第一章 绪 论

§1 程序设计及其语言

所有计算机软件的构成都离不开程序模块，即软件的实体是程序。因此，有必要先了解程序设计基础以及程序设计语言的概貌。

§1.1 程序设计过程

随着科学技术的高度发展，电子计算机正以惊人的速度迅速地进入各个生活与生产领域中，尤其是科学计算、数据处理、自动控制等方面已经发展到离不开计算机的程度。然而，无论计算机所完成的工作多么浩繁、复杂或精细，它必须是按人们预先编制好的程序（即指令串）进行工作。那么，怎样设计计算机程序呢？通常认为，一个完整的程序设计过程应包括以下几个步骤：

(1) 明确目标和要求；
(2) 分析问题，构造模型；
(3) 算法和处理方式的确定；
(4) 处理过程的描述；
(5) 程序的编制；
(6) 程序的调试与检查。

一、明确目标和要求

在程序设计之初，一定要明确该程序所完成的功能要求以及最终结果的精度要求，这一环节虽看起来简单，却是不可忽视的。假如方案未经慎重考虑便匆忙设计，很可能造成不必要的返工。例如设计一个学生管理的数据库，事先没考虑"籍贯"这一栏，则当框架设计完，数据填入之后，再想查询"籍贯"信息，就不可能了，而且修改库框架是很麻烦的（往往会破坏已输入的全部数据信息）。又如，由于对精度考虑不够，造成输出结果距要求相差很远，只得重新构造模型或重新选择算法，浪费了人力和机时。总之，在设计之初就必须对任务要求、性质和规模有详细、确切的了解才有可能作到事半功倍。

二、分析问题，构造模型

实际工作中的问题往往十分复杂。对于数值类的科学计算问题，若已有现成的公式，当然就很简单了，只需依公式计算即可。但实际上很少遇到这种情况，限于计算机字长、内存容量等原因，一般不能直接利用理论公式，要进行适当修正，考虑量化和积累数应，使公式化为适应于计算机的计算模式。面对于非数值类的数据处理，更无现成的公式可循，完全要靠分析问题来构造模型了。实际上，许多应用领域里的问题不能直接在计算机上解决，需要经过简化而后建立起相应的数学模型，才可能上机运算。

三、算法和处理方式的确定

模型一旦建立起来，就要选定一个合适的算法。一般说来，运算结果的精确度与算法有着密切的关系，同样，运算速度的快慢也与算法有很大关系。算法选择的好，则运算过程短，运算结果精确度高；反之，则可能出现相反的效果。不同算法可能使同一问题的运算过程长短差别很大，或运算结果的精确度差别很大。而算法的选择也是极有讲究的，这一点在《计算机算法》课程中专有讲述。当算法选择好之后，还要考虑数据的存放形式，即选择合适的数据结构。数据的存放和组织形式直接影响数据存、取的速度，也影响处理的效率，因此，数据的组织形式同样是一个重要的因素。另外，还要考虑问题的全部处理过程，即明确处理方法和计算机运算的各个步骤。例如，当我们需要从大批量的数据中查找出符合某特征的数据时，虽有许多查找算法都可以达到查找的目的，但为了更有效地查找，我们必须根据这批数据的组织存放特点来选择适当的查找算法，如果还需要对数据进行计算，那么就可以进一步根据精度要求选择合适的计算模型和算法。总之，整个处理方法和处理步骤一定要明确和有效。

四、处理过程的描述

处理方法及步骤明确以后，就要对整个过程进行准确的描述。可以根据需要采用不同方法来描述同一处理过程。

例如，对于一维数组 A(1:10)各元素累加这一处理过程，我们可以用以下三种形式进行描述。

第一，用书面语言形式对过程进行描述：

输入：数组 A(1:10)

输出：累加结果 B

1. 令 B=0，　I=1；
2. 令 B=B+A(I)；
3. 如果 I<10，则 I=I+1，返回步骤 2。否则，输出 B，过程结束。

这种描述方法步骤明晰，不能编程序的人也可以据此了解一个处理过程，但对于复杂的算法，这样描述会显得很繁锁。

第二，用流程图形式对过程进行描述：

输入：数组 A(1:10)

输出：累加结果 B

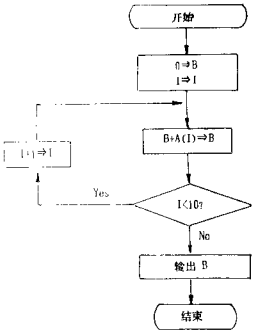
见图 1.1

用流程图描述过程是最常用的方法，它既可以展示出算法的各步骤，又可使人对算法中的循环、分支结构及其嵌套一目了然。这种方法常用于程序编制之前，通常先画出流程图的粗轮廓，再逐步细化和完善。但上述形式的流程图阅读起来不方便，尤其对较长的算法，循环、分支及各类嵌套很多时，往往使流程图中线条繁多零乱。为克服这一缺点，近年来有人提出一种结构化流程图，也称 N-S 流程图和盒图。用 N-S 流程图对累加过程进行描述有如下形式：

输入：数组 A(1:10)

输出：累加结果 B

见图 1.2



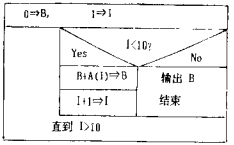


图 1.2(上)
图 1.1(左)

第三，可以用类程序设计语言的形式对处理过程进行描述，这种方法在目前许多教科书中采用，可以清楚地展示程序的结构，且由于与程序语言相近，对编程者有一定的方便之处。对于上述累加过程，可用本书所选的算法描述语言进行描述。

```
输入: 数组 A (1: 10)
输出: 累加结果 B
      B: = 0
      I: = 1
      WHILE I≤10 DO
    {
      B: = B+A (I)
      I: = I+1
    }
     OUTPUT (B)
     RETURN
```

对于算法语言中的各种符号，后面还将介绍。当然，也可以根据需要进行其它约定。总之，利用约定好的各种符号，均可以对算法进行类程序设计语言形式的描述。

以上介绍了三种对处理过程，即对算法的描述方法。这三种方法实际上只描述了程序的粗框，直正对处理进行描述的是程序本身。

要编制一个能在计算机上运行的、能得到正确结果的程序，首先要根据实际需要和环境选择一种合适的程序设计语言。程序设计语言在程序设计过程中起着重要的作用。一个良好的程序设计语言可以使得程序的结构清晰、简洁，可以正确地记述待解决的问题，同时，还可以准确地表示过程以便数据的抽象和模块化。反之，程序设计也要根据需要和具体问题选择合适的语言。首先，要求语言尽可能地接近自然语言，使用灵活、易学、易写、易记、易检

查、易修改；其次，要求语言能较好地解决某一类问题，且适应于多种计算机。对于不同类型的问题，要根据其特点，选择不同的语言，例如遇到枚举数据类型和递归问题，用FORTRAN语言就不方便了，而选择PASCAL语言则可顺利地解决问题。又如PASCAL语言和FORTRAN语言均不能直接调用系统的功能，而C语言就能灵活地解决这类问题。

仅编制一个可运行、结果正确的程序是不够的，应该努力提高程序设计的技术和改善程序的质量。一个高质量的程序，应具备以下条件：

(1) 建立正确的数学模型和确立有效的计算方法；

(2) 运行结果必须是正确的，且在精度及各方面均满足要求；

(3) 程序本身具有良好的结构，逻辑清楚，易读易懂；

(4) 程序运行时间尽可能短，同时尽可能地合理使用内存；

(5) 便于检查、修正、移植和维护。

目前，随着计算机的迅速发展和普及，程序的规模和复杂程度日益增加。因此，一定要科学合理地编制程序，否则难以进行调试和维护。

六、程序的调试和检查

程序编制完成之后，要进行检查和调试，目的是查找和改正程序中存在的错误，使程序得以在计算机上顺利运行。不仅如此，还应验证结果是否与预期的一致。程序的调试是一项复杂的工作，有时，调通一个程序所花费的时间远多于编写这个程序所花的时间。因此程序的检查与调试的重要性是不容忽视的。

为便于检查和调试，在编制程序时就要有所考虑，尽量使程序模块化和结构化，以便分块检查和调试。

程序编完之后，首先要进行静态检查。所谓静态检查，就是不在计算机上运行程序，而将自己设想成计算机，对源程序进行仔细检查，包括对笔误，语法规则和逻辑结构等方面的检查。

当静态检查结束之后，就要上机进行实际调试了，这也称为动态调试。动态调试表现在编译、装配链接和运行三方面。

编译程序是调试的好工具，可以指出程序中大部分语法错误，人们可借助于编译程序以及自己积累的经验很快查出程序中的一些错误。

装配链接程序则可以指出外部调用、存储区设置和程序模块之间接口等方面的错误。

即使编译和装配链接均通过，可以上机运行的程序也还可能有错误，这就要通过各种现象(如输入输出格式不正确，运行结果与预期不一致或死循环等等)，再进行修改和调试。

实际中可能有许多因素同时影响调试，要进行综合分析，从复杂的现象中理出头绪，将错误一个个地排除，直到试算结果与预期的完全一致，并保证必要的有效位数，还要对程序的一切特殊分支都检验之后，才能认为程序调通了。

以上简单介绍了程序设计的过程。当然，实际中并非一定严格遵守上述步骤，可根据自己的习惯和经验进行程序设计，但一定要养成良好的编程习惯和建立良好的程序设计风格。

§1.2 程序设计语言及其处理

编制程序要使用程序设计语言。为使计算机按照人们的意图工作，就必须使用计算机

所能理解、接受和执行的特殊语言。目前，通常使用的程序设计语言可大致分为低级语言和高级语言。

低级语言又分为两类。

第一类是机器语言。每一种计算机都有自己的一套指令系统，指令系统中的每一条指令又称机器指令，而机器语言就是机器指令的集合。显然，机器语言程序是用机器语言编制的程序。因为计算机可以不经任何转换地直接执行这种程序，所以运行速度较高。但因用机器语言编程序的工作量很大，既繁锁又枯燥，且由于每一计算机都由自己特定的机器语言，所以，机器语言程序无通用性。

第二类是汇编语言。汇编语言采用特定的助记符号来描述机器指令。由于计算机只认识机器语言，所以，汇编语言还需要经过汇编器翻译成机器语言。汇编语言程序与机器语言程序基本上可一一对应，只是汇编语言程序因采用助记符号而较为容易编制。它具有与机器语言同样的缺点。

高级语言不是面向机器而是面向问题的，不依赖于具体机器，具有良好的通用性。高级语言的表达方式接近于被描述的问题，又由于接近于自然语言和数学语言，从而易于为人们掌握和书写。

完成同样的任务，高级语言程序要比低级语言程序直观和简单。当然，计算机是不认识高级语言的，也不能直接执行高级语言程序，必须经过一定的处理之后才能被计算机执行。由于高级语言不依赖于具体机器，具有通用性，因而操作人员不必深入了解机器结构本身就可以根据需要来编制适用于各种机器的程序。

目前，已经有很多种高级语言了。从其应用情况来看，可大致分为三类。

第一类是基础语言。这类语言已经被广泛地使用，应用范围广，软件成品很多。这类语言包括：BASIC，FORTRAN，COBOL 和 ALGOL。

第二类是结构化语言。这类语言提供结构化的控制结构，具有很强的过程描述能力和数据结构能力。象 PASCAL，C 和 ADA 都属这类语言。

第三类是专用语言。上述两类语言都是通用语言，可用于各种领域，而专用语言是为某种特殊应用而设计的，应用范围相应窄一些。例如，APL、BLISS、FORTH、LISP 和 PROLOG 等均属这类语言。

对于一个用某种程序设计语言编制的程序，通常要经过如下的处理过程才可以在计算机上运行。

第一，要进行编辑，如同人用笔在纸上写文章一样，一种称为编辑程序的程序模块可将人们编写的程序"写"入计算机内部。计算机通过编辑程序记录下人们的程序，这些未经任何处理的程序（实际上是一些符号的集合）称为源程序。编辑程序还可以使用户很方便地修改源程序，在源程序的任何位置添加、删除、甚至将其中几行移到需要位置上，直到用户满意为止。这一过程就仿佛象人用笔修改写在纸上的文字一样容易。

第二，要经过翻译。如前所述，所有的程序均转换成机器语言的形式，才能被机器执行。这一转换是由翻译程序来完成的。翻译程序除了要完成语言间的转换外，还要进行语法、语义等方面的检查工作。通常的翻译程序有三种。

（一）汇编程序

汇编程序将汇编语言程序（源程序）翻译成机器语言程序（目标程序）。这一翻译过程称

为汇编。汇编程序通过对源程序的扫描，用一定方法进行机器码与源程序符号码的转换，同时对各种形式的错误进行检查和分析，如有错误，就以某种方式输出错误的类型及有关信息，以便用户修改。

（二）编译程序

编译程序将高级语言(源程序)翻译成机器语言程序(目标程序)，这个翻译过程称为编译。对汇编而言，通常是将一条汇编语言指令翻译成一条机器语言指令。对编译而言，则往往需要将一条高级语言的语句转换成若干条机器语言指令。高级语言的结构比汇编语言的结构复杂得多，源程序经过编译之后，若无错便生成了目标程序，再经一定处理之后，便可以运行了。运行时与源程序及编译程序无关，但若源程序作了某些修改，那么，就必须再重新进行编译。

（三）解释程序

解释程序是边扫描边翻译执行的翻译程序，解释过程不产生目标程序。解释程序将源程序一句一句读入，对每个语句进行分析和解释，有错误便随时通知用户，无错误就按照解释结果执行所要求的操作。程序的每次运行都要求源程序与解释程序的参加。

解释方式很灵活、方便，但因为是边解释边执行，所以程序运行时间长，且运行时离不开翻译程序。编译方式使程序的运行与翻译程序无关，因此运行速度要快得多，但源程序稍加修改，就要重新编译，不甚灵活。解释和编译原理在后续章节中还将详细介绍，此处就不详述了。

第三，要经过装配链接。一般说来，经汇编或编译之后生成的目标程序是不能直接运行的，目标程序可能调用一系列内部函数、外部过程和库函数或其它程序模块，这时，就需要装配链接程序将全部的目标程序块、库过程和系统库链接起来，使其成为一个可调入内存运行的程序模块，这种程序称为可执行程序。

以上是对程序处理的几个步骤，随机器类型的不同，可能此过程稍有不同。只有经上述处理之后，一个用程序设计语言编写的程序才真正成为一个可在计算机上执行的可执行模块。

§1.3 程序设计风格

仅设计和编制一个运行结果正确的程序，即使对一个未经训练的初学者来说也不是很困难的，但是，若没有培养和训练良好的程序设计风格，则很容易犯如下通病：编制的程序中错误很难发现，很难修改，即使无错也令人难以阅读和理解，甚至连设计者本人时间长了也读不懂自己编写的程序了。这样的人设计出的软件质量差，生产周期长，成本贵。所以，决不能随意地设计程序，而要严格地依照一定风格来设计程序。

程序的风格体现在以下几方面：

（一）程序的模块化、结构化

程序的模块化是指按照功能划分模块的设计方法，目的是使程序层次和作用更明确，也易于修改和调试。而程序的结构化是指按照结构化进行程序设计。这种设计方法使得程序具有合理的结构，便于保证和验证其正确性。

结构化程序设计是一种自顶向下的设计，即将设计过程分为明确的层次，复杂过程分成多个简单过程，以确保设计过程清楚，不重复，不丢失，逻辑性强。结构化程序设计必须依照

规定的结构形式来设计程序，所规定的三种结构形式是顺序结构、分支结构和循环结构。结构化的程序应仅由这三种基本结构组成。这些结构还可互相嵌套，以组成更为复杂的程序。总之，结构化程序应具有以下特点：

(1) 全部程序均由规定的三种基本结构组成，不包含其它类型的结构；

(2) 只有一个入口和一个出口；

(3) 程序的执行是有限的，无死循环；

(4) 无死语句，即每条语句均有被执行的机会。

由上可见，结构化程序实际上是由许多相对完整独立的程序段构成，每一段均只有一个入口和出口，条理清楚，层次分明，即使程序长一些，却仍然易读可靠，修改某一段一般不影响其它段。

(二) 程序内部的文档

程序内部的文档通常包括以下几方面：

(1) 必要的说明。在程序的关键部分，程序、过程、函数和段落的开头，一些公式之前等处，应加上简明扼要的注解和说明语句。

(2) 正确使用标识符。这体现在正确使用变量的类型及变量名两点上。要充分利用程序设计语言本身的特点，根据不同需要分别选用整、实、字符、枚举、集合、记录和数组等各种变量类型。例如，为方便地表示一周的七天，则可利用提供枚举变量类型的 PASCAL 语言，定义一个有七个取值的枚举变量 WEEK = (MON, TUE, WED, THU, FRI, SAT, SUN)，这样使人看起来很习惯，也很自然。又如对学生成绩统计问题，则可选择记录这种变量类型。以上两类问题利用整、实型变量表示就不直观，不易记忆。为了使程序易读，要尽量采用容易记忆且反映问题特征的变量名，如力可用 F 表示，角度可用 ALFA 或 BETA 表示，比较接近习惯的数学表示方法。诸如此类问题，要尽量采用人们熟悉的、接近自然语言或数学语言的符号来表示。此外，为了使人便于记忆，不宜使用过多的变量。

(3) 充分利用分隔符。利用逗号、空格、空白行和连接符等的作用，以便明显隔开不同语句成分。

(4) 正确采用缩进规则。在程序中各种嵌套之处采用适当的缩进表示，如内层语句较外层语句缩进若干空格，以便明确程序的层次关系。

(5) 明确程序间调用关系。在程序的首部，对该程序调用的各种函数，过程应作说明。

(三) 数据说明

数据的说明应该标准化，以便查阅、测试、调试和维护。

每一变量都应作说明，当多个变量名字在一个说明语句中时，应该按字母顺序排列这些变量。不同层次最好不要使用相同名字的变量。

(四) 语句的构造

语句应简单而直接，不要为了节省空间或所谓的提高效率而使语句排列拥挤或程序十分复杂。

(五) 效率

合理解决运行时间和内存的矛盾，不要为提高效率牺牲程序的清晰性和可读性。

§2 计算机软件概况

§2.1 什么叫计算机软件

在五十年代之前，人们均认为软件就是程序，直到七十年代才有人提出这样一个观点：软件是由程序和开发它、使用它、维护它所需要的一切文档所组成。这一观点强调了文档在软件研制中的重要性，也强调了文档是软件的有机组成部分。到 1983 年 IEEE 组织明确地给软件作了一个定义：软件是计算机程序、方法、规则相关的文档以及在计算机上运行它时所必需的数据。这一定义深刻阐述了软件的实质，也充分表明了软件与程序的区别。

计算机软件内容是很丰富的，对其进行严格分类也比较困难，仅从用途来划分，大致分为三类：

(一) 服务类软件

这类软件是面向用户，为用户服务的，包括各种语言处理程序、各种专用和通用的工程计算程序、常用库函数程序和软件包等等。

(二) 维护类软件

此类软件是面向计算机维护的，包括错误诊断和检查程序、测试程序、各种调试用软件等等。

(三) 操作管理类软件

此类软件是面向计算机操作和管理的，包括计算机管理程序、操作系统、网络通讯系统等等。

若从计算机系统角度来看，软件又可分为系统软件和应用软件。

系统软件指不是为解决科学计算问题和信息数据处理问题，而是为管理、控制和维护计算机及外设、以及提供计算机与用户界面等的软件。如操作系统，数据库管理系统，各种语言处理程序和编辑程序等。

系统软件以外的其它软件称为应用软件。应用软件主要用于解决一些实际应用问题，例如一些标准函数库、子程序包、通用软件和用于某一领域的专用软件等等，均属于应用软件。

§2.2 计算机软件与硬件的关系

一个计算机系统是由硬件系统和软件系统所构成。

自计算机问世至今日，计算机的模式仍属冯·诺依曼模式。图 1.3 显示了冯·诺依曼计算机的基本构成。

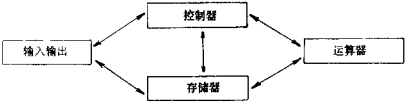


图 1.3 计算机的基本结构

由图 1.3 可知，计算机至今由五个部分构成，即控制器、运算器、存储器、输入设备和输出设备。这一结构就是计算机的基本硬件结构。

仅由硬件构成的、未配有任何软件的计算机称为裸机。若用户直接使用裸机的话，只能利用所提供的机器指令进行操作，这些当然是十分困难的。一个计算机系统要正常工作且充分发挥其硬件的各种功能，必须配备完善的软件系统。

软件与硬件的关系体现在以下三个方面。

(一) 互相依存

计算机软件随硬件技术的迅速发展而发展，而软件的不断发展与完善又促进硬件的新发展，两者密切地交织着发展，缺一不可。可以说，硬件如骨，软件如肉，骨肉相依，才有可能组成一个有机的整体。例如，象计算机的中断的实现，既要依靠硬件来实现中断屏蔽、中断响应及保留现场，又要依靠软件来完成中断的分析和处理。同样，操作系统虽说主体是一些程序模块，但许多功能需要硬件的支持。这充分说明了计算机中软硬密切的相辅相成的关系。

(二) 无严格界面

在许多情况下，计算机的某些功能既可以由硬件实现，也可以由软件来实现。因此，二者之间并无绝对严格的界面，即这一界面是浮动的。

随着硬件技术的迅速发展，许多以往依靠软件来处理的工作，都可以"硬"化成专用芯片或专用硬件单元来实现。例如，计算机图形处理中的三维图形的产生、旋转的处理工作，过去依靠软件来实现，处理的时间很长，而现在用专用芯片来完成，其速度是以往所难以想象的。

反之，也有一些过去靠硬件完成的功能，现在也可以结合软件技术实现了。例如用微指令产生时序信号来代替复杂的定时电路，既安全可靠又便于修改。

(三) 相互促进

从以上两点可看出软件和硬件之间互相影响和互相依赖的关系，同样，硬件或软件的发展都会产生推动和促进对方的作用。例如，过去由于内存的成本昂贵而限制用户程序的容量，随磁盘技术的提高，现在采用内存与外存相配合的虚拟存储技术，大大放松了对用户程序的限制，又如各种系统软件和应用软件的不断趋于完善，又促进了各种工作站、智能终端、图形终端的发展。

§2.3 软件系统的发展过程

自第一台计算机问世以来，计算机经历了电子管时代、晶体管时代、集成电路时代和大规模集成电路时代四个时代，而计算机软件自然是随着硬件而发展起来，软件发展经历了汇编语言、高级语言、操作系统、计算机网络和数据库系统四个时期。为了解计算机软件发展演变过程，我们简单回顾计算机软件在计算机发展过程中的地位和重要性的演变。

从第一代计算机发展到第二代计算机这一时期，人们对计算机硬件的研究花费很多功夫，但都不甚重视计算机软件的研究和发展，仅根据需要来编制可以运行的程序而并不考虑系统地开发软件。程序基本上是解决某个具体问题而研制，不考虑其通用性，谁编制的程序仍由谁来查错和修改，只要使程序运行起来得到结果，这个程序的任务就结束了，因此一般无须保存资料说明等文档。这是一种个体化的软件产生环境。

从第二代计算机发展到第三代计算机这一时期，由于硬件技术的变革，计算机硬件需要功能较强的软件来充分发挥效率和潜力，随之出现操作系统、数据库管理系统等软件。这个时期人们广泛使用产品软件，即许多用户可以不用自编软件，而去购买或定制软件。这是因为此时的软件无论从规模还是从复杂性来说，都使普通用户无力开发。这是一种"作坊"式的软件环境。也正是从这一时期开始出现了软件危机。

从第三代计算机发展到第四代计算机这一时期，硬件上已出现大规模和超大规模集成电路，因此计算机系统的复杂程度大大提高，计算机应用领域也大大扩展。硬件的发展速度之快，以至产生对支持软件的迫切要求，若没有可开发硬件潜力的有效软件，就不可能充分利用计算机系统。这一时期，软件的开发远不能满足新系统的需要，且维护费用占数据处理总成本的百分之七十以上，软件危机又进一步加剧了。

计算机软件由过去的无足轻重发展到今天这样在计算机系统中占如此重要的地位，然而，软件库的迅速膨胀和危机的出现，又使人们承受不了软件的资源耗费。因此，有必要对软件的工程化作进一步研究。

§3 软件工程概述

软件工程学是研究软件开发和维护的普遍原理和技术的一门工程学科，自六十年代末期开始发展至今，已成为计算机科学中一个重要分支。软件工程学的研究对象包括技术方法、工具和管理等方面。在软件研制开发过程中，若能严格遵循软件工程的方法论，便可提高软件开发的成功率，减少软件开发及维护中出现的问题。

§3.1 问题的提出

随着软件的发展演变，人们面临软件危机的困扰，软件危机体现在以下几点：

(1) 不能准确估计软件开发的成本和进度。硬件的迅速发展使其成本大大降低，但是大规模的软件开发却是十分困难的，因此，软件开发的费用及进度都难以事先准确估计。在实际中，常常出现实际成本大大高于所估成本，实际进度慢于预期进度的现象。

(2) 人们对现在软件不满意。由于软件开发人员对用户的要求未作深入了解和领会，仅凭对待解决问题的粗知来编制软件，以至使软件产品不合用户的实际需要。

(3) 可靠性差。软件产品质量不能保证可靠，而现在软件的可靠性又直接影响社会的各方面，例如一些军事设施及民用系统均直接取决于计算机系统的动作。

(4) 难以维护。实际中的许多程序难以修改和更新，更不用说很快地适用于更新后的硬件环境了。

(5) 无完整的文档。正如软件定义所明确的，软件应包括程序和文档。文档是随软件开发而产生的，可用以管理和评价开发工程的状况，也可作为交流信息、系统维护的有力工具。缺少必要的文档，会严重影响软件的开发和维护。

(6) 软件成本上升。由于开发软件是需要大量人力的，又由于硬件成本的下降，所以，软件成本在计算机系统成本中所占比例逐年上升。

(7) 供不应求。软件开发的速度远远赶不上计算机应用迅速普及和深入的趋势。

软件危机的产生与软件本身的特点及软件开发人员的弱点有关。为解决软件危机，就

要采取必要的技术、组织管理措施。软件工程正是从管理和技术两方面研究更好地开发和维护计算机软件的一门新学科。

软件工程强调生存周期方法学和各种结构分析及结构设计技术。

软件生存期指软件从设计开始到报废为止这一周期。软件生存期包括以下几个阶段：

(1) 问题定义

(2) 可行性研究

(3) 需求分析

(4) 系统设计

(5) 详细设计

(6) 编码

(7) 测试

(8) 运行和维护

通常将软件生存期又划分为几个时期。具体划分方法有许多种，例如将上述前七个阶段称为软件开发期，而将最后一个阶段称为维护期。本书为使各时期的任务更明确，采取如下划分方法：

软件定义期：包括问题定义、可行性研究和需求分析。

软件开发期：包括系统设计、详细设计、编码和测试。

软件维护期：包括运行和维护。

§3.2 软件定义期

软件定义期包括问题定义、可行性研究和需求分析。

一、问题定义

这一阶段的主要目的是确定问题的性质，工程的目标以及规模，这是软件生存期的第一阶段，应力求保证开发人员、用户及使用部门负责人在这一阶段就对问题的性质、工程规模和目标取得完全一致的看法，这对确保软件开发的成功非常重要。在问题有了明确认识之后，分析员应提交书面报告给用户及使用部门负责人审查。

二、可行性研究

可行性研究的目的是进一步研究前一阶段所定义的问题的解是否可行。在问题定义的基础上，通过复查系统的规模和目标，并研究现在正使用的系统，导出试探性的解，反复审查和修正，重新定义问题等步骤，导出新系统的高层逻辑模型，然后根据导出的系统逻辑模型设想可能的物理系统，从各方面分析物理系统的可行性，最后，推荐一个可行方针，供有关部门审批。

在描述物理系统时，常常采用系统流程图这一工具。系统流程图是用约定的图形符号以黑盒方式对系统内部各部件进行描述，它是物理数据流程图而不是程序流程图。

而描绘系统逻辑模型的最好工具则是数据流图。数据流图利用四种基本符号描绘信息在系统中流动和处理的情况。通常，数据字典和数据流图共同构成系统的逻辑模型。数据字典是关于数据的信息集合，由数据流、数据元素、数据存储和处理四项组成。没有数据字典精确定义数据流图中每个元素，数据流图就不够严谨；而没有数据流图，数据字典就不够直观，难以发挥作用。

在此阶段，往往还需对成本／效益进行分析.

最后，还要提交必要的文档.

三、需求分析

需求分析从可行性研究阶段提交的文档，尤其是数据流图出发，对目标系统提出清晰、准确、具体的要求，即明确系统必须作什么．需求分析具体的任务包括：

(1) 确定对系统的综合要求，即功能要求、性能要求、运行要求及将来可能提出的要求.

(2) 对系统的数据要求进行分析，包括数据元素的分类和规范化，描绘实体关系图，进行事务分析及数据库模型的建立.

(3) 推导出系统的逻辑模型，在前面分析的基础上可以导出系统的详细逻辑模型.

(4) 修正开发计划和建立模型系统．根据前面的分析可以较确切地估计系统的成本和进度，对以前制定的开发计划进行进一步的修正．为了能够显示系统的主要功能，往往要建立一个模型系统.

需求分析的进行首先从数据流图着手，在沿数据流图回溯的过程中，更多的数据元素被划分出来，更多的算法被搞清楚，将得到的有关数据元素的信息记录在数据字典中，而将对算法的简明描述记录在输入／处理／输出(即 IPO)图中，补充的数据流、数据存储和处理添加到数据流图的适当位置上，然后提交用户复查以补遗．经过反复地进行上述分析之后，分析员对系统的数据和功能都有了深入了解，此时可通过功能的分解将数据流图细化，即将数据流图中较复杂的处理功能分解成多个子功能，这些较低层的子功能又重新组成一张数据流图，当然还包括所产生的数据存储和处理．细化过程中得到的数据信息记录在数据字典中，面导致新的或精细化的算法描述也需要记下来．经过以上分析，就可以修正开发计划，然后书写必要的文档．文档的内容包括：系统的功能说明，主要由数据流图和 IPO 图(或其它形式的算法描述记录)组成，描述了目标系统的概貌及对系统的综合要求；系统的数据要求，主要由数据字典和描述数据结构的层次方框图或 warnier 图组成；用户系统描述，即初步的用户手册，包括对系统功能和性能的扼要描述、使用方法和步骤等.

在转入下一阶段之前，还须要进行审查和复查，通过之后开发工程才可进行.

§3.3 软件开发期

软件开发期包括系统设计、详细设计、编码和测试.

一、系统设计

这一阶段的任务是划分出构成系统的各物理元素，如程序、文件、数据库、人工过程和文档等，还应设计出软件的结构，即确定系统中每个程序所包括的模块以及这些模块间的关系.

设计过程通常有如下几个步骤：

(1) 提出可选择的方案．应该设想各种可能实现的方案，再进行充分的比较和分析．可以从上一阶段的数据流图出发，设想数据流图中的处理分组的各种可能的方法，摈弃技术上行不通的分组法，剩下的分组方法代表可能的实现策略，且可启示供选择的物理系统.

(2) 选择合理方案．从上面提供的方案中选取若干合理的方案，至少选用低、中、高三种成本的方案．选择时需要考虑系统的工程规模和目标．对每一种方案，应准备系统流程

图、组成系统的物理元素清单、成本／效益分析和实现系统的进度计划四份资料。

(3) 推荐最佳方案。经过对几个合理方案的综合分析和对比，推荐出一个最佳的方案，并且制定出详细的实现计划。最后要由用户和有关专家对方案进行审查，若认为能满足用户需要且有实现的条件，再交使用部门负责人审批，通过之后就进入了系统结构的设计了。

(4) 功能分解。通常，大型程序设计包括结构设计和过程设计两部分。结构设计是本阶段的任务，过程设计是下一阶段的任务。为了确定软件的结构，首先应从实现的角度将复杂的功能进一步分解，应结合算法分析数据流图中的每个处理，若某一处理功能过于复杂，必须将其分解成一系列简单功能。功能分解使数据流图进一步细化，同时也应用 IPO 图或其它方式记录下细化后的算法描述。

(5) 设计软件结构。程序中每一个模块应完成一个子功能，所有模块应按照从上向下调用的原则构成层次系统。最高层模块调用其下一层模块可实现程序的完整的功能，而其它层次的模块调用其下层模块可完成程序的某一子功能，最底层模块实现某一具体功能。软件的层次结构可用层次图或结构图来描绘。若系统还需要建立数据库，那么，还应设计数据库的模式、子模式等等。

(6) 制定测试计划。尽早为将来的测试作好准备，所以，在这一阶段就要考虑测试的问题。

(7) 提交文档。用正式文档来记录设计的结果，包括系统说明、用户手册、测试计划、详细的实现计划及数据库设计结果。

此阶段的最后结果要进行严格的技术审查，然后由使用部门负责人从管理的角度来复审。

二、详细设计

详细设计的任务是怎样具体地实现目标系统，即应对系统作出精确的描述，以便在编码阶段可直接将这一描述用程序设计语言编制成程序。除了应该保证程序的可靠性之外，此阶段最重要的目标就是保证将来的程序易读、易理解、易测试，易修改和易维护。因此，结构程序设计技术就成为实现上述目标的基本保证，也是详细设计的逻辑基础。

前面已提到结构化程序设计法是采用自顶向下、逐步求精的设计方法和单入口单出口的控制结构。

在软件开发期采用结构化程序设计技术有以下几点好处：

(1) 自顶向下逐步求精方法是解决复杂问题的普遍规律，可显著提高软件开发工程的成功率和生产率。

(2) 程序开发采用先全局后局部、先整体后细节、先抽象后具体的过程，使程序有清晰的层次结构，易读易理解。

(3) 使用单入口单出口的控制结构，使程序的静态与动态执行保持一致，同时保证程序易读、易理解，也容易保证其正确性。

(4) 按照规定的逻辑模式设计程序的控制结构，保证源程序清晰流畅，易读易懂又易测试。

(5) 修正和重新设计时代码变动最少。

(6) 逻辑结构清晰有利于程序正确性证明。

详细设计阶段常用的设计工具有程序流程图、N-S 流程图等。

三、编码

编码是对软件的系统设计和详细设计的结果翻译成用某种程序设计语言书写的程序。虽然程序的质量基本上由设计的质量决定，但是编码中也有几个因素对程序质量有相当大的影响。

首先，要选择适当的程序设计语言。从实际应用的角度来看，高级语言较汇编有很多优点，因此，除非在非用不可的场合，一般不要采用汇编语言编程。至于选择何种高级语言，可从实用观点来考虑以下几点：

(1) 选择用户熟悉的语言，便于今后的系统维护。另外，在不产生矛盾的条件下，最好选择程序员已熟悉的语言，以便提高编程效率。

(2) 根据目标系统将来的运行环境进行选择，例如根据所提供的编译程序选择与之相适应的语言。

(3) 根据工程规模选择。若规模很大且有必要的话，可选择专用语言，甚至为之设计一种语言。

(4) 根据目标系统的应用领域选择合适的语言。若目标系统将来要在不同的硬件环境中运行，那么就要考虑移值性好的语言。

其次，在编码过程中，要使程序内部有良好的文档资料，规律的数据格式说明，简单清晰的语句构造和输入／输出格式，这些都可大大提高程序的可读性，而且可改进程序的可维护性。

另外，要充分利用已有的软件工具来辅助编码，以提高效率和减少差错。

四、测试

在目前，软件测试仍然是保证软件可靠性的主要手段。软件测试是软件开发过程中最艰巨最繁重的工作。测试的目的是查找程序中的错误，但绝不能证明程序中无错。

测试常用黑盒法和白盒法。黑盒测试法将程序看成是一个黑盒子，不考虑其内部结构和处理过程，只在程序接口进行测试，检查程序的功能是否满足需要。白盒测试法将程序看成被装在透明的白盒子里，即完全了解程序的内部结构和处理过程，按照程序内部的逻辑进行测试。

软件测试通常要经过以下几步：

(1) 模块测试。在具有良好层次结构的软件系统中，每一模块完成了一个清晰定义的功能。因此，可将每个子模块作为一个单独实体来测试，也较容易设计测试方案。模块测试的目的是保证每个模块可作为一个单元来正确运行。所以，也称为单元测试。

(2) 子系统测试。将经过模块测试的模块形成一个子系统进行测试，即着重测试模块的接口。

(3) 系统测试。将经过测试的子系统装配成一个完整的系统进行测试，不仅应查找和发现设计与编码中的错误，还应验证系统确实可提供要求说明书中指定的功能，而且系统的动态性能也要符合要求。子系统测试和系统测试有检测和组装的含义，所以也称为集成测试。

(4) 验收测试。将系统作为单一实体进行测试。这是在用户参与下进行的，可能主要使用实际数据，目的是验证系统是否确实满足用户需要。

(5) 平行运行。验收之后，一般将新开发的系统和将被取代的旧系统同时运行，以比较

两系统的处理结果.

设计测试方案是测试阶段的关键技术问题,基本目标是选用最少数的高效测试数据,作到尽可能地发现软件中的问题. 设计时常采用黑盒法设计基本测试方案,再用白盒法补充一些必要的测试方案.

在测试过程中发现的错误应及时纠正,这是调试的任务,调试过程中最困难的任务就是确定出故障的位置.

现有的调试技术主要有三种. 第一种是输出存储器的内容,即输出以某种代码表示的存储单元的内容. 第二种是打印语句,可在程序的各个部分插入标准打印语句以便显示主程序的动态行为. 第三种是利用自动工具,即利用程序设计语言自身的调试功能或专用软件工具分析程序的动态行为.

调试的关键并非上述技术,而是推断错误原因的方法,常用方法有:

(1) 试探法. 根据现象,猜测错误的大概位置.

(2) 回溯法. 人工沿程序控制流往回追踪源代码,直至找到错误源或确定出故障范围为止. 也可以采用正向追踪,即用输出语句打印一系列中间结果,以确定出现错误的起始位置.

(3) 对分查找法. 设置若干关键点,将程序分段隔离查找,不断缩小故障范围.

(4) 归纳法. 即从个别推断到全体的方法. 从线索(错误的表象),通过分析这些线索之间的关系而找出故障.

(5) 演译法. 先列出所有可能的原因或假设,然后逐一排除列举出的原因,最后,证明剩下的原因确实是错误的根据.

测试和调试是软件测试阶段的两个关系极密切的过程,通常交替地进行

§3.4 软件维护期

维护是软件生存周期的最后一个阶段,也是持续时间最长、付出代价最大的阶段,而软件工程学的目的就在于提高软件的维护性,同时设法降低维护的代价.

软件维护通常有四类:为纠正在使用中出现的错误而进行的改正性维护;为适应环境变化而进行的适应性维护;为改进原有软件而进行的完善性维护;还有为将来的可维护和可靠而进行的预防性维护.

软件的可理解性、可测试性和可修改性这几个因素影响和决定软件的可维护性. 软件生存周期的各阶段都与软件可维护性有关. 良好的设计、完善的文档资料、以及一系列严格的复审和测试,会使错误一旦出现就较为容易诊断和纠正,当用户有所要求或外部环境有变化时软件能比较容易适应,并且能够减少维护的付作用. 因此,在软件生存周期的各个阶段都必须充分考虑维护的问题,并且为维护作好准备.

软件维护不仅包括程序代码的维护,还包括文档的维护,即文档也是影响软件可维护性的因素,甚至比可执行的程序代码更重要. 文档可以分为用户文档和系统文档两类. 不论是那类文档都必须和程序代码同时维护,只有和程序代码完全一致的文档才有意义和价值.

目前已有许多软件工具能帮助建立文档,不仅可以提高书写文档的效率和质量,还有助于文档的及时维护.

§4 算法描述语言

为了便于描述算法，本书约定一种算法描述语言，下面简单介绍这种算法描述语言的使用。

一、标识符

与一般程序设计语言相类似，标识符由字母开头的字母数字串构成。

二、运算符

算术运算符有：+，-，*，/

关系运算符有：=，≠，<，>，<=，>=。

逻辑运算符有：AND，OR，NOT

三、赋值

赋值是用如下形式表示：

a:=b

其中 a 是变量或数组元素，b 是表达式或常数。

四、控制结构

1. 选择分支结构

选择分支结构有如下形式：

```
IF  <条件>      THEN  语句
                ELSE  语句
```

对于复合语句，可用如下形式：

```
IF  <条件>      THEN
  {
    语句 1
    语句 2
      .
      .
      .
    语句 n
  }
ELSE
  {
    语句 1
    语句 2
      .
      .
      .
    语句 m
  }
```

2. 循环结构

循环结构有两种:

(1) FOR 循环有如下形式:

FOR 循环变量=初值 TO 终值 [BY 步长] DO 语句

同样,若遇到复合语句,用花括号括起来.

(2) WHILE 循环有如下形式:

WHILE <条件> DO 语句

同样,若遇到复合语句,用花括号括起来.

五、其它

为明确表示算法的功能,在算法的首部应写明必要的参数以及输入变量与输出变量的名称.

用 INPUT (变量表列)表示输入过程.

用 OUTPUT (变量表列)表示输出过程.

用 RETURN 表示整个算法处理过程的结束.

读者可结合书中例题加深对这种算法描述语言的理解.

习 题 一

1. 什么是结构化程序?与非结构化程序相比,有什么特点?
2. 100 元钱要买 100 只鸡,小鸡每只 0.5 元,公鸡每只 2 元,母鸡每只 3 元.请列举不同方案,从中选出最佳算法.并说明理由.
3. 用普通流程图,N-S 流程图和算法描述语言描述上述的算法,并进行比较.
4. 比较三种翻译过程,各有什么特点.
5. 什么是软件?软件与程序的区别是什么?软件与硬件有何种关系?
6. 什么是软件生存周期?它包括哪几个阶段?