INTERRUPT ENABLE

Istruzione
INTERRUPT ENABLE

Funzione

Consente la gestione da parte dell'utente di interruzioni interne ed esterne.

Formati

1. INTERRUPT ENABLE (I,funam)
2. INTERRUPT ENABLE (E,funam,intmsk,priort)

dove:

I

specifica che le interruzioni interne, cioè errori rilevati durante l'esecuzione, saranno gestiti dall'utente

E

specifica che le interruzioni esterne, cioè quelle generate da unità periferiche esterne saranno gestite dall'utente

Per maggiori informazioni sulla gestione di interruzioni esterne, cioè il formato qui contraddistinto dal numero 2, si faccia riferimento al manuale "P6066 I/O con periferiche esterne"

funam

è una costante stringa o una variabile stringa il cui valore può essere o un carattere alfabetico maiuscolo o un asterisco. Il carattere alfabetico specifica il nome di una definizione di funzione numerica multilinea.

Azione

L'esecuzione dell'istruzione con funam che ha per valore un carattere alfabetico fa sì che ogni successivo errore di sistema passi il controllo alla funzione multilinea richiamata dall'istruzione.

L'esecuzione dell'istruzione con funam che ha per va-

lore un asterisco fa sì che ogni successivo errore sia gestito dal sistema.

Note

1. L'azione richiesta dall'istruzione INTERRUPT ENABLE resta valida fino ad una successiva istruzione INTERRUPT ENABLE eseguita.

2. L'istruzione DEF della funzione multilinea deve avere come parametri due variabili numeriche semplici. Alla prima variabile viene assegnato come valore il codice d'errore che ha causato l'interruzione; alla seconda variabile viene assegnato come valore il numero di linea dell'istruzione in cui si è verificato l'errore. L'utente può quindi analizzare questi valori per determinare quale azione intraprendere.

3. La funzione multilinea deve assegnare un valore alla pseudo variabile FN*. Se il valore assegnato è 0, l'utente potrà gestire l'errore. Se il valore assegnato è diverso da 0, l'utente dovrà accettare l'azione intrapresa dal Sistema. In quest'ultimo caso, l'esecuzione del programma resta sospesa e ci sarà il messaggio:

   ERROR cod-errore IN LINE num. linea

   Si potrà poi decidere quale azione intraprendere a seconda se l'errore sarà o no recuperabile.

4. Se FN*= 0 e l'errore è recuperabile, l'esecuzione riprenderà, a completamento della funzione multilinea, con la prima istruzione eseguibile che segue l'istruzione che ha causato l'errore.

5. Se FN*= 0 e l'errore non è recuperabile, l'esecuzione riprenderà, a completamento della funzione multilinea, con l'istruzione che ha causato l'errore.

6. L'istruzione sarà ignorata nel caso dei seguenti errori non recuperabili:

   - 65 - non c'è spazio sufficiente in memoria utente per continuare l'esecuzione

   - 74 - è stato superato il numero massimo di riferimenti ad altre definizioni di funzione mono-

linea o multilinea all'interno di una definizione monolinea o multilinea (255).

7. Se insorge un errore durante l'esecuzione della funzione multilinea chiamata dall'istruzione INTERRUPT ENABLE, non è possibile apportare correzioni. Il sistema invierà il messaggio d'errore 282.

8. Vedere l'appendice D per l'elenco completo degli errori.

Esempi

1. L'esempio che segue mostra l'istruzione INTERRUPT ENABLE utilizzata per gestire una condizione di underflow (codice d'errore 4). L'underflow avviene al passo 96 del ciclo FOR/NEXT, ma l'esecuzione del programma non viene sospesa. Dopo aver cambiato il valore di A alla linea 90 e dopo che il controllo è ritornato all'istruzione successiva a quella che ha causato l'errore, il programma va avanti fino alla fine, senza che il sistema commuti in stato di debugging, come succederebbe se non si fosse utilizzata l'istruzione INTERRUPT ENABLE e la routine di gestione errori.

```
FILE     UKERR4

0005 INTERRUPT ENABLE (I,"A")
0010 LET L=100
0015 LET A=.0001
0020 FOR I=1 TO L STEP 1
0030 LET A=A/10
0040 NEXT I
0050 GOTO 140
0060 DEF FNA(C,L)
0070 IF C<>4 THEN 110
0080 PRINT "SI E' VERIFICATO UNDERFLOW IN LINEA ";L;"AL PASSO";I
0090 LET A=1E5
0095 LET FN*=0
0100 GOTO 130
0110 PRINT "SI E' VERIFICATO ERRORE ";C;" IN LINEA ";L
0120 LET FN*=1
0130 FNEND
0140 PRINT "A =",A
0150 END

END OF LISTING



SI E' VERIFICATO UNDERFLOW IN LINEA  30 AL PASSO 96
A = 10
```

2. In questo esempio è stata eliminata la linea 15 del programma precedente. Il primo errore che si incontra è il codice di errore 1 "non è stato assegnato alcun valore ad una variabile numerica". L'istruzione PRINT alla linea 110 tiene conto dell'errore. L'istruzione FN* della linea 120 richiede poi la normale gestione degli errori da parte del sistema. Come risultato si avrà la sospensione dell'esecuzione del programma e il sistema che commuterà in stato di debugging.

```
FILE     UKERR4

0005 INTERRUPT ENABLE (I,"A")
0010 LET L=100
0020 FOR I=1 TO L STEP 1
0030 LET A=A/10
0040 NEXT I
0050 GOTO 140
0060 DEF FNA(C,L)
0070 IF C<>4 THEN 110
0080 PRINT "SI E' VERIFICATO UNDERFLOW IN LINEA ";L;"AL PASSO";I
0090 LET A=1E5
0095 LET FN*=0
0100 GOTO 130
0110 PRINT "SI E' VERIFICATO ERRORE ";C;" IN LINEA ";L
0120 LET FN*=1
0130 FNEND
0140 PRINT "A =";A
0150 END

END OF LISTING




SI E' VERIFICATO ERRORE  1  IN LINEA  30
```

Istruzione LET

Funzione

Assegna i valori alle variabili di programma.

Formato

[LET] { **num-var [= num-var] ... = num-exp** / **string-var [= string-var] ... = string-exp** }

dove:

num-var
: è una variabile numerica (semplice o con indice) a cui viene assegnato il valore della espressione numerica specificata alla destra dell'ultimo segno uguale (ultimo da sinistra)

num-exp
: è una espressione numerica che specifica il valore da assegnare alla variabile od alle variabili indicate alla sinistra dell'ultimo segno uguale

string-var
: è una variabile stringa (semplice o con indice) a cui viene assegnato il valore della espressione stringa specificata alla destra dell'ultimo segno uguale

string-exp
: è una espressione stringa che specifica il valore da assegnare alla variabile od alle variabili indicate alla sinistra dell'ultimo segno uguale

Azione

L'espressione a destra dell'ultimo segno uguale è eseguita ed il valore ottenuto è assegnato alle variabili indicate alla sinistra dello stesso segno.

Note

1. La parola chiave LET è opzionale.

2. Il numero di variabili a sinistra del segno di assegnazione è limitato solamente dal numero massimo di caratteri che possono comporre una linea (80).

3. La lunghezza attuale di una variabile stringa presente in una istruzione di assegnazione diventa uguale al numero di caratteri che costituiscono la stringa risultato della espressione stringa indicata.

4. Se la lunghezza di allocazione di una variabile stringa è inferiore al numero di caratteri della stringa risultante dalla esecuzione della espressione stringa, quest'ultima viene troncata dei caratteri eccedenti. Il sistema commuta nello stato di debugging e viene segnalato un errore di tipo recuperabile.

5. L'indice di una variabile con indice che compare in una istruzione LET può essere espresso, in generale, mediante una espressione numerica. Se l'espressione suddetta contiene delle variabili i cui valori sono modificati durante l'esecuzione della istruzione LET, l'espressione viene valutata con i valori che le variabili avevano prima della esecuzione della istruzione LET (vedi l'esempio più avanti).

6. Se ad una variabile numerica in singola precisione viene assegnato un valore nella zona di OVERFLOW per la singola precisione, la variabile suddetta assume il valore 9.99999E63 oppure -9.99999E63. Il sistema è nello stato di debugging e viene visualizzato un errore di tipo recuperabile.

7. Se il risultato del calcolo di una espressione numerica è un valore nella zona di OVERFLOW per la doppia precisione, alla variabile in doppia precisione, specificata alla sinistra del segno uguale, viene assegnato il valore 9.999999999999E99 oppure -9.999999999999E99. Il sistema è nello stato di debugging ed è visualizzato un errore di tipo recuperabile.

8. Se ad una variabile numerica è assegnato un valore nella zona diUNDERFLOW (relativa al tipo di precisione con cui è dichiarata) allora la variabile suddetta assume il valore zero. Il sistema è nello stato di debugging e viene visualizzato un errore di tipo recuperabile.

9. Se nella espressione numerica specificata con num-exp compare una variabile a cui non è ancora stato

assegnato un valore, l'espressione è valutata assegnando alla variabile suddetta il valore zero. Il sistema commuta nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile.

10. Se nella espressione stringa specificata con string-exp compare una variabile a cui non è stato ancora assegnato un valore, l'espressione è valutata assegnando alla variabile suddetta il valore di stringa nulla. Il sistema commuta nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile.

Esempi

1. Nella routine sottostante possiamo notare che con una istruzione LET si può assegnare contemporaneamente lo stesso valore a 33 variabili. La variabile A è quindi utilizzata come un contatore. Infine il valore della variabile A1 è modificato assegnandole il valore della espressione specificata a destra del segno uguale nella istruzione 80.

```
LIST
FILE    +LET

0010A=B=C=D=E=F=G=H=G=J=K=L=M=N=O=P=Q=R=S=T=U=V=W=Z=A0=A1=A2=A3=A4=A5=A6=A7=A8=4
0020 PRINT A,B,C,D,G
0030 LET A=0
0040 LET A=A+1
0050 PRINT A
0060 IF A<=4 THEN 40
0070 PRINT "A1=";A1
0080 LET A1=PI*A5+SQR(P*Q+R-5)/LGT(W*Z)
0090 PRINT "A1=";A1
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 4                4                4                4                4
 1
 2
 3
 4
 5
A1= 4
A1= 15.782814
```

2. Vediamo alcuni esempi di impiego della istruzione LET per assegnare un valore ad una variabile stringa. Si veda, istruzioni 150 ÷ 180, come si possono aggiungere via via dei caratteri al valore di una variabile stringa.

```
LIST
FILE    *LET1


0010 PRINT
0020 PRINT
0030 LET A$="I capitoli"
0040 LET B$=" 1 "
0050 LET C$=" 2 "
0060 LET D$=" 3 "
0070 LET E$=" 4 e 5 "
0080 DCL 80 (F$,G$,H$)
0090 LET F$=" descrivono il sistema."
0100 LET G$="descrivono il linguaggio."
0110 LET H$=A$+B$+",)"+C$+"e"+D$+F$
0120 PRINT "H$=";H$
0130 LET H$=A$+E$+G$
0140 PRINT "H$=";H$
0150 LET S$=""
0160 FOR I=0 TO 9 STEP 1
0170 LET S$=S$+CHR$(I)
0180 NEXT I
0190 PRINT "S$=";S$
0200 END

END OF LISTING



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


H$=I capitoli 1 , 2 e 3  descrivono il sistema.
H$=I capitoli 4 e 5 descrivono il linguaggio.
S$=■┌└┘≥■◦o↖→
```

Istruzione NEXT

Funzione

Definisce il termine di un ciclo iterativo.

Formato

NEXT control-var

dove:

control-var

è la variabile semplice numerica specificata come variabile di controllo nella corrispondente istruzione FOR.

Azione

Vedi istruzione FOR.

Istruzione ON...GOSUB

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma scelto tra un insieme di sottoprogrammi in funzione del valore assunto da una espressione specificata.

Formato

ON num-exp GOSUB line-num [, line-num] ...

dove:

num-exp

indica un espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica quale sottoprogramma deve essere eseguito tra quelli che iniziano con l'istruzione il cui numero di linea è specificato dopo la parola chiave GOSUB

line-num

è il numero di linea della prima istruzione di un sottoprogramma

Azione

L'espressione numerica è eseguita ed il suo valore arrotondato all'intero più prossimo. Il controllo della esecuzione del progamma è trasferito all'istruzione il cui numero di linea si trova nella istruzione ON... GOSUB nella posizione (da sinistra a destra) indicata dal numero ottenuto come risultato della espressione numerica di cui sopra. Così una espressione il cui valore è 3.85, trasferisce il controllo alla istruzione il cui numero di linea è indicato al quarto posto nell'insieme dei numeri di linea che costituiscono gli operandi della istruzione ON...GOSUB. Ognuna delle istruzioni il cui numero di linea è indicato nella istruzione ON...GOSUB è la prima di un insieme di istruzioni BASIC che costituiscono altrettanti sottoprogrammi. L'ultima istruzione di ogni sottoprogramma è l'istruzione RETURN che trasferisce il controllo della esecuzione del progamma alla prima istruzione esecuti-

va (in ordine di numero di linea) successiva alla istruzione ON...GOSUB.

Note

1. In ogni sottoprogramma vi può essere più di una istruzione RETURN.

2. I sottoprogrammi possono far parte di una funzione multilinea: in questo caso anche l'istruzione ON...GOSUB deve essere un'istruzione della funzione multilinea.

3. Se il valore della espressione numerica, approssimato all'intero più prossimo, è minore di 1 o maggiore del numero "numeri di linea" presenti nella istruzione, l'esecuzione del programma prosegue dall'istruzione esecutiva successiva alla istruzione ON...GOSUB.

4. Una istruzione ON...GOSUB può essere contenuta in un ciclo FOR/NEXT, ma i sottoprogrammi non possono essere contenuti in un ciclo FOR/NEXT.

5. Per ulteriori note si veda l'istruzione GO SUB.

Esempio

Nel seguente programma sono riportati quattro sottoprogrammi a cui si fa riferimento mediante una istruzione ON...GOSUB. Si osservi, attraverso le diverse esecuzioni, come il controllo della esecuzione del programma dipenda essenzialmente dal valore della espressione A*B.

```
LIST
FILE    *ONGOS

0010 PRINT "Ecco un esempio di programma che utilizza piu` sottoprogrammi."
0020 LET A=1
0030 DISP "Scegli il sottoprogramma                    ";
0040 INPUT B
0044 PRINT
0046 PRINT "A*B=";A*B
0048 PRINT
0050 ON A*B GOSUB 100,200,300,400
0060 IF A=2 THEN 430
0070 PRINT "Non e` stato eseguito alcun sottoprogramma!"
0080 GOTO 430
0100 PRINT "E` stato eseguito il primo sottoprogramma!"
0110 LET A=2
0120 RETURN
0200 PRINT "E` stato eseguito il secondo sottoprogramma!"
0210 LET A=2
0220 RETURN
```

```
0300 PRINT "E` stato eseguito il terzo sottoprogramma!"
0310 LET A=2
0320 RETURN
0400 PRINT "E` stato eseguito il quarto sottoprogramma!"
0410 LET A=2
0420 RETURN
0430 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
-15

A*B=-15

Non e` stato eseguito alcun sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
1.45

A*B= 1.45

E` stato eseguito il primo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
1.56

A*B= 1.56

E` stato eseguito il secondo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
3

A*B= 3

E` stato eseguito il terzo sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
4

A*B= 4

E` stato eseguito il quarto sottoprogramma!
RUN
Ecco un esempio di programma che utilizza piu` sottoprogrammi.
Scegli il sottoprogramma                          ?
4.56

A*B= 4.56

Non e` stato eseguito alcun sottoprogramma!
```

ON ... GOTO

Istruzione ON...GOTO

Funzione

Trasferisce il controllo dell'esecuzione di un programma ad una istruzione scelta tra un insieme di istruzioni in funzione del valore assunto da una espressione specificata.

Formato

ON num-exp GOTO line-num [, line-num] ...

dove:

num-exp
: è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica a quale istruzione di programma, tra quelle il cui numero di linea è specificato dopo la parola chiave GOTO, deve essere trasferito il controllo della esecuzione

line-num
: indica il numero di linea di una istruzione del programma

Azione

L'espressione numerica è eseguita ed il suo valore arrotondato all'intero più prossimo. Il controllo della esecuzione del programma è trasferito alla istruzione il cui numero di linea si trova nella istruzione ON...GOTO nella posizione (da sinistra a destra) indicata dal numero ottenuto come risultato della espressione numerica di cui sopra. Così una espressione il cui valore è 2.75, trasferisce il controllo della esecuzione del programma alla istruzione il cui numero di linea è indicato al terzo posto nell'insieme dei numeri di linea che costituiscono gli operandi della istruzione ON...GOTO.

Note

1. L'istruzione ON...GOTO può trasferire il controllo della esecuzione del programma ad una istruzione di una funzione multilinea, se anche l'istruzione

ON...GOTO fa parte dell'insieme di istruzioni della funzione multilinea.

2. Se il valore della espressione approssimato all'intero più prossimo è minore di 1 o maggiore del numero di "numeri di linea" presenti nella istruzione, l'esecuzione del programma prosegue dalla istruzione esecutiva successiva alla istruzione ON...GOTO.

3. Se l'istruzione ON...GOTO è compresa in un ciclo FOR/NEXT anche le istruzioni i cui numeri di linea sono indicati nella istruzione stessa devono far parte del ciclo FOR/NEXT.

4. Se l'istruzione ON...GOTO è esterna ad un ciclo FOR/NEXT i numeri di linea specificati come operandi non possono appartenere ad istruzioni interne al ciclo suddetto.

Esempio

Il programma sottostante mostra come si può utilizzare l'istruzione ON...GOTO in un ciclo FOR/NEXT. Si hanno cinque esecuzioni della suddetta istruzione per ogni esecuzione completa del programma. Sono stati forniti, in successione, i valori 0, 1.23, 2., 2.8, 4 e 89 che permettono di esemplificare tutti i casi possibili per il controllo del corretto funzionamento del programma stesso.

```
LIST
FILE

0010 PRINT "Ecco un esempio di programma che utilizza l'istruzione ON...GOTO.
0015 FOR I=1 TO 6 STEP 1
0020 DISP "Scegli l'instruzione da eseguire.                ";
0030 INPUT A
0040 ON A GOTO 100,200,300,400
0050 PRINT "Hai scelto di terminare subito il programma."
0060 GOTO 420
0100 PRINT "Hai scelto di eseguire la prima istruzione del set!"
0110 GOTO 420
0200 PRINT "Hai scelto di eseguire la seconda istruzione del set!"
0210 GOTO 420
0300 PRINT "Hai scelto di eseguire la terza istruzione del set!"
0310 GOTO 420
0400 PRINT "Hai scelto di eseguire la quarta istruzione del set!"
0410 GOTO 420
0420 NEXT I
0450 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio di programma che utilizza l'istruzione ON...GOTO.
Scegli l'instruzione da eseguire.                ?
0
Hai scelto di terminare subito il programma.
Scegli l'instruzione da eseguire.                ?
1.23
Hai scelto di eseguire la prima istruzione del set!
Scegli l'instruzione da eseguire.                ?
2.
Hai scelto di eseguire la seconda istruzione del set!
Scegli l'instruzione da eseguire.                ?
2.8
Hai scelto di eseguire la terza istruzione del set!
Scegli l'instruzione da eseguire.                ?
4
Hai scelto di eseguire la quarta istruzione del set!
Scegli l'instruzione da eseguire.                ?
89
Hai scelto di terminare subito il programma.
```

Istruzione PAD

Funzione

Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione aggiungendo in coda dei caratteri predefiniti.

Formato

PAD string-var, n

dove:

string-var
è una variabile stringa alla quale sono aggiunti come caratteri di riempimento, fino ad eguagliare la dimensione attuale della variabile con quella di allocazione, i caratteri ISO corrispondenti al numero intero n

n
è un numero intero compreso tra Ø e 255 che indica quale carattere ISO deve essere utilizzato come carattere di riempimento

Azione

Alla variabile stringa string-var sono aggiunti caratteri di riempimento corrispondenti, secondo la tabella ISO (vedi appendice E), al numero n, finchè la sua lunghezza attuale eguagli la lunghezza di allocazione.

Nota

L'istruzione PAD permette la generazione di record di dati con lunghezza prefissata.

Esempi

1. Nell'esempio che segue si vede che l'istruzione PAD è una istruzione di assegnazione del carattere specificato alla variabile indicata. Se infatti alla variabile A$ non si assegna alcun valore, ad essa viene assegnata, istruzione 20, una stringa di tanti asterischi (valore corrispondente 42) quanto è la lunghezza di allocazione (in questo caso 16 caratteri).

```
LIST
FILE


0010 PAD A$,42
0011 PRINT
0012 PRINT
0013 PRINT
0020 PRINT "A$=";A$
0030 END

END OF LISTING


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


A$=****************
```

2. Ecco un tipico esempio d'impiego dell'istruzione PAD per la creazione di record di eguale lunghezza da registrare su file dati esterno.

```
LIST
FILE    +PAD


0010 FILES A
0020 SCRATCH :1
0030 FOR I=1 TO 10 STEP 1
0040 DISP "Introduci il record
0050 INPUT A$
0060 PAD A$,42
0070 WRITE :1,A$
0080 NEXT I
0090 PRINT
0100 PRINT
0110 PRINT
0120 PRINT "Ecco i dati registrati nel file A;prima stampo tutto il record "
0130 PRINT "e poi solo il suo contenuto."
0140 PRINT
0150 PRINT
0160 RESTORE :1
0170 FOR I=1 TO 10 STEP 1
0180 READ :1,B$
0190 PRINT B$
0200 DEPAD B$,42
0210 PRINT B$
0220 NEXT I
0230 END

END OF LISTING
```

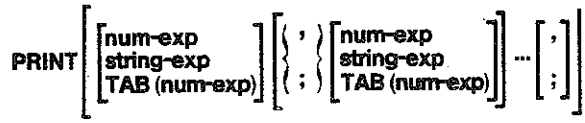
```
RUN
Introduci il record              ;
?
Mario
Introduci il record              ;
?
Piero
Introduci il record              ;
?
Giovanni
Introduci il record              ;
?
Giacomo
Introduci il record              ;
?
Nicola
Introduci il record              ;
?
Enzo
Introduci il record              ;
?
Giulio
Introduci il record              ;
?
Enrico
Introduci il record              ;
?
Sandro
Introduci il record              ;
?
Carlo


Ecco i dati registrati nel file A;prima stampo tutto il record
e poi solo il suo contenuto.


Mario**********
Mario
Piero**********
Piero
Giovanni*********
Giovanni
Giacomo*********
Giacomo
Nicola**********
Nicola
Enzo***********
Enzo
Giulio**********
Giulio
Enrico**********
Enrico
Sandro**********
Sandro
Carlo***********
Carlo
```

Istruzione PRINT

Funzione

Stampa dati e testi in un formato standard.

Formato



dove:

num-exp
indica una espressione numerica il cui valore deve essere stampato

string-exp
indica una espressione stringa il cui valore deve essere stampato

TAB (num-exp)
è una funzione di sistema che indica la posizione, corrispondente al valore di num-exp arrotondato all'intero più prossimo, in cui si deve posizionare il pointer del buffer di stampa

,
indica uno spostamento standard per il pointer del buffer di stampa

;
indica che il pointer del buffer di stampa deve rimanere nella posizione in cui si trova

Azione

I valori delle espressioni specificate nella istruzione sono stampati, nell'ordine con cui compaiono nella istruzione, con il formato descritto nelle note successive.

La posizione nella linea di stampa dei caratteri che sono stampati è controllata mediante l'impiego della virgola (,) del punto e virgola (;) e della funzione TAB, come specificato nel paragrafo "Controllo della posizione dei caratteri nel buffer di stampa".

Note

1. Il valore di una espressione stringa viene stampato con la sequenza dei caratteri che la compongono.

2. Il valore di una espressione numerica viene stampato anteponendo ad esso uno spazio (se è un numero positivo) od il segno meno (se è un numero negativo) ed aggiungendo in coda ad esso un altro spazio.

3. I numeri interi sono stampati con al massimo 8 cifre significative.

4. I numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono stampati, con al massimo 8 cifre significative (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale), nel formato in virgola fissa.

5. I numeri con valore assoluto minore di 0.0999999995 che possono essere rappresentati con 8 cifre significative, sono stampati nel formato in virgola fissa.

6. Tutti gli altri numeri sono stampati nel formato in virgola mobile.

7. Le costanti stringa devono essere comprese tra virgolette.

8. Virgola e punto e virgola sono elementi separatori e finali. Sono obbligatori come elementi separatori fra le espressioni specificate come operandi. Sono opzionali come elementi finali.

9. Una istruzione PRINT senza operandi stampa il contenuto del buffer di stampa (vedi buffer di stampa più avanti) e pone il pointer nella prima posizione. Se il buffer non contiene caratteri l'effetto prodotto è una interlinea.

10. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), l'istruzione PRINT visualizza sul display le informazioni relative alle espressioni in essa specificate. Ad ogni visualizzazione l'esecuzione del programma si interrompe e riprende premendo il tasto CONTINUE. Per leggere le informazioni prodotte in una linea si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel

capitolo 1, par. "La tastiera".
Se le informazioni prodotte occupano più di una linea, le informazioni relative ad ogni linea successiva alla prima sono visualizzate premendo ogni volta CONTINUE. Quando, premendo il tasto CONTINUE, il sistema emette una segnale acustico, l'ultima linea visualizzata è la fine del testo prodotto dall'istruzione PRINT.

Controllo della posizione dei caratteri nel buffer di stampa

L'esecuzione di una istruzione PRINT genera in un registro di 80 caratteri, detto buffer di stampa (vedi figura 5-2), la stringa di caratteri corrispondenti al valore dell' espressione che compare nella istruzione stessa. Un secondo registro, detto pointer, indica la posizione del buffer in cui deve iniziare la stringa di caratteri suddetta. Ogni volta che un carattere è generato nel buffer di stampa, il pointer avanza di una posizione. Quando il buffer di stampa è pieno, il suo contenuto viene stampato sulla riga di stampa ed il pointer è rimesso ad 1 (indica la prima posizione del buffer).

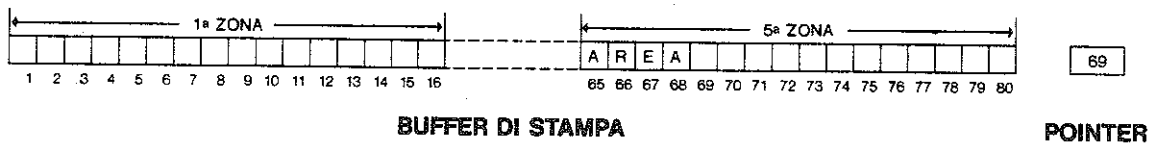


Figura 5-2 Buffer di stampa e relativo pointer

Ogni volta che è eseguita una istruzione PRINT senza virgola o punto e virgola come elementi finali, il contenuto del buffer di stampa è stampato ed il pointer di stampa è rimesso ad 1.

L'impiego della funzione TAB (num-exp) permette il controllo della posizione dei caratteri nel buffer di stampa. Infatti il valore dell'espressione numerica num-exp è arrotondato all'intero più prossimo e assegnato al pointer del buffer di stampa. Se il valore suddetto è minore di uno viene segnalato un errore.

Se il valore suddetto è maggiore di 80, è ridotto modulo 80; ossia è posto uguale ad n-80*INT((n-1)/80), dove n è il valore, arrotondato all'intero più prossimo, della espressione numerica num-exp. Se il pointer del buffer di stampa indica una posizione superiore al valore ritornato dalla funzione TAB, il contenuto del buffer è stampato ed il pointer indica la posizione ad esso assegnata in seguito alla valutazione di TAB. Per essere sicuri che i caratteri successivi siano generati a partire dalla posizione assegnata al pointer da TAB (num-exp) è bene far seguire tale funzione da un punto e virgola (;).

L'impiego della virgola (,) permette la stampa dei dati a distanza standard, di 16 posizioni, l'una dall'altra: il buffer di stampa è diviso in 5 zone, ognuna di 16 posizioni (vedi figura 5-2). In un programma con le seguenti istruzioni:

```
50 PRINT ,
60 PRINT "A"
```

l'istruzione 50 pone il pointer del buffer di stampa in posizione 17 per cui la successiva istruzione genera il carattere A nella posizione 17 del buffer medesimo. In un programma con le seguenti istruzioni:

```
50 PRINT "A", "B", "C", "D"
60 PRINT "E",
70 PRINT "F"
```

l'istruzione 50 genera nel buffer di stampa il carattere A nella prima posizione, il carattere B nella 17-esima posizione, il carattere C nella 33-esima posizione, il carattere D nella 49-esima posizione e li stampa nelle posizioni suddette. L'istruzione 60 genera il carattere E nella prima posizione del buffer di stampa. Il pointer indirizza la 17-esima posizione del buffer di stampa, per cui la successiva istruzione PRINT genera il carattere F nella posizione suddetta. Si noti che se, quando è eseguita una istruzione PRINT riferita ad una virgola, il pointer di stampa indica una posizione nella 5ª zona, il contenuto del buffer di stampa è stampato ed il pointer indirizza la posizione uno.

L'impiego del punto e virgola (;) permette di accodare nel buffer di stampa due o più valori corrispondenti ad espressioni distinte. Se in un programma si hanno le istruzioni:

```
20 PRINT "A"; "B"
30 PRINT "C";
40 PRINT "D"
```

l'istruzione 20 genera i caratteri A e B uno di seguito all'altro. Le istruzioni 30 e 40 producono lo stesso effetto per C e per D.

Si osservi che se il valore corrispondente ad una espressione specificata in una istruzione PRINT è tale per cui il numero dei caratteri ad esso corrispondente è superiore al numero di posizioni rimaste libere nel buffer di stampa, il contenuto del buffer di stampa è stampato ed i caratteri suddetti sono generati dall'inizio del buffer. Se la stringa suddetta è composta da più di 80 caratteri è stampata su più linee andando a campo ogni 80 caratteri.

Nel seguito diamo una tabella che riassume l'impiego della virgola e del punto e virgola come elementi separatori in una istruzione di tipo PRINT. Il separatore menzionato segue il dato, specificato nella prima colonna, nella istruzione PRINT relativa.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di stampa	Valore del pointer dopo la generazione dei caratteri nel buffer di stampa
Espressione numerica	","	Il valore corrispondente all'espressione è generato nel buffer di stampa dalla posizione indicata dal pointer, se le posizioni libere nel buffer sono sufficienti a contenerlo, altrimenti il contenuto del buffer è stampato ed il valore è generato nel buffer a partire dalla prima posizione.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer di stampa, il contenuto del buffer è stampato ed il valore del pointer è 1.
	";" punto e virgola		Il pointer indica la posizione successiva all'ultima posizione occupata dal valore generato nel buffer.
Espressione stringa	"," virgola	La stringa corrispondente all'espressione è generata nel buffer di stampa dalla posizione indicata dal pointer, se le posizioni libere nel buffer sono sufficienti a contenerla, altrimenti il contenuto del buffer è stampato e la stringa è generata nel buffer a partire dalla prima posizione. Se la stringa ha più di 80 caratteri, ogni volta che il buffer è pieno, il suo contenuto è stampato ed i rimanenti caratteri della stringa sono generati dall'inizio del buffer.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer di stampa, il contenuto del buffer è stampato ed il valore del pointer è 1.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di stampa	Valore del pointer dopo la generazione dei caratteri nel buffer di stampa
Espressione stringa	";" punto e virgola		Il pointer indica la posizione successiva all'ultima posizione occupata dalla stringa generata nel buffer.
stringa nulla	"," virgola	Non viene generato alcun carattere nel buffer di stampa.	Il valore del pointer avanza delle rimanenti posizioni della zona di stampa. Se viene raggiunta la fine del buffer, il contenuto del buffer di stampa è stampato ed il valore del pointer è 1.
	";" punto e virgola		Il valore del pointer non è modificato.

Tabella 5-2 Impiego della virgola e del punto e virgola con PRINT

Esempi

1. Nella routine seguente si può notare la funzione di tabulatore standard della virgola (istruzione 70), come vengono accodati i dati con il punto e virgola ed infine come la istruzione END fa stampare il contenuto del buffer di stampa (quando è eseguita l'istruzione DISP i dati sono ancora nel buffer di stampa; è l'istruzione END, in questo caso, che li trasmette alla stampante).

```
LIST
FILE    *PRINT


0010 PRINT
0020 PRINT
0030 PRINT "Per poter controllare la posizione dei caratteri nella linea, "
0040 PRINT " si osservi la seguente stampa:"
0050 PRINT "12345678901234567890123456789012345678901234567890123456789012345678901234567"
0060 PRINT
0070 PRINT ,,123456789,,12
0080 PRINT 45E88;45E88;45E88;-45E88;-45E88
0090 PRINT -89E-78;-34E-56;
0100 DISP "I dati della istruzione 90 sono ancora nel buffer di stampa!"
0110 DELAY 300
0120 END

END OF LISTING
```

```
RUN

Per poter controllare la posizione dei caratteri nella linea,
 si osservi la seguente stampa:
12345678901234567890123456789012345678901234567890123456

                                    1.2345679E+08                    12
 4.5000000E+89  4.5000000E+89  4.5000000E+89 -4.5000000E+89 -4.5000000E+89
I dati della istruzione 90 sono ancora nel buffer di stampa!
-8.9000000E-77 -3.4000000E-55
```

2. Vediamo come i numeri stampati in virgola mobile sono distanziati diversamente se separati da virgola o da punto e virgola. Vediamo infine come si possono effettuare più interlinee utilizzando solo tre istruzioni 60, 70 ed 80.

```
LIST
FILE    *PRINT2

0010 PRINT -12345678E-78;-12345678E-89;-12345678E-78;-12345678E-56;-12345678E-56
0020 PRINT -12345678912E-56,
0030 PRINT -12345678912E-23,-123456789E-62,-1234567896E-78
0040 PRINT 12
0050 PRINT " ECCO COME SI POSSONO FARE 10 INTERLINEE:"
0060 FOR I=1 TO 10 STEP 1
0070 PRINT
0080 NEXT I
0090 PRINT "SONO STATE FATTE 10 INTERLINEE!"
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
-1.2345678E-71 -1.2345678E-82 -1.2345678E-71 -1.2345678E-49 -1.2345678E-49
-1.2345679E-46  -1.2345679E-13  -1.2345679E-54  -1.2345679E-69
 12
 ECCO COME SI POSSONO FARE 10 INTERLINEE:




SONO STATE FATTE 10 INTERLINEE!
```

3. Come si può vedere dalla esecuzione della routine sottostante, se il contenuto di una variabile di cui si comanda la stampa è superiore ad 80 caratteri, viene generato un a capo ogni 80 caratteri.

```
LIST
FILE


0010 DCL 256 A$
0020 LET A$=""
0030 FOR I=0 TO 255 STEP 1
0040 LET A$=A$+CHR$(I)
0050 NEXT I
0060 PRINT A$
0070 END

END OF LISTING


RUN
ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖÜ¢£¥₧ƒáíóúñÑªº¿⌐¬½¼¡«» !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}~⌂
```

4. In questo esempio, istruzione 40, il dato è stampato dall'inizio della linea successiva poichè è composto da più di 16 caratteri (lunghezza di una zona di stampa).

```
LIST
FILE


0010 PRINT "OLIVETTI"
0020 PRINT ,,,,,
0030 PRINT "P6066"
0035 PRINT 1,2,3,4,5
0040 PRINT 1,2,3,4,"Questa stringa e` piu` lunga della zona di stampa "
0050 END

END OF LISTING


RUN
OLIVETTI

P6066
 1              2              3              4              5
 1              2              3              4
Questa stringa e` piu` lunga della zona di stampa
```

5. In questo esempio si vede che come operàndo di una istruzione PRINT si può avere una espressione stringa (istruzione 50) od una espressione numerica (istruzione 110).

```
LIST
FILE     PRINT


0010 DCL 70 A$,40(B$,C$)
0020 LET B$="Il mese di agosto e` caldo."
0030 LET C$="Il mese di febbraio e` freddo."
0034 PRINT
0035 PRINT
0036 PRINT
0040 LET A$="La primavera e` la stagione piu` bella dell`anno!"
0050 PRINT EXT$(B$+A$+C$,28,76)
0060 LET A=10
0070 LET B=3156
0080 LET C=A*B
0090 PRINT
0100 PRINT
0110 PRINT (A*SQR(B*LGT(C))-10)/LOG(A)
0120 END

END OF LISTING


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


La primavera e` la stagione piu` bella dell`anno!


 513.16547
```

Istruzione PRINT USING

Funzione

Stampa dati e testi in un formato predefinito in una istruzione immagine.

Formato

PRINT USING {line-num | string-var} , {num-exp | string-exp} [, {num-exp | string-exp}] ...

dove:

line-num
: indica il numero di linea di una istruzione immagine

string-var
: indica una variabile stringa il cui contenuto è la immagine con cui i valori specificati nell'istruzione devono essere stampati

num-exp
: è una espressione numerica il cui valore deve essere stampato

string-exp
: è una espressione stringa il cui valore deve essere stampato

Azione

I valori delle espressioni sono convertiti nel formato specificato nella istruzione immagine di formatc il cui numero di linea è indicato con line-num o nel formato specificato con il contenuto della variabile stringa string-var; sono quindi stampati da sinistra a destra nell'ordine in cui compaiono nella istruzione.

L'associazione tra i valori da stampare ed i campi della immagine di formato è fatta da sinistra a destra nell'ordine con cui i valori compaiono nell'istruzione ed i campi immagine nella immagine di formato.

Note

1. Ogni istruzione PRINT USING stampa i valori delle espressioni presenti nella istruzione a partire da una nuova riga di stampa, anche se una precedente istruzione PRINT terminava con "," o ";".

2. Se vi sono più valori da stampare, nella PRINT USING, che campi di formato, nella immagine di formato, i valori in più sono stampati sulle righe di stampa successive con lo stesso formatc.

3. Se vi sono più campi di formato, nella immagine di formato, che valori da stampare, nella PRINT USING, in corrispondenza dei campi immagine eccedenti vengono generati degli spazi.

4. I valori da stampare, nella PRINT USING, ed i campi di formato, nella immagine di formato, devono essere coerenti: ad un valore numerico deve corrispondere un campo di formato numerico etc..

5. Le costanti stringa si devono specificare racchiuse tra virgolette.

6. Per ulteriori informazioni si veda l'istruzione IMMAGINE.

7. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), l'istruzione PRINT visualizza sul display le informazioni relative alle espressioni in essa specificate. Ad ogni visualizzazione l'esecuzione del programma si interrompe e riprende premendo il tasto CONTINUE. Per leggere le informazioni prodotte in una linea si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel capitolo 1, par. "La tastiera". Se le informazioni prodotte occupano più di una linea, le informazioni relative ad ogni linea successiva alla prima sono visualizzate premendo ogni volta CONTINUE. Quando, premendo il tasto CONTINUE, il sistema emette un segnale acustico la linea visualizzata è l'ultima del testo prodotto dalla istruzione PRINT.

Esempi

1. Dalla esecuzione della seguente routine vediamo come vengono stampati i dati relativi ai campi di tipo numerico.

```
LIST
FILE    +PUSING


0010 PRINT
0020 PRINT
0030 PRINT
0040 FOR I=1 TO 4 STEP 1
0050 DCL 80A$
0060 DISP "Introduci la IMMAGINE!      ";
0070 INPUT A$
0080 DISP
0090 DISP "Introduci i dati,             ";
0100 INPUT B,C,D
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT USING A$,B,C,D
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT
0200 NEXT I
0210 END

END OF LISTING


RUN


Introduci la IMMAGINE!      ?
Numero intero:   ############################        **        ####

Introduci i dati,             ?
-1234567890123E11,-9,123.87



Numero intero:   -12345678901230000000000        -9         123



Introduci la IMMAGINE!      ?
Numero decimale:  ##.####################        ##.#              ####.

Introduci i dati,             ?
0.123456789123E-10,0.5,123.87



Numero decimale:   0.000000000123456789         0.5               124.



Introduci la IMMAGINE!      ?
Numero esponenziale:  ####################.##########   ##.↑↑↑↑       ####.↑↑↑↑

Introduci i dati,             ?
-12345678.12345E-15,-5E-15,123.87E12
```

```
Numero esponenziale:  -1234567812345000000.00000E-26   -5.E-15        124.E+12


Introduci la IMMAGINE!      ?
Numero con segno $:   $$$$$$$$$$$$$$$$$$$$$$$$$$     $$$       $$$$$.

Introduci i dati.             ?
123456890123,-5,123.87



Numero con segno $:               $123456890123     $-5       $124.
```

* Per ulteriori esempi si veda l'istruzione IMMAGINE *

2. Vediamo un esempio d'impiego della funzione TAB.

```
LIST
FILE     TAB


0010 PRINT "Ecco un esempio della funzione TAB nell'istruzione PRINT"
0020 PRINT "*** Ricorda di racchiudere tra parentesi l'argomento di TAB ***"
0030 PRINT TAB(5);"A";TAB(20);"B"
0040 PRINT TAB(7);"A";TAB(22);"B"
0050 PRINT TAB(9);"A";TAB(24);"B"
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Ecco un esempio della funzione TAB nell'istruzione PRINT
*** Ricorda di racchiudere tra parentesi l'argomento di TAB ***
    A               B
      A               B
        A               B
```

Istruzione RANDOMIZE

Funzione	Permette la generazione di numeri casuali.
Formato	**RAN [DOMIZE]**
Azione	L'istruzione RANDOMIZE, posta prima di una istruzione che richiama la funzione di sistema RND, fa in modo che i valori numerici ritornati da RND siano casuali non solo nell'ambito della esecuzione del programma, ma anche tra diverse esecuzioni dello stesso programma.
Nota	Quando si reinizializza il sistema e si riesegue il programma è rigenerata la stessa sequenza di numeri casuali.
Esempio	Vediamo nella stampa sottostante come la prima routine, ogni volta che è eseguita, produce la stessa sequenza di numeri casuali compresi tra zero ed uno. Se prima della routine suddetta, si pone l'istruzione RANDOMIZE (si noti che l'istruzione può essere introdotta abbreviando la parola RANDOMIZE in RAN), ogni volta che la nuova routine è eseguita si ottiene una sequenza diversa di numeri casuali compresi tra zero e uno.

```
LIST
FILE    *RAN

0010 FOR I=1 TO 20 STEP 1
0030 PRINT RND,
0040 NEXT I
0050 END

END OF LISTING


RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
5 RAN
LIST
FILE    *RAN

0005 RAND
0010 FOR I=1 TO 20 STEP 1
0030 PRINT RND,
0040 NEXT I
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 .46928183        .93340150        .73981330        .93170842        .66550775
 4.9468396E-02    .41467724        .16474361        4.9257438E-02    .45449761
 .29001342        .73709782        1.0500667E-02    .92415374        .38637937
 .26806888        .15637405        7.7169896E-03    .18172449        3.9252707E-03
RUN
 .30027045        .84385646        .14483391        .59239684        .91177673
 .51377738        .21853162        .35957285        .80906657        .55912861
 .96844639        .33748557        .87592708        .94394928        .14757729
 .35964649        .44423446        .63348580        .61190501        .13459401
```

Istruzione READ

Funzione

Assegna alle variabili di programma specificate i valori contenuti nel file dati, interno al programma, generato con le istruzioni DATA.

Formato

READ | num-var / string-var | [, | num-var / string-var |] ...

dove:

num-var
è una variabile numerica, semplice o con indice, alla quale viene assegnato un valore numerico prelevato dal file dati interno generato con le istruzioni DATA del programma

string-var
è una variabile stringa, semplice o con indice, alla quale viene assegnato un valore numerico prelevato dal file dati interno suddetto

Azione

I valori nel file dati interno sono assegnati nell'ordine (da sinistra a destra) alle variabili indicate nella istruzione READ, iniziando dalla posizione indicata in quel momento dal pointer del file dati interno (vedi istruzione DATA).

Note

1. Il pointer del file dati interno può essere riposizionato all'inizio del file stesso mediante l'istruzione RESTORE (vedi istruzione RESTORE).

2. I valori degli indici delle variabili con indice sono assegnati nel momento in cui compaiono nelle istruzioni READ; così una variabile presente in una istruzione READ può essere utilizzata come indice in una successiva variabile con indice presente nella stessa istruzione READ.

3. I valori assegnati dal file dati interno devono essere coerenti con il tipo di variabili presenti nella istruzione READ.

4. Se c'è una istruzione READ in un programma deve esserci almeno una istruzione DATA.

5. Se vi è una variabile in una istruzione READ a cui non può essere assegnato un valore perchè il file dati interno è esaurito, allora l'esecuzione del programma è sospesa; premendo BREAK il sistema commuta nello stato comandi.

6. Per ulteriori informazioni si veda l'istruzione DATA.

Esempio

1. Nella routine seguente si vede che è possibile, con una sola istruzione READ, assegnare il valore all'indice di una variabile con indice e quindi assegnare un valore alla variabile stessa.

```
LIST
FILE    +READ

0005 LET I=9
0006 LET A(9)=45
0010 DATA 1,2
0020 READ I,A(I)
0030 PRINT "I=";I,"A(1)=";A(1),"A(9)=";A(9)
0040 END

END OF LISTING


RUN
I= 1            A(1)= 2         A(9)= 45
```

Istruzione READ:

Funzione

Assegna alle variabili di programma specificate i valori contenuti in un file esterno.

Formato

READ: file-designator, ⌈**num-var** / **string-var**⌉ [, ⌈**num-var** / **string-var**⌉] ... [**EOF line-num**]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica da quale file dati esterno devono essere prelevati i dati da assegnare alle variabili specificate nella istruzione

num-var
è una variabile numerica, semplice o con indice; alla quale è assegnato il relativo dato numerico prelevato dal file esterno, specificato con file-designator

string-var
è una variabile stringa, semplice o con indice, alla quale è assegnato il dato prelevato dal file dati esterno specificato con file-designator

line-num
è il numero di linea della istruzione a cui viene ceduto il controllo della esecuzione del programma se è raggiunta la fine logica (file sequenziale) o la fine fisica (file ad accesso diretto) del file dati specificato con file-designator

Azione

L'espressione numerica è eseguita ed il valore ottenuto, arrotondato all'intero più prossimo nd, costituisce il numero designatore del file esterno da cui sono prelevati i valori dei dati da assegnare alle variabili di programma indicate nella istruzione.

I valori dei dati sono prelevati dal file con numero

designatore nd iniziando dal dato su cui è posizionato il pointer del file ed assegnati nell'ordine (da sinistra a destra) alle variabili presenti nella istruzione.

Il valore del pointer del file esterno è aggiornato in modo da indirizzare la posizione successiva all'ultimo dato letto.

Se il pointer di un file dati sequenziale indirizza una posizione successiva all'ultimo dato registrato nel file, l'esecuzione dell'istruzione READ: dà una segnalazione d'errore e l'esecuzione del programma è sospesa, se, però, è presente l'opzione EOF, il controllo della esecuzione del programma passa all'istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Se il pointer di un file dati ad accesso diretto indirizza la posizione successiva all'ultima parola allocata per il file (vedi comando CREATE), l'esecuzione dell'istruzione READ: dà una segnalazione di errore e l'esecuzione del programma è sospesa; se è presente l'opzione EOF il controllo della esecuzione del progamma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Note

1. Con l'istruzione READ: si può leggere anche il contenuto di un file testo. Il file testo è considerato un file dati sequenziale contenente stringhe di caratteri ognuna composta da una linea del file testo con incluso il numero di linea.

2. Per poter eseguire una istruzione READ: il file dati esterno deve essere stato prima aperto all'accesso da parte del programma mediante una istruzione FILES o FILE:.

3. Dopo l'esecuzione di una istruzione WRITE:, se il file dati è di tipo sequenziale, si deve eseguire una istruzione RESTORE: (vedi istruzione RESTORE:) prima di una istruzione READ:.

4. Se il file è stato dichiarato, con il comando CREATE (vedi capitolo 3) ad accesso diretto si può leggere qualunque dato utilizzando prima dell'i-

struzione READ: l'istruzione SETW: (vedi istruzione SETW:) per assegnare al pointer l'indirizzo del dato che si vuol leggere.

5. I dati letti dal file devono essere coerenti con le variabili specificate nell'istruzione READ:.

6. Il risultato della espressione numerica (specificata con file-designator), arrotondato all'intero più prossimo, deve essere maggiore di Ø e minore o uguale al numero di file che possono essere utilizzati contemporaneamente dal programma (vedi istruzione FILES).

7. In una istruzione READ: si può specificare una variabile utilizzata come indice in una variabile con indice specificata successivamente nella stessa istruzione.

8. Se ad una variabile stringa viene assegnata, da un file dati esterno, una stringa con più caratteri di quelli definiti per la sua lunghezza di allocazione, la stringa è assegnata alla variabile suddetta troncata dei caratteri eccedenti sulla destra. Il sistema è nello stato di debugging e visualizza un messaggio di errore di tipo recuperabile. Premendo il tasto di console CONTINUE l'esecuzione del programma continua; premendo BREAK l'esecuzione termina.

9. Se viene assegnato ad una variabile numerica, rappresenta in singola precisione, un valore numerico, rappresentato su file dati in doppia precisione, con esponente al di fuori del range della singola precisione, il sistema le assegna il valore zero, se l'esponente è nella zona di UNDERFLOW per la singola precisione, od il valore +9.99999E63 (oppure -9.99999E63), se l'esponente è nella zona di OVERFLOW per la singola precisione. Il sistema visualizza un messaggio di errore recuperabile ed è nello stato di debugging. Premendo il tasto di console CONTINUE l'esecuzione continua; premendo BREAK l'esecuzione del programma termina.

Esempi

1. Nella routine sottostante si mostra che se il numero designatore è espresso con un valore che arrotondato all'intero più prossimo è minore di 1 o maggiore del numero di file accessibile contemporaneamente da programma (ricavato dall'istruzione FILES), è segnalato un errore di tipo non recuperabile (vedi sotto le prime due esecuzioni dopo la digitazione dei primi due comandi RUN) e l'esecuzione del programma è sospesa. Premendo BREAK il sistema commuta nello stato comandi. Si notino gli altri casi: ad esempio quando il numero introdotto è 1.8, il file letto è quello con numero designatore 2, etc.

```
LIST
FILE


0010 FILES APPEND;ISOT;DIRET1;C1
0020 FOR I=1 TO 6 STEP 1
0030 DISP "Quale file vuoi leggere";
0040 INPUT A
0050 READ :A,B
0052 IF A-INT(A)<=0.5 THEN 57
0053 LET A=INT(A)+1
0054 GOTO 60
0057 LET A=INT(A)
0060 PRINT "Ho prelevato";B;"nel file indicato nella posizione";A;"in FILES."
0070 NEXT I
0080 END

END OF LISTING


RUN
Quale file vuoi leggere?
0
ERROR 78  IN LINE 50
RUN
Quale file vuoi leggere?
5
ERROR 77  IN LINE 50
RUN
Quale file vuoi leggere?
1.3
Ho prelevato 1 nel file indicato nella posizione 1 in FILES.
Quale file vuoi leggere?
1.8
Ho prelevato 0 nel file indicato nella posizione 2 in FILES.
Quale file vuoi leggere?
3.2
Ho prelevato 1 nel file indicato nella posizione 3 in FILES.
Quale file vuoi leggere?
3.75
Ho prelevato 1 nel file indicato nella posizione 4 in FILES.
Quale file vuoi leggere?
4.4
Ho prelevato 8 nel file indicato nella posizione 4 in FILES.
Quale file vuoi leggere?
2
Ho prelevato 0 nel file indicato nella posizione 2 in FILES.
```

2. Nella routine sottostante si può vedere come nella istruzione READ: si può assegnare un valore da un file dati esterno ad una variabile che è utilizzata nella stessa istruzione come indice di una variabile con indice.

```
LIST
FILE


0010 FILES DIRET1
0020 LET I=A(10)=10
0030 READ :1,I,A(I)
0040 PRINT "I=";I,"A(1)=";A(1),"A(10)=";A(10)
0050 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
I= 1            A(1)= 2          A(10)= 10
```

3. Vediamo un esempio d'impiego dell'opzione EOF in un'istruzione READ: che legge i dati di un file sequenziale.

```
LIST
FILE


0010 DCL 80A$
0020 FILES Q
0030 FOR I=1 TO 5 STEP 1
0040 READ :1,A$ EOF 70
0050 PRINT A$
0060 NEXT I
0070 PRINT "Nel file Q non vi sono piu` dati!"
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
1234567890123456789
Nel file Q non vi sono piu` dati!
```

4. Con il programma sottostante sono registrati nel file ad accesso diretto PROVA, creato in una libreria allocando per esso 128 byte, 30 dati numerici in singola precisione, per cui rimangono le due ultime parole ancora non registrate (ad esse il sistema aveva assegnato, in fase di inizializzazione, il valore zero). Quando il file PROVA viene letto dall'inizio ad un certo punto il sistema si pone nello stato di debugging e visualizza il messaggio recuperabile ERROR 1; premendo il tasto CONTINUE la situazione si ripete ma ad una successiva pressione di CONTINUE l'esecuzione continua fino al termine del file. Solo quando è raggiunta la fine fisica del file PROVA entra in azione l'opzione EOF che rimanda la esecuzione alla istruzione 130.

```
0010 FILES PROVA
0020 DCL SA
0030 LET A=0
0040 FOR I=1 TO 30 STEP 1
0050 LET A=I
0060 WRITE :1,A
0070 NEXT I
0080 SETW :1 TO 1
0085 FOR I=1 TO 40 STEP 1
0090 READ :1,B EOF 130
0100 PRINT B,
0120 NEXT I
0130 PRINT "Sono giunto all'ultima parola del file PROVA!"
0140 END

END OF LISTING


RUN
 1              2              3              4              5
 6              7              8              9              10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
ERROR 1   IN LINE 90
ERROR 1   IN LINE 90
 0              0             Sono giunto all'ultima parola del file PROVA!
```

5. Vediamo qui una routine che legge un file testo registrato in una libreria col nome MANUAL. Sono lette e stampate le prime 20 linee del file testo.

```
FILE

0010 DCL 80(A$)
0020 FILES MANUAL
0030 FOR I=1 TO 20 STEP 1
0040 READ :1,A$
0050 PRINT A$
0060 NEXT I
0070 END

END OF LISTING



0010
0020
0030
0040                          4. BASIC: DATI, VARIABILI, ESPRESSIONI E FILE DATI
0050
0060  Questo capitolo offre al lettore che ha gia'una certa familiarita' con i
0070 concetti del linguaggio BASIC la possibilita' di effettuare una rapida e
0080 completa consultazione di tutte le possibilita' offerte dal linguaggio
0090 BASIC realizzato per il sistema P6066.
0100
0110
0120   4.1 CARATTERI DEL LINGUAGGIO BASIC
0130
0140  I caratteri che hanno un ruolo sintattico nel linguaggio BASIC sono
0150 classificabili in tre categorie:
0160
0170                                      CARATTERI ALFABETICI
0180                                      CARATTERI NUMERICI
0190                                      CARATTERI SPECIALI
0200
```

Istruzione REMARK

Funzione

Permette di inserire in un programma dei commenti che rendono facile la lettura del relativo listing.

Formato

REM [ARK] comment

dove:
comment
è una successione di caratteri del set ISO (vedi appendice E) che rappresenta un commento per il programmatore.

Azione

Non è una istruzione eseguibile. Il commento appare nel listing del programma, ma non produce alcun effetto durante l'esecuzione del programma.

Esempio

Come si vede dall'esempio l'istruzione REMARK può essere introdotta abbreviando la parola chiave in REM.

```
NEW
10 REM Questa istruzione ti dice che cosa fanno le istruzioni che seguono.
20 END
LIST
FILE


0010 REM Questa istruzione ti dice che cosa fanno le istruzioni che seguono.
0020 END

END OF LISTING
```

Istruzione RESTORE

Funzione

Posiziona il pointer del file dati interno all'inizio del file stesso.

Formato

RESTORE

Azione

Il pointer del file dati interno (vedi istruzione DATA) viene rimosso dalla posizione attuale e posizionato all'inizio del file stesso.

Note

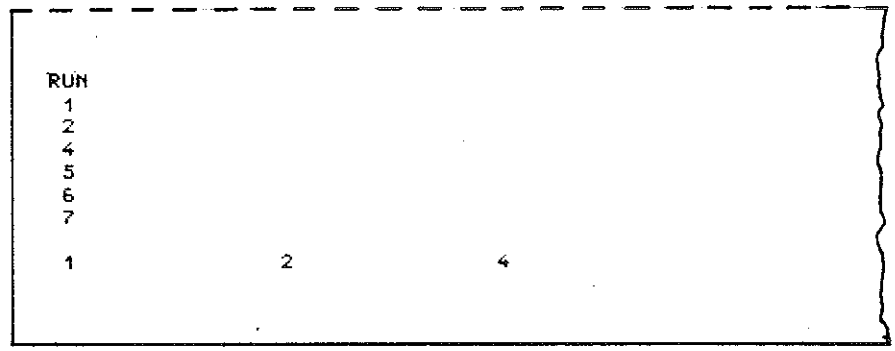
1. Una istruzione RESTORE dopo una precedente istruzione RESTORE (senza istruzioni READ interposte tra le due istruzioni suddette) non provoca alcun effetto.

2. Una istruzione RESTORE in un programma senza istruzioni DATA non provoca alcun effetto.

Esempio

La routine sottostante mostra un impiego dell'istruzione RESTORE.

```
LIST
FILE     RESTOR

0010 DATA 1,2,4,5,6,7,8,9
0020 FOR I=1 TO 7 STEP 1
0030 READ I
0040 PRINT I
0050 NEXT I
0060 RESTORE
0070 READ A,B,C
0080 PRINT
0090 PRINT A,B,C
0100 END

END OF LISTING
```


RUN
1
2
4
5
6
7
1
2
4

Istruzione RESTORE:

Funzione

Posiziona il pointer di un file esterno all'inizio del file e, se il file è di tipo sequenziale, ne permette la lettura.

Formato

RESTORE: file-designator

dove:
file-designator
è una espressione numerica il cui valore arrotondato all'intero più prossimo indica un designatore di file (vedi istruzioni FILES e FILE:).

Azione

L'espressione numerica viene eseguita ed il valore calcolato è arrotondato all'intero più prossimo nd.

Il puntatore del file esterno il cui numero designatore è nd è posizionato all'inizio del file stesso.

Note

1. Se il file è sequenziale, dopo l'esecuzione dell'istruzione RESTORE:, può essere letto con l'istruzione READ:

2. Se il file con numero designatore nd è di tipo ad accesso diretto, l'esecuzione dell'istruzione RESTORE: posiziona il pointer all'inizio del file.

3. Il valore della istruzione numerica arrotondato all'intero più prossimo deve essere maggiore di zero e minore od uguale al numero di file contemporaneamente aperti all'accesso da parte del programma, stabilito con l'istruzione FILES.

4. L'istruzione RESTORE: può essere usata con riferimento ad un file testo e pone il pointer all'inizio del file medesimo.

Esempi

1. Vediamo un impiego dell'istruzione RESTORE: con un file di tipo sequenziale.

```
]LIST
 FILE


 0010 FILES FILES1
 0020 FOR I=1 TO 16 STEP 1
 0030 READ :1,A
 0040 PRINT A
 0050 NEXT I
 0060 RESTORE :1
 0070 FOR I=1 TO 6 STEP 1
 0080 READ :1,B
 0090 PRINT "B=";B
 0100 NEXT I
 0110 RESTORE :1
 0120 READ :1,C
 0130 PRINT "C=";C
 0140 END

 END OF LISTING


 RUN
  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
  11
  12
  13
  14
  15
  16
B= 1
B= 2
B= 3
B= 4
B= 5
B= 6
C= 1
```

2. Con questa routine si rileggono i dati di un file ad accesso diretto dall'inizio. Si noti come l'istruzione SETW: 1 TO 1 svolge l'azione svolta nella routine precedente per il file sequenziale dalla istruzione RESTORE:.

```
LIST
FILE    *REST03


0010 FILES FILED2
0020 FOR I=1 TO 10 STEP 1
0030 READ :1,A
0040 PRINT "A=";A
0050 NEXT I
0060 SETW :1 TO 1
0070 READ :1,B,C,D
0080 PRINT "B=";B,"C=";C,"D=";D
0090 END

END OF LISTING


RUN
A= 1
A= 2
A= 3
A= 4
A= 5
A= 6
A= 7
A= 8
A= 9
A= 10
B= 1            C= 2            D= 3
```

Istruzione RETURN

Funzione	Trasferisce il controllo della esecuzione di un programma all'istruzione successiva ad una istruzione GOSUB oppure ON...GOSUB.
Formato	**RETURN**
Azione	Passa il controllo della esecuzione del programma alla prima istruzione esecutiva successiva alla istruzione GO SUB, o ON...GOSUB, che ha ceduto il controllo della esecuzione del programma al sottoprogramma di cui l'istruzione RETURN fa parte.
Note	1. In un sottoprogramma vi possono essere più istruzioni RETURN. 2. Per una facile lettura del programma, è bene utilizzare una sola istruzione RETURN in un sottoprogramma alla quale ci si può riferire da diversi punti del sottoprogramma stesso mediante delle istruzioni GOTO o IF...THEN.
Esempi	Vedi istruzioni GOSUB ed ON...GOSUB.

RKB

Istruzione RKB

Funzione

Assegna ad una variabile stringa i caratteri introdotti da tastiera che sono scelti tra i caratteri del set ISO (vedi appendice E).

Formato

RKB string-var

dove:
string-var
è una variabile stringa, semplice o con indice, a cui vengono assegnati i caratteri introdotti da tastiera.

Azione

L'esecuzione del programma è interrotta e sul display è visualizzato il carattere ?.

La stringa di caratteri digitata è assegnata alla variabile indicata nella istruzione.

Note

1. L'istruzione RKB permette di assegnare ad una variabile di programma il carattere virgolette (").

2. Se i caratteri introdotti superano la lunghezza di allocazione della variabile stringa specificata, il sistema assegna alla variabile la stringa troncata dei caratteri eccedenti la lunghezza di allocazione e commuta nello stato di debugging visualizzando un un messaggio sul display.

Esempi

1. Vediamo in questo esempio come, utilizzando RKB, si possono assegnare ad una variabile stringa i caratteri che si vogliono. In particolare si noti che si possono introdurre frasi comprese tra virgolette; infatti anche il carattere virgolette è considerato

parte del valore da assegnare alla variabile specificata. Si noti come la variabile stringa può essere una variabile con indice.

```
0010 DCL 80A$,80A$()
0020 RKB A$
0030 RKB A$(1)
0040 PRINT
0050 PRINT "A$=";A$
0060 PRINT
0070 PRINT "A$(1)=";A$(1)
0080 END
RUN
**** FORMALLY CORRECT PROGRAM ****
?
"Come si vede si possono assegnare i caratteri che si vogliono!"
?
▥o↖↓↓↓↓↖←0↖o↖←←⊥┘▥▥▥¥≡↖o/≷≷√?┘⊥0←▥▥▥o≷0

A$="Come si vede si possono assegnare i caratteri che si vogliono!"

A$(1)=▥o↖↓↓↓↓↖←0↖o↖←←⊥┘▥▥▥¥≡↖o/≷≷√?┘⊥0←▥▥▥o≷0
```

2. In questo esempio si può vedere cosa accade se il numero dei caratteri introdotti da tastiera supera la lunghezza di allocazione della variabile specificata nell'istruzione RKB. Il sistema commuta nello stato di debugging e visualizza il messaggio di errore sottoriportato. Premendo **CONTINUE** l'esecuzione del programma riprende ed alla variabile sono assegnati i primi 16 caratteri digitati (perchè la lunghezza di allocazione di A$ è di 16 caratteri).

```
NEW
10 DISP"INTRODUCI I CARATTERI CHE VUOI    ";
20 RKB A$
30 PRINT "A$=";A$
40 END
RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI I CARATTERI CHE VUOI    ?
1234567890123456789
ERROR 8   IN LINE 20
A$=1234567890123456
```

Istruzione SCRATCH:

Funzione

Posiziona il pointer di un file dati esterno, di tipo sequenziale, all'inizio del file e permette di registrare in esso dei dati con una successiva istruzione WRITE:.

Formato

SCRATCH: file-designator

dove:
file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file (vedi istruzioni FILES e FILE:).

Azione

L'espressione numerica è eseguita ed il valore ottenuto è arrotondato all'intero più prossimo nd.

Il pointer del file esterno il cui numero designatore è nd è posizionato all'inizio del file ed il contenuto del file è cancellato. Il file è posto in modalità di scrittura, ossia si possono eseguire delle istruzioni WRITE: per registrare in esso dei dati.

Note

1. Il valore dell'espressione numerica, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore od uguale al numero di file dichiarati accessibili contemporaneamente da programma mediante l'istruzione FILES.

2. La successiva istruzione di I/O deve essere una istruzione WRITE: e non READ:.

3. L'istruzione SCRATCH: può essere usata solo con file sequenziali.

Esempio

Ecco un esempio di impiego dell'istruzione SCRATCH:. Si noti come l'istruzione SCRATCH: cancella il precedente contenuto del file FILES1; infatti dopo aver letto il quinto dato l'istruzione READ: esegue l'opzione EOF che indica la mancanza di ulteriori dati nel file.

```
OLD *SCRATC
LIST
FILE    *SCRATC


0010 FILES FILES1
0020 FOR I=1 TO 10 STEP 1
0030 READ :1,A
0040 PRINT "A=";A,
0050 NEXT I
0060 SCRATCH :1
0070 WRITE :1,21,22,23,24,25
0080 RESTORE :1
0090 FOR I=1 TO 10 STEP 1
0100 READ :1,B EOF 130
0110 PRINT "B=";B,
0120 NEXT I
0130 PRINT "Non si possono leggere altri dati nel file FILES1 perche` ... "
0140 PRINT "... l'istruzione SCRATCH li ha cancellati."
0150 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1            A= 2            A= 3            A= 4            A= 5
A= 6            A= 7            A= 8            A= 9            A= 10
B= 21           B= 22           B= 23           B= 24           B= 25
Non si possono leggere altri dati nel file FILES1 perche` ...
... l'istruzione SCRATCH li ha cancellati.
```

Istruzione SETW:

Funzione

Posiziona il pointer all'inizio della parola specificata, di un file dati esterno ad accesso diretto.

Formato

SETW: file-designator TO word-num

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file (vedi istruzioni FILES e FILE:)

word-num
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di parola nell'ambito del file dati specificato con file-designator.

Azione

Le espressioni numeriche sono eseguite e i valori ottenuti sono arrotondati agli interi più prossimi: nd (designatore di file) e np (designatore di parola).

Il puntatore del file di numero designatore nd è posizionato all'inizio della np-esima parola del file stesso.

Note

1. Il file di numero designatore nd deve essere ad accesso diretto.

2. Dopo l'istruzione SETW: si possono registrare dati sul file (istruzione WRITE:) o leggere dati dal file (istruzione READ:).

3. Il risultato della espressione numerica, arrotondato all'intero più prossimo nd, deve essere maggiore di zero e minore od uguale al numero di file che

sono contemporaneamente accessibili dal programma, come specificato dall'istruzione FILES.

4. Il risultato della espressione numerica, arrotondato all'intero prossimo np, deve essere maggiore di zero e minore od uguale al numero di parole allocate per il file con il comando CREATE (vedi comando CREATE).

Esempio

Col seguente programma si registrano nei file dati ad accesso diretto FILED1, FILED2, FILED3 e FILED4 i numeri 1,2,3 e 4 per tutta l'estensione dei file. Si noti che i numeri sono assegnati a variabili dichiarate in singola precisione per cui sono registrati ognuno su di una parola dei file suddetti. Quindi si moltiplicano i dati registrati rispettivamente nelle parole 5, 16, e 8 dei file FILED2, FILED3 e FILED4. Il risultato è registrato nella parola 3 del file FILED1. Per poter analizzare il risultato prodotto dal programma nei quattro file suddetti si vedano le stampe riportate che si riferiscono al contenuto dei file. La prima stampa è relativa al contenuto del file FILED1, la seconda al file FILED2 etc..

```
CIS
FILE    *SETW2

0004 DCL SINGLE
0005 FILES FILED1;FILED2;FILED3;FILED4
0010 FOR J=1 TO 4 STEP 1
0020 LET A=J
0030 GOSUB 200
0040 NEXT J
0050 SETW :2 TO 5
0060 WRITE :2,10
0070 SETW :3 TO 16
0080 WRITE :3,15
0100 SETW :4 TO 8
0110 WRITE :4,8
0120 REM Ora moltiplico i dati conenuti nelle parole 5, 16 e 8 dei file  ...
0130 REM ... FILED2,FILED3 e FILED4 e metto il risultato nella parola 3 ...
0140 REM ... del file FILED1.
0145 SETW :2 TO 5
0146 SETW :3 TO 16
0147 SETW :4 TO 8
0150 READ :2,X
0151 READ :3,Y
0152 READ :4,Z
0160 SETW :1 TO 3
0170 WRITE :1,X*Y*Z
0174 SETW :1 TO 3
0175 READ :1,C
0180 PRINT "Questo e`il numero posto in FILED1:";C
0190 GOTO 250
0200 SETW :A TO 1
```

```
0210 FOR I=1 TO 32 STEP 1
0220 WRITE :A,A
0230 NEXT I
0240 RETURN
0250 END

END OF LISTING



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


Questo e'il numero posto in FILED1: 1200


              FILED1
1              1
1              1           1200          1              1
1              1              1          1              1
1              1              1          1              1
1              1              1          1              1
1              1              1          1              1
1              1              1          1              1

              FILED2
2              2              2          2
2              2              2          2             10
2              2              2          2              2
2              2              2          2              2
2              2              2          2              2
2              2              2          2              2
2              2              2          2              2

              FILED3
3              3
3              3              3          3              3
3              3              3          3              3
15             3              3          3              3
3              3              3          3              3
3              3              3          3              3
3              3              3          3              3

              FILED4
4              4
4              4              4          4              4
4              4              3          4              4
4              4              4          4              4
4              4              4          4              4
4              4              4          4              4
4              4              4          4              4
```

Istruzione STOP

Funzione

Interrompe l'esecuzione di un programma e commuta il sistema nello stato di debugging.

Formato

STOP

Azione

L'esecuzione del programma è sospesa.

Sul display è visualizzato il messaggio "STOP numero di linea" dove "numero di linea" è quello della istruzione stessa.

Le variabili di programma conservano i loro contenuti ed il sistema si trova nello stato di debugging, per cui si possono effettuare tutte le operazioni descritte nel capitolo 7.

Note

1. L'istruzione STOP è utile nella fase di debugging (vedi capitolo 7) del programma.

2. L'istruzione STOP non deve essere compresa in una funzione multilinea.

3. L'esecuzione del programma riprende non appena si preme il tasto CONTINUE od il tasto STEP. Per maggiori dettagli sull'impiego dei due tasti suddetti si veda il capitolo 7.

4. L'istruzione STOP è anche utile per fermare l'esecuzione del programma e permette all'operatore di digitare uno dei tasti funzione che può rinviare l'esecuzione ad una routine scelta dall'operatore stesso (si veda l'esempio allegato all'istruzione FKEY#).

5. Se l'istruzione STOP è preceduta da una istruzione DISP con alla fine il punto e virgola, o la virgola, allora il messaggio relativo all'istruzione DISP permane sul display ed il messaggio "STOP numero di linea" non viene visualizzato.

Esempi

Si vedano gli esempi indicati nei capitoli 7, 8 e nella istruzione FKEY #.

TRACE OFF

Istruzione TRACE OFF

Funzione	Termina la stampa dei numeri di linea delle istruzioni di programma eseguite.
Formato	**TRACE OFF**
Azione	La stampa dei numeri di linea delle istruzioni esecutive del programma, predisposta con la precedente istruzione TRACE ON, è interrotta. La luce di console TRACE si spegne.
Note	1. Se l'istruzione TRACE OFF non esiste nel programma, o comunque non è incontrata durante l'esecuzione del programma, la stampa dei numeri di linea continua fino alla esecuzione della istruzione END. 2. L'istruzione TRACE OFF annulla anche l'effetto prodotto premendo il tasto di console TRACE ON. Se dopo si preme di nuovo il tasto di console TRACE il sistema riprende a stampare i numeri di linea delle istruzioni esecutive.
Esempi	Si veda l'esempio 2 dell'istruzione TRACE ON.

TRACE ON

Istruzione TRACE ON

Funzione

Esegue la stampa del numero di linea di ogni successiva istruzione di programma eseguita.

Formato

TRACE ON

Azione

I numeri di linea delle istruzioni esecutive successive vengono stampati nell'ordine in cui sono eseguite. La luce del tasto di console TRACE si accende.

Note

1. Non sono stampati i numeri di linea delle istruzioni non esecutive.

2. I numeri di linea delle istruzioni delle funzioni definite dall'utente (monolinea e multilinea) non sono stampati.

3. Il numero di linea della istruzione TRACE ON non viene stampato, mentre viene stampato il numero di linea della istruzione TRACE OFF.

4. I numeri di linea stampati sono interposti tra altre linee di stampa comandate da programma.

5. L'effetto prodotto da TRACE ON viene annullato quando è eseguita l'istruzione TRACE OFF; se non vi è una istruzione TRACE OFF allora la stampa dei numeri di linea continua fino all'istruzione END (anche di essa è stampato il numero di linea).

6. L'istruzione TRACE ON è utile durante il debugging di un programma (si veda il capitolo 7).

7. Se si preme il tasto di console TRACE si ottiene lo stesso effetto prodotto con l'esecuzione della istruzione TRACE ON. In questo caso anche il numero

di linea della istruzione TRACE ON è stampato. Quindi, se prima di introdurre il comando RUN si preme il tasto TRACE i numeri di linea delle istruzioni esecutive sono stampati dall'inizio anche se l'istruzione TRACE ON viene eseguita più avanti nel programma.

Esempi

1. Nel programma seguente l'istruzione TRACE ON è la prima e non vi è una istruzione TRACE OFF, per cui tutti i numeri di linea delle istruzioni esecutive vengono stampati. Si noti come i numeri di linea sono preceduti dal segno # e non sono stampati sulla stessa linea di altre stampe prodotte dal programma.

```
LIST
FILE    *TRAC1


0005 TRACE ON
0010 REM Ecco un esempio di impiego dell'istruzione TRACEON
0020 DCL SA
0030 DATA 1,2,3,4,5
0040 READ A,B,C,D,E
0050 FILES FILED1;FILED2;FILED3
0060 SETW :1 TO 10
0065 READ :1,A
0070 PRINT "Ho letto questo dato nel file FILED1:";A
0080 SETW :2 TO 4
0090 READ :2,A
0100 PRINT "Ho letto questo dato nel file FILED2:";A
0110 SETW :3 TO 8
0120 READ :3,A
0130 PRINT "Ho letto questo dato nel file FILED3"
0150 LET X=B*C*D*E
0160 PRINT "X=";X
0170 END

END OF LISTING


RUN
#40
#60
#65
#70
Ho letto questo dato nel file FILED1: 1
#80
#90
#100
Ho letto questo dato nel file FILED2: 2
#110
#120
#130
Ho letto questo dato nel file FILED3
#150
#160
X= 120
#170
```

2. Il programma sottostante è identico al precedente con l'aggiunta dell'istruzione TRACE OFF che, come si vede, annulla l'effetto prodotto dalla istruzione TRACE ON. Si noti come sia stampato anche il numero di linea dell'istruzione TRACE OFF (95).

```
LIST
FILE    *TRAC1


0005 TRACE ON
0010 REM Ecco un esempio di impiego dell'istruzione TRACEON
0020 DCL SA
0030 DATA 1,2,3,4,5
0040 READ A,B,C,D,E
0050 FILES FILED1;FILED2;FILED3
0060 SETW :1 TO 10
0065 READ :1,A
0070 PRINT "Ho letto questo dato nel file FILED1:";A
0080 SETW :2 TO 4
0090 READ :2,A
0095 TRACE OFF
0100 PRINT "Ho letto questo dato nel file FILED2:";A
0110 SETW :3 TO 8
0120 READ :3,A
0130 PRINT "Ho letto questo dato nel file FILED3"
0150 LET X=B*C*D*E
0160 PRINT "X=";X
0170 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
#40
#60
#65
#70
Ho letto questo dato nel file FILED1: 1
#80
#90
#95
Ho letto questo dato nel file FILED2: 2
Ho letto questo dato nel file FILED3
X= 120
```

Istruzione WHERE:

Funzione

Permette di determinare su quale parola di un file dati esterno è posizionato il relativo pointer ed il tipo di dato da esso indirizzato.

Formato

WHERE: file-designator, num- var$_1$ [, num-var$_2$ [, num-var$_3$]]

dove:

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

num-var$_1$
è una variabile numerica alla quale viene assegnato il valore corrente (in numero di parole) del pointer del file esterno specificato con file-designator

num-var$_2$
è una variabile numerica alla quale è assegnato un valore numerico che specifica il dato su cui è posizionato il pointer

num-var$_3$
è una variabile numerica che, se il tipo di dato su cui è posizionato il pointer è una stringa di caratteri, ne specifica la lunghezza.

Azione

Il valore attuale del pointer del file dati esterno specificato con il valore arrotondato all'intero più prossimo dell'espressione numerica file-designator è assegnato alla variabile numerica num-var$_1$.

Alla variabile num-var$_2$ è assegnato uno dei seguenti valori numerici:

Valore	Interpretazione
0	Se nella posizione corrente del pointer non è riconosciuto alcun identificatore
1	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico rappresentato in singola precisione (pointer posizionato all'inizio di un dato numerico in singola precisione)
2	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico rappresentato in doppia precisione (pointer posizionato all'inizio di un dato numerico in doppia precisione)
3	se nella posizione corrente del pointer è riconosciuto un identificatore di un dato stringa (pointer posizionato all'inizio di una stringa di caratteri)
4	se il pointer è posizionato alla fine del file dati esterno
5	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico non inizializzato rappresentato in singola precisione
6	se nella posizione corrente del pointer è riconosciuto un identificatore di dato numerico non inizializzato rappresentato in doppia precisione
7	se nella posizione corrente del pointer è riconosciuto un identificatore di dato stringa non inizializzato

Alla variabile num-var$_3$ è assegnato il numero di caratteri che compongono la stringa su cui è posizionato il pointer, quando il valore di num-var$_2$ è 3; se il valore di num-var$_2$ è diverso da 3, la variabile num-var$_3$ è posta a zero.

Note

1. Si osservi che anche se il pointer è posizionato all'interno di una stringa di caratteri, particolari codici vengono riconosciuti come identificatore di inizio dato.

2. Gli operandi num-var$_2$ e num-var$_3$ si possono specificare se il file dati esterno è stato aperto in lettura.

Esempio

Nel programma seguente l'istruzione WHERE: permette di leggere e stampare solamente le stringhe di dati che sono presenti nel file DATI. La funzione multilinea FNA fa in modo che l'esecuzione del programma non sia interrotta quando è raggiunta la fine del file.

```
LIST
FILE      WHERE


0010 LET C1=0
0020 INTERRUPT ENABLE (I,"A")
0030 FILES DATI
0040 WHERE : 1,P,T
0050 IF C1=1 THEN 240
0060 IF T=3 THEN 100
0070 IF T=1 THEN 150
0080 IF T=2 THEN 140
0090 SETW :1 TO P+1
0100 READ :1,A$
0110 PRINT "LA PAROLA ";P;"CONTIENE: ";A$
0130 GOTO 40
0140 SETW :1 TO P+2
0150 GOTO 40
0160 SETW :1 TO P+1
0180 GOTO 40
0190 DEF FNA(C,L)
0200 LET C1=1
0210 LET P=0
0220 LET FN*=0
0230 FNEND
0240 END

END OF LISTING


LA PAROLA  1 CONTIENE: OLIVETTI
LA PAROLA  4 CONTIENE: P6066
LA PAROLA  8 CONTIENE: NEW YORK
LA PAROLA  11 CONTIENE: ROMA
LA PAROLA  60 CONTIENE: MILANO
LA PAROLA  100 CONTIENE: GINEVRA
```

Istruzione WRITE:

Funzione

Registra in un file dati esterno i valori delle espressioni specificate.

Formato

WRITE: file-designator, { **num-exp** / **string-exp** } [, { **num-exp** / **string-exp** }] ... [**EOF line-num**]

dove:

file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo nd, indica il designatore di un file (vedi istruzioni FILES e FILE:)

num-exp
è una espressione numerica il cui valore è registrato sul file dati esterno specificato con file-designator

string-exp
è una espressione stringa il cui valore è registrato sul file dati esterno specificato con file-designator

line-num
è il numero di linea della istruzione di programma alla quale viene trasferito il controllo della esecuzione quando il pointer del file esterno si trova alla fine del file

Azione

I valori corrispondenti alle espressioni specificate nell'istruzione dopo file-designator sono registrati nel file dati esterno identificato dal numero designatore di file nd. I valori sono registrati dalla posizione indirizzata dal pointer e nell'ordine con cui sono specificati nell'istruzione.

Se non c'è spazio sufficiente per registrare un dato sul file (sequenziale o ad accesso diretto) l'esecuzione dell'istruzione WRITE: dà una segnalazione di errore e l'esecuzione del programma è sospesa, ma i dati

specificati nell'istruzione prima del dato suddetto sono registrati nel file esterno. Se è presente l'opzione EOF il controllo della esecuzione del programma passa alla istruzione il cui numero di linea è specificato nella opzione stessa e non vi è alcuna segnalazione di errore.

Note

1. Se il file è sequenziale la prima istruzione WRITE: eseguita deve essere preceduta da una istruzione SCRATCH: o APPEND:.

2. Se il file è ad accesso diretto e si vogliono registrare i dati iniziando da una parola specificata del file, l'istruzione WRITE: deve essere preceduta da una istruzione SETW:.

3. Se l'istruzione WRITE: è eseguita dopo una istruzione FILES, FILE:, SCRATCH: (solo file sequenziali) o RESTORE:, la registrazione sul file inizia dalla prima parola del file stesso.

4. nd deve essere maggiore di zero e minore od uguale al numero di file a cui il programma può accedere contemporaneamente, dichiarato con l'istruzione FILES.

5. Le costanti stringa devono essere specificate tra apici (es. "OLIVETTI").

6. Se il file a cui fa riferimento l'istruzione è ad accesso diretto, conviene non specificare come operando una espressione numerica perchè è difficile prevedere se il risultato ottenuto sarà espresso in singola o doppia precisione (quindi se occuperà 4 od 8 byte sul file esterno). Infatti, solo nel caso in cui tutti gli operandi della espressione hanno valori espressi in singola precisione, il risultato sarà espresso in singola precisione. Se invece, anche un solo operando della espressione è espresso in doppia precisione, il risultato della espressione sarà espresso in doppia precisione. Si ricorda che il risultato ritornato da una funzione numerica di sistema è sempre espresso in doppia precisione.

Esempi

1. Nel seguente esempio i file FILED1, FILED2, FILED3, FILES1, FILES2 e FILES3 hanno una lunghezza di allo-

cazione di 128 byte. Il programma registra nei file suddetti i numeri da 1 a 32, in singola precisione, che, poichè occupano 4 byte ciascuno, richiedono esattamente 128 byte. Dopo l'esecuzione del programma è stampato il contenuto dei file, meno che per i file FILES2 e FILES3 che, del resto, hanno lo stesso contenuto dei file precedenti.

```
LIST
FILE    *WRITE2

0010  OCL SA
0010 FILES FILED1;FILED2;FILED3;FILES1;FILES2;FILES3
0020 FOR I=1 TO 3 STEP 1
0030 SETW :I TO 1
0040 FOR J=1 TO 32 STEP 1
0045 LET A=J
0050 WRITE :I,A
0060 NEXT J
0070 NEXT I
0080 FOR I=4 TO 6 STEP 1
0090 SCRATCH :I
0100 FOR J=1 TO 32 STEP 1
0105 LET A=J
0110 WRITE :I,A
0120 NEXT J
0130 NEXT I
0140 END

END OF LISTING
RUN
**** FORMALLY CORRECT PROGRAM ****
            FILED1
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILED2
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILED3
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32

            FILES1
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16             17             18             19             20
 21             22             23             24             25
 26             27             28             29             30
 31             32
```

2. Il programma sottostante registra le costanti numeriche da 1 a 16 nei file dati dell'esempio precedente. Si noti come le costanti numeriche sono registrate nei file in doppia precisione (infatti su 128 byte vengono registrati 16 dati, un dato ogni due parole, 8 byte del file). Dopo l'esecuzione del programma è stato stampato il contenuto relativo ad ogni file.

```
LIST
FILE    *WRITE1


0010 FILES FILED1;FILED2;FILED3;FILES1;FILES2;FILES3
0020 FOR I=1 TO 3 STEP 1
0030 SETW :I TO 1
0050 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
0070 NEXT I
0080 FOR I=4 TO 6 STEP 1
0090 SCRATCH :I
0110 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
0130 NEXT I
0140 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
            FILED1
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16

            FILED2
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16

            FILED3
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16

            FILES1
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16

            FILES2
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16

            FILES3
 1              2              3              4              5
 6              7              8              9             10
 11             12             13             14             15
 16
```

3. Il programma sottostante mostra l'impiego dell'istruzione APPEND: per aggiungere al file sequenziale FILES1 la stringa "Manuale generale". Il file viene poi letto dallo stesso programma.

```
LIST
FILE    *WRITE4

0010 FILES FILES1
0020 APPEND :1
0030 WRITE :1,"Manuale generale"
0040 RESTORE :1
0050 FOR I=1 TO 10 STEP 1
0060 READ :1,A$ EOF 90
0070 PRINT A$
0080 NEXT I
0090 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
OLIVETTI
OLIVETTI
Manuale generale
```

4. La routine sottostante mostra come per leggere un dato in un file ad accesso diretto, dopo averlo registrato, si deve spostare con l'istruzione SETW: il relativo pointer all'inizio della parola da cui è stato registrato.

```
LIST
FILE    *WRITE5

0005 DCL 20A$
0010 FILES FILED1
0020 WRITE :1,"Primo dato del file"
0030 SETW :1 TO 1
0040 READ :1,A$
0050 PRINT A$
0060 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
Primo dato del file
```

5. In questo programma si può vedere come agisce l'opzione EOF in una istruzione riferita ad un file ad accesso diretto (FILED1) ed in un file sequenziale (FILES1).

```
LIST
FILE    *WRITE6

0010 FILES FILED1;FILES1
0020 SCRATCH :2
0030 LET I=0
0040 LET I=I+1
0050 IF I>2 THEN 90
0060 PRINT
0070 WRITE :I,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19 EOF 270
0090 SETW :1 TO 1
0100 RESTORE :2
0120 LET I=0
0130 LET I=I+1
0140 IF I>2 THEN 290
0150 PRINT
0160 PRINT
0170 PRINT "Ecco il contenuto del file numero designatore";I;":"
0180 FOR J=1 TO 17 STEP 1
0190 READ :I,A EOF 250
0200 PRINT A,
0210 NEXT J
0240 GOTO 290
0250 PRINT "Nel file con numero designatore";I;"non vi sono piu` dati."
0260 GOTO 130
0270 PRINT "Nel file con numero designatore";I;"non ci stanno piu` dati!"
0280 GOTO 40
0290 END

END OF LISTING


RUN

Nel file con numero designatore 1 non ci stanno piu` dati!

Nel file con numero designatore 2 non ci stanno piu` dati!


Ecco il contenuto del file numero designatore 1 :
 1             2             3             4             5
 6             7             8             9             10
 11            12            13            14            15
 16           Nel file con numero designatore 1 non vi sono piu` dati.


Ecco il contenuto del file numero designatore 2 :
 1             2             3             4             5
 6             7             8             9             10
 11            12            13            14            15
 16           Nel file con numero designatore 2 non vi sono piu` dati
```

6. Il programma sottostante mostra come si può aggiornare un dato in un file ad accesso diretto (FILED1) prelevando i dati dallo stesso file e da un altro file dati. Per vedere da quali parole dei file sono prelevati i dati e come il file FILED1 si modifica è stato stampato il contenuto relativo ai suddetti file.

```
EXEC FLPRINT,FILED1
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16
END OF PRINT
EXEC FLPRINT,FILED2
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16               17               18               19               20
 21               22               23               24               25
 26               27               28               29               30
 31               32
END OF PRINT
EXEC FLPRINT,FILED3
 1                2                3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16               17               18               19               20
 21               22               23               24               25
 26               27               28               29               30
 31               32
END OF PRINT


OLD *WRITE7
LIST
FILE    *WRITE7


0010 FILES FILED1,FILED2,FILED3
0020 SETW :2 TO 10
0030 SETW :3 TO 5
0040 SETW :1 TO 3
0050 READ :1,A
0060 READ :2,B
0070 READ :3,C
0080 LET A=A*B*C
0090 SETW :1 TO 3
0100 WRITE :1,A
0110 END

END OF LISTING


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


EXEC FLPRINT,FILED1
 1                100              3                4                5
 6                7                8                9                10
 11               12               13               14               15
 16
END OF PRINT
```