第一部分

计算机基础知识

视频讲解

第 1 章

计算与计算机系统基础知识

学习目标和内容

本章主要介绍计算机系统的基本构成与功能，计算机中的信息表示、算法的概念等基础知识，在学习完本章后，你可以：

- 理解计算机系统的基本概念和结构；
- 掌握计算机中的信息表示方法；
- 掌握计算机解决问题的基本步骤；
- 理解算法以及如何评价算法效率。

1.1 计算与计算机系统

1.1.1 计算

从小学开始，“计算”这个词就不断出现在日常生活、数学作业中，例如，对“苹果 18 元一千克，算一下买 3 千克苹果要多少钱?”这个问题，一般可用两种方法进行解答：一是 3 个 18 相加，二是 18 乘以 3。对第一种方法，通常列出竖式，个位与个位对齐，十位与十位对齐，然后将个位上的 3 个 8 相加，得到 24，直接在结果的个位写上 4，2 进位到十位，与 3 个 1 相加得到 5，结果为 54。对第二种方法，也可列竖式，首先将 18 的个位 8 与 3 相乘，得到 24，将 4 写到结果的个位上，2 进位到十位，十位的 1 与 3 相乘得 3，与进位的 2 相加得 5，结果也为 54。当然，这样的问题也可直接用计算器求解，输入 18×3 就能得到结果。

从这个例子，可以看出计算的一些特性，给出如下定义：计算指的是在某计算装置上，根据已知条件，从某一初始点开始，在完成一组良好定义的操作序列后，得到预期结果。

对这个定义，有以下两点需要注意。

(1) 计算的过程可由人或某种计算装置执行。

(2) 同一个计算可由不同的技术实现。

在人类历史上，计算的作用受到了人脑运算速度和手工记录计算结果的制约，使得能通过计算解决的问题规模非常小。相对于制约计算的人的因素，计算机(Computer)非常擅长于做(也只能做)两件事情：运算和记住运算的结果。随着计算机的出现，以及计算机运算速度的不断提高，能通过计算解决的问题越来越多，问题规模越来越大，即越来越多的问题被证明存在计算的解(Computational Solution)。所谓有计算的解，指的是对某个问题，能通过定义一组操作序列，按照该操作序列行为得到该问题的解。

1.1.2 计算机系统

一般来说，计算机是一种可编程的机器，它接收输入、存储并且处理数据，然后按某种有意义的格式进行输出。可编程指的是能给计算机下一系列的命令，并且这些命令能被保存在计算机中，并在某个时刻能被取出执行。

通常所说的计算机实际上指的是计算机系统，它包括硬件和软件两大部分。硬件系统指的是物理设备，包括用于存储并处理数据的主机系统，以及各种与主机相连的用于输入和输出数据的外部设备，如键盘、鼠标、显示器和磁带机等。计算机的硬件系统，是整个计算机系统运行的物理平台。计算机系统要能发挥作用，仅有硬件系统是不够的，还需要具备完成各项操作的程序，以及支持这些程序运行的平台等条件，这就是软件系统。一个实际的计算机系统通常由图 1.1 所示的结构构成。

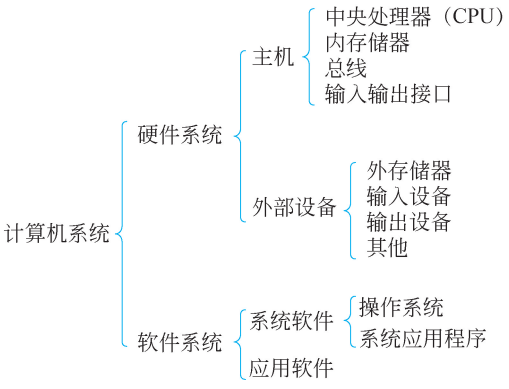

图 1.1 计算机系统的构成

目前，占主流地位的计算机硬件系统结构是冯·诺依曼体系结构，由美国科学家冯·诺依曼等在 1946 年提出。在此之前出现的各种计算辅助工具，如差分机等，其用途是固定的，即各种操作是在制造机器的时候就固定下来，不能用于其他用途。以常见的计算器为例，人们只能用它进行各类定制好的运算，而无法用它进行文字处理，更不能打游戏。要使这类机器增加新的功能，只能更改其结构，甚至重新设计机器。这类计算装置是不可编程的。

冯·诺依曼体系结构的核心思想——存储程序——改变了这一切。通过创造一组指令集，并将各类运算转换为一组指令序列，使得不需改变机器结构就能使其具备各种功能。在冯·诺依曼体系结构中，程序和数据都是以二进制形式存放在计算机存储器中，程序在控制单元的控制下顺序执行。程序是计算机指令的一个序列，指令是计算机执行的最小单位，由操作码和操作数两部分构成。操作码表示指令要执行的动作，操作数表示指令操作的对象是什么，即数据。

在该体系结构中，计算机由 5 部分组成：存储器、运算器、控制器、输入和输出设备，如图 1.2 所示。需要执行的程序及其要处理的数据保存于存储器中，控制器根据程序指令发出各种命令，控制运算器对数据进行操作、控制输入设备读入数据以及控制输出设备输出数据。

在冯·诺依曼体系结构形成之前，人们将数据存储于主存中，而程序被看成是控制器的一部分，两者是区别对待和处理的。将程序与数据以同样的形式存储于主存中的特点，对于

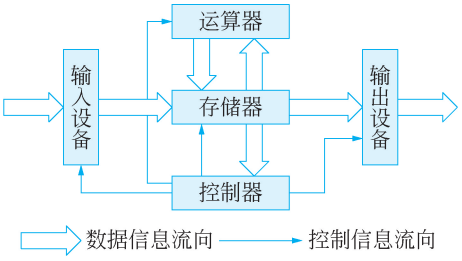

图 1.2　冯·诺依曼体系结构

计算机的自动化和通用性,起到了至关重要的作用。

冯·诺依曼体系结构指的是单机体系结构。为了提高计算机的性能,科学家提出了各种体系结构。例如,由多个计算机构成的并行处理结构、集群结构等。它们的出现是为了满足特定任务的要求,这些任务要求计算机系统有更高的能力,以满足诸如气象预报、核武器数值模拟、航天器设计等任务的需求。目前,主流的并行计算结构有对称多处理系统(Symmetric Multi Processing,SMP)、大规模并行处理系统(Massively Parallel Processing,MPP)和集群等。

除了看得见摸得着的硬件之外,计算机系统中还包含各种计算机软件系统,简称为软件。计算机科学对软件的定义是,“软件是在计算机系统支持下,能够完成特定功能和性能的程序、数据和相关的文档”。于是,软件可形式化地表示为

软件=知识+程序+数据+文档

软件系统是用户与硬件之间的接口,着重解决如何管理和使用计算机的问题。用户主要是通过软件系统与计算机进行交流。软件是计算机系统设计的重要依据。为了方便用户,以及使计算机系统具有较高的总体效用,在设计计算机系统时,必须通盘考虑软件与硬件的结合,以及用户的要求和软件的要求。没有任何软件支持的计算机称为裸机,其本身不能完成任何功能,只有配备一定的软件才能发挥功效。

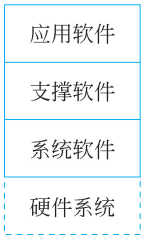

图 1.3　计算机软件系统的结构

软件的分类原则、方法很多。从软件的功能上分为系统软件和应用软件,从实时性上分为实时软件和非实时软件,从软件运行环境上分为单机软件和网络软件,从加工的数据类型上分为事务处理软件、科学和工程计算软件,从计算方法上分为基于传统算法的软件、基于符号演算和推理规则的人工智能软件,等等。

计算机软件系统的结构如图 1.3 所示。这是典型的分层结构,下层系统向上层系统提供服务,上层系统利用下层系统提供的服务,以及特定的程序,可以完成指定的任务。使用计算机并不会直接操作计算机硬件,而是通过在操作系统和各种应用软件上的操作来控制计算机完成各种任务。

1.1.3　硬件系统的关键部件

硬件系统的关键部件在冯·诺依曼体系结构中扮演主要角色,如图 1.4 所示。图中,冯·诺依曼体系结构中的控制器和运算器被集成于 CPU 中,分别对应控制器和算术逻辑单元;主存对应存储器,各种输入输出设备分别对应体系结构中的输入设备和输出设备;各

种总线(图 1.4 中以空心箭头表示)对应于冯·诺依曼体系结构图中的互连线,用于传输命令和数据。

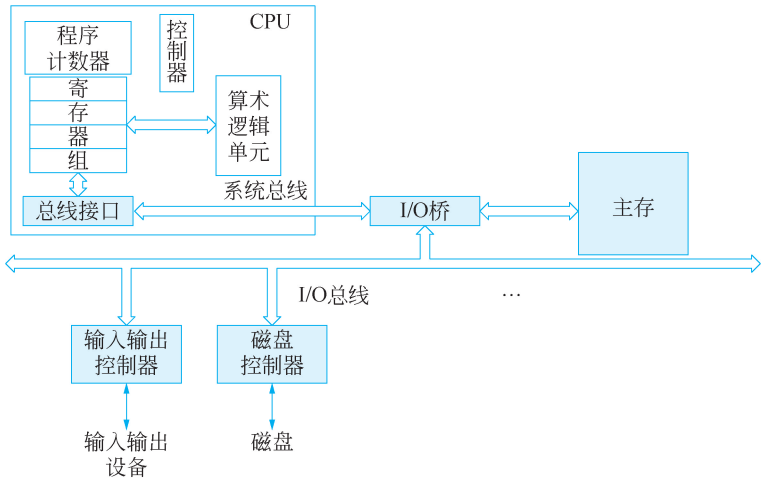

图 1.4 典型的计算机硬件组织结构

1. 中央处理器

一般把中央处理器简称为处理器,它是执行存储在主存中的指令的引擎。CPU 一般由算术逻辑运算器(Arithmetic and Logic Unit,ALU)、控制单元(Control Unit,CU)和寄存器组成,由 CPU 内部总线将这些构成连接为有机整体,如图 1.5 所示。

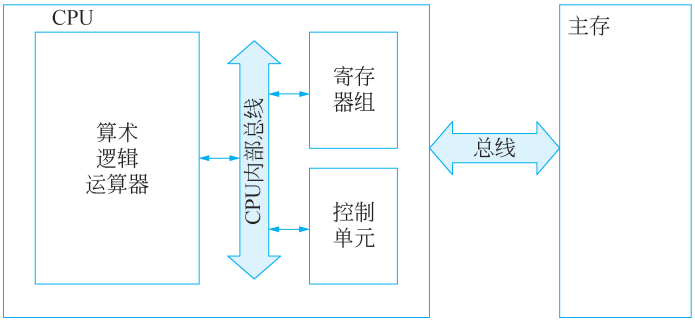

图 1.5 CPU 的内部结构

【例 1.1】 一个程序示例。为了便于阅读,采用汇编指令编写,分号后面是程序的注释,帮助人们阅读和理解程序,而计算机将忽略这些注释。

```
1  ;1_1.asm,汇编程序示例
2  mov #0, RO        ;将寄存器 RO 置为 0
3  mov #1, R1        ;将寄存器 R1 置为 1
4  loop: add R1, RO ;将 R1 与 RO 相加,结果保存到 RO
5  add #1,R1         ;R1 加 1
6  cmp R1,#1000      ;比较 R1 与 1000 的大小
7  ble loop          ;如果 R1 小于或等于 1000,从 loop 那条指令开始执行
8  halt              ;程序结束
```

这段程序用于计算 1+2+…+1000 的值。利用寄存器 R1 和 R0,开始时将 RO 设为 0,R1 设为 1。然后将 R1 的值加到 R0 上,同时 R1 增 1。此后将 R1 的值与 1000 进行比较,

如果 R1 比 1000 小,则重复执行将 R1 加到 R0 上以及 R1 增 1 的操作,然后再比较。这种重复执行将在 R1 大于 1000 时结束,同时将结束程序的运行。程序中,add 是操作指令,ble 是控制指令。ble 与 cmp 一起使用,当 cmp 比较结果为小于或等于 1000 时,该指令被执行,将执行顺序跳转到 loop 所标示的指令。

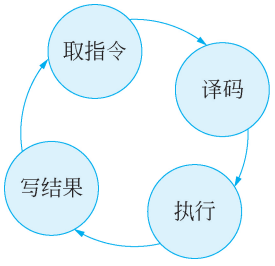

图 1.6　指令执行常见节拍划分

CPU 的工作过程是循环执行指令的过程。指令的执行过程是在控制单元的控制下,精确地、一步一步地完成的。这个执行的步骤顺序称为指令周期,其中的每一步称为一个节拍。不同的 CPU 可能执行指令的节拍数不同,但是通常都可归为以下 4 个阶段(见图 1.6)。

(1) 取指令。指令通常存储在主存中,CPU 通过程序记数器获得要执行的指令存储地址。根据这个地址,CPU 将指令从主存中读入,并保存在指令寄存器中。

(2) 译码。由指令译码器对指令进行解码,分析出指令的操作码,所需的操作数存放的位置。

(3) 执行。将译码后的操作码分解成一组相关的控制信号序列,以完成指令动作,包括从寄存器读数据、输入 ALU 进行算术或逻辑运算。

(4) 写结果。将指令执行节拍产生的结果写回到寄存器,如果有必要,将产生的条件反馈给控制单元。

以上的节拍划分是粗粒度的,通常每个节拍所包含的动作很难在一个时钟周期内完成,因此,会进一步将每个节拍进行细化,细化后的每个动作可在一个时钟周期内完成,不可再细分。

2. 存储系统

计算机系统中的存储器一般分为主存(又称为内存)和辅存(又称为外存),主存可与 CPU 直接进行信息交换,其特点是运行速度快,容量相对较小,在系统断电后,其保存的内容会丢失。辅存属于外部设备的范畴,如硬盘、光盘等,它们通过各种专门接口与计算机通信。辅存与 CPU 之间不能直接交换数据,其特点是存储容量大,存取速度比主存慢,系统断电后其保存的信息不会丢失,存储的信息很稳定。

主存中存储电路的原理类似于电容,主存中通过对电路进行充电来存储信息,但是这很容易流失,因此需要在很短的时间内不断地充电,称为刷新。采用这种技术的主存又称为动态存储器(Dynamic RAM)。

根据存储能力与电源的关系可将主存分为易失性存储器和非易失性存储器,计算机系统主存一般都包含这两类存储器。前者指的是当电源供应中断后,存储器所存储的数据便会消失的存储器,如 RAM、DRAM 等。后者指即使电源供应中断,存储器所存储的数据并不会消失,重新供电后,就能够读取其中数据的存储器,如只读存储器(Read-Only Memory,ROM)等,ROM 也可随机访问。在现代计算机系统的主存中,一般都包含这两种存储器。

一直以来,CPU 的速度总比主存访问速度快。虽然随着工艺水平的提高,主存的访问速度也在不断提高,但是仍然赶不上 CPU 速度的提高。CPU 发出访问主存请求后,往往要等多个时钟周期后才能得到主存内容。存储器越慢,CPU 等待的时间就越长。目前,技术上的解决办法是利用更小更快的存储设备与大容量低速的主存组合使用,以适中的价格得

到速度和高速存储器差别不大的大容量存储器。这种更小更快的存储设备被称为高速缓存存储器(Cache),简称为高速缓存。高速缓存逻辑上介于 CPU 和主存之间,可以将其集成到 CPU 内部,也可置于 CPU 之外。图 1.7 给出了一种典型的高速缓存配置方式。

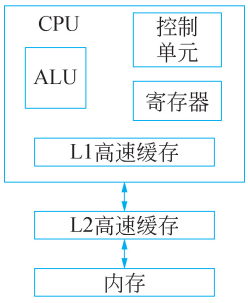

图 1.7 典型的高速缓存配置

对现代高性能 CPU 而言,高速缓存的地位越来越重要。目前在进行高速缓存的设计时,需要考虑高速缓存的容量、高速缓存块容量、如何组织高速缓存、指令与数据是否分开缓存,以及高速缓存个数等问题。一般来说,高速缓存的容量越大,性能越好,但是成本也越高。对于高速缓存个数的问题,常见的是两个,如图 1.7 所示。

不论是主存还是辅存,其访问速度与 CPU 的运行速度都有很大的差距,通常是数量级上的差别(如微秒和纳秒的差别)。为了让慢速设备配合 CPU 的快速运行,通常的做法是在 CPU 和主存之间插入一个更小、更快的存储设备(如 Cache)。实际上,计算系统中的存储器都被组织成存储层次结构(见图 1.8),称为存储系统。最上层是 CPU 中的寄存器,其存储速度能满足 CPU 的要求。下一层是高速缓存,一般容量是 32KB 到几兆字节。再往下是主存,然后是辅存,用于保存永久存放的数据。最后是用于后备存储的磁带和光盘存储器。

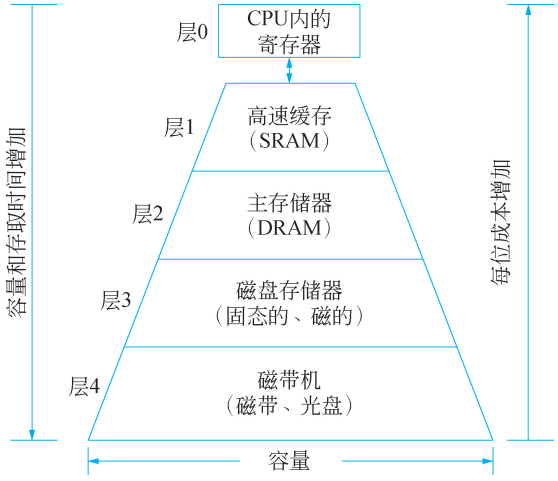

图 1.8 存储系统的层次结构示例

在该层次中,自上而下,读写时间、存储容量和位价比逐渐增大。首先,CPU 中的寄存器的访问时间是纳秒级的,如几纳秒,高速缓存的访问时间是寄存器访问时间的几倍,主存的访问时间是几十纳秒,而磁盘的访问时间为 10ms 以上。其次,CPU 中寄存器一般为 128B 或更多一点,高速缓存可以是几兆字节,主存是几十～数千兆字节,而磁盘的容量是几吉字节(GB)或几十吉字节(GB)。再次,主存的价格一般为几美元/兆字节,而磁盘的价格则为几美分/兆字节。

3. 输入输出系统

输入输出设备是计算机与外界的联系通道,如用于用户输入的鼠标和键盘,用于输出的显示器,以及用于长期存储数据和程序的磁盘。每个输入输出设备通过一个控制器或适配

器与输入输出总线连接。控制并实现信息输入输出的就是输入输出系统(Input/Output System,I/O 系统)。

1.1.4 操作系统

人们在使用现代计算机系统时,通常通过运行各种应用程序来完成各种任务。一般来说,当人们通过双击鼠标或输入一个命令来运行程序时,计算机系统要为程序分配主存空间,并将程序从磁盘中读入为其分配的主存空间中。然后,程序第一条指令在主存中的地址将被赋值给 CPU 的指令记数器,在控制单元的控制下,从这个地址读指令到 CPU,开始程序的执行。如果程序运行过程中需要进行输入输出操作,则还将利用某种输入输出方式与外设交换数据。

在没有操作系统之前,这些运行程序涉及的大部分操作都要程序员自己来开发并实现,使得程序员花费大量的时间在与业务无关的编程上,极大地降低了效率。因此,人们考虑将每个程序运行都涉及的、对计算机系统资源的操作独立出来,由专门的程序对其进行管理,而程序员专心于与应用直接相关的编程工作,从而出现了操作系统。

在现代计算机系统中,操作系统是计算机系统中最基本的系统软件,是整个计算机系统的控制中心。操作系统通过管理计算机系统的软硬件资源,为用户提供使用计算机系统的良好环境,并且采用合理有效的方法组织多个用户共享各种计算机系统资源,最大限度地提高系统资源的利用率。

按操作系统在计算机系统中发挥的作用,可认为它具有资源管理者和用户接口两重角色。

1) 资源管理者

计算机系统的资源包括硬件资源和软件资源。从管理角度看,计算机系统资源可分为四大类:处理机、存储器、输入输出设备和信息(通常是文件)。操作系统的目标是使整个计算机系统的资源得到充分有效的利用,为达到该目标,一般通过在相互竞争的程序之间合理有序地控制系统资源的分配,从而实现对计算机系统工作流程的控制。作为资源管理者,操作系统的主要工作是跟踪资源状态、分配资源、回收资源和保护资源。由此,可以把操作系统看成是由一组资源管理器(处理机管理、存储器管理、输入输出设备管理和文件管理)组成的。

2) 用户接口

在计算机系统组成的 4 个层次中,硬件处于最底层。对多数计算机而言,在机器语言级上编程是相当困难的,尤其是对输入输出操作编程。需要一种抽象机制让用户在使用计算机时不涉及硬件细节。操作系统正是这样一种抽象,用户使用计算机时,都是通过操作系统进行的,不必了解计算机硬件工作的细节。通过操作系统来使用计算机,操作系统就成为用户和计算机之间的接口。

1.2 计算机中的信息表示

1.2.1 信息编码

所谓信息编码,就是为了方便信息的存储、检索和使用,采用少量基本符号(数码)和一

定的组合原则来区别和表示信息。基本符号的种类和组合原则是信息编码的两大要素。现实生活中的编码例子并不少见，如用字母的组合表示汉语拼音；用 0～9 这 10 个数码的组合表示数值等。0～9 这 10 个数码又称为十进制码。

信息编码的目的在于为计算机中的数据与实际处理的信息之间建立联系，提高信息处理的效率。在计算机中，信息编码的基本元素是 0 和 1 两个数码，称为二进制码。计算机采用二进制码 0 和 1 的组合来表示所有的信息称为二进制编码。计算机存储器中存储的都是由 0 和 1 组成的信息编码，它们分别代表各自不同的含义，有时表示一个数值，有时表示英文字符或标点符号，有时表示图像，有时又表示声音，等等，它们都分别采用各自不同的编码方案。

计算机处理信息之前，必须将文字、数值、图像、声音等信息转换为 0 和 1 构成的符号串，才能在计算机中存储和使用。将声、光、电、磁等信号及语言、图像、报文等信息转变成为符号串编码后进行处理、存储、传递，称为信息的数字化。

虽然计算机内部均采用二进制编码来表示各种信息，但计算机与外部交往仍采用人们熟悉和便于阅读的形式，如十进制数据、中英文文字显示及图形描述。其间的转换则由计算机系统内部来实现。

1.2.2 数据的进制及转换

在日常生活中，常用不同的规则来记录不同的数，如 1 年有 12 个月，1 小时为 60 分钟，1 分钟为 60 秒；1 米等于 10 分米，1 分米等于 10 厘米等。按进位的方法，表示一个数的记数方法称为进位记数制，又称数制。在进位记数制中，最常见的是十进制，此外还有十二进制、十六进制等。当将数值输入计算机中时必须将十进制转换为二进制，而将计算机中的数值输出时，一般要将其转换为十进制，以便于人阅读和理解。

1. 十进制

十进制记数方法为“逢十进一”，一个十进制数的每一位都只有 10 种状态，分别用 0～9 等 10 个数符(数码)表示，任何一个十进制数都可以表示为数符与 10 的幂次乘积之和。如十进制数 5296.45 可写成：

$$5296.45=5\times10^{3}+2\times10^{2}+9\times10^{1}+6\times10^{0}+4\times10^{-1}+5\times10^{-2}$$

上式称为数值按位权多项式展开，其中 10 的各次幂称为十进制数的位权，10 称为基数。

2. 二进制

基数为 2 的记数制称为二进制，二进制是“逢二进一”，每一位只有 0 和 1 两种状态，位权为 2 的各次幂。任何一个二进制数，同样可以用多项式之和来表示，如：

$$1011.01=1\times2^{3}+0\times2^{2}+1\times2^{1}+1\times2^{0}+0\times2^{-1}+1\times2^{-2}$$

二进制数整数部分的位权从最低位开始依次是 2^{0}、2^{1}、2^{2}、2^{3}、2^{4}、……小数部分的位权从最高位依次是 2^{-1}、2^{-2}、2^{-3}、2^{-4}、……其位权与十进制数值的对应关系如表 1.1 所示。

表 1.1 二进制小数部分的位权与十进制数值的对应关系

位　权	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}
数　值	16	8	4	2	1	1/2	1/4	1/8

3. 十进制到二进制的转换

1) 十进制整数转换成二进制整数

十进制整数转换成二进制整数通常采用“除 2 取余,逆序读数”。就是将已知十进制数反复除以 2,每次相除后若余数为 1,则对应二进制数的相应位为 1;若余数为 0,则相应位为 0。首次除法得到的余数是二进制数的最低位,后面的余数为高位。从低位到高位逐次进行,直到商为 0。

【例 1.2】 求$(13)_{10}=(?)_2$。

解:该数为整数,用“除 2 取余法”,即将该整数反复用 2 除,直到商为 0;再将余数依次排列,先得出的余数在低位,后得出的余数在高位。

由此可得$(13)_{10}=(1101)_2$。

```
2 | 13      余1   最低位
  2 | 6     余0    ↑
    2 | 3   余1    ↑
      2 | 1 余1   最高位
        0
```

2) 十进制纯小数转换成二进制纯小数

十进制纯小数转换成二进制纯小数采用“乘 2 取整,顺序读取”。就是将已知十进制纯小数反复乘以 2,每次乘 2 后所得新数的整数部分若为 1,则二进制纯小数相应位为 1;若整数部分为 0,则相应位为 0。从高位向低位逐次进行,直到满足精度要求或乘 2 后的小数部分是 0 为止。

【例 1.3】 求$(0.3125)_{10}=(?)_2$。

解:

```
0.3125×2=0.6250      取整 0    最高位
0.6250×2=1.25        取整 1      ↓
0.25×2=0.5           取整 0      ↓
0.5×2=1.0            取整 1    最低位
```

由此可得$(0.3125)_{10}=(0.0101)_2$。

多次乘 2 的过程可能是有限的也可能是无限的。当乘 2 后的数其小数部分等于 0 时,转换即告结束。当乘 2 后小数部分总不为 0 时,转换过程将是无限的,这时应根据精度要求取近似值。若未提出精度要求,则一般小数位数取 6 位;若提出精度要求,则按照精度要求取相应位数。因此,小数转换有时不能实现精确匹配。

3) 十进制混合小数转换成二进制数

混合小数由整数和小数两部分组成。只需要将其整数部分和小数部分分别进行转换,然后再用小数点连接起来即可得到所要求的混合二进制数。

【例 1.4】 求$(13.3125)_{10}=(?)_2$。

解:只要将前面两例的结果用小数点连接起来即可。

由此可得$(13.3125)_{10}=(1101.0101)_2$。

1.2.3 计算机二进制的优势体现

与十进制码相比,二进制码并不符合人们的习惯,但是计算机内部仍采用二进制编码表

示信息，其主要原因有以下 4 点。

1. 容易实现

二进制数中只有 0 和 1 两个数码，易于用两种对立的物理状态表示。如用开关的闭合或断开两种状态分别表示 1 和 0；用电脉冲有或无两种状态分别表示 1 和 0。一切有两种对立稳定状态的器件（即双稳态器件），均可以表示二进制的 0 和 1。而十进制数有 10 个数码，则需要一个 10 稳态的器件，显然设计前一类器件要容易得多。

2. 可靠性高

计算机中实现双稳态器件的电路简单，而且两种状态所代表的两个数码在数字传输和处理中不容易出错，因而电路可靠性高。

3. 运算简单

在二进制中算术运算特别简单，加法和乘法仅各有得 3 条运算规则，如下。

加法：0＋0＝0，0＋1＝1，1＋1＝10。

乘法：0×0＝0，0×1＝1×0＝0，1×1＝1。

因此可以大大简化计算机中运算电路的设计。相对而言，十进制的运算规则复杂很多。

4. 易于逻辑运算

计算机的工作离不开逻辑运算，二进制数码的 1 和 0 正好可与逻辑命题的两个值“真”（True）与“假”（False），或“是”（Yes）与“否”（No）相对应，这样就为计算机进行逻辑运算和在程序中的逻辑判断提供了方便，使逻辑代数成为计算机电路设计的数学基础。

1.2.4 计算机数值的编码表示

在计算机中，数值型的数据有两种表示方法，一种叫作定点数，另一种叫作浮点数。所谓定点数，就是在计算机中所有数的小数点位置固定不变。定点数有两种：定点小数和定点整数。定点小数将小数点固定在最高数据位的最左边，因此，它只能表示小于 1 的纯小数。定点整数将小数点固定在最低数据位的最右边，因此定点整数表示的也只是纯整数。由此可见，定点数表示数的范围较小。

为了扩大计算机中数值数据的表示范围，将 12.34 表示为 0.1234×10^{2}，其中 0.1234 叫作尾数，10 叫作基数，可以在计算机内固定下来。2 叫作阶码，若阶码的大小发生变化，则意味着实际数据小数点的移动，因而把这种数据叫作浮点数。由于基数在计算机中固定不变，因此，可以用两个定点数分别表示尾数和阶码，从而表示这个浮点数。其中，尾数用定点小数表示，阶码用定点整数表示。

在计算机中，无论是定点数还是浮点数，都有正负之分。在表示数据时，专门有 1 位表示符号位。对单符号位来讲，通常用“1”表示负号，用“0”表示正号，而且符号位都处于数据的最高位。

1. 定点数的表示

一个定点数，在计算机中可用不同的码制来表示，常用的码制有原码、反码和补码三种。不论用什么码制来表示，数据本身的值并不发生变化，数据本身所代表的值叫作真值。

原码的表示方法为：如果真值是正数，则最高位为 0，其他位保持不变；如果真值是负数，则最高位为 1，其他位保持不变。

【例 1.5】 写出 13 和－13 的原码(取 8 位码长)。

解:13＝$(1101)_2$,故

13 用 8 位二进制表示的原码是 00001101

－13 用 8 位二进制表示的原码是 10001101

提示:采用原码,优点是转换非常简单,只要根据正负号将最高位置 0 或 1 即可。但原码表示在进行加减运算时很不方便,符号位不能参与运算,并且 0 的原码有两种表示方法:+0 的原码是 00000000,－0 的原码是 10000000。

反码的表示方法为:如果真值是正数,则最高位为 0,其他位保持不变;如果真值是负数,则最高位为 1,其他位按位求反。

【例 1.6】 写出 13 和－13 的反码(取 8 位码长)。

解:13＝$(1101)_2$,故

13 用 8 位二进制表示的反码是 00001101

－13 用 8 位二进制表示的反码是 11110010

提示:反码与原码相比较,符号位虽然可以作为数值参与运算,但计算完后,仍需要根据符号位进行调整。另外 0 的反码同样也有两种表示方法:+0 的反码是 00000000,－0 的反码是 11111111。

为了克服原码和反码的上述缺点,人们又引进了补码表示法。补码的作用在于能把减法运算化成加法运算,现代计算机中一般采用补码来表示定点数。

补码的表示方法为:如果真值是正数,则最高位为 0,其他位保持不变;如果真值是负数,则最高位为 1,其他位按位求反后再加 1。

【例 1.7】 写出 13 和－13 的补码(取 8 位码长)。

解:13＝$(1101)_2$,故

13 用 8 位二进制表示的补码是 00001101

－13 用 8 位二进制表示的补码是 11110011

提示:补码的符号可以作为数值参与运算,且计算完后,不需要根据符号位进行调整。另外,0 的补码表示方法也是唯一的,即 00000000。

2. 浮点数的表示方法

浮点数表示法类似于科学记数法,任一数均可通过改变其指数部分,使小数点发生移动,如数 23.45 可以表示为:$10^1\times2.345$、$10^2\times0.2345$、$10^3\times0.02345$ 等各种不同形式。浮点数的一般表示形式为:$N=2^E\times D$,其中,D 称为尾数,E 称为阶码。图 1.9 为浮点数的一般形式。精度由尾数确定,大小由阶码确定。

阶码符号位	阶码 E	尾数符号位	尾数 D

图 1.9 浮点数的一般形式

对于不同的机器,阶码和尾数各占多少位,分别用什么码制表示都有具体规定。在实际应用中,浮点数的表示首先要进行规格化,即转换成一个纯小数与 2^E 之积,并且小数点后的第一位是 1。

【例 1.8】 写出浮点数$(-101.11101)_2$的机内表示(阶码用 4 位原码表示,尾数用 8 位补码表示,阶码在尾数之前)。

解：$(-101.11101)_2=(-0.10111101)_2\times 2^3$

阶码为 3，用原码表示为 0011

尾数为 -0.10111101，用补码表示为 1.01000011

因此，该数在计算机内表示为 00111.01000011。

1.2.5 字符的编码表示

字符信息是最基本的信息类型之一。一个字符是指独立存在的一个符号，例如汉字、大小写形式的英文字母、日文的假名、数字和标点符号等。还有一类控制字符，用于通信、人机交互等方面，起控制作用，如“回车符”“换行符”等。

在人类文明发展的过程中，发明了各种各样的符号体系，用来表征事物，交流思想。其中典型的符号体系是人类所使用的语言。在一个符号体系中，存在一组基本符号，它们可构成更大的语言单位。这组基本符号的数目一般比较小。例如英语的字母，用之可构成英文的单词。英文单词有成千上万个，但英文字母加起来仅有 52 个(区分大小写)。例外的情况是汉语，汉语中可由汉字构成有意义的词或词组，但汉字数目比较大。

计算机内部用二进制对字符对象进行编码。对于任意一个字符对象集合，不同的人都可设计自己的编码体系。但是为了减少编码体系之间转换的复杂性，提高处理效率，相关组织发布了标准编码方案，以便信息交换和共享。

1. ASCII 码

ASCII 码中所含字符个数不超过 128(见图 1.10)，其中包含控制符、通信专用字符、十进制数字符号、大小写英文字母、运算符和标点符号等。打印出来时，控制字符和通信专用字符是不可见的，不占介质空间，它们指明某种处理动作。其他字符是可见的，所以称为可视字符。

	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	′	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	-	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	.	7	G	W	g	w
1000	BS	CAN	)	8	H	X	h	x
1001	HT	EM	(	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	EAC	+	;	K	[	k	{
1100	FF	ES	′	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	0	DEL

图 1.10 ASCII 码编码表

一个 ASCII 码由 8 位二进制(1B)组成，实际使用低 7 位，最高位恒为 0。所以，ASCII

码中的字符个数不能超过128个。8位二进制能够编码256个符号,有一半编码空置,这主要是为以后的应用留下扩展空间,或最高位留作他用。例如,大写字母A的编码为01000001,十进制为65;数字符号0的编码为00110000,十进制为48。从这里可以看出,数字符号的ASCII码与它所代表的数值是完全不同的两个概念。分析ASCII码表,可看出其中常见编码的大小规则,即0～9<A～Z<a～z。数字符0的编码比数字符9的编码小,并按0～9的顺序递增,如'5'<'8'数字符编码小于英文字母编码,如'9'<'A';字母A的编码比字母Z的编码小,并按A～Z顺序递增。如'A'<'Z';同一个英文字母,其大写形式的编码比小写形式的编码小32,如'a'-'A'=32。

2. 汉字编码

汉字编码适用于汉字信息的交换、传输、存储和处理。中国大陆、新加坡等地广泛采用的汉字编码标准是GB 2312—80,它由中国国家标准局发布,于1981年5月1日开始实施。其全称是"信息交换用汉字编码字符集——基本集",GB 2312是标准文件的代码,其中GB是"国标"这两个汉字拼音的首字母,2312是标准序号。GB 2312收录汉字6763个。另外,还收录了包括汉字拼音符与注音符、拉丁字母、希腊字母、日文平假名和片假名、俄语西里尔字母、运算符、数字符号、标点符号和序号等682个全角字符。

在计算机内部,GB 2312编码占两个字节,第一个字节保存GB 2312编码中对应区码的部分,第二个字节保存GB 2312编码中对应位码的部分。为了与ASCII码区分开来,约定每个字节的最高位恒为1。在计算机内部的这种编码形式,称为汉字的机内码。例如,"计算机"这3个汉字的区位码分别为2838、4367和2790,GB 2312编码分别为6070、7599和59122(低3位对应位码部分)。它们的机内码分别为

1011110011000110　1100101111100011　1011101111111010

其机内码的十六进制表示分别为BCC6、CBE3和BBFA。

3. Unicode编码

Unicode码又称为统一码、万国码或单一码,1994年开始研发,1994年公布第一个版本,并不断在完善和改进中。2006年发布了最新版本Unicode 5.0.0。Unicode是基于通用字符集(Universal Character Set,UCS)的标准开发。Unicode给世界上每种语言的文字、标点符号、图形符号和数字等字符都赋予一个统一且唯一的二进制编码,以满足跨语言、跨平台进行文本转换、处理的要求。随着计算机应用的广泛发展,Unicode码逐步得到普及。

计算机使用Unicode编码时,要将其转换成相关类型的数据。数据类型不同,转换方法也不同。计算机网络中有很多转换程序可供下载,在此不再详述转换方法。

1.3 算法

1.3.1 算法的概念

程序设计人员的基本技能之一是设计算法,并根据算法写出程序。因此,对于初学者来说,掌握一些常用算法是非常重要的。许多初学者常常把要解决的问题首先和程序设计语言中的语句联系在一起,影响了程序设计质量。设计算法和编写程序要分开考虑,在学习程

序设计语言之前，就应该学会针对一些简单问题来设计算法。

算法就是解决问题的具体步骤或方法，一个有限的、确定的、有效的步骤序列，它告诉计算机(或人)——“怎么做”，输入输出的每一步该干什么。例如，要计算 1～100 的自然数的和，可以有不同算法。

算法 A(循环累加)：从 1 加到 100，逐步相加。

算法 B(公式法)：使用高斯公式 $n(n+1)/2$，一步得到结果。

两者都是算法，只是解决同一个问题的效率不同。

算法具备以下特性：

有输入：问题的初始数据。

有输出：问题的结果或答案。

有穷性：执行步骤必须在有限时间内结束。

确定性：每一步的操作是明确的、无歧义的。

可行性：每一步都能在有限时间内完成。

算法与程序是有区别的，算法关注思想与逻辑，即怎么做才最优，程序关注如何用代码实现，怎么让机器执行。因此，算法是灵魂，程序是载体。

1.3.2 算法的表示

人们可以用不同的方法来描述一个算法。常用的有自然语言、流程图、伪代码和计算机语言等表示法。

1. 自然语言表示法

自然语言就是人们日常使用的语言，可以是汉语、英语或其他语言。

【例 1.9】 求 $m!$。

分析：如果 $m=6$，即要求 $1\times2\times3\times4\times5\times6$。先设 s 表示累乘积，t 表示乘数。自然语言表示 $m!$ 的算法描述如下：

第 1 步，使 $s=1$，$t=1$；

第 2 步，使 $s\leftarrow s\times t$；

第 3 步，使 $t\leftarrow t+1$；

第 4 步，如果 $t\leqslant m$，返回第 2 步重复执行，否则输出 s 后结束。

用自然语言描述算法具有通俗易懂的优点，但它的缺点如下。

(1) 冗长。自然语言表示算法往往要用一段冗长的文字才能说清楚所要进行的操作。

(2) 容易出现歧义。自然语言往往要根据上下文才能正确判断出其含义，不太严谨。如“张三对李四说他的儿子考上了大学”就存在“歧义性”，因为究竟指谁的儿子不明确。

(3) 用自然语言表示顺序执行的步骤比较好懂，但如果算法中包含判断或转移时就不够直观。

因此，除了那些很简单的问题之外，一般不用自然语言表示算法。

【例 1.10】 有两个瓶子 A 和 B，A 中盛放酒，B 中盛放醋。现要求将其中的酒和醋互换，即 A 中盛放醋，B 中盛放酒。设计一个交换两个瓶子中的酒和醋的算法。

分析：这是一个非数值运算问题。一般地，两个瓶子中的液体不能直接交换。要解决这一问题，最好的办法是引入第 3 个空的瓶子 C。这样，交换步骤可描述为如下算法：

第 1 步,将 A 瓶中的酒装入 C 瓶中;

第 2 步,将 B 瓶中的醋装入 A 瓶中;

将 C 瓶中的酒装入 B 瓶中。

思考:可将 3 个瓶子看成是 3 个变量,瓶子中的酒和醋看作变量中存放的值,这个算法就可以引申到交换两个变量的值。

2. 流程图表示法

用一些几何图形代表各种不同性质的操作,这种表示方式称为算法的流程图表示法。美国国家标准化协会规定了一些常用的图形符号(如图 1.11 所示),这些符号已被世界各国的程序员普遍接受和采用。

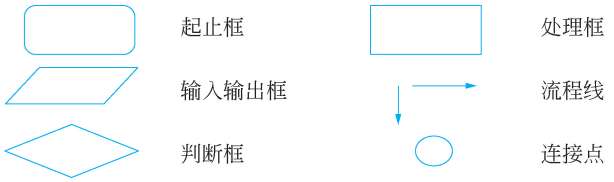

图 1.11　常用图形符号

(1) 起止框。表示算法的开始和结束。一般内部只写“开始”或“结束”。

(2) 输入输出框。表示算法请求输入输出需要的数据或算法将某些结果输出。一般内部常常填写“输入……”,“打印/显示……”。

(3) 判断框(菱形框)。主要是对一个给定条件进行判断,根据给定的条件是否成立来决定如何执行其后的操作。

(4) 处理框。表示算法的处理步骤,内部常常填写赋值操作。

(5) 流程线。用于指示程序的执行方向。

(6) 连接点。用于将画在不同地方的流程线连接起来,同一个编号的点是相互连接在一起的。使用连接点,还可以避免流程线的交叉或过长,使流程图更加清晰。

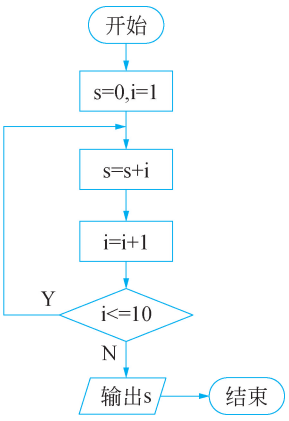

图 1.12　算法流程图

【例 1.11】　用流程图表示求 $\sum_{i=1}^{10} i$ 的算法。

分析:令 s 表示累加和,初值为 0。用流程图描述的算法如图 1.12 所示。

流程图比较直观、灵活,并且较易掌握,但是这种流程图对于流程线的走向没有任何限制,可以任意转向,在描述复杂的算法时所占的篇幅较多,费时费力且不易阅读。

3. 伪代码表示法

使用流程图表示算法,清晰易懂,但如果要修改,工作量会很大。在设计算法的过程中,通常需要反复修改,不断完善。为了在设计算法时方便,经常使用伪代码作为描述工具。伪代码是一种近似于高级语言又不受高级语言语法约束的一种算法描述方式,这在英语国家中使用起来更加方便。

【例 1.12】 用伪代码表示求 $\sum_{i=1}^{10} i$ 的算法。

```
sum ← 0
FOR i ← 1 TO 10 DO
    sum ← sum + i
ENDFOR
```

伪代码书写格式比较自由，容易修改，但用伪代码表示的算法没有用流程图表示的算法直观。

4. 计算机语言表示法

计算机是无法识别流程图和伪代码的，只有用计算机语言编写的程序才能被计算机翻译后执行。例如，已经给出例 1.9 中求 $m!$ 的算法，但还没有求出确切的结果。只有实现了算法，才能得到运算结果。因此，描述一个算法后，还需将它转换成相应的计算机语言程序。用计算机语言表示算法时，必须严格遵循所用语言的语法规则，这是和伪代码不同的。C++ 语言的语法规则和程序设计方法等将在以后的章节展开讨论。

1.3.3 算法优劣的评价标准

评价一个算法的优劣主要着眼于两大核心方面：正确性（功能）与效率（性能）。

正确性是算法的基石，它确保算法能够可靠地完成其预定任务。正确性是指：算法是否对所有合法输入都能产生正确输出？算法是否能在有限时间内停止？算法能否处理异常或边界情况？例如，一个排序算法若在遇到重复元素时出错，那这个算法就不正确。

效率则衡量算法执行任务的成本效益，算法的效率包括时间复杂度、空间复杂度、可读性与可维护性、算法的鲁棒性。

时间复杂度用来衡量算法运行所需的时间随输入规模 n 增长的变化趋势/常用英文斜体大写字母 O 表示法。$O(1)$ 表示常数时间的时间复杂度，即运行所需的时间与输入数据的规模 n 之间没有任何关系，是个时间常量。$O(\log n)$ 是对数级别的时间复杂度，例如，二分查找的时间复杂度即为 $O(\log n)$。$O(n)$ 线性级别，注意这里的线性级别运行所需的时间不一定就是 n，可能是 kn，只要 k 远远小于 n 即可。归并和快速排序的时间复杂度为 $O(n\log n)$，冒泡和选择排序的时间复杂度为 $O(n^2)$。除此之外，还有指数和阶层级别的 $O(2^n)$、$O(n!)$，这两种时间复杂度通常被认为太高，一般不采纳。

空间复杂度用来衡量算法运行时所需的额外内存空间，一般也用英文斜体大写字母 O 表示。如 $O(1)$ 和 $O(n)$，如辅助数组则需要额外的线性空间，空间复杂度为 $O(n)$。

可读性与通用性指的是代码是否容易理解和修改？结构是否清晰，有无注释？这在软件工程中尤为重要。仍然以上述求 1～100 的和的例子为例，虽然看起来算法 B 效率更高，但是如果改成求任意输入的 100 个整数的和，算法 A 很容易修改，算法 B 却无法修改，通用性不好。

鲁棒性是指面向异常输入如空值、边界值时，是否能稳定工作，有没有异常处理。

例如，查找一个数是否在数组中，如果采用线性查找，从头到尾逐个比对，则时间复杂度为 $O(n)$，只需一个变量保存当前索引或比较结果，不用额外空间，因此空间复杂度为 $O(1)$。

而二分查找则需要数据提前排好序,时间复杂度为 $O(\log n)$,空间复杂度为 $O(n)$。哈希查找把数组元素存入哈希表,然后直接查,时间复杂度为 $O(1)$,需要额外 $O(n)$空间来存储哈希表结构,空间复杂度为 $O(n)$,此方法通过哈希表实现了以空间换时间的效果。

算法的终极目标,是用最少的资源和最快的速度,根据输入数据得到正确结果。

1.4 使用程序设计语言解决问题的基本步骤

如何使用程序设计语言解决问题呢?

第一步,问题描述。首先需要对所需解决的问题进行准确的描述,称为问题描述,通过问题描述,明确要解决的问题是什么,包括输入和预期输出。

第二步,数学模型。基于问题描述,建立问题的数学模型。

第三步,计算机模型。将数学模型转换为计算机模型,即设计算法或逻辑流程来解决这个问题,可以使用伪代码或流程图来表示。

第四步,代码实现。根据问题的需求、个人技能以及运行环境要求,选择一门编程语言来实现算法,也就是将设计好的算法转换为代码。本教材采用 C++ 语言,并使用 VC++ 2010 Express 作为集成开放环境。一个良好的程序开发者,应注重保持代码的可读性和可维护性,在编写代码的过程中,适当加上注释,以增加程序的可读性。

第五步,测试。对已经写好的代码进行测试,测试包括代码的正确性和算法的效率。使用各种预期的数据将进行测试,确保程序能处理正常和异常的情况,并测试边界条件和极端情况下程序的表现。

第六步,改进。根据测试结果,对程序中可能存在的问题进行改正,并分析程序的性能,对效率低下的部分进行优化,考虑算法的时间复杂度和空间复杂度。在空间复杂度不是特别高的情况下,优先考虑如何降低算法的时间复杂度。

第七步,部署应用。撰写软件开发和使用文档,并部署应用。该步骤如图 1.13 所示。

1.5 本章小结

本章介绍了计算机系统的基础知识,内容涵盖了计算的定义和计算机系统的基本组成部分,包括硬件和软件系统的构成与功能。我们详细探讨了计算机硬件系统,如中央处理器、存储系统、总线和输入输出系统,并解释了操作系统的主要功能,包括进程管理、存储管理、文件管理和设备管理等。

在信息表示方面,本章详细讲解了各种编码技术、数值表示方法,以及字符、声音和图像的数字化表示方法,帮助我们理解计算机如何处理和存储不同类型的信息。

本章还介绍了算法的概念,以及使用什么样的指标可以评价算法的优劣;并从软件开发角度,详细描述了通过程序设计语言解决问题的基本步骤,让读者从宏观层面对计算机和程序设计语言有一个深入的了解。

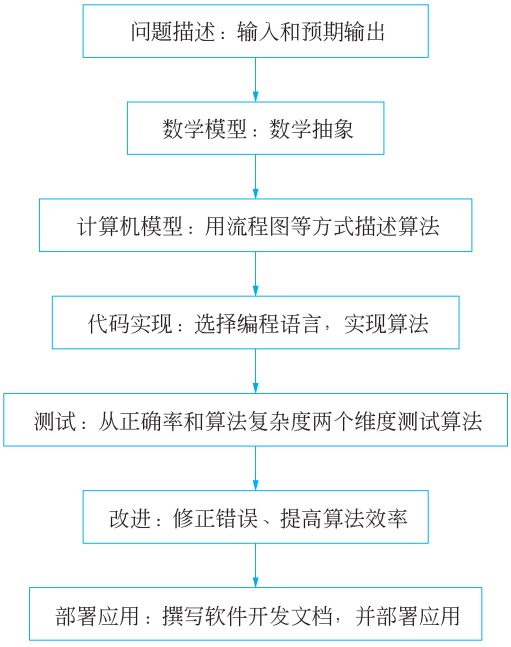

图 1.13　使用程序设计语言解决问题的基本步骤

1.6　习题

1. 计算机的发展经历了哪几个阶段？各个阶段的主要特征是什么？

2. 设字长为 8 位，写出下列各数的补码。125、－125、－1、255、－256、－2345。

3. 将下列十进制数转换为二进制数、八进制数和十六进制数：102、378、126.125、40.25。

4. 将二进制数、八进制数和十六进制数转换为对应的十进制数：(11010111)2、(567)8、(15B)16。

5. 某学生 5 门功课的成绩为 80，95，78，87，65，请写出求平均成绩的算法，并画出流程图。

6. 任意输入 3 个数，将它们按从小到大的顺序排列输出。请写出相应的算法，并画出流程图。

习题 1