序为例，而不只是代码片断。这些示例提供了实践的基础，你可以任意改动它们，研究改动后的效果。

只是理解应用于独立语言元素上的机制还不足以让你理解在特定背景中的程序设计。为了帮助你理解这一点，本书每章都以一个较复杂的程序作结，该程序应用了前面示例中的知识。这些程序有助于获得开发程序的能力和信心，可了解如何综合运用各种语言元素，构成大型的程序。最重要的是，它们能让你了解设计真实程序时会遇到的问题以及如何管理真实的代码。

学习任何程序设计语言，都要认识到三点。首先，尽管要学的东西很多，但是这意味着掌握它们之后，你会有极大的成就感。其次，学习的过程很有趣，你将会体会到这一点。最后，学习程序设计语言比你想像的容易得多，所以你肯定能掌握它。

## 如何使用本书

我认为动手实践是最好的方法，你将会立刻开始编写自己的第一个程序。每一章都有一些把理论应用于实践的程序，这些示例是本书的重点。建议读者输入并运行文中的示例，因为输入程序对记住语言元素有极大的帮助。此外，你还应该做每章后面的练习。当你第一次使一个程序运行起来，尤其是在试图解决自己的问题时，快速的进展会使你有很大的成就感。

刚开始，我们的进展不会太快，不过随着逐渐深入，我们会加快学习的速度。每一章都有

很多基础知识，因此你需要花些时间，在学习新的内容之前，确保理解了前面学过的所有知识。实验代码，尝试实现自己的想法，这是学习过程的一个重要部分。尝试修改程序，看看你还能让它们做什么，这才是最有趣的。不要害怕尝试，如果不明白某一点如何使用，多输入几种变体，看看会出现哪些情况。较好的学习方法是先通读整章，了解它涵盖的范围，然后再从头运行其中的所有示例。

你可能会觉得某些章末尾的程序非常难。如果第一次读这样的程序没有完全理解，不必担心。第一次难免会觉得难以理解，因为它们通常都是把你所学的知识应用到了相当复杂的问题中。如果你真的不能理解，可以略过那些章末尾的程序，继续学习下一章，然后再回头研究这些程序，甚至不必担心它们，通读全书都可以。之所以有这些程序，因为即使读完了本书，它们对你来说仍是非常有用的资源。

## 本书的适用对象

本书的目的是教你如何尽可能简单快速地编写有用的程序，如果你属于下列情况之一，那么本书就是你的指南手册：

- 刚接触程序设计，不过想直接深入了解C语言，从头开始学习程序设计及编写C语言程序。
- 以前做过一些程序设计，因此对其基本概念有一定了解，也许曾经使用过BASIC或PASCAL。现在想学习C语言，进一步提高自己的程序设计技能。

本书并未假设此前你对程序设计的知识有所了解，不过本书会很快地从基本概念深入到实战介绍。学完了本书，你就为自己的C语言程序设计奠定了全面的基础。

## 使用本书前的准备工作

要使用本书，需要一台安装了C语言编译器和库的计算机，这样才能执行书中的示例，此外为创建源代码文件，还需要一个程序文本编辑器。市面上有多种C语言编译器可供选择，Internet上也有很多共享版本。最理想的是选择与当前美国国家标准协会（ANSI）为C语言制定的标准一致的编译器。当今大多数C++编译器都能编译C语言程序，所以可以根据自己的需要进行选择。

可以采用任何纯文本编辑器创建源程序文件，如Notepad或Vi。不过，采用专为编辑C语言代码设计的编辑器更有帮助。不要使用Microsoft Word这样的字处理软件，它们通常会在文本中嵌入编译器不能理解的格式化代码。

要最大限度地发挥本书的功效，你需要有学习的意愿，成功的渴望，当学习不顺利、觉得前途渺茫时，还要有坚持下去的决心。几乎每个人在初次学习程序设计时都会在某处觉得迷茫。当你发现自己艰难地掌握了C语言的某个方面时，要坚持下去，迷雾一定会消散，你会觉得为什么当初我会不明白这一点呢？也许你明白要做到这些将会很难，不过我相信你一定会惊讶自己能在较短的时间内取得很大进步。我会帮助你开始自己的实践之旅，使你成为成功的程序设计员。

## 本书采用的规约

本书的文本和布局采用了许多不同的样式，以便区分各种不同的信息。大多数样式表达的含义都很明显。程序代码如下：

```
void main()
{
   cout << "Beginning C";
}
```

如果代码片段是从前面的实例修改而来的，修改过的代码行就用粗体显示，如下所示：

```
void main()
{
   cout << "Beginning C by Ivor Horton";
}
```

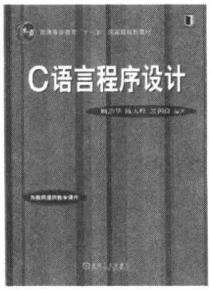
如果代码出现在文本中，那么它将采用不同的文本样式，如：`cout`。

程序代码中还使用了各种“括号”。它们之间的差别非常重要，不能互换。我称( )为圆括号，{ }为花括号，[ ]为方括号。

文本中的重要文字由黑体显示。

## 本书的代码

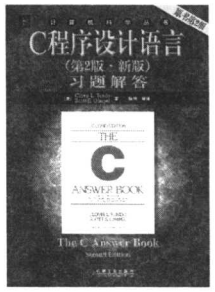
从Apress的站点`http://www.apress.com`可以下载本书中的所有代码。



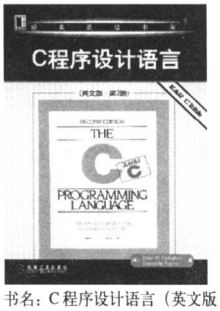

书名：C语言程序设计
ISBN：978-7-111-20761-0
作者：顾治华
价格：￥25.00


书名：C程序设计教程
ISBN：7-111-07952-3
作者：H.M.Deitel
价格：￥33.00

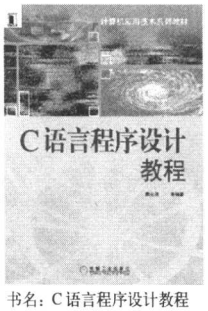

书名：C程序设计语言（第2版·新版）
ISBN：7-111-12806-0
作者：Brian W.Kernighan
价格：￥30.00



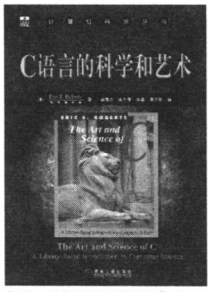

书名：C程序设计语言(第2版·新版)习题解答
ISBN：7-111-12943-1
作者：Clovis L
价格：￥15.00




书名：C程序设计语言（英文版·第2版）
ISBN：7-111-19626-0
作者：Brian W.Kernighan
价格：￥35.00

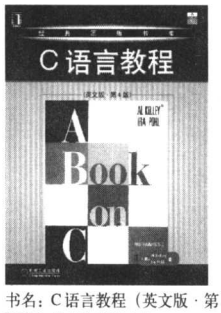



书名：C语言程序设计教程
ISBN：7-111-14403-1
作者：顾元刚
价格：￥29.00

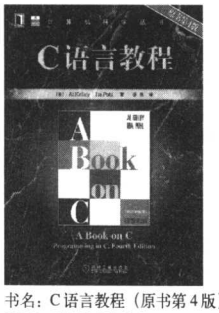



书名：C语言的科学和艺术
ISBN：7-111-15971
作者：Eric S.Roberts
价格：￥55.00


书名：C语言的科学和艺术（英文版）
ISBN：7-111-13991-7
作者：Eric S.Roberts
价格：￥60.00

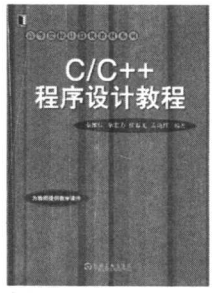



书名：C语言教程（英文版·第4版）
ISBN：7-111-13414-1
作者：Al Kelley
价格：￥65.00



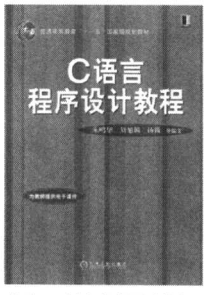

书名：C语言教程（原书第4版）
ISBN：7-111-20213-9
作者：Al Kelley
价格：￥45.00


书名：从问题到程序——程序设计与C语言引论
ISBN：7-111-16756-2
作者：裘宗燕
价格：￥36.00




书名：C/C++程序设计教程
ISBN：978-7-111-20609-5
作者：秦维佳等
价格：￥29.00


书名：C语言程序设计
ISBN：7-111-20078-0
作者：刘振安
价格：￥29.00




书名：C语言程序设计教程
ISBN：978-7-111-20683-5
作者：朱鸣华 等
价格：￥26.00


书名：C程序设计课程设计
ISBN：7-111-14619-0
作者：刘振安
价格：￥19.00


书名：C语言参考手册(原书第5版)
ISBN：7-111-12219-4
作者：Samuel P.Harbison Ⅲ等
价格：￥39.00

# 目　录

# 第1章 C语言程序设计

C语言是一种功能强大结构紧凑的计算机语言，用C语言编写出的程序能够正确地执行你想让计算机进行的操作。是你在控制一切，你创建程序（即一组指令），计算机将执行这些程序。

C语言的程序设计并不难，你马上会发现这一点。我会以令人愉快的、容易理解的方式讲解C语言程序设计的所有基础原理。学完了本章，你就能编写自己最初的几个C语言程序了。就是如此简单！

在本章你将学到：

- 如何创建C语言程序
- C语言程序是如何组织的
- 如何编写在屏幕上显示文本的程序

## 1.1 创建C语言程序

创建C语言程序有四个基本阶段或步骤：

- 编辑
- 编译
- 连接
- 执行

很快你就会发现创建C语言的手段就像你的另一双手（因为你将如此轻松、频繁地执行它们），不过首先需要了解每个步骤是什么，在创建C语言程序的过程中发挥什么作用。

### 1.1.1 编辑

这个步骤是创建和编辑C语言**源代码**（即编写的程序指令的总称）。有些C语言编译器附带了专用的编辑器，能够对管理程序提供很多帮助。事实上，编辑器通常会为编写、管理、开发和测试程序提供一个完整的环境。这种环境有时称为**集成开发环境**（Integrated development environment, IDE）。

你也可以采用其他编辑器创建源文件，不过必须把代码保存为没有嵌入任何格式数据的纯文本文件。如果你采用的编译器系统具有编辑器，那么这个编辑器通常会提供许多更易于编写和组织源程序的特性。它们通常是一些自动化的工具，能够正确地排出文本的布局，用各种颜色高亮显示重要的语言元素，这样不仅使代码更易读，在输入关键字出现错误时，也能明确地指出来。

如果你是在UNIX平台上工作，那么最常用的编辑器是`vi`。不过你可能更喜欢使用`emacs`编辑器。

对于PC来说，有很多免费的或共享的程序设计编辑器可供使用。它们通常能够高亮显示代码中的语法，对代码进行自动缩进，对确保代码的正确性很有帮助。不要使用Microsoft Word这样的字处理软件，它们不适合编写程序代码。

### 1.1.2 编译

编译器将把代码转换成机器语言，在编译过程中检测及报告代码中的错误。这个步骤的输入是在编辑阶段生成的文件，通常我们称这个文件为**源文件**。

编辑器能够检测出的错误的范围很广，这些错误可能是由无效的或不能识别的程序代码引起，也可能是结构上的错误，如程序的某个部分绝对不会被执行到。编译器的输出称为**目标代码**，存放目标代码的文件称为**目标文件**，这种文件的扩展名通常是.obj。在翻译过程中，编译器可以检测出几种不同类型的错误，它们中的大多数会阻止生成目标文件。

**注意**：在UNIX中，目标文件的扩展名是.o。

成功编译生成的结果是一个与源文件同名，但扩展名是.obj的文件。

如果在UNIX平台工作，编译C语言程序的标准命令是cc。可以如下使用cc命令：

```
cc -c myprog.c
```

其中myprog.c是你要编译的程序。注意，如果省略了-o标志，程序还会被自动连接。如果使用的是GNU的Not UNIX编辑器（GNU），应该输入下列命令：

```
gcc -c myprog.c.
```

成功编译的结果是一个目标文件。

大多数C语言编译器都有一个标准编译选项，这个选项可能出现在命令行中（如cc myprog.c），也可能出现在IDE（在其中你会发现一个Compile菜单项）的菜单项中。

### 1.1.3 连接

连接器将把编译器从源代码文件生成的各个模块联合起来，加上C语言提供的程序库中的代码模块，把所有代码连接成一个可执行的整体。连接器也可以检测并报告错误，例如，程序中的某部分缺失了，或者引用了不存在的库组件。

实际上，如果程序非常大，那么它可以由几个独立的源代码文件构成，然后将这些文件连接起来即可。大型程序很难用一段代码写完。把它分割成几个较小的源文件，可以使程序开发容易许多。源文件可以分别编译，这样有助于减少字面上的错误。此外，通常可以对程序进行增量式开发。每个源文件具有自己的文件名，整个程序由一个**项目名**引用，构成该程序的一组源文件集成在这个项目名下。

程序库提供的程序能执行不属于C语言的操作，从而支持和扩展了C语言。例如，程序库中包括执行输入和输出、计算平方根、对比两个字符串或获取日期和时间信息的操作的程序。

连接阶段失败，意味着需要重新编辑源代码。连接成功会生成一个可执行文件。在Microsoft Windows环境中，可执行文件的扩展名是.exe。在UNIX中，没有这样的扩展名，不过文件类型是可执行的。

在UNIX中，要连接的模块都列在cc命令中，例如：

```
cc myprog.c mod1.c mod2.o
```

这三个模块将被连接在一起。注意，最后一个模块的扩展名是.o。因为它之前被编译过了，扩展名.o告诉编译器该模块等待连接，不必再编译了。这个阶段输出的文件叫做a.out，应该把它重命名，使它的名字更有说明性。

该命令的另一种形式如下：

```
cc -o myprog myprog.c mod.c mod2.o
```

该命令将编译和连接模块myprog.c，创建可执行的文件myprog（在-o标志后直接定义）。

许多C语言编译器都有Build选项，可以一次完成编译和连接操作。通常在IDE的Compile菜单中可以找到这个选项，有时它有自己的菜单。

### 1.1.4 执行

执行阶段是在前面的处理都完成之后运行程序。遗憾地是，这个阶段可能生成各种各样的错误，从生成错误的输出，到不能运行，什么都不做，甚至使计算机崩溃，无所不有。一旦出现这些状况，就需要返回编辑阶段，检查源代码。

下面是个好消息，最终在这个阶段，你将看到计算机会精确地执行你的指令。

在UNIX和DOS中，要执行程序，只需输入已经编译和连接好的文件名即可。

在大多数IDE中会找到适当的菜单命令，Run或者Execute编译好的程序。该选项可能有自己的菜单，也可能位于Compile菜单项下。

在Windows中，可以用Windows Explorer找到程序的.exe文件，双击该文件即可执行它。

无论在何种环境中，用何种编译语言，开发程序的基本过程都是编辑、编译、连接和执行。图1-1总结了在开发C语言程序时，如何执行每个步骤。

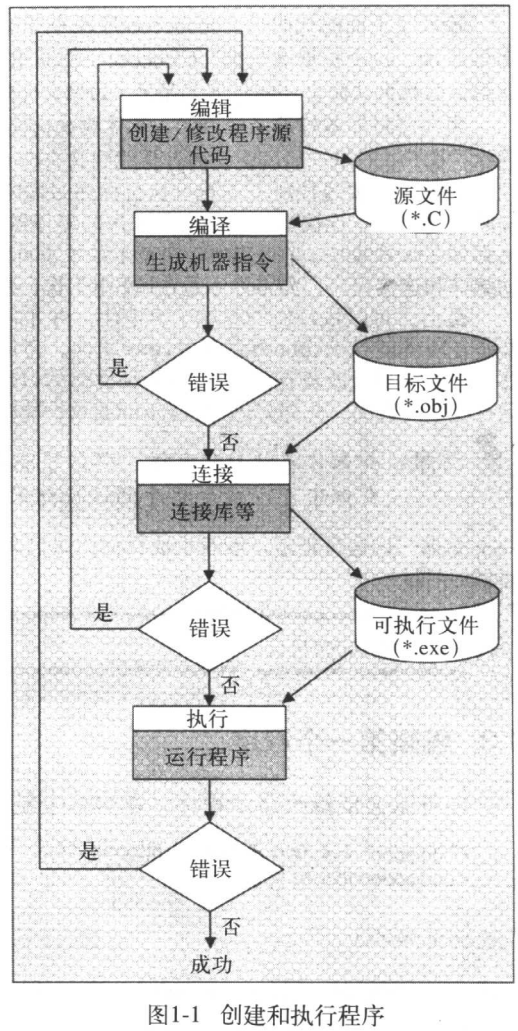



图1-1 创建和执行程序

## 1.2 创建第一个程序

是创建你的第一个程序的时候了。我们将执行创建一个简单C语言程序的每个步骤，从输入程序代码开始，到执行它为止。如果此时你不能理解自己所输入的代码，不必担心，我会逐步向你解释每个细节。

### 试试看：一个C语言示例程序

运行编辑器，正确输入下列程序。注意，输入的标点要和你看到的完全一样。第四行和最后一行的括号是花括号，既不是方括号，也不是圆括号，这点非常重要。此外，要确保用了正确的斜线（/），而不是反斜线（\）。不要忘了分号（;）。

```
/* Program 1.1 Your Very First C Program - Displaying Hello World */
#include <stdio.h>

int main()
{
        printf("Hello world");
        return 0;
}
```

输入了上面的代码后，把这个程序保存为hello.c。你可以用任何名字代替hello，不过扩展名一定要是.c。这个扩展名是编写C语言程序的通用规约。扩展名把文件的内容标识为了C语言源代码。大多数编译器都默认源文件的扩展名是.c，如果文件的扩展名不是.c，编译器可能会拒绝处理它。

现在已经准备好编译程序了。具体要如何操作，取决于你使用的编译器。如果你的C语言编译器可以让你在IDE中工作，那么应该很容易找到具有Compile选项的菜单。在UNIX中，编译命令是cc。

接下来，要连接所有创建可执行程序必需的各个程序段。这会加入程序所需的标准库中的代码。同样的，如何进行连接操作也是由你采用的编译器决定的。如果你使用的是UNIX，那么就把需要加入的模块都列在cc命令中。在IDE中，在某个菜单下可以找到Link选项。记住，许多IDE都有Build选项，能够自动编译和连接程序。如果你不是在IDE中工作，那么应该查询文档，找出运行连接器的命令。

最后，可以执行你的程序了。记住，有几种方法可以运行程序。如果你使用的是Windows，那么通常是在Windows Explorer中双击.exe文件。还可以从命令行编辑器中运行程序。只需要打开命令行编辑器，把当前目录改成存放程序的.exe文件的目录，然后输入程序名即可运行它。在IDE中，可能有直接运行程序的选项。此外，还可能有Run菜单，能够一口气执行编译、连接和运行程序的操作。

**注意：** *如果你在IDE中工作，那么可能需要使用Window菜单来改变Output窗口，在该窗口中可以看到程序的运行结果。*

如果一切运行正常，没有生成任何错误消息，你就成功了。这是你的第一个程序，你会在屏幕上看到下列消息：

```
Hello world
```

## 1.3 编辑第一个程序

你可以尝试修改这个程序，在屏幕上显示些其他消息。例如，可以试着编辑如下程序：

```
/* Program 1.2 Your Second C Program */
#include<stdio.h>

void main()
{
   printf("If at first you don\'t succeed, try, try, try again!");
}
```

要显示的文本中的\'序列叫做**转义序列**，它是在文本中加入单引号的专用方法。因为单引号用于说明一个字符常量的开头和结尾，所以当你需要一个单引号，而不是字符常量的起始字符时，需要专门的方法来说明这一点。很快你就会对转义序列有更多了解。你可以尝试重新编译这个你修改过的程序，重新连接它，再运行一次。有了前面的提示和一点点运气，你已经在编辑自己的第一个程序了。你已经用编辑器编写了一个程序，编辑了它，然后编译、连接并执行了它。

## 1.4 处理错误

是人都会犯错，所以出了错误没必要感到不好意思。幸运地是，计算机通常不会出错，而且很善于指出你的疏忽之处。编译器迟早会呈献给你一个源代码中的错误列表。在出错的语句处通常会有指示。如果出现了这种情况，就需要返回编辑阶段，找出错误代码中的问题，修正它。记住，一个错误可能会使原本正确的一系列语句都出现错误消息。如果一个语句引用的某种东西是具有错误的语句定义的，通常会出现这种情况。当然，如果一个语句定义的东西本身具有错误，那么定义的这个东西也不会正确。

让我们在程序中制造一个错误，看看程序代码不正确时会发生什么。编辑第二个示例程序，删除`printf()`结尾处的分号。如下所示：

```
/* Program 1.2 Your Second C Program */
#include<stdio.h>

void main()
{
   printf("If at first you don\'t succeed, try, try, try again!")
}
```

如果编译这个程序，就会看到错误消息，这些消息会根据使用的编译器而稍有不同。常见的错误消息如下：

```
Syntax error : missing ';' before '}'
HELLO.C - 1 error(s), 0 warning(s)
```

这个编译器能够精确地判断出错误是什么，错误在哪里。在`printf()`行的结尾确实应该有个分号。由于你刚开始编写程序，所以很可能在编译过程中遇到许多由标点符号引起的错误。太容易忘记加逗号或括号了，或者只是敲错了键。不必为此担心，即使经验丰富的程序员，在经过多年的实践之后，也会犯同样的错误。

如我前面所说的，一个错误可能会导致一串编译错误，编译器会为它认为不对的地方抛给你一堆错误消息。不要看到报告了大量的错误就感到沮丧。仔细研究这些消息后，基本的方法就是重新编辑源代码，修正那些你能理解的错误，忽略不能理解的错误。然后再次编译源文件。如果幸运地话，这次的错误就会少了。

要纠正示例程序中的错误，打开编辑器，重新输入分号即可。编译该程序，检查其他的错误，你的程序应该能够再次运行了。