程序设计方法学

冯树椿 徐六通 编著

浙江大学出版社

程序设计方法学

冯炳春 徐六通 编著

浙江大学出版社

1988·杭州

内 容 简 介

本书是一本介绍程序设计方法学的概貌及各种实用方法的教科书。内容涉及结构程序的概念、程序设计的主要控制结构、程序正确性的证明、数据类型抽象、递归方法、逐步求精法、模块设计和推导技术，最后介绍近几年新发展的程序变换技术。

本书可供计算机专业的师生选用，也可供计算机技术人员参考。同时，本书是中央电大“程序设计方法”课程的教学用书。

责任编辑：尤建忠

程序设计方法学

冯树椿　徐六通　编著

*　　*　　*

浙江大学出版社出版
余杭县三墩印刷厂印刷
新华书店浙江省总店经销
*　　*　　*
开本：850×1168　1/32　印张11.875　字数298千
1988年7月第1版　1988年7月第1次印刷
印数：1—7700
ISBN 7-308-00091-5
TP·010 定价：2.95元

前　言

随着计算机的迅速发展，用户与机器速度以及内存的矛盾逐步趋向缓和。一个优秀的合格程序设计人员已不再拘泥于追求省几条指令，压缩有限空间或提高一点可怜的速度，相反是在程序设计的一开始就考虑结构上的合理以及正确性，从而使自己的创造性作品获得好结构的风格。

今天计算机程序设计已从一种技巧发展成为一门科学，而结构程序设计则作为一门科学的新概念得到了公认和推广。人们对程序的要求已逐渐转向易读易写和结构简明易懂（即所谓的好结构）程序，程序的正确性也得到了高度重视。因此，介绍和学习程序设计的方法是很有现实意义的。

本书是在大学多年讲授程序设计方法的基础上，不断完善、整理讲稿而编写成的，故并不是程序设计方法的专题介绍，而是对整个程序设计方法学的概貌及各种实用方法的一般叙述。对读者来说，从认识规律出发训练各种良好习惯，在学习过程中深化知识，提高能力，开阔眼界，再通过自己的咀嚼与消化去体会程序设计领域的纷繁灿烂则是大有裨益。

全书共分七章。第一章叙述程序设计引论与结构程序概念。第二、三章讨论程序设计中主要控制结构的易读易写原则，从而引出程序基本结构的正确性证明及方法。第四章介绍数据类型的概念及抽象数据类型代数规范的描述。第五、六章介绍程序设计中递归、逐步求精、模块设计和推导技术等方法，第七章介绍近几年新发展的程序变换技术。

本书以PASCAL语言为背景，但为叙述方便也引入了一些在内容及形式上均类似PASCAL的语言，这对理解内容，方便教与学都是很有帮助的。

本书内容是针对计算机系大学生而定稿的，故每章附有练习。但在叙述及选例方面也考虑了计算机科学工作者的参考需要。

特别有幸的是在编写本书过程中得到了中央电大的重视。在本书出版的同时将发行由我们节选编辑、杭州电大拍摄的“程序设计方法”电视教学片，这对理解本书与深化阅读均会带来有利的条件。

在本书编写过程中上海交通大学孙永强教授对编写大纲内容等方面提供了建树性的意见，浙江大学李千生副教授仔细阅读了手稿并提出了一些合理的修改意见，浙江大学计算与信息中心提供了很多方便和支持，编写者在此对他们表示衷心的感谢。

由于时间仓促，加之编者水平有限，书中错误在所难免，诚恳希望得到各方面的批评和指教。

编　者

1987年12日

目　录

带有“*”的为专科选读内容

第 一 章

引　论

§1.1　程序设计发展与程序设计方法学

计算机是当代科学技术的一次重大发明，也是打开人类智力解放大门的一把钥匙。计算机的生产和应用是衡量一个国家现代化水平的重要标志。

计算机的发展与应用，引起了世界各界的重视。计算机工业可以说是发展最迅速的工业之一。回顾历史，不难看出程序设计的方法也在不断地变化、发展。

现代数字计算机之所以能进行程序设计，是因为它有一有穷的指令表和执行指令的设备及存贮记忆设备，且能对数字离散地一步步处理，又因计算机内部指令和数据都是用二进制数表示，且他们共享存贮，因而产生如下特征：

1）程序P结束，存贮器中又可接受新的程序Q，并顺序执行。

2）根据某一程序产生的一系列结果（数），可作为第二步要执行的程序。

3）A计算机上表示的程序或数据可转换成B计算机上的程序或数据。所以，它具有灵活性和广泛的可应用性。

五十年代用机器指令代码进行手编程序是相当麻烦的，因此在工程技术人员中很难推广。当时程序设计是繁杂而又仔细的劳动，重复处理0与1又得花费大量的时间来编制与调试程序，故后期逐渐被用符号指令的汇编程序设计所代替。汇编程序在记忆、直观等方面有了提高，但在本质上毕竟没有什么飞跃，因此当时的程序设计观念中评价一个程序的好坏是指令条数要少，运行速度要快，存贮单元要省。这种少、快、省教学模式的影响至今还深深地留在人们记忆中。随着计算机的发展，这种观念将会日益影响计算机的普及应用以及效率的发挥。

第一个高级语言 FORTRAN 的出现，大大简化了程序设计。用高级语言编写的程序，基本上与机器无关。我们可以摆脱机器本身的特性限制，而集中精力于算法本身。在使用机器时的一些常有错误又可通过编译程序时检查出来，因而高级语言的发展显示了强大的生命力。它们随着计算机的应用推广而渗透到各学科和技术领域。一系列不同程序风格和不同服务对象的专用语言和通用语言的大量出观，也推动了程序设计方法的变化。迄今已有上千种高级语言在运行，而世界上通用的也有数十种，诸如 FORTRAN, BASIC, Algol, COBOL, LISP, PL/1, Prolog, PASCAL, C…

在程序设计发展过程中，特别是在七十年代初期，大型系统软件操作系统、数据库等的出现给程序设计带来了新的问题。不但编制程序需要大量资金和人力，而且产品的可靠性、维护、修改和移植等本身又需要人力和财力。使人更为扫心的是常常花费几千人年工作量所得到的软件还蕴含着大量的错误。为此，人们把程序设计这种困境叫做“软件危机”。

上述种种问题促使人们开始对程序设计方法进行研究，从而提出了应该怎样设计程序，设计程序的基本指导思想是什么，采用什么样的程序设计方法为好等问题。

1969年 Dijkstra 首先提出了结构程序设计的思想与概念，他强调从程序结构上来研究与改变传统的设计方法。经大批计算机科学工作者的实践、努力、争论、探索，结构程序设计得到了普遍应用，而程序设计也逐步向规范化、工程化发展。用结构程序设计方法编制的程序不仅结构清晰、易读、易学，且也易于证明程序的正确性。

今天，方法学的研究已获得了不少成果。除了结构程序设计方法日趋完善推广外，模块程序化、递归程序设计和逐步求精方法等均成为当前程序设计中十分有效的方法。抽象数据的代数规范和程序的形式推导技术目前还在发展中，特别是程序变换技术和自动化方面虽然还不很成熟，但已取得了可喜的进展。

§1.2 程序设计的一般途径

程序设计中最为关心的是程序的效率与程序的正确性。程序的效率常由算法的效率来决定。程序的正确性研究要保证程序的易读性、可靠性、可维护性等。

应用结构程序设计、自上而下的逐步求精及单功能模块方法已能解决大部分中小型程序的设计问题，但结构程序错误仍然会发生，这样就促使程序自动验证及调试技术的不断发展。图1.1粗略表示了程序设计的一般途径。

在程序设计中，十全十美的方法是不存在的。所谓好的程序设计方法只能是针对某些特定的情况而言的。如我们认为递归方法是一种较好的方法，它结构清晰，算法精练，又易推导等，但它常常是运行效率不高。所以，我们在学习方法学时应该着重在程序设计方法本质的理解和灵活的应用，这是十分重要的。

我们的方法学课主要立足于小规模程序的训练，对于较大的程序系统将应用抽象、枚举、归纳原则，采用分块、逐步求精等

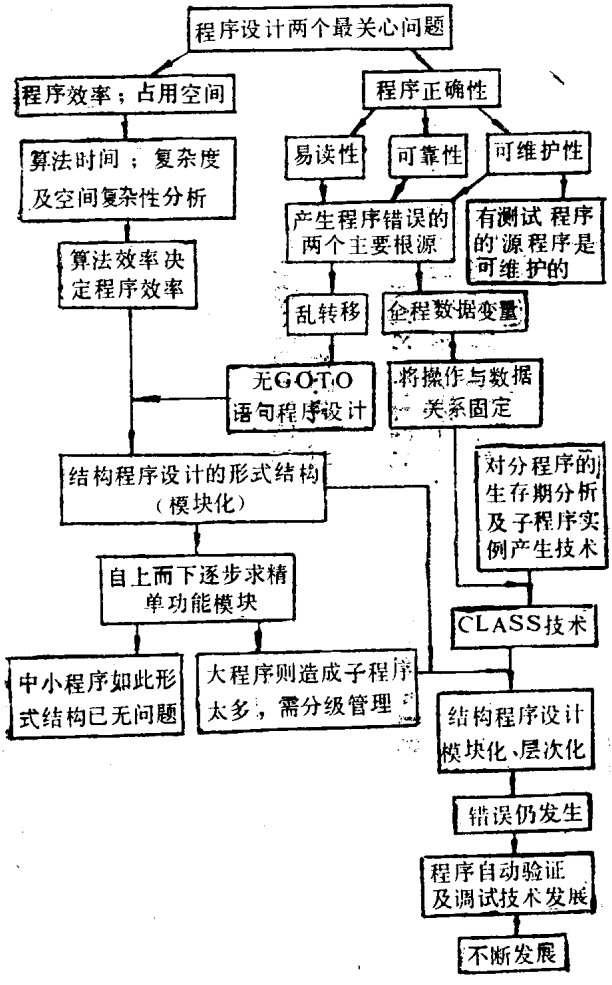


图1.1

方法，将其化大为小，逐步解决。

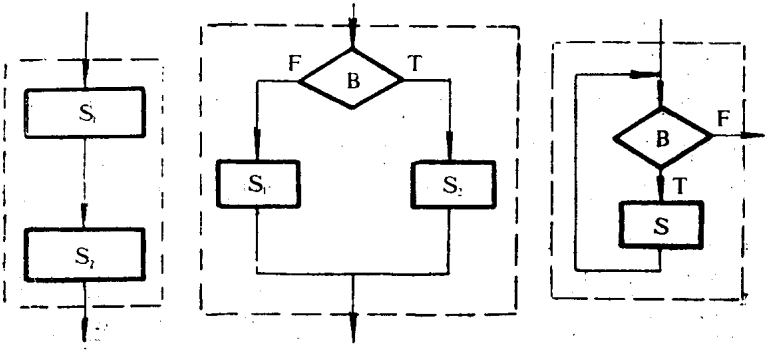
我们主张程序结构清晰，易于读写，易于验证，追求可靠性好、可维护的“好结构”程序。一旦效率与“好结构”发生矛盾，我们宁可降底效率，而确保程序的好结构。

考虑到大部分学校已在教学中选用了PASCAL语言，且该语言也适合方法学的教学、基础概念的系统训练，本教材中的程序我们采用标准PASCAL语言编写。这无论是从逻辑叙述，推理方法，思维习惯，还是从易于理解等考虑，都是比较合适的。

§1.3 结构程序设计概念

用结构程序设计方法设计一个程序，需要对整个问题进行不断分解、求精，直至每一个子问题都简单明了，即可以轻而易举地用高级语言来描述。用这种方法设计的程序结构良好，易读、易写、易于修改和维护。今天，这种设计思想与方法已得到广泛的承认和应用。

七十年代可以说是结构程序设计统治的年代，当时出现了许多结构程序设计的定义。曾有一段时间不少人误认为没有GOTO



(a)顺序　(b)选取　(c)重复

图1.2

语句的程序就是结构程序。在众多的定义中，有一个共同的特征，那就是结构程序有顺序、选取、重复三种基本控制结构和程序块只具有“一个入口和一个出口”的原则。如图1.2所示，其中B是布尔表达式，S,S_1,S_2 是语句，可以嵌套。

当然，在实际应用中，还允许使用图1.3所示的两种结构。

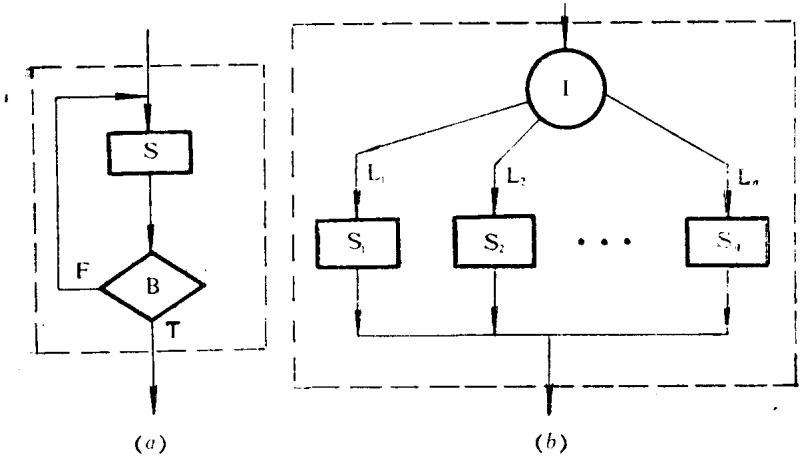
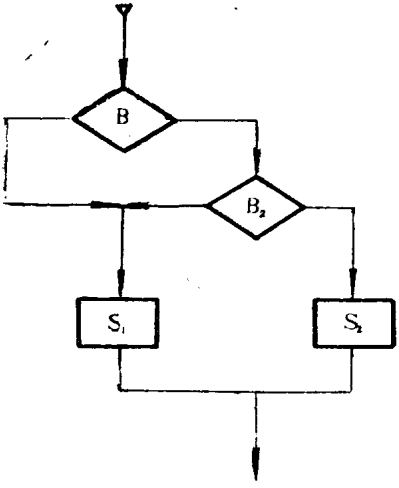


图1.3

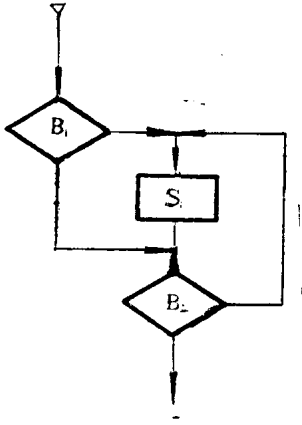
简单地说，在程序中只用以上几种基本结构来表示的程序，我们就称为结构程序。

在传统的程序设计框图中，经过实践应用发现，图1.4所示五种情况是属于非结构化的。它们都是单入口多出口。我们常将前二种情况称为异常选择回路，而将后三种情况称为异常循环出口。不难看出，虚线框内程序的特征是有二个出口。

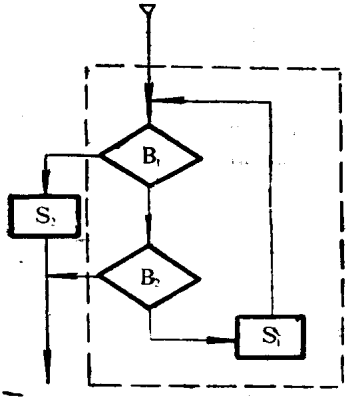
既然已找到了一些非结构的框图成分，那么我们自然会想到能否将这些非结构的框图转换成结构化框图成分。请先看一个例子。对于图1.5(a)所示框图，根据程序分析知识能找到图1.5(b)所示的等价框图。



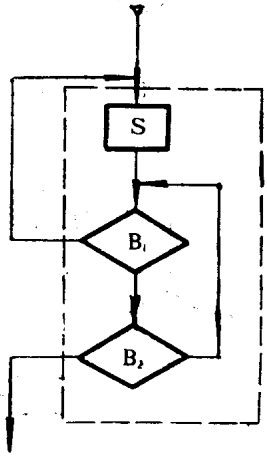
(a)转入分支



(b)转入循环体

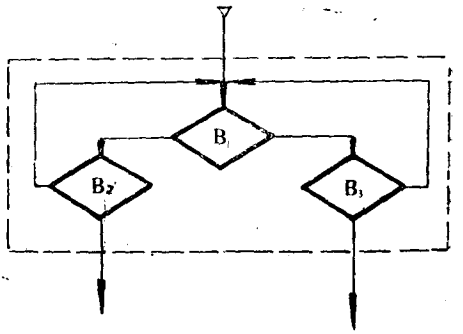


(c)循环多出口



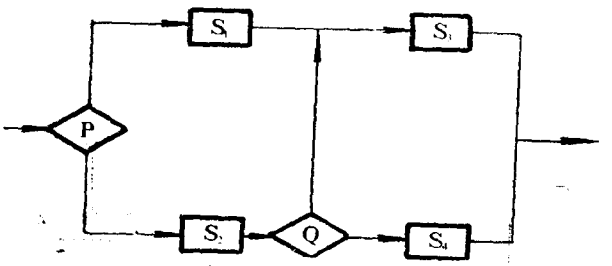
(d)重选循环

图1.4

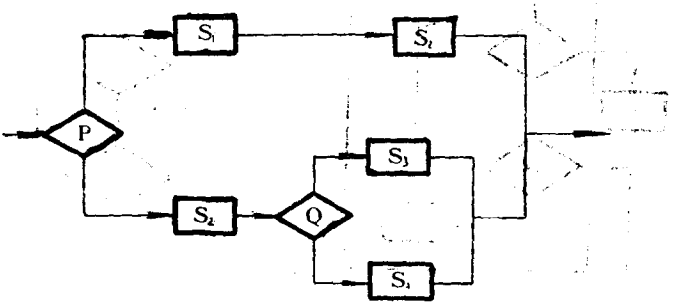


(*c*)并行循环

图1.4(续)



(*a*)



(*b*)

图1.5

显然，图1.5(b)所示框图是一种 **if** P **then** A **else** B；的结构，其中A是S_1；S_2 的复合结构，B是S_2；**if** Q **then** S_3 **else** S_4 结构，它们都是基本控制结构。

早在1966年，Bohm和Jacopini就证明了程序中只要有三种基本形式（顺序、选取和重复）控制结构就足以表示出其它各式各样形式的结构。

那么，能否将程序中所有的非结构成分转化为结构成分呢？回答是肯定的。实际上，在很大程度上转化主要是对转移语句的处理。

下面，我们介绍非结构化框图转换的几种方法。为方便起见，引入记号→。←，表示这个集合为二个输入端和一个输出端。

（1）交换法

图1.6(a)可以等价交换成图1.6(b)，其作用是相同的。作为一个例子，我们来分析图1.7所示非结构框图的转化。首先，我们可将其转换成中间结果框图，如图1.8所示。然后，按照交换规则可得到图1.9所示框图。

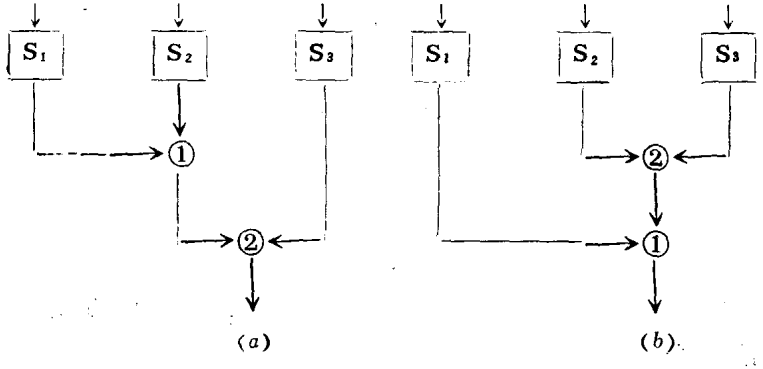


图1.6

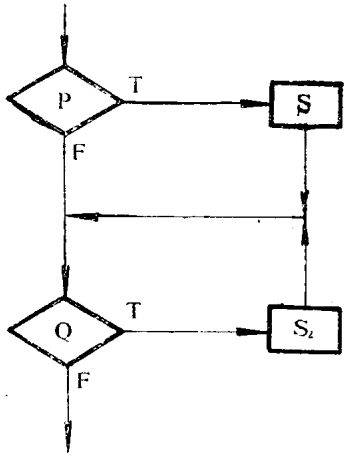


图1.7

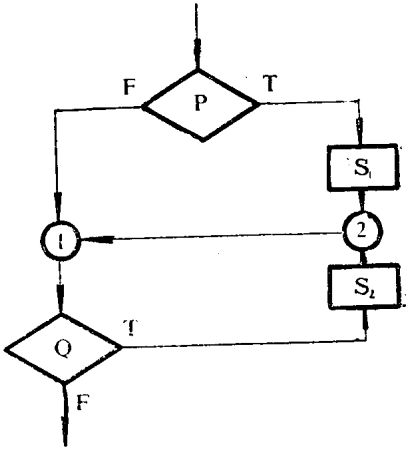


图1.8

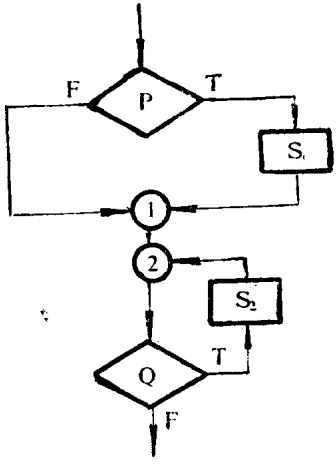


图1.9

可以看出，图1.9所示框图是if P then S_1与 while Q do S_2结构的结合，故是结构化的。

(2) 转位法