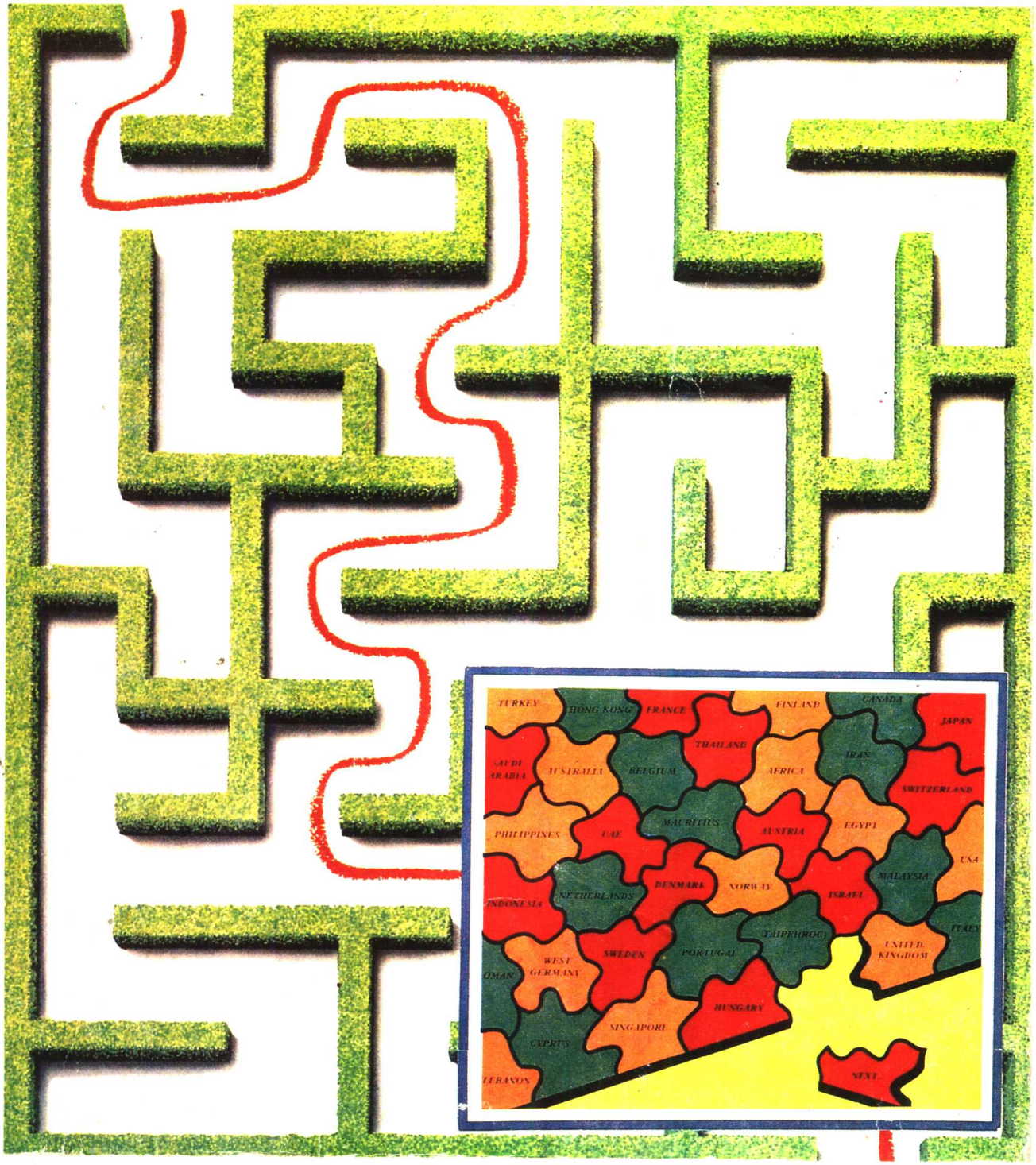
《现代电子技术》增刊

结构化程序设计教程

方法与练习

宁正元·王秀丽 编著



陕西电子杂志社

结构化程序设计教程

(方法与练习)

宁正元
王秀丽 编著

陕西电子杂志社

内 容 简 介

本书从同构的观点出发，采用独立于任何特定语言、易读易理解的伪代码介绍各种基本的程序设计方法。主要介绍了自顶向下逐步求精细化、分而治之、模块化、结构化、杰克逊法等与程序控制结构有关的结构化程序设计方法；同时也较深入地介绍了运用各种数据结构解决实际设计问题的具体技术。提供各种例题和习题千余道，涉及文理工管、商贸金交、农林医体等多学科领域；涉及计算机专业的数据库、数据结构、编译原理、操作系统、软件工程、系统结构、信息处理、智能模拟、辅助教学等多方面内容。

本书可作为各种程序设计课程的补充读物和习题集；更适宜作为计算机水平考试和非计算机专业学生等级考试等考前的复习指导和练习手册；也可供从事计算机工作的科技人员和各种程序设计课程的教师阅读参考。

前　言

随着计算机技术的飞速发展，其应用触角几乎已进入各行各业各个领域。为适应现代社会的需要，人们都迫切要求学习计算机知识，以解决本行业的各种问题。程序设计方法的学习与训练已至关重要，为愈来愈多的人们所共识。国家 90 年成立了软件水平与资格考试委员会，每年对计算机专业人员举行水平与资格考试；陕西、福建、上海等许多省市也相继做出决定，对高校非计算机专业学生实行计算机应用能力等级考试。而程序设计在水平考试和等级考试中占据着极端重要的位置。

但乞今为止，在各种程序设计书籍中，系统性和完整性比较薄弱。不仅普及性课程，甚至在计算机专业的程序设计课程中，都倾向于从简单易做的题目中选择少量练习题，难度上、数量上都显得有些不足，很难适应水平考试和等级考试对程序设计课程的要求。为此，作者在认真总结十多年讲授 ALGOL、BASIC、FORTRAN、PASCAL、C 等程序设计课程经验的基础上，用同构的观点看待各种不同的程序设计语言，力图编写一本题材独立于任何特定语言，又适合于各种语言程序设计训练的，设计方法比较系统完整、选题有一定深度和广度、难度分级渐进式练习的指导书籍，以帮助各级各类读者在各种程序设计课程中使用，较快提高程序设计的能力与水平。

本书是作者在认真研究消化国外有关资料的基础上，结合我国程序设计教育的现状、特点与要求，花了近两年时间两易其稿最后完成的。着重介绍了各种基本的程序设计方法，如自顶向下逐步求精细化、分而治之、模块化、结构化、杰克逊设计方法等；也介绍了许多重要的具体设计技术，如三值选择、结束标志、循环效率、检索排序、归纳划分、递推递归等。除了介绍与三种基本控制结构有关的方法技术，也对与数据结构有关的设计方法与技术进行了较深入的讨论（如栈、队、链、树、图、表、库、记录、文件等）不仅可使读者学会写一般程序，还可学会运用数据结构与归纳划分等技术解决较复杂问题。在例题与习题的选择上数量充足、范围较广。涉及文理工、商贸管、农林医、金交体等多学科领域；涉及计算机专业的数据结构、编译原理、操作系统、软件工程、智能模拟、信息处理、辅助教学、数据库、系统结构等多方面内容。以期通过这样的安排能激发不同类型读者对程序设计的浓厚兴趣，进而达到提高设计能力与水平之目的。

在讲述例题的算法描述上，我们采用了独立于任何特定语言，又与每种语言中语句有着确定关系、符合结构化原则、接近自然语言易读易理解的伪代码描述。这样避免了因具体语言语法规定等次要问题而干扰程序设计这个中心，易为各类读者所接受。当然读者总是要联系某种特定语言学习练习并上机调试实践的。为此，我们专劈一章介绍程序设计的辅助手段，介绍本书所用伪代码与高级语言之间的结构转换关系，并给出了伪代码程序到几种常用高级语言程序（如 BASIC、C、FORTRAN、PASCAL 等）的转化实例。运用这些方法，读者可以很方便地把伪代码程序转化为特定语言书写的程序。

在该书中，是以程序设计的类型分章，各章又以难度等级分节的。提供各种例题习题共千余道（其中例题 112 道，随节练习题 260 道，综合练习题 700 道）。这些题目都是仔细地按题型分类，各类中又以难易复杂程度细分了等级的，较好地体现了由浅入深、循序

渐进的学习训练原则．在给出练习题时，不仅给出题目要求，同时也阐述了问题的输入输出数据及类型，对某些难点给予必要的提示，这样可使读者逐步养成仔细分析思考给定问题的良好习惯，逐步提高分析设计能力．每组练习题目，较前面的是为配合各种程序设计语言课程安排的，是每人都应该练习的；而较后面的那些题目，是针对能力考试和水平考试的练习安排的，即使较有天赋的学生也会感到具有一定难度，这样可满足不同层次读者的不同需求．

本书既可作为程序设计方法学的教材；又可作为某种程序设计语言课的补充读物和练习题集；还可作为程序设计课程之后进一步提高的方法指导书籍和练习手册．若作为计算机专业水平考试和非计算机专业等级考试以及各级各类程序设计竞赛的考前训练使用，无疑是非常合适的．也可供各种程序设计课程的教师选题或参考。

在本书写作期间，西安市计算机协会理事长、西北大学计算机科学系罗景仁教授对本书手稿提出过许多宝贵的指导意见；也得到了西北大学计算机科学系主任刘德安副教授、福建林学院森工系主任黄祖泰副教授、教研室主任高君强副教授以及学友胡爱珠、邢为民、唐永明、吴建宁等同志的关心和支持；作者在此一并表示真诚的谢意。本书能较快与读者见面，还要感谢《陕西电子》杂志社为读者不拘一格多出书、出好书的出版服务宗旨；感谢该社领导和全体编、排、校、印人员所付出的艰辛劳动。

鉴于时间紧促和作者水平所限，不足和错误之处在所难免，欢迎计算机科学界同行和广大读者不吝赐教。

作　者
1994.4

目　录

第一章　程序设计的基本概念

程序设计的最终结果总是要用某种特定的程序设计语言中的语句来表示，而程序设计语言却又各种各样。为了用独立于任何特定语言的方法介绍程序设计方法，本章从同构的观点出发剖析程序设计语言的控制结构和数据结构，进而介绍结构化程序设计的基本思想和实现方法。为此，我们先简单介绍计算机与程序设计语言发展的基本情况。

§1.1　计算机与语言

自从第一台电子计算机"ENIAC"于美国宾夕法尼亚大学诞生之后，带来了人类在计算技术发展史上的一场深刻革命。电子计算机从电子管、晶体管、集成电路到大规模和超大规模集成电路，从大、中、小型到巨型、微型机，从每秒5000次运算到每秒数十亿次运算，从实验室研究到社会化大规模生产制造，从尖端技术领域使用到步入千家万户普通居民家庭……，这一切都足以说明这场革命如排山倒海，迅猛异常势不可挡。特别是微型计算机的飞速发展，使得各行各业各个领域都用上了计算机。在今天，计算机已不只仅用于计算，在工业自动化控制，辅助设计制造、人工智能模拟、图形图象识别、文字信息处理等许多方面都已得到了广泛的应用。这种应用正在继续向纵深发展，已经和正在带来人类历史上空前的新技术革命浪潮。那么，什么是计算机呢？计算机是怎样工作的呢？人又如何使用计算机呢？

在今天，人们所讲的计算机已不是一台硬设备的"裸机"，而是一个完整的计算机系统。一个计算机系统，通常由硬件系统和软件系统两部分组成。硬件系统是指构成计算机系统的物质设备，如主机、外设等。软件系统是指为了管理、使用和发挥计算机效率的各种程序，如操作系统程序、编译系统程序、各种支撑软件程序和各种应用系统程序等。

计算机的主机包括运算器、存贮器和控制器，又常把运算器和控制器合称作中央处理机（CPU）。存贮器是计算机存贮数据信息的记忆装置，它又有内存贮器和外存贮器之分。内存贮器一般由磁芯或半导体元件构成，存取速度较快，但成本较高，故设计的容量一般都不太大；而外存贮器（如软盘、硬盘等）一般是由磁性介质构成，成本较低，但存贮速度也较慢。即使用内存贮器又使用外存贮器，有效地解决了容量、速度和成本之间的矛盾。控制器是计算机的中枢机构，它负责协调各部件有条不紊地工作：一般由指令寄存器、指令计数器、指令译码器和操作控制部件几部分组成。内存中的指令是逐条取入指令寄存器中，经指令译码器译码辩认，由操作控制部件发出相应的各种操作控制信号。而运算器是计算机执行算术运算和逻辑运算的装置，它的主要组成部分是一个全加器和一些移位线路。通过使用适当的数制（二进制）和码制（原码、补码、反码等），运算器就可以完成加、减、乘和除等基本算术运算；辅之以相应的指令系统，就可以完成其它更复杂的运算和处理。

在计算机中，所有的数据、指令和信息符号都是以二进制数形式表示的，或者说计算

机只认识0和1这两种信号。所以，计算机在出厂时配置的机器指令系统中的每一条指令，都是一些赋予了专门意义的0、1代码序列。早期的电子计算机使用，就是用这样一些0、1代码指令（称作机器指令）编写程序的。通常把机器指令系统规定的语言称作机器语言，而把用机器指令代码编写的程序称之为机器语言程序。当时，机器语言是人与机器之间的唯一接口，人要使用计算机算题，就必须先掌握所用机器的机器指令系统，学会用机器语言编写程序。所以编写程序仅是少数计算机专家们的事，而且不同机型的研制者要想使用对方研制出的机器都是十分困难的事情。用机器语言编写程序，既不易读又极易出错，且不易查错和排错。为了寻求出较好的方法解决这些问题，于是便产生了汇编语言。

汇编语言的前身是符号语言，它是把每一条机器语言指令都用相应唯一的指令符号来代替。在编写程序时用指令符号编写，地址使用相对地址，数据使用易于转换为二进制的八或十六进制，这样便于设计、检查和修改。在上机计算前再"代真"，即把指令符号用相应的机器指令代码来替换，把相对地址换为实际地址（也称绝对地址），把数也换成相应的二进制形式。这种代真过程比较繁锁，且代真的过程中又可能出现新的错误，所以人们编制了专门的汇编程序来做这项代真工作，也就是说汇编程序完成由汇编语言程序到机器语言程序的自动翻译工作；同时把某些专门的常用计算成设计成子程序，用相应的汇编符号来表示，这样在较大程度上改善了程序设计的状况。但是，汇编语言毕竟是和机器语言在指令上一一对应着的，是面向机器的低级语言。人们要编写程序，需要考虑指令、数据的地址分配等与特定机器有关的许多具体问题。低级语言的这种局限性，使得计算机的使用和推广受到了很大的限制。计算机的程序设计和应用，仍然为少数计算机专家所有。

把计算机从计算机科学家手中解放出来，变成为广大人民群众手中的工具，是高级程序设计语言诞生之后的事情，也只有在这时，计算机的推广、普及和应用才变为现实。高级程序设计语言也称作算法语言，是面对问题处理过程的语言，统称作面向过程语言。这些语言与具体的机器无关，接近数学公式表示，增强了可读性和易排错性，降低了程序设计的难度。高级程序设计语言一经出现，就如雨后春笋一般，各种各样用途的高级程序设计语言、同种高级程序设计语言的各种不同版本都相继问世。目前至少已有几百种高级程序设计语言，国内外比较通用的高级语言也有几十种之多。最常见应用最普遍的高级语言有 Ada、ALGOL、BASIC、C、COBOL、FORTRAN、LISP、PASCAL、PL／1、SNOBOL 等。现在已出现了一些面向对象的程序设计语言，如 DBASE、FOXBASE 等等。程序设计难度将会愈来愈低。

前面讲过，计算机只认识0、1代码，自然也就不认识高级语言程序了。如同汇编语言程序需要用汇编程序进行翻译一样，高级语言程序也有一个从高级语言源程序到机器语言目标程序的翻译过程。实现这种翻译一般有两种方式，一种是编译方式，一种是解释方式。解释方式是由一个称之为"解释程序"的系统程序把高级语言程序中的语句逐句翻译成为机器指令代码，翻译一句就立即执行一句，是一种边翻译边执行的工作方式。程序设计语言 BASIC 就是采用这种解释方式进行翻译的。而编译方式则是由一个称之为"编译程序"的系统程序把高级语言程序逐句翻译成机器指令代码，在进行适当的代码优化工作以后，形成质量较高的目标代码程序整体，最后一次执行的。这种翻译方式是一种整体翻译集中执行的方式。程序设计语言 C、FORTRAN、PASCAL 等大多数语言都是采用这种

方式翻译的。

计算机是一种运算速度快、计算精度高、具有记忆和逻辑判断能力的自动电子装置。它的运算速度来自构成它的电子元器件速度，它的精度来自表示数据信息的二进制位数，它的记忆能力来自存贮器和寄存器，它的逻辑判断能力来自使用二进制可以方便地进行逻辑代数运算，而它的自动化程度来自于程序。然而程序是要靠程序员编写和测试的，程序员要根据实际问题确定相应的计算方法，构造数学模型，画出程序执行的控制流程图或写出伪代码，反复修改正确无误后再选择适当的程序设计语言编写源程序，并上机反复调试测试，直到满足实际需要为止交付使用，这就是程序设计的全过程。

程序员的职责是设计出高质量的程序。所谓高质量程序的基本要求是在满足结构化程序设计原则，做到可靠正确、清晰易读易排错的前提下，在程序设计的过程中程序员要能正确地处理好时间与空间的关系，设计出计算速度快、运行时间少、程序短少精练、选择数据结构适宜、占据内存空间少的高效率程序，同时还要注意做到所设计出来的程序通用性强、可移植性好等方面。换句话说，程序员要做到处理好良好的结构与较高的效率、较少的执行时间与较小的使用空间、具体问题与同类问题，特定的具体机器与其它机型等方面的关系，学会把握主要矛盾，设计出高质量的程序交付用户使用。

§1.2 程序设计语言的控制结构

面向过程的程序设计语言有各种各样，都逐一地去把每一种语言搞清楚，是要花费巨大的精力和相当长时间的，不仅不可能做到，其实也没有必要。各种程序设计语言提供的设施大体上都是相同的或相似的，无非是具体设施的多与少、功能上的强与弱、使用上的方便程度如何等一些区别罢了。从可以实现的功能结构上讲，我们可不严格地讲这些语言都是同构的。在控制结构方面，不外乎三种基本结构及其它们的某些变形；在数据结构方面，各种数据结构或难或易一般都可以实现。这一节，我们用同构的观点对程序设计语言的控制结构进行一些必要的讨论。为此，我们先引入语言设施中本原操作（或曰基本操作）的概念。

所谓本原操作，是指程序设计语言中提供的那些最小操作单位的设施，这些最小操作单位不能再分解为更小的一类操作。在对具体的程序设计任务提出之后，确定了计算方法，构造出数学模型；然后就是自顶向下逐层分解任务，画出程序流程图或写出伪代码；最后用某种语言来编写程序。这个自顶向下逐层分解的目标，就是分解到语言中所提供的本原操作设施一级，这样才能由本原操作的某种控制序列构成源程序。那么，程序设计语言中的最小操作单位究竟是些什么？程序设计的过程中到底应该如何把握？这往往是初学程序设计的人比较头疼不易掌握的东西。

本原操作是依不同的程序设计语言在提供本原操作的数量上和操作的大小上而有所区别的。对于面向机器的机器语言和汇编语言，其本原操作就是语言提供的一条条指令；而对于面向过程的高级程序设计语言来说，其本原操作就是那些一条条基本语句，是比低级语言更大的操作。如各种高级语言中的：

* 给一个或多个变量提供数据的语句，它从终端键盘或某输入文件中为相应的变量提供数据。如 BASIC 语言中的 INPUT 语句，READ / DATA 语句，RESTORE 语句；

C 语言中的 getc 语句，getchar 语句，scanf 语句和 read 语句；FORTRAN 语言中的 READ 语句、FORMAT 语句；PASCAL 语言中的 READ 语句和 READLN 语句；等等。

* 执行运算或处理的语句，它完成各类表达式的求值，变量信息值之间的传送等任务。如 BASIC 语言中的 LET 语句；C 语言中的赋值语句（=）和自反运算操作设施；FORTRAN 语言中的赋值语句（=）；PASCAL 语言中的赋值语句(:=)等。

* 输出变量、表达式值或字符信息的输出打印语句，它把结果输出到终端显示屏幕，打印机或其它输出文件中去。如 BASIC 语言中的 PRINT 语句，C 语言中的 putc 语句、putchar 语句、printf 语句和 write 语句，FORTRAN 语言中的 WRITE 语句和 FORMAT 语句，PASCAL 语言中的 WRITE 语句和 WRITELN 语句等。

此外还有各种子程序的调用语句和各种基本运算操作等等。在这些本原操作的基础上，各种程序设计语言都相应提供了若干构造型语句，这些构造型语句提供了对本原操作语句流的一些控制方法和手段。各种高级程序设计语言所提供的控制结构，都不外乎是三种最基本的控制结构——顺序结构、选择结构和循环（或重复）结构和这三种基本结构的组合或变形。

顺序结构是指语句的执行顺序是从前到后依次执行的结构。一个程序从总体上把握是若干基本结构的顺序结构；从构成总任务的各子任务上讲，其内部操作的执行顺序也是若干本原操作或（和）结构操作的顺序序列，最低一级子任务是本原操作的顺序序列。在程序设计语言中提供的顺序结构为复合语句，一般是由 BEGIN 开始 END 结束的一个语句序列。也有些程序设计语言没有提供复合语句设施，如 BASIC 语言和 FORTRAN 语言等，其执行顺序主要体现在语句标号的大小或语句书写的先后次序上。

选择结构是依据某个条件是否得到满足而决定程序执行的动作或方向的控制结构。最基本的选择结构为匹配式的条件选择和非匹配的条件选择两种形式；还有由它们派生出来的多选择结构大多数程序设计语言都有提供。几种最常用语言提供的选择结构如下：

BASIC:	IF——THEN语句;
	IF——THEN——ELSE语句;
	ON——GOTO开关语句等.
C:	if——else语句;
	Switch语句;
	Case语句;
	default语句;
FORTRAN:	算术IF语句;
	逻辑IF语句;
	块IF语句;
	计算转向GOTO语句.
PASCAL:	IF——THEN—ELSE语句;
	CASE语句.

有了这些选择结构，程序员在程序中就可依据某些算术值或逻辑值的不同而选择执行不同的操作，从而实现对程序流程的灵活控制，准确或正确地执行既定的计算或处理任务。

循环结构是为了重复执行某种处理或任务若干次而设计的控制结构。最基本的循环结

构是当型循环和直到型循环两种结构；许多语言提供了由它们派生出来的步长型循环结构。在基本 BASIC 中只提供了 FOR——NEXT 步长型循环，没有提供当型循环和直到型循环；但当型循环可由 IF——THEN 语句和 GOTO 语句联合使用来实现，直到型循环仅单独使用 IF——THEN 语句就可实现；在扩展 BASIC 中大都提供有专门的当型循环。在 C 语言中有专门的当型循环 while 语句和专门的直到型循环 do——while 语句，也有专门的步长型循环 for 语句。在 FORTRAN 语言中也没有专门的当型循环和直到型循环，需要用 IF 语句和 GOTO 语句来实现；但专门提供有步长型循环 do 语句。在 PASCAL 语言中提供了当型循环 WHILE——DO 语句，直到型循环 REPEAT——UNTIL 语句和递增型步长型循环语句 FOR——TO——DO 以及递减型步长型循环语句 FOR——DOWNTO——DO。

这三类循环的关系是：凡能用步长型循环解决的问题，就一定可以用当型或直到型循环来解决，反之不然；凡能用当型循环解决的问题，就一定可以用直到型循环解决，反之亦然。进一步说，步长型循环解决问题的范围较小，只是对那些循环次数确定的或可预知的循环问题适用，对于那些循环次数难以确定的循环问题只有利用当型循环或直到型循环来解决。当然，在可使用步长型循环的情况下，不宜使用当型循环或直到型循环来解决；因为步长型循环简单明快，设计难度小，方便好用。当型循环和直到型循环可用于设计任何循环问题，同一问题设计的当型循环和直到型循环之间可以相互转化。其差别仅在于当型循环是先比较条件是否得到满足决定其后是否执行循环体各语句，而直到型循环是先执行循环体各语句然后判断是否满足退出循环的条件。直到型循环至少执行一次循环体，而当型循环则有一次也不执行循环体的可能性。在进行当型循环和直到型循环之间的转化时，一定要注意其条件正好相反，也就是说在转化时只要互取其相反条件即可。

程序设计的主要任务就是在选取适当数据结构的基础上，利用顺序、选择和循环这三种基本控制结构，把逐级分解而得到的一系列本原操作组织起来，形成某种特定语言如 BASIC，C，FORTRAN 或 PASCAL 等语言书写的源程序。早在 1966 年，Bohm 和 Jacopini 就已经证明了在程序设计语言中只要有这三种结构，就足以表示出各式各样的其它形式的结构。换句话说，不论是什么样的问题，不管问题的规模与复杂程度有多么大，只要用这三种基本结构编程，问题都可以得到解决。所以，这三种基本结构的熟练掌握，是程序设计的基础，是一个程序员应该具备的最起码能力，必须引起每个程序员的足够重视。对这三种基本结构的进一步理解和掌握并达到熟练运用的程度，是要通过大量的程序设计练习来实现的，当然最好结合某种特定语言进行必要的上机操作实践。

§ 1.3 程序设计语言的数据结构

数据结构在程序设计中占据着十分重要的位置，程序等于数据结构加算法是计算机科学界早就公认的观点。所以在程序设计语言中不仅提供有与程序流程有关的控制结构，同时也提供有与程序中数据信息组织有关的数据结构，特别是从七十年代起所设计的高级程序设计语言大都提供有数据类型的概念。通常在语言中提供有称之为基本类型的基本数据结构，同时也程度不同地提供了如何由基本类型构造出所需的称之为构造类型的复杂数据结构的一些方法和手段。

在用高级语言所写的程序中，每一变量、常量、函数或表达式的值，都隶属于确定的数据类型。虽然在程序执行期间变量的值在不断地变化，但变量的所有可能的取值以及在这些值上可以执行的操作都在程序中或明显地或隐含地规定了。所以说数据类型的概念，是程序设计语言和程序设计过程中的一个非常重要的概念。

数据类型的概念，具有以下显著特征:

(1) 类型决定了变量或表达式所有可能取值的全体成员集合;

(2) 每一个值隶属于且仅隶属于某一个类型;

(3) 任何常量、变量或表达式的类型，都可以从其形式上或所处的上下文关系中推断出来，无须了解在程序运行时计算出的具体值;

(4) 每一种操作都要求一定类型的操作数据，且得出一定类型的操作结果;

(5) 一种类型的值及其在该类型上规定的基本操作的性质可由一组公理来阐明;

(6) 高级程序设计语言使用类型信息去防止程序中出现无意义的结构，又由类型信息确定在计算机中的数据表示和数据处理方法。

基于这些准则，在七十年代以后出现的高级程序设计语言如C语言和PASCAL语言等，都大体上提供了三种不同范畴的数据类型——简单类型、结构类型和指针类型。而较早出现的一些语言如BASIC和FORTRAN等，没有指针类型，仅有简单类型和构造类型，而且提供的简单类型尤其是构造类型在数量上也相对较少。

所谓简单类型是程序设计语言所提供的那些最基本的类型，在一个类型中不含有其它类型成分的非结构数据类型。通常含有整型INTEGER、实型REAL、布尔型BOOLEAN和字符型CHAR这四种标准类型，此外还可含有枚举、子界等一些其它简单类型。它们是构成所有结构类型的最基本类型成分。在BASIC语言中含有四种标准类型，但在基本BASIC中没有提供类型定义或说明的方法和手段，要靠隐含规则和上下文关系来辩识。在C语言中的简单类型有整型int、字符型char、单精度浮点型float、双精度浮点类型double、长整数类型long和无符号整数型unsigned等，提供的类型比较细，也比较丰富。在FORTRAN语言中提供了整型、实型、双精度实型、复型和逻辑型，但允许在程序中对整型和实型运用隐含规则而不必作变量的类型定义和说明。在PASCAL语言中提供了四种标准类型和枚举类型与子界类型，但没有隐含规则，变量数据的类型必须进行显式说明。

常见的结构类型有数组、记录、集合、字符串和文件等。它们都是由已知类型通过一定的构造方法构造出来的新类型，已知类型称之为新类型的构造成分类型，当仅只有一个构造成分类型时又常常把这个构造成分类型称作为基类型。当然，允许构造成分类型本身又是某个构造类型。在BASIC语言中提供的构造类型有数组、文件和串，在C语言中有数组、结构、联合和文件，在FORTRAN语言中有数组和文件，而在PASCAL语言中有数组、记录、集合、串和文件。每种高级程序设计语言都最起码提供了数组类型，这是科学计算和数据处理应有的基本数据类型，记录和文件是进行信息处理的基本数据类型。只有具备或可构造出相应的数据类型，程序的控制结构才真正有用武之地。

还有一些程序设计语言提供了指针类型数据，如PASCAL和C都提供了；也有许多语言如BASIC和FORTRAN等没有提供，但可利用数组等类型来模拟实现指针结构。有了指针类型或指针类型的概念，程序员在程序中就可以实现堆栈、队列、链表、循环链

表、树、二叉树和有向图等许多复杂的数据结构，并用之组织所要求解问题的数据信息，形象地描述刻划求解的问题，降低程序设计的难度。

各种不同的数据类型可取值的范围不同，可执行的运算或操作也相应有别。例如，整型数据的取值范围，依赖于实现该高级程序设计语言的具体机器，在16位字长的微机上的整数范围是-32768～32767之间，可执行的运算为加、减、乘、整除、取余和各种比较操作。实型数据的范围也依赖于所实现的具体机器的实数表示方式，可执行的运算和操作是加、减、乘和除以及关系比较等。布尔型数据的取值只能是真或假（True或False，也可为1或0），可执行的操作可以是逻辑乘and、逻辑加or和逻辑否定not等。字符型数据的取值可以是在计算机上能表示的任何字符，可执行的操作是关系比较、求前趋后继等。而构造型数据的取值范围和可执行的操作种类完全依赖于构造成分类型或基类型。如数组类型元素的取值和操作与基类型完全一致；对记录类型各组成成分（或各字段）的取值与操作，与相应组成成分（或字段）的类型完全一致；集合类型的基类型的基数必须是有限的，其执行的操作可以是集合的交、并、补、差和判元素是否在集合中等；串类型的基类型是字符，可进行的操作是求串长、求子串位置、串的联结和串的各种关系比较等，如此等等。在使用各种不同的数据类型时，一定要注意它们的取值范围和在其上可施行的操作种类，这样才能避免数值越界和非法操作等错误出现。

§1.4 结构化程序设计及实现

在六十年代末到七十年代初出现了一些大型的软件系统，如操作系统、数据库系统等，给程序设计带来了新的问题。大系统往往需要花费大量的人力和财力，然而研制出来的软件却常常可靠性差、错误多，系统维护和修改也相当困难。一个大型操作系统的研制有时需要几千人年的工作量，而在所得到的系统中又常常会隐藏着成百上千个错误，当时人们称这种现象为"软件危机"。软件危机震动了整个计算机科学界，软件工作者们开始又重新研究起程序设计中的一些最为基本的问题了。诸如，程序设计的基本组成部分是什么？应该用什么样的方法来设计程序？程序设计的主要方法和技术应该如何规范化？等等。这才导致了结构化程序设计和软件工程学的发展。

Dijkstra在一九六九年首行提出了结构化程序设计的概念，强调从程序的结构上和风格上来研究程序设计方法，提倡利用三种基本结构进行规范化程序设计，使程序具有良好的结构框架。经过几年的争论、探索和实践，结构化程序设计的应用确实取得了很好的成效。用结构化程序设计方法得到的程序不仅在结构上良好，清晰易读易写，而且易维护、易排错、易于验证正确性。结构化程序设计方法的迅速推广和流行，其意义远不止于当前的这种实用价值，更深远地意义在于它向人们揭示了研究程序设计方法的重要性。用工程学的方法研究软件开发和设计的软件工程学就是一个很好的例证。

程序设计方法中有三个基本原则——抽象原则、枚举原则和归纳原则。为了解决一个复杂的实际设计问题，由于人脑思维能力的局限性和要处理实际问题的复杂性，决定了程序员不可能一下子就能触及问题的具体细节；在分析清楚问题的目的要求之后，总是先要建立起抽象的数学模型，在抽象数据上实施一系列操作，设计出抽象算法。这些数据和操作反映了问题的本质和属性，而将其具体细节全部抽象掉了。然后作为下一步要考虑的问

题才是这些抽象数据和操作的具体实现．在抽象级程序员主要只关心"做什么"，而在实现级才去考虑"如何做"．当然这种抽象技术的运用是与掌握特定程序设计语言的抽象数据类型和操作设施难以分开的，但由于抽象技术的使用可以把大问题划分为若干相对独立的子问题，故只涉及局部环境和条件，可以相对容易地逐一得到解决．

枚举原则也称作穷举原则，它是把实际问题的各个环节和抽象数据类型的各种取值都考虑周全的一种方法或技术．运用枚举原则可以使问题的求解算法设计得全面、完整和准确．程序设计语言中条件选择语句（条件句、开关句、分情况句等）是枚举原则的具体应用．而归纳原则的运用，可以寻找出求解问题中某些规律性的东西，从而降低问题求解的难度，程序设计语言中的循环结构和递归子程序结构，都是归纳原则的很好应用．

程序设计方法研究的内容，包括数据类型抽象、程序的形式化推导、程序的变换、程序的自动生成、程序的正确性证明或验证、程序可靠性和可维护性、程序的结构和效率等许多方面．结构化程序设计是其中一个重要的组成部分．前面提到了结构化程序就是用三种基本的控制结构设计的具有良好结构的程序，那么，需要采用怎样具体方法和技术来保证得到的程序结构是由三种基本结构组成的呢？或者说结构化程序设计的具体实现方法和手段是些什么呢？

具体地说，结构化程序设计是采用

(1) 自顶向下逐步求精的分析设计方法；

(2) 分而治之的分割划分技术；

(3) 模块化的组织结构形式。

来进行程序设计的．

在拿到一个设计问题之后，两种不同的处理办法可供选择，一种是自顶向下逐步求精细化，另一种是自底向上逐步堆砌积累．由于人脑思维很难一下触及问题细节，所以自底向上的方法较难运用．即使运用也很难设计出清晰的程序层次结构，前后内容不易相对独立，当需要对设计好的某个问题进行修改时，会株连程序的其它部分要做相应的修改，有时甚至得全部推倒重新设计，显然自底向上的方法不是结构化程序设计需要的好方法．结构化程序设计提倡自顶向下的逐步求精分析设计方法，从求解问题到得出一系列本原操作逐层展开细化．而对所设计出的程序进行测试修改时是先对每一个子问题进行测试而最后联调测试，这样测试规模小、错误少，测试难度低、易修改．也就是说提倡自底向上的测试修改过程，这就如同要建一幢楼房的图纸设计过程一样，先总体规划确定建筑方案，再进行各部分的轮廓设计，最后进行详细设计，而决不会在没有制定出总体方案就先设计楼道和卫生间．在完成设计后交付有关部门评议审核，根据收集到的意见修改一个个的详细设计以至达到对总体方案的修改．楼房的施工阶段一砖一瓦实现每一个局部部分最后形成实际的楼房，和程序的执行阶段一句一句执行每个子任务最后得出问题的求解结果一样，是自底向上的过程；楼房建成后的检查验收阶段和程序设计好的测试阶段一样，也是局部到全局的自底向上过程．

自顶向下逐步求精的分析设计方法，其分析设计过程是将问题求解方法由抽象逐步到具体化的过程，程序员应该通过程序设计的实践训练逐步熟练掌握．采用这种设计方法，易于保证和验证程序的正确性．由于求精细化的过程是逐层进行的，每层的细化工作都不会太复杂，只要从问题开始的每一层设计都是在检查正确之后再进行下一层次的细化，每

层设计都正确，就保证了整个程序是正确的。由于结构层次清晰，逐层检查验证或证明程序的正确性也就方便容易多了。

如果程序员拿到的是一个比较复杂、难度较大的问题，就要尝试采用分而治之的分割划分技术，把要求解的问题划分为若干相对较小的容易解决的小问题，通过一个个小问题的解决求得整个问题的解决。分而治之的划分分割技术，是军事上"划整为零，各个击破"的战略思想在程序设计中的应用，目的是为了逐级降低问题求解的难度直到全部获得解决。比如说欲求 N!，可以把它划分为求 (N-1)! 与 N 的乘积问题；为了求出 (N-1)!，又可划分为求 (N-2)! 与 N-1 的乘积……继续这种划分直到 2! 可划分为 1! 与 2 的乘积，而 1! =1，则求 N! 问题就最终得到解决。分而治之技术是一种行之有效的技术，许多表面上看起来很难的问题，采用分而治之技术就可以很容易得到解决。

当我们拿到一个很大的问题时，很自然会想到把它划分成若干相对独立的子问题，若这些子问题还比较大，自然还会做进一步的分割划分，直到达到容易编程的某种规模。然后对应用分而治之技术得到的若干相对独立的子问题逐一设计其程序块，由这些程序块的全体构成问题求解的答案程序。这就是所谓的模块化结构组织形式。在这种组织形式之下，各程序块就象一个个积木块，而整个程序则象由积木块形成的建筑造型。程序的结构清晰，功能明确，易于排错修改，是结构化程序设计所期望的结果。所以说模块化结构组织形式是结构化程序设计的重要手段之一。

自顶向下逐步求精细化，分而治之，模块化是实现结构化程序设计的基本方法和手段。对一具体问题的设计，往往是三种方法的综合运用，如在逐级划分模块时也要用到自顶向下逐步展开的方法等，同一问题的不同子问题中，也可能使用不同的方法和技术去进行下一层次的设计。所以在程序设计的过程中，要把这几种方法有机地结合起来使用，绝对不能把它们割裂开来。

在设计过程中到了可用一些本原操作表示并用三种基本控制结构控制的时候，就要用某种特定的高级语言语句来编码。有些高级语言如 Ada、C、PASCAL、True BASIC 等提供有与三种基本结构对应的语句，这种编程工作就非常简单易做。如果使用非结构化语言如基本 BASIC、FORTRAN 等语言，则要善于组织语句来实现基本结构，具体实现可参看下一章 § 2.4 到 § 2.7 节内容。运用这几种方法进行结构化编程的许多实例，在后续各章节中我们会陆续给读者介绍。