I DUE BASIC DEL PC 128

Dopo due puntate della nostra rubrica, dedicate alla spiegazione delle nozioni preliminari del Basic, e prima di affrontare in modo più approfondito degli argomenti specifici, quali la grafica, la musica, il trattamento dei testi e dei dati e molti altri ancora, è nostra intenzione introdurre quello che può essere definito il vocabolario di questo linguaggio.

3ª Parte

Come potete vedere dalle pagine che seguono, non abbiamo parlato a caso di un vocabolario, infatti è proprio questa la struttura data all'esposizione di ogni singolo comando e non solo, ma ognuno di essi è posto rigorosamente in ordine alfabetico, così da non privilegiare nessuno e nel contempo, è questo il vero motivo, facilitare la ricerca dei singoli comandi.

Non c'è ancora molto da dire, se non augurarvi un buon lavoro e magari di avere un po' di pazienza, tutto ciò che imparate ora sarà utilissimo per comprendere meglio quegli argomenti a cui siete più interessati e da cui avrete molte soddisfazioni.

I comandi del BASIC del PC 128

La struttura di spiegazione di un comando sarà la seguente:

NOME DEL COMANDO

TIPO: cioè l'ambito d'uso del comando.

SINTASSI: cioè la forma corretta del comando, composta dal nome del comando e dai suoi parametri.

COMMENTO: cioè la spiegazione del comando e del suo uso. Per fare ciò saranno usati dei piccoli programmi d'esempio.

ABS
TIPO: Funzione di conversione.

SINTASSI: ABS (espressione)

COMMENTO:

Dove espressione può essere composta sia da numeri che da variabili, ma deve dare come risultato un numero.

Questo comando ritorna il valore assoluto di un numero e cioè il numero stesso, se esso è positivo, oppure il numero moltiplicato per -1 se questo è negativo.

Per comprendere meglio provate ad introdurre i seguenti micro programmi.

```
10 A=10: B=2
20 PRINT ABS(A*B)
```

```
10 A=10: B=-2
20 PRINT (A*B)
```

```
10 A=10: B=-2
20 PRINT ABS (A*B)
```

ASC
TIPO: Funzione di conversione.

SINTASSI: ASC (stringa)

COMMENTO:

Dove stringa può essere sia una variabile stringa che una stringa racchiusa tra virgolette.

Questo comando ritorna il valore corrispondente al codice ASCII, del primo carattere componente la stringa. Se la stringa in argomento è nulla, cioè due virgolette congiunte, il sistema ci risponderà con un "Illegal Function Call in n.", dove n. è il numero di linea in cui è riportata la stringa nulla.

Un discorso a parte va fatto per le lettere accentate, quest'ultime infatti, saranno composte da una sequenza di tre numeri, di cui il primo è sempre 22, che corrisponde appunto ad un carattere accentato.

Vediamo ora cos'è il codice ASCII di cui abbiamo parlato. Questo termine è l'acronimo di "*American Standard Code for Information Interchange*", ovvero codice standard americano per l'interscambio di informazioni. Con esso si rappresentano, tramite corrispondenza numerica, sia le lettere dell'alfabeto (minuscole e maiuscole), sia i numeri, sia dei caratteri di controllo del tipo: fine riga, a capo, eccetera. Ciò permette, perfino, l'interscambio di dati fra calcolatori diversi.

```
10 A$=""
20 PRINT ASC(A$)
```

```
10 A$="CIAO"
20 PRINT ASC("CIAO")
30 PRINT ASC(A$)
```

ATN

TIPO: Funzione matematica.

SINTASSI: ATN (numero)

COMMENTO:
Dove numero può essere: un numero in cifre, una variabile numerica, oppure un'espressione numerica.

Il comando da' come risultato l'angolo, espresso in radianti, avente come tangente l'argomento fornito.

```
10 Q=(2*(-4+2) )
20 PRINT ATN(Q)
30 PRINT ATN(2*(-4+2) )
40 PRINT ATN(8)
```

ATTRB

TIPO: Istruzione

SINTASSI: ATTRB larghezza, altezza

COMMENTO:
Dove i valori di larghezza e altezza possono essere solo 0 oppure 1.

Tramite questo comando si può controllare la larghezza e l'altezza di un carattere sullo schermo. Esso cioè raddoppia le dimensioni di un carattere nel seguente modo:

ATTRB 0,0 Lascia inalterato il formato.

ATTRB 1,0 Raddoppia la larghezza del carattere e ne lascia inalterata l'altezza.

ATTRB 0,1 Raddoppia l'altezza del carattere e ne lascia inalterata la larghezza.

ATTRB 1,1 Raddoppia sia l'altezza che la larghezza del carattere.

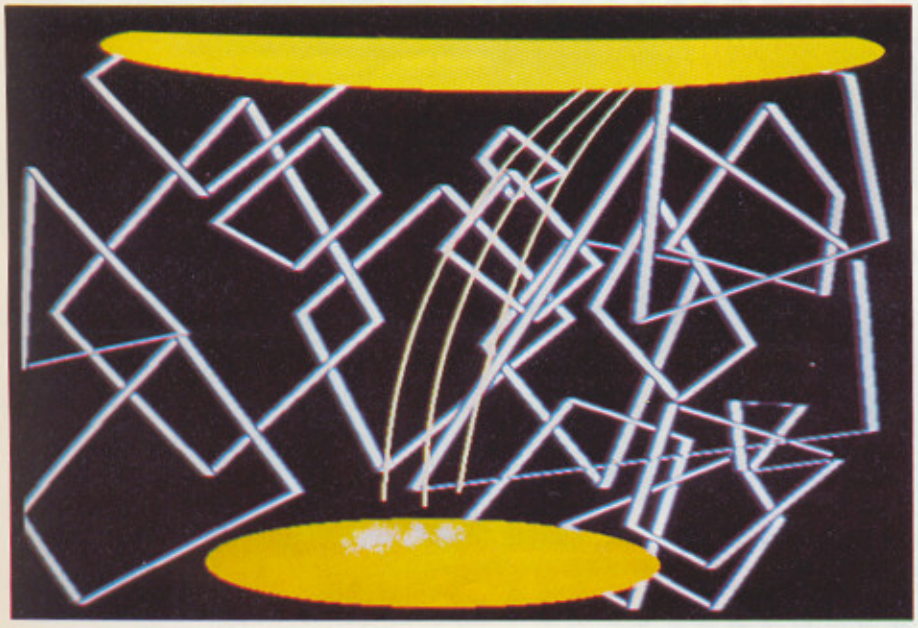
Come si può facilmente intuire, questo comando comporta, per il suo uso, una particolare attenzione. Infatti provate a scrivere:

```
10 ATTRB 1,1
20 CLS
30 PRINT "PIPPO"
40 END
```

Visto?
Provate invece a scrivere:

```
10 CLS
20 ATTRB 0,0
30 LOCATE 1,9
40 PRINT "PIPPO";
50 ATTRB 1,0
60 PRINT "PIPPO";
70 ATTRB 0,1
80 PRINT "PIPPO";
90 ATTRB 1,1
100 PRINT "PIPPO"
110 END
```

AUTO



TIPO: Comando diretto

SINTASSI: AUTO numero, passo

COMMENTO:
Dove numero è il valore iniziale della numerazione e passo è l'incremento della riga successiva.

Il comando provoca la numerazione automatica dei numeri di linea.

Se i due valori vengono ommessi, la numerazione, per default, partirà da 10 con incremento di 10. Se viene omesso il primo valore, la numerazione inizierà, per default, da 10 con l'incremento specificato in argomento. Se viene specificato solo il primo valore, la numerazione avrà inizio dalla riga specificata in argomento, con incremento, per default, di 10.

La numerazione automatica può essere esclusa sia tramite il comando CTRL-C, che CLS.

Il campo entro il quale devono essere compresi i numeri di linea va da 0 a 63999, qualsiasi valore esterno a questo ambito, causerà un Syntax Error. Nel caso in cui si numeri una riga già esistente, ciò causerà un errore di tipo Line not empty.

AUTO

AUTO20,15

AUTO30,

AUTO,25

BACKUP

TIPO: Comando di sistema

SINTASSI: BACKUP dr. 0 to dr. 1

COMMENTO:
Dove dr. 0 è il numero del primo drive e dr. 1 è il numero del secondo drive.

Tramite questo comando si esegue la copiatura dell'intero contenuto di un dischetto (disco sorgente) su un altro (disco destinazione). Se il nostro sistema è composto da due drive si scriverà:

BACUP 0 TO 1 inserendo il disco

sorgente nel drive 0 e il disco destinazione nel drive 1.

Se si possiede un solo drive la sintassi è la seguente:

BACKUP 0 Ciò comporterà la sostituzione alternata del dischetto sorgente e del dischetto destinazione, fino alla fine dell'operazione.

BANK

TIPO: Funzione

SINTASSI: BANK

COMMENTO:

Tramite questo comando si ottiene il numero del banco di memoria corrente. Il valore ottenuto sarà compreso tra 1 e 6. Per default, all'accensione del sistema, esso sarà 6.

```
10 print BANK
```

BANK

TIPO: Comando

SINTASSI: BANK numero

COMMENTO:

Seleziona il banco di memoria specificato in argomento (numero). Come visto sopra, il campo d'interesse di numero è da 0 a 6.

Questa istruzione fa riferimento ai banchi di memoria compresi fra gli indirizzi $H6000 e $H9FFF in esadecimale.

BEEP

TIPO: Comando

SINTASSI: BEEP

COMMENTO:

Per mezzo di questa istruzione, è possibile, da programma, far emettere un suono all'altoparlante del nostro monitor.

Ciò può essere utile in diverse circostanze, quali: giochi, avviso d'errore, eccetera.

```
10 CLS
20 BEEP
30 FOR A=1 TO 20
40 PRINT A
50 BEEP
60 NEXT A
70 END
```

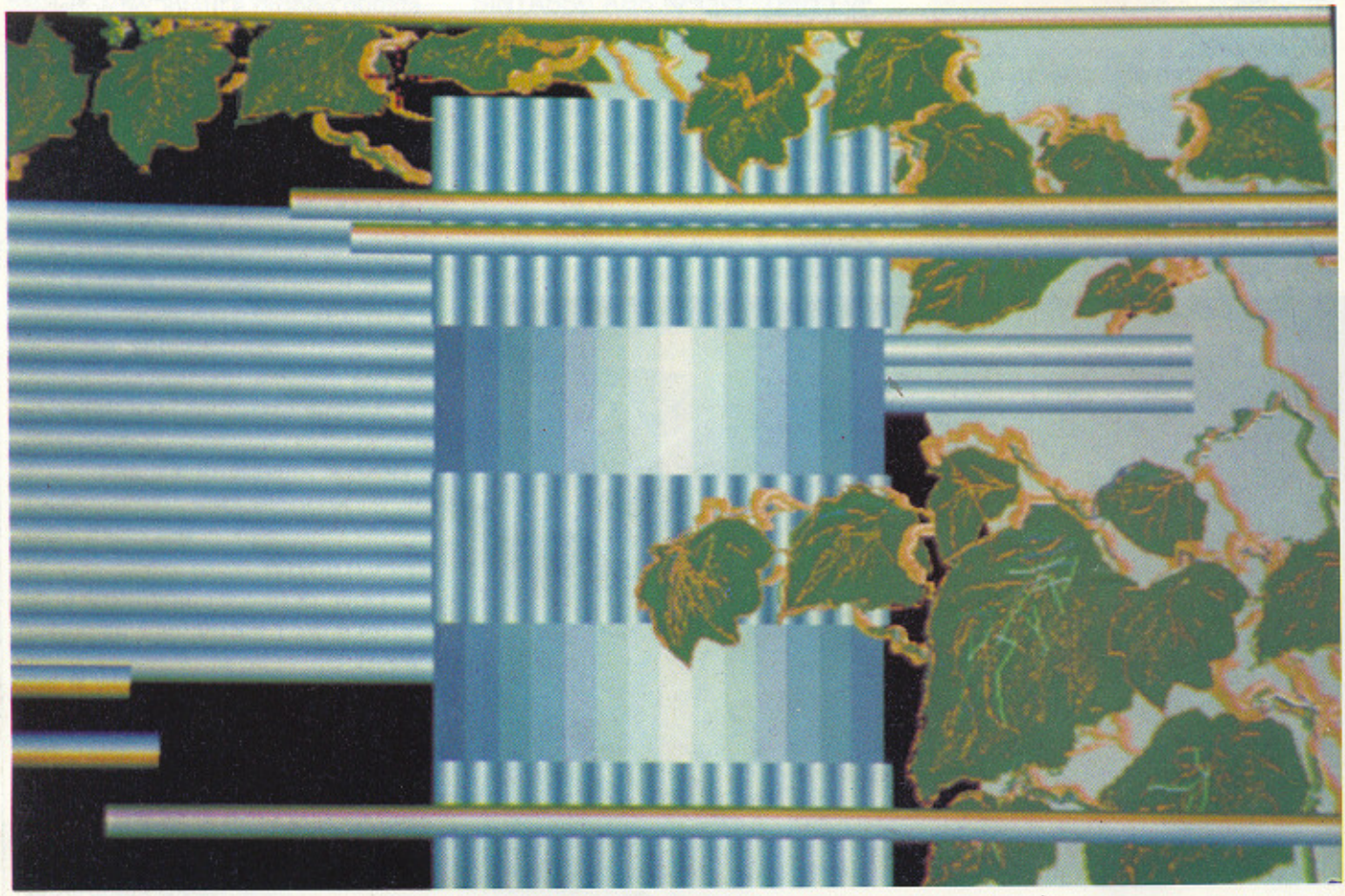
BOX

TIPO: Comando

SINTASSI: BOX (nc1, nr1) - (nc2, nr2) "caratt.", col., sfondo, inv.
BOX (nc1, nr1) - (nc2, nr2), colore

COMMENTO:

Le due sintassi si riferiscono, ri-



spettivamente, al modo carattere la prima e al modo grafico la seconda. I due tipi di grafica usati determinano il range entro cui si trovano i dati di riferimento dello schermo: nel primo caso, modo carattere, nc va da 0 a 79 e nr da 0 a 24, nel secondo caso, modo grafico, nc va da 0 a 319 e nr da 0 a 199.

I valori (nc1, nr1) individuano l'angolo superiore sinistro del rettangolo, tramite l'intersezione della colonna1 con la riga1. I valori (nc2, nr2), invece, individuano l'angolo inferiore sinistro tramite l'intersezione della colonna2 con la riga2.

La determinazione del modo carattere, invece del modo grafico, è data dalla presenza di un carattere, fra virgolette, nell'argomento del comando.

L'argomento colore è espresso tramite un numero e determina il colore del segno.

L'argomento sfondo, è un numero che determina il colore su cui viene scritto il segno.

In assenza dei dati della prima parentesi, il punto superiore sinistro del rettangolo è ricavato dall'ultimo punto dell'ultimo comando grafico del tipo:

BOX; BOXF; LINE; PSET; oppure avrà le coordinate 0,0 se non c'è alcun comando precedente.

Per un maggior approfondimento di questo comando, vedere le istruzioni COLOR, WINDOW, CONSOLE E STEP.

```
10 BOX (1,9) - (79,15) "C", 6,4,0
20 BOX - (15,22) "A", 3,5,1
30 BOX (10,30) - (216,48), 2
40 BOX - (30,120),6
```

BOXF

TIPO: Comando

SINTASSI: BOXF (nc1, nr1) - (nc2, nr2) "caratt.", col., sfondo, inv.
BOXF (nc1, nr1) - (nc2, nr2), colore

COMMENTO:

Con questo comando, identico nella sintassi al precedente, si ottengono dei rettangoli riempiti di un carattere, nel modo carattere, e da un PATERN o da un colore, nel modo grafico.

```
10 CLS
20 BOXF (33,10) - (150,200) "X", 8,1
30 BOXF (100,100) - (240,20), 3
```

CDBL

TIPO: Funzione di conversione.

SINTASSI: CDBL (numero)

COMMENTO:

Converte un numero di precisione qualsiasi in un numero in precisione doppia.

Per precisione singola si intende un numero che ha 6 cifre significative, mentre per precisione doppia si intende un numero con 16 cifre significative. Quanto appena detto, si riferisce di solito al modo in cui un calcolatore memorizza un numero, e più precisamente al numero di bit che questo mette a disposizione per ogni numero. Da ciò, un numero in precisione singola avrà meno bit a disposizione, di un numero a precisione doppia.

Altre nozioni sulla memorizzazione dei numeri, verranno date in un altro articolo.

Vediamo ora tre esempi di una divisione, che a prima vista dovrebbero dare lo stesso risultato, ma provate ad eseguirle, come sotto riportato, e rifletteteci sopra.

```
10 PRINT 1/3
20 PRINT CDBL (1/3)
30 PRINT CDBL (1)/3
```

Come potete notare, si hanno tre risultati diversi:

```
1) .333333
2) .3333333432674408
3) .3333333333333333
```

Il primo esempio è un numero a precisione singola, quindi il suo risultato sarà disposto in sei cifre.

Il secondo trasforma (1/3) in un numero in doppia precisione, quindi mostra il risultato in sedici cifre, grazie all'aggiunta di 10 cifre qualunque.

Il terzo esempio divide per 3 il numero a precisione doppia 1 e di conseguenza il risultato sarà mostrato in sedici cifre.

CHAIN

TIPO: Comando

SINTASSI: CHAIN "*nome file*", nr1-nr2, nr3

COMMENTO:

Dove "*nome file*" è il nome del file che si vuole unire al file in memoria, nr1 e nr2 sono i numeri di linea che si vogliono cancellare (opzionali), nr3 è il numero di linea da cui si vuole che inizi l'esecuzione del programma (opzionale). Se quest'ultima annotazione viene omessa, il programma avrà inizio dalla prima riga del programma.

Attraverso questo comando è possibile unire due programmi, di cui il primo residente in memoria e il secondo su memoria di massa esterna, senza che ciò comporti la perdita del contenuto delle variabili. Più esattamente, verranno mantenuti i valori delle variabili protette dal comando COMMON, il cui uso vedremo più avanti.

Bisogna fare attenzione che, a differenza del comando MERGE, dove i file da unire dovevano trovarsi in codice ASCII con CHAIN, i file devono trovarsi in formato binario.

Nel caso in cui il file caricato da disco abbia alcuni numeri di riga corrispondenti al file in memoria, le righe del primo programma sostitueranno le equivalenti del secondo.

Vediamo ora tre esempi esplicativi:

```
CHAIN"1:PIPPO", 300-700,50
CHAIN"PIPPO",300-
CHAIN"PIPPO"
```

Nel primo esempio le righe comprese tra 300 e 700 verranno cancellate e il file "*PIPPO*" sarà caricato dal drive 1. Il programma inizierà dalla riga 50:

Nel secondo esempio le righe di programma da 300 in poi verranno cancellate, il file "*PIPPO*" sarà caricato dal drive e il programma avrà inizio dalla prima riga.

Nell'ultimo esempio, il file "*PIPPO*" sarà caricato dal drive, nessuna riga di programma verrà cancellata e il programma avrà inizio dalla prima riga.