高等学校计算机基础教育教材精选

C 程序设计教程与实训

（第 2 版）

高敬阳　李芳　主编
马静　李国捷　尤枫　吴蕾　编著
朱群雄　主审

清 华 大 学 出 版 社
北　京

内 容 简 介

本书通过案例教学的方式，由浅入深，让学生在模仿—训练—应用的过程中，快速掌握程序设计的基本思想和基本方法。

本书共9章，主要内容包括C程序概述、用C语言编写简单程序、分支结构、循环结构、数组、函数、指针、结构体与共用体、文件。各章均给出了内容丰富又有代表性的例题，全部程序都在Visual C++ 6.0中调试通过，同时也对Visual C++环境进行了介绍。书后配有各章习题分析及部分习题答案，供读者参考。此外，还提供了教学资源丰富的课程网站作为教学活动的课外补充。

本书可作为高等学校各专业C程序设计课程的教材，也可以作为各类计算机培训班的教材和成人教育同类课程教材及自学教材。

图书在版编目(CIP)数据

C程序设计教程与实训/高敬阳，李芳主编；马静等编著. —2版. —北京：清华大学出版社，2010.3
(高等学校计算机基础教育教材精选)
ISBN 978-7-302-22204-0

Ⅰ. ①C… Ⅱ. ①高… ②李… ③马… 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(2010)第031449号

责任编辑：袁勤勇
责任校对：白　蕾
责任印制：

出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编：**100084
社　　总　　机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫海金澳胶印有限公司
装 订 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185×260　　**印　　张：**14.25　　**字　　数：**325千字
版　　次：2010年3月第2版　　**印　　次：**2010年3月第1次印刷
印　　数：1～0000
定　　价：00.00元

出版说明

高等学校计算机基础教育教材精选

在教育部关于高等学校计算机基础教育三层次方案的指导下，我国高等学校的计算机基础教育事业蓬勃发展。经过多年的教学改革与实践，全国很多学校在计算机基础教育这一领域中积累了大量宝贵的经验，取得了许多可喜的成果。

随着科教兴国战略的实施以及社会信息化进程的加快，目前我国的高等教育事业正面临着新的发展机遇，但同时也必须面对新的挑战。这些都对高等学校的计算机基础教育提出了更高的要求。为了适应教学改革的需要，进一步推动我国高等学校计算机基础教育事业的发展，我们在全国各高等学校精心挖掘和遴选了一批经过教学实践检验的优秀的教学成果，编辑出版了这套教材。教材的选题范围涵盖了计算机基础教育的三个层次，包括面向各高校开设的计算机必修课、选修课以及与各类专业相结合的计算机课程。

为了保证出版质量，同时更好地适应教学需求，本套教材将采取开放的体系和滚动出版的方式（即成熟一本、出版一本，并保持不断更新），坚持宁缺毋滥的原则，力求反映我国高等学校计算机基础教育的最新成果，使本套丛书无论在技术质量上还是文字质量上均成为真正的“精选”。

清华大学出版社一直致力于计算机教育用书的出版工作，在计算机基础教育领域出版了许多优秀的教材。本套教材的出版将进一步丰富和扩大我社在这一领域的选题范围、层次和深度，以适应高校计算机基础教育课程层次化、多样化的趋势，从而更好地满足各学校由于条件、师资和生源水平、专业领域等的差异而产生的不同需求。我们热切期望全国广大教师能够积极参与到本套丛书的编写工作中来，把自己的教学成果与全国的同行们分享；同时也欢迎广大读者对本套教材提出宝贵意见，以便我们改进工作，为读者提供更好的服务。

我们的电子邮件地址是 jiaoh@tup.tsinghua.edu.cn。联系人：焦虹。

清华大学出版社

前言

程序设计能力是计算机基础教育的重要组成部分,是高等学校学生应具备的基本技能之一。程序设计知识的学习有助于使学生真正理解计算机工作原理,了解计算机解决问题的方法,有效训练学生的逻辑思维和抽象思维,同时开阔学生的视野,培养丰富的想象力和创造力,最终帮助学生更好地使用计算机解决本专业科研、工作和生活中的相关问题。

程序设计是既有挑战性,又颇有成就感的过程。有经验的程序员在重新审视 C 语言的学习时,常常会感觉这门课程其实很简单。然而,在实际面对初学者的教学过程中,却面临着比想象中多得多的困难。通常的问题是,开课之初学生有很大的热情,但随着学习的深入,到了循环、数组部分,有些学生仍然迟迟不能入门,慢慢地失去了学习的兴趣,造成恶性循环,最终甚至放弃了该课程的学习。学生普遍反映对于抽象的 C 程序设计课程难于找到入门的捷径。这些情况的出现,原因是多方面的。其中很重要的原因就是长期以来,程序设计课程过多强调语言本身及其表达细节,忽视了程序设计的本质,造成很多学生过多地陷入具体细节的旋涡里,无法站在一定的高度欣赏程序设计的美。同时 C 程序设计又是一门实践性很强的课程,学生必须通过较多的编程训练才能掌握。因此,如果能让学生一开始就很清楚自己要做的事情,循序渐进地领会程序设计的精妙,在实践中形成良好的程序设计风格,并自始至终兴趣浓厚,相信 C 语言的教学工作将会收到事半功倍的效果。

鉴于此,我们决定从教材入手,转换思路。在教材的编写过程中,本着从始至终简化语法,培养学生动手编程能力的初衷,力争独辟蹊径,写出特色,让学生了解 C 程序的编写其实远没有传说中的那样困难。

本教材全书共分为 9 章,涵盖了 C 程序设计教程应包含的基本内容。并将文件的基本使用方法提前至数组一章,让学生提前了解文件的应用,并在后续知识的学习中反复使用,加深理解。

同时,每章均由引例开始,引出该章将要引入的新知识,采用“提出问题—分析问题—引入新知识—解决问题—模仿编程—总结提高”这样一个循序渐进、螺旋式上升的教学模式。将一个个典型的、针对性强的、贴近现实或贴近专业的案例程序设计作为贯穿始终的主线,将课程内容抽茧拔丝般解析开来。学生可通过课堂练习题、课后习题和课后综合提高题等几个环节提升程序设计能力,达到由浅入深、举一反三进行程序设计实训的目的。

此外,本书重要章节(如循环、数组、函数、指针等章节)的课后习题均增加了面向各类

专业的应用与提高的部分习题，为各类专业学生了解计算机在本专业的应用提供感性认识。

本书还提供了教学资源丰富的课程网站。资源网站(http://202.4.152.136/)有电子教案、CAI 动画课件、自我测试题等供下载。

培养学生程序设计能力的方法仍在研究和探索中，最大限度地提高学生的学习效果是我们永恒的奋斗目标。

本书由从事了多年计算机基础课程教学、具有丰富教学实践经验的一线教师集体编写完成。第 1、2 章由李国捷编写，第 3 章及附录由高敬阳编写，第 4 章由吴蕾编写，第 5、9 章由马静编写，第 6、7 章由李芳编写，第 8 章由尤枫编写。全书由高敬阳、李芳组织编写并统稿，由朱群雄教授主审。

由于作者水平有限，书中难免有错误和不妥之处，恳请读者批评指正。

作者联系信箱：gaojy@mail.buct.edu.cn，lifang@mail.buct.edu.cn。

作者

2010 年 1 月

目录

第1章 概述

本章主要内容：

- C语言程序的基本结构；
- 程序设计基本概念；
- C语言的发展与特点；
- C语言的字符集；
- C语言的词法符号；
- 运行C程序的步骤和方法。

1.1 引 例

首先让我们看两个用C语言编写的程序。

例1-1 在屏幕上显示一行信息“This is the first C program!”。

程序代码如下：

```
#include <stdio.h>                                  /* 编译预处理命令 */
void main()                                         /* 定义主函数 main() */
{
    printf ("This is the first C program!\n");  /* 调用 printf()函数输出文字 */
}
```

运行结果：

```
This is the first C program!
```

程序中的#include <stdio.h>是编译预处理命令，因为后面调用的printf()函数是C语言提供的标准输出函数，在系统文件stdio.h中声明。

程序中/*……*/是程序的注释，用来说明程序的功能。

程序中的void main()定义了一个名称为main()的函数，关键字void表示函数无返回值。

一对大括号把构成函数的语句括起来，称为函数体。例1-1的函数体只有一条语句。

语句“printf("This is the first C program!\n");”由函数调用和分号两部分组成。“printf("This is the first C program!\n")”是一个函数调用，它的作用是将双引号中的内容原样输出；“\n”是换行符，即在输出“This is the first C program!”后换行；而分号表示该语句的结束。

例 1-2 求两数之和。

程序代码如下：

```
#include <stdio.h>                    /*编译预处理命令*/
void main()                           /*定义主函数 main()*/
{
    int a,b,sum;                      /*定义变量 a、b、sum 为整型*/
    a=66;                             /*为变量 a 赋值*/
    b=88;                             /*为变量 b 赋值*/
    sum=a+b;                          /*将 a 与 b 的和赋值给变量 sum*/
    printf ("sum is %d\n",sum);       /*调用 printf()函数输出 sum 的值*/
}
```

运行结果：

```
sum is 154
```

程序的第 4 行定义三个变量 a、b、sum 为整型(int)变量；程序的第 5、6 行都是赋值语句，分别使 a 的值为 66，b 的值为 88；程序的第 7 行也是一条赋值语句，使 sum 的值为 a+b；程序的第 8 行是输出函数调用语句，其中的“%d”是输入输出格式说明，表示“以十进制整数类型”输入输出相应的数据(详见第 2 章)；括号内逗号的右端 sum 是要输出的变量，现在它的值为 154(即 66 与 88 之和)；此函数调用后，双引号括起来的“sum is”按原样输出，在“%d”的位置上显示变量 sum 的值 154。

1.2 C 语言程序的基本结构

通过 1.1 节中的两个例子，我们可以看到 C 语言程序有以下的结构。

① C 程序由函数组成，函数是程序的基本单位。main 是一个特殊的函数名，一个程序总是从 main()函数开始执行。

② 函数由函数首部和函数体两部分组成。

函数首部用于定义函数的名称、函数的返回值类型和各种参数名称及数据类型(也可能没有参数及数据类型)。例如 void main()即是函数首部。

③ 函数体一般包括数据定义部分和执行部分，它们都是 C 语句。

④ 每条语句用分号“;”作结束符，分号是 C 语句必不可少的组成部分。

⑤ 在 C 语言中，一行可以写多条语句，一条语句也可写成几行。如将例 1-2 的三条语句合并成如下形式：

```
a=66;b=88;sum=a+b;
```

其结果和输出格式均不改变。

⑥ 可以对 C 程序中的任何部分做注释。一个好的、有使用价值的程序应当加上必要的注释，以改善程序的可读性和可维护性。注释可以占一行的一部分，也可以单独占一行，还可以占若干行。

1.3 程序设计基本概念

初学者应对下面几个有关程序设计的基本概念有所了解。

1.3.1 程序

所谓程序，就是一系列遵循一定规则和思想并能正确完成指定工作的代码(也称为指令序列)。通常，一个计算机程序主要描述两部分的内容，其一是描述问题的每个对象及它们之间的关系，其二是描述对这些对象进行处理的规则。其中关于对象及它们之间的关系涉及数据结构的内容，而处理规则却是求解某个问题的算法。因此，对程序的描述，经常有如下等式：

程序＝数据结构＋算法

一个设计合理的数据结构往往可以简化算法，而且一个好的程序有可靠性、易读性、可维护性等良好特性。

1.3.2 程序设计

所谓程序设计，就是根据计算机要完成的任务，提出相应的需求，在此基础上设计数据结构和算法，然后再编写相应的程序代码并测试该代码运行的正确性，直到能够得到正确的运行结果为止。通常，程序设计是很讲究方法的，一个良好的设计思想方法能够大大提高程序的效率和合理程度。通常程序设计有一套完整的方法，也称为程序设计方法学，因此有人提出如下关系：

程序设计＝数据结构＋算法＋程序设计方法＋语言工具和环境

程序设计方法学在程序设计中被提到比较高的位置，尤其对于大型软件更是如此。

1.3.3 程序设计语言

为了描述程序所制定的一组规则，即语法规则(主要包括词法规则与句法规则)。就像汉语与英语都有各自一整套的语法规则一样，众多的计算机语言，如 Basic 语言、Fortran 语言以及我们将要学习的 C 语言也都有各自一整套的语法规则。

1.4 C语言的发展与特点

1.4.1 C语言的发展

C语言是一种国际上广泛流行的、深受程序员喜爱的程序设计语言。

1967年，英国剑桥大学的Martin Richards在ALGOL 60的CPL语言基础上推出了BCPL (Basic Combined Programming Language)语言。

1970年，美国贝尔实验室的Ken Thompson在BCPL语言的基础上，设计了既简单又接近硬件的B语言(以BCPL首字母命名)。

1972年，贝尔实验室的Dennis M. Ritchie在B语言的基础上发明了C语言(以BCPL第二个字母命名)。

1978年，Brian W. Kernighan与Dennis M. Ritchie合作写出了著名的*The C Programming Language*。该书成为后来广泛使用的C语言版本的基础，称为标准C。

1983年，美国国家标准协会(ANSI)为C语言制定了一套标准，称为ANSI C。

1987年，ANSI又公布了87 ANSI C(新标准)。

1990年，国际标准化组织(ISO)接受87 ANSI C为ISO标准。目前流行的C语言版本都以此为基础。

1980年，贝尔实验室的Bjarne Stroustrup及其同事对C语言进行了改进，并将类的概念扩充到C语言中，在1983年由Rick Maseitti提议正式命名为C++语言。

C++是C的超集，C是C++的基础，用C语言编写的许多程序不经修改就可以在C++环境下运行。因此，学习C语言可为进一步学习C++及其他相关语言打下坚实的基础。

本教程以ANSI C标准为基础，书中的例题均在Microsoft Visual C++ 6.0集成环境下运行测试过。

1.4.2 C语言的特点

与其他高级语言相比，C语言的主要特点如下。

1. C语言是结构化、模块化的程序设计语言

C语言通过9种结构控制语句可描述各种结构的程序；以函数作为程序的基本单位，从而可实现模块化的程序设计。

2. C语言有强大的处理能力，适用面广

C语言既具有高级语言的功能，又能像低级语言一样对计算机最基本的工作单元(位、字节和地址)进行直接操作。因此，它既适宜编写大型系统程序，又适宜编写小型控

制程序，也适用于科学计算，并具有强大的图形处理功能。

3. C 语言语句简洁、紧凑，使用方便、灵活

C 语言一共只有 32 个保留字和 9 种控制语句，程序书写形式自由，压缩了一切不必要的成分。

4. 目标代码的效率高

用 C 语言程序生成的目标代码的效率可达到汇编语言目标代码效率的 80% ～90%。

5. 可移植性强

C 语言的输入输出不依赖于计算机硬件，使之能适应多种操作系统，如 DOS、UNIX、Windows 等，也适应多种机型。从而便于在各种不同的机器间实现程序的移植。

由于 C 语言具有上述特点，因此它得到了广泛的应用。

1.5 C 语言的字符集

字符是组成语言的最基本的元素。C 语言字符集由字母、数字、空白符、标点和特殊字符组成。在字符常量、字符串常量和注释中还可以使用汉字或其他可显示的图形符号。

① 英文字母：小写字母 a～z、大写字母 A～Z。

② 阿拉伯数字：0～9。

③ 空白符：空格符、制表符、换行符等统称为空白符。

空白符只在字符常量和字符串常量中起作用。在其他地方出现时，只起间隔作用，编译程序忽略它们。因此在程序中使用空白符与否，对程序的编译不发生影响，但在程序中适当的地方使用空白符将提高程序的清晰性和可读性。

④ 标点和特殊字符：

```
!     #     %     ^     &     *     _(下划线)
+     =     -     ~     <     >     /     \     '
"     ;     .     ,     ()    [ ]   { }   ?     :
```

1.6 C 语言的词法符号

词法符号是最小的词法单元。

C 语言的词法符号分为以下几类：关键字、标识符、运算符、分隔符、常量和注释符。

1. 关键字

关键字是 C 语言规定的具有特定意义的字符串，也称为保留字。C 语言有以下 32 个

关键字：

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

2. 标识符

在程序中使用的变量名、函数名、标号等统称为标识符。除库函数的函数名由系统定义外，其余都由用户自定义。C语言规定，标识符是英文字母或下划线开始的，英文字母、下划线及阿拉伯数字组成的字符串。

以下标识符是合法的：

```
a,x,_6y,UNIT_1,sum
```

以下标识符是非法的：

```
3c(以数字开头), t*v(出现非法字符*), -3m(以减号开头), unit-1(出现非法字符-)
```

在使用标识符时还必须注意以下几点：

① 用户定义的标识符不允许与关键字相同。

② 标准C不限制标识符的长度，但它受各种版本的C语言编译系统限制，同时也受到具体机器的限制。例如，在Turbo C 2.0中规定标识符前32位有效，当两个标识符前32位相同时，则被认为是同一个标识符。

③ 标识符中，大小写是有区别的。例如，xyz和XYZ是两个不同的标识符。

④ 标识符虽然可由程序员随意定义，但标识符是用于标识某个量的符号。因此，命名应尽量有相应的意义，做到“见名知义”。

3. 运算符

C语言中含有十分丰富的运算符。运算符与常量、变量和函数一起组成表达式，表示各种运算功能。运算符由一个或多个字符组成，例如加(+)、减(-)、乘(*)、除(/)、大于(>)、小于(<)、条件运算(?:)等。

4. 分隔符

在C语言中采用的分隔符有逗号和空格两种。逗号主要用在类型说明和函数参数表中分隔各个变量。空格多用于语句中分隔各单词。

5. 常量

C语言中使用的常量可分为数值常量、字符常量、字符串常量、符号常量、转义字符等

多种。在第 2 章中将专门给予介绍。

6. 注释符

程序编译时,不对注释作任何处理。注释可出现在程序中的任何位置。注释用来向用户提示或解释程序的意义。在调试程序时对暂时不使用的语句也可用注释符,使编译跳过不作处理,待调试结束后再去掉注释符。

1.7 运行 C 程序的步骤和开发环境

1.7.1 运行 C 程序的步骤

如何使用 C 语言写出代码,并调试程序直至得出运行结果呢?一般来说包含如下的步骤:

1. 编辑

编辑的过程指用程序设计语言写出源代码的过程。一些常用的编辑软件,如记事本、Microsoft Word 等都可以完成此功能。

2. 编译

对程序进行编译是将源程序翻译成机器能够识别的目标程序的过程。此过程必须借助一些专门的编译程序(编译器)来完成。

3. 连接

简单地讲,连接过程是将不同的模块连接成一个完整模块的过程。假如一个程序包含多个文件,在分别对每个源程序文件进行编译并得到多个目标程序后,连接就是要把这些目标程序以及系统提供的资源(通常是一些库函数)组合起来形成一个整体。此过程必须通过连接程序(连接器)来完成,从而形成一个完整的可执行程序。

4. 执行

一个程序经过了编辑、编译、连接过程,就得到了可执行程序,于是可以执行了。我们可以在命令行方式下敲入文件名,按回车键执行该程序。也可以在 Windows 环境中,双击该可执行程序执行。

上述编辑、编译、连接直至执行的过程可用如图 1-1 所示的程序调试流程来描述。

这一过程最初是分别进行的,即要完成一个程序的调试,必须首先找到相应的编辑、编译和连接工具,依次进行编辑、编译和连接,得到可执行程序,从而进一步执行程序得到最终的结果。如果此过程中的任何一个步骤出现问题,则需要进行修改并重复相应步骤。

可见,由于人们必须分散地使用各个工具,这使得整个程序调试流程相当繁杂。有没

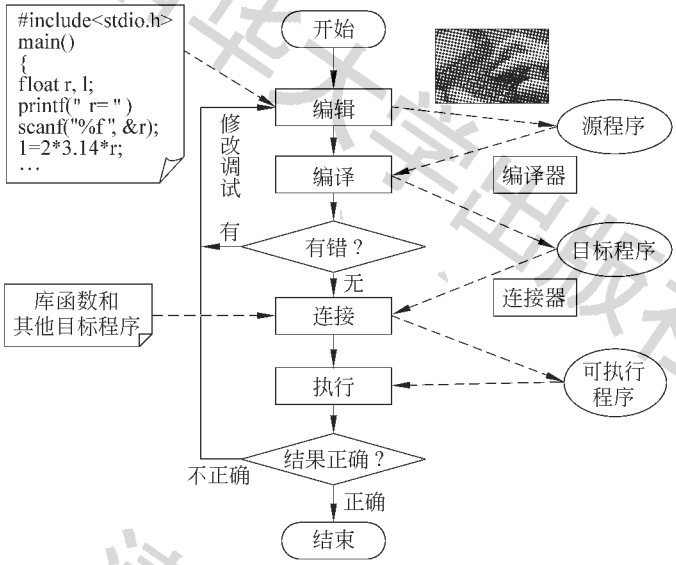

图 1-1　程序调试流程

有更简单的方法呢？很快，人们就找到了将上述步骤集成起来的方法，于是就形成了集成开发环境(IDE，Integrated Development Environment)。所谓集成开发环境就是集源程序编辑、编译、连接、运行和调试于一体，用菜单驱动的集成化的软件开发工具。

常见的 C 语言集成开发环境有 Turbo C 2.0、Turbo C 3.0 以及 Visual C++ 6.0 (简称 VC++ 6.0)等。

下面仅对使用 Visual C++ 6.0 集成开发环境开发 C 语言程序的过程进行简单介绍。

1.7.2　集成开发环境

先在磁盘上新建一个文件夹，用于存放源程序，如在磁盘 E:上建立 E:\200840000 (自己的学生学号)文件夹。

然后按照以下步骤进行操作，了解开发一个 C 程序的基本步骤：

① 启动 VC++ 6.0(操作系统为 Windows XP 环境)。选择“开始”→“所有程序”→Microsoft Visual Studio 6.0→Microsoft Visual C++ 6.0 命令，进入 VC++ 6.0 集成开发环境(如图 1-2 所示)。

② 新建文件。选择 File→New 命令，单击 Files 标签(如图 1-3 所示)，先在 Location 目录下拉列表框中选择已经建立好的文件夹，如 E:\200840000；然后在 File 文本框中输入 test001，把 C 语言源程序文件命名为 test001.cpp，最后双击C++ Source File 选项，于是在 E:\200840000 文件夹下就新建了文件test001.cpp，并显示出编辑窗口和信息窗口。

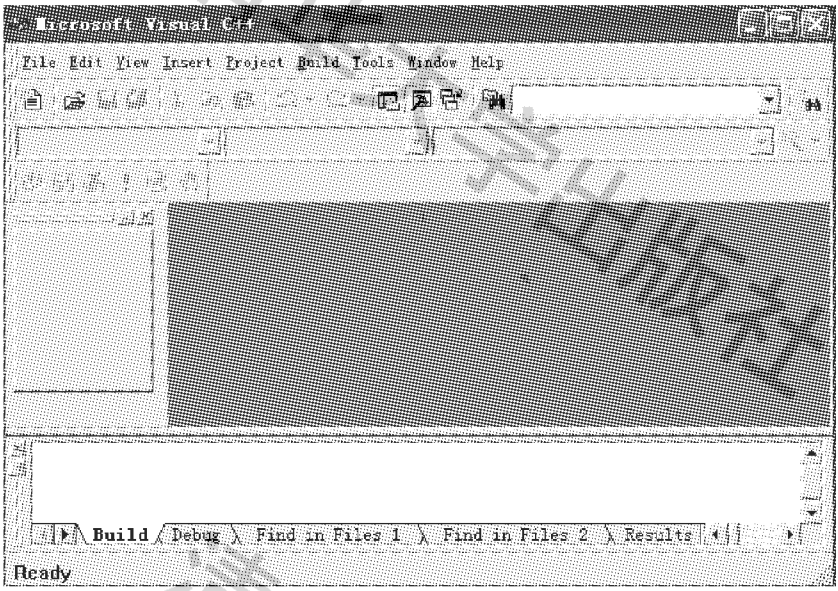

图 1-2　VC++ 6.0 集成开发环境主窗口

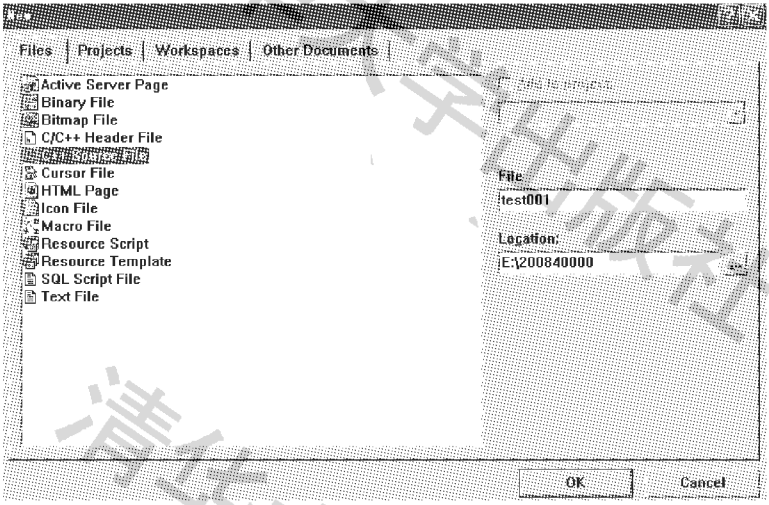

图 1-3　新建文件窗口

③ 编辑源文件并保存。在编辑窗口中输入源程序(如图 1-4 所示)，然后选择 File→Save 命令，保存该源文件。

④ 编译。选择 Build→Compile test001.cpp 命令(如图 1-5 所示)，在弹出的消息框中单击 OK 按钮，开始编译，并在信息窗口中显示编译信息(如图 1-6 所示)。

在图 1-6 所示的信息窗口中出现了“test001.obj-0 error (s)，0 warning (s)”，表示编译通过，没有发现错误和警告，于是生成了目标文件 test001.obj。

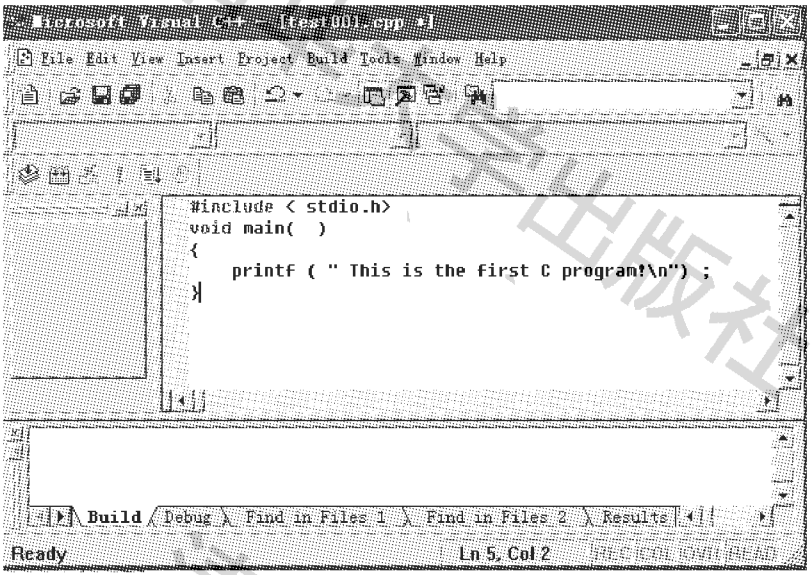

图 1-4　编辑源程序窗口

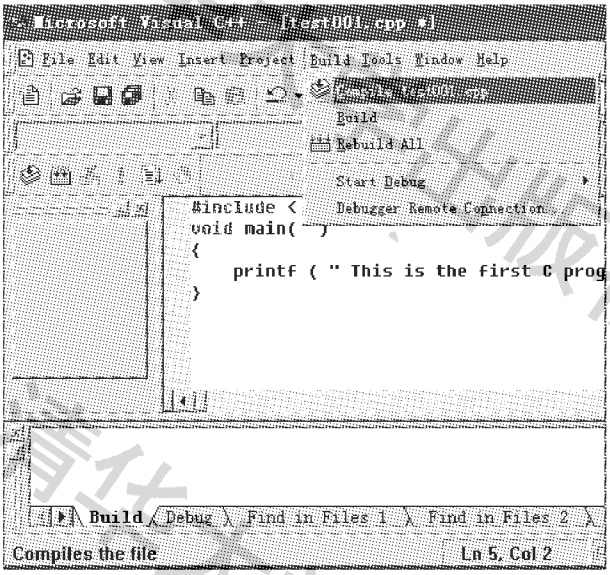

图 1-5　编译源程序窗口

如果显示错误信息，说明程序中存在严重的语法错误，必须修改；如果显示警告信息，说明这些错误暂时未影响目标文件的生成，但最好查明原因予以修改。

⑤ 连接。选择 Build→Build test001.exe 命令，开始连接，并在信息窗口中显示连接信息。

在信息窗口中如果出现“test001.exe - 0 error (s)，0 warning (s)”，则表示连接成功，于是生成了可执行文件 test001.exe。

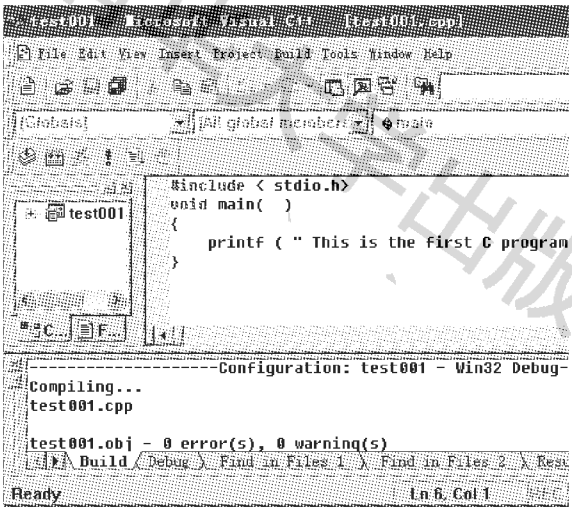

图 1-6　编译正确窗口

⑥ 运行。选择 Build→Execute test001.exe 命令(如图 1-7 所示),自动弹出运行窗口(如图 1-8 所示),显示运行结果“This is the first C program!”。其中,Press any key to continue 是系统提示用户按任意键退出运行窗口,返回到 Visual C++ 编辑窗口。

⑦ 关闭程序工作区。选择 File→Close Workspace 命令(如图 1-9 所示),在弹出的对话框中单击 OK 按钮,关闭工作区。

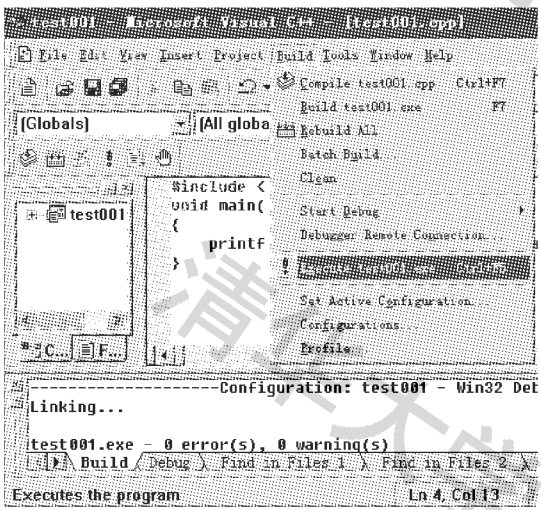

图 1-7　运行程序窗口

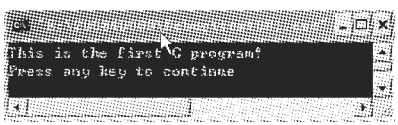

图 1-8　运行结果窗口

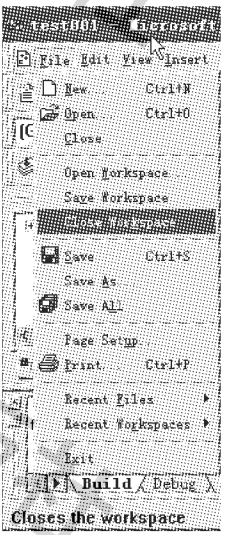

图 1-9　关闭工作区窗口

⑧ 退出VC++ 6.0。选择 File→Exit 命令，退出VC++ 6.0集成开发环境。

经过编辑、编译、连接和执行后，在文件夹 E:\200840000 和 E:\200840000\Debug 中存放着新生成的相关文件。其中，源程序 test001.cpp 在文件夹 E:\200840000 中，目标程序 test001.obj 和可执行程序 test001.exe 都在文件夹 E:\200840000\Debug 中。

本章小结

本章简述了 C 语言程序的基本结构、几个程序设计的基本概念、C 语言的发展与特点、C 语言的字符集和词法符号以及运行 C 程序的步骤和开发环境。

习　题　1

1. C 语言的基本单位是什么？
2. C 源程序经过哪两个步骤才能生成可执行程序？
3. 编写一个程序，显示以下信息：

```
******************************
    I am a good student!
******************************
```

4. 编写一个程序，求两数之差。

第2章 用C语言编写简单程序

本章主要内容：

- 数据类型、运算符和表达式；
- 格式输入与输出；
- C语言的语句；
- 顺序结构程序设计。

2.1 引 例

例 2-1 求摄氏温度100℃对应的华氏温度，计算公式如下：$f=\frac{9}{5}c+32$，式中：c表示摄氏温度，f表示华氏温度。

分析：如何将数学公式转换成符合C语言语法规则的语句是本题的关键。

程序代码如下：

```
#include <stdio.h>
void main()
{
    float celsius,fahr;                    /*定义两个实型变量*/
    celsius=100;                           /*对变量 celsius 赋值*/
    fahr=9.0/5.0*celsius+32;               /*温度转换计算*/
    printf("celsius=%f,fahr=%f\n",celsius,fahr);     /*显示计算结果*/
}
```

运行结果：

```
celsius=100.000000, fahr=212.000000
```

如何将“$f=\frac{9}{5}c+32$”转换为“fahr=9.0/5.0*celsius+32”是本章要解决的主要问题。“fahr=9.0/5.0*celsius+32;”中涉及了常量、变量、运算符、表达式以及语句几个概念，如图2-1所示。

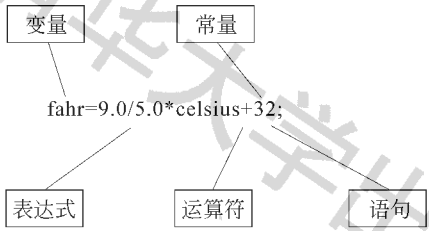

图 2-1　例 2-1 程序关键语句的分析

2.2　数 据 类 型

2.2.1　C 语言的数据类型

数据类型是 C 语言中允许使用的数据的种类。不同数据类型决定了该类型数据的取值范围及精度等属性。C 语言提供了如图 2-2 所示的数据类型。

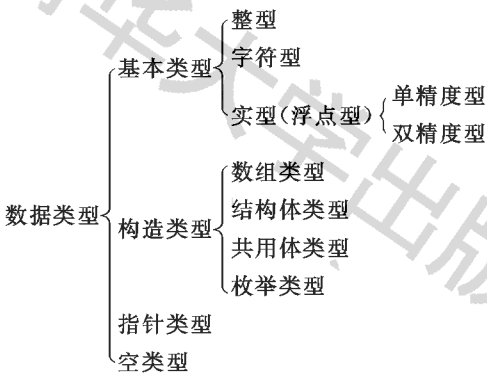

图 2-2　C 语言的数据类型

在编写程序时要正确地使用这些数据类型。

2.2.2　常量和变量

对于基本类型数据，按其取值是否可改变分为常量和变量两种。在程序执行过程中，其取值不发生改变的量称为常量，取值可变的量称为变量。常量可以不经说明而直接引用，而变量则必须先说明后使用。变量定义的一般形式为：

数据类型 变量名表；

在 C 语言中，对于某些有特定含义的、经常使用的常量可以用符号常量来代替。使用符号常量，可增加程序的可读性和可维护性。符号常量定义的一般格式为：

#define　符号常量　常量

其中，#define 是定义符号常量的预处理命令，详见附录 D。例如：

```
#define PI 3.14159
```

定义了一个符号常量 PI，用来代替常量 3.14159。

符号常量习惯用大写字母表示。

2.2.3 整型数据

整型数据包括整型常量、整型变量。

1. 整型常量

整型常量就是整常数。在 C 语言中，整型常量有八进制、十六进制和十进制三种。

1）八进制整常数

八进制整常数必须以 0 开头，即以 0 作为八进制数的前缀。

以下各数是合法的八进制数：

016(十进制为 14)　0101(十进制为 65)

2）十六进制整常数

十六进制整常数的前缀为 0X 或 0x。

以下各数是合法的十六进制整常数：

0X1A(十进制为 26)　0XA0 (十进制为 160)　0XFFFF(十进制为 65535)

3）十进制整常数

以下各数是合法的十进制整常数：

－258　　678　　1828

2. 整型变量

整型变量定义的一般形式为：

整型类型 变量名表；

可以根据数据的取值范围和所占内存的字节数，将变量定义为 6 种整型类型，详见表 2-1。

表 2-1　VC++ 6.0 中各整型类型的表示、分配的字节数和取值范围

类型定义关键字	名　　称	分配的字节数	数的取值范围
[signed] int	基本整型	4	－2 147 483 648～2 147 483 647
[signed] short [int]	短整型	2	－32 768～32 767
[signed] long [int]	长整型	4	－2 147 483 648～2 147 483 647
unsigned [int]	无符号基本整型	4	0～4 294 967 295
unsigned short [int]	无符号短整型	2	0～65 535
unsigned long [int]	无符号长整型	4	0～4 294 967 295

例如,“int a,b,c;”定义了三个基本整型变量。

实际上,变量名是一个与某一存储单元相联系的符号地址,而变量值是指存放在该存储单元中的数据。换言之,每一个变量具有两种值,其一为变量的值,其二为变量的地址值。假如基本整型变量 a、b 的值分别为 100、200,其地址分别为 11020000、11020004,则两个变量的名称、值和地址的关系如图 2-3 所示。

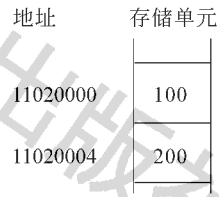

图 2-3　变量名、变量值和地址的关系

在程序中,我们经常会从变量中存取数据,实际上是先通过变量名找到相应存储单元的地址,然后再对该地址所对应的存储单元进行存入或取出数据操作。

2.2.4　实型数据

实型数据包括实型常量、实型变量。

1. 实型常量

实型也称为浮点型。实型常量也称为实数或者浮点数。在 C 语言中,实数只采用十进制。它有两种形式:十进制小数形式和指数形式。

1) 十进制小数形式

由 0～9 和小数点组成。例如,0.0、－2.87、3.8、4. 和.77 均为合法的实数。

2) 指数形式

由十进制数加 e 或 E 以及指数组成,其一般形式为:

a E n

上面式子中,a 为十进制数,n 必须为十进制整常数,其值为 $a\times10^n$,如 3.4E6 (等于 3.4×10^6), 9.6E-4(等于 9.6×10^{-4})。

2. 实型变量

实型变量分为两类:单精度型和双精度型,类型说明关键字分别为 float 和 double。单精度型一般占 4 个字节(32 位)内存空间,其数值范围为－3.4E＋38～3.4E＋38,只能提供 7 位有效数字;双精度型一般占 8 个字节(64 位)内存空间,其数值范围为－1.7E＋308～1.7E＋308,可提供 15～16 位有效数字。

实型变量说明的一般形式与整型相同。例如:

```
float x,y;                /* x、y 为单精度实型变量 */
double a,b,c;             /* a、b、c 为双精度实型变量 */
```

实型常数默认为双精度型。

2.2.5 字符型数据

字符型数据包括字符常量、字符变量。

1. 字符常量

字符常量是用单引号括起来的一个字符。如'x'、'y'、'$'、'?'都是合法的字符常量。

除了以上形式的字符常量外，C 语言还允许使用一种特殊形式的字符常量，就是以反斜线\开头的字符序列。此字符序列具有特定的含义，故称“转义”字符。例如，在前面各例题 printf 函数的格式串中用到的\n 就是一个转义字符，其功能是“换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。

常用的转义字符及其功能如表 2-2 所示。

表 2-2 常用转义字符及其功能

\n	换行
\t	横向跳到下一制表位置
\"	双引号
\'	单引号
\\	反斜线
\ddd	1～3 位八进制数所代表的字符(ASCII)
\xhh	1～2 位十六进制数所代表的字符(ASCII)

2. 字符变量

字符变量的取值是字符常量，即单个字符。字符变量的类型说明符是 char。字符变量说明的一般形式如下：

char 变量表；

例如：char a,b;

每个字符变量被分配一个字节的内存空间，因此只能存放一个字符。将一个字符常量存放到一个字符变量中，实际上并不是把该字符本身存放到内存单元中去，而是将该字符所对应的 ASCII 码存放到变量的内存单元中。如 x 的十进制 ASCII 码是 120，y 的十进制 ASCII 码是 121。对字符变量 a 和 b 分别赋予'x'和'y'值：“a='x';b='y';”，实际上是在 a 和 b 两个单元内存放 120 和 121 的二进制代码，如图 2-4 所示。

图 2-4 在 a 和 b 两个单元内存放 120 和 121 的二进制代码

所以也可以把它们看成是整型值。C 语言允许对字符变量赋以整型值，也允许对整型变量赋以字符值。在输出时，允许把字符型数据按整型数据形式输出，也允许把整型数据按字符型数据形式输出。但整型数据至少占两字节的内存空间，而字符型数据仅占单字节的内存空间，当整型数据按字符型数据处理时，只有低八位字节参与处理。

2.2.6 变量赋初值

在程序中常常需要对一些变量预先设置初值，以便使用变量。C 语言程序中可在定义变量的同时赋以初值，这种方法称为初始化。在变量定义中赋初值的一般形式为：

类型说明符　变量 1=值 1,变量 2=值 2,…;

例如：

```
int a=5, b=6;
float x=4.7, y=38.6, z=8.72;
```

2.3 运算符与表达式

2.3.1 C 语言运算符简介

C 语言中运算符和表达式数量之多，在高级语言中是少见的。正是丰富的运算符和表达式使得 C 语言功能十分完善。这也是 C 语言的主要特点之一。

C 语言的运算符不仅具有不同的优先级，而且还有一个特点，就是它的结合性。C 语言中，运算符的运算优先级共分为 15 级。1 级最高，15 级最低。在表达式中，优先级较高的先于优先级较低的进行运算。而在一个运算量两侧的运算符优先级相同时，则按运算符的结合性所规定的结合方向处理。C 语言中各运算符的结合性分为两种，即左结合性(自左至右运算)和右结合性(自右至左运算)。详见附录 B。

2.3.2 算术运算符与算术表达式

算术运算符用于各类数值运算，包括加(+)、减(-)、乘(*)、除(/)、求余(或称模运算，%)、负值(-)、自增(++)和自减(--)运算符。

1. 基本的算术运算符

① 加法运算符+，为双目运算符，即前后应有两个量参与运算，具有左结合性。

② 减法运算符-，为双目运算符，具有左结合性。

③ 乘法运算符*，为双目运算符，具有左结合性。

④ 除法运算符/，为双目运算符，如a/6、18/4，具有左结合性。此运算符前后参与运算的量均为整型时，结果也为整型，舍去小数。18/4的计算结果为4而非4.5。

⑤ 求余运算符%，为双目运算符，如7%3，具有左结合性。此运算符要求参与运算的量均为整型，计算的结果为两数相除的余数。7%3的计算结果为1，8%4的计算结果为0。

运算符*、/、%优先级相同，运算符+、-优先级也相同，但*、/、%的优先级比+、-的优先级高，即“先算乘除后算加减”。

2. 负值运算符

负值运算符-，为单目运算符，如-5和-y，具有右结合性。

3. 自增、自减运算符

自增运算符记为“++”，其功能是使变量的值增1。自减运算符记为“--”，其功能是使变量的值减1。自增、自减运算符均为单目运算符，都具有右结合性，且只能用于变量。可有以下几种形式：

++i　i增1后再参与其他运算。

--i　i减1后再参与其他运算。

i++　i参与运算后，i的值再增1。

i--　i参与运算后，i的值再减1。

例如，当有如下语句时

```
int x=5,y;
y=x++;
```

则y的值为5；

然而，当有如下语句时

```
int x=5,y;
y=++x;
```

则y的值为6。

4. 算术表达式

算术表达式是由算术运算符和括号将运算对象连接起来的式子，如：

```
(a*5)/b   (x+y)/(a-b)     ++i   sin(x)+cos(y)   (++i)+(k--)
```

2.3.3 赋值运算符与赋值表达式

赋值运算符记为“=”。由“=”连接的式子称为赋值表达式。其一般形式为：

变量=表达式

例如：

```
x=a+b
y=b+x++
```

这两个式子都是正确的赋值表达式。赋值表达式的功能是先计算表达式的值再赋予左边的变量。赋值运算符具有右结合性。因此，“x＝y＝z＝8”可理解为“x＝(y＝(z＝8))”。

在有些高级语言中，赋值构成了一条语句，称为赋值语句。而在C语言中，把＝定义为运算符，从而组成赋值表达式。

如果赋值运算符两边的数据类型不相同，系统将自动进行类型转换，即把赋值号右边的类型换成左边的类型。

在赋值运算符＝之前加上其他二目运算符可构成复合赋值运算符。这些运算符共有10个：＋＝、－＝、＊＝、/＝、％＝、<<＝、>>＝、&＝、^＝和|＝。

例如，“x＋＝8”等价于“x＝x＋8”，“a＊＝b＋6”等价于“a＝a＊(b＋6)”。

复合赋值运算符的这种写法，对初学者可能不习惯，但十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

2.3.4 逗号运算符与逗号表达式

C语言中逗号(,)也是一种运算符，称为逗号运算符。其功能是把两个表达式连接起来组成一个表达式，称为逗号表达式。逗号表达式的一般形式为：

表达式1,表达式2,…,表达式n

其求值过程是先计算表达式1的值，然后计算表达式2的值，依次类推，最后计算表达式n的值，并以表达式n的值作为整个逗号表达式的值。

例如，逗号表达式“a＝3,b＝5,c＝a＋b”的值为8。

并不是在所有出现逗号的地方都组成逗号表达式，如在变量说明、函数参数表中逗号只是用作各变量之间的分隔符。

2.4 各类数值型数据间的混合运算

在C语言中，允许整型、实型、字符型数据间进行混合运算。首先按运算符的优先级将运算符两侧的数据转换成同一类型，然后再进行运算。数据类型的转换包括自动转换和强制转换。自动转换由C语言编译系统自动完成，强制转换则通过特定的运算完成。

2.4.1 自动类型转换

1. 非赋值运算的类型转换

数据类型的自动转换是将占用字节数少的向占用字节数多的数据类型转换。对于各

种数据类型，其转换规则如图 2-5 所示，具体说明如下：

① 水平方向的转换：自动从右向左转换，见图 2-5 中的左箭头方向。

② 垂直方向：经过水平方向的转换后，若参加运算的数据的类型仍然不同，再将这些数据自动转换成其中级别最高的类型。见图 2-5 中的上箭头方向。

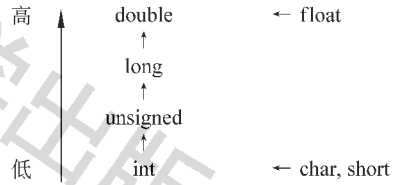

图 2-5　数据类型自动转换规则

例如：int 型与 double 型数据进行运算，直接将 int 型数据转换为 double 型进行运算，而不是先转换成 unsigned 型，然后再转换成 long 型，最后再转换成 double 型。

2. 赋值运算的类型转换

当赋值运算符两侧的类型不一致时，其转换规则是将赋值运算符右侧的类型转换为左侧变量的类型，然后再进行赋值运算。这种情况下，可能会引起数值溢出或产生舍入误差。

例如：

```
int i;
i=5.67;
```

则 i 的值为 5，即以整数形式存储在整型变量中。

2.4.2　强制类型转换

使用强制转换运算符，可以将一个表达式转换成给定的类型。其一般形式为：

(数据类型名)表达式

例如，(long)a 强制将 a 临时转换为长整型，(float)(x+y)强制将 x+y 的结果临时转换为单精度实型。

无论是强制转换还是自动转换，都只是一种作用于本次运算的临时性转换，而不会改变数据原来的类型。

2.5　数据的输入输出

在 C 语言中，所有的数据输入输出都是由库函数完成的。在程序中只需调用这些函数来完成相应的输入输出操作。本节先介绍标准字符输入输出函数，然后再介绍格式输出 printf()函数和格式输入 scanf()函数。需要指出的是，本节中介绍的输入输出函数在使用前，都需要使用预处理命令“#include”将头文件 stdio.h 包含到用户源程序中，即 #include <stdio.h>。

2.5.1 标准字符输入输出函数

C语言函数库提供了getchar()、putchar()等以键盘为标准输入设备、以显示器为标准输出设备的标准字符输入输出函数。

1. 标准字符输出函数 putchar()

putchar()函数向标准输出设备(一般为显示器)输出一个字符,其调用的一般形式为:

```
putchar(c)
```

其中,c可以是一个字符常量、字符变量、整型常量、整型变量或整型表达式。

2. 标准字符输入函数 getchar()

getchar()函数从标准输入设备(一般为键盘)读入一个字符,并立即在显示器上显示该字符(称作回显),其调用的一般形式为:

```
getchar()
```

2.5.2 格式输出函数 printf()

printf()函数的作用是向显示器输出若干个任意类型的数据。printf()函数调用的一般形式为:

```
printf("格式控制字符串",输出表列)
```

其中格式控制字符串用于指定输出格式。格式控制字符串可由格式字符串和非格式字符串两部分组成。格式字符串是以%开头的字符串,在%后面跟有各种格式字符,以说明输出数据的类型、长度、小数位数等。非格式字符串原样输出,在显示中起提示作用。输出表列中列出了各个输出项,要求格式字符串和各输出项在数量和类型上应该一一对应。

格式字符串的一般形式为:

```
%[附加格式说明符]格式字符
```

其中,格式字符用于说明输出数据的类型。printf()函数中可使用的格式字符及其功能详见表2-3。

可选项附加格式说明符位于%与格式字符之间,用于说明输出数据的宽度、小数位数、数据对齐形式等。表2-4给出了各种附加格式说明符及其作用。

表 2-3　printf()函数中可使用的格式字符及其功能

d、i	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 0)
x、X	以十六进制形式输出无符号整数(不输出前缀 0x)。用 x 时,以小写形式输出包含 a～f 的十六进制数;用 X 时,以大写形式输出包含 A～F 的十六进制数
u	以十进制形式输出无符号整数
c	输出单个字符
s	输出字符串
f	以小数形式输出单、双精度实数
e、E	以指数形式输出单、双精度实数
g、G	以%f、%e 中较短的输出宽度输出单、双精度实数

表 2-4　printf()函数的附加格式说明符及其作用

整数 m	m 为十进制正整数,表示输出的最少位数。若实际位数多于定义的宽度,则按实际位数输出;若实际位数少于定义的宽度,则补空格或 0
.	将整数 m 与整数 n 分开
整数 n	n 为十进制正整数,对于实数,表示小数的位数;对于字符串,则表示截取的字符个数
字母 l	输出 long 型整数
0	若数据的实际位数少于定义的宽度,则左边补 0
+	输出数据右对齐,并在正数前输出+
−	输出的数字或字符向左对齐

例 2-2　格式输出整型数和实型数。

```
#include  <stdio.h>
void main()
{   int a=18;
    float b=3.1415926;
    printf("a=%d,b=%f\n",a,b);
}
```

运行结果:

```
a=18,b=3.141593
```

2.5.3　格式输入函数 scanf()

scanf()函数称为格式输入函数,即按用户指定的格式从键盘上把数据输入到指定的变量之中。scanf()函数调用的一般形式为:

scanf("格式控制字符串",地址表列)

其中,格式控制字符串的作用与 printf()函数相同,但不能显示非格式字符串,也就是不能显示提示字符串。地址表列是需要接受输入数据的所有变量的地址,而不是变量本身,这与 printf()函数完全不同。若有多个地址,各地址之间要用逗号“,”分隔。地址是由变量名前加地址运算符 & 组成的。例如,&a、&b 分别表示变量 a 和变量 b 的地址,这个地址就是编译系统在内存中给变量 a、b 分配的地址。在 C 语言中,使用了地址这个概念,这是与其他语言不同的。变量的具体地址是什么读者可以不必关心。

scanf()函数中格式字符串的一般形式为:

%[附加格式说明符]格式字符

其中格式字符与 printf()函数中的格式字符相同,附加格式说明符主要有两个:字母 l 和整数 m,其作用见表 2-5。

表 2-5 scanf()函数的附加格式说明符及其作用

字母 l	用于输入 long 型整数(%ld)或 double 型实数(%lf)
整数 m	用于指定输入数据的宽度

使用 scanf()函数还必须注意以下几点:

① scanf()函数中没有精度控制,如“scanf("%8.4f",&a);”是非法的。不能企图用此语句输入 4 位小数的实数。

② 在输入多个数值数据时,若格式控制串中没有非格式字符作输入数据之间的间隔,则可用空格、Tab 键或回车作间隔。若格式控制串中有非格式字符作输入数据之间的间隔,则必须用该非格式字符作间隔。

例如,下面一条语句:

```
scanf("%d,%d,%d",&a,&b,&c);
```

在运行时,必须以“6,7,8”(逗号分隔)的方式才能正确地将此三个数分别输入到变量 a、b、c 中。

例 2-3 将例 2-1 改为输入一摄氏温度,求其对应的华氏温度。

程序代码如下:

```
#include <stdio.h>
void main()
{
    double celsius,fahr;
    printf("请输入摄氏温度:");
    scanf("%lf",&celsius);
    fahr=9.0/5.0*celsius+32;
    printf("celsius=%lf,fahr=%lf\n",celsius,fahr);
}
```

运行情况：

```
请输入摄氏温度：100↙
celsius=100.000000,fahr=212.000000
```

练习 2-1 键盘输入一个字符，输出该字符及其对应的 ASCII 码。

输入输出是程序设计的基本操作，而 C 语言的格式输入又较难掌握，数据输入不当就得不到预期的结果。初学者应先重点掌握最常用的一些规则，然后伴随程序的编写和调试再逐步深入理解许多细节问题。

2.6 顺序结构程序设计

2.6.1 C 语言的语句

C 语言程序的执行部分是由语句组成的。程序的功能也是由执行语句实现的。C 语言的语句可分为以下 5 类：表达式语句、函数调用语句、控制语句、空语句和复合语句。

1. 表达式语句

表达式语句由表达式加上分号(;)组成。执行表达式语句就是计算表达式的值。

2. 函数调用语句

由函数名、实际参数加上分号(;)组成。执行函数语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，详见第 6 章。例如

```
printf("This is the first C program");
```

调用库函数，输出字符串。

3. 控制语句

控制语句用于完成一定的控制功能，以实现结构化程序设计。C 语言有九种控制语句。可分成以下三类：

① 条件判断语句：if 语句、switch 语句。

② 循环语句：while 语句、do while 语句、for 语句。

③ 转向语句：goto 语句、break 语句、continue 语句、return 语句。

4. 空语句

只有分号(;)组成的语句称为空语句。空语句是什么也不执行的语句。

5. 复合语句

把若干条语句用大括号{}括起来组成的语句称为复合语句。在程序中把复合语句看

成是一条语句，而不是多条语句，例如

```
{
    z=x+y;
    t=z/10.0;
    printf("%f",t);
}
```

是一条复合语句。复合语句内的每一条语句都必须以分号“;”结尾。

2.6.2 顺序结构程序设计举例

顺序结构由一组按先后书写顺序执行的程序块组成。顺序结构是结构化程序设计中最简单的一种。

下面介绍几个顺序结构程序设计的例子。

例 2-4 输入三角形的三条边长，求该三角形的面积。

分析：设输入的三角形三条边长为 a、b、c，则三角形面积的计算公式为：$\sqrt{s(s-a)(s-b)(s-c)}$，其中 $s=\frac{1}{2}(a+b+c)$。

程序代码如下：

```
#include <stdio.h>
#include <math.h>
void main()
{
  double a,b,c,s,area;
  printf("请输入三角形的三条边 a,b,c:");
  scanf("%lf,%lf,%lf",&a,&b,&c);          /*注意输入数据之间的分隔符*/
  s=1.0/2*(a+b+c);                        /*注意整型常量相除及乘号*/
  area=sqrt(s*(s-a)*(s-b)*(s-c));         /*公式转换*/
  printf("a=%7.2f  b=%7.2f  c=%7.2f\n",a,b,c);
  printf("s=%7.2f  area=%7.4f\n",s,area);
}
```

运行情况：

```
请输入三角形的三条边 a,b,c: 6,8,10
a=      6.00    b=      8.00 c=     10.00
s=     12.00    area=24.0000
```

例 2-5 从键盘上输入一个小写字母，要求转化为大写字母并输出。

程序代码如下：

```
#include <stdio.h>
void main()
```

```
{
    char c1,c2;
    printf("Please input a lower letter: ");
    scanf("%c",&c1);
    c2=c1-32;       /*小写字母与大写字母的ASCII差值为32*/
    printf("Upper letter is %c\n",c2);
}
```

运行情况：

```
Please input a lower letter: d
Upper letter is D
```

练习 2-2 从键盘上输入一个大写字母，将其转化为小写字母并输出。

例 2-6 读入圆的半径 r，计算该圆的周长及面积。

程序代码如下：

```
#define PI 3.141593
#include <stdio.h>
void main()
{
    double r,circum,area;
    printf("请输入半径:");
    scanf("%lf",&r);                    /*双精度变量格式输入必须用%lf*/
    circum=2*r*PI;
    area=PI*r*r;
    printf("r=%10.2lf,circum=%10.2lf,area=%10.2lf\n",r,circum,area);
}
```

运行情况：

```
请输入半径：3
r=              3.00,circum=          18.85,area=          28.27
```

本章小结

C 语言的数据类型、运算符及表达式非常丰富，输入输出格式说明较为复杂。初学者应先重点掌握基础内容，然后在后续编写程序中逐步掌握细节问题。

习 题 2

1. 下列变量名中合法的是_____。

A) B. C. Tom　　B) 3a6b　　C) _6a7b　　D) $ABC

2. 正确的定义变量的语句是_____。

A) int ab_;　　B) int-ab;　　C) char mm　　D) float a3. b;

3. 求下面算术表达式的值：

① 3.5+9%4 * 3 * (1/2)-1.5

② (int)x+6%(int)(y/2.0)，设 x=2.8，y=4.4

4. 写出下面程序的运行结果：

```
#include <stdio.h>
void main()
{
    int a=5,x,y=8,z;
    x=++a;
    z=-y--;
    printf("%d,%d,%d\n",x,y,z);
}
```

5. 输入一球体的半径 r，求该球体的表面积和体积。

6. 输入两组数据 $x1$、$y1$ 和 $x2$、$y2$，分别代表平面直角坐标系中的两个点，求此两点间的距离。

7. 输入一个三位数的正整数，分别求出该数的百位数、十位数和个位数的数值。

8. 输入两个字符到字符变量 a、b 中，交换 a、b 的值，并输出交换后 a、b 的值。

第3章 分支结构程序设计

本章主要内容：

- 条件分支 if 语句，包括 if-else、if 和 if-else if 三种形式；
- 开关分支 switch 语句。

3.1 引　　例

例 3-1 有一个函数，定义如下：$y=f(x)=\begin{cases}0 & (x<0)\\ x & (x\geqslant 0)\end{cases}$

编写一程序，根据用户输入的自变量 x 的值，计算函数值。

分析：首先从键盘输入自变量 x 的值，然后根据其值判断所属的范围，执行对应的语句。

程序流程图如图 3-1 所示。

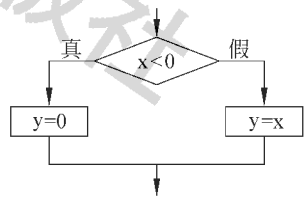

图 3-1　求函数值流程图

程序代码如下：

```
#include <stdio.h>
void main()
{ int x,y;
  printf("Please input  x:");
  scanf("%d",&x);                 /*输入数据*/
  if ( x<0 )                      /*对数据 x 进行判断*/
       y=0;                       /*如果 x<0,则执行 y=0*/
  else
       y=x;                       /*如果 x≥0,则执行 y=x*/
  printf("y=%d\n",y);             /*输出函数值*/
}
```

例 3-2 输入三角形的三条边长，求三角形面积。

分析：此例在第 2 章中已经做过。为了使其完整，首先要判断输入的三条边是否能够构成三角形。任意两边之和都要大于第三边，这是构成三角形的基本条件。

程序代码如下：

```
#include <stdio.h>
#include <math.h>
void main()
{ float a,b,c,s,area;
  printf("请输入三角形的三条边 a,b,c:");
  scanf("%f,%f,%f",&a,&b,&c);
  if(a+b>c && a+c>b && b+c>a)          /*判断能否构成三角形*/
  {                                    /*条件成立,以下复合语句求面积并输出结果*/
    s=1.0/2*(a+b+c);
    area=sqrt(s*(s-a)*(s-b)*(s-c));
    printf("a=%7.2f  b=%7.2f c=%7.2f\n",a,b,c);
    printf("s=%7.2f area=%7.4f\n",s,area);
  }
  else
  printf("此三条边不能构成三角形!\n");/*条件不成立,输出相关信息*/
}
```

本节例 3-1 程序中，x<0 是判断 x 的值是否小于 0，是关系运算；例 3-2 程序中，a+b>c && a+c>b && b+c>a 是判断输入的三条边长值中任意两边之和是否大于第三边，这是逻辑运算。下节将介绍关系运算和逻辑运算。

3.2 关系运算和逻辑运算

3.2.1 关系运算

C 语言提供了 6 种关系运算符，如表 3-1 所示。

表 3-1 关系运算符

运算符	<	<=	>	>=	==	!=
含义	小于	小于等于	大于	大于等于	等于	不等于
优先级	6				7	

例如，a+b>c、a=b>c 和 a==b<c 都是关系表达式。关系表达式的值是“真”或“假”。表达式 a+b>c 中，如果 a+b 之和大于 c 值，条件成立，该表达式值为“真”；如果 a+b 之和不大于 c 值，条件不成立，该表达式值为“假”。

说明：

① 关系运算符的优先级低于算术运算符，而高于赋值运算符。

② ==是关系运算符，而=是赋值运算符。

3.2.2 逻辑运算

C 语言提供了 3 种逻辑运算符，如表 3-2 所示。

表 3-2　逻辑运算符

运算符	!	&&	‖
含义	逻辑非	逻辑与	逻辑或
优先级	2	11	12

例如：a‖b 和 a＞b&& x＞y 都是逻辑表达式。逻辑表达式的值只有两个，“真”或“假”。在 C 语言中，任何一个非零值都表示“真”，零表示“假”。

注意：

① a&&b：当 a 和 b 均为“真”时，结果为“真”；否则，结果为“假”。

② a‖b：当 a 和 b 均为“假”时，结果为“假”；否则，结果为“真”。

③ !a：当 a 为“真”时，结果为“假”；当 a 为“假”时，结果为“真”。

④ a&&b&&c，只有 a 为真(非 0)时，才需要判别 b 的值，只有 a 和 b 都为真时才需要判别 c 的值。只要 a 为假，就不必判别 b 和 c。即对于 && 运算符来说，只有 a≠0，才继续进行右面的运算。

⑤ a‖b‖c，只要 a 为真(非 0)，就不必判断 b 和 c。只有 a 为假，才判别 b。a 和 b 均为假才判别 c。即对于‖运算符，只有 a＝0，才继续进行其右面的运算。

例如：已知 a＝7、b＝5、ch＝'m'、d＝5e＋12，求逻辑表达式 a＞b&&ch＞'t'‖d 的值。表达式自左至右扫描求解。首先处理 a＞b(因为关系运算符优先于逻辑运算符 &&)结果为真，即值为 1；再进行 ch＞'t'的运算，结果为假，即值为 0；表达式变成 1&&0‖d 的运算；根据优先次序，先进行 && 运算结果为假(值为 0)，最后进行‖运算得到结果为真(值为 1)。

3.3　if　语　句

if 语句是用来判定所给定的条件是否满足，根据判定的结果来选择不同的执行语句。if 语句有 if-else、if 和 if-else if 三种形式，而且条件语句还可以嵌套。

3.3.1　if-else 形式

if-else 分支结构一般形式为：

```
if(表达式)
    语句 1;
else
    语句 2;
```

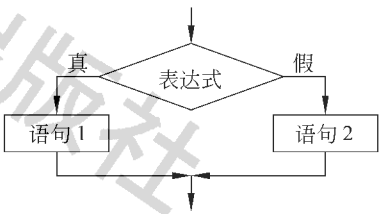

图 3-2　if-else 分支结构图

执行过程为：先计算表达式的值，若表达式结果为“真”，则执行语句 1；否则(表达式为“假”)，执行语

句 2。其执行流程如图 3-2 所示。

例 3-3 输入一个整数，判断该数是奇数还是偶数。

程序代码如下：

```
#include <stdio.h>
void main()
{  int   x;
   printf("请输入一个整数:");
   scanf("%d",&x);
   if (x%2==0)                          /* x 与 2 的余数为 0,则为偶数,否则为奇数 */
      printf("该数是偶数.\n");
   else
      printf("该数是奇数.\n");
}
```

运行结果：

```
请输入一个整数: 78
该数是偶数.
```

上例程序中，对于输入的 x 值进行判断，x 与 2 的余数如果为 0，即表达式 x%2==0 为真，则执行 printf(“该数是偶数.\n”);，否则表达式为假，执行 printf(“该数是奇数.\n”);。

例 3-4 输入两个整数，将较大的数输出。

程序代码如下：

```
#include <stdio.h>
void main()
{    int a,b;
     printf("\n Please input a,b:");
     scanf("%d,%d",&a,&b);
     if (a>b)
          printf("max=%d\n",a);
     else
          printf("max=%d\n",b);
}
```

运行结果：

```
Please input a,b: 100,99
max=100
```

练习 3-1 输入一个整数，判断其是否大于 100。

例 3-5 输入两个数，按数值由小到大的次序输出这两个数。

程序代码如下：

```
#include <stdio.h>
void main()
{    float a,b,t;
     printf("Please input a,b:");
     scanf("%f, %f",&a,&b);
     if (a>b)                          /* if 子句有多条语句,必须用大括号括起来 */
     {  t=a;                           /* a⇔b ;以下三语句实现 a 和 b 两数互换 */
        a=b;
        b=t;
     }
     else
     ;
     printf("从小到大的次序:%5.2f, %5.2f\n",a,b);
}
```

运行结果：

```
Please input a,b:99.9,10.5
从小到大的次序: 10.50,99.90
```

注意：

① if 子句(图 3-2 中的语句 1)语法上要求只能是一条语句。如果是多条语句，就要用大括号括起来。用大括号括起来的多条语句称为复合语句，复合语句在语法上当作一条语句。

② else 子句语法上要求也只能是一条语句，如果是多条语句，必须用括号括起来形成复合语句。

③ if-else 一般形式中，if 条件表达式及 else 后如果直接出现分号，表示执行的是空语句。

练习 3-2 将输入的英文字母大小写互换，即大写字母转换成小写，小写字母转换成大写。

3.3.2 if 形式

if 分支结构的一般形式为：

```
if(表达式)
  语句 1;
```

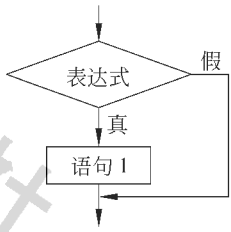

图 3-3 if 分支结构图

该语句执行过程为：先计算表达式的值，若表达式结果为“真”，则执行语句 1；否则什么都不做，跳过语句 1。其流程如图 3-3 所示。

例 3-6 输入一个字符，判断该字符是否为英文字母。

分析：输入的字符 ch 如果在 a～z 之间，或者 A～Z 之间，则

为英文字母。

程序代码如下：

```
#include <stdio.h>
void main()
{    char  ch;
     printf("\n Please input ch :");
     scanf("%c",&ch);
     if (ch>='a'&&ch<='z'‖ch>='A'&&ch<='Z')
          printf("Yes!\n");
}
```

运行结果：

```
Please input ch :m
Yes!
```

程序中表达式 ch>='a'&&ch<='z' ‖ ch>='A'&&ch<='Z'是判断字符 ch 是否为英文小写或大写字母，它等价于 ch>=97&&ch<=122 ‖ ch>=65&&ch<=90(参见附录 A 的 ASCII 码表)。

例 3-7 若输入一个整数是非零数，则显示“OK!”，否则什么也不显示。

程序代码如下：

```
#include <stdio.h>
void main()
{    int a;
     printf("\n Please input a :");
     scanf("%d",&a);
     if (a)                              /*表达式 a 的值为非零,结果为真*/
         printf("OK!");
}
```

运行结果：

```
Please input a: 99
OK!
```

程序中首先判断 if 括号中的表达式，表达式为 a。如果 a 为非零数，则表达式就为真，因而执行 printf("OK!");。

练习 3-3 判断输入的数据是否介于 0 和 100 之间。如果该数不在此范围内，输出“Error score!”。如果该数在此范围内，不输出。

3.3.3 if 语句的嵌套

在 if-else 分支语句中还包括另外的 if 语句，则称为嵌套的 if 语句。其结构形式为：

```
if(表达式 1)
    if(表达式 2)语句 1;
    else  语句 2;          } 内嵌 if 语句
else
    if(表达式 3)语句 3;
    else  语句 4;          } 内嵌 if 语句
```

例 3-8　任意输入三个整数，求其中最大的数。

分析：先将 x 和 y 进行比较，得到 x，y 之中的较大数，再将其与 z 比较，得到三个数中的最大数。流程图如图 3-4 所示。

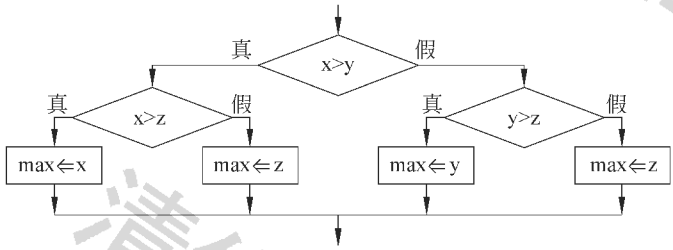

图 3-4　例 3-8 流程图

程序代码如下：

```
#include <stdio.h>
void main()
{    int x,y,z,max;
     printf("\n Please input x,y,z :");
     scanf("%d  %d  %d",&x,&y,&z);
     if ( x >y )
     {   if (  x>z )
                max=x;          } 内嵌 if 语句
         else  max=z;
     }
     else
     {   if ( y>z )
                max=y;          } 内嵌 if 语句
         else max=z;
     }
     printf("\n max=%d",max);
}
```

运行结果：

```
Please input x,y,z: 100 5 29
max=100
```

例 3-9　写程序，输入某年的年份，判断此年是否是闰年。判断闰年的方法是：能被 4

整除，但不能被 100 整除的年份是闰年；能被 100 整除，又能被 400 整除的年份是闰年。不符合以上这两个条件的年份不是闰年。

分析：用变量 flag 作为标记，如果判断输入的年份是闰年，则 flag＝1，非闰年 flag＝0。最后根据 flag 的值来输出是否是闰年的信息。

程序代码如下：

```
#include <stdio.h>
void main()
{    int year,flag;
     printf("\n 请输入年份:");
     scanf("%d",&year);
     if (year%4==0)
            if (year %100 !=0)
                   flag=1;
            else
                   if (year %400 !=0)
                          flag=0;
                   else
                          flag=1;
     else
         flag=0;
     if(flag)                              /*表达式等同于 flag==1*/
         printf("%d 年是闰年.\n",year);
     else
         printf("%d 年不是闰年.\n",year);
}
```

运行结果：

```
请输入年份：2008
2008 年是闰年.
```

要说明的是，本例中 flag 值只有两个：1 或 0，常常被称为开关变量，用来标记两种状态。在程序设计中经常用到此方法。

例 3-10 有一个函数，定义如下：$y=f(x)=\begin{cases}0 & (x<0)\\ x & (0\leqslant x\leqslant 10)\\ x^2 & (x>10)\end{cases}$

分析：此例比例 3-1 稍复杂一些，在第一个表达式为假时，要进行进一步判断。

程序代码如下：

```
#include <stdio.h>
void main()
{    float  x,y;
     scanf("%f",&x);
```

```
    if (x<0)
        y=0;
    else
        if (x<=10)
            y=x;
        else  y=x*x;
        printf("x=%f, y=%f",x,y);
}
```

注意：

① else 子句不能作为语句单独使用，它必须与 if 配对使用。

② 为使程序结构清晰、层次分明，常常采用程序行缩进的书写格式，if 和其对应的 else 写在一列上。但是，书写格式不能代替逻辑结构。

③ if 和 else 的配对关系。一个 else 总是与其上面距它最近的，并且没有其他 else 与其配对的 if 相配对。

例如有以下形式的 if 语句结构：

```
if(表达式 1)
    if(表达式 2)语句 1;
else
    if(表达式 3)语句 3;
    else 语句 4;
```

虽然第一个 else 与第一个 if 在书写格式上对齐，但实际上它与第二个 if 配对，因为它们相距最近，而且第二个 if 还没有配对。

如果要使第一个 else 与第一个 if 对应，方法一：第二个 if 结构加花括号；方法二：第二个 if 结构增加 else 部分，这样 if 和 else 的数目相同，一一对应，不易出错。

```
if(表达式 1)
    { if (表达式 2) 语句 1;}
else
    if(表达式 3)语句 3;
    else 语句 4;
```

```
if(表达式 1)
    if(表达式 2)语句 1;
    else;
else
    if(表达式 3)语句 3;
    else 语句 4;
```

练习 3-4 输入两个整数，分别存放在变量 x 和 y 中，若两个数不等，则输出其中较大的数；如相等，则输出“x=y”和 x 的值。

3.3.4 if-else if 形式

if-else if 分支结构一般形式为：

```
if(表达式 1)语句 1;
else if(表达式 2)语句 2;
```

```
⋮
else if(表达式 n-1)语句 n-1;
else 语句 n;
```

执行过程是：先判断表达式 1 如果为“真”，则执行语句 1，然后退出该 if 结构；否则（表达式 1 为“假”）判断表达式 2，若成立，则执行语句 2，然后退出该 if 结构；依次类推，其执行过程如图 3-5 所示。

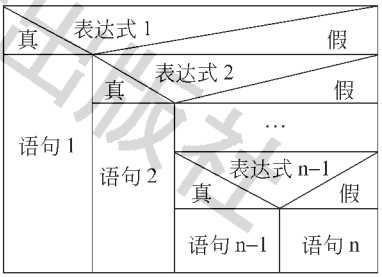

图 3-5　if-else if 分支结构图

下面把例 3-10 程序中部分语句改写如下：

```
if (x<0)
    y=0;
else if (x<=10)
        y=x;
     else y=x * x;
```

其中，把 else 子句中包含的 if 语句写在其同一行，形成 else if 结构。

例 3-11　输入三角形的三条边长，判断它们能否构成三角形。若能，再判断是何种三角形（等腰三角形、等边三角形、一般三角形）。

分析：首先按照例 3-2 的方法判断三条边能否构成三角形，如果成立，继续判断是否是等边三角形，若不成立，再判断是否是等腰三角形，否则为一般三角形。

程序代码如下：

```
#include <stdio.h>
void main()
{ float a,b,c;
  printf("请输入三角形的三条边 a,b,c:");
  scanf("%f,%f,%f",&a,&b,&c);
  if(a+b>c && a+c>b && b+c>a)
  {
     if(a==b&&b==c&&a==c)
          printf("能构成等边三角形.\n")
     else  if(a==b || b==c || a==c)
          printf("能构成等腰三角形.\n");
     else
          printf("能构成一般三角形.\n");
  }
  else
      printf("此三条边不能构成三角形.\n");
}
```

练习 3-5　在例 3-11 程序中进一步求出各种类型三角形的面积。

例 3-12　学生成绩分 A、B、C、D、E 五等。输入一个百分制成绩，判断它属于哪一等，其中 90～100 分为 A，80～89 分为 B，70～79 分为 C，60～69 分为 D，0～59 分为 E，其他数据显示出错信息。

程序代码如下：

```
#include <stdio.h>
void main()
{    float score;
     printf("\nEnter a score :");
     scanf("%f",&score);
     if (score>100) printf("Error Data!");
     else  if (score>=90) printf("A \n");
        else  if (score>=80) printf("B \n");
             else  if (score>=70) printf("C \n");
                  else  if (score>=60) printf("D \n");
                       else  if (score>=0) printf("E \n");
                            else printf("Error Data!");
}
```

3.3.5 条件运算符及条件表达式

如果 if 语句中的 if 子句和 else 子句都执行一个赋值语句且向同一个变量赋值时，可以用一个条件运算符来处理。如以下 if 语句：

```
if (x>y)
    max=x;
else
    max=y;
```

可以用“max=(x>y)? x:y;”来代替。

条件表达式的一般形式为：

表达式 1? 表达式 2: 表达式 3

说明：

① 条件运算符由?和:组成，是 C 语言中唯一的三目运算符，要求有 3 个操作对象，运算级为 13。

② 条件运算符的执行顺序：先求表达式 1，若为真(非 0)，以表达式 2 的值作为整个条件表达式的值，否则，以表达式 3 的值作为整个条件表达式的值。

③ 条件运算符优先于赋值运算符，因此上面赋值表达式的求解过程是先求解条件表达式，再将它的值赋给 max。

3.4 switch 语句

开关分支 switch 语句是分支结构的另一种形式，该语句执行时它根据条件表达式的取值来选择程序中的一个分支。switch 分支语句的一般形式为：

```
switch(表达式 e)
{  case  常量表达式 c1: 语句 1;[break;]
   case  常量表达式 c2: 语句 2;[break;]
   ⋮
   case  常量表达式 cn: 语句 n;[break;]
   [default: ]       语句 n+1;[break;]
}
```

switch 语句的执行过程如图 3-6 所示。switch 首先求表达式 e 的值，然后判断 e 值与 c1、c2、…、cn 中哪个值相等，若与其中某个值相等，则执行其后的相应的语句；若不与任何一个常量表达式值相等，则执行 default 后面的语句。当执行某一分支中的语句后，遇 break 语句则退出 switch 语句结构。

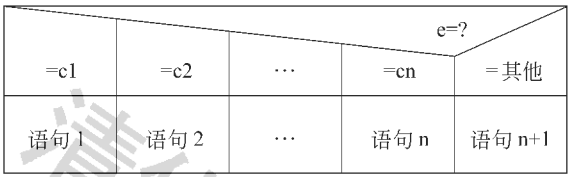

图 3-6　switch 分支结构图

例 3-13　观察程序执行过程。

```
#include <stdio.h>
void main()
{    int  x;
     scanf("%d",&x);
     switch(x)
     {    case  1:  printf("good  morning!\n");
          case  2:  printf("good  afternoon!\n");  break;
          case  3:
          case  4: printf("good  night!\n");  break;
          default: printf("wrong!\n");
     }
}
```

说明：

① switch 后面括号内的表达式可以是整型、字符型或枚举类型。

② 当表达式的值与某一个 case 后面的常量表达式值相等时，就执行此 case 后面的语句；若所有 case 后面的值没有与之相匹配的，就执行 default 后面的语句。如若输入的 x 值非 1、2、3 或 4，则执行 default 后面的语句。

③ 各个 case 的出现次序不影响执行结果。

④ 多个 case 可以共用一组执行语句，例如，x 的值为 3 或 4 时都执行同一组语句。

⑤ break 语句使控制退出 switch 结构。若没有 break 语句，则程序继续执行下一个 case 的语句组。如 x=1，则执行 case 1 后面的语句组之后，继续执行 case 2 后面的语句

组，直到遇到 break 语句终止 switch 语句的执行。

例 3-14 重新使用 switch 语句完成上节中的例 3-12。

程序代码如下：

```
#include <stdio.h>
void main()
{    int e,grade;
     printf("input grade(0-100): ");
     scanf("%d",&grade);
     if(grade>100 || grade<0)
        printf("Error!\n");
     else
     {     e=grade/10;                    /*将百分制分数值转换成 0~10 的整数*/
           switch ( e )
           {   case 6: printf("grade D\n"); break;
               case 7: printf("grade C\n"); break;
               case 8: printf("grade B\n"); break;
               case 9: printf("grade A\n"); break;     /*可以省略此组语句*/
               case 10: printf("grade A\n"); break;
               default: printf("grade E\n"); break;
           }
     }
}
```

例 3-15 编写一个实现两个操作数四则运算的程序。

分析：两个操作数为 a 和 b，四则运算符为 op，它表示＋、－、＊和/，运算结果为 c。

程序代码如下：

```
#include <stdio.h>
void main()
{    int  a,b,c;
     char   op;
     printf("请输入操作数和操作符:");
     scanf("%d%c%d", &a, &op, &b);
     switch( op )
     {    case '+':    c=a+b;  printf("%d  %c  %d=%d\n",a, op, b,c); break;
          case '-':    c=a-b;  printf("%d  %c  %d=%d\n",a, op, b,c);break;
          case '*':   c=a*b;  printf("%d  %c  %d=%d\n",a, op, b,c);break;
          case '/':    if (b!=0) {c=a/b;printf("%d  %c  %d=%d\n",a, op, b,c);}
            break;
          default:    printf("操作符错误!\n"); break;
     }
}
```

例 3-16 用菜单进行四则运算。

程序代码如下：

```
#include <stdio.h>
#include <conio.h>
void main()
{    int  a,b,c;
     char op;
     scanf( "%d,%d", &a, &b);
     printf( "****************************************\n" );
     printf( "*          请输入选项代码(0~4)          *\n" );
     printf( "*                  1——加法              *\n" );
     printf( "*                  2——减法              *\n" );
     printf( "*                  3——乘法              *\n" );
     printf( "*                  4——除法              *\n" );
     printf( "*                  0——退出              *\n" );
     printf( "****************************************\n" );
     op=getch();
     switch( op )
     {    case '1': c=a+b; printf("%d  +  %d=%d\n",a, b,c);break;
          case '2': c=a-b;  printf("%d  -  %d=%d\n",a, b,c);break;
          case '3': c=a*b;  printf("%d  *  %d=%d\n",a, b,c);break;
          case '4': if (b!=0){ c=a/b; printf("%d  /  %d=%d\n",a, b,c);} break;
          case '0': break;
          default: break;
     }
}
```

程序代码中用到了函数 getch()，它必须引入头文件 conio.h，getch()的作用是从键盘接收一个字符，而且并不把这个字符显示出来，也就是说，按一个键后它并不在屏幕上显示所按的字符，而继续运行后面的代码。

与 getch()很相似的还有 getche()函数，它也需要引入头文件 conio.h，不同之处就在于 getch()无返回显示，getche()有返回显示。即 getche()函数将读入的字符回显到显示屏幕上。

前面讲到的 getchar()函数也是从键盘上读入一个字符，回显到屏幕上，并在回车后响应。

本章小结

C 语言中，使用条件语句来实现选择分支，if 语句和 switch 语句用于实现分支结构。if 语句可以实现单分支、双分支和多分支选择，switch 语句可以比 if 语句更加简易地实现多路分支。

习 题 3

1. 判断学生成绩是否合格，如果合格，输出 pass；不合格，输出 failure。
2. 编写一个程序，若用户输入的一个整数是 4 的倍数，则显示 OK。
3. 输入三个整数 a、b、c，按从小到大的顺序输出。
4. 从键盘输入字符，判断字符的类型（数字、大写字母、小写字母或其他类型）。
5. 编程实现：输入一个整数，判断它能否被 3、5 和 7 整除，并输出以下信息之一：

① 能同时被 3,5,7 整除

② 能被其中两数（要指出哪两个）整除

③ 能被其中一个数（要指出哪一个）整除

④ 不能被 3,5,7 任一个整除

6. 编写一个课表查询程序，输入一周中的星期值，能够显示出当天的课程。
7. 某超市商品打折促销。假定购买某商品的数量为 x 件，折扣情况如下：

$$\begin{cases} x<5 & \text{不折扣} \\ 5\leqslant x<10 & 1\%\text{折扣} \\ 10\leqslant x<20 & 2\%\text{折扣} \\ 20\leqslant x<30 & 4\%\text{折扣} \\ x\geqslant 30 & 6\%\text{折扣} \end{cases}$$

该商品原价格由键盘输入，编程计算购买 x 件商品应付总金额。要求分别用 if 语句和 switch 语句编程实现。

8. 以下程序的输出结果是_____。

```
#include <stdio.h>
void main()
{  int x,a=1,b=3,c=5,d=4;
   if(a<b)
   if(c<d)  x=1;
   else
        if(a<c)
         if(b<d)    x=2;
         else       x=3;
        else  x=6;
   else x=7;
printf("x=%d\n",x);
}
```

9. 若有

```
int x=10,y=20,z=30;
```

以下语句执行后 x,y,z 的值是_____。

```
if (x>y)
z=x;x=y;y=z;
```

A) x=10,y=20,z=30　　B) x=20,y=30,z=30

C) x=20,y=30,z=10　　D) x=20,y=30,z=20

10. 以下程序段的输出结果是_____。

```
#include <stdio.h>
void main()
{  int a=0,b=0,c=0;
   if(a=b+c) printf("***\n");
   else    printf("$$$\n");
}
```

A) 有语法错误不能通过编译　　B) 可以通过编译但不能通过连接

C) * * *　　D) $ $ $

11. 若有：

```
int a=1,b=2,c=3,d=4,m=2,n=2;
```

则执行

```
(m=a>b)&&(n=c>d)
```

后 n 的值是_____。

A) 1　　B) 2　　C) 3　　D) 0

12. 以下程序的输出结果是_____。

```
#include <stdio.h>
void main()
{  int x=1,y=0;
   switch(x)
     {  case  1:   switch(y)
                   {case  0:  printf("first\n");  break;
                    case  1:  printf("second\n");break;
                   }      /*若在此加语句 break;将会怎样?*/
        case  2: printf("third\n");
     }
}
```

第4章 循环结构程序设计

本章主要内容：

- while 语句；
- do-while 语句；
- for 语句；
- 循环嵌套；
- break 和 continue 语句。

4.1 引 例

例 4-1 输入三角形的三条边长，求三角形面积。

分析：此例在上章中已经做过。为了使其一次运行，可测试多组三角形三边，将其用循环结构实现。当三角形三边均为 0 时，结束循环。

程序代码如下：

```
#include <stdio.h>
#include <math.h>
void main()
{ float a,b,c,s,area;
  while(1)
{
    printf("请输入三角形的三条边 a,b,c:");
    scanf("%f,%f,%f",&a,&b,&c);
    if(a==0 && b==0 && c==0)break;
    if(a+b>c && a+c>b && b+c>a)
   {
      s=1.0/2*(a+b+c);
      area=sqrt(s*(s-a)*(s-b)*(s-c));
      printf("a=%7.2f  b=%7.2f c=%7.2f\n",a,b,c);
      printf("s=%7.2f area=%7.4f\n",s,area);
    }
```

```
    else
      printf("此三条边不能构成三角形!\n");
  }
}
```

例 4-2 一行打印 60 个 * 。

程序代码如下：

```
#include <stdio.h>
void main()
{   int i;
    i=1;
    while(i<=60)          /* 重复输出 * 60 次——循环 */
    {
        printf("*");
        i=i+1;
    }
    printf("\n");
}
```

如果一行打印 5 个 * ，可用 printf("*****\n")；实现，但本例要求打印 60 个 * ，在字符串中敲 60 下 * ，并不可取。本例用短短几行代码完成了一项重复输出 * 60 次的工作，解决了一行输出 60 个 * 的问题。

这种有规律的、重复性的工作在程序设计中称为**循环**。循环结构是结构化程序设计的三种基本结构之一，是学习程序设计语言的基础，在本课程中占有重要地位。

例 4-1 使用了 while 语句进行循环处理。在 C 语言中实现循环的语句除了 while 外，还有 do-while、for 以及用 goto 语句和 if 语句联合构成的循环语句等几种形式。

需要说明的是，不论采用哪种语句实现循环，循环都是由**循环条件**和**循环体**两部分构成的。循环条件用于确定循环什么时候开始，什么时候结束，如本例中的 i<=60；而循环体则负责完成那件有规律的、重复性的工作，在本例中就是一对花括号包含的部分，主要完成了输出一个 * 的工作。当循环体的工作在循环条件的控制下重复执行直至结束时，程序就完成了一个循环的工作。在循环条件中用于控制判断的变量叫**循环控制变量**。

本章后面各节将针对各种循环语句的使用方法分别进行介绍。

4.2 while 语句

while 语句属于当型循环，先判断，后执行。while 语句的一般形式为：

```
while(表达式)
  循环体语句;
```

其程序流程如图 4-1 所示。

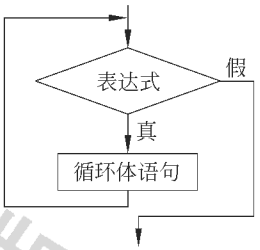

图 4-1　while 循环流程图

while 语句的执行过程是：当表达式的结果值为真时，执行循环体语句；当表达式的结果值为假时，循环体语句不执行，终止循环。

关于 while 语句，有如下几点说明：

① 循环体语句可以是单个语句、空语句(;)，也可以是一个复合语句（程序块）。

② while 语句的作用范围：循环体如果包含一个以上的语句，应该用花括号括起来作为复合语句，否则 while 循环体的作用范围只到 while 后面的第一个分号处。例如：

```
while(a>1);
{  a++;
}
```

复合语句“{a＋＋;}”不是循环体，“while(a＞1);”后的分号所代表的空语句才是这里的循环体内嵌语句。

③ 表达式为循环条件，可以是任意类型。当表达式为非 0 值时，表示条件为真，则循环可以进行；相反，表达式值为 0 时，表示条件为假，则循环终止。如果在执行过程中，第一次条件判断表达式值就为假，则循环体将一次也不执行。

④ 当循环条件表达式为永真时，例如 while(1)，循环永远无法结束，称为**“死循环”**，在程序设计过程中应尽量避免出现这种情况。

例 4-3　求 $\sum_{i=1}^{100} i = 1+2+3+\cdots+100$ 的值。

分析：此题是循环次数固定的实例，本题是一道典型的累加求和的题目，采用**递推算法**。数列求和须观察每个数列项的变化规律。本题各数之间相差 1，依次将每个数加入和值中，加 100 次，和值初始应为 0。数列项变量初始为 1，每次加 1，到 100 为止。

设 i 为数列项变量和循环控制变量，sum 为累加和变量，解题步骤如下：

第 1 步：设置循环初值：sum＝0；i＝1；

第 2 步：若 i ＜＝100，执行第 3 步；否则，结束循环，执行第 5 步。

第 3 步：sum＋i⇒sum，累加求和。

第 4 步：i＋1⇒i，改变循环控制变量 i 的值，使循环条件趋于结束的语句。返回第 2 步(其最终 i＞100，使循环结束)。

第 5 步：执行循环后面的语句，即输出结果。

程序代码如下：

```
#include <stdio.h>
void main()
{    int i=1,sum=0;                    /*设循环初值*/
     while(i<=100)                     /*循环条件判断*/
```

```
    {    sum=sum+i;                    /*循环主体:累加求和*/
         i++;                          /*修改循环控制变量*/
    }
    printf("sum=%d\n",sum);            /*输出结果*/
}
```

运行结果：

```
sum=5050
```

练习 4-1 求 1+3+5+…+99(100 以内奇数和)。

练习 4-2 计算 $1^2+2^2+3^2+\cdots+10^2$。

例 4-4 求 300～800 之间 7 的倍数之和。

分析：上述的练习题目均采用递推算法，本例也可以采用递推算法求解，即将循环初值 i 的值改为大于 300 的第一个被 7 整除的数 301，这样计算初值显然不方便。因此，本例采用另一种**穷举法**求解，即在给定范围内，列举出每一种可能性，然后按照特定的条件进行筛选。在 300～800 之间对每一个数判断其是否是 7 的倍数，如果是，就将该数加入累加器中；不是，就不加入。

程序代码如下：

```
#include <stdio.h>
void main()
{    int i=300,sum=0;                  /*设循环控制变量初值为 300*/
     while(i<=800)                     /*循环条件:小于 800*/
     {    if (i%7==0)                  /*寻找 7 的倍数,进行筛选*/
              sum=sum+i;               /*循环主体:累加求和*/
          i=i+1;                       /*修改循环变量*/
     }
     printf("sum=%d\n",sum);           /*输出结果*/
}
```

运行结果：

```
sum=39564
```

练习 4-3 键盘输入 10 个整数，找出其中的最大值。

例 4-5 依次输入一批正数，并求所有输入的正数之和，当输入负数或 0 时结束。

分析：此题是循环次数不固定的实例。

读入一个整数，判断其值是否大于 0。若大于 0，则累加之；否则，结束循环，打印结果。

设读入值为 x，和值为 sum，其初值为 0 。

程序代码如下：

```
#include <stdio.h>
void main()
{
    float x,sum;
```

```
    sum=0.0;
    scanf("%f", &x);                /*循环初值*/
    while (x>0.0)                   /*循环条件*/
    {   sum=sum+x;                  /*循环主体:累加求和*/
        scanf("%f", &x);            /*改变循环条件变量的语句,再次读入一个新的x值*/
    }
    printf("sum=%f\n",sum);         /*输出结果*/
}
```

运行情况：

```
1 2 3 4 5 -1↙
sum=15.000000
```

注意：运行循环程序，并不是输入一个数得一个结果，再输入一个数再得一个结果，而是连续输入一串数及该批数的结束值(本例中应是一个负数或0)，每个数以空格或回车间隔，最后输入回车键，才能得到运行结果。

4.3 do-while 语句

do-while 语句属于直到型循环，先执行，后判断。其一般形式如下：

```
do
{
    循环体语句;
} while(表达式);
```

其程序流程如图 4-2 所示。

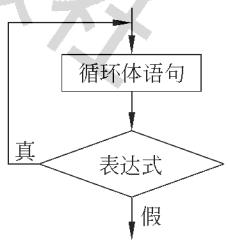

图 4-2　do-while 语句流程图

do-while 语句的执行过程是：先执行循环体语句，后判断循环条件，如果表达式的结果值为真时，继续执行循环体语句，表达式的结果值为假时，结束循环。如果条件表达式的初值为假，循环体语句至少执行一次。

关于 do-while 语句，有如下几点说明：

① do-while 语句作为一个整体，while 的表达式后必须加分号(;)。

② do-while 循环语句中，不管循环体是否为单一语句，都用花括号把它括起来，并把 while()直接写在}的后面，以免把 while()部分误认为另一个新的 while 循环的开始。

下面仍以例 4-4 为例，介绍 do while 语句的使用。

例 4-6　依次输入一批正数，求这批正数之和，当输入负数或 0 时结束。用 do wile 语句完成。

程序代码如下：

```
#include <stdio.h>
```

```
void main()
{   float x,sum;
    sum=0.0;                        /*循环初值*/
    do
    {    scanf("%f", &x);           /*改变循环条件变量的语句*/
         sum=sum+x;                 /*累加求和——累加器*/
     }while (x>0.0);                /*循环条件*/
    printf("sum=%f\n",sum);         /*输出结果*/
}
```

运行情况

```
1 2 3 4 5 -1↙
sum=14.000000
```

值得注意的是，从运行结果看与用 while 循环完成的程序得到的结果不一致，原因是读入的最后一个负数也加入和值 sum 了，因此在输出结果处，需输出 sum-x 的值，而不是 sum 的值，即把多加的最后一个数再减去，这样输出结果才正确。通过此例可以看出：解决循环问题时，有时需考虑循环初始及循环结束时的不同情况，即对循环初次或循环最后一次的计算情况，加以分析，观察运行结果是否正确，如果不正确，进行合理更改。

例 4-7 利用公式

$$\frac{\pi}{4} \approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$$

求 π 的近似值，直到最后一项的绝对值小于 10^{-6} 为止。

分析：此题主要解决数列各项的正负号问题。此数列是奇数的倒数和，且每项正负号不同，根据符号位变化，负负得正，设符号位为 s，初值＝1，每次 s＝－s 或 s＝－1＊s，改变符号位。循环控制变量 n，初值＝1，每次增 2。单个数列项 t，初值＝1，t＝s＊1/n，每次累加至和值 pi 中，pi 的初值＝0。当 t 的绝对值小于 10^{-6} 时，结束循环，fabs(t)为绝对值函数，其头文件是 math.h。其中 1/n 是分式运算，为避免整除问题，应写为 1./n 或将 n 定义为实型变量。

程序代码如下：

```
#include <stdio.h>
#include <math.h>
void main()
{    float  n, s, t, pi;
     t=1;  pi=0;  n=1.0;  s=1;  /*循环初值*/
     do
     {   pi=pi+t;               /*累加 t*/
         n=n+2;                 /*循环变量增值*/
         s=-s;                  /*求符号位 s,正负号变化*/
         t=s*1./n;              /*求一个数列项的值 t*/
     } while((fabs(t))>=1e-6);  /*fabs(t)为绝对值函数*/
     pi=pi*4;
     printf("pi=%f\n",pi);
}
```

运行结果：

```
pi=3.141594
```

练习 4-4 输入数字 n，求前 n 项的和数 $s=1+\frac{1}{2}-\frac{1}{3}+\frac{1}{4}-\cdots+\frac{1}{n}$。

4.4 for 语句

4.4.1 for 语句格式

for 语句在功能上与 while 语句相似，都属于当型循环，先判断，后执行。但 for 语句在使用上更加灵活、方便。其一般形式如下：

```
for(表达式 1;表达式 2;表达式 3)
  循环体语句;
```

其程序流程如图 4-3 所示。

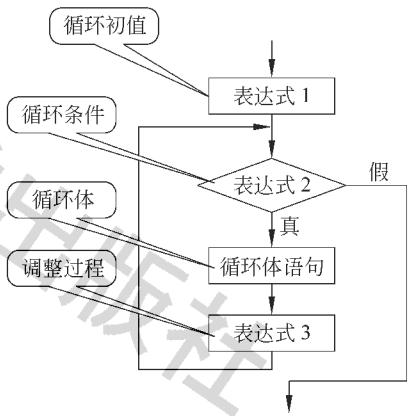

图 4-3 for 语句流程图

for 语句的执行步骤如下：

第 1 步：计算表达式 1(循环初值)。

第 2 步：计算表达式 2(循环条件)，判断其值，若为真(非 0)，则执行循环体语句，即第 3 步；若为假(0)，则结束循环，执行循环后面的语句，即第 6 步。如果表达式 2 的初值为假，循环体语句一次也不执行。

第 3 步：执行循环体语句。

第 4 步：计算表达式 3(修改循环控制变量)。

第 5 步：转去重复执行第 2 步。

第 6 步：执行循环后面的语句。

下面仍以例 4-2 为例，介绍 for 语句的使用。

例 4-8 用 for 循环完成求 $\sum_{i=1}^{100} i = 1+2+3+\cdots+100$ 的值。

程序代码如下：

方法一：和值初值在循环前赋值

```
#include <stdio.h>
void main( )
{  int i, sum;
   sum=0;
   for (i=1; i<=100; i++)
     sum=sum+i;
   printf("sum=%d \n",sum);
}
```

方法二：循环初值采用逗号表达式

```
#include <stdio.h>
void main( )
{  int i, sum;
   for (sum=0, i=1; i<=100; i++)
     sum=sum+i;
   printf("sum=%d\n",sum);
}
```

可以看出，此题采用 for 语句完成，更加简洁方便。

关于 for 语句，有如下几点说明：

(1) 循环体语句可以是单个语句或空语句(;)，也可以是一个复合语句（程序块）。

(2) for 语句的作用范围：循环体如果包含一个以上的语句，应该用花括号括起来(作为复合语句)，否则 for 循环体的作用范围只到 for 后面的第一个分号处。

(3) 三个表达式可以是任意类型，三个表达式均可省略，但其中的分号“；”不能省略。

① 表达式 1：用于循环变量赋初值。进入 for 语句首先执行而且仅执行一次表达式 1，表达式 1 一般为赋值表达式或逗号表达式；表达式 1 可以放在 for 循环之前，即表达式 1 可以为空。上例可写为：

```
sum=0;i=1;
for(;i<=100;i++)sum=sum+i;
```

② 表达式 2：用于循环条件判断，当表达式 2 的结果值为真时，执行循环体语句；当表达式 2 的结果值为假时，循环体语句不执行，结束循环。

③ 表达式 3：用于改变循环变量的值。循环体语句每次执行完后要继续执行表达式 3，表达式 3 可以放入循环体内最后一句，因此表达式 3 可以为空，此时在循环体内应另外设法使循环正常结束。上例可写为：

```
for(sum=0,i=1;i<=100;)
    {  sum=sum+i;
       i++;
    }
```

(4) for 语句与 while、do while 语句的用法在本质上是相同的。如将 for 语句改写成如下左侧的形式，和 whlie 语句相对照，二者的一致性更加明显。

for 语句格式：

```
赋初值;
for(;循环条件;)
{  循环体语句;
   修改循环变量语句;
}
```

while 语句格式：

```
赋初值;
while(循环条件)
{  循环体语句;
   修改循环变量语句;
}
```

4.4.2 for 语句实例

例 4-9 打印 Fibonacci 数列：1,1,2,3,5,8,…的前 20 个数，并按每行打印 5 个数的格式输出。

分析：Fibonacci 数列问题起源于一个古典的有关兔子繁殖的问题。假设在第 1 个月时有一对小兔子，第 2 个月时成为大兔子，第 3 个月时成为老兔子，并生出一对小兔子(一对老，一对小)。第 4 个月时老兔子又生出一对小兔子，上个月的小兔子变成大兔子

(一对老,一对大,一对小)。第5个月时上个月的大兔子成为老兔子,上个月的小兔子变成大兔子,两对老兔子生出两对小兔子(两对老,一对中,两对小)……这样,各月的兔子对数为:1,1,2,3,5,8,…。

此题是循环次数固定的实例。数列单项变化规律为:$F_1=1$,$F_2=1$,$F_n=F_{n-1}+F_{n-2}$,($n\geqslant 3$)。每行打印5个数,即每打印完5个数,输出一个换行符,用if语句实现。设n为循环控制变量,Fibonacci数列前2个数列值为1,则n从3到20的变化,F_n用f_3表示,按数列变化规律,每次等于前两项之和,即$f_3=f_1+f_2$。采用迭代法解题,每次迭代改变f_1、f_2的值,即$f_1=f_2$,$f_2=f_3$,$f_3=f_1+f_2$。

程序代码如下:

```
#include <stdio.h>
void main()
{
   int n=3, f1=1,f2=1,f3;  /*定义 Fibonacci 数列前 2 个数列值赋初值 1,n 从 3 开始*/
   printf("%14d%14d",f1,f2); /*输出前两个数列值,每个值占 14 位*/
   for(n=3;n<=20;n++)        /*n 从 3 到 20 的变化*/
    {    f3=f1+f2;           /*按照 Fibonacci 数列的规则,得到下一个数列值 f3*/
         f1=f2;
         f2=f3;              /*迭代 f1、f2 的值*/
         printf("%14d",f3);  /*循环主体:输出一个数列值*/
         if(n%5==0)          /*每行打印 5 个数,即打印 5 个数,输出一个换行符*/
             printf("\n");
    }
}
```

运行结果:

```
  1             1             2             3             5
  8            13            21            34            55
 89           144           233           377           610
987          1597          2584          4181          6765
```

例4-10 计算1!+2!+3!+…+n!,其中n=1,2,…,20。

分析:此题为连乘积及连加和运算,设n为数列项变量和循环控制变量,n的变化范围为1～20,s为累加和变量,初值为0。t为阶乘变量,初值为1。s、t应为实型。

程序代码如下:

```
#include <stdio.h>
void main()
{
    float t,s;
    int n;
    for(s=0,n=1,t=1;n<=20;n++)     /*n 从 1 到 20 的变化*/
    {
```

```
            t=t*n;                    /*计算 n!, 累乘积 t*/
            s=s+t;                    /*计算 n!的累加和 s*/
        }
    printf("1!+2!+3!+...+n!=%e\n",s);
}
```

运行结果：

```
1!+2!+3!+...+n!=2.561327e+018
```

练习 4-5 计算 $s=m!+n!$，m 和 n 需键盘输入其值。

4.4.3 三种循环语句的比较

上述 while、do while 和 for 三种循环语句的使用在本质上都是一致的，都遵循当循环条件表达式结果为真时执行循环体，为假时结束循环的规律。C 语言的三种循环语句可以用来处理同一问题，一般情况下可以互换。但是功能和灵活程度不同。for 语句功能最强，最灵活，使用最多，任何循环都可以用 for 实现；其次是 while，do-while 用得最少。

for 循环与 while 循环属于当型循环结构，先判断循环条件，后执行循环体语句；do-while 循环属于直到型循环结构，先执行一次循环体语句，然后才判断循环条件。因此，当循环体语句有可能一次也不被执行时，必须采用当型循环；当循环体语句至少执行一次时，既可以采用直到型循环，也可以采用当型循环。

一般情况下，循环次数固定的场合，例如数列求和、求积运算，采用 for 语句完成更为简洁方便，循环次数不固定的场合，采用 while 循环更为方便。

while 和 do-while 的循环变量初始化是在循环语句之前完成的，而 for 语句循环变量的初值是在 for 中的表达式 1 中实现的。

for 循环语句中的第一个和第三个表达式可以是逗号表达式，它扩充了 for 语句的作用范围，使它有可能同时对若干参数（如循环变量、重复计算数等）进行初始化和修整等。

4.5 循环嵌套

在循环体内部的语句，可以是任意执行语句，当然也可以是一个循环语句。一个循环体内又包含另一个完整的循环结构，称为**循环的嵌套**，又叫**多重循环**。

循环嵌套的原则：循环相互嵌套时，被嵌套的一定是一个完整的循环结构，即两个循环结构不能相互交叉。

例：

```
while()          for(;;)          for(;;)
{                {                {
    while()          for (;;)         while()   …
    {  }             {  }             {  }
}                }                }
```

注意：为使程序结构清晰，在书写嵌套程序时，尤其应采用逐层缩进的方式，以增加程序的可读性。

关于嵌套循环，有如下几点说明：

① 内循环必须完全嵌套在外循环内，不得互相交叉。

② 嵌套循环的循环控制变量不可同名，并列循环的循环控制变量可以同名。

③ 嵌套循环从外循环开始执行，也从外循环结束。设外循环执行 m 次，内循环执行 n 次，则每执行一次外循环，就会执行内循环 n 次，因此整个循环执行次数＝外循环次数×内循环次数＝m×n 次。

例 4-11 打印九九乘法表。

程序代码如下：

```
#include <stdio.h>
void main()
{
  int i,j;
  for(i=1;i<=9;i++)                          /*外循环打印 1~9 行*/
  {
    for(j=1;j<=i;j++)
        printf("%d*%d=%2d",i,j,i*j);         /*内循环打印在一行,打印 1~i 列*/
    printf("\n");                            /*退出内循环,打印换行符*/
  }
}
```

运行结果：

```
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12  4*4=16
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

例 4-12 中国古代数学家张丘建在《算经》里曾提出一个世界数学史上有名的百鸡问题："鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问鸡翁、母、雏各几何？"

分析：根据题意，公鸡 1 只 5 元，母鸡 1 只 3 元，小鸡 3 只 1 元。设公鸡 x 只、母鸡 y 只、小鸡 z 只，建立如下方程组：

$$\begin{cases} x+y+z=100 & \text{（百鸡）} \\ 5x+3y+z/3=100 \Rightarrow 15x+9y+z=300 & \text{（百钱）} \end{cases}$$

其中，x、y、z 只能为整数。

采用穷举法解题，买100只鸡，公鸡x可以从0到100，母鸡y可以从0到100，小鸡z可以从0到100，列举各种情况测试有无满足联立方程式条件的x、y、z，用三重循环解决。

程序代码如下：

```
#include <stdio.h>
void main()              /*法一*/
{
int x,y,z;
  for (x=0;x<=100;x++)
     for (y=0;y<=100;y++)
        for (z=0;z<=100;z++)
           if (x+y+z==100&&15*x+9*y+z==300)
            printf("%d %d %d\n",x,y,z);
}
```

运行结果：

```
0    25    75
4    18    78
8    11    81
12    4    84
```

按此算法，内循环运行次数：10^6。

本题还可进一步简化：已知公鸡1只5元，百元最多可买20只，公鸡x可以从0到20，母鸡1只3元，百元最多可买33只，母鸡y可以从0到33，而小鸡可买z只，根据百鸡公式$x+y+z=100$，当x、y的值通过外循环确定后，$z=100-x-y$，因此最内层循环可取消，条件式可省去$x+y+z==100$。

程序代码如下：

```
#include <stdio.h>
void main()          /*法二*/
{
  int x,y,z;
 for (x=0;x<=20;x++)
     for (y=0;y<=33;y++)
     {  z=100-x-y;
        if (15*x+9*y+z==300)
             printf("%d %d %d\n",x,y,z);
     }
}
```

按此算法，内循环运行次数：660。

通过本题，可得到如下结论：应尽量减少内循环次数，以减少整个循环的运行次数，提高运行速度。

练习4-6 求$i^3+j^3+k^3=3$的完全整数解。其中$-5\leqslant i\leqslant 11$，$-10\leqslant j\leqslant 9$，$-6\leqslant k\leqslant 18$。

4.6 break 和 continue 语句

C 语言的三种循环语句，都是根据循环条件表达式值的真假来控制循环结束，循环在条件表达式的值为假时结束是正常的循环结束。如果要在循环的中途非正常地退出循环，则要用 break 或 continue 语句来实现。

4.6.1 break 语句

对于 break 语句，读者并不陌生，在上一章中的 switch 语句的 case 分支中，break 语句的作用是退出 switch 语句，转去执行 switch 语句后面的语句。本节中的 break 语句，则是出现在 C 语言的三种循环语句中。

break 语句的一般形式：

```
break;
```

break 语句的功能是跳出循环体，即提前结束循环，接着执行循环语句后面的语句。

注意：break 语句只能退出当前循环结构或当前 switch 结构，不能用于其他语句。并且，若 break 语句处于多重循环中，break 语句只是跳出当前层循环。

例 4-13 韩信点兵：韩信有一队兵，他想知道有多少人，便让士兵排队报数：按从 1 至 5 报数，最末一个士兵报的数为 1；按从 1 至 6 报数，最末一个士兵报的数为 5；按从 1 至 7 报数，最末一个士兵报的数为 4；最后再按从 1 至 11 报数，最末一个士兵报的数为 10。下面程序的主要功能是计算韩信至少有多少兵。

分析：设 x 为循环控制变量，题目中“按从 1 至 5 报数，最末一个士兵报的数为 1”可用条件式：x%5==1 来描述，其余条件以此类推，多个条件用“与”连接。采用穷举法，x 从 1 到任意值，一一测试，因为求至少有多少兵，即找到第一个满足条件的 x 值，中途用 break 退出循环体。

程序代码如下：

```
#include <stdio.h>
void main()
{
    int x;
    for (x=1;  ; x++)
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {   printf(" x=%d\n", x);
            break;
         }
}
```

运行结果：

```
x=2111
```

练习 4-7 爱因斯坦数学题。爱因斯坦曾出过这样一道数学题：有一条长阶梯，若每步跨 2 阶，则最后剩下 1 阶；若每步跨 3 阶，则最后剩下 2 阶；若每步跨 5 阶，则最后剩下 4 阶；若每步跨 6 阶，则最后剩下 5 阶；只有每步跨 7 阶，最后才正好 1 阶不剩。请问，这条阶梯共有多少阶？

例 4-14 判断一个整数 m 是否是素数。

分析：素数就是只能被 1 和它自身整除的数。判断一个数 m 是不是素数，是用此数之前从 $2\sim m-1$ 之间或 $2\sim\sqrt{m}$之间的所有数来试除，看其是否能被整除，如果都不能被整除，则认为该数是素数，否则不是素数。

程序流程如图 4-4 所示。

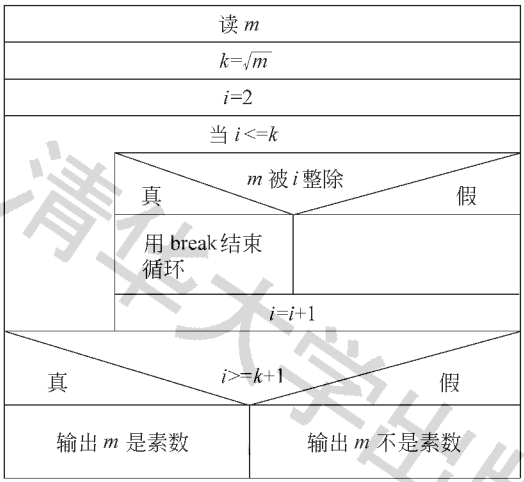

图 4-4 判断素数程序流程图

程序代码如下：

```
#include <math.h>
#include <stdio.h>
void main()
{
  int m,i,k;
  scanf("%d",&m);
  k=sqrt(m);
  for(i=2;i<=k;i++)
     if(m %i==0)break;     /* 遇到一个可整除的数,即 m 不是素数,则跳出循环体 */
  if(i>=k+1)               /* 当 i 小于 k+1,表示循环是中途退出,不是正常退出 */
     printf("%d is a prime number\n",m);
  else
     printf("%d is not a prime number\n",m);
}
```

实质上 break 语句也可使用标志变量来代替。设标志变量 flag，设初值为 1。若在循环过程中，发现了能整除 m 的数，则置 flag 为 0，表示 m 不是素数。若整个循环过程中没有能整除 m 的数，则 flag 始终为 1。把判断 flag 是否为 1 作为循环条件表达式的一部分，当 flag 为 0 时立刻中断循环。由此，可得到如下程序：

```
#include <stdio.h>
void main()
{
  int m,n,flag=1;
  scanf("%d",&m);
  /*标志变量 flag 作为循环条件表达式的一部分，当 flag 为 0 时，跳出循环体*/
  for(n=2;n<=m/2&&flag==1;n++)
     if(m%n==0) flag=0;        /*m 遇到可整除的数，置 flag=0*/
  if(flag)                     /*循环结束后，若 flag 为 1，则表示 m 为素数*/
     printf("%d is a prime number\n",m);
  else
     printf("%d is not a prime number\n",m);
}
```

4.6.2 continue 语句

continue 语句只能用于循环结构。其功能是：结束本次循环，即跳过循环体中 continue 下面尚未执行的语句，继续执行下一次的循环条件判断。

continue 语句的一般形式为：

```
continue;
```

在执行 continue 的过程中，整个循环并未结束。

例 4-15 编一个程序。求输入的 10 个数中正数的个数及平均值。

程序代码如下：

```
#include <stdio.h>
void main()
{
     int i,n=0;
     float sum=0,f;
     printf("Enter a real number:\n");
     for(i=1;i<=10;i++)
     {
        scanf("%f",&f);
        if(f<=0)
             continue;          /*筛选掉部分负数数据*/
        sum=sum+f;              /*累加正数:求正数和*/
        n++;                    /*累加 1:求正数个数*/
     }
     printf("n=%d \n",n);
```

```
    printf("sum=%f\n",sum);
    printf("average=%f\n",sum/n);
}
```

练习 4-8 输出 100～200 中不能被 3 整除的数。

需要注意的是，continue 与 break 语句在循环体中使用时，一般都会与 if 语句配合出现，即满足一定条件下结束本次循环或中途退出循环，而不会无条件地中途结束。

4.7 goto 语句

goto 语句是无条件转向语句，本节只简单介绍 goto 语句的用法。

在 C 语言中，可执行语句前均可加语句标号，C 语言的语句标号不是一个整数，而是一个标识符，它标识程序的一个特定位置。语句标号的一般形式是：

标识符：可执行语句；

语句标号可以和执行语句处在同一行，也可以单独成行，处在执行语句的上一行。被定义的语句标号以冒号结尾。它和其所标识的语句之间可有一个或多个空格，不许出现其他字符。它标识出 goto 语句在程序中跳转的位置。

语句标号的引用：语句标号只能在 goto 语句中引用，其他地方不能被引用。

goto 语句的一般形式：

goto 语句标号；

goto 语句的功能是将程序转移到语句标号指定的位置继续执行。

用 goto 语句和 if 语句可构成循环，可以用 while 语句或 for 语句替代。

前文中的例 4-2(计算 1＋2＋3＋…＋100 的和)可用 goto 语句和 if 语句构成的循环来完成。

程序代码如下：

```
    ⋮
loop:if(i<=100)
    {    sum=sum+i;
         i++;
         goto   loop;
     }
     ⋮
```

goto 语句的使用范围仅局限于函数内部。

在结构化程序设计中，不提倡使用 goto 语句，因为 goto 语句的自由跳转，会使程序的基本结构遭到破坏，从而降低程序的可读性和可维护性。但是，在某些特定情况下，使用 goto 语句，在某种程度上也可为程序设计带来一些特定的方便。

4.8 循环应用

循环结构程序设计是程序设计的重要基础，在很多领域都有重要的应用。

例 4-16 求两个非负整数的最大公约数。

分析：此题是循环次数不固定的实例，采用相除取余的**迭代算法**。

已知：两正整数 x、y，两数的余数为 r。

根据数学原理，x 和 y 的公约数等于 y 和 r 的公约数(其中 r 是 x 除 y 的余数)，解题步骤如下：

第 1 步：求 x 与 y 的余数 r。

第 2 步：将 y 赋给 x，迭代 x 值。

第 3 步：将 r 赋给 y，迭代 y 值。

第 4 步：判断 y 不等于 0，返回第 1 步，直到第二项 y 为 0，第一项 x 值即为最大公约数。

例如：(x,y) ＝＞ (24,9)＝＞ (9,6)＝＞ (6,3)＝＞ (3,0)＝＞3

余数 r：　　6　　3　　0　　y＝0，3 为最大公约数

程序代码如下：

```
#include <stdio.h>
void main()
{
    int x, y, r;
    scanf("%d%d",&x, &y);                    /*输入已知量*/
    while (y !=0)                            /*循环条件*/
    {  r=x %y;                               /*相除取余*/
       x=y;
       y=r;                                  /*迭代改变循环条件变量 y 的值*/
    }
    printf("最大公约数=%d\n",x);              /*输出结果*/
}
```

运行情况：

```
24  9↙
最大公约数=3
```

例 4-17 用牛顿迭代法求方程 y：$2x^3-4x^2+3x-6=0$ 在 1.5 附近的根。

分析：牛顿迭代公式：$x_{n+1}=x_n-\frac{y}{y'}$，直到：$|x_{n+1}-x_n|<10^{-4}$。

根据方程 y 得到 y' 的公式：$y'=6x^2-8x+3$。

设 x 的初值为 1.5，根据迭代公式求出新的 x 值，如果新旧两个 x 值之差的绝对值小

于 10^{-4}，那么 x 即为方程的根，否则利用迭代公式再次求下一个 x 的值。

程序代码如下：

```
#include <stdio.h>
#include "math.h"
void main()
{
  float x,x0,y,y1;
  x=1.5;                                /*循环初值*/
  do
  {    x0=x;                            /*迭代 x0 的值,改变循环变量的值*/
       y=((2*x0-4)*x0+3)*x0-6;          /*求方程 y 在 x0 处的值*/
       y1=(6*x0-8)*x0+3;                /*求方程 y'在 x0 处的值*/
       x=x0-y/y1;                       /*用迭代公式根据 x0 求 x 的值*/
  } while(fabs(x-x0)>=1e-4);            /*循环条件:前后两个 x 差的绝对值大于 10-4*/
  printf("root=%6.2f\n",x);             /*退出循环,输出结果 x*/
}
```

运行结果：

```
root=   2.00
```

注意：本题中最关键的语句是“x0＝x;”将新计算出的 x_{n+1} 值作为下一次迭代的 x_n 的值，即改变循环变量的值，否则将会死循环。本题是用 do-while 循环完成的典型实例。因为循环条件是判断两个 x 的差值，循环初值需要先计算 y 和 y′的值，得到新的 x 值，循环体内部也要对新的 x 值再次计算 y 和 y′的值。如果用 while 循环完成，则要在循环初值及循环体内分别计算两遍 y 和 y′的值，而用 do-while 循环，计算一遍即可。

例 4-18 编一个程序求满足下列条件的四位数：第一、三位数字之和为 10，第二、四位数字之积为 12。

分析：根据题意，设 j 是一个四位数，分离个、十、百、千各位数：

设 a 为 j 的千位数，$a=j/1000$。

设 b 为 j 的百位数，$b=j/100-a*10$，或 $b=j/100\%10$。

设 c 为 j 的十位数，$c=j/10-a*100-b*10$，或 $c=j/10\%10$。

设 d 为 j 的个位数，$d=j-a*1000-b*100-c*10$，或 $d=j\%10$。

$j=a*1000+b*100+c*10+b$。

四位整数：1000～9999 之间，用一重循环实现。采用穷举法，判断并输出满足条件的数。

程序代码如下：

```
#include <stdio.h>
void main()
  { int j,a,b,c,d;
    for(j=1000;j<=9999;j++)
    {
       a=j/1000;                        /*分离各位数*/
```

```
        b=j/100-a*10;
        c=j/10-a*100-b*10;
        d=j-a*1000-b*100-c*10;
        if(a+c==10&&b*d==12)      /*第一、三位数字之和 10,第二、四位数字之积 12*/
            printf("%d ",j);      /*满足条件,输出该数*/
    }
}
```

运行结果：

```
1296  1394  1493  1692  2286  2384  2483  2682  3276  3374  3473  3672  4266
4364  4463  4662  5256  5354  5453  5652  6246  6344  6443  6642  7236  7334
7433  7632  8226  8324  8423  8622  9216  9314  9413  9612
```

例 4-19 用等距梯形法计算定积分 $\int_a^b f(x)\mathrm{d}x$，$f(x)=x^2+12x+4$，x 的取值范围是 $[-1,5]$。

分析：面积法求定积分原理：**即用等距梯形法求定积分**。

函数 $y=f(x)$ 的定积分 $\int_a^b f(x)\mathrm{d}x$ 在数值上等于如图 4-5 所示的曲边梯形（粗线条围出的区域）的面积。为了近似地表示这块面积，采用数学上的等距梯形法来计算，即将 a，b 区间划分成 n 个长度相等的小区间。用每个小区间所对应的窄梯形来代替窄曲边梯形，从而求得定积分的近似值，如图 4-5 所示。

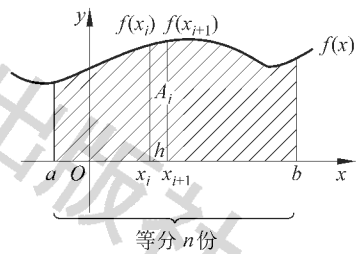

图 4-5 用等距梯形法求定积分

将曲边面积 n 等分，则每一等分的梯形面积为：（上底＋下底）＊高/2，其中：梯形的高为一等分的宽度 h，$h=(b-a)/n$，梯形的上底为 $f1=f(x)$，其值是 $f(x)$ 在 x 处的取值，根据题意 $f(x)=x^2+12x+4$，梯形的下底为 $f2=f(x+h)$，其值是 $f(x)$ 在 $x+h$ 处的取值，根据题意 $f(x+h)=(x+h)^2+12(x+h)+4$，x 值从 a 到 b 的变化，每次 x 值增加 h，累加各等分梯形的面积。即：

$$A_i=(f(x_i)+f(x_i+h))/2*h$$

$$\mathrm{AREA}=\sum_{i=1}^{n}A_i$$

程序代码如下：

```
#include <stdio.h>
void main()
{
    float x,a,b, h,f1,f2,area;
    int i,n;
    scanf("%f%f%d", &a, &b, &n);      /*输入区间 a、b,等分值 n*/
     h=(b-a)/n;                       /*将区间[a,b] n 等分,求步长*/
```

```
    area=0;
    x=a;                                /*x=区间[a,b]的初值*/
    f1=x*x+12*x+4;                      /*计算f(a)的值,梯形上底*/
    for(i=1;i<=n;i++)
    {  x=x+h;                           /*计算梯形下底的x值*/
       f2=x*x+12*x+4;                   /*计算f(x)的值,梯形下底*/
       area=area+(f1+f2)/2*h;           /*计算梯形面积,累加器*/
       f1=f2;                           /*迭代:下一个梯形上底=上一个梯形的下底*/
    }
    printf("area=%.2f\n", area);
}
```

运行情况：

```
1 4 1000↙
area=123.00
```

例 4-20 生成两个 0～9 的随机数，选菜单完成两个随机数的算术四则运算。

分析：此题采用菜单完成，主要训练学生用程序实现人机交互及程序的持续执行。

本题在上一章四则运算例 3-16 的基础上，加上循环 while (1)，在循环中使用菜单实现与用户反复交互，可执行多遍运算。选菜单 0 退出循环，用 break 实现；选菜单值超界，继续显示菜单，用 continue 实现。生成两个 0～9 的随机数，用函数 rand()%10 获取。函数 srand(time(0))使每次初始产生的随机数不同。

随机数的使用：C 语言有如下几种产生随机数的常用函数，其头文件为 stdlib. h 和 time. h。

① rand()：该函数产生一个 0～32767 之间的随机数。因此可用 rand()%10 产生一个 0～9 之间的随机数。

② srand(time(n))：该函数和 rand 函数配合使用，产生随机数的起始发生数据，n 为任意时间数值。例如 srand(time(0))是随机数初始化函数，使每次初始产生的随机数不同。

程序代码如下：

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
void main()
{
    int op;
    int a=1,b=2,c;
    srand(time(0));                 /*随机数初始化函数,使每次初始产生的随机数不同*/
    while (1)                       /*循环使用菜单*/
    {
        a=rand()%10;                /*生成两个 0~9 的随机数 a,b*/
        b=rand()%10;
```

```
        printf("          ****************************************\n");
        printf("          *          请输入选项代码(0~4)          *\n");
        printf("          *                1——加法               *\n");
        printf("          *                2——减法               *\n");
        printf("          *                3——乘法               *\n");
        printf("          *                4——除法               *\n");
        printf("          *                0——退出               *\n");
        printf("          ****************************************\n");
        scanf("%d",&op);      /*接收菜单选项值 0~4*/
        switch(op)            /*判断所选菜单值,做相应的四则运算之一*/
        {
            case  1:        c=a+b;  break;
            case  2:        c=a-b;  break;
            case  3:        c=a*b;  break;
            case  4:        if (b!=0) c=a/b;  break;
            case  0:        break;
            default:        break;
        }
        if(op<0||op>=5) continue;  /*选菜单值超界,继续显示菜单*/
        if(op==0) break;           /*选菜单 0 退出循环*/
        printf("a=%d b=%d c=%d\n",a,b,c);
    }
}
```

例 4-21 打印输出以下图案

```
   *
  ***
 *****
*******
```

程序代码如下：

```
#include <stdio.h>
void main()
{
int i,j,k;
for(i=0;i<=3;i++)
  {for(j=0;j<=2-i;j++)
        printf(" ");
   for(k=0;k<=2*i;k++)
        printf("*");
   printf("\n");}
}
```

本章小结

循环结构是结构化程序设计三大结构之一，是学习程序设计语言的基础和重点之一，同时也是难点之一，是初学者学习程序设计语言编写程序遇到的第一个坎儿。

本章主要介绍了C语言的三种循环语句的语法形式及运行流程，以及循环嵌套和循环中途退出语句等概念。C语言提供了while、do-while、for三种语句来实现循环结构。for语句功能最强，使用最多。break语句用于结束其所在的switch分支结构或while、do-while、for循环结构；continue语句用于结束本次循环。循环嵌套要注意不允许交叉。

在C语言中，while语句和for语句是当型循环语句，先判断后执行循环体；do-while语句是直到型循环语句，执行一次循环体后才判断。

编写循环程序，一般可采用穷举法、迭代法或递推法，但有时也需要具体问题具体分析，采用其他的算法解题，结合不同的应用也会有不同的解题方法。读者可通过本章大量的不同类型的实例的学习，拓宽解题思路，从中学会使用循环解决问题的逻辑思维方式及编程技巧，为后续章节课程的学习做铺垫。

习 题 4

基础部分

1. 求2+4+6+…+100(100以内偶数之和)。

2. 输入正整数n，求前n项数之和，$S=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}$。

3. 用公式$\frac{\pi}{2}=\frac{2\cdot 2}{1\cdot 3}\times\frac{4\cdot 4}{3\cdot 5}\times\frac{6\cdot 6}{5\cdot 7}\times\cdots\times\frac{(2n)(2n)}{(2n-1)(2n+1)}$求$\pi$的近似值，设$n=100$。

4. 求$s=a+aa+aaa+\cdots+aa\cdots a$的值，其中$a$为一个数字，例如3+33+333+3333，$n=4$，$n$为$a$的个数，由键盘输入。

5. 鸡、狗与九头鸟同笼。如果笼中有100个头，100只脚，并且三种动物皆有，问共有几种可能？每种又各有几只？

6. 输出100～200之间的全部素数。

7. 编写程序：把键盘输入的字符串中的数字删除，在屏幕上显示删除数字后的字符串，以回车符结束输入。

8. 找出100～999之间的所有“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身，例如，$153=1^3+5^3+3^3$，故153是水仙花数。

9. 下述程序接收从键盘输入的若干学生成绩，统计并输出最高成绩和最低成绩，当输入的成绩为负数时结束，填空完成程序。

```
#include < stdio.h>
void main( )
{ float x,fmax,fmin;
   scanf("%f",&x);
   fmax=fmin=x;
   while(__A__)
     { if(x>fmax)
          fmax=x;
       else
          if(__B__)
            fmin=x;
       scanf("%f",&x);
     }
  printf("\nmax=%f,min=%f",fmax,fmin);
}
```

10. 编一个程序求满足下列条件的四位数：该数是一个完全平方数；第一、三位数字之和为 10，第二、四位数字之积为 12。

11. 下面程序的输出结果是_____。

```
#include <stdio.h>
void main()
{
  int i,j;float s;
  for(i=7;i>4;i--)
  { s=0.0;
    for(j=i;j>3;j--)s=s+i*j;
  }
  printf("%f\n",s);
}
```

A) 154　　B) 90　　C) 45　　D) 60

12. 写出程序结束时，i、j 和 k 的值。

程序如下：

```
#include <stdio.h>
void main()
{
   int a=10,b=5,c=5,d=5;
   int i=0,j=0,k=0;
   for(;a>b;++b)i++;
   while(a>++c)j++;
   do
   {
      k++;
```

```
    } while (a>d++);
    printf("i=%d j=%d k=%d\n",i,j,k);
}
```

13. 阅读下面程序，若用户输入“BEIJING#”，程序将输出_____。

```
#include <stdio.h>
void main()
{ char ch;
  while((ch=getchar())!='#')
  { if(ch>='A'&&ch<='Z')
        ch+=32;
    putchar(ch);
  }
}
```

应用与提高部分

1. 打印出如图 4-6 所示的图案。

A
ABA
ABCBA
ABCDCBA
ABCDEDCBA
ABCDEFEDCBA

图 4-6　需打印的图案

2. 求两个非负整数的最小公倍数。

3. 用迭代法求 $x=\sqrt{a}$。求平方根的迭代公式为 $x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)$，要求：$|x_{n+1}-x_n|<10^{-5}$。

4. 求分式算术四则运算：已知$\frac{b}{a}$和$\frac{d}{c}$求二者的+、-、*、/四则运算，其结果为$\frac{y}{x}$，要求对$\frac{y}{x}$进行约分。例如$\frac{1}{6}+\frac{1}{6}=\frac{2}{6}$，约分后为$\frac{1}{3}$，即 $y=1,x=3$。

5. 用矩形法计算定积分$\int_a^b f(x)\mathrm{d}x$，$f(x)=x^2$，x 的取值范围是 $[2,3]$。

6. 用二分法求方程 $2x^3-4x^2+3x-6=0$ 在$(-10,10)$之间的根，精度为 10^{-5}。

第5章 数组

本章主要内容：

- 数组的定义、初始化以及数组元素的引用；
- 二维数组在内存中的存储；
- 字符串与字符串结束标志；
- 文件数据的处理。

5.1 引 例

例 5-1 从键盘输入 10 个整数，计算其和值。

方法一：使用前面章节中介绍的普通变量编程。

程序代码如下：

```
#include <stdio.h>
void main()
{
     int a,sum,i;
     sum=0;                              /*计算前将和 sum 设为 0*/
     printf("Please input 10 integers:\n");
     for(i=0;i<10;i++)                   /*输入 10 个整数,依次相加*/
     {
     scanf("%d",&a);
     sum+=a;
     }
    printf("The sum is %d.\n",sum);      /*输出和值*/
}
```

运行情况：

```
Please input 10 integers:
1 2 3 4 5 6 7 8 9 10↙
```