高等学校教材

BASIC
程序设计教程

（第二版）

谭浩强　张基温　编著

高等教育出版社

高等学校教材

BASIC 程序设计教程

（第二版）

谭浩强　张基温　编著

高等教育出版社

(京)112 号

内 容 提 要

本书以结构化程序设计思想为中心，引导读者掌握规范化的程序设计方法。主要内容包括：算法、计算机与程序，BASIC 程序设计初步，BASIC 程序的基本数据结构，模块化程序设计，输入与输出设计，数据文件，动态数据结构和算法设计举例等。

本书文字叙述通俗生动，阐述方法深入浅出。书中讲授的内容具有较强的应用价值，使读者能够容易地将所学的数据结构知识、典型的算法设计方法及结构化程序设计方法有机地结合起来并应用到实际问题中，提高解决实际问题的能力，同时也增加了读者的学习兴趣。书中介绍的程序算法具有普遍性，它们很容易用其它语言实现。书中还给出了丰富的例题与习题。

本书可作为各类高等学校和各种计算机学习班的教材，也可供自学。

图书在版编目(CIP)数据

BASIC 程序设计教程/谭浩强，张基温编著 .—2 版 .—北京：高等教育出版社，(1999 重印)

ISBN 7-04-005383-7

Ⅰ.B… Ⅱ.①谭… ②张… Ⅲ.BASIC 语言－程序设计－高等学校－教材 Ⅳ.TP312BA

中国版本图书馆 CIP 数据核字(95)第 10586 号

*

高等教育出版社出版

北京沙滩后街 55 号

邮政编码：100009 传真：64014048 电话总机：64016633

新华书店总店北京发行所发行

高等教育出版社印刷厂印装

*

开本 787×1092 1/16 印张 18.75 字数 460 000

1987 年 5 月第 1 版

1996 年 3 月第 2 版 1999 年 2 月第 4 次印刷

印数 75 097－97 107

定价 15.20 元

凡购买高等教育出版社的图书，如有缺页、倒页、脱页等质量问题者，请与本社经营办公室联系调换，电话：64054588

第二版前言

本书第一版自1987年出版以来，受到广大读者的欢迎和爱护。许多高校将其选为教材。根据几年来使用的情况，我们对它进行了修订，现在第二版又与大家见面了。

在修订中，对初版中部分内容进行了增删或改写，以便于读者更易于理解，更适应于计算机技术发展的需要。

近年来我们常收到一些读者来信说，有人认为BASIC语言已经过时，没有学习的价值了。问是否真的如此。我们不这样认为。我们觉得不能抽象地评论哪一种语言绝对地好，哪一种语言绝对地不好；每一种语言都有它自己的优点和弱点，都有自己适用的领域。与一切事物一样，都有自己发生、发展和消亡的规律，不以个人的意志为转移。如果一种语言几十年来始终受到人们的喜爱和欢迎，必然有它的理由。只能说哪一种语言适合于哪一个领域。BASIC语言以其易学易用受到广大初读者的欢迎。如果让一个计算机初学者一开始就去学Ada语言显然是不合实际的，尽管Ada是很优秀的现代语言。打一个比方：马路上有小汽车、大卡车、公共汽车、摩托车、三轮车和自行车。有人问哪一种车最好？无法一言以蔽之。不同的人会做出不同的回答。对广大老百姓来说，最受欢迎的可能是自行车，因为自行车方便灵活、不用司机、不用车房、不需汽油、价格便宜、家家买得起、人人都能骑…。不能说，由于出现了小汽车就应当取缔自行车，因为这显然脱离国情、脱离群众。

事实上，各种语言都在竞争中发展。如果某一种语言墨守成规，固步自封，就必然会落伍、会被淘汰。BASIC语言也在不断变化发展，不断推出新的版本。它们已经和初期的基本BASIC有很大的差别，千万不能以60年代基本BASIC的情况来评论今天的BASIC。从本书介绍的内容可以看到，BASIC语言的功能是丰富的，除了本书中介绍的内容以外，BASIC语言还有一个重要的功能，就是画图。由于篇幅关系，本书未介绍画图功能。过去有人以为BASIC只能用来解决或处理很简单的问题。实际上现在的BASIC不仅易于学习，也适用于各种应用，无论国内还是国外，有不少的应用程序是用BASIC语言写的。本书专门写了第七、八章，介绍如何用BASIC语言去实现复杂的数据结构和较难的算法。有了这些知识，就可以使用BASIC编写程序去解决某些较复杂的问题。

当然，每一种语言都有它的局限性。例如BASIC语言没有提供指针类型数据。人们应当根据本人的基础和要处理问题的特点来选择应该用哪一种语言，注意扬其长避其短。事实上，真正学好了一种语言之后，在需要时再学习其它语言，是能举一反三、事半功倍的。

国家教育委员会工科计算机基础课程教学指导委员会制订的“高级语言程序设计”课程基本要求中，规定大学工科学生可以选学BASIC, FORTRAN, PASCAL和C四种语言中的一种或几种。

我们仍然认为，BASIC语言是计算机初学者的一种较好选择。当然，不能误认为：学会了BASIC就是学会了计算机。BASIC只是计算机科学技术之中的沧海一粟，入门之后需要继续提高。

本书的内容比国内大多数BASIC教材要深一些。希望本书能有助于提高学生程序设计的水平。

由于作者水平有限，经验不足，本书定有不少缺点甚至错误，敬请广大专家、教师和读者不吝指正。

作者谨识

1995.2

前　言

随着计算机科学技术日益发展与普及，程序设计语言学习的重心，也相应由计算机语言本身转向程序设计。

算法是程序的精髓。要设计出好的程序，必须先设计出好的算法。本书以算法设计贯穿始终，列举了大量典型算法，并介绍了算法设计方法。通过对典型算法的了解，读者不仅可以掌握算法设计的基本方法，而且还丰富了读者的“算法库”，遇到实际问题时，能很快理出一条正轨的解题思路，不至于束手无措、盲目乱凑。

程序的处理对象是数据。数据结构的形式，直接或间接地影响着算法的复杂性和解题效果。因此在本书中也较系统地介绍了数据结构知识，并把数据结构的设计看作程序设计的重要内容之一。

决定程序质量的另一关键是程序的风格和程序设计者所采用的设计方法。结构化程序设计方法是一种比较成熟的程序设计方法。它要求程序设计者必须遵循一定的规范，不能随心所欲、无规则地编写程序。这样使得程序质量不再主要取决于个人的素质与兴趣；程序也不再是一种“艺术品”，而是可以用工程方法处理的规范化的产品。

本书把算法、数据结构、结构化程序设计方法三者有机地融合在一起。也就是说：程序＝算法＋数据结构＋结构化方法。

离开计算机语言，就谈不上程序设计。本书中的程序都用 BASIC 语言来描述。BASIC 是一种简单易学、应用最为广泛的程序设计语言。为此，我们使用了一定的篇幅来介绍 BASIC 语言的基本知识。语言的介绍，仅仅是为了使读者掌握一种工具并对程序设计有更生动而真实的理解。本书并不把重点放在各种语句及语法规则的介绍上，以免喧宾夺主。读者要进一步地掌握语言知识，是很容易找到一本参考手册的。需说明的是，BASIC 语言具有“方言性”。本书以在长城 0520，IBM－PC，APPLE－Ⅱ（加 Z80 插件，并在 CP/M 支持下），ALTOS 等多种微型计算机上都可运行的 Microsoft BASIC 为蓝本，也兼顾了一般 BASIC 的特点，以便大多数读者都有实践的机会。

有人可能会问，当今像 Pascal，C，Ada 这样一些优秀的计算机语言已经问世，为什么还要选用 BASIC 语言作为本书中描述程序的工具呢？诚然，Pascal，C，Ada 语言是一些学者所推崇的计算机语言，它们是按照最新的观念设计的。然而，BASIC 语言毕竟是一种应用最为广泛的并已解决了大量实际问题的计算机语言。英国学者 M·詹姆斯说得好：“忽略了这一事实，将使我们回到与中世纪类似的一种情形中。那时候，上流社会喜欢用书面拉丁语进行社交，绝大多数人是用口头英语表达思想。后来，英语得到了发展，而拉丁语却僵化了。”我们不应当用一种语言排斥另一种语言，每种语言都有它的应用领域。我们认为，对初学者来说，BASIC 有它的优势。事实上，能得到广泛应用本身，就足以说明 BASIC 语言也具有优于其它语言的诱人之处，而且面对那些熟悉早期的 BASIC 版本的人们的责难，BASIC 已得到发展和完善。特别是 True BASIC 和 Quick BASIC 等并不逊色于 PASCAL 和其它语言，它们的绘图等功能却是其它一些语言所望尘

莫及的。

考虑到读者的基础不同，并为满足不同层次的教学需要，本书采取了新的体系结构。书中内容，从第一章起，每增加一章都可构成一个独立体系，形成了可供各类读者自学或教学使用的多层次结构。具体说来，第一章到第四章，是最基本的部分，中级学习者则可以学习第一到第六章；第七章、第八章是提高部分，可供读者进一步深入学习使用。在作教材使用时，如果学时较少，可以学习前六章为主，七、八章作为选修或将部分内容穿插到有关章节中介绍。对理工科各专业，如学时数够的话，建议最好学完第七、八两章，使读者在学完本课程后具有较强的程序设计能力。在计算机技术日益普及的今天，我们认为适当增加本课程的内容、提高本课程的深度是适宜的，也是可能的。对各章中一些内容较深的部分、在目录中加了“*”号、初次学习时可以不学，留待读者今后参考。

本书每一章后面都附有一定数量的习题，可以根据不同程度读者的需要选做其中的一部分。其中有一些习题难度较大，可供程度较高的读者选做或思考。

本书是根据全国高等学校计算机基础教育研究会和全国非计算机专业教材评审组多次会议上许多同志提出的建议而编写的。在正式出版前，曾作为内部交流教材，提供给部分高校使用以征询意见。许多单位的同志给予了热情的支持与鼓励，并对本书提出了不少宝贵的意见。本书出版前，清华大学计算机系吴文虎教授审阅了该书稿。在此对曾经关心、支持和帮助过本书出版的同志致以诚挚的谢意。

本书作为深化计算机基础教育的一种尝试，定有不妥之处，乞请指教。

作者

1987.10

目　录

第一章　算法与计算机

计算机是一种人们进行脑力劳动的辅助工具。它的任务是按照人的旨意解题或处理问题。程序就是人们解题旨意的表现，它规定了计算机的解题方法和步骤。对不同的问题人们要设计不同的程序来驱动计算机的解题过程。初学者在学习程序设计之前首先应当了解算法与计算机的基本知识。

§1.1　算　　法

1.1.1　算法概述

概要地说，任何解题过程都可以归纳为

理解题意→建立模型→表现模型→分析结果

过程模型是程序设计中一种传统的解题方法。它表现为在规定环境下，按一定规则组成的操作序列。这个操作序列称为算法。下面是几个算法的例子。

例 1.1　两盘磁带(x 和 y)内容互换。

有两盘磁带，各录有不同的内容，例如 x 录有音乐，y 录有英语会话。要交换两盘磁带的内容，必须借助于第三盘磁带(这里称其为 temp)。其算法可以描述如下：

step1：将 x 带上的音乐复制(保存)到 temp 带，记作 x⇒temp。

step2：将 y 带上英语会话复制到 x 带，记作 y⇒x。

step3：将 temp 带上的音乐复制到 y 带，记作 temp⇒y。

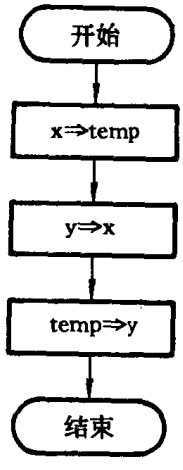


图 1.1

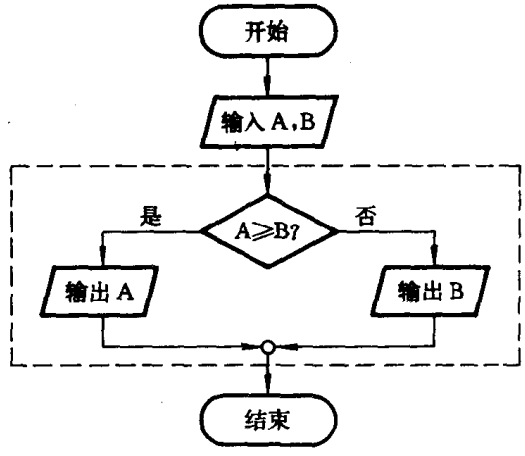


图 1.2

顺序地执行完过程 step1，step2，step3 后，便可以将 x 盘与 y 盘上的内容互换。这一算法称

为 x, y 内容互换算法。它也可以用图 1.1 所示的流程图描述。图中注有“开始”与“结束”的框称为端点符，矩形框称为处理框，带箭头的线称为流程线。用流程图描述的算法比用自然语言描述的算法要简洁，容易理解。

例 1.2 两数中取大者。

输入两数 A, B，输出其中大者。其算法实现如下：

step1：输入两数 A, B。

step2：比较 A 与 B，若 A≥B，则输出 A；否则输出 B。

该算法的流程图描述如图 1.2 所示。图中，斜边框称为 I/O 符或数据框，菱形框称为判断框。经过判断框后，算法流程分为排它的两支：或者执行“是”分支，或者执行“否”分支，二者不可兼得。如果把虚线中的部分（相当于 step2）看作一个整体，算法也可以看成由顺序执行的两个框组成。

例 1.3 求 $\sum_{n=1}^{5} n$，即 1 + 2 + 3 + 4 + 5 的值。

除 n 之外，再取一个量 sum 存放相加过程中的中间结果。这样求 5 以内各自然数之和的算法可以描述为：

step0：设初值，1⇒n，0⇒sum。

step1：sum + n⇒sum，n + 1⇒n（执行完 step1 后，sum 值为 1，n 值为 2）。

step2：sum + n⇒sum，n + 1⇒n（执行完 step2 后，sum 值为 3，n 值为 3）。

step3：sum + n⇒sum，n + 1⇒n（执行完 step3 后，sum 值为 6，n 值为 4）。

step4：sum + n⇒sum，n + 1⇒n（执行完 step4 后，sum 值为 10，n 值为 5）。

step5：sum + n⇒sum，n + 1⇒n（执行完 step5 后，sum 值为 15，n 值为 6）。

step6：（因为 n 值已超出需计算的范围，不再求和，sum 的值变为最终值）输出 sum。

在上述算法中，sum 与 n 是在算法执行过程中其值不断变化的两个量，这种量称为变量。

分析上述算法可以发现 step1 ~ step5 是完全相同的，即同样的“sum + n⇒sum，n + 1⇒n”要重复执行 5 次。若要计算 $\sum_{n=1}^{100} n$，则重复执行 100 次。这种情形可以采用如下简明的算法描述：

step0：设初值，1⇒n，0⇒sum。

step1：判断，若 n≤5，则执行 sum + n⇒sum，n + 1⇒n，然后再执行 step1；否则执行 step2。

step2：输出 sum。

这一算法的流程图描述如图 1.3 所示。请注意虚线框内的部分。图 1.2 中，经过判断框把流程分为两个分支，有一支将流程返回到该判断框之前，使得算法中的某一部分在一定的条件下（这里是 n≤5）重复执行。通常把这种算法结构称为重复结构或循环结构。显然，重复结构的算法要比完成同样操作的顺序结构算法简洁、高效。

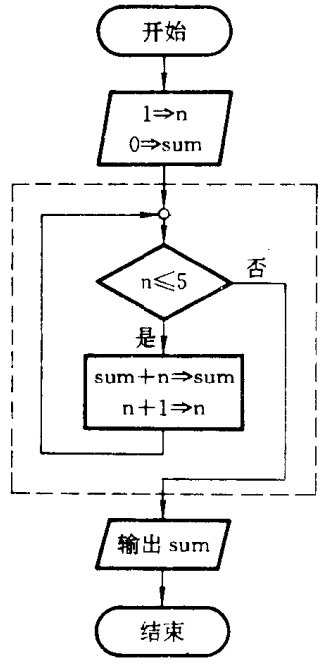


图 1.3 前 5 个自然数相加

例 1.4 用筛法求 1 到 1000 之间的全部素数(质数)。

求素数表的方法很多,与众不同的是"埃拉托色尼(Eratost－henes)筛法"。埃拉托色尼是与欧几里德差不多同时代的希腊著名数学家,他求出了世界上第一张素数表。他的办法是:在一张纸上写上 1 到 1000 的全部整数,然后进行以下操作:

step1:挖掉 1。

step2:重复执行如下操作:用下一个未被挖掉的数 x_i 去除它后面的每一个未被挖掉的数 x_i,把能整除的数挖掉,直到这个数后面没有可被挖掉的数为止。

step3:输出纸上所剩下的数。

执行这个算法后,纸被挖得象一个筛子,如图 1.4 所示。所以这种求素数的方法被称为筛法。

① 2 3 ④ 5 ⑥ 7 ⑧ ⑨ ⑩ 11 ⑫ 13 ⑭ ⑮ ⑯ 17 ⑱ 19 ⑳

㉑ ㉒ 23 ㉔ ㉕ ㉖ ㉗ ㉘ 29 ㉚ 31 ㉜ ㉝ ㉞ ㉟ ㊱ 37 ㊳ ㊴ ㊵

41 ㊷ 43 ㊹ ㊺ ㊻ 47 ㊽ ㊾ ㊿ … …

图 1.4 埃拉托色尼筛法

这一算法的流程图结构与图 1.3 相似,留给读者自己去完成。

例 1.5 有 n(n>2)个数,找出其中最大者。

本题的解法如同打擂台:先令一人上台,再令第 2 人到第 n 人依次上台与台上的人较量,胜者留台上,败者下台。最后一人(第 n 人)上台比赛后,留在台上的便为此 n 人中的王者。

n 个数中取大的算法与此相仿:先把第一个数放入变量 max 中,然后依次将 max 中的数与第 2 到第 n 个数比较,每次比较后将大者放入 max 中,直到全部比较完为止。形式化的描述如下:

step0:设置计数器 1⇒i。

step1:输入第一个数,将其存入 max。

step2:若 i≤n,则执行

　　step2.1:输入下一个数,存入 x 中,

　　step2.2:若 x>max,则 x⇒max,

　　step2.3:i+1⇒i,

　　step2.4:返回 step2;

　　否则执行 step3。

step3:输出 max。

上述算法的流程图如图 1.5 所示。

前面例举了 5 个简单的算法。其实在生活中处处存在着算法,如打算盘、做算术题、操作机器、演奏音乐、炒菜、做饭,打扑克、下棋、做广播操、打太极拳等,都离不开算法。一般地说,算法描述了一个过程,这个过程由一套准确的规则所规定,按这些规则所选定的操作序列,可以用有限的步骤提供特定类型问题的解答。更进一步,可以把算法的特征归纳为以下几点:

(1) 有穷性:一个算法应当包括有限个操作步骤,或者说它是由一个有穷的操作系列组成,而不应该是无限的。例如,创作一首小提琴曲谱可以称为小提琴算法。如果有人说他设计一首

永远演不完的小提琴曲谱，人们将无法演奏完这个曲子，因而它不能称为算法。

不仅如此，有穷性还常常理解为实际上能够容忍的合理限度。例如，执行一个计算机算法需要让计算机运行 10000 年，便是不能容忍的。

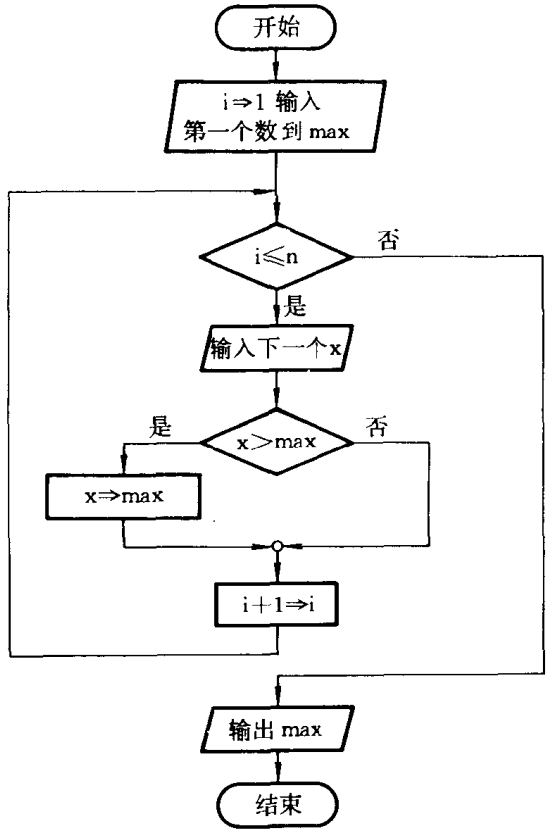


图 1.5

还需要强调的是，前面我们指出“算法描述了一个过程”，但并非所有的过程都能称为算法。因为过程可以不受有穷性的限制，它的操作可以是无穷尽的，例如“求$\sqrt{2}$的值”的运算过程便是无穷的。这种过程便不能称为算法。

(2) 确定性。算法中每一步的含义都应该是清楚无误的，不能模棱两可，也就是说不应该存在“歧义性”(即可作两种或两种以上不同的解释)。请读者分析下面这句话：张三对李四说他的孩子考上了大学。这句话就是“歧义性”的，可以理解为张三的孩子考上了大学，也可理解为李四的孩子考上了大学。因此在表示算法时要使用明确的文字或数字语言。

(3) 有效性。前面已经提到的一个算法必须遵循特定条件下的解题规则，即组成它的每一个操作都应该是在特定的解题规则中允许使用的、可以执行的，并且最后能得出确定的结果。例如，当算法中出现操作$\frac{a}{b}$，且 b=0 时，这个操作便是不可执行的，从而使含有它的算法不具有有效性。

除上述三点之外，一个算法应该有一个或多个输出。因为算法是问题模型的表现，或者说是解题方法、步骤的描述。算法的结果是解，解便是输出。对无解的问题应给出“无解”的信息。没有输出的算法是无意义的。算法不强调输入，有的算法可以没有输入，如打太极拳可以不必有任

何输入。

1.1.2 算法描述工具与流程图标准符号

算法是解题方法和步骤的精确描述。采用什么样的工具来描述算法，将直接影响算法的质量。就计算机程序设计而言，可以使用三类工具来描述算法：

(1) 自然语言。即人们日常使用的语言，如汉语、英语、日语等。上节的例子中基本上都是用汉语描述的算法。使用自然语言描述算法，人们比较习惯，容易接受，但它有以下缺点：

① 自然语言容易有“歧义性”，不太严格。例如前面说到的“张三对李四说他的孩子考上了大学”这句话，就不严格，具有歧义性。

② 用自然语言描述比较冗长。例如，“将 i 的值增 1”，不如表示成“i + 1⇒i”简洁。

为避免上述缺点，人们往往使用符号语言，如“i + 1⇒i”，或者在不致引起歧义的情况下使用简洁而有限定的自然语言来描述算法。

(2) 专用工具。为了更好地、准确地描述算法，人们创造了许多专用的算法工具。这些算法工具有图形的(如流程图、结构化流程图——N - S 图、PAD 图、IPO 图、Warnier 图等)，也有表格的(如判定表等)，还有使用代码符号的(如伪代码)。

为了便于使用和交流，采用专用算法描述工具时，应采用已有的标准符号。图 1.6 为国家标准局批准的 GB1526 - 89 中推荐使用的流程图标准符号，它与国际标准化组织公布的 ISO 5807 - 85 中的规定是一致的。

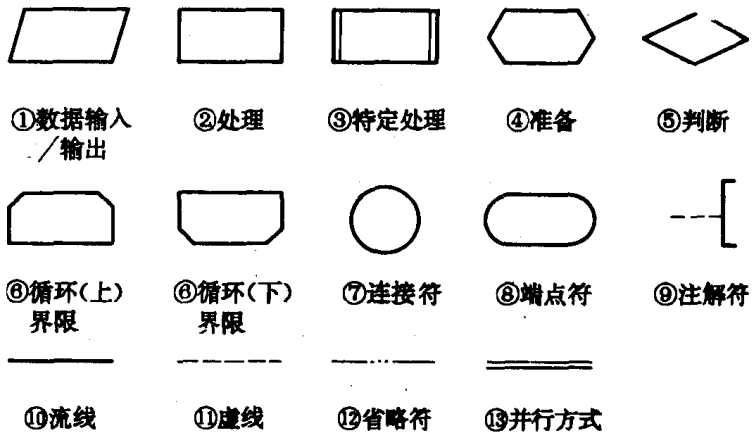


图 1.6 流程图标准符号

这里要重点介绍一下循环界限符。在例 1.3、1.5 中曾使用了由判断框构成的循环结构。为便于使用，标准符号中提供了专门的循环上、下限符。图 1.7 是图 1.3 所示算法的另两种描述形式。其中(a)称为“当”型循环，即当满足循环条件(这里是 n≤5)时，重复执行循环体(这里是 sum + n⇒sum，n + 1⇒n)，直到循环条件不再满足为止；(b) 称为“直到”型循环，即重复执行循环体，直到循环条件(这里是 n>5)满足为止。

(3) 计算机程序设计语言。计算机程序设计语言是可以让计算机直接或间接接受、理解和执行的算法描述工具。任何计算机算法最终都要用程序设计语言描述出来，才能在计算机上实现。本书将要系统介绍的 BASIC 语言就是一种计算机程序设计语言。

关于计算机程序设计语言的一般知识,将在§1.3中进一步介绍。

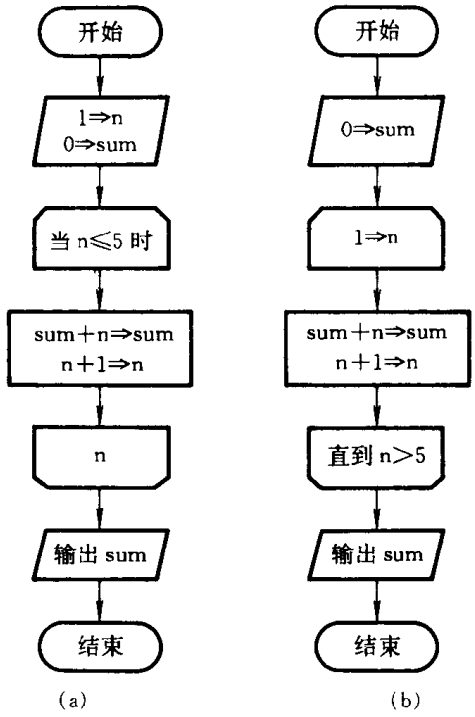


图 1.7

1.1.3 算法的三种基本结构与 N-S 图

顺序、选择、循环称为算法的三种基本结构。这三种基本结构具有如下特征:

(1) 单入口、单出口。即如把它们各看作一个整体,相当于一个处理框。

(2) 每一个框内的功能都有机会被执行。即对每一个框来说,应当有一条从入口到出口的路径通过它。图 1.8 中的 B 框就不具有这样的性质。

(3) 不包含死循环(即无休止的循环)。图 1.9 中的 A 框就不具有这样的性质。

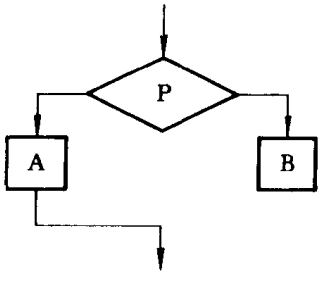


图 1.8

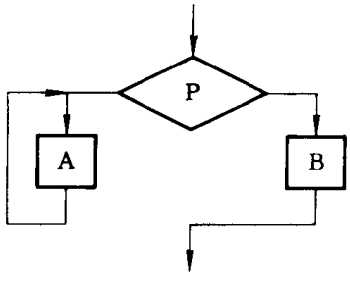


图 1.9

当然,还可以有其它的结构形式。事实上,早期的程序设计就没有三种基本算法结构的概念。当时由于受机器容量和速度的限制,人们以高效率为唯一的目标,而不注意算法的结构。同时,由于流程图中的流程线可以轻而易举地把流程从一个地方转移到另一个地方,更助长了算法设计者随心所欲地构筑各种结构的倾向。随着问题的规模和复杂程度的增加,算法结构会变得越来越混乱难以理解。图 1.10 是这种算法结构的示意图,其中每一根横线代表一个框的功能。

由于这种算法结构象“面条”一样(象一碗面条一样互相缠绕在一起,难以找到头和尾),故被称之为BS(Bowl of Spaghetti)型,也有的称其为耗子窝(rat's nest)型。

显然,这种BS型结构不利于阅读和交流,也不便于修改,从而使算法的可靠性和可维护性难以保证。为了提高算法的质量,必须限制箭头的滥用,使算法结构规范化。

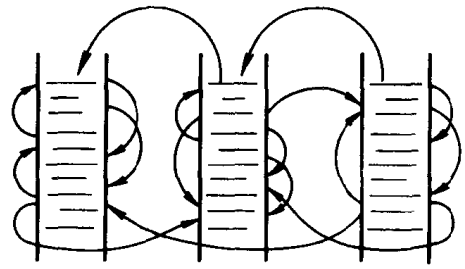
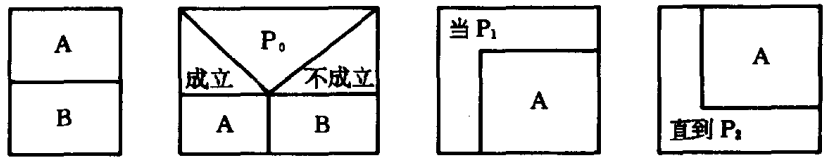


图 1.10

60年代末期,面对第一次“软件危机”,人们提出了一套完整的结构化程序设计的思想和方法。就算法结构而言,要求只用三种基本结构构筑算法。由于每一个基本结构都具有单入口单出口的特点,使算法的各单元之间关系简单,互相依赖性较少,它们是互相独立的“元部件”,人们能够象搭积木一样将它们搭成各种不同的大结构。整个算法看起来就象项链上的一串珍珠一样,由各基本结构顺序组成,结构清晰,从而可以使算法的质量显著提高。符合这一原则的算法称为结构化算法。其相应的流程图称为结构化流程图。

在强调清晰易理解的算法同时,人们发现,流程图中流程线太灵活,太方便,是造成算法结构不好的祸根。如果去掉了流程线就会迫使人们不得不按照结构化的要求去构筑算法。1973年美国学者I.Nassi和B.Shneiderman提出了一种新的流程图工具。由于他们二人的名字是以N和S开头的,而将这种流程图称为N-S图。N-S图完全去掉了在算法描述中引起麻烦的带箭头的流程线,全部算法写在一个大的矩形框内。规定了一些图形元素,以表示三种基本结构,每个元素用不同的框来表示。或者说,一个算法框是由一些代表基本结构的基本框象堆积木一样构造而成的。由于N-S图象一个多层的盒子,所以也称盒图(box diagram)。图1.11为它的三种基本结构图示意图。



(a) (b) (c) (d)

图 1.11 N-S图的三种基本结构

(1) 顺序结构:如图1.11(a)所示,其中A和B分别代表一个基本操作(例如加、减、乘、除运算,打印、赋值、……)。两个或多个矩形框顺序组成一个顺序结构。

(2) 选择结构:如图1.11(b)所示。它表示当P_0条件成立时执行A操作,P_0条件不成立时执行B操作。

(3) 循环结构:

(i) 当型循环:用图1.11(c)的形式表示。当型表示当P_1条件满足时执行A操作,然后再返回判断P_1条件是否满足,如满足再执行A,……,如此重复下去,直到P_1条件不满足为止。

(ii) 直到型循环:见图1.11(d)。它表示,先执行A操作,再判断P_2条件是否满足,如不满足则返回再执行A操作,如满足则不再继续执行循环。

图1.11(c)(d)中,“当P_1”和“直到P_2”,也可以写成“while P_1”和“until P_2”。有时也可以省写