4. BASIC: DATI, VARIABILI, ESPRESSIONI E FILE DATI

Questo capitolo offre al lettore che ha già una certa familiarità con i concetti del linguaggio BASIC la possibilità di effettuare una rapida e completa consultazione di tutte le possibilità offerte dal linguaggio BASIC realizzato per il sistema P6066. Nel capitolo successivo sono descritte tutte le istruzioni del linguaggio BASIC.

I caratteri del linguaggio BASIC

I caratteri che hanno un ruolo sintattico nel linguaggio BASIC sono classificabili in tre categorie:

- caratteri alfabetici
- caratteri numerici
- caratteri speciali

Tutti gli elementi che costituiscono un programma BASIC sono composti con caratteri che appartengono ad una delle suddette categorie, meno le istruzioni di commento (REM) e le costanti stringa che possono essere composte con qualsiasi carattere che si può introdurre da tastiera (vedi set di caratteri P6066 nell'appendice E).

Caratteri alfabetici

I caratteri alfabetici sono tutte le lettere maiuscole dell'alfabeto inglese.

Caratteri numerici

I caratteri numerici sono le cifre del sistema decimale da Ø a 9.

Caratteri speciali

I caratteri speciali sono indicati nella tabella 4-1.

Carattere	Nome
	Spazio
=	Uguale o simbolo di assegnazione
+	Segno più
-	Segno meno
*	Asterisco o segno di moltiplicazione
/	Divisione
↑	Elevamento a potenza
(	Parentesi aperta
)	Parentesi chiusa
,	Virgola
"	Apice
;	Punto e virgola
.	Fine periodo e punto decimale
:	Due punti
>	Maggiore di
<	Minore di
#	Simbolo di numero
$	Simbolo di dollaro

Tabella 4-1 Caratteri speciali

Alcuni caratteri speciali possono essere combinati per comporre i seguenti elementi che hanno un ruolo sintattico nel linguaggio BASIC:

>= maggiore o uguale a
=> maggiore o uguale a
<= minore o uguale a
=< minore o uguale a
<> non uguale a
>< non uguale a

Gli spazi

Gli spazi non hanno un significato sintattico meno che nelle costanti stringa e nella istruzione immagine (vedi capitolo 5) che specifica il formato dei dati da stampare, visualizzare sul display o trasferire ad una unità periferica.

Gli spazi, tuttavia, non sono ammessi all'interno di:

- un numero di linea
- una parola chiave
- un nome di variabile o di funzione

- una costante numerica
- un operatore di confronto.

Dati numerici

Nel linguaggio BASIC i dati numerici sono dati con un valore numerico espresso nel sistema decimale.

Grandezza di un numero

Si definisce grandezza di un numero il suo valore assoluto. Il BASIC P6066 permette di trattare numeri la cui grandezza sia superiore od uguale a 10^{-99} e minore di od uguale a $9.999999999999*10^{99}$.

Precisione

Il linguaggio BASIC permette di utilizzare due tipi di precisione per le variabili numeriche: singola e doppia. Il grado di precisione per le due forme è rispettivamente di 6 e 13 cifre significative. Si può scegliere il tipo di precisione con cui devono essere rappresentati i valori delle variabili di un programma impiegando l'istruzione DCL (vedi capitolo 5).

Nella rappresentazione in singola precisione il valore di una variabile è rappresentato, in memoria principale, con un numero nella forma $M*10^N$. M è un numero che assume la forma X.YYYYY (con $1 \leq X \leq 9$ e $0 \leq Y \leq 9$), mentre N è un numero intero compreso tra -63 e +63. Il valore di una variabile numerica in singola precisione occupa 8 byte in memoria principale. Nella figura 4-1 è rappresentato schematicamente il campo dei valori numerici che possono assumere le variabili numeriche in singola precisione.

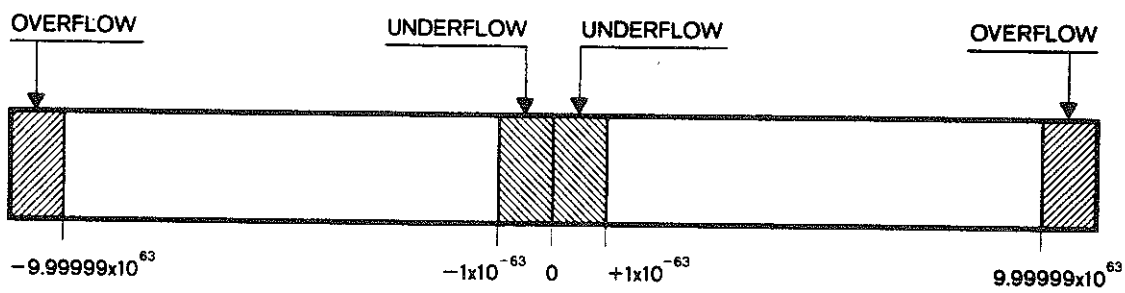

Figura 4-1 Campo di rappresentazione interna dei valori assegnati a variabili in singola precisione

I valori minori di $-9.99999*10^{63}$ e maggiori di $9.99999*10^{63}$, sono nella zona di OVERFLOW: essi esprimono valori più grandi in valore assoluto di quelli rappresentabili in memoria principale in singola precisione. I valori maggiori di $\pm 1*10^{-63}$ e minori di $1*10^{-63}$ (escluso lo zero), sono nella zona di UNDERFLOW: essi esprimono valori minori di quelli rappresentabili in memoria principale in singola precisione.

Nella rappresentazione in doppia precisione il valore di una variabile è rappresentato, in memoria principale, con un numero nella forma $R*10^{E}$. R è un numero che assume la forma X.YYYYYYYYYYY ($1 \leq X \leq 9$ e $0 \leq Y \leq 9$), mentre E è un numero intero compreso tra -99 e +99. Il valore di una variabile numerica in doppia precisione occupa 8 byte in memoria principale. Le espressioni sono calcolate sempre in doppia precisione, se almeno uno degli operandi specificati è espresso in doppia precisione, ed i risultati sono poi arrotondati al tipo di precisione scelto per la variabile a cui sono assegnati. I valori delle variabili numeriche sono assunti in doppia precisione se non vi è una dichiarazione contraria nell'ambito del programma. Nella figura 4-2 è rappresentato schematicamente il campo di rappresentazione dei valori numerici che possono assumere le variabili numeriche in doppia precisione.

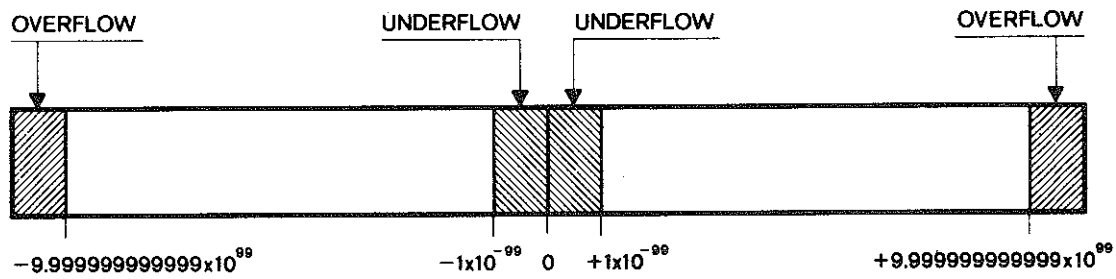

Figura 4-2 Campo di rappresentazione dei valori numerici assegnati a variabili in doppia precisione

I valori minori di $-9.999999999999*10^{99}$ e maggiori di $9.999999999999*10^{99}$ sono nella zona di OVERFLOW: essi esprimono valori più grandi in valore assoluto di quelli rappresentabili in memoria principale in doppia

precisione.

I valori maggiori di $-1*10^{-99}$ e minori di $1*10^{-99}$ (escluso lo zero), sono nella zona di UNDERFLOW: essi esprimono valori minori di quelli rappresentabili in memoria principale in doppia precisione.

Formato dei dati numerici

I dati numerici possono essere introdotti, visualizzati o stampati come:

- interi
- decimali in virgola fissa
- decimali in virgola mobile

La scelta del formato dipende dalla grandezza del numero e dalla precisione richiesta. I numeri possono essere positivi o negativi. I numeri negativi devono essere preceduti dal segno meno. I numeri positivi possono essere preceduti dal segno più.

Numeri interi: I numeri interi sono rappresentati con al massimo 13 cifre significative, precedute dal segno algebrico. Esempi di numeri interi sono:

```
 0          4          +4          -4
 9999999999999 (massimo numero intero)
-9999999999999 (minimo numero intero)
```

Decimali in virgola fissa: I numeri espressi in virgola fissa sono costituiti, al massimo, da 13 cifre significative, precedute opzionalmente dal segno, con interposto tra di esse il punto decimale per distinguere le cifre intere da quelle decimali. Esempi di numeri nel formato in virgola fissa sono:

```
-.86        .7        5.        +2.3
9999999999999.  (massimo valore assoluto esprimibile
in virgola fissa)
```

Decimali in virgola mobile: I numeri espressi in virgola mobile (detta anche rappresentazione scientifica), sono rappresentati con un segno (opzionale se il numero è positivo) seguito da un numero intero od in virgola fissa, seguito dal carattere E. Dopo il carattere E segue un altro numero di due cifre preceduto, eventualmente, dal segno. Il valore di un numero rappresentato in virgola mobile è uguale al numero che

precede il simbolo E (mantissa) moltiplicato per 10, elevato alla potenza con esponente indicato dal numero che segue il simbolo E (esponente). Si osservi che il simbolo E deve essere sempre preceduto da un numero, quindi 10^8 deve essere introdotto come: 1E8. Esempi di numeri nel formato in virgola mobile sono: .99E-5 +1E5 2.E3 -3.0E+1 i cui valori sono rispettivamente: .0000099 100000 2000 -30

Costanti numeriche

Una costante numerica è un numero intero, un numero decimale in virgola fissa od un numero decimale in virgola mobile, il cui valore non viene mai modificato durante l'esecuzione di un programma. Negli esempi che seguono 94, 9.4 e 9.4E-10, sono costanti numeriche:

```
150 LET A=A+94
160 DATA 9.4 , 9.4E-10
```

La costante numerica π è predefinita dal sistema; il suo valore è: 3.141592653590. Per utilizzare π in un programma la si richiama mediante il nome PI, come nell'esempio che segue

```
200 LET A=PI*R*R
```

Una costante numerica occupa un numero di byte, in memoria principale, variabile in funzione del numero di cifre di cui si compone secondo la seguente ella:

Cifre	Byte occupati
1	5
2	6
3	6
4	7
5	7
6	8
7	8
8	9
9	9
10	10
11	10
12	11
13	11

così ad esempio:

123.21	occupa 7 byte
-.83913E-23	occupa 7 byte
1234.5678E12	occupa 9 byte
31	occupa 6 byte
1	occupa 5 byte
1.0E2	occupa 6 byte

Se in un programma la stessa costante è riferita più di una volta si può risparmiare spazio di memoria assegnandola ad una variabile numerica e quindi utilizzando quest'ultima al suo posto.

Variabili semplici numeriche

Una variabile semplice numerica è un dato numerico, richiamato in una istruzione di programma mediante un nome, il cui valore può essere modificato durante l'esecuzione del programma. Il nome associato ad una variabile semplice numerica è costituito da una lettera maiuscola dell'alfabeto inglese o da una lettera seguita da una cifra decimale (da Ø a 9).

Sono esempi di variabili semplici numeriche: Z B6 M2. Ad una variabile semplice numerica può essere assegnato un valore da tastiera (vedi istruzione INPUT) o da programma (vedi istruzioni READ, DATA o LET). In un programma si possono utilizzare fino a 123 variabili semplici numeriche.

Quando s'inizia l'esecuzione di un programma, tutte le variabili numeriche sono inizializzate con un valore "non definito". Se ad una variabile numerica non è stato assegnato un valore, nel momento in cui essa è utilizzata in una istruzione di programma, il sistema le assegna il valore zero ed esegue l'istruzione. Eseguita l'istruzione il sistema è nello stato di debugging e viene visualizzato un messaggio di errore di tipo recuperabile (vedi appendice D). Si può continuare l'esecuzione del programma, premendo il tasto CONTINUE, oppure richiederne la terminazione, premendo il tasto BREAK.

Una variabile semplice numerica è rappresentata in memoria principale in doppia precisione, a meno che non si sia precisata, con l'istruzione DCL, la semplice precisione; in questo caso l'elaborazione è più rapida. Per assegnare il risultato della esecuzione di un'espressione numerica ad una variabile numerica, si

utilizza l'istruzione LET. Se il risultato della suddetta espressione è un valore compreso in una zona di OVERFLOW (vedi paragrafo "Precisione"), alla variabile è assegnato il valore $-9.999999999999*10^{99}$ oppure $9.999999999999*10^{99}$ ($-9.99999*10^{63}$ oppure $9.99999*10^{63}$, se la variabile è stata dichiarata in singola precisione). Se il risultato della suddetta espressione è un valore compreso in una zona di UNDERFLOW, alla variabile è assegnato il valore zero. Sia nel caso di OVERFLOW che nel caso di UNDERFLOW l'esecuzione del programma è interrotta, il sistema commuta nello stato di debugging, e sul display appare un messaggio di errore di tipo recuperabile (vedi appendice D). L'operatore può scegliere di proseguire nell'esecuzione del programma, premendo il tasto di console CONTINUE, o di terminare l'esecuzione del programma, premendo il tasto di console BREAK.

Nota: Una variabile semplice numerica sia in singola che in doppia precisione occupa dieci byte in memoria principale. In entrambi i casi due byte sono occupati dal nome della variabile ed il resto dal valore numerico. Nel caso in cui le variabili semplici siano dichiarate in singola precisione (vedi istruzione DCL) le relative elaborazioni sono eseguite più rapidamente che nel caso in cui esse siano in doppia precisione.

Dati non numerici

Si definiscono dati non numerici tutti quei dati a cui non corrisponde un valore numerico. I dati non numerici si distinguono in:

- costanti stringa
- variabili stringa.

Costanti stringa

Una costante stringa è costituita da una qualunque sequenza di caratteri del set P6066 (vedi appendice E) racchiusa tra virgolette. Il carattere virgolette non può far parte di una costante stringa. In alcuni casi si può definire una costante stringa senza utilizzare le virgolette come inizio e fine della stringa (vedi l'istruzione DATA nel capitolo 5). Esempi di costanti sono:

```
" GRANDEZZA: Temperatura "
"*** FINE DELLA ROUTINE MATRICI***"
" 1815 - 1945 "
```

Si definisce lunghezza di una costante stringa il numero di caratteri che la compongono, inclusi gli spazi ed escluse le virgolette (la cui unica funzione è quella di delimitare l'inizio e la fine della stringa). La massima lunghezza di una costante stringa è di 75 caratteri. Una costante stringa occupa in memoria principale tanti byte quanti sono i caratteri che la compongono, più due byte.

Variabili semplici stringa

Una variabile semplice stringa è costituita da una sequenza di caratteri del set P6066 che può essere modificata durante l'esecuzione di un programma. Le istruzioni di un programma fanno riferimento ad una variabile stringa mediante un nome che è composto da una lettera maiuscola dell'alfabeto (da A a Z) seguita dal simbolo $ oppure da una lettera maiuscola seguita da una cifra decimale (da Ø a 9) e dal simbolo $. Alcuni esempi di nomi di variabili stringa sono:

Z$ B6$ GØ$.

In un programma si possono utilizzare fino a 256 variabili stringa. Il valore può essere assegnato ad una variabile stringa da tastiera (vedi istruzioni INPUT ed RKB) o da programma (vedi istruzioni READ, DATA e LET).

Quando inizia l'esecuzione di un programma, tutte le variabili stringa sono inizializzate con un valore "non definito". Se, quando una variabile stringa è utilizzata in una istruzione di programma, non è stato assegnato ad essa un valore, il sistema le assegna il valore "stringa nulla". La "stringa nulla" è una stringa priva di caratteri; in questo caso, quindi, al nome della variabile suddetta non è associato alcun carattere. L'istruzione viene eseguita ed il sistema commuta nello stato di debugging visualizzando sul display un errore di tipo recuperabile (vedi appendice D). Si può proseguire nella esecuzione del programma, premendo il tasto di console CONTINUE, oppure terminare l'esecuzione, premendo il tasto di console BREAK.

Il massimo numero di caratteri che può essere assegnato ad una variabile stringa (detto lunghezza di allocazione) è specificato mediante una istruzione DCL. Se non viene fatta alcuna dichiarazione, la variabile assume una lunghezza di allocazione di 16 caratteri.

La massima lunghezza di allocazione dichiarabile per una variabile stringa è di 1023 caratteri. Il numero di caratteri che compongono i valori di una variabile stringa è detto lunghezza attuale della variabile stringa. Quando viene assegnata una stringa ad una variabile stringa e le rispettive lunghezze differiscono:

1. Se la stringa ha meno caratteri della lunghezza di allocazione della variabile, quest'ultima assume una lunghezza attuale uguale alla lunghezza della stringa (quindi se, ad esempio, si visualizza il contenuto della variabile, sul display appare solo la stringa assegnata).

2. Se la stringa ha più caratteri della lunghezza di allocazione della variabile, la stringa viene troncata nei caratteri eccedenti tale lunghezza sulla destra e quindi assegnata alla variabile (in questo caso il sistema commuta nello stato di debugging e sul display appare una segnalazione di errore di tipo recuperabile; l'operatore può scegliere di proseguire nell'esecuzione del programma, premendo il tasto di console CONTINUE, oppure di terminare la esecuzione, premendo il tasto di console BREAK).

Nota: Una variabile stringa occupa in memoria principale nove byte, più tanti byte quanti sono stati dichiarati esplicitamente od implicitamente.

Variabili multiple

Si definisce variabile multipla un insieme omogeneo di elementi ognuno dei quali è una variabile numerica od una variabile stringa (i due tipi di variabili non possono coesistere in una variabile multipla).

Una variabile multipla può aver una o due dimensioni. Una variabile multipla ad una dimensione (detta vettore) può essere pensata come una successione naturale di elementi. Una variabile multipla a due dimensioni può essere pensata come una matrice con righe e colonne.

Ad ogni variabile multipla è assegnato un nome. Le regole per l'assegnazione del nome dipendono dal tipo di variabile multipla (vedi paragrafi "Variabili multiple numeriche" e "Variabili multiple stringa"). Agli elementi di una variabile multipla ci si riferisce con lo stesso nome della variabile multipla con in più un

indice (se la variabile è ad una dimensione) o due indici (se la variabile è a due dimensioni) che ne indicano la posizione nell'ambito della variabile suddetta.

Se A è il nome di un vettore i suoi elementi saranno: A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10). Quindi A(4) è il quarto elemento del vettore A. Se Z è il nome di una matrice i suoi elementi saranno:

Z(1,1)	Z(1,2)	Z(1,3)	Z(1,4)
Z(2,1)	Z(2,2)	Z(2,3)	Z(2,4)
Z(3,1)	Z(3,2)	Z(3,3)	Z(3,4)

Quindi Z(2,3) è l'elemento contenuto nella seconda riga e terza colonna della matrice Z. Gli indici possono essere una qualunque espressione numerica il cui valore, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore o uguale alla corrispondente dimensione che è dichiarata come descritto nel paragrafo successivo.

Dichiarazione delle variabili multiple

Dichiarare una variabile multipla significa specificare:

- il numero di dimensioni (una o due)
- il tipo delle variabili che la compongono (numeriche o stringa)
- il numero di variabili componenti per ogni dimensione
- l'occupazione di memoria principale delle variabili componenti.

Le suddette specificazioni possono essere esplicite od implicite ed esse danno di conseguenza luogo ad altrettante dichiarazioni implicite od esplicite. Le dichiarazioni esplicite sono effettuate mediante le istruzioni DIM e DCL, così le istruzioni:

```
10 DIM A(5,7),B(50,70),C(20),A$(25,24),B$(15)
20 DCL S(A(),B()), 25 A$() , 30 B$()
```

dichiarano esplicitamente che:

A e B sono variabili multiple numeriche a due dimensioni (matrici);
C è una variabile multipla numerica ad una dimensione (vettore);
A$ è una variabile multipla a due dimensioni di tipo stringa;
B$ è una variabile multipla ad una dimensione di tipo stringa.

A può avere al massimo 35 componenti (variabili con indice); in esse il primo indice può variare da 1 a 5 ed il secondo da 1 a 7.

B può avere al massimo 3500 componenti (variabili con indice); in esse il primo indice può variare da 1 a 50 ed il secondo da 1 a 70.

C può avere al massimo 20 componenti (variabili con indice da C(1) a C(20)).

A$ può avere al massimo 600 componenti (variabili con indice); in esse il primo indice può variare da 1 a 25 ed il secondo da 1 a 24.

B$ può avere al massimo 15 componenti (variabili con indice da B$(1) a B$(15)).

A e B hanno come componenti variabili numeriche in singola precisione.

Tutte le variabili componenti di A$ possono avere al massimo 25 caratteri.

Tutte le variabili componenti di B$ possono avere al massimo 30 caratteri.

Nota: Per ulteriori informazioni sulle dichiarazioni esplicite dei diversi attributi delle variabili di un programma si vedano le istruzioni DCL e DIM (capitolo 5).

Le dichiarazioni implicite sono effettuate come conseguenza dell'assenza di istruzioni DIM e DCL riferite alle variabili multiple. Ad esempio in un programma in cui compaiono le istruzioni:

```
90 MAT INPUT A$
100 LET Z(5) = 99
120 LET Y(5,7) = -990.5
250 MAT INPUT A
260 LET B$(5) = "DEVIAZIONE MEDIA"
```

ma non compaiono istruzioni DIM e DCL riferite alle variabili A, A$, B$, Z ed Y, le istruzioni suddette dichiarano implicitamente che:

Z è una variabile multipla numerica ad una dimensione composta da 10 variabili con indice numeriche (da Z(1) a Z(10)), ognuna rappresentata in doppia precisione in memoria principale;

A$ è una variabile multipla stringa a due dimensioni, composta da 25 variabili con indice stringa; ognuna può avere fino a 16 caratteri ed il primo e secondo indice possono variare da 1 a 5;

B$ è una variabile multipla stringa ad una dimensione, composta da 10 variabili con indice stringa (da B$(1) a B$(10)); ognuna può avere fino a 16 caratteri;

Y ed A sono variabili multiple numeriche a due dimensioni composte da 100 variabili numeriche; ognuna è rappresentata in doppia precisione in memoria principale ed il primo e secondo indice possono variare da 1 a 10.

Nota: Ulteriori informazioni sulle dichiarazioni implicite degli attributi delle variabili di un programma sono contenute nella descrizione delle istruzioni DCL e DIM (vedi capitolo 5).

Attenzione: Le dichiarazioni esplicite prevalgono su quelle implicite. L'ultima dichiarazione esplicita (in ordine di numero di linea) prevale sulle precedenti riferite alle stesse variabili multiple.

Ridimensionamento delle variabili multiple

Le dimensioni delle variabili multiple possono essere modificate da alcune istruzioni di programma (per l'elenco completo vedi le istruzioni di tipo MAT nel capitolo 5). Ad esempio nel seguente programma:

```
10 DIM A(6,7), A$(25,10)
- - -
90 MAT INPUT B(15,2)
100 MAT INPUT A(2,9)
110 MAT READ: 1,A$(6,5)
- - -
9999 END
```

Dopo aver dichiarato, per la matrice A, 42 variabili con indice numeriche (6*7), e per la matrice B 100 variabili con indice numeriche (dichiarazione implicita 10*10), si modificano tali numeri con le istruzioni 90 e 100: infatti A assume 18 variabili con indice numeriche disposte su 2 righe e 9 colonne e B assume 30 variabili con indice numeriche disposte su 15 righe e 2 colonne.

Dopo aver dichiarato, per la variabile multipla a due dimensioni A$, 250 variabili con indice stringa (25* 10), l'istruzione 110 modifica tale numero in 30 (6*5).

Infatti per le variabili multiple si hanno dimensioni di allocazione e dimensioni attuali. Le dimensioni di allocazione sono quelle dichiarate esplicitamente (istruzione DIM) od implicitamente (10,10*10 e 5*5) ed esprimono il massimo numero di componenti delle variabili multiple. Le dimensioni attuali si riferiscono al numero di componenti che sono effettivamente utilizzate dalle istruzioni di programma (tale numero è sempre minore od uguale al precedente).

Variabili multiple numeriche

Una variabile multipla numerica contiene come elementi delle variabili con indice numeriche; può avere una dimensione (vettore) o due dimensioni (matrice). Le istruzioni di programma fanno riferimento ad una variabile multipla numerica mediante il suo nome che è costituito da una lettera maiuscola dell'alfabeto inglese. Così A può essere il nome di una variabile multipla numerica mentre A2 non può esserlo. In un programma si può assegnare lo stesso nome ad una variabile semplice numerica e ad una variabile multipla numerica: così B può indicare contemporaneamente il nome di una variabile semplice numerica ed il nome di una variabile multipla numerica. In un programma non si può, tuttavia, assegnare lo stesso nome ad un vettore e ad una matrice. In un programma si possono u-

tilizzare fino a 26 variabili multiple numeriche.
Una variabile multipla numerica può essere ridimensionata.

Le singole componenti di una variabile multipla numerica ad una dimensione (vettore), sono richiamate nelle singole istruzioni di un programma mediante il nome della variabile multipla ed un indice numerico (per questo sono dette variabili con indice) che ne indica la posizione nell'ambito di essa.

Le singole componenti di una variabile multipla a due dimensioni (matrice) sono richiamate nelle singole istruzioni di programma mediante il nome della variabile multipla e due indici numerici (per questo anche esse sono dette variabili con indice) che ne indicano la posizione nell'ambito di essa. Ad esempio nel seguente programma:

```
10 DIM A(50), B(25,25)
-  -  -
-  -  -
-  -  -
100 LET A(50) = 100
110 LET B(10,10) = 1000
-  -  -
-  -  -
-  -  -
9999 END
```

Dopo aver dichiarato 50 componenti per il vettore A e 625 componenti per la matrice B, l'istruzione 100 assegna il valore 100 al cinquantesimo componente del vettore A, mentre l'istruzione 110 assegna il valore 1000 al decimo componente della decima riga della matrice B.

Quando inizia l'esecuzione di un programma tutte le variabili numeriche con indice sono inizializzate con un valore "non definito" e valgono le stesse considerazioni sottolineate nel caso delle variabili semplici numeriche.

Nota: Per calcolare lo spazio di memoria principale occupato da una variabile multipla numerica si deve utilizzare una delle seguenti formule:

1. Per le variabili multiple i cui elementi sono variabili con indice numeriche in singola precisione:
   N = 10 byte + n * 4 byte.

2. Per le variabili multiple i cui elementi sono variabili con indice numeriche in doppia precisione: N = 10 byte + n * 8 byte.

   Dove N è il numero di byte occupato in memoria principale dalla variabile multipla ed n è il numero di elementi che la compongono.

Variabili multiple stringa

Una variabile multipla stringa contiene come elementi delle variabili con indice stringa e può avere una o due dimensioni. Una variabile multipla stringa è indicata con un nome costituito da una lettera maiuscola dell'alfabeto e dal segno di dollaro: A$ può essere il nome di una variabile multipla stringa mentre A1$ non può esserlo.

In un programma si può assegnare lo stesso nome ad una variabile semplice stringa e ad una variabile multipla stringa: così R$ può indicare nello stesso programma una variabile semplice stringa ed una variabile multipla stringa.

In un programma non si può, tuttavia, assegnare lo stesso nome ad una variabile multipla stringa ad una dimensione e ad una variabile multipla stringa a due dimensioni. Il massimo numero di variabili multiple stringa presenti in un programma è 26.

Le singole componenti di una variabile multipla stringa sono richiamate nelle singole istruzioni di un programma con il nome della variabile multipla seguito da un indice numerico (se la variabile multipla è ad una dimensione) o da due indici numerici (se la variabile multipla è a due dimensioni). Ad esempio, nel seguente programma:

```
10 DIM A$(20), B$(12,12)
-  -  -
-  -  -
-  -  -
100 PRINT A$ (1), B$(1,1)
-  -  -
-  -  -
-  -  -
9999 END
```

Dopo aver dichiarato 20 componenti per la variabile multipla stringa ad una dimensione A$ e 144 componenti per la variabile multipla stringa a due dimensioni B$,

l'istruzione 100 stampa il contenuto della prima componente di A$ ed il contenuto della prima componente della prima riga di B$.

Quando inizia l'esecuzione di un programma, tutte le variabili con indice stringa sono inizializzate con il valore "non definito" e valgono le medesime considerazioni già sottolineate nel caso delle variabili semplici stringa (vedi paragrafo "Variabili semplici stringa").

Nota: Per calcolare lo spazio di memoria principale occupato da una variabile multipla stringa, si deve utilizzare la seguente formula: N = 13 byte + n*(a+2) dove N è il numero di byte occupato in memoria principale della variabile multipla, n è il numero di componenti della variabile multipla ed a è la dimensione di allocazione dichiarata esplicitamente od implicitamente per ogni elemento della variabile multipla.

Nomi delle variabili

Riassumiamo nella tabella 4-2 le regole per la generazione dei nomi delle variabili in un programma BASIC.

Variabile	Nome	Esempi
Variabile semplice numerica	Alfa [Cifra]	Z, JØ, Z9
Variabile semplice stringa	Alfa [Cifra] $	Z$, JØ$, Z9$
Variabile multipla numerica	Alfa	Z, J
Variabile multipla stringa	Alfa $	Z$, J$
Variabile con indice numerica	Alfa (Indice[,Indice])	Z(12),J(14,7)
Variabile con indice stringa	Alfa $ (Indice[,Indice])	Z$(5), J$(8,9)

Dove: Alfa è una lettera maiuscola dell'alfabeto inglese (da A a Z)
Cifra è una cifra decimale (da Ø a 9)
Indice è una espressione numerica che, arrotondata all'intero più prossimo, deve fornire un numero maggiore di zero (nel caso di due indici, il prodotto dei due non deve essere maggiore del prodotto 256*256).

Tabella 4-2 Regole per la generazione dei nomi delle variabili

Area comune

I valori assegnati a variabili semplici e variabili multiple possono essere scambiati fra programmi distinti utilizzando un'istruzione COMMON (vedi capitolo 5).

Funzioni di sistema

Il linguaggio BASIC P6066 fornisce la possibilità di utilizzare nelle istruzioni alcune funzioni di sistema di tipo numerico e stringa. Le funzioni di sistema sono richiamabili nelle singole istruzioni di programma mediante un nome e, se richiesto, uno o più argomenti posti tra parentesi. Gli argomenti sono costanti, variabili o espressioni su cui si applica la funzione stessa. Elenchiamo nei successivi paragrafi i due tipi di funzioni disponibili.

Funzioni numeriche di sistema

Si noti che,oltre alle funzioni di sistema, l'utente può utilizzare delle funzioni numeriche e stringa (funzioni utente) che ha definito nel programma (vedi capitolo 5: istruzioni DEF monolinea e multilinea ed istruzione FKEY#). Una funzione numerica è un algoritmo che, applicato a costanti, variabili od espressioni, che ne costituiscono gli argomenti, fornisce come risultato un numero. Una funzione numerica può essere usata in qualsiasi istruzione in cui possa essere presente una espressione numerica, ad esempio:

```
50 LET Z = SIN(X*PI)
70 PRINT SQR(ABS(Y))
```

Nella tabella 4-3 sono riassunte, in ordine alfabetico, tutte le funzioni numeriche di sistema disponibili; la lettera X indica l'argomento della funzione. Quando l'argomento della funzione COT (X) è π radianti od un suo multiplo, il valore ritornato dalla funzione è + 9.999999999999E+99. Quando l'argomento della funzione TAN (X) è $\frac{\pi}{2} + (K*\pi)$, con K intero maggiore di 1, il valore ritornato dalla funzione è + 9.999999999999E+99. In entrambi i casi il sistema visualizza un messaggio di errore recuperabile commutando nello stato di debugging.

Si noti che le funzioni DET ed RND non hanno argomento. In effetti nella funzione DET l'argomento è implicito ed è rappresentato dall'ultima matrice quadrata di cui, nel programma, si è calcolata la sua matrice inversa. Per poter eseguire la funzione DET si deve aver prima eseguito un comando OPTIONS specificando l'operando MAT.

Si noti infine che il valore numerico ritornato dalle funzioni numeriche di sistema è sempre rappresentato in doppia precisione.

Nome	Descrizione
ABS(X)	Valore assoluto di X
ACS(X)	Arcocoseno (in radianti) di X
ASN(X)	Arcoseno (in radianti) di X
ATN(X)	Arcotangente (in radianti) di X
COS(X)	Coseno di X radianti
COT(X)	Cotangente di X radianti
DEG(X)	Valore, in gradi, di X radianti
DET	Determinante di una matrice quadrata
EXP(X)	Esponenziale in base e di X
HCS(X)	Coseno iperbolico di X radianti
HSN(X)	Seno iperbolico di X radianti
HTN(X)	Tangente iperbolica di X radianti
INT(X)	Il più grande numero intero minore od uguale ad X
LGT(X)	Logaritmo in base 10 di X
LOG(X)	Logaritmo naturale di X
RAD(X)	Valore, in radianti, di X gradi
RND	Numero casuale compreso tra zero ed uno
SGN(X)	Segno di X (+1 per X positivo, Ø per zero, -1 per X negativo)
SIN(X)	Seno di X radianti
SQR(X)	Radice quadrata di X
TAN(X)	Tangente di X radianti

Tabella 4-3 Funzioni numeriche di sistema

Funzioni stringa di sistema

Una funzione stringa di sistema può essere richiamata in tutte le istruzioni di programma nelle quali può apparire una espressione stringa, ad esempio:

```
50 PRINT CHR$(70)
80 LET A$ = B$ + EXT$(C$,5,8)
```

Nella tabella 4-4 sono riassunte, in ordine alfabetico, le funzioni stringa di sistema disponibili. Le variabili stringa, specificate come argomenti, possono essere, in generale, delle espressioni stringa; le variabili numeriche, specificate come argomenti, possono essere, in generale, delle espressioni numeriche.
Si osservi che le funzioni BLN$ e REP$ possono essere eseguite solamente se prima è stato eseguito un comando OPTIONS specificando l'operando STR.

Nome	Descrizione
BLN$(X,A$,B$)	Permette l'accesso ai bit che costituiscono i caratteri del valore di due espressioni stringa e la loro combinazione mediante un operatore booleano specificato come argomento della funzione. Si veda la completa spiegazione al termine della presente tabella.
CHR$(X)	Converte un numero compreso tra Ø e 255 nel corrispondente carattere del set P6066 (vedi appendice E).
EXT$(A$,I,T)	Fornisce la sottostringa di A$ che inizia dall'I-esimo carattere e termina con T-esimo carattere. Nota: Se I>T, o I≤Ø, o T≤Ø, o T maggiore della lunghezza attuale di A$ il valore di ritorno è la stringa nulla ed il sistema commuta nello stato di debugging segnalando un errore di tipo recuperabile.
LEN(A$)	Fornisce il numero di caratteri che sono contenuti nella variabile A$.
REP$(A$,B$,C$,N,I)	Ritorna il valore della stringa A$ modificato sostituendo la sottostringa B$ con C$, N volte a partire dall'I-esimo carattere.

Note: 1) Se N=Ø non viene effettuata alcuna modifica in A$.

2) Se N<Ø in A$ sono sostituite tutte le occorrenze del valore di B$ con il valore C$ a partire dall'I-esimo carattere.

3) Se N>Ø ed il valore di B$ è la stringa nulla, in A$ viene inserito il valore di C$, N volte a partire dall'I-esimo carattere.

4) Se N<Ø ed il valore di B$ è la stringa nulla, A$ non viene modificata ed il sistema commuta nello stato di debugging.

5) Se il valore di C$ è la stringa nulla ed il valore di N<Ø, allora in A$ sono cancellate tutte le ricorrenze del valore di B$ a partire dall'I-esimo carattere.

6) Se la stringa di caratteri ritornata dalla funzione ha un numero di caratteri superiore alla lunghezza di allocazione di A$, allora il sistema commuta nello stato di debugging troncando la stringa dei caratteri eccedenti la lunghezza suddetta e segnalando un errore recuperabile. Se il primo argomento è una espressione stringa e la stringa di caratteri ritornata dalla funzione ha più caratteri del valore dell'espressione suddetta, il sistema commuta nello stato di debugging troncando la stringa dei caratteri eccedenti e segnalando un errore recuperabile.

7) Se I è maggiore della lunghezza attuale di A$, il valore ritornato dalla funzione è il valore di A$. Nel caso in cui il primo argomento della funzione sia una espressione stringa ed I sia maggiore del numero di caratteri che ne compongono il valore, il valore ritornato dalla funzione è il valore del primo argomento.

SCN(A$,B$,Y,X,)

Se Y>Ø e X>Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di B$, a partire dal carattere nella posizione indicata da X. La scansione in A$ è effettuata muovendosi di un carattere alla volta. (Se in A$, a partire dalla X-esima posizione, non è presente il valore di B$ viene ritornato il valore zero).

Se Y<Ø e X>Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di B$, a partire dal carattere nella posizione indicata da X. La scansione in A$ è effettuata muovendosi con passi uguali alla

	lunghezza attuale di B$. (Se durante la scansione in A$, a partire dalla X-esima posizione, non è trovato il valore di B$ viene ritornato il valore zero). Se Y>Ø e X<Ø la funzione SCN fornisce la posizione in A$ della Y-esima ricorrenza di una stringa diversa del valore di B$, a partire dal carattere nella posizione indicata dal valore assoluto di X. La scansione in A$ è effettuata muovendosi di un carattere alla volta. (Se in A$, a partire dalla X-esima posizione, non è presente alcuna stringa diversa dal valore di B$, viene ritornato il valore zero). Se Y<Ø e X<Ø la funzione SCN fornisce la posizione della Y-esima ricorrenza di una stringa diversa dal valore di B$, a partire dal carattere nella posizione indicata dal valore assoluto di X. La scansione in A$ è effettuata muovendosi con passi uguali alla lunghezza attuale di B$. (Se,durante la scansione in A$, a partire dalla X-esima posizione, non è trovata alcuna stringa diversa dal valore di B$, viene ritornato il valore zero). Nota: Per qualsiasi funzione SCN,se il valore assoluto di X è maggiore della lunghezza attuale di A$,viene ritornato il valore zero.

Tabella 4-4 Funzioni stringa di sistema

Nel seguito diamo una spiegazione completa della funzione BLN$ ed alcuni esempi d'impiego delle funzioni stringa di sistema definite nella tabella 4-4; le note riferite agli esempi ne ampliano la descrizione.

Funzione BLN$: Il formato generale della funzione è:

BLN$ (num-exp, string-exp$_1$, string-exp$_2$)

dove:

num-exp

è una espressione numerica il cui valore, arrotondato all'intero più prossimo, specifica l'operatore booleano con cui sono combinati i bit corrispondenti ai caratteri del valore delle due espressioni stringa

string-exp$_1$ e string-exp$_2$

sono due espressioni stringa.

Il valore ritornato dalla funzione è la stringa di caratteri che si ottiene combinando i bit corrispondenti ai caratteri dei valori delle due espressioni stringa secondo le regole algebriche dell'operatore booleano specificato con il valore, arrotondato all'intero più prossimo, di num-exp. Diamo qui di seguito il significato dei diversi operatori booleani che si possono specificare.

Valore di num-exp	Valore ritornato
0	Stringa con caratteri corrispondenti ad 8 bit uguali a zero
1	"string$_1$" AND "string$_2$"
2	"string$_1$" AND (NOT "string$_2$")
3	"string$_1$"
4	(NOT "string$_1$") AND "string$_2$"
5	"string$_2$"
6	"string$_1$" XOR "string$_2$"
7	"string$_1$" OR "string$_2$"
8	"string$_1$" NOR "string$_2$"
9	NOT ("string$_1$" XOR "string$_2$")
10	NOT "string$_2$"
11	"string$_1$" OR (NOT "string$_2$")
12	NOT "string$_1$"
13	(NOT "string$_1$") OR "string$_2$"
14	("string$_1$" NAND "string$_2$")
15	stringa con caratteri corrispondenti ad 8 bit a 1.

Le virgolette indicano che l'operazione è eseguita considerando i valori delle due stringhe string-exp$_1$ e string-exp$_2$.

Se il valore di num-exp, arrotondato all'intero più prossimo, è minore di zero o maggiore di 15 allora la funzione ritorna una stringa nulla.

Se il valore di num-exp è zero o 15 la stringa ritornata ha la lunghezza della prima.

Se le lunghezze di string$_1$ e string$_2$ sono diverse alla stringa con meno caratteri sono aggiunti caratteri corrispondenti ad 8 bit a zero, in modo da eguagliare la lunghezza delle due stringhe, prima che l'operazione booleana specificata sia eseguita.

Diamo nel seguito le tabelle di verità degli operatori booleani suddetti; all'interno delle caselle sono riportati i valori ottenuti mentre all'esterno sono indicati i valori dei bit su cui si applica il relativo operatore booleano. Nell'appendice E è riportata la corrispondenza tra i caratteri del set ISO P6066 e la relativa rappresentazione su 8 bit.

Operatore AND :

	0	1
0	0	0
1	0	1

Operatore OR :

	0	1
0	0	1
1	1	1

Operatore NOT :

0	1
1	0

Operatore XOR :

	0	1
0	0	1
1	1	0

Operatore NOR :

	0	1
0	1	0
1	0	0

Operatore NAND :

	0	1
0	1	1
1	1	0

Funzione CHR$: Diamo un esempio d'impiego della funzione CHR$ in una routine che permette di stampare tutti i caratteri del set P6066.

```
10 FOR I=0 TO 255
20 PRINT I,CHR$(I)
30 NEXT I
40 END
```

Eseguendola si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
 0                Ç
 1                ü
 2                é
 3                â
 4                ä
 5                à
 6                å
 7                ç
 8                ê
 9                ë
 10               è
 11               ï
 12               î
 13               ì
 14               Ä
 15               Å
 16               É
 17               æ
 18               Æ
```

e così via fino al 256-esimo carattere. Per brevità si è riportata solo la parte iniziale della stampa prodotta.

Funzione EXT$: Diamo un esempio d'impiego della funzione EXT$ in una routine che stampa una parte predefinita di una variabile stringa.

```
NEW
10 DCL 21(B$)
20 B$="LUNGHEZZA AREA VOLUME"
30 PRINT TAB(35);EXT$(B$,11,14)
40 END
```

Eseguendo la routine si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
                                  AREA
```

Funzione LEN: Diamo un esempio d'impiego della funzione LEN in una routine che permette di determinare se effettivamente il numero di caratteri di una variabile stringa può essere 1023.

```
NEW
AUT
10 DCL 1023(B$)
20 A$="*Olivetti*"
30 B$=""
40 FOR I=1 TO 102
50 B$=B$+A$
60 NEXT I
70 B$=B$+"END"
80 PRINT "B$ ha";LEN(B$);"caratteri"
90 PRINT B$
100 END
```

Eseguendo la routine si ha:

```
RUN
**** FORMALLY CORRECT PROGRAM ****
B$ ha 1023 caratteri
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*
*Olivetti**Olivetti**Olivetti**Olivetti**Olivetti**Olivetti*END
```

Funzione REP$: Diamo un esempio di come si può sostituire, utilizzando la funzione REP$, una sottostringa di una stringa.

```
LIST
FILE     REPVIG

0010 DCL 40(A$)
0020 LET A$="AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH"
0025 PRINT A$+" e` A$"
0027 PRINT
0030 PRINT REP$(A$,"HHH","ZZZ",2,7)
0040 PRINT REP$(A$,"HHH","ZZZ",0,7)
0050 PRINT REP$(A$,"HHH","ZZZ",-1,7)
0060 PRINT REP$(A$,"HHH","ZZZ",-100,7)
0070 PRINT REP$(A$,"","ZZZ",3,7)
0080 PRINT REP$(A$,"","ZZZ",-1,7)
0090 PRINT REP$(A$,"HHH","",-1,7)
0100 END

END OF LISTING
```

Eseguendo la routine si ha:

```
RUN
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH e` A$

AAAHHHBBBZZZCCCZZZDDDHHHEEEHHH
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH
AAAHHHBBBZZZCCCZZZDDDZZZEEEZZZ
AAAHHHBBBZZZCCCZZZDDDZZZEEEZZZ
AAAHHHZZZZZZZZZBBBHHHCCCHHHDDDHHHEEEHHH
AAAHHHBBBHHHCCCHHHDDDHHHEEEHHH
ERROR 2   IN LINE 80
AWAHHHBBBCCCDDDEEE
```

Come si vede, l'istruzione 30 stampa il contenuto di A$ modificato sostituendo HHH con ZZZ per due volte a partire dal settimo carattere.

Nell'istruzione 40, il numero di volte con cui HHH deve essere sostituito con ZZZ è zero per cui, come si vede, il contenuto di A$ rimane invariato.

Nell'istruzione 50, il numero di volte con cui HHH deve essere sostituito con ZZZ è indicato con un numero negativo, per cui, a partire dal settimo carattere, tutte le ricorrenze di HHH vengono sostituite con ZZZ (vedi la stampa relativa).

Questa è una regola che è valida per qualunque numero negativo, infatti, nella istruzione 60 il numero -1 è stato sostituito con -100 ed il risultato ottenuto è identico al precedente, come si può vedere dalla relativa stampa.

L'istruzione 70 mette in evidenza che avendo indicato come stringa da sostituire la stringa nulla (""), a partire dal settimo carattere, viene inserita per tre volte la stringa ZZZ in successione.

Nell'istruzione 80, avendo specificato come stringa da sostituire la stringa nulla e come numero di ricorrenza di essa da sostituire un numero negativo, la stringa contenuta in A$ non viene modificata ed il sistema commuta nello stato di debugging. L'operatore può scegliere se continuare nell'esecuzione del programma, premendo il tasto di console CONTINUE, o terminare l'esecuzione e commutare il sistema nello stato comandi, premendo il tasto di console BREAK.

Nella istruzione 90 è stata utilizzata la funzione REP$ per cancellare tutte le ricorrenze della sottostringa HHH nella stringa contenuta in A$ a partire dal settimo carattere. Questo perchè come terzo argomento si è specificata la stringa nulla e come quarto un numero negativo.

Funzione SCN: Vediamo un esempio d'impiego della funzione SCN per rintracciare la presenza di un carattere in una stringa.

```
LIST
FILE     SCN


0010 DCL 80(A$,B$),150(M$)
0020 DISP "INTRODUCI TESTO PER A$ SENZA PUNTO";
0030 INPUT A$
0040 DISP "INTRODUCI TESTO CON PUNTO PER B$";
0050 INPUT B$
0060 LET M$=A$+B$
0070 PRINT "Nel testo vi sono ";SCN(M$,".",1,1);"caratteri."
0080 IF SCN(M$,"!",1,1)=0 THEN 120
0090 DISP "INTRODUCI TESTO CHIUSO CON PUNTO";
0100 INPUT M$
0110 IF SCN(M$," ",1,70)+1=1 THEN 140
0120 PRINT "Il carattere ! non e' nel testo."
0130 GOTO 90
0140 PRINT "Il testo ha meno di 70 caratteri."
0150 END

END OF LISTING
```

Eseguendo la routine si ha:

```
RUN
INTRODUCI TESTO PER A$ SENZA PUNTO?
L'ITALIA E' UNA PENISOLA
INTRODUCI TESTO CON PUNTO PER B$?
NEL SUD DELL'EUROPA.
Nel testo vi sono  44 caratteri.
Il carattere ! non e' nel testo.
INTRODUCI TESTO CHIUSO CON PUNTO?
L'AMERICA E' UN CONTINENTE IMMENSO.
Il testo ha meno di 70 caratteri.
```

Come si vede dall'istruzione 80, se il carattere o la stringa ricercata non esiste nella stringa analizzata, viene fornito come risultato lo zero. Il risultato zero viene dato anche se, vedi istruzione 110, la posizione di inizio della ricerca è superiore al numero di caratteri della stringa.

Vediamo altri esempi della funzione SCN per diversi valori degli argomenti. Se si ha A$ = "AAAAAABAAAAA"

la funzione SCN (A$,"B",1,1) ritorna il valore 7

la funzione SCN (A$,"AA"",2,1) ritorna il valore 2

la funzione SCN (A$,"AA",-2,1) ritorna il valore 3

la funzione SCN (A$,"A",1,-1) ritorna il valore 7

la funzione SCN (A$,"AAAA",1,1) ritorna il valore 4

la funzione SCN (A$,"AAAA",-1,-1) ritorna il valore 5

la funzione SCN (A$,"AB",1,1) ritorna il valore 6

infine la funzione SCN (A$,"AB",-1,1) ritorna il valore Ø.

Funzione speciale di sistema

Il linguaggio BASIC P6066 permette l'impiego, nelle istruzioni DISP e PRINT (vedi capitolo 5), di una funzione speciale di sistema che è richiamata, nelle suddette istruzioni, con il formato:

TAB (num-exp)

La funzione TAB posiziona il pointer del buffer di display od il pointer del buffer di stampa nella posizione il cui valore corrisponde al valore di num-exp arrotondato all'intero più prossimo. Si veda per ulteriori dettagli la relativa descrizione delle istruzioni DISP e PRINT nel capitolo 5.

Espressioni

Una espressione è qualsiasi rappresentazione di un numero o di una stringa. Quindi le costanti, le variabili semplici, le variabili con indice, le funzioni di sistema e le funzioni di utente sono espressioni.

Sono inoltre espressioni le rappresentazioni che si ottengono combinando qualunque degli elementi suddetti (operandi) con particolari simboli detti operatori. Un operatore può specificare:

- una relazione tra gli operandi
- una operazione da eseguire sugli operandi
- se gli operandi sono numeri positivi o negativi

Per esempio i simboli =, * e - sono operatori che specificano, rispettivamente, la relazione uguale, l'operazione di moltiplicazione ed il segno meno. (= e - possono specificare anche, rispettivamente, l'operazione di assegnazione e l'operazione di sottrazione). Una particolare categoria di espressioni, dette espressioni di confronto, è utilizzata nell'istruzione IF per scegliere quale, tra diverse routine, eseguire come conseguenza di un confronto tra dati del programma.

Espressioni numeriche

Una espressione numerica può essere una costante numerica, una variabile semplice numerica, una variabile con indice numerica, un riferimento ad una funzione numerica di sistema od un riferimento ad una funzione numerica definita dall'utente; ma può essere anche una qualunque combinazione degli elementi suddetti (operandi) separati da operatori numerici e parentesi e preceduti dal segno algebrico. Alcuni esempi di espressioni numeriche sono:

```
A
B (5,4)
-25.4
SQR(X)
FNA(Y,Z)
(FNA(Y,Z) * SQR(X) + B (5,4) / (25.4)) - A↑3
```

Il valore di una espressione numerica è ottenuto eseguendo le operazioni specificate, sugli operandi indicati, secondo le regole descritte nel paragrafo successivo.

Note:

1. Se durante l'esecuzione di una espressione numerica si ottiene un risultato intermedio il cui valore è nella zona di OVERFLOW (definita per la rappresentazione in doppia precisione), il sistema continua a valutare l'espressione assumendo come valore intermedio $-9.999999999999*10^{99}$ oppure $9.999999999999*10^{99}$. Il valore finale dell'espressione è assegnato alla variabile specificata nella corrispondente istruzione di assegnazione LET, ma il sistema commuta nello stato di debugging e visualizza un messaggio di errore recuperabile.

2. Se durante l'esecuzione di una espressione numerica si ottiene un risultato intermedio il cui valore è nella zona di UNDERFLOW (definita per la rappresentazione in doppia precisione), il sistema continua a valutare l'espressione assumendo come valore intermedio zero. Il valore finale dell'espressione è assegnato alla variabile della corrispondente istruzione di assegnazione (LET) ma il sistema commuta nello stato di debugging e visualizza un messaggio di errore recuperabile.

3. Se l'indice (o gli indici) di una variabile con in-

dice è dato in forma di espressione numerica, l'espressione viene calcolata e il risultato è arrotondato all'intero più prossimo. Quindi se x è il valore dell'espressione e n la sua parte intera l'indice assumerà il valore:

n	se	n.0	≤	x	≤	n.5
n+1	se	n.5	≤	x	≤	(n+1).0

cioè	se	x=5.49	allora n=5
	se	x=5.5	allora n=6

4. L'impiego contemporaneo, in una espressione numerica, di quantità molto piccole e molto grandi (in valore assoluto) può produrre dei risultati inattesi, dovuti al tipo di rappresentazione interna dei dati; così:

```
10 PRINT  2↑63-1-2↑63+1
20 END
RUN
1
```

il risultato ottenuto è dovuto al fatto che nella espressione contenuta nell'istruzione 10 l'esecuzione della sottrazione 2↑63-1 non modifica 2↑63. Cambiando la sequenza si ha:

```
10 PRINT  2↑63-2↑63+1-1
20 END
RUN
0
```

quindi il risultato esatto.

Operatori numerici

Gli operatori numerici definiscono quale operazione deve essere eseguita sui valori numerici degli operandi specificati. Essi producono come risultato un numero. Gli operatori numerici che si possono utilizzare sono:

Operatore numerico	Funzione
↑	Elevamento a potenza
/	Divisione
*	Moltiplicazione
+	Addizione e segno più
-	Sottrazione e segno meno

Le espressioni numeriche sono eseguite secondo il livello di priorità degli operatori che le costituiscono. Le operazioni con il più alto livello di priorità sono eseguite per prime; quelle con lo stesso livello di priorità sono eseguite nell'ordine da sinistra.

Il linguaggio BASIC osserva le regole dell'algebra per la definizione del livello di priorità di esecuzione di una operazione nell'ambito di una espressione numerica, per cui i livelli di priorità sono:

Operatore numerico	Livello di priorità
↑	il più alto
* e /	↓
+ e -	il più basso

Le parentesi (e) possono essere usate per cambiare l'ordine di esecuzione delle operazioni nell'ambito di una espressione numerica. Una espressione racchiusa tra parentesi è trattata come un singolo elemento numerico: viene valutata per ottenere il suo valore numerico, quindi tale valore è utilizzato per la valutazione della parte restante di una espressione più complessa di cui essa fa parte. Se più di una espressione è compresa tra parentesi, il calcolo inizia con la valutazione delle parentesi più interne. Nel seguito diamo ulteriori informazioni relative agli operatori numerici.

Elevamento a potenza:

B↑E	indica che il valore di B deve essere elevato all'esponente E
A↑B↑C	viene eseguita come se fosse (A↑B)↑C
Se B=Ø ed E<Ø	si ha una segnalazione di OVERFLOW
Se B<Ø ed E non è intero	si ha una segnalazione di errore recuperabile
Se B=Ø ed E=Ø	B↑E è uguale ad uno
Se B=Ø ed E>Ø	B↑E è uguale a zero

Per calcolare la radice N-esima di un numero positivo si deve eseguire l'espressione X↑(1/N). Se X è negativo viene segnalato un errore recuperabile ed il si-

stema commuta nello stato di debugging.

Moltiplicazione ed addizione:

A*B	equivale a B*A
A+B	equivale a B+A

come si vede, per la moltiplicazione e l'addizione vale la proprietà commutativa.

A*(B*C)	non equivale sempre a (A*B)*C
A+(B+C)	non equivale sempre a (A+B)+C

perchè l'operazione tra parentesi in alcuni casi può dare un risultato arrotondato o troncato.

Divisione e sottrazione:

A/B	indica A diviso per B
se B=Ø	si ha segnalazione di OVERFLOW
A-B	indica A-B

Segno:

-B+(-A)+C-(-Z)	è ammesso
A+-B	non è ammesso

Il segno + ed il segno - possono essere usati dopo una parentesi aperta prima di una espressione numerica. Si noti che avendo l'operatore ↑ un più alto livello di priorità rispetto al segno le due espressioni seguenti non hanno lo stesso significato: -5 ↑ 3.2 (-5) ↑ 3.2

Quando è eseguita la prima espressione il risultato ottenuto è -172.46621, mentre quando è eseguita la seconda espressione il risultato ottenuto è 172.46621 ed il sistema è nello stato di debugging mentre sul display appare un messaggio di errore recuperabile.

Espressioni stringa e operatori

Una espressione stringa è costituita da qualsiasi costante stringa, variabile semplice stringa, variabile con indice stringa, funzione stringa di sistema o definita dall'utente; oppure può essere una qualsiasi sequenza degli elementi suddetti (detti operatori) separati da parentesi e da un operatore binario.

L'unico operatore binario è il simbolo di concatenazione + che ad esempio in D$=A$+B$ produce come risultato, in D$, una stringa che è composta dai caratteri di A$ e quelli di B$ aggiunti in sequenza. Quindi la lunghezza della stringa risultante da concatenazione di una o più stringhe è uguale alla somma della lunghezza delle singole stringhe. Esempi di espressioni stringa sono:

```
A$
"Area di base"
REP$ (A$, B$, C$, 3, 1)
A$+"Area di base"+REP$(A$,B$, C$,3,1)
```

Espressioni di confronto

Una espressione di confronto paragona il valore di due espressioni numeriche o stringa. Il risultato del confronto è un valore di verità (vero o falso). Gli operatori di confronto sono:

Operatore	Significato
=	Uguale
<> oppure ><	Non uguale
>= oppure =>	Maggiore o uguale di
<= oppure =<	Minore o uguale di
>	Maggiore di
<	Minore di

Il formato generale della espressione di confronto è:
e1 operatore-confronto e2

e1 ed e2 possono essere qualunque espressione meno che una espressione di confronto. Solo due espressioni si possono confrontare in una espressione di confronto. Le espressioni da confrontare debbono essere entrambe numeriche od entrambe stringa. Nelle espressioni di confronto tra stringhe il paragone avviene carattere per carattere da sinistra a destra; per giudicare se un carattere è maggiore di un altro, si deve osservare la appendice E in cui sono riportati tutti i caratteri del SET P6066 ed i relativi valori decimali; fra due caratteri è maggiore quello a cui corrisponde, nella suddetta tabella, un numero decimale maggiore. Così si ha che le seguenti espressioni di confronto sono vere:

```
  "A" < "B"
"ABC" < "ABD"
"ABC" < "CAB"
```

Quando si confrontano due stringhe con lunghezza diversa il confronto avviene tra la stringa più corta e la parte di sinistra della stringa più lunga. Se il confronto dà risultato di eguaglianza viene considerata maggiore la stringa più lunga. Così si ha che le seguenti espressioni di confronto sono vere:

```
ABC < ABCD
HAZ < HAZL
ZAB > ABCDE
```

Espressioni con variabili multiple numeriche

Una espressione può contenere delle variabili multiple numeriche (matrici); in questo caso l'espressione suddetta appare alla destra del segno uguale. Alcune istruzioni in cui compaiono delle intere matrici allo interno di espressioni sono:

```
10 MAT A = B
20 MAT B = C + D
30 MAT Z = C - D
40 MAT A = (5)*B
```

Per ulteriori informazioni si vedano le istruzioni MAT nel capitolo 5.

File dati

Con il linguaggio BASIC si possono generare ed elaborare insiemi di dati di tipo numerico e/o stringa che prendono il nome di file dati. I file dati sono distinti in file dati interni e file dati esterni.

Si definiscono file dati interni quegli insiemi di dati numerici e/o stringa che sono interni ad un programma e quindi tutti presenti in memoria principale contemporaneamente al programma che li elabora.

Si definiscono file dati esterni quegli insiemi di dati numerici e/o stringa che sono registrati in una libreria e sono richiamati in memoria principale nelle parti che devono essere elaborate. Essi possono essere quindi utilizzati da diversi programmi e possono contenere più dati dei precedenti,poichè la loro dimensione non dipende dalla capacità della memoria principale.

Nei paragrafi successivi vengono analizzati separatamente i due tipi di file suddetti.

File dati interni

I file dati interni sono costituiti da sequenze di dati numerici e/o stringa definite in un programma BASIC. Per definire un file dati interno si utilizza la istruzione DATA. L'istruzione:

```
10 DATA 15, "AREA", 17, "VOLUME"
```

definisce un file dati interno i cui elementi sono in sequenza: il numero 15, la stringa AREA, il numero 17 e la stringa VOLUME. Si possono utilizzare più istruzioni DATA in un programma; in questo caso il file dati corrispondente è costituito da tutti i valori numerici e stringa espressi nelle istruzioni DATA ed ordinati nella sequenza con sui sono presenti, da sinistra a destra, nelle istruzioni stesse. Quindi le istruzioni:

```
10 DATA  1,2,3,4
20 DATA  5,6,7,8
30 DATA  9,A,B,C
40 DATA  D,E,F,G,H,I,L,M,N,O,P,Q
50 DATA  R,S,T,U,V,W,X,Y,Z
```

definiscono un file dati interno ad un programma che inizia con il dato numerico 1 e prosegue poi in sequenza con le cifre da 2 a 9 e quindi con le lettere dell'alfabeto da A a Z. Quindi A sarà il decimo elemento del suddetto file.

Per accedere ai dati del file così definito si utilizza l'istruzione READ, che permette di assegnare alle variabili di programma i valori contenuti nel file suddetto. Nell'esempio seguente:

```
10 DATA  1,2,3,4
20 DATA  5,A,B,C
30 READ  A,B,C
40 READ  D
50 READ  E,A$,B$,C$
```

dopo aver definito il file dati i cui elementi sono in sequenza 1,2,3,4,5,A,B e C, con le istruzioni 30,40 e 50, si assegnano tali valori nell'ordine alle variabili: A,B,C,D,E,A$,B$ e C$.

L'assegnazione dei dati alle variabili è fatta in modo

dinamico ossia, mentre l'ordine con cui si succedono i dati nel file è determinato dall'ordine con cui i dati compaiono nelle istruzioni DATA (da sinistra a destra) e dall'ordine con cui le istruzioni DATA compaiono (secondo il numero di linea) nel programma, l'ordine di assegnazione coincide con l'ordine con cui sono eseguite le istruzioni READ e nell'ambito dell'istruzione READ l'assegnazione procede dalla variabile più a sinistra verso destra. Il file così prodotto è un file sequenziale; infatti i dati sono assegnati alle variabili uno dopo l'altro.

Si può riprendere l'assegnazione dei dati alle variabili dall'inizio del file utilizzando l'istruzione RESTORE. In un programma si può avere un solo file dati interno.

Per ulteriori informazioni sui file dati interni si vedano nel capitolo 5 le istruzioni: DATA, READ e RESTORE.

File dati esterni

I file dati esterni sono insiemi di dati numerici e/o stringa che sono registrati in una libreria.
Questo permette l'impiego di archivi di dati che non sono limitati dalla capacità della memoria principale.
Riferendosi al metodo di elaborazione i file dati esterni si distinguono in:

- file ad accesso sequenziale
- file ad accesso diretto

Un file è ad accesso sequenziale se per accedere ad un suo dato si devono prima leggere tutti i dati che lo precedono sul supporto su cui il file è registrato.

Un file è ad accesso diretto se si può accedere direttamente ad un suo qualsiasi dato senza dover leggere i dati che lo precedono.

Un file dati esterno non può essere generato da tastiera, ma deve essere creato eseguendo un comando CREATE (vedi capitolo 3) ed un programma BASIC. Il comando CREATE alloca lo spazio per il file; il programma genera i dati che saranno registrati nel file. Per ottenere informazioni relative ad un file dati esterno, quale il modo in cui può essere elaborato, lo spazio che occupa etc. si deve eseguire il comando CATALOG.

Creazione di un file dati esterno: Per generare un file dati esterno si deve prima allocare per esso un certo spazio in una libreria utilizzando il comando di sistema CREATE (vedi capitolo 3). Lo spazio da allocare deve essere uguale al numero di byte che si prevede di occupare con i dati del file arrotondato ad un multiplo intero di 128, se il file è creato in una libreria residente su floppy disk, oppure 256 se il file è creato in una libreria residente su disco. Il sistema si riserva in più 128 byte o 256 byte (a seconda se floppy disk o disco) per registrare delle informazioni di servizio. Oltre al numero di byte da riservare al file dati, con il comando CREATE si comunica al sistema il nome del file dati, il tipo di sottolibreria (è implicito nel nome) e la libreria in cui deve essere allocato. I dati sono registrati in un file esterno con un formato diverso da quello con cui sono rappresentati in memoria principale per essere elaborati. Nell'appendice F si possono confrontare le diverse richieste di spazio nei due casi. Per calcolare lo spazio richiesto dal contenuto di un file dati esterno sul suo supporto, si devono tener presenti le seguenti considerazioni:

1. I dati numerici in singola precisione occupano 4 byte.

2. I dati numerici in doppia precisione occupano 8 byte.

3. Le costanti numeriche sono sempre registrate, nel file esterno, in doppia precisione, qualunque sia il loro valore.

4. I dati di tipo stringa occupano 4*INT ((n-1)/4+2) byte, dove INT è la funzione che fornisce la parte intera del valore calcolato tra parentesi ed n è il numero di caratteri che compone la stringa.

Il nome da assegnare ad un file dati può essere costituito da al massimo 7 caratteri, se il file dati deve essere registrato in una sottolibreria package od in una sottolibreria comune; da 6 caratteri, se il file dati deve essere registrato in una sottolibreria utente.

Il primo carattere deve essere asterisco (*) nel caso di un file dati da registrare in una sottolibreria

package, mentre deve essere più (+) nel caso di un file dati da registrare in una sottolibreria comune; in entrambi i casi il secondo carattere deve essere una lettera maiuscola dell'alfabeto inglese (da A a I) ed i restanti caratteri possono essere costituiti da una lettera maiuscola, come il precedente, oppure da una cifra decimale (da Ø a 9).

Nel caso di file dati registrati in una sottolibreria utente il primo carattere deve essere una lettera maiuscola dell'alfabeto inglese ed i restanti caratteri possono essere una lettera, come il precedente, oppure una cifra decimale.

Ricerca di un dato in un file dati esterno: Al fine della ricerca di un dato in un file dati esterno si deve tener presente che i file dati esterni sono suddivisi in parole contigue e ad essi è associato un pointer che indica su quale parola del file può avvenire l'accesso; una parola è costituita da 4 byte; sotto questo aspetto i diversi tipi di dati occupano il seguente spazio:

1. I dati numerici in singola precisione sono contenuti in una parola.

2. I dati numerici in doppia precisione sono contenuti in due parole.

3. Le costanti numeriche sono contenute sempre in due parole.

4. I dati di tipo stringa sono contenuti in INT((n-1)/4+2) parole; dove INT è la funzione che fornisce la parte interna del valore calcolato tra parentesi ed n è il numero di caratteri che compone la stringa. L'ultima parola può non essere completamente occupata dal dato; in questo caso è riempita con spazi aggiunti in coda.

Eseguendo l'istruzione WHERE: (vedi capitolo 5) si può sapere su quale parola del file dati esterno è posizionato il pointer.

Apertura e chiusura di un file dati esterno: Un file si dice aperto nei confronti di un programma se il programma può accedere ad esso. Un file si dice chiuso nei confronti di un programma, se il programma non può

accedere ad esso. Perchè un file possa essere utilizzato da un programma deve essere aperto specificandone il nome in una istruzione FILES. Il numero di file dati che possono essere contemporaneamente aperti all'accesso di un programma,in un certo istante, è limitato dal numero di caratteri che costituiscono l'istruzione FILES.

I file dati aperti all'azione di un progamma mediante l'istruzione FILES possono essere chiusi utilizzando l'istruzione FILE:che può contemporaneamente aprirne degli altri.

Per poter leggere i dati contenuti in un file esterno o registrare in esso dei dati si deve utilizzare l'istruzione FILES (vedi capitolo 5) che assegna ad ogni file specificato in essa 172 byte di memoria principale, se il file è in una libreria residente su floppy disk, oppure 292 byte se il file è in una libreria residente su disco. L'ordine con cui i nomi dei file dati esterni si susseguono nell'istruzione FILES è importante perchè ad esso corrisponde un numero designatore per ogni file che, riportato nelle istruzioni di elaborazione dei file, specifica su quale file deve essere eseguita l'operazione indicata dalla relativa parola chiave.

Con l'istruzione FILE: si possono riassegnare i numeri designatori di file dati esterni a file che non sono stati specificati nell'istruzione FILES (i dettagli sono ampliamente descritti nel capitolo 5 alle istruzioni FILES e FILE).

Elaborazione di un file dati esterno: I file dati esterni di tipo sequenziale, dopo le operazioni preliminari suddette, possono essere letti mediante l'istruzione READ:, i dati letti sono assegnati alle variabili specificate nell'istruzione iniziando dal primo dato ed il pointer indica l'inizio della parola successiva all'ultimo dato letto; in questo modo i dati sono letti in sequenza uno dopo l'altro.

Se si vogliono registrare dei dati dall'inizio del file esterno, si utilizza l'istruzione SCRATCH: e quindi con l'istruzione WRITE: si registrano i dati contenuti nelle variabili in essa specificate.

Se invece si vogliono accodare dei nuovi dati a quelli

esistenti si utilizza l'istruzione APPEND:, il pointer indicherà così l'inizio della parola successiva all'ultimo dato del file esterno e con successive istruzioni WRITE: si potranno registrare nuovi dati in sequenza.

Se dopo l'esecuzione di istruzioni WRITE: si vogliono leggere i dati contenuti nel file esterno si deve prima eseguire l'istruzione RESTORE: che ripone il pointer all'inizio del file e quindi le relative istruzioni READ: leggeranno in sequenza i dati in esso contenuti.

I file dati esterni di tipo ad accesso diretto offrono una maggiore elasticità nelle elaborazioni. Infatti dopo le operazioni preliminari, che sono comunque obbligatorie, eseguite dall'istruzione FILES o da una istruzione FILE:, si possono immediatamente leggere dei dati contenuti nel file o registrare in esso dei dati. Inoltre è possibile leggere un dato posizionato dall'inizio di una parola qualsiasi del file esterno purchè si specifichi prima la posizione di tale parola mediante una istruzione SETW: dopo di che si possono leggere con istruzioni READ: i dati che seguono in sequenza sul file esterno il punto specificato. Naturalmente si possono leggere, volendo tutti i dati in sequenza dall'inizio, eseguendo l'istruzione SETW: specificando la prima parola del file, e quindi le relave istruzioni READ:.

Anche la registrazione può avvenire dall'inizio di una parola qualunque del file dati esterno; questa viene specificata, come suddetto, con l'istruzione SETW:, dopo di che si registrano i dati con le istruzioni WRITE:.

Per aggiungere dati alla fine di quelli già presenti in un file dati esterno non si può utilizzare, nel caso di file ad accesso diretto, l'istruzione APPEND:, ma si può specificare tale posizione con l'istruzione SETW:. Naturalmente si possono registrare, volendo, i dati in sequenza dall'inizio del file esterno eseguendo l'istruzione SETW: specificando la prima parola del file (oppure l'istruzione RESTORE:), e quindi le relative istruzioni WRITE:.

Nota: Nel caso di file esterni, i dati da registrare sono memorizzati in un buffer in memoria principale prima di essere trasferiti nel file esterno. I dati sono trasferiti solamente quando (1) il buffer è pieno, (2) è eseguita una istruzione FILE: oppure END, (3) il tasto BREAK è premuto. Quindi se viene a mancare la tensione di alimentazione alla macchina durante l'esecuzione di un programma che registra dati su un file esterno, alcuni dati possono essere perduti ed il file può restare aperto. Si può determinare se un file dati è rimasto aperto eseguendo il comando CATALOG. (Per chiudere tale file si esegua il comando VALIDATE).