第 1 章

程序设计基础

随着科学技术的飞速发展，计算机技术日新月异，计算机程序设计语言也层出不穷，作为程序设计初学者，首先应该了解什么是程序、什么是程序语言、如何进行程序设计等初学者所要面临的基本问题、共性问题。本书作为一门程序设计的教材，将以 C 语言程序设计为主线，详细讲述 C 程序设计的基本概念、语法规则和基本方法。本章首先就程序设计技术的基本知识进行概述，然后在对程序设计语言、程序设计技术等概念介绍的基础上，重点就算法的概念、特征、算法描述方式、软件开发的过程和 C 语言特点以及运行环境等内容进行逐一介绍。

知识点：

- 程序设计语言和程序设计的概念
- 算法的概念、特征以及描述方式
- 软件编制的基本步骤
- C 语言特点及其运行环境
- 认识简单 C 程序

1.1 程序设计与计算思维

对于使用计算机的大多数人来讲，当希望计算机来完成某一项工作时，将面临两种情况：一是可以借助现成的应用软件完成，例如，设计一个网页可以使用 Dreamweaver，写一份报告可以使用 Word，做一个产品介绍可以使用 PowerPoint，处理一幅图片可以使用 Photoshop 等；二是没有完全适合你的应用软件。这时就必须将要解决问题的步骤编写成一条条指令，而且这些指令还必须被计算机间接或直接地接受并能够执行。换句话说，为了使计算机达到预期目的，就要先得到解决问题的步骤，并依据对该步骤的数学描述编写计算机能够接受和执行的指令序列——程序，然后运行程序得到所要的结果，这就是程序设计。

学习程序设计，主要是进一步了解计算机的工作原理和工作过程。例如，知道数据是怎样存储和输入/输出的，知道如何解决含有逻辑判断和循环的复杂问题，知道图形是用什么方法画出来以及怎样画出来的等。这样在使用计算机时，不但知其然而且还知其所以然，能够更好地理解计算机的工作流程和程序的运行状况，为以后维护或修改应用程序以适应新的需要打下良好的基础。

程序设计是计算机应用人员的基本功。一个有一定经验和水平的计算机应用人员不应当和一般的计算机用户一样，只满足于能使用某些现成的软件，而且还应当具有自己开发应用程序

的能力。现成的软件不可能满足一切领域的多方面的需求，即使是现在有满足需要的软件产品，但是随着时间的推移和条件的变化它也会变得不适应。因此，计算机应用人员应当具备能够根据本领域的需要进行必要的程序开发工作的能力。

1．程序

程序就是完成或解决某一问题的方法和步骤。它是为完成某个任务而设计的，由有限步骤所组成的一个有机的序列。它应该包括两方面的内容：做什么和怎么做。计算机是一个由物理元件组成的机器，像飞机、汽车或割草机等机械设备一样，一台计算机只有成功启动并运行相应的控制程序，才能完成操作人员想要做的任务。然而，区分计算机和其他机械类型机器的关键就是它们各自是如何执行任务的。例如，一辆汽车是由坐在车内的司机控制和驾驶的，而对于一台计算机来说，这个驾驶员就是程序。所谓程序就是按某种顺序排列的，使计算机能完成或解决某种任务（例如解题、检索数据或对一个系统进行控制等）的连续执行的一条条指令集合，也就是说，程序是计算机指令的序列，编制程序工作就是为计算机安排指令序列。

计算机程序是为了使计算机完成一个预定的任务而设计的一系列的语句或指令的集合。因此，可以说，“ 程序 ” 是为了解决某一特定问题而用某种计算机程序设计语言编写出的代码序列。如果我们需要计算机完成什么工作，就要将其步骤用多条指令的形式描述出来，并把指令存放于计算机内部存储器中，需要结果时就向计算机发出一个简单的命令，计算机就会自动逐条命令顺序执行，直到全部指令执行完毕并得到预期的结果。然而，这些编写程序的指令是只有计算机才能理解的二进制 0 和 1 编码，这种编码方式编写的程序让人不好掌握和记忆，也不利于程序编写和软件发展。因此，计算机科学家们研制了各种计算机能够识别、而且又接近于人类自然语言的计算机语言，这就是常说的编写软件的程序设计语言。

2．程序设计语言

程序设计语言，通常称为编程语言，是一组用来定义计算机程序的语法规则。一种程序设计语言能够准确地定义计算机所需要使用的数据，并能精确定义在不同情况下所应当采取的操作。程序设计语言按照使用的方式和功能可分为：机器语言、汇编语言等低级语言和面向过程、面向对象的高级语言。

（1）机器语言

机器语言（Machine Language）是直接用二进制编码指令表示的计算机语言，就是机器指令的集合，它与计算机同时诞生，属于第一代计算机语言，其指令是由 0 和 1 组成的一串代码，有一定的位数，并被分成若干段，各段的编码表示不同的含义。机器语言也称为面向机器的语言，用机器语言编写的程序称为机器语言程序或指令程序（机器码程序），其机器本身能直接识别和执行这种目标程序机器码。机器语言对不同型号的计算机来说一般是不同的。

如某种计算机字长为 16 的指令 1011011000000000，它表示让计算机进行一次加法操作；而指令 1011010100000000 则表示进行一次减法操作。早期的程序设计均使用机器语言。程序员们将用 0、1 数字编成的程序代码打在纸带或卡片上，1 打孔，0 不打孔，再将程序通过纸带机或卡片机输入计算机，进行运算。例如应用 8086CPU 完成运算 s=768+12288-1280，机器码如下：

```
10110000000000000000000011
00000101000000000000110000
00101101000000000000000101
```

假如将程序错写成以下这样，请你找出错误，小伙伴们马上头大了吧。

```
10110000000000000000011
00000101000000000110000
00010110100000000000101
```

机器语言的每条二进制编码指令仅由 0 和 1 组成，不易记忆、不易查错、不易修改，可移植性和重用性差。上面只是一个非常简单的小程序，就暴露了机器码的晦涩难懂和不易查错。写如此小的一个程序尚且如此，一个至少要有几十行机器码的程序，其情况将更加复杂。为了克服上述缺点，出现了另一种采用一定含义的符号英文单词缩写指令助记符来表示指令的程序语言——汇编语言。

（2）汇编语言

汇编语言（Assembly Language）也是面向机器的程序设计语言。在汇编语言中，用助记符（Mnemonic）代替操作码，用地址符号（Symbol）或标号（Label）代替地址码。所以这种用符号代替机器语言的二进制码的计算机语言也被称为符号语言。

汇编语言编写的程序，机器不能直接识别，必须用一种软件将汇编语言翻译成机器语言，这种具有翻译功能的软件就是汇编软件。汇编软件把汇编语言翻译成机器语言的过程称为汇编。

汇编语言相对机器语言而言易读、易写、易记，它与机器语言指令是一一对应的，所以汇编语言不具有高级语言（如 Pascal 语言、C 语言）通用性强的特点，而是与计算机内部硬件结构密切相关，为某种计算机独有。汇编语言源程序汇编的过程如图 1-1 所示。

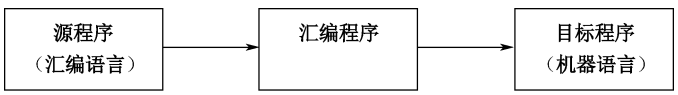


图 1-1　汇编语言源程序汇编的过程

尽管汇编语言具有执行速度快和易于实现对硬件的控制等优点，但它仍存在着机器语言的某些缺点：汇编语言依赖于硬件体系，与 CPU 的硬件结构紧密相关，不同的 CPU 其汇编语言是不同的，汇编语言程序不能移植；其次，进行汇编语言程序设计，必须了解所使用硬件的结构与性能，对程序设计人员有较高的要求，因此这种语言难以普及应用，为此便产生了第三代接近人类自然语言的高级语言。

（3）面向过程的高级语言

高级语言语法和结构更类似普通英文，更关键的是不依赖于特定计算机的结构与指令系统，用同一种高级语言缩写的源程序，一般可以在不同计算机上运行而获得同一结果，与计算机的硬件结构没有多大关系。

目前常用的高级语言有 BASIC、FORTRAN、COBOL、PASCAL、PL/M、C 等。一般来说，高级语言在编程时不需要对机器结构及其指令系统有深入的了解，而且用高级语言编写的程序通用性好、便于移植。相对面向机器的低级语言而言高级语言具有以下优点：

- 高级语言更接近人类自然语言，易学、易掌握，一般工程技术人员只要几周时间的培训就可以胜任程序员的工作；

- 高级语言为程序员提供了结构化程序设计的环境和工具，使设计出来的程序可读性好、可维护性强、可靠性高；
- 高级语言与具体的计算机硬件关系不大，因而所写出来的程序可移植性好，重用率高；
- 将繁杂琐碎的事务交给编译程序做，自动化程度高，开发周期短，使程序员可以集中时间和精力去提高程序的质量。

高级语言完全采用了符号化的描述形式，用类似自然语言的形式描述对问题的处理过程，使得程序员可以认真分析问题的求解过程，不需要了解和关心计算机的内部结构和硬件细节，更易于被人们理解和接受。随着计算机的发展，从 20 世纪 80 年代以来，众多的第四代非过程化语言、第五代智能化语言竞相推出，第四代语言将原来程序员告诉计算机怎么做，变成了程序员告诉计算机做什么，这是一种全新的开发方式，第四代语言就是被人们称为的面向对象语言。

（4）面向对象的高级语言

面向对象语言（Object-Oriented Language）是以对象作为程序基本结构单位的程序设计语言，程序设计的核心是对象，对象是程序运行的基本成分。面向对象语言的发展有两个方向：一种是纯面向对象语言，如 Smalltalk、Eiffel 等；另一种是混合型面向对象语言，即在过程式语言及其他语言中加入类、继承等成分，如 C++、Objective-C 等。面向对象语言刻画客观系统较为自然，便于软件扩充与复用。

综上所述，以上四种计算机语言各有优缺点。程序员在使用时，需根据应用场合选用。一般在实时控制中，特别是在对程序的空间和时间要求很高，需要直接控制设备的场合，通常采用汇编语言；在系统程序设计、多媒体应用、数据库等诸多领域采用面向对象语言比较合适，然而面向过程是程序设计的基础，所以对程序设计的初学者来说，只有学习好面向过程的程序设计，才能对程序设计的学习打下坚实的基础，所以本书采用 C 程序设计语言为背景，介绍程序设计的基本概念和方法。

3．计算思维

计算思维（Computational Thinking）是运用计算机科学的基础概念进行问题求解、系统设计、以及人类行为理解等涵盖计算机科学之广度的一系列思维活动。

2006 年 3 月，美国卡内基·梅隆大学计算机科学系主任周以真（Jeannette M·Wing）教授在美国计算机权威期刊《Communications of the ACM》杂志上给出并定义计算思维。周教授为了让人们更易于理解，又将它更进一步地定义为：通过约简、嵌入、转化和仿真等方法，把一个看来困难的问题重新阐释成一个我们知道问题怎样解决的方法；是一种递归思维，是一种并行处理，是一种把代码译成数据又能把数据译成代码，是一种多维分析推广的类型检查方法；是一种采用抽象和分解来控制庞杂的任务或进行巨大复杂系统设计的方法，是基于关注分离的方法（SoC 方法）；是一种选择合适的方式去陈述一个问题，或对一个问题的相关方面建模使其易于处理的思维方法；是按照预防、保护及通过冗余、容错、纠错的方式，并从最坏情况进行系统恢复的一种思维方法；是利用启发式推理寻求解答，也即在不确定情况下的规划、学习和调度的思维方法；是利用海量数据来加快计算，在时间和空间之间，在处理能力和存储容量之间进行折衷的思维方法。

计算思维吸取了问题解决所采用的一般数学思维方法，现实世界中巨大复杂系统的设计与评估的一般工程思维方法，以及复杂性、智能、心理、人类行为的理解等的一般科学思维方法。计算思维建立在计算过程的能力和限制之上，计算方法和模型使我们敢于去处理那些原本无法由

个人独立完成的问题求解和系统设计。计算思维最根本的内容是抽象（Abstraction）和自动化（Automation）。计算思维中的抽象完全超越物理的时空观，并完全用符号来表示，其中，数字抽象只是一类特例。与数学和物理科学相比，计算思维中的抽象显得更为丰富，也更为复杂。

计算思维所关注的核心问题是人的思维方式及问题求解能力的培养。在本课程的学习中，一个重要的内容是以系统化、逻辑化的计算思维方式去思考问题和解决问题，着重培养计算思维能力，强化工程化、系统化程序设计的观念和能力。

1.2　算法

算法是程序设计的精髓。计算机科学家、PASCAL 语言的发明者尼克劳斯·沃思（Niklaus Wirth）曾提出一个著名的公式：程序=算法 + 数据结构 。计算机解题的过程中，无论是形成解题思路还是编写程序，都是在实施某种算法。前者是推理实现的算法，后者是操作实现的算法。学习程序设计，还要养成一种严谨的软件开发习惯，熟悉软件工程的基本原则。

1．算法的概念

什么是算法？当代著名计算机科学家 D·E·Knuth 在他的一本书中写到：“ 一个算法，就是一个有穷规则的集合，其中之规则规定了一个解决某一特定类型的问题的运算序列。” 简单地说，算法（Algorithm）就是确定的解决问题方法和有限步骤。

在计算机科学中，算法要用计算机算法语言描述，算法代表用计算机解决一类问题的精确、有效的方法。需要明确的是，不是只有科学计算才有算法，在日常生活中做任何一件事情，都是按照一定规则，一步一步地进行。比如在工厂中生产一部机器，首先把零件按一道道工序进行加工，然后，再把各种零件按一定规则组装成一部完整机器，这个工艺流程其实就是算法；在农村，种庄稼有耕地、播种、育苗、施肥、中耕、收割等各个环节，这些栽培技术也是算法。

计算机算法通常可以分为两大类：一类是用于解决数值计算，就是数值计算的算法，如科学计算中的数值积分、解线性方程等计算方法；另一类是用于解决非数值计算，就是非数值计算的算法，如信息管理、文字处理、图像图形处理，进行排序、分类、查找等操作。

下面通过三个问题的解决过程来说明算法设计的基本思维方法。

【例 1-1】求 1×3×5×7×9。

算法分析：这是一最原始方法：

步骤 1：先求 1×3，得到结果 3。

步骤 2：将步骤 1 得到的乘积 3 乘以 5，得到结果 15。

步骤 3：将 15 再乘以 7，得 105。

步骤 4：将 105 再乘以 9，得 945。

这样的算法虽然正确，但太繁琐，也不具有解决此类问题的通用性。

改进的算法：

步骤 1：使 t=1。

步骤 2：使 i=3。

步骤 3：使 t×i，乘积仍然放在在变量 t 中，可表示为 t×i➞t。

步骤 4：使 i 的值+2，即 i+2➞i。

步骤 5：如果 i≤9，返回重新执行步骤 3 以及其后的步骤 4 和步骤 5；否则，算法结束。

这样改进后，可以解决符合条件有规律序列的诸如此类的问题，更具有通用性。

【例 1-2】 计算函数 $f(x)$的值。函数 $f(x)$为：

$$f(x)=\begin{cases}3x+a & x\leqslant a\\ ax+b+c & x>a\end{cases}$$

其中，a,b,c 是常数。

算法分析：本题中属于一种函数题目，有两个不同的表达式子，根据输入 x 的值决定采用哪个公式计算，使用计算机解题的算法如下：

步骤 1：将 a,b,c 和 x 的值输入到计算机中。

步骤 2：判断 $x\leqslant$a 是否成立，如果条件成立，执行步骤 3，否则执行步骤 4。

步骤 3：按照表达式 $3x+a$ 计算出 $f(x)$的结果，然后执行步骤 5。

步骤 4：按照表达式 $ax+b+c$ 计算出 $f(x)$的结果，然后执行步骤 5。

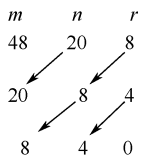
步骤 5：输出 $f(x)$的值。

步骤 6：算法结束。

【例 1-3】对给定的两个正整数 m 和 n(m 大于等于 n)，求它们的最大公约数。

算法分析：本题中属于一种数学数值运算题目，利用其成熟的算法可以很好地解决这个问题，这个算法就是辗转相除法求解最大公约数。

例如：假设 m=48，n=20，余数用 r 表示，用辗转相除法求它们的最大公约数的步骤为：



第一次：m/n=48/20，得余数 r 为 8，将 n 作为新的 m，以 r 作为新的 n，继续相除；

第二次：m/n=20/8，得余数 r 为 4，将 n 作为新的 m，以 r 作为新的 n，继续相除；

第三次：m/n=8/4，得余数 r 为 0，当余数 r 为 0 时，此时的 n 就是两数的最大公约数，所以 48 和 20 的最大公约数为 4。

由于 m 和 n 取值的不同，辗转的次数是不固定的，辗转的结束条件应该以判断余数 r 是否为 0。

使用计算机解题的算法描述如下：

步骤 1：将两个正整数分别存放到变量 m 和 n 中。

步骤 2：求余数：将 m 除以 n，所得到的余数存放到变量 r 中。

步骤 3：判断余数 r 是否为 0，如果余数为 0 则执行步骤 5，否则执行步骤 4。

步骤 4：更新被除数和除数：将 n 的值放入 m 中，余数 r 的值放入 n 中，然后转向执行步骤 2。

步骤 5：输出 n 当前的值。

步骤 6：算法结束。

通过以上几个例子，可以初步了解如何针对问题设计算法，每一个算法都是由一系列的操作指令组成的，主要包括基本操作和控制结构两个基本要素，基本操作包括加、减、乘、除、判断、置数等功能，控制结构包括顺序、分支、重复等基本结构。研究算法的目的不是精确地求问题的解，而是研究怎样把各种类型的问题的求解过程分解成一些基本的操作。

2．算法的特性

算法是有穷规则的集合，通过这些规则可以确定出解决某些问题的运算序列。对于该类问题的任何输入值，都需要一步一步地执行计算，经过有限步骤后终止计算并产生输出结果。归纳起来，算法包括以下基本特性：

（1）有穷性：有穷性的限制是指一个实用的算法，不仅操作的步骤是有限的，不能无休止地执行下去，而且尽可能的少。例如一个算法需要计算机运算上万年的时间才能完成，那么这个算法虽然有穷也没有实际使用意义。有穷性应该是在合理的范围内，合理限度应该以人类常识和需要来进行界定，没有统一的标准。

（2）确定性：算法中的每一步操作的内容和顺序必须确切，不能有模棱两可的二义性。

（3）可行性：算法中的每一步操作都必须是可执行的，也就是说算法中的每一步都能通过手工和机器在有限时间内完成，这也称之为有效性。例如，一个数被 0 除的操作就是无效的，应当避免这种操作。

（4）零个或多个输入：一个算法中有零个或多个输入。这些输入数据应在算法操作前提供。如例 1-2 算法中有 4 个输入，即需要输入 a、b、c 和 x 四个初始数据，有的算法也可以没有输入，又如：程序只输出一段文字信息到屏幕。

（5）一个或多个输出：一个算法中有一个或多个输出。算法的目的是用来解决一个给定的问题，如例 1-3 求两个数的最大公约数，经过运算，输出两个数的最大公约数，这就是输出的信息，否则，算法就没有意义了。

对于程序设计人员，必须会设计算法，并能根据算法写出程序。通常算法都要满足以上 5 个特征。

3．算法的描述

原则上来说，算法可以用任何形式的语言和符号工具来表示，采用不同的算法描述工具对算法的质量有很大的影响。通常算法常用的表示方法有自然语言、传统流程图、结构化 N-S 图、伪代码、程序设计语言等。这里重点介绍流程图和 N-S 图。

（1）自然语言

自然语言就是人们日常生活中使用的语言，可分为英语、汉语或其他语言等，理想状态下，算法的描述过程应当使用自然语言表达，因为其表示时算法通俗易懂，无需任何的专业训练就能看明白。然而，自然语言描述算法也有很多不足之处，如用自然语言表示算法篇幅冗长、语法和语意上不太严格，容易出现描述的歧义性。例如：小王对小李说他的母亲今天来了。从这样一句话可以理解成小王的母亲来了，也可以理解成小李的母亲来了，理解上存在歧义性。另外在描述分支和循环等算法方面也很不方便，因此除了很简单的问题，一般不用自然语言表示算法。

（2）程序流程图

程序流程图（Program Flow Chart）是最早提出的用图形表示算法的工具，也称为传统流程图。用它表示算法，具有直观性强、清晰性好、便于阅读、易于理解等特点，也便于转化成计算机程序设计语言，是软件开发人员经常使用的算法描述方式，具有程序无法取代的作用，但对程序流程图符号运用不规范，会使得算法显得混乱，应该要特别注意算法的结构化。图 1-2 所示是常用的程序流程图符号。

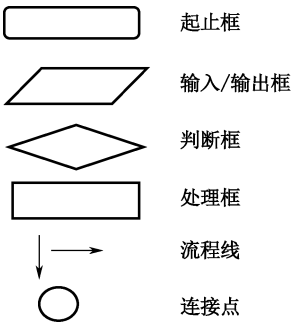


图 1-2　常用的程序流程图符号

【例 1-4】将【例 1-1】求 1×3×5×7×9 的算法用流程图表示，流程图如图 1-3 所示。如果需要打印出来运算结果，这可以用流程图 1-4 表示。

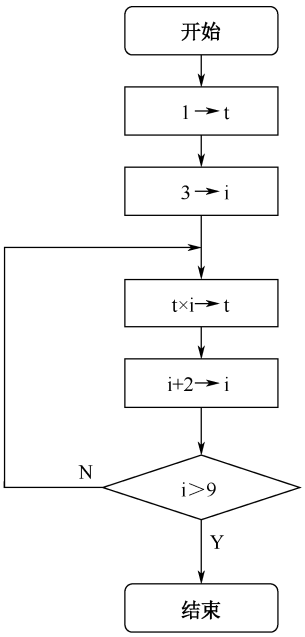


图 1-3　【例 1-4】流程图（1）

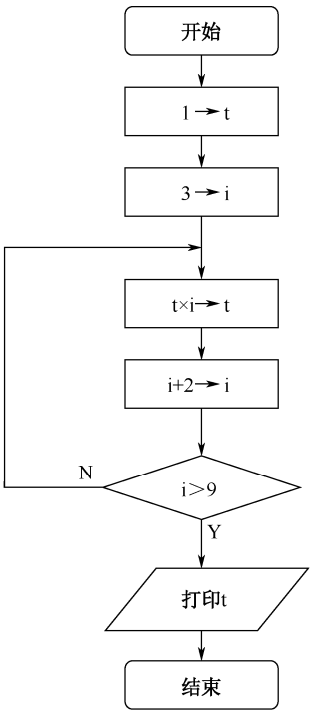


图 1-4　【例 1-4】流程图（2）

【例 1-5】将【例 1-2】的算法用流程图表示，流程图如图 1-5 所示。

【例 1-6】将【例 1-3】的算法用流程图表示，流程图如图 1-6 所示。

在结构化的程序设计中，人们规定了三种基本结构：顺序结构、分支结构（选择结构）和循环结构，这些结构按照一定的规律组成算法结构，可以保证算法质量。顺序结构的基本流程图如图 1-7 所示，分支结构的基本流程图如图 1-8 所示，循环结构的基本流程图如图 1-9 所示。

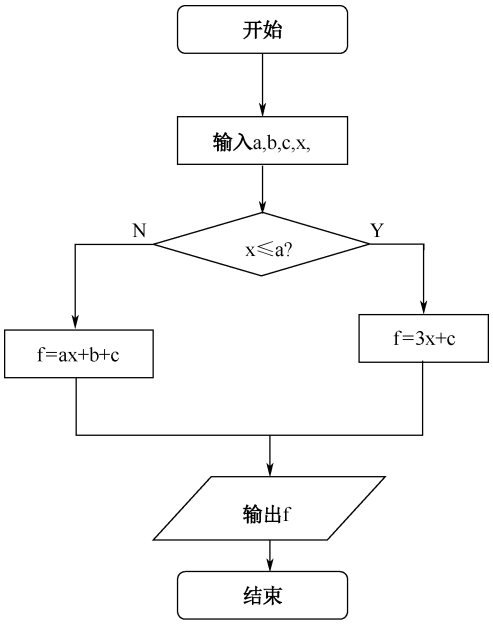


图 1-5　【例 1-5】流程图

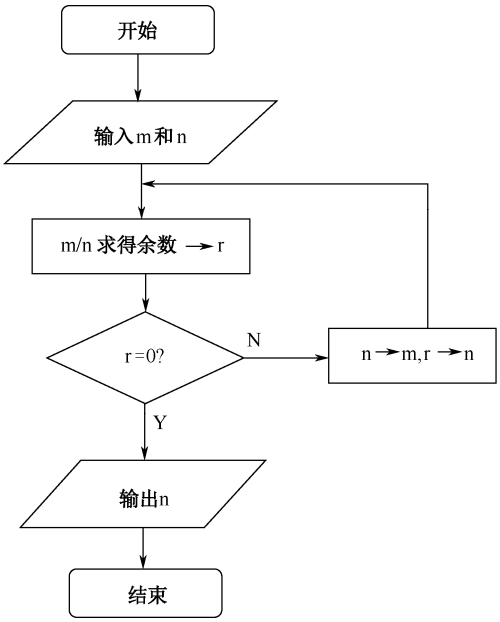


图 1-6　【例 1-6】流程图

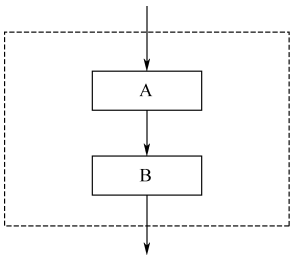


图 1-7　顺序结构的程序流程图

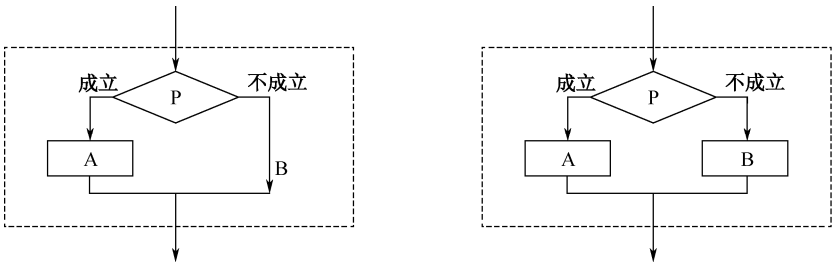
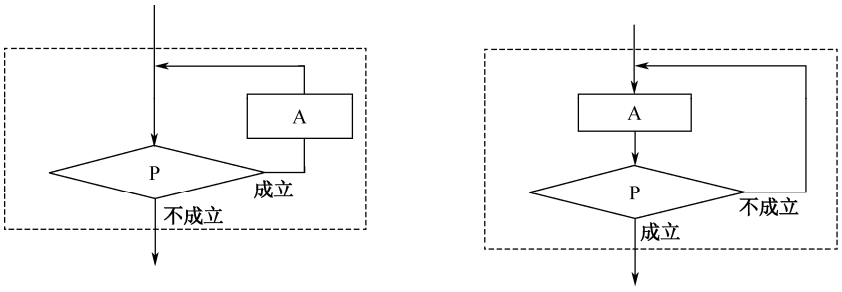


图 1-8　分支结构的程序流程图



（a）当型循环　　（b）直到型循环

图 1-9　循环结构的程序流程图

（3）N-S 流程图

N-S 流程图，是 1973 年美国学者 I·Nassi 和 B·Shneiderman 提出的一种新型流程图形式，其中 N-S 是两位学者的姓氏的首字母。这是一种去掉了流程线的流程图，算法的描述是在一个矩形框内，每个框内又可以包含下级矩形框，一个矩形框表示一个独立功能的 N-S 图，因为其符合结构化程序设计要求的特点，比较适合在软件工程中进行使用。三种基本程序结构的 N-S 流程图符号表示如下：

①顺序结构：顺序结构表示如图 1-10 所示，程序流从 A 矩形框到 B 矩形框。

②选择结构：选择结构流程图如图 1-11 表示，当条件 P 成立时，程序执行 A 框流程；当条件 P 不成立时，执行 B 框流程。

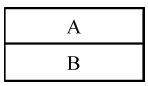


图 1-10　顺序结构的 N-S 图

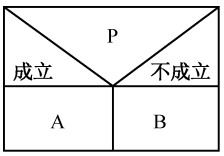
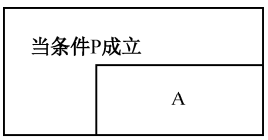
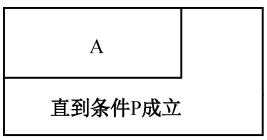


图 1-11　选择结构的 N-S 图

③循环结构：循环结构分别用图 1-12（a）和图 1-13（b）来表示。图 1-13（a）表示当条件 P 成立时，始终执行 A 框流程；否则，就终止执行 A 框流程。图 1-13（b）表示始终执行 A 框流程，直到条件 P 成立时终止执行 A 框流程。



（a）当型循环



（b）直到型循环

图 1-12　循环结构的 N-S 图

（4）伪代码

虽然用流程图描述算法直观易理解，但画起来比较麻烦，当算法稍微复杂时，算法修改很不方便。为使算法设计容易修改，使用伪代码描述算法比较合适。伪代码也称为类程序设计语言，是一种近似高级语言但又不受语法约束的一种算法语言描述形式，是介于自然语言和计算机语言之间的文字符号来描述算法。通过用伪代码描述的算法可以看到，其中可以包含计算机语言语句，也不必像计算机语言语法那样严格，有些地方还可以用自然语言进行描述。这样的方式方便修改算法，很大程度上可以简化算法的设计工作。

（5）计算机语言

直接使用计算机语言书写算法，可以令算法由设计到实现一步到位，通常对计算机语言非常熟悉的程序员来说，直接使用计算机语言编写算法很方便，这种方法的优点是避免了算法到计算机语言的转换，其缺点是要求程序员在书写时严格谨慎，必须按照程序语言的语法规则编写语句，否则无法运行，同时改成其他计算机语言编程时，又需按照新的计算机语言规则重新编写，对于不熟悉程序语言的人来说，看不懂书写的算法。

综上所述，对于设计好的任何一个算法，无论用哪一种算法描述方法，目的是一样的，即规范、准确地描述出解决问题的步骤，以便于下一步的编写程序。

1.3　软件的编制步骤

日常生活中可以使用计算机进行画图、制表或者闲暇时听歌曲、看电影，计算机可以帮我们做很多的事情，计算机作为一种批量生产的设备，只有在安装了不同的软件，才能处理问题，换句话说，如果没有计算机软件的产生及其广泛应用，计算机也就不会在今天的工作以及生活中产生如此巨大的影响。

一个软件的开发，包含需求分析、可行性分析、初步设计、详细设计、形成文档、建立初步模型、编写详细代码、测试修改、发布等多个步骤。首先要找到解决问题的突破口（先要搞明白需要做什么，然后再考虑如何做）。至于采用什么表示方法（简单文本、UML 图、E-R 图）、什么语言开发工具都是次要的问题。总体来看软件开发过程主要包括以下四个阶段：

第一阶段：确定软件开发需求。

第二阶段：软件设计与开发。

- 问题分析
- 选择一个全面的解决问题的算法方案
- 编写程序
- 调试程序

第三阶段：文档整理。

第四阶段：软件维护。

1．确定软件开发需求

这个阶段主要包含问题定义和可行性研究两个阶段。

（1）问题定义阶段。这个阶段必须回答的关键问题是：“ 要解决的问题是什么 ”。

（2）可行性研究阶段。这个阶段要回答的关键问题是：“ 上一个阶段所确定的问题是否有行得通的解决办法 ”。

软件需求分析阶段的任务仍然不是具体地解决客户的问题，而是准确地回答“ 目标系统必须做什么 ”这个问题。这个阶段的另外一项重要任务，是用正式文档准确地记录目标系统的需求，这份文档通常称为规格说明(Specification)，主要包括用户视图、数据词典和用户操作手册。除了以上工作，作为项目设计者应当完整地做出项目的性能需求说明书，因为往往性能需求只有懂技术的人才可能理解，这就需要技术专家和需求方进行真正的沟通和了解。

2．软件设计与开发

对于程序设计的初学者来说，编写出来一个符合语法规则，运行结果正确的程序是程序设计的首要目标，对于优秀的程序员来说，除了程序的正确性以外，其更加注重的是程序的可读性、可靠性以及较高的结构化程序设计理念。一般程序设计主要包含以下四个步骤：

（1）分析问题，建立模型。这个步骤务必确定待解决的问题已经非常明确，能够为解决问题的算法提供必要的信息。因为只有完全理解了问题，才能清晰地定义和分析问题。这个步骤中针对解决的问题，必须明确输出的对象、输入的数据、预期输出的结果以及输入到输出过程中的数学模型。一般情况下，许多问题的数学模型和解决方法非常简单，以至于我们都未感觉到解决问题模型的存在。但是对更多的问题来说，数学模型建立的对错与好坏在很大程度上决定了程序开发的正确性和复杂性。

（2）明确一个全面的算法解决方案。在这个步骤中，根据建立的数学模型，选择并确定一个合适的解决问题的算法。这里的算法是泛指解决问题的方法和步骤，而不仅仅是具体的精确

“计算”。有时候确定一个全面的算法是比较容易的，也有时选择一个全面的解决方案是比较困难的。

（3）编写程序。这个步骤将确定好的解决问题算法转换为计算机程序，这就涉及对算法的编码。若问题分析和解决方案具体步骤已经确定，编写程序代码就是按部就班的机械式工作，按照确定好的数据结构和算法解决方案，采用某种程序设计语言严格地描述出来。

（4）调试程序。调试程序最终目的就是测试程序运行的结果是否正确，是否能够满足用户的需求，根据测试结果进行调整，直到测试取得预期的结果。在测试的过程中，不仅要用预期正确的用例，还要尽量多采用预期错误的用例，以尽可能发掘出程序的设计漏洞。调试的过程是“测试—修改—再测试—再修改”往复循环的过程，直至设计出的程序是正确的、健壮的。

3．文档整理

实际上，绝大多数程序员会忘记他们几个月前自己所开发程序的大部分细节。如果他们或者其他程序员需要对程序做后续的修改工作，不得不将大量宝贵的时间浪费在了解原来的程序是如何工作的这个事情上。然而，清晰的说明文档可以避免这些问题的发生。很多的案例教训给我们了这样的提醒，如果没有足够的说明文档，会造成很多工作不得不重新去完成，这足以说明文档整理在整个软件开发过程中具有相当重要的作用，事实上，很多的开发文档在程序的分析、设计、编码和测试阶段都要整理出来。完成文档整理需要收集文档信息、增加辅助资料，然后按照你和你的团队需要的格式表示出来。文档整理阶段一般开始于程序开发需求确定的第一阶段，并且会一直延续到系统的维护阶段。

尽管对问题解决来说，不是所有人都是按照这个方式进行文档整理的，但是以下六个文档整理是必需的：需求分析文档、已经编码的算法描述文档、编写代码期间对程序代码的注释、对程序后续的修改和变化描述文档、每次包含输入和运行取得结果的测试运行文档、一个能详细讲述如何使用程序的用户手册。

4．系统维护

完成系统测试、验收后的整体项目才算告一段落。然而系统升级、维护等工作是系统开发中一个不可缺少的环节，有必要通过各种维护活动使系统一直满足用户的需要，作为一个完善的系统开发过程，需要跟踪软件的运营状况并持续修补升级，直到这个软件被彻底淘汰为止。在系统维护中通常有以下四类维护：

（1）改正性维护，也就是诊断和改正在使用过程中发现的软件错误。

（2）适应性维护，即修改软件以适应环境的变化。

（3）完善性维护，即根据用户的要求改进或扩充软件使它更完善。

（4）预防性维护，即修改软件为将来的维护活动预先做准备。

软件开发的实践表明，在开发的早期阶段越仔细，后期的测试和维护费用就会越少。因而，在系统分析和设计阶段应特别重视。

1.4　C 程序设计语言的产生与特点

C 语言是一种得到广泛重视并普遍应用的计算机语言，也是国际上公认的最重要的几种通用程序设计语言之一。C 语言是一种结构化程序设计语言，它层次清晰，便于按模块化方式组织程序，易于调试和维护。C 语言的表现能力和处理能力极强。它不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构。它还可以直接访问内存的物理地址，进行位（bit）一

级的操作。由于C语言实现了对硬件的编程操作，因此C语言集高级语言和低级语言的功能于一体，它既可用来编写系统软件，也可用来编写应用软件。

C 语言是在 20 世纪 70 年代初问世的。Dennio.M.Ritchie 于 1972 年在美国贝尔实验室设计实现了 C 语言，C 语言直接来源于 B 语言，但它的根源可以追溯到 ALGOL60。ALGOL60 结构严谨，其设计者非常注重语法、分程序结构，因此对于后来许多重要的程序设计语言，如 PASCAL、PL/I、SIMULA67 产生过重要的影响。但它是面向过程的语言，与计算机硬件相距甚远，不适合编写系统软件。1963 年英国剑桥大学在 ALGOL60 的基础上推出更接近硬件的 CPL 语言，但 CPL 太复杂，难于实现。1967 年，剑桥大学的 Matin.Rinchards 对 CPL 语言作了简化，推出了 BCPL 语言。1970 年贝尔实验室的 Ken.Thompson 以 BCPL 为基础，设计了更简单也更接近硬件的 B 语言（取 BCPL 的第一个字母）。B 语言是一种解释性语言，功能上也不够强，为了很好地适应系统程序设计的要求，Ritchie 把 B 发展成称之为 C 的语言（取 BCPL 的第二个字母）。C 语言既保持了 BCPL 和 B 的优点（如精练，接近硬件），又克服了它们的缺点(如过于简单,数据无类型等)。1973 年 Ken.Thompson 和 Dennio.M.Ritchie 用 C 改写了 UNIX 代码，并在 PDP-11 计算机上加以实现，即 UNIX 版本 5，这一版本奠定了 UNIX 系统的基础，使 UNIX 逐渐成为最重要的操作系统之一。图 1-13 展示了 C 语言的由来。

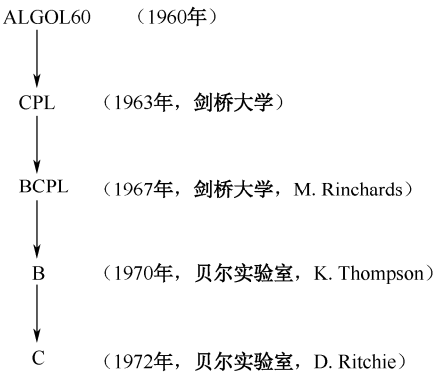


图 1-13　C 语言的由来

C 语言设计的目的是为描述和实现 UNIX 操作系统提供一种工具语言。但并没有被缚束在任何特定的硬件或操作系统上，它具有良好的可移植性。C 语言的使用覆盖了几乎计算机的所有领域，包括操作系统、编译程序、数据库管理程序、CAD、过程控制、图形图像处理等。

C 语言之所以能够被世界计算机界广泛接受，正是其本身具有不同于其他语言的突出特点。C 语言主要特点如下。

（1）表达能力强。丰富的数据类型和运算符，其中运算符有 34 种，C 把括号、赋值、逗号等都作为运算符处理，从而使 C 的运算类型极为丰富，可以实现其他高级语言难以实现的运算。

（2）可直接访问内存物理地址和硬件寄存器，能实现二进制位运算。

（3）C 语言结构清晰，流程控制结构化，程序结构模块化。具有顺序、分支、循环三种结构化控制结构，便于开发大型软件。

（4）语言简练、紧凑。如 i=i+1，在 C 中可写为 i++。

（5）生成的目标代码质量高，程序执行速度快。编译后生成的目标程序运行速度快，占用存储空间少，几乎与汇编语言相媲美。

1.5　简单程序设计

1.5.1　C 语言的字符集

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

（1）字母：小写字母 a～z 共 26 个，大写字母 A～Z 共 26 个。

（2）数字：0～9，共 10 个。

（3）空白符：空格符、制表符、换行符等统称为空白符。空白符只在字符常量和字符串常量中起作用。在其他地方出现时，只起间隔作用，编译程序对它们忽略。因此在程序中使用空白符与否，对程序的编译不发生影响，但在程序中适当的地方使用空白符将增加程序的清晰性和可读性。

（4）标点和特殊字符：主要有！、#、%、^、&、+、-、*、/、=、～、<、>、\、|、,、:、;、'、"、(、)、{、}、[、]等。

在 C 语言中使用的词汇分为 6 类：标识符、关键字、运算符、分隔符、常量、注释符。

1．标识符

在程序中使用的变量名、宏名、函数名等统称为标识符。除库函数的函数名由系统定义外，其余都由用户自定义。

在 C 语言中规定：

（1）标识符只能是字母(A～Z，a～z)、数字(0～9)、下划线(_)组成的字符串；

（2）第一个字符必须是字母或下划线。

以下标识符是合法的：

```
a,  x,  BOOK_1, sum5
```

以下标识符是非法的：

```
3s        （以数字开头）
s*T       （出现非法字符*）
-3x       （以减号开头）
bowy-1    （出现非法字符-(减号)）
```

在使用标识符时还必须注意以下几点：

（1）标准 C 不限制标识符的长度，但它受各种版本的 C 语言编译系统限制，同时也受到具体机器的限制。例如在某版本 C 中规定标识符前 8 位有效，当两个标识符前 8 位相同时，则被认为是同一个标识符。

（2）在标识符中，大小写是有区别的。例如 N 和 n 是两个不同的标识符。

（3）标识符虽然可由程序员随意定义，但标识符是用于标识某个量的符号。因此，命名应尽量有相应的意义，以便阅读理解，作到“ 见名知义 ”。

2．关键字

关键字是由C语言规定的具有特定意义的字符串，通常也称为保留字。用户定义的标识符不应与关键字相同。C 语言的关键字分为以下几类：

① 类型说明符

用于定义、说明变量、函数或其他数据结构的类型。如 int, double 等。

② 语句定义符

用于表示一个语句的功能。如 if else 就是条件语句的语句定义符。

③ 预处理命令字

用于表示一个预处理命令。如 include。

3．运算符

C 语言中含有相当丰富的运算符。运算符与变量函数一起组成表达式，表示各种运算功能。

运算符由一个或多个字符组成。

4．分隔符

在 C 语言中采用的分隔符有逗号和空格两种。逗号主要用在类型说明和函数参数表中，分隔各个变量。空格多用于语句各单词之间，作间隔符。在关键字标识符之间必须要有一个以上的空格符作间隔，否则将会出现语法错误，例如把 int a 写成 inta，C 编译器会把 inta 当成一个标识符处理，其结果必然出错。

5. 常量

C 语言中使用的常量可分为数字常量、字符常量、字符串常量、符号常量、转义字符等多种。

6. 注释符

C 语言的注释符是以“ /* ”开头并以“ */ ”结尾的串。在“ /* ”和“ */ ”之间的即为注释。以“ // ”开头的为单行注释。程序编译时，不对注释作任何处理。注释可出现在程序中的任何位置。注释用来向用户提示或解释程序的意义。在调试程序中对暂不使用的语句也可用注释符括起来，使编译程序跳过不作处理，待调试结束后再去掉注释符。

1.5.2　简单 C 程序举例

为了说明 C 语言源程序结构的特点，先看一个简单的程序。这个程序表现了C语言源程序在组成结构上的特点。虽然有关内容还未介绍，但可从这个例子中了解到组成一个 C 源程序的基本部分和书写格式。

【例 1-7】输出一段欢迎语。

打开 VC++6.0 编译器，新建控制台应用程序，创建 C++源文件，在程序编辑窗口输入如下程序：

```
#include <stdio.h>
void main()
{
    printf("\n");
    printf("    ************************************************\n");
    printf("    *       欢迎来到C语言课堂，让我们一起探索C的奥秘       *\n");
    printf("    ************************************************\n");
    printf("\n");
}
```

程序输入后，编辑界面如图 1-14 所示。

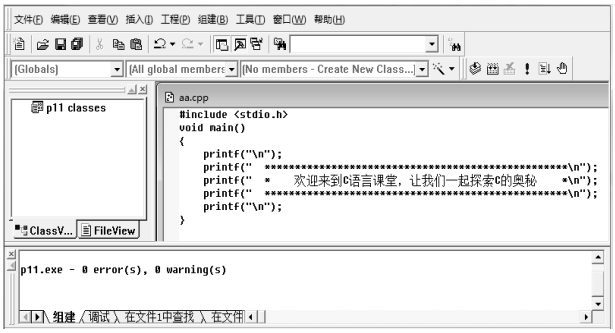


图 1-14　【例 1-7】编辑界面

对程序进行“编译”—“连接”—“运行”，可输出如图 1-15 所示的运行界面。

图 1-15　【例 1-7】运行界面

【例 1-8】输入三角形的三边长，求三角形面积。

分析：已知三角形的三边长 *a*,*b*,*c*，则该三角形的面积公式为：

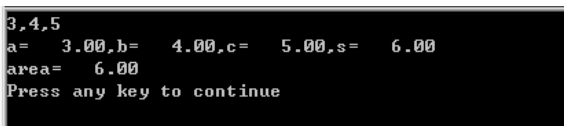
$$\text{area}=\sqrt{s(s-a)(s-b)(s-c)}\,,$$

其中 $s=(a+b+c)/2$，所以只需要输入三个边，就可以得到三角形的面积。

```
#include <math.h>                                          //预处理语句
#include <stdio.h>                                         //预处理语句
int main()
{
    float  a,b,c,s,area;                                   //变量定义语句
    scanf("%f,%f,%f",&a,&b,&c);                            /* 输入语句，输入边长a,b,c */
    s=1.0/2*(a+b+c);
    area=sqrt(s*(s-a)*(s-b)*(s-c));                        /* 计算三角形面积 */
    printf("a=%7.2f,b=%7.2f,c=%7.2f,s=%7.2f\n",a,b,c,s);
                                                           /* 输出三边长a,b,c及s*/
    printf("area=%7.2f\n",area);                           /* 输出三角形面积 */
    return 0;                                              //返回0，main()函数结束
}
```

程序的运行情况为：

若输入三边长：3,4,5，程序运行结果如下：



这是一个典型的简单数据计算的程序，程序仅由一个主函数组成，main()函数主体结构包括变量定义、数据输入、数据运算、结果输出。程序中使用到的变量必须首先定义，要注意的是一个程序可以有零个或多个输入，但必须有一个或多个输出。

【例 1-9】输入两个正整数 *m* 和 *n*，求它们的最大公约数。

这里利用前面介绍过的辗转相除算法，编制程序如下：

```
#include <stdio.h>
int main()
{
    int m,n,r;
    scanf("%d,%d",&m,&n);
    r=m%n;
    while (r!=0)
    {
        m=n;
```

```
            n=r;
            r=m%n;
        }
        printf("公约数为：%d\n",n);
        return 0;
    }
```

输入前面分析过的数据 48 和 20，程序运行结果如下：

```
48,20
公约数为：4
Press any key to continue
```

读者可对照前面的算法描述和这里的 C 程序代码，领会由算法设计到编程的过程。

下面看一个包含多个函数的例子。

【例 1-10】 计算两个数的和。

```
#include "stdio.h"
int main()
{
    int sum(int x,int y);                    //对被调函数sum的声明
    int a,b,c;                               //定义变量a、b、c
    scanf("%d,%d",&a,&b);                    //输入变量a和b的值
    c=sum(a,b);                              //调用sum函数，将得到的值赋给c
    printf("sum=%d\n",c);                    //输出c的值
}
int sum(int x,int y)                         //定义sum函数，函数值为整型，形式参数x、y为整型
{
    int z;
    z=x+y;                                   //将x、y的值相加赋给变量z
    return(z);                               //将z的值返回，通过sum带回到调用函数的位置
}
```

这是一个典型的包含函数调用的例子，本程序包含两个函数：主函数 main()和被调函数 sum()。sum()函数的作用是将 x 和 y 的和赋给变量 z。return 语句将 z 的值返回给主调函数 main()。由于 sum()函数在 main()函数之后，为了编译系统能够正确识别和调用 max()函数，需要在 main()函数中对被调函数 sum()进行声明。

程序运行时，首先由 main()函数入口，执行输入语句 scanf("%d,%d",&a,&b)，为变量 a、b 输入值，然后执行到 c=sum(a,b) 语句时，程序的走向转到函数 int sum(int x,int y) 执行，遇到 return(z)语句，则将 z 的值带回，返回到 main()函数中继续往下执行，直到 main()函数结束。

程序运行情况如下：

```
6,12
sum=18
Press any key to continue
```

首先用户按照 scanf("%d,%d",&a,&b)语句格式输入“ 6,12 ”，程序运行的结果按照 printf("sum =%d\n",c)语句格式输出“ sum=18 ”。

通过这几个例子，我们直观地了解了 C 语言源程序的组成，具体的语法细节将在以后的章节学习到。

1.6 错误解析

对于每位程序设计的初学者来说，在编写程序的过程中不断出现其他初学者已经出现过的错误，这是学习程序设计语言的必然部分，每类程序语言都有它本身的一系列的疏忽问题而产生的错误，而且这些错误有时会觉得很低级。以后每章节都会将和本章有关的常见错误及解决方法列出以便引起读者的注意。

（1）在没有花费足够时间去仔细研究待解决的问题和未进行正确算法的设计，就急急忙忙进行程序开发和运行。这样就会造成所编写的程序因考虑不全面而导致无法得到预期的正确结果。解决办法：要想编写一个成功的程序，前提是对相关问题充分理解的基础上，设计好正确的算法，然后再开始编写程序代码，否则会造成编写程序匆忙，而需要花费在调试和修改程序上的时间就会越长。

（2）忘记对编写程序进行备份。几乎所有的程序编写初学者都会犯这个错误。解决办法：为避免出现程序开发过程的意外问题，应及时将开发的程序进行备份。

（3）计算机不会理解通过自然语言描述的算法，计算机只识别用某种计算机编程语言编写的具体程序指令。解决办法：要让计算机为我们工作，就必须编写计算机能够识别的程序，才能让计算机完成指定的任务。

（4）书写标识符时,忽略了大小写字母的区别。比如，程序中定义了两变量 a，A；然后在本该用 a 的时候误写为 A，结果自然是两个变量在计算中出现冲突，无法得到正确的结果。解决办法：加强编程规范，统一命名规则，避免使用（仅大小写不同的）同名标识符。

（5）忘记加分号。分号是 C 语句中不可缺少的一部分，语句末尾必须有分号。解决办法：语法错误，编译会报错，要学会读懂错误提示，要培养正确习惯，提高编程效率。

本章小结

通过本章的学习，我们了解了程序设计语言由低级语言发展到高级语言的过程，了解了计算思维的概念，了解了算法的概念及特征，学习了算法的三种描述方法：程序流程图、N-S 流程图、伪语言，了解了 C 语言的产生和发展，通过实例了解了 C 语言的程序组成及调试过程。

C 语言在程序设计语言中算得上是唯一一门历史悠久且目前依然流行的语言，它之所以在信息飞速发展的时代还能独树一帜、经久不衰，是因为 C 语言直接能与计算机底层打交道，程序精巧、灵活、高效，所以在很多领域依然在使用。

习题 1

一、选择题

1. 下列关于 C 语言注释的叙述中错误的是（　　）。
   A. 以“ /* ”开头并以“ */ ”结尾的字符串为 C 语言的注释内容
   B. 注释可出现在程序中的任何位置，用来向用户提示或解释程序的意义
   C. 程序编译时，不对注释作任何处理
   D. 程序编译时，需要对注释进行处理
2. 在 Visual C++ 6.0 环境下，C 源程序文件名的默认后缀是（　　）。