计算机程序逻辑与结构化设计

Haroed J. Rood 著

怡始 译 罗经宇 校

清华大学出版社

计算机程序逻辑和结构化设计

Harold J. Rood 著

怡 始 译

罗经宇 校

清华大学出版社

内 容 简 介

这是一本介绍计算机程序的逻辑和结构化设计的入门书，全面系统地讨论了现代程序设计的常用方法。

全书共分九章，包括计算机和流程图、集合的逻辑、结构化流程图、真值函数逻辑和判定表、Warinér/Orr 图、伪代码、Nassi-Shnei derman 图、数组和数据处理、编辑和文件处理程序以及由 Warnier 图编制 COBOL 程序的方法等内容。每章均有例题和习题，书末给出了部分练习的答案。

本书作为计算机程序设计的教材，在编写上尽量做到满足不同读者的需要。讨论中不涉及具体的计算机语言，因此初学者也能够看懂，并从中学到逻辑分析的基本知识。熟悉计算机语言的读者则可以将书中介绍的方法应用于具体的程序设计和系统分析之中。

计算机程序逻辑和结构化设计

Harold J.Rood 著

怡 始 译

罗经宇 校

☆

清华大学出版社出版

北京 清华园

清华大学印刷厂印刷

新华书店北京发行所发行

开本：787×1092 1/16 印张：23.5 字数：557 千字

1989 年 5 月第 1 版 1989 年 5 月第 1 次印刷

印数：0001—4000

ISBN 7-302-00158-8/TP·60

定价：4.55 元

译 者 的 话

本书概括了程序设计中的共同逻辑问题，全面介绍了计算机数据处理与程序设计中逻辑分析及设计的常用方法，是计算机程序设计课的教材。本书重点放在方法的讨论上，不涉及具体的计算机语言，通过大量的例子来阐明要点，因此适用于不同层次的读者阅读。不懂计算机语言的读者学完本书后，打下了逻辑分析的基础，这对于进一步学习计算机语言十分有利。已经掌握了计算机语言的读者可以将本书介绍的方法与具体的语言结合起来编写实用的程序。

本书由刘明华（承担前言和 1、7 章）、潘真微（承担 2、3 章和附录）、王如琦（承担 4、6 章）、郭学书（承担 5、9 章）和金文织（承担 8 章）等同志翻译，由罗经宇教授审校。

在翻译过程中，虽然对译文进行了反复的推敲和多次校核，力求翻译正确、流畅，但仍难免有处理不当或失误之处，敬请读者批评指正。

译 者

1988 年 1 月 20 日于清华园

前　言

这是一本介绍数据处理逻辑的入门书，是专为准备学习程序设计但尚未起步的读者，特别是为那些数学和逻辑知识比较欠缺的读者编写的。书中除了简单的字母、数字和文字外，完全不涉及具体的语言，因此适用范围比较广泛。由于不讨论语言的语法问题，只谈逻辑问题，所以读者可以集中精力于问题的逻辑方面而勿需顾及标点、列数等内容。本书概括了对于许多程序设计语言都是共同的逻辑问题，为读者提供了众多程序设计教材必不可少的逻辑知识。

本书的编排方式既适用于学生借助文字说明学习，也适用于学生通过一个一个的实例来学习。书中列举程序设计的例子是为了说明设计的要点，而不是作为范例，故有意简化了程序设计的许多方面，以便读者深入掌握重点阐述的内容。由于计算机语言之间千差万别，因此在其它方面也进行了一些简化。换句话说，选择例子的出发点是为了说明问题，其次才考虑设计是否完美和实用等方面。

书中介绍的设计工具有流程图、结构化流程图、Warnier/Orr 图、Nassi-Shneiderman 图和伪代码。这些工具既可以用于程序设计，也可以用于系统分析和设计。本书除讨论系统流程图的那一小节外，只介绍这些工具在程序设计中的用法，而不论述它们在系统设计中的应用。

本书所介绍的工具和方法以便于初学者掌握为限。他们还应在以后学习具体的程序语言和系统的过程中，在实际工作中，不断提高自己设计程序的能力。作者不打算在这本书中对程序设计加以全面论述，也不打算把这本书写成推荐给大家的有关程序设计工具的专业标准或正式说明。

本书首先引入流程图作为讨论程序设计逻辑的工具。因为非结构化流程图对于初学者说来是最容易的，所以先加以介绍。第 4 章讨论结构化流程图。第 6 章介绍 Warnier/Orr 图。第 7 章介绍伪代码和 Nassi-Shneiderman 图。以上每一章都要求读者应用这些结构程序设计方法重复做前面的练习。到第 7 章末，读者就会比较深刻地了解以下四种现行的程序设计或文件编制工具：结构化流程图、伪代码、Nassi-Shneiderman 图和 Warnier/Orr 图。

第 8 章讨论一维和二维数组，提出了一些在流程图和 Warnier 图设计中对于初学者有一定难度的问题。

第 9 章介绍文件处理，讨论了文件结构，包括文件的编辑、提取以及更新程序。因为顺序处理涉及到文件处理中最有趣的逻辑问题，而且，掌握了顺序处理，也就为理解一般的文件处理打下了良好的基础，所以本章详细讨论了顺序处理。为了便于比较，还简单地介绍了加下标顺序法（ISAM）和直接文件处理法。

第 2、3 和 5 章介绍集合论和真值函数逻辑。在第 2 和第 3 章要达到下述三个密切相关的目的：第一，介绍集合论，包括 Venn 图和布尔（Boolean）代数以及如

何把自然语句转换成集合论语句；第二，介绍一些用集合论的技巧简化程序设计的方法；第三，利用集合论中的描述和语句解决包含有多次判定（嵌套 if-then）的流程图设计问题（不过，略去第 3 章并不会破坏全书的体系）。在第 5 章，详细介绍了真值函数逻辑，包括真值表、判定表以及等值规则。第 5 章中最重要的内容，看来是关于把英语陈述语句转换成真值函数语句的那一节。介绍集合论和真值函数逻辑的这几章都提到了应用这些逻辑去简化程序设计问题的一些方法。虽然最简单的解法并不总是最适当的方法，而且许多设计人员也不熟悉它们，但是这些逻辑对于掌握了它们，知道其用法，而且又善于加以运用的程序设计人员却是非常有用的。

修完本教程的学生将掌握良好的逻辑知识，这对于他们学好计算机语言课十分有利。这样的学生理解一般程序设计的逻辑，能够判别复杂程序的逻辑，因此当他们上计算机语言课的时候，就可以集中精力于具体语言的特点和运用这些语言去解决程序问题。

如果先修基于本教材的课程，程序设计语言课就可以进行得快一些。这门课使学生熟悉那些在各种语言课开始时（和一段时间内）一带而过的内容。

关于课程安排的建议

本书的安排使之适于作为各种不同逻辑课程的教材。无论是强调传统逻辑，还是强调程序设计逻辑，或者是突出应用的课程都可以采用。书中第 1、2 和 4 章是后续章节的基础知识。下面推荐可以满足不同侧重要求的半学年和短学期的课程表。

半学年的课程表

周数	周学时为 3 的教学内容				周学时为 5 的教学内容
	强调传统逻辑	强调程序设计逻辑	强调应用	一般情况	
1	1.1—1.4	1.1—1.4	1.1—1.4	1.1—1.4	1.1—1.5
2	1.5	1.5	1.5	1.5	1.6—1.10
3	1.6—1.9	1.6—1.10	1.6—1.10	1.6—1.9	2.1—2.6
4	2.1—2.6	2.1—2.6	2.1—2.6	2.1—2.6	2.7—2.8,3.1
5	2.7—2.8	2.7—2.8	2.7—2.8	2.7—2.8	3.2—3.6
6	3.1—3.2	4.1—4.3	4.1—4.3	4.1—4.3	4.1—4.5
7	3.3—3.6	4.4	4.4	4.5	5.1—5.5
8	4.1—4.3	4.5	4.5	5.1—5.3	5.6—5.9
9	4.5	6.1—6.2	5.11—5.12	5.4—5.6	5.11—5.12
10	5.1—5.3	6.3—6.4	6.1—6.4 或 7.1 或 7.2	5.7—5.9	6.1—6.3
11	5.4—5.6	6.5—6.6	8.1—8.5	5.11—5.12	6.4—6.6
12	5.7—5.8	7.1	8.6—8.7	6.1—6.2	7.1—7.2
13	5.9—5.10	7.2	8.8—8.10	6.3—6.4	8.1—8.5
14	5.11—5.12	8.1—8.5	9.1—9.3	8.1—8.5	8.6—8.10
15	6.1—6.2	8.6—8.7	9.4	8.6—8.7	9.1—9.4
16	6.3—6.4	8.8—8.10	附录 A	8.8—8.10	附录 A

小学期的课程表

周数	周学时为 3 的教学内容			周学时为 5 的教学内容		
	强调传统逻辑	强调程序设计逻辑	强调应用	强调传统逻辑	强调程序设计逻辑	强调应用
1	1.1—1.4	1.1—1.4	1.1—1.4	1.1—1.5	1.1—1.5	1.1—1.5
2	1.5	1.5	1.5	1.6—1.9	1.6—1.9	1.6—1.10
3	1.6—1.9	1.6—1.9	1.6—1.10	2.1—2.6	2.1—2.6	2.1—2.6
4	2.1—2.6	2.1—2.6	2.1—2.6	2.7—2.8	2.7—2.8	2.7—2.8
				3.3—3.4	4.1—4.3	4.1—4.3
5	2.7—2.8	2.7—2.8	2.7—2.8	3.5—3.6	4.4—4.5	4.4—4.5
				4.1—4.3		
6	3.3—3.6	4.1—4.3	4.1—4.3	4.5	6.1—6.3	5.11—5.12
						6.1
7	4.1—4.3	4.5	4.5	5.1—5.5	6.4—6.6	6.2—6.4
8	5.1—5.3	6.1—6.2	5.11—5.12	5.6—5.9	7.1—7.2	8.1—8.5
9	5.4—5.7	6.3—6.4	8.1—8.5	5.11—5.12	8.1—8.5	8.6—8.10
				6.1—6.4		
10	5.11—5.12	7.1—7.2	8.6—8.7	7.1—7.2	8.6—8.10	9.1—9.4

表中推荐的进度可以使初学者掌握所安排的内容。即，他们能够顺利地通过与指定的习题水平相当的闭卷考试。对已经学完几门程序设计课的学生，进度可以更快些。其他的学生会感到吃力，因此进度可以放慢一些。用加“*”与否来区别内容和习题的难易，加“*”的对于初学者是最困难的部分。

目　录

1. 计算机和流程图

1.1 计算机和逻辑

逻辑是探讨进行正确推理的各种原则或者工具，并研究如何将这些工具用于解决问题的一门学科。本书介绍逻辑工具在程序设计中的应用，讨论逻辑的两个传统分支——集合论和真值函数逻辑以及四种现代逻辑工具，即流程图、Warnier 图、Nassi-Shneiderman 图以及伪代码。并用这些工具来分析和解决各种类型的程序设计问题。

计算机运算的特点使程序设计一定要有正确的推理。尽管我们有时听说计算机也会出差错，实际上，计算机既能预测，又十分可靠。计算机会准确地执行接受到的指令，但是不能象人那样进行思维。计算机没有推理能力，只会盲目地执行命令。因此，如果要计算机完成一项任务，程序员必须进行为完成该任务所需要的全部推理，并把这种推理过程以计算程序的形式输入到计算机中。如果程序员的推理是正确的，并且在他的程序中准确地体现了这种推理，他就一定能获得正确的结果。反之，若推理失误，得到的结果也必定是错误的。这是因为，计算机连程序设计上最显而易见的错误也不能纠正。错误的程序必然导致错误的结果。因此，通晓逻辑有助于程序员避免程序设计上的错误，并设计出能如愿运行的程序。

大部分实际的程序设计问题都是非常复杂的。逻辑工具对于分析这些问题或将它们分解成简单的步骤和简单的指令是非常有用的。这种分析十分重要，因为计算机只能理解极其简单的指令。计算机处理复杂问题的能力完全取决于程序员把这些问题分解成一系列相当简单的指令的本领。利用 FORTRAN 或 COBOL 之类的程序设计语言，程序员就不必操心开关电路的开关操作，只要给出相加、读数、打印以及检索信息之类的指令就行了。但是，即使采用了程序设计语言，有关指令也还是要比较简单才行。计算机程序设计的关键就是要把复杂的问题剖析成计算机能够接受的许多简单的指令。正确地应用逻辑工具使这样的分析工作容易得多了。

1.2 算　法

算法是完成一项任务或解决一个问题的一套规则或者说一套指令。算法指令一步接一步，每一步都要依据前面那些指令执行的结果来确定。一个计算机程序就是用程序设计语言为计算机编写的一种算法。本书不打算讨论程序设计语言，而是广泛地讨论作为计算机程序基础的算法。

我们首先来看一个为个人设计的而不是为计算机设计的算法。这是一个统计销售发票存根的算法，权且叫做“销售报告”算法，其内容是用计算器汇总销售额，并给出按

三类顾客划分的清单，购物花了 1000 美元以上的顾客列入 A 类，多于 100 美元不超过1000美元的列入 B 类，其余的归入 C 类。因为发票存根是按顺序排列的，所以遇到空白存根统计就可终止。以下是完成这项任务的一个算法：

1. 计算器清零；
2. 读发票存根；
3. 若存根为空白，用笔记下计算器上的总数，停止计算；否则，继续进行；
4. 将销售额加到计算器的总数上；
5. 若销售额大于 1000 美元，将顾客名字填入 A 类，回到第 2 步；否则，继续进行；
6. 若销售额大于 100 美元，将顾客名字填入 B 类，回到第 2 步；否则，继续进行；
7. 将顾客名字填入 C 类，回到第 2 步。

以上这一套做法说明就是一个算法。它告诉进行统计的人该做什么，什么时候做，什么时候停止。这里不要求进行统计的人去判断要做什么，而是根据前面各步执行的结果指明他该怎样做。

在计算机程序设计中经常利用被称为流程图的一些图解来设计算法。如果把上面说到的“销售报告”算法画成流程图，就得到图 1.1（流程图中用到的各种图形将在 1.3 节中加以解释）。

这个“销售报告”算法十分简单，反映的只是一项极其简单的作业。这项作业若让计算机来完成就困难得多了；不过，每一步说明仍然要象这个算法中的一样简单。

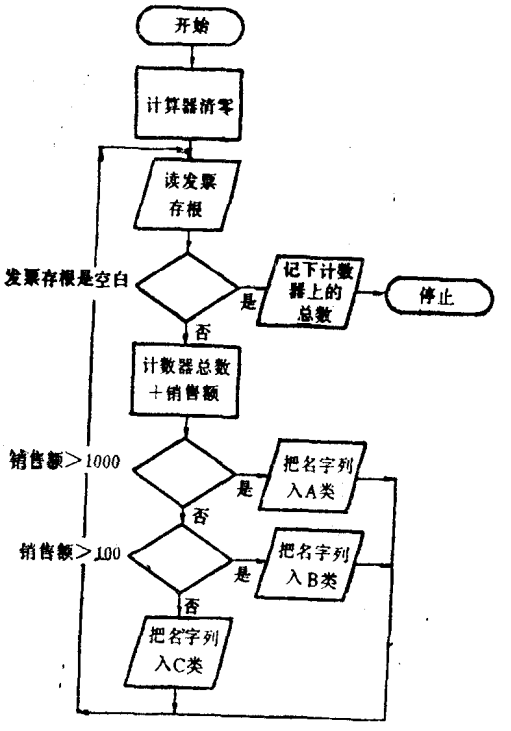


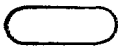
图 1.1 “销售报告”算法的流程图

1.3 流 程 图

流程图一直就是计算机程序设计员的主要逻辑工具。流程图是算法的图解。在计算机程序设计中，流程图是程序员对所要完成的任务的理解与用来指示计算机完成该项任务的技术语言之间的一个中间步骤。而且，在许多情况下，流程图或其它类似的工具还是阐明和理解所要完成任务的必不可少的手段。对于任何一种比较复杂的程序，由于众多的变量，经常可能碰到的意外情况以及错综复杂的逻辑关系，就连最优秀的程序员也不容易一下子就搞清楚。流程图使我们把问题分成许多部分来考虑，在着手下一部分之前，

先要确定前一部分的解答，并把这个答案记录下来。由于流程图是图形，故用它表示的程序的逻辑当然要比用文字表达的程序逻辑更容易理解。此外，流程图还有一个优点是程序员不受具体语言的语法的约束，可以集中研究所给问题的逻辑。

流程图中使用的一些符号已经标准化（参见左图）：

1. 

（1）起止符　表示过程的开始和终止。该符号中常框有START、STOP 或 END 等词。

2. 

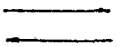
（2）输入/输出符（或 I/O 符）　表示新的数据输入计算机或者把计算结果记录下来。该符号中常框有要求读记录或者打印输出的说明。

3. 

（3）判定符　表示程序进行中的分支。判定符中必须框有指明下一步走向的说明。框内通常是一个提问。对该问题的回答标注在判定符的每个分支上。

4. 

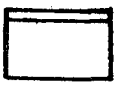
（4）过程符　框中是不为其它符号所表示的说明，常常是计算说明。

5. 

（5）流向线　表示程序进行的方向。流向线的箭头画在两条流向线相交的地方以及流向线与其它符号的会合处。

6. 

（6）连结符　是用来把流程图上的区域连结起来的符号。连结符总是成组使用，表示程序(或控制)从一点转移到另一点。同一组的几个连结符全都标以相同的字符或数符（字母或数字）。这种连结符只在相互连结的各点出现在同一页上时才用到。

7. 

（7）条纹符　表示预先定义的过程（或一组操作），而且是指定义该过程的那些流程图包括在现有流程图中的情况。

8. 

（8）预先定义的过程符　是表示预先定义的过程，但是定义该过程的流程图不包括在现有的流程图中。

这些标准符号和流向线反映了程序逻辑的不少信息，即使这些符号本身没有标注任何信息也是如此。例如，我们从图 1.2 中符号的形状和流向线就可以知道采用这个流程图的任何程序都包含下述内容：

1. 开始；
2. 读某些信息[1]；
3. 作判断；
4. 根据判断结果进行这种或那种过程；
5. 无论哪种情况都要打印结果；
6. 结束。

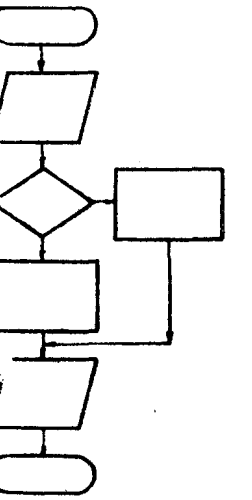


图 1.2　标准符号和流向线本身所代表的众多的程序逻辑

采用流程图的一个优点是使我们可以不必考虑程序的内容就能分析计算程序的逻辑。当程序很复杂时，这种不纠缠于细节，而集中精力去分析和处理逻辑结构的能力是非常重要的。

1　因为 I/O 符位于流程图的开头，我们可以认为它代表输入待处理的数据。

图 1.3 是一个比较复杂的流程图。在您阅读下面的说明之前，根据图中遇到的符号，试分析这个流程所代表的程序的步骤。

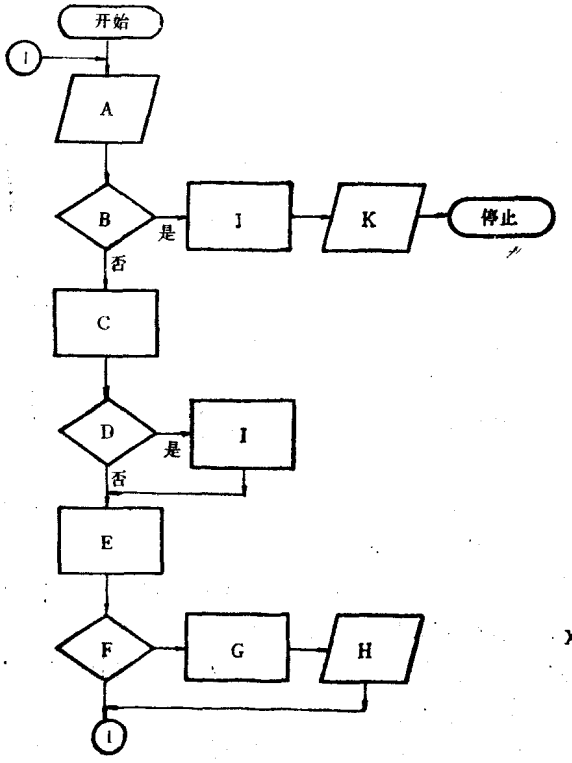


图 1.3 分析该流程图所描述的程序的步骤

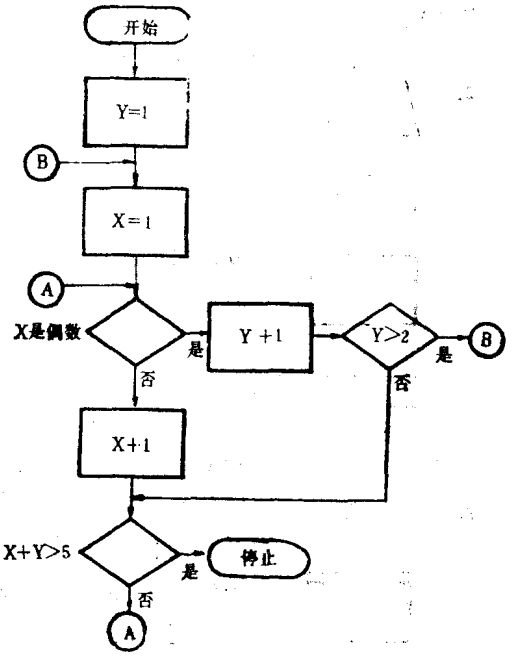


图 1.4 分析该程序执行中 X、Y 的变化

图 1.3 所表示的程序步骤如下：

程序一开始，读信息 A[2]，然后提问 B。若回答是否定的，完成过程 C，继而提问 D；若对 D 的回答是肯定的，完成过程 I。无论哪种情况下，均接着完成过程 E，并提问 F。若对 F 的回答是肯定的，则完成过程 G，打印 H。然后经过连结符 1 控制流程返回顶端。上述步骤重复进行，直至对问题 B 的回答为肯定时方告终。此时完成过程 J，打印 K，而后程序结束。

但是，图 1.4 的符号却包含有指令。研究该流程图，试分析随着程序的执行 X 和 Y 的变化。

该程序由置 X、Y 均为 1 开始。接着，由于 X = 1，不是偶数，所以将 X 加 1 得 2，因此 X + Y = 3，第二个判定的回答是否定的，控制通过连结符转移到顶端的 A。现在 X 是偶数了，所以 Y 应加 1，Y = 2。此时 Y 不大于 2，X + Y = 4，控制 2 转移到顶端的 A。X 仍为偶数，Y 应加 1，Y = 3。现在 Y 大于 2，控制经过连结符 B 移到顶端的 B。X 再置 1，X 不是偶数，X 应加 1，控制转移到顶端的 A。如此，继续跟踪程序的进行，可以证实，程序终止时 X = 2，Y = 4。

2 因为 I/O 位于流程图开头，我们设它代表输入数据。

这样跟踪哪怕是简单的流程图也是很麻烦的，尤其是想用心算的方法记下数值的变化过程就更困难。采用数值表可以使流程图的跟踪容易得多。数值表由许多列组成，每一列对应程序中一个不同的字母。在分析该流程图时，随时记下每个字母所获得的值。出现新的数值就划掉旧的值[注]，记下新值。若一个数值在两个或更多的运算中保持不变，则在表中重复出现。表 1.1 是图 1.4 所示流程图的数值表。

数值表帮助我们跟踪程序。表底的数值是程序终止时计算机存储的数值。重要的是要记住，只有最后列入表中的数值才是要求的 X、Y 值。如表 1.1 所示，每次 X 为 1 时，Y 都不为 1。数值表不回答诸如“当 Y = 3 时，X 为何值？”的问题，答案只能从观察填表时数值的变化得到。虽然如此，数值表在说明流程图所描述的详细过程时仍然很有用。

表 1.1 图 1.4 所示流程的数值表

X	Y
$1^{\times}$	$1^{\times}$
$2^{\times}$	$2^{\times}$
$1^{\times}$	$3^{\times}$
$2^{\times}$	
$1^{\times}$	

表 1.2 图 1.4 所示流程图扩充了的数值表

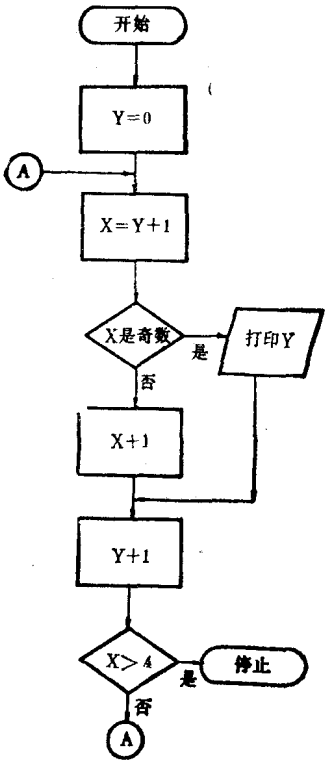
X	Y	第一个判定符	X+1
$1^{\times}$	$1^{\times}$	$1^{\times}$	$1^{\times}$
$2^{\times}$	$2^{\times}$	$2^{\times}$	$2^{\times}$
$1^{\times}$	$3^{\times}$	$3^{\times}$	3
$2^{\times}$	4	$4^{\times}$	
$1^{\times}$		$5^{\times}$	
2		6	

有时，我们还想记录某个判定符或过程符执行的次数，或者什么样的数值已经打印出来了。为了不断记录这些附加的程序特性，我们将数值表适当地加以扩充。表 1.2 也是图 1.4 所描述的流程的数值表。不过，除了表 1.1 记录的数据外，还记录了第一个判定符达到的次数以及 X 加 1 的次数。

练 习 1.1—1.3

1. 分析下页上面的流程图，填写数值表，并回答问题。
   a. 打印何数？（用“PRINT Y”列记录打印的每个数值）
   b. 程序结束时 X、Y 为何值？
   c. 第二个判定符执行的次数。
2. 分析下页下面的流程图，填写数值表，并回答问题。
   a. 程序结束时 X、Y 和 Z 为何值？
   b. 第一个判定符执行的次数。

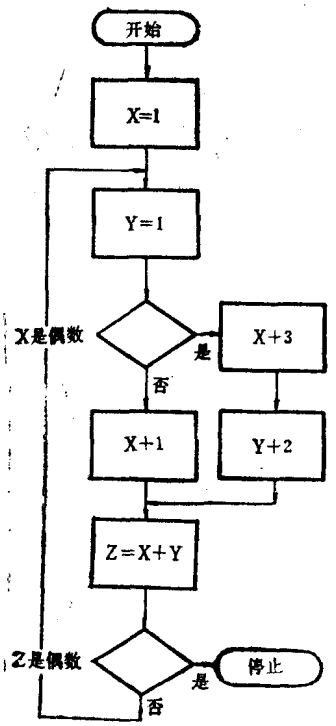
注：在数字右上角加“×”表示划掉的字。——译者注。



数 值 表

Y	X	打印 Y	第二个判定符

练习 1



数 值 表

X	Y	Z	第一个判定符

练习 2