21 世纪高等学校规划教材·计算机基础教育系列

新编C语言程序设计教程

林碧英 主 编
王默玉 吴耀红 王素琴 副主编

中国电力出版社
www.infopower.com.cn

21 世纪高等学校规划教材·计算机基础教育系列

新编C语言程序设计教程

林碧英 主 编
王默玉 吴耀红 王素琴 副主编

中国电力出版社
www.infopower.com.cn

内容提要

本书是作者多年来在讲授“C 语言程序设计”的基础上，总结多年的教学经验，对授课内容做了深入细致的研究后，整理而成的。全书的主要内容分为 12 章，知识覆盖面广，例题多而丰富。每章均配有多种题型的习题，供读者选用。

为便于教师授课和学生学习，本书在内容安排的顺序上做了较大调整，如将指针放到函数之前进行介绍。在指针一章中，要求掌握指针的概念以及用指针表示变量、字符、数组元素的方法，为理解函数的参数传递打下一定的理论基础，从而使 C 语言中的两大难点——指针和函数的教与学化难为易。

本书适合作为普通高校“C 语言程序设计”课程的教材，也可作为其他层次教育的教学用书，授课内容、例题和习题可根据实际情况进行选用。对从事软件开发的人员，本书也具有较高的参考价值。

图书在版编目（CIP）数据

新编 C 语言程序设计教程 / 林碧英主编. —北京：中国电力出版社，2006.2

21 世纪高等学校规划教材 • 计算机基础教育系列

ISBN 7-5083-3893-6

Ⅰ.新... Ⅱ.林... Ⅲ.C 语言－程序设计－高等学校－教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2005）第 156781 号

书　　名：新编 C 语言程序设计教程

出版发行：中国电力出版社

地　　址：北京市三里河路 6 号　　邮政编码：100044

电　　话：（010）68362602　　传　　真：（010）68316497，88383619

本书如有印装质量问题，我社负责退换

服务电话：（010）88515918（总机）　　传　　真：（010）88518169

E-mail：infopower@cepp.com.cn

印　　刷：北京同江印刷厂印刷

开本尺寸：185×260　　**印　　张**：21.5　　**字　　数**：525

书　　号：ISBN 7-5083-3893-6

版　　次：2006 年 2 月北京第 1 版

印　　次：2006 年 2 月第 1 次印刷

印　　数：0001—4000 册

定　　价：29.80 元

前　言

高级程序设计语言已是当今大学生的必修课程。C 语言是面向过程的高级程序设计语言，它不仅具有高级语言的特点，还能够实现低级语言所能完成的操作。利用它不仅可以编写应用软件，还可以编写系统软件。因此，现在各个高等院校几乎所有的专业，都开设了“C 程序设计”这门课。为了使教师有一本较好的教材，学生有一本易学易懂的课本，软件开发人员有一本较全面的参考书，为此，我们特组织一批具有多年本科教学经验的教师编写了这本《新编 C 语言程序设计教程》。

我们在编写本书的过程中，根据目前各个学校学时压缩导致的时间紧、任务重的特点，从内容的选择、问题的引入以及问题的解决办法方面都充分体现了由浅入深、循序渐进、通俗易懂的原则。各个章节都配有大量例题，例题的编写渗透了教师多年的教学经验，大多数例题都配有分析过程、运行结果，很多例题还给出了多种算法，使学生全面掌握语法结构和程序设计方法。每章的最后附有各种类型的习题，知识点覆盖全面，从不同的角度测试学生掌握的程度，与国家计算机二级等级考试要求紧密相关。总而言之，我们的目的是使学生通过本书的学习、课堂面授和课后复习，掌握 C 语言的语法规则和基本的编程方法，具有一定的程序设计能力。

本书的内容分为 12 章，其中，第 1 章、第 2 章和第 3 章由王素琴编写，第 4 章、第 5 章、第 6 章和第 10 章由王默玉编写，第 7 章、第 8 章和第 9 章由吴耀红编写，第 11 章、第 12 章由林碧英编写，最后由林碧英审阅定稿。

由于编写时间比较仓促，书中难免还存在不足之处。欢迎广大读者批评指正。

《新编 C 语言程序设计教程》编写组

2005 年 12 月

目　录

第 1 章　程序设计概述

随着计算机科学与技术的迅速发展，计算机的应用领域越来越广泛，不断涌现出大量的适用于各种应用领域的计算机语言。对于初学者来讲，首先必须弄清什么是计算机语言，计算机语言的分类以及如何利用计算机语言进行程序设计来解决实际的问题。

1.1　计算机语言及程序设计

1.1.1　计算机语言

过去，一提到语言这个词，人们想到的是汉语、英语等，它们是人与人之间进行交流不可缺少的工具，我们称之为自然语言。现在，计算机已经介入到我们工作和生活的各个方面，我们除了要和人进行交流以外，还必须和计算机进行交流。那么，用什么方式和计算机进行交流呢？人们自然想到了最古老也最方便的形式——语言。人与人之间的交流用的是双方都能理解的自然语言，同样，人和计算机交流也要用人和计算机都能够理解和接受的语言，这就是计算机语言。人们用自然语言讲述和书写，目的是向其他人传递信息。我们使用计算机语言把我们的意图表达给计算机，目的是使用计算机。可以说计算机语言架起了人和计算机之间沟通的桥梁。

作为沟通工具的计算机语言应该兼顾人和计算机双方的特点。但实际上，目前的计算机语言更多地是根据计算机的特点而编制的（当然，以后会越来越向人的特点靠拢）。它没有自然语言那么丰富多彩，只是有限规则的集合，所以它简单易学。但是，因为它是根据计算机的特点编制的，所以在人机交流中无法进行意会和言传，而更多地表现出“说一不二”的特点，体现了“规则”的严谨性。例如该是“;”的地方不能写成“.”，该写“a”的地方不能写成“A”等。这使人们在开始学习计算机语言时会有些不习惯，不过只要认识到计算机语言的特点，注意学习方法，把必须的严谨和恰当的灵活结合起来，一切都会变得得心应手。

1.1.2　计算机语言的发展

自计算机问世以来，先后产生了数百种不同的计算机语言，这直接推动了计算机的应用和普及。按照计算机语言的发展历史，它们大致可分为 3 类：机器语言、汇编语言和高级语言。

1．机器语言

机器语言是以二进制指令代码表示的指令集合，是计算机能直接识别和执行的语言。用机器语言编写的程序运行速度快，占用内存少，但缺点是面向机器，通用性差。用机器语言编程序时不直观，生产率低，容易出错。现在，除了计算机生产厂家的专业人员外，绝大多数程序员已经不再去学习机器语言了。

2. 汇编语言

为了克服机器语言难读、难编和易出错的缺点，人们用与指令代码实际含义相近的英文缩写词、字母或数字等符号来取代指令代码，于是就产生了汇编语言。汇编语言是一种用助记符表示指令的面向机器的计算机语言。由于采用了助记符来编写程序，比用机器语言的二进制代码编程要方便一些，在一定程度上简化了编程过程。用汇编语言编写的程序称为源程序，计算机不能直接识别和执行，必须翻译成二进制形式的目标程序后才能执行，这一翻译工作由“汇编程序”来完成，翻译的过程称为“汇编”。

汇编语言主要用来开发系统软件和过程控制软件，其目标程序占用内存空间少，运行速度快，可实现一般语言难以实现的功能。但汇编语言仍然是面向机器的语言，通用性差。

汇编语言和机器语言都依赖于具体的机器，统称为“低级语言”。

3. 高级语言

机器语言和汇编语言同人们习惯使用的自然语言差别很大，不容易学习和掌握。它们还要求使用者必须对计算机硬件结构及其工作原理都十分熟悉，这对非计算机专业人员来说是难以做到的。所以人们一直在寻求与自然语言相近的通用易学的计算机语言，这种计算机语言就称为高级语言。高级语言是完全符号化的语言，用类似自然语言的形式描述问题的处理过程，用数学表达式的形式描述数据的计算过程，易于被人们理解和接受。较早产生的典型的高级语言有 BASIC、FORTRAN、PASCAL 和 C 等。这些语言只要求人们向计算机描述问题的求解过程，而不关心计算机的内部结构，所以又称为“面向过程的语言”。

用高级语言编写的源程序也不能被计算机直接识别和执行，必须将它翻译成二进制形式的目标程序后才能执行。翻译方式一般分为编译和解释两种。

（1）编译方式：将源程序一次性地翻译成目标程序，然后再执行该目标程序。完成翻译工作的程序叫“编译程序”，翻译的过程称为“编译”。

（2）解释方式：将源程序逐句地翻译，翻译一句执行一句。完成翻译工作的程序叫“解释程序”。

这两种翻译方式各有所长，编译方式执行速度快，解释方式执行速度慢，但占用内存少。

随着计算机技术的迅猛发展，从 20 世纪 80 年代以来，出现了众多的面向对象的语言和面向应用的语言（第四代语言）。但是，“面向过程”仍然是所有程序设计的基础。作为初学者，首先还是应该掌握一门面向过程的语言。在本书中，我们将以面向过程的 C 语言为基础，详细介绍程序设计的基本概念和基本方法。

1.1.3 程序与程序设计

在日常工作中，我们可以使用计算机进行数学运算、编写文档、绘制表格和收发邮件等。也就是说，尽管计算机本身只是一种用现代化方式批量生产出来的通用电子产品，但是在计算机上运行不同的程序，就能使它解决不同的问题，完成各种各样的任务。今天，计算机之所以能够产生如此大的影响，其原因不仅在于人们发明了机器本身，更重要的是人们为计算机开发出了不计其数的能够指挥计算机完成各种各样工作的程序，正是这些功能丰富的程序给了计算机无穷的生命力。

程序是用计算机语言编写的，因此，计算机语言通常又被称为“程序设计语言”。确切地说，所谓程序，就是用计算机语言对所要解决的问题中的数据以及处理问题的方法和步骤

做出的完整而准确的描述，而得到这个描述的过程就称为程序设计。对数据的描述就是指明数据结构形式，对处理的方法和步骤的描述就是下一节要讨论的算法问题。可见，数据结构与算法是程序设计过程中密切相关的两个方面。因此，著名的计算机科学家 Nikiklaus Wirth 提出了一个公式：

数据结构 + 算法 = 程序

进行程序设计时，一般包含以下 4 个步骤。

（1）分析问题，建立数学模型。使用计算机解决具体问题时，首先要对问题进行充分的分析，确定已知条件是什么，要得到什么结果，以及解决问题的详细步骤。将解题过程归纳为一系列的数学表达式，建立各种量之间的关系。需要注意的是，有许多问题的数学模型非常简单，以致于我们没有感觉到需要建立模型。但有更多的问题需要进行认真分析，构造出数学模型。模型的对与错、好与坏，在很大程度上决定了程序的正确性和复杂性。

（2）确定数据结构和算法。根据数学模型以及需要的输入输出数据，确定存放数据的数据结构，选择合适的算法加以实现。算法的概念将在下一节中介绍。

（3）编制程序。根据已经确定的数据结构和算法，选择合适的程序设计语言，将解决方案正确地描述出来，形成程序。

（4）调试程序。根据选定的语言平台，通过编译器找出程序中存在的错误，经过分析加以改正，这个过程可能要反复多次，直到程序能够正确地运行为止。

1.1.4 结构化程序设计

长时间以来，人们一直认为程序是给机器执行的，只要程序没有错误，能够输出正确的结果就可以了。但随着计算机技术的迅速发展，程序的规模越来越大，人们发现读程序的时间可能比写程序的时间要长得多。不仅程序员需要读程序，测试人员和维护人员也需要读程序。如果程序很难读懂，也就无法进行相应的测试和维护。因此，衡量程序的质量不仅要看它的逻辑是否正确，性能是否满足要求，还要看它是否容易阅读和理解，即程序的清晰性是必须要考虑的。结构化程序设计是实现程序清晰易懂的关键技术。它的基本思想是规定出几种基本结构，然后由这些基本结构按一定规律组成程序，如同用一些基本构件搭建房屋一样。整个程序的结构是由各个基本结构自上而下顺序排列组成的。

结构化程序设计的概念最早由 E.W.Dijkstra 于 1965 年提出。1966 年 Bohra 和 Jacopini 进一步证明了，只用 3 种基本控制结构就能实现任何单入口、单出口的程序。这 3 种基本控制结构分别是顺序结构、选择结构和循环结构。

1．顺序结构

顺序结构表示程序中的各个操作是按照它们出现的先后顺序来执行的，如图 1.1 所示。图中的 A 和 B 表示两个操作，它们是顺序执行的，先执行 A，然后执行 B。顺序结构是一种最简单的基本结构。

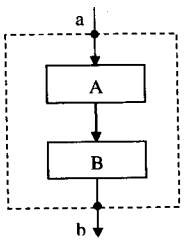


图 1.1 顺序结构

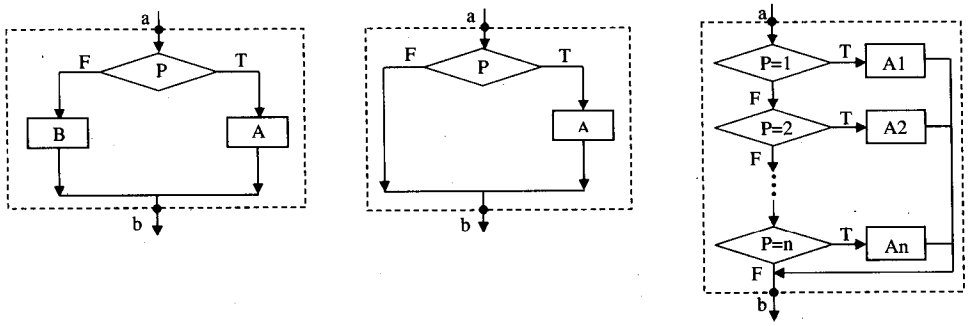
事实上，不论程序中包含了什么样的结构，程序的总流程都是顺序结构。

2．选择结构

选择结构也称为分支结构，如图 1.2 所示。先判断给定的条件是否成立，再决定执行哪种操作。选择结构分为两种：两个分支的选择结构，如图 1.2（a）和（b）所示；多分支的

选择结构，如图 1.2（c）所示。

两个分支的选择结构：首先判断条件 P 是否成立，如果成立，则执行 A 操作，否则，执行 B 操作。这里，条件 P 是否成立用英文字母来表示：T 表示成立，F 表示不成立。注意，无论条件 P 是否成立，只能执行 A 或 B 其中之一，执行后经出口点 b，脱离本选择结构，不可能既执行 A 又执行 B。A 和 B 两个操作中可以有一个为空，如图 1.2（b）所示。



（a）　　（b）　　（c）

图 1.2　选择结构

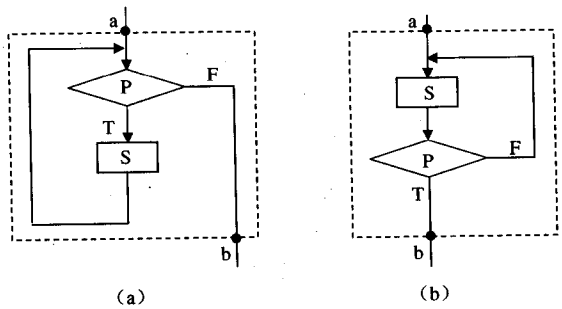
多分支选择结构是指选择结构中有多种可能的情况，有多个可执行的分支。程序执行哪个分支根据条件 P 的取值来决定，不论执行了哪一个分支，执行后都直接到达出口 b 处，接着执行下面的其他语句。如果所有分支的条件都不满足，则直接到达出口处。

3．循环结构

循环结构又称“重复结构”，即反复执行某个或某些操作。循环结构分为两种：当型循环结构和直到型循环结构。

当型循环结构如图 1.3（a）所示，它的功能是：先判断条件 P 是否成立，如果成立则执行操作 S；执行 S 后，再判断 P 是否成立，如果仍然成立，再执行 S；如此反复执行操作 S，直到条件 P 不成立为止，退出循环结构。因为是“当条件满足时执行循环”，即先判断后执行，所以称为当型循环。其中，P 称为循环条件，S 称为循环体（一组重复执行的操作）。

直到型循环结构如图 1.3（b）所示。它的功能是：先执行循环体 S，然后判断条件 P 是否成立，如果 P 不成立则再执行 S；执行后再去判断条件 P 是否成立，如果 P 仍然不成立，又执行 S；如此反复执行操作 S，直到条件 P 成立为止，退出循环结构。因为是“直到条件为真时停止循环”，所以称为直到型循环。同当型循环一样，P 称为循环条件，S 称为循环体。



（a）　　（b）

图 1.3　循环结构

在使用循环结构时，最主要的是确定什么情况下执行循环（即循环条件），哪些操作需要重复执行（即循环体）。

以上介绍的这些基本结构可以顺序组合，也可以嵌套（一个基本结构中可以包含另一个或多个基本结构）。一个程序无论功能多么强大、逻辑多么复杂，都可以用这 3 种基本控制结构嵌套组合而成。依据结构化思想编写出来的程序称为“结构化程序”。结构化程序只有一个入口和一个出口，整个程序具有清晰的线索，容易理解、容易维护、可靠性高。

1.2 算　　法

1.2.1 算法的概念

无论做什么事情，都有一定的步骤，必须按照步骤循序渐进地进行。例如大学的期末考试，考试开始前，监考教师先到考试办公室取试卷，然后到指定教室组织学生进行考试，考试结束后，监考教师收卷，经任课教师阅卷后给出成绩。这些步骤缺一不可，顺序错了有时会产生严重的后果。因此，做任何事情之前，都必须考虑好实施的步骤，然后按部就班地进行，才能避免产生错误和混乱。

广义地说，为解决一个问题而采取的方法和步骤，就称为算法。

当然，同一个问题，可以有多种解决方法或操作步骤。例如：从北京到大连去旅游，可以选择不同的交通工具，如火车、汽车或飞机等。乘坐不同的交通工具，所花费的时间和金钱是大不相同的。人们可以根据自己的实际情况选择最合适的交通工具。再如，求 1+2+3+…+100，可以先算 1+2，再加 3，再加 4，一直加到 100，一共要做 99 次加法运算，得到结果 5050；也可以用等差数列的求和公式（1+100）×100/2，同样得到结果 5050。从中可以看出，同一个问题，可以有多种算法，而算法不同，效率也不同。因此，我们不仅要保证算法的正确性，还要考虑算法的优劣，根据实际情况，选择合适的算法。

作为一本 C 语言的教材，本书涉及的算法都是计算机算法。所谓计算机算法，是指用计算机解决一个实际问题的方法和步骤。计算机算法分为两大类：数值运算算法和非数值运算算法。数值运算的目的是求解数值，如求方程的根、图形的面积等；非数值运算包含的范围很广，如人员管理、网上购物等。最初，计算机主要应用在数值运算领域，由于数值运算有现成的模型，可以运用数值分析方法，因此，人们对数值运算算法的研究非常深入，各种数值运算都有比较成熟的算法可供选用。后来，计算机更多地用于解决非数值运算方面的问题。由于非数值运算的种类繁多、要求各异，难以规范化，因此，只对一些典型的算法（如排序算法）做比较深入的研究。至于其他的非数值运算问题，使用者可以参考已有的算法，自己来设计解决问题的特定算法。

1.2.2 算法的特征

算法具有以下基本特征。

（1）有穷性：一个算法包含的操作步骤应该是有限的，而不能是无限的，要在执行有限个操作步骤后终止。

事实上，“有穷性”往往指在合理的范围内，如果让计算机执行一个历时 100 年才结束

的算法，这虽然是有穷的，但超过了合理的限度，人们也不把它视作有效的算法。

（2）确定性：算法中每个步骤的含义都必须是确定的，不能是模糊的、有歧义的。例如有这样一句话："当 i 为 0，j 为 0 时加 1"，通过这句话无法判断要给哪个变量加 1，可能是 i，也可能是 j，或者是其他变量，因此无法执行。

（3）有效性：算法中每一个操作都应该能有效地执行，并得到确定的结果。一个不可执行的操作是无效的。例如，一个数被 0 除的操作就是无效的，应当避免这种操作。

（4）有零个或多个输入：输入是指在执行算法时所需要的初始数据。输入数据的多少取决于特定的问题。例如，求 N! 就需要先输入 N 的值；求一元二次方程的根，需要输入 3 个系数 a、b 和 c 的值；求前 100 个自然数的和则不需要输入数据。

（5）有一个或多个输出：算法的目的就是要得到问题的"解"，"解"就是输出。在一个完整的算法中至少应有一个输出。例如，求 N!，要输出 N! 的值；求一元二次方程的根，要输出该方程的根；求前 100 个自然数的和，要输出求和的结果。没有输出的算法是没有意义的。

1.2.3 算法的描述

在分析、设计算法的过程中，我们需要将算法表达出来。表达算法的工具很多，常用的有自然语言、程序流程图、N-S 图和伪代码等。

1. 用自然语言表示算法

自然语言就是人们日常使用的语言，如汉语、英语等。用自然语言描述算法具有通俗易懂的优点。例 1.1 就是用自然语言描述了求前 10 个自然数之和的算法。

【例 1.1】求 $\sum_{i=1}^{10} i$，即 1+2+3+4+5+6+7+8+9+10 的值。

可以用最直接的方法进行计算。

步骤 1：先求 1+2，得到结果 3。

步骤 2：将步骤 1 得到的和加上 3，得到结果 6。

步骤 3：将 6 再加上 4，得 10。

步骤 4：将 10 再加上 5，得 15。

……

步骤 8：将上步得到的结果 36 再加上 9，得 45。

步骤 9：将 45 再加上 10，得 55。

这样的算法虽然正确，但太繁琐了。如果要求 $\sum_{i=1}^{100} i$ 的和，则要写 99 个步骤，显然是不可取的。而且每次都直接使用上一步的计算结果，不太方便，应该找一个更通用的表示方法。

可以设两个变量，代表两个加数，直接将每一步求得的和放在一个加数变量中。现设 sum 和 i 分别代表两个加数，每一步求得的和也存放在 sum 中。用循环算法来求结果，改进的算法如下所示。

步骤 1：0→sum（含义是：使 sum 的值为 0）；

步骤 2：1→i（含义是：使 i 的值为 1，以此类推，以后将不再注明）；

步骤 3：求 sum+i 的和，结果仍然放在变量 sum 中，可表示为 sum+i→sum；

步骤 4：使 i 的值加 1，即 i+1→i；

步骤 5：如果 i>10，算法结束，否则，返回重新执行步骤 3、步骤 4 和步骤 5。

可以看出，步骤 3、4、5 组成一个循环，在实现算法时，要反复多次执行这 3 个步骤，直到执行步骤 5 时，经过判断，i 已经超过规定的数值 10 而不返回步骤 3 为止，算法结束。此时变量 sum 的值就是所求的结果。

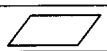
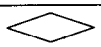
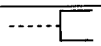
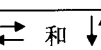
用这种方法表示的算法具有较强的通用性和灵活性。如果要计算 $\sum_{i=1}^{100} i$，只需要将步骤 5 中的 i >10 改成 i >100 即可。

用自然语言描述算法虽然通俗易懂，但用计算机语言实现该算法时常常需要做较大改动，而且往往文字冗长，表示的含义也不太严格，容易产生歧义性（即可能被理解成多种含义）。例如，有这样一句话：“张三对李四说他取得了好成绩”，通过这句话无法判断是张三取得了好成绩，还是李四取得了好成绩。因此，除了很简单的问题，一般不用自然语言来表示算法。

2．用程序流程图表示算法

程序流程图又称为程序框图，是最早提出的用图形表示算法的工具。它采用一组特定的图形、流程线以及文字说明来表示算法中的基本操作和控制流程，具有形象直观、简单易懂等特点。美国国家标准化协会 ANSI（American National Standard Institute）规定了一些常用的图形符号，这些符号已为世界各国的广大程序设计工作者普遍接受和采用，表 1.1 列出了其中的一部分符号。

表 1.1　程序流程图的常用图形符号

符号名称	符　号	功　能
起止框		表示算法的开始和结束
输入/输出框		表示算法的输入/输出操作，框内填写需输入或输出的各项
处理框		表示算法中的各种处理操作，框内填写处理说明或算式
判断框		表示算法中的条件判断操作，框内填写判断条件
注释框		表示算法中某操作的说明信息，框内填写文字说明
流程线	 和	表示算法的执行方向
连接点		表示流程图的延续

（1）起止框：表示算法的开始或结束。每个程序流程图中都必须有而且只能有一个开始框和一个结束框，开始框只能有一个出口，没有入口，结束框只有一个入口，没有出口。

（2）输入输出框：表示算法的输入数据或输出结果，可以是一项或多项数据，多项之间用逗号分隔。输入/输出框只能有一个入口和一个出口。

（3）处理框：表示程序中的各种操作，如计算、赋值等。处理框内填写处理说明或具体的算式。可在一个处理框内描述多个相关的处理。一个处理框只能有一个入口和一个出口。

（4）流程线：用箭头表示，它指出各框的执行顺序。流程线箭头的方向就是算法执行的方向。事实上，流程线非常灵活，它可以到达流程图的任意位置。没有限制的灵活就是随意，程序设计的随意性是一定要杜绝的，因为它会使软件的可读性、可维护性大大降低。所以使

用程序流程图表示算法时，流程线不能随意地流转，必须遵守结构化程序设计的要求。

（5）判断框：用菱形表示，它是对一个给定的条件进行判断，根据条件是否成立决定如何进行其后的操作。

（6）注释框：注释框中包含的内容称为注释。注释是对整个算法或其中部分操作的解释说明。正确的注释有助于对程序的理解。

（7）连接点（小圆圈）：用于将画在不同地方的流程线连接起来，避免流程线的交叉或过长，使流程图更清晰。

用程序流程图表示算法时，必须考虑结构化程序设计的要求。图 1.1～图 1.3 中显示了在程序流程图中，基本控制结构的表示方法。

【例 1.2】 将例 1.1 中求 $\sum_{i=1}^{10} i$ 的算法用程序流程图表示，见图 1.4。

用流程图表示算法形象、直观、易于理解。但流程图也存在一些严重的缺点，如所使用的符号不够规范，常常使用一些习惯性用法，特别是对流程线的使用没有严格的限制，可以不受约束，随意转移控制。所以，使用程序流程图描述的算法不一定满足结构化程序设计的要求。许多人建议停止使用它，但至今仍在广泛使用，不过总的趋势是越来越多的人不再使用程序流程图了。

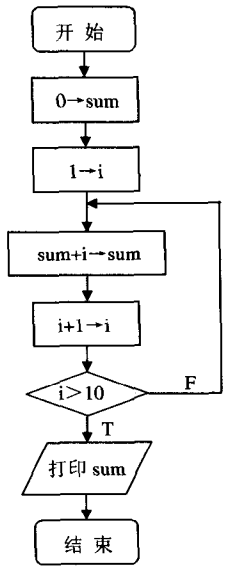


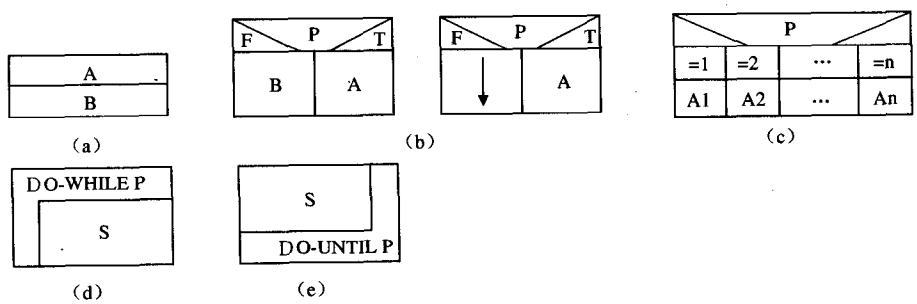
图 1.4 求 $\sum_{i=1}^{10} i$ 算法的程序流程图

3. 用 N-S 图表示算法

既然用基本控制结构可以表示任何复杂的算法，那么基本控制结构之间的流程线就属多余了。1973 年，Nassi 和 Shneiderman 提出了一种结构化的流程图：N-S 流程图。在这种流程图中，完全去掉了带箭头的流程线。它的基本单元是矩形框，程序的 3 种基本控制结构分别用不同的矩形框表示。在矩形框内还可以包含其他从属于它的矩形框，即 N-S 图由矩形框组合嵌套而成，因此又称为盒图。图 1.5 列出了 N-S 图中基本控制结构的画法。

图 1.5（a）表示顺序结构，先执行 A，再执行 B。

图 1.5（b）表示两个分支的选择结构：首先判断条件 P 是否成立，如果成立，则执行 A 操作，否则，执行 B 操作。若是空操作，则用箭头“↓”表示。



（a）顺序结构 （b）两分支选择结构 （c）多分支选择结构 （d）当型循环结构 （e）直到型循环结构

图 1.5 N-S 图中基本控制结构的画法

图 1.5（c）表示多分支选择结构：根据条件 P 的取值，执行相应框内的操作。

图 1.5（d）和（e）表示两种类型的循环结构：P 是循环条件，S 是循环体。其中，图 1.5（d）表示当型循环结构，当条件 P 成立时，反复执行 S 操作，直到条件 P 不成立为止；图 1.5（e）表示直到型循环结构，先执行 S 操作，然后判断条件 P 是否成立，如果不成立则再执行 S 操作，执行后再去判断条件 P 是否成立……如此反复，直到条件 P 成立为止。

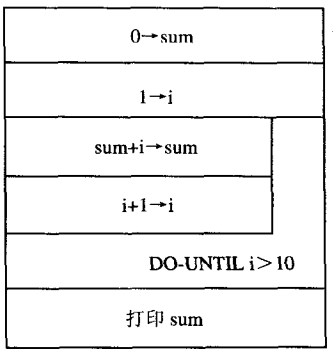


图 1.6　用 N-S 图表示求 $\sum_{i=1}^{10} i$ 的算法

【例 1.3】 将例 1.1 中求 $\sum_{i=1}^{10} i$ 的算法用 N-S 图表示，如图 1.6 所示。

用 N-S 图描述的算法杜绝了流程的无条件转移，结构清晰，容易理解，完全符合结构化程序设计的要求，在程序设计中得到了广泛的应用。

4．用伪代码表示算法

使用程序流程图和 N-S 图表示算法，清晰易懂，但如果要修改，工作量会很大。在设计算法的过程中，通常需要反复修改，不断完善。因此，流程图适合表示一个最终的算法，而在设计算法过程中使用却不太理想。为了设计算法时方便，经常使用伪代码作为描述工具。

自然语言通俗易懂，但容易产生歧义，计算机语言不会产生歧义，但是却需要深入地学习才能够掌握。伪代码集两者之所长，使用介于自然语言和计算机语言之间的文字和符号来描述算法，既能够通俗易懂又减少了歧义性。语言的正文用基本控制结构进行分割，具体操作用自然语言来表示。它如同一篇文章，自上而下逐行写下来，每一行（或几行）表示一个操作。

【例 1.4】 将例 1.1 中求 $\sum_{i=1}^{10} i$ 的算法用伪代码表示。

```
sum 赋初值为 0
for(i=1;i<=10;i++)
  使 sum=sum+i
打印 sum 的值
```

从上例可以看出，伪代码书写格式比较自由，容易修改，但用伪代码表示的算法没有用流程图表示的算法直观。

以上介绍了几种常用的算法表示工具，在程序设计中可以根据需要或习惯来选用。在本书中，主要采用 N-S 图来表示算法。

5．用计算机语言实现算法

要完成一项工作，不仅要设计算法，还要实现算法。例如：我们已经清楚了求 $\sum_{i=1}^{10} i$ 的算法，但还没有求出确切的结果。只有实现了算法，才能得到运算结果。

实现算法的方法很多，如可以用心算、笔算、算盘或计算器求出结果，但我们的任务是用计算机实现算法。计算机是无法识别流程图和伪代码的，只有计算机语言编写的程序才能被计算机编译执行。因此，描述一个算法后，还要将它转换成相应的计算机语言程序。