C语言程序设计

- 解决初学者的疑惑和难点
- 训练编程者的思想和方法
- 导学基本知识和编程技巧
- 导引独立编程和上机调试

杜友福 主编

科学出版社
www.sciencep.com

C 语言程序设计

杜友福　主　编

科学出版社
北　京

内 容 简 介

本书为C语言程序设计课程教材，全书共十三章，通过大量的示例，特别是通过算法和结果的分析，帮助读者理解C语言的基本概念、基本算法以及各种语法规则，学会编程的方法。每章后均配有习题。为了便于教学和自学，本书配有《C语言程序设计导学》。

本教材特别适合于本专科非计算机专业的初学者，也可供计算机等级考试者和其他各类学习者使用和参考。

图书在版编目(CIP)数据

C语言程序设计/杜友福主编. —北京：科学出版社，2003.12
ISBN 7-03-012574-6

Ⅰ.C… Ⅱ.杜… Ⅲ.C语言-程序设计-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第110411号

责任编辑：冯贵层 王雨舸/责任校对：王望荣
责任印制：高 嵘/封面设计：晓阳工作室

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
湖北省京山金美印刷有限责任公司印刷
科学出版社出版 各地新华书店经销
*
2004年1月第 一 版 开本：787×1092 1/16
2004年1月第一次印刷 印张：17 1/4
印数：1—10 000 字数：428 000

定价：23.60元

（如有印装质量问题，我社负责调换）

前　言

计算机已成为在校学生必须掌握的一种工具，要能熟练地使用计算机，特别是利用计算机开发软件，必须至少熟练地掌握一门计算机语言。C 语言以其独特的功能，成为深受广大学生青睐的一种语言，许多学校已把C 语言作为第一语种讲授。由于C 语言语法的复杂性和语言的灵活性，使得许多初学者学习起来感到有一定的难度。基于此，本书作者结合多年的教学经验编写了此书。书中通过大量的示例，特别是通过算法和结果的分析，帮助读者理解C 语言的各种语法规则和各种编程的方法。

学习程序设计不能局限于能听懂、能看懂，更重要的是要能自己编写程序。因此要求读者一方面要通过书中的例题，学会分析编程的思想和方法，另一方面要自己独立地编写程序，并上机进行调试，以理解性地记忆C 语言的一些基本知识和编程技巧。

本书共十三章，每章后面均配有习题。在做选择题时，不仅要选择答案，更重要的是要学会分析，为什么是对的？错在什么地方？这样才有收获；在做程序填空题时，最好是将填空后的完整程序在机器上调试一下，这样有助于对程序及算法的分析和理解；在做编程题时，要自己独立编写并上机调试。通过一系列环节的训练，才能达到学好C 语言的目的。

本书由杜友福教授任主编，并负责全书的统稿，李新玉、李克清、周云才、伍良坤、曹芝兰、丁兴亚任编委。第一、二章由伍良坤、曹芝兰、丁兴亚编写，第三、四、五章由李新玉副教授编写，第六、七、八、九章由杜友福教授编写，第十章由李克清副教授编写，第十一、十二、十三章由周云才副教授编写，李新玉副教授参加了第一、二章的修改。

由于作者水平有限，书中难免会有疏漏和不足之处，敬请各位读者和专家提出宝贵意见，以便再版时予以修订。

杜友福

2003 年 10 月

目　录

第一章　程序设计基本概念

1.1　程序和程序设计

计算机是在程序的控制下工作的。要让计算机能够正常运行，需要预先编写好计算机工作步骤的指令序列，即系统程序；要利用计算机来解决一个具体的实际问题，同样需要编写程序，即应用程序。这些都要进行程序设计工作。

程序设计的工作是由程序员完成的，程序员必须了解一些有关程序运行和测试的知识，还要了解计算机所提供的软、硬件的支持情况，否则难以完成程序设计的全部工作。我们把实现程序设计所需要的软、硬件支持统称为程序设计环境(programming environment)。不同类型、不同年代、不同语言的机器支持环境是不一样的。

1.1.1　程序

程序是指令的序列。程序设计语言就是用户用来编写程序的语言，它是人与计算机之间进行交流的工具，实际上也是人指挥计算机工作的工具。通常，用户在用程序设计语言编写程序时，必须满足相应语言的语法格式，并且逻辑要正确。只有这样，计算机才能根据程序中的指令做出相应的动作，最后完成用户所要求完成的各项工作。程序设计语言是软件系统的重要组成部分，一般可分为机器语言、汇编语言和高级语言三类。

1. 机器语言

所有的计算机都只能接受和识别由“0”和“1”组成的二进制信息。每种类型的计算机都分别规定了由若干个二进制数据组成的一组指令，这种由“0”和“1”组成的二进制编码表示的命令，称为机器指令。机器指令一般由操作码和操作数两个部分组成，操作码表示该指令所要完成的功能，操作数部分提供完成这个功能所需要的数据或数据在内存中的地址。一条机器指令对应于一种基本的操作，例如某种类型的计算机规定以“10000000”表示一个“加法”操作，以“10010000”表示一个“减法”操作。

机器语言就是直接使用机器指令的集合作为程序设计手段的语言。用机器语言编写的程序，计算机硬件可以直接识别，因此，它的执行速度比较快，基本上充分发挥了计算机的高速性能。但是针对一种计算机所编写的机器语言程序，一般不能在另一种类型的计算机上运行，而且程序的编写难度较大，修改、调试也不方便，容易出错，程序的直观性较差。

2. 汇编语言

为了便于理解和记忆，人们采用能够帮助记忆的指令助记符来代替机器语言指令代码中的操作码，用地址符号或十进制数来代替操作数。某条指令的指令助记符一般是描述该指令功能的英文单词的缩写，如用“ADD”表示加法操作，用“SUB”表示减法操作。这种用指令助记符及地址符号或十进制数表示的指令称为汇编指令，而用汇编指令编写的程序则称为汇编语言源程序。

由于汇编语言采用了助记符，因此，汇编语言又称为符号语言。它比机器语言直观，容易记忆和理解，用汇编语言编写的程序比机器语言程序易读、易检查、易修改。但是，汇编语言中的

指令与机器语言中的指令是一一对应的，因此，对于不同类型的计算机，汇编语言源程序还是不能通用。而且，计算机不能直接识别用汇编语言编写的程序，必须由一种专门的翻译程序，即汇编程序，将汇编语言源程序翻译成机器语言程序，计算机才能执行。

3. 高级语言

机器语言和汇编语言都是面向机器的语言，一般称为低级语言。它们对机器的依赖性很大，用它们开发出的程序通用性差，而且要求程序的开发者必须熟悉和了解计算机硬件的每一个细节，因此，它们面对的用户一般是计算机专业人员。随着计算机技术的发展及计算机应用领域的不断扩大，计算机用户的队伍也在不断壮大，而且这个队伍中绝大部分人不是计算机专业人员。为此，从20世纪50年代中期开始，逐步发展了面向问题的程序设计语言，称为高级语言。高级语言与具体的计算机硬件无关，其表达方式接近于被描述的问题，接近于自然语言和数学语言，易为人们接受和掌握。用高级语言编写程序要比用低级语言容易得多，大大简化了程序的编制和调试过程，使编程效率得到大幅度的提高。

高级语言的显著特点是独立于具体的计算机硬件，通用性和可移植性好。例如，一个用Basic语言编写的程序，可以在装有Basic解释程序的所有机器上执行。这样，各行各业的计算机用户不必深入了解机器的硬件结构就可以根据需要来编制适用于各种机器的程序。

目前，计算机高级语言已有上百种之多，得到广泛应用的也有十几种，并且几乎每一种高级语言都有其适用的领域。下面列出几种最常用的高级语言及其最适用的领域：

Basic	教学和微小型应用程序的开发
Fortran	科学及工程计算程序的开发
Pascal	专业教学和应用程序的开发
C	中、小型系统程序的开发
C++	面向对象程序的开发
Cobol	商业、交通和银行等行业应用程序的开发
Lisp	人工智能程序的开发
Prolog	人工智能程序的开发
FoxBase	数据库管理程序的开发
Visual FoxPro	数据库管理程序的开发

1.1.2 程序设计

程序设计就是设计、书写及检查程序的过程。用某种程序设计语言编写的程序，通常要经过编辑处理、语言处理、装配链接处理后，才能够在计算机上运行。

所谓编辑处理是指人们通过编辑程序将编写的源程序送入计算机。编辑程序可以使用户方便地编辑源程序，包括添加、删除、修改等操作，直到用户满意为止。

所谓语言处理，是指将用户编写的源程序转换成机器语言的形式，以便计算机能够识别和运行。这一转换过程是由翻译程序自动完成的，翻译程序除了要完成语言间的转换外，还要进行语法、语义等方面的检查。翻译程序统称为语言处理程序，一般可分为：汇编程序、编译程序和解释程序三种类型。

1. 汇编程序

汇编程序是一种由专业的软件开发商提供的系统软件，它能将用汇编语言编写的源程序翻译成某种类型的计算机的机器语言程序，即目标程序，这一翻译过程称为汇编。图1.1是汇编程序的功能示意图。

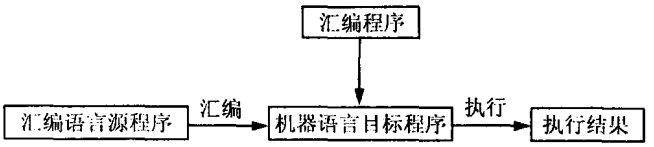


图 1.1　汇编过程示意图

汇编程序通常是将一条汇编语言指令翻译成一条机器语言指令。在翻译的过程中，还要对每一条指令进行语法检查，若有错误，就中断汇编过程，并以某种方式输出错误的类型及有关信息，以便用户进行修改。若没有错误，就自动生成相应的目标程序，再经过一定处理之后，便可以运行了。运行的是可执行的目标代码，与源程序及汇编程序无关。若源程序作了某些修改，则必须再重新进行汇编。

2. 编译程序

编译程序的功能是将用高级语言编写的源程序翻译成机器语言程序，即目标程序，这一翻译过程称为编译。如图 1.2 所示。

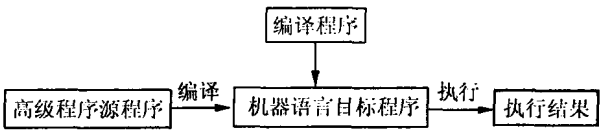


图 1.2　编译过程示意图

与汇编程序不同的是，编译程序往往需要将一条高级语言的语句转换成若干条机器语言指令，而且翻译的过程也要复杂得多。

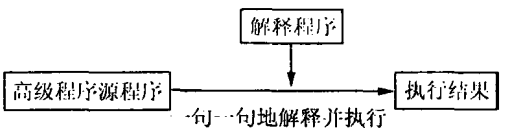


图 1.3　解释过程示意图

3. 解释程序

解释程序的工作过程是将高级语言源程序一句一句地读入，每读入一个语句都要对它进行分析和解释，若有错误就即时中断其解释过程，并通知用户进行修改，若没有错误就按照解释结果执行所要求的操作。如图 1.3 所示。

解释方式灵活、方便，交互性好，解释执行程序的过程中也不会产生目标程序。但因为是边解释边执行，所以程序的执行速度很慢，如果源程序中出现循环结构，解释程序也要重复多次地解释循环体中的每一条语句，造成很大浪费，而且解释方式在运行时始终离不开翻译程序。

要设计出一个好的程序，首先必须了解利用计算机解决实际问题的过程，其次必须掌握程序设计的基本技术，最后还要熟练地掌握一种程序设计语言。图 1.4 简略地描述了用计算机解决问题的基本过程。

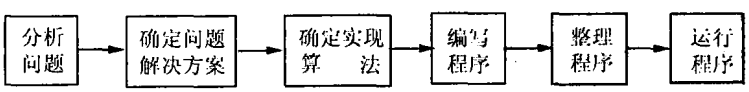


图 1.4　计算机解决问题的基本过程

如何才能编写出高质量的程序呢？设计程序时应遵循以下基本原则：

(1) 正确性。是指程序本身必须具备且只能具备程序设计规格说明书中所列举的全部功能。它是判断程序质量的首要标准。

(2) 可靠性。是指程序在多次反复使用过程中不失败的概率。

(3) 简明性。目标是要求程序简明易读。

(4) 有效性。程序在计算机上运行需要使用一定数量的计算机资源，如CPU 的时间、存储器的存储空间。有效性就是要在一定的软、硬件条件下，反映出程序的综合效率。

(5) 可维护性。程序的可维护可分为校正性维护、适应性维护和完善性维护。一个软件的可维护性如何直接关系到程序的可用性，因此应特别予以关注。

(6) 可移植性。程序主要与其所完成的任务有关，但也与它的运行环境有着一定的联系。软件的开发应尽可能远离机器的特征，以提高它的可移植程度。

为了有效地进行程序设计，不仅要掌握一门高级语言，还应该学会对各类问题拟定出有效的解题步骤——算法设计。有了正确的算法，才能够编制程序。算法的好坏决定了程序的优劣，因此，程序设计的核心任务之一就是设计算法。

1.2 算　　法

算法是一组有穷规则的集合，该规则规定了一个解决某一特定问题的方法和操作序列。例如，期末考试前的复习计划，就是“复习算法”；到医院看病，先挂号，后诊断、检查，再取药等，是“看病算法”。而计算机算法就是为计算机解题设计的有明确意义的运算步骤的有限集合。世界著名的计算机科学家Wirth(沃思)提出了一个著名的公式表达程序设计的实质，即：

程序＝算法＋数据结构

其含义是：程序是在数据的特定的表示方式和基础上，对抽象算法的具体描述。

算法具有如下特点：

(1) 有穷性。它是一个有穷动作序列，且每一步都能在有限的时间内完成。

(2) 确定性。每一步是确定的，不能“模棱两可”。

(3) 有效性。每一步都能得到有效执行，并产生有效和确定的结果。

(4) 有零个或多个输入。

(5) 有一个或多个输出。

上述所讲的算法特性，约束人们去正确地书写算法，使之能够正确无误地执行，达到求解问题的预期效果。同时，算法还应该具有直观、清晰、易懂的表示形式，以利于维护、修改和交流。

1. 简单算法举例

例 1.1　给出求 x1＋x2＋x3＋x4＋x5 的值的算法。

算法分析：

(1) 手工计算步骤为：

① 求 x1 与 x2 的和，得到两个数之和；

② 将上一步的和与 x3 相加，得到 3 个数之和；

③ 将上一步的和与 x4 相加，得到 4 个数之和；

④ 将上一步的和与 x5 相加，得到 5 个数之和。

从手工计算过程中可知：其运算方法类似于用算盘计算该题的过程，每次仅求出两个数之

和，其中一个加数为上一步所得的结果，另一加数为多项式中的一项，重复这个过程，直到加到最后一项为此。

(2) 适合计算机处理的算法。该算法能更加简洁地表达上述解题过程，并具有通用性。先定义几个变量：设置变量 s 表示多项式之和，其初值为零；设置变量 a 表示多项式中的一项，它的值可以为 x1，x2，…，x5；用 i 记录被加了几次，其初值为 1。解题步骤为：

① s←0；

② i←1；

③ a←xi；　　(使 a 等于多项式中的第 i 项)

④ s←s+a；　　(求和，并将结果保留在 s 中)

⑤ i←i+1；　　(计数增值)

⑥ 若 i≤5，则重复③，④，⑤各步，否则，计算结束；

⑦ 输出 s。

2. 算法的描述

算法的描述可以用自然语言(如汉语或数学公式)表达，也可以用伪代码表示，或自然语言与伪代码联合使用，还可以用流程图或结构化流程图来表示。

(1) 用自然语言表达。例1.1 就是用自然语言描述的算法，这样的算法通俗、易懂、直观、容易掌握，但其算法的表达与计算机的高级语言形式差距较大，通常用在较简单的问题中。

(2) 用伪代码表示。伪代码(pseudo code)是一种介于自然语言和计算机语言之间的算法描述方法。其特点是不拘泥于语言的语法结构，而着重以灵活的形式表现被描述的对象。伪代码没有标准化的形式，程序设计人员可自行约定。

例 1.2　输入一个整数，将它倒过来输出，例如输入 12345，输出为 54321。试给出其算法。

解法 1　用自然语言描述：

① 输入一个整数送给 x；

② 求 x 除以 10 的余数，结果送给 d，并输出 d；

③ 求 x 除以 10 的整数商，结果送给 x；

④ 重复②，③步，直到 x 变为零时终止。

解法 2　用伪代码表达为：

① 输入一个整数送 x；

② while(x≠0)do

```
{  d=x%10;
   输出 d;
   x=x/10;
}
```

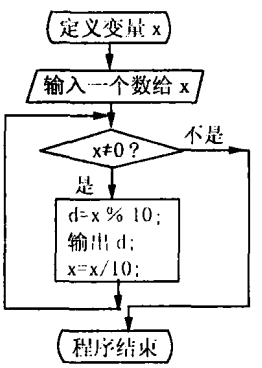


图 1.5　用传统流程图表示算法

(3) 用传统流程图描述算法。流程图(flowchart)是用各种几何图形、流线及文字说明来描述计算过程的框图，如例 1.1 的算法就可以用图 1.5 所示的流程图来表示。

用流程图描述算法的优点是：直观，设计者的思路表达得清楚易懂，便于检查修改。传统流程图中常用的符号如表 1.1 所示。

例 1.3　从键盘输入 100 个数，求出这 100 个数的累加和，并找出它们的最大值和最小值。试用传统流程图描述解决这个问题的算法。流程图见图 1.6。

表 1.1　传统流程图常用的符号

图　形	名　称	说　明
→	流向线	表示算法流程方向，箭头方向为入口，相背方向为出口
	开始、结束框	开始框仅有流向线从其流出，而结束框仅有流向线流入
	处理框	也称矩形框，表示确定的处理、步骤
	输入输出框	表示原始数据的输入和处理结果的输出
	判断框	允许有一个入口，两个或两个以上的可选择的出口
	连接点	用来将画在不同地点的流程线连接起来
	功能调用框	表示执行模块或子程序
……[	注释框	用于书写注释信息

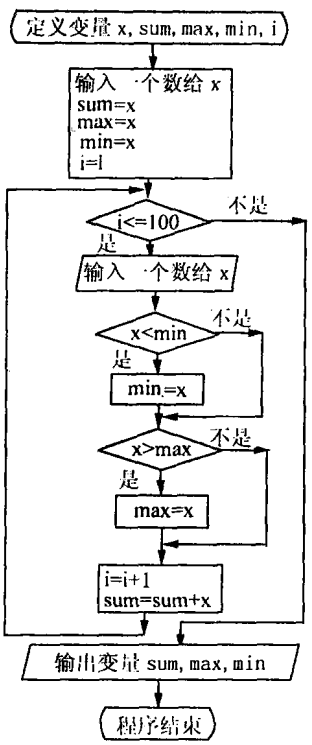


图 1.6　传统流程图的应用

1.3　结构化程序设计方法

结构化程序设计方法已经成为程序设计人员必须遵循的一种规范，该方法要求在程序设计工作中贯穿“自上而下，逐步细化和程序结构模块化”的设计思想，把算法、数据结构和结构化程序设计方法有机地结合起来。

1. 三种基本结构

人们经过长期的实践，从程序的结构中归纳出常用的、具有良好性能的三种基本结构，即顺序结构、选择结构和循环结构。三种基本结构称为结构化流程图的基本图或基本结构元素（见图1.7）。基本图可以嵌套、组合，形成复杂的程序流程图。图 1.7(a)的顺序结构表示“处理 1”先于“处理 2”执行；图 1.7(b)的选择结构表示当条件满足时执行“处理 1”，否则执行“处理 2”，两者之中只能有一个处理得到执行；在图 1.7(c)的循环结构中，首先判断条件是否满足，若条件满足，则执行“处理”，之后返回再判断条件，由此构成循环，直至条件不满足为止。

2. 结构化的程序设计方法

结构化程序设计方法要求在设计程序时只能使用上述三种基本结构，程序设计的基本过程一般由分析所求解的问题、抽象数学模型、选择合适算法、编写程序、调试运行程序直至得到正确的结果等几个阶段所组成。其设计步骤可以分为：

(1) 确定要解决的问题。接到任务后，首先对要处理的对象进行调查，了解其要求，然后确定哪些任务由计算机完成，哪些任务手工完成，并对计算机要完成的任务给出详细的原始数据、处理后的结果及处理功能。

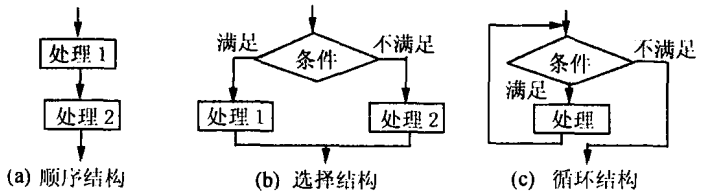


图 1.7　传统结构化流程图的基本图

(2) 分析问题，建立数学模型。对要解决的问题进行分析，找出它们运算操作和变化的规律，然后进行归纳，并用抽象的数学语言描述出来。

(3) 选择计算方法。根据特定的数学模型，选择适合计算机解决问题的方法。当同一问题有多个解决方案时，可根据计算的速度、求值的精度、计算要求与计算机硬件性能的关系进行选择。

(4) 确定数据结构和算法。选择好计算方法后，还要合理地安排数据的组织形式，并针对这样组织的数据设计与之相适应的算法。

(5) 绘制流程图。根据已确定的算法，画出程序的流程图。这样能使人们思路清晰，减少编写程序的错误。

(6) 编写程序。把流程图描述的算法用计算机语言描述，然后将其转变成能由计算机识别和运行的目标程序。应该注意的是，要选择一种合适的语言来适应实际算法和现有的计算机环境的需要，并要正确地使用语言，准确地描述算法。

(7) 调试程序。调试程序就是对送入计算机的程序进行排错、试运行的过程，调试的结果是得到一个能正确运行的程序。

这里需要指出的是：程序中的错误有两种，一种是语法错误，另一种是逻辑错误。语法错误通常在源程序被编译时能被编译程序及时发现，提示出错信息，因此相对比较容易排除。而逻辑错误常常是潜在性的，编译程序可能事先无法查出。例如，在程序中使用了零作除数，将导致程序执行时出现异常，这就要求程序设计者要根据设计文档，反复地、不厌其烦地阅读源程序，并精心地设计测试方案和测试数据，力求及时发现程序中的错误。

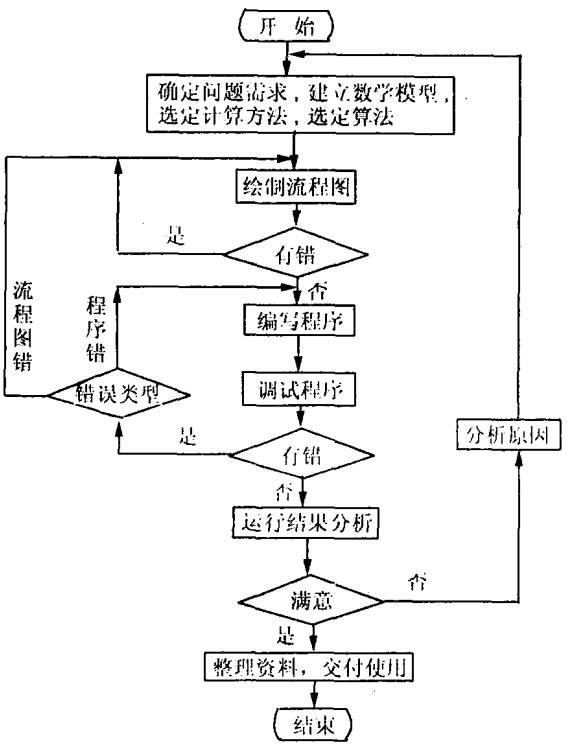


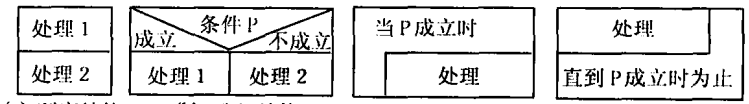
图 1.8　程序设计的工作流程

(8) 整理资料，交付使用。程序调试通过后，应将有关资料进行整理，编写程序使用说明书，交付用户使用。

图 1.8 描述了程序设计过程的工作流程。

前面介绍了结构化的算法，提出了用三种基本结构可以组成结构化的算法。用这种方法来设计程序，如同使用预购件来搭盖房屋一样简单。描述结构化的算法，可以使用 N-S 流程图。

N-S 流程图是由美国学者I. Nassi 和B. Shneiderman 提出的一种新的流程图形式。在这种流程图中，完全去掉了带箭头的流程线，禁用程序的转向语句，将全部算法都写在一个矩形框内，因此，N-S 流程图也叫盒图。在N-S 流程图中只能出现如图1.9 所示的4 种图形符号，用它们可以组合出任意复杂的程序结构。



(a) 顺序结构　(b) 选择结构　(c) 当型循环　(d) 直到型循环

图 1.9　N-S 流程图的基本结构

例 1.4　从键盘输入一个整数，判别该数是否为素数(素数，也称为质数，是指只能被 1 及其自身整除且大于 1 的正整数，如 2、3、5、7 等都是素数)。

用N-S 流程图描述解决这个问题的算法如图 1.10 所示。

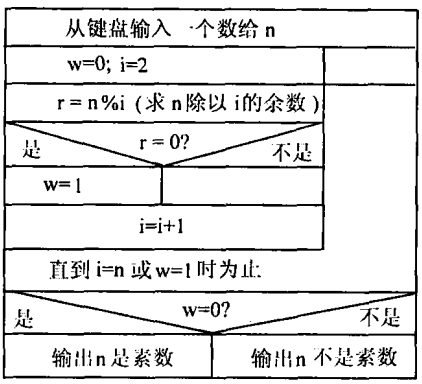


图 1.10　判断素数的N-S 流程图

怎样去设计出一个程序呢？有两种截然不同的思考问题的方法，一种叫做“自上向下”的方法，一种叫做“自下向上”的方法。例如，要写一部著作，往往先确定目标和主题，然后构思若干个章，定出各章的题目和大致内容，把每一章又分解为若干个节，再细分为若干段……即由粗到细，逐步具体化，直到能执笔写出文字为止。这种方法是先有全局，先进行整体设计，然后再进行下一层的设计，逐步实现精细化。这种方法称为“自上向下，逐步细化”的方法。“自下向上”的方法则与此相反，先从具体的局部工作入手，由若干个局部构成一个整体。如建筑工人盖房子，先从一砖一瓦入手，完成一个个局部的结构，最后整体结构也就完成了。

显然，采用“自上向下”的程序设计方法能使程序员做到胸有全局、通盘考虑，不致顾此失彼、头重脚轻。过去不少程序设计人员往往采用“自下向上”的方法，拿到题目以后，没有做充分考虑就开始编写程序，这种程序通常漏洞百出，质量较差，可读性低。

要设计出结构化的程序，应当采用以下的方法：

(1) 自上向下。

(2) 逐步细化。

(3) 程序结构模块化。

所谓模块化，是指将一个大任务分解成若干个较小的部分，每一部分承担一定的功能，称为“功能模块”。各个模块都可以分别由不同的人来编写程序和调试，便于组织人力完成较复杂的任务。

自上向下、逐步细化和模块化技术三者是紧密结合的，是结构化程序设计方法不可分割的三个要点。

例 1.5　编程打印出 3～5000 之间的所有质数。

分析：我们采用自上向下、逐步细化的方法来处理这个问题。即先把这个问题分解为两个相对较小的问题：① 判别某数是否为质数；② 利用循环判别 3～5000 之间的所有的数是否为质数，若是就把它打印出来。

把这两个小问题分别用两个程序模块来实现，并由此给出解决这个问题的程序流程图，如图1.11(a)所示。模块的功能定了，但其内容还是“笼统”的、“抽象”的，因为还没有解决怎样才能实现“判别某数是否为质数”的功能，需要进一步“细化”。在这里我们给出一种算法：用整数i除以从2，3，4…一直到i－1的每一个正整数，如果都除不尽，则i是质数；否则，只要其中有一个被除数能除尽，则i就不是质数。其实现算法如图1.11(b)所示，该程序模块运行结束后，若变量flag的值为1，则i是质数，否则i不是质数。

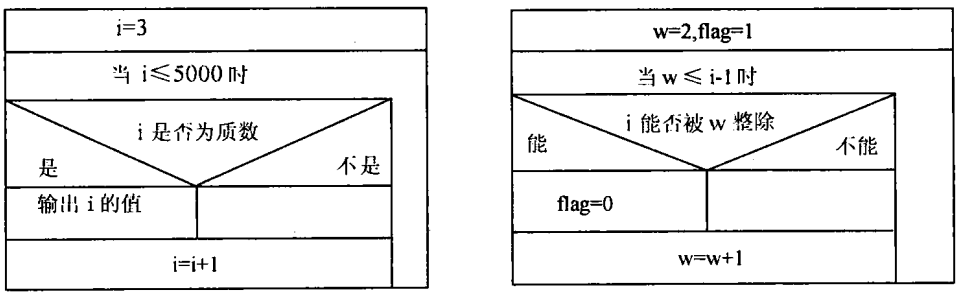


图1.11　输出3～5000之间所有的质数

到此为止，已全部细化完毕，每一部分都可以分别直接用高级语言的语句来表示。由此可以看出：“自上而下，逐步细化”是结构化程序设计方法的核心。应当学会利用这种方法把一个总的任务（即抽象化的要求）逐步具体化。可以说，逐步细化的过程是使求解问题“由抽象到具体”的过程。

1.4　C语言程序的基本结构

下面介绍几个简单的C程序，然后从中分析C程序的特性。

例1.6

```
main( )
{
  printf( "This a c program.\n");
}
```

本程序的作用是输出以下一行信息：

```
This is a c program.
```

其中main函数是主函数，每一个C程序都必须有一个main函数，函数体由花括弧{ }括起来。本例中主函数内只有一个输出语句，printf是C语言中的输出函数。双引号内的字符串原样输出；“\n”是换行符，即在输出“This is a c program.”后回车换行；语句以分号结尾。

例1.7

```
main ( )                    /*求两数之和*/
{ int a,b,sum;              /*定义变量*/
  a=123; b=456;
  sum=a+b;
  printf ("sum is %d\n", sum);
```