第 3 章

顺序结构程序设计

Project 3

顺序结构是C程序中最简单、最基本、最常用的一种程序结构。也是进行复杂程序设计的基础。顺序结构的特点是完全按照语句出现的先后顺序依次执行程序。在日常生活中,需要“按部就班、依次进行”处理和操作的问题随处可见。赋值操作和输入/输出操作是顺序结构中最典型的操作。

本章学习目标

1. 掌握结构化程序设计基础知识
2. 掌握输入语句的使用方法
3. 掌握输出语句的使用方法

3.1 结构化程序设计基础

初学者常常会有这样的一种感觉:读别人编写的程序比较容易,自己虽然学了程序设计语言,可编写程序,却不知从何下手。其中一个重要的原因就是没有掌握程序设计的灵魂——算法。所以,多了解、掌握和积累一些计算机常用的算法,养成编写程序前先设计好算法的习惯至关重要。

3.1.1 算法的概念

1. 基本概念

一个程序应包括对数据的描述和对数据处理的描述。

对数据的描述,即数据结构(Data Structure)。数据结构是计算机学科的核心课程之一,有许多专门著作论述,本书不再赘述。

对数据处理的描述,即算法(Algorithm)。算法是为解决一个问题而采取的方法和步骤,是程序的灵魂。为此,著名计算机科学家尼克劳斯·沃思(Niklaus Wirth)提出了一个公式:

程序=数据结构+算法

实际上,一个程序除了以上两个主要因素之外,还应考虑程序设计的方法以及用何种计算机语言来描述。因此,程序还可以这样表示:

程序＝算法＋数据结构＋程序设计方法＋语言工具和环境

所以，在设计一个程序时要综合运用这 4 方面的知识。在这 4 个方面中，算法是灵魂，数据结构是要处理的对象，语言是工具，编程需要采用合适的方法。算法是解决“做什么”和“怎么做”的问题。程序中的操作语句，实际上就是算法的体现。

无论是解决数学问题，还是解决日常生活和工作中的问题，都必须采用一定的方法，并且按照一定的步骤来解决。因此，广义地认为，算法是指为解决一个问题而采取的方法和步骤。

对于不同的问题有不同的算法，而对于同一个问题，也可以有不同的算法。例如，求 1＋2＋3＋4＋5＋6＋7＋8＋9＋10 的和，就有多种算法。

其一，按照数的先后顺序，从左至右一个数一个数地相加，直到加完 10 为止；

其二，将 1 与 9 相加，2 与 8 相加……即原式＝(1＋9)＋(2＋8)＋(3＋7)＋(4＋6)＋10＋5＝5×10＋5＝55。

还有其他算法。显然，在这里第二种算法比第一种算法优越，也就是说，对于同一个问题，不仅有不同的算法，而且这些算法又有优劣之分。有的算法只需很少的步骤，而有些算法则需要较多的步骤。一般地说，应采用简单的和运算步骤少的算法。因此，为了有效地进行解题，不仅需要保证算法正确，还要考虑算法的质量，选择好的算法。

对于计算机而言，考虑的当然只限于计算机算法，即计算机所能执行的算法。例如，1＋2＋3＋4＋5，或者是将 50 名学生的成绩打印出来，并且统计及格学生人数。这是计算机可以做到的。

2. 算法分类

计算机算法可分为两大类，数值运算算法和非数值运算算法。

数值运算算法，如求方程的根，求一个函数的定积分等。其目的是求数值解，它们属于数值运算范围。

非数值运算算法，如图书检索、人事管理、车辆调度管理等，其范围十分广泛。

目前，计算机在非数值运算方面的应用远远超过了在数值运算方面的应用。由于数值运算有现成的模型，可以运用数值分析的方法，因此，对数值运算的算法研究比较深入，算法比较成熟，对各种数值运算都有比较成熟的算法可供选用。人们常常将这些算法汇编成册(写成程序形式)，或将这些程序存放在磁盘上，供用户调用。而非数值运算的种类繁多，要求各异，难以规范化，因此，只能对一些典型的非数值运算算法(如排序算法)作比较深入的研究。其他的非数值运算问题，往往需要用户参考已有的类似算法，重新设计解决特定问题的专门算法。

3.1.2　算法的特点

算法实际上是一种抽象的解题方法，它具有动态性。因此，算法的行为非常重要。作为一个算法，应具有以下特性。

1. 有效性(Effectiveness)

算法的有效性包括两个方面。一是算法中的每一个步骤必须能够实现。例如,在算法中,不允许分母为零的情况;在实数范围内不能求一个负数的平方根等。二是算法执行的结果要能达到预期的结果。通常,针对实际问题设计的算法,人们总是希望能够得到满意的结果。

2. 确定性(Definiteness)

算法的确定性,是指算法中的每一个步骤都必须有明确的定义,不允许有模棱两可的解释,也不允许有多义性。这一特征也反映了算法与数学公式的明显差异。在解决实际问题时,可能会出现这样的情况:针对某种特殊问题,数学公式是正确的,但按此数学公式设计的计算过程可能会使计算机系统无法认知。这是因为根据数学公式设计的计算过程只考虑了正常使用的情况,而当出现异常情况时,该计算过程就不适应了。例如,某计算公式规定,大于 100 的数被认为比 1 大很多,而小于 100 的数不能认为比 1 大很多;并且在正常情况下出现的数或是大于 100,或是小于 100。但指令"输入 X,若 X 比 1 大很多,则输出数字 1,否则输出数字 0"是不确定的。这是因为,在正常的输入情况下,这一指令的执行可以得到正确的结果,但在异常情况下(输入的 X 在 10～100),这一指令执行的结果就不确定了。

3. 有穷性(Finiteness)

算法的有穷性是指算法必须在有限的时间内执行完,即算法必须能在执行有限个步骤之后终止。数学中的无穷级数,在实际计算时只能取有限项,即计算无穷级数的过程只能是有穷的。因此,一个数的无穷级数的表示只是一种计算公式,而根据精度要求确定的计算过程才是有穷的算法。

算法的有穷性还包括合理的执行时间。如果一个算法的执行时间是有穷的,但需要执行千万年,显然这就失去了算法的实用价值。例如,克莱姆(Cramer)法则是求解线性代数方程组的一种数学方法,但不能以此为算法,这是因为,虽然总可以根据克莱姆法则设计出一个计算过程用于计算所有可能出现的行列式,但这样的计算过程所需的时间实际上是不能容忍的。再如,从理论上讲,总可以写出一个正确的弈棋程序,而且这也不是一件很困难的工作。由于在一个棋盘上安排棋子的方式总是有限的,而且,根据一定的规则,在有限次移动棋子之后比赛一定结束。因此,弈棋程序可以考虑计算机每一次可能的移动,它的对手每一次可能的应答,以及计算机对这些移动的可能应答等,直到每个可能的移动停止下来为止。此外,由于计算机可以知道每次移动的结果,因此,总可以选择一种最好的移动方式。但是即使如此,这种弈棋程序还是很难执行,因为所有这些可能移动的次数太多,所要花费的时间不能容忍。由上述两个例子可以看出,虽然许多计算过程是有限的,但仍有可能无实用价值。

4. 有零个或多个输入

输入是指在执行算法时需要从外界所取得的必要的信息。

5. 有一个或多个输出

算法的目的是为了求解,“解”就是输出。但算法的输出不一定就是计算机的打印输出,一个算法得到的结果就是算法的输出。没有输出的算法是没有意义的。

3.1.3 算法的描述

算法的描述语言是表示算法的一种工具,它只面向用户,不能直接作用于计算机,但却很容易转换为计算机上能执行的程序。常用的算法描述有自然语言、流程图、N-S 图、伪代码等。

1. 用自然语言描述

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。用自然语言描述通俗易懂,但文字冗长,容易出现歧义。自然语言表示的含义往往不太严格,要根据上下文才能判断其正确含义。此外,用自然语言描述包含分支和循环的算法,很不方便。所以,自然语言描述一般适用于算法比较简单的情况。

【例 3-1】 求 1×2×3×4×5 的结果。

步骤 1:先求 1×2,得到结果 2。

步骤 2:将步骤 1 得到的结果 2 乘以 3,得到结果 6。

步骤 3:将步骤 2 结果 6 乘以 4,得 24。

步骤 4:将步骤 3 结果 24 乘以 5,得 120。这就是最后的结果。

使用这种算法是可以的,但如果此题要求 1×2×3×4×…×1000,则需要写 999 个步骤,那么,采用这种算法就显得太烦琐,也就是说这种算法是不可取的。如何改进呢?

分析上述算法,就会发现:步骤 2 要使用步骤 1 的结果,步骤 3 要使用步骤 2 的结果,也就是说后一步骤要用到前一步骤的计算结果,这样可以设两个变量,一个变量代表第 1 个乘数,一个变量代表第 2 个乘数,不另设变量存放乘积结果,而直接将每一步骤的积放在第 1 个乘数变量中。设 p 为第 1 个乘数,i 为第 2 个乘数。用循环算法求结果。将此算法可改写如下。

步骤 1:使 p=1。

步骤 2:使 i=2。

步骤 3:使 p×i,乘积仍放在变量 p 中,可表示为 p×i→p。

步骤 4:使 i 的值加 1,即 i+1→i。

步骤 5:如果 i 不大于 5,返回重新执行步骤 3～5;否则算法结束。最后得到 p 的值就是 1×2×3×4×5 的值。不难看出,这个算法比前面列出的算法简练。

如果题目改为 1×5×10×15×20×25。

上述算法只需作很小的改动即可得到它的算法。

步骤 1：1→p。

步骤 2：5→i。

步骤 3：p×i→p。

步骤 4：i+5→i。

步骤 5：若 i≤25，返回步骤 3；否则结束。

可以看出这种方法表示的算法具有通用性、灵活性。从步骤 3～5 组成一个循环，在实现算法时，反复多次地执行步骤 3～5，直到满足条件 i≤25 时，此算法结束，变量 p 的值就是所求的结果。

【例 3-2】 求一个班学生的平均成绩。设 A 等(85 分)15 人，B 等(70 分)24 人，C 等(60 分)8 人，D 等(50 分)3 人。

对于此例，可以采取如下的算法：设 SUM 为总分数，AVER 为人均成绩。

步骤 1：15→A。

步骤 2：24→B。

步骤 3：8→C。

步骤 4：3→D。

步骤 5：85×A+70×B+60×C+50×D→SUM。

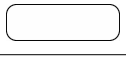
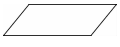
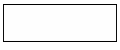
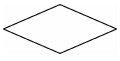
步骤 6：SUM÷(A+B+C+D)→AVER。

AVER 就是所要求的结果。此例中，步骤 1、步骤 2、步骤 3、步骤 4 分别将成绩为 A 等、B 等、C 等、D 等的学生人数存在变量 A、B、C、D 中；步骤 5 计算总分数并将其存放在变量 SUM 中；步骤 6 求出人均成绩并将其存放在变量 AVER 中，算法结束。

2. 用流程图描述

流程图是用一些图框表示各种操作。用图形表示算法，直观形象，易于理解。美国国家标准化协会 ANSI(American National Standards Institute)规定了一些常用的流程图符号，已被世界各国程序工作者普遍采用，如表 3-1 所示。

表 3-1 流程图的基本符号及其含义

图形符号	名 称	含 义
	起止框	表示算法的开始和结束
	输入/输出框	表示输入输出操作
	处理框	表示处理或运算的功能
	判断框	对一个给定的条件进行判断，根据给定的条件是否成立来决定其后的执行操作。它有一个入口两个出口
	流线	表示程序执行的路径，箭头表示流程的方向
	连接符	用于转接到另一页，避免流线交叉，避免流线太长

需要注意的是，流程图仅仅描述了算法，但计算机是无法识别和执行流程图表示的算