Istruzione FILE:

Funzione

Chiude l'accesso di un programma ad un file esterno e lo apre ad un file specificato.

Formato

FILE:file-designator,{string-exp / *}

dove:

file-designator
: indica una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica un designatore di file

string-exp
: è una espressione stringa il cui valore indica il nome di un file esterno presente in una libreria

*
: chiude l'accesso del programma al file a cui era stato assegnato il numero designatore di file specificato con file-designator.

Azione

Al file esterno il cui nome corrisponde al valore della espressione stringa specificata con string-exp è associato il designatore di file indicato con il valore, arrotondato all'intero più prossimo, dell'espressione numerica specificata con file-designator. In questo modo le istruzioni di programma riferentesi al designatore di file suddetto, che sono eseguite successivamente, agiranno sul file con il nome specificato dal valore di string-exp. Il file che era prima associato al designatore di file suddetto è chiuso all'accesso da parte del programma, per cui le istruzioni che sono eseguite successivamente non possono riferirsi ad esso.

Se il secondo operando della istruzione è asterisco (*), è chiuso il file il cui numero designatore è specificato con file-designator. Le istruzioni ese-

guite successivamente non possono più far riferimento a detto file. L'area di memoria utilizzata come buffer del file è a disposizione dell'utente.

Note

1. Il file specificato con il valore di string-exp è cercato dal sistema nelle librerie aperte, secondo l'ordine di apertura. Non deve essere un file già aperto all'accesso da parte del programma.

2. Il file con il nome specificato nella istruzione deve essere stato creato con un comando CREATE (vedi capitolo 3).

3. Utilizzando l'istruzione FILE: un programma può accedere successivamente ad un numero indefinito di file dati esterni.

4. E' talvolta utile poter chiudere l'accesso di un programma ad un file esterno utilizzando l'opzione asterisco (*). Questa operazione protegge il file esterno nel caso di malfunzionamento del sistema (es. caduta di tensione).

5. Il valore della espressione numerica, file-designator, arrotondato all'intero più prossimo, deve essere maggiore di zero e non superiore al numero massimo di file che possono essere elaborati contemporaneamente dal programma (specificato con l'istruzione FILES).

Esempio

Nella routine sottostante si mostra come l'impiego dell'istruzione FILE: permette di assegnare ai designatori di file dei nuovi nomi di file. Le prime istruzioni WRITE: e READ: si riferiscono ai file A ed A1 che sono stati aperti dall'istruzione FILES.

```
LIST
FILE    +FILES1


0010 SCRATCH :1
0020 WRITE :1,"Ho scritto nel file A."
0030 WRITE :1,"Ora ho letto queste due frasi dal file A."
0040 SCRATCH :2
0050 WRITE :2,"Ho scritto nel file A1."
0060 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0070 RESTORE :1
0080 RESTORE :2
0090 DCL 80(A$,B$)
0100 READ :1,A$,B$
0110 PRINT A$
0120 PRINT B$
0130 READ :2,A$,B$
0140 PRINT
0150 PRINT A$
0160 PRINT B$
0170 FILE : 1,"A2"
0180 SCRATCH :1
0190 WRITE :1,"Ho scritto nel file A2."
0200 WRITE :1,"Ora ho letto queste ultime due frasi dal file A2."
0210 RESTORE :1
0220 READ :1,A$,B$
0230 PRINT
0240 PRINT A$
0250 PRINT B$
0260 FILE : 2,"A3"
0270 FILES A;A1
0280 SCRATCH :2
0290 WRITE :2,"Ho scritto nel file A3."
0300 WRITE :2,"Ora ho letto queste ultime due frasi dal file A3."
0310 RESTORE :2
0320 READ :2,A$,B$
0330 PRINT
0340 PRINT A$
0350 PRINT B$
0360 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.

Ho scritto nel file A2.
Ora ho letto queste ultime due frasi dal file A2.

Ho scritto nel file A3.
Ora ho letto queste ultime due frasi dal file A3.
```

Istruzione FILES

Funzione

Specifica quanti file esterni possono essere elaborati contemporaneamente dal programma.

Formato

FILES { filename / * } [, { filename / * }] ...

dove:

filename
: indica il nome di un file esterno che deve essere elaborato dal programma

*
: permette di aprire un file esterno specificato in una istruzione FILE:

Azione

I file esterni, specificati nella istruzione con il nome filename, sono resi accessibili al programma per operazioni di:

- lettura (file sequenziali)
- lettura e/o registrazione (file ad accesso diretto)

Ad ogni file viene associato un numero intero positivo, detto designatore di file, che coincide con il numero d'ordine con cui il nome del file compare nell'ambito della istruzione FILES (l'ordine si intende da sinistra a destra): il primo nome di file avrà come numero designatore 1, il secondo 2, etc....

Se nella istruzione compaiono uno o più asterischi anche ad essi è assegnato un numero designatore di file, corrispondente al numero d'ordine con cui compaiono nell'istruzione. A tale designatore di file può essere associato un file mediante una istruzione FILE: (vedi l'istruzione FILE:).

Note

1. L'istruzione FILES non è una istruzione di tipo eseguibile: dichiara quanti file dati esterni possono essere elaborati dal programma e con quali numeri designatori essi saranno riferiti nelle istruzioni di programma.

2. In un programma deve esservi una sola istruzione FILES.

3. In una istruzione FILES lo stesso nome di file deve comparire una sola volta.

4. I file con i nomi specificati nella istruzione devono essere stati creati con un comando CREATE (vedi capitolo 3).

5. Il numero di file che possono essere elaborati contemporaneamente dal programma è dato dal numero di operandi specificato nell'istruzione FILES.

6. Il sistema ricerca i file specificati con l'istruzione nelle librerie aperte secondo l'ordine di apertura.

Esempi

1. La routine sottostante mostra che l'istruzione FILES può essere posta in qualunque punto di un programma.

```
LIST
FILE


0010 SCRATCH :1
0020 WRITE :1,"Ho scritto nel file A."
0030 WRITE :1,"Ora ho letto queste due frasi dal file A."
0040 SCRATCH :2
0050 WRITE :2,"Ho scritto nel file A1."
0060 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0070 RESTORE :1
0080 RESTORE :2
0090 DCL 80(A$,B$)
0100 READ :1,A$,B$
0110 PRINT A$
0120 PRINT B$
0130 READ :2,A$,B$
0140 PRINT
0150 PRINT A$
0160 PRINT B$
0170 FILES A;A1
0180 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.
```

2. Nella routine sottostante si vede come si possa ottenere lo stesso risultato della routine precedente utilizzando una istruzione FILES con asterischi e due istruzione FILE:.

```
LIST
FILE    +FILES1


0010 FILE : 1,"A"
0020 SCRATCH :1
0030 WRITE :1,"Ho scritto nel file A."
0040 WRITE :1,"Ora ho letto queste due frasi dal file A."
0050 FILE : 2,"A1"
0060 SCRATCH :2
0070 WRITE :2,"Ho scritto nel file A1."
0080 WRITE :2,"Ora ho letto queste ultime due frasi dal file A1."
0090 RESTORE :1
0100 RESTORE :2
0110 DCL 80(A$,B$)
0120 READ :1,A$,B$
0130 PRINT A$
0140 PRINT B$
0150 READ :2,A$,B$
0160 PRINT
0170 PRINT A$
0180 PRINT B$
0190 FILES *;*
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ho scritto nel file A.
Ora ho letto queste due frasi dal file A.

Ho scritto nel file A1.
Ora ho letto queste ultime due frasi dal file A1.
```

Istruzione FKEY#

Funzione

Assegna un contenuto ai tasti funzione.

Formato

FKEY# n, string-constant

dove:

n

è un numero intero compreso tra 1 e 16

string-constant

è una stringa di caratteri del set ISO.

Azione

Al tasto funzione specificato con n viene assegnata la stringa di caratteri specificata con string-constant. Quando, durante l'esecuzione del programma, viene premuto il tasto indicato con n, la stringa string-constant è introdotta nel buffer di tastiera e visualizzata sul display.

Note

1. Per introdurre nel buffer di tastiera le stringhe associate con le funzioni da F9 ad F16 si devono premere i relativi tasti insieme con il tasto (SHIFT). Per poter premere i tasti funzione l'esecuzione del programma deve essere interrotta mediante l'istruzione STOP (vedi l'esempio seguente).

2. Il carattere due punti (:) usato come ultimo carattere di string-constant è interpretato come comando di END OF LINE.

3. Ai tasti funzione si può assegnare un comando START line-num che è eseguito quando si preme il corrispondente tasto, se l'ultimo carattere assegnato è due punti: ad esempio START 100: (vedi anche l'esempio seguente).

3. Ai tasti funzione si può assegnare un comando START line-num che è eseguito quando si preme il corrispondente tasto, se l'ultimo carattere assegnato è due punti: ad esempio START 100: (vedi anche l'esempio seguente).

4. I tasti funzione mantengono il contenuto ad essi assegnato finchè non viene effettuata una nuova assegnazione da un altro programma o durante lo stato di calcoli immediati, oppure il sistema è spento.

5. Quando viene acceso il sistema o sono eseguiti i comandi CONFIGURE, LDKEYS, OPTIONS (vedi capitolo 3), ai tasti funzione è associato il contenuto registrato sul disco sistema o floppy disk sistema mediante il comando STKEYS (vedi capitolo 3).

6. La somma dei caratteri di tutte le stringhe associate a tutti i tasti funzione non può essere maggiore di 238.

Esempi

1. Il programma seguente mostra un tipico impiego dei tasti funzione. Dopo aver assegnato ai tasti funzione un contenuto l'istruzione STOP interrompe la esecuzione del programma e l'operatore può premere il tasto funzione che preferisce. In questo caso premendo uno dei tasti funzione specificati (f1, f2 o f9) il sistema esegue una delle tre routine che compongono il programma. Si noti come avendo messo il punto e virgola alla fine della istruzione DISP (numero di linea 130), il messaggio relativo rimane sul display dopo l'esecuzione dell'istruzione STOP. Come si vede sono state riportate le esecuzioni delle tre routine del programma; per l'ultima si lascia al lettore la ricerca della risposta esatta al quesito posto dal sistema ...

```
LIST
FILE    +FKEY

0010 FKEY #1,START 500:
0020 FKEY #2,START 600:
0090 FKEY #9,START 820:
0110 DISP "Quale routine vuoi eseguire?"
0120 DELAY 50
0130 DISP "F1,F2,F9?";
0140 STOP
0500 PRINT "Io sono la routine che inizia dal numero di linea 500 e ti stampo "
0510 PRINT "la tabella ISO in righe e colonne,come puoi vedere:"
0520 PRINT
```

```
0530 FOR I=0 TO 15 STEP 1
0540 FOR J=I TO 255 STEP 16
0550 PRINT TAB((INT(J/16)+1)*5);CHR$(J);
0560 NEXT J
0570 NEXT I
0580 PRINT
0590 GOTO 1500
0600 PRINT "Io sono la routine che ti saluta,guarda il display!"
0605 PRINT "Premi il tasto PRINTALL per non rallentare la visualizzazione."
0610 DCL 256 A$
0620 LET A$=""
0630 FOR I=1 TO 15 STEP 1
0640 READ B$
0650 LET A$=A$+B$
0660 NEXT I
0670 LET L=LEN(A$)
0680 FOR J=1 TO 10000 STEP 1
0690 FOR R=1 TO L-31 STEP 1
0700 DISP EXT$(A$,R,R+31)
0710 FOR X=1 TO 30 STEP 1
0720 LET A=5+9
0730 NEXT X
0740 NEXT R
0750 DISP
0760 NEXT J
0770 DATA "****************"
0780 DATA "Salve, come st","ai? Come vedi l'","istruzione FKEY"
0790 DATA " ti permette la ","realizzazione di"," un programma ","con molte "
0800 DATA "routine.","  Tu scegli ...","  quella che ","vuoi eseguire."
0802 DATA " Per interromp","ermi premi BREAK","...SALUTE!"
0810 GOTO 1500
0820 PRINT " Io sono la routine che ti chiede un indovinello."
0830 PRINT
0840 PRINT "Un uomo vuole entrare in un castello ma non conosce la parola "
0850PRINT"d'ordine. Decide di nascondersi dietro ad un albero ed aspettare ..."
0860 PRINT " ...  Dopo un ora arriva un brigante e dal castello una voce "
0870 PRINT "dice :-DIECI?- ...  il brigante risponde :-CINQUE!-"
0875 PRINT "... la porta si apre ed il brigante entra."
0880 PRINT " ...  Dopo un ora arriva un  altro brigante. La voce dal castello "
0890 PRINT "dice:-OTTO?-. Il brigante risponde :-QUATTRO!- ... la porta si  "
0900 PRINT "apre ed il brigante entra."
0910 PRINT "Dopo un altra ora ecco arrivare un altro brigante. "                "
0920 PRINT "La voce dice:-SEI?- ... Il brigante risponde :-TRE!- ... ed entra. ""
0930 PRINT "L'uomo appostato dietro l'albero ora e' sicuro di poter entrare nel"
0940 PRINT "castello. Dopo un ora si presenta davanti al castello ... sente "  ."
0950 PRINT "la voce che dice:-QUATTRO?- ...  lui immediatamente risponde:-DUE!-""
0960 PRINT "Dal castello escono due briganti, lo prendono e ... lo impiccano!!!"
0970 PRINT "MA ALLORA COSA DOVEVA RISPONDERE? ... "
0975 PRINT "DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO."
0980 INPUT A$
0990 IF A$="SETTE" THEN 1400
1000 PRINT "Hai sbagliato!"
1110 DISP "Vuoi provare ancora";
1120 INPUT A$
1130 IF A$="SI" THEN 975
1140 GOTO 1500
1400 PRINT "Bravo e' esatta!!!"
1500 END

END OF LISTING
```

```
RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 500
Io sono la routine che inizia dal numero di linea 500 e ti stampo
la tabella ISO in righe e colonne,come puoi vedere:

   ▦   8       0   @   P   `   p   §   |||  |||  |||  |||  |||  |||  |||
   Γ   ◙   !   1   A   Q   a   q   |||  |||  |||  |||  |||  |||  |||  |||
   ⊥   @   "   2   B   R   b   r   |||  |||  |||  |||  |||  |||  |||  |||
   ┘   ◙   #   3   C   S   c   s   |||  |||  |||  |||  |||  |||  |||  |||
   ≷   ◙   $   4   D   T   d   t   |||  |||  |||  |||  |||  |||  |||  |||
   ▦   ÷   %   5   E   U   e   u   |||  |||  |||  |||  |||  |||  |||  |||
   √   Л   &   6   F   V   f   v   |||  |||  |||  |||  |||  |||  |||  |||
   o   ┤   '   7   G   W   g   w   |||  |||  |||  |||  |||  |||  |||  |||
   ↖   Σ   (   8   H   X   h   x   |||  |||  |||  |||  |||  |||  |||  |||
   →   ≬   )   9   I   Y   i   y   |||  |||  |||  |||  |||  |||  |||  |||
   Ξ   ς   *   :   J   Z   j   z   |||  |||  |||  |||  |||  |||  |||  |||
   ↓   θ   +   ;   K   [   k   {   |||  |||  |||  |||  |||  |||  |||  |||
   ≸   ▦   ,   <   L   \   l   |   |||  |||  |||  |||  |||  |||  |||  |||
   ←   ▦   -   =   M   ]   m   }   |||  |||  |||  |||  |||  |||  |||  |||
   θ   ▦   .   >   N   ↑   n   ‾   |||  |||  |||  |||  |||  |||  |||  |||
   θ   ▦   /   ?   O   _   o   ▒   |||  |||  |||  |||  |||  |||  |||  |||


RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 600
Io sono la routine che ti saluta,guarda il display!
Se PRINTALL e` attivo inibiscilo.


RUN
Quale routine vuoi eseguire?
F1,F2,F9
START 820
 Io sono la routine che ti chiede un indovinello!

Un uomo vuole entrare in un castello ma non conosce la parola
d'ordine. Decide di nascondersi dietro ad un albero ed aspettare ...
 ... Dopo un'ora arriva un brigante e dal castello una voce
dice  : - DIECI? - ... Il brigante risponde: - CINQUE -
... La porta si apre ed il brigante entra.
 ... Dopo un'ora arriva un altro brigante. La voce dal castello
dice: - OTTO? -. Il brigante risponde: - QUATTRO! - ... La porta si
apre ed il brigante entra.
Dopo un'altra ora ecco arrivare un altro brigante.
La voce dice: - SEI? - ... Il brigante risponde: - TRE - ed entra.
L'uomo appostato dietro l'albero ora e` sicuro di poter entrare nel
castello. Dopo un'ora si presenta davanti al castello ... sente
la voce che dice: - QUATTRO? - Lui immediatamente risponde: - DUE! -
Dal castello escono due briganti, lo prendono e ... lo impiccano!!!
MA ALLORA COSA DOVEVA RISPONDERE? ...
DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO.


?
DUE
Hai sbagliato!
Vuoi provare ancora?
SI
DIGITA UNA PAROLA ... CON I CARATTERI TUTTI IN MAIUSCOLO.
```

```
?
TRE
Hai sbagliato!
Vuoi provare ancora?
NO
```

2. Vediamo un esempio di una routine che permette di elaborare 10 dati "corretti" anche nel caso in cui s'introduca un dato"scorretto". Se l'utente si accorge che un dato introdotto era "scorretto", preme [f1] e quindi introduce il dato "corretto".

```
FILE

0010 LET C=1
0020 FKEY #1,9999999999999:
0030 INPUT A
0040 IF A=9999999999999 THEN 100
0050 LET B=10000*A
0060 PRINT "A=";A,"B=";B
0070 LET C=C+1
0080 IF C>10 THEN 120
0090 GOTO 30
0100 LET C=C-1
0110 GOTO 30
0120 END

END OF LISTING


A= 1            B= 10000
A= 2            B= 20000
A= 3            B= 30000
A= 4            B= 40000
A= 5            B= 50000
A= 45           B= 450000
A= 6            B= 60000
A= 7            B= 70000
A= 8            B= 80000
A= 9            B= 90000
A= 10           B= 100000
```

Istruzione FNEND

Funzione	Termina la definizione di una funzione multilinea.
Formato	**FNEND**
Azione	Vedi DEF/FNEND
Nota	Ogni definizione di funzione multilinea deve essere chiusa con una istruzione FNEND.

Istruzione FOR

Funzione

Inizia l'esecuzione di un ciclo iterativo.

Formato

FOR control-var = num-exp$_1$ TO num-exp$_2$ [STEP num-exp$_3$]

•
•
•

ISTRUZIONI BASIC

•
•
•

NEXT control-var

dove:

control-var
indica una variabile semplice numerica, detta variabile di controllo, che permette di controllare il numero di esecuzioni successive dell'intero insieme di istruzioni comprese tra l'istruzione FOR e l'istruzione NEXT

num-exp$_1$
indica una espressione numerica il cui valore viene assegnato alla variabile di controllo

num-exp$_2$
indica una espressione numerica il cui valore è confrontato con il valore della variabile di controllo per decidere se l'insieme di istruzioni suddette deve essere eseguito o saltato

num-exp$_3$
indica una espressione numerica il cui valore viene aggiunto al valore della variabile di controllo dopo ogni esecuzione dell'insieme delle istruzioni suddette

Azione

Quando è eseguita l'istruzione FOR, alla variabile di controllo è assegnato il valore della espressione numerica specificata con num-exp$_1$. Il valore della espressione numerica specificata con num-exp$_2$ è confrontato

con il valore della variabile di controllo; se quest'ultimo è minore od uguale al primo, sono eseguite le successive istruzioni fino alla istruzione NEXT. Quando è eseguita l'istruzione NEXT, il valore della variabile di controllo è incrementato del valore specificato con l'espressione num-exp$_3$ (detto incremento). La variabile di controllo è quindi confrontata con il valore di num-exp$_2$, se il suo valore è ancora minore od uguale a quest'ultimo sono nuovamente eseguite le istruzioni fino alla NEXT. Il ciclo di operazioni descritto viene ripetuto finchè il valore della variabile di controllo non supera quello di num-exp$_2$. Quando ciò accade, l'esecuzione del programma passa alla prima istruzione eseguibile successiva all'istruzione NEXT. La variabile di controllo mantiene l'ultimo valore assunto.

Se il valore della espressione num-exp$_1$ è negativo, il ciclo FOR/NEXT è eseguito finchè il valore della variabile di controllo non è minore del valore della espressione num-exp$_2$.

Se il valore della espressione num-exp$_3$ è zero allora il ciclo FOR/NEXT viene ripetuto senza fine.

Note

1. Se non si specifica l'opzione STEP num-exp$_3$, il valore da aggiungere come incremento al valore della variabile di controllo è +1.

2. La variabile di controllo control-var specificata nella istruzione NEXT deve essere la stessa che è specificata nell'istruzione FOR.

3. Due o più cicli FOR/NEXT possono essere nidificati ma non intersecati, per cui la seguente routine è corretta:

```
50 FOR I = 1 TO 10
- - - - -
100 FOR J = 2 TO 20
- - - - -
200 NEXT J
- - - - -
300 NEXT I
```

mentre la seguente routine non è corretta:

```
50 FOR I = 1 TO 10
- - - - -
100 FOR J = 2 TO 20
- - - - -
150 NEXT I
200 NEXT J
```

4. Due o più cicli FOR/NEXT nidificati non devono avere la stessa variabile di controllo.

5. In un programma ad una istruzione FOR deve sempre corrispondere una istruzione NEXT.

6. Le istruzioni del programma esterne ad un ciclo FOR/NEXT non possono rinviare l'esecuzione ad istruzioni interne al ciclo suddetto; fa eccezione l'istruzione RETURN come sottospecificato.

7. In un ciclo FOR/NEXT vi possono essere istruzioni di tipo GOSUB ed ON...GOSUB che rinviano l'esecuzione ad istruzioni esterne al ciclo FOR/NEXT. Le relative istruzioni RETURN riportano poi l'esecuzione all'interno del ciclo FOR/NEXT.

8. In un ciclo FOR/NEXT vi possono essere delle istruzioni GOTO, ON...GOTO, IF...THEN che rinviano l'esecuzione ad istruzioni esterne al ciclo FOR/NEXT. In questo caso la variabile di controllo mantiene l'ultimo valore assunto.

9. Si può modificare il valore della variabile di controllo con delle istruzioni di assegnazione (LET) nel ciclo FOR/NEXT.

10. Un ciclo FOR/NEXT non può intersecarsi con una definizione di funzione multilinea, ad esempio non si può avere:

```
50 FOR I = 1 TO 10
60 A = B*C
70 DEF FNA (X,Y)
80 PRINT Z
90 FN* = X*Y
100 NEXT I
120 FNEND
```

11. Se la variabile di controllo, il valore finale e l'incremento sono stati dichiarati in singola precisione, l'esecuzione del ciclo FOR/NEXT è più rapida che nel caso in cui siano rappresentati in doppia precisione.

Esempi

1. La routine seguente mostra l'impiego dell'istruzione FOR senza specificare esplicitamente l'incremento della variabile di controllo. Come si vede il sistema assume un incremento (STEP) implicito pari ad uno. Si osservi che, in questo caso, la variabile di controllo assume l'ultimo valore aumentato di uno quando termina l'esecuzione del ciclo FOR/NEXT relativo.

```
0010 FOR J=1 TO 10
0020 PRINT J,
0030 NEXT J
0040 PRINT
0050 PRINT "J=";J
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1             2             3             4             5
 6             7             8             9             10

J= 11
```

2. Nella routine sottostante, alla variabile di controllo viene assegnato un valore iniziale negativo.

```
LIST
FILE


0010 FOR I=-90 TO 10 STEP 10
0020 PRINT I,
0030 NEXT I
0040 PRINT
0045 PRINT "I=";I
0050 END

END OF LISTING


RUN
-90           -80           -70           -60           -50
-40           -30           -20           -10            0
 10
I= 20
```

3. Nell'istruzione FOR si è specificato un valore finale per la variabile di controllo minore di quello iniziale ma l'incremento è positivo. Il ciclo FOR/NEXT non è eseguito (infatti la variabile di controllo mantiene il valore iniziale e l'istruzione PRINT interna al ciclo non è eseguita).

```
LIST
FILE


0010 FOR I=1 TO -10 STEP 1
0020 PRINT I,
0030 NEXT I
0040 PRINT
0050 PRINT "I=";I
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

I= 1
```

4. Questa volta il valore finale è minore di quello iniziale, ma l'incremento specificato è negativo. Il ciclo FOR/NEXT è eseguito. Si osservi come la variabile di controllo assume il valore finale decrementato di uno (passo), al termine della esecuzione del ciclo FOR/NEXT.

```
LIST
FILE


0010 FOR I=1 TO -10 STEP -1
0020 PRINT I,
0030 NEXT I
0040 PRINT
0050 PRINT "I=";I
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1              0              -1             -2             -3
-4             -5              -6             -7             -8
-9             -10
I=-11
```

5. Nella routine sottostante si vede che i cicli FOR/ NEXT possono essere nidificati.

```
LIST
FILE    +ISOTAB

0010 PRINT
0020 PRINT "Ecco i caratteri della tabella ISO ordinati in righe e colonne:"
0030 PRINT
0040 FOR I=0 TO 15 STEP 1
0050 FOR J=I TO 255 STEP 16
0060 PRINT TAB((INT(J/16)+1)*5);CHR$(J);
0070 NEXT J
0080 NEXT I
0090 PRINT
0100 PRINT "I=";I,"J=";J
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

Ecco i caratteri della tabella ISO ordinati in righe e colonne:

    @   A       0   `   P   `   p
    A   B   !   1   a   Q   a   q
    B   C   "   2   b   R   b   r
    C   D   #   3   c   S   c   s
    D   E   $   4   d   T   d   t
    E   F   %   5   e   U   e   u
    F   G   &   6   f   V   f   v
    G   H   '   7   g   W   g   w
    H   I   (   8   h   X   h   x
    I   J   )   9   i   Y   i   y
    J   K   *   :   j   Z   j   z
    K   L   +   ;   k   [   k   {
    L   M   ,   <   l   \   l   |
    M   N   -   =   m   ]   m   }
    N   O   .   >   n   ^   n   ~
    O   P   /   ?   o   _   o
I= 16           J= 271
```

6. In questo ciclo FOR/NEXT si impiega una istruzione di salto (IF), per cui il controllo della esecuzione viene ceduto al progamma principale, quando la condizione specificata si avvera. Si noti come la variabile di controllo assume l'ultimo valore assegnatole prima dell'esecuzione dell'istruzione suddetta.

```
LIST
FILE

0010 FOR I=1 TO 10 STEP 1
0020 PRINT I,
0030 IF I=5 THEN 50
0040 NEXT I
0050 PRINT "I=";I
```

```
0060 PRINT "E' stata esequita l'istruzione IF."
0070 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1              2              3              4
I= 5
E' stata esequita l'istruzione IF.
```

7. Nel ciclo FOR/NEXT seguente si utilizza una istruzione di salto ad un sottoprogramma esterno al ciclo suddetto. Si noti come la variabile di controllo mantiene il valore assunto prima del salto al sottoprogramma.

```
LIST
FILE


0010 FOR I=1 TO 10 STEP 1
0020 GOSUB 80
0030 PRINT I*10
0040 NEXT I
0050 PRINT
0060 PRINT "I=";I
0070 GOTO 100
0080 PRINT "GOSUB numero";I,
0090 RETURN
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
GOSUB numero 1    10
GOSUB numero 2    20
GOSUB numero 3    30
GOSUB numero 4    40
GOSUB numero 5    50
GOSUB numero 6    60
GOSUB numero 7    70
GOSUB numero 8    80
GOSUB numero 9    90
GOSUB numero 10                   100

I= 11
```

8. Nel ciclo FOR/NEXT seguente si vede che il valore della variabile di controllo può essere modificato nel suo interno, mentre il valore finale e l'incremento non possono essere modificati.

```
LIST
FILE


0010 LET F=100
0020 LET P=5
0030 FOR I=1 TO F STEP P
0040 PRINT "I=";I,
0050 LET I=I+5
0060 PRINT "I+5=";I
0070 LET F=G=200
0080 PRINT "F=";F,"P=";P
0090 NEXT I
0100 PRINT "I=";I
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
I= 1              I+5= 6
F= 200            P= 5
I= 11             I+5= 16
F= 200            P= 5
I= 21             I+5= 26
F= 200            P= 5
I= 31             I+5= 36
F= 200            P= 5
I= 41             I+5= 46
F= 200            P= 5
I= 51             I+5= 56
F= 200            P= 5
I= 61             I+5= 66
F= 200            P= 5
I= 71             I+5= 76
F= 200            P= 5
I= 81             I+5= 86
F= 200            P= 5
I= 91             I+5= 96
F= 200            P= 5
I= 101
```

GO SUB

Istruzione GO SUB

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma.

Formato

GO SUB line-num

dove:
line-num

che indica un numero di linea di una istruzione del programma.

Azione

L'esecuzione del programma è trasferita all'insieme di istruzioni (detto sottoprogramma) che inizia con l'istruzione il cui numero di linea è specificato con line-num e termina con l'istruzione RETURN (vedi istruzione RETURN). Quando è eseguita l'istruzione RETURN, l'esecuzione del programma passa alla prima istruzione esecutiva successiva alla istruzione GO SUB.

Note

1. Nel sottoprogramma vi può essere più di una istruzione RETURN, ma, come buona norma di programmazione, è consigliabile l'impiego di una sola istruzione RETURN a cui fare riferimento con istruzioni GO TO da diversi punti del sottoprogramma.

2. Le istruzioni RETURN sono collegate alle istruzioni GO SUB in modo dinamico: se sono eseguite, ad esempio, 10 istruzioni GO SUB, la prima istruzione RETURN che sarà successivamente eseguita rinvierà il controllo della esecuzione del programma alla istruzione esecutiva successiva alla decima istruzione GO SUB eseguita; la seconda istruzione RETURN eseguita rinvierà l'esecuzione del programma alla istruzione esecutiva successiva alla nona istruzione

GO SUB eseguita e così via.

3. Il numero di istruzioni GO SUB che possono essere eseguite prima che sia eseguita una istruzione RETURN è limitato dalla dimensione della memoria utente disponibile.

4. In un sottoprogramma vi può essere una istruzione che trasferisce l'esecuzione alla istruzione GO SUB che richiama il sottoprogramma stesso.

5. In un ciclo FOR/NEXT si possono usare istruzioni GO SUB che si riferiscono ad una istruzione esterna al ciclo suddetto, ma non vi possono essere istruzioni GO SUB che si riferiscono ad una istruzione interna al ciclo.

6. line-num non può essere il numero di linea di una istruzione interna ad un ciclo FOR/NEXT.

7. Un sottoprogramma può far parte di una funzione multilinea; in questo caso anche l'istruzione GO SUB deve essere una istruzione della funzione multilinea.

8. Ogni volta che una istruzione RETURN è eseguita vi deve essere almeno una istruzione GO SUB per la quale non sia stata eseguita la relativa istruzione RETURN.

9. Da un sottoprogramma si può uscire con una istruzione di salto (esempio GO TO, IF... THEN) prima che venga eseguita l'istruzione RETURN, ma è meglio evitare l'impiego di tale possibilità.

Esempi

1. La routine sottostante contiene un sottoprogramma che permette di calcolare il massimo comun divisore di tre numeri introdotti da tastiera.

```
LIST
FILE    +GOSUB

0010 DISP "Valori per A, B e C";
0020 INPUT A,B,C
0030 LET X=A
0040 LET Y=B
0050 GOSUB 140
0060 LET X=G
0070 LET Y=C
0080 GOSUB 140
```

```
0090 PRINT "A=";A,"B=";B,"C=";C,"MCD=";G
0100 DISP "si=SI,no=NO; CONTINUI";
0110 INPUT A$
0120 IF A$="NO" THEN 220
0130 GOTO 10
0140 LET Q=INT(X/Y)
0150 LET R=X-Q*Y
0160 IF R=0 THEN 200
0170 LET X=Y
0180 LET Y=R
0190 GOTO 140
0200 LET G=Y
0210 RETURN
0220 END

END OF LISTING


RUN
Valori per A, B e C?
10,1110,150
A= 10            B= 1110          C= 150           MCD= 10
si=SI,no=NO; CONTINUI?
SI
Valori per A, B e C?
125,45,75
A= 125           B= 45            C= 75            MCD= 5
si=SI,no=NO; CONTINUI?
NO
```

2. Il programma sottostante è composto da due sotto-programmi nidificati che permettono di calcolare il valore medio ed il massimo comun divisore di tre numeri introdotti da tastiera.

```
LIST
FILE    +GOSUB1


0010 DISP "Valori per A, B e C";
0020 INPUT A,B,C
0030 LET X=A
0040 LET Y=B
0050 GOSUB 131
0060 LET X=G
0070 LET Y=C
0080 GOSUB 140
0090 PRINT "A=";A,"B=";B,"C=";C,"MCD=";G
0100 DISP "si=SI,no=NO; CONTINUI";
0110 INPUT A$
0120 IF A$="NO" THEN 220
0130 GOTO 10
0131 GOSUB 211
0140 LET Q=INT(X/Y)
0150 LET R=X-Q*Y
0160 IF R=0 THEN 200
0170 LET X=Y
0180 LET Y=R
0190 GOTO 140
0200 LET G=Y
0210 RETURN
0211 LET D=(A+B+C)/3
0212 PRINT "A=";A,"B=";B,"C=";C,"Valore medio =";D
0213 RETURN
0220 END

END OF LISTING
```

```
RUN
Valori per A, B e C?
10,20,30
A= 10          B= 20          C= 30          Valore medio = 20
A= 10          B= 20          C= 30          MCD= 10
si=SI,no=NO; CONTINUI?
SI
Valori per A, B e C?
120,150,300
A= 120         B= 150         C= 300         Valore medio = 190
A= 120         B= 150         C= 300         MCD= 30
si=SI,no=NO; CONTINUI?
NO
```

3. In questo esempio, dopo l'istruzione GOSUB con numero di linea 10, è eseguita quattro volte l'istruzione GOSUB con numero di linea 70. L'istruzione RETURN rinvia per quattro volte all'esecuzione dell'istruzione 72 ed, infine, all'istruzione 30.

```
LIST
FILE    +GOSUB4


0005 LET A=0
0010 PRINT "Vediamo un esempio di sottoprogramma che richiama se stesso."
0020 GOSUB 50
0030 GOTO 30
0050 LET A=A+1
0055 IF A>5 THEN 80
0060 PRINT "Esecuzione n";A;"del sottoprogramma."
0070 GOSUB 50
0072 LET A=A-1
0075 PRINT "Eseguita la RETURN relativa al GOSUB n";A
0080 RETURN
0090 END

END OF LISTING


RUN
Vediamo un esempio di sottoprogramma che richiama se stesso.
Esecuzione n 1 del sottoprogramma.
Esecuzione n 2 del sottoprogramma.
Esecuzione n 3 del sottoprogramma.
Esecuzione n 4 del sottoprogramma.
Esecuzione n 5 del sottoprogramma.
Eseguita la RETURN relativa al GOSUB n 5
Eseguita la RETURN relativa al GOSUB n 4
Eseguita la RETURN relativa al GOSUB n 3
Eseguita la RETURN relativa al GOSUB n 2
Eseguita la RETURN relativa al GOSUB n 1
```

Istruzione GOTO

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata.

Formato

GOTO line-num

dove:
line-num
è un numero intero positivo compreso tra 1 e 9999 che indica il numero di linea di una istruzione del programma.

Azione

L'esecuzione del programma passa alla istruzione il cui numero di linea è specificato con line-num.

Note

1. Se line-num è il numero di linea di una istruzione non esecutiva allora l'esecuzione del programma passa alla successiva istruzione esecutiva.

2. Una istruzione GO TO esterna ad una definizione di funzione multilinea non può avere come operando, line-num, un numero di linea di una istruzione interna alla definizione di funzione suddetta.

3. Una istruzione GOTO interna ad una definizione di funzione multilinea, non deve avere come operando, line-num, un numero di linea di una istruzione esterna alla definizione di funzione suddetta.

4. Una istruzione GOTO esterna ad un ciclo FOR/NEXT non può avere come operando, line-num, un numero di linea di una istruzione interna al ciclo suddetto.

5. Una istruzione GOTO interna ad un ciclo FOR/NEXT

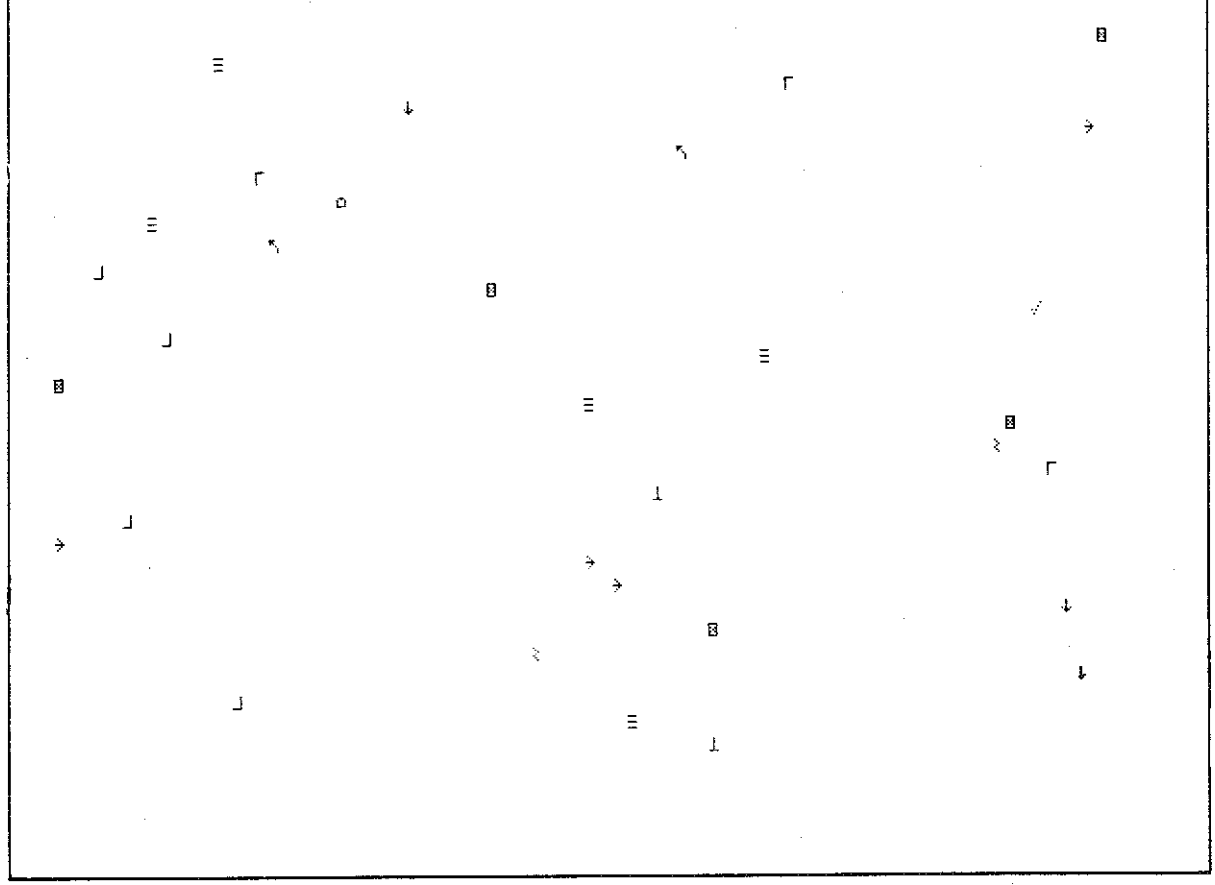
può avere come operando, line-num, un numero di linea di una istruzione esterna al ciclo suddetto.

6. Come si vede nei testi successivi, la parola chiave può avere o no interposto uno spazio, quindi si può specificare : GO TO o GOTO.

Esempi

1. Vediamo un esempio d'impiego dell'istruzione GOTO.

```
LIST
FILE


0010 PRINT "Quando vuoi fermarmi premi BREAK!"
0020 PRINT
0030 PRINT
0040 PRINT TAB(INT(RND*1000));CHR$(RND*10+1)
0050 GOTO 40
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Quando vuoi fermarmi premi BREAK!
```



2. Come si vede è possibile dall'interno di un ciclo FOR/NEXT rinviare l'esecuzione all'esterno del ciclo stesso.

```
LIST
FILE


0010 FOR I=1 TO 10 STEP 1
0020 PRINT I
0030 GOTO 50
0040 NEXT I
0050 PRINT "Sono uscito da un ciclo FOR/NEXT!"
0060 END

END OF LISTING


RUN
 1
Sono uscito da un ciclo FOR/NEXT!
```

Istruzione IF...THEN

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata, nel caso che si verifichi la condizione predefinita.

Formato

```
IF { ({num-exp1 rel-opr num-exp2 }) { AND } ({num-exp3 rel-opr num-exp4 }) }  THEN line-num
   { ({string-exp1 rel-opr string-exp2}) { OR  } ({string-exp3 rel-opr string-exp4}) }
                  { num-exp1 rel-opr num-exp2 }
                  { string-exp1 rel-opr string-exp2 }
```

dove:

num-exp$_1$
è una espressione numerica che viene confrontata con l'espressione numerica specificata con num-exp$_2$

string-exp$_1$
è una espressione di tipo stringa che viene confrontata con string-exp$_2$

rel-opr
è un operatore di confronto scelto tra uno dei seguenti:

<>	oppure	><	non uguale
	=		uguale
>=	oppure	=>	maggiore o uguale
<=	oppure	=<	minore o uguale
	>		maggiore di
	<		minore di

num-exp$_2$
è una espressione numerica che viene confrontata con num-exp$_1$

string-exp$_2$
è una espressione di tipo stringa che viene confrontata con string-exp$_1$

AND
specifica l'operazione di congiunzione tra due proposizioni logiche

OR
specifica l'operazione di disgiunzione tra due proposizioni logiche

num-exp$_3$
è una espressione numerica che viene confrontata con l'espressione numerica specificata con num-exp$_4$

string-exp$_3$
è una espressione stringa che viene confrontata con l'espressione stringa specificata con string-exp$_4$

num-exp$_4$
è una espressione numerica che viene confrontata con l'espressione numerica specificata con num-exp$_3$

string-exp$_4$
è una espressione stringa che viene confrontata con l'espressione stringa specificata con string-exp$_3$

line-num
è un numero intero positivo che indica il numero di linea di una istruzione del programma.

Azione

Se è specificato l'operando AND, le quattro espressioni specificate sono calcolate ed i loro valori sono confrontati a due a due come indicato dai rispettivi operatori di confronto. Se entrambe le relazioni sono soddisfatte, l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. In ogni altro caso (una sola delle due relazioni od entrambe non sono soddisfatte) l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alle istruzioni IF...THEN.

Se è specificato l'operando OR, le quattro espressioni specificate sono calcolate ed i loro valori sono confrontati a due a due come indicato dai rispettivi operatori di confronto. Se una sola od entrambe le relazioni sono soddisfatte, l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. Se nessuna delle due relazioni è soddisfatta, l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alle istruzioni IF... THEN.

Se non è specificato nè l'operando AND nè l'operando OR, le due espressioni sono calcolate ed i valori ottenuti sono confrontati secondo l'operatore di confronto specificato. Se la relazione è soddisfatta l'esecuzione del programma passa all'istruzione con numero di linea specificato con line-num. Se la relazione non è soddisfatta l'esecuzione del programma prosegue con l'istruzione esecutiva successiva alla istruzione IF...THEN.

Note

1. L'impiego dell'operatore di uguaglianza tra due espressioni numeriche in una istruzione IF...THEN può, in alcuni casi, dar luogo a degli inconvenienti a causa della rappresentazione dei numeri che è necessariamente limitata. Ad esempio se in un programma ci sono delle istruzioni come:

   ```
   10   A = 1.0/3 *3
   20   IF A = 1 THEN 75
   ```

   allora l'istruzione 20 non trasferirà il controllo all'istruzione 75 perchè il valore assegnato ad A non è 1 ma 0.9999999999999 oppure 0.999999 a seconda del tipo di precisione dichiarata per A (doppia o singola).

2. Una istruzione IF...THEN esterna ad una definizione di funzione multilinea non può avere come operando, line-num, un numero di linea di una istruzione interna alla definizione di funzione suddetta.

3. Una istruzione IF...THEN che fa parte di una definizione di funzione multilinea non può avere come operando un numero di linea, line-num, di una istruzione che è esterna alla definizione di funzione suddetta.

4. Una istruzione IF...THEN esterna ad un ciclo FOR/NEXT non può avere come operando, line-num, un numero di linea di una istruzione interna al ciclo suddetto.

5. Le espressioni confrontate devono essere omogenee: entrambe numeriche od entrambe stringa.

Esempi

1. Vediamo un esempio di impiego dell'istruzione IF...THEN in cui viene fatto un confronto fra stringhe.

```
LIST
FILE    +IFTHEN

0010PRINT"Ecco un esempio di istruzione IF in cui si confrontano delle stringhe"
0020 PRINT
0030 DISP "Introduci il nome di una citta`!  "
0040 INPUT A$
0050 IF A$="MILANO" THEN 80
0060 PRINT "Non e` la citta` di cui voglio parlarti. Digitane un altra."
```

```
0070 GOTO 40
0080 PRINT "Milano e` proprio la citta` di cui voglio parlarti."
0090 PRINT "Ma ora non ho molto tempo per cui ti saluto ..."
0100 PRINT "  ... ne riparleremo un altra volta."
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di istruzione IF in cui si confrontano delle stringhe

Introduci il nome di una citta`!
?
ROMA
Non e` la citta` di cui voglio parlarti. Digitane un altra.
?
MILANO
Milano e` proprio la citta` di cui voglio parlarti.
Ma ora non ho molto tempo per cui ti saluto ...
  ... ne riparleremo un altra volta.
```

2. Vediamo un esempio in cui nell'istruzione IF...THEN vi è un confronto tra valori numerici.

```
LIST
FILE


0010 DISP "Introduci quanto guadagni!      ";
0020 INPUT A
0030 IF A<=1000000 THEN 70
0040 IF A<=5000000 THEN 90
0050 IF A<=10000000 THEN 110
0060 PRINT "Guadagni troppo per cui pagherai";A*50/100;"lire di tasse!"
0070 PRINT "Non guadagni molto per cui pagherai";A*1/100;"lire di tasse."
0080 GOTO 120
0090 PRINT "Guadagni abbastanza per cui pagherai";A*5/100;"lire di tasse."
0100 GOTO 120
0110 PRINT "Guadagni parecchio per cui pagherai";A*10/100;"lire di tasse."
0120 END

END OF LISTING


RUN
Introduci quanto guadagni!      ?
650000
Non guadagni molto per cui pagherai 6500 lire di tasse.
RUN
Introduci quanto guadagni!      ?
1500000
Guadagni abbastanza per cui pagherai 75000 lire di tasse.
RUN
Introduci quanto guadagni!      ?
5600000
Guadagni parecchio per cui pagherai 560000 lire di tasse.
RUN
Introduci quanto guadagni!      ?
11000000
Guadagni troppo per cui pagherai 5500000 lire di tasse!
Non guadagni molto per cui pagherai 110000 lire di tasse.
```

3. Vediamo come funziona la seguente routine che impiega una istruzione IF...THEN con l'operando AND.

```
LIS
FILE


0010 INPUT A,B
0020 INPUT A$,B$
0030 IF (A=B)AND (A$=B$) THEN 60
0040 PRINT "L'esecuzione e' continuata in sequenza!"
0050 GOTO 70
0060 PRINT "L'esecuzione non e' continuata in sequenza!"
0070 GOTO 10
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
?
10,10
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,10
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
10,12
?
PARIGI,PARIGI
L'esecuzione e' continuata in sequenza!
?
10,12
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
```

4. Diamo qui sotto un esempio d'impiego della istruzione IF...THEN con l'operando OR.

```
FILE


0010 INPUT A,B
0020 INPUT A$,B$
0030 IF (A=B)OR (A$=B$) THEN 60
0040 PRINT "L'esecuzione e' continuata in sequenza!"
0050 GOTO 70
0060 PRINT "L'esecuzione non e' continuata in sequenza!"
0070 GOTO 10
0080 END

END OF LISTING
```

```
RUN
?
10,10
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,10
?
PARIGI,ROMA
L'esecuzione non e' continuata in sequenza!
?
10,12
?
PARIGI,PARIGI
L'esecuzione non e' continuata in sequenza!
?
10,12
?
PARIGI,ROMA
L'esecuzione e' continuata in sequenza!
?
```

Istruzione IMMAGINE

Funzione

Specifica un formato predefinito utilizzato dalla istruzioni PRINT USING, DISP USING, BUILD USING, MAT PRINT USING.

Formato

: { **literal-field** / **image-field** } [**literal-field** / **image-field**] ...

dove:

literal-field

è un campo costituito da caratteri che sono riprodotti esattamente come sono specificati

image field

è un campo costituito da caratteri che sono sostituiti con i valori delle espressioni specificate nella istruzione che fa riferimento alla istruzione immagine suddetta.

Azione

Ogni volta che una istruzione PRINT USING, MAT PRINT USING, DISP USING o BUILD USING, contenente come operando il numero di linea della istruzione IMMAGINE di formato, è eseguita, i caratteri da stampare, visualizzare su display o trasferire in una variabile stringa, sono generati rispettivamente nel buffer di stampa, nel buffer di display o nella variabile stringa con il formato descritto nel paragrafo "Formato associato ai valori delle espressioni presenti nelle istruzioni PRINT USING, DISP USING, BUILD USING".

Campi immagine di formato (image field)

I "campi immagine" presenti in una istruzione IMMAGINE di formato possono essere campi "immagine di numero", associati ai valori di espressioni numeriche, o campi "immagine di stringa", associati ai valori di espressioni stringa.

I campi "immagine di numero" possono essere:

campi "immagine di numero intero"
campi "immagine di numero decimale"
campi "immagine di numero esponenziale"
campi "immagine con segno $"

1. Campo "immagine di numero intero" - è costituito da una sequenza di segni di numero

   formato: # #[#]...

   estensione: minimo 2 massimo 25 caratteri

2. Campo "immagine di numero decimale"- è composto da una sequenza di segni di numero # e dal punto decimale (.)

   formato: # # [#]...[#]...

   estensione: minimo 3 massimo 26 caratteri

3. Campo "immagine di numero esponenziale"- è composto da una sequenza di segni di numero #, dal punto decimale e dal simbolo immagine dell'esponente

   formato: campo "immagine di numero decimale" ↑↑↑↑

   estensione: minimo 7 massimo 30 caratteri

   nota: il simbolo immagine dell'esponente è sempre l'ultimo simbolo nel campo numerico esponenziale.

4. Campo "immagine con segno $" - è costituito da due segni $ oppure da uno o più segni di $ seguiti, eventualmente, da un campo "immagine di numero intero", o da un campo "immagine di numero decimale", o da un punto decimale seguito da uno o più segni di numero #.

   formato: $ [$] ... [campo "immagine di numero intero" | campo "immagine di numero decimale" | .#[#]...]

estensione: minimo 2 massimo 30 caratteri

I campi "immagine di stringa" sono costituiti dal segno di apostrofo seguito eventualmente da una o più lettere L o R o C.

formato: { '[L] ... / '[R] ... / '[C] ... }

estensione: minimo 1 massimo 74 caratteri

Formato associato ai valori delle espressioni presenti nelle istruzioni PRINT USING, MAT PRINT, DISP USING e BUILD USING

La conversione dei valori delle espressioni presenti nelle istruzioni PRINT USING, MAT PRINT USING, DISP USING e BUILD USING è realizzata secondo le seguenti regole:

1. Il valore numerico associato ad un campo "immagine di numero intero" è generato con ogni cifra in corrispondenza di ogni segno # . Il numero è allineato a destra all'interno del campo. Se il numero non è intero viene troncata la parte decimale. Se il numero è positivo viene anteposto uno spazio alla prima cifra. Se il numero è negativo viene anteposto un segno meno alla prima cifra. Se il numero ha, nella parte interna, più cifre dei segni # che costituiscono il campo, è generato un asterisco in corrispondenza di ogni segno # del campo.

2. Il valore numerico associato ad un "campo immagine di numero decimale" è generato con ogni cifra in corrispondenza di un segno # . Il punto decimale è nella stessa posizione in cui è indicato nel campo immagine. La parte intera del numero è generata a sinistra del punto decimale. La parte decimale è generata a destra del punto decimale. Se il numero ha più cifre decimali di quanti sono i segni # dopo il punto decimale, è arrotondato e quindi troncato dalle cifre eccedenti il campo. Se il numero è positivo viene anteposto uno spazio alla prima cifra. Se il numero è negativo, un segno meno è anteposto alla prima cifra. Se il numero ha più cifre intere di quanti sono i segni # prima del punto decimale, è generato un asterisco in corrispondenza di ogni carattere del campo.

3. Il valore numerico associato ad un campo "immagine di un numero esponenziale" è generato con le stesse modalità indicate nel punto 2. I segni ↑↑↑↑ sono sostituiti rispettivamente con il carattere E, il segno + o - e due cifre da 0 a 9 che indicano l'esponente in base 10 della potenza per la quale è moltiplicato il numero decimale che precede la sostituzione di ↑↑↑↑.

4. Il valore numerico associato ad un campo con segno $ viene generato anteponendo ad esso un segno $. Il numero è allineato nell'ambito del campo secondo le modalità descritte nei punti 1 e 2. Se il campo immagine è composto da soli segni di $, il valore numerico deve essere intero (se non lo è viene troncato nella parte decimale) e viene generato allineato a destra nell'ambito del campo. Se il campo immagine è composto da più segni di $ seguiti da un campo immagine numerico ed il valore numerico ad esso associato è costituito da un numero di cifre inferiore al campo numerico suddetto, il segno di $ viene generato nell'ultima posizione a destra in cui compare nel campo immagine. Se il campo immagine è composto da più segni di $ seguiti da un campo immagine numerico ed il valore numerico associato è costituito da un numero di cifre superiore, nella parte intera, alla parte del campo numerico a sinistra del punto decimale, i segni di $ diversi dal primo sono considerati come dei segni di #.

5. Il valore stringa associato ad un campo immagine di stringa viene generato allineato a sinistra, a destra od al centro, nell'ambito del campo suddetto, a seconda che dopo l'apostrofo vi siano rispettivamente la lettera L, R o C. Se la stringa eccede il campo immagine è troncata a destra. Nel caso in cui nell'immagine vi sia la lettera C ed il numero di caratteri del campo sia superiore a quello della stringa, ma la differenza tra le due lunghezze sia un numero dispari pari a 2n+1, la stringa è fatta precedere da n spazi e seguire da n+1 spazi. Se un campo immagine di stringa è composto dal solo apostrofo, è generato il primo carattere del valore stringa ad esso associato.

Note

1. L'istruzione immagine non è di tipo esecutivo.

2. I caratteri specificati al posto di un campo del tipo literal-field non possono essere:

   - l'apostrofo, se è seguito dalla lettera L od R o C

   - il punto, se è preceduto dal segno $ oppure dal segno # .

3. Come si vede dal formato delle istruzioni BUILD USING, DISP USING, PRINT USING e MAT PRINT USING, l'immagine di formato può essere assegnata ad una variabile stringa mediante l'istruzione di assegnazione:

   string-var = "immagine di formato"

4. Nel dimensionare un campo immagine di un numero, si ricordi cha al numero delle cifre previste per il numero deve essere aggiunto un carattere che occupi un posto per il segno algebrico.

5. Non si può specificare una istruzione immagine senza specificare almeno un campo immagine.

Esempi

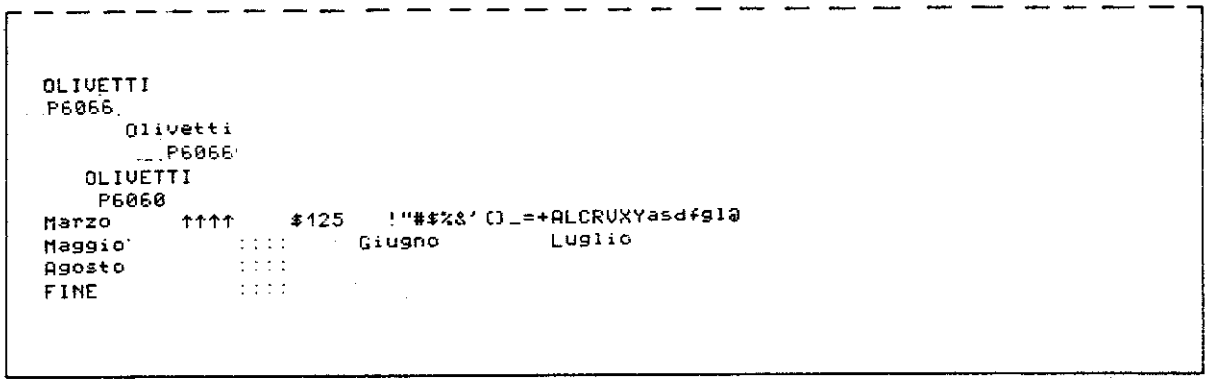
1. Vediamo come si può utilizzare l'istruzione immagine in un programma per stampare dei valori numerici. La stampa effettuata con l'istruzione 90 mostra come nell'istruzione immagine vi possono essere dei caratteri che sono stampati esattamente come specificati. Il numero intero -5 è stampato allineato a destra nel relativo campo della istruzione immagine specificata. Il numero positivo 6 è preceduto da uno spazio. L'istruzione 100 stampa cinque asterischi perchè il valore 12345, considerando il segno + (implicito), è maggiore del numero di posizioni del campo ad esso relativo nella istruzione immagine 60 (5 posizioni). L'istruzione 110 stampa i valori in essa specificati su altrettante linee di stampa, tra loro distinte, perchè l'istruzione immagine relativa contiene un solo campo. L'istruzione 120 stampa due valori troncati delle relative cifre decimali, perchè il campo numerico specificato è un campo immagine di numero intero. L'istruzione 130 stampa i valori specificati allineati,

nel relativo campo immagine, rispetto al punto decimale. Dopo il punto decimale sono stampati degli zeri nelle posizioni previste dal campo, se il numero ha meno cifre decimali di quelle previste. Un valore nel formato esponenziale viene trasformato nel formato decimale corrispondente (vedi il valore 12 E 10). L'istruzione 150 si riferisce ad una immagine di stampa che è stata assegnata ad una variabile stringa; si osservi come il valore specificato (-25.8) è arrotondato e non troncato nella parte decimale perchè il campo immagine relativo è del tipo immagine di numero decimale (infatti è stato aggiunto un punto dopo i segni di numero). Infine le istruzioni 170 e 190 si riferiscono rispettivamente a campi immagine di numero esponenziale e con segno di $.

```
LIST
FILE    +IMAGE


0010 PRINT "Per poter controllare le posizioni relitive ai diversi campi della"
0020 PRINT "istruzione IMMAGINE si osservi la seguente stampa."
0030 PRINT
0040PRINT"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789"
0050 :############################   primo valore      ##   secondo valore.
0060 :      #####
0080 :################.###
0090 PRINT USING 50,-5.6
0100 PRINT USING 60,12345
0110 PRINT USING 60,0001,0002,0003
0120 PRINT USING 60,123.4,123.75
0130 PRINT USING 80,1.1,12E10
0140 LET A$="###."
0150 PRINT USING A$,-25.8
0160 :######.##########################↑↑↑↑
0170 PRINT USING 160,-12,12345678901E77,-123,1234567890
0180 :$$$$$$$$$$$$     $######.####       $$$$.###↑↑↑↑
0190 PRINT USING 180,1500,25.45,25.15E15
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Per poter controllare le posizioni relitive ai diversi campi della
istruzione IMMAGINE si osservi la seguente stampa.

12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
                      -5   primo valore       6   secondo valore.
     *****
         1
         2
         3
       123
       123
              1.100
   120000000000.000
-26.
-1200.000000000000000000E-02
 1234.567890100000000000E+84
-1230.000000000000000000E-01
 1234.567890000000000000E+06
        $1500     $   25.4500       $251.500E+14
```

2. Vediamo l'impiego della istruzione immagine in un programma per la stampa di stringhe. Come si vede nel programma sottostante, le istruzioni immagine di formato possono essere poste anche all'interno di un ciclo FOR/NEXT (infatti sono istruzioni non esecutive). Le istruzioni 150, 160 e 170 stampano i relativi valori allineati a sinistra, a destra ed al centro rispetto al relativo campo immagine. L'istruzione 180 stampa un valore stringa ed un valore numerico oltre che i caratteri specificati come "literal-field". L'istruzione 190 prosegue la stampa nella linea successiva con la stessa immagine di stampa, perchè i valori specificati superano in numero i relativi campi immagine. Infine dalla stampa relativa all'istruzione 200 si vede che se si specificano meno valori di campi immagine i campi immagine in eccedenza sono ignorati.

```
LIST
FILE      MNB

0010 PRINT
0020 PRINT "Per poter controllare le posizione relative ai diversi campi della"
0030 PRINT "istruzione IMMAGINE si osservi la seguente stampa."
0040 PRINT
0050PRINT"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789"
0060 FOR I=1 TO 3 STEP 1
0070 PRINT
0080 :'LLLLLLLLLLLL
0090 :'RRRRRRRRRRRR
0100 :'CCCCCCCCCCCC
0110 :'LLLLLLL  ↑↑↑↑    $$$$    !"#$%&'()_=+ALCRUXYasdfgl@
0120 PRINT I
0130 NEXT I
0140 :'LLLLLLLL      ::::    'CCCCCCC    'RRRRRRR
0150 PRINT USING 80,"OLIVETTI","P6066"
0160 PRINT USING 90,"Olivetti","P6066"
0170 PRINT USING 100,"OLIVETTI","P6066"
0180 PRINT USING 110,"Marzo",125
0190 PRINT USING 140,"Maggio","Giugno","Luglio","Agosto"
0200 PRINT USING 140,"FINE"
0210 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

Per poter controllare le posizione relative ai diversi campi della
istruzione IMMAGINE si osservi la seguente stampa.

12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789

 1

 2

 3
```


OLIVETTI
P6066.
Olivetti
P6066
OLIVETTI
P6060
Marzo ↑↑↑↑ $125 !"#$%&' ()_=+ALCRVXYasdfgl@
Maggio' :::: Giugno Luglio
Agosto ::::
FINE ::::

Istruzione INPUT

Funzione

Assegna i valori introdotti da tastiera alle variabili di programma specificate.

Formato

INPUT { num-var / string-var } [, { num-var / string-var }] ...

dove:

num-var
: è una variabile numerica, semplice o con indice, a cui è assegnato il valore numerico introdotto da tastiera

string-var
: è una variabile stringa, semplice o con indice, a cui è assegnata una stringa di caratteri introdotta da tastiera

Azione

L'esecuzione del programma si ferma e sul display è visualizzato un punto interrogativo. L'utente può introdurre dei valori, separati da virgola, che sono assegnati nell'ordine con cui sono introdotti alle variabili presenti nella istruzione INPUT. Dopo aver introdotto tutti i valori richiesti dalla istruzione, premendo (END OF LINE) l'esecuzione del programma continua dall'istruzione esecutiva successiva.

Note

1. Se l'operatore introduce meno dati di quelli richiesti dalla istruzione INPUT sul display appare ?? ed il sistema attende l'introduzione dei restanti dati.

2. I dati introdotti da tastiera devono essere coerenti con il tipo di variabile cui sono assegnati, ma si può assegnare un numero ad una variabile stringa.

3. Se si introducono più dati di quanti richiesti dall'istruzione INPUT i dati in eccesso sono ignorati e l'esecuzione del programma continua. Sul display appare il messaggio TOO MUCH INPUT-EXCESS IGNORED.

4. Una stringa contenente la virgola, o spazi iniziali e finali, deve essere introdotta da tastiera tra virgolette.

5. Non si può introdurre il carattere virgolette (esso può essere introdotto con l'impiego dell'istruzione RKB).

6. E' opportuno far precedere l'istruzione INPUT da una istruzione DISP o PRINT per indicare quali dati devono essere introdotti.

7. Se una virgola è posta all'interno di un dato essa provoca l'assegnazione della prima parte del dato alla variabile ad esso associata e lo spostamento di tutte le assegnazioni successive.

8. Se in un programma vi è la seguente sequenza di istruzioni:

```
140   I  =  10
150  INPUT I, B (I)
```

   allora l'indice della variabile con indice è 10 e non il nuovo valore assegnato ad I da tastiera.

9. Se viene assegnata una stringa ad una variabile numerica, viene annullata solamente la linea contenente il dato errato e sul display appare il messaggio:

   INCORRECT FORMAT - RETYPE LINE

   L'operatore deve quindi reintrodurre l'intera linea. Anche nel caso in cui si assegna un valore numerico, con esponente in valore assoluto maggiore di 63, ad una variabile dichiarata in singola precisione, compare sul display il messaggio suddetto. L'operatore deve quindi reintrodurre l'intera linea. Infine, lo stesso messaggio appare sul display se tra un dato ed il successivo sono introdotte da tastiera due o più virgole; l'operatore può correggere la linea introdotta ripetendo l'introduzione dei dati.

10. Se si assegna ad una variabile stringa una stringa con un numero di caratteri maggiore di quello dichiarato per la variabile suddetta, è segnalato un errore ed il sistema commuta nello stato di debugging assegnando alla variabile la stringa introdotta troncata a destra dei caratteri eccedenti la lunghezza di allocazione della variabile stringa.

11. Quando il sistema è in attesa di dati da tastiera si può premere il tasto STEP per commutare il sistema nello stato di debugging. Premendo una seconda volta il tasto STEP oppure il tasto CONTINUE sul display riappare il punto interrogativo ed il sistema è sempre in attesa dell'introduzione del dato precedente. Se si era premuto CONTINUE dopo che i dati sono introdotti ed assegnati alle variabili l'esecuzione del programma continua, mentre se si era premuto STEP il sistema ricommuta nello stato di debugging e sul display appare il numero di linea della istruzione che sarà eseguita successivamente.

Esempi

1. Eseguendo cinque volte la routine sottostante si possono mettere in luce le seguenti situazioni. Si può assegnare ad una variabile stringa un valore numerico senza che sia incluso tra virgolette (vedi il numero 12 assegnato ad A$). Si può assegnare ad una variabile stringa una stringa contenente la virgola se la stringa è compresa tra virgolette (vedi AREA, PESO). Se non si introducono tutti i dati previsti da una istruzione INPUT, sul display appare la richiesta di attesa di altri dati da tastiera (??). Durante la terza esecuzione della istruzione INPUT si sono introdotti più dati di quelli richiesti (A,1,2,B) per cui l'esecuzione continua ed il sistema ignora i dati in eccedenza (B) visualizzando sul display il messaggio TOO MUCH INPUT-EXCESS IGNORED. Durante la quarta esecuzione della istruzione INPUT viene introdotto un valore (V) non coerente con il tipo di variabile per cui il sistema visualizza il messaggio INCORRECT FORMAT-RETYPE LINE ed attende una nuova introduzione. Durante la quinta esecuzione della istruzione INPUT si introducono due virgole di seguito per cui viene visualizzato di nuovo il messaggio suddetto. L'intera linea deve essere digitata di nuovo. Come terzo dato si è introdotta una stringa di 19 caratteri

per cui il sistema visualizza l'errore ERROR 8 IN LINE 20. Il sistema è nello stato di debugging (la luce del tasto STEP è accesa). Premendo CONTINUE l'esecuzione del programma viene portata a termine e la stringa è troncata dopo i primi 16 caratteri (B$ infatti ha una lunghezza di allocazione di 16 caratteri).

```
LIST
FILE    +INPUT


0010 LET B=1
0020 INPUT A$,A,B$
0030 PRINT "A$=";A$,"A=";A,"B$=";B$
0040 LET B=B+1
0050 IF B<=5 THEN 20
0060 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
12,12,"AREA,PESO"
A$=12            A= 12           B$=AREA,PESO
?
!!!!!!
??
126
??
######
A$=!!!!!!       A= 126          B$=######
?
A,1,2,B
TOO MUCH INPUT-EXCESS IGNORED
A$=A             A= 1            B$=2
?
F
??
U
INCORRECT FORMAT-RETYPE LINE
5,S
A$=F             A= 5            B$=S
?
Q,,1
INCORRECT FORMAT-RETYPE LINE
Q,1,12345678901234567890
ERROR 8   IN LINE 20
A$=Q             A= 1            B$=1234567890123456
```

2. Nella routine sottostante vi è una istruzione INPUT in cui compare una variabile che è utilizzata anche come indice di una successiva variabile con indice. Come si vede la variabile B (2) mantiene il valore precedente mentre il valore introdotto da tastiera (90) è assegnato alla variabile B (10).

```
LIST
FILE    +INPUT1

0010 DISP "Introduci i dati"
0020 LET B(2)=8
0030 LET I=10
0040 INPUT I,B(I)
0050 PRINT "I=";I,"B(10)=";B(10),"B(2)=";B(2)
0060 END

END OF LISTING


RUN
Introduci i dati
?
2,30
I= 2            B(10)= 30       B(2)= 8
```

3. Nell'esempio seguente si mostrano due esecuzioni della routine sottostante. Durante la prima esecuzione, quando viene eseguita l'istruzione INPUT si preme il tasto STEP. Il sistema è nello stato di debugging. Premendo il tasto STEP di nuovo sul display riappare il punto interrogativo e introducendo il valore l'istruzione INPUT viene portata a termine; premendo successivamente STEP il programma è eseguito passo a passo. Durante la seconda esecuzione, quando viene eseguita l'istruzione INPUT si preme il tasto STEP. Il sistema è nello stato di debugging. Questa volta si preme il tasto CONTINUE e sul display appare di nuovo il punto interrogativo. Introducendo il dato richiesto l'esecuzione del programma viene portata a termine.

```
LIST
FILE

0010 DCL S A
0020 INPUT A
0030 PRINT "A=";A
0040 END

END OF LISTING
```

```
RUN
#20
?
STEP        IN LINE 20
#20
?
124
STEP        IN LINE 30
#30
A= 124
STEP        IN LINE 40
#40


 RUN
 #20
 ?
 STEP        IN LINE 20
 #20
 ?
 124
 #30
 A= 124
 #40
```