〔英〕B.L.米克 P.M.希斯 N.J.拉什比 编
袁崇义 朱培华 译

实用程序设计

Guide to Good Programming Practice

国防工业出版社

实用程序设计

B. L. 米 克
〔英〕 P. M. 希 斯 编
N. J. 拉什比

袁崇义 朱培华 译

国防工业出版社

内 容 简 介

本书是由英国十名有几十年程序设计经验的专家共同撰写的。它介绍了成功进行程序设计实践的各个主要方面和作为一个合格的程序员应该具备的知识和技巧。全书共五章。第一章介绍程序整体设计；第二章介绍程序实际编写中的问题，如语言选择、输入输出和人机接面等；第三章讨论程序开发过程中的防错、排错及改善性能等问题；第四章讨论大数据量、大程序、运行时间长的程序和实时程序等问题；第五章讨论程序员与其他人之间的相互关系。

本书既可作计算程序设计人员的指导性参考指南，也可供一般工程技术人员、大专院校师生阅读。

GUIDE TO GOOD PROGRAMMING PRACTICE
B. L. MEEK
P. M. HEATH
N. J. RUSHBY
JOHN WILEY & SONS
Second edition published in 1983

*

实用程序设计
B. L. 米 克
〔英〕P. M. 希 斯 编
N. J. 拉什比
袁崇义 朱培华 译

*

国防工业出版社出版
新华书店北京发行所发行 各地新华书店经售
国防工业出版社印刷厂印刷

*

850×1168 1/32 印张6 5/8 165千字
1988年5月第一版 1988年5月第一次印刷 印数：3,001—7,000册

ISBN7-118-00181-3/TP17 定价：1.95元

译者的话

我国目前还没有一支职业程序员的队伍，但这支队伍势必会形成，会壮大，会成为四化中不可缺少的力量。如果我们翻译的这本书能为这支队伍的形成有点滴的帮助，那就是我们最大的心愿，也是汪贤驯先生的心愿。汪先生是物理学博士，在加拿大和美国从事计算机工作多年，对程序员会遇到的问题有深刻的认识。汪先生认为这本书是每个称职的程序员必读的，因而向我们推荐。

乍读本书的人也许会感到内容平常，并无惊人之语。然而本书的价值恰恰就在于它总结了这些看似平常实际却十分重要的经验。有些会编几个小程序的人把程序设计看得很简单，编起程序来很随便，认为似乎靠点小聪明就够了。这并不奇怪，因为他们对真正的程序设计毫无经验。其实大部分从事程序设计工作的人（包括译者在内）都曾或多或少地有这种看法。外国人也有类似的经历，不同的是他们吃尽了苦头，现在醒悟了。扉页上的"秋千图"就是他们对自己吃过的苦头的形象描述。这本书乃是直接或间接吃够了苦头的人的肺腑之言，其宝贵正在于此。

我国的计算机应用还没有真正开展起来，因而实践不够，体会不深，苦头吃的也不多。这样就难免有人执迷不悟，把宝贵的经验看成"小题大作"。我们希望读者不要轻视这些经验，以免重走别人的老路。

本书的题目原文是"Guide to Good Programming Practice"，乃是培养优良程序设计习惯之指南的意思。我们将它译为"实用程序设计"，强调"实用"二字；这既不偏离原文太远，又合我国实际，想来是能为读者接受的。

第五章的标题原文是"Other People"，直译应为"其他人"，谈的是程序员和其他人（用户，其他程序员，即服务对象和协作

者）的关系。我们将这一标题按其内容译为“关系学”，一则可以使它更醒目，二则想借此机会为“关系学”正名。现在在汉语中“关系学”一词已蒙受了不白之冤，被一些专搞歪门斜道的人玷辱了。这一章当然不能反映我们这个社会主义国家人与人之间正确关系的全貌，但至少能体现其一般，在是与非之间划上一条小小的界线。

译者本人也没有程序设计的经验，业务水平、英语水平都有限，错译之处在所难免。欢迎读者批评指正。

汪贤驯先生不仅两次购赠本书原版（第一版，第二版），而且在翻译工作上给了我们极大的帮助，在此再一次表示感谢。

本书作者简介

M. 克拉克博士（Dr. Michael Clarke） 伦敦市玛丽皇后学院计算机科学系讲师

P. 希斯（Patricia Heath） 普利茅斯工学院计算中心程序设计部经理

C. 休来特（Carol Hewlett） 伦敦经济学院计算机部咨询服务主管

P. 海德（Paul Hide） 伦敦市伊丽莎白女皇学院高级程序员，主管计算机部咨询服务

I. D. 席尔博士（Dr. I. D. Hill） 医学研究处统计师，1978～1980 年任皇家统计学会高级荣誉秘书

B. 米克（Brian Meek） 伊丽莎白女皇学院计算机部主任

R. 欧弗里尔博士（Dr. Richard Overill） 伦敦市皇帝学院计算机部顾问程序员

N. 拉什比（Nicholas Rushby） 伦敦皇家学院 CEDAR 主任

J. 史蒂尔（John Steel） 伦敦市玛丽皇后学院计算中心应用组主管

M. 威尔生（Martin Wilson） 未来技术系统有限公司软件经理

如何使用本书

人们通常都是从学习一种程序语言开始他们的程序设计生涯，程序语言则往往是通过入门课学习的。初期，人们必然只关心如何掌握语言结构，掌握分解问题并把它重组成适于应用这些结构的基本功。尽管有些较好的入门课本确实重视象结构程序设计技术这样的并不专门针对某类问题或某种语言的原则，但有些入门课本却过分强调语言及如何使用语言表述问题的解法，以至于实际上把语言以外的一切都排除了。往往要经过一段时间，当人们（或由于成为职业程序员，或由于程序设计是其工作的重要辅助部分）已真正开始程序设计的时候，才能逐渐体会到作为一个程序员（无论是职业程序员还是业余程序设计师）并不只是简单地编码。当然，有些人似乎永远没有这种体会。

本书的目的是简明扼要地介绍一下作为一个认真而合格的程序员应该具备的知识。这些知识是已经在程序设计领域活跃多年的一些人的经验总结，无论是不是职业程序员都应该具备。全书重点是实质性的原则，除必要的解释之外不涉及细节，细节可参考文献。当然，并不是本书的每项内容都直接与所有读者或一切程序设计问题密切相关，但当你的工作有变动或开始新课题时，原来不相关的内容也许就相关了。有些地方也许是老生常谈，若是这样那也很好。如果有些内容能帮助你从已有的认识中找出有价值的东西，并将其系统而合理地归纳成几条程序设计基本原理，那就更好了。若有值得你深思的新思想（或许有可能使你豁然开朗！），那是再好也没有了。

《实用程序设计》并不是教你一种特定程序语言的入门课本。它假定你已经知道怎样写简单的程序（也许是从入门课上学习的），如果你已有编写解决“真”问题的小（或不那么小）的程序的经验，而不是只会作练习，那就更好。至于学过和用过哪种

程序语言，那并不重要，也没有太大关系。重要的是你个人有了处理某些实际问题的经验并知道编写“真”程序的难处。

本书分为五章，顺序是按一定的逻辑关系排的，当然这并非唯一可能的逻辑顺序。第一章首先处理整体方案的问题，继而讨论程序的设计原则。当然，后者并非紧跟在前者后面，因为在它前面还要讨论一下如何使用文献及程序包等等。“第二次发明轮子”的事在人类活动中是可能发生的，在程序设计中更是流行的做法；因此这一部分的目的就是要强调指出，查一查你所需要的程序或它的关键部分是否已有现成的可用是值得的。许多程序都是在已有同样好甚至更好的程序可以完成同样任务的情况下编写的——若是作为练习这样做尚可原谅，否则是不应该的。

第二章处理程序的实际编写，包括程序语言（有一节专门讨论语言标准化这一棘手的问题）问题，输入输出的考虑，人机接面问题，当然还包括结构程序设计。第三章从程序开发的角度继续讨论结构程序设计，包括防错、排错及改善性能。第四章把最后一个问题深入一步，讨论在某些类应用中可能出现的特殊问题。最后，第五章讨论一个有时完全被忽略的方面，即程序员与可能涉及到的其他人员的关系及他们对其他人应负的责任，其中包括写文件说明这个重要问题。

本书讨论的思想和技术都用程序或程序框架加以解释，这些程序或程序框架是用 Algol60，Algol68，Fortran Ⅳ和 Pascal，或以它们为基础的“冒牌语言”写成的。这并不意味着这些就是我们选出来的“最好的”语言，而只不过反映出我们试图提供能为大多数读者接受的具体的例子——选用他们最了解的或喜欢用的语言而不管这些语言是什么样的。选用 Fortran 是因为它最接近于一种“世界语”，我们期望许多读者已经接触过 Fortran；其他几种语言由于具有 Fortran 所缺少的特征而被选用，因为它们看来可读性更好。而且，还要强调一下本书并非以一种特定语言为基础而写的。经过一番考虑后，我们排除了用 Cobol 和 Basic 写的例子，因为我们感到这样的例子太冗长，需要很多的解释才

能说清我们想说的要点；不过，即使你是个商业程序员或对语言有偏爱的程序员，这本书也仍然适用。

本书末有供读者参考的要点一览和一个文献目录，这是为你进一步阅读提供的建议。文献目录是按章排列的，以便帮助你把有特殊兴趣的问题或特别有关的问题探索到底。

序　言

在计算机程序设计的基本课程中，教师往往只注意指导学生掌握某个程序语言的各种成分，而忽略做为一个程序员应该掌握的更广泛的知识。类似地，程序设计手册和这方面的教科书也倾向于只介绍语言成分。尽管目前有些较好的书也谈点程序设计的好风格和好方法，但有些书除语言成分以外却别无内容。由于很明显的原因，即便是这些较好的书，除与写程序直接有关的好经验外也从不越雷池半步。读了这些书以后，人们还是要在实践中摸索，在工作中学习，没有机会系统地讨论他们需要的广泛知识。

本书以简单易懂的形式介绍了成功的程序设计实践的各个主要方面。尽管为了说明得更清楚，我们给出了几个用各种常见语言写的例子，但本书内容没有涉及任何特定语言或特定计算机。因此，本书的目的是补充而不是代替那些入门手册、书籍和课程。希望本书能为那些羽毛未丰的程序员提供一个指导和参考的源泉，在他们初出茅庐时就能对他们有所裨益。由于篇幅的限制，本书不能把各种技术全都包罗进去。例如，只是就一般原则讨论了一下结构程序设计，而没有介绍更适合于程序设计课程的详细方法；判定表的使用也留给专门的教科书去介绍了；虽然有点不情愿，但还是略去了程序正确性证明的讨论，因为这还是一个处于研究和实验之中的领域，并非成熟的技术。在附录中我们提供了进一步阅读的文献，深望读者能借助这些文献跟上技术发展的形势。

本书的作者都有几十年程序设计经验，了解各种程序设计环境。虽然他们大部分人在教育部门工作，但也有几位一直从事工业和商业活动。作者们不仅分头写各章节，而且还互相阅读手稿，互相做评论，提建议。编者还要感谢作者们高兴地接受了编

辑上的变动（根据作者相互的评论，有些变动是实质性的，有些变动是为了保持内容的连续性）。此外，伊丽莎白女皇学院物理系的白路济（H. Balyuzi）先生和拉壳维克（S. Lackovic）先生对4.5节提出了许多改进建议。该学院计算机室的陈（TAN）小姐做了大量打字、复印以及其他文书工作，编者在此谨表谢意。最后，我们还要感谢吉本斯（P. Gibbins）先生和伦敦大学《Computer Centre Newsletter》的编辑，他们同意我们复制其"秋千"漫画，这组漫画比我们知道的任何别的东西更清楚地抓住了程序设计这个行当的真象。

B. L. 米克
P. M. 希斯
1979年3月

《实用程序设计》的这一修订版增加了N. 拉什比(N. Rushby）作为第三位编者。事实上大部分修订工作是他完成的。

这只是个修订版，并不是完全重写的。除若干更正和改进外，本版还加进了一些新内容，其中最引人注目的就是有关人机接面的整整一节。此外还加了个"要点一览"，我们希望它对读者有所帮助。参考文献也作了更新（尽管它跟前一版一样，仍然只能告诉读者到哪里去寻找进一步的信息，而不是完整的文献目录）。

我们十分感谢私下提出的和公开评论的有关第一版的建设性批评和建议。本修订版考虑了这些批评和建议，例如，对某些例子的叙述做了改进。有些建议要我们增加某些我们认为应属于程序设计入门书和训练课的材料，尽管这种书和课程并不总是包含这些内容，我们还是没有采纳这些建议，因为本书是作为这些书和课程的补充而纂写的。建议增加的材料包括详细讨论使用某些特定语言可能遇到的，而不是我们作为一般性说明已简单叙述过的问题，还包括初学者必须克服的有关作业控制和命令语言（command languages）的难题等等。我们还决定不增加软件度量（software metrics）及估价技术等内容，而且仍旧略去正确

性证明；从程序设计行业的现状看，至少对本修订版来说，我们认为这些主题仍旧必须由更高级更专门的著作来论述。

最后，我们采取了相当大胆的一步，要求出版者更换了封面上的漫画，我们希望本修订版能证明它是“用户所需要的”，至少更接近这样。

P. M. 希斯
B. L. 米克
N. J. 拉什比
1982年5月

目　录

第一章　策略和设计

写程序就如同做衣服和建造房屋一样，在你真正着手工作之前，你必须清楚你究竟想要做什么，并且要有个设计方案。第一章考虑的就是最初的设计，包括完全避免写新程序的途径。

1.1　吃透问题

经验告诉我们，接受入门训练的人所写的程序往往不能象预期的那样工作，究其原因可以归结到这一事实，即当他写程序时，并没有充分理解他所要解决的问题的要求究竟是什么。因此，我们提出的第一条建议就是，在你开始写程序之前，要确实把问题弄清。看起来这似乎是老生常谈。

尽管这可能是老生常谈，还是值得一提。关于程序设计最常见的误解之一就是以为，程序设计要做的第一件事就是在终端机前面坐下来，或者在程序纸上写下自己的名字，然后就开始写程序。殊不知事实上在人们能达到这一步之前该做的工作往往关系到课题的成败。

参加启蒙课的初学者常常受到鼓励，要他们按这种方式一个劲儿往下钻，这样做部分地是要他们树立自信心，部分地是提供推动力。必须强调指出的是，这种课程一开始所设置的练习必然是经过简化的，其中要解决的问题都是仔细挑选的，目的是使得除设计练习时所要测验的难点外，其它难点得以避免或根本就不存在。不幸的是，这种练习有时会使人以为所有的程序设计都不过如此。这实在是莫大的误会，在任何实际课题中，几乎不可能出现类似做练习的情况。

刚学入门课的学生应该把自己想象成一个婴儿，尽管现在吃的是细心选择的容易消化的食品，但应该明白过一段时间后他就得用较粗糙（但也较有味）的食品磨利自己的牙齿，这种食品是

需要做点咀嚼的。不仅如此，他还得做点准备工作，然后才能坐下来就餐。

这种经过简化的练习的一个特点是，它们不仅简单易做，而且陈述精确。但今后需解决的实际问题几乎不可能这么简单，也不可能提得那么精确。如果问题由你自己设计（例如你是一位认真的业余程序员，由于另一工作的需要得写一个程序），那么重要的是在设计程序之前先把你所要达到的目的尽可能清晰地系统地提出来。一个办法就是把问题的规范说明写在纸上，其中包括初始数据，要获得什么结果，可用的手段以及必须满足的条件（如时间和资源），等等。这样做的本身往往会帮助你理清思路，但最好还是把所写的规范说明交给一位具有必要的背景知识的同事看看，看他是否能理解，或许他能看出什么错误或模棱两可的地方以及遗漏之处。

如果课题不由你设计（例如你是个受雇于人的程序员），那么问题就可能以任何方式提交给你。也许是一般的口头指示；也许是完整的书面规范说明；也许是介乎两者之间的某种方式。如果是规范说明，那么这时关键的是要透彻地研究它，直到你有把握已经完全理解为止。如果其中有你不能理解的东西，就要求给你进一步阐明。不要因怕丢面子或怕人笑话而不敢发问，以为随后一切都能弄清楚。否则你就要冒风险，其中最严重的后果莫过于按错误的理解写出文不对题的程序。类似地，若规范说明不很精确、不完整或根本没有书面说明，那么我们强烈地建议你自己动手写出精确的规范说明并力求得到认可，这正是要求你做的。如果课题涉及多人，而不仅仅是一个程序员和一个需要程序的人，那么每个有关的人对于正在从事的工作都必须有同样的理解。无论如何，你仍然值得花点功夫，以便对规范说明是否完整是否清楚得出你自己独立的见解。本书扉页上的“秋千图”，一定打动了许多有经验的程序员敏感的心弦，因为他们是吃尽苦头才懂得这一教训的。

第二点建议不那么显而易见。我们认为，应该对整个课题

(而不只是对程序设计的问题)从一开始就要有全面的看法。前面曾提到了本书内容的逻辑顺序，这一顺序的确在某种程度上反映了从设计（decign)、编写（writing)、开发（development)直到产生完整的有详细文件说明的最终产品这一发展过程。可是，这一顺序并不是程序员在考虑各种因素时必须遵循的次序。文件说明（documentation）的工作必须（与课题规范说明一起）从一开始就要做而且要贯彻始终，这就是一个特例。我们把这项工作排在最后，是因为只有在其他工作都完成之后才可以最后完成文件说明。事实上各阶段的工作都是息息相关的。设计和编写程序必须考虑到随后必然要做的调试（debugging）工作。第四章讨论的那些专题，例如程序是否需要长期或连续运行，都将影响最初的算法设计。对问题的要求所做的分析，很可能会影响到选取什么程序语言或使用什么输入输出设备等。反过来也是这样，有哪些语言或输入输出设备可供使用也必将影响算法设计。诸如这样的要点在后面的叙述中都会变得很明显。

这就是对课题全面考虑为什么如此重要的原因，要试着估计众多因素中最可能起主导作用的一个，并在用于判断各种可能的选择方案优劣的标准中确定相对优先顺序。而且要在课题的整个进展过程中始终把这些记在心中。当然这些原始判断有可能最后是错的，也可能因条件的改变而失效，但事情绝不会总是这样，没有理由不作这些判断。

IBM公司的口号“思考”被计算机界的人们当做笑话，但讥笑改变不了事实，“思考”的确是个很好的建议。在这一节里我们一直在反复说的就是，无论期限多么紧迫，无论压力多么大，花点时间把整个工作从头到尾做一次全面的研究以获得正确的理解，这决不是浪费。我们还建议你和别人一起讨论一下你的课题，因为这有助于你对问题的理解。在课题进展中寻求特别的帮助那是以后的事，不过这里值得做几点说明。在初始阶段，重要的是讨论本身，至于和谁讨论在某种意义上那是第二位的。当然我们指的是以分析问题为目的真正的讨论，而不是做给别人看看的走

过场。你可以和同事讨论，也可以和你的当事人或上级讨论，这些人都有可能提出有益的建议或提供有用的信息。从使你思路清晰这一目的来说，任何乐于听你讲的人都行。

总之，无论你干什么，在动手之前都应尽可能把课题透彻地想一遍。

1.2 使用文献

既然在任何课题中首要的工作是思考；是细心地考虑究竟要干什么；是保证要解决的问题得到充分理解，那么任何有助于理解的工作都是重要的。我们已经指出了与其他人讨论的重要性，例如，和未来的程序用户讨论，和课题领导或课题组其他成员讨论。进一步来说，对问题的充分理解并不一定意味着可以毫不费力地找到问题的解法，也不一定马上就能弄清楚从计算的角度来看究竟需要什么条件，例如需要哪种类型和大小的资源。

因此，往往首先使用文献查阅课题领域的背景资料，以便获得对问题更全面的理解，并可得到解题的启发。通过讨论和阅读，程序员对问题的理解不断深化，问题的各种解法就会出现在脑海中。有些解法可能是直接从别人那里得到的；有些则可能是自己的想法。各种解法的优劣也会越来越清楚。例如，有些解法可能更适合于某个特定问题或更适用于某些计算设施。不过在过于倾心于某一特定方法之前，重要的是收集材料并考虑还有哪些可供选择的方法。

在确定解题方案的这一阶段还有一件事值得一提。有时在课题任务交给你的时候就已经带有现成的解法，例如课题领导人或未来的用户都可能提供解法。课题领导提供的解法很可能是正确的。然而你若想干得象个行家那样漂亮，那么就有责任使自己确信这是个正确的解法或是正确解法之一。优秀的课题领导自然不仅会告诉你用什么方法求解，而且也会解释选择这个解法的原因。如果是用户提供的解法，情况就比较复杂了，不过你仍然有责任弄明白这个解法是否正确。例如，用户也许要求对他的数据做些特