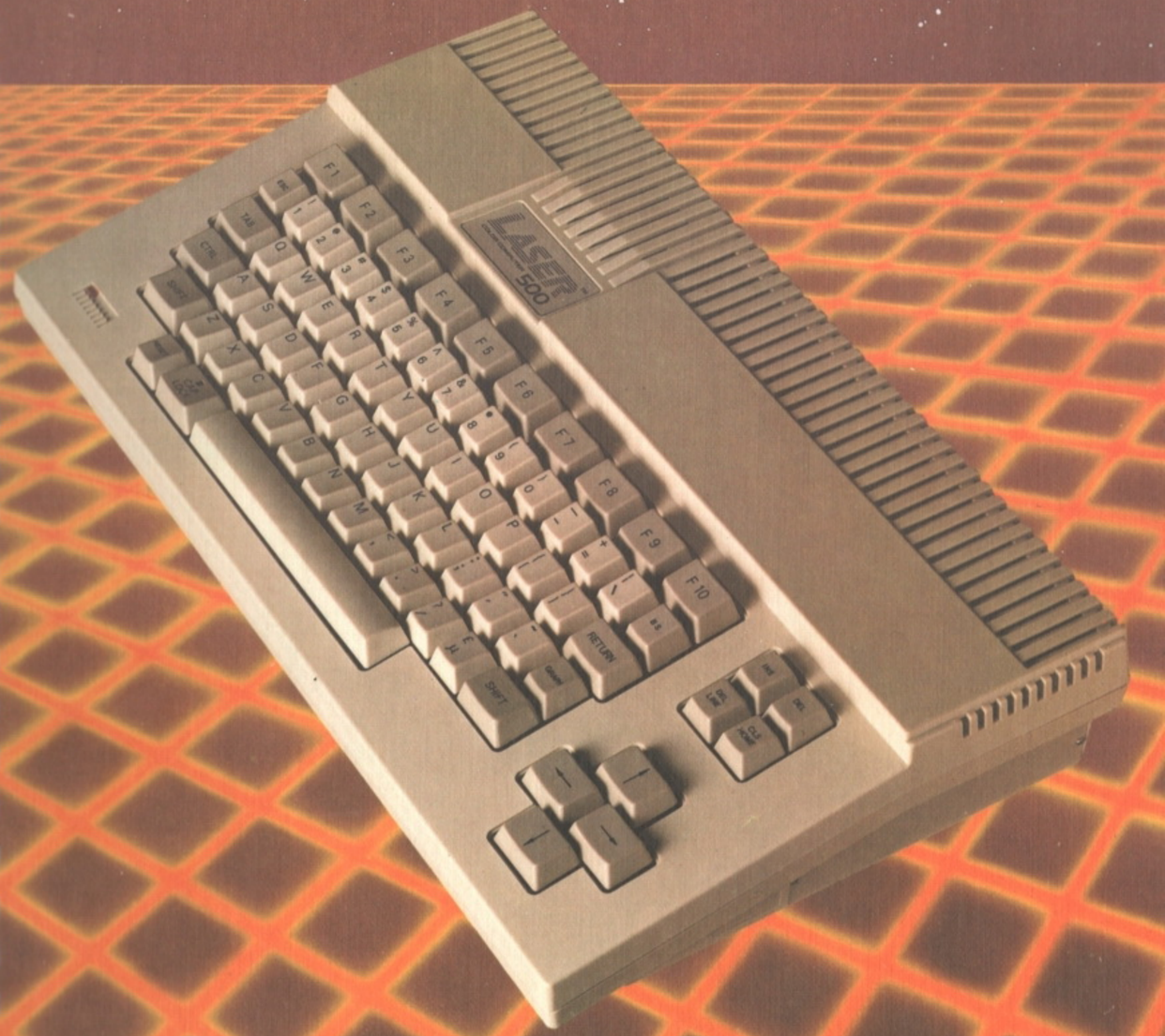
CORSO DI INFORMATICA

LIBRO ESERCIZI - 2



SCUOLA Scheidegger

La strada della perfezione è sempre difficile, e molte volte è anche lunga e faticosa da percorrere.

La Scuola Internazionale Scheidegger ha dimostrato a tutto il Mondo che è possibile studiare, ed imparare, in modo semplice, divertente, avvincente ed in tempi brevi.

Come è possibile questo?

Applicando sempre metodologie didattiche che permettano di valorizzare il lavoro dell'uomo, facendolo partecipare, e coinvolgendolo nello studio, attraverso la pratica e l'esercitazione.

Per questo la Scuola Scheidegger riscuote grande successo in campo internazionale.

Il nostro corso di informatica richiede che l'utente partecipi in modo attivo sin dalla prima lezione, per acquisire in breve tempo una completa autonomia nei confronti degli strumenti di lavoro. Per questo le esercitazioni pratiche hanno una fondamentale importanza per conseguire la più ampia soddisfazione.

Anche le verifiche, realizzate con i test, sono proposte come momento di riflessione, perchè l'utente possa valutare in modo autonomo quali sono le eventuali lacune da colmare.

L'unica cosa che chiediamo a chi segue i nostri corsi è questa:

PARTECIPATE IN MODO ATTIVO ALLE LEZIONI, ESEGUITE TUTTE LE ESERCITAZIONI PRATICHE, ED APPLICATEVI IN MODO COSTRUTTIVO.

Seguendo queste semplici regole, non avrete alcuna difficoltà a conseguire il successo che desiderate.

BUON LAVORO ! ! !

Scuola Internazionale Scheidegger
LA DIREZIONE
Cav. A Cagienard

LEZIONE 4

OBIETTIVI

1. Chiarire il concetto di LOOP per ottimizzare e velocizzare le operazioni ripetitive.
2. Definire l'uso dell'istruzione FOR ... TO ... STEP ..... NEXT, in un unico CICLO.
3. Approfondire l'uso dell'istruzione FOR ... NEXT per cicli concatenati, chiarendo limiti ed errori di logica.
4. Definire il concetto di funzione alfanumerica.
5. Chiarire l'uso delle funzioni LEFT$ (X$,I) MID$ (X$,I,J,) e RIGHT$ (X$,I) e CONCATENAZIONE di STRINGHE
6. Chiarire l'uso delle funzioni di conversione LEN (X$), STR$ (X), VAL (X$), CHR$ (X), ASC (X$).

1 CHIARIRE IL CONCETTO DI LOOP PER OPERAZIONI RIPETITIVE

Il programma TABELLE MATEMATICHE, (esercitazioni a casa della precedente lezione), offre lo spunto per spiegare un'altra istruzione molto importante del BASIC.

Si tratta di un'istruzione complessa, cioè di una FRASE, realizzata per mezzo di due o più parole.

La sintassi è la seguente:

FOR ... = ... TO ... STEP ...
istruzione 1 : istruzione 2 : istruzione 3
istruzione 4 : istruzione 5 : istruzione 6
NEXT ...

Vediamo ora come deve essere interpretata e completata, (cioè cosa bisogna mettere dove ci sono i puntini).

Dopo la parola FOR bisogna sempre mettere il NOME di una variabile numerica. In particolare, alcuni computer accettano soltanto NOMI di variabili numeriche REALI.

Dopo il NOME della variabile c'è il segno = (uguale).

Dopo il segno uguale bisogna mettere UN VALORE NUMERICO REALE.

Si può mettere (secondo l'esigenza), o una COSTANTE NUMERICA (cioè un numero), o il nome di una variabile numerica.

Dopo questo valore c'è la parola TO e dopo questa parola occorre indicare un altro valore numerico (una COSTANTE, o una variabile con NOME diverso dalla precedente).

Dopo tale valore numerico si può mettere la parola STEP, seguita da un altro valore numerico come già visto sopra.

In questo modo la PRIMA PARTE della FRASE è completa.

Dopo questa frase ci saranno altre istruzioni BASIC qualsiasi; alla fine bisogna mettere l'istruzione NEXT, seguita dal NOME di variabile che avevamo messo dopo la parola FOR.

FOR..... NEXT.

```
LOOP
10 FOR I = 1 TO 1Ø STEP 1
20 LET N = I * I
30 PRINT I ; N
40 NEXT I
50 END
```

La frase completa può essere tradotta nel modo seguente:

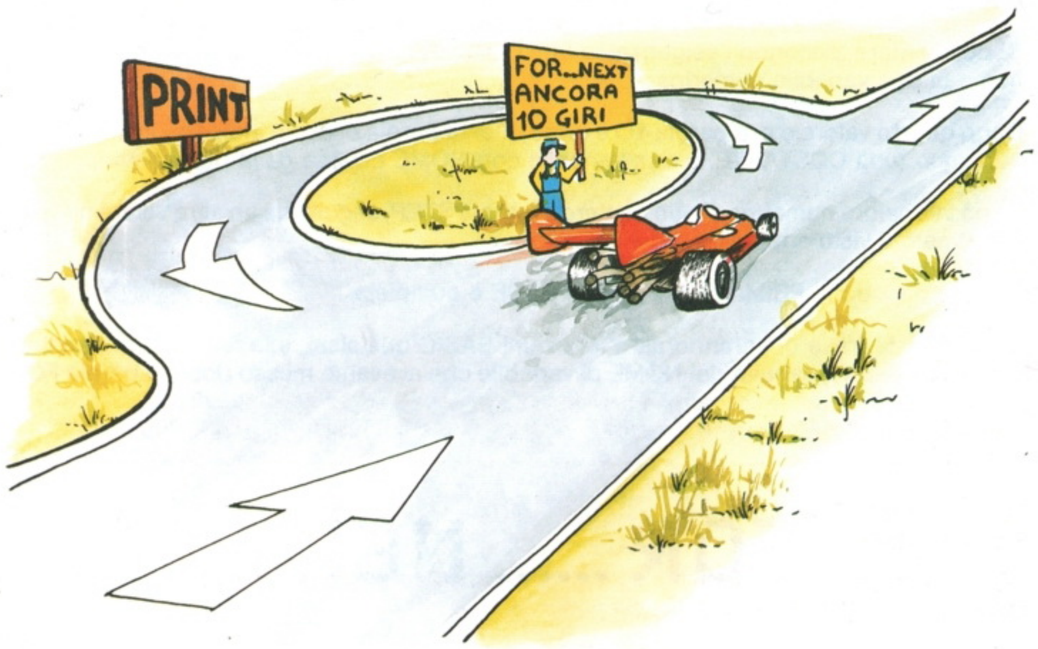
PER tutti i valori della variabile, iniziando con il valore indicato dopo il segno uguale, fino al valore finale che è indicato dopo la parola TO, incrementando con la quantità precisata dopo la parola STEP, RIPETI tutte le istruzioni di programma comprese tra la FRASE FOR ... e l'istruzione NEXT.

Il computer eseguirà una prima volta le istruzioni indicate.

Quando incontrerà la parola NEXT, eseguirà l'incremento, CONTROLLANDO se è stato SUPERATO il VALORE FINALE.

Se il VALORE FINALE NON è superato il computer eseguirà ancora le ISTRUZIONI ed incrementerà la variabile ogni volta, finchè la variabile avrà SUPERATO il valore finale.

Tale ciclo di operazioni (SALTO indietro e RIESECUZIONE), prende il nome di LOOP (ANELLO), e può essere usato per eseguire operazioni ripetitive, come avremo modo di constatare.



2 DEFINIRE L'USO DELL'ISTRUZIONE FOR ..... NEXT

La frase FOR ..... NEXT è una delle istruzioni di ELABORAZIONE tra le più complesse del BASIC, in quanto permette di ripetere un gruppo di istruzioni, per diversi VALORI di una VARIABILE, controllandone ogni volta il valore.

È possibile eseguire il conteggio in senso CRESCENTE o DECRESCENTE, con (o senza) INTERVALLI.

ESEMPIO 1:
10 FOR I=1 TO 10 STEP 1 : LET N=I * I : PRINT I : N : NEXT I

le istruzioni LET e PRINT vengono eseguite 10 volte, per i valori della variabile I, con inizio da 1 fino a 10.

ESEMPIO 2:
10 FOR I=10 TO 1 STEP –1 : LET N=I * I : PRINT I;N : NEXT I

le istruzioni LET e PRINT vengono eseguite 10 volte, per i valori della variabile I, con inizio da 10, fino ad 1, e con decremento pari a 1 (cioè 10, 9, 8, 7, etc.).

ESEMPIO 3:
10 FOR I=10 TO 100 STEP 10 : LET N=I * I : PRINT I;N : NEXT I

le istruzioni LET e PRINT sono eseguite 10 volte, per i valori compresi tra 10 e 100, e incremento di 10, (cioè 10, 20, 30, 40, 50, etc.).

Occorre notare quanto segue:
Le istruzioni comprese tra FOR e NEXT (cioè le istruzioni del LOOP), sono SEMPRE ESEGUITE, almeno UNA VOLTA.

Il computer esegue il controllo SOLO quando incontra l'istruzione NEXT, e solo allora è in grado di stabilire se è necessario RIPETERE o NON RIPETERE il CICLO (LOOP).

ESEMPIO: 10 FOR I=1 TO 0 STEP 4 : PRINT I : NEXT I

In questo caso l'istruzione PRINT I viene eseguita la prima volta, il computer trova l'istruzione NEXT I incrementa la variabile (che era 1 e diventa 1+4=5). Il nuovo valore è MAGGIORE del valore finale indicato (che è 0), ed il CICLO non viene RIESEGUITO.

Se la parola STEP ed il valore di incremento, NON sono indicati il computer sottointende un incremento POSITIVO, pari ad 1.

Ciò significa che la frase: FOR I=1 to 10 STEP 1 può essere semplificata nel modo seguente: FOR I = 1 TO 10 mantenendo inalterato il significato.

3 APPRENDERE L'USO DI FOR ... NEXT PER CICLI CONCATENATI

Vediamo ora un altro aspetto importante del CICLO FOR ... NEXT.

Come già detto: tra la frase FOR e l'istruzione NEXT, è possibile inserire QUALSIASI ISTRUZIONE BASIC, compreso un'altro CICLO FOR ... NEXT.

Il BASIC offre questa possibilità, ma è necessario operare secondo una regola precisa, che è la seguente:

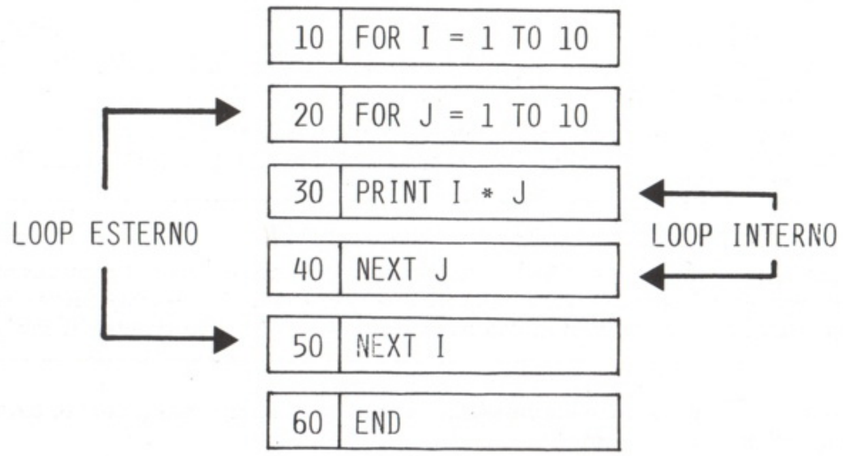
Se all'interno di un CICLO si mette un'altra istruzione FOR ... il nuovo ciclo DEVE essere CONTENUTO COMPLETAMENTE nel precedente.

L'istruzione NEXT relativa al SECONDO CICLO deve pertanto essere scritta PRIMA dell'istruzione NEXT relativa al PRIMO CICLO.

ESEMPIO:
10 FOR I=1 TO 10 : FOR J=1 TO 10 : PRINT I * J : NEXT J : NEXT I

L'istruzione PRINT I * J viene eseguita 100 volte, in quanto il primo CICLO (relativo alla variabile I), contiene un altro CICLO (relativo alla variabile J).
Per ogni valore di I (da 1 al 10) viene eseguito il secondo CICLO (per i valori di J da 1 a 10). L'istruzione PRINT è all'interno di questo secondo CICLO, per cui sarà eseguita per 100 volte.

L'istruzione NEXT J (relativa al SECONDO CICLO), è scritta PRIMA dell'istruzione NEXT I, come richiesto dal BASIC.



Il programma TABELLE MATEMATICHE (esercizi a casa LEZIONE 3), usa i CICLI CONCATENATI (CICLI contenuti in altri CICLI), sia per il calcolo della TAVOLA PITAGORICA (righe 110-160), sia per la ricerca dei numeri PRIMI (righe 320-370).

Per imparare l'uso dei CICLI FOR ... NEXT è necessario eseguire prove con il proprio computer, per verificare l'esecuzione.

È molto utile eseguire mentalmente le operazioni, per controllare se il computer opera nel modo previsto oppure no.

È facile commettere errori di logica, che il computer non è in grado di controllare, soprattutto se all'interno di un CICLO si mette un'istruzione di SALTO incondizionato (GOTO), oppure una frase IF ... THEN ... con un SALTO CONDIZIONATO.

È importante sapere che il computer considera ESAURITO un CICLO SOLO e SOLTANTO quando, (trovando l'istruzione NEXT di quel CICLO) ed incrementando il valore della variabile, trova tale valore SUPERIORE (se l'incremento è positivo), o INFERIORE (se l'incremento è negativo), al valore FINALE.

In ogni altro caso il CICLO rimane APERTO e può generare errori.

4 DEFINIRE IL CONCETTO DI FUNZIONE ALFANUMERICA

Abbiamo già parlato delle FUNZIONI MATEMATICHE e delle FUNZIONI di EDITING; parleremo ora di altre FUNZIONI molto utili del BASIC.

Prima però è meglio ripetere che cosa sono le FUNZIONI nel linguaggio BASIC.

Le funzioni sono parole (solitamente di tre lettere), che permettono di eseguire operazioni complesse, su un certo valore.

Questo valore (chiamato ARGOMENTO della funzione), è indicato tra le parentesi che ci sono dopo il nome della funzione.

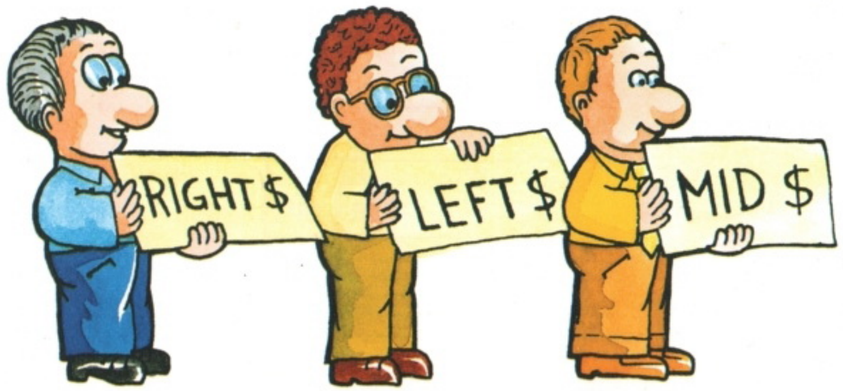
L'ARGOMENTO di una funzione può essere di diverso tipo, in relazione alla FUNZIONE stessa.

Le funzioni che vedremo ora sono chiamate FUNZIONI ALFANUMERICHE, perchè hanno come ARGOMENTO COSTANTI o VARIABILI ALFANUMERICHE.

Vedremo inoltre le FUNZIONI di CONVERSIONE NUMERICA-ALFANUMERICA, e le FUNZIONI di CONVERSIONE ALFANUMERICA-NUMERICA.

Parleremo anche delle FUNZIONI di conversione ASCII, per la codifica e decodifica dei CARATTERI.

Le FUNZIONI ALFANUMERICHE permettono di operare su costanti e variabili di tipo ALFANUMERICO.



Le principali FUNZIONI ALFANUMERICHE del BASIC sono le seguenti:

RIGHT$(arg$,N)
LEFT$(arg$,N)
MID$(arg$,N1,N2)

in cui RIGHT, LEFT e MID sono il NOME della FUNZIONE.

Il segno $ (DOLLARO) indica che si tratta di FUNZIONI ALFANUMERICHE,
arg$ sta ad indicare un ARGOMENTO alfanumerico, cioè una COSTANTE alfanumerica, oppure il NOME di una VARIABILE alfanumerica,
N, N1 ed N2 stanno ad indicare dei VALORI NUMERICI, (COSTANTI, VARIABILI o FUNZIONI numeriche).

Si può notare che le funzioni RIGHT e LEFT hanno il NOME di 5 e 4 lettere rispettivamente, mentre (come già visto), la maggior parte delle FUNZIONI ha il nome di tre lettere.

```
10 CLS
20 PRINT "CALCOLO DEL GIORNO"
30 PRINT "DELLA SETTIMANA":PRINT
40 PRINT "INSERISCI LA DATA"
50 PRINT "NEL SEGUENTE MODO:":PRINT
60 PRINT "2 CIFRE PER IL GIORNO,":PRINT
70 PRINT "2 CIFRE PER IL MESE,":PRINT
80 PRINT "4 CIFRE PER L'ANNO,":PRINT
90 PRINT "SENZA SEPARAZIONI !:":PRINT
100 INPUT "DATA";DT$
110 IF LEN(DT$)<>8 THEN 10
120 ER%=0:FOR I=1 TO 8:A$=MID$(DT$,I,1)
130 IF A$>"9" OR A$<"0" THEN ER%=1
140 NEXT I : IF ER%=1 THEN 10
150 G$=LEFT$(DT$,2):G0=VAL(G$)
160 M$=MID$(DT$,3,2):M0=VAL(M$)
170 A$=RIGHT$(DT$,4):A0=VAL(A$)
180 IF (M0=2 AND G0>29) THEN 10
190 IF (M0=2 AND G0>29 AND A0/4<>INT(A0/4)) THEN 10
200 IF G0>31 OR M0>12 OR A0>2100 THEN 10
210 IF G0<1 OR M0<1 OR A0<1800 THEN 10
220 IF G0>30 AND (M0=4 OR M0=6 OR M0=9 OR M0=11) THEN 10
230 GOSUB 290:PRINT
240 PRINT G$;"/";M$;"/";A$;"=";WD$(G2)
250 PRINT
260 INPUT "ALTRA DATA (S/N)";Y$
270 IF Y$="S" THEN 10
280 CLS:END
290 M1=M0+1:A1=A0
300 IF M0<3 THEN A1=A0-1:M1=M0+13
310 M2=INT(30.6001*M1)
320 A2=INT(A1*365.25)
330 G1=A2+M2+G0+5
340 G2=G1/7:G3=INT(G1/7)
350 G2=INT((G2-G3)*7+.9)
360 WD$(0)="DOMENICA"
370 WD$(1)="LUNEDI"
380 WD$(2)="MARTEDI"
390 WD$(3)="MERCOLEDI"
400 WD$(4)="GIOVEDI"
410 WD$(5)="VENERDI"
420 WD$(6)="SABATO"
430 RETURN
```

Il programma qui a fianco permette di verificare il giorno della settimana di una data qualsiasi, compresa tra 1/1/1800 ed il 31/12/2100.
Fa uso di molti controlli per determinare la validità della data inserita, ed usa molte funzioni alfanumeriche.

5 CHIARIRE L'USO DELLE FUNZIONI LEFT$, RIGHT$, e MID$ E CONCATENAMENTO STRINGHE

Il BASIC permette inoltre il CONCATENAMENTO di VARIABILI e COSTANTI ALFANUMERICHE, per mezzo di un operatore che può essere il segno + (più), o @ (A commerciale), o altro simbolo.

"CONCAT" + "ENAMENTO"

⇩

"CONCATENAMENTO"

Le funzioni alfanumeriche (e l'operatore + o @) permettono di eseguire operazioni sulle parole, cioè su una serie di caratteri alfanumerici, che in BASIC possono essere di due tipi: COSTANTI ALFANUMERICHE o VARIABILI ALFANUMERICHE.

Di solito una FUNZIONE è usata con altre istruzioni, come LET o PRINT, per ASSEGNARE un valore a una variabile, o scrivere sul video.

ESEMPIO 1: LET A$=LEFT$("SCUOLA SCHEIDEGGER",6) l'istruzione LET permette di ASSEGNARE alla variabile di nome A$ il valore indicato dalla FUNZIONE alfanumerica.

L'ARGOMENTO è la COSTANTE alfanumerica "SCUOLA SCHEIDEGGER".

La funzione usata ha il nome LEFT (più il simbolo DOLLARO) e permette di PRELEVARE dei CARATTERI dall'argomento indicato.

Il numero di caratteri da prelevare è indicato dopo la virgola.

Inoltre la funzione LEFT$ indica di prelevare caratteri a partire dal primo carattere a sinistra (LEFT=SINISTRA).

L'istruzione LET di cui sopra ASSEGNERÀ la parola SCUOLA (i primi sei caratteri della costante indicata), alla variabile A$.

ESEMPIO 2: LET A$=RIGHTS("SCUOLA SCHEIDEGGER",11)
La funzione RIGHT$ opera analogamente alla funzione LEFT$, ma preleva i caratteri partendo da DESTRA (RIGHT=DESTRA), pertanto l'istruzione LET assegnerà alla variabile A$ gli ultimi 11 caratteri della costante indicata.

ESEMPIO 3: LET A$=MID$("SCUOLA SCHEIDEGGER",5,14).

La funzione MID$, preleva una serie di caratteri partendo da un certo punto della parola, che è indicato dopo la prima virgola (nell'esempio il QUINTO carattere).

La quantità di caratteri da prelevare è indicata dopo la seconda virgola (nell'esempio 14), per cui alla variabile A$ sarà assegnata la parola LA SCHEIDEGGER, cioè 14 caratteri, partendo dal quinto, e tenendo conto che anche lo SPAZIO è un CARATTERE ALFANUMERICO. Gli esempi fatti per la COSTANTE "SCUOLA SCHEIDEGGER" possono essere ripetuti per altre COSTANTI o VARIABILI. per usare le VARIABILI, come ARGOMENTO delle funzioni, occorre mettere il NOME della variabile.

La variabile indicata DEVE contenere una parola, altrimenti la FUNZIONE non effettua il prelievo, o in molti casi, il computer dà una segnalazione di errore.

ESEMPIO 4: LET A$="SCUOLA"+"SCHEIDEGGER".

L'operatore + (più), o altro simbolo specifico per ogni computer, permette di CONCATENARE "SCUOLA" con "SCHEIDEGGER".

La variabile A$ conterrà quindi la parola completa.

Il CONCATENAMENTO può essere fatto anche tra VARIABILI ALFANUMERICHE.

VERIFICARE L'EFFETTO DEI SEGUENTI ESEMPI:

```
10 CLS
20 A$="SCORRIMENTO LENTO"
30 FOR I=1 TO LEN(A$)
40 PRINT MID$(A$,I,1);
50 FOR TM=1 TO 50 : NEXT TM.
60 NEXT I
70 GOTO 10
```

```
10 CLS
20 A$="FRASE VERTICALE"
30 FOR I=1 TO LEN(A$)
40 PRINT MID$(A$,I,1)
50 FOR TM=1 TO 50 : NEXT TM
60 NEXT I
70 GOTO 10
```

```
10 CLS
20 A$="ALORAP AL ETREVNI"
30 FOR I=LEN(A$) TO 1 STEP -1
40 PRINT MID$(A$,I,1);
50 FOR TM=1 TO 50 : NEXT TM
60 NEXT I
70 GOTO 10
```

6 CHIARIRE L'USO DELLE FUNZIONI: LEN, STR$, VAL, CHR$, ASC

Parleremo ora delle funzioni di conversione ALFANUMERICA-NUMERICA, e di conversione NUMERICA-ALFANUMERICA.

Non tutti i computers adottano gli stessi NOMI per le funzioni, per cui elencheremo i nomi più comuni, usati dalla maggior parte dei "dialetti" BASIC.

LEN (arg$) è una funzione che permette di ricavare un NUMERO da un argomento (indicato con arg$) di tipo alfanumerico.

Il numero ottenuto è la LUNGHEZZA dell'argomento, cioè il numero di caratteri della costante (o variabile) alfanumerica che è argomento della funzione.

ESEMPIO: LET A=LEN("CIAO"): la variabile numerica A prende valore 4.

STR$(arg) è una funzione che permette di trasformare un NUMERO in CARATTERI (equivalenti al numero indicato).

ESEMPIO: LET A$=STR$(1234) la COSTANTE NUMERICA 1234 viene convertita in STRINGA di caratteri " 1234"; questo valore viene assegnato alla variabile ALFANUMERICA A$.

Occorre notare che il primo carattere della stringa è lo spazio.

Questo avviene per TUTTI i NUMERI POSITIVI perchè anche il segno del numero viene convertito, ed in BASIC il segno compare solo per numeri negativi (cioè quando è il segno –).

VAL(arg$) è una funzione che esegue l'operazione INVERSA rispetto alla funzione STR$ di cui sopra.

Questa funzione converte una serie di caratteri in un numero.

ESEMPIO: LET A=VAL("–12.25") alla variabile numerica A viene assegnato il valore NUMERICO –12.25 che può essere usato per calcoli, che non avremo potuto eseguire con la stringa "–12.25".

Se l'argomento alfanumerico NON contiene caratteri traducibili in numero (es: "ABCD") la funzione dà come risultato 0.

Accenniamo ora alle funzioni di conversione ASCII (pronuncia EISCHII).

Queste funzioni hanno un uso particolare che esula dagli scopi del corso per cui verranno solo accennate:

CHR$(arg) converte un numero nel CARATTERE equivalente ASCII.

ASC(arg$) converte un CARATTERE nell'equivalente numero di codice ASCII.

I caratteri ASCII (American Standard Code for Information Interchange) sono codificati secondo una TABELLA che assegna un valore NUMERICO di IDENTIFICAZIONE ad ogni segno o simbolo usati per i computer.

Anche se questa codifica dovrebbe essere valida per tutti i computer, in realtà ogni costruttore dà la sua versione, per cui ci possono essere differenze di codifica da un computer ad un altro.

Alcuni esempi (validi per la maggior parte dei computer):

IL CARATTERE "A" ha il codice (decimale) ASCII pari a 65.

Il codice 48 (decimale) indica il carattere "0".

È opportuno consultare il manuale del proprio computer, per verificare quale codifica è stata usata dal costruttore.

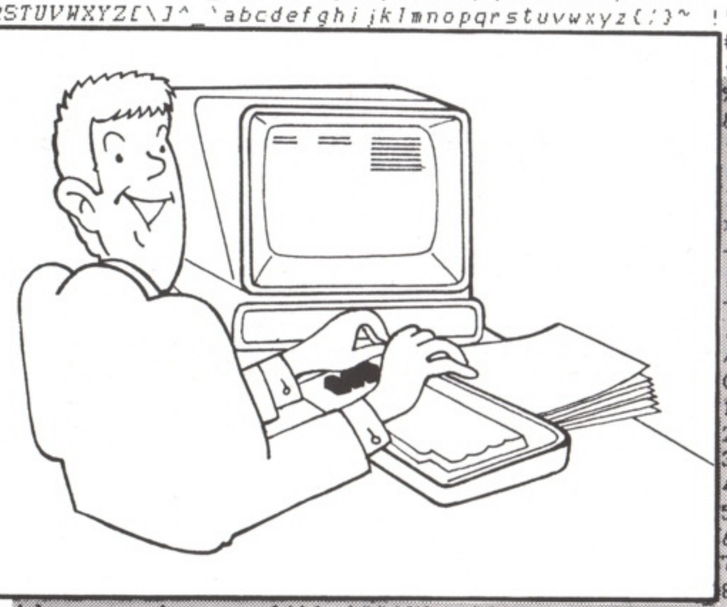
TAVOLA CODICI ASCII STANDARD

Bits b7 b6 b5 / b4 b3 b2 b1 (Colonna / Riga)	0 (0 0 0)	1 (0 0 1)	2 (0 1 0)	3 (0 1 1)	4 (1 0 0)	5 (1 0 1)	6 (1 1 0)	7 (1 1 1)
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	A	LF	SUB	*	:	J	Z	j	z
1 0 1 1	B	VT	ESC	+	;	K	[	k	{
1 1 0 0	C	FF	FS	,	<	L	\	l	¦
1 1 0 1	D	CR	GS	-	=	M	]	m	}
1 1 1 0	E	SO	RS	.	>	N	^	n	~
1 1 1 1	F	SI	US	/	?	O	—	o	DEL

NOTA

La tabella riporta la codifica binaria ed esadecimale dei codici ASCII standard.
Verificare sul manuale del proprio computer la codifica adottata.

ESERCIZI



ESERCITAZIONI E STUDI DA ESEGUIRE A CASA

1. Esercitarsi nell'uso delle istruzioni, funzioni, etc. spiegate durante la lezione.
2. COMPLETARE L'ELENCO del VOCABOLARIO BASIC (preparato come esercizio della precedente lezione), inserendo le VOCI nuove apprese in questa lezione.
3. Digitare il programma allegato, secondo le regole già viste per i programmi delle lezioni precedenti.
4. Quando il programma è corretto, e "funziona", trasferirlo su nastro, con il titolo LEZIONE 5.
5. Ricordarsi di portare il nastro ed il listato alla prossima lezione.

NOTE PER LA DIGITAZIONE DEL PROGRAMMA

Il BASIC del computer usato per realizzare il programma disponeva dell'istruzione CLS per cancellare lo schermo.

Se il vostro computer NON possiede tale istruzione, occorrerà SOSTITUIRE tutte le istruzioni CLS con l'equivalente istruzione che permette la cancellazione dello schermo.

Tale sostituzione DEVE essere eseguita in tutte le righe in cui compare l'istruzione CLS.

Controllare inoltre se il vostro computer accetta tutte le istruzioni, funzioni, etc. che compaiono nel listato.

In caso contrario consultare attentamente il manuale del vostro computer, per trovare le istruzioni equivalenti a quelle del listato e sostituitele con le istruzioni che il vostro computer accetta.


```
10 REM **      LEZIONE 5      **
15 REM ** VETTORI SUBROUTINES **
20 CLS
25 PRINT "MAGAZZINO":PRINT
30 PRINT "1-INSERIM.   2-LETTURA":PRINT
35 PRINT "3-MODIFICA   4-FINE":PRINT
40 INPUT "COSA SCEGLI";R$
50 IF R$="1" THEN M$="INSERIMENTO MAGAZZINO":GOSUB 100
60 IF R$="2" THEN M$="LETTURA MAGAZZINO":GOSUB 200
70 IF R$="3" THEN M$="MODIFICA MAGAZZINO":GOSUB 300
80 IF R$="4" THEN CLS:END
90 GOTO 10
100 REM ** INSERIMENTO **
110 FOR I=1 TO 10
120 CLS:PRINT M$:PRINT
125 PRINT "ART.";I:PRINT
130 INPUT "DESCRIZIONE ";D$(I):PRINT
135 INPUT "PREZZO ";P(I):PRINT
140 PRINT "F=FINE   A=ALTRO INSER.":PRINT
145 INPUT "COSA SCEGLI";S$
150 IF S$<>"A" AND S$<>"F" THEN 140
160 IF S$="F" THEN I=11
170 NEXT I
180 RETURN
200 REM ** LETTURA **
210 FOR I=1 TO 10
220 CLS:PRINT M$:PRINT
225 PRINT "ART.";I:PRINT
230 PRINT "DESCRIZIONE ";D$(I):PRINT
235 PRINT "PREZZO ";P(I):PRINT
240 INPUT "ALTRO ARTICOLO (S/N)";S$
250 IF S$<>"S" AND S$<>"N" THEN 240
260 IF S$="N" THEN I=11
270 NEXT I
280 RETURN
300 REM ** MODIFICA **
310 CLS:PRINT M$:PRINT
320 INPUT "ART.";I:PRINT
325 IF I<1 OR I>10 OR I<>INT(I) THEN 300
330 INPUT "DESCRIZIONE ";D$(I) :PRINT
335 INPUT "PREZZO ";P(I):PRINT
340 INPUT "ALTRO ARTICOLO (S/N)";S$
350 IF S$="S" THEN 310
360 IF S$<>"N" THEN 340
370 RETURN
```

LEZIONE 5

OBIETTIVI

1. Riprendere il concetto di variabile, per introdurre alle variabili indicizzate semplici (vettori).
2. Insegnare l'uso delle variabili con indice, associandole all'istruzione FOR... NEXT.
3. Chiarire i limiti delle variabili con indice, e spiegare l'istruzione DIM per valori di indice superiori a 10.
4. Fornire le tecniche di programmazione, relative alla presentazione di un programma, chiarendo il concetto di MENU ed introducendo alla scomposizione di un programma con più scelte.
5. Definire il concetto di sottoprogramma, e di organizzazione generale di un programma complesso.
6. Chiarire il concetto di MAIN PROGRAM.
7. Definire l'uso dell'istruzione GOSUB... RETURN
8. Finalizzare le nozioni acquisite alla stesura di un programma complesso, organizzato in MAIN e SUBROUTINES.

1 RIPRENDERE IL CONCETTO DI VARIABILE PER INTRODURRE QUELLO DI VARIABILI INDICIZZATE SEMPLICI (VETTORI)

Le variabili di cui abbiamo parlato fino ad ora sono di due TIPI: numeriche (intere e reali), ed alfanumeriche.

Se assegnamo un nome ed un valore ad una variabile, il computer colloca il valore nella memoria RAM. Per leggere il valore inserito, è sufficiente richiamarlo, usando il nome che gli avevamo dato.

Abbiamo potuto risolvere semplici problemi con l'uso delle variabili e con le istruzioni apprese fino ad ora.

Affrontando problemi più complessi, come la GESTIONE di una serie di DATI dello stesso tipo (ad esempio gli articoli di un magazzino), le variabili sino ad ora studiate non ci permettono di operare nel migliore dei modi. Vediamo quali limiti hanno, analizzando il problema relativo alla gestione di un magazzino:

1) In un magazzino ci sono molti prodotti (che vengono chiamati articoli).

2) Ogni articolo del magazzino ha un proprio nome (che viene chiamato genericamente DESCRIZIONE del prodotto).

3) Ogni articolo di magazzino ha un suo prezzo di costo.

4) Ogni articolo di magazzino può avere una o più caratteristiche di altro tipo (ad esempio Unità di Misura o altro).

5) Ogni articolo di magazzino inoltre può essere stato acquistato da un fornitore, o da un'altro.

L'elenco potrebbe continuare a lungo, ma limitiamo il nostro problema ai primi due punti: la DESCRIZIONE ed il PREZZO. Se usiamo una variabile alfanumerica, in cui inserire la DESCRIZIONE, chiamandola (ad esempio) D$, questa variabile potrà contenere SOLTANTO la descrizione di UN prodotto. Così pure usando una variabile numerica, di nome P per il Prezzo, questa variabile potrà contenere SOLTANTO il prezzo di UN prodotto.

Per un secondo articolo del magazzino avremmo bisogno di altre due variabili: una per la descrizione ed una per il prezzo. Con le variabili studiate fino ad ora (chiamate variabili semplici), dovremmo definire un NOME NUOVO per ogni variabile di DESCRIZIONE e di PREZZO degli articoli del magazzino.
Il BASIC però dispone di un altro tipo di variabili che permette di risolvere il problema: sono le VARIABILI con indice (INDICIZZATE).

Codice Prodotto	DESCRIZIONE PRODOTTI
ø	
1	
2	
3	
4	
5	
6	
7	
8	
9	
1ø	
11	
12	

Codice Prodotto	PREZZO PRODOTTI
ø	
1	
2	
3	
4	
5	
6	
7	
8	
9	
1ø	
11	
12	

2 INSEGNARE L'USO DELLE VARIABILI INDICIZZATE

Le variabili con INDICE possono essere di diversi tipi: per ora prendiamo in considerazione variabili indicizzate con un solo INDICE, che sono anche chiamate VETTORI.

Queste variabili seguono le regole, già trattate, riguardo l'ATTRIBUZIONE del NOME ed il SIMBOLO di identificazione.

Avremo quindi: VARIABILI con INDICE numeriche reali, numeriche intere ed alfanumeriche.

La differenza rispetto alle variabili SEMPLICI è la seguente:
DOPO il NOME e il SIMBOLO che identificano la variabile, c'è una parentesi rotonda che contiene un ARGOMENTO NUMERICO.
ESEMPIO: D$(1): variabile indicizzata di tipo ALFANUMERICO con nome D$

L'ARGOMENTO numerico tra parentesi si chiama INDICE, e le variabili di questo tipo si chiamano VARIABILI con INDICE, oppure VARIABILI INDICIZZATE.

ESEMPIO:
A$(1) = "LUNEDI"
A$(2) = "MARTEDI"
A$(3) = "MERCOLEDI"
A$(4) = "GIOVEDI"
A$(5) = "VENERDI"
A$(6) = "SABATO"
A$(7) = "DOMENICA"

Tutte le operazioni già studiate per le variabili semplici possono essere realizzate anche con le variabili indicizzate.

L'indice (l'argomento tra parentesi) segue le regole già viste per gli ARGOMENTI delle istruzioni complesse (es: FOR ... NEXT).

L'INDICE può essere quindi: una COSTANTE numerica, una VARIABILE o una funzione matematica; però se il VALORE assegnato all'indice NON è un NUMERO INTERO, il computer dà una segnalazione errore: TYPE MISMATCH ERROR (ERRORE per confusione di tipo), oppure ignora la parte decimale del numero.

L'INDICE di una variabile indicizzata rappresenta il NUMERO d'ordine del CONTENUTO di quella variabile.
Per chiarire: volendo inserire 10 descrizioni diverse, per gli articoli di un magazzino, potremo operare come segue: la prima descrizione la inseriremo nella variabile D$(1), cioè al PRIMO posto della variabile indicizzata; la seconda descrizione al SECONDO posto: D$(2) la terza al TERZO posto: D$(3), è così di seguito, finchè non avremo inserito tutte le nostre descrizioni.

Il problema si semplifica ancora se usiamo come INDICE una variabile numerica, e cambiamo il valore di questa variabile con un CICLO FOR ... NEXT, come è stato fatto nel programma dato come esercitazione alla precedente lezione.

3 CHIARIRE I LIMITI DELLE VARIABILI CON INDICE SUPERIORE A 10

Le variabili indicizzate sono collocate in MEMORIA RAM ed il computer riserva uno spazio per contenere i valori di queste variabili.

Però in condizioni normali lo spazio che il computer riserva è sufficiente soltanto per 11 (undici) valori diversi. Il computer è in grado di accettare solo valori compresi tra 0 e 10 come INDICE delle variabili.
Un valore di indice superiore a 10 genera una segnalazione di errore.

```
? BAD SUBSCRIT ERROR
```

È possibile fare accettare valori superiori a 10, usando una nuova istruzione BASIC.
Questa istruzione è DIM ed il suo significato è DIMENSIONAMENTO.

```
DIM
```

Con l'istruzione DIM si può dire al computer di riservare più spazio per le variabili il cui nome è indicato dopo l'istruzione DIM. I nomi delle variabili devono essere separati tra loro dalla virgola. In alcuni BASIC è prevista anche l'istruzione CLEAR, per riservare spazio alle variabili. In tal caso occorrerà definire quanta memoria destinare (es. CLEAR 5000), prima del DIMENSIONAMENTO.
Tornando al nostro esempio di magazzino:
se indichiamo con D$ il nome della variabile che contiene la descrizione degli articoli di magazzino e con P il nome della variabile che contiene il prezzo, supponendo di avere 30 articoli di magazzino, ognuno con il suo prezzo, possiamo dare al computer la seguente istruzione:

```
DIM D$(30),P(30)
```

indicando così le due variabili indicizzate D$ e P, contraddistinte da 31 possibili valori.
NOTA: i valori possibili sono 31 e NON 30 in quanto il computer considera anche il valore 0 (ZERO) come valore di INDICE, per cui sarà possibile inserire valori per tutte le posizioni comprese tra 0 e 30.

È MOLTO IMPORTANTE fare attenzione a quanto segue:
In un programma si possono usare tutte le istruzioni DIM che servono, e l'unico limite è lo SPAZIO disponibile in memoria, cioè dipende dalla capacità di memoria del computer.

Dopo aver DIMENSIONATO con DIM una variabile non è possibile eseguire un'altra volta l'istruzione DIM per quella stessa variabile. È quindi opportuno mettere le istruzioni DIM all'inizio del programma, in una RIGA di istruzioni che non venga MAI RIESEGUITA: cioè una riga a cui non si debba tornare con SALTI (condizionati o incondizionati).
Alcuni BASIC permettono di RIESEGUIRE il DIMENSIONAMENTO con l'istruzione REDIM. Gli home-computer, di norma, NON hanno tale istruzione.

4 TECNICHE DI PROGRAMMAZIONE: (MENU, SCOMPOSIZIONE PROGRAMMI)

Vediamo ora le tecniche di programmazione più importanti da seguire nella stesura di un programma complesso, che abbia come scopo la soluzione di uno o più problemi relativi ad una certa quantità di dati.

Facciamo sempre riferimento all'esempio del magazzino e dei suoi diversi articoli: è necessario considerare il problema in generale e scomporlo nelle diverse parti che hanno uno scopo ben definito:

1 Il problema GENERALE è la GESTIONE del MAGAZZINO.
Per GESTIONE si intende: TUTTE le operazioni che possono essere eseguite sui diversi articoli del magazzino.

2 Le OPERAZIONI che permettono la GESTIONE del magazzino costituiscono i problemi da risolvere e possono avere caratteristiche proprie.

Vediamo, per ora, solo alcuni di questi problemi:

a) L'INSERIMENTO della descrizione e degli altri dati, (es: il prezzo) dei diversi articoli.
b) L'AGGIORNAMENTO, cioè la possibilità di modificare uno, o più di uno dei dati di cui al punto "a".
c) La LETTURA, cioè la possibilità di consultare le diverse voci che compongono il magazzino.
d) La STAMPA su carta delle diverse voci.
e) L'ANNULLAMENTO di una o più voci.
f) Il RIORDINO delle diverse voci (per es. in ordine alfabetico).

Naturalmente quelli indicati sono soltanto alcuni dei problemi che si possono considerare ed il programma per la gestione del magazzino può considerarli tutti, o soltanto qualcuno, o può tener conto di problemi e scelte diverse.
Per il momento analizzeremo solo alcuni dei problemi indicati sopra. Come si può intuire ognuno dei problemi considerati deve poi essere sviluppato, per identificare quali sono i dati in INPUT, cioè quali dati dovranno essere inseriti, e quali CONTROLLI eseguire su questi dati.
Occorrerà anche stabilire le eventuali ELABORAZIONI da eseguire, identificando i dati in OUTPUT, cioè i risultati delle elaborazioni.

Infine ognuna di queste operazioni dovrà tenere conto delle tecniche elementari di programmazione già viste: per la presentazione dei diversi problemi sul video e la possibilità di scelta offerte all'utente.

Queste SCELTE dovranno comparire sul video all'inizio del programma, costituendo il MENU, cioè le alternative che il programma offre.

5 DEFINIRE IL CONCETTO DI SOTTOPROGRAMMA ED ORGANIZZAZIONE GENERALE DI UN PROGRAMMA

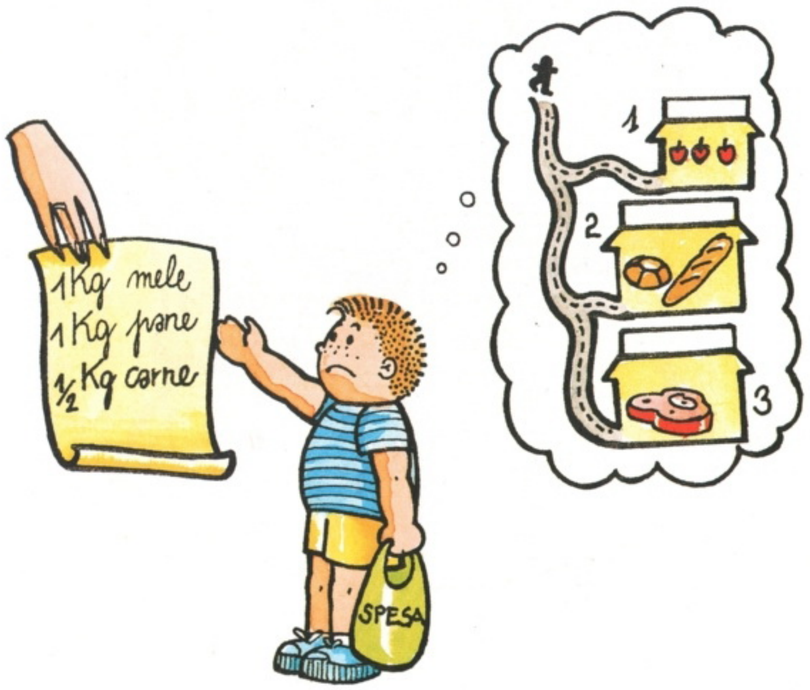
Dopo aver preso in considerazione il problema generale scomponendo le diverse parti del programma siamo in grado di analizzare le singole parti del programma per risolverne i relativi problemi.

La scomposizione eseguita permette di prendere in considerazione soltanto un problema per volta, facilitando la stesura del programma.

Possiamo immaginare il programma generale suddiviso in piccoli programmi, indipendenti tra loro, che hanno uno scopo specifico, (nostro esempio la gestione di un magazzino).

Le parti di programmi così identificate prendono il nome di SOTTOPROGRAMMI. Il termine inglese che solitamente si usa per identificare i sottoprogrammi è SUBROUTINES.

Naturalmente è necessario che le diverse SUBROUTINES di un programma facciano riferimento ad una parte di programma principale; in tale parte di programma si deciderà quale subroutine deve essere eseguita.



La parte principale del programma dovrà definire altre cose molto importanti come il DIMENSIONAMENTO delle variabili, se si usano variabili indicizzate, e la presentazione del MENU PRINCIPALE, cioè le possibilità offerte dal programma.

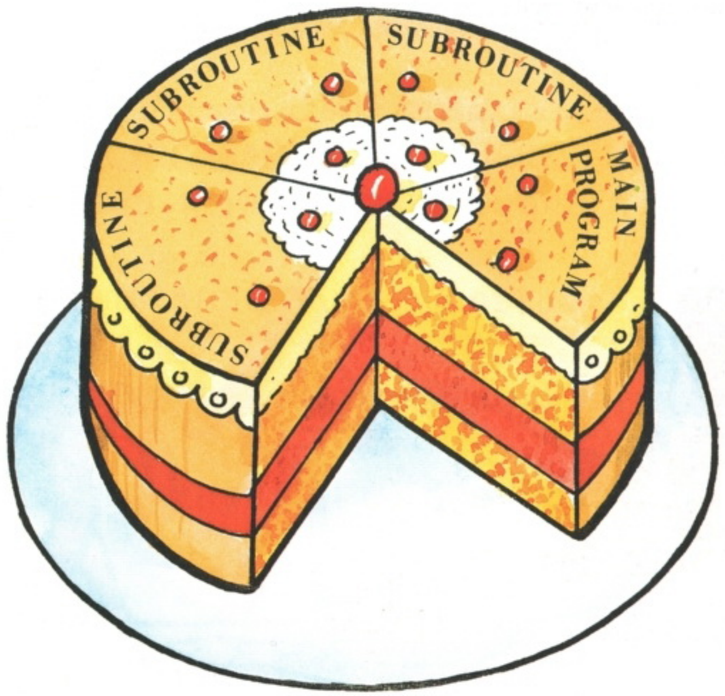
Possiamo quindi affermare che un programma complesso deve avere le caratteristiche seguenti:

1) - Avere una parte principale che serve per coordinare le diverse parti secondarie.

2) - Avere delle parti secondarie, che si chiamano SOTTOPROGRAMMI (SUBROUTINES), ognuna delle quali risolve uno specifico problema.

È importante notare che la suddivisione tra il programma principale e le subroutines è SOGGETTIVA, cioè dipende dalla capacità del programmatore di scomporre il problema in altri problemi, adottando per ognuno una possibile risoluzione.

La parte di programma che coordina le subroutines e che abbiamo chiamato PROGRAMMA PRINCIPALE, può essere chiamata con il termine inglese MAIN PROGRAM.



6 CHIARIRE IL CONCETTO DI MAIN PROGRAM

PRINCIPALE

Il MAIN PROGRAM è la parte principale di un programma complesso.

La necessità di suddividere un programma complesso in tante parti è dovuta al fatto che è più facile risolvere piccoli problemi, che affrontare un unico grande problema.

È necessario considerare come si comporta il computer dopo il comando RUN, cioè come esegue normalmente un programma.

Il computer inizia sempre dalla prima istruzione della riga che ha il numero più basso, proseguendo con tutte le istruzioni che incontra, fino all'ultima riga di programma.

Questo avviene solamente se nel programma non ci sono istruzioni di SALTO INCONDIZIONATO o SALTO CONDIZIONATO, che possono modificare la sequenza delle istruzioni.

Se abbiamo suddiviso il programma in tante parti, cioè se lo abbiamo scomposto in SOTTOPROGRAMMI, questi devono essere eseguiti secondo l'ordine che noi vogliamo.
Occorre pertanto avere una PARTE di PROGRAMMA che permetta il salto ai SOTTOPROGRAMMI.

Questa parte di programma è il MAIN PROGRAM, o programma principale.

In particolare: il MAIN PROGRAM non deve svolgere nessuna elaborazione, ma deve coordinare le SUBROUTINES.

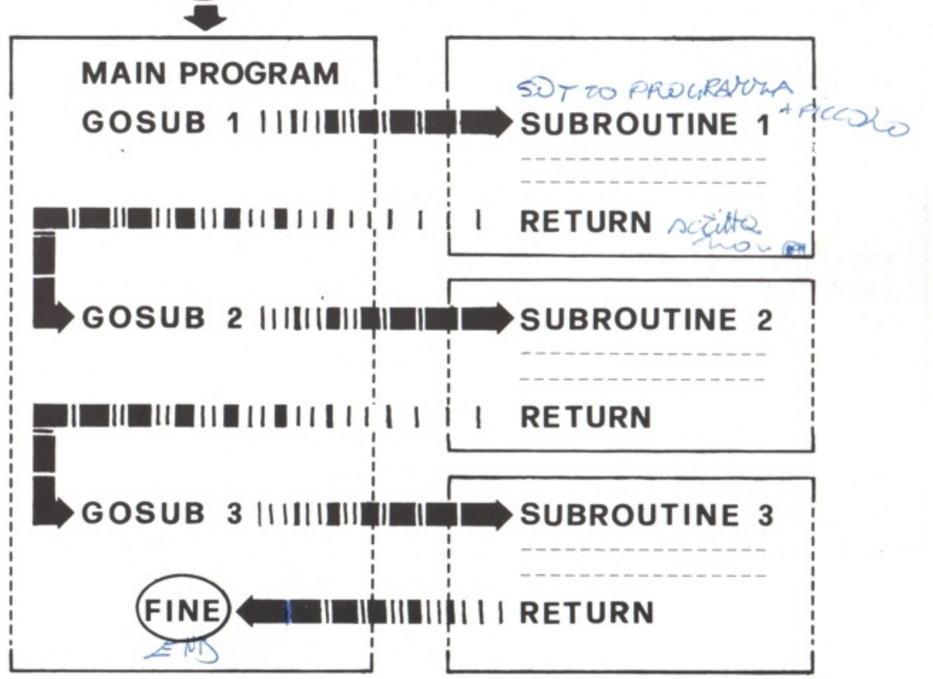
Un linguaggio di programmazione EVOLUTO consente la scomposizione di un programma in sottoprogrammi, mettendo a disposizione istruzioni complesse, per un'elaborazione veloce, e per l'organizzazione dei sottoprogrammi.

In BASIC è disponibile l'istruzione GOSUB ... RETURN

GOSUB..... RETURN

Questa istruzione complessa ha di solito la seguente sintassi: GOSUB nnn in cui la parola GOSUB significa SALTA alla SUBROUTINE, ed nnn è un numero di riga (come già visto per l'istruzione GOTO).

L'istruzione GOSUB è però diversa dall'istruzione GOTO perchè NON comanda un semplice SALTO, ma un SALTO CON RITORNO, e trasferisce l'elaborazione alla SUBROUTINE. Il RITORNO, segnalato alla fine della SUBROUTINE, consente di riprendere l'elaborazione dell'istruzione IMMEDIATAMENTE SUCCESSIVA al comando GOSUB.



7 DEFINIRE L'USO DELL'ISTRUZIONE GOSUB ... RETURN

Vediamo come opera il computer quando incontra in un programma l'istruzione GOSUBnn con un esempio:

MAIN PROGRAM

```
10 CLS
20 INPUT "INSERISCI UN NUMERO"; N1
30 INPUT "INSERISCI UN ALTRO NUMERO"; N2
40 PRINT "1- SOMMA 2-MOLTIPLICA"
50 PRINT "3-FINE"
60 INPUT "COSA SCEGLI"; S$ : IF S$<"1" OR S$ >"3" THEN 60
70 IF S$ = "1" THEN GOSUB 130
75 IF S$ = "2" THEN GOSUB 150
80 IF S$ = "3" THEN 120
90 PRINT M$; N3
100 INPUT "ALTRO CALCOLO (S/N)"; R$: IF R$ = "S" THEN 10
110 IF R$ <>"N" THEN 100
120 CLS : END
```

SUBROUTINE 1

```
130 LET N3 = N1+N2 : LET M$ = "SOMMA ="
140 RETURN
```

SUBROUTINE 2

```
150 LET N3 = N1*N2 : LET M$ = "PRODOTTO="
160 RETURN
```


Il programma è molto semplice, ma è stato scritto secondo le regole dei programmi complessi: si può identificare una parte principale, in cui avvengono le operazioni di INPUT dei dati per la scelta di una operazione, ed i sottoprogrammi che eseguono le operazioni richieste.

Il computer dopo aver controllato la validità della scelta trasferisce l'esecuzione alla SUBROUTINE relativa. Se per esempio scegliamo la SOMMA, l'esecuzione SALTA alla riga 130 questo salto viene comandato dall'istruzione GOSUB 130.

Dopo le istruzioni di assegnazione della riga 130 il programma prosegue alla riga successiva cioè la riga 140.

Nella riga 140 però c'è una istruzione RETURN.

L'istruzione RETURN (DA NON CONFONDERE CON il TASTO RETURN), indica al computer la FINE di una SUBROUTINE.

L'elaborazione RITORNA alla riga di programma IMMEDIATAMENTE SUCCESSIVA a quella che ha ordinato il salto.

Nel nostro caso il programma ritorna alla riga 75.

Poichè la variabile S$ non ha cambiato valore (è ancora "1"), vengono considerati FALSI i TEST delle righe 75 ed 80.

Alla riga 90 e seguenti il computer scrive sul video il risultato, poi chiede se si vuole fare un altro calcolo, e in base alla risposta (S/N), RIESEGUE o CHIUDE il programma.

8 FINALIZZARE LE NOZIONI ACQUISITE ALLA STESURA DI PROGRAMMA

Per comprendere meglio quanto spiegato nel corso della lezione modifichiamo il programma, dato come esercitazione alla lezione precedente, suddividendolo nel seguente modo:

1) MAIN PROGRAM che coordina le subroutines e presenta le scelte.
2) SUBROUTINES per l'INSERIMENTO, la LETTURA e la MODIFICA dei dati.

Inoltre per usare le VARIABILI con INDICE, per valori superiori a 10, scriveremo una SUBROUTINE, che chiameremo INIZIALIZZAZIONE, che permetterà di DIMENSIONARE le variabili con INDICE, per operare con un numero di articoli a nostro piacere.

Supponiamo di avere un magazzino di 100 articoli: possiamo definire per la variabile NM il valore 100, che indica il numero massimo di articoli. DIMENSIONANDO di conseguenza le variabili D$ e P con indice NM, (100 elementi per ogni variabile. Questo DIMENSIONAMENTO dovrà essere fatto all'inizio del programma, e NON dovrà essere ripetuto.
Possiamo aggiungere una riga nel MAIN program, come segue:

```
5 GOSUB 400 : REM INIZIALIZZA
```

La SUBROUTINE 400 sarà la seguente:

```
400 REM INIZIALIZZA VARIABILI
410 NM = 100 : DIM D$(NM),P(NM)
420 RETURN
```

NOTA: se il computer usato dispone dell'istruzione CLEAR, sarà necessario inserire anche la seguente riga: 3 CLEAR 1000

Le SUBROUTINES già esistenti dovranno essere modificate, per tenere conto dei seguenti fattori:

1. Gli articoli ora sono molti, per cui il computer dovrà ricordare quale è l'ultimo articolo inserito, evitando all'utente di inserire ogni volta TUTTI gli articoli partendo dal primo.
2. La LETTURA dovrà interrompersi all'ultimo articolo inserito.
3. La MODIFICA dovrà essere resa possibile SOLTANTO per gli articoli inseriti.

Tali possibilità potranno essere realizzate modificando il programma, come indicato nel listato allegato, che ha come riferimento il titolo LEZIONE 5/REV. 1.

Di seguito c'è il commento che descrive le diverse parti.

COMMENTO AL PROGRAMMA:
TITOLO PROGRAMMA: LEZIONE 5 / REV. 1
OGGETTO: GESTIONE MAGAZZINO con uso di variabili ad indice e subroutines
PARTI DEL PROGRAMMA:
MAIN PROGRAM righe da 1 a 90
SUBROUTINE per INSERIMENTO DATI: righe da 100 a 195
SUBROUTINE per LETTURA DATI: righe da 200 a 295
SUBROUTINE per MODIFICA DATI: righe da 300 a 370
SUBROUTINE per INIZIALIZZAZIONI: righe da 400 a 420

NOMI e contenuti delle VARIABILI usate nel programma:

NM = Numero Massimo di articoli del magazzino
UR = Codice del primo articolo che può essere inserito, (che viene inizializzato a 1)
D$(NM) = Descrizione degli articoli di magazzino
P(NM) = Prezzo degli articoli di magazzino
R$ = Scelta dell'utente (valori ammessi da "1" a "4")
I = Contatore per inserimento, lettura e/o modifica record

NOTA: in INSERIMENTO la variabile I assume il valore di UR;
In LETTURA e/o MODIFICA la variabile I assume il valore che viene digitato dall'utente.

S$ = Risposta utente: può contenere "S" oppure "N"
NOTA: in INSERIMENTO la variabile S$ può contenere "A" o "F"

ESECUZIONE DEL PROGRAMMA

1. Se prevista, viene eseguita l'istruzione CLEAR per riservare posto alle variabili nella MEMORIA RAM.
2. Viene eseguita la subroutine 400 di INIZIALIZZAZIONE.
3. Il MAIN PROGRAM scrive a video il MENU e permette una scelta, trasferendo l'elaborazione alla SUBROUTINE RELATIVA.
   Se la scelta non è valida, il programma ritorna alla riga 10, ripresentando il MENU con le scelte possibili.
   NOTA: Il programma ritorna al MENU anche DOPO aver eseguito la subroutine relativa ad una delle scelte possibili. La scelta "4" invece permette di terminare l'esecuzione.

Nelle SUBROUTINE sono eseguiti i controlli (sui dati in INPUT), che impediscono all'utente di commettere errori.

Tali controlli verificano costantemente la coerenza degli inserimenti, ad esempio: RIGA 120 se UR è maggiore di NM il programma salta alla riga 185 dove viene segnalato che non è più possibile inserire altri articoli (questo avviene dopo aver inserito NM articoli).

RIGA 210 e riga 310: se si cerca di leggere o modificare gli articoli, senza aver effettuato inserimenti (cioè quando UR = 1), il programma salta alla riga 280 per segnalare l'errore.

MAIN PROGRAM

```
1 REM ** LEZIONE 5 / REV.1  **
2 REM **     MAGAZZINO      **
3 CLEAR1000
5 GOSUB 400 : REM INIZIALIZZA
10 REM ** MENU E SCELTE **
20 CLS
25 PRINT "MAGAZZINO" : PRINT
30 PRINT"1-INSERIM.   2-LETTURA" : PRINT
35 PRINT"3-MODIFICA   4-FINE" : PRINT
40 INPUT"COSA SCEGLI";R$
50 IFR$="1" THEN M$="INSERIMENTO MAGAZZINO":GOSUB100
60 IFR$="2" THEN M$="LETTURA MAGAZZINO" : GOSUB200
70 IFR$="3" THEN M$="MODIFICA MAGAZZINO" : GOSUB300
80 IFR$="4" THEN CLS : END
90 GOTO 10
```

SUBROUTINE INSERIMENTO

```
100 REM ** INSERIMENTO **
110 I=UR
115 CLS : PRINT M$ : PRINT
120 IF UR>NMTHEN 185
125 PRINT"ART.";I  : PRINT
130 INPUT"DESCRIZIONE:";D$(I) : PRINT
135 INPUT"PREZZO:";P(I) : PRINT
140 PRINT"F=FINE   A=ALTRO INSER." : PRINT
145 INPUT"COSA SCEGLI";S$
150 IF S$<>"A"ANDS$<>"F"THEN 145
160 UR=UR+1
170 IF S$="F"THEN RETURN
180 GOTO100
```

SUBROUTINE SEGNALAZIONE MAGAZZINO PIENO

```
185 PRINT:PRINT"MAGAZZINO PIENO":PRINT
190 PRINT"PER AGGIUNGERE ARTICOLI":PRINT
192 PRINT"DEVI MODIFICARE NM":PRINT
194 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN194
195 RETURN
```

SUBROUTINE LETTURA

```
200 REM ** LETTURA **
205 CLS : PRINT M$ : PRINT
210 IF UR=1THEN280
220 INPUT"ART.";I  : PRINT
225 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 200
230 PRINT"DESCRIZIONE: ";D$(I) : PRINT
235 PRINT"PREZZO: ";P(I) : PRINT
240 INPUT"ALTRO ARTICOLO (S/N)";S$
250 IF S$<>"S" AND S$<>"N" THEN240
260 IF S$="N"THEN RETURN
270 GOTO 200
```


SUBROUTINE SEGNALAZIONE ERRORE

```
280 PRINT:PRINT"NESSUN ARTICOLO PRESENTE":PRINT
285 PRINT"(PRIMA DEVI INSERIRE)":PRINT
290 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN290
295 RETURN
```

SUBROUTINE MODIFICA

```
300 REM ** MODIFICA **
305 CLS : PRINT M$ : PRINT
310 IF UR=1THEN280
320 INPUT"ART.";I  : PRINT
325 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 300
330 INPUT"DESCRIZIONE:";D$(I) : PRINT
335 INPUT"PREZZO:";P(I) : PRINT
340 INPUT"ALTRO ARTICOLO (S/N)";S$
350 IF S$="S" THEN 300
360 IF S$<>"N"THEN 340
370 RETURN
```

SUBROUTINE INIZIALIZZAZIONE

```
400 REM ** INIZIALIZZA VARIABILI **
410 NM=100 : UR=1: DIM D$(NM),P(NM)
420 RETURN
```

ESERCIZI

ESERCITAZIONI E STUDI DA ESEGUIRE A CASA:

1. Esercitarsi nell'uso delle istruzioni, funzioni, etc. che sono state spiegate durante la lezione.

2. COMPLETARE l'ELENCO del VOCABOLARIO BASIC, aggiungendo le nuove VOCI apprese in questa lezione.

3. PREPARARE LO STUDIO E L'ANALISI DI UN PROGRAMMA PER LA GESTIONE DI UN MAGAZZINO.

Il Programma dovrà tenere conto di quanto segue:
- il numero massimo di articoli del magazzino deve essere 300
- Per ogni articolo di magazzino dovrà essere possibile inserire:
  a) La DESCRIZIONE del prodotto
  b) La QUANTITÀ esistente
  c) Il PREZZO unitario
  d) L'ALIQUOTA IVA %
  e) Il NOME del fornitore del prodotto.

Ognuna delle caratteristiche di cui sopra dovrà essere inserita in una variabile indicizzata.

Il programma deve offrire le seguenti possibilità:
- L'INSERIMENTO di nuovi prodotti, con le loro caratteristiche (DESCRIZIONE, QUANTITÀ, PREZZO, IVA, FORNITORE).
- La LETTURA delle caratteristiche di cui sopra.
- La MODIFICA delle caratteristiche di cui sopra.
- La VALORIZZAZIONE dell'inventario: cioè il totale delle QUANTITÀ per IL PREZZO, di tutti i prodotti esistenti.

Il programma dovrà essere organizzato con un MAIN PROGRAM e relative SUBROUTINES.

La presentazione sul video delle possibili scelte dovrà essere fatta con un MENÙ.
Il MENÙ dovrà prevedere, oltre alle scelte di cui sopra, anche la scelta di FINE PROGRAMMA.

Dovranno essere eseguiti tutti i necessari controlli sulla scelta inserita dall'utente.

Ogni parte di programma dovrà essere presentata sul video con un TITOLO, che identifichi la SCELTA dell'utente.

La SUBROUTINE per la VALORIZZAZIONE dell'inventario deve operare nel seguente modo: per ogni prodotto esistente in magazzino, deve essere calcolato il VALORE: QUANTITÀ per PREZZO UNITARIO.

Il VALORE dei singoli prodotti deve essere sommato, per ottenere il VALORE TOTALE del magazzino. Il risultato finale sarà il seguente: NR. ARTICOLI: .... VALORE TOTALE: ....

LEZIONE 6

OBIETTIVI

1. Riepilogare gli argomenti trattati, per preparare gli studenti al TEST di verifica.

2. Presentare il TEST di verifica delle nozioni acquisite.

3. Sottoporre gli studenti al TEST di verifica.

4. Fornire le risposte al TEST e commentare i risultati.

5. Consigliare gli studenti sui ripassi da fare, in base ai risultati ottenuti individualmente nella risoluzione del TEST.

1 RIEPILOGO ARGOMENTI TRATTATI IN PREPARAZIONE DEL TEST

Nel corso delle precedenti lezioni abbiamo appreso le informazioni di base sulla programmazione in generale, rapportandole al linguaggio di programmazione BASIC ed all'uso degli HOME-COMPUTER in dotazione.

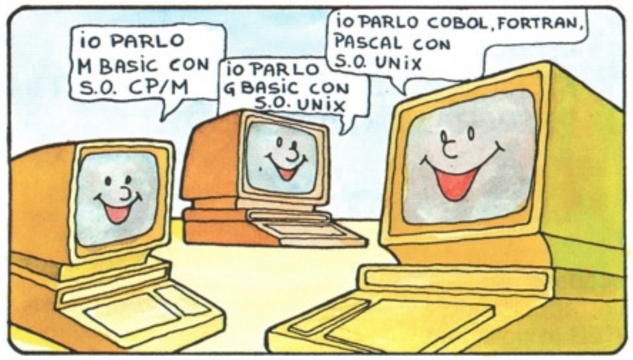
Riepiloghiamo brevemente i concetti più importanti che gli studenti devono aver approfondito con lo studio e le esercitazioni a casa.

Un sistema per l'elaborazione dati, realizzato su computer, si avvale di apparecchiature (HARDWARE) quali:
La CPU (Unità Centrale di Processo) che è il cuore del sistema.
Le PERIFERICHE che permettono di comunicare con la CPU o ricevono informazioni dalla CPU.
Per essere in grado di elaborare i dati tali apparecchiature necessitano di opportuni PROGRAMMI.

I programmi costituiscono il SOFTWARE.

I programmi che permettono di "dialogare" con la CPU costituiscono il SOFTWARE di BASE o SOFTWARE RESIDENTE, cioè il FIRMWARE.

Il SOFTWARE di BASE permette di operare con il computer, grazie al SISTEMA OPERATIVO ed ai LINGUAGGI di PROGRAMMAZIONE.



Il BASIC è un linguaggio di programmazione evoluto, e consente la scrittura e l'esecuzione di programmi di utilità generale.

Esistono diverse versioni di BASIC che differiscono tra loro, ma i concetti principali sono gli stessi, e soprattutto sono validi per qualsiasi linguaggio si programmazione.

Un programma per computer ha sempre queste caratteristiche:

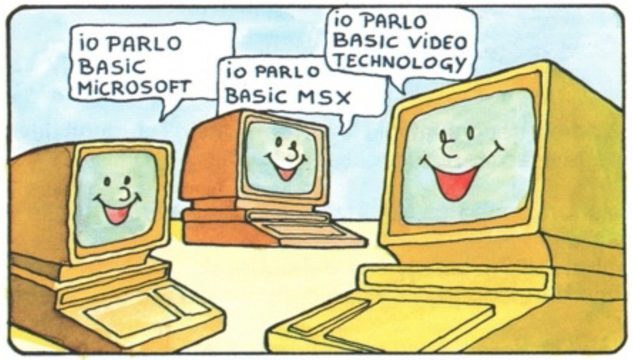
DATI in ingresso cioè INPUT.

DATI che vengono ELABORATI.

DATI in USCITA cioè OUTPUT.

Per inserire, elaborare, o ricevere dati dalla memoria centrale i linguaggi di programmazione mettono a disposizione del programmatore numerose istruzioni.

Le istruzioni sono diverse da un linguaggio di programmazione all'altro; in alcuni casi possono essere diverse anche per diverse versioni dello stesso linguaggio.



Ogni linguaggio ha una propria SINTASSI, e le istruzioni devono essere scritte in un modo preciso, rispettando le regole del linguaggio usato.

Il BASIC trattato sino ad ora ha le regole e la sintassi che sono state spiegate ed approfondite con esempi ed esercitazioni.

Le conoscenze, di carattere generale fornite dal corso, permetteranno agli studenti di poter operare in futuro, anche su computer diversi da quelli usati durante le lezioni.

2 PRESENTARE IL TEST DI VERIFICA DELLE NOZIONI ACQUISITE

In questa lezione potremo verificare le conoscenze acquisite.

Il miglior modo di verifica è la scrittura di un programma complesso, in BASIC.

Lo studio e l'analisi del programma è già stato realizzato come esercitazione a casa della precedente lezione.

Ripresentiamo il problema, aggiungendo informazioni di carattere generale.

Il programma deve permettere la GESTIONE di un magazzino di prodotti (che chiameremo ARTICOLI).

Ogni ARTICOLO è caratterizzato dai seguenti DATI:

1 - Una DESCRIZIONE.

2 - Una QUANTITÀ esistente.

3 - Un PREZZO unitario.

4 - Una aliquota IVA percentuale.

5 - il NOME del FORNITORE di quel prodotto.

Il programma dovrà offrire le seguenti possibilità (SCELTE):

1 - L'INSERIMENTO delle caratteristiche di ogni articolo.

2 - La LETTURA delle caratteristiche dei prodotti inseriti.

3 - La possibilità di MODIFICARE le caratteristiche di cui sopra.

4 - La VALORIZZAZIONE del magazzino: TOTALE del PREZZO, moltiplicato per la QUANTITÀ di tutti gli articoli inseriti.

```
GESTIONE MAGAZZINO

1 - INSERIMENTO
2 - LETTURA
3 - MODIFICA
4 - VALORIZZAZIONE
5 - FINE
```

La struttura del programma deve essere organizzata nel modo seguente:

Ci deve essere un MAIN PROGRAM (Programma Principale) che coordina i sottoprogrammi, cioè le SUBROUTINES.

Il programma deve essere scritto tenendo conto delle TECNICHE DI PROGRAMMAZIONE spiegate per la presentazione del MENÙ iniziale, oltre alle elementari tecniche relative alla pulizia del video, la presentazione dei TITOLI, ed il CONTROLLO dei dati in INPUT.

Per rendere il programma più razionale occorrerà usare le VARIABILI con INDICE, nel seguente modo:

variabile D$(300) per le DESCRIZIONI

variabile Q$(300) per le QUANTITÀ

variabile P$(300) per i PREZZI

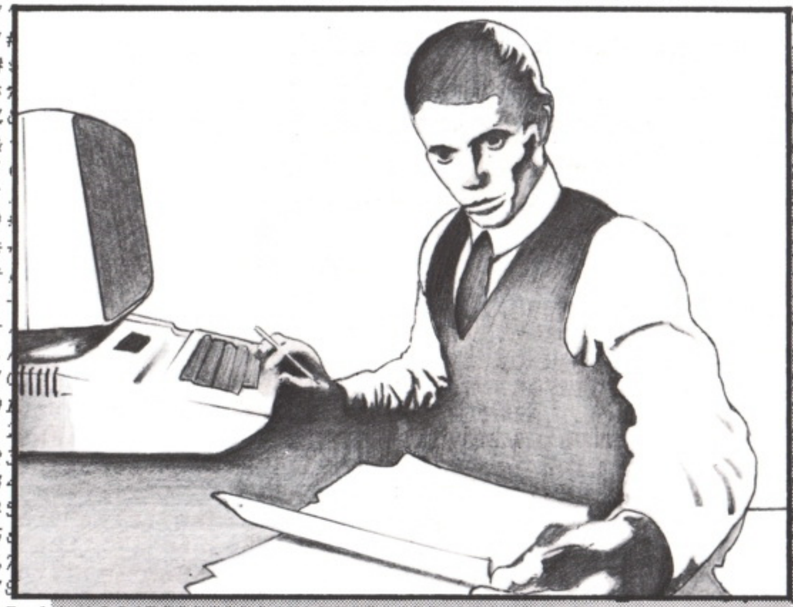
variabile IV$(300) per l'IVA

variabile NF$(300) per il Nome del Fornitore.

L'uso di VARIABILI ALFANUMERICHE permetterà di evitare la segnalazione REDO FROM START.

I valori usati per calcoli (PREZZO e QUANTITÀ) nella SUBROUTINE di VALORIZZAZIONE, dovranno essere convertiti nel modo opportuno con la funzione VAL (arg$).

ESERCIZI



ESERCITAZIONI E STUDI DA ESEGUIRE A CASA:

1) Ripassare tutte le istruzioni, funzioni, etc. apprese fino ad ora.

2) Digitare il programma MAGAZZINO LEZIONE 7 come da listato allegato.

   NOTA: Il BASIC del computer usato per scrivere il programma disponeva dell'istruzione CLS per la cancellazione dello schermo, e richiedeva l'istruzione CLEAR per riservare spazio per le variabili.

   Se il vostro computer NON possiede l'istruzione CLS occorrerà sostituire l'istruzione CLS, che compare nel listato con l'equivalente istruzione per cancellare lo schermo.

   Se il vostro computer NON ha l'istruzione CLEAR (da non confondere con l'istruzione CLR), occorrerà NON tenere conto dell'istruzione CLEAR presente nel listato.

Il programma allegato è una nuova versione per la GESTIONE MAGAZZINO.

In tale programma vengono usate le VARIABILI con più INDICI (che vengono anche chiamate TABELLE).

Inoltre viene eseguita l'operazione di RIORDINO degli articoli di magazzino, con una apposita SUBROUTINE.

Nel programma compaiono anche le istruzioni READ, DATA e RESTORE.

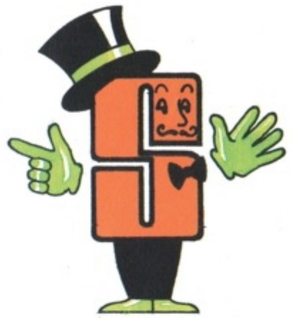
Tutti gli argomenti di cui sopra saranno spiegati nel corso della prossima LEZIONE 7, per cui si raccomanda di portare il listato ed il programma (registrato su nastro) alla prossima lezione.

ESERCITAZIONI A CASA - LISTATO PROGRAMMA (Prima parte)

```
1 REM ** MAGAZZINO LEZIONE 7 **
2 REM **                      **
3 CLEAR1000
5 GOSUB 400 : REM INIZIALIZZA
10 REM ** MENU E SCELTE **
20 CLS
25 PRINT "MAGAZZINO" : PRINT
30 PRINT"1-INSERIM.   2-LETTURA" : PRINT
33 PRINT"3-MODIFICA   4-FINE PRG" : PRINT
35 PRINT"5-TOTALIZZA  6-RIORDINO" : PRINT
40 INPUT"COSA SCEGLI";R$
50 IFR$="1" THEN M$="INSERIMENTO MAGAZZINO":GOSUB190
60 IFR$="2" THEN M$="LETTURA MAGAZZINO" : GOSUB200
70 IFR$="3" THEN M$="MODIFICA MAGAZZINO" : GOSUB300
75 IFR$="4" THEN CLS : END
80 IFR$="5" THEN M$="TOTALIZZAZIONE" : GOSUB500
85 IFR$="6" THEN M$="RIORDINO" : GOSUB600
90 GOTO 10
100 REM ** INSERIMENTO **
110 I=UR
115 CLS : PRINT M$ : PRINT
120 IF UR>NMTHEN 185
125 PRINT"ART.";I   : PRINT
130 FOR J = 1 TO NC : PRINT N$(J); : INPUT D$(I,J) : PRINT
135 NEXTJ : PRINT
140 PRINT"F=FINE   A=ALTRO INSER." : PRINT
145 INPUT"COSA SCEGLI";S$
150 IF S$<>"A"ANDS$<>"F"THEN 145
160 UR=UR+1
170 IF S$="F"THEN RETURN
180 GOTO100
185 PRINT:PRINT"MAGAZZINO PIENO":PRINT
190 PRINT"PER AGGIUNGERE ARTICOLI":PRINT
192 PRINT"DEVI MODIFICARE NM":PRINT
194 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN194
195 RETURN
200 REM ** LETTURA **
205 CLS : PRINT M$ : PRINT  .
210 IF UR=1THEN280
220 INPUT"ART.";I : PRINT
225 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 200
230 FOR J = 1 TO NC :PRINT N$(J);D$(I,J) : PRINT:NEXT J : PRINT
240 INPUT"ALTRO ARTICOLO (S/N)";S$
250 IF S$<>"S" AND S$<>"N" THEN240
260 IF S$="N"THEN RETURN
270 GOTO 200
280 PRINT:PRINT"NESSUN ARTICOLO PRESENTE":PRINT
285 PRINT"(PRIMA DEVI INSERIRE)":PRINT
290 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN290
295 RETURN
```

ESERCITAZIONI A CASA - LISTATO PROGRAMMA (Seconda parte)

```
300 REM ** MODIFICA **
305 CLS : PRINT M$ : PRINT
310 IF UR=1THEN280
320 INPUT"ART.";I  : PRINT
325 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 300
330 FOR J=1TONC:PRINTN$(J);:INPUT D$(I,J) : PRINT
335 NEXT J : PRINT
340 INPUT"ALTRO ARTICOLO (S/N)";S$
350 IF S$="S" THEN 300
360 IF S$<>"N"THEN 340
370 RETURN
400 REM ** INIZIALIZZA VARIABILI **
410 NM=100 :  NC=5 : UR=1 : DIM D$(NM,NC),N$(NC)
420 RESTORE:FOR I=1TONC:READ N$(I):NEXT
430 DATA "DESCRIZIONE  : "
431 DATA "QUANTITA'... : "
432 DATA "PREZZO ..... : "
433 DATA "IVA % ..... : "
434 DATA "FORNITORE .. : "
450 RETURN
500 REM * TOTALIZZA INVENTARIO *
505 CLS : PRINT M$ : PRINT
510 T=0:FOR I = 1 TO UR-1
530 Q=VAL(D$(I,2)): P=VAL(D$(I,3))
540 T = T + Q*P
550 NEXT I : PRINT
560 PRINT"ARTICOLI PRESENTI : ";UR-1 : PRINT
570 PRINT"VALORE TOTALE =";T : PRINT
580 INPUT"TORNO AL MENU' (S/N)";S$:IFS$<>"S"THEN 500
590 RETURN
600 REM * RIORDINO *
605 CLS : PRINT M$ : PRINT
610 IF UR=1THEN280
620 PRINT"ATTENDERE MENTRE RIORDINO":PRINT
630 FOR!CMD=1TO UR-2
635 FOR J=I+1 TO UR-1
640 IF D$(I,1)>D$(J,1)THEN GOSUB 700
650 NEXT J,I
655 PRINT"FINE RIORDINO":PRINT
660 FOR TM=1 TO 2000: NEXT
670 RETURN
700 REM * INVERSIONI PER RIORDINO *
710 FOR K=1 TO NC
720 D$(I,0)=D$(I,K):D$(I,K)=D$(J,K):D$(J,K)=D$(I,0)
730 NEXT
740 RETURN
```



LEZIONE 7

OBIETTIVI

1. Fornire i criteri per la stesura di un programma impostato in modo professionale.
2. Chiarire l'uso dell'istruzione GET (o INKEY$).
3. Introdurre al concetto di dati, attraverso semplici esempi.
4. Spiegare le istruzioni READ, DATA e RESTORE.
5. Spiegare le variabili con più indici (TABELLE).
6. Insegnare le tecniche elementari per il riordino di tabelle.

1 FORNIRE CRITERI PER LA SCRITTURA DI PROGRAMMI PROFESSIONALI

Un programma per computer è costituito da numerose istruzioni, semplici o complesse, che consentono di risolvere un problema.

Per ottenere dei risultati è però indispensabile conoscere i diversi aspetti del problema ed i metodi risolutivi.

Un programma di tipo professionale non si limita a risolvere il problema, ma lo risolve nel MODO MIGLIORE.

La differenza tra un programma dilettantistico ed un programma professionale NON riguarda il PROBLEMA che viene risolto, ma il MODO in cui viene affrontato e risolto il problema.

Esistono programmi professionali che trattano argomenti quali i DIVERTIMENTI (VIDEOGIOCHI) o altro (TOTOCALCIO, SCACCHI, etc.), ed esistono programmi scritti in modo dilettantistico per risolvere problemi complessi quali la gestione dei dati, fatturazione, trattamento della parola (WORD PROCESSING), o anche argomenti di svago o divertimento (GIOCHI).

Come detto sopra: la DIFFERENZA non sta nell'ARGOMENTO trattato dal programma, ma nel fatto che il programma professionale oltre a risolvere il problema (che è l'argomento principale del programma), risolve anche numerosi aspetti collaterali.

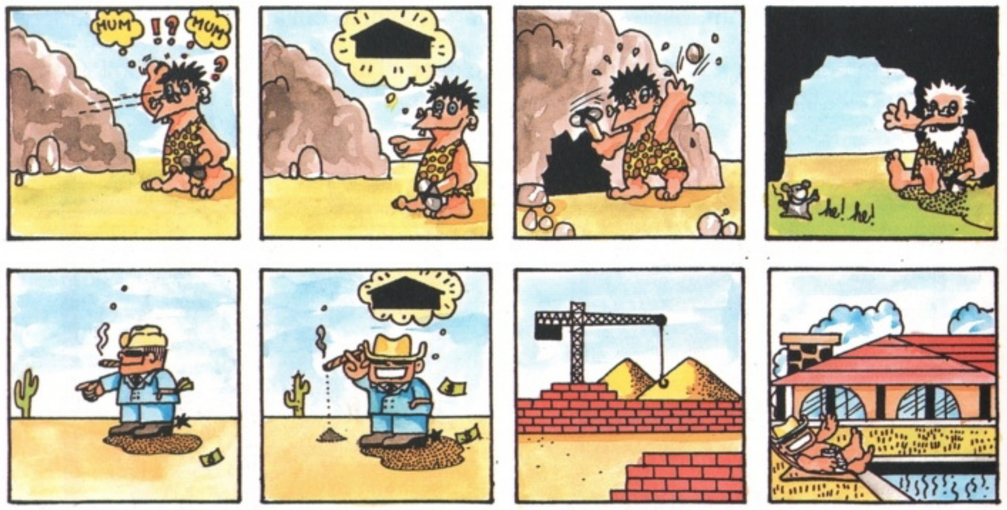
Un programma professionale è quindi più complesso di un programma dilettantistico, anche se l'argomento trattato è lo stesso.

Nel corso delle lezioni sono state insegnate alcune TECNICHE di PROGRAMMAZIONE che riguardano i programmi professionali:

1. PRESENTAZIONE del programma e del MENÙ.
2. IDENTIFICAZIONE degli INPUT e degli OUTPUT in modo chiaro.
3. CONTROLLO degli INPUT.

Altre tecniche di tipo professionale invece riguardano la scrittura del programma:

1. SCOMPOSIZIONE del programma in BLOCCHI (MAIN e SUBROUTINES).
2. INSERIMENTO DI COMMENTI nel PROGRAMMA (con istruzioni REM).
3. USO RAZIONALE delle VARIABILI.
4. USO RAZIONALE delle possibilità offerte dal linguaggio.



Un programma professionale consente all'UTENTE di operare senza difficoltà (purchè conosca la tastiera del computer ed il programma in generale).

Il listato di un programma professionale può essere letto,e compreso facilmente da chi conosca il linguaggio di programmazione ed il problema generale.

2 CHIARIRE L'USO DELLA FUNZIONE INKEY$

Per consentire il controllo dei dati in ingresso (INPUT), il linguaggio BASIC mette a disposizione una FUNZIONE.

Il NOME di tale FUNZIONE ed il suo MODO di OPERARE può variare da un computer all'altro, per cui spiegheremo quale operazione svolge, indicando i nomi usati più frequentemente nel BASIC.

INKEY$

Uno dei NOMI è INKEY$, ma alcuni computers chiamano questa funzione GET oppure GET-KEY$. L'operazione svolta è la lettura del BUFFER della tastiera.
Il BUFFER della tastiera è la zona della MEMORIA RAM in cui sono messi i CODICI ASCII dei tasti che vengono premuti sulla tastiera.

La FUNZIONE INKEY$ converte tali valori nel corrispondente carattere, per cui è possibile ASSEGNARLO ad una variabile alfanumerica con l'istruzione LET.
ESEMPIO: 10 LET A$ = INKEY$
L'istruzione GET invece assegna immediatamente il valore alla variabile, il cui nome è posto dopo la parola GET
ESEMPIO: 10 GET A$
La variabile A$ contiene il CARATTERE relativo all'ultimo tasto premuto.
La funzione INKEY$ associata alla istruzione LET può essere usata al posto dell'istruzione INPUT, per ottenere il controllo COMPLETO dell'INPUT, verificando ogni TASTO che viene premuto sulla tastiera dall'utente.

Naturalmente i controlli dovranno essere fatti con uno o più TEST, usando l'istruzione IF ... THEN ...

```
ESEMPIO: 10 PRINT "PREMI UN TASTO QUALSIASI"
         20 LET A$ = INKEY$ : IF A$ = "" THEN 20
         30 PRINT "HAI PREMUTO IL TASTO"; A$
         40 END
```

L'istruzione LET A$ = INKEY$ associata al test IF A$ = "" THEN 20 permette di interrompere l'esecuzione del programma finchè l'utente non preme un tasto.

NOTA: Il valore "" indica NESSUN VALORE, (se non viene PREMUTO NESSUN TASTO).
La riga 20 deve essere interpretata nel seguente modo: ASSEGNA alla variabile alfanumerica A$ il valore corrispondente al codice del tasto premuto sulla tastiera.
SE la variabile A$ ha valore NULLO (se non si preme alcun tasto) ALLORA torna ad eseguire la riga 20.

3 INTRODURRE AL CONCETTO DI DATI CON SEMPLICI ESEMPI

Per completare le conoscenze della programmazione in generale, e del BASIC in particolare, è necessario chiarire alcuni punti, riguardo l'elaborazione dei dati.

Il programma per la gestione del magazzino che è stato realizzato permetteva la manipolazione di una serie di DATI.

Occorre che sia chiaro cosa si intende per DATI, per comprendere le possibilità offerte dal linguaggio per MANIPOLARE (cioè GESTIRE) questi dati.

I DATI sono INFORMAZIONI che l'utente o il programmatore hanno interesse a conservare nella memoria del computer, o su una memoria di massa.

Le informazioni di uno stesso tipo possono essere raggruppate, analizzate, o manipolate, secondo le esigenze proprie.

Ad esempio considerando delle persone, potremo classificare le caratteristiche fisiche di tali persone come DATI SOMATICI.

Queste caratteristiche, cioè i DATI SOMATICI delle persone, potranno poi essere classificate in molti modi.

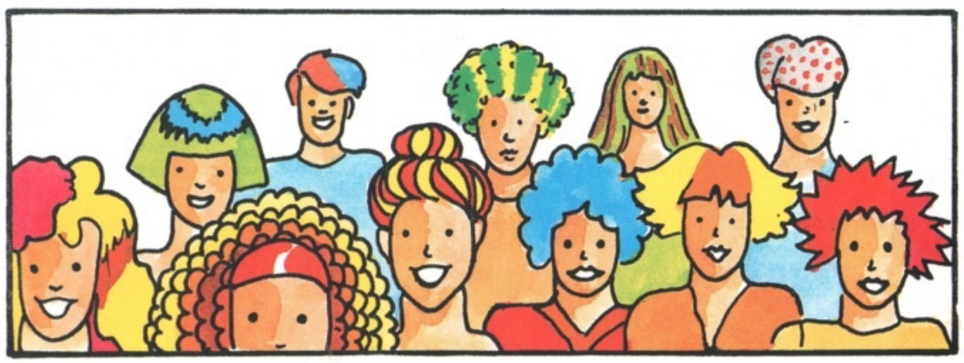
ESEMPIO: potremmo avere una classificazione di questo tipo:

DATI SOMATICI delle PERSONE:
1 - COLORE DEI CAPELLI
2 - ALTEZZA
3 - PESO
4 - FORMA DEL VISO
5 - COLORE DEGLI OCCHI

ognuna di queste caratteristiche sarà classificata secondo un CRITERIO ANALITICO INDIVIDUALE, o GENERALE.

1 – (INTERPRETAZIONE GENERALE)
a) CASTANI b) NERI c) ROSSI d) BIONDI

oppure:
2 – (INTERPRETAZIONE PERSONALIZZATA)
a) MARRONI b) NERI c) COLOR CAROTA d) GIALLI



NATURALMENTE LA PRIMA INTERPRETAZIONE È COMPRENSIBILE A CHIUNQUE, LA SECONDA È PIÙ DIFFICILE DA CAPIRE.

Il primo passo per l'elaborazione DATI è una classificazione in gruppi omogenei, facilmente identificabili.

4 SPIEGARE LE ISTRUZIONI READ, DATA E RESTORE

I DATI possono essere distinti in due tipi:

1) DATI che interessano al programmatore per la stesura del programma.
2) DATI che interessano l'UTENTE, cioè DATI che vengono ELABORATI.

Il BASIC permette di lavorare con grandi quantità di dati, sia per il primo, che per il secondo scopo. L'unico limite è dato dalle caratteristiche del computer usato, per CAPACITÀ di MEMORIA e VELOCITÀ di ELABORAZIONE.

La CAPACITÀ di MEMORIA limita la quantità di dati che si possono manipolare; la velocità di elaborazione pone il problema TEMPO di elaborazione necessario per ottenere i risultati richiesti.

Con l'uso delle variabili (semplici o indicizzate), è possibile conservare DATI nella memoria; però le variabili servono principalmente per la MANIPOLAZIONE dei dati.

DATA

Il BASIC dispone di una istruzione per conservare DATI nella memoria, senza usare le variabili. QUESTA ISTRUZIONE si chiama DATA (pronuncia: DEITA). L'uso dell'istruzione DATA è molto semplice: è sufficiente scrivere il numero di RIGA, seguito dalla parola DATA, e dai dati che si vogliono depositare. I dati devono essere separati tra loro dal segno VIRGOLA.

ESEMPIO: 10 DATA 1250, 33.25, 3970, CARLO, 3.14, GIUSEPPE

La riga del programma definisce una serie di dati numerici ed alfanumerici.
Occorre chiarire i seguenti punti:

1 I numeri che hanno una parte decimale devono essere scritti con il PUNTO DECIMALE, (come richiesto dal BASIC).

2 Le parole, cioè i dati ALFANUMERICI possono essere scritti senza virgolette.

3 Se un DATO ALFANUMERICO deve contenere degli spazi, o il segno VIRGOLA, è INDISPENSABILE usare le virgolette.

ESEMPIO: 10 DATA "SCUOLA SCHEIDEGGER VIA CASTELNUOVO, 2", COMO

Il DATO SCUOLA SCHEIDEGGER ... etc. è messo tra virgolette perchè contiene degli spazi ed il segno virgola; il dato COMO non necessita delle virgolette.

L'istruzione DATA permette di depositare dati all'interno di un programma BASIC.
Quando il computer incontra una riga con l'istruzione DATA non esegue alcuna operazione; il programma prosegue, come già visto per l'istruzione REM.
Per usare i dati indicati dall'istruzione DATA è necessario usare un'altra istruzione: READ (che significa LEGGI).

L'istruzione READ, associata al nome di UNA o PIÙ variabili legge uno o più DATI, assegnandone il valore alla variabile (o ALLE variabili) indicata dopo la parola READ.

ESEMPIO: 10 DATA 375, LUNEDI
20 READ A,A$

READ

L'istruzione DATA definisce due DATI; l'istruzione READ li legge, assegnando il primo alla variabile numerica A, e il secondo alla variabile alfanumerica A$.
Un errore nel tipo di variabile viene segnalato dal computer SOLTANTO se si LEGGE un valore ALFANUMERICO e lo si assegna ad una variabile NUMERICA.

ESEMPIO DI ISTRUZIONE ERRATA: 10 DATA 375, LUNEDI: READ A$,A
Nell'esempio l'istruzione READ legge il primo dato (cioè 375), e lo assegna alla variabile ALFANUMERICA A$ senza alcun errore, poi legge il secondo dato (cioè LUNEDI), e trovando incongruenza tra il TIPO della variabile (numerica) ed il dato, segnala l'errore.
È necessario fare delle prove con il proprio computer per verificare il modo di operare delle istruzioni READ e DATA, in quanto NON tutti i computer SI COMPORTANO COME INDICATO SOPRA, anche se quanto indicato è l'interpretazione più comune.

Ogni volta che viene eseguita l'istruzione READ il computer LEGGE un dato e lo assegna alla variabile indicata, posizionandosi sul DATO successivo.
Una nuova istruzione READ permetterà di leggere il prossimo dato, perchè il computer si ricorda quali dati ha già letto.

Se si effettua una lettura (con READ), dopo che sono stati letti TUTTI i dati il computer segnala con un messaggio (OUT OF DATA ERROR) di non avere più DATI da leggere.

Il NUMERO di DATI deve essere SEMPRE UGUALE al NUMERO di LETTURE.

OUT OF DATA ERROR

Per RILEGGERE dall'inizio i dati occorre indicare al computer di posizionarsi ancora sul primo dei dati.

Il BASIC permette tale operazione con l'istruzione RESTORE.

ESEMPIO: 10 DATA 375, LUNEDI : READ A,A$
20 RESTORE : READ B,B$

Alla riga 10 vengono definiti i dati e l'istruzione READ assegna alle variabili A ed A$ i dati letti.

Alla riga 20 l'istruzione RESTORE riposiziona sul primo dato (375) e l'istruzione READ legge i due valori, assegnandoli alle variabili B e B$.

RESTORE

Con le istruzioni DATA, READ e RESTORE è possibile conservare e leggere DATI che sono nel PROGRAMMA.
Questi dati sono molto utili al programmatore, per semplificare la programmazione.

Solitamente i DATI interni al programma (letti con READ), vengono assegnati a VARIABILI INDICIZZATE.

Leggendo il LISTATO del programma MAGAZZINO LEZIONE 7 (esercitazione della lezione 6) è possibile verificare l'applicazione delle istruzioni READ, DATA e RESTORE, con il CICLO FOR ... NEXT e con le variabili indicizzate.

COMMENTIAMO LE RIGHE DI PROGRAMMA RELATIVE:

RIGA 400 inizio subroutine di INIZIALIZZAZIONE del programma.

RIGA 410 vengono definite le variabili numeriche NM, NC e UR, assegnando a ognuna un VALORE (NM = 100, NC = 5, UR = 1), inoltre sono definite le variabili con INDICE D$ (NM, NC) e N$(NC)

NOTA: La variabile D$(NM,NC) è una variabile con DUE INDICI, il cui uso sarà spiegato in seguito.

RIGA 420 l'istruzione RESTORE permette di POSIZIONARSI sul PRIMO dei dati (indicati alle righe DATA).

L'istruzione FOR I=1 TO NC inizia un LOOP per tutti i valori di I da 1 fino a 5 in quanto NC = 5. L'istruzione READ N$(I) viene eseguita quindi 5 volte. La prima volta il computer LEGGERÀ il PRIMO DATO ASSEGNANDOLO alla VARIABILE N$(1), in quanto l'INDICE tra parentesi (cioè la variabile I), ha valore 1.

L'istruzione NEXT fa RIPETERE il LOOP per cui la variabile I diventerà 2 e l'istruzione READ N$(I) leggerà il SECONDO dato, assegnandolo alla variabile N$(2).

Il LOOP verrà ripetuto per tutti i valori compresi tra 1 e 5, per cui i CONTENUTI della variabile N$(I), alla FINE del LOOP saranno i seguenti:

N$ (1) conterrà il DATO "DESCRIZIONE :" (primo dato alla RIGA 430)
N$ (2) conterrà il DATO "QUANTITÀ :" (secondo dato)
N$ (3) conterrà il DATO "PREZZO :" (terzo dato)
N$ (4) conterrà il DATO "IVA % :" (quarto dato)
N$ (5) conterrà il DATO "FORNITORE :" (quinto dato)

Le RIGHE da 430 a 434 contengono i DATI di cui sopra, identificati con l'istruzione DATA come già spiegato.

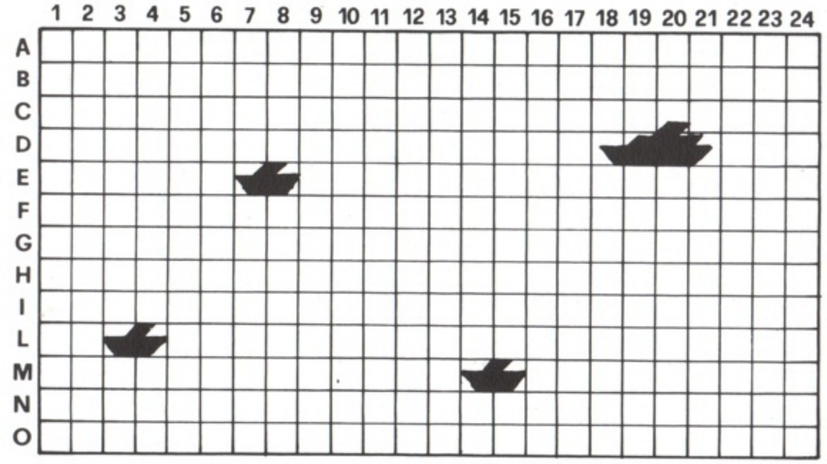
Dopo aver trasferito i DATI nella VARIABILE N$(I) sarà possibile usare tali DATI nel corso del programma.

5 SPIEGARE LE VARIABILI CON PIÙ INDICI (TABELLE)

Nel programma MAGAZZINO LEZIONE 7 è stata inserita la VARIABILE INDICIZZATA D$(NM,NC), cioè una variabile con DUE INDICI.

Il BASIC permette di definire, oltre alle variabili indicizzate semplici (chiamate VETTORI), anche delle variabili indicizzate con diversi INDICI (che sono solitamente chiamate TABELLE).

L'uso delle variabili a più indici è analogo a quello dei VETTORI, solo che, al posto di un INDICE, se ne devono indicare DUE o più.
La variabile D$(NM,NC) di cui sopra è una TABELLA costituita da ELEMENTI (VALORI che può contenere), organizzati su DUE DIMENSIONI.
Per capire tale organizzazione basta pensare alla TAVOLA PITAGORICA, o al GIOCO della BATTAGLIA NAVALE.

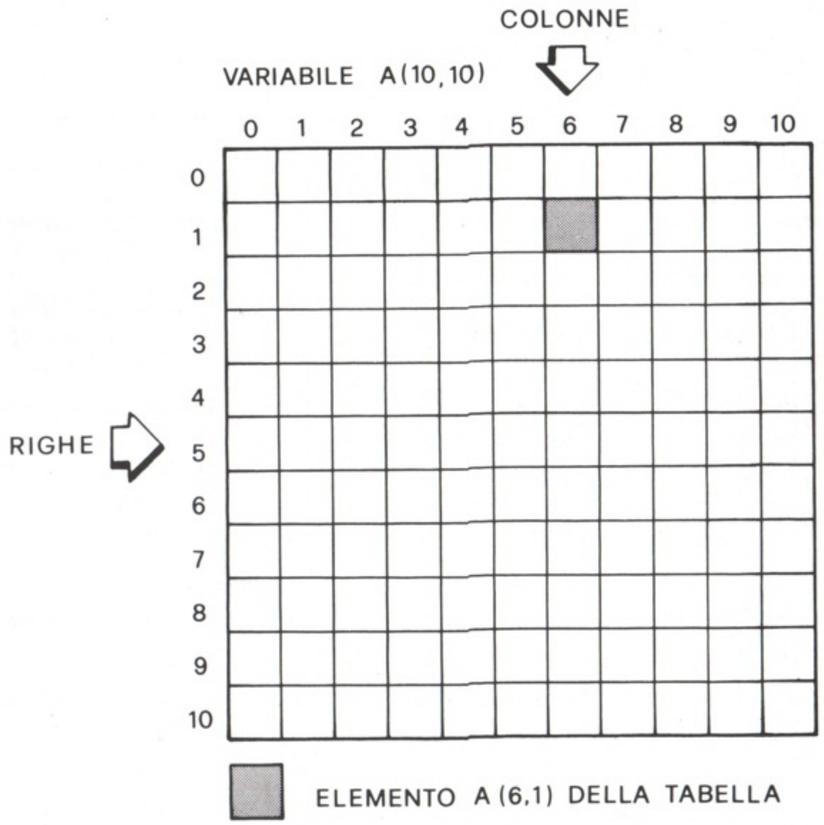


Nella TAVOLA PITAGORICA i numeri sono disposti su righe e colonne. Nel punto di incontro tra una riga ed una colonna si trova il prodotto tra i numeri scritti sulla prima riga e prima colonna.

Analogamente, nel gioco della BATTAGLIA NAVALE i giocatori dispongono su una tabella la propria "flotta" e la posizione dei pezzi è indicata da un NUMERO (RIGA) ed una lettera (COLONNA), o viceversa.

Una variabile a DUE DIMENSIONI può contenere tanti elementi in relazione agli INDICI che ci sono tra parentesi. Il numero TOTALE degli elementi è dato dal prodotto degli INDICI, ricordando che il computer tiene conto anche del valore 0 (ZERO).

ESEMPIO: DIM A(10,10) variabile a DUE DIMENSIONI che può contenere 121 elementi (cioè 10+1 per 10+1)
DIM A$(99,9,4) variabile a TRE DIMENSIONI che può contenere 5000 elementi (cioè 99+1 per 9+1 per 4+1)



Solitamente le TABELLE sono usate con l'istruzione FOR ... NEXT per eseguire le operazioni in modo veloce e razionale.

Il valore degli INDICI viene cambiato per mezzo di variabili.

Nel programma MAGAZZINO LEZIONE 7 è possibile vedere tale applicazione, che viene commentata qui di seguito, per la SUBROUTINE INSERIMENTO.

COMMENTO DELLA SUBROUTINE INSERIMENTO del programma MAGAZZINO LEZIONE 7.

La SUBROUTINE di INSERIMENTO, che inizia con la riga 100, e termina alla riga 180 esegue le seguenti operazioni:

100 INIZIO SUBROUTINE

110 La variabile I assume il valore della variabile UR (all'inizio del programma tale valore è 1)

115 Cancella lo schermo e scrive il titolo

120 CONTROLLO: se il numero di articolo da inserire è MAGGIORE del numero massimo di ELEMENTI disponibili (NM che ha valore 100), il programma SALTA alla riga 185 dove è segnalato che il MAGAZZINO è pieno, per cui non è possibile effettuare altri inserimenti.

125 scrive sul video il numero dell'articolo e lo identifica con la parola ART.

130 Inizio di un LOOP per la variabile J e per i valori da 1, fino a 5 (NC ha il valore di 5); le istruzioni PRINT N$(J), INPUT D$(I,J) e PRINT saranno eseguite per 5 volte, per tutti i valori di J da 1 a 5; la PRIMA VOLTA la variabile J ha valore 1; la variabile N$(J) contiene la parola "DESCRIZIONE :" come già spiegato.
Tale parola viene scritta dall'istruzione PRINT N$(J);. Analogamente l'istruzione INPUT D$(I,J) chiede un VALORE, che verrà ASSEGNATO all'elemento I,J (I=1 e J=1), per cui il valore sarà assegnato alla variabile D$(1,1).
Per i successivi passaggi del LOOP la variabile I mantiene il valore 1, mentre la variabile J diventerà 2-3-4 e 5 e le istruzioni PRINT e INPUT si riferiranno all'elemento 2-3-4-5 della variabile N$(J) ed all'elemento (1,2)-(1,3) etc. della variabile D$(I,J).

135 l'istruzione NEXT J provvede a fare rieseguire il LOOP per la variabile J. Alla fine del LOOP c'è un'istruzione PRINT a vuoto.

140 scrive un messaggio sul video.

145 richiede l'inserimento di una risposta al messaggio di cui sopra.

150 CONTROLLA la risposta per vedere se è una delle risposte ammesse.

160 AUMENTA DI 1 il numero dell'articolo (cioè predispone per l'inserimento dell'articolo successivo).

170 SE l'utente ha fatto la scelta "F" la SUBROUTINE termina ed il programma torna al MENU principale.

180 ALTRIMENTI il programma torna alla riga 100 per un nuovo inserimento.

NOTA: le righe da 185 a 195 sono una ESTENSIONE della SUBROUTINE e servono per segnalare che il MAGAZZINO È PIENO. Queste righe sono eseguite solo se il numero di articolo è maggiore di NM, cioè dopo 100 inserimenti, in quanto nel nostro caso abbiamo definito NM = 100.

6 INSEGNARE LE TECNICHE ELEMENTARI PER IL RIORDINO DI TABELLE

Un uso molto comune delle TABELLE è quello relativo al RIORDINO degli elementi della tabella, in relazione ad un ordine prestabilito.
Solitamente l'ordine che si stabilisce è uno dei seguenti:

1 ALFABETICO CRESCENTE o DECRESCENTE per elementi alfanumerici.

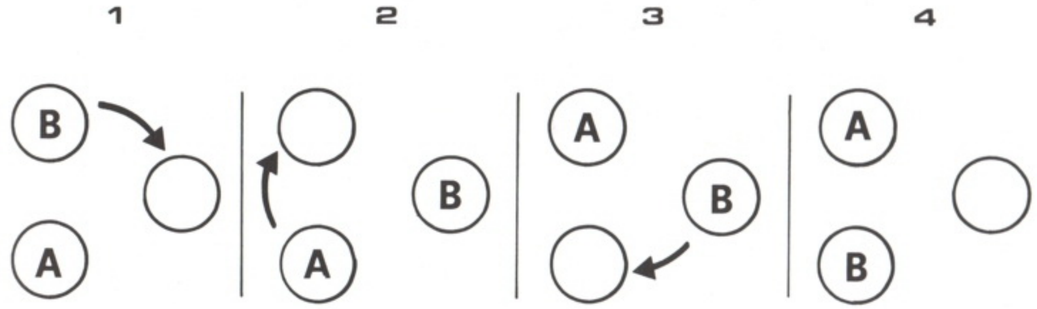
2 NUMERICO CRESCENTE o decrescente per i numeri.

Il RIORDINO ALFABETICO crescente è quello che permette di avere in ordine da A a Z gli elementi di una TABELLA ALFANUMERICA.

Il BASIC NON ha una istruzione che permette di ordinare una tabella, ma è possibile ottenere il riordino con una SUBROUTINE apposita.
Naturalmente il riordino può essere eseguito in molti modi in base alle conoscenze del programmatore. Il termine tecnico per indicare un riordino è la parola inglese SORT.
Vedremo ora come è possibile riordinare gli elementi di una TABELLA con una SUBROUTINE semplice e di facile comprensione. Tale SUBROUTINE è costituita da DUE LOOP, UN CONFRONTO IF ... THEN, ed una SUBROUTINE per l'INVERSIONE degli elementi.
Commentiamo il programma MAGAZZINO LEZIONE 7:

600 INIZIO SUBROUTINE di RIORDINO
605 Cancella lo schermo e scrive titolo
610 CONTROLLO: se non ci sono ELEMENTI da riordinare (cioè se UR=1) il riordino non viene eseguito ed il programma salta alla riga 280 dove viene segnalato quanto sopra.
620 scrittura messaggio di ATTESA per far capire che il computer sta elaborando, anche se l'utente non se ne accorge.
630 INIZIO primo LOOP per la variabile I e per tutti i valori da 1 fino al PENULTIMO elemento inserito, cioè per tutti gli articoli inseriti escluso l'ultimo.
635 INIZIO secondo LOOP per la variabile J.
NOTA: il valore di INIZIO per la variabile J cambia ogni volta in quanto DIPENDE dal valore della variabile I (FOR J = I+1) per cui assumerà, di volta in volta, i valori 2-3- etc. mentre il valore FINALE del secondo LOOP è UR-1 cioè l'ultimo articolo che è stato inserito.
640 CONFRONTO tra il contenuto della variabile D$(I,1) e quello della variabile D$(J,1). In questo modo si determina se eseguire la SUBROUTINE di INVERSIONE (che inizia alla riga 700).
650 Istruzione NEXT J,I che fa ripetere i DUE LOOP. Naturalmente PRIMA viene completato il LOOP per la variabile J, e DOPO quello relativo alla variabile I.
655 Messaggio di FINE RIORDINO sul video.
660 LOOP da 1 a 2000 che non esegue nessuna istruzione.

Vediamo ora in dettaglio la subroutine che esegue le INVERSIONI:



700 INIZIO SUBROUTINE INVERSIONI
710 INIZIO di un LOOP per la variabile K per tutti i valori compresi tra 1 ed il valore di NC (cioè tra 1 e 5).
720 INVERSIONE degli ELEMENTI della TABELLA a DUE INDICI di NOME D$. Per poter affettuare tale inversione è stato necessario usare un metodo un po' complicato che ora cercheremo di spiegare: l'obiettivo era di portare TUTTI gli ELEMENTI relativi all'articolo indicato da I nei CORRISPONDENTI ELEMENTI dell'articolo INDICATO da J e viceversa. Per fare questo il BASIC ha l'istruzione SWAP, ma il computer usato non ha tale istruzione. Per non perdere nessun dato, è necessario trasferire uno dei due dati in una terza variabile: D$(I,Ø).
L'elemento relativo alla variabile I è poi stato cambiato, rendendolo uguale al secondo: D$(I,K) = D$(J,K).
In ultimo si è cambiato l'elemento relativo alla variabile J facendolo diventare uguale al contenuto della variabile D$(I,0), che contiene il PRIMO dei due dati.
730 l'istruzione NEXT (che è riferita alla variabile K) fa ripetere il LOOP per tutti i valori da 1 a 5 in modo da invertire tutti gli elementi della variabile (e non solo la DESCRIZIONE).
740 RETURN cioè fine della SUBROUTINE per l'INVERSIONE. L'esecuzione del programma torna alla riga 650, cioè alla PRIMA istruzione che c'è IMMEDIATAMENTE dopo il comando GOSUB 700.

NOTA: Il metodo usato nel programma per riordinare gli elementi della tabella è stato realizzato, a scopo didattico, per evidenziare come una SUBROUTINE può richiamarne un'altra.

In realtà la SUBROUTINE 700 che effettua le INVERSIONI NON è necessaria, e la parte centrale della SUBROUTINE di RIORDINO può essere scritta nel seguente modo:

```
640 IF D$(I,1)<= D$(J,1) THEN 650
641 FOR K=1 TO NC
642 D$(I,Ø)=D$(I,K):D$(I,K)=D$(J,K):D$(J,K)=D$(I,Ø)
643 NEXT K
650 NEXT J,I
```

APPENDICE: ALGORITMI DI RIORDINO

L'ALGORITMO è un procedimento logico, che consente di risolvere un problema.

In pratica l'algoritmo consiste nella rappresentazione grafica o verbale di un METODO RISOLUTIVO.

Lo studio di un algoritmo avviene analizzando le informazioni che si possiedono, in funzione della soluzione desiderata.

Come abbiamo già potuto constatare nelle prime lezioni, parlando dello pseudo-linguaggio, è possibile applicare al computer un metodo risolutivo, usando il linguaggio di programmazione.

Una conoscenza approfondita delle risorse del computer consentono inoltre l'ottimizzazione, cioè il risultato migliore, che sfrutta totalmente le possibilità della macchina (computer).

Abbiamo già analizzato il problema del riordino di una serie di elementi, attraverso un algoritmo di riordino (SORT).

Il procedimento applicato prende il nome di SELECTION SORT, in quanto opera una selezione degli elementi, individuando l'elemento di valore più basso, collocandolo immediatamente al proprio posto. Il procedimento si ripete sui rimanenti elementi, fino alla risoluzione, cioè il riordino di tutti gli elementi.

A scopo informativo riportiamo qui di seguito i listati di brevi programmi che consentono il riordino di un vettore attraverso procedimenti differenti, ma tutti ugualmente validi:

1 - BUBBLE SORT (riordino a bolle)

2 - SHELL SORT (riordino di Shell)

3 - QUICK SORT (riordino veloce)

Si tratta dei più famosi algoritmi che permettono il riordino di una serie di elementi, ed ognuno ha caratteristiche proprie, con prestazioni ottimali proporzionali al quadrato degli elementi da riordinare (valore medio).

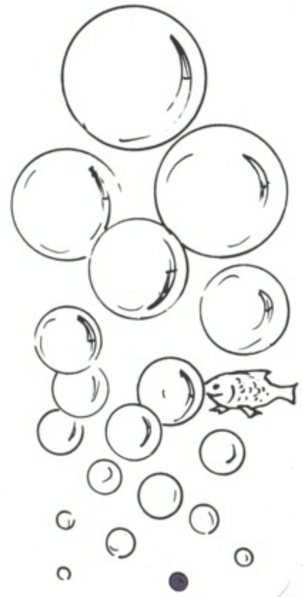
Il QUICK SORT è più veloce degli altri, (valore medio proporzionale a N*logN), ma necessita di una maggiore quantità di memoria.

Altri famosi algoritmi di riordino sono:

– INSERTION SORT (riordino per inserimento)

– HEAP-SORT (riordino per accumulo)

– MERGE-SORT (riordino per fusione)



BUBBLE SORT

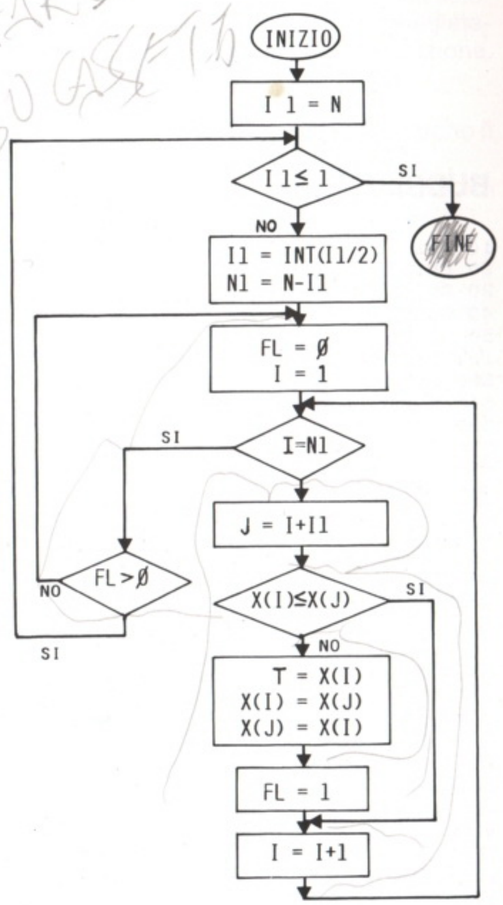
```
10 CLS
20 PRINT "PROVA BUBBLE SORT"
30 GOSUB 500  ' INIZIALIZZA
40 GOSUB 600
50 GOSUB 1000
100 END
500 REM INIZIALIZZA  VALORI PER PROVE
510 N=100 : DIM X(N)
520 FOR I=1 TO N
530 X(I)=N+1-I
540 NEXT I
550 RETURN
600 PRINT "VALORI PRIMA DEL RIORDINO"
610 FOR I=1 TO N
620 PRINT X(I); : NEXT : PRINT
650 RETURN
1000 REM BUBBLE SORT
1010 N1=N-1
1020 FOR I= 1 TO N1
1030 J=I+1
1040 IF X(I) <= X(J)THEN 1090
1050 T=X(I)
1060 X(I)=X(J)
1070 X(J)=T
1090 NEXT I
1100 N1=N1-1
1150 IF N1 >=1 THEN 1020
1160 PRINT"VALORI DOPO RIORDINO"
1170 FOR I=1 TO N
1180 PRINT X(I); : NEXT : PRINT
1190 RETURN
```

QUICK SORT

```
10 CLS
20 PRINT "PROVA QUICK SORT"
30 GOSUB 500  ' INIZIALIZZA
40 GOSUB 600
50 GOSUB 1000
100 END
500 REM INIZIALIZZA  VALORI PER PROVE
510 N=100 : DIM X(N),S(N,1)
520 FOR I=1 TO N
530 X(I)=N+1-I
540 NEXT I
550 RETURN
600 PRINT "VALORI PRIMA DEL RIORDINO"
610 FOR I=1 TO N
620 PRINT X(I); : NEXT : PRINT
650 RETURN
1000 REM QUICK SORT
1010 I1=1
1020 J1=N
1030 I=I1
1040 J=J1
1050 FL=-1
1060 IF X(I)<= X(J) THEN 1110
1070 T=X(I)
1080 X(I)=X(J)
1090 X(J)=T
1100 FL=-FL
1110 IF FL = 1 THEN I=I+1 ELSE J=J-1
1120 IF I<J THEN 1060
1130 IF I+1 >=J1 THEN 1170
1140 K=K+1
1150 S(K,0)=I+1
1160 S(K,1)=J1
1170 J1=I-1
1180 IF I1<J1 THEN 1030
1190 IF K=0 THEN 1300
1200 I1=S(K,0)
1210 J1=S(K,1)
1220 K=K-1
1230 GOTO 1030
1300 PRINT "VALORI DOPO QUICK SORT"
1310 FOR I=1 TO N
1320 PRINT X(I);
1330 NEXT I
1350 RETURN
```

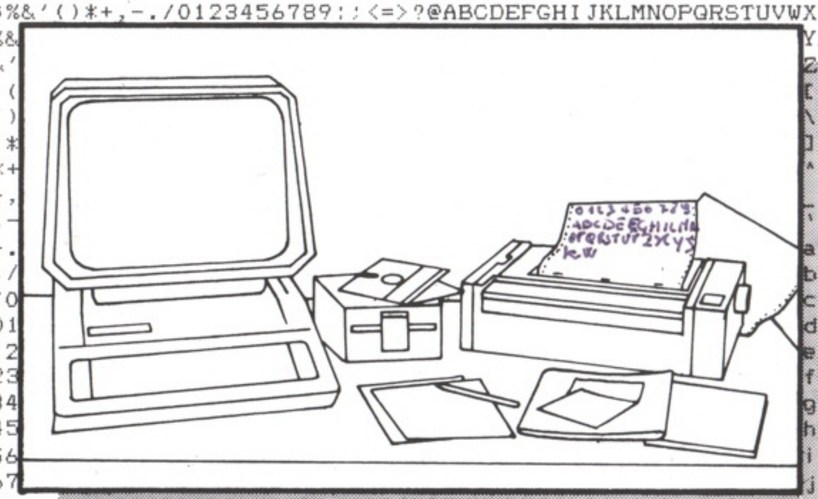
SHELL SORT

```
10 CLS
20 PRINT "PROVA SHELL SORT"
30 GOSUB 500  ' INIZIALIZZA
40 GOSUB 600
50 GOSUB 1000
100 END
500 REM INIZIALIZZA  VALORI PER PROVE
510 N=100 : DIM X(N)
520 FOR I=1 TO N
530 X(I)=N+1-I
540 NEXT I
550 RETURN
600 PRINT "VALORI PRIMA DEL RIORDINO"
610 FOR I=1 TO N
620 PRINT X(I); : NEXT : PRINT
650 RETURN
1000 REM SHELL SORT
1010 I1=N
1020 IF I1<=1 THEN 1160
1030 I1=INT(I1/2)
1040 N1=N-I1
1050 FL=0
1060 FOR I=1 TO N1
1070 J=I+I1
1080 IF X(I)<=X(J) THEN 1130
1090 T=X(I)
1100 X(I)=X(J)
1110 X(J)=T
1120 FL=1
1130 NEXT I
1140 IF FL>0 THEN 1050
1150 GOTO 1020
1160 PRINT"VALORI DOPO RIORDINO"
1170 FOR I=1 TO N
1180 PRINT X(I); : NEXT : PRINT
1190 RETURN
```



FLOW CHART
(Diagramma di flusso)
del riordino di shell

ESERCIZI



ESERCITAZIONI E STUDI DA ESEGUIRE A CASA:

1) Esercitarsi nell'uso delle istruzioni spiegate durante la lezione.

2) COMPLETARE l'ELENCO del VOCABOLARIO BASIC, aggiungendo le VOCI apprese in questa lezione:
   – istruzioni READ, DATA e RESTORE,
   – VARIABILI INDICIZZATE a più dimensioni (TABELLE).

3) COMPLETATE IL PROGRAMMA MAGAZZINO LEZIONE 7 in modo da renderlo IDENTICO al LISTATO allegato, ricordando quanto segue: l'istruzione CLS (che compare nel listato), serve per cancellare lo schermo, mentre l'istruzione CLEAR ha la funzione di riservare posto alle variabili.

   Se il BASIC del computer usato NON ha tali istruzioni, sarà necessario apportare le seguenti modifiche:

   a) – sostituire l'istruzione CLS con l'equivalente istruzione.
   b) – sostituire, o eliminare l'istruzione CLEAR.

   CAMBIARE TITOLO al programma, chiamandolo come segue:

   MAGAZZINO LEZIONE 8

4) SALVARE il programma su NASTRO (DOPO AVERLO CONTROLLATO), e portare il NASTRO ed il LISTATO alla prossima LEZIONE.

NOTE: Il listato contempla le SUBROUTINES per la LETTURA e SCRITTURA su NASTRO dei dati (SUBROUTINE 800 e SUBROUTINE 900).

Poichè ogni computer ha proprie regole riguardo tali operazioni è necessario verificare, sul manuale del vostro computer, quali sono le ISTRUZIONI necessarie per poter SCRIVERE e LEGGERE un FILE (pronuncia: FAIL) SEQUENZIALE su nastro.

Alcuni computer prescrivono l'uso dell'istruzione OPEN per "aprire" la comunicazione con la periferica (registratore), prima di eseguire la SCRITTURA (con l'istruzione PRINT#), o prima della LETTURA (con l'istruzione INPUT#).

In tale caso è anche prevista l'istruzione CLOSE per "chiudere" la comunicazione dopo la SCRITTURA o dopo la LETTURA.

Le ISTRUZIONI di lettura e scrittura del FILE sequenziale dovranno essere scritte NEL MODO RICHIESTO DAL VOSTRO COMPUTER, altrimenti le SUBROUTINES relative NON eseguiranno le operazioni previste, causando segnalazioni di errore.

ESERCITAZIONI A CASA - LISTATO PROGRAMMA (Prima parte)

```
1 REM ** MAG L8  PER LASER   **
2 REM **  (DATI SU NASTRO)   **
3 CLEAR1000
5 GOSUB 400 : REM INIZIALIZZA
10 REM ** MENU E SCELTE **
20 CLS
25 PRINT "MAGAZZINO" : PRINT
30 PRINT"1-INSERIM.    2-LETTURA" : PRINT
33 PRINT"3-MODIFICA    4-FINE PRG" : PRINT
35 PRINT"5-TOTALIZZA   6-RIORDINO" : PRINT
36 PRINT"7-LETT NASTRO 8-SCR NASTRO" : PRINT
40 INPUT"COSA SCEGLI";R$
50 IFR$="1" THEN M$="INSERIMENTO MAGAZZINO":GOSUB100
60 IFR$="2" THEN M$="LETTURA MAGAZZINO" : GOSUB200
70 IFR$="3" THEN M$="MODIFICA MAGAZZINO" : GOSUB300
75 IFR$="4" THEN CLS : END
80 IFR$="5" THEN M$="TOTALIZZAZIONE" : GOSUB500
85 IFR$="6" THEN M$="RIORDINO" : GOSUB600
86 IFR$="7" THEN M$="LETTURA  NASTRO" : GOSUB800
87 IFR$="8" THEN M$="SCRITTURA NASTRO" : GOSUB900
90 GOTO 10
100 REM ** INSERIMENTO **
110 I=UR
115 CLS : PRINT M$ : PRINT
120 IF UR>NMTHEN 185
125 PRINT"ART.";I   : PRINT
130 FOR J = 1 TO NC : PRINT N$(J); : INPUT D$(I,J) : PRINT
135 NEXTJ : PRINT
140 PRINT"F=FINE   A=ALTRO INSER." : PRINT
145 INPUT"COSA SCEGLI";S$
150 IF S$<>"A"ANDS$<>"F"THEN 145
160 UR=UR+1
170 IF S$="F"THEN RETURN
180 GOTO100
185 PRINT:PRINT"MAGAZZINO PIENO":PRINT
190 PRINT"PER AGGIUNGERE ARTICOLI":PRINT
192 PRINT"DEVI MODIFICARE NM":PRINT
194 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN194
195 RETURN
200 REM ** LETTURA **
205 CLS : PRINT M$ : PRINT
210 IF UR=1THEN280
220 INPUT"ART.";I : PRINT
225 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 200
230 FOR J = 1 TO NC :PRINT N$(J);D$(I,J) : PRINT:NEXT J : PRINT
240 INPUT"ALTRO ARTICOLO (S/N)";S$
250 IF S$<>"S" AND S$<>"N" THEN240
260 IF S$="N"THEN RETURN
270 GOTO 200
280 PRINT:PRINT"NESSUN ARTICOLO PRESENTE":PRINT
285 PRINT"(PRIMA DEVI INSERIRE)":PRINT
290 INPUT"CAPITO (S/N)";S$:IFS$<>"S"ANDS$<>"N"THEN290
295 RETURN
```

LISTATO PROGRAMMA (Seconda parte)

```
300 REM ** MODIFICA **
305 CLS : PRINT M$ : PRINT
310 IF UR=1THEN280
320 INPUT"ART.";I  : PRINT
325 IF I<1 OR I>UR-1 OR I<>INT(I) THEN 300
330 FOR J=1TONC:PRINTN$(J);:INPUT D$(I,J) : PRINT
335 NEXT J : PRINT
340 INPUT"ALTRO ARTICOLO (S/N)";S$
350 IF S$="S" THEN 300
360 IF S$<>"N"THEN 340
370 RETURN
400 REM ** INIZIALIZZA VARIABILI **
410 NM=100 :  NC=5 : UR=1 : DIM D$(NM,NC),N$(NC)
420 RESTORE:FOR I=1TONC:READ N$(I):NEXT
430 DATA "DESCRIZIONE  : "
431 DATA "QUANTITA'... : "
432 DATA "PREZZO ..... : "
433 DATA "IVA % ...... : "
434 DATA "FORNITORE .. : "
450 RETURN
462 BLOCKS FREE.
500 REM * TOTALIZZA INVENTARIO *
505 CLS : PRINT M$ : PRINT
510 T=0:FOR I = 1 TO UR-1
530 Q=VAL(D$(I,2)): P=VAL(D$(I,3))
540 T = T + Q*P
550 NEXT I : PRINT
560 PRINT"ARTICOLI PRESENTI : ";UR-1 : PRINT
570 PRINT"VALORE TOTALE =";T : PRINT
580 INPUT"TORNO AL MENU' (S/N)";S$:IFS$<>"S"THEN 500
590 RETURN
600 REM * RIORDINO *
605 CLS : PRINT M$ : PRINT
610 IF UR=1THEN280
620 PRINT"ATTENDERE MENTRE RIORDINO":PRINT
630 FOR I=1TO UR-2
635 FOR J=I+1 TO UR-1
640 IF D$(I,1)>D$(J,1)THEN GOSUB 700
650 NEXT J,I
655 PRINT"FINE RIORDINO":PRINT
660 FOR TM=1 TO 2000: NEXT
670 RETURN
```

LISTATO PROGRAMMA (Terza parte)

```
700 REM * INVERSIONI PER RIORDINO *
710 FOR K=1 TO NC
720 D$(I,0)=D$(I,K):D$(I,K)=D$(J,K):D$(J,K)=D$(I,0)
730 NEXT
740 RETURN
800 REM ** LEGGE NASTRO *
805 CLS : PRINT M$ : PRINT
815 PRINT"INSERIRE NASTRO DATI"
816 PRINT"E POSIZIONARE NASTRO"
820 INPUT"FATTO (S/N)";S$
825 IFS$<>"S"THENRETURN
830 CLS: PRINT M$ : PRINT : PRINT"PREMERE TASTO PLAY  "
835 INPUT#"MAGAZZINO",UR$:UR=VAL(UR$)
840 FOR I=1TOUR-1
850 FORJ=1TONC
860 INPUT#"MAGAZZINO",D$(I,J)
870 NEXTJ,I
880 CLS : PRINT"PREMI TASTO STOP"
881 PRINT"SUL REGISTRATORE"
885 INPUT"FATTO (S/N)";S$
890 IFS$<>"S"THEN880
895 RETURN
900 REM ** SCRIVE NASTRO *
905 CLS : PRINT M$ : PRINT
910 IF UR=1THEN280
915 PRINT"INSERIRE NASTRO DATI"
916 PRINT"E POSIZIONARE NASTRO"
920 INPUT"FATTO (S/N)";S$
925 IFS$<>"S"THENRETURN
926 CLS : PRINT M$ : PRINT
930 PRINT"PREMERE TASTI REC & PLAY"
931 PRINT"SUL REGISTRATORE"
932 INPUT"FATTO (S/N)";S$
933 IFS$<>"S"THEN926
934 UR$=STR$(UR)
935 PRINT#"MAGAZZINO",UR$;CHR$(13);
940 FOR I=1TOUR-1
950 FORJ=1TONC
960 PRINT#"MAGAZZINO",D$(I,J);CHR$(13);
970 NEXTJ,I
980 CLS:PRINT"PREMI TASTO STOP"
981 PRINT"SUL REGISTRATORE"
985 INPUT"FATTO (S/N)";S$
990 IFS$<>"S"THEN980
995 RETURN
```

LEZIONE 8

OBIETTIVI

1. Ripresentare il problema relativo alla gestione dei dati.
2. Chiarire la differenza tra memoria del computer e memoria di massa.
3. Spiegare cosa è un FILE di dati.
4. ACCENNARE ai diversi tipi di FILES, spiegando i FILES SEQUENZIALI.
5. Spiegare le istruzioni OPEN, e PRINT ##
6. Spiegare le istruzioni CLOSE e INPUT ##
7. Finalizzare quanto spiegato alla scrittura e lettura di un FILE SEQUENZIALE, con l'uso del registratore.

1 RIPRESENTARE IL PROBLEMA RELATIVO ALLA GESTIONE DEI DATI

Con l'uso delle variabili indicizzate è possibile conservare nella memoria del computer una notevole quantità di dati.

L'esempio proposto nel corso delle lezioni, relativo alla gestione di un magazzino, ci ha permesso di verificare l'uso delle istruzioni e delle tecniche di programmazione apprese fino ad ora.

In particolare abbiamo potuto constatare che ci sono due TIPI di DATI che possono essere elaborati:

DATI interni al programma, che servono soprattutto al programmatore, e sono memorizzati nel programma con l'istruzione DATA.

Tali DATI sono sempre disponibili quando viene CARICATO in memoria il programma.

Nel corso del programma l'istruzione READ permette di ASSEGNARE questi dati e delle variabili, per poterli usare.
I DATI che interessano l'utente invece NON SONO immediatamente disponibili, ma vengono inseriti dall'utente di volta in volta.

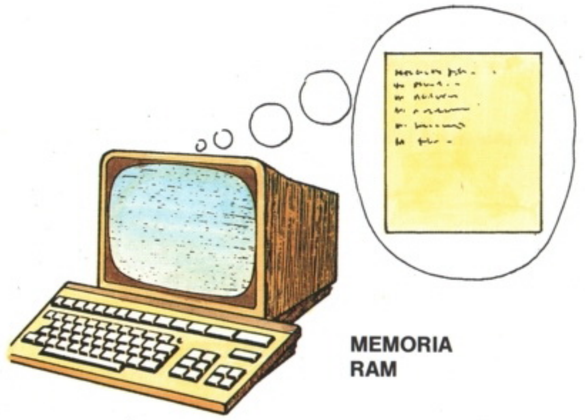
Il programma MAGAZZINO realizzato permette di elaborare dei dati.

Ogni volta che si spegne il computer, o si RIESEGUE il programma, con il comando RUN i dati inseriti dall'utente vengono PERSI.

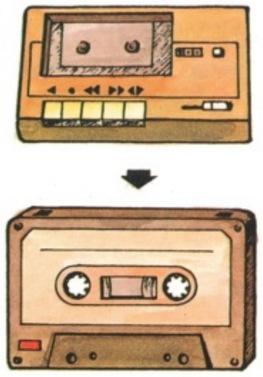
Questo avviene proprio per il fatto che tali DATI sono conservati nelle variabili (nel nostro caso TABELLE).

OGNI volta che si dà il comando di sistema RUN il computer CANCELLA il contenuto di tutte le VARIABILI.

Per conservare in modo permanente i dati inseriti è necessario TRASFERIRLI su una MEMORIA DI MASSA.



MEMORIA RAM



MEMORIA DI MASSA

Il programma dovrà quindi permettere tale trasferimento.

Il programma dovrà permettere anche di LEGGERE i dati DALLA memoria di massa su cui sono stati trasferiti.

Per trasferire UN PROGRAMMA dalla memoria del computer, al nastro del registratore, si usa il comando di sistema CSAVE (o SAVE) e per leggere dal NASTRO un PROGRAMMA si usa il comando CLOAD (o LOAD).

```
LOAD

CLOAD

SAVE

CSAVE
```

Tali comandi possono essere usati SOLTANTO per i programmi.

Per i DATI è necessario inserire NEL programma le opportune istruzioni.

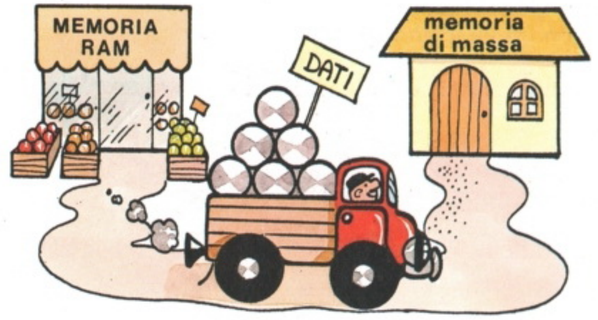
2 CHIARIRE DIFFERENZA TRA MEMORIA RAM E MEMORIA DI MASSA

Abbiamo parlato della MEMORIA RAM, che è la MEMORIA a nostra disposizione per le elaborazioni.

Quando scriviamo un programma usiamo una parte di tale memoria.

Nella MEMORIA RAM inoltre vengono conservati anche i NOMI ed i VALORI delle VARIABILI che usiamo nel programma (oppure in modo diretto).

Per poter conservare il valore di tali variabili è necessario trasferire questi valori su una memoria di MASSA.



Anche se può sembrare superfluo è meglio chiarire che la MEMORIA di MASSA è una cosa MOLTO diversa dalla MEMORIA del computer.

La principale differenza tra la memoria del computer ed una MEMORIA di MASSA sta nel fatto che la memoria del computer contiene una grande quantità di dati che vengono usati dal SISTEMA OPERATIVO.

Tali dati non interessano l'utente o il programmatore: servono al computer per eseguire i programmi RESIDENTI su ROM (cioè i programmi realizzati dal costruttore del computer).

Nella memoria di massa invece vengono trasferite SOLTANTO le informazioni che possono essere poi elaborate.

In una memoria di massa è possibile trasferire due TIPI fondamentali di dati:

1. PROGRAMMI.
2. DATI il cui valore è contenuto in variabili (cioè DATI che servono per l'elaborazione).

Solitamente si usa la parola inglese FILE (pronuncia FAIL) per indicare i DATI che si trovano su di una memoria di massa.
Avremo quindi i FILES PROGRAMMI ed i FILES DATI.

3 SPIEGARE COSA È UN FILE DI DATI

Un FILE programma può essere trasferito dalla memoria del computer, ad una memoria di massa con l'istruzione CSAVE (o SAVE).
Il sistema operativo provvede ad eseguire il trasferimento delle informazioni che gli interessano.

Analizzare quali sono queste informazioni, e come vengono trasferite esula dagli obiettivi del corso.

Il comando CLOAD (o LOAD) permette di LEGGERE le informazioni relative ad un programma per "CARICARE" nella memoria RAM.

Vediamo invece cosa è un FILE di dati:

Un FILE di dati è una serie di VALORI che l'utente desidera conservare, in modo da poterli usare all'occorrenza.

Questi valori possono essere SCRITTI o LETTI in molti modi diversi, in base alle esigenze dell'utente, oppure in relazione alla natura dei dati stessi.

Per eseguire la scrittura di un FILE di DATI è necessario stabilire DUE COSE MOLTO IMPORTANTI.

1. QUALI SONO i dati che si desiderano trasferire.

2. Come realizzare il trasferimento (cioè la scrittura) di tali dati.

La LETTURA dei dati è una diretta conseguenza della SCRITTURA, in quanto per LEGGERE i DATI da una MEMORIA DI MASSA è indispensabile sapere COME SONO STATI SCRITTI tali dati.

Ogni computer permette di utilizzare più di un MODO per scrivere e leggere i FILES di dati.

Questi modi diversi dipendono anche dal TIPO di MEMORIA di MASSA che si intende usare.

Le memorie di MASSA più usuali sono di TRE TIPI:

1) NASTRO (memoria in SCRITTURA e LETTURA)

2) DISCHI (rigidi, flessibili etc.) (memoria in SCRITTURA e LETTURA).

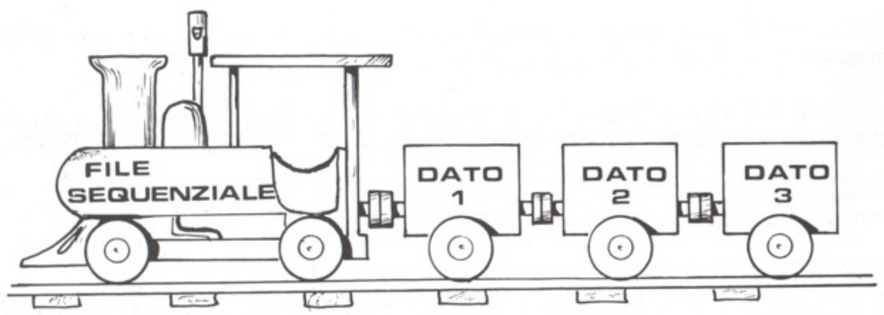
3) CARTA (MEMORIA SOLAMENTE in SCRITTURA).

4 ACCENNARE AI DIVERSI TIPI DI FILES SPIEGANDO I SEQUENZIALI

I FILES che possono essere usati dal BASIC sono solitamente i seguenti:

1. FILES SEQUENZIALI (ad accesso sequenziale)
2. FILES RANDOM (ad accesso TOTALMENTE causale).
3. FILES RANDOM RELATIVE (ad accesso RELATIVAMENTE causale).

I FILES SEQUENZIALI sono quelli che hanno un più facile utilizzo.
Tali FILES possono essere usati per trasferire o leggere dati, sia su NASTRO che su DISCO.



I FILES RANDOM possono essere usati solo con i DISCHI, e il loro uso richiede una serie di conoscenze che esulano dagli obiettivi del corso, per cui non verranno più ripresi.

Un FILE SEQUENZIALE è una serie ordinata di DATI che sono scritti su NASTRO o DISCO.

La scrittura di un FILE SEQUENZIALE deve essere eseguita sempre dal PRIMO elemento, FINO all'ULTIMO elemento. Analogamente la LETTURA di un FILE SEQUENZIALE deve avvenire dal PRIMO, fino all'ULTIMO elemento.

La lettura e la scrittura avvengono in SEQUENZA, per questo sono detti SEQUENZIALI.

Ogni computer ha proprie regole per l'uso dei FILES, per cui la sintassi delle istruzioni di scrittura e lettura dei FILES può cambiare in relazione al computer usato, o alla periferica (registratore, disk drive).

Vedremo le principali REGOLE per usare i FILES SEQUENZIALI, consigliando di consultare il manuale del proprio computer per verificare differenze nell'uso delle istruzioni.

5 SPIEGARE LE ISTRUZIONI OPEN E PRINT

Per scrivere dei DATI in un FILE sequenziale si deve usare l'istruzione BASIC:

PRINT#, var, var, var ...

in cui PRINT# è il nome dell'istruzione, che deve essere SEMPRE seguito dal segno virgola e dal NOME della VARIABILE (o delle variabili) di cui si vuole scrivere il contenuto.

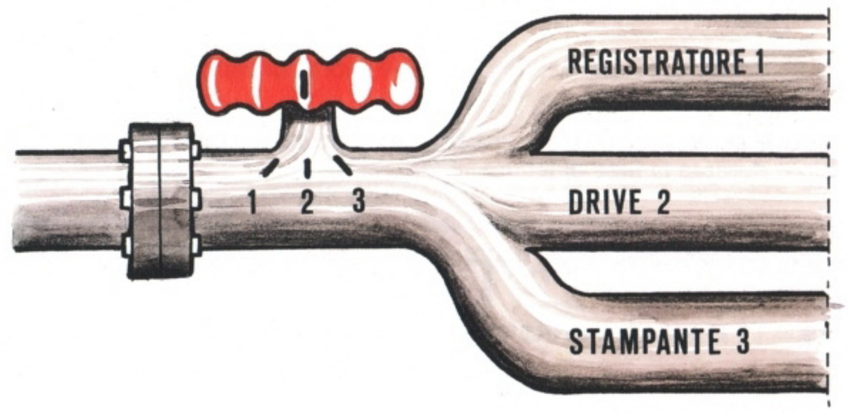
PRINT#

Se si vuole scrivere più di una valore per volta è indispendabile che i nomi delle variabili siano separati da un segno di sintassi valido (virgola o punto e virgola).

Se si scrive più di un VALORE PER VOLTA è preferibile mettere, dopo ogni valore, anche un segno di separazione (solitamente CHR$ (13), che è il CODICE ASCII del TASTO RETURN).

ESEMPIO: Per scrivere su nastro il contenuto delle variabili A, B e C si scriverà PRINT#, A;CHR(13);B;CHR$(13);C;CHR$(13);

NOTA IMPORTANTE:
Alcuni computer prevedono che, per poter scrivere su di un FILE, si debba innanzitutto APRIRE un canale di comunicazione con la periferica (nel nostro caso è il registratore).



L'APERTURA di un canale (che viene anche detta APERTURA DEL FILE), è ottenuta con l'istruzione OPEN.

La sintassi dell'istruzione OPEN varia da computer a computer: solitamente dopo l'istruzione OPEN c'è un NUMERO che indica il CANALE che viene aperto, un secondo numero che indica la PERIFERICA, ed un terzo numero che indica il tipo di operazione (LETTURA o SCRITTURA).

OPEN

ESEMPIO:
OPEN 1,1,1 ... comando di apertura di un FILE SEQUENZIALE, sul registratore, per scrittura.

In tale esempio il primo 1 è il numero di CANALE, il secondo 1 sta ad indicare la PERIFERICA registratore, il terzo 1 indica l'operazione di SCRITTURA.

Quando il computer necessita della istruzione OPEN, anche l'istruzione PRINT# ha una sintassi diversa, in quanto occorre fare riferimento al CANALE aperto dall'istruzione OPEN per poter eseguire la scrittura.

Nel nostro caso potremo scrivere:
PRINT# 1,A;CHR$(13);B;CHR$(13);C;CHR$(13);
cioè: SCRIVI il FILE sul CANALE 1.

La sintassi delle istruzioni BASIC che permettono di "DIALOGARE" con le periferiche dipende strettamente dal computer usato.

Studiare con attenzione il manuale del proprio computer ed eseguire le opportune prove pratiche.

NOTA:
Quando si scrive un FILE è opportuno controllare che le variabili alfanumeriche non abbiano il valore " " (stringa vuota).

Molto spesso la scrittura su un file di una stringa vuota impedisce al computer di riconoscere tale valore durante la fase di lettura.

Pertanto controllare sempre in scrittura il valore delle variabili alfanumeriche, e se tale valore è " " porlo forzatamente ad un valore preciso che potrà essere controllato in lettura.

ESEMPIO:
1000 IF A$ = " " THEN A$ = CHR$ (255)

6 SPIEGARE LE ISTRUZIONI CLOSE E INPUT#

Quando un computer richiede l'istruzione OPEN per APRIRE un canale di comunicazione, richiede anche che DOPO l'operazione di scrittura (o di lettura) il CANALE venga chiuso.

Per chiudere il canale di comunicazione si usa l'istruzione CLOSE, indicando il numero del canale che si vuole chiudere.
Nel caso visto prima dovremo specificare CLOSE 1.

CLOSE

Quando un computer prescrive le istruzioni OPEN e CLOSE, occorre eseguire le seguenti operazioni:

1. APRIRE IL CANALE di comunicazione specificando la periferica e il tipo di operazione da eseguire.
2. ESEGUIRE TUTTE le operazioni ritenute necessarie (cioè tutte le letture o scritture).
3. CHIUDERE il canale di comunicazione.

Vediamo ora l'istruzione che permette di LEGGERE un FILE SEQUENZIALE che sia stato precedentemente SCRITTO.

L'istruzione è INPUT#, var, var, var

e significa: LEGGI un FILE, assegnando i valori letti, alle variabili indicate dopo la virgola.

INPUT#

ESEMPIO: per RILEGGERE dal nastro i valori delle variabili A, B, e C che abbiamo scritto precedentemente potremo scrivere la seguente istruzione: INPUT#,A,B,C.

NOTA: Se il computer lo richiede, è necessario APRIRE un CANALE di comunicazione, specificando, l'operazione di LETTURA, con l'istruzione OPEN, e l'istruzione INPUT# dovrà riferirsi al canale che è stato aperto, esempio: OPEN 1,1,0: INPUT# 1,A,B,C,:CLOSE 1.

È possibile dare un NOME al FILE che si vuole scrivere indicando tale nome tra virgolette.

ESEMPIO: (Se il computer NON prevede le istruzioni OPEN e CLOSE)

```
PRINT#"MAGAZZINO",A;CHR$(13);B;CHR$(13);C;CHR$(13);
```

oppure: (se previste le istruzioni OPEN e CLOSE)

```
OPEN 1,1,1,"MAGAZZINO"
PRINT# 1,A;CHR$(13);B;CHR$(13);C;CHR$(13);
CLOSE 1
```

NOTA: Se in scrittura è stato usato il controllo:

```
100 IF A$ = " " THEN A$ = CHR$ (255)
```

in lettura è necessario riconvertire il valore della variabile con un nuovo controllo

```
2000 IF A$ = CHR$ (255) then A$ = " ".
```

Verificare le regole e le istruzioni disponibili sul Vs. computer per la lettura e scrittura di files sequenziali.

7 FINALIZZARE QUANTO SPIEGATO PER SCRITTURA E LETTURA DI FILES

Possiamo completare il programma MAGAZZINO, con le SUBROUTINES per la SCRITTURA e LETTURA di un FILE SEQUENZIALE, per conservare i dati che ci interessano e poterli rileggere.

Vediamo i punti che dobbiamo considerare:

1. Aggiungere al MENÙ le due SCELTE di LETTURA e SCRITTURA FILE.
2. Scrivere le due SUBROUTINES relative, tenendo conto di quali sono i DATI che ci servono.

Analizziamo ora i dati che dobbiamo "salvare" su nastro, e in quali variabili si trovano.

1. Il numero del PRIMO ARTICOLO che è possibile inserire, che è contenuto nella variabile UR.
2. Tutti i DATI del MAGAZZINO contenuti nella variabile a due indici di nome D$.

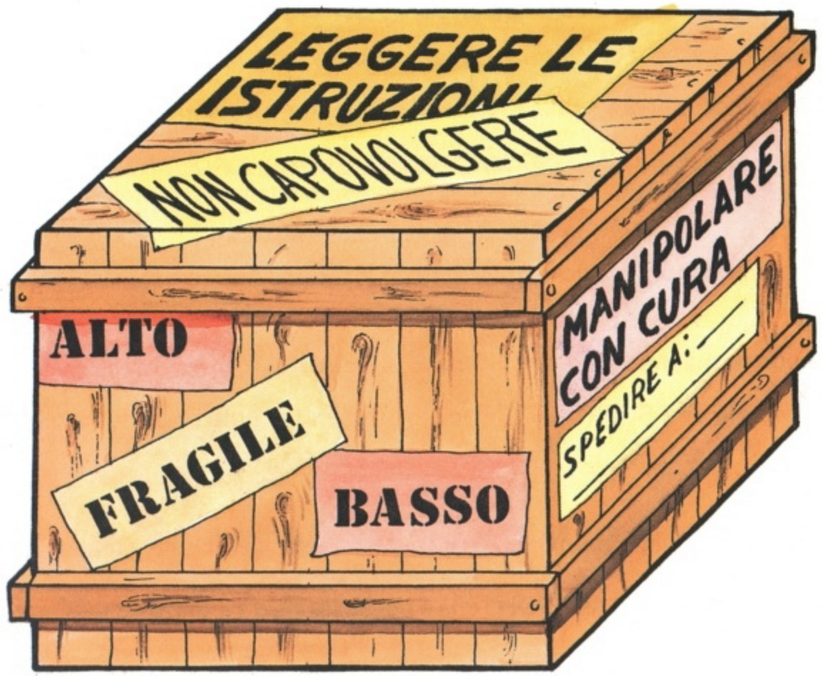
Naturalmente i DATI che vengono scritti dalla SUBROUTINE di scrittura del FILE, verranno poi letti dalla SUBROUTINE di lettura.

Se leggiamo il listato del programma MAGAZZINO LEZIONE 8 (dato come esercitazione alla lezione precedente), possiamo trovare le due SUBROUTINE già realizzate:

SUBROUTINE di SCRITTURA FILE da riga 900 a riga 995.

SUBROUTINE di LETTURA FILE da riga 800 a riga 895.

È da notare che in tali SUBROUTINE sono stati inseriti molti MESSAGGI che hanno lo scopo di GUIDARE l'utente nell'esecuzione delle operazioni (esempio: per posizionare il nastro al punto di INIZIO del FILE).

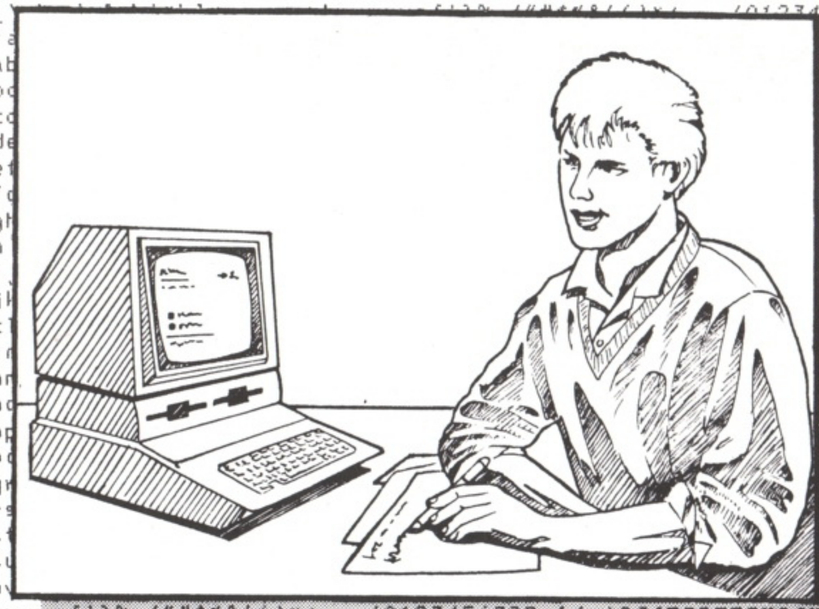


Tali messaggi sono MOLTO IMPORTANTI in quanto agevolano il lavoro dell'utente, facendogli comprendere il significato delle istruzioni che sta eseguendo.

Un programma di tipo professionale DEVE tener conto di tutte le possibilità, senza lasciare alcun dubbio nell'utente, e deve sempre considerare che l'utente non è tenuto a CONOSCERE il computer, nè tanto meno deve essere obbligato a fare SUPPOSIZIONI su come usare un certo programma.

Il programmatore deve tener conto di questo nella stesura del programma predisponendo gli opportuni messaggi o controlli.

ESERCIZI



ESERCITAZIONI E STUDI DA ESEGUIRE A CASA:

1) Esercitarsi nell'uso delle istruzioni spiegate durante la lezione in relazione al TIPO di COMPUTER USATO.

2) COMPLETARE L'ELENCO del VOCABOLARIO BASIC, aggiungendo le VOCI apprese in questa lezione, cioè ISTRUZIONI PRINT# INPUT# (OPEN e CLOSE).

L'inconveniente più frequente che il programmatore incontra nel trattamento dei FILES è il seguente:

Il computer non legge i dati che noi abbiamo scritto sul nastro o sul disco.

Questo non significa che il registratore o il drive dei dischi funzionano male, ma soltanto che non si è usata la giusta tecnica ed i controlli necessari alla scrittura e lettura del FILE stesso.

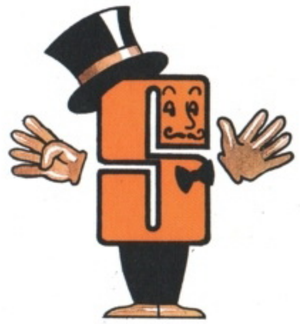
Purtroppo è indispensabile che verifichiate di persona il modo di operare del vostro computer, seguendo alcuni consigli di carattere generale.

Controllate sempre che le variabili alfanumeriche non abbiano valore nullo.
(come eseguire tale controllo è già stato spiegato).

Se una variabile alfanumerica contiene solo degli spazi vuoti (esempio: " ") il computer la scrive sul nastro, ma non riesce più a leggerla.
Per tale motivo è opportuno scrivere sul file le variabili alfanumeriche facendole precedere e seguire dal carattere CHR$ (34) che è il codice solitamente attribuito alle virgolette esempio:

```
1000 .....
1010 PRINT , CHR$(34);A$;CHR$(34)
```

Questa tecnica vi permetterà di trovare sempre, nella fase di lettura, i dati che avete scritto.



LEZIONE 9

OBIETTIVI

1. Fornire informazioni di carattere generale sull'organizzazione della memoria di un computer.
2. Insegnare l'uso delle FUNZIONI PEEK e POKE.
3. Presentare alcune applicazioni delle funzioni PEEK e POKE.
4. Completare l'elenco delle funzioni matematiche.
5. Spiegare le funzioni predefinite dal programmatore, come programmarle con l'istruzione DEF FNx, e come usarle.
6. Insegnare l'uso delle istruzioni STOP e CONT, per eseguire DUMPING e DEBUGGING, per il controllo di un programma.
7. Completare le informazioni di carattere generale sui computers (HARDWARE e SOFTWARE).
8. Accennare alle funzioni grafiche e sonore, elencando qualche comando tra quelli che è possibile trovare implementati su HOME e PERSONAL computers.

1 FORNIRE INFORMAZIONI GENERALI SULLA MEMORIA RAM

Abbiamo parlato diverse volte della memoria RAM, cioè della memoria che il computer mette a disposizione dell'utente per la scrittura di programmi o per conservare i modi ed i valori delle variabili.

Abbiamo anche detto che la memoria RAM è utilizzata anche nel sistema operativo per eseguire particolari operazioni o controlli.

Vediamo ora in modo generale come può essere suddivisa la memoria RAM.

Come già detto per altri argomenti l'organizzazione della memoria RAM NON è la stessa per tutti i computer, per cui accenneremo soltanto ad argomenti di carattere generale che possono essere comuni a tutti i computers.

Possiamo immaginare la MEMORIA RAM come una serie di CASELLE NUMERATE.
In ogni CASELLA è possibile inserire o leggere dei numeri.
La numerazione di tali caselle solitamente inizia da 0 e prosegue di un valore per volta.
Le CASELLE così identificate sono chiamate LOCAZIONI DI MEMORIA e il numero di una certa LOCAZIONE è chiamato INDIRIZZO.
Avremo quindi la LOCAZIONE di MEMORIA 0, 1, 2, 3, 4, etc. fino alla capacità di memoria totale del computer.

MEMORIA RAM

SISTEMA OPERATIVO
PROGRAMMI BASIC
VARIABILI
LOCAZIONI MEMORIA VIDEO
ALTRI USI

Solitamente le locazioni di memoria sono suddivise nel seguente modo:

1. LOCAZIONI RISERVATE al SISTEMA OPERATIVO
2. LOCAZIONI DESTINATE AI PROGRAMMI
3. LOCAZIONI DESTINATE ALLE VARIABILI
4. LOCAZIONI DESTINATE AL VIDEO
5. LOCAZIONI DESTINATE AD ALTRI SCOPI (GRAFICA, SUONO o ALTRO)

Quando scriviamo un programma, ad esempio, scriviamo sul video il numero di riga con le istruzioni. Alla pressione del TASTO RETURN il sistema operativo trasforma le nostre istruzioni in numeri, che sono inseriti nelle locazioni di memoria destinate ai programmi.

Il BASIC permette anche di LEGGERE il contenuto di una locazione di memoria, oppure di SCRIVERE un valore DIRETTAMENTE in una locazione con apposite istruzioni.

Tali istruzioni devono comunque essere usate con MOLTA ATTENZIONE, in particolar modo per quanto riguarda la SCRITTURA perchè operando DIRETTAMENTE sulla MEMORIA del computer non è possibile beneficiare dei controlli che il computer di solito esegue sulle istruzioni BASIC.

2 INSEGNARE L'USO DELLE FUNZIONI PEEK E POKE

Per LEGGERE in modo diretto il contenuto di una locazione di memoria si può usare la FUNZIONE: PEEK (locazione).
Naturalmente per vedere sul video il risultato è necessario associare la FUNZIONE alla istruzione PRINT.

ESEMPIO: PRINT PEEK(230)

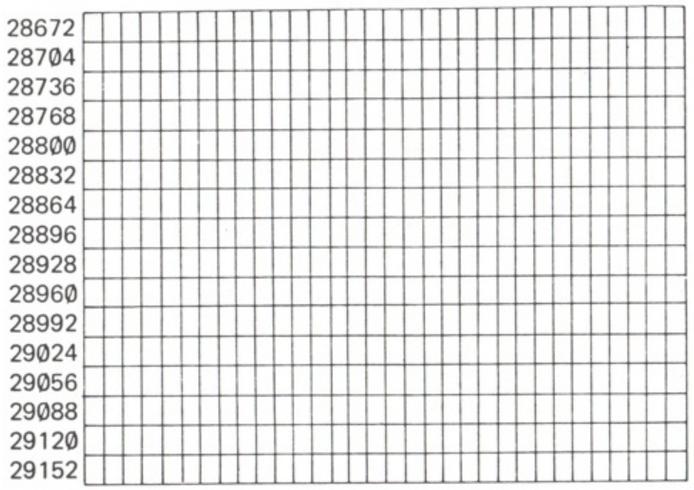
Il computer scriverà sul video un NUMERO compreso tra 0 a 255, che rappresenta il valore contenuto nella locazione 230.

È possibile leggere il contenuto di qualunque LOCAZIONE di memoria ed ogni volta si avrà come risposta un numero compreso tra 0 e 255.

La lettura della memoria con la funzione PEEK ha applicazioni molto particolari ed i numeri che vengono letti possono avere significati diversi, in relazione al computer usato.
Analogamente la scrittura nella MEMORIA può essere fatta con la FUNZIONE: POKE locazione, valore che significa: SCRIVI nella locazione di memoria indicata il valore indicato (il valore deve essere un NUMERO compreso tra 0 e 255).

Per fare un esempio vedremo una semplice applicazione delle due funzioni di cui sopra:

Per prima cosa leggiamo sul manuale del nostro computer quali LOCAZIONI di MEMORIA sono utilizzate dal VIDEO.



ESEMPIO MEMORIA VIDEO DI UN HOME COMPUTER

Inoltre sul manuale leggiamo che la lettera A ha il codice 1 per le POKE.

Se scriviamo la seguente istruzione: POKE (locazione), 1 e premiamo RETURN vedremo comparire sul video il carattere A.

Se (dopo l'istruzione POKE di cui sopra) scriviamo: PRINT PEEK (locazione) e premiamo RETURN, il computer scriverà sul video 1, cioè il codice della lettera A.

Con le FUNZIONI PEEK e POKE è possibile esplorare o modificare il contenuto della MEMORIA RAM, ma come già detto, è necessario usare tali funzioni SOLTANTO in casi molto particolari, purchè si sappia cosa si va a modificare, come visto nell'esempio di cui sopra.

Molto spesso modificando la memoria con la FUNZIONE POKE si può causare il BLOCCO del computer. Se questo avviene occorre spegnere il computer e riaccenderlo per ripristinare il corretto funzionamento, o agire sul pulsante RESET, se il computer lo possiede.
L'esempio visto può essere ripetuto per qualsiasi computer di cui si conosca la LOCAZIONE VIDEO e il CODICE CARATTERI.

NOTA:
Il codice dei caratteri relativo alle funzioni PEEK e POKE è solitamente diverso dal codice ASCII dei caratteri.

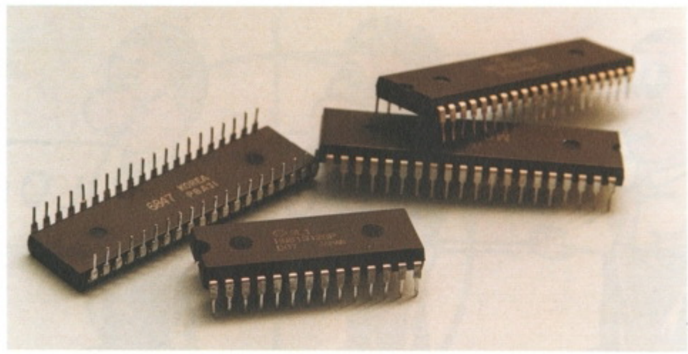
Controllate sul manuale del vostro computer i codici da usare con le funzioni PEEK e POKE.

3 PRESENTARE ALCUNE APPLICAZIONI DELLE FUNZIONI PEEK E POKE

Le funzioni PEEK e POKE permettono di comunicare con il computer per mezzo di numeri, e possono essere usate, soprattutto negli HOME COMPUTER per MODIFICARE il SISTEMA OPERATIVO, ottenendo dal computer particolari funzioni.

Per fare questo è però necessario conoscere il SISTEMA OPERATIVO, ed i CODICI MACCHINA che sono usati dal microprocessore.

Il MICROPROCESSORE è una delle componenti HARDWARE del COMPUTER.
Ogni computer può avere uno o più MICROPROCESSORI, che gli consentono di funzionare in un certo modo (voluto dal costruttore).



Per usare le funzioni PEEK e POKE è INDISPENSABILE conoscere quali LOCAZIONI di MEMORIA sono usate dal SISTEMA OPERATIVO ed in QUALE MODO.

L'elenco dettagliato delle LOCAZIONI di MEMORIA di un computer, con le indicazioni relative all'USO di tali LOCAZIONI, prende il nome di MAPPA DELLA MEMORIA del COMPUTER.

Se NON si ha la MAPPA DI MEMORIA del COMPUTER e NON si conosce l'uso del CODICE MACCHINA, le funzioni PEEK e POKE non possono essere usate con profitto.

È comunque possibile effettuare delle prove, ma senza risultati apprezzabili.

Occorre ricordare che l'uso della FUNZIONE POKE, se non è fatto per scopi MOLTO PRECISI, e con COMPETENZA, può causare il blocco del computer (cioè il computer NON è più in grado di riconoscere NESSUN COMANDO). Come già detto basta SPEGNERE il computer e RIACCENDERLO, per ripristinare il funzionamento.

È facilmente comprensibile che l'uso delle FUNZIONI PEEK e POKE è strettamente legato al tipo di computer usato, ed alla conoscenza della MAPPA di MEMORIA e del SISTEMA OPERATIVO del computer stesso.

Non è pertanto possibile fornire esempi generici che siano validi per tutti i computers.

Fate riferimento ai manuale del vostro computer, o ad altri testi che trattano specificatamente tali argomenti.

4 COMPLETARE L'ELENCO DELLE FUNZIONI MATEMATICHE

Abbiamo già visto che ogni computer è in grado di eseguire un certo numero di operazioni complesse che sono chiamate FUNZIONI MATEMATICHE.

In particolare sono state spiegate le FUNZIONI INT(x) RND(x) SQR(x).

Possiamo completare l'elenco delle FUNZIONI MATEMATICHE che solitamente è possibile trovare negli HOME COMPUTER.

1 ABS (arg) determina il VALORE ASSOLUTO dell'argomento indicato.
ESEMPIO: PRINT ABS(10–35) ... il computer scriverà il numero POSITIVO 25 (cioè il risultato della differenza 10–35 senza tenere conto del segno).

2 SGN (arg) determina il SEGNO dell'argomento. In questo caso il computer risponde nel modo seguente:
1 indica che il segno è positivo.
–1 indica che il segno è negativo.
0 indica che l'argomento è 0 e quindi NON HA SEGNO.
ESEMPIO: PRINT SGN(5–3) ... la risposta è 1.

3 LOG (arg) determina il LOGARITMO (solitamente in BASE e) dell'argomento.
NOTA: l'argomento DEVE ESSERE un valore MAGGIORE DI ZERO.

4 EXP(arg) determina il valore ESPONENZIALE con BASE e = 2.71828 cioè esegue la funzione inversa di LOG(arg) ed equivale all'operazione 2.71828↑(arg).

FUNZIONI TRIGOMETRICHE

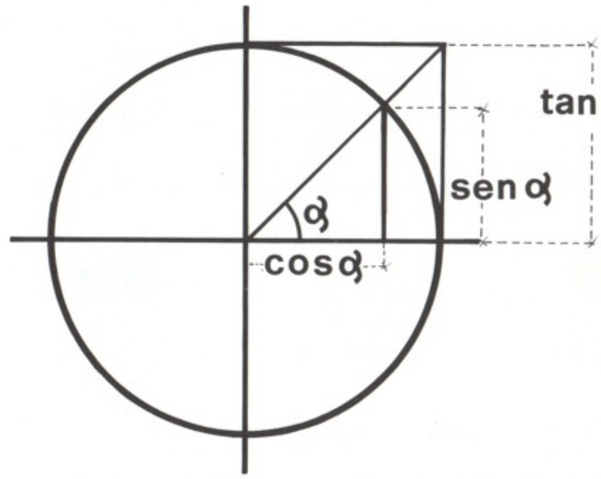
Le principali sono le seguenti:

5 SIN(arg) determina il seno dell'angolo (il cui valore è indicato con l'argomento).

6 COS(arg) determina il coseno dell'angolo.

7 TAN(arg) determina la tangente dell'angolo.

8 ATN(arg) determina il valore dell'arcotangente cioè l'angolo la cui tangente sottende l'arco di circonferenza indicato, (è la FUNZIONE INVERSA rispetto a TAN(arg).



CIRCONFERENZA TRIGONOMETRICA

NOTA:

La maggior parte dei computers usa i valori degli angoli espressi in RADIANTI e per poter usare valori in GRADI SESSAGESIMALI è necessario convertire tali GRADI in RADIANTI, ricordando che 1 GRADO SESSAGESIMALE è circa uguale a 0.0174532925 RADIANTI e 1 RADIANTE è circa uguale a 57.2957796 GRADI SESSAGESIMALI.

5 FUNZIONI PREDEFINITE DALL'UTENTE

Per quanto riguarda le FUNZIONI che si possono usare con il BASIC occorre ancora segnalare che alcuni computer permettono di definire FUNZIONI particolari.

Tali FUNZIONI sono chiamate FUNZIONI PREDEFINITE dal programmatore.

Come dice il termine è necessario che il programmatore DEFINISCA le funzioni, e questo è reso possibile dall'istruzione DEFFN.

La sintassi di tale istruzione è solitamente la seguente:

DEF FNnome(VARIABILE)= *sviluppo*

in cui DEF FN è l'istruzione che permette di DEFINIRE la funzione.

Nome è il NOME che si vuole dare alla funzione (che DEVE rispettare le regole già viste per i NOMI delle VARIABILI).

(VARIABILE) è il NOME di una qualsiasi variabile ad esempio X, e va messo tra parentesi.

Sviluppo sta ad indicare lo svolgimento della funzione, cioè le operazioni che devono essere eseguite.

ESEMPIO:
10 DEFFNA(X)= INT (X+0.9)

Questa funzione predefinita permette di ottenere (quando la si richiama) l'arrotondamento all'unità superiore del valore indicato.

Per eseguire tale funzione è sufficiente scrivere il nome della funzione indicando tra le parentesi il valore per cui deve essere calcolata (come già visto in tutte le FUNZIONI MATEMATICHE).

Per avere l'arrotondamento del numero 10.25 potremo scrivere:

20 PRINT FNA(10.25)

ed il computer scriverà il numero 11 (cioè 10.25 arrotondato alla unità superiore).

È buona norma VERIFICARE il risultato delle FUNZIONI PREDEFINITE, in quanto molti computer sono in grado di eseguire soltanto funzioni SEMPLICI, e non danno risultati validi per FUNZIONI COMPLESSE.

NOTA: Non tutti i computers danno la possibilità all'utente di definire funzioni speciali. Come sempre fate riferimento al manuale del vostro computer.

6 USO DELL'ISTRUZIONE STOP PER ESEGUIRE DUMP E DEBUG

Parlando delle tecniche di programmazione abbiamo già detto che è importante eseguire uno studio preventivo del problema che si vuole risolvere, scomponendo il programma in BLOCCHI ben definiti.

Vedremo ora alcune tecniche di programmazione che possono servire al programmatore, per VERIFICARE se il programma esegue in modo corretto le istruzioni, oppure se ci sono errori nella LOGICA del programma.

Come già detto il computer NON è in grado di VERIFICARE gli errori di LOGICA che sono commessi dal programmatore.

ESEMPIO di ERRORE LOGICO:

```
10 FOR A=1 TO 100
20 LET A = A * A : PRINT A
30 NEXT A
```

Il VALORE della variabile A viene MODIFICATO all'interno del LOOP, per cui le istruzioni NON saranno eseguite per tutti i valori da 1 a 100.

Gli esempi potrebbero essere molti, ed il programmatore deve sempre CONTROLLARE che l'esecuzione di un programma sia corretta, qualunque sia la complessità del programma stesso.

Per eseguire tali controlli alcuni computer dispongono di ISTRUZIONI di SISTEMA apposite, quali ad esempio DUMP, DEBUG, TRON etc.

Gli HOME-COMPUTER solitamente non dispongono di tali istruzioni.

Cerchiamo di capire cosa sono il DUMPING ed il DEBUGGING per poterli eseguire anche con gli HOME-COMPUTER.

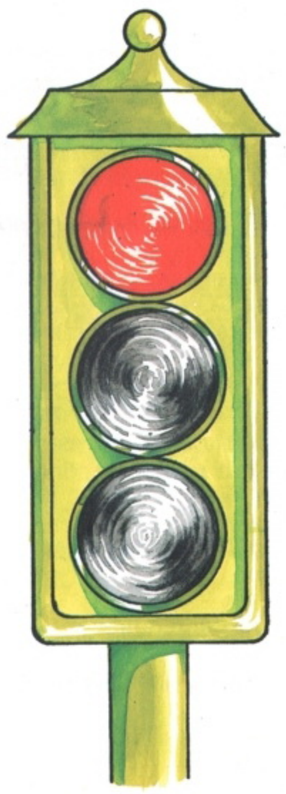
Il DUMPING riguarda le variabili e consiste in un controllo continuo del contenuto delle variabili durante l'esecuzione.
Il DEBUGGING riguarda il programma in generale e consiste nel controllo dell'ORDINE DI ESECUZIONE delle righe di un programma

DUMPING DEBUGGING

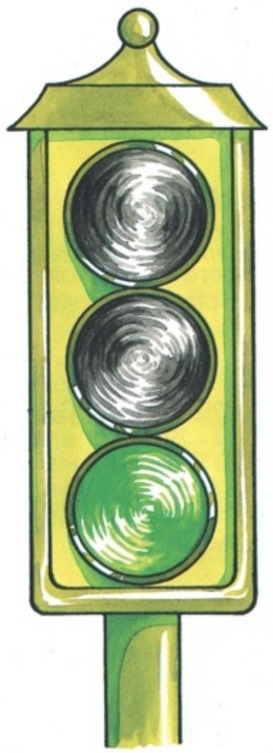
Con un HOME-COMPUTER è possibile eseguire sia il DUMP delle variabili sia il DEBUG del programma.

Naturalmente se il computer NON dispone delle istruzioni apposite la realizzazione di tali funzioni deve essere fatta dal programmatore, usando le istruzioni di cui dispone.

Le istruzioni che permettono di ottenere il DUMPING ed il DEBUGGING, su qualsiasi computer con il linguaggio BASIC, sono l'istruzione STOP, l'istruzione CONT e l'istruzione PRINT, usate nel modo che vedremo qui di seguito.



STOP



CONT

6 USO DELL'ISTRUZIONE STOP PER ESEGUIRE DUMP E DEBUG

Per poter controllare il contenuto delle variabili e l'ordine di esecuzione di un programma è possibile usare uno dei seguenti sistemi:

1 INSERIRE nel punto in cui su presuppone che si verifichi l'errore logico, una istruzione STOP.
EFFETTO: il programma si arresterà in quel punto per cui si potrà controllare il contenuto delle variabili con l'istruzione PRINT e dopo avere controllato la validità dei valori si potrà continuare ad eseguire il programma con l'istruzione CONT.

2 INSERIRE nel programma una serie di istruzioni PRINT per scrivere il contenuto delle varibili ed il numero di riga, senza interrompere l'esecuzione (oppure interrompendola con STOP).
EFFETTO: questo metodo evita di dover digitare ogni volta le PRINT necessarie al controllo delle variabili, e permette anche di sapere quale riga è stata eseguita (DEBUGGING).

```
ESEMPIO:  10 FOR A=0 TO 10 STEP B
          20 IF A=20 THEN 50
          30 A=A+10
          40 NEXT A
          50 B=B-A : A=25
          60 GOTO 10
```

Questo LOOP contiene MOLTI ERRORI di LOGICA; vediamo quali sono:

1 La variabile B usata come STEP alla riga 10 ha valore 0.

2 Nel LOOP c'è un TEST IF ... THEN che fa eseguire un SALTO CONDIZIONATO in modo scorretto (cioè SENZA CHIUDERE il LOOP).

3 All'interno del LOOP viene MODIFICATO il valore della VARIABILE A, per cui il LOOP non verrà eseguito per i valori indicati (da 1 a 10).

MODIFICHIAMO le righe per avere il DUMP delle variabili A e B, ed il DEBUG del programma:

```
10 FOR A=1 TO 10 STEP B : PRINT"RIGA 10 : A=";A"B=";B : STOP
20 IF A=20 THEN PRINT"RIGA 20 : A=";A;";B=";B : GOTO 50
30 A=A+10 : PRINT"RIGA 30 : A=";A;"B=";B
40 PRINT"RIGA 40 (PRIMA DI NEXT A) : A=";A;"B=";B : NEXT A
41 PRINT"RIGA 41 : (DOPO NEXT A DI RIGA 40) : A=";A;"B=";B
50 B=B–A : A = 25 : PRINT"RIGA 50 : A=";A;"B=";B
60 PRINT"RIGA 60 (PRIMA DI GOTO 10) : A=";A"B=";B : GOTO 10.
```

Con queste modifiche il programma potrà essere controllato RIGA per RIGA sia per quanto riguarda l'esecuzione delle righe, sia per il contenuto delle variabili A e B.

L'istruzione STOP inserita alla riga 10 permette di interrompere l'esecuzione che potrà essere continuata digitando in modo DIRETTO l'istruzione CONT (più tasto RETURN).

7 INFORMAZIONI GENERALI SUI COMPUTERS (HARDWARE E SOFTWARE)

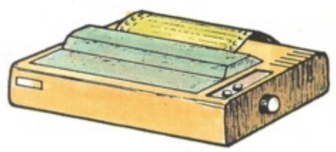
Come abbiamo ripetuto più volte il termine COMPUTER è molto generico e viene usato per identificare un insieme di apparecchiature che permettono l'elaborazione dei dati.

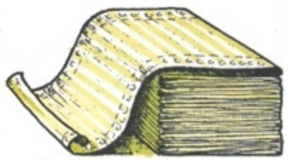
Nel corso delle lezioni abbiamo accennato ad alcune componenti HARDWARE di un sistema per l'elaborazione dati ed ora che siamo giunti al termine è necessario riparlarne per cercare di capire l'uso e l'importanza di alcune periferiche.

La STAMPANTE è una periferica molto utile in quanto permette di tenere un archivio di CARTA che può essere facilmente consultato, copiato, o trasmesso per corrispondenza.

Importante è anche il fatto che con la stampante è possibile ottenere il LISTATO dei programmi per controllare PASSO PASSO (STEP BY STEP) tutto il programma, mentre con il solo VIDEO questo non è possibile.

stampante



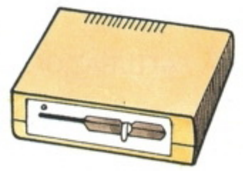


carta

Il DRIVE per DISCHI è un'altra periferica importante che permette di archiviare FILES PROGRAMMI e FILES DATI su dischi (solitamente di piccolo diametro e flessibili chiamati FLOPPY DISK).

Con il DRIVE ed i FLOPPY la velocità delle operazioni di LETTURA e SCRITTURA dei dati o dei programmi è molto più elevata, rispetto al registratore con i nastri a cassette. Inoltre la lettura e scrittura è anche molto più precisa ed affidabile.

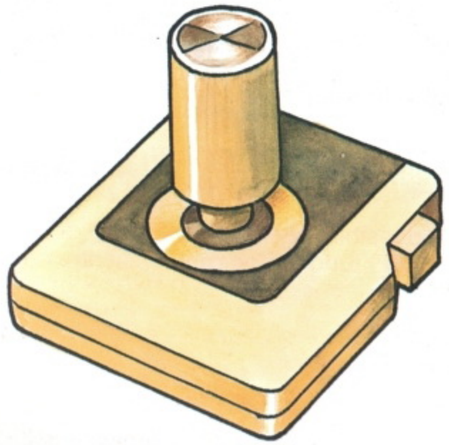
unitá a dischi



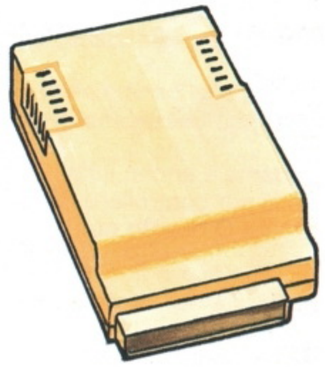
floppy disk

Altre periferiche di uso particolare sono le ESPANSIONI di MEMORIA che permettono di lavorare con una maggiore quantità di dati e di realizzare anche programmi molto complessi senza problemi di spazio di memoria.

Ci sono poi i JOYSTICK per chi intende utilizzare il proprio HOME-COMPUTER per PROGRAMMARE e ESEGUIRE GIOCHI d'azione o di movimento.



JOYSTICK



ESPANSIONE DI MEMORIA

L'elenco delle periferiche potrebbe continuare ancora per molto; citiamo ad esempio i PLOTTER, le INTERFACCE TELEFONICHE, le PENNE OTTICHE etc.

Ognuno deve valutare le prestazioni che desidera ottenere dal computer per determinare le periferiche più adatte alle proprie esigenze.

Anche per quanto riguarda il SOFTWARE è possibile trovare in commercio numerosi programmi di tipo diverso quali:

- PROGRAMMA DI UTILITÀ (UTILITIES) che permettono di AUMENTARE il VOCABOLARIO BASIC del proprio computer, oppure di usare altri linguaggi di programmazione, oppure ottenere particolari prestazioni in campo GRAFICO, MUSICALE etc.
- PROGRAMMI GESTIONALI di tipo diverso (CONTABILITÀ FATTURAZIONE etc.)
- PROGRAMMI DI DIVERTIMENTO (giochi realizzati in modo professionale).

Anche in questo caso occorre valutare opportunamente le proprie esigenze.

I COMPUTER PER L'ATTIVITÀ AZIENDALE

Per un uso professionale del computer è opportuno valutare le proprie esigenze reali, ed aventualmente rivolgersi a computers quali PERSONAL COMPUTERS o MICRO SISTEMI.

In commercio sono disponibili numerosi modelli di PERSONAL COMPUTERS, in grado di soddisfare le necessità di qualsiasi professionista, o piccola azienda.

Il personal computer trova applicazione anche nelle grandi aziende, in cui viene affiancato ai grandi sistemi di elaborazione dati come "terminale intelligente".

Per i personal computers esistono programmi (PAKAGES) professionali di grande utilità, che consentono di rendere più produttiva e razionale la gestione della propria attività.

I principali pakages in commercio riguardano i seguenti argomenti:

- WORD PROCESSING (elaborazione testi)
  Si tratta di programmi che consentono di creare un archivio di testi (lettere, relazioni, etc.), e permettono di manipolare questi testi in diversi modi.

- SPREAD SHEET (foglio elettronico)
  Il termine SPREAD SHEET significa letteralmente FOGLIO ESTESO, ma viene comunemente chiamato foglio elettronico.
  Si tratta di programmi che consentono una rapida gestione di dati, con la possibilità di realizzare statistiche, piani di produzione, bilanci, budget etc.

- DATA BASE
  Sono programmi molto sofisticati, che consentono di creare e gestire archivi di dati di qualunque tipo, permettono l'inserimento, l'aggiornamento, la ricerca, il riordino etc. di questi archivi.

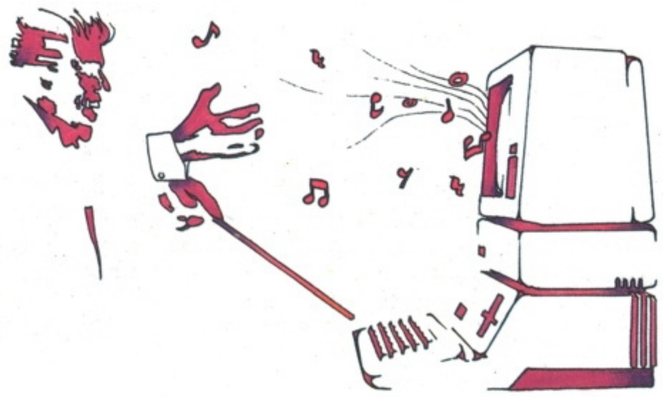
- PAKAGES INTEGRATI (programmi integrati)
  Si tratta di prodotti di notevole utilità, che assumono le caratteristiche dei programmi spiegati qui sopra, con la possibilità di scambiare o fondere i dati di un programma con quelli di uno o di tutti gli altri. È possibile ad esempio realizzare grafici a video, o su carta, sulla base dei dati inseriti in un foglio di statistica fatto con il foglio elettronico, ed inserire il tutto in una relazione scritta con il word processor.

- PROGRAMMI GESTIONALI
  Si tratta di programmi per la gestione della azienda, ed hanno caratteristiche di alta affidabilità. È possibile trovare programmi di tipo specifico, o generico, per contabilità, o per la gestione di magazzino, per la realizzazione di paghe, per l'amministrazione di condomini etc.

8 CENNI SULLE FUNZIONI GRAFICHE E SONORE

Molti HOME-COMPUTERS possiedono caratteristiche GRAFICHE e SONORE.

Questo significa che è possibile realizzare programmi che si servono di tali caratteristiche per ottenere una migliore prestazione, o programmi basati principalmente su tali caratteristiche a scopo di insegnamento (didattica) o divertimento (gioco).



Per poter usare le caratteristiche GRAFICHE o SONORE del proprio computer è necessario consultare il manuale per verificare se esistono istruzioni di tipo "EVOLUTO" o se è necessario usare il CODICE MACCHINA.

Nel primo caso si potranno trovare istruzioni (o meglio FUNZIONI) grafiche e sonore, ed ogni FUNZIONE avrà la propria SINTASSI.

Citiamo ad esempio i nomi più comuni di alcune funzioni:

SOUND PLOT COLOR DRAW LINE MODE CIRCLE SET RESET

Ognuna di tali FUNZIONI ha un preciso scopo ed una propria sintassi.

Naturalmente per utilizzare con profitto tali possibilità è necessario conoscere le basi fondamentali della geometria (per la grafica) o della musica (per il suono).

Senza tali basi è comunque possibile usare le caratteristiche grafiche e sonore digitando programmi realizzati da altri oppure servendosi dei programmi in commercio già registrati su nastri o dischi.

Come per altri argomenti già trattati precisiamo che i NOMI e la SINTASSI delle funzioni grafiche e sonore possono cambiare da un computer ad un altro, per cui è indispensabile consultare il manuale del proprio computer.