Istruzione DATA

Funzione

Crea un file dati interno al programma.

Formato

DATA num-constant / string-constant [, num-constant / string-constant] ...

dove:
num-constant
è una costante numerica
string-constant
è una costante stringa

Azione

Prima dell'esecuzione del programma viene generato, in memoria principale, un file dati interno al programma stesso, composto con tutti i dati numerici e/o stringa specificati con tutte le istruzioni DATA del programma. I dati sono ordinati in sequenza secondo il numero di linea delle istruzioni DATA e, nell'ambito di ogni istruzione DATA, da sinistra a destra. Al file dati interno così prodotto viene associato un pointer che indica quale dato deve essere assegnato di volta in volta alle variabili di programma che compaiono nelle istruzioni READ o MAT READ, quando queste sono eseguite. Quando inizia l'esecuzione del programma, il suddetto pointer indica il primo dato del file interno, che coincide con la prima costante specificata con la prima istruzione DATA, in ordine di numero di linea. I dati contenuti nel file dati interno sono assegnati alle variabili di programma quando sono eseguite le istruzioni READ o MAT READ e dopo ogni assegnazione il pointer viene associato al successivo dato del file suddetto.

Note

1. La istruzione DATA non è una istruzione eseguibile.

2. Le istruzioni DATA si possono collocare in qualun-

que punto del programma.

3. Le costanti stringa specificate in una istruzione DATA sono racchiuse tra virgolette, ma possono essere specificate senza essere racchiuse tra virgolette se i caratteri che le compongono non sono:

   - spazi iniziali e/o finali
   - il carattere virgola (,)

   in questo caso la costante stringa che viene assegnata al file dati interno è la sequenza di caratteri compresa tra il primo e l'ultimo carattere diverso da spazio. In particolare, un numero può essere assegnato ad una variabile stringa specificandolo, nella corrispondente istruzione DATA, racchiuso o non racchiuso tra virgolette (nel secondo caso il numero è rappresentato in memoria principale come una sequenza di caratteri numerici).

4. Il carattere virgolette (") non può far parte di una costante stringa specificata in una istruzione DATA. Per assegnare tale carattere ad una variabile stringa si deve usare l'istruzione RKB (vedi la istruzione omonima).

5. La virgola separa le costanti specificate in una istruzione DATA, per cui è importante non interporla all'interno di una costante, se questa non è racchiusa tra virgolette, altrimenti quest'ultima è assegnata alle variabili di programma divisa in due parti.

6. Se ad una variabile numerica in singola precisione è assegnata una costante numerica nel formato in virgola mobile, con una mantissa con più di 6 cifre significative ed un esponente compreso tra -63 e +63, la mantissa viene troncata ed il sistema non fornisce alcuna segnalazione di errore.

7. Se ad una variabile numerica dichiarata in singola precisione viene assegnato un dato numerico nel formato in virgola mobile con un esponente maggiore di 63, il sistema assegna alla variabile suddetta il valore 9.99999E+63 oppure -9.99999E+63 e dà una segnalazione di errore recuperabile (OVERFLOW)commutando nello stato di debugging (tasto di console STEP acceso).

8. Se ad una variabile numerica dichiarata in singola precisione viene assegnato un valore numerico nel formato in virgola mobile con un esponente minore di -63 il sistema assegna alla variabile suddetta il valore zero e dà una segnalazione di errore recuperabile (UNDERFLOW) commutando nello stato di debugging.

9. Mediante l'istruzione RESTORE si può riassociare il pointer del file dati interno al primo dato, riprendendo così l'assegnazione delle costanti alle variabili di programma dall'inizio del file stesso.

10. Se, dopo aver assegnato ad una variabile di programma l'ultimo dato di un file interno, non si riporta il pointer all'inizio del file mediante l'istruzione RESTORE, l'esecuzione di una successiva istruzione READ o MAT READ produce una segnalazione di errore e l'interruzione della esecuzione del programma. Premendo BREAK il sistema commuta nellllo stato comandi.

11. Ad una variabile numerica non deve essere assegnata una costante stringa.

12. Ad una variabile stringa non deve essere assegnata una costante stringa con più caratteri della sua lunghezza di allocazione.

13. Per ulteriori informazioni si vedano le istruzioni READ, MAT READ e RESTORE.

Esempi

1. Dalla stampa prodotta con l'esecuzione del programma sottostante si può notare come le istruzioni DATA sono considerate ordinate secondo il numero di linea e per esse non hanno alcun effetto le istruzioni di salto (es. GOTO). Se sono presenti in cicli ripetitivi (FOR/NEXT) sono considerate una volta sola. Si noti, infine, come sia possibile specificare delle costanti stringa senza che siano racchiuse tra virgolette.

```
LIST
FILE


0010 DCL 19(H$,G$)
0020 DATA !!!!!,&&&&&,'''''',``````
0030 GOTO 100
0040 DATA Olivetti P6066,1980
0050 FOR I=1 TO 10 STEP 1
0060 PRINT I,
0070 DATA " Manuale Generale "
0080 NEXT I
0090 GOTO 130
0100 DATA 123456789,987654321
0110 DATA Ultima assegnazione
0120 GOTO 50
0130 READ A$,B$,C$,D$,E$,F$,G$,A,B,H$
0140 PRINT "A$=";A$,"B$=";B$,"C$=";C$,"D$=";D$
0150 PRINT "E$=";E$,"F$=";F$,"G$=";G$
0160 PRINT "A=";A,"B=";B
0170 PRINT "H$=";H$
0180 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1              2              3              4              5
 6              7              8              9              10
A$=!!!!!       B$=&&&&&       C$='''''       D$=``````
E$=Olivetti P6066             F$=1980        G$= Manuale Generale
A= 1.2345679E+08              B= 9.8765432E+08
H$=Ultima assegnazione
```

2. Dalla stampa prodotta con l'esecuzione del programma sottostante si nota che:

   - il primo dato numerico viene assegnato alla variabile A, dichiarata in singola precisione, troncando il valore della mantissa alle prime sei cifre significative e non viene data segnalazione di errore perchè l'esponente è compreso tra -63 e +63

   - l'assegnazione dei successivi dati numerici alle relative variabili, dichiarate in singola precisione, provoca ogni volta una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging (luce di console STEP accesa); premendo il tasto CONTINUE, dopo ogni segnalazione, l'esecuzione del programma riprende ed alle rispettive variabili sono assegnati i valori che si possono vedere (B=9.99999E+63 e D=-9.99999E+63, perchè i valori da assegnare cadono nelle zone di OVERFLOW per la singola precisione; C=Ø ed E=Ø, perchè i valori da assegnare cadono nelle zone di UNDERFLOW per la singola precisione).

```
LIST
FILE


0010 DCL S(A,B,C,D,E)
0020 DATA 123456789E10,123456789E70,1234567899E-90,-123456789E70,-123456789E-90
0030 READ A
0040 READ B
0050 READ C
0060 READ D
0070 READ E
0080 PRINT "A=";A,"B=";B,"C=";C
0090 PRINT "D=";D,"E=";E
0100 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 3    IN LINE 40
ERROR 4    IN LINE 50
ERROR 3    IN LINE 60
ERROR 4    IN LINE 70
A= 1.2345600E+18              B= 9.9999900E+63              C= 0
D=-9.9999900E+63              E= 0
```

3. Nel programma sottostante non vi sono sufficienti dati, nel file interno generato dall'istruzione DATA, per soddisfare tutte le assegnazioni dell'istruzione READ per cui viene segnalato un errore e l'esecuzione del programma è sospesa; premendo BREAK il sistema commuta nello stato comandi per cui lo si può modificare. Aggiunti i dati al file interno, il programma è eseguito completamente senza provocare segnalazioni di errore.

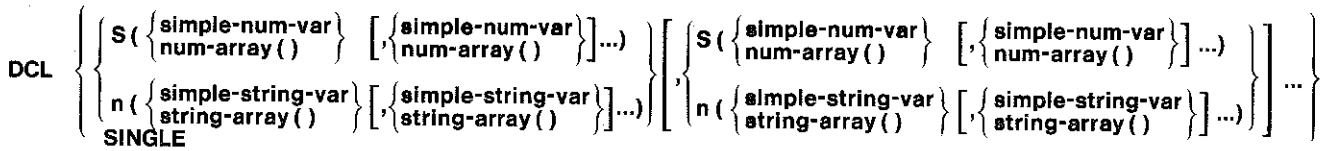
```
LIST
FILE


0010 DATA A,B,C,D
0020 READ A$,B$,C$,D$,E$,F$,G$
0030 PRINT "A$=";A$,"B$=";B$,"C$=";C$,"D$=";D$
0040 PRINT "E$=";E$,"F$=";F$,"G$=";G$
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 88  IN LINE 20
FET 10
0010 DATA A,B,C,D
0010 DATA A,B,C,D,E,F,G
RUN
**** FORMALLY CORRECT PROGRAM ****
A$=A          B$=B          C$=C          D$=D
E$=E          F$=F          G$=G
```

Istruzione DCL

Funzione

Dichiara la lunghezza di allocazione delle variabili stringa e la singola precisione per le variabili numeriche.

Formato



dove:

S
è l'omonimo carattere e specifica che il valore delle variabili numeriche indicate tra parentesi deve essere rappresentato in singola precisione

simple-num-var
è una variabile numerica il cui valore numerico deve essere rappresentato in singola precisione

num-array ( )
indica una variabile multipla (ad una o due dimensioni) numerica i cui elementi (variabili con indice) devono contenere valori numerici rappresentati in singola precisione

n
è un numero intero compreso tra 1 e 1023 che specifica la lunghezza di allocazione delle variabili stringa indicate tra parentesi

simple-string-var
indica una variabile stringa per la quale sono allocati, in memoria principale, n caratteri

string-array ( )
indica una variabile multipla stringa (ad una o due dimensioni) per i cui elementi (variabili con indice) sono allocati n byte in memoria principale

SINGLE
è la parola omonima e specifica che i valori nume-

rici di tutte le variabili numeriche del programma devono essere rappresentati in singola precisione.

Azione

L'istruzione DCL dichiara esplicitamente lo spazio di memoria principale che deve essere riservato alle variabili di programma (vedi appendice F).

L'opzione S, con una o più variabili numeriche tra parentesi, specifica quali variabili numeriche, semplici o con indice, devono avere i relativi valori rappresentati in singola precisione. I valori rappresentati in singola precisione sono elaborati più rapidamente di quelli rappresentati in doppia precisione.

L'opzione n, con una o più variabili stringa tra parentesi, specifica quali variabili stringa, semplici o con indice, potranno avere, come valore, n caratteri.

L'opzione SINGLE specifica che i valori di tutte le variabili numeriche del programma, semplici o con indice, devono essere rappresentati in singola precisione.

Note

1. Se l'opzione S si riferisce ad una sola variabile numerica, semplice o multipla, le parentesi non sono necessarie.

2. Se una opzione n si riferisce ad una sola variabile stringa, semplice o multipla, allora le parentesi non sono necessarie.

3. L'istruzione DCL non è esecutiva.

4. In un programma si possono avere più istruzioni DCL che si riferiscono alla stessa variabile stringa; in questo caso viene assunta la dichiariazione fatta con l'istruzione con numero di linea più alto. In particolare, se per una variabile stringa si fanno più dichiarazioni nell'ambito della stessa istruzione, viene assunta quella più a destra.

5. Se una variabile numerica non compare in alcuna istruzione DCL e nel programma non vi è una istruzione DCL con l'opzione SINGLE, allora i valori della variabile suddetta sono rappresentati in doppia precisione (10 byte sono riservati in memoria principale). Se la variabile numerica suddetta è una variabile multipla, tutti i valori delle variabili con indice che la compongono sono rappresentati in doppia precisione.

6. Se una variabile stringa non compare in alcuna istruzione DCL, allora i valori della variabile suddetta possono essere composti da 16 caratteri al massimo. Se la variabile stringa suddetta è una variabile multipla, tutti i valori delle variabili con indice che la compongono possono essere composti da 16 caratteri al massimo.

7. Il formato DCL SINGLE dichiara la singola precisione solamente per le variabili numeriche, non per le costanti numeriche ed i valori delle funzioni di sistema o i risultati delle espressioni numeriche.

Esempi

1. Nella stampa prodotta dal programma sottostante si vede come i valori delle variabili numeriche sono assunti in singola precisione, infatti i valori ad esse assegnati dalle istruzioni READ sono tutti troncati alla sesta cifra significativa. Alle variabili stringa sono state assegnate sequenze di caratteri con un numero di caratteri eguale al massimo dichiarato nelle relative istruzioni DCL.

```
LIST
FILE      DCL

0010 DCL SA,SB,10A$,20B$
0020 DCL 20(A$(),B$(),C$())
0030 DCL SA(),SB(),10D$()
0040 READ A,B,A(1),A(2),A(10),B(1,1),B(2,2),B(10,10)
0050 READ A$,B$,C$,A$(1),A$(2),A$(10)
0060 READ B$(1,1),B$(2,2),B$(5,5)
0070 DATA 123456789,987654321,123456789,987654321,123456789,987654321,123456789
0080 DATA 987654321,AAAAAAAAAA,BBBBBBBBBBBBBBBBBB,CCCCCCCCCCCCCCCCCCC
0090 DATA 12345678901234567890,22222222222222222222,ZZZZZZZZZZZZZZZZZZZZ
0100 DATA $$$$$$$$$$$$$$$$$$$$,********************,!!!!!!!!!!!!!!!!!!!!
0120 PRINT "A=";A,"A(1)=";A(1)
0130 PRINT "A(2)=";A(2),"A(10)=";A(10)
0140 PRINT "B(1,1)=";B(1,1),"B(2,2)=";B(2,2)
0150 PRINT "B(10,10)=";B(10,10)
0160 PRINT "A$=";A$,"B$=";B$
0170 PRINT "C$=";C$,"A$(1)=";A$(1)
0180 PRINT "A$(2)=";A$(2),"A$(10)=";A$(10)
```

```
0190 PRINT "B$(1,1)=";B$(1,1),"B$(2,2)=";B$(2,2)
0200 PRINT "B$(5,5)=";B$(5,5)
0210 END

END OF LISTING

RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1.2345600E+08                A(1)= 1.2345600E+08
A(2)= 9.8765400E+08             A(10)= 1.2345600E+08
B(1,1)= 9.8765400E+08           B(2,2)= 1.2345600E+08
B(10,10)= 9.8765400E+08
A$=AAAAAAAAAA   B$=BBBBBBBBBBBBBBBBBBBB
C$=CCCCCCCCCCCCCCCCCCCC         A$(1)=12345678901234567890
A$(2)=22222222222222222222      A$(10)=ZZZZZZZZZZZZZZZZZZZZ
B$(1,1)=$$$$$$$$$$$$$$$$$$$$    B$(2,2)=********************
B$(5,5)=!!!!!!!!!!!!!!!!!!!!
```

2. Si noti come, se si fanno più dichiarazioni relative ad una stessa variabile, l'ultima dichiarazione è quella che vale.

```
NEW
10 DCL 10 A$
20 DCL 15A$
30 DCL 5B$(),10B$()
40 READ A$,B$(10)
50 DATA ABCDEFGHILMNOPQ,AAAAAAAAAA
60 PRINT "A$=";A$,"B$(10)=";B$(10)
70 END
RUN
**** FORMALLY CORRECT PROGRAM ****
A$=ABCDEFGHILMNOPQ              B$(10)=AAAAAAAAAA
```

3. Nel programma sottostante non esistono istruzioni DCL per cui per le variabili utilizzate valgono le dichiarazioni implicite che prevedono la doppia precisione per la rappresentazione dei valori delle variabili numeriche (il numero stampato è arrotcndato alla ottava cifra significativa, come prevede il formato standard di stampa e non alla sesta cifra significativa) e 16 caratteri per i valori delle variabili stringa.

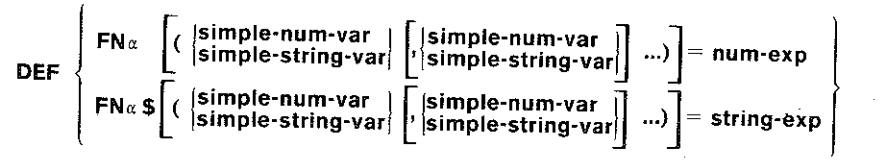
```
NEW
10 READ A,B(1),A$,B$(2)
20 DATA 123456789,987654321,AAAAAAAAAAAAAAAA,BBBBBBBBBBBBBBBB
30 PRINT "A=";A,"B(1)=";B(1)
40 PRINT "A$=";A$,"B$(2)=";B$(2)
50 END
RUN
**** FORMALLY CORRECT PROGRAM ****
A= 1.2345679E+08                B(1)= 9.8765432E+08
A$=AAAAAAAAAAAAAAAA             B$(2)=BBBBBBBBBBBBBBBB
```

Istruzione DEF

Funzione

Definisce una funzione monolinea.

Formato



dove:

FNα
indica il nome di una funzione monolinea di tipo numerico; α è sostituito da una lettera maiuscola dell'alfabeto inglese

simple-num-var
indica una variabile semplice numerica, definita come parametro della funzione; ad essa sono assegnati i valori specificati nel richiamo di funzione

simple-string-var
indica una variabile semplice stringa, definita come parametro della funzione; ad essa sono assegnati i valori specificati nel richiamo di funzione

num-exp
è una espressione numerica che definisce l'algoritmo della funzione numerica

FNα\$
indica il nome di una funzione monolinea di tipo stringa; α è sostituito con una lettera maiuscola dell'alfabeto inglese

string-exp
è una espressione stringa che definisce l'algoritmo della funzione stringa

Azione

Il formato con FNα definisce una funzione monolinea di tipo numerico.

La parte a sinistra del segno uguale specifica il nome della funzione ed a quali variabili, dette parametri, sono assegnati i valori forniti dagli argomenti. La funzione viene infatti richiamata, in una qualunque istruzione di programma in cui possono apparire delle espressioni numeriche, con il suo nome seguito eventualmente da costanti, variabili ed espressioni, i cui valori sono detti argomenti della funzione e sono assegnati nell'ordine, da sinistra a destra, ai parametri della funzione. Gli argomenti devono essere compresi tra parentesi e separati con una virgola. I parametri sono distinti dalle variabili del programma, per cui si possono avere variabili di programma con lo stesso nome delle variabili che sono utilizzate come parametri di una funzione. La parte a destra del segno uguale è una espressione numerica che può contenere come operandi, oltre alle variabili definite come parametri, altre variabili che sono utilizzate nel programma e per tale motivo sono dette variabili globali. Come operandi di detta espressione vi possono essere anche dei richiami di funzione di sistema o definite dall'utente (monolinea o multilinea). Quando viene eseguita una istruzione di programma che contiene il richiamo alla funzione suddetta, alle variabili definite come parametri sono assegnati i valori posseduti in quel momento dagli argomenti e alle variabili globali quelli che esse hanno in quel momento; l'espressione è eseguita ed il valore numerico ottenuto è utilizzato dalla istruzione di programma suddetta.

Il formato con FNα$ definisce una funzione monolinea di tipo stringa. Per essa valgono le stesse definizioni (parametri, argomenti) e considerazioni del punto precedente. In questo caso il risultato ottenuto dalla valutazione dell'espressione a destra del segno uguale è una stringa di caratteri.

Note

1. La funzione FNα$ può essere richiamata in qualunque istruzione di programma in cui può apparire una espressione stringa.

2. L'istruzione DEF non è eseguibile e può essere posta in qualunque punto di un programma, anche dopo l'istruzione in cui è richiamata.

3. In un programma si possono definire 26 funzioni monolinea per ciascun tipo (numerico o stringa) e per ogni definizione si possono utilizzare parametri a-

venti lo stesso nome.

4. Il numero di parametri di una funzione monolinea (numerica o stringa) non può essere maggiore di 15.

5. I valori delle variabili numeriche specificate nell'istruzione DEF come parametri sono rappresentati in doppia precisione, anche se le variabili numeriche, specificate nel richiamo di funzione come argomenti, sono state dichiarate in singola precisione.

6. Le variabili stringa, specificate nell'istruzione DEF come parametri, hanno una lunghezza di allocazione pari a quella dei relativi argomenti.

7. Non si possono definire in uno stesso programma due funzioni monolinea con lo stesso nome.

8. Se vi sono delle segnalazioni di errore riferite ad una funzione durante l'esecuzione di un programma, queste fanno riferimento al numero di linea della istruzione contenente il richiamo di funzione stesso.

9. E' bene non fare delle definizioni ricorsive come, per esempio:

```
          DEF FNA = FNA          (è proibita)
oppure    DEF FNA = X+FNB  }     (non è proibita ma è
  con     DEF FNB = Y+FNA  }     sconsigliabile)
```

10. Gli argomenti specificati nel richiamo di una funzione monolinea devono corrispondere in numero, ordine e tipo, ai parametri specificati nella relativa definizione di funzione.

11. Si noti che lo spazio di memoria principale, occupato dalla variabile usata come parametro, è rilasciato al termine della esecuzione della funzione.

12. Il valore di ritorno di una funzione monolinea di tipo stringa non può contenere più di 16 caratteri.

Esempi

1. Nel programma sottostante si richiama due volte una funzione monolinea numerica, la seconda volta per utilizzare il valore di ritorno di un'altra espressione numerica.

```
LIST
FILE


0010 DEF FNA(A,B,C)=A*B*C+F
0020 LET F=10
0030 LET A=2
0040 LET B=3
0050 LET C=5
0060 LET D=100
0070 LET F=A*B*C
0080 LET Z=9
0090 LET Y=6
0100 LET F=150
0110 PRINT "FNA=";FNA(4,5*5,SQR(D)),"FNA+B=";FNA(Z,Z+A,Y)+B
0120 END

END OF LISTING


RUN
FNA= 1150        FNA+B= 747
```

2. Nel programma sottostante si definisce una funzione monolinea numerica che utilizza come purametri delle variabili stringa.

```
LIST
FILE


0010 DEF FNA(A$,B$)=SCN(A$,B$,1,1)
0020 LET B$="AAABBB"
0030 LET C$="B"
0040 LET Z=FNA(B$,C$)
0050 PRINT "Z=";Z
0060 END

END OF LISTING


RUN
Z= 4
```

3. Ecco un altro esempio di definizione di funzione monolinea numerica con variabili stringa come parametri.

```
NEW
10 DEFFNB(A$,B$,A)=LEN(A$)+LEN(B$)+A
20 D$="DDDDDDDDDD"
30 F$="FFFFF"
40 X=100
50 PRINT "FNB(D$,F$,X)=";FNB(D$,F$,X)
60 END
RUN
**** FORMALLY CORRECT PROGRAM ****
FNB(D$,F$,X)= 115
```

4. Nel programma sottostante viene definita ed utilizzata una funzione monolinea di tipo stringa.

```
LIST
FILE


0010 DEF FNA$(A$,A,B,B$,C$,D$,C,D)=EXT$(A$,A,B)+REP$(B$,C$,D$,C,D)
0020 LET A$="AREA VOLUME PESO"
0030 LET I=6
0040 LET F=11
0050 LET S$="DILLASFIRA"
0060 LET X$="I"
0070 LET Y$="E"
0080 PRINT "FNA$=";FNA$(A$,I,F,S$,X$,Y$,2,1)
0090 END

END OF LISTING




RUN
**** FORMALLY CORRECT PROGRAM ****
FNA$=VOLUMEDELLASFERA
```

Istruzioni DEF/FNEND

Funzione

Definiscono una funzione multilinea.

Formato

DEF { **FN**α / **FN**α**$** } [({ **simple-num-var** / **simple-string-var** } [, { **simple-num-var** / **simple-string-var** }] ...)] [{ **simple-num-var** / **simple-string-var** } [, { **simple-num-var** / **simple-string-var** }] ...]

[Istruzione BASIC]

[**LET**] { **FN*** = **num-exp** / **FN*$** = **string-exp** }

[Istruzione BASIC]

[[**LET**] { **FN*** = **num-exp** / **FN*$** = **string-exp** }]

[Istruzione BASIC]

FNEND

dove:

FNα
: indica il nome di una funzione multilinea di tipo numerico; α è sostituito con una lettera maiuscola dell'alfabeto inglese

FNα$
: indica il nome di una funzione multilinea di tipo stringa; α è sostituito con una lettera maiuscola dell'alfabeto inglese

simple-num-var (compresa tra parentesi tonde)
: indica una variabile numerica semplice, definita come parametro della funzione (numerica o di tipo stringa)

simple-string-var (compresa tra parentesi tonde)
: indica una variabile stringa semplice, definita come parametro della funzione (numerica o di tipo stringa)

simple-num-var (non compresa tra parentesi tonde)
: indica una variabile numerica semplice, definita come variabile locale della funzione (numerica o di tipo stringa)

simple-string-var (non compresa tra parentesi tonde)
indica una variabile stringa semplice, definita come variabile locale della funzione (numerica o di tipo stringa)

FN*
è una variabile numerica (pseudovariabile) non utilizzabile nel programma al di fuori della definizione di funzione multipla numerica, alla quale viene assegnato il valore della espressione numerica specificata alla destra del segno uguale

FN*$
è una variabile stringa (pseudovariabile) non utilizzabile nel programma, al di fuori della definizione di funzione monolinea del tipo stringa, alla quale viene assegnato il valore dell'espressione stringa specificata alla destra del segno uguale

num-exp
indica una espressione numerica, specificata nello ambito di una definizione di funzione multilinea numerica, il cui valore è assegnato alla pseudovariabile numerica FN* specificata alla sinistra del segno uguale

string-exp
indica una espressione stringa, specificata nello ambito di una definizione di funzione multilinea di tipo stringa, il cui valore è assegnato alla pseudovariabile stringa FN*$

FNEND
è una parola chiave BASIC (vedi istruzione FNEND). Specifica che l'istruzione precedente, in ordine di numero di linea, è l'ultima istruzione della definizione di funzione multilinea (numerica o di tipo stringa).

Azione

Il formato con FNα definisce una funzione multilinea di tipo numerico. Le istruzioni BASIC, comprese tra l'istruzione FNα e l'istruzione FNEND, sono eseguite quando in una istruzione del programma compare il richiamo di funzione corrispondente. Il richiamo di funzione è costituito dal nome della funzione, seguito, eventualmente, da una o più costanti, variabili o espressioni, comprese tra parentesi; i cui valori sono detti argomenti della funzione. I valori degli argomenti sono assegnati nell'ordine, da sinistra a destra, alle variabili specificate nella definizione di funzione come parametri. La prima istruzione della definizione di funzione specifica, oltre al nome della

funzione stessa, le variabili che sono utilizzate dalla funzione come parametri ed eventuali variabili locali alle quali sono assegnati i valori con istruzioni contenute nell'ambito della definizione di funzione. Nell'ambito della definizione di funzione vi possono essere una o più istruzioni di assegnazione del valore di una espressione numerica ad una pseudovariabile numerica FN*.

Tutte le istruzioni della definizione di funzione possono contenere variabili definite come parametri, variabili locali e variabili specificate in altre istruzioni del programma (variabili globali). Quando la funzione è richiamata, le variabili definite come parametri sono inizializzate con il valore dei rispettivi argomenti, le variabili locali non sono inizializzate e le variabili globali assumono l'ultimo valore ad esse assegnato nell'ambito del programma.

Il valore numerico restituito dalla funzione è quello assegnato ad FN* prima dell'esecuzione dell'istruzione FNEND.

Il formato in cui compare FNα$ definisce una funzione multilinea di tipo stringa. Valgono per questo formato tutte le considerazioni suddette ma si deve osservare che in questo caso la pseudovariabile a cui viene assegnato un valore è di tipo stringa (FN*$) e l'espressione relativa, a destra del segno uguale, è anch'essa di tipo stringa.

Note

1. Se, durante l'esecuzione di un programma, è incontrata una definizione di funzione, il controllo della esecuzione passa direttamente alla prima istruzione successiva alla istruzione FNEND.

2. Il richiamo di una funzione multilinea può essere utilizzato in tutte le istruzioni di programma in cui può apparire una espressione.

3. Una definizione di funzione multilinea può essere presente in qualunque punto di un programma.

4. In una definizione di funzione multilinea, la somma delle variabili specificate come parametri e delle variabili locali non può essere maggiore di 15.

5. E' importante notare che lo spazio di memoria principale occupato dai parametri e dalle variabili locali è rilasciato al termine della esecuzione della funzione.

6. I parametri e le variabili locali sono distinti dalle variabili globali aventi lo stesso nome. Non si possono avere parametri e variabili locali con lo stesso nome.

7. In un programma si possono definire 26 funzioni multilinea di tipo numerico e 26 funzioni multilinea di tipo stringa. Per ogni definizione di funzione si possono utilizzare variabili locali e parametri aventi lo stesso nome.

8. Alle variabili locali si deve assegnare un valore prima del loro impiego, altrimenti, quando sono eseguite istruzioni che le utilizzano, viene fornita una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging in attesa di una decisione da parte dell'utente (il sistema assegna implicitamente alle variabili locali numeriche il valore zero ed alle variabili locali stringa il valore stringa nulla).

9. Il valore di una variabile specificata come parametro può essere modificato all'interno della funzione.

10. Le variabili definite come parametri devono corrispondere in numero, ordine e tipo agli argomenti specificati nel relativo richiamo di funzione.

11. Il valore di una variabile globale può essere modificato durante l'esecuzione di una funzione ed il nuovo valore viene mantenuto dalla variabile quando il controllo dell'esecuzione ritorna al programma chiamante.

12. Le variabili numeriche definite come parametri sono rappresentate in doppia precisione. Le variabili stringa definite come parametri assumono la stessa lunghezza di allocazione degli argomenti.

13. I valori delle variabili locali numeriche sono rappresentati in doppia precisione. Si può definire la lunghezza di allocazione di una variabile locale

di tipo stringa mediante una istruzione DCL nello ambito della definizione di funzione.

14. I valori delle variabili globali numeriche sono rappresentati con la stessa precisione definita per esse nel programma principale. Le variabili globali di tipo stringa hanno la stessa lunghezza di allocazione definita per esse nel programma principale. Non si possono modificare il tipo di rappresentazione delle variabili globali numeriche e la lunghezza di allocazione delle variabili globali di tipo stringa, utilizzando un'istruzione DCL all'interno di una definizione di funzione multilinea.

15. Nell'ambito di una definizione di funzione multilinea, si può definire la lunghezza di allocazione della pseudovariabile FN*$ con un'apposita istruzione DCL.

16. Non si possono utilizzare alla destra del segno uguale la pseudovariabile FN* ed FN*$.

17. In tutte le istruzioni di una definizione di funzione multilinea in cui può apparire una espressione, possono apparire dei richiami di funzione di sistema o definite dall'utente (monolinea o multilinea).

18. Le segnalazioni d'errore riguardanti l'esecuzione di una funzione multilinea sono riferite al numero di linea dell'istruzione in cui compare il relativo richiamo di funzione. Le segnalazioni di errore segnalate durante la preesecuzione di una funzione multilinea sono riferite alla istruzione FNEND.

19. Non si può definire un'altra funzione multilinea all'interno di una definizione di funzione multilinea.

20. In una definizione di funzione multilinea deve esserci almeno una istruzione di assegnazione di valore alla relativa pseudovariabile (FN* se la funzione è numerica od FN*$ se la funzione è di tipo stringa).

21. Una istruzione esterna od una definizione di funzione multilinea non può passare il controllo del-

l'esecuzione del programma ad una istruzione interna a tale definizione.

22. Una istruzione interna ad una definizione di funzione multilinea non può passare il controllo della esecuzione del programma ad una istruzione esterna a tale definizione.

23. All'interno di una definizione di funzione non vi può essere una istruzione STOP.

24. In un programma una funzione multilinea con un dato nome può essere definita una sola volta.

25. Una funzione multilinea può richiamare se stessa.

Esempi

1. Nel programma sottostante viene riportata una definizione di funzione multilinea numerica. I parametri sono A,B e C. Le variabili locali sono D,E ed F. Si utilizzano anche le variabili globali H e C$. Gli argomenti sono 5,6 ed 8. Si dimostra come il valore ritornato da FN* è quello calcolato nella ultima istruzione di assegnazione relativa eseguita prima di FNEND. In un caso tale valore è calcolato con la prima espressione riferita ad FN*, nell'altro caso con la seconda espressione riferita ad FN*. Per terminare l'esecuzione di questo programma si deve premere il tasto di console BREAK.

```
LIST
FILE


0010 DEF FNA(A,B,C)D,E,F
0020 DISP "INTRODUCI D";
0030 INPUT D
0040 DISP "INTRODUCI E";
0050 INPUT E
0060 DISP "INTRODUCI F";
0070 INPUT F
0080 DISP "SALTO";
0090 INPUT C$
0100 IF C$="SI" THEN 140
0110 PRINT "NON HO SALTATO L'ISTRUZIONE 120"
```

```
0120 LET FN*=(A+B+C+D+E+F)/H
0130 GOTO 160
0140 PRINT "HO SALTATO L'ISTRUZIONE 120"
0150 LET FN*=A+D+LEN(A$)
0160 FNEND
0170 LET H=100
0180 LET A$="Olivetti P6066"
0190 PRINT "FNA=";FNA(5,6,8)
0200 DISP "VUOI RIPETERE IL PROGRAMMA";
0210 INPUT B$
0220 IF B$="SI" THEN 170
0230 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
INTRODUCI D?
1
INTRODUCI E?
2
INTRODUCI F?
3
SALTO?
NO
NON HO SALTATO L'ISTRUZIONE 120
FNA= .25
VUOI RIPETERE IL PROGRAMMA?
SI
INTRODUCI D?
1
INTRODUCI E?
2
INTRODUCI F?
3
SALTO?
SI
HO SALTATO L'ISTRUZIONE 120
FNA= 20
VUOI RIPETERE IL PROGRAMMA?
SI
INTRODUCI D?
20
INTRODUCI E?
30
INTRODUCI F?
40
SALTO?
NO
NON HO SALTATO L'ISTRUZIONE 120
FNA= 1.09
VUOI RIPETERE IL PROGRAMMA?
NO
```

2. Nel programma sottostante si può vedere che se si hanno variabili locali e variabili globali con lo stesso nome, il valore utilizzato dalle istruzioni della funzione multilinea e quello corrispondente alle variabili locali. Quando è eseguita l'istruzione 65, il sistema commuta nello stato di debugging. Premendo B ed END OF LINE sul display appare il valore della variabile globale B.

```
LIST
FILE


0010 DEF FNA B
0020 LET B=10
0030 PRINT "B=";B
0040 LET FN*=B
0050 FNEND
0055 LET B=15
0060 PRINT "FNA=",FNA
0065 STOP
0070 END

END OF LISTING



RUN
B= 10
FNA= 10
STOP       IN LINE 65
B
 15
```

3. Nel programma sottostante si vede come un parametro ed una variabile globale possono avere lo stesso nome. E' l'argomento B che fornisce il valore al parametro A.

```
LIST
FILE


0010 DEF FNZ(A)
0020 PRINT "A=";A
0030 LET FN*=2*A
0040 FNEND
0050 LET A=115
0060 LET B=5
0070 PRINT "FNZ=";FNZ(B)
0080 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A= 5
FNZ= 10
```

4. Nel programma seguente viene utilizzata una funzione multilinea di tipo stringa.

```
LIST
FILE     WWW


0010 PRINT "SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO"
0020 DCL 256 G$,80(F$,H$,I$,X$,K$,L$,Z$,U$),1600$,500M$
0030 LET F$="Il Sistema P6066 puo` funzionare"
0040 LET H$=" in diversi modi che assumono il nome di Stati,"
0050 LET I$=" Lo Stato Comandi e` il modo di funzionamento che"
0060 LET X$=" permette di introdurre ed eseguire tutti i comandi di sistema,"
0070 LET K$="Lo Stato di Debugging e` il modo di funzionamento che per"
0080 LET L$="mette l'impiego degli strumenti di verifica dei programmi,"
00902$="Lo Stato Calcoli Immediati e` il modo di funzionamento del Sistema in c"
0100 LET U$="ui si possono eseguire immediatamente delle espressioni numeriche,"
0110 LET O$=Z$+U$
0120 PRINT FNA$("FINE"," DI")
0130 DEF FNA$(A$,B$)C$
0140 DISP "VUOI MODIFICARE UN TESTO";
0150 INPUT S$
0160 IF S$="SI" THEN 340
0170 FILES CODISO
0180 SCRATCH :1
0190 FOR I=0 TO 255 STEP 1
0200 WRITE :1,CHR$(I)
0210 NEXT I
0220 DISP "REGISTRATI I CODICI ISO"
0230 DELAY 60
0240 LET C$=" CODICI"
0250 LET FN*$=A$+B$+C$
0260 RESTORE :1
0270 LET G$=""
0280 FOR I=0 TO 255 STEP 1
0290 READ :1,D$
0300 LET G$=G$+D$
0310 NEXT I
0320 PRINT G$
0330 GOTO 450
0340 DISP "MODIFICO UN TESTO"
0350 LET C$=" TESTO"
0360 DELAY 100
0370 LET FN*$=A$+B$+C$
0380 LET M$=F$+H$+I$+X$+K$+L$
0390 PRINT "Il vecchio testo e`:"
0400 PRINT M$
0410 PRINT
0420 PRINT
0430 PRINT "Il nuovo testo e`:"
0440 PRINT REP$(M$,EXT$(M$,81,191),O$,1,1)
0450 FNEND
0460 END

END OF LISTING



RUN
SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO
VUOI MODIFICARE UN TESTO?
SI
MODIFICO UN TESTO
Il vecchio testo e`:
Il Sistema P6066 puo` funzionare in diversi modi che assumono il nome di Stati,
Lo Stato Comandi e` il modo di funzionamento che permette di introdurre ed esegu
ire tutti i comandi di sistema,Lo Stato di Debugging e` il modo di funzionamento
 che permette l'impiego degli strumenti di verifica dei programmi,


Il nuovo testo e`:
Il Sistema P6066 puo` funzionare in diversi modi che assumono il nome di Stati,
Lo Stato Calcoli Immediati e` il modo di funzionamento del Sistema in cui si pos
sono eseguire immediatamente delle espressioni numeriche,Lo Stato di Debugging e
` il modo di funzionamento che permette l'impiego degli strumenti di verifica de
i programmi,
FINE DI TESTO
```

```
RUN
SE RISPONDI NO LA FUNZIONE REGISTRA I CODICI ISO
VUOI MODIFICARE UN TESTO?
NO
REGISTRATI I CODICI ISO
▇┌┴┘≀▩⁄▫↖→≡↓⇟←▩▩▩▩▩▩▩⁄⊓⊣⊠⌇⸮▩▩▩▩▩ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}¯※$▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥
▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥
▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥
FINE DI CODICI
```

6. Nel programma sottostante è utilizzata una funzione multilinea stringa nella cui definizione di funzione si è specificata, mediante un'istruzione DCL, una lunghezza di allocazione di 256 caratteri per la pseudovariabile FN*$.

```
FILE


0010 DEF FNR$
0020 DCL 256 FN*$
0030 LET FN*$=REP$(B$,EXT$(B$,130,162),C$,3,1)
0040 FNEND
0050 LET C$="*##############$$$$$$$$$$!!!!!!!!!!*"
0060 FILES CODISO
0070 DCL 256 B$,36 C$
0080 LET B$=""
0090 FOR I=0 TO 255 STEP 1
0100 READ :1,A$
0110 LET B$=B$+A$
0120 NEXT I
0130 PRINT "La funzione modifica la sequenza dei caratteri della tabella ISO";
0140 PRINT " fornendo la sequenza di caratteri:"
0150 PRINT
0160 PRINT FNR$
0170 END

END OF LISTING



La funzione modifica la sequenza dei caratteri della tabella ISO
 fornendo la sequenza di caratteri:

▇┌┴┘≀▩⁄▫↖→≡↓⇟←▩▩▩▩▩▩▩⁄⊓⊣⊠⌇⸮▩▩▩▩▩ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}¯※$*##############$$$$$$$$$$!!!!!!!!!!
*▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥
▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥▥
```

Istruzione DELAY

Funzione

Ritarda l'esecuzione dell'istruzione successiva.

Formato

DELAY time

dove:
time
è un numero intero positivo che indica di quanti decimi di secondo deve essere ritardata l'esecuzione dell'istruzione successiva.

Azione

L'istruzione successiva è eseguita dopo tanti decimi di secondo quanti sono specificati con il numero specificato con time.

Note

1. Premendo il tasto di console CONTINUE l'istruzione di programma successiva all'istruzione DELAY viene eseguita immediatamente.

2. L'istruzione DELAY permette l'impiego di successive istruzioni DISP poichè, posta dopo ogni istruzione DISP, permette la lettura del relativo messaggio sul display.

3. L'istruzione DELAY, utilizzata dopo una istruzione DISP, permette la lettura di messaggi con più di 32 caratteri. In questo caso, poichè il messaggio rimane sul display per il tempo programmato, si può, premendo il tasto [→], spostare il testo del messaggio sul display verso sinistra.

   Esempio: La routine sottostante mostra l'impiego dell'istruzione DELAY. Il messaggio prodotto dall'istruzione 10 si può leggere, perchè rimane sul display per 10 secondi.

Premendo il tasto [→] si può spostare verso sinistra il testo sul display.

```
LIST
FILE     DELAY

0010 DISP "Tra dieci secondi vedrai un altro messaggio"
0020 DELAY 100
0030 DISP "Questa routine mostra"
0040 DELAY 20
0050 DISP "l'impiego della istruzione"
0060 DELAY 20
0070 DISP "DELAY"
0080 DELAY 20
0090 END

END OF LISTING


RUN
Tra dieci secondi vedrai un altro messaggio
Questa routine mostra
l'impiego della istruzione
DELAY
```

Istruzione DEPAD

Funzione

Rimuove in una variabile stringa i caratteri di riempimento specificati.

Formato

DEPAD string-var, n

dove:

string-var

è una variabile stringa di cui si vuole modificare il valore, eliminando i caratteri di riempimento specificati con n

n

è un numero intero, compreso tra Ø e 255, che specifica quali sono i caratteri da togliere al valore della variabile specificata con string-var.

Azione

Nella variabile stringa sono rimossi i caratteri corrispondenti nella tabella ISO al numero specificato con n, se detti caratteri sono in coda nella stringa che costituisce il valore della variabile suddetta.

Esempi

1. Vediamo dall'esecuzione del seguente programma come agisce l'istruzione DEPAD. Si noti come in A$ sono tolti gli asterischi in coda e non quelli all'inizio della stringa che ne costituisce il valore.

```
LIST
FILE


0010 DCL 70 (A$,B$,C$),150 D$
0020 LET A$="*** Il sistema P6066 puo` essere **********"
0030 LET B$="utilizzato per eseguire ######"
0040 LET C$="immediatamente dei calcoli algebrici***$$$$$"
0050 LET D$=A$+B$+C$
0060 PRINT "In D$ c'e` il testo:"
0070 PRINT D$
0080 DEPAD A$,42
0090 DEPAD B$,35
```

```
0100 DEPAD C$,36
0110 PRINT
0120 PRINT "Ora in D$ c'e` il testo:"
0130 LET D$=A$+B$+C$
0140 PRINT D$
0150 PRINT
0160 PRINT "Questo perche` in A$,B$ e C$ sono stati tolti i caratteri in coda,"
0170 PRINT "infatti ora i rispettivi contenuti sono:"
0180 PRINT "A$=";A$
0190 PRINT "B$=";B$
0200 PRINT "C$=";C$
0210 END

END OF LISTING


·RUN
In D$ c'e` il testo:
*** Il sistema P6066 puo` essere **********utilizzato per eseguire ######immedia
tamente dei calcoli algebrici***$$$$$

Ora in D$ c'e` il testo:
*** Il sistema P6066 puo` essere utilizzato per eseguire immediatamente dei calc
oli algebrici***

Questo perche` in A$,B$ e C$ sono stati tolti i caratteri in coda,
infatti ora i rispettivi contenuti sono:
A$=*** Il sistema P6066 puo` essere
B$=utilizzato per eseguire
C$=immediatamente dei calcoli algebrici***
```

2. L'esecuzione dell'istruzione DEPAD non modifica, in questo caso, il contenuto di A$. Come si vede, infatti, in coda al contenuto di A$ non esiste il carattere asterisco.

```
LIST
FILE


0010 DCL 23 A$
0020 LET A$="## Olivetti ** P6066 ##"
0030 DEPAD A$,42
0040 PRINT "A$=";A$
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
A$=## Olivetti ** P6066 ##
```

Istruzione DIM

Funzione

Specifica le dimensioni delle variabili multiple di un programma.

Formato

DIM array (rows [, columns]) [, array (rows [, columns])]...

dove:

array
: è il nome di una variabile multipla numerica o di tipo stringa

rows
: è un numero intero positivo che specifica quante sono le componenti (variabili con indice) della variabile multipla ad una dimensione di nome array

columns
: è un numero intero positivo che, insieme al numero specificato con rows, indica quante sono le componenti (variabili con indice) della variabile multipla a due dimensioni di nome array.

Azione

Quando, dentro le parentesi che seguono il nome di una variabile multipla, è specificato un solo numero, la variabile ha una sola dimensione ed è composta da un numero di variabili con indice pari a row.

Quando, dentro le parentesi che seguono il nome di una variabile multipla, sono specificati due numeri separati da una virgola, la variabile ha due dimensioni ed è composta da un numero di variabili con indice pari al prodotto rows * columns.

Note

1. L'istruzione DIM è non esecutiva, quindi può essere specificata in qualunque posizione nell'ambito di un programma.

2. In un programma vi può essere più di una istruzione DIM.

3. In un programma, non vi possono essere variabili multiple ad una dimensione e variabili multiple a due dimensioni, con lo stesso nome.

4. Se in un programma vi sono più istruzioni DIM che si riferiscono alle stesse variabili multiple, viene assunta la dichiarazione relativa all'istruzione DIM con numero di linea più alto.

5. Se in una istruzione DIM compare più volte la stessa variabile multipla, viene assunta la dichiarazione più a destra nell'ambito della istruzione.

6. Una variabile multipla ad una dimensione, che non compare in alcuna istruzione-DIM, ha 10 componenti. Le relative variabili con indice potranno avere un indice variabile da 1 a 10.

7. Una variabile multipla numerica a due dimensioni, che non compare in alcuna istruzione DIM, ha 10*10 componenti.

8. Una variabile multipla stringa a due dimensioni, che non compare in alcuna istruzione DIM, ha 5x5 componenti.

9. Per una variabile multipla a due dimensioni, non si possono dichiarare dimensioni il cui prodotto superi 65535.

10. Alcune istruzioni (vedi le istruzioni MAT alla fine del capitolo) permettono di modificare il numero di righe e di colonne di una matrice numerica, purchè il loro prodotto non superi il prodotto delle dimensioni specificate nella istruzione DIM (oppure 10*10, se la matrice non è specificata in una istruzione DIM).

Esempi

1. La routine sottostante mostra come vengono dichiarate le dimensioni di due variabili multiple numeriche.

```
LIS
FILE


0010 DIM A(20),B(20,15)
0020 LET A(1)=A(20)=100
0030 LET B(1,1)=B(20,15)=12
0040 PRINT A(1),A(20),B(1,1),B(20,15)
0050 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 100              100              12              12
```

2. Vediamo, nella routine sottostante, come nel caso di più dichiarazioni con le istruzioni DIM, quella che vale è l'ultima.

```
FILE


0010 DIM A(20),A(30)
0020 DIM B(12,13)
0030 DIM B(15,14)
0040 LET A(30)=30
0050 LET B(15,14)=1514
0060 PRINT A(30),B(15,14)
0070 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 30               1514
```

3. Nella routine sottostante si vede che l'indice di una variabile con indice può essere maggiore di quello specificato nella istruzione DIM, se una precedente istruzione MAT INPUT modifica le dimensioni della matrice.

```
LIST
FILE


0010 DIM A(5,4)
0020 MAT INPUT A(2,10)
0030 PRINT "A(2,10)=";A(2,10)
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
A(2,10)= 20
```

Istruzione DISP

Funzione

Visualizza dati e testi, sul display, in formato standard.

Formato

dove:

num-exp
: indica una espressione numerica il cui valore deve essere visualizzato sul display

string-exp
: indica una espressione stringa il cui valore deve essere visualizzato sul display

TAB (num-exp)
: è una funzione di sistema che indica la posizione, corrispondente al valore di num-exp arrotondato all'intero più prossimo, indirizzata dal pointer del buffer di display

,
: indica che il pointer del buffer di display deve indirizzare la successiva zona di display

;
: indica che il pointer del buffer di display non deve modificare l'indirizzo

Azione

I valori delle espressioni specificate nella istruzione sono visualizzati sul display, nell'ordine con cui compaiono nella istruzione, come descritto nelle note successive.

La posizione sul display dei caratteri che sono visualizzati 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 display".

L'istruzione DISP senza operandi cancella il contenuto del buffer di display e posiziona il relativo pointer nella prima posizione. Anche i caratteri visualizzati sul display sono cancellati.

Note

1. Il valore di una espressione numerica è visualizzato preceduto da uno spazio (se è un numero positivo) o dal segno meno (se è un numero negativo) e seguito da uno spazio.

2. I numeri interi sono visualizzati con al massimo 8 cifre significative.

3. I numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono visualizzati, 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.

4. I numeri con valore assoluto minore di 0.0999999995, rappresentabili con 8 cifre significative, sono visualizzati nel formato in virgola fissa.

5. Tutti gli altri numeri sono visualizzati nel formato in virgola mobile.

6. Il valore di una espressione stringa è visualizzato con la sequenza dei caratteri che la compongono.

Controllo della posizione dei caratteri nel buffer di display

L'esecuzione di una istruzione DISP genera in un registro di 80 byte, detto buffer di display, la stringa di caratteri corrispondente 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 display, il pointer avanza di una posizione. Quando il buffer di display è pieno, la generazione di un successivo carattere avviene dalla prima posizione del buffer ed il precedente contenuto è cancellato completamente.

Quando l'istruzione DISP è eseguita, il contenuto del buffer di display è visualizzato a partire dalla prima posizione del buffer. Se il buffer di display con-

tiene più di 32 caratteri, per visualizzare i rimanenti caratteri si deve premere il tasto [→] eventualmente con [SHIFT] o [REPEAT].

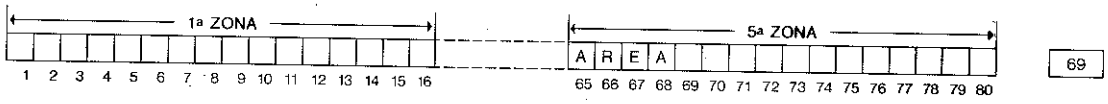


Figura 5-1 Buffer di display e relativo pointer

Ogni volta che una istruzione DISP è eseguita, se la istruzione DISP precedentemente eseguita non ha come carattere finale una virgola (,) od un punto e virgola (;), il contenuto del buffer di display è cancellato e la generazione dei nuovi caratteri riprende dall'inizio.

L'impiego della funzione TAB (num-exp) permette il controllo della posizione dei caratteri nel buffer di display. Il valore dell'espressione numerica num-exp è arrotondato all'intero più prossimo ed assegnato al pointer del buffer di display. Se ad esempio si utilizza in un programma l'istruzione:

```
50 DISP TAB (15); "A"
```

la lettera A sarà visualizzata nella posizione 15 del display che corrisponde alla posizione 15 del buffer di display. Utilizzando le istruzioni in sequenza:

```
50 DISP TAB (50); "A"
60 DELAY 200
```

la lettera A viene generata nella posizione 50 del buffer di display e, premendo i tasti [→] e [SHIFT] contemporaneamente, visualizzata sul display nell'ultima posizione. L'istruzione DELAY mantiene per 20 secondi il carattere A nel buffer di display, permettendone così la visualizzazione sul display.

Come si vede dai suddetti esempi è bene far seguire TAB (num-exp) da punto e virgola (;).

L'impiego della virgola (,) permette di visualizzare i dati distanziati di 16 posizioni l'uno dall'altro. Il buffer di display è diviso in 5 zone, ognuna di 16 posizioni (vedi figura 5-1). In un programma con le seguenti istruzioni:

```
50 DISP
60 DISP"A"
70 DELAY 100
```

l'istruzione 50 pone il pointer del buffer di display in posizione 17 e la successiva istruzione visualizza il carattere A nella posizione 17 del display. In un programma con le seguenti istruzioni:

```
50 DISP "A", "B", "C", "D"
60 DELAY 100
70 DISP "D",
80 DISP "C"
```

l'istruzione 50 genera nel buffer di display il carattere A nella prima posizione, il carattere B nella 17-esima posizione, il carattere C nella 33-esima posizione ed il carattere D nella 49-esima posizione. Sul display A e B appaiono nelle posizioni suddette. Per visualizzare C e D si devono premere i tasti [→] e [REPEAT] contemporaneamente: i due caratteri sono visualizzati distanziati di 16 posizioni sul display. L'istruzione 70 genera il carattere D nella prima posizione del buffer di display e cancella tutti i caratteri precedentemente contenuti nel buffer. Il pointer indirizza la 17-esima posizione del buffer di display (perchè l'istruzione è chiusa con la virgola) e la successiva istruzione DISP visualizza il carattere C in tale posizione.

L'impiego del punto e virgola (;) permette di visualizzare caratteri corrispondenti al valore dell'espressione successiva dalla posizione indirizzata dal pointer in quel momento. Se in un programma si hanno:

```
20 DISP "A"; "B"
30 DELAY 100
40 DISP "C" ;
50 DISP "D"
60 DELAY 100
```

L'istruzione 20 visualizza i caratteri A e B uno di seguito all'altro (nella prima e seconda posizione). Le istruzioni 40 e 50 producono lo stesso risultato per C e D. L'impiego del punto e virgola è particolarmente utile quando l'istruzione DISP è seguita da una istruzione INPUT. In questo caso l'istruzione INPUT mantiene sul display il messaggio precedente seguito da punto interrogativo (?) (vedi nel seguito l'istruzione INPUT).

Nel seguito diamo una tabella che riassume l'impiego della virgola e del punto e virgola come elementi separatori in una istruzione DISP. Il separatore menzionato segue, nella istruzione, il dato specificato nella prima colonna. Nella quarta colonna è indicata la posizione indirizzata dal pointer dopo che i caratteri sono stati inseriti nel buffer.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di display	Posizione indirizzata dal pointer
Espressione numerica	"," virgola	Il valore dell'espressione è registrato nel buffer a partire dalla posizione indirizzata dal pointer, se le posizioni libere sono sufficienti a contenerlo, altrimenti il precedente contenuto del buffer è cancellato ed il valore della espressione è registrato a partire dalla prima posizione.	Il pointer indirizza l'inizio della successiva zona di display.
	";" punto e virgola		Il pointer indirizza la posizione successiva all'ultima occupata dal valore generato nel buffer.
Espressione stringa	"," virgola	La stringa di caratteri corrispondente al valore della espressione è registrata	Il pointer indirizza l'inizio della successiva zona di display.

Tipo di dato	Separatore	Posizione dei caratteri nel buffer di display	Posizione indirizzata dal pointer
		nel buffer a partire dalla posizione indirizzata dal pointer, se le posizioni libere sono sufficienti a contenerla, altrimenti il precedente contenuto del buffer è cancellato ed il valore della espressione è registrato a partire dalla prima posizione.	
	";" punto e virgola		Il pointer indirizza la posizione successiva all'ultima occupata dalla stringa generata nel buffer.
stringa nulla	","	Non viene generato alcun carattere nel buffer di display.	Il pointer indirizza l'inizio della successiva zona di display.
	";" punto e virgola		Il pointer non è modificato.

Tabella 5-1 Impiego della virgola e del punto e virgola con DISP

Esempi

1. Nell'esempio seguente si può osservare come vengono visualizzati i valori numerici sul display (che sono anche stampati perchè PRINT ALL è attivo). Il valore della variabile A viene visualizzato arrotondato all'intero più prossimo, perchè il numero delle cifre significative è maggiore di 8, ma in virgola fissa, perchè è compreso nel campo definito nelle note precedenti. I numeri, come si vede, occupano al massimo 14 posizioni del display. Se si utilizza la virgola coma separatore i numeri sono generati nel buffer di display iniziando dalle posizioni: 1,17,33,49 e 65 (per vedere gli ultimi tre si devono usare i tasti SHIFT e →). Se il separatore è il punto e virgola i numeri sono visualizzati interponendo uno spazio tra un valore e l'altro.

```
LIST
FILE


0010 LET A=0.0999999995
0020 LET B=99999999.4
0030 LET C=0.0000999
0040 LET D=9999999999999
0050 LET E=0.000999999999
0060 PRINT "12345678901234567890123456789012345678901234567"
0070 DISP A;-A;B;-B;C
0080 DELAY 200
0090 DISP A;-A;B;-B;C;-C
0100 DELAY 200
0110 PRINT "1234567890"
0120 DISP TAB(9);-D
0130 DELAY 100
0140 PRINT "123456789012345678"
0150 DISP TAB(6);-D
0160 DELAY 100
0170 PRINT "12345678901234567"
0180 DISP ;-E
0190 DELAY 100
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
12345678901234567890123456789012345678901234567
 .10000000       -.10000000        99999999.       -99999999.        .0000999
 .10000000 -.10000000  99999999. -99999999.  .0000999 -.0000999
1234567890
        -1.0000000E+13
12345678901234567
                -1.0000000E+13
12345678901234567
                -1.0000000E-03
```

2. Nell'esempio seguente si può vedere che, anche quando i valori da visualizzare non sono dei numeri, l'effetto prodotto dalla virgola è uguale a quello dell'esempio precedente. Se i valori da visualizzare sono separati dal punto e virgola, vedi l'istruzione 120, i caratteri sono visualizzati uno di seguito all'altro. La stringa BRASILE è stata stampata perchè il tasto PRINT ALL è attivo, l'utente non la vede sul display, poichè l'istruzione successiva la cancella.

```
LIST
FILE


0010 PRINT "12345678901234567890123456789012345678901234567890123456789012345678901234567"
0020 DISP "A"
0030 DELAY 50
0040 DISP ,"B"
0050 DELAY 5
0060 DISP ,,"C"
0070 DELAY 100
0080 DISP "INTRODUCI 1";
0090 INPUT A
0100 DISP "INTRODUCI 2",
0110 INPUT B
0120 DISP "A";"B";"C"
0130 DELAY 50
0140 DISP "ZA";
0150 DISP "RA"
0160 DELAY 10
0170 DISP "BRASILE"
0180 DISP
0190 DELAY 50
0200 DISP TAB(9),"AAA"
0210 DELAY 50
0220 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
12345678901234567890123456789012345678901234567890123456789012345678901234567
A
                B
                                C
INTRODUCI 1?
1
INTRODUCI 2     ?
2
ABC
ZARA
BRASILE

                AAA
```

3. Vediamo una routine che impiega l'istruzione DISP per qualificare i dati richiesti dall'istruzione INPUT.

```
LIST
FILE


0010 LET A$="AREA"
0020 LET B$=", VOLUME"
0030 LET C$="PESO"
0040 DISP A$+B$;
0050 INPUT D,E
0060 DISP A$;TAB(10);D;TAB(28);"mq"
0070 DELAY 50
0080 DISP B$,
0090 DISP E;TAB(29);"mc"
0100 DELAY 50
0110 DISP "INTRODUCI PESO SPECIFICO";
0120 INPUT P
0130 DISP C$;TAB(10);P*E;TAB(29);"Kg"
0140 DELAY 50
0150 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
AREA, VOLUME?
10,5
AREA       10                mq
, VOLUME           5          mc
INTRODUCI PESO SPECIFICO?
4
PESO       20                Kg
```

DISP USING

Istruzione DISP USING

Funzione

Visualizza dati e testi, sul display, nel formato specificato dall'utente.

Formato

DISP 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 visualizzati sul display

num-exp
è una espressione numerica il cui valore deve essere visualizzato sul display

string-exp
è una espressione stringa il cui valore deve essere visualizzato sul display

Azione

I valori delle espressioni sono visualizzati, da sinistra a destra, nell'ordine con cui sono specificati nella istruzione, e nelle posizioni specificate dalla immagine di formato definita con l'istruzione il cui numero di linea è line-num o contenuta nella variabile stringa string-var.

Si veda la descrizione dell'istruzione IMMAGINE, nel seguito, per una completa comprensione del modo con cui i valori suddetti sono visualizzati.

1. Se vi sono più espressioni nella istruzione DISP USING che campi di formato nella immagine di formato, i valori in eccedenza sono visualizzati con la stessa immagine di formato iniziando dal primo campo dell'immagine.

2 Se vi sono più campi di formato nella immagine di formato che espressioni nella istruzione DISP USING, in corrispondenza dei campi immagine eccedenti, non sono visualizzati caratteri sul display.

3. Le espressioni presenti nella DISP USING devono essere coerenti con i campi di formato ad essi associati: ad una espressione stringa deve corrispondere un campo immagine di stringa.

Esempi

1; Nel seguente programma si vede come sono visualizzati sul display alcuni valori numerici e stringa impiegando l'istruzione DISP USING.

```
LIST
FILE    *DISPUS


0010 PRINT "Le posizioni del display sono:"
0020 PRINT "12345678901234567890123456789012"
0030 :########      ####.######
0040 : ##.########        $$######.###
0050 :'LLLLLLLLLLLLLLLLLL     'RRRRRRRR
0060 DISP " UN INTERO ED UN DECIMALE";
0070 GOSUB 220
0080 INPUT A,B
0090 GOSUB 220
0100 DISP USING 30,A,A*B
0110 DELAY 50
0120 GOSUB 220
0130 DISP "NUMERO NEL FORMATO ESPONENZIALE";
0140 INPUT C
0150 GOSUB 220
0160 DISP USING 40,C,A+B
0170 DELAY 50
0180 GOSUB 220
0190 DISP USING 50,"OLIVETTI","P6066"
0200 DELAY 100
0210 GOTO 240
0220 PRINT "Sul display si vede:"
0230 RETURN
0240 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
Le posizioni del display sono:
12345678901234567890123456789012
Sul display si vede:
 UN INTERO ED UN DECIMALE?
15,25.45
Sul display si vede:
      15       381.750000
Sul display si vede:
NUMERO NEL FORMATO ESPONENZIALE?
12E87
Sul display si vede:
  1.200E+88          $   40.450
Sul display si vede:
OLIVETTI                       P6066
```

2. Nel seguente programma si vede che se il numero d campi della immagine di formato è maggiore del numero di valori da visualizzare, i campi immagine eccedenti sono ignorati.

   Se il numero dei valori da visualizzare (istruzione 90 e 100) è maggiore del numero di campi dell'immagine di formato, i valori eccedenti sono visualizzati con la stessa immagine di formato ed i valori precedenti sono cancellati (in questo caso si vedono solo Y &&&&& Z e 1° dato 15 2° dato 16, mentre gli altri rimangono per un tempo brevissimo sul display).

```
LIST
FILE    *DISPU1

0010 : 1o dato  ###    2o dato    ###
0020 : 'LLLLLL        &&&&&     'RRRRRR
0030 PRINT "Le posizioni sul display sono:"
0040 PRINT "12345678901234567890123456789012"
0050 GOSUB 210
0060 DISP USING 20,"C","D"
0070 DELAY 50
0080 GOSUB 210
0090 DISP USING 20,"W","X","Y","Z"
0100 DELAY 50
0110 : 'L     'L     'L
0120 GOSUB 210
0130 DISP USING 110,"AB"
0140 DELAY 50
0150 GOSUB 210
0160 DISP USING 10,12
0170 DELAY 50
0180 GOSUB 210
0190 DISP USING 10,13,14,15,16
0200 GOTO 230
0210 PRINT "Sul display si vede:"
0220 RETURN
0230 END

END OF LISTING
```

```
RUN
**** FORMALLY CORRECT PROGRAM ****
Le posizioni sul display sono:
12345678901234567890123456789012
Sul display si vede:
 C                &&&&&             D
Sul display si vede:
 W                &&&&&             X
 Y                &&&&&             Z
Sul display si vede:
 AB
Sul display si vede:
 1o dato   12     2o dato
Sul display si vede:
 1o dato   13     2o dato      14
 1o dato   15     2o dato      16
```

Istruzione END

Funzione

Definisce la fine di un programma.

Formato

END

Azione

L'esecuzione del programma termina.

Il contenuto delle variabili del programma è cancellato.

I file esterni riferiti nel programma sono chiusi.

Le operazioni di input/output eventualmente in corso di esecuzione sono portate a termine: il contenuto dei buffer relativi ai file dati esterni è registrato nel file esterno; il contenuto dei buffer di stampa è stampato.

Note

1. In un programma l'istruzione END deve essere sempre presente.

2. Dopo l'esecuzione della istruzione END il programma rimane in memoria principale e sul display appare il messaggio "READY", se non è stato inibito quando è stato registrato nella libreria con il comando SAVE (vedi capitolo 3).

3. Se dopo l'esecuzione dell'istruzione END, si introduce il comando FETCH senza operandi, è trasferita nel buffer di tastiera, e visualizzata, la prima istruzione del programma.

4. L'istruzione END deve avere il numero di linea più alto.

5. In un programma deve esservi una sola istruzione END.