BASIC 教程 上卷

基本BASIC

潘正伯 车克健 主编

科学出版社

BASIC 教 程

上卷 基本BASIC

潘正伯 车克健 主编

科学出版社
1991

内 容 简 介

本书以国内微型机上广泛使用的 BASIC 为基础，系统而完整地讲述 BASIC 语言及其程序设计方法，分上下两卷出版.

上卷包含基本 BASIC 的全部内容及扩展 BASIC 的内容，介绍了最基本的编程技术，诸如输入、函数、分支、循环、数组、字符串、子程序、顺序文件和随机文件等；下卷包含到目前为止 BASIC 最新版本的内容，即 Quick BASIC 4.0/4.5，介绍了各类结构程序设计，以及过程、文件、图形与图象、错误与事件捕获等较复杂的编程技术. 上下卷内容既密切相关，又自成体系，独立成册.

本书遵循程序设计的结构化思想和软件工程学的原则编写，结构紧密、层次清晰，内容安排由浅入深、循序渐进，含有其它同类书中未涉及的实用知识和大量具有代表性的例题及习题.

本书可作为大专院校、中等专业学校和各类计算机培训班的教材，也适宜有关人员自学.

BASIC 教 程

上卷 基本 BASIC

潘正伯 车克健 主编

责任编辑 那莉莉

科学出版社出版

北京东黄城根北街16号

邮政编码：100707

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1991年8月第 一 版 开本：787×1092 1/16

1991年8月第一次印刷 印张：16 1/8

印数：0001—8 600 字数：368 000

ISBN 7-03-002509-1/TP·187

定价：6.50 元

《BASIC 教程》编写组人员

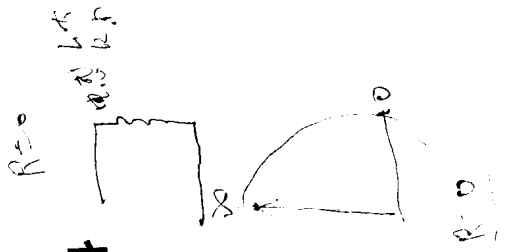
主　　编： 潘正伯　车克健

编审人员：（以姓氏笔划为序）

付佑平　孙忠林　朱鹏飞　李小平

杜学东　杨卫平　周书真　贾作皆

耿国庆　崔海源　程　勇　韩耀军



前　言

自80年代后期（确切地说是从1987年)以来，BASIC 语言在美国成了计算机程序设计的多种高级语言中最通用、最好用因而也是最重要的一种．如果某些读者对这一演变了解不多，请先浏览本书第一章第2～4页．此外，我们还推荐读者翻阅以下两种资料：

一种是 Robert Jourdain 编写的“Programer's Problem Solver for The IBM PC XT/AT”(陈学谦等译，IBM PC XT/AT 编程指南，电子工业出版社，1988年)．的序言．该书采用汇编语言编程、操作系统调用和高级语言编程三种方式介绍了150余项硬件控制任务，讲述了许多编程技术．在众多的高级语言中，作者选中了 BASIC．对此，他在序言中是这样解释的：

> 高级编程介绍了用高级语言编写的任务．尽管所用概念完全适用于 Pascal 或C语言，但实例却是用 BASIC 语言编写的．之所以选用 BASIC，一方面它是计算机领域内广泛使用的语言，每个 IBM 微型机用户都配有 BASIC；另一方面，Microsoft 公司的 BASIC 在应用 IBM 硬件方面，是其它程序语言不能与之相比的．

另一种是 Bill Gates（Microsoft 公司的创办人，现任公司总经理兼总裁，PC DOS 操作系统和 Quick BASIC 的开发人）为纪念 BASIC 诞生25周年发表在“BYTE”杂志1989年10月一期上的文章 THE 25th BIRTHDAY of BASIC（载该期第268～276页)．该文最后是这样结束的：

> 70年代后，正当C语言风行一时之际，有些人预报了 COBOL 和 FORTRAN 语言的衰亡，有的则说 Pascal 敲响了 BASIC 的丧钟．然而所有这些语言至今仍在使用．我相信，就 BASIC 而论，这种语言正在欣欣向荣地成长，一旦 BASIC 变成了一种通用的宏语言，依我看它正向此方向发展，那么它将比其它任何语言都长寿．实际上，BASIC 可能比我们这些人活的时间还长．

Robert Jourdian 和 Bill Gates 是就美国的情况说的，由于我国计算机装机量中，微型机所占比重极大，因此 BASIC 语言在计算机各种高级语言中的重要性必将比美国还要突出．

认清这一点，对于任何一个经常与计算机打交道的人来说，是有好处的．从这一点，即从承认 BASIC 语言的通用、好用和重要性出发，不难得出以下三个推论．

推论之一：计算机的初学者，最好循着 BASIC 语言这条捷径，尽快地掌握基本编程技术．

推论之二：具有一般程序员水平的专业和非专业软件开发者，应尽快熟悉并充分利用 BASIC 提供的优越的编程环境，提高自己的程序质量和编程效率，为社会提供更多、更好的软件产品．

推论之三：从事计算机教育的工作者，应设法在最短时间内，把 BASIC 语言的丰富内容教给学生，提高学生的编程能力，使他们今后能得心应手地使用计算机．

本教程的作者们几年来煞费苦心,反复研讨,就是希望能尽量圆满地去完成这一任务.

目前我国各大、中专院校中有的以 IBM PC 及其兼容机为教学用机,也有不少院校仍以 APPLE II 及其兼容机为教学用机,而且这种情况大概不会很快改变. 有鉴于此,我们既要顾及在 APPLE II 及其兼容机上的基本 BASIC 和扩展 BASIC 的教学,更要积极推广在 IBM PC 及其兼容机上的 Quick BASIC 的教学. 为此我们设想可以采取国外某些课程的作法,将 BASIC 语言课分为上卷和下卷来讲授. 上卷包含基本 BASIC 和扩展 BASIC 的内容,约需 40～50 学时,在大、中专院校的低年级开设;下卷包含当前 BASIC 的最新成果,即 Quick BASIC 4.0/4.5 的内容,约需 70～80 学时,在大、中专院校的二三年级开设. 通过上卷的学习,学员应能掌握基本的编程技术,写一些规模不大的程序; 通过下卷的学习,学员应能掌握更复杂的编程技术,其中的佼佼者将具有很强的编程能力和很高的编程效率.上述构想的实现必将大大提高我国学生和科技人员使用计算机的能力.

本书为 BASIC 教程的上卷,即基本 BASIC 卷,正是为体现上述构想而编写的,实际上它包含了扩展 BASIC 的全部内容,并以 APPLE II 其及兼容机为假想教学用机,同时兼顾 IBM PC 及其兼容机.

1985 年以后陆续出现了三种带编译器的、结构化的 BASIC. 它们是 True BASIC, Turbo BASIC 和 Quick BASIC. 其中 Microsoft 公司的 Quick BASIC 4.0/4.5 推出最晚,它吸收了前两种 BASIC 的精华,一凭 Microsoft 公司本身雄厚的软件实力,二凭它与 IBM 公司特殊的伙伴关系,加上公司总裁 Bill Gates 对 BASIC 的偏爱,使得 Quick BASIC 4.0/4.5 成了带编译器的、结构化 BASIC 的集大成者. 经我们试用,证明了 Quick BASIC 4.0/4.5 是软件开发中功能最强同时又是最好用的一种工具. 它可与其它语言混合编程,为软件开发者提供了一个非常理想的编程环境.

我们在长期的教学实践中,对潜藏在当前流行的某些教材中的弊端感受很深,因此在本教程的编写过程中随时注意阐明(至少是不要违背)程序设计的结构化思想和工程化方法,让学员一开始就知道软件的质量标准,养成编写文档的习惯. 所有这些无疑将有助于他们日后在工作中开发出高质量的软件.

此外,本教程还具有以下特点:

(1) 兼顾多层次的需要. 首先顾及我国的装机水平,既考虑到苹果系列机,又考虑到 PC 系列机; 其次顾及各学校、各专业计算机语言课的教学计划. 本教程适用于 40～100 多学时不同的教学计划,既可选作大、中专院校或培训班的教材,又可供科技人员自学之用.

(2) 充分重视上机操作. 根据历年指导学员上机实习的经验,在第二、三章中,辟专节写出"上机须知",其中包含了许多在其它教材中未涉及的实用知识. "上机须知"可以留给学员自学.

(3) 加入了计算方法的基本内容.考虑到我国理工科学生(除计算机和数学专业外)一般不学计算方法课,但如果连最基本的计算方法都不会,光学会语言是没有用途的,故第十二章采取综合练习方式,既全面应用了所学过的语言知识,又介绍了 17 种最常用的计算方法. 使用中可根据学时数和学员水平对这些内容进行取舍. 即使不讲,这章内容也是留给读者自学的很好的教材.

(4) 采用"键入"(输入)与"显示"(输出)一一对应的格式. 对于学校或培训班,可以

节省教师的辅导工作量;对于自学者,可以减少他们暗中摸索的苦恼.

(5) 结合章节内容编写了大量的例题. 学习编程一般是从摹仿入手的,对于这些丰富的例题,教师可以酌情选讲,学员则可用之作为编程的范例.

(6) 两卷可分可合. 本教程的上下两卷,保持着相对的独立性. 读者既可同时选用上下两卷,也可根据本单位的实际情况(如是否有 IBM PC/XT 及其兼容机)或本人的实际水平 (如是否已经掌握了基本和扩展 BASIC) 而选用其中的一卷.

(7) 有长期使用和保存价值. 本教程既是按教材的要求编写的,又保留了足够的常用资料,读者在学完之后,可把它当作 BASIC 用户手册长期保存使用.

我们感谢在美国进修的杜殿海讲师,他及时为我们提供了不少有价值的信息.

本教程的作者们虽然都是教学、科研、设计和管理部门的计算机工作者,但写这样的书尚属首次. 虽然我们竭尽努力,对书稿进行了反复修改,但终因水平所限,不能尽如人意,甚至有可能潜藏着某些我们尚未发现的错误,敬祈指正,不胜感谢.

潘正伯　车克健

1990 年 12 月 2 日

目　　录

第一章 基本概念

1.1 计算机语言

人类社会发展到今天，已经步入了以电子计算机技术为核心的新技术革命时代。它要求人们必须熟练地掌握多种工具(如外语、各种测试工具、作图和摄影工具、通讯和交通工具、打字和文印工具……)。在多种工具中，电子计算机是非常重要的一种。

要学会使用计算机，就要学会用计算机的语言给它正确地下达命令。计算机是人类的忠实奴仆，它会迅速地、不折不扣地执行它的主人给它下达的命令。因此，要使用计算机，第一步就要通晓计算机语言。

最初，计算机只有*机器语言*。它们是与机器的硬件设计同时产生的。机器语言程序表现为一组由二进制代码（只有0和1两种代码）组成的指令和数据。

在实用中机器语言有很多缺点：

(1) 这种程序难于看懂和调试。在连续工作几个小时后，你几乎会把各个二进制数看成都是相同的。

(2) 由于必须将每一位代码单个地送入计算机，所以程序的输入速度很慢。

(3) 人们无法从程序看出要计算机执行的是什么任务。

(4) 程序太长，编制程序很费劲。

(5) 程序员稍不小心就会出错，而这些错误又很难被发现。

由于机器语言的烦琐、单调，给使用和记忆都带来很多困难，因此人们在实践中想对它加以改进，这就是对每条指令都给予一个名称。这种指令码的名称叫做“助记符”。使用助记符的语言叫做*汇编语言*。

汇编语言仍然存在着一些明显的缺点：

(1) 像机器语言一样，汇编语言的指令系统（如把某两个寄存器的内容相加，把累加器的内容向左或向右移1位等)与用户通常要计算机做的事情(如检查模拟读数是否超过了终限值、在适当的时间启动继电器等）差别很大，程序员必须把用户要计算机做的事情翻译成一系列简单的计算机指令。这种翻译工作有一定的难度，并且很费时间。

(2) 汇编语言要求用户详细了解所使用的计算机的硬件性能和各种指令，了解计算机所采用的寻址方式以及其它许多知识。而这些知识与你要求计算机最终执行的任务无关。

(3) 汇编语言对计算机硬件的依赖性很大，所以是不能移植的。

(4) 为了完成某一特定任务，常常需要自己动手设计程序，很难从别人那里受益。

50年代末期，出现了*高级语言*。高级语言是面向过程的语言。这种语言可以使你用面向问题的形式而不是用面向计算机的形式描述任务。高级语言中的每一语句完成一个可识别的功能，一般它与许多条汇编语言指令相对应。然后，用一种编译程序将高级语言

编写的源程序翻译为计算机能识别并执行的机器语言指令.

对完成不同类型的任务,有许多可供选择的不同的高级语言. 例如,如果你想让计算机完成的任务用代数符号表示,就可选用 FORTRAN (公式翻译语言), ALGOL (算法语言)或 Pascal (另一种代数语言);如果要处理面向商业的任务,通常是选用面向商业的语言 COBOL. PL/I 是 FORTRAN, ALGOL 和 COBOL 的综合. 在分时系统中,则更多的采用 APL 或 BASIC.

一般地说,高级语言具有下列优点:

(1) 可以更容易、更快速地写出程序. 通常估计,用高级语言写程序大约比用汇编语言快 10 倍.

(2) 高级语言都有自己的语法. 许多高级语言的语法,已由国家或国际标准确定,从而保证了它有相当广阔的应用范围.

(3) 高级语言并不依附于任何特定的硬件结构——中央处理器 (CPU) 的型号,这样就使得程序员无需了解他们为之编写程序的计算机的任何特性,使程序员能把注意力集中到处理自己的任务上.

(4) 用高级语言写的程序,一般都是可以移植的. 因而当你给一台新的计算机作程序设计时,往往有许多已用高级语言为其它计算机所写的同类程序供你参考.

自然,高级语言也有它的缺点:

(1) 要掌握高级语言必须学会它的语法,这是要花费精力和时间的. 而且有些高级语言相当繁难,舍不得下功夫和功夫下得不够的人,最终不一定能学会.

(2) 高级语言需要通过一种编译程序将它翻译成用机器指令表示的目标程序. 编译程序的价格很贵,而且要占用大量的内存空间,这样就使用户能有效使用的内存空间变小.

(3) 由于整个编译过程是用程序自动完成的,它不能像熟练而又优秀的程序员那样,想方设法走捷径既缩短执行时间又减少存储器的占用量,同时它也不能方便地利用特定计算机的特定功能,结果使得程序的效率低,也难以在提高运行速度和减少存储器占用量两方面实现代码最佳化.

在众多的高级语言中,使用的范围和受欢迎的程度也不尽相同.

1981 年美国一家信息处理公司对 995 位管理信息系统经理作过一次关于所用程序设计语言的调查. 43.2% 的调查对象认为 COBOL 是最重要的语言,并有 37.8% 的人乐于用它, FORTRAN 的重要性占第二位,乐于选用者则占第四位. PL/I 因出世较晚,在重要性方面占第五位,乐于选用方面仅次于 COBOL 而占第二位. 在上述调查之后的六年,即 1987 年,美国波士顿计算机学会对其成员作了类似的调查,结果如下:

BASIC	80 ~ 90%
FORTRAN	60%
Pascal	35%
C	35%
汇编	30%
LISP	10%
COBOL	5%

PROLOG	5%
其它	2%

当然，这类调查因其对象的业务差异会有不同的结果，但是 BASIC 语言跃居榜首，却决不是偶然的.

1987 年英国出版的《朗曼当代英语词典》(第二版)第 827～828 页讲解 "program" 一词时，增添了一幅插图（见图 1.1）. 图上是一段很短的同时又是完整的 BASIC 程序. 为什么在众多的程序设计语言中单单选中了 BASIC? 当然这也决不是毫无道理的.

Program

```
READY
10 LET B=21
20 LET C=5
30 LET D=B+C
40 PRINT D
50 END
RUN
```

图 1.1

1960 年美国达特姆斯大学的凯梅尼 (John G. Kemeny) 和库尔茨 (Thomas E. Kurtz) 教授，发表了一种叫做 BASIC 的高级语言. 在一个长时期内，人们都认为 BASIC 不过是在它之前已经发表并广为流行的 FORTRAN 的一个子集. 后来人们才认识到这种看法并不正确.

BASIC 一词在英汉词典上的释义是：基础的，基本的，根本的. 然而，该语言的开发者并不仅仅是按照这层字面上的意义给该语言命名的.这两位教授玩了一个文字游戏，他们说 BASIC 这个词，是集 Beginners All-purpose Symbolic Instruction Code (初学者通用符号指令代码) 这些词的首字母缩写而成的. 这样一来，由五个字母组成的 BASIC 一词，就具有了双关的意义.

BASIC 语言本身具有什么特点？为什么它能博得如此广泛用户的青睐？

(1) BASIC 是普及面最广的计算机语言. 从装机量看，数量最多的是微型机. 任何一台微型机甚至像 PC-1211, PC-1500, HX-20, PB700 这些很小（但数量却相当庞大）的袖珍计算机上，都配有或仅配有 BASIC 语言.

(2) BASIC 是最易学的程序设计语言. 世界公认 BASIC 是最易学的程序设计语言，就我国的情况来说，这个特点更为突出和重要. 为此我们不妨将 BASIC 与 COBOL 作个比较. 80 年代初，国外正是 COBOL 语言独占鳌头的年代，国内的不少专家作了大量的工作，电视台也办过讲座. 但是 COBOL 语言在我国却始终没有热起来. 这与它的繁难和大量使用一长串一长串的保留字不无关系. BASIC 则不然，它的保留字很少超过 6 个字母(据统计 Quick BASIC 中保留字超过 6 个字母的仅 11 个，占 5%)，且都排在每行之首，一目了然.

(3) BASIC 是变化最大、发展最快的语言. BASIC 的发展变化可以从两个方面去看，首先是量的变化. BASIC 的版本繁杂，数量惊人.从 APPLE 机的 SOFT BASIC 到 PC 机的 PC BASIC, BASICA, GW BASIC, NBASIC, 直到 80 年代后期出现的 True BASIC, Turbo BASIC, Quick BASIC, 其版本多不胜数. 显然，每一版本都有自己的特点，而每一新版本总是要对老版本作出不同程度的改进. 举例来说，基本 BASIC 只有 17 条语句，11 种内部函数；而 True BASIC 1.0 版本，已经有 155 条语句，43 种内部函数，另外还增加了 30 条命令；Turbo BASIC 1.0 版本共有语句和函数 219 种，保留字 248 个. 其次是质的变化. 80 年代中期，BASIC 是解释型的非结构化语言.从 1985 年克米尼和库尔兹两位教授发表 True BASIC 起，BASIC 跃变成为兼具解释与编译功能的结构化程序设计语言.

相比之下，其它任何一种计算机语言都没有发生过这样大的变化.

尽管 BASIC 语言的变化很大，但是最新的 BASIC 却具有极好的继承性. 基本 BASIC 中的全部语句和函数，在新版 BASIC 中仍然通用. 在 Turbo BASIC 和 Quick BASIC 中，只要作极少修改，不仅能编译甚至能用更快的速度直接运行原先用 PC BASIC 或 GW BASIC 等版本编写的程序.

(4) BASIC 对用户来说是最友好的语言，对软件开发者来说是生产率最高的语言. 由于 BASIC 最初是解释型语言，因此编制和调试程序都很方便. 波士顿计算机学会的调查中说："一些程序员认为，用解释型 BASIC 编程序的速度快、效率高. 一旦完成编译，用户就不知道（或者不关心）程序是用什么语言编写的了."

另外，BASIC 本身具有的功能，很多在其它语言中难以找到. 如人机对话功能、字符串功能、日期功能、时钟功能、通讯功能、绘图功能、动画功能、音乐功能等等，其它任何一种语言都不如 BASIC 这样全面.

这些大概就是美国计算机用户和程序员，纷纷放弃了他们原先采用的计算机语言而改用 BASIC 的基本原因.

1.2 软件开发的工程化方法

1.2.1 软件危机和软件工程学

60 年代以来，在一些技术先进的国家中，计算机的应用领域越来越广，几乎涉及到社会生活的各个方面，如工厂管理、银行事务、档案管理、图书查询、航班、铁路和旅馆的订票等. 这些系统所用的软件都相当庞大，逻辑上很复杂，而且其功能需要不断地修改和扩充.

软件是抽象的、逻辑性的产品. 因为软件不是实物性的，看起来它似乎很"容易"修改，只要改动系统中的几个语句，系统的功能就可能发生很大的变化，因此修改软件似乎只是"举手之劳". 实际上软件系统的用户和开发人员也总是在不断地修改它，而且修改的工作量相当大.

通过多年的实践人们才逐渐认识到，软件系统的开发需要投入大量的人力和物力，而成果的质量却难以保证. 软件开发本质上是一个"思考"的过程，很难对它加以控制. 开发人员往往按照各自的爱好和习惯进行工作，没有统一的标准可循；管理人员事先难以准确地估计项目所需的时间和经费；技术人员在项目完成前也难以预料系统能否成功；更糟的是，失败的系统往往无可挽回除非从头做起，但由于时间和经费的限制，这常常是不可能的.

电子技术日新月异，计算机硬件的功能和质量在不断提高，价格却在大幅度下降. 软件的规模在迅速地扩大，而软件的开发方式却没有太大的变化. 因此同硬件投资相比，软件投资所占比例急速上升. 从图 1.2 中可以看出，美国对计算机的总投资中，1955 年软件投资低于 20%，1970 年上升到 60% 左右，1985 年达到 85%.

国外在研制一些大型软件系统时，遇到了一系列的困难，导致有些系统的彻底失败；有些系统虽然完成了，但却比原计划推迟了好几年；有些系统未能完全满足用户最初的期望；有些系统则无法进行修改维护. 两个著名的例子是 IBM 公司的 OS/360

系统和美国空军某后勤系统．这两个系统都花费了几千人年的努力，历尽艰辛，但是结

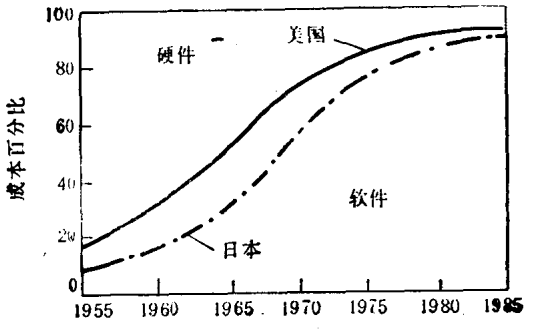


图 1.2　总投资中软、硬件的构成

果却令人失望．　OS/360 系统负责人布鲁克斯（Brooks）对当时的困难局面作了这样的描述："像野兽在泥潭中作垂死挣扎，挣扎得越猛，泥浆就沾的越多，最后没有一只野兽能逃脱淹没在泥潭中的命运……程序设计就像是一个泥潭……一批批程序员在泥潭中挣扎……没有人料到问题竟会这样棘手."

这种现象就是所谓的"软件危机".

软件危机的出现，推动人们去寻找克服危机的办法．人们逐渐认识到，沿用五六十年代个人编写小程序那种手工艺方式来研制大型软件系统是行不通的．正如不能用制造独木船的方式来建造远洋巨轮一样．

参考了机械、土建等一些古老行业的生产技术演变过程，产生了软件生产工程化的想法．在 1968 年北大西洋公约组织的学术会议上第一次创造了"软件工程"（Software Engineering）这个名词．后来经过 60 年代和 70 年代初的努力，终于形成一门比较完整的"软件工程学".

软件工程学是指导计算机软件开发和维护的工程科学．它采用工程的概念、原理、技术和方法来开发和维护软件．软件工程强调使用生存周期方法学和各种结构分析及结构设计技术．它所采用的生存周期方法学是人从时间角度对软件开发和维护的复杂问题进行分解，"各个击破"．整个生存周期分为若干阶段，每一阶段的开始和结束都有严格的标准．每一阶段结束之前都必须进行严格的技术审查和管理复审．审查的一条主要标准就是每个阶段都要交出"最新式的"（即与所开发软件完全一致的）高质量的文档资料，从而保证在软件开发工程结束时，有一个完整、准确的软件配置交付使用．

1.2.2　文档的重要性

软件工程学诞生前，软件开发者们都按照自己的习惯、爱好和风格编写程序，并且认为"研制软件"就是"编程序"，软件就是程序清单．

早就该纠正这种有害的看法了！

软件工程创始人之一的贝姆（B. Boehm）在 1976 年给软件下了一个新的定义："软

件是程序以及开发、使用和维护程序所需的所有文档”. 这个新的定义纠正了过去软件工作者重程序、轻文档的偏向，强调在研制过程中，及时地按一定规格产生各种文档是研制工作的有机组成部分，必须充分重视.

1986 年出版的由日本科学技术联盟组织主编的“软件质量管理丛书”，将程序与文档在软件中的比重列出了一张表：

文档 + 程序	
15 年以前	程序
10 年以前	程序（主）+ 文档（从）
5 年以前	文档（主）+ 程序（从）
将来	文档 + [程序]
硬件的低成本生成技术	由文档 [半] 自动化地生成程序

由此可见，随着计算机技术的发展，文档在软件中所占的比重越来越大.

本书的读者是 BASIC 语言的初学者，我们无法对各阶段文档的具体要求作详细描述，但是我们要求程序设计语言的初学者，在编写程序的同时对按规格编写文档一事给予足够的重视，并在学习 BASIC 语言时按照下列七项内容，编写文字和图表说明，以养成及时编写文档的良好习惯：

(1) 题目.
(2) 算法分析或算法简介.
(3) 流程图.
(4) 变量说明.
(5) 程序清单.
(6) 使用方法.
(7) 运行结果（最好是有代表性的二三组）.

这七部分中，流程图占的分量最重，有了流程图很容易写出相应的程序来.

1.2.3 流程图

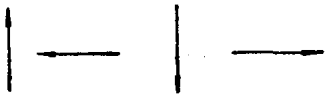
流程图是算法的图形表示. 它通过一些通用的、已经标准化的图框和流向线等，比较清楚地描绘程序的构造和流向. 事实上，看懂一张流程图比看懂一份程序要容易得多 .

尽管各国的计算机科学家对流程图的作用评价不一，特别在结构化程序设计语言诞生后，程序的流向受到很多约束，并且显著地简化了，但是流程图在计算机界至今仍被广泛地应用着.

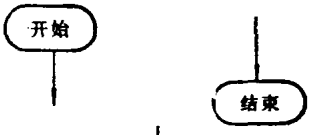
1. 流程图的符号

流程图的符号，国家有明确的规定，国际上也有统一的标准. 我国国家标准局批准的国家标准（GB-1526-89）与国际标准化组织公布的国际标准（ISO-5807-85）是一致的，其中最常用的有以下几种：

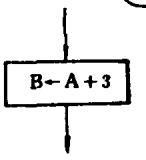
流向线. 箭头所指的方向就是程序的流向.



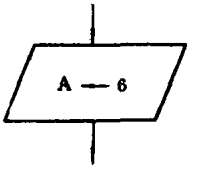
起止框．标志程序的开始、停止、中断或结束．



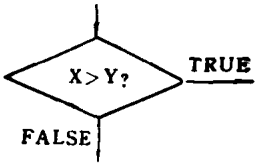
处理或操作框．框中写明进行的是什么操作．这里所表示的是把 $A+3$ 的结果放入 B 单元．



输入、输出框．框中写明输入、输出的内容．这里是表示向 A 单元中输入 6．

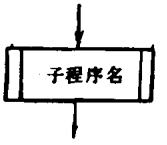


判别框．菱形的判别框中书写着一个逻辑关系表达式(如 $X>Y$?)，如果此表达式为真，程序就转向 TRUE(也可用 YES、"真"、"是") 箭头的方向执行；否则就转向 FALSE(也可以用 NO、"假"、"非")箭头的方向执行．

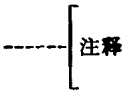


连接点即接点．流向线画到边沿处无法再画下去，让它进入连接点，再到另一个地方从同标号的连接点往下画．

既定准备框．表示子程序等在别的区域定义的一组命令或者由几个操作构成的处理过程．



注释框．对程序或输出信息作的简短解释．



2. 流程图举例

例 1.1 绘出将华氏温度变换为摄氏温度的流程图．

华氏温度变换为摄氏温度的计算公式如下：

$$Tc = 5(Tf - 32)/9$$

其中，Tf——已知的华氏温度数据；Tc——要求的摄氏温度数据．

开机后先输入 Tf (已知的华氏温度)．经过运算得到要求的 Tc (摄氏温度)．最后将 Tc 的值输出．

这就是一个小小的计算程序．图 1.3 就是这个程序的流程图．

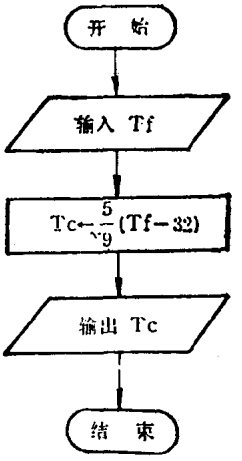


图 1.3

例 1.2 为鼓励节约用电，规定居民每月耗电不超过 25 度者，每度按 0.18 元收费，超此限者每度按 0.23 元收费．现绘出电费计算的流程图，如图 1.4 所示．图中，

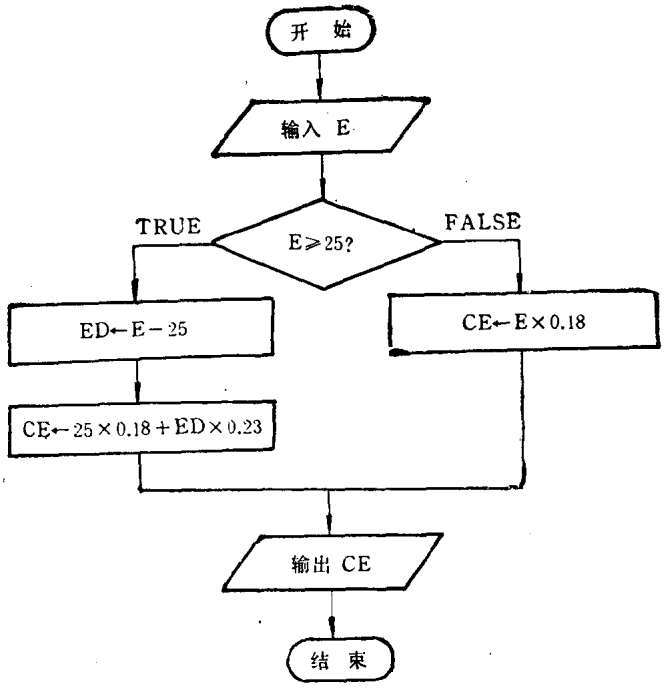


图 1.4

E——某户居民某月耗电量（度），

ED——超过 25 度的部分（度）；

CE——该居民户该月应交纳的电费（元）.

例 1.3 绘出看电影找座位的流程图，如图 1.5 所示.

这是一个智力游戏类的流程图，如果它能引起读者的兴趣，使读者感到"流程图并不难绘"，就达到我们的目的了.

1.2.4 软件的质量标准

软件工程学的目标是按计划获得高质量的软件，如何评价软件的质量呢？

早期，人们往往强调程序的技巧、效率、正确性等. 近年来随着计算机硬件功能的增强，价格的下降，随着软件规模的扩大和复杂性的增加，人们已倾向于从以下几个方面对软件作出更全面的评价：

(1) 可理解性. 软件产品的可理解性是指，该软件产品的用途应该是清楚的. 这就是说，软件产品应书写得清楚、简单；画有精练的流程图，具有恰当定义的专用符号；前后一致的通用符号；完全相符的人机对话和注释；整个程序中尽量使用一望而知的带说明性的变量名.

(2) 完整性. 要列入软件产品的所有组成部分，同时对组成部分的每一个组成要作全面充分的说明.

(3) 简明性. 软件产品的简明性是指它没有包含多余的资料. 过分冗长和重复的叙