

30

PROGRAMMI

BASIC

PER LO ZX 80



LIBRERIA ELETTRONICA



Guida mondiale dei circuiti integrati TTL
L. 20.000 (Abb. L. 18.000) **Cod. 6010**

Costruiamo un vero microelaboratore elettronico
L. 4.000 (Abb. L. 3.600) **Cod. 3000**

Appunti di elettronica
L. 8.000 (Abb. L. 7.200) **Cod. 2300**

Selezione di progetti elettronici
L. 9.000 (Abb. L. 8.100) **Cod. 6008**

Digit 2
L. 6.000 (Abb. L. 5.400) **Cod. 6011**

Transistor cross-reference guide
L. 8.000 (Abb. L. 7.200) **Cod. 6007**

Tabelle equivalenze semiconduttori tubi professionali
L. 5.000 (Abb. L. 4.500) **Cod. 6006**

Corso di progettazione dei circuiti a semiconduttore
L. 8.400 (Abb. L. 7.560) **Cod. 2002**

300 circuiti
L. 12.500 (Abb. L. 11.250) **Cod. 6009**

Le radiocomunicazioni
L. 7.500 (Abb. L. 6.750) **Cod. 7001**

Alla ricerca dei tesori
L. 6.000 (Abb. L. 5.400) **Cod. 8001**

Le luci psichedeliche
L. 4.500 (Abb. L. 4.050) **Cod. 8002**

Accessori elettronici per autoveicoli
L. 6.000 (Abb. L. 5.400) **Cod. 8003**

Manuale di sostituzione dei transistori giapponesi
L. 5.000 (Abb. L. 4.500) **Cod. 6005**

100 riparazioni TV illustrate e commentate
L. 10.000 (Abb. L. 9.000) **Cod. 7000**

Sommario

| | |
|--|-----------|
| PREFAZIONE | 1 |
| SCHEDA DEL SINCLAIR | 3 |
| GIOCO DELLE RANE della <i>Dr.ssa R. Bonelli</i> | 4 |
| MESSAGGI SEGRETI della <i>Dr.ssa R. Bonelli</i> | 5 |
| GIOCO DEI RESTI della <i>Dr.ssa R. Bonelli</i> | 6 |
| GOMOKU di <i>M. Oliva</i> | 7 |
| LIFE di <i>A. Napoletano</i> | 13 |
| BATTAGLIA NAVALE di <i>E. Vighi</i> | 15 |
| BLACK JACK di <i>E. Vighi</i> | 19 |
| ROUTINE ASSEMBLER di <i>M. Oliva</i> | 22 |
| DISEGNI di <i>A. Napoletano</i> | 29 |
| NIM di <i>S. Nichelini</i> | 32 |
| ALLUNAGGIO di <i>S. Gioia</i> | 34 |
| PINCK di <i>L. Castagna</i> | 35 |
| CAPITALI di <i>E. Vighi</i> | 39 |
| BUBBLE SORT di <i>G. Rocca</i> | 42 |
| SISTEMA 4 EQUAZIONI IN 4 INCOGNITE di <i>M. Oliva</i> | 43 |
| LABIRINTO di <i>R. Rozzi</i> | 50 |
| HAI MEMORIA? di <i>P. Ciceri</i> | 52 |
| CARATTERI 8x8 di <i>P. Ciceri</i> | 54 |
| CORSA DI CAVALLI di <i>S. Papes</i> | 55 |
| CARICATORE ASSEMBLER di <i>P. Ciceri</i> | 56 |
| NUMERI PRIMI di <i>S. Papes</i> | 59 |
| RIPASSO TABELLINE di <i>R. Gibelli</i> | 60 |
| CHOMP di <i>N. Cremaschi</i> | 62 |
| L'IMPICCATO di <i>P. Ciceri</i> | 64 |
| GIORNI DELLA SETTIMANA di <i>A. Planamente</i> | 67 |
| GIOCO DEL TRIS di <i>M. Lefons</i> | 69 |
| MASTER MIND di <i>G. Bortone</i> | 73 |
| IMPARIAMO LA MATEMATICA della <i>Dr.ssa R. Bonelli</i> | 74 |
| RADICE QUADRATA CON 3 DECIMALI di <i>G. Bortone</i> | 75 |
| SISTEMA 2 EQUAZIONI IN 2 INCOGNITE di <i>G. Bortone</i> | 76 |

Supplemento al n. 5 di

SPERIMENTARE



JACOPO CASTELFRANCHI EDITORE

© Tutti i diritti di riproduzione e traduzione degli articoli pubblicati sono riservati.



Mensile associato all'USPI
Unione Stampa Periodica Italiana

Rivista mensile di elettronica pratica; Editore: J.C.E. - Direttore responsabile: RUBEN CASTELFRANCHI - Direttore Editoriale: GIAMPIETRO ZANGA - Capo redattore: GIANNI DE TOMASI - Redazione: SERGIO CIRIMBELLI, DANIELE FUMAGALLI, TULLIO LACCHINI, MARTA MENEGARDO - Grafica e impaginazione: MARCELLO LONGHINI - Laboratorio: ANGELO CATTANEO, LORENZO BARRILE - Contabilità: ROBERTO OSTELLI, M. GRAZIA SEBASTIANI - Diffusione e abbonamenti: PATRIZIA GHIONI - Collaboratori LUCIO VISINTINI, FILIPPO PIPITONE, LUCIO BIANCOLI, FEDERICO CANCARINI, LODOVICO CASCIANINI, SANDRO GRISOSTOLO, GIOVANNI GIORGINI, ADRIANO ORTILE, AMADIO GOZZI, PIERANGELO PENSA, GIUSEPPE CONTARDI - Direzione, Redazione, Amministrazione: Via dei Lavoratori, 124 - 20092 Cinisello Balsamo - Milano, Telefono 6172671 - 6172641. - Sede legale: Via Vincenzo Monti, 15 - 20123 Milano. - Autorizzazione alla pubblicazione: Tribunale di Monza, numero 258 del 28-11-1974. - Stampa Litografia del Sole - Albairate (MI). - Concessionario esclusivo per la diffusione in Italia e all'Estero SODIP - Via Zuretti, 25 - 20125 Milano. - Spedizione in abbonamento postale gruppo III/76.

PREFAZIONE

Lo ZX-80 non è frutto di una tecnologia particolare: poteva essere fatto più o meno identico già cinque o dieci anni fa! Il Sinclair ZX-80 è *eccezionale* perchè fare un computer semplice, anzi semplicissimo è molto difficile! Gli ingredienti sono i soliti: un microprocessore (lo Z-80), memorie RAM e ROM, una manciata di componenti. Ma (e qui sta la genialità) presentate tutto questo in un contenitore che può stare in una tasca, munitelo di una tastiera completa, dategli la possibilità di collegarsi direttamente con un comune televisore e con un registratore, fatelo parlare in BASIC, e otterrete il Sinclair ZX-80.

Ma non basta: per chi non ama le scatole chiuse, per chi vuole sapere com'è fatto un computer, lo ZX-80 è fornito in tutte le sedi GBC anche in kit. Vuol dire cioè, partendo dal gradino più basso (il montaggio), arrivare subito ad un oggetto immediatamente e comodamente utilizzabile.

Lo ZX-80 è realizzato su un'unica piastra. Il circuito stampato fa da supporto anche alla tastiera, che occupa circa un terzo dell'intera piastra. In pratica ogni tasto è realizzato con due piste molto vicine fra loro: la pressione del dito su uno dei tasti "disegnati" fa scendere della plastica conduttrice, che stabilisce il collegamento tra le due piste sottostanti. Semplice, vero? Ma soprattutto una soluzione poco costosa rispetto, ad esempio, ad una tastiera del tipo capacitivo, con cui a prima vista può essere confusa.

Il resto del circuito stampato è occupato dai componenti: pochi in verità (solo 22 circuiti complessivamente), il che fra l'altro permette un popolamento della scheda non particolarmente fitto. Questa circostanza è evidenziata per due motivi: un popolamento fitto creerebbe in primo luogo problemi di dissipazione di calore, ed in secondo luogo richiederebbe in fase di montaggio un'attenzione maggiore (ricordo ancora che lo ZX-80 è fornito *anche* in kit).

Per partire è sufficiente collegare il sistema ad un comune televisore, e quindi alimentarlo con una tensione di 9 V. Dopo essersi sintonizzati sul canale 36 UHF, comparirà, sul fondo dello schermo, il cursore, rappresentato dalla lettera K (visualizzata in modalità inversa) e posizionata nell'angolo in basso a sinistra. Sul video sono rappresentabili 23 linee di 32 caratteri ciascuna.

Il Sinclair ZX-80 lavora direttamente in BASIC residente in una ROM da 4 Kbyte. In realtà nella ROM sono contenuti anche i programmi di gestione delle varie periferiche (video, cassetta magnetica e tastiera), nonché i dati che definiscono le configurazioni dei caratteri alfanumerici e dei caratteri grafici che compaiono sul video, dato che manca l'apposito integrato generatore dei caratteri, di solito presente.

Ma alla Sinclair non si sono fermati a questo punto: sempre nella medesima ROM risiede un originale e potente editor le cui caratteristiche più significative sono quelle di operare la distinzione tra linea in fase di impostazione e linea già impostata, e di fare un controllo sulla validità della sintassi della linea impostata, già in fase di editazione, senza cioè attendere il momento dell'esecuzione. A ciò bisogna aggiungere che la maggior parte dei tasti ha associati due o tre significati: se non è premuto il tasto SHIFT è generata la relativa lettera (maiuscola), diversamente (cioè con SHIFT premuto) sarà generato il carattere grafico o il segno d'interpunzione associato al tasto medesimo. E fin qui nulla di nuovo, a parte il fatto di poter contare su un insieme (anche se limitato) di simboli grafici. *La novità* è che il sistema riconosce quando si è nella fase d'introduzione delle parole-chiave del BASIC, per cui predispone l'intera tastiera (esclusa la linea dei tasti superiori) in modo che ogni tasto abbia il significato della parola-chiave BASIC che compare sul relativo layout di tastiera. Allora, per esempio, per scrivere l'istruzione 10 PRINT "CIAO", è sufficiente, dopo aver dichiarato l'identificatore di linea, premere il tasto con la lettera O per vedere immediatamente per intero su video la parola-chiave PRINT, dopo la quale si posiziona il cursore ad indicare la posizione dove verranno posti i successivi caratteri. A questo punto occorre un'ulteriore precisazione: ho detto prima che il cursore è rappresentato dalla lettera K in modalità inversa. Ebbene, la lettera K sta per *keyword* (che in inglese vuol dire parola-chiave), e sta ad indicare quando la tastiera è predisposta all'introduzione delle parole-chiave: infatti, *dopo* aver premuto il tasto O (cioè PRINT), il cursore è rappresentato non più dalla lettera K ma dalla lettera L (sempre in modalità inversa).

Eccezionale, vero? In altre parole, il sistema comunica in che stato è la tastiera, per non generare confusione! Infatti, quando il cursore è rappresentato da L, alla tastiera sono associati gli abituali caratteri alfanumerici e grafici. Allora, tornando all'esempio precedente, possiamo tranquillamente completare l'istruzione battendo carattere per carattere, le virgole e la parola CIAO.

La chiusura dell'istruzione è fatta premendo il tasto NEWLINE: solo a questo punto l'istruzione è memorizzata nella memoria di programma. Questo passaggio è segnalato in modo evidente: la linea in fase di editaggio è sempre visualizzata nell'*ultima* riga del video, mentre, dopo il trasferimento nella memoria di programma, è visualizzata in alto nello schermo in coda alle eventuali istruzioni già introdotte. In questo consiste la distinzione tra linea in corso d'impostazione (quella che compare nell'ultima riga del video) e linea già impostata (quella o quelle che compaiono nella parte superiore dello schermo), segnalata in precedenza.

Accanto a questa utile modalità, l'editor delle ZX-80 presenta un'altra interessante caratteristica: controlla i caratteri introdotti nella linea in fase d'impostazione allo scopo di segnalare immediatamente errori di sintassi. In caso di errore infatti l'introduzione è bloccata, mentre il cursore da K o L diventa S (come syntax). Questa caratteristica permette di ottenere dalla fase di editaggio sempre linee corrette dal punto di vista sintattico, per cui durante l'iniziale esecuzione del programma si possono efficacemente analizzare solo gli eventuali errori logici fatti durante la concezione del programma. Questi brevi cenni mostrano come le funzionalità di editing siano particolarmente sofisticate in riferimento al tipo di sistema: senz'altro sono da riportare ad una volontà di agevolare l'uso dello ZX-80 da parte di utilizzatori inesperti.

Ad eccezione comunque, delle funzioni matematiche e trigonometriche, si contano ben 34 tra istruzioni riportate nella scheda successiva, funzioni e comandi, che, pur con una serie di varie limitazioni, permettono comunque una programmazione abbastanza elastica.

A tutto questo occorre aggiungere la considerazione più importante, quella del prezzo: il prezzo dello ZX-80, montato o in kit, è comunque uno dei più bassi in assoluto.

Ecco, sono questi elementi quelli, che hanno fatto sì che questo prodotto riuscisse ad imporsi sui vari mercati esteri tanto rapidamente. È chiaro comunque che il fornire un computer a basso costo si riflette sulle prestazioni; ma il nostro parere è che il discorso sulle prestazioni diventa di secondaria importanza quando il fatto principale è che prodotti come lo ZX-80 permettono a tutti, dato il costo, di lavorare con un computer, capire che cosa si può fare con un computer.

Quest'ultima considerazione è fatta con riferimento soprattutto alle numerose richieste di lettori che, desiderando iniziare a lavorare con un computer, sono fermati da cifre non accessibili a tutti, o ancora da parte di tutti coloro che vogliono imparare a programmare in BASIC. Ora, noi riteniamo che in questi casi iniziare a lavorare con un computer, vedere sperimentalmente che cosa un computer può o non può fare sia veramente produttivo, didattico, efficace, soprattutto se si dispone, come in questo caso di un libro-manuale quale "Impariamo a programmare in BASIC con lo ZX-80" del Gruppo Editoriale Jackson, in vendita presso la Jackson stessa, le migliori librerie e tutte le sedi GBC con il codice TL/1450-01 al prezzo di L. 4.500.

Forse è veramente arrivato il momento in cui ciascuno può avere il proprio computer personale!

Poiché tutti i motivi precedentemente detti fanno dello ZX-80 "un computer" da sperimentare, con questo volume vogliamo fare conoscere 30 programmi applicativi, già fatti "girare" con successo. Essendo programmi *pronti all'uso* si rivolgono soprattutto ai non programmatori, quale valido ausilio didattico, nonché prima implementazione del BASIC studiato, ma possono essere, da parte dei più esperti, anche base di partenza per ulteriori elaborazioni. Buoni programmi!

Rita Bonelli

La scheda del Sinclair ZX-80

Variabili:

Intere primo carattere alfabetico, caratteri successivi o cifre o lettere senza spazi, contenuto numeri interi compresi tra — 32768 e + 32767.

Stringhe: nome formato da una lettera seguita da \$ (dollaro), non c'è limite al numero dei caratteri contenuti.

Costanti

Intere: numeri compresi fra — 32768 e + 32767.

Stringhe: delimitate dagli apici, lunghezza a piacere, possono contenere qualunque carattere salvo gli apici.

Variabili con indice:

Intere: nome formato da una sola lettera, un solo indice e come indice espressione intera.

Variabili di controllo:

Intere: e con nome formato da una sola lettera.

Espressioni aritmetiche:

Operatori aritmetici:

- ** (elevato a)
- (unitario)
- * prodotto
- / divisione
- + somma (somma e sottrazione non hanno ordine di precedenza tra loro)
- sottrazione

uso delle parentesi
ordine di valutazione da sinistra a destra con la precedenza con la quale sono stati listati gli operatori.

Espressioni relazionali:

Operazioni relazionali

= > < (senza ordine di precedenza tra loro)
Valore — 1 per condizione vera; 0 per condizione falsa.

Operatori logici:

NOT, AND, OR (le precedenze sono quelle date dalla lista)

Espressioni Booleane:

usano gli operatori logici.

Istruzioni:

NEW inizializza il calcolatore e cancella la memoria.

LOAD carica programmi e dati da nastro magnetico.

SAVE memorizza programmi e dati su nastro magnetico.

RUN manda in esecuzione un programma azzerando le variabili.

RUN n come sopra, ma con partenza dalla linea n.

CONTINUE fa continuare da n se n è nel messaggio del sistema, fa continuare da n + 1 dopo uno STOP.

REM commenti a scopo documentativo.

IF n THEN istruzione esegue istruzione se la condizione n è vera.

INPUT dest

legge e memorizza in dest.

PRINT lista

stampa il contenuto di lista, separatori di campo; e.

LIST n

lista il programma con il puntatore di linea ad n.

LIST

lista il programma dall'inizio.

STOP

ferma il programma, per continuare CONTINUE.

DIM A(n)

predispone una variabile numerica con indice formata da n + 1 elementi.

FOR K = n1 TO n2

gestisce con il contatore K un ciclo per valore iniziale di K = n1 e valore finale di K = n2 dando ad ogni giro l'incremento di 1 a K.

GOTO n

salta alla linea n.

POKE n1, n2

scrive all'indirizzo n1 il valore n2.

RANDOMISE n

pone l'inizio per la generazione dei numeri a caso al valore n.

RANDOMISE

come sopra, ma n = valore del contatore dei fotogrammi dello schermo.

CLEAR

cancella tutte le variabili.

CLS

azzerla la parte superiore dello schermo.

GOSUB n

come GOTO, ma conserva nello STACK l'indicazione per ritornare al programma principale.

RETURN

fa prelevare dallo STACK l'indicazione per tornare al programma principale.

NEXT K

chiude il ciclo iniziato da FOR, incrementa K e ne controlla il valore.

LET

consente di fare qualunque operazione di assegnazione o di calcolo.

Esiste il tasto BREAK per interrompere l'esecuzione di un programma se non è in attesa di INPUT.

Funzioni implementate:

RND(n)

genera un numero pseudo-random minore o uguale a n. La sequenza è influenzata nel punto di partenza da RANDOMISE.

ABS (espress.)

fornisce il valore assoluto della espressione.

PEEK(n)

fornisce il contenuto del byte di memoria di indirizzo n.

USR(n)

permette di andare ad eseguire un codice macchina memorizzato a partire da n.

CHR\$(x)

fornisce il carattere corrispondente al codice numerico x

TL\$ (stringa)

ritorna la stringa senza il suo primo carattere.

CODE (stringa)

fornisce il codice numerico del primo carattere della stringa.

STR\$(x)

fornisce una stringa di caratteri corrispondente al numero x

IL GIOCO DELLE RANE

Autore: Dr.ssa R. Bonelli
Programma utilizzante:
1 K di memoria



Si hanno 9 caselle, numerate da 1 a 9 e 8 rane, 4 nere e 4 bianche. Lo stato iniziale delle rane è quello dello schema:

```

      0 0 0 0
    1 2 3 4 5 6 7 8 9
  
```

Il giocatore muove una rana per volta, dando le coordinate (numero $N=XY$) della posizione di partenza (X) e di quella di arrivo (Y), nella forma XY .

Le mosse consentite sono:

- una rana può passare nella casella vicina, se vuota;
- una rana può scavalcare un'altra rana andando ad occupare una casella vuota.

Si deve raggiungere, con il minor numero di mosse, la situazione di avere o tutte le rane a destra, o tutte le rane a sinistra, senza caselle vuote tra loro e con tutte le rane dello stesso colore vicine tra loro.

I movimenti iniziali possibili sono: $XY=35$, $XY=45$, $XY=65$, $XY=75$.

le variabili del programma sono:

$P(I)$ con $I = 1, 2, \dots, 8, 9$ per le caselle;

C = contatore delle mosse del giocatore;

N = numero che dà la mossa;

X = posizione di partenza della rana;

Y = posizione di arrivo della rana;

$R\$$ = risposta SI o NO

$P(I)$ contiene 129 se occupato da rana nera e 128 se occupato da rana bianca, contiene zero se vuoto.

Quando il giocatore ha vinto, il programma chiede se vuol giocare ancora; se la risposta è SI, il gioco ricomincia.

```

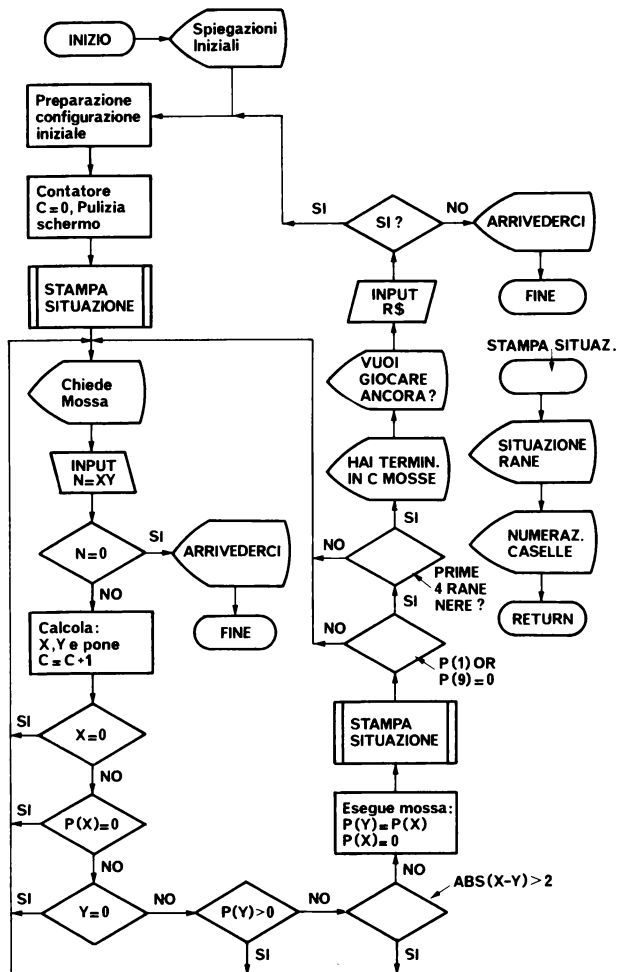
10 REM GIOCO RANE
20 PRINT "GIOCO DELLE RANE"
21 PRINT "SI MUOVE SCRIVENDO N DI 2 CIFRE"
22 PRINT "N=XY; X=POSIZIONE PARTENZA"
23 PRINT "Y=POSIZIONE ARRIVO"
30 DIM P(9)
40 LET P(5) = 0
50 FOR I = 1 TO 4
60 LET P(I) = 129
70 LET P(I+5) = 128
80 NEXT I
100 LET C = 0
150 CLS
200 GOSUB 1000
300 PRINT "MUOVI"
310 INPUT N
320 IF N = 0 THEN GO TO 550
330 LET X = N/10
340 LET Y = N-10*X
345 LET C = C+1
350 IF X = 0 THEN GO TO 300
360 IF P(X) = 0 THEN GO TO 300
370 IF Y = 0 THEN GO TO 300
380 IF P(Y) > 0 THEN GO TO 300
390 IF ABS(X-Y) > 2 THEN GO TO 300
400 LET P(Y) = P(X)
410 LET P(X) = 0
450 GO SUB 1000
460 IF P(1)=0 OR P(9) = 0 THEN GO TO 470
465 GO TO 300
470 LET X = 2
480 IF P(9) = 0 THEN LET X=1
490 FOR I = X TO X + 3
495 IF NOT P(I) = 129 THEN GO TO 300
  
```



```

500 NEXT I
510 PRINT "HAI TERMINATO IN ";C;" MOSSE"
520 PRINT "VUOI GIOCARE ANCORA?"
530 INPUT R$
540 IF CODE(R$) = 56 THEN GO TO 40
550 PRINT "ARRIVEDERCI"
560 STOP
1000 REM SOTTOPROGRAMMA STAMPA SITUAZIONE
1005 FOR I = 1 TO 9
1010 PRINT CHR$( P(I)); "Δ";
1020 NEXT I
1030 PRINT
1040 FOR I = 1 TO 9
1050 PRINT I; "Δ";
1060 NEXT I
1070 PRINT
1080 RETURN

```



Autore: **Dr.ssa R. Bonelli**
Programma utilizzante:
1 K di memoria

**MESSAGGI
SEGRETI**



MESSAGGI SEGRETI

Con questo programma si possono preparare messaggi segreti, provvedendo a una decodifica del messaggio. La chiave per trasformare i messaggi è il generatore dei numeri pseudo random dello ZX80. Inoltre l'utente del programma dà un numero negativo, o positivo in valore assoluto minore di 256, per innescare la decodifica o la codifica. Il messaggio decodificato può essere capito solo da chi possiede lo ZX80 e questo programma.

Le variabili del programma sono:

A\$ per contenere il messaggio;

T = numero negativo o positivo scelto ($-255 < T < -1$);
o ($1 < T < 255$)

B = caratteri singoli del messaggio in fase di lavoro,
R = variabile di comodo.

Alla fine del programma viene chiesto se si vuol continuare; per continuare rispondere SI.

```
10 REM MESSAGGI SEGRETI
50 PRINT "MESSAGGI SEGRETI"
100 PRINT "SCRIVI UN MESSAGGIO"
110 INPUT A$
120 PRINT A$
130 PRINT "SCRIVI IL NUMERO - T PER
    DECODIFICARE. +T PER CODIFICARE"
135 PRINT "T DEVE ESSERE IN VALORE ASSOLUTO
    MINORE DI 256"
140 INPUT T
150 PRINT T
160 RANDOMISE ABS(T)
170 IF T < 0 THEN PRINT "DE";
180 PRINT "CODIFICA MESSAGGIO"
190 LET R = RND(26)
200 IF T < 0 THEN LET R = 26 - R
210 IF A$ = " " THEN GO TO 300
220 LET B = CODE(A$)
230 LET A$ = TL$(A$)
240 IF B = 0 THEN GO TO 280
250 LET B=B+R - 38
260 LET B=B - 26 x (B/26)
270 LET B=B+38
280 PRINT CHR$(B);
290 GO TO 190
300 PRINT
310 PRINT "ALTRO MESSAGGIO?"
320 INPUT A$
330 IF NOT CODE(A$) = 56 THEN GO TO 360
340 CLS
350 GO TO 100
360 PRINT "ARRIVEDERCI"
370 STOP
```

IL GIOCO DEI RESTI

Autore: Dr.ssa R. Bonelli
Programma utilizzante:
1 K di memoria

PENSATE UN NUMERO N compreso tra 1 e 315, ma non lo date al calcolatore.

Il programma vi chiede: - il resto di N/5
- il resto di N/7
- il resto di N/9.

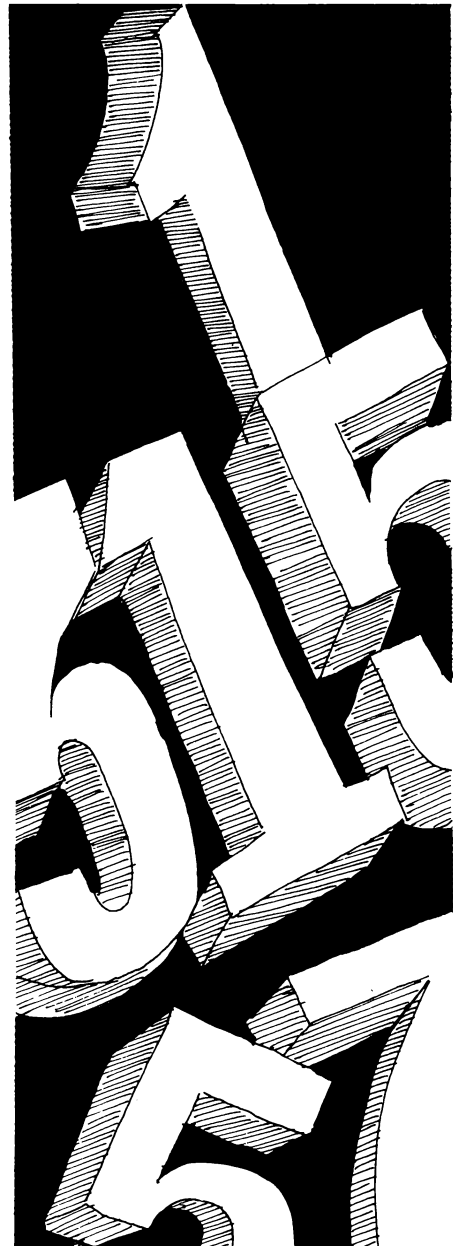
Applicando un algoritmo il programma vi dice quale numero avete pensato.

La formula di calcolo è:

$$N = 126 \times R(1) + 225 \times R(2) + 280 \times R(3)$$

$$N = N - 315 \times (N/315)$$

dove R(i) sono i 3 resti richiesti.



Le variabili del programma sono:

R(I) I = 1, 2, 3, per i 3 resti

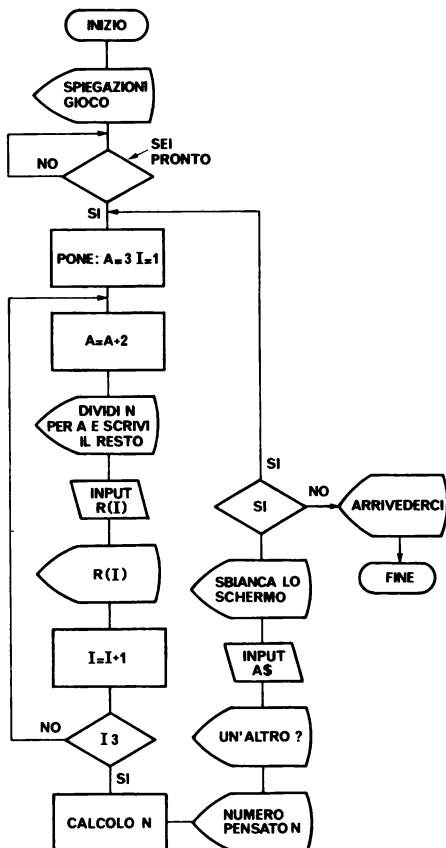
A\$ per le risposte

A per il numero calcolato.

```

10 REM GIOCO DEI RESTI
20 PRINT "GIOCO DEI RESTI"
30 PRINT "PENSA UN NUMERO N COMPRESO TRA
  1 E 315"
40 PRINT "PREMI (NEW LINE) QUANDO SEI
  PRONTO"
50 INPUT A$
100 DIM R(3)
200 LET A = 3
210 FOR I = 1 TO 3
220 LET A = A + 2
230 PRINT "DIVIDI N PER"; A
240 PRINT "SCRIVI IL RESTO"
250 INPUT R(I)
260 PRINT R(I)
270 NEXT I
300 LET A = 126 * R(1) + 225 * R(2) + 280 * R(3)
310 LET A = A-315*(A/315)
320 PRINT "NUMERO PENSATO: N="; A
330 PRINT "un altro?"
340 INPUT A$
350 CLS
360 IF CODE(A$) = 56 THEN GO TO 100
370 PRINT "ARRIVEDERCI"
380 STOP

```



GOMOKU

Autore: M. Oliva

Programma utilizzante:

4 K di memoria

Questo è un gioco giapponese che si svolge tra due giocatori su una scacchiera (di solito di 19x19 caselle). Obiettivo di ciascuno di essi è di porre il proprio simbolo (uno zero o una crocetta) in un certo numero di caselle consecutive orizzontalmente, verticalmente o diagonalmente sulla scacchiera. Vince il giocatore che per primo realizza la serie voluta.

Nella versione presentata, la scacchiera è composta di 49 caselle (7 righe per sette colonne). Il gioco avviene tra un giocatore ed il calcolatore. La prima mossa spetta al giocatore il cui simbolo è costituito dal carattere "■" a differenza di quello del calcolatore "x".

Il numero di simboli consecutivi richiesti per vincere è di 5, siano essi posti su una riga, su una colonna o su una diagonale.

È bene comunque precisare che la partita svolta dal calcolatore è di tipo difensivo cioè tale da impedire al giocatore di realizzare la serie vincente e non di vincere la partita.

Ciò non toglie però al calcolatore la possibilità di sconfiggere il giocatore o di pareggiare.

Dopo avere scritto il programma seguendo la procedura che più avanti sarà illustrata è possibile dare il comando di < RUN >.

Allora sullo schermo comparirà la scacchiera di gioco le cui caselle, inizialmente vuote sono indicate dal carattere "■", la scacchiera rimarrà visualizzata continuamente nel corso della partita.

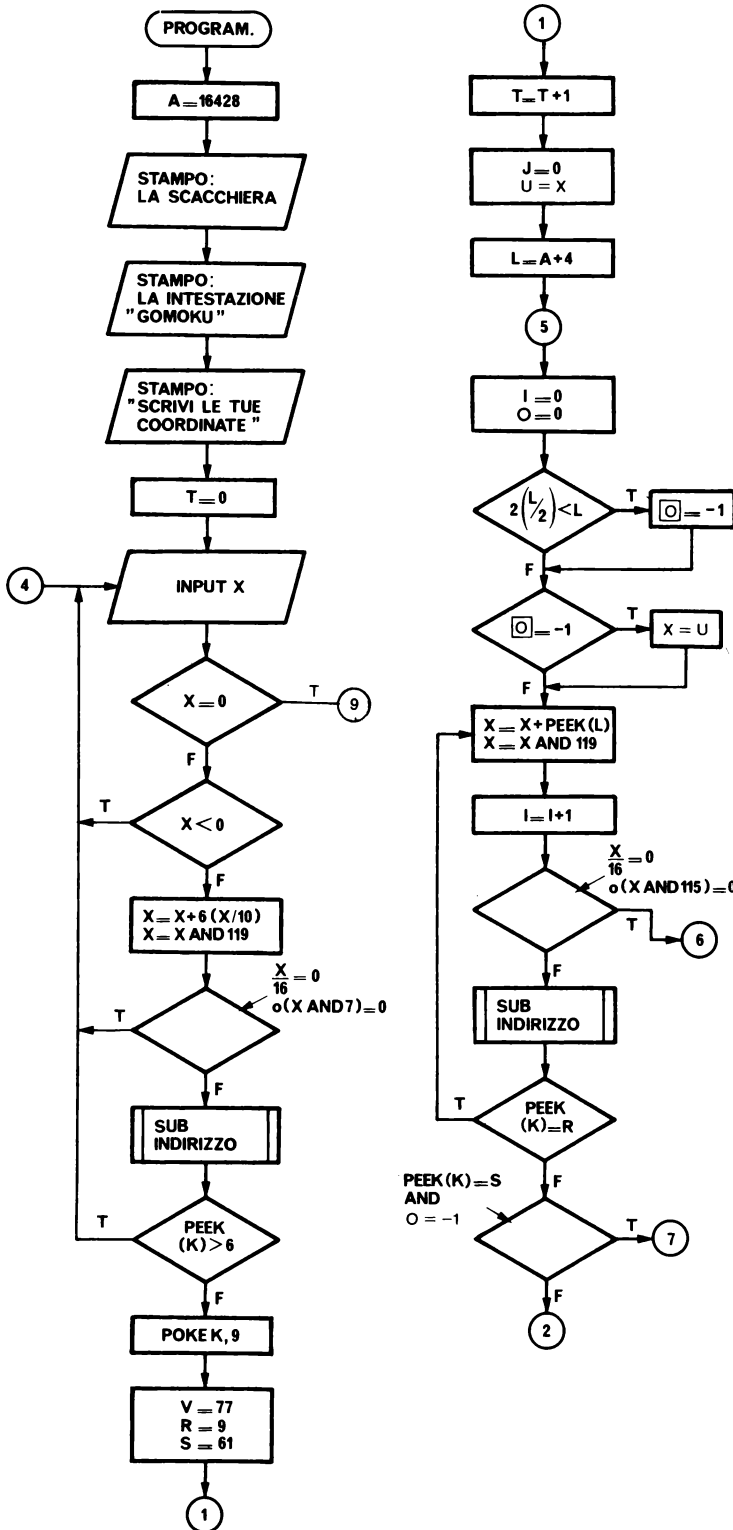
Alla richiesta delle coordinate il giocatore deve introdurre quelle relative alla mossa che intende eseguire che devono così essere indicate: riga e colonna. Ad esempio "23" indica la casella individuata dalla seconda riga e dalla terza colonna. Qualora le coordinate inserite dal giocatore siano inesatte oppure si riferiscono a caselle già utilizzate viene effettuata una nuova richiesta. La mossa del giocatore, così introdotta, viene indicata sulla scacchiera dal carattere "■".

Se questa mossa non è vincente il calcolatore risponde opportunamente ponendo in una certa casella libera il proprio simbolo "x".

Quando o il giocatore o il calcolatore riescono a comporre la sequenza di 5 simboli consecutivi il gioco ha termine segnalando il vincitore e il numero di mosse che sono state necessarie per risolvere la partita.

La partita termina invece in parità se dopo l'ultima mossa del giocatore (la 25^a) nessuna sequenza vincente è stata realizzata, oppure quando alla richiesta delle coordinate il giocatore, volendo interrompere la partita, introduce il valore "0" (zero).

GOMOKU



STRUTTURA DEL PROGRAMMA E SUO CARICAMENTO

Il programma che realizza il gioco "Gomoku" ha una struttura un poco complessa che deve essere attentamente esaminata prima della sua realizzazione. Per prima cosa il programma non tratta la scacchiera come un normale vettore. Utilizza invece l'area di memoria riservata al video. Infatti il programma, nella prima fase, visualizza la scacchiera, ciò consente poi di riempire e di controllare le caselle direttamente nell'area di memoria corrispondente allo schermo video. Per far ciò il programma utilizza una breve subroutine la quale traduce le coordinate di una casella della scacchiera in un indirizzo di memoria nella cui locazione è contenuto il carattere grafico visualizzato. Tale routine non è altro che una formuletta:

$$K = \text{USR}(A) + 2 * (X \text{ AND } 15) - 24 * (X/16) + 175$$

dove X è la coordinata della casella e K è l'indirizzo che si vuole determinare.

In questa formula si nota la presenza della funzione

USR(A)

che richiama una routine in linguaggio macchina a partire dalla locazione A (A=16428).

La routine è la seguente, con relativa codifica;

| | |
|----------------|-----|
| LD HL, (16396) | 42 |
| | 12 |
| | 64 |
| RET | 201 |

Essa fa sì che dopo la sua esecuzione si possa disporre del contenuto del puntatore D-File (di indirizzo 16396 e 16397) che indica l'indirizzo del primo dei byte di memoria componenti l'area video.

Più avanti vedremo come si introdurrà questa routine assembler.

La seconda fase consiste nella introduzione delle coordinate della mossa del giocatore e nella sua analisi di validità.

Ipotizzata corretta la mossa del giocatore, subentra la fase di controllo di essa.

Spiegare quale è l'algoritmo che consente di individuare se la mossa è vincente e di indicare l'eventuale risposta da parte del calcolatore è assai complesso per questo si rimanda al listato e al diagramma a blocchi del programma.

In ogni caso in questa fase vengono utilizzate delle costanti di incremento le quali saranno introdotte nella fase di realizzazione del programma.

A seconda del risultato scaturito da questa fase il programma finisce se il giocatore ha vinto, si è ritirato o si è esaurita la scacchiera, oppure prosegue con la mossa del calcolatore.

Ora si presentano due casi, se la fase precedente ha consentito di individuare una casella nella quale il calcolatore possa introdurre il proprio simbolo ciò è fatto subito, altrimenti le coordinate della sua mossa sono determinate mediante l'utilizzo della funzione RND tale da generare una mossa casuale.

La mossa così fatta viene anch'essa esaminata per vedere se è vincente o meno.

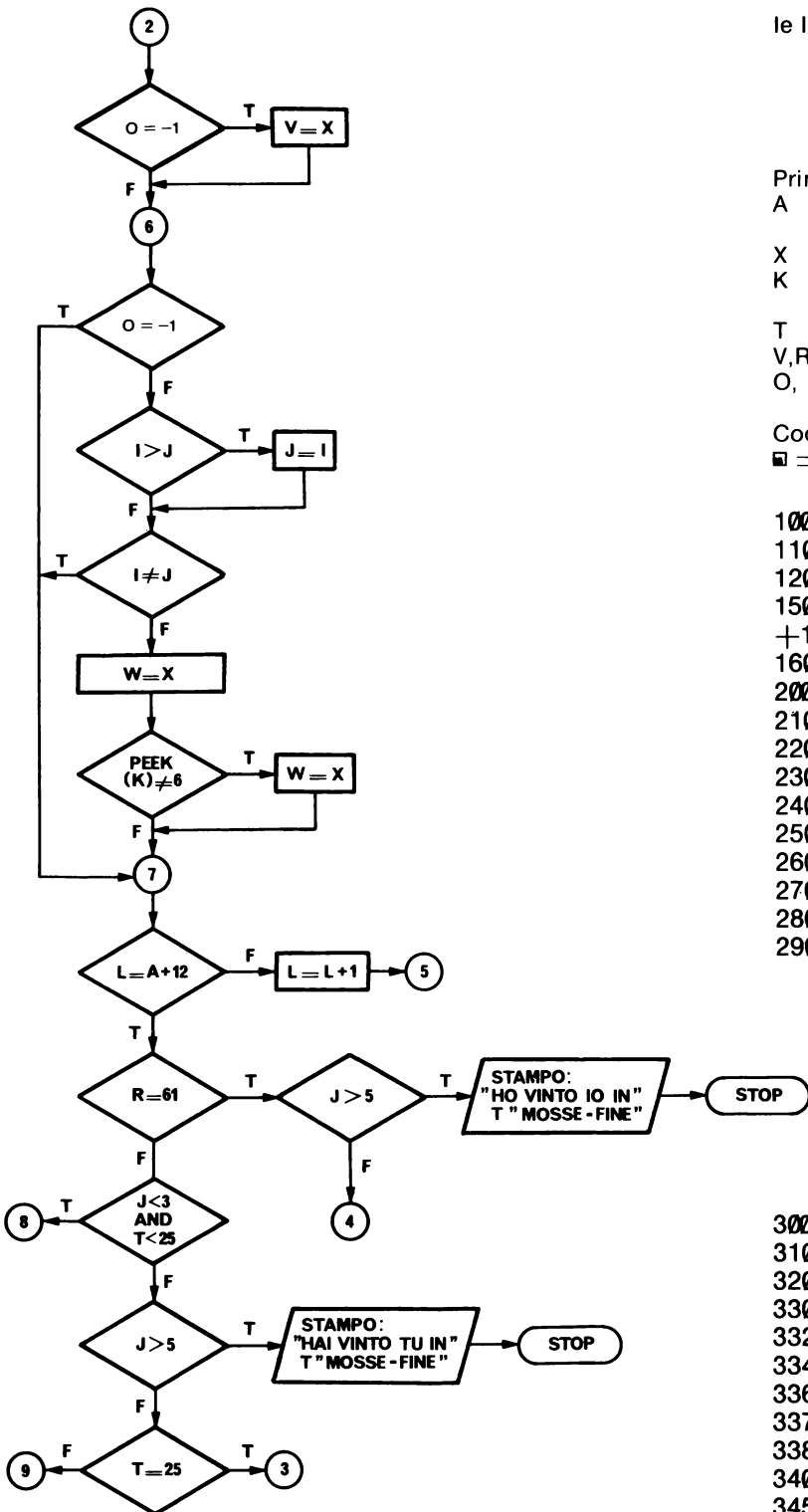
Si noterà la presenza, alle linee 1000-1090, di una routine la quale consente di inserire nel programma sia il programmino assembler che le costanti di incremento già citate. Più precisamente essa carica nelle locazioni di memoria occupate dalla codifica della istruzione:

Quindi dopo aver caricato in memoria l'intero programma occorre mandare in esecuzione la routine con il comando immediato:

| | |
|-------|-----|
| 16428 | 42 |
| 16429 | 12 |
| 16430 | 64 |
| 16431 | 201 |
| 16432 | 9 |
| 16433 | 113 |
| 16434 | 23 |
| 16435 | 1 |
| 16436 | 7 |
| 16437 | 17 |
| 16438 | 119 |
| 16439 | 16 |
| 16440 | 112 |

le linee 200-340 visualizzano la scacchiera, una intestazione e la richiesta delle coordinate:

GOMOKU



le linee ~~400-490~~

consentono l'introduzione delle coordinate del giocatore e ne controllano la correttezza;

le linee ~~500~~-740

esaminano la mossa del giocatore e del calcolatore indicando l'eventuale vittoria o suggerendo al calcolatore la mossa di risposta;

le linee 800-840

generano l'eventuale mossa casuale del calcolatore;

le linee 850-970

eseguono la visualizzazione del risultato finale della partita;

le linee 1000-1090

contengono la routine mandata in esecuzione appositamente per caricare il programma in linguaggio assembler e le costanti di incremento.

Principali variabili usate

| | |
|------------|--|
| A | indirizzo dal quale introdurre il programma assembler e le costanti; |
| X | coordinate mossa; |
| K | indirizzo di memoria relativo ad una coordinata; |
| T | contatore mosse; |
| V,R,S,U,W, | variabili analisi mossa; |
| O, I, J | variabili analisi mossa. |

Codici caratteri grafici utilizzati:

■ = 6, ■ = 9, x = 61

100 REM

110 LET A = 16428

120 GO TO 200

```
150 LET K=USR (A)+2 * (X AND 15) - 24 * (X/16)
+175
```

160 RETURN

```
200 FOR K = 1 TO 7
```

210 LET J = 8 - K

220 PRINT J,

```
230 FOR I = 1 TO 7
```

240 PRINT "AΔ":

250 NEXT I

260 PRINT

270 PRINT

280 NEXT K

290 PRINT

```
300 FOR K = 1 TO 7
```

```
310 PRINT K: "Δ";
```

320 NEXT K

330 PRINT

332 PRINT

334 PRINT "*****"

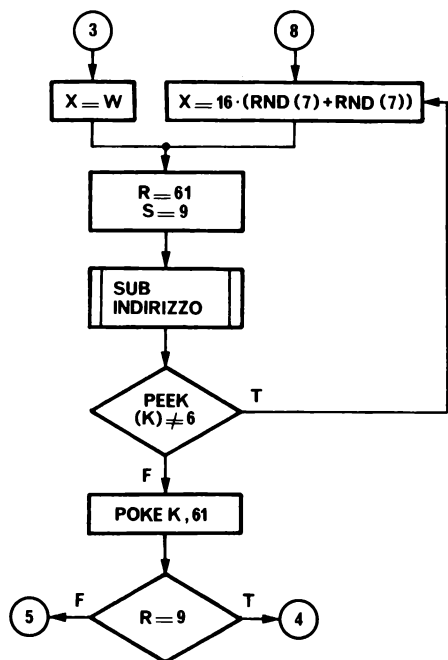
```
336 PRINT "*** GOMOKU ***"
```

337 PRINT "*****"

338 PRINT

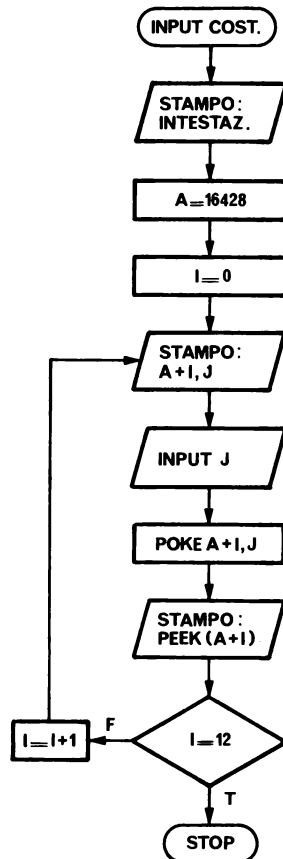
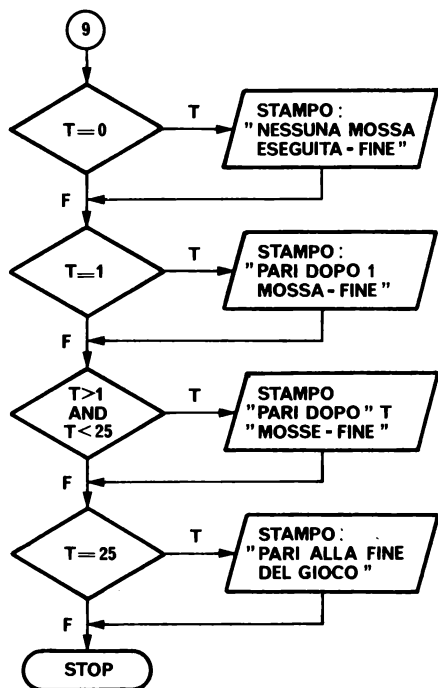
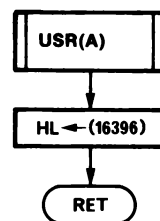
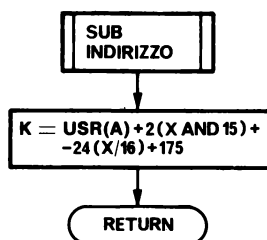
340 PRINT "SCRIVILE TUE COORDINATE:"

345 LEFT T - 0



```

400 INPUT X
410 IF X = 0 THEN GO TO 900
415 IF X < 0 THEN GO TO 400
420 LET X = X + 6 * (X/10)
430 LET X = X AND 119
440 IF X/16 = 0 OR (X AND 7) = 0 THEN GO TO 400
450 GO SUB 150
460 IF PEEK (K) > 6 THEN GO TO 400
470 POKE K, 9
480 LET V = 77
484 LET R = 9
488 LET S = 61
490 LET T = T + 1
500 LET J = 0
510 LET U = X
520 FOR L = A + 4 TO A + 12
530 LET I = 0
  
```



GOMOKU

```

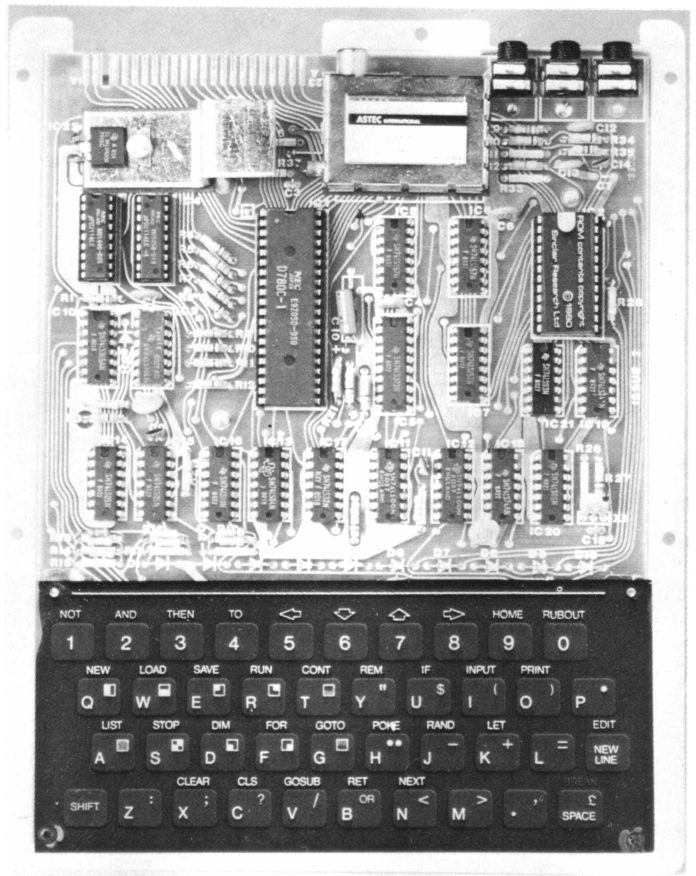
540 LET O = 0
550 IF 2 * (L/2) < L THEN LET O = -1
560 IF O THEN LET X = U
570 LET X = X + PEEK (L)
580 LET X = X AND 119
590 LET I = I + 1
600 IF X/16 = 0 OR (X AND 15) = 0 THEN GO TO 650
610 GO SUB 150
620 IF PEEK (K) = R THEN GO TO 570
630 IF PEEK (K) = S AND O THEN GO TO 700
640 IF O THEN LET V = X
650 IF O THEN GO TO 700
660 IF I > J THEN LET J = I
670 IF NOT I = J THEN GO TO 700
680 LET W = X
690 IF NOT PEEK (K) = 6 THEN LET W = V
700 NEXT L
702 IF R = 61 THEN GO TO 836
710 IF J < 3 AND T < 25 THEN GO TO 800
720 IF J > 5 THEN GO TO 850
725 IF T = 25 THEN GO TO 900
730 LET X = W
740 GO TO 802
800 LET X = 16 + RND (7) + RND (7)
802 LET R = 61
804 LET S = 9
810 GO SUB 150
820 IF NOT PEEK (K) = 6 THEN GO TO 800
830 POKE K, 61
832 IF R = 9 THEN GO TO 400
834 GO TO 500
836 IF J > 5 THEN GO TO 950
840 GO TO 400
850 PRINT
860 PRINT "HAI VINTO TU IN"; T; "MOSSE - FINE"
870 STOP
900 PRINT
905 IF T = 0 THEN PRINT "NESSUNA MOSSA ESEGUITA - FINE"
910 IF T = 1 THEN PRINT "PARI DOPO 1 MOSSA - FINE"
920 IF T > 1 AND T < 25 THEN PRINT "PARI DOPO "; T; " MOSSE - FINE"
930 IF T = 25 THEN PRINT "PARI ALLA FINE DEL GIOCO"
940 STOP
950 PRINT
960 PRINT "HO VINTO IO IN "; T; "MOSSE-FINE"
970 STOP
1000 PRINT "GOMOKU - INIZIO INPUT COSTANTI"

```

```

1001 PRINT
1002 PRINT
1003 PRINT "LOCAZIONE", "COSTANTE"
1004 PRINT
1008 LET A = 16428
1010 FOR I = 0 TO 12
1020 PRINT A + I,,
1030 INPUT J
1040 POKE A + I, J
1050 PRINT PEEK (A + I)
1060 NEXT I
1070 PRINT
1075 PRINT
1080 PRINT "GOMOKU - FINE INPUT COSTANTI"
1090 STOP

```



LIFE

Autore: A. Napoletano
Programma utilizzante:
4 K di memoria

Questa è una versione molto piccola e necessariamente limitata del famoso programma LIFE, ma nonostante le sue limitazioni esso rimane ancora molto interessante. Il programma simula una coltura di cellule, la sopravvivenza di ogni cellula dipende da quante cellule essa ha vicino.

Troppe cellule vicine implicano una diminuzione del cibo e quindi la cellula muore; troppo pochi vicini implicano la mancanza di supporto quindi la cellula non si riproduce.

La regola che governa la nascita di una nuova cellula è che essa viene generata quando in un suo intorno essa ha esattamente tre cellule vicine, la cellula muore quando ha meno di 2 o più di tre cellule vicine.

La popolazione viene inizialmente generata a caso.

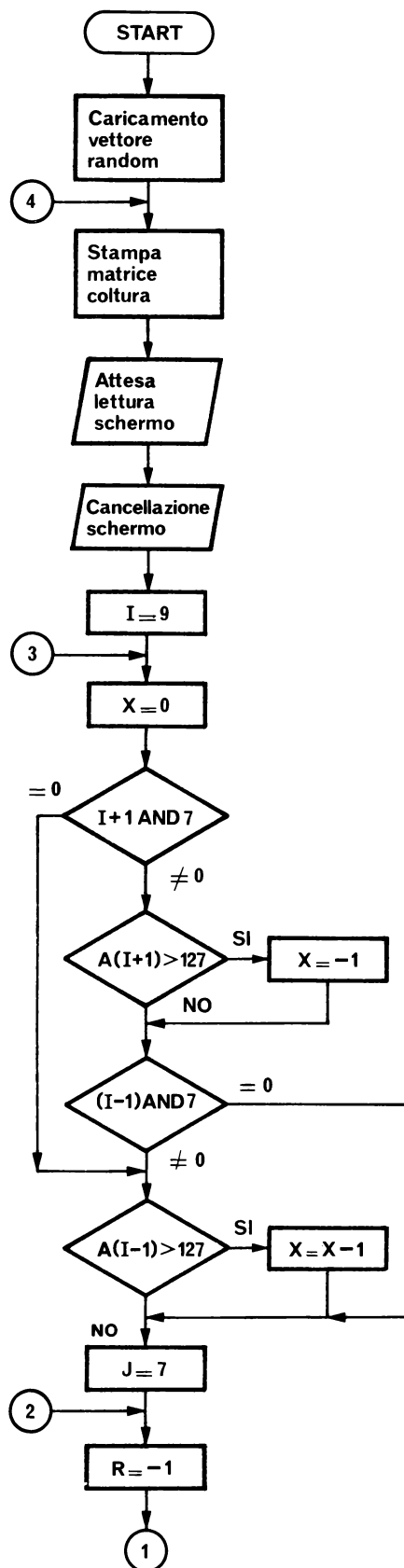
Il programma è abbastanza lento, perciò siate pazienti.

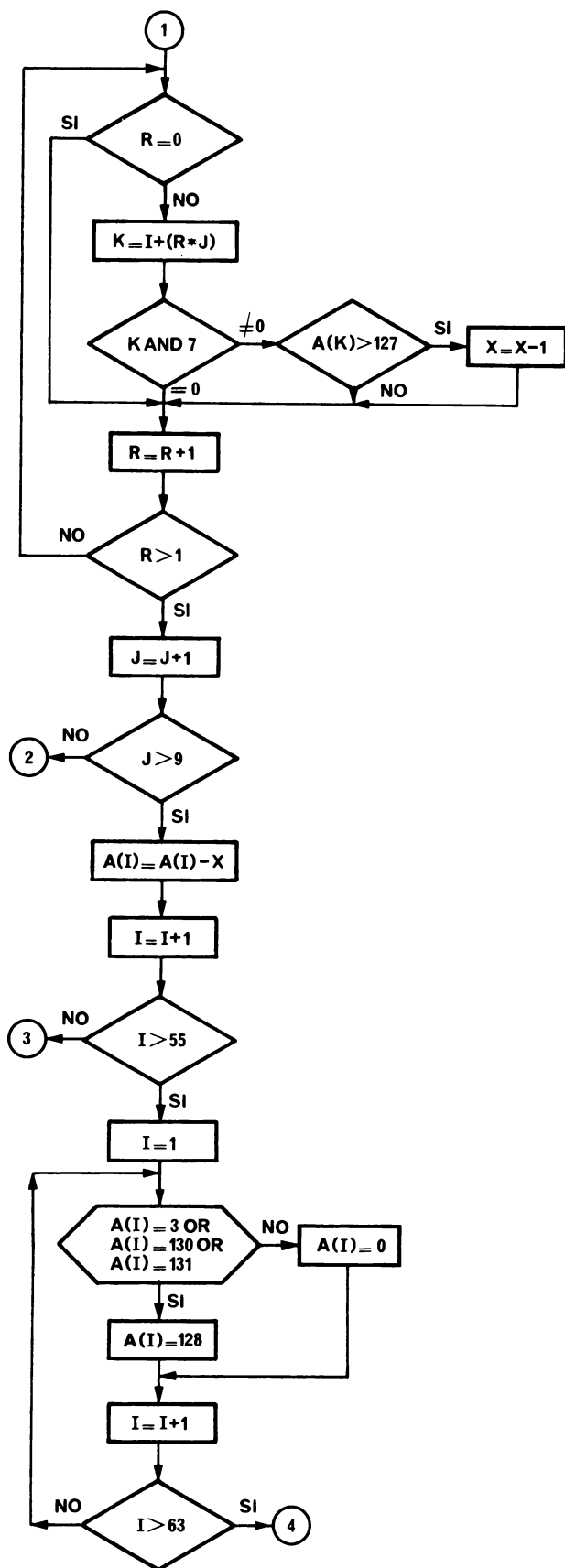
■ Cellula □ Spazio

```

100 DIM A (63)
105 LET B=0
106 LET E=0
107 LET T=0
110 FOR I=1 TO 63
120 LET A(I) = 128 * (RND(2)-1)
125 IF A(I)=128 THEN LET B=B+1
130 NEXT I
131 PRINT "      **** LIFE ****"
132 PRINT
133 PRINT
134 PRINT "CELLULE NELLA COLTURA : " ; B
135 GOTO 142
136 PRINT "      **** LIFE ****"
137 PRINT
138 PRINT
139 PRINT "NUOVE NATE : " ; E
140 PRINT
141 PRINT "CELLULE MORTE : " ; T
142 FOR I=1 TO 3
143 PRINT
144 NEXT I
145 LET G=0
146 LET E=0
147 LET T=0
200 FOR L=0 TO 8

```





```

205 PRINT "ΔΔΔΔΔΔΔΔΔΔ";
210 FOR J=1 TO 7
220 LET I=J+7*L
230 PRINT CHR$(A(I));
240 NEXT J
250 PRINT
260 NEXT L
265 PRINT
266 PRINT
270 PRINT "PREMI <N/L> PER CONTINUARE"
280 INPUT A $
285 IF A $ = "Δ" THEN GOTO 290
287 CLS
288 STOP
290 CLS
300 FOR I = 9 TO 55
310 LET X = 0
320 IF ((I+1)AND 7) = 0 THEN GOTO 350
330 LET X = (A (I+1)> 127)
340 IF ((I-1)AND 7) = 0 THEN GOTO 360
350 LET X = X + (A(I-1)> 127)
360 FOR J = 7 TO 9
370 FOR R = -1 TO 1
380 IF R = 0 THEN GOTO 420
390 LET K = I + R*J
400 IF (K AND 7) = 0 THEN GOTO 420
410 LET X=X + (A(K)> 127)
420 NEXT R
430 NEXT J
440 LET A (I) = A (I) -X
460 NEXT I
465 FOR I=1 TO 63
470 IF A (I)=3 OR A (I)=130 OR A (I)=131 THEN
GOTO 500
480 LET A (I)=0
485 LET T= T+1
490 GOTO 510
500 LET A (I)=128
505 LET E=E+1
510 NEXT I
512 FOR I=1 TO 63
513 IF A (I)=0 THEN LET G=G+1
514 NEXT I
515 IF G>58 THEN GOTO 530
520 GOTO 136
530 CLS
535 PRINT "      **** LIFE ****"
536 PRINT
537 PRINT
538 PRINT
539 PRINT "LE CELLULE SI SONO
AUTODISTRUTTE"
540 PRINT
541 PRINT "ALTRA SIMULAZIONE <S/N>"
545 INPUT B $
546 IF B $ = "S" THEN CLS
547 IF B $ = "S" THEN GOTO 100
550 GOTO 287

```

BATTAGLIA NAVALE

Autore: E. Vighi
Programma utilizzante:
4 K di memoria

Si tratta del notissimo gioco in cui si cerca di affondare la flotta nemica; che inizialmente sembra solo un passatempo ma che può diventare appassionante quando i giocatori si trovano in una situazione critica, sia perché sono vicini alla vittoria, sia perché la loro flotta è stata decimata e sono quindi nella condizione di dover salvare le ultime navi rimaste.

Nella versione presentata, il giocatore e l'elaboratore dispongono entrambi di una scacchiera con 64 caselle, suddivisa in 8 righe ed 8 colonne. Sulla scacchiera vengono disposte 8 navi, nelle posizioni che sembrano offrire la maggiore sicurezza, cioè dove si pensa che l'avversario non sparerà.

Si spara un colpo alla volta alternandosi: vince chi riesce a distruggere per primo la flotta avversaria, altrimenti, se non si vuole continuare fino a quel punto, il vincitore è chi ha colpito più navi avversarie; è chiaro che se non si arriva alla fine del gioco è possibile che si verifichi anche una situazione di parità, in cui entrambi i giocatori hanno colpito lo stesso numero di navi.

Per sparare un colpo si devono indicare le coordinate (numero di riga e numero di colonna) della posizione che si intende colpire; l'avversario verificherà quale delle seguenti situazioni è accaduta:

- colpo a vuoto: in quella posizione non c'era una nave
- nave colpita e affondata
- colpo già sparato in quella posizione

In quest'ultimo caso, poiché non è consentito sparare su una casella già colpita in precedenza, si deve scegliere un'altra posizione non ancora colpita fino a quel punto su cui sparare, cioè si ripete il colpo.

PROGRAMMA

Le due scacchiere, una per l'elaboratore ed una per il giocatore, sono trattate sotto forma di vettori a 64 posizioni o elementi, che inizialmente vengono riempiti con il codice del carattere grafico "■". Questo carattere indica una posizione in cui non è presente una nave e nella quale non si è ancora sparato.

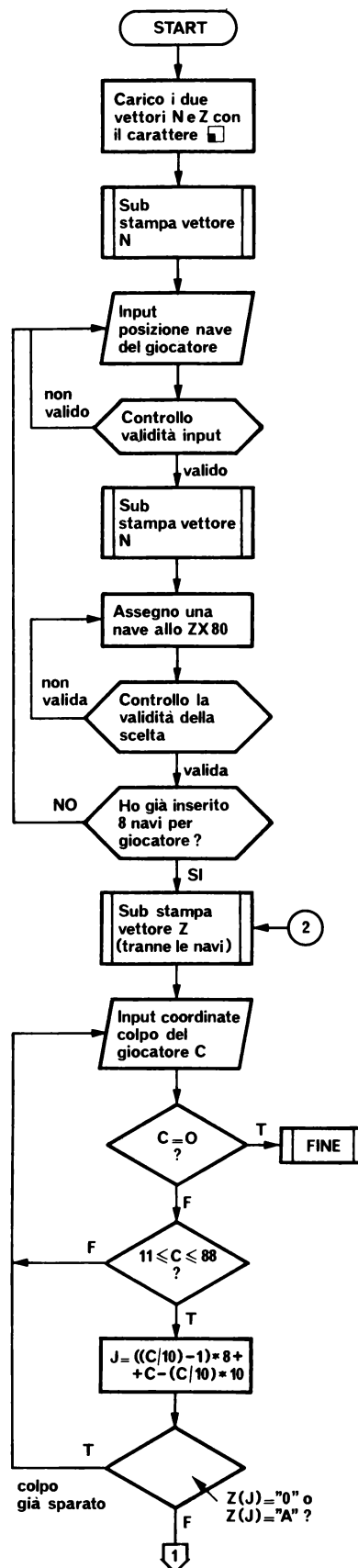
Una volta inizializzati i due vettori, si inseriscono in ciascuno di essi 8 navi; le navi del giocatore sono poste nelle posizioni da lui scelte, mentre quelle del calcolatore vengono inserite mediante la funzione di casualità RND.

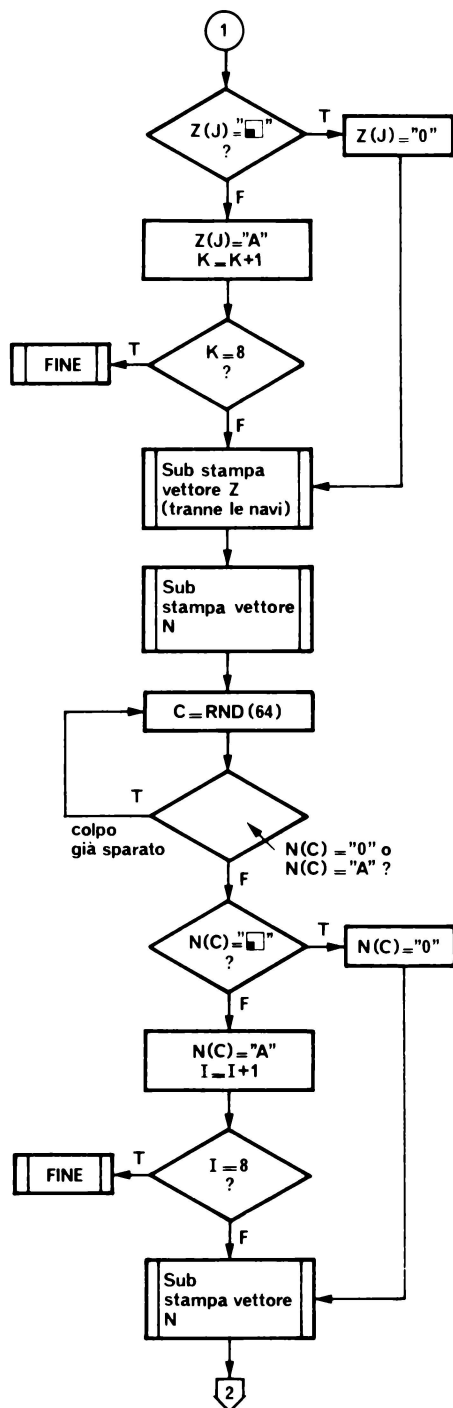
In caso di errore nella scelta della posizione delle navi, sia perché si tenta di mettere due navi nella stessa casella, sia perché le coordinate sono errate, il programma ripete l'operazione.

Il numero di navi è fissato ad 8; se lo si volesse modificare basterebbe cambiare la linea 90, inserendo come limite superiore del ciclo il numero di navi voluto.

A questo punto il giocatore spara un colpo, inserendo le coordinate della posizione che vuole colpire.

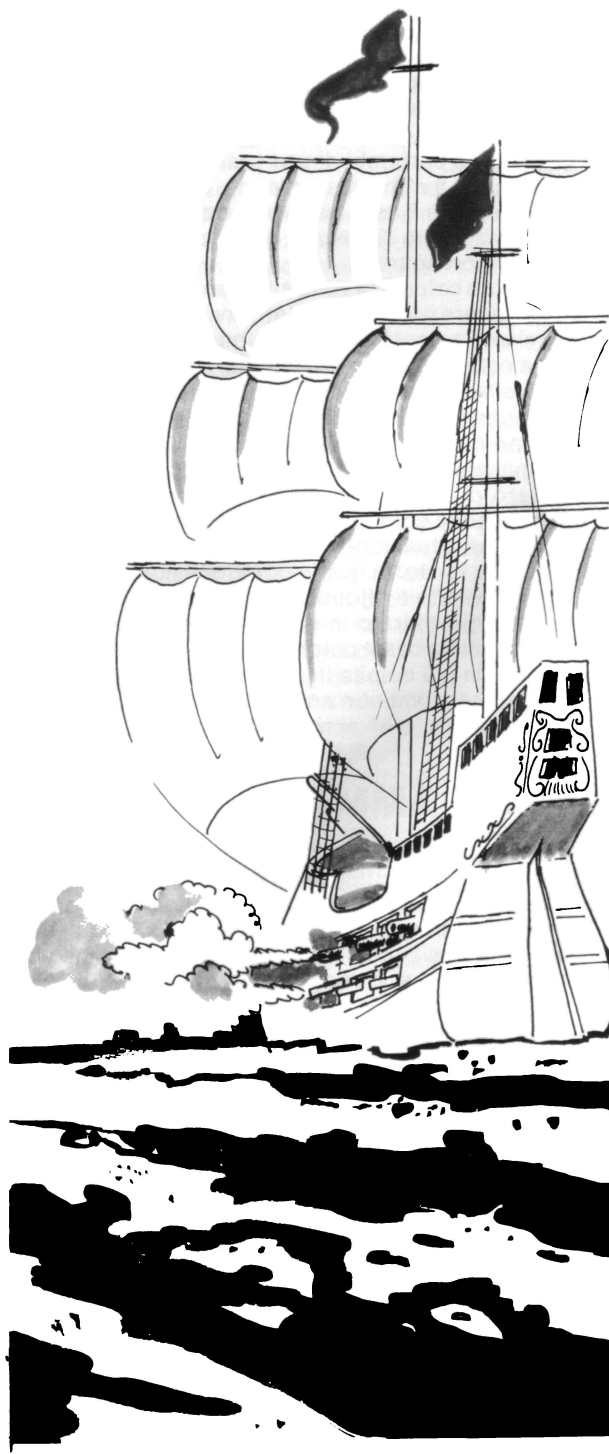
Se inserisce il valore zero, il programma esegue la fase finale in cui si stabilisce il vincitore. Se le coordinate sono errate o la posizione indicata è già stata colpita, si ripete l'operazione.





Se non c'è una nave nella posizione indicata, si inserisce il carattere 0 per indicare che in quella casella si è già sparato; mentre se c'è una nave si inserisce il codice del carattere "A", per indicare la presenza di una nave affondata, e si incrementa il contatore delle navi colpite dal giocatore. Se si sono colpite tutte le navi dell'avversario, entra in esecuzione la fase finale del programma. Dopo ogni colpo del giocatore, le stesse operazioni si ripetono per il calcolatore, che spara i suoi colpi sfruttando la funzione RND.

Poiché il giocatore vede la scacchiera come una matrice 8x8, la posizione delle navi che inserisce e la posizione dei colpi che spara sono una coppia di numeri compresi fra 1 ed 8. Detta C questa coppia di numeri deve essere $11 \leq C \leq 88$.



Per il calcolatore, però, la scacchiera è un vettore di 64 elementi perciò è necessaria una conversione di C in un indice per il vettore citato.

Ciò è realizzato mediante la formula:

$$J = ((C/10)-1) * 8 + C - (C/10) * 10$$

Come si vede, la somma di $((C/10)-1) * 8$, che è il numero di posizioni presenti nelle righe che precedono quella valuta, e di $C - (C/10) * 10$, che è la posizione della casella voluta all'interno della riga considerata ci dà il numero d'ordine della casella in questione all'interno del vettore suddetto.

Le linee 40 - 70

realizzano il caricamento iniziale dei vettori con il codice del carattere "■", che è 6 servono all'inserimento delle navi

le linee 90 - 200

gestiscono il colpo sparato dal giocatore

le linee 380 - 470

gestiscono il colpo sparato dal calcolatore

La subroutine che inizia alla linea 1000 visualizza la scacchiera del giocatore, quella che inizia alla linea 2000 visualizza la scacchiera del calcolatore.

La subroutine che inizia alla linea 3000 gestisce la fine del gioco, e non è una vera e propria subroutine, poichè è richiamata con un GOTO e da esso non si torna ad un punto del programma, ma si termina l'esecuzione.

Le linee dalla 4000 in poi costituiscono una subroutine che visualizza l'intestazione del programma.

Repertorio

N (64) vettore contenente la scacchiera del giocatore
Z (64) vettore contenente la scacchiera del calcolatore

K contatore navi colpite dal giocatore

I contatore navi colpite dal calcolatore

F flag usato nella routine di visualizzazione del vettore Z; se $F=1$ si visualizza il vettore in forma di scacchiera così com'è, altrimenti se $F=0$ al posto delle navi nella scacchiera del calcolatore si visualizzano dei punti per non far conoscere al giocatore avversario la posizione delle navi ancora da colpire del calcolatore

B \$ variabile di comodo usata nel programma per consentire al gioco di ritardare opportunamente le visualizzazioni successive.

Tabella codici simboli

"0" corrisponde 28 colpo a vuoto, posizione già colpita

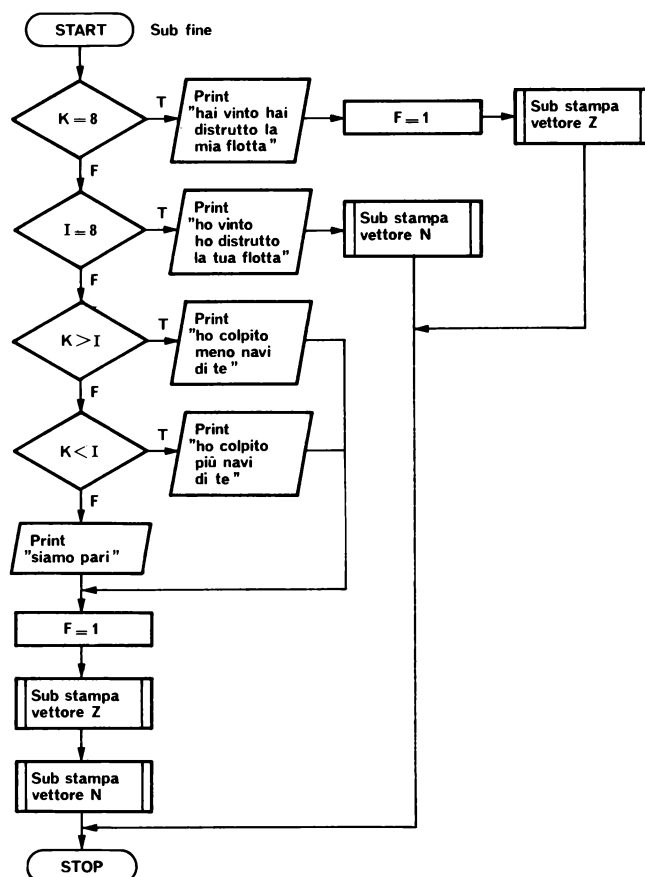
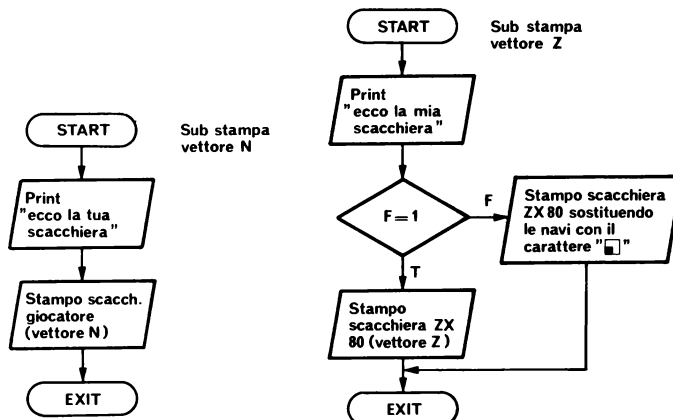
"■" corrisponde 6 posizione libera ancora da colpire

"■" corrisponde 9 posizione occupata da una nave

"A" corrisponde 38 nave affondata

È importante ricordare che per terminare l'esecuzione del programma è sufficiente inserire la coordinata zero quando si ha la corrispondente richiesta da parte del calcolatore, invece di inserire un numero compreso fra 11 e 88; se non si vuole interrompere il programma, esso terminerà automaticamente non appena una delle due flotte sarà stata affondata completamente.

```
10 GO SUB 4000
15 LET K=0
16 LET I=0
17 LET F=0
20 DIM N (64)
30 DIM Z (64)
40 FOR T=1 TO 64
50 LET N (T)=6
```



| | | | |
|------|---------------------------------------|------|--|
| 60 | LET Z (T)=6 | 1220 | RETURN |
| 70 | NEXT T | 2000 | PRINT "ECCO LA MIA SCACCHIERA : " |
| 80 | GO SUB 1000 | 2010 | PRINT |
| 90 | FOR T=1 TO 8 | 2080 | FOR S=1 TO 8 |
| 100 | PRINT "COORDINATE NAVE : " ; | 2090 | PRINT S, |
| 110 | INPUT C | 2100 | FOR R=(S-1)*8 TO S*8-1 |
| 120 | IF C<11 OR C>88 THEN GOTO 110 | 2110 | IF Z(R)=9 AND F=0 THEN PRINT "■ΔΔ"; |
| 130 | LET J=((C/10)-1)*8+C- (C/10)*10 | 2115 | IF NOT Z(R)=9 THEN PRINT CHR\$(Z(R)); |
| 140 | IF NOT N(J)=6 THEN GO TO 110 | | "Δ"; |
| 150 | LET N(J)= 9 | 2118 | IF F=1 AND Z(R)=9 THEN PRINT CHR\$(|
| 160 | GO SUB 4000 | | Z(R)); "Δ"; |
| 165 | GO SUB 1000 | 2120 | NEXT R |
| 170 | LET J=RND (64) | 2130 | PRINT |
| 180 | IF NOT Z(J)=6 THEN GO TO 170 | 2140 | PRINT |
| 190 | LET Z(J)=9 | 2150 | NEXT S |
| 200 | NEXT T | 2160 | PRINT, |
| 202 | GO SUB 4000 | 2170 | FOR S=1 TO 8 |
| 204 | GO SUB 1000 | 2180 | PRINT S; "Δ"; |
| 206 | PRINT "PREMI <N/L > PER SPARARE" | 2190 | NEXT S |
| 208 | INPUT B \$ | 2200 | PRINT |
| 210 | GO SUB 4000 | 2210 | PRINT |
| 215 | GO SUB 2000 | 2220 | RETURN |
| 220 | PRINT "COORDINATE COLPO : " ; | 3000 | GO SUB 4000 |
| 230 | INPUT C | 3005 | LET F=1 |
| 235 | IF C = 0 THEN GO TO 3000 | 3010 | IF NOT K=8 THEN GO TO 3080 |
| 240 | IF C <11 OR C >88 THEN GO TO 230 | 3020 | PRINT "**** HAI VINTO ****" |
| 260 | LET J= ((C/10)-1) *8 +C- (C/10) *10 | 3030 | PRINT |
| 270 | IF Z(J)= 28 OR Z(J)=38 THEN GO TO 230 | 3040 | PRINT "HAI DISTRUTTO LA MIA FLOTTA" |
| 280 | IF NOT Z(J)=6 THEN GO TO 310 | 3050 | PRINT |
| 290 | LET Z(J)=28 | 3052 | PRINT |
| 300 | GO TO 340 | 3054 | PRINT " <N/L > PER VEDERE LE MIE NAVI" |
| 310 | LET Z(J)=38 | 3056 | INPUT B\$ |
| 320 | LET K=K + 1 | 3058 | GO SUB 4000 |
| 330 | IF K=8 THEN GO TO 3000 | 3060 | GO SUB 2000 |
| 340 | GO SUB 4000 | 3070 | STOP |
| 350 | PRINT "PREMI <N/L > PER CONTINUARE" | 3080 | IF NOT I=8 THEN GO TO 3150 |
| 360 | INPUT B \$ | 3090 | PRINT "**** HO VINTO ****" |
| 370 | GO SUB 4000 | 3100 | PRINT |
| 375 | GO SUB 1000 | 3110 | PRINT "HO DISTRUTTO LA TUA FLOTTA" |
| 380 | PRINT " <N/L > PER FARMI SPARARE" | 3120 | PRINT |
| 390 | INPUT B \$ | 3122 | PRINT |
| 400 | LET C=RND (64) | 3124 | PRINT " <N/L > PER VEDERE LE TUE NAVI" |
| 410 | IF N(C)=28 OR N(C)=38 THEN GO TO 400 | 3126 | INPUT B\$ |
| 420 | IF NOT N(C)=6 THEN GO TO 450 | 3128 | GO SUB 4000 |
| 430 | LET N(C)=28 | 3130 | GO SUB 1000 |
| 440 | GO TO 480 | 3140 | STOP |
| 450 | LET N(C)=38 | 3150 | GO SUB 4000 |
| 460 | LET I=I +1 | 3160 | PRINT |
| 470 | IF I=8 THEN GO TO 3000 | 3170 | IF K>I THEN PRINT "HO COLPITO MENO |
| 480 | GO SUB 4000 | | NAVI DI TE" |
| 485 | GO SUB 1000 | 3180 | IF K<I THEN PRINT "HAI COLPITO MENO |
| 490 | PRINT "PREMI <N/L > PER SPARARE" | | NAVI DI ME" |
| 500 | INPUT B \$ | 3190 | IF K=I THEN PRINT "SIAMO PARI" |
| 510 | GO TO 210 | 3200 | PRINT |
| 1000 | PRINT "ECCO LA TUA SCACCHIERA : " | 3210 | PRINT " <N/L > PER VEDERE LE MIE NAVI" |
| 1010 | PRINT | 3225 | INPUT B\$ |
| 1080 | FOR S=1 TO 8 | 3228 | GO SUB 4000 |
| 1090 | PRINT S, | 3240 | GO SUB 2000 |
| 1100 | FOR R= (S-1) *8 TO S*8 -1 | 3250 | PRINT " <N/L > PER VEDERE LE TUE NAVI" |
| 1110 | PRINT CHR\$(N(R)); "Δ"; | 3270 | INPUT B\$ |
| 1120 | NEXT R | 3275 | GO SUB 4000 |
| 1130 | PRINT | 3280 | GO SUB 1000 |
| 1140 | PRINT | 3290 | PRINT "**** CIAO ****" |
| 1150 | NEXT S | 3300 | STOP |
| 1160 | PRINT, | 4000 | CLS |
| 1170 | FOR S=1 TO 8 | 4010 | PRINT "**** BATTAGLIA NAVALE ****" |
| 1180 | PRINT S; "Δ"; | 4020 | PRINT |
| 1190 | NEXT S | 4030 | RETURN |
| 1200 | PRINT | | |
| 1210 | PRINT | | |

BLACK JACK

Autore: E. Vighi
Programma utilizzante:
4 K di memoria

Regole di gioco

Si tratta di un gioco di carte, che si realizza con un mazzo di 52 carte da poker. A ciascuna carta è assegnato un valore, secondo lo schema seguente:

Le carte da 2 a 10 valgono il numero indicato; le figure, cioè il fante, la donna ed il re, valgono 10; l'asso vale normalmente 11, ma se considerandolo con questo valore si supera il totale di 21, allora gli si assegna il valore 1.

Un giocatore gioca contro il banco: vince chi, prendendo quante carte vuole, arriva più vicino al totale di 21 senza superarlo; se si supera il 21 si "sballa", cioè si perde.

Esistono due versioni del black jack:

- 1) il giocatore può chiedere quante carte vuole, il banco però deve raggiungere almeno un totale di 17 punti.
Per esempio, se il giocatore ha solo 13 punti ed il banco ne ha 14, il banco deve prendere un'altra carta, rischiando di sballare.
Se invece il giocatore ha 19 punti ed il banco per esempio ne ha 18, allora il banco non prende più carte ed il giocatore vince.
- 2) il giocatore può chiedere quante carte vuole, il banco continua a chiedere carte finché non ha superato il totale del giocatore oppure ha sballato.
Per esempio, se il giocatore ha solo 13 punti ed il banco ne ha 14, il banco ha vinto e non chiede altre carte. Se invece il giocatore ha 19 punti ed il banco per esempio ne ha 18, allora il banco prende un'altra carta per tentare di realizzare un totale compreso fra 19 e 21 e quindi pareggiare o vincere, anche se prendendo un'altra carta rischia di "sballare".

Il programma non segue la prima versione presentata, ma se si volesse giocare secondo le regole esposte al punto 1) basterebbe modificare le linee 621 e 622 in:

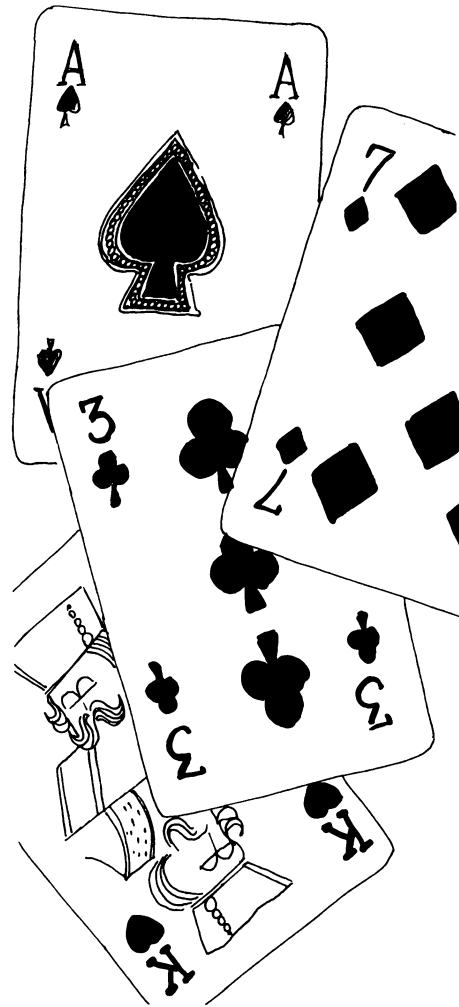
```
621 IF I = 0 THEN GO TO 530
622 GO TO 670
```

Inizialmente il giocatore dispone di un capitale di 100 L.

Ad ogni modo scommette una somma, che non può essere maggiore del capitale posseduto, né può essere negativa. Se la somma scommessa è uguale a zero, il programma termina.

Per quanto riguarda le vincite si ha:

- 1) Si vince il doppio della posta, se si realizza 21 con 2 carte ed il banco non realizza anch'esso 21.
- 2) Si vince la posta scommessa se si realizzano più punti del banco o se il banco sballa.
- 3) Non si vince e non si perde nulla se si realizzano gli stessi punti del banco.
- 4) Si perde la posta scommessa se il banco realizza più punti del giocatore senza sballare, oppure se si sballa.



PROGRAMMA

Il programma usa un metodo particolare per memorizzare e mischiare il mazzo di carte: le 52 carte sono immagazzinate in un'istruzione REM all'inizio del programma, con le seguenti convenzioni:

i numeri da 2 a 9 per le corrispondenti carte

- D per il 10 (in tal modo è sufficiente un solo carattere anziché due)
J per il fante,
Q per la donna,
K per il re,
A per l'asso.

Le linee 310-360 del programma sono usate per "mescolare" il mazzo, scambiando una carta con un'altra scelta a caso nel mazzo. Dopo ogni partita, alla linea 100 del programma potremo vedere l'ultima versione del mazzo mescolando, perché si usano le istruzioni PEEK e POKE che modificano fisicamente il contenuto dello statement 100. Si noti la brevità e la completezza del metodo usato.

Ad ogni pescata, il totale del giocatore e quello del calcolatore vengono aggiornati e visualizzati.

Si noti che alla richiesta:

CARTA: SI/NO

si può rispondere, nel caso affermativo, sia con una S che un altro carattere qualsiasi, anche solo premendo il tasto NEW LINE; se invece non si desiderano altre carte è necessario introdurre il carattere N.

Il programma non può girare in 1 K di memoria: se si volesse ridurre le dimensioni per adottarlo in 1 K basterebbe eliminare quasi totalmente le stampe parziali.

I commenti sono molto concisi, ma volendo si potrebbe rendere più gradevole il programma con delle stampe esplicative.

PRINCIPALI VARIABILI USATE

N(0) numero di carte chieste dal giocatore (TUE)
 N(1) numero di carte prese dal banco (ZX80)
 N(2) valore attuale della mano del giocatore
 N(3) valore attuale della mano del banco
 A(0) numero degli assi di cui si considera come valore l'11 per il giocatore
 A(1) numero degli assi di cui si considera come valore l'11 per il banco
 M somma a disposizione del giocatore nella mano corrente
 B somma scommessa nella mano corrente
 V coefficiente di vincita/perdita.

PROGRAMMA

Linee 100-140 inizializzazione variabili
 150-180 calcolo del nuovo valore della mano
 200-300 routine di visualizzazione risultati
 310-360 mischiamo il mazzo di carte
 370-450 fase iniziale nuova partita
 500-700 corpo del programma: gestione della mano
 760-820 definizione della vincita/perdita e visualizzazione risultati.

```

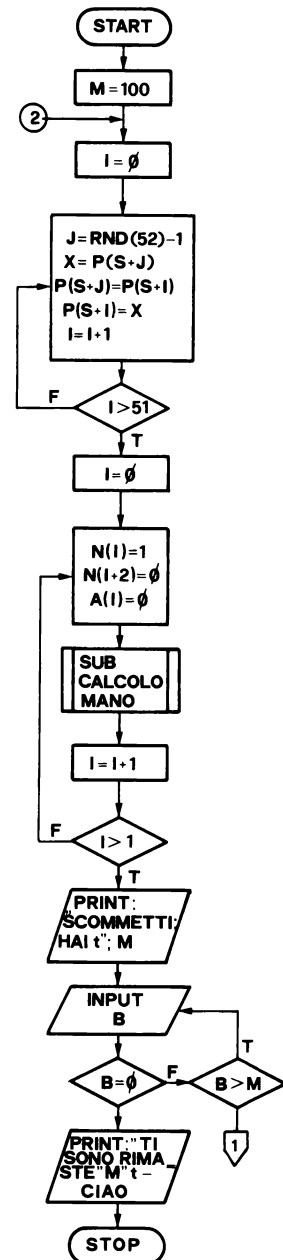
100 REM 23456789DJQKA23456789DJQKA2345
    6789DJQKA23456789DJQKA
110 DIM N(3)
115 DIM A(1)
120 LET S = 16427
130 LET M = 100
140 GO TO 310
150 LET X = PEEK (S + 6 * I + N(I)) - 28
160 LET N(I+2) = N(I+2) - 10 * (x > 10) - x * (x < 10) - 11 * (x = 10)
170 LET A(I) = A(I) - (x = 10)
180 RETURN
200 PRINT "HAI SCOMMESSO L"; B
201 PRINT
202 PRINT
203 PRINT "TUE", "ZX80"
210 FOR K = 0 TO 1
220 FOR L = 6 * K + 1 TO 6 * K + N(K)

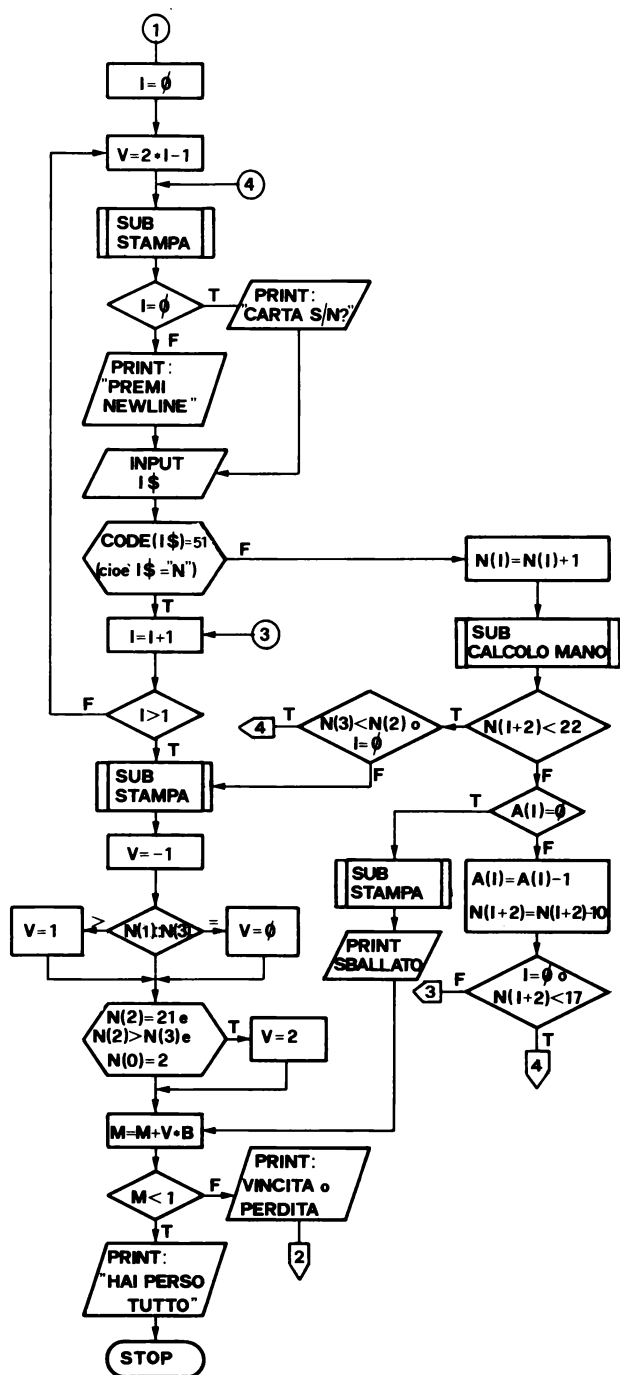
```

```

230 LET x = PEEK (S + L)
240 PRINT CHR$ (x); "Δ";
250 NEXT L
255 IF N(0) > 3 THEN PRINT,
256 IF NOT N(0) > 3 THEN PRINT,,
270 NEXT K
275 PRINT
276 PRINT
280 PRINT "TOTALE:"Δ"
290 PRINT N(2), N(3)
295 PRINT
300 RETURN
310 FOR I = 0 TO 51

```

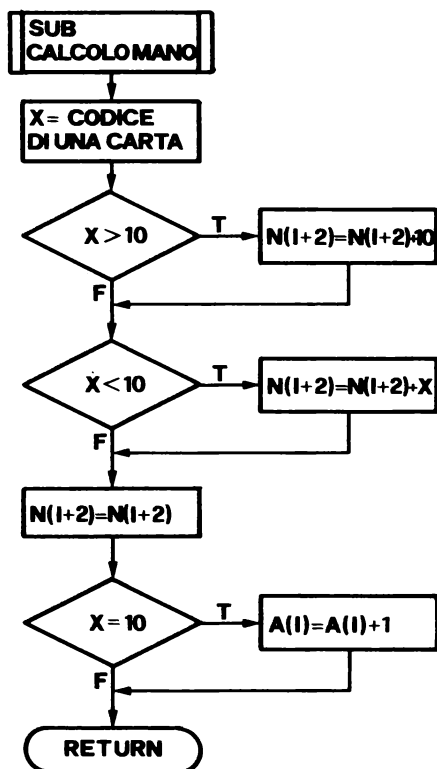
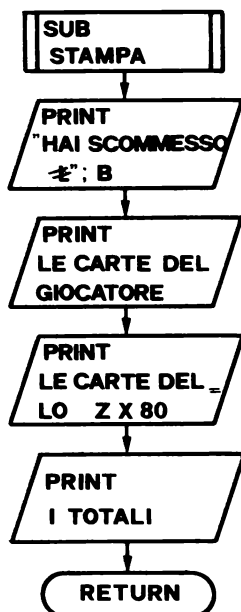




```

320 LET J = RND (52) - 1
330 LET X = PEEK (S + J)
340 POKE S+J, PEEK (S+I)
350 POKE S+I, X
360 NEXT I
370 FOR I=0 TO 1
380 LET N(I) = 1
390 LET N(I+2) = 0
400 LET A(I) = 0
410 GO SUB 150
420 NEXT I
426 PRINT
427 PRINT
430 PRINT "SCOMMETTI: HAI L."; M
440 INPUT B
441 IF NOT B=0 THEN GO TO 445
442 CLS
443 PRINT "TI SONO RIMASTE"; M;
    "L. - CIAO"
444 STOP
445 IF B>M OR B<0 THEN GO TO 440
450 CLS
500 FOR I=0 TO 1
510 LET V = 2 * I - 1
530 GO SUB 200
540 PRINT
542 IF I=0 THEN PRINT "CARTA: S/N ?"
550 IF I = 1 THEN PRINT "PREMI < NEW LINE >"
570 INPUT I$
580 CLS
590 IF CODE (I$) = 51 THEN GO TO 700
600 LET N(I) = N(I) + 1
610 GO SUB 150
620 IF NOT N(I + 2) < 22 THEN GO TO 623
621 IF N(3) < N(2) OR I=0 THEN GO TO 530
622 GO TO 750
623 IF NOT A(I) = 0 THEN GO TO 640
624 GO SUB 200
625 PRINT "SBALLATO"
630 GO TO 800
640 LET A(I) = A(I) - 1
650 LET N(I + 2) = N(I + 2) - 10
670 IF I = 0 OR N(I + 2) < 17 THEN GO TO 530
700 NEXT I
750 GO SUB 200
760 LET V=-1
765 IF N(2) = N(3) THEN LET V=0
770 IF N(2) > N(3) THEN LET V=1
780 IF (N(2) = 21 AND N(0) = 2 AND N(2) > N(3))
    THEN LET V=2
800 LET M = M + V * B
810 IF M < 1 THEN PRINT "HAI PERSO TUTTO"
815 IF M < 1 THEN STOP
816 IF V = 1 THEN PRINT "HAI VINTO L."; B
817 IF V = 0 THEN PRINT "PARI"
818 IF V = -1 THEN PRINT "HAI PERSO L."; B
819 IF V = 2 THEN PRINT "HAI VINTO L."; 2 * B
820 GO TO 810.

```



Si prevedono delle stampe di spaziatura e di interlinea adatte.
 Se $X = 10$, la carta è un Asso
 se $X > 10$, la carta è una fra le seguenti: J, Q, K
 se $X < 10$, la carta è una delle restanti.

ROUTINE ASSEMBLER

Autore: M. Oliva
 Programma utilizzante:
 4 K di memoria

Si tratta di un programma didattico che consente di scrivere, di provare e di verificare delle semplici routine in linguaggio assembler.

Le funzioni che esso è in grado di soddisfare sono essenzialmente tre:

- 1) Scrittura di locazioni di memoria consecutive, byte per byte, a partire da un indirizzo specifico;
- 2) Lettura di locazioni di memoria consecutive, byte per byte, a partire da un indirizzo indicato, con la possibilità di correggere il contenuto;
- 3) Esecuzione della routine caricata in memoria a partire da un indirizzo specificato, con la visualizzazione del risultato finale.

Ognuna di queste fasi è svolta consecutivamente se all'appropriata richiesta si risponde "S" o si preme il tasto <NEW LINE>, mentre viene saltato se si risponde "N"

PROGRAMMA

La struttura del programma segue fedelmente la logica del funzionamento, già illustrato.

Inizialmente, dopo la chiamata alla subroutine che visualizza l'intestazione il programma provvede a porre la variabile A uguale a un valore pari all'indirizzo limite sotto il quale il programma non deve consentire la scrittura o la correzione delle locazioni di memoria.

Nella prima fase il programma richiede se si intende scrivere in memoria. Se la risposta è "N" si salta alla fase successiva altrimenti si procede. In questo caso il programma richiede l'indirizzo di partenza dal quale incominciare il caricamento, verificando che esso non sia minore di A. Dopo il programma provvede a richiedere il contenuto delle locazioni di memoria. Il valore che si deve introdurre deve variare da 0 (zero) a 255. Qualora esso fosse uguale a -1 la sequenza di caricamento si interrompe ed il controllo passa alla fase successiva. Al contrario il valore inserito è posto nella locazione prescelta mediante l'istruzione POKE, dopo di che, avendo incrementato di 1 l'indirizzo si ricicla.

Nella seconda fase il programma chiede se si desidera procedere alla visualizzazione della memoria. Se la risposta è "N" si salta alla fase successiva altrimenti il programma richiede l'indirizzo dal quale iniziare la visualizzazione. In questo caso esso può essere minore di A anche se ciò non consentirà di modificare il contenuto. Comunque l'indirizzo prescelto minimo è 16384, essendo questo l'indirizzo più basso della memoria.

Alla visualizzazione dell'indirizzo e del relativo contenuto il programma richiede di rispondere <NEW LINE> se si vuole vedere il contenuto del byte successivo <C> per correggere se è consentito e <U> per uscire.

La correzione comporta l'inserimento del nuovo valore e dei relativi controlli.

L'uscita consente invece di interrompere la serie di visualizzazione e correzioni e trasferisce il controllo alla

prossima fase. Tutte le letture in memoria sono effettuate per mezzo dell'istruzione PEEK.

Nella terza fase il programma chiede se si desidera procedere all'esecuzione della routine assembler caricata. Se la risposta è "N" il programma chiede se si vuole tornare all'inizio del programma. Se la risposta è ancora "N" il programma ha termine, diversamente si torna nuovamente alla prima fase.

Qualora invece la prima risposta indica che si deve eseguire la routine. Il programma richiede quale deve essere l'indirizzo di partenza. Se esso non è minore di A la routine è svolta per mezzo della linea

640 LET X = USR (P)

dove P è l'indirizzo dal quale inizia la routine ed X conterrà, alla fine dell'esecuzione, il valore assunto dalla coppia di registri HL.

Tale valore sarà poi visualizzato dal programma prima di chiedere se si intende concludere così il programma o se si vuole ritornare alla prima fase.

Come abbiamo già detto, il programma non consente la scrittura e la correzione di locazioni di memoria il cui indirizzo risulti minore del valore limite A. Ciò perché altrimenti si potrebbe modificare il contenuto di bytes di memoria occupati dal programma stesso. È evidente che quindi se la routine assembler fosse sbagliata, la sua esecuzione potrebbe causare la perdita del programma dalla memoria. Si consiglia quindi, oltre alla naturale attenzione circa il programma assembler, di salvare il programma su una cassetta prima della sua esecuzione.

le linee 10-150 eseguono il caricamento in memoria di byte singoli

le linee 200-470 eseguono la lettura della memoria e l'eventuale correzione

le linee 500-860 consentono l'esecuzione della routine assembler caricata

le linee 1000-1040 contengono la subroutine che visualizza l'intestazione

PRINCIPALI VARIABILI UTILIZZATE

- A contiene l'indirizzo limite sotto il quale il programma non consente la scrittura o la correzione in memoria
- B\$ stringa per le risposte alle diverse domande.
- I locazione dal quale iniziare una delle procedure (in input)
- L contenuto locazione (in input)
- P indirizzo dal quale si vuole eseguire la routine assembler
- X valore finale della coppia di registri HL dopo l'esecuzione della routine assembler.

Diamo ora come esempio, quattro semplici routine assembler da provare a verificare.

1) Routine trasferimento byte

Questa routine esegue il trasferimento delle locazioni di memoria a partire da quella di indirizzo 18500 nelle locazioni a partire da quella di indirizzo 18600. Il numero massimo di byte trasferiti è di 16, il trasferimento ha comunque fine se uno dei byte è zero.

Si noti che nella codifica della routine, così come delle altre, gli indirizzi di locazioni di memoria, occupando ciascuno 16 bit, devono essere scomposti in due byte: ad esempio l'indirizzo

$$18000_{10} = 0100100001000100 = 4844_{16}$$

$$\text{cioé nei due byte } 48_{16} = 72_{10} \text{ e } 44_{16} = 68_{10}$$

La routine è la seguente (il primo numero indica l'indirizzo di memoria del primo dei byte componenti la codifica dell'istruzione, gli altri il contenuto dei byte seguenti della codifica simbolica in notazione esadecimale, ed infine i commenti):

| INDIRIZZO | CODIFICA | SIMBOLICO | COMMENTI |
|-----------|-----------|------------------|--|
| 18000 | 6 16 | LD B, 10H | ; B = 10 ₁₀ - NUM. MAX. BYTE |
| 18002 | 33 68 72 | LD HL, 4844H | ; HL = 18500 ₁₀ - IND. PARTENZA |
| 18005 | 17 168 72 | LD DE, 48A8H | ; DE = 18600 ₁₀ - IND. ARRIVO |
| 18008 | 126 | LOOP: LD A, (HL) | ; A = (HL) - CARICAMENTO BYTE |
| 18009 | 183 | OR A | ; OR DEL CONTENUTO DI A |
| 18010 | 40 5 | JR Z, FINE | ; SE A=0 SALTO A FINE |
| 18012 | 18 | LD (DE), A | ; (DE) = A - TRASFERIMENTO BYTE |
| 18013 | 19 | INC DE | ; DE = DE + 1 |
| 18014 | 35 | INC HL | ; HL = HL + 1 |
| 18015 | 16 247 | DJNZ LOOP | ; B=B-1 - SE B>0 SALTO A LOOP |
| 18017 | 201 | FINE: RET | ; RITORNO AL PROGRAMMA BASIC |

ROUTINE ASSEMBLER

Per provare questa routine si possono caricare a partire dalla locazione 18500 un certo numero di valori, eseguire la routine e verificare l'avvenuto trasferimento dei byte

Ad esempio si potrebbe avere

| PRIMA | | DOPO | |
|-------|---|-------|---|
| 18500 | 1 | 18500 | 1 |
| 18501 | 2 | 18501 | 2 |
| 18502 | 3 | 18502 | 3 |
| 18503 | 4 | 18503 | 4 |
| 18504 | 0 | 18504 | 0 |
| 18600 | 0 | 18600 | 1 |
| 18601 | 0 | 18601 | 2 |
| 18602 | 0 | 18602 | 3 |
| 18603 | 0 | 18603 | 4 |

Dopo l'esecuzione della routine si può verificare che la coppia di registri HL conterrà 18504 che è l'indirizzo del byte uguale a zero.

2) Routine di traslazione di un blocco di n byte

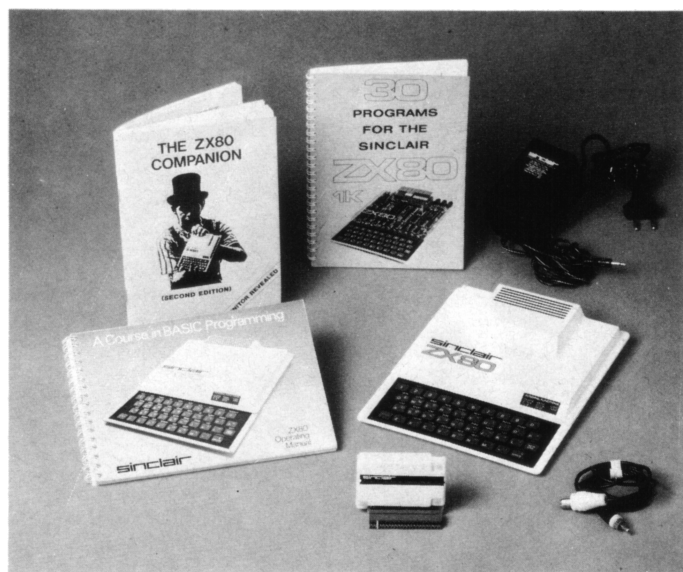
Essa esegue lo spostamento di un blocco di n byte dalla locazione di indirizzo 18500 a quella di indirizzo 18501

Nella codifica che si riporta si è supposto il blocco di lunghezza 5 così da consentire la seguente prova

| PRIMA | | DOPO | |
|-------|---|-------|---|
| 18500 | 1 | 18500 | 1 |
| 18501 | 2 | 18501 | 1 |
| 18502 | 3 | 18502 | 2 |
| 18503 | 4 | 18503 | 3 |
| 18504 | 5 | 18504 | 4 |
| 18505 | 0 | 18505 | 5 |

Alla fine della prova il registro HL conterrà 18499

| INDIRIZZO | CODIFICA | SIMBOLICO |
|-----------|----------|-------------|
| 18000 | 1 5 0 | LD BC 0005H |
| 18003 | 33 68 72 | LD HL 4844H |
| 18006 | 43 | DEC HL |
| 18007 | 9 | add HL, BC |
| 18008 | 84 | LD D, H |
| 18009 | 93 | LD E, L |
| 18010 | 19 | INC DE |
| 18011 | 237 184 | LDDR |
| 18012 | 201 | RET |



3) Routine creazione tabella

Essa esegue, su una tabella del tipo:

| INDIRIZZO | (IND + 0) | (IND + 1) | (IND + 2) |
|-----------|-----------|-----------|-----------|
| 18500 | x | x | 0 |
| 18500 | x | x | 0 |
| 18500 | x | x | 0 |
| 18500 | x | x | 0 |
| 18500 | 0 | 0 | 0 |

Precedentemente caricata in memoria, la somma dei primi due elementi su ogni riga ponendo il risultato nel terzo. Essa si forma quando incontra nella prima colonna un elemento uguale a zero: in tal caso pone nella prima colonna, il totale dei valori della stessa, mentre nella locazione seguente, cioè nella seconda colonna il totale dei valori di essa e nella terza il totale dei totali di riga. Quest'ultimo è poi riportato nella coppia di registri HL.

COMMENTI

```
; BC = 5 - NUM. BYTE BLOCCO
; HL = 1800010 IND. 1° BYTE
; HL = HL-1
; HL = HL + BC - ULTIMO BYTE SOR.
; D = H
; E = L
; DE = DE + 1 - ULTIMO BYTE DEST.
; TRASFERIMENTO BYTES
; RETURN AL PROGRAMMA
; BASIC
```

La codifica è la seguente:

| IND. | CODIFICA | SIMBOLICO | COMMENTI |
|-------|--------------|-------------------|---|
| 18000 | 1 0 0 | LD BC, 0000H | ; BC = 0 |
| 18003 | 221 33 68 72 | LD IX, 4844H | ; IX = 18500 ₁₀ - IND. 1° BYTE |
| 18007 | 17 3 0 | LD DE, 0003H | ; DE = 3 - COST. INCREMENTO IND. |
| 18010 | 62 0 | LOOP: LD A, 00H | ; A = 0 |
| 18012 | 221 190 0 | CP (IX+00H) | ; CONFRONTO ELEMENTO (IX+00H) |
| 18015 | 40 23 | SR Z, FINE | ; (IX+00H) = 0? SE SI → FINE |
| 18017 | 221 126 0 | LD A, (IX+00H) | ; A = (IX+00H) - CARICAMENTO |
| 18020 | 221 134 1 | ADD A, = (IX+01H) | ; A = A + (IX+01H) - SOMMA |
| 18023 | 221 119 2 | LD (IX+02H), A | ; (IX+02H) = A - DEPOSITO |
| 18026 | 221 126 0 | LD A, (IX+00H) | ; A = (IX+00H) |
| 18029 | 128 | ADD A, B | ; A = A + B - SOMMA TOTALI 1° COLONNA |
| 18030 | 71 | LD B, A | ; B = A |
| 18031 | 221 126 1 | LD A, (IX+01H) | ; A = (IX+01H) |
| 18034 | 129 | ADD A, C | ; A = A + C - SOMMA TOTALI 2° COLONNA |
| 18035 | 79 | LD C, A | ; C = A |
| 18036 | 221 25 | ADD IX, DE | ; IX = IX + DE - IND. PROSSIMO BYTE |
| 18038 | 24 226 | JR LOOP | ; SALTO A LOOP |
| 18040 | 221 112 0 | LD (IX+00H), B | ; (IX+00H) = B - TOTALE 1° COLONNA |
| 18043 | 221 113 1 | LD (IX+01H), C | ; (IX+01H) = C - TOTALE 2° COLONNA |
| 18046 | 120 | LD A, B | ; A = B |
| 18047 | 129 | ADD A, C | ; A = A + C |
| 18048 | 221 119 2 | LD (IX+02H), A | ; (IX+02H) = A - TOTALE 3° COLONNA |
| 18051 | 38 0 | LD H, 00H | ; H = 0 - TOTALE 3° COLONNA |
| 18053 | 111 | LD L, A | ; L = A - POSTO IN HL |
| 18054 | 201 | RET | ; RETURN AL PROGRAMMA |
| | | | ; BASIC |

Un esempio di prova può essere il segnale

| PRIMA | | | |
|-----------|-----------|-----------|-----------|
| INDIRIZZO | INDIRIZZO | (IND + 1) | (IND + 2) |
| 18500 | 1 | 2 | 0 |
| 18503 | 16 | 4 | 0 |
| 18506 | 35 | 19 | 0 |
| 18509 | 6 | 36 | 0 |
| 18512 | 0 | 0 | 0 |

| DOPO | | | |
|-----------|-----------|-----------|-----------|
| INDIRIZZO | INDIRIZZO | (IND + 1) | (IND + 2) |
| 18500 | 1 | 2 | 3 |
| 18503 | 16 | 4 | 20 |
| 18506 | 35 | 19 | 54 |
| 18509 | 6 | 36 | 42 |
| 18512 | 58 | 61 | 119 |

Alla fine del "RUN" della routine la coppia di registri HL conterrà il totale di tutti gli elementi contenuti in origine cioè 119.

4) Routine analisi dispositivi:

Essa suppone che al calcolatore siano collegati 8 dispositivi tali che variando il proprio stato da ON a OFF o viceversa rispetto a una configurazione precedentemente registrata, variano pure dei bit ai particolari registri a loro connessi.

Si suppone inoltre che:

- dispositivo ON = 1, dispositivo OFF = 0
- la configurazione precedente è nel registro B, inizialmente caricato da programma
- la configurazione attuale è nel registro A, inizialmente caricato da programma.
- il registro H indica con 1 i bit corrispondenti ai dispositivi che hanno cambiato stato da ON a OFF.
- il registro L indica con 1 i bits corrispondenti ai dispositivi, che hanno cambiato stato da OFF a "ON"

la lista della routine comprende già l'esempio di una prova.

il registro A contiene 137 (configurazione attuale)

il registro B contiene 27 (configurazione precedente)



Allora per quanto dato si otterrà:

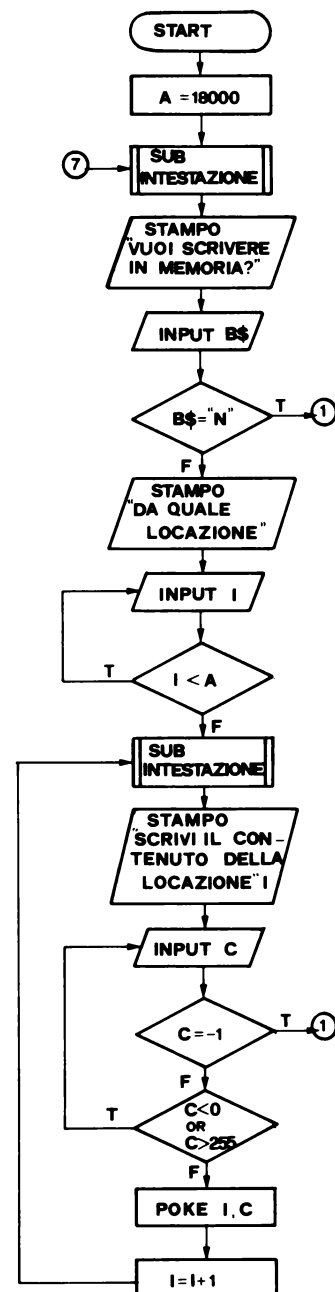
| | | | | | | | | |
|-----------------------|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B (PRECEDENTE) = 27 = | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| A (ATTUALE) = 137 = | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| H (DA 1 → 0) | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| L (DA 0 → 1) | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Allora alla fine della prova la coppia di registri HL conterrà:

$0001001010000000_2 = 4736_{10}$

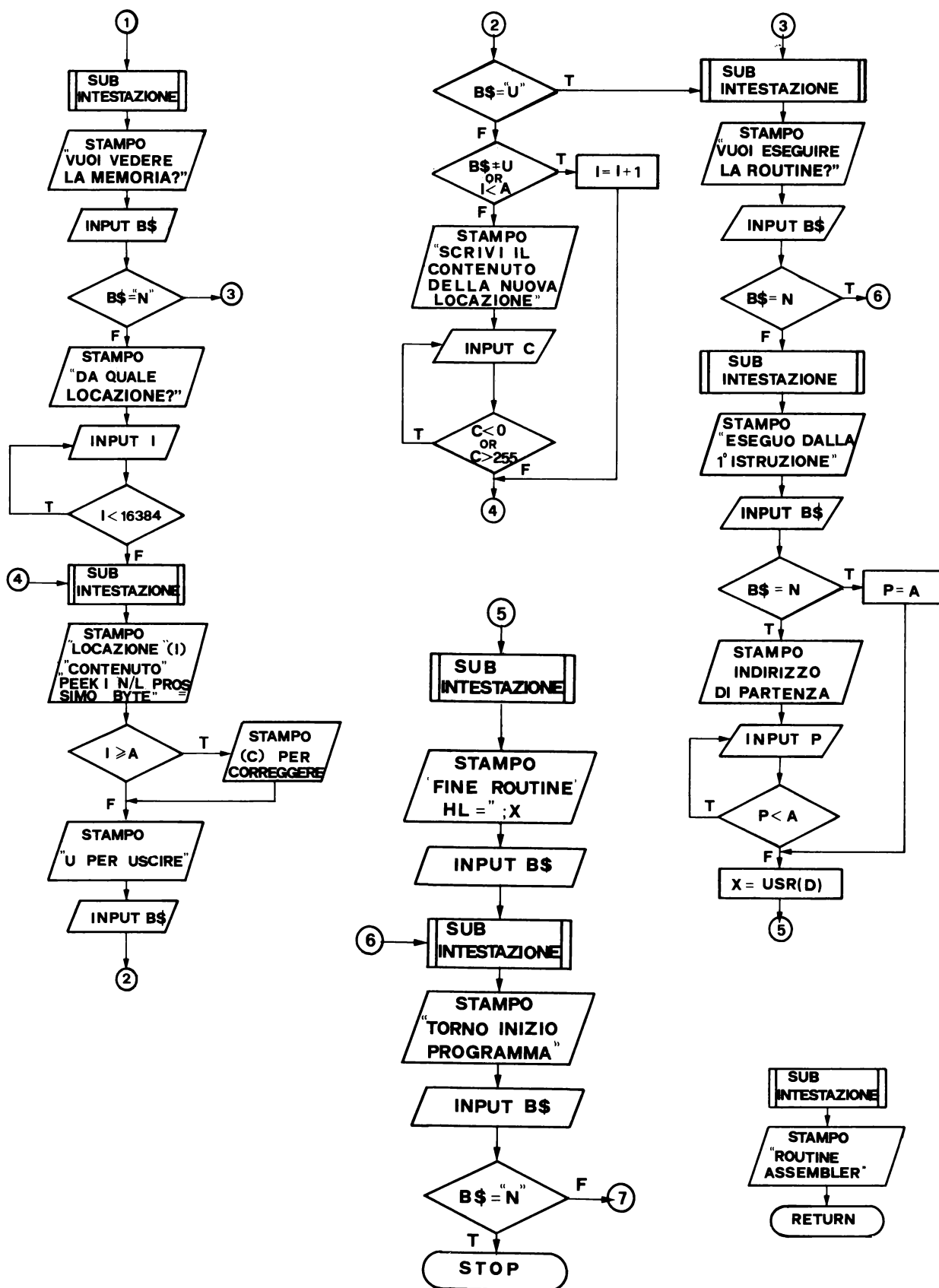
La copia della routine è la seguente:

| IND. | CODIFICA | SIMBOLICO |
|-------|----------|-----------|
| 18000 | 62 137 | LD A, 89H |
| 18002 | 6 27 | LD B, 1BH |
| 18004 | 245 | PUSH A |
| 18005 | 197 | PUSH B |
| 18006 | 245 | PUSH A |
| 18007 | 47 | CPL A |
| 18008 | 160 | AND B |
| 18009 | 103 | LD A, A |
| 18010 | 120 | LD A, B |
| 18011 | 193 | POP B |
| 18012 | 47 | CPL A |
| 18013 | 160 | AND B |
| 18014 | 111 | LD L, A |
| 18015 | 193 | POP B |
| 18016 | 241 | POP A |
| 18017 | 201 | RET |



COMMENTI

; A = 137₁₀ - CONF. ATTUALE
 ; B = 27₁₀ - CONF. PRECEDENTE
 ; A → NELLA STACK AEREA
 ; B → NELLA STACK AEREA
 ; A → NELLA STACK AEREA
 ; SE A = B
 ; ALLORA BIT DI H = 1
 ;
 ; A = B
 ; CARICA B DALLA STACK AREA
 ; SE A = B
 ; ALLORA BIT DI L = 1
 ;
 ; SI RIPRISTINA B
 ; SI RIPRISTINA A
 ; RETURN AL PROGRAMMA BASIC



ROUTINE ASSEMBLER

```

10 LET A = 18000
15 GO SUB 1000
20 PRINT "VUOI SCRIVERE IN MEMORIA (S/N)?"
25 PRINT
30 INPUT B$
40 IF B$ = "N" THEN GO TO 200
50 PRINT
55 PRINT "DA QUALE LOCAZIONE?"
60 PRINT
65 INPUT I
70 IF I < A THEN GO TO 65
75 GO SUB 1000
80 PRINT "SCRIVI IL CONTENUTO"
85 PRINT "DELLA LOCAZIONE"; I
90 PRINT "<-1> PER USCIRE:"
95 PRINT
100 INPUT C
110 IF C = -1 THEN GO TO 200
120 IF C < 0 OR C > 255 THEN GO TO 100
130 POKE I, C
140 LET I = I + 1
150 GO TO 75
200 GO SUB 1000
210 PRINT "VUOI VEDER LA MEMORIA (S/N)?"
215 PRINT
220 INPUT B$
230 IF B$ = "N" THEN GO TO 500
240 PRINT
250 PRINT "DA QUALE LOCAZIONE?"
255 PRINT
260 INPUT I
270 IF I < 16384 THEN GO TO 260
280 GO SUB 1000
290 PRINT
300 PRINT "LOCAZIONE: "; I
310 PRINT
320 PRINT "CONTENUTO: "; PEEK (I)
330 PRINT
335 PRINT
340 PRINT "<N/L> PER IL PROSSIMO BYTE"
350 IF NOT I < A THEN PRINT "<C>
    PER CORREGGERE"
360 PRINT "< U > PER USCIRE"
365 PRINT
370 INPUT B$
380 IF B$ = "U" THEN GO TO 500
390 IF NOT B$ = "C" OR I < A THEN GO TO 460

400 PRINT
405 PRINT
410 PRINT "SCRIVI IL NUOVO CONTENUTO"
414 PRINT "DELLA LOCAZIONE"; I; ":"
415 PRINT
420 INPUT C
430 IF C < 0 OR C > 255 THEN GO TO 420
440 POKE I, C
450 GO TO 280
460 LET I = I + 1
470 GO TO 280
500 GO SUB 1000
510 PRINT "VUOI ESEGUIRE LA ROUTINE (S/N)?"
515 PRINT
520 INPUT B$
530 IF B$ = "N" THEN GO TO 800
535 GO SUB 1000
540 PRINT "ESEGUO LA ROUTINE DALLA"
550 PRINT "PRIMA ISTRUZIONE (S/N)?"
555 PRINT
560 INPUT B$
570 IF B$ = "N" THEN GO TO 600
580 LET P = A
590 GO TO 640
600 PRINT
610 PRINT "INDIRIZZO DI PARTENZA?"
615 PRINT
620 INPUT P
630 IF P < A THEN GO TO 620
640 LET X = USR (P)
650 GO SUB 1000
660 PRINT
670 PRINT "ALLA FINE DELLA ROUTINE"
680 PRINT "LA COPPIA DI REGISTRI HL"
690 PRINT "CONTIENE:"; X
700 PRINT
705 PRINT
710 PRINT "<N/L> PER CONTINUARE"
715 PRINT
720 INPUT B$
800 GO SUB 1000
810 PRINT "TORNO A INIZIO PROGRAMMA (S/N)?"
815 PRINT
820 INPUT B$
830 IF NOT B$ = "N" THEN GO TO 10
840 GO SUB 1000
845 PRINT
850 PRINT "FINE PROGRAMMA"
860 STOP
1000 CLS
1010 PRINT " ***ROUTINE ASSEMBLER*** "
1020 PRINT
1030 PRINT
1040 RETURN

```


DISEGNI

Autore: **A. Napoletano**
Programma utilizzante:
4 k di memoria

PROGRAMMA

Questo programma non ti insegnerà come diventare un Van Gogh, ma ti permetterà di giocare con i caratteri grafici cosicché i tuoi programmi potranno beneficiare di una migliore visualizzazione dei risultati.

Le pagine seguenti illustrano alcuni dei caratteri grafici con il loro numero di codice associato.

Questi caratteri sono anche disponibili in stampa in qualsiasi momento dell'esecuzione del programma.

Per ottenere la tabella dei caratteri e dei codici associati basta premere <1> e New Line in risposta a questa domanda:

CODICE CARATTERE, < 1 > PER LA TABELLA

La selezione del codice carattere si può anche ottenere con la tabella visualizzata nello schermo, inserendo il codice prescelto in risposta alla domanda:

COD. CARATT. <1> PER IL DISEGNO

Se si risponde <1>, viene visualizzato di nuovo il disegno e quindi si può inserire il codice carattere prescelto tenendo presente il disegno.

Per cancellare un carattere in una determinata posizione basta inserire le coordinate del carattere, e quindi inserire il codice 0.

Viene effettuato anche un controllo sulla correttezza delle coordinate, se vengono impostate coordinate che eccedono o non sono comprese nella misura prescelta esse vengono ignorate, e richieste di nuovo. Per terminare la composizione si deve inserire <0> (zero) alla richiesta:

SCRIVI LE COORDINATE, < 0 > PER USCIRE

Si può ottenere infine anche la tabella con i codici impiegati nella composizione.

Il programma inoltre permette più di una composizione in quanto di ferma attendendo una risposta alla domanda.

VUOI COMPORRE ANCORA, <S/N>?

```
75 LET W = 0
85 CLS
90 PRINT "ΔΔΔΔΔ****DISEGNI****"
95 LET S = 0
96 PRINT
97 PRINT
100 PRINT
110 PRINT
120 PRINT "MISURA (3 - 7)?"
130 INPUT N
```

```
140 IFN <3 OR N> 7 THEN GO TO 120
150 DIM A (N * N - 1)
155 GOTO 200
199 INPUT W $
200 CLS
202 PRINT "ΔΔΔΔΔ****DISEGNI****"
203 PRINT
204 PRINT
205 PRINT
206 PRINT
207 PRINT
210 FOR K = 1 TO N
220 LET J = N + 1 - K
225 PRINT "ΔΔΔΔΔΔΔΔ";
230 PRINT J,
240 FOR I = 1 TO N
250 PRINT CHR $ (A(J-N-1+N*I));
260 NEXT I
270 PRINT
```

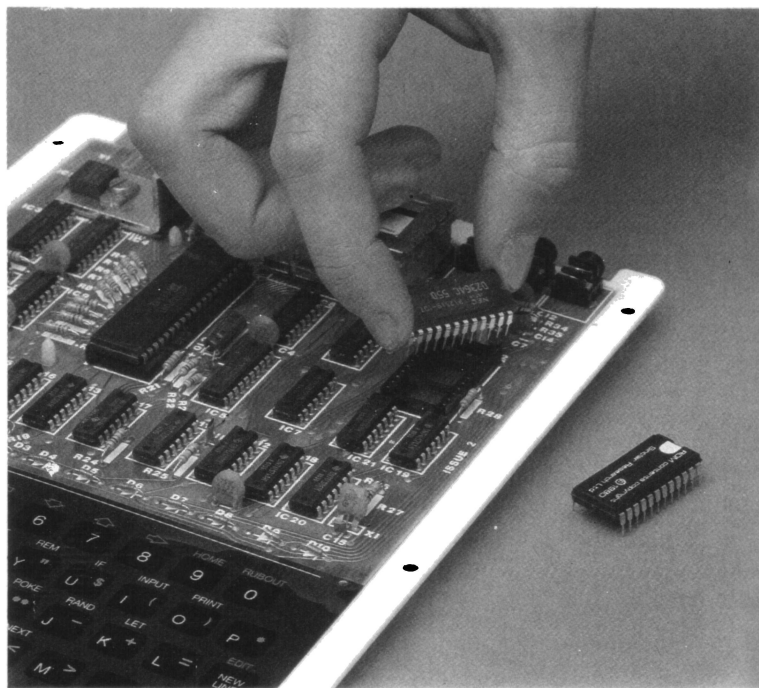



```

280 NEXT K
290 PRINT
300 PRINT,
310 FOR I = 1 TO N
320 PRINT I;
330 NEXT I
340 PRINT
345 PRINT
347 IF W = 1 THEN GOTO 460
400 PRINT "SCRIVI LE COORDINATE, <0> PER USCIRE"
410 INPUT X
420 IF X = 0 THEN GOTO 491
425 LET J = X/10
426 LET I = X-10 * J
430 IF I > N OR I < 1 THEN GOTO 200
435 IF J > N OR J < 1 THEN GOTO 200
440 LET U = J
441 LET Z = I
450 PRINT X
460 PRINT "CODICE CARATTERE, <1> PER LA TABELLA"
470 INPUT X
474 IF NOT X = 1 AND W = 0 THEN GOTO 485
475 IF X = 1 THEN GO SUB 700
476 IF W = 1 THEN GOTO 485
477 PRINT "COD. CARATT., <1> PER DISEGNO"
478 INPUT W
479 IF W = 1 THEN GOTO 200
480 LET X = W
485 LET A (U - N - 1 + N * Z) = X
486 LET W = 0
490 GOTO 200
491 CLS

```

Lo ZX80 può diventare più potente e versatile semplicemente sostituendo la ROM preesistente con la nuova ROM 8K. Le fasi della modifica, a partire dallo scoperchiamento qui sotto riportato, sono illustrate a lato e a pagina 39.

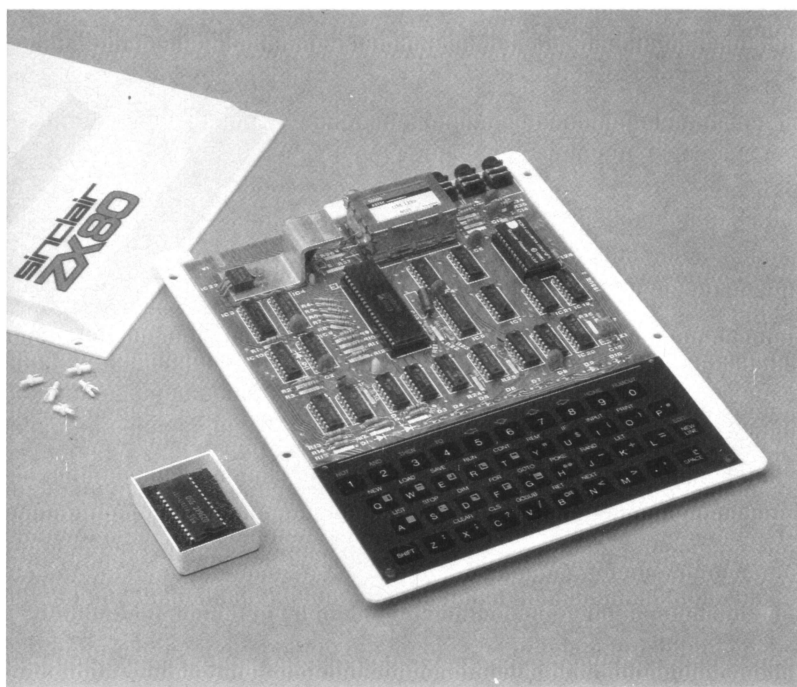


La delicata fase di sostituzione della ROM.

```

493 PRINT
495 PRINT "ΔΔΔΔΔΔ****DISEGNO FINALE****"
496 PRINT
497 PRINT
498 PRINT
500 FOR K = 1 TO N
510 LET J = N + 1 - K
515 PRINT "ΔΔΔΔΔΔ";
520 FOR I = 1 TO N
530 PRINT CHR $ (A(J - N - 1 + N * I));
540 NEXT I
550 PRINT
560 NEXT K
565 PRINT
566 PRINT
567 PRINT
606 PRINT
607 PRINT
608 PRINT
610 PRINT "VUOI LA TABELLA DEI CODICI IMPIEG. <S/N>?"
615 INPUT A$
620 IF A$ = "N" THEN GOTO 655
625 CLS
626 PRINT " ΔΔΔΔΔΔ****DISEGNI**** "
627 PRINT
628 PRINT
629 PRINT
630 PRINT "CODICI IMPIEGATI:"
631 PRINT "L CODICI"
632 PRINT
633 PRINT
634 FOR K = 1 TO N
635 LET J = N + 1 - K
637 PRINT J; "ΔΔ";

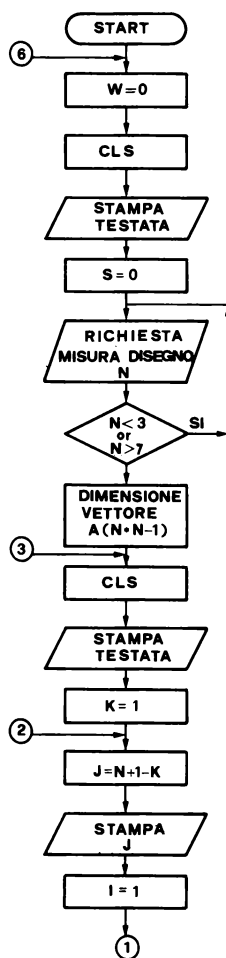
```



```

640 FOR I = 1 TO N
642 PRINT "Δ"; A (J - N - 1 + N * I);
643 LET A (J - N - 1 + N * J) = 0
644 NEXT I
645 PRINT
646 PRINT
650 NEXT K
651 INPUT Q$
655 CLS
660 PRINT "ΔΔΔΔΔ****DISEGNI****"
663 PRINT
664 PRINT
665 PRINT
666 PRINT
670 PRINT "VUOI COMPORRE ANCORA, <S/N>?"
680 INPUT A$
690 IF A$ = "S" THEN GOTO 75
695 CLS
696 STOP
700 CLS

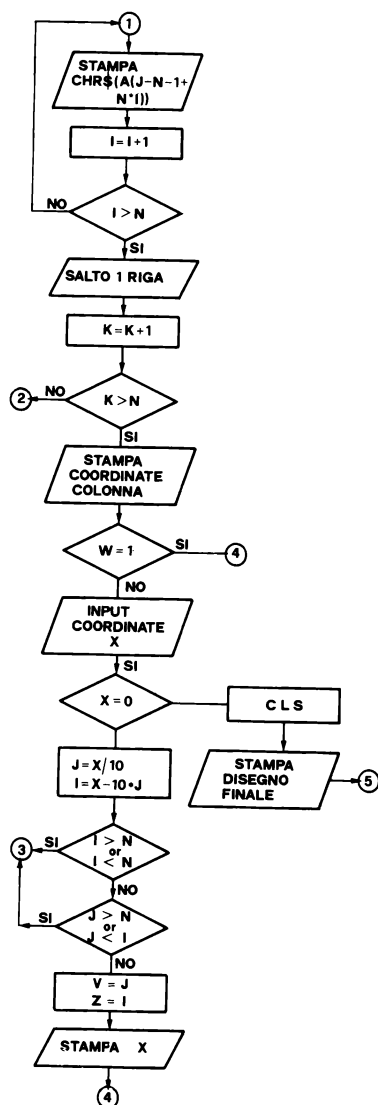
```

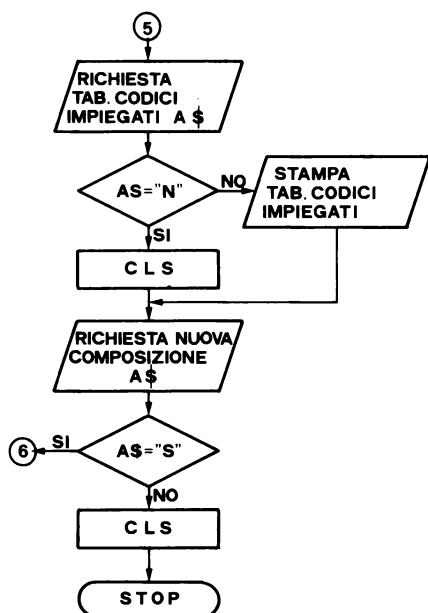
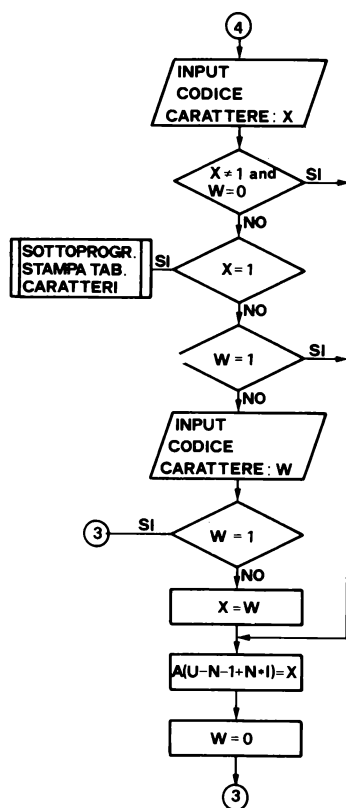


```

705
710 PRINT "ΔΔΔΔ****TABELLA CARATTERI****"
720 PRINT
735 LET S = 2
740 LET V = S + 1
745 PRINT
746 PRINT S, CHR$ (S), V, CHR$ (V)
750 LET S = S + 2
755 IF S > 10 THEN GOTO 765
760 GOTO 740
765 LET S = 128
770 LET V = S + 1
775 PRINT
776 PRINT S, CHR$ (S), V, CHR$ (V)
780 LET S = S + 2
785 IF J = 136 THEN LET W = S + 6
790 IF S = 136 THEN GOTO 775
795 IF S > 136 THEN GOTO 805
800 GOTO 770
805 RETURN

```





NIM

Autore: **S. Nichelini**
Programma utilizzante:
4 k di memoria

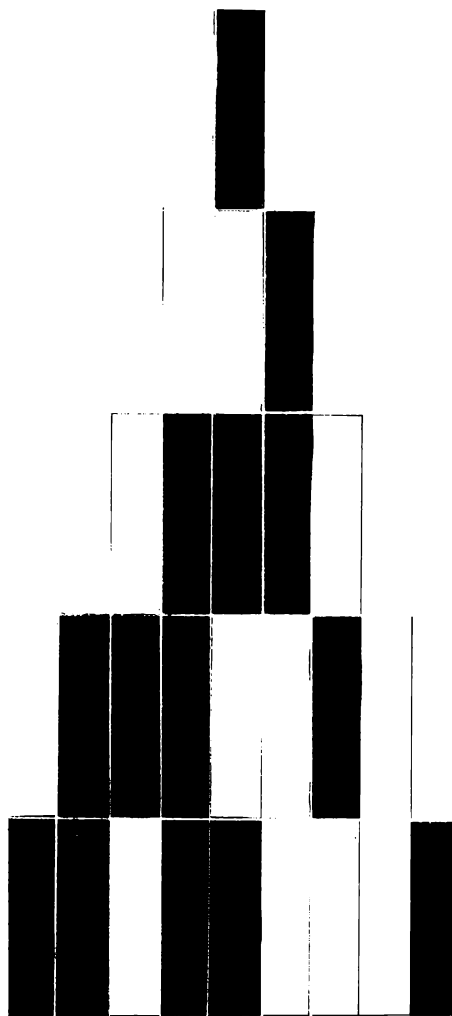
ISTRUZIONI GENERALI

NIM - Il gioco reso celebre dal film "Last year at Marienbad"

Questo gioco, che si pratica in due, consiste nell'eliminare, a turno, dei quadratini da uno schema visualizzato sul video. Inizialmente tale schema è composto da 5 righe di, rispettivamente, 9, 7, 5, 3 e 1 quadratini.

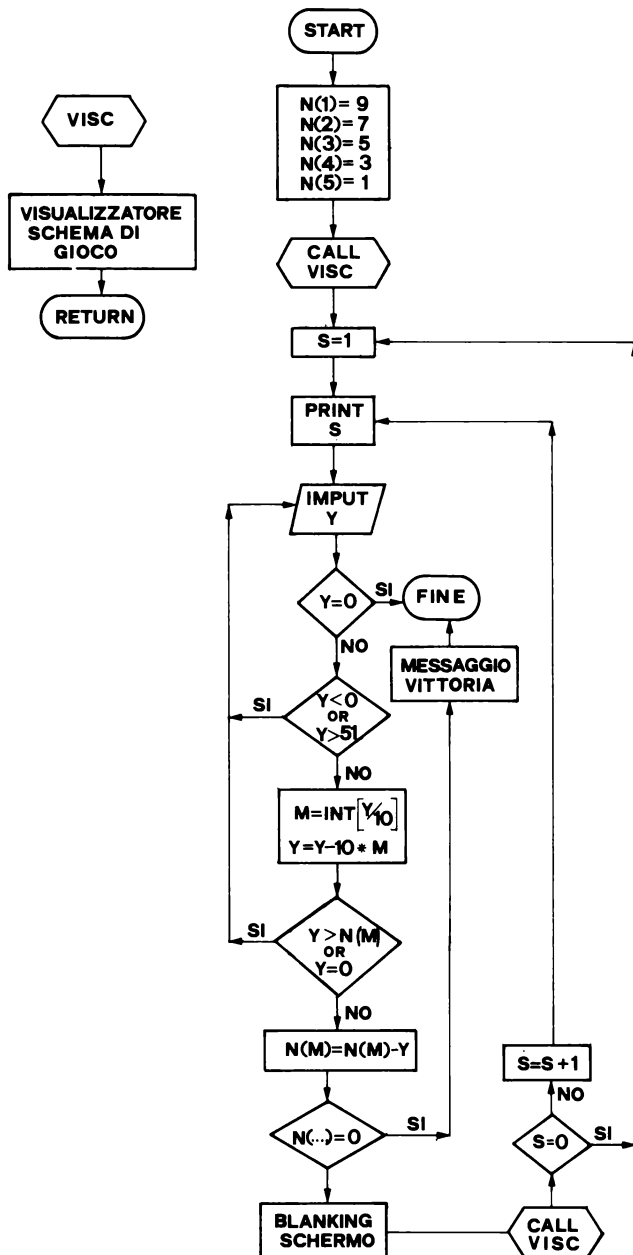
Durante il gioco si possono eliminare, da una sola riga quanti quadratini si vuole, con un minimo di 1.

Esistono varie versioni di questo gioco: obiettivo di questa versione è quello di eliminare l'ultimo o gli ultimi quadratini.



Per indicare il numero di riga ed il numero di quadratini da eliminare, si usano due cifre; la prima per il numero di riga, la seconda per il numero di quadratini (es: scrivendo 14 si eliminano 4 quadratini dalla prima riga); nel caso in cui si tenti di eliminare un numero di quadratini maggiore di quelli attualmente presenti sulla determinata riga, verrà richiesto di nuovo l'inserimento delle due cifre.

Nel caso in cui si volesse uscire dal gioco prima della vittoria di uno dei due giocatori, alla richiesta si deve inserire 0.



```

100 DIM N(5)
110 LET N(1) = 9
120 LET N(2) = 7
130 LET N(3) = 5
140 LET N(4) = 3
150 LET N(5) = 1
160 GOTO 500
200 PRINT "*** ... 32 asterischi"
201 PRINT "ΔΔΔ*** NIM ***"
202 PRINT "*** ... 32 asterischi"
203 PRINT
204 PRINT
209 FOR R = 1 TO 5
210 PRINT R;
220 FOR M = 1 TO N(R)
230 IF (N(R) = 0) THEN GOTO 260
240 PRINT "Δ";
250 NEXT M
260 PRINT
270 PRINT
280 NEXT R
290 RETURN
500 GOSUB 200
510 LET S = 1
520 PRINT
522 PRINT "GIOCATORE:" ; S
524 PRINT
525 PRINT
530 PRINT "NUMERO DI RIGA E DI ELEMENTI"
540 INPUT Y
550 IF Y = 0 THEN GOTO 900
560 IF Y < 0 OR Y > 51 THEN GOTO 540
570 LET M = Y/10
580 LET Y = Y - 10 * M
590 IF Y > N(M) OR Y = 0 THEN GOTO 540
600 LET N(M) = N(M) - Y
610 IF N(1) = 0 AND N(2) = 0 AND N(3) = 0 AND
N(4) = 0 AND N(5) = 0 THEN GOTO 800
620 CLS
630 GOSUB 200
640 IF S = 2 THEN GOTO 510
650 LET S = S + 1
660 GOTO 520
800 CLS
801 GOSUB 200
802 PRINT
804 PRINT
806 PRINT "HA VINTO IL GIOCATORE": S
810 PRINT "COMPLIMENTI ..."
900 PRINT
905 PRINT
910 PRINT "LA PARTITA SI CHIUDE".

```

ALLUNAGGIO

Autore: S. Gioia
Programma utilizzante:
4 k di memoria

Questo programma permette al giocatore di tentare l'allunaggio con una astronave che inizialmente si trova in queste condizioni:

| Altezza | Velocità | Carburante |
|---------|-----------|------------|
| 1500 m | - 50 km/h | 10.000 l |

Ogni velocità negativa avvicina l'astronave al satellite, viceversa ogni velocità positiva tende ad allontanare la navicella dalla luna.

Per modificare l'altezza, la velocità ed eventualmente il carburante è necessario operare su alcuni input (2) che il programma richiede in fase di esecuzione:

SPINTA (0 - 99)?
DURATA (1 - 10)?

In entrambi questi input occorre inserire un numero appartenente all'insieme numerico delimitato dai numeri messi tra parentesi; contrariamente il programma richiede automaticamente lo stesso input fino a che non si verifica la condizione detta.

Come si nota la spinta è sempre maggiore o uguale a zero; in questo ultimo caso l'astronave diminuisce di velocità in quanto è soggetta ad attrazione gravitazionale lunare.

La durata rappresenta il tempo per cui deve mantenere la spinta.

In caso di allunaggio il programma calcola la percentuale di errore in base alla velocità di arrivo; ciò rende il gioco più interessante e competitivo.

NOTA - Il programma deve essere mandato in esecuzione con l'istruzione GOTO 100; nel caso il giocatore desse l'istruzione RUN, avrà il compito di inserire un vettore di 20 posizioni che permette di visualizzare l'astronave. I numeri di inserire in sequenza sono:

0, 0, 156, 0, 0, 0, 8, 3, 136, 0, 0,
2, 3, 130, 0, 134, 131, 3, 131, 135.

Si ricorda che ogni volta che si manda in esecuzione un programma con l'istruzione RUN, tutte le variabili vengono azzerate.


| | | |
|-----------|-------|-----------------------------------|
| VARIABILI | H,V,R | Altezza, Velocità, Carburante. |
| USATE | F, T | Spinta, Durata. |
| | A (I) | Figura dell'astronave. |
| | L | Linee tra l'astronave e la luna. |
| | X, S | variabili di comodo |

DESCRIZIONE GENERALE

| | |
|---------|--|
| 50- 85 | caricamento vettore A (I) necessario per la stampa dell'astronave. |
| 200-260 | stampa dell'astronave. |
| 300-350 | INPUT e relativi controlli. |
| 370-730 | elaborazione e controlli. |
| 50 | DIM A (19) |
| 55 | FOR I = 0 TO 19 |
| 60 | PRINT I, |
| 65 | INPUT X |
| 70 | LET A(I) = X |




```

75 PRINT A(I)
80 NEXT I
85 CLS
100 V = -50
110 H = 1500
120 R = 10000
122 PRINT "**** LUNAR - LANDER ****"
124 PRINT
130 GOTO 500
200 FOR I = 0 TO 3
207 PRINT "ΔΔΔΔΔΔΔΔΔΔ"
210 FOR X = 0 TO 4
220 PRINT CHR$(A(X + 5 * I));
230 NEXT X
240 PRINT
250 NEXT I
260 RETURN
300 PRINT "SPINTA (0 - 99)?",
310 INPUT F
316 IF F < 0 OR F > 99 THEN GO TO 310
320 PRINT F
330 PRINT "DURATA (1 - 10)?",
340 INPUT T
344 IF T < 1 OR T > 10 THEN GO TO 340
350 CLS
370 LET R = R - F * T * 10
400 LET A = F - 32
410 LET H = A * T ** 2 + V * T + H
420 LET V = 2 * A * T + V
500 LET L = H/100
510 IF L < 0 THEN LET L = 0
520 IF L > 12 THEN LET L = 12
530 FOR I = L TO 12
540 PRINT
550 NEXT I
560 IF L > 8 THEN GOSUB 200
564 PRINT "ΔΔΔ VEL", "ALT", "CARB"
568 IF H > 0 AND H < 150 AND V < 1 AND V >
- 99 THEN LET H = 0
569 IF R > 0 THEN GO TO 578
570 PRINT "ΔΔ"; V, H, "0"
571 PRINT
572 PRINT "**** CARBURANTE FINITO ****"
577 STOP
578 LET S = H
580 IF S < 0 THEN LET S = 0
582 IF H < 0 AND V > 0 THEN LET V = - V
585 IF V > 0 AND H = 0 THEN LET S = 10
595 PRINT "ΔΔΔ"; V, S, R
596 IF NOT L > 8 THEN GOSUB 200
600 IF H = 0 THEN GO TO 650
620 FOR I = 1 TO L - 1
630 PRINT
640 NEXT I
650 IF H < 0 THEN PRINT "CRASH"
655 IF H = 0 AND V < - 99 THEN PRINT
"CRASH"
660 FOR I = 1 TO 16
670 PRINT "  ";
680 NEXT I
690 IF H > 0 OR V > 0 THEN GO TO 300
695 IF H < 0 OR V < - 99 THEN GO TO 730
705 LET V = - V
707 PRINT
710 PRINT "PERC. ERR. 0,"; V; "%"
720 STOP
730 PRINT "TERRIBILE"

```

PINCK

Autore: L. Castagna
Programma utilizzante:
4 k di memoria

È un gioco apparso per la prima volta nel 1980 in America, è l'equivalente del gioco giapponese GO. Due giocatori alternativamente, collocano delle "pietre" su una tavola a 8 posizioni: un gruppo di pietre consecutive è considerato catturato se è circondato da ogni lato. Un gruppo sul bordo è il più facile da catturare. È un gioco interessante. Una volta studiate le strategie delle 8 posizioni, si può provare con 9 posizioni, in questo caso le strategie saranno differenti.

```

95 PRINT "PINCH"
96 PRINT
97 PRINT
105 DIM T(2)
110 PRINT "NOME I° GIOCATORE"
120 INPUT Y $
130 PRINT
135 PRINT "NOME II° GIOCATORE"
140 INPUT Z $
150 LET A $ = Y $
160 LET B = 1
170 GO TO 600
200 IF G(I) = 0 THEN RETURN
202 IF G(I) = 1 THEN GO TO 220
204 LET C = 2
206 LET D = 1
208 GO TO 220
220 LET C = 1
215 LET D = 2
220 LET K = I
221 LET K = K - 1
222 IF K = 0 OR G(K) = D THEN GO TO 235
225 IF G(K) = C THEN GO TO 221
230 IF G(K) = 0 THEN GO TO 340
235 LET M = K + 1
240 LET K = I
245 LET K = K + 1
250 IF K = 9 OR G(K) = D THEN GO TO 265
255 IF G(K) = C THEN GO TO 245
260 IF G(K) = 0 THEN GO TO 340
265 LET N = K - 1
270 FOR K = M TO N
275 LET G(K) = 0
280 NEXT K
285 LET T(C) = T(C) + (N-M) + 1
340 CLS
350 PRINT
360 PRINT "**** PINCH ****"
370 PRINT
375 PRINT
380 PRINT
390 FOR J = 1 TO 8
400 PRINT "ΔΔ"; CHR$(6 + 3 * G(J));
410 NEXT J
420 PRINT
430 PRINT "Δ";
440 FOR J = 1 TO 8

```

PINCK



```

450 PRINT "Δ [X] Δ";
460 NEXT J
470 PRINT
480 PRINT
490 FOR J = 1 TO 8
500 PRINT "Δ Δ"; J;

```

Esempi:

A)

1 2 3 4 5 6 7 8

B)

1 2 3 4 5 6 7 8

- A) Se in questo esempio volesse muovere il giocatore X, una mossa che mettesse una pietra sulla posizione 7 catturerebbe il gruppo alla posizione 8. Se dovesse muovere il giocatore 0, muovendo una pietra sul 2 catturerebbe il gruppo alle posizioni 3 e 4.

- B) Se in questo esempio dovesse muovere il giocatore X, posizionando una pietra sulla posizione 1, si autocatturerebbe o autochiuderebbe.

ANALISI

Il programma inizia chiedendo i nomi dei 2 giocatori. Dopo aver assegnato il codice ad ogni giocatore (1 o 2) è possibile risalire al simbolo di ogni singolo giocatore tramite l'istruzione `CHR$(6 + 3 * B)` con B uguale a 1 o a 2. Le linee che partono dallo 630 e arrivano alla 680 servono a controllare la validità della mossa e a controllare che il giocatore non voglia posizionare la propria pietra in una posizione occupata del tavolo da gioco. Scrivendo 0 alla richiesta della mossa si ottiene che il programma termini con la stampa dei risultati finali. Mediante una subroutine richiamata dalle linee che vanno dalla 700 alla 720, viene stabilito se la pietra posata dal giocatore fa parte di un gruppo chiuso; potrebbe succedere cioè, che un giocatore si autocatturi con una sua stessa mossa. Se è così allora la mossa ha autochiuso il giocatore che quindi regala punti all'avversario. Tramite le linee 710-720 si esaminano le posizioni immediatamente a destra e immediatamente a sinistra per determinare se queste ultime fanno parte di gruppi chiusi. In questo caso il giocatore realizza punti a proprio favore. Infine vi è il caso in cui una mossa sia senza risultato, se così fosse nessun giocatore guadagnerebbe punti. Determinato ciò si incrementa il contatore relativo al diritto a muovere da parte di un giocatore; se questo contatore è 3, esso viene riposizionato a 1 per riprendere il giro. Quando alla richiesta della mossa si risponde con uno 0 il programma salta alla linea 900 dalla quale inizia la fase di stampa dei risultati finali e del nome del vincitore.

Per stabilire il nome del vincitore viene analizzato il contatore dei punti accumulati da entrambi i giocatori (T(C)) e a seconda del valore di T (1) e T (2) si ha la stampa del vincitore. Le linee 340-515 servono per la stampa della tavola da gioco con le 8 posizioni.

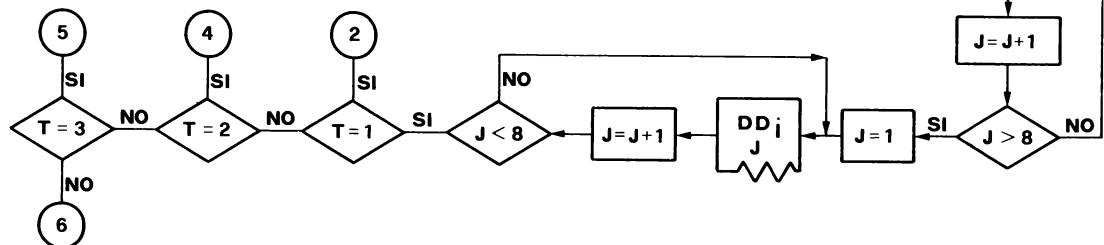
```

510 NEXT J
515 PRINT
520 RETURN
600 GO SUB 340
610 PRINT
620 PRINT A$; "(", CHR$(6 + 3 * B); "): MOSSA?"
630 INPUT I$
640 LET I = CODE(I$) - 28
650 IF I = 0 THEN GO TO 900
660 IF I > 0 AND I < 9 THEN GO TO 680
670 GO TO 630
680 IF NOT G(I) = 0 THEN GO TO 630
690 LET G(I) = B
700 GO SUB 200
710 LET I = I + 1
712 IF I > 8 THEN GO TO 716
714 GO SUB 200
716 LET I = I - 2
718 IF I < 1 THEN GO TO 740
720 GO SUB 200
740 LET B = B + 1
750 IF B = 3 THEN LET B = 1
760 IF B = 2 THEN LET A$ = Z$
770 IF B = 1 THEN LET A$ = Y$
780 GO TO 610

```


REPERTORIO

I\$: mossa eseguita dal giocatore attualmente in gioco
I : mediante l'istruzione CODE (I\$) - 28, la mossa del giocatore, che è contenuta in I\$, viene tradotta nella posizione corrispondente della tavola.



CAPITALI

Autore: E. Vighi
Programma utilizzante:
4 K di memoria

Questo programma mostra come si può usare un calcolatore per scopi didattici.

All'operatore vengono proposte in un ordine casuale 10 domande sulle capitali di alcune nazioni del mondo, se egli risponde subito in modo corretto, gli viene assegnato un punteggio 10, se la prima risposta è errata, ma la seconda risposta alla domanda è giusta, gli viene assegnato un punteggio 5, se la risposta alla domanda è per due volte sbagliata gli viene assegnato il punteggio 0 e si visualizza la risposta esatta.

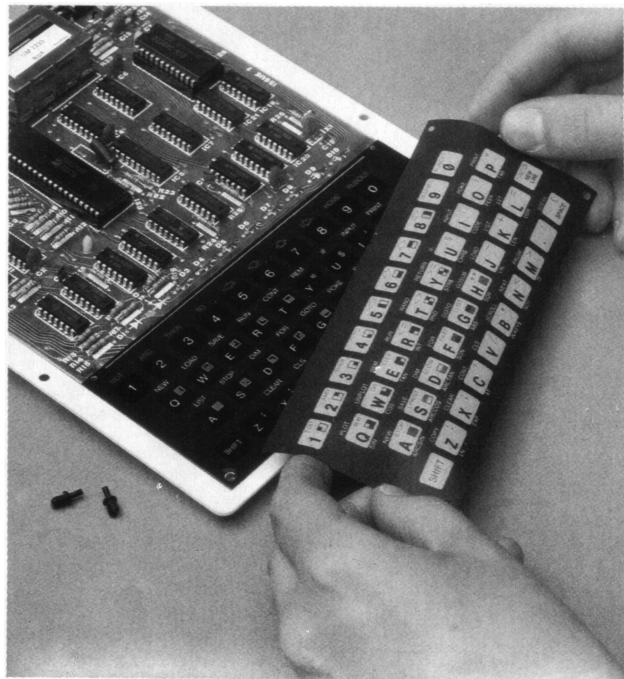
Ripetendo questo procedimento per tutte le 10 domande, alla fine del questionario è sufficiente sommare i punteggi ottenuti per ciascuna domanda per ottenere un indice percentuale della conoscenza dell'argomento. L'intervallo dei valori che possono essere assunti dall'indice citato è costituito da tutti i multipli di 5 compresi fra 0 e 100.

È evidente che lo stesso schema di programma può essere facilmente adattato a qualsiasi tipo di domande.

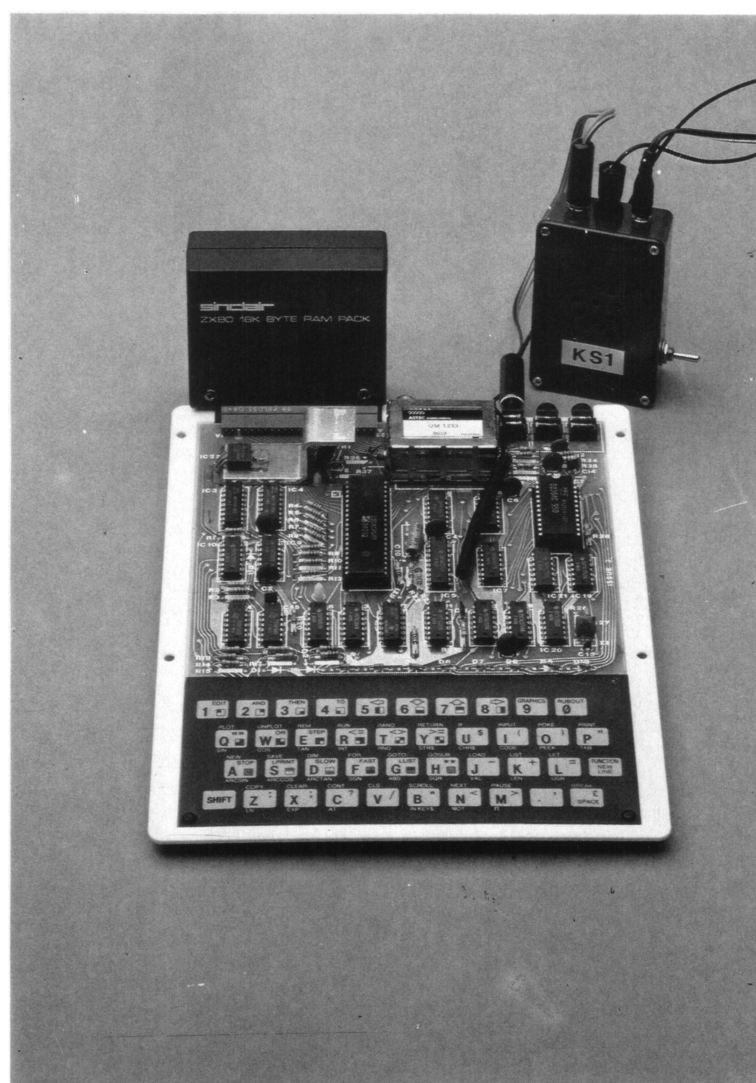
Si noti che sia le domande che le risposte vengono elaborate in una sola variabile (A\$), nella quale sono inserite nella forma:

<nome nazione> ? <nome capitale>

La stampa del <nome nazione> avviene prendendo un carattere alla volta dalla stringa, con la funzione CODE, e togliendo poi il carattere già stampato, mediante la funzione TL\$.



Applicazione della nuova mascherina dopo aver rimosso le clips.



Lo ZX80 dotato di nuova ROM, tastiera, espansione a 16K e piccolo altoparlante.

Ricordo che la CODE (A\$) riporta il codice del primo carattere della variabile A\$, mentre la TL\$ (A\$) priva la variabile A\$ del suo primo carattere.

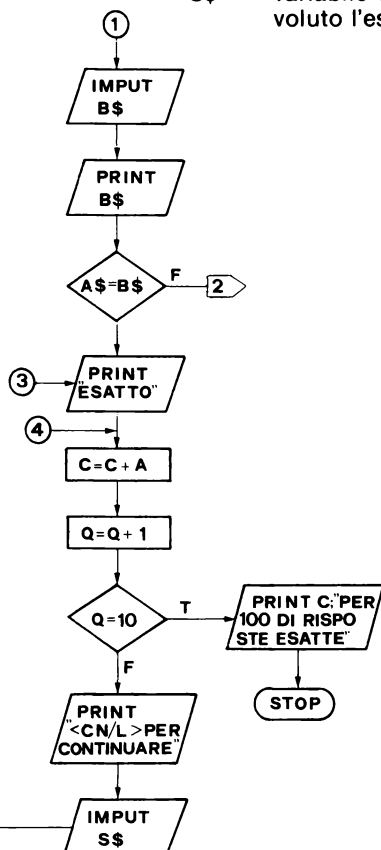
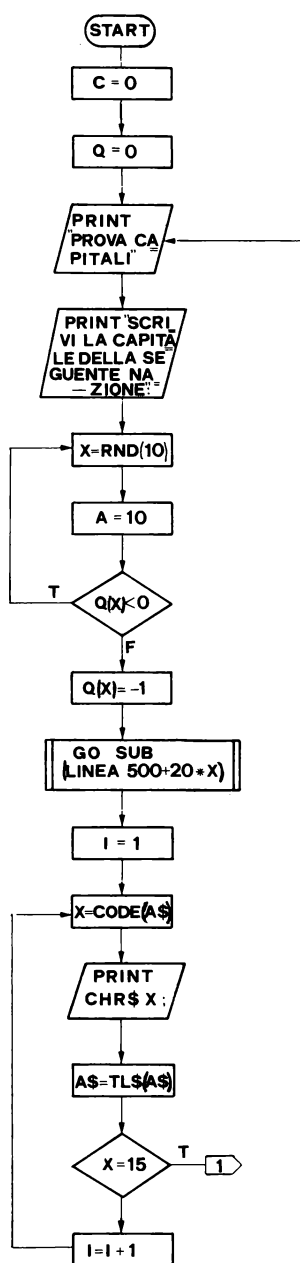
Il ciclo di stampa descritto precedentemente è ripetuto fino al carattere "?", il cui codice è 15. Nel programma si è fatto uso di un ciclo 200 FOR I = 1 TO 30 che in realtà non verrà mai eseguito per 30 volte, poichè i nomi delle nazioni sono tutti più brevi, quindi il carattere "?" verrà trovato prima della fine del ciclo.

L'istruzione di controllo in questo caso è la 240 IF x = 15 THEN GO TO 300.

Le linee 305 Print B\$ e 415 PRINT B\$, che seguono delle istruzioni di lettura della variabile B\$, sono state introdotte allo scopo di mantenere visualizzata sullo schermo la risposta ad una domanda anche dopo la lettura di questa risposta nella variabile B\$.

Le linee 350 e 355 servono ad interrompere per il tempo desiderato l'esecuzione del programma e consentono di mantenere sul video la situazione alla fine di ogni domanda per tutto il tempo che si vuole, prima di passare alla domanda successiva.

CAPITALI



Variabili usate

Q(10) vettore di controllo delle domande già fatte. Se $Q(x) = -1$, la domanda contenuta nella subroutine della linea $(500 + 20 * x)$ è già stata fatta; altrimenti se $Q(x) = 0$ la domanda della subroutine che inizia alla linea $(500 + 20 * x)$ non è ancora stata posta.

C contatore dei punti realizzati

A punti realizzati nella domanda considerata. Valori possibili 0,5, 10

Q contatore domande già fatte; varia da 1 a 10

A\$ variabile usata per elaborare le domande e le risposte.

B\$ variabile usata per leggere le risposte.

S\$ variabile di comando usata per ritardare come voluto l'esecuzione del programma.

Le 10 subroutine non sono state indicate nel diagramma a blocchi per la loro estrema semplicità

Descrizione generale programma:

Linee 100-120 : inizializzazione variabili

Linee 130-148 : intestazione

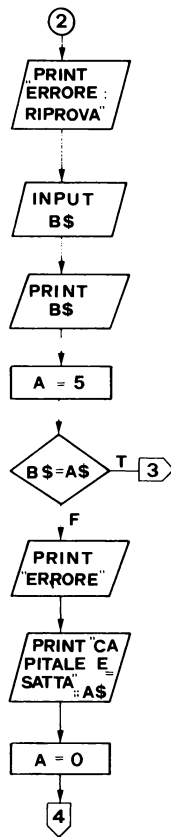
Linee 150-190 : scelta casuale di una domanda fra quelle disponibili e non ancora scelte.

Linee 200-250 : stampa del nome della nazione

Linee 300-450 : gestione della risposta alla domanda

Linee 500-510 : fase di chiusura

Linee 520-710 : subroutine per le 10 domande che verranno poste in ordine casuale.



```

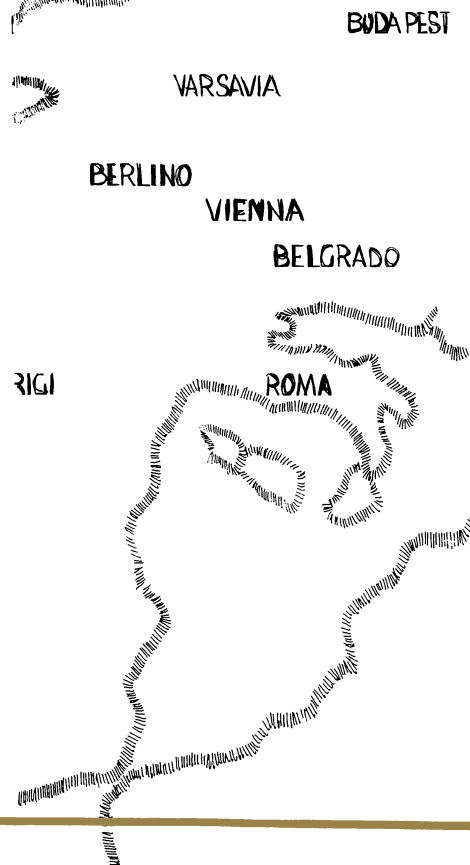
100 DIM Q(10)
110 LET C=0
120 LET Q=0
130 PRINT "**** PROVA CAPITALI ****"
135 PRINT
140 PRINT "SCRIVI LA CAPITALE"
145 PRINT "DELLA SEGUENTE NAZIONE:"
148 PRINT
150 LET x=RND (10)
160 LET A=10
170 IF Q(x)< 0 THEN GO TO 150
180 LET Q(x)=-1
190 GO SUB 500 +20 * x
200 FOR I=1 TO 30
210 LET x=CODE (A$)
220 PRINT CHR$ (x);
230 LET A$=TL$ (A$)
240 IF x=15 THEN GO TO 300
250 NEXT I
300 INPUT B$
305 PRINT B$
310 IF NOT A$=B$ THEN GO TO 400
315 PRINT
316 PRINT "ESATTO"
320 LET C= C+A
330 LET Q=Q+1
340 IF Q=10 THEN GO TO 500
345 PRINT
346 PRINT
350 PRINT "<N/L> PER CONTINUARE"
355 INPUT S$
360 CLS
370 GO TO 130
400 PRINT "ERRORE - RIPROVA:"

```

```

410 INPUT B$
414 PRINT
415 PRINT B$
420 LET A=5
425 IF B$=A$ THEN GO TO 315
426 PRINT
427 PRINT "ERRORE"
430 PRINT "CAPITALE ESATTA: "; A$
440 LET A=0
450 GO TO 320
500 PRINT C; "PER 100 DI RISPOSTE ESATTE"
510 STOP
520 LET A$= "CECOSLOVACCHIA ? PRAGA"
530 RETURN
540 LET A$= "TURCHIA? ISTANBUL"
550 RETURN
560 LET A$= "U.S.A.?WASHINGTON"
570 RETURN
580 LET A$= "OLANDA? AMSTERDAM"
590 RETURN
600 LET A$= "AUSTRALIA? CANBERRA"
610 RETURN
620 LET A$= "GIAPPONE? TOKIO"
630 RETURN
640 LET A$= "INDIA? NUOVA DEHLI"
650 RETURN
660 LET A$= "POLONIA? VARSAVIA"
670 RETURN
680 LET A$= "SVEZIA? STOCCOLMA"
690 RETURN
700 LET A$= "PORTOGALLO? LISBONA"
710 RETURN

```



BUBBLE SORT

Autore: G. Rocca
Programma utilizzante:
4 K di memoria

ANALISI

Questa routine serve per porre in ordine crescente una serie di numeri dati in input.

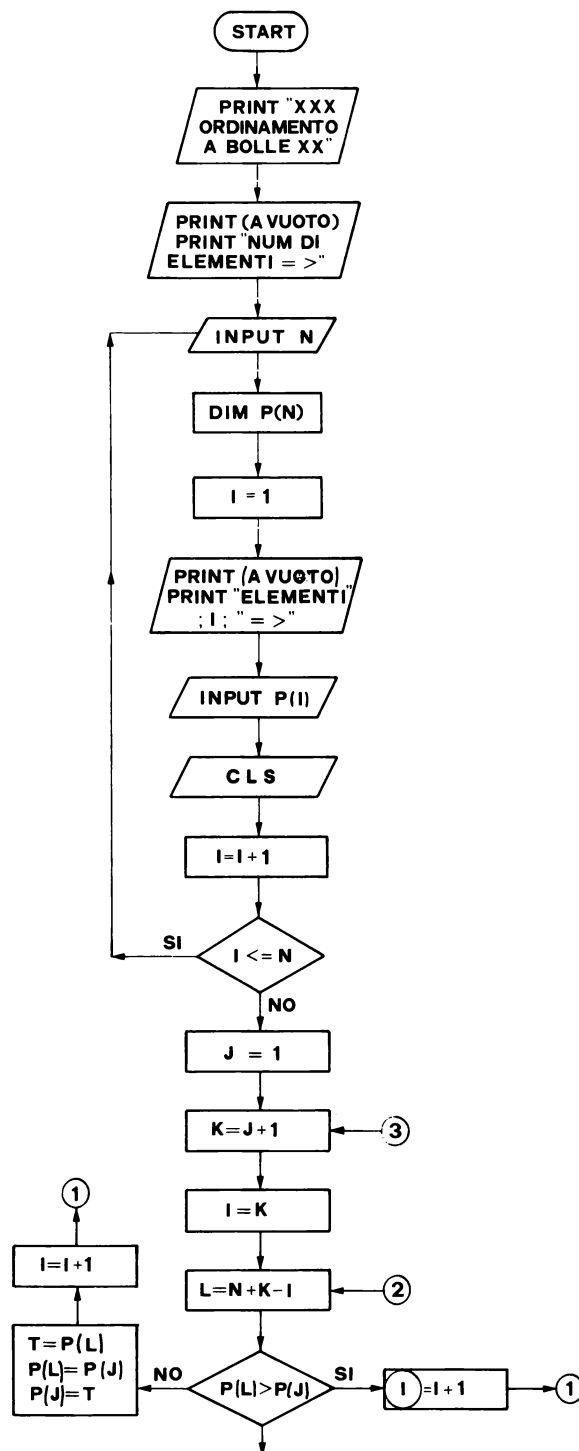
La prima operazione del programma è di caricare una quantità "N" di numeri in un vettore P(I).

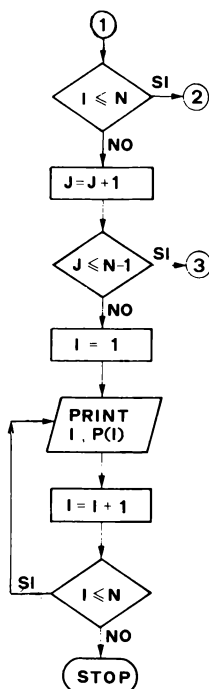
Il vettore P(I) passa poi alla fase di ordinamento confrontando il primo numero della serie con l'ultimo. Se l'ultimo risulta maggiore del primo si passa al confronto con il penultimo, con il terzultimo e così via fino ad arrivare al numero in esame. Altrimenti incremento il puntatore I di 1 e passo al confronto della variabile I con N e seguio. Dopo aver esaminato il primo numero della serie si incrementa il puntatore e si analizza il successivo ripetendo l'operazione come nel numero precedente. L'ordinamento finisce quando il numero in esame puntato da J corrisponde al penultimo della serie da ordinare.

Per controllare e vedere il risultato dell'ordinamento si effettua una routine di stampa dei numeri contenuti nel vettore P(I), con il rispettivo puntatore I.

```

5  PRINT "**** ORDINAMENTO A BOLLE ****"
6  PRINT
10 PRINT "NUM DI ELEMENTI = >"
15 INPUT N
20 DIM P(N)
25 FOR I=1 TO N
26  PRINT
30  PRINT "ELEMENTO"; I; " = >"
35  INPUT P(I)
40  CLS
45  NEXT I
100 FOR J=1 TO N-1
110  LET K = J+1
120  FOR I = K TO N
130  LET L = N+K-I
140  IF P(L) > P(J) THEN GO TO 180
150  LET T = P(L)
160  LET P(L) = P(J)
170  LET P(J) = T
180  NEXT I
190  NEXT J
200 FOR I=1 TO N
210  PRINT I, P(I)
220  NEXT I
    
```





SISTEMA 4 EQUAZIONI IN 4 INCOGNITE

Autore: M.Oliva
Programma utilizzante:
4K di memoria

Si tratta di un programma che consente la risoluzione di un sistema lineare di 4 equazioni in 4 incognite normale e non omogeneo un sistema cioè del tipo:

$$\begin{aligned} a_{11}X + a_{12}Y + a_{13}Z + a_{14}t &= a_{15} \\ a_{21}X + a_{22}Y + a_{23}Z + a_{24}t &= a_{25} \\ a_{31}X + a_{32}Y + a_{33}Z + a_{34}t &= a_{35} \\ a_{41}X + a_{42}Y + a_{43}Z + a_{44}t &= a_{45} \end{aligned}$$

in cui almeno un coefficiente di ciascuna varia ille (x, y, z, t) sia diverso da zero e almeno un termine noto (a₁₅, a₂₅, a₃₅, a₄₅) non sia meno.

Per la risoluzione di un sistema di questo tipo si utilizza la regola di Kramer.

Essa, nel nostro caso, ci dice che un sistema non omogeneo di 4 equazioni lineari in 4 incognite con il determinante dei coefficienti non nullo, ammette una e una sola soluzione costituita dai valori ottenibili dividendo per il determinante dei coefficienti (D) i determinanti Dx, Dy, Dz, Dt che si trovano sostituendo ai coefficienti di ciascuna incognita, che si vuole determinare, i termini noti a₁₅, a₂₅, a₃₅, a₄₅.

Ricordiamo allora che:

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} \text{ e che:}$$

$$D_x = \begin{vmatrix} a_{15} & a_{12} & a_{13} & a_{14} \\ a_{25} & a_{22} & a_{23} & a_{24} \\ a_{35} & a_{32} & a_{33} & a_{34} \\ a_{45} & a_{42} & a_{43} & a_{44} \end{vmatrix};$$

$$D_z = \begin{vmatrix} a_{11} & a_{12} & a_{15} & a_{14} \\ a_{21} & a_{22} & a_{25} & a_{24} \\ a_{31} & a_{32} & a_{35} & a_{34} \\ a_{41} & a_{42} & a_{45} & a_{44} \end{vmatrix};$$

$$D_y = \begin{vmatrix} a_{11} & a_{15} & a_{13} & a_{14} \\ a_{21} & a_{25} & a_{23} & a_{24} \\ a_{31} & a_{35} & a_{33} & a_{34} \\ a_{41} & a_{45} & a_{43} & a_{44} \end{vmatrix}$$

$$D_t = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{45} \end{vmatrix} \text{ e le soluzioni sono:}$$

$$x = \frac{D_x}{D}; \quad y = \frac{D_y}{D}; \quad z = \frac{D_z}{D}; \quad t = \frac{D_t}{D}$$

Il problema si riduce, dunque, al sviluppo di un determinante del 4° ordine.

Si noti che gli elementi del determinante sono 4² = 16 e che i termini dello sviluppo saranno 4! = 4·3·2·1 = 24. Lo sviluppo A del determinante sarà allora dato da

$$A = \sum_{q=1}^{24} (-1)^s a_{q11} \cdot a_{q22} \cdot a_{q33} \cdot a_{q44}$$

dove q è una delle 24 permutazioni degli indici (1,2,3,4) ed s è il numero di spostamenti della permutazione (lo spostamento è il valore che indica che un indice maggiore precede uno minore).

In realtà il calcolo è assai più semplice di quanto non possa ora simulare. Vediamo ora come ciò si è prodotto nel programma.

SISTEMA 4 EQUAZIONI IN 4 INCOGNITE

Esso incomincia visualizzando la spiegazione iniziale e dopo l'introduzione del carattere <NEWLINE> ha inizio il cambiamento delle operazioni componenti il sistema che si vuole risolvere.

Sono quindi richiesti i coefficienti di x y z t per ciascuna equazione, esse questi sono tutti nulli si ha la segnalazione d'errore e la richiesta di nuovi valori, poi viene richiesto il termine noto. Successivamente il programma controlla che almeno uno dei coefficienti di ciascuna variabile e del termine noto non sia uguale a zero, altrimenti visualizza un messaggio di errore e incomincia di nuovo il caricamento.

Se i coefficienti ed i termini noti sono posti correttamente il programma calcola i valori delle variabili secondo il procedimento già esposto.

La risoluzione del sistema è segnalata dalla visualizzazione del sistema e dai valori calcolati per ciascuna variabile. A tal proposito occorre ricordare che queste soluzioni possono presentarsi in due forme: in forma intera e in forma di frazione. Infatti lo ZX-80 non trattando numeri decimali non consentirebbe l'esecuzione corretta di divisioni nelle quali il dividendo non fosse multiplo del divisore. In questo caso il programma visualizza il risultato sotto forma di frazione senza però che questa sia ridotta per forza ai minimi termini.

Consideriamo ora i due esempi chiarificatori.

Il sistema:

$$\begin{aligned} x - 2y + 5z - 2t &= -14 \\ 2x + y - 3z + t &= 10 \\ x + 3y - z + 2t &= 14 \\ 3x - y + 2z + 3t &= 8 \end{aligned}$$

le cui soluzioni sono:

$$\begin{aligned}x &= 1 \\y &= 2 \\z &= -1 \\t &= 3\end{aligned}$$

introdotta nel programma produce la seguente visualizzazione finale:

*** SOLUZIONE SISTEMA EQUAZ. ***

IL SISTEMA:

$$\begin{aligned} +x - 2y + 5z - 2t &= -14 \\ +2x + y - 3z + t &= 10 \\ +x + 3y - z + 2t &= 14 \\ +3x - y + 2z + 3t &= 8 \end{aligned}$$

HA LE SOLUZIONI:

$x = 1$
 $y = 2$
 $z = -1$
 $t = 3$
 9/550

Invece il sistema:

$$\begin{aligned} 4x - 3y + 2z - 7t &= -2 \\ x + 6y - z &= 2 \end{aligned}$$

$$\begin{aligned} x + 2z + 14t &= -13 \\ 3x - 3y + z - 21t &= 2 \end{aligned}$$

le cui soluzioni sono:

$$\begin{aligned}x &= 3 \\y &= 5/3 \\z &= -9 \\t &= 1/7\end{aligned}$$

produce la seguente visualizzazione finale.

*** SOLUZIONE SISTEMA EQUAZ. ***

IL SISTEMA:

$$\begin{aligned} + 4x - 3y + 2z - 7t &= -2 \\ + x + 6y - z &= 2 \\ + x + 2z + 14t &= -13 \\ + 3x - 3y + z - 21t &= 2 \end{aligned}$$

HA LE SOLUZIONI:

$$\begin{aligned}x &= 3 \\y &= -(700/420) \\z &= -9 \\T &= -60/420 \\9/550\end{aligned}$$

Infatti:

$$y = -700/420 = -5/3 \quad e$$

Come si è detto il programma utilizza il metodo delle permutazioni degli elementi nel calcolo del valore dei determinanti.

Tali permutazioni in forma di indici, relativi ai coefficienti e ai termini noti delle 4 equazioni, devono essere introdotte nei byte occupati dalla codifica della linea:

```
1 REM ████████████████████████████████████████████████████████████
```

Per potere, dunque caricare il programma correttamente occorre operare nel modo seguente. Per prima cosa scrivere in memoria tutto il programma così come appare nel listato; fatto ciò eseguire in modo immediato l'istruzione

GO TO 5000

Dove alla linea 5000 ha inizio la routine di caricamento delle costanti rappresentanti le permutazioni. Esse sono 24, e vengono introdotte a partire dal byte il cui indirizzo è 16428

Le costanti che devono essere introdotte sono:

| INDIR. BYTE | COSTANTE |
|-------------|----------|
| 16428 | 134 |
| 16429 | 143 |
| 16430 | 130 |
| 16431 | 220 |
| 16432 | 242 |
| 16433 | 224 |
| 16434 | 214 |
| 16435 | 34 |
| 16436 | 43 |
| 16437 | 30 |
| 16438 | 210 |
| 16439 | 241 |
| 16440 | 24 |
| 16441 | 42 |
| 16442 | 141 |
| 16443 | 114 |
| 16444 | 20 |
| 16445 | 110 |
| 16446 | 221 |
| 16447 | 212 |

| | |
|-------|-----|
| 16448 | 113 |
| 16449 | 131 |
| 16450 | 32 |
| 16451 | 23 |

Una volta concluso il caricamento delle costanti la linea 1 non è più visualizzabile: per cui *non dare mai* il comando

LIST

Se si vuole visualizzare il programma dall'inizio (linea 1 esclusa ovviamente) si può utilizzare il comando:

LIST 4

Se poi si vuole memorizzare su cassetta il programma possono essere cancellate le linee 5000-5140, mentre diversamente è consigliabile conservare la routine di caricamento delle costanti così da potere ricaricare i valori in caso di occidentale pressione del tasto LIST. Avendo invece memorizzato su cassetta il programma è possibile ricaricarlo con le costanti corrette in casi analoghi.

DESCRIZIONE GENERALE DEL PROGRAMMA

- | | |
|-----------|--|
| 1 | Linea per l'immagazzinamento delle costanti relative alle permutazioni degli elementi dei determinanti. |
| 4-40 | Visualizzazione della spiegazione iniziale e di inizializzazione variabili |
| 42-98 | Caricamento dei coefficienti e dei termini noti delle equazioni con relativo controllo. |
| 100-112 | Controllo normalità e non omogeneità del sistema introdotto |
| 114-320 | Calcolo del determinante fondamentale D e dei determinati Dx, Dy, Dz, Dt |
| 330-450 | Visualizzazione del sistema introdotto |
| 460-550 | Visualizzazione delle soluzioni e fine del programma. |
| 1000-1050 | Subroutine che visualizza l'intestazione del programma. |
| 1500-1550 | Subroutine che visualizza un messaggio di errore. |
| 2000-2070 | Subroutine che legge il contenuto di una locazione determinata facente parte della codifica della linea 1, e traduzione in indici di permutazione. |
| 2500-2570 | Subroutine che visualizza il coefficiente e la variabile ed il suo segno nella visualizzazione delle equazioni componenti il sistema. |
| 3000-3110 | Subroutine che visualizza per ciascuna variabile la soluzione o in forma intera o in forma frazionaria. |
| 5000-5140 | Routine, da eseguire dopo la scrittura dell'intero programma, che consente di caricare nei byte costituenti la codifica della linea 1, a partire da quello di indirizzo 16428, le costanti relative alle permutazioni degli elementi dei determinanti. |

Principali variabili utilizzate:

V (indirizzo 1° byte costanti di permutazione (16428))

A(20) vettore contenente i coefficienti ed i termini noti delle equazioni componenti il sistema

B(25) vettore calcolo dei determinanti

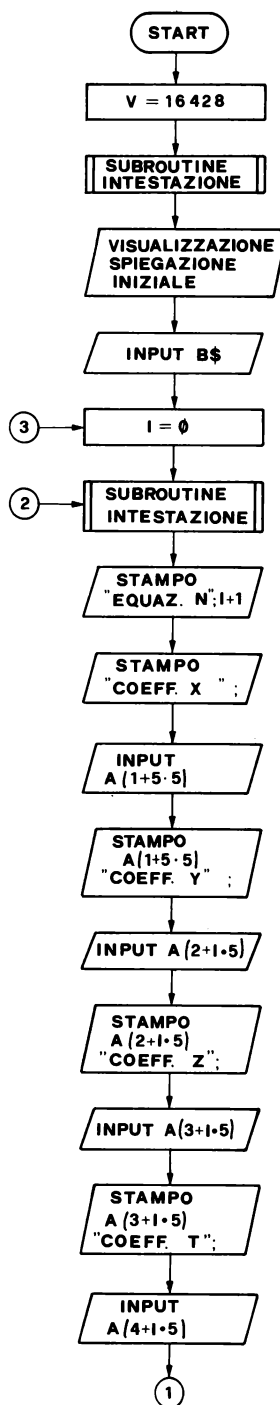
C(3) indici di permutazione

Q(5) valori rispettivamente di D, Dx, Dy, Dz, e Dt

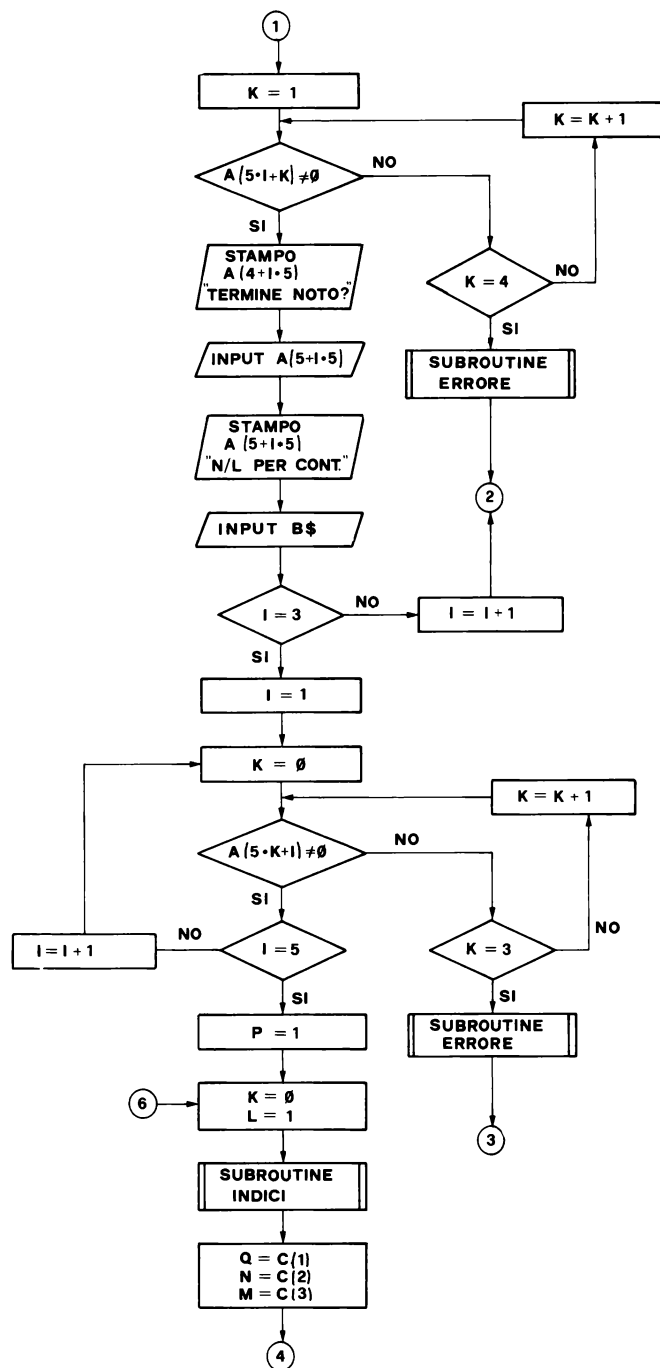
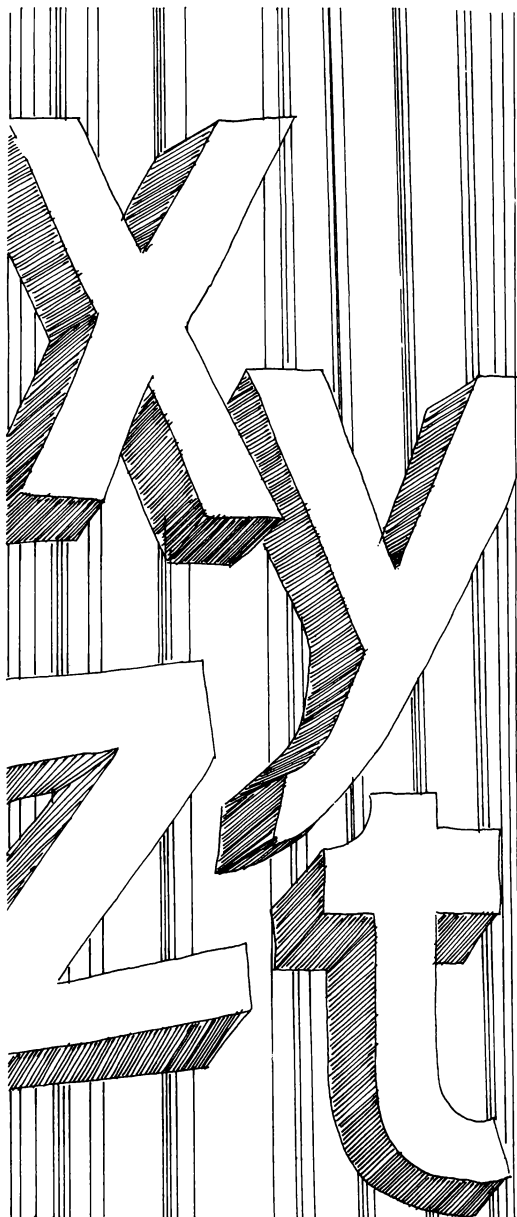
L,M,N,O,F,G,H indici calcolati dalle permutazioni

I,K,P,E,W indici dei cicli

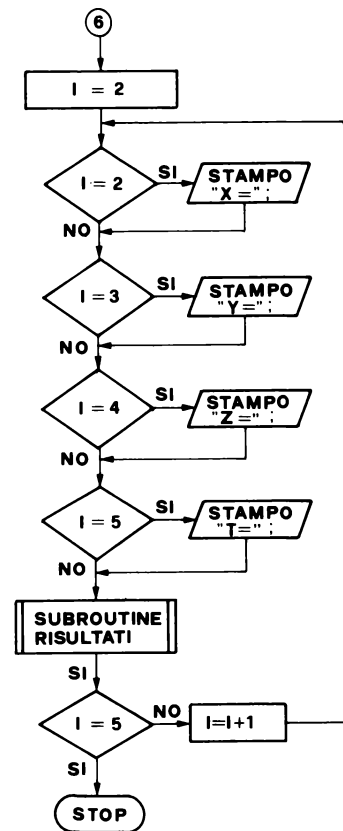
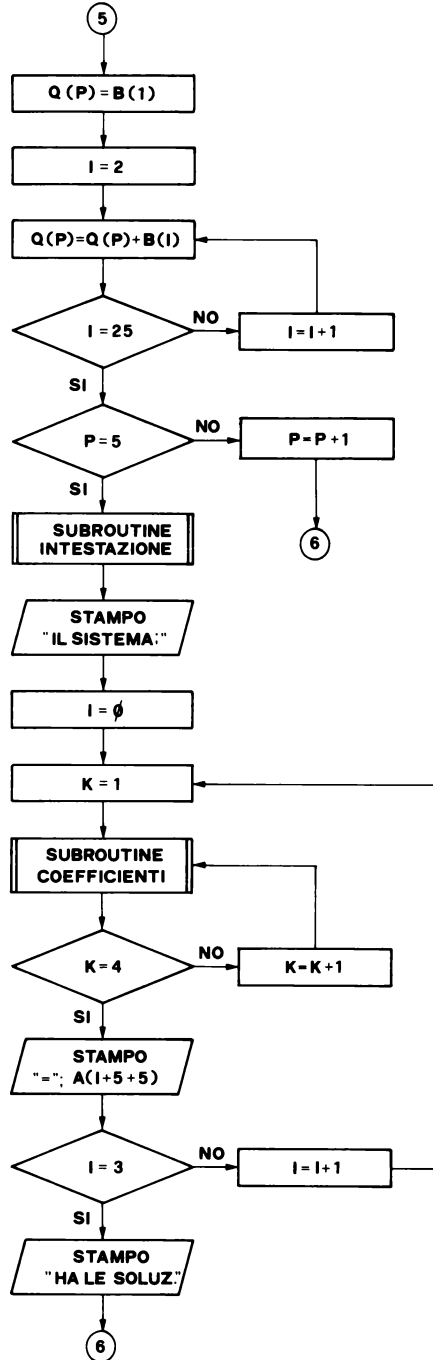
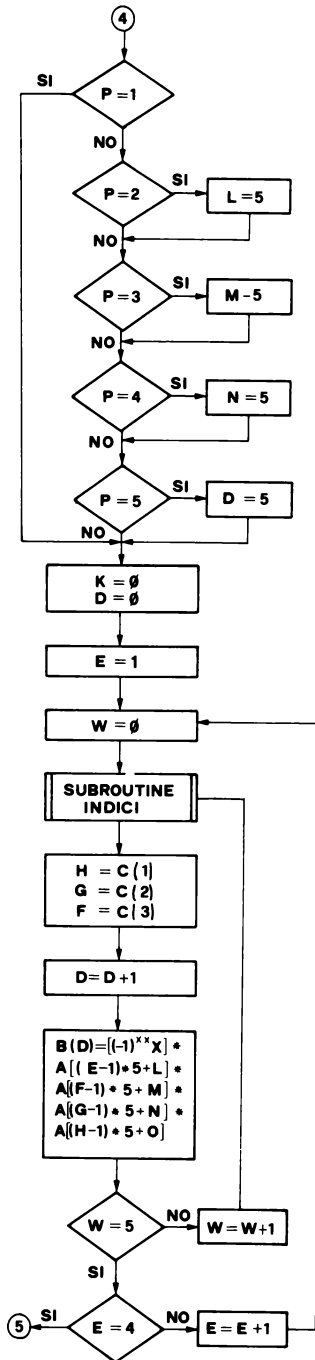
SISTEMA 4 EQUAZIONI IN 4 INCOGNITE



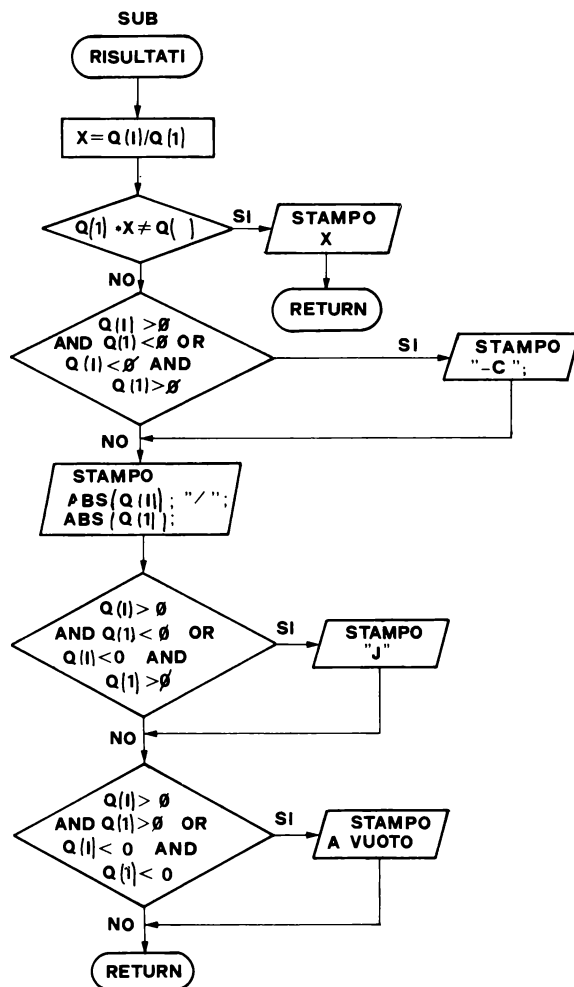
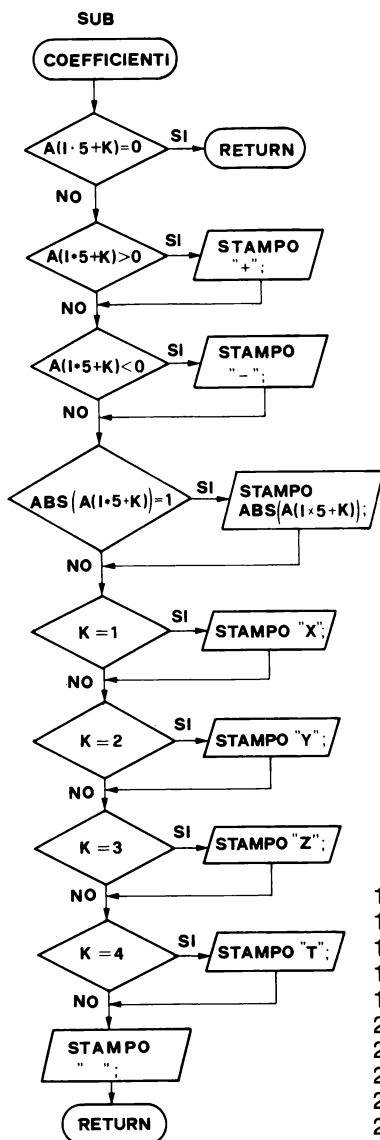
SISTEMA 4 EQUAZIONI IN 4 INCOGNITE



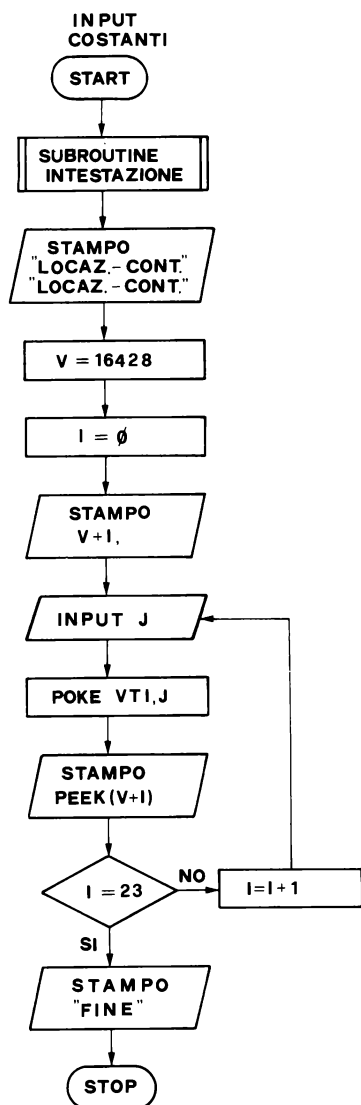
SISTEMA 4 EQUAZIONI IN 4 INCOGNITE



48



48



```

62 PRINT A(2 + I*5)
64 PRINT "COEFF. Z?";
66 INPUT A(3 + I*5)
68 PRINT A(3 + I*5)
70 PRINT "COEFF. T?";
72 INPUT A(4 + I*5)
74 FOR K = 1 TO 4
76 IF NOT A(5*I + K) = 0 THEN GO TO 84
78 NEXT K
80 GO SUB 1500
82 GO TO 44
84 PRINT A(4 + I*5)
86 PRINT "TERMINE NOTO?";
88 INPUT A(5 + I*5)
90 PRINT A(5 + I*5)
92 PRINT
  
```

```

94 PRINT "<N/L> PER CONTINUARE"
96 INPUT B$
98 NEXT I
100 FOR I = 1 TO 5
102 K = 0 TO 3
104 IF NOT A(I + K*5) = 0 THEN GO TO 112
106 NEXT K
108 GO SUB 1500
110 GO TO 42
112 NEXT I
114 FOR P = 1 TO 5
116 LET K = 0
118 LET L = 1
120 GO SUB 2000
122 LET O = C(1)
124 LET N = C(2)
126 LET M = C(3)
128 IF P = 1 THEN GO TO 180
130 IF P = 2 THEN LET L = 5
140 IF P = 3 THEN LET M = 5
150 IF P = 4 THEN LET N = 5
170 IF P = 5 THEN LET O = 5
180 LET K = 0
190 LET D = 0
200 FOR E = 1 TO 4
210 FOR W = 0 TO 5
212 GO SUB 2000
214 LET H = C(1)
216 LET G = C(2)
218 LET F = C(3)
230 LET D = D + 1
240 LET B(D) = ((-1)**W)*A((E-1)*5 + L)*A((F
- 1)*5 + M)*A((G-1)*5 + M)*A((H-1)*5 + O)
250 NEXT W
260 NEXT E
270 LET Q(P) = B(1)
280 FOR I = 2 TO 25
290 LET Q(P) = Q(P) + B(I)
310 NEXT I
320 NEXT P
330 GO SUB 1000
340 PRINT "IL SISTEMA:"
350 PRINT
370 FOR I = 0 TO 3
390 FOR K = 1 TO 4
400 GO SUB 2500
410 NEXT K
420 PRINT "=Δ"; A(I*5+5)
430 NEXT I
450 PRINT
460 PRINT "HA LE SOLUZIONI:"
470 PRINT
480 FOR I = 2 TO 5
490 IF I = 2 THEN PRINT "x = Δ";
500 IF I = 3 THEN PRINT "y = Δ";
510 IF I = 4 THEN PRINT "z = Δ";
520 IF I = 5 THEN PRINT "t = Δ";
530 GO SUB 3000
540 NEXT I
550 STOP

1000 CLS
1020 PRINT "**** SOLUZIONE SISTEMA EQUAZ. ****"
1040 PRINT
1045 PRINT
1050 RETURN
  
```


SISTEMA 4 EQUAZIONI IN 4 INCOGNITE

```

1500 PRINT
1505 PRINT
1510 PRINT "**** ERRORE INPUT ****"
1520 PRINT
1535 PRINT "<N/L> PER CONTINUARE"
1540 INPUT B$
1550 RETURN

```

```

2000 LET U = PEEK (U + K) + 100
2010 IF (U/10)* 10=U THEN LET U=U/10 +400
2020 FOR I=1 TO 3
2030 LET C(I)= U-(U/10)*10
2040 LET U = U/10
2050 NEXT I
2060 LET K=K + 1
2070 RETURN

```

```

2500 IF A(I*5+K)=0 THEN RETURN
2520 IF A(I*5+K)>0 THEN PRINT "+Δ";
2522 IF A(I*5+K)<0 THEN PRINT "-Δ";
2524 IF NOT ABS(A(I*5+K))=1 THEN PRINT ABS
(A(I*5+K));
5230 IF K=1 THEN PRINT "x";
2540 IF K=2 THEN PRINT "y";
2550 IF K=3 THEN PRINT "z";
2560 IF K=4 THEN PRINT "t";
2565 PRINT "Δ";
2570 RETURN

```

```

3000 LET X=Q(I)/Q(1)
3010 IF NOT Q(1)* X=Q(I) THEN GO TO 3040
3020 PRINT X
3030 RETURN
3040 IF Q(I)>0 AND Q(1)<0 OR Q(I)<0 AND Q(1)>
0 THEN PRINT "- (";
3080 PRINT ABS (Q(I)); "/"; ABS (Q(1));
3090 IF Q(I)>0 AND Q(1)<0 OR Q(I)<0 AND Q(1)>
0 THEN PRINT ")";
3100 IF Q(I)>0 AND Q(1)>0 OR Q(I)<0 AND Q(1)<
0 THEN PRINT
3110 RETURN

```

```

5000 GO SUB 1000
5010 PRINT "INIZIO INPUT COSTANTI"
5020 PRINT
5030 PRINT "LOCAZ. ΔΔ CONT. ΔΔΔ
LOCAZ. ΔΔ CONT."
5040 PRINT
5050 LET V=16428
5060 FOR I=0 TO 23
5070 PRINT V+I,
5080 INPUT J
5090 POKE V+I, J
5100 PRINT PEEK (V + I),
5110 NEXT I
5120 PRINT
5125 PRINT
5130 PRINT "FINE INPUT COSTANTI"
5140 STOP

```

IL LABIRINTO

Autore: R. Rozzi
Programma utilizzante:
4 K di memoria

Questo programma genera e visualizza un labirinto. Il giocatore dovrà percorrere il labirinto cercando di raggiungere l'uscita, non è detto che ogni labirinto di sponga di una via d'uscita, ma se essa esiste non è difficile trovarla.

Il labirinto viene creato e visualizzato riga per riga utilizzando il vettore w di 32 posizioni, ogni riga è ottenuta modificando le precedenti, eseguendo più volte il programma avremo labirinti sempre differenti.

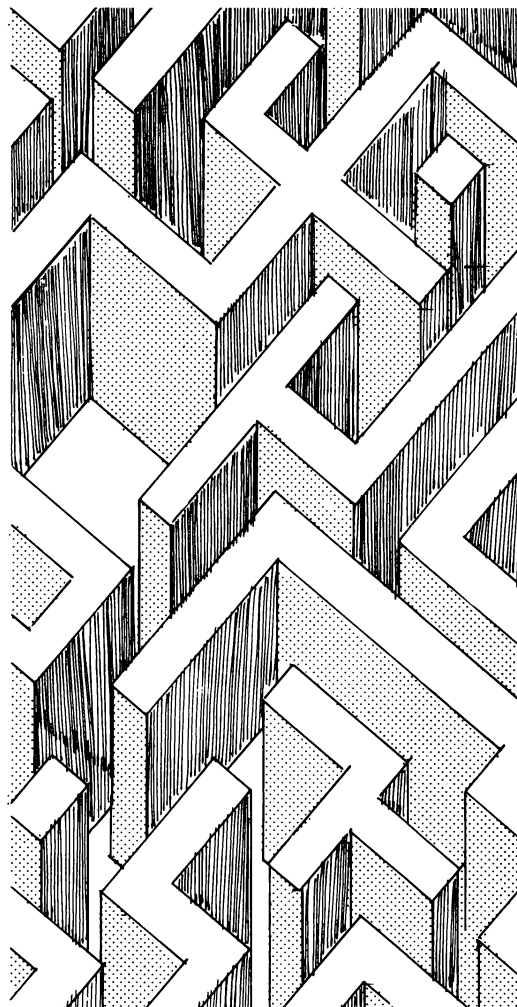
Descrizione programma

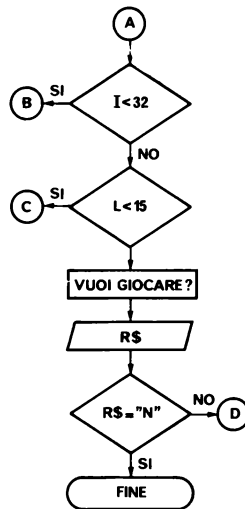
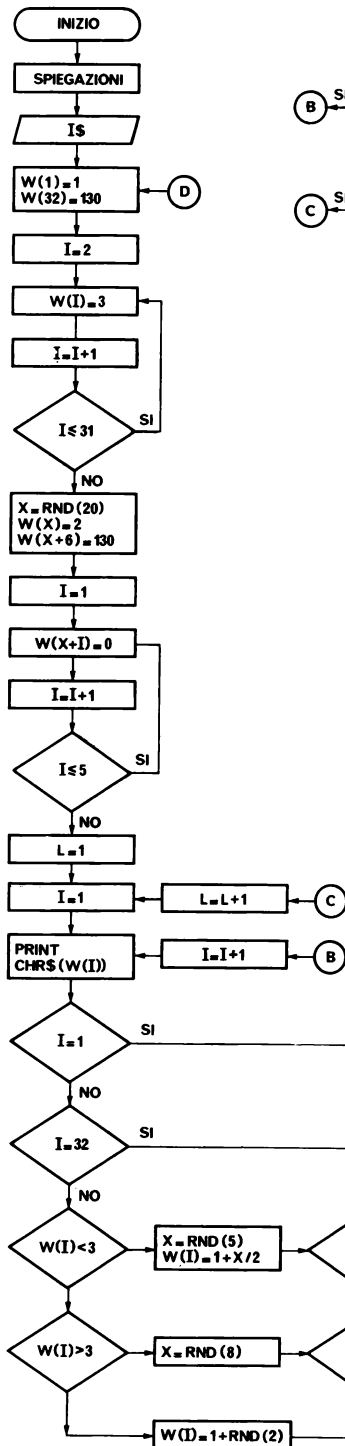
| | |
|-----------|----------------------------|
| 5 - 130 | SPIEGAZIONI |
| 140 - 240 | CREAZIONE PRIMA LINEA |
| 340 - 470 | CREAZIONE LINEE SUCCESSIVE |

```

5 PRINT
10 PRINT
11 PRINT

```





```

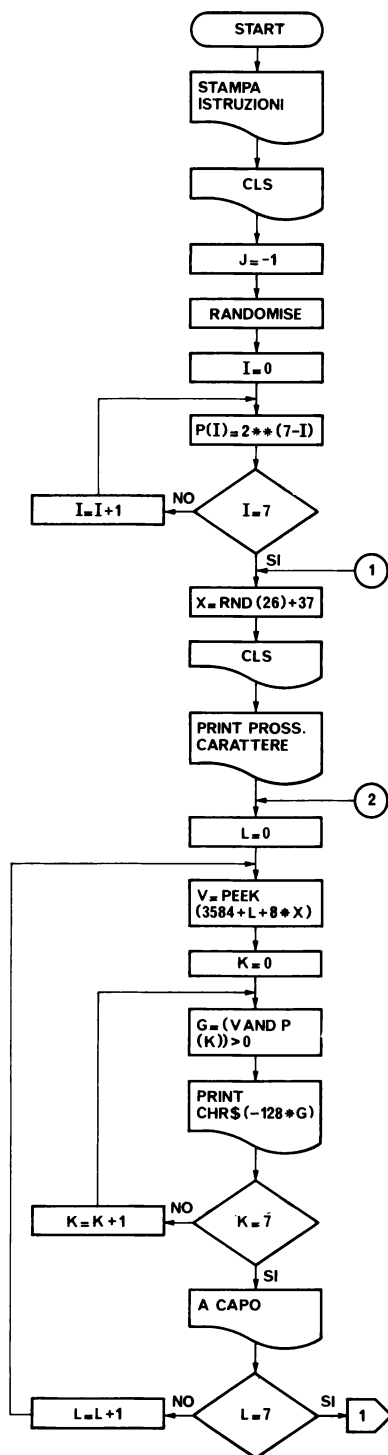
15 PRINT "IL PROGRAMMA COSTRUISCE DEI"
20 PRINT "LABIRINTI DA CUI DEVI CERCARE"
25 PRINT "DI USCIRE. LA ENTRATA E'"
26 PRINT "POSTA NELLA PARTE SUPERIORE"
27 PRINT "DELLO SCHERMO."
28 PRINT
30 PRINT "QUANDO SEI PRONTO DAI N/L"
35 INPUT I$
40 CLS
100 DIM W(32)
110 PRINT
120 PRINT "PROVA A USCIRE DA QUI!"
122 PRINT "BUON DIVERTIMENTO ..."
130 PRINT
140 LET W(1)=2
150 LET W(32)=130
160 FOR I=2 TO 31
170 LET W(I)=3
180 NEXT I
190 LET X=RND (20)
200 LET W(X)=2
210 LET W(X+6) =130
220 FOR I=2 TO 5
230 LET W(X+I)=0
240 NEXT I
300 FOR L=1 TO 15
310 FOR I=1 TO 32
320 PRINT CHR$( W(I));
330 IF I=1 OR I=32 THEN GOTO 500
340 IF W(I) <3 THEN GOTO 450
350 IF W(I) >3 THEN GOTO 400
360 LET X= RND(8)
370 IF X <3 THEN LET W(I)=2
380 IF X=8 THEN LET W(I)=7
390 GOTO 500
400 LET W(I)=1+RND (2)
410 GOTO 500
450 LET X=RND (5)
460 LET W(I)= 1+X/2
470 IF X=1 THEN LET W(I)=7
500 NEXT I
510 NEXT L
520 PRINT
525 PRINT
530 PRINT "VUOI GIOCARE ANCORA (S/N)?"
540 INPUT R$
550 IF R$ = "N" THEN STOP
560 CLS
570 GOTO 100
  
```

HAI MEMORIA?

Autore: P. Ciceri
Programma utilizzante:
4 K di memoria

Il programma permette di provare le capacità della propria memoria. Viene generata casualmente una sequenza di lettere che il giocatore dovrà cercare di tenere a mente e riscrivere. Ad ogni tentativo esatto ZX-80 aggiungerà un'altra lettera alla sequenza.

La parte più interessante del programma è l'uso dei caratteri 8x8 da parte del calcolatore per la rappresentazione della nuova lettera della sequenza, non ci addentriamo comunque nei particolari in quanto esiste un programma (Caratteri 8x8) che tratta specificatamente l'argomento.



Variabili usate

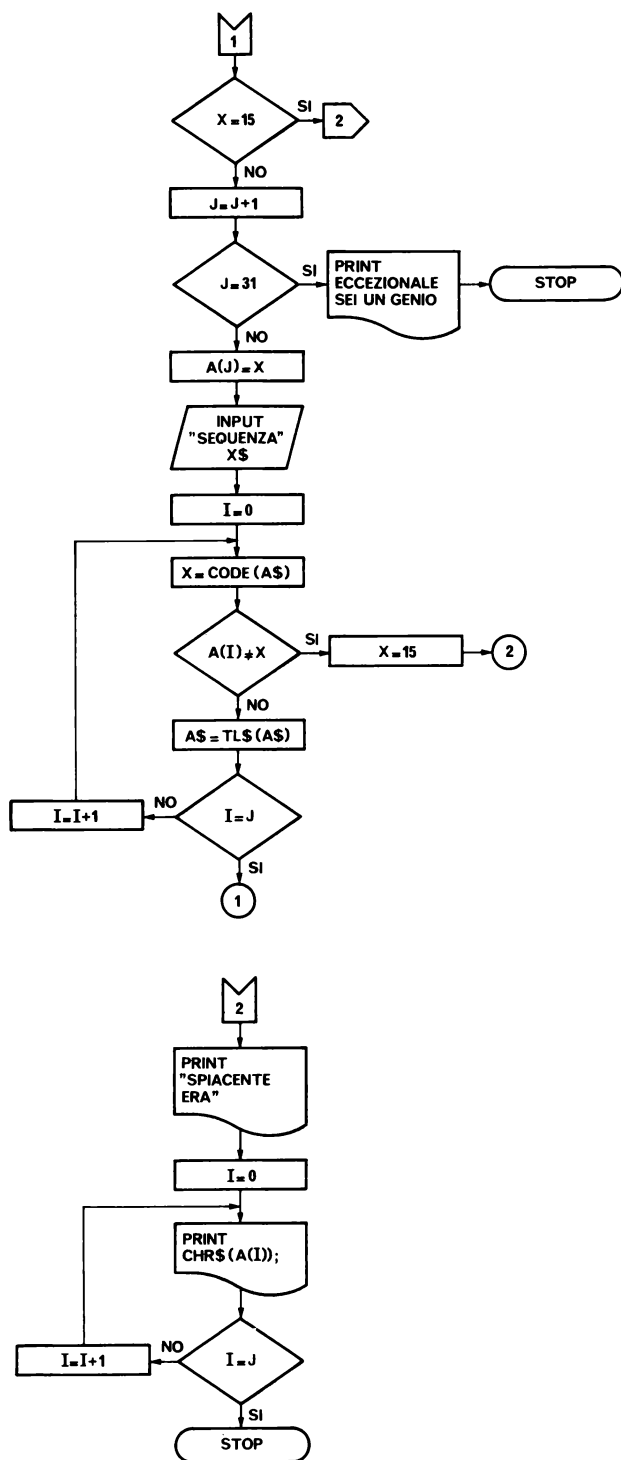
P(7) Vettore per la stampa dei caratteri 8x8
A(30) Vettore contenente i codici demme lettere della sequenza
X Codice della nuova lettera della sequenza
G,V Variabili per la stampa 8x8
J Contatore caratteri della sequenza
K,L,I Contatori
V\$ Variabile di comodo

LISTING - HAI MEMORIA? -

```

10 PRINT " AGGIUNGERE 2 BLANKS
*** HAI MEMORIA? ***"
15 PRINT
20 PRINT
25 PRINT "IL GIOCO CONSISTE NEL RICOR-
DARE"
30 PRINT "UNA SEQUENZA DI CARATTERI CHE"
35 PRINT "ZX-80 TI VISUALIZZA"
40 PRINT "AD OGNI TENTATIVO IL GIOCO"
45 PRINT "DIVENTA PIU' DIFFICILE IN QUANTO"
50 PRINT "VIENE AGGIUNTO UN ALTRO CA-
RATTERE"
55 PRINT
60 PRINT "QUANDO SEI PRONTO DAI N/L"
65 INPUT V$
70 CLS
100 DIM A(30)
110 LET J = -1
120 RANDOMISE
130 DIM P(7)
140 FOR I=0 TO 7
150 LET P(I)=2 ** (7-I)
160 NEXT I
200 LET x= RND(26) + 37
210 CLS
220 PRINT "PROSS.CARR."
230 FOR L=0 TO 7
240 LET V= PEEK (3584+L+8*X)
250 FOR K=0 TO 7
260 LET G= (V AND P(K))> 0
270 PRINT CHR$ (-128 * G);
280 NEXT K
290 PRINT
300 NEXT L
400 IF x=15 THEN GO TO 700
410 LET J=J+1
420 IF J=31 THEN GO TO 800
430 LET A(J) = x
440 PRINT "SCRIVI LA SEQUENZA"
450 INPUT A$
460 PRINT A$
470 FOR I=0 TO J
480 LET X= CODE (A$)
490 IF NOT A(I)=x THEN GO TO 600
500 LET A$ = TL$ (A$)
510 NEXT I
520 GO TO 200
600 LET x=15
610 GO TO 230
700 PRINT "SPIACENTE ERA:"
710 FOR I=0 TO J
720 PRINT CHR$ (A(I));
730 NEXT I
740 STOP
800 PRINT "ECCEZIONALE SE UN GENIO"

```



CARATTERI 8x8

Autore: P. Ciceri
Programma utilizzante:
4 K di memoria

Questo programma illustra una possibilità del Sinclair, cioè la possibilità di visualizzare caratteri in un formato otto volte superiore al normale per mezzo di una routine.

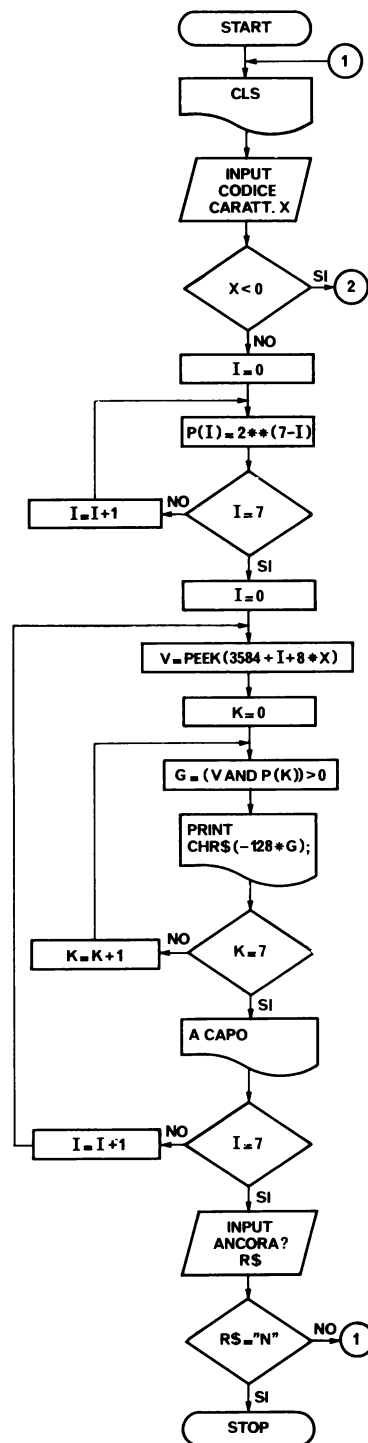
La routine di visualizzazione è costituita dalle linee comprese tra la 40 e la 160, linee che potranno essere inserite in un qualsiasi programma come routine qualora si verifichi la necessità di visualizzare caratteri in formato 8x8 (vedi Hai memoria?).

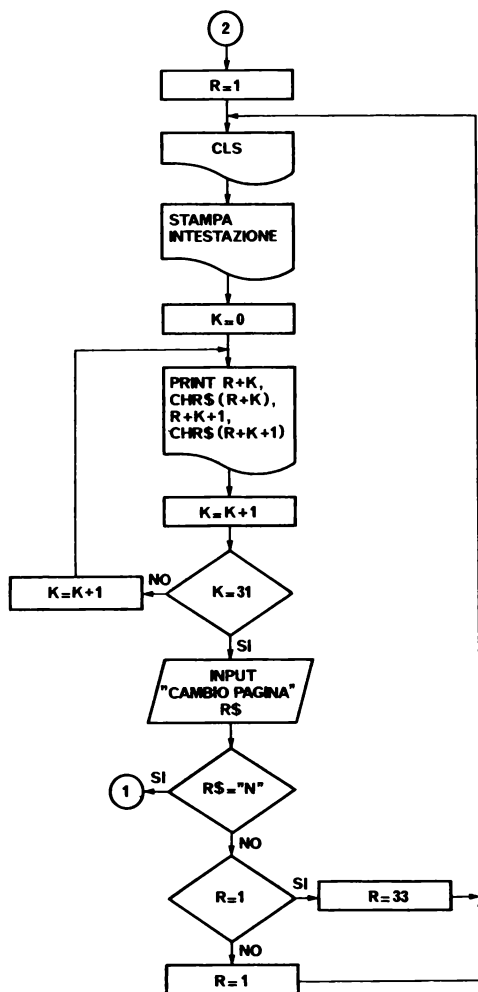
È inoltre possibile con questo programma avere una tabella dei codici ASCII inserendo un numero minore di zero alla richiesta di codice carattere.

```

10 CLS
15 PRINT "    *** CARATTERI 8x8 ***"
17 PRINT
20 DIM P(7)
25 PRINT "PER LA TAB. CODICI DARE UN COD.
<0"
26 PRINT
30 PRINT "COD. CARATTERE = (>0)"
40 INPUT x
45 IF X < 0 THEN GO TO 500
50 FOR I=0 TO 7
60 LET P(I) = 2 ** (7-I)
70 NEXT I
90 FOR I=0 TO 7
100 LET V= PEEK (3584+I+8 * X)
110 FOR K=0 TO 7
120 LET G= (V AND P(K)) > 0
130 PRINT CHR$ (-128*G);
140 NEXT K
150 PRINT
160 NEXT I
170 PRINT "ANCORA (S/N)?"
180 INPUT R$
190 IF R$ = "N" THEN STOP
200 GO TO 10
500 LET R=1
505 CLS
510 PRINT "    TABELLA CODICI"
520 PRINT
535 PRINT "-COD -- CAR ----COD --CAR --"
530 FOR K=0 TO 31
540 PRINT "ΔΔ"; R+K, CHR$ (R+K); "ΔΔΔ";
R+H+1, "ΔΔΔ"; CHR$ (R+K+1)
545 LET K=K+1
550 NEXT K
560 PRINT "CAMBIO PAGINA = >"
570 INPUT R$
580 IF R$ = "N" THEN GO TO 620
590 IF R=1 THEN GO TO 605
592 LET R=1
600 GO TO 505
605 LET R=33
610 GO TO 505
620 CLS
630 GO TO 15

```





CORSA DI CAVALLI

Autore: S. Papes
Programma utilizzante:
4 K di memoria

Questo programma riproduce una corsa di cavalli. Il numero dei cavalli concorrenti è di quattro (A,B,C,D). La gara si svolge in un circuito con lo sviluppo di giri (Z) desiderato dal giocatore. Ad ogni cavallo corrisponde un pronostico (contenuto nel vettore O di indice I) creato casualmente che naturalmente può incidere sul corso della gara. Il vettore D di indice I (i valori da 1 a 4) contiene le posizioni intermedie e finali dei rispettivi cavalli.

In D(0) è memorizzata la posizione del cavallo in testa mentre J indica quale cavallo conduce la gara.

In questo gioco può partecipare un solo giocatore per ogni corsa che dispone inizialmente di una somma pari a L.100. Ad ogni corsa il giocatore scommette una somma (B) che naturalmente deve essere non maggiore delle proprie disponibilità (M). Il cavallo su cui si è puntato è indicato da H\$.

La prima parte del programma (fino all'istruzione 300) escludendo il sottoprogramma (100 - 140) è una fase di inizializzazione della corsa e consiste nell'azzeramento dei vettori nonché nella richiesta della somma puntata e del cavallo scommesso. Il sottoprogramma sopra indicato è una routine che permette la visualizzazione della situazione della gara alla conclusione di ogni giro.

La gara vera e propria si svolge nella parte successiva, all'interno del ciclo che inizia con l'istruzione 340, mentre nell'ultima fase avviene l'aggiornamento della disponibilità del giocatore.




```

40 PRINT "**** CORSA DI CAVALLI ****"
45 PRINT
50 RANDOMIZE
60 DIMO(4)
70 DIM D(4)
80 LET M = 100
90 GO TO 180
100 PRINT I,
110 FOR C = 1 TO D
120 PRINT " ";
130 NEXT C
140 RETURN
150 FOR I = 0 TO 4
160 LET D(I) = 0
170 NEXT I
180 PRINT "HAI UNA DISPONIBILITA' DI L.";M
190 PRINT
200 PRINT "I PRONOSTICI SONO"
210 PRINT
215 FOR I = 1 TO 4
220 LET O(I) = RND (9)+1
230 PRINT CHR$ (I+37), O(I);"/1"
235 PRINT
240 NEXT I
250 PRINT "QUANTO SCOMMETTI E PER QUALE
CAVALLO"
260 INPUT B
265 INPUT H$
270 IF NOT B = 0 THEN GO TO 320
280 IF Y = 1 THEN GO TO 600
290 LET Y = 1
300 PRINT "SPUTA LA LIRA, ALTRIMENTI NON
GIOCHI"
310 GO TO 250
320 LET y=0
330 PRINT "SU QUANTI GIRI DEVE SVOLGERSI
LA GARA?"
335 INPUT z
340 FOR L=1 TO z
350 CLS
360 PRINT "GIRO"; L
370 FOR I=1 TO 4
380 LET D(I)=D(I)+5+7/O(I)+RND(2*O(I)/3)
390 IF (D(I) > D(O)) THEN LET D(O)=D(I)
400 IF D(O) = D(I) THEN LET J = I
410 NEXT I
420 FOR I = 1 TO 4
430 PRINT CHR$ (I+37)
440 LET D = D(I) - D(O) + 15
450 GO SUB 100
460 PRINT
470 NEXT I
473 PRINT
474 PRINT "QUANDO HAI VISTO LA SITUAZIONE"
475 PRINT "PREMI IL TASTO NEW LINE"
476 INPUT R$
480 NEXT L
500 PRINT "IL VINCITORE È"; CHR$ (J + 37)
510 PRINT
515 LET S = 0
520 IF CODE (H$) = (J+37) THEN LET S=1
530 LET M=M - B + S * B * O(J)
540 IF M > 0 THEN GO TO 570
550 PRINT "HAI FINITO LA SOMMA"
560 GO TO 600
570 PRINT "PREMI IL TASTO N SE VUOI SMET-
TERE"
580 INPUT D$
590 IF NOT D$ = "N" THEN GO TO 150
600 STOP

```

CARICATORE ASSEMBLER

Autore: P. Ciceri
Programma utilizzante:
4 K di memoria

Il programma permette di caricare programmi in linguaggio macchina, farli eseguire e controllare aree contigue di memoria sia RAM che ROM. Esso si divide in una parte principale e da tre routine che adempiano ciascuna ad una funzione particolare.

Parte principale

Dopo avere stampato una lista di opzioni permette all'utente di selezionarne una particolare. Se viene scritto END si esce dall'esecuzione.

Routine WRT:

Questa routine permette di scrivere nella memoria utente, possono essere scritti programmi in linguaggio macchina, dati ecc. (si richiama il programma Routine assembler per una più completa trattazione sull'uso del



linguaggio macchina). La routine WRT comprende tre opzioni F per uscire, I per scrivere l'indirizzo di partenza ed S per la scrittura effettiva. Per uscire durante la scrittura dare un numero minore di zero. Viene segnalato errore se si cerca di scrivere senza avere prima dato l'indirizzo di partenza.

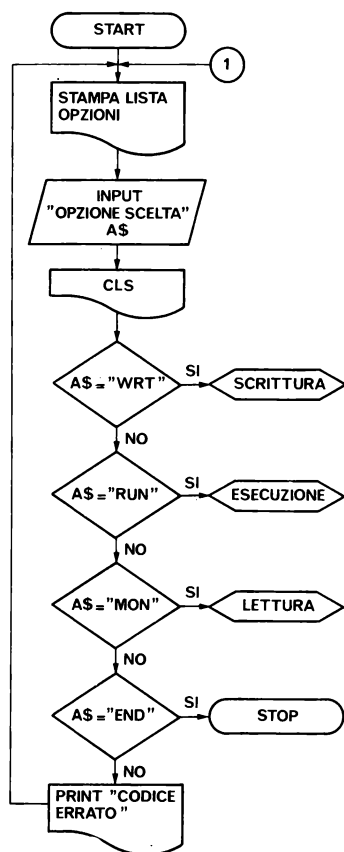
Routine RUN:

Permette di eseguire programmi in linguaggio macchina partenti da un indirizzo dato in input. Per uscire dalla routine dare un numero minore di zero alla richiesta di indirizzo. Viene ritornato il valore della funzioneUSR (I) dove I è l'indirizzo dato.

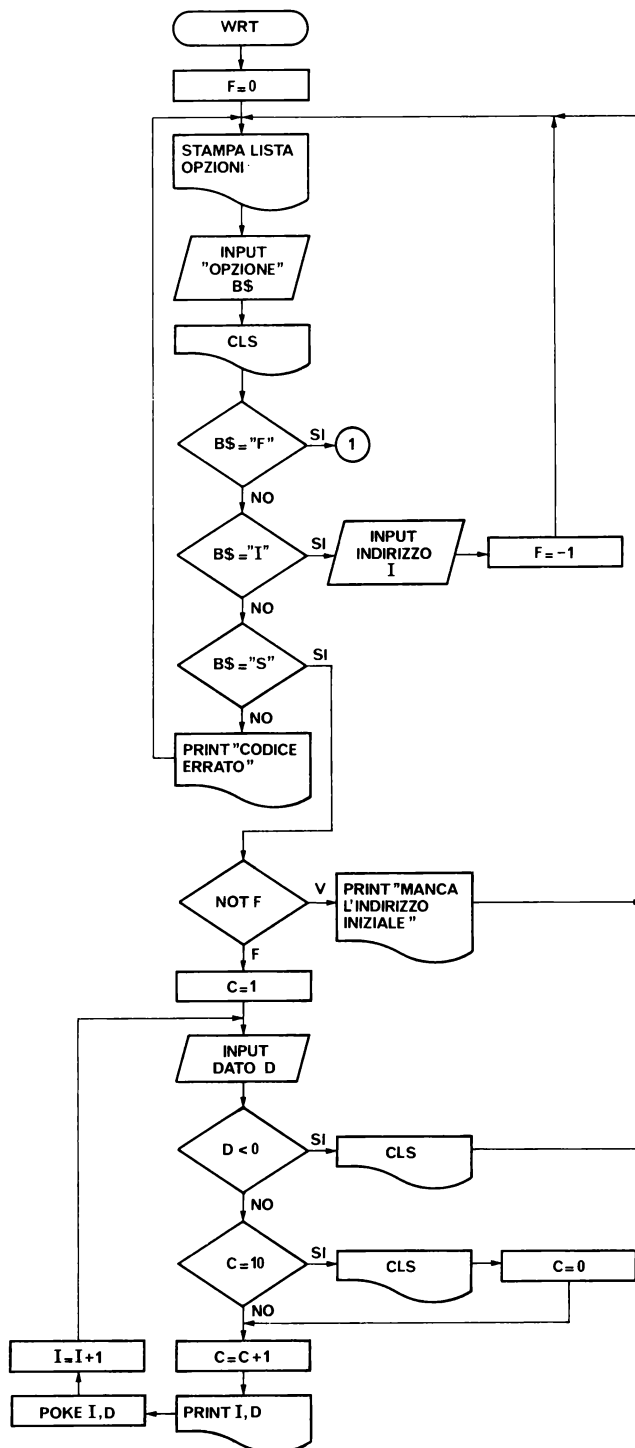
Routine MON:

Permette di controllare aree di memoria comprese tra due indirizzi specificati. Può essere usata per controllare aree di memoria ROM o per rileggere ciò che si è appena scritto con la routine WRT. Se il numero di locazioni di memoria supera le capacità del video richiesto un cambio pagina che continuerà la visualizzazione da dove era stata interrotta.

CARICATORE ASSEMBLER

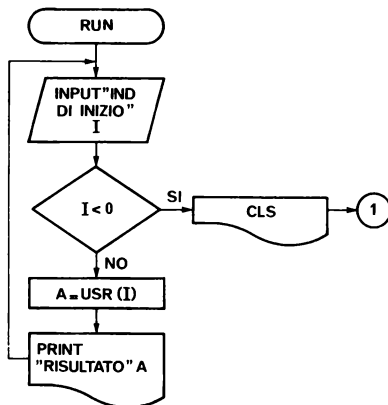


ROUTINE DI SCRITTURA WRT



CARICATORE ASSEMBLER

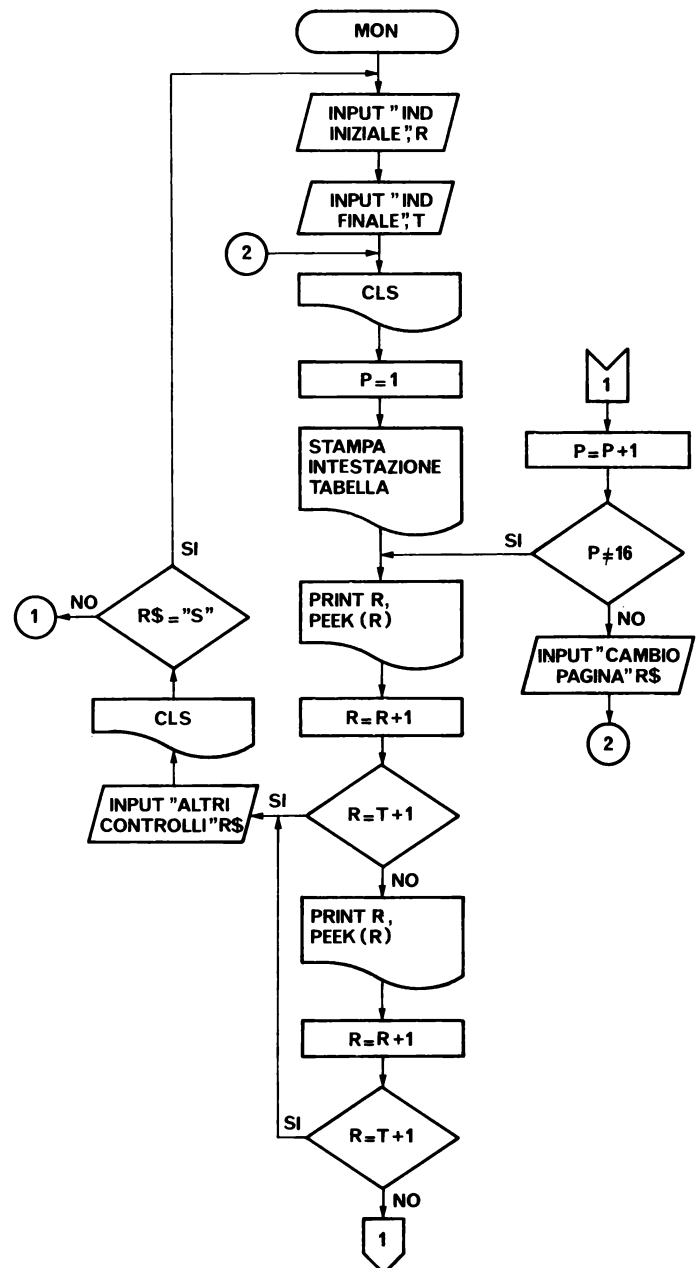
ROUTINE DI ESECUZIONE RUN



LISTING - CARICATORE ASSEMBLER

```

10  REM CARICATORE ASSEMBLER Z 80
20  PRINT
30  PRINT "ΔΔ***CARICATORE ASSEMBLER***"
40  PRINT
50  PRINT
60  PRINT "ΔΔΔΔΔ - LISTA OPZIONI -"
70  PRINT
80  PRINT "WRT - SCRITTURA"
90  PRINT "RUN - ESECUZIONE"
100 PRINT "MON-LETTURA"
110 PRINT "END - FINE"
120 PRINT
130 PRINT "ΔΔ OPZIONE = >";
140 INPUT A$
150 CLS
160 IF A$ = "WRT" THEN GO TO 1000
170 IF A$ = "RUN" THEN GO TO 2000
180 IF A$ = "MON" THEN GO TO 3000
190 IF A$ = "END" THEN STOP
200 PRINT "--CODICE ERRATO--"
210 GO TO 40
1000 REM ROUTINE WRT
1010 LET F=0
1020 PRINT "ΔΔΔΔΔ - LISTA OPZIONI -"
1025 PRINT
1030 PRINT "I - IND. INIZIO PROG."
1040 PRINT "S - SCRITTURA"
1050 PRINT "F-FINE"
1060 PRINT
1070 PRINT "ΔΔ OPZIONE = >"
1080 INPUT B$
  
```



```

1090 CLS
1100 IF B$ = "F" THEN GO TO 40
1110 IF B$ = "I" THEN GO TO 1150
1120 IF B$ = "S" THEN GO TO 1200
1130 PRINT "-- CODICE ERRATO --"
1140 GO TO 1020
1150 PRINT ">";
1160 INPUT I
1170 LET F = -1
1180 PRINT
1190 GO TO 1020
1200 IF NOT F THEN PRINT "-- ERRORE MANC/
IND. INIZIALE"
1210 IF NOT F THEN GO TO 1060
1215 LET C=1
1217 PRINT "PER USCIRE DARE UN NUM < 0"
1220 PRINT ">";
1230 INPUT D
1235 PRINT
1240 IF D < 0 THEN CLS
1250 IF D < 0 THEN GO TO 1020
1260 IF C=10 THEN CLS
1270 IF C=10 THEN LET C=0
1280 LET C= C+1
1290 PRINT I; "----";D
1300 POKE I,D
1310 LET I= I+1
1320 GO TO 1220
2000 REM ROUTINE RUN
2005 PRINT "PER USCIRE DARE UN NUM <0"
2010 PRINT
2020 PRINT "ΔΔ IND. INIZIO ROUTINE"
2030 INPUT I
2040 IF I < 0 THEN CLS
2050 IF I < 0 THEN GO TO 20 I < 0
2060 LET A=USR(I)
2070 PRINT
2080 PRINT "ΔΔ RISULTATO = >"; A
2090 PRINT
2100 PRINT "*****"
2110 GO TO 2010
3000 REM ROUTINE MON
3010 PRINT "IND. INIZIALE = >"
3020 INPUT R
3025 CLS
3030 PRINT "IND. FINALE = >"
3040 INPUT T
3045 PRINT
3050 CLS
3060 LET P=1
3070 PRINT "--IND---CONT---IND---CONT--"
3090 PRINT "Δ"; R, "ΔΔ"; PEEK (R),
3100 LET R= R+1
3110 IF R= (T+1) THEN GO TO 3190
3120 PRINT R, "Δ"; PEEK (R)
3130 LET R= R+1
3140 IF R= (T+1) THEN GO TO 3190
3145 LET P= P+1
3150 IF NOT P=16 THEN GO TO 3090
3160 PRINT "CAMBIO PAGINA >";
3170 INPUT R$
3180 GO TO 3050
3190 PRINT
3195 PRINT "ALTRI CONTROLLI (S/N)?";
3200 INPUT R$
3210 CLS
3220 IF R$ = "S" THEN GO TO 3010
3230 GO TO 20

```

NUMERI PRIMI

Autore: S. Papes
Programma utilizzante:
4 K di memoria

Questo programma ricerca tutti i numeri primi compresi tra 0 ed M (richiesto in input).

Questi vengono in seguito memorizzati in un vettore A e quindi visualizzati.

Per permettere una gestione ottimale di questo vettore si fa uso oltre a Y di un altro indice R che punta alla ultima locazione occupata.

Per verificare se un numero è primo o meno si compiono una serie di test, nei quali il numero da esaminare viene diviso con i numeri primi contenuti in A: se il risultato è un intero, il numero risulta divisibile e quindi non primo, in caso contrario si incrementa R e si memorizza in A(R) il numero.

Per sveltire l'elaborazione si è usato un accorgimento; infatti il numero preso in considerazione non è diviso per tutti i numeri primi contenuti in A ma solo con quelli minori della propria radice quadrata in quanto se non esiste un divisore tra questi non potrà matematicamente esserle tra quelli maggiori.

```

80 PRINT "**** NUMERI PRIMI ****"
90 PRINT
100 DIM A (75)
110 LET R=1
120 LET A (1) = 2
130 PRINT "SCRIVI IL NUMERO FINO A CUI SI"
135 PRINT "DEVONO RICERCARE I NUMERI PRIMI"
140 INPUT M
145 PRINT
147 PRINT A (1); "Δ";
150 FOR X=1 TO M
160 IF X > A(R)** 2 THEN GO TO 210
170 FOR Y = 1 TO R
180 IF A(Y)** 2 > X THEN GO TO 210
190 IF X = (X/A(Y))*A(Y) THEN GO TO 240
200 NEXT Y
210 LET R = R + 1
220 LET A(R) = X
230 PRINT X; "Δ";
235 IF R = (R/5)*5 THEN PRINT
240 LET X = X + 1
250 NEXT X
260 STOP

```



RIPASSO TABELLINE

Autore: R. Gibelli
Programma utilizzante:
4 K di memoria

Il programma in questione genera le tabelline dall'1 al 12.

Possiamo suddividerlo in due fasi principali.

La prima crea la tabellina del numero chiesto dall'esterno, ne stampa i primi 12 valori e chiede se si vuole proseguire con la stampa di altre tabelle.

In caso affermativo il ciclo di ripete altrimenti viene mandata in esecuzione la seconda fase.

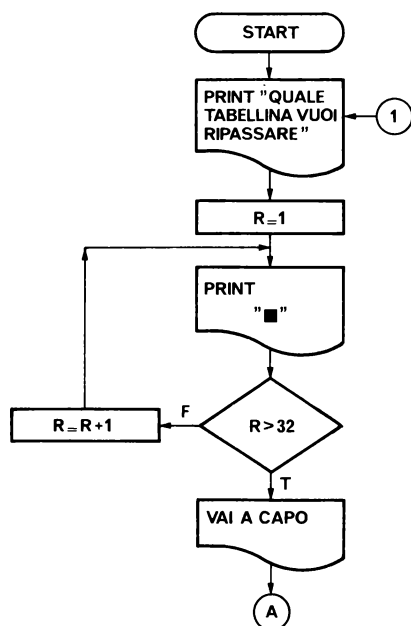
Con la seconda fase inizia l'esame di ripasso delle tabelline.

Si chiede quanti quesiti si desiderano ed il tempo disponibile per rispondere ad ognuno di essi.

Mediante Random vengono creati due valori che costituiranno la domanda in questione.

Dal momento in cui viene posta la domanda un apposito contatore calcolerà il tempo che intercorrerà tra questa e la risposta. Verrà quindi stampato se la risposta è esatta o meno. Nel 1° caso verranno stampati i complimenti ed il tempo impiegato per poi passare al prossimo quesito.

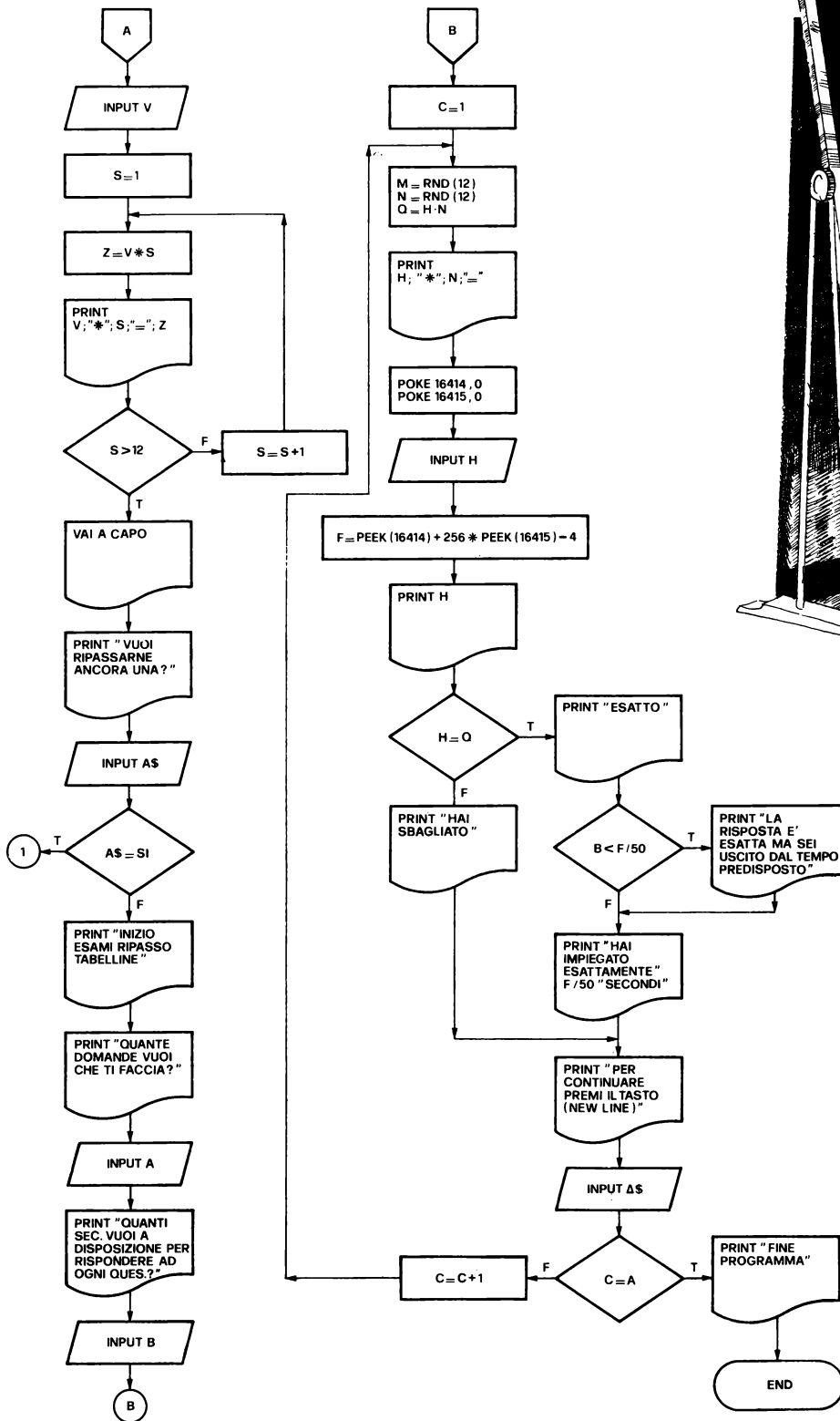
nel 2° verrà stampato il messaggio di errore e si passerà alla domanda successiva.



CODIFICA

```

10 PRINT "RIPASSO TABELLINE"
20 PRINT
40 CLS
100 PRINT "QUALE TABELLINA VUOI RIPASSARE?"
103 FOR R=1 TO 32
104 PRINT "■";
105 NEXT R
106 PRINT
107 PRINT
110 INPUT V
115 PRINT
120 FOR S=1 TO 12
130 LET Z = V*S
140 PRINT V; " * "; S; " = "; Z
160 NEXT S
165 PRINT
167 PRINT
170 PRINT "VUOI RIPASSARNE ANCORA UNA?"
180 INPUT A$
190 IF A$ = "SI" THEN GO TO 40
195 CLS
200 PRINT "INIZIO ESAME RIPASSO TABELLINE"
210 PRINT
220 PRINT
230 PRINT "QUANTE DOMANDE VUOI?"
240 INPUT A
245 CLS
250 PRINT "QUANTI SECONDI VUOI A DISPOSIZIONE PER RISPONDERE AD OGNI QUESITO?"
260 INPUT B
265 CLS
270 FOR C = 1 TO A
280 LET M = RND (12)
290 LET N = RND (12)
300 LET Q = M.N
310 PRINT M; " * "; N; " = ";
311 POKE 16414, 0
312 POKE 16415, 0
320 INPUT H
321 LET F = PEEK (16414) + 256 * PEEK (16415) - 4
325 PRINT H
330 IF H=Q THEN GO TO 360
340 PRINT "HAI SBAGLIATO"
350 GO TO 390
360 PRINT "ESATTO"
365 PRINT
370 IF B < F/50 THEN PRINT "RISPOSTA ESATTA MA FUORI TEMPO"
375 PRINT
380 PRINT "HAI IMPIEGATO ESATTAMENTE"; F/50; "SECONDI"
390 PRINT
400 PRINT "PER CONTINUARE PREMI IL TASTO (NEW LINE)"
410 INPUT D$
420 CLS
430 NEXT C
440 STOP
  
```



"CHOMP"

Autore: N. Cremaschi
Programma utilizzante:
4 K di memoria

Questo programma riproduce un divertente gioco chiamato Chomp.

Chomp: cioè mangiare un pezzo di formaggio, tutti insieme e poco alla volta; perde chi mangia l'ultimo pezzettino di formaggio. Si può giocare in due o più giocatori che faranno le proprie mosse alternativamente. Il pezzo di formaggio è rappresentato da una matrice 7x7 di elementi tutti uguali che sono dei quadratini, tranne uno, il primo elemento della matrice che è rappresentato da uno 0. Ogni concorrente nel suo turno di gioco deve mangiare un pezzettino di formaggio, cioè cancellare un pezzo della matrice. Questo è possibile farlo inserendo le coordinate di un quadratino, alla richiesta fatta dal programma. Le coordinate vanno inserite sotto forma di un numero di due cifre di cui la prima rappresenta il numero della riga prescelta, la seconda il numero della colonna prescelta.

Il programma provvederà automaticamente a cancellare i quadratini a destra del numero di colonna e al di sotto del numero di riga prescelto. Come si è detto perde il concorrente che mangia l'ultimo pezzettino di formaggio, quindi perde il concorrente che è costretto ad inserire le coordinate del punto 0. Il giocatore perdente verrà segnalato dal programma mediante un messaggio.



ANALISI

La prima parte del programma è a carattere discorsivo; si stampa il nome del programma e alcune domande per sapere se i concorrenti desiderano sapere le principali regole del gioco. Nel caso in cui la risposta è affermativa faccio stampare le frasi esplicative che sono poste in fondo al programma; una volta fatto questo comincio il programma vero e proprio. Dimensiono un vettore di 49 elementi e svolgo il primo ciclo.

I ciclo (180 - 210) riempie il vettore mettendo in ogni posizione il valore 6 che corrisponde al carattere grafico quadratino nella prima posizione il valore 52 che corrisponde allo 0.

A questo punto vi è la prima fase di input dove si chiede quanti sono i giocatori che partecipano e inizializzo una variabile che servirà a determinare il turno di gioco dei concorrenti.

II e III ciclo (260 - 330) sono due ciclo concatenati che mi permettono di stampare il contenuto del vettore come se fosse una matrice 7x7 con i loro rispettivi numeri riga.

IV ciclo (350 - 370) con questo ciclo posso stampare il numero delle colonne della matrice visualizzandoli opportunamente sotto la rispettiva colonna.

A questo punto si può considerare finita la prima parte del programma che consiste nel far visualizzare all'utente il gioco, mediante la stampa della matrice.

Ora si può cominciare a giocare.

Chiedo in INPUT dal primo giocatore le coordinate del quadratino prescelto e controllo se sono giuste altrimenti torno a chiedere le coordinate.

Mediante alcuni artifici si creano delle variabili che si servono per individuare la posizione dei quadratini da cancellare nel vettore a seconda della scelta fatta dal giocatore nella matrice. Con dei controlli mi assicuro che le coordinate siano esatte, cioè che rappresentino una posizione del vettore, e che quella posizione non sia già stata cancellata.

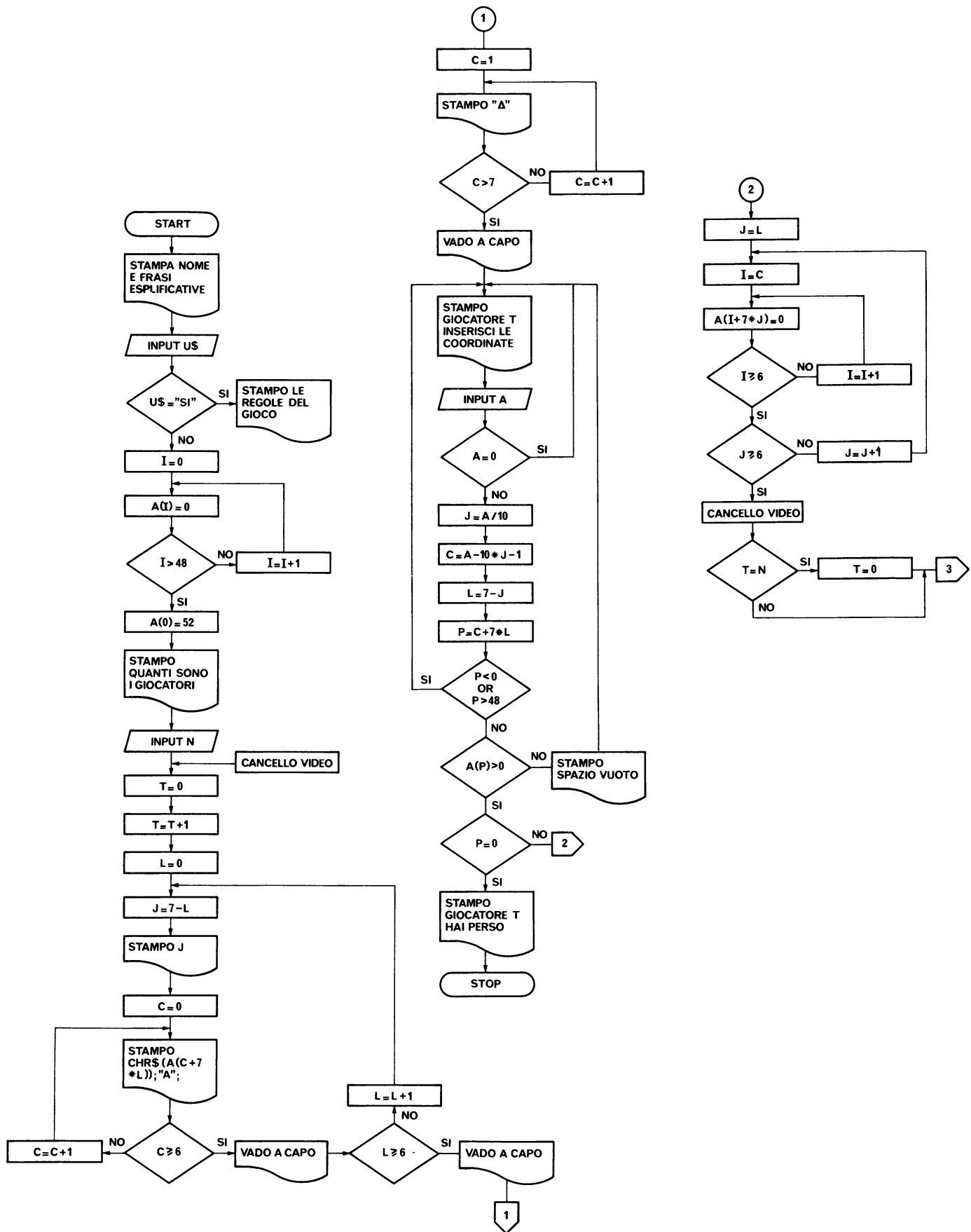
Se le coordinate non sono esatte stampo messaggi di errore e torno all'input.

Se la coordinata e quella dello 0 stampo che il giocatore che ha inserito queste coordinate ha perso.

Se tutto è regolare cioè la coordinata rappresenta un quadratino svolgo V e VI ciclo (610 - 650) due cicli concatenati che mi cancellano la parte della matrice che voglio eliminare, cioè inseriscono al posto dei quadratini tanti blank.

Il giocatore ha così esaurito il suo turno di gioco e dopo aver pulito il video con opportune istruzioni passo al giocatore successivo e torno all'inizio del programma.

Le ultime istruzioni sono la stampa di frasi esplicative che illuminano parzialmente sull'andamento del gioco.



CODIFICA

```

100 PRINT "CHOMP"
110 PRINT
120 PRINT
130 PRINT
140 PRINT "VUOI CONOSCERE LE PRINCIPALI
    REGOLE DEL GIOCO"
150 INPUT U$
160 IF U$ = "SI" THEN GO TO 800
170 DIM A(48)
180 FOR I=0 TO 48
190 LET A(I) = 6
200 NEXT I
210 LET A(0) = 52
220 PRINT "QUANTI SONO I GIOCATORI?"
230 INPUT N
235 CLS
240 LET T=0
250 LET T=T+1
260 FOR L=0 TO 6
270 LET J=7-L
280 PRINT J
290 FOR C=0 TO 6
300 PRINT CHR$(A(C+7*L)); "Δ";
310 NEXT C
320 PRINT
330 NEXT L
340 PRINT,
350 FOR C=1 TO 7
360 PRINT C; "Δ";
370 NEXT C
375 PRINT
380 PRINT "GIOCATORE:"; T
410 PRINT "INSERISCI LE COORDINATE"
420 INPUT A
430 IF A=0 THEN GO TO 410
440 LET J= A/10
450 LET C= A-10*J-1
460 LET L= 7-J
470 LET P=C + 7 * L
480 IF P < 0 OR P > 48 THEN GO TO 420
490 IF A(P) > 0 THEN GO TO 600
500 PRINT "SPAZIO VUOTO"
510 GO TO 410
600 IF P=0 THEN GO TO 700
610 FOR J = L TO 6
620 FOR I=C TO 6
630 LET A (I+7*J)=0
640 NEXT I
650 NEXT J
660 CLS
670 IF T=N THEN LET T=0
680 GO TO 250
700 PRINT "GIOCATORE"; T; "- HAI PERSO -"
710 STOP
800 PRINT "CHOMP È UN GIOCO CHE È COSTI-
    TUITO DA UNA MATRICE 7x7 DI ELEMENTI
    TUTTI UGUALI"
810 PRINT "TRANNE UNO IL PRIMO. POSSONO
    GIOCARE DUE O PIU' GIOCATORI. OGNI
    CONCORRENTE NEL SUO"
820 PRINT "TURNO DI GIOCO PUO' ELIMINARE
    UN CERTO NUMERO DI QUADRATINI.
    PERDE IL CONCORRENTE CHE RIMANE NEL
    SUO TURNO DI GIOCO"
830 PRINT "CON SOLO L'ULTIMO ELEMENTO"
840 GO TO 170

```

L'IMPICCATO

Autore: P. Ciceri
Programma utilizzante:
4 K di memoria

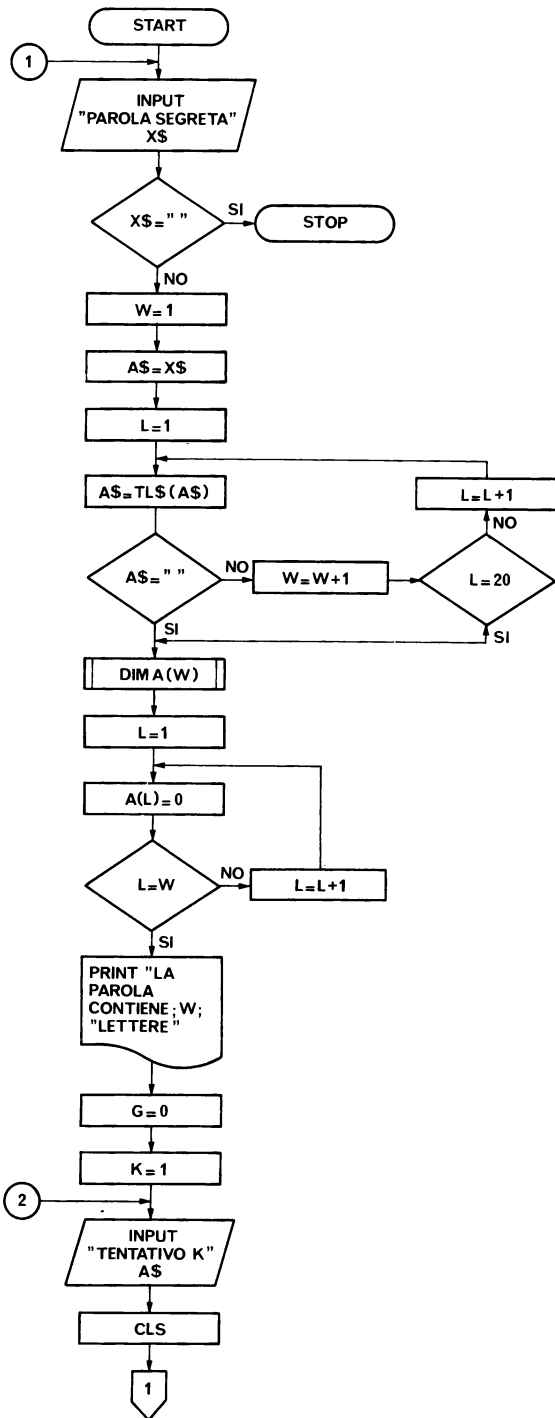
Il gioco consiste nell'indovinare una parola segreta che un nostro amico ha precedentemente inserito. Il tentativo consiste nell'inserire una lettera che noi pensiamo sia presente nella parola segreta, se c'è vedremo comparire nello schermo della parola la lettera nella sua giusta posizione, se invece ci siamo sbagliati si aggiungerà un pezzo all'omino impiccato, dopo 10 errori si finisce impiccati e viene richiesta una nuova parola segreta.

Se durante il gioco ci si arrende basta dare NEW LINE alla richiesta di tentativo; se invece ci si è stancati di giocare si dà NEW LINE alla richiesta di parola segreta.

Variabili usate

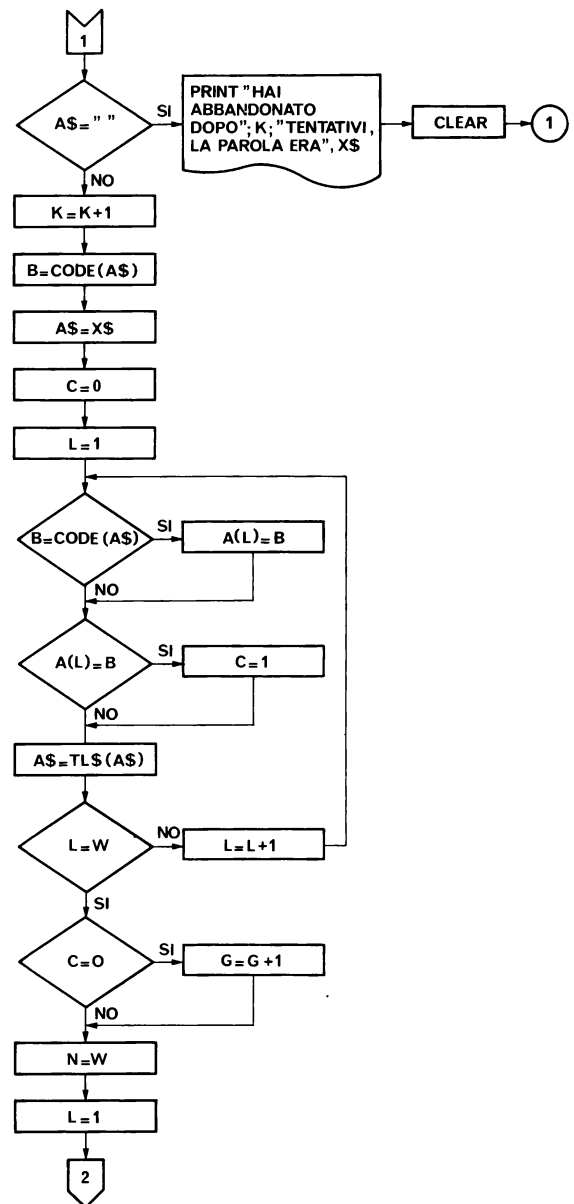
| | |
|-------|----------------------------------|
| X\$ | Parola segreta |
| W | Numero di lettere componenti X\$ |
| K | Contatore tentativi |
| A\$ | Tentativo, variabile di comodo |
| B | Codice ASCII tentativo |
| G | Contatore tentativi sbagliati |
| C | Flag |
| N, L | Contatori |
| A (W) | Vettore tentativi |





```

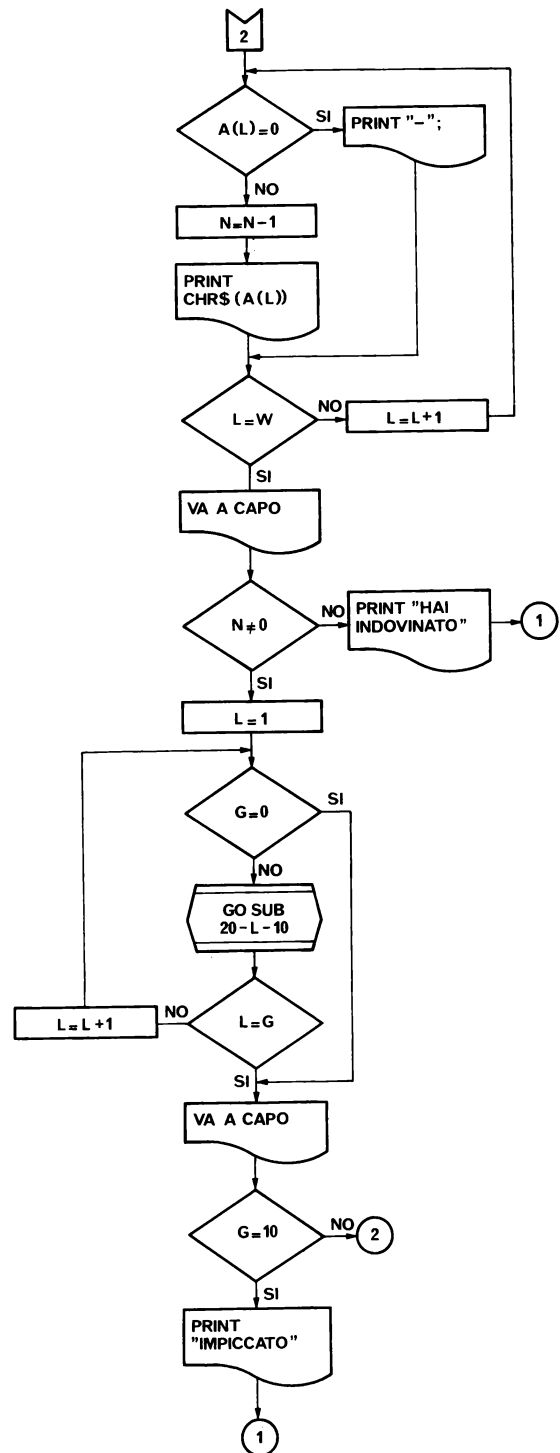
05 GO TO 300
10 PRINT "■■■■"
20 RETURN
30 PRINT "■■■■"
40 RETURN
50 PRINT "ΔΔΔ ■"
60 RETURN
70 PRINT "ΔΔ ■"
80 RETURN
90 PRINT "ΔΔ"; CHR$(135); CHR$(135); "■"
100 RETURN
110 PRINT "ΔΔ"; CHR$(134);
120 RETURN
130 PRINT CHR$(134); "■"
140 RETURN
150 PRINT "ΔΔ"; CHR$(130); CHR$(128)
160 RETURN
  
```



```

170 PRINT "ΔΔ"; CHR$(132)
180 RETURN
190 PRINT CHR$(130); "■"
200 RETURN
210 FOR L=1 TO G
220 IF G=0 THEN GO TO 250
230 GO SUB 20*L-10
240 NEXT L
250 PRINT
260 PRINT
270 IF G=10 THEN GO TO 290
280 GO TO 490
290 PRINT "IMPICCATO"
300 PRINT
310 PRINT "INSERISCI LA PAROLA SEGRETA"
320 INPUT X$
330 IF X$="" THEN GO TO 900
340 CLS
350 LET W=1
360 LET A$= X$
370 FOR L=1 TO 20
380 LET A$ = TL$ (A$)
390 IF A$="" THEN GO TO 420
400 LET W = W+1
410 NEXT L
420 DIM A (W)
430 FOR L=1 TO W
440 LET A (L)=0
450 NEXT L
460 PRINT "LA PAROLA CONTIENE Δ"; W; "Δ LET-
TERE"
470 LET G=0
480 LET K=1
490 PRINT "TENTATIVO Δ"; K;" = >"
500 INPUT A$
510 CLS
520 IF A$="" THEN GO TO 540
530 GO TO 570
540 PRINT "HAI ABBANDONATO DOPO Δ"; K; "Δ
TENTATIVI"
550 PRINT "LA PAROLA ERA Δ"; X$
550 CLEAR
560 GO TO 300
570 LET K=K+1
580 LET B= CODE (A$)
590 LET A$ = X$
600 LET C=0
610 FOR L=1 TO W
620 IF B= CODE (A$) THEN LET A (L)=B
630 IF A (L)=B THEN LET C=1
640 LET A$ = TL$ (A$)
650 NEXT L
660 IF C=0 THEN LET G=G+1
670 LET N=W
680 FOR L=1 TO W
690 IF A(L)= THEN PRINT "-";
700 IF A(L)=0 THEN GO TO 730
710 LET N=N-1
720 PRINT CHR$ (A(L));
730 NEXT L
740 PRINT
750 PRINT
760 IF NOT N=0 THEN GO TO 210
780 PRINT "HAI INDOVINATO LA PAROLA"
790 GO TO 300
900 PRINT
910 PRINT "ARRIVEDERCI ALLA PROSSIMA VOL-
TA"

```



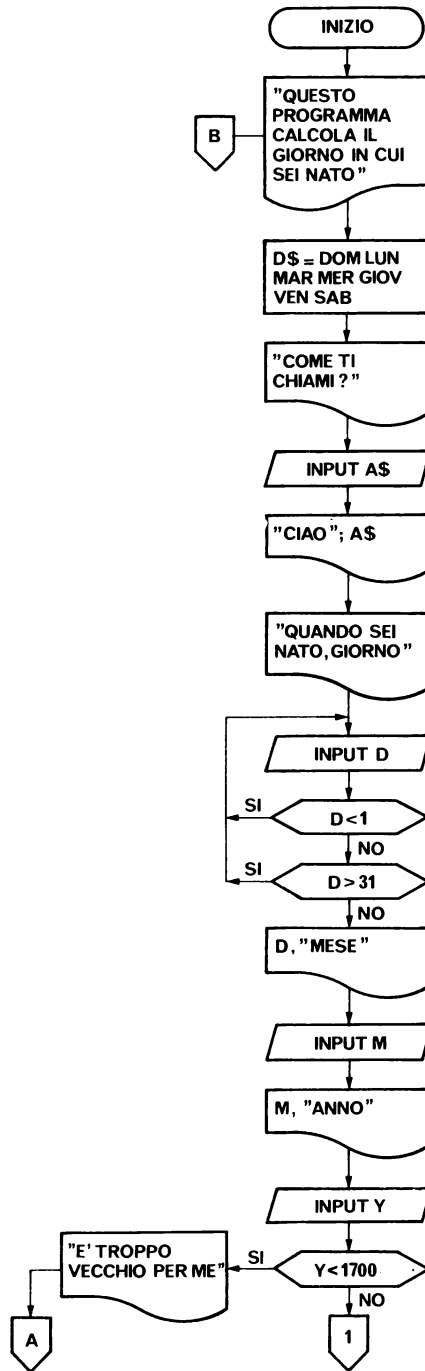
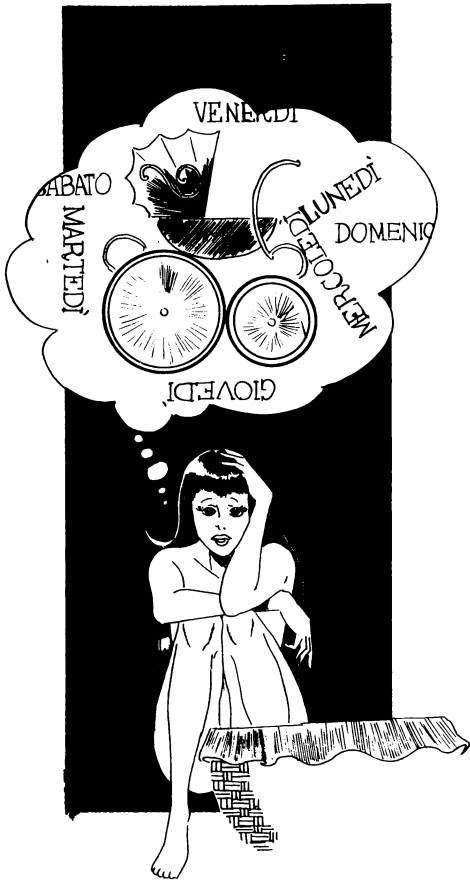
GIORNI DELLA SETTIMANA

Autore: A. Planamente
Programma utilizzante:
4 K di memoria

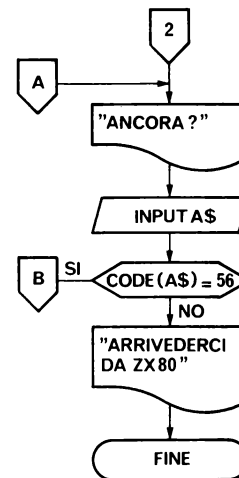
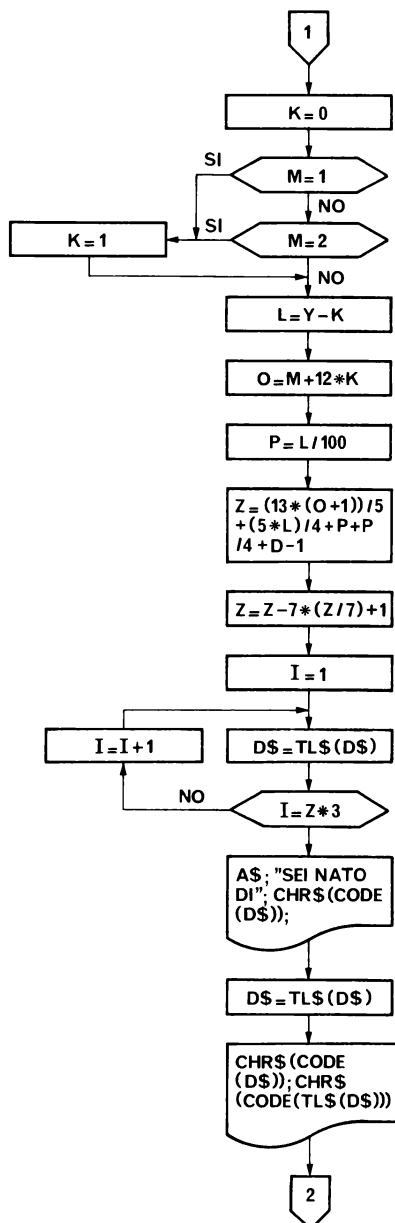
Questo programma permette di soddisfare la nostra curiosità nel caso volessimo sapere in quale giorno della settimana siamo nati. Per ottenere ciò, basta inserire la data di nascita completa, al momento dell'esecuzione del programma, ovviamente quando viene richiesta.

Premettiamo innanzitutto, però, che vi sono dei limiti per quanto riguarda il funzionamento del programma, ossia vengono fornite risposte solo per date che vanno dal 1700 in poi. Detto ciò vediamo di comprendere con quale tecnica il programma riesce a fornire tali risposte, cioè descriviamo l'algoritmo utilizzato. Distinguiamo due casi a secondo se l'anno è bisestile oppure no; a questo punto utilizzando due laboriosi calcoli, che non stiamo a precisare, determiniamo quali caratteri della stringa D\$ considerare, avendo precedentemente memorizzato in D\$ i caratteri corrispondenti ai 7 giorni della settimana.

Il programma termina quanto alle richieste di una nuova elaborazione si risponde negativamente, altrimenti inserendo una nuova data il programma ripeterà l'operazione.



GIORNI DELLA SETTIMANA



CODIFICA

```

90  PRINT "**** GIORNO DELLA SETTIMANA ****"
95  PRINT
100  PRINT "QUESTO PROGRAMMA CALCOLA IL"
105  PRINT "GIORNO IN CUI SEI NATO"
110  LET D$ = "DOMLUNMARMERGIOVENSAB"
115  PRINT
120  PRINT "COME TI CHIAMI?"
130  INPUT A$
140  PRINT "CIAO"; A$
145  PRINT
150  PRINT "QUANDO SEI NATO?: GIORNO",
160  INPUT D
170  IF D < 1 OR D > 31 THEN GO TO 160
180  PRINT D, "MESE",
190  INPUT M
200  PRINT M, "ANNO"
210  INPUT Y
215  PRINT
220  IF Y < 1700 THEN PRINT A$, "È TROPPO VEC-
    CHIO PER ME"
230  IF Y < 1700 THEN GO TO 440
240  CLS
250  LET K=0
260  IF M=1 OR M=2 THEN LET K=1
270  LET L=Y-K
280  LET O=M+12*K
290  LET P = L/100
300  LET Z=(13*(O+1))/5+(5*L)/4-P+P/4+D-1
310  LET Z= Z-7*(Z/7)+1
320  FOR I = 1 TO Z * 3
330  LET D$ = TL$(D$)
340  NEXT I
400  PRINT A$, "SEI NATO DI"; CHR$(CODE (D$));
410  LET D$ = TL$(D$)
420  PRINT CHR$(CODE (D$)); CHR$(CODE (TL$(D$)))
440  PRINT "ANCORA?"
450  INPUT A$
460  IF CODE (A$)=56 THEN GO TO 110
470  PRINT "ARRIVEDERCI DA ZX-80"
475  PRINT
  
```

GIOCO DEL TRIS

Autore: **M. Lefons**
Programma utilizzante:
4 K di memoria

Il seguente programma realizza l'algoritmo per il cosiddetto "gioco del tris". Gioco che consiste nel riuscire ad allineare 3 pedine in un quadrato diviso in 9 parti: assegnate le rispettive pedine a 2 giocatori essi le devono alternativamente porre nelle 9 sezioni del quadrato in modo da ottenere una sequenza consecutiva orizzontale, verticale o diagonale formata, appunto, da 3 pedine uguali. Vince chi riesce a realizzare per primo una terzina.

Il programma permette quindi di giocare a tris contro l'elaboratore ZX-80 che esegue di diritto la prima mossa ponendo una sua pedina (contrassegnata da una "X") nella zona centrale del quadrato. Le rimanenti 8 posizioni sono numerate in modo che il giocatore possa comodamente selezionare scrivendo il numero corrispondente.

La visualizzazione su video di questa "griglia" quadrata di 9 sezioni avviene mediante cicli di stampa ridotti a sottoprogrammi interni.

Essa quindi dipenderà ogni volta dal numero di sezione scelto dal giocatore che causerà a sua volta una determinata scelta da parte dell'elaboratore avversario. La pedina del giocatore è contrassegnata da uno 0.

Molto utile alla visualizzazione delle mosse è la funzione di stringa CHR\$(N) che permette di associare a determinati codici numerici le cifre da 1 a 8, la pedina X e a pedina 0:

| N | CHR\$(N) |
|----|----------|
| 28 | 0 |
| 29 | 1 |
| 30 | 2 |
| 31 | 3 |
| 32 | 4 |
| 33 | 5 |
| 34 | 6 |
| 35 | 7 |
| 36 | 8 |
| 37 | 9 |
| 61 | X |

Le linee dalla 10 alla 40 servono alla visualizzazione di alcuni messaggi di presentazione del gioco.

Le linee 110-180 generano l'insieme di codici numerici per la CHR\$ che vengono posti nel vettore A.

Le linee 200-230-310 comprendono praticamente i 2 sottoprogrammi di stampa della griglia (uno che parte dalla linea 200 e uno che parte dalla linea 230) che fanno uso della CHR\$.

Le linee 400-830 comprendono la parte di programma che si occupa di passare i giusti parametri alle sub a

seconda delle mosse del giocare e una parte finale che visualizza il resoconto delle partite giocate e che elenca il numero di sconfitte e il numero di pareggi conseguiti. La vittoria infatti può essere solo dell'elaboratore perché a lui spetta di diritto la prima mossa. Il giocatore può, al massimo, non essere sconfitto e pareggiare la partita.

RIPARTIZIONE DELLE SEZIONI E LORO NUMERAZIONE

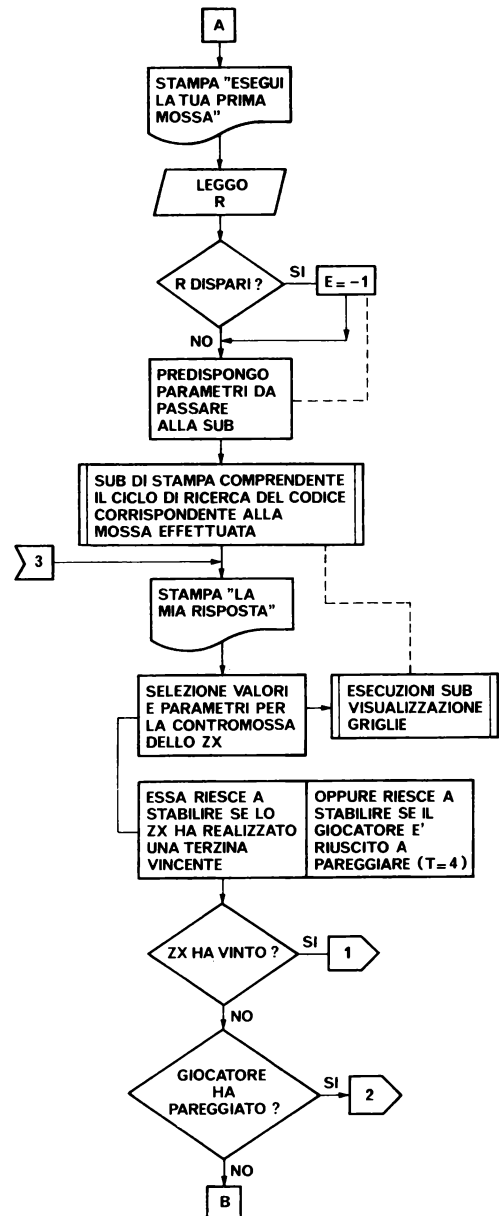
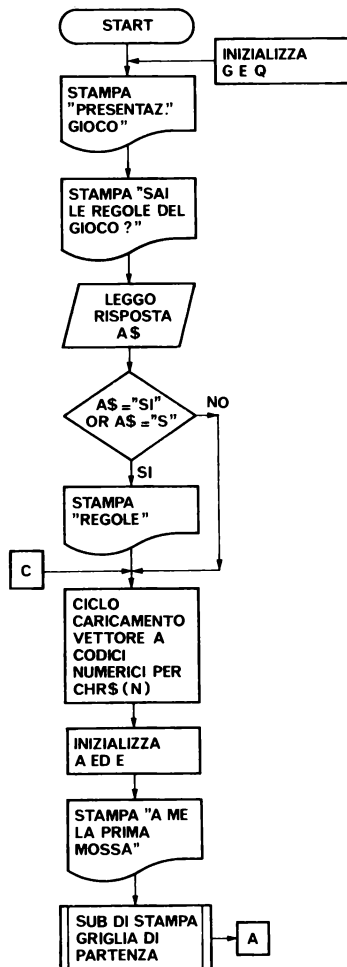
Ricordo che la prima mossa spetta di diritto all'elaboratore, la "griglia" iniziale che viene visualizzata è la seguente:

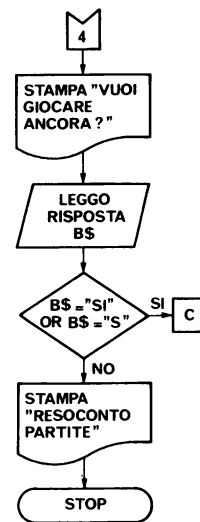
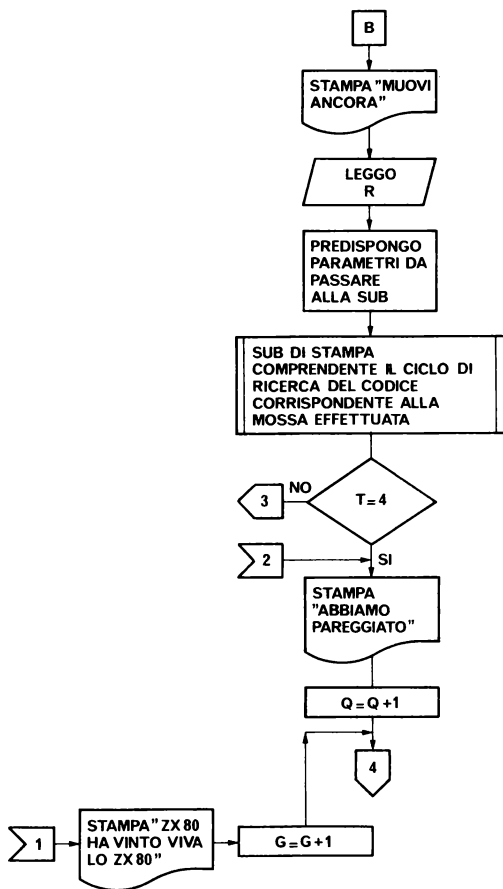
| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | X | 4 |
| 7 | 6 | 5 |

ELENCO VARIABILI, CONTATORI, INDICI

- Q = contatore sconfitte
- G = contatore pareggi
- F = contatore per la stampa della cornice di presentazione gioco
- A\$ = contiene la stringa risposta alla domanda di linea 27
- A() = vettore di 9 posizioni contenente gli indici numerici per la CHR\$
- J e I = sono le variabili-indici che, legate da operatori numerici, formano gli indici del vettore A
- A = contiene di volta in volta dei valori che, a seconda delle mosse del giocatore, vengono sommati ad altri valori per passare i giusti parametri alle sub
- E = flag di controllo mosse per la modifica di A (viene posto = -1 se la mossa del giocatore corrisponde a una sezione contrassegnata da un numero dispari. Le sezioni dispari sono quelle che occupano i 4 vertici del quadrato)
- R = contiene il numero di sezione selezionato del giocatore
- M = contiene di volta in volta dei valori che, sommati a 28, danno il codice numerico (da ricercare nel vettore A) corrispondente al numero sezione della mossa richiesta dal giocatore o della contro-mossa voluta dall'elaboratore stesso
- V = contiene il valore 28 se la mossa da visualizzare è quella del giocatore (infatti CHR\$(28) = 0), contiene il valore 61 se la mossa da visualizzare è quella dell'elaboratore (infatti CHR\$(61) = X)
- T = contatore mosse dopo le 2 iniziali (la prima dello ZX, la seconda del giocatore)
- P = ha in pratica la stessa funzione di A con l'unica differenza che P è la variabile che passa direttamente a M i valori-parametro che occorrono alle sub per la stampa
- B\$ = contiene la stringa risposta alla domanda di linea 800.

GIOCO DEL TRIS





GIOCO DEL TRIS

CODIFICA BASIC

```

10  REM PROGRAMMA GIOCO DEL TRIS
11  LET Q=0
12  LET G=0
15  PRINT, "GIOCO DEL TRIS"
17  FOR F = 1 TO 32
19  PRINT "■";
21  NEXT F
23  PRINT
25  PRINT
27  PRINT "SAI LE REGOLE DEL GIOCO?"
28  PRINT
29  PRINT
30  INPUT A$
31  IF A$ = "SI" OR A$ = "S" THEN GO TO 100
33  PRINT "CONSISTE NELL'ALLINEARE 3 PEDINE
    IN UN QUADRATO DI 9 POSIZIONI"
35  PRINT "LA PEDINA DELLO ZX LA X, LA TUA
    PEDINA E LO 0"
36  FOR F = 1 TO 32
37  PRINT "■";
38  NEXT F
39  PRINT
40  PRINT
100 DIM A(9)
110 FOR J=0 TO 2
120 FOR I=1 TO 3
130 LET A(I+3J)=28+I+4 J+2 I*(J>1)
140 NEXT I
150 NEXT J
160 LET A(4)=36
170 LET A(5)=61
180 LET A(6)=32
190 GO TO 400
200 FOR I = 1 TO 9
210 IF A(I)=M+28 THEN LET A(I)=V
220 NEXT I
230 FOR I= 0 TO 2
240 FOR J=1 TO 3
250 PRWT CHR$( A(J+3I)); "Δ";
260 NEXT J
270 PRINT
280 PRINT
290 NEXT I
300 PRINT

310 RETURN
400 LET A=0
410 LET E=0
420 PRINT "A ME LA PRIMA MOSSA"
430 GO SUB 230
440 PRINT "ESEGUI LA TUA PRIMA MOSSA"
450 INPUT R
460 CLS
470 IF R= 2 (R/2)
480 LET M=R
490 LET V=28
500 GOSUB 200
510 LET P=R
520 FOR T=1 TO 4
530 PRINT "LA MIA RISPOSTA:"
540 LET V=61
550 LET A=A+1
560 IF T=1 OR R=P+4 OR R=P-4 THEN GO TO
    620
570 LET P=P+4
580 IF P > 8 THEN LET P=P-8
590 LET M=P
600 GOSUB 200
610 GO TO 790
620 IF A=3 AND E THEN LET A=7
630 IF A=4 THEN LET A=6
640 LET P=P+A
650 IF P > 8 THEN LET P=P-8
660 LET M=P
670 GOSUB 200
680 IF A=7 THEN GO TO 790
690 IF T=4 THEN GO TO 770
700 PRINT "MUOVI ANCORA"
710 INPUT R
720 CLS
730 LET M=R
740 LET V=28
750 GOSUB 200
760 NEXT T
770 PRINT "ABBIAMO PAREGGIATO"
771 LET Q=Q+1
773 GO TO 800
790 PRINT "HO VINTO-VIVA LO ZX-80"
792 LET G=G+1
794 PRINT
795 PRINT
800 PRINT "VUOI GIOCARE ANCORA?"
801 PRINT
802 PRINT
810 INPUT B$
820 CLS
821 IF B$ = "SI" OR B$ = "S" THEN GO TO 100
823 PRINT "HAI GIOCATO"; G+Q; "PARTITE"
824 PRINT "HAI PERSO"; G; "VOLTE"
825 PRINT "SEI RIUSCITO A PAREGGIARE"; Q;
    "VOLTE"
830 STOP

```

MASTER MIND

Autore: **G. Bortone**
Programma utilizzante:
4 K di memoria

Il gioco consiste nell'indovinare per tentativi le cifre di una combinazione segreta. Non è sufficiente indovinare le singole cifre, ma si deve indovinare anche la posizione di ogni singola cifra.

La combinazione segreta può essere formata da 1 a 7 cifre numeriche. Il giocatore sceglie il numero delle cifre della combinazione ed il livello di difficoltà; cioè quanti valori diversi può assumere ogni cifra (non meno del numero delle cifre e non più di 10).

Una cifra non può figurare più di una volta nella combinazione.

Le cifre del primo tentativo possono essere trovate a caso o secondo una logica di gioco per i più esperti.

Per introdurre le cifre desiderate, si premono i relativi tasti e poi NEW LINE. La prima cifra inserita viene accettata come prima cifra a sinistra. Se le cifre introdotte sono troppe, quelle in più a destra vengono ignorate. Se si vuole correggere il numero introdotto prima si preme NEW LINE si può usare SHIFT & REBOUT.

Un numero accettato dal calcolatore non può essere corretto. Il programma accetta anche lo spazio al posto di una cifra e considera lo spazio come cifra non presente. Il programma memorizza il numero introdotto e lo confronta con la combinazione segreta. Dopo questo confronto viene evidenziato:

- il numero di cifre esatte presenti anche se non in giusta posizione;
- il numero di cifre esatte presenti in giusta posizione
- "xxx" per segnalare che tutte le cifre introdotte sono presenti nella combinazione.

Utilizzando i risultati di ogni tentativo il giocatore deve indovinare la combinazione segreta.

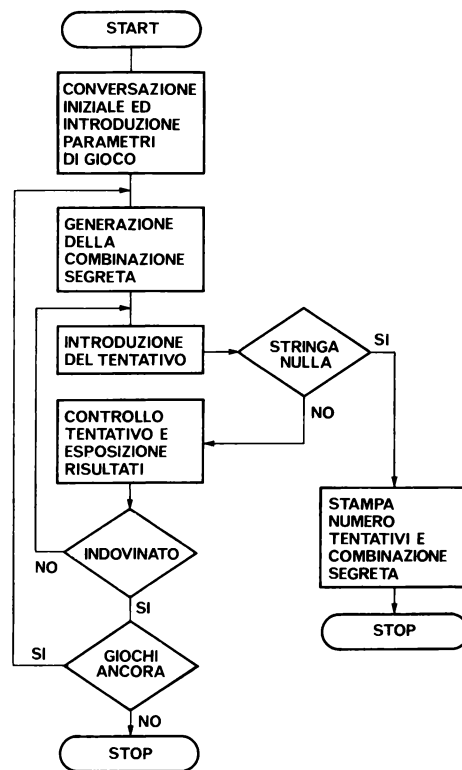
Una combinazione di 4 cifre con valori tra 0 e 6, può essere indovinata in 4, 5 o 6 mosse.

Se si vuole abbandonare la partita basta premere NEW LINE ed il programma stampa la combinazione segreta. Se si vuole interrompere il programma nella fase delle domande iniziali basta introdurre zero.

Se non si indovina la combinazione entro il quindicesimo tentativo lo schermo si riempie completamente ed il programma dà errore di tipo 5. Si può schiacciare CONT. per continuare, ma si perde il contenuto dello schermo.

LISTA VARIABILI

I, J indici per il controllo di cicli
N = numero delle cifre
A(N) vettore che contiene la combinazione segreta
D = livello di difficoltà
X = cifra della combinazione
G = numero dei tentativi
B\$ = tentativo
H(N) = vettore del tentativo
R = numero cifre indovinate
C = numero posizioni indovinate.



MASTER MIND

```

100 PRINT "BUON DIVERTIMENTO CON MASTER
110 MIND"
120 PRINT
125 PRINT "CON QUANTE CIFRE VUOI GIOCARE?"
130 PRINT "MASSIMO 7"
135 INPUT N
136 PRINT N
137 IF N > 7 THEN GO TO 110
138 IF N < 1 THEN STOP
140 LET N = N - 1
150 DIM A(N)
160 PRINT
170 PRINT "CHE LIVELLO DI DIFFICOLTA' VUOI?"
175 PRINT "SCEGLI TRA"; N+1; "E 10"
180 INPUT D
190 PRINT D
194 IF D < 1 THEN STOP
195 IF D < (N+1) OR D > 10 THEN GO TO 160
196 FOR I=0 TO N
197 LET A(N)=10
198 NEXT I
200 FOR I = 0 TO N
210 LET X = RND(D) - 1
220 FOR J=0 TO I
230 IF X=A(J) THEN GO TO 210
240 NEXT J
250 LET A(I) = X
260 NEXT I
270 LET G=0
  
```


MASTER MIND

```

300 CLS
302 IF NOT N=0 THEN GOTO 305
303 PRINT "—SCRIVI"; N+1; "CIFRE TRA 0
    E"; D—1;
304 GOTO 310
305 PRINT "—SCRIVI"; N+1; "CIFRE TRA
    0 E"; D—1;
310 PRINT "< N > -TENTATIVI--CIFRE--POSIZIONI"
320 PRINT
330 INPUT B$
333 IF B$ = " " THEN GO TO 700
335 LET G=G+1
340 DIM H(N)
350 LET R=0
360 LET C=0
400 FOR I=0 TO N
410 LET X= CODE (B$)—28
415 LET H(I) =X
420 IF X=A(I) THEN LET C=C+1
430 FOR J=0 TO N
440 IF X= A(J) THEN LET R=R+1
450 NEXT J
460 LET B$ =TL$ (B$)
470 NEXT I
500 PRINT G,
501 FOR I=0 TO N
502 IF H(I) <0 OR H(I) >9 THEN PRINT "Δ";
503 IF H(I) <0 OR H(I) >9 THEN GO TO 507
506 PRINT H(I);
507 NEXT I
508 IF R=N+1 THEN PRINT, R-C; "Δxxx", C
509 IF R=N+1 THEN GO TO 511
510 PRINT, R-C, C
511 IF C=N+1 THEN GOTO 600
520 GOTO 330
600 PRINT
605 IF NOT G=1 THEN GOTO 610
606 PRINT "SEI RIUSCITO IN"; G; "TENTATIVO"
607 GO TO 620
610 PRINT "SEI RIUSCITO IN "G" TENTATIVI"
620 PRINT
630 PRINT "VUOI GIOCARE ANCORA? S/N";
640 INPUT S$
650 IF S$ = "S" THEN GOTO 196
655 STOP
700 PRINT
710 PRINT "ABBANDONI DOPO"; G; "TENTATIVI"
720 PRINT
730 PRINT "LA COMBINAZIONE SEGRETA ERA"
740 FOR I=0 TO N
750 PRINT A(I);
760 NEXT I

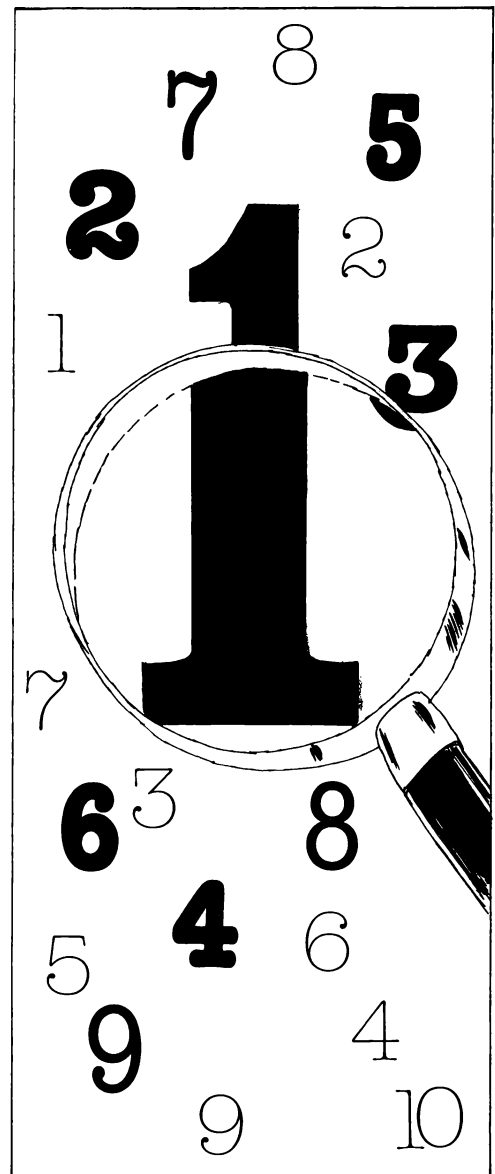
```

IMPARIAMO LA MATEMATICA

Autore: Dr.ssa R. Bonelli
Programma utilizzante:
4 K di memoria

Questo programma propone l'esecuzione di semplici calcoli algebrici usando i caratteri ingranditi. Può quindi essere interessante per esercitarsi nel calcolo rapido mentale e per imparare le tecniche di ingrandimento dei caratteri.

Vengono scelti a caso due numeri compresi tra 1 e 49 e vengono proposti o in somma o in sottrazione. Nelle linee da 200 a 350 è contenuta la routine per ottenere l'ingrandimento dei caratteri.



RADICE QUADRATA CON 3 DECIMALI

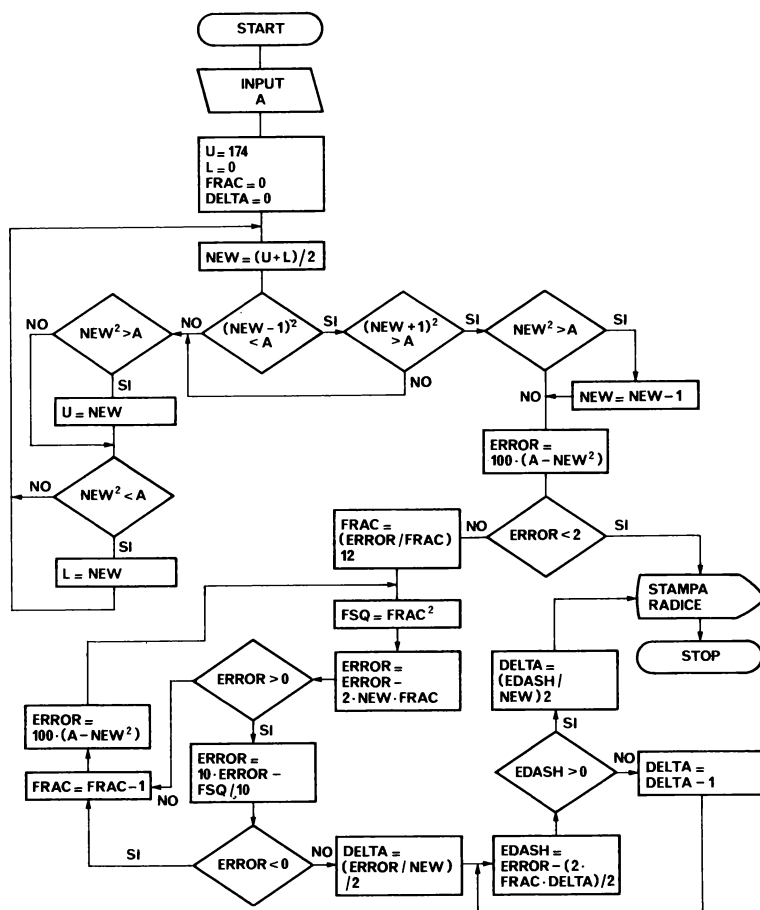
Autore: G. Bortone
Programma utilizzante:
1 K di memoria

```

10 REM IMPARIAMO LA MATEMATICA
20 PRINT "IMPARIAMO LA MATEMATICA"
100 RANDOMISE
110 DIM C(3)
120 DIM P(7)
130 FOR J = 0 TO 7
140 LET P(J) = 2*(7 - J)
150 NEXT J
160 GOTO 400
200 IF S = -1 THEN LET C(1) = 18
210 LET C(2) = X/10
220 LET C(3) = X - 10* C(2) + 28
230 IF C(2) > 0 THEN LET C(2) = C(2) + 28
240 FOR L=0 TO 6
250 FOR I = 1 TO 3
260 LET V = PEEK (3584 + L + 8*C(I))
270 FOR J = 0 TO 7
280 LET G = (V AND P(J)) > 0
290 PRINT CHR$ (-128*G);
300 NEXT J
310 NEXT I
320 PRINT
330 NEXT L
340 PRINT
350 RETURN
400 LET K = 0
410 LET C(1) = 0
420 LET S=1
430 LET A = RND (49)
440 LET X = A
450 GOSUB 200
460 LET C(1) = 19
470 LET S = 2*RND (2) -3
480 LET B = RND (49)
490 LET X = B
500 GOSUB 200
600 INPUT C
610 IF C = A + S*B THEN GOTO 700
620 LET K = K + 1
630 IF K > 2 THEN GOTO 660
640 PRINT "TENTA DI NUOVO"
650 GOTO 600
660 PRINT "LA RISPOSTA ERA"; A + S*B
664 PRINT
665 PRINT "ANCORA?"
666 INPUT A$
667 IF NOT A$ = "SI" THEN STOP
668 CLS
669 RUN
700 PRINT "VA BENE"
710 GOTO 664

```

Il programma calcola la radice quadrata con 3 decimali di qualunque intero minore di 32767 e positivo. Il metodo iterativo usato nelle linee 200 - 240 risolve i problemi di supero di capacità relativi allo ZX-80. Si usano nomi mnemonici estesi per indicare alcune variabili.





```

10  REM RADICE QUADRATA
20  PRINT "CALCOLO RADICE QUADRATA CON
    3 DECIMALI"
100 PRINT "SCRIVI UN NUMERO"
110 INPUT A
120 CLS
130 PRINT "RADICE QUADRATA DI"; A; " =";
140 LET U = 174
150 LET L = 0
160 LET FRAC = 0
170 LET DELTA = 0
200 LET NEW = (U + L)/2
210 IF (NEW - 1)**2 < A AND (NEW + 1)**2 > A
    THEN GOTO 250
220 IF NEW**2 > A THEN LET U = NEW
230 IF NEW**2 < A THEN LET L = NEW
240 GOTO 200
250 IF NEW**2 > A THEN LET NEW = NEW - 1
300 LET ERROR = 100*(A-NEW**2)
310 IF ERROR < 2 THEN GOTO 470
320 LET FRAC = (ERROR/NEW)/2
330 LET FSQ = FRAC**2
340 LET ERROR = ERROR - 2*NEW*FRAC
350 IF ERROR > 0 THEN GOTO 390
360 LET FRAC = FRAC - 1
370 LET ERROR = 100*(A-NEW**2)
380 GOTO 330
390 LET ERROR = 10 * ERROR - FSQ/10
400 IF ERROR < 0 THEN GOTO 360
410 LET DELTA = (ERROR/NEW)/2
420 LET EDASH = ERROR - (2*FRAC*DELTA)/2
430 IF EDASH > 0 THEN GOTO 460
440 LET DELTA = DELTA - 1
450 GOTO 420
460 LET DELTA = (EDASH/NEW)/2
470 PRINT NEW; " "; FRAC; DELTA

```

SISTEMA DUE EQUAZIONI IN DUE INCOGNITE

Autore: G. Bortone
Programma utilizzante:
1 K di memoria

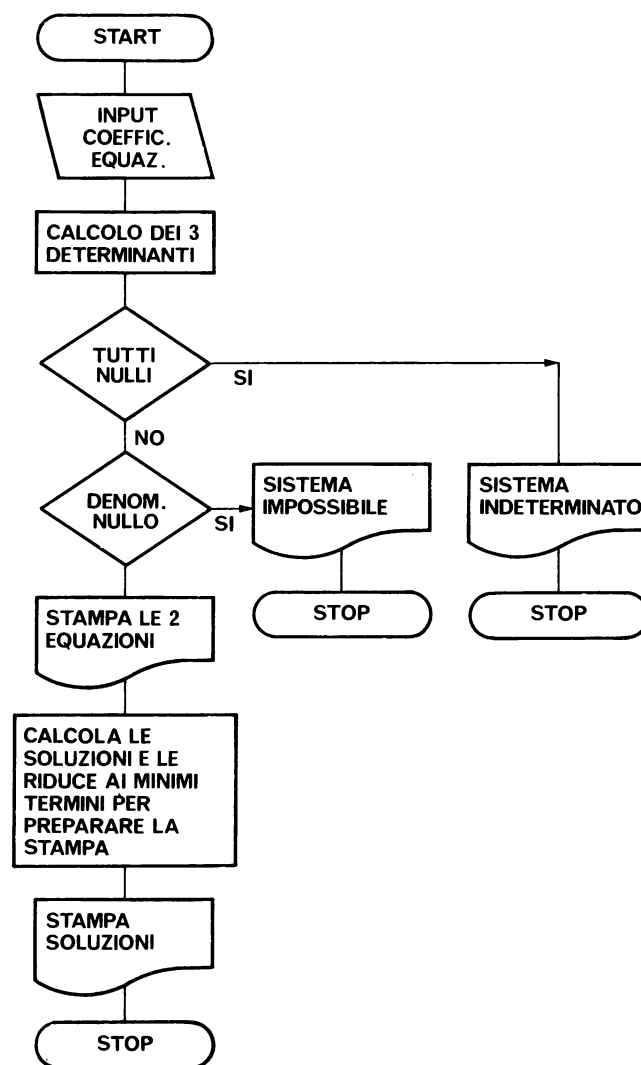
Questo programma risolve un sistema di due equazioni in due incognite di primo grado:

$$AX + BY + C = 0$$

$$DX + EY + F = 0$$

I dati devono essere introdotti nell'ordine A,B,C,D,E,F, ma vengono chiamati A,B,C,A,B,C.

I risultati vengono dati in numero intero o in frazioni ridotte ai minimi termini



Variabili usate: X(6) vettore per i coefficienti
 A(2) vettore per i coefficienti
 D = determinante
 GCD = massimo comune divisore
 I indice
 A,B,Q variabili per il calcolo
 S variabile per il segno

```

10  REM SISTEMA DUE EQUAZIONI
100 DIM X(6)
110 DIM A(2)
115 PRINT
120 PRINT "SISTEMA DI 2 EQUAZIONI DEL TIPO:"
130 PRINT "A*X + B*Y + C = 0"
140 PRINT
150 PRINT "INGRESSO DATI"
160 FOR I = 1 TO 6
170 PRINT CHR$( 37 + I - ((I - 1)/3)*3),
180 INPUT X(I)
190 PRINT X(I)
200 NEXT I
300 LET D = X(1)*X(5) - X(2)*X(4)
320 LET A(1) = X(3)*X(5) - X(2)*X(6)
330 LET A(2) = -X(3)*X(4) + X(1)*X(6)
331 IF D = 0 AND A(1) = 0 AND A(2) = 0 THEN
    GOTO 950
332 IF D = 0 THEN GOTO 900
334 CLS
335 PRINT
336 PRINT "|Δ"; X(1); "X+"; X(2); "Y+"; X(3); "=0"; "Δ|"
337 PRINT
338 PRINT "|Δ"; X(4); "X+"; X(5); "Y+"; X(6); "=0"; "Δ|"
340 PRINT
341 PRINT
360 LET D = -D
370 LET S = D/ABS(D)
400 FOR I = 1 TO 2
410 LET B = ABS(D)
420 LET A = ABS(A(I))
500 LET Q = A/B
510 LET R = A - Q*B
520 LET A = B
530 LET B = R
540 IF R > 0 THEN GOTO 500
550 LET GCD = A
555 PRINT
556 IF I = 1 THEN PRINT "X =";
557 IF I = 2 THEN PRINT "Y =";
600 PRINT S*A(I)/GCD;
610 IF NOT GCD = 1 AND NOT ABS(D/GCD) = 1
    THEN PRINT "/"; ABS(D/GCD)
620 PRINT
630 NEXT I
640 STOP
900 PRINT
901 PRINT
902 PRINT
903 PRINT "NESSUNA SOLUZIONE"
910 STOP
950 PRINT
951 PRINT
952 PRINT
953 PRINT "SISTEMA INDETERMINATO"
955 STOP
  
```

Tagliando ordine libri Jackson da inviare a:
 Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

| | | | |
|--|-------|--|--|
| Nome Cognome | | | |
| Indirizzo | | | |
| Cap. | Città | | |
| Codice Fiscale (indispensabile per le aziende) | | | |

Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° di L.
 (in questo caso la spedizione è gratuita)

| Codice Libro | Quantità | Codice Libro | Quantità | Codice Libro | Quantità |
|--------------|----------|--------------|----------|--------------|----------|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

☐ Non abbonato ☐ Abbonato Data

Firma

N.B. È possibile effettuare versamenti anche sul ccp n° 11666204 intestato a: Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento e non inviare questo tagliando.

Tagliando ordine libri JCE da inviare a:
 JCE - Via dei Lavoratori, 124 - 20092 Cinisello Balsamo (Mi)

| | | | |
|--|-------|--|--|
| Nome Cognome | | | |
| Indirizzo | | | |
| Cap. | Città | | |
| Codice Fiscale (indispensabile per le aziende) | | | |

Inviatemi i seguenti libri:

☐ Pagherò al postino il prezzo indicato nella vostra offerta speciale + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° di L.
 (in questo caso la spedizione è gratuita)

| Codice Libro | Quantità | Codice Libro | Quantità | Codice Libro | Quantità |
|--------------|----------|--------------|----------|--------------|----------|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

☐ Non abbonato ☐ Abbonato

Data Firma

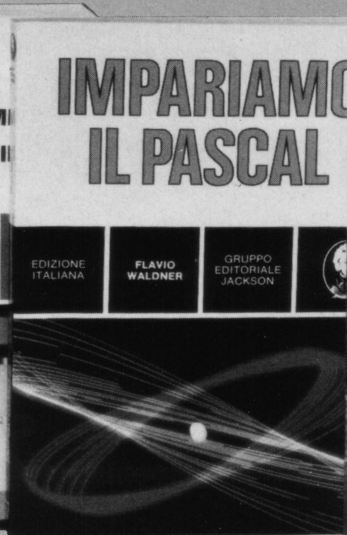
N.B. È possibile effettuare versamenti anche sul ccp n° 315275 intestato a JCE via dei Lavoratori, 124 20092 Cinisello B. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento e non inviare questo tagliando.

IL MEGLIO DELLA

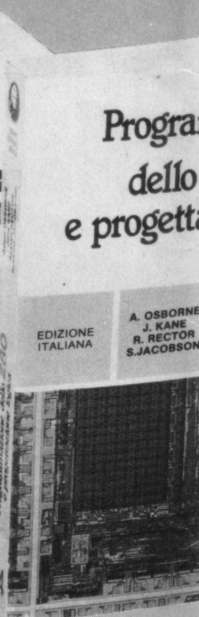
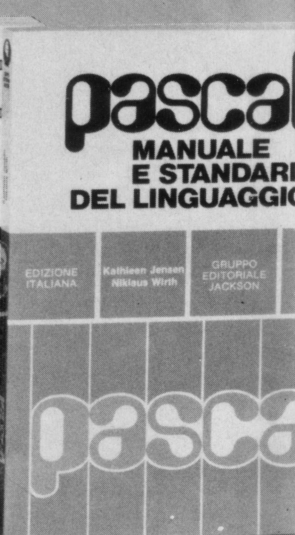
1



2



3



6

7

8

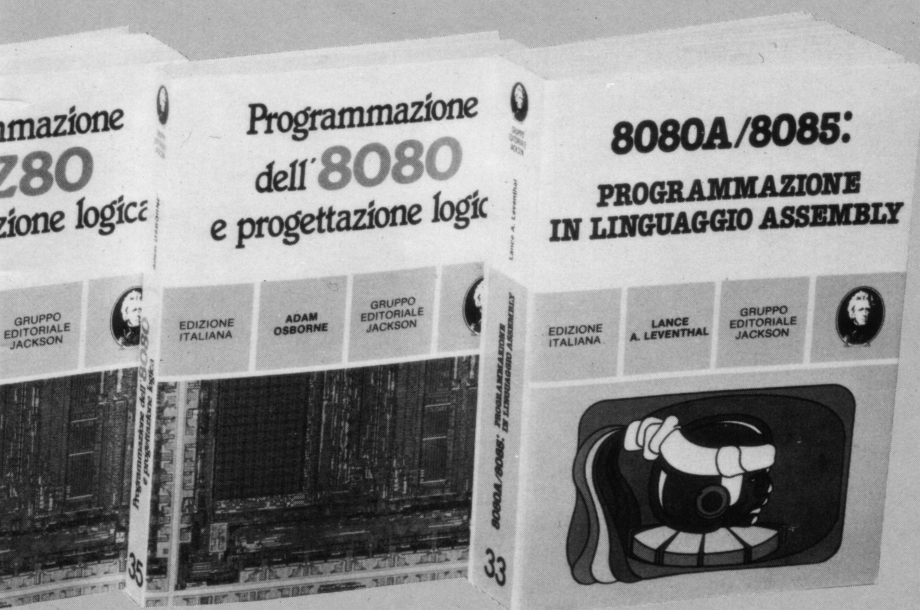
1. Vogliamo incominciare così
Impariamo a programmare in BASIC con il VIC/CBM
L. 11.000 (Abb. 9.900) Cod. 507A
2. Impariamo il PASCAL
L. 10.000 (Abb. 9.000) Cod. 501A
3. PASCAL - Manuale e standard del linguaggio
L. 10.000 (Abb. 9.000) Cod. 500P

4. Impariamo a programmare in BASIC con il PET/CBM
L. 10.000 (Abb. 9.000) Cod. 506B
5. Impariamo a programmare in BASIC con lo ZX80
L. 4.500 (Abb. 4.050) Cod. 317B

PROGRAMMAZIONE

4

5



9

10

6. Introduzione al BASIC
L. 18.500 (Abb. 16.650) Cod. 502A

7. Z80 - Programmazione in Linguaggio Assembly
L. 29.500 (Abb. 26.550) Cod. 326P

8. Programmazione dello ZX80 e progettazione logica
L. 19.000 (Abb. 17.100) Cod. 324P

9. Programmazione dell'8080 e progettazione logica
L. 16.500 (Abb. 14.850) Cod. 325P

10. 8080A/8085 - Programmazione in Linguaggio Assembly
L. 24.000 (Abb. 21.600) Cod. 323P

A ciascuno il suo computer.

Anche voi avete bisogno del computer personale

Tutti hanno sentito parlare di microelettronica e di microprocessori. Molti ne conoscono i vantaggi ma vorrebbero saperne di più molti amerebbero sapere tutto.

Qui si svela che ZX80 è l'apparecchio più importante del nostro tempo. Ciò che molti anni fa era costosamente consentito solo ai grandi organismi, ora è alla portata di tutti; del professionista, della piccola azienda, del nucleo familiare, persino della persona singola.

Lo ZX80 della Sinclair offre servizi di gran lunga superiori al suo prezzo. Pesa solo 350 grammi. È applicabile a qualunque televisore.

Può essere collegato a un registratore di cassette per la memorizzazione permanente di istruzioni e dati.

È un piccolo apparecchio che può mettere ordine in tutte le vostre cose e aiutarvi più di una schiera di segretari.

Il primo computer personale veramente pratico

ZX80 anticipa i tempi. Le sue qualità colgono di sorpresa anche i tecnici, poichè il raggiungimento delle caratteristiche che lo distinguono sarebbero dovute apparire fra molto tempo.

È conveniente, facile da regolare, da far funzionare e da riporre dopo l'uso. Soddisfa l'utente più preparato.

Esempio di microelettronica avanzata

La semplicità circuitale è il primo pregio dello ZX80, la potenza è il secondo pregio. Insieme, ne fanno l'apparecchio unico nel suo genere.

Alcune applicazioni

A casa memorizza i compleanni, i numeri telefonici, le ricette di cucina, le spese e il bilancio familiare, e altre mille applicazioni di cui si può presentare la necessità.

Per aziende

Piccole gestioni di magazzino, archivio clienti e fornitori eccetera.

Per professionisti

Calcoli matematici e trigonometrici, elaborazione di formule, archivio.

Per il tempo libero

Lo ZX80 gioca alle carte, risolve le parole incrociate, fa qualsiasi gioco gli venga messo in memoria.



sinclair ZX80



Chiedere opuscolo illustrato a:
GBC Italiana, casella postale 10488 Milano
Dimostrazioni e vendita presso i

CARATTERISTICHE TECNICHE

MICRO - Z80A
LINGUAGGIO - BASIC
MEMORIA - 1 K RAM ESPANDIBILE A 16 K
TASTIERA - KEYPLATE CON SUPERFICIE STAMPATA
VISUALIZZAZIONE - SU QUALUNQUE TELEVISORE
GRAFICA - 24 LINEE A 32 CARATTERI
MEMORIA DI MASSA - SU QUALUNQUE REGISTRATORE
MAGNETICO

BUS - CONNETTORE CON 44 LINEE, 37 PER CPU 0V, 5V, 9V, CLOCK
SISTEMA OPERATIVO - 4K ROM
ALIMENTAZIONE - 220V, 50 Hz CON ALIMENTATORE ESTERNO
OPZIONALE

LISTINO PREZZI IVA ESCLUSA

| | |
|--|-----------------------|
| • COMPUTER ZX80 | TC/0080-00 L. 285.000 |
| • COMPUTER ZX80 KIT | TC/0081-00 L. 240.000 |
| • MODULO PER ESPANSIONE DI MEMORIA FINO A 3K RAM | TC/0083-00 L. 39.500 |
| • COPPIE DI CIRCUITI INTEGRATI PER OGNI K DI MEMORIA | TC/0082-00 L. 17.000 |
| • ALIMENTATORE | TC/0085-00 L. 12.900 |
| • LIBRO "IMPARIAMO A PROGRAMMARE IN BASIC CON LO ZX80" | TL/1450-01 L. 4.400 |
| • MODULO DI ESPANSIONE DI 16 K RAM COMPLETO DI INTEGRATI | TC/0087-00 L. 191.500 |
| • ALIMENTATORE PER ZX80 CON ESPANSIONE DI 16 K RAM | TC/0086-00 L. 22.000 |
| • FLOATING POINT ROMS 8K | TC 0088-00 L. 60.000 |

