ALLA SCOPERTA DEL BASIC SPECTRUM

Mike Lord

EDIZIONE ITALIANA

GRUPPO EDITORIALE JACKSON

ALLA SCOPERTA DEL BASIC SPECTRUM

Mike Lord

GRUPPO EDITORIALE JACKSON
Via Rosellini, 12
20124 Milano

TRADUTTORE: Carlo Tognoni
COPERTINA: Silvana Corbelli
GRAFICA E IMPAGINAZIONE: Francesca Di Fiore
COORDINAMENTO EDITORIALE: Daria Gianni
FOTOCOMPOSIZIONE: CorpoNove - Bergamo
STAMPA: Grafika '78 - Via Trieste, 20 - Pioltello (MI)

SOMMARIO

INTRODUZIONE

Benvenuti, amici esploratori, nel mondo dello Spectrum.

Che mondo affascinante è mai questo! Un mondo di cui potrete avere il controllo totale. Questo libro tratta il linguaggio BASIC usato dal Sinclair Spectrum e completa, in tutto e per tutto, l'ottimo manuale in dotazione allo Spectrum stesso:

- esaltando alcuni concetti fondamentali della programmazione in BASIC che, secondo l'esperienza dell'autore, i neofiti stentano ad apprendere;
- esplorando le caratteristiche principali del BASIC dello Spectrum come l'alta risoluzione grafica, il suono e i colori;
- mostrando come lo Spectrum possa essere usato per i giochi, affari e altre applicazioni ancora;
- provvedendo a fornire un valido supporto di programmi per illustrare i punti di cui sopra;
- tendendo ad essere una fonte di idee per creare programmi vostri.

Se la vostra attenzione è rivolta ai programmi dei giochi, a quelli degli interessi personali o ancora a quelli di applicazioni ingegneristiche, ecco il libro che fa per voi.

Mi auguro, comunque, che lo stimolo più forte che trarrete da queste pagine sarà appunto quello di voler comporre nuovi programmi, addentrandovi nei misteri di questa meravigliosa macchina.

MIKE LORD

Alcune precisazioni circa i programmi riportati in questo libro:

- tutti girano sullo Spectrum da 16 K, sebbene alcuni possano risultare più utili se ne avete uno da 48 K;
- il numero zero è scritto "Ø" per distinguerlo dalla lettera "0" nei listati del programma;
- fate attenzione a non confondere il numero 1 con la lettera minuscola "l". In caso di dubbio confrontatelo con il numero 1 che appare almeno in un numero di riga del listato;
- le keyword (cioé le parole riservate del BASIC) sono state scritte interamente in maiuscolo nel testo:

  RUN GO TO PI

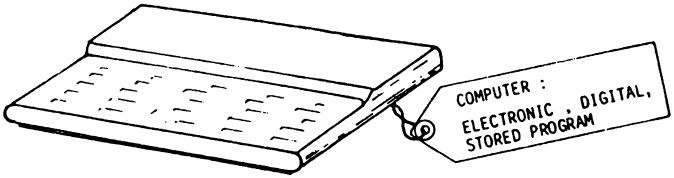
  ricordate che vengono ottenute sullo Spectrum premendo un solo tasto;
- anche le funzioni "<=", ">=" e "< >" sono ottenute battendo un singolo tasto;
- i caratteri grafici standard sono stati scritti come appaiono sullo schermo; il simbolo "□" rappresenta il carattere del tasto 6. Qualora vi fossero delle difficoltà nel riconoscere i caratteri o siano stati impiegati caratteri definiti dall'utente, il listato del programma suggerirà quale carattere usare:
- lo Spectrum ignora gli spazi, a meno che questi non siano tra virgolette.

CAPITOLO 1
PARTENZA!

Imparare ad usare il computer è pressapoco come iniziare a orientarsi in una città sconosciuta. Ciò che importa all'inizio sono le vie principali. Ma più tardi dovrete esplorare le vie laterali e le altre zone della città. Questo libro vuole essere una scorciatoia per raggiungere in breve tempo la via desiderata.

Se volete evitare di perdervi troppo spesso è utile avere una carta topografica della città o una guida che vi possa segnalare le cose che meritano di essere viste, ma è il desiderio di esplorare e di fare esperienza che vi devono spingere a continuare da soli.

COMPUTER?



La parola "computer", può venire attribuita ad una grande quantità di dispositivi a partire dal semplice pallottoliere fino a raggiungere il paranoico HAL del film "2001 Odissea nello spazio". La caratteristica comune è quella di essere macchine in grado di manipolare dati. Così, come l'addetto ad un deposito di legname manipola il legno, selezionandolo, accumulandolo e perfino cambiandone le dimensioni, nello stesso modo il computer manipola i dati, che possono essere prezzi di prodotti, elenchi di numeri di targa di macchine rubate o qualsiasi altra cosa.

Possiamo definire un po' meglio lo Spectrum dicendo che è un "calcolatore elettronico digitale programmabile" che cioé manipola i dati sotto forma di cifre. I computer "analogici", per contro, lavorano direttamente con il valore delle variabili, non con i numeri che rappresentano questi valori. Lo Spectrum accetta lettere dell'alfabeto e altri simboli che al suo interno vengono rappresentati da numeri.

Dicendo che lo Spectrum è "programmabile" abbiamo sottolineato una caratteristica molto importante. Per programma intendiamo un elenco di istruzioni che noi diamo al computer per fargli fare ciò che vogliamo. Il punto cruciale è che si può memorizzare nello Spectrum quella lista di istruzioni perché venga eseguita quando si vuole.

Per scrivere, ad esempio, un programma atto a fargli risolvere problemi matematici, dobbiamo inserire le istruzioni ad una ad una. Il computer le accetterà, sequenzialmente, ponendosi ogni volta in attesa dell'istruzione successiva. Una volta caricato l'intero programma, il computer lo leggerà rapidissimamente ripetendo più volte e automaticamente, i "loop" (gli anelli) di istruzione insiti nella lista. Tra le altre cose, il computer può rileggersi l'intero programma senza che dall'esterno gli venga dato alcun comando specifico; così come può registrarlo in una memoria di massa esterna, quale può essere un nastro

magnetico o un disco. Tale prerogativa risulta assai utile per poter richiamare i programmi, in caso di necessità, che possono essere modificati o corretti a piacimento senza dover stendere nuovamente l'intero listato. Con la procedura di 'Editing' è possibile infatti cambiare anche una singola istruzione lasciando inalterato il resto del programma.

COS'È UNA MEMORIA?

Come si può vedere sul manuale di programmazione, lo Spectrum possiede due tipi di memoria: la ROM e la RAM. Per aiutarvi a capire che cosa siano, è necessario dare una rapida occhiata a quello che è il cuore della macchina, ovvero il chip del microprocessore Z80. Si tratta di un integrato veramente complesso, che svolge i calcoli richiesti durante lo sviluppo del programma, in un tempo minimo dell'ordine di un milione di istruzioni al secondo, ma che non è in grado di capire il linguaggio BASIC. Le istruzioni gli vengono infatti presentate molto più dettagliatamente sottoforma di "linguaggio macchina".

La ROM (Read Only Memory = Memoria a sola lettura) contiene numerose routine scritte in linguaggio macchina, routine che, per tutto il tempo che il computer resta acceso, vengono esplorate dallo Z80 per permettere l'esecuzione delle logiche principali, che sono:

- permettere l'accettazione e l'editing del vostro programma;
- far girare il programma in BASIC. A questo scopo la ROM agisce da interprete trasformando le varie istruzioni inserite, in BASIC, dalla tastiera in linguaggio macchina;
- provvedere alla registrazione dei programmi su unità esterne o caricarle da queste nella memoria locale;
- fornire i segnali di scrittura per il video o la stampante.

Tutte queste informazioni vengono inserite nel chip della ROM direttamente dal costruttore e non possono venire né cancellate né modificate (da qui il nome ROM) ma solamente lette.

La RAM (Random Access Memory = Memoria ad accesso casuale)

invece, è molto più flessibile. Rende possibile, oltre alla lettura, anche l'alterazione dei dati in essa memorizzati scrivendone di nuovi.

L'unico svantaggio delle RAM è la volatilità del loro contenuto che va irrimediabilmente perso ogniqualvolta venga a mancare l'alimentazione. Come riportato nel capitolo 24 del manuale, essa è suddivisa in diverse aree, le più importanti delle quali sono:

- l'area relativa al display e ai caratteri, che viene letta continuamente da un circuito per generare il segnale video. Contiene campioni dei caratteri e dei simboli che appaiono sullo schermo accompagnati dalle informazioni (attributi) relative al colore e alla luminosità delle varie zone dell'immagine;
- l'area destinata alle variabili di sistema e ai diversi stack usati dallo Spectrum per tenere nota dell'attività del computer in ogni momento;
- l'area del BASIC accoglie i programmi inseriti dall'utente;
- l'area delle variabili rimane a disposizione per le variabili di programma.

Bit, Byte e K

I neofiti del computer potrebbero rimanere assai perplessi dinanzi al modo in cui viene misurata la memoria: tutto ciò che si riferisce ai "byte" e ai "K" a prima vista appare piuttosto confuso. La spiegazione in realtà è molto semplice; la memoria è organizzata in modo tale da adattarsi alle esigenze del microprocessore che, essendo un dispositivo binario, si trova a suo agio solo con le potenze di 2. Possiamo immaginare il banco di memoria di un computer come una lunga fila di cellette identiche, ciascuna delle quali numerata con numeri crescenti, in modo che il microprocessore possa usare tali numeri (o "indirizzi") per selezionare quella che gli interessa. In ogni locazione ("celletta") risiede un numero binario formato da 8 bit (Binary digit) ognuno dei quali può assumere il valore "0" o "1". Il gruppo degli 8 bit, è conosciuto col nome di "byte". Se ne deduce che sono possibili 256 (2 elevato all'ottava) combinazioni degli otto "1" o "0" per cui ogni locazione può racchiudere un numero intero decimale compreso tra 0 e 255. Lo Spec-

trum, comunque, può manipolare sia numeri posti al di fuori di questa gamma, sia frazioni, usando un gruppo di 5 byte come descritto nel capitolo 26 del manuale. Il microprocessore Z80 può elaborare in totale 65536 (2 elevato alla sedicesima) differenti locazioni di memoria, il primo quarto delle quali prelevabili dalla ROM e le restanti, a seconda del modello, possono essere parzialmente o totalmente prelevate dalla RAM. Una abbreviazione idonea per esprimere la quantità di memoria, è la lettera "K" che sta per 1024 (2 elevato alla decima). Lo Spectrum possiede complessivamente 64 Kbyte di memoria dei quali 16 K destinati alla ROM e 48 K alla RAM.

BASIC

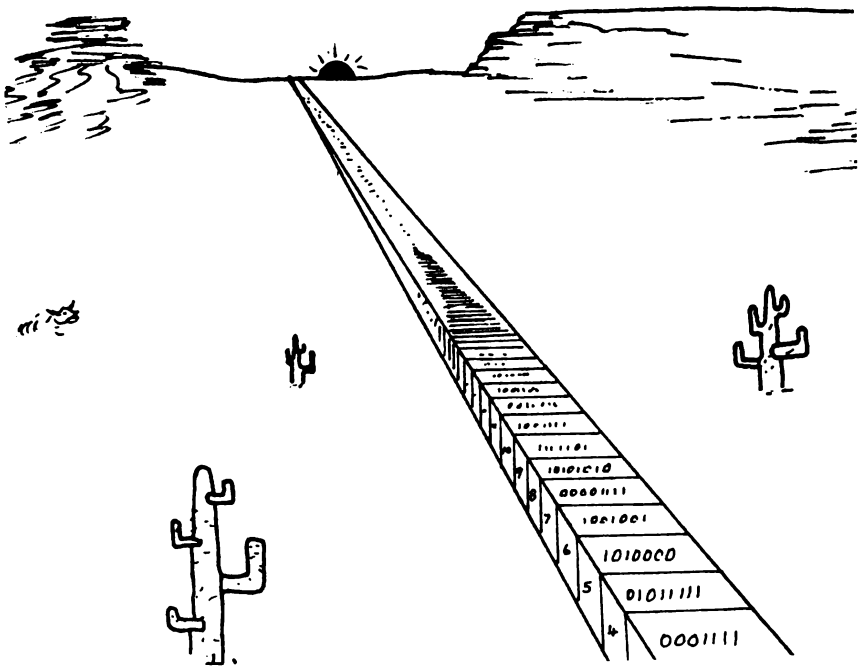
La parola BASIC è un acronimo di "Beginners All Purpose Symbolic Instruction Code" e si riferisce ad un linguaggio sviluppato al Dartmouth College (U.S.A.) negli anni '60 per consentire un facile approccio all'informatica. Altri linguaggi, più complessi e sofisticati come quelli sottoelencati non sarebbero alla portata dei neofiti:

- FORTRAN: uno dei primi linguaggi; è adatto alla gestione di problemi matematici: molti programmi scientifici e di ingegneria vengono ancora scritti in questo linguaggio;
- COBOL: è un linguaggio vecchio ma molto potente per applicazioni commerciali; è impiegato sui grossi computer;
- PASCAL: simile al più vecchio "Algol", è nato con caratteristiche "accademiche", ma è via via cresciuto in popolarità fino ad ottenere serie applicazioni sui microcomputer. È un linguaggio formale strutturato, idoneo a manipolare tipi diversi di dati.
- LISP: è uno strano linguaggio sviluppato nell'ambito delle ricerche sull'intelligenza artificiale. Di fatto è un linguaggio per il trattamento delle stringhe e il suo apprendimento implica cambiare il modo di vedere i problemi;
- FORTH: molto apprezzato da alcuni, è profondamente odiato da altri. Le sue principali caratteristiche sono la totale incomprensibilità

per chi non sia un abile programmatore in FORTH e la sua capacità di definire nuove istruzioni.

Come tutti gli altri linguaggi, anche il BASIC vuole essere una via di mezzo tra il linguaggio umano e quello assai restrittivo della macchina. La trasformazione del BASIC in linguaggio macchina spetta alle routine dell'"Interprete BASIC" che risiede nella ROM, mentre la traduzione del programma dal linguaggio umano al BASIC è a carico vostro.

Il BASIC è disponibile su quasi tutti i computer e probabilmente è il linguaggio più usato. Tuttavia, curioso a dirsi, tutti i tentativi di standardizzarlo sono falliti. Sembra che ogni computer abbia la sua versione del BASIC, con dei potenziamenti spesso utili e comunque sostenuti da una pubblicità sfrenata. Il BASIC dello Spectrum, abbastanza potente, si avvale di un vocabolario di circa 90 parole che vengono usate per formare gli statement. Ogni statement (o frase di programma) è un'istruzione da inviare al computer per fargli svolgere determinate funzioni. Più statement posti in successione formano il programma BASIC.



CAPITOLO 2
PRINT E PROGRAMMI

La parola PRINT (scrivi) è molto importante nel linguaggio BASIC in quanto rappresenta per il neofita un punto di partenza ideale mostrando immediatamente quello che fa.

Accendete ora il vostro Spectrum e vedrete apparire verso il basso dello schermo la scritta inerente al copyright. Battete ora:

```
PRINT 123
```

(La parola PRINT viene inserita semplicemente azionando il tasto P). In questa fase vedrete visualizzato quanto avete battuto sullo schermo in basso, seguito dal cursore lampeggiante. Questa posizione della scritta sta ad indicare che lo Spectrum lavora in Editing, vale a dire che è in attesa che siate sicuri dell'esattezza di quanto battuto. Quando siete soddisfatti di ciò che avete battuto, premete ENTER. Nel nostro caso, sparirà la scritta presente verso la parte bassa del display per far posto a

```
123
```

nell'angolo in alto a sinistra, che non è altro che il risultato di quanto chiesto. In basso a sinistra ecco ora

```
0 OK, 0:1
```

che costituisce il 'rapporto' dello Spectrum il quale avverte di aver eseguito il programma senza errori. Provate ad inserire un altro statement di PRINT, ad esempio

```
PRINT 456
```

e vedrete apparire, dopo aver dato l'ENTER, il numero 456, sempre sulla sinistra dello schermo, appena al disotto del 123.

Questo dimostra che il risultato del PRINT precedente rimane anche eseguendo PRINT successivi. Per cancellare tutto, è necessario, in questo caso, dare un secondo ENTER. Provate.

Risulta anche evidente che ogni PRINT usa una nuova linea di schermo. Questo, però, non è del tutto vero; la posizione di scrittura viene in realtà determinata dal computer in modo più flessibile, perché lo Spectrum si ricorda il punto in cui ha scritto l'ultimo carattere. A schermo libero, la prima posizione di scrittura risulta, come abbiamo appena visto, nell'angolo in alto a sinistra e, con semplici PRINT, si sistema via via all'estrema sinistra di ogni nuova riga.

■
,

È possibile evitare questo salto a nuova riga cambiando leggermente lo statement PRINT. Inserite i seguenti comandi:

```
PRINT 12;
PRINT 34
```

Come risultato si otterrà, nell'angolo a sinistra, il numero 1234; infatti il punto e virgola alla fine del primo statement PRINT dice allo Spectrum di non andare a capo ma di ricominciare a scrivere dal punto in cui è, cioé dopo il 2. Così nei quattro comandi che seguono:

```
PRINT 0;
PRINT 12;
PRINT 34;
PRINT 5
```

ogni punto e virgola posto alla fine degli statement stabilisce la posizione della cifra successiva.

,

Se al posto del punto e virgola, mettiamo una virgola (,) variamo la posizione di scrittura spostandola:

— al centro della linea se il carattere precedente si trova prima del centro;

— a capo sulla linea seguente, se l'ultimo carattere si trova al centro od oltre.

Questa accortezza ci permette di scrivere su due colonne ben distinte; provate ad inserire i quattro comandi.

```
PRINT 0,
PRINT 12,
PRINT 34,
PRINT 5
```

Separatori

Dopo la parola PRINT possiamo inserire anche più di un numero, avendo cura di separare le cifre per mezzo di una virgola o di un punto e virgola: una virgola porterà la posizione di scrittura all'inizio della riga o al suo centro:

```
PRINT 1,23,456,7
```

mentre il punto e virgola significa che la posizione di scrittura non deve essere spostata:

```
PRINT 1;23;456;7
```

Questo potrebbe sembrare inutile, in quanto è possibile ottenere lo stesso risultato nel modo seguente:

```
PRINT 1234567
```

Ma vedremo più avanti come il punto e virgola sia un segno separatore importantissimo quando si scrivono cose più complesse che numeri.

Un terzo separatore è l'apostrofo (') che trovate riportato in rosso sul tasto 7. Esso sposta la posizione di scrittura all'inizio della linea seguente; provate a battere:

```
PRINT 1'23''456'7
```

tenendo presente che ogni singolo apostrofo alla fine di uno statement di PRINT viene ignorato.

TAB

Questa parola permette la scrittura in qualsiasi posizione lungo una riga. Più precisamente "TAB n" sposta in avanti di 'n' colonne la posizione di scrittura, essendo le colonne numerate da 0 (la prima a sinistra) a 31 (l'ultima sulla destra). Se la posizione di scrittura ha già superato la colonna 'n', si sposterà di 'n' colonne sulla linea seguente.

Se invece 'n' supera 31, lo Spectrum sottrae automaticamente da esso il valore 32 fino a che 'n' non rientri nella gamma 0/31.

TAB n può essere inserito in uno statement di PRINT e deve essere seguito da un senso separatore, di solito il punto e virgola, in quanto qualsiasi altro segno annulla l'effetto del TAB. Provate a battere:

```
PRINT 1;TAB 2;2;TAB 1;3
```

AT

TAB muove la posizione di scrittura entro una riga, partendo dal punto in cui questa si trovava in precedenza. AT y,x è molto più potente poiché può spostare la posizione di scrittura su qualsiasi colonna (x) e su qualsiasi riga (y). I valori di x e y vanno sempre separati da una virgola ed anche qui le colonne sono numerate da 0 a 31. Per quanto riguarda le linee, lo Spectrum ne visualizza 24 ma le due situate in basso sono riservate ai messaggi del computer e ai vostri input; in tal modo il numero delle linee utili cala a 22 (0 per la prima in alto, fino a 21).

AT y,x come già per TAB, può essere inserito in uno statement di PRINT seguito dal punto e virgola. Digitate

```
PRINT AT 10,0;10;AT 9,1;9
```

e qualsiasi altra cosa vi venga in mente.

Nello stesso statement di PRINT possono essere mescolate combinazioni di numeri, segni separatori, funzioni TAB e AT, considerando come unica regola che un segno separatore è inserito fra ciascuna coppia di parole e che i valori associati a AT e TAB devono essere significativi.

Provate, dopo aver dato ENTER per pulire lo schermo, ad inserire

```
PRINT 1;AT 21,31;2;AT 10,15;3;TAB 31;4,5
```

Semplice Editing

L'esempio precedente era piuttosto complicato e, probabilmente, quando avete battuto l'ENTER, sullo schermo dello Spectrum è apparso in tutta risposta soltanto un punto interrogativo lampeggiante in campo nero in qualche punto della riga. Questo significa che lo Spectrum ha trovato un errore di *sintassi* su quella riga. Il punto interrogativo è posto dove si presume sia stato commesso l'errore, ma non sempre lo Spectrum ci "azzecca", quindi è meglio controllare l'intera riga.

Se, tuttavia, siete stati molto precisi, lo Spectrum ha accettato il vostro messaggio, quindi potete provare a sbagliare volontariamente battendo:

```
PRINT AT 2;3;4
```

Questa volta è sicuramente sbagliato!

Avendo commesso un errore (dopo la cifra 2 ci sarebbe voluta una virgola e non un punto e virgola) bisogna procedere alla correzione. E qui entra in gioco il cursore lampeggiante contenente la lettera L. Nello stesso modo in cui i caratteri vengono inseriti, ogni volta alla sinistra del cursore con la L, questi possono venire anche cancellati, digitando contemporaneamente CAPS SHIFT e DELETE.

In questo modo potete correggere l'errore cancellando a ritroso i caratteri, fino a raggiungere quello incriminato e quindi ribattere nel modo esatto quanto cancellato. Ma c'è un sistema più semplice, in quanto la tastiera prevede due tasti per il movimento del cursore: il 5 (⇦) per lo spostamento a sinistra e l'8 (⇨) per quello verso destra.

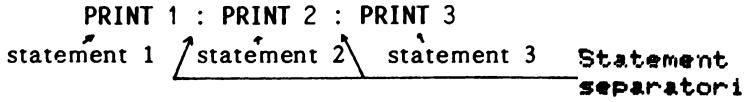
Premendo CAPS SHIFT e il tasto 5 (⇦) contemporaneamente, sposterete il cursore di uno spazio a sinistra; analogamente, premendo CAPS SHIFT e il tasto 8 (⇨), lo muoverete di uno spazio a destra.

In questo modo potete portare il cursore nella posizione immediatamente alla destra del simbolo errato, senza essere costretti a cancellare tutto ciò che si trova sulla destra di questo.

Fatto ciò, la linea può venir fatta accettare dallo Spectrum con ENTER, senza necessariamente riportare il cursore con la L alla fine della linea. Se la linea stessa risultasse piena di errori, è possibile cancellarla completamente battendo contemporaneamente CAPS SHIFT e EDIT (tasto del numero 1).

Statement Multipli

Nelle pagine precedenti siamo giunti a scrivere fino a quattro statement disposti su altrettante linee, ma è possibile, per risparmiare spazio, inserirli sulla stessa riga separandoli con due punti (:)



In questo modo lo Spectrum eseguirà innanzitutto il primo statement, quindi il secondo e così via fino al termine della riga. Un comando assai usato è CLS, che cancella lo schermo e sposta la posizione di scrittura nell'angolo in alto a sinistra. Possiamo facilmente combinarlo con lo statement di PRINT nel modo seguente:

```
CLS:PRINT 123
```

Nel caso, invece, che il vostro comando fosse il seguente:

```
PRINT 123: CLS
```

probabilmente non avreste il tempo di vedere il 123 sullo schermo prima che venga cancellato.

Gli statement multipli pongono la domanda: "quanto può essere lunga una riga"? Per quanto riguarda il *video*, ogni riga è ovviamente limitata a 32 caratteri.

Sovrapposizione

Ci si può chiedere che cosa accade scrivendo in una posizione dello schermo già occupata da un altro messaggio. Provate ad esempio la seguente istruzione:

```
PRINT AT 0,0;1234;AT 0,0;0
```

darà come risultato

```
0234
```

In questo caso, lo 0 si sistema al posto della prima cifra (1) senza influenzare le rimanenti. Aggiungendo, invece, una virgola alla fine dello statement, otterremo come risultato

```
0
```

in quanto la virgola sposta la posizione di scrittura al centro della riga riempiendola di spazi; il discorso diventa molto più chiaro battendo le due righe seguenti:

```
PRINT AT 0,13;123456
PRINT AT 0,13;0,
```

Il risultato sarà:

```
0 456
```

Tale metodo si rivela molto comodo quando sia necessario cancellare una riga interamente o a metà:

```
PRINT AT 16,0,,
```

cancella l'intera linea 16.

Lo stesso discorso vale per la funzione TAB, come si può vedere dall'esempio che segue:

```
PRINT AT 0,0;123456
PRINT AT 0,1;TAB 3;0
```

in questo caso, il risultato sarà:

```
1 056
```

È possibile quindi cancellare una parte di linea:

```
PRINT AT 16,10;TAB 14
```

elimina le colonne da 10 a 13 della riga 16. Si noti che la funzione AT e l'apostrofo (') influiscono sul display solamente spostando la posizione di scrittura, esattamente come l'"a capo" prodotto da un'istruzione PRINT che non termini con una virgola o con un punto e virgola.

Un programma

Fino ad ora avete fatto eseguire allo Spectrum, premendo il tasto ENTER, i comandi in modo immediato, a cui il computer risponde presentando il risultato e cancellando la linea di comando. Vediamo ora come il computer memorizza i programmi.

La differenza tra una linea di comando diretto ed una linea di programma è che quest'ultima possiede un numero chiamato numero di riga e compreso tra 1 e 9999. Il numero di linea, oltre a comunicare allo Spectrum che quella introdotta è una riga di programma, serve anche per farvi riferimento.

Spegnete lo Spectrum per pochi secondi, riaccendetelo, aspettate la scritta di copyright e quindi battete:

```
20 PRINT 12345
```

Date l'ENTER e vedrete che questa volta non si presenterà la scritta 12345, ma la copia della linea appena battuta si trasferirà nella parte superiore dello schermo. In questo modo, avete inserito in memoria una linea di programma senza aver chiesto al computer di eseguirla; per far ciò dovete battere il comando diretto:

```
RUN
```

(tasto R) seguito, come al solito, da ENTER. Ecco che nell'angolo in alto a sinistra del display, apparirà 12345 ed in quello basso, sempre a sinistra:

```
0 OK, 20:1
```

il quale rivela che il programma è stato eseguito correttamente (OK) ed è terminato alla linea 20, statement 1.

Azionate nuovamente ENTER e la linea di programma riapparirà, premete nuovamente RUN e il programma verrà di nuovo eseguito; potete continuare all'infinito. Digitate ora quest'altra riga.

```
10 PRINT 0
```

date ENTER ed ecco che la nuova linea viene inserita prima della precedente. Ciò è dovuto al suo più basso numero di riferimento, infatti è lo

stesso computer ad ordinare in modo crescente le linee senza tener conto della sequenza con la quale queste vengono introdotte. Battete una terza linea:

```
30 PRINT 6
```

e date il RUN: sullo schermo, in alto, apparirà:

```
0
12345
6
```

a dimostrazione del fatto che lo Spectrum esegue sequenzialmente le linee di programma in base all'ordine del numero di riga.

Editing di Programma

Come visto in precedenza, è assai facile modificare il contenuto di una linea di programma quando questa è scritta nella parte bassa dello schermo, ma come fare nel caso in cui sia già stata trasferita in memoria? Prendiamo il caso delle tre linee di programma appena viste:

```
10 PRINT 0
20 PRINT 12345
30 PRINT 6
```

sono già in memoria; ora, per esempio, battete:

```
20
```

Dopo aver battuto ENTER apparirà una nuova versione del programma, priva della riga 20, essendo stata cancellata con l'operazione appena eseguita. Pertanto, volendo eliminare una riga dal listato, sarà sufficiente battere il numero e dare l'ENTER, senza aggiungere altro. Eseguite ora

```
20 PRINT 3
```

ed il programma tornerà sì ad essere di tre linee, ma la linea 20 non è più quella di prima. Quest'ultima potrà essere però reinserita battendo nuovamente

```
20 PRINT 12345
```

Ecco riottenuto il programma nella sua primitiva versione.

Avrete sicuramente notato, a questo punto, la presenza del simbolo ">" accanto ad una delle linee. È il cursore "Line Editor" che può essere spostato di linea in linea per mezzo dei seguenti tasti:

- CAPS SHIFT assieme al 6 (⇩) per lo spostamento verso il basso, alla successiva linea di programma;
- CAPS SHIFT assieme al 7 (⇧) per lo spostamento verso l'alto, alla linea precedente.

Portatelo, ad esempio, in corrispondenza della linea 10 e premete CAPS SHIFT assieme al tasto 1 (EDIT). La linea interessata apparirà all'istante nella parte bassa dello schermo.

A questo punto potrà essere modificata a piacere e reinserita immediatamente, semplicemente dando l'ENTER. Il cursore > può anche venir spostato direttamente su una particolare linea per mezzo del comando diretto LIST.

```
LIST 20
```

porta il cursore alla linea 20. Questo comando si rivela molto utile in programmi lunghi. Listando una linea inesistente, il cursore scompare come potete constatare inserendo ad esempio

```
LIST 13
```

ma può venire richiamato o listando una linea esistente, oppure battendo CAPS SHIFT unitamente ai tasti 6 o 7. Se volete cancellare l'intero programma, togliete l'alimentazione al computer per qualche secondo, per poi riaccenderlo, oppure battete semplicemente il comando diretto

```
NEW                                    (tasto A)
```

Nell'introdurre programmi di una certa lunghezza, il listato non entrerà più nel quadro del video, ma non spaventatevi, la parte esclusa non viene persa poiché lo Spectrum possiede un'area di memoria sufficientemente vasta per far fronte alle vostre necessità.

Stringhe

Fino ad ora abbiamo scritto solamente numeri, ma è ovvio che è possibile scrivere qualsiasi simbolo o carattere presente sulla tastiera. L'unico accorgimento da tener ben presente è che la scritta, di qualsiasi natura essa sia, va racchiusa entro due apici o virgolette (" visibili in rosso sul tasto P).

Vedremo la ragione di ciò nel capitolo seguente, ora accontentiamoci di classificare qualsiasi cosa presente entro una coppia di apici come una "stringa di caratteri" che verrà stampata così come è stata inserita in fase di programmazione.

Provate a battere la stringa che segue, formata esclusivamente da lettere:

```
PRINT "POSSIBILITA' DELLO SPECTRUM"
```

e poi quest'altra formata solo di numeri e simboli:

```
PRINT "1+1=5"
```

Come avete visto, lo Spectrum non riconosce la validità o meno di quanto racchiuso tra virgolette, ma stampa tale e quale tutto quanto compreso in quell'area. Inserendo:

```
PRINT "Sono""andato""a casa"
```

la parola "andato" si presenterà tra virgolette nell'ambito dell'intera frase compresa tra il primo e l'ultimo apice. È possibile battere più stringhe nello stesso PRINT separandole dai segni (;) oppure (,) oppure (') come ad esempio:

```
PRINT "Oggi e'""'giovedi'"," giorno ";12
```

Le stringhe possono comprendere anche caratteri grafici e scritte in negativo (bianco su nero), ottenibili usando i tasti INV VIDEO e TRUE VIDEO.

LPRINT, LLIST e COPY

Se possedete una stampante, potete provare gli esempi sin qui fatti sostituendo la parola PRINT con LPRINT (sopra al tasto C). Tenete pre-

sente che in questo caso, le parti di linea sottoposte alla funzione AT, non vengono stampate per cui interviene la necessità di rimpiazzare detta funzione inserendo una linea successiva. Il comando LLIST (sopra il tasto V) abilita alla stampa dell'intero programma, mentre LLIST n stampa tutte le linee successive alla n. Il comando COPY è assai usato in quanto stampa unicamente la copia delle 22 linee visualizzate sullo schermo con una qualità migliore di quella resa dal LLIST per battere l'intero listato.

CAPITOLO 3
VARIABILI, INPUT E ESPRESSIONI

Negli esempi del capitolo precedente abbiamo imparato come scrivere delle istruzioni che consentono di visualizzare lettere o numeri fino ad ora stampati possono essere considerati delle "costanti" in quanto composti:

- da numeri come ad esempio 123 oppure 747;
- da stringhe come può essere "Spectrum" (Il termine "literal" sarà a volte usato come "costante").

Nello scrivere un programma non sempre, però, è possibile conoscere a priori il valore di tutti i numeri che verranno usati. Ad esempio, se ad un certo punto il programma chiedesse il nome e l'età dell'utente, noi dobbiamo essere in grado di accontentarlo anche senza conoscere in partenza tali dati. La soluzione è simile a quella adottata in algebra, vale a dire la rappresentazione di un'incognita tramite una o più lettere.

Nell'esempio sopra citato, l'età dell'utente potrebbe essere rappresentata dalla lettera "a". Diremo che "a" è una "variabile", tenendo conto che qualsiasi variabile BASIC possiede due importanti caratteristiche:

- il tipo; possiamo avere sia variabili numeriche che rappresentano un numero, sia variabili stringa che rappresentano un insieme di caratteri;
- il nome; risulta soggetto a regole molto restrittive che stabiliscono se il nome di una variabile è accettabile o meno; inoltre vari tipi di nomi distinguono i diversi tipi di variabile.

Ogni volta che viene introdotta una variabile, questa viene collocata automaticamente, tramite l'interprete BASIC, in una locazione di memoria RAM dalla quale il computer la potrà richiamare, ogni qualvolta risulti necessario, per mezzo dell'indirizzo.

Variabili Numeriche

Le variabili numeriche sono le più semplici e si riferiscono a dei numeri. Il nome di queste variabili può essere formato da lettere, da numeri o anche da intere parole; l'importante è che il primo carattere sia una lettera, ad esempio:

```
A
A
A2
Record
```

e non

```
2
2A
```

Lo Spectrum, tra l'altro, non distingue tra maiuscole e minuscole nel nome di una variabile; inoltre ignora gli spazi; per cui scrivere FRANCO, FRanco, franco, fraNCO oppure FR anco, è per lui la stessa cosa. Per poter usare una variabile, è necessario darle un valore. Il modo più semplice è quello di usare la funzione LET, per es.:

```
Let A=100
```

che equivale a dire allo Spectrum:

- crea in RAM lo spazio destinato alla variabile A;
- memorizza in quello spazio il valore 180.

Con le due linee che seguono

```
10 LET A=100
20 LET A=200
```

alla variabile A verrà assegnato il valore 200, in quanto il secondo statement va a sovrapporsi al primo, cancellandolo.

Battete ora di seguito una terza linea

```
30 PRINT A
```

Osservate un attimo il comando sopra: in esso non si dice allo Spectrum di battere la lettera A; per fare ciò, essa dovrebbe essere posta fra virgolette nel comando:

```
30 PRINT "A"
```

mentre il comando precedente ordina allo Spectrum di scrivere il valore della variabile A.

Vediamo che cosa accade chiedendo allo Spectrum di scrivere il valore di una variabile non ancora usata, introducendo il seguente comando:

```
NEW
```

che cancella dalla memoria qualsiasi tipo di programma o variabile; inserendo ed eseguendo il programma:

```
10 PRINT B
```

la risposta dello Spectrum sarà

```
variable not found
```

Semplici somme

Lo Spectrum può addizionare due numeri senza alcuna difficoltà. Provate:

```
10 LET a=5
20 LET b=7
30 LET c=a+b+2
40 PRINT c
```

Il risultato sarà ovviamente 14 (5+7+2). Variate ora la linea 30 per ottenere una sottrazione:

```
30 LET c=b-a
```

oppure una moltiplicazione:

```
30 LET c=a*b
```

o, ancora, una divisione:

```
30 LET c=b/2
```

Chiunque abbia nozioni di algebra, potrebbe rimanere esterrefatto davanti ad uno statement BASIC del tipo

```
LET a=a+1
```

ma bisogna tener conto che il segno "=" non ha qui lo stesso significato di quello algebrico in quanto non stabilisce l'eguaglianza di due grandezze ma rappresenta una precisa istruzione per il computer:

— ricava il valore dell'espressione posta alla destra del segno =;

— e quindi attribuiscila alla variabile presente alla sua sinistra.

L'esempio di addizione riportato qui sopra aggiunge infatti il valore 1 alla variabile A.

Lo Spectrum possiede anche l'operatore aritmetico "↑" (sul tasto H) che sta a significare "elevamento a potenza di".

Ad esempio:

```
5↑2 e' l'equivalente di 5*5
5↑4 e' l'equivalente di 5*5*5*5
```

Alla sinistra del segno "↑" non vanno posti numeri negativi;

```
(-3)↑2
```

ha come risposta un messaggio di errore.

Espressioni più complicate

Non vi è alcun limite alla complicazione delle espressioni scritte alla destra del segno =. Ricordatevi, comunque, che le operazioni di elevazione a potenza (contrassegnate da ↑) vengono risolte per prime, se-

guite in ordine dalle moltiplicazioni e/o divisioni e dalle addizioni e/o sottrazioni. Per calcolare, ad esempio

```
LET c=a+b/7
```

lo Spectrum eseguirà prima la divisione b/7 e quindi addizionerà a al risultato. Similmente:

```
LET c=5+6*2-1
```

assegnerà a c il valore 16 come conseguenza delle operazioni:

```
  5
+ 6*2
- 1
```

È possibile ordinare al computer la priorità di particolari operazioni racchiudendo queste ultime tra parentesi come segue:

```
LET c=(5+6)*2-1
```

che avrà come risultato 21, e:

```
LET c=(5+6)*(2-1)
```

che invece darà 11.

Sebbene la tastiera disponga di diverse forme di parentesi, le uniche da usarsi nelle espressioni aritmetiche sono quelle tonde disegnate in rosso sui tasti 8 e 9. Può succedere anche di trovare parentesi racchiuse... tra parentesi:

```
(1+(2+3)*4)*5

(1+  5  *4)*5
=(1+  20  )*5
=    21    *5
=      105
```

In algebra capitano spesso espressioni del tipo

```
2(a+b)
```

le quali sottintendono implicitamente, tra la cifra "2" e la parentesi a si-

nistra, la presenza del segno di moltiplicazione; ciò in BASIC non è valido e la stessa espressione va scritta:

```
2*(a+b)
```

In generale

Il BASIC dello Spectrum è, in questo senso, estremamente agile: dove la sintassi dello statement richiede un numero, lo potete sostituire con un'espressione numerica, purché il risultato sia numerico. Poiché, come abbiamo visto nel capitolo precedente, la parola PRINT può essere seguita da un numero, se ne deduce che la stessa parola può essere fatta seguire da un'espressione numerica:

```
PRINT 22+33
```

Provate e vedrete che ciò è vero, come dimostrato dal seguente esempio:

```
10 LET min=4
20 LET max=10
30 PRINT "La media e' ";(min+max)/2
```

INPUT

Gli unici valori sui quali il programma ha lavorato finora sono stati quelli già compresi nel programma. Il comando INPUT permette di inserire nel programma, durante l'esecuzione, i valori dati dall'operatore. Nella sua forma più semplice, uno statement di INPUT consiste nell'espressione BASIC "INPUT" (tasto I) seguito dal nome di una variabile:

```
10 INPUT a
20 PRINT a
```

Facendo girare queste due righe, lo schermo si cancella presentando solamente, nell'angolo in basso a sinistra, il cursore L lampeggiante. Ciò è dovuto allo statement di INPUT, che pone lo Spectrum in attesa dell'immissione di un numero. Battetelo e date l'ENTER, lo vedrete sparire dalla parte bassa dello schermo per riapparire nell'angolo in alto a

sinistra. La riga 10, in effetti, non fa altro che attribuire ad "a" il valore che avete inserito. Provate ora ad immettere, anzichè un numero, un'espressione numerica, ad esempio:

```
22/2
```

vedrete che il programma accetta l'espressione visualizzando il risultato. Il solo problema con questo breve programma è che non è molto "amichevole" (in gergo tecnico si dice: "user friendly"). Sostanzialmente vi presenta uno schermo vuoto e aspetta che facciate la cosa giusta (cioè battere un numero). Questo metodo, adatto a programmi corti, come quello visto, non lo è per quelli più complessi, che richiedono più di un INPUT. Può essere necessario qualche PROMPT (richiesta) ogni qualvolta si debba inserire un valore. Naturalmente, potete usare uno statement di PRINT:

```
10 PRINT "Inserisci il numero"
20 INPUT n
30 PRINT "Hai inserito";n
```

Il comando INPUT può venire anche impiegato per visualizzare un messaggio; per far ciò bisogna considerare che:

- un INPUT può essere fatto seguire da qualsiasi numero di elementi separati fra di loro nello stesso modo visto per lo statement di PRINT;
- se un elemento inizia con una lettera, questa è considerata come il nome di una variabile e si mette in attesa che l'utente inserisca il relativo valore;
- qualsiasi altra cosa viene battuta come se facesse parte di un PRINT, ma in basso sullo schermo. Potete usare anche TAB e AT, ma il numero in AT sarà quello di una riga della parte bassa dello schermo;
- non appena eseguito lo statement di INPUT, il computer cancellerà tutto ciò che tale istruzione ha fatto visualizzare sullo schermo.

Ecco una dimostrazione:

```
10 INPUT "Inserisci il numero";a
20 PRINT "Hai inserito ";a
```

Il comando INPUT può attribuire il valore anche a più di una variabile.

```
10 INPUT "Inserisci a";a'"Inserisci b ";b'
   "Inserisci c ";c
20 PRINT a'b'c
```

Eseguendo questo programma, vedrete come lo Spectrum proceda passo passo nell'eseguire l'istruzione di INPUT.

Variabili Stringa

Una variabile può anche far riferimento ad una stringa la quale, come già visto, è un insieme di caratteri alfabetici e grafici o qualsiasi altro simbolo disponibile sulla tastiera. Per far capire allo Spectrum se la variabile fa riferimento a un numero o a una stringa, si è adottato, per queste ultime, il singolare accorgimento di far seguire la lettera interessata, dal segno $

```
A$
x$
```

Il computer non distingue tra lettere maiuscole e minuscole (A$, quindi, equivale ad a$), mentre riconosce una variabile di stringa (A$) da una variabile numerica (A). Per dare il valore ad una variabile stringa, si usa lo statement LET come avviene per le variabili numeriche;

```
10 LET a$="Spectrum"
20 LET b$=a$
30 PRINT b$
```

Se la variabile scritta alla sinistra dell'uguale è una variabile stringa, anche il risultato dell'espressione a destra dovrà essere una stringa. Questo fatto vale anche per le variabili numeriche. Non è pertanto possibile mescolare le due grandezze come segue:

```
LET a="Spectrum"
LET a$=12+34
```

È possibile inserire variabili stringa anche nello statement di INPUT:

```
10 INPUT a$
20 PRINT a$
```

All'esecuzione del programma, il display presenta il cursore L racchiuso tra virgolette nell'angolo inferiore sinistro. A questo punto, quanto inserito viene riportato tale e quale. E potrete notare che tutto ciò

che battete viene stampato esattamente come l'avete introdotto, senza che venga eseguito alcun calcolo. Battendo ad esempio:

```
1+2
```

viene scritto

```
1+2
```

e non

```
3
```

Talvolta l'uso delle virgolette si rivela una seccatura; qualora ve ne voleste sbarazzare, battete l'istruzione LINE (tasto 3) prima del nome della variabile stringa;

```
10 INPUT "Il tuo nome ?";LINE n$
20 INPUT "Ciao ";(n$)'"Quanti anni hai ? ";y
30 PRINT n$;" hai ";y;" anni "
```

Notate l'utilità della riga 20: non è possibile stampare il valore n$ (cioé il vostro nome) come parte del prompt di INPUT, in quanto l'INPUT crede che qualsiasi messaggio che inizi con una lettera sia una variabile da inserire; ma racchiudendo tra parentesi il nome della variabile, il vostro messaggio non inizierà più con una lettera! In questo modo potete battere n$ (il vostro nome).

Trattamento delle stringhe

Così come avviene per i numeri, se alla sinistra dell'uguale vi è il nome di una variabile stringa alla sua destra dovrà comparire una stringa. Vi sono due metodi per costruire un'espressione di stringa, il primo dei quali si avvale del segno "+" per unire due stringhe:

```
LET a$="ORA"+"RIO"
```

rende a$ uguale a

```
10 LET a$="TRE"
20 LET b$=a$+"NO"
30 PRINT b$
```

che dà come risultato "TRENO", ottenibile anche con

```
10 LET a$="TRE"
20 LET a$=a$+"NO" : PRINT a$
```

Il BASIC dello Spectrum può anche copiare solamente parte della stringa per mezzo dell'operatore TO (tasto F). In generale:

```
LET a$=b$(x TO y)
```

mette in a$ i caratteri dall'x-esimo all'y-esimo di b$. Ad esempio, con

```
LET m$="11-SET-82"(4 TO 6)
```

rende m$="SET".

Se il numero alla sinistra di TO è 1, ciò significa che vogliamo iniziare a copiare cominciando dal primo carattere, quindi possiamo ometterlo

```
"SINCLAIR"( TO 3)   e'  "SIN"
```

Analogamente, volendo copiare dei caratteri a partire da qualsiasi punto della stringa fino alla fine — e quindi il numero dopo il TO sarebbe uguale alla lunghezza della stringa — possiamo ometterlo anche in questo caso:

```
"SINCLAIR"(5 TO )   e'  "LAIR"
```

Dovete specificare un numero di caratteri pari a quelli realmente presenti; istruzioni come

```
"ABC"( TO 4)
```

portano ad errore (subscript wrong) poiché ABC sono in tutto tre soli caratteri. Volendo estrapolare un solo carattere, si può scrivere

```
"OXO"(2 TO 2)
```

ma ancora più semplicemente

```
"OXO"(2)
```

L'operatore TO si impiega anche per variare parte della stringa. Ad esempio

```
10 LET a$="123456"
20 LET a$(3 TO 4)="AB"
30 PRINT a$
```

darà come risultato

```
12AB56
```

in quanto la linea 20 chiede allo Spectrum di sostituire il terzo e il quarto carattere di a$ con quelli presenti alla destra del segno =. Una eventuale

```
20 LET a$(6 TO 7)="AB"
```

porterebbe ad un errore non esistendo in a$ un settimo carattere.

E se l'espressione alla destra dell'uguale dovesse risultare troppo lunga o troppo corta? In entrambi i casi provvede lo Spectrum. Se è troppo lunga, riporta solo quanto necessario:

```
20 LET a$(3 TO 4)="ABCD"
```

porta al risultato di

```
12AB56
```

Se è invece troppo corta, vengono aggiunti spazi:

```
20 LET a$(3 TO 4)="A"
```

dà come esito

```
12A 56
```

Dovendo sostituire un solo carattere, conviene usare la forma

```
LET a$(x)=
```

al posto di

```
LET a$(x TO x)=
```

Funzioni

Il BASIC comprende un numero di parole conosciute come "funzioni" le quali svolgono operazioni pre-programmate che ottengono risultati sia numerici che di stringa. Una di queste funzioni è SQR che calcola la radice quadrata:

```
PRINT SQR 4
```

risulta 2.

Il valore sul quale la funzione agisce prende il nome di argomento della funzione. Nell'esempio sopra riportato l'argomento della SQR è 4. Alcune funzioni operano su di un solo argomento, altre su due e pochissime non richiedono alcun argomento; anche una variabile può essere impiegata come argomento:

```
PRINT SQR A
```

oppure una espressione numerica

```
PRINT SQR (9+16)
```

In questo caso è necessario racchiudere l'espressione entro parentesi in quanto

```
PRINT SQR 9+16
```

darebbe un risultato differente.

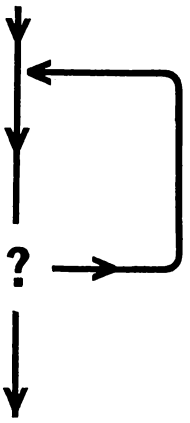
L'elenco delle funzioni a disposizione è presentato sul manuale. Andatelo a sfogliare e noterete che quelle relative a risultati composti da stringhe, terminano col segno $ (esempio CHR$). Altre (COS, EXP, etc.) sono matematiche e il loro impiego è ovvio nella stesura del programma. Ve ne sono anche alcune che sembrerebbero strane e che vedremo più avanti, ma delle quali potrete scoprire già ora la funzione, facendole precedere dallo statement di PRINT per vedere a che risultato portano.

CAPITOLO 4
LOOP E DECISIONI

Un programma scritto in BASIC si presenta come una lista di linee numerate in ordine crescente. Quando si dà il RUN, il computer prende in considerazione la linea di programma con il numero inferiore, la esegue, passa alla successiva e così via fino a giungere alla fine del listato dove si ferma.

Ma un computer può fare qualcosa di più complesso che compiere un tragitto del genere: è infatti possibile fargli ripetere una riga o un gruppo di righe di programma quante volte vogliamo.

È possibile anche che, dietro particolari ordini, il computer salti a numeri di linea più alti oppure più bassi (ignorando quanto sta in mezzo). Sono appunto queste possibilità che conferiscono al sistema la flessibilità necessaria ad eseguire programmi di una certa complessità.



GO TO

L'istruzione GO TO (tasto G) si impiega per variare l'ordine in cui lo Spectrum esegue gli statement facendolo saltare a linee diverse da quella successiva. Nella sua forma più elementare, lo statement GO TO è seguito semplicemente da un numero di riga:

```
10 PRINT "Spectrum.";
20 GO TO 10
```

Provate: è una routine elementare molto efficace! Quando il computer esegue la riga 20, compie un balzo all'indietro tornando alla 10 e così via fino a riempire l'intero schermo e a far apparire nell'angolo in basso a sinistra la domanda:

```
scroll ?
```

Se si risponde premendo un tasto qualsiasi all'infuori di N o di BREAK, il programma continuerà e lo schermo si riempirà un'altra volta.

Provate ora ad aggiungere una terza riga:

```
15 POKE 23692,0
```

e vedrete che la scritta continuerà a scorrere verso l'alto senza più chiedere il vostro permesso. Se non ne comprendete il significato, non preoccupatevi: consideratela per ora una parola magica. Questa è la tipica struttura di un "loop senza fine", per interrompere il quale è necessario dare il BREAK (CAPS SHIFT e BREAK assieme oppure togliere l'alimentazione all'apparecchio).

Potete far eseguire al computer anche salti in avanti, come dimostra il programma seguente, che non esegue mai la linea 30:

```
10 PRINT "LO SPECTRUM E' IL "
20 GO TO 40
30 PRINT "MIGLIORE "
40 PRINT "COMPUTER "
```

I giorni della settimana

Il numero che segue GO TO può essere seguito da un'espressione numerica, per introdurre nel salto un elemento di scelta:

```
  10 INPUT "Inserisci i giorni d
ella settimana con i numeri""1-
7 ";d
  20 CLS   PRINT "Il giorno ";d;
" e'";
  30 GO TO 100+d
 100 GO TO 10
 101 PRINT "Lunedi'": GO TO 10
 102 PRINT "Martedi'": GO TO 10
 103 PRINT "Mercoledi'": GO TO 1
0
 104 PRINT "Giovedi'": GO TO 10
 105 PRINT "Venerdi'": GO TO 10
 106 PRINT "Sabato": GO TO 10
 107 PRINT "Domenica": GO TO 10
```

In questo programma la riga 30 provoca un salto alla 101 se avete inserito il numero "1", alla riga 102 se avete invece inserito il "2" e così via. Il problema principale, in queste forme di GO TO "calcolato" si presenta quando viene inserito un valore al di fuori di quelli stabiliti. Se, ad esempio, introducete il numero "0", la riga 30 farà compiere un salto alla 100 che è stata appunto inclusa per ovviare a tale errore. Se invece la cifra introdotta è maggiore di 7, il salto avviene su righe inesistenti, poste fuori dal limite massimo del programma.

Fortunatamente il BASIC dello Spectrum è molto elastico in questo senso, così, se cercate di saltare ad una linea inesistente, questo vi porterà alla riga di programma più alta dopo quella in cui vi trovavate e, nel caso non ne esistessero, il programma si fermerà.

IF... THEN

La combinazione IF-THEN mette il programma in condizione di effettuare una scelta per operare in una certa direzione anziché in un'altra.

Le due parole sono sempre abbinate nello statement che deve avere la forma

```
IF espressione logica THEN statement
```

Un'espressione logica è una particolare espressione che può dare un risultato "vero" o "falso". Esempio:

```
A=B
Nero=Bianco
2<=5
a$<>"SI"
```

Lo statement o gli statement che seguono THEN possono essere delle qualsiasi istruzioni BASIC, devono appartenere alla stessa riga di programma di IF-THEN e vengono eseguite soltanto se la "espressione logica" dà come risultato "vero". Se invece, il risultato è "falso" lo Spectrum salta alla linea di programma successiva.

Se THEN è seguito da più di uno statement, ognuno di questi va separato dai due punti come avviene in qualsiasi altra linea di programma. Facciamo un esempio pratico:

```
IF anno=2084 THEN PRINT "Centenario"
```

Il vocabolo "Centenario" viene scritto solo quando la variabile "anno" ha il valore "2084".

```
IF punti>record THEN PRINT "BRAV
O": LET record=punti
```

è una routine assai usata da mettere alla fine di qualsiasi programma di giochi. IF-THEN possono essere seguiti anche da un secondo IF-THEN come mostra l'esempio che segue:

```
IF mese=4 THEN IF giorno=23 THEN
 PRINT "Buon compleanno"
```

Nella maggior parte dei casi IF-THEN è però seguita da GO TO come nella routine che segue, la quale scrive i numeri dall'1 al 10.

```
10 LET A=1
20 PRINT A
30 LET A=A+1
40 IF A<=10 THEN GO TO 20
```

Per quanto riguarda i valori "veri" o "falsi", ricordatevi che quando lo Spectrum svolge espressioni logiche dà sempre un risultato numerico che è "1" se l'espressione è "vera" oppure "0" se l'espressione è "falsa". Dimostriamolo battendo:

```
PRINT 2=2,2=3
```

che dà come risultato:

```
1    0
```

Similmente la linea

```
LET a=espressione logica
```

attribuisce ad "a" il valore "1" se l'espressione logica è vera e "0" se è falsa.

In pratica, lo Spectrum considera "vero" qualsiasi valore diverso da "0" per cui la linea

```
IF a THEN PRINT "VERO"
```

presenta "VERO" in ogni caso meno che per a=0. Ricordate che i simboli < > (diverso da), <= (minore o uguale a) e >= (maggiore o uguale a) vanno inseriti tramite istruzioni singole e non come combinazioni di <,> con =.

NOT

È una funzione che trasforma "vero" in "falso" e viceversa.

```
IF NOT (2=3) THEN PRINT "2 non uguale a 3)
```

AND

L'istruzione AND è un operatore logico per cui può venire inserita nel contesto di espressioni logiche. Se scrivete

```
p AND q
```

dove "p" e "q" sono espressioni logiche, il risultato sarà vero solo quando sia "p" che "q" sono veri. Esempio:

```
IF (mese=12) AND (giorno=9) THEN PRINT "Buon
   Compleanno"
```

può essere un'alternativa alla routine vista in precedenza in cui si illustrava la funzione di due IF nella stessa riga. AND può essere usata anche con espressioni numeriche,

```
A AND B
```

dà come risultato: A se B <> 0 e 0 se B = 0

B può anche essere una espressione logica per cui possiamo scrivere lo statement che segue:

```
LET altezza=altezza - (10 AND altezza >=10)
```

che sottrae 10 alla variabile "alto" solo se "alto" è >= 10. AND può essere impiegata anche con espressioni stringa.

```
A$ AND B
```

porta come risultato A$ per B<>0 o a una stringa nulla (" ") per B=0.

Ciò può essere utile anche in statement PRINT come

```
PRINT ("BUONO" AND X)+("CATTIVO AND NOT X)
```

che scrive "BUONO" oppure "CATTIVO" a seconda del valore di x. Affinando ulteriormente le nostre capacità, possiamo scrivere espressioni numerico-logiche dopo GO TO:

```
GO TO (100 AND x<0)+(200 AND x=0)+(300 AND x>0)
```

Questo esempio ordina al computer di saltare alla linea 100, 200 o 300 a seconda del valore attribuito a x.

OR

Anche la parola OR (sul tasto U), è un operatore logico.

```
p OR q
```

Il risultato è vero se solo "p" oppure solo "q" oppure entrambi sono veri, mentre è falso solo nel caso in cui tutte e due le espressioni logiche siano false. Un esempio di impiego:

```
IF (potenza=0) OR (aria=0) THEN PRINT "SEI MORTO !"
```

Come AND, anche OR può essere impiegata in espressioni numeriche

```
A OR B
```

dà come risultato 1 per B<>0 (vero)
A per B=0 (falso)

Per cui:

```
LET A=A OR (A<1)
```

assicura che il valore della variabile A non scenda al disotto di 1. OR non può essere usata con espressioni stringa.

FOR-TO-STEP... NEXT

Poco sopra abbiamo presentato un semplice programma per scrivere i numeri dall'1 al 10, per mezzo dello statement IF-THEN-GO TO che chiudeva il loop non appena raggiunta la decima cifra. Ecco un secondo metodo per ottenere gli stessi risultati: il loop FOR - NEXT. È una struttura del BASIC molto utile che permette la ripetizione di tutto o di parte del programma, per un determinato numero di volte, prima di pro-

seguire o di fermarsi. Tale forma necessita di due statement ben distinti:

```
FOR a=x TO y STEP z

NEXT a
```

dove FOR, TO, STEP e NEXT sono parole rintracciabili sui tasti F,F,D e N. La parte del programma soggetta a ripetizione è quella posta tra il FOR e il NEXT. "a" è detta "variabile di controllo" del loop; in realtà è una variabile numerica come tutte le altre, eccettuato il fatto che il suo nome deve essere costituito da una sola lettera, mentre "x", "y" e "z" sono espressioni numeriche. Quando lo Spectrum incontra lo statement FOR, calcola innanzitutto le espressioni "x", "y" e "z" e ne memorizza i risultati; quindi pone la variabile di controllo "a" uguale al valore di partenza "x". Esegue poi gli statement successivi al FOR fino a trovare il NEXT che possiede la stessa variabile di controllo del FOR. A questo punto, lo Spectrum aggiunge alla variabile di controllo "a" il valore di STEP "z" (STEP significa appunto "passo") dopodiché, se il valore di "a" così incrementato supera quello del valore limite "y" (o è al disotto se il valore di STEP "z" è negativo), salta allo statement che segue il NEXT, proseguendo nello svolgimento del programma. Viceversa se il valore di "a" rimane entro il limite imposto da "y" il programma esegue un loop a ritroso, tornando al FOR e ripetendo il ciclo. Eccezioni a queste regole si verificano quando:

- il valore della variabile di controllo è maggiore del valore limite "y" e lo STEP "z"è positivo;
- il valore della variabile di controllo è minore del valore limite "y" e lo STEP è negativo.

In questi casi il computer salta direttamente allo statement successivo al NEXT senza eseguire gli statement posti tra il FOR e il NEXT.

Se la parola STEP e il suo valore (z) vengono omessi, il valore del passo è inteso come 1. Molto spesso gli esempi valgono più delle paro-

le per cui ecco qualche semplice esempio:

```
10 FOR a=1 TO 10
20 PRINT a
30 NEXT a
```

scrive i numeri dall'1 al 10, mentre

```
10 FOR a=10 TO 1 STEP -1
20 PRINT a
30 NEXT a
```

li scrive in ordine inverso.

Aggiungete a questi due esempi la riga

```
40 PRINT a
```

e vi verrà mostrato il valore della variabile di controllo alla fine del loop. Variando la linea 10 in:

```
10 FOR a=1 TO 0
```

oppure in:

```
10 FOR a=0 TO 1 STEP -1
```

si verificano le condizioni per le quali gli statement nel loop non vengono eseguiti affatto come sopra accennato. Volendo, è possibile cambiare il valore della variabile di controllo come qualsiasi altra variabile:

```
10 FOR f=1 TO 5
20 IF f=13 THEN LET f=14
30 PRINT f
40 NEXT f
```

otterrete il numero delle stanze di un hotel americano.

Tabelline

Un'applicazione pratica del loop FOR-NEXT è il programma che segue, relativo alla tabellina di moltiplicazione:

```
10 INPUT "Tavola per numero?";t
20 FOR a=1 TO 12
30 PRINT a;" moltiplicato ";t;" = ";a*t
40 NEXT a
```

Sinusoide

Un'ulteriore esemplificazione di come viene utilizzato un loop FOR NEXT è il seguente programma che disegna, sommariamente, una sinusoide; se non sapete cosa essa sia, limitatevi a guardarla:

```
10 FOR a=0 TO 31
20 PRINT AT (11+10*SIN (a*PI/16)),a;"■"
30 NEXT a
```

Il quadrato nero posto tra virgolette si ottiene azionando contemporaneamente INV VIDEO e CAPS SHIFT. Se preferite una curva più fitta, variate la riga 10 inserendo un passo più piccolo:

```
10 FOR a=0 TO 31 STEP .5
```

Dando il valore 0 alle dimensioni del passo, otterrete un loop senza fine, ma, con questo programma, non vi porterà molto lontano!

Numeri primi

Ecco un esempio di loop FOR-NEXT usato più di una volta in un programma per il calcolo dei numeri primi fino ad 80, dopo il 2. La struttura è assai complessa, ma le frecce indicano chiaramente il cammino che il programma effettua prima di raggiungere la riga col numero via via sempre più alto. Con un esempio di questo tipo potete avere un'idea di come lavora un programma, e questo potrà tornarvi utile in futuro. Il loop principale del programma scorre tra le righe 20 e 80, selezionando

man mano tutti i numeri primi; la variabile di controllo "A" viene usata anche dalla riga 70 per presentare i numeri ben incolonnati:

```
10 LET t=1
20 FOR a=0 TO 79
30 LET t=t+2
40 FOR f=3 TO SQR t STEP 2
50 IF t=f*INT (t/f) THEN GO TO
30
60 NEXT f
70 PRINT TAB 8*a;t;
80 NEXT a
```

Nel corpo di questo loop esterno si valuta se la variabile "t" è un numero primo. Partendo da un numero dispari e aggiungendo 2 ogni volta (riga 30) il programma evita di provare numeri pari che non possono essere primi. Il test per vedere se "t" è primo, avviene tra le righe 40 e 60 e rivela se gli interi dispari compresi fra 3 e la radice quadrata di "t" sono un fattore di "t" stesso. Se sì, "t" non è un numero primo e il programma torna alla linea 30 per definire il seguente valore di "t" da esaminare. Se invece non risultano fattori, ciò significa che "t" è un numero primo e viene stampato dalla riga 70. È degno di nota il fatto che, diversamente dal BASIC dello Spectrum, non tutti i dialetti BASIC concepiscono l'uscita dal loop FOR-NEXT in un programma.

NESTING

L'inserimento di un loop FOR-NEXT all'interno di un altro

```
FOR A=B TO C
FOR X=Y TO Z
- - - -
NEXT X
NEXT A
```

è noto come "nesting" (annidamento) ed è conforme al linguaggio BASIC, mentre non lo è, almeno per lo Spectrum, una struttura di FOR

NEXT incrociata come quella che segue:

```
┌FOR A=B TO C
┌FOR X=Y TO Z
│- - - -
└NEXT A
└NEXT X
```

Numeri binari

Questo programma è stato sviluppato per dimostrare un "nesting" che usa otto loop per formare tutti i 256 numeri binari ad otto bit scrivendoli assieme agli equivalenti decimali. Se siete in possesso di una stampante per lo ZX, potete stamparvi la tabella di conversione Binario-Decimale semplicemente inserendo alla linea 110 LPRINT al posto di PRINT.

```
100 FOR a=0 TO 1
101 FOR b=0 TO 1
102 FOR c=0 TO 1
103 FOR d=0 TO 1
104 FOR e=0 TO 1
105 FOR f=0 TO 1
106 FOR g=0 TO 1
107 FOR h=0 TO 1
110 PRINT a*128+b*64+c*32+d*16+
e*8+f*4+g*2+h;TAB 4;a;b;c;d;e;f;
g;h
120 NEXT h
121 NEXT g
122 NEXT f
123 NEXT e
124 NEXT d
125 NEXT c
126 NEXT b
127 NEXT a
```

```
0     00000000
1     00000001
2     00000010
3     00000011
4     00000100
5     00000101
6     00000110
7     00000111
8     00001000
9     00001001
10    00001010
11    00001011
12    00001100
13    00001101
14    00001110
15    00001111
248   11111000
249   11111001
250   11111010
251   11111011
252   11111100
253   11111101
254   11111110
255   11111111
```

CAPITOLO 5
ARRAY E DATI

Come abbiamo visto, è molto facile programmare lo Spectrum per fargli svolgere più di una volta la stessa routine, usando un loop FOR NEXT o uno statement IF THEN GOTO.

Quando il programma deve manipolare una quantità ingente di dati, è necessario organizzarli in una sezione destinata ad essere richiamata ogni volta che se ne presenti la necessità. Con le semplici variabili numeriche di stringa viste finora, è possibile rappresentare solamente una singola voce mentre per la manipolazione di set di dati, il BASIC prevede l'uso di variabili "array" (a schiera).

Array numerici

Supponete di essere degli insegnanti e di voler memorizzare nel computer i voti di fine trimestre dei vostri allievi. Per semplificare le cose momentaneamente, supponete di dover memorizzare un solo voto per ogni alunno. Userete una variabile diversa per ognuno e, poiché lo Spectrum, per quanto riguarda il nome attribuito alle variabili, non pone restrizioni di spazio, potete inserire i nomi al completo:

```
LET DINO ROSSI = 20
LET PIERA VERDI = 73
LET SANDRO CONTI = 100
```

in questo modo i dati sono stati memorizzati, inoltre potrete richiamare il voto di ogni alunno semplicemente con lo statement

```
PRINT Angelo Bianchi
```

In questi termini il programma non è però in grado di manipolare i

dati inseriti. Infatti per compiere un'operazione semplice, come il calcolo del voto medio della classe, il programma dovrebbe conoscere in anticipo tutti i nomi degli alunni. Se non vi rendete conto dell'entità del problema, provate ad inserire un programma per la somma dei voti di trenta persone con i voti riferiti a variabili individuali (come mostrato nell'esempio precedente) senza conoscere il nome degli alunni al momento della stesura del programma! L'ostacolo viene aggirato riferendo direttamente i nomi ad altrettanti numeri, dimenticando momentaneamente i nomi.

Nel caso dei 30 allievi, userete i numeri dall'1 al 30, varando il nuovo comando DIM (sul tasto G) in uno statement del genere:

```
DIM m(30)
```

che dirà allo Spectrum di riservare uno spazio in memoria per inserirci le 30 variabili m (1)—m (30). Tali variabili vengono usate alla stregua delle altre variabili numeriche, ma non possono entrare a far parte di loop FOR NEXT come variabili di controllo. Fra parentesi possono trovar posto anche espressioni numeriche o variabili.

```
m(7)
m(8-1)
m((a-1)/2)     dove 'a' vale 15
```

sono tutte forme legali che si riferiscono alla stessa variabile. Questo sistema permette al programma di manipolare a turno tutte e trenta le variabili per mezzo di un semplice loop. Ecco l'esempio di una piccola routine che vi permetterà di inserire i voti:

```
100 FOR p=1 TO 30
110 INPUT "Voto per allievo ";(p);m(p)
120 NEXT p
```

Il set delle trenta variabili da m(1) a m(30) è conosciuto come "array". Vi è qualche nota sul suo uso:

— lo spazio in memoria destinato all'array, deve essere prenotato dallo statement DIM prima ancora di usare qualsiasi elemento dell'array stesso;

- il nome dell'array (m nell'esempio precedente) deve essere formato da una singola lettera. Anche qui lo Spectrum non fa distinzione tra maiuscole e minuscole;
- oltre a prenotare spazio in memoria, DIM azzera il valore di tutti gli elementi;
- l'elemento "m(n)" dell'array è una variabile diversa dalla variabile numerica "m";
- in presenza di due statement DIM uguali situati nello stesso programma, il primo viene cancellato dal secondo quando questo è eseguito dallo Spectrum;
- l'elenco inizia sempre dal numero 1 ed il limite superiore è quello della capacità di memoria. Se ogni elemento occupa ad esempio 5 byte, avrete a disposizione 1600 elementi di array con lo Spectrum da 16 K e ben 8200 con quello da 48 K.

Array numerici multidimensionali

Tornando al vostro ruolo di insegnanti, avete visto come attribuire un voto ad ogni allievo, o, il che è lo stesso, come trattare una colonna di numeri. Qualora voleste espandere il programma aggiungendo altre voci, ad esempio altre materie, è possibile usare un array diverso per ognuna:

array f(30) si riferisce ai voti di Francese
array m(30) si riferisce ai voti di Matematica
array c(30) si riferisce ai voti di Computisteria

È possibile semplificare il tutto usando un unico array con due indici. Considerando, ad esempio, cinque materie, dovremmo formare un array con questo statement:

```
DIM m(5,30)
```

si può immaginare come un foglio di carta suddiviso in 5 colonne di 30

righe ciascuna per cui l'elemento singolo precedente

```
m(s,p)
```

corrisponde sul foglio al numero che risiede nella colonna "s" alla riga "p" e cioè nell'esempio precedente la quotazione dell'allievo 30 nel campo 5. Avendo due indici... dovremo inserire in programma due loops FOR NEXT annidati:

```
100 FOR s=1 TO 5
110 CLS : PRINT "SOGGETTO ";s
120 FOR p=1 TO 30
130 INPUT "Voto per scolaro ";(p);m(s,p)
140 NEXT p
150 NEXT s
```

si sarebbe potuto usare anche un array del tipo:

```
DIM m(30,5)
```

il quale attribuisce il numero dell'allievo a "p" (primo indice) e quello del soggetto a "s" (secondo indice). Può anche essere aggiunto un terzo indice, per esempio quello relativo ai tre trimestri componenti l'anno scolastico:

```
DIM (3,5,30)
```

pensando il tutto come tre fogli di carta, uno per ogni trimestre.

Se sarete riusciti ad ottenere la promozione ad insegnante di ruolo, potrete inserire anche una quarta voce relativa al numero (supponiamo 6) delle classi affidatevi:

```
DIM (6,3,5,30)
```

È questo un array quadridimensionale; ma in realtà potete impiegare il numero di dimensioni che volete, entro i limiti della memoria dello Spectrum.

Array di stringhe/caratteri

Finora abbiamo riferito il nome degli allievi ad altrettanti numeri. Ora vediamo di costruire un array contenente i nomi veri ricorrendo al concetto di stringa. Sorge subito un interrogativo: quanto è lunga una stringa? Lo statement DIM non ha problemi nel riservare spazio ad un array numerico dato che ogni suo elemento è lungo 5 byte; non è però così per la stringa, che possiede lunghezze variabili. Lo Spectrum aggira l'ostacolo permettendo al DIM di prendere in considerazione un array di caratteri a patto che il nome della variabile sia seguito dal segno $. Ogni carattere occupa un byte di memoria per cui

```
DIM a$(10)
```

forma un array singolo di dieci caratteri e

```
DIM a$(5,10)
```

ne forma uno da 5 x 10. Anche in questo caso potete usare quante dimensioni volete, col solito limite imposto dalla capacità di memoria. Nel momento in cui viene eseguito uno statement DIM di caratteri, si cancellano sia gli array sia le variabili stringa che portano lo stesso nome. Non è possibile avere allo stesso tempo una a$ e una a$(3). Notate che alla stregua degli array numerici, anche gli array di caratteri hanno come nome una singola lettera (ma seguita da $).

Inoltre l'istruzione DIM riempie con degli spazi gli array di caratteri. In generale è preferibile avere a che fare con stringhe piuttosto che con singoli caratteri, così lo Spectrum consente di trattare parte di array di caratteri come se fossero stringhe.

Ecco alcune regole:

- se, riferendovi ad un elemento di un array di caratteri, omettete l'ultimo indice, per lo Spectrum vi state riferendo ad una stringa;
- tale stringa possiede il numero di caratteri dato dall'ultimo indice dello statement DIM. Ad esempio:

```
DIM a$(3,5)
```

consente di fare riferimento alle tre stringhe a$(1), a$(2), a$(3) ognuna delle quali ha una lunghezza di cinque caratteri. Un caso particolare, e molto utile, è quello in cui l'istruzione DIM ha un solo argomento

```
DIM a$(32)
```

in questo caso si ha a disposizione un array di 32 spazi utilizzabile come stringa facendo riferimento ad a$.

Vi tornerà più facile capire il concetto di array di carattere e di stringa, battendo la routine che segue:

```
10 DIM n$(2,3)
20 LET n$(1)="IO"
30 LET n$(2)="NOI"
```

la quale porta ad un array di caratteri del genere:

I	O	
N	O	I

La lunghezza di ogni stringa è stabilita dallo statement DIM. Inserendo una stringa troppo corta, lo Spectrum la completerà con degli spazi; inserendone una troppo lunga, prenderà in considerazione solo il numero di caratteri precisato dal DIM. Per esempio, variando la linea 20 come segue

```
20 LET n$=(1)="LORO"
```

si ottiene l'array di caratteri:

L	O	R
N	O	I

Tornando alla registrazione degli allievi, sarà necessario stabilire il numero dei caratteri da prevedere per ogni nome, diciamo 20, e poi formare un array di stringa di 20 caratteri.

```
DIM n$(30,20)
```

mette a disposizione 30 stringhe da n$ (1) a n$(30) di 20 caratteri ognuna, necessarie per una classe. Invece

```
DIM n$(6,30,20)
```

memorizza i nomi di 6 classi.

Comparazione di stringhe

Il fatto che le stringhe site nell'array siano tutte della stessa lunghezza fa sorgere un problema quando si deve procedere ad una comparazione tra due stringhe di cui solo una fa parte di un array. Questo accade perché lo Spectrum non riconoscerà mai come uguali le due stringhe a meno che non abbiano la stessa lunghezza. Ad esempio la routine.

```
10 DIM n$(3,5)
20 LET x$="TU"
30 LET n$(1)=x$
40 IF n$(1)=x$ THEN PRINT "SEI TU"
```

non presenterà mai la scritta "SEI TU" perché le stringhe confrontate nella linea 40 hanno lunghezze diverse. Ne avrete la dimostrazione aggiungendo

```
35 PRINT LEN n$(1), LEN x$
```

la quale mosterà per mezzo di LEN (tasto K) il numero dei caratteri di una stringa. Per risolvere l'enigma è necessario dimensionare x$ come un array di 5 caratteri inserendo la linea.

```
15 DIM x$(5)
```

ORDINAMENTO

Una volta inserite le informazioni in un array, vi chiederete come sia possibile il loro ordinamento. Questo breve programma, che fa uso di un array di 10 stringhe di 12 caratteri, vi chiede 10 parole da inserire e quindi le presenta in ordine alfabetico.

Il contenuto dell'array viene stampato ad ogni tappa del processo di riordino: è molto interessante seguirlo.

```
  10 DIM n$(10,12)
 100 FOR a=1 TO 10
 110 INPUT "Parola";(a); LINE n$
(a)
 120 PRINT AT a,0;n$(a)
 130 NEXT a
 200 LET c=0
 210 FOR a=1 TO 9
 220 IF n$(a)<=n$(a+1) THEN GO T
O 260
 230 LET t$=n$(a): LET n$(a)=n$(
a+1): LET n$(a+1)=t$
 240 PRINT AT a,0;n$(a);n$(a+1)
 250 LET c=1
 260 NEXT a
 270 IF c=1 THEN GO TO 200
```

L'algoritmo di ordinamento è semplice, infatti prende in considerazione due parole alla volta tra quelle presenti nella lista, invertendole di posizione se non risultano in ordine alfabetico (linea 230). Giunto alla fine dell'elenco, il programma controlla il flag "c" per vedere se è stato cambiato qualcosa. In caso affermativo, riesegue l'algoritmo.

Potete rallentare il processo fino a farlo diventare visibile aggiungendo:

```
255 PAUSE 20
```

PAUSE

La parola Pause che avete appena usato, spiega da sola il significato dell'istruzione. Lo statement

```
PAUSE n
```

ordina al computer di aspettare n/50 secondi (n/60 in U.S.A.) prima di passare all'esame dello statement successivo, quando il coefficiente "n" ha un valore compreso tra 1 e 65535. Inserendo un valore non intero, lo Spectrum lo trasformerà nell'intero ad esso più vicino come accade con altri comandi tipo TAB, AT e PLOT i quali lavorano appunto esclusivamente su numeri interi. Il ritardo può essere interrotto a piacere premendo qualsiasi tasto ad eccezione di CAPS SHIFT e SYMBOL SHIFT.

PAUSE 0 fa aspettare per un tempo indefinito o almeno fino a che non venga premuto un tasto; è preferibile durante le operazioni usare un loop FOR NEXT vuoto del tipo

```
FOR a=1 TO 100   NEXT a
```

particolarmente adatto quando è necessario ottenere un ritardo prestabilito senza correre il rischio di accelerare l'esecuzione del programma tenendo involontariamente premuto un tasto. Il loop si ripete all'incirca 200 volte al secondo o anche più lentamente se non si trova all'inizio del programma. Curiosamente, si può porre termine ad un PAUSE prima ancora che esso venga eseguito

```
10 PRINT 1
20 FOR a=1 TO 500: NEXT a
30 PRINT 2
40 PAUSE 100
50 PRINT 3
```

che dovrebbe visualizzare "1", aspettare un paio di secondi (riga 20) prima di visualizzare "2", aspettare altri due secondi circa (riga 40) prima di visualizzare "3"; finchè non toccherete la tastiera, il programma farà esattamente come vi abbiamo detto. Se però azionerete un tasto appena apparso l'"1" (mentre il computer sta eseguendo il loop di linea

20) verrà annullato anche il ritardo della seguente linea 40 e il "3" verrà presentato immediatamente dopo il "2". Per ovviare a tale inconveniente si ricorre a un doppio PAUSE, il primo dei quali, con un ritardo insignificante, ha lo scopo di "consumare" la pressione di un qualsiasi tasto battuto in precedenza. Nel programma precedente, sostituite la riga 40 con

```
40 PAUSE 1    PAUSE 100
```

DATA, READ E RESTORE

Volendo riempire un array con dati costanti, come per esempio i colori dell'iride, potete usare una serie di statement di LET

```
10 DIM c$(7,8)
20 LET c$(1)="viola"
30 LET c$(2)="indaco"
       ecc.
```

Tale sistema si rivela però noioso per cui il BASIC dello Spectrum propone un'altra soluzione: includere le voci in statement DATA per poi rileggerle all'interno dell'array. Uno statement DATA è formato dall'istruzione DATA (sul tasto D) seguita da una o più voci separate da virgole; esempio:

```
DATA 1,2," BIANCO      "
```

Questa funzione può venir inserita in qualsiasi punto del programma e, quando questo viene avviato, il "data pointer" (puntatore dei DATA) si posiziona sul primo statement di DATA del listing.

Lo statement READ è formato dalla istruzione READ (sul tasto A) seguita da una o più variabili separate da virgole:

```
READ a,b,c$
```

Quando il programma esegue lo statement READ attribuisce ad ogni

variabile il valore seguente situato nella lista dei dati e sposta anche il "data pointer". Quest'ultimo esempio porta al seguente risultato:

```
Alla variabile 'a' viene attribuito il valore '1'
Alla variabile 'b' viene attribuito il valore '2'
Alla variabile 'c$' viene attribuito il valore
'BIANCO'
```

I dati possono essere introdotti con un numero qualsiasi di DATA, perché READ prende ciascuna voce via via che appare nel listato del programma.

Volendo mettere in un array stringa i colori dell'arcobaleno, possiamo usare:

```
  10 DIM c$(7,6)
  20 DATA "violetto","indaco","b
lu","verde"
  30 DATA "giallo","arancio","ro
sso"
  40 FOR a=1 TO 7: READ c$(a): N
EXT a
```

Le voci collocate negli statement DATA non devono necessariamente essere delle costanti, ma possono essere anche espressioni numeriche o stringhe:

```
DATA a/a,SQR 4,CHR$ 66 + " BIANCO     "
```

L'unico accorgimento da tener presente è che le espressioni poste nello statement DATA devono avere la stessa natura delle variabili presenti nel corrispondente statement READ. Ad esempio, una variabile numerica nel READ va accoppiata con una espressione numerica nel DATA, e una variabile stringa con una espressione stringa.

Quando il "data pointer" ha terminato di leggere l'ultimo dato, ulteriori tentativi di lettura portano ad un messaggio di errore:

```
Out of data
```

In questo caso si impone l'uso dell'istruzione RESTORE:

```
RESTORE
```

il quale sposta il "data pointer" sul primo dato inserito nel primo DATA del programma.

```
RESTORE n
```

sposta il "data pointer" alla prima voce inserita alla riga "n" oppure alla prima voce inserita nel primo statement DATA che si trova dopo la riga n-esima, se questa non contiene DATA. A differenza di quanto afferma il manuale dello Spectrum, CLEAR non provoca alcun RESTORE automatico.

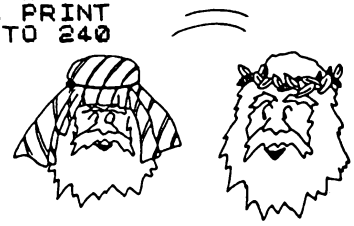
ARABI-ROMANI

L'esempio che segue vi mostra un array abbinato ad un DATA e converte i normali numeri "arabi" negli equivalenti "romani".

```
da 1984 a MCMLXXXIV
```

Le righe 100 e 130 dispongono i dati costanti in modo da essere usati nel programma (r$ è impiegato come array di 7 caratteri). Il loop dalla linea 230 alla 270 controlla quale dei sette numeri romani deve essere visualizzato.

```
 100 DIM r(9)
 110 DATA 1000,500,100,50,10,5,1
,0,0
 120 FOR x=1 TO 9: READ r(x): NE
XT x
 130 LET r$="MDCLXVI"
 200 INPUT "Numero (0 a stop) ?
";a
 210 IF a=0 THEN STOP
 220 PRINT 'a;" = ";
 230 FOR x=1 TO 7
 240 IF a>=r(x) THEN PRINT r$(x)
;: LET a=a-r(x): GO TO 240
 250 LET y=1+2*INT ((x+1)/2)
 260 IF a>=r(x)-r(y) THEN PRINT
r$(y);: LET a=a+r(y): GO TO 240
 270 NEXT x
 300 GO TO 200
```



CAPITOLO 6
STESURA DI UN PROGRAMMA

La canzone di trionfo del programmatore

Il primo che ho scritto è quel che ho venduto
secondo era quello che funzionava
ma io — coraggioso — l'ho scritto di nuovo
e ancora di nuovo per farlo migliore.
Adesso ce l'ho, è bello, è conciso:
amico, per dollari non te lo vendo
se vuoi comperarlo, acquistalo in yen!

Vi sarete accorti, a questo punto, che far girare programmi composti da altri è già motivo di soddisfazione, specialmente per quelli di voi che sono entrati a contatto per la prima volta con il mondo dei computer. Vi assicuro però che riuscire a stendere un programma da soli procura una soddisfazione molto maggiore. Questo capitolo prende in esame gli aspetti essenziali della creazione di programmi validi.

Struttura

Il primo e più importante punto per la stesura di un programma di successo è che la successione delle righe non va scritta come se si stesse scrivendo una lettera ad un amico, cioè iniziando con "caro Sandro..." e scrivendo tutto quello che ci viene in mente, magari chiudendo con un P.S. perché ci era sfuggito qualcosa. I programmi vanno invece creati come si crea un'opera d'arte o come si costruisce un edi-

ficio, partendo dalla base, rifinendolo poco per volta e accertandosi via via che la struttura si regga in piedi prima di passare ai dettagli successivi. Dovrete pertanto resistere alla tentazione di inserire subito le prime righe nel computer e aspettare di essere in possesso dell'intera struttura del sistema. Ogni programma di una certa consistenza comprende parecchi dettagli che vanno affrontati uno alla volta e che è impossibile tenere tutti a mente. È impossibile infatti poter vedere l'intero bosco a causa degli alberi stessi che lo compongono. È come se il computer procedesse per la sua strada attraverso la foresta del vostro programma, compiendo brevi movimenti tra un albero e l'altro; ma se voi non riuscite a vedere tutta la foresta insieme, come fate ad essere sicuri che il computer è sulla strada giusta?

L'algoritmo e i dati

"Algoritmo" è un termine che sta ad indicare il modo migliore per risolvere un determinato problema. Le soluzioni di solito sono molteplici ed è compito vostro cercare quella più congeniale al vostro computer. Ad esempio, il loop FOR NEXT verrà impiegato quando il programma necessiterà di un algoritmo in grado di ripetere più volte la stessa operazione. Nello stesso modo, anche i dati vanno trattati col metodo più regolare possibile; cercate le somiglianze fra differenti voci di dati e sfruttatele per semplificare il programma. Per esempio, il successo di molti programmi finanziari è legato al fatto di avere i dati organizzati in modo da essere manipolati con facilità.

Rapidità di apprendimento

Per acquisire pratica dovete scrivere voi stessi dei programmi e studiarne altri non fatti da voi, per confrontare il vostro metodo con quello altrui.

Velocità, spazio, chiarezza

Sono tre fattori normalmente incompatibili tra di loro; infatti, i programmi più sono veloci più occupano spazio in memoria e viceversa.

Aumentare la velocità o ridurre la quantità di memoria occupata, spesso comporta l'adozione di espedienti che rendono il listato di difficile interpretazione.

Lo Spectrum è piuttosto veloce e possiede abbastanza memoria per la maggior parte delle applicazioni. È comunque meglio, per le prime volte, anteporre la chiarezza dei contenuti alla velocità e allo spazio.

La terza versione

In pratica, scrivere un programma è più complicato che partire con un'idea generica (ampia) e gradualmente completarla con dettagli fino ad avere il programma pienamente efficiente. Spesso accade che a metà del lavoro si presenti un problema che vi costringe a tornare indietro.

Inoltre, di solito ci si accorge, una volta completato il programma, che poteva essere scritto molto meglio. Per questa ragione i programmi vengono riscritti generalmente due o tre volte e si impiega quindi per stenderli molto più tempo del previsto.

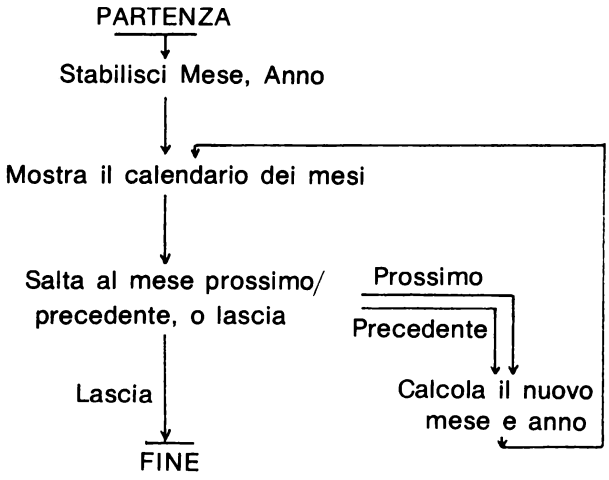
Ad alti livelli ciò è inevitabile; un progetto è un processo iterativo; tuttavia è forse possibile evitare questa perdita di tempo con una corretta "educazione" fin dai primi passi.

Calendario: un esempio

Come sempre, gli esempi valgono più delle parole: vediamo di illustrare quanto esposto creando un programma che sia in grado di produrre un calendario.

In primo luogo vediamo che cosa deve fare il programma: deve visualizzare il calendario di un intero anno, ma, poiché questo non entra sullo schermo, si accontenterà di visualizzare un mese. Dopo aver precisato l'anno, verrà chiesto all'utente di inserire il mese da presentare. Per rendere il tutto più agile, si può includere un'opzione che permetta di avanzare o retrocedere di un mese alla volta sullo schermo. Per evitare di generare un loop senza fine, questa opzione deve mettere in grado l'operatore di fermare dolcemente il programma.

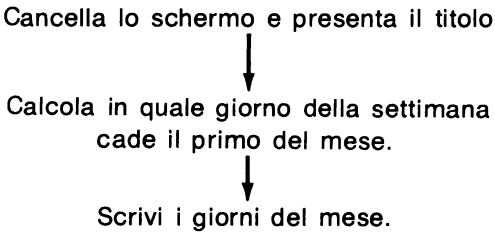
La miglior struttura del programma dovrebbe essere qualcosa di questo genere:



Il blocco: "Stabilisci Mese, Anno" è abbastanza semplice da non richiedere ulteriori espansioni prima della stesura delle righe effettive di programma. Per semplicità, alla richiesta del computer, il mese va inserito come numero (da uno a dodici). Per rendere il programma il più completo possibile, l'ideale sarebbe poter lavorare su tutti gli anni. Pur-

troppo ciò è reso impossibile dal cambio dal calendario "Giuliano" a quello "Gregoriano". Pertanto, poiché tale cambio ha avuto luogo in anni differenti in nazioni differenti, il programma accetta solamente gli anni dal 1752 in poi; in quest'anno, infatti, l'Inghilterra fece tale cambio.

La presentazione dei mesi avviene in tre fasi:



Per calcolare in quale giorno della settimana cade il primo del mese, usate l'algoritmo che segue:

```
LET D=3+Y+INT((Y-1)/4) - INT((Y-1)/100)
LET D=D+ totale giorni dei mesi dell'anno passati
LET D=D-7*INT(D/7)
```

in cui il valore finale di D è il giorno della settimana (0-6) e Y l'anno.

Potete trovare il numero totale dei giorni del mese precedente, per mezzo di un semplice loop FOR NEXT:

```
FOR a=1 TO M-1    LET D=D+ giorni del mese a
NEXT a
```

in cui M è il numero (1-12) del mese da presentare. Notate come questo algoritmo non permetta allo Spectrum di eseguire gli statement relativi al FOR NEXT, se il valore di partenza della variabile di controllo è più elevato di quello finale; nel nostro caso M=1 (Gennaio).

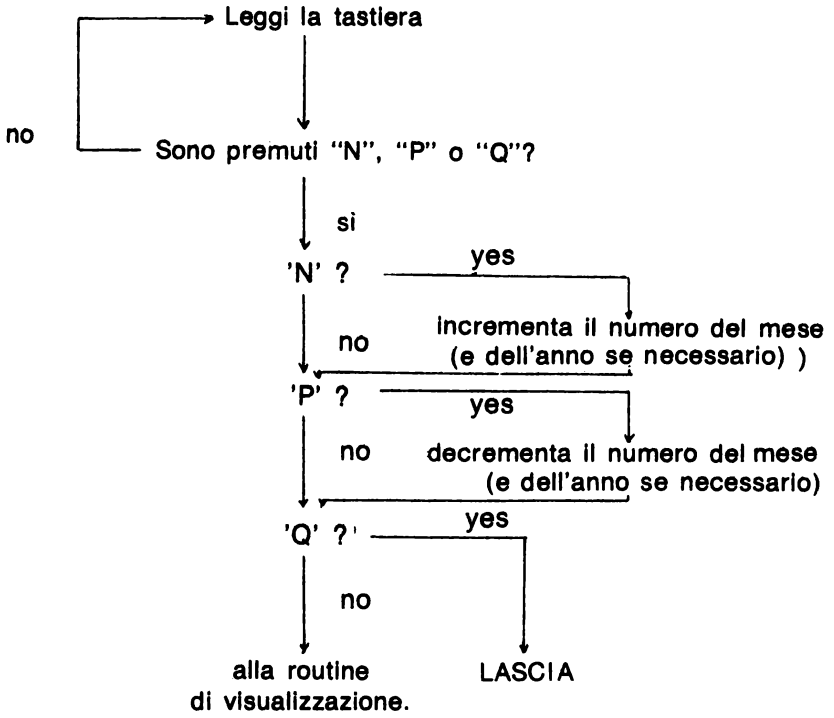
La scrittura dei giorni del mese viene eseguita tramite un secondo loop FOR NEXT che espone i numeri da 1 all'ultimo del mese in esame, con l'aiuto della voce TAB che li allinea sullo schermo nella giusta posizione.

Tornando al programma originale, il secondo blocco riguarda la possibilità di avanzare o retrocedere un mese alla volta oppure di interrompere. Per ottenere il risultato cercato, usate la funzione INKEY$ che

permette all'operatore di agire azionando un tasto solo. I tasti interessati sono:

N per avanzare al mese successivo
P per retrocedere al mese precedente
Q per interrompere.

Perché il programma scorra correttamente, è necessario usare diversi statement IF. Date un'occhiata al diagramma di flusso seguente.



Vediamo ora come i dati vengono inseriti nel programma. Oltre alle variabili semplici, come il giorno, il mese e l'anno, il programma necessita di un elenco del numero dei giorni di ogni mese. Inoltre è meglio se

Il programma dispone del nome reale dei mesi o almeno delle tre lettere più significative per l'abbreviazione; necessita anche di una lista di questi nomi.

Potremo ricorrere a dodici array entro cui sistemare i dodici numeri e i dodici nomi per poi ripescarli con gli statement DATA e READ, oppure compilare una schiera di LET. Visto però l'esiguo numero di dati, non ricorreremo né all'uno né all'altro, ma li memorizzeremo semplicemente come stringhe, usando, per estrarli, le funzioni di sezionamento delle stringhe. Pertanto, nel listato che vedete, m$ rappresenta i dodici nomi abbreviati in tre lettere relativi al mese e n$ i dodici numeri a due cifre relativi al numero di giorni di ogni mese.

```
  10 LET m$="GenFebMarAprMagGiuL
ugAgoSetOttNovDic"
  20 LET n$="312831303130313130313
13031"
 100 INPUT "Anno (>1752) ";y: IF
 y<1753 THEN GO TO 100
 110 INPUT "Mese (1-12) ";m: IF
m<1 OR m>12 THEN GO TO 110
 200 CLS : PRINT AT 4,8;m$(3*m-2
 TO 3*m);TAB 18;y
 210 PRINT AT 7,2;"Dom Lun Mar M
er Gio Ven Sab"
 220 LET n$(3 TO 4)="28"
 230 IF y=4*INT (y/4) AND y<>100
*INT (y/100) THEN LET n$(3 TO 4)
="29"
 240 LET d=3+y+INT ((y-1)/4)-INT
 ((y-1)/100)
 250 FOR a=1 TO m-1: LET d=d+VAL
 n$(2*a-1 TO 2*a): NEXT a
 260 LET d=d-7*INT (d/7)
 300 PRINT AT 9,0;
 310 FOR a=1 TO VAL n$(2*m-1 TO
2*m)
 320 LET p=2+4*(a+d-7*INT ((a+d)
/7))
 330 PRINT TAB p;a;
 340 NEXT a
 400 PRINT AT 20,0;"Premi i tast
i 'n' 'p' o 'q'"
 410 LET i$=INKEY$
 420 IF i$<>"n" AND i$<>"p" AND
i$<>"q" THEN GO TO 410
 430 IF i$="n" THEN LET m=m+1
 440 IF m>12 THEN LET m=1: LET y
=y+1
 450 IF i$="p" THEN LET m=m-1
 460 IF m<1 THEN LET m=12: LET y
=y-1
 470 IF i$<>"q" THEN GO TO 200
```

Le righe 10,20 comprendono i dati fissi.

Le righe 100, 110 accettano il mese e l'anno dall'operatore e ne controllano la validità.

Le righe 200-260 scrivono le intestazioni di ogni mese e calcolano in quale giorno della settimana cade il primo del mese.

Le righe 300-340 scrivono il numero dei giorni del mese.

Le righe 400-470 permettono all'operatore di avanzare o retrocedere di un mese oppure di terminare.

```
          Dic          1999

  Dom  Lun  Mar  Mer  Gio  Ven  Sab

                 1    2    3    4
   5    6    7    8    9   10   11
  12   13   14   15   16   17   18
  19   20   21   22   23   24   25
  26   27   28   29   30   31

Premi i tasti 'n' 'p' o 'q'
```

DEBUGGING

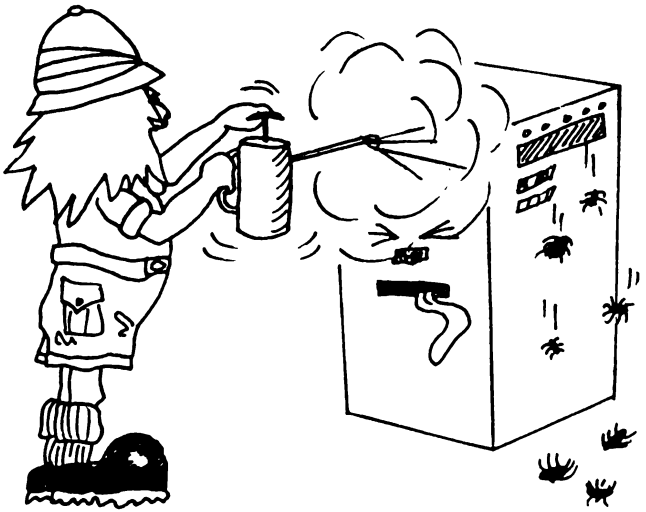
Può succedere che il programma da voi scritto non giri, quando date il RUN per la prima volta. Non scoraggiatevi perché potreste anche non essere voi la causa del mancato funzionamento, in quanto accade spesso che i programmi siano affetti da "bug", che noi potremmo chiamare "malefiche bestie nere". Il processo che ha la funzione di stanare i "bug" passa sotto il nome di "de-bugging".

Questo è un procedimento di deduzione logica che consiste nell'andare a ritroso dal punto sbagliato per trovare che cosa ha condotto all'errore. Le tecniche per fare ciò consistono:

— nell'aggiungere dei PRINT extra posizionati in modo da rilevare se le principali variabili lavorano correttamente e per vedere quale direzione sta prendendo il programma (è questa una buona ragione per lasciare degli intervalli fra i numeri di riga);

- nell'aggiungere STOP nei punti strategici (programmatori sofisticati faranno uso di statement IF-THEN e STOP che bloccano l'esecuzione del programma nei punti richiesti);
- nell'eliminare provvisoriamente dal programma righe critiche, aggiungendo REM appena dopo il numero di riga.

Se il programma si blocca, per qualsiasi ragione, usate gli statement LET e PRINT in modo immediato (senza il numero di riga) per esaminare ed eventualmente cambiare il valore delle variabili. Fatto ciò, potrete far ripartire il programma dallo stesso punto, dando il comando CONT, oppure farlo partire da una riga "n" dando il GO TO n.



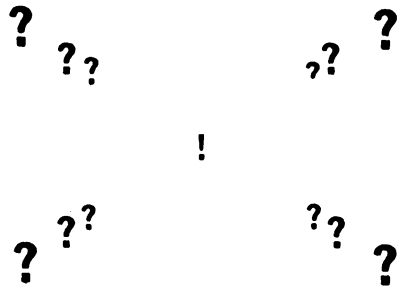
CAPITOLO 7
È BELLO RISCHIARE

I computer sono macchine fondamentalmente precise. Se chiedete loro quanto fa sei per sette, vi sentirete sempre rispondere 42: questa è la loro grande forza.

Ma a volte può essere anche un handicap specialmente quando questa precisione si scontra con gli esseri umani (su argomenti in cui sia necessaria la creatività). Quello che manca al computer, quando state giocando una partita oppure quando state realizzando un'opera d'arte, è appunto la fantasia: le sue risposte saranno sempre meccaniche e ripetitive.

Così, il BASIC dello Spectrum mette a disposizione la funzione RND.

Essa stabilisce una frazione decimale compresa tra zero e uno, o meglio tra 0.00000 e quasi uno, cioè 0.99999. Naturalmente lo Spectrum non decide coscientemente quale numero casuale scegliere ma, come descritto nel manuale, genera una lunghissima sequenza di numeri in modo che non vi sia un legame apparente tra un numero e quello successivo. Ogni volta che usate la funzione RND, il computer mostra il numero successivo a quello raggiunto in quel preciso istante. Più esattamente dovremmo chiamarli numeri "pseudo casuali", ma in effetti sono più che sufficienti per le nostre applicazioni pratiche.



CLIVE

Il programma che presentiamo, denominato "Clive", è in grado di estrarre, come in una lotteria, un numero vincente posto tra un limite inferiore ed uno superiore, stabiliti direttamente da voi. Estrarre il numero è facile, notate la riga 220, quasi tutto il resto del programma è destinato a renderlo più interessante e a creare un senso di attesa e di eccitazione durante la scelta del numero. A questo fine appaiono 100 numeri random, che sfilano fino a fermarsi su quello scelto, in corrispondenza del quale si mette a lampeggiare il bordo. Variando il colore, BRIGHT, FLASH e includendo i comandi di suono, il gioco si fa più divertente: provate!

```
  100 RANDOMIZE
  110 CLS
  120 INPUT "numero piu' basso ";
a;" piu' alto ";b
  130 IF a>=b THEN GO TO 120
  200 FOR c=1 TO 100
  210 CLS
  220 PRINT AT 10,12;INT (a+(1+b-
a)*RND)
  230 PAUSE 1+c/4
  240 NEXT c
  300 FOR c=1 TO 10
  310 BORDER 0: PAUSE 10
  320 BORDER 7: PAUSE 10
  330 NEXT c
  400 PRINT AT 21,0;"Altro tentat
ivo ? "
  410 LET d$=INKEY$: IF d$="" THE
N GO TO 410
  420 IF d$="y" OR d$="Y" THEN RU
N
```

100-130 Partenza ed inserimento dei limiti alto e basso con relativa verifica.

200-240 Comparsa dei 100 numeri random.

300-340 Presentazione del numero vincente.

400-420 Ripetizione del gioco con un altro numero vincente; viene inoltre evitato il rapporto "0/..", che può trarre in inganno prima che abbiate finito di usare il programma.

RND

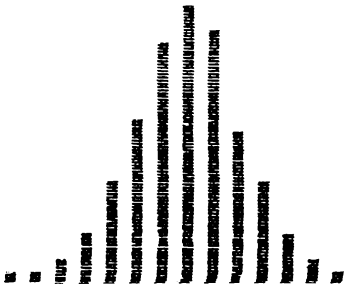
Mostreremo ora la funzione RND dello Spectrum: vale la pena di dare uno sguardo alla casualità dei numeri che essa fornisce.

Vediamo di effettuare un test con il programma che segue, il quale impiega RND per produrre una serie di interi casuali, compresi tra 1 e 16. Viene tenuto il conteggio di quante volte esce ogni intero e, nello stesso tempo, viene disegnato l'istogramma del risultato.

```
 10 DIM a(16)
 20 LET s=.1
 30 RANDOMIZE
100 LET b=1+INT (16*RND)
110 LET a(b)=a(b)+s
120 IF a(b)>21 THEN STOP
130 PRINT AT 21-a(b),2*b-1;"■"
140 GO TO 100
```

La variabile "s" è usata come fattore scalare ed il minimo valore gli viene assegnato alla linea 20. Successivamente si accumulano altri numeri fino a raggiungere la sommità dello schermo in corrispondenza della quale il programma si ferma.

L'istogramma rappresentato in figura si ottiene modificando la versione originale del programma dando ai numeri random dall'1 al 16 una distribuzione "Gaussiana". Provateci anche voi sostituendo la linea 100 con le quattro linee che seguono:



```
 100 LET c=0: FOR d=1 TO 12: LET
 c=c+RND: NEXT d
 102 LET sdev=2: LET mean=7.5: L
ET b=INT (1+(c-6)*sdev+mean)
 104 IF b<1 THEN LET b=1
 106 IF b>16 THEN LET b=16
```

(Le righe 104 e 106 impediscono al programma di bloccarsi qualora vengano dati dei valori troppo alti o troppo bassi).

La funzione è basata sulla formula

```
GRND = (SRND-6)* SDEV+MEAN
```

dove GRND è un numero casuale con distribuzione Gaussiana, SDEV e MEAN i valori necessari a produrre GRND e SRND è la somma dei 12 numeri casuali compresi tra 0 e 1.

CRO??WOR?

Come esempio dell'estrazione di lettere a caso, provate questo programma, scritto per chi si dedica alle parole incrociate o per chi cerca un nome per un nuovo programma. Appena inserita la parola (o la sigla mnemonica), con le lettere in dubbio sostituite da punti interrogativi, il programma riempirà lo schermo, con le parole proposte, mettendo al posto dei punti interrogativi delle lettere a caso, generate dal processo di random. Quando lo schermo risulterà pieno, apparirà la richiesta "scroll ?": premete N oppure BREAK per chiudere, o qualsiasi altro tasto per ottenere un secondo "pieno".

Provate questo programmino, magari impiegando il vostro nome. Ne vedrete delle belle!

```
100 INPUT "Parola ? ";w$
110 FOR a=1 TO LEN w$
120 LET c$=w$(a)
130 IF c$="?" THEN LET c$=CHR$
(65+25*RND)
140 PRINT c$;
150 NEXT a
160 PRINT
170 GO TO 110
```

ROULETTE RUSSA

In alcuni programmi potremmo volere che un evento accada soltanto occasionalmente, non come risultato di determinate operazioni né in risposta a particolari "input", ma semplicemente come conseguenza della funzione RND.

Un esempio tradizionale è la Roulette Russa in cui un certo numero di concorrenti si passa una rivoltella caricata con una sola pallottola. Ognuno di essi si punta alla tempia l'arma e preme il grilletto. La durata della partita dipende dalla fortuna dei singoli partecipanti.

Nella versione del computer è lo Spectrum a decidere quale sia il "colpo" fatale. La linea 340 stabilisce gli scatti innocui quando il numero random prodotto è più alto di 1/6; per dirla in altre parole, uccide, in media, un giocatore ogni sei colpi. Il programma prevede anche l'uso dei "flags", che sono delle variabili non usate per particolari valori, ma per ricordare se un particolare evento è accaduto o no in precedenza. Per esempio l'array p( ) registra i giocatori uccisi durante la partita. Tutti gli elementi di p( ) vengono posti inizialmente a zero dalla istruzione DIM presente alla riga 150, ma quando uno dei concorrenti muore, l'elemento corrispondente dell'array viene portato a 1 dalla riga 360. Tutto questo toglie al programma l'imbarazzo di invitare un giocatore già morto a puntarsi il grilletto al suo turno (riga 220).

In maniera analoga, la variabile "f" decide se continuare la partita, controllando se ci sono giocatori rimasti vivi (righe 200, 230, 410).

```
  100 RANDOMIZE
  110 DIM z$(15)
  120 PRINT AT 10,5;"<<ROULETTE R
USSA>>"
  130 INPUT "Numero di giocatori
?";n
  140 IF n<1 THEN STOP
  150 DIM p(n)
  200 LET f=0
  210 FOR a=1 TO n
  220 IF p(a)=1 THEN GO TO 390
  230 LET f=1
  300 CLS : PRINT AT 6,2;"Giocato
re ";a;" e' il tuo turno"'"di ri
schiare."
  310 PRINT AT 8,27;"▄▄▄";AT 9,3
0;"▀"
```

```
 320 PRINT '"Premi ENTER"'"per f
ar fuoco": INPUT LINE i$
 330 PRINT AT 8,0;z$'z$'z$'z$'z$
 340 IF RND>1/6 THEN PRINT AT 8,
20;"click": GO TO 380
 350 PRINT AT 8,20;" ▌ ▐";AT 9,1
9;"▐▄▄▄▌";AT 9,0;"Sei morto !"
 360 LET p(a)=1
 380 PAUSE 50
 390 NEXT a
 400 IF f=1 THEN GO TO 200
 410 PRINT AT 14,2;"--lieto di a
verti conosciuto--"
```

100-150 Presentazione, scelta del numero dei giocatori. La riga 110 produce una stringa di caratteri di 15 spazi necessaria alla linea 330 per cancellare parte dello schermo.

200-390 Passa a turno la rivoltella tra i giocatori.

400-410 Propone un altro giro se non sono morti tutti, viceversa saluta cordialmente.

Nota; i caratteri tra virgolette alle linee 310 e 350 sono ottenuti usando i tasti:
Linea 310 prima parte:
SHIFT/GRAPHICS 5 SHIFT/3 SHIFT/3 SHIFT/8 SHIFT/8 9
seconda parte:
SYMBOL-SHIFT/7 SHIFT/GRAPHICS SHIFT/8 9
Linea 350 prima parte:
SHIFT/GRAPHICS 4 SPACE SPACE SHIFT/3 9
seconda parte:
SHIFT/GRAPHICS SHIFT/8 SHIFT/2 SHIFT/3 SHIFT/3 SHIFT/2 9

CARTA ALTA; CARTA BASSA

Abbiamo visto come usare RND per estrarre numeri da una gamma limitata di valori numerici. Esaminiamo ora la possibilità di scegliere anche tra altre cose. Ad esempio, in questo programma lo Spectrum estrae carte da gioco, tra le quali figurano anche il Fante (Jack), la Regina (Queen) e il Re (King).

Per ottenere una selezione di questo tipo, è necessario stendere una lista delle voci quindi abbinare loro un numero all'interno della lista.

Nel programma, questa trova posto alla linea 10 e comprende le 13 carte sottoforma di una stringa di 13 caratteri. Lo Spectrum sceglierà una carta generando un numero random da 1 a 13 (vedere la linea 210).

La partita è giocata da un solo giocatore e dal computer che sceglie due carte, ne mostra una e chiede al giocatore di scommettere se l'altra è più alta o più bassa di quella presentata. La quota di partenza è di Lit. 100; il gioco avrà termine o quando avrete esaurito i vostri soldi oppure quando sarete riusciti a prosciugare il banco delle sue Lit. 9999.

```
  10 LET c$="23456789DJQKA"
  20 RANDOMIZE
  30 LET m=100
 100 LET a=4+INT (8*RND)
 110 LET b=1+INT (13*RND): IF b=
a THEN GO TO 110
 120 CLS : PRINT AT 2,0;"Ho scel
to due carte ;"
 130 PRINT AT 5,6;c$(a);TAB 16;"
"
 140 INPUT "La seconda carta e'
piu' alta (h)" "o piu' bassa (l)
della prima ?"; LINE i$
 150 IF i$<>"h" AND i$<>"l" THEN
 GO TO 140
 160 LET d$="piu' alta": IF i$="
l" THEN LET d$="piu' bassa"
 170 PRINT AT 8,0;"Pensi che la
seconda sia ";d$' "Hai £";m
 180 INPUT "Quanto vuoi scommett
ere ?";e: IF e<0 OR e>m THEN GO
TO 180
 190 PRINT "Scommetti £";e: IF e
=0 THEN GO TO 300
 200 FOR p=1 TO 200: NEXT p
 210 PRINT AT 5,16; FLASH 1;c$(b
)
```

```
 220 IF (i$="h" AND b>a) OR (i$=
"l" AND b<a) THEN GO TO 250
 230 LET m=m-e: PRINT AT 13,0;'"
Malasorte": IF m>0 THEN GO TO 27
0
 240 PRINT '"Sei rovinato !!": S
TOP
 250 LET m=m+e: PRINT AT 13,0;'"
Ben fatto"
 260 IF m>9999 THEN PRINT "Hai v
uotato la banca !": STOP
 270 PRINT '"Ora hai £";m
 300 PRINT '''"Premi ENTER per un
 altro tentativo": INPUT LINE i$
: GO TO 100
```

10-30 Assegnazione dei valori iniziali.

100-130 Sceglie due carte (a & b) e mostra la carta a.

140-190 Chiede al giocatore di effettuare la scommessa e ne verifica la validità.

200-270 Mostra la carta b, calcola e presenta i risultati. La linea 200 introduce un piccolo ritardo per generare un po' di suspence.

POESIA HAIKU

È un programma che produce una sequenza senza fine della peggiore poesia "Haiku" (in quartina), prendendo una frase a caso da ciascuno dei quattro gruppi di cui parleremo sotto. Tale programma propone un altro tipo di utilizzazione di voci casuali non numeriche.

Per fare ciò, queste vengono inserite in statement DATA, poi ne viene letto un numero a caso e infine viene usata l'ultima che è stata letta.

I quattro gruppi di frasi vengono tenuti in quattro gruppi di statement di DATA, partendo dalle righe 1000, 1100, 1200 e 1300. Il gruppo da utilizzare viene selezionato ad opera delle righe 100 e 110.

Se volete divertirvi, componete voi stessi la lista di frasi: non è necessario che si tratti di poesia, potete utilizzare spezzoni di articoli, discorsi ufficiali o altro. Con lunghe liste di frasi otterrete risultati più vari; tuttavia, per un corretto funzionamento del programma, ogni gruppo deve contenere lo stesso numero di frasi e la riga 10 va adattata a tale numero.

```
 10 LET m=6
 20 RANDOMIZE
100 FOR a=1000 TO 1300 STEP 100
110 RESTORE a
120 FOR b=1 TO 1+m*RND: READ c$
 NEXT b
130 PRINT c$
140 NEXT a
150 POKE 23692,0
160 PAUSE 200
170 PRINT ""
180 GO TO 100
1000 DATA "Un pesce salta","Il c
ielo e' verde","I cormorani si t
uffano","Due uomini"
1010 DATA "L'ondeggiare del riso
","Gatti"
1100 DATA "Presso lo stagno","Mo
rmorio del vento","I grilli cant
ano"
1110 DATA "Un fulmine","Farfalle
","Lungo il sentiero"
1200 DATA "Con il bue","Fumo in
lontananza","Una mano applaude"
1210 DATA "E' quasi mezzogiorno"
,"Il bambu' cresce","Un piccolo
fiore"
1300 DATA "Stanno arrivando lent
amente","Il nido degli uccelli n
egli alti alberi"
1310 DATA "Succede come ieri","I
 fanciulli sono silenziosi"
1320 DATA "Gli aratri riposano n
ell'erba alta","Gli alberi cresc
ono solitari"
```

10-20 Presentazione dei valori inizlali.

100-140 Stampa una frase da ogni gruppo.

150-180 Permette lo "scroll" allo schermo, pausa per applausi, inizio di una nuova "composizione".

CRYPTO MACHINE

Questo programma è indispensabile alle spie di Roma, di Omsk, o di qualunque altra parte del globo, in quanto trasforma qualsiasi messaggio in un codice indecifrabile.

Per la decodifica, è necessario conoscere la "chiave" usata per la codifica onde sbloccare lo Spectrum. Ogni carattere è codificato (o decodificato) nel messaggio usando un tasto scelto in base alla sequenza generata dalla funzione RND.

L'operatore inserisce il valore di partenza (da 1 a 65535) che viene usato dallo statement RANDOMIZE alla linea 140 per predisporre la sequenza corretta della codifica o della decodifica.

```
 100 INPUT "Battere D per decodi
ficare,E per codificare"; LINE e
$
 110 IF e$<>"e" AND e$<>"E" AND
e$<>"d" AND e$<>"D" THEN GO TO 1
00
 120 INPUT "Inserire il numero (
1-65535) ";k
 130 IF k<1 OR k>65535 THEN GO T
O 120
 140 RANDOMIZE k
 150 INPUT "Battere il messaggio
;"; LINE i$: PRINT i$
 200 FOR a=1 TO LEN i$
 210 LET c=CODE i$(a): IF c<32 O
R c>127 THEN GO TO 250
 220 LET x=96*RND: IF e$="d" OR
e$="D" THEN LET x=-x
 230 LET c=c+x: IF c>127 THEN LE
T c=c-96
 240 IF c<32 THEN LET c=c+96
 250 PRINT CHR$ c;
 260 NEXT a
```

100-130 Istruzioni per codificare e decodificare + tasto di partenza.

140 Dispone il generatore del numero RANDOM al valore di partenza.

150 Accettazione del messaggio (i$).

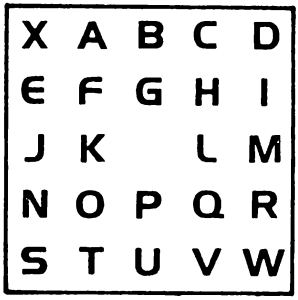
200-260 Codifica/Decodifica e scrittura in successione di ogni carattere. Il numero chiave per ognuno di essi, generato alla linea 220, viene sommato o sottratto al CODE del carattere stesso. Se il risultato esce dal range dei numeri CODE per il numero dei caratteri "normali" (32-127), viene addizionato o sottratto 96 per riportarlo all'interno dell'intervallo.

MIXUP

Qui la funzione RND crea un problema che voi dovrete risolvere.

Mixup è infatti la versione per computer del vecchio gioco della tavoletta con le lettere. Dovrete cioè cercare di riordinare secondo l'ordine alfabetico le 24 lettere racchiuse nel quadrato da 5×5, muovendo una lettera alla volta, entro il quadratino vuoto. Alla partenza, il programma vi presenta le lettere in ordine corretto, dopodiché queste saranno mescolate più o meno a lungo in funzione del grado di abilità. Se nel riordinare impiegherete più di 9999 mosse, lo Spectrum vi esprimerà il suo disgusto!

Il programma lavora direttamente con l'immagine sullo schermo e la funzione SCREEN$ rileva quanto è scritto in una particolare posizione. La linea che delimita l'area di gioco, impedisce di confondere gli spazi esterni col quadratino vuoto interno entro il quale vanno mosse le lettere.



```
  10 DATA 0,-1,0,1,-1,0,1,0
  20 DIM h(4): DIM v(4)
  30 FOR a=1 TO 4: READ h(a),v(a
): NEXT a
 100 PLOT 101,69: DRAW 45,0: DRA
W 0,45: DRAW -45,0: DRAW 0,-45
 110 FOR y=8 TO 12: FOR x=13 TO
17
 120 PRINT AT y,x;CHR$ (12+x+5*y
)
 130 NEXT x: NEXT y
 140 LET x=17: LET y=12: PRINT A
T y,x;" "
 200 RANDOMIZE : LET x1=0: LET y
1=0
 210 INPUT TAB 8;"Abilita' (1-9)
 ?";s: IF s<1 OR s>9 THEN GO TO
210
 220 FOR a=1 TO 5*s
 230 LET c=1+INT (4*RND): LET x2
=x+h(c): LET y2=y+v(c)
 240 IF SCREEN$ (y2,x2)="" OR (x
1=x2 AND y1=y2) THEN GO TO 230
 250 PRINT AT y,x;SCREEN$ (y2,x2
);AT y2,x2;" "
 260 LET x1=x: LET y1=y: LET x=x
2: LET y=y2
 270 NEXT a
 300 FOR m=1 TO 9999: PRINT AT 1
8,12;"Muovi ";m
 310 PRINT AT 20,11;"Lettera ?"
 320 LET i$=INKEY$: IF i$="" THE
N GO TO 320
 330 IF CODE i$>96 THEN LET i$=C
HR$ (CODE i$-32)
 340 FOR a=1 TO 4: IF SCREEN$ (y
+v(a),x+h(a))=i$ THEN GO TO 400
 350 NEXT a
 400 PRINT AT 20,0,,
 410 PRINT AT y,x;SCREEN$ (y+v(a
),x+h(a))
 420 LET x=x+h(a): LET y=y+v(a):
 PRINT AT y,x;" "
 500 FOR p=8 TO 12: FOR q=13 TO
17
 510 IF p=12 AND q=17 THEN GO TO
 600
 520 IF SCREEN$ (p,q)<>CHR$ (12+
q+5*p) THEN GO TO 540
 530 NEXT q: NEXT p
 540 NEXT m
 600 PRINT AT 20,10;"*FINE*"
```

10-30 Predispone gli array h( ) e v( ) in modo tale che contengano le componenti orizzontali e verticali delle quattro direzioni possibili di movimento.

100-140	Disegna il quadrato con le 24 lettere in ordine.
200-210	Chiede il grado di abilità.
220-270	Mescola le lettere un numero di volte proporzionale a "s"; "x" ed "y" sono la posizione attuale del quadratino vuoto, "xl" e "yl" la posizione precedente e "x2" e "y2" la posizione successiva; le posizioni vengono ricordate per impedire che il programma effettui un rimescolamento solo tra due quadrati.
230-240	Prende una delle quattro posizioni circostanti e controlla che non sia stata usata l'ultima volta.
250-260	Muove la lettera nel quadratino vuoto e aggiorna xl e yl.
300-540	Esegue il loop ad ogni mossa del giocatore.
300-330	Presenta il numero della mossa, prende in considerazione la lettera che il giocatore intende spostare e se necessario la converte in maiuscola.
340-350	Si accerta che la lettera scelta sia adiacente al quadratino vuoto; in caso contrario, attende una nuova lettera.
400	Cancella la scritta "Lettera?" dallo schermo.
410-420	Sposta la lettera scelta nel quadratino vuoto.
500-530	Controlla se le lettere sono nel giusto ordine; se lo sono salta alla linea 600, se non lo sono retrocede alla 300.
600	Fine del gioco.

Ricordatevi di battere lo spazio tra gli apici delle linee: 140, 250 e 420.

CAPITOLO 8
PEEK, POKE, IN e OUT

Lo Spectrum può indirizzare fino a 65536 (64 K) locazioni di memoria. Le prime 16384 sono occupate dalla ROM, le rimanenti 16384 (nella versione a 16K) o 49152 (nella versione a 48K) sono a disposizione della RAM. Come descrive anche il manuale di programmazione, è possibile esplorare tutte le 65536 locazioni, per mezzo dell'istruzione PEEK, ed anche cambiare il contenuto di quelle relative alla RAM, tramite il comando POKE.

Ognuna delle locazioni, sia che si trovi in RAM sia che si trovi in ROM, contiene un "byte" formato da 8 bit i quali possono essere espressi a seconda del contesto:

- un numero binario a 8 bit nel range 00000000-11111111;
- un numero esadecimale nell'intervallo 00-FF;
- uno qualunque dei caratteri del set riportato nell'appendice del manuale dello Spectrum. Possono essere "keyword", simboli grafici oppure caratteri di controllo della stampa, come caratteri alfanumerici;
- numeri decimali compresi nel range 0-255;
- istruzioni in codice macchina per il micro processore Z80.

Sia PEEK che POKE usano numeri in forma decimale. Molto spesso ci si trova di fronte alla necessità di leggere o di variare il contenuto di una particolare locazione di memoria. In questo caso la funzione da usare è POKE con la quale è possibile sia memorizzare in RAM nuovi va-

lori che leggere e sostituire quelli delle Variabili di Sistema che stabiliscono il comportamento dello Spectrum. Ad esempio, lo scroll continuo dello schermo (senza l'apparire della solita richiesta "scroll?"), è possibile alterando la memoria nel modo che segue:

```
POKE 23692,0
```

Nel corso della lettura di questo libro troverete altri esempi sull'uso di PEEK e di POKE con le variabili di sistema; potrete inoltre trovarne un sunto nell'Appendice 1.

PEEK legge, come già detto, il contenuto della ROM, pertanto, se volete divertirvi un po' provate ad entrare nell'area della ROM contenente la serie di punti necessaria alla formazione dei caratteri normali (codici da 32 a 127). Ogni carattere, come spiegheremo più in dettaglio, viene presentato sullo schermo, o stampato dalla ZX printer, come un array di 8×8 punti elementari chiamati "pixel" i quali danno forma ad ogni carattere, per l'intervento di 8 byte (8×8 bit) siti nella ROM. Poiché la Variabile di Sistema CHARS (23606,7) contiene un numero che è 256 volte al disotto dell'indirizzo ROM di partenza della tavola dei punti, possiamo calcolare l'indirizzo del primo di questi byte nel seguente modo:

Contenuto di CHARS + 8 * CODE del carattere

Prendiamo ad esempio il carattere 0 di cui facciamo scrivere allo Spectrum il valore degli 8 byte relativi per mezzo della routine

```
10 PRINT "ADDR","CONTENTS"
20 LET s=PEEK 23606+256*PEEK 23607+8*CODE "0"
30 FOR a=s TO s+7: PRINT a,PEEK a: NEXT a
```

```
ADDR          CONTENTS
15744         0           00000000
15745         60          00111100
15746         70          01000110
15747         74          01001010
15748         82          01010010
15749         98          01100010
15750         60          00111100
15751         0           00000000
```

La colonna di destra è stata aggiunta per mostrare il contenuto di ogni locazione sottoforma di un numero binario da 8 bit. Immaginate, infatti, di sostituire gli "1" con dei quadratini neri e gli "0" con spazi bianchi e vedrete apparire la forma del carattere.

STENDARDI

Le serie di punti site nella ROM sono qui impiegate per ottenere scritte gigantesche simili a stendardi. Se siete in possesso di una stampante per lo ZX, otterrete messaggi che verranno stampati per il lungo, in modo da poterne regolare la lunghezza, e la cui altezza corrisponde al formato della carta.

Controllare l'ampiezza dei caratteri: inserendo il valore "4", il rapporto lunghezza/altezza rimane inalterato, mentre, ad esempio, col "2" i caratteri saranno alti il doppio della loro larghezza. Il programma può anche manipolare caratteri grafici qualora il CODE sia compreso tra 144 e 164.

In questo caso si va a leggere la Variabile di Sistema UDG (23675,6) per cercare la partenza in RAM delle serie di punti che definiscono i caratteri grafici.

```
 100 INPUT "Inserisci il messagg
io";"m$: PRINT m$
 110 INPUT "Larghezza del caratt
ere ? ";w: PRINT "Larghezza ";w
 120 FOR a=1 TO LEN m$
 130 LET b=PEEK 23606+256*PEEK 2
3607+8*CODE m$(a)
```

```
 140 IF CODE m$(a)>143 AND CODE
m$(a)<165 THEN LET b=PEEK 23675+
256*PEEK 23676+8*(CODE m$(a)-144
)
 150 DIM x(8): FOR c=0 TO 7: LET
 x(8-c)=PEEK (b+c): NEXT c
 200 LET d=128
 210 FOR c=0 TO 7
 220 FOR e=1 TO w
 230 FOR f=1 TO 8: IF x(f)<d THE
N GO TO 250
 240 LPRINT TAB 4*f-4;"    ";: I
F e=w THEN LET x(f)=x(f)-d
 250 NEXT f: LPRINT
 260 NEXT e: LET d=d/2
 270 NEXT c
 280 NEXT a
```

Tra gli apici presenti alla linea 140 vanno inseriti quattro spazi "inverse video".

100-110 Stesura del messaggio e definizione della sua ampiezza.

120-280 Loop principale del programma. Viene eseguito ad ogni carattere.

130-140 Assegna a "b" l'indirizzo del primo byte, relativo alla serie di punti del carattere da stampare.

150 Inserisce gli 8 byte della serie di punti negli 8 elementi x( ).

200-270 Routine di scrittura carattere.

210-270 Loop eseguito per ogni colonna della serie di punti.

220-260 Loop che ripete la scrittura di una colonna per un numero "w" di volte.

230-250 Loop di scrittura di una colonna.

L'algoritmo usato per convertire il valore decimale (0-225) dato da PEEK nei singoli bit binari, è il seguente:

```
LET d=128
FOR c=0 TO 7
IF x>=d THEN PRINT "Bit ";7-c;" e' 1" : LET x=x-d
LET d=d/2
NEXT c
```

dove "x" è appunto il valore ottenuto dallo statement di PEEK.

È stata provata anche una versione che impiegava l'operatore "1" ma era incredibilmente lenta.

OROLOGIO

Questo programma usa la Variabile di Sistema FRAMES (la cui locazione RAM è 23672/3/4) per procurarsi una precisa frequenza di clock per l'orologio digitale. FRAMES, come vi sarete accorti, è una variabile a tre byte incrementata una volta ogni cinquantesimo di secondo (ogni sessantesimo per la versione U.S.A.). Gran parte del programma è rivolto allo sviluppo del display che possiede cifre quattro volte più grosse del normale. Ciò si ottiene eseguendo dei PEEK alle serie di punti nella ROM e usando l'informazione per mostrare ogni cifra come un array 4×4 dei caratteri grafici standard (codice 128-143). Poiché l'ingrandimento diretto dei caratteri è una funzione piuttosto lenta e poiché il programma necessita di una certa velocità per poter mostrare una cifra al secondo, è necessario memorizzare a priori, in un array di stringhe, i caratteri grafici interessati che potranno così essere scritti velocemente prima di iniziare la simulazione dell'orologio. Ciò significa che, alla partenza del programma, dovrete attendere una ventina di secondi con lo schermo vuoto prima di poter inserire, su richiesta, l'ora esatta.

```
  10 GO TO 1000
 100 LET t=PEEK 23672+256*PEEK 2
3673+65536*PEEK 23674
 110 LET t1=PEEK 23672+256*PEEK
23673+65536*PEEK 23674: IF t1>t
THEN LET t=t1
 120 LET t=INT (t/50)
 130 LET h=INT (t/3600): LET t=t
-3600*h
 140 LET m=INT (t/60): LET s=t-6
0*m
 150 IF h>12 THEN LET h=h-12: GO
 TO 1220
 160 LET h$=STR$ h: LET m$=STR$
m: LET s$=STR$ s
 170 LET p$=h$+":"+m$+":"+s$
 200 FOR a=1 TO 8: LET b=4*a-4
 210 LET c=CODE p$(a)-CODE "0"+2
: IF c<1 OR c>12 THEN LET c=1
 220 PRINT AT 9,b;c$(c,1);AT 10,
b;c$(c,2);AT 11,b;c$(c,3);AT 12,
b;c$(c,4)
 230 NEXT a
 240 GO TO 100
1000 REM messa a punto
1100 DIM c$(12,4,4): DIM h$(2):
DIM m$(2): DIM s$(2)
```

```
1110 FOR a=2 TO 12: LET b=PEEK 2
3606+256*PEEK 23607+8*(CODE "0"+
a-2)
1120 FOR c=1 TO 4: LET d=PEEK (b
+2*c-2): LET e=PEEK (b+2*c-1)
1130 LET f=64
1140 FOR g=1 TO 4
1150 LET d1=INT (d/f): LET d=d-f
*d1
1160 LET e1=INT (e/f): LET e=e-f
*e1
1170 LET c$(a,c,g)=CHR$ (128+d1+
4*e1): LET f=f/4
1180 NEXT g: NEXT c: NEXT a
1200 INPUT "Set:ore ; ";h;TAB 7;
"min ; ";m;TAB 7;"sec : ";s
1210 INPUT "Premi ENTER per far
partire l'orologio "; LINE a$
1220 LET t=50*(s+60*m+3600*h)
1230 LET t1=INT (t/65536): LET t
=t-65536*t1
1240 LET t2=INT (t/256): LET t=t
-256*t2
1250 POKE 23674,t1: POKE 23673,t
2: POKE 23672,t
1260 GO TO 100
```

10	La parte di inizializzazione del programma viene messa alla fine del listato (alla linea 1000) per aumentare la velocità del loop principale.
100-240	Loop principale del programma eseguito continuamente per aggiornare il display.
100-120	Ottiene l'ultimo tempo in secondi eseguendo un doppio PEEK alla variabile FRAMES e prendendo il risultato più alto.
130-140	Conversione in secondi, minuti e ore.
150	Reset dell'orologio dopo 12:59:59.
160-170	Costruzione di una stringa da 8 caratteri (p$) contenente il tempo sotto forma di ore: minuti: secondi.
200-230	Loop per mostrare ogni carattere di p$.
210	Calcolo dell'array (c$) da usare. Scrittura di uno spazio se il carattere in p$ non è 0-9 oppure:
220	Scrive quattro stringhe di 4 caratteri (da c$) visualizzando una cifra sullo schermo.
1000	Partenza della routine iniziale. Vedere la linea 10.

1100-1180 Stabilisce l'array c$ e lo riempie con simboli grafici appropriati. C$ contiene un array di 4×4 caratteri grafici per ognuno dei dodici simboli da mostrare. Il primo di tali simboli è uno spazio, gli altri undici corrispondono alle cifre da 0 a 9 e ai due punti. Un'estensione di questa tecnica può anche essere applicata a dei giochi, usando un range di caratteri più vasto.

1200-1250 Accetta il tempo impostato dall'utente e predispone di conseguenza la variabile FRAMES.

SOMME VELOCI

Il programma che segue usa la variabile FRAMES per misurare il tempo impiegato a risolvere una serie di dieci semplici addizioni o sottrazioni. Alla fine viene dato il numero delle risposte esatte e il tempo totale.

```
 100 RANDOMIZE : LET s=0
 110 POKE 23672,0: POKE 23673,0:
 POKE 23674,0
 200 FOR g=1 TO 10: CLS : PRINT
AT 4,10;"Domanda ";g;AT 10,10;
 210 IF RND<.5 THEN GO TO 400
 300 LET a=INT (90*RND)+10: LET
b=INT (90*RND)+10
 310 LET c=a+b: PRINT a;" + ";b;
" = ";
 320 GO TO 500
 400 LET a=INT (90*RND)+10: LET
b=INT (90*RND)+5
 410 IF b>=a THEN GO TO 400
 420 LET c=a-b: PRINT a;" - ";b;
" = ";
 500 INPUT LINE a$: IF a$="" THE
N GO TO 500
 510 FOR x=1 TO LEN a$: IF a$(x)
<"0" OR a$(x)>"9" THEN GO TO 500
 520 NEXT x
 530 LET ans=VAL a$: PRINT ans
 540 IF ans=c THEN LET s=s+1: PR
INT AT 14,12;"ESATTO"
 550 IF ans<>c THEN PRINT AT 14,
12;"SBAGLIATO"
```

```
 560 PRINT AT 18,7;"Punteggio ";
s;" su ";g
 570 FOR t=1 TO 200: NEXT t
 580 NEXT g
 600 LET t=PEEK 23672+256*PEEK 2
3673+65536*PEEK 23674
 610 LET t1=PEEK 23672+256*PEEK
23673+65536*PEEK 23674
 620 IF t<t1 THEN LET t=t1
 630 PRINT AT 4,0,,AT 14,0,,
 640 PRINT AT 10,5;"Tempo impieg
ato ";INT (t/50);" secondi"
```

100-110	Azzera FRAMES il cui valore è usato da RANDOMIZE prima del CLEAR.
200-580	Loop principale eseguito ad ogni domanda.
210	Decide se proporre un'addizione o una sottrazione.
300-320	Propone un'addizione.
400-420	Propone una sottrazione.
500-530	Questa routine accetta la risposta come stringa, verifica che questa contenga solamente cifre da 0 a 9 e la converte in numero.
540-560	Aggiorna il punteggio e scrive il commento in funzione della risposta.
570	Breve ritardo per ottenere il quale non è possibile usare PAUSE, perché quest'ultimo potrebbe essere interrotto dalla pressione accidentale di un tasto.
600-620	Calcola il tempo trascorso eseguendo due PEEK successivi al FRAMES e prendendo il valore più alto.
630-640	Cancella parte del display e mostra il tempo impiegato.

SPAZIO I/O

Per manipolare le 65536 locazioni di memoria, la CPU Z80 può accedere ad altrettanti indirizzi di I/O (Input-Output = ingresso-uscita), numero che però è notevolmente inferiore in conseguenza del tipo di hardware su cui lo Spectrum opera. L'accesso a queste locazioni è possibile tramite i comandi BASIC:IN e OUT che agiscono come PEEK e POKE, ma che, a differenza di questi, sono appunto rivolti alle locazioni I/O anziché alla memoria. Con espansioni di memoria esterne le istruzioni ritornano alla loro prima funzione. Il comando IN si può usare in un modo del tutto particolare, facendogli leggere la tastiera meglio di come faccia la funzione INKEY$, la quale non rileva niente se vengono premuti due tasti contemporaneamente o se viene azionato il tasto SHIFT da solo. Anche se la maggior parte delle volte ciò non comporta problemi, in particolari situazioni, come nell'uso di programmi di giochi, la funzione INKEY$ può essere insufficiente. Facendo interrogare la tastiera da IN si superano tali limitazioni con la semplice aggiunta di una routine per sapere esattamente quale o quali tasti sono stati schiacciati. Per contro, il comando IN non è dotato, come INKEY$, di adeguato ritardo per ottenere un effetto antirimbalzo, per cui è possibile che durante il suo azionamento si generino on-off consecutivi con strane conseguenze per il vostro programma.

I risultati ottenuti da un'istruzione IN rivolta alla tastiera si possono calcolare nel modo seguente:

- 1 azionando il tasto esterno di una mezza riga
- 2 azionando il tasto seguente di una mezza riga
- 4 azionando il tasto centrale di una mezza riga
- 8 azionando il tasto seguente di una mezza riga
- 16 azionando il tasto interno di una mezza riga.

Per esempio IN 32766 porta
255 se non viene premuto alcun tasto da "B" a "SPACE".
254 se viene premuto "SPACE".
253 se viene premuto "SIMBOL SHIFT".
231 se vengono premuti contemporaneamente "B" e "N".

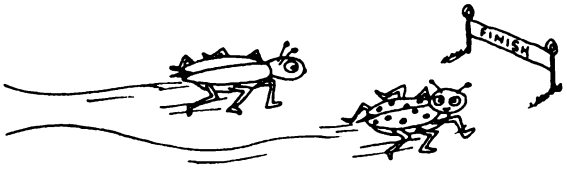
TEMPO DI REAZIONE

Questa è una semplice gara tra due giocatori che devono agire ognuno su un proprio tasto ("B" per quello a sinistra, "SPACE" per l'altro), il più velocemente possibile, all'apparire di una stella al centro dello schermo. I giocatori hanno dieci tentativi a disposizione e il punteggio viene costantemente aggiornato. Se entrambi premono contemporaneamente (è difficile) i relativi tasti, non viene dato, per quella prova, nessun punteggio. I due tasti vanno sempre scelti nella medesima mezza-riga poiché devono essere presi in esame da un solo statement di IN.

```
  10 RANDOMIZE   LET l=0: LET r=
0
 100 FOR a=1 TO 10
 110 FOR b=100 TO 100+300*RND
 120 IF IN 32766<>255 THEN PRINT
 AT 15,10;"VIA LE DITA": BEEP .2
,30: CLS : GO TO 110
 130 NEXT b
 140 PRINT AT 15,15;"*"
 150 LET i=IN 32766: IF i=255 TH
EN GO TO 150
 160 LET s=-24: IF i=239 THEN LE
T l=l+1: LET s=0
 170 IF i=254 THEN LET r=r+1: LE
T s=12
 180 PRINT AT 15,0;l;TAB 15;" ";
TAB 30;r: BEEP 1,s
 190 NEXT a
```

10	Predisposizione dei valori iniziali.
100-190	Attesa casuale. Si accerta che nessuno dei due contendenti agisca prima del tempo.
140	Disegna la stella.
150	Attesa per l'intervento nella mezza riga "B" - "SPACE".
160-170	Calcolo del risultato.
180	Presenta il punteggio aggiornato ed emette un suono appropriato.

Nota: lo statement BEEP, che tratteremo più avanti, è stato introdotto per rendere più attraente la gara e per introdurre un ritardo che sarebbe stato impossibile realizzare con PAUSE, che potrebbe venire interrotto dalla pressione accidentale di un tasto.

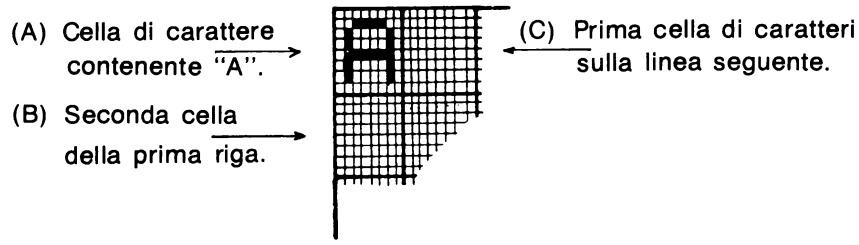


CAPITOLO 9
BELLE ARTI

Uno dei pregi più accattivanti dello Spectrum sta nella sua capacità di elaborare facilmente disegni in alta risoluzione grafica. Nel capitolo 4 abbiamo imparato come disegnare un grafico per mezzo del comando PRINT AT, grafico con una risoluzione di 32 punti orizzontali e 22 verticali. Usando l'alta risoluzione grafica avrete a disposizione ben 256 punti orizzontali e 176 verticali in grado di fornire dettagli al pari di un normale apparecchio televisivo. Per capire come lavora l'alta risoluzione grafica, è necessario conoscere il modo in cui lo Spectrum genera l'immagine sul video. Pertanto, prima di affrontare i programmi, eccovi qualche accenno sull'argomento.

L'intera area del display, comprese le linee in basso riservate normalmente ai commenti, ai comandi INPUT e all'editing, è suddivisa in 24×32 settori ognuno dei quali (character cell) è ulteriormente formato da un array di 8 "pixel" (abbreviazione di "picture element" = elemento di figura). Il carattere si concretizza sullo schermo attribuendo ad alcuni pixel il colore nero ad altri il colore bianco (tratteremo i colori in un ulteriore capitolo).

Volendo scrivere nell'angolo in alto a sinistra dello schermo la lettera "A", tracciate come segue la mappa dei pixel.

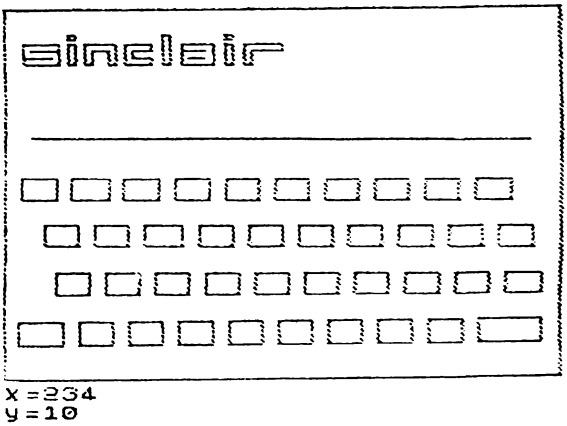


Contando anche le ultime due linee in basso destinate ai commenti e alla scrittura, vi sono a disposizione 22×32 e 22×8×32×8 = 176×256 pixel.

È disponibile un'area totale larga 256 pixel ed alta 176.

Ciascuno di questi punti è pilotato dai comandi di HRG (Hight Resolution Graphics=alta risoluzione grafica) PLOT, DRAW e CIRCLE i quali possono scegliere entro le gamme 0-175 e 0-255 anziché tra 1-176 e 1-256. In definitiva questi piccoli pixel vengono usati sia per ottenere l'HRG che per formare i caratteri. Nel primo caso, i comandi agiscono direttamente su ogni singolo pixel, attribuendogli il colore bianco oppure nero (più precisamente i colori "INK" o "PAPER" come vedremo tra poco), nel secondo il comando PRINT interessa un array di 8×8 pixel disposti in funzione del carattere.

FIGURE



Il prossimo programma vi permetterà di disegnare figure sullo schermo e deve essere inteso come esercizio per imparare il funzionamento dello statement PLOT x,y e delle coordinate X-Y. Viene anche introdotto l'uso del PLOT INVERSE 1; x,y per cancellare l'ultimo punto disegnato.

Il disegno si esegue usando i tasti 5,6,7,8 per spostare attraverso lo schermo il piccolo cursore lampeggiante. Per aiutarvi, nel caso voleste eseguire un disegno di alta precisione, vengono presentate nell'angolo in basso a sinistra dello schermo le coordinate correnti. La cancellazione è possibile azionando assieme ai tasti sopracitati lo SHIFT e ripercorrendo il tratto errato. Una volta terminata l'opera d'arte, date SHIFT-BREAK per interrompere il programma ed, eventualmente, COPY per la stampa su ZX printer. Se invece desiderate memorizzarlo su nastro, date:

```
SAVE "nome" SCREEN$
```

per poterlo richiamare in futuro.

Se, una volta fermato il programma, vi doveste accorgere che il disegno è incompleto, non date il RUN (cancellereste lo schermo), ma per richiamarlo date il CONTINUE.

Nota bene: il programma è strutturato come un loop senza fine, normalmente da evitare, ma che qui è accettabile per la continua attività richiesta ed anche per non costringere il computer a formulare, dopo ogni mossa, la richiesta "vuoi continuare"?

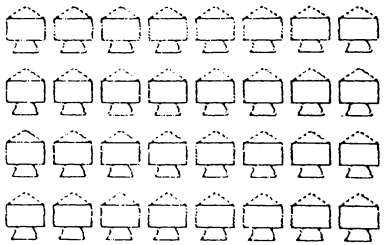
INKEY$ ritorna una stringa che viene convertita in un CODE numerico per permettere l'uso dei codici 8-11 di movimento del cursore, che non sono visualizzabili. Solo per questo motivo si è dovuta scegliere la soluzione descritta, perché altrimenti sarebbe stato meglio rendere K una variabile stringa e confrontarla con il contenuto delle stringhe nelle righe 120-180, ad esempio:

```
IF k$="5" - - -
```

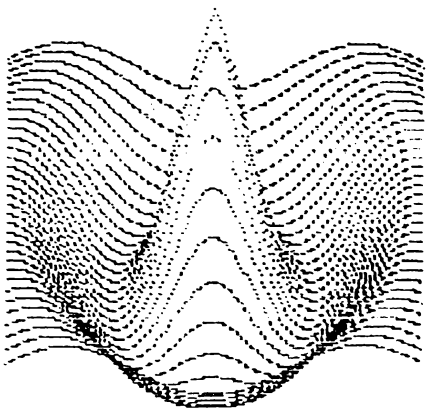
```
  10 LET x=127: LET y=88
 100 FOR c=0 TO 1: PLOT INVERSE
c;x,y
 110 LET k=CODE INKEY$
 120 IF k>=53 AND k<=56 THEN PLO
T x,y
 130 IF k>=8 AND k<=11 THEN PLOT
 INVERSE 1;x,y
 140 PRINT AT 20,0;"x=";x;"  "
y=";y;"  ";
 150 IF (k=8 OR k=53) AND x>0 TH
EN LET x=x-1
 160 IF (k=9 OR k=56) AND x<255
THEN LET x=x+1
```

```
 170 IF (k=10 OR k=54) AND y>0 T
HEN LET y=y-1
 180 IF (k=11 OR k=55) AND y<175
 THEN LET y=y+1
 190 NEXT c
 200 GO TO 100
```

10	Predispone i valori di partenza delle coordinate x - y.
100-190	Loop principale che fa lampeggiare il cursore.
110	Esplorazione tastiera.
120	Se risulta premuto uno dei tasti 5,6,7,8 disegna in nero il punto corrispondente alla posizione del cursore.
130	Se risulta premuto uno dei tasti 5,6,7,8, assieme a SHIFT, disegna in bianco il punto corrispondente alla posizione del cursore.
140	Scrive i valori aggiornati delle coordinate. Gli spazi seguono i numeri per assicurare ogni volta la loro cancellazione.
150-180	Calcolo delle nuove coordinate.
200	Ritorno per un altro flash del cursore.



SOMBRERO



Esiste sicuramente un certo fascino nell'uso del computer per disegnare grafici tridimensionali, ottenuti da funzioni matematiche. Un esempio è il "Sombrero" che rappresenta la funzione:

$$z = \cos r * \exp(-r/3)$$

in cui "r" rappresenta la distanza dal centro del piano X-Y al punto da plottare ricavato da $x^2 + y^2$

Il programma non è lungo da battere, però per completare il disegno, il computer impiega circa 25 minuti!

```
100 FOR x=40 TO 215
110 LET b=999: LET t=0
120 FOR y=16 TO 144 STEP 4
130 LET r=SQR ((x-127)*(x-127)+
(y-80)*(y-80))/15
140 LET z=INT (y+90*EXP (-r/3)*
COS r)
150 IF z<b OR z>t THEN PLOT x,z
160 IF z<b THEN LET b=z
170 IF z>t THEN LET t=z
180 NEXT y
190 NEXT x
```

100-190	Loop di x attraverso tutti i punti da 40 a 215.
110	"b" e "t" eliminano le linee "nascoste" portandole fuori dal tracciato.
120-180	Loop di y attraverso i multipli di 4 da 16 a 144.
130-140	Calcolo dell'altezza alla quale disegnare il punto.
150	Disegna il punto se non è "nascosto".
160-170	Aggiorna "b" e "t" e plotta il punto più alto e quello più basso.

CAPSULE SPAZIALI

Lo statement DRAW x,y offre la possibilità di tracciare una linea retta tra due punti qualsiasi dello schermo. La linea parte dal punto plottato in precedenza a quello calcolato dai due valori che seguono DRAW. I due numeri non sono però le coordinate x e y del punto d'arrivo della linea, bensì le distanze orizzontale e verticale del nuovo punto nei confronti del primo. Ad esempio DRAW 10,10 traccia dal punto di partenza una linea retta con inclinazione di 45 gradi terminante nel punto spostato di 10 sulla destra e di 10 in alto.

Tutto questo può sembrare, all'inizio, un po' strano, ma sicuramente avvantaggia la routine che, usando esclusivamente statement di DRAW, disegna una certa sagoma duplicandola.

Il programma che segue capita giusto a puntino e riproduce sullo schermo il medesimo disegno di ben 32 capsule spaziali. Provate voi a far meglio!

```
  100 FOR v=15 TO 135 STEP 40
  110 FOR h=25 TO 235 STEP 30
  120 PLOT INVERSE 1; OVER 1;h,v
  130 RESTORE
  140 READ m,x,y
  150 IF m=0 THEN DRAW INVERSE 1;
  OVER 1;x,y
  160 IF m=1 THEN DRAW x,y
  170 IF m<>9 THEN GO TO 140
  180 NEXT h
  190 NEXT v
 1000 DATA 0,-12,7,1,24,0,1,0,-14
 ,1,-24,0,1,0,14
 1010 DATA 1,12,8,1,12,-8
 1020 DATA 0,-7,-14,1,3,-8,1,-16,
 0,1,3,8,9,0,0
```

100-110 Con le linee 180 e 190 stabiliscono le coordinate del centro di ognuno dei 32 disegni.

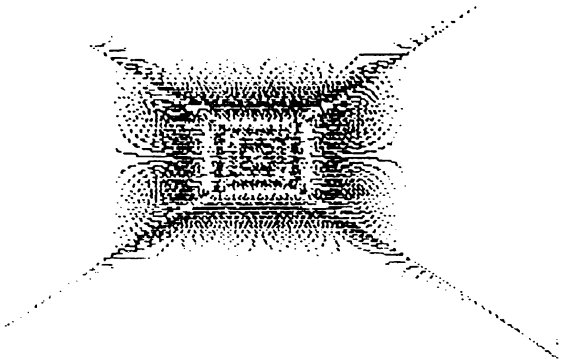
120 Sposta la posizione da cui iniziare a disegnare dal centro di un disegno all'altro, senza lasciare tracce sullo schermo.

140 Riposiziona il "data pointer" alla prima voce dati del programma, in questo caso alla linea 1000.

140-170 Disegna la capsula.

Gli elementi DATA (dalla linea 1000 in poi) sono organizzati in gruppi di tre numeri dei quali il secondo e il terzo vengono usati dagli statement DRAW alle linee 150 e 160. Il primo numero di ogni gruppo, decide invece come agire: se è 0, la linea 150 sposta la posizione di plot senza lasciare traccia, se invece è 1, la linea 160 disegna la retta.

QUADRI D'AUTORE



Essendo disponibile solamente un numero limitato di pixel, le linee rette tracciate in diagonale, appaiono leggermente frastagliate. Tale imperfezione non comporta alcuno svantaggio, anzi, in molti casi, abbellisce i disegni, come vediamo dal programma che segue, il quale presenta un centinaio di bei quadri in sequenza casuale.

Il risultato viene in parte esaltato dai piccoli difetti propri del ricevitore TV. Per prima cosa i TV tendono a confondere i piccoli dettagli presentando, ad esempio, come grigie le linee bianche e nere alternate in rapida successione. In secondo luogo, pur disegnando il programma esclusivamente in bianco e nero, la figura apparirà contornata da chiazze di colori baluginanti. In virtù di questi effetti, l'esempio che risulta qui stampato è molto meno attraente di quello che si può ammirare dal vivo.

```
  10 RANDOMIZE
 100 FOR p=1 TO 100
 110 LET s=2+INT (3*RND)
 120 LET c=INT (2*RND)
 130 FOR x=255 TO 0 STEP -s
 140 PLOT x,0: DRAW OVER c;255-2
*x,175
 150 NEXT x
 160 FOR y=0 TO 175 STEP s
 170 PLOT 0,y: DRAW OVER c;255,1
75-2*y
 180 NEXT y
 190 NEXT p
```

100-190 Loop per contare 100 quadri.

110 Sceglie s, come passo per le linee 130 e 180, per mezzo di un numero random tra 2 e 4.

120 Sceglie c tra i valori random 0 e 1. "c" viene usato nelle linee 140 e 170 per far loro tracciare due linee nere (c=0) oppure linee inverse da quelle già disegnate sullo schermo (c=1).

130-150 Disegna metà quadro.

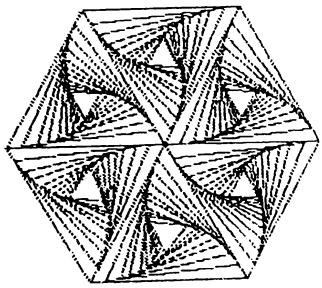
160-180 Disegna l'altra metà.

ESAGONI

Usando la capacità del computer di ripetere più volte una formá base in dimensioni sempre più ridotte, potete realizzare disegni veramente piacevoli e originali. Il programma sotto riportato disegna un gruppo di triangoli, uno successivo all'altro, con dimensioni sempre minori. Il ciclo viene ripetuto sei volte fino a formare un esagono. Potete variare i grafici tralasciando la linea 150 oppure modificando la 140 come segue:

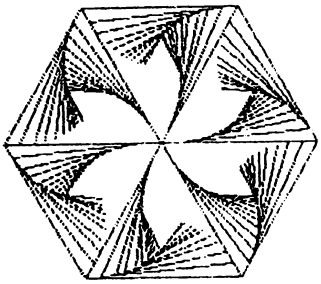
```
140 DRAW INVERSE 1; OVER 1; - - -
```

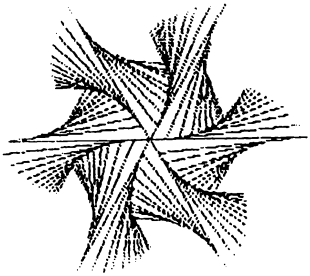
che sposta la posizione di plot alla partenza della linea successiva, senza tracciarla. (OVER e INVERSE saranno trattati in dettaglio nel prossimo capitolo).



```
 100 FOR a=0 TO 1.7*PI STEP PI/3
 110 LET x=127: LET y=88: LET l=
100
 120 FOR d=a TO a+1 STEP .1
 130 PLOT x,y: DRAW l*COS d,l*SI
N d
 140 DRAW l*SIN (PI/6-d)-l*COS d
,l*COS (PI/6-d)-l*SIN d
 150 DRAW -l*SIN (PI/6-d),-1*COS
 (PI/6-d)
 160 LET x=x+.1*l*COS d: LET y=y
+.1*l*SIN d: LET l=.85*l
 170 NEXT d
 180 NEXT a
```

100-180 Loop per disegnare sei gruppi di triangoli.
110 Predisposizione delle condizioni di partenza di ogni gruppo.
120-170 Loop per disegnare i 10 triangoli componenti ogni gruppo.
130-150 Disegna un triangolo.
160 Predispone il punto di partenza e le dimensioni di ogni triangolo.





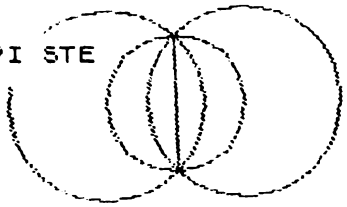
RECINTO

Aggiungendo un terzo valore al comando DRAW

```
DRAW x,y,a
```

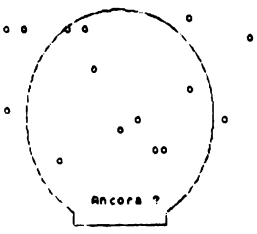
è possibile fargli disegnare una parte di cerchio. Ve lo dimostra la routine che segue.

```
 100 FOR a=-1.5*PI TO 1.5*PI STE
P PI/2
 110 PLOT 127,58
 120 DRAW 0,58,a
 130 NEXT a
```



Generalmente parte di questa routine viene utilizzata per comporre forme complesse, ma, come dimostrazione, eccovi un gioco semplicissimo: il suo scopo è quello di catturare il maggior numero di "uova spaziali" rinchiudendole in un recinto a forma di bolla, di cui potete controllare la dimensione. Le uova toccate dalla linea di recinzione si rompono e non sono valide.

```
 100 FOR a=1 TO 15: PRINT AT 15*
RND,30*RND;"o": NEXT a
 110 PLOT 80,10: DRAW 0,-10: DRA
W 85,0: DRAW 0,10
 200 INPUT "Inserire la grandezz
a della bolla (1-9) ";s
 210 IF s<1 OR s>9 THEN GO TO 20
0
 220 PLOT 80,10: DRAW OVER 1;85,
0,-(4.35+s/10)
 300 PRINT AT 19,12;"Ancora ?"
 310 IF INKEY$="y" THEN RUN
 320 IF INKEY$<>"n" THEN GO TO 3
10
```



100-110 Disegna le 15 uova spaziali in posizioni casuali e la base di lancio.

200-220 Accetta la dimensione del recinto, ne testa la validità e lo disegna. La funzione OVER 1 rivela quando il contorno tocca un uovo.

300-320 Attesa per l'inizio di un altro tentativo.

CARATTERI PERSONALIZZATI

Se non siete soddisfatti dei caratteri e dei simboli grafici in dotazione allo Spectrum, potete aggiungerne altri di vostro gradimento. Questa prerogativa è oltremodo utile, non solo per ottenere caratteri insoliti (ad esempio lettere giapponesi), ma anche per introdurre nuove forme. Nel programma "Space Invaders", per esempio, è necessario disegnare caratteri speciali che rendano l'idea di un alieno: caratteri da presentare sullo schermo per mezzo di PRINT, alla stregua di qualsiasi altra lettera. Se sono necessarie forme più larghe di un carattere, potete disegnare due o più grafici speciali che poi uniti daranno l'effetto desiderato.

Come già spiegato, ogni carattere è formato da un array di 8×8 pixel ognuno dei quali può assumere il colore nero o bianco. Più in particolare, ogni pixel è rappresentato da un singolo bit (binary digit) che ha valore 0 se il pixel è bianco e valore 1 se è nero (vedremo i colori nel prossimo capitolo).

Il carattere completo è messo a disposizione da 8 byte di 8 bit ognuno. La lettera "A" viene scritta:

```
Byte 1 ; 00000000
     2 ; 00111100
     3 ; 01000010
     4 ; 01000010
     5 ; 01111110
     6 ; 01000010
     7 ; 01000010
     8 ; 00000000
```

e voi potete inserire allo stesso modo nell'array un simbolo di vostra invenzione.

Ogni carattere possiede un numero di codice che lo Spectrum può prelevare ed inserire nei suoi calcoli. L'Appendice A del manuale BASIC elenca tutti i 256 codici a disposizione. Tenete presente che alcuni di questi codici (0-5 e 24-31) non vengono usati, altri (6-23) sono codici di controllo, altri ancora (165-255) rappresentano le parole-funzione.

Quelli che a noi interessano sono compresi entro la gamma 144-164 e sono conosciuti come "Grafici Utente A-U".

Raggiungiamo il primo di questi tramite il comando

```
PRINT CHR$ 144
PRINT "shift/graphics A  graphics"
```

che vi presenterà la lettera maiuscola "A".

Tornando al fatto che i caratteri sono rappresentati da una serie di 8 byte, è possibile l'inserzione di caratteri anomali. Lo Spectrum preserva, a tale scopo, 21×8=168 bytes situati nella parte superiore della RAM che automaticamente vengono raggiunti per soddisfare tale esigenza. La funzione che permette un simile risultato è la USR. USR "n" rende l'indirizzo del primo byte dell'area RAM in corrispondenza del quale viene sistemato "n" il quale può essere inserito come lettera maiuscola, minuscola o come carattere grafico.

Per aiutarvi ad inserire il nuovo carattere, il computer mette a disposizione la funzione BIN la quale deve essere seguita da un numero binario formato da "1"e "0" e dà l'equivalente valore decimale. Se volete fare un esperimento togliete l'alimentazione per qualche secondo al vostro Spectrum, dopodichè battete e fate girare quanto segue:

```
10 PRINT CHR$ 144
20 POKE USR "a",BIN 10000001
30 PRINT CHR$ 144
```

La seconda lettera "A" presentata avrà le "orecchie" per effetto della linea 20 la quale aggiunge, infatti, due punti neri agli estremi della linea superiore dei pixel che compongono la lettera "A", il cui codice è appunto 144. A questo punto non vi rimane altro da fare che POKare un altro valore oppure spegnere il computer.

Visto che il binario 10000001 corrisponde al decimale 129, come dimostrato dal comando diretto

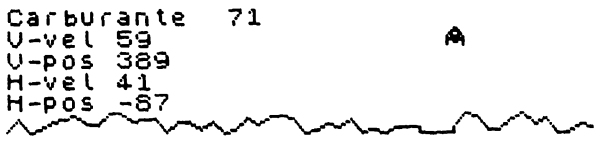
```
PRINT BIN 10000001
```

la linea 20 può essere scritta nel modo seguente:

```
20 POKE USR "a",129
```

Naturalmente per definire completamente un nuovo carattere, è necessario POKare tutte e otto le locazioni di memoria in funzione del carattere stesso. La dimostrazione nel prossimo programma.

ALLUNAGGIO



Nessun libro di programmi per computer è completo se non riporta l'allunaggio, che offre l'opportunità di mostrare le capacità grafiche della macchina.

Lo scopo del gioco è quello di far scendere il vostro veicolo lunare su un tratto di superficie piana.

Quando inizia il gioco, voi siete a 2100 metri di altezza, vi state muovendo orizzontalmente a 30 metri al secondo ed iniziate la discesa. Il vostro computer di bordo è di nuovo guasto, per cui siete costretti ad atterrare manualmente con i motori propulsori funzionanti solo in modo binario: on e off.

I vostri controlli sono i tasti 5,7,8 che azionano i propulsori rispettivamente a sinistra, sopra e a destra, ma che vanno azionati uno alla volta! Dovrete toccare la superficie lunare con velocità orizzontale e verticale, nonché con posizione orizzontale il più vicino possibile allo zero. Buona fortuna e occhio al carburante.

```
  5 LIST
 10 RANDOMIZE
 20 LET f=100: LET vv=0: LET vp
=2100: LET hv=30: LET hp=2300
100 POKE USR "a",BIN 00011000
110 POKE USR "a"+1,BIN 00111100
120 POKE USR "a"+2,BIN 01100110
130 POKE USR "a"+3,BIN 10111101
140 POKE USR "a"+4,BIN 11111111
```

```
  150 POKE USR "a"+5,BIN 10011001
  160 POKE USR "a"+6,BIN 10000001
  170 POKE USR "a"+7,0
  200 LET y=0: PLOT 0,0
  210 FOR a=0 TO 250 STEP 5
  220 IF a=180 THEN DRAW 0,-y: DR
AW 15,0: DRAW 0,y: LET a=195
  230 LET dy=5-INT (11*RND): IF y
+dy<0 OR y+dy>7 THEN GO TO 230
  240 DRAW 5,dy: LET y=y+dy
  250 NEXT a
  300 PRINT AT 16,0;"Carburante
";f;"#"
  310 PRINT "V-vel ";vv;"##";"V-p
os ";vp;"##"
  320 PRINT "H-vel ";hv;"##";"H-p
os ";hp;"##"
  330 PRINT OVER 1;AT 21-vp/100,2
3-hp/100;CHR$ 144;CHR$ 8;
  340 IF vp=0 OR (vp<50 AND ABS (
hp-2300)>49) THEN GO TO 500
  350 IF hp=-800 THEN GO TO 600
  360 FOR a=1 TO 50: NEXT a
  400 LET k$=INKEY$: IF f=0 THEN
LET k$=""
  410 IF k$="5" OR k$="7" OR k$="
8" THEN LET f=f-1
  420 LET vp=vp-vv: IF vp<0 THEN
LET vp=0
  430 LET vv=vv+1: IF k$="7" THEN
 LET vv=vv-2
  440 LET hp=hp-hv: IF hp<-800 TH
EN LET hp=-800
  450 IF k$="8" THEN LET hv=hv+1
  460 IF k$="5" THEN LET hv=hv-1
  470 PRINT "#": GO TO 300
  500 PRINT AT 0,0;"Sei giu', "'
  510 IF vv>10 OR ABS hv>10 THEN
PRINT "Ma sei morto !": STOP
  520 IF vv>5 OR ABS hv>5 THEN PR
INT "-perfetto-"
  530 IF ABS hp>49 THEN PRINT "Ma
 nel posto sbagliato."
  540 IF vv<5 AND ABS hv<5 AND AB
S hp<50 THEN PRINT "ben arrivato
   sig.Rossi"
  550 STOP
  600 PRINT AT 0,0;"Sei fuori dal
 mondo !"
```

10-20 Predisposizione dei valori iniziali.

100-170 Trasforma la lettera "A" nel modulo lunare.

200-250 Disegna la superficie lunare in modo random all'infuori del tratto pianeggiante disegnato dalla linea 220.

300-470 Loop principale del programma.

300-320 Aggiornamento della velocità, della quantità del carburante ecc.

330 Disegna il modulo lunare nella posizione appropriata. L'opzione OVER 1 permette di non far sparire la superficie lunare quando voi vi atterrate sopra. Notate CHR$ 8, muove all'indietro la posizione di scrittura, dopo aver disegnato il modulo.

340 Lascia il loop principale se avete atterrato.

350 Lascia il loop principale se il modulo è uscito dallo schermo.

360 Tempo generato dal loop per rallentare la discesa entro limiti ragionevoli. Non può essere usato PAUSE.

400 Cerca il tasto premuto. Lo ignora se è finito il carburante.

410-460 Calcola la nuova velocità e le nuove posizioni, tenendo conto degli azionamenti dei tasti 5,7 o 8.

470 Cancella l'ultima immagine del modulo sovrapponendo uno spazio, quindi torna alla partenza del loop principale.

500-550 Stampa i commenti rituali dopo aver atterrato.

600 Fine del programma se siete usciti dall'area dello schermo.

Nota: rispettate gli spazi tra gli apici alle linee 300,310,320,470.

CAMBIAMENTO COMPLETO

Volete sapere come fare per ridefinire tutti i caratteri alfanumerici dello Spectrum (codici da 32 a 127)?

Come già visto nel quinto capitolo, la Variabile di Sistema CHARS (23606,23607) contiene un numero 256 volte inferiore all'indirizzo ROM di partenza del set dei punti riguardanti i caratteri standard. Tale Variabile viene posta a 15360 all'accensione dello Spectrum oppure al comando di NEW.

Cambiando il valore sopra citato, lo Spectrum andrà a pescare la serie di punti nella locazione di memoria specificata. Accertatevene battendo il comando diretto.

```
POKE 23606,8
```

Vedrete che battendo "A", viene scritto "B", battendo "B" viene scritto "C" e così via. Non avete fatto altro che spostare il valore di CHARS di 8 byte equivalenti appunto ad un carattere.

È possibile usare questa proprietà facendo puntare CHARS in un'area RAM ove POKare i propri caratteri.

Dando un'occhiata alla mappa di memoria dello Spectrum, troviamo che i primi 168 byte sono riservati appunto alle creazioni grafiche da parte dell'utente, ma per poterli utilizzare, è necessario abbassare il limite superiore della RAM (RAMTOP) per mezzo di uno statement CLEAR.

Vediamo il programma.

```
 100 CLEAR 32767-768-168-1
 110 LET u=32767-768-168
 120 FOR a=0 TO 767: POKE (u+a),
PEEK (15616+a): NEXT a
 130 LET v=u-256: POKE 23606,v-2
56*INT (v/256): POKE 23607,INT (
v/256)
 140 LET x=v+8*CODE "R"
 150 POKE x,0: POKE x+1,62
 160 POKE x+2,66: POKE x+3,66
 170 POKE x+4,62: POKE x+5,34
 180 POKE x+6,66: POKE x+7,0
 190 PRINT "USSR"
 200 POKE 23606,0: POKE 23607,60
 210 PRINT "USSR"
```

100	Prenota uno spazio in RAM di 768 byte per i caratteri.
110-120	Copia la vecchia serie della ROM. (Impiega circa 15 sec)
130	Aggiorna CHARS.
140-180	Cambia la forma per "R".
190	Dimostrazione.
200-210	Ridispone la ROM come in partenza.

CAPITOLO 10
SUONI E COLORI

Un uso intelligente delle capacità sonore e cromatiche dello Spectrum, affina sicuramente la qualità dei giochi e può aiutare enormemente l'operatore, attirandone l'attenzione su informazioni particolarmente importanti in programmi applicativi.

L'impiego del comando BEEP non comporta alcuna difficoltà. Lo si batte da tastiera (tasto Z) facendolo seguire da due numeri, il primo dei quali stabilisce la lunghezza della nota e il secondo il tono. Esercitatevi facendo girare le seguenti routine.

```
FOR a=0 TO 30 : BEEP .01,a : NEXT a
```

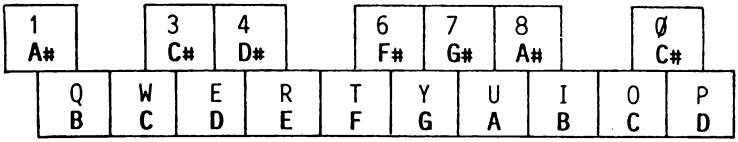
```
FOR a=0 TO 5: FOR b=12 TO 4 STEP -8: BEEP .5,b:
NEXT b: NEXT a
```

```
FOR a=1 TO 40 : BEEP 1/a,.5 : NEXT a
```

L'unico piccolo inconveniente sta nel fatto che l'uso di BEEP non è possibile in contemporanea con altre funzioni, infatti quando lo Spectrum lo esegue, si disabilita verso il resto del programma. Tale fatto lo potete notare dallo svolgimento di una gara che preveda immagini in movimento: durante il periodo in cui viene emesso il suono le figure si bloccano, per poi riprendere la loro corsa al termine del messaggio musicale.

UNA TASTIERA SONORA

Questo programma vi permette di suonare musica usando le prime due righe della tastiera dello Spectrum, come se fossero i tasti bianchi e neri di un pianoforte.



Naturalmente i tasti andranno premuti uno alla volta. Azionandoli assieme al tasto SHIFT, le note della scala musicala verranno alzate di una ottava. La durata di ogni nota viene fissata a 0.2 secondi dalla linea 200, ma tale valore può essere abbassato fino a 0.02 per ottenere effetti particolari; provate. Nello scrivere un programma del genere, l'unico problema serio sta nel trasformare il codice ottenuto dalla pressione del tasto in un valore appropriato da porre nello statement di BEEP. Il programma deve, inoltre, risultare insensibile ad azionamenti accidentali. La soluzione si ottiene predisponendo un array con elementi sufficienti a rappresentare tutti i codici, compreso lo 0, che vengono via via generati dallo statement INKEY$. A ciascun elemento dell'array viene affidato un particolare valore di frequenza oppure, nel caso questo sia un codice anomalo, un numero speciale che il programma possa facilmente riconoscere.

```
  100 DIM p(123): FOR a=1 TO 123:
  LET p(a)=-99: NEXT a
  110 FOR n=-2 TO 14: READ k: LET
  p(k+1)=n: NEXT n
  120 FOR n=10 TO 26: READ k: LET
  p(k+1)=n: NEXT n
  200 LET n=p(1+CODE INKEY$): IF
n>-60 THEN BEEP .2,n
  210 GO TO 200
  900 DATA 49,113,119,51,101,52,1
14,116,54,121,55,117,56,105,111,
48,112
  910 DATA 7,81,87,4,69,5,82,84,1
0,89,11,85,9,73,79,12,80
```

100 Predispone l'array con tutti gli elementi.

110 Inserisce nell'array i valori di frequenza corrispondenti all'ottava inferiore. I valori sono interi da 2 a 14.

120 Inserisce nell'array i valori di frequenza corrispondenti all'ottava superiore (10-26).

200 Suona la nota se il tasto premuto è considerato valido.

210 Ritorno per il tono successivo.

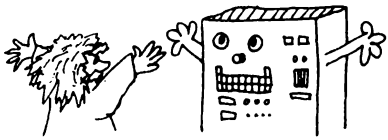
910 Codici validi per l'ottava superiore.

SEQUENZA

In questa divertente sfida, lo Spectrum genera una sequenza random di note che voi dovrete ripetere senza commettere errori. Potete scegliere tra cinque livelli di difficoltà, ricordandovi che selezionando il più alto, il computer emette un maggior numero di note. Ogni partita prevede dieci tentativi, ognuno dei quali ha una sequenza di note più lunga di quella precedente. Il punteggio viene presentato solo a gara terminata.

```
  10 RANDOMIZE : DIM p(10): DIM
i$(1)
 100 PRINT AT 7,6;"Abilita' (1-5
)?"
 110 INPUT s: IF s<1 OR s>5 THEN
 GO TO 110
 120 PRINT AT 6,19;s: LET s=s+3
 130 PRINT AT 10,0;"Questo progr
amma usa ";s;" note."
 140 PAUSE 20: PRINT '"Esse sono
 numerate;"
 150 FOR n=1 TO s: PAUSE 20: PRI
NT AT 12,20;n: BEEP 1,n: NEXT n
 200 LET c=0
 210 FOR g=1 TO 10: PAUSE 100: C
LS : PRINT AT 5,12;"Vai ";g
 220 FOR n=1 TO g
 230 PAUSE 20: LET p(n)=1+INT (s
*RND): BEEP 2/s,p(n)
 240 NEXT n
 300 PRINT AT 10,8;"Ripeti quest
a !"
 310 FOR n=1 TO g
 320 LET k=CODE INKEY$-CODE "0":
 IF k<1 OR k>8 THEN GO TO 320
```

```
 330 BEEP .5,k: IF k<>p(n) THEN
GO TO 360
 340 NEXT n: PRINT AT 13,10;"Ben
 fatto"
 350 GO TO 400
 360 PRINT AT 13,12; FLASH 1;"ER
RATO": BEEP 1,-30
 370 PAUSE 50: PRINT AT 15,8;"Er
a la nota ";p(n): BEEP 1,p(n)
 400 IF n>1 THEN LET c=c+n+g
 410 FOR a=1 TO 200: NEXT a
 420 NEXT g
 500 IF c>99 THEN PRINT AT 8,10;
"Eccellente"
 520 PRINT AT 14,7;"Altra prova
?"
 530 INPUT LINE i$: IF i$="n" TH
EN STOP
 540 IF i$<>"y" THEN GO TO 530
 550 PRINT AT 14,8;"Stessa abili
tà ?"
 560 INPUT LINE i$: IF i$="n" TH
EN RUN
 570 IF i$="y" THEN GO TO 200
 580 GO TO 560
```



10	Inizializzazione.
100-120	Inserzione del livello di difficoltà da parte del giocatore.
130-150	Mostra il numero delle note usate.
200	Partenza; mette il punteggio a zero.
210-420	Loop principale eseguito ad ogni tentativo.
220-240	Suona una sequenza random di note e le inserisce nell'array p( ).
300	Legge la risposta dell'operatore.
310-340	Loop per stabilire l'esatto numero di note.
320	Testa la validità del tasto premuto.
330	Suona la nota scelta dall'operatore. Se la scelta è errata, interrompe il loop saltando alla linea 360.
350	Manda alla linea 400 se tutte le note sono state introdotte nella giusta sequenza.
360-370	Comunica al giocatore che ha sbagliato e la nota che ha battuto.

400-420 Aggiorna il punteggio. Attende un attimo (loop FOR NEXT alla linea 410; PAUSE è inadeguato), quindi torna per il tentativo seguente.

500-510 Scrive il punteggio e commenta.

520-580 Invita l'operatore a giocare nuovamente.

ATTRIBUTI DEL VIDEO

Allo stesso modo in cui è possibile scrivere caratteri o disegnare forme sullo schermo, è anche possibile controllarne il colore e la luminosità; perfino facendone lampeggiare diverse sezioni. Questo insieme di funzioni è conosciuto sotto il nome di "Attributi" video. Lo Spectrum ne ha in tutto quattro; sono:

PAPER. È questo il colore usato per lo sfondo. Prende il colore nero all'accensione del computer e al comando NEW. I colori possibili sono quelli riportati sulla tastiera in corrispondenza dei tasti numerati da 0 a 7.

INK. È il colore usato dal carattere o dal disegno. Prende il colore bianco all'accensione e al comando NEW. Come per PAPER, gli otto colori possibili sono quelli dei tasti 0-7.

BRIGHT. Lo Spectrum può fare assumere allo schermo TV, due livelli di luminosità.

BRIGHT 0 : luminosità normale.

BRIGHT 1 : luminosità accentuata.

FLASH. È possibile evidenziare parte dello schermo facendola lampeggiare con l'inversione continua dei due colori INK e PAPER.

FLASH 0 : condizioni normali.

FLASH 1 : lampeggio.

I quattro attributi operano su tutto il carattere, controllano cioé le caratteristiche di ognuna delle 24x32 posizioni possibili dei caratteri sullo schermo, e non i singoli pixel che compongono il carattere; pertanto l'intero array di 8x8 pixel che forma il carattere possiede i medesimi attri-

buti di INK o PAPER, che possono invece essere diversi nella cella adiacente.

I 768 set di attributi per le 32x24 celle di carattere sono collocati nell'area RAM loro riservata che li distingue per mezzo di 768 byte, uno per ogni cella. I bit 0-2 di ciascun byte controllano il colore INK, i bit 3-5 il colore di PAPER, il 6 controlla il BRIGHT e il 7 il FLASH. Il valore dei byte interessati viene richiamato dalla funzione ATTR.

Attributi Permanenti

Le parole funzione PAPER, INK, BRIGHT e FLASH, possono trovare posto in statement differenti:

```
10 PAPER 7
20 BRIGHT 0
```

Sotto questa forma agiscono nel momento in cui lo Spectrum esegue gli statement e così restano fino a quando non sopraggiungano statement successivi che li modifichino. Provate ad inserire:

```
PAPER 3 : INK 4 : PRINT "ABCD"
```

e vedrete comparire la scritta "ABCD" in verde su di uno sfondo rosso. Se ora ordinate allo Spectrum di scrivere ancora, ad esempio con

```
PRINT "CIAO"
```

la scritta e lo sfondo rimarranno dello stesso colore dei precedenti. Cambiando invece i colori di PAPER e INK non cambia quello delle scritte stampate in precedenza e delle due linee poste nella parte bassa dello schermo dedicate all'input e ai commenti.
Provate inserendo

```
PAPER 0 : INK 7 : PRINT "COLORI"
```

BRIGHT e FLASH vengono usati alla stessa maniera;

```
BRIGHT 1 : PRINT "BRIGHT"
FLASH 1 : PRINT "FLASH"
```

Gli attributi selezionati in tale maniera, vale a dire in statement che abbiano INK, PAPER, BRIGHT o FLASH come prima parola, vengono chiamati attributi "Permanenti". Per variare gli attributi di tutte e 22 le linee contemporaneamente devono essere impiegati in combinazione con CLS. Battete:

```
BRIGHT 1 : CLS          oppure
PAPER 3 : CLS           oppure
FLASH 3 : CLS
```

La routine che segue dimostra come, anche usando comandi di HRG, gli attributi non possano agire all'interno della cella. Vengono disegnate due linee rette che si intersecano tra di loro di colore diverso di cui la seconda modifica, nel punto d'incrocio, il colore della prima.

```
10 INK 2 : PLOT 0,20 : DRAW 255,30
20 INK 5 : PLOT 0,50 : DRAW 255,-30
```

Border

Ed eccoci giunti alla parola-funzione BORDER (tasto B). Lo statement

```
BORDER n
```

dove "n", che rappresenta uno dei colori associati ai numeri 0-7, stabilisce il colore del bordo, cioè della parte dello schermo esterna all'area display governata da PAPER. Influisce anche sulle due linee riservate agli input e ai commenti, ma solamente quando lo Spectrum scrive un messaggio o esegue uno statement di INPUT. Accertatevene battendo:

```
10 BORDER 0
20 PAUSE 50
30 INPUT "Premi ENTER";a$
40 FOR a=1 TO 200: NEXT a
50 BORDER 7
```

e riprovate eliminando la linea 30.

Attributi Temporanei

Potete anche includere le parole-funzione PAPER, INK, BRIGHT e FLASH entro statement di PRINT, PLOT, DRAW o CIRCLE.

```
PRINT INK 3;"Hi"
```

In questi casi l'effetto è limitato a quel particolare statement e quanto scritto o disegnato su statement seguenti ricade sotto gli attributi permanenti. Inserite:

```
10 FLASH 0
20 PRINT FLASH 1;"FLASH"
30 PRINT "NO FLASH"
```

CAMBRIDGE TARTAN

È un programma assai semplice che mostra un attraente display idoneo, tra l'altro, alla messa a punto dei controlli del vostro TV. Distinguerete sicuramente la differenza tra i due livelli di luminosità di ogni colore.

```
10 FOR p=1 TO 7
20 FOR b=0 TO 1
30 PRINT BRIGHT b; PAPER p;"#"
40 NEXT b
50 NEXT p
60 GO TO 10
```

Colori 8 e 9

A INK e PAPER possono essere attribuiti i colori 8 e 9. In tal caso lo Spectrum prende determinate decisioni sui colori da usare per la scrittura o per il disegno. Più precisamente:

'8' mette a disposizione un colore 'trasparente' comunicando allo Specatrum di non apportare alcuna variazione ai colori scelti in precedenza;
'9' stabilisce il contrasto. Il computer sceglie automaticamente il colore adeguato per meglio contrastare con le scritte o con lo sfondo già esistenti. Quando sia a INK che a PAPER viene attribuito il valore 9, si ottengono scritte in nero su sfondo bianco.

Anche BRIGHT e FLASH possono accettare il valore 8 lasciando inalterati i due attributi relativi. Provate:

```
10 PRINT BRIGHT 1;"#"
20 PRINT FLASH 1;"#"
30 PRINT AT 0,0; BRIGHT 8;"AB"
40 PRINT FLASH 8;"CD"
```

Ricordatevi di inserire lo spazio tra gli apici alle linee 10 e 20.

CALEIDOSCOPIO

Il programma che segue gira per circa dieci minuti, mostrando una sequenza di piacevoli combinazioni sempre più complesse.

```
100 FOR c=50 TO 1 STEP -1
110 FOR a=0 TO 11
120 FOR b=a TO 19
130 LET d=INT (a*b/c+(a+b)/c+c)
: PAPER d-9*INT (d/9)
140 PRINT AT a,b;" ";AT a,31-b;
" ";AT 21-a,b;" ";AT 21-a,31-b;"
"
150 PRINT AT b,a;" ";AT b,31-a;
" ";AT 21-b,a;" ";AT 21-b,31-a;"
"
160 NEXT b
170 NEXT a
180 NEXT c
```

Anche qui rammentatevi gli spazi alle linee 140 e 150.

Disegni

Una istruzione PRINT usa sempre gli ultimi valori in uso per gli attributi del carattere che visualizza, siano essi attributi permanenti stabiliti dalle istruzioni INK, PAPER, BRIGHT o FLASH, oppure attributi temporanei stabiliti all'interno della stessa istruzione PRINT.

Le istruzioni in alta risoluzione PLOT, DRAW e CIRCLE, però, si comportano in modo differente, perché modificano solo il colore dei pixel delle celle su cui scrivono, mentre gli altri attributi vengono alterati solo dall'uso delle istruzioni PAPER, BRIGHT o FLASH all'interno delle istruzioni PLOT, DRAW o CIRCLE. Fate girare, ad esempio, la routine:

```
10 PAPER 7: INK 0: CLS
20 PAPER 6: CIRCLE 100,100,50
```

la quale traccia un cerchio nero in campo bianco. Lo statement PAPER 6 presente alla linea 20 non ha alcun effetto. Ma se cambiamo la riga 20:

```
20 CIRCLE PAPER 6;100,100,50
```

le celle di carattere interessate dal cerchio assumeranno il colore giallo.

Tutto ciò torna utile nel caso in cui voleste modificare velocemente il colore di una riga o di una colonna disegnandovi sopra una linea per mezzo di un DRAW comprendente un PAPER.

QUADRATI VIVENTI

Una dimostrazione pratica è offerta dal prossimo programma che disegna una serie di quadratini concentrici di colore variabile. Vengono mantenuti gli stessi colori di INK e PAPER, ma, mentre INK è definito una volta per tutte (linea 110), i colori di PAPER vengono inclusi in ogni singolo statement di DRAW (linee 120 e 130).

```
  50 BORDER 0: PAPER 0: CLS : LE
T c=1
 100 FOR a=-8 TO 8 STEP 16
 110 FOR b=41-5*a TO 41+5*a STEP
 a: INK c
 120 PLOT 128-b,87-b: DRAW PAPER
 c,b+b,0: DRAW PAPER c,0,b+b
 130 DRAW PAPER c,-b-b,0: DRAW P
APER c,0,-b-b
 140 LET c=c+1: IF c>7 THEN LET
c=1
 150 NEXT b
 160 NEXT a
 170 GO TO 100
```

50 Clear dello schermo. Predispone il valore iniziale e di "c".

100-160 Loop per alternare il disegno dei quadretti dall'interno all'esterno e viceversa.

110-150 Loop per disegnare la sequenza dei quadrati concentrici.

120-130 Disegna un quadrato col colore "c".

140 Cambia colore.

170 Ripetizione della sequenza.

OVER E INVERSE

Le funzioni OVER e INVERSE sono simili agli attributi INK, PAPER, BRIGHT e FLASH, in quanto agiscono sulla scrittura o sul disegno. Possono essere usati come prima parola dello statement, nel caso debbano stabilire valori permanenti, oppure possono venire inseriti in statement di PRINT, DRAW, PLOT o CIRCLE per determinare valori temporanei. Questi valori non trovano però posto nell'area di RAM destinata agli Attributi, bensì sono considerati istruzioni che lo Spectrum interpreta per mezzo degli stessi comandi PRINT, PLOT, DRAW e CIRCLE.

Le loro funzioni sono:

OVER 0: Cancella il contenuto della cella di carattere (in scrittura) o il pixel (in disegno).

OVER 1: Usato assieme a PLOT, DRAW o CIRCLE, scambia il colore di NK nel pixel, con quello di PAPER. Usato con PRINT, tutti i pixel di colore PAPER nel nuovo carattere

restano invariati, ma quelli che sono di colore ink nel nuovo carattere cambiano in:

- INK se erano colore PAPER;
- PAPER se erano colore INK.

INVERSE 0: Scrive o disegna usando il colore di ink; è la condizione usuale di lavoro.

INVERSE 1: Scrive o disegna usando il colore di paper.

Impiegato con PRINT, porta lo sfondo del carattere in colore ink facendolo apparire in inverse video.

Usando OVER 1 e INVERSE 1 insieme a PLOT, DRAW o CIRCLE, non si modifica lo schermo.

Usando OVER 1 e INVERSE 1 assieme al comando PRINT, ne risulta una combinazione dei vecchi caratteri con i nuovi per cui il risultato si presenta in inverse.

Lo statement di PRINT cambia sempre i colori di ink e paper delle celle, siano questi permanenti (dati da singoli statement di PAPER o INK) o temporanei (stabiliti da funzioni INK o PAPER comprese nello statement di PRINT).

Potete sfruttare tale proprietà per cambiare i colori di paper e ink della cella senza influenzare il carattere quivi presente, per mezzo di:

```
PRINT AT r,c; OVER 1; INK i; PAPER p;" "
```

infatti battendo lo spazio con OVER 1 lasciate il carattere esattamente dove era.

Tutto ciò vi permette di cambiare velocemente i colori di paper e di ink su tutta l'area dello schermo senza modificare le scritte fatte in precedenza.

```
DIM z$(22*32): PRINT AT 0,0; OVER 1; INK i;
PAPER p;z$
```

(Il primo statement della linea genera la stringa z$ contenente tanti spazi quanti bastano per riempire lo schermo). Volendo alterare solamente il colore di ink, lasciando tale e quale quello di paper, è necessario attribuire al colore di paper il valore '8' (trasparente).

Analogamente, la combinazione INVERSE 1 OVER 1, usata assieme a DRAW, PLOT e CIRCLE, modifica i colori ink e paper delle celle, senza influire sulla forma di quanto scritto o disegnato. Volendo cambiare il colore di paper, la funzione PAPER deve essere inserita entro uno dei comandi DRAW, PLOT o CIRCLE.

Un esempio di tale tecnica, è dato dalla routine che segue; provate

```
 100 PAPER 7: INK 0: INVERSE 0:
OVER 0: CLS
 110 FOR a=1 TO 22*32: PRINT "*"
;: NEXT a
 120 INK 4: INVERSE 1: OVER 1: P
LOT 0,0: DRAW PAPER 2;255,175
```

la quale riempie dapprima lo schermo di stelle nere su campo bianco, quindi le modifica lungo la linea diagonale, dipingendole in verde su campo rosso.

CARATTERI DI CONTROLLO

Se andate a consultare l'Appendice A del manuale BASIC dello Spectrum, troverete che i caratteri con codice da 6 a 23 corrispondono alle funzioni di scrittura AT, INVERSE ecc.. Sono questi caratteri che voi potete includere in stringhe da far elaborare poi allo Spectrum. È possibile concretizzare tali caratteri, usando la funzione CHR$ X la quale restituisce il singolo carattere dal codice X. Ad esempio le righe:

```
LET= A$="1"+CHR$ 6 +"2"
LET A$="1 2": LET A$(2)=CHR$ 6
```

scrivono una stringa il cui secondo carattere è una virgola.

```
CLS: PRINT A$
```

scriverà "1" all'inizio della prima linea dello schermo e "2" al centro come se fosse stato eseguito

```
CLS: PRINT "1","2"
```

CHR$ 8 è molto utile in quanto sposta la posizione di scrittura indietro di un posto;

```
CLS: PRINT "AB"+CHR$ 8 +"CD"
ACD
```

dà ACD

CHR$ 9-12, sfortunatamente non producono alcun risultato

CHR$ 13 sposta la posizione di scrittura alla linea seguente.

CHR$ 16 (INK) e CHR$ 17 (PAPER) possono essere seguite da un singolo carattere con un codice da 0 a 9 corrispondente ai rispettivi colori. Battete:

```
10 FOR c=0 TO 9
20 PRINT CHR$ 16+CHR$ C+" INK
= ";C
30 PRINT CHR$ 17+CHR$ C+"PAPER
= ";C
40 NEXT C
```

Similmente anche i codici 18-21, corrispondenti a FLASH, BRIGHT, INVERSE e OVER, possono essere seguiti da caratteri singoli con codi-

ce 0, 1 oppure 8 (trasparente). Inserite

```
PRINT "BUIO"+CHR$ 19 + CHR$ 1 + "LUCE"
```

CHR$ 22 (AT) deve essere seguita da due byte rappresentanti i numeri di linea e di colonna;

```
PRINT CHR$ 22 +CHR$ 7 +CHR$ 8 +"CIAO"
```

(è lo stesso di) `PRINT AT 7,8;"CIAO"`

CHR$ 23 (TAB) deve anch'essa essere seguita da due bytes; il primo rappresenta il numero di colonna, mentre il secondo viene ignorato ma è obbligatorio:

```
PRINT CHR$ 23 +CHR$ 18 +CHR$ 99 +"ARRIVEDERCI"
```

(è equivalente a) `PRINT TAB 18;"ARRIVEDERCI"`

Negli esempi fatti, per congiungere le singole voci è stato usato il segno "+", che all'occorrenza può essere sostituito dal punto e virgola. Provate.

```
PRINT CHR$ 23; CHR$ 18; CHR$ 99;"ARRIVEDERCI"
```

Gli Attributi controllati in questo modo sono temporanei e durano solamente fino alla fine dello statement di PRINT che li include.

Tutto ciò può sembrare alquanto complicato, ma ricordatevi che in questo modo potete comporre delle stringhe contenenti dei codici che divengono attivi ogni volta che la stringa viene stampata.

COLORI DIRETTI

È possibile il controllo diretto da tastiera dei colori senza usare i comandi PAPER, INK ecc.

Se, inserendo una linea di programma, azionate assieme CAPS SHIFT e SIMBOL SHIFT, in modo che il cursore vada in E mode, potrete, premendo uno dei tasti da 0 a 7, cambiare il colore di paper in uso in un altro a vostro piacimento.

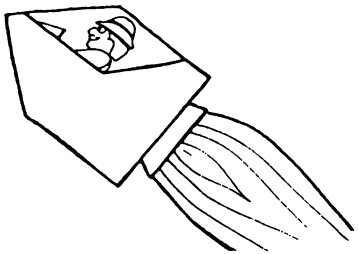
Richiamando il cursore in E mode, tenendo premuto CAPS SHIFT e azionando uno dei tasti colore (0-7), è possibile anche cambiare il colore di ink di tutto il testo seguente. E non è tutto! Inserendo il 9 al posto di 0-7, i caratteri seguenti verranno presentati con una luminosità superiore oppure, azionando insieme CAPS SHIFT e 9, lampeggeranno. Il tasto 8 provoca il processo inverso, ristabilendo la luminosità normale e cancellando il lampeggio.

Potrete ottenere tutti questi effetti anche inserendo i caratteri di controllo nel listato del programma, ed in particolare perché agiscano durante l'esecuzione devono essere presenti come stringhe di caratteri.

I caratteri di controllo non sono visibili listando il programma, ma se evitate una riga che li contiene potete notare che il cursore sembra esitare quando vi passa sopra.

Un'idea interessante: se stabilite un colore di ink uguale al colore di paper nel modo visto, potete nascondere parte del vostro programma da sguardi indiscreti; ricordatevi però che la parte "nascosta" ridiventa visibile se il programma viene listato su stampante.

CAPITOLO 11
MOVIMENTO



I giochi più divertenti sono sicuramente quelli ricchi di azione in cui le sagome si muovono rapidamente attraverso lo schermo. In questo capitolo vediamo come programmare lo Spectrum per conferire movimento agli oggetti.

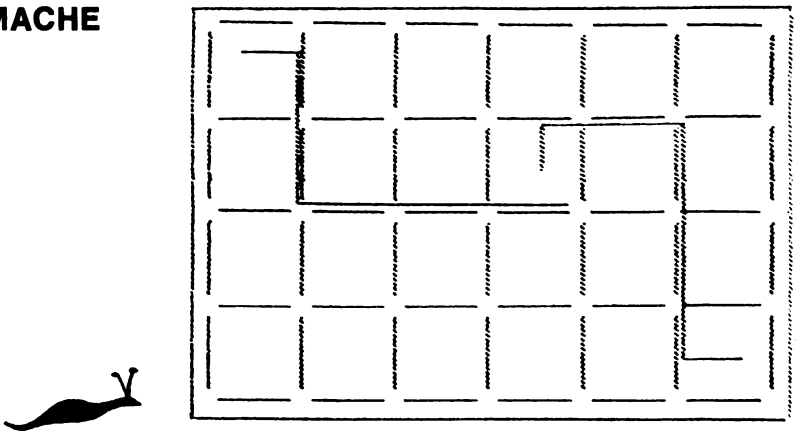
Va detto subito però che, con un programma scritto in BASIC, è impossibile ottenere gli stessi effetti di un game tradizionale, a causa della lentezza del linguaggio. Malgrado ciò i programmi presentati in questo capitolo sono ugualmente divertenti poiché in molti di essi sono state semplificate le forme, pur di mantenere una notevole velocità di movimento. Se volete scrivere giochi complessi e veloci, dovete invece usare il Linguaggio Macchina o il Forth, che girano dieci volte più veloci del BASIC, ma che esulano dal contenuto di questo volume.

La tecnica base per creare l'impressione di movimento è assai semplice: è sufficiente disegnare o scrivere ripetutamente un oggetto in posizioni dello schermo sempre diverse:

```
FOR a=0 TO 255: PLOT a,100: NEXT a
FOR a=0 TO 31: PRINT AT 0,a;"*": NEXT a
```

Facendo girare queste due linee, non otterrete alcun oggetto in movimento, bensì una linea che continua a crescere. Sebbene questo non sia l'effetto che volevamo ottenere, talvolta può risultare utile e infatti è alla base del gioco "Lumache".

LUMACHE



Questa è una gara tra due giocatori, ognuno dei quali tenta di intrappolare l'altro in una viscida traccia.

Le due lumache avanzano progressivamente nei meandri del labirinto, lasciandosi dietro tracce velenose. Se una delle due lumache tocca la scia o uno dei muri del labirinto, muore, e la partita è finita. Lo scopo del gioco è quindi quello di manovrare la propria lumaca in modo da intrappolare quella avversaria, tenendo presente che sia l'una che l'altra si muovono in continuazione, col pericolo di morire all'istante.

La lumaca 1 parte dall'angolo in alto a sinistra dello schermo e i quattro tasti che la controllano sono W, A, S e Z rispettivamente per spostamenti in alto, a sinistra, a destra e in basso.

La lumaca 2 parte dall'angolo in basso a destra ed è controllata dai tasti O, K, L e SYMBOL SHIFT.

Il programma usa le variabili "a" e "b" per tenere aggiornata la posizione della lumaca 1 sullo schermo, e le variabili "c" e "d" per ricordarsi la direzione di movimento. "c" vale 1 se la lumaca si sposta verso destra, —1 se si sposta verso sinistra.

Similmente, "d" è 1 se la lumaca si sposta verso l'alto, e —1 se lo spostamento avviene verso il basso. Poiché nessuna delle due lumache resta mai ferma, il programma non porterà mai a zero "c" e "d" contemporaneamente. Il calcolo della posizione successiva, avviene aggiungendo "c" ad "a" e "d" a "b".

In modo analogo le variabili u, v, w e x agiscono sulla lumaca 2.

Per disegnare l'ultima posizione della lumaca sullo schermo, viene usato il comando PLOT, mentre la funzione POINT rileva se questa va a toccare da qualche parte. Per esplorare la tastiera si usa IN (vedere capitolo 8) al posto di INKEY$, in modo da evitare anomalie nel caso i due contendenti premano assieme i tasti.

```
    5 GO TO 300
   10 REM Lumaca 1
   20 LET e=(IN 65022=253)-(IN 65
022=254)
   30 LET f=(IN 64510=253)-(IN 65
278=253)
   40 IF e<>0 OR f<>0 THEN IF NOT
  (e<>0 AND f<>0) THEN LET c=e: L
ET d=f
   50 LET a=a+c: LET b=b+d
   60 IF POINT (a,b) THEN LET s=2
   GO TO 200
   70 PLOT a,b
  100 REM Lumaca 2
  110 LET y=(IN 49150=253)-(IN 49
150=251)
  120 LET z=(IN 57342=253)-(IN 32
766=253)
  130 IF y<>0 OR z<>0 THEN IF NOT
  (y<>0 AND z<>0) THEN LET w=y: L
ET x=z
  140 LET u=u+w: LET v=v+x
  150 IF POINT (u,v) THEN LET s=1
: GO TO 200
  160 PLOT u,v
  170 GO TO 20
  200 REM fine
  210 PAUSE 1: PAUSE 100: CLS
  220 PRINT AT 10,10;"Lumaca ";s,
" Ha vinto"
  230 PRINT AT 12,8;"Tenta di nuo
vo ";(NOT (s-1))+1
  240 STOP
  300 LET a=20: LET b=155: LET c=
1: LET d=0
  310 LET u=235: LET v=25: LET w=
-1: LET x=0
  320 CLS
  330 LET m=40
  340 FOR n=1 TO 5
  350 FOR p=1 TO 7
  360 IF n<=5 AND p<=6 THEN PLOT
m*p-28,m*n-32: DRAW 30,0
  370 IF n<=4 THEN PLOT m*p-33,m*
n-27: DRAW 0,30
  380 NEXT p
  390 NEXT n
  400 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175
  410 GO TO 10
```

20-170	Loop principale del programma eseguito per muovere ogni lumaca di un pixel.
20-40	Rileva se qualche tasto del giocatore di sinistra (W, A, S, Z) viene azionato e quindi predispone le variabili "c" e "d".
50	Calcola la nuova posizione della lumaca 1.
60	Fine programma se è andata a picchiare da qualche parte.
70	Disegna la nuova posizione della lumaca 1.
100-160	Routine simile per la lumaca 2.
200-240	Fine della routine.
300-410	Partenza della routine che genera il labirinto e predisposizione delle direzioni iniziali delle lumache. Queste funzioni sono state messe alla fine del listato per poter dare al loop principale (linee 20-170) velocità massima.

PUNTI IN MOVIMENTO

Nella maggior parte dei giochi è più utile muovere degli oggetti invece di far aumentare la lunghezza di una riga. Per ottenere ciò è necessario cancellare la vecchia immagine disegnandone una nuova:

```
  10 FOR a=1 TO 255
  20 PLOT a,0: PLOT INVERSE 1;a-
1,0
  30 NEXT a
  40 PLOT INVERSE 1;255,0: GO TO
 10
```

(La linea 40 cancella l'ultimo punto prima di ripetere il movimento).

Facendo girare la routine, si ha l'impressione, grazie a PLOT, di un movimento continuo, anche se l'oggetto è piccolo e la velocità non eccessiva.

Provate inserendo il comando PRINT AT:

```
  10 FOR a=1 TO 31
  20 PRINT AT 0,a;"*";AT 0,a-1;"
#"
  30 NEXT a
  40 PRINT AT 0,31;"#": GO TO 10
```

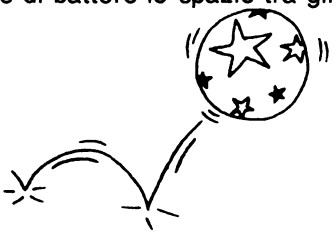
che rende molto meglio l'idea, infatti potete rallentare ulteriormente la corsa, aggiungendo la linea;

```
25 PAUSE 3
```

per mezzo della quale potrete vedere più distintamente la stella. In pratica, il programma deve fare molte altre cose per far muovere un oggetto, confermando l'inadeguatezza del BASIC per la generazione dei movimenti, i quali, comunque, sono spesso sufficientemente veloci.

In entrambe le routines viste abbiamo disegnato il nuovo oggetto prima di cancellare il vecchio, poiché in questo modo si riesce ad ottenere un movimento continuo, specie se tra i due eventi vi è un certo ritardo.

Non dimenticate di battere lo spazio tra gli apici alle linee 20 e 40.



CADUTA DI ELEFANTI

Una caratteristica interessante di PRINT AT per mostrare degli oggetti in movimento è la possibilità di usare anche caratteri grafici.

La prossima gara, semplice ma divertente, si rifà al detto che gli elefanti scendono dagli alberi sedendosi su una foglia e aspettando l'autunno.

Il gioco inizia con 32 elefanti liberi in cielo. Cadono però uno alla volta e voi dovrete accorrere sotto di essi con un telo di salvataggio per recuperarli (usando i tasti 5 e 8), prima che si schiantino a terra. Il tutto è accompagnato da "musica".

Alla fine della gara ci viene mostrato il numero di elefanti rimasti in vita. Se qualcuno si lamenta del fatto che il telo di salvataggio non può muoversi abbastanza velocemente per salvare tutti gli elefanti, consolatelo dicendogli che "é la vita".

```
  100 DATA 0,0,0,0,0,51,126,255
  110 DATA 0,48,126,95,127,127,17
9,51
  120 DATA 0,0,0,0,0,0,195,126
  130 FOR a=0 TO 23: READ b: POKE
  USR "a"+a,b: NEXT a
  140 RANDOMIZE
  150 DIM h(32): LET m=16: LET s=
0
  160 BORDER 1: PAPER 1: CLS
  170 PRINT AT 21,0; PAPER 4;,,
  180 FOR a=31 TO 0 STEP -1: PRIN
T INK 7;AT 0,a;"A": PAUSE 10: NE
XT a
  200 FOR t=1 TO 32
  210 LET x=INT (32*RND): IF h(x+
1)<>0 THEN GO TO 210
  220 LET h(x+1)=1
  300 FOR y=1 TO 19
  310 LET n=m
  320 IF INKEY$="5" AND m>0 THEN
LET n=n-1
  330 IF INKEY$="8" AND m<31 THEN
  LET n=n+1
  340 PRINT AT 19,m;" ";AT 19,n;
INK 3;"U";AT y,x; INK 7;"A";AT y
-1,x;" "
  350 LET m=n: BEEP .005,25-y
  360 NEXT y
```

```
 400 IF x<>m THEN PRINT AT 19,x;
" ";AT 20,x; INK 2;"▄": BEEP .2,
-50: GO TO 430
 410 BEEP .05,10: BEEP .05,15: B
EEP .1,22: LET s=s+1
 420 PRINT AT 19,x;" "; INK 7;AT
 20,x;"▀"
 430 NEXT t
 500 PRINT AT 5,2; INK 9;"HAI SA
LVATO ";s;" ELEFANTI"
```

Tenete presente che le lettere maiuscole "A", "B", e "C", inserite nel listato, vanno battute come caratteri grafici A, B e C.

Dopo aver eseguito il programma una volta, essi appariranno nel listato come tali.

100-130	Stabilisce i tre caratteri grafici i cui otto valori binari, necessari alla definizione di ogni carattere, vengono trasformati in altrettanti numeri decimali per una più facile codifica (linee 100-120).
150	Predispone i 32 elementi dell'array h( ) mettendoli inizialmente a zero. Ciò serve da "flag" alla caduta di ciascuno dei 32 elefanti. Posiziona "m" (telo di salvataggio) a 16 e il punteggio "s" a zero.
160-170	Predispone lo sfondo e disegna la terra.
180	Genera la linea dei 32 elefanti.
200-430	Loop principale del programma eseguito per ogni elefante.
210-220	Sceglie un elefante che non sia ancora caduto e lo fa scendere.
300-360	Loop interno eseguito alla caduta di ogni elefante.
310-330	Calcola la nuova posizione del piattino (n) partendo dalla vecchia (m) e permette l'uso dei tasti 5 e 8.
340	Cancella le vecchie immagini dell'elefante in caduta e del piattino disegnandone nuove nella posizione che segue.
350	Aggiorna la posizione del vecchio piattino e genera il rumore.
360	Chiude il loop del numero degli elefanti.

400-420 L'elefante è sceso al livello del piattino; genera l'adeguato rumore e la sagoma adatta a seconda che l'elefante sia caduto o meno (x=m).

430 Lettura per il prossimo elefante.

500 Presenta il punteggio finale. Il colore di INK è portato a 9 (contrasto) per evitare la scelta di un colore.

BREAKOUT

Quando muovete gli oggetti sullo schermo, generalmente dovete anche rilevare quando questi entrano in collisione tra loro. Un sistema per ottenere ciò è di tenere in memoria la posizione di ciascun oggetto sullo schermo. Supponete di predisporre un array come "mappa" dello schermo e vedrete che ogni singolo valore degli elementi costituenti viene presentato costantemente come una parte dello schermo stesso. Un altro sistema è quello di stendere il programma in modo che le differenti voci vengano scritte con diversi colori (o con differenti attributi BRIGHT o FLASH). La funzione ATTR vi dice ciò che si verifica in ogni punto dello schermo.

ATTR necessita di una coppia di valori come la voce AT nello statement di PRINT, ma, a differenza di questa, deve essere posta tra parentesi (che linguaggio pignolo!). La funzione rende un valore intero compreso tra 0 e 255 come somma di:

```
  128 in caso di lampeggio
+  64 in caso di maggior luminosità
+   8 colori di paper (0-7)
+     il colore di ink (0-7).
```

Per estrarre i singoli attributi, bisogna usare le routine che seguono;

```
LET a=ATTR(y,x)
flash =INT(a/128)
bright=INT((a-128*INT(a/128))/64)
paper =INT((a-64*INT(a/64))/8)
ink=a-8*INT(a/8)
```

Il gioco del BREAKOUT è un ottimo esempio di come usare ATTR per estrarre quanto è presente sullo schermo. Come saprete, il gioco inizia presentando un muro costituito da più righe di mattoni, che va abbattuto colpendolo con una palla. All'impatto con la vostra palla ogni mattone svanisce, aumentando di 1 il punteggio. Più i mattoni sono interni e più il loro valore aumenta.

Si inizia con 10 palle, e se ne perde una ogni volta che si sbaglia la battuta. La vostra racchetta è controllata dai tasti K (su) e M (giù); la palla viene rinviata con un'inclinazione che dipende dal punto di incon-

tro. Se riuscite ad aprire una falla nel muro e la palla vi penetra attraverso, distruggerà i mattoni più interni, che danno un punteggio più alto. Dando colori diversi alle varie righe di mattoni, la funzione ATTR non solo rivelerà se il muro è stato toccato, ma anche quale riga sia stata interessata. Tutto ciò rende il gioco particolarmente attraente. Usando ATTR, non lasciatevi trarre in inganno dal fatto che tutte le celle di carattere presenti sullo schermo hanno il medesimo colore, anche se nella cella non compare nulla. Così:

```
PAPER 7: CLS: PRINT AT 3,3;INK 4;" "
```

può portare anche un display vuoto, valutando con ATTR il colore INK della cella 3,3 troverete il valore 4.

Per tale ragione, il programma usa colori permanenti di INK e PAPER con valore 0, cambiandoli temporaneamente per mezzo delle funzioni PAPER e INK incluse in statement di PRINT. La variazione avviene esclusivamente quando è necessario che appaia qualcosa sullo schermo. Perciò:

```
PRINT AT y,x;" "
```

non solo cancella ogni vecchia immagine, ma attribuisce anche il valore zero al colore di ink.

```
 100 PAPER 0: INK 0: BORDER 7: I
NVERSE 0: OVER 0: FLASH 0: CLS
 110 DATA 0,126,126,126,126,126,
126,0
 120 DATA 0,60,126,126,126,126,6
0,0
 130 FOR a=USR "a" TO USR "a"+15
: READ b: POKE a,b: NEXT a
 140 RANDOMIZE : LET t=10: LET s
=0: LET c=10
 150 PRINT INK 6;"PUNTI : 0"
 160 FOR a=1 TO 10: PRINT INK 6;
AT 0,21+a;CHR$ 145: NEXT a
 170 FOR a=1 TO 5: FOR b=1 TO 21
 180 PRINT PAPER 7; INK 6-a;AT b
,a;CHR$ 144
 190 NEXT b: NEXT a
 200 FOR t=1 TO 10: PRINT AT 0,2
1+t;" "
 300 LET x=6: LET y=1+INT (21*RN
D)
```

```
 310 LET h=1: LET v=1: IF RND>.5
 THEN LET v=-1
 400 PRINT INK 6;AT y,x;CHR$ 145
 410 IF INKEY$="k" AND c>1 THEN
PRINT AT c,31;" ": LET c=c-1
 420 IF INKEY$="m" AND c<21 THEN
 PRINT AT c,31;" ": LET c=c+1
 430 PRINT INK 7;AT c,31;CHR$ 13
8: IF x>30 THEN GO TO 600
 440 IF x=30 THEN LET d=c-y: IF
d=0 OR d=v THEN LET h =-1: BEEP
.02,12: IF d=v THEN LET v=-v
 450 IF x<1 THEN LET h=1: BEEP .
02,0
 460 IF y<2 THEN LET v=1: BEEP .
02,0
 470 IF y>20 THEN LET v=-1: BEEP
 .02,0
 480 PRINT AT y,x;" ": LET x=x+h
: LET y=y+v: LET a=ATTR (y,x)
 490 LET a=a-8*INT (a/8): IF a=0
 OR a=7 THEN GO TO 400
 500 PRINT INK 7;AT y,x;"*": BEE
P .05,12+3*a
 510 LET s=s+a: PRINT INK 6;AT 0
,8;s; INK 0;AT y,x;" "
 520 GO TO 300
 600 BEEP .1,-20: PRINT AT y,x;"
 "
 610 NEXT t
 700 PRINT INK 6;AT 0,20;"FINE P
ARTITA"
```

100 Predispone gli attributi del display al loro valore nominale.

110-130 Abilita i caratteri grafici "A" e "B" per fargli prendere il posto rispettivamente dei mattoni e della palla.
Nel listato si riferiscono a "CHR$ 144' e 'CHR$155".

140 Predispone i valori di partenza del punteggio e della posizione della racchetta.

150-190 Scrive il titolo e disegna le righe di mattoni.

200-610 Loop principale del programma eseguito per ogni palla.

300-310 Lancio della palla dalla posizione x,y con velocità orizzontale h=1 e velocità verticale v di +1 o −1.

400-490 Loop interno eseguito per muovere la palla attraverso lo schermo finché questa non sparisca dal lato destro o colpisca i mattoni. Permette anche al giocatore di muovere la racchetta.

400 Disegna la palla in posizione aggiornata.

410-420 Cancella la vecchia racchetta e calcola la sua nuova posizione alla pressione dei tasti K o M.

430 Algoritmo per preparare la palla ad un corretto rimbalzo sulla superficie o sulle sponde della racchetta se questa è posizionata correttamente.

450-470 Predispone la palla al rimbalzo sui lati superiore, inferiore o di sinistra dell'area di gioco.

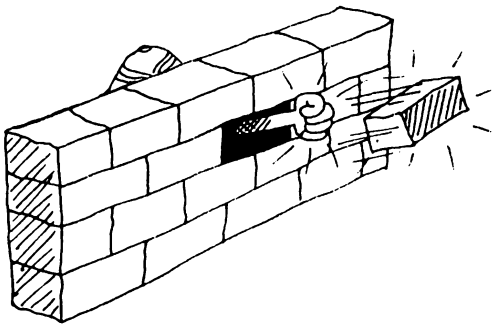
480 Cancella la vecchia palla, calcolandone la nuova posizione.

490 Se il colore INK della nuova posizione della palla è 0 (nero=posizione vuota) o 7 (bianco=racchetta), esce dal loop alla linea 400 per muovere nuovamente la palla.

500-520 La palla ha toccato i mattoni, in tale posizione lampeggia una stella, viene generato un suono, incrementa il punteggio quindi torna alla linea 300 per un nuovo servizio della palla.

600-610 La palla è fuori gioco, se ve ne sono ancora ne prende una e il gioco continua.

700-710 Avete perso l'ultima palla, la partita è finita.



UCCELLINI

Potete creare l'impressione di movimento anche eseguendo ripetuti "scrolling" su tutto lo schermo come mostrerà questo programma.

Immaginate di essere un uccello, da qui il nome, che vola beatamente lungo il lato superiore dello schermo guidato verso sinistra dal tasto "5" e verso destra dall'"8". Come natura comanda, per sopravvivere dovete mangiare, catturando il numero più alto possibile di insetti verdi, liberi sotto di voi (sullo schermo appaiono come dei punti a causa delle loro piccole dimensioni). Il numero sulla parte bassa dello schermo, indica quante riserve di cibo avete a disposizione: se non darete la caccia agli insetti, tale numero scenderà sempre di più fino a raggiungere lo zero che vi indicherà la prossima fine. Come se non bastasse, vi vedrete arrivare addosso uno sciame di frecce, lanciate dai cacciatori, e voi dovrete fare di tutto per evitarle.

Nel programma "Breakout" visto poco fa, abbiamo usato la funzione ATTR per distinguere ogni singola posizione sullo schermo. Qualcosa di simile accade in "Uccellini" per rilevare quando un insetto o una freccia entra in contatto col volatile.

La funzione qui usata non è più ATTR bensì SCREEN$.

SCREEN$ (y,x) restituisce il carattere presente alla colonna x della linea y solo se questo è un carattere compreso tra lo "spazio" e il simbolo di copyright (tra i codici 32-127). Qualora venisse presentato qualsiasi altro carattere, la stringa di ritorno sarebbe nulla (lunghezza zero). Provate a dare un'occhiata ai caratteri riportati nell'Appendice 1. La funzione SCREEN$ non ha accesso all'area degli attributi in RAM, ma solamente all'area display, e non riconosce i caratteri normali dai caratteri inversi.

```
 100 RANDOMIZE : PAPER 7: INK 0:
 BORDER 7: FLASH 0: BRIGHT 0: OV
ER 0: INVERSE 0
 110 DATA 0,0,24,126,189,36,0,0,
0,0,153,126,60,36,00
 120 FOR a=0 TO 15: READ a: POKE
 a+USR CHR$ 144,a: NEXT a
 130 LET s=50: LET x=15
 200 FOR f=144 TO 145: LET s=s-1
: PRINT AT 0,x; INK 1;CHR$ f
 210 IF INKEY$="5" AND x>0 THEN
LET x=x-1
```

```
  220 IF INKEY$="8" AND x<31 THEN
  LET x=x+1
  230 PRINT INK 2;AT 19,31*RND;"↑
";AT 19,31*RND;"↑"
  240 IF RND<.3 THEN PRINT INK 4;
AT 19,31*RND;"."
  250 POKE 23692,0: PRINT ',,,s:
IF s<=0 THEN GO TO 300
  260 LET p$=SCREEN$ (0,x): IF p$
="↑" THEN GO TO 300
  270 IF p$="." THEN LET s=s+20:
BEEP .05,30
  280 NEXT f
  290 GO TO 200
  300 FOR a=0 TO 19: BEEP .05,20-
a
  310 PRINT INK 1;AT a,x;" ";AT a
+1,x;CHR$ 145
  320 NEXT a
  330 BEEP .1,-40
```

100 Stabilisce le condizioni iniziali, predispone i due caratteri grafici, i valori iniziali del punteggio (s) e la posizione orizzontale dell'uccello.

200-280 Loop di volo che alterna di due caratteri grafici.

200 Disegna l'uccello nella sua ultima posizione.

210-220 Calcola la posizione seguente se vengono battuti i tasti 5 o 8.

230 Disegna a caso due nuove frecce sullo schermo in basso.

240 Disegna a caso un nuovo insetto sullo schermo in basso.

250 Scrive il numero della riserva di cibo e provvede allo "scroll" di una linea.

260-270 Rileva la presenza di qualunque oggetto venga a trovarsi nella posizione dell'uccello e prende le opportune decisioni.

290 Torna per il prossimo volo.

300-330 Morte dell'uccello.

ESAME DI GUIDA

Potete fare in modo che una linea appaia in movimento cancellandola e ridisegnandola continuamente in posizioni sempre adiacenti. È questo un processo assai lento, ma per l'esame di guida è sufficiente. Lo schermo vi mostra la pista così come la si vede al disopra del cofano della vostra auto da corsa. Non dovete fare altro che mantenere la vostra auto entro la carreggiata, sterzando con i tasti 5 (a sinistra) e 8 (a destra). State in guardia perché col passare dei chilometri, la pista si restringe sempre di più!

```
 100 DATA 0,224,224,224,224,224,
224,255
 101 DATA 0,0,0,60,126,255,255,2
55
 102 DATA 0,7,7,7,7,7,7,255
 110 FOR a=0 TO 23: READ b: POKE
 a+USR CHR$ 144,b: NEXT a
 120 LET c$=CHR$ 144+CHR$ 145+CH
R$ 146
 130 BORDER 7: PAPER 4: INK 0: C
LS
 140 LET m=0: LET d=123: LET r=d
: LET d1=d: LET r1=r: LET s=0: L
ET w=40
 150 FOR a=1 TO 10: PRINT PAPER
1;: NEXT a
 200 LET m=m+(INKEY$="5")-(INKEY
$="8"): LET d=d+m+4*SIN s
 210 IF d<0 THEN LET d=0
 220 IF d>255 THEN LET d=255
 230 LET r=127+.5*(d-127): IF r<
25 THEN LET r=25
 240 IF r>225 THEN LET r=225
 250 PLOT INVERSE 1;r1-w,0: DRAW
 INVERSE 1;d1-r1+w,136: DRAW INV
ERSE 1;w+r1-d1,-136
 260 LET w=w-.1: PRINT AT 21,14;
c$
 270 PLOT r-w,0: DRAW d-r+w,136:
 DRAW w+r-d,-136
 280 LET d1=d: LET r1=r: LET s=s
+.1
```

```
 290 IF w-ABS (123.5-r)>12 THEN
GO TO 200
 300 PRINT FLASH 1; INK 3;AT 2,3
;" HAI PICCHIATO "
 310 PRINT AT 5,7;"dopo ";s;" ch
ilometri"
```

100-150 Predisposizione delle condizioni iniziali, comprese le tre stringhe di caratteri c$ che disegnano il cofano della vostra auto. Notate la linea 150 che riempie di blu la parte superiore dello schermo, raffigurando il cielo.

200-290 Loop principale.

200 Ritocca la variabile "m", alla pressione dei tasti 5 e 8. Notate che il primo statement di questa linea, equivale alle due linee che seguono:

```
IF INKEY$="5" THEN LET m=m+1
IF INKEY$="8" THEN LET m=m-1
```

210-220 Posizione "d" della congiunzione della pista all'orizzonte che non va mai oltre ai bordi dello schermo.

230-240 Calcola "r" che è la posizione del termine vicino della pista impedendogli di avvicinarsi troppo ai lati dello schermo.

250 Cancella il vecchio tracciato della pista.

260-270 Restringe sempre di più la pista, disegna il cofano dell'auto e la pista nelle nuove posizioni sullo schermo.

280-290 Incrementa la distanza percorsa (s) e salta alla linea 200 se l'auto non è ancora uscita di strada.

300-310 Fine della routine.

CAPITOLO 12
GO SUB, DEF FN E BELLO STILE

Avete visto che spesso è necessario includere gli stessi (o simili) set di istruzioni, in varie sezioni dello stesso programma. Il BASIC vi permette di scrivere tali set una sola volta per poi richiamarli, come subroutine, quando si presenti la necessità.

Una "Subroutine" BASIC sembra una qualsiasi altra parte del programma, con la sola differenza che termina con lo statement RETURN. Conviene spesso stendere il programma come una routine principale che si avvale di routine secondarie, richiamabili più volte, in qualsiasi momento.

Per usare, o meglio, per richiamare una subroutine, battete

```
GO SUB n
```

nel contesto del programma, considerando "n" il numero della prima linea della subroutine. GO SUB è simile a GO TO con la sola differenza che specifica allo Spectrum in quale parte del programma si trova lo statement di GO SUB richiamato. Quando si raggiunge RETURN, posto alla fine di ogni subroutine, il programma compie un salto all'indietro portandosi allo statement immediatamente successivo a quello del GO SUB di chiamata. Ad esempio,

```
 10 PRINT 10
 20 GO SUB 100
 30 PRINT 30
 40 STOP
100 PRINT 100
110 RETURN
```

scriverà i numeri nell'ordine seguente:

```
10
100
30
```

Le linee 10 e 40 compongono il programma principale e le 100, 110 la subroutine.

Indispensabile è lo statement di STOP (linea 40) la cui mancanza farebbe proseguire il programma alle linee 100 e 110.

Tralasciando la linea 110, si incapperebbe nel messaggio di errore

```
RETURN without GO SUB
```

perché lo Spectrum non saprebbe a cosa ritornare.

Per dimostrare che la subroutine può essere richiamata quante volte si vuole, e che GO SUB può apparire anche più di una volta in statement multipli, provate:

```
  10 PRINT 1: GO SUB 100: PRINT
2: GO SUB 100
  20 GO SUB 100: PRINT 3
  30 STOP
 100 PRINT "subr": RETURN
```

che darà

```
1
subr
2
subr
subr
3
```

Anche se è più comodo mettere le subroutine alla fine del programma, nessuno vieta di disporle dove più vi aggrada, facendo attenzine che vengano eseguite solo per opera di un GO SUB.

Una subroutine ne può chiamare un'altra:

```
  10 PRINT 1: GO SUB 100
  20 PRINT 2: GO SUB 200
  30 STOP
 100 PRINT 100: GO SUB 200: RETU
RN
 200 PRINT 200: RETURN
```

da cui risulta la lista

```
1
200
200
2
200
```

Parte della RAM è riservata al "GOSUB Stack". In questa zona vengono memorizzati i numeri di linea e gli statement ai quali si trova GO SUB, in modo che lo Spectrum possa sapere dove saltare quando incontra RETURN.

L'informazione è memorizzata come un elenco al quale il computer aggiunge, dopo aver eseguito il GO SUB, il numero di linea e quello dello statement. Quando lo Spectrum incontra il RETURN, toglie dal fondo dell'elenco il numero di riga e di statement inseriti per ultimi. Il programma perciò ritorna sempre all'istruzione che segue l'ultimo GO SUB eseguito.

Abbiamo visto come una subroutine possa richiamarne una seconda; non vi è alcun limite al loro numero, se non quello dell'area di RAM, necessaria alla memorizzazione degli indirizzi di ritorno. Il BASIC dello Spectrum permette anche che una subroutine richiami se stessa (processo chiamato "ricursione"), fino al raggiungimento di un certo risultato. Ad esempio, il programma che segue calcola il fattoriale di "n" (n!)

```
(n!=n*(n-1)*(n-2) - - - *1)

  10 INPUT "n ? ";n
  20 LET x=n: LET y=1: GO SUB 10
0
  30 PRINT "Fattoriale ";n;" = "
;y
  40 STOP
 100 IF x>0 THEN LET y=y*x: LET
x=x-1: GO SUB 100
 110 RETURN
```

DEF FN

Capita spesso che nello stesso programma, siano presenti calcoli o espressioni simili; in tal caso è possibile scriverli una sola volta, per poi richiamarli con una subroutine.

Queste però hanno lo svantaggio di lavorare solamente con i nomi di variabili specificate nella routine stessa. Ad esempio, volendo calcolare il valore medio di due numeri, è necessario agire come segue:

```
  100 REM LA SUBROUTINE DA' LA ME
DIA DI X E Y
  110 LET media=(x+y)/2
  120 RETURN
```

ma questa subroutine fornisce solo la media dei valori contenuti nelle variabili x e y. Può darsi che nel vostro programma vogliate calcolare anche la media di p e di q, e di c e d. Per usare la subroutine dovreste riassegnare i valori delle variabili ogni volta:

```
  20 LET x=p: LET y=q: GO SUB 10
0
  55 LET x=c: LET y=d: GO SUB 10
0
```

In tutte le funzioni già presenti nello Spectrum si fa però riferimento alle variabili interessate, per cui scriverete SQR(a) per ottenere la radice quadrata di "a" e SQR (z) per ricavare quella di "z". Definendo una nuova funzione potete analogamente specificare la variabile su cui fare le operazioni.

Per definire una nuova funzione bisogna inserire nel programma una istruzione del tipo:

```
DEF FN a(argomenti)=espressioni
```

in cui "a" rappresenta il nome da dare alla nuova funzione. Può essere sia una semplice lettera, sia una lettera seguita da "$" se la funzione porta ad un risultato di stringa. Come già per i nomi, anche qui lo Spectrum non fa distinzioni tra lettere maiuscole e minuscole. Gli "argomenti" posti tra parentesi sono nomi fittizi per i valori da trattare ed anch'essi sono rappresentati da una singola lettera o da una lettera seguita da "$".

Per calcolare la media di due numeri, è necessario definire la funzione FN a, attraverso lo statement:

```
DEF FN a(x,y)=(x+y)/2
```

facendo riferimento a FN a(e,f) dove "e" e "f" sono i due valori dei quali vogliamo ricavare la media. Quando lo Spectrum incontra la funzione FN a(e,f) la ricerca, partendo dalla prima linea del listato, fino a che non trova lo statement "DEF FN a". Trovatolo, risolve l'espressione alla destra dell'uguale rimpiazzando le variabili fittizie (in questo caso "x" e "y") con quelle aggiornate (e ed f). Come al solito, gli esempi valgono di più delle parole per cui provate.

```
10 DEF FN m(x,y,z)=(x+y+z)/3
20 PRINT FN m(1,2,3)
30 LET p=5: LET q=6
40 PRINT FN m(p,7,q)
```

darà come risultato

```
2
6
```

Se gli argomenti tra parentesi sono più di uno, vanno separati dalla virgola, mentre se l'argomento manca le parentesi rimarranno vuote. L'esempio

```
  10 DEF FN r()=PEEK 23730+256*P
EEK 23731
  20 PRINT FN r()
```

stampa il contenuto della Variabile di Sistema RAMTOP.

L'espressione dopo il segno uguale dello statement DEF FN, può essere di qualsiasi tipo comprendendo anche altre funzioni o variabili reali; battete

```
10 DEF FN a(x)=x+y
20 LET y=1
30 PRINT FN a(2)
```

che darà come risultato 3.

Se nel programma vi è una variabile reale con lo stesso nome di una fittizia, viene ignorata al momento del calcolo della funzione. Se per esempio la linea 20 del precedente programma fosse stata

```
20 LET x=99: LET y=1
```

il risultato non sarebbe cambiato.

BELLO STILE

Per quanto il BASIC sia un buon linguaggio per i principianti e per scrivere programmi corti, presenta i suoi inconvenienti quando si tenta di scrivere un programma di una certa lunghezza. Una delle cause, quella meno importante, riguarda la velocità di esecuzione. Dato che il BASIC dello Spectrum è un linguaggio "interpretato", per mezzo del quale il computer traduce o "interpreta" ciascuna istruzione quando la incontra, ne deriva che non è altrettando veloce del linguaggio macchina o di uno dei linguaggi "compilati", quali possono essere il FORTRAN, il FORTH o il Pascal (in linguaggio "compilato" l'intero programma viene tradotto in linguaggio macchina o quantomeno in qualcosa di molto vicino ad esso, prima di essere eseguito).

Ogni volta che il "flusso" di un programma cambia a causa di una istruzione GO TO, di un GO SUB o NEXT, lo Spectrum deve trovare la riga alla quale è indirizzato. Ciò significa che i loop eseguiti alla fine di un programma abbastanza lungo, impiegheranno più tempo ad essere completati che non quelli situati all'inizio. Tale ritardo si verifica anche quando si usa una funzione definita dall'utente con l'istruzione DEF FN.

Il problema principale nello stendere lunghi programmi BASIC non sta però nel computer bensì nell'operatore che li scrive, infatti tanto più lungo è il programma tanto più facile è sbagliare.

Considerate ad esempio, la struttura di un programma, cioé il modo in cui progredisce da una funzione alla successiva. Una istruzione IF..THEN GO TO origina due strade alternative, per cui un programma con sole 10 istruzioni del genere possiede oltre un migliaio di percorsi diversi. Come essere certi che siano tutti giusti? Nel capitolo 4 abbiamo rappresentato graficamente il flusso di un programma: se i vostri grafici assomigliano ad un piatto di spaghetti, con ogni probabilità anche il vostro modo di pensare è altrettanto aggrovigliato. Cercate di evitare, appena possibile, l'uso di GO TO e per rendere le cose più comprensibili, dividete il programma in blocchi chiaramente definiti dotati, se possibile, di una sola entrata e di una sola uscita. I linguaggi più sofisticati, come il Pascal, vi costringono a scrivere in questo modo, nel BASIC dovrete arrangiarvi per conto vostro usando, magari, un blocco separato

di numeri di righe per ciascuna sezione del programma, come per quelli presentati su questo stesso libro.

Potete scrivere anche un programma "principale", molto breve, in grado di richiamare diverse subroutine che assolvano le varie funzioni. In questo caso, il meccanismo delle subroutine servirà come aiuto per comprendere meglio il funzionamento, e non per effettuare successivi richiami. Ad esempio, la sezione principale di un programma di giochi, potrebbe essere

```
  100 REM record=0
  110 GO SUB 1000: REM preparazio
ne
  120 GO SUB 2000: REM gioca la p
artita
  130 GO SUB 3000: REM punteggi
  140 INPUT "Un'altra partita (y/
n)? ";i$
  150 IF i$="y" THEN GO TO 110
  160 IF i$<>"n" THEN GO TO 140
  170 STOP
```

Le subroutine 1000, 2000 e 3000 potrebbero, a loro volta, chiamare altre subroutine per frammentare maggiormente il programma ottenendo unità dimensionate in modo da essere più maneggevoli. Il listato del programma diventa più comprensibile se:

— impiegate istruzioni REM per contrassegnare le sezioni principali. Le righe interessate consistono in un solo numero e nella funzione REM;

— se possibile, fate in modo che la riga assolva una azione completa; per esempio:

```
  100 FOR a=1 TO 10: LET y(a)=a:
NEXT a
  110 FOR a=16 TO 24: LET y(a)=33
: NEXT a
```

è più facile da comprendere che non

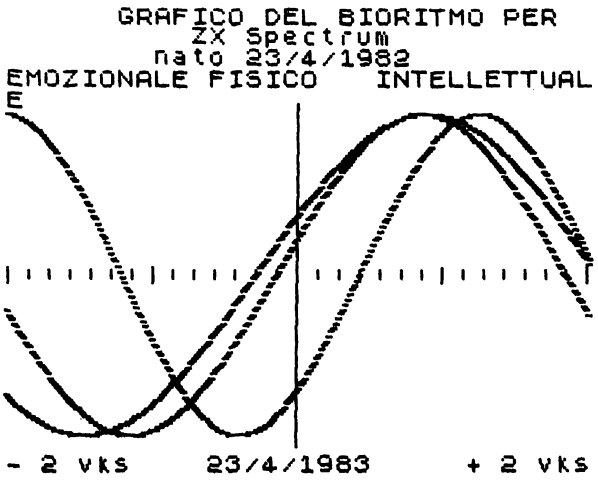
```
  100 FOR a=1 TO 10: LET y(a)=a:
NEXT a: FOR a=16 TO 24
  110 LET y(a)=33: NEXT a
```

— mettete le funzioni FOR e NEXT all'inizio delle righe, tranne quando il loop completo è contenuto sulla medesima linea;

— usate nomi di variabili che abbiano un significato.

Altro problema che potrebbe sorgere, per l'impossibilità di avere sott'occhio contemporaneamente tutti i particolari, è la duplicazione involontaria di nomi di variabili. Anche in questo caso, i linguaggi tipo Pascal, sono superiori, perché permettono l'uso di variabili "locali" valide solamente entro una certa sezione del programma, in modo analogo a quanto avviene per le variabili "fittizie" in istruzioni DEF FN. Nel BASIC dello Spectrum, tutte le variabili sono da considerare "globali", quindi tutto ciò che potete fare è prendere nota dettagliatamente di ogni variabile usata. La parte restante di questo capitolo riguarda una serie di programmi ragionevolmente lunghi, sia per vostro divertimento sia per illustrare i punti sopra descritti.

BIORITMI



Questo programma illustra come si richiamano più volte, e da diverse parti, alcune subroutines.

Esso riprende la teoria secondo la quale il ciclo fisico, quello emozionale e quello intellettuale, si ripetono, nei singoli soggetti, rispettivamente ogni 23, 28 e 33 giorni, partendo dal giorno della nascita. Le alternanze delle tre curve vengono disegnate per la durata di quattro settimane, attorno alla data prescelta. I giorni critici sono quelli in corrispondenza dei quali le curve intersecano la linea centrale. Potete disegnare i grafici con vari colori, sperimentandone diversi e scegliendo quelli che secondo voi sono esteticamente più gradevoli. Un buon effetto si ha tracciando le curve e gli assi con i colori magenta (rosso), verde, ciano e giallo il tutto su sfondo nero. Per una migliore resa, i grafici sono spessi solamente tre pixel.

Il programma comprende due subroutine, con inizio alle linee 1000 e 2000 ripetute continuamente, con piccole variazioni.

```
  100 PAPER 7: INK 0: BORDER 7: I
NVERSE 0: OVER 0: FLASH 0: CLS
  110 PRINT AT 5,10; INK 3;"BIORI
TMI": INPUT "Come ti chiami ?";
LINE n$
```

```
 120 PRINT AT 8,0;"Ciao ";n$;AT
10,0;"Quando sei nato ?"
 130 GO SUB 1000: PRINT AT 10,19
;d$: LET b$=d$: LET z=x
 140 PRINT AT 12,0;"Quale data i
nteressa ?": GO SUB 1000
 150 PRINT AT 12,17;"  ";d$: LET
 d=x-z
 160 INK 3: PRINT AT 15,0;"A que
sta data"
 170 PRINT "hai ";d;" anni,"
 180 PRINT "hai mangiato ";3*d;"
 pasti,"
 190 PRINT "e hai dormito per";8
*d;" ore."
 200 INPUT "Premi ENTER per il t
uo grafico "; LINE i$
 210 PAPER 0: INK 6: BORDER 0: C
LS
 220 PRINT TAB 6;"GRAFICO DEL BI
ORITMO PER "
 230 PRINT TAB 15-LEN n$/2;n$'TA
B 12-LEN b$/2;"nato ";b$
 240 FOR a=1 TO 255 STEP 9: PLOT
 a,73: DRAW 0,3: NEXT a
 250 FOR a=1 TO 255 STEP 63: PLO
T a,71: DRAW 0,6: NEXT a
 260 PLOT 127,10: DRAW 0,128
 270 PRINT AT 21,0;"- 2 yks";TAB
 25;"+ 2 yks"
 280 PRINT AT 21,15-LEN d$/2;d$
 300 INK 3: PRINT AT 3,0;"EMOZIO
NALE": LET c=28: GO SUB 2000
 310 INK 4: PRINT AT 3,11;"FISIC
O": LET c=23: GO SUB 2000
 320 INK 5: PRINT AT 3,20;"INTEL
LETTUALE": LET c=33: GO SUB 2000
 330 PAPER 7: INK 0: BORDER 7
 340 STOP
1000 DATA 0,31,28,31,30,31,30,31
,31,30,31,30,31
1010 INPUT "Anno ? ";y: LET x=36
5*y+INT (y/4)-INT (y/100)
1020 INPUT "Mese (1-12) ? ";m: I
F m<1 OR m>12 THEN GO TO 1020
1030 RESTORE : FOR a=1 TO m: REA
D b: LET x=x+b: NEXT a
1040 LET l=y=4*INT (y/4) AND y<>
100*INT (y/100)
1050 IF l AND m>2 THEN LET x=x+1
1060 READ b: IF l AND m=2 THEN L
ET b=29
1070 INPUT ("Giorno (1-";b;") ?
");d: IF d<1 OR d>b THEN GO TO 1
070
1080 LET x=x+d: LET d$=STR$ d+"/
"+STR$ m+"/"+STR$ y
1090 RETURN
```

```
2000 FOR a=0 TO 253: LET b=d-14+
a/9
2010 PLOT a,74+60*SIN (2*PI*b/c)
: DRAW 2,0
2020 NEXT a
2030 RETURN
```

100-150 Chiede la data di nascita e quella attuale per mezzo della subroutine alla linea 1000. Calcola il numero di giorni (d) che intercorrono tra di loro e forma due stringhe (b$ e d$) contenenti le date.

160-190 Scrive alcune informazioni curiose.

200 Attende il via dall'operatore.

210-280 Disegna il reticolo e stampa i titoli. Notate la tecnica adottata alle linee 230 e 280 per centrare le stringhe rispetto alle linee.

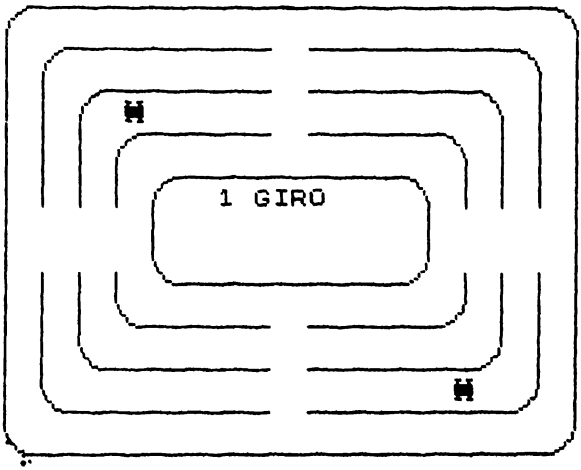
300-320 Disegna i tre grafici con diversi colori usando la subroutine alla linea 2000.

330-340 Varia i colori di paper, ink e border a seconda dei vostri gusti, quindi si ferma.

1000 Prima subroutine usata per la data: controlla la validità del mese e del giorno e genera la stringa standard contenente la data sotto la forma "dd/mm/yyyy". Fornisce il numero "x" necessario al calcolo dei giorni trascorsi tra le due date. La linea 1040 è particolarmente interessante in quanto riconosce "vera" la variabile "l" se l'anno è bisestile, "falsa" nel caso contrario. Tale variabile è usata anche dalle linee 1050 e 1060 per la sistemazione del numero dei giorni.

2000 Subroutine che disegna le curve.

COLLISIONI



Voi e lo Spectrum state guidando, ognuno con la propria auto da corsa, in un tracciato a quattro piste. Ma non si tratta di una vera e propria corsa: le auto, infatti, procedono in direzione opposta e quella guidata dallo Spectrum farà di tutto per venirvi addosso! Quanti giri saprete resistere prima che avvenga l'inevitabile fine?

Alla partenza il progamma vi chiederà il grado di difficoltà che dovrete inserire battendo un numero compreso tra 1 e 9. Ovviamente, più questo è alto e più aumenta la vostra pazzia suicida e omicida del vostro avversario. Potete eseguire un movimento alla volta e l'unico controllo che avrete a disposizione è quello dello spostamento da una pista all'altra all'altezza delle aperture. Il tasto "I" sposta verso l'interno, il tasto "O", verso l'esterno.

```
  100 GO SUB 9000: REM preparati
per la prima corsa
  200 IF yy=21-yt THEN LET nyx=yx
+1: IF yx=30-yt THEN LET nyy=yy-
1: LET y$=CHR$ 145
  210 IF yy=yt THEN LET nyx=nyx-1
: IF yx=yt+1 THEN LET nyy=yy+1:
LET y$=CHR$ 145
  220 IF yx=yt THEN LET nyy=yy+1:
  IF yy=20-yt THEN LET nyx=yx+1:
LET y$=CHR$ 144
  230 IF yx=31-yt THEN LET nyy=yy
-1: IF yy=yt+1 THEN LET nyx=yx-1
: LET y$=CHR$ 144
  240 IF nyx=15+(yy>12) OR nyy=11
  THEN LET yf=2*((yt<7 AND INKEY$
="i")-(yt>1 AND INKEY$="o"))
```

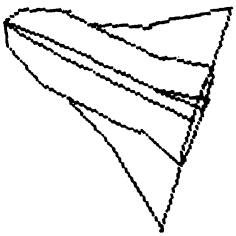
```
 250 IF yf<>0 THEN LET nyx=nyx+(
(nyx=yt)-(nyx=31-yt))*SGN yf: LE
T nyy=nyy+((nyy=yt)-(nyy=21-yt))
*SGN yf: LET yt=yt+SGN yf: LET y
f=yf-SGN yf
 260 IF nyy=cy AND nyx=cx THEN G
O TO 400
 300 IF cy=21-ct THEN LET ncx=cx
-1: IF cx=ct+1 THEN LET ncy=ncy-
1: LET c$=CHR$ 145
 310 IF cy=ct THEN LET ncx=cx+1:
 IF cx=30-ct THEN LET ncy=cy+1:
LET c$=CHR$ 145
 320 IF cx=ct THEN LET ncy=cy-1:
 IF cy=ct+1 THEN LET ncx=cx+1: L
ET c$=CHR$ 144
 330 IF cx=31-ct THEN LET ncy=cy
+1: IF cy=20-ct THEN LET ncx=cx-
1: LET c$=CHR$ 144
 340 IF (ncx=15+(ncy>8)) OR ncy=
11 THEN IF RND<s/10 THEN LET cf=
2*SGN (yt-ct)
 350 IF cf<>0 THEN LET ncx=ncx+(
(ncx=ct)-(ncx=31-ct))*SGN cf: LE
T ncy=ncy+((ncy=ct)-(ncy=21-ct))
*SGN cf: LET ct=ct+SGN cf: LET c
f=cf-SGN cf
 400 PRINT AT yy,yx;" ";AT cy,cx
;" ";AT nyy,nyx; INK 1;y$;AT ncy
,ncx; INK 2;c$
 410 LET yx=nyx: LET yy=nyy: LET
 cx=ncx: LET cy=ncy
 420 IF yx=16 AND yy<8 THEN LET
lap=lap+1: PRINT AT 9,12;lap;" G
IRO";("s" AND lap>1)
 430 IF yx<>cx OR yy<>cy THEN GO
 TO 200
 500 FOR a=1 TO 6: FOR b=144 TO
145: BEEP .03,-40: PRINT INK a;A
T yy,yx;CHR$ b: NEXT b: NEXT a
 510 PRINT AT yy,yx;CHR$ 146
 520 IF lap>hi THEN LET hi=lap:
PRINT AT 11,12;"  PUNTI  ";AT 12
,15;hi
 530 INPUT "Premi ENTER per un'a
ltra corsa"; LINE i$
 540 PRINT AT yy,yx;" ": GO SUB
9200: GO TO 200
9000 REM disegna il tracciato
9010 INK 0: PAPER 7: FLASH 0: BR
IGHT 0: OVER 0: INVERSE 0: BORDE
R 7: CLS
9020 FOR a=32 TO 160 STEP 32: PL
OT a/2,a/2-13
9030 DRAW 256-a,0: DRAW 12,12,PI
/2
9040 DRAW 0,176-a: DRAW -12,12,P
I/2
```

```
9050 DRAW a-256,0: DRAW -12,-12,
PI/2
9060 DRAW 0,a-176: DRAW 12,-12,P
I/2
9070 NEXT a
9080 FOR a=2 TO 6: PRINT AT a,15
;"  ";AT a+13,15;"  ": NEXT a
9090 FOR a=10 TO 12: PRINT AT a,
2;"      ";AT a,25;"      ": NEXT
a
9100 REM auto
9110 DATA 231,66,255,255,255,255
,66,231
9120 DATA 189,255,189,60,60,189,
255,189
9130 DATA 36,90,189,126,126,189,
90,36
9140 RESTORE 9100
9150 FOR a=0 TO 23: READ b: POKE
 USR "a"+a,b: NEXT a
9160 LET hi=0
9200 REM valori di partenza
9210 LET yx=15: LET yy=1: LET yt
=1: LET nyx=yx: LET nyy=yy: LET
y$=CHR$ 144: LET yf=0
9220 LET cx=16: LET cy=7: LET ct
=7: LET ncx=cx: LET ncy=cy: LET
c$=CHR$ 144: LET cf=0
9230 LET lap=0: PRINT AT 9,12;"
"
9240 INPUT "Difficolta' (1-9)? "
;s
9999 RETURN
```

100	Subroutine per disegnare le piste, predisposizione dei caratteri e dei valori di partenza.
200-430	Loop principale del programma.
200-230	Calcola la nuova posizione della vostra auto.
240	Esplora i tasti "I" e "O" stabilendo di conseguenza il movimento del flag "yf".
250	Sposta la vostra auto da una pista alla successiva se yf< >0.
260	Abbandona alla collisione con l'avversario.
300-330	Calcola la nuova posizione dell'auto guidata dallo Spectrum.
340	Muove il flag "cf" se lo Spectrum decide di cambiare pista.
350	Sposta l'auto dello Spectrum da una pista alla successiva se cf< >0.

400-410	Cancella il vecchio disegno delle due auto e aggiorna le coordinate.
420	Aggiorna il numero dei giri.
430	Salta alla linea 200 se l'auto avversaria non vi ha centrato.
500-540	Fine della routine; aggiorna il record se lo avete migliorato e vi invita ad una nuova gara.
9000	Subroutine di preparazione alla corsa.
9200	Subroutine per la posizione di partenza di un'altra gara.
yx,yy	Coordinate x e y della vostra auto.
yt	Numero della vostra pista: la pista 1 è all'esterno e le altre sono numerate 3, 5, 7, verso l'interno.
cx,cy,ct	Coordinate e numero della pista dell'auto avversaria.
nyx,nyy	Valori successivi di yx e yy.
ncx,ncy	Valori successivi di cx e cy.
yf,cf	Flags posti a +2 o −2 quando l'auto cambia pista.

3D



Questo programma elabora grafici tridimensionali, che vi potranno essere presentati sotto qualsiasi angolazione e di cui potrete variare le dimensioni e ricavare la prospettiva.

Ciascun grafico è composto da segmenti rettilinei (numero massimo 400) le cui coordinate delle estremità x, y e z, sono contenute in una matrice, che può essere memorizzata o letta separatamente su nastro. Il programma è suddiviso in un certo numero di moduli, selezionati dal "menu" principale, e comprende subroutine e funzioni definite dall'utente per raggiungere gli scopi desiderati.

Quando si dà il RUN al programma, viene innanzitutto presentato il menù principale.

```
Premere il tasto:
   D-Per visualizzare i dati
   I-Per inserire i dati
   L-Per caricare i dati
   P-Per scrivere i dati
   Q-Per lasciare
   S-Per registrare
   V-Per mostrare
```

L'opzione D vi chiederà a partire da quale segmento vorrete che i dati vengano visualizzati, le linee usate per comporre il grafico sono numerate da 1 a 400. Il programma presenterà poi le coordinate x, y e z per ciascuna delle estremità di 17 segmenti consecutivi e vi chiederà se ne volete vedere altri oppure tornare al menù principale.

L'opzione I permette di aggiungere nuovi dati, o di modificare quelli esistenti. Per ogni segmento del grafico, sarà necessario impostare:

— il numero di riga (intero da 1 a 400);

— le coordinate x, y e z per una estremità;

— le coordinate x, y e z per l'altra estremità.

Ogni valore delle coordinate deve essere un intero compreso tra —100 e 100. I sette numeri sono impostati su di una singola riga e i diversi valori vanno separati da spazi. Se uno qualsiasi dei valori impostati non risultasse valido, il programma ignora l'intera riga e si pone in attesa che vengano inseriti i dati idonei. I valori accettati verranno poi presentati sullo schermo ed il cursore si sposterà a capo per la definizione di un nuovo segmento. Potrete continuare l'impostazione in tale modo, oppure tornare al programma principale battendo una riga vuota (quindi premendo semplicemente ENTER). L'ordine dei segmenti non ha alcuna importanza e i vecchi valori possono essere sostituiti battendo una nuova linea con lo stesso numero.

L'opzione L vi permette di caricare un file di dati da nastro, cancellando tutti tutti i dati precedentemente contenuti nel computer. Una volta che il file è stato caricato, lo potrete verificare: se il controllo dà esito negativo, il programma si arresterà e potrà venir fatto ripartire solamente dando un GO TO 1000.

L'opzione P permette di ottenere la stampa di un certo blocco di segmenti per mezzo della stampante. Il programma, in questo caso, richiede il primo e l'ultimo numero dei segmenti componenti il blocco da stampare, i quali andranno inseriti su di una stessa riga, separati da uno spazio.

L'opzione Q arresta il programma, che potrà essere fatto ripartire con GO TO 1000 conservando i dati o con RUN cancellandoli.

L'opzione S registra i dati sotto un nome di file, il quale, come tutti i nomi di file relativi allo Spectrum, non dovrà eccedere i dieci caratteri. A registrazione avvenuta, il programma vi consiglierà la verifica per vedere se tutto è stato eseguito correttamente. In caso negativo, fate ripartire il tutto col solito GO TO 1000 per conservare i dati in memoria.

Per registrare il programma vero e proprio, e non i dati, è necessario bloccarlo (usando l'ozpione Q, oppure premendo CAPS/SHIFT e 6, se il programma stesso è in attesa dell'impostazione dei dati).

Dare CLEAR per cancellare le matrici di dati e tutte le altre variabili, per ridurre il tempo di save e, successivamente, di load del programma.

L'opzione V permette infine di osservare il grafico. Per fare ciò, il programma vi chiede

- i numeri del primo e dell'ultimo segmento del blocco che volete visualizzare;
- un fattore di "scala" sottoforma di un numero intero, compreso tra 1 e 999, che determina le dimensioni del grafico (per iniziare, è meglio impostare 50);
- la rotazione orizzontale in gradi: numero intero tra 0 e 360;
- la rotazione verticale, sempre in gradi e sempre come numero intero tra 0 e 360;
- un valore di profondità (intero tra 0 e 9).

Quest'ultimo valore determina la profondità di prospetto: la cifra 0 non dà alcun effetto, mentre la vista migliore si ottiene con i valori 3 o 4. Ripetiamo che i valori devono essere impostati tutti su di una stessa riga e separati da uno spazio. Se anche uno solo dei dati non risulta idoneo, il programma ignora l'intera riga ed attende dati corretti. Il grafico è basato sui dati relativi al blocco di segmenti preso in considerazione, per cui potrete registrare sotto un solo file i dati riguardanti più di un grafico, oppure visualizzare solo una parte del disegno con dimensioni maggiorate. Se il grafico supera la grandezza dello schermo, il programma si blocca e per farlo ripartire dovrete usare GO TO 1000. A lavoro eseguito, il programma vi chiederà se volete stamparlo su carta per mezzo della stampante.

```
  100 LET ml=400
  110 DIM p$(ml,6): DIM z$(6)
  120 LET t$=" LINE x1 y1 z1 x2 y
2 z2"
1000 REM Menu principale
1010 CLS : PRINT AT 5,7;"Premi i
l tasto";
1020 PRINT TAB 8;"D-display dati
"
1030 PRINT TAB 8;"I-input dati"
1040 PRINT TAB 8;"L-caricamento
dati"
1050 PRINT TAB 8;"P-print dati"
```

```
1060 PRINT TAB 8;"Q-lasciare"
1070 PRINT TAB 8;"S-save dati"
1080 PRINT TAB 8;"V?visualizzazi
one"
1100 LET k$=FN a$(INKEY$)
1110 IF k$="D" THEN GO TO 2000
1120 IF k$="I" THEN GO TO 3000
1130 IF k$="L" THEN GO TO 4000
1140 IF k$="P" THEN GO TO 5000
1150 IF k$="S" THEN GO TO 6000
1160 IF k$="V" THEN GO TO 7000
1170 IF k$<>"Q" THEN GO TO 1100
1200 CLS : PRINT AT 5,0; FLASH 1
;" PROGRAMMA FINITO "
1210 PRINT '''"Per ripartire,ente
r;"
1220 PRINT AT 10,5;"RUN (cancell
a i dati)"
1230 PRINT "o  GO TO 1000";TAB 5
;"(conserva i dati)"
1240 STOP
2000 REM Display dati
2010 CLS : PRINT AT 10,5;"Dalla
linea ?"
2020 GO SUB 8100: LET l=i: IF l<
1 OR l>m1 THEN GO TO 2020
2100 CLS : PRINT t$
2110 FOR a=l TO l+16: IF a>m1 TH
EN GO TO 2200
2120 PRINT 'a;: IF p$(a)=z$ THEN
 GO TO 2140
2130 FOR b=1 TO 6: PRINT TAB 1+4
*b;CODE p$(a,b)-133;: NEXT b
2140 NEXT a
2200 PRINT '''"Ancora (Y/N) ?"
2210 LET k$=FN a$(INKEY$)
2220 IF k$<>"N" AND k$<>"Y" THEN
 GO TO 2210
2230 IF k$="N" OR a>m1 THEN GO T
O 1000
2240 LET l=a: GO TO 2100
3000 REM Input dati
3010 CLS : PRINT "Batti il numer
o di linea,seguito dai dati x,y
& z per il      primo punto,poi
i dati x,y & z per il secondo pu
nto,      Tutte le voci separate
 da spazi."
3020 PRINT '"Oppure premi ENTER
per tornare al menu principale"'
't$
3100 INPUT LINE i$: IF i$="" THE
N GO TO 1000
3110 GO SUB 8000: LET l=i: IF l<
1 OR l>m1 THEN GO TO 3100
3120 FOR a=1 TO 6
3130 GO SUB 8000: IF ABS i>100 T
HEN GO TO 3100
3140 LET p$(l,a)=CHR$ (133+i)
```

```
3150 NEXT a
3160 POKE 23692,0: PRINT #1;
3170 FOR a=1 TO 6: PRINT TAB 2+4
*a;CODE p$(l,a)-133;: NEXT a
3180 GO TO 3100
4000 REM Carica i dati dal nastr
o
4010 CLS : PRINT AT 5,0;"Nome de
l data file ?";CHR$ 8;
4020 INPUT LINE i$: PRINT i$
4030 LOAD i$ DATA p$()
4040 INPUT "Verifica (Y/N)? "; L
INE a$
4050 IF a$="N" OR a$="n" THEN GO
 TO 1000
4060 IF a$<>"Y" AND a$<>"y" THEN
 GO TO 4040
4070 PRINT '"Nuovamente play pe
r verificare"
4080 VERIFY i$ DATA p$()
4090 GO TO 1000
5000 REM Scrivi i dati
5010 GO SUB 8200
5020 PRINT #1;t$
5030 FOR a=l1 TO l2: LPRINT #1;a;
 IF p$(a)=z$ THEN GO TO 5050
5040 FOR b=1 TO 6: LPRINT TAB 2+
4*b;CODE p$(a,b)-133;: NEXT b
5050 NEXT a
5060 LPRINT #1;
5080 GO TO 1000
6000 REM Save data
6010 CLS : PRINT AT 5,0;"Nome de
l data file ?";CHR$ 8;
6020 INPUT LINE i$: IF i$="" THE
N GO TO 6020
6030 IF LEN i$>10 THEN LET i$=i$
( TO 10)
6040 PRINT i$
6050 SAVE i$ DATA p$()
6060 PRINT '"Registra per verifi
care"
6070 PRINT '"Se la verifica si b
locca,il programma si ferma"'"U
sa GO TO 1000 per ricuperare."'"
non RUN"
6080 VERIFY i$ DATA p$()
6090 GO TO 1000
7000 REM Vista disegno
7010 CLS : PRINT AT 5,0;"Inseris
ci;"''
7020 PRINT TAB 4;"Numero della p
rima linea",TAB 4;"Numero dell'U
ltima linea"
7030 PRINT TAB 4;"Fattore di sca
la (1-999)"
7040 PRINT TAB 4;"Rotazione oriz
zontale (0-360)"
```

```
7050 PRINT TAB 4;"Rotazione vert
icale (0-360)"
7060 PRINT TAB 4;"Profondita' (0
-9)"
7070 PRINT '"Separati da spazi"
7100 GO SUB 8100: LET l1=i: IF l
1<1 OR l1>ml THEN GO TO 7100
7110 GO SUB 8000: LET l2=i: IF l
2<l1 OR l2>ml THEN GO TO 7100
7120 GO SUB 8000: LET s=i/100: I
F s<.01 OR s>9.9 THEN GO TO 7100
7130 GO SUB 8000: LET hr=i: IF h
r<0 OR hr>360 THEN GO TO 7100
7140 GO SUB 8000: LET vr=i: IF v
r<0 OR vr>360 THEN GO TO 7100
7150 GO SUB 8000: LET d=i: IF d<
0 OR d>9 THEN GO TO 7100
7200 CLS
7210 LET ch=COS (hr*PI/180): LET
 sh=SIN (hr*PI/180)
7220 LET cv=COS (vr*PI/180): LET
 sv=SIN (vr*PI/180)
7300 FOR l=l1 TO l2: IF p$(l)=z$
 THEN GO TO 7400
7310 LET x1=FN b(1): LET y1=FN b
(2): LET z1=FN b(3)
7320 LET x2=FN b(4): LET y2=FN b
(5): LET z2=FN b(6)
7330 LET xx1=x1*ch-y1*sh: LET xx
2=x2*ch-y2*sh
7340 LET yy1=y1*ch+x1*sh: LET yy
2=y2*ch+x2*sh
7350 LET yy1=yy1*cv-z1*sv: LET y
y2=yy2*cv-z2*sv
7360 LET zz1=z1*cv+yy1*sv: LET z
z2=z2*cv+yy2*sv
7370 LET p=1+d*zz1/1000: LET xx1
=xx1*p: LET yy1=yy1*p
7380 LET p=1+d*zz2/1000: LET xx2
=xx2*p: LET yy2=yy2*p
7390 PLOT xx1+127,yy1+87: DRAW x
x2-xx1,yy2-yy1
7400 NEXT l
7410 INPUT "Stampa (Y/N) ? "; LI
NE i$
7420 IF i$="y" OR i$="Y" THEN CO
PY : GO TO 1000
7430 IF i$="n" OR i$="N" THEN GO
 TO 1000
7440 GO TO 7410
8000 REM Converti il primo numer
o in i$ da i,corto i$
8010 LET sgn=1
8020 LET i=0: IF i$="" OR i$=" "
 THEN RETURN
8030 IF i$(1)<>"-" AND (i$(1)<"0
" OR i$(1)>"9") THEN LET i$=i$(2
 TO  ): GO TO 8020
```

```
8040 IF i$(1)="-" THEN LET sgn=-
1: LET i$=i$( 2 TO ): GO TO 8020
8050 FOR c=1 TO LEN i$: LET c$=i
$(c)
8060 IF c$<"0" OR c$>"9" THEN GO
 TO 8080
8070 NEXT c
8080 LET i=sgn*VAL i$( TO c-1):
LET i$=i$(c TO )
8090 RETURN
8100 REM Input i$,call 8000
8110 INPUT LINE i$: GO TO 8000
8200 REM Stabilisci la prima e l
'ultima linea l1 & l2
8210 CLS : PRINT AT 10,0;"Inseri
sci il primo e l'ultimo numero d
i linea separati da uno spazio;"
8220 INPUT LINE i$: GO SUB 8000:
 LET l1=i
8230 IF l1<1 OR l1>ml THEN GO TO
 8200
8240 GO SUB 8000: LET l2=i
8250 IF l2<1 OR l2>ml THEN GO TO
 8200
8260 RETURN
9000 DEF FN a$(a$)=CHR$ (CODE a$
-(32 AND CODE a$>96))
9100 DEF FN b(z)=(CODE p$(l,z)-1
33)*s
```

1000-1170 Presentazione del menù principale, scelta dell'operatore e salto al blocco selezionato.

1200-1240 Lascia il blocco, il display avverte dell'arresto.

2000-2240 Display del blocco dati.

2010-2030 Accetta il primo numero di linea, lo controlla, cancella lo schermo e scrive il titolo t$.

2110-2140 Loop eseguito 17 volte per visualizzare altrettante linee.

2120-2130 Display del numero di linea e dei 6 dati se esistono.

2200-2240 Chiede all'operatore se vuol vedere altre linee o se vuol tornare al menù principale.

3000-3180 Blocco introduzione dei dati.

3010-3020 Visualizzazione delle istruzioni e del titolo t$.

3100-3180 Loop eseguito per ogni linea.

3100 Accetta la stringa di input dall'operatore. Se è vuota, torna al menù principale.

3110 Preleva il numero di linea dalla stringa e lo controlla.

3120-3150 Preleva col loop le 6 coordinate dalla stringa, le controlla e le sistema nell'array "p".

3160-3180 Mostra il numero di linea e le 6 coordinate (eseguendo se necessario lo scroll dello schermo), quindi torna per il numero successivo.

4000-4090 Carica il blocco dati. Accetta dall'operatore il nome del file, lo mostra, carica l'array p$() verificandolo se richiesto e torna al menù principale.

5000-5080 Scrive il blocco dati.

5010-5020 Prende il primo e l'ultimo numero di linea (11 e 12) del blocco dati presentato. Scrive il titolo t$.

5030-5050 Loop eseguito per scrivere in successione le linee da 11 a 12. Non scrive le coordinate se nella linea non è presente alcun dato.

5040 Loop per disegnare le 6 coordinate della linea.

5060-5080 Disegna le linee e torna al menù.

6000-6090 Registra il blocco dati.

6010-6040 Chiede il nome del file, non accetta la stringa nulla e considera solo i primi dieci caratteri. Visualizza il nome.

6050-6090 Registra su nastro l'array p$() ed esegue la verifica prima di tornare al menù.

7000-7440 Presenta il blocco.

7010-7150 Accetta dall'operatore il primo numero di linea, l'ultimo, i fattori di scala, di rotazione, di profondità e quindi li testa.

7200-7220 Cancella lo schermo e calcola i valori necessari.

7300-7400 Loop eseguito ad ogni disegno della linea.

7310-7320 Estrae da p$() le 6 coordinate per la linea.

7330-7340 Calcola le coordinate per permettere la rotazione orizzontale.

7350-7360 Permette la rotazione verticale.

7370-7380 Funzione per dare l'effetto profondità.

7390 Disegna la linea sullo schermo.

7410-7440 Stampa la hard copy se richiesto e torna al menù principale.

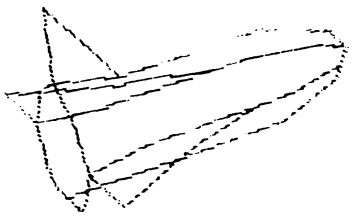
8000-8090 Subroutine che verifica se in i$ c'è un numero intero; dà il valore a "i" e toglie il numero da i$. "i" porta il valore 0 se in i$ non è stato trovato alcun numero. I successivi valori di i$ vengono estratti ripetendo la chiamata continuamente.

8100	Subroutine che accetta dall'operatore la stringa di input (i$) e ne estrae il primo valore usando la linea 8000.
8200	La subroutine chiede all'operatore di inserire i due valori dei numeri di linea (11 e 12), quindi li controlla per vedere se sono idonei.
9000	Definizione del carattere.
9100	Estrazione dall'array dati del valore della coordinata z (z=da 1 a 16) della linea "l".

I dati sono accatastati nell'array di caratteri p$ (400,6) e ogni carattere fa riferimento ad una coordinata in base alla formula:

```
p$(L,A)=CHR$(valore coord +133)
```

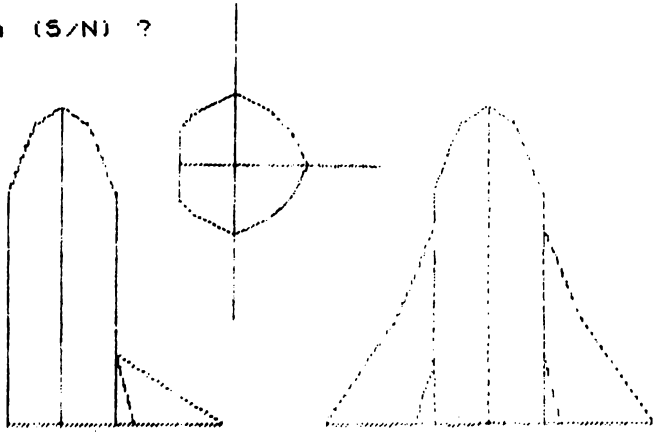
Ciò permette di far lavorare il programma con valori interi da −100 a +100 e non con caratteri tipo "spazio" (codice 32), usato per segnalare la mancanza dati. Le dimensioni dell'array si adattano anche allo Spectrum da 16 K. In quello da 48 K potete aggiungere altre 400 linee, cambiando la variabile "ml" (linea massima) presente alla linea 100 ed attribuendole valori fino a 5800!



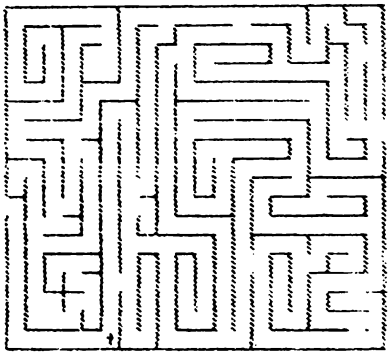
Elenco dei valori per lo Shuttle.

linee	x1	y1	z1	x2	y2	z2
1	-90	0	0	-80	0	15
2	-90	0	0	-80	0	-15
3	-90	0	0	-80	-15	0
4	-90	0	0	-80	15	0
5	-80	15	0	-40	30	0
6	-80	-15	0	-40	-30	0
7	-80	0	-15	-40	0	-30
8	-80	0	15	-40	0	30
9	-40	30	0	50	30	0
10	-40	-30	0	50	-30	0
11	-40	0	30	50	0	30
12	-40	0	-30	90	0	-30
13	-20	30	0	30	50	0
14	-20	-30	0	30	-50	0
15	30	50	0	90	90	0
16	30	-50	0	90	-90	0
17	90	90	0	90	40	0
18	90	-90	0	90	-40	0
19	90	40	0	90	30	-20
20	90	40	0	90	30	20
21	90	-40	0	90	-30	20
22	90	-40	0	90	-30	-20
23	90	40	0	50	30	0
24	90	-40	0	50	-30	0
25	50	0	30	90	0	90
26	90	0	90	90	0	40
27	50	0	30	90	0	40
28	90	0	40	90	-20	30
29	90	0	40	90	20	30
30	90	20	30	90	30	20
31	90	-20	30	90	-30	20
32	90	20	-30	90	30	-20
33	90	-20	-30	90	-30	-20
34	90	-20	-30	90	20	-30

Ancora (S/N) ?



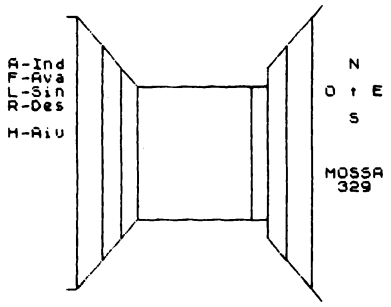
LABIRINTO

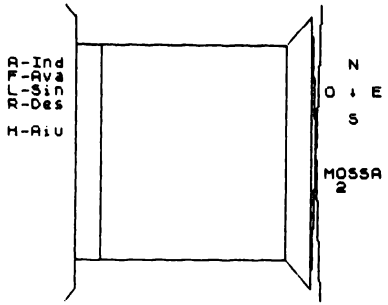


Questo programma genera un labirinto la cui struttura è casuale e nel quale voi dovete cercare la via d'uscita. Il labirinto è costruito su di una griglia a scacchi di cui potete scegliere le dimensioni, entro la gamma 3×3/30×2. I movimenti all'interno del labirinto si effettuano con i tasti:

F — per muovere avanti di un quadretto, se il muro non si trova proprio di fronte.
L/R — per voltarsi a sinistra o a destra.
A — per voltarsi indietro.

Per un più agevole orientamento, contemporaneamente ai corridoi, il display mostra anche i punti cardinali e il numero delle mosse eseguite. Se vi siete persi, ricorrete al tasto H il quale presenterà la pianta del labirinto con il punto in cui siete e la direzione che avete. Ogni aiuto di questo genere, vi costerà, però, una punizione di 50 mosse. L'uscita è indicata da due grosse diagonali incrociate.





```
1000 REM Formazione e presentazi
one del labirinto
1010 RANDOMIZE : INK 0: PAPER 7:
 FLASH 0: BRIGHT 0: OVER 0: INVE
RSE 0: BORDER 7
1020 GO SUB 9100
1030 INPUT "DIMENSIONI DEL LABIR
INTO ; X (3-30) ";mx;TAB 12;" Y
(3-20) ";my
1040 LET mx=INT mx: IF mx<3 OR m
x>30 THEN GO TO 1030
1050 LET my=INT my: IF my<3 OR m
y>20 THEN GO TO 1030
1060 LET move=0
2000 REM Costruzione del labirin
to
2010 FOR a=1 TO mx+1: PLOT 8*a-1
,167: DRAW 0,-8*my: NEXT a
2020 FOR a=1 TO my+1: PLOT 7,175
-8*a: DRAW 8*mx,0: NEXT a
2030 LET x=1: LET y=INT (1+my/4+
RND*my/2): PRINT AT y,0;">"
2040 PLOT 8,8*y-1: DRAW INVERSE
1;0,7
2050 PRINT AT y,x; PAPER 6; OVER
 1;" "
2060 LET dx=1: LET dy=0
2070 LET yourx=x: LET youry=y: L
ET d=0
2100 IF RND>.1 AND dy=0 THEN LET
 dy=INT (2*RND)*2-1: LET dx=0
2110 IF RND>.8 OR y+dy>my OR y+d
y<1 THEN LET dx=1: LET dy=0
2120 GO SUB 9000
2130 IF x<=mx THEN GO TO 2100
2140 PRINT AT y,x;">": LET x=x-1
2200 LET ny=y+dy: LET nx=x+dx
2210 IF RND>.4 AND ny<=my AND ny
>=1 AND nx<=mx AND nx>=1 THEN IF
 ATTR (ny,nx)=56 THEN GO TO 2290
2220 LET u=0: DIM u(4,2)
2230 IF x<mx THEN IF ATTR (y,x+1
)=56 THEN LET u=u+1: LET u(u,1)=
1
2240 IF x>1 THEN IF ATTR (y,x-1)
=56 THEN LET u=u+1: LET u(u,1)=-
1
2250 IF y<my THEN IF ATTR (y+1,x
)=56 THEN LET u=u+1: LET u(u,2)=
1
2260 IF y>1 THEN IF ATTR (y-1,x)
=56 THEN LET u=u+1: LET u(u,2)=-
1
2270 IF u=0 THEN GO TO 2300
2280 LET u=INT (1+u*RND): LET dx
=u(u,1): LET dy=u(u,2)
2290 GO SUB 9000: GO TO 2200
```

```
2300 PRINT AT y,x; PAPER 5; OVER
 1;" "
2310 LET cx=8*x+3: LET cy=171-8*
y
2320 IF POINT (cx+4,cy)=0 THEN I
F ATTR (y,x+1)=48 THEN LET x=x+1
: GO TO 2220
2330 IF POINT (cx-4,cy)=0 THEN I
F ATTR (y,x-1)=48 THEN LET x=x-1
: GO TO 2220
2340 IF POINT (cx,cy-4)=0 THEN I
F ATTR (y+1,x)=48 THEN LET y=y+1
: GO TO 2220
2350 IF POINT (cx,cy+4)=0 THEN I
F ATTR (y-1,x)=48 THEN LET y=y-1
: GO TO 2220
2400 FOR z=1 TO mx*my/10
2410 LET x=2+INT ((mx-2)*RND): L
ET y=2+INT ((my-2)*RND)
2420 LET dx=2*INT (2*RND)-1: LET
 dy=0
2430 IF RND>.5 THEN LET dy=2*INT
 (2*RND)-1: LET dx=0
2440 GO SUB 9000
2450 NEXT z
3000 REM Disegno del labirinto a
 m$()
3010 INPUT "Premi ENTER per comi
nciare "; LINE i$
3020 DIM m$(my+2,mx+2)
3030 POKE 23606,PEEK 23675: POKE
 23607,PEEK 23676-1
3040 FOR y=0 TO my+1: FOR x=0 TO
 mx+1
3050 LET m$(y+1,x+1)=SCREEN$ (y,
x): PRINT AT y,x;" "
3060 NEXT x: NEXT y
3070 POKE 23606,0: POKE 23607,60
4000 REM Movimento attraverso il
 labirinto
4010 CLS : LET p1=87: LET p=80:
LET x=yourx: LET y=youry
4020 PRINT AT 4,29;"N";AT 6,27;"
O ";CHR$ (149+d);" E";AT 8,29;"S
"
4030 LET move=move+1: PRINT AT 1
2,27;"MOSSA";TAB 28;move
4040 PRINT AT 4,0;"A-Ind";"F-Ava
";"L-Sin";"R-Des";"H-Aiu"
4100 FOR z=1 TO 20
4110 PLOT 127-p,87-p: DRAW 0,2*p
: PLOT 127+p,87-p: DRAW 0,2*p
4120 LET c=CODE m$(y+1,x+1): LET
 w=CODE m$(y+1,x): LET s=CODE m$
(y+2,x+1)
4130 IF d=0 THEN LET wr=(s=33)+(
s=35): LET wl=(c=33)+(c=35): LET
 wf=(c=33)+(c=34)
```

```
4140 IF d=1 THEN LET wr=(w=33)+(
w=34): LET wl=(c=33)+(c=34): LET
 wf=(s=33)+(s=35)
4150 IF d=2 THEN LET wr=(c=33)+(
c=35): LET wl=(s=33)+(s=35): LET
 wf=(w=33)+(w=34)
4160 IF d=3 THEN LET wr=(c=33)+(
c=34): LET wl=(w=33)+(w=34): LET
 wf=(c=33)+(c=35)
4170 IF z=1 THEN LET fw=wf
4200 LET dp=p1-p: LET dwr=dp AND
 wr: LET dwl=dp AND wl
4210 LET er=NOT wr AND ((d=3 AND
 x=mx) OR (d=1 AND x=1))
4220 LET el=NOT wl AND ((d=3 AND
 x=1) OR (d=1 AND x=mx))
4230 IF NOT er THEN PLOT 127+p,8
7-p: DRAW dp,-dwr: PLOT 127+p,87
+p: DRAW dp,dwr
4240 IF NOT el THEN PLOT 127-p,8
7-p: DRAW -dp,-dwl: PLOT 127-p,8
7+p: DRAW -dp,dwl
4250 IF er THEN PLOT 127+p,87+p:
 DRAW dp,-(2*p+dp): PLOT 127+p,8
7-p: DRAW dp,2*p+dp
4260 IF el THEN PLOT 127-p,87+p:
 DRAW -dp,-(2*p+dp): PLOT 127-p,
87-p: DRAW -dp,2*p+dp
4270 IF wf THEN PLOT 127-p,87-p:
 DRAW 2*p,0: PLOT 127-p,87+p: DR
AW 2*p,0: GO TO 5000
4300 IF (d=0 AND x=mx) OR (d=2 A
ND x=1) THEN LET pd=.9*p: PLOT 1
27-pd,87-pd: DRAW 2*pd,2*pd: PLO
T 127-pd,87+pd: DRAW 2*pd,-2*pd:
 GO TO 5000
4310 LET x=x+(d=0)-(d=2): LET y=
y+(d=1)-(d=3): LET p1=p: LET p=I
NT (.8*p)
4320 NEXT z
5000 REM Movimento giocatori
5010 LET k$=INKEY$: IF k$>"Z" TH
EN LET k$=CHR$ (CODE k$-32)
5020 IF k$<>"H" AND k$<>"L" AND
k$<>"R" AND k$<>"A" AND k$<>"F"
THEN GO TO 5000
5030 IF k$="H" THEN GO TO 6000
5040 IF k$="F" THEN GO TO 5100
5050 IF k$="L" THEN LET d=d-1: I
F d<0 THEN LET d=3
5060 IF k$="R" THEN LET d=d+1
5070 IF k$="A" THEN LET d=d+2
5080 IF d>3 THEN LET d=d-4
5090 GO TO 4000
5100 IF fw THEN GO TO 5000
5110 IF yourx=1 AND d=2 THEN PRI
NT AT 10,11; FLASH 1;"INGRESSO";
AT 11,11;" CHIUSO ": GO TO 5000
```

```
5120 IF yourx=mx AND d=0 THEN GO
 TO 7000
5130 LET yourx=yourx+(d=0)-(d=2)
: LET youry=youry+(d=1)-(d=3)
5140 GO TO 4000
6000 REM Presentazione del labir
into con posizione aggiornata
6010 CLS : POKE 23606,PEEK 23675
: POKE 23607,PEEK 23676-1
6020 LET xp=INT ((30-mx)/2): LET
 yp=INT ((20-my)/2)
6030 FOR y=1 TO my+2: PRINT AT y
p+y-1,xp;m$(y): NEXT y
6040 POKE 23606,0: POKE 23607,60
6050 LET move=move+50: PRINT OVE
R 1;AT 0,8;"PUNIZIONE DI 50 MOSS
E"
6060 PRINT AT yp+youry,xp+yourx;
 OVER 1;CHR$ (149+d)
6070 INPUT "Premi ENTER per torn
are al labirinto "; LINE i$
6080 GO TO 4000
7000 REM Finale
7010 PAPER 5: CLS
7020 PRINT AT 8,11;"FUORI DOPO";
AT 10,14;move;AT 12,13;"MOSSE"
7030 DATA 0,5,9,5,9,12,9,12,15,1
5
7040 FOR t=1 TO 3: RESTORE 7000
7050 FOR a=1 TO 10: READ b: BEEP
 .05,b: NEXT a
7060 NEXT t: PAPER 7
7070 STOP
9000 REM Distruzione del muro a
x+dx,y+dy
9010 IF dy=0 THEN PLOT 8*(x+(dx=
1))-1,167-8*y: DRAW INVERSE 1;0,
7
9020 IF dx=0 THEN PLOT 8*x-1,175
-8*(y+(dy=1)): DRAW INVERSE 1;7,
0
9030 LET x=x+dx: LET y=y+dy
9040 PRINT AT y,x; PAPER 6; OVER
 1;" "
9060 RETURN
9100 REM User
9110 DATA 0,0,0,0,0,0,0,0
9120 DATA 255,1,1,1,1,1,1,1
9130 DATA 1,1,1,1,1,1,1,1
9140 DATA 255,0,0,0,0,0,0,0
9150 DATA 1,0,0,0,0,0,0,0
9160 DATA 0,0,0,8,124,8,0,0
9170 DATA 0,0,16,16,16,56,16,0
9180 DATA 0,0,0,32,124,32,0,0
9190 DATA 0,0,16,56,16,16,16,0
9200 RESTORE 9100
9210 FOR a=0 TO 71: READ b: POKE
 USR "a"+a,b: NEXT a
9220 RETURN
```

La struttura generale del programma è assai semplice, la sua notevole lunghezza è dovuta alla necessità di fornire diverse possibilità per la costruzione del labirinto e per il tracciato dei suoi corridoi. Si usano due subroutine, la prima per creare la configurazione del labirinto, la seconda, fuori dal programma principale, per disegnare i caratteri grafici durante il gioco.

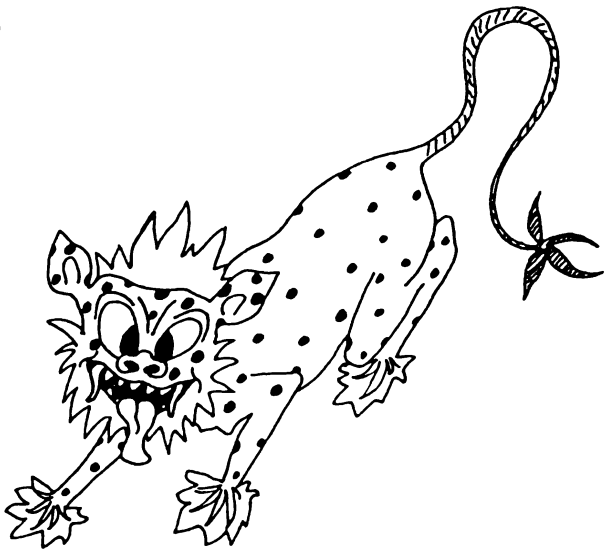
1000-1060 Accetta le dimensioni del labirinto e ne controlla la validità.
2000-2450 Costruzione del labirinto.
2110-2020 Disegna una griglia completa di quadretti.
2030-2070 Apre un punto d'ingresso dalla parte sinistra della griglia.
2100-2130 Traccia un percorso random, cancellando tratti di muro fino a raggiungere il lato destro della griglia. Il percorso è colorato in giallo per farsi riconoscere in seguito dal programma.
2140 Stabilisce l'uscita.
2200-2350 Il programma ricalca il percorso in giallo, aggiungendo altri tracciati, dove può, fino a raggiungere l'uscita.
2200-2210 Colora in bianco le zone circostanti, se il quadrato seguente non è stato esplorato (ATTR=56, bianco).
2220-2280 Cerca un quadrato adiacente libero e se non ve ne sono, salta alla linea 2300.
2290 Toglie il muro e colora il quadrato di giallo.
2300 Non potendo estendere oltre il tracciato in quella direzione, colora il quadretto di ciano.
2310-2350 Trova quale dei quattro quadretti circostanti sia il prossimo del percorso (ATTR=48, giallo), quindi avanza di uno o torna alla linea 2230.
2410-2450 Confonde il percorso aprendo altri varchi a caso.
3000-3070 Copia dallo schermo il labirinto definitivo e lo memorizza in m$( ). I primi 5 caratteri (vedere 9100) sono quelli necessari al disegno del labirinto.
4000-4040 Scrive la bussola, la lista dei comandi e il numero delle mosse.
4100-4320 Disegna la veduta tridimensionale. Il loop dalla linea 4100 alla 4320, limita la profondità della veduta a 20 quadretti, mentre le linee 4270 e 4300 bloccano il disegno quando si presenta un muro o l'uscita.

5000-5140 Accetta il vostro comando, ne controlla la validità, modifica le coordinate (yourx,youry), o la direzione (d) e quindi torna alla linea 4000 per una nuova veduta o alla 6000 se è stato premuto H.

6000-6080 Mostra la pianta scrivendo m$( ) e con la Variabile di Sistema CHARS opportunamente modificata.

7000-7070 Routine per una uscita trionfale.

DRAGONE



È una gara di abilità e di velocità disputata in cima ad un altopiano rettangolare. Potete muovervi usando i tasti direzione 5, 6, 7 e 8 senza però cadere fuori dai bordi del rettangolo!

L'altopiano è ricoperto da boschi e paludi che ostacolano la vostra corsa, rallentandola. Fate attenzione che le paludi sono invisibili! In un punto a caso lampeggerà sempre un numero che rappresenta un tesoro da prelevare; effettuato un prelievo, apparirà un altro numero e così via.

Nel rettangolo di gioco insieme a voi, c'è, però, anche un dragone feroce e affamato la cui corsa è pure rallentata da boschi e paludi. Nella sua disperata rincorsa, abbatte gli alberi passandovi sopra e riducendo così il numero degli ostacoli presenti sull'altopiano. Se dovesse passare sopra al tesoro da raggiungere, questo sparirebbe, e starebbe a voi ricordare il punto esatto dove era per poterlo raccogliere.

Quanto tempo saprete resistere al dragone? Battete il programma che segue e lo vedrete.

```
   5 CLS : RESTORE   LET h=0
  10 FOR a=0 TO 7
  20 READ b: POKE USR CHR$ 116+a
,b
  30 READ b: POKE USR CHR$ 100+a
,b
  40 READ b: POKE USR CHR$ 109+a
,b
  50 NEXT a
  60 FOR a=20 TO 5 STEP -1: BEEP
 .1,a: NEXT a
  70 FOR a=5 TO 20: BEEP .1,a: N
EXT a
  80 GO SUB 800
 100 REM Tue mosse
 110 LET b=b+(b<=0): BEEP .01,10
 120 LET p(w,z)=e: LET i=w: LET
j=z
 130 IF a>=0 THEN LET w=w+(INKEY
$="6")-(INKEY$="7"): LET z=z+(IN
KEY$="8")-(INKEY$="5")
 140 IF p(w,z)=4 OR p(w,z)=9 THE
N LET r=p(w,z): GO TO 600
 150 IF w=x AND z=y THEN GO SUB
500: GO TO 170
 160 IF a>=0 AND p(w,z)<>0 THEN
LET a=p(w,z)-2: BEEP .1,-10
 170 PRINT AT i,j-2;(" " AND e<=
0); INK 4;(CHR$ 163 AND e=1); IN
K 6;(CHR$ 79 AND e=2)
 180 LET e=p(w,z): LET p(w,z)=5
 190 PRINT AT w,z-2;CHR$ 156
 300 REM Mosse del dragone
 310 LET a=a+(a<=0): BEEP .01,-2
0
 320 LET c=0: LET d=0
 330 IF ABS (w-u)>=ABS (z-v) THE
N LET c=SGN (w-u)
 340 IF c=0 THEN LET d=SGN (z-v)
 350 LET p(u,v)=(e AND e<>1)
 360 PRINT AT u,v-2;(" " AND f<>
2); INK 6;(CHR$ 79 AND f=2)
 370 IF b>=0 THEN LET u=u+c: LET
 v=v+d
 380 PRINT AT u,v-2;CHR$ 147
 390 IF p(u,v)=5 THEN LET r=5: G
O TO 600
 400 IF b>=0 AND p(u,v)>0 THEN L
ET b=b-4: BEEP .1,-38
 410 LET f=p(u,v): LET p(u,v)=4
 420 GO TO 100
 500 REM Punteggio
 510 BEEP 1.5,45
 520 LET s=s-k: PRINT AT 0,0;"Re
cord = ";h;TAB 16;"Tuoi punti =
";s
 530 IF p(x,y+1)<7 AND l=1 THEN
LET p(x,y+1)=1: PRINT AT x,y-1;
INK 4;CHR$ 163
```

```
 540 LET p(x,y)=l: PRINT AT x,y-
2; INK 4;(CHR$ 163 AND l=1)
 550 LET x=INT (RND*20+2): LET y
=INT (RND*32+2)
 560 IF ABS (x-u)<6 AND ABS (y-v
)<6 THEN GO TO 550
 570 LET k=-INT (RND*9+1): LET l
=p(x,y)
 580 LET p(x,y)=k: PRINT AT x,y-
2; FLASH 1; INK 6; PAPER 2;-k
 590 RETURN
 600 REM Fine
 610 IF r=9 THEN FOR a=1 TO 15:
BEEP .3,-a-30: NEXT a
 620 IF r<>9 THEN BEEP 5,-38
 630 CLS
 640 IF r>7 THEN PRINT AT 12,6;"
Sei caduto nel burrone": GO TO 6
70
 650 IF r=4 THEN PRINT AT 12,0;"
STUPIDO-Sei finito nel dragone"
 660 PRINT AT 14,8;"Ti ha divora
to !"
 670 PRINT AT 16,10;"Sei morto !
"
 680 PRINT AT 8,10;"Record ";h;T
AB 10;"Hai fatto ";s
 690 IF s>h THEN PRINT AT 4,0; I
NK 1; PAPER 6;"************CAMPI
ONE************"
 700 IF s>h THEN LET h=s
 710 PRINT AT 20,7;"Altra gara ?
 (y/n)"
 720 FOR a=25 TO 5 STEP -1: BEEP
 .1,a: NEXT a
 730 IF INKEY$="y" THEN GO TO 60
 740 IF INKEY$<>"n" THEN GO TO 7
30
 750 BORDER 7: BRIGHT 0: CLS   S
TOP
 800 REM Preparazione
 810 DIM l$(32): DIM p(22,34)
 820 LET s=0: LET a=0: LET b=0
 830 LET c=0: LET d=0: LET e=0:
LET f=0
 840 BORDER 2: INPUT ""
 850 FOR a=1 TO 30
 860 LET x=INT (RND*20+2): LET y
=INT (RND*32+2)
 870 LET p(x,y)=2: PRINT AT x,y-
2; INK 6;CHR$ 79
 880 NEXT a
 890 LET u=INT (RND*20+2): LET v
=INT (RND*32+2)
 900 LET w=INT (RND*20+2): LET z
=INT (RND*32+2)
 910 IF ABS (w-u)<6 AND ABS (z-v
)<6 THEN GO TO 900
```

```
 920 PAUSE 200: CLS : GO SUB 550
 930 PRINT AT 0,0;"RECORD = ";h;
TAB 16;"TOUI PUNTI = ";s, PAPER
2;l$
 940 PRINT AT w,z-2;CHR$ 156: LE
T p(w,z)=5
 950 PRINT AT u,v-2;CHR$ 147: LE
T p(u,v)=4
 960 FOR a=1 TO 175
 970 LET p=INT (RND*20+2): LET q
=INT (RND*32+2)
 980 IF p(p,q)<>0 THEN GO TO 970
 990 LET p(p,q)=1: PRINT AT p,q-
2; INK 4;CHR$ 163
1000 IF a<35 THEN LET p(1,a)=9:
LET p(22,a)=9
1010 IF a<23 THEN LET p(a,1)=9:
LET p(a,34)=9
1020 NEXT a
1030 FOR a=1 TO 10: BEEP .5,20+(
10*(-1 AND a=2*INT (a/2))): NEXT
 a
1040 RETURN
9000 DATA 24,195,24,60,102,24,21
9,36,0,219,126,126,60,195,24,126
,219,24,219,195,36,24,126,36
```

5-80 Predisposizione dei caratteri grafici ed esecuzione della melodia d'apertura. Usa poi la subroutine alla linea 800 per disegnare l'altopiano e memorizzarne l'immagine nell'array p( ).

100-420 Loop principale del programma eseguito in continuazione per muovere sia il dragone che voi. La vostra posizione è stabilita da "w" e "z", quella del dragone da "u" e "w" e quella del tesoro da "x" e "y".

500-590 Subroutine richiamata ad ogni prelievo del tesoro.

600-750 Diversi tipi di fine seguiti dalla richiesta di una nuova gara.

CAPITOLO 13
APPLICAZIONI

Sembra che la maggior parte del software scritto per lo Spectrum sia destinato a giochi di diverso tipo. Ciò non è necessariamente un male, in quanto scrivere programmi di giochi è un modo divertente per imparare a programmare, studiando le strutture e le semplici regole che spesso li caratterizzano; e poi, in fondo, giocare è divertente!

Non dimentichiamoci però che lo Spectrum è, sotto molti aspetti, assai più potente dei computer della prima generazione, studiati per ufficio o per applicazioni tecnico-commerciali. La nostra macchina, infatti, non è limitata ad impieghi di calcolo e ragioneria ma, se munita di una adeguata porta I/O, diventa un dispositivo di controllo ideale per operare nei più svariati laboratori scientifici.

In questo capitolo troverete alcuni esempi di programmi utili che potrete adattare a seconda delle vostre particolari esigenze. Visto che lo scopo di questo capitolo è quello di indicare alcuni possibili impieghi, e non quello di insegnare tecniche di programmazione, non abbiamo fornito le consuete spiegazioni dei listati.

In definitiva, lo Spectrum è particolarmente adatto ad applicazioni che comportino quantità non troppo rilevanti di dati in ingresso o in uscita (fino a file che abbiano un massimo di 30000 caratteri); per il resto, i soli limiti di impiego, sono la vostra immaginazione e il numero di ore disponibili in una giornata!

MEDIE

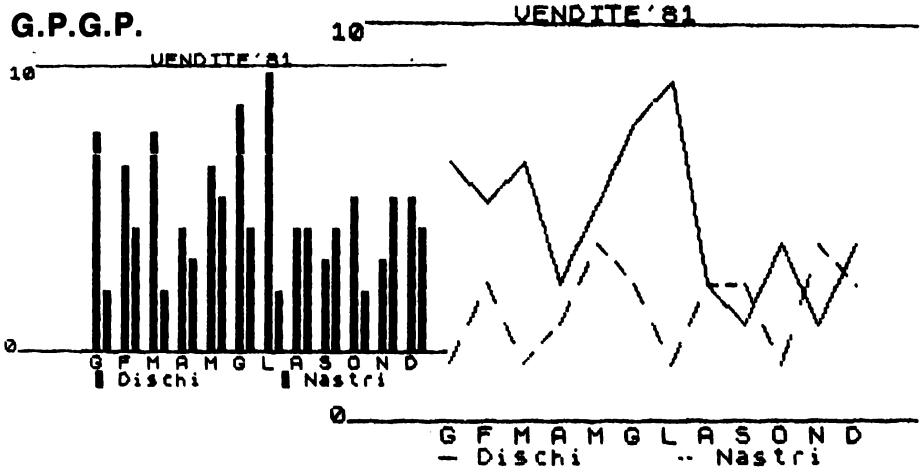
```
Input dati
1 : 1                2 : 2
3 : 4                4 : 6
5 : 15               6 : 32
7 : 64               8 : 80
9 : 90               10 : 95
11 : 95              12 : 90
13 : 80              14 : 64
15 : 32              16 : 16
17 : 8               18 : 4
19 : 2               20 : 1
Media aritmetica= 39.05
Media geometrica = 16.144337
Media armonica = 4.8985292
Valore medio = 53.811244
```

Questo è un programma rivolto agli amanti delle statistiche, in grado di accettare quanti dati numerici si voglia. Di tali numeri calcola la media Aritmetica, quella Geometrica, quella Armonica e anche la Radice quadrata. I dati vanno inseriti uno alla volta. Per terminare la serie, battete "s".

```
  10 REM MEDIE
  20 LET a$="": LET n=0: PRINT "
Input dati :"
  30 INPUT "Dato ";(n+1);i$: IF
i$<>"s" THEN LET n=n+1: LET a$=a
$+i$+CHR$ 124: GO TO 30
  40 DIM x(n)
  50 FOR a=1 TO n
  60 LET b=0
  70 LET b=b+1: IF a$(b)<>CHR$ 1
24 THEN GO TO 70
  80 LET x(a)=VAL a$(1 TO b-1):
LET a$=a$(b+1 TO ): PRINT a;" :
";x(a)
  90 NEXT a
 100 REM Media aritmetica
 110 LET b=0
 120 FOR a=1 TO n: LET b=b+x(a):
 NEXT a
 130 PRINT '"Media aritmetica= "
;b/n
 200 REM Media geometrica
 210 LET b=1
 220 FOR a=1 TO n: IF x(a)>0 THE
N LET b=b*x(a): NEXT a
 230 IF a<=n THEN PRINT "Media g
eometrica impossibile": GO TO 30
0
 240 PRINT "Media geometrica = "
;b↑(1/n)
```

```
  300 REM Media armonica
  310 LET b=0
  320 FOR a=1 TO n: IF x(a)<>0 TH
EN LET b=b+1/x(a): NEXT a
  330 IF a<=n THEN PRINT "Media a
rmonica impossibile": GO TO 400
  340 PRINT "Media armonica = ";n
/b
  400 REM Valore medio
  410 LET b=0
  420 FOR a=1 TO n: LET b=b+x(a)↑
2: NEXT a
  430 PRINT "Valore medio = ";SQR
  (b/n)
  440 INPUT "Enter per ripartire"
;i$: RUN
```

All'ingresso di ogni voce (linea 30), il dato viene aggiunto alla fine della stringa a$ e separato da CHR$ 124. Tale operazione si ripete fino a che non venga inserita "s".



Questo tracciacurve universale (General Purpose Graph Plotter), è di particolare aiuto nelle riunioni di affari quando si rende necessaria una documentazione scritta degli andamenti. Presenta fino a 9 file di dati, contenenti ognuno i valori relativi ai 12 mesi dell'anno. I dati di ogni coppia di file possono essere tracciati sia tramite barre che tramite punti. La scelta dell'asse verticale è automatica e il massimo è dieci elevato a 1,2 o 5. Possono essere registrati (col programma) diversi insiemi di file con vari nomi.

Quando non visualizza un grafico, il programma presenta all'utente un menù con le scelte possibili:

- disegna un nuovo grafico da uno o due file e stampalo se richiesto;
- richiama i dati di ognuno dei nove file e cambiali;
- conferma o sostituisci il titolo per l'insieme dei nove file;
- registra il programma contenente i dati sul nastro usando il titolo come nome.

Una volta caricato da nastro, il programma si autoavvia. Qualora dovesse, per qualsiasi ragione, arrestarsi, fatelo ripartire dando un GO TO 1000 al posto di RUN, che cancellerebbe tutti i dati (naturalmente il RUN va dato almeno una volta, all'inizio, per inizializzare gli array dei dati). Per bloccare il programma dovete battere CAPS/SHIFT e BREAK mentre è in corso un calcolo oppure CAPS/SHIFT e 6 mentre è in attesa di un ingresso.

```
 100 DIM d(9,13): DIM n$(9,10):
DIM t$(10): DIM x$(13)
 110 LET l$="G F M A M G L A S O
 N D"
 120 LET m$="GenFebMarAprMagGiuL
ugAgoSetOttNovDic"
 130 GO SUB 9100
1000 REM PARTE PRINCIPALE
1010 PAPER 7: INK 0: CLS : PRINT
 TAB 10;t$
1020 FOR a=1 TO 9: PRINT AT a+1,
9;a;TAB 12;n$(a): NEXT a
1100 PRINT AT 15,6;"Inserisci:"
1110 PRINT AT 17,7;"P - plot";TA
B 7;"R - revisione dati"
1120 PRINT TAB 7;"S - registra";
TAB 7;"T - batti titolo"
1130 INPUT LINE i$: LET i$=FN a$
(i$)
1140 FOR a=15 TO 21: PRINT AT a,
0,: NEXT a
1150 IF i$="R" THEN GO TO 2000
1160 IF i$="P" THEN GO TO 3000
1170 IF i$="S" THEN GO TO 4000
1180 IF i$="T" THEN GO TO 5000
1190 GO TO 1100
2000 INPUT "File (1-9) ? "; LINE
 i$
```

```
2010 GO SUB 9000: IF i<1 OR i>9
THEN GO TO 2000
2020 LET f=i: PRINT AT f+1,9; OV
ER 1; FLASH 1;" "
2100 FOR m=1 TO 12
2110 PRINT AT FN c(),FN d();m$(m
*3-2);" ";d(f,m)
2120 NEXT m
2200 INPUT "Cambi (Y/N) ? "; LIN
E i$
2210 IF i$="n" OR i$="N" THEN GO
 TO 1000
2220 IF i$<>"y" AND i$<>"Y" THEN
 GO TO 2200
2230 INPUT "Nome del file ? "; L
INE n$(f): PRINT AT f+1,12;n$(f)
2300 FOR m=1 TO 12
2310 INPUT "Dati per ";(m$(m*3-2
 TO m*3));" ? "; LINE i$
2320 GO SUB 9000: IF e<>0 THEN G
O TO 2310
2330 LET d(f,m)=i: LET x$=STR$ i
2340 PRINT AT FN c(),2+FN d();x$
2350 NEXT m
2360 INPUT "Premi ENTER per torn
are al menu'"; LINE i$
2400 LET a=0
2410 FOR m=1 TO 12: IF d(f,m)>a
THEN LET a=d(f,m)
2420 NEXT m
2430 FOR b=-30 TO 30: IF 10↑b>=a
 THEN GO TO 2450
2440 NEXT b
2450 LET m=10↑b: IF m/2>=a THEN
LET m=m/2
2460 IF .2*10↑b>=a THEN LET m=.2
*10↑b
2470 LET d(f,13)=m
2480 GO TO 1000
3000 REM Plot
3010 INPUT "Linea o barra (L/B)
? "; LINE p$: LET p$=FN a$(p$)
3020 IF p$<>"L" AND p$<>"B" THEN
 GO TO 3010
3030 INPUT "Primo file (1-9) ? "
; LINE i$
3040 IF LEN i$<>1 OR i$<"0" OR i
$>"9" THEN GO TO 3030
3050 LET f1=CODE i$-CODE "0"
3060 PRINT AT 21,6; INK 2;CHR$ (
133+(11 AND p$="L")); INK 0;" ";
n$(f1)
3100 INPUT "Secondo file (1-9) ?
 "; LINE i$
3110 IF i$="" THEN LET f2=0: GO
TO 3200
3120 IF LEN i$<>1 OR i$<"0" OR i
$>"9" THEN GO TO 3100
3130 LET f2=CODE i$-CODE "0"
```

```
3140 PRINT AT 21,19; INK 1;CHR$
(133+(12 AND p$="L")); INK 0;" "
;n$(f2)
3200 FOR a=0 TO 20: PRINT AT a,0
;: NEXT a
3210 LET hi=d(f1,13)
3220 IF f2>0 THEN IF d(f2,13)>hi
 THEN LET hi=d(f2,13)
3300 PLOT 0,17: DRAW 255,0: PLOT
 0,167: DRAW 255,0
3310 PRINT AT 0,10;t$;AT 1,0;hi;
AT 19,0;0;AT 20,6;l$
3320 IF p$="B" THEN GO TO 3500
3400 LET s=FN b(d(f1,1)): PLOT 5
1,s
3410 FOR a=2 TO 12: LET y=FN b(d
(f1,a)): DRAW 16,y-s: LET s=y: N
EXT a
3420 IF f2=0 THEN GO TO 3900
3430 LET s=FN b(d(f2,1)): PLOT 5
1,s
3440 FOR a=2 TO 12: LET y=FN b(d
(f2,a))
3450 DRAW 4,(y-s)/4
3460 PLOT 27+16*a,s+(y-s)/2: DRA
W 4,(y-s)/4
3470 LET s=y: PLOT 35+16*a,s
3480 NEXT a
3490 GO TO 3900
3500 FOR a=1 TO 12
3510 FOR s=34+a*16 TO 37+a*16
3520 INK 2: PLOT s,17: DRAW 0,FN
 b(d(f1,a))-17
3530 IF f2>0 THEN INK 1: PLOT s+
6,17: DRAW 0,FN b(d(f2,a))-17
3540 NEXT s
3550 NEXT a
3560 INK 0
3900 INPUT "Disegno (Y) ? "; LIN
E i$
3910 IF i$="y" OR i$="Y" THEN CO
PY
3920 GO TO 1000
4000 REM registrazione
4010 SAVE t$ LINE 4020
4020 GO SUB 9100: GO TO 1000
5000 REM Input titolo
5010 INPUT "Titolo ? "; LINE t$
5020 GO TO 1000
8000 REM Funzioni
8010 DEF FN a$(a$)=CHR$ (CODE a$
-(32 AND CODE a$>96))
8020 DEF FN b(v)=INT (v*150/hi+1
7)
8030 DEF FN c()=13+m-6*INT ((m-1
)/6)
8040 DEF FN d()=16*INT ((m-1)/6)
9000 REM conversione di i$ a i
```

```
9010 LET i=0: LET e=1: IF i$=""
THEN RETURN
9020 LET dp=0
9030 FOR c=1 TO LEN i$: LET c$=i
$(c)
9040 IF c$="." THEN LET dp=dp+1:
 GO TO 9060
9050 IF c$<>"+" AND (c$<"0" OR c
$>"9") THEN RETURN
9060 NEXT c: IF dp>1 THEN RETURN

9070 LET i=VAL i$: LET e=0: RETU
RN
9100 REM definizione caratteri
9110 DATA 0,0,0,0,255,0,0,0,0,0,
0,0,204,0,0,0
9120 RESTORE : FOR a=0 TO 15: RE
AD b: POKE USR "a"+a,b: NEXT a
9130 RETURN
```

AGENDA

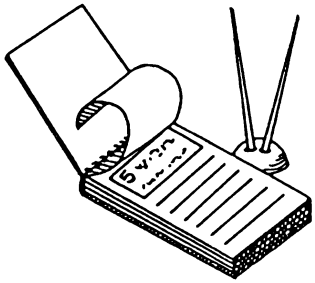
D'ora in poi non avrete più scuse se vi dimenticate qualche compleanno, oppure qualche appuntamento importante! Il programma che segue trasforma lo Spectrum in un'agenda da tavolo. Potete inserire messaggi di qualunque lunghezza con qualsiasi data, ve li ritroverete memorizzati in ordine cronologico. Il programma accetta anche due messaggi con la stessa data. È possibile controllare il contenuto esplorando fra due date, quindi, trovato quanto interessa, tornare al menù principale e, quando un'informazione non serve più, potete cancellarla per fare spazio in memoria a nuovi dati. Naturalmente potete salvare su nastro le informazioni della vostra agenda. Le spiegazioni sono contenute nel programma stesso. Se, per una ragione qualsiasi, si dovesse bloccare, riavviatelo con GO TO 130 e non col distruttivo RUN.

```
  10 REM Preparazione
  20 DEF FN f(x)=INT (x-100*INT
(x/100))
  30 DEF FN f$(x)=("0" AND FN f(
x)<10)+STR$ FN f(x)
  40 DEF FN g$(x$,x)=x$(x+4 TO x
+5)+"/"+x$(x+2 TO x+3)+"/"+x$(x
TO x+1)
  50 LET a$="": LET l$=""
  60 LET c=0
  70 DIM b$(5)
 100 REM Menu
 110 PRINT AT 20,0;"Premi qualsi
asi tasto per continuare",,,
 120 PAUSE 0
 130 CLS
 140 PRINT AT 2,10;" Diario   "
 150 PRINT AT 5,5;"(1)...Inseris
ci messaggio"
 160 PRINT AT 7,5;"(2)...Cancell
a giorno"
 170 PRINT AT 9,5;"(3)...Display
"
 180 PRINT AT 11,5;"(4)...Save '
Diario'"
 190 PRINT AT 21,0;"Inserisci la
 voce "
 200 INPUT n: LET n=INT n: IF n<
1 OR n>4 THEN GO TO 200
 210 CLS
 220 IF n=1 THEN GO TO 300
 230 GO TO 300*n+300
 300 REM input
 310 PRINT AT 0,12;"Batti il dia
rio"
 320 LET c=c+1
 330 LET m$="": LET a=1
 340 INPUT "Giorno   ";d'"Mese "
;m'"Anno   ";y
 350 LET d$=FN f$(y)+FN f$(m)+FN
 f$(d): LET e$=FN g$(d$,1)
 360 CLS : PRINT AT 0,0;e$;TAB 1
2;"Batti il diario"
 370 INPUT "Pragrafo ";(a),i$: P
RINT AT 2,2;i$
 380 LET l=LEN i$-32*INT (LEN i$
/32)
 390 FOR z=l TO 29: LET i$=i$+"
": NEXT z
 400 LET m$=m$+"  "+i$
 410 PRINT AT 21,0;"Ancora ? (y/
n)"
 420 IF INKEY$="y" THEN LET a=a+
1: GO TO 360
 430 IF INKEY$<>"n" THEN GO TO 4
10
 440 LET b$=STR$ (LEN a$+1): LET
 l$=l$+b$
```

```
 450 PRINT AT 2,0;m$: LET a$=a$+
m$
 460 LET b$=STR$ (LEN m$): LET l
$=l$+b$+d$
 470 FOR z=1 TO 16*c-16 STEP 16
 480 LET p=VAL l$(z TO z+4)
 490 LET r=(l$(z+10 TO z+15)=d$)
+(2 AND l$(z+10 TO z+15)>d$)
 500 IF r=1 THEN LET c=c-1: GO T
O 660
 510 IF r=2 THEN GO TO 540
 520 NEXT z
 530 GO TO 100
 540 LET v=VAL l$(16*c-15 TO 16*
c-11): LET p=VAL l$(z TO z+4)
 550 LET a$=a$( TO p-1)+a$(v TO
)+a$(p TO v-1)
 560 LET h$=l$(16*c-10 TO 16*c)
 570 FOR y=16*c-31 TO z STEP -16
 580 LET l$(y+21 TO y+31)=l$(y+5
 TO y+15)
 590 NEXT y
 600 LET l$(z+5 TO z+15)=h$
 610 FOR y=z TO c*16-16 STEP 16
 620 LET b$=STR$ (VAL l$(y TO y+
4)+VAL l$(y+5 TO y+9))
 630 LET l$(y+16 TO y+20)=b$
 640 NEXT y
 650 GO TO 100
 660 FOR y=z+16 TO 16*c STEP 16
 670 LET l$(y TO y+4)=STR$ (VAL
l$(y TO y+4)+LEN m$)
 680 NEXT y
 690 LET v=VAL l$(z TO z+4)+VAL
l$(z+5 TO z+9)
 700 LET a$=a$( TO v-1)+m$+a$(v
TO LEN a$-LEN m$)
 710 LET l$(z+5 TO z+9)=STR$ (VA
L l$(z+5 TO z+9)+LEN m$)
 720 LET l$=l$( TO LEN l$-16)
 730 GO TO 100
 900 REM Cancellazione
 910 PRINT AT 0,10;"Cancellazion
e"
 920 INPUT "Cosa cancelli ?"'"Gi
orno  ";d'"Mese ";m'"Anno  ";y
 930 LET d$=FN f$(y)+FN f$(m)+FN
 f$(d)
 940 LET e$=FN g$(d$,1)
 950 FOR z=1 TO 16*c STEP 16
 960 LET p=VAL l$(z TO z+4)
 970 IF l$(z+10 TO z+15)=d$ THEN
 GO TO 1010
 980 IF l$(z+10 TO z+15)<d$ THEN
 NEXT z
 990 PRINT AT 5,7;"Non valido";T
AB 12;e$
1000 GO TO 100
```

```
1010 LET a$=a$( TO p-1)+a$(VAL l
$(z+5 TO z+9)+p TO )
1020 LET v=VAL l$(z TO z+4)
1030 FOR y=z TO 16*c-20 STEP 16
1040 LET l$(y+5 TO y+15)=l$(y+21
 TO y+31)
1050 LET v=VAL l$(y TO y+4)+VAL
l$(y+5 TO y+9)
1060 LET l$(y+16 TO y+20)=STR$ v
1070 NEXT y
1080 LET l$=l$( TO 16*c-16)
1090 PRINT AT 5,10;"Batti : ";e$
;TAB 13;"e' cancellato"
1100 LET c=c-1
1110 GO TO 100
1200 REM Scrittura
1210 PRINT AT 0,9;"Display"
1220 INPUT "Da quando ?"'"Giorno
   ";d'"Mese ";m'"Anno  ";y
1230 LET d$=FN f$(y)+FN f$(m)+FN
 f$(d)
1240 INPUT "A quando ?"'"Giorno
  ";d'"Mese ";m'"Anno  ";y
1250 LET e$=FN f$(y)+FN f$(m)+FN
 f$(d)
1260 IF e$<d$ THEN LET i$=e$: LE
T e$=d$: LET d$=i$
1270 LET f=0
1280 FOR z=1 TO 16*c STEP 16
1290 LET p=VAL l$(z TO z+4)
1300 IF d$<=l$(z+10 TO z+15) THE
N GO TO 1340
1310 NEXT z
1320 LET f=1
1330 GO TO 1440
1340 IF e$<l$(z+10 TO z+15) THEN
 LET f=z: GO TO 1410
1350 CLS
1360 PRINT AT 0,0;FN g$(l$,z+10)
1370 PRINT AT 2,0;a$(p TO p-1+VA
L l$(z+5 TO z+9))
1380 PRINT AT 20,0;"Premi 'y' pe
r continuare"
1390 PRINT AT 21,0;"Premi 'n' pe
r la fine"
1400 IF INKEY$="n" THEN GO TO 13
0
1410 IF INKEY$<>"y" THEN GO TO 1
400
1420 NEXT z
1430 IF f<>1 THEN FOR z=1 TO 50:
 NEXT z: GO TO 100
1440 LET d$=FN g$(d$,1)
1450 LET e$=FN g$(e$,1)
1460 PRINT AT 5,7;"Non inserire
tra";TAB 12;d$;" &";TAB 12;e$
1470 GO TO 100
1500 REM Save
1510 SAVE "diario" LINE 1520
```

```
1520 PRINT AT 5,5;"Riavvolgi il
nastro per VERIFY";AT 7,5;"Premi
 qualsiasi tasto quando sei pron
to"
1530 PAUSE 0
1540 VERIFY "diario"
1550 GO TO 100
```



SOCI

Questo programma è nato per agevolare l'esistenza dei segretari di club, notoriamente molto indaffarati. Contiene i dati per un massimo di 370 soci (con uno Spectrum da 48 K), memorizzabili su nastro e vi permette di modificare o stampare le informazioni in diversi modi. I dati annotati per ciascun socio sono:

- numero di tessera, allocato in sequenza ogni volta che si aggiungano al file nuovi soci, partendo da un numero iniziale impostato alla creazione del file;
- nome, telefono ed indirizzo fino a un totale di 94 caratteri;
- due "chiavi" da 3 caratteri, usate per contenere informazioni come, per esempio, il livello di associazione e il mese in cui il socio deve pagare la sua quota.

Se il vostro club è veramente grande, potrete utilizzare una cassetta separata per memorizzare i dati di ciascun gruppo di 370 soci. È possibile far girare il programma anche su uno Spectrum da 16 K, ma dovrete limitare il numero di soci a 42 cambiando la prima istruzione della riga 100 in LET m=42. Se il programma dovesse fermarsi per un motivo

qualsiasi, non usate RUN per farlo ripartire, in quanto verrebbero distrutti tutti i file. Usate invece GO TO 1000.

Lo schermo presenterà questo aspetto normalmente:

```
Num      :
Nome     :
Tel      :
Ind      :
         :
         :
         :
         :
Tasto 1:
Tasto 2:
Num + - C N P S

Tasto 1:
Tasto 2:
Pieno    :
```

Il quadratino nero nell'ultima riga, è un cursore lampeggiante usato per tutto il programma, al fine di segnalare la richiesta di impostare ulteriori dati.

Il menù principale vi invita a impostare:

Numero: battendo il numero d'iscrizione, verrà visualizzata la situazione del socio al quale si riferisce il numero stesso.

+: verranno presentati i particolari del socio con il numero più elevato.
—: mostrerà i dati del socio precedente.
C: vi permette di cambiare i dati del socio al momento visualizzato.
N: per aggiungere un nuovo socio all'elenco.
P: per stampare i dati dei soci mediante una stampante.
S: per memorizzare il programma su nastro.

Impostando "C" o "N", scomparirà la riga principale del menù sul fondo dello schermo ed il cursore lampeggiante si porterà vicino alla parola "nome:" e voi dovrete inserire il nome del socio. Eseguita l'operazione, il cursore si porterà verso il basso per farvi battere il numero di telefono. Continuate a riempire le righe di dati fino alla "chiave 2:"; dopo di ciò riapparirà il menù principale. Se in una delle voci eccederete nell'impostare i dati, il programma ignorerà l'eccesso mostrando sullo schermo solo i caratteri compresi nello spazio disponibile.

Selezionando P (Print), il display assumerà l'aspetto

Chiave 1:
Chiave 2:
Completo:

e si porrà in attesa dei dati prima di iniziare la stampa. Per le righe "Chiave 1" e "Chiave 2" avete la scelta tra il non impostare nulla (semplicemente premendo il tasto ENTER), oppure una chiave di tre caratteri al massimo. Se impostate una chiave, verranno stampati solo i dati dei soci contrassegnati da una chiave uguale. In questo modo potrete, per esempio, stampare un elenco dei soci che devono pagare la quota in "AGOsto". La terza riga (Completo:) necessita di una risposta Y o N. Impostando Y, verranno stampati tutti i dati di ciascun membro per poter, ad esempio, stampare indirizzi su etichette. Battendo N, sarà stampato solo il numero del socio, le due chiavi e il nome. L'opzione S (Save) si presta per memorizzare su nastro l'ultima versione della lista. Il programma vi chiederà il nome del file, permettendovi di controllare i dati registrati. Il programma viene salvato assieme ai dati in modo da girare automaticamente una volta caricato.

```
 100 LET m=370: DIM d$(m,100)
 110 DIM z$(25): DIM x$(3): DIM
k$(1)
 120 INK 0: PAPER 7: FLASH 0: BR
IGHT 0: OVER 0: INVERSE 0: BORDE
R 7: CLS
 200 PRINT AT 8,4;"Nuovo file di
 dati"
 210 PRINT AT 15,0;"Numero del p
rimo socio ? "; FLASH 1;""
 220 GO SUB 9000: IF i=0 THEN BE
EP .1,20: GO TO 220
 230 LET first=i: LET next=i: LE
T curr=0
1000 REM stampa il menu' princip
ale
1010 CLS
1020 PRINT AT 3,0;"Num.   :";AT 5
,0;"Nome  :"
1030 PRINT AT 7,0;"Tel.   :";AT 9
,0;"Indir.:"
1040 PRINT AT 10,6;":";AT 11,6;"
:";AT 12,6;":";AT 13,6;":"
```

```
1050 PRINT AT 15,0;"Cod.1 :";AT
16,0;"Cod.2 :"
2000 REM scelta dal menu' princi
pale
2010 PRINT AT 21,0;"Numero + - C
 N P S "; FLASH 1;" "
2020 GO SUB 9000: PRINT AT 21,0,
,: IF i>0 THEN GO TO 3000
2030 IF CODE i$>90 THEN LET i$=C
HR$ (CODE i$-32)
2040 IF i$="+" THEN GO TO 3100
2050 IF i$="-" THEN GO TO 3200
2060 IF i$="C" THEN GO TO 4000
2070 IF i$="N" THEN GO TO 4100
2080 IF i$="P" THEN GO TO 5000
2090 IF i$="S" THEN GO TO 6000
3000 REM ricerca il record numer
ato
3010 IF i<first OR i>next THEN G
O TO 2000
3020 LET curr=i: GO TO 3300
3100 REM ricerca il record succe
ssivo
3110 IF curr>=next-1 THEN GO TO
2000
3120 LET curr=curr+1: GO TO 3300
3200 REM ricerca il record prece
dente
3210 IF curr<=first THEN GO TO 2
000
3220 LET curr=curr-1
3300 REM ricerca il record attua
le
3310 FOR a=3 TO 16: PRINT AT a,7
;z$: NEXT a
3320 PRINT AT 3,8;curr: LET pos=
7
3330 FOR l=5 TO 13: IF l=6 OR l=
8 THEN LET l=l+1
3340 PRINT AT l,8;
3350 IF pos>100 THEN GO TO 3500
3360 LET q=CODE d$(curr+1-first,
pos): LET pos=pos+1
3370 IF q<128 THEN PRINT CHR$ q;
: GO TO 3350
3380 PRINT CHR$ (q-128);
3390 NEXT l
3500 PRINT AT 15,8;d$(curr+1-fir
st,4 TO 3)
3510 PRINT AT 16,8;d$(curr+1-fir
st,4 TO 6)
3520 GO TO 2000
4000 REM cambio record attuale
4010 IF curr<first OR curr>=next
 THEN GO TO 2000
4020 GO TO 4200
4100 REM aggiunta nuovo record
```

```
4110 IF next>=first+m THEN PRINT
 AT 19,0; FLASH 1;"FILE PIENO":
BEEP 1,10: GO TO 2000
4120 LET curr=next: LET next=nex
t+1
4130 FOR a=3 TO 16: PRINT AT a,7
;z$: NEXT a
4200 REM input dati per i record
4210 PRINT AT 3,8;curr: LET pos=
7
4220 FOR l=5 TO 13: IF l=6 OR l=
8 THEN LET l=l+1
4230 PRINT AT l,7; FLASH 1;" "
4240 LET j$=z$: IF pos>100 THEN
GO TO 4300
4250 INPUT LINE i$: IF LEN i$>24
 THEN LET i$=i$( TO 24)
4260 IF i$="" THEN LET i$=" "
4270 LET k=LEN i$: IF pos+k>100
THEN LET i$=i$( TO 101-pos)
4280 LET k=LEN i$: LET j$(2 TO k
+1)=i$: LET i$(k)=CHR$ (CODE i$(
k)+128)
4290 LET d$(curr+1-first,pos TO
pos+k-1)=i$: LET pos=pos+k
4300 PRINT AT l,7;j$
4310 NEXT l
4320 PRINT AT 15,7; FLASH 1;" "
4330 INPUT LINE i$: LET d$(curr+
1-first, TO 3)=i$
4340 PRINT AT 15,7;" ";d$(curr+1
-first,4 TO 3)
4350 PRINT AT 16,7; FLASH 1;" "
4360 INPUT LINE i$: LET d$(curr+
1-first,4 TO 6)=i$
4370 PRINT AT 16,7;" ";d$(curr+1
-first,4 TO 6)
4380 GO TO 2000
5000 REM stampa
5010 CLS : PRINT AT 5,0;"Cod.1 :
";;"Cod.2 :";;"Pieno:"
5020 PRINT AT 5,7; FLASH 1;" "
5030 INPUT LINE i$: LET x$=i$: I
F i$<>"" THEN LET i$=x$
5040 PRINT AT 5,7;" ";i$;AT 7,7;
 FLASH 1;" "
5050 INPUT LINE j$: LET x$=j$: I
F j$<>"" THEN LET j$=x$
5060 PRINT AT 7,7;" ";j$;AT 9,7;
 FLASH 1;" "
5070 INPUT LINE k$: IF CODE k$>9
0 THEN LET k$=CHR$ (CODE k$-32)
5080 IF k$<>"S" AND k$<>"N" THEN
 GO TO 5070
5090 PRINT AT 9,7;" ";k$
5100 LPRINT : LPRINT
5110 FOR c=first TO next-1: LET
l$=d$(c-first+1)
```

```
5120 IF (i$<>"" AND i$<>l$( TO 3
)) OR (j$<>"" AND j$<>l$(4 TO 6)
) THEN GO TO 5230
5130 IF k$="S" THEN LPRINT
5140 LPRINT c;l$( TO 6);" ";
5150 LET p=7
5160 FOR l=1 TO 7: IF k$="N" AND
 l>1 THEN GO TO 5220
5170 IF p>100 THEN GO TO 5210
5180 LET q=CODE l$(p): LET p=p+1
5190 IF q<128 THEN LPRINT CHR$ q
;: GO TO 5180
5200 LPRINT CHR$ (q-128);
5210 LPRINT
5220 NEXT l
5230 NEXT c
5240 LPRINT
5250 GO TO 1000
6000 REM salva dati e programma
6010 INPUT "Nome file?"' LINE n$
: IF n$="" THEN GO TO 6010
6020 IF LEN n$>10 THEN LET n$=n$
( TO 10)
6030 LET i$="": SAVE n$ LINE 610
0
6040 CLS : PRINT AT 6,0;"VERIFIC
A I DATI SALVATI SUL NASTRO:"
6050 PRINT '"SE NON SONO ESATTI,
 IL PROGRAMMA SI FERMA CON ERR 9
"
6060 PRINT '"PUOI ALLORA FARLO R
IPARTIRE CON 'GOTO 1000'"
6100 INPUT "Verifica (S/N) ?"; L
INE i$
6110 IF i$="n" OR i$="N" THEN GO
 TO 1000
6120 IF i$<>"s" AND i$<>"S" THEN
 GO TO 6100
6140 PRINT AT 20,0;" Riavvolgi i
l nastro per verificare"
6150 LET i$="": VERIFY n$
6160 GO TO 1000
9000 LET i=0: INPUT LINE i$
9010 FOR a=1 TO LEN i$
9020 IF (i$(a)<"0" OR i$(a)>"9")
 AND i$(a)<>" " THEN GO TO 9040
9030 NEXT a
9040 IF a>1 THEN LET i=VAL i$( T
O a-1)
9050 RETURN
```

In programmi come questo, in cui si richiede la memorizzazione di dati dalla lunghezza variabile come nomi e indirizzi, sorge sempre il problema di scegliere tra l'uso di record dalla lunghezza fissa, comodi

per la loro veloce memorizzazione, ma troppo ingombranti in quanto vanno proporzionati tutti in funzione della lunghezza del dato più lungo, e l'uso di record la cui lunghezza varia di volta in volta.

Qui sono stati usati entrambi, cioé entro record di lunghezza fissa (100 byte) trovano posto dati di lunghezza variabile (come ad esempio il nome).

m — Numero massimo di record (370 per 48 K, 42 per 16 K)

d$(m,100) — Array usato per ospitare il file di dati. Ogni record a disposizione per ogni socio occupa 100 byte di cui i primi 6 memorizzano le due "chiavi" di 3 byte ciascuna. Il nome, il numero di telefono e le cinque linee relative all'indirizzo, occupano i 94 byte restanti. Alla fine di ogni carattere viene inserito un CHR$ 128 per ottenere un separatore di "fine linea".

first — Numero del primo socio della lista.

curr — Numero del socio richiamato.

next — Numero del socio successivo da aggiungere in elenco.

SISTEMI DI EQUAZIONI

Il programma che segue risolve simultaneamente qualsiasi sistema di equazioni lineari (N equazioni di N incognite) a patto che queste non siano interdipendenti o inconsistenti.

```
  10 REM EQUAZIONI SIMULTANEE
  20 LET m=0: LET l=0: LET j=1
  30 INPUT "Quante variabili hai
?";n
  40 DIM x(n): DIM a(n,n+1): DIM
b(n,n+1)
  50 DEF FN f(u,v)=a(u,v)/a(u,u)
  60 DEF FN g(u,v,w)=a(v,w)-a(u,
w)*a(v,u)
  70 DEF FN h(u,v)=x(u)-a(u,v)*x
(v)
 100 REM Input
 110 FOR c=1 TO n
 120 CLS : PRINT AT 0,0;"Equazio
ne ";c;TAB 18;n;" Equazioni";AT
```

```
20,0;"ENTER coefficienti"
 130 FOR d=1 TO n: INPUT "X";(d)
;" : ";i: LET a(c,d)=i: LET b(c,
d)=i: PRINT AT d+2,1;"X";d;" : "
;i: NEXT d
 140 INPUT "Costante (R.H.S.) :"
;i: PRINT AT d+2,1;"Cos = ";i: L
ET a(c,d)=i: LET b(c,d)=i
 150 INPUT "Correzione (y/n) ?";
TAB 0;c$: IF c$(1)="y" THEN GO T
O 130
 160 NEXT c
 170 CLS
 200 REM Esecuzione & calcolo
 210 FOR c=1 TO n
 215 LET p=a(c,c)
 220 FOR d=c+1 TO n: LET p=a(c,c
): IF ABS p>=ABS a(d,c) THEN GO
TO 240
 230 FOR e=1 TO n+1: LET s=a(c,e
): LET a(c,e)=a(d,e): LET a(d,e)
=s: NEXT e
 240 NEXT d
 250 FOR d=n+1 TO c STEP -1: IF
ABS p<1E-5 THEN GO TO 600
 260 LET a(c,d)=FN f(c,d)
 270 NEXT d
 280 FOR d=c+1 TO n
 290 FOR e=n+1 TO c STEP -1: LET
 a(d,e)=FN g(c,d,e): NEXT e
 300 NEXT d
 310 NEXT c
 400 REM Calcolo
 410 LET x(n)=a(n,n+1)
 420 FOR c=n-1 TO 1 STEP -1: LET
 x(c)=a(c,n+1)
 430 FOR d=n TO c+1 STEP -1: LET
 x(c)=FN h(c,d): NEXT d
 440 NEXT c
 500 REM Output
 510 FOR c=1 TO n
 520 FOR d=1 TO n-1: PRINT b(c,d
);"*X";d;" + ";: NEXT d
 530 PRINT b(c,n);"*X";d;" = ";b
(c,d+1);TAB 0;
 540 NEXT c
 550 IF l=1 THEN RETURN
 560 FOR c=1 TO n: PRINT AT 20-n
+c,1;"X";c;" =";TAB 5+(1 AND x(c
)>=0);x(c): NEXT c
 570 GO TO 710
 600 REM Dipendente o inconsiste
nte
 610 FOR k=c TO n
 620 FOR h=c TO n: LET a(k,h)=a(
k,h+1): NEXT h
 630 LET a(k,n)=0
 640 NEXT k
```

```
 650 IF m<>c THEN LET m=c: LET j
=m: GO TO 220
 660 LET j=j+1: IF j<n-c+1 THEN
GO TO 220
 670 LET l=1: GO SUB 500
 680 FOR k=m TO n: IF ABS a(k,n+
1)>1E-5 THEN PRINT AT 21,0;"INCO
NSISTENTE": GO TO 710
 690 NEXT k
 700 PRINT AT 21,0;"DIPENDENTE"
 710 STOP
```

P	Elemento pivot
a(x,y)	Array di lavoro
b(x,y)	Coefficiente originale di y(x) nell'equazione x
x(c)	Variabile x(c) parte della soluzione
FN f(u,v)	Aiuta a dividere per il pivot (equazione pivot solamente)
FN g(u,v,w)	Aiuta ad eliminare le variabili
FN h(u,v)	Aiuta a calcolare le variabili.

```
1*X1 + 2*X2 + 3*X3 + 4*X4 = 10          X1 = 1
4*X1 + 3*X2 + 2*X3 + 1*X4 = 10          X2 = 1
0*X1 + 0*X2 + 1*X3 + 1*X4 = 2           X3 = 1
1*X1 + 1*X2 + 1*X3 + 0*X4 = 3           X4 = 1
```

BANCA

```
          Situazione        2/12/84
          Bilancio Gennaio

(1)       Mario Rossi
          Assegno                £78.00
     Entrate 2 a 4 non mostrate

     Bilancio totale             £0.00
     ----------------------------
```

D'ora in poi potrete amministrare le vostre finanze nel modo più adeguato!

Questo programma dispone di un file che può mettere a disposizione fino a 99 entrate ognuna delle quali contenenti una descrizione, la data, il tipo di transazione e l'ammontare della cifra. Il tutto è organizzato attorno ad un menù principale che vi permette le seguenti scelte:

- ottenere una vista singola o panoramica delle entrate. Entrambe le funzioni possono essere sia visualizzate sullo schermo che stampate su carta;
- inserire un Credito (Riscossione) o un Debito (Pagamento);
- fare una correzione;
- registrare il programma. Lo fa in modo che questo si autolanci al caricamento. Il nome é il titolo usato all'inizio.

Quando il data file è pieno, si rinnova automaticamente, cancellando i primi 89 ingressi e spostando i rimanenti 10 a formare l'inizio del nuovo file. State quindi attenti quando vi restano poche righe a disposizione! Se per qualche ragione il programma si bloccasse, fatelo ripartire con GO TO 100 e non con RUN per la solita ragione. Il programma fa un uso insolito di PRINT per dirigere i dati sullo schermo (PRINT #2;) o alla stampante (#3;).

```
  10 DIM c$(100,12): DIM d$(100,
8): DIM n$(100,12)
  20 DIM m(100): DIM b$(32)
  30 LET n=0: LET i=1: LET t=0:
LET o=2: LET j=1: LET i$=""
  40 DEF FN f$(x)=STR$ (x-100*IN
T (x/100))
  50 CLS : INPUT "titolo",a$
  60 GO TO 130
 100 REM Menu'
 110 PRINT AT 21,0;"Qualsiasi ta
sto per continuare"
 120 PAUSE 0
 130 CLS : PRINT TAB 9;a$''
 140 PRINT AT 3,5;"(1)...Situazi
one"
 150 PRINT AT 5,5;"(2)...Credito
 (Ricezione)"
 160 PRINT AT 7,5;"(3)...Debito
(Pagamento)"
 170 PRINT AT 9,5;"(4)...Bilanci
o"
 180 PRINT AT 11,5;"(5)...Correz
ione"
 190 PRINT AT 13,5;"(6)...Save "
;a$
 200 IF i>96 THEN PRINT AT 19,4;
100-i;" Entrate lasciate prima"
' FLASH 1; INK 2;"rinnovo automa
tico a file pieno"
 210 INPUT TAB 5;"Scelta operazi
one  ";k: IF k<1 OR k>6 THEN GO
TO 210
 220 PRINT AT 19,0;b$''b$
 230 GO SUB (k*200+100)
 240 GO TO 100
 300 REM Situazione
 310 PRINT AT 3,4;"*"
 320 INPUT "Screen (1) o Paper (
2) ? ";o: IF o<>1 AND o<>2 THEN
GO TO 320
 330 INPUT "Entrate da ";e;" a "
;f: IF f>=i THEN LET f=i-1
 340 IF f<1 THEN LET f=1
 350 IF e<1 THEN LET e=1
 360 IF e>f THEN GO TO 330
 370 CLS : LET o=o+1: PRINT #o;T
AB 9;"Situazione";TAB 24;d$(i)''
TAB 9;a$''
 380 FOR g=e+1 TO f+1
 390 PRINT #o''"  (";g-1;")";TAB
 9;n$(g);TAB 9;c$(g);
 400 LET po=m(g)-m(g-1): LET x=2
9: GO SUB 1010
 410 IF po<0 THEN PRINT #o;"-";
 420 NEXT g
 430 IF f<i-1 THEN PRINT #o''TAB
 5;"Entrate ";f+1;" a ";i-1;" no
n mostrate"
```

```
 440 LET x=29: LET g=i: LET po=m
(g)
 450 PRINT #o;''TAB 5;"Bilancio
totale";
 460 GO SUB 1000
 470 PRINT #o;TAB 5;"----------
----------"''
 480 LET o=2
 490 RETURN
 500 REM Credito
 510 LET s=1
 520 GO TO 750
 600 REM Rinnovo
 610 FOR c=2 TO 11
 620 LET c$(c)=c$(c+89): LET m(c
)=m(c+89)
 630 LET n$(c)=n$(c+89): LET d$(
c)=d$(c+89)
 640 NEXT c
 650 FOR c=12 TO 100
 660 LET c$(c)=b$: LET m(c)=0: L
ET n$(c)=b$: LET d$(c)=b$
 670 NEXT c
 680 LET i=12
 690 RETURN
 700 REM Debito
 710 LET s=-1
 720 GO TO 750
 730 REM Credito & Debito
 740 PRINT AT k*2+1,4;"*";AT 15,
0;
 750 LET i=i+1: IF i=101 THEN GO
 SUB 600
 760 INPUT "data : giorno ";d;"
mese ";m;" anno ";y
 770 LET d$(i)=FN f$(d)+"/"+FN f
$(m)+"/"+FN f$(y): PRINT AT 15,2
;d$(i);
 780 INPUT "Ammontare ";a
 790 LET a=ABS a
 800 LET m(i)=.01*INT (100*(a*s+
m(i-1)+1E-3))
 810 LET x=16: LET po=m(i)-m(i-1
): LET g=i
 820 GO SUB 1010
 830 PRINT TAB 20;(" Credito" A
ND s>0);(" Debito" AND s<0)
 840 INPUT "Pagante";((CHR$ 8+"e
") AND s<0),n$(i): PRINT 'TAB 9;
n$(i)
 850 INPUT "Transazione ";c$(i)
: PRINT 'TAB 9;c$(i): IF n=5 THE
N RETURN
 860 PRINT '"Corretto (y/n) ?"
 870 IF INKEY$<>"y" AND INKEY$<>
"n" THEN GO TO 870
 880 IF INKEY$="n" THEN LET q=i-
1: PRINT AT 21,0;b$: GO SUB 1120
 890 RETURN
```

```
 900 REM Bilancio
 910 LET po=ABS m(i): LET x=13
 920 PRINT AT 16,8;"Bilancio",b$
,b$,b$;AT 20,0;
 930 GO SUB 1010
 940 IF m(i)<0 THEN PRINT INK 2;
 FLASH 1;AT 21,8;"CONTO IN ROSSO
"
 950 RETURN
1000 IF m(g)<0 THEN PRINT #o;TAB
 x-LEN STR$ INT ABS po-3;"-";
1010 LET p$=STR$ INT (100*(ABS p
o-INT (ABS po+.001))+.1)
1020 IF LEN p$<2 THEN LET p$="0"
+p$
1030 PRINT #o;TAB x-LEN STR$ INT
 (.1+ABS po)-2;"£";INT (.001+ABS
 po);".";p$;
1040 RETURN
1100 REM Correzione
1110 PRINT AT 11,4;"*": INPUT "Q
uale entrata ? ";q: IF q<=0 OR q
>=i THEN GO TO 1110
1120 PRINT AT 11,4;"*": INPUT "d
-cancella a-altera  ";q$
1130 IF q$<>"d" AND q$<>"a" THEN
 GO TO 1120
1140 LET n=5: LET q=INT (q+.8):
IF q$="d" THEN GO TO 1230
1150 INPUT "Credito (1),Debito (
2) ";k: IF k<>1 AND k<>2 THEN GO
 TO 1150
1160 LET ii=i: LET k=k+1: LET i=
q: LET p=m(q+1)
1170 GO SUB k*200+100
1180 FOR d=q+2 TO ii+(1 AND ii<=
99)
1190 LET w=m(d)+m(q+1)-p: LET m(
d)=.01*SGN w*INT (ABS w*100+.1)
1200 NEXT d
1210 LET n=0: LET i=ii
1220 RETURN
1230 LET p=m(q+1)
1240 FOR d=q+1 TO i
1250 LET m(d)=.01*INT ((m(d+1)-p
+m(q))*100+.1)
1260 LET n$(d)=n$(d+1): LET c$(d
)=c$(d+1): LET d$(d)=d$(d+1)
1270 NEXT d
1280 LET n=0: LET m(i)=0: LET i=
i-1
1290 RETURN
1300 REM Save
1310 LET z$=(a$ AND LEN a$<=10):
 IF LEN a$>10 THEN LET z$=a$( TO
 10)
1320 SAVE z$ LINE 1400
1330 PRINT AT 17,0;a$'"Registrat
o a$"'z$
```

```
1340 PRINT AT 20,0;"Fa partire i
l nastro per VERIFY"'"poi premi
un tasto qualsiasi"
1350 PAUSE 0
1360 VERIFY z$
1370 RETURN
1400 CLS : GO SUB 1340: GO TO 10
0
```

CAPITOLO 14

PROGRAMMI DI UTILITÀ, STRANEZZE E ROUTINE RICORRENTI

AVVISO PERMANENTE

Se desiderate che nella prima riga dei vostri programmi appaia una istruzione REM contenente il vostro nome o qualche altro avviso, tipo copyrigt, impostatela ed eseguite:

```
POKE 1+PEEK 23635+256*PEEK 23636,0
```

Questo accorgimento darà alla linea il numero 0 in modo da non poterla più cancellare o cambiare. Con un po' di pratica potrete facilmente cambiare il numero di linea in altri modi più consoni, effettuando il

POKE di qualsiasi altro valore, come ad esempio "1". Una protezione più raffinata per il vostro messaggio di avviso (che vi permetta di proteggere anche più di una riga) la otterrete nel modo seguente:

— battete, come prima cosa, la riga o le righe da proteggere;

— eseguite

```
POKE PEEK 23635+256*PEEK 23636,40
```

(potete anche effettuare un POKE per qualsiasi altro valore compreso tra 40 e 63). L'accorgimento modifica il numero della prima linea in una lettera o in un qualsiasi altro simbolo seguito da tre numeri;

— impostate il resto del programma.

All'esecuzione del POKE, le linee interessate si trasferiranno alla fine del programma, da dove è assai complicato smuoverle o cancellarle, in quanto l'indirizzo a cui dovrebbe essere eseguito il nuovo POKE, andrebbe fatto ad una locazione di memoria superiore a quelle usuali. Il "non plus ultra" si ottiene adottando la procedura appena descritta ed in più effettuando un POKE maggiorato di 64. In tal modo la linea (o le linee) scomparirà e, pur restando memorizzata nell'area di programma, non verrà listata.

CONTROLLO RAM

Avete mai avuto la sensazione che la vostra RAM o parte di essa fosse in procinto di andare fuori uso?

Se sì, questo programma vi potrà, in futuro, aiutare. Esso esegue il controllo completo di tutta la RAM del sistema, eccetto che per le aree riservate al programma, al display e agli attributi. Effettua i POKE di una sequenza di numeri interi casuali quindi dei PEEK, allo scopo di verificare la loro presenza. Eseguita la passata di tutta l'area interessata, sullo schermo verrà presentato un "OK" se tutto è in ordine; viceversa

apparirà "Guasto a", seguito dall'indirizzo della locazione guasta. La sequenza di controllo si ripeterà finché non la fermerete, o finché non si esaurisca lo spazio sullo schermo. Al RUN il programma vi chiede l'indirizzo di partenza, che dovrà essere maggiore o uguale a 24500, poi quello di arrivo, minore o uguale a 65536 (32768 per la versione 16 K). Il controllo avviene naturalmente tra i due indirizzi. Il programma riportato è stato steso per Spectrum privi di Microdrive, in caso contrario dovrete elevare l'indirizzo più basso di partenza (24500), per far posto all'area RAM necessaria al Microdrive stesso. Oltre al controllo dell'hardware, un tale programma può servire, ad esempio, per verificare se il vostro computer è soggetto ad interferenze elettriche, interessando solo una piccola area di RAM (circa 200 byte) per risparmiare tempo.

```
  100 CLEAR 24500: LET s=24500: R
ANDOMIZE : LET r=INT (1000*RND)
  110 PRINT "Dall'indirizzo; ";
  120 INPUT a: IF a<s OR a>65535
THEN GO TO 120
  130 PRINT a;TAB 23; "A ;";
  140 INPUT b: IF b<=a OR b>65535
 THEN GO TO 140
  150 PRINT b'
  200 RANDOMIZE r
  210 FOR c=a TO b: POKE c,INT (2
56*RND): NEXT c
  220 RANDOMIZE r: LET e=0
  230 FOR c=a TO b: IF PEEK c<>IN
T (256*RND) THEN PRINT '"Guasto
a ";c: LET e=e+1
  240 NEXT c
  250 IF e>0 THEN PRINT 'e;" erro
re a questo passo"'
  260 IF e=0 THEN PRINT " ok ";
  270 LET r=INT (1000*RND)
  280 GO TO 200
```

100 Abbassa la RAMTOP in modo che l'area RAM al di sopra di 24500 si libera, quindi estrae un punto di partenza casuale.

110-150 Riceve dall'operatore gli indirizzi di partenza e di arrivo.

200 Predispone il numero di partenza della sequenza dei numeri casuali.

210 Riempie la RAM da controllare con interi casuali.

220 Resetta il punto di partenza della sequenza random.

230-240 Controlla che tutte le locazioni provate contengano i valori originali.

250-260 Scrive il risultato del test.

270-280 Estrae un nuovo numero di partenza per la prossima sequenza casuale quindi torna per un altro test.

Noterete che il programma è abbastanza lento; infatti impiega circa un minuto per controllare 1 Kbyte di RAM.

CONTROLLO ROM

Il programma, che gira in circa tre minuti, testa il contenuto della ROM del vostro Spectrum, sommando i contenuti di ogni byte fino ad ottenere il risultato prestabilito. La prova, effettuata su due Spectrum perfettamente funzionanti, ha portato al risultato 1926175. Qualora riscontraste valori diversi, i casi sono due: o la vostra ROM è guasta, o il vostro Spectrum monta un tipo di ROM più recente.

Nel secondo caso, potete controllare il risultato con quello ottenuto da uno dei vostri amici in possesso di uno Spectrum di tipo uguale al vostro. Come già detto, il programma che segue gira solo se allo Spectrum non sono collegate periferiche del tipo Microdrive.

```
  100 LET a=0
  110 FOR b=0 TO 16383: LET a=a+P
EEK b: NEXT b
  120 PRINT a
```

CATALOGO

Come saprete, il sistema più semplice per sapere quanto sia stato registrato su di un nastro è quello di eseguire il comando diretto:

```
CLS: VERIFY "?"
```

(Presumendo che non esista un file col titolo "?").

Per interrompere date il BREAK e per stampare, il COPY.

Eseguendo tale esperimento sul nastro allegato a questo libro, si ottiene il risultato che segue

```
Program: sideb
Bytes: logo
Program: wall
Bytes: wallg
Bytes: c
Program: bubblesort
Bytes: mcode
Bytes: char
Program: evolution
Bytes: mcode
Bytes: bits
Program: life
Bytes: p
Bytes: p
Program: draw
Bytes: c
Program: montecarlo
Bytes: p
Program: character
Bytes:
Program: waves
Bytes: m
```

RENUMBER

Questo programma rinumera parte o tutte le linee di qualsiasi programma BASIC, eccetto la propria. Può essere registrato su nastro col nome "renumber", quindi, in caso di bisogno, dovete caricare il programma da rinumerare e poi eseguire:

```
MERGE "renumber"
```

Caricato il programma di renumber, date

```
GO TO 9990
```

Il programma vi chiederà per prima cosa il numero di partenza e quello di arrivo della parte da rinumerare; se volete rinumerare l'intero programma, inserite come valore di partenza 0 e come valore di arrivo 9989. Vi chiederà poi i nuovi numeri di start e di end nonché lo step che stabilirà la spaziatura tra una linea e la successiva della nuova versione. Tenete presente che il programma così trasformato non tiene conto di eventuali linee superiori a 9989. Da notare anche che il programma non altera i numeri successivi alle funzioni GO TO, GO SUB e RESTORE, per cui bisognerà provvedere manualmente.

```
9990 REM Renumber
9991 INPUT "Vecchi numeri; parte
nza ";rs;TAB 11;"fine ";re''"Nuov
i numeri;partenza";rn;TAB 10;"pa
sso ";ri
9992 LET rp=PEEK 23635+256*PEEK
23636
9993 LET rv=PEEK 23627+256*PEEK
23628
9994 LET rl=256*PEEK rp+PEEK (rp
+1)
9995 IF rp>=rv OR rl>re THEN STO
P
9996 IF rl>=rs THEN POKE rp,INT
(rn/256): POKE rp+1,rn-256*INT (
rn/256): LET rn=rn+ri
9997 LET rp=rp+PEEK (rp+2)+256*P
EEK (rp+3)+4: GO TO 9994
```

9991 Accetta dall'operatore i numeri di linea degli estremi dell'intervallo da rinumerare.

9992,9993 Dispone "rp" all'indirizzo di partenza del programma nell'area RAM, e "rv" all'indirizzo finale.

9994-9997 Loop principale eseguito per ogni linea di programma.

9994 Assegna ad "rl" il numero della vecchia linea.

9995 Termina se si raggiunge la fine del programma o se si supera il limite superiore impostato.

9996 Cambia il numero di linea se il vecchio rientra nel range stabilito.

9997 Muove il pointer "rp" all'indirizzo di partenza della linea successiva, quindi torna alla linea 9994.

DA BASE A BASE

È un utilissimo programma che converte un numero scritto con una certa base, in un numero avente una base diversa. La massima base che si può manipolare è 36. Le cifre maggiori di 9 sono rappresentate da lettere dell'alfabeto (sia maiuscole che minuscole, in quanto il programma non fa distinzioni), quindi 12 cifre di un sistema a base 12 saranno:

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

Le linee 20-30 stabiliscono che il simbolo grafico CHR$ 155 sia il simbolo di equivalenza

```
Da base : 2      A base : 10
11001001    ≡    201
```

```
  10 REM conversioni tra basi di
verse
  20 FOR a=0 TO 7: READ d: POKE
USR CHR$ 155+a,d: NEXT a
  30 DATA 0,126,0,126,0,126,0,0
 100 INPUT "Da quale base ?";a;'
"A quale base :?";b
 110 INPUT "Da base :";a,"A base
:";b
 120 INPUT "Che numero vuoi conv
ertire ?"' LINE a$
 200 REM da base a a base 10
 210 LET s=0
 220 FOR c=LEN a$ TO 1 STEP -1
 230 LET d=CODE a$(c)-48-(7 AND
a$(c)>="A")-(32 AND a$(c)>="a")
 240 LET s=s+d*a↑(LEN a$-c)
 250 NEXT c
 300 REM da base 10 a base b
 310 LET b$="": PRINT
 320 LET r=INT (s-b*INT (s/b)+.5
): LET s=INT (s/b)
 330 LET b$=CHR$ (r+48+(7 AND r>
9))+b$
 340 IF s<>0 THEN GO TO 320
 350 PRINT a$;TAB 12;CHR$ 155;TA
B 16;b$;TAB 9
 370 GO TO 100
```

ROSONI

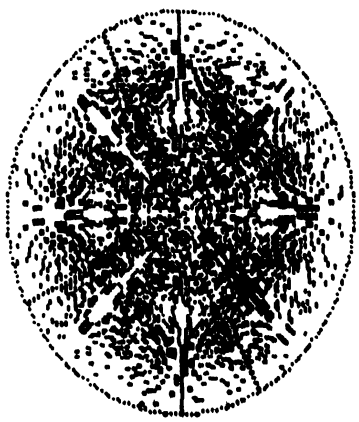
Lo statement DRAW, durante il calcolo delle coordinate di parecchie centinaia di punti componenti un arco, disegna una parte di cerchio raccordando i vari punti con delle linee rette. Di solito, in tutto questo non vi è nulla di speciale; però, dando un valore abbastanza alto alla terza grandezza che segue DRAW, ne scaturisce un risultato assai spettacolare. Fate un esperimento con la routine

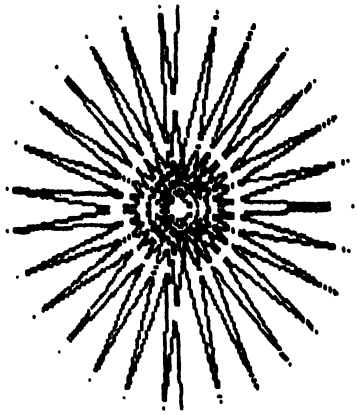
```
10 INPUT a: PRINT a
20 PLOT 127,10: DRAW 0,150,a*PI
```

cambiando magari la linea 20 in

```
20 PLOT 127,10: DRAW OVER 1;0,150,a*PI
```

Altri interessanti rosoni si genereranno inserendo i valori 119, 253, 261, 383 e 99999.





–65536 NON C'È

La ROM in dotazione allo Spectrum presenta un piccolo neo. Essa confonde i valori –65536 e –1E-38. Ad esempio

```
PRINT -65534-2                    -1E-38
```

per cui falsa qualsiasi combinazione di numeri interi negativi la cui somma sia –65536. Ciò porta ad altri effetti come

```
PRINT INT -65536                  -1
```

Se siete curiosi provate a battere la seguente routine e valutatene il risultato

```
FOR a=-65530 TO -65540 STEP -1: PRINT a: NEXT a
```

SCREEN$

Un altro difetto della ROM riguarda la funzione SCREEN$, la quale rende il carattere presente alla linea y, colonna x del display. Battete la routine che segue

```
 10 PRINT AT 0,0,"a"
 20 PRINT SCREEN$ (0,0)
 30 PRINT "b"+SCREEN$ (0,0)
 40 PRINT CODE SCREEN$ (0,0)
 50 PRINT 2+CODE SCREEN$ (0,0)
 60 IF CODE "a"=CODE SCREEN$ (0
,0) THEN PRINT "ok"
```

la quale dovrebbe portare a

```
a
a
ba
97
99
OK
```

mentre invece presenta

```
a
a
aa
97
97
```

Il problema esiste perché lo Spectrum deve prima valutare un'altra parte dell'espressione, per poi eseguire SCREEN$. Se la linea 50 viene sostituita da

```
PRINT CODE SCREEN$ (0,0)+2
```

il risultato è esatto. Per cui è necessario assegnare il valore reso da SCREEN$ ad una variabile prima dell'esecuzione.

Esempio

```
LET x$=SCREEN$( )
```

STRANI STR$

Esiste un problema anche per quanto riguarda la funzione STR$, ma solo se si verificano particolari condizioni.

In generale, l'espressione

```
a$+STR$ b
```

"dimentica" il valore di a$ se "b" è compreso tra −0.999999 e +0.999999 (salvo che non sia 0). Potete rendervene conto inserendo

```
PRINT "A"+STR$ (0.1)+"B"
```

che dovrebbe portare ad un risultato del genere

```
A0.1B
```

ma che invece dà:

```
0.1B
```

Per cui è meglio evitare di usare STR$ dopo il segno "+" in una espressione di stringa, a meno che non siate certi che il valore attribuitogli non sia una frazione inferiore a 1.

APPENDICE 1
LOCAZIONI DI PEEK E POKE

Per la maggior parte dei programmi BASIC non è necessario sapere quello che sta facendo il microprocessore Z80 dello Spectrum. Potete usare, ad esempio, una variabile senza preoccuparvi di come e dove questa viene memorizzata. Userete correntemente l'istruzione PRINT, pur senza sapere esattamente quali locazioni del Display File e dell'area degli Attributi vengono interessate. Tutti questi particolari decisamente noiosi potrebbero essere lasciati allo Z80 e al programma interprete BASIC dello Spectrum insito nella ROM. Senonché, di tanto in tanto, è opportuno sapere in dettaglio quanto succede nella macchina, per poter esercitare un controllo più diretto sugli avvenimenti di quello consentito dal BASIC. PEEK e POKE vi forniscono la chiave per scoprire il contenuto delle singole locazioni di memoria, e perfino di modificarlo. In pratica, non si può fare troppo affidamento su tale sistema in quanto i particolari delle routine operative dello Spectrum sono, in buona parte, protetti in ROM, quindi inaccessibili. Vi sono però, a vostra disposizione alcuni artifizi dei quali potete far tesoro; li elenchiamo quindi in questa Appendice.

Tratteremo, in generale, il contenuto dei singoli byte, che può essere un numero decimale da 0 a 255; verificheremo anche il contenuto di coppie di locazioni usate per memorizzare numeri binari di 16 bit (2 byte). Poiché lo Spectrum pone al primo posto il byte meno significativo, per leggere il contenuto combinato delle locazioni n e n+1, bisognerà agire nel modo seguente:

```
LET v=PEEK n+256*PEEK (n+1)
```

per inserire un nuovo valore

```
POKE n,v-256*INT (v/256): POKE n+1,INT (v/256)
```

Variabili di Sistema

```
23552 - 23559    8 bytes   KSTATE
```

Sono usate dalle routine ROM dello Spectrum per la lettura della tastiera. Provate ad usare la routine

```
10 FOR a=0 TO 7: LET b=a+23552
20 PRINT AT a,0; b;" ";PEEK b;
30 NEXT a
40 GO TO 10
```

per vedere cosa succede.

Gli 8 byte sono suddivisi in due gruppi di 4, per permettere la pressione di un secondo tasto prima che il primo venga rilasciato. Le locazioni da 23556 a 23559 riguardano la metà principale: 23556 conterrà 255, se non viene premuto nessun tasto; oppure se l'altro gruppo di 4 locazioni contiene informazioni riguardanti il tasto in quel momento azionato.

Premendo un tasto, la locazione 23556 accetta il codice relativo a quello stesso tasto senza tener conto della pressione dei tasti CAPS o SYMBOL SHIFT. La locazione 23559 memorizza il codice dell'ultimo tasto premuto, che ha cosi modificato questo gruppo di 4 byte. In questo caso, il codice può essere funzione dei tasti CAPS e SYMBOL SHIFT.

La locazione 23558 funge da temporizzatore a scalare, partendo dal valore contenuto in REPDEL (Variabile di Sistema contenuta in 23561) fino a giungere a 0 per poi ripartire dal valore contenuto in REPPER (Variabile di Sistema contenuta in 23562), per poter fornire la funzione di ripetizione automatica.

Anche la locazione 23557 è un temporizzatore utilizzato per la routine dell'antirimbalzo dei contatti dei tasti.

```
23560              1 byte    LASTK
```

Contiene il codice dell'ultimo tasto premuto. Provate

```
10 PAUSE 0: PRINT PEEK 23560: GO TO 10
```

Al pari di INKEY$, anche questa istruzione non distingue tra le parole chiave Extended Mode (quelle scritte in colore verde sopra i tasti e

quelle scritte in rosso sotto), oppure la grafica o la scrittura in lettere maiuscole (tramite CAPS SHIFT), dagli altri modi possibili della tastiera. Ecco i valori:

```
4   se premete CAPS SHIFT e TRUE VIDEO
5   "      "      "     "   " INV VIDEO
6   "      "      "     "   " CAPS LOCK
7   "      "      "     "   " EDIT
8   "      "      "     "   " <    (tasto 5)
9   "      "      "     "   "      (tasto 8)
10  "      "      "     "   "      (tasto 6)
11  "      "      "     "   "      (tasto 7)
12  "      "      "     "   " DELETE
13  "      "      "     "   " ENTER
14  "      "      "     "   " SYMBOL SHIFT
15  "      "      "     "   " GRAPHICS
```

A differenza di INKEY$, non rileva quanto avviene sulla tastiera, ma segnala solamente quale tasto è stato premuto per ultimo.

23561 **1 byte** **REPDEL**

Contiene il tempo di durata della pressione del tasto, prima che si verifichi la ripetizione automatica. Tale tempo è espresso in 50esimi di secondo (negli USA, in 60esimi). All'accensione del computer o all'istruzione NEW la locazione contiene il numero 35. POKando il numero "1", otterrete il tempo minimo, POKando lo "0" avrete un ritardo di circa 5 secondi.

23562 **1 byte** **REPPER**

Contiene il ritardo in 50esimi (o 60esimi) di secondo tra due ripetizioni dell'effetto di un tasto quando questo venga tenuto premuto in continuazione. All'accensione o dopo NEW tale locazione contiene un "5", ma potete variare tale valore in "1" (ritardo minimo) o anche in "0" o "255", che daranno un ritardo talmente lungo da ritenere disattivata la funzione di autorepeat.

POKE 23511,1: POKE 23562,1 renderà la tastiera praticamente inutilizzabile!

23563,4 **2 bytes** **DEFADD**

Queste locazioni contengono un numero maggiore di uno rispetto a quello dell'indirizzo RAM relativo alla cifra di sinistra, compresa tra pa-

rentesi, nella funzione DEF FN. Quando tale funzione non viene valutata, il contenuto vale 0. Esempio

```
  10 DEF FN a(x)=PEEK 23563+256*
PEEK 23564
  20 PRINT FN a(0),CHR$ PEEK FN
a(0)
  30 PRINT PEEK 23563+256*PEEK 2
3564
```

stamperà qualcosa come: 23762 x
0

23606,7 **2 bytes** **CHARS**

Contiene il puntatore alla serie di punti dei caratteri da 32 (spazio) a 127 (copyright). L'indirizzo iniziale della serie di punti, relativa ad un determinato carattere, è dato da

```
PEEK 23606+256*PEEK 23607
```

L'indirizzo contenuto in CHARS viene normalmente predisposto a 15360, mentre l'indirizzo della prima serie di punti nella ROM è 15360+32*8=15616

Potrete inserire nella RAM serie di punti per formare caratteri completamente nuovi, per usare i quali sarà necessario eseguire POKE 23606/7.

La funzione SCREEN$ riconosce soltanto caratteri compresi tra "spazio" e "c", cioè quelli della tabella delle sezioni di punti contenute nella ROM (osservate che i codici dei caratteri grafici "espansi" da 128 a 143 non appaiono nella tabella). Se desiderate riconoscere i caratteri grafici definiti dall'utente, dovrete ridefinire temporaneamente CHARS:

```
POKE 23606,PEEK 23675: POKE 23607,PEEK 23676-1
LET c=CODE SCREEN$ (y,x)+112
POKE 23606,0: POKE 23607,60
```

ciò renderà "c" uguale al codice del carattere grafico definito dall'utente, in corrispondenza dei punti x e y sullo schermo.

23608 **1 byte** **RASP**

Contiene la durata del suono del cicalino in 50esimi (o 60esimi) di secondo. All'accensione il suo valore è 64 e non viene influenzato da NEW.

23609 **1 byte** **PIP**

Contiene la lunghezza del "click" che si ode quando si preme un tasto. Vale 0 all'accensione e non viene influenzato da NEW.

Potrete ottenere un suono più avvertibile POKando un "5", mentre inserendo valori maggiori (ad esempio 20) verrà prodotto un suono tipo "beep".

23610 **1 byte** **ERR.N**

In questa locazione è contenuto il codice di errore diminuito di uno. Il valore solito è —1, letto, tramite PEEK, come 255.

POKando qualsiasi altro valore, apparirà il relativo codice di errore col messaggio visualizzato sullo schermo all'arresto del programma.

23618,9 **2 bytes** **NEWPPC**

Una istruzione GO TO o GO SUB inserisce in questa locazione il numero della linea di destinazione del salto (sottoforma di codice binario a 2 byte).

23620 **1 byte** **NSPPC**

Contiene il numero dello statement inserito nella linea alla quale deve essere effettuato il salto. Di solito il numero è 255. Inserendo, tramite POKE, un numero qualsiasi tra 0 e 127, verrà imposto un salto all'istruzione che ha il numero di riga contenuto in NEWPCC, per esempio:

```
  10 POKE 23618,100: POKE 23619,
0
  20 POKE 23620,2: PRINT "20"
  30 PRINT "30"
 100 PRINT "100.1": PRINT "100.2
": PRINT "100.3"
```

che visualizza

```
100.2
100.3
```

È questo il solo modo per saltare ad una istruzione che si trova nel mezzo di una linea di programma.

```
23621,2          2 bytes  PPC
```

Contiene il numero di linea dell'istruzione in corso di esecuzione (è un numero binario a 2 byte). L'inserimento mediante POKE di un altro numero, influirà su qualsiasi messaggio di errore causato da questa riga. Una istruzione GO SUB, memorizzerà il valore di PCC come quello della linea alla quale sarà necessario ritornare incontrando un RETURN. Esempio

```
100 POKE 23621,10: GO SUB 1000
```

causerà un ritorno alla linea 10 invece che a quella successiva alla 100.

```
23623            1 byte   SUBPPC
```

Contiene il numero dell'istruzione situata entro la linea in corso di esecuzione. Come per PCC, viene usata dalla routine di messaggio di errore ed anch'essa memorizzata per ritorno da qualsiasi GO SUB.

```
23624            1 byte   BORDCR
```

Contiene il numero del colore di border (da 0 a 7) moltiplicato per 8, più gli Attributi utilizzati per la metà inferiore dello schermo. La modifica del valore non provoca alcun effetto, a meno che il POKE non sia seguito da CLS oppure INPUT. In questo ultimo caso, alla pressione del tasto ENTER, il colore di border varierà.

```
23627,8          2 bytes  VARS
```

Contiene l'indirizzo iniziale dell'area delle variabili nella RAM e, poiché questo segue immediatamente la fine del vostro programma, potrete scoprire quanti sono i byte occupati dallo stesso per mezzo di

```
PRINT PEEK 23627+256*PEEK 23628-PEEK 23635-256
*PEEK 23636
```

```
23629,23630      2 bytes  DEST
```

Contiene l'indirizzo RAM della variabile di destinazione (cioè quella a sinstra del segno "="), di una istruzione LET. Per cui

```
LET a=PEEK 23629+256*PEEK 23630
```

renderà "a" uguale all'indirizzo nel quale è memorizzata. Se la variabile "a" non è stata mai usata in precedenza, DEST punterà ad un carattere "a" nella linea di programma. Il contenuto di questa locazione può essere impiegato per trovare la cella di memoria usata per una variabile numerica.

```
23635,6          2 bytes   PROG
```

Contiene l'indirizzo iniziale del vostro programma BASIC (vedere VARS 23627).

```
23653,4          2 bytes   STKEND
```

Contiene l'indirizzo iniziale dello spazio di riserva nella RAM. La RAM che si trova tra questo indirizzo e quello dato in RAMTOP (23730/1) viene utilizzata soltano per gli stack di macchina e di GO SUB. Il numero dei bit della zona RAM libera si ottiene con

```
PEEK 23730+256*PEEK 23731-PEEK 23653-256*
PEEK 23654
```

```
23658            1 byte    FLAGS2
```

Usato per diversi flag del sistema, contiene normalmente uno 0, a meno che non sia stato predisposto un CAPS LOCK, nel qual caso conterrà un 8. Viceversa un POKE di valore 8, attiverà CAPS LOCK.

```
23659            1 byte    DF.SZ
```

Contiene il numero delle linee riservate (compresa una vuota) che si trovano nella parte inferiore dello schermo. Si dispone a 2 all'accensione del computer e tutte le volte che si esegue un comando diretto o una istruzione INPUT. Se in questa locazione viene inserito un valore "n" (maggiore di 2 e minore di 25), non potrete più eseguire alcun PRINT nelle "n" linee in basso. Tali linee verranno cancellate da una istruzione INPUT o dall'emissione di un messaggio di errore. POKando il valore "1", stamperete su 23 linee.

```
 10 POKE 23659,1
 20 FOR a=0 TO 22: PRINT AT a,0
;a: NEXT a
 30 PAUSE 100
 40 POKE 23659,2
```

Ricordatevi di eseguire nuovamente un POKE di valore "2" prima che il programma termini o esegua un INPUT o un CLS. Inserendo un valore 0, vi sarà permesso di scrivere su tutte le 24 linee dello schermo, senza poter usare però AT per scrivere sulla 24esima. Per poterlo fare dovrete agire come segue:

```
PRINT AT 22,31;TAB 10;"LINE 24"
```

Sarà anche qui necessario inserire il valore "2" prima di un INPUT, di un CLS o della fine del programma, per evitare il blocco irrimediabile del programma. Un simile procedimento sta alla base dei metodi usati per impedire a chiunque di copiare o listare il vostro programma.

```
23670,1          2 bytes  SEED
```

Qui viene inserito il generatore di numeri casuali. La locazione contiene uno 0, all'accensione o dopo il NEW, ma viene portato al valore dell'argomento di RANDOMIZE (se questo è diverso da 0), oppure al numero inserito nella Variabile di Sistema FRAMES da RANDOMIZE o RANDOMIZE 0.

```
23672,3,4        3 bytes  FRAMES
```

Conta il numero dei quadri TV in 50esimi di secondo (o in 60esimi). Il capitolo 18 del manuale Sinclair riporta una serie di funzioni per ottenere il valore totale dei 3 byte interessati. La versione data nella prima edizione del manuale è errata, in quanto la linea 30 deve essere variata come segue

```
30 DEF FN t( )=FN m(FN u( ),FN u( ))
```

Volendo predisporre FRAMES ad un valore particolare, dovrete inserire per primo il byte meno significativo. Esempio

```
POKE 23672,0: POKE 23673,0: POKE 23674,0
```

```
23675,6          2 bytes  UDG
```

Contiene l'indirizzo della prima matrice di punti grafici definita dall'utente, cioè quella destinata a CHR$ 144. Al momento dell'accensione,

la locazione prende il valore di P-RAMT (Variabile di Sistema 23732/3) diminuito di 167. Il contenuto di questa locazione non viene influenzato da NEW. Se avete assoluta necessità di spazio in memoria e non volete utilizzare tutti i 21 caratteri definiti dall'utente, potete eseguire POKE UDG con un valore più elevato, ricordandovi però di abbassare RAMTOP di uno rispetto a tale valore, usando una istruzione di CLEAR. Per esempio, volendo eliminare completamente lo spazio riservato in memoria ai caratteri grafici definiti dall'utente, dovrete battere:

```
POKE 23675,255
POKE 23676,255 (o 127 per Spectrum 16 K)
CLEAR 65534    (o 32766 per Spectrum a 16 K)
```

tenendo presente anche le note riguardanti la Variabile di Sistema CHARS (23606/7)

```
23677          1 byte    X-COORD
23678          1 byte    Y-COORD
```

Queste locazioni contengono le coordinate X e Y dell'ultimo punto tracciato e vengono utilizzate come punto di partenza per i comandi DRAW e CIRCLE. Pertanto un POKE in una di queste locazioni influenzerà la successiva operazione di DRAW o di CIRCLE.

```
23679          1 byte    P.POSN
```

Contiene il valore 33 meno il numero della colonna in cui apparirà il successivo carattere stampato con LPRINT. Se, ad esempio, questa locazione contiene "33", il carattere successivo comparirà nella colonna "0", mentre il valore "2" farà apparire il successivo carattere alla fine della linea.

Il valore "1" indica che è già stato stampato il 32esimo carattere di quella linea, ma che la posizione di scrittura non è ancora stata trasferita alla linea seguente. Un POKE a questa locazione non ha, di per sé, effetto alcuno.

```
23680          1 byte    PR.CC
```

Il manuale dello Spectrum riporta che in questa locazione è contenuto il byte meno significativo dell'indirizzo della posizione successiva, riferito al buffer della stampante. È 0 quando il buffer è vuoto e viene

incrementato di 1 ad ogni carattere stampato mediante LPRINT. POKando altri valori, influenzerete la posizione del carattere successivo da stampare, a patto che il valore inserito si adatti a P POSN (23679) per il riconoscimento di fine riga.

23688 1 byte S.POSN

Contiene 32 meno il numero di colonna della posizione di PRINT sullo schermo. È analoga a P POSN e un POKE in tale locazione non ottiene alcun effetto.

23689 1 byte

Contiene 24 meno il numero di linea della posizione di scrittura sullo schermo. Il POKE di un nuovo valore influenza la posizione dei caratteri stampati dopo la successiva istruzione di PRINT, oppure dopo il successivo simbolo di interlinea (‘). Potete usare questa locazione assieme alla Variabile di Sistema 23688 per verificare l'attuale posizione di scrittura sullo schermo.

23692 1 byte SCRCT

All'accensione e dopo NEW e RUN contiene il valore "1", mentre prima di arrestare il listato del programma e porre la domanda "scroll?" contiene il numero di righe visualizzate più uno, e viene reinizializzato a 22 se premete un qualunque tasto diverso da N o BREAK. Effettuando il POKE di un valore opportuno in questa locazione di memoria potete ottenere lo scroll automatico del listato sul video. Per avere la massima lunghezza possibile di scroll automatico usate:

```
POKE 23692,0
```

23693 1 byte ATTR.P

Contiene gli Attributi permanenti generati dalle istruzioni FLASH, PAPER ecc. Si predispone a 56 all'accensione e al comando NEW. Con un POKE di adeguato valore, potete predisporre tutti gli Attributi Permanenti in una sola volta, oppure (tramite PEEK 23693), rilevare i loro valori.

23694 1 byte MASK.P

Riguarda gli Attributi Permanenti "trasparenti". Ogni bit a livello "1", eguaglia il corrispondente attributo a quello già in atto sullo schermo nella posizione considerata, anziché a quello contenuto in ATTR P. Il contenuto viene azzerato all'accensione o al NEW.

23730,1 2 bytes RAMTOP

Contiene l'indirizzo dell'ultimo byte dell'area di sistema del BASIC nella RAM, che è inferiore di uno a quello del primo byte dell'area dei caratteri grafici definiti dall'utente (UDG). Il contenuto di questi due byte è predisposto a 65367 (per lo Spectrum a 48 K), oppure a 32599 (16 K) all'accensione del computer. Questi valori non vengono influenzati da NEW. Un POKE alla locazione non provoca alcun effetto; potete cambiare RAMTOP solo con una istruzione CLEAR n.

23732,3 1 byte P-RAMT

Contiene l'indirizzo dell'ultimo byte di RAM del sistema. All'accensione vale 65535 (48 K) oppure 32767 (16 K). POKE di diversi valori non hanno alcun effetto. Questa variabile viene di solito usata per comunicare al programma su che tipo di Spectrum sta funzionando:

```
IF PEEK 23733=255 THEN PRINT "48K"
```

Display File 16384-22527

Non ha in realtà molto significato effettuare PEEK o POKE nell'area del display file della RAM, in quanto le istruzioni PLOT, PRINT, POINT e SCREEN$ eseguono sempre le necessarie funzioni con maggior facilità. Questa breve descrizione mira più che altro a mostrarvi il funzionamento del display file.

La posizione di ogni carattere sullo schermo, corrisponde ad 8 byte di RAM, ciascuno dei quali determina una riga di 8 punti. La locazione RAM del byte più alto di un carattere è data da:

```
16384 +32*(y+56*INT(y/8))+x
```

dove "x" e "y" sono le stesse usate in una istruzione PRINT AT y, x. Gli altri sette byte per la posizione del carattere sono disseminati nella RAM ad intervalli di 256 byte. Vale a dire che l'indirizzo del byte che corrisponde alla seconda riga di punti in un carattere, ha un valore più alto di 256 rispetto a quello della prima fila in alto, e l'indirizzo dell'ultimo byte possiede un valore pari a 7×256 volte quello della prima riga.

Come dimostrazione provate:

```
 100 FOR y=0 TO 23: IF y<=21 THE
N PRINT AT y,15;y
 110 LET p=16384+32*(y+56*INT (y
/8))
 120 FOR x=0 TO 31 STEP 8
 130 FOR r=0 TO 7: POKE p+x+r+25
6*r,BIN 10101010: NEXT r
 140 NEXT x: NEXT y
 150 PAUSE 0
```

Il programma mostra il modo (anche se non molto pratico) di disegnare un grafico sulle ultime due righe dello schermo.

Area degli Attributi 22528-23295

Questa area di RAM contiene i byte degli Attributi per ciascuna delle 24×32 posizioni di carattere sullo schermo. La locazione di memoria del byte che corrisponde al carattere posto nella riga y, colonna x, sarà

```
22528+x+32*y
```

È più facile cambiare i contenuti usando l'istruzione

```
PRINT OVER 1;AT y,x;" "
```

che comprende una funzione temporanea PAPER, BRIGHT, INK o FLASH ed esaminare il contenuto con

```
ATTR (y,x)
```

Per controllare se POKE funziona, usate la seguente routine che cambia il colore di PAPER a tutte e 24 le linee.

```
  100 FOR a=0 TO 21: PRINT AT a,1
5;a: NEXT a
  110 FOR p=0 TO 56 STEP 8
  120 FOR x=0 TO 31: FOR y=0 TO 2
3
  130 POKE 22528+x+32*y,p
  140 NEXT y: NEXT x: NEXT p
```

Buffer Stampante 23296-23551

Questa area di RAM contiene l'immagine dei punti di una fila di 32 caratteri destinata alla stampante e, inoltre, una serie di zeri impiegati all'accensione, dopo un NEW, e dopo avere inviato una riga alla stampante. I primi 32 byte contengono le righe di punti superiori di tutti i 32 caratteri; i successivi 32 byte riguardano le seconde righe, e così via.

Avrete la dimostrazione battendo la routine

```
  100 LPRINT "A";
  110 FOR a=23296 TO 23551
  120 IF PEEK a<>0 THEN PRINT a;T
AB 8;a-23296,PEEK a
  130 NEXT a
  140 LPRINT
```

che visualizzerà

```
23328   32      60      00111100
23360   64      66      01000010
23392   96      66      01000010
23424   128     126     01111110
23456   160     66      01000010
23488   192     66      01000010
```

(la trascrizione in binario, riportata sulla destra, è stata aggiunta in seguito)

Area di Programma

È un'area della RAM che inizia all'indirizzo contenuto nella Variabile di Sistema PROG (23635/6) e termina immediatamente prima dell'indirizzo contenuto nella Variabile di Sistema VARS (23627/8). Naturalmente, è usata per contenere i vostri programmi BASIC. Possiamo vedere come il programma sia effettivamente allocato nella RAM, mediante una routine del tipo sotto indicato, che stampa tre colonne relative a:

— l'indirizzo della locazione di RAM da controllare;

— il contenuto di questo indirizzo, espresso in numeri decimali interi;

— i contenuti degli indirizzi espressi, nei limiti del possibile, da uno dei caratteri o delle parole-chiave dello Spectrum.

```
   5 REM abcd
 100 LET a=PEEK 23635+256*PEEK 2
3636
 110 PRINT 'a;" ";PEEK a;
 120 IF PEEK a>=32 THEN PRINT TA
B 10;CHR$ PEEK a;
 130 LET a=a+1: GO TO 110
```

Nella riga 5, potrà essere impostato qualsiasi tipo di istruzione desideriate. In questo caso, abbiamo scelto una semplice istruzione REM, e, con l'aiuto del capitolo 24 del manuale Sinclair, è possibile interpretare il risultato dell'esecuzione di questa routine.

```
23755,6  Linea numero 5
23757,8  Lunghezza 6 bytes
23759    Parola REM
23760,3  Caratteri dopo REM
23764    Fine del codice linea
23765,6  Linea numero 100
```

```
23755 0
23756 5
23757 6
23758 0
23759 234 REM
23760 97  a
23761 98  b
23762 99  c
23763 100 d
23764 13
23765 0
23766 100 d
```

Canali

L'area di RAM relativa ai canali d'informazione, comprende quattro sezioni operative da 5 byte che occupano le locazioni da 23734 a 23753 (lo Spectrum non deve essere equipaggiato di Microdrive). I primi due byte di ciascuna sezione operativa contengono gli indirizzi della routine ROM, usata per stampare i dati, i successivi due byte contengono sia l'indirizzo della routine ROM utilizzata per ottenere l'input da tastiera, che l'indirizzo di una routine che fornisce il messaggio di errore "INVALID I/O DEV". Il quinto byte di ciascuna sezione, contiene rispettivamente il codice ASCII dei caratteri K, S, R e P.

I comandi PRINT, LPRINT e INPUT possono essere seguiti dal simbolo (#). Per esempio, in PRINT # n, "n" è noto come numero di canale. I canali 0 e 1 impiegano il primo ingresso dell'area RAM per scrivere nella metà inferiore dello schermo e per impostare dati mediante tastiera. Il canale 2 usa il secondo ingresso e si occupa della scrittura sulla metà superiore dello schermo. Il canale 3 usa il quarto ingresso per inviare i dati alla stampante.

INPUT #2 e INPUT #3 forniscono il messaggio di errore "INVALID I/O DEV". Di conseguenza, PRINT #3 può essere usato al posto di LPRINT, e PRINT #0 è un modo per scrivere sulla parte inferiore dello schermo, normalmente riservata ad INPUT ed ai messaggi di errore.

APPENDICE 2
VELOCIZZAZIONE

Desiderando far girare un programma il più velocemente possibile, è necessario conoscere quanto tempo impiega lo Spectrum per eseguire ogni operazione. Lo potrete facilmente scoprire da soli usando la routine che segue, la quale fornisce il tempo in ms (millisecondi, millesimi di secondo) necessario ad eseguire un certo numero di righe, numerate da 1000 a 1999.

```
  10 POKE 23672,0: POKE 23673,0:
 POKE 23674,0
  20 FOR i=1 TO 100: GO SUB 1000
: NEXT i
  30 LET t=PEEK 23672+256*PEEK 2
3673+65536*PEEK 23674
  40 PRINT AT 0,0;(t-39)/5
  50 STOP
2000 RETURN
```

Facendo girare la routine cosi come è, otterrete il risultato "0", mentre aggiungendo una linea come ad esempio

```
1000 LET a=.5
```

e dando nuovamente il RUN, apparirà il risultato di 1,8 ms che è il tempo impiegato dallo Spectrum per eseguire la linea aggiunta. Variando la linea 1000 potete ricavare i tempi di qualsiasi funzione; troverete che lo Spectrum impiega da 2 a 4 ms per sommare o sottrarre due numeri e da 4 a 5 ms per dividere o moltiplicare, a seconda dei valori. L'operatore di elevazione a potenza (↑) è tra i più lenti, richiedendo ben 110 ms per calcolare .5↑.5. Gli operatori logici (=, <>,<=,>=) impiegano da 3 a 4 ms, come si può vedere battendo la linea

```
1000 LET a=27<33
```

Espressioni di stringa del tipo

```
LET a$="TIMEX"
LET a$="I"+"BM"
```

impiegano da 3 a 6 ms, secondo la lunghezza delle stringhe stesse. Eccovi i tempi caratteristici per le diverse funzioni.

```
SIN.5          44mS  ASN.5       183mS  TAN.5        89mS
ATN.5          62mS  COS.5        45mS  ACS.5       185mS
LN.5           71mS  EXP.5        43mS  SQR.5       111mS
SGN.5           3mS  INT.5         3mS  ABS.5         3mS
BIN 10101010    3mS  PI            3mS  ATTR(0,0)     5mS
POINT (0,0)     5mS  PEEK 1000     3mS  IN 1000       3mS
VAL".5"        13mS  STR$.5       22mS  CHR$32        6mS
INKEY$          4mS  CODE "a"      3mS
VAL"1234.5678" 41mS
SCREEN$ (0,0) da 7 a 11mS in funzione del carattere.
```

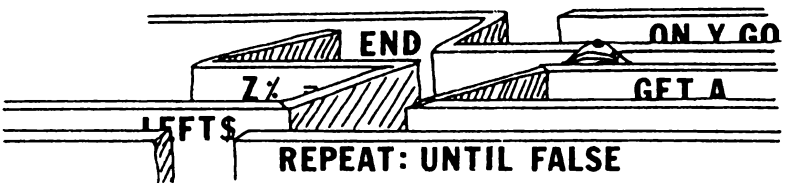
e quelli riguardanti istruzioni di stampa e di disegno

```
PRINT                          2mS  PRINT.5                17mS
PRINT"A"                       3mS  PRINT"ABCDEFGHIJKL"     8mS
PRINT"A";                      3mS  PRINT"A",              10mS
PRINT TAB 0;"A"                4mS  PRINT TAB 30;"A"       19mS
PRINT AT 0,0;"A"               6mS  PLOT 100,100            3mS
PLOT 100,100: DRAW 0,0           6mS
PLOT 100,100: DRAW 100,50       25mS
PLOT 100,100: DRAW 100,50,1    600mS
CIRCLE 100,100,50              800mS
```

I comandi temporanei INK, PAPER, FLASH, BRIGHT, OVER e INVERSE richiedono circa 2 ms per l'esecuzione. Quando lo Spectrum esegue una delle istruzioni GO TO, GO SUB, RETURN, RESTORE, NEXT, oppure una FN definita dall'utente, deve cercare la linea a cui andare scorrendo l'intero programma a partire dall'inizio. Per sondare ciascuna riga ci vogliono 0,3 ms, per cui un ciclo FOR NEXT che abbia inizio, ad esempio, alla 100esima linea, non potrebbe seguire più di 30 loop al secondo. Per tale motivo, i loop usati più frequentemente vanno posti all'inizio del listato. Oltre al tempo di ricerca, le istruzioni GO TO, GO SUB, RETURN e RESTORE richiedono ciascuna da 1 a 3 ms circa e NEXT circa 4. Il tempo di ricerca nell'area RAM delle variabili incontrate durante il programma, vale 0.05 ms per ognuna (gli array contano come le variabili).

APPENDICE 3
ALTRI BASIC

Potete trovare una grande quantità di programmi scritti in BASIC sia in altri libri che su riviste specializzate; sfortunatamente, però, esistono molte versioni di BASIC che, pur simili, non sono intercambiabili, per cui un programma scritto per altri computer non girerà sullo Spectrum, se prima non viene adeguatamente modificato. Se desiderate effettuare tali modifiche, consultate i paragrafi che seguono, i quali riportano le differenze principali tra i più diffusi "dialetti" del BASIC. Tenete presente che la vastità ed il tipo di modifiche necessarie dipenderanno dalla struttura del programma da convertire, dal modo di gestire i dati e, cosa più importante, dalle caratteristiche di visualizzazione sullo schermo di una particolare macchina. Per risparmiare tempo, dovrete necessariamente capire la struttura del programma, prima di farlo girare.



Righe con Istruzioni Multiple

La maggior parte dei BASIC permette di scrivere, come avviene per lo Spectrum, più di una istruzione sulla stessa riga.

Variabili

La maggior parte dei BASIC moderni usa l'aritmetica a virgola mobile, con variabili analoghe a quelle dello Spectrum, ma alcuni permettono anche di specificare variabili intere aggiungendo, di solito, al nome della variabile il simbolo %. Di conseguenza:
A indicherebbe una variabile a virgola mobile;
A% indicherebbe invece una variabile intera.

Quest'ultima è una variabile che può assumere esclusivamente valori interi (senza decimali) di solito usata perché occupa meno spazio in memoria che non quelle a virgola mobile e anche perché l'aritmetica delle variabili intere è assai più veloce. In questi casi non vi è alcun pericolo ad usare le variabili a virgola mobile dello Spectrum, anziché quelle intere. Talvolta il programma può ricorrere all'aritmetica intera, specialmente nella divisione tra numeri il cui risultato debba essere arrotondato al valore inferiore, per dare appunto un numero intero. Se ritenete che questo sia il caso, usate la funzione INT per rendere intero il risultato di una divisione. Ad esempio:

```
10 LET C%=Y/100
```

diventerebbe

```
10 LET C=INT (Y/100)
```

Alcuni BASIC, come per esempio l'Integer BASIC dell'Apple II ed il BASIC del vecchio ZX80 da 4K, non hanno la possibilità di usare la virgola mobile; tutte le loro variabili sono perciò di tipo intero, anche se non sono seguite dal segno %.

La maggior parte dei BASIC che permettono di usare sia lettere maiuscole che minuscole, distinguono tra queste due forme, cosicché "A" risulterebbe una variabile diversa da "a". Ciò non avviene invece con lo Spectrum.

Array

Non ci dovrebbero essere difficoltà nell'uso delle matrici numeriche, in quanto lo Spectrum è in questo caso uguale alle altre macchine, ma

occorre ricordare che la maggior parte dei BASIC impiega, per gli array, indici che iniziano da 0 e non da 1. Di conseguenza, DIM A(3) definisce di solito una matrice di quattro elementi, da A0 ad A3. Per lo Spectrum, invece, questa sarebbe una matrice di tre elementi, da A(1) ad A(3). Gli array di alcuni BASIC usano nomi di variabili costituiti da più caratteri. Nello Spectrum questi nomi dovranno essere convertiti in caratteri singoli. Gli array di stringhe dello Spectrum sono molto simili a quelli della maggior parte degli altri BASIC (tranne per il fatto che, anche in questo caso, gli altri dialetti fanno partire la numerazione degli indici da 0 anziché da 1). Esistono, però, differenze nel modo in cui concepiscono la lunghezza delle stringhe in una matrice. L'istruzione DIM, oltre a definire in numero di stringhe in una matrice, definisce anche la lunghezza massima di ciascuna stringa, in quanto, se queste sono troppo lunghe, viene emesso il segnale di errore. Lo Spectrum, dal canto suo, regola la lunghezza di una stringa in modo da adattarla esattamente alla lunghezza DIMensionata per la stringa stessa. Questo può, talvolta, creare qualche problema quando viene eseguito il confronto tra due stringhe, di cui una compresa in un array. Potreste trovarvi di fronte ad una istruzione DIM di stringa come la seguente

```
DIM A$(2,2) [10]
```

che definisce un array di stringhe 2×2 (oppure 3×3), nella quale ciascun elemento ha una lunghezza massima di 10 caratteri. Ciò equivale alla forma Spectrum

```
DIM A$(2,2,10)
```

oppure

```
DIM A$(3,3,10)
```

Taluni BASIC richiedono che tutte le variabili stringa (anche quelle che non sono array) siano dimensionate in modo da riservare loro il necessario spazio in memoria. Altri BASIC necessitano di istruzioni DIM esclusivamente per variabili di stringa, che si suppone siano più lunghe, per esempio, di 10 caratteri. In questi casi, potreste incontrare linee di programma come la seguente

```
DIM B$(15)
```

superflue nella versione Spectrum. Alcuni BASIC permettono il dimensionamento di più matrici con una sola istruzione DIM. Ciò non può essere fatto con lo Spectrum, di conseguenza:

```
10 DIM A(2,2),B(50)
```

dovrà essere riscritta così:

```
10 DIM A(2,2): DIM B(50)
```

Funzioni Aritmetiche

Le funzioni aritmetiche usate dallo Spectrum, sono molto simili a quelle disponibili su altri computer tranne per il fatto che alcuni BASIC usano il simbolo "**" anziché quello "↑" per l'innalzamento a potenza.

Alcuni computer hanno le seguenti funzioni

LOG: logaritmo in base 10
LOG (x) è equivalente a LN(x)/LN(10)

DIV: restituisce la parte intera del risultato di una divisione
A DIV B
va cambiato in
INT (A/B)

MOD: fornisce il resto dopo una divisione intera;
ad esempio:
21 MOD 5
restituisce 1. In generale:
A MOD B
va cambiato in
A - B*INT (A/B)

ASC, CODE

ASC(X$) dà il valore decimale del codice del primo carattere della stringa X$ ed è equivalente a CODE. Se state convertendo un programma scritto per lo ZX 80 e lo ZX 81, fate attenzione che i codici dei loro caratteri sono differenti da quelli usati nello Spectrum, il quale usa gli standard ASCII per le lettere, i numeri ed i simboli più comuni.

END

Deve essere sostituito da STOP.

FOR - TO - STEP - NEXT.

Alcuni BASIC permettono di usare un nome di variabile a più caratteri per il controllo di un ciclo FOR-NEXT. Lo Spectrum non lo fa. Un piccolo problema potrebbe sorgere dal fatto che molte versioni BASIC eseguono sempre, almeno una volta, le istruzioni che stanno tra FOR e NEXT, anche se il valore iniziale della variabile di controllo è maggiore del valore limite. Per cui

```
10 FOR I=1 TO 0
20 PRINT "LINEA 20"
30 NEXT I
```

provocherebbe in altri casi la stampa di "LINE 20", ma non con lo Spectrum. Vi sono dei BASIC che tralasciano addirittura il nome della variabile dopo il NEXT in quanto presumono che si tratti della variabile usata dal FOR più recente. Lo Spectrum non ammette neanche questo.

GET, GET$, INKEY$

In alcuni BASIC, un'istruzione del tipo:

```
GET A$ oppure GET A oppure LET A$=GET$
oppure LET A=GET
```

farà attendere il computer finché l'utente non avrà premuto un tasto, e quindi ritornerà una stringa costituita da un solo carattere (oppure dal codice numerico) che rappresenta il tasto premuto.

L'istruzione INKEY$ dello Spectrum è analoga, ma non attende la pressione di un tasto; per ottenere ciò si dovrà aggiungere una piccola routine.

```
10 LET A$=INKEY$: IF A$="" THEN GO TO 10
10 LET A=CODE INKEY$: IF A=0 THEN GO TO 10
```

Alcuni computer hanno la funzione INKEY$(X) che li fa attendere finché venga premuto un tasto, ma solo per un tempo X (espresso in decimi o centesimi di secondo a seconda della macchina).

Questa istruzione può essere sostituita dalla riga

```
10 PAUSE X: LET A$=INKEY$
```

poiché la pressione di un qualsiasi tasto pone fine alla pausa (PAUSE).

GO TO, GO SUB

Alcuni BASIC non permettono un GO TO o un GO SUB calcolato come il GO TO A*100 dello Spectrum. Traendo vantaggio da questa possibilità, sarete in grado di semplificare il programma. Certi dialetti includono una etichetta (label) all'inizio di una riga, immediatamente prima, o dopo, il numero della linea stessa.

Questa etichetta, che di solito ha un nome simile a quello delle variabili, può essere seguita da un punto e virgola e viene usata dopo un GO TO o un GO SUB, al posto del numero di linea che costituisce il punto d'arrivo. Esempio

```
10 GO TO A
- - - - -
100 A;LET C=D
```

che va convertita in

```
10 GO TO 100
- - - - -
100 LET C=D
```

Diversi BASIC utilizzano le forme "ON..GOTO" oppure "ON..GOSUB" le quali fanno eseguire al programma un GOTO o un GOSUB ver-

so una linea in funzione del valore di una variabile. Ad esempio

```
ON I GO TO 100,105,130
```

determinerà un salto alla riga 100 se I=1, alla riga 105 se I=2 o alla riga 130 se I=3.

Secondo le esigenze, queste forme possono essere sostituite da un GO TO (o un GO SUB) calcolato, oppure da una sequenza di righe di programma IF-THEN-GO TO. Con un po' di abilità potrete utilizzare l'operatore AND, cosicché la versione Spectrum della linea suddetta, sarà

```
GO TO (100 AND I=1)+(105 AND I=2)+(130 AND I=3)
```

IF-THEN-ELSE

Molti BASIC non danno importanza al fatto che venga o meno inclusa la parola THEN,cosicché vi potreste trovare nella situazione seguente

```
IF A=B GO TO 100
IF A$="SI" PRINT"NO"
```

Osservate che i BASIC differiscono tra loro nell'interpretazione di un valore come "vero" o come "falso"; per esempio

```
IF A THEN PRINT "VERO"
```

Scriverà "VERO" su uno Spectrum se "A" ha un qualsiasi valore che non sia 0. Ciò potrebbe anche non accadere per altri BASIC che potrebbero usare valori negativi per "falso" e il valore 0 per "vero". Talvolta è possibile trovare ELSE, che è una delle appendici più utili della costruzione IF..THEN. Esempio

```
IF A>B THEN PRINT "Piu' grande A" ELSE PRINT
"Piu' piccolo A"
```

che, nel BASIC dello Spectrum, si dovrebbe scrivere

```
10 IF A>B THEN PRINT "Piu' grande A": GO TO 12
11 PRINT "Piu' piccolo A"
12 - - - -
```

o anche

```
10 PRINT ("Piu' grande A" AND A>B)+("Piu' piccolo
A" AND A<=B)
```

INSTR(A$, B$)

È una funzione molto utile che controlla se B$ è compresa in A$. In caso positivo, la funzione vi dice a quale carattere di A$ ha inizio B$

```
INSTR("SINCLAIR","IN")
```

darà come risultato 2. Se B$ non viene trovato in A$, viene ritornato il valore 0.

Per svolgere questa funzione col BASIC dello Spectrum, si rende necessaria una speciale routine

```
  10 FOR A=0 TO LEN A$-LEN B$
  20 IF B$=A$(A+1 TO A+LEN B$) T
HEN LET A=A+1: GO TO 40
  30 NEXT A: LET A=0
  40 REM - - -
```

che equivale a

```
LET A=INSTR(A$,B$)
```

LEFT$, MID$, RIGHT$

Tali funzioni si trovano in molte versioni di BASIC. Esse agiscono su di una stringa e danno come risultato una parte di essa.

LEFT$(X$,Y) darà i primi Y caratteri di X$; il suo equivalente nello Spectrum è X$(TO Y).

RIGHT$(X$, Y) fornisce il numero Y di caratteri che si trovano più a destra nella stringa X$. La sua traduzione per lo Spectrum è X$(LEN X$–Y+1 TO).

MID$(X$,Y,Z) fornisce il numero Z di caratteri della stringa X$ che parte dell'Y-esimo carattere. Ciò equivale a X$(Y TO Y+Z–1).

LET

In molti BASIC, la parola LET può essere omessa, ad esempio

```
10 A=100
```

Per lo Spectrum, l'espressione dovrà essere riscritta nel seguente modo

```
10 LET A=100
```

MAT

Le versioni di BASIC più sofisticate hanno comandi che permettono di operare direttamente sulle matrici. Esempio

```
10 DIM A(X,Y)
20 DIM B(X,Y)
- - -
100 MAT A=B
```

rende tutti gli elementi della matrice A( ) uguali ai corrispondenti elementi della matrice B( ). Per ottenere il medesimo risultato sullo Spectrum è necessaria una speciale routine:

```
100 FOR p=1 TO X: FOR q=1 TO Y
101 LET A(p,q)=B(p,q)
102 NEXT q: NEXT p
```

PEEK, POKE, USR, CALL, LINK

LINK e CALL sono analoghe ad USR, in quanto richiamano una routine in linguaggio macchina, ma esse non trasferiscono alcun valore al programma BASIC che ha effettuato la chiamata. L'effetto di questi comandi dipende interamente dalla particolare macchina che viene usata. Per convertire un programma che contenga una qualsiasi di queste istruzioni, allo scopo di farlo funzionare sullo Spectrum, dovrete scoprire cosa esattamente debba fare il comando, per poi scrivere l'istruzione dello Spectrum in grado di fare la stessa cosa.

PLOT, DRAW

Sembra che ciascuna versione di BASIC in possesso delle funzioni di disegno impieghi una forma diversa di istruzioni PLOT e DRAW. Tutto quello che potete fare è trovare i compiti che avevano le istruzioni originali e scrivere una routine Spectrum che sia in grado di farli ugualmente.

PRINT

Le istruzioni di stampa dovranno spesso essere cambiate, per far uso del tracciato di schermo dello Spectrum. Vi potrà capitare anche di trovare funzioni CHR$ usate entro istruzioni PRINT, per la visualizzazione di codici speciali di controllo, o simboli grafici. Queste funzioni differiscono da un computer all'altro. PRINT USING è un comando molto potente, disponibile solo in alcuni BASIC. Esso permette al programmatore di controllare il formato della stampa, secondo una configurazione inserita in una sezione del programma. Tale configurazione è, di solito, destinata a specificare la spaziatura, la giustezza di destra e di sinistra degli stampati, nonché il numero delle cifre presenti dopo la virgola decimale.

PROCedure

Il BASIC dello Spectrum vi permette di definire una nuova funzione con una sola riga di programma, come

```
DEF FN a(x,y)=INT((x+y)/2)
```

Alcuni BASIC, invece, permettono una definizione della funzione su righe multiple

```
DEF FN a(x,y)
LET s=0
FOR k=x TO y
LET s=s+k
NEXT s
=s
```

che darà come risultato la somma dei numeri da x a y.

Ma esiste anche un'altra costruzione, piuttosto simile, usata in alcune forme progredite di BASIC e conosciuta come "procedura".

Questa è un incrocio tra una funzione e una subroutine, in quanto può usare variabili fittizie, come nel caso di una funzione, mentre il programma viene scritto come se fosse una subroutine. La procedura è

definita da una sezione di programma, che inizia con la parola DEFPROC, seguita da un nome e termina con la parola ENDPROC. Un esempio potrebbe essere

```
DEF PROC printotl(x,y,z)
PRINT "Totale = ";x+y+z
ENDPROC
```

Tale forma può essere inserita nel programma, quando necessario, semplicemente scrivendo il nome della procedura seguito dagli effettivi valori sui quali si deve operare. Di conseguenza le istruzioni

```
printotl (5,6,20)
printotl (A,B,C)
```

stamperebbero rispettivamente il valore 31 e la somma di A, B e C.

RND

In certi BASIC, l'istruzione RND deve essere seguita da un numero posto tra parentesi, per esempio

```
LET A=RND(1)
```

Detto numero può, di solito, essere ignorato, tranne quando il programma riguarda un BASIC in grado di elaborare soltanto valori interi (come il BASIC dello ZX80 da 4K). In questi casi, RND(n) fornirà un valore intero casuale compreso tra 1 e "n" oppure tra 0 e "N–1".

Concatenazione di Stringhe

Alcuni BASIC impiegano il simbolo "&" oppure "!" per concatenare due stringhe

```
LET A$=B$ & C$      oppure      LET A$=B$ ! C$
```

equivalgono all'istruzione dello Spectrum

```
LET A$=B$+C$
```

REPEAT/DO - UNTIL

È questa una struttura che fa ripetere le istruzioni comprese tra le parole chiave REPEAT e UNTIL, fino a che non risulti vera l'espressione condizionale associata alla parola UNTIL.

Esempio

```
10 REPEAT
20 INPUT "Nome";n$
30 UNTIL n$="Gino Rossi"
```

Il modo più semplice per adattare una tale funzione allo Spectrum, è quello di cambiare REPEAT in una REM e la UNTIL in un IF-THEN-GO TO. Per esempio

```
10 REM
20 INPUT "Nome";n$
30 IF n$<>"Gino Rossi" THEN GO TO 10
```

Spesso è possibile trovare

```
REPEAT - - - - UNTIL 0
REPEAT - - - - UNTIL FALSO
```

ed è un ciclo senza fine. Basterà sostituire l'istruzione UNTIL con un GO TO non condizionato. Alcuni BASIC usano la parola DO invece di REPEAT.

VAL

La funzione VAL dello Spectrum è scomoda in quanto ferma il programma se il suo argomento stringa non contiene una espressione numerica perfettamente valida. Altri BASIC si limitano a rendere il valore 0.

?

Usato da molti BASIC come abbreviazione dell'istruzione PRINT.

Questo libro è rivolto a tutti i possessori di uno Spectrum, siano essi principianti o programmatori esperti.
Nel testo sono spiegate le caratteristiche principali del BASIC Spectrum e viene mostrato come si possono scrivere programmi di giochi o di applicazioni più serie.
Sono riportati i listati di oltre cinquanta programmi, tutti commentati dettagliatamente, e sono incluse tre appendici con alcune utilissime informazioni sulle locazioni di memoria dello Spectrum.
Chiunque voglia imparare il BASIC dello Spectrum e cerchi trucchi ed effetti "speciali" per i suoi programmi, troverà sicuramente interessante questo libro.

L. 34.000

COD 418 D ISBN 88-7056-228-X

GRUPPO
EDITORIALE
JACKSON

Mike Lord

ALLA SCOPERTA DEL BASIC SPECTRUM

193