dBASE Ⅲ应用程序设计方法

田怀录 编

科学出版社

dBASE Ⅲ
应用程序设计方法

田怀录　编

(京)新登字 092 号

内 容 简 介

本书主要介绍结构化程序设计的方法.通过剖析一个样板软件 Accounts (记帐)系统的实例，讲述了 dBASEⅢ 应用程序的结构、内容和设计步骤，并概述了软件系统和数据库管理系统设计的一般原理.

本书读者为计算机程序人员及用户，也可作大专院校学生 dBASEⅢ 后继课程的教材.

dBASE Ⅲ 应用程序设计方法

田怀录 编

责任编辑 张启男

科学出版社出版

北京东黄城根北街 16 号

邮政编码 100707

上海王桥印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

1992 年 7 月第 一 版 开本: 787×1092 1/32

1992 年 7 月第一次印刷 印张: 8 3/8

印数 1—3.000 字数: 193 000

ISBN 7-03-003289-6/TP·243

定价: 6.50 元

前　言

程序设计方法的研究已经历了两个阶段：第一阶段以结构程序设计为特征，于70年代末80年代初成熟．第二阶段是80年代兴起的面向对象的程序设计(OOP)．这种方法现仍在研究中，尚不能大范围应用．

程序设计方法的研究进展较快，但就其推广应用来说，远不够广泛，存在着理论与实际脱节的现象，这主要表现在对应用程序的开发上．尽管国内计算机用户很多，也有着大量的具有专业知识的计算机工作人员，但总的来讲，其应用程序的开发能力却很差，致使目前软件开发的费用高、时间长，特别是对已开发使用的软件还不能作及时的维护(修改扩充)．造成这种现象的原因是程序设计方法的教学跟不上，缺乏适宜的教材．

我认为，向应用领域的人员讲述程序设计方法，应着重讲解结构程序设计方法，然后由此引导到新的方法——面向对象的程序设计方法．为此，本书将以特殊语言 dBASE Ⅲ 为例，通过解剖样板系统——Accounts(记帐)系统，向极为广泛的企业管理领域的广大软件工作者讲解结构程序设计方法，为程序设计方法的研究与推广打下良好的基础．

目前出现的 FoxBASE⁺，dBASE Ⅳ 较 dBASE Ⅲ 先进．但就其程序设计方法来说是共同的，而且 dBASE Ⅲ 目前正被广泛使用着，FoxBASE⁺，dBASE Ⅳ 只能对其兼容．

本书首先系统讲述结构程序设计的一般原理，而后论证 dBASE Ⅲ 程序的结构性，并提出 dBASE Ⅲ 应用程序的内

2515/6

容与要求，以及具体的设计步骤。由于 Accounts 是一个系统，因此接着简单介绍了软件结构的设计、模块、数据库与报表．并结合实例详细讲述了输入(input)，数据处理(process)与打印输出(output)这三类程序设计的基本内容．

剖析 Accounts 系统，不可能讲述其中的一个程序，本书只讲述那些笔者认为最有代表性的程序，而且以模式的方式给出程序结构．由于篇幅所限，对于程序的最后连接，特别是辅助语句的插入，请读者自己去查阅原程序．

在剖析原程序时，本书以讲述原算法的基本策略及其推导思想为主，对程序中的不足之处也提出一些看法，或再给出另外方案．

原程序是通过对 dBASE Ⅱ 手册中的 Accounts 系统改写而成的，但还保留有 dBASE Ⅲ 已不存在的指令．原程序错误的地方，书中已加了说明．这样做，一方面是使读者看到原始材料，自己去学习、分析；另一方面也是为了防止笔者判断有误．为了阅读方便，书中已将程序中的英文说明(内部文档)译成中文．

学方法容易，掌握方法难．这是因为方法简单，客观对象复杂．只有通过练习才能逐步掌握方法．本书不足之处是未选出一定数量的习题．我想读者可以对 Accounts 系统各程序再提出一些方案作为练习．也可以通过一两个实际问题作为练习．

限于笔者的学术水平，书中错误与不足之处在所难免，望广大读者批评指正．

本书是在学校资助之下才得以出版，在此表示感谢．

山西大学　田怀录

1991 年 6 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第一章　结构程序设计原理

本章讲述的结构程序设计方法，是指导程序设计的基本方法，是最成熟、最实用的方法，作为一名程序设计人员应该掌握.

随着计算机的发展，自60年代末到70年代初，开始出现大型软件系统如操作系统、数据库管理系统，人们再用过去那种工匠技艺式方法来编制这些程序已是不可能了. 事实正是如此，一个大型软件往往需要花费几千人年的工作量，不但工作效率低而且可靠性差，又难以维护，这就是所谓的"软件危机". 这个时候程序的主要缺点是不清晰，几乎无层次无结构，阅读程序时常常由这一页一下就需要向前跳几页或向后跳几页，弄得人眼花缭乱.

1965年荷兰Eindhoven大学的E.W.Dijkstra教授首先倡导用结构的方式构造程序. 他把程序"灾难"归之为GOTO语句的使用. 自此人们开始认识到软件系统不是工艺产品，而是工程、是科学，也应像社会及一切工程那样分层次、有结构，这才易建造、易了解、易维护，从而也才能减少错误. 然而程序如何分层次，程序的结构又是什么？这个问题的解决于1966年由Bohm和Jacopini发表的一篇文章才有了新的进展. 他们论证了用三个基本控制结构——顺序结构、选择结构和循环结构就足以表示各种程序. 后来经过十多年的争论、研究以及结构性语言的提出与应用，于70年代末80年代初结构程序设计的方法才完全成熟，也才为人们广泛使用.

结构程序设计是否就是解决软件困难的万能良方呢？结构程序设计是否就是我们的全部目的呢？当然不是！今天人们讨论了程序设计的主要目标应该是正确、可靠、实用、安全而高效率；易用、易读、易维护、易调试测试，研制维护费用少；程序能查错、抗干扰，被破坏后能恢复；尤其在人机对话的情况下要有良好的人机界面.上述目标的实现，关键是结构化程序设计.

结构程序设计不是目的而是手段. 当结构程序使程序过于复杂的时候，也可以在某些地方放弃这种手段(如采用GOTO).这就把结构程序设计的讨论引向正确方向，放在了恰当的位置.

正如事实所表明的，结构程序设计对编制正确、易读、易维护的程序，确实起到很大作用. 后来人们研究了程序证明理论，从理论上证明程序是否能正确达到预期的目标，只要程序的前提条件能够用形式逻辑公式表示. 当然，正如人们由于考虑不周而使程序有错误一样，程序证明也会由于考虑不周而不正确. 由于篇幅所限，程序证明本书不去涉及.

程序的查错、抗干扰能力的恢复，以及良好的人机界面的形成，这些只能在程序的内容上去解决，程序的结构形式是无济于事的.

§1.1 结 构 程 序

1.1.1 程序的三个基本控制结构

后面用于表示程序结构的代码或符号是自己规定的，它类似于一切语言但不同于任何语言，这在讲抽象方法时大多如此. Bohm 和 Jacopini 提出的三个基本控制结构是：

1. 顺序结构　顺序结构就是语句序列

$S_1; S_2; \cdots; S_n$

S_i 是程序语句、程序段或过程，分隔符";"表示顺序连接．其含意是：仅当前一语句(段)执行完后才执行后一句(段)．顺序结构的框图见图 1.1(a)．

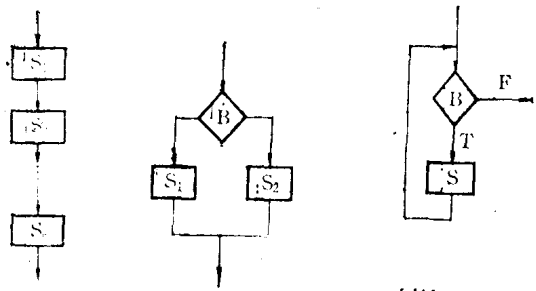

图 1.1

2. 选择结构　其表达形式为

IF B → S_1

ㄱB → S_2

ENDIF

其意义是：检查布尔式 B 的值，当 B 为真时，则执行语句段 S_1；否则(B 为假)，执行语句段 S_2．不论执行哪一语句段，执行完后，控制都达同一出口 ENDIF 处，执行 ENDIF 后面的语句．其框图见图 1.1(b)．当然 S_1, S_2 可能有一个为空语句，如 S_2 为空，可表示为

IF B → S_1

ENDIF

意即 B 为真时执行 S_1，否则什么也不作，控制亦转到 ENDIF 处，执行 ENDIF 后面的语句．其框图见图 1.2．

此为试读，需要完整PDF请访问：www.ertongbook.com

例 IF $a \leqslant b \rightarrow x := a$

$a > b \rightarrow x := b$

ENDIF $\{x = \min\{a, b\}\}$

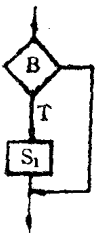

图 1.2

IF 语句的扩充 CASE 语句：

CASE

$B_1 \rightarrow S_1$

$B_2 \rightarrow S_2$

……

$B_n \rightarrow S_n$

$BB \rightarrow S$

ENDCASE

其含意是：布尔式 B_1 的值为真时，执行语句段 S_1；B_2 为真时执行语句段 S_2；…；B_n 为真时执行语句段 S_n；当 $B_1, B_2, \cdots, B_n$ 都假时执行语句段 S. 当然 $B_1, B_2, \cdots, B_n$ 只能有一个真，或都假. 这种语句自然可以用 IF 语句的多层嵌套而得. 如

CASE

$B_1 \rightarrow S_1$

$B_2 \rightarrow S_2$

$BB \rightarrow S$

ENDCASE

其框图见图 1.3. CASE 语句相当于

IF $B_1 \rightarrow S_1$

$\nearrow B_1 \rightarrow$ IF $B_2 \rightarrow S_2$

$\nearrow B_2 \rightarrow S$

ENDIF

ENDIF

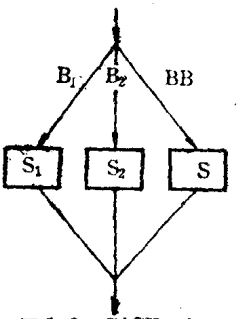

图 1.3 CASE语句图

3. 循环结构 其表达式为

WHILE B

S

ENDWHILE

含意是：当条件 B 取真值时执行语句段 S(循环体)；而后再计算 B 的值，若仍为真则再执行 S，如此返复直到 B 值为假时停止执行 S. 控制达 ENDWHILE，执行 ENDWHILE 后面的语句.

例 计算自然数的阶乘.

```
u:=n;z:=1;
WHILE u>0
z:=z*u;
u:=u-1
ENDWHILE {z=n!}
```

此外，常见的 FOR 语句及 REPEAT 语句也都可以用此 WHILE 语句表示：FOR 语句只不过是预先知道循环次数的 WHILE 语句；REPEAT 语句是先执行一次循环体，而后再执行一个其循环条件与循环体和 REPEAT 语句全同的 WHILE 语句.

从上述可知，任何一个程序基本控制结构都只有一个入口和一个出口，这是一个重要特性. 这使我们在设计程序时只要程序接口(出入口)的条件不变，中间的程序段如何改变也不影响程序的执行结果. 这就使我们能很方便地用更好的程序段去代替原来的程序段.

1.1.2 结构程序及其表示

结构程序可以这样定义：其控制流只用三个基本控制结构——顺序、选择和循环来描述.

这句话的含义是：从一个控制语句看，程序所用的结构

是这三个结构之一，从一个程序块看，整个程序是这三个结构有限次地嵌套而成.这就自然得出结构程序的一个重要特点：任何大小的程序段都是单入口单出口的. 这种程序的伪码(pseudocode)表示如下：

```
S1;
IF B1 → IF B2 → WHILE B3
                        S4;
                        IF B4 → S5
                         ∕B4 → S6
                        ENDIF
                  ENDWHILE
          ∕B2 → S3
          ENDIF
   ∕B1 → S2;
          WHILE B5
              S7
          ENDWHILE
ENDIF;
S8
```

这就是所谓缩进排表示法：语句的正文与语句，以及一种结构嵌套到另一结构内，在书写时一定要向右缩进至少一格。同在一个层次的语句排列也要一致. 这样排列只增加程序的清晰度而毫无语义之意. 此程序的流程图见图 1.4.

从图 1.4(a)看出，从最高层看此程序是个顺序结构：S_1; P; S_8[见图1.4(b)]，其中 P[图(a)中用虚线框起来部分]是个选择结构，它的两个分支是 P_1, P_2. 而 P_1 又是个选择结构，它的两个分支是 S_3 与 P_{11}. P_{11} 是个循环结构，循环体是 P_{111}. P_{111} 又是个顺序结构，它是由程序段 S_4 与一个选择结构连接

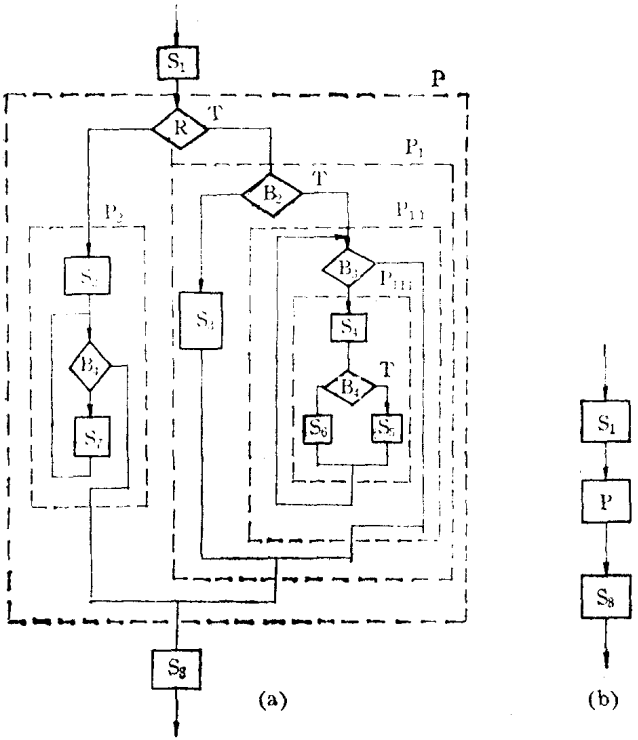

图 1.4　程序流程图

而成．而 P_2 亦是顺序结构，它是由程序段 S_2 及一个循环结构连接而成．由此可知整个程序都是由这三种基本结构嵌套而成的．而且任一级的结构都只有一个入口一个出口．

那么 $S_1, S_2, \cdots, S_8$ 又是什么结构呢？一般说来它可能同样是由这三种基本结构嵌套而成，或者只是一个赋值语句(或称原子语句)．

谈了结构程序，但什么是非结构的程序呢？是不符合上述要求的程序自然非结构的．具体说来，如果程序中用了任意转向的 GOTO 语句就是非结构的，或结构不好的．因为此时控制可以由一个结构之外不经正常入口直接转向内，亦

可由内不经正常出口直接转向外；也可由程序前一部分转向后一部分，也可由后一部分转向前一部分(因而称GOTO为非正常出口语句)．这样无规则转来转去的程序自然难读、也易出错，改错或扩充都很困难，既看不出什么结构，也破坏了单入口单出口的原则．所以结构程序设计Pascal语言的一些版本就把GOTO语句排斥在外．即使允许使用也加以限制，而且声明：依此限制使用GOTO语句设计的程序，完全可以改变成不用GOTO语句而只用三种基本控制结构的程序．这种限制就是：只许在一层内转移，或由内层转到外层，不许从外层转向内层，更不许由过程外转入过程内．在这种限制下使用GOTO的程序仍称之为结构程序．GOTO语如句此不好，为什么还要用呢？这是因为有时候在受限制之下使用一个GOTO语句会给程序设计带来方便，而且使程序简单．当然这种使用是很少的．此时如果不使用GOTO语句就需要另设新的变量，对此这里不再讲述，待讲述dBASE Ⅲ的类似情况时我们再详细叙述．

1.1.3 dBASE Ⅲ 程序控制结构

dBASE Ⅲ全部指令可分两类：一类是对文件及外部设备的操作，以及人机对话指令；另一类就是程序控制指令(或结构)．全部程序控制结构亦为顺序、选择、循环以及非正常出口指令．

一、顺序结构

其表示法是

S_1

S_2

⋮

S_n

这里 $S_1, S_2, \cdots, S_n$ 是命令(或指令、语句)、程序段或过程. 其含意与基本控制结构中的顺序结构全同. 这里顺序段之间不加分隔符";". dBASE 的分隔符是一条指令的移行标志.

二、选择结构

dBASE Ⅲ 同样提供了 IF 结构及其扩充 CASE 结构.

IF 结构为

```
IF <条件>
    <命令>
[ELSE]
    <命令>
ENDIF
```

CASE 结构为

```
DO CASE
CASE<条件>
      <命令>
[CASE<条件>]
      <命令>
[OTHERWISE]
      <命令>
ENDCASE
```

这些命令的含意与基本结构中讲的相同.

三、循环结构

句法为

```
DO WHILE<条件>
          <命令>
ENDDO
```

其含意亦与前面定义的全同,不再多述.

上述三种结构 dBASE Ⅲ 亦允许相互嵌套使用.

此为试读,需要完整PDF请访问: www.ertongbook.com