程序设计技术基础

陈松乔　邰晓英　编著

刘尚威　审

程序設計技术基础

陈松乔　邰晓英　著编

刘尚威　审

中南工业大学出版社

程序设计技术基础

陈松乔 邹晓英 编著

责任编辑：何彩章

•

中南工业大学出版社出版发行

湖南省地质测绘印刷厂印装

湖南省新华书店经销

•

开本：850×1168 1/32 印张：13 字数：338千字

1989年8月第1版 1989年8月第1次印刷

印数：0001—2000

•

ISBN 7-81020-234-0／TP·010

定价：3.35元

内 容 提 要

本书全面系统地介绍设计优质、高效程序的方法和技术，包括程序风格、数据结构，设计、测试、证明、调试、维护技术，效率分析和文档组织等。重点讨论程序的优雅的表示方法和优良的程序特性，结构程序设计方法、逐步求精和模块化技术，测试、调试以及提高程序和设计效率的技术。本书还列举了大量的程序实例和比较大型的实用程序，以利于消化和巩固所学知识，提高程序设计的实际技能。

本书可作为计算机或非计算机专业的本科、专科学生教材或教学参考书，也可供涉及程序设计的工程技术人员阅读和参考。

前 言

计算机应用的日益扩展，要求越来越多的工程技术人员、管理人员，专职或非专职地从事程序设计或软件开发工作。然而，要设计出一个正确、可靠、易读和效率高的程序，仅有计算机语言的知识是不够的，还必须掌握程序设计的方法和技术。这与学习了语文的语法知识后，还应学会如何进行文学创作才能写出优秀作品是一个道理。

在一般的程序设计教科书中，通常只是介绍某一种语言的语法和程序设计中最基本的概念。而本书的目的是让读者在学完诸如PASCAL一类高级语言之后，能够掌握和运用程序开发过程中的全部概念和技术，以提高设计大型程序的适应性和创造性。为此，本书内容涉及软件开发过程中的各个阶段，涉及编写优质、高效程序的方法和技术的各个方面，包括程序设计风格、数据结构、设计方法、测试、证明、调试、维护，效率分析以及文档组织等。

全书共分七章。第一章介绍程序设计技术的发展和问题、软件生命周期；第二章讨论程序设计风格，包括优雅的表示方式和优良的程序特性；第三章介绍各种常用的数据结构，如数组、记录、栈、队、链表、串、树和文件等；第四章则系统讨论结构程序的特性、结构化程序设计方法和逐步求精、模块化技术，以及程序员队伍的最优组织方式；第五章讨论测试，证明方法和调试技术；第六章分析程序效率、包括对人、算法和编码效率的分析，提高效率的措施；最后一章介绍程序文档，包括用户文档、技术文档的组织和书写方法，软件维护技术等。

本书的每一部份均列举了大量符合本书风格、结构和设计技术的实例，结合陈述的内容，还介绍了一些比较大型的实用程序

的开发和设计过程，每章之后有大量的练习。通过实践，可以帮助读者消化和巩固所学知识，提高程序设计的实际技能。

本书主要参考G.M.Schnelder、S.C.Bruell编著的“ADVANCED PROGRAMMING AND PROBLEM SOLVING WITH PASCAL”一书，还参阅了国内有关专著，并结合作者本人多年教学和科研的实践经验。全书力图概念清楚、重点突出、理论联系实际。本书既可作为大、中专学生的计算机教科书，也可供有关工程技术人员参考。

本书在编写过程中得到中南工业大学、宁波大学许多同行的帮助，在此表示衷心感谢！

由于作者水平有限，书中难免存在不足之处，敬希读者批评指正。

作　者

1988年11月

目 录

第一章 绪 论

§1 程序设计技术的发展和问题

程序设计技术的发展可以分为三个阶段。50年代，主要用机器语言或汇编语言编制程序。编制程序的过程相当烦锁，首先必须熟悉机器的指令系统，然后按设计任务把它们分成顺序、分支、循环等几部分分别进行编制。人们把这称之为“按裤子做坐椅”的技艺。当时程序编制的好坏是以程序的灵活性和程序的技巧性来衡量的，要求程序运行时间短，占用主存少。这种程序设计方法，一方面要求程序员要有专门训练，另一方面其编制的程序也不便于推广使用，不同机器之间的移植几乎是不可能的，且由于无一般规律可循、编制和调试一个大程序常常要花很长时间。解决问题的周期长，软件生产效率低。同时由于注意设计技巧，以及灵活性，使得程序难于阅读，难于调试，更难于维护，一段时间以后，就连程序员自己，也难于读懂自己的程序，因此当遇到故障要修改时，整个程序便成了“一碗实心丐”，无法折开。

60年代高级语言得到了迅速发展，用高级语言编写程序大大简化了程序设计，它允许软件工作者和程序员独立于机器。程序编制速度可提高一个数量级，且可以不一定要由计算机专业人员完成，各种专业的一般工程技术人员稍作训练之后，便能编制程序。高级语言还提供了分程序、嵌套、递归等许多重要概念，促进了程序设计技术的发展。

当前已有200多种高级程序设计语言在应用，但广泛使用的却没有超过十多种。这些语言大致分为三类。一类为基础语言，主要用于科学和商业计算，如FORTRAN和COBOL等；第二类

为结构化语言，这种语言可提供许多复杂的数据结构、子程序定义，语句分组（分程序结构）和逻辑构造，如PASCAL、ADA、PL/1和C语言等；第三类为专用语言，这一类语言具有一些特定的性质，以适用于特定的软件应用，如APL、PROLOG、LISP等。

70年代初，出现了大型软件系统的研制，包括操作系统、数据库管理系统、专家系统等。系统软件和许多大型应用软件的研制，常常需要花费许多人力和资金，而研制出来的成品可靠性差、错误多、维护修改困难，因此出现了所谓“软件危机”。

软件危机系指在开发计算机软件时遇到的一系列问题。这些问题不仅指功能不正常的软件，还包括如何开发软件，如何维护数量不断增加的已有软件，以及如何满足对更多软件日益增长的需求等有关的一系列问题。

软件危机表现在：

1．软件生产具有一些不利的特殊性

(1)其产品是程序，其种类和表现方法多种多样，无法“批量”生产。如程序类型不同，所用语言不同，程序结构不同，执行机器不同，设计方法不同等。

(2)软件生产看不见，摸不着，掌握生产过程中的实际情况困难，组织和管理这些生产也困难。

(3)一个大型软件，一般是集体劳动的成果，但在很大程度上依赖于个人能力，特别是脑力（聪明、才智、逻辑性），因此是匠人式的艺术品。

(4)软件产品不同于硬件，没有明显的制造阶段，产品不会用旧和用坏（故无备件）。如有问题，则是由设计而引入，测试而漏网的，因此有繁重的调试和维护过程。

2．程序愈来愈长，复杂性越来越高

一个程序的任务一般不能仅用数学方式描述，往往长达几百、上千上万条语句，要由几百人分若干组共同完成，而各组之间的关

系不是用通常语言所能描述的。生产周期长，不能用盲目加人的办法来加快生产的进度。

3．费用大

其原因在于过去的程序设计没有规范化，也就是说不能利用前人的设计成果，都得重新开始，因此开支极大。一般来说，软件开发费用要占整个计算机系统成本的70～80%。

4．可靠性、适用性、可移植性差

一个大程序，设由N个小程序组成，每个小程序正确性概率为P_i($P_i \leqslant 1$)，那么整个大程序正确性概率为$P < P_i^N$。可见程序越大，可靠性越差，导致维护工作量也越大。许多人认为维护代价要占整个软件费用的50～60%。

产生软件危机的原因，除上面所述原因之外，更重要的是设计者缺乏科学的组织方法和设计方法，程序设计技术急待改进和提高。

软件危机的产生，引起人们的广泛注意。在这种情况下，如何组织软件生产，促进生产更多质量高、成本低的软件，就显得特别重要。人们期望，能象航空工程、建筑工程一样组织软件生产。为软件开发的所有阶段提供更好的工具；为软件实现提供功能更强的“构件”和设计技术；研究软件质量保证的自动技术；研究如何合作、控制和管理的科学方法，即所谓“软件工程”。用工程方法偏写程序，调试、维护、修改程序。

用工程方法开发软件，涉及的范围相当广泛，包括管理学、经济学、心理学、计算机科学等。在开发技术中还涉及到程序设计方法学中的许多问题，包括数据结构、程序风格，结构程序设计方法，程序的测试，证明和调试技术，程序效率和文档等方面的问题。特别是1969年E.W.Dijkstra首先提出了“结构程序设计”的概念之后，人们越来越多地研究程序的结构问题，强调广泛采用这些设计技术，开发出更多的、可靠性可读性好的规范化程序，使程序计设技术进入了一个新的阶段。

§2 软件的生命周期

开发一个大型软件，从提出任务到稳定运行，一般需经过五个阶段，即分析、设计、编码、测试和运行维护。在整个生命周期中，每个阶段都有确定的任务，并产生一定规格的文档交给下一阶段，下一阶段则在前一阶段的基础上继续工作。对各个阶段的基本任务、工作结果和参加人员阐述如下：

1. 需求分析和说明阶段

将用户不完全的、粗略的要求（即软件功能、性能、工作环境、条件、价格、完成时间等），用明确的形式语言进行精确描述，写出系统说明书（或称需求说明书）。它是相关者对软件的共同理解、共同意志的表现，是程序设计负责人和设计者之间的接口，是设计者后续工作的依据。用户和软件人员（主要是系统分析员或高级程序员）应共同参与这个阶段的工作。

为什么要研究“需求分析”呢？这是因为：

(1)软件错误大部分来源于分析阶段，往往由于忽视了用户的某一项要求，系统达不到预期目标，或不能运行，甚至要重新设计。且软件错误象一个“先进后出栈”，即前阶段的错误更难发现、更难更正，所需代价也更大。

(2)设计过程中，有时还要根据需求定义修改硬件（部件、配置等）设计，以满足任务要求。

(3)需求说明是软件设计的基础，也是测试验收的依据。

(4)软件生产过程中，根据需求说明书对软件生产过程进行管理。

可见，分析阶段在软件生命周期中占有十分重要的地位。一般在软件生产中所费时间的比例为：分析∶设计（包括编码）∶测试＝4∶3∶3。为了克服某些程序开发人员急于编码的习惯，

软件开发的组织者往往要求程序人员延时编码。

需求定义的基本内容为：系统所期望的功能和性能，期望的输入和输出，系统的约束条件（如硬件结构、操作系统结构等），以及其它方面的要求，如维护性、可靠性、可扩充性等。最后提供反映系统各数据之间逻辑关系的数据流图，以及与此相关的数据辞典等。

2．设计阶段

设计阶段是在系统说明书的基础上，建立软件系统结构的过程。具体分二步完成：第一步按有机整体集中对软件进行处理，即设计模块结构和数据结构；第二步则考虑每个模块的各个过程方面，并应用设计工具提供该软件元素的详细描述。

设计阶段采用结构程序设计方法、模块化等一系列设计技术，其结果是产生模块说明书，数据结构说明书。模块说明书应包括模块的层次结构图、开发顺序，每个模块的结构、输入、输出，以及应完成的功能、测试数据等。

设计阶段一般由高、中级程序人员完成。

3．编码阶段

编码阶段的任务是选用合适的程序设计语言，产生符合设计要求的高质量程序。

编码风格、编码的清晰性和编码效率是十分重要的，为此应十分注意选用合适的语言，注意程序设计技术和风格，采用先进的程序设计方法和工具。

编码结果是产生经预审合格的源程序清单。由于本阶段工作较为简单，故可由一般程序人员完成。

4．正确性检查阶段

本阶段是确保产品满足需求的过程，应当采用各种手段检查程序的正确性，看看是否达到各类说明书的要求；检查一致性，看看各项之间是否存在矛盾；还应检查程序的完全性，保证不漏掉任何一部分。

正确性检查可采用测试或证明的方法。测试一般是选择若干典型的输入数据，检查是否能得到预期的输出。这种方法比较方便，常被人们采用。但它的基本思想是由一个程序对部分输入数据而言是正确的这一事实，推论出对任何输入数据而言都是正确的这一结论。这只能报告程序有错误，而不能保证程序没有错误。为此，近期来人们又开始采用对程序正确性进行证明的方法，即象证明几何题一样，证明程序是否有错误，以确保程序的正确性。

由于人们大都自觉或不自觉地存在着一种不愿看到自己所写程序出错的心理状态，因此测试或证明最好由一个独立部门（未参加设计和编码的人员）来完成。该部门以能尽可能多地发现错误来衡量自己的工作成就，保证最大限度地发现问题，确保正确的软件交用户运行。

5．运行维护阶段

维护工作主要有两方面：

一是发现和改正错误。程序经测试没有发现错误，但并不等于绝对没有错误，有的错误只能在特殊条件下才能暴露。因此运行过程中（甚至多年之后）还有可能发现错误，并要及时改正它。

二是修改或扩充程序功能。由于用户要求的提高或软硬件环境的变化（如内、外存扩充，操作系统版本更新等），一个老的系统，其原来程序已不再适用，必须对原程序进行修改或扩充，以适应新情况。

由此可见，运行维护是一项十分重要和要求很高的工作，其工作量往往占整个生命周期中各工作量总和的一半以上，必须由有经验、有水平的程序人员担任。

将软件生命周期分成五个阶段，为软件的工程化提供了工作框架。但在实际工作中，不可能直线进行，中间往往存在反复，甚至要从后面阶段返回前面阶段，修改前面的工作和相应的文档，反复进行，直到系统能满足用户要求为止。另外，各阶段之间，任务不同、结果不同，但并无明显界线，在某些情况下也可并行

工作，切不可机械执行。

§3 程序设计方法简介

在现实的生产活动和社会活动中，程序人员面临着大量的、规模有大有小的各种软件开发任务。对于大型系统软件和应用软件，设计人员必须遵循软件工程的方法，开发出正确可靠的软件。而对于大量的中小型任务，由于任务的需求较为简单明了，程序人员往往简化分析说明阶段而直接进入设计、编码等阶段的工作。可见，不论在何种情况下，每一个程序设计人员，均必须采用正确合理的程序设计方法，设计出符合规范、符合要求的高质量程序。

设计一个合格的程序，仅仅掌握几种程序语言是十分不够的。这是因为一个合格程序，不仅要求正确，而且还应当有很好的可靠性、可读性、可维护性，有很好的编码和运行效率。而要做到这些，就必须改变传统的程序设计方法和概念，掌握一套科学的程序设计方法和技术

程序设计方法是一门新的学科。1968年，E·W·Dijkstra 首先提出“GOTO语句是有害的”，向传统的程序设计方法提出了挑战，引起了人们对程序设计方法的普遍重视和广泛讨论。人们提出了许多新的观点、新的方法和新的技术，促进了程序设计方法这一学科的形成和发展。

程序设计方法和技术，包括的内容十分丰富。例如结构程序、模块化设计技术，程序正确性证明、程序变换和程序的形式推导技术等。本书面向工程技术人员，重点介绍在当前程序设计中常用的、已经成熟的一些技术和方法，例如数据结构、编码风格，自顶向下的程序设计方法，模块化设计技术，程序测试方法，证明、调试和维护技术，程序编码和运行效率分析，提高程序效

率的技术措施等。

程序设计方法和技术是一门理论和实践相结合的科学。理解和掌握这些方法及技术，必须进行大量的实践，上机编制和调试大量的程序，并从中发现问题，按所学方法解决问题，以加深体会，巩固和消化所学知识。本书提供的大量实例，正是适于这一目的，希望对读者有所帮助。