C语言程序设计

C YUYAN CHENGXU SHEJI

张颖江 胡燕 主编

科学出版社

C 语言程序设计

张颖江　胡　燕　主编
崔洪芳　周双娥　汤练兵　副主编
瞿　坦　主审

科学出版社
1998

内 容 简 介

本书以标准C语言为基本框架，以个人计算机上流行的Turbo C语言为背景，介绍了C语言程序设计的基本方法。全书共分十一章，内容包括程序设计的基本思想，C语言的基本概念、语法规则，C语言编程的方法及基本实验等。

本书根据作者多年的教学经验，直接针对没有学过程序设计与高级语言的非计算机专业读者编写教材，是一次极具挑战性的尝试。全书层次清晰，深入浅出，注重理论与实践的结合，可供大、中专院校非计算机专业学生，成人与职业、技校学生，计算机培训班学员及广大计算机自学者、爱好者使用。书中每一章都配有习题，第十一章还给出了基本实验内容，进一步的实验内容可参阅科学出版社出版的《计算机应用基础实验教程》。

图书在版编目(CIP)数据

C语言程序设计/张颖江，胡燕主编. -北京：科学出版社，1998.7
ISBN 7-03-006803-3

Ⅰ.C… Ⅱ.①张…②胡… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(98)第16252号

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
中国科学院武汉分院科技印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
1998年7月第 一 版 开本：787×1092 1/16
1998年7月第 一次印刷 印张：14 3/4
印数：1～15 000 字数：360 000
定价：17.80元

前　言

计算机程序设计语言的产生与其环境、用途有着直接的关系。许多年来，人们一直把BASIC语言作为学习计算机程序设计的入门语言，其理由是BASIC语言语法简单，书写容易，上机实验环境极易构造，因而较适合于程序设计语言的初学者学习。尽管如此，BASIC语言的能力确实极其有限。在现实环境中，人们极少用BASIC语言书写计算机应用软件。这就是说，即便学会了BASIC语言，要想真正编写计算机应用软件还得至少再学习一门其他程序设计语言，如C语言。

提到C语言，它一直被认为是计算机专业人员使用的程序设计语言，是需要有其他程序设计语言作基础才能问津和掌握的语言。那么，能不能让程序设计语言的初学者直接学习C语言呢？本书作者根据多年的教学经验认为答案是肯定的。

虽然C语言的概念比BASIC语言复杂，规则比BASIC语言要多，但它比BASIC语言更严谨，更加结构化，程序书写效率更高，能帮助计算机程序设计语言初学者养成良好的编程习惯，提高程序设计的起点和能力。

我们针对上述问题并根据计算机程序设计语言初学者的特点，结合近年来计算机技术的发展趋势，在力图将C语言比较复杂的概念通俗化、叙述条理化，使之同样适合初学者学习和掌握的同时，根据不同的计算机系统、不同的出版商、不同的C语言版本有一定的区别的实际情况，以标准C语言为基本框架，以个人计算机上流行的Turbo C语言为背景，逐步使读者建立起C语言的基本概念，掌握利用C语言进行程序设计的基本方法和技能，为今后进一步提高计算机程序设计语言的能力奠定基础，特别是为学习Windows环境下的程序设计、为学习面向对象的程序设计语言打下坚实的基础。

全书共分十一章。第一章介绍了程序设计的基本思想；第二章介绍了C语言的数据类型及其运算；第三章介绍了赋值语句和输入输出操作；第四章介绍了流程控制语句；第五章介绍了数组；第六章介绍了函数；第七章介绍了指针；第八章介绍了结构、联合和枚举；第九章介绍了文件；第十章介绍了编译预处理和行编译；第十一章介绍了与实验有关的内容。本书中的例题均已通过上机调试，许多例子还有运行结果。

全书层次清晰，深入浅出，注重理论与实践的结合，适用于大中专院校本、专科学生，也可供成人与职业学校学生、计算机培训班学员及广大计算机自学者、爱好者使用。书中每一章都配有习题，第十一章还给出了基本实验内容，进一步的实验内容则可参阅科学出版社出版的《计算机应用基础实验教程》。

本书由张颖江、胡燕主编，崔洪芳、周双娥、汤练兵副主编。其中，第一章由张

颖江编写，第二章和第七章由王春枝编写，第三章和第五章由胡燕编写，第四章由崔洪芳编写，第六章由周双娥编写，第八章和第九章由汤练兵编写，第十章由贾亦青编写，第十一章和附录由黄启荃编写。

难能可贵的是，全国高等教育教材研究工作委员会委员、全国计算机等级考试委员会委员、湖北省高等学校计算机基础教学指导委员会主任委员、华中理工大学博士生导师瞿坦教授，在百忙之中抽出宝贵时间，担任本书主审，提出了许多宝贵的意见，在此深表谢意！

在没有学过程序设计与高级语言的非计算机专业读者中直接讲授C语言，是一次极具挑战性的尝试，由于编者水平有限，纰漏疏忽之处在所难免。今后我们将根据大家的意见与建议在再版时对本书做必要的修改和补充，使之不断充实和完善，以达到在非计算机专业读者中普及C语言的目的。

编者

1998年6月于武汉

目　　录

第一章　程序设计概论

语言是人类用于信息交流的工具，计算机语言是人与计算机交换信息的中间媒介和工具。人们要使用计算机，就需告诉计算机“做什么”，“怎样做”——即用计算机可识别的语言编写加工、处理的步骤，或者说编写程序。计算机按照程序的规定进行处理，并将处理结果通知人，这就是人们要掌握计算机语言的根本原因。显然，计算机语言与程序设计有着自然的联系，学习计算机语言就是为了设计程序，更好地使用计算机，让计算机为人所用。

1.1　程序设计语言概述

1.1.1　程序设计语言分类

从广义上讲，程序是指事情进行的先后次序。例如，生产计划、访问计划、课程表都是程序。在计算机中，程序是指为解决某一问题而设计的计算机指令的集合。程序设计语言则是编写计算机程序所使用的语言，即为描述计算机操作步骤而制定的一套标记符号、表达格式及使用的语法规则。

程序设计语言是人们根据描述问题的需要而设计的。如今，人们已经设计出了许多种程序设计语言，但其中只有极少部分得到了比较广泛的应用。对程序设计语言的分类可以从不同的角度进行，如面向机器的程序设计语言、面向对象的程序设计语言、面向过程的程序设计语言等。其中，最常见的分类方法是根据程序设计语言与计算机硬件的联系程度将其分为三类，即机器语言，汇编语言和高级语言。前两类依赖于计算机硬件，有时统称为低级语言；而高级语言与计算机硬件无关。因此可以说程序设计语言经历了由低级向高级发展的过程。

1. 机器语言

计算机所能理解和执行的是以“0”和“1”组成的二进制编码表示的命令，称为机器指令。机器指令由操作码和操作数组成，其具体的表现形式和功能与计算机系统的结构相关联。机器语言就是直接用这种机器指令的集合作为程序设计手段的语言。机器语言的优点是计算机能够直接识别；其缺点是难记忆、难书写、编程困难、可读性差且容易出错。机器语言是面向机器的语言，因机器而异，可移植性极差。

2. 汇编语言

为了克服机器语言的缺点，人们采用了助记码和符号地址来代替机器指令中的操作码与操作数。如用 ADD 表示加法操作，用 SUB 表示减法操作，且操作数可用二进制、八进制、十进制和十六进制数表示。这种表示计算机指令的语言称为汇编语言，它是从这种简单助记忆符方式发展起来的一种程序设计语言。

汇编语言也是一种面向机器的语言，但计算机不能直接执行汇编语言程序。用它编写的程序必须经过汇编程序翻译成机器指令后才能在计算机上执行。目前，许多系统软件的核心部分仍采用汇编语言编制。

3. 高级语言

所谓高级语言就是更接近自然语言、更接近数学语言的程序设计语言。它是面向应用的计

算机语言，其优点是符合人类叙述问题的习惯，而且简单易学。使用较多的高级语言有 Basic，Pascal，Fortran，Cobol，C，C++，Java 等。其中：Basic 是一种简单易学的小型会话式语言；Pascal 是适合结构化程序设计的语言；Fortran 是一种问世早、且适合于科学计算的语言；Cobol 是适用于数据处理的语言；C 是适用于系统开发的语言；而 Java 是面向网络环境的程序设计语言。目前高级语言正朝着非过程化发展，即只需告诉计算机"做什么"，"怎样做"则由计算机自动处理。高级语言的发展将以更加方便用户使用为宗旨。

用高级语言编写的程序，用户不能在计算机上直接执行，它必须通过"翻译"变成机器指令，才能在计算机上执行。

1.1.2 程序的编译与解释

我们知道，在计算机语言中，除机器语言之外的其他语言书写的程序必须经过翻译或解释才能在计算机上执行。因此，计算机上能提供的各种语言，必须配备相应语言的"编译程序"，或"解释程序"。通过"编译程序"或"解释程序"使人们编写的程序能够最终得到执行的工作方式分别称为程序的编译方式和解释方式。

1. 编译方式

编译是指将用高级语言编写好的程序（又称源程序），经编译程序编译，形成可由计算机执行的机器指令程序（称为目标程序）的过程。图 1.1 描述了编译程序将源程序编译成目标程序的过程。

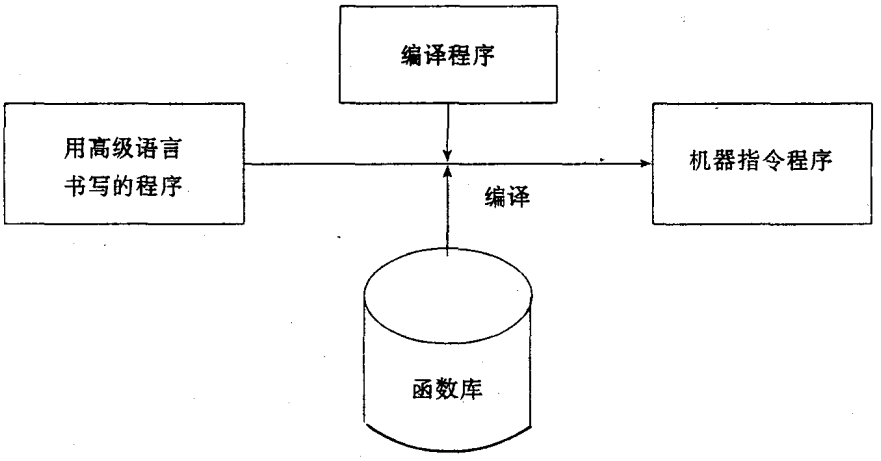


图 1.1 编译方式

在图 1.1 中，函数库中存放的是一些称为库函数的标准程序。编译系统或解释系统提供这些程序的目的是为了减少程序员的程序设计工作量，同时规范程序设计。譬如说：要在某程序中计算 sin(x)的值，如果 sin(x)不是库函数，我们就必须利用如下公式进行计算：

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots+(-1)^{n-1}\cdot\frac{x^{2n-1}}{(2n-1)!}$$

显然，这里存在两个问题，一是增加了编程的工作量，二是程序规范性不良。如某个程序在计算 sin(x)值时，可能使用级数的前 10 项，而另一个程序可能只使用级数的前 6 项。两者计算的目的相同，但计算结果却不完全一样。

如果将常用的函数事先编写好，放在一起，形成一个函数库，供程序员编程时调用，则上述

问题即可得到解决。这正是编译系统或解释系统要提供库函数的原因。

在编写程序时，人们往往希望使用自己熟悉的程序设计语言，这就为共同编写较大的程序带来了问题。为了能够将多种不同语言编写的程序段有机地结合在一起，形成一个机器指令程序，在编译方式下，通常又将编译过程分成编译和连接两个阶段。

在编译阶段，编译程序将源程序翻译成一种中间代码并以文件形式存入存储介质(如磁盘)，然后提交给连接程序，由连接程序将所有中间代码文件以及程序所用到的库函数连接成一个可以直接执行的机器指令程序，其过程如图 1.2 所示。

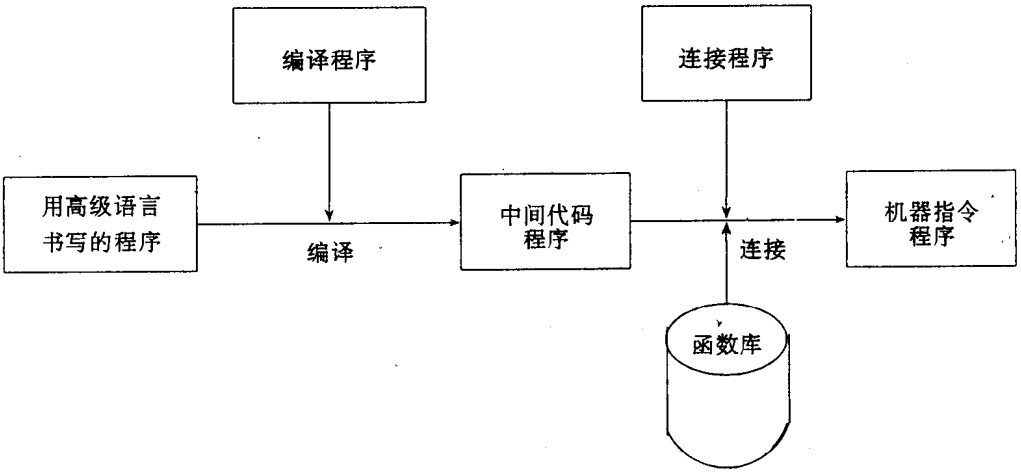


图 1.2　程序的编译与连接过程

编译方式的优点主要有：

(1) 目标程序可以脱离编译程序而独立运行。

(2) 目标程序在编译过程中可以通过代码优化等手段提高执行效率。

其缺点是：

(1) 目标程序调试相对困难。

(2) 目标程序调试必须借助其他工具软件。

(3) 源程序被修改后必须重新编译连接生成目标程序。

2. 解释方式

解释是将高级语言编写好的程序逐条翻译并执行的过程。它不像编译方式那样把源程序翻译成目标程序，而是将源程序解释一句立即执行一句。图 1.3 是解释方式的工作原理。

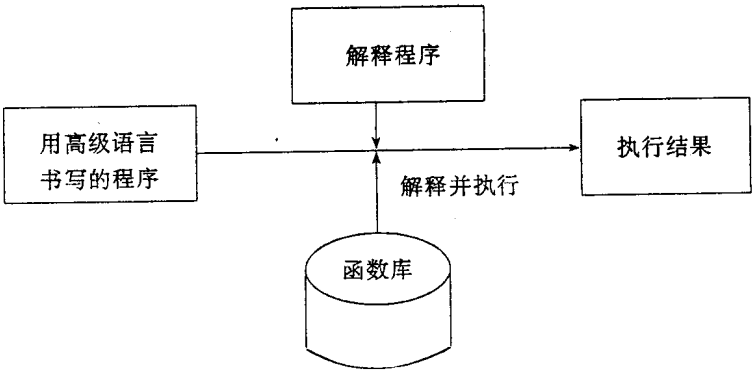


图 1.3　解释方式

解释方式的优点包括：

(1) 可以随时对源程序进行调试。

(2) 调试程序手段方便。

(3) 可以逐条调试源程序代码(称为源代码)。

其主要缺点是：

(1) 被执行程序不能脱离解释环境。

(2) 程序执行速度慢。

(3) 程序未经代码优化，工作效率低。

无论是编译程序还是解释程序，都需要事先送入计算机内存中，才能对源程序(也应在内存中)进行编译或解释。

为了综合上述两种方法的优点，克服缺点，目前，许多编译软件都提供了集成环境，以方便程序设计者。所谓集成环境是指将程序编辑、编译、运行、调试集成在同一环境下，使程序设计者既能高效地执行程序，又能方便地调试程序，甚至是逐条调试和执行源程序。

1.2 程序设计技术

简单地说，程序设计就是设计、书写及检查程序的过程。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，最后要熟练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。

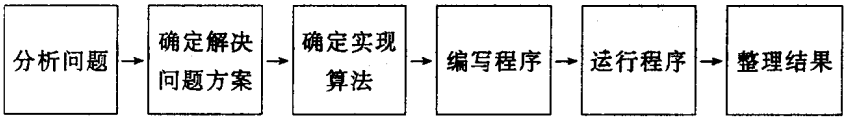


图 1.4 计算机解决问题的基本过程

如何才能编写出高质量的程序呢？下面是设计程序时应遵循的基本原则。

(1) 正确性。正确性是判断程序质量的首要标准。所谓正确性是指程序本身具备且只具备程序设计规格说明书中所列举的全部功能。

(2) 可靠性。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。简明性的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。例如，用高级语言编写程序就比用汇编语言编写程序的可移植性好。

为了有效地进行程序设计，不仅要掌握一门高级语言，还应该学会针对各类问题拟定出有效的解题方法和步骤——即算法设计。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。

1.2.1 算法与算法描述

1. 算法的概念

所谓算法，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的操作序列。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：

程序＝算法＋数据结构

就是说：“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”。

算法具有如下性质：

(1) 有穷性：它是一个有穷动作序列，且每一步在有限时间内完成。

(2) 确定性：每一步是确定的，不能“模棱两可”。

(3) 有效性：每一步应当有效地执行，并产生有效和确定的结果。

(4) 有零个和多个输入。

(5) 有一个或多个输出。

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

2. 简单算法举例

例 1.1 求 $x_1+x_2+x_3+x_4+x_5$ 的值。

算法分析：

(1) 手工计算步骤为：

a. 求 x_1 与 x_2 的和；得到两个数之和

b. 将上一步的和与 x_3 相加；得到三个数之和

c. 将上一步的和与 x_4 相加；得到四个数之和

d. 将上一步的和与 x_5 相加；得到五个数之和

从手工计算过程可知：其运算规则类同于用算盘计算，每次仅求两数之和，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止。

(2) 适合计算机处理的算法。为更简洁地表达上述算法，并具有通用性，可先定义几个变量：设变量 s 表示多项式之和，其初值为零；设 a 表示多项式中的一项，它的值可以为 $x_1,x_2,\cdots,x_5$；用 i 记录被加了几次，其初值为 1，则算法步骤为：

a. s←0

b. i←1

c. a←x_i　　(使 x 等于多项式中的第 i 项)

d. s←s＋a　　(求和，并将结果保留在 s 中)

e. i←i＋1　　(计数增值)

f. 若 i≤5 则重复 c,d,e 各步；否则，计算结束

g. 输出 s

例 1.2 求正整数 n,m 的最大公约数。

人工求解时，先将它们分解成质因数的乘积，然后找出“公共因数中的最大者”。如 n＝6，

m=8,有 n=2×3 和 m=2×2×2,所以最大公约数是 2。

这样的思维过程,目前计算机还不能做。因此要找出适合计算机的求解算法,可以根据

$$\frac{m}{q}-\frac{n}{q}=\frac{m-n}{q}$$

即有两个数 m,n 分别可以被 q 整除,则它们的差也能被 q 整除的原理,用欧几里得辗转相减算法来完成。其步骤为:

a. 输入 n,m 的值

b. 若 n>m,则 n=n−m

c. 若 m>n,则 m=m−n

d. 若 n=m,则计算结束,输出 n(或 m),它就是最大公约数;否则,重复 b,c,d 各步

3. 算法的描述

算法的描述,可以用自然语言(如汉语或数学语言)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表示。

(1) 用自然语言表达。例 1.1 和例 1.2 就是用自然语言描述的算法,这样描述的算法通俗易懂,直观且容易掌握,但其表达的算法与计算机的具体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。它结构性较强,比较容易书写和理解,修改起来也相对方便。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对象。伪代码没有标准化,可自行约定。

例 1.3 用伪代码表示例 1.2 的算法为:

```
a. 给 n 和 m 赋初值
b. while (n≠m)
   { 2-1  if(n>m) n=n−m;
     2-2  if(m>n) m=m−n;
   }
c. 输出 n
```

例 1.4 输入一个整数,将它倒过来输出。例如输入 12345,输出为 54321。

思路:一般写数时,是从高位到低位顺序写,识别它的大小时,从低位到高位顺序地读。这就给解题提供了一个"直接"的启示,其算法为:

自然语言描述:a. 输入一个整数送给 x

b. 取 x 的最右边的一位数送 d,输出 d

c. 从 x 中去掉 d

d. 重复 b,c 步,直到 x 变为零时终止

用伪代码表达为:

```
a. 输入一个整数送 x;
b. while (x≠0) do
   { 2-1  d=x% 10;
     2-2  输出 d;
     2-3  x=x/10;
   }
```

(3) 用传统流程图描述算法。流程图(Flow chart)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。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。

图 1.5 是用传统流程图描述算法时使用的常用符号。

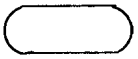
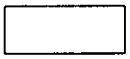
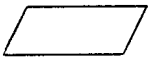
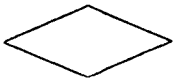
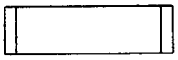
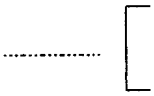
图　形	名　称	说　　明
	流向线	表示算法流程方向；箭头方向为入口，相背方向为出口
	开始、结束框	开始框仅有流向线从其流出，而结束框仅有流向线流入
	处理框	也称矩形框，表示确定的处理、步骤
	输入输出框	表示原始数据的输入和处理结果的输出
	判断框	允许有一个入口，两个或两个以上的可选择的出口
	连接点	用来将画在不同地点的流程线连接起来
	功能调用框	表示执行模块或子程序
	注释框	用于书写注释信息

图 1.5　传统流程图常用的符号

人们经过长期的实践，将流程图归纳为三种基本的简单结构，即顺序、选择和重复（又称循环）。这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（图 1.6）。

图 1.6c 和图 1.6d 都是重复结构，但图 1.6d 描述的是先执行“处理”，然后判断，即“处理”至少被执行一次。而图 1.6c 描述的是先判断后“处理”，因此，在“条件”一开始就不满足的情况下，“处理”部分未被执行。

基本图可以嵌套，形成复杂形式的流程图。由基本图的嵌套结构组成的流程图称为是结构化的，否则就是非结构化的。如果一个流程图是结构化的，则其相应的程序就是一个结构化程序，否则是非结构化程序。

例 1.5　图 1.7 所示的流程描述了例 1.1 的算法。

（4）N-S 结构化流程图。N-S 结构化流程图是在美国学者 I. Nassi 和 B. Schneiderman 1973 年提出的方法的基础上形成的一种以他们的名字命名的流程图，其主要特点是取消了流程线，全部算法由一些基本的矩形框图顺序排列组成一个大矩形表示，即不允许程序任意转移，而只能顺序执行，从而使程序结构化。图 1.8～1.11 是 N-S 流程图与传统流程图在表示上的对应关系，其中 a 表示传统流程图，b 表示 N-S 流程图。

例 1.6　画出例 1.1 的 N-S 流程图，见图 1.12。

例 1.7　某班有若干学生，要求根据学生成绩打印优（85 分以上），良（75～84 分），及格（60～74 分），不及格（60 分以下）。