全国计算机应用软件人员水平考试辅导丛书

程序设计考试指导

（程序员级 高级程序员级）

郑人杰 朱慧真 主编

清华大学出版社

全国计算机应用软件人员水平考试辅导丛书

程序设计考试指导

（程序员级　高级程序员级）

郑人杰　朱慧真　主编

清华大学出版社

内 容 简 介

本书为计算机应用软件人员水平考试（程序员级、高级程序员级）的复习辅导教材，它是在中国软件行业协会考试指导中心组织下，由北京大学、清华大学等院校的有关专家在原编写出版的《软件人员水平考试辅导》（杨德元主编）基础上，依照1990年公布的《中国计算机应用软件人员水平考试大纲》编写而成的。内容包括：程序编制、语言基础（FORTRAN，COBOL，PASCAL，C，CASL）及语言处理程序基础。各章均有一定数量的例题、习题，以指导软件人员参加水平考试。

本书可供计算机应用软件人员学习、参考，也可供大专师生和工程技术人员学习参考。可供各部门举办辅导班用作教材（配有录像带）。本书也是中央电视台电视辅导讲座教材。

全国计算机应用软件人员水平考试辅导丛书

程 序 设 计 考 试 指 导

（程序员级 高级程序员级）

郑人杰 朱慧真 主编

☆

清华大学出版社出版

北京 清华园

北京市联华印刷厂印装

新华书店总店科技发行所发行

☆

开本：787×1092 1/16 印张：16.5 字数：391 千字

1991 年 3 月第 1 版 1991 年 6 月第 2 次印刷

印数：20001—55000

ISBN 7-302-00835-3/TP·303

定价：7.10 元

《程序设计考试指导》编委会

前　言

计算机软件人员水平考试是造就宏大的多层次计算机人才队伍的一项重要措施，从1990年起，在全国实施《中国计算机应用软件人员专业技术职务任职资格（水平）考试暂行规定》，资格考试级别分为程序员级（相当于技术员、助理工程师）、高级程序员级（相当于工程师）、系统分析员级（相当于高级工程师）。

为了指导应用软件人员的学习，帮助准备参加软件人员水平考试的学员应试，我们在原编写出版的《全国计算机应用软件人员水平考试辅导丛书》（杨德元主编，1987—1989）基础上，重新聘请了北京大学、清华大学、中国科大研究生院、上海交通大学、南京大学、南京东南大学等单位的专家组成编委会，根据1990年2月公布的考试大纲，进一步改编成本丛书。

本丛书共包括：《程序员水平考试指导》、《高级程序员水平考试指导》、《程序设计考试指导（程序员级、高级程序员级）》、《软件人员水平考试题例和选解》及《系统分析员教程》。

必须说明，本丛书是供水平考试应考人员参考的综合性辅导教材，供读者在自学基础知识后复习提高用，它的主要特点是全面（覆盖了考试大纲的主要内容）、系统（而不是给出零碎的知识），并且体现了基本要求（包括基本概念、基本方法和基本原理）。限于篇幅，各章节不可能全面展开。为解决读者自学问题，我们将另行组织编写《全国计算机应用软件人员水平考试系列教材》丛书。

本丛书亦可用作在职应用软件人员进修教材，或计算机应用专业师生的参考书籍，本丛书也是中央电视台播放的软件人员水平考试电视辅导讲座的教材。

本书供程序员与高级程序员级用，共包括有关程序编制、语言基础及语言处理程序基础三部分。本书由郑人杰、朱慧真主编，参加编写的有：清华大学严俭和（第三章）；北京大学陈向群（第一章）、韩奕（第四章）、丁文魁（第八章）、朱慧真（第六、七章）；中国地质大学苏民生（第二章）；北京信息工程学院孟庆昌（第五章）等，徐国平参加了编审。

在本丛书的编写过程中，曾得到中国计算机学会的指导和帮助。清华大学出版社为使本丛书迅速出版付出了辛勤劳动，在此一并致谢！

中国软件行业协会考试指导中心

1990年11月

目　录

第一章　程序设计与流程图

1.1 程 序 设 计

1.1.1 程序设计及其步骤

程序设计，简单地说，就是根据所提出的任务，用某种程序设计语言编制一个能正确完成该任务的计算机程序。

一般程序设计过程包括以下几个步骤：

(1) 问题定义——把问题、所涉及的输入数据以及希望得出的结果，用日常语言尽可能清晰、完整地表达出来；

(2) 算法设计——确定怎样解决问题，并把任务分解成计算机能够执行的几个特定操作，即：确定解决问题的合适的算法；

(3) 流程图设计——用形象的、适于编写程序的方法表达算法；

(4) 程序编制——用选定的语言，按流程图提供的步骤写出程序；

(5) 程序调试——即排错，查出在程序执行过程中出现的错误及其位置，并予以纠正；

(6) 程序测试——确认程序在各种可能情况下均可正确执行，且输出结果准确无误；

(7) 文档资料编制——编写程序的使用和维护说明书；

(8) 维护和再设计。

以上程序设计过程的各个步骤一般是顺序执行的。开发一个成功的程序需要反复思考和进行这些步骤，并对初始步骤的工作给予充分重视。

1.1.2 问题分析

程序员要解决某一问题时，首先要分析这个问题，即要确切知道所要解决的问题究竟是什么问题。

一般，问题分析的四个步骤如下：

(1) 精确地确定问题；

(2) 识别问题的输入和变量；

(3) 识别问题要求的输出；

(4) 建立问题的数学模型。

1. 问题说明

在解题过程中，仔细准确地说明问题，明确问题的范围是非常必要的。对于简单的问题，有时仅用一句话说明就够了；而对于较复杂的问题，问题说明要用更长一段文字来叙述。另外，问题说明常常包括很多小问题，也必须准确加以说明。

在进行问题分析时，切忌建立错误的问题说明，不完备的问题说明，以及过分复杂的问

题说明。

2. 问题的输入和变量

在分析问题时，对需要解决的问题识别输入和变量是很重要的。

问题的输入由数据组成，并通过算法对它们进行处理，从而得到问题的解答。例如，在工资管理问题中，输入可由职工的下列数据组成：工作证号码、姓名、基本月工资、各项杂款、扣款等等。

问题的变量是指：问题的输入及输出、一些必要的常数及中间结果等。

3. 问题的输出

识别问题的输出（或结果）与问题说明密切相关。在这一阶段，要求确定认为合适的输出总量。要防止两种情况：一种情况是产生超过需要的更多的输出，这不仅意味着计算机资源的浪费，同时也容易使程序员混淆主次；另一种情况是产生过少的输出，这将导致算法不能产生问题的正确答案。

4. 数学模型的建立

建立数学模型是问题分析的出发点和结局。为了得到给定问题的最优解答，一种有意义的并且很方便的方法是用数学形式将问题写出来。这种数学描述或表达称为问题的数学模型。

一般说来，在模型上处理问题和求解，比原来的非数学形式要容易些。如果给定问题的数学模型能够用解析法或数值法求解，那么可以将这个解应用于原问题。假如数学模型能较好地表示问题，那么模型的解将是问题的较好的解。反之，一个差的数学模型，即使求解很准确，其解也不是问题的较好的解。

许多问题能用一些不同的模型来表示，但是其中总有一个模型通常比其它的模型更适合，过去几十年中，在计算机的各个应用领域已建立了一批具有适当解法的独特模型。对计算机应用来说，任何一个方案的目标是要设法确定应用问题的最合适的数学模型，然后或者用已有的算法去解，或者开发新的算法。

一般来讲，数值领域中的问题容易建立数学模型，而非数值领域中的问题不易建立数学模型。因此，在实际应用中，有可能无法为给定的问题建立数学模型；或者也可以建立数学模型，但是解该模型的准确方法不能立即运用现成的算法。遇到这几种情况，通过问题分析，不必列出数学模型，可采用直观法或试探法，直接由给定问题设计算法。

1.1.3 算法设计

1. 算法的定义

一个问题经过适当地说明和分析之后，必须建立一个能提供计算机用以求解的步骤的算法。

算法是一个过程，由一组有限个清晰的规则组成，这些规则给出了解决某类特定问题的一个运算序列。于是对于某类特定问题的任何输入，按照这个运算序列一步步计算，通过有限步后，计算终止，并产生相应的输出。

算法定义的要点如下：

(1) 算法是由一组规则所组成的一个过程；

(2) 组成算法的规则必须是清晰的，没有歧义性；

(3) 由这些规则指定的运算，必须按一定的顺序执行；

(4) 这个过程必须给出特定类型问题的解；

(5) 解答必须由有限的步骤得到。

2. 算法的表示

根据算法的性质可知，算法实际上是一个抽象的概念，用它来描述解题过程。而算法的表示方法是多种多样的，下面是常见的四种表示：

(1) 自然语言（如汉语、英语）；

(2) 程序设计语言；

(3) 判定表；

(4) 流程图语言。

其中，使用自然语言描述的算法往往不够清晰，对其加以限制以后，常用的是结构化语言。程序流程图在我国则是较为普遍使用的算法描述工具。本章的1.4节将详细介绍流程图。

3. 算法的来源

能否找到现成的算法来解决实际中遇到的问题？如果能找到一个合适的现成算法来解决问题，那么可以省略建立一个新算法的过程，可以节省大量的时间，还可以避免出错。

解决特定问题的算法可以到很多地方去查找，如专业机构的出版刊物、计算机制造厂内部建立的程序库，还如解决数学和统计学问题的科学程序包、软件公司的标准程序包、正式出版物以及其它来源等。

以数据结构和科学计算为例。对于排序，有选择排序、气泡排序、分段交换排序、基数排序、快速排序、堆排序等算法；对于检索，有线性检索、折半检索、检索树、散列表技术及散列函数等算法。这些都是非数值算法，而用于科学计算的有许多数值算法，如矩阵运算、方程求解、迭代、插值、数值积分等算法；统计计算中求平均值、方差、均方差及标准偏差的方法；此外，还有求素数的各种方法，求最大公约数及求最小公倍数的方法等。

在选择算法的时候，要考虑算法的兼容性和可靠性。有时，已有的算法与要解决的问题所需要的算法不完全相同，稍作改动才能成为适合解决给定问题的算法。

1.1.4 程序设计训练

经验证明，要成为一名好的程序员，除了要熟练掌握一些语言（高级语言或汇编语言），掌握语言的各种语句和基本结构外，还必须多读多练。

多读即多阅读各种程序，学习各种程序设计方法和技巧，从而开阔眼界，丰富自己的知识；多练即要亲自动手，多编程序、改程序并且上机调试程序。

只有通过程序设计的实践，才能牢固掌握并灵活应用所学的计算机语言，不断发现和总结出程序设计的规律性，从而进一步掌握程序设计的方法和技巧，提高自己的程序设计能力。

1.2 程序设计语言

1.2.1 程序语言的组成

程序是计算机处理的对象和计算规则的描述。数据结构＋算法＝程序。

程序设计语言是用来书写计算机程序的语言。语言的基础是一组记号和规则，根据规则由记号构成的记号序列就是语言。

在了解程序设计语言时，应该注意它的三个方面，即语法、语义、语用。语法表示程序的结构或形式，即表示构成语言的各个记号之间的组合规律，但不涉及这些记号的特定含义，也不涉及使用者。语义表示程序的含义，亦即表明按照各种方法所表示的各个记号的特定含义，但不涉及使用者。语用表示程序和使用者的关系。

语言的种类千差万别。但是，一般说来，都应包含下列四种成分：

(1) 数据成分：描述程序中涉及的数据；

(2) 运算成分：描述程序中所包含的运算；

(3) 控制成分：描述程序中的控制结构；

(4) 传输成分：表示程序中数据的传输。

程序设计语言可以影响到使用是否方便，也关系到程序人员写出程序的质量。

1.2.2 程序设计语言的发展

语言的发展是从低级语言发展到高级语言的，它的发展对计算机技术的应用具有重大的作用。

低级语言的语法规则与具体计算机的指令系统紧密相关。不同机种的计算机有不同的低级语言，机器语言和汇编语言均为低级语言。机器语言是机器指令的集合，计算机能直接执行使用机器语言所编的程序。汇编语言是面向具体机种的符号式程序设计语言，用汇编语言书写的程序比用机器语言写的程序容易阅读、检查和修改，而且由于汇编语言基本保留了机器语言的灵活性，使用汇编语言也能发挥机器的特性，得到质量较高的程序。

高级语言的语法规则与计算机的指令系统没有直接关系，不同机种可以用相同的高级语言。它可以摆脱具体计算机的符号语言，以更接近自然语言（如英语）或数学符号的形式来编写程序，使不具备计算机专业知识的人能较快学会高级语言。

常用的高级语言有BASIC、FORTRAN、COBOL、PASCAL、C 等。

BASIC 语言是一种交互式会话语言，常以解释方式在计算机上执行。

FORTRAN语言是一种流行的数值计算语言，适用于解决科技和工程中的数值计算问题。FORTRAN的标准程序库十分丰富，利用标准程序，可以提高工作效率，加快开发速度。

COBOL是一种通用的事务处理用语言。它的特点是采用了树型数据结构形式，且把数据描述与程序分开，组织处理大量的数据。这种形式接近于商业、银行、财会以及各种办公室事务处理的非数值数据格式，因而易于为商业、工业和行政管理部门接受。COBOL 语言产生于60年代初期，经过修改完善，于1974年为国际标准化组织所承认，是世界上最早标准化的计算机语言。

PASCAL 语言是70年代最有影响和最重要的语言之一，也是第一个系统地体现“结构化程序设计”思想的语言，因此被认为是程序设计语言发展的一个里程碑。PASCAL 语言简单，直观易读，便于程序交流，而且由于它提供了丰富的数据类型和构造数据结构的方法，使得PASCAL 使用起来灵活方便，具有较强的功能。此外，用 PASCAL 书写的程序结构清晰，易于验证。

C 语言是为了实现 UNIX 的设计思想而发展出来的语言工具，是一种通用编程语言。它

的一个重要特色是：它既是高级语言，又是低级语言。利用C语言进行编程时，程序员不必卷入汇编语言，但在需要的时候，又可以利用机器的硬件指令，这为程序员提供了巨大的编程灵活性。由于C语言的简洁，其程序为模块化的函数，具有丰富的操作符、基本控制结构、灵活性及可移植性，使其成为当今非常流行的语言之一。

目前已经设计和实现了数百种不同的程序设计语言，而且还在不断地设计新的语言。通常认为：机器语言以及其它一些只适合于一台特定的机器的语言，统称为面向机器的语言；可用于各种机器计算各种题目的语言，统称为面向过程的语言；凡是适用于各种机器但只宜于处理专门问题的语言，叫做面向问题的语言；而专门处理特定部分问题的语言则叫作专用语言。

在语言发展的进程中，Ada语言可谓过程化语言的“顶峰”，其目标是：使语言能够体现现代的程序设计水平，并且发展一种完善的编程环境，以体现现代的软件工程设计原理，并为软件的开发和维护提供各种支持工具。Ada语言一般用于实时计算机系统。

非过程化语言是比过程化语言更高一级的语言，它只需要描述要做什么或需要什么，而无需描述怎样做或如何满足这种需要。伴随着数据库技术的广泛应用，非过程化语言迅速发展，例如SQL语言已被公认为关系数据库系统的标准化语言。在非过程化语言发展的基础上，近年来又提出了第四代程序设计语言。

系统程序设计语言是用来书写象编译系统、操作系统等系统软件的语言。

随着计算机应用的发展和“软件危机”的出现，60年代末期Dijkstra首先指出了高级语言中的“GOTO”问题，导致了结构程序设计方法的进一步兴起，使程序设计的水平和程序质量出现了质的飞跃。

同时，为了摆脱冯·诺依曼思想的束缚，出现了函数程序设计语言。

而在人工智能领域里，LISP语言一直用得很广，它是专门用于人工智能和符号处理的计算机语言。LISP利用符号来表达和处理知识，它是专门用于处理符号数据的第一种高级语言，而其它人工智能语言和工具程序容易在其基础上发展而成。PROLOG是一种具有推理功能的逻辑型语言，适合于建立专家系统的知识库，故被称为知识库语言，它也是一种面向问题的语言。

而为了将计算技术更有效地为人工智能服务，逻辑程序设计也出现了新的面貌。

随着计算机走进办公室，面向客体的程序设计正日益受到重视。Smalltalk-80就是1980年公布的一种新型编程语言。

1.2.3 程序语言的基础知识

程序结构包括数据结构和控制结构两个方面，下面分别讨论之。

1. 程序语言提供的数据结构

程序语言中通常提供一些数据类型，并提供利用这些数据类型构造数据结构的方法。但是不同的语言所提供的数据类型数目及构造数据结构的方法是不同的，在学习具体语言时，要对数据类型及构造数据结构的方法予以充分重视。

一般程序设计语言中提供了整数、实数、字符、数组、记录、文件和指针等类型，并且提供了利用这些类型构造出栈、队列、树等复杂数据结构的方法。

2. 程序语言提供的控制结构

计算机程序是算法的体现，而算法是由按某种预定次序执行的计算步骤组成的，这些步

骤的执行次序是由程序的控制结构确定的。

一般程序设计语言中提供了三种基本控制结构：即顺序结构、选择结构和循环结构；另外程序设计语言中还有一个常用的控制结构，即 GOTO 语句。

(1) 顺序结构

顺序结构意为各个语句按其被列出的先后顺序执行。一般地，设有 n 个语句 $S_1, S_2, \cdots, S_n$，则图1.1表示了顺序结构。

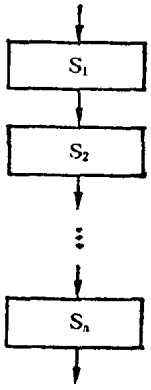
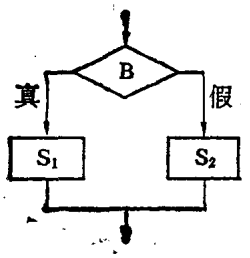


图 1.1 顺序结构示意图



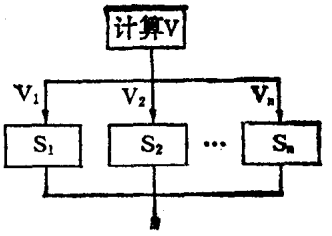


图 1.2 选择结构示意图

(2) 选择结构

选择结构意为在多个可能的语句中按一定的条件选取一个执行。

在最简单的情况下，条件表现为一个为真或为假的陈述，可供选择的语句有两个，根据条件的真或假选取其中一个执行。在程序设计语言中，一般用 if-then-else 语句来描述。

在一般的情况下，条件可以象多路开关那样在多个语句中进行选择。比如条件可以是一个变量所取的所有值，根据每一个值选择一个语句执行。在程序设计语言中，一般用分情况语句表达。

图1.2表示了两种选择结构。

其中，B是一个取真或假值的表达式（称为布尔表达式），V是一个变量，$V_1, V_2, \cdots, V_n$是V可能取的具体值；S_1，S_2，…，S_n 为语句。

注意，选择结构有一些变体，例如：我们可能判断布尔表达式B取真值还是取假值，在取真值时执行语句S，而取假值时什么也不做。这时用 if-then 方式来描述。

(3) 循环结构

循环结构意为重复，即在某条件B成立时，反复执行语句S；或者说，反复执行语句S，直到某条件B成立。

图1.3给出了先判断条件成立与否，再执行语句S的循环结构，图1.4给出了先执行语句S，再检查条件是否成立的循环结构。

在程序设计语言中，有各种各样表示循环结构的方式。例如，在 PASCAL 语言中，有 WHILE-DO 语句（与图1.3直接对应）；REPEAT-UNTIL 语句（与图1.4 直接对应）；以及 FOR 语句。

(4) GOTO 语句

GOTO 语句明确指出下一步转向何处去执行。由于七十年代初以来，结构化程序设计的

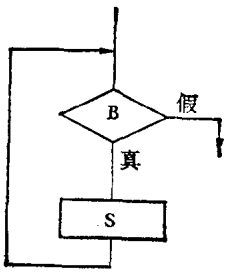


图 1.3 一种形式的循环结构示意图

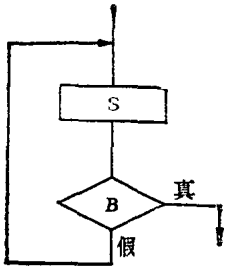


图 1.4 另一种形式的循环结构示意图

概念已被计算机界普遍接受，而结构化程序设计特别限制GOTO语句的使用，认为不加限制地使用GOTO将使程序难以理解，难以保证正确性；同时又从理论上证明了，上面介绍的三种基本结构可以构造任何程序结构，所以在编制程序时，尽量不使用GOTO语句。

但是，在一些情况下，GOTO语句还是有用的，特别是有些程序设计语言，完全不用GOTO语句将使程序编制很困难。

3. 子程序

虽然从理论上讲，有顺序、选择、循环三种基本控制结构就足够了，但在实践中，还需要引进子程序这一重要程序结构。子程序的引入大大丰富了程序设计的手段，它不仅能提高程序编制的效率，减少冗余的编码，还能使程序更加清晰、易读、易理解，并使多人共同编制一个大型程序成为可能。

在程序设计语言中，子程序的概念是由两个不可分割的部分组成的：子程序说明和子程序调用。子程序说明给出了子程序的名字，形式参数，和一个语句序列；子程序调用为主程序对子程序的引用，包括传递实在参数。在子程序说明中的形式参数与在子程序调用时的实在参数之间必须保持严格的对应关系，不仅它们的数目和类型一一对应，而且对于不同种类的形式参数，其实在参数分别受到一定的限制。

形式参数匹配是程序设计的难点之一。程序员在学习、使用某一具体程序设计语言时，一定要仔细阅读和理解该语言对形式参数匹配的具体规定。

从结构化程序设计思想来看，子程序反映了程序的两级抽象。在子程序调用一级，人们只关心它能“做什么”，而在子程序说明一级，人们才给出如何实现的细节，即“怎么做”。这种两级抽象完全符合结构化程序设计中所谓“逐步求精”的程序开发技术和“分而至之”的策略。

模块化程序设计比子程序的概念又前进了一步，但在此不再详述了。

子程序在程序设计语言中一般分为过程与函数两种。

习 题 1.1

从供选择的答案中选择正确的答案：

1. 进入80年代后，已迅速成为最常用的程序设计语言之一是[ ]。

① Smalltalk-80 ② Ada ③ C ④ PROLOG

2. [ ]是由美国国防部制定的一种新语言，它实际上是一个包含语言在内的程序设计环境。

① PL/1 ② ALGOL ③ Ada ④ C

3. [ ]是描述UNIX操作系统所用的语言，具有较强的可移植性。

① Ada ② C ③ PASCAL ④ BASIC

4. 程序的三种基本控制结构是[A]，它们的共同特点是[B]。

A. ①过程，子程序，分程序 ②顺序，条件，循环
③递归，堆栈，队列 ④调用，返回，转移

B. ①不能嵌套使用 ②只能用来写简单的程序
③已经用硬件实现 ④只有一个入口和一个出口

5. 70年代初，由N.Wirth开发的ALGOL系列的面向结构化程序设计的高级语言是[ ]。

①Ada ② FORTRAN ③ PASCAL ④ COBOL

1.3 结构化程序设计思想

1.3.1 评价程序的标准

一个好的程序首先应该是正确的，即能完成给定的任务；并且具有可靠性、易读易理解性、高效性和易维护性。

为了达到上述目标，为了提高程序质量，不仅要遵循程序设计步骤，同时还要采取适当的程序设计方法，如结构程序设计方法，模块化设计方法等。

1.3.2 结构程序设计方法

60年代末到70年代初，出现了大型软件系统，如操作系统、数据库管理系统，这给程序设计带来了新的问题。大系统常常需要花费大量的资金和人力，但研制出来的产品却常常是可靠性差、错误多，维护和修改很困难。这种现象就称为“软件危机”。“危机”震动了软件界，这就促使人们开始重新研究程序设计中的一些最为基本的问题。1969年，Dijkstra首先提出了结构程序设计概念，它强调了从程序结构和风格上来研究程序设计。结构程序设计，着眼于程序的思路清晰和结构清晰，采用自顶向下、逐步求精的设计方法设计出结构化的程序（具有良好结构的程序）。

结构程序设计强调程序结构清晰，易于理解，便于阅读。有时宁可损失一些程序执行的效率，也要保持程序的好结构。好结构程序还能提高程序的可靠性，便于检查和维护，也易于验证。实质上，在70年代初，关于要不要GOTO语句的争论，就在于是讲究好结构还是讲究效率。一般结构化的程序设计应当尽量避免使用GOTO语句。

1.3.3 程序编制风格

在编写程序时，要特别注意培养优良的程序设计风格。

程序设计风格是针对程序的易读性提出来的。要对程序进行调试和修改，程序员必须首先读懂程序，因此程序应具有易读性，即简单的编码，显意命名，消除冗余字和冗余语句，同时要条理清楚，逻辑流程简单明了。

下面是提高程序易读性的主要原则和方法。

1. 提高程序易读性的一般原则

(1) 使程序具有简单朴实的风格。即：程序中尽量采用简单的算法，不使用华而不实的古怪技巧和过于复杂的算法。