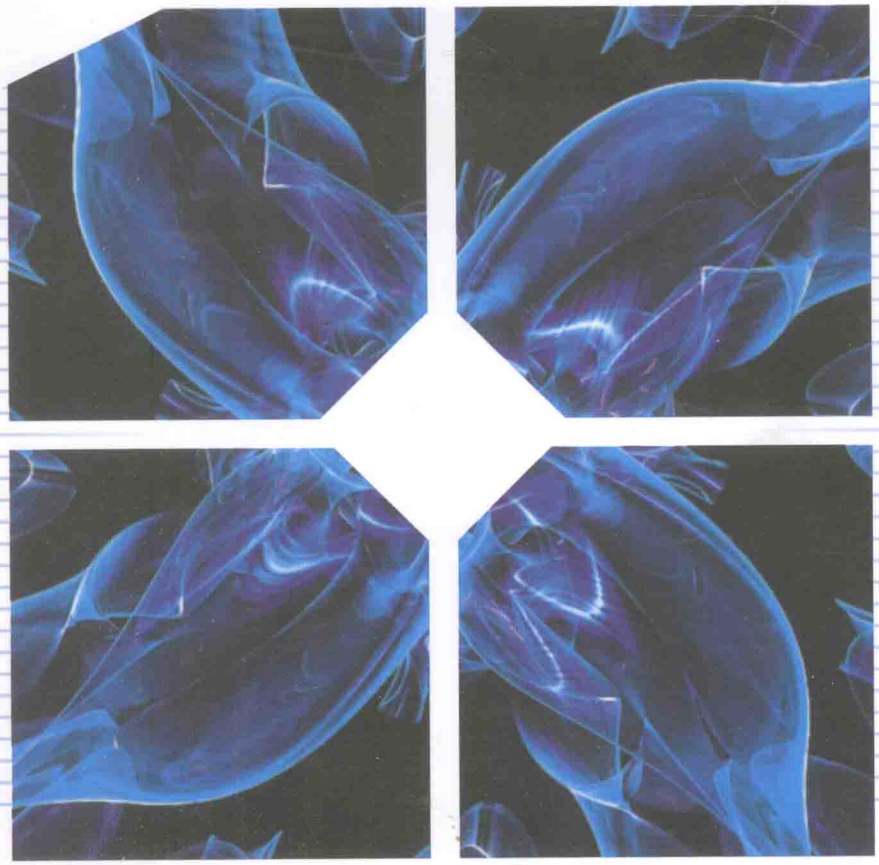
普通高等教育计算机系列规划教材

C语言程序设计

（第二版）

姜海涛　主编



科学出版社

普通高等教育计算机系列规划教材

C 语言程序设计

（第二版）

姜海涛　主编

闫　超　曹震中　武　楠　王　妍　刘红娟　叶永凯　卫　娜　副主编

科学出版社

北　京

内 容 简 介

本书根据全国计算机等级考试二级 C 语言程序设计考试大纲要求，结合目前高等院校学生学习计算机程序设计课程的情况组织内容，全面地介绍了 C 语言程序设计的基础知识，系统地讲述了 C 语言程序设计的基本方法和技巧。

本书以 ANSI C 语言标准为依据，深入浅出地介绍了 C 语言的基本数据类型、控制结构、数组、指针、结构体、文件、输入/输出等内容。在讲解语法规则的同时，结合具体实例讨论了用 C 语言解决实际问题的方法和技巧，使学生对 C 语言程序的开发过程有整体的了解，掌握基本的计算机程序设计方法，培养学生利用计算机分析问题和解决问题的能力。本书共分为 10 章，每章的后面都提供了习题，并附有参考答案。

本书可作为高等院校计算机普及教材，也可作为计算机等级考试辅导教材。

图书在版编目(CIP)数据

C 语言程序设计/姜海涛主编. —2 版. —北京：科学出版社，2014
（普通高等教育计算机系列规划教材）
ISBN 978-7-03-041624-7

Ⅰ. ①C… Ⅱ. ①姜… Ⅲ. ①C 语言-程序设计-高等学校-教材
Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 186317 号

责任编辑：李太铼 / 责任校对：柏连海
责任印制：吕春珉 / 封面设计：耕者设计工作室

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com
新科印刷有限公司 印刷
科学出版社发行　各地新华书店经销
*
2011 年 1 月第 一 版　开本：787×1092　1/16
2014 年 8 月第 二 版　印张：19 1/4
2014 年 8 月第六次印刷　字数：438 000
定价：38.00 元
（如有印装质量问题，我社负责调换<新科>）
销售部电话 010-62134988　编辑部电话 010-62135763-1012 HP03

前　言

C 语言是一种应用十分广泛的通用程序设计语言。它功能丰富，表达能力强，使用方便灵活，程序执行效率高，可移植性好。C 语言是学生学习计算机课程的必修语言，也是非计算机专业计算机等级考试（二级）最为普及的课程之一。

本书由讲授 C 语言程序设计多年、具有丰富教学经验的一线任课教师编写，能够帮助读者快速掌握用 C 语言编写程序的方法和技巧。本书力求全面介绍 C 语言的各种主要特性，同时结合具体实例指导读者进行学习。书中大部分实例可以直接运行，而不是孤立的程序段。通过这些实例，不仅演示了如何有效地使用 C 语言，同时向读者介绍了一些常用的算法、良好的程序设计风格及结构化程序设计的原则。

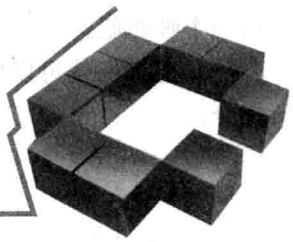
本书内容安排合理，讲解简明扼要、通俗易懂，能够使初学者轻松掌握 C 语言语法规则和程序设计方法，主要内容包括：C 语言基本数据类型、运算符和表达式、输入/输出、基本控制结构、函数、数组、指针、结构体、文件。

本书第一版于 2011 年出版，第 1、2、8 章由闫超编写，第 3、5 章由姜海涛编写，第 4 章由刘红娟、王妍编写，第 6、7 章由曹震中编写，第 9、10 章由武楠编写，全书由姜海涛统稿。针对任课老师及学生的反馈意见，编者于 2014 年上半年对本书进行了修订，修订工作主要由姜海涛、叶永凯、曹震中、卫娜完成。曹宝香教授审阅了全书，并提出了宝贵意见，在此表示衷心的感谢。

由于时间仓促且编者水平有限，书中不足之处在所难免，恳请广大读者提出宝贵意见。

目　录

第1章 概　述



本章要点

- 程序与程序设计语言；
- C 程序的结构；
- C 程序的开发环境及开发过程。

学习目标

- 了解程序设计语言的发展简史和分类；
- 掌握 C 程序的基本结构；
- 掌握 C 程序的编辑、编译、链接和执行的过程。

1.1 什么是程序

程序是为完成某一特定任务而定义的一组指令的序列。我们先来看一个游戏：游戏中有两个人，其中一个被布蒙上双眼，另一个人是你。场地中混乱地摆上许多啤酒瓶。游戏任务是由你发号施令，指挥被蒙眼者从场地一端穿越到另一端，其间不允许碰倒任何一个啤酒瓶。

这里我们就可以明白指令的概念，指令就是一组符号。我们可以用以下方式发出指令：

向左转
迈左脚
前进半步
右脚跟进
向右转
迈左脚
前进一步
右脚跟进
向右转
跳跃
……

从广义上讲，这就是为完成穿越障碍而制定的程序。不难看出，只要严格按照我们发出的指令，所有的人均可穿过障碍，到达目的地。另一方面，程序的执行必须严格按照指令发出的顺序执行，否则将不能到达目的地。

计算机执行程序的过程与这个游戏类似。我们先来看一下程序在计算机上执行的原理。计算机硬件只能按部就班地执行指令，计算机要想工作，必须通过执行程序来实现。在这里，计算机就是指令接收者，而程序就是我们向计算机发送的指令序列。计算机通过逐条执行程序中定义的计算机能够识别的指令来完成规定的任务。另外，类似于人类能够理解的指令有限，计算机能够识别的指令也是有限的（比人类能理解的要少很多）。因此，程序必须由计算机能够识别的指令组成。

1.2 程序设计语言

语言就广义而言，是一套共同采用的沟通符号、表达方式与处理规则。人类沟通所使用的语言称为自然语言。程序设计语言是程序员与计算机交流的主要工具。程序员采用某种特定的程序设计语言编写程序，计算机执行程序以完成规定的任务。目前世界上已知现存的语言有 3000 多种，而程序设计语言的种类也多种多样。从程序设计语言的发展来看，程序设计语言分为低级语言和高级语言两大类。低级语言又分为机器语言与

汇编语言。

1.2.1 机器语言

机器语言是机器指令的集合。机器指令就是计算机能够直接识别并执行的指令。计算机的机器指令是一个二进制编码。例如，应用 8086 CPU 完成计算 s=768+12288−1280 的三条机器指令如下：

```
10110000000000000000000011
00000101000000000000110000
00101101000000000000000101
```

假如将程序错写成如下形式，请找出错误。

```
10110000000000000000000011
00000101000000000000011000
00101101000000000000000101
```

不难看出，用机器语言编写程序非常困难，因为记住这些二进制编码或者找出其中的错误都是非常困难的事情。

1.2.2 汇编语言

早期的程序员们很快就发现了使用机器语言带来的麻烦，于是汇编语言产生了。汇编语言是汇编指令的集合。汇编语言与机器语言的区别在于指令的表示方法。机器语言是面向计算机的语言，由于计算机只能识别二进制形式的指令，因此机器语言均采用了二进制的形式；而汇编指令则是面向程序员的语言，它采用了类似人类所使用的自然语言的语法来表示这些指令，从而便于程序员阅读和记忆。例如，将寄存器 BX 的内容传送到寄存器 AX 的机器指令是“1000100111011000”，而对应的汇编指令则为“mov ax,bx”。很明显，后者更方便程序员的阅读和记忆。需要说明的是，计算机只能识别机器指令，因此需要将采用汇编语言编写的程序翻译成计算机能够识别的指令序列，这一工作由称为“汇编程序”的专门程序来完成。

1.2.3 高级语言

汇编指令与机器指令基本上是一一对应，它的执行同机器语言一样受底层硬件平台的限制。更重要的是，用一条条指令实现一个程序的编写过于繁琐。从最初与计算机交流的痛苦经历中，人们意识到，应该设计一种这样的语言，它接近于数学语言或人类的自然语言，同时又不依赖于计算机硬件，编出的程序能在不同体系结构的计算机上执行。

回到上一节提到的游戏，一个动作可能需要两步以上的指令才能完成。如果路线比较复杂，那么我们可能需要非常长的一个指令序列。在指令接收者能理解的前提下，我们可以把指令进行精简，上述指令序列可精简如下：

```
向左迈进半步
```

向右迈进一步
跳跃
……

需要说明的是，我们的精简指令是在指令接收者能够理解的前提下，即指令接收者能够自动地将精简后的指令转换为上述的单步指令执行。例如，当指令接收者接收到“向左迈进半步”的指令时，他仍然按照“向左转、迈左脚、前进半步，右脚跟进”的步骤一步步执行。因此，从本质上来讲，精简指令只是提高了发指令者的效率，而指令接收者（也是执行者）的执行效率并没有因此而提高。

高级语言是对汇编语言的进一步抽象，它更接近于人类使用的自然语言。例如，求两个数的最大值的 C 语言代码如下所示：

```
if (a > b)
    max = a;
else
    max = b;
```

不难看出，高级语言更接近于人类的自然语言描述。但需要注意，计算机能识别的只有机器语言，因此用高级语言编写的程序也需要经过专门的编译器程序翻译成机器指令才能在计算机上执行。

1.3 C 语言程序

C 语言是目前世界上普遍流行、使用非常广泛的高级程序设计语言之一。鉴于 C 语言对底层硬件操作方面的优势，C 语言广泛应用于操作系统（如 Windows、Linux、Unix 等操作系统）、工业控制等软件的开发；另外，C 语言具有绘图能力强、可移植性好的特点，并具备很强的数据处理能力，因此也适用于二维、三维图形动画软件（如 3D 游戏）的开发。

有效学习一门新程序设计语言的唯一途径就是使用它编写程序。对于大多数的初学者来说，编写的第一个程序几乎都是相同的，即在屏幕上输出以下内容：

```
Hello, World!
```

在 C 语言中，我们可以使用下列程序代码来实现：

```
/*程序 1-1，输出"Hello,World!"的简单 C 程序*/
#include <stdio.h>
void main()
{
    printf("Hello, World!\n"); /*调用格式化输出函数*/
}
```

在解释上述代码之前，首先进行如下说明：语言均具有一定的语法规则，程序设计

语言也不例外。与自然语言不同，程序设计语言的规定更为严格。例如，上述代码中的“#”、“()”、“{}”等均不能省略，printf语句也不能随意换行，否则程序将无法执行。

需要特别注意的是，C语言程序严格区分代码的大小写形式。例如，在上述程序代码中，main不能写成MAIN、Main等形式。

在详细介绍程序的组成要素之前，下面再给出一个比程序1-1略复杂的程序。程序1-2实现了计算两个整数12与23和的功能，程序执行后，在运行窗口中显示：

```
sum=35
```

以下是完整的程序代码：

```
/*程序 1-2，计算两个整数之和的简单程序*/
#include <stdio.h>
/*下面是 main 函数的定义*/
void main()
{     int a,b,sum;
      a=12;
      b=23;
      sum=add(a,b); /*main 函数调用 add 函数,由 add 函数计算整数的和*/
      printf("sum= %d\n",sum);
}
/*add 函数实现求两个整数之和的功能*/
int add(int x,int y)
{   int  z;
    z=x+y;
    return(z);
}
```

1.3.1 注释

程序中，“/*”和“*/”之间包含的内容属于注释，“/*”表示注释的开始，“*/”表示注释的结束。注释可以单独占一行，也可以和程序中的其他代码放在一行，并且注释可以占多行。注释一般分为序言性注释和功能性注释。序言性注释通常放在程序的开始处，用于说明程序的名称、功能、设计思想、版本、设计者等信息；功能性注释通常放在程序代码内部，用于说明关键数据、语句、控制结构的含义和作用。

为程序适当增加一些注释是一种良好的程序设计习惯。注释可以提高程序的可读性，同时便于程序的维护。

注释不影响程序的执行，注释只存在于源程序中。源程序在编译时，编译器会忽略

注释，生成的目标程序中不包含这些注释。

1.3.2 关键字

上述程序中的 include、void 是 C 语言的关键字，关键字是被 C 语言本身所使用的、具有特殊含义和功能的词汇，不能被用作其他用途。在后面的章节中，我们会逐渐接触到 C 语言的其他关键字。

注意：C 语言中的关键字全部使用小写形式。

1.3.3 预处理命令

程序代码中的“#include <stdio.h>”是一个预处理命令。预处理命令均以“#”符号开始，并且每个预处理命令要独占一行。include 表示命令名，称为“文件包含命令”。“#include <stdio.h>”用于告诉编译器本程序要将一个叫做“stdio.h”的文件内容包含进来。“stdio.h”（stdio 即为 standard input output 的缩写）是 C 语言标准函数库中定义的一个头文件，由于 C 语言中的输入/输出操作均由已在标准函数库中定义的输入/输出函数来实现，而在 stdio.h 文件中包含了这些输入/输出函数的说明信息。因此，在包含了该头文件的内容之后，我们便可在程序中直接使用这些输入/输出函数。比如，我们在写文章时，引用了其他文章的内容，就需要在参考文献里进行说明，“#include <stdio.h>”就类似于参考文献里的说明。因为在程序中我们使用了 stdio.h 文件内说明的输出函数 printf，所以需要在程序的开头进行说明。

注意：与参考文献不同的是，参考文献列表通常放在文章的最后，而#include 预处理命令通常放在程序的开头。

程序中经常要使用 C 语言标准函数库中提供的输入/输出函数来完成数据的输入/输出，因此，一般都带有这样一个预处理命令。

1.3.4 函数

程序 1-1 中的其他代码给出了 main 函数的定义，main 是函数名，可称为主函数。程序 1-2 中不但包含一个 main 函数，还定义了一个名为 add 的函数。

函数（function）是用来构建 C 语言程序的模块，是 C 语言程序的基本组成单位。通过使用函数可以降低程序开发的难度，并让程序具有良好的结构。

函数的概念来自于数学。在数学中，假定函数 f 和函数 g 的定义如下：

$$f(x) = x^3$$
$$g(x,y)=f(x)+3y+1$$

其中，f、g 称为函数名，x、y 称为函数的自变量（在程序设计中称为函数的参数）。f(x)、g(x,y)的定义给出了通过自变量计算函数值的方法。另外可以看出，函数 g(x,y)中调用了

函数 f(x)，即在进行 g(x,y)的计算时，x^3 的计算交由 f(x)完成。C 语言中的函数与数学中的函数有相似之处，也包括函数名、参数以及具体操作的定义。

程序 1-1 中 main 函数的定义可以分为两部分，函数首部和函数体。

```
void  main()    ←————————— 函数首部

{
   printf("Hello, World!\n");  ←— 函数体
}
```

函数首部依次给出函数类型、函数名称和函数参数定义，参数定义放在函数名后的一对小括号中。函数体放在一对大括号中，其中可以包含一系列的语句，这些语句给出了函数执行的操作。

和数学函数不同的是，C 语言的函数可以有确定的计算结果，也可以没有。对于没有明确计算结果的函数应将其类型指定为 void。另外，C 语言函数可以有参数，也可以没有参数。对于没有参数的函数，其参数定义可以为空白，但函数名后的一对小括号不能省略。

程序 1-2 中的 add 函数是一个既有参数，也有返回值的函数。参数 x、y 表示 add 函数需要从外部得到两个整数（这两个整数由 mian 函数在调用 add 函数时提供）。add 函数的处理结果就是两个整数之和，因此它的类型指定为 int（int 表示一种整型）。

main 函数是 C 语言程序中的一个特殊函数，每个程序必须而且只能包含一个 main 函数，它代表程序运行时的入口。程序运行时，首先找到 main 函数，然后依次执行 main 函数中包含的每条语句，直到 main 函数的结束。

每个函数（包括用户自定义函数和系统定义函数）都用于实现某一特定的功能，并且可以相互调用。调用函数时，只需要使用函数名加上小括号括起来的参数即可。

main 函数可以调用其他函数，从而将一部分工作交给其他函数完成，被调用的函数执行完成后将返回 main 函数，main 函数继续执行直到程序结束。例如，在程序 1-1 中 main 函数调用 printf 函数完成一行文本的输出。其中 printf 为被调用函数名，小括号中的“Hello, World!\n”为向该函数提供的参数，即输出的内容。

我们在程序中使用的函数可以分为两类，一类是我们为了实现某个功能自己编写的函数，通常称为“自定义函数”，程序 1-2 中的 add 函数就属于自定义函数；另一类是由我们使用的编译器提供的函数库中的函数，通常称为“库函数”，程序 1-1 中所使用的 printf 函数就是一个库函数。

1.3.5 语句

C 语言中的语句是程序执行时向计算机发出的指令，语句给出了计算机要执行的操作。预处理命令、变量定义等内容不算作语句。语句出现在函数体内，一个函数的执行过程就是依次执行函数体内语句的过程，这些语句实现了函数的功能。

在程序 1-1 中，main 函数体内只包含一个语句：