

APPLE II

程序设计及磁盘操作系统

冯 庆 慈 编

广州地区高等学校科学技术开发中心

广 东 工 学 院

APPLE II 程序设计及磁盘操作系统

冯庆慈 编

一九八四年元月

前 言

APPLE II 是目前世界上最流行的微型计算机之一。它具有使用方便、工作可靠，价格便宜等一系列优点。我国目前已进口了不少APPLE II 微型计算机系统，并且国内也有不少厂家组装APPLE II 微型计算机。

为了加速我省省属高等院校计算机的广泛应用、提高教学、科研和管理水平。广东省高等教育局和广州地区高等学校科学技术开发中心决定举办一系列微型计算机培训班；并且委托广东工学院负责具体培训工作。以对省属院校有关的教学、科研、管理和实验人员进行全面的培训。

这本书是作为微型计算机培训班的教材而编写的。适用于微型计算机培训中的提高班。因此，要求读者有一定的微型计算机基础知识、或者曾参加过微型计算机的普及班或一般班的学习。

本书着重介绍利用APPLESOFT进行程序设计的方法。

首先介绍程序设计的一般方法和步骤，然后介绍程序的几种基本结构形式（顺序、分支、循环和子程序）的设计方法，最后介绍磁盘操作系统的各种命令及利用这些命令进行管理程序的设计方法。

为使学习更具体，在介绍每种设计方法时，都结合工程计算上或生产管理上的实例进行。并选择了一些工程计算中常常遇到的计算问题。如方程求根、数值积分、线性方程组求解、曲线拟合等系统地介绍有关的算法和相应的程序。

为使过去未接触过APPLE II 微型机的同志能更快地熟悉APPLE II 微型机的使用，本书对APPLESOFT特有的语句和命令作较详尽的解释，并在附录中列出了APPLE-SOFT的全部命令和语句，供同志们查阅。

本教材是在较短时间内编写出来的，时间较仓促，加上编者水平有限，因而很可能存在各种谬误和缺点。希望读者多提宝贵意见，以求不断改进。

在编写本书的过程中，得到广东省高教局、广州地区高等学校科学技术开发中心有关同志的大力支持，并得到我院不少同志的支持和帮助。特别是我院计算中心的李怀香、刘桂云老师，对教材的编写提出了不少有益的建议。为了使本书能及时排印，为本教材绘制了大量插图。

佛冈县印刷厂在本书的印刷过程中，给予大力协助。谨此一并致谢。

广东工学院计算中心 冯庆慈

一九八四年一月

目 录

第一章 概述.....	(1)
第一节 程序设计的基本步骤.....	(1)
第二节 流程图.....	(2)
第二章 程序设计的基本方法.....	(6)
第一节 顺序程序的设计.....	(6)
第二节 分支程序的设计.....	(10)
第三节 循环程序的设计.....	(14)
第四节 子程序和自定义函数.....	(22)
第三章 程序设计中的一技巧.....	(27)
第一节 函数的运用.....	(27)
第二节 数组的运用.....	(34)
第四章 图形的输出.....	(44)
第一节 屏幕的显示方式.....	(44)
第二节 低分辨率图象的显示.....	(46)
第三节 高分辨度图象显示.....	(48)
第四节 动画绘制.....	(54)
第五章 数值积分.....	(60)
第一节 矩形法和梯形法.....	(60)
第二节 辛卜生法.....	(64)
第三节 变步长的求积计算.....	(68)
第六章 方程求根.....	(73)
第一节 二分法.....	(73)
第二节 迭代法.....	(75)

第三节	切线法	(80)
第四节	弦截法	(82)
第七章	线性方程组的数值解法	(85)
第一节	线性方程组的直接解法	(85)
第二节	线性方程组的迭代解法	(95)
第八章	曲线拟合与经验公式	(101)
第一节	回归直线	(101)
第二节	曲线拟合	(107)
第九章	磁盘操作系统	(113)
第一节	引 论	(113)
第二节	DOS的命令分类	(115)
第三节	DOS的管理命令	(116)
第四节	存取类命令	(122)
第五节	顺序文本文件命令	(123)
第六节	随机文本文件命令	(129)
第七节	机器语言文件命令	(130)
第十章	系统设计的一个例子	(131)
第一节	系统的功能	(131)
第二节	文本文件的组织	(133)
第三节	模块设计	(134)
附 录	APPLESOFT语句、命令、函数一览表	(140)

第一章 概 述

第一节 程序设计的基本步骤

利用数字计算机解决实际问题，通常要经过如下几个步骤。

一 建立数学模型

利用计算机解决问题，首先要对问题进行充分的了解，对研究的对象数量化。确定什么是待求的未知量，什么是已知的常量或可测定的参量。这些量之间具有怎样的关系（等量关系，函数关系，微分关系……）在了解问题、分析问题时既要注意全面又要分清主次，突出主要问题，略去次要因素，对问题进行适当的简化，以便形成一个符合实际但又相对简洁的数学模型。

二 选择计算方法

确定数学模型以后，就应该根据模型的类型选择相应的数值解法。有关这方面的内容，可参阅专门的“计算方法”书刊。本书的第五—八章将介绍这方面的一些基本内容。

三 画出总框图

选定计算方案以后，就应着眼于方案的实现。一般地说，由于模型的复杂性，往往把整个问题分解为若干个功能较简单的子问题。为了反映各个子问题之间的相互关系，往往画出一个总框图，用以形象地描述各个子问题的相互关系，表达各个子问题的作用，并明确地反映求解过程的顺序和进程。

四 编写程序

总框图确定后,就可以逐个编写求解子问题的程序。在编写时首先画出求解各子问题的细框图,这种细框图是用以表达程序进程的一串有向图,用以反映子问题的每一细节。这种有向图采用不同的几何图形表示不同的程序功能(见下节),因而根据细框图,可以相应地写出程序。

五 程序调试

程序编写完毕以后,应对程序进行调试。调试时,可输入一些已知答案的参量让程序进行运算,看看结果是否相符。调试时可先调试子问题程序。调好后再调试整个程序。同时还应注意在特殊情况下使用程序以检查程序的适应性和通用性。

六 上机运行

程序调试通过后,就可以上机运行,输入已知原始数据,然后在计算机的输出机构中取得结果。通常,一个复杂的问题,上机运行后还可能有错误,这或者由于程序编写考虑不全面,或者由于原来建立的数学模型不反映问题的本质,这时就应该作出相应的修正。

综上所述,程序设计的过程,可以概括地表示如下:

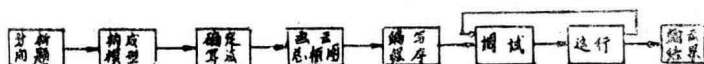


图 1 — 1

第二节 流程图

在上一节中,我们已经指出,为了便于分析,常常将一个复杂问题分解为若干子问题,并借助总框图表示它们之间的相互关系。在一个子问题中,为了便于编写程序,借助细框图来形象地表示该子问题程序的基本结构。总框图、细框图都是一种流程图。流程图是用各种几何图形和文字说明来直观地描述计算过程的有向图。它的优点是形象、直观,既能表达程序的设计思想,又便于检查、修改、阅读程序。因而它被广泛地采用于各种高级语言的程序设计中。

一 基本的流程图符号

流程图符号分成如下两类：

1 流线

流程图用以表示程序处理的前后关系，因此，必须用流线表示程序的走向，在流程图中流线用带箭头的折线来表示，箭头的指向，表示程序执行的方向。

2 框图

根据不同的程序功能，采用不同的几何图形分别表示各种程序功能，它分为下列几种：

①输入输出框 用以表示程序的原始数据输入，或将计算结果打印输出，其形状如下图所示。分别与读入卡片纸和输出打印纸相似，在图形内填写上相应的输入和输出的内容，输入输出框各有一个流线入口，一个流线出口。

②叙述框 它用以表示程序各种基本操作，例如赋值、运算等，叙述框用矩形方框表示，并在框内写上实现的操作功能。叙述框有一个流线入口，一个流线出口。

③判断框 它的作用是根据程序执行的情况来判断程序下一步的走向，一般用菱形框来表示，在框内填入用作判断的条件，它有一个入口，二个或多个出口，通常用于分支程序。

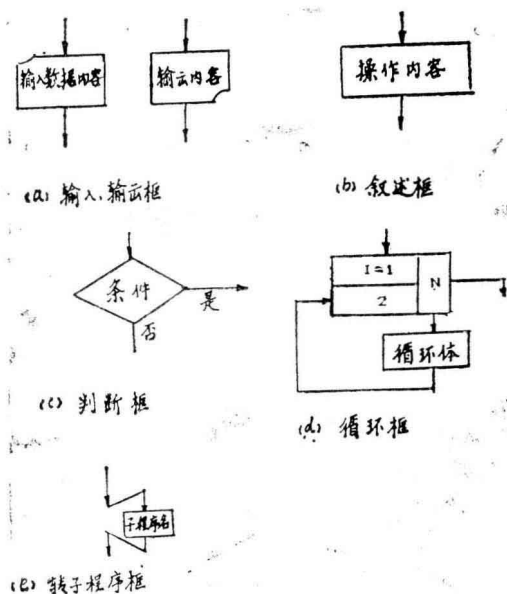


图 1—2 流程图

④循环框 它的作用用以表示一个循环程序段（见图 1—1d）。在循环框 的顶部填入循环变量的名字、初值、终值和步长，下面连接每次循环执行的循环体。每个循环框有一个入口，一个出口。

⑤转子程序框 它的作用用以表示一个转子程序的执行，在转子程序框中填上子程序名，它有一个入口，一个出口。

二 流程图的例子

例 1、确定求A, B, C中极大值的流程图

- 算法： 1 输入A, B, C
2 若 $A > B$ ，且 $A > C$ 时，输出A。
3 若 $B > A$ ，且 $B > C$ 时，输出B。
4 若 $C \geq A$ ，且 $C \geq B$ 时，输出C。

画出框图

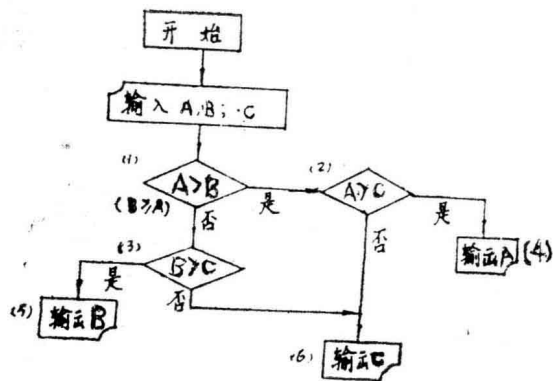


图 1—3

执行时：若A最大，流程将按（1）（2）（4）框的顺序执行。

若B为最大，流程将按（1）（3）（5）的顺序执行。

若C为最大，流程将按（1）（3）（6）或（1）（2）（6）的顺序执行。

例 2

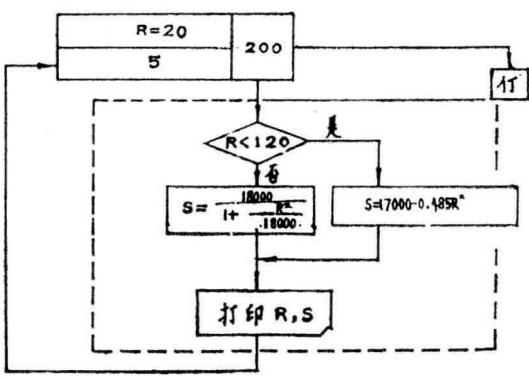


图 1—4

表示重复执行虚线围住的循环体。开始时R取值20，以后每次加5，直到R超过200时停止，实际上是求如下函数在区间〔20，200〕上的函数值，要求自变量每隔5计算一个值。

$$f(R) = \begin{cases} 17000 - 0.485R^2 & R < 120 \\ \frac{18000}{1 + \frac{R^2}{18000}} & R \geq 120 \end{cases}$$

第二章 程序设计的基本方法

任一程序不管怎样复杂，都是由以下四种基本程序结构组合而成。这四种基本结构就是顺序、分支、循环和子程序（包括自定义函数）。在这一章中我们分述一下这几种结构的设计方法。并通过例子说明一些设计中应该注意的问题。

第一节 顺序程序的设计

所谓顺序程序是指语句执行时，是按语句的行号大小顺序依次执行。这种程序的设计只须按照计算步骤的先后次序，用语句把计算公式表达出来。一般使用于计算一些烦琐的计算公式。其编写方式通常按输入已知数据，运算，输出结果的顺序编写。编写时主要使用数据输入语句、赋值语句（即INPUT语句、READ、DATA语句、RESTORE语句、LET语句）和PRINT语句。

下面通过一个例子，说明这种程序的设计方法。

例1 已知如下一个并联电路，试编写一个计算电流I的程序。

由电学并联电路计算公式，可知

$$\frac{1}{R_{\text{并}}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}$$

$$I = U/R_{\text{并}}$$

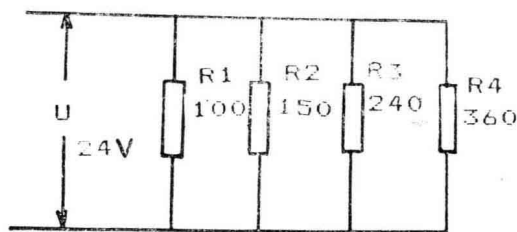


图 2—1

于是，我们可设计程序如下

```
10    R1=100
20    R2=150
30    R3=240
40    R4=360
50    U=24
60    I=U*( 1/R 1 + 1 /R2 + 1 /R3 + 1 /R4 )    计算
70    PRINT "I=" ; I    输出结果
```

从计算这个例题的要求来看，上述程序是正确的，但是如果上述电路中电阻的阻值分别改变为 $R1=50\Omega$ ， $R2=110\Omega$ ， $R3=200\Omega$ ， $R4=30\Omega$ ，电压 U 的值也改为 $U=20V$ ，那么，我们就要把上述程序的10—50行改为另一组赋值语句。这表明上述程序的通用性很差。如果把程序中数据输入部份改为键盘输入方式（如下程序2）。

```
10    INPUT "R 1 , R 2 , R 3 , R 4 , U=? " ; R 1 , R 2 , R 3 , R 4 , U
20    I=U*( 1/R1+ 1 /R 2 +1/R 3 +1/R 4 )
30    PRINT "I=" ; I
```

每次运行时，只须从键盘上打入不同的电阻和电压值，就可求出正确的结果而无须改动程序。程序的通用性就强多了，但是，每计算一次，计算机都要停机一次，等待用户输入，这样一来，如果需要计算一批参数不同的电路时，等待输入的机时就比较长，如果把程序改为下述方式：

```
10    READ    R1, R2, R3, R4, U
20    I=U*( 1/R1+1/R2+1/R3+1/R4 )
30    PRINT "I=" ; I
40    GOTO    10
50    DATA    100, 150, 240, 360, 24, 50, 110, 200, 30, 20
```

那么，程序每次从DATA语句中读入数据，计算后输出结果，又从DATA语句读入下一组数据，如此重复上述步骤直到数据读完为止。这就大大减少了等待输入的机时，当然这里使用了GOTO语句，已经改变了运行的次序，实际上已不是一个顺序程序了，这里，主要想说明三种数据输入语句的一些区别，它们虽然都有给变量赋值的作用，但各有优缺点。下面列表加以比较，供大家参考。

LET	READ/DATA	INPUT
1、都可以给变量赋值		
2、每一语句只能对一个变量赋值	每一语句可同时对多个变量赋值	每一语句可同时对多个变量赋值
3、对变量输入不同数值时，要改变程序语句	对变量输入不同数值时，只需改变DATA语句	对输入变量不同数值时，无需改变程序
4、占用机时少	同左	运行时要等待输入占机时长
5、可以进行运算，然后将结果赋值给变量	不能进行运算，DATA语句中不准出现变量或表达式	不能进行运算，从键盘中只能输入数值
适合输入计算过程中不变动的参数，或用作记录运算的结果	适用于输入数据较多的程序，作输入数据之用	适用于参数变化，需要输入多组参数并比较其运算结果的程序作输入参数之用

表 2—1

在上述例子中，提到一个“通用性”问题，在程序设计中是经常要考虑的原则，设计一个程序不能只适用于某一个具体计算。它应该适用于一类计算，这时就要考虑各种各样的可能情况，光用顺序程序设计方法就不够了。

例 2 设计一个求解二次方程 $ax^2+bx+c=0$ 的程序。

由二次方程求根公式，我们知道，方程的两个根是

$$X_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$X_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

我们可以把这个求解过程写成程序

```

10  INPUT "A, B, C=? "; A, B, C
20  X1 = ( -B + SQR ( B ^ 2 - 4 * A * C ) ) / 2 / A
30  X2 = ( -B - SQR ( B ^ 2 - 4 * A * C ) ) / 2 / A
40  PRINT "X1=" ; X1, "X2=" ; X2
50  END

```

上述程序在运行时,当输入A, B, C的值后,计算机即输出解 X_1 和 X_2 的值。例如当屏幕询问A, B, C=? 时,如果回答以 1, 2, -3 RETURN 屏幕上就给出方程的根 $X_1=1$ $X_2=-3$

回答以另一组值10, 72, -91.3时(这相当于求解方程 $10x^2+72x-91.3=0$)屏幕上就给出方程的根 $X_1=1.1$ $X_2=-8.3$

但当输入数值 1, 2, 3 时(这相当于求解 $X^2+2X+3=0$), 屏幕上并不显示出解的值, 而显示出 **ILLEGAL FUNCTION CALL IN 20** 而停机。为什么会出现这种情况呢? 原来在A=1, B=2, C=3 时, $B^2-4AC=-8$, 因而进行运算 $SQR(B^2-4*A*C)$ 时, 出现非法调用函数的错误。这种情况的出现是由于我们在设计时, 只考虑了 b^2-4ac 非负的情况, 当 $b^2-4ac<0$ 时, 如果由人来计算, 知道 $\sqrt{b^2-4ac}=\sqrt{4ac-b^2} \cdot i$ (这里i为虚数单位), 但计算机并不会自动掌握这个知识, 这就要求我们在设计中把这种情况考虑进去, 教会计算机先进行判别, 然后选用不同的算法, 这时我们就需要使用分支程序的设计方法了(见下节)。

最后我们要指出APPLE II 机的输出语句中一些应该注意的问题。大家知道输出语句有二种格式, 即紧凑打印和分区打印二种方式。在语句书写格式中, 前者用分号“;”分隔各打印项, 后者用逗号“, ”分隔各打印项输出时, 前者各打印项, 一个连一个打印在一起, 中间不留任何空白; 而分区打印, 则把各打印项按序打印在各分区上(见图2-2)。

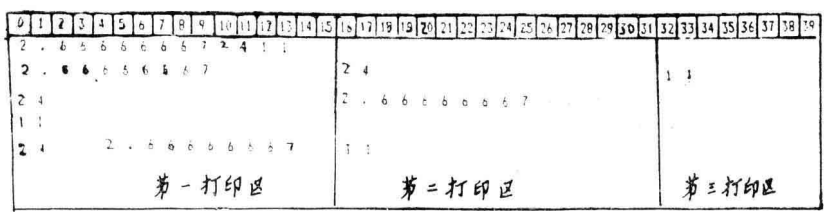


图 2-2

例如PRINT 8/3; 8*3; 8+3在屏幕上将显示出上图第一行的字样, 三个结果连在一起。若采用分区打印格式PRINT 8/3, 8*3, 8+3则屏幕上显示出第二行打印字样, 这时三个结果分别打印在不同区上。但是如果把上述打印项中的第一, 二项交换, 执行PRINT 8*3, 8/3, 8+3屏幕上三个结果分别打印在第一区, 第二区, 下一行的第一区中, 如图上3, 4行所示。这是什么原因呢?

原来APPLE II 的分区输出打印格式有如下的规定, 即打印第二区时仅在第15格为空时才能执行, 打印第三区时则要求第23格—31格均为空, 上述语句中当在第二区打印8/3时其长度为10个字符, 第23、24、25格已印有字符6、6和7, 因而不能在第三区打印, 从而将8+3移到下一行的第一区打印。这说明如果想按一、二、三区输出计算结

果，必须控制第二区的打印项长度不超过 7 位。

如果不用分区打印而又想隔开各打印项，可以使用TAB函数和SPC函数，即在各打印项前加入TAB (n) 或SPC (n) 其中n为数字。

例如 PRINT 8*3; SPC (2) 8/3; SPC (2) 8 + 3
 PRINT 8*3; TAB (4) 8/3; TAB (16) 8 + 3

那么执行的结果都是在屏幕上打印出图 2—2 上第 5 行的 字样。SPC (n) 是使打印光标移动n个格的意思，因此打印24后，移动 2 个空格再打印2.66666667，又空 2 格再打印11，即打印出第 5 行的样子。TAB (n) 是使打印光标移到离该行第 1 格n个 空格 处开始打印，即从第 4 格开始打印2.66666667，从第16格开始打印11，这效果与上一语句一样，从而打印出同样的格式。

输入输出语句虽然不影响计算的结果值，但在程序设计中，总希望输出一个合意的打印格式，这就要求灵活使用上述语句。在应用中，输出格式的设计成为设计的一个组成部份。

第二节 分支程序的设计

像上述 § 6,1 例 2 中指出的那样，在程序设计中，常常要判断某些条件是否满足，然后选用不同的计算公式。在程序设计时，就要根据条件是否满足转移到不同的程序段。像上述的例子中，就要根据 $b^2 - 4ac$ 的值是否 < 0 转入不同的程序段，这时可用图 2—3 的流程图来表示这个过程。

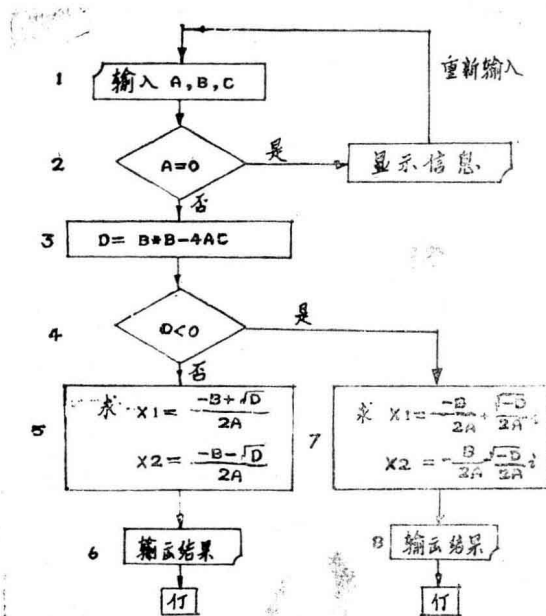


图 2—3

图中第二框, 设置了 $A=0$ 的判断框, 是为了保证第5, 第7框中除法运算的可行性。事实上, 当 $A=0$ 时, 方程已不是二次方程, 因而采用重新输入方程系数的措施。增加这一回路是为了防止一些偶然的误输入。

在流程图中框2和框4的输出口都不止一个, 就像大路上出现分支的样子, 所以这种程序叫做分支程序。

实现分支程序主要使用各种转移语句。如GOTO语句, IF—THEN语句, ON—GOTO语句, ON—GOSUB语句, ONERR—GOTO语句。下面通过一些例子说明这类程序的设计方法。

例1 写出实现图2—3流程图的程序。

这只需按流程的顺序书写相应的语句即可, 程序如下:

```
10  INPUT "A, B, C=" ; A, B, C
20  IF A=0 THEN PRINT "ENTER ERROR! REENTER" ;
    GOTO 10
30  D=B*B-4*A*C
40  IF D<0 THEN 90
50  X1=(-B+SQR(D))/2/A
60  X2=(-B-SQR(D))/2/A
70  PRINT "X1=" ; X1, "X2=" ; X2
80  END
90  R=-B/2/A
100 I=SQR(-D)/2/A
110 PRINT "X1=" ; R, "+" ; I, "I"
120 PRINT "X2=" ; R, "-" ; I, "I"
130 END
```

例2 设计一个百货公司收费计算的程序。

程序要求营业员顺序输入某顾客选购的商品、单价、数量后, 要求计算机马上在屏幕上显现出来, 并同时显示出该货物的价格, 直到该顾客选购的全部货物已全部输入后, 计算机要显示出总收费。

剖析: 输入单价、数量、计算出价格这些都是很容易实现的。要进行累计也不难。可在开始计算时, 设置一个变量T, 并令 $T=0$, 然后每次向T累加求出的某类商品的价格, 即组成如下的流程图(图2—4左)。

存在的问题是怎样让计算机知道累计结束, 由于各个顾客选购商品有多有少, 不能事先规定统计多少次结束, 为此, 可用设定结束标志的方法来解决这个问题。例如约定 $P=0$ 作为某次计算结束的标志。当某顾客的货物已全部累计完毕后, 可以输入 $P=0, N=0$, 告诉计算机累计已结束, 计算机即输出累计值。在程序中只须在输入框(2)后

增加一个判断框(见图2—4右),当 $P=0$ 时,输出累计值,并开始另一次计算。

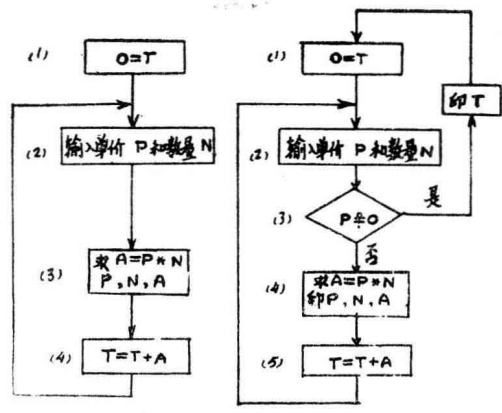


图 2—4

相应程序如下:

```

10   T = 0
20   INPUT "P, N=" ; P, N
30   IF P = 0 THEN PRINT T; GOTO 10
40   A = P * N
50   PRINT P, N, A
60   T = T + A
70   GOTO 20
    
```

例 3 下面30个数据是某班学生数学科期中测验的成绩,如果认为90分以上作为优秀,75—89分作为良好,60—74分作为及格,60分以下作为不及格,试设计一个分别累计四类成绩的学生人数,并计算各类学生占全班人数的百分比。

62, 78, 85, 72, 68, 88, 92, 77, 83, 66, 95, 100, 73, 62, 64, 79, 87, 52, 65, 58, 74, 68, 76, 64, 87, 93, 94, 60, 79, 42。

在实际的累计过程中,工作人员对每个成绩实际上经过分类和累加这二个步骤,即看到成绩后,先判断它属那一类,然后就在那一类的累计数上添上一,直到全部成绩都分析完为止。