COMPUTER

高等院校计算机技术与应用系列规划教材

C语言程序设计实践

◎应 震 卢 敏 编著

ZHEJIANG UNIVERSITY PRESS
浙江大学出版社

C语言程序设计实践

应　震　卢　敏　编著

ZHEJIANG UNIVERSITY PRESS
浙江大学出版社

图书在版编目（CIP）数据

C语言程序设计实践 / 应震，卢敏编著. —杭州：浙江大学出版社，2017.8

ISBN 978-7-308-16939-4

Ⅰ.①C… Ⅱ.①应…②卢… Ⅲ.①C语言－程序设计 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字（2017）第109958号

内容简介

本书是C语言程序设计课程的配套实践指导用书。全书共7章，内容包括：程序设计的有关概念，问题分析和算法设计，编码和程序风格，调试的基本方法，课程相关实验，综合实训案例，在线练习系统使用指南。

本书以应用为背景，面向工程实践和编程能力训练，从解决实际问题出发，循序渐进引出知识点，让读者在不知不觉中逐步掌握C语言程序设计的思想、方法和技巧，以及基本的程序调试方法。

本书适合作为高等学校C语言程序设计课程的配套教材，也可作为学生自学C语言程序设计的辅助教材。

C语言程序设计实践

应 震 卢 敏 编著

责任编辑 吴昌雷
责任校对 陈静毅 刘 郡
封面设计 续设计
出版发行 浙江大学出版社
（杭州市天目山路148号 邮政编码310007）
（网址：http://www.zjupress.com）
排 版 杭州中大图文设计有限公司
印 刷 杭州杭新印务有限公司
开 本 787mm×1092mm 1/16
印 张 5.5
字 数 136千
版 印 次 2017年8月第1版 2017年8月第1次印刷
书 号 ISBN 978-7-308-16939-4
定 价 20.00元

前　言

程序设计是高校理工科各专业的一门重要基础课程，在理工科各专业的教学计划中占有重要地位和起着关键性作用。通过该课程的学习，学生掌握基本的编程能力，以及逻辑思维和抽象能力，为后续课程的学习打好基础。

C语言程序设计是一门实践性很强的课程。学习者以程序设计为主线，以编程应用为驱动，通过案例和问题，以求更好地掌握程序设计的思想和方法。因此，C语言程序设计课程的重点应该是结合编程训练，穿插介绍相关的语言知识，培养学生的语言应用能力。

本书是《C语言程序设计基础》的配套实践指导用书，是编者长期从事计算机基础课程教学，在总结教学经验基础上编写完成的。本书以强化实践教学，提高学生程序设计能力为目的。全书共7章，第1章介绍程序设计的有关概念以及指导程序设计过程的方法等；第2章介绍问题分析需要注意的事项和算法设计使用的技术和工具等；第3章介绍编码和程序风格等；第4章介绍调试的有关方法和技术等；第5章根据C语言程序设计基础课程教学的有关要求，设计了4个实验项目，实验以循序渐进的任务驱动方式，引导学生编写程序，加深学生对结构化程序设计思想和方法的理解；第6章提供综合实训案例，引导学生以项目小组为单位，编写一定规模的综合应用程序，从而培养学生团队协作精神，以及分析和解决工程技术问题的能力；第7章介绍在线练习系统的使用，该系统具有作业在线评测、反抄袭和试题自动生成、自动阅卷等功能，学生可以随时上网进行大量的编程训练。

本书在编写过程中得到了丽水学院工学院的全力支持，同时还得到了丽水学院计算机系所有老师的大力帮助，在此表示衷心的感谢！

本书所配套的电子教案和教学相关资源可以联系编者信箱：zjlszjz@163.com。

因编者水平有限，书中难免会出现不足之处，恳请读者来信批评指正。

编　者

2016年9月

目　录

第 1 章

程序设计概述

在编程语言的学习过程中，除了掌握基本的语法外，更重要的是掌握如何利用编程语言去解决实际的问题。下面对程序设计的有关概念及指导程序设计过程的方法进行简单介绍。

1.1 程序设计概念及过程

程序设计是给出解决特定问题程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。对于 C 语言来说，程序设计过程主要包括分析、设计、编码、调试、排错等不同阶段。各阶段的主要任务如下：

1. 问题分析

首先对于接受的任务要进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法。

2. 算法设计

用适当的方法描述出解题的方法和具体步骤。

3. 编写程序

将算法翻译成计算机程序设计语言，对源程序进行编辑、编译和连接。

4. 运行程序，分析结果

运行可执行程序，得到运行结果。能得到运行结果并不意味着程序正确，还要对结果进行分析，看它是否合理。结果不合理的要对程序进行调试，即通过上机发现和排除程序中的错误。

5. 编写程序文档

许多程序是提供给别人使用的，如同正式的产品应当提供产品说明书一样，正式提供给用户使用的程序必须向用户提供程序文档。其内容应包括：程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据，以及使用注意事项等。另外，从程序的可维护性角度，程序文档还应该包括需求说明书、系统设计说明书等。程序文档的编写应贯穿于整个程序设计过程。

1.2 程序设计方法及软件工具

程序设计作为一个复杂的实践活动，如果没有正确的方法指导，那将是困难和盲目的。另外，在程序设计过程，各种合适的软件工具也是成功的重要保证。

1. 面向过程的结构化程序设计方法

各种编程语言都有自己的特点，进而形成各种指导编程过程的方法。对C语言来说，由于组成程序的单元为函数(也可称之为过程)，所以C语言程序设计时采用的指导方法也称之为面向过程的程序设计方法。这种方法主要遵循“程序＝数据结构＋算法”的思路，把程序理解为由一组数据和一组在数据上处理的过程组成。编程时，先设计数据结构，再围绕数据结构编写其算法过程。另外，在具体的问题分析和算法设计环节，指导的技术也有好几种，其中基于结构化分析和设计的技术是比较主流的一种。这种技术主要以模块化设计和逐步细化的思想作为指导，利用各种工具进行问题分析和算法设计。综上所述，C语言整体指导方法也可称为面向过程的结构化程序设计方法。

面向过程的结构化程序设计方法的主要优点是结构清晰、阅读方便。但由于它把数据和处理数据的过程分离为相互独立的实体，故当数据结构改变时，所有相关的处理过程都要进行相应的修改，每一种相对于老问题的新方法都要带来额外的开销，程序的可重用性差。面向过程的结构化程序设计方法一般适用于规模较小的软件的开发过程。

2. 软件工具及作用

“工欲善其事，必先利其器”，人们早就认识到工具在生产过程中的重要作用。熟练掌握各种软件工具，将大大提高开发效率和成功率。根据程序设计的各个阶段对信息需求的不同，主要的软件开发工具有三类：设计工具、分析工具、计划工具。

(1)设计工具是指在实现阶段对人们提供帮助的工具，主要有编辑、编译、连接和调试工具等。现阶段设计工具主要以集成开发环境(IDE)方式出现。对于C语言来说，主流的集成开发环境有Visual C++(简称VC)、C-Free等。

(2)分析工具主要是指用于支持需求分析和系统设计的工具，它们不是直接帮助开发人员编写程序，而是帮助人们认识与表述信息需求与信息流程，从逻辑上明确软件的功能与要求。对于C语言来说，微软Office系列中的Visio就是一个非常好用的分析工具。

(3)计划工具则是从更宏观的角度去看待软件开发。它不仅从项目管理的角度帮助人们组织与实施项目，把有关进度、资源、质量、验收情况等信息有条不紊地管理起来，而且考虑到了项目的反复循环、版本更新，实现了跨生命周期的信息管理与共享，为信息以及软件的复用创造了条件。对于C语言来说，微软Office系列中的Project就是一个非常好用的计划工具。

1.3 阅读在C语言学习过程的作用

了解了程序设计过程的各个阶段及程序设计方法，下面介绍在C语言初学阶段比较有用的一种途径，即阅读程序。

“熟读唐诗三百首，不会吟诗也会吟”，通过阅读大量别人编写的优秀程序，既可以加快对语法的熟悉，也可以快速地积累编程技巧，还可以通过阅读一些操作系统代码来加深对有关的基本原理的理解。下面介绍程序阅读的一般步骤。

1. 资料的收集

程序阅读的第一步就是收集所有与程序有关的资料。一般来说，资料可以分为以下几种类型。

(1)基础资料:如基本的语法资料和函数手册。

(2)和程序有关的专业资料:如要开发一个财会系统,必须先收集了解相关的财会资料。

(3)项目开发的相关资料:如需求分析报告和系统设计报告等。

2. 程序备份,构造可运行的环境

拿到程序后首先要备份,以避免阅读修改后找不到最初的程序。备份完成后最好给程序构造一个可运行的环境,可运行和不可运行的程序阅读起来的难度相差非常大。

3. 从头开始,分层次阅读

阅读程序一般要先找到 main()函数,然后逐层去阅读。总体来说,程序阅读也和结构化设计一样,最好先了解整体,然后了解细节,过早地关心细节容易“只见树木,不见森林”。

4. 写注释

对于风格不好的程序,写注释是程序阅读中最重要的一个步骤。通过注释,既可以加深对程序的理解,也可以对自己的阅读过程进行标注,避免看了后面忘了前面。另外,注释也必须言简意赅,没用的注释对程序的理解没有任何益处。

5. 重复阅读

不要希望一次性将程序阅读明白,只有通过反复阅读才能逐渐完成对程序的理解。

第 2 章

问题分析和算法设计

在程序设计过程中，接受任务后首先要做的是问题分析，真正了解要解决的问题，然后还要对解决的过程进行设计，也就是算法设计。下面对问题分析需要注意的事项，以及算法设计使用的技术和工具进行简单介绍。

2.1 问题分析

一般来说，问题分析首先是要对接受的任务进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题方法的过程，也就是通常所讲的“看懂题目”。与大规模软件的严格的需求分析过程类似，小规模程序设计的问题分析过程可以从以下几方面考虑。

1. 了解问题的有关专业基本知识

和程序阅读类似，问题分析的第一步，也是要对问题的有关专业基本知识有一定了解。例如程序设计中经常会碰到数学问题中的“水仙花数”问题，如果对于水仙花数的概念没有一定的了解，想解决这个问题是不太现实的。

2. 分析问题的输入和输出数据

问题的输入和输出数据主要涉及后面设计时数据的存储结构和处理算法。在 C 语言中，通过分析问题的输入数据确定需要采用的数据类型和存储结构，进而确定输入的格式和处理过程。其次，通过对输入数据的数据规模分析，可以预估问题的时间复杂度和空间复杂度，进而选择不同的算法。

3. 分析问题的处理过程

对于一般的算法问题，对处理过程的分析其实也是对算法的分析和选择过程。而对于一般的数据处理问题，对处理过程的分析主要是系统功能的确定和划分过程，也可以借助数据流图等工具辅助分析。

【例 2-1】 累加求和问题分析举例。

输入一个数字 $n(0\leqslant n\leqslant 10000000)$，请输出 $1+2+3+\cdots+(n-1)+n$ 的值。

问题分析过程：

(1)简单来看，本问题就是输入一个数字，循环累加后求出结果。

(2)从数据的输入和输出角度分析，输入的数据的存储可以考虑使用整型(int)，但从数据输出角度看，运算的中间结果使用长整型(long)也可能会溢出，需考虑使用 64 位整数(long

long)。

(3)从输入数据的规模和基本处理过程来看,如果单纯地使用循环可能运行时间较长,如果有运行的时间限制则最后实现的算法可以考虑使用累加公式 $n(n+1)/2$。

2.2　算法设计

“欲速则不达”,很多初学者看懂了问题后,一般会马上进行编码。这对于一般的小程序可能没什么问题,但若是输入数据复杂和规模较大的程序,还需要进行算法的设计。从软件工程角度看,算法设计包含在系统设计阶段内,主要包括数据设计(主要是数据结构)和算法设计两个部分。下面简单介绍数据设计及算法设计所采用的技术和描述工具,并举例说明设计的过程。

2.2.1　数据设计

数据设计是在分析问题的基础上进行的。一般程序的数据设计主要是对重要的变量和数据结构进行设计和命名。数据设计是否合理,对于后续的算法设计有着重要的影响。一般情况下,数据设计先于算法设计,也可在算法设计中穿插数据设计。

对于重要变量的设计,主要考虑的问题是数据的类型和命名。数据的类型其实主要根据问题的数据要求确定,而变量的命名则属于程序的风格。例如根据是否有负数来确定是 signed 或 unsigned,根据数据的大小来确定是 short、int 或 long,根据数据的精度要求来确定是 float 或 double。另外,数据在不同的 C 语言版本和编译器中是不同的。例如在 VC 中 short 是 16 位的,如果要单独处理字节数据一般设计为 char 类型。另外,VC 中 long 只有 32 位,如果要使用 64 位数据则可设计为 __int64 类型。一般常用的循环变量或临时变量可以不用特别考虑,在算法设计或编码时再定义也可以。

关于数据结构设计,建议做到以下几点:

(1)尽量使用简单的数据结构。一般来说,数据结构简单,操作相对也会简单。

(2)尽量使用经典的数据结构。经典的数据结构,理解起来相对容易。例如普通的二叉树大家可能都接触过,而像线段树这类不怎么使用的数据结构理解起来就比较困难。

(3)尽量使用顺序的数据结构。相对来说,动态的指针结构理解和调试起来难度会大很多。

(4)正确处理数据的冗余和关联,尽量做到平衡。

(5)设计数据结构时尽量考虑可复用性。可复用的数据结构会给下次开发带来很多方便。

(6)对于复杂的数据结构,还可以给出图形和文字描述,以便于理解。

2.2.2　结构化程序设计技术

结构化程序设计(Structured Programming)的概念最早是由迪克斯特拉(E. W. Dijkstra)在 1965 年提出的,是软件发展的一个重要的里程碑。一般来说,结构化程序设计是在结构分析的基础上进行的,也可以直接在设计环节使用结构化程序设计方法。对于结构化程序设计来说,主要观点有以下几点:

1. 清晰第一的设计风格

在结构化程序设计概念刚提出来时，迪克斯特拉就公开提出了在高级语言中取消 GOTO 语句的主张，进而引发了有关程序设计“是效率第一，还是清晰第一”的争论。这场辩论最后改变了许多人单纯强调程序效率的旧观，进而确定了“清晰第一，效率第二”的设计风格。这一设计风格也成了结构化程序设计应遵守的设计风格。

2. 结构化的控制结构

通过 20 世纪 60 年代以来一系列论文观点的不断综合，基本确定结构化的程序设计使用单入口、单出口，以及使用顺序、选择、循环三种基本控制结构的原则来构造程序。由这三种基本结构或三种基本结构复合嵌套构成的程序称为结构化程序。结构化程序具有结构清晰、层次分明及良好的程序可读性。

3. 逐步细化的实现方法

“自顶向下，逐步求精”是一种分而治之的思维方式，它早已经在日常生活和工作中被频繁使用，只不过我们不自觉或没意识到罢了。例如写一本书，首先要写一个提纲，全书分成几章；再对每一章列出本章分几节；然后对每一节分出几小节等；最后再具体写每个小节。

2.2.3 算法的描述工具

为了描述一个算法，可以有很多种方法。常用的算法表示方法有 PDL 语言、N-S 流程图等。

1. PDL 语言(Program Design Language)

作为在软件设计中广泛使用的设计语言之一，PDL 语言是由美国的 Caine 和 Gordon 在 1975 年提出的。它是介于自然语言和形式化语言之间的一种类自然语言，也称为结构化语言。除了用英语表示外，PDL 也可以使用中文表示。图 2-1 和图 2-2 分别表示了利用 PDL 描述选择结构和循环结构。

```
if<条件>then
  <PDL语句>
else
  <PDL语句>
end if
```

```
if<条件>then
  <PDL语句>
else if<条件>then
  <PDL语句>
else
  <PDL语句>
end if
```

图 2-1 利用 PDL 描述选择结构

```
WHILE型循环
loop while<条件>
  <PDL语句>
end if
```

```
UNTIL型循环
loop until<条件>
  <PDL语句>
end loop
```

图 2-2 利用 PDL 描述循环结构

例 2-2 和例 2-3 说明了如何使用英文和中文 PDL 对例 2-1 的累加求和问题进行算法描述。

【例 2-2】 累加求和。算法用 PDL 描述(英文)。

```
Read N
Set Sum = 0
Set I = 0
Loop while I< = N
    Sum = Sum + I
End loop
Print  Sum
```

【例 2-3】 累加求和。算法用 PDL 描述(中文)。

```
输入 N
初始化 Sum 和 I
循环 N 次
    利用 Sum 对 I 进行累加
输出 Sum
```

对于 PDL 的使用,以下几点需要注意:

(1)相对于自然语言,PDL 要严谨一些,有一定的语法,但是在实际的使用过程中,也不一定要严格遵守,如程序的控制结构也可使用中文的"如果……则……"结构,另外如例 2-3 中的"初始化 Sun 和 I"的简单过程描述,只要大家都能看懂,也是可以的。

(2)对于复杂问题用 PDL 进行描述,也可以使用结构化程序设计的"逐步求精"思想,先描述程序总体的控制结构,中间某些处理过程可以先用自然语言进行总体的描述,再进行细化描述。

2. N-S 图

N-S 图是结构化程序设计方法中用于表示算法的图形工具之一。对于结构化程序设计来说,传统流程图显得太复杂和有所欠缺。因为传统流程图出现得较早,它更多地反映了机器指令系统设计和传统程序设计方法的需要,难以保证程序的结构良好。另外,结构化程序设计的一些基本结构在传统流程图中没有相应的表达符号。例如,在传统流程图中,循环结构仍采用判断结构符号来表示,这样不易区分到底是哪种结构。为此,两位美国学者 Nassi 和 Shneiderman 于 1973 年就提出了一种新的流程图形式,并以两位创作者姓名的首字母取名为 N-S 图。

N-S 图的基本单元是矩形框,它只有一个入口和一个出口。长方形框内用不同形状的线来分割,可表示顺序结构、选择结构和循环结构。在 N-S 图中,完全去掉了带有方向的流程线,程序的三种基本结构分别用三种矩形框表示,将这种矩形框进行组装就可表示全部算法。这种流程图从表达形式上就排除了随意使用控制转移对程序流程的影响,限制了不良程序结构的产生。

与顺序、选择和循环这三种基本结构相对应的 N-S 图的基本符号如图 2-3 所示。图 2-3(a)和图 2-3(b)分别表示顺序结构和选择结构,图 2-3(c)表示循环结构。由图可见,在

N-S图中，流程总是从矩形框的上面开始，一直执行到矩形框的下面，这就是流程的入口和出口，这样的形式不可能出现无条件转移的情况。

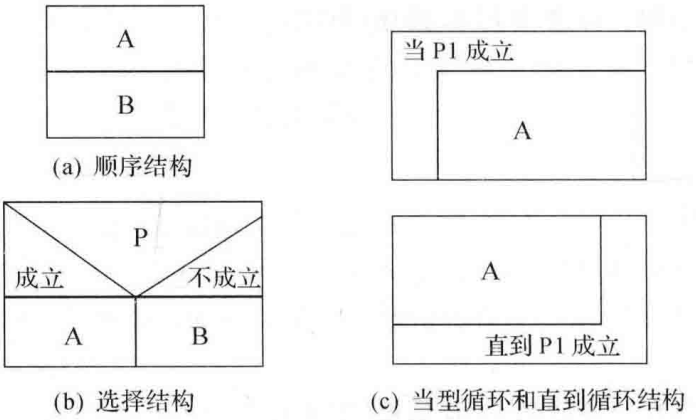

图 2-3 三种基本结构对应的 N-S 图

【例 2-4】 利用 N-S 图描述累加求和，如图 2-4 所示。

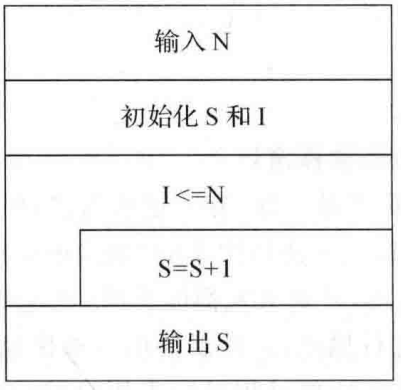

图 2-4 累加求和问题用 N-S 图描述

对于 N-S 图的使用，也有以下几点需注意：

(1)N-S 图符合结构化程序设计的基本思想，同时也可以使用“逐步求精”方法进行描述。

(2)对于初学者来说，N-S 图存在一定问题，当程序内嵌套的层数增多时，内层方框会越来越小，这会增加画图难度，并影响图形的清晰度。这个问题可以通过使用调用或转义方式。如图 2-5 所示，如果一个盒子小到一定程度后可以另外再画一个盒子进行描述。

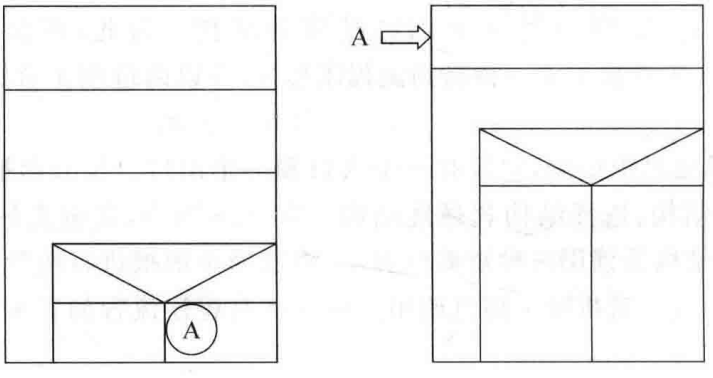

图 2-5 N-S 图转义画法示意

2.2.4　结构化程序设计示例

在了解了结构化程序设计的基本思想和算法描述工具后，下面介绍利用 N-S 图来对选择排序算法进行算法设计。

【例 2-5】 利用 N-S 图选择排序算法设计。

第一步，进行数据结构设计。首先，如果问题没有特殊要求，排序的数据可以使用数组进行存储。其次，根据可理解性和可维护性要求，先定义一个数组的最大长度的变量 DATASIZE。最后，为方便操作，可定义数组 data 及当前的数组长度 DataLen。

第二步，将问题的算法的处理过程先分解为输入、处理和输出三个部分，并画出 N-S 图，结果如图 2-6 所示。

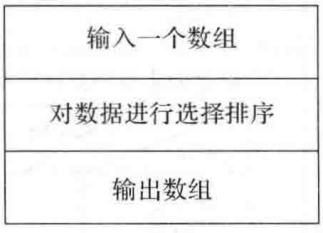

图 2-6　最开始的 N-S 图

第三步，一般情况下，可先对处理过程的关键过程进行细化，这里先对“对数据进行选择排序”的处理过程进行细化，得到如图 2-7 所示的结果。

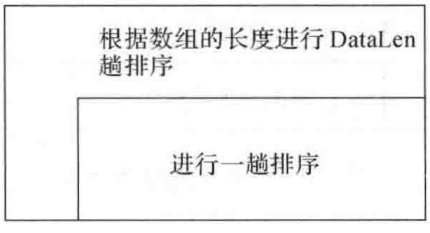

图 2-7　对数据进行选择排序的 N-S 图

第四步，在图 2-7 中的“进行一趟排序”还是比较模糊的，再进行细化后得到如图 2-8 所示的排序。当然，这里也可以对一趟排序中涉及的变量进行定义。

图 2-8　一趟排序的 N-S 图

第五步，再对图 2-6 的输入和输出进行细化，可以得到如图 2-9 所示的 N-S 图。当然，还可以对循环过程中涉及的循环变量进行定义，并对其中的几个条件加以改进。另外，还可以对其中的一些处理过程进行细化。

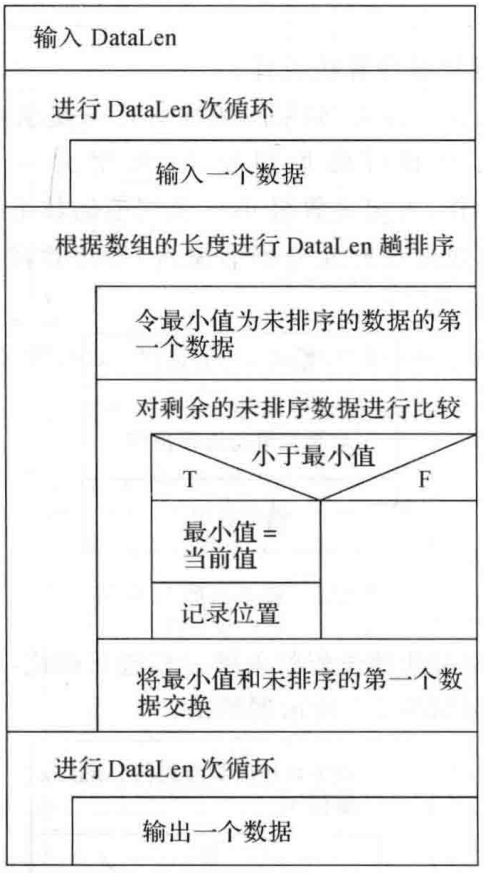

图 2-9 基本完成的 N-S 图

几点总结：

(1)很多初学者觉得进行算法设计并且进行描述很麻烦，很多时候都是编好了程序再回头进行算法的描述。当然，这种方法虽然很简单，但是，正如前面所介绍的，这种方法对付小程序可能没什么问题，但对于复杂的程序就不行了。“自顶向下，逐步求精”的分而治之的思想能降低解决问题的难度，希望初学者能坚持使用。

(2)很多初学者一开始很纠结算法设计时的细化程度，觉得算法设计要确定所有的变量及处理过程，最好细化到每条语句。其实程序的算法设计细化程度可以因人而异，只要清楚了处理过程，编码时没有难度和二义性就可以了。

第 3 章

编码和程序风格

完成了问题分析和设计，程序设计就进入了编码阶段。一般来说，比较完整的算法设计完成后，编码只是将设计结果翻译成 C 语言的过程，基本上是体力活，因此很多编程人员自嘲地称自己为“码农”。在编码阶段，除了需要掌握 C 语言的语法及编程工具外，编程的风格也是非常重要的方面。

3.1 C 语言版本和编程环境的选择

C 语言的版本和编程环境是相互关联的，不同的编程环境支持不同的 C 语言版本。对于一般的初学者，是没必要关心 C 语言版本的，常用的编程环境之间代码都可以通用。但是对于 C 语言的各个版本和编程工具，也有一些要注意的问题，下面将进行简单的介绍。

3.1.1 C 语言版本

在 C 语言的发展历程中，出现了许多不同的版本，各版本之间大同小异，主要有以下 3 个版本。

1. C90

为统一 C 语言版本，1983 年美国国家标准学会(ANSI)成立了一个委员会来制定 C 语言标准。1989 年 C 语言标准被批准，通常称之为 ANSI C。又由于这个版本是 1989 年完成制定的，因此也被称为 C89。

后来 ANSI 把这个标准提交到国际化标准组织(ISO)，并于 1990 年被 ISO 采纳为国际标准，称为 ISO C。又因为这个版本是 1990 年发布的，因此也被称为 C90。

除了标准文档在印刷编排上的某些细节不同外，ISO C(C90)和 ANSI C(C89)在技术上完全一样。目前，几乎所有的开发工具都支持 ANSI/ISO C 标准。这是 C 语言中使用最广泛的一个版本。

2. C99

在 ANSI 标准化发布了 C89 标准以后，C 语言的标准在一段相当长的时间内都保持不变。标准在 20 世纪 90 年代才经历了改进，这就是 ISO/IEC 9899:1999。这个版本就是通常提及的 C99，也是继 C90 后的第二个官方标准。

3. C11

C11 标准是 C 语言标准的第三个官方标准。2011 年 12 月 8 日，国际标准化组织(ISO)和