软 件 设 计 方 法

上海快必达软件出版发行公司
1985.10

上海快必達軟件出版发行公司

SHANGHAI COMPUTER SOFTWARE CORP

告 用 户 读 者

我公司是由上海市计算技术研究所和上海外文图书公司（原上海外文书店）联合组成的单位。采用“技贸结合”的形式，将计算机软件产品作为“特种书刊读物”，纳入图书发行系统，是一家计算机软件出版发行的专业公司。

业 务 介 绍

一、出版发行各类通用或专用的软件及软件包；
二、办理进口原版软件的统一征订及发行；
三、引进移植国外先进的软件包；
四、为全国各单位及个人代销有推广价值的软件包；
五、为软件产品用户提供硬件配套、技术培训和维护服务；
六、开发、出版、发行供出口的具有中国特色的软件包；
七、编辑、翻译、出版、发行软件技术资料和书籍；
八、向单位或个人征集、特约开发各类有实用价值的软件或软件包。

本公司定期编印进口原版和国内版软件目录，办理征订。竭诚欢迎用户读者光临赐教。

地　　址：上海福州路 390 号

电　　话：223200（总机）

电报挂号：0831

银行帐号：上海分行营业部 4465280

目　录

第1章 软　件

1.1 计算机和软件

众所周知，计算机技术大致可以分为硬件(hardware)和软件(software)二种。当然，它们不是各自独立的，而是密切相关的，只有将它们两者融为一体，才能达到某种目的。

硬件系指计算机的(人眼看得见的)机械部分，而软件的概念则较难理解。有时软件，也译为计算机的"应用技术"，但这种译法的含义模糊，容易被人误解。

为了理解软件，我们以汽车为例作个适当的比喻。汽车的机械是类似计算机的硬件，人眼能看到。然而，只要有这种硬件，就能使汽车"开动"吗？不，它还必须具备运行操作、交通规则、排除故障等方面的知识。这些知识相当于汽车的"应用技术"，犹如计算机软件。这些软件(知识或信息)人眼是看不到的。当然，这些知识通常以"交通规则集"或"驾驶教练本"等书的形式出现。这是人眼能看到的东西。但是，人眼能看到的只是信息的介质，而不是信息本身。例如，假定把在电视讲座中教授汽车的应用技术摄录于录象带里，就有可能把与书本相同的信息存在于别的介质上。

由于软件是人眼看不见的信息，因而通常把软件看作是独立于信息的介质。在计算机中往往采用磁带作为信息的介质。买1卷磁带约3000日元，但记录在磁带中的软件往往要几亿日元。

计算机的一个重要特征是，软件所占的比重比其他机械(例如汽车)要大得多。也就是说，如果以开发硬件的技术为主，若对软件的应用技术抱以无足轻重、可有可无的观点是错误的。"应用技术"这个词之所以容易被人误解，就是由于这个理由所引起的。计算机所以不同于其他机械，正是因为计算机具有通用性。亦即，计算机只要执行一定的程序，那么无论什么工作都能进行*。如对于桥的强度计算、工资计算、银行存款出纳的联机处理、许多交通信号机的系统控制、发生地震时高楼大厦摇动的模拟等，在原理上都可以采用相同的计算机来进行。因此，计算机不像其他汽车和电气扫除机之类的单功能(单目的)机械，而是所有的计算和数据处理(只要编制程序)都能进行。在这个意义上说，计算机硬件与其说是通用，还不如说是尚未决定使用目的的机械，因使用目的是由程序给出的。这里所说的程序(program)是指为了让计算机进行工作而描述该工作顺序的指令集合。只有给出这种指令的集合，计算机才能进行某种工作。编出什么程序，就进行什么工作。这就是计算机所具有"通用性"的含义，从而表明了一条至理：程序(软件)的重要性不亚于计算机硬件。

通常，程序必须采用由计算机能解读的指令来书写。我们将书写(即设计和编制)这种程序的过程称为程序设计(programming)。把为编写计算机能够理解的程序而规定的语言体系称为程序设计语言(programming language)。计算机能直接理解的语言称为机器

* 计算机虽是通用的，但并非是万能的。固然编写了程序，任何工作都能进行，但实际上有的工作是无法编写程序的。

语言(machine language)，这随计算机的种类而异。此外，还有许多通过翻译使得计算机能够间接理解的程序设计语言。这将在1.3节中叙述。

如上所述，程序是依赖于各种计算机硬件和程序设计语言来进行描述的。也就是说，它用特定的程序设计语言编写，由特定的计算机来执行。一最理想的是编写具有互换性的程序，无论哪种计算机都能执行，但要完美地实现这个理想是困难的。如果剔除程序中依赖于计算机和程序设计语言的部分，纯粹地让计算机去进行工作，这样的处理顺序叫做算法(algorithm)。程序需要用特定的计算机来直接执行，算法可以不用计算机来执行，而表示为抽象化了的处理顺序。

一提起"软件"，人们往往会理解是程序，其实有时也包括程序的处理对象——数据(data)、程序数据的使用方法和描述内部结构的文档(document)。因为在某种意义上，这些都是让计算机硬件进行工作所必需的信息。

1.2 计算机硬件

A. 硬件结构

为了理解计算机硬件的功能，我们首先用图1.1所示的人利用台式电子计算机来计算 x^2+y^2 时的顺序。

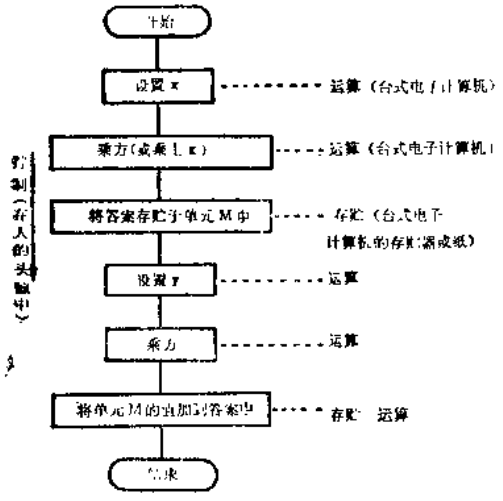

图 1.1 x^2+y^2 计算的顺序

如图右边所示的那样，必要的各种操作大致可以分为运算和存贮二种功能。"设置 x"操作是运算，这也许给人以一种奇异的感觉。但可以认为，在计算机中，它也是"置数"的一种运算。实际上，有的计算机也用"把 x 加到0"的形式来实现"设置 x"。如果台式电子计算机具有存贮器，则 x^2 的值就存贮在存贮器中。如果没有存贮器，或将它已经全部用于其他目的，那末就记录在外部的纸上。前者相当于计算机的内存贮器(主存贮器)，后者则相当于计算机的外存贮器(辅助存贮器)。此外，还必须把再次取出这个已被记录的数值的操作视为与记录操作相对的存贮功能的一部分。

另外，还需要有"控制"功能，它能正确无误地依次逐个执行图中的一连串顺序。用台式电子计算机进行的计算(只要不是程序台式电子计算机)是在人的头脑中进行的。

因此，为了不借助人来进行与上述相同的计算，计算机硬件必须具有图1.2所示的结构。

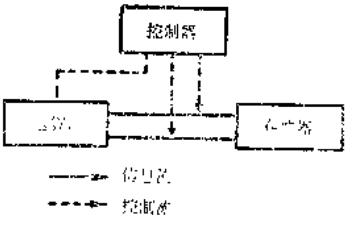

图 1.2 计算机硬件结构

但是，在这种结构的计算机中，不能与人进行交换信息。它只具有将 x 和 y 的值输入到计算机，或输出计算答案(结果)的功能。在台式电子计算机中，数字的按钮是输入装置，数值的显示窗是输出装置。

假若把输入输出包括在内再次表示 x^2+y^2 的计算顺序，则如图 1.3 所示。因此，在图 1.2 中如果附加输入输出设备的话，就能获得图 1.4 这样结构的计算机硬件。这里，标有星号(*)的数据流将在后面叙述。

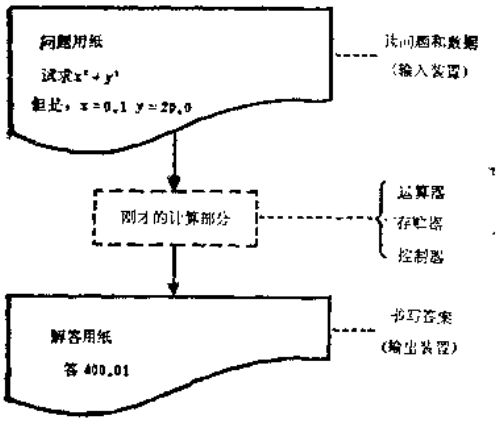

图 1.3 包含输入输出的计算顺序

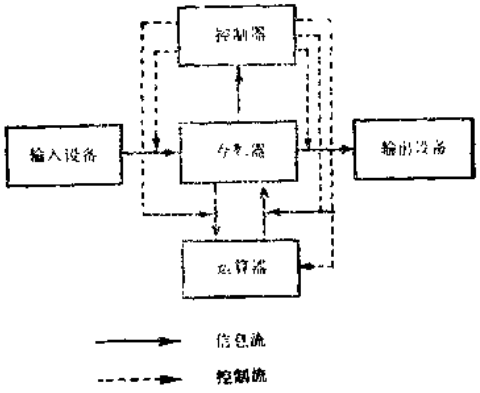

图 1.4 包含输入输出的计算机硬件结构

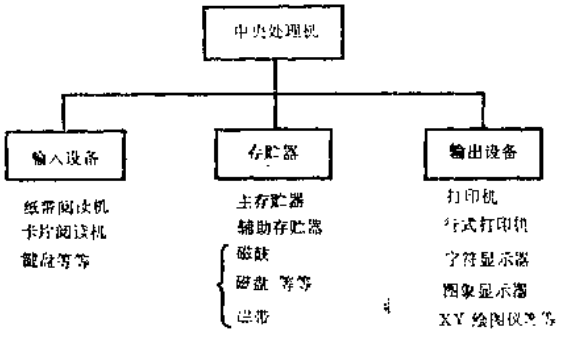

图 1.5 实际的计算机硬件系统

通常，由控制器和运算器构成一个整体，称为中央处理机(central processing unit，缩写为 CPU)。图 1.4 是概念性设备的分类，实际上人眼能看到的"物体"如图 1.5 所示。

存贮器可以分为与中央处理机之间的数据传送速度快的主存贮器 (main memory) 和传送速度慢的辅助存贮器。主存贮器的存贮空间以字(word) 为单位，并分别附加了地址 (address)* (见图 1.6)。为了简单起见，可以考虑在一个字里能存放一个数据(或命令——后述)。字由若干个位(bit)构成，位是信息的最小单位，它能表示二个值0或1中的任意一个。

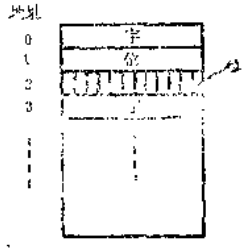

图 1.6 主存贮器的结构

B. 程序方式

因为计算机具有计算速度快的优点，所以在计算中因借助人的判断而等待其回答的

* 还有其他的方式，本书将作简单的介绍。

情况是很少见的*。例如，即使是极为普通的计算机，在人考虑的一秒钟中，计算机却已进行了上百万次的加法。因此，为了在计算中不让人介入，那末对于计算和处理中可引起的所有情况都事先要给出完整的指令集合。这种指令的集合被称为程序(前一节中已作了叙述)，而这种指令的给出方法则称为程序方式。

例1.1 考虑一下"当给定系数 a，b，c 的值时，求出二次方程式 $ax^2+bx+c=0$ 的2个根 x_1，x_2 后，再进行打印"的程序。通常，在计算机中，对实数和复数进行区别后再处理，所以在复数场合下，即判别式 b^2-4ac 为负时，则如何处理？为了编制计算机程序，必须事先将计算机在所有的场合下应该做些什么编写在程序上，而不是查找 b^2-4ac 是正、0或负后，人再决定如何做。例如，有下面的一个方式：

若 $b^2-4ac\geqslant 0$，则打印出 2 个实根 x_1，x_2 之值。

若 $b^2-4ac<0$，则用这种形式打印出 2 个复根 $r\pm ic$ 之值。

[练习1.1] 由上可知，$ax^2+bx+c=0$ 是真正二次方程式(即 $a\neq 0$)。对于 a，b，c 的任意值，为了求出 x 值后，进行打印(在不决定和不可能的情况下，显示其意思)，在怎样的情况下只要进行怎样的处理就行了。

[练习1.2] 假若在书店和图书馆里列有名为"程序方式的××"或"程序学习的××"的书，则请看一看其内容。在哪种意义上，它是程序方式。

C. 程序存贮方式

目前几乎所有的计算机都采用了程序存贮(stored program)方式。它是由冯诺以曼(von Neumann)先生早在 1946 年提出的，亦称为冯诺以曼型计算机。在这种方式中，程序与数据一样，也从输入设备中输入，并存贮在存贮器中。若要执行该程序，便将程序中的各条命令依次逐条读出到控制器中，然后将它解读后，使运算器进行必要的运算(参见图 1.7)。在图 1.4 中的横线箭头是指信息流。因为是逐一依次执行命令的，所以也把这种计算机称为逐次控制型(sequentially controlled)。可以说，目前的计算机大体上均以逐次控制型为基础的，但也有例外。确切地说，我们这里称谓的计算机也许都应该称为逐次控制型程序存贮方式数字电子计算机。

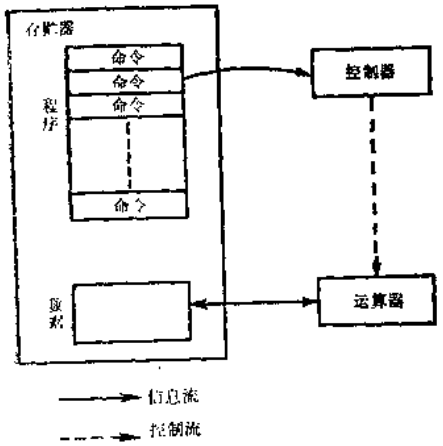

图 1.7 程序存贮方式

1.3 程序设计语言和翻译程序

程序设计语言的种类有许多。计算机能直接理解的，即中央处理机能解读和执行的程序设计语言称为机器语言，这已经在前一节中作过介绍。因为机器语言是数字的罗列，所以使人难于理解，且易于写错。将机器语言符号化、使人易于理解的程序设计语言称为汇编语言(assembly language)，汇编语言通常与机器语言一一对应。更进一步说，从与机

* 为了提高计算机的效率，有的计算机使用方法借助人来进行判断，称为分时系统(time-sharing system 缩写为TSS)或会话型系统(interactive system)。

器语言的对应中分离开来，从人的角度来看工作的顺序，然后扼要地进行描述的程序设计语言，称为编译语言(compiler language)或高级语言(high level laguage)。虽然汇编语言与机器语言一样要依赖于不同类型的计算机，但高级语言的设计依赖于计算机的部分少，并且在努力地实现标准化。典型的高级语言有FORTRAN,COBOL,PL/I, PASACAL, LISP*, SNOBOL, BASIC*等。图1.8表示某计算机的机器语言、汇编语言、编译语言(FORTRAN)的例子。该例描述了在某数M中加上某数N，然后减去1，最后将结果送入K中的计算。

```
016701          MOV  M, R1        K=M+N-1
000026          ADD  N, R1
066701          DEC  R1
000027          MOV  R1, K
005301
010167
000024
```

(a) 机器语言　　(b) 汇编语言　　(c) 编译语言

图 1.8　机器语言、汇编语言、编译语言(FORTRAN)

用汇编语言和编译语言编写的程序，计算机是不能执行的。计算机能直接解释和执行的仅是机器语言程序。因此，用汇编语言或编译语言编写的程序必须"翻译"成机器语言。如果这种翻译靠人来进行，那么这就与最初使用令人麻烦的机器语言来编写程序没有什么两样了。因此，为了发挥计算机的通用性，必须让计算机本身去执行这种翻译**。由上可知，为了让计算机来执行这种翻译，就必须有执行翻译的指令的集合——翻译程序。将汇编语言翻译成机器语言的程序称为汇编程序(assembler)；将编译语言翻译成机器语言的程序称为编译程序(compiler)，也可将它们总称为语言处理程序(language processor)，或简称为处理程序。例如，用FORTRAN编写的程序(源程序)要使用FORTRAN编译程序翻译成机器语言程序，然后再执行***(见图1.9)。与各种语言的编译程序相对应，在执行由编译程序翻译好的机器语言程序时，大多数需要借助执行机器语言程序的程序，称为执行该语言时的处理程序。

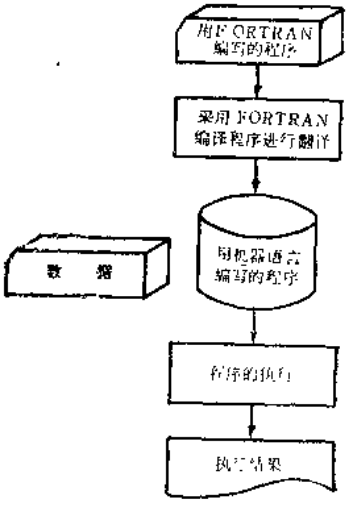

图 1.9　FORTRAN 程序的翻译和执行

1.4 程序设计的过程

程序设计通常是按照图1.10所示的过程来进行的。但是，在图中不仅包括狭义的程

* LISP和BASIC多数是采用翻译方式，而不是采用编译方式。两者的区别可参阅有关编译程序方面的教科书。

** 编译程序语言将文法和含义设计成一种意义，所以把编译语言翻译成机器语言远比将英文翻译成日文容易。虽说能进行人工语言即程序设计语言间的翻译，但希望不要误解，在同等程度的容易性方面能进行我们日常使用的自然语言之间的翻译。

*** 实际上，在翻译和执行机器语言程序之间，往往还有另一种称为"连接编辑"的处理。其理由和意义，这里不再赘述。

序设计，而且也包括成为其前阶段的系统分析和设计的过程，下面我们依次说明。

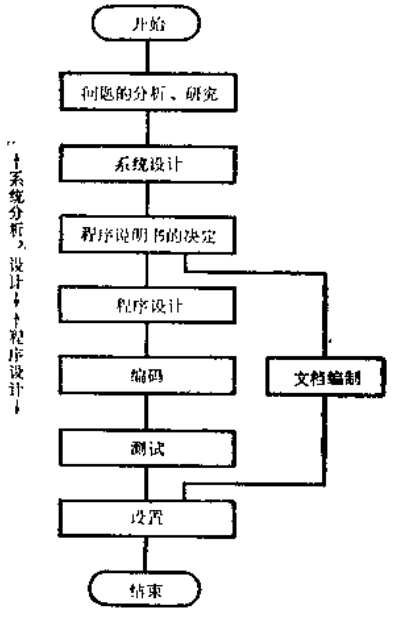

图 1.10 程序设计的流程

(1) 问题的分析和研究　在这个阶段中，研究已给出(应解决)的问题，并考虑采用哪些解决方案。当然，考虑的解决方案不一定仅使用计算机。在世界上，有许多问题即使不使用计算机也能很顺利地进行。此外，还必须明确究竟是什么问题，也就是解决的方案是什么，这称之为需求定义 (requirement analysis)。

(2) 系统设计　对于在(1)中考虑的最佳解决方案，要使细节部分的构思具体化。也就是说，决定对哪些数据流用人工、计算机或其他机械来进行哪些处理。

(3) 程序说明书的规定　根据上述(2)来决定应编写满足哪些要求的程序。

上面是系统分析和设计的工作，以后则是狭义的程序设计。亦即，在决定编制"做什么"程序和决定"应该如何来编制程序"方面，要区分这二种工作。它们两者的连接点就是程序的说明书，即描述应编制"做什么"程序的文档。此外，还包括系统分析和设计的过程，在广义上称为程序设计。

(4) 程序设计　考虑编制"怎样地"进行在说明中所描述的工作程序，并决定程序和使用数据的内部结构。

(5) 编码　用程序设计语言编写程序。

(6) 测试　编制程序的测试一般分成几个阶段来进行*。一个阶段是检查是否正确地满足(3)中规定的说明书的程序。如有错误，则进行修改，这种作业也称为调试(除虫——debugging)。它出自将程序的错误俗称为虫(bug)。

总之，在上面的阶段中，要对程序或包括它在内的系统是否成了(1)中给出的解决问题的方案进行测试。尽管编制了能满足说明书规定的程序，但也经常会发生解决方案并不妥当的情况，这是由于(2)的系统设计和(3)的程序说明书的规定不合适所引起的，在这种情况下，问题就从狭义的程序设计范围再次返回到系统分析和设计的阶段。

(7) 文档的编制　编制有关的系统设计，程序说明书及其内部结构、测试方法与成绩以及程序的使用方法等说明书。这种作业与上述(2)～(6)的作业将并行进行。

(8) 设置　将程序或包括程序的系统(即(2)中所述的解决方案)存入所需要的处所，并进行编排。

本书中使用的范围规定为狭义的程序设计部分，即从(4)至(6)的前半部分和(7)的一部分。也就是说，假定应编制的程序说明书已基本给出，而且仅对是否满足此说明书进行测试。可是，在许多场合，所给出的说明书一般具有模糊性，所以在程序说明书的规定和(狭义的)程序设计之间通常不能明确地划线，这一点将在第5章中说明。

* 可以分为独立测试、连接测试、系统测试、设置测试等，这里大致分为二种。

1.5 软件的重要性

本节将叙述为什么软件比硬件更为重要和为什么读者必须深入地学习软件的编制方法。

A. 软件的规模

目前，用户能直接使用计算机硬件几乎是没有的。正如图1.11所示那样，必须要给裸机的硬件的周围加上易于使用的软件的外衣，才能让用户使用一台比较完整的计算机。这种当作"外衣"的软件称为系统程序(system program)或操作系统 (operating system)。例如，在1.3中叙述的编译程序和汇编程序等的翻译程序就是操作系统的一部分。

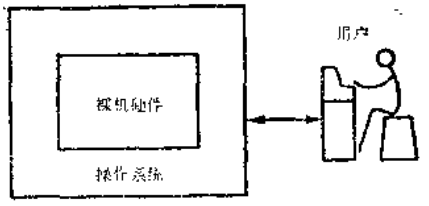

图 1.11 操作系统的作用

目前，在操作系统中最大的可能是 IBM370 系列，以及随后相继推出的机种：OS/MVS 的操作系统。据说这种操作系统大小约500万步(排列500万条命令)。至于该操作系统究竟多么庞杂，只需你自己编制了十几行程序而感到麻烦时，就会有切身体会了。

OS/360 是 OS/MVS 的祖先，其规模为一百几十万步，花费了4年时间(1963年～1966年)，工作量总计有5000人年，耗资达2亿美元。当初，预计这种操作系统于1965年完成，但后来推迟了一年才完成，费用比原先估计的多几倍。勃罗克斯(F.B.Brooks Jr.)先生曾经在他的书中较详细地叙述了这一期间的情况。

确切地说，操作系统不是一种程序，而是相关的一系列的程序群。诸如银行联机户头、日本国铁的绿色窗口(日本铁路的购票窗口。——译者)、新闻出版的自动编辑和排版等大规模联机系统的应用都拥有几十万步以上的程序群。从单一的程序来看，超过十万行的程序目前是不罕见的。

B. 编制大型程序的难度

通常，编制10,000步的程序是否要花1,000步程序的10倍时间或人力呢？如果软件的规模为10倍，那么，编制程序的困难就超过10倍，其理由至少可列举如下二点：

(1) 小组作业的生产性　大型程序靠一个人是无法编制的。假定有的程序需要10人年的工作量，若由一个人来编制它，那末10年后完成的话，几乎是无意义的。因为对软件来说，时间性至关重要，假若上面的程序由十个人合作编制，则一年后就能完成了。

让我们考虑如下模型。假设有 n 人一起合作编制一个程序，若设一个人的能力为 a，为了协作，在任意二个人之间联系和调整，其所需的二人工作量为 $2pa(p<1)$ 则整个 n 人小组的能力为：

$$A = na - n(n-1)pa$$

这个式子是与 n 有关的二次方程式，如图1.12所示，成为向上凸的抛物线。因此，若增加组的人数 n，开始增加了能力 A，到某个地方成为最大，但以后却相应地减少，最后成为负值。例如，设 $p=0.05$，则 $n=10$，11，A 取最大值 $5.5a$，当 $n\geqslant 21$ 时，则 $A\leqslant 0$！也就是说，在这个假设下，小组无论增加多少人数，当超过5.5人的能力就不能进行工作。

如果过多地增加人数，则反而会降低效率。其原因在于，如果在许多人中间分担工作，则光是在各人之间的联系势必要花费时间，因而不能进行有效的工作。

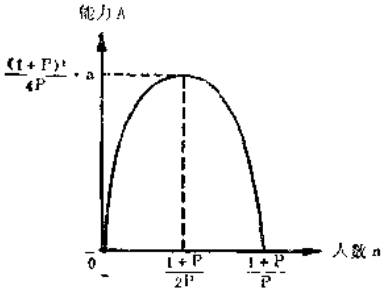

图 1.12 小组的能力A和人数与n的关系

上面的模型太简单。实际上，当许多人从事一项工作时，通常采用金字塔形的组织来减少各人之间的相互干扰。尽管如此，上面的模型毕竟明显地表示了进行分工编制大型程序的难度的起因。

威伯格(G.M.Weinberg)先生根据经验叙述了三个程序员的小组只能进行二个人的工作。当各由三个人组成的三个小组一起合作进行工作时，只能进行二个小组(即4个人)的工作。

[练习1.3] 即使采用金字塔形的组织，增加该小组的人数 n 和小组的能力 A，也只能按小于与 n 成比例的速度增加，请表示之。

[练习1.4] 上面的简单模型还可用于其他方面。你是否陈述一下假定你成了日本电报电话公司的总裁，要是电话机越普及，电话费就越提高的理由。

(2) 软件的可靠性 如果某种系统由 n 个元素构成，设各元素的出错概率为 P，并且这种概率是相互独立的，则整个系统正常工作的概率为

$$P(n)=(1-p)^n$$

若对固定的 p，增加元素个数 n，则 $P(n)$ 逐渐变小，系统正常工作的概率几乎成为0。例如，假设在10万步程序中，将程序进行测试后留下来的出错概率(残存出错率)每步为0.0001，则可得

$$P(100,000)=(1-0.0001)^{100000}=0.00005$$

这样，几乎无法指望程序能正确运行。

基于上述二个理由(及其他理由)，如果是10000步左右的程序，那么平均一个人每天可以编制6～25步，而如果是200万步的程序，那么就相差每人每天编制1步左右的速度。但是，这里的程序生产性的值不光是编写程序，而是相对于程序说明书的规定、设计、编码、测试、文档编制等程序设计的所有工作过程的值。

C. 软件的费用

图1.13表示在采用计算机进行工作的总费用中，硬件与软件费用所占比例的变化。由图可知，在1955年的时候，硬件费用超过80%，软件费用不到20%，而如今却倒过来了，硬件费用为20%，软件费用为80%，预计到1985年软件费用将占90%。

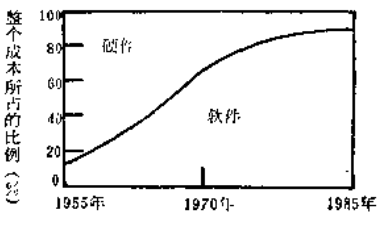

图 1.13 硬件和软件费用比例的推移

出现这种倾向的原因有二个：一个是由于计算机应用的复杂性日趋增加，在软件方面不得不承担比硬件更多的费用；另一个则是硬件与软件在技术进步上的差距。

在1950～1980年的30年里，计算机硬件的价格性能比至少提高了 10^4 倍(10年约为20倍)。也就是说，在30年前计算需要1亿日元，而如今1万日元就够了。而且，预料这种

趋势还可能持续10年。另一方面，软件的生产效率在1950～1970年的20年里只提高了约5～6倍(10年为2倍多)。软件终究是靠人来编制的，由于人工费逐年在递增，所以从价格性能比来看，上述数字将更加可观。

如上所述，如果在采用计算机进行信息处理的费用中，软件占80～90%的话，那么即使将来计算机硬件的价格进一步下降，对整个计算机费用的影响也不大。因此，为了降低信息处理的费用，不得不积极地设法降低软件费用。

正是由于软件的规模、复杂性的增加、可靠性不够、需要削减费用等原因，从六十年代末起，出现了"软件危机"。现在，人们对软件的技术革新(提高生产效率)，不断提出十分严格的要求。此外，从软件编制人员和开发人员来看，一方面是名人的技术，而另一方面是外行人玩弄花招专横一时，还有师徒制的教育、人海战术等旧的手工业生产方式的软件生产，针对这种情况，提出了软件工程(software engineering)。由此，在程序说明书的描述方法、程序设计方法、测试方法、程序设计组织和管理方法等整个程序设计过程中，已取得或正在取得重要的发展。

本书是在这些软件工程的成果中，参考了已经确立、评价的方法，同时也考虑了软件的编制方法。

复　习

[复习1.1] 简单说明如下名词：
(a) 程序　(b) 程序设计语言　(c) 算法　(d) 数据　(e) 软件

[复习1.2] 简单说明如下名词：
(a) 硬件　(b) 控制器　(c) 中央处理机(CPU)　(d) 程序方式
(e) 程序存贮方式

[复习1.3] 简单说明如下名词：
(a) 机器语言　(b) 汇编程序　(c) 编译程序　(d) 系统分析、设计
(e) 编码　(f) 调试　(g) 文档　(h) 操作系统　(i) 软件工程

[复习1.4] 说明程序的编制过程。

[复习1.5] 叙述学习软件编制方法的意义。

第2章 算 法

2.1 日常生活中的算法

A. 算法的表示

所谓算法，是指"进行某一群工作所需要的整个顺序"。

举例来说，要教会一个从来也没有打过电话的人来打电话，应该怎样说明才好呢？其方法大致如下。

[打电话的算法 1]

1. 拿受话器；
2. 拨对方的电话号码；
3. 交谈内容；
4. 将受话器放置原处。

同一算法可以用好几种方法来表示。假设将上面的表示方法称为"顺序表"。还可以用流程图来(flow chart)表示，如图 2.1 所示。根据流程图，可以用人眼来领会算法。

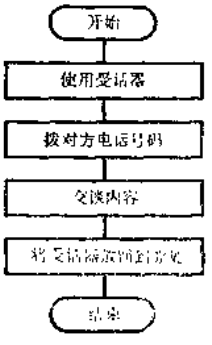

图 2.1 打电话算法 1 的流程图

上面的算法可采用最简单的结构，即从"开始"开始，依次执行下一条命令，以"结束"结束。这种结构称为逐次结构(s quence)。通常，控制应按什么顺序执引命令的结构，称为控制结构(contro structure)。在控制结构中，除了逐次结构以外，大致还有选择结构(selection)和重复结构(iteration 或 repetition)。通过使用选择结构和重复结构，可以描述更复杂、更有趣的算法。

B. 选择结构

为了执行别的命令，可根据各种情况使用选择结构。例如，在上例中，如果不知道对方的电话号码怎么办？可按算法 2 进行。

[打电话的算法 2]

1. 如果不知道对方的电话号码，就查电话簿。
2. 拿受话器。
3. 拨对方的电话号码。
4. 交谈内容。
5. 将受话器放置原处。

若用流程图来表示本算法，即图 2.2 所示。要注意，菱形框表示判断。这种流程图符号的使用方法是根据日本工业标准规格(JIS C 6270)决定的。

[练习2.1] 请按照日本工业标准规格来查对流程图符号的使用方法。

上述算法还可以用第三种方法来表示，这种方法称为虚拟代码*(pseudo code)。

* 所谓虚拟代码，指算法是用人所使用的自然语言编写，而不是用计算机能理解的程序设计语言编写。这里将其意思稍加扩充后使用。由虚拟代码描述的称为虚拟代码。

```
begin
    if(如果)不知道对方的电话号码，then(那么)查电话簿；
     拿受话器；
     拨对方的电话号码；
     交谈内容；
     将受话器放置原处。
end
```

这里，begin，if，then，end是为了表示控制结构而专门配备的词(称为关键字)。；是表示一条命令(亦称为语句)和下一命令的分隔符号。

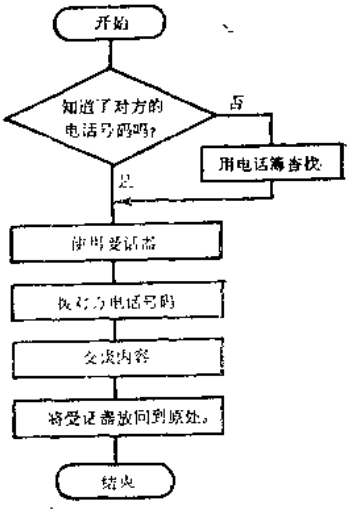

图 2.2 打电话算法2的流程图

[练习2.2] 试采用顺序表和流程图来表示用虚拟代码表示的算法。但是，if P then A else B 表示如果 P 成立，则执行 A，如果不成立，则执行B。

```
begin
    if 不知道对方的电话号码
      then if 电话簿在身边
        then 查电话簿 else 查号台进行询问；
     拿受话器；
     拨对方的电话号码；
     交谈内容；
     将受话器放置原处。
end
```

[练习2.3] 假定打电话的算法 1 不是公用电话的电话机，但当你使用的却是公用电话，则需修改此算法。可是只限于市内通话。试采用顺序表、流程图、虚拟代码来表示算法。

C. 重复结构

假如在算法 1 中的"拨对方电话号码"这条命令说明不清楚，希望你作更详尽的说明。图 2.3 表示了这部分更详细的算法例子。

将图 2.3 中的重复部分那样的控制结构称为重复结构或循环(loop)。请注意，需要印有菱形的判断框。因为如果没有这个判断框，就变成图 2.4，不能转移到下一个"交谈内容"框中去。通常将它称为"永久循环"。在循环中，必须有跳出用的判断框。

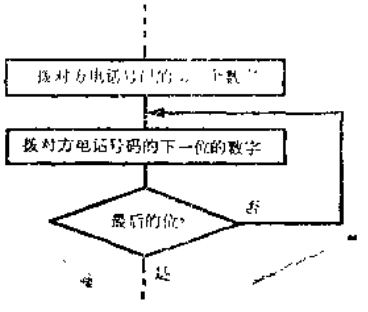

图 2.3 打电话算法 3 的流程图的一部分

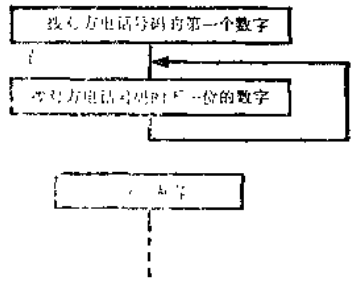

图 2.4 错误的流程图(永久循环)

若用虚拟代码来描述图 2.3 的算法，则如下所示：

拨对方电话号码的第一个数字；

repeat 拨对方电话号码的下一个数字；

until 最后的一位数字。

这里，repeat A until P 表示在条件 P 成立之前，反复执行 A。它是实现重复结构的一种控制结构。

此外，如果将图 2.3 的算法用顺序表来描述，则该部分为：

2a. 拨对方电话号码的第一个数字。

2b. 拨对方电话号码的下一个的数字。

2c. 是否最后一位数？若是(yes)，则转向 3，若不是(no)，则转向 2b。2c 中的"若是(yes)，则转向 3"可以省略。这是因为除了明确地指定了去向的场合外，还规定转入下一条命令。

[练习2.4] 试描述如下算法。当你打电话时，也有可能对方正在通话。正在通话时，规定等 5 分钟后再打一次。如果你重复三次(包括最初的尝试在内，共四次)还未接通时，就停止。此算法是否可以采用 repeat～until～，并使用虚拟代码写。

要注意，在结束这一节时，对于日常生活中的算法描述经常含糊不清(任意性)。一种是不同场合所伴随的任意性。正如由上所知的"是否知道了电话号码"、"电话簿是否在身边"、"是否是公用电话"、"是否正在通话中？"等因素是改变算法的条件。当然，如果再列举"即使查电话簿也不知道时"、"对方没有应答时"、"对方不在家但有别人时"、"想打电话但没有钱币时"、"公用电话有人占线时"等应考虑的条件，那就没完没了。

另一个任意性是关于"详细描述到何种程度"。在图 2.1 和图 2.3 的流程图中，详细描述的级不同。或者说，假如说明的对象是真正不知道怎样打电话的人，那么"拨对方电话号码的第一个数字，也许应更详细地描述如下(不是按键电话的场合下)：

2a1. 把手指塞入对方电话号码第一位数字的孔中。

2a2. 按顺时针方向拨电话机拨号盘，一直拨到手指不能往前移动的位置。

2a3. 放开手指后，一直等到拨号盘转回到原处。

在日常生活中的算法描述中，只能按照常识来酌定应该考虑的哪些情况和描述到何种详细程度，不能得出唯一而明确的解答*。相反，在计算机执行工作的算法中，如果有这种任意性那就麻烦了。这一点将在后面第2节中讨论。

[练习2.5] 试用算法来描述日常生活中的各种作业的方法，例如：

(a) 从早上起来一直到学校(或工厂)为止。

(b) 拍一张照片之前的照相机操作。

(c) 从大学(或其他学校)的入学考试到毕业。

(d) 做拉面。

(e) 得到汽车的驾驶执照。

---

* 关于这一点，已经指出"日常生活中的算法"的表示本来就是矛盾的词。因为"算法"具有严密的定义，如后面在 2.3 节中所述。作者认为这种评价比较恰当。然而，在学习使用计算机处理的算法之前，从算法上看日常生活中的各种作业顺序是一种很好的练习。此外，还具有它本身一种"达到目的的观点"价值。从这个意义上说，希望读者理解在本节中脱离了算法的严密的定义，而使用了不够严密定义的词。

(f) 将调频收音机的音乐录在盒式磁带上。

(g) 写信后投寄。

(h) 用现款挂号信汇款。

(i) 发动停止的汽车。

(j) 用现金卡提取存款。

[练习2.6] 在上面的练习中，可以用顺序表或流程图来描述，但有没有不能用虚拟代码描述的算法。如有的话，那是为什么？

[练习2.7] 在上面的练习中，有没有用顺序表、流程图、虚拟代码都不能描述的算法，那是怎么回事？

2.2 处理数值和字符的算法

计算机能进行的是处理数值的计算和用字符、符号表示的数据处理。因此，上述的日常生活中的算法用(如果也不是装有计算机的机器人)计算机是不能进行的。即使由机器人执行某种动作，那末由计算机执行的处理部分也只是数据的存贮、处理和加工等。

顾名思义，计算机是"进行计算的机械"，又因为包括了数值以外的数据，所以也是"处理数据的机械"。例如，当把几千个人的人名按一、二、三、四、五顺序重新排例，或给出某一人名时，找出与该人有关的数据的作业，如应用计算机将进行得十分频繁。在通常的意义下，很难说这些是数的"计算"。此外，在1.3节中已经谈到程序是由汇编语言和编译语言编写的，并由计算机本身翻译成机器语言。如果扩大对"计算"解释，就包括这种翻译作业。把类似这样的除数值以外还包括字符和符号的广义的数据，从一种形式变换到更符合目的的其他形式，称之为数据处理(data processing)。对于计算机在各个领域中的使用方法，大概除数值计算以外的数据处理占很大的比例。所以，对于计算机，也经常要使用数据处理系统(data processing system)的别名。

这里，要考虑的是计算机硬件如何来处理字符和符号。如果计算机处理的字符和符号的种类有限，那么就能给这些字符和符号分配固定的编号，这些固定的编号称为代码。表2.1中列出了日本工业标准规格(JIS C 6220)所规定的部分字符符号(代码)。因为根据符号字符和记号可一一对应地变换成数值，所以在原则上可以根据处理数值的运算来处理字符和记号。

然而，对于算法，最好字符(包括记号)的处理与数值不同。这些均可用数据类型(data type)的概念来加以区别。也就是说，数值和字符分别具有数值型(实际上可进一步分为整数类型和实数类型)和字符型的数据类型。数据类型将在第4章中作详细的讨论。

最后，讨论几个算法。

[例题2.1] 将二个整数M，N值的大小进行比较，然后编制将大值代入MAX，小值代入MIN的算法。

图2.5表示解法。这里，M、N、MAX、MIN都是在存贮器中能分别存放一种数据的单元(字)所附加的名字*(见图2.6)，称之谓变量名(variable name)。此外，：＝这个符

* 实际上不是附加名字，而是在硬件上用地址的数字来区别数据的存贮单元(字)。存贮一个数据的单元大小将根据数据类型而异。