第 1 章　C 语言与程序设计

本章主要内容

- C 语言发展简史与特点
- 计算机解题过程
- 算法及其表示
- 常用算法介绍
- 结构化程序设计方法

众所周知，计算机系统由硬件和软件组成，程序是软件系统中不可或缺的重要组成部分，而程序是用计算机语言编写的。学习计算机语言和程序设计技术，就是要学会用计算机语言编写解决实际问题的程序，这对于软件开发人员来说是重要的一步。

指示计算机完成特定操作的命令称为指令，计算机硬件所有指令的集合称为该计算机的指令系统或机器语言。使用机器语言编写程序效率低且不方便，没有人直接使用机器语言编写程序。C 语言是一种既低级又高级的计算机程序设计语言，用 C 语言代替机器语言编写程序，可以达到机器语言程序的效果——程序执行速度快且占用存储空间小，同时 C 语言又是一种高级程序设计语言，编写程序的效率与其他高级语言并无差异。

学习“C 语言程序设计”主要包括两个方面：一是学习 C 语言的语法规则；二是学习程序设计方法。在掌握 C 语言语法规则的基础上，运用程序设计方法设计解决问题的算法，从而设计出解决问题的 C 语言程序。

学习计算机语言，当然也包括 C 语言，主要是学习语法和语义两个方面。计算机语言的语法是严格定义的，不允许有语法错误。语法是一些规定，如各种语句及各种语法成分都有各自的规定。同时计算机语言的语义是唯一的，没有二义性，各种语句及语法成分有各自的语义。只有掌握计算机语言的语法，理解语义，才能正确使用计算机语言编写没有语法错误和逻辑错误的程序。

程序设计是给出解决特定问题程序的过程，包括问题分析、算法设计、程序源代码设计、测试、调试和维护。源代码可以用 C 语言编写，也可以用其他计算机语言编写。程序设计的目的是得到一个指令序列(即程序)，提供给计算机来执行，使问题得到解决。在程序设计过程中，往往需要不同方面、不同领域的专业知识，包括应用领域知识、特定算法知识和形式逻辑知识等。

学习程序设计时，需要了解和掌握计算机的解题过程、算法的设计和表示、结构化程序设计等内容。同时需要用具体的计算机语言编写程序。在这个过程中，还需要学习和掌握程序的编辑、编译、调试和运行等有关操作。

“C 语言程序设计”是一门实践性很强的课程。在学习过程中，不仅要掌握基本概念、语法和语义等内容，还需要注重实验环节。通过自己动手编写程序，在完成 C 语言程序的编辑、编译、调试和运行的过程中，加深对 C 语言的理解，体会编写程序的过程及关键技术。与此同时，培养编程兴趣，提高编程能力。

通过学习 C 语言，可以为进一步学习其他计算机语言，包括目前常用的面向对象语言如 C++、Java 等各种计算机语言奠定基础。

1.1 C 语言发展简史

C 语言是一种广泛使用的面向过程的计算机程序设计语言，既适用于系统程序设计，又适用于应用程序设计。C 语言在操作系统、软件工程、图形学、数值分析等诸多领域得到了广泛的应用。

早期的计算机语言，除汇编语言之外，高级程序设计语言有 FORTRAN 语言、ALGOL 语言和 COBOL 语言等。后来出现了大量的各种各样的高级语言，较有影响的有 BASIC 语言、PASCAL 语言、C 语言、C++语言和 Java 语言等。C 语言的发展历程大致如图 1-1 所示。

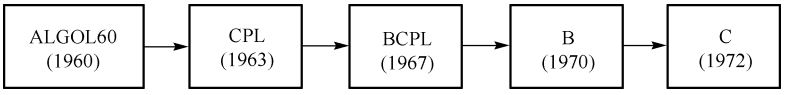


图 1-1　C 语言的发展历程

1960 年出现的 ALGOL60 很受程序设计人员欢迎，用 ALGOL60 描述算法很方便，但硬件操作能力较弱，不宜用来编写系统程序。1963 年，英国剑桥大学在 ALGOL 语言的基础上推出了 CPL(复合程序设计语言)，增强了直接对硬件的处理能力，但由于规模大，因此实现困难。1967 年剑桥大学的马丁·理查德(Matin Richards)对 CPL 语言进行了简化，推出了 BCPL(基本复合程序设计语言)。1970 年，美国贝尔实验室的肯·汤普逊(Ken Thompson)对 BCPL 语言进一步简化，突出了硬件处理能力，取名 B 语言(BCPL 的第一个字母)，并用 B 语言写了第一个 UNIX 操作系统程序，但 B 语言过于简单，功能有限。1972 年，贝尔实验室的丹尼斯.M.里奇(Dennis M. Ritchie)对 B 语言进行了完善和扩充，保留了 B 语言的硬件处理能力，扩充了数据类型，强调了通用性，这就是 C 语言(BCPL 的第二个字母)。1973 年，肯·汤普逊和丹尼斯.M.里奇两人合作，用 C 语言重写了 UNIX 操作系统，并在 PDP-11 计算机上加以实现。C 语言伴随着 UNIX 操作系统成为了一种最受欢迎的计算机程序设计语言。

UNIX 操作系统因简洁、实用和高效率而受到人们欢迎，C 语言功不可没。1977 年出现了不依赖于具体机器的 C 语言编译版本“可移植 C 语言编译程序”，使 C 语言移植到各种不同的机器变得非常简单。1978 年，贝尔实验室的布朗.W.卡尼汉(Brian W.Kernighan)和丹尼斯.M.里奇(Dennis M. Ritchie)合著了 *The C Programming Language* 一书，对 C 语言的语法进行了规范化的描述，成为了以后广泛使用的 C 语言的基础。随着微型计算机的普及，产生了各种不同的 C 语言版本，为了统一标准，美国国家标准学会(ANSI)于 1983 年

制定了 C 语言标准，这就是 ANSI C 标准。1990 年，C 语言成为国际标准化组织(ISO)通过的标准语言，这一 C 语言版本称为 C89 或 C90。1999 年通过的 ISO/IEC9899:1999 新标准称为 C99。

目前流行的 C 语言编译系统大多是以 ANSI C 为基础进行开发的，但不同版本的 C 编译系统所实现的语言功能和语法规则略有差异。在实际编写程序过程中，应该了解和注意 C 语言的标准，使编写的程序具有可移植性，可以运行于不同的计算机系统。

1.2　C 语言的特点

C 语言是一种通用的程序设计语言，语言本身简洁、灵活、表达能力强，被广泛用于系统软件和应用软件的开发，并且具有良好的可移植性。

C 语言的特点可概括如下。

(1) C 语言简洁、紧凑、灵活。C 语言的核心内容很少，只有 32 个关键字、9 种控制语句；程序书写格式自由，压缩了一切不必要的成分。

(2) 表达方式简练、实用。C 语言有一套强有力的运算符，达 44 种，可以构造出多种形式的表达式，用一个表达式就可以实现其他语言可能需要多条语句才能实现的功能。

(3) 具有丰富的数据类型。数据类型越多，数据的表达能力就越强。C 语言具有现代语言的各种数据类型，如字符型、整型、实型、数组、指针、结构体和共用体等，可以实现诸如链表、栈、队列、树等各种复杂的数据结构。其中，指针类型使参数的传递简单、迅速，同时节省内存空间。

(4) 具有低级语言的特点。C 语言具有与汇编语言相近的功能和描述方法，如地址运算和二进制数位运算等，还可以对硬件端口等资源进行直接操作，充分使用计算机的资源。C 语言既具有高级语言便于学习和掌握的特点，又具有机器语言或汇编语言对硬件的操作能力。因此，C 语言既可以作为系统描述语言，又可以作为通用的程序设计语言。

(5) C 语言是一种结构化语言，适合大型程序的模块化设计。C 语言提供了编写结构化程序的基本控制语句，如 if-else 语句、switch 语句、while 语句和 do-while 语句等。C 语言程序是函数的集合，函数是构成 C 语言程序的基本单位，每个函数具有独立的功能，函数之间通过参数传递数据，程序员可以编写自己的函数。同时，不同操作系统的编译器都为程序员提供了大量的标准库函数，例如输入/输出函数、数学函数和字符串处理函数等。灵活地使用标准库函数可以简化程序设计，提高编写程序效率。

(6) 各种版本的编译器都提供了预处理命令和预处理程序。预处理扩展了 C 语言的功能，提高了程序的可移植性，为大型程序的调试提供了方便。

(7) 可移植性好。程序从一个环境不经改动或稍加改动就可以移植到另一个完全不同的环境中运行。这是因为标准库函数和预处理程序将可能出现的与机器有关的因素和源程序分隔开来，使针对不同的计算机硬件环境，都可以重新定义有关的内容。

(8) 生成的目标代码质量高。由 C 源程序编译和链接得到的目标代码的运行效率比汇编语言代码的运行效率也只低 10%～20%，可充分发挥机器的效率。

(9) C 语言语法限制不严，程序设计自由度大。C 语言程序在运行时不做诸如数组下标越界和变量类型兼容性等检查，而是由编程者自己保证程序的正确性。C 语言几乎允许所有数据类型的转换，字符型和整型可以自由混合使用，所有类型均可作为逻辑型，可以自己定义新的类型，还可以把某类型强制转换为指定的类型。实际上，这使编程者有了更大的自主性，能编写出灵活、优质的程序，同时也给初学者增加了一定的难度。所以，只有在熟练掌握 C 语言程序设计后，才能体会到其灵活性。

现在，可以先简单了解以上特点，等掌握 C 语言之后，再次阅读就会有更为深刻的领会。

C 语言有许多优点，是一种优秀的计算机语言，但也存在一些缺点。了解这些缺点，有助于实际使用 C 语言编写程序时做到扬长避短。

(1) C 语言程序的错误更隐蔽。C 语言的灵活性使用它编写程序时更容易出错，而且 C 语言的编译器不检查这样的错误。与汇编语言类似，需要程序运行时才能发现这些逻辑错误。C 语言还会有一些隐患，需要引起程序员的重视，如将比较的“= =”写成赋值的“=”，虽然语法上没有错误，但是这样的逻辑错误往往不易发现，想要找出错误往往十分费时。

(2) C 语言程序有时会难以理解。C 语言语法成分相对简单，是一种小型语言。但是，其数据类型多，运算符丰富且结合性多样，使对其理解有一定的难度。有关运算符的优先级和结合性，可以用人们最常说的一句话“先乘除，后加减，同级运算从左到右”来理解，但优先级更多且有两个运算方向(从左到右和从右到左)，比较而言，C 语言的运算符及表达式更复杂一些。为了减少字符输入，C 语言比较简明，同时也使 C 语言可以写出常人几乎难以理解的程序。

(3) C 语言程序有时会难以修改。考虑到程序规模的大型化或者巨型化，现代编程语言通常会提供“类”和“包”之类的语言特性，这样的特性可以将程序分解成更加易于管理的模块。然而 C 语言缺少这样的特性，维护大型程序显得比较困难。

1.3 计算机解题过程

使用计算机解决现实世界的问题是一种必然的选择。如何使用计算机解题？也就是如何编写解决问题的程序或软件？这是一个比较复杂的问题，需要使用软件工程的方法来解决，超出本书的范围。这里从学习 C 语言的角度简要认识计算机解题的过程。

计算机的解题过程如图 1-2 所示，大致分为 4 个阶段，分别是分析问题、设计算法、编写程序和运行验证。

1．分析问题

详细分析需要解决的问题，清楚了解问题的需求。已知的原始数据有哪些？使用什么方法或数学模型？要得到什么结果？分析问题就是明确做什么的过程(What to do)。

2．设计算法

明确了做什么之后，就需要考虑怎么做(How to do)。通常将解决问题的方法或数学模

型转换为解决问题的步骤，即设计算法。设计算法是计算机解题过程中的一个具有创造性的重要环节，设计的算法是否合理、正确，直接关系到问题能否解决。同时，解决问题的方法往往不是唯一的，可能有多种解决方法，需要设计出多种算法，通过分析比较，从中找出合适的或最优的算法。

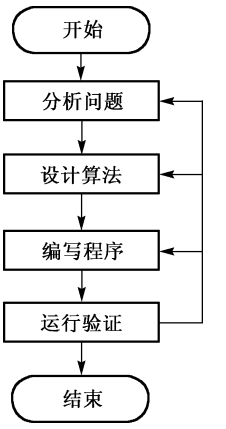


图 1-2　计算机解题过程

3．编写程序

确定了解决问题的算法之后，需要使用一种计算机程序设计语言编写程序。编写程序就是将设计的算法等价映射(转换)为计算机程序，所编写的程序从逻辑上看是算法的一种表现形式。

4．运行验证

编写的程序有时不能正确地满足解决实际问题的需求，还需要调试，即在计算机上运行并且排除潜在错误。必要时，还要使用测试数据对程序进行测试，验证程序的正确性。如果程序测试不充分，则有可能导致潜在错误的存在。除此之外，问题需求也可能随着时间的推移而变化，使程序使用一段时间后还需要进行修改和完善，这个过程称为维护。

1.4　算法及其表示

大型软件的开发一般采用软件工程的方法，经过分析、设计、编码、调试和运行等阶段。每个阶段都有明确的任务，后一阶段的工作是在前一阶段结果的基础上进行的。对于初学者而言，需要解决的问题和编写的程序一般比较简单，只要将问题分析清楚，找到解决问题的方法，编写程序不是一件困难的事情。

C 语言解题时，在程序中有两方面的描述，即数据描述和处理步骤(算法)描述，后者处理前者的数据。因此，编写 C 语言程序之前，应该将问题所涉及的数据、已知条件和问题的解决步骤分析清楚。下面通过两个简单的实例来说明这一点。

【例 1-1】 求任意两个数的和与平均值。

(1) 数据描述

问题中有 4 个数据，即两个任意数、任意数之和与它们的平均值。在程序中定义 4 个浮点型变量存储这些数据，如：

```
float a, b, sum, average;
```

(2) 处理步骤描述

第 1 步：输入两个任意数，存储在变量 a 和 b 中。

第 2 步：计算两个数的和与平均值，存储在变量 sum 和 average 中。

第 3 步：输出变量 sum 和 average 的值，即得到问题所要求的结果。

根据上述分析，编写程序如下：

```
#include <stdio.h>
void main()
{   float a, b, sum, average;                    /* 定义变量 */
    scanf("%f,%f", &a, &b);                      /* 输入两个实型数 */
    sum = a + b;                                 /* 求两个数的和 */
    average = (a + b) / 2;                       /* 求两个数的平均值 */
    printf("sum=%f,average=%f\n", sum, average);  /* 输出结果 */
}
```

【例 1-2】 某体育比赛中，有 10 名裁判为参赛选手打分，参赛选手最后得分的计算方法是：去掉一个最高分和一个最低分后其他分数的平均值。求参赛选手的最后得分。

(1) 数据描述

问题中的原始数据有 10 个，解题过程中需要计算最高分、最低分和最后得分。在程序中可定义一个数组 s 存储 10 个分数、三个浮点型变量 max、min、score 分别用来存储最高分、最低分和最后得分，另外还需要若干辅助变量。

(2) 处理步骤描述

第 1 步：输入 10 个分数，存储在数组 s 中。

第 2 步：求 10 个数的最高分、最低分以及它们的和，并存储在变量 max、min 和 sum 中。

第 3 步：从 sum 中减去 max 和 min 并且除以 8(10–2)，求得最后得分，并将其存储在变量 score 中。

第 4 步：输出变量 score 的值，则得到问题所要求的结果。

对于第 1 步和第 2 步还需给出更详细的步骤，以便程序实现。根据上述分析，编写程序如下：

```
#include <stdio.h>
void main()
{   float s[10],max, min, sum, score; /* 定义变量 */
    int i;
    for(i=0;i<10;i++)                 /* 输入 10 个数 */
        scanf("%f", &s[i]);
    max = min = sum = s[0];           /* 求最高分、最低分及分数和 */
    for(i=1;i<10;i++)
    {
        if(max<s[i]) max=s[i];
        if(min>s[i]) min=s[i];
        sum+=s[i];
    }
    score=(sum - max - min )/ 8;      /* 求最后得分 */
    printf("score=%.4f", score);      /* 输出结果 */
}
```

上述两个例子中的处理步骤就是解决相应问题的算法。很显然，有了算法，写出计算机语言程序就容易多了。

使用计算机解题时，解题步骤可归纳如下。

(1) 了解题目要做什么。

(2) 明确处理的数据及数据特征，确定数据结构。程序处理的对象是数据，程序中如

何存储数据？输入的数据有哪些？输出的结果又有哪些？这些都应该明确并且做到心中有数；而选择适当的数据结构，对于设计处理步骤是至关重要的。

(3) 清楚地描述对数据的处理步骤——算法设计，以便得到预期的结果。算法设计是解决问题的核心，是程序的灵魂，是程序设计过程中的一个重要环节。

(4) 对处理步骤进一步细化——逐步求精算法，直到能够使用程序设计语言编写程序实现。算法设计和逐步求精算法是使用计算机解题过程中最为困难的一步，是唯一的有创造性的工作。

(5) 将算法转换为程序。将算法转换为程序是一项包含众多技巧的工作，需要完整的计算机程序设计语言的知识。

(6) 运行程序，分析结果。通过对源程序进行编辑、编译和链接，得到可执行的程序，并且运行所得到的可执行程序。如果出现编译错误或运行结果不正确，则要修改源程序，重新进行编译、链接和运行，直到得到满意的结果。

对于一个复杂的问题，可以将其分解成若干个容易处理、可单独解决的子问题，然后分别加以解决。

1.4.1　算法的概念

算法是精确定义的一系列规则的集合，这些规则规定了解决特定问题的一系列操作，以便在有限的步骤内产生出问题的答案。【例 1-1】和【例 1-2】中的处理步骤就是一种用自然语言描述的算法。

编者注：本书算法数字语言描述部分，按出版规范，单字母变量应采用斜体表示，但为与对应代码一致，均采用正体表达。

【例 1-3】求两个正整数 m 和 n 的最大公约数(即同时能够整除 m 和 n 的最大正整数)。

欧几里得阐述了求两个数的最大公约数的过程——欧几里得算法。

第 1 步：以 n 除 m，并令 r 为所得余数(r=m mod n，显然 0≤r＜n)。

第 2 步：若 r＝0，算法结束，n 为 m 和 n 的最大公约数。

第 3 步：若 r≠0，令 m←n，n←r，返回第 1 步继续。

算法中的符号←表示将符号右边变量或表达式的值送到左边的变量中。

按照上述计算步骤，给定任意两个正整数 m、n，经过第 1 步和第 3 步计算总能不断缩小 m、n 的值，使 r 的值为 0，算法结束，得到最大公约数。

算法中不仅各步骤间的顺序是重要的，在每步内的动作次序也同样重要。例如，上述算法的第 3 步中，“令 m←n，n←r”绝对不能写成“令 n←r，m←n”，因为这样做的结果是在变量 m、n 和 r 中存储了同一个值，即都变成了第 1 步除法运算的余数。

算法是为解决一个特定的问题所采取的确定的有限步骤。它告诉计算机如何一步一步地进行计算，直至解决问题。对于一个给定的问题，可以有不同的解题方法，即可以有不同的算法。如【例 1-2】求参赛选手的最后得分问题，可以先将 10 个数排序，求中间 8 个数的平均值，还可以有其他方法。为了有效地解题，不但要求算法正确，还要考虑算法的质量，通常选择简单明了、结构清晰、计算高效的算法。

算法由操作和数据组成，而这些操作又是按照一定的控制结构所规定的次序执行的。因此，算法由数据、操作和控制结构三要素组成。

数据是指操作对象和操作结果。由于算法层出不穷、千变万化，其操作对象数据和操作结果数据名目繁多、不胜枚举。最基本的有字符、整数、实数和布尔值等；稍复杂的有向量、矩阵和记录等；更复杂的有集合、树、图，还有声音、图形和图像等。

算法中，每一步都能分解成计算机的基本操作，否则算法就是不可行的。算法的操作种类也是五花八门、多姿多彩。最基本的有赋值运算、算术运算、关系运算和逻辑运算等；稍复杂的有函数运算、向量运算等，更复杂的有表、栈、队列、树和图的运算等。

算法的控制结构给出了算法的框架，决定了各种操作的执行次序。任何复杂的算法都可以用顺序结构、分支结构和循环结构 3 种控制结构组合而成。

求解问题时，算法可以千变万化、简繁各异，但是算法必须具有以下特性。

- 有穷性：算法在执行了有限步骤之后结束，并且每一步都可以在有穷的时间内完成。
- 确定性：算法中每种操作必须有确切的含义，即无二义性。同时，无论如何算法都只有唯一的一条执行路径，即相同的输入只能得出相同的输出。
- 可行性：算法中描述的操作都可以通过已经实现的基本操作执行有限次数来实现。
- 输入：有零个或多个输入，即算法需要的初始数据。
- 输出：有一个或多个输出，输出的是与输入有某些特定关系的数据，没有输出的算法是无意义的。

1.4.2 算法的描述

为求解问题设计了算法后，为了与人交流，让他人能看懂、理解算法，需要用一定的方法进行描述。算法有多种描述方法，常用的描述方法如下。

(1) 自然语言描述。
(2) 传统流程图。
(3) N-S 流程图。
(4) 伪代码。
(5) 程序设计语言。

1. 自然语言描述

【例 1-2】和【例 1-3】介绍的算法就是用自然语言描述的。自然语言可以是中文、英文、其他民族语言或数学表达式等。用自然语言描述算法通俗易懂，其缺点是文字有可能冗长，不太严格，容易产生歧义，表达分支和循环结构不方便等。

2. 传统流程图

流程图是用约定的图框和流程线表示运算或操作流程的图示形式，在图形上使用简明的文字和符号表示各种不同性质的操作，用流程线表示算法的执行方向。其优点是直观形象、易于理解。美国国家标准学会 ANSI 规定的一些常用流程图符号如图 1-3 所示。

【例 1-3】中的欧几里得算法的传统流程如图 1-4 所示。

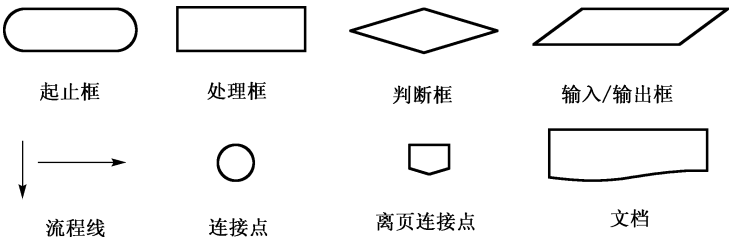


图 1-3　传统流程图常用符号

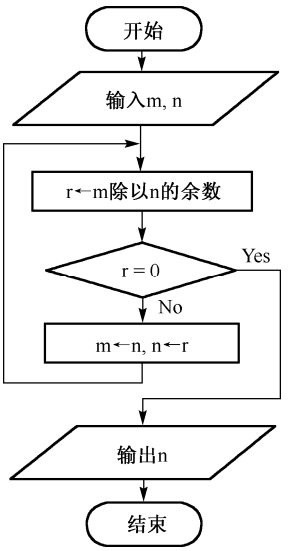
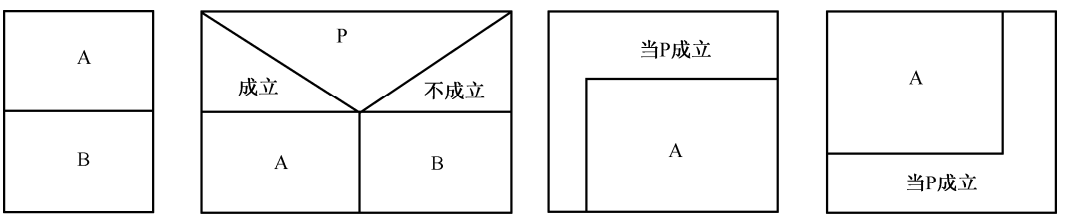


图 1-4　欧几里得算法的传统流程图

3．N-S 流程图

N-S 流程图是一种结构化流程图，适合于表示结构化算法。N-S 流程图完全取消了带箭头的流程线，全部算法写在一个矩形框内，框内还可以包含从属于它的框。

N-S 流程图使用的流程图符号如图 1-5 所示，图中 A 和 B 为某种操作或功能模块，P 为判断条件。



(a) 顺序结构　(b) 选择结构　(c) 循环结构：while　(d) 循环结构：do-while

图 1-5　N-S 流程图基本结构

（1）顺序结构：A 操作和 B 操作按顺序进行，先进行 A 操作，再进行 B 操作。

（2）选择结构：根据判断条件 P 是否成立，在 A 和 B 两种操作中选择一种。若判断条件 P 成立，选择 A 操作执行；否则，选择 B 操作执行。

（3）循环结构：根据判断条件 P 是否成立，决定 A 操作是否被重复执行。C 语言中的循环结构分为 while 循环和 do-while 循环。while 循环是先判断条件 P，若成立，则重复 A 操作；do-while 循环是先执行 A 操作，再判断条件 P，若成立，则重复 A 操作。

【例 1-3】中的欧几里得算法的 N-S 流程如图 1-6 所示。

由于图 1-4 中的重复操作不符合结构化特征，需要对操作步骤进行调整。图 1-6(a)表示的算法使用 do-while 循环控制，循环结束时，n 和 r 的值均为 0，结果为 m 的值；图 1-6(b)表示的算法使用 while 循环控制，需在循环之前先求一次余数存储在 r 中，循环结束时，结果为 n 的值。

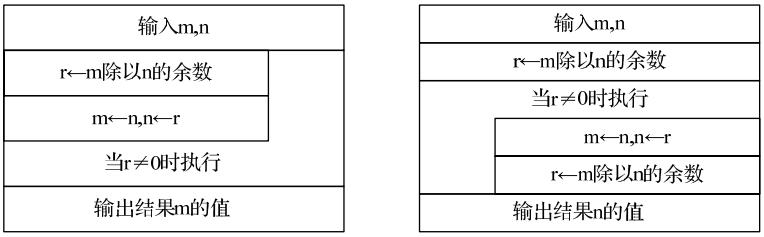


图 1-6　求最大公约数的 N-S 流程图

4．伪代码

伪代码是一种用于表达算法的语言，与程序设计语言相似，但更简单易学。程序表达算法的目的是在计算机上执行，而伪代码表达算法的目的是便于理解。伪代码易于阅读、简单、结构清晰，介于自然语言和程序设计语言之间，不拘泥于程序设计语言的具体语法和实现细节。伪代码表示算法不用图形符号，比使用传统流程图和 N-S 流程图表示算法更方便、紧凑并且易懂，便于将算法转换为各种计算机语言的程序。

由于伪代码在语法结构上的随意性，目前并不存在一个通用的伪代码语法标准。人们往往以具体的高级程序设计语言为基础，简化后进行伪代码的编写。最常见的这类高级程序设计语言包括 C、Pascal、Java 等，由此而产生的伪代码往往称为“类 C 语言”“类 Pascal 语言”等。

欧几里得算法的伪代码(类 C 语言)如下(与图 1-6(a)描述的算法一致)。

```
算法开始
输入 m,n;
do {
    r←以 n 除 m 的余数;
    m←n;
    n←r;
} while(r≠0);
输出 m;
算法结束
```

可以看出，用伪代码表示的算法与计算机程序设计语言已经很接近，很容易将其改写成计算机程序。对计算机程序设计语言比较熟悉时，使用伪代码(如伪 C 语言)表示算法易于转换为实际的计算机语言程序。

5．程序设计语言

程序设计语言描述是指用计算机高级语言如(C、Java、VB 等)描述算法，它是可以在计算机上运行并获得结果的算法描述，通常也称程序。

欧几里得算法的 C 语言描述如下(与图 1-6(a)描述的算法一致)。

```
//gcd 函数，功能是求正整数 m、n 的最大公约数
int gcd(int m,int n)
```

```
{
    int r;
    do{
        r=m%n;          // %是 C 语言中的求余运算符
        m=n;
        n=r;
    }while(r!=0);    // !=是关系运算符，其含义是不等于
    return m;
}
```

1.5　常用算法策略介绍

算法是解决问题的方法，不同的领域有不同的方法。根据问题领域的不同，算法可分为数值问题的算法和非数值问题的算法。数值问题的算法解决传统的数学问题，例如求方程解的算法和求定积分算法等。与数值问题的算法相比，计算机科学对非数值问题的算法给予了更多的重视。计算机在非数值领域的应用十分广泛，不同的应用领域有不同的非数值问题算法，已远远超过了其在数值问题方面的应用。

针对一个给定的问题，要找到行之有效的算法，就需要了解或掌握一些基本的算法设计策略，这些算法设计策略已潜移默化地应用在生活的很多方面。基本的算法策略主要有穷举法、递推法(迭代法)、递归法、回溯法、分治法、贪心法和动态规划法等。本节简要介绍前 3 种方法，程序设计中常常用到这些算法策略，其他的算法策略请读者自行阅读算法相关的书籍。

1.5.1　穷举法

穷举法又称枚举法、试凑法或蛮力法。该方法通过逐一考察问题的所有可能解，找出问题真正的解。枚举法要求问题的可能解必须是有限的，而且这些可能解是已知的。穷举法通常用于解决“是否存在”或“有多少种可能”等类型的问题。

【例 1-4】 给定一个正整数，确定它的整数立方根是否存在，若存在，则找出这个立方根。

问题分析：一个正整数的整数立方根如果存在，一定在 1 和这个数之间，而这之间正整数的个数是有限的，所以可以设计一个基于枚举法的算法。

```
算法开始
输入一个正整数赋值给 n;
x←1;
while(x≤n 且 x*x*x≠n){
    x←x+1;
}
if(x≤n)
    找到 n 的整数立方根，输出 x 的值;
else
    输出 n 的整数立方根不存在信息;
算法结束
```

1.5.2 递推法

递推法是从已知的初始条件出发，根据递推公式算出新值，再用新值代替旧值，用同样的递推公式求得另一个新值，以此类推，经过有限次递推即可求得最终结果。在理想状态下，每递推一次，结果逐渐接近问题的最后解。

递推法在数值算法中又称迭代法。迭代法常用于求近似解的问题，根据对前一步结果误差的不同处理方法，迭代法又有逼近迭代和试探迭代等不同方法。数值计算要注意解的稳定性问题，即在迭代中每一步的解越来越接近真正的解，否则迭代不会成功。

【例 1-5】 计算一个正整数 N 的阶乘。

根据数学知识 N!=1×2×3×…×N，为了求得 N!，可以从 1 开始，求得 1 的阶乘后再求 2 的阶乘，再和 3 相乘得 3 的阶乘，以此类推，最后求得 N 的阶乘。

```
算法开始
输入一个正整数赋值给 N;
t←1;
i←1;
while(i≤N){
    t←t×i;
    i←i+1;
}
输出结果 t;
算法结束
```

1.5.3 递归法

与递推法类似，递归法是利用大问题与其子问题的递推关系来解决问题，通常把一个问题层层转化为一个与原问题相同或相似，但规模较小的子问题来求解。

在算法或程序设计中，具体的处理办法是一个过程(或函数)在其定义或说明中又直接或间接地调用自身。一个直接或间接调用过程(或函数)自身的算法称为递归算法，一个函数如果调用自身进行计算则称该函数为递归函数。一些问题的算法描述中，递归法往往比非递归法更加简洁易懂。

【例 1-6】 计算一个正整数 N 的阶乘。

阶乘函数 f 的递归定义为

f(1)=1　　(1!=1，N=1 时)

f(N)=N×f(N–1)　　(N!=N×(N–1)!，如果 N>1)

前一式子给出函数的初始值，初始值是非递归定义的。每个递归函数都应该给出非递归定义的初始值，否则这个函数无法计算。后一式子用较小自变量的函数值来表达较大自变量的函数值，在定义式的两边都引用了阶乘函数 f，所以是一个递归定义。下面是求阶乘的 C 语言递归函数。

```
//f 函数，功能是求正整数 N 的阶乘
int f(int N)
{
    int t;                    //用 t 保存阶乘值
    if(N==1||N==0) t=1;       // ||是 C 语言中的逻辑运算符，含义是或者
    else t=N*f(N-1);          //调用 f 函数自身
    return t;
}
```

上面介绍的算法策略在后续的程序设计中经常使用，希望读者能仔细体会和掌握。

1.6　结构化程序设计方法

编写程序如果没有一整套规范，就会造成程序可读性差、无法维护等问题。结构化程序设计方法有助于解决这个问题。对于过程性的计算机语言来说，使用结构化程序设计方法可以提高程序的可阅读性和可修改性，提高编写程序的效率，缩短修改程序的时间。

1.6.1　结构化程序设计基本思想

结构化程序设计是由 E.W.Dijikstra 在 1965 年提出的，对程序设计及软件开发产生了深远的影响。结构化程序设计以功能模块和过程设计为主，其要点如下。

（1）自顶向下，逐步求精

自顶向下是指对设计的系统有一个全面的了解，从问题的全局入手，把一个复杂的问题分解成若干独立的子问题，然后对每个子问题再做进一步的分解，如此反复，直到每个子问题都容易解决为止。

逐步求精是指程序设计是一个渐进的过程，先把问题用一个程序模块来描述，再把每个模块的功能逐步分解细化为一系列的具体步骤，直到能用计算机语言的语句来实现。逐步求精与自顶向下结合使用，一般把逐步求精视为自顶向下设计的具体实现。

（2）模块化

将复杂的问题分解成多个相对简单的子问题来解决，即是将程序模块化的过程。划分出的子模块一定要有相对独立的功能。模块化是指程序设计应从功能入手，一个大型的程序经过层层分解后，可以细化为一个个子模块。

（3）结构化

在每个模块内只使用顺序、选择和循环 3 种基本控制结构。3 种基本结构通过有序组合或嵌套可以描述复杂的算法。

结构化程序设计要求程序中的每个基本结构都是单入口、单出口的，没有死循环，没有死语句，即没有绝对不会被执行的语句。

C 语言提供了与 3 种基本控制结构相对应的语句，可以方便地编写结构化的程序。有关结构化程序设计更详细的内容，请参考有关的书籍和资料。

1.6.2 三种基本程序结构

编写 C 语言程序时，应该按照结构化程序设计的基本思想设计程序，在程序中尽量使用顺序、选择和循环这三种基本结构，避免使用 goto 语句，避免编写的程序是多入口或多出口的。

顺序结构的程序流程如图 1-7 所示，选择结构的程序流程如图 1-8 所示，循环结构的程序流程如图 1-9 所示。

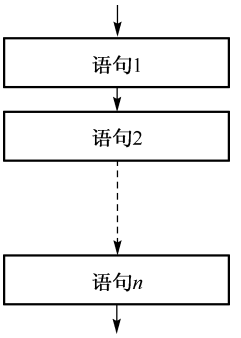
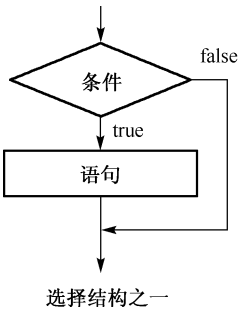


图 1-7 顺序结构的程序流程



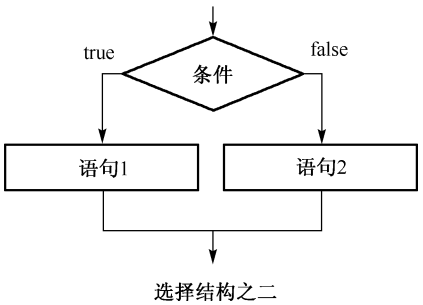
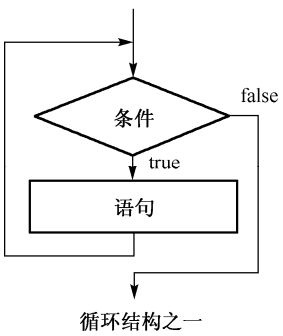


图 1-8 选择结构的程序流程



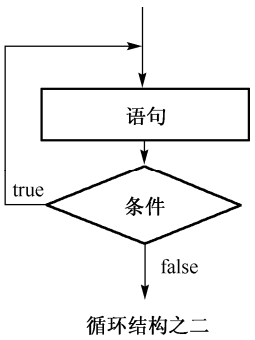


图 1-9 循环结构的程序流程

本章小结

本章主要介绍了以下内容。

(1) C 语言发展简史

简要介绍了 C 语言的发展历程，C 语言与 UNIX 操作系统的关系，C 语言的标准化。

(2) C 语言的特点

C 语言的特点主要包括：简洁性，表达式简练实用，丰富的数据类型，具有低级语言的特点，结构化的语言，可移植性强，代码效率高等。

(3) 计算机解题过程

简要介绍了计算机的解题过程，包括分析问题、设计算法、编写程序和运行验证。

(4) 算法及其表示

介绍了算法的基本概念以及 5 种算法的表示方法。这些算法的表示方法包括自然语言

描述、传统流程图、N-S 流程图、伪代码和程序设计语言。

（5）常用算法策略介绍

介绍了最常用的算法策略，包括枚举法、递推法和递归法。

（6）结构化程序设计方法

简要介绍了结构化程序设计的基本思想，以及结构化程序设计中的三种基本结构。

习题一

一、简答题

1．什么是程序？什么是程序设计？

2．利用计算机解题的过程大致分为哪几个阶段？每个阶段的任务是什么？

3．什么是算法？算法有哪些特性？

4．分析图 1-6 的流程图，(a)图的输出结果是 m 的值，(b)图的输出结果是 n 的值，为什么？

5．欧几里得算法使用了哪种算法策略？

6．简述结构化程序设计的基本思想。

7．简述在计算机上运行一个 C 程序的过程。

二、设计题

1．参考图 1-6(b)流程图，试写出对应的伪代码算法。

2．参考【例 1-2】，编写一个 C 语言程序，输入三个整数，输出它们的最大值。

3．用传统流程图或 N-S 流程图表示下列各题的算法。

（1）交换两个存储单元 a 和 b 的内容。

（2）判断一个整数 n 能否同时被 3 和 7 整除。

（3）求方程：$ax^2+bx+c=0$ 的根。假设 $b^2-4ac\geqslant 0$，要区分两个不相等的根和两个相等的根。

（4）求 1+2+3+…+100。

（5）鸡兔同笼，已知鸡兔共有 30 个头，有 80 只脚，问鸡兔各有多少只(用枚举法)。

（6）菲波那契数列的前两项是 1，从第 3 项开始，每项的值是前两项的和，即该数列为 1,1,2,3,5,…，求该数列的第 20 项(用递推法)。