信息技术系列丛书——剑桥信息技术(CIT)全国信息应用技术证书指定用书

C语言程序设计

李宗葛

CIT

国家教委考试中心
中英教育测量学术交流中心
华中理工大学出版社

C语言程序设计

信息技术系列丛书

——剑桥信息技术(CIT)
全国信息应用技术证书 指定用书

C語言程序設計

李宗萬

华中理工大学出版社

(鄂)新登字 10 号

内 容 简 介

本书由上海复旦大学计算机系教师根据剑桥信息技术(CIT)第05模块《程序设计》的课程要求编写。

全书共分六章。第一章概述了程序设计的四个阶段;第二章提出了两个实例并进行了分析;第三章介绍了基本算法,并对两个实例选择了算法和数据结构,还细化了算法;第四章是全书主要部分,分六节详细讨论了C语言的基本规范并给出了两个实例的编程结果;第五章讨论程序测试方法并给出了对两个实例程序测试的过程和结果;第六章讨论程序文档并给出两个实例程序的用户文档和技术文档。

为了便于读者自学,在各主要章节之后均安排了一定数量的习题,供读者复习和练习。本书深入浅出,有大量实例,适合各层次初学C语言的读者。可作为相应课程的教材。

图书在版编目(CIP)数据

C语言程序设计/李宗葛一武汉:华中理工大学出版社 1996.4.
(信息技术系列丛书.剑桥信息技术(CIT)、全国信息应用技术证书指定用书)
ISBN 7-5609-1323-7
Ⅰ.C… Ⅱ.①李… Ⅲ.微型计算机一语言一程序设计 Ⅳ.TP312C

出版者:华中理工大学出版社(武昌喻家山,430074)
印刷者:中国科学院武汉分院科技印刷厂
发行者:新华书店北京发行所
开　本:787×1092　1/16　印张:14.25　字数:336 000
版　次:1996 年 5 月第 1 版　　1996 年 5 月第 1 次印刷
书　号:ISBN 7-5609-1323-7/TP.180
印　数:1-6 000 册
定　价:17.50 元

信息技术系列丛书编委会

丛书序言

人类社会已经进入了信息时代。信息技术的应用日益成为人类生活、工作、学习必备的一种基本能力。英国早我们几年就开始了这种变革。最初，教育人们学习一些深奥的理论和复杂的技术，不仅感到很困难，而且不实用。后来，对这种高技术的教学改为以实践为主的培训，这就是“CIT”。它以实际应用为目的，采用规范的标准，内容编排上为模块式结构。学习时可以根据需要自由选择。它是通过培训而不是考试，来帮助学习者掌握需要的信息技术。正因为如此，在英国以及世界上许多国家和地区，它都受到了热烈的欢迎。上述经验，对我们也有启发。毫无疑问，中国需要一批人掌握深奥的信息技术理论与复杂的信息技术，但是对于大多数人来说，只掌握需要的实用技术就够了。因为CIT具有上述一些优点，国家教委考试中心决定引进这项培训。

正是由于CIT的实用性，就要求这种教育其内容与规范必须与中国的实际相结合，而不是照搬国外的具体做法。所以，我们决定在坚持CIT规范标准的前提下，选择国内最新和最流行的计算机应用软件，编写系列丛书，作为CIT的指定用书。以期通过培训来帮助大多数人学会计算机技术的应用。这套丛书图文并茂、循序渐进，易学易懂。

我们邀请国内一些著名的专家编写这套丛书，他们夜以继日地紧张工作，圆满完成了任务。在此谨向他们致以衷心感谢。

由于我们缺乏经验，本书编写中不足之处在所难免，敬请各位读者及关心我们的同志批评指正。

国家教委考试中心主任　杨学为

1996.1

前　言

根据剑桥信息技术的模块大纲，105 模块（程序设计）是为那些没有编程经验，希望学习计算机程序设计的入门知识并能用高级语言编写程序的学员开设的。因此学员应当学会程序设计的基本方法和一种高级语言（本书中为 C 语言）的基本使用方法。具体来说，应达到下列几方面的要求：

（1）对问题进行分析；

（2）对算法和数据结构进行选择和设计并加以细化；

（3）把算法变为程序（编程）；

（4）对程序进行测试；

（5）书写文档。

由于学员的背景和课时的限制，决定了本教材在程序设计方法和 C 语言的介绍方面都应当是基本的，对算法和数据结构等仅作简单介绍和讨论。

为达到上述目的，我们在第一章中将程序设计和开发的整个过程的主要环节一一列出并作一概括介绍，给出一些基本概念。从第二章到第六章则以两个实例为背景，分别对问题分析，算法选择和细化，C 语言介绍和编程（重点），程序测试和文档书写这五个问题作介绍和讨论。

复旦大学吴立德教授审阅了全书，作了多处重要的修改；么宝刚老师参加了本书大纲初稿的制定；江杰老师对本书的出版自始至终非常关心和支持，在此一并致谢。

由于水平所限，错误难免，敬祈读者指正。

编者

1996. 3.

目 录

第一章　程序开发设计总论

本章提要

本章对程序设计的任务、目标和过程作概括性讨论。对程序设计四个阶段的任务、方法结合一个例子作出说明。以期使学员对程序设计有一个总体概貌的了解。

计算机的核心是CPU，其功能是执行规定的机器指令序列。让CPU执行不同的指令序列就能使计算机完成截然不同的工作。人们针对某个问题的要求为计算机安排的指令序列称为求解该问题的程序，程序连同有关的说明资料（文档）则称为软件。程序设计就是根据问题要求合理组合指令序列的过程。由于软件技术的发展，绝大多数程序员已不必直接去安排机器指令（或汇编指令），而是用高级程序设计语言书写源程序，再由编译软件将其翻译成机器指令序列。

程序设计的目标是得到一个正确的、高效的、可维护性好的程序。由于错误的程序达不到预期的信息处理要求，因而是无用的；效率低（时间、空间资源消耗过大）的程序代价太高或者实际上无效；另外，程序在其使用过程中常会被要求修改或扩展，只有可读性强、易于修改的程序，才能被人们理解和接收。

程序开发和设计的过程大致可分为四个阶段，见图1.1。

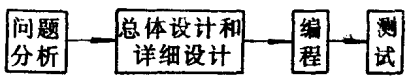

图1.1　程序开发和设计的全过程

(1)分析阶段，对程序要求解的问题作出明确的、细致的分析，确定解题的步骤；

(2)设计阶段，根据程序设计要求和解题步骤，进行总体设计和详细设计，产生程序设计说明；

(3)编程阶段，根据程序设计说明，用高级语言编写源程序，产生源程序代码；

(4)测试阶段，对程序代码、模块接口和分析阶段的错误进行检测，发现错误，纠正错误，直到最终得到正确可用的程序为止。

下面分别对四个阶段的工作作一讨论。

§1.1　问题的分析

程序分析的第一步应是对要求解的问题进行充分的分析，确定问题是什么，解决问题的步骤又是什么。程序设计中一般使用逐步求精的分析方法来完成这一步工作。

逐步求精方法的基本思想是问题分解，即把要求解的问题分解为若干功能上相对独立的子问题。如果有的子问题足够简单，可直接求解和写出程序段，则不必再分解；否则对该子问题

继续应用上述分解方法进一步分解，产生一组新的子问题。反复应用这个方法，直到所有子问题均可解为止。随着逐步求精的展开，也就可以清晰地得到解决问题的步骤。

作为逐步求精的第一步是确定问题的输入、输出和处理三大部分。

程序的输出可以是符号(数据或文字)，也可以是图形甚至于图像或语音，还可以是以上各种对象的综合。它们可以在输出设备(屏幕、打印机、照片、喇叭等)上输出，或者更一般地记录在磁盘文件中。根据求解的问题确定具体的输出对象就确定了程序执行的目标。

程序的输入可以是符号、图形、图像或语音，或者是它们的综合。它们可以从输入设备(键盘、扫描仪、摄像机、话筒等)输入，也可以从磁盘中输入。程序的输入是指为求得程序的输出而必不可少的那些原始信息或数据。根据求解的问题确定具体的输入对象就确定了程序求解问题的原始依据(出发点)。

为了简化讨论，在以下的讨论中把程序的输出和输入限于符号。

程序的主要部分是处理部分。它应当从输入数据出发，根据数学的、物理的、化学的、生物的(以及经济的乃至社会的)科学定理和公式进行计算或推理；在没有公式可循的情况下，还得根据人的有关知识进行搜索、判断和推理，最终求得所要求的结果。一般讲这一部分尤其要使用逐步求精才能最后完成问题的分析。

【例 1.1】分析以下问题：根据三角形三边长度计算三角形面积。

这是一个很简单的问题，可作如下的顶层分析(第一步分解)：

程序输出：三角形面积，一般这是一个实数。

程序输入：三角形三条边长，它们也应当都是实数。而且，由于这些实数是作为三角形边长，决定了它们必须满足一定的要求而不能是任意实数。例如，它们应当大于0，而且任意两数之和应当大于第三数，因此对它们的值应有一个检查的过程。

处理程序：在本例中是根据三角形三条边长计算三角形面积。

我们可以用图 1.2 来表示这顶层分析的结果。

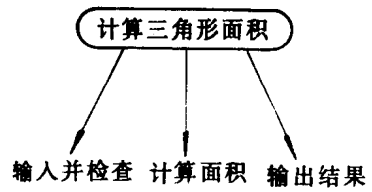

图 1.2 计算三角形面积问题的分解

也可以用图 1.3 的流程框图来表示：

其中输出计算结果的任务是直截了当的，可以不必再分解。其余两个子问题(图 1.3 中 A 和 B)可进一步分解为图 1.4 的流程框图。

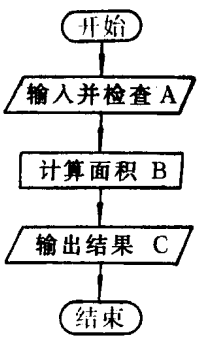

图 1.3 计算三角形面积流程框图

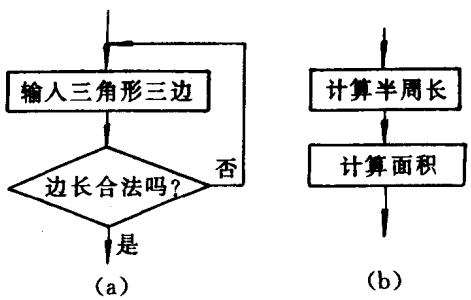

图 1.4 问题 A 和 B 的进一步分解
(a)问题 A；(b)问题 B

至此，这个简单问题的分析基本完成。

§1.2　程序的总体设计和详细设计

程序的总体设计是把问题求解步骤转换为程序的结构，确定各功能模块、设计模块之间的数据接口和控制接口，设计数据结构和文件等。

详细设计是总体结构的细化，用文字、框图、伪码或其它工具描述各功能模块的内部过程，描述其所用的算法。这一步也要采用逐步求精的方法，使最后得到的程序设计说明足够详细，以致于可以很方便地进行编程，写出程序代码。

对例 1.1 的简单例子，我们可以把它看成由 A、B、C 三个模块构成(图 1.3)。A 模块产生 a、b、c 三个边长数据作为模块输出；B 模块以 A 模块的输出为输入，产生面积 area 作为输出；C 模块以 area 为输入，进行显示输出。其模块间的控制是顺序进行的。数据全为实数，不使用文件。以上就相当于总体设计。

在详细设计时，要为各模块选择算法，确定变量，并将模块各步细化，得到程序设计说明。我们在这里和本书以后的讨论中采用文字叙述和框图相结合的方法给出程序设计说明。

由于不同的处理要求采用不同的计算方法，甚至同样的处理有时也有不同的计算方法，所以计算方法或者更一般地说算法的选择在程序设计中有很重要的作用。对算法选择的要求首先是要正确，同时要求效率高，即节省计算时间和空间。由于计算机容量的增大，有时空间的考虑比较不重要，但时间的考虑一般还是很重要的。例如三角形面积计算就有许多公式。如

$$s=\frac{1}{2}ab\sin C=\frac{1}{2}bc\sin A=\frac{1}{2}ca\sin B$$
$$=\sqrt{s(s-a)(s-b)(s-c)}$$

其中 a,b,c 为三条边长，A,B,C 为三个角，s 为半周长 $\frac{1}{2}(a+b+c)$，s 为面积。前三个公式中 $\sin A,\sin B,\sin C$ 的值未知，必须用三角公式再从 a,b,c 求出，如

$$\cos C=(a^2+b^2-c^2)/(2ab)$$
$$\sin C=\sqrt{1-\cos^2 C}$$

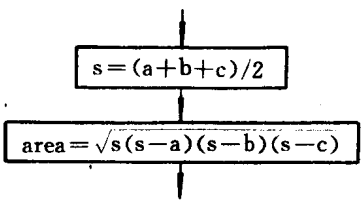

图 1.5　模块 B 的细化框图

而最后一个公式中求 s 仅为 $s=(a+b+c)/2$，然后相乘再开方。可见最后一个公式较方便。因此计算面积的子问题 B 可以细化为图 1.5 的框图：

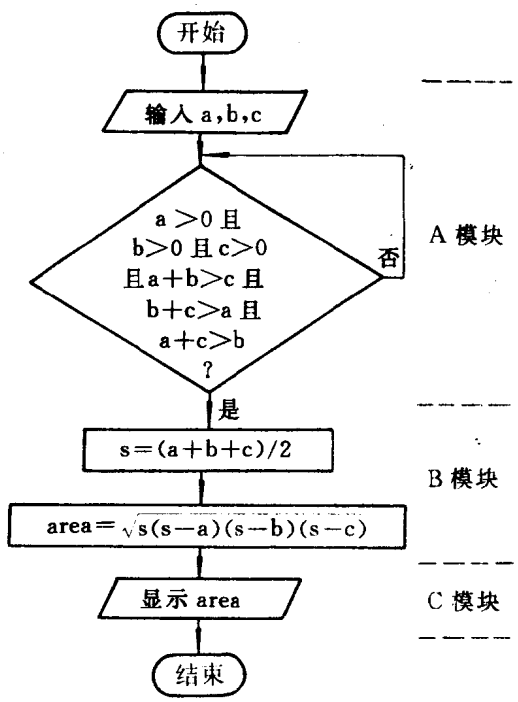

图 1.6　例 1.1 的细化程序设计说明

再将模块 A 中对三角形三条边的合法性条件检查进行细化，就可以最后得到图 1.6 的框

图形式的程序设计说明。

§1.3 编 程

编程是程序设计的实现阶段。这个阶段将已经细化的详细设计说明转换成所要求的用某个高级语言书写的源程序。

编程是前一阶段设计的自然结果，它的质量主要取决于前一阶段设计的质量。但是所选择的高级语言的特性、编程方法和风格也会对程序的可靠性、可读性、可测试性和可维护性产生重大影响。此外，编程时还很容易引入各种错误，因此还必须进行代码复查以及下一步的程序测试等步骤。

本节对编程阶段的有关问题作一些原则讨论，详细的实施将在第四章中进行。

1.3.1 结构化程序设计

结构化程序设计的原理思想是由 E. W. Dijkstra 等人于 60 年代后期提出的。按照这一原理，一个程序的任何逻辑问题均可用“顺序”、“选择”和“循环”这三种基本逻辑结构来描述。结构化的程序应由若干个基本结构(块)组成，每个块可以有一个或若干个语句。

结构化设计应遵循下列准则：

(1)使用有限数量的基本逻辑结构；

(2)利用基本逻辑结构组成块；

(3)每块都有且只能有一个入口和一个出口；

(4)少用和不用 GOTO 语句。

使用结构化程序设计的优点是：

(1)结构化构造减少了程序的复杂性，提高了可靠性、可测试性和可维护性；

(2)使用少数基本结构，使程序结构清晰，易读易懂；

(3)容易验证程序的正确性；

(4)构造过程符合人的思维从粗到细的过程，可逐步求精、细化。

结构化设计中使用的三种基本结构是：

1. 顺序结构(见图 1.7)

先执行 A 操作，再执行 B 操作。A、B 为块结构，两者之间是顺序执行的关系。

2. 选择结构(见图 1.8)

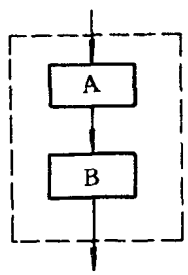

图 1.7 顺序结构示意

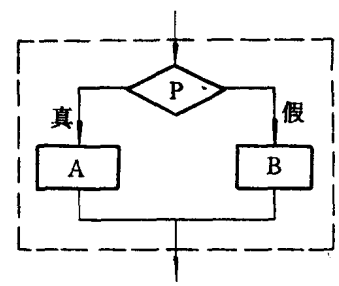

图 1.8 选择结构示意

其中 P 代表一个条件判别，当 P 条件成立(或称为“真”)时执行 A 操作，否则执行 B 操作；

两者只能择一执行。两条路径会合后由一个出口出去。

3. 循环结构(见图 1.9)

(1)图 1.9(a)为当(while)型循环结构。当 P 条件为"真"时,反复执行 A 操作,直到 P 为"假"时才停止循环,由出口出去。

(2)图 1.9(b)为直到(until)型循环结构。先执行 A 操作,再判断 P 是否为"假";若为"假",再执行 A 操作,如此反复,直到 P 为"真"时才停止循环,由出口出去。

由选择结构可以派生出另一种基本结构,见图 1.10。根据 K 的值(K_1,K_2,…,K_n)不同而决定执行 A_1,A_2,…,A_n 之一,可称之为多分支选择结构。

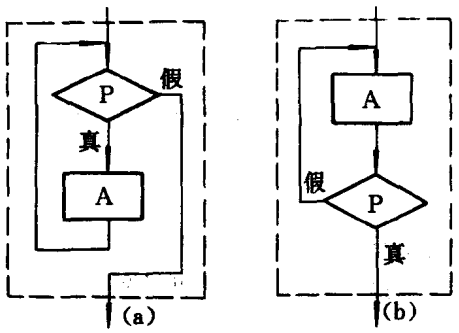

图 1.9　循环结构示意

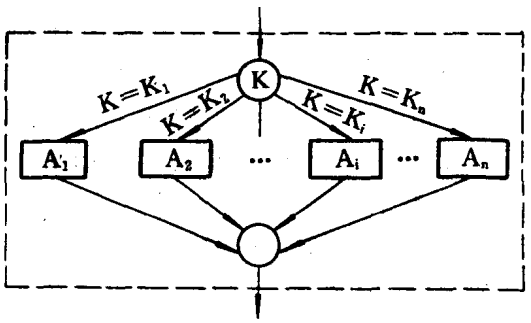

图 1.10　多分支选择结构示意图

已经证明,由以上基本结构组成的程序能处理任何复杂的问题。我们已经说明,图 1.7～图 1.10 中方框中的 A,B,A_1,A_2,…,A_n 等是块结构,它可以是一个简单语句,也可以是一个基本结构或复合结构,它们还可以是嵌套的。

我们在编程中应当采用结构化程序设计的思想和方法,遵循上述各项原则。

1.3.2　高级语言的选择

高级语言本身的特点和构造细节对作出选择有决定的作用。

一、高级语言的分类

1. 基础语言

基础语言是开发早(50～60 年代),早期应用广泛的语言,如 FORTRAN 和 COBOL 等。

FORTRAN 是工程和科学计算使用的主要程序设计语言。它的早期版本 FORTRAN-66 为计算问题的求解提供了较强有力的工具。其缺点是不能直接支持结构化的构造,数据类型少,不易支持字符串处理等。较新的标准(ANSI FORTRAN-77)纠正了早期版本中的某些缺陷。

COBOL 作为标准语言已为商业数据处理广泛应用。它有很好的数据描述,支持商业应用领域内涉及的各种处理。

2. 结构化语言

结构化语言如 PASCAL,C 等,有很强的结构化特点和数据结构描述功能。它们在工程、科学、商业和系统控制方面有广泛的应用。

PASCAL 是 70 年代初开发的一种结构化程序设计语言。它具有许多结构化特点:块结构,丰富的数据类型,支持递归等。

C 语言是作为 UNIX 操作系统的主要语言发展起来的。在各种型号的机器上它已独立于

UNIX 而存在。由于高效率、可移植性强而得到越来越广泛的应用。它不仅用于开发系统软件，而且用于开发应用软件。再往后，它又被发展成为 90 年代的面向对象的高级语言(C++)。因此，其适应性很强。这也是选用 C 语言作为我们的工作语言的原因。

3. 专用语言

专用语言是为特定应用范围设计的高级语言，一般来说，这类语言的应用面较小。例如：

APL 是为数组和向量处理而设计的，是解决数学问题的有力工具，但不支持结构化构造和数据类型。

APT 是为机械工具的数值控制而设计的，包含描述零件几何形状的方法、切割工具动作命令和通用语句。

LISP 是为人工智能的符号处理而设计的函数型语言。

PROLOG 是为人工智能的逻辑推理而设计的逻辑型语言。

专用语言对其应用领域是适当的，但与前两类语言相比，相对缺乏可移植性和可维护性。

二、选择高级语言的依据(实践标准)

1. 编程和维护成本

语言选择不当会显著增加程序的编码量和日常维护的困难。一般来说，专用语言要求较少编码量，但可读性不一定好，也不一定容易维护。因此在选择语言时要兼顾编码量和维护工作量。

2. 兼容性

在一个大的系统中可能有多台机器，这时必须考虑主系统和子系统之间的兼容性，选用共同的语言更合适。

3. 程序的可移植性

如果程序的生存周期很长，又可在不同硬件环境中运行，则选择可移植性好的语言是很重要的。

4. 语言本身的特性

对选择语言有影响的特性包括控制结构，抽象数据类型，模块及子程序功能，意外错误处理功能等。它们会影响程序的可靠性、可读性和可维护性。

5. 应用领域

如对于科学计算可选 FORTRAN，对于操作系统或其它系统软件应选 C，商业数据处理可选用 COBOL，人工智能问题则应考虑用 LISP 或 PROLOG。应当说，C 语言的适应面相当广，既可写系统软件，又可写应用软件。

1.3.3 编程风格

程序的编写人员和维护人员是不同的。因此，建立一套为程序员和程序阅读者公认的编程原则是很重要的。只有按这些原则编程，才能编写出正确的，高效的，可维护的，易读的程序。应当尽量避免深奥的，技巧性强的特殊编程风格。下面我们讨论一些常用的编程风格，对于某些具体符号的含义可暂时不必介意，以后再逐步介绍它们。

一、使用注解

程序中的注解有很重要的作用，正确使用它能改进和加强程序的可读性、可维护性。注解可分为三类：

1. 序言性注解

每个程序或子程序开始处应有注解，描述其功能、用法、特殊变量的说明等。

2. 工作性注解

表示何时何人对某程序段进行过修改的注解，可使人们了解程序修改变动的历史。

3. 解释性注解

在程序不够直观的地方，或在重要循环和条件语句之前加上解释性注解，解释程序正准备做什么，帮助程序阅读者了解程序的控制流和数据流。

高级语言的使用大大减少了注解的数量，只须在重要的分支、动作或功能块上才加适当注解。加注解一般应当在写程序时进行，因为当时最了解细情，而且可以为以后的设计、调试和测试阶段所利用。

二、空格和空白行的使用

空格应放在能增强可读性的地方。如：

```
if(i==5)j++;
```

采用空格后更易读：

```
if  (i==5)  j++;
```

空白行是增强程序外观的一种方法，用它可以分隔程序的段落或语句组，便于在程序中查找有关段落。

三、标识符的选择

标识符是程序中用到的文件名、变量名、函数名、子程序名等用户定义的名字的统称。恰当选择标识符是增强程序可读性的一个重要方面。不仅要符合高级语言本身的语法规范，而且应当恰如其分地表达其内容含义。例如

$$Z=X+Y$$

如无说明，其含义是非常不清的。而

$$Price=Cost+Profit$$

则直观地表达了等式的含义，即价格等于成本加利润。在程序中应尽量避免容易混淆且毫无意义的标识符，也不要使用高级语言的保留字(关键字)作标识符，因为它们已经有专用含义，不应再用。

当高级语言对标识符长度具有较严限制时，可采用标准缩写以满足语法要求，它可便于程序阅读者展开和理解。标准缩写的规则为：

(1)必须保留第一个字母；

(2)辅音字母优于元音字母；

(3)字的开始优于结尾；

(4)缩写到 3～15 个字符之间。

缩写的做法是从右向左连续删除第一个字母之外的所有元音，直到满足长度限制；如还太长，则再从右至左删除辅音，直到满足长度要求。如 TRANSACTION 缩写为 TRNSCTN。

四、语句的位置

语句在版面上的位置，在一定程度上反映了程序的结构，养成一个好的习惯后编写的程序其易读性也会更好。一般最好每行写一个语句，便于增删语句，编译报错时易于对错误定位，也便于增加注解等。语句位置安排的一个重要规则是锯齿排列。这样能较明显地表示程序的层次结构。如：

```
for  (i=0;  i<10;  i++)
```