

计算机实用软件技术系列丛书
软件开发人员必备工具书



代码大全

STEVE MCCONNELL 著

天奥 译

熊可宜 校

学苑出版社

1993

B 测试版

缺陷报告: codecomplete@163.net

本书网站: www.delphidevelopers.com

(京)新登字 151 号

内容摘要

本书从软件质量和编程思想等方面论述了软件构造问题,并详细论述了紧跟潮流的新技术、高屋建瓴的观点、通用的概念,并含有丰富而典型的程序示例。本书中所论述的技术不仅填补了初级与高级编程技术之间的空白,而且也为程序员们提供了一个有关编程技巧信息来源。

本书适合经验丰富、自学成才的程序员阅读,也适合于那些几乎不懂什么编程技巧的新程序员阅读。

欲购本书的用户,请直接与北大 8721 信箱联系。邮编:100080,电话:2562329。

版权声明

本书中文版权由 Microsoft 授予北京希望电脑公司和学苑出版社独家出版、发行。未经出版者书面许可,本书的任何部分不得以任何形式或任何手段复制或传播。Microsoft, MS, Ms Dos 和 Windows 是微软公司注册商标。

计算实用软件技术系列丛书
软件开发人员必备工具书

代码大全

著者: STEVE McCONNELL

翻译: 天奥

审校: 熊可宜

责任编辑: 徐建军

出版发行: 学苑出版社 邮政编码: 100032

社址: 北京市西城区成方街 33 号

印刷: 兰空印刷厂

开本: 787X1092 1 / 16

印张: 33,625 字数: 777 千字

印数: 1~10000 册

版次: 1993 年 11 月第 1 版第 1 次

ISBN7—5077—0876—4 / TP • 25

本册定价: 66.00

学苑版图书 印、装错误可随时退换

前 言

近年来，关于软件开发的研究，进展是非常迅速的，但是开发实践却并非如此。许多程序仍然是错误百出，充斥着过时的技术，从而无法满足用户需要。软件工业界和学术界的 researchers，基本上已经解决了七十年代和八十年代在编程中遇到的问题，并发展了相应的技术。但是直到现在，这些技术中的大部分仍然没有在软件编程中广泛采用，其主要原因是这些研究成果主要发表在高度专业性的学术刊物中，普通的程序员们无心顾及。Sridhar Raghavan 和 Donald Chand (1989) 的研究表明，一项新技术从诞生到被工业界广泛采用大约需要 5 到 15 年的时间。本书的目的就是希望能够缩短新技术推广周期，使广大的程序员们可以迅速地获得软件开发的最新方法与手段。

本书所面向的对象

本书中所收集的研究和编程经验，将有助于你编写出高质量的软件，并且使得开发周期缩短。通过阅读本书，你将会对自己过去所犯过的错误有更深刻的理解，并懂得今后如何避免它们。同时，书中所收集的丰富的编程经验也将使你在控制大规模项目和按要求对软件进行修改和维护时感到得心应手。下面是适合阅读本书的几类人：

经验丰富的程序员

本书适合于想要得到一本全面易用的软件设计指南的那些资深程序员们阅读。由于本书的中心内容是广大程序员们所熟知的实现过程，因此，无论是受过正规训练而已经经验丰富的程序员，还是完全靠自学成长起来的程序员，都能容易读懂本书所论述的先进技术和方法。

自学成才的程序员

本书尤其适合于很少受过正式专业训练的程序员阅读。1988 年有 100,000 人加入了程序员大军，但其中只有 40,000 人是从计算机专业毕业的本科生，其余则几乎全是靠自学成才的。同时，还有不计其数的其他各行各业的人员需要在工作中自己动手编一些程序。无论你所受到的正规计算机专业训练多或少，本书都将使你对有效的编程方法和技巧有更深刻的理解。

学生

本书不仅适于实践经验丰富但理论基础薄弱的自学者阅读，同时也适于那些理论基础较好但几乎不懂得什么编程诀窍的毕业生们阅读。新程序员们的某些实践经验来自于经验丰富的同事，但主要还是靠自己——吃一堑，长一智——获得的，这往往是一个艰苦而缓慢的过程。通过本书，可以使你在短时期内获得大量的经验和技巧，从而脱颖而出，所以，不妨一试。

本书的主要特点

完备的软件创建参考

本书从质量和编程思想等方面论述了软件构造问题。几乎囊括了生成子程序、数据的输入输出与控制结构、调试、代码调整策略与技术等各方面的细节。在使用本书时不必逐页阅读每一个细节，只要需要在需要时查阅你所感兴趣的章节即可。请把本书作

为手册而不是作为教科书来使用。

方便住实用的检查表 书中附有用于检查软件的结构设计、设计方法、模块和子程序等质量的检查表，以供评估软件质量之用。同时，关于变量名、控制结构、方案布置、测试用例等等检查表也将使你获益匪浅。

紧跟潮流的新技术 书中论述了许多目前最先进的技术，其中许多还只是刚刚投入应用。由于本书取材于实践经验和最新研究成果两个方面，因此书中所提供的技术在相当长的时间内都不会过时。

高屋建瓴的观点 阅读本书将使你跳出日常琐碎工作的圈子，对软件开发有一个总体上的把握与认识。繁杂的日常工作往往使程序员们穷于应付而无暇阅读浩如烟海的书籍与资料，本书丰富而翔实的第一手资料将弥补这一缺憾，使你对软件开发的策略作出正确决策而不致陷入旷日持久的消耗战中。

通用的概念 无论你用的是 Pascal、C、C++、Ada、Basic、Fortran 还是 COBOL，都可以从本书所论述的概念、方法和技巧中获得教益。

丰富而典型性的程序示例 书中含有大约 500 多个正反两方面的程序示例。之所以引入这么多的示例，是因为笔者就是从各种例程中吸取了大部分的知识、经验与诀窍，因此笔者认为最好的学习方法是多看例程。例程是用多种语言写成的，因为对于程序员来说，掌握多种语言是其必不可少的基本素质之一。而且，只有掌握了不受语法规则限制的编程准则，才能真正有效地提高你的编程效率和质量。为了减轻由于使用多种语言所带来的额外负担，在例程中除非确有必要，尽量避开了各个语言过于独特的部分。事实上，如果你真正注意每个例程所要说明的问题的话，那么不必详细理解每个程序段，你也可以清楚地懂得程序的意义。同时，为了进一步减轻读者的负担，对程序中有意义的部分作了标记。

本书的独特内容

本书关于创建活动的内容是从多个渠道获得的。有关创建活动的资料不仅分布得非常分散，而且往往没有成文资料，事实上，卓有成效的优秀程序员们所使用的技术并不神秘，但由于日常事务的繁重和工作任务的重压，程序员们很少有互相交流切磋的时间，因而，他们往往缺乏有关编程技巧的有效信息来源。

本书中所论述的技术不仅填补了初级与高级编程课本之间的空白，而且也为程序员们提供了一个有关编程技巧的信息来源。比如当你读过 C 语言初级教程之后，你可以再读 C 语言高级教程，然后再去读 C 语言高级的高级教程，但读完这些书后，你还能再读什么书呢？你可以再去读关于 PC、Macintosh 或 UNIX 等硬件或操作系统的书或者其它有关编程细节的书——因为你如果不了解实现环境详情的话是无法充分有效地使用语言和程序的。但这只是讨论了编程的一个方面，最有效的编程技术是那些不受实现环境及语言限制的技术。其它书往往忽略了这一点，但这恰恰是本书的重点。

写作本书的目的

需要一本关于软件开发有效技术的书，是软件工程界所公认的。由计算机科学技术委员会所发表的一份报告认为，提高程序质量和生产效率的最有效途径是出版一本关于软件开发有效

技术的书，而且这本书应该以手册的形式来组织。

同时，计算机编程技术的发展史也证明急需一本这方面的书，本书正是出于这个目的才出版的。

创建活动未受到应有的重视

在一段时期内，软件开发与编码被当作是一回事，但随着软件开发周期中的其它活动被认识，这一领域内的主要努力全部集中到了项目管理、需求分析、设计和测试等方面，创建活动成了被遗忘的角落。

与这种现象相对应的思想是认为创建活动是软件开发中无关紧要的部分。于是，刚入门的程序员被派去进行创建工作，为那些已经由上一阶段设计好的子程序编码。工作几年之后，他可能会被提升至需求分析或项目管理部门。于是，这位程序员也会自豪地感到他不必再去编码了。

创建活动是非常重要的

创建活动被忽视的另一个原因是：研究者和程序员们错误地认为与其它开发活动相比，创建活动是一相对来说比较机械的活动，没有什么值得改进的。没有什么比这种想法离事实更远了。

在小规模项目中，创建活动约占工作量的 80%，在中型项目中也要占 50% 的工作量，而发生在创建活动中的错误则占总错误的 50% 到 75%。一项会产生 50% 到 75% 错误的工作是有许多待改进之处的。

一些人认为，虽然创建时的错误占到总错误的 50% 到 75%，但修改它们的费用与分析、设计错误相比要少得多。的确，创建时的错误修改费用与前期工作错误修改费用相比是要少一些，但是绝对数并不少。Gerald Weinbers 曾在 1983 年报道过三个错误，每个错误的修改费用都高达数百万美元，而每个错误都县一行编码层次上的代码错误。因此，绝不能以修改费用相对少为理由来忽视创建活动。

具有讽刺意味的是，被忽视的创建活动事实上是唯一任何规模项目都必不可少的活动。需求可以进行猜想而不必分析；结构可以被省略而不必设计。系统测试也可以不进行。但是，如果你想有一个程序的话，你就不得不进行创建活动。

本书的独特性

如果创建活动的重要性是非常明显的话，那么本书恐怕就没有出版的必要了。但事实上几乎没有什么书详细论述了这一主题。只有 15 年前出版过一本类似内容的书，讲述的是 ALGOL、PL / I、Ratfor 等早已过时的语言中的具体问题。其它偶尔也有几本这方面的书，但却是教授们针对教学用的演示性项目而写的，没有涉及到真正的工程问题。有些则偏激地推崇新技术而不恰当地贬低了一些非常实用的成熟技术。总之，就内容的新颖、翔实、丰富和实用来看，目前似乎还没有与本书相匹敌的关于创建活动的书。

编 者

目 录

第一章 欢迎进入软件创建世界	1
1.1 什么是软件创建	1
1.2 软件创建的重要性	3
1.3 小结	4
第二章 利用隐喻对编程进行更深刻的理解	5
2.1 隐喻的重要性	5
2.2 如何使用软件隐喻	6
2.3 通常的软件隐喻	7
2.4 小结	11
第三章 软件创建的先决条件	12
3.1 先决条件重要性	12
3.2 问题定义先决条件	16
3.3 需求分析先决条件	16
3.4 结构设计先决条件	20
3.5 选择编程语言先决条件	26
3.6 编程约定	29
3.7 应花在先决条件上的时间	29
3.8 改变先决条件以适应你的项目	30
3.9 小结	30
第四章 建立子程序步骤	31
4.1 建立程序步骤概述	31
4.2 程序设计语言（PDL）	31
4.3 设计子程序	33
4.4 子程序编码	37
4.5 检查子程序	42
4.6 小结	44
第五章 高质量子程序特点	45
5.1 生成子程序的原因	47
5.2 子程序名称恰当	51
5.3 强内聚性	52
5.4 松散耦合性	56
5.5 子程序长度	60
5.6 防错性编程	61
5.7 子程序参数	67

5.8	使用函数	71
5.9	宏子程序	72
5.10	小结	74
第六章	模块化设计	75
6.1	模块化：内聚性与耦合性	75
6.2	信息隐蔽	77
6.3	建立模块的理由	84
6.4	任何语言中实现模块	85
6.5	小结	90
第七章	高级结构设计	92
7.1	软件设计引论	92
7.2	结构化设计	95
7.3	面向对象	98
7.4	对目前流行设计方法的评论	102
7.5	往返设计	105
7.6	小结	109
第八章	生成数据	111
8.1	数据识别	111
8.2	自建数据类型的原因	113
8.3	自建类型的准则	115
8.4	使变量说明更容易	115
8.5	初始化数据的准则	120
8.6	小结	120
第九章	数据名称	121
9.1	选择名称	121
9.2	特定数据类型的命名	124
9.3	命名约定	128
9.4	非正式命名约定	129
9.5	匈牙利命名约定	132
9.6	短名称	136
9.7	要避免的名称	137
9.8	小结	139
第十章	变量	141
10.1	作用域	141
10.2	持久性	143
10.3	赋值时间	144
10.4	数据结构与控制结构的关系	145
10.5	变量功能单一性	146
10.6	全局变量	148

10.7	小结·····	153
第十一章	基本数据类型·····	154
11.1	常数·····	154
11.2	整型数·····	155
11.3	浮点数·····	157
11.4	字符和字符串·····	159
11.5	逻辑变量·····	161
11.6	枚举类型·····	162
11.7	命名常量·····	164
11.8	数组·····	166
11.9	指针·····	167
11.10	小结·····	175
第十二章	复杂数据类型·····	176
12.1	记录与结构·····	176
12.2	表驱动方法·····	179
12.3	抽象数据类型 (ADTs) ·····	192
12.4	小结·····	198
第十三章	顺序程序语句·····	199
13.1	必须有明确顺序的程序语句·····	199
13.2	与顺序无关的程序语句·····	201
13.3	小结·····	207
第十四章	条件语句·····	208
14.1	if 语句·····	208
14.2	case 语句·····	213
14.3	小结·····	216
第十五章	循环语句·····	217
15.1	选择循环类型——·····	217
15.2	控制循环 (Controlling The Loop) ·····	222
15.3	编写循环的简单方法——从里到外·····	230
15.4	循环与数组的关系——·····	232
15.5	小结·····	233
第十六章	少见的控制结构·····	234
16.1	goto 语句·····	234
16.2	return 语句·····	243
16.3	递归调用·····	244
16.4	小结·····	248
第十七章	常见的控制问题·····	249
17.1	布尔表达式·····	249
17.2	复合语句 (块) ·····	257

17.3	空语句	257
17.4	防止危险的深层嵌套	258
17.5	结构化编程的作用	264
17.6	用 goto 模拟结构化结构	267
17.7	控制结构和复杂性	269
17.8	小结	271
第十八章	布局 and 风格	272
18.1	基本原则	272
18.2	布局技巧	279
18.3	布局风格	280
18.4	控制结构布局	285
18.5	单条语句布局	292
18.6	注释布局	301
18.7	子程序布局	303
18.8	文件、模块和程序布局	306
18.9	小结	311
第十九章	文档	313
19.1	外部文档	313
19.2	编程风格作文档	314
19.3	注释还是不注释	316
19.4	有效注释的关键	318
19.5	注释方法	322
19.6	小结	337
第二十章	编程工具	338
20.1	设计工具	338
20.2	源代码工具	339
20.3	执行代码工具	343
20.4	面向工具的环境	345
20.5	建立自己的编程工具	346
20.6	理想编程环境	347
20.7	小结	350
第二十一章	项目大小如何影响创建	351
21.1	项目大小	351
21.2	项目大小时开发活动的影响	352
21.3	项目大小对错误的影响	356
21.4	项目大小对生产效率的影响	357
21.5	小结	358
第二十二章	创建管理	359
22.1	使用好的代码	359

22.2	配置管理	361
22.3	评估创建计划	364
22.4	度量	369
22.5	将程序员视为普通人	370
22.6	如何对待上司	374
22.7	小结	374
第二十三章	软件质量概述	375
23.1	软件质量特点	375
23.2	提高软件质量的方法	377
23.3	各种方法的效果	379
23.4	何时应作质量保证	381
23.5	软件质量的一般原则	381
23.6	小结	382
第二十四章	评审	384
24.1	评审在软件质量保证中的地位	384
24.2	检查	386
24.3	其它评审方法	389
24.4	小结	391
第二十五章	单元测试	393
25.1	单元测试在软件质量中的作用	393
25.2	单元测试的一般方法	395
25.3	测试技巧	396
25.4	典型错误	404
25.5	测试支持工具	408
25.6	提高测试质量	411
25.7	测试记录	412
25.8	小结	412
第二十六章	调试	414
26.1	概述	414
26.2	找错	417
26.3	修改错误	423
26.4	调协心理因素	425
26.5	调试工具	427
26.6	小结	430
第二十七章	系统集成	431
27.1	集成方法重要性	431
27.2	分段与递增集成	432
27.3	递增集成法	434
27.4	改进的公布法	439
27.5	小结	445

第二十八章 代码调整策略	446
28.1 功能综述	446
28.2 代码调整介绍	448
28.3 低效率情况	454
28.4 代码调整方法	457
28.5 小结	457
第二十九章 代码调试技术	459
29.1 循环	459
29.2 逻辑	460
29.3 数据转换	469
29.4 表达式	474
29.5 子程序	48
29.6 汇编语言再编码	484
29.7 调试技术快速参考	485
29.8 小结	486
第三十章 软件优化	487
30.1 软件优化种类	487
30.2 软件优化指南	488
30.3 编写新程序	489
30.4 小结	499
第三十一章 个人性格	501
31.1 个人性格是否和本书的主题无关	501
31.2 聪明和谦虚	502
31.3 好奇	503
31.4 诚实	504
31.5 交流和合作	506
31.6 创造力和合作	507
31.7 懒惰	507
31.8 不是你想象中那样起作用的性格	508
31.9 习惯	508
31.10 小结	510
第三十二章 软件开发方法的有关问题	511
32.1 克服复杂价	511
32.2 精选开发过程	513
32.3 首先为人编写程序，其次才是计算机—	514
32.4 注重约定使用	515
32.5 根据问题范围编程	516
32.6 当心飞来之锅	517

32.7	重复	519
32.8	不要固执己见	520
32.9	小结	521
第三十三章	从何处获取更多的信息	522
33.1	软件领域的资料库	522
33.2	软件创建信息	523
33.3	创建之外的主题	523
33.4	期刊	524
33.5	参加专业组织	525