

SAMS

• 畅销全球**20**余年

• 中文版累计销量过**50000**册

• C语言初学者的**最佳**轻量级教程

• 针对**C11**标准和**C标准库**全面更新

Sams Teach Yourself C
Programming in One Hour a Day
Seventh Edition

21天学通 C语言 (第7版)

[美] Bradley Jones 著
Peter Aitken
Dean Miller

姜佑 译

 人民邮电出版社
POSTS & TELECOM PRESS

21天学通 C语言 (第7版)

Bradley Jones
[美] Peter Aitken 著
Dean Miller

姜佑 译

人民邮电出版社
北京

图书在版编目(CIP)数据

21天学通C语言：第7版 / (美) 琼斯 (Jones, B.) ,
(美) 艾特肯 (Aitken, P.) , (美) 米勒 (Miller, D.) 著;
姜佑译. — 北京：人民邮电出版社, 2014. 11
ISBN 978-7-115-35537-9

I. ①2… II. ①琼… ②艾… ③米… ④姜… III. ①
C语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2014)第088990号

版 权 声 明

Bradley Jones, Peter Aitken and Dean Miller: Sams Teach Yourself C in One Hour a Day (7th Edition)
ISBN: 978-0789751997

Copyright © 2013 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Pearson Education, Inc.

All rights reserved.

本书中文简体字版由美国 Pearson 公司授权人民邮电出版社出版。未经出版者书面许可, 对本书任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。

◆ 著 [美] Bradley Jones Peter Aitken Dean Miller

译 姜 佑

责任编辑 傅道坤

责任印制 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

大厂聚鑫印刷有限责任公司印刷

◆ 开本: 787×1092 1/16

印张: 28.75

字数: 629 千字

2014 年 11 月第 1 版

印数: 1-4000 册

2014 年 11 月河北第 1 次印刷

著作权合同登记号 图字: 01-2013-5583 号

定价: 59.00 元

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

内容提要

本书是初学者学习 C 语言的经典教程。本版按最新的 C11 标准 (ISO/IEC 9899:2011)，以循序渐进的方式介绍了 C 语言编程的基本知识，并提供了丰富的程序示例和大量的练习。通过理论学习结合课后实践，读者将逐步了解、熟悉并掌握 C 语言。

本书共分为 4 部分。第 1 部分是 C 语言基础，介绍了 C 程序的组成、变量和常量、语句、表达式、运算符、函数、基本程序控制和信息读写；第 2 部分介绍了数值数组、指针、字符和字符串、结构、联合、typedef、变量作用域、高级程序控制、输入/输出；第 3 部分介绍了指针数组、链表、磁盘文件、操纵字符串、函数库、内存管理以及编译器的高级用法等；第 4 部分是附录，收录了 ASCII 表、C/C++ 关键字、常用函数以及习题答案。

本书针对初级程序员编写，可作为学习 C 语言的入门教程或参考资料。

作者简介

夏製容内

Bradley L. Jones

Developer.com 网站的管理者，负责管理 Developer.com、CodeGuru 和 DevX 等网站，有使用 C、C#、C++、SQL Sever、PowerBuilder、Visual Basic、HTML5 等开发系统的经验。他的推特是@BradleyLJones。

Peter Aitken

杜克大学医学中心的职员，负责开发供牙科医学研究使用的计算机程序。他是 IT 领域应用与编程方面的资深作家，在计算机杂志上发表文章 70 多篇，编写图书 40 多本。Aitken 目前是制药工程方面的顾问。

Dean Miller

在出版和授权消费产品业务方面有 20 多年经验的作者兼编辑。期间，他策划并推出了大量畅销书籍和系列，包括 *Teach Yourself in 21 Days*、*Teach Yourself in 24 Hours* 以及 *Unleashed* 系列，这些都由 Sams 出版社出版。

致 谢

感谢 Bradley Jones 和 Peter Aiken 编写了一本 C 语言编程指南的好书，他们有 20 多年的教学经验，教会成千上万的人如何使用 C 语言编程。感谢 Mark Taber 给我这次机会以新的格式策划本书，感谢 Mandie Frank、San Dee Phillips 和 Siddhartha Singh 提供原始文本，并将我添加的内容完善成更好的作品。以个人名义，感谢我的妻子 Fran，我的孩子 John、Alice 和 Margaret，感谢他们给予的爱与支持。将本书献给我的两个姐姐 Sheryn 和 Rebecca，她们在逆境中微笑面对生活的勇气时刻鼓励着我。

——Dean Miller

首先要感谢我的合著者 Brad Jones 付出的努力。我也非常感谢 Sams 出版社的所有人，篇幅有限，无法一一列举每个帮助我的人，请原谅。

——Peter Aitken

首先感谢我的妻子，在我编写本书的过程中对我的理解和支持。一本好书是多人努力的成果。感谢读者、编辑和其他所有提供帮助和对上一版给予反馈的人。我们针对反馈作了修订，相信本书会让学习 C 语言编程的人更容易上手。

——Bradley L. Jones

前言

从书名便可看出，通过学习本书，你可以自学 C 程序设计语言。在众多语言（如 C++、JAVA 和 C#）中，C 仍然是学习程序设计语言的首选。第 1 课中将详细介绍其中的原因。选择 C 作为程序设计语言是明智之举。

与市面上其他 C 语言的书籍相比，本书的讲解逻辑更清晰，初学者更容易理解。之前的 6 个版本一直在畅销书排行榜上遥遥领先，广受读者赞誉！本书为读者量身定制，每天只需花一小时便可学完一课内容。读者不需要有任何编程经验，当然，如果有其他语言的基础（如 BASIC），学起来会更快。本书的重点是介绍 C 语言，不会指定计算机和编译器。无论你的计算机使用的是 Windows 系统、Mac OS 系统还是 UNIX 系统，都可以学习 C 语言。

本书特色

本书包含一些特殊栏，有助于启发读者学习。语法框教你如何使用 C 语言中特殊的概念，提供精炼的示例和详尽的解释。可以查看下面的示例，预览语法框的概要。（你尚未学习第 1 课，不理解其中的内容没关系。）

语 法

```
#include <stdio.h>
printf( 格式字符串[, 参数, ...] );
```

printf() 函数接受一系列的参数。每个参数都在格式字符串中有相应的转换说明。printf() 将格式的信息打印在标准输出设备上（通常是显示屏）。使用 printf() 函数时，必须包含标准输入/输出头文件 stdio.h。

格式字符串必不可少，而参数是可选的。每个参数都必须有转换说明。格式字符串中可包含转义序列。以下所示的程序示例中调用了 prin 转换说明 tf()，并打印输出。

程序示例 1

输入

```
#include <stdio.h>
int main( void )
{
    printf( "This is an example of something printed!" );
}
```

输出

```
This is an example of something printed!
```

程序示例 2

输入

```
printf( "This prints a character, %c\n a number, %d\n a floating point,
        %f", 'z', 123, 456.789 );
```

输出

```
This prints a character, z
a number, 123
a floating point, 456.789
```

本书的另一大特色是 DO/DON'T 栏，其中指出读者应该做什么，不应该做什么。

DO	DONT
阅读本节余下内容，将解释本课末尾的课后研习部分。	请勿跳过随堂测验和课后练习。如果你可以完成本课的课后研习，说明你已准备好学习新的内容。

另外，本书还包含大量的提示、注意和警告栏。提示提供使用 C 语言的一些捷径和技巧。注意提供其他具体的细节，帮助读者更好地理解 C 语言的概念。警告提醒读者避免一些潜在的问题。

本书用大量的程序示例解释 C 语言的特性和概念，可以把这些示例用在自己的程序中。每个程序都分成 3 部分（输入、输出和分析）来讨论，并分别以黑体字和▼标出。

每课末尾的答疑部分包含本课相关内容的常见问题与回答。另外，每课还设有课后研习，包含小测验和练习题。小测验用于检测读者对本课概念的掌握程度。如果要核对或查看答案，请翻阅附录 D。

只阅读本书是学不会 C 语言的，如果你想成为一个程序员，就必须写程序。解答小测验中的问题是很好地练习，还应该尝试解答每一道练习题。写代码是学习 C 语言的最佳途径。

排错练习最有益。在 C 语言中，bug 是程序的错误。排错练习中的代码示例中包含一些常见错误（bug）。你要找出这些错误并修正它们。如果遇到任何困难，可以查阅附录 D。

随着进一步学习本书，有些练习的答案会很长，有些练习会有多种答案。因此，后面的课程可能不会提供所有练习的答案。

本书使用的约定

本书使用不同的字体，不仅帮助读者在阅读过程中区别 C 代码和常规英文，而且有助于读者识别重要的概念。本书中出现的 C 代码一律使用等宽字体（如，monospace）。在程序示例的输入和输出中，用户键入的内容均使用加粗等宽字体（如，**blod monospace**）。占位符（术语，代表代码中你实际键入的内容）用斜体等宽字体（如，*italic monospace*）¹。新术语和重要术语用斜体（如 *italic*）标出²。

¹ 译者注：译文采用仿宋体（如，仿宋体）。

² 译者注：译文采用楷体及中英并陈（如，函数（*function*））。

目 录

第1部分 C语言基础

第1课 初识C语言.....	1
1.1 C语言发展简史.....	1
1.2 为何要使用C语言.....	1
1.3 准备编程.....	2
1.4 程序开发周期.....	3
1.4.1 创建源代码.....	3
1.4.2 使用编辑器.....	3
1.4.3 编译源代码.....	4
1.4.4 链接以创建可执行文件.....	4
1.4.5 完成开发周期.....	5
1.5 第1个C程序.....	6
1.5.1 输入并编译hello.c.....	7
1.5.2 编译错误.....	8
1.5.3 链接器错误消息.....	9
1.6 小 结.....	9
1.7 答 疑.....	9
1.8 课后研习.....	10
1.8.1 小测验.....	10
1.8.2 练习题.....	11
第2课 C程序的组成部分.....	12
2.1 简短的C程序.....	12
2.2 程序的组成部分.....	13
2.2.1 main()函数.....	13
2.2.2 #include 和#define 指令.....	13
2.2.3 变量定义.....	14
2.2.4 函数原型.....	14
2.2.5 程序语句.....	14
2.2.6 函数定义.....	15
2.2.7 程序的注释.....	15
2.2.8 使用花括号.....	16
2.2.9 运行程序.....	16
2.2.10 补充说明.....	16

2.3 学以致用.....	17
2.4 小 结.....	18
2.5 答 疑.....	19
2.6 课后研习.....	19
2.6.1 小测验.....	19
2.6.2 练习题.....	19
第3课 储存信息：变量和常量.....	21
3.1 计算机的内存.....	21
3.2 用变量储存信息.....	22
3.3 数值类型.....	23
3.3.1 变量声明.....	26
3.3.2 typedef 关键字.....	26
3.3.3 初始化变量.....	26
3.4 常 量.....	27
3.4.1 字面常量.....	27
3.4.2 符号常量.....	28
3.5 小 结.....	31
3.6 答 疑.....	32
3.7 课后研习.....	32
3.7.1 小测验.....	32
3.7.2 练习题.....	33
第4课 语句、表达式和运算符.....	34
4.1 语 句.....	34
4.1.1 在语句中留白.....	34
4.1.2 创建空语句.....	35
4.1.3 复合语句.....	35
4.2 理解表达式.....	36
4.2.1 简单表达式.....	36
4.2.2 复杂表达式.....	36
4.3 运算符.....	37
4.3.1 赋值运算符.....	37
4.3.2 数学运算符.....	37
4.3.3 运算符优先级和圆括号.....	41

4.3.4 子表达式的计算顺序	43	第 6 课 基本程序控制	78
4.3.5 关系运算符	43	6.1 数组: 基本概念	78
4.4 if 语句	44	6.2 控制程序的执行	79
4.5 对关系表达式求值	49	6.2.1 for 语句	79
4.6 逻辑运算符	51	6.2.2 嵌套 for 语句	83
4.7 详议真/假值	52	6.2.3 while 语句	85
4.7.1 运算符的优先级	52	6.2.4 嵌套 while 语句	88
4.7.2 复合赋值运算符	54	6.2.5 do...while 循环	89
4.7.3 条件运算符	54	6.3 嵌套循环	92
4.7.4 逗号运算符	55	6.4 小 结	93
4.8 运算符优先级归纳	55	6.5 答 疑	94
4.9 小 结	56	6.6 课后研习	94
4.10 答 疑	56	6.6.1 小测验	94
4.11 课后研习	57	6.6.2 练习题	94
4.11.1 小测验	57	第 7 课 信息读写基础	96
4.11.2 练习题	57	7.1 在屏幕上显示信息	96
第 5 课 函 数	59	7.1.1 printf() 函数	96
5.1 理解函数	59	7.1.2 printf() 的格式字符串	97
5.1.1 函数定义	59	7.1.3 使用 puts() 显示消息	103
5.1.2 函数示例	59	7.2 使用 scanf() 输入数值数据	104
5.2 函数的工作原理	61	7.3 三字符序列	108
5.3 函数和结构化程序设计	62	7.4 小 结	109
5.3.1 结构化程序设计的优点	63	7.5 答 疑	109
5.3.2 规划结构化程序	63	7.6 课后研习	109
5.3.3 自上而下的方法	64	7.6.1 小测验	109
5.4 编写函数	65	7.6.2 练习题	110
5.4.1 函数头	65	第 2 部分 C 语言应用	
5.4.2 函数体	67	第 8 课 数值数组	112
5.4.3 函数原型	71	8.1 什么是数组	112
5.5 给函数传递实参	72	8.1.1 一维数组	113
5.6 调用函数	72	8.1.2 多维数组	116
5.7 函数的位置	75	8.2 命名和声明数组	116
5.8 内联函数	75	8.2.1 初始化数组	119
5.9 小 结	76	8.2.2 初始化多维数组	120
5.10 答 疑	76	8.3 小 结	123
5.11 课后研习	76	8.4 答 疑	123
5.11.1 小测验	76	8.5 课后研习	124
5.11.2 练习题	77		

8.5.1	小测验	124	10.7.1	用 gets()函数输入字符串	154
8.5.2	练习题	124	10.7.2	用 scanf()函数输入字符串	157
第9课	指针	126	10.8	小结	159
9.1	什么是指针	126	10.9	答疑	160
9.1.1	计算机的内存	126	10.10	课后研习	160
9.1.2	创建指针	127	10.10.1	小测验	160
9.2	指针和简单变量	127	10.10.2	练习题	161
9.2.1	声明指针	127	第11课	结构、联合和 typedef	163
9.2.2	初始化指针	128	11.1	简单结构	163
9.2.3	使用指针	128	11.1.1	声明和定义结构	163
9.3	指针和变量类型	130	11.1.2	访问结构的成员	164
9.4	指针和数组	131	11.2	复杂结构	166
9.4.1	数组名	131	11.2.1	包含结构的结构	166
9.4.2	储存数组元素	131	11.2.2	包含数组的结构	169
9.4.3	指针算术	134	11.3	结构数组	171
9.5	指针的注意事项	137	11.4	初始化结构	173
9.6	数组下标表示法和指针	137	11.5	结构和指针	175
9.7	给函数传递数组	137	11.5.1	包含指针成员的结构	175
9.8	小结	141	11.5.2	创建指向结构的指针	177
9.9	答疑	142	11.5.3	使用指针和结构数组	179
9.10	课后研习	142	11.5.4	给函数传递结构实参	181
9.10.1	小测验	142	11.6	联合	182
9.10.2	练习题	143	11.6.1	声明、定义并初始化联合	182
第10课	字符和字符串	144	11.6.2	访问联合成员	183
10.1	char数据类型	144	11.7	用 typedef 创建结构的别名	187
10.2	使用字符变量	145	11.8	小结	187
10.3	使用字符串	147	11.9	答疑	187
10.3.1	字符数组	147	11.10	课后研习	188
10.3.2	初始化字符数组	148	11.10.1	小测验	188
10.4	字符串和指针	148	11.10.2	练习题	188
10.5	未储存在数组中的字符串	148	第12课	变量作用域	190
10.5.1	在编译期分配字符串的空间	149	12.1	什么是作用域	190
10.5.2	malloc()函数	149	12.1.1	演示作用域	190
10.5.3	malloc()函数的用法	150	12.1.2	作用域的重要性	192
10.6	显示字符串和字符	153	12.2	创建外部变量	192
10.6.1	puts()函数	153	12.2.1	外部变量作用域	192
10.6.2	printf()函数	154	12.2.2	何时使用外部变量	192
10.7	读取从键盘输入的字符串	154	12.2.3	extern 关键字	193

16.5	课后研习	285	18.5	查找字符串	324
16.5.1	小测验	285	18.5.1	strchr()函数	324
16.5.2	练习题	286	18.5.2	strrchr()函数	325
第 17 课	磁盘文件	287	18.5.3	strcspn()函数	326
17.1	将流与磁盘文件相关联	287	18.5.4	strspn()函数	327
17.2	磁盘文件的类型	287	18.5.5	strpbrk()函数	328
17.3	文件名	288	18.5.6	strstr()函数	328
17.4	打开文件	288	18.6	将字符串转换为数字	329
17.5	读写文件数据	291	18.6.1	将字符串转换为整型值	329
17.5.1	格式化输入和输出	291	18.6.2	将字符串转换为 long	330
17.5.2	字符输入和输出	294	18.6.3	将字符串转换为 long long 类型值	330
17.5.3	直接文件输入/输出	296	18.6.4	将字符串转换为浮点值	330
17.6	文件缓冲: 关闭和刷新文件	299	18.7	字符测试函数	331
17.7	顺序文件访问和随机文件访问	300	18.8	小结	335
17.7.1	ftell()函数和 rewind()函数	301	18.9	答疑	335
17.7.2	fseek()函数	303	18.10	课后研习	336
17.8	检测文件末尾	305	18.10.1	小测验	336
17.9	文件管理函数	307	18.10.2	练习题	336
17.9.1	删除文件	307	第 19 课	函数的高级主题	338
17.9.2	重命名文件	308	19.1	给函数传递指针	338
17.9.3	拷贝文件	308	19.2	void 指针	341
17.10	临时文件	310	19.3	带可变数目参数的函数	344
17.11	小结	312	19.4	返回指针的函数	346
17.12	答疑	312	19.5	小结	348
17.13	课后研习	313	19.6	答疑	348
17.13.1	小测验	313	19.7	课后研习	348
17.13.2	练习题	313	19.7.1	小测验	348
第 18 课	操纵字符串	315	19.7.2	练习题	349
18.1	确定字符串长度	315	第 20 课	C 语言的函数库	350
18.2	拷贝字符串	316	20.1	数学函数	350
18.2.1	strcpy()函数	316	20.1.1	三角函数	350
18.2.2	strncpy()函数	317	20.1.2	指数函数和对数函数	350
18.3	拼接字符串	319	20.1.3	双曲线函数	351
18.3.1	strcat()函数	319	20.1.4	其他数学函数	351
18.3.2	strncat()函数	320	20.1.5	演示数学函数	351
18.4	比较字符串	321	20.2	处理时间	352
18.4.1	比较字符串本身	322	20.2.1	表示时间	352
18.4.2	比较部分字符串	323			

20.2.2	时间函数	353	21.7.1	小测验	386
20.2.3	使用时间函数	355	21.7.2	练习题	386
20.3	错误处理	357	第 22 课 编译器的高级用法 388		
20.3.1	assert()宏	357	22.1	多源代码文件编程	388
20.3.2	errno.h 头文件	359	22.1.1	模块化编程的优点	388
20.3.3	perror()函数	359	22.1.2	模块化编程技术	388
20.4	查找和排序	361	22.1.3	模块化的组成部分	392
20.4.1	用 bsearch()函数进行查找	361	22.1.4	外部变量和模块化编程	392
20.4.2	用 qsort()函数进行排序	362	22.2	C 预处理器	393
20.4.3	演示查找和排序	362	22.2.1	#define 预处理器指令	393
20.5	小结	367	22.2.2	#include 指令	397
20.6	答疑	367	22.2.3	#if、#elif、#else 和#endif	397
20.7	课后研习	367	22.2.4	使用#if...#endif 帮助调试	398
20.7.1	小测验	367	22.2.5	避免多次包含头文件	399
20.7.2	练习题	368	22.2.6	#undef 指令	399
第 21 课 管理内存 370			22.3	预定义宏	400
21.1	类型转换	370	22.4	命令行参数	400
21.1.1	自动类型转换	370	22.5	小结	402
21.1.2	显示转换	372	22.6	答疑	402
21.2	分配内存存储空间	373	22.7	课后研习	403
21.2.1	用 malloc()函数分配内存	374	22.7.1	小测验	403
21.2.2	用 calloc()函数分配内存	374	22.7.2	练习题	403
21.2.3	用 realloc()函数分配更多内存	375	第 4 部分 附录		
21.2.4	用 free()函数释放内存	377	附录 A	ASCII 表	405
21.3	操控内存块	378	附录 B	C/C++关键字	409
21.3.1	用 memset()函数初始化内存	378	附录 C	常用函数	411
21.3.2	用 memcpy()函数拷贝内存的数据	379	附录 D	参考答案	415
21.3.3	用 memmove()函数移动内存的数据	379			
21.4	位	380			
21.4.1	移位运算符	381			
21.4.2	按位逻辑运算符	382			
21.4.3	求反运算符	383			
21.4.4	结构中的位字段	383			
21.5	小结	384			
21.6	答疑	384			
21.7	课后研习	385			

初识 C 语言

欢迎学习本课程！本课将是你成为 C 程序员高手之路的开始。本课将介绍以下内容：

- 在众多程序设计语言中，为什么 C 语言是首选
- 程序开发周期中的步骤
- 如何编写、编译和运行第 1 个 C 程序
- 编译器和链接器生成的错误消息

1.1 C 语言发展简史

读者一定非常好奇 C 语言的起源及其名字的由来。1972 年，丹尼斯·里奇（Dennis Ritchie）在贝尔电话实验室发明了 C 语言。发明该语言的初衷是为了设计 UNIX 操作系统（现在用于许多计算机上）。从一开始，C 语言就致力于为程序员排忧解难。

C 语言因功能强大和灵活，很快便广泛应用于贝尔实验室以外的各领域。世界各地的程序员都开始用它来编写程序。不久，各个组织开始使用自己版本的 C 语言，但是各种版本 C 语言的实现存在细微差别，这让程序员头疼不已。为解决这一问题，美国国家标准协会（ANSI）于 1983 年成立了一个委员会，制定了 C 语言的标准定义，即 ANSIC 标准。当前的 C 编译器（极少例外）都支持这个标准。

注意

C 语言极少改动，最新的改动是 2011 年的 ANSI C11 标准。该标准在 C 语言中添加了一些新特性（本书已涵盖）。但是，旧的编译器可能不支持目前的最新标准。

为什么叫 C 语言？之所以叫 C 语言，是因为它的前辈是 B 语言。B 语言由贝尔实验室（Bell Labs）的肯·汤普逊（Ken Thompson）发明。至于为什么叫 B 语言就不言而喻了。

1.2 为何要使用 C 语言

如今，在计算机程序设计领域，可供选择的高级语言很多，如 C、Perl、BASIC、Java、PHP 和 C#。这些语言都能很好地完成程序设计任务，但是基于以下几个原因，使得 C 语言仍是众多计算机专业人士的首选。

- C 语言功能强大、十分灵活。你完全想不到 C 语言能帮你完成什么工作。语言本身不会给你带来任何限制。C 语言可用于各种操作系统、文字处理软件、图形、电子表格、甚至是其他语言的编译器。

- C 语言非常流行，是众多专业程序员的首选。因此，市面上很容易找到各种 C 语言的编译器和辅助工具。
- C 是一门可移植语言。可移植意味着在一台计算机系统（如，IBM PC）中写出的 C 程序，无需修改或少量修改便可在其他操作系统（如，DEC VAX 系统）中运行。除此之外，在微软 Windows 操作系统中编写的程序，也可移到 Linux 系统的机器中（无需修改或少量修改）。ANSI 标准进一步改善了 C 语言的可移植性，为 C 编译器添加了一系列规则。
- C 语言短小精悍，只包含少量关键字（*keyword*）。C 语言以关键字为基础构建语言的功能，C 语言编程离不开关键字。你可能认为一门语言中关键字（有时也称为保留字（*reserved word*））越多功能越强大，其实并非如此。
- C 语言是模块化的。C 代码可以（且应该）以例程形式编写，这些例程称为函数（*function*）。函数可复用于其他程序或应用。通过传递信息给函数，可以创建有用且能重复使用的代码。

正是由于上述特性，使得 C 语言成为程序设计语言的首选。那 C++ 怎样？你可能在别处了解过 C++ 和面向对象程序设计（*object-oriented programming*）技术。C 与 C++ 有何不同？是否应该自学 C++？

别担心！C++ 是 C 的超集，这意味着 C++ 中不仅包含了 C 的所有内容，还添加了面向对象程序设计的新特性。如果你继续学习 C++，会发现 C 中所学的知识几乎都可应用于 C++。学习 C 语言，不仅学了一门现在功能最强大、最流行的程序设计语言，而且还为学习面向对象程序设计做好了准备。

另一门备受关注的语言是 Java。Java 和 C++ 一样，也是基于 C 语言发展起来的。如果将来学习 Java，你会发现在 C 语言中学过的所有知识点都用得上。

最新的一门语言是 C#（发音为“See-Sharp”）。与 C++ 和 Java 类似，C# 也是一门源自 C 的面向对象语言。同样，在 C 语言中学到的内容可以直接用于 C# 程序设计。

注意

许多学过 C 语言的人，后来都会选择学习 C++、Java 或 C#。先学习 C 语言，再学其他语言会轻松许多。

1.3 准备编程

解决问题之前肯定要经过一些步骤。首先，必须定义问题。如果不知道问题是什么，根本不可能找到解决方案！知道问题所在，才能设法修正。如果你想到一个解决方案，通常会去实施它。之后，你必须知道问题是否解决了。这样解决问题的逻辑可以应用到各个领域，包括程序设计。

用 C 语言创建一个程序（或者说，用任意语言编写的计算机程序）时，应该遵循以下步骤的顺序。

1. 确定程序的目标。
2. 确定在编程时所使用的方法。

3. 创建程序解决问题。

4. 运行程序查看结果。

目标（步骤 1）可能是编写一个文字处理软件或数据库程序。简单的目标可以是在屏幕上显示你的姓名。没有目标就不可能写出程序，因此，如果要写程序就已经准备好第一步了。

接下来，第 2 步要确定你编写的程序中所使用的方法。你是否需要计算机程序来解决问题？要记录什么信息？程序中要使用什么公式？在这一步中，应该明确自己需要了解什么知识以及实施方案的具体步骤。例如，假设你要写一个计算圆面积的程序。第 1 步已经完成，因为目标明确：计算圆的面积。第 2 步，分析如何确定圆的面积。例如，假设用户提供圆的半径。了解这些以后，便可通过公式 πr^2 计算得出结果。现在，可以继续第 3 步和第 4 步，进入程序开发周期。

1.4 程序开发周期

程序开发周期的步骤是：第 1 步，使用编辑器创建磁盘文件保存源代码；第 2 步，编译源代码创建目标文件；第 3 步，链接已编译的代码创建可执行文件；第 4 步，运行程序，看看程序是否能按原计划运行。

1.4.1 创建源代码

源代码 (*source code*) 是一系列语句或命令，指导计算机执行特定的任务。如前所述，程序开发周期的第 1 步是在编辑器中输入源代码。例如，下面这行 C 源代码：

```
printf("Hello,Mom!")
```

这行语句命令计算机在屏幕上显示消息 Hello,Mom!（到目前为止，不用管如何执行语句）。

1.4.2 使用编辑器

绝大多数 C 程序都是在集成开发环境 (IDE) 中进行编译的，集成开发环境能让你在它的编辑器中输入程序并编译程序，同时还能调试和创建应用（这些概念今后会学到）。你完全不必使用额外的编辑器。尽管如此，绝大多数计算机系统仍包含可用作编辑器的程序。如果使用 Linux 或 UNIX 系统，可以使用 `ed`、`ex`、`edit` 或 `vi` 编辑器。如果使用微软 Windows 系统，可以使用 Notepad 或 WordPad。

大部分文字处理软件都使用特殊的代码格式化文档，在其他程序中无法正确读取这些代码。美国信息交换标准码 (ASCII) 几乎为所有的程序（包括 C）指定了一个标准文本。许多文字处理软件（如 WordPerfect、Microsoft Word、WordPad 和 WordStar）都能以 ASCII 形式保存源文件（将源文件作为文本文件而不是文档文件保存）。如果要以 ASCII 文件保存一个文字处理软件的文件，在保存时必须选择 ASCII 选项或文本选项。

如果不想用上述编辑器，也可以购买其他的编辑器。市面上有许多专门为输入源代码而设计的软件包（包括商业的和共享软件）。