第 1 章 程序设计与 C 语言

自 1946 年世界上第一台数字计算机诞生以来，计算机技术得到了飞速发展，计算机系统已具有相当高的水准，但仍采用冯·诺依曼体系结构，即存储程序结构，这说明计算机的执行必须有程序控制，因此利用计算机解决问题，首先要解决程序设计问题。之后程序设计语言也得到了迅速发展，先后出现了机器语言、汇编语言和高级语言。

C 语言是目前比较流行、使用比较广泛的高级程序设计语言。C 语言的主要特色是兼顾了高级语言和汇编语言的特点，简洁，丰富，可移植性强。目前有许多系统软件和应用软件都是用 C 语言编写的。因此 C 语言受到了人们的极大推崇。

1.1 程序与程序设计语言

什么是程序以及如何利用程序设计语言进行程序设计是初学者面临的第一个问题。本节将从程序、程序设计及程序设计语言的发展等方面加以阐释，以引导读者对程序设计的基本思想有一个全面的了解。

1.1.1 程序

在现实世界中，我们对程序一词其实并不陌生，而且也总是在不断地编写程序和执行程序。例如，我们要用全自动洗衣机洗衣服，应该如何去做呢？这就需要编写一个程序，洗衣服的程序如下所示：

(1)把要洗的衣服放入洗衣机。

(2)插好电源，打开水龙头。

(3)设置洗衣机的控制面板。

(4)放入洗涤剂。

(5)按下开始按钮。

(6)等待洗衣机清洗完毕。

清洗完毕后,蜂鸣器发出响声,这时可将衣服取出晾晒。这就是简单的程序形式。描述这种程序,也就是给出了一个包含其中各个基本步骤的序列。如果按顺序实施这些步骤的动作,其整体效果就是完成了该项工作。

在计算机中,程序是指导计算机执行某个功能或功能组合的一组指令。每一条指令都让计算机执行完成一个具体的操作,一个程序所规定的操作全部执行完毕后,就能产生计算结果。

计算机程序设计的过程一般由以下 4 个步骤组成。

(1)分析问题。在着手解决问题之前,应该通过分析充分理解问题,明确原始数据、解题要求、需要输出的数据及形式等。

(2)设计算法。算法就是一步步的解题过程。首先集中精力于算法的总体规划,然后逐层降低问题的抽象性,逐步充实细节,直到最终把抽象的问题具体化成可用程序语句表达的算法。这是一个自上而下、逐步细化的过程。

(3)编码。利用程序设计语言表示算法的过程称为编码。程序是一个用程序设计语言通过编码实现的算法。

(4)调试程序。程序调试包括编译和链接等操作。编译程序对程序员编写的源程序进行语法检查,程序员根据编译过程中的错误提示信息,查找并改正源程序的错误后再重新编译,直到没有语法错误为止,编译程序将源程序转换为目标程序。大多数程序设计语言往往还要使用链接程序把目标程序与系统提供的库文件进行链接以得到最终的可执行文件。

1.1.2 程序设计语言

人与人之间交换信息最重要的手段是语言,包括口头语言、文字语言和图形语言等。但不同国家和不同民族的人有不同的语言,我们把这些语言统称为自然语言。目前,通用的计算机不能识别自然语言,只能识别特定的计算机语言。计算机语言,即程序设计语言是程序设计的工具,按其发展先后出现了机器语言、汇编语言和高级语言。

1)机器语言

现代计算机以冯·诺依曼体系结构为主体,其内部采用二进制形式,即用 0 和 1 表示数据和指令,这种结构的计算机能够直接识别和处理由 0 和 1 编码组合而成的二进制代码,称为机器指令,一种计算机系统的全部机器指令的集合就是该计算机的机器语言。机器语言是一种面向机器的计算机语言,与计算机型号有关,每种计算机只能识别自己的机器语言。例如,001111 就是一条机器指令,通常表示向累加器送入一个数据。

用机器语言编写的计算机程序可以被计算机直接识别和处理,运算效率较高,但是缺点也是显而易见的,即这种程序只能在特定的计算机上使用,不具有可移植性。机器指令本身很难表达其含义,对于编程者来说记忆困难。每条机器指令的功能过于单一,编程效率低。

2)汇编语言

机器语言的缺点制约了计算机的使用效率和使用范围,之后出现了汇编语言。汇编语

言使用具有一定含义的符号表示机器指令，方便了编程人员的记忆和书写，提高了编程效率。例如，ADD AX,2 是一条汇编指令，通常表示将寄存器 AX 中存储的数据加上 2 再存入寄存器 AX 中。从指令本身可以直接看出这条指令能够完成加法操作。

与机器语言相比，汇编语言容易记忆、含义明显、便于编程人员使用。但是，使用汇编语言编写的程序不能直接被计算机识别与处理，必须经过处理将其转换成二进制机器指令执行，这种处理过程称为汇编，完成这种处理功能的程序称为汇编程序。汇编语言中的汇编指令与机器语言中的机器指令是一一对应的，汇编过程主要就是将助记符式的汇编指令转换为二进制机器指令的过程，因此汇编语言也是机器相关的计算机语言。

3)高级语言

高级语言的出现使计算机语言脱离了对计算机硬件的依赖，成为一种接近于人类自然语言的计算机语言，使高效编程与计算机的广泛应用成为可能，是计算机语言发展史上的一次飞跃。例如，“y＝x＋1;”就是一条高级语言的语句，表示将变量 x 的值加 1 再存入变量 y 中。

与汇编语言相比，高级语言具有以下优点：

(1)高级语言通常有一套特定的语法，这个语法与具体的计算机系统无关，用高级语言编写的程序，可以独立于具体的计算机系统，也就是说，使用高级语言编写的程序，几乎不需作任何修改就可以运行在支持该语言的计算机系统上，因此具有可移植性。

(2)高级语言更接近于人类常用的表述方式，语句的含义更加明显，容易学习与记忆。

(3)一条高级语言语句所完成的功能相当于多条机器指令可以完成的功能，因此编程效率较高。

自从高级语言问世以来，曾得到广泛应用的高级语言有 BASIC、FORTRAN、COBOL、PASCAL、C、Ada 以及 LISP 等，这些语言都有各自的特点和适用领域。例如，FORTRAN 适用于数值计算，COBOL 适用于商业管理，C 适用于编写系统软件。

目前，面向对象编程语言已经成为继上述高级语言之后日益成熟并得到广泛应用的计算机语言，如 Smalltalk、C++ 、Java 等，而且将可视化、事件驱动等新技术与计算机语言结合在一起构建的各种集成开发环境已成为应用软件开发的重要工具，但是 C 语言的基础性作用不容忽视。

1.2　C 语言的发展及主要特点

C 语言因其功能强大和诸多优点，自诞生以来便逐渐为人们所认识，到了 20 世纪 80 年代，C 语言很快在各大、中、小和微型计算机上得到了广泛应用，成为当时最优秀的程序设计语言之一。

1.2.1　C 语言的发展历史

1987 年，美国电话电报公司(AT&T)的贝尔实验室正式发布了 C 语言。同时由

B. W. Kernighan和D. M. Ritchie合著了著名的 *The C Programming Language* 一书，通常简称为K&R，也称为K&R标准。但是，在K&R中并没有定义一个完整的标准C语言。

1983年，美国国家标准协会(American National Standards Institute，ANSI)认识到标准化将有助于C语言在商业化编程中的普及，因此成立了一个委员会为C语言及其运行制定标准，即ANSI C标准。

1989年，国际标准化组织(ISO)采纳了ANSI C标准，称为ANSI/ISO standard C (ANSI x3.159—1989)，通常称为ANSI C89标准。之后，C语言标准在很长一段时间内都保持不变，在20世纪90年代才经历了改进，这就是ISO/IEC 9899:1999，简称C99，它被ANSI于2000年3月采用。

在C语言的发展过程中，出现了多种版本的C语言。例如，Microsoft公司开发的Microsoft C或称MS C，Borland公司开发的Turbo C和Borland C，AT&T公司开发的AT&T C等，这些C语言版本不仅遵守了ANSI C标准，而且在此基础上各自进行了一些扩充，使之更加方便和易用。

1.2.2 C语言的主要特点

在众多高级程序设计语言中，C语言之所以能够存在和发展，并具有旺盛的生命力，主要源于其自身的优良特性。这些特点可概括为以下几点：

(1)C语言的标识符要求区分大小写。C语言以英文小写字母为基础，符合人们日常阅读和书写的习惯。同一个单词的大小写代表不同含义的标识符。

(2)语言简洁、紧凑，使用方便、灵活。C语言一共有32个关键字、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。

(3)模块化程序设计。C语言由函数组成，因此程序功能结构比较清楚，而且每个函数都是独立的，可以单独进行编译。

(4)运算符丰富。在C语言中，括号、赋值、强制类型转换等都作为运算符处理，从而使C语言的运算类型极其丰富，表达式类型多样化，灵活使用各种运算符可以实现其他高级语言难以实现的运算。

(5)数据结构丰富。C语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂的数据结构，如链表、树、栈等的运算。

(6)具有结构化的控制语句。C语言具有典型的结构化控制语句。例如，选择语句(if语句和switch语句)、循环语句(while语句、do…while语句和for语句)等均符合现代编程风格的要求。

(7)语法限制不太严格，程序设计自由度大。例如，对数组下标越界不作检查，由程序编写者自己保证程序的正确。对变量的类型使用比较灵活。例如，整型变量与字符型数据以及逻辑型数据可以通用。

(8)C语言允许直接访问物理地址，能够进行位操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此，C语言既具有高级语言的功能，又具有低级语言的许多功能，可以用来编写系统软件，因此有人把C语言也称为中级语言。

(9)生成的目标代码质量高，程序执行效率高。

(10)可移植性好。用 C 语言编写的程序基本上不用进行修改就能用于各种操作系统。

1.3　初识 C 语言程序

C 语言作为一种高级程序设计语言,其语法与自然语言具有一定的相似性,因此在学习 C 语言之前,有必要对 C 程序的基本结构有一个初步的感性认识。下面从最简单的例子着手,来分析 C 语言程序的构成及运行情况。

资料
等级考试重难点

【例 1-1】 编写一个简单的 C 程序,用于输出指定信息。

```
// FileName: chap1_1.c
#include <stdio.h>
int main( )
{
    printf("This is the first C program.\n");        //输出语句
    return 0;
}
```

程序运行结果如下:

```
This is the first C program.
```

【例 1-1】是一个非常简单的程序,但它体现了 C 语言的几个重要组成部分。下面分别对该程序中的各行加以分析。

程序中第 1 行以//开始的一段文字称为注释,注释文字可以由任意字符组成。注释不参与程序的运行,主要用于对程序的某些关键部分进行说明,其目的是提高程序的可读性。因此,在程序的适当位置添加必要的注释是一种良好的编程习惯,注释可以出现在程序中的任意地方。

第 3～7 行是该程序的主要组成部分,在 C 语言中称为主函数,函数名为 main。可见,C 程序是由函数构成的。任何 C 程序有且仅有一个主函数,主函数可以出现在程序的任意位置,C 程序就从这个主函数开始执行。

第 4～7 行是函数 main( )的函数体,由花括号{ }括起来。左花括号是函数体的开始,右花括号是函数体的结束。这一对花括号以及其中的程序也被称为程序块。

在函数 main( )中,第 5 行和第 6 行是完成函数功能的主要成分,在 C 语言中称为语句。每一条语句都以“;”作为结束标记。可见,C 函数是由语句构成的。

程序第 5 行语句中的 printf 是另一个函数的名字,功能是输出其后圆括号中的字符串。为了编程方便,系统提供了许多标准函数供用户使用,这样的函数称为库函数。使用这些库函数之前,需要在程序的开始包含该库函数所在的头文件,即该程序中的第 2 行。

在第 6 行的语句中可以看到一个英文单词 return,它在 C 语言中有特殊的作用,即结束本函数的执行,返回函数的调用处。这种规定了特殊作用的单词称为关键字,关键字是 C 语言语句的一种重要组成要素,每个关键字都表示某种特定的意义。

另外,C 语言不关心程序在文本行的开始位置,可以在任意位置输入程序。因此,编程

人员在输入源程序代码时，可以对源程序进行排版，使用 Tab 键缩进某些行是一种较好的排版方式，这样写出的程序层次分明，更易于阅读和理解。

1.4 C语言程序的调试

利用 C 语言编写程序的最终目的是高效地解决现实世界各领域中的实际问题，对实际问题进行分析，以 C 语言构建程序的思想为指引设计解决问题的方案，是构建 C 语言程序的第一步，通常称为程序设计。在此基础上，按照 C 语言的规则编写出 C 程序并将其存储在计算机中，运行后产生正确的结果，是构建 C 语言程序的第二步，通常称为程序生成。经过这两步的工作，达到了以计算机为工具解决实际问题的目的。在程序生成过程中，还需要不断检查程序中的语法错误，反复修改，直至得到正确的结果。

1.4.1 C语言程序的开发过程

从确定 C 语言程序算法开始编写代码到上机运行得到结果，C 语言程序的开发过程一般包括以下几个步骤：

1)程序编辑

把编写好的程序输入计算机，以文件的形式存储在磁盘中的过程，称为程序编辑。能够完成这项工作的软件称为编辑软件(也称为编辑器)。在编辑方式下建立起来的程序文件称为源程序文件，简称源文件，源文件中的程序叫做源程序。绝大多数 C 语言编译器将文件名结尾为.c 的文件看做 C 程序的源文件(Visual C++ 6.0 环境下生成的 C 程序源文件扩展名为.cpp)。例如，可以将【例 1-1】中的 C 语言源程序输入计算机并保存到一个名为 chap1_1.c 的源文件中。

2)程序编译

把编辑好的源程序翻译成目标代码的过程，称为程序编译。能够完成这项工作的软件称为编译软件(也称为编译器)。目标代码是指计算机能识别的二进制代码，存放目标代码的文件称为目标文件。通常情况下，目标文件名与源文件名相同，文件扩展名为.obj。但不同的操作系统与编译软件，也可能采用不同形式的后缀。

在程序编译过程中，对源程序中的每一条语句，编译器都要进行语法检查，当发现错误时，会在屏幕上显示错误位置和错误类型的信息。用户看到这类提示信息后，要再次调用编辑器对源程序中的错误进行修改，然后再次编译，直到排除所有的语法和语义错误。正确的源程序经过编译后在磁盘上生成目标文件。例如，对源文件 chap1_1.c 编译完成后，系统会自动生成一个名为 chap1_1.obj 的目标文件，存放在源文件所在的存储路径下。

3)程序链接

编译后产生的目标文件不能在机器上直接运行。程序中会用到库函数或者其他函数，它们都是可重定位的程序模块，需要把它们连成一个统一的整体，这个过程称为程序链接。

目标代码经过链接后，生成可以运行的可执行程序，存储可执行程序的文件称为可执行文件，通常存放在目标文件所在的存储路径下。通常情况下，可执行文件名也与源文件名相同，文件扩展名为. exe。例如，对目标文件 chap1_1. obj 完成链接后，系统会自动生成一个名为 chap1_1. exe 的可执行文件。

4)程序运行

生成可执行文件后，就可以在操作系统控制下运行了。若执行程序后达到了预期目的，则 C 语言程序的开发工作到此完成。否则，要进一步对程序进行调试，重复编辑、编译、链接以及运行的过程，直到取得预期结果。调试是指发现程序中的错误并改正错误的过程，是任何程序开发都需要经历的一个非常重要的过程，需要编程人员发挥聪明才智，根据错误的表现分析错误的原因并找到错误的位置，然后进行正确的修改。

图 1-1 所示为开发一个 C 语言程序的基本过程。

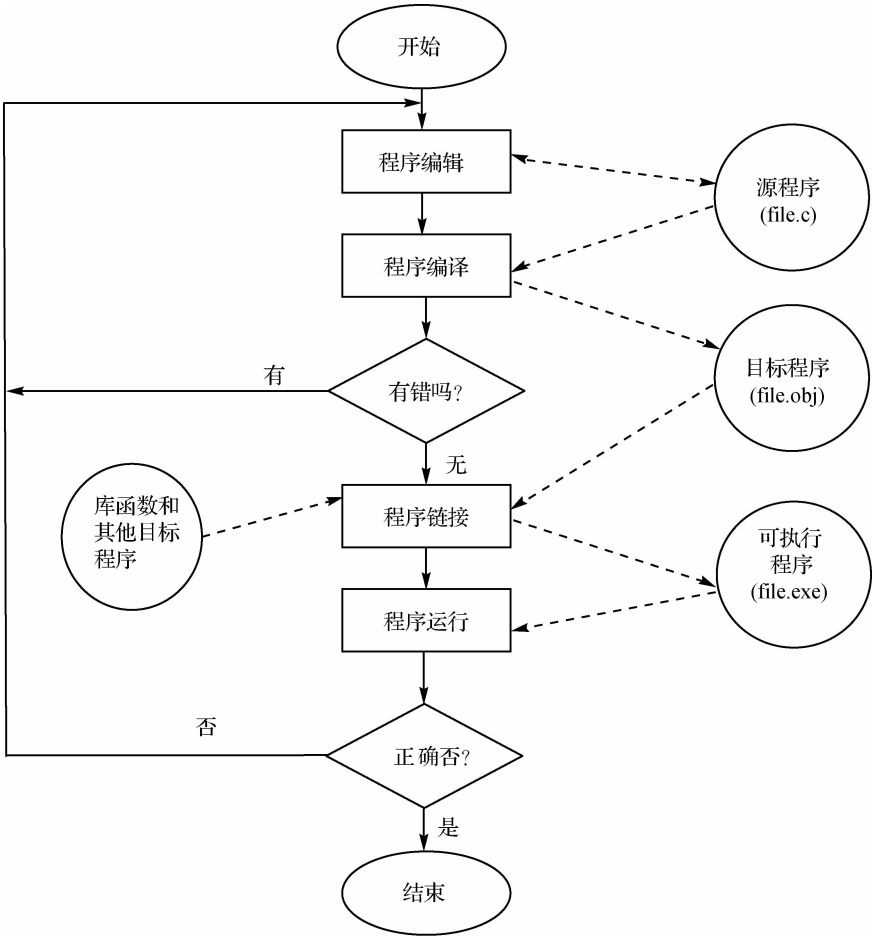

图 1-1　C 语言程序开发流程图

1.4.2　Visual C++ 集成开发环境

目前，有多种可用的 C 语言程序开发环境，如 Turbo C、Borland C、Visual C++ 、DEV－C

等,这些开发工具已将程序的编辑、编译、链接和运行的过程集成在一起,由单个应用程序来完成上述 4 项工作,并提供了多种帮助用户编写和修改程序的手段,使用非常方便。这样的应用程序称为集成开发环境(integrated development environment, IDE)。IDE 通常基于窗口环境,在窗口中完成对程序的编辑、编译、链接、运行和调试的全部工作,可以大大简化应用程序的开发过程。本书以 Microsoft Visual C++ 6.0(简称为 VC6.0)作为 C 语言程序的集成开发环境。

1)熟悉 Visual C++ 6.0 开发环境

Visual C++ 6.0 开发环境界面由标题栏、菜单栏、工具栏、项目工作区窗口、文档窗口、输出窗口及状态栏等组成,如图 1-2 所示。

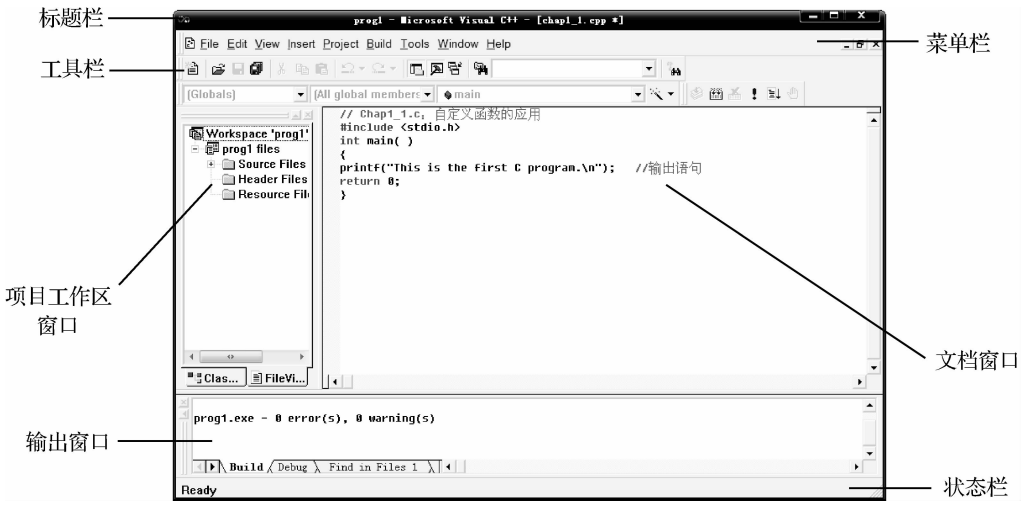

图 1-2　Visual C++ 6.0 开发环境界面

标题栏一般有“最小化”“最大化”或“还原”以及“关闭”按钮,单击“关闭”按钮将退出开发环境。标题栏上还显示出当前被操作文档的文件名。

菜单栏包含了开发环境中几乎所有的命令,它为用户提供了文档操作,程序的编译、调试,窗口操作等一系列功能。菜单中的一些常用命令还被排列在相应的工具栏上,以便用户更好地操作。

项目工作区窗口包含用户项目的一些信息,包括类(ClassView 页面)、项目文件(FileView页面)和资源(ResourceView 页面)等。在项目工作区窗口中的任何标题或图标处右击,都会弹出相应的快捷菜单,其中包含当前状态下的一些常用操作。

文档窗口一般位于开发环境窗口右侧,各种程序代码的源文件、资源文件、文档文件等都可以通过文档窗口显示出来。

输出窗口一般出现在开发环境窗口的底部,包括 Build(编译)、Debug(调试)、Find in Files(查找文件)等相关信息的输出。这些输出信息是以多页面标签的形式出现在输出窗口中的。例如,Build 页面标签显示的是程序在编译和链接时的进度及错误信息。

状态栏一般位于开发环境窗口的最底部,用来显示当前操作状态、注释、文本光标所在的行号和列号等信息。

用户可以通过鼠标拖动,将菜单栏、工具栏、项目工作区窗口和输出窗口等随意更换位

置，也可以隐藏其中的某些窗口，在不同的位置右击时会弹出相应可操作的快捷菜单命令。

2)在 Visual C++ 6.0 环境中开发和调试 C 程序的方法与步骤

在 Visual C++ 6.0 环境中开发的任何软件都是以项目形式进行管理的。Visual C++ 6.0 可以开发多种类型的项目，为了简单起见，本书将在 Visual C++ 6.0 环境中通过设计 DOS 控制台程序来学习 C 语言。

通过项目能方便组织和管理编写的程序，一个项目对应磁盘上的一个文件夹。下面介绍项目创建的过程。

(1)创建空项目 Chapter1_1。创建空项目可以分为如下几步：

①启动 Visual C++ 6.0，执行 File→New 菜单命令，弹出 New 对话框。在 Projects 选项卡中选择 Win32 Console Application 选项，如图 1-3 所示。在 Project name 文本框中输入 Chapter1_1，在 Location 文本框中指定一个路径，或者单击该文本框右侧的按钮，在弹出的 Choose Directory 对话框中选择一个路径，最后单击 OK 按钮。

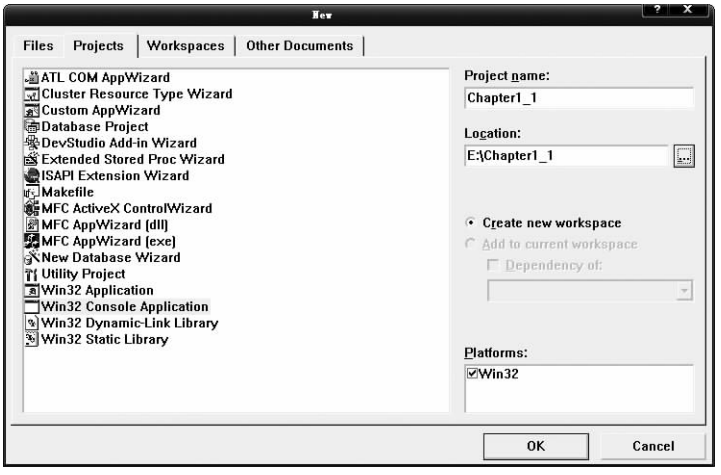

图 1-3　New 对话框

②在弹出的 Win32 Console Application-Step 1 of 1 对话框中选中 An empty project 单选按钮，单击 Finish 按钮，如图 1-4 所示。

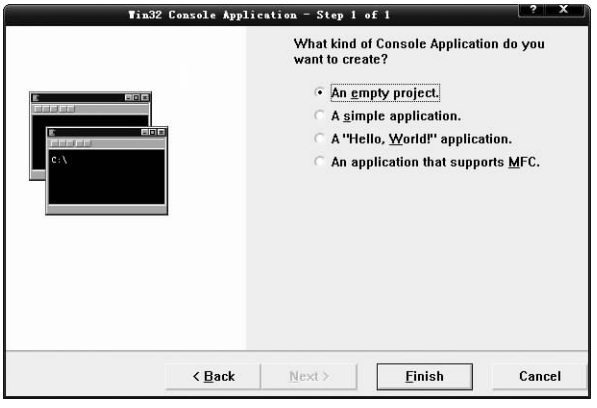

图 1-4　Win32 Console Application-Step 1 of 1 对话框

③在弹出的如图 1-5 所示的 New Project Information 对话框中单击 OK 按钮完成项目的创建，如图 1-6 所示。

图 1-5　New Project Information 对话框

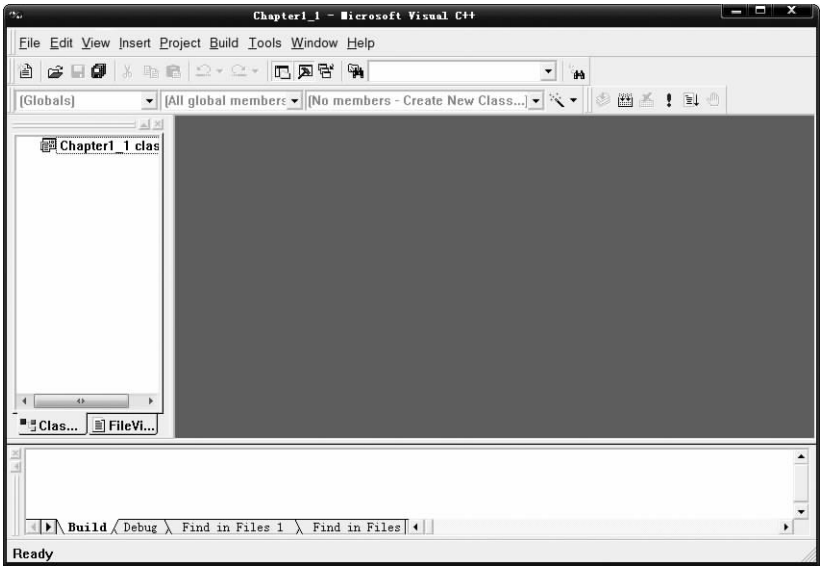

图 1-6　空工程创建完成

(2)添加空白 C 源程序文件 chapter1_1. c。在如图 1-6 所示窗口中执行 Project→Add to Project→Files 菜单命令，弹出 Insert Files into Project 对话框，如图 1-7 所示。在“文件名”文本框中输入 chapter1_1. c，单击 OK 按钮，在弹出的提示框中单击“是”按钮，打开如图 1-8 所示窗口。

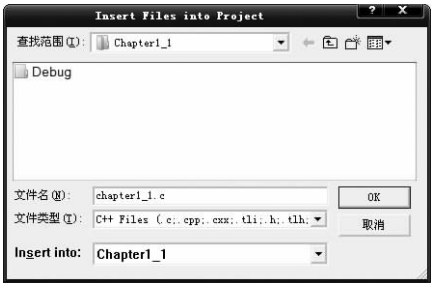

图 1-7　Insert Files into Project 对话框

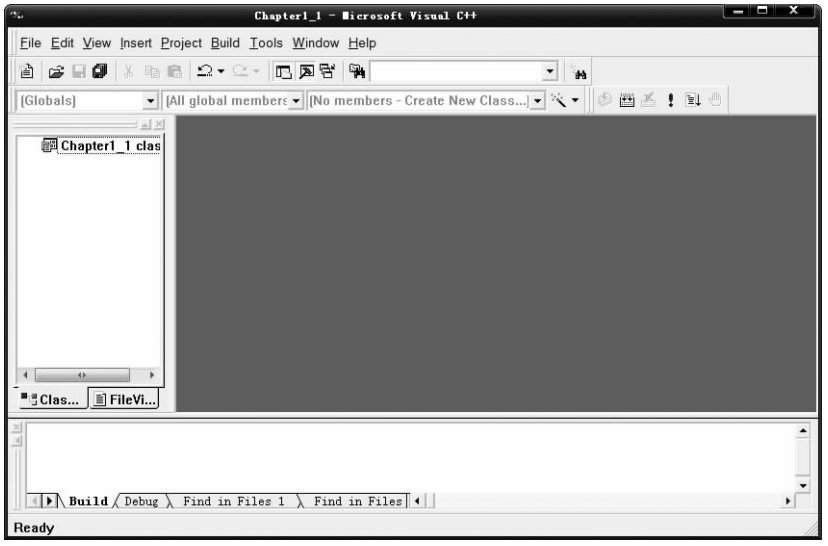

图 1-8　成功添加空白 C 源程序文件

(3)编辑 C 源程序文件 chapter1_1.c。

①在图 1-8 所示窗口中的项目工作区窗口中单击 FileView 标签,如图 1-9 所示。

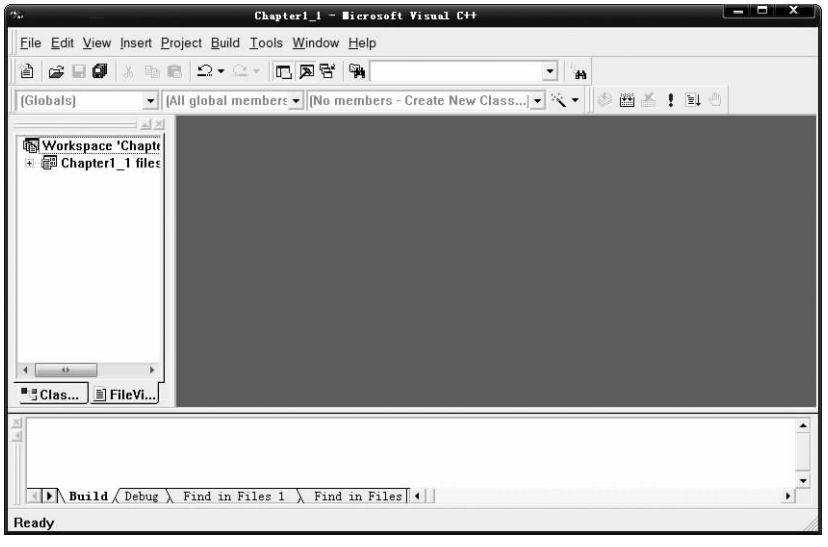

图 1-9　单击 FileView 标签

②依次展开 Chapter1_1 files→Source Files 结点，双击 chapter1_1. c 选项，在弹出的提示框中单击“是”按钮，进入源程序文件编辑状态，如图 1-10 所示。

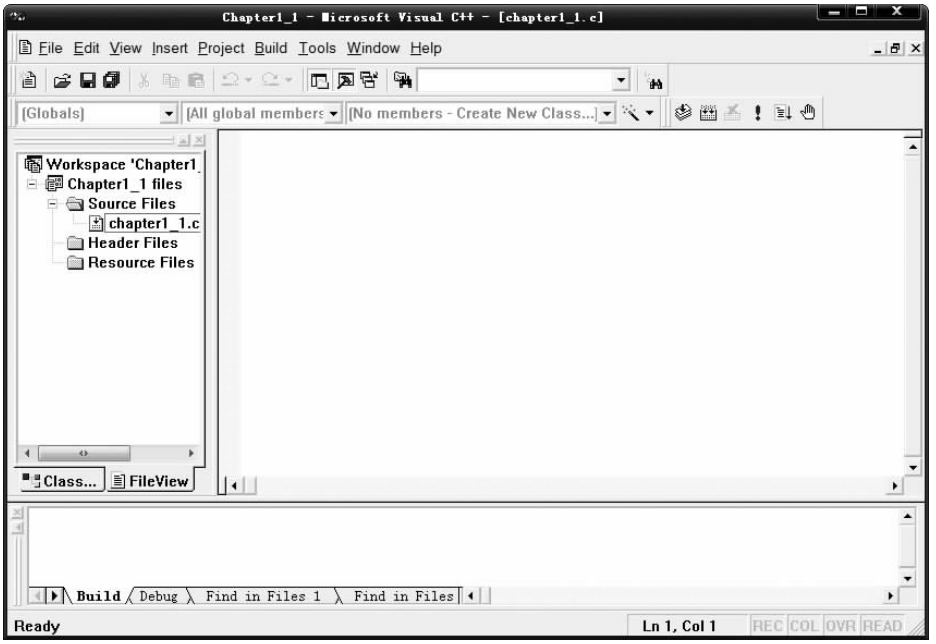

图 1-10　源程序文件编辑 1

③在文档窗口中输入【例 1-1】中的程序，如图 1-11 所示。

(4)编译。

在图 1-11 中单击工具栏 中的 按钮，或者按 Ctrl+F7 组合键，或者执行 Build→Compile chapter1_1. c 菜单命令，3 种方式都可以进行程序的编译。执行编译命令后，输出窗口会显示编译结果。

(5)链接。编译成功后，在图 1-11 中单击工具栏 中的 按钮，或者按 F7 键，或者执行 Build→Build chapter1_1. exe 菜单命令，3 种方式都可以进行程序的链接。执行链接命令后，输出窗口会显示链接结果。

(6)运行。链接成功后，在图 1-11 中单击工具栏 中的 按钮，或者按 Ctrl+F5 键，或者执行 Build→Execute chapter1_1. exe 菜单命令，3 种方式都可以运行程序。可执行程序运行后，将显示为 DOS 控制台状态，如图 1-12 所示。按任意键返回 Visual C++ 6. 0 环境。

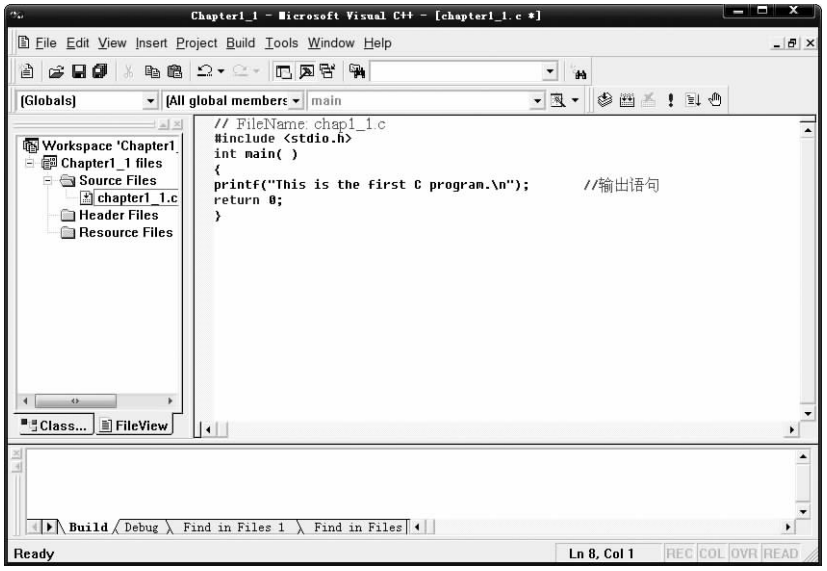

图 1-11　源程序文件编辑 2

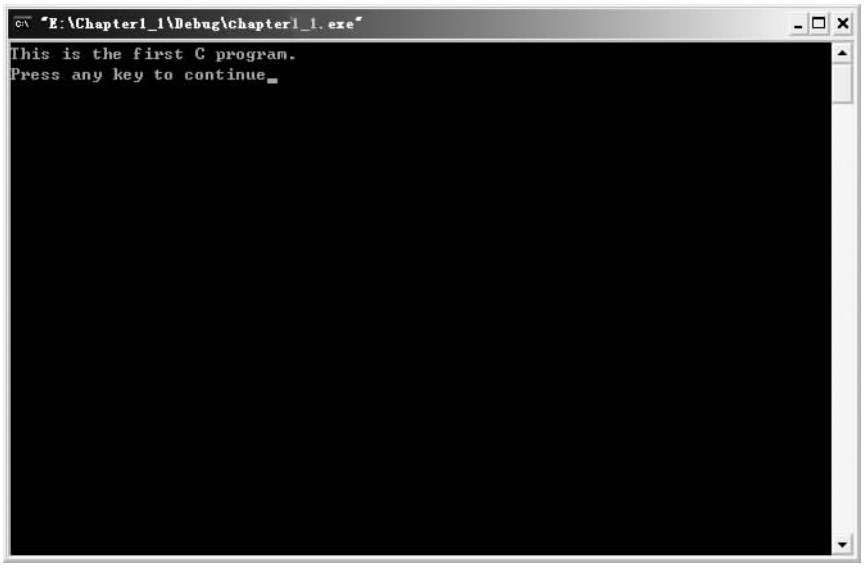

图 1-12　DOS 控制台状态

真题测试

本 章 小 结

本章介绍了程序设计的基本知识，包括程序设计的相关概念、程序设计的方法和程序设计语言的发展、C 语言的发展和特点、C 程序的组成、C 程序设计的一般步骤和使用 Visual C++ 6.0 开发和调试 C 程序的步骤。

程序设计语言先后经历了机器语言、汇编语言和高级语言三个阶段，C 语言同时兼备低级语言和高级语言的特点，因此适合于编写系统软件，这也是 C 语言区别于其他程序设计语言的一个主要特点。

C 语言是由函数组成的，一个 C 语言程序至少应该包含一个函数，即主函数，也可以包含多个子函数，从而通过函数间的相互调用来完成给定的任务。因此 C 语言也称为函数式语言。

任何一个 C 语言程序的开发过程都应包括编辑、编译、链接和执行几个过程。其中，编辑主要是在支持 C 语言开发的平台上，如 Visual C++ 6.0 等集成开发环境中输入源程序代码，编译过程主要检查语法错误，从而生成目标程序，链接过程包括将该程序用到的系统函数或其他用户自定义的函数包含进来，从而形成一个整体，最后生成可以在任何平台下运行的可执行程序。

习　题　1

1)选择题

(1)在计算机上可以直接运行的程序是(　　)。

A. 高级语言程序　　B. 汇编语言程序

C. 机器语言程序　　D. C 语言程序

(2)一个 C 语言程序由(　　)。

A. 若干函数组成　　B. 若干过程组成

C. 若干主程序组成　　D. 若干子程序组成

(3)C 语言不具有的特点是(　　)。

A. 具有结构化的控制语句

B. 数据类型丰富

C. 语法限制不太严格，程序设计自由度大

D. 在可移植性上，C 语言比其他语言差

(4)以下叙述不正确的是(　　)。

A. 一个 C 语言程序可由一个或多个函数组成

B. 一个 C 语言程序必须包含一个主函数

C. C 语言程序的基本组成单位是函数

D. 在 C 语言程序中，注释说明只能位于一条语句的后面

(5)以下叙述正确的是(　　)。

A. C 语言比其他语言高级

B. C 语言可以不用编译就能被计算机识别和执行

C. C 语言以接近英语国家的自然语言和数学语言作为语言的表达形式

D. C 语言出现的最晚，所以具有其他语言的一切优点

(6)编译 C 语言程序时，程序中的注释部分将(　　)。

A. 不参加编译，也不会出现在目标程序中

B. 参加编译，但不会出现在目标程序中

C. 不参加编译，但会出现在目标程序中

D. 参加编译，并会出现在目标程序中

2)填空题

(1)计算机语言的发展经历了_____、_____和_____阶段。

(2)C 语言既适合编写_____，也适合编写应用软件。

(3)简单地说，设计和调试 C 语言程序要经过_____、_____、_____和_____4 个阶段。

3)编程题

(1)编写一个简单的 C 语言程序，使得在屏幕上显示下列信息。

```
**********************************************************
                C is very fun.
**********************************************************
```

(2)编写一个简单的 C 语言程序，使得在屏幕上显示下列信息。

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

第 2 章 C 语言的基本知识

作为一种程序设计语言，C 语言有一套严格的字符集和语法规则，程序设计人员根据实际问题的需要，使用这些字符和语法规则编写 C 语言程序。本章主要介绍 C 语言的构成元素、C 语言的基本数据类型、常量、变量、运算符及表达式等相关概念，并从实际编程的角度介绍其具体用法。

2.1 标识符和关键字

在人们学习汉语、英语等语言时，已经了解到一门语言大体都是由本语言的符号、字、词、句、段落和文章等组成的。C 语言作为一门程序设计语言，其构成元素与汉语、英语等人类自然语言有些类似，如表 2-1 所示。

表 2-1 自然语言与 C 语言组成要素的对比

自然语言	字	词		句	段	章
		单词	短语			
C 语言	字符	标识符	表达式	语句	函数	程序

字符是 C 语言最基本的语言要素，采用一种编码形式描述，即美国国家标准信息交换码（American Standard Code for Information Interchange，ASCII），将所有的字符组织在一起形成 C 语言的字符集。将字符集中的字符按照一定的规则进行组织，构成了 C 语言中的关键字或标识符。将它们按照 C 语言规定的语法规则进行组织，构成了 C 语言中的各种语句。根据要完成的特定功能将某些语句按照一定的规则组织在一起，构成了 C 语言的函数。多个函数组合在一起构成了 C 语言程序，该程序能够完成指定的功能。因此，按照“字—词—句—段—章”的自然语言的学习顺序来学习 C 语言是一种非常有效的学习方法。

1)字符集

字符是组成语言的最基本元素,国际上使用最广泛的计算机字符编码是 ASCII 码,标准 ASCII 码字符集包括 128 个字符,主要由字母字符、数字字符、空格符、特殊字符和其他字符组成。

(1)字母字符。字母字符包括大写字母 A～Z 以及小写字母 a～z 共 52 个字符。在 ASCII 码表中,字母编码的排列顺序符合通常的自然语言顺序,而且对应的大、小写字母的 ASCII 码值相差 32。例如,大写字母 A 的 ASCII 码值为 65,大写字母 B 的 ASCII 码值为 66,依此类推。与之相对应,小写字母 a 的 ASCII 码值为 65＋32＝97,小写字母 b 的 ASCII 码值为 66＋32＝98。

(2)数字字符。数字字符包括 0～9 共 10 个字符。在 ASCII 码表中,数字编码的排列顺序也满足通常的顺序,即 0～9,并且用 ASCII 码的十六进制值表示时,个位码正好与其对应的数字相同。例如,数字 0 的 ASCII 码值为 48,用十六进制表示为 30,数字 9 的 ASCII 码值为 57,用十六进制表示为 39。

(3)空格符。空格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,因此在程序中使用一个或多个空格符对程序的编译不产生影响,但在程序中恰当地使用空格符能增加程序的清晰性和可读性。

(4)特殊字符。特殊字符是不可显示、不可打印的字符,用于计算机设备的操作控制以及在数据通信时进行传输控制,因此也称为控制符。例如,FP 表示换页,用于在打印输出时进行换页控制。

(5)其他字符。其他字符包括图形符、标点符和运算符等。例如,$ 的 ASCII 码值为 36,运算符＋的 ASCII 码值为 43。

2)标识符

所谓标识符就是用来标识在 C 语言程序中出现的符号常量、变量、数据类型、函数和语句的字符序列,C 语言中的标识符由字符组成,满足一定的构成规则。C 语言规定,标识符由字母、数字和下划线组成,且第一个字符不能使用数字字符。例如,a、_3x、BOOK1、sum5、student_1 等都是合法的标识符,而－3x、bowy－1、3Student、Sum@Mul、a＞b 等都是不合法的,不能用做 C 语言的标识符。

在符合标识符命名规则的前提下,在程序中使用含义清晰的标识符能够提高程序的可读性和可理解性,从而为程序的编写和维护提供方便,因此,建议程序编写者在命名标识符时尽量做到见名知意,尽量避免使用没有实际含义的标识符。例如,存储面积值的标识符命名为 area,表示累加和的标识符命名为 sum,而不使用 value1、abc 等这样的标识符。

用户在程序中自定义标识符时必须注意以下几点:

(1)关键字和特定标识符不能作为用户自定义标识符。

(2)标准 C 不限制标识符的长度,但有些 C 语言编译系统限制标识符的长度,同时标识符的长度也受到具体处理器的限制。

(3)在标识符中区分大小写字母。例如,Book 和 book 是两个完全不同的标识符。

(4)避免使用易混淆的字符,如整数 1 和小写字母 l,数字 0 和小写字母 o,数字 2 和小写字母 z 等,尤其对于初学者来说,这种错误在调试过程中很难发现。

(5)用户定义标识符不要与 C 语言的库函数同名。如果出现这种情况,库函数就将失去

原有含义。

3)关键字

关键字是系统定义的、具有特定含义、专门用于特定用途的 C 语言标识符，也称为保留字。关键字一般为小写字母，在使用时必须遵循一定的语法规则，如果随意使用关键字，可能会出现意想不到的错误，有时程序虽然编译通过，但运行结果却不正确，并且很难检查出来。标准 C 语言中共有 32 个关键字，如表 2-2 所示。

表 2-2　C 语言的关键字

关 键 字	含　　义	类　　型
int	整型	数据类型
short	短整型	
long	长整型	
float	单精度浮点型	
double	双精度浮点型	
char	字符型	
void	无值型	
unsigned	无符号型	
signed	有符号型	
const	常量	
struct	结构体型	
union	联合型	
enum	枚举型	
volatile	易变型	
sizeof	求字节数	运算符
if	条件语句	流程控制
else	与 if 配合使用	
switch	开关语句	
case	与 switch 配合使用	
default	与 switch 配合使用	
for	循环语句	
while	循环语句	
do	循环语句	
break	间断语句	
continue	接续语句	
return	返回语句	
goto	跳转语句	
auto	自动类型	存储类型
extern	外部类型	
static	静态类型	
register	寄存器类型	
typedef	用户自定义类型命名	

另外，C99 标准中新增了一些关键字，如_Bool、_Complex、_Imaginary、inline 和restrict。

2.2 数 据 类 型

所谓数据类型是指被定义变量的性质，是按表示形式、占据存储空间的多少和构造特点来划分的。在 C 语言中，数据类型可分为基本数据类型、构造数据类型、指针数据类型和空类型，如图 2-1 所示。

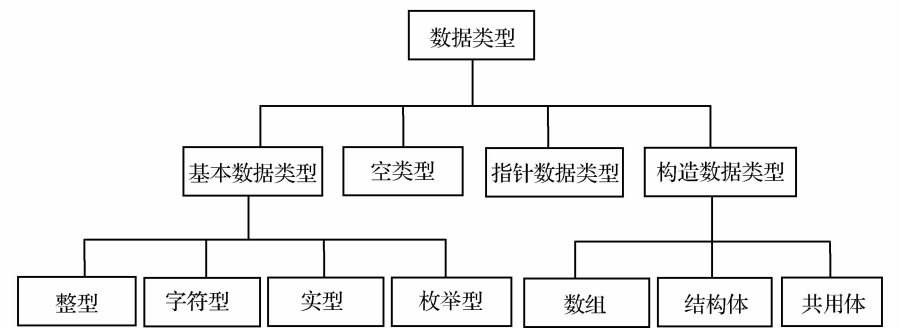

图 2-1　C 语言的数据类型

C 语言程序中用到的数据在内存中都要根据其对应的数据类型分配一定大小的内存空间，分配的内存空间的大小与具体的硬件和编译软件有关，本书在不加说明的情况下均以 16 位机 VC 编译器为例。本章主要介绍基本数据类型，其他数据类型将在后续章节中学习。ANSI C 标准规定了整型、实型和字符型的最小长度和数值范围，如表 2-3 所示。

表 2-3　基本类型的最小长度和数值范围

类型名称	中文名称	字节数/byte	位数/bit	数值范围	备　　注
char	字符型	1	8	−128～127	$-2^7 \sim (2^7-1)$
int	整型	2	16	−32 768～32 767	$-2^{15} \sim (2^{15}-1)$
float	单精度实型	4	32	$-3.4\times10^{-38} \sim 3.4\times10^{38}$	6～7 位有效数字
double	双精度实型	8	64	$-1.7\times10^{-308} \sim 1.7\times10^{308}$	15～16 位有效数字

ANSI C 标准规定，简单类型的前面还可以加上修饰符，从而使简单类型的语义更加丰富，方便 C 编程人员选用恰当的数据类型。这样的修饰符共有 4 种，即 signed(有符号)、unsigned(无符号)、long(长型)和 short(短型)。组合后形成的类型如表 2-4 所示。

表 2-4　ANSI C 标准中基本类型的最小长度和数值范围

数据类型	中文名称	字节数/byte	位数/bit	数值范围	备　注
char	字符型	1	8	−128～127	—
unsigned char	无符号字符型	1	8	0～255	—
signed char	有符号字符型	1	8	同 char	—
int	整型	2	16	−32 768～32 767	—
unsigned int	无符号整型	2	16	0～65 535	可省略整型说明符 int
signed int	有符号整型	2	16	同 int	
short int	短整型	2	16	−32 768～32 767	
unsigned short int	无符号短整型	2	16	0～65 535	
signed short int	有符号短整型	2	16	同 short int	
long int	长整型	4	32	$-2^{31}\sim 2^{31}-1$	
unsigned long int	无符号长整型	4	32	$0\sim 2^{32}-1$	
signed long int	有符号长整型	4	32	同 long int	
float	单精度实型	4	32	$-3.4\times 10^{-38}\sim 3.4\times 10^{38}$	6～7 位有效数字
double	双精度实型	8	64	$-1.7\times 10^{-308}\sim 1.7\times 10^{308}$	15～16 位有效数字

另外，C99 标准在 C89 标准基础上进行了一些修改，增加了_Bool、_Complex 和_Imaginary 三种基本类型，同时还增加了修饰符 long long，即出现了 long long int、unsigned long long int 等数据类型，并允许以 LL 或 ll 为后缀来表示 long long 型。

2.3　常量与变量

对于基本数据类型量，按其取值是否可改变又分为常量和变量两种。在程序中，常量是可以不经说明而直接引用的，而变量则必须先定义，后使用。

2.3.1　常量与变量的概念

1)常量

在程序运行过程中值保持不变的量称为常量。常量可以分为符号常量和直接常量两种。符号常量是指用标识符预定义的常量，从字面上不能直接看出其类型和值。直接常量又称字面常量，包括整型常量、浮点型常量、字符常量和字符串常量。

在 C 语言中，直接常量是直接以自身的存在形式体现值和类型的。例如，123、−5 是整型常量，1.5、−1.2E−2 是实型常量，'x'是字符常量，"first"是字符串常量。

在 C 语言中，符号常量是采用宏定义命令定义的常量，定义形式如下：

#define 符号常量名　常量

其中，符号常量名应遵循标识符的命名规则，#define 是宏定义的专用定义符，将在后面章节中进行详细讲解。例如，#define PI 3.14159 表示定义 PI 为一个符号常量，C 编译系统在处理程序时会将程序中的全部 PI 均用 3.14159 代替。

2)变量

在程序运行过程中值可以被改变的量称为变量。变量在内存中根据其数据类型占据大小不同的存储单元，用来存入可能变化的值。

变量包括变量类型、变量名和变量值 3 个基本概念。变量类型表明变量用来存放什么类型的数据，变量名用来区分并引用不同的变量，在变量的存储单元中存放的数据称为变量值。

(1)变量的定义。C 语言中的变量遵循“先定义，后使用”的原则，就是必须先对将要使用的变量进行定义，说明变量的数据类型，然后才能使用该变量。这样做的目的如下：

①变量定义是为变量指定数据类型。确定变量的数据类型后，在编译时就能在内存中分配相应的存储单元。

②能保证变量名的正确使用。

③便于在编译时根据变量的数据类型检查该变量所作的运算是否合法。例如，只有整型变量之间才能进行求余运算。

变量定义的一般形式如下：

类型说明符　变量名标识符，变量名标识符，…；

其中，类型说明符用于确定变量的数据类型，变量名标识符必须符合用户自定义标识符的命名规则。

举例如下：

```
int i,j,k;          //定义 i,j,k 为整型变量
float f1,f2;        //定义 f1,f2 为单精度实型变量
char c1,c2;         //定义 c1,c2 为字符型变量
```

在定义变量时应该注意以下几点：

①变量的定义必须在变量使用之前进行，一般放在函数体开头的声明部分。

②允许同时定义同一数据类型的多个变量，多个变量之间用“,”分隔。

③最后一个变量名后必须以“;”结束。

④类型说明符与变量名之间至少要有一个空格。

(2)对变量赋初值。C 语言中允许在定义变量的同时对变量赋初值，这也称为变量的初始化。

举例如下：

```
int a=2;
char c='x';
float x=1.2, y=2.4;
```

(3)对变量的基本操作。变量可以被看成存储数据的容器。有两种对变量的基本操作：一是向变量中存入数据，这种操作被称为赋值；二是取得变量当前的值，以便在程序运行时

使用，这种操作被称为取值。

举例如下：

```
a=2;
b=a+3;
```

第一条语句是将 2 赋值给变量 a，即变量 a 对应的存储单元中存放 2。第二条语句先将变量 a 的当前值 2 取出，加上 3 后得到 5 的结果再赋值给变量 b。

2.3.2 整型常量与变量

整型数据没有小数部分，根据占有存储空间的长度，分为基本整型、短整型和长整型 3 种，类型说明符分别为 int、short int(或 short)以及 long int(或 long)。根据存储单元中是否有符号位，整型数据又可分为有符号类型和无符号类型。对于有符号类型，在存放整数的存储单元中，最左面的一位表示符号，该位为 0 时，表示数值为正；该位为 1 时，表示数值为负。

整型数据在内存中是以二进制方式存放的，最高位为符号位，并以补码表示。将一个十进制整数转化为补码表示的方法如下：

(1)对于正数，其补码表示与原码相同。

(2)对于负数，其补码表示为反码加 1(负数的反码为其绝对值的所有位取反)。

例如，求－10 的补码的方法如下：

①取－10 的绝对值 10。

②10 的二进制形式为 0000000000001010(一个整数占 16 位)。

③对 0000000000001010 取反得到 1111111111110101。

④再加 1 得 1111111111110110，如图 2-2 所示。

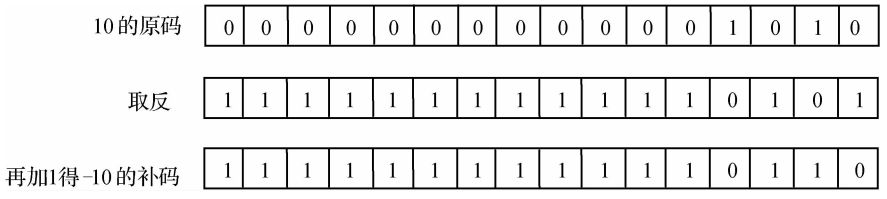

图 2-2 求－10 的补码

1)整型常量

在 C 语言中，整型常量可以表示为十进制、八进制和十六进制 3 种。

(1)十进制整型常量。十进制整型常量的形式为 d。其中，d 可以是 0～9 的一个或多个十进制数，首位不能为 0。进位规则为逢十进一。例如，236、－578、65 535、1 627 均是合法的十进制整型常量，而 025、23A 是非法的十进制整型常量，因为 025 首位为 0，23A 中含有非十进制数码 A。

(2)八进制整型常量。八进制整型常量的形式为 0d。其中，d 可以是 0～7 的一个或多个八进制数，起始 0 是必需的引导符。进位规则为逢八进一。例如，016(十进制的 14)、0111(十进制的 73)均是合法的八进制整型常量，而 256、03A2 是非法的八进制整型常量，因为 256 没有前缀 0，03A2 中含有非八进制数码 A。

(3)十六进制整型常量。十六进制整型常量的形式为 0xd。其中,d 可以是一个或多个十六进制数(0～9 的数字,或 a～f 的字母)。引导符 0 是必需的,字母 X 可以用大写或小写,进位规则为逢十六进一。例如,0X2B(十进制为 43)、0XA0(十进制为 160)、0XFFFF(十进制为 65 535)均是合法的十六进制整型常量,而 5A、0X3H 是非法的十六进制常量,因为 5A 没有前缀 0X,0X3H 中含有非十六进制数码 H。

需要强调的是,十进制、八进制和十六进制只是整数的三种不同的表示形式,在计算机内部都将转换成相应的二进制数存储。因此,同一个整数可以有三种不同的表示方法。下面的例子分别给出了整数 10 的十进制、八进制和十六进制表示。

```
10     // 十进制整数 10,在内存中对应二进制数 0000000000001010
012    // 八进制整数 12,在内存中对应二进制数 0000000000001010
0xa    // 十六进制数 a,在内存中对应二进制数 0000000000001010
```

默认情况下,在－32 768～32 767 范围内的整型常数的数据类型是 int 型,超过此范围而在－2 147 483 648～2 147 483 647 范围内的整型常数的数据类型是 long 型。也可以通过在整型常数后面加上字母后缀来强制指定其数据类型,C 语言规定的字母后缀的具体含义如下:

- 后缀 l 或 L 表示 long 型常数。例如,－12l、01235456720L。
- 后缀 u 或 U 表示 unsigned 型常数。例如,12u、034u、0x2fdU。
- 后缀 lu 或 LU 表示 unsigned long 型常数。例如,123246875LU。

2)整型变量

整型变量可以分为以下几类:

(1)基本整型。类型说明符为 int,在内存中占 2 字节,其取值范围为－32 768～32 767。

(2)短整型。类型说明符为 short int 或 short,在内存中所占字节数和取值范围均与基本整型相同。

(3)长整型。类型说明符为 long int 或 long,在内存中占 4 字节,其取值范围为－2 147 483 648～2 147 483 647。

(4)无符号型。类型说明符为 unsigned,此类型的整数没有负数,分为以下 3 种:

①无符号基本整型。类型说明符为 unsigned int 或 unsigned,在内存中占 2 字节,其取值范围为 0～65 535。

②无符号短整型。类型说明符为 unsigned short int 或 unsigned short,在内存中所占字节数和取值范围均与无符号基本整型相同。

③无符号长整型。类型说明符为 unsigned long int 或 unsigned long,在内存中占 4 字节,其取值范围为 0～4 294 967 295。

整型变量在使用过程中要先定义后使用。

【例 2-1】 整型变量的定义与使用。

```
//FileName: chap2_1.c
#include <stdio.h>
int main( )
{
    int a,b,c,d;                    //定义 a、b、c、d 为基本整型变量
```

```
    unsigned u;                    //定义 u 为无符号基本整型变量
    a=10; b=-20; u=5;
    c=a+u;
    d=b-u;
    printf("c= %d,d= %d\n",c, d);
    return 0;
}
```

程序运行结果如下：

```
c=15,d=-25
```

2.3.3 实型常量与变量

实型数据又称为浮点型数据，是带小数部分的数据。根据能够表示的大小和精度，浮点型数据分为单精度、双精度两类，类型说明符分别为 float 和 double。浮点型数据在内存中以指数形式存储。C 语言将一个浮点型数据分成小数和指数两个部分存储。例如，实数 3.141 59 在内存中的存放形式如图 2-3 所示。

真题测试

+	.314159	1
符号	小数部分	指数部分

$+ \quad 0.314159 \times 10^{1}$

图 2-3 实数 3.141 59 在内存中的存放形式

图 2-3 是用十进制数来表示的，在计算机中实际上是用二进制数来表示符号和小数部分，用 2 的幂次来表示指数部分。分配给实型数的若干字节中，到底使用多少位来表示小数部分及指数部分，ANSI C 未作具体规定，由 C 编译系统自行决定。

1)实型常量

在 C 语言中，实型常量有两种表示形式：十进制小数形式和指数形式。

(1)十进制小数形式。由正负号、数码 0～9 和一个小数点组成，小数点前面和后面可以没有数字。例如，下面都是正确的实型常数。

```
.123        // 表示实数 0.123
-.123       //表示实数-0.123
123.        //表示实数 123.0
0.          //表示实数 0.0,也可以写成.0,但不可以把两个 0 都省略
```

(2)指数形式。由十进制小数(或整数)与字母 e(或 E)组成。一般形式如下：

*a*E*n*

或

*a*e*n*

其中，a 可以是十进制小数或整数，n 必须为十进制整数，整体表示数 $a\times10^{n}$。对于以指数形式表示的实型常量，要求字母 e(或 E)前面必须有数字，后面必须为整数。例如，2.1E5、

3.7E7、－2.8E－2 等均是合法的实数，而 E7、2.7E 不是合法的实数，因为 E7 中字符 E 之前没有数字，2.7E 中字符 E 之后没有整数。

实型常量在不加任何后缀时，系统默认为双精度型。实型常量的后缀用大写字母 F 或小写字母 f 表示单精度型，用大写字母 L 或小写字母 l 表示长精度 long double 型。

2)实型变量

实型变量可以分为以下 3 类：

(1)单精度型。类型说明符为 float，在内存中占 4 字节，其取值范围的绝对值为 10^{-38}～10^{38}，提供 6～7 位有效数字。

(2)双精度型。类型说明符为 double，在内存中占 8 字节，其取值范围的绝对值为 10^{-308}～10^{308}，提供 15～16 位有效数字。

(3)长双精度型。类型说明符为 long double，在内存中占 10 字节，其取值范围的绝对值为 $10^{-4\,932}$～$10^{4\,932}$，提供 18～19 位有效数字。

实型数据的存储形式决定了它能提供的有效数字是有限的，有效数字位数以外的数字会被舍去，所以实型数据一般存在误差。

【例 2-2】 实型变量的定义与使用。

```
//FileName: chap2_2.c
#include <stdio.h>
int main( )
{
    float a,b;                    //定义 a、b 为单精度浮点型变量
    double d;                     //定义 d 为双精度浮点型变量
    a=3.56; b=12345.678;
    d=12345.6789;
    printf("%f,%f,%f\n",a,b,d);
    return 0;
}
```

程序运行结果如下：

```
3.560000,12345.677734,12345.678900
```

需要说明的是，变量 b 为 float 类型，有效数字为 6～7 位，故为 b 赋值为 12 345.678，输出时却显示 12 345.677 734，前面 7 位数字是有效的，后面出现误差。

2.3.4　字符型常量与变量

计算机中处理的数据不仅是整数、实数这样的数值，还包括字符型数据。在 C 语言中字符型数据包括字符和字符串两种。字符型数据在内存中是以字符的 ASCII 码值的二进制形式存储的，一个字符占用一个字节。

例如，字符'a'在内存中的存储形式如图 2-4 所示。

1)字符常量

C 语言中的字符常量分为普通字符常量和转义字符两种：

0	1	1	0	0	0	0	1

图 2-4　字符'a'的内存存储形式

(1)普通字符常量。普通字符常量是用单引号括起来的一个字符。该字符可以是数字、字母等 ASCII 字符集中除'和\之外的所有可显示字符。例如,'a'、'E'、'3'、'+'、'MYM'等都是合法的字符常量。

(2)转义字符。转义字符是一种特殊的字符常量。转义字符以反斜杠\开头,后跟一个或几个字符。转义字符具有特定的含义,不同于字符原有的意义,故称"转义"字符。例如,'\n'就是一个转义字符,表示换行。常用转义字符的具体含义如表 2-5 所示。

表 2-5　常用转义字符及其含义

转义字符	含　义	ASCII 码值
\a	响铃(BEL)	7
\b	退格(BS)	8
\f	换页(FF)	12
\n	换行(LF)	10
\r	回车(CR)	13
\t	水平制表(HT)	9
\v	垂直制表(VT)	11
\\	反斜杠	92
\?	问号字符	63
\'	单撇号字符	39
\"	双撇号字符	34
\0	空字符(NULL)	0
\ddd	任意字符	1～3 位八进制
\xhh	任意字符	1～2 位十六进制

可将表 2-5 中转义字符的使用方法归纳为以下 3 种:

①反斜杠后面跟某些特定字符表示不可打印的控制字符和特定功能的字符。

②表示具有特定含义的单撇号、双撇号和反斜杠字符。

③反斜杠后面跟一个八进制或十六进制数表示任意字符,其中八进制或十六进制数是该字符对应的 ASCII 码。例如,'\102'表示 ASCII 码为八进制 102 的字符'B'。

【例 2-3】 转义字符的使用。

```
//FileName: chap2_3.c
#include <stdio.h>
int main( )
{
```

```
    printf("\101 \x42 c\n");
    printf("I say:\"How are you? \"\n");
    printf("\\C program\\\n");
    return 0;
}
```

程序运行结果如下：

```
A  B  C
I say: "How are you? "
\C program\
```

2)字符变量

字符变量的类型说明符为 char。字符变量的定义与其他类型变量相同，如下所示：

```
char a,b;
```

每个字符变量被分配一个字节的内存空间。由于字符变量在内存中存放的是字符的 ASCII 码值，所以也可以把它们看成整型量。字符型数据与整型数据之间的转换比较方便。字符数据可以参与算术运算，也可以与整型量相互赋值，还可以按照整数形式输出。

【例 2-4】 分析以下程序的执行结果。

```
//FileName: chap2_4.c
#include <stdio.h>
int main( )
{
    short int n=97;  //字符'a'的 ASCII 码为 97
    printf("%d, %c, %d, %c\n", n, n, n+1, n+1);
    return 0;
}
```

程序运行结果如下：

```
97, a, 98, b
```

在【例 2-4】中，字符'a'的 ASCII 码为 97，变量 n 以整型数据方式输出时为 97，以字符方式输出时为'a'，如图 2-5 所示。n+1 的结果为 98，字符'b'的 ASCII 码为 98，n+1 以整型数据方式输出时为 98，以字符方式输出时为'b'。

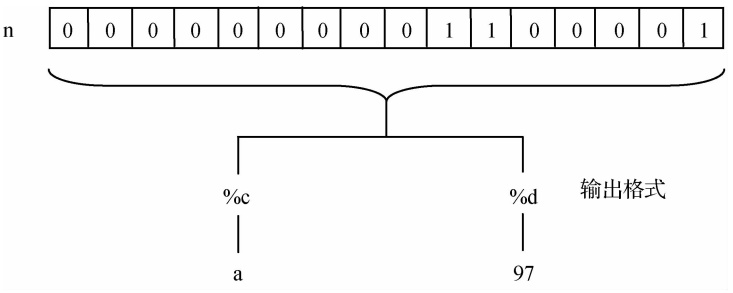

图 2-5　变量 n 的两种输出方式

3)字符串常量

C 语言中的字符串常量是由一对双引号括起来的字符序列。例如,"C program"、"China"、"123.456"等都是合法的字符串常量。

字符串常量和字符常量是不相同的量,其区别如下:

(1)从表示形式上看,字符常量是由单引号括起来的,字符串常量是由双引号括起来的。

(2)从字符的个数上看,字符常量只能是单个字符,字符串常量可以包含 0 个或多个字符。

(3)有字符变量,但没有字符串变量。C 语言没有专门的字符串类型变量,而是使用字符型数组或字符型指针来存储字符串。

(4)字符常量在内存中占 1 字节,字符串常量在内存中的字节数为字符个数加 1。这是因为 C 语言规定,每一个字符串的末尾加一个字符串结束标志\0(ASCII 码为 0)。

例如,字符串"China"在内存中实际占 6 字节,如图 2-6 所示。

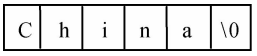

图 2-6 字符串常量"China"在内存中的存储

所以,字符常量'a'与字符串常量"a"虽然都只有一个字符,但两者截然不同,'a'在内存中占 1 字节,而"a"在内存中占 2 字节。

2.4 运算符和表达式

运算符是告诉编译程序执行特定算术或逻辑操作的符号,C 语言的内部运算符很丰富,除控制语句和输入/输出语句外,几乎所有基本操作都被作为运算符处理。C 语言有三种运算符:算术运算符、关系与逻辑运算符、位操作运算符。另外,C 语言还有一些特殊的运算符用于完成一些特定的任务。C 语言的运算符分类如表 2-6 所示。

表 2-6 C 语言的运算符

运算符类型	运算符
算术运算符	+、-、*、/、%、++、--
关系运算符	>、<、==、>=、<=、!=
逻辑运算符	!、&&、||
位操作运算符	<<、>>、~、|、^、&
赋值运算符	=及其复合赋值运算符
条件运算符	?:
逗号运算符	,
指针运算符	*、&
求字长运算符	sizeof
特殊运算符	( )、[ ]、->、.

C 语言提供的运算符很多，学习时应该注意以下几点：

(1)运算符的功能。首先要记住运算符的功能。有的运算符可能有双重功能，当连接的运算对象不同时，功能也不同。例如，若有“int x;”，则“&x”表示取变量 x 的地址，而“5&3”表示对 5 和 3 进行位运算。

(2)运算符的优先级。C 语言中的运算符优先级共分为 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。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。

(3)运算符的结合性。C 语言中各运算符的结合性分为两种：左结合性(自左至右)和右结合性(自右至左)。例如，算术运算符是左结合性运算符，即先左后右；赋值运算符是右结合性运算符，即先右后左。

(4)运算结果及其表示。参与运算的数据类型不同，得到的结果就不同，因而表示方法和输出方法也不同。

按照一定的 C 语言语法规则，用运算符将常量、变量和函数连接起来组合而成的式子，就是 C 表达式。在 C 表达式中，还可以使用圆括号( )将需要优先计算的部分括起来。单个常量或变量可以看成 C 表达式的特例。任何表达式按一定规则计算后都会得到一个结果值，该结果具有相应的数据类型。

2.4.1　算术运算符及算术表达式

1)算术运算符

算术运算符在很多计算机语言中都是最常用的，C 语言的算术运算符可以分为基本算术运算符、自增和自减运算符以及正负号运算符。

(1)基本算术运算符。基本算术运算符包括＋(加)、－(减)、×(乘)、/(除)、％(取余) 5 种。

基本算术运算符都是双目运算符，即应有两个运算对象参与运算，如 a＋b、a－b、a×b、a/b、a％b 等。基本算术运算符的优先级遵循“先乘除，后加减”的规则。×、/、％为同一级别，＋和－为同一级别，且×、/、％的优先级别高于＋和－。基本算术运算符的结合性为左结合。

使用算术运算符时应注意以下两个问题：

①如果有一个实型数参与运算，那么运算结果就是 double 类型，因为所有实数都按 double 类型进行运算。其中，/运算在参与运算的量均为整型时，结果也为整型。

②取余运算要求参与运算的量均为整型，运算的结果等于两数相除后的余数。

【例 2-5】 /(除)和％(取余)运算符举例。

```
//FileName: chap2_5.c
#include <stdio.h>
int main( )
{
    printf("%d, %d\n",20/7,-20/7);
    printf("%f, %f\n",20.0/7,-20.0/7);
```

```
    printf("%d\n",100%3);
    return 0;
}
```

程序运行结果如下：

```
2,-2
2.857143,-2.857143
1
```

(2)自增和自减运算符。C 语言中的自增运算符为“++”,自减运算符为“--”,都是单目运算符,具有右结合性。运算符“++”表示操作数加 1,运算符“--”表示操作数减 1。

自增和自减运算符可用在操作数之前,也可用在操作数之后。例如,“x=x-1”可写成“x--”或“--x”,但在表达式中这两种用法是有区别的。自增或自减运算符在操作数之前,表示在引用操作数之前就先执行加 1 或减 1 操作;自增或自减运算符在操作数之后,表示先引用操作数的值,之后再进行加 1 或减 1 操作。

【例 2-6】 自增和自减运算符举例。

```
//FileName: chap2_6.c
#include <stdio.h>
int main( )
{
    int i,k;
    i=5; k=++i;                    //赋值时,i 先增 1,再将 i 的值赋给 k
    printf("k=%d,i=%d\n",k,i);
    i=5; k=--i;                    //赋值时,i 先减 1,再将 i 的值赋给 k
    printf("k=%d,i=%d\n",k,i);
    i=5; k=i++;                    //赋值时,先将 i 的值赋给 k,再将 i 增 1
    printf("k=%d,i=%d\n",k,i);
    i=5; k=i--;                    //赋值时,先将 i 的值赋给 k,再将 i 减 1
    printf("k=%d,i=%d\n",k,i);
    return 0;
}
```

程序运行结果如下：

```
k=6,i=6
k=4,i=4
k=5,i=6
k=5,i=4
```

在使用自增和自减运算符时,应注意以下两个问题：

①自增和自减运算符的优先级高于基本算术运算符。

②自增和自减运算符的操作数只能是变量,不能是常量和表达式。

(3)正负号运算符。正负号运算符是单目运算符,如-a、-b、-5、+8 等。

正负号运算符的优先级与自增和自减运算符同级,高于基本算术运算符。它的结合方

向为自右向左。

举例如下：

```
int i,k;
i=5;
k=-i--;
```

表达式“-i--”相当于“-(i--)”，因此 i 的最终值为 4，k 的最终值为-5。

2)算术表达式

用算术运算符将运算对象连接起来，符合 C 语法规则，并能说明运算过程的式子，称为算术表达式。算术表达式的构成规则如下：

(1)数值型常量、数值型变量、数值型函数调用。

(2)+(算术表达式)、-(算术表达式)。

(3)++整型变量、--整型变量、整型变量++、整型变量--。

(4)(算术表达式)双目算术运算符(算术表达式)。

(5)有限次使用上述规则获得的运算式也是算术表达式。

算术表达式的类型可能是整型、单精度实型或双精度实型。由于 C 语言中其他类型表达式的值也是整型或实型，所以也可以当做算术表达式来使用。

2.4.2　关系运算符及关系表达式

关系运算实际上就是比较运算，即将给定的两个运算对象进行比较，判断比较的结果是否符合给定的条件。例如，a>b 中的>表示一个大于关系运算，如果 a=5，b=3，那么大于关系运算的结果为真，即条件成立；如果 a=2，b=3，那么大于关系运算的结果为假，即条件不成立。

1)关系运算符

C 语言提供了 6 种关系运算符，如表 2-7 所示。

表 2-7　关系运算符

关系运算符	含　义
<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
!=	不等于

在关系运算符中，<、<=、>、>=的优先级相同，==和!=的优先级相同，且前 4 个运算符的优先级高于后 2 个。关系运算符的优先级低于算术运算符。

2)关系表达式

关系表达式是由关系运算符连接表达式构成的，具体构成如下：

表达式　关系运算符　表达式

其中，表达式主要是算术表达式，也可以是字符型数据或关系表达式、逻辑表达式、条件表达式、赋值表达式或逗号表达式等。

关系表达式的值为逻辑值，有 true(用 1 表示)和 false(用 0 表示)两种。

例如，假设 a=3，b=4，c=5，则各种表达式的值如下所示：

(1)“a>b”的值为 false(0)。

(2)“(a>b)!=c”的值为 true(1)。因为“a>b”的值为 false(0)，而 0 不等于 c，所以该关系表达式成立，即为 true(1)。

(3)“a<b<c”的值为 true(1)。因为“a<b”的值为 true(1)，而 1 小于 c 成立，所以该关系表达式成立，即为 true(1)。

(4)“(a<b)+c”的值为 6。因为“a<b”的值为 true(1)，所以 1+5=6。

2.4.3　逻辑运算符及逻辑表达式

C 语言提供逻辑运算符，逻辑运算的结果为 true(真)或 false(假)。

1)逻辑运算符

表 2-8 列出了 C 语言中的逻辑运算符。ANSI C 标准规定，参与逻辑运算的操作数可以不是逻辑值，该操作数非 0 时表示真，为 0 时表示假，但逻辑运算的结果只可以取逻辑值(真或假)，返回值为 1 或 0。

表 2-8　逻辑运算符

逻辑运算符	名称及含义	举　例
!	逻辑非(单目)	!x
&&	逻辑与(双目)	x&&y
||	逻辑或(双目)	x||y

逻辑与运算符 && 和逻辑或运算符||都是双目运算符，逻辑非运算符!是单目运算符。其运算规则与数学中的定义相同，表 2-9 列出了逻辑运算的真值表。

表 2-9　逻辑运算的真值表

a	b	!a	!b	a&&b	a||b
非 0	非 0	0	0	1	1
非 0	0	0	1	0	1
0	非 0	1	0	0	1
0	0	1	1	0	0

从表 2-9 可以看出：

(1)对于逻辑与运算，当 a 和 b 同时为真时，a&&b 的值为真，否则为假。

(2)对于逻辑或运算，当 a 和 b 同时为假时，a||b 的值为假，否则为真。

(3)对于逻辑非运算,就是对操作数进行取反操作。

逻辑运算符中 &&. 和||的优先级低于关系运算符,!的优先级高于算术运算符。

2)逻辑表达式

用逻辑运算符和圆括号将操作数连接起来的、符合 C 语言语法规则的式子称为逻辑表达式。具体构成如下:

单目逻辑运算符　表达式

或

表达式　双目逻辑运算符　表达式

其中,表达式主要是关系表达式,也可以是字符型或算术表达式、条件表达式、赋值表达式、逗号表达式等。

需要强调的是,在求解逻辑表达式时,并不是所有逻辑运算符都要被执行,当表达式的运算结果能够确定时,运算过程将立即终止,后面的部分将不予执行。这种现象称为逻辑运算符的短路现象,也称懒惰求值法。具体情况如下:

(1)x &&. y &&. z。只有 x 为真(非 0)时,才需要判断 y 的值,只有 x 和 y 都为真的情况下才需要判断 z 的值。因此只要判定 x 为假,系统就终止运算,此时整个表达式的值已经确定为假。

(2)x||y||z。只有 x 为假时,才需要判断 y 的值,只有 x 和 y 都为假才需要判断 z 的值。因此只要判定 x 为真,系统就终止运算,此时整个表达式的值已经确定为真。

下面举例说明逻辑表达式的应用。

【例 2-7】 假设 x=10,y=20,分别计算以下各逻辑表达式的值。

①!x。

②x &&. y。

③!x+5||10 % y>=x-10<y。

上述逻辑表达式的计算过程及结果如下:

①由于 x=10,根据 C 语言规定,非 0 为真,因此表达式!x 的计算结果为假,表达式返回 0。

②操作数 x 和 y 的值都是非 0 值,根据运算符 &&. 的运算规则,表达式 x&&.y 的计算结果为真,表达式返回 1。

③该表达式等价于((!x)+5)||(((10%y)>=(x-10))<y),逻辑运算符||前面的表达式结果为真,根据逻辑运算符的短路现象,不需要计算后面表达式的值便知整个表达式的结果为真,表达式返回 1。

【例 2-8】 写出满足要求的合法的 C 语言逻辑表达式。

①用 x 表示 0～9 的字符。

②x 和 y 都是大于 0 的数。

③判断 x 的取值范围在 40～100 之间,即 40≤x≤100。

④判断某一年是闰年。

满足上述条件的合法的 C 语言逻辑表达式如下:

①x>=48 &&. x<=57。

②x>0 &&. y>0。

③x>=40 && x<=100。

④year % 4==0 && year % 100 !=0||year % 400==0。

2.4.4 赋值运算符及赋值表达式

真题测试

赋值运算符(=)用于赋值运算,是 C 语言中最基本的运算符,分为基本赋值运算和复合赋值运算。由=连接的式子称为赋值表达式。

1)基本赋值运算

基本赋值运算符的符号为=,其作用是将右侧表达式的值赋给左侧变量,基本赋值运算符是双目运算符。例如,x=10 表示执行一次赋值运算,把常量 10 赋给变量 x。

由赋值运算符将一个变量和表达式连接起来的式子称为赋值表达式。这里要注意的是,赋值运算符的右侧可以是任意一个合法的 C 语言表达式,包括常量或者另一个赋值表达式,但左侧必须是一个变量,不可以是常量或表达式。例如,x=10 和 x=y=10 都是合法的赋值表达式,但 5=8 和 x+y=8 都是不合法的赋值表达式。

ANSI C 标准规定,赋值运算符的优先级低于算术运算符、关系运算符和逻辑运算符,其结合性为右结合。

【例 2-9】 假设变量 x 为整型,计算以下各赋值表达式的值。

①x=y=10。

②x=10+(y=20)。

③x=10+(y=20)/(z=30)。

上述赋值表达式的计算过程及结果如下:

①根据赋值运算符的右结合性,先将 10 赋值给变量 y,此时变量 y 的值为 10,同时表达式"y=10"的值也为 10,然后将表达式的值 10 赋给变量 x,此时 x 的值为 10,因此整个赋值表达式的值也为 10。

②首先计算赋值表达式"y=20"的值,得到结果 20,同时变量 y 被赋值为 20,然后计算 10+20 的值得到结果 30,再进行赋值运算,此时 x 的值为 30,整个表达式的值也为 30。

③先计算表达式"y=20"和"z=30"的值,得到结果分别为 20 和 30,然后计算表达式"10+20/30"的值得到结果为 10,再进行赋值运算,此时 x 的值为 10,整个表达式的值也为 10。

在进行赋值运算时,如果赋值运算符两侧操作数的数据类型不一致,系统会将右侧操作数的类型转换成左侧变量的数据类型,然后再进行赋值运算。具体情况如下:

(1)左侧变量和右侧操作数都是整型或字符型时,根据左侧变量和右侧操作数所占字节数的不同进行相应处理。假设左侧变量和右侧操作数分别为 a 位和 b 位,则具体处理方法如下:

当 a=b 时,直接赋值即可。

当 a<b 时,仅将右侧操作数低端的 a 位赋值给左侧变量即可。

当 a>b 时,将右侧操作数赋值给左侧变量低端的 b 位,高 a-b 位的处理方式是:如果左侧变量是无符号数或正数,那么全部补 0,否则全部补 1。

(2)右侧操作数为实型,左侧变量为整型或字符型,这时将操作数变为整型(舍去小数部

分),再赋值给左侧变量。例如,变量 x 为 int 型,则经过表达式“x=5.8”的赋值运算后,x 的值将是 5。

(3)右侧操作数为整型或字符型,左侧变量为实型,这时按照左侧变量的数据类型,将右侧操作数补足有效位后再赋值给左侧变量。例如,变量 f 为 float 型,则经过表达式“f='a'”的赋值运算后,f 的值将是 97.000 000。

2)复合赋值运算

在赋值运算符=之前可以加上算术运算符或移位运算符构成复合赋值运算符。C 语言规定可以使用 10 种复合赋值运算符,分别为+=、-=、*=、/=、%=、<<=、>>=、&=、^=和|=。本节只介绍前 5 种,其他将在后续章节进行详细介绍。

对复合的赋值表达式求值时,先将该运算符右侧的操作数与左侧的变量进行指定的复合运算,然后将计算结果赋值给左侧的变量,并作为该赋值表达式的值。复合赋值运算符的优先级与赋值运算符相同,结合性也是右结合。

【例 2-10】 假设变量 x=10,y=20,计算各赋值表达式的值。

①x+=10。

②x*=y+20。

③x+=x-=x / 10。

上述复合的赋值表达式的计算过程及结果如下:

①根据复合的赋值表达式的求解规则,“x+=10”等价于“x=x+10”,先将变量 x 的值与操作数 10 相加,得到计算结果为 20,然后进行赋值运算,将 20 赋给变量 x,此时整个表达式的值为 20,变量 x 的值也是 20。

②该复合的赋值表达式等价于“x=x*(y+20)”,首先将表达式“y+20”的值与变量 x 的值 10 相乘,然后将结果 400 赋值给变量 x。此时整个表达式的值为 400,变量 x 的值也是 400。

③由于赋值运算符的右结合性,该表达式等价于“x=(x+(x=x-(x/10)))”。因此,先计算表达式“x=x?(x/10)”的值,得到结果为 9,此时变量 x 的值也是 9。然后计算表达式“x=x+9”的值,得到结果为 18,此时整个表达式的结果为 18,变量 x 的值也是 18。

根据实际问题构造表达式时,采用复合赋值运算符有两点好处:一是可以简化程序的书写,使程序精炼;二是可以提高编译效率,产生质量较高的目标代码。

2.4.5　条件运算符及条件表达式

条件运算符(?:)是 C 语言中唯一的一个三目运算符,其目的是进行条件判断。

条件运算符的一般格式如下:

表达式 1? 表达式 2:表达式 3

“表达式 1”“表达式 2”和“表达式 3”的类型既可以是一个简单的表达式,又可以是由各种运算符组成的复合表达式。条件运算符的运算规则可以描述为:如果“表达式 1”的值为非 0,即逻辑真,那么运算结果等于“表达式 2”的值;否则,运算结果等于“表达式 3”的值,如图 2-7 所示。

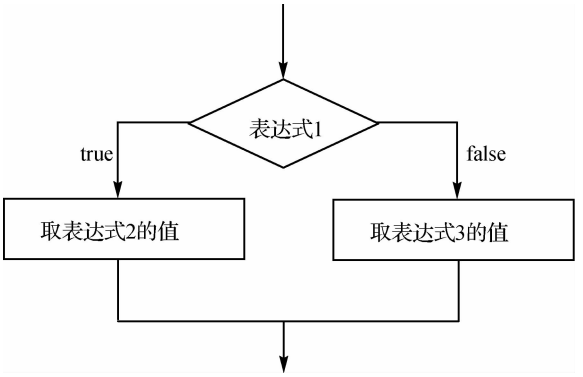

图 2-7　条件运算符的运算规则

条件运算符的优先级高于赋值运算符，但低于关系运算符和算术运算符。其结合性为右结合。

【例 2-11】 编写一个 C 语言程序，判断并输出用户输入的整数是奇数还是偶数。

```
//FileName: chap2_11.c
#include <stdio.h>
int main( )
{
    int n;
    printf("输入一个整数 n:");
    scanf("%d",&n);
    printf("%d 是一个 %s\n",n,(n%2==0 ? "偶数":"奇数"));
    return 0;
}
```

程序运行结果如下：

```
输入一个实数 n:5↙
5 是一个奇数
```

从功能上讲，后面章节介绍的选择结构的 if 语句可完全实现条件运算符的功能，但在某些简单情况下，使用条件运算符可使程序更加简洁。

2.4.6　逗号运算符及逗号表达式

C 语言提供一种特殊的运算符，即逗号运算符。其功能是用它将两个表达式连接起来组成一个表达式，称为逗号表达式。逗号运算符的结合性是自左至右，其优先级是最低的。

逗号表达式的一般形式如下：

表达式 1,表达式 2

逗号表达式的运算过程是：分别求两个表达式的值，并以表达式 2 的值作为整个逗号表达式的值。

【例 2-12】 逗号运算符举例。

```
//FileName: chap2_12.c
```

```
#include <stdio.h>
int main( )
{
    int a=2, b=4, c=6, x, y;
    y=((x=a+b), (b+c));
    printf("y= %d,x= %d\n\n", y, x);
    return 0;
}
```

程序运行结果如下：

```
y=10,x=6
```

在使用逗号运算符时应注意以下问题：

(1)逗号表达式一般形式中的“表达式 1”和“表达式 2”也可以是逗号表达式。例如，“表达式 1,(表达式 2,表达式 3)”形成了嵌套情形。因此，可以把逗号表达式扩展为以下形式：

表达式 1,表达式 2,…表达式 n

整个逗号表达式的值等于“表达式 n”的值。

(2)并不是在所有出现逗号的地方都组成逗号表达式，在变量说明中，函数参数表中的逗号只是作为各变量之间的分隔符。

2.4.7　sizeof 运算符及 sizeof 表达式

sizeof 是 C 语言中的一种单目运算符。从形式上讲，它与后面章节要讲解的函数很相似，但并不是函数。将 sizeof 运算符与操作数组合在一起构成的式子称为 sizeof 表达式。sizeof 运算符用来获得一个数据或数据类型在内存中所占空间的字节数。sizeof 表达式的一般形式如下：

```
sizeof(表达式)
```

或

```
sizeof(数据类型名)
```

【例 2-13】 编写一个程序获得当前使用的编译系统所分配的数据类型、常量、变量和表达式所占内存字节数。

```
//FileName: chap2_13.c
#include <stdio.h>
int main( )
{
    int a=2;
    printf("%d,", sizeof(int));
    printf("%d,", sizeof(long));
    printf("%d,", sizeof(float));
    printf("%d,", sizeof(double));
    printf("%d,", sizeof(char));
    printf("%d,", sizeof(6));
```

```
    printf("%d,", sizeof(a));
    printf("%d", sizeof(a+6));
    return 0;
}
```

程序运行结果如下：

4,4,4,8,1,4,4,4

需要注意的是，同一种数据类型在不同的编译系统中所占空间不一定相同。例如，在基于 16 位的编译系统中，int 型数据占用 2 字节，但现在普遍使用的是基于 32 位的编译系统，int 型数据要占 4 字节。Visual C++ 6.0 编译系统是 32 位。

到目前为止学过的 C 语言运算符的优先级与结合性如表 2-10 所示。

表 2-10　C 语言运算符的优先级与结合性

运 算 符	要求运算对象的个数	优 先 级	结 合 性
!,++,--,+,?,sizeof	1(单目运算符)	高 ↓ 低	右结合
×,/,%	2(双目运算符)		左结合
+,?	2(双目运算符)		左结合
<,<=,>,>=	2(双目运算符)		左结合
==,!=	2(双目运算符)		左结合
&&	2(双目运算符)		左结合
||	2(双目运算符)		左结合
?:	3(三目运算符)		右结合
=,+=,?=,*=,/=,%=	2(双目运算符)		右结合
,	2(双目运算符)		左结合

2.5　不同数据类型数据间的混合运算

很多情况下，在进行某种数值运算的过程中，会对操作数的数据类型进行转换，有些转换由系统自动进行，有些则由程序员人为指定。对于系统自动进行的类型转换通常要遵循一定的转换规则，如图 2-8 所示。

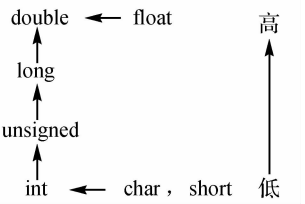

图 2-8　数值型数据在运算过程中的转换规则

图 2-8 体现了系统自动进行的两种类型转换：自动类型转换（也称隐式转换）和强制类型转换（也称显式转换）。

2.5.1　自动类型转换

自动类型转换发生在不同数据类型的量进行混合运算时，由编译系统自动完成。自动类型转换遵循以下规则：

（1）若参与运算量的类型不同，则先转换成同一类型，然后再进行运算。

（2）转换按数据长度增加的方向进行，以保证精度不降低。如 int 型和 long 型运算时，先把 int 型转成 long 型后再进行运算。

（3）所有浮点运算都是以双精度进行的，即使仅含有单精度量运算的表达式，也要先转换成 double 型再进行运算。

（4）char 型和 short 型量参与运算时，必须先转换成 int 型。

【例 2-14】 计算各表达式的值，注意数据类型转换的过程。

①10－3.0/2。

②5L＋8。

上述表达式的计算过程及结果如下：

①操作数 3.0 为 double 型数据，因此操作数 10 和 2 会由系统转换为 10.0 和 2.0 参与运算，最终得到的结果为 double 型数据 8.500 000。

②操作数 5L 为 long 型数据，因此操作数 8 会由系统转换为 8L 参与运算，最终得到的结果为 long 型数据 13L。

2.5.2　强制类型转换

除赋值转换外，系统自动进行的数据类型转换都是低精度类型向高精度类型的转换，如果需要将高精度类型的数值转换为低精度类型，必须由程序员人为指定，即强制转换。因此，强制类型转换需指定转换后的数据类型。

强制类型转换的一般形式如下：

（类型说明符）（表达式）

其功能是把表达式值的数据类型强制转换成类型说明符所指定的类型。例如，“(float)a”表示将变量 a 转换成 float 型，而“(int)(x＋y)”表示将表达式“x＋y”的值转换成 int 型。

使用强制类型转换时，需要注意以下问题：

（1）强制类型转换过程中有可能造成信息的丢失。例如，将实型数值转换为整型数值时会舍去小数部分。例如，表达式“(int)5.8f”的值是 5。

（2）如果被转换的数值在指定的结果类型中无法表示，那么虽然符合 C 语言的语法，这种转换也是没有意义的。例如，把 52 769.8 强制转换成 int 型，即(int)52 769.8，得到的结果为－12 767，一般来说没有意义。

（3）无论是强制类型转换还是自动类型转换，都只是为了本次运算需要而对变量的数据长度进行的临时性转换，而不改变数据说明时对该变量定义的类型，即原变量不会发生任何

变化。例如，假设 float 型变量 f 的值为 5.6f，进行强制类型转换后，得到一个 int 型的数据 5，但变量 f 的值仍然是 5.6f。

(4)C 语言将强制转换类型符看做单目运算符，单目运算符的优先级高于双目运算符。例如，表达式“(int)(5.6+8.8)×3”的值是 42，而表达式“(int)5.6+8.8×3”的值是 31.4。

【例 2-15】 强制类型转换举例。

```
//FileName: chap2_15.c
#include <stdio.h>
int main( )
{
    float x;
    int i;
    x=4.5;
        i=(int)x;  //将 x 临时强制转换成整型，离开本行 x 还是单精度型
        printf("x= %f,i= %d", x,i);
        return 0;
    }
```

程序运行结果如下：

```
x=4.500 000,i=4
```

资料
等级考试重难点

本 章 小 结

本章主要讲解了构成 C 语言程序的最基本要素，即标识符、关键字、常量、变量、运算符及表达式，使学习者对 C 语言程序设计的基本要素有一个全面认识。

C 语言提供了丰富的数据类型，通常可以将其分为两大类，分别是基本类型和构造类型。本章着重介绍了基本类型中的简单类型，主要包括整型、实型和字符型，并简要介绍了简单数据类型的长度、占用字节数和数值范围等。

C 语言中的数据有常量和变量之分，它们分别属于不同的数据类型，在程序运行过程中其值不能被改变的量称为常量，可以被改变的量称为变量。变量在使用过程中必须遵循“先定义，后使用”的原则，变量名的命名要符合标识符的命名规则，并且尽量做到见名知意，以增强程序的可读性。

为了完成复杂问题的求解，C 语言提供了运算符，包括算术运算符、关系运算符、逻辑运算符、赋值运算符、条件运算符、逗号运算符和求字节运算符。由这些运算符和圆括号将运算对象连接起来构成的符合 C 语言语法规则的式子称为表达式。不同类型的数值在运算过程中为了达到一致的数据类型需要经过数据类型转换，C 语言的数据类型转换分为自动类型转换和强制类型转换。

习　题　2

1)选择题

(1)选出可以作为 C 语言用户标识符的一组标识符(　　)。

A. void，define，WORD　　B. A3_B3，_123，abc

C. FOR，－abc，Case　　D. 2a，Do，Sizeof

(2)下列运算符优先级最高的是(　　)。

A. 关系运算符　　B. 赋值运算符

C. 算术运算符　　D. 逻辑运算符

(3)sizeof(float)是(　　)。

A. 一种函数调用　　B. 一个不合法的表示形式

C. 一个整型表达式　　D. 一个浮点表达式

(4)下列字符串常量不正确的是(　　)。

A. 'abc'　　B. "12'12"　　C. "0"　　D. " "

(5)下列 4 个选项均是合法整型常量的是(　　)。

A. 160，－0xffff，011　　B. －0xcdf，01a，0xe

C. －01，986，012，0668　　D. －0x48a，2e5，0x

(6)以下选项不属于 C 语言类型的是(　　)。

A. signed short int　　B. unsigned long int

C. unsigned int　　D. long short

(7)数值 029 是一个(　　)。

A. 八进制数　　B. 十六进制数

C. 十进制数　　D. 非法数

(8)在 C 语言中，要求运算数必须是整型的运算符是(　　)

A. /　　B. ＋＋　　C. !＝　　D. %

(9)当 c 的值不为 0 时，以下能将 c 的值赋给变量 a、b 的是(　　)。

A. c＝b＝a　　B. (a＝c)||(b＝c)

C. (a＝c)&&(b＝c)　　D. a＝c＝b

(10)若有说明语句“char c＝'\72';”，则变量 c(　　)。

A. 包含 1 个字符　　B. 包含 2 个字符

C. 包含 3 个字符　　D. 说明不合法，c 的值不确定

(11)设有说明语句“char w;int x;float y;double z;”，则表达式“w * x＋z－y”值的数据类型为(　　)。

A. float　　B. char　　C. int　　D. double

(12)长整型数据在内存中的存储形式是(　　)。

A. ASCII 码　　B. 原码　　C. 反码　　D. 补码

(13)字符型常量在内存中的存储形式是(　　)。

A. ASCII 码　　B. BCD 码　　C. 内部码　　D. 十进制码

(14)若 x、i、j 和 k 都是 int 型变量,则计算表达式"x=(i=4,j=16,k=32)"后 x 的值为(　　)。

A. 4　　B. 16　　C. 32　　D. 52

(15)若有代数式$\frac{3ae}{bc}$,则下列 C 语言表达式不正确的是(　　)。

A. a/b/c * e * 3　　B. 3 * a * e/b/c

C. 3 * a * e/b * c　　D. a * e/c/b * 3

(16)表达式"109!=99"的值是(　　)。

A. 1　　B. 非空值　　C. 0　　D. true

(17)若变量 t 为 double 类型,表达式"t=1,t * 5",则 t 的值为(　　)。

A. 1　　B. 6.0　　C. 2.0　　D. 1.0

(18)表示关系"x<=y<=z"的 C 语言表达式为(　　)。

A. (x<=y)&&(y<=z)　　B. (x<=y) AND (y<+z)

C. (x<=y<=y)　　D. (x<=y)&(y<=z)

(19)在 C 语言中,int、char 和 short 三种类型数据所占用的内存(　　)。

A. 均为 2 字节　　B. 由用户自己定义

C. 由所用机器的字长决定的　　D. 是任意的

(20)下列程序的执行结果是(　　)。

```
#include <stdio.h>
void main()
{
    int i=2;
    printf("%d,%d\n", ++i, --i);
}
```

A. 1, 1　　B. 2, 1　　C. 1, 2　　D. 2, 2

(21)字符串常量"\\\22a,0\n"的长度是(　　)。

A. 8　　B. 7　　C. 6　　D. 5

(22)下列叙述错误的是(　　)。

A. 在 C 语言程序中,逗号运算符的优先级最低

B. 在 C 语言程序中,APH 和 aph 是两个不同的变量

C. 若变量 a 和 b 的类型相同,在计算了赋值表达式"a=b"后,b 中的值将放入 a 中,而 b 中的值不变

D. 当从键盘输入数据时,对于整型变量只能输入整型数值,对于实型变量只能输入实型数值

(23)下列叙述正确的是(　　)。

A. C 语言中既有逻辑类型也有集合类型

B. C 语言中没有逻辑类型但有集合类型

C. C 语言中有逻辑类型但没有集合类型

D. C 语言中既没有逻辑类型也没有集合类型

2)填空题

(1)已知“int x=6;”,则执行语句“x+=x--=x*x;”后 x 的值是_____。

(2)若 w=1, x=2, y=3, z=4, 则条件表达式“w>x?w:y<z? y:z”的结果是_____。

(3)若“int m=5,y=2;”,则计算表达式“y+=y--=m*=y”后 y 的值是_____。

(4)在 C 语言中,一个 int 型数据在内存中如果占 2 字节,则 int 型数据的取值范围为_____。

(5)已知字母 a 的 ASCII 码为十进制数 97,且设 ch 为字符型变量,则表达式“ch='a'+'8'-'3'”的值为_____。

(6)若 x 和 n 均为 int 型变量,且 x 和 n 的初值均为 5,则计算表达式“x+=n++”后 x 的值为_____,n 的值为_____。

(7)若有定义“int a=2, b=3;float x=3.5,y=2.5;”,则表达式“(float)(a+b)/2+(int)x%(int)y”的值为_____。

(8)下列程序的输出结果是_____。

```
#include <stdio.h>
int main()
{
    int k=2,i=2,m;
    m=(k+=i*=k);
    printf("%d,%d\n",m,i);
    return 0;
}
```

(9)以下程序的执行结果是_____。

```
#include <stdio.h>
int main()
{
    int a,b,c;
    a=b=1;
    c=a++-1;
    printf("%d,%d,", a, c);
    c+=-a+++(++b||++c);
    printf("%d,%d\n",a,c);
    return 0;
}
```

(10)以下程序的执行结果是_____。

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

```
{
    int n=023;
    printf("%d\n",--n);
    return 0;
}
```

(11)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    short int i=10;
    printf("%d,%d,%d",--i,--i,i--);
}
```

(12)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    short int a=-32768, b;
    b=a-1;
    printf("a=%d,b=%d", a, b);
}
```

(13)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    int x=042, y=067, z;
    z=(x>>2) & (y<<3);
    printf("%d\n",z);
}
```

(14)以下程序的执行结果是_____。

```
#include <stdio.h>
void main()
{
    int x=10,y=9;
    int a,b,c;
    a=(--x==y++)?--x:++y;
    b=x++;
    c=y;
    printf("%d,%d,%d\n",a,b,c);
}
```

(15)以下程序输入 123456789,其输出结果是_____。

```
#include <stdio.h>
void main()
{
    int a,b;
    float f;
    scanf("%2d%*2d%2d%f", &a, &b, &f);
    printf("%d,%d,%f\n", a,b,f);
}
```

第 3 章　控 制 结 构

C 语言是结构化的程序设计语言。结构化的程序通常包括数据的描述和操作的描述两方面内容。数据的描述是指程序中数据的类型和组织形式，即数据结构。前面介绍的数据类型、常量、变量及后续章节要介绍的数组、结构体、共用体等都属于这方面的内容。操作的描述是指程序中对数据的操作方法和操作步骤，即算法。数据的描述和操作的描述是程序设计过程中必不可少的组成部分。著名的瑞士计算机科学家 N. 沃思提出了“数据结构＋算法＝程序”的思想。

本章将介绍算法的概念及描述工具，重点介绍结构化程序设计的三大控制结构，即顺序结构程序设计、选择结构程序设计和循环结构程序设计。

3.1　算法的基本知识

从广义上来讲，算法是解决某一问题的方法和步骤，狭义的算法指的是计算机算法，即对特定问题求解步骤的一种描述，它是计算机指令的有限序列，其中每一条指令表示计算机可以进行的一个或多个操作。程序设计时应认真分析问题，找出合适的算法和数据结构。解决同一问题的算法可能有很多，但它们的效率却可能相差很多，选择合适的算法可能会大大降低程序设计的复杂程度，提高程序运行的效率和存储效率。

3.1.1　算法的特性和要素

1)算法的特性

计算机所能执行的算法必须具备以下几个特性：

(1)有穷性。算法是一个有穷的计算机操作序列，即计算机可以按照算法的规定从一个

唯一的初始动作开始，经过有限次数的操作后终止。

(2)确定性。算法中的每个操作应执行何种动作必须是确定的，即无二义性，且每个操作都只有一个后继操作。对于一组给定的数据，同一个算法对应的程序在计算机上的执行过程是可以再现的，执行结果也是正确的。

(3)可行性。算法中规定的每个操作都是计算机可以执行的基本操作。

(4)输入。一个算法可以有0个或多个输入，即算法中要用到一组初始数据，可以在算法中确定，也可以在程序运行时由用户通过输入设备输入计算机中。

(5)输出。一个算法必须产生一个或多个输出，即程序运行时将产生一组与输入的初始数据相对应的输出数据。一个没有输出的算法是没有任何意义的。

2)算法的要素

一个计算机所能执行的算法必须具备以下两个要素：

(1)基本操作。计算机的基本操作主要包括算术运算、关系运算、逻辑运算、函数运算、位运算及I/O操作等。由于不同计算机语言所对应的操作集略有不同，所以在设计算法时，应首先确定编程语言。

(2)控制结构。每个算法都由一系列操作组成。同一操作序列，按不同的顺序执行，就会得到不同的结果。控制结构就是如何控制组成算法的各操作的执行顺序。一个算法只能由3种结构组成，即顺序结构、选择结构和循环结构。

3.1.2　算法的描述

算法的描述方法有很多种，最常用的有自然语言、伪代码、流程图、N-S图、PAD和程序设计语言等。下面主要介绍如何用自然语言、流程图、N-S图和程序设计语言来描述算法。

1)自然语言

自然语言是人们在日常生活、工作和学习中的通用语言，一般不需要专门学习和训练就能理解这种语言所表达的意思。但用自然语言描述程序(或算法)的流程时，一般要求直接而简练，尽量减少语言上的修饰。例如，用自然语言描述sum=1+2+3+…+99+100的算法如下：

(1)定义变量sum，并置初始值为0。

(2)计算1+2+3+…+99+100的和，并将结果赋给变量sum。

(3)输出变量sum的值。

2)流程图

流程图是一种传统的算法表示方法，用图框表示各种操作，用流程线表示操作的执行顺序。用图形表示算法，直观形象、易于理解。ANSI(American National Standard Institute)规定了一些常用的流程图符号，如图3-1所示。

描述sum=1+2+3+…+99+100算法的流程图如图3-2所示。

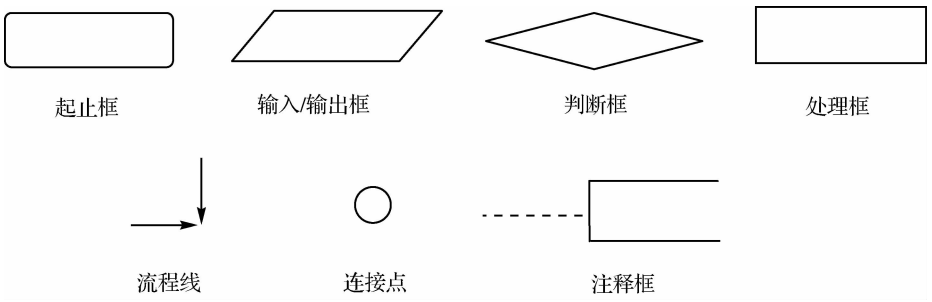

图 3-1　常用流程图符号

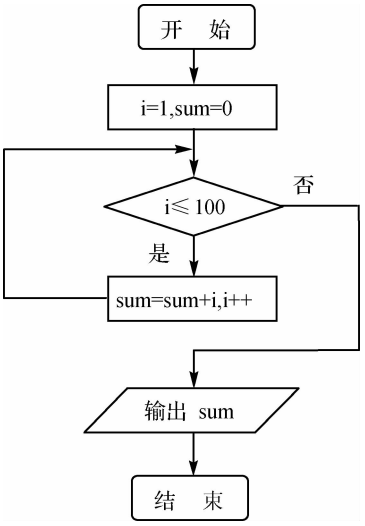

图 3-2　计算 sum＝1＋2＋3＋…＋99＋100 的流程图

3)N-S 图

N-S 图是由 I. Nassi 和 B. Shneiderman 于 1973 年共同提出的一种结构化描述方法。在这种流程图中,去掉了所有流程线,算法写在一个矩形框内,在该矩形框内还可以包含其他矩形框。N-S 流程图的符号如图 3-3 至图 3-7 所示。

(1)顺序结构。顺序结构用图 3-3 所示形式表示。A 和 B 两个框组成一个顺序结构。

(2)选择结构。选择结构用图 3-4 所示形式表示。当条件 p 成立时执行 A 操作,不成立时执行 B 操作。

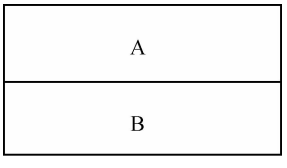

图 3-3　顺序结构

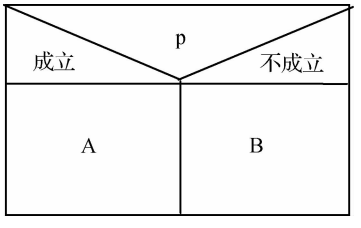

图 3-4　选择结构

(3)循环结构。“当”型循环结构用图 3-5 所示形式表示。当条件 p 成立时反复执行 A 操作,直到条件不成立为止。“直到”型循环结构用图 3-6 所示形式表示。

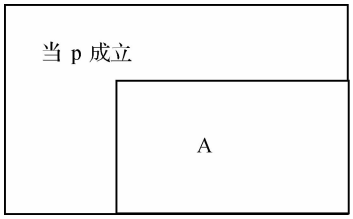

图 3-5　“当”型循环结构

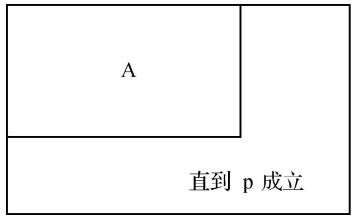

图 3-6　“直到”型循环结构

描述 sum＝1＋2＋3＋…＋99＋100 算法的 N-S 图如图 3-7 所示。

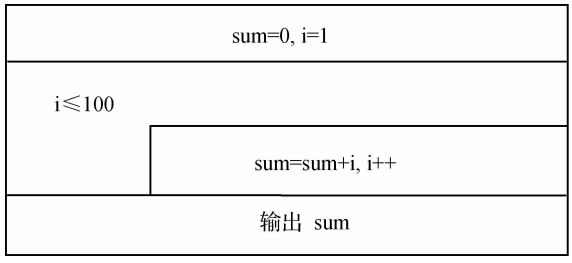

图 3-7　计算 sum＝1＋2＋3＋…＋99＋100 的 N-S 图

4)程序设计语言

对于采用自然语言、流程图和 N-S 图描述的算法，计算机是不能执行的。要让计算机执行一个算法，必须把该算法转换成计算机语言。用计算机语言表示算法必须严格遵循所使用语言的语法规则。例如，sum＝1＋2＋3＋…＋99＋100 算法的 C 语言描述如下所示。

【例 3-1】　编程求 1＋2＋3＋…＋99＋100 的累加和。

```
//FileName: chap3_1.c
#include <stdio.h>
int main( )
{
    int i,sum;
    i=1; sum=0;
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    printf("The sum is %d\n",sum);
    return 0;
}
```

程序运行结果如下：

```
The sum is 5050
```

为了使读者熟练掌握常用的算法描述方式，本书中的例题将根据内容的不同，采用自然

语言、流程图和 N-S 图方式等不同的算法描述方式。

3.2 顺序结构程序设计

真题测试

顺序结构是 3 种结构中最简单的一种，语句按照位置的先后顺序执行，处理流程通常是先输入参数，然后完成相应的计算和处理，最后输出相应结果。

C 语言中的语句可以归纳为 4 类，分别是表达式语句、函数调用语句、空语句和复合语句。

3.2.1 表达式语句

表达式语句由表达式加上分号组成。其一般形式如下：

```
表达式;
```

这里的表达式是指 C 语言中任何合法的表达式。举例如下：

```
x=y+z; //赋值表达式语句
y+z;    //加法运算语句,但计算结果不能保留,无实际意义
i++;    //自增 1 语句,即 i 的值增 1
```

赋值语句的功能和特点都与赋值表达式相同。它是程序中使用最多的语句之一，使用赋值语句时需要注意以下几点：

(1)在赋值符号右边的表达式也可以是一个赋值表达式。例如，“a=b=c=5;”按照赋值运算符的右结合性，此式实际等效于：

```
c=5;
b=c;
a=b;
```

(2)注意赋值表达式和赋值语句的区别。赋值表达式可以出现在任何允许表达式出现的地方，而赋值语句则不能。例如，下述语句是合法的。

```
if((x=y+5)>0) z=x;
```

该语句的功能是：若表达式“x=y+5”大于 0，则 z=x。

而下述语句是非法的。

```
if((x=y+5;)>0) z=x;
```

因为“x=y+5;”是语句，所以不能出现在表达式中。

3.2.2 函数调用语句

函数是 C 语言程序的基本组成单位，一个函数的执行是通过在程序中调用这个函数来实现的，调用函数的操作由 C 语言语句来完成，通常称为函数调用语句。其一般形式如下：

```
函数名(参数表);
```

例如，“printf("hello!");”就是一个函数调用语句。关于函数的具体内容将在后续章节中介绍，这里仅简单介绍实现输入/输出操作的C语言库函数。

一个完整的计算机程序应具备输入和输出功能，C语言本身并不提供数据的输入/输出语句，有关的输入/输出操作都是通过调用C标准库函数来实现的。C语言提供的输入/输出标准库函数有getchar( )、putchar( )、scanf( )和printf( )等。

需要注意的是，引用C语言标准库函数时，必须用编译预处理命令“#include”将头文件“stdio.h”包含到用户源程序中，即在程序的开始写一行命令#include <stdio.h>或者#include "stdio.h"。

1)字符输入函数 getchar( )

字符输入函数的一般形式如下：

```
int getchar()
```

功能：接收从终端输入的一个字符，并返回其ASCII码值。

举例如下：

```
int ch=getchar( );  //从输入终端(如键盘)接收一个字符并把它赋给变量ch
```

注意：函数getchar()只能接收单个字符，因此尽管用户可以通过终端输入多个字符，但该函数仅接收第一个字符，并返回该字符的ASCII码值，其他字符不予处理。

2)字符输出函数 putchar( )

字符输出函数的一般形式如下：

```
int putchar(char ch)
```

功能：向终端输出一个字符，并返回该字符的ASCII码值。

【例3-2】 分析以下程序的运行结果，注意函数getchar()和putchar()的使用。

```
//FileName: chap3_2.c
#include <stdio.h>
int main()
{
    char ch1,ch2,ch3,ch4;          //定义4个变量用于接收输入的4个字符
    ch1=getchar();
    ch2=getchar();
    ch3=getchar();
    ch4=getchar();
    putchar(ch1);
    putchar('\n');
    putchar(ch2);
    putchar('\n');
    putchar(ch3);
    putchar('\n');
    putchar(ch4);
    return 0;
```

```
}
```

程序运行结果如下：

```
C++↙
C
+
+
```

说明：函数 putchar( )和 getchar( )只能处理单个字符。回车符也被看做一个普通字符存储在变量 ch4 中。

函数 putchar( )不仅可以使用字符变量和字符常量完成字符的输出，如【例 3-2】中的语句“putchar(ch1);”和“putchar('\n');”，这里的后一个语句输出了一个回车符，从而实现了换行功能。还可以直接使用 ASCII 码值输出对应的字符，如语句“putchar(65);”输出一个 ASCII 码值为 65 的字符'A'。

3)格式输出函数 printf( )

字符输出函数 putchar( )仅能输出单个字符，对多个字符进行输出处理时，要使用格式输出函数 printf( )，并且格式输出函数还可用于输出任意类型的数据。

函数 printf( )的一般形式如下：

```
int printf(格式控制字符串,输出列表)
```

功能：按照格式控制要求，向终端输出列表中各个输出项的值。其中，格式控制字符串用于指定数据的输出格式，是由双撇号括起来的字符串，该字符串中除格式说明符外的所有字符将原样输出，输出列表列出了各个输出项。其中，格式控制字符串中的格式说明符和输出列表中的输出项在数量和类型上要一一对应。

举例如下：

```
printf("The cal is %d+%d=%f", num1,num2,sum);
```

这里%d 和%f 是格式说明符，上述语句中共有 3 个格式说明符，分别对应输出列表中的 3 个变量。格式控制字符串中的"The cal is"、"+"、"="直接在显示器上输出。

C 语言中常用的格式说明符有 4 种，分别是整型数据的格式说明符、实型数据的格式说明符、字符型数据的格式说明符和字符串数据的格式说明符。

(1)整型数据的格式说明符。

①%d，以十进制形式输出一个整数。

②%u，以十进制形式输出一个无符号整数。

③%o，以八进制形式输出一个整数。

④%x，以十六进制形式输出一个整数。

⑤%ld，以十进制形式输出一个长整数。

⑥%md，以指定宽度输出一个整数。m 为指定的输出宽度(包括符号位)，如果数据的实际位数小于 m，则左端补空格，如果大于 m，则按照实际位数输出。

⑦ %－md，以指定宽度输出一个整数。m 为指定的输出宽度(包括符号位)，如果数据的实际位数小于 m，则右端补空格，如果大于 m，则按照实际位数输出。

【例 3-3】 分析以下程序的运行结果，注意格式说明符的使用。

```
//FileName: chap3_3.c
```

```
#include <stdio.h>
int main()
{
    int a=15;
    printf("%d,%o,%x\n", a, a, a);
    printf("%5d\n", a);
    return 0;
}
```

程序运行结果如下：

```
15,17,f
   15
```

说明：第1个printf( )函数分别采用十进制、八进制和十六进制3种形式输出整数a的值，得到的结果分别为15,17,f,在第2个printf( )函数中，通过%5d指定以5个字符的宽度输出整数a的值，而a的值为15,仅占两个字符的位置，所以输出时整数15的前面有3个空格。

(2)实型数据的格式说明符。

①%f,输出一个单精度实数，该实数的整数部分按实际位数输出，小数部分保留6位。如果输出一个双精度实数，那么格式说明符为%lf,即在字符"f"之前加上字符"l"。

②%m.nf,指定以m个字符的宽度输出一个实数，其中小数位数为n位。如果数据的实际位数小于m,那么左端补空格。如果大于m,那么按照实际位数输出。小数位数小于n,则右端补0。小数位数大于n,则采用四舍五入方式进行处理。

③%－m.nf,与%m.nf基本相同，不同的是输出的数据是向左靠拢，右端补空格。

④%e,以指数形式输出一个实数。

⑤%g,根据数值的大小在f格式或e格式中占宽度较小的一种输出，并且不输出无意义的零。

需要强调的是，由于实数都有有效数字位数的限制，当输出的实数位数超出了允许的有效数字范围时，输出的实数可能与原数值不同。

【例3-4】 分析以下程序的运行结果，注意格式说明符的使用。

```
//FileName: chap3_4.c
#include <stdio.h>
int main()
{
    float x;
    x=12345.129871;
    printf("%f\n", x);
    printf("%11.8f\n", x);
    printf("%11.2f\n", x);
    return 0;
}
```

程序运行结果如下：

```
12345.129883
12345.12988281
  12345.13
```

说明：由于输出的实数为 float 型，有效数字为 7 位，因此，输出的实数中仅有 7 位数字是有效的，并不是所有打印出来的都是有效数字。对于第三条输出语句，小数位数设置为 2，因此，对实数 12 345.129 871 进行四舍五入后输出的结果为 12 345.13。

(3)字符型数据的格式说明符。

输出字符型数据的格式说明符为%c 表示输出一个字符。

【例 3-5】 分析以下程序的运行结果，注意格式说明符的使用。

```
//FileName: chap3_5.c
#include <stdio.h>
int main()
{
    char c='A';
    int a=65;
    printf("%c\n", c);
    printf("%c\n", a);
    printf("%d\n", a);
    return 0;
}
```

程序运行结果如下：

```
A
A
65
```

说明：当格式说明符为%c 时，整型变量 a 的值 65 被看做相应字符的 ASCII 码值，因此输出对应的字符'A'。

(4)字符串数据的格式说明符。

输出字符串的格式说明符为%s，表示输出一个字符串。

【例 3-6】 分析以下程序的运行结果，注意格式说明符的使用。

```
//FileName: chap3_6.c
#include <stdio.h>
int main()
{
    printf("%s","Welcome to C programming world! \n");
    return 0;
}
```

程序运行结果如下：

```
Welcome to C programming world!
```

4)格式输入函数 scanf( )

函数 scanf( )的一般形式如下：

int scanf(格式控制字符串,地址列表)

功能：按照格式控制的要求，将从终端输入的数据赋值给地址列表中的各个变量。格式控制字符串的含义和函数 printf( )中格式控制字符串的含义类似，地址列表列出了各变量的地址，由取地址运算符 & 后跟变量名组成。

【例 3-7】 分析以下程序的执行结果，注意函数 scanf( )的使用。

```
//FileName: chap3_7.c
#include <stdio.h>
int main()
{
    int a, b, temp;
    printf("Please input a b:\n");
    scanf("%d %d", &a, &b);
    temp=b;
    b=a;
    a=temp;
    printf("a=%d,b=%d\n", a, b);
    return 0;
}
```

程序运行结果如下：

```
Please input a b:
2 3↙
a=3,b=2
```

说明：scanf( )函数中的两个格式说明符%d之间是一个空格符，因此，在输入 a 和 b 的值时用空格作为间隔符，当用户输入 2 和 3 并按回车键结束输入过程后，变量 a 和 b 分别被赋值为 2 和 3。程序中的三条语句“temp=b; b=a; a=temp;”是典型的交换两个变量值的方法，因此，最后使用 printf( )函数输出变量 a 和 b 的值时，结果为“a=3, b=2”。

需要强调的是，如果两个格式说明符之间没有分隔符，如“%d%d”，那么在输入时要注意用空格、回车或 Tab 键作为输入数据之间的间隔。例如，对于输入语句“scanf("%d%d%d", &a, &b, &c);”，下面的输入方式：

①1 2 3↙

②1↙

2↙

3↙

③1(按 tab 键)2↙

3↙

都是正确的。

如果格式控制字符串中出现了除格式说明符之外的其他字符，那么在进行数据输入时

也要在对应的位置输入该字符。

举例如下：

```
scanf("%d,%d,%d", &a, &b, &c);  // 字符“,”作为间隔符,输入形式为:5,6,7↙
scanf("a=%d,b=%c", &a, &b);     // 输入形式为:a=5,b=7↙
```

如果不采用上述输入形式，虽然不会出现编译错误，但会导致错误的运行结果。

3.2.3 空语句

仅由分号组成的语句称为空语句。空语句是不执行任何操作的语句。

例如，

```
while(getchar( )!='\n')
{
    ;    //循环体为空语句
}
```

空语句通常起占位作用，在程序没有完全开发完成前，可用空语句占位，以便后续开发再填充代码。

3.2.4 复合语句

把多个语句用大括号括起来组成的语句称为复合语句。在语法上，复合语句相当于单条语句，而不是多条语句。其一般形式为：

```
{
    语句 1;
      ⋮
    语句 n;
}
```

复合语句可以放在能够使用语句的任何地方，它建立一个新的作用域或块。复合语句是 C 语言中唯一不用分号结尾的语句。它在分支结构、循环结构中使用十分广泛。

【例 3-8】 输入摄氏温度 c 的值，计算华氏温度 f 的值(计算公式为 f=9*c/5+32)。

分析：摄氏温度 c 和华氏温度 f 之间具有 f=9*c/5+32 的关系，因此只要给定 c 的值，代入上述公式即可求得相应的 f 值。考虑到实际情况，c 和 f 均应定义为 float 类型。

```
//FileName: chap3_8.c
#include <stdio.h>
int main()
{
    float c,f;
    printf("Please input c:");
    scanf("%f", &c);
    f=9*c/5+32;
    printf("f=%6.2f\n", f);
```

```
    return 0;
}
```

程序运行结果如下：

```
Please input c: 28↙
f=82.40
```

【例 3-9】 利用输入的三角形的 3 条边长，求三角形的面积。

分析：假设三角形的 3 条边长分别为 a、b、c，则面积 area=sqrt(s*(s-a)*(s-b)*(s-c))，其中 s=(a+b+c)/2，从输入设备接收到 3 条边长之后，即可通过上述公式求解。

```
//FileName: chap3_9.c
#include <math.h>
#include <stdio.h>
int main()
{
    float a,b,c,area,s;
    printf("Please input a,b,c:");
    scanf("%f,%f,%f", &a,&b,&c);
    s=1.0/2*(a+b+c);
    area=sqrt(s*(s-a)*(s-b)*(s-c));
    printf("a=%.2f,b=%.2f,c=%.2f\n", a,b,c);
    printf("area=%.2f\n",area);
    return 0;
}
```

程序运行结果如下：

```
Please input a,b,c: 3.0,4.0,5.0↙
a=3.00,b=4.00,c=5.00
area=6.00
```

说明：因为程序中用到了 math.h 头文件中的开方函数 sqrt( )，所以在程序起始位置通过预编译命令#include 将 math.h 文件包含进程序中。程序中第 11 和 12 行语句中的%.2f 表示结果输出两位小数。

【例 3-10】 输入一个英文小写字符，将它转化为英文大写字符并输出。

分析：英文大小写字符的 ASCII 码值相差 32，因此当输入一个英文小写字符时，只要减去 32 即可得到对应字符的大写字符。

```
//FileName: chap3_10.c
#include <stdio.h>
int main()
{
    char ch;
    printf("Please input a lower case letter:");
    ch=getchar();
```

```
    ch=ch-32;
    putchar(ch);
    printf("\n");
    return 0;
}
```

程序运行结果如下：

```
Please input a lower case letter: g↙
G
```

3.3 选择结构程序设计

思维导图

在顺序结构中，出现在程序中的所有语句都会顺序执行，不能跳转，但在解决某些实际问题时，可能涉及多种互斥性的操作，对于这些操作，满足一定的条件执行一定的操作，满足另外的条件执行另外一种操作，这就需要用到选择结构。

C 语言提供两种选择结构语句，即 if 语句和 switch 语句。

3.3.1 if 语句

if 语句是实现选择结构最常用的语句，其作用是根据给定的条件，判断执行哪些语句，要执行的语句可能有一条或多条。if 语句包括基本 if 语句、双分支 if 语句、多分支 if 语句和嵌套 if 语句 4 种形式。

1)基本 if 语句

该语句的一般形式如下：

```
if(表达式)
    语句;
```

功能：如果表达式的值为真，那么执行其后的语句，否则不执行该语句。执行过程如图 3-8 所示。

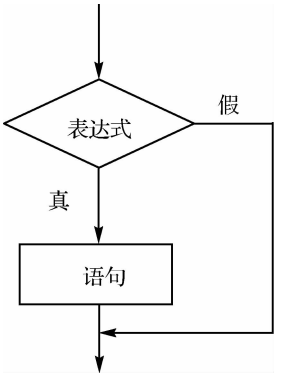

图 3-8　基本 if 语句的执行过程

【例 3-11】 输入两个数，输出其中较大的那个数。

分析：定义两个变量 a、b，用于接收用户输入的两个数。首先假设 a 的值较大，将其赋给最大值变量 max，然后将 b 与 max 比较，如果 b 大于 max，那么将 b 的值赋给 max，最后输出 max 的值即为最大值。

```
//FileName: chap3_11.c
#include <stdio.h>
int main( )
{
    float a,b,max;                    //定义 a、b、max 为基本整型变量
    scanf("%f,%f",&a,&b);
    max=a;
    if(b>max)
    max=b;
    printf("max is %f\n",max);
    return 0;
}
```

程序运行结果如下：

```
1,2↙
max is 2.000000
```

需要强调的是，if 语句中的表达式可以是任何合法的 C 表达式，语句可以是任何合法的 C 语句。例如，可以是表达式语句、函数调用语句、空语句、复合语句等。

2)双分支 if 语句

这是 if 语句比较完整的形式，包含一条 if 子句和一条 else 子句。该语句的一般形式如下：

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

功能：若表达式的值为真，则执行语句 1，否则执行语句 2。执行过程如图 3-9 所示。

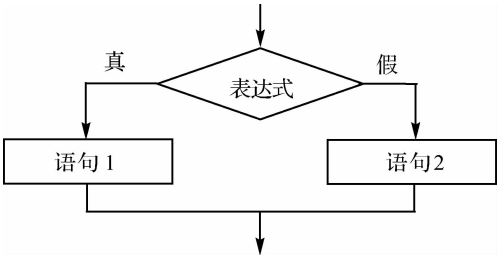

图 3-9　双分支 if 语句执行过程

【例 3-12】 输入学生成绩，若大于或等于 60 分，则打印 passed，否则打印 not passed。

分析：定义一个变量 score，用于接收用户输入的学生成绩，比较 score 和 60 的关系，若存在大于等于关系，则输出 passed，否则输出 not passed。

```
//FileName: chap3_12.c
#include <stdio.h>
int main( )
{
    float score;
    printf("Please input a score:");
    scanf("%f",&score);
    if(score>=60)
        printf("passed\n");
    else
        printf("not passed\n");
    return 0;
}
```

程序运行结果如下：

```
Please input a score:80↙
passed
```

3)多分支 if 语句

前两种形式的 if 语句一般用于两个分支的情况，当有多个分支可供选择时，可采用多分支 if 语句。该语句的一般形式如下：

```
if(表达式 1)
    语句 1;
else if(表达式 2)
    语句 2;
else if(表达式 3)
    语句 3;
⋮
else if(表达式 n)
        语句 n;
else
    语句 n+1;
```

功能：依次判断表达式的值，当出现某个表达式的值为真时，则执行其对应的语句，然后跳转到整个 if 语句之后继续执行程序。如果所有的表达式的值均为假，那么执行 else 后的语句，即语句 n+1。执行过程如图 3-10 所示。

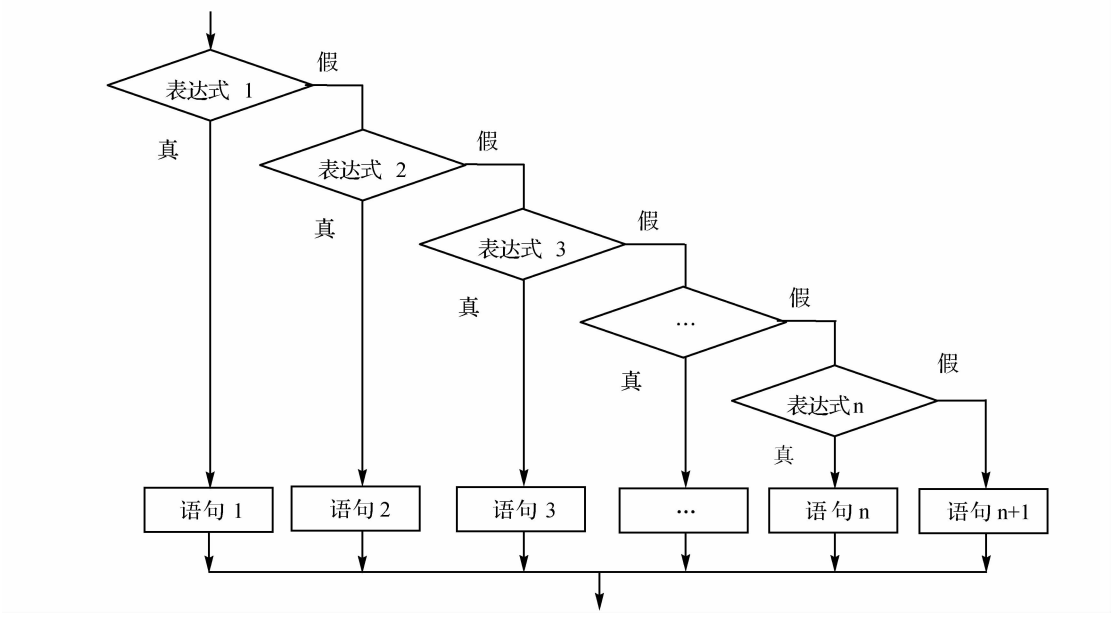

图 3-10　多分支 if 语句执行过程

【例 3-13】 输入一个百分制成绩，要求输出对应的成绩等级。百分制成绩与成绩等级的对照关系如下。

90 分～100 分为等级 A，80 分～89 分为等级 B，70 分～79 分为等级 C，60 分～69 分为等级 D，0 分～59 分为等级 E。

分析：定义一个变量 score，用于接收用户输入的百分制成绩，然后判断该成绩满足哪个条件，从而输出对应的五分制成绩。

```
//FileName: chap3_13.c
#include <stdio.h>
int main( )
{
    float score;
    char grade;
    printf("Please input a score:");
    scanf("%f",&score);
    if(score>=90 && score<=100)
    grade='A';
    else if(score>=80 && score<=89)
    grade='B';
    else if(score>=70 && score<=79)
    grade='C';
    else if(score>=60 && score<=69)
    grade='D';
    else if(score>=0 && score<=59)
```

```
    grade='E';
    else
    {
        printf("Your score is wrong!\n");
        grade='0';
    }
    printf("score is %.2f, grade is %c\n",score, grade);
    return 0;
}
```

程序运行结果如下：

```
Please input a score:82 ↙
score is 82.00, grade is B
```

4)嵌套 if 语句

当 if 语句的操作语句中包含其他 if 语句时，称为嵌套 if 语句。该语句的基本形式如下：

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

功能：如果表达式 1 的值为真(非 0)，那么执行 if 子句的内嵌 if 语句，否则执行 else 子句的内嵌 if 语句。内嵌 if 语句就是 if 子句和 else 子句的操作语句，可以是上述 if 语句 3 种形式中的任意一种。

在嵌套 if 语句结构中，一定要注意 else 与 if 之间的对应关系。在 C 语言中规定的对应原则是：else 总是与它前面最近的一个尚未匹配的 if 相匹配。

一般在书写程序时应注意对应的 if 和 else 对齐，将内嵌的语句缩进，这样可增加程序的可读性和可维护性，但要特别注意的是，C 语言编译系统并不是按缩进的格式来查找 else 与 if 之间的对应关系的，它只是按“else 总是与它前面最近的一个尚未匹配的 if 相匹配”这一基本原则来查找 else 与 if 之间的对应关系。如果使用了错误的对齐格式，只会起到误导读者的作用，并不会影响程序的执行结果。

以下面的语句段为例：

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

```
    语句 2;
```

它的实际意义与程序编写者的想法是不同的。从程序编写者的想法看，第一个 else 应该与第一个 if 对应，但事实上 else 是与第二个 if 配对的，因为它们相距最近。这样就会造成程序混乱，甚至导致出错。因此在使用时，应通过加大括号的方法来确定配对关系。

可以把上面的例子改成如下形式：

真题测试

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

这样，{ }限定了嵌套 if 语句的范围，此时 else 与第一个 if 相对应。

【例 3-14】 输入 3 个任意整数，找出其中的最大值。

分析：求解该题可采用嵌套 if 语句形式。定义 4 个整型变量 a、b、c、max，其中，a、b、c 用于接收 3 个任意整数，max 用于存放最大值，其算法描述如图 3-11 所示。

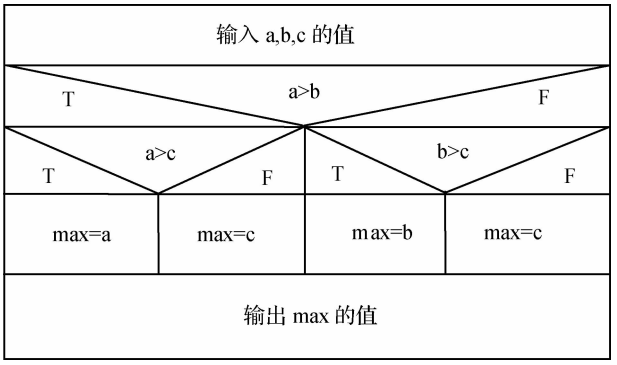

图 3-11　算法描述的 N-S 图

```
//FileName: chap3_14.c
#include <stdio.h>
int main( )
{
    int a,b,c,max;
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
        if(a>c)
            max=a;
        else
            max=c;
    else
        if(b>c)
```

```
        max=b;
    else
        max=c;
    printf("max= %d\n",max);
    return 0;
}
```

程序运行结果如下：

```
12  10  21↙
max=21
```

【例 3-15】 编写程序，输入一个 x 的值，按以下函数计算并输出 y 的值。

$$y=\begin{cases}-1 & (x<0)\\ 0 & (x=0)\\ 1 & (x>0)\end{cases}$$

分析：该题属于分段函数求解问题，y 的值根据 x 的取值而定。关于分段函数求解问题的算法描述如图 3-12 所示。

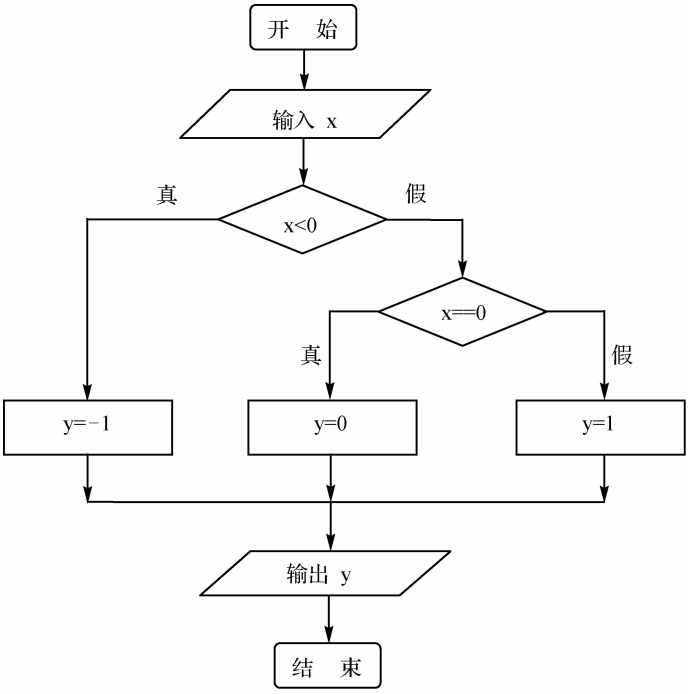

图 3-12 分段函数算法流程图

```
//FileName: chap3_15.c
#include <stdio.h>
int main( )
{
    int x,y;
    scanf("%d",&x);
```

```
    if(x<0)
        y=-1;
    else
        if(x==0)
            y=0;
        else
            y=1;
    printf("y= %d\n",y);
    return 0;
}
```

程序运行结果如下：

```
10↙
y=1
```

【例 3-16】 求任意一个一元二次方程 $ax^2+bx+c=0$ 的解。

分析：解一元二次方程有以下几种不同的情况。

①$a=0$，不是二次方程。

②$b^2-4ac>0$，方程有两个不相同的实根。

③$b^2-4ac<0$，方程有两个共轭复根。

④$b^2-4ac=0$，方程有两个相等的实根。

算法流程图如图 3-13 所示。

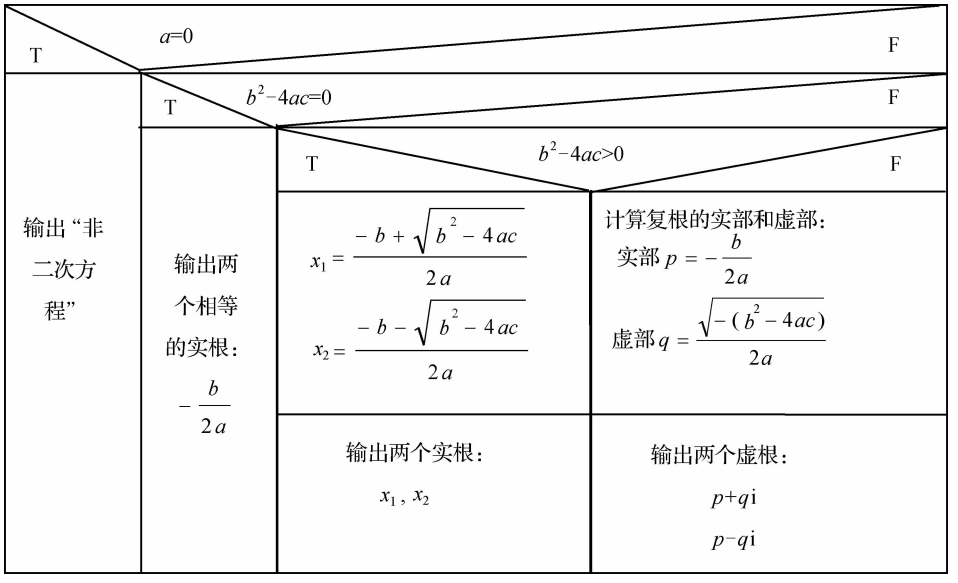

图 3-13　解一元二次方程算法的 N-S 流程图

```
//FileName: chap3_16.c
#include <stdio.h>
#include <math.h>
```

```
int main( )
{
    float a,b,c,d,x1,x2,p,q;
    scanf(" %f, %f, %f",&a,&b,&c);
    if(a==0)
        printf("Input error!");
    else
    {
        d=b*b-4*a*c;
        if(d==0)
            printf("x= %f\n",-b/(2*a));
        else if(d>0)
        {
            x1=(-b+sqrt(d))/(2*a);
            x2=(-b-sqrt(d))/(2*a);
            printf("x1= %.2f, x2= %.2f\n",x1,x2);
        }
        else
        {
            p=-b/(2*a);
            q=sqrt(-d/(2*a));
            printf("x1= %.2f+ %.2fi\n",p,q);
            printf("x2= %.2f- %.2fi\n",p,q);
        }
  }
}
```

程序运行结果如下：

```
1,2,3↙
x1=-1.00+2.00i
x2=-1.00-2.00i
```

3.3.2 switch 语句

【例 3-13】采用了多分支 if 语句来实现成绩的等级分类，使程序变得复杂冗长，降低了程序的可读性。C 语言提供了一种 switch 语句专门处理多分支情形，可以使程序变得简洁易懂。

switch 语句的一般形式如下：

```
switch(表达式)
{
    case 常量表达式1：语句1；
    case 常量表达式2：语句2；
                ⋮
    case 常量表达式n：语句n；
    default：语句n+1；
}
```

功能：首先计算 switch 后面括号内表达式的值，然后依次与各个 case 后面常量表达式的值进行比较，当表达式的值与某一个 case 后面常量表达式的值相等时，就选择这个标号作为入口，执行该 case 子句后面的语句，并继续执行其后的所有 case 子句直到程序结束。如果表达式的值与所有 case 后面的常量表达式的值都不相等，那么执行 default 后面的语句。

【例 3-17】 输入 1～7 的整数，要求输出对应的星期的英文单词。

```
//FileName：chap3_17.c
#include <stdio.h>
int main( )
{
    int a;
    printf("input integer number:");
    scanf("%d",&a);
    switch(a)
    {
        case 1：printf("Monday\n")；break;
        case 2：printf("Tuesday\n")；break;
        case 3：printf("Wednesday\n")；break;
        case 4：printf("Thursday\n")；break;
        case 5：printf("Friday\n")；break;
        case 6：printf("Saturday\n")；break;
        case 7：printf("Sunday\n")；break;
        default：printf("error\n")；
    }
    return 0；
}
```

程序运行结果如下：

```
input integer number:2↙
Tuesday
```

说明：【例 3-17】中出现了 break 语句，在 C 语言中，可以利用 break 语句终止该语句下面所有 case 子句和 default 子句的执行，直接跳出 switch 语句。这种用法在实际编程中比较常见。break 语句的具体用法参见 3.5 节。

在使用 switch 语句时，有以下几点需要注意：

(1)switch 后的表达式可以是整型表达式、字符型表达式或枚举型表达式。

(2)case 后的表达式一定是常量表达式，不允许是变量。

(3)在 case 后允许有多个语句，可以不用{}括起来。

(4)各 case 和 default 子句的先后顺序可以变动，而且不会影响程序的执行结果。

(5)default 子句可以省略不用。

(6)各 case 后常量表达式的值不能相同，否则会出现错误。

下面用带有 break 语句的 switch 语句重新解决【例 3-13】的问题，注意程序结构与可读性的变化。

【例 3-18】 输入一个百分制成绩，要求输出对应的成绩等级。

```
//FileName: chap3_18.c
#include <stdio.h>
int main( )
{
    float score;
    int temp;
    char grade;
    printf("Please input the score:");
    scanf("%f",&score);
    temp=(int)score/10;
    switch(temp)
    {
        case 10:
        case 9:grade='A';break;
        case 8:grade='B';break;
        case 7:grade='C';break;
        case 6:grade='D';break;
        default:grade='E';
    }
    printf("score is %.2f, grade is %c\n",score, grade);
    return 0;
}
```

程序运行结果如下：

```
Please input the score:65↙
score is 65.00, grade is D
```

说明：在【例 3-18】中，通过语句“temp=(int) score/10;”将合法的分数值 score 转换为 0～10 的整数，以此作为 switch 语句中表达式的值，然后依次与 case 子句中的常量表达式的值进行匹配，而不是直接用实型变量 score。ANSI C 标准规定，switch 后面括号内表达式的值可以是任意类型，但一般情况下是整型或字符型，而 case 后面常量表达式的值必须是整

型、字符型或者枚举类型。

【例 3-19】 输入年份和月份，打印输出该年该月有多少天。

分析：除 2 月份外，一年中的其他月份总是满足“大月 31 天，小月 30 天”的规则，因此，根据输入的月份是大月(1 月、3 月、5 月、7 月、8 月、10 月、12 月)还是小月(4 月、6 月、9 月、11 月)，就可以确定该月的具体天数。2 月份的天数与该年是平年还是闰年有关，该年为闰年则 2 月份为 29 天，否则为 28 天，因此需要判断该年是否为闰年。闰年的判断规则是：如果能被 400 整除或者能被 4 整除但不能被 100 整除，那么该年为闰年，否则为平年。为了程序的完备性，还应考虑输入的月份值超出 1～12 的情况，此时应该给出相应的提示信息。

算法描述如下：

(1)定义变量 year、month 和 day，分别用来存储年份、月份和天数。

(2)定义变量 leap，用来标识 year 是否为闰年，若 year 为闰年，则 leap←1，否则leap←0。

(3)如果 month 为大月，则天数 day←31；如果 month 为小月，则 day←30；如果 month 为 2 月并且 year 为闰年，则 day←29，否则 day←28。

(4)输出 day。

```
//FileName: chap3_19.c
#include <stdio.h>
int main()
{
    int year, month, day;
    int leap;
    printf("please input the year number:");
    scanf("%d", &year);
    printf("please input the month number:");
    scanf("%d", &month);
    if((year%400==0)||(year%4==0 && year%100 !=0))
        leap=1;
    else
        leap=0;
    switch(month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:day=31;break;
        case 4:
        case 6:
```

```
            case 9:
            case 11:day=30;break;
            case 2:if(leap==0)
                        day=28;
                   else
                        day=29;
                   break;
            default:day=1;
        }
        if(day==1)
            printf("Invalid month input!\n");
        else
            printf("%d.%d has %d days.\n", year, month, day);
        return 0;
    }
```

程序运行结果如下：

```
please input the year number:2012 ↙
please input the month number:5 ↙
2012.5 has 31 days.
```

【例 3-20】 编程设计一个简单的计算器程序，要求根据用户从键盘输入的表达式计算其值，指定的运算符为加(+)、减(-)、乘(*)、除(/)。

分析：从输入设备接收表达式后，取出运算符号，然后根据运算符号决定程序执行何种算术运算，最后输出计算结果。算法流程图如图 3-14 所示。

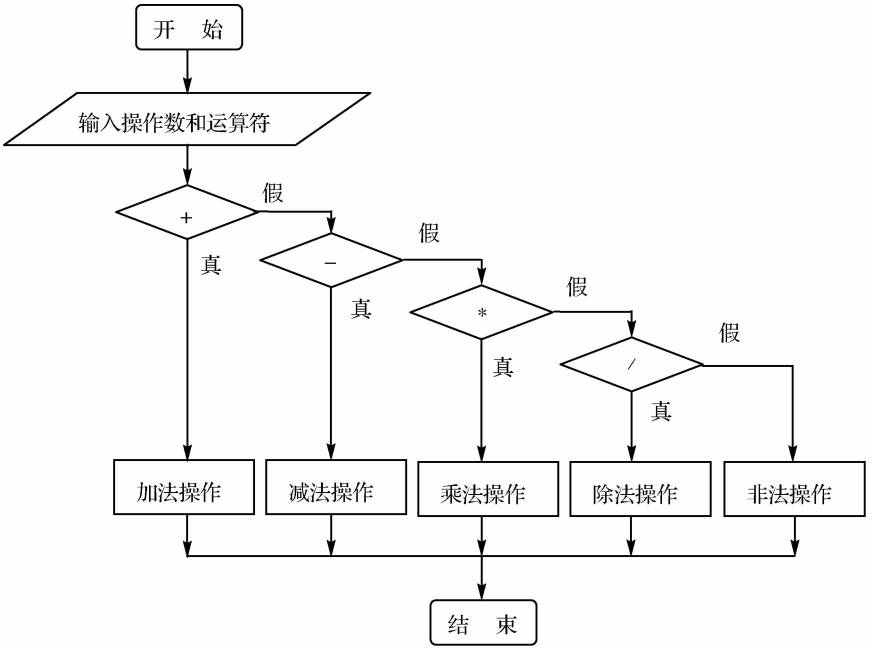

图 3-14 简单的计算器程序算法流程图

```
//FileName: chap3_20.c
#include <stdio.h>
int main()
{
    int data1,data2;
    char op;
    printf("please input the expression:");
    scanf("%d%c%d", &data1,&op,&data2);
    switch(op)
    {
        case '+': printf ("%d+%d=%d\n", data1,data2,data1+data2);
                  break;
        case '-': printf ("%d-%d=%d\n", data1,data2,data1-data2);
                  break;
        case '*': printf("%d*%d=%d\n", data1,data2,data1*data2);
                  break;
        case '/': if(data2==0)
                      printf("division by zero!\n");
                  else
                      printf("%d/%d=%d\n", data1,data2,data1/data2);
                    break;
        default: printf("unknown operation!\n");
    }
    return 0;
}
```

程序运行结果如下：

```
please input the expression:3*8↙
3*8=24
```

思维导图

3.4　循环结构程序设计

所谓循环，就是指反复执行某些语句或操作，执行次数由循环条件决定。循环结构是结构化程序设计的3种基本结构之一，经常与顺序结构、选择结构相互组合以解决较复杂的问题。在程序设计中使用循环结构能够将复杂问题简单化，从而降低程序书写长度，提高程序可读性和执行速度。

C语言提供3种循环语句，分别是while语句、do…while语句和for语句。利用这些语句可以实现不同形式的循环结构。循环语句通常都要有终止条件，根据终止条件判断时机

的不同，可以将循环语句分为两种类型，分别称为“当型”循环和“直到型”循环。“当型”循环是指首先判断条件是否满足，如果满足才进入循环，否则什么都不做，取“当条件成立才执行”之意。相应地，“直到型”循环是指首先进入循环，然后再判断条件是否满足，取“一直执行直到条件不成立为止”之意。

3.4.1 while 语句

while 语句可用来实现“当型”循环结构的控制。其一般形式如下：

```
while(表达式)
    语句;
```

功能：先计算表达式的值，如果为真，则执行循环体语句，当循环体语句全部执行完一次后，再次计算表达式的值，如果为真，则再次执行循环体语句，如此反复，直到表达式的值为假，此时不再执行循环体语句，而是退出循环，转去执行 while 语句后面的语句。while 语句的执行过程如图 3-15 所示。

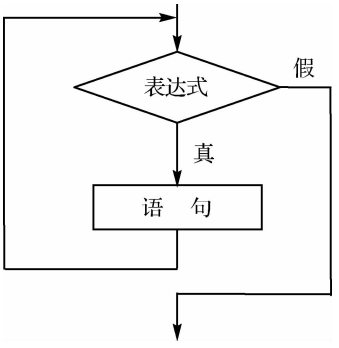

图 3-15　while 语句的执行过程

在使用 while 语句的过程中，应注意以下几点：

(1)表达式部分可以是任意类型的表达式，若表达式的值非 0，则表示循环条件判断结果为真，执行循环体语句；若表达式的值为 0，则表示循环条件判断结果为假，不执行循环体语句，从而结束循环，执行循环结构的下一条语句。

(2)语句部分为循环体，表示在循环条件成立时要反复执行的操作，可称为循环体语句。循环体语句可以是一条复合语句，即当需要反复执行的循环体部分由多条语句构成时，必须用一对大括号将这些语句括起来，使它们能从语法上被理解为一条语句。

(3)设计循环结构程序时，要注意让循环体执行若干次后能够结束，避免出现死循环。所谓死循环就是指由于循环控制不当，使得程序永远执行下去，不能结束循环。在上机操作时，如果所调试程序出现死循环时，可以通过按 Ctrl＋Break 组合键的方法中断程序执行。

【例 3-21】 用 while 循环求 1～100 的和。

分析：计算 1～100 的和，即计算 1＋2＋3＋…＋100，通常的做法是：先计算 1＋2 得到结果 3，再用这个结果与后一个数 3 相加，即 3＋3 得到结果 6，如此反复下去，直到加到 100 为止。除第 1 次加法外，后面的加法运算都是用前一次相加的结果与后一个数相加，这就是本题的规律。如果对第 1 次加法稍微作一点儿改变，也就是把“1＋2＝3”变为“0＋1＝1”和

“1＋2＝3”，那么求 1～100 的和就变成了每次都用前一次相加的结果与后一个数相加的过程，这里，最初的结果设为 0。

这样，可以定义一个变量 sum 保存每次相加的结果，初值为 0，利用循环不断在 sum 上累加，即 sum←sum＋i，变量 i 从 1 开始，每次增 1，直到 100。这时的 sum 常被形象地称为累加器。算法流程图如图 3-16 所示。

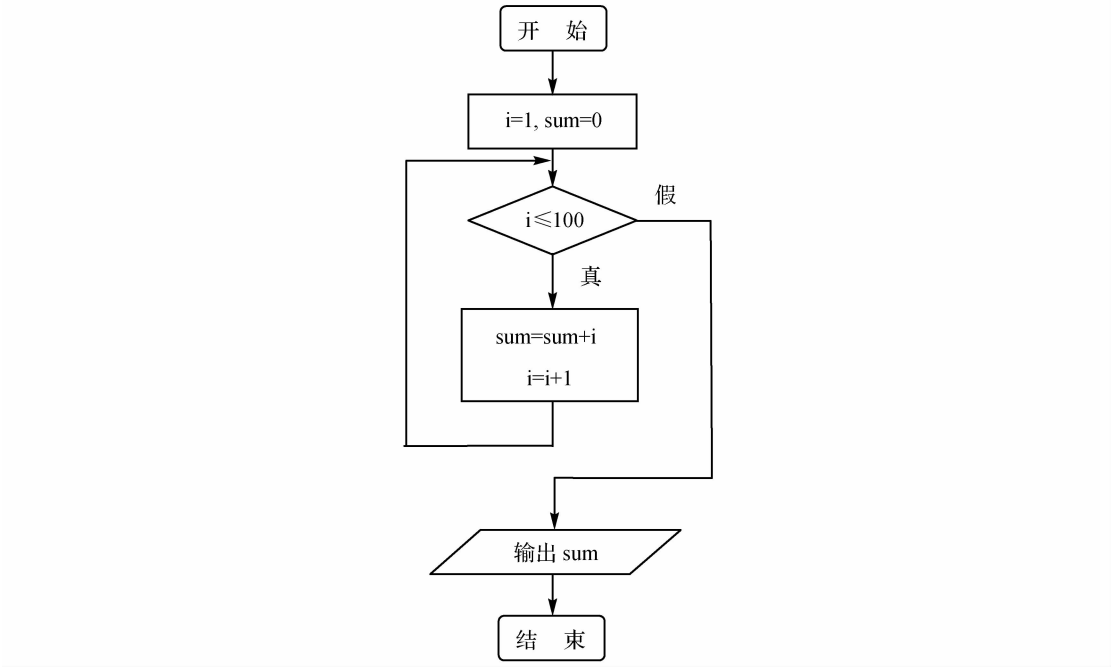

图 3-16　连续求和算法流程图

```
//FileName: chap3_21.c
#include <stdio.h>
int main()
{
    int sum=0, i=1;
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    printf("sum= %d\n", sum);
    return 0;
}
```

程序运行结果如下：

```
sum=5050
```

上述程序中所用到的算法是一个比较典型的迭代算法。所谓迭代算法就是不断用变量

的新值来取代变量的旧值，或由变量的旧值不断递推出变量的新值的过程。如【例 3-21】中的变量 i 和 sum 的取值就是一个迭代的过程。设计迭代算法时的关键因素如下：

①初值，如【例 3-21】中的 i＝1 和 sum＝0。

②迭代方式，如【例 3-21】中的“sum＝sum＋i；i＋＋；”。

③迭代条件，如【例 3-21】中的 i＜＝100。

【例 3-22】 任意输入 10 个整数，找出其中的最大值。

分析：在 10 个数中找出最大值，可以使用“打擂台”的方式，即先将输入的第 1 个数作为擂主（如 max），其他数作为挑战者（如 x），擂主与挑战者之间要进行 9 次较量，每次较量后的胜利者将成为新擂主。也就是说要将以下操作重复 9 次：输入一个 x 值（相当于上来一个挑战者去打擂）；若 max＜x，则 max＝x（相当于擂主被打败，挑战者成为新擂主）；9 次比较之后产生的新擂主即为 10 个数中的最大者。算法描述如图 3-17 所示。

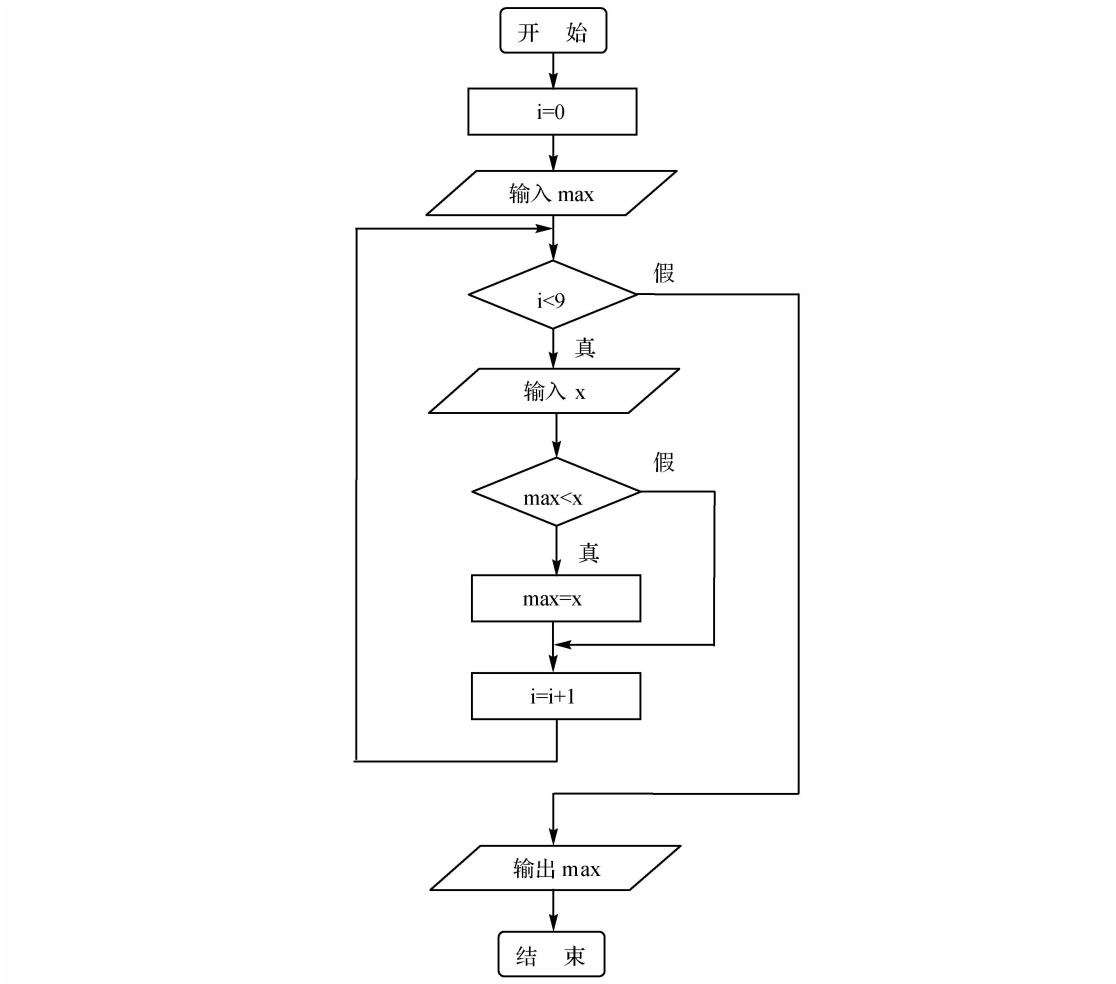

图 3-17　在 10 个数中找出最大值的算法流程图

```
//FileName: chap3_22.c
#include <stdio.h>
```

```
int main()
{
    int x,max,i;
    scanf("%d",&max);
    i=0;
    while(i<9)
    {
        scanf("%d",&x);
        if(max<x)
            max=x;
        i++;
    }
    printf("max=%d\n",max);
    return 0;
}
```

程序运行结果如下：

```
1  6  8  4  1  3  2  9  4  1↙
max=9
```

【例 3-23】 输入两个任意正整数，求它们的最大公约数。

分析：求最大公约数可以使用“辗转相除”的方法，即先把两个数中较大的一个作为被除数(如 m)，较小的一个作为除数(如 n)，求这两个数相除的余数(如 r)，当余数不为 0 时，则将原来的除数作被除数(即 $m=n$)，原来的余数作除数(如 $n=r$)，再求它们相除的余数，若余数不为 0，则重复以上操作，直到余数为 0 时为止，这时的除数即为这两个数的最大公约数。算法描述如图 3-18 所示。

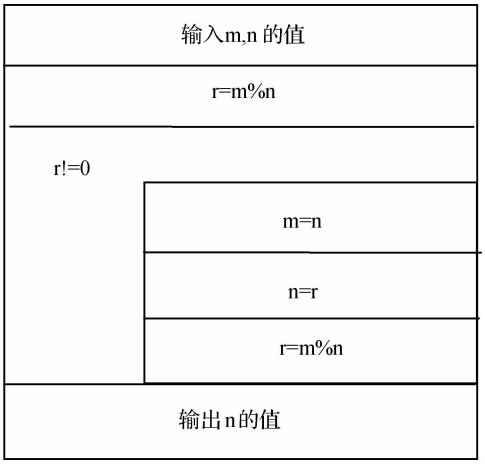

图 3-18　求最大公约数的算法 N-S 流程图

```
//FileName: chap3_23.c
#include <stdio.h>
```

```
int main()
{
    int m,n,r,t;
    scanf("%d,%d",&m,&n);
    if(m<n)
    {t=m; m=n; n=t; } //当 m<n 时,将 m 与 n 的值交换,以确保较大的数作为被
                      //除数
    r=m%n;
    while(r!=0)
    {
        m=n;
        n=r;
        r=m%n;
    }
    printf("%d\n", n);
    return 0;
}
```

程序运行结果如下：

```
8,12↙
4
```

3.4.2 do…while 语句

do…while 语句是用来实现“直到型”循环的循环语句。

do…while 语句的一般形式如下：

```
do
  语句;
while(表达式);
```

功能：先执行一次循环体语句，然后计算表达式的值，若为真，则继续执行循环体语句，然后再次计算表达式的值，如此反复，直到表达式的值为假，此时不再执行循环体语句，而是退出循环，转去执行 do…while 语句后面的语句。do…while 语句的执行过程如图 3-19 所示。

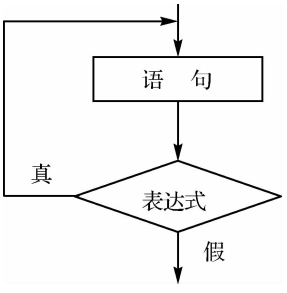

图 3-19　do…while 语句的执行流程图

从执行过程来看，do…while 语句的执行特点是：先执行循环体语句，后判断表达式的值，与 while 语句的执行特点正好相反。在 while 语句中，如果初始表达式的值为假，那么循环体语句一次也不执行，而在 do…while 语句中至少要执行一次循环体语句。

【例 3-24】 用 do…while 循环求 n!。

分析：计算 $n!$，即计算 $1\times2\times3\times\cdots\times n$，与【例 3-21】中计算 $1+2+3+\cdots+n$ 具有相似的规律，不同之处仅是把重复作加法变成了重复作乘法，随之而来的变化就是定义一个保存相乘结果的变量 multi，初值应该为 1，而不是 0。这时的 multi 常被形象地称为累乘器。

算法描述如下：

①定义变量 n，接受用户输入。

②定义累乘器变量 multi，multi←1。

③定义变量 i，i←1。

④如果 n＜0，则给出相应提示信息；如果 n＝0，则直接输出变量 multi 的值 1。否则，执行第⑤～⑥步。

⑤如果 i＜＝n，则循环执行以下语句。

```
multi=multi * i;//实现累乘功能
i++;
```

⑥输出 multi 的值。

```
//FileName: chap3_24.c
#include <stdio.h>
int main()
{
    int n, i=1;
    long int multi=1;
    printf("please input n(n>=0):");
    scanf("%d",&n);
    if(n<0)
    printf("invalid input! ");
    else if(n==0)
        printf("%d!=%ld\n", n, multi);
    else
    {
        do
        {
            multi=multi * i;
            i++;
        } while(i<=n);
    }
    printf("%d!=%ld\n", n, multi);
    return 0;
```

```
}
```

程序运行结果如下：

```
please input n(n>=0):5↙
5!=120
```

【例 3-25】 设有一个分数数列：2/1，3/2，5/3，8/5，13/8，21/13，…。编程求出这个数列的前 20 项之和。

分析：从分数数列数据项的关系可以发现，后一项的分母等于前一项的分子，后一项的分子等于前一项的分子与分母之和。这种根据前一项递推出后一项的方法也是迭代算法。设当前项的分子为变量 b，分母为变量 a，分数数列各项的和为变量 sum，用变量 i 作为循环次数的计数器，可找出迭代的以下 3 个要素：

①初值：i=1，a=1，b=2，sum=0。

②迭代公式：sum=sum+b/a，b=a+b，a=b-a，i++。

③迭代条件：i<=20。

算法描述如图 3-20 所示。

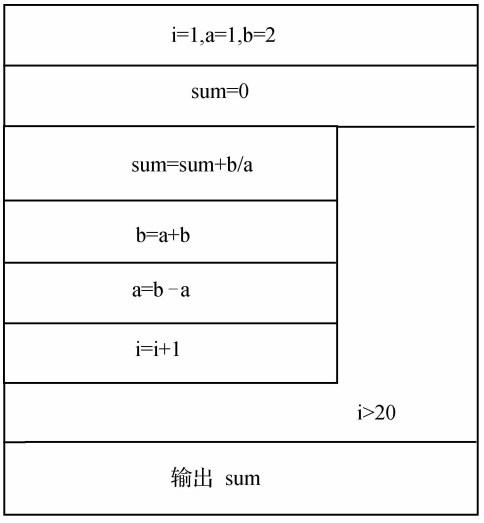

图 3-20 求数列之和的算法 N-S 流程图

```
//FileName: chap3_25.c
#include <stdio.h>
int main()
{
    int a=1,b=2,i=1;
    float sum=0;
    do
    {
        sum=sum+(float)b/a;
        b=a+b;
```

```
        a=b-a;
        i++;
    }while(i<=20);
    printf("sum= %f\n",sum);
    return 0;
}
```

程序运行结果如下：

```
sum=32.660259
```

【例 3-26】 编写程序，找出满足条件 1+2+3+…+n<500 的最大的 n 值。

分析：定义一个累加器 sum，然后从自然数 1 开始依次进行累加，直到与自然数 n 相加后累加器 sum 的值大于 500，则 n 的前一个自然数 n-1 即为所求。算法流程图如图 3-21 所示。

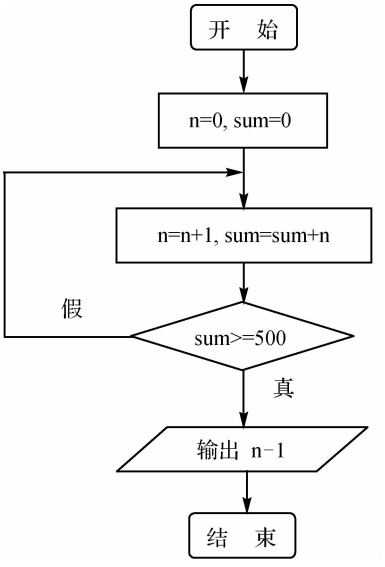

图 3-21　找出 n 值的算法流程图

```
//FileName: chap3_26.c
#include <stdio.h>
int main()
{
    int n=0, sum=0;
    do
    {
        n++;
        sum=sum+n;
    }while(sum<500);
    printf("s= %d\n", n-1);
```

```
    return 0;
}
```

程序运行结果如下：

```
s=31
```

3.4.3 for 语句

真题测试

C 语言提供了另外一种循环语句，即 for 语句。与 while 语句和 do…while 语句相比，for 语句使用更方便，也更灵活，它是最复杂的循环语句。

for 语句的一般形式如下：

```
for(表达式 1;表达式 2;表达式 3)
    语句;
```

功能：先计算“表达式 1”的值，再计算“表达式 2”的值，如果“表达式 2”的值为真，则执行循环体语句，执行一次后，计算“表达式 3”的值，然后再次计算“表达式 2”的值，如果仍为真，则继续执行循环体语句，如此反复，直到“表达式 2”的值为假，此时不再执行循环体语句，而是退出循环，转去执行 for 语句后面的语句。for 语句的执行过程如图 3-22 所示。

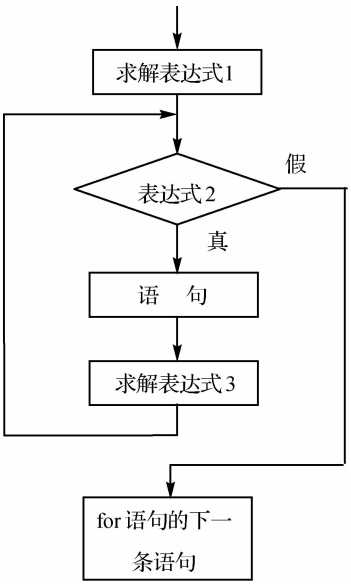

图 3-22　for 语句的执行流程图

下面给出这种应用最广泛，也最容易理解的 for 语句的一般形式：

```
for(循环变量赋初值;循环条件;循环变量值改变)
    语句;
```

这里将控制循环次数的变量称为循环变量。

【例 3-27】 计算 1～n 的自然数的平方和。

分析：采用 for 语句实现求解自然数的平方和问题，需要定义循环变量 i，初值为 1，循环条件为 i<=n，循环变量 i 每次增 1。然后定义一个累加器 sum，对 i 的平方，即 i * i 实现累

加即可。

算法描述如下：

①接收用户输入的 n 值，并进行有效性检验。

②定义循环变量 i，定义累加器变量 sum，sum←0。

③循环变量 i←1，循环条件 i<=n，循环变量 i++，循环执行以下语句：

```
sum=sum+i*i;
```

④输出 sum 的值。

```
//FileName: chap3_27.c
#include <stdio.h>
int main()
{
    int i, n;
    long int sum=0;
    printf("Please input n(n>=1): ");
    scanf("%d", &n);
    if(n<1)
    {
        printf("Invalid input!");
    }
    else
    {
        for(i=1;i<=n;i++)
        {
            sum=sum+i*i;
        }
        printf("The result is: %ld\n", sum);
    }
    return 0;
}
```

程序运行结果如下：

```
Please input n(n>=1):5↙
The result is:55
```

相关说明如下：

①for 循环中的“表达式 1”“表达式 2”“表达式 3”都是选择项，都可以省略，但是分号不能省略。

②“表达式 1”通常用于循环变量赋初值，如果省略，表示不对循环变量赋初值或者已经把赋初值语句放在了 for 语句前面。

例如，【例 3-27】中的循环语句可以用下面的语句替换：

```
i=1;
```

```
for(;i<=n;i++)
{
    sum=sum+i*i;
}
```

③“表达式 2”通常用于表示循环条件，如果省略，不作其他处理时便成为死循环，这就需要在循环体语句中放有循环结束的语句。省略表达式 2，系统默认循环条件永远为真。

例如，【例 3-27】中的循环语句可以用下面的语句替换：

```
for(i=1;;i++)
{
    if(i>n)
        break;              //强制退出循环体语句的执行
    sum=sum+i*i;
}
```

④“表达式 3”通常用于循环变量增值，如果省略，则不对循环控制变量进行操作，这时应在循环体语句中增加能够实现对应功能的语句。

例如，【例 3-27】中的循环语句可以用下面的语句替换：

```
for(i=1;i<=n;)
{
    sum=sum+i*i;
    i++;
}
```

⑤“表达式 2”一般为关系表达式或逻辑表达式，但也可以是数值表达式或字符表达式，只要其值非零，就执行循环体。

【例 3-28】 输入一串字符，统计输入字符的个数。

分析：C 语言中不能定义字符串型变量，因此该题可以定义一个变量，通过循环依次接收字符串中的各个字符，每接收一个，计数加 1，当所有字符被接收完毕（通常情况下以回车作为输入字符串的结束标志），就得到了输入字符的个数。

```
//FileName: chap3_28.c
#include <stdio.h>
int main()
{
    int n;
    char c;
    for(n=0;(c=getchar())!='\n';n++)
        ;
    printf("The number of letter is %d\n",n);
    return 0;
}
```

程序运行结果如下：

```
The C program ↙
The number of letter is 13
```

说明：该程序中并没有循环变量，变量 n 用来统计用户输入的字符个数，“表达式 1”将变量 n 的初始值赋为 0。“表达式 2”首先获取用户输入的一个字符，然后与'\n'比较，如果不相等，那么是一个有效的字符，即该字符是需要统计的一个字符，否则循环结束，“表达式 3”并非循环变量增值，而是用于计数，即将 n 的值加 1。循环体语句为空语句。

【例 3-29】 输出 Fibonacci 数列 1，1，2，3，5，8，13…的前 20 项，要求每输出 5 项后换行。

分析：Fibonacci 数列具有以下规律：前项为 1，从第 3 项开始，每一项都为前两项之和。设变量 f 表示要求的当前项，变量 f1 和 f2 依次为当前项的前两项。例如，若当前项 f=5，则 f1=2，f2=3；若当前项 f=8，则 f1=3，f2=5。算法描述如图 3-23 所示。

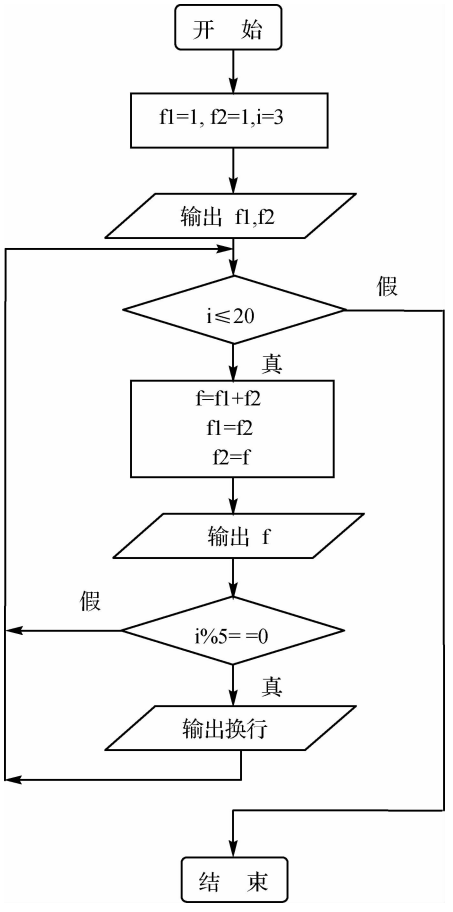

图 3-23　Fibonacci 数列的算法流程图

```
//FileName: chap3_29.c
#include <stdio.h>
int main()
{
    int f1,f2,f,i;
```

```
    f1=f2=1;
    printf("%10d%10d",f1,f2);          //前 2 项先输出
    for(i=3;i<=20;i++)                 //迭代从第 3 项开始
    {
        f=f1+f2;
        f1=f2;
        f2=f;
        printf("%10d",f);
        if(i%5==0)                     //每行输出够 5 个,就换行
            printf("\n");
    }
    return 0;
}
```

程序运行结果如下：

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

用 for 循环能够解决的问题，往往也可以用 while 循环或 do…while 循环解决，对于一个实际问题，应该使用哪种循环语句并没有一个绝对的标准，这在很大程度上依赖于个人喜好。但是在实际应用中，使用哪种语句会更自然、更方便，也有一般性的规律。

(1)如果循环次数在执行循环体之前就已确定，一般使用 for 语句；如果循环次数是由循环体的执行情况确定的，一般使用 while 语句或者 do…while 语句。

(2)当循环次数未知，循环体至少执行一次时，用 do…while 语句；反之，如果循环体可能一次也不执行，那么应该用 while 语句。

3.4.4 循环的嵌套

如果将一个循环语句放在另一个循环语句的循环体中，就构成了循环的嵌套。内嵌的循环结构内还可嵌套循环结构，从而构成多层嵌套。3 种循环语句都可以在循环嵌套中使用。

【例 3-30】 输入 n 的值，计算并输出 1～n 的阶乘之和。

分析：该题是要计算阶乘之和。当用户输入 n 的值之后，可依次计算出 1～n 各项的阶乘，并逐步累加，最终得到结果。求某一项的阶乘需要采用循环结构，而累计求各项阶乘的和，也需要采用循环结构，前者是后者内嵌的一个循环结构，构成了循环的嵌套。根据分析，外循环采用 while 语句，内循环采用 for 语句。算法描述如图 3-24 所示。

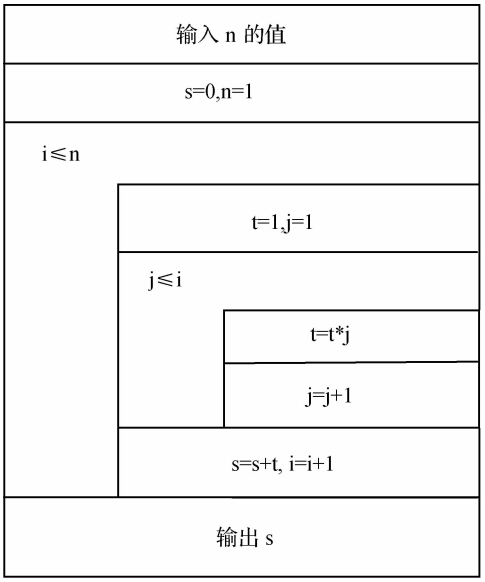

图 3-24　求阶乘和的算法 N-S 流程图

```
//FileName: chap3_30.c
#include <stdio.h>
int main()
{
    int n,s,i,j,t;
    scanf("%d",&n);
    s=0;
    i=1;
    while(i<=n)
    {
        t=1;
        for(j=1;j<=i;j++)
            t=t*j;
        s=s+t;
        i++;
    }
    printf("The result is %d\n",s);
    return 0;
}
```

程序运行结果如下：

```
4↙
The result is 33
```

说明：在【例 3-30】中，变量 s 用于保存阶乘之和，初始值为 0，随着变量 i 的变化逐步累

加各项的阶乘 t;i 为循环变量,用于计算阶乘之和,取值范围是 1～n;t 用于计算第 i 项的阶乘,在计算第 i 项的阶乘前,其值置为 1,随着变量 j 的变化逐步求出第 i 项的阶乘;j 也是循环变量,用于计算第 i 项的阶乘,取值范围是 1～i。

如果以上程序的内循环和外循环均采用 for 语句,将更简洁,程序代码如下:

```
#include <stdio.h>
int main()
{
    int n,s,i,j,t;
    scanf("%d",&n);
    s=0;
    for(i=1;i<=n;i++)
    {
        t=1;
        for(j=1;j<=i;j++)
          t=t*j;
        s=s+t;
    }
    printf("The result is %d\n",s);
    return 0;
}
```

【例 3-31】 打印 n 行 m 列的星形矩阵。假设 $n=4$,$m=5$,打印出来的图形如下所示。

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

分析:图形共有 n 行 m 列,对于每一行,都要依次打印 m 个星号。因此对于某一行星号的打印,可以采用一个循环语句(本题选用 for 语句)来实现。由于一行星号打印完成后,下一行还是从屏幕的第一个字符位置开始打印,因此在这个循环语句的下面要输出一个换行符。本题要求输出 n 行,这就要将前面的过程重复 n 次,因此把前面的过程作为另外一个循环的循环体语句,就完成了星形矩阵图形的输出。通常将这两个循环语句分别称为内层循环和外层循环,内层循环控制输出的列数,外层循环控制输出的行数。这样外层循环每执行一次,就输出一行。

算法描述如下:

①接收用户输入 n 和 m 的值。

②定义外层循环变量 i 和内层循环变量 j。

③循环变量赋初值 i←1,循环条件 i<=n,循环变量 i++,循环执行:循环变量赋初值 j←1,循环条件 j<=m,循环变量 j++,循环输出 *,最后输出\n。

```
//FileName: chap3_31.c
#include <stdio.h>
```

```
int main()
{
    int n,m;
    int i,j;
    printf("Please input n(n>=1):");
    scanf("%d",&n);
    printf("Please input m(m>=1):");
    scanf("%d",&m);
    if(n<1||m<1)
        printf("Invalid input!");
    else
        for(i=0;i<n;i++)                    // 控制行
        {
            for(j=0;j<m;j++)                //控制列
              printf("*");
            printf("\n");                    //换行
        }
    return 0;
}
```

程序运行结果如下：

```
Please input n(n>=1):4↙
Please input m(m>=1):5↙
* * * * *
* * * * *
* * * * *
* * * * *
```

【例 3-32】 用穷举法解决搬砖问题：36 块砖，36 人搬，男人搬 4 块，女人搬 3 块，2 个小孩抬 1 块，要求一次搬完，问需要男、女、小孩各多少人？

分析：这是一个非常典型的利用穷举法解决的问题。穷举法是一种最直接、实现最简单的解决实际问题的算法思想，但这种算法非常耗费时间，运行效率十分低下。然而，随着 CPU 运算速度的不断提高以及多处理器并行计算技术的发展，穷举法也不失为一种较好的选择。

穷举法的基本思想是：在可能的解空间中穷举出每一种可能的解，并对每一个可能解进行判断，从中得到问题的答案。使用穷举思想解决实际问题，最关键的步骤是划定问题的解空间，并在该解空间中逐一枚举每一个可能的解。因此，使用该算法必须要注意两个问题：一是解空间的划定必须保证覆盖问题的全部解，二是解空间集合及问题的解集一定是离散的集合，也就是集合中的元素是可列的、有限的。对于搬砖问题，可设 3 个变量 m、w、c，分别表示男人、女人和小孩的人数，根据题意可列出以下 2 个方程。

$$\begin{cases} m+w+c=36 \\ 4*m+3*w+c/2=36 \end{cases}$$

根据题意确定 m 和 w 的取值范围为：$1\leqslant m<9$，$1\leqslant w<12$。一旦 m 和 w 的值确定之后，c 的值便为 $36-m-w$。

```
//FileName: chap3_32.c
#include <stdio.h>
int main()
{
    int m,w,c;
    printf("%10s%10s%10s\n","men","women","children");
    for(m=1;m<9;m++)
        for(w=1;w<12;w++)
        {
            c=36-m-w;
            if(m*4+w*3+c/2.0==36)
                printf("%10d%10d%10d\n",m,w,c);
        }
    return 0;
}
```

程序运行结果如下：

```
  men       women       children
   3          3             30
```

说明：若将程序中的语句“if(m*4+w*3+c/2.0==36) printf("%10d%10d%10d\n",m,w,c);”中的 c/2.0 换成 c/2，则运行结果如下：

```
  men   women   children
   1      6       29
   3      3       30
```

显然第一行结果并不符合题意，这是由于在 C 语言中，c/2.0 与 c/2 是有区别的，c/2.0 表示除法操作，c/2 表示取整操作。

3.5 转移控制语句

在循环语句执行过程中，当循环条件不满足时，可以正常退出循环，转而执行循环后面的语句。但是在实际应用中，可能会存在这样的情况，即在循环条件仍满足的条件下终止整个循环或者终止本次循环，这时就要用到转移控制语句，即 break 语句或 continue 语句。

3.5.1　break 语句

break 语句除用于退出 switch 结构外，还可用于由 while、do…while 和 for 构成的循环结构中。当执行循环体遇到 break 语句时，break 所在循环将立即终止，接着从循环语句后的第一条语句开始继续向下执行。

break 语句的一般形式如下：

```
break;
```

break 语句在 while、do…while 和 for 语句构成的循环结构中的执行过程如图 3-25 所示。

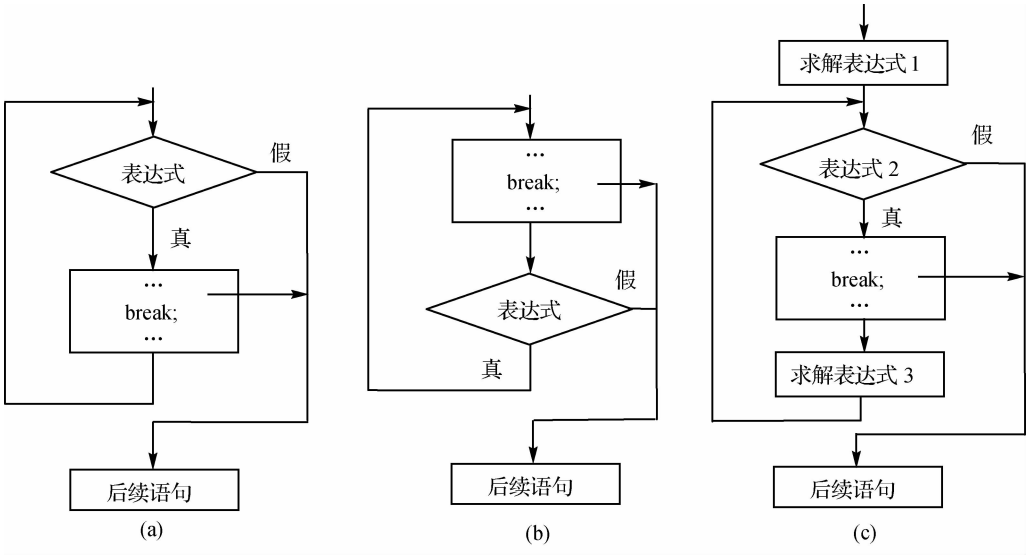

图 3-25　break 语句在循环结构中的执行流程图

(a)while 循环；(b)do…while 循环；(c)for 循环

【例 3-33】 求 1 000～2 000 之间最小的，同时能够被 11 和 17 整除的数。

分析：该问题可采用穷举法求解，定义一个循环变量 i，从 1 000 至 2 000 递增，对每一个 i，判断其是否能同时被 11 和 17 整除，若可以，则结束循环，该值即为所求。算法描述如图 3-26 所示。

```
//FileName: chap3_33.c
#include <stdio.h>
int main()
{
    int i;
    for(i=1000;i<=2000;i++)
    {
        if(i%11==0 && i%17==0)
        {
```

```
                printf("%d\n",i);
                break;
            }
        }
        return 0;
    }
```

程序运行结果如下：

1122

说明：程序运行时，i 逐步递增，对于每一个 i，都利用 i%11==0&&i%17==0 判断是否满足条件，当满足条件时，执行 break 语句，结束 for 循环，输出当前 i 的值。

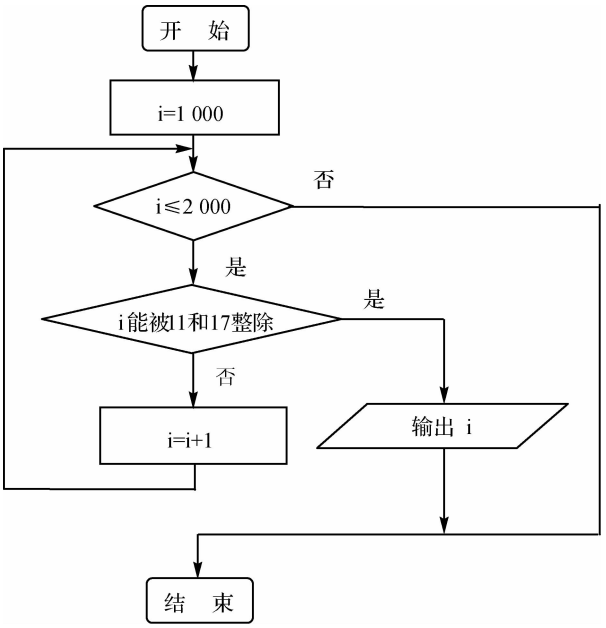

图 3-26　【例 3-33】算法流程图

使用 break 语句时，应注意以下问题：

①break 语句只能用于 switch 语句或 for 语句、while 语句和 do…while 语句。

②在使用 break 语句时，应注意 break 语句所在的位置，break 语句只能终止所在的最近一层的循环语句或 switch 语句。

【例 3-34】　判断任意一个数是否为素数。

分析：所谓素数是指只能被 1 和它本身整除的数。判断一个数 m 是否为素数即判断 m 能否被 2～(m－1)范围内的数整除，如果一个都不能整除，那么该数为素数，否则该数不是素数。

算法描述如下：

①接收用户输入的 m 的值。

②循环变量赋初值 i←2，循环条件 i<=(m－1)，循环变量 i＋＋，循环执行语句：若 m 能被 i 整除，则结束循环。

③循环结束后判断 m 与 i 的关系，如果 m 等于 i，则输出 m 是素数，否则输出 m 不是素数。

```
//FileName: chap3_34.c
#include <stdio.h>
int main()
{
    int m,i;
    scanf("%d",&m);
    for(i=2;i<=m-1;i++)
    {
        if(m%i==0)
            break;
    }
    if(i==m)
        printf("%d is prime number\n",m);
    else
        printf("%d is not prime number\n",m);
    return 0;
}
```

程序运行结果如下：

```
41↙
41 is prime number
```

说明：对【例 3-34】进行适当修改，即可求出 100～200 的所有素数。

```
#include <math.h>
#include <stdio.h>
int main()
{
    int n,i,k;
    for(n=100;n<=200;n++)
    {
        k=sqrt(n);
        for(i=2;i<=k;i++)
        {
            if(n%i==0)
                break;
        }
        if(i==k+1)
        printf("%d \n",n);
    }
    return 0;
}
```

3.5.2 continue 语句

continue 语句与 break 语句不同，当在循环体中遇到 continue 语句时，程序将不执行 continue 语句后面尚未执行的语句，而是开始下一次循环，即只结束本次循环的执行，并不终止整个循环的执行。

continue 语句的一般形式如下：

```
continue;
```

continue 语句在 while、do…while 和 for 语句构成的循环结构中的执行过程如图 3-27 所示。

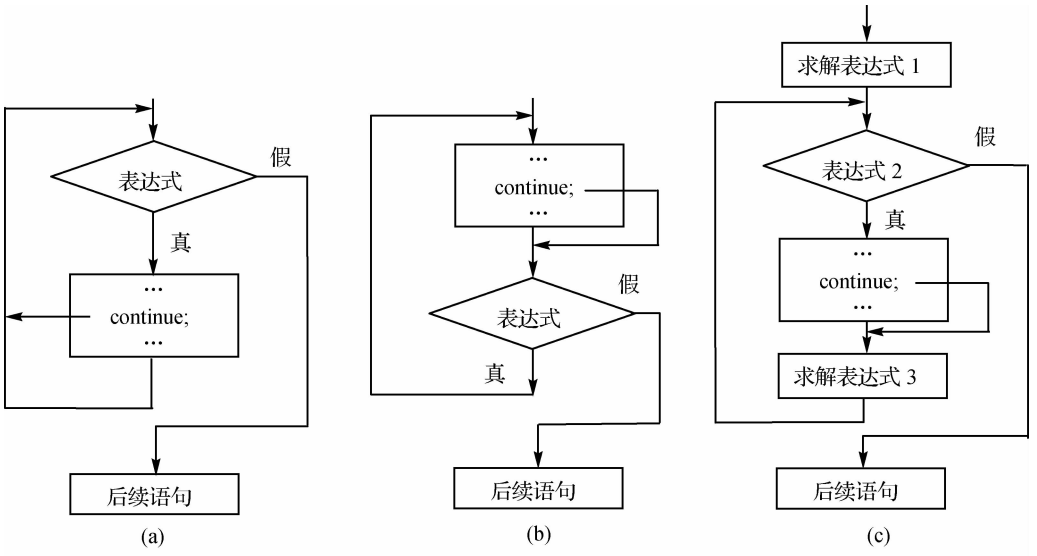

图 3-27 continue 语句在循环结构中的执行流程图

(a)while 循环；(b)do…while 循环；(c)for 循环

【例 3-35】 输出 100～120 之间不能被 3 整除的整数。

分析：该题同样可以采用穷举法求解，定义一个整型变量 i，初值为 100，每次通过自加 1 操作，便可取到该范围内的每一个值，然后判断该数是否能被 3 整除，如果能被 3 整除，就接着取下一个数进行判断，否则，输出该数。算法描述如图 3-28 所示。

```
//FileName: chap3_35.c
#include <stdio.h>
int main()
{
    int n;
    for(n=100;n<=120;n++)
    {
        if(n%3==0)
            continue;
```

```
            printf("%d  ",n);
        }
        return 0;
}
```

程序运行结果如下：

```
100  101  103  104  106  107  109  110  112  113  115  116  118  119
```

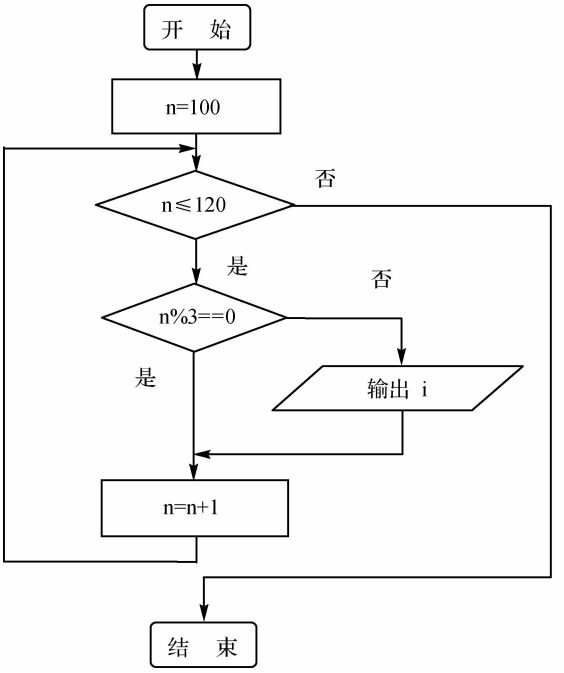

图 3-28　【例 3-35】算法流程图

【例 3-36】　在半径为 1～10 的圆中，输出面积超过 100 的圆的半径和面积。

分析：求解本题可以将半径作为循环变量 r，初值为 1，循环条件为 r<＝10，在循环体中计算圆的面积 s＝π＊r＊r。若 s<＝100，则不输出任何信息，继续进行下一次循环，否则输出 r 和 s。

在圆的面积计算公式中，用到了一个常量 π，因此在编程时定义一个常量并命名为 PI，用于保存 π 的值，以提高程序的可读性和可维护性。

算法描述如下：

①定义常量 PI，值为 3.14。

②定义循环变量 r 和存储圆的面积值的变量 s。

③循环变量赋初值 r←1，循环条件 r<＝10，循环变量 r＋＋，循环执行以下语句：

- 计算圆的面积 s＝PI＊r＊r。
- 若 s<＝100，则执行 continue 语句，结束本次循环。
- 输出 r 和 s 的值。

```
//FileName: chap3_36.c
#include <stdio.h>
```

```
#define PI 3.14     //定义符号常量
int main()
{
    int r;
    float s;
    for(r=1;r<=10;r++)
    {
        s=PI*r*r;
        if(s<=100.0)
            continue;
        printf("r= %d, s= %f\n", r, s);
    }
    return 0;
}
```

程序运行结果如下：

```
r=6, s=113.040001
r=7, s=153.860001
r=8, s=200.960007
r=9, s=254.339996
r=10, s=314.000000
```

3.5.3 goto 语句

break 语句和 continue 语句都能改变程序的控制流程，但控制转向的位置是有限制的。break 语句只能转向 switch 语句或循环语句的下一条语句，continue 语句只能转向循环开始处，从而使程序具有模块化结构，因此可以把这两种语句称为有条件的转向语句。若要实现无条件转向功能，则可以使用 goto 语句。

goto 语句的一般形式如下：

goto <语句标号>;

功能：程序流程无条件转向语句标号标识的语句。其中，goto 是关键字，语句标号是合法的 C 语言标识符，用来标识 goto 语句所要转到的语句。

如果程序中使用了 goto 语句，那么在程序中一定要有一条 C 语句，即它的前面有语句标号，该语句标号与 goto 语句中的语句标号一致。具有语句标号的 C 语言语句的一般形式如下：

<语句标号>:<语句>

goto 语句一种常见的用法是与 if 语句配合使用。

【例 3-37】 计算 1～100 的和，要求使用 goto 语句。

```
//FileName: chap3_37.c
#include <stdio.h>
```

```
int main()
{
        int i=1, sum=0;
        loop: if(i<=100)          //添加语句标号 loop
        {
            sum=sum+i;
            i++;
            goto loop;       //程序转到 loop 语句标号处执行
        }
        printf("%d\n",sum);
        return 0;
}
```

程序运行结果如下：

```
5050
```

需要强调的是，由于 goto 语句可以实现程序控制的随意转移，违背了结构化程序设计的思想，从而破坏了程序的模块化结构，因此，通常建议在 C 语言程序中尽量不采用 goto 语句。但是，并不是不能使用 goto 语句，在某些特殊情况下，使用 goto 语句还可能会提高程序的可读性。

3.6 综合实例

【例 3-38】 统计选票。现有选票如下：3,1,2,1,1,3,3,2,1,2,3,3,2,1,1,3,2,0,1,4,−1。其中−1 是结束标志。设 1 选李，2 选张，3 选王，0 和 4 为废票，问谁会当选？

分析：求解本题，就是要分别计算出各个数字的个数，因此，用户每输入一个数字，就应该判断该数字所代表的对象。若该数字不是−1，则在代表相应对象的累加器上进行加 1 操作，若输入的数字为−1，则程序结束。该算法的程序流程图如图 3-29 所示。

```
//FileName: chap3_38.c
#include <stdio.h>
int main()
{
    int vote,lvote=0, zvote=0, wvote=0, invalidvote=0;
    scanf("%d",&vote);
    while(vote!=-1)
    {
        switch(vote)
        {
            case 1: lvote++;break;
```

```
                case 2: zvote++;break;
                case 3: wvote++;break;
                case 0:
                case 4: invalidvote++;
            }
            scanf("%d",&vote);
        }
        printf("Li: %2d,Zhang: %2d,Wang: %2d,Invalid: %2d", lvote, zvote,
wvote, invalidvote);
        return 0;
    }
```

程序运行结果如下：

```
3 1 2 1 1 3 3 2 1 2 3 3 2 1 1 3 2 0 1 4 -1↙
Li: 7, Zhang: 5, Wang: 6, Invalid: 2
```

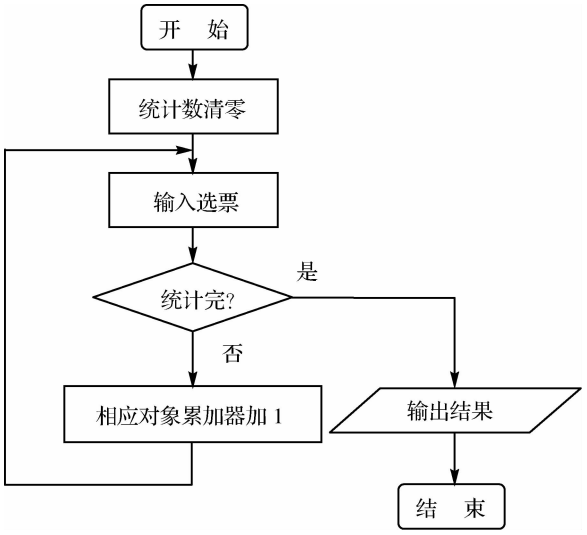

图 3-29　统计选票算法流程图

【例 3-39】 输出下列图形。

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

分析：该图形共有 4 行，每行都是由空格符和星号组成的，第一行的构成规则是：3 个空格、1 个星号、3 个空格。第二行的构成规则是：2 个空格、3 个星号、2 个空格。第三行的构成规则是：1 个空格、5 个星号、1 个空格，……这里，由于右端的空格输出与否并不影响输出图形的效果，因此可以不予考虑。左端空格的个数与所在的行数存在关系：空格数＋行数＝4，星号的个数与所在的行数存在关系：星号个数＝2×行数－1。可以用双层循环实现，外层

循环控制行的变化，内层循环有两个循环语句：第一个循环输出空格，第二个循环输出星号。本题采用 for 语句比较合适。

算法描述如下：

①定义循环变量 row，col_space，col_star。

②循环变量赋初值 row←1，循环条件 row<=4，循环执行以下语句。

- 循环变量赋初值 col_space←1，循环条件 col_space<=4－row，循环输出空格符。
- 循环变量赋初值 col_star←1，循环条件 col_star<=2×row－1，循环输出星号。
- 输出换行符“\n”。

```
//FileName: chap3_39.c
#include <stdio.h>
int main()
{
    int row, col_space, col_star;
    for(row=1;row<=4;row++)
    {
        for(col_space=1;col_space<=4-row;col_space++)
              printf(" ");
        for(col_star=1;col_star<=2*row-1;col_star++)
              printf("*");
        printf("\n");  //输出一行内容后换行
    }
    return 0;
}
```

程序运行结果如下：

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

【例 3-40】 输入任意一个长整型数，求它的各位数字之和。

分析：设输入的长整型数为 m，它的个位数字 n 可通过 $n=m\%10$ 得到，其他各位数字可以通过 $m=m/10$ 得到，重复上述操作，将每次所得的个位数字累加求和，直到这个数变为 0。该算法的流程图如图 3-30 所示。

```
//FileName: chap3_40.c
#include <stdio.h>
int main()
{
    long m;
    int n, sum=0;
    scanf("%ld",&m);
```

```
    while(m>0)
    {
        n=m%10;
        sum=sum+n;
        m=m/10;
    }
    printf("sum=%d\n",sum);
    return 0;
}
```

程序运行结果如下：

1234↙

sum=10

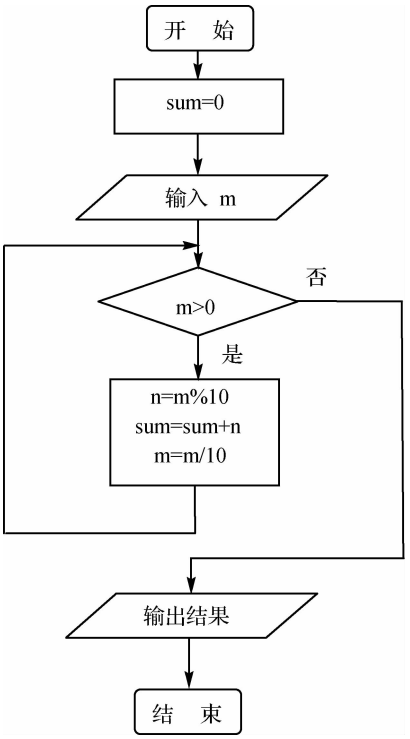

图 3-30 【例 3-40】算法流程图

资料

等级考试重难点

本 章 小 结

算法是程序设计的灵魂，也是编程的基础。本章首先介绍了算法的基本概念，重点讲述了算法的几种主要的描述方法，即自然语言、流程图和 N-S 图，并且在程序设计过程中反复使用了上述算法描述方法。

程序流程控制语句是C语言程序设计中很重要的部分，可分为顺序语句、选择语句和循环语句。顺序语句包括表达式语句、函数调用语句、空语句和复合语句，选择语句包括if语句和switch语句，循环语句包括while语句、do…while语句和for语句。

if语句中还可以嵌套if语句，这时会出现多个if和多个else重叠的情况，要特别注意if和else的配对问题，else语句总是与它前面最近的那个未分配的if语句配对。为了使程序条理更加清晰、明确，可以使用语句的嵌套，但是要注意嵌套语句之间的逻辑关系，嵌套的语句之间不能交叉。

循环语句可以实现大量的重复工作，当需要某段程序至少执行一次时可以选择do…while语句，而某段程序可能一次也不执行时可以选择while语句。

break语句用在switch语句或循环语句中，以跳出switch语句或者本层循环而去执行下一条语句，而continue语句只能用在循环语句中，它的作用是结束本次循环，继续下一次循环条件的判断与执行。要特别注意break语句与continue语句的用法和区别。

习 题 3

1)选择题

(1)C语言的if语句中，用做判断的条件表达式为(　　)。

A. 任意表达式　　B. 逻辑表达式　　C. 关系表达式　　D. 算术表达式

(2)若希望当x的值为奇数时，表达式的值为真，x的值为偶数时，表达式的值为假，则以下不能满足要求的表达式是(　　)。

A. x%2==1　　B. x%2　　C. !(x%2)　　D. !(x%2==0)

(3)已知x=12，y=10，执行语句“y=x>12 ? x+1: x-1;”后y的值为(　　)。

A. 13　　B. 11　　C. 0　　D. 10

(4)已知a=1、b=2、c=3、d=4、m=2、n=2，执行语句“(m=a>b)&&(n=c>d);”后n的值为(　　)。

A. 1　　B. 2　　C. 3　　D. 4

(5)下列程序运行后输出(　　)。

```
#include <stdio.h>
int main()
{
    int x=10,y=20,z=30;
    if(x>y) z=x; x=y; y=z;
    printf("%d,%d,%d",x,y,z);
    return 0;
}
```

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

(6)下列程序(　　)。

```
#include <stdio.h>
int main()
{
    int a=5,b=0,c=0;
    if(a=b+c)
        printf("111\n");
    else
        printf("222\n");
    return 0;
}
```

A. 有语法错误　　B. 输出 111

C. 不能输出　　D. 输出 222

(7)下列程序运行后输出(　　)。

```
#include <stdio.h>
int main()
{
    int m=1;
    if(m++>1)
        printf("%d",m);
    else
        printf("%d",m--);
    return 0;
}
```

A. 0　　B. 1　　C. 2　　D. 3

(8)若 x=4、y=-2、z=5,则表达式"++x-y+z++"的值为(　　)。

A. 10　　B. 11　　C. 12　　D. 13

(9)下列程序的输出结果为(　　)。

```
#include <stdio.h>
int main()
{
    int x=12;
    while(x--);
    printf("%d",x);
    return 0;
}
```

A. -1　　B. 0　　C. 11　　D. 1

(10)C 语言允许 if…else 语句嵌套使用,规定 else 总是和(　　)配对。

A. 之前最近的 if　　B. 第 1 个 if

C. 缩进位置相同的 if　　D. 之前最近的且尚未分配的 if

(11)下列程序执行后的输出结果是(　　)。

```
#include <stdio.h>
int main()
{
    int i=1,j=2,k=3;
    if(i++==1 &&(++j==3||k++==3))
        printf("%d  %d  %d",i,j,k);
    return 0;
}
```

A. 1　2　3　　B. 2　3　4　　C. 2　2　3　　D. 2　3　3

(12)下列叙述正确的是(　　)。

A. break 语句只能用于 switch 语句中

B. 在 switch 语句中必须使用 default

C. break 语句必须与 switch 语句中的 case 配对使用

D. 在 switch 语句中不一定使用 break 语句

(13)当从键盘输入 32 时,下列程序的输出结果为(　　)。

```
#include <stdio.h>
int main()
{
    int x,y;
    scanf("%d",&x);
    y=0;
    if(x>=0)
    {
        if(x>0) y=1;
    }
    else
        y=-1;
    printf("%d",y);
    return 0;
}
```

A. 0　　B. 1　　C. −1　　D. 不确定值

(14)对于下列程序的判断正确的是(　　)。

```
#include <stdio.h>
int main()
{
    int x=3,y=0,z=0;
    if(x=y+z)
```

```
        printf("****");
    else
        printf("####");
    return 0;
}
```

A. 有语法错误不能通过编译
B. 输出****
C. 可以通过编译,但是不能通过链接,因而不能运行
D. 输出####

(15)对于下列程序段的描述正确的是(　　)。

```
int a=10;
while(a=0) a=a-1;
```

A. 循环体语句执行 10 次　　B. 循环体语句一次也不执行
C. 循环是无限循环　　D. 循环体语句仅执行一次

(16)语句"while(!A);"中的表达式"!A"等价于(　　)。

A. A==0　　B. A==1　　C. A!=0　　D. A!=1

(17)下列程序的运行结果是(　　)。

```
#include <stdio.h>
int main()
{
    int a=1,b=2,c=4,t;
    while(a<b<c)
    {
        t=a; a=b; b=t; c--;
    }
    printf("%d,%d,%d",a,b,c);
    return 0;
}
```

A. 2,1,3　　B. 2,1,4　　C. 1,2,0　　D. 1,2,2

(18)下列程序的执行结果是(　　)。

```
#include <stdio.h>
void main()
{
    int a=0, i;
    for(i=1;i<5;i++)
    {
    switch(i)
        {
          case 0:
```

```
            case 3: a+=2;
            case 1:
            case 2: a+=3;
            default: a+=5;
        }
    }
    printf("%d\n",a);
}
```

A. 31　　B. 13　　C. 10　　D. 20

(19)对下列程序的描述正确的是(　　)。

```
#include <stdio.h>
int main()
{
    int a=3;
    do
    {
        printf("%d\n",a--2);
    }while(!(--a));
    return 0;
}
```

A. 输出 1　　B. 输出 1 和－2　　C. 输出 3 和 0　　D. 是死循环

(20)for(表达式 1;;表达式 3)可理解为(　　)。

A. for(表达式 1;0;表达式 3)

B. for(表达式 1;1;表达式 3)

C. for(表达式 1;表达式 1;表达式 3)

D. for(表达式 1;表达式 3;表达式 3)

(21)下列描述正确的是(　　)。

A. continue 语句的作用是结束整个循环的执行

B. break 语句和 continue 语句的作用相同

C. 只能在循环体内使用 continue 语句

D. 从循环嵌套中退出时,只能使用 goto 语句

(22)下列程序段不是死循环的是(　　)。

A.
```
int i=100;
    while(1)
    {
        i=i%100+1;
        if(i>100)  break;
    }
```

B.
```
for( ; ; );
```

C. int k=0;
```
    do { ++k; } while(k<=0);
```
D. int s=36;
```
    while(s); --s;
```
(23)下列程序的运行结果是(　　)。
```
#include <stdio.h>
int main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        if(i%2)
            printf("*");
        else
            continue;
        printf("#");
    }
    printf("MYM\n");
    return 0;
}
```
A. *#*#*#MYM　　B. #*#*#*
C. *#*#MYM　　D. #*#*
(24)下列程序的运行结果是(　　)。
```
#include <stdio.h>
int main()
{
    int i,j,x=0;
    for(i=0;i<2;i++)
    {
        x++;
        for(j=0;j<=3;j++)
        {
            if(j%2)
                continue;
            x++;
        }
        x++;
    }
    printf("x= %d\n",x);
```

```
    return 0;
}
```

A. x=4　　B. x=8　　C. x=6　　D. x=12

(25)若 w、x、y、z、m 均为 int 型变量,则执行以下语句后的 m 值是(　　)。

```
w=1;x=2;y=3;z=4;
m=(w<x)? w:x;
m=(m<y)? m:y;
m=(x<z)? m:z;
```

A. 1　　B. 2　　C. 3　　D. 4

(26)下列程序段(　　)。

```
for(t=1;t<=100;t++)
{
    scanf("%d",&x);
    if(x<0)
        continue;
    printf("%3d",t);
}
```

A. 当 x<0 时整个循环结束　　B. x>=0 时什么也不做

C. printf()函数永远不会被执行　　D. 最多允许输出 100 个非负整数

2)填空题

(1)在 C 语言中,break 语句只能用于_____语句和_____语句中。

(2)当 a=3,b=2,c=1 时,表达式"f=a>b>c"的值是_____。

(3)下列程序段的功能是从键盘输入的字符中统计数字字符的个数,用换行符结束循环。请填空。

```
int n=0, c;
c=getchar( );
while(_____)
{
    if(_____)  n++;
    c=getchar( );
}
```

(4)下列程序的功能是计算 2～100 的偶数的累加和。请填空。

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

```
    return 0;
}
```

(5)下列程序的输出结果是_____。

```
#include <stdio.h>
int main()
{
    int a=100,x=10,y=20,ok1=5,ok2=0;
    if(x<y)
            if(y!=10)
                if(!ok1)
                    a=1;
                else
                    if(ok2)
                        a=10;
    a=-1;
    printf("%d\n",a);
    return 0;
}
```

(6)下列程序运行后输出_____。

```
#include <stdio.h>
int main()
{
    int x=1,y=2;
    switch(x)
    {
        case 1:
            switch(y)
            {
                case 1: printf("%d",x);break;
                case 2: printf("%d",y);break;
            }
        case 2: printf("3");
    }
    return 0;
}
```

(7)下列程序的功能是找出整数的所有因子。请填空。

```
#include <stdio.h>
int main()
{
```

```
    int n,i;
    scanf("%d",&n);
    for(i=1;_______; i++)
    {
        if(_______)
            printf("%3d",i);
    }
    printf("\n");
    return 0;
}
```

(8)下列程序执行后的输出结果是_____。

```
#include <stdio.h>
int main()
{
    int a=3;
    do
    {
        printf("*");
        a--;
    }while(!a==0);
    return 0;
}
```

(9)下列程序的执行结果是_____。

```
#include <stdio.h>
int main()
{
    int i=1,n=0,s=1;
    do
    {
        n=n+s*i;
        s=-s;
        i++;
    }while(i<=9);
    printf("%d",n);
    return 0;
}
```

(10)下列程序运行后输出_____。

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

```
{
    int a,b=19;
    while(a=b-1)
    {
        b-=3;
        if(b%5==0)
        {
            a++;
            continue;
        }
        else if(b<5)
        break;
        a++;
    }
    printf("%d,%d\n",a,b);
    return 0;
}
```

(11)下列程序的运行结果是_____。

```
#include <stdio.h>
int main()
{
    int i=1;
    while(i<=15)
        if(++i%3!=2)
            continue;
        else
            printf("%d\t",i);
        printf("\n");
    return 0;
}
```

(12)下列程序的功能是打印 100 以内个位数为 6 且能被 3 整除的所有数。请填空。

```
#include <stdio.h>
int main()
{
    int i,j;
    for(i=0;_____;i++)
    {
        j=i*10+6;
        if(_____)
```

```
            continue;
        printf("%d",j);
    }
    return 0;
}
```

(13)下列程序的运行结果是_____。

```
#include <stdio.h>
int main()
{
    int i=5;
    do
    {
        switch(i%2)
        {
            case 4:i--;break;
            case 6:i--;continue;
        }
        i--;i--;
        printf("%d\t",i);
    }while(i>0);
    return 0;
}
```

(14)下列程序的功能是计算分数数列前 20 项的和:$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\cdots$。请填空。

```
#include <stdio.h>
int main()
{
    float s=0.0;
    int i,a,b,t;
    for(_____;i<=20;i++)
    {
        s+=_____;
        t=a+b;
        b=a;
        a=t;
    }
    printf("s=%.2f",s);
    return 0;
}
```

(15)下列程序的功能是根据公式 $e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots$求 e 的近似值,要求精度为 10^{-6}。

请填空。

```
#include <stdio.h>
int main()
{
    int i;
    double e,new;
    _______;
    new=1.0;
    for(i=1;_______;i++)
    {
        new/=(double)i;
        e+=new;
    }
    printf("e= %f\n",e);
    return 0;
}
```

3)编程题

(1)从键盘输入任意一个整数,判断其是否为偶数。

(2)有如下函数:

$$y=\begin{cases} x & (x<1) \\ 2x-1 & (1\leqslant x<10) \\ 3x-1 & (x\geqslant 10) \end{cases}$$

编写程序,使输入 x 时,输出相应的 y 值。

(3)输入一个整数,若该数能够被 3 和 7 整除,则输出 2;若只能被 3 和 7 中的任意一个整除,则输出 1;若不能被 3 和 7 中的任意一个整除,则输出 0。

(4)爱因斯坦的阶梯问题。爱因斯坦曾提出这样一道有趣的数学题:有一个长阶梯,若每步上 2 阶,最后剩 1 阶;若每步上 3 阶,最后剩 2 阶;若每步上 5 阶,最后剩 4 阶;若每步上 6 阶,最后剩 5 阶;只有每步上 7 阶,最后刚好一阶不剩。问该阶梯至少有多少阶? 编写程序解决这个问题。

(5)输出所有的水仙花数。水仙花数是一个 3 位整数,它的各位数字的立方和等于该数本身。

(6)编写程序,打印九九乘法表。

(7)编写程序,打印如下图形。

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

(8)用公式$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$求π的近似值,直到最后一项的绝对值小于$10^{-6}$为止。

(9)一个数如果恰好等于它的因子之和,就称其为完数。例如,6的因子为1、2、3,而6=1+2+3,因此6是完数。编写程序找出1 000以内的所有完数,并按下面格式输出其因子:6 its factors are 1,2,3。

(10)给出任意一个不多于5位的正整数,要求如下:

①求出它是几位数。

②分别输出每一位数字。

③按逆序输出各位数字。例如,原数为321,则输出123。