DJS—21计算机编译程序的改进

数学力学系计算数学专业软件组

在DJS—21计算机(以下简称121机)上使用算法语言或汇编语言算题时，经常需要对源程序或数据进行修改。121机的ALGOL—60编译系统虽具修改功能，但在使用过程中尚有欠缺之处，如为了修改某行中的个别字符或加入几个字符，都必须在修改信息中打出整行的内容；要删去连续的几行，也必须逐行删除……。这样的修改方式要求输入一些多余的信息，做了一些不必要的操作，且易带来新的错误，降低上机效率。

针对上述情况，我们对原来的修改子程序作了一些改进。除用消错符消错外，源程序和数据的增删和改换均通过此程序来实现。修改指示可以穿孔在纸带上由光电机输入，也可以在上机时直接用电传机打入。只要使用格式符合，改进后的修改程序适用于121机上使用的其他语言。

(一) 修改指示

修改指示指明具体的修改内容，书写方式用语法公式表示：

‹修改指示›::=‹GY|GS›{‹修改元›}^{48}，

‹修改元›::=‹行›(‹字符串›)|‹行›:‹列›(‹字符串›)|
‹行›:‹列›—‹行›:‹列›(‹字符串›)

‹行›::=‹无符整数›

‹列›::=‹无符整数›

‹无符整数›::=‹数字›|‹无符整数›‹数字›

‹数字›::=0|1|2|3|4|5|6|7|8|9

‹字符串›::=‹字符›|‹字符串›‹字符›

‹字符›见55型电传机字符表。

其中{‹修改元›}^{48}，表示允许有一至四十八个‹修改元›。

修改元允许三种形式，其中括号(……)内的字符串是替换内容，可以为空；也可用方括号〔……〕代替，以便进行括号不配对的修改。

第一种修改元中的‹行›是要删去的某行序号。例如：

5(……)

表示把第五行删去。换以(……)中的字符串。

第二种修改元中的<行>:<列>指示插入位置。例如：

3:7(……)

表示在第三行第七个字符前插入(……)中的字符串。

第三种修改元中，前一个<行>:<列>是删去的字符串的起点，应写为要删去的一个字符串头一个字符的行列位置；后一个<行>:<列>表示终点，应写为要删去的字符串后面第一个要保留的字符的行列位置。例如：

3:2——4:6(……)

表示从第三行第二个字符起到第四行第五个字符均删去，然后换以(……)中字符串。

修改元允许：

1、<列>为1时，可连同前面的":"一起省去，例如：

3:1——5:1　　可写为　3——5

2、终点的<行>数与起点<行>相同时，可省去。例如：

2:3——2:7——可写为　2:3——:7

特别的，3:7——3:7与3:7指示同样的插入位置；5(……)、5:1—6:1(……)与5—6(……)都同样表示换掉第五行。

字符的列应按消错符消错之后计算，且空格不计。

GY(GS)引入源程序(数据)的修改指示。一个修改指示内最多允许四十八个修改元。每个修改指示后均需打五个以上字母键，以示结束。

在同一次输入中，GY最多允许十个，GS最多允许五个，同类修改指示按后进先处理的原则进行修改。每个修改指示按前次修改后的结果计算行列。因此，同类修改指示的顺序是不能颠倒的。例如一次有三个GY：

GY{<修改元>}^{N1}五个字母键GY{<修改元>}^{N2}五个字母键Y五个字母键GY{<修改元>}^{N3}五个字母键J

这时，修改程序是这样工作的：首先按最后一个修改指示的N3个修改元对Y进行修改，得到Y1；其次按第二个修改指示的N2个修改元对Y1进行修改，得到Y2；然后按最前面的修改指示的N1个修改元对Y2进行修改，得到Y3，其中$Ni \leqslant 48$，$i=1,2,3$。

如果在同一个修改指示中，有某两个修改元的删去字符串有公共部份，则以后者为准，前者无效。

输入时，除同类修改指示有先后之外，Y,S,GY,GS之间的顺序可任意。

(二) 修改过程输出的信息

在正常情况下，本程序按修改指示进行修改，不输出任何信息。只是在异常情

况下才有可能输出如下信息：

(1)G－DUO表示GY多于十个，或GS多于五个，或某修改指示的修改元超过48个，或修改时内存满。

(2)G－CUO序号，表示序号所示修改元格式不对，或前一修改元替换内容中多了闭括号。

(3)G－BPD序号，表示序号所示修改元的替换内容中多了开括号。

以上三种情况发生后，均不进行修改，转到等待命令状态。

(4)G－CHUN表示有一个修改元的删去字符串与前面某修改元重了，此时将前一修改元作废，继续修改。

(三) 实 现

修改任务由“修改记鼓”、“修改1”和“修改”三个子程序配合完成。前者负责修改时内存与磁鼓之间的调度，在需要修改时调用后者。后两个进行具体的修改工作。

1、修改记鼓

在输入时（光电或电传）遇到五个字母键后面的结束标志J，或者电传命令K均调用本程序一次。

本程序边记鼓边检查是否需要修改，需要修改时，做好内存布局的准备，然后调用“修改1”子程序进行修改，改完后记鼓；不需要修改时，记鼓后便返回。

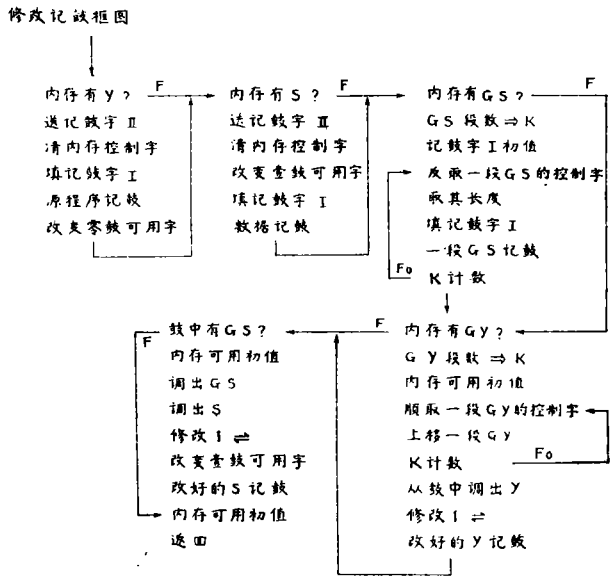

图 1

2、修　改 1

每类修改调用本程序一次。其功能是按后进先处理的原则提供一个相应的控制字，调用修改子程序进行修改，直到这一类修改指示全部执行完后便返回。

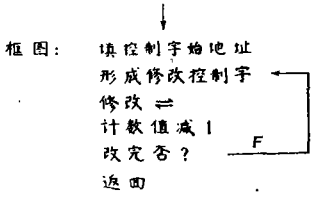

图　2

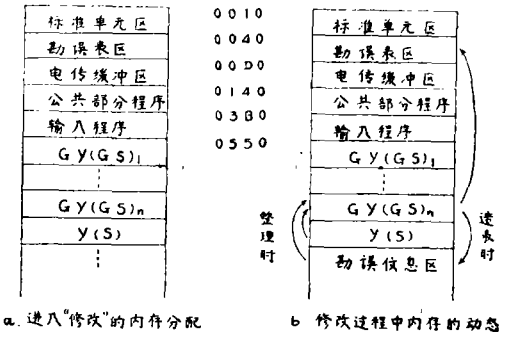

图　3

3、修　改

勘误表区暂时没有内容；若干个GY(或GS)从0550开始顺序存放；紧接着存放Y(或S)。修改工作便是按"修改1"提供的控制字，用最后一个修改指示，即Y(或S)前面的一个，对Y(或S)进行修改。

"修改"子程序的工作分为两大部份，造表和整理。造表时，根据每个修改元中的行列，建立勘误表送往表区，把相应替换内存送到勘误信息区，直到所有修改元处理完毕。整理时，造过表的修改指示GY(或GS)已无作用，所占用的单元用来存放整理后的结果，称之为整理区。整理程序参照勘误表分别把Y(或S)中的保留部分和勘误信息区的替换内容顺序搬往整理区，便完成整理工作。图b为工作示意图，由此可见，修改时Y(或S)是边改边往前移的。

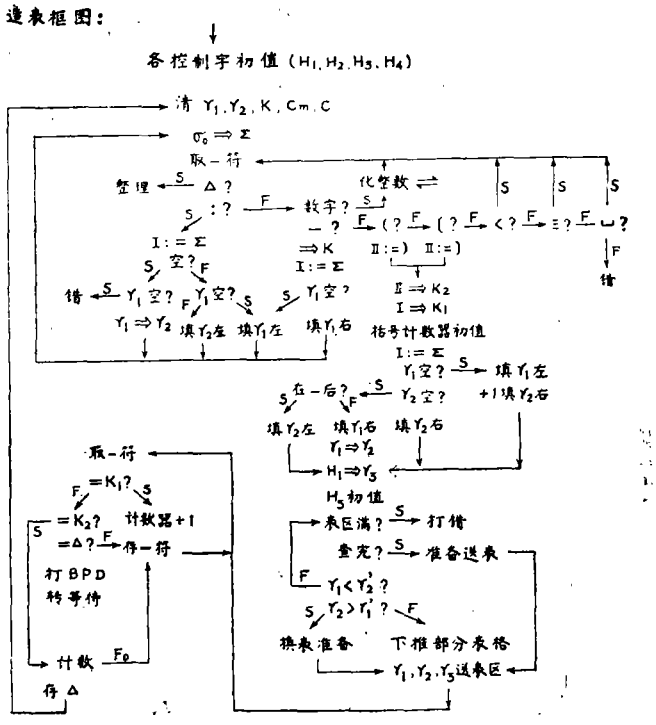

图　4

(1) 造表工作

造表时对GY(或GS)进行扫描，一面扫视、一面做相应的工作。这工作又分为填表、送表和搬信息三步来完成。

a、填 表

在扫视每个修改元的删去范围或插入位置时，就按所示行列进行填表，也就是在三个固定工作单元r_1,r_2,r_3中填进相应内容。

对第一种修改元，例如3(……)，填入：

r_1：	000	0003	000	0001
r_2：	000	0004	000	0001
r_3：	001	0000	000	KSD

对第二种修改元，例如5:6(……)，填入：

r_1：	000	0005	000	0006
r_2：	000	0005	000	0006
r_3：	001	0000	000	KSD

对第三种修改元，例如2:5－6:2(……)，填入：

r_1：	000	0002	000	0005
r_2：	000	0006	000	0002
r_3：	001	0000	000	KSD

即r_1填起点的行列，r_2填终点的行列，r_3填相应替换内容搬到勘误信息区后的开始地址KSD。

b、送表

填表工作完成，转到送表工作，把填好的三个单元的内容送往勘误表区。送表时需检查两件事：一是删去范围与先送去的表有否重叠，如重就以当前的换掉已重的；二是确定送表位置，以保证各修改元的勘误表按起点的大小顺序排列，这是后面整理工作的需要。

具体地说，就是在送表之前将r_1与表区已有的r_2'逐个比较大小，如$r_1 \geqslant r_2'$，再比下一个，如果把已有的表查完，都是$r_1 \geqslant r_2'$，则将三个单元送到表区。接着原有表存放，表计数器加3。如果有某个$r_2' > r_1$，再把相应的r_1'与r_2比较，若$r_2 > r_1'$则表示重叠了，就以r_1,r_2,r_3的内容去代替相重的那三个单元；若$r_1' \geqslant r_2$，就表示没重，但r_1的起点小，应把r_1'开始以下的勘误表下移三个单元，然后把三个单元的内容插在下移者的前面。这就保证了各表按起点大小顺序排列。

C、搬信息

填表、送表后，便扫视到修改元的开括号（ 。跳过这个字符，然后边检查边搬替换内容，凡不是对应闭括号）或结束号△的，都搬往勘误信息区，遇到对应的）时表

示替换内容已搬完，便加放一个结束号△，然后转到填表部份去处理下一个修改元。替换内容未搬完便遇到结束号△，作出错处理（括号不配对）。

每一个修改元都经以上三步处理后，在填表部分扫视到结束号△，意味着这个修改指示已造表完毕，转去整理。

(2) 整理工作

整理工作的依据是前面已按起点的大小顺序排列的勘误表，在整理的准备工作结束以后，先从勘误表区取头三个单元，到工作单元r_1，r_2，r_3中，作为应删字符的比较依据。

然后对Y(S)进行扫描，每取一字符，便将其行列位置与r_1比较，不等便送往整理区，相等便按r_3所示地址改取替换内容送往整理区，这也是逐个字符进行的，直到出现一个结束号为止。送完替换内容后再回头来跳过（只取不送）应删字符串，即继续方才中断的对Y(S)的扫描，这时是将字符的行列与r_2比较，不等就再取下一个，直到相等为止，相等表示已删完。

至此，便完成了一个修改元的修改工作，接着从表区顺序再取三个单元至r_1,r_2,r_3，重复上述工作。如此循环，直至在Y(S)中扫视到结束△，便完成了整理工作。

最后，按整理区的Y(S)的地址和长度填好相应的控制字，就完成了全部修改工作。

整理框图

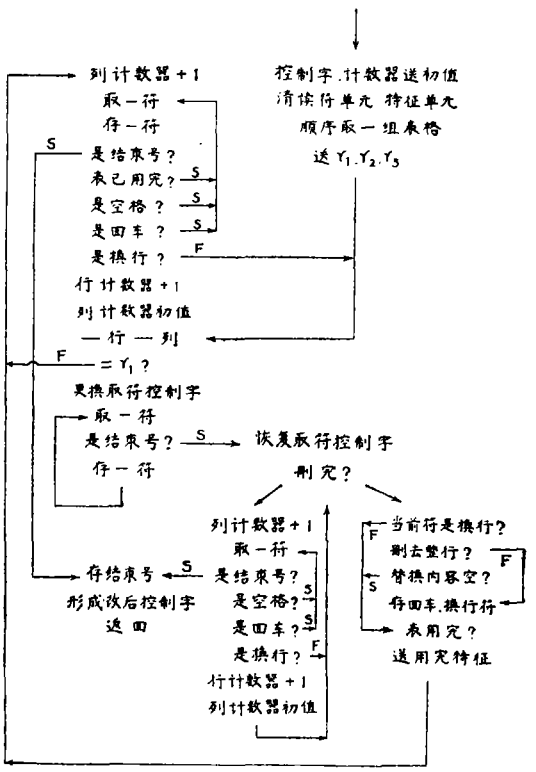

图　5

另外，还须补充说明两点：

1、整理区通常会延伸到原来存放Y(S)的区域，但这种延伸总是落在扫视的后面，不会影响工作的正确性，反而节省了内存空间。

2、当勘误表用完后，可能扫描尚未结束，但以后的扫描无须再进行行列比较。为此，我们用了一个特征单元，当勘误表用完时，使该单元为负，便不再进行行列比较，这样通常能节省一点时间。