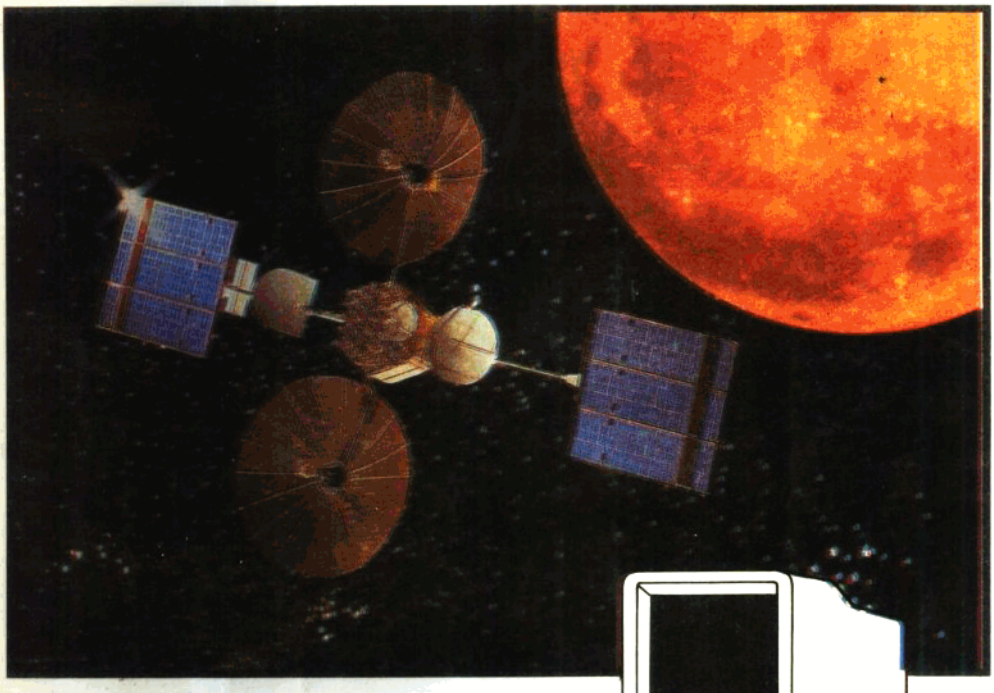
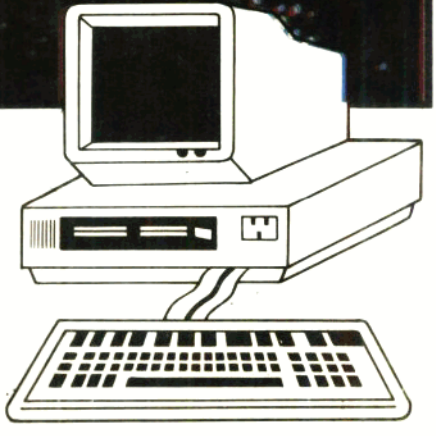
HOPE 北京希望电脑公司计算机技术丛书

计算机高级语言实用大全

1





海 洋 出 版 社

北京希望电脑公司计算机技术丛书

计算机高级语言实用大全

朱巧明　崔志明　陈时飚　　编著

上　　册

海　洋　出　版　社
1993年·北京

内 容 简 介

随着计算机技术的不断发展，计算机程序设计语言也层出不穷。目前已设计和实现的程序设计语言有数百种。本书精选了这数百种程序设计语言中在微型计算机上经常使用的十多种计算机高级语言。书中把这些高级语言按其使用对象分为BASIC语言类、C语言类、面向对象的程序设计语言类、数据库管理语言类、人工智能与知识工程语言类以及不在这些类语言中但常用的多种高级语言。本书系统地介绍了这些高级语言的发展、特点、命令、函数以及操作。

全书共分上、下两册，内容丰富、重点突出、实用性强。它可作为大专院校软件专业学生、研究生程序设计课程的教学参考书，也可作为从事计算机软件工作人员的参考手册。

需要本书的用户，请直接与北京8721信箱联系，邮码：100080，电话：2562329

(京)新登字087号

北京希望电脑公司计算机技术丛书

计算机高级语言实用大全

朱巧明　崔志明　陈时飚　编著

*

海洋出版社出版（北京市复兴门外大街1号）

海洋出版社发行　常熟市教育印刷二厂印刷

开本：787×1092　1/16　印张：52.75　字数：1260千字

1993年7月第一版　1993年7月第一次印刷

印数：1—5000

*

ISBN 7-5027-3557-7/TP·199　　定价：55.00元/套(2册)

前　言

随着计算机技术的不断发展，计算机程序设计语言也层出不穷。目前已设计和实现的程序设计语言已有数百种。在这些程序设计语言中绝大多数是高级语言。本书主要介绍微型计算机上常用的十多种高级语言。

尽管介绍各种高级语言的书已有很多，但至今还没有一本把这些高级语言分门别类、系统归纳、比较完整的书籍介绍给读者。通过多年来的程序设计课程教学与实践，特别是对各种高级语言进行研究之后，我们觉得，有些语言是相通的，有些语言则虽是同一类别，但由于属不同版本而使得其操作使用、执行方法都不相同。

从面向对象的程序设计技术来看，高级语言可分为：面向过程的、面向问题的和专用的三类；从过程化和非过程化来分，可分为过程化程序设计语言和非过程化程序设计语言。在本书中，我们按高级语言的使用对象分，分为 BASIC 语言类、C 语言类、面向对象的程序设计语言类、数据库管理语言类、人工智能与知识工程语言类以及不在这些类中但经常使用的多种高级语言。

全书共七章，第一章介绍了程序设计语言基础知识；第二章介绍了各种版本的 BASIC 语言；第三章介绍了标准 C 和 Turbo C 以及 Microsoft C；第四章介绍了 Smaltalk 和 Barland C^{++} 两种面向对象的程序设计语言；第五章介绍了 dBASE 和 FOXBASE 两种数据库管理语言；第六章介绍了人工智能与知识工程语言，包括 LISP，PROLOG，TURBO PROLOG，M·1 和 OPS5；第七章介绍了 FORTRAN，COBOL，PL/I，PL/M－86，ADA，LOGO 和 PASCAL 等高级语言。书中对这些语言的特点、发展、命令和函数、编程技术作了全面的介绍。

在本书的形成过程中，得到了钱培德教授的热情鼓励、帮助和指导。另外为本书进行审稿的教授和专家，对书稿提出了不少有益的意见。我们谨在此向这些同志深表谢意。

本书由朱巧明、崔志明、陈时飚、邵斌、王俊标合作完成，其中第一章、第五章及第七章的第七节由朱巧明执笔，第二章由陈时飚执笔，第三章由邵斌执笔，第四章、第六章及第七章的第三、第四节由崔志明执笔，第七章的第一、第二、第五、第六节由王俊标执笔，另外，杨季文、吕强、邵刚等同志也参加了部分工作。全书由朱巧明主持编写并最后定稿。由于我们水平有限，所以尽管在各方面作了很大的努力，但是书中一定仍会有错误和不足之处，还望读者不吝指教。

作者

1993 年 4 月于苏州大学计算机工程系

计算机高级语言实用大全

目　录

第一章　程序设计语言基础

第一节　概　　述

一、程序设计语言的组成

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。它是由一组记号和规则组成，并根据规则构成的记号序列。

每一个程序设计语言都包括语法、语义和语用三个方面。语法表示程序结构形式，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，即表示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，都应包含下列四个成分：

(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的方便性，也关系到程序员编写出程序的质量。

二、程序设计语言的发展

计算机语言的发展是从低级到高级发展的。它的发展对计算机技术的应用具有重大的作用。从40年代第一台电子计算机ENIAC问世至今，计算机语言已经历了三个阶段：机器语言阶段；汇编语言阶段和高级语言阶段，现在已出现了第四代语言。

机器语言就是计算机的指令系统。其指令代码即为一串二进制数。汇编语言采用一定的助记符来表示机器语言中的指令和数据。用汇编语言编写的程序较之用机器语言编写的程序容易阅读和修改。高级语言采用与人们日常使用的表达形式相近的形式编写程序，根据其执行方式，高级语言有编译执行和解释执行两种。第四代语言一般是指具有下列特点的高级语言：

(1)程序设计效率高，编码时间短；

(2)系统开发工作由专门程序向最终用户转移；

(3)程序设计在较高层次上进行，实现一定程度功能所需源程序语言数目有所减少。

从语言的语法规则、语义和语用特征来看，机器语言和汇编语言的语法规则与具体机器的

指令系统紧密相关，因此它们属于低级语言。不同机种的计算机有不同的低级语言。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。高级语言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 语言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，且把数据描述和程序分开，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 语言于 60 年代初期研制成功，经过修改完善，于 1974 年为国际标准化组织(ISO)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，使得程序人员使用起来灵活方便，具有较强的功能。此外，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的数百种程序设计语言可归纳分类成面向机器的语言、面向过程的语言、面向问题的语言和专用语言。

面向机器的语言包括机器语言以及其它一些只适合于一台特定的机器的语言；面向过程的语言指用于各种机器计算各种题目的语言；面向问题的语言是指适用于各种机器但只宜处理专门问题的语言；专用语言则指专门处理特定部分问题的语言。

在计算机语言发展进程中，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。

随着计算机应用的发展和“软件危机”的出现，60 年代末期 Dijkstra 首先提出了高级语言中的“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，在这期间，为了摆脱 Von Neumann 思想的束缚，出现了函数程序设计语言。

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。

三、基本概念

1. 数据类型与数据结构

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。

数据结构是一门研究计算机程序加工处理的信息之间的结构关系的学科。它主要研究数据的逻辑结构、物理结构以及两者间的关系，其中数据的物理结构是指数据结构在计算机中的具体表现；同时它还研究在数据的结构上定义相应的封闭运算，其中的封闭是指经过运算后的新结构不改变其结构类型。

一般的程序设计语言提供了整数、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。

值得注意的，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。

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，这些步骤的执行次序是由程序的控制结构确定。

一般程序设计语言中提供三种基本控制结构：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：GOTO 语句。

顺序结构是指程序中各条语句按其被列出的先后顺序执行。这种结构比较简单、直观，但只能处理一些极其简单的问题。

选择结构意为在多个可能的语句中按某一条件选取其中一个分支执行。在最简单的情况下，条件只有一个，或为真，或为假。这时可供选择的语句（转移分支）也只有两个。在程序设计语言中一般用 if－then－else 语句来描述。在条件不至一个情况下，程序控制可以象多路开关那样在多个语句（转移分支）中选择其中之一执行。在程序设计语言中，一般用分情况语句表达。

循环结构即为反复执行的一个结构。当条件成立时，反复执行一句或一段语句；反之，则退出循环体。在程序设计语言中，有各种各样表示循环结构的方式，如在 PASCAL 中，有 WHILE－DO 语句；REPEAT－UNTILE 语句，在 C 语言中有 FOR 语句，WHILE 语句和 DO－WHILE 语句。

GOTO 语句明确指出下一步转向何处去执行。由于 70 年代初以来，结构化程序设计已被计算机界普遍接受，而结构化程序设计中特别限制 GOTO 语句的使用。

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。

在程序设计语言中，子程序的概念由两个不可分割的部分组成：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的实在参数之间必须保持严格的对应关系，不仅其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。

从结构化程序设计思想看，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“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“怎样做”。这种两级抽象

符合结构化程序设计中的“逐步求精”程序开发技术和“分而治之”的策略。

第二节　结构化程序设计

一、结构化程序设计概念

在60年代中期以前，程序员只凭自己的经验和习惯来编写程序，由于没有一种共同的程序设计标准，因此，对同一模块，即便使用相同的程序设计语言，不同的程序员也有不同的设计方法，编写出来的源程序千差万别。由于程度的复杂度较小，尚能满足当时的应用要求。随着计算机技术的飞速发展，以及计算机应用的日益扩大和深入，对技术的要求日益复杂，程序就越来越复杂。程序的维护、调试要比编写一个程序所化费的时间大得多。其中最突出的问题是程序的可维护性。

另一方面，程序员编写程序的工具极有限，只有几个程序流程图的符号，用这些符号把自己解题过程表达出来。这种表达方式往往是任意的，并没有一种准则可循，因此也就很难证明自己的程序逻辑结构是正确的。

因此多年来，人们一直致力于程序设计方法的研究，希望制订一套标准的方法和原则，以便程序员依照共同的准则和方法来从事程序设计工作。E·W·Dijkstra是最先提出结构化程序设计方法来构造一个程序。他建议取消GOTO语句以降低程序的复杂性，但他并没有具体地提出结构化的方法所包含的内容。直到1971年，N·Wirth在《用逐步求精的方法开发一个程序》的论文中提出用一系列求精的步骤来构造一个程序，其中每一个步骤都是把一个任务分解成若干个子任务的过程。对程序描述求精过程应和对数据描述的求精过程同时进行，以便解决在子任务之间的通讯问题。从此，就建立了结构化程序设计的概念。

所谓结构化程序设计是指：用一组标准的准则和工具从事程序设计，这些准则和工具包括一组基本控制结构，自顶向下地扩展原则，模块化和逐步求精。

结构化程序设计的主要优点是程序易于理解、使用和维护。程序员采用结构化程序设计方法，降低了程序的复杂性，因此容易编写程序。程序员能够进行逐步求精、程序测试和证明，以确保程序的正确性，程序容易阅读并被人理解，便于用户使用和维护。另外，通过结构化程序设计，提高了程序设计的工作效率，降低了软件开发成本。由于结构化程序设计方法能够把错误控制到最低限度。因此能够减少调试时间。

二、程序的基本结构

任何一个程序，它所执行的控制流程和指令，都可以直接用一张流程图来表示，这个程序中的每一条指令对应着流程图中的一个结点，程序中每一个可能的控制流程序对应着流程中的一条带箭头的直线。程序的结构一般可用线性结构、条件结构、循环结构和选择结构表述。

1. 线性结构

线性结构程序中，语句或结构按顺序连续执行，如图1－1所示。在序列中，计算机先执行S1，在执行S2，最后执行S3。线性程序只有一个入口和出口，是程序结构中最简单的一种结

构。

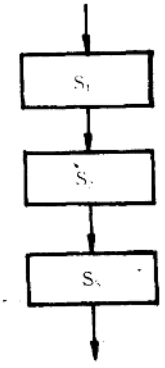


图 1－1 线性结构

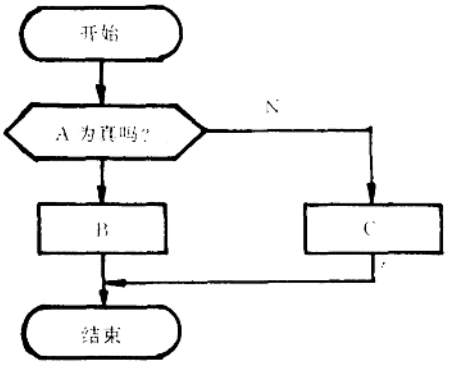


图 1－2 条件结构

2. 条件结构

常用的条件结构是 IF－THEN－ELSE。假设条件 A，B 和 C 是语句或语句序列，如果 A 为真，则执行 B；否则执行 C，图 1－2 给出了这种条件结构。同样，它也只有一个入口和一个出口。

3. 循环结构

循环结构的典型例子是“WHILE C DO S”结构，其中 C 是条件，S 是一语句或语句序列。该语句的含义是先检查条件 C，若 C 为真，则执行 S，否则退出循环。若 C 一开始就为假，则 S 就根本不执行。这种循环结构也只有一个入口和出口。这种结构如图 1－3 所示。

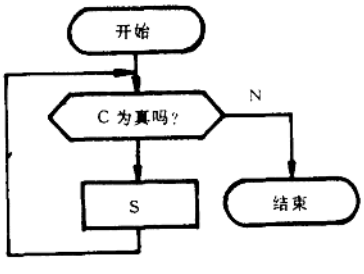


图 1－3 循环结构一

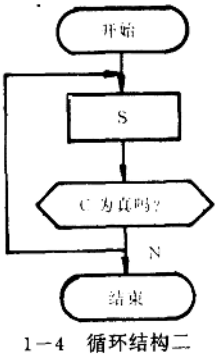


图 1－4 循环结构二

循环结构的另一种形式如图 1－4 所示。其基本结构是“DO S Until C”，其中 C 是条件，S 是一语句或语句序列，这种结构能保证循环体 S 至少执行一次。

4. 选择结构

选择结构也是一个单输入单输出结构。典型的结构是“CASE I OF S0，S1，…，SN”结构。其中 I 是下标，S0，S1，…，SN 是语句或语句序列，若 I＝0，则执行 S0，若 I＝1，则执行 S1 等等。程序只执行 N 个语句中的一个。执行完之后控制便转到此选择语句组之后的下一句语句。如果 I 大于 N，则不执行任何语句而直接转到选择语句组之后的下一语句，选择结构如图 1－5 所示。

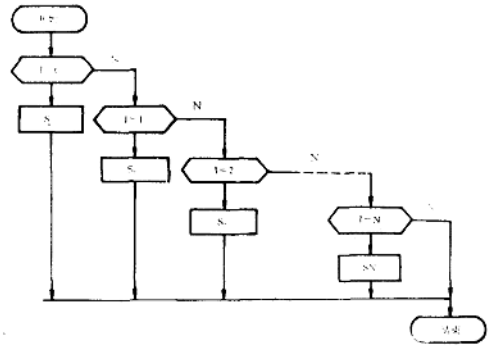


图 1－5 选择结构

三、自顶向下的模块化设计

1. 自顶向下的程序设计原则

自顶向下程序设计的第一条原则是先把一个程序高度抽象，看作是一个最简单的控制结构，即功能结构。

在完成系统分析后，系统设计员就把系统结构图中的各个模块连同其说明书一起交给程序员，由程序员进行程序设计。程序员根据系统分析人员提供的模块说明书进行设计工作。

假设模块说明书中的程序名是PROG，输入数据是x，输出数据是y，则根据自顶向下程序设计的第一条原则，该模块可用一条赋值语句完成：

PROG：X＝Y

这样就明确了该程序的主要功能，最理想的是，存在这样一台机器，只要写入“PROG”四个字，程序员的工作就算完成。这台机器在接收到这条指令后，能够自动运行，满足模块说明书中所规定的要求。但实际上至今还未存在这样的机器。因此，还需要人工来完成设计。

自顶向下的第二条原则是为了完成这个主要功能，需要进一步分解成若干个较低层的模块，每一个下层模块都有一个名称，表达一个较小的功能。通过这种逐层扩展直到最低一层的每个模块都非常简单，功能很小，能够很容易地用基本结构实现为止。

2. 层次模块图

层次模块图可以把抽象后的层次模块具体地表达出来，以反映一个程序中各层次模块之间的关系。假设把PROG看作是最高层，运用自顶向下扩展原则，分解成P_1，P_2，P_3三个下层模块，再反复运用该原则，又扩展更低一层模块，如图1－6所示。

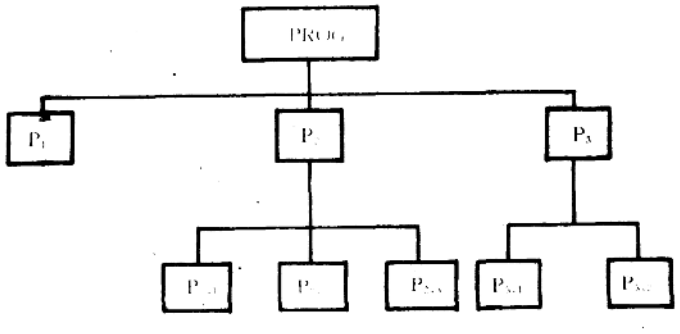


图 1－6 程序PROG的层次模块图

层次模块不同于程序流程图，前者不反映数据流向，也不反映判断逻辑或循环逻辑，它只反映程序的功能，以及在这个程序中各个功能间的关系。因此，层次模块图反映了程序要执行的功能，而程序流程图反映了程序的执行过程。前者强调要做什么，后者强调如何去做，即逻辑与物理的关系。

运用自顶向下的扩展原则把一个程序分解成具有层次结构的功能模块，这些模块应尽可能地彼此独立，这样便于维护。

四、逐步求精

层次化方法将一个大型的程序系统自顶向下地逐层分解为较小的模块，从而降低了程序的复杂性。用一组基本控制结构来构造一个结构化程序，其目的是提高程序的生产效率。然而，这两种方法最初都只是面向代码级的，而且是互不相干的，实际情况是根据设计说明书的要求来编制程序的，即先分析功能要求，再设计程序。

所谓逐步求精是指这样一个过程：把一个模块的功能一步一步地分解成一组子功能，而这组子功能可以通过执行若干个程序步来完成该模块的全部功能。

逐步求精实际上是和自顶向下的程序设计并行的，后者是用层次模块图的工具把一个程序分解成若干个模块，但是它只表达了程序中各个功能之间的关系，却不能表达模块的内部逻辑。采用逐步求精的方法能把一个模块的功能逐层地进行分解。

第三节　各种语言编程举例

程序员一旦完成程序设计，就进入编码阶段。编写同一个程序的实际步骤因所使用的语言不同而不同。但是就像系统设计一样，程序编写也有一定规则可循。例如，要实现打印机换行功能。尽管这个程序简单且很小，但也足以说明问题。完成这一任务可分为四个步骤，其处理过程可描述如下：

```
begin
    {
  storę formfead code to temporary
  open printer device
  send temporary to printer
  close
    }
end
```

第一步，先与设备（或文件）进行通讯，以建立起数据通道；第二是对发送的数据分配一个暂存单元；第三是发送一个数据；最后要关闭设备或文件。

在程序的编写过程中，程序员必须同时建立文件的档案材料，这个材料的内容可以很仔细，包括每条指令的注释，但也可很粗，只有几个字或几行字。下面是用各种语言实现的例子。

如果用 8086 汇编语言来编写程序的话，则源码如下：

```
;Formfeed Utility,Send a Formfeed 0CH to the
;Current List Device
;
    MOV  AH,  5CH          ;置 5 号功能调用号
    MOV  DL,  0CH          ;置 0CH 换页符
    INT  21H               ;功能调用
    MOV  AH,  4CH          ;通过 4CH 功能调用
    INT  21H               ;实现程序正常结束
```

若用 BASIC 语句，则程序如下：

```
10  REM  SEN  PROMFEED  TO  PRINT
20  LPRINT  (HR$112);
30  END
```

用 C 语言，则程序如下：

```
/* Send formfeed to system printer device */
    #inclucle <stdio.h>
    FILE *fp
    main( )
      { fp=fopen ("PRN",)
        putchar (fp,0X0C);
        fclose (fp);
      }
```

若使用 dBASE，则

```
*Send formfeed to printer
eject
```

从上述例子可以看出，程序源码文件由可执行的代码和注释部分组成。前者是程序的核心，而后者则是程序的必要部分，其它程序就变得不可理解和维护性差。

另外，使用各种不同的语言其目标代码的长度，程序的执行方式等都是不同的。在学习程序设计语言或用程序设计语言解决某一问题时，都要考虑到当前计算机的硬件环境和系统软件环境等。这些内容我们将在以后的章节中作充分的介绍。

第二章　BASIC 语言

第一节　基本 BASIC

一、BASIC 语言的发展史及特点

BASIC 语言是在国际上流行一时的一种计算机小型算法语言，其全称是 Beginner's All－Purpose Symbolic Instruction Code，即初学者通用符号指令代码。

BASIC 是由 FORTRAN 语言发展而来的，它抛弃了 FORTRAN 复杂难学的缺点，诸如繁琐的书写格式等。在对 FORTRAN 语言进行简化精炼之后，1964 年，BASIC 语言的第一个版本问世了。BASIC 语言诞生之初，以其简单易学和强大的对话功能而立刻受到广大用户的欢迎，尤其受到非专业用户和初学者的青睐。1978 年和 1980 年分别建立了基本 BASIC 语言的美国国家标准和国际标准。众多的 BASIC 版本针对基本 BASIC 标准的不足而开发增加了丰富的音乐、图形和字符处理功能，使 BASIC 语言从仅可进行单纯的数值处理转化为可以进行复杂的数据处理，从而将 BASIC 语言的应用领域从单一的科学计算扩充到各行各业。其中较为出色的版本有：Microsoft GW－BASIC、Ture－BASIC、Turbo－BASIC 及 Quick－BASIC 等等。时至今日，小型机和微型机基本上仍都配有 BASIC 语言，大多数非专业初学者更是把 BASIC 语言做为入门学习的首选语言。

BASIC 语言的基本特点归纳有如下四点：

1. BASIC 语言是解释型语言。

计算机语言就其执行方式而言，不外乎可分为解释型和编译型两种。解释型语言是编译一句执行一句，并且不产生最后的可执行代码。编译型语言是将源程序全部编译成最后的可执行代码文件，然后再执行该文件。作为解释型的语言，BASIC 语言具有强大的人机对话功能。用户输入的源程序若有错误，计算机将及时指出，此时用户就可以立即修改错误之处，这一点很受初学者的欢迎。当然，解释执行的速度较编译执行要慢，但在 BASIC 语言的几个扩充版本中，都增加了 BASIC 语言的编译功能，这样就既可满足初学者对人机对话的要求，也可以满足熟练用户对速度的要求。另外，BASIC 语言还提供键盘运算功能，即具有全仿真科学函数计算器功能。

2. BASIC 语言简单易学。

基本 BASIC 语言共有 17 条语句和 12 个基本函数。作为一种计算机高级语言，BASIC 语言的语句定义符采用英文单词，而运算符则和数学中的符号基本相同，所以学习和记忆都很方便。

3. BASIC 语言有一定的数据处理功能。

从数值处理到数据处理是计算机应用的一个飞跃，其中也有 BASIC 语言的一份努力。BASIC 语言的几个扩充版本在音乐、图形和字符处理方面都有出色表现，使 BASIC 语言完全可以胜任一般事务管理的编程。

4. 基本 BASIC 语言是非结构化的算法语言。

BASIC 语言早期的几个版本中，GOTO 语句的使用是较为频繁的，这就使 BASIC 语言的程序设计风格受到影响，不可能是结构化程序设计。BASIC 语言稍后的几个版本，注意到这一点，提供了完善的循环控制语句，使设计风格转向结构化设计风格。BASIC 语言诞生之时，由于硬件条件限制，所以使用的硬件资源很有限，例如最大可支配内存容量为 64K，这就限制了 BASIC 语言编程的规模，例如数的范围、变量个数和数组维数等都受到限制。随着计算机硬件的发展，BASIC 语言的扩充版为充分发挥计算机硬件的潜能做了很大努力，内存支配权从 64K 扩大到 384K 甚至更多。

二、BASIC 语言的基础知识

计算机语言是英文单词、数字和特定符号的集合，它为人们提供用能被计算机理解的方式去描述问题的手段，即人机接口。通俗地说计算机语言也就是人和计算机进行交流时采用的语言。作为语言，就必然会有语法结构、词汇等，计算机语言也不例外，它对应有程序结构(program structure)、关键字(keyword)。下面看一个 BASIC 程序实例。

```
10  REM 计算半径为 10m 的圆面积
20  LET R=10 : LET P=3.1415926
30  LET S=P*R*R
40  PRINT  "面积=";S
50  END
```

以上就是一个 BASIC 源程序(Source program)。通过这个例子程序，可以看出 BASIC 语言的基本结构如下：

1. BASIC 程序的基本单位是行，一行内可以有一个以上简单语句，语句之间以冒号间隔。

2. 每条语句由三部分组成。

(1)标号：这是 BASIC 语言的特色所在。标号是一个无符号整数，它指示了程序的执行顺序，即按标号从小到大执行。源程序输入时，标号顺序可以是任意的，输入完毕后，计算机会自动将源程序按标号从小到大排列好。如果一行内有多条语句，那么从第二个语句开始，以冒号代替语句标号，例如上面例子程序 20 句的书写格式。此外，为了便于修改时插入语句方便，语句标号最好有一定间隔，当使用系统提供的标号自动产生功能(AUTO 功能)时，标号间隔是 10。

(2)关键字：它是语句中的操作符，规定语句要执行的操作。例如 REM 表示注解操作、LET 表示赋值操作、PRINT 表示打印操作。

(3)语句体：它是语句中的操作数，为关键字执行时提供数据，其位置在关键字后面。

3. 程序通常以 END 做为结束符。执行程序时，遇到 END 语句或没有更大标号的语句时就停止执行，所以当 END 语句的标号是最大标号时，可以省略该语句(在某些情况下)。

4. 程序输入计算机后，就被保存到内存中，键入 RUN 命令后程序被执行。除非退出 BA