TRUE BASIC
结构化程序设计

李印清　李文华　编著

北京航空航天大学出版社

True BASIC 结构化程序设计

李印清　李文华　编著

北京航空航天大学出版社

(京)新登字 166 号

内 容 简 介

本书全面、系统地介绍了 True BASIC(2.03 版)的各种语言成分,着重讲述了结构化程序设计以及算法设计的基本方法和策略。全书共十二章和两个附录。第一章介绍了算法的概念、表示,算法设计的基本方法和策略,以及富有生命力的一种软件设计表现法——PAD。第二至十一章介绍了 True BASIC 程序设计的主要内容,如输入、输出、处理、数组、函数、子程序、库和模块,并结合结构化程序设计思想,讲述程序的模块化设计、菜单驱动程序技术、递归技术、文件处理、作图技术、调试和陷井技术等高级程序设计的概念和技巧。第十二章详尽地介绍了 True BASIC 程序的上机操作。书末给出了两个附录,以供读者查阅。

本书可作为高等院校非计算机专业计算机学习的基础教材或教学参考书,同时可作为计算机培训班的教材,还可供从事经济工作、企事业管理及其他有关人员自学和参考。

书　　名: True BASIC 结构化程序设计
True BASIC JIEGOUHUA CHENGXUSHEJI
编　　著: 李印清　李文华
责任编辑: 娄铁军　齐桂森
出　　版: 北京航空航天大学出版社(北京市学院路 37 号,100083)
发　　行: 新华书店总店科技发行所
销　　售: 本社及各地新华书店
印　　刷: 朝阳科普印刷厂
开　　本: 787×1092　1/16
印　　张: 18.75
字　　数: 480 千字
印制日期: 1994 年 10 月第 1 次印刷
印　　数: 13000 册
书　　号: ISBN 7-81012-509-5/TP・126
定　　价: 12.95 元

前 言

True BASIC(2.03 版本)是 BASIC 创始人 John G. Kemeny 和 Thomas E. Kurtz 于 1987 年推出的最新 BASIC 版本。它保留了 BASIC 语言的人机交互、易学易用的基本特点，并对 BASIC 和 True BASIC 1.0 版本作了大量修改，同时吸收 FORTRAN、PASCAL 等语言的长处，使之成为一种完全结构化的程序设计语言，为用户提供了一整套按照软件工程化要求开发软件的重要手段。

True BASIC(2.03)的主要特点是：

(1) 典型的结构化程序设计语言

这使得设计的程序结构清晰，易阅读，易维护。在程序的编制过程中，如果大量使用 BASIC 的 GOTO、GOSUB、ON-GOTO 等语句，势必使程序的逻辑流程紊乱，结构缠绕不清。True BASIC 从根本上改变了这种结构，抛弃了行号，废除了 GOTO 等语句，充实了选择结构和循环结构。另外，引入了 EXIT 簇语句，以解决结构化程序有时执行效率低下的问题。

(2) 是一种"两栖型"语言

True BASIC 是一种既有解释器又有编译器的"两栖型"语言。在编写、调试程序时使用它的解释器，此时用户处于全屏幕编辑状态，可以对指定语句行或标定的行模块——多行组、一个子程序、一个函数、一个图画进行编辑、复制、删除、移动等操作，使查错、纠错简单易行，提高编写、调试程序的效率。程序一旦调试通过，则用它的编译器将程序优化编译，并经汇集以生成执行文件，日后使用时可只用它的可执行文件。

(3) 功能丰富，有足够的能力处理各种较大的应用课题

与任何一种高级语言相比，在语言提供的功能方面 True BASIC 毫不逊色。它具有极其丰富的各种功能，如字符串处理功能；矩阵运算功能；文件处理功能；高效率的作图、动画功能；库文件和外部函数、子程序和图画的递归调用功能；音响功能；多窗口操作功能；以及时钟、日期功能；菜单功能；以绝对地址方式读写内存功能等等。

True BASIC(2.03)还提供了与 DOS 操作系统的简练而方便的接口方式，可在程序中直接调用 DOS 的命令和功能，并可与外部的可执行文件进行链接。

(4) 良好的通用性

True BASIC 是严格按照美国国家 BASIC 标准编写的，具有良好的可移植性。与 BASIC 有较强的兼容性，在 True BASIC(2.03)中还提供了自动变换程序。

为了适应计算机科学技术的迅速发展，满足计算机基础教学的需要，我们在总结多年教学实践的基础上编写了此书。

全书共十二章。第一章可作为全书的导引，讲述算法的概念、表示，算法设计的基本方法和策略，以及富有生命力的一种软件设计表现法——PAD。第二至十一章全面系统地介绍了 True BASIC 程序设计的主要内容，如输入、输出、处理、选择、循环、数组、函数、子程序、库和模

块，并结合结构化程序设计思想，讲述程序的模块化设计、菜单驱动程序技术、递归技术、文件处理、作图技术、调试和陷井技术等高级程序设计的概念和技巧。第十二章详尽地介绍了 True BASIC 程序的上机操作。书末给出了两个附录，以供读者查阅。

“授之一鱼，莫如授之以渔。”本书注重基本算法、结构化程序设计方法的讲述和训练。书中不是简单地罗列语法规定和使用细节，而是在分析问题的基础上着重于被处理对象（数据）的抽象以及从计算机的操作的角度对解题过程的抽象。本书的另一特点是利用先进的图形工具 PAD 来描述算法，使算法执行逻辑透明可视。

本书编写时力求做到由浅入深、通俗易懂，以及知识结构的逻辑性。在此基础上列举了大量例题，以帮助读者掌握算法设计和编写程序的方法，不少例题取材新颖且具有实用价值。书中各章习题数量较多，难易程度合理搭配，为教学提供了充分的选择余地。

本书可作为高等院校非计算机专业计算机学习的基础教材或教学参考书，讲授学时为 50～70。在学时少的情况下，可根据学生的情况酌情删去书中某些内容，本书目录页中给出了参考性意见（即建议删去带“*”的章节）。本书 同时可作为计算机培训班的教材，还可供从事经济工作、企事业管理及其他有关人员自学和参考。

本书编写过程中，得到许多同志的热情帮助。王保正、罗道昆、赵逢余等同志为全书的编写提出许多宝贵的意见，在手稿处理过程中还得到牛晓太、吴昊等同志的不少帮助。藉此机会谨向他们以及本书中所借鉴、引用的参考文献的所有作者一并表示衷心的感谢。

本书第 1、6、7、8、10、12 章由李印清编写，第 2、3、4、5、9、11 章由李文华编写，李印清拟定了该书的编写大纲，编撰了两个附录，并对全部书稿进行了统修。

由于时间仓促，学识和水平有限，书中谬误之处在所难免，敬请读者批评指正。

编著者

1994 年 5 月

目 录

第一章　算法设计与PAD

要设计出好的程序，必须先设计出好的算法。算法是精确定义的一系列规则，指出怎样从给定的输入信息经过有限步骤产生所求的输出信息。

在这一章中，首先讲述算法的概念、特征，流程图表示以及设计算法的基本方法和策略，然后介绍极富生命力的软件设计表现法——PAD，以及基于PAD的编程方法。

本章可以作为全书的导引，其中介绍的有些概念和方法可能在后续章节或者读完全书后才会有深刻的理解。

第一节　算　法

一、算法的概念

什么是算法？著名计算机科学家D. E. Knuth在《THE ART OF COMPUTER PROGRAMMING》一书中称："一个算法，就是一个有穷规则的集合，其中之规则规定了一个解决某一特定类型的问题的运算序列。"

实际上，设计算法并非神秘莫测，它与人们做任何事情相仿，均要在规定的解题环境中，采用所允许的基本操作去构造解题的步骤。注意，这里所说的"解题"泛指解决某一类问题，而不仅仅是指"计算"。

设计由计算机执行的算法，必然涉及计算机的功能，即它能做什么。尽管我们今天可以见到种类繁多、大小各异的计算机，但它们都具备以下最基本的操作功能：

算述运算：加、减、乘、除。

逻辑运算："与"、"或"、"非"。

比较运算：$>$、$<$、$=$、$\neq$。

数据传送：输入、输出、赋值。

控制转移。

虽然上述运算和操作在不同系列的计算机上的表现形式不尽相同，但在设计算法时均应按照解题要求，从上述基本操作中选择合适的操作构成解题的操作序列。

下面通过几个简单问题说明设计算法的思维方法。

〔**例1-1**〕　有两个变量a和b，要求将它们的值互换。

算法分析：由于计算机存取信息的特征是：将一个数据或某变量的值自某个存储单元向另一存储单元传送时，该单元中的原有数据将由新值替代，而发送项存储单元的值保持不变。因此不能直接做a、b的值互相传递的操作，需要引入一个临时变量t(t对应于一个存储单元)。算法可设计为：

s1：输入a、b的值；

s2：作a⇒t；

s3:作 b⇒a;

s4:作 t⇒b;

s5:输出 a、b 的值;

s6:结束。

其中,s1, s2, ……分别表示操作序列的第一步,第二步,……。"a⇒t"表示赋值传送,即将 a 的值传到 t 所对应的存储单元中。

〔**例 1-2**〕 计算函数 M(x)的值。函数 M(x)为:

$$M(x)=\begin{cases} bx/h & x\leqslant a \\ a(h-x)/h & x>a \end{cases}$$

其中,a, b, h 为常数。

算法分析:首先引入变量 M,它代表要计算的函数值。根据计算机具有逻辑判断的基本功能,算法可书写为:

s1:输入 a, b, h, x 的值;

s2:判断 x≤a? 若条件成立执行 s2.1,否则执行 s2.2;

s2.1:作 bx/h⇒M;

s2.2:作 a(h−x)/h⇒M;

s3:输出 M 的值;

s4:结束。

算法中利用了计算机具有逻辑判断的功能,即判断"x≤a",若真(true),执行"bx/h⇒M",然后执行"输出 M 的值";若假(false),执行"a(h−x)/h⇒M",然后执行"输出 M 的值"。

〔**例 1-3**〕 计算 Fibonacci 数列:1,1,2,3,5,8,13,21…前 n 项的值。

算法分析:从数列的第 3 项开始,每项都是其前两项之和,即 $F_i=F_{i-2}+F_{i-1}$(i=3,4,5,6,…)。算法可设计为:

s1:输入 n 值;

s2:作 1⇒f1, 1⇒f2;

s3:输出 f1, f2 的值;

s4:作 3⇒i;

s5:判断 i≤n? 若条件成立,执行 s5.1～s5.5 规定的操作;否则执行 s7。

s5.1:作 f1+f2⇒f;

s5.2:输出 f 的值;

s5.3:作 f2⇒f1;

s5.4:作 f⇒f2;

s5.5:作 i+1⇒i;

s6:转向 s5;

s7:结束。

需要说明的是,算法中的 s5 和 s6 构成了一个循环处理过程。"f1+f2⇒f"表示计算下一项的值,"f2⇒f1"和"f⇒f2"操作是为下一次重复处理作准备工作。也就是说,每次重复处理时,f1 和 f2 已经获得新值。

进一步分析上述算法,可以发现其中的 s5、s6、s7 三个操作步骤的执行顺序可改写为:

s5：当 i≤n 时，重复执行下面的操作：

s5.1：作 f1＋f2⇒f；

⋮

s5.5：作 i＋1⇒i；

s6：结束。

由上述例题可以看出，一个算法由若干操作步骤构成，并且这些操作是按一定的控制结构所规定的次序执行。如例 1-1 中的各操作是顺序地执行的，称之为顺序结构。例 1-2 中的 s2 又包含 s2.1 和 s2.2 两个操作，它们不能同时被执行，需要根据条件成立与否决定执行哪个操作，这种结构称为选择结构。例 1-3 中的 s5（指改写后的）中的 s5.1～s5.5 是重复执行的，并且一直延续到条件“i≤n”不成立为止，这种结构称为重复结构，亦称为循环结构。

人们已经证明：任何复杂的算法都可以用顺序、选择、重复三种控制结构组合而成。这三种控制结构称为算法的三种基本控制结构。

说读者而言，理解算法由一步一步的操作和控制结构两个基本要素组成是必要的。设计算法与演绎数学有明显区别，后者以公理系统为基础，通过有限次推演完成问题的求解，每次推演都是对问题作更进一步的描述，如此不断推演，直到能将问题的解描述出来为止。而设计算法则是利用解题环境提供的基本操作对输入数据进行逐步加工、变换和处理，从而实现解题目标的。

二、算法的特征

依据 D. E. Knuth 的观点，算法应具有以下几个特征：

1. 确定性

算法的每一步其含义必须有确切的定义，即每一步运算应该执行何种操作必须是明确无误的，不应该存在歧义性，使执行者无所适从。

2. 有效性

算法中的每一步操作都应该是能有效地执行的，如果算法中有一个操作是不可执行的，整个算法就不具有有效性。例如，出现“计算 s/0”就不能进行有效的操作。

3. 有零个或多个输入

所谓算法的输入是指在算法开始之前对算法最初给定的量。这些输入取自于特定的对象集合。例如，例 1-1 算法中有 2 个输入，即需要输入 a 和 b 两个值。

4. 有一个或多个输出

所谓输出是指与输入有某种特定关系的量。例 1-2 算法的输出有一个，即 M 值。没有输出的算法是没有意义的。例如，求方程 $ax^2+bx+c=0$ 的实根，若任输入一组值，方程可能有解，也可能无解，无解时算法应给出无解信息。

5. 有穷性

一个算法应当包括有限个操作步骤，而不应该是无限的。换言之，一个算法必须总是在执行了有穷步的运算之后终止。

应当说明，就算法的应用性而言，有穷性的限制是不强的。一个有实用价值的算法应该不仅要求有穷的步骤，而且应该是很有限的步骤。例如，求线性方程组的解，理论上可以用行列式解法，但是要解 n 阶方程组，要计算 n＋1 个 n 阶行列式的值，总共要做(n!)(n－1)(n＋1)次乘

法运算。设 n 为 20，采用每秒 1 千万次的计算机，要连续运算 3 千万年才能完成。显然这种算法是没有实际意义的。由此可知，对算法的效率要作分析，对不能在相当有穷步内终止的算法则不必在计算机上运行，对这类问题的求解应另辟蹊径。

凡是一个算法，都必须满足以上特征。需要说明的是，仅满足前四个特征的一组规则不能称为算法，而把它称作计算过程。操作系统就是计算过程的一个重要例子。设计操作系统的目的是为了控制各种作业的运行，当没有作业时，这一计算过程并不终止，而是处于等待状态，一直等到一个新的作业进入。

把一个算法视为一个函数，对深入理解算法概念会有所帮助。数学中，如果一个函数定义为：$f(x)=x^2$，则$-\infty<x<+\infty$是 f(x)的定义域，$0\leqslant f(x)<+\infty$是其值域。类似地，把一个算法视为一个函数，是因为它把输入值的集合变换为输出值的集合，算法的定义域是一切使该算法有定义的输入值的集合，算法的值域是算法输出的一切可能值的集合。可见，一个算法定义了一个函数或一个映射，它把输入值变换为输出值。算法的输入和输出值可以是任何可表示的符号，可以是数、字母字符、特殊字符或任何其他有意义的能用计算机表示的符号。

三、算法的表示

语言是思想的外壳，设计的算法也要用“语言”恰当地表示出来。通常，可以使用以下三种类型的工具描述算法。

1. 自然语言

自然语言即人们使用的语言，如汉语、英语等。用自然语言描述算法通俗易懂，但它存在着难以克服的缺陷。

(1) 易产生歧义性。自然语言往往要根据上下文才能判别其含义，不太严格。

(2) 语句比较繁琐冗长，并且很难清楚地表达算法的逻辑流程。如果算法中包含判断，循环处理，尤其是这些处理的嵌套层数增多，用自然语言描述其流程既不直观又很难表达清楚。

(3) 当今的计算机尚不能处理用自然语言表示的算法。

2. 专用工具

为了形象地描述算法，人们创造了许多专用工具来描述算法。常用的有流程图，PAD 图，N-S 图等。除图形工具之外，人们可使用代码符号(如伪代码)描述算法。

3. 程序设计语言

程序设计语言是可以让计算机接受、辨认和执行的算法描述工具。任何算法最终都要用程序设计语言描述出来，才能在计算机上实现算法欲解决的问题。

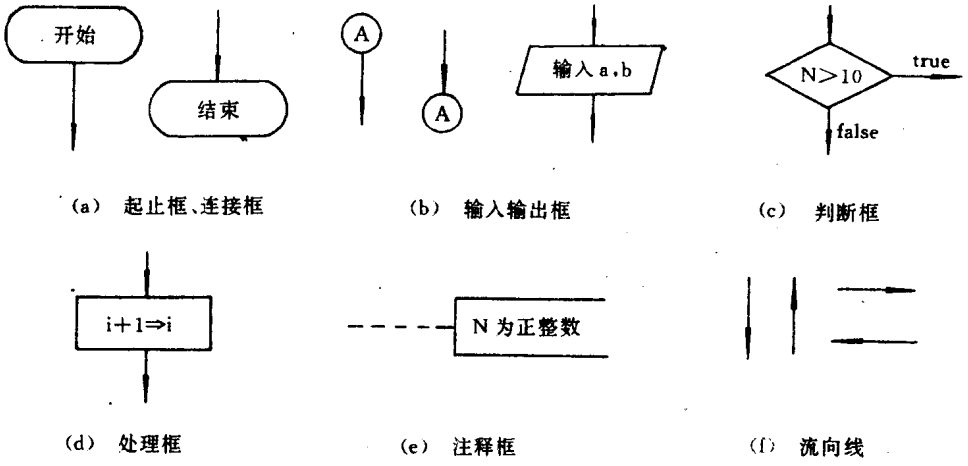
由于用程序设计语言描述的算法其表现形式是串行的，难以直观地反映算法的逻辑结构，因此，在算法设计阶段人们仍然需要利用图形工具描述算法，然后再利用程序设计语言将算法翻译成可以由计算机处理的语句序列，即程序。这一工作过程被称之为编码。

下面简要介绍流程图的使用方法。

流程图是采用不同的几何图形来描述算法的逻辑结构，每个几何图形表示不同性质的操作。图 1-1 是 ANSI(美国国家标准化协会)规定的一些常用流程图符号。

起止框、连接框：表示算法的起始、终止和连接，框内填写文字或字母，如图 1-(a)所示。

输入输出框：表示输入数据和输出计算结果，框内应填写需要输入或输出的量。如图 1-1 (b)所示。



(a) 起止框、连接框　(b) 输入输出框　(c) 判断框

(d) 处理框　(e) 注释框　(f) 流向线

图 1-1　常用流程图符号

判断框：根据条件判断算法继续执行的走向，框内应填写条件。如图 1-1(c)所示。

处理框：表示执行计算赋值操作，框内用文字或符号表明具体实现的操作。如图 1-1(d)所示。

注释框：表示对算法中的某一操作作必要的备注说明，框内用文字说明备注信息。如图 1-1(e)所示。

流向线：表示算法的走向，箭头的方向表示算法执行的方向。如图 1-1(f)所示。

下面给出例 1-1，例 1-2，例 1-3 解题算法的流程图。

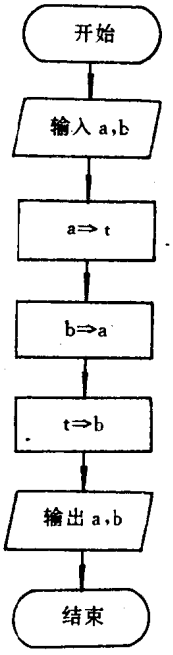


图 1-2　交换 a,b 两变量的值

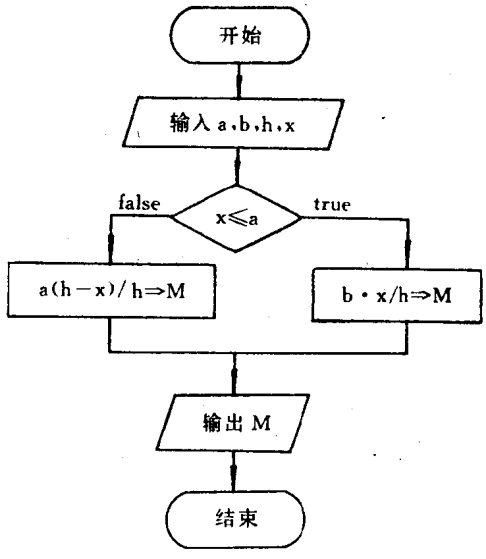


图 1-3　计算函数 M(x)的值

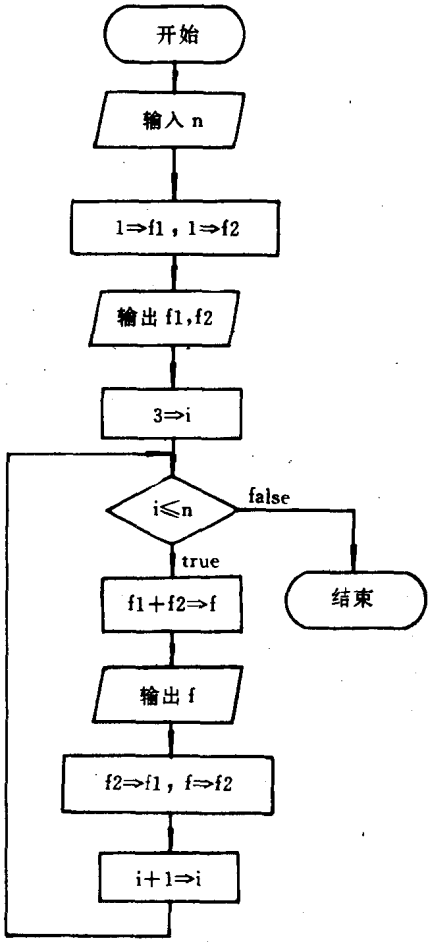


图 1-4　计算 Fibonacci 数列前 n 项的值

以上介绍了算法的概念、特征及其流程图表示。实际上，设计算法还要涉及对算法所处理数据的研究。算法所处理的数据通常并不是杂乱无章的堆积，往往具有重要的关系，这就是数据结构的内容。实践证明，对数据进行合理的组织，选取合适的数据结构将会直接影响算法的选择及其执行效率。瑞士计算机科学家 N. wirth 用一个公式来揭示算法与数据结构之间的联系：算法＋数据结构＝程序。

数据结构是介于数学、计算机硬件、计算机软件之间的一门交叉学科，它研究的内容包括：数据的逻辑结构、物理结构以及数据结构上所施加的运算。

本书将程序设计中涉及到的有关数据结构的内容放在后续相关章节中讲述，届时，读者应给予足够的重视。

第二节　设计算法的基本方法和策略

为了获得一个有效的算法，读者必须了解一些解题的基本思想和方法。本节仅对构造算法所依据的一些基本策略（公式、方案、原则），即解题思想进行讨论，其目的是使读者遇到实际问