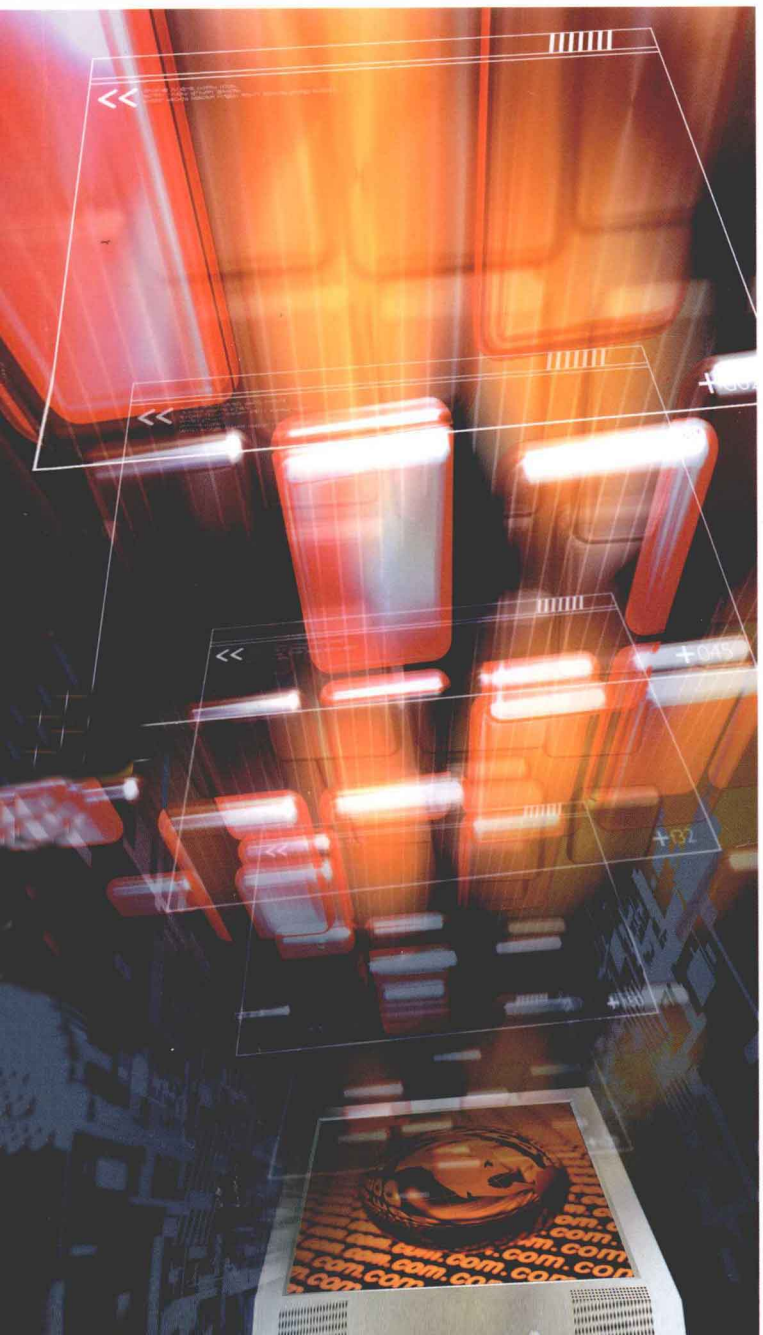
21世纪高等院校计算机应用规划教材

C语言程序设计教程

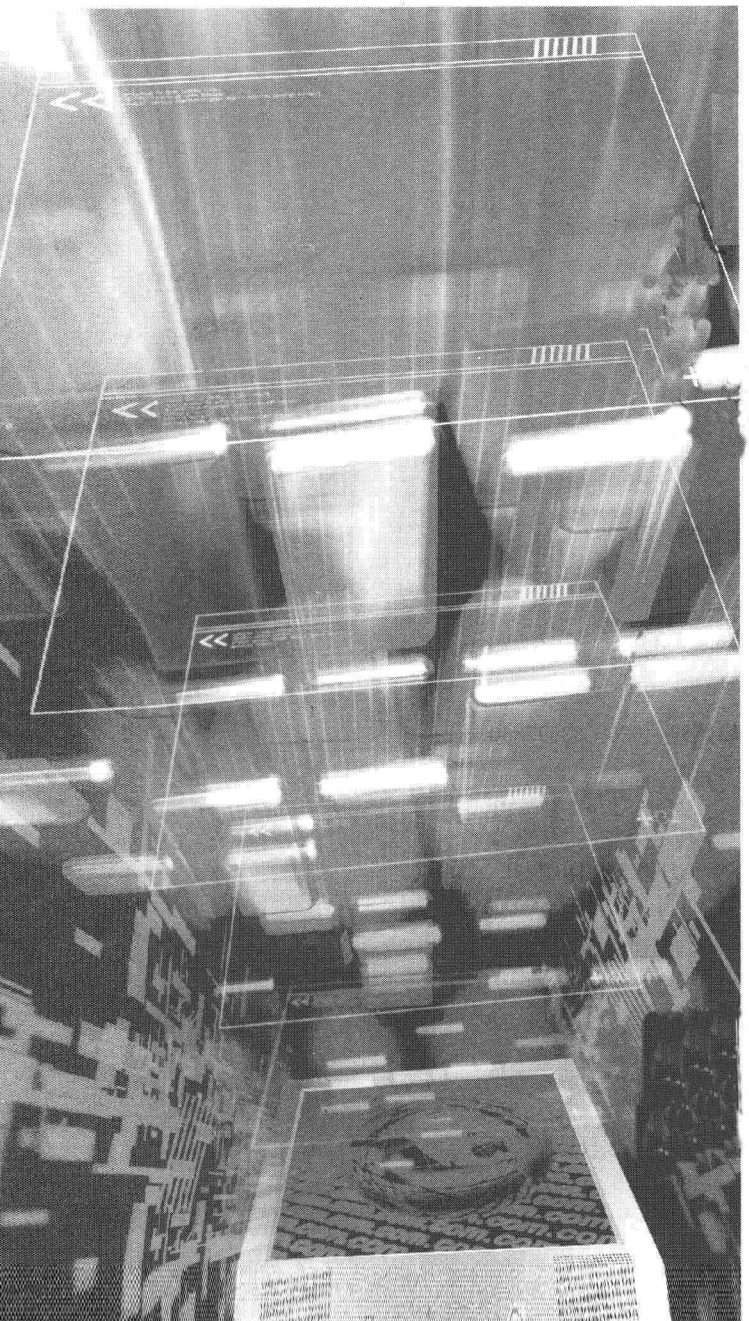


吕俊 谢旻 张军强 编著

南京大学出版社

21世纪高等院校计算机

C语言程序设计教程



吕 俊 谢 旻 张军强 编著

南京大学出版社

图书在版编目(CIP)数据

C语言程序设计教程 / 吕俊，谢旻，张军强编著. -- 南京 : 南京大学出版社，2014.1
21世纪高等院校计算机应用规划教材
ISBN 978-7-305-12869-1

Ⅰ. ①C… Ⅱ. ①吕… ②谢… ③张… 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011646号

出版发行　南京大学出版社
社　　址　南京市汉口路22号　　邮　　编　210093
网　　址　http://www.NjupCo.com
出 版 人　左　健

丛 书 名　21世纪高等院校计算机应用规划教材
书　　名　C语言程序设计教程
编　　著　吕　俊　谢　旻　张军强
责任编辑　王秉华　单　宁　　编辑热线　025-83595860

照　　排　南京南琳图文制作有限公司
印　　刷　南京人文印务有限公司
开　　本　787×1092　1/16　印张25　字数618千
版　　次　2014年1月第1版　　2014年1月第1次印刷
ISBN　978-7-305-12869-1
定　　价　49.00元

发行热线　025-83594756　83686452
电子邮箱　Press@NjupCo.com
　　Sales@NjupCo.com(市场部)

前　言

C语言是得到广泛应用的程序设计语言之一，它功能丰富、使用灵活方便，既具有高级语言的优点，又具有直接操纵计算机硬件的能力，是许多计算机专业人员和计算机爱好者学习程序设计语言的首选。

传统的C语言教材侧重于介绍语法规则和程序结构，学生即使掌握了所有知识点，但在解决实际问题时仍会茫然不知所措。本书从计算思维培养的角度出发，以应用为背景，通过对实际案例的思考分析，借助任务驱动的模式将知识点串接起来，形成逻辑清晰的脉络和主线，加深读者对C语言的理解和驾驭能力，提升分析问题和解决问题的能力。

作为程序设计语言的入门教材，编者在编写过程中力求从读者的角度出发，由浅入深地安排内容、简洁而准确地阐述ANSI C的基本概念，并配以详实的图表。通过实例引导读者思考解决问题的方法和步骤，着力培养程序设计能力和语言的应用能力。文中所有例题均在Visual C++6.0环境下运行通过。全书共十四章，分为三个部分，第1～5章侧重C语言基本语法知识和程序结构的介绍，包括数据表达中的基本数据类型，数据处理中的表达式，以及流程控制中的顺序、分支、循环三种语句及控制方式；第6～13章以语言应用能力提升为目的，介绍了函数、构造数据类型以及指针和文件的使用；第14章以实际应用为背景，通过一个实例系统阐述用结构化程序设计思想思考解决复杂问题的过程，并使用C语言来实现系统功能。在每章内容后，还安排了适量的习题，这些习题从易到难地帮助读者在理解、掌握基本概念和语法规则的基础上，一步一步提高编程能力。

本书的编者都是长期工作在计算机程序设计语言教学一线的教师，有着丰富的教学经验，对程序设计语言初学者的学习情况比较熟悉。因此，编者将教学过程中的体会、经验和教训融会到教材的编写中，力求为读者呈现一本实用

而易于理解的教材。本书可作为全国或省级计算机等级考试(二级C语言)的参考用书,也可作为各类院校相关专业的C语言程序设计的教材。

全书由吕俊、谢敏、张军强主编,其中第1、8、10、12～14章由吕俊编写,第2～5章由谢旻编写,第6、7、9、11章由张军强编写。同时衷心的感谢在编写过程中曾经帮助过我们的同志。

由于编者的水平有限,加之编写时间仓促,错误和疏漏在所难免,敬请读者批评指正。

编　者

2013年12月

目　录

第一章　绪　论

1.1　程序与算法

1.1.1　程序与程序设计语言

迄今为止，我们所使用的计算机大多是按照匈牙利数学家冯·诺依曼提出的“存储程序控制”的原理进行工作的，即一个问题的解决步骤（程序）连同它处理的数据都使用二进制表示，并预先存放在存储器中。程序运行时，CPU 从内存中一条一条地取出指令和相应的数据，按指令操作码的规定，对数据进行运算处理，直到程序执行完毕为止。

程序（Program）是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合，是为实现预期目的而进行操作的一系列语句和指令。由于计算机不能理解人类的自然语言，所以必须要用一种特殊的语言来和计算机进行交流。用于编写计算机可执行程序的语言称为程序设计语言，程序设计语言按其发展的先后可分为机器语言、汇编语言和高级语言。

1. 机器语言

机器语言是伴随着第一台计算机的诞生而出现的，在形式上它是由“0”和“1”构成的一串二进制代码，每台计算机都有自己的一套机器指令。机器指令的集合就是机器语言，例如下面这段代码就是实现两数相加的机器语言代码的片段：

```
1011111000100110011
```

机器语言由于直接用二进制表示，是计算机硬件系统真正理解和执行的唯一语言。因此它的效率最高，执行速度最快且无需翻译。但它与人们所习惯的自然语言、数学语言等差别很大，难学、难记、难读，难以用来开发实用的计算机程序。

2. 汇编语言

汇编语言将机器指令映射为一些可以被人们读懂的助记符，如 ADD、SUB 等，同时又用变量取代各类地址，这样就构成了计算机符号语言。用汇编语言编写的源程序必须经过翻译（即汇编）变成机器语言后，才能被计算机识别和执行，所以其执行速度要慢于机器语言编写的程序。

汇编语言用指令代替了相应的机器代码，例如前面提到将两数相加的那段机器代码，用汇编指令描述为：

```
add ax, bx
```

汇编语言在一定程度上克服了机器语言难以辨认和记忆的缺点，但对大多数用户来说，

仍然是不方便理解和使用的。

3. 高级语言

为了克服机器语言、汇编语言的缺点，人们又设计了高级程序设计语言。在语言表示和语义描述上，高级语言更接近人类的自然语言(英语)和数学语言，具有学习容易、使用方便、通用性强、移植性好等特点。如前面提到的那段机器代码，用高级语言可以直接描述为：

C=A+B

早期应用比较广泛的几种高级语言有 FORTRAN、BASIC、PASCAL 和 C 等，在此之后，又诞生了上百种高级程序设计语言，并根据应用领域的不同和语言本身侧重点差异，分成了许多类别。但高级语言的本质都是相通的，在学会了一门经典语言之后，就能很容易地掌握其他高级语言。

高级语言程序(称为源程序)虽然编写方便，但计算机不能直接执行，必须经过一定的软件(例如编译和连接程序)对其进行加工，生成由机器指令表示的程序(称为目标程序)，然后才能由计算机来执行。这种加工过程可以分为编译和解释两种方式。

编译方式是使用编译程序将高级语言源程序整个翻译成目标程序，然后通过连接程序将目标程序连接成可执行程序的方式。C 语言源程序采用的就是编译方式来执行的，编译过程如图 1-1 所示。

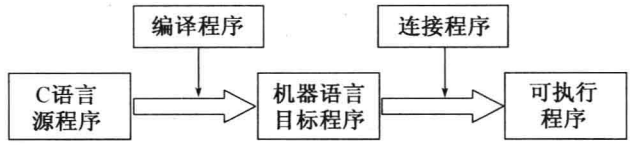


图 1-1 C 语言编译过程

解释方式是将源程序逐句翻译、逐句执行的方式，解释过程不产生目标程序，基本上是翻译一行执行一行，边翻译边执行。如果在解释过程中发现错误就给出错误信息，并停止解释和执行，如果没有错误就解释执行到最后的语句。

无论是编译程序还是解释程序，都起着将高级语言编写的源程序翻译成计算机可以识别与执行的机器指令的作用。但这两种方式是有区别的，区别在于：编译方式将源程序经编译、连接得到可执行程序文件后，就可脱离源程序和编译程序而单独执行，所以编译方式的效率高，执行速度快；而解释方式在执行时，源程序和解释程序必须同时参与才能运行，由于不产生目标文件和可执行程序文件，解释方式的效率相对较低，执行速度较慢。

1.1.2 算法

算法是指为解决某一特定问题而采取的有限步骤，它是一组有穷序列或是一组有穷动作序列。例如：要计算 1+2+3+4+5 的值，一般是先将 1 和 2 相加得 3，再将 3 加 3 得 6，再将 6 加 4 得 10，最后再加 5 得 15。无论手算、心算、或用算盘、计算器计算，都要经过有限的、事先设计好的步骤，解决这一问题而采取的方法步骤就称为算法。算法是解题方法的精确描述，解决一个问题的过程就是实现一个算法的过程。

例如，给定两个正整数 p 和 q，如何求出它们的最大公约数，古希腊数学家欧几里得(Euclid)给出了一个著名的算法——辗转相除法：

(1) 如果 p < q,交换 p 和 q;

(2) 求 p/q 的余数 r;

(3) 如果 r=0,则 q 就是所求的结果,否则反复做以下工作:

将 q 的值赋给 p,r 的值赋给 q,重新计算 p/q 的余数,直到 r=0 为止,q 的值即为原来两个正整数的最大公约数。

计算机算法一般分为两大类:数值运算和非数值运算。如求若干数之和、求方程的根、求一个函数的定积分等都属于数值运算。而将若干人名按字母顺序排序、图书情报资料检索、计算机绘图等则属于非数值计算。目前,数值运算的算法比较成熟,对各类数值计算问题大部分都有成熟的算法可供选用。

1. 算法的表示

算法表示的常见形式有:自然语言、流程图、N-S 图、伪代码和计算机语言。用计算机语言来表示算法的过程就是程序设计。

(1) 自然语言

自然语言就是人们日常使用的语言。用自然语言描述的算法直观、通俗易懂,为人们所熟悉,但很难"系统"并"精确"地表达算法。因此,自然语言描述的算法要变成在计算机上执行的程序还要做大量的工作。此外,用自然语言描述包含分支和循环结构的算法很不方便。

(2) 流程图

流程图是用各类图形(如矩形框、菱形框)、流程线及文字说明描述计算过程的框图。流程图是描述算法的常用工具,其优点是直观形象、易于理解,能清楚地表示程序设计思路。流程图表示的算法不依赖于任何具体的计算机和计算机语言,有利于程序设计工作。一些常用的流程图符号如图 1-2 所示:

图 1-2 常用的流程图符号

(3) N—S 结构图

N—S 结构图又称为盒图,去掉了流程线,算法写在一个称之为元素框的矩形框里。元素框有 3 种形式:顺序框、选择(分支)框和循环框,如图 1-3 所示。

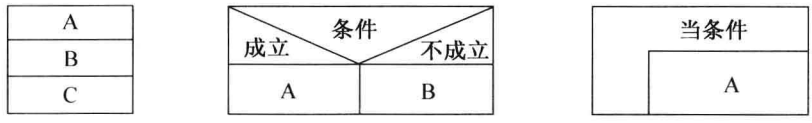


图 1-3 N—S 结构图元素框

(4) 伪代码

伪代码是介于自然语言和计算机语言之间的、用文字和符号描述算法的一种语言形式,是描述算法的常用工具。用伪代码表示算法没有严格的语法规则限制,重在把意思表达清楚,它书写方便、格式紧凑,也比较好懂,便于过渡到计算机语言表示的算法(即程序)。

(5) 计算机语言表示算法

由于计算机是无法识别用自然语言、流程图和伪代码等形式表示的算法，只有用计算机语言编写的程序才能被计算机执行(当然要经过编译、连接形成目标程序后才能被计算机识别和执行)。因此在用流程图或伪代码表示出一个算法后，还要将该算法转换成计算机语言程序。用计算机语言表示算法必须严格遵循所用计算机语言的语法规则。

【例 1.1】 有两个杯子 A 和 B，分别盛放酒和醋，要求将它们互换(即 A 杯原来盛放酒，现在改盛醋，B 杯则相反)。

【问题分析】

根据常识，必须增加一个空杯 C 作为过渡，其操作步骤(用自然语言表示的算法)如下：

步骤 1：将 A 杯中的酒倒在 C 杯中。

步骤 2：将 B 杯中的醋倒在 A 杯中。

步骤 3：将 C 杯中的酒倒在 B 杯中。

这就是以后要用到的交换两个变量值的算法，相应的伪代码表示的算法如下：

```
input A and B
C←A
A←B
B←C
print A and B
```

用 C 语言程序可以表示为：

【程序代码】

```
#include <stdio.h>                          /*包含头文件 stdio.h 到本程序中*/
int main(void)                              /*主函数*/
{
    int a, b, c;                            /*定义三个变量,等价于问题中的三个杯子*/
    printf("Please input a and b:\n");      /*在屏幕上提示用户输入变量 a 和 b 的值*/
    scanf("%d%d", &a, &b);                  /*用户输入变量 a 和 b 值*/
    c = a;                                  /*将变量 a 的值存放在中间变量 c 中*/
    a = b;                                  /*将变量 b 的值存放在变量 a 中*/
    b = c;                                  /*将中间变量 c 的值存放在变量 b 中*/
    printf("a=%d  b=%d",a,b);               /*输出交换后的变量 a 和变量 b 的值*/
    return 0;
}
```

【运行结果】

```
Please input a and b:
3  5  ↙                                     /*键盘输入*/
a = 5  b = 3
```

2. 常用算法简介

(1) 列举法

列举法的基本思想是根据提出的问题，列举所有可能的情况，然后根据问题给定的条件

判断哪些是成立的，哪些是不成立的。列举法常用于解决“是否存在”或“有多少种可能”等类型的问题，例如求解不定方程。

列举法的特点是算法比较简单，但当列举的可能情况较多时，列举算法的工作量将会很大。因此，在用列举法设计算法时，应该重点注意方案尽可能优化，运算量尽可能减少。通常在设计列举算法时，要对实际问题进行详细分析，将与问题有关的知识条理化、完善化、系统化，从中找出规律；或对所有可能的情况进行分类，引出一些有用信息，这样就可以大大减少列举量。

(2) 归纳法

归纳法的基本思想是，通过少量的特例，经过分析，最后找出一般关系。显然，归纳法要比列举法更能反映问题的本质，并且可以解决列举量为无限的问题。但是，从一个实际问题中归纳总结出一般关系，并不是一件容易的事情，尤其是要归纳出一个数字模型更为困难。从本质上讲，归纳就是通过观察一些简单而特殊的情况，最后总结出一般性的结论。

归纳是一种抽象，即从特殊现象中找出一般关系。但由于在归纳的过程中不可能对所有的情况进行列举，因此，最后由归纳得到的结论还只是一种猜测，还需要对这种猜测加以必要的证明。

(3) 递推算法

所谓递推，就是指从已知的初始条件出发，逐次推出所要求的各个中间结果和最后结果。其中初始条件，或是问题本身已经给定，或是通过对问题的分析与化简而确定。递推本质上也属于归纳法，工程上许多递推关系式实际上是通过对实际问题的分析与归纳而得到的，因此，递推关系式往往是归纳的结果。

(4) 递归算法

人们在解决一些复杂问题时，为了降低问题的复杂程度(如问题的规模等)，一般总是将问题逐层分解，最后归结为一些最简单的问题。这种将问题逐层分解的过程，实际上并没有对问题进行求解，但当解决了最后那些简单的问题，再沿着原来分解的逆过程进行综合后，便得到了问题的解，这就是递归的基本思想。由此可以看出，递归的基础也是归纳。在工程实际中，有许多问题就是用递归来定义的，数学中的许多函数也是用递归来定义的。递归在可计算性理论和算法设计中占有很重要的地位。

1.2 C 语言简介

1.2.1 C 语言的发展

C 语言是在 B 语言的基础上发展而来的。1969 年，美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 为 DEC PDP—7 计算机设计了一个操作系统软件，即最早的 UNIX 系统。此后 Ken Thompson 又根据剑桥大学的 Martin Richards 设计的 BCPL(Basic Combined Programming Language)语言为 UNIX 设计了一种便于编写系统软件的语言，命名为 B 语言。B 语言是一种无类型的语言，过于简单，在描述客观世界事物时会遇到许多困难，因此并没有流行起来。

1972—1973 年间，贝尔实验室的 Dennis Ritchie 在 B 语言的基础上设计出了 C 语言。

C语言既保持了B语言的优点(精练、接近硬件),又克服了它的缺点(过于简单、数据无类型等)。最初的C语言是为描述和实现UNIX操作系统提供一种工作语言而设计的。1973年,Ken Thompson和Dennis Ritchie两人合作把UNIX操作系统的90%以上程序用C语言改写,增加了多道程序设计能力,同时大大提高了UNIX操作系统的可移植性和可读性。后来,C语言又做了多次改进,渐渐形成了不依赖于具体机器的C语言编译程序,于是C移植到其他机器时所需做的工作大大简化了,成为如今广泛应用的计算机语言之一。

随着C语言使用得越来越广泛,C语言的编译程序也有不同的版本。一般来说,1978年B. W. Kernighan和Dennis Ritchie合著的*The C Programming Language*是各种C语言版本的基础,称之为旧标准C语言或"K&R C"。1989年,美国国家标准局(ANSI)颁布了第一个官方的C语言标准,称为"ANSI C"或"C89"。1990年,这个标准被国际标准化组织(ISO)采纳,将其命名为"ISO C"或"C90"。目前使用的如Microsoft C、Turbo C等版本都把ANSI C作为一个子集,并在此基础上做了合乎它们各自特点的扩充。

1.2.2 简单的C语言程序实例

下面通过两个简单的实例,说明C源程序的基本组成。

【例1.2】 输入圆的半径,求面积。

【程序代码】

```
#include<stdio.h>                    /*包含头文件stdio.h到本程序中*/
#define PI 3.14159                   /*定义符号常量PI*/
int main(void)                       /*主函数*/
{
    double r, s;                     /*定义r和s为双精度实型的变量*/
    printf("Please input radius:\n"); /*在屏幕上输出提示信息*/
    scanf("%lf", &r);                /*从键盘输入圆的半径,保存到变量r中*/
    s = PI * r * r;                  /*根据输入的半径计算圆的面积,保存到变
                                       量s中*/
    printf("The area is %f\n", s);   /*在屏幕上输出圆的面积,即变量s的值*/
    return 0;                        /*结束主函数的执行,返回0值到系统*/
}
```

【运行结果】

```
Please input radius:
3        ↙
The area is 28.274310
```

说明:

(1) 程序中用"/*"和"*/"括起来的内容称为注释。它的作用是对程序进行说明,提高程序的可读性。在编译时,注释将被忽略。

(2) 一般源程序开始处的"include"是文件包含命令,以"#"开头,其作用是指示编译预处理程序将指定的文件嵌入到该源程序文件中。

(3) "#define PI 3.1415926"是宏定义命令,以"#"开头,定义PI为符号常量,代表

3.1415926。在程序设计时，凡是需要书写 3.1415926 的地方都可以用 PI 代替。因为宏定义命令不属于 C 语言语句范畴，所以在末尾一般不加分号。

(4) main( )函数称为主函数。C 程序必须包括一个且只能包括一个 main( )函数，程序的运行从 main( )函数开始，当 main( )函数执行结束时，整个程序也就结束了。main( )函数的一般形式为：

```
int main(void)          /* 函数首部，int 表示 main()函数返回的值为整型 */
{                       /* 函数体，包含若干条语句，完成特定的功能 */
    语句组;
}
```

(5) scanf( )为输入函数，可以从键盘输入数据到指定的变量，printf( )为输出函数，可以将运算结果显示在屏幕上，程序的一般结构可以表示为如图 1-4 所示的流程。

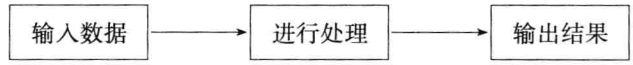


图 1-4 程序的一般结构

【例 1.3】 求两个数之和。

【程序代码】

```
#include<stdio.h>
int add(int x, int y)                   /* 定义 add()函数，求两个数之和 */
{
    int z;                              /* 定义 z 为整型变量 */
    z = x + y;                          /* 将 x+y 的值赋给变量 z */
    return z;                           /* 结束 add()函数的执行，并将 z 值返回给
                                           主函数 */
}
int main(void)                          /* 定义主函数 */
{
    int a,b,c;                          /* 定义 a,b,c 三个整形变量 */
    printf("Please input a and b:\n");  /* 在屏幕上输出提示信息 */
    scanf("%d%d", &a, &b);              /* 从键盘输入两个整数值保存在变量 a 和
                                           b 中 */
    c = add(a, b);                      /* 调用 add 函数，计算 a+b，并把结果赋给
                                           c */
    printf("The sum is %d\n", c);       /* 在屏幕上输出两数之和 c 的值 */
    return 0;                           /* 结束主函数的执行，返回 0 值到系统 */
}
```

【运行结果】

```
Please input a and b:
4  6  ↙
```

The sum is 10

说明：

(1) 该程序由 main( )和 add( )两个函数组成，虽然 add( )函数写在 main( )函数的前面，但程序仍然是从 main( )函数开始执行的；

(2) 语句“c = add(a, b);”的作用是调用 add( )函数计算 a + b 的值，并将结果赋给变量 c。add( )函数是一个用户自定义函数，其功能是求两个数的和。程序从 main( )函数开始执行，执行到“c = add(a, b);”语句时调用 add( )函数，依次执行 add( )函数的各条语句直至 add( )函数结束，再回到 main( )函数继续执行后面的语句直至 main( )函数结束。执行过程如图 1-5 所示。

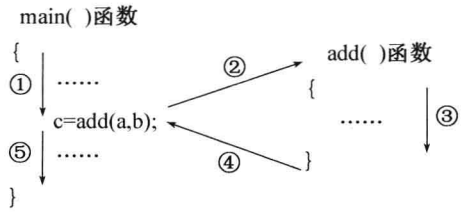


图 1-5 C语言函数执行过程

从上面两个实例，可以看到 C 语言源程序的组成及书写规则为：

(1) C 程序是由一个或多个函数组成的，其中必须要有一个且只能有一个 main( )函数。无论这个函数的位置在哪里，程序总是从它开始执行。main( )函数可以调用其他函数，但是其他函数不能调用 main( )函数。

(2) 在一个函数内，语句的执行顺序是从上到下的。

(3) C 语言程序书写形式自由，一行可以写多条语句，每条语句以分号结束(为了程序格式的清晰，最好一行只写一条语句)。程序中的所有标点符号都是英文符号。

(4) C 语言严格区分大小写，即大写字母“A”和小写字母“a”被认为是不同的符号。

(5) 注释是对程序的注解，注释的文本必须包含在“/ * ”、“ * /”之间，如“/ * 求两数之和 * /”。添加注释不影响程序的编译和运行，但可以提高程序的可读性，编程者应养成给程序注释的好习惯。

1.2.3 C 程序的开发过程

C 语言的源程序必须经过编辑、编译、连接生成可执行文件后方可运行。使用 C 语言开发一个应用程序大致要经过以下步骤：

(1) 首先要根据实际问题确定解题思路，包括选用适当的数学模型。

(2) 根据上一步思路或数学模型编写程序。

(3) 编辑源程序。输入的源程序一般以文件形式存放。

(4) 编译和连接。C 语言源程序很接近人类的自然语言，因此需要将源程序转换为计算机可直接执行的指令。这项工作又可以分为编译和连接两个步骤，编译阶段将源程序转换成目标程序，连接阶段将目标程序连接成可执行程序。

(5) 调试与测试。一个程序经过编译、连接产生了可执行程序后，就可以开始调试和测