# BASIC
## OVVERO COME COMUNICARE CON IL VOSTRO PC 128S

Cominciamo da questo numero la trattazione dei comandi Basic, elencandovi in ordine alfabetico tutte le funzioni disponibili.

Il basic, come sapete, è un linguaggio molto semplice da usare, ed è anche molto facile da imparare!

La scelta di spiegare i comandi in ordine alfabetico è motivata dal fatto che le pagine di questa rivista vogliono costituire, più che una semplice lettura, un manuale, dal quale attingere informazioni preziose. Diamo il via quindi, dopo l'articolo introduttivo comparso sulle pagine del numero 4, alla spiegazione del linguaggio che vi permetterà di colloquiare con il vostro computer.

I comandi sono spiegati seguendo questo schema:

Nome comando: è il nome del comando che viene trattato.

Tipo: dà un'indicazione di massima del campo di applicazione del comando spiegato.

Sintassi: indica l'esatto modo di scrivere il comando, in modo tale che il computer possa comprendere esattamente cosa deve eseguire.

Commento: breve spiegazione delle possibili applicazioni del comando spiegato; solitamente è completato da programmi esemplificativi.

Di ogni comando trattato, tra parentesi, è riportata l'eventuale abbreviazione disponibile.

Rimbocchiamoci quindi le maniche, e iniziamo con la spiegazione dei comandi Basic!!

### ABS

TIPO: Funzione di conversione.

SINTASSI: ABS (espressione).

COMMENTO:

La funzione permette di restituire il valore positivo di qualsiasi espressione numerica. L'espressione in esame può avere valori negativi o positivi.

Provate sul vostro PC 128S questi brevi programmi che vi aiuteranno a comprendere l'uso di questa funzione.

```
10 Numero = – 10
20 PRINT ABS (Numero)
```

sul video comparirà il numero 10, positivo.

```
10 Numero = 10
20 PRINT ABS (Numero)
```

il video mostrerà il valore di Numero, senza che la funzione ABS abbia provocato nulla sul valore stesso, essendo esso già positivo.

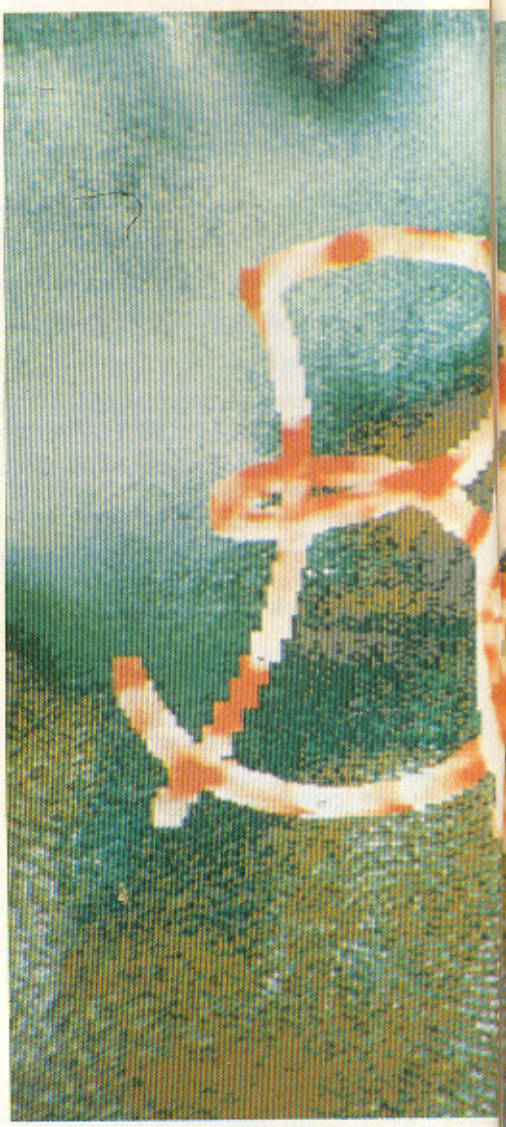
### ACS

TIPO: Funzione matematica.

SINTASSI: ACS (espressione).

COMMENTO:

Questa opzione servirà soprat-



tutto a coloro i quali si occupano di matematica.

Dato un numero, compreso tra –1 e 1 questa funzione restituisce l'arcocoseno, espresso in radianti.

In questa sede non approfondiremo cosa è l'arcocoseno, in quanto è una nozione matematica che sarebbe lungo spiegare in modo esauriente.

Il breve programma riportato di seguito può aiutare a capire l'uso della funzione a chi già ne conosce le implicazioni matematiche.

```
10 angolo = 1
20 PRINT ACS (angolo)
```

## ADVAL (AD.)

TIPO: Funzione di Input/Output.

SINTASSI: ADVAL (numero).

COMMENTO:

Usato per leggere o controllare dati da porte di Input/Output, quali la tastiera, la porta RS423 e altre.

L'uso del comando è di grande utilità solo se le basi di programmazione che l'utente possiede sono già solide, quindi è consigliabile non cominciare proprio da questo comando lo studio del Basic del PC 128S.

Il comando ha tre utilizzi principali spiegati di seguito uno alla volta, secondo il valore che possiede (numero).

1. (numero) = negativo. In questo caso il numero negativo corrisponde a un numero di buffer; deve essere compreso tra 1 e 9.

| (numero) | porta | Input/Output | Caratteristica | |
|---|---|---|---|---|
| – 1 | Tastiera | (lettura) | caratteri usati | 0- 31 |
| – 2 | RS 423 | (lettura) | caratteri usati | 0-255 |
| – 3 | RS 432 | (uscita) | caratteri liberi | 0-191 |
| – 4 | Stampante | (uscita) | caratteri liberi | 0- 63 |
| – 5 | Voce 0 | (uscita) | numero byte liberi | 0- 15 |
| – 6 | Voce 1 | (uscita) | numero byte liberi | 0- 15 (*) |
| – 7 | Voce 2 | (uscita) | numero byte liberi | 0- 15 (*) |
| – 8 | Voce 3 | (uscita) | numero byte liberi | 0- 15 (*) |

Il simbolo (*) significa che ogniqualvolta si vuole scrivere o leggere in questo buffer sono necessari tre byte per formare un dato completo.

2. (numero) = zero.
   In questo caso il comando prende informazioni sullo stato del tasto fire del joystick.

3. (numero) = positivo.
   (numero) in questo caso rappresenta il numero del canale analogico; i valori possibili sono da 1 a 4.
   Il valore nel campo 0 –65520 deve essere incrementato a passi di 16. Al valore letto dal canale, si associa una tensione elettrica, al numero 0 corrisponde una tensione di 0 Volt, mentre il valore di 65520 corrisponde ad una tensione di uscita di 1,8 Volt, approssimativamente.

Questo comando, per poterlo usare nel modo migliore, richiede una conoscenza di base, sia del PC 128S, sia delle tecniche di programmazione proprie del linguaggio macchina.

In particolare, sull'argomento Input/Output, sarebbe possibile scrivere un libro, senza, peraltro, esaminare tutte le problematiche ad esso legate.

Importante comunque è ricordare che questo comando è messo a disposizione dal PC 128S

per controllare strumentazioni esterne, sia ad uso didattico che professionale.

## AND (A.)

TIPO: Operatore aritmetico-logico.

SINTASSI: (espressione) AND (espressione).

COMMENTO:

Il principale uso di questo comando è quello di unire logicamente due eventi; un esempio in questo caso è indispensabile per capire a fondo il comando:

IF ruote = 4 AND motore = 1600 THEN PRINT "auto cc. 1600"
traducendo letteralmente:
se le ruote sono 4 e il motore è 1600 allora scrivi sul video "auto cc. 1600".

L'operatore AND è usato quindi per controllare la contemporaneità di due situazioni, quali possono essere il numero di ruote e la cilindrata di un motore. Questi due eventi permettono di 'capire' che l'automobile ha una cilindrata di 1600 centimetri cubici.

Come potete comprendere, questa funzione ha molti usi, anche più importanti di questo esempio; l'esperienza vi insegnerà più di qualsiasi ulteriore spiegazione.

## ASC

TIPO: Funzione di conversione.

SINTASSI: ASC (stringa).

COMMENTO:

Questa funzione, data una stringa, restituisce il numero abbinato al primo carattere incontrato nella stringa stessa.

Questo abbinamento è fisso, ed è in pratica la tabella ASCII, tabella che potete consultare a pagina 184-185 della 'guida all'uso del sistema PC 128S', il manuale in dotazione con il vostro computer. Ricordiamo che la stringa può contenere sia lettere che numeri, e per applicazioni particolari anche caratteri diversi.

L'esempio di seguito illustrato chiarisce ogni dubbio al riguardo di questa utile funzione.

```
10 saluto$ = "ciao"
20 PRINT ASC (saluto$)
```

il programma mostrerà il numero corrispondente alla lettera c, cioè 67.

Se il numero ritornato dalla funzione è –1, allora la stringa è completamente vuota.

Comunque il numero ritornato da questa funzione rimane sempre compreso nel campo 0 - 255.

## ASN

TIPO: Funzione matematica.

SINTASSI: ASN (espressione).

COMMENTO:

Ricordando il discorso fatto a proposito di funzioni matematiche complesse, riguardo la conoscenza d'uso dal punto di vista matematico, questa funzione permette di ricavare l'arcoseno.

Il campo di valori che (espressione) può accettare, senza segnalare errori, è tra i valori –1 e 1.

Il risultato verrà espresso in radianti.

Il programmino esplicativo è semplice, nonostante la difficoltà intrinseca della funzione matematica.

```
10 angolo = .89
20 PRINT ASN (angolo)
```

## ATN

TIPO: funzione matematica.

SINTASSI: ATN (espressione).

COMMENTO:

Questa funzione matematica ritorna il valore dell'arcotangente, in radianti, dell'(espressione).

L'(espressione) può avere qualsiasi valore numerico. Il classico programmino d'esempio potrà aiutare chi conosce la funzione nel suo significato matematico, ma sarà di scarso ausilio a coloro i quali non possiedono ancora queste basi matematiche.

```
10 angolo = 1.45
20 PRINT ATN (angolo)
```

## AUTO (AU.)

TIPO: Ausilio alla programmazione.

SINTASSI: AUTO (numero prima riga), (incremento).

COMMENTO:

Questo utile comando permette di demandare al computer l'oneroso compito della numerazione delle linee durante la fase di programmazione in Basic. (numero prima riga) deve essere un numero compreso tra 0 e 32767, inoltre dev'essere rigorosamente intero.

(incremento) è l'incremento che il comando darà alle linee.

Ad esempio: se la linea d'inizio ha il valore 1000, e l'incremento è 10, le linee generate saranno del tipo:

1010
1020
1030

e così di seguito.

Per fermare la numerazione automatica è necessario premere il tasto Escape.

Il comando Auto si fermerà automaticamente se il numero di linea supererà il valore 32767.

## BGET# (B.#)

TIPO: Operatori su file.

SINTASSI: BGET# (canale)

COMMENTO:

Prende il prossimo carattere (byte) contenuto nel file: quest'ultimo doveva essere già stato aperto in precedenza.

Il comando che apre i file si chiama OPENIN, ed è bene conoscere anche questo comando per poter operare con tutta tranquillità.

(canale) è il numero dato al file attualmente aperto, dal quale intendiamo leggere i caratteri.

Il valore restituito è un numero intero compreso tra 1 e 255.

Importante è ricordare che per controllare se tutti i caratteri sono stati letti, è possibile verificare se la funzione EOF# è vera; se cioè il file è finito.

Successivamente, se abbiamo già letto l'ultimo carattere, una lettura tornerà il valore 254.

Ogni tentativo di leggere altri caratteri darà come unica conseguenza la segnalazione di un er-

rore di «fine del file».

Per un programma esemplificativo rimandiamo per completezza alla trattazione del comando OPENIN.

## BPUT# (BP.#)

TIPO: Operatori su file.

SINTASSI: BPUT# (canale), (espressione).

COMMENTO:

Il comando permette di scrivere sul file definito in uscita.

(canale) è il numero associato al file nel quale vogliamo scrivere, mentre (espressione) è un valore intero compreso tra 0 e 255, ed è il codice ASCII del carattere inviato al file (vedi la funzione ASC). Il programma d'esempio sarà riportato, come già spiegato nel comando BGET#, quando verrà affrontata la spiegazione dei comandi riguardanti l'apertura dei file; tutto questo per dare una visione completa, e non limitata dell'uso dei comandi basic disponibili sul vostro PC 128S.

## CALL (CA.)

TIPO: Comando.

SINTASSI: CALL (indirizzo), (variabile1), (variabile n).

COMMENTO:

Il comando mette in esecuzione una routine in linguaggio macchina, passando ad essa eventuali valori contenuti nelle variabili.

Il comando richiede una preparazione di base sul linguaggio macchina, per poter capire tutta la sua utilità e tutta la sua potenza.

(VARIABILE N) vuole significare che possono esservi più d'una variabile: quante si rendono necessarie nella logica del programma.

(indirizzo) è un numero compreso da 0 a 65535, in esadecimale 0 - FFFF.

In quel punto preciso la routine inizia, ed è quindi importante chiamare l'indirizzo esatto; una chiamata ad un indirizzo non corretto potrebbe avere effetti disastrosi per il vostro programma.

Vi è una parte di memoria chiamata 'memoria di parametri' dove sono memorizzate le informazioni relative alle variabili da passare al programma in linguaggio macchina. L'indirizzo di questa tabella è, in esadecimale, &600; mentre il formato è il seguente:

&600 Numero di variabili
&601 indirizzo della variabile numero 1
&603 tipo della variabile numero 1
&605 indirizzo della variabile numero 2
&607 tipt della variabile numero 2
&609 indirizzo della variabile numero 3
&60A è così via di seguito...

Il massimo numero di parametri è 85. il minimo zero, è quindi possibile non passare alcun parametro alla routine in linguaggio macchina, se la logica della routine non lo prevede.

I tipi delle variabili sono qui elencati di seguito, verrà dato, inoltre, ampio chiarimento su ogni punto trattato.

| | Tipo | Descrizione | Esempio |
|---|---|---|---|
| 1. | &00 | numero di byte, intero | ?a% |
| 2. | &04 | numero intero di quattro byte | a% |
| 3. | &05 | numero reale di cinque byte | a |
| 4. | &80 | una stringa ad un indirizzo | $a% |
| 5. | &81 | una variabile stringa | a$ |

1. Il valore che può assumere questo tipo di variabile è compreso tra 0 e 255 o tra −128 e +127 se il dato è interpretato come un numero dotato di segno algebrico.

2. In questo caso l'indirizzo punta ad un intero, formato da quattro byte, dove il byte meno significativo si trova nell'indirizzo più basso. Il numero, intero, può variare nel campo − 2147483648 + 2147483647.

3. Il numero reale è rappresentato nell'usuale formato esponente-mantissa. Il numero di byte usato è cinque. Il primo di essi è l'esponente in eccesso, può quindi variare tra −128 e +127; i restanti quattro formano la mantissa. Per i programmatori più esigenti è bene far notare che il segno della mantissa è dato dal bit più significativo del secondo byte.

4. L'indirizzo di una stringa è composto da un byte, varia quindi nel campo 0 - 255. Qualsiasi carattere può far parte di una stringa, escludendo il carattere ASCII 13, il return, in quanto usato dal computer per riconoscere la fine della stringa stessa.

5. L'indirizzo punta alle informazioni della stringa. In particolare l'indirizzo è composto da quattro byte i quali vengono usati per fornire tre informazioni. I primi due byte contengono l'indirizzo della memoria dove è contenuto l'inizio della stringa in esame. Il terzo byte contiene il numero di caratteri che la stringa contiene; è evidente quindi che la lunghezza massima di una stringa sarà di 255 caratteri. Il quarto e ultimo byte fornisce il numero di caratteri che la stringa contiene attualmente.

Al comando CALL possono essere passate anche variabili di tipo matriciale. Una matrice è sempre memorizzata in modo che gli elementi costituenti la matrice stessa siano conseguenti, cioè l'elemento numero 4 sarà sempre dopo l'elemento 3 e sempre prima dell'elemento 5. Questa precisazione viene fatta perché in particolari condizioni, questo tipo di memorizzazione degli elementi di una matrice non viene usato. Ricordate comunque che l'uso della funzione CALL richiede, come già detto in precedenza, approfondita conoscenza del linguaggio macchina del processore 65C12.

## CHAIN (CH.)

TIPO: Comando.

SINTASSI: CHAIN (espressione).

COMMENTO:

Il comando CHAIN permette di concatenare ed eseguire da programma un altro programma.

(espressione) può essere una stringa o direttamente il nome del programma che si intende richiamare.

CHAIN (test)

Carica dalla directory corrente il programma di nome test. Anche questo modo di lavorare, illustrato di seguito, è sintatticamente corretto.

```
a$ = "test1"
CHAIN(a$)
```

## CHR$

TIPO: Funzione di conversione.

SINTASSI: CHR$ (valore).

COMMENTO:

Questa funzione ritorna il carattere corrispondente al (valore).

(valore) deve essere compreso tra 0 - 255, ma i caratteri visibili sul video partono dal valore 32, il quale corrisponde allo spazio.

Ad esempio il numero 65 corrisponde alla lettera 'a', il numero 66 alla lettera 'b', e così di seguito.

Per capire meglio provate ora questo breve programma.

```
10 FOR I = 32 TO 255
20 PRINT CHR$(I)
30 NEXT I
```

sul video appariranno di seguito tutti i caratteri disponibili nel set ASCII.

## CLEAR (CL.)

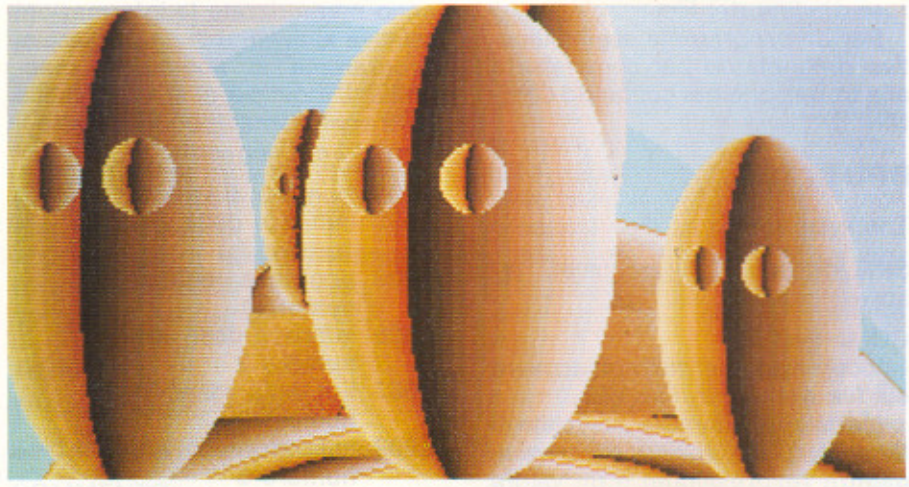
TIPO: Comando.

SINTASSI: CLEAR

COMMENTO:

Il comando CLEAR cancella tutti i valori delle variabili contenuti in memoria al momento del comando.

L'unica eccezione: il tipo di variabile chiamato 'residente' non verrà interessato da questo comando. Una volta che il comando è stato dato, non esiste in basic un comando che svolga la funzione di ripristinare il valore originario delle variabili; attenzione quindi a non cancellare involontariamente tutti i vostri dati numerici dalla memoria!



## CLG

TIPO: Comando.

SINTASSI: CLG.

COMMENTO:

Il comando cancella l'attuale finestra grafica mostrando il video interamente riempito del colore di fondo scelto.

Questo comando è l'unico che permette di cancellare un disegno. Per cancellare il video selezionato nel modo scrittura caratteri, il comando da usare è di CLS, riportato e spiegato più avanti.

## CLOSE# (CLO.#)

TIPO: Operatori su file.

SINTASSI: CLOSE# (canale).

COMMENTO:

Il comando CLOSE# permette di chiudere un file aperto. La scelta di quale file chiudere è selezionabile dando il corretto valore a (canale). (canale) è un numero che viene associato ad un particolare file in fase di apertura.

Esempio dell'uso di questo comando è fatto nella trattazione del comando OPENIN, questo per dare una visione completa dell'uso dei file nei vostri programmi.

## CLS

TIPO: Comando.

SINTASSI: CLS

COMMENTO:

Questo comando cancella tutto il video nel modo caratteri, il modo disponibile all'accesione della macchina.

È bene usarlo sempre prima di ogni programma basic. Provate i due programmi riportati di seguito che vi faranno capire l'utilità di questo semplice comando.

```
10 PRINT "comincia dalla
         posizione attuale"
20 PRINT "del cursore, ma dove
         è il"
30 PRINT "cursore in questo
         momento?"
```

la scritta comincia da dove il cursore era già posizionato; le parole scritte in precedenza sul video vengono solo spostate verso l'alto, ma non cancellate.

```
10 CLS
20 PRINT "comincia dalla
         posizione attuale"
30 PRINT "del cursore, ma dove
         è il"
40 PRINT "cursore in questo
         momento?"
```

in questo caso la scritta verrà visualizzata sul video a cominciare dall'alto, e i caratteri presenti prima, come potete vedere se provate il programma sul computer, sono stati cancellati.

## COLOR o COLOUR (C.)

TIPO: Comando.

SINTASSI: COLOR (espressione)

| Colore | Modo 2 | Modo 4 | Modo 16 |
|---|---|---|---|
| 0 | nero | nero | nero |
| 1 | bianco | rosso | rosso |
| 2 | nero | giallo | verde |
| 3 | bianco | bianco | giallo |
| 4 | nero | nero | blu |
| 5 | bianco | rosso | magenta |
| 6 | nero | giallo | celeste |
| 7 | bianco | bianco | bianco |
| 8 | nero | nero | nero/bianco |
| 9 | bianco | rosso | rosso/celeste |
| 10 | nero | giallo | verde/magenta |
| 11 | bianco | bianco | giallo/blu |
| 12 | nero | nero | blu/giallo |
| 13 | bianco | rosso | magenta/verde |
| 14 | nero | giallo | celeste/rosso |
| 15 | bianco | bianco | bianco/nero |

COMMENTO:

Il comando spiegato ora è necessario se il programmatore desidera usare particolari colori per evidenziare messaggi sul video.

(espressione) può variare tra 1 e 255.

I numeri tra 1 e 15 settano il colore dei caratteri scritti sul video; aggiungendo il numero 128 si setterà il colore di fondo del testo.

Il numero di colori disponibile è in funzione del tipo di modo usato.

La tabella qui sotto servirà a chiarire qualsiasi dubbio; inoltre sarà un utile riempimento a tutti coloro che fanno spesso uso del colore nei programmi.

Nel modo a sedici colori, i colori dall'ottavo al quindicesimo compreso sono lampeggianti.

Ad esempio, settando il colore numero 14, il testo verrà scritto in celeste e, lampeggiando, diverrà alternativamente rosso-celeste.

Per cambiare il modo di visualizzazione dei colori fare riferimento al comando MODE.

Provate questo programma che vi scriverà lettere nere su fondo bianco:

```
10 COLOR 1
20 COLOR 128
```

## COS

TIPO: Funzione matematica.

SINTASSI: COS (espressione)

COMMENTO:

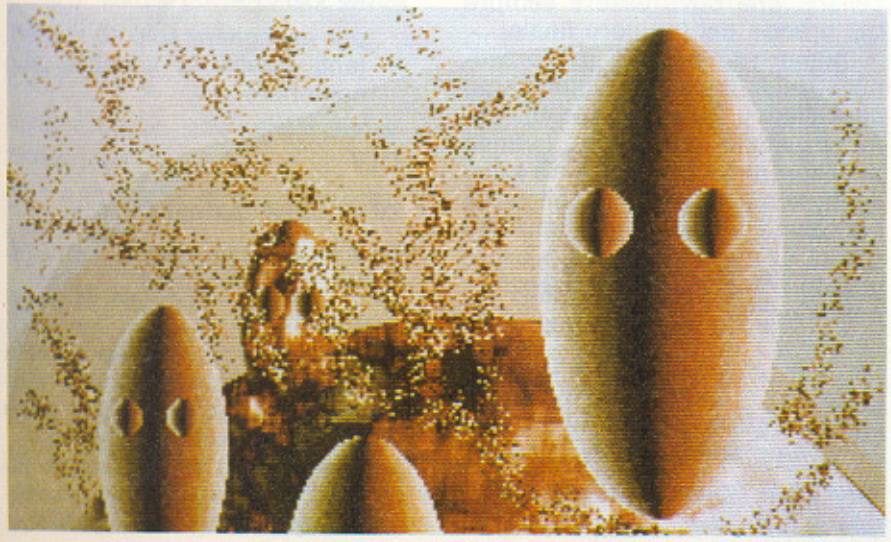
Questa funzione matematica restituisce il coseno di un angolo espresso in radianti.

Il valore restituito è di tipo reale compreso tra −1 e +1.

Il programma sotto proposto mostra uno dei possibili usi di questa funzione.

```
10 angolo = 3.14 15926
20 PRINT COS (angolo)
```

il numero visualizzato dovrà essere −1.



## COUNT

TIPO: Ausilio alla programmazione.

SINTASSI: COUNT (COU.)

COMMENTO:

Il comando appena analizzato, permette di avere un aiuto, specialmente nella stampa di tabelle. COUNT avrà, infatti, il valore corrispondente al numero dei caratteri stampati dopo l'ultimo return, valore compreso nel campo 0 - 255.

Il valore di questa variabile di sistema viene incrementato automaticamente ogni volta che un'istruzione PRINT, INPUT o REPORT viene eseguita; non viene modificato il valore se si usa un comando VDU o qualsiasi altro comando grafico.

COUNT assume il valore zero dopo ogni istruzione tipo CLS o MODE.

È possibile usare il comando anche per scrivere un definito numero di caratteri.

Un esempio è qui di seguito riportato.

```
10 REPEAT PRINT "=";
20 UNTIL COUNT = 30
30 SCRITTI = COUNT
```

## DATA (D.)

TIPO: Comando.

SINTASSI: DATA (dato1), (dato2), (dato...)

COMMENTO:

questo comando deve essere usato sempre in congiunzione con il comando READ.

(dato1), (dato2), (dato...) significa che possono essere elencati una serie di dati che potranno es-

sere letti in seguito dal comando READ.

Se vogliamo avere in memoria, e quindi sempre disponibili i mesi dell'anno, la linea DATA potrà avere la seguente forma:

```
10 DATA gennaio, febbraio, marzo, aprile, maggio, giugno
20 DATA luglio, agosto, settembre, ottobre, novembre, dicembre
```

Ricordate che per accelerare quanto più è possibile l'esecuzione dei programmi basic, le linee DATA devono trovarsi sempre all'inizio del programma.

Infatti l'interprete basic cerca nel programma i DATA a partire dall'inizio del programma presente in memoria.

Nei DATA possono convivere pacificamente anche variabili di tipo diverso, sarà compito infatti delle linee dove sono contenuti i comandi READ gestire correttamente la lettura delle diverse variabili.

## DEF

TIPO: Comando.

SINTASSI: DEF FN(nome), (lista variabili)
DEF PROC(nome), (lista variabili)

COMMENTO:

DEF viene usato per definire una funzione o una procedura.

Questo comando deve essere posto in testa a tutta la procedura, deve essere quindi obbligatoriamente posto come prima riga della stessa.

(nome) è il nome che si vuole dare alla procedura, richiamabile poi in tutto il programma solo con il nome.

(lista variabili) rappresenta l'insieme dei parametri che si vogliono trasferire da e per la procedura stessa, in modo da poter usare una sola procedura generale per lo svolgimento di compiti simili e ripetitivi.

## DEG

TIPO: Funzione matematica.

SINTASSI: DEG (valore)

COMMENTO:

Questa funzione matematica dà come risultato un numero reale uguale a questa espressione 180*N/PI; dove PI è il valore della costante pigreco (3.14 15926) ed N è il numero introdotto come (valore).

Converte quindi la misura di un angolo effettuata in radianti in un valore espresso in gradi.

Esempio:

```
10 PRINT DEG (PI/2)
```

sul video apparirà il valore in gradi dell'angolo retto; con una certa approssimazione dipendente dalla precisione di PI (pigreco).

## DELETE (DEL.)

TIPO: Ausilio alla programmazione.

SINTASSI: DELETE (numero iniziale), (numero finale)

COMMENTO:

Questo comando và usato con la certezza di conoscere con precisione la sua funzione. Questo per evitare di perdere parti di programmi in memoria.

(numero iniziale) e (numero finale) devono essere degli interi compresi nel campo 0 - 32767.

Questi numeri sono numeri di linea del programma che avete nella memoria di lavoro del basic.

(numero iniziale) è il primo numero di linea che si intende cancellare, che deve essere sempre più grande del (numero finale).

(numero finale) è il numero finale delle linee che intendiamo cancellare. Quando questo comando viene confermato, le linee di programma comprese tra i due limiti vengono irrimediabilmente cancellate. Fate molta attenzione quindi ai numeri di linea e usate questo comando con prudenza, è pericoloso!!

## DIM

TIPO: Comando.

SINTASSI: DIM (variabile(spazio riservato))

COMMENTO:

Questo comando permette di dimensionare, di riservare cioè una certa quantità di memoria, alla (variabile). Deve essere specificato lo spazio necessario alla (variabile), e se lo spazio è eccessivo vi verrà segnalato un errore di limitata disponibilità di memoria.

La lunghezza di una qualsiasi stringa non deve essere dimensionata, in quanto una stringa può avere una lunghezza massima di 255 caratteri.

Solo le variabili intere o matrici, anche bidimensionali devono usare l'istruzione DIM; essa va posta sempre all'inizio del vostro programma.

```
10 DIM a (10)
20 FOR I = 1 TO 11
30 A(I) = I
40 PRINT A(I)
50 NEXT I
```

questo programma è sbagliato, infatti la matrice di nome (a) è stata dimensionata per contenere 10 valori numerici, ma abbiamo voluto assegnarne 11. La correzione del programma è possibile:

1. modificando la linea 10 in questo modo:

   ```
   10 DIM a (11)
   ```

   per contenere anche l'undicesimo valore numerico.

2. cambiando la durata del ciclo FOR-NEXT:

   ```
   20 FOR I = 1 TO 10
   ```

   per assegnare solo 10 numeri, tanti quanti la variabile può contenere.

## DIV

TIPO: Funzione matematica.

SINTASSI: (espressione1) DIV (espressione2)

COMMENTO:

Il comando dà il quoziente della divisione tra (espressione1) e (espressione2) trascurando il resto dell'operazione matematica.

Il quoziente non è esatto, è un valore intero, e quindi arrotondato.

(espressione2) deve essere diversa da zero, altrimenti verrà segnalato un errore dal computer.

Esempio:

```
10 num1 = 100
20 num2 = 25
30 PRINT num1 DIV num2
```

sul video verrà visualizzato il numero 4.

Se invece di 100 la variabile num1 è 119, il numero visualizzato dal computer sarà sempre 4; infatti come già detto poco sopra, questa funzione non ritorna valori esatti, ma interi approssimati per difetto.

Per verifica provate questo programma:

```
10 num1 = 10
20 num2 = 25
30 PRINT num1 DIV num2
```

sul video dovreste leggere sempre il numero 4.

## DRAW (DR.)

TIPO: Comando.

SINTASSI: DRAW (espressione1), (espressione2)

COMMENTO:

Il comando, usato per il tracciamento di linee, parte dalle coordinate in cui si trova il cursore grafico attualmente, e si porta nelle coordinate date dai valori di (espressione1) e (espressione2). I limiti dei valori di (espressione1) e (espressione2) sono –32768/+32767.

## EDIT (ED.)

TIPO: Ausilio alla programmazione.

SINTASSI: EDIT (linee), (stringa)

COMMENTO:

L'uso di questo comando permette di usare l'editor per correggere testi basic.

Per capire l'importante funzione che questo comando svolge è bene sapere cos'è un Token.

Ogni comando basic presente in un programma in memoria viene convertito in un numero. Questo numero (Token appunto) permette al computer di riconoscere tra loro i comandi e, dando un codice a ciascuno di essi, consente un notevole risparmio di spazio.

Per esempio: il comando PRINT viene convertito nel valore esadecimale &F1.

L'editor prima di mostrare il testo «traduce» i Token in parole chiave basic.

A questo punto, quando il testo è pronto in memoria nel suo formato «tradotto», potete lavorare come con qualsiasi altro testo.

Quando poi terminerete il lavoro l'editor si incaricherà di tradurre nuovamente, da parola estesa (PRINT) a codice Token (&F1).

(linee) sono rispettivamente il numero iniziale e quello finale della parte del programma basic che desiderate correggere o rivedere.

(stringa) è un parametro che può essere omesso: è usato per far selezionare all'editor solo le linee di programma che contengano la (stringa), effettuando una ricerca in tutto il programma basic. Da notare che ambedue i testi saranno presenti in memoria, quindi se il programma basic è molto lungo, potrebbero nascere problemi di memoria.

A questo punto, se un errore di memoria è stato segnalato, è necessario dare il comando CLEAR.

## ELSE

TIPO: Comando.

SINTASSI: ELSE (comando)

COMMENTO:

Il comando ELSE può essere usato in una parte qualsiasi del programma, ma solo in congiunzione con la funzione IF ... THEN ... ELSE.

Inoltre può essere usato anche con il comando ON. spieghiamo ora i due fondamentali usi di questo comando per mezzo di programmi d'esempio in basic.

Modo d'uso 1

```
10 a = 1
20 IF a = 1 THEN PRINT "A vale UNO" ELSE PRINT "A diverso da UNO".
```

Provate a dare il RUN a questo programma, sul video comparirà la scritta "A vale UNO", ma se cambiate il valore nella linea 10, e al posto del numero 1 mettete un qualsiasi altro valore, sul video apparirà la scritta "A diverso da UNO".

Modo d'uso 2

```
10 scelta = 1
20 ON scelta GOTO 100,200,300,400 ELSE PRINT "scelta errata"
```

se il valore di (scelta) è compreso tra 1 e 4 allora la sequenza del programma si sposterà alle linee indicate dopo il GOTO.

Nel caso che (scelta) abbia un valore fuori dal campo 1-4 allora in questo caso sul video apparirà la scritta "scelta errata".

## END

TIPO: Comando.

SINTASSI: END

COMMENTO:

Questo comando indica al computer la fine di un programma basic.

Il computer si ferma comunque sempre all'ultima linea del vostro programma, ma la logica può richiedere la fine dopo che particolari condizioni sono state incontrate.

## ENDPROC

TIPO: Comando.

SINTASSI: ENDPROC

COMMENTO:

Il comando definisce la fine di una procedura definita dall'utente.

Quando chiamato, esso causa l'interruzione della procedura attualmente in uso e richiama le variabili globali, cioè le variabili in comune con il programma basic principale.

Il programma continuerà, passando il controllo al comando che si trova subito dopo quello che ha chiamato la procedura stessa.

Questo comando deve essere usato per chiudere una procedura, altrimenti un errore di mancata chiusura di procedura vi verrà segnalato.