THOMSON

21世纪 高等院校计算机系列教材

# 编程逻辑与结构化程序设计（原书第三版）

# Logic and Structured Design for Computer Programmers (Third Edition)

[美] Harold J.Rood 著

杜大鹏 龚小平 管英强 钱富达 等译

杜国梁 审校

中国水利水电出版社

www.waterpub.com.cn

21世纪高等院校计算机系列教材

# 编程逻辑与结构化程序设计

（原书第三版）

[美] Harold J.Rood 著

杜大鹏 龚小平 管英强 钱富达 等译

杜国梁 审校

中国水利水电出版社

## 内 容 提 要

本书是学习计算机编程语言的预备课程教科书。本书使用简明易懂的语言和丰富的示例讲解并图示设计结构化程序所需的工具和算法逻辑方面的基本知识。设计工具包括结构化流程图、Warnier 框图、伪代码和 Nassi-Shneiderman 框图；算法逻辑知识包括集合论和真值函数分析方法。本书还包括有关数组和文件处理方面的内容。由于本书并不涉及特定编程语言的细节，因而其内容适合于学习各种计算机编程语言的读者。

本书可作为高等院校计算机及其相关专业编程课程的先行教材。对于那些有志于学习计算机编程语言的其他读者也是很好的参考读物。

Harold J.Rood, Logic and Structured Design for Computer Programmers (Third Edition).

EISBN: 0-534-37386-0



**北京市版权局著作权合同登记号：图字 01-2004-3011**

**图书在版编目（CIP）数据**

编程逻辑与结构化程序设计：第 3 版 /（美）鲁德（Rood, H. J.）著；杜大鹏等译. —北京：中国水利水电出版社，2004.6

（21 世纪高等院校计算机系列教材）

书名原文：Logic and Structured Design for Computer Programmers

ISBN 7-5084-2126-4

Ⅰ. 编… Ⅱ. ①鲁…②杜… Ⅲ. 结构化程序设计—程序设计：逻辑设计—高等学校—教材 Ⅳ. TP311.11

中国版本图书馆 CIP 数据核字（2004）第 047515 号

| | |
|---|---|
| 书　　名 | **编程逻辑与结构化程序设计** |
| 作　　者 | [美] Harold J.Rood　著 |
| 译　　者 | 杜大鹏　龚小平　管英强　钱富达　等译 |
| 审　　校 | 杜国梁 |
| 出版　发行 | 中国水利水电出版社（北京市三里河路 6 号　100044）<br>网址：www.waterpub.com.cn<br>E-mail：mchannel@263.net（万水）<br>sales@waterpub.com.cn<br>电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水） |
| 经　　售 | 全国各地新华书店和相关出版物销售网点 |
| 排　　版 | 北京万水电子信息有限公司 |
| 印　　刷 | 北京蓝空印刷厂 |
| 规　　格 | 787mm×1092mm　16 开本　24.25 印张　577 千字 |
| 版　　次 | 2004 年 6 月第 1 版　2004 年 6 月第 1 次印刷 |
| 印　　数 | 0001—5000 册 |
| 定　　价 | 34.00 元 |

# 译者序

计算机程序是计算机的灵魂，离开计算机程序，计算机将一事无成。计算机专业软件开发人员编写了计算机操作系统、通用应用软件供一般人使用。这些专业人员当然离不开计算机编程。不过计算机编程知识对于使用计算机的不少人来说，也是必要的。操作系统和通用软件不能解决所有问题，对于具体工作有时还需要编写解决具体问题的应用程序。因而，除了有志于从事计算机软件开发的人员必然要学习编程知识之外，一般人要想成为计算机应用的高手，学习一些必要的编程知识是取得成功的事半功倍的方法。

学习计算机编程知识并不难。在掌握具体使用的计算机编程语言之前，需要理解一些并不依赖于具体编程语言的计算机编程方面的通用知识。本书提供了这方面的内容，对于那些想要掌握一些编程知识的人来说，本书是很好的入门教材。

由于计算机科学是在以英语为母语的国家中发展起来的，各种编程语言甚至计算机操作命令都与英语有千丝万缕的联系。学习计算机编程知识及具体的编程语言离不开英语。当然英语越好，对学习计算机越有利。在翻译本书时，译者曾想尽量翻译得“贴切”一些，以便于没有什么英语基础的人阅读，但仔细想一想，这种努力也许是不必要的。比如说，READ（读），恐怕在任何编程语言中都有这一命令，如果我们将其译为“读”，倒显得有点不伦不类，因为在任何编程语言中也看不到“读”这个指令或语句。计算机程序毕竟不是英语，试想即使真的有人一点英语不懂而又想学习计算机编程，那也没有太大关系，只要会查字典，一般来说，已经能够解决绝大多数英语拦路虎问题了，更何况现在我国绝大多数中学都开设了英语课，大学无论是本科还是专科，英语可能都是必修课。有鉴于此，我们在翻译时保留了作为指令、语句等的英语表示，而且示例图也未加翻译，这可能不会对理解本书造成障碍。

本书是多人努力的成果。杜大鹏、龚小平、管英强、钱富达等参加了翻译工作，其中，杜大鹏翻译了第 1 章到第 3 章及附录，龚小平翻译了第 4 章到第 7 章，管英强翻译了第 8 章和第 9 章，钱富达翻译了第 10 章和第 11 章。全书由杜国梁审校并统稿。参加本书其他工作（录入、打印、校对等）的有魏全宇、梁国珍、杜墨、梁大伟、马相生、刘发来、董明、迟春和杨天华等。没有这些人的共同努力，要想顺利完成本书的翻译工作是不可能的，在此对他们表示感谢。

译者　二〇〇四年一月二十九日
于防化指挥工程学院

# 前　言

《编程逻辑与结构化程序设计》原书第三版是作为计算机编程算法逻辑的入门书籍而编写的。本书的潜在读者为那些打算学习编程但还没有开始学习编程的人，特别是那些没有什么数学或者算法知识背景的人。本书避免引用特定的编程语言，将算法问题与句法问题分离开来，以便让读者将精力集中于问题的算法上，而不必考虑标点符号、列标号等具体编程语言上的问题。本书叙述许多编程语言共同存在的逻辑问题，并提供许多编程教科书和课程的预备性背景知识。本书并不打算推荐或补充任何特定的编程方法或风格；本书提供的是各种算法逻辑和设计工具，同时提出许多不同的应用建议。

本书对于那些通过示例学习以及通过解释学习的学生都是有用的。编程示例是为说明特定的观点而不是作为范例而设计的。程序示例的许多方面已经特意地加以简化，从而使读者能够掌握其他方面的细节。由于编程的详情对于不同的语言差别极大，因而有些方面进行了刻意的简化。换句话说，示例是按照教学工具的有用性加以选择的；至于实际情况下的可用性和优美与否则是次要的考虑方面。（当然，这些示例也不打算引用任何实际的人或情况。）

在本书中，设计工具包括流程图、结构流程图、Warnier 框图、Nassi-Shneiderman 框图和伪代码。每一种设计工具既可用于系统流程图，也可用于系统分析与设计中。其中例外的是，用较短的章节来叙述系统流程图，不同的工具都是作为程序设计工具来加以叙述的，而不考虑其系统设计能力。

本书提及的工具及技术是以相对单纯的学生可以理解的方式叙述的，这些学生将随着学习编程语言和系统的细节的学习以及未来事业的发展逐步提高其编程能力。本书并不想对程序设计有决定性的影响。本书也并不打算成为业界的标准推荐或是在涉及所提到的编程工具时作为决定性的说明。

本书从将编程逻辑作为一种工具流程图开始。第 1 章使用非结构化的流程图讨论简单循环、计数器以及累加器，使读者将精力集中于所讨论的内容上，而不必关心结构。第 2 章介绍结构化流程图，第 7 章介绍 Warnier 框图，而第 8 章介绍伪代码和 Nassi-Shneiderman 框图。这几章中的每一章都让读者使用这些结构化的程序设计方法重复前面的练习。到第 8 章结束时，读者应该已经较好地理解了 4 种不同的设计和描述工具：结构化流程图、伪代码、Nassi-Shneiderman 框图和 Warnier 框图。

第 3 章介绍控制中断程序。第 9 章讲述一维和二维数组并为设计流程图和 Warnier 框图提供适度复杂的（在入门级别上说）问题。第 10 章详细地讲述了顺序访问和随机访问文件处理程序。第 11 章论及单行和全屏幕的交互式程序。交互式程序既可访问数组也可访问文件，而且包括交互式文件处理以及交互式记录程序。

第 4、5 和 6 章详细叙述集合理论以及真值函数逻辑。第 4 章和第 5 章服务于 3 个很好的整合的意图。这 3 个意图，包括 Venn 框图、布尔代数以及将自然语言语句简化为集合论的语句。这些内容提供了使用来自集合论的技术来简化程序问题的方法。最后，这几章还使用来自

集合论的描述和语句为包括多个判式（嵌套的 if-then 结构）的问题绘制流程图提供素材。（不过，第 5 章可以略去而不会对本书的理解造成障碍。）第 6 章介绍真值函数逻辑。本章以适度的细节解释了真值表、判断表和等价规则。或许本章最重要的内容是讲述将英语语句翻译成真值函数语句的一节。讲述集合论和真值函数逻辑的这几章，都提供了使用这些逻辑来简化编程问题的建议。

本书的某些章节和某些个别练习标有星号。这些章节和练习在学习时可以略去，而不会损及连续性。

依据本书完成学业的学生应该获得较好的算法逻辑的背景知识，在开始其计算机语言课程的学习时应该处于较为有利的地位，因而很可能在学习这些课程时获得较好的成绩。这些学生一般来说应该理解编程算法而且能够为相当复杂的程序指定算法。因而，当这些学生开始学习计算机语言课程时，他们应该能够将精力集中于特定语言的特点上并能够应用该语言来实现程序的解决方案。

在教学计划中纳入依据本书开设的课程可使得编程语言课程进度可以更快一些。因为这门课程已使学生事先熟悉了原来要在每门不同的语言课程的授课之初要急忙讲述的内容（而且要贯穿整个课程讲述到一定的深度）。

# 致　　谢

笔者衷心地感谢许多人在完成本书的工作中以某种方式提供的帮助。学生、同事以及审稿人所提的能够补充或加强本书的意见和建议都已经结合到本书的第三版中来了。此处没有更多的篇幅提及所有的有贡献的人。即使如此，还是有几个人值得特别提及的。

Frank Severance 首先向我介绍了 Warnier 框图并提供许多讲课时使用的想法。Frank 及其在蓝新社区学院（Lansing Community College）开设的课程给了笔者以编写这本教科书的灵感。虽然笔者的工作现在已经换了一个方向，但如果没有他的支持这本书早就流产了。

渥士玻恩大学（Washburn University）的教授 A. Allan Riveland 和 Gary E. Schmidt 以及堪萨斯州的 Blue Cross and Blue Shield 公司的高级信息系统的财政及符合审计员 Larry E. Rinker 阅读了这本书的大多数内容的最早的草稿，提出了使本书继续写作所需的建议和鼓励。这些人通过回答有关计算机编程方面的许许多多的问题，从而在写作本书的全过程中给予了极大的帮助。

渥士玻恩大学的 Gary Schmidt 教授对于笔者对编程问题产生持续的兴趣起了很大的作用。他教笔者学习 RPG III 并向笔者提供在实际工作环境中参加交互式应用程序的设计与编码的机会。

渥士玻恩大学的教授 Ricky Barker、Robert Boncella、Jack Decker、Billy Milner、Don Mitchell、Cecil Schmidt 和 Nancy Tate 以及渥士玻恩地区的信息技术服务部的主任 David Bainum 与副主任 Robert Stoller 都回答过有关计算机语言、文档及系统操作等方面的许多问题。

笔者感谢出版社的审稿人，他们阅读了本书草稿和各版本并提出了意见，这些人包括 Lincoln Andrews（Miami Dade Community College）、Albert R. Banocy（Sinclair Community College）、Janis Bitely（Henry Ford Community College）、Arnold Bradford（Northern Virginia Community College）、Jonah Eng（Florida Junior College at Jacksonville）、Robert Fortune（Ferris State University）、Mary Garrett（Lansing Community College）、Raj Gill 和 Dick Seabrook（Anne Arundel Community College）、Gary Haggard（University of Maine）、Lister Horn（Pensacola Junior College）、Wes Jones（Georgia State University）、Susumu Kasai（St. Louis Community College at Meramec）、Edward L. Kosiewicz（Cuyahoga Community College）、F. P. Mathur（California State Polytechnic）、Dennis McNeal（Delta College）、Carl Penziul（Corning Community College）、Edward Polhamus（Danville Community College）、Erwin Vernon（Sinclair Community College）和 Richard K. Wiersba（Bentley College）。

出版社的本书第三版的审稿人提出了许多对本书有很大帮助的建议和建设性的批评。这些审稿人包括 Theresa McDonald（Texarkana College）、Paul Smith（South Puget Sound Community College）、Rusty Thurman（Morehead State University）和 Charles Wolfe（Mount Sierra College）。

由于本教材曾经在我讲课的班级中使用过，因而本书还处在不断的变化与修订中。笔者特别要感谢 Shelley Bauman、Aaron Cook、Sharon Deters、Brigid L. Foster、Michelle D. Hubach、

Janet Lassiter、LuAnn J. Pickens、Lajean Rinker、Sandra Rood、Kathleen Thompson、Jayne M. Weliking 和 Tracy L. Stotts，是他们校对了本书并对较早的草稿和版本提出了修改建议。还要特别感谢笔者在渥士玻恩大学开设的课程的第一位辅导教师 John T. Tice II，他的批评和建议对本书早期草稿中选择讲述的专题上有很大的帮助。

感谢 John Rood 和 Ezra Rush 的细心工作，他们在本书写作的各个阶段校对了这一版的副本。

笔者还要感谢渥士玻恩学院安息基金会的 Mary B. Sweet 的慷慨支持，他的支持使促成本书的研究工作成为可能。笔者还要感谢 Art Craft Display Co., Inc 和 Greg C. Huseth, CPA, P.A 这两个机构，是他们提供了编写本书的初始和修订的草稿和手稿的办公空间和设备。

最后，笔者要感谢 John Rood 为本书的这一版本的新材料打字。

Harold J. Rood

# 目　　录

# 第 1 章　计算机与流程图

## 1.1　计算机与算法逻辑

算法逻辑研究正确推理的方法或工具，也研究这些工具在解决问题时的应用。本书主要讲述逻辑工具在编程问题上的应用。笔者将考虑算法逻辑的两个传统分支：集合论及真值函数逻辑，还要考虑 4 个现代算法逻辑工具：流程图、Warnier 框图、Nassi-Shneiderman 框图、伪代码。这些工具将被用于分析各种类型的编程问题以及为这些编程问题设计解决方案。

计算机操作的本质使得使用正确的推理成为编程的必需。尽管人们经常听到所谓的计算机错误一说，计算机是可预见的而且是可靠的。计算机准确地执行给予它们的指令。但是，与人类不同，计算机不能思考，不能推理，只能盲目地按照命令行事。因而，如果计算机要完成一项任务，则完成任务所要执行的推理都必须由程序员来做，并以计算机程序的形式输入到计算机中。当程序员正确地推理并在程序中准确地体现该正确的推理时，他或她就可确保获得正确的结果。另一方面，未能正确推理的程序员只能得到错误的结果，因为计算机不会改正即使是最明显的错误。错误的程序只能产生错误的结果。因而，对于算法逻辑的很好的理解在帮助程序员避免编程错误并编写出按照所希望的方式运行的程序方面是很有价值的。

大多数实际的编程问题是相当复杂的。逻辑工具在分析这类问题或是将其分解为简单的步骤以及简单的指令方面是有用的。这种分析是重要的，因为计算机只能理解很简单的指令。计算机执行复杂任务的能力完全依赖于程序员将那些任务分解为一系列的相对简单的指令的能力。使用编程语言可使程序员避免与开关电路的通还是断打交道，从而允许程序员编写较为复杂的指令。但是，即使使用了编程语言，指令也必须是相对简单的。计算机编程的最大挑战是将复杂任务分解为可用所选编程语言编码的一系列指令。正确地应用逻辑工具可大大减轻这种分解工作。

## 1.2　算法

算法是一套完成某一任务或解决某一问题的规则或指令。算法是一系列承上启下的指令，其中每个后继的步骤是由上一步骤的结果来决定的。计算机程序只不过是使用某种编程语言编写的用于计算机的算法。虽然本书并不关心具体的编程语言，但却要深入地讲述作为计算机程序基础的算法。

笔者将要研究的第一个算法是供人类使用而不是供计算机使用的。该算法用于审阅销售收据簿，使用计算器将销售额加以汇总并在 3 列中列出顾客。这里将其称为“销售报告”算法。消费金额大于$1000 的顾客列在 A 列，消费大于$100 但不大于$1000 的顾客在 B 列，而所有其他的顾客在 C 列。由于收据簿中的收据是按顺序开出的，当到达第一张空白的收据时，则任务完成。下面的算法描述了完成这一任务的一种方法：

1. 计算器清零。

2. 读入一张收据。
3. 如果收据是空白的，则写下计算器上的汇总数并停止，否则继续。
4. 将销售额添加到计算器上的汇总项上。
5. 如果销售额大于$1000，则将购买者的姓名输入到 A 列并返回到步骤 2；否则继续。
6. 如果销售额大于$100，则将购买者的姓名输入到 B 列并返回到步骤 2；否则继续。
7. 将购买者的姓名输入到 C 列并返回到步骤 2。

这一套指令就是一种算法。它告诉人们要做什么以及什么时候做，包括何时停止。这一算法不需要人们做什么判定，但是要根据前一步发生的事情来指明下一步要采取的行动。

对于计算机编程来说，算法常常是使用被人们称为流程图的特种框图来设计的。将销售报告变为流程图的形式，就得到了图 1.1。（流程图中使用的不同形状将在 1.3 节中加以解释。）

“销售报告”算法是相当简单的。此算法代表了相当简单的任务。人们想让计算机完成的任务将会比这要困难得多。虽然如此，每一步骤的指令仍然必须是相对简单的。

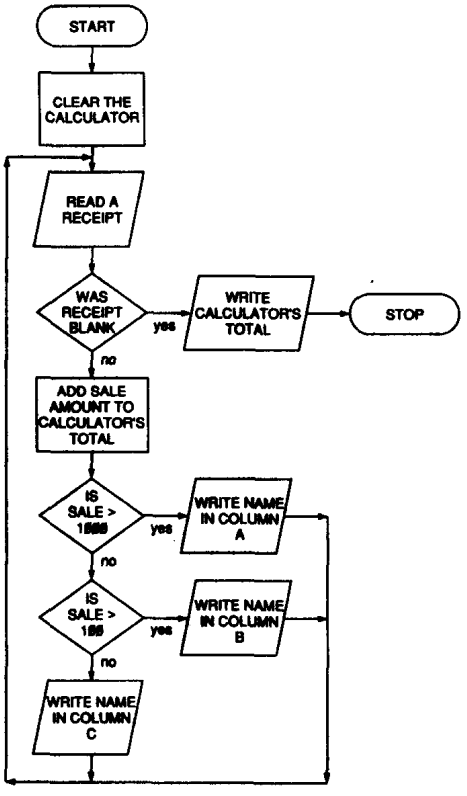



图 1.1 为“销售报告”算法绘制的流程图

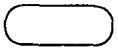
## 1.3 流程图

历史上，计算机程序员用的主要逻辑工具就是流程图。流程图是算法的图形表示。在计算机编程中，流程图是人类对要完成的任务的理解与指示计算机来完成任务的不同步骤两者间的中间步骤。此外，在许多情况下，流程图或另一种类似的工具在开发并理解任务上是重要的。在任何相当复杂的程序中，变化与逻辑关系迅速地超出了即使是最优秀程序员所能同

时思考的能力。流程图是考虑程序的一部分的一种手段，并可在继续到下一步之前用于判定并记录对该部分程序的解决方法。流程图是图形表示，因而，它们所代表的程序逻辑比编写出的程序逻辑更容易被人理解。流程图提供了附加的优越性，使得程序员可以设计问题的逻辑关系而不必考虑特定语言的句法。

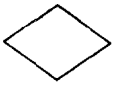
现代计算机环境强调程序及流程图要遵循使其成为“结构化的”设计需求。结构化的流程图需求以及使用结构化流程的原因将是第 2 章的主题。结构化的流程图对于那些程序设计新手来说，常常看起来是不必要的复杂，因而要等到读者理解了流程图的基本组成以及如何将基本组件结合起来描述可运行的程序之后，笔者才来讲解这部分内容。在第 2 章中，笔者才来考虑将程序结构化所需的重新设计。

用于流程图的某些符号已经成为标准[原文注]：

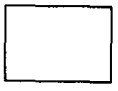


终止符号指明处理的开始与结束，常常包括单词 START、STOP 或 END。

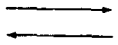
输入/输出符号（I/O 符号）指明计算机要获得新的数据或是记录下计算的结果。这个符号常常包括读取记录或是打印报告行的指令。



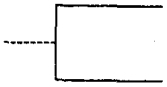
判式符号代表程序流程中的分支；它必须包括指明程序流向哪个分支的指令。这一符号通常包括一个问题，而每个分支是用答案来标记的。



处理符号包括由其他符号不能代表的指令。处理符号通常包括用于计算或是用于存储数据的指令。

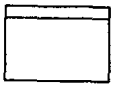


流程方向线指明程序的流向。箭头画在流程方向线交会处或是流程方向线与其他符号的交会处。

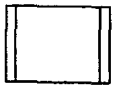


注释或注解符号用于说明其他条目。虚线表明注释的位置。

连接符用于连接流程图的区域。连接符常以组的形式使用，用来表明程序的流程（或控制）从一点向另一点的转移。一组中的所有连接符包括相同的字母字符（字母或数字）。仅当连接点出现在同一页时才使用连接符。



当定义处理过程的流程图包括在当前一套流程图中时，条状符号用于代表预先定义的处理过程（或一系列的操作）。



当定义处理过程的流程图不包括在当前一套流程图中时，预定义的处理过程符号用于代表预处理过程。

当程序连接点出现在不同页上时，换页连接符号用来连接流程图的不同区域。

许多标准符号和流程方向线揭示了有关程序逻辑的相当多的信息，即使符号本身并不包括任何信息。请看一下图 1.2。从符号的形状和流程方向线来看，人们就可以了解使用这一流

[原文注] 流程图已由美国标准局（American Standards Institute）加以标准化。American Standard Flowchart Symbols and Their Usage in Information Processing（美国标准流程图符号及其在信息处理中的用法），ANSI X3.5-1970，New York，1971.

程图的程序将

1. 开始。
2. 读入某些信息[原文注]。
3. 作出判定。
4. 根据上面的判定执行一个处理过程或是另一处理过程。
5. 打印任何情况下的结果。
6. 停止。

使用流程图的一个好处是，可使人们考虑计算机程序的逻辑，而不必同时考虑程序的内容。当程序变得复杂时，决定并理解逻辑结构而不被细节所迷惑的能力是必须的。

图 1.3 有点复杂。先考察一下此图，在接着往下阅读之前，请读者看一下自己能否根据所遇到的符号决定将要发生什么。读者应该看出，这一程序将会开始运行、读入信息 A，然后提出问题 B。如果 B 问题的答案是否定的，则将完成过程 C，然后再提出问题 D。如果 D 问题的答案是肯定的，则将完成过程 I；无论何种情况，都将完成过程 E，然后提出问题 F。如果 F 的回答是肯定的，则将完成过程 G 并打印 H。程序流程接着通过连接符 1 又回到程序的顶部。这些步骤将继续，直到问题 B 的答案为肯定的。当 B 问题的答案为肯定时，将完成过程 J，打印 K，然后程序停止。

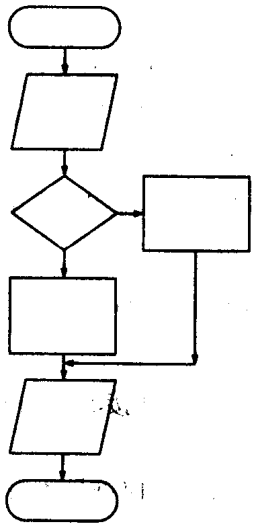


图 1.2 只有标准符号与流程方向线也能描述程序逻辑的许多方面

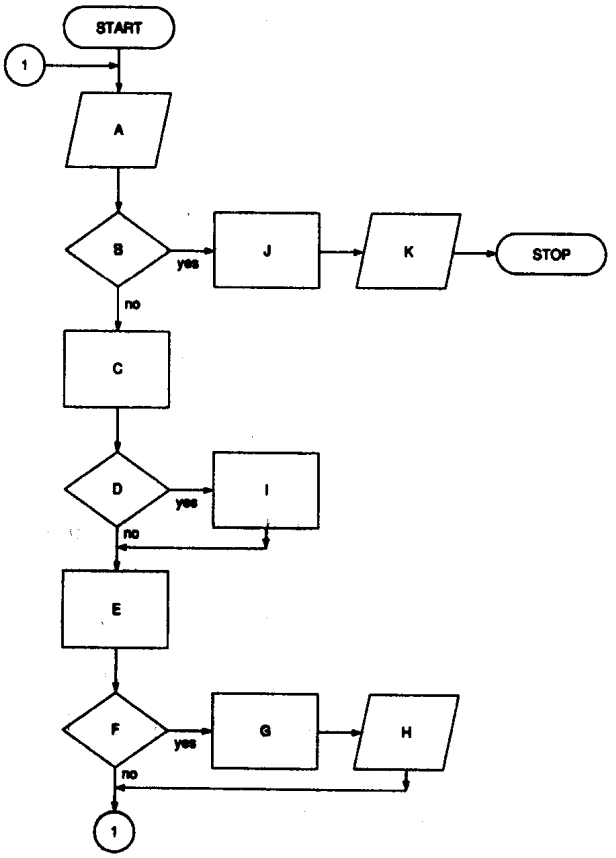



图 1.3 随着流程图所描述的程序的运行，决定将要发生的事情

[原文注] 由于此符号放置在流程图的开始处，我们假设 I/O 符号代表输入要处理的数据。