

CORSO DI BASIC PER IL PC 128S

Terza Parte

Con questa terza parte prosegue il corso dedicato alla spiegazione dei comandi BASIC disponibili per il vostro computer Olivetti Prodest PC 128S.

Come già detto nel numero precedente, 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 una guida, un manuale, dal quale attingere informazioni preziose.

Continuiamo pertanto la spiegazione del linguaggio BASIC, linguaggio che vi permetterà di colloquiare con il vostro computer e di creare programmi che risolvano i vostri problemi scolastici o professionali.

Come di consueto, i comandi verranno esposti secondo lo schema qui di seguito riportato :

Nome comando : è il nome del comando che viene trattato; ricordate di scriverlo sempre in maiuscolo, altrimenti il vostro Olivetti Prodest PC 128S non sarà in grado di comprenderlo.

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 compren-

dere 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.

Con un po' di impegno da parte vostra, seguendo con attenzione queste pagine, imparerete molte cose utili : comunque, per qualsiasi chiarimento, potete sempre scrivere alla redazione di Olivetti Prodest User .

Andiamo quindi a continuare l'elencazione e la spiegazione dei comandi Basic.

ENVELOPE (ENV.)

TIPO : Comando

SINTASSI: ENVELOPE (dato1),
...(dato14)

COMMENTO :

Il presente comando definisce l'inviluppo sonoro. L'inviluppo è un po' difficile da spiegare in poche parole diciamo pertanto che regola l'evoluzione nel tempo di una forma d'onda sonora.

Visto così l'inviluppo non è molto chiaro, vediamo quindi di spiegare prima uno per uno i quattordici parametri di cui questo comando è composto, poi, per mezzo di sem-



plici diagrammi, di quali parti principali si compone un'onda sonora.

Di seguito sono descritti dettagliatamente i quattordici dati da fornire al comando ENVELOPE: questi hanno valori compresi tra 0 - 255; in caso che i valori siano diversi, vi verrà segnalato quali valori sono ammessi.

Essi controllano vari aspetti della forma d'onda, generata con il comando SOUND.

Il numero tra parentesi indica quale dato, o dati, tra i quattordici esistenti, si sta trattando. In questo modo, variando opportunamente i quattordici dati previsti, è possibile creare qualsiasi suono, da quello di un pianoforte, al rumore tipico di

un'astronave in fase di atterraggio.

Con questo comando potrete quindi dare un tocco di realismo in più ai vostri programmi.

Alla fine delle spiegazioni teoriche è presente un grafico con la relativa "traduzione" del comando ENVELOPE e del comando SOUND: anch'esso necessario per far "parlare" il vostro PC 128S.

(1) N - Numero di inviluppo

Il numero di inviluppo può variare tra 1 e 16. Nel caso in cui l'RS 423 sia usata i soli valori permessi per l'inviluppo variano tra 1 e 4.

Questo numero identificherà in modo permanente, all'interno del vostro programma Basic, il tipo di

suono. Saranno quindi selezionabili, se precedentemente definiti, ben 16 tipi diversi di sonorità.

(2) T - Intervallo

Questa è la lunghezza di un singolo intervallo di tempo nell'inviluppo: ovvero, controlla la velocità di cambiamento tra la frequenza e l'ampiezza. Il tempo di variazione è misurato in centesimi di secondo.

(3 - 5) P11, P12, P13 incrementi della nota fondamentale

L'inviluppo della nota fondamentale è diviso in tre parti. Questi parametri controllano la quantità di variazione della nota in ogni singolo passo di inviluppo.

Essi sono trattati come byte con segno, quindi il valore è compreso tra -128 e +127.

(6 - 8) PN1, PN2, PN3 passi di nota fondamentale.

Questi tre parametri danno il numero di passi in ogni sezione dell'inviluppo della nota.

Il campo di valori ammessi è 0 - 255.

La nota varia con P11 per PN1 volte, poi con P12 per PN2 volte e, infine, con P13 per PN3 volte.

Il cambiamento di nota verrà effettuato ogni T centesimi di secondo.

L'ammontare della variazione della nota durante l'inviluppo è dato dalla formula matematica riportata di seguito: $P11 * PN1 + P12 * PN2 + P13 * PN3$.

Dopo che è trascorso il tempo dato dalla seguente espressione, $T * (PN1 + PN2 + PN3)$, l'inviluppo del suono termina.

Se il valore di T è compreso tra 0 - 127, allora l'inviluppo della nota viene ripetuto, altrimenti la nota rimane setata al valore finale.

(9) AA - Variazione dell'altezza nella fase di attacco (ATTACK)

Sebbene la nota di un inviluppo controllato parta sempre dal valore impostato dal comando SOUND,





l'ampiezza parte sempre da zero, per poi portarsi al valore settato.

Per mezzo di questo comando è possibile, ogni T centesimi di secondo, incrementare l'ampiezza fino al valore specificato in AA.

Il valore di AA può variare tra -128 e +127.

Notate che se l'ampiezza iniziale è zero, saranno ammessi solo numeri positivi; nel caso contrario, in cui si parta con ampiezza zero e si fornisca un incremento negativo, vi verrà segnalato un errore.

(10) AD - Variazione dell'altezza nella fase di decadimento (DECAY)

Il decadimento parte da un valore di altezza definita e fa decrescere il volume sonoro fino al limite inferiore definito.

Questo è il punto iniziale dato del valore dello sostentamento.

Il parametro AD può variare da -128 fino a +127.

(11) AS - Variazione dell'altezza nella fase di sostentamento (SUSTAIN)

Quando il decadimento, nel suo decrescere verso il valore zero, incontra il valore definito per il sostentamento, interrompe il suo decrescere e lascia il controllo alla fase di sostentamento.

Il valore del sostentamento è solitamente zero, ma può essere settato come si desidera, tenendo conto che i valori ammessi, valevoli anche per la fase di decadimento, variano tra -127 +128.

(12) AR - Variazione dell'altezza nella fase di rilascio.

Alla fine della fase di sostentamento inizia la fase finale dell'involuppo, il rilascio.

Durante questa fase l'altezza viene variata ogni T centesimi di secondo in dipendenza dell'ammontare del valore dato in AR, il quale, naturalmente, varia tra -127 +128.

La fase di rilascio termina quando l'ampiezza diventa uguale a zero.

(13) ALA - Obiettivo del livello di attacco

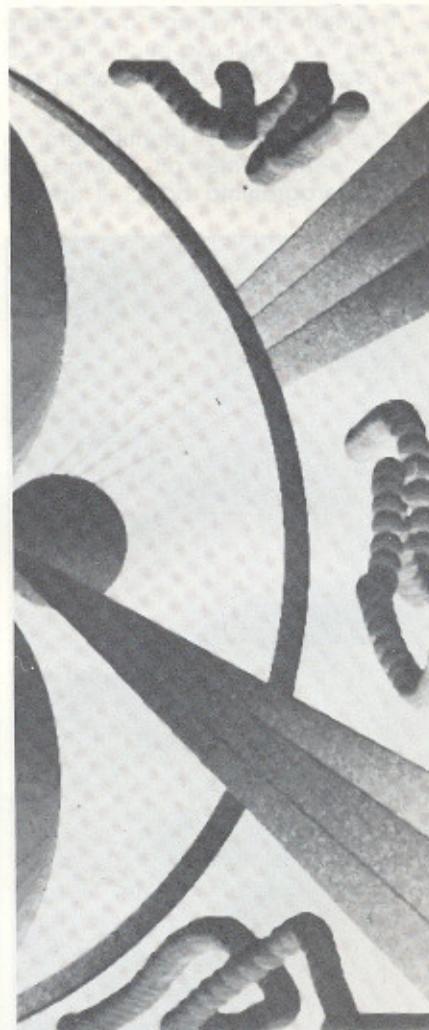
Il volume del suono è incrementato, secondo AA, ogni T centesimi di secondo fino ad arrivare al valore di

ALA. Notate che il valore di ALA può variare tra 0 e 126.

(14) ALD - Obiettivo del livello di decadimento.

Il volume del suono incrementa, secondo AD, ogni T centesimi di secondo fino ad incontrare il valore fissato da ALD.

ALD, come del resto ALA, può va-



riare nel campo compreso tra il limite inferiore 0 e quello superiore 126.

Queste sono le spiegazioni relative ai vari parametri, come promesso, nella tabella è riportato il grafico relativo ad un'involuppo ed al relativo comando SOUND (trattato più avanti) necessario anch'esso per creare il suono descritto dalla figura A.

EOF

TIPO : Comando.

SINTASSI : EOF # (Numero del file).

COMMENTO :

Questo comando è indispensabile per comprendere se il file dal quale stiamo leggendo i dati, è finito, oppure vi sono altri dati da leggere.

Se vi sono altri dati da leggere EOF# è FALSO, mentre se la fine del file è stata incontrata EOF# fornirà il valore booleano VERO.

EOR

TIPO : Operatore

SINTASSI : (relazione) EOR (relazione)

COMMENTO :

Questo utile comando permette di eseguire l'operazione di OR esclusivo.

L'EOR è una operazione dell'algebra booleana e il suo funzionamento è relativamente semplice.

In pratica l'EOR è vero solo se le due relazioni sono diverse.

ERL

TIPO : Funzione

SINTASSI : ERL

COMMENTO :

Il valore di ERL contiene il numero di linea relativo ad un nostro programma, al quale si è verificato l'errore incontrato dall'interprete.

Molto utile, per testare i vostri programmi, permette di andare direttamente alla linea che ha causato l'errore.

ERR

TIPO : Funzione

SINTASSI : ERR

COMMENTO :

Questa funzione ritorna il numero che identifica l'ultimo errore occorso durante l'esecuzione del programma attualmente in memoria.

Il risultato è un valore numerico compreso tra 0 - 255.

È importante ricordare che i valori numerici ammessi devono essere compresi tra 1 - 127 e sono errori prodotti da programmi BASIC, men-

tre gli errori occorsi al sistema operativo hanno un numero di errore superiore a 127.

Questi errori possono essere controllati dall'istruzione ON ERROR: eccetto l'errore che porta il numero zero, il quale causa un errore fatale non controllabile neanche dall'istruzione di seguito spiegata, ON ERROR.

ERROR [ON ERROR] (ERR.)

TIPO : Comando

SINTASSI : ON ERROR (comando)

COMMENTO :

Il comando ON ERROR viene spesso usato per controllare se nel corso del programma viene incontrato un errore.

In questo caso il controllo del programma viene mandato alla riga che contiene l'istruzione ON ERROR.

È quindi possibile gestire gli errori in modo dinamico, nello stesso momento in cui vengono rilevati permettendo di creare programmi completi e di livello professionale.

EVAL

TIPO : Funzione

SINTASSI : EVAL(valore)

COMMENTO :

Il comando EVAL tenta di valutare una stringa come una espressione valida del BASIC. Questo comando segnalerà un errore se verranno assegnati valori non appropriati.

EXP

TIPO : Funzione

SINTASSI : EXP (valore)

COMMENTO :

Funzione matematica che ritorna il valore dell'esponenziale dell'argomento.

Il valore del risultato sarà positivo nel campo da 0 al massimo numero permesso dalla precisione della macchina.

Esso può essere rappresentato come "e" elevato all'argomento specificato tra parentesi (valore).

Ricordate che "e" è una costan-

te, essa vale 2.718281828. Per gli esperti matematici, di seguito, è riportato come ottenere il coseno iperbolico, facendo uso proprio della funzione EXP.

DEF FNcosh(x) = (EXP(x) + EXP (-x)) / 2

EXT

TIPO : Comando

SINTASSI :

1. EXT # (valore)
2. EXT # (valore) = (espressione)

COMMENTO :

Spieghiamo separatamente i due modi di utilizzo di questo comando: questa è una pseudo-variabile che controlla l'estensione di un file.

Il file deve essere prima aperto.

1. (valore) deve corrispondere al numero di file precedentemente definito da un comando OPEN. Il risultato sarà un numero intero che rappresenterà la quantità di dati presente nel file: sarà compresa tra 0 fino al valore massimo 2147483648.

2. (valore) deve corrispondere al numero di un file precedentemente aperto con un'istruzione OPEN.

(espressione) è la lunghezza desiderata del file: tenere sempre presente che la lunghezza massima è limitata dalla capacità del sistema operativo, mentre la lunghezza minima è zero. Con questo comando è possibile restringere la lunghezza di un file; perdendo, al suo interno alcuni dati. Attenzione quindi all'uso di quest'opzione. I dati persi non saranno recuperabili con tecniche semplici.

FALSE

TIPO : Funzione

SINTASSI : FALSE (FA.)

COMMENTO :

Ritorna il valore logico Booleano FALSE. Il valore numerico associato è zero.

Per capire meglio a cosa può servire questo comando pseudo-variabile

riportiamo un esempio.

segnalino = FALSE

In questo esempio il valore di "segnalino" sarà zero. FALSE, inoltre, può essere usato come "bandiera" (flag nella dizione anglosassone), per controllare e gestire appropriatamente svariate operazioni.

Nell'esempio citato, segnalino può essere usato in un ciclo REPEAT UNTIL come variabile di controllo: se segnalino è FALSE continua; se invece segnalino = Vero allora termina il ciclo ed il controllo viene passato all'istruzione immediatamente successiva.

FN

TIPO : Comando

SINTASSI :

1. DEF FN (parte procedura)
2. FN (parte procedura)

COMMENTO :

Come di consueto, quando sono presenti due diverse sintassi, dividiamo il commento in due parti.

1. Il formato della (parte procedura) è riportato di seguito qui sotto. L'esempio riporta la definizione della funzione che ritorna il valore minimo tra due valori passati alla funzione stessa:

```
1000 DEF FNmin(a%,b%) IF
a% > b% THEN =a% ELSE =b%
```

a% e b% sono parametri chiamati "formali". Ad essi vengono passati i valori che si desidera far valutare da FNmin. È possibile usare qualsiasi comando BASIC che sia necessario alla logica del programma, compresi cicli (FOR - NEXT), comandi di condizione (IF), ecc. È possibile definire, inoltre, variabili locali, per preservare il valore delle variabili usate nel programma principale.

2. Il secondo uso è quello di attivare una funzione precedentemente definita. L'uso della funzione, definita nella prima parte della spiegazione del comando FN, potrebbe essere il seguente:

PRINT FNmin(2 * banane% , 3 * mele% + 1)

Sul video apparirà stampata la quantità minore tra il numero delle banane moltiplicato per due e il numero delle mele moltiplicato per tre più ancora un'altra mela. Naturalmente è possibile utilizzare questa funzione anche per usi diversi dall'esempio fatto. È con-

a scoprire cose che tutti usano e che comunque già sono note.

FOR (F.)

TIPO : Comando

SINTASSI:

FOR (variabile numerica)
= (espressione1) TO (espressione2)
STEP (espressione3)



igliabile prendere nota di questi esempi: in tale modo se in un programma ci sarà la necessità di cercare il maggiore tra due numeri, non sarà necessario creare un'altra funzione, ma basterà copiare questa proposta. Nei paesi di lingua inglese c'è un detto, che equivale al nostro: "Non scoprire l'acqua calda.", cioè non perdere tempo, e quindi denaro,

COMMENTO :

Intestazione di un ciclo che comincia da (espressione1) fino a ' TO ' (espressione2) a passi ' STEP ' (espressione3).

STEP non è obbligatorio, quindi se omissso si intende che gli incrementi saranno effettuati aggiungendo il valore intero uno (1) a (espressione1) sino a farla coincide-

re, come valore numerico, a (espressione2).

(variabile numerica) può essere una qualsiasi variabile valida, come ad esempio I oppure LUNGHEZZA o altro. (espressione) può essere una qualsiasi espressione numerica, oppure, un numero.

L'unica condizione obbligatoria è che (espressione1) deve avere un valore esclusivamente intero.

Ricordate sempre, quando usate questa istruzione che il comando FOR è parte del comando FOR - NEXT : l'uso del FOR senza il NEXT darà luogo alla segnalazione di un errore. Per tale motivo l'esempio verrà effettuato nella trattazione del comando NEXT.

GCOL (CG.)

TIPO : Comando

SINTASSI : GCOL (espressione1), (espressione2)

COMMENTO :

Questo comando è usato per settare i colori nel modo grafico.

Il primo argomento (espressione1), è necessario per settare i valori di plottaggio: i valori ammessi sono compresi nel campo che va da 0 a 255.

Esso determina direttamente gli effetti delle prossime azioni del comando PLOT sul video.

Di seguito, nella tabella riportata qui di seguito, sono indicati i valori disponibili correntemente usati :

VALORE SIGNIFICATO

- | | |
|---|--|
| 0 | Memorizza il colore dell'(espressione2) sul video; |
| 1 | Effettua la fusione dei colori tra (espressione2) e lo schermo (or); |
| 2 | Effettua la somma dei colori tra il valore dell'(espressione2) e il colore dello schermo (and); |
| 3 | Effettua l'EOR (or esclusivo) del colore definito dall'(espressione2) e quello presente sul video; |
| 4 | Inverte il colore corrente, ignorando il valore dell'(espressione2). |
| 5 | Non ha nessun effetto sullo schermo corrente. |

(espressione2) determina il colore che deve essere combinato con quello presente sullo schermo, secondo le azioni descritte poco sopra.

Il campo dei valori ammessi varia da 0 a 127. Aggiungendo 128 verrà cambiato non il colore in primo piano, ma il colore di fondo.

Di seguito sono riportati due esempi:

```
GCOL 4,128 : CLG : REM inverte
la finestra grafica
GCOL 1,2 : REM OR del video con
il colore 2
```

Provateli sul vostro PC 128S e cambiate i valori per vedere come lo schermo varia al variare dei valori di (espressione1) e di (espressione2).

GET

TIPO : Funzione

SINTASSI : GET

COMMENTO :

Questa funzione ritorna un carattere dall'input definito. L'input può essere sia da tastiera che, caso più raro, dalla porta RS 432.

Questa funzione resta in "ascolto" sino a che un carattere non è disponibile dall'input definito.

L'esempio di seguito riportato resta in attesa, e non cede il controllo alla riga successiva, fino a quando non viene premuta la barra spaziatrice.

```
10 PRINT " Premi la barra spaziatrice
per continuare " : REPEAT UNTIL GET = 32
20 PRINT " Ok , possiamo continuare "
```

Ovviamente, è possibile variare il tasto da premere: per far ciò è sufficiente, al numero 32, sostituire il codice ASCII corrispondente al tasto scelto.

La tabella completa dei codici ASCII corrispondenti ai tasti disponibili sulla tastiera, comprese le maiuscole e i segni di interpunzione, è nell'appendice 3, riportata a pag. 184 della vostra "Guida all'uso

del sistema PC 128S", il manuale in dotazione con il vostro computer.

GET\$(GE.)

TIPO : Funzione

SINTASSI : GET\$

COMMENTO :

Questa funzione ritorna un carat-

tere dal dispositivo di input corrente dal valore 65, mentre GET\$ ritornerà proprio "a".

GOSUB (gos.)

TIPO : Comando

SINTASSI : GOSUB (espressione)

COMMENTO :

Questo comando passa il control-



tere dal dispositivo di input corrente.

La funzione è molto simile a quella assoluta dal comando GET, ma a differenza di questo, GET\$ ritorna direttamente il carattere, non il numero ASCII corrispondente.

Quindi, nel caso in cui premiate il tasto corrispondente alla lettera "a", la funzione GET ritornerà il

lo alla linea specificata dal valore di (espressione).

La linea richiamata deve essere presente e deve contenere una subroutine, che deve terminare obbligatoriamente con l'istruzione RETURN.

(espressione) può essere anche un valore calcolato, cioè il risultato di una espressione matematica: con la condizione che la formula stessa ritorni un numero intero positivo.

Ricordate che è importante ritornare da una chiamata ad una sub-routine con la parola chiave BASIC RETURN; un errore di " chiamata alla sub-routine senza istruzione di ritorno " verrà segnalata dal computer.

Il comando RENUMBER, il quale permette di variare i numeri di linea, non lavora correttamente con il comando GOSUB; cioè le linee indicate nell'istruzione GOSUB non verranno variate: fate attenzione quindi e controllate le righe GOTO e GOSUB manualmente.

Esempi di GOSUB validi :

```
GOSUB 1000
GOSUB (500*i % )
GOSUB (10 + LEN(a$) )
```

Esempi di GOSUB non validi :

```
GOSUB (10/3)
GOSUB (100/7)
```

GOTO (G.)

TIPO : Comando

SINTASSI : GOTO (espressione)

COMMENTO :

Questo comando permette di trasferire il controllo del programma ad una riga definita da un numero. (espressione) contiene il numero di linea.

Il seguente programma mostra l'uso, in questo caso decisamente massiccio, del comando GOTO.

Solitamente, non è bene che i programmi abbiano una struttura di GOTO così complicata, questo deve pertanto rimanere solo un esempio.

```
10 GOTO 30
20 END
30 PRINT " Sono alla riga 30 "
40 GOTO 70
50 PRINT " Sono alla riga 50 "
60 GOTO 20
70 PRINT " Sono alla riga 70 "
80 GOTO 50
```

Il programma seguente è completamente identico a quello che avete appena letto, ma fa uso delle abbreviazioni che il BASIC dell'Olivetti Prodest PC 128S permette di usare.

Questo per aiutarvi a familiarizzare con le abbreviazioni dei comandi: le quali permettono una maggiore

velocità di stesura dei programmi.

```
10 G. 30
20 END
30 P. " Sono alla riga 30 "
40 G. 70
50 P. " Sono alla riga 50 "
60 G. 20
70 P. " Sono alla riga 70 "
80 G. 50
```

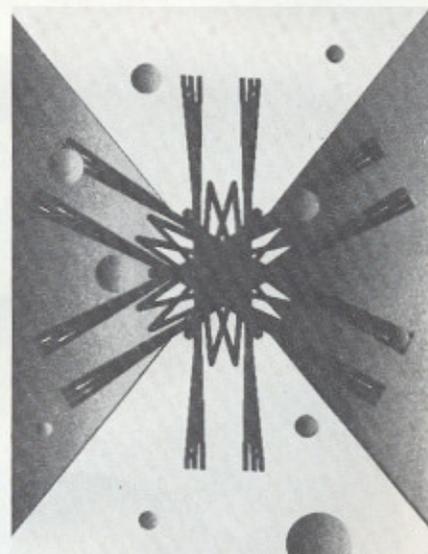
È necessario ricopiare esattamente il programma, compresi i punti dopo le lettere G e le lettere P.

HIMEM (H.)

TIPO : Comando

SINTASSI :

1. HIMEM



2. HIMEM = (espressione)

COMMENTO :

Per questo comando ci sono due possibili sintassi, le quali, come di consueto vengono spiegate separatamente. Il primo modo di usare il comando HIMEM è quello usato per conoscere quale è la locazione di memoria da cui inizia lo stack.

Lo stack è una parte riservata della memoria dove risiedono tutte le variabili del programma BASIC che attualmente risiede in memoria.

Lo spazio dedicato allo stack cresce durante l'esecuzione del programma, pertanto questo breve programma vi dirà, con una approssimazione molto buona, quanta memo-

ria è ancora disponibile nel computer.

10 PRINT ABS (HIMEM - LOMEM)

La seconda modalità d'uso, per il comando HIMEM, è quella che permette di settare da programma lo spazio dedicato alle variabili del programma.

Tale settaggio è da fare in modo molto preciso, non è il caso di effettuare esperimenti, in quanto il programma residente in memoria potrebbe avere dei gravi malfunzionamenti. Infatti, modificando direttamente i dati presenti in memoria, se non si è a conoscenza della locazione precisa dove termina il programma, è possibile anche cancellare parti dello stesso.

IF

TIPO : Comando

SINTASSI :

IF (espressione) THEN (comandi)
ELSE (comandi)

COMMENTO :

Questo comando permette di effettuare delle scelte, secondo il valore di alcuni parametri.

Se l'(espressione) è vera, verranno eseguiti comandi posti dopo THEN, altrimenti verranno eseguiti quelli posti dopo la parola chiave BASIC ELSE.

Il programma che segue mostra un esempio dell'uso di questa istruzione.

```
10 IF A = 0 THEN PRINT " Il valore di A è ZERO ." ELSE PRINT " Il valore della variabile di nome A è diverso da ZERO . . . . "
```

INKEY

TIPO : Comando

SINTASSI :

1. INKEY (espressione)
2. INKEY (espressione)
3. INKEY (espressione)

COMMENTO :

Il presente comando permette di svolgere principalmente tre funzioni, di seguito descritte esaurientemente una alla volta.

1. In questo primo modo di utilizzare l'istruzione INKEY l'argomento deve essere un numero intero, nel campo 0 - 32767, e rappresenta il tempo massimo concesso per l'attesa del carattere da leggere dal dispositivo di input. Il tempo è misurato in centesimi di secondo, quindi, ad esempio, il valore cento corrisponderà al tempo di attesa di un secondo. Il valore ritornato è il codice ASCII, se un carattere è stato rilevato, o il valore -1 se il tempo scade senza rilevare alcun carattere.

```
10 REM PROVA DI RIFLESSI, IL TEMPO MASSIMO
20 REM È 30 CENTESIMI DI SECONDO
30 variabile = INKEY(30)
40 IF variabile < -1 THEN PRINT "
OK .. SEI UN VERO LAMPO"
ELSE PRINT " AOOO ?.??.? SVEGLIA CHE È GIA' TARDI . "
```

Per provare i vostri riflessi subito dopo avere premuto RUN, e poi il tasto di conferma, premete velocemente, per quanto vi è possibile, un altro tasto e osservate il video. Variando il valore 30, presente nella riga 30, è possibile diminuire o aumentare il tempo di attesa.

2. La seconda possibilità di impiego sono in test utilizzati per conoscere quale tasto è premuto nell'istante in cui il comando viene eseguito. Per poter utilizzare questa opzione è bene sapere che il carattere deve essere specificato come numero negativo. Ad esempio: se vogliamo che il programma aspetti sino a quando il tasto relativo alla lettera "h" non venga premuto, è sufficiente ricopiare il programma qui di seguito riportato:

```
10 IF INKEY (-72) THEN REPEAT
UNTIL NOT INKEY (-72)
20 PRINT "OK FINALMENTE LO
HAI PREMUTO ... "
```

Notate che se volete cambiare il tasto da aspettare è sufficiente andare a consultare il manuale a pagina 184, l'Appendice 3 infatti mostra tutti i numeri che nel codice ASCII corrispondono a lettere, numeri, o simboli.

3. Il terzo e conclusivo uso di que-

sto comando permette di conoscere la versione del sistema operativo attualmente in uso.

```
10 PRINT INKEY (-256)
```

Il valore stampato sarà il numero di codice del sistema operativo attualmente installato sul vostro computer.

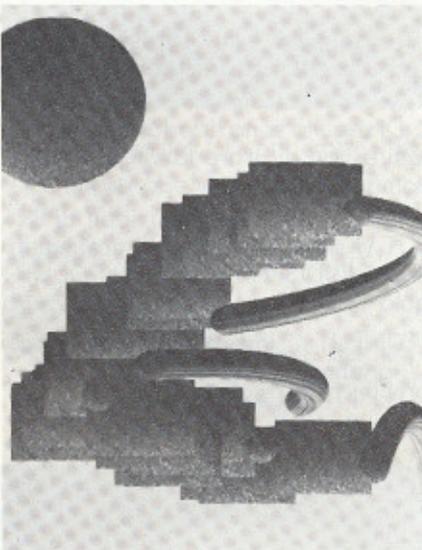
INKEYS

TIPO : Comando

SINTASSI : INKEY\$ (variabile)

COMMENTO :

Il comando INKEY\$, come uso, è molto simile al comando INKEY, nel modo 1.



(variabile) è una espressione numerica che indica il tempo (in centesimi di secondo), durante il quale il computer resta in totale ascolto della tastiera.

Al contrario del comando INKEY il valore ritornato è il carattere stesso, non il suo corrispondente codice ASCII.

Il breve programmino d'esempio servirà sicuramente a farvi comprendere, sino in fondo, l'uso di questo comando.

```
10 REPEAT UNTIL INKEY$ (500) =
"a"
20 P."Tempo scaduto o tasto Premuto ."
```

Questo programmino aspetta per il tempo specificato (500 centesimi

di secondo) che venga premuto il tasto contraddistinto dalla lettera "a".

Trascorso questo tempo, il controllo del programma verrà ceduto alla linea successiva (linea 20).

Se INKEY\$ fornirà il valore -1, allora vuol dire che non è stato premuto, nel tempo definito, nessun tasto.

INPUT (I.)

TIPO : Comando

SINTASSI : L'istruzione INPUT ha una sintassi sicuramente non difficile, ma certamente non riassumibile nel modo consueto. Per supplire a questa deficienza, il numero di esempi sarà, ovviamente, superiore a quelli riportati per altri comandi.

COMMENTO :

Come già spiegato nella parte solitamente dedicata alla spiegazione della sintassi, il comando INPUT si presta a molteplici usi.

Fondamentalmente è stato incluso nel set di istruzioni del vostro computer PC 128S in quanto è il comando che permette l'operazione di acquisizione dei dati.

Con questo comando potrete ricevere dalla tastiera vari dati, siano essi numerici che stringa.

Vi sono vari modi di ricevere informazioni dall'esterno: di seguito verranno riportati i più usati.

Alla fine della digitazione dei dati da fornire al programma, è obbligatorio premere il tasto di conferma (RETURN).

1. La prima possibilità offerta è quella di effettuare un ingresso di dati non commentato. Il computer in seguito ad una istruzione di INPUT visualizza solo un punto di domanda, senza specificare che tipo di dato è richiesto. La riga BASIC che effettua questa funzione è la seguente:

```
10 INPUT A$
20 PRINT A$
```

Di seguito è riportato quello che vedrete sul video:

```
? ciao
ciao
```

Alla domanda di immettere i dati, nell'esempio è stata digitata la pa-



rola ciao, successivamente (riga 20), il computer ha stampato la stringa data in ingresso.

2. Sostanzialmente la seconda possibilità è uguale a quella appena spiegata, ma in questo caso l'INPUT è del tipo commentato, ciò spiega cosa introdurre dalla tastiera.

```
10 INPUT "Dimmi quanti anni hai ",
    eta%
20 PRINT "Hai visto per ben
    ";eta%; " volte la primavera . ."
```

Sul video appariranno i seguenti caratteri :

```
Dimmi quanti anni hai ? 32 Hai visto
per ben 32 volte la primavera . .
```

Notate che il punto di domanda dopo "Dimmi quanti anni hai" viene aggiunto automaticamente dal BASIC.

3. In questo terzo caso vi mostreremo come evitare di vedere obbligatoriamente il punto di domanda alla fine della stringa di commento dell'istruzione di INPUT.

```
10 INPUT " NOME "nome$
20 PRINT " Lieto di fare la tua
    conoscenza ";nome$
```

In questo caso, evitando di mettere la virgola dopo la stringa di

commento, è stato possibile sopprimere il punto di domanda, che molte volte può dare fastidio.

Il consiglio che vi diamo, comunque, è quello di usare sempre un INPUT di tipo commentato (esempio 2 o 3), e in generale conviene usare il tipo descritto nell'esempio numero 2.

Infatti nella maggioranza dei casi sia il modo 1 che il modo 3 non vengono usati.

Il comando di INPUT ha delle regole ferree, che se non rispettate, vi daranno notevoli problemi.

La prima regola da osservare è la seguente :

Il terminatore del dato da inserire in ingresso dalla tastiera è, oltre al tasto RETURN, anche la virgola.

La semplice e innocua virgola può causare dei problemi non indifferenti.

Di seguito sono riportati i modi corretti e non di usare la virgola con il comando INPUT. Modo esatto :

nel caso che nell'istruzione di INPUT vi siano più variabili da immettere è **INDISPENSABILE** usare la virgola :

```
10 INPUT " Immettere nome e co-
    gnome ", nome$, cognome$
20 PRINT " Ciao, carissimo " ; no-
    me$ ; " " ; cognome$
```

sul video troverete queste scritte :
Immettere nome e cognome ? Pinco-
Pallino Ciao, carissimo Pinco Pallino

Modo scorretto :

Il modo scorretto di usare la virgola è rappresentato tipicamente da questo programma :

```
10 INPUT " Dimmi chi vuoi salutare
    ";saluti$
20 PRINT " Bene facciamo i saluti a
    ";saluti$
```

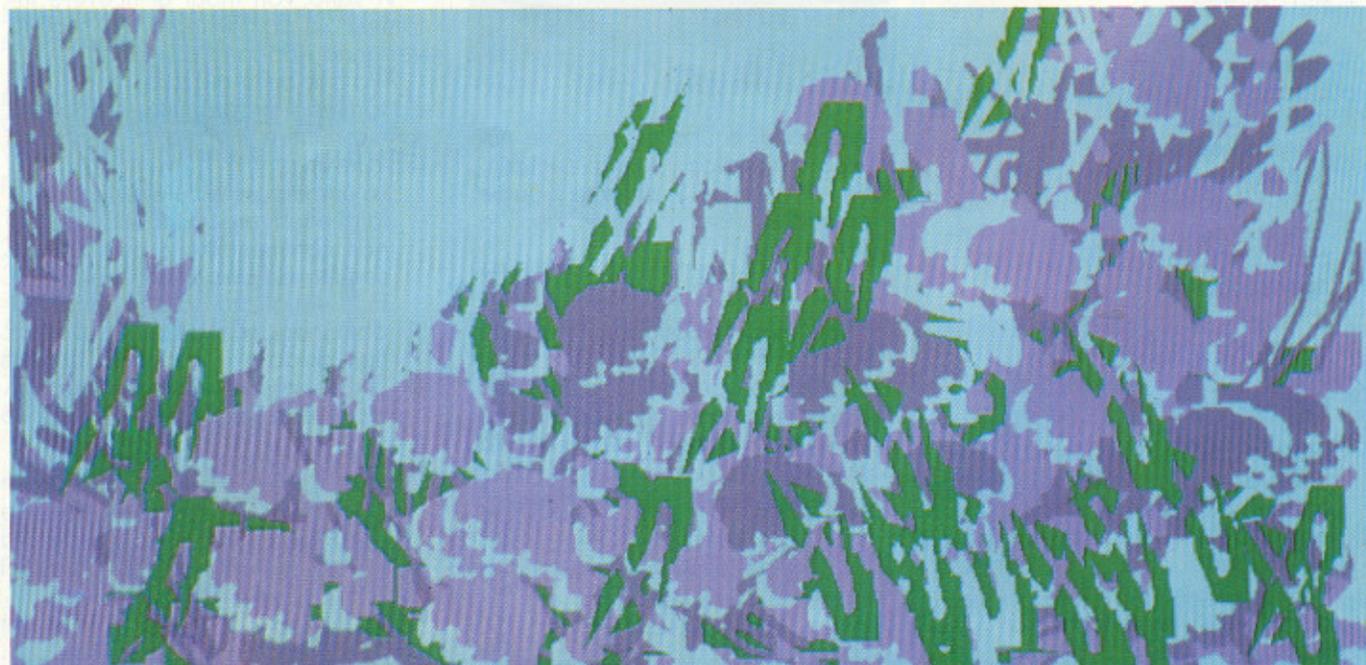
sul video, dopo la domanda effettuata dal computer con il commento dell'istruzione INPUT rispondete come in esempio :

```
Dimmi chi vuoi salutare ? Mamma-
Papà e Fratello Bene facciamo i sa-
luti a Mamma
```

Come potete ben capire, tutto quello che c'era dopo la virgola è stato completamente ignorato. Attenzione quindi nell'immissione di stringhe, non usare assolutamente la virgola.

Se volete a tutti i costi leggere una virgola, come è stato spiegato in modo chiaro, non potete usare il comando INPUT.

Ma molti si chiederanno : "Non è possibile leggere un testo che al suo interno abbia dei segni di interpunzione ?".





Naturalmente a tutto c'è rimedio.

Per leggere questa virgola, che ci crea non pochi problemi, è sufficiente usare il sub-comando **LINE**: questo legge dalla tastiera tutto ciò che viene battuto, compresi gli spazi e le nostre carissime virgole.

L'esempio, dei saluti, riportato precedentemente, può essere riscritto in questo modo:

```
10 INPUT LINE " Dimmi chi vuoi salutare ";saluti$
20 PRINT " Bene facciamo i saluti a ";saluti$
```

In questo caso dopo la domanda effettuata dal computer con il commento dell'istruzione **INPUT** rispondete nuovamente come in esempio:

```
Dimmi chi vuoi salutare ? Mamma-
Papà e Fratello
Bene facciamo i saluti a Mamma-
Papà e Fratello
```

riportando tutto ciò che è stato immesso.

Ricordando tutti i consigli riportati, per il corretto uso della parola chiave Basic **INPUT**, potrete finalmente instaurare un "dialogo approfondi-

to" con il vostro Olivetti Prodest PC 128S.

INPUT # (I. #)

TIPO : Comando

SINTASSI: INPUT # (nome del file),
(variabile1), (variabile2)
... (variabileN)

COMMENTO :

L'uso di questa istruzione non è così laborioso come il nome potrebbe far sembrare.

Questo comando è usato per leggere dati da un file specificato che, una volta letti, vengono immessi nelle variabili specificate.

Un esempio può sicuramente far capire meglio di un numero elevato di parole.

```
10 input# data,nome$,cognome$,
indirizzo$
```

questa linea di programma leggerebbe, se il file fosse presente sul disco, tre dati:

- il nome, posto nella variabile nome\$;
- il cognome, posto nella variabile cognome\$;
- l'indirizzo della persona specificata dai due dati precedenti.

INSTR (INS.)

TIPO : Comando

SINTASSI : INSTR(espressione1),
(espressione2), (espressione3)

COMMENTO :

Questo comando è stato introdotto nel BASIC dell'Olivetti Prodest PC 128S per aiutare l'operatore nella manipolazione delle stringhe.

Infatti la parola chiave **INSTR** indica se una certa stringa è contenuta in un'altra.

(espressione1) è la stringa nella quale volete cercare una certa sequenza di caratteri.

(espressione2) è la serie di caratteri che viene confrontata con (espressione1) per trovare in quale posizione c'è; se è presente, la stringa (espressione2).

Il valore che (espressione3) prenderà dipende dalla posizione in cui verrà trovata (espressione2) nella stringa (espressione1).

Nel caso questa ricerca fallisse, il valore ritornato sarebbe zero.

La lunghezza massima di (espressione1) è di 255 caratteri, di conseguenza in valore di (espressione3) sarà nel campo 1 - 255.