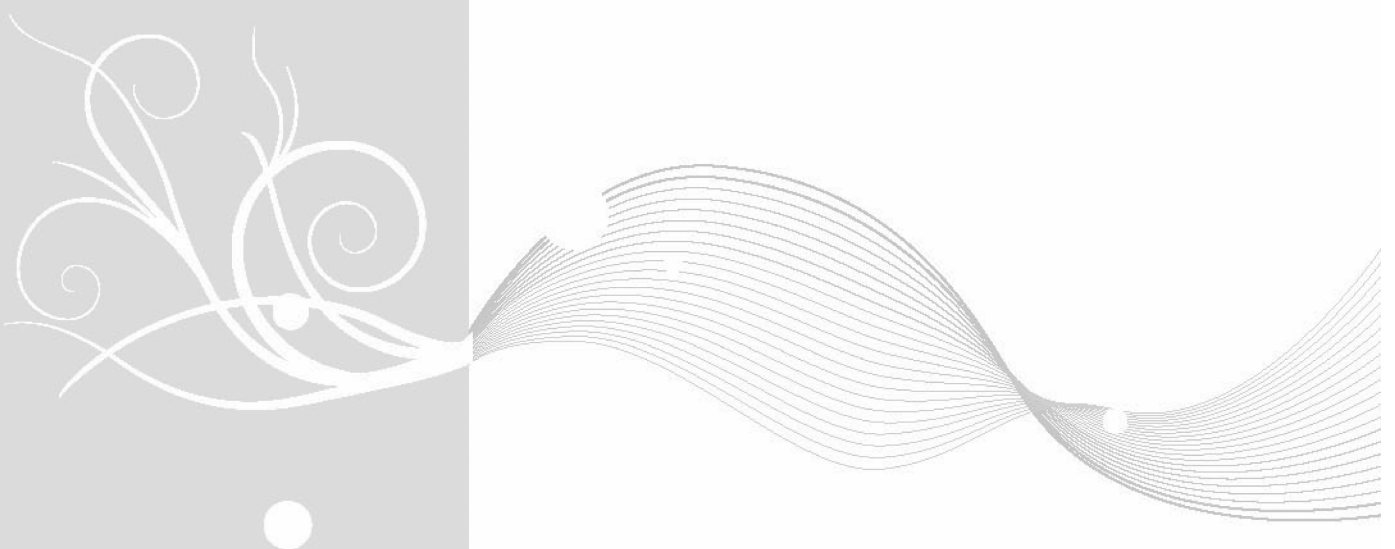
第1部分 初 级 篇



- 第1章　C语言学习基础
- 第2章　C程序设计初步
- 第3章　控制结构和数据文件

第1章 C语言学习基础

——并不是有了工作才有目标，而是有了目标才能确定每个人的工作。

学时分配

课堂教学：6 学时。自学：4 学时。自我上机练习：2 学时。

本章教学目标

(1) 了解 C 语言的基本知识。
(2) 认识计算机辅助问题求解过程。
(3) 理解算法的概念并掌握算法描述方法。
(4) 认识程序的三种基本结构。
(5) 应用 C 语言基本词汇描述简单问题。
(6) 了解数据及代码在内存中的存储与运行。
(7) 模仿例题编制自己的第一个程序。

本章项目任务

软件界面的初始设计。

1.1* 预备知识：计算机系统的硬件与软件

计算机系统由硬件(Hardware)系统和软件(Software)系统两大部分组成。硬件系统由运算器、控制器、存储器、输入设备、输出设备五大基本部件构成，如图 1-1 所示。软件系统是指程序、数据和文档资料的总称。

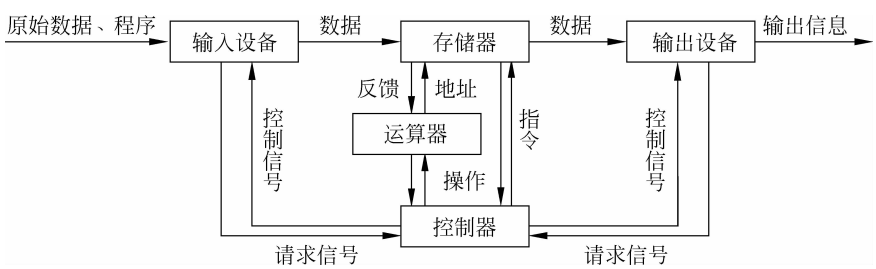


图 1-1 计算机硬件系统的组成

1. 计算机硬件

(1) CPU

中央处理器(Central Processing Unit，CPU)是计算机的指挥中心和心脏，主要完成算术和逻辑运算，控制计算机各部件统一协调工作。CPU 由运算器(Arithmetic Logical Unit，ALU)和控制器(Control Unit，CU)组成，是决定计算机性能的核心部件。

(2) 存储器

存储器(Storage)是用来存放程序和数据(Data)的装置,它是计算机中存储信息(Information)的部件。通常用千字节(Kilobyte,KB)、兆字节(Megabyte,MB)或吉字节(Gigabyte,GB)来衡量存储空间的大小。存储器一般分为主存储器(内存,Memory)和辅助存储器(外存)。

计算机把正在运行的程序和一些临时的或少量的数据调入内存,如 Windows 操作系统、应用软件、游戏软件等。通常将要永久保存的程序、大量的数据存储在外存上。

计算机系统中,CPU 能够访问的存储空间也叫寻址空间。所谓寻址,就是 CPU 搜索指令或数据在内存中的地址。

可以把内存简单看作是一个一维的字节空间排列,如图 1-2 所示。标识每个字节内存单元的数字地址(Address),代表其在内存的特定位置,叫内存地址。一般内存中第一个字节的地址为 0,第二个字节的地址为 1,……,依次类推。内存地址通常以十六进制数字表示。

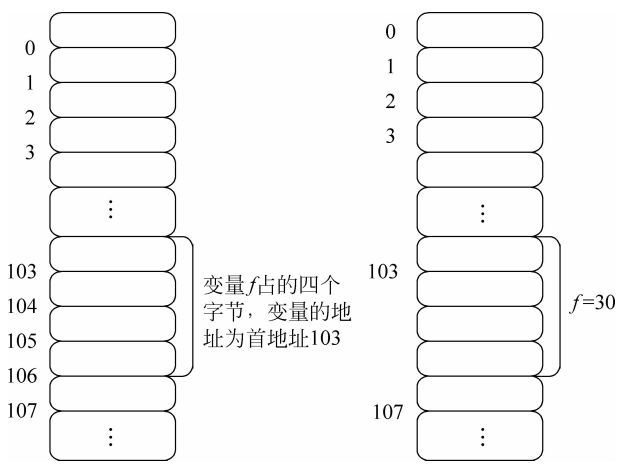


图 1-2 内存编址图

通过内存地址可以随时访问内存的任何位置。在程序设计语言中,执行程序和访问数据就是访问内存单元中的数据。根据需要,多个字节可以组合起来构成更大的内存单元,存放简单变量、数组和结构体等数据。例如,C 语言中一个整型数据可占 4 个连续的字节。

(3) I/O 设备(输入输出设备)

输入设备是计算机从外部获取信息的设备。输入设备把输入的信息转换成计算机可以理解、识别和接收的信号,再通过接口设备把这些信号送到存储器中。微型计算机中,常见的输入设备有键盘、鼠标、扫描仪、摄像机等。输出设备是向用户传递处理结果的设备。它能把计算机产生的结果转换成用户习惯接受的字符、图形、图像、表格或声音等信息形式。常见的输出设备有显示器、打印机、绘图仪等。

2. 计算机软件系统

计算机软件系统是指挥计算机工作的程序与运行时所需要的数据,以及与这些程序和数据有关的文字说明和图表资料,是计算机上可运行的全部程序(Program)的总和。

软件系统分为系统软件(System Software)和应用软件(Application Software)。系统

软件负责整个计算机系统资源的管理、调度、监视和服务，通常包含操作系统(Operating System)、语言处理程序、程序设计语言、数据库管理系统(Database Management System)、支持软件等。应用软件指各个不同领域的用户为各自的需要而开发的各种应用程序。

3. 计算机语言(Computer Language)

程序设计语言是用户编写应用程序使用的语言。程序设计语言分为机器语言(Machine Language)、汇编语言(Assembly Language)和高级语言(High-level Language)。

机器语言可直接被计算机识别并执行，是低级语言，依赖于硬件系统，直接用0/1二进制代码表示指令序列，因此编程困难，程序可读性差，没有可移植性。

汇编语言是用有意义的符号作为编程用的语言，其中使用了很多英语单词的缩写词，实际上是一种符号语言。汇编语言的语句和机器指令一一对应，仍是面向机器的语言。

高级语言一般与人们所习惯的自然语言、数学语言相近，具有使用方便、通用性强等优点。高级语言经历了从早期语言到结构化程序设计语言，从面向过程到非过程化语言的过程。

第一个高级语言FORTRAN诞生于1954年。1969年提出结构化程序设计方法。1970年，第一个结构化程序设计语言——Pascal语言出现，标志着结构化程序设计时期的开始。结构化程序设计技术进一步提高了语言的层次。在此之后，包括BASIC、COBOL、C和Java在内的上百种高级程序设计语言出现并发展、完善。这些高级语言一般都是为某一类应用开发的，适用于不同的应用领域，具有某种特色或侧重点，这就造成了它们之间或多或少会存在一些差异。不过这种差异并不是根本的，因为从本质上说，所有高级程序设计语言都是相通的。因此掌握一门经典语言之后，再学习其他高级语言是非常容易的。20世纪80年代初，出现了面向对象(Object-Oriented)程序设计，C++、VB、Delphi是典型代表。

高级语言的可读性强，可靠性好，利于维护，大大提高了程序设计效率。高级语言的下一个发展目标是面向应用。

1.2 C语言简介

C语言是一种面向过程的程序设计语言，兼有高级语言和汇编语言的优点。常用的C语言集成开发环境(Integrated Development Environment，IDE)有Microsoft Visual C++、Borland C++、Microsoft C、Turbo C等。本书所有程序都在VC++ 6.0上调试通过。

1.2.1 C语言的发展历史

1963年，剑桥大学将ALGOL 60语言发展成CPL(Combined Programming Language)语言。1967年，剑桥大学的马丁·理查德(Matin Richards)对CPL语言进行了简化，于是产生了BCPL语言。1970年，UNIX的研制者丹尼斯·里奇(Dennis Ritchie)和肯·汤普森(Ken Thompson)在B语言的基础上发展和完善了C语言。C语言广泛应用于UNIX、MS-DOS、Microsoft Windows及Linux等不同的操作系统。在C语言基础上发展起来的有支持多种程序设计风格的C++语言、Java、JavaScript和微软的C#等。

(1) K&R C 与 ANSI C。1978 年,里奇(Ritchie)和 Bell 实验室的程序专家柯奈汉(Kernighan)合写了著名的 *The C Programming Language*,将 C 语言推向全世界。由这本书定义的 C 语言后来被人们称作 K&R C。

K&R C 支持最基本的 C 语言部分,作为 C 语言的最低要求仍然要编程人员掌握。

(2) ANSI C。随着 C 语言的广泛使用,出现了许多新问题。1988 年 10 月颁布了 ANSI 标准 X3.159—1989,也就是后来人们所说的 ANSI C 标准。这个标准定义的 C 语言称作 ANSI C。

(3) C99。在 ANSI 标准化后,WG14 小组继续致力于改进 C 语言。1999 年很快推出新标准 ISO 9899:1999,被 ANSI 于 2000 年 3 月采用,即 C99。

(4) 目前最流行的 C 语言。目前最流行的 C 语言有 Microsoft C、Borland Turbo C、AT&T C 等。这些 C 语言版本不仅实现了 ANSI C 标准,而且在此基础上各自作了一些扩充,使之更加方便、完美。

(5) 面向对象的程序设计语言。在 C 的基础上,1983 年贝尔实验室的本贾尼·斯特劳斯特卢普(Bjarne Stroustrup)又推出了 C++。C++进一步扩充和完善了 C 语言,成为一种面向对象的程序设计语言。C++目前流行的最新版本是 Borland C++、Symantec C++ 和 Microsoft Visual C++。

(6) C 和 C++。C 是 C++的基础,C++和 C 语言在很多方面兼容。掌握了 C 语言,再进一步学习 C++就能以一种熟悉的语法来学习面向对象的语言,从而达到事半功倍的目的。

1.2.2 C 语言的特点

1. 简洁紧凑、灵活方便

C 语言有 32 个关键字,9 种控制语句,程序书写自由,主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性结合起来。C 语言可以像汇编语言一样对位、字节和地址进行操作,而这三者是计算机最基本的工作单元。

2. 运算符丰富

C 语言共有 34 个运算符,包含的范围很广泛。C 语言把括号、赋值、强制类型转换等都作为运算符处理,从而使 C 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在其他高级语言中难以实现的运算。

3. 数据结构丰富

C 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,能实现各种复杂数据类型的运算,并引入了指针的概念,使程序效率更高。另外,C 语言具有强大的图形功能,支持多种显示器和驱动器,且计算功能、逻辑判断功能强大。

4. C 是结构化程序设计语言

结构化程序设计语言的显著特点是代码及数据的分隔化,即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰,便于使用、维护和调试。C 语

言是以函数形式提供给用户的，这些函数可方便地调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。

5. C语法限制不太严格，程序设计自由度大

6. C语言允许直接访问物理地址，可以直接对硬件进行操作

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，能够像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元，可以用来写系统软件。

7. C语言程序生成代码质量高，程序执行效率高

8. C语言适用范围大，可移植性好

C语言的一个突出优点是适合多种操作系统和多种机型，如DOS、UNIX。

C语言也有其局限，它的灵活性给编程人员带来自由的同时，可能也埋下一定的风险：指针使得程序的执行过程难以跟踪，简洁使得程序难以阅读，等等。但鉴于其众多的优点，C语言仍然不失为人们首选的编程语言之一。

1.3 计算思维和计算机辅助问题求解过程

计算机科学本质上源自数学思维和工程思维。计算思维是运用计算机科学的基础概念求解问题，设计系统和理解人类行为。它选择合适的方式陈述一个问题，对一个问题的相关方面建模，并用最有效的办法实现问题求解。然而，计算思维远远不只是为计算机编程，它是抽象的多个层次上的思维，与“读写能力”一样，是人类的基本思维方式。有了计算机，人类就能用自身智慧解决那些计算机时代之前不敢尝试的问题。信息科技不仅是一种高科技工具、一种辅助性学科，而且是21世纪经济社会必需的普适资源和增值资产。

问题解决方案可能不仅涉及该问题的抽象思考，还会涉及问题环境中的实验性学习。解决问题都有一定的求解方法，使用计算机同样有一定的问题求解方法。计算机辅助问题求解过程(Computer-Assisted Problem Solving Process)是工程与科学课程中的一个关键部分。

下面介绍计算机辅助问题求解的一般方法。

【例1-1】 计算平面上两点间的距离。

分析如下。

(1) 陈述问题：当一个问题提出需要用软件实现时，明晰且精确的问题陈述可以避免产生任何误解。“陈述问题”是为了清楚“需求分析”。本例问题描述清楚。

(2) 需求分析：主要是确定软件程序需要实现的目标，确定软件处理的数据或信息，建立问题域数据结构，进行程序设计可行性分析。

这一步需要仔细描述为解决问题而提供的信息，指出最后所需要的结果或结论。这两项代表问题的输入与输出(I/O)信息，如图1-3所示。这时，问题被“抽象”为一个黑盒子，并未定义决定输出信息的“处理步骤”。本问题中，定义解决问题的输入输出信息如图1-4

所示。

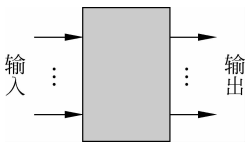


图 1-3 问题的输入输出

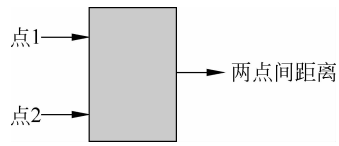


图 1-4 “计算平面上两点间的距离”的输入输出

(3) 数学建模或处理流程示例：为了准确理解并处理问题，有必要利用问题域的一个简单数据集手动模拟解答问题，从中找出问题解答的细节或过程。

设点 p_1、p_2 的坐标为 $p_1=(1,5)$，$p_2=(4,7)$。则计算两点间距离就是一个直角三角形的斜边长。使用毕达哥拉斯定理计算距离：

$$\text{distance}=\sqrt{\text{side}_1^2+\text{side}_2^2}=\sqrt{(4-1)^2+(7-5)^2}=\sqrt{13}=3.61$$

从而得该问题的数学模型为 $p_1=(a,b)$，$p_2=(c,d)$，$\text{distance}=\sqrt{(c-a)^2+(d-b)^2}$。

这一步正是找出解决处理问题的一般方法或步骤。

(4) 用计算思维确定算法：在前面分析基础上，进一步写出解决问题的详细算法步骤。本例算法用自然语言描述如下。

① 给定两个点坐标，即给两个平面点坐标赋值。

② 计算由此两点构成的直角三角形的两直角边长度。

③ 根据两直角边长求斜边长。

④ 输出斜边长，即两点间的距离。

也可把(3)、(4)步骤称为“系统设计”。

(5) 编码：根据算法设计(或系统设计)的结果，用程序设计语言编程实现所定义的处理过程，最终实现软件系统的功能。本例用 C 语言编写的程序代码如下。

```
/* program ch1-1.c */                    /* 注释 */
#include <stdio.h>                       /* 头文件包含,标准输入输出函数库 */
#include <math.h>                        /* 头文件包含,数学函数库 */
void main(void)                          /* 函数首部,此函数为主函数 */
{   double x1=1,y1=5,x2=4,y2=7,side1,side2,dist; /* 定义两点和边变量并赋值 */
    side1=x2-x1,side2=y2-y1;                    /* 计算两直角边长 */
    dist=sqrt(side1*side1+side2*side2);         /* 调用平方根函数,计算两点间距离 */
    printf("两点间的距离是%5.2f",dist);          /* 输出结果 */
}
```

(6) 程序测试：问题求解的最后一步是测试结果是否正确。应该利用问题域的数据集多次测试，确保答案也适用于其他有效数据集。

计算机辅助问题求解过程(Computer-Assisted Problem Solving Process)一般有 6 个步骤。

(1) 明晰地陈述问题。

(2) 准确的需求分析。

(3) 对一个简单数据集手动地建立数学建模或进行处理流程示例。

(4) 用计算机思维确定算法。

(5) 用计算机语言实现编码。

(6) 利用多种数据上机测试程序答案。

程序设计是给出求解特定问题程序的过程,该过程应当包括分析、设计、编码、测试、排错等不同阶段。应该用这种思维和方法解决所遇到的所有问题。解决问题的6个步骤中,算法设计是非常重要的一步,是程序设计的核心,是程序的灵魂。它将最终决定解决该问题的方法。

1.4 算法及其表示

1.4.1 算法的基本概念

分析问题的处理步骤是算法设计,将处理步骤描述出来就是算法描述。

1. 算法(Algorithm)的概念

算法一词源于算术(Algorism),即算术方法,是指一个由已知推求未知的运算过程。我国古代数学家刘徽成功地设计了计算圆周率π的算法——“割圆术”,而秦九韶给出的计算多项式值的算法可以大大减少计算次数。传统算法可以理解为有基本运算及规定的运算顺序所构成的完整的解题步骤。后来,人们把它推广到一般,把进行某一工作的方法和步骤称为算法。广义地说,算法就是做某一件事的步骤或程序。

但由于计算工具的限制,许多良好的算法应用受到了限制。而计算机无可比拟的运算速度和惊人的存储量使许多以前无法完成的复杂计算算法成为可能,并且使计算精度大大提高。可见,算法的实现是与计算工具相关的,程序设计中需要设计计算机能执行的算法。

计算机算法以一步接一步的方式来详细描述计算机如何将输入信息转化为所要求的输出的过程,或者说,算法是对计算机上执行的计算过程的具体描述,是一系列解决问题的清晰指令,代表着用系统的方法描述解决问题的策略机制。也就是说,能够对一定规范的输入,在有限时间内获得所要求的输出。如果一个算法有缺陷,或不适合某个问题,执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一个算法的优劣可以用空间复杂度与时间复杂度来衡量。

现代意义上的“算法”通常是指可以用计算机来解决的某一类问题的程序或步骤,这个程序或步骤是按照要求设计好有限的确切的计算或处理序列,并且这样的步骤和序列可以解决一类问题。计算机算法可分为两大类,一类是数值运算算法,另一类是非数值运算算法。数值运算算法主要是求数值解,如求方程的解、求函数的定积分等;非数值运算的范围则非常广泛,如人事管理、图书检索等。

2. 算法分析

算法分析的目的在于选择合适算法和改进算法。算法分析的任务是对设计出的每一个具体的算法,利用数学工具,讨论各种复杂度,以探讨某种具体算法适用于哪类问题,或某类问题宜采用哪种算法。算法质量的优劣将影响到算法乃至程序的效率。算法优劣可用空间复杂度与时间复杂度来衡量。

算法的时间复杂度是指算法需要消耗的时间资源。一般来说,计算机算法是问题规模 n 的函数 $f(n)$,算法的时间复杂度也因此记做:

$$T(n)=O(f(n))$$

因此,问题的规模 n 越大,算法执行的时间的增长率与 $f(n)$ 的增长率正相关,称作渐进时间复杂度(Asymptotic Time Complexity)。

算法的空间复杂度是指算法需要消耗的空间资源。其计算和表示方法与时间复杂度类似,一般都用复杂度的渐进性来表示。同时间复杂度相比,空间复杂度的分析要简单得多。

算法可以广义地分为三类。

(1) 有限的,确定性算法。这类算法在有限的一段时间内终止,可能要花很长时间来执行指定的任务,但仍将在一定的时间内终止。这类算法得出的结果常取决于输入值。

(2) 有限的,非确定算法。这类算法在有限的时间内终止。然而,对于一个(或一些)给定的数值,算法的结果并不是唯一的或确定的。

(3) 无限的算法。由于没有定义终止定义条件,或定义的条件无法由输入的数据满足而不终止运行的算法。通常,无限算法的产生是由于有不能确定的终止条件。

3. 算法的特征

一个算法必须具备以下性质——正确性、可行性、确定性、有穷性、有效性。

算法首先必须是"正确的",即对于任意的一组输入,包括合理的输入与不合理的输入,总能得到预期的输出。如果一个算法只是对合理的输入才能得到预期的输出,而在异常情况下却无法预料输出的结果,那么它就不是正确的。

"可行性"指算法中的每一步都能实现。例如,不能出现负数开平方、零作除数、负数取对数等。在设计算法时应避免出现类似的情况。

"确定性"指算法的每一步骤必须有明确的含义,不允许有模糊的解释或多义性,如"如果 $x>0$,就将 x 加上一个正数"就是错误的,必须指出这个正数是什么。又如"如果 x 比 y 大很多,就令 $y=x$"也是错误的,因为"大很多"是个模糊的概念。再如"如果 $x>0$ 就将 x 加上 1 或 2"也是错误的,这里出现了多义性。算法的确定性保证了算法是"机械"的,因此可交由计算机执行。

"有穷性"指算法能在有限步内结束。数学中某些求值问题,往往是用求无穷多项的和得到的,这时就只能用有限多项的和来近似代替。例如,可以证明

$$\frac{\pi^2}{6}=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^3}+\cdots+\frac{1}{n^2}+\cdots$$

式子右边有无穷多项,设计算法时,只能取前面有限多项(如取 $n=100$)求 $\frac{\pi^2}{6}$ 的近似值。

"有效性"是指算法执行的结果能达到预期的目的,即每个算法对满足某种条件的问题应能得到正确的结果。对一个计算机算法而言,算法有效性还包括 0 个或多个输入,以描述问题的初始情况。所谓 0 个输入是指算法不需要初始条件。一个算法有一个或多个输出,以反映对输入数据加工后的结果,没有输出的算法是毫无意义的。

如求方程 $f(x)=0$ 的根的二分法,当 $f(x)$ 的函数图像在 $[a,b]$ 区间为不间断的连续曲

线，且 $f(a)$与 $f(b)$异号时必能求得 $f(x)=0$ 在$[a,b]$内的一个实根。再如，判别一个大于 1 的正整数是不是素数的算法一定要能给出这数是不是素数的确切判断，等等。算法的有效性实际上还包括了合理执行时间的含义，从理论上讲，应考虑算法的时间复杂性和空间复杂性，对于计算次数过多(因而占机时数不可容忍)、占用内存单元过多的算法都是不可取的。

除以上 5 性之外，一个良好的算法还应该具备下面一些特征。

(1) 可读性：算法的书写、命名等应便于阅读和交流，杜绝晦涩难懂的算法。

(2) 健壮性：算法对非法数据能做出正确的反应，并进行适当的处理。

(3) 普遍性：指算法能解决一类问题而不仅仅是某个问题。例如，在计算机上求一元二次方程根的算法，如果将方程系数固定，则只能解特定的一个方程；而如果方程的系数在执行时临时输入，则可解所有一元二次方程。当然普遍性是相对的，例如上面提到的 $f(x)=0$ 的二分法，就不能解所有的一元方程。例如，$f(x)=(x-8)^2=0$，虽然它在$[6,10]$内有根 8，但 $f(6)f(10)>0$，不满足算法须满足的条件。

1.4.2 算法的表示

算法可采用任何形式的语言或符号来描述，通常有自然语言、伪代码、传统流程图、N-S 图、PAD 图等多种方法。重点介绍自然语言和 N-S 图表示法。

1. 自然语言

自然语言(Natural Language)是指人们日常生活中所使用的语言。自然语言可以是中文、英文、数学表达式等，可以使用自然语言来表示算法。

【例 1-2】 计算某学生两门课程成绩的和。

分析如下。

(1) 陈述问题和需求分析：计算两个成绩的和，定义成绩为整数，即计算两个整数的和。输入两个整数，输出两数之和。

(2) 确定算法：用自然语言描述算法可表示如下。

① 用三个变量表示两个操作数和两数之和：num1、num2、sum。

② 分别给两个操作数 num1 和 num2 赋值。

③ 计算和值 sum=num1+num2。

④ 把 sum 的结果输出到屏幕。

这是一个顺序结构算法，即计算机按照算法步骤顺序执行。自然语言表示算法清楚易懂，但易冗长，有时会产生二义性。所谓二义性，是指一种语言语法的不完善说明，应避免出现。所以除了简单问题外，一般不采用自然语言表示算法。

2. 传统流程图

流程图(Flow Charts)是流经一个系统的信息流、观点流或部件流的图形代表。程序设计中，程序流程图是使用图形表示对解决问题的方法、思路或算法的一种描述。流程图采用简单规范的符号，画法简单；结构清晰，逻辑性强；便于描述，容易理解。传统流程图在汇编语言和早期的 BASIC 语言环境中得到应用，由于其中的转向过于任意，带来了许多副作用，现已趋向消亡。

1) 传统流程图

传统流程图是用各种几何图形、流程线及文字说明来描述算法过程的框图。ANSI(American National Standards Institute,美国国家标准协会)规定一些常用传统流程图符号代表各种性质不同的操作,图框中的文字和符号表示操作的内容。一般用椭圆表示“开始”与“结束”,矩形表示处理计算,菱形表示问题判断或判定,箭头代表工作流方向,输入输出为平行四边形,其含义如图 1-5 所示。

图 1-5 传统流程图的常用符号

【例 1-3】 求任意两数中较大的数。

分析如下。

① 陈述问题和需求分析:该问题明确,输入任意两个数,输出两数中较大的数。

② 确定算法:根据两个数的比较结果决定程序的输出结果。用传统流程图描述算法如图 1-6 所示。这是一个选择结构算法,按照条件比较结果,选择某个分支执行。

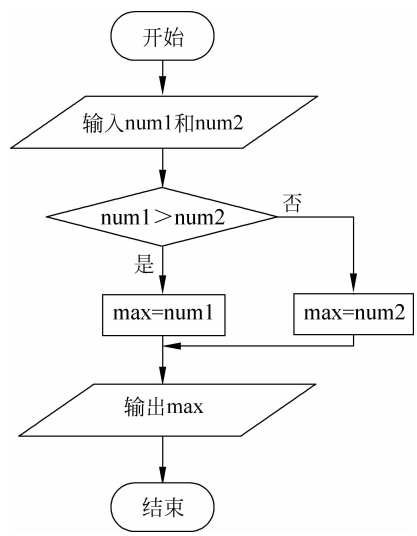


图 1-6 例 1-3 的流程图

传统流程图的优点是方便、直观、灵活、流程清晰、无“歧义性”。但占用面积大,且有一个缺点,它允许流程线指向任意一个框。对程序流程不加任何限制,对大程序而言就会导致算法的逻辑难以让人理解。这种描述方法的可读性、可靠性及可维护性差。

2) 盒图

盒图是取代传统流程图的一种描述方式,它是由 Ike Nassi 和 Ben Shneiderman 在 1973 年提出的,也称 NS 图。NS 图完全去掉了流程线,算法的每一步都用一个矩形框来描述,把一个个矩形框按执行的次序连接起来就是一个完整的算法描述。NS 图以结构化程序设计(Structure Programming,SP)方法为基础,仅含有图 1-7 所示的 5 种基本成分,它们分别表示 SP 方法的几种标准控制结构。

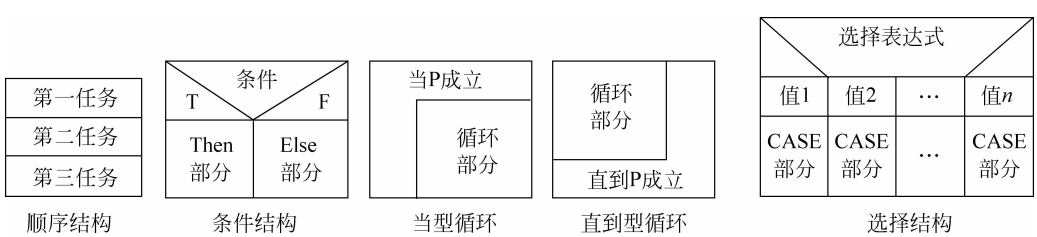


图 1-7 NS 图的几种标准控制结构

NS 图中，每个“处理步骤”用一个盒子表示。所谓“处理步骤”可以是语句或语句序列。需要时，盒子中还可以嵌套另一个盒子，嵌套深度一般没有限制，只要整张图在一页纸上能容纳得下。由于只能从上边进入盒子然后从下边走出，除此之外没有其他的入口和出口，所以，NS 图限制了随意的控制转移，保证了程序的良好结构。

NS 图除了表示几种标准结构的符号之外，不再提供其他描述手段，所以它强制设计人员按 SP 方法进行思考并描述设计方案，这有效地保证了设计质量，从而也保证了程序质量。第二，NS 图形象直观，具有良好的可见度。例如循环的范围、条件语句的范围都是一目了然的，所以容易理解设计意图，为编程、复查、选择测试用例、维护都带来了方便。第三，NS 图简单，易学易用，可用于软件教育和其他方面。但 NS 图手工修改比较麻烦。

用 NS 图作为详细设计的描述手段时，常用两个盒子：数据盒和过程盒。前者描述有关的数据，包括全程数据、局部数据和模块界面上的参数等，后者描述执行过程，如图 1-8 所示。

【例 1-4】 求某班 30 个学生某门课程成绩的平均值。

分析如下。

① 陈述问题和需求分析：输入 30 个学生成绩数据，输出它们的平均值。

② 确定算法：首先，30 个学生成绩数据需要求和，然后用和值除以 30 取平均值。

算法设计的关键是如何求“30 个学生成绩数据和”。“求多个数和值”的方法有多种。用计算机求多个数据和值的最好算法是用循环结构。使用 NS 流程图描述解决该问题的循环结构算法如图 1-9 所示。

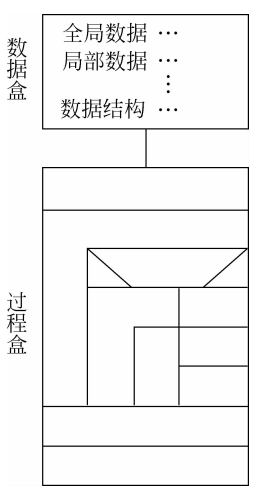


图 1-8　数据盒和过程盒

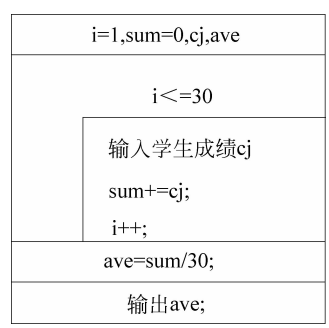


图 1-9　例 1-4 的图

NS 图形象直观，具有良好的可见度。例如循环的范围、条件语句的范围都是一目了然的，所以容易理解设计意图，为编程、复查、选择测试用例、维护都带来了方便。但是当问题很复杂时，NS 图可能很大，手工修改比较麻烦，这是有些人不用它的主要原因。

虽然算法与计算机程序密切相关，但二者也存在区别：计算机程序是算法的一个实例，是将算法通过某种计算机语言表达出来的具体形式；同样的任务可能用不同的算法来完成；同一个算法可以用任何一种计算机语言来表达。

1.5 结构化程序设计

1.5.1 程序设计方法

计算机程序设计方法伴随着计算机硬件技术的提高而不断发展。硬件环境对软件设计既有严重的制约作用,也有积极的推动作用。程序设计是一种技术,需要相应的理论、技术、方法和工具来支持。程序设计技术的发展主要经过了最初的经验式程序设计,到功能模块分离的结构化程序设计(Structured Programming),发展为通过继承来实现比较完善的代码重用功能的面向对象的程序设计(Object-Oriented Programming),和在源程序级别重用的组件对象模型(COM / CORBA)程序设计等。

经验式程序设计是一种工艺性、技艺性的方法,没有固定的方法和技术,因人而异。因此,程序设计过程和质量取决于程序设计者本身的个人经验。程序设计员好比是一个艺人、工匠。经验式程序设计导致程序质量差,维护(修改,扩充,移植)困难。

20 世纪 70 年代初提出了“结构化程序设计”的思想和方法。这种方法的基本原则是“分解原则”:把一个复杂的程序功能划分成若干子功能,使每一个子功能能独立设计,并且使程序的复杂性得到简化。如果子功能仍然比较复杂,则再对其进行进一步划分成更小的子功能。每一个子功能便称为一个“功能模块”。

程序设计方法学的目标是能设计出可靠、易读而且代价合理的程序。程序设计方法学的基本内容是:结构程序设计,程序理论在程序设计技术中的应用,以及规格说明和变换技术。程序设计方法学也与软件工程关系密切。方法学对软件的研制和维护起指导作用。软件工程要求程序设计规范化,建立新的原则和技术。

1.5.2 结构化程序设计

结构化程序设计(Structured Programming,SP)方法是与结构化分析 SA 和结构化设计 SD 方法相衔接的。在程序的规模和复杂性越来越大的今天,程序的不规范、可读性差和难于修改形成了人们常说的“软件危机”。1965 年,荷兰学者迪克斯特拉(E. W. dijkstra)提出了“结构化程序设计”的思想,这是软件发展的一个重要的里程碑。

结构化程序设计的基本原则以模块功能和处理过程的详细设计为主。它的主要观点是采用自顶向下、逐步求精的程序设计方法,使用顺序、选择、重复三种基本控制结构构造程序。

用三种基本结构组成的程序必然是结构化的程序,这种程序便于编写、阅读、修改和维护,减少了程序出错的机会,提高了程序的可靠性,保证了程序的质量。结构化程序设计强调程序设计风格和程序结构的规范化,提倡清晰的结构。结构化程序设计方法的基本思路是,把一个复杂问题的求解过程分阶段进行,每个阶段处理的问题都控制在人们容易理解和处理的范围内。具体来说,采取以下方法保证得到结构化的程序。

(1) 自顶向下;

(2) 逐步细化;

(3) 模块化设计；

(4) 结构化编码。

结构化程序设计要求人们尽量避免使用 goto 语句，在不得不使用时应十分谨慎，不能跳得很远，只限于一个结构内部跳，不能从一个结构跳到另一个结构。

1. 模块化程序设计方法

一个系统的设计，由多个具有单一功能、易于理解的多个模块组成。这就是模块化程序设计方法。按什么原则划分模块？如何组织各模块之间的联系？有以下几条规则。

1) 按操作功能划分模块

要求各模块功能尽量单一，各模块之间联系尽量地少，所编程序可读性和可理解性好，当修改某一功能时只涉及一个模块。

2) 按层次结构组织模块

上层模块只实现对下层模块的调用，因此，只指出“做什么”；下层(即最低层)模块才实现“如何做”，对“如何做”作精确描述，按具体任务进行算法分析、算法实现，组成独立模块，给上层调用。

2. 逐步细化的设计方法

逐步细化的设计方法是“自顶向下，逐步细化”的基本方法。采用此方法可以将一个复杂的问题分解成多层次的模块结构。直到把一个模块功能逐步细化为一系列的处理步骤，甚至于高级语言的一个语句或一条指令。

采用逐步细化的优点如下。

① 符合人们解决复杂问题的普遍规律，可以显著提高程序设计的效率。

② 先全局后局部、先整体后细节、先抽象后具体的逐步细化过程设计出来的程序具有清晰的层次结构，容易阅读和理解。

3. 结构化程序的三种基本结构

顺序结构、选择(分支)结构及循环结构称为程序设计的基本结构，由它们组成的程序称为结构化程序。一个结构化程序具有易读性好、可靠性高、便于维护和易于移植等优点。

任何一个结构化程序只能由以下三种基本结构组成。

1) 顺序结构(Sequence Structure)

顺序结构是最简单、最基本的程序结构。在这种结构中，程序的各块是按其书写顺序依次执行的。例 1-1 和例 1-2 的算法是顺序结构算法。

2) 选择结构(Select Structure)

选择结构也称为分支结构。选择结构通过测试一个条件，依据条件的结果选择执行程序的某个路径，而不是严格按照语句出现的顺序执行。例 1-3 的算法是选择结构算法设计。

3) 循环结构(Loop Structure)

循环结构允许在测试条件为真时重复执行某一组语句，可以减少源程序重复书写的工作量。循环结构用来描述被重复执行的程序段，这种结构充分发挥了计算机的特长。例 1-4 是循环结构算法设计。

NS流程图强制设计人员按SP方法进行思考并描述设计方案,有效地保证了设计质量,从而也保证了程序质量。

【例 1-5】 用结构化程序设计方法求解一元二次方程 $ax^2+bx+c=0$ 的根。

分析如下。

(1) 问题陈述和需求分析:方程求根计算抽象为一个"模块",对该模块输入该方程的系数参数 a、b、c 即输入一元二次方程,该模块输出方程的根,如图 1-10 所示。这是对问题求解的整体描述。

(2) 数学模型和问题处理过程如下。

① 进一步细化"计算处理"模块,求出问题解决方案。

步骤 1:对于输入的参数,判断能否构成一元二次方程,若 a 为 0,输出出错信息。

步骤 2:利用数学中的求根公式求解一元二次方程的根。

细化求精后,出现了分支选择结构。用 NS 流程图描述该模块,如图 1-11 所示。

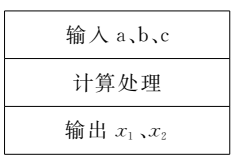


图 1-10 问题模块整体描述

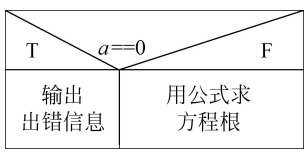


图 1-11 选择结构模块流程图

② 细化子步骤 2 中的"用公式求方程根",具体步骤如下。

步骤 2.1:确定数学中的求根公式为(建立数学模型):

$x_1=\dfrac{-b+\sqrt{\delta}}{2a}$,$x_2=\dfrac{-b-\sqrt{\delta}}{2a}$,用 NS 流程图描述如图 1-12 所示。

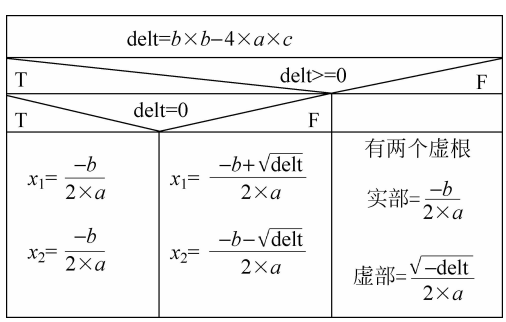


图 1-12 用公式求方程根

③ 对求根公式再继续分析并细化。中间数据 δ 设为变量 delt,需进一步计算并判断。

步骤 2.1.1:delt=$b\times b-4\times a\times c$。

步骤 2.1.2:若 delt$\geqslant$0,且 delt=0,有两个相等实根,否则有两个不等实根;否则,有两个虚根。用 NS 流程图描述算法步骤,如图 1-12 所示。

编写结构化程序时,需要对两个方面进行描述。

(1) 对数据的描述(数据流):指定数据的类型和数据的结构。不同的语言对数据的定义不同,本书中将学习 C 语言的数据类型和数据结构。

(2) 对操作的描述(控制流)：要指定操作的步骤，即先执行什么后执行什么，这就是算法。算法具有通用性，它脱离于语言之外，是程序设计的灵魂。

1.6 C 程序基本结构

1.6.1 简单 C 程序举例

暂时忽略 C 语言的词法和语法细节，阅读几个简单的 C 语言程序示例，了解 C 语言程序的基本组成元素和结构，感受用 C 语言进行程序设计的方法和思想。

【例 1-6】 在屏幕上显示如图 1-13 所示的输出结果。

分析如下。

(1) 问题陈述：在屏幕上按图显示“欢迎学习应用型 C 语言程序设计”。

图 1-13 例 1-6 的输出结果

(2) 需求分析：没有数据输入，只需要在屏幕上按位置布置输出信息。

(3) 处理流程：解决该问题时，顺序逐行输出所需要的信息。

(4) 确定算法：采用顺序结构程序设计方法，应用 printf 函数来实现，算法用自然语言描述如下：

输出第 1 行信息；

输出第 2 行信息；

……(逐行输出信息)

(5) 根据上面所设计的算法，使用 C 语言编写的程序代码如下。

```
/* program ch1-6.c */                                  /* 注释语句 */
#include <stdio.h>                                     /* 包含头文件的命令行 */
void main(void)                                        /* 函数首部,main 函数 */
{   /* 函数体开始 */
    printf("..........................\n");            /* 输出第 1 行 */
    printf(".欢迎学习应用型 C 语言程序设计.\n");      /* 输出第 2 行 */
    printf("..........................\n");            /* 输出第 3 行 */
    return;                                            /* 返回语句 */
}                                                      /* 函数体结束 */
```

程序剖析：

这是一个完整的 C 语言源程序，通过这个 C 源程序，理解如下知识点。

① C 源程序(Source Program)。用 C 语言编写的程序称为 C 语言源程序，C 源程序文件后缀为“.c”。每个 C 源程序文件由可读的 C 程序设计语言写成。本源程序存储的文件名为 ch1-6.c。

② C 函数(Function)。函数是构成 C 源程序的基本单位。一个源程序文件由一个或多个函数组成。函数是完成特定功能的程序段，由函数首部(Head)和函数体(Body)两部分组成。函数首部确定了函数的函数名、形参、返回值类型，函数体中包括函数所执行的步骤，这些步骤称为语句，语句构成函数体。函数体部分由花括号引起来。本例函数名为 main，

函数体中有 4 个语句。

③ 主函数(The Main Function)。C 程序也称为函数式程序。本程序只有一个 main()函数,也称为主函数。任何一个 C 程序中有且只能有一个 main 函数,C 程序总是从主函数开始执行,并且结束于主函数。一个 C 程序可以有一个或多个 C 源程序文件。

④ 语句(Statement)。C 语言的语句以分号(;)作结束标志。printf("...\n");是一个函数调用语句,实现了信息输出功能。

信息或数据的输入输出是程序与用户交互的重要手段。C 语言不提供输入输出语句,通过调用 C 提供的标准函数实现输入和输出。如 printf 函数实现数据信息的输出。printf 函数是系统提供的标准库函数,能够按照用户指定的格式输出各种信息。本例是 printf 函数的最简单应用形式——原样输出双引号中的所有字符。双引号中的普通字符,程序运行时原样输出。其中"\n"是一个转义(即转变原来含义)字符,它的功能是输出一个回车换行符,起换行的作用。

⑤ 头文件包含。#include <stdio.h>是头文件包含命令。stdio.h 是系统提供的标准输入输出头文件。系统有不同的头文件,提供不同的标准函数声明或数据定义,供编程者使用。

printf 函数是标准库函数,该函数的原型说明包含在"stdio.h"头文件中,要在程序中使用这个函数,必须在程序的开头写上包含该头文件的命令行。".h"后缀文件叫头文件。

⑥ 注释(Comment)。/*…*/是注释符号,必须成对出现,两者之间的所有字符(可以是多行)均为注释文字,增加了程序的可读性,不作为程序代码运行。

(6) 在 C++环境中运行测试 C 程序。程序编写完成后,为了验证程序编写得是否正确,应该运行程序验证其结果。

C 语言源程序是不能被计算机直接执行的,编写好的 C 语言源程序还需要使用"编译程序"编译成"目标程序",然后将目标程序与系统的函数库和其他目标程序连接起来,才能得到可执行程序。运行一个 C 语言源程序需要经过如下 4 个过程。

① 编辑。打开 Visual C++ 6.0 应用程序,新建 C 源程序文件,在 Visual C++ 6.0 的工作空间输入程序代码后,将源程序文件的扩展名写为 c,保存为"program ch1-6.c"。

② 编译。应用 Visual C++ 6.0 应用程序界面中"组建"菜单中的"编译"命令或使用 Ctrl+F7 快捷键来编译程序,将"program ch1-6.c"源程序转换为"program ch1-6.obj"目标程序。如果程序在编译过程中出现错误,根据提示更改错误,直至编译没有错误为止。

③ 连接。应用 Visual C++ 6.0 应用程序界面中"组建"菜单中的"组建"命令或使用 F7 快捷键来连接程序,将"program ch1-6.obj"目标程序转换为"program ch1-6.exe"可执行程序。如果程序在连接过程中出现错误,根据提示更改错误,直至没有错误为止。

④ 执行。应用 Visual C++ 6.0 应用程序界面中"组建"菜单中的"执行"命令或使用 Ctrl+F5 快捷键来执行程序。如果程序的执行结果和预想的结果不同,还要分析程序的逻辑结构,修改程序后继续重复上述几个步骤,直至程序的运行结果正确为止。

【例 1-7】 求任意两数中较大的数。其算法 NS 流程图如图 1-14 和图 1-15 所示。

```
/* program ch1-7.c */                    /* 文件头注释 */
#include "stdio.h"                       /* 包含头文件的命令行 */
/* 求两数中较大的自定义函数 */            /* 程序段功能注释 */
```

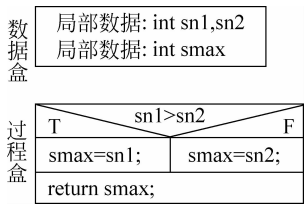


图 1-14 Getmax 函数

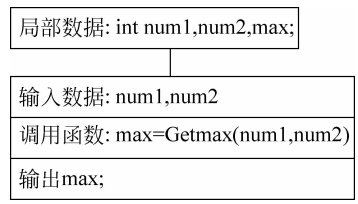


图 1-15 main 函数

```
int Getmax(int sn1,int sn2)                    /* 函数首部,用户自定义函数名 */
{   int smax;                                  /* 变量定义 */
    if(sn1 > sn2) smax = sn1;                  /* 比较选择,分支语句 */
    else smax = sn2;
    return smax;                               /* 返回语句 */
}
void main(void)                                /* 函数首部,main 函数 */
{    int num1,num2,max;                        /* 定义变量 */
     printf("请输入两个成绩");                  /* 提示用户输入的信息 */
     scanf("%d%d",&num1,&num2);                /* 调用 scanf 函数输入两个数 */
     max = Getmax(num1,num2);                  /* 调用用户自定义函数 Getmax 求两数中较大数 */
     printf("较大的数为%d\n",max);             /* 输出结果 */
}
```

程序剖析,理解如下知识点:

① C 程序(Program)。C 程序是比 C 源程序更大的概念,一个 C 语言程序由一个或多个源程序文件组成。一个简单的 C 程序可以是只有一个包含主函数的源程序文件构成;而一个较为复杂的 C 程序,可能会由包括主函数在内的多个源文件或源程序构成。

该程序由一个源程序构成,该源程序中有两个函数,一个是主函数 main,另一个是名为 Getmax 的用户自定义函数,该自定义函数有两个形式参数,返回值为 int 型。主函数的名字 main 是系统规定的,用户不能更改,但用户可以编写主函数的功能。用户自定义函数由用户自己设计编写,包括函数名字、函数返回值类型和函数形式参数。

所以 C 语言的一个源程序文件由一个或多个函数组成。一个源程序文件就是一个编译单位。C 程序由一个或多个源程序文件构成,这样分别编写、分别编译、提高调试效率。

② 程序的执行与函数的调用与返回。函数间存在调用和被调用的关系。程序总是从 main 函数开始执行并结束于 main 函数。该程序由 main 函数开始执行,它调用 Getmax 函数,并最终返回 main 函数。

③ 程序中的变量。变量是程序处理的对象之一。程序中的变量有 sn1、sn2、smax、num1、num2、max。变量存放操作数据,在内存占据一定的存储单元,其中 &num1 和 &num2 表示该变量所占存储空间的起始地址,程序运行时输入的两个数据存放在以 &num1 和 &num2 为起始地址的存储单元中。变量名字由不同的字符编写组成。

④ 输入函数。和 printf 一样,scanf 函数也是一个库函数,其原型说明也在 stdio.h 头文件中。C 语言系统提供了大量的库函数供用户使用,附录中列出了各类库函数及其说明。编写程序时,用户可以查阅并使用系统提供的库函数完成一些功能。

⑤ 程序交互界面。程序运行时,为了给用户一个良好的人机交互操作界面,需要设计

一些提示信息。如程序中的语句“printf("请输入两个成绩");”提示用户输入数据。

⑥ 函数可以返回或不返回函数值,“return 表达式;”语句向调用者返回函数结果值。

⑦ 注释的作用。程序中的注释便于用户理解程序。一般对某程序或语句段的注释放在程序段之前,而对某语句的注释放在该语句之后。培养良好的程序书写习惯是程序员的素质之一。

⑧ 运行程序时,需要输入事先准备的测试数据。考虑到各种数据情况,测试数据应该具有代表性和典型性,避免遗漏某些数据情况,使程序存在漏洞。

如第一次运行程序时,先输入 82 和 94,第二次运行再输入 94 和 82,分别测试程序的输出结果。如果两次执行时输出结果都是 94,说明程序正确,否则需要对程序进行修改。

1.6.2 C 程序基本结构

通过例题,可以总结 C 程序的基本结构特点如下,C 程序基本结构一般示意图如图 1-16 所示。

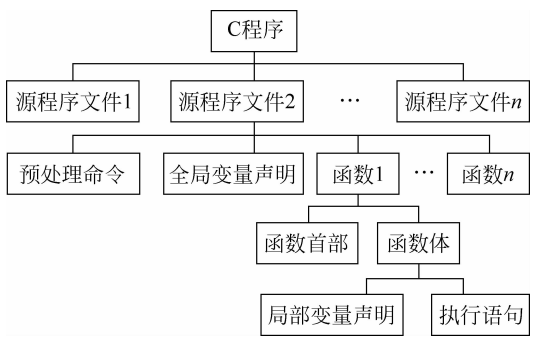


图 1-16 C 程序基本结构一般示意图

(1) C 程序由 C 源程序构成;C 源程序由函数构成,函数是 C 程序的基本单位。每个 C 程序有且仅一个主函数,主函数名为 main。

(2) 函数的定义分为两部分:函数首部和函数体。函数体中一般包括说明语句部分和执行语句部分。函数体中的数据说明语句,必须位于可执行语句之前。换句话说,数据说明语句不能与可执行语句交织在一起。

(3) C 语句以分号(;)作为结束。

(4) C 程序总是从主函数开始执行,并在主函数中结束。主函数在程序中的位置是任意的,即函数的定义次序不影响其引用次序。其他函数总是通过函数调用语句来执行的。因此,C 程序实质上是一系列相互独立的函数的定义,函数之间只存在调用和被调用的关系。

(5) C 语言程序书写格式灵活自由,要求区分大小写字符,一行内可以写几个语句,一个语句也可以分写在多行上。

(6) 用/*…*/作为注释说明程序段或语句的功能,有文件注释、函数注释、行注释和块注释。注释是写给人看的,而不是写给计算机的。程序中的空行不影响程序执行,只是起到分隔程序各部分(如功能段)的作用,可提高程序的可读性。