Commodore Italiana s.p.a.
Via Fratelli Gracchi, 48
20092 Cinisello Balsamo (MI)

commodore COMPUTER

COMMODORE 64

SIMONS' BASIC

COD. 64108

114 COMANDI BASIC ADDIZIONALI

commodore COMPUTER

SIMONS' BASIC
GUIDA PER L'UTENTE

COD. 64108

Commodore Italiana s.p.a.
Via F.lli Gracchi, 48 - 20092 Cinisello Balsamo (MI)
Tel. (02) 618321

DUE PAROLE SULL'AUTORE

Quando David Simons compì 13 anni, suo padre gli regalò un computer COMMODORE. David iniziò così ad interessarsi di computer e sviluppò una conoscenza veramente notevole per la sua età. Da questa esperienza è nato il SIMONS' BASIC. Spinto dal desiderio che il suo COMMODORE fosse dotato di un set di comandi il più vasto e potente possibile, David studiò le possibilità offerte dal BASIC di altri microcomputer e addirittura di qualche mini; scegliendo il meglio tra le varie possibilità, è nato il set di 114 nuovi comandi che ora costituisce il SIMONS' BASIC.

Ed è una soddisfazione, non disgiunta da una punta di orgoglio, che oggi COMMODORE mette a disposizione dei suoi clienti il potente SIMONS' BASIC, nato dalla passione per i computer di David, uno studente di appena 16 anni.

COMMENTI E SEGNALAZIONI DI ERRORI

PER IL LETTORE

Questo manuale è stato accuratamente riveduto e corretto prima della stampa. Se tuttavia doveste riscontrare qualche errore vi saremo grati se ci farete pervenire segnalazioni a riguardo. Sono inoltre graditi eventuali commenti, critiche e suggerimenti.

AugurandoVi buon lavoro con SIMONS' BASIC porgiamo i migliori saluti.

Rinaldo Farina

R. Farina

Software Manager

Commodore Italiana S.p.A.
Via F.lli Gracchi, 48
20092 Cinisello Balsamo (MI)

DIRITTI D'AUTORE SUL SOFTWARE

DIRITTI D'AUTORE SUL MANUALE

RECLAMI

Sebbene i programmi siano stati accuratamente controllati dalla COMMODORE prima della loro distribuzione, non verrà accettato alcun reclamo riguardante questo software. La Società COMMODORE ITALIANA s.p.a. ed i suoi distributori non si assumono obblighi, nè responsabilità alcuna relativi a danni o perdite di qualunque tipo derivanti dall'uso di questi programmi. Applicazioni del software per scopi particolari debbono essere accuratamente ed interamente provate, prima della loro utilizzazione. Se avete bisogno di informazioni sull'installazione o sulla manutenzione del software, o su training relativi, siete pregati di rivolgervi alla COMMODORE ITALIANA s.p.a.

INDICE GENERALE

INDICE DELLE ILLUSTRAZIONI

Figura

CAPITOLO 1
INTRODUZIONE AL SIMONS' BASIC

1.1 INTRODUZIONE

La cartuccia del SIMONS' BASIC è stata progettata per darvi la possibilità di accedere a tutta la potenza del vostro computer COMMODORE 64. Ciò è possibile grazie ad un set di 114 comandi BASIC addizionali che si aggiunge a quello del BASIC standard del COMMODORE 64. Questi comandi supplementari appartengono a dodici diversi gruppi qui elencati e descritti:

comandi di **AUSILIO ALLA PROGRAMMAZIONE,** come ad esempio KEY e TRACE, che vi permettono di programmare in BASIC in modo più veloce ed efficace;

comandi per il **TRATTAMENTO DI STRINGHE DI CARATTERI,** come ad esempio INSERT e PLACE, che vi permettono il controllo totale della gestione delle stringhe;

comandi per la **GESTIONE DEI TESTI,** come ad esempio CENTRE e PRINT AT, che vi permettono una più semplice formattazione dello schermo;

comandi potenziati di **INPUT,** come ad esempio FETCH ed INKEY, che vi permettono il controllo totale di tutto ciò che introducete da tastiera;

OPERATORI ARITMETICI supplementari, come ad esempio MOD e DIV, che vi forniscono un metodo più immediato per effettuare la divisione intera;

comandi di **CONVERSIONE NUMERICA,** che vi permettono di trasformare numeri in base binaria od esadecimale negli equivalenti in base decimale;

comandi di **PROGRAMMAZIONE STRUTTURATA,** come ad esempio PROC ed IF.. THEN..ELSE, che vi permettono di scrivere programmi in forma più facilmente interpretabile;

comandi per la **GESTIONE DELLO SCHERMO,** come ad esempio SCRV o COPY, che vi permettono di memorizzare e di richiamare i dati sullo schermo e/o di produrre una copia stampata di ciò che appare sullo schermo, sia ad alta che a bassa risoluzione;

comandi per la **GRAFICA,** come ad esempio CIRCLE e PAINT, che vi permettono di produrre disegni sul video;

comandi per definire e gestire SPRITE e per definire SIMBOLI su specifiche dell'utente, come ad esempio DESIGN, MOB SET, DETECT e CHECK, che vi permettono di creare ed animare «oggetti grafici personalizzati» e di disegnare caratteri grafici secondo le vostre esigenze;

comandi **MUSICALI,** come ad esempio WAVE ed ENVELOPE, che vi permettono di creare effetti sonori e di comporre e suonare musica;

comandi per la **GESTIONE DELL'UNITÀ A FLOPPY DISK,** come ad esempio DIR, che vi permettono una gestione dei file di estrema semplicità.

Il set completo di comandi permessi dal SIMONS' BASIC, fa di quest'ultimo uno strumento indispensabile per chiunque sia interessato ad ottenere le massime prestazioni dal suo COMMODORE 64.

Lo scopo del presente manuale non è quello di insegnarvi a programmare in BASIC sul COMMODORE 64. Se non conoscete ancora le nozioni fandamentali di programmazione in linguaggio BASIC, siete pregati di consultare uno dei seguenti testi:

COMMODORE 64 - Manuale per l'utente (in dotazione al vostro computer)

An introduction to BASIC - Parte I e II, di Andrew COLIN.

1.2 IL MANUALE DEL SIMONS' BASIC

Questo manuale è costituito da tredici capitoli, qui elencati e descritti.

CAPITOLO 1 - INTRODUZIONE AL SIMONS' BASIC
In questo capitolo sono descritti a grandi linee i comandi del SIMONS' BASIC, la procedura per inserire nel COMMODORE 64 la cartuccia relativa ed il modo di inserire nel computer i comandi stessi. Sono inoltre riportate le convenzioni utilizzate nel manuale per descrivere ogni comando. Infine, è discussa la compatibilità del SIMONS' BASIC con il BASIC standard del COMMODORE 64 e sono riportate le istruzioni su come memorizzare, richiamare e lanciare i programmi scritti in SIMONS' BASIC.

CAPITOLO 2 - AUSILII ALLA PROGRAMMAZIONE
In questo capitolo sono descritti comandi come AUTO e TRACE, grazie ai quali potrete programmare in BASIC in modo più veloce ed efficace. È inoltre descritto il comando KEY che permette di programmare i tasti funzione del COMMODORE 64.

CAPITOLO 3 - CONFERMA DELL'INPUT E GESTIONE DEI TESTI
In questo capitolo sono descritti comandi come INSERT e PLACE, che potenziano il trattamento delle stringhe di caratteri. Sono inoltre compresi i comandi FETCH ed INKEY, che vi permettono entrambi di controllare gli input che effettuate da tastiera. Infine, sono descritti in questa sezione i comandi per la formattazione dello schermo, come ad esempio CENTRE e PRINT AT.

CAPITOLO 4 - OPERATORI ARITMETICI SUPPLEMENTARI
In questo capitolo sono descritti tre operatori aritmetici supplementari: MOD, DIV e FRAC. I primi due comandi sono relativi alla divisione intera, mentre il terzo vi permette di estrarre la parte frazionaria di un numero qualunque. Sempre in questo capitolo, sono descritti i comandi % e $, utilizzati per convertire un numero rispettivamente in base binaria o esadecimale, nell'equivalente numero in base decimale. Infine, è descritto il comando EXOR che vi permette di effettuare la relativa operazione logica booleana.

CAPITOLO 5 - COMANDI PER LA GESTIONE DELL'UNITÀ A FLOPPY DISK
In questo capitolo sono descritti i due comandi DISK e DIR. Il primo di essi vi permette di effettuare alcune operazioni sui floppy, quali ad esempio la formattazione e la cancellazione di file con un unico comando: ciò significa che il canale sul floppy viene chiuso automaticamente dopo che è stata portata a termine l'operazione desiderata. Il comando DIR, invece, vi permette di visualizzare sul video tutto o in parte il directory del floppy in uso.

CAPITOLO 6 - GRAFICA
In questo capitolo è descritta una vasta gamma di comandi per la grafica, propri del SIMONS' BASIC. Questi comandi vi permettono di disegnare sullo schermo i contorni di superfici e quindi di colorare le stesse con un colore a vostra scelta fra i 16 possibili con il COMMODORE 64.

CAPITOLO 7 - GESTIONE DELLO SCHERMO
In questo capitolo è descritto come potete effettuare lo scroll di una parte di ciò che appare sullo schermo in una direzione qualunque. Sono inoltre descritti i comandi che vi permettono di spostare una parte di ciò che appare sullo schermo in un'altra posizione, di cambiare il colore dei caratteri presenti sullo schermo, nonchè di memorizzare i dati presenti sullo schermo e quindi di richiamarli sullo stesso. Sono infine descritti i comandi che vi permettono di stampare ciò che è rappresentato sullo schermo, sia nel modo a bassa risoluzione, che in quello ad alta risoluzione.

CAPITOLO 8 - SPRITE E SIMBOLI DEFINIBILI DALL'UTENTE
In questo capitolo sono descritti i comandi relativi alla creazione ed all'animazione degli SPRITE. Sono inoltre descritte le istruzioni che vi permettono di creare dei simboli grafici secondo le vostre esigenze.

CAPITOLO 9 - PROGRAMMAZIONE STRUTTURATA
In questo capitolo è descritto il set di comandi per la programmazione strutturata, previsto dal SIMONS' BASIC.

CAPITOLO 10 - INTERCETTAZIONE DEGLI ERRORI
In questo capitolo sono descritti dei comandi che vi permettono di intercettare alcuni errori BASIC, prendendo quindi le opportune decisioni per portare comunque a buon fine i vostri programmi.

CAPITOLO 11 - EFFETTI SONORI E MUSICALI CON IL SIMONS' BASIC
In questo capitolo sono descritti i comandi del SIMONS' BASIC che vi permettono di comporre e suonare musica con il vostro COMMODORE 64.

CAPITOLO 12 - FUNZIONI READ
In questo capitolo sono descritte funzioni, come ad esempio PENX e POT, che vi permettono di controllare il programma per mezzo di dispositivi particolari, come ad esempio un joystick.

CAPITOLO 13 - ESEMPI DI PROGRAMMAZIONE IN SIMONS' BASIC
In questo capitolo sono riportati i listati di 4 programmi scritti in SIMONS' BASIC per dare un'idea delle possibilità ottenibili per mezzo della relativa cartuccia.

APPENDICE A - MESSAGGI DI ERRORE
L'elenco dei messaggi di errore che si possono presentare quando usate il SIMONS' BASIC. Per ognuno di essi, sono pure riportate le probabili cause di errore.

GLOSSARIO
In questo capitolo sono infine elencati alcuni termini tecnici utilizzati nel manuale.

1.3 COME INIZIARE CON IL SIMONS' BASIC

La cartuccia contenente il SIMONS' BASIC va sempre inserita o rimossa dal COMMODORE 64 a computer spento (interruttore di accensione in posizione OFF). Inserite la cartuccia, con l'etichetta rivolta in alto, nel relativo connettore posto sul retro del computer (fate riferimento al «Manuale per l'utente» del vostro COMMODORE 64).
A questo punto, per iniziare ad usare il SIMONS' BASIC, vi basta accendere il computer con la cartuccia in posizione. Appare sullo schermo il seguente messaggio:

*** EXPANDED CBM V2 BASIC ***
30719 BYTES FREE

Nel sistema operativo del vostro COMMODORE 64, sono ora contenuti tutti i comandi del SIMONS' BASIC che possono essere utilizzati in ogni istante, come qualunque altro comando BASIC. Il SIMONS' BASIC utilizza circa 8 kbyte della memoria del COMMODORE 64.

1.4 I COMANDI DEL SIMONS' BASIC

Troverete qui di seguito l'elenco di tutti i comandi che sono stati aggiunti al sistema operativo del vostro COMMODORE 64, per mezzo della cartuccia del SIMONS' BASIC.

Comandi per effettuare l'enter, il debug ed il list dei programmi:

KEY, DISPLAY, AUTO, RENUMBER, PAUSE, MERGE, PAGE, OPTION, DELAY, FIND, TRACE, RETRACE, DUMP, COLD, OLD, RESET, CGOTO, DISAPA, SECURE.

Comandi per effettuare la gestione dei tasti, la formattazione dello schermo e la conferma dell'input:

INSERT, INST, PLACE, DUP, USE, CENTRE, AT, LIN, FETCH, INKEY, ON KEY, DISABLE, RESUME.

Comandi per effettuare la divisione intera, le conversioni di base numerica ed un'operazione logica booleana supplementare:

MOD, DIV, FRAC, %, $, EXOR.

Comandi per effettuare la gestione dell'unità a floppy disk:

DISK, DIR.

Comandi per la grafica:

COLOUR, HIRES, MULTI, NRM, HICOL, LOW COL, PLOT, LINE, REC, CIRCLE, ARC, ANGL, BLOCK, PAINT, DRAW, ROT, CHAR, TEXT, TEST, CSET.

Comandi per memorizzare, stampare ed elaborare dati sullo schermo:

scrolling LEFT/RIGHT/UP/DOWN, BCKGNDS, FLASH, OFF, BFLASH, FCHR, FCOL, FILL, MOVE, INV, SCRSV, SCRDL, COPY, HRDCOPY.

Comandi per generare ed animare sprite e creare simboli personalizzati:

DESIGN, , CMOB, MOB SET, MMOB, RLOCMOB, DETECT, CHECK, MOB OFF, MEM.

Comandi per la programmazione strutturata:

IF..THEN..ELSE, REPEAT..UNTIL, LOOP..EXIT IF..END LOOP, PROC, CALL, EXEC, END PROC, RCOMP, LOCAL, GLOBAL, NO ERROR ON ERROR, OUT.

Comandi per la sintesi musicale:

WAVE, ENVELOPE, MUSIC, VOL, PLAY, SOUND.

Funzioni per utilizzare con i vostri programmi dispositivi per giochi:

PENX, PENY, POT, JOY.

1.5 INTRODUZIONE DEI COMANDI

Tutti i comandi del SIMONS' BASIC sono introdotti nello stesso modo dei comandi BASIC standard COMMODORE. La maggior parte dei comandi del SIMONS' BASIC può essere usata direttamente sia da tastiera che da programma. Alcune eccezioni a questa regola sono indicate nel paragrafo introduttivo di ogni capitolo.

1.6 CONVENZIONI USATE NEL MANUALE

In questo manuale, il formato di ogni comando del SIMONS' BASIC viene presentato seguendo le convenzioni qui descritte:

1. le parentesi ed i termini scritti a lettere maiuscole debbono essere battuti esattamente come sono rappresentati;

2. i termini scritti a lettere minuscole indicano un elemento che deve essere introdotto dall'utente, oppure un elemento variabile, ad esempio delle coordinate o un colore usato per la grafica;

3. altri simboli, come ad esempio le virgolette e le virgole, debbono essere battute esattamente come sono rappresentati;

4. la scritta < RETURN > significa che occorre premere il tasto RETURN;

5. i tasti che non sono caratteri alfanumerici vengono indicati nel listato con il termine scritto sulla loro superficie, racchiuso tra i simboli < >, come ad esempio < CLR/HOME >. Essi appaiono sullo schermo visualizzati in video inverso.
   Se due tasti sono racchiusi tra i simboli < >, ad esempio < CTRL RVS ON >, significa che occorre tenere premuto il primo di essi e quindi premere il secondo.

6. ad eccezione del comando FIND (vedi paragrafo 2.10), tutte le parole chiave dei comandi in SIMONS' BASIC debbono essere separate dal primo parametro del comando stesso per mezzo di uno spazio.

CAPITOLO 2
AUSILII PER LA PROGRAMMAZIONE

2.1 INTRODUZIONE

Il SIMONS' BASIC mette a vostra disposizione numerosi comandi particolarmente utili per effettuare l'entry, il debug ed il listing dei programmi in BASIC, sia che comprendano comandi del SIMONS' BASIC, sia che si tratti di comandi del BASIC standard.

Il comando KEY vi permette di programmare i tasti funzione del COMMODORE 64, mentre con il comando DISPLAY potete listare le assegnazioni che avete effettuato agli stessi tasti. I comandi AUTO e RENUMBER effettuano la numerazione automatica delle linee di programma, mentre il comando MERGE vi permette di unire un programma in BASIC presente nella memoria di massa ad un programma già presente nella memoria del vostro COMMODORE 64.

Il comando PAGE vi permette di specificare il numero di righe dello schermo che desiderate impegnare con il listato di un programma. Il comando OPTION evidenzia i comandi propri del SIMONS' BASIC nel listato di un programma. Il comandi DELAY vi permette di controllare la velocità con cui si effettua lo scroll sullo schermo del listato di un programma.

I comandi TRACE e RETRACE visualizzano i numeri delle linee di programma, via via che le stesse vengono eseguite. Il comando DUMP elenca i valori di tutte le variabili, escluse le matrici. Il comando FIND localizza tutte le ricorrenze in cui si presenta una data stringa di caratteri.

Il comando PAUSE vi permette di introdurre nel programma una pausa.
Il comando CGOTO effettua un salto ad un numero di linea di programma costituito dal risultato di un calcolo. Il comando RESET istruisce il COMMODORE 64 a leggere i dati in una specificata linea di programma. I comandi SECURE e DISAPA cancellano (solo sul listato) linee di programma specificate per impedire la lettura del vostro programma a persone non autorizzate. Il comando COLD fa ritornare il COMMODORE 64 allo schermo iniziale del SIMONS' BASIC. Il comando OLD vi permette di recuperare sullo schermo un programma che era stato cancellato con il tasto NEW.

Tutti i comandi descritti in questo capitolo possono essere utilizzati sia direttamente da tastiera che come parte di un programma.

2.2 ASSEGNAZIONE DI COMANDI AI TASTI FUNZIONE

2.2.1 KEY

FORMATO: KEY numero, «codice del comando».

SCOPO: assegnare un comando ad un tasto funzione.

KEY vi permette di assegnare i comandi che desiderate ai tasti funzione del Commodore 64 e di variare dette assegnazioni secondo le vostre esigenze. Il numero presente nel formato del comando indica quale tasto funzione, da 1 a 16, desiderate usare. Il secondo parametro nel formato costituisce il codice del comando che desiderate assegnare a quel tasto. Ad ogni tasto funzione è possibile assegnare un massimo di 15 caratteri. Per accedere alle 16 funzioni definite, dovete procedere come segue: premendo semplicemente i quattro tasti funzione otterrete rispettivamente le funzioni F1, F3, F5 ed F7; tenendo premuto il tasto SHIFT e premendo uno per volta i quattro tasti funzione, otterrete rispettivamente le funzioni F2, F4, F6 ed F8; tenendo premuto il tasto con il logo COMMODORE (il primo tasto in basso a sinistra guardando la tastiera) e premendo uno per volta i quattro tasti funzione, otterrete rispettivamente le funzioni F9, F10, F11 ed F12; infine, tenendo premuti contemporaneamente i tasti logo COMMODORE e SHIFT, e premendo uno per volta i quattro tasti funzione, otterrete rispettivamente le ultime quattro funzioni F13, F14, F15 ed F16. Si noti che il codice del comando che desiderate assegnare ad ogni tasto funzione deve essere scritto tra virgolette.

ESEMPIO: assegnare il comando MOB SET al tasto funzione F8.

COMANDO: KEY 8, «MOB SET» < RETURN >

RISULTATO: il codice del comando in SIMONS' BASIC, MOB SET, è stato assegnato al tasto funzione F8 e sarà visualizzato ogni volta che premerete il tasto stesso.

2.2.2 RETURN AUTOMATICO

Per eliminare la necessità di premere il tasto RETURN dopo ogni tasto funzione, potete aggiungere l'operazione all'assegnazione stessa, come descritto sotto:

a) assegnate il comando desiderato al tasto funzione (vedere paragrafo 2.2.1); battete le virgolette di chiusura, ma non premete RETURN;

b) battete + CHR$(13) e quindi premete RETURN.

In questo modo, quando premerete quel tasto funzione, verrà generato automaticamente un RETURN, subito dopo il comando assegnato.

ESEMPIO: assegnare il comando BASIC LIST seguito da un RETURN automatico, al tasto funzione F7.

COMANDO: KEY 7, «LIST» + CHR$(13) < RETURN >

RISULTATO: premendo semplicemente il tasto funzione F7 potete ora listare un programma.

2.2.3 DISPLAY

FORMATO: DISPLAY

SCOPO: listare i comandi assegnati ai tasti funzione.

Vi permette di rivedere i comandi che in quel momento sono assegnati ai tasti funzione.

ESEMPIO: listare le assegnazioni dei tasti funzione presenti dopo aver effettuato gli esempi relativi ai due paragrafi precedenti.

COMANDO: DISPLAY < RETURN >

SCHERMO:

```
KEY 1""
KEY 2""
KEY 3""
KEY 4""
KEY 5""
KEY 6""
KEY 7"LIST" + CHR$(13)
KEY 8"MOB SET"
KEY 9""
KEY 10""
KEY 11""
KEY 12""
KEY 13""
KEY 14""
KEY 15""
KEY 16""
```

2.3 AUTO

FORMATO: AUTO numero della riga di partenza, incremento

SCOPO: generare automaticamente la numerazione delle linee di programma con l'incremento specificato.

Introducendo il comando AUTO, viene visualizzato il numero della linea di programma che avete definito come linea di partenza, seguito

dal cursore, in attesa che introduciate una linea di codici. Ogni volta che battete una linea di codici e premete RETURN, il valore dell'incremento che avete specificato viene aggiunto al numero della linea di programma precedente. Il valore risultante rappresenta il numero della linea di programma successiva e viene visualizzato. Per disabilitare questa funzione, premete RETURN subito dopo che è stato visualizzato un nuovo numero di linea.

ESEMPIO: generare automaticamente la numerazione delle linee di programma, con incrementi di 5 ed iniziando dalla linea 10.

COMANDO: AUTO 10,5 < RETURN >

SCHERMO: 10

BATTERE: GET A$ < RETURN >

SCHERMO: 10 GET A$
15

BATTERE: IF A$ = "" THEN 10 < RETURN >

SCHERMO: 10 GET A$
15 IF A$ = "" THEN 10
20

RISULTATO: ogni volta che introducete una linea di programma e premete RETURN, viene visualizzato un nuovo numero di linea maggiore di 5 rispetto al precedente.

ESEMPIO: disabilitare la numerazione automatica delle linee di programma, nel programma scritto sopra.

BATTERE: < RETURN >

COMANDO; LIST < RETURN >

VIDEO: 10 GET A$
15 IF A$ = "" THEN 10
READY

RISULTATO: la numerazione automatica delle linee è stata disabilitata.

2.4 RENUMBER

FORMATO: RENUMBER numero della riga di partenza, incremento

SCOPO: rinumerare automaticamnte tutte le linee di un programma.

Cambia automaticamente il numero di tutte le linee di programma. Secondo la nuova numerazione, il programma inizia dal numero che avete assegnato e continua con le linee successive, secondo l'incremento specificato. Questo comando si mostra particolarmente utile quando avete bisogno di spazio in un programma per inserire dei nuovi comandi.

NOTA

Il comando RENUMBER non effettua la rinumerazione delle istruzioni GOTO e GOSUB. Per evitare questo inconveniente il SIMONS' BASIC sostituisce queste due istruzioni con la programmazione strutturata. Fate riferimento al capitolo 9.

ESEMPIO: rinumerare tutte le linee del programma che segue.

INTRODURRE:
```
1 PRINT "< SHIFT CLR/HOME >"
2 FOR X = 1 TO 20
3 Z = RND(1) ★ 255
4 POKE 53280,Z
5 FOR Y = 1 TO 250: NEXTY,X
```

COMANDO: RENUMBER 100,10 < RETURN >

BATTERE: LIST < RETURN >

SCHERMO:
```
100 PRINT "< SHIFT CLR/HOME>"
110 FOR X = 1 TO 20
120 Z = RND(1) ★ 255
130 POKE 53280,Z
140 FOR Y = TO 250 : NEXTY,X
```

2.5 PAUSE

FORMATO: PAUSE "messaggio", numero di secondi

oppure: PAUSE numero di secondi

SCOPO: sospendere l'esecuzione del programma per un periodo di tempo fissato.

Provoca un'attesa nell'esecuzione del programma per un intervallo di tempo specificato misurato in secondi. Non sono accettate frazioni di secondi. Il comando PAUSE può essere usato in due modi diversi, con o senza la visualizzazione di un messaggio per l'operatore.

Se nel comando PAUSE si scrive un messaggio fra virgolette, lo stesso viene visualizzato per tutta la durata della pausa. Premendo in qualunque momento il tasto RETURN, si interrompe la pausa e viene ripresa l'esecuzione del programma.

Se dopo il codice PAUSE non è presente nessun messaggio, l'esecuzione del programma si interrompe finchè non è trascorso l'intervallo di tempo specificato.

ESEMPIO: interrompere l'esecuzione del programma per 10 secondi.

INTRODURRE:
```
100 PAUSE 10
```

RISULTATO: quando il programma giunge alla linea 100, si pone in stato di attesa per 10 secondi.

ESEMPIO: visualizzare un messaggio e interrompere l'esecuzione del programma per 1 minuto.

INTRODURRE:
```
100 PAUSE "PREMERE RETURN PER CONTINUARE",60
```

RISULTATO: quando viene eseguita questa linea di programma, viene visualizzato il messaggio PREMERE RETURN PER CONTINUARE ed il programma rimane in stato di attesa per 1 minuto, oppure fino a quando non si preme il tasto RETURN.

2.6 CGOTO

FORMAT: CGOTO espressione

oppure: CGOTO operando operatore variabile

SCOPO: calcolare il numero della linea alla quale deve passare l'esecuzione del programma.

Vi permette di saltare ad un numero variabile della linea di programma, determinato dal risultato di un calcolo.

ESEMPIO: saltare successivamente a cinque differenti linee di programma il cui numero è in funzione della variabile del loop.

INTRODURRE:
```
10 REM"★★★ ESEMPIO DI CGOTO ★★★
20 FOR I = 1 TO 5
30 CGOTO I ★ 10 + 40
40 END
50 PRINT"I = 1":NEXT
60 PRINT"I = 2":NEXT
70 PRINT"I = 3":NEXT
80 PRINT"I = 4":NEXT
90 PRINT"I = 5":NEXT
```

RISULTATO: ad ogni valore di I del loop, da 1 a 5, il numero di linea viene calcolato alla linea 30 e quindi si effettua il salto a quella linea che viene eseguita.

2.7 RESET

FORMATO: RESET numero di linea

SCOPO: spostare il puntatore di linea su una specificata linea di dati.

Nel BASIC standard, i dati vengono sempre letti in modo sequenziale, cioè il primo elemento dei dati viene utilizzato con la prima frase READ, il secondo elemento con la frase READ successiva, e così via. RESET vi permette di fissare la linea di programma, all'interno di un blocco di dati, dalla quale desiderate che inizi la lettura. Ciò è utile quando desiderate che la lettura dei dati non inizi dal primo elemento della serie oppure quando volete saltare un certo numero di elementi, fino ad un punto determinato.

ESEMPIO: visualizzare una serie di dati, in funzione di ciò che introduce l'utente.

INTRODURRE:
```
 10 REM"★★★ ESEMPIO DI RESET★★★
 20 PRINT"< SHIFT CLR/HOME >
 30 PRINT "QUALE GRUPPO?":PRINT:PRINT
 40 PRINT" 1) CANI","2) GATTI","3) UCCELLI","4) PESCI"
 50 INPUT A: IF A<0 OR A>5 THEN PRINT "< SHIFT CURSOR
UP >":GOTO 50
 60 IF A = 1 THEN RESET 100
 70 IF A = 2 THEN RESET 110
 80 IF A = 3 THEN RESET 120
 90 IF A = 4 THEN RESET 130
 95 FOR I = 1 TO 5
 97 READ A$:PRINT A$:NEXT I
 99 PAUSE 10:GOTO 20
100 DATA ALSAZIANO,BULL DOG,TERRIER,LABRADOR,SPANIEL
110 DATA PERSIANO,SORIANO,ANGORA,SIAMESE,BIRMANO
120 DATA PASSERO,STORNELLO,FRINGUELLO,CANARINO,COLOMBO
130 DATA TROTA,SALMONE,GHIOZZO,LUCCIO,CAVEDANO
```

BATTERE: RUN < RETURN >

INTRODURRE: 3 < RETURN >

SCHERMO: PASSERO
STORNELLO
FRINGUELLO
CANARINO
COLOMBO

RISULTATO: il programma legge cinque elementi dei dati a partire dalla linea di programma avente il numero correlato all'input effettuato dall'utente.

AZIONE: tenendo premuto il tasto RUN/STOP premere il tasto RESTORE.

RISULTATO: il programma si arresta.

2.8 MERGE

FORMATO: MERGE «nome del programma», numero della periferica

SCOPO: prelevare un programma precedentemente memorizzato in una periferica di memoria di massa e caricarlo nella memoria di lavoro del COMMODORE 64.

Il numero della periferica vi permette di identificare il dispositivo di memoria di massa in cui è contenuto il programma su cui volete effettuare il MERGE. Questo numero vale 1 per l'unità a cassette ed 8 per l'unità a floppy disk. Se non specificate il numero della periferica, il programma assume automaticamente il valore 1 (unità a cassette)

ATTENZIONE
IL PROGRAMMA SU CUI SI EFFETTUA IL MERGE VIENE CARICATO IN MEMORIA PARTENDO DALLA PRIMA POSIZIONE LIBERA DOPO PROGRAMMI EVENTUALMENTE GIÀ PRESENTI. PRIMA DI FAR GIRARE IL PROGRAMMA OCCORRE RINUMERARE LE LINEE CON UN RENUMBER (vedi paragrafo 2.4).

ESEMPIO: effettuare il MERGE di un programma su cassetta chiamato «SIMONS' BASIC1» con il programma presente nella memoria di lavoro.

AZIONE: scrivere un breve programma e memorizzarlo su cassetta con il nome «SIMONS' BASIC1»

COMANDO: NEW < RETURN >

AZIONE: scrivere un altro breve programma.

COMANDO: MERGE «SIMONS' BASIC1», 1 < RETURN >

SCHERMO: PRESS PLAY ON TYPE

AZIONE: premereil tasto PLAY dell'unità a cassette.

SCHERMO: LOADING SIMONS' BASIC1
READY

RISULTATO: i due programmi in BASIC sono stati uniti.

2.9 AUSILII PER LISTARE I PROGRAMMI

2.9.1 PAGE

FORMATO: PAGE n

SCOPO: dividere il listato di un programma in pagine, ognuna composta da n linee.

Vi permette di specificare il numero di linee dello schermo che desiderate occupare quando listate un programma. Dopo aver impostato il comando, premendo LIST otterrete sullo schermo la prima linea del programma. Ogni successiva sezione del programma di n linee viene visualizzata sullo schermo premendo il tasto RETURN. Ponendo il parametro n pari a zero, si ottiene nuovamente il listato del programma nel modo normale. Si noti che il parametro n indica il numero di linee dello schermo effettivamente occupate dal listato e non le linee di programma, ognuna delle quali può anche occupare più di una linea dello schermo. Se una linea di programma supera i limiti che avete fissato sullo schermo, nella posizione successiva di listato appare nuovamente la linea completa.

ESEMPIO: listare un programma utilizzando solo 5 linee dello schermo.

AZIONE: caricare dalla memoria di massa o introdurre da tastiera un programma con più di 10 linee.

COMANDO: PAGE 5 < RETURN >

BATTERE: LIST < RETURN >

RISULTATO: viene visualizzata la prima linea del programma

COMANDO: premendo il tasto RETURN

RISULTATO: vengono visualizzate le prime 5 linee del programma.

COMANDO: premere il tasto RETURN.

RISULTATO: vengono visualizzate sullo schermo 5 linee di programma successive

COMANDO: PAGE Ø < RETURN >

BATTERE: LIST < RETURN >

RISULTATO: il vostro programma viene listato normalmente

2.9.2 OPTION

FORMATO: OPTION n

SCOPO: evidenziare tutti i comandi in SIMONS' BASIC presenti nel listato di un programma.

OPTION, seguito dal parametro 10, provoca la visualizzazione in video inverso di tutti i comandi SIMONS' BASIC presenti nel listato di un programma, sia sullo schermo, che sulla stampante. Ponendo n diverso da 1Ø (tra Ø e 255), si esclude l'effetto del comando.

ATTENZIONE
LA STAMPA DI LISTATI CON IL COMANDO OPTION ABILITATO, PROVOCA UN INTENSO CONSUMO DEL NASTRO. PERTANTO SI RACCOMANDA DI LIMITARE LA STAMPA DI QUESTO TIPO DI LISTATI.

Si noti che alcuni comandi utilizzati nel prossimo esempio, non sono ancora stati descritti. Essi sono stati utilizzati semplicemente per illustrare meglio l'uso del comando OPTION.

ESEMPIO: evidenziare in video inverso tutti i comandi in SIMONS' BASIC presenti nel seguente programma.

```
1Ø HIRES Ø,1
2Ø CIRCLE 16Ø,12Ø,Ø,28,1ØØ
3Ø REC 16Ø,12Ø,16Ø,12Ø,Ø
4Ø PAUSE 1Ø
5Ø CSET Ø:END
```

COMANDO: OPTION 1Ø < RETURN >

BATTERE: LIST < RETURN >

RISULTATO: tutti i comandi SIMONS' BASIC presenti nel programma vengono visualizzati sullo schermo in video inverso nel listato.

COMANDO: OPTION Ø < RETURN >

BATTERE: LIST < RETURN >

RISULTATO: il programma viene nuovamente listato in modo normale, senza comandi in video inverso.

2.9.3 DELAY

FORMATO: DELAY n

SCOPO: variare la velocità di scroll durante la visualizzazione del listato di un programma.

Se tenete premuto il tasto SHIFT durante il listing di un programma, potete rallentare la velocità con cui il listato viene visualizzato sullo schermo. DELAY vi permette di variare la velocità di questo rallentamento. Il parametro n che segue il comando varia tra 1 e 255 e determina il valore della velocità. Un valore maggiore di n equivale ad uno scroll proporzionalmente più lento e viceversa.

ESEMPIO: listare un programma alla minima velocità possibile.

COMANDO: DELAY 255 < RETURN >

BATTERE: LIST < RETURN >

AZIONE: tenere premuto il tasto SHIFT.

RISULTATO: il listato del programma viene visualizzato carattere per carattere.

COMANDO: rilasciare il tasto SHIFT.

RISULTATO: il listato del programma viene visualizzato a velocità normale.

NOTA
Quando si effettua il listing di un qualunque programma BASIC con COMMODORE 64, è possibile arrestare lo scroll dello schermo tenendo premuto il tasto CTRL. Rilasciando tale tasto, lo scroll riprende normalmente.

2.10 FIND

FORMATO: FINDcodice comando

oppure: FINDstringa di caratteri

SCOPO: effettuare la ricerca in un programma BASIC di un codice di comando o di una stringa di caratteri dati e visualizzare il numero delle linee di programma dove essi compaiono.

FIND vi permette di localizzare dove appare un dato comando od una data stringa nell'ambito di un programma in BASIC.

Con questo comando potete visualizzare tutti i numeri delle linee che contengono il comando o la stringa. Notate che eventuali spazi tra il comando FIND ed i caratteri specificati o tra l'ultimo carattere ed il RETURN, sono considerati parte della stringa che state cercando. Per questo motivo dovete prestare attenzione a NON INTRODURRE spazi superflui tra il comando FIND e gli elementi che seguono.

ESEMPIO: trovare la stringa di caratteri ABCD nel programma che segue.

```
10 REM TROVA ABCD
20 REM STAMPA "ABCD" IN VERTICALE
30 PRINT "ABCD IN VERTICALE"
40 A$ = "ABCD"
50 FOR C = 1 TO LEN (A$)
60 PRINT MID$(A$,C,1):NEXT
70 REM ABCD STAMPATO
```

COMANDO: FIND "ABCD" < RETURN >

SCHERMO: 20 30 40

RISULTATO: vengono visualizzati tutti i numeri delle linee di programma che contengono la stringa ABCD racchiusa tra virgolette. Notate che i numeri delle linee 10 e 70 non sono visualizzati perchè la stringa ABCD compare in esse senza le virgolette.

2.11 AUSILII PER IL DEBUG DEL PROGRAMMA

2.11.1 TRACE

FORMATO: TRACE n

SCOPO: visualizzare il numero della linea di programma che sta per essere eseguita

TRACE deve essere introdotto pirma di lanciare il programma. Ponendo il parametro uguale a 10, quando lanciate il programma, appare nell'angolo in alto a destra dello schermo una finestra che visualizza i numeri delle linee via via che vengono eseguite. Nello stesso istante, potete vedere fino a sei linee di programma. Il formato della visualizzazione è:
(numero di linea). Le linee della finestra effettuano automaticamente uno scroll verticale dal basso verso l'alto, in modo che il numero dell'ultima linea di programma eseguita è sempre il primo numero in basso nella finestra. Tenendo premuto il tasto con il logo COMMODORE, è possibile fare eseguire il programma passo-passo. Ponendo il parametro n pari a 0, disabilitate il comando TRACE. Notate che il comando TRACE NON PUO' essere usato sullo schermo ad alta risoluzione.

ATTENZIONE
LA FINESTRA DI TRACE SI SOVRAPPONE A QUALUNQUE ALTRA INFORMAZIONE SCRITTA SU QUELLA PORZIONE DI SCHERMO. FATE PERCIÒ ATTENZIONE QUANDO USATE IL COMANDO TRACE CHE NELLA PORZIONE DI SCHERMO, IN CUI SI CREA LA FINESTRA NON VI SIA NESSUNA INFORMAZIONE.

ESEMPIO: visualizzare i numeri delle linee di programma durante l'esecuzione del seguente programma;

```
10 PRINT "< SHIFT CLR/HOME >"
20 FOR X = 65 TO 96
30 PRINT "< CLR/HOME >";CHR$(X)
40 FOR Z = 1 TO 250: NEXT Z
50 NEXT X
60 GOTO 20
```

COMANDO: TRACE 10 < RETURN >

AZIONE: tenere premuto il tasto con il logo COMMODORE.

BATTERE: RUN < RETURN >

RISULTATO: non appena viene eseguita una linea di programma, appare nella finestra il numero relativo.

AZIONE: quando il programma è terminato, lasciare il tasto con il logo COMMODORE.

BATTERE: RUN < RETURN >

RISULTATO: la finestra scompare ed il programma viene eseguito normalmente.

2.11.2 RETRACE

FORMAT: RETRACE

SCOPO: per riabilitare il comando TRACE dopo aver effettuato l'editing di un programma.

Quando state utilizzando il comando TRACE, se arrestate l'esecuzione del programma e cancellate lo schermo, la finestra di TRACE scompare. Il comando RETRACE attiva nuovamente TRACE e visualizza nella finestra l'ultimo gruppo di numeri delle linee che erano state eseguite immediatamente prima dell'arresto del programma. Quando il programma viene nuovamente lanciato, ricompare la normale finestra di TRACE. L'esecuzione del programma non riprende dal punto in cui era stato arrestato, ma dall'inizio del programma stesso. Si noti che il comando RETRACE non può essere usato se il comando TRACE è stato disattivato.

ESEMPIO: utilizzare il programma del paragrafo precedente, cancellare lo schermo e rilanciare il programma con TRACE.

COMANDO: TRACE 10 < RETURN >

BATTERE: RUN < RETURN >

RISULTATO: il numero delle linee di programma che via via sono eseguite compare nella finestra di TRACE.

AZIONE: premete il tasto RUN/STOP

BATTERE: LIST < RETURN >

INTRODURRE: 30 PRINT "< CLR/HOME >";CHR$(X),X

AZIONE: tenete premuto il tasto SHIFT e premete contemporaneamente il tasto CLR/HOME

RISULTATO: lo schermo viene cancellato

COMANDO: RETRACE < RETURN >

RISULTATO: la finestra nell'angolo in alto a destra dello schermo ricompare e visualizza i numeri delle linee che erano presenti sullo schermo nel momento in cui si era arrestato il programma.

AZIONE: portare il cursore in fondo alla finestra.

BATTERE: RUN < RETURN >

RISULTATO: il programma viene lanciato e i numeri di linea vengono nuovamente visualizzati nella finestra di TRACE.

2.12 DUMP

FORMATO: DUMP

SCOPO: visualizzare i valori di tutte le variabili che non sono matrici.

DUMP vi permette di visualizzare i valori di tutte le variabili, tranne di quelle contenute in matrici. I valori mostrati sono quelli assunti dalle variabili quando l'esecuzione del programma viene arrestata sia premendo il tasto RUN/STOP, che raggiungendo un terminatore nel programma stesso. Le variabili vengono elencate nello stesso ordine in cui sono state definite nel programma e sono visualizzate nel formato:

nome della variabile = valore

NOTA

Se nel vostro programma sono presenti più di 25 variabili, per impedire lo scroll sullo schermo, tenete premuto il tasto CTRL. Per continuare la visualizzazione dell'elenco rilasciate il tasto.

ESEMPIO: visualizzare le variabili nel seguente programma:

```
10 A$ = "COLORI CASUALI"
20 PRINT "SHIFT/CLR/HOME",A$
30 X = INT(RND(8) * 15)
60 POKE 53281,X
70 FOR C = 1 TO 100:NEXT C
80 GOTO 30
```

BATTERE: RUN < RETURN >

AZIONE: dopo che lo schermo ha cambiato colore per alcune volte, premete il tasto RUN/STOP e quindi premete il tasto RESTORE.

COMANDO: DUMP < RETURN >

SCHERMO: A$ = "COLORI CASUALI"
X = 9
C = 80

Notate che poichè i valori delle variabili X e C sono generati in modo casuale, i numeri visualizzati dipendono dal punto in cui è stata arrestata l'esecuzione del programma.

2.13 COLD

FORMATO: COLD

SCOPO: riportare il COMMODORE 64 all'inizio del SIMONS' BASIC

COLD cancella qualunque programma contenuto nella memoria del COMMODORE 64 e visualizza lo schermo che appare quando accendete il vostro computer con inserita la cartuccia del SIMONS' BASIC.

ATTENZIONE
QUANDO USATE IL COMANDO COLD, VIENE CANCELLATO QUALUNQUE PROGRAMMA PRESENTE NELLA MEMORIA DEL COMPUTER. A QUESTO PUNTO POTETE ANCORA RICHIAMARE UN EVENTUALE PROGRAMMA CANCELLATO PER ERRORE CON IL COMANDO OLD (FATE RIFERIMENTO AL PARAGRAFO 2.15). SE PERÒ AVETE GIÀ INSERITO UNA PARTE QUALUNQUE DEL NUOVO PROGRAMMA, NON È PIÙ POSSIBILE RICHIAMARE QUELLI VECCHI.

ESEMPIO: riportare il COMMODORE 64 alle condizioni iniziali del SIMONS' BASIC.

ESEMPIO: riportare il COMMODORE 64 alle condizioni iniziali del SIMONS' BASIC

COMANDO: COLD < RETURN >

RISULTATO: viene visualizzato lo schermo iniziale del SIMONS' BASIC.

2.14 AUSILII PER LA SICUREZZA DEI PROGRAMMI

2.14.1 INTRODUZIONE

Il SIMONS' BASIC mette a vostra disposizione due comandi che vi permettono di nascondere delle linee specificate nel listato del programma, in modo da impedirne l'esame a persone non autorizzate. Il comando DISAPA vi permette di indicare quali linee di programma volete nascondere, mentre il comando SECURE effettua la cancellazione delle stesse dal listato. Questi comandi sono particolarmente utili per cancellare parole d'ordine, numeri di serie, ecc.

ATTENZIONE
NON ESISTE ALCUN MODO PER ANNULLARE L'EFFETTO DI QUESTI COMANDI, SE NON QUELLO DI RISCRIVERE LE LINEE NASCOSTE. PER TANTO, PRIMA DI USARE QUESTO COMANDO, È BENE FARE UN DUPLICATO DEL PROGRAMMA ORIGINALE PER USO PERSONALE.

2.14.2 DISAPA

FORMATO: DISAPA

SCOPO: indicare che determinate linee di programma debbono essere nascoste nel relativo listato.

DISAPA deve essere utilizzato come primo comando su una linea di programma e specifica che i codici presenti su quella stessa linea debbono essere nascosti. Il comando SECURE (paragrafo seguente) provoca quindi la sparizione di dette linee. Il comando DISAPA pone automaticamente 4 caratteri (:) prima del codice, in ogni linea in cui appare.

Notate che è necessario prevedere nella linea lo spazio relativo a questi 3 caratteri, e ciò significa che il campo massimo che può essere nascosto è di 30 caratteri (escluso il comando DISAPA ed i caratteri :).

ESEMPIO: indicare che i codici nelle linee 10 e 40 del seguente programma debbono essere nascosti.

```
10 PRINT "CIAO"
20 PRINT SA
30 SA = SA + 1
40 GOTO 10
```

INTRODUZIONE:
```
10 DISAPA: PRINT "CIAO"
40 DISAPA: GOTO 10
```

COMANDO: LIST < RETURN >

SCHERMO:

```
10 DISAPA: : : : : PRINT "CIAO"
20 PRINT SA
30 SA = SA + 1
40 DISAPA: : : : : GOTO 10
```

RISULTATO: quando usate il comando SECURE, (paragrafo seguente) le linee di programma contenenti DISAPA vengono nascoste.

2.14.3 SECURE

FORMATO: SECURE Ø

SCOPO: nascondere tutte le linee di programma che iniziano con il comando DISAPA.

SECURE impedisce la scrittura nel listato di tutte le linee di programma che iniziano con DISAPA (paragrafo precedente). Il comando nascosto viene comunque eseguito normalmente.

ESEMPIO: nascondere le linee 10 e 40 nel programma esempio scritto nel paragrafo precedente.

COMANDO: SECURE Ø < RETURN >

BATTERE: LIST < RETURN >

SCHERMO:

```
10
20 PRINT SA
30 SA = SA + 1
40
```

RISULTATO: quando effettuate il listato del programma, le linee 10 e 40 appaiono senza codici. Sebbene essi non siano visualizzati, i numeri delle linee compaiono ed il programma funzionale normalmente.

2.15 OLD

SCOPO: ottenere l'effetto opposto a quello del comando NEW.

OLD vi permette di richiamare ed eseguire nuovamente un programma che era stato apparentemente cancellato dalla memoria con il comando NEW. Il comando non richiede parametri. (Da un punto di vista più tecnico, il comando OLD resetta i puntatori di pagina-zero all'inizio ed alla fine del BASIC).

ESEMPIO: cancellare con NEW il programma seguente e quindi richiamarlo.

```
10 REM COMANDO OLD
20 A$ = "COMMODORE 64"
30 FOR C = 1 TO LEN(A$)
40 PRINT"< CLR/HOME >              ",LEFT$ (A$,C)
50 FOR X = 1 TO 100:NEXT X,C
```

COMANDO: NEW < RETURN >

BATTERE: LIST < RETURN >

SCHERMO: READY

COMANDO: OLD < RETURN >

BATTERE: LIST < RETURN >

SCHERMO:

```
10 REM COMANDO OLD
20 A$ = "COMMODORE 64"
30 FOR C = 1 TO LEN(A$)
40 PRINT"< CLR/HOME >              ",LEFT$ (A$,C)
50 FOR X = 1 TO 100;NEXT X,C
```

CAPITOLO 3
CONFERMA DELL'INPUT E GESTIONE DEI TESTI

3.1 INTRODUZIONE

Il SIMONS' BASIC mette a vostra disposizione una serie completa di comandi per il trattamento delle stringhe di caratteri, la formattazione dello schermo e la conferma dell'input.

Il comando INSERT vi permette di creare una stringa di caratteri più lunga, inserendo una stringa in un'altra. INST vi permette di sovrapporre ad una stringa un'altra, a partire da un carattere specificato. Il comando PLACE vi permette di determinare la posizione di un gruppo di caratteri all'interno di una stringa. DUP vi permette di ottenere una stringa dalla duplicazione di una stringa più breve, il numero desiderato di volte.

Il comando LIN visualizza il numero della riga dello schermo su cui è posizionato il cursore. CENTRE vi permette di centrare un testo su una linea dello schermo. PRINT AT vi permette di specificare in quale punto dello schermo deve essere visualizzato un testo. USE vi permette di incolonnare dati numerici.

Il comando FETCH vi permette di impostare i parametri per l'input dell'utente. INKEY vi permette di verificare quali tasti funzione sono stati premuti. ON KEY trasferisce l'esecuzione del programma ad una linea che dipende da ciò che viene introdotto da tastiera. DISABLE disabilita questo comando, mentre RESUME lo riabilita.

Utilizzati assieme ai comandi per il trattamento delle stringhe del BASIC standard del COMMODORE 64, queste caratteristiche vi assicurano la completa gestione delle stringhe di testo. Tutti i comandi descritti in questo capitolo possono essere usati sia in modo diretto, che indiretto.

3.2 TRATTAMENTO DI STRINGHE DI CARATTERI

3.2.1 INSERT

FORMATO: INSERT ("sotto-stringa", "stringa principale", p)

SCOPO: inserire una stringa di caratteri in un'altra.

INSERT vi permette di porre un gruppo di caratteri (sotto-stringa) nell'interno di un'altra stringa di caratteri (stringa principale), creando così una stringa più lunga.

Il parametro p indica la posizione del carattere, nella stringa principale, DOPO il quale desiderate inserire la sotto-stringa. La sotto-stringa e la stringa principale possono essere una qualunque espressione racchiusa tra virgolette, oppure delle variabili di stringa, cioè ad esempio "aaaaa" oppure a$. La massima lunghezza della nuova stringa ottenuta con l'operazione INSERT non può essere superiore a 255 caratteri.

Il comando INSERT può anche essere usato per confrontare due stringhe di caratteri con un'operazione logica di tipo vero/falso. Ciò significa che il risultato di questa operazione di confronto sarà 1 nel caso di vero e 0 nel caso di falso.

Se questo comando viene usato in modo non corretto, possono essere generati due tipi di errore con relativo messaggio. Il primo di essi consiste in:

? INSERT PARAMETER TOO LARGE

e si ha quando la posizione specificata del punto di inserimento, con il carattere p, è maggiore della lunghezza della stringa principale.

Il secondo consiste invece nel messaggio:

? CREATED STRING TOO LONG

e si ha quando la nuova stringa creata con l'operazione INSERT è più lunga di 255 caratteri, che costituisce la massima lunghezza che può essere trattata dal BASIC.

ESEMPIO: inserire la parola "DISSE" nella stringa di caratteri "EGLI CIAO".

INTRODUZIONE:
```
100 PRINT INSERT ("DISSE","EGLI CIAO", 5)
```

BATTERE: RUN < RETURN >

SCHERMO: EGLI DISSE CIAO

RISULTATO: la sotto-stringa "DISSE" è stata inserita nella stringa principale "EGLI CIAO", a partire dal sesto carattere.

ESEMPIO: creare una variabile di stringa più lunga.

INTRODURRE:
```
100 B$ = "DISSE "
105 A$ = INSERT (B$,"EGLI CIAO", 5)
110 PRINT A$
```

BATTERE: RUN < RETURN >

SCHERMO: EGLI DISSE CIAO

BATTERE: DUMP < RETURN >

SCHERMO: B$ = "DISSE"
A$ = "EGLI DISSE CIAO"

ESEMPIO: confrontare due stringhe di caratteri

INTRODURRE:
```
100 A= (INSERT("DISSE","EGLI CIAO", 5) = "EGLI DISSE CIAO")
110 PRINT A
```

BATTERE: RUN < RETURN >

SCHERMO: −1

RISULTATO: poichè le due stringhe confrontate sono identiche, e quindi la condizione è vera, il risultato dell'operazione è −1, valore che viene visualizzato. Se invece la condizione fosse stata falsa, sarebbe stato visualizzato un valore pari 0.

3.2.2 INST

FORMATO: INST ("sotto-stringa", "stringa-principale", p)

SCOPO: sovrapporre una stringa (principale) con un'altra (sotto-stringa) a partire da un carattere di posizione specificata (con p).

INST sostituisce una stringa di caratteri con un'altra stringa che si sovrappone alla stringa principale, a partire dalla posizione specificata. Sia la sotto-stringa che la stringa principale possono essere una qualunque espressione, ad esempio "aaaaa" oppure XX$. Il valore di p indica la posizione DOPO la quale la sotto-stringa sostituisce i caratteri della stringa principale.

Se questo comando viene utilizzato in modo non corretto, può essere generato un tipo di errore con relativo messaggio, che consiste nella frase:

? CREATED STRING TOO LONG

e si ha quando la nuova stringa creata con l'operazione INST è la più lunga di 255 caratteri.

ESEMPIO: sostituire la parola "BUONO" con la parola "MIGLIORE" nella frase "EGLI FU BUONO"

INTRODURRE:
```
5 A$ = "EGLI FU BUONO"
10 A$ = INST("MIGLIORE",A$,8)
20 PRINT A$
```

BATTERE: RUN < RETURN >

SCHERMO: EGLI FU MIGLIORE

COMANDO: DUMP < RETURN >

SCHERMO: A$ = "EGLI FU MIGLIORE"

3.2.3 PLACE

FORMATO: PRINT PLACE("sotto-stringa", "stringa principale")

SCOPO: determinare la posizione di una sotto-stringa all'interno della stringa principale.

PLACE ricerca un gruppo di caratteri specificato (sotto-stringa) all'interno di una stringa di caratteri. Se questo gruppo viene trovato, viene visualizzata la posizione del primo carattere dello stesso nella stringa principale. Se, invece il gruppo non è trovato, viene visualizzato un valore pari a zero. La lunghezza della sotto-stringa deve sempre essere inferiore a quella della stringa principale. Questo comando permette anche di confrontare due variabili numeriche.

ESEMPIO: determinare la posizione della sotto-stringa "MIGLIORE" nella stringa principale "EGLI FU MIGLIORE".

INTRODURRE:
```
10 A$ = INST("MIGLIORE","EGLI FU BUONO",7)
20 PRINT PLACE ("MIGLIORE",A$)
```

BATTERE: RUN < RETURN >

SCHERMO: 8

ESEMPIO: ecco un semplice esempio di testi di grammatica:

INTRODURRE:
```
10 PRINT "INTRODURRE LA POSIZIONE DEL PRIMO CARATTERE
   DELL'AVVERBIO NELLA SEGUENTE FRASE:":PRINT
20 PAUSE 1
30 A$ = "EGLI ENTRÒ IN CASA RUMOROSAMENTE"
40 B = PLACE("RUMOROSAMENTE",A$):B$ =
   "RUMOROSAMENTE":PRINT A$
50 INPUT A
60 IF A = B THEN 80
70 PRINT"SBAGLIATO":PRINT"LA RISPOSTA ESATTA È" B
75 PRINT"L'AVVERBIO È";B$:END
80 PRINT"RISPOSTA ESATTA":END
```

BATTERE: RUN < RETURN >

SCHERMO: INTRODURRE LA POSIZIONE DEL PRIMO CARATTERE DELL'AVVERBIO NELLA SEGUENTE FRASE:

EGLI ENTRÒ IN CASA RUMOROSAMENTE

BATTERE: 8 < RETURN >

SCHERMO: SBAGLIATO
LA RISPOSTA ESATTA È 21
L'AVVERBIO È RUMOROSAMENTE

3.2.4 DUP

FORMATO: DUP ("stringa",n)

SCOPO: duplicare n volte una stringa di caratteri.

DUP vi permette di ottenere una nuova stringa di caratteri duplicando n volte una stringa già esistente. Il parametro n indica appunto il n di volte che deve essere riprodotta la vecchia stringa.

Si noti che, se la nuova stringa creata è maggiore di 255 caratteri, viene visualizzato il seguente messaggio di errore:

? CREATED STRING TOO LONG

ESEMPIO: duplicare tre volte una stringa di caratteri e quindi aggiungere un'altra stringa.

INTRODURRE:
```
10 A$ = DUP ("CIAO-",3)
20 B$ = "CHI STA ARRIVANDO ?"
30 C$ = A$+B$:PRINT C$
```

BATTERE: RUN < RETURN >

SCHERMO: CIAO-CIAO-CIAO-CHI STA ARRIVANDO?

3.2.5 CENTRE

FORMATO: CENTRE "stringa di caratteri"

SCOPO: centrare una stringa di caratteri su una linea dello schermo

CENTRE vi permette di visualizzare una stringa di testo nel centro di una linea sullo schermo. Per usare questo comando, non è necessario conoscere la lunghezza del testo.

ESEMPIO: centrare la stringa "COMMODORE 64"

COMANDO: CENTRE "COMMODORE 64" < RETURN >

SCHERMO: COMMODORE 64

3.2.6 AT

FORMATO: PRINT AT (c,r) "stringa di caratteri"

oppure: PRINT "prima stringa"AT(c,r)"seconda stringa"

SCOPO: visualizzare una stringa di caratteri in un determinato punto dello schermo.

AT vi permette di specificare il punto dello schermo dove volete che inizi la visualizzazione di una stringa di caratteri. Il comando costituisce un'alternativa all'uso dei caratteri di controllo del cursore per posizionare il testo. I parametri c ed r definiscono rispettivamente le coordinate della colonna e della riga dove deve iniziare la stringa di caratteri che segue i parametri stessi. In una singola frase è possibile combinare anche più di un comando AT.

ESEMPIO: posizionare la stringa "COMMODORE 64" alla colonna 13, riga 8.

COMANDO: PRINT AT(13,8)"COMMODORE 64" < RETURN >

SCHERMO: mostrato in fig. 3-1.

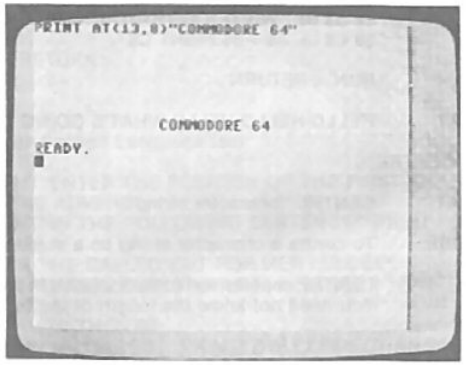

FIGURA 3-1: EFFETTO DI UN COMANDO 'AT' SINGOLO

ESEMPIO: stampare la stringa di caratteri "CBM 64" partendo dalla colonna 13, riga 8 e la stringa "SIMONS' BASIC" due linee al di sotto e 3 caratteri a destra della prima stringa.

COMANDO: PRINT AT(13,8)"CBM 64"AT(15,11)"SIMONS' BASIC" < RETURN >

SCHERMO: mostrato in fig. 3-2

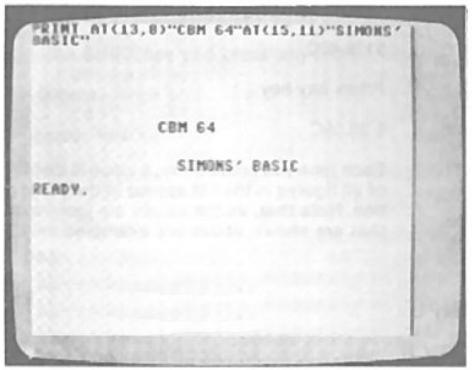

FIGURA 3-2: EFFETTO DI UN COMANDO 'AT' COMPOSTO

3.2.7 USE

FORMATO: USE " # # # . # # # # ",vs:PRINT

oppure: USE " # # testo . # # # testo",vs:PRINT

SCOPO: formattare un dato numerico.

USE vi permette di formattare gli elementi di un elenco di numeri, ad esempio, per allineare i punti decimali. La quantità dei simboli # prima e dopo il punto decimale indicano al COMMODORE 64 il modo in cui visualizzare il il numero corrispondente di caratteri dalla stringa ralativa a questa posizione. È anche possibile inserire un testo tra i simboli. Il parametro vs è la rappresentazione sotto forma di stringa del numero che utilizzate con USE. Notate che PRINT deve seguire la stringa, perchè il comando USE non forza un RETURN.

ESEMPIO: stampare un tabulato di prezzi generato in modo casuale.

INTRODURRE:
```
10 REM"★★★ ESEMPIO ★★★
20 A$ = STR$(RND(1)★199)
30 USE"$ # # # . # # C",A$:PRINT
40 GET A$:IF A$ = "" THEN 40
50 GOTO 20
```

BATTERE: RUN < RETURN >

AZIONE: premere un tasto qualunque

SCHERMO: $ 126.45C

AZIONE: premere un tasto qualunque

SCHERMO: $ 35.36C

RISULTATO: ogni volta che premete un tasto qualunque, viene visualizzato un valore. Il punto decimale di ogni numero rappresentato nel tabulato compare sempre nella stessa posizione dello schermo.

3.3 CONFERMA DELL'INPUT

3.3.1 FETCH

FORMATO: FETCH "carattere di controllo", I, stringa definita come valida

SCOPO: FETCH vi permette di controllare quali caratteri sono accettati come input da tastiera. Il carattere di controllo tra virgolette determina quale tipo di caratteri è accettato. Ecco i tipi di caratteri validi ed i rispettivi caratteri di controllo:

CARATTERE DI CONTROLLO:	CARATTERI VALIDI
CLR/HOME	caratteri alfabetici non shiftati
CURSOR DOWN	caratteri numerici
CURSOR LEFT	caratteri numerici e shiftati

Il parametro I del comando FETCH è un numero che specifica la massima quantità di caratteri che possono essere introdotti dall'utente. Il terzo parametro specifica la variabile di stringa nella quale deve essere inserito l'input.

ESEMPIO: limitare l'input ad un massimo di 8 caratteri alfabetici non shiftati e porlo nella variabile stringa A$.

INTRODURRE:
```
10 PRINT:PRINT"COME TI CHIAMI?"
20 FETCH:"< CLR HOME >",8,A$
30 PRINT"CIAO "A$
```

BATTERE: RUN < RETURN >

SCHERMO: COME TI CHIAMI?
(cursore)

AZIONE: tenere premuto il tasto SHIFT e premere una lettera qualunque

RISULTATO: non accade nulla

AZIONE: premere un tasto numerico

RISULTATO: ancora non accade nulla

BATTERE: MIKE < RETURN >

COMANDO: DUMP < RETURN >

SCHERMO: A$ = "MIKE"

RISULTATO: solo una stringa con un massimo di 8 caratteri alfabetici non shiftati è accettata come input per A$

3.3.2 INKEY

FORMATO: INKEY

SCOPO: verificare quale tasto funzione è stato premuto.

INKEY permette di determinare quale tasto funzione è stato premuto, visualizzando il numero relativo da 1 a 16. Questo comando è particolarmente utile in programmi pilotati da menu, dove i tasti funzione permettono di scegliere delle opzioni o di effettuare determinate operazioni.

ESEMPIO: verificare se vengono premuti i tasti funzioni F1 ed F2

INTRODURRE:
```
10 A = INKEY
20 ON A GOSUB 1000,2000
30 GOTO 10
1000 PRINT"HAI PREMUTO F1":RETURN
2000 PRINT"HAI PREMUTO F2":RETURN
```

BATTERE: RUN < RETURN >

AZIONE: premere il tasto funzione F1

SCHERMO: HAI PREMUTO F1

AZIONE: tenere premuto il tasto SHIFT e quindi premere il tasto funzione F1 (=F2)

SCHERMO: HAI PREMUTO F2

3.3.3 ON KEY

FORMATO: ON KEY "carattere (i)":GOTO numero linea

SCOPO: saltare ad uno specificato punto del programma.

ON KEY ordina al COMMODORE 64 di effettuare la scansione della tastiera per cercare l'input di uno dei caratteri definiti nel comando. Qualunque tasto non definito viene ignorato. Dopo aver riconosciuto un carattere valido, l'esecuzione del programma passa alla linea specificata nel GOTO. La variabile riservata ST mantiene il valore CHR$ del testo che è stato premuto (nella "Guida per l'utente", in dotazione al COMMODORE 64, potete trovare l'elenco completo dei codici CHR$). Questo comando è particolarmente utile nei programmi pilotati da menu.

NOTA
Quando viene eseguito un comando ON KEY, il COMMODORE 64 continua ad effettuare la scansione della tastiera, anche dopo che è stato introdotto uno dei caratteri accettati. Pertanto, per disabilitare il comando ON KEY, dovete utilizzare il comando DISABLE (paragrafo seguente).

ESEMPIO: definire una serie di caratteri validi per l'input

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME > PREMI UN TASTO (E TO END)"
20 B$ = "DGHNVMLPOE"
×0 ON KEY B$,:GOTO 50
40 GOTO 20
```

RISULTATO: quando gira questa parte del programma, la sua esecuzione si arresta in attesa dell'introduzione di uno dei caratteri definiti come accettabili.

3.3.4 DISABLE

SCOPO: disabilitare il comando ON KEY

DISABLE vi permette di disattivare la scansione della tastiera attivata con il comando ON KEY (paragrafo precedente). Se è stato utilizzato il comando ON KEY, il comando DISABLE deve SEMPRE essere utilizzato. In caso contrario, si ottengono dei salti ricorsivi, cioè, ogni volta che viene introdotto uno dei caratteri definiti come accettabili dal comando ON KEY, l'esecuzione del programma continua a saltare alla linea di programma indicata dal GOTO contenuto nel comando ON KEY.

ESEMPIO: disabilitare il comando ON KEY

ESEMPIO: disabilitare il comando ON KEY

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME > PREMI UN TASTO (F = FINE)"
20 B$ = "DGHNVMLPOF"
30 ON KEY B$,:GOTO 50
40 GOTO 20
50 DISABLE
```

RISULTATO: dopo che questa parte del programma è stata eseguita, il comando ON KEY viene disattivato introducendo uno dei caratteri definiti come validi.

3.3.5 RESUME

FORMATO: RESUME

SCOPO: riattivare un comando ON KEY precedentemente attivato e quindi disabilitato.

RESUME vi permette di riattivare l'ultimo comando ON KEY definito e disattivato. In tal modo, il programma pone il computer in uno stato di attesa, fino a quando viene introdotto uno dei caratteri definiti nel comando ON KEY.

ESEMPIO: continuare l'esempio del paragrafo precedente per riattivare il comando ON KEY.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME > PREMI UN TASTO (E TO END)"
20 B$ = "DGHNVMLPOE"
30 ON KEY B$,:GOTO 50
40 GOTO 30
50 DISABLE
60 A$ = CHR$(ST):X = PLACE(A$,B$)
70 ON X GOTO 80,90,100,110,120,130, 140,150,160,170
80 PRINT"HAI BATTUTO D":RESUME
90 PRINT"HAI BATTUTO G":RESUME
100 PRINT"HAI BATTUTO H":RESUME
110 PRINT"HAI BATTUTO N":RESUME
120 PRINT"HAI BATTUTO V":RESUME
130 PRINT"HAI BATTUTO M":RESUME
140 PRINT"HAI BATTUTO L":RESUME
150 PRINT"HAI BATTUTO P":RESUME
160 PRINT"HAI BATTUTO O":RESUME
170 PRINT"HAI BATTUTO F":FINE
```


BATTERE: RUN < RETURN >

BATTERE: V < RETURN >

SCHERMO: HAI BATTUTO V

BATTERE: X < RETURN >

RISULTATO: non viene visualizzato nulla

BATTERE: E < RETURN >

SCHERMO:
```
HAI BATTUTO E
READY
```

RISULTATO: introducendo uno dei caratteri compresi tra quelli definiti con il comando ON KEY, viene visualizzato il messaggio relativo al carattere stesso. Battendo un qualunque altro carattere, lo stesso è ignorato e non appare nulla sullo schermo.

CAPITOLO 4
AUSILII NUMERICI

4.1 INTRODUZIONE

In questo capitolo sono descritti alcuni comandi che vi aiuteranno nel trattamento di dati numerici. I comandi MOD e DIV vi permetteranno di effettuare la divisione intera su dei numeri positivi. Tutti i risultati sono presentati arrotondati. Il comando FRAC vi permette di estrarre la parte frazionaria di un numero. Inoltre, sono descritti in questo capitolo anche i comandi per convertire un numero in base esadecimale o in base binaria, nell'equivalente numero in base decimale. Un operatore logico booleano, l'or esclusivo (EXOR) completa i comandi descritti in questo capitolo.

4.2 OPERATORI ARITMETICI SUPPLEMENTARI

4.2.1 MOD

FORMATO: MOD (x,y)

SCOPO: calcolare e visualizzare il resto della divisione di un numero intero per un altro numero intero.

MOD dà come risultato il resto della divisione di un numero intero per un altro numero intero. Il comando MOD può essere usato sia direttamente da tastiera che da programma.

ESEMPIO: dividere 15 per 4 e visualizzare il resto.

BATTERE: PRINT MOD(15,4) < RETURN >

SCHERMO: 3

4.2.2 DIV

FORMATO: DIV(x,y)

SCOPO: visualizzare il massimo numero intero che, moltiplicato per y, dà un numero ugualeo minore ad x.

DIV vi permette di dividere un numero floating-point per un altro e di visualizzare il risultato in formato intero troncato. Ciò significa che la parte frazionaria del risultato viene ignorata.

ESEMPIO: dividere 1Ø per 3 e presentare il risultato in forma intera.

INTRODURRE: 1Ø A = DIV (1Ø,3)
2Ø PRINT A

BATTERE: RUN < RETURN >

SCHERMO: 3

4.2.3 FRAC

FORMATO: FRAC(n)

SCOPO: estrarre e visualizzare la parte frazionaria di un numero.

FRAC vi permette di estrarre la parte frazionaria di un numero floating-point; in sostanza viene presentata la parte del numero che segue il punto decimale, fino ad un massimo di 9 cifre decimali.

ESEMPIO: dividere 22 per 6 e visualizzare la parte frazionaria del risultato.

COMANDO: PRINT FRAC(22/6) < RETURN >

SCHERMO: .666666667

ESEMPIO: visualizzare la parte frazionaria di p-greco.

INTRODURRE: 1Ø PRINT π

BATTERE: RUN < RETURN >

SCHERMO: 3.14159265
.141592653

4.3 CONVERSIONE DI BASE NUMERICA

4.3.1 CONVERSIONE DA BASE BINARIA A BASE DECIMALE

FORMAT: PRINT % numero in base binaria

SCOPO: convertire un numero in base binaria in un numero in base decimale.

Il comando % converte un numero in base binaria nel suo equivalente in base decimale.

Se come argomento del comando non viene utilizzato un numero in base binaria, viene visualizzato il messaggio di errore:

? NOT BINARY CHARACTER

ESEMPIO: convertire il numero binario 10110101 nel suo equivalente in base 10.

COMANDO: PRINT %1Ø11Ø1Ø < RETURN >

SCHERMO: 181

4.3.2 CONVERSIONE DA BASE ESADECIMALE A BASE DECIMALE

FORMATO: PRINT $ numero in base esadecimale

SCOPO: convertire un numero in base esadecimale in un numero in base decimale.

Il comando $ converte un numero in base esadecimale nel suo equivalente in base decimale.

Se come argomento del comando non viene utilizzato un numero in base esadecimale, viene visualizzato il messaggio di errore:

? NON HEX CHARACTER

ESEMPIO: convertire il numero esadecimale EB38 nel suo equivalente in base 10.

COMANDO: PRINT $EB38 < RETURN >

SCHERMO: 6Ø216

4.3.3 COMBINAZIONE DEI COMANDI DI CONVERSIONE

I due comandi di conversione visto possono essere utilizzati nell'ambito di un'unica espressione.

ESEMPIO: sommare un numero in base binaria ad un numero in base esadecimale e presentare il risultato in base decimale.

COMANDO: PRINT %1Ø11Ø1Ø1 + $EB38 < RETURN >

SCHERMO: 6Ø397

4.4 EXOR

FORMATO: EXOR(n,n1)

SCOPO: effettuare l'operazione logica OR esclusivo tra due numeri

EXOR vi permette di effettuare l'OR esclusivo tra due numeri in base binaria e quindi li confronta, bit per bit. Se due bit corrispondenti sono uguali, il risultato è posto pari a Ø, se sono diversi è posto pari ad 1.

ESEMPIO: porre in OR esclusivo i numeri 87 e 45.

COMANDO: PRINT EXOR(87,45)

SCHERMO: 122

La routine usata per arrivare a questo risultato è mostrata qui sotto:

primo numero: 87 = Ø1Ø1Ø111
secondo numero: 45 = ØØ1Ø11Ø1

risultato = Ø1111Ø1Ø = 122

ESEMPIO: visualizzare sullo schermo dei caratteri in video inverso.

INTRODURRE:
```
 5 PRINT "< SHIFT CLR/HOME >"
 7 FOR C = 1 TO 1Ø
 8 PRINT "SIMONS' BASIC":PRINT:NEXT
1Ø FOR X = Ø TO 999
2Ø A = PEEK(1Ø24 + X)
3Ø IF A = 32 THEN 6Ø
4Ø K = EXOR(A,128)
5Ø POKE 1Ø24 + X,K
6Ø NEXT:GOTO 1Ø
```

BATTERE: RUN < RETURN >

RISULTATO: ogni carattere presente sullo schermo viene cambiato in video inverso e quindi, nuovamente in modo normale.

CAPITOLO 5
GESTIONE DELL'UNITÀ A FLOPPY DISK

5.1 INTRODUZIONE

Il SIMONS' BASIC è dotato di due comandi, DISK e DIR, che vi permettono di effettuare in modo più semplice e diretto la gestione dei floppy disk. Il comando DISK elimina la necessità di specificare il numero delle file, il numero del dispositivo e l'indirizzo secondario, durante l'apertura di un canale per l'unità a floppy disk. Inoltre, il comando chiude automaticamente il canale quando l'operazione è stata completata. Il comando DIR sostituisce il comando del BASIC standard LOAD"$",8 e vi permette di listare, con un solo comando, il file directory, completo o in parte, del floppy in uso.

5.2 DISK

FORMATO: DISK,"operazione"

SCOPO: aprire un canale per il floppy e quindi richiuderlo ad operazione completa.
DISK sostituisce i seguenti codici comando del BASIC standard: OPEN file logico, numero del dispositivo, indirizzo secondario PRINT # file logico.

Il comando apre un canale sull'unità a floppy disk e quindi lo richiude automaticamente quando l'operazione specificata è stata completata.

ESEMPIO: formattare un nuovo floppy disk ed intitolarlo "TEST"

AZIONE: porre un floppy nuovo nell'unità a floppy disk

COMANDO: DISK"NØ:TEST,Ø1" < RETURN >

RISULTATO: dopo alcuni minuti, il nuovo floppy è stato formattato con il titolo "TEST" e l'indicatore luminoso dell'unità si spegne per indicare che il canale del floppy è stato chiuso.

ESEMPIO: cancellare un programma da un floppy.

AZIONE: battere il seguente breve programma e memorizzarlo sul floppy sotto il nome di "ARANCIO".

```
1Ø REM"★★★ ESEMPIO DI DISK ★★★
2Ø REM"★★★ CANCELLAZIONE DI UN PROGRAMMA ★★★
```

COMANDO: DISK"SØ:ARANCIO" < RETURN >

RISULTATO: il programma "ARANCIO" viene cancellato dal floppy e quindi il canale del floppy viene chiuso.

5.3 DIR

FORMATO: DIR "$

oppure: DIR "$:stringa di caratteri★

oppure: DIR "$:?stringa di caratteri

oppure: DIR "$:?stringa di caratteri★

SCOPO: listare tutto o in parte il file directory di un floppy disk.

DIR sostituisce il comando in BASIC standard: LOAD"$",8 e vi permette di listare completamente o in parte il file directory del floppy disk. Potete con questo comando listare ad esempio solo quei file il cui nome inizia con un carattere particolare, oppure con una stringa di caratteri particolari, introducendo nel comando il carattere o la stringa, seguiti da un asterisco. Oppure potete visualizzare solo quei file dove un determinato carattere, o una determinata stringa di caratteri si trovano in una specificata posizione, all'interno del nome del file, sostituendo i caratteri precedenti con dei punti interrogativi(?).

ESEMPIO: visualizzare il listato completo del file directory

COMANDO: DIR"$ < RETURN >

RISULTATO: il file directory del floppy disk nel dispositivo N. 8 viene visualizzato sullo schermo.

ESEMPIO: listare solo quei file dove il terzo carattere del loro nome è "S".

COMANDO: DIR"$:??S* < RETURN >

RISULTATO: lo schermo mostra l'elenco dei nomi dei file presenti nel file directory, in cui il terzo carattere è una S.

CAPITOLO 6
GRAFICA CON IL SIMONS' BASIC

6.1 INTRODUZIONE

Questo capitolo descrive il completo set di comandi per la grafica del SIMONS' BASIC. Grazie ad essi, potete tracciare punti e figure, introdurre testi e colorare delle superfici sullo schermo, usando uno qualunque dei 16 colori del COMMODORE 64, senza dover accedere alle locazioni di memoria.

Il comando COLOUR vi permette di scegliere il colore dello schermo e quello del bordo dello schermo, HIRES pone il COMMODORE 64 nel modo grafico "high-resolution" (alta risoluzione), in cui ogni punto è individuato sullo schermo pixel per pixel (minima parte indirizzabile dello schermo), MULTI pone il computer nel modo grafico "multicolour", in cui ogni punto occupa due pixel. Sia con HIRES, che con MULTI, potete scegliere il colore che desiderate utilizzare per tracciare le vostre figure. Con il comando LOW COL potete cambiare questi colori, mentre il comando HI COL vi permette di riottenere i colori originariamente scelti.

Il comando PLOT vi permette di tracciare dei punti singoli sullo schermo, mentre TEST vi mette in grado di verificare lo stato di una data locazione dello schermo, cioè se è in essa presente un punto, e, in caso positivo in quale colore esso è stato tracciato.

I comandi REC e CIRCLE vi permettono di tracciare rispettivamente dei rettangoli e delle circonferenze, mentre con ARC potete tracciare un determinato arco di circonferenza, e con ANGL il rispettivo raggio.

Con il comando PAINT potete riempire con un dato colore le figure tracciate, mentre con BLOCK si visualizzano dei blocchi sfumati di colore. DRAW e ROT vi permettono di variare le dimensioni e l'angolazione di una qualunque figura già tracciata.

Il comando CSET vi permette di scegliere tra i set di caratteri Maiuscolo/Minuscolo o Maiuscolo/Grafico del COMMODORE 64. Lo stesso comando vi mette anche in grado di richiamare e di visualizzare l'ultimo schermo grafico che era stato presente. Infine, con i comandi CHAR e TEXT potete rispettivamente inserire su uno schermo grafico singoli caratteri o stringhe di caratteri.

La prima parte di questo capitolo descrive la configurazione dello schermo e le differenze tra la grafica "high-resolution" e quella "multi-colour". Inoltre, sono elencati i 16 colori di cui è dotato il COMMODORE 64, ed è spiegato come effettuare la loro scelta. Nella seconda parte del capitolo, viene descritto il formato è l'uso di ogni comando grafico. A tale proposito, i comandi sono elencati e descritti nello stesso ordine in cui debbono essere usati durante la stesura dei programmi tipo quelli riportati nel capitolo 13 del manuale.

Si noti che, ad eccezione di COLOUR, i comandi descritti in questo capitolo possono essere usati solamente nell'ambito di un programma.

6.2 CONFIGURAZIONE DELLO SCHERMO

Durante l'uso grafico, lo schermo del COMMODORE 64 viene configurato in una griglia, dove ogni punto è individuato dalle sue coordinate x e y, proprio come avviene in un grafico. Ad esempio, il punto di coordinate 0,0 (origine degli assi) è il punto nell'angolo in alto a sinistra dello schermo. Il passo della griglia dipende dal modo grafico in cui si trova: in modo high-resolution lo schermo è costituito da una griglia di 320x200 punti, mentre, in modo multi-colour, la griglia è di 160x200 punti. Ciò significa che ogni punto dello schermo, in modo high-resolution, è costituito da 1 pixel, mentre in modo multi-colour è costituito da 2 pixel. Il pixel è la minima porzione di schermo indirizzabile singolarmente.

6.3 I COLORI DEL COMMODORE 64

Sia nel modo grafico high-resolution, che in quello multi-colour, è possibile utilizzare contemporaneamente al massimo TRE dei 16 colori disponibili, per ogni quadrato di lato uguale ad 8 pixel (8x8 pixel = 64 pixel di area).

I 16 colori disponibili sul COMMODORE 64 sono qui DI SEGUITO elencati, con accanto i valori ad essi associati:

Ø Nero	8 Arancione
1 Bianco	9 Arancione chiaro
2 Rosso	1Ø Rosa
3 Ciano	11 Ciano chiaro
4 Porpora	12 Porpora chiaro
5 Verde	13 Verde chiaro
6 Blu	14 Blu chiaro
7 Giallo	15 Giallo chiaro

Quando lavorate in modo grafico, il colore che desiderate utilizzare viene specificato per mezzo del numero ad esso associato.

6.4 PLOT TYPE

Tutti i comandi grafici del SIMONS' BASIC hanno come caratteristica comune la necessità che sia specificato il "plot-type", cioè il modo in cui deve essere tracciato ogni punto. Ecco l'elenco dei vari plot-type, sia in modo high-resolution, che nel modo multicolour.

MODO HIGH-RESOLUTION

TIPO	FUNZIONE EFFETTUATA
Ø	spegne un punto sullo schermo (OFF)
1	accende un punto sullo schermo (ON)
2	inverte un punto sullo schermo (cioè spegne il punto se è acceso e accende il punto se è spento)

MODO MULTI-COLOUR

TIPO	FUNZIONE EFFETTUATA
Ø	spegne un punto sullo schermo (OFF)
1	accende un punto sullo schermo nel colore 1 definito con il comando MULTI LOW COL
2	accende un punto sullo schermo nel colore 2 definito con il comando MULTI/LOW COL
3	accende un punto sullo schermo nel colore 3 definito con il comando MULTI/LOW COL
4	inverte ciclicamente il colore di un punto sullo schermo, cioè: un punto di colore Ø diventa di colore 3 un punto di colore 1 diventa di colore 2 un punto di colore 2 diventa di colore 1 un punto di colore 3 diventa di colore Ø

6.5 COMANDI GRAFICI

6.5.1 COLOUR

FORMATO: COLOUR sc,bo

SCOPO: scegliere i colori dello sfondo e del bordo dello schermo. Il comando COLOUR vi permette di scegliere il colore dello sfondo in modo low-resolution (bassa risoluzione) ed il colore dei bordi per i modi low- ed gigh-resolution. Per scegliere i colori, occorre specificare il numero ad essi associato (vedi paragrafo 6.3).

Si noti che il colore dello sfondo nel modo low-resolution rimane lo stesso finchè non si usa nuovamente COLOUR con un colore diverso. Il colore dello schermo nei modi grafici high-resolution e multi-colour viene scelto con il comando HIRES (vedi paragrafo seguente).

ESEMPIO: scegliere il colore ciano per lo sfondo ed il colore blu per i bordi.

COMANDO: COLOUR 3,6 < RETURN >

RISULTATO: appare uno schermo ciano circondato da bordi blu.

6.5.2 HIRES

FORMATO: HIRES pc,sb

SCOPO: inizializzare il modo grafico high-resolution e scegliere il colore del disegno ed il colore dello sfondo.

Il comando HIRES pone lo schermo nel modo grafico high-resolution, per cui ogni punto occupa un pixel. Il primo parametro, pc, rappresenta il numero del coplore che si desidera usare per il disegno (fate riferimento al paragrafo 6.3). Il secondo parametro, sb, permette di scegliere il colore dello sfondo di ogni quadrato di 8x8 pixel su cui si effettua il disegno. Si noti che occorre lasciare uno spazio bianco tra il comando HIRES ed il primo parametro.

ESEMPIO: scegliere il colore nero per disegnare su fondo bianco

INTRODURRE:
```
1Ø HIRES Ø,1
2Ø GOTO 2Ø
```

BATTERE: RUN < RETURN >

RISULTATO: viene visualizzato uno schermo completamente bianco

AZIONE: premere il tasto RUN/STOP

RISULTATO: riappare lo schermo normale

6.5.3 REC

FORMATO: REC x,y,xl,yl, plot type

SCOPO: tracciare un rettangolo

Il comando REC vi permette di disegnare sullo schermo un rettangolo. I primi due parametri (x,y) del comando vi permettono di scegliere le coordinate dell'angolo in alto a sinistra del rettangolo, rispetto all'origine dello schermo in alto a sinistra. I parametri x1 e x2 indicano rispettivamente il lato orizzontale e quello verticale del rettangolo stesso. L'ultimo parametro, il plot type, è stato descritto nel paragrafo 6.4.

ESEMPIO: tracciare un rettangolo nel modo grafico high-resolution, con un angolo nell'origine degli assi dello schermo (in alto a sinistra).

INTRODURRE:
```
1Ø HIRES Ø,1
2Ø REC Ø,Ø,4Ø,2Ø,1
3Ø GOTO 3Ø
```

BATTERE: RUN < RETURN >

RISULTATO: viene visualizzato un rettangolo nero su fondo bianco.

AZIONE: premere il tasto RUN/STOP

RISULTATO: viene nuovamente visualizzato lo schermo normale.

6.5.4 MULTI

FORMATO: HIRES pc,sb: MULTI c1,c2,c3

SCOPO: inizializzare il modo grafico multi-colour e scegliere la terna di colori che si desidera usare.

Usando il comando MULTI dopo il comando HIRES, viene attivato il modo grafico multi-colour, in cui ogni punto occupa due pixel. I tre parametri che seguono il comando MULTI permettono appunto di definire i 3 colori che si desiderano usare. Ogni colore viene definito facendo riferimento alla sua posizione nel comando MULTI, come plot type (vedi paragrafo 6.4).

ESEMPIO: inizializzare il modo grafico multi-colour specificando i colori rosso, ciano e blu, e quindi disegnare tre rettangoli, ognuno di colore diverso.

INTRODURRE:
```
10 HIRES0,1:MULTI 2,3,6
30 REC 0,0,40,20,1
40 REC 20,20,40,20,2
50 REC 40,40,40,20,3
60 GOTO 60
```

BATTERE: RUN < RETURN >

RISULTATO: vengono disegnati 3 rettangoli, uno rosso, uno ciano ed uno blu.

AZIONE: premere il tasto RUN/STOP

RISULTATO: viene nuovamente visualizzato lo schermo normale.

6.5.5 NRM

FORMATO: NRM

SCOPO: riportare il COMMODORE 64 da modo grafico a modo low-resolution.

Il comando NRM vi permette di cancellare uno schermo in modo grafico (high-resolution o multi-colour) e di ritornare in modo low-resolution.

ESEMPIO: usando il programma del paragrafo precedente, ritornare in modo normale, dopo aver visualizzato per 5 secondi lo schermo grafico.

INTRODURRE:
```
10 HIRES 0,1:MULTI 2,3,6
30 REC 0,0,40,20,1
40 REC 20,20,40,20,2
50 REC 40,40,40,20,3
60 PAUSE 5
70 NRM
```


BATTERE: RUN < RETURN >

RISULTATO: sono visualizzati per 5 secondi i tre rettangoli dell'esempio precedente. Successivamente lo schermo ritorna in modo normale, con la scritta READY ed il cursore lampeggiante.

6.5.6 LOW COL

FORMATO: LOW COL c1,c2,c3

SCOPO: cambiare i colori scelti in precedenza

Il comando LOW COL vi permette di scegliere un set di colori differenti da quelli scelti in precedenza con i comandi HIRES o MULTI.

NOTA

Poichè nel modo grafico high-resolution si possono usare solo due colori (fate riferimento al paragrafo 6.5.2), il terzo colore scelto con il comando LOW COL non ha effetto nel modo high-resolution. Tuttavia, il formato del comando LOW COL richiede lo stesso tre parametri, indipendentemente dal modo in cui è stato inizializzato il modo grafico.

ESEMPIO: disegnare un rettangolo nero nel modo grafico high-resolution, e quindi cambiare in giallo il colore dei lati.

INTRODURRE:
```
10 HIRES 0,1
20 LOW COL 0,7,0
30 REC 20,20,60,60,1
40 PAUSE 5
50 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: appare un rettangolo tracciato in nero sullo schermo bianco; successivamente ogni rettangolo di 8x8 pixel sul quale è stato disegnato il rettangolo diventa giallo. Dopo 5 secondi, appare nuovamente lo schermo in modo normale.

ESEMPIO: tracciare tre rettangoli, in modo grafico multi-colour, ognuno di colore diverso e quindi, cambiare i colori originali e disegnare altri tre rettangoli, ognuno in uno dei nuovi colori.

INTRODURRE:
```
10 HIRES 0,1: MULTI 2,3,6: Z= 10
20 FOR X = 1 TO 3
30 REC 10,Z,30,30,X
40 Z = Z + 40: NEXT
50 LOW COL 4,5,7 : Z = 10
60 FOR X = 1 TO 3
70 REC 50,Z,30,30,X
80 Z = Z + 40: NEXT
90 PAUSE 5
100 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: sono stati ottenuti sei rettangoli, ognuno di un colore differente. Dopo 5 secondi appare nuovamente lo schermo in modo normale.

6.5.7 HI COL

FORMATO: HI COL

SCOPO: riottenere i colori originariamente scelti.

Il comando HI COL vi permette di ripristinare i colori originariamente scelti, cioè quelli scelti con i comandi HIRES e MULTI, dopo aver usato il comando LOW COL (fate riferimento al capitolo precedente).

ESEMPIO: disegnare nove rettangoli i colori differenti.

INTRODURRE:
```
10 HIRES 0,1: MULTI 2,3,6: Z = 10
20 FOR Y = 10 TO 50 STEP 40
30 FOR X = 1 TO 3
40 REC Y,Z,30,30,X
50 Z = Z + 40:NEXT X: Z = 10:LOW COL 4,5,7:NEXT Y
60 HI COL
70 FOR X = 1 TO 3
80 REC Y,Z,30,30,X
90 Z = Z + 40:NEXT
100 PAUSE 5
110 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: tre rettangoli sono visualizzati nei colori scelti originariamente, tre nei colori assegnati con il comando LOW COL ed altri tre usando ancora i colori scelti originariamente. Dopo 5 secondi appare nuovamente lo schermo in modo normale.

6.5.8 PLOT

FORMATO: PLOT x,y,plot type

SCOPO: per disegnare un punto.

PLOT permette di disegnare un singolo punto sullo schermo in modo grafico. I parametri x ed y specificano rispettivamente le coordinate orizzontale e verticale del punto che deve essere tracciato. Il plot type è stato utilizzato descritto nel paragrafo 6.4.

ESEMPIO: disegnare un punto nero in modo grafico multi-colour.

INTRODURRE:
```
10 HIRES 0,1:MULTI 0,1,2
20 PLOT 160,100,1
30 PAUSE 5
40 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: un punto nero viene visualizzato nel centro dello schermo. Dopo 5 secondi, appare nuovamente lo schermo normale.

ESEMPIO: disegnare una curva punteggiata.

INTRODURRE:
```
10 HIRES 0,1
20 FOR X = 0 TO 320 STEP .5
30 Y = 100 + SIN(X/30)*90
40 PLOT X,Y,1
50 NEXT
1000 GOTO 1000
```

BATTERE: RUN < RETURN >

RISULTATO: viene disegnata un'onda sinusoidale punteggiata in nero.

6.5.9 TEST

FORMATO: variabile = TEST (x,y)

SCOPO: determinare se in una determinata posizione dello schermo è rappresentato qualche cosa.

TEST vi permette di esaminare lo stato di una qualunque locazione sullo schermo grafico. I parametri x ed y costituiscono le coordinate sullo schermo del punto in cui si desidera effettuare la verifica. Se un punto è stato disegnato in quella posizione, viene visualizzato il plot type del punto stesso (vedi paragrafo 6.4). Se non è rappresentato alcun punto, viene visualizzato un valore pari a zero. Il punto può costituire un punto qualunque di un disegno.

ESEMPIO: tracciare una linea che termina quando ne tocca un'altra.

INTRODURRE:
```
10 REM"**** ESEMPIO DI TEST ***
20 HIRES 0,1
25 FOR X = 0 TO 200
30 PLOT 200,X,1:NEXT
40 FOR I = 1 TO 320
50 IF TEST(I,100) = 1 THEN 70
60 PLOT I,100,1:NEXT
70 PAUSE 5
80 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciata una linea orizzontale che termina quando incontra la linea verticale. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.5.10 LINE

FORMATO: LINE x,y,x1,y1,plot type

SCOPO: tracciare una linea tra due punti dati.

LINE traccia una linea da un punto dello schermo ad un altro. I parametri x ed y sono le coordinate del punto di partenza della linea, mentre x1 ed y1 costituiscono le coordinate del punto terminale della linea stessa. Il plot type è descritto al paragrafo 6.4.

ESEMPIO: tracciare una linea diagonale attraverso lo schermo.

INTRODURRE:
```
10 HIRES 0,1
20 LINE 0,0,320,200,1
30 PAUSE 5
40 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciata una linea nera che parte dall'angolo in alto a sinistra e termina nell'angolo in basso a destra dello schermo. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.5.11 CIRCLE

FORMATO: CIRCLE x,y,xr,plot type.

SCOPO: tracciare una linea circolare.

CIRCLE vi permette di tracciare una linea circolare sullo schermo grafico. I parametri x ed y specificano le coordinate del centro sullo schermo, mentre i parametri xr ed yr indicano rispettivamente il raggio

orizzontale e verticale. Variando questi ultimi, è quindi possibile tracciare circonferenze od ellissi di varie dimensioni. Il plot type è stato descritto al paragrafo 6.4.

NOTA

Poichè lo schermo è rettangolare e non quadrato, per tracciare una circonferenza, i raggi x ed y non sono uguali. Nel modo high-resolution, il raggio x è pari ad 1,4 volte il raggio y, mentre, nel modo multi-colour, x è pari ad 1,6 volte il raggio y. Tuttavia, volendo ottenere una copia stampata di una circonferenza da uno schermo high-resolution o multi-colour, occorre porre i due raggi uguali. Per maggiori dettagli, vedi il paragrafo 7.13.2.

ESEMPIO: tracciare una circonferenza in modo high-resolution.

INTRODURRE:
```
10 HIRES 0,1
20 CIRCLE 160,100,56,40,1
30 PAUSE 5
40 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciata una circonferenza nera nel centro dello schermo. Dopo 5 secondi, appare nuovamente lo schermo normale.

ESEMPIO: tracciare un'ellisse in modo multi-colour.

INTRODURRE:
```
10 HIRES 0,1:MULTI 2,3,4
20 CIRCLE 80,100,60,30,1
30 PAUSE 5
40 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciata una ellisse rossa. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.5.12 ARC

FORMATO: ARC,x,y,sa,ea,i,sr,yr,plot type

SCOPO: tracciare un arco di forma circolare.

Il comando ARC vi permette di tracciare un arco. I parametri x ed y costituiscono le coordinate del centro della curva a cui appartiene l'arco; sa ed se indicano gli angoli di inizio e fine dell'arco. Il parametro i specifica l'incremento del tracciato cioè l'intervallo in gradi tra ogni

punto tracciato dell'arco. Per ottenere un arco continuo, tale valore è 1. Con un valore maggiore, i punti che compongono l'arco risultano distanziati. I parametri xr ed yr indicano rispettivamente i raggi orizzontale e verticale della curva a cui appartiene l'arco. Il plot type è stato descritto al paragrafo 6.4.

ESEMPIO: tracciare due archi appartenenti alla stessa circonferenza.

INTRODURRE:
```
10 HIRES 0,1
20 ARC 160,100,30,150,1,40,40,1
30 ARC 160,100,210,330,1,40,40,1
40 PAUSE 5
50 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: è stata tracciata una coppia di parentesi nere. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.5.13 ANGL

FORMATO: ANGL x,y,angolo, xr,yr,plot type

SCOPO: tracciare il raggio di una circonferenza.

Il comando ANGL vi mette in grado di tracciare il raggio di un cerchio senza dover tracciare la circonferenza stessa. I parametri x ed y sono le coordinate del centro, "angle" è l'angolo, espresso in gradi in senso orario rispetto alla verticale, cioè, un raggio a 90° è nella stessa posizione delle ore 3 sull'orologio. I parametri xr ed yr sono rispettivamente i raggi orizzontale e verticale della curva a cui appartiene il raggio. Il plot type è stato descritto al paragrafo 6.4.

ESEMPIO: tracciare una ruota.

INTRODURRE:
```
10 HIRES 0,1
20 CIRCLE 160,100,40 * 1.4,40,1
30 CIRCLE 160,100,45 * 1.4,45,1
40 FOR X = 0 TO 360 STEP 22.5
50 ANGL 160,100,X,40 * 1.4,40,1
60 NEXT
70 PAUSE 10
80 NRM
```


BATTERE: RUN < RETURN >

RISULTATO: è stata tracciata una ruota a raggi nera. Dopo 10 secondi, appare nuovamente lo schermo normale.

ESEMPIO: tracciare un ventaglio.

INTRODURRE:
```
10 HIRES 0,1
20 FOR X = 0TO 170 STEP 5
30 ANGL 160,100,X,40,40,1
40 NEXT
50 FOR X = 170 TO 0 STEP-5
60 ANGL 160,100,X,40,40,0
70 NEXT
80 PAUSE 10
90 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: è stato tracciato un ventaglio che viene aperto e quindi chiuso. Dopo 10 secondi, appare nuovamente lo schermo normale.

6.5.14 PAINT

FORMATO: PAINT X, Y, PLOT type (0,1,2,3 solo)

SCOPO: riempire di colore una superficie.

PAINT riempie una superficie tracciata, con il colore definito dal plot type (vedi paragrafo 6.4). La superficie deve essere chiusa, altrimenti viene colorato tutto lo schermo. La superficie viene identificata per mezzo delle coordinate di un punto qualunque compreso in essa. Nel modo high-resolution, la stessa superficie può essere colorata una sola volta, ma è possibile superare questa limitazione cancellando lo schermo, cambiando i colori con LOW COL (paragrafo 6.5.6), ritracciando la figura e quindi colorandola nuovamente. Nel modo multicolour la stessa superficie può essere riempita con colori diversi il numero di volte desiderato.

ESEMPIO: tracciare un rettangolo nero e quindi riempirlo di giallo.

INTRODURRE:
```
10 HIRES 0,1
20 REC 120,60,40,1
30 LOW COL 7,1,0
40 PAINT 130,70,1
50 PAUSE 5
60 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciato un quadrato nero nel centro dello schermo e quindi viene riempito di giallo. Dopo 5 secondi, appare nuovamente lo schermo normale.

ESEMPIO: tracciare un diagramma a torta colorato.

INTRODURRE:
```
10 HIRES 0,1:MULTI 5,4,6
20 CIRCLE 80,100,48,78,1
30 ANGL 80,100,120,48,78,1
40 ANGL 80,100,160,48,78,1
50 ANGL 80,100,220,48,78,1
60 ANGL 80,100,330,48,78,1
70 PAINT 90,35,1
80 PAINT 60,60,3
90 PAINT 90,120,2
105 LOW COL 7,4,6
110 PAINT 80,110,1
120 PAUSE 5
130 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene tracciato nel centro dello schermo un diagramma a torta diviso in quattro settori, ognuno colorato in modo differente. Dopo 5 secondi appare lo schermo normale.

6.5.15 BLOCK

FORMATO: BLOCK x,y,x1,y1,plot type

SCOPO: tracciare un rettangolo completamente colorato.

Il comando BLOCK traccia un rettangolo e contemporaneamente lo riempie di colore. Pertanto effettua la stessa funzione dei due comandi REC e PAINT: BLock effettua le due operazioni contemporaneamente. Occorre però notare che in questo caso il colore dei lati del rettangolo non può essere diverso da quello con cui viene riempito.

Il comando BLOCK è utile desiderando creare numerosi blocchi adiacenti di differenti colori senza separarli con linee. I parametri x ed y danno le coordinate dell'angolo in alto a sinistra, mentre x1 ed y1 sono le coordinate dell'angolo in alto a sinistra, mentre x1 ed y1 sono le coordinate dell'angolo in basso a destra. Il plot type è stato descritto al paragrafo 6.4.

ESEMPIO: tracciare due blocchi di colori diversi

INTRODURRE:
```
10 HIRES 0,1:MULTI 2,6,1
20 BLOCK 10,50,50,90,1
30 BLOCK 51,50,90,90,2
40 PAUSE 5
50 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: viene disegnato un blocco rosso adiacente ad uno blu. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.5.16 DRAW

FORMATO: DRAW "nnnnnnn....9" ,x,y,plot type

SCOPO: disegnare una figura.

Il comando DRAW vi permette di preparare una figura e quindi di visualizzarla sullo schermo. La figura viene composta come quando si disegna su di un foglio di carta, senza staccare la matita da esso. Esiste tuttavia una notevole differenza: è infatti possibile comandare al COMMODORE 64 di spostare la matita, senza tuttavia tracciare il segno corrispondente (vedere l'esempio al paragrafo 6.5.17).

I parametri x ed y del comando DRAW sono le coordinate del punto sullo schermo dove inizia la figura. Ogni valore n fra virgolette rappresenta un'istruzione per il COMMODORE 64 su come muovere la matita per disegnare la figura. Su ogni linea di programma è possibile porre al massimo 74 istruzioni di questo tipo tra virgolette. È tuttavia possibile aggiungere delle stringhe di istruzioni, fino ad un massimo di 255. Per continuare la figura oltre questo limite, occorre definire una nuova origine dove è terminata la stringa precedente. Sono elencate qui sotto le istruzioni ed il valore associato.

NUMERO	ISTRUZIONE
0	muovere un pixel a destra
1	muovere un pixel in alto
2	muovere in pixel in basso
3	muovere un pixel a sinistra
5	muovere un pixel a destra e tracciare un punto
6	muovere un pixel in alto
7	muovere un pixel in basso
8	muovere un pixel a sinistra
9	figura terminata

Notare che facendo uso solamente delle istruzioni elencate, è possibile costruire qualunque figura. La figura però, non viene tracciata finchè i comandi DRAW e ROT non sono eseguiti (paragrafo precedente). Il comando DRAW definisce la forma mentre ROT la genera. Il plot type è stato descritto al paragrafo 2.4.

ESEMPIO: memorizzare il disegno di una campana.

INTRODURRE:
```
10 A$ = "5757575787878757575757575
   777777777777575757575757578888888888888"
20 A$ = A$ + "88888888888888656565656565
   6666666666666"
30 A$ = A$ + "56565656565686868656565656"
```

RISULTATO: quando sono eseguite queste istruzioni, le istruzioni per disegnare una campana sono memorizzate nella variabile A$.

6.5.17 ROT

FORMATO: ROT r,s

SCOPO: visualizzare una figura con determinate dimensioni ed angolazione.

Il comando ROT vi permette di visualizzare una figura creata con DRAW (vedi paragrafo precedente) specificando dimensioni ed angolo di rotazione. Il parametro r indica di quanto la figura deve essere ruotata rispetto alla perpendicolare passante per la sua origine, cioè quel punto dello schermo dove è iniziata la figura. Questo valore di r, è compreso tra Ø e 7 e definisce l'angolo di rotazione, come segue:

NUMERO	GRADI
Ø	Ø
1	45
2	9Ø
3	135
4	18Ø
5	225
6	27Ø
7	315

Il secondo parametro di ROT definisce le dimensioni della figura. Ponendo s pari ad 1, la figura viene tracciata come è stata memorizzata, cioè ogni parametro nel comando DRAW rappresenta 1 pixel. Aumentando il valore di s si ottiene un aumento delle dimensioni della figura.

NOTA

Se impostate delle dimensioni troppo grandi della figura che avete memorizzato, la stessa sparisce dallo schermo perchè non può essere in esso contenuta. Assicuratevi quindi di mantenere le dimensioni della figura a valori rappresentabili sullo schermo.

ESEMPIO: visualizzare a dimensioni normali e quindi maggiorate la figura dell'esempio precedente.

INTRODURRE:

```
 1Ø A$ = "5757575787878757575757575
    777777777777757575757575757888888888888"
2Ø A$ = A$ + "88888888888888656565656565
    6666666666666
 3Ø A$ = A$ + "56565656565686868656565656"
 4Ø HIRES Ø,1
 45 FOR Y = Ø TO 7
 5Ø FOR X = 1 TO 3
 6Ø ROT Y,X
 7Ø DRAW A$,16Ø,8Ø,1
 8Ø PAUSE 1
 9Ø DRAW A$, 16Ø,8Ø,Ø
1ØØ NEXT:NEXT
11Ø FOR X = 3 TO 1 STEP -1
115 FOR Y = 7 TO Ø STEP -1
12Ø ROT Y,X
13Ø DRAW A$,16Ø,8Ø,1
14Ø PAUSE 1:DRAW A$,16Ø,8Ø,Ø:NEXT:NEXT
15Ø GOTO 45
```

BATTERE: RUN < RETURN >

RISULTATO: viene visualizzata una campana con sette diversi angoli di rotazione ed in tre dimensioni differenti.

6.5.18 CSET

FORMATO: CSET n

SCOPO: scegliere uno dei set di caratteri del COMMODORE 64 o richiamare e visualizzare l'ultimo schermo grafico.

Il comando CSET può effettuare tre funzioni, a seconda del parametro n. CSET Ø vi permette di scegliere il set di caratteri Maiuscolo/Grafico. CSET 1 imposta il set Maiuscolo/Minuscolo e CSET 2 rivisualizza l'ultimo schermo grafico che era apparso.

NOTA

Quando richiamate uno schermo grafico multicolour, dovete sempre far seguire CSET 2 con il comando MULTI (paragrafo 6.5.4) con gli stessi parametri originariamente usati con questo comando.

ESEMPIO: visualizzare una stringa di caratteri usando set alternati di caratteri.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 PRINT AT (12,14)"SIMONS' BASIC"
30 CSET 1:PAUSE 1:CSET 0:PAUSE 1
40 GOTO 10
```

BATTERE: RUN < RETURN >

RISULTATO: la stringa di caratteri "SIMONS' BASIC" viene visualizzata nel centro dello schermo, prima a caratteri maiuscoli e quindi a caratteri minuscoli.

ESEMPIO: rivisualizzare uno schermo grafico high-resolution creato in precedenza e quindi cancellato.

INTRODURRE:
```
10 HIRES 0,1:MULTI 0,4,6
15 FOR I = 1 TO 20
20 A = INT(90 * RND(1)) + 2: B = INT(90 * RND(1)) + 2
25 C = INT(90 * RND(1)) + : D = INT(60 * RND(1)) + 2
27 P = INT(3 * RND(1)) + 1
30 REC A,B,C,D,1
35 PAINT A + 1,B + 1,P
40 PAUSE 2:CSET 0
50 PRINT"< SHIFT CLR/HOME > PREMERE UN TASTO QUALUNQUE
   PER RIVISUALIZZARE"
60 PRINT"(< CURSOR DOWN >) L'ULTIMO SCHERMO
70 GET A$: IF A$ = "" THEN 70
80 CSET 2: MULTI 0,4,6: PAUSE 2:CSET 0
90 GOTO 50
```

BATTERE: RUN < RETURN >

RISULTATO: vengono tracciati 20 quadrati colorati su schermo grafico multi-colour. Dopo alcuni secondi, appare lo schermo normale ed il messaggio: PREMERE UN TASTO QUALUNQUE PER RIVISUALIZZARE L'ULTIMO SCHERMO.

AZIONE: premere un tasto qualunque.

RISULTATO: viene nuovamente visualizzato lo schermo grafico con i quadrati.

6.6 STAMPA DI UN TESTO SU SCHERMO GRAFICO

FORMATO: CHAR x,y, codice poke, plot type, dimensione.

SCOPO: visualizzare un carattere su schermo grafico.

Il comando CHAR vi permette di visualizzare un testo carattere per carattere nei modi high-resolution e multi-colour. I parametri x ed y specificano la posizione del carattere sullo schermo. Il successivo parametro rappresenta il codice poke di quel carattere (vedere l'elenco dei codici poke nella Guida per l'utente del vostro COMMODORE 64). Il plot type è stato descritto al paragrafo 6.4. L'ultimo parametro di questo comando specifica l'altezza del carattere, nella gamma da 1 ad 8. Il valore 1 indica un carattere di dimensioni normali, cioè alto 8 pixel. Aumentando il parametro, aumenta in proporzione l'altezza del carattere. Ad esempio, per un valore pari a 3, il carattere è alto 24 pixel. La larghezza del carattere NON PUÒ essere variata.

NOTA
I simboli grafici definiti dall'utente non possono essere usati in modo grafico high-resolution e multi-colour.

ESEMPIO: visualizzare i caratteri con dimensioni doppie di quelle normali.

INTRODURRE:
```
10 REM"*** ESEMPIO DI < CHAR > ***
20 HIRES 0,1
30 FOR J = 1 TO 12
40 FOR I = 0 TO 40
50 CHAR I * 8, 15,I + J * 40,1,2
60 NEXT:NEXT
70 PAUSE 5
80 NRM
```

BATTERE: RUN < RETURN >

RISULTATO: l'intero set di caratteri del COMMODORE 64 viene visualizzato con altezza doppia del normale. Dopo 5 secondi, appare nuovamente lo schermo normale.

6.6.2 TEXT

FORMATO: TEXT x,y, "(CTRL a) stringa di caratteri",plot type,s,i

oppure: TEXT X,Y, "(CTRL b) stringa di caratteri", plot type,s,i

SCOPO: visualizzare una stringa di caratteri su uno schermo grafico.

TEXT vi permette di visualizzare stringhe di caratteri in modo grafico. I parametri x ed y indicano le coordinate sullo schermo del primo carattere della stringa. Il parametro successivo è la stringa stessa. Il carattere di controllo che precede indica se il testo deve essere visualizzato a caratteri maiuscoli o minuscoli. Per ottenere il testo a caratteri maiuscoli:

1. battere il primo gruppo di virgolette

2. tenere premuto il tasto CTRL e premere il tasto "a" (viene visualizzata una A in campo inverso)

3. introdurre la stringa di caratteri

4. introdurre il secondo gruppo di virgolette.

Per visualizzare il testo a caratteri minuscoli:

1. battere il primo gruppo di virgolette

2. tenere premuto il tasto CTRL e premere il tasto "b" (viene visualizzata una B in campo inverso)

3. introdurre la stringa di caratteri

4. introdurre il secondo gruppo di virgolette.

Potete anche utilizzare nella stessa stringa caratteri maiuscoli e minuscoli. Per fare ciò, tenete premuto CTRL e premete "a" prima dei caratteri maiuscoli e tenete premuto CTRL e premete "b" prima dei caratteri minuscoli.

Il plot type è stato descritto al paragrafo 6.4. Il parametro s specifica l'altezza dei caratteri della stringa nella gamma da 1 a 8. Il valore 1 indica dimensioni normali; aumentando il parametro, aumenta in proporzione l'altezza del carattere, fino ad 8 che è il valore massimo. La larghezza dei caratteri non può essere cambiata. Il parametro i definisce invece il numero di pixel tra ogni carattere della stringa che vale, normalmente 8. Aumentando questo valore, aumenta di conseguenza la spaziatura.

ESEMPIO: visualizzare due stringhe di caratteri sullo schermo high-resolution.

INTRODURRE:
```
10 REM"*** ESEMPIO DI TEXT ***
20 HIRES 0,1
30 FOR I = 1 TO 30
40 X = INT(320 * RND(1)):Y = INT(200 * RND(1))
50 LINE 160,100,X,Y,1:NEXT
60 TEXT 60,20,"< CTRL B > TEXT SU UNO SCHERMO HIRES", 1,2,8
70 TEXT 20,180, "< CTRL A > DOVUNQUE < CTRL B >
   TU LO VOGLIA!",1,1,16
80 PAUSE 5
90 NRM
```

RISULTATO: vengono tracciate delle linee a caso.
Sono visualizzati due messaggi, il primo a lettere minuscole 8 volte più grandi di quelle normali, il secondo a lettere maiuscole e minuscole con doppia spaziatura tra i caratteri.

CAPITOLO 7
GESTIONE DELLO SCHERMO

7.1 INTRODUZIONE

Tra le potenti caratteristiche del SIMONS' BASIC sono anche compresi numerosi comandi che assicurano la gestione completa di grafici a bassa risoluzione e del trattamento dei dati sullo schermo.

Il comando BCKGNDS vi permette di definire il colore dello sfondo di un carattere. Il comando FLASH commuta caratteri in un determinato colore in campo inverso, e quindi li riporta nuovamente nel modo normale, dopo un intervallo di tempo determinato. Il comando BFLASH provoca il lampeggio dei bordi che circondano lo schermo, mentre il comando OFF disattiva il comando FLASH.

Il comando FCHR permette di riempire totalmente una specificata area dello schermo con un unico carattere; il comando FCOL vi permette di riempire una determinata superficie dello schermo con un colore prescelto ed il comando FILL combina questi due ultimi, permettendovi di riempire una superficie definita dello schermo con un determinato carattere, nel colore desiderato.

Il comando MOVE duplica una specifica porzione dello schermo in un'altra posizione dello stesso, mentre il comando INV vi permette di invertire il modo di rappresentazione dei caratteri in una data sezione dello schermo, poichè i caratteri normali vengono rappresentati in video inverso e viceversa.

Con il comando COPY potete stampare il contenuto dello schermo, grafico per mezzo della stampante seriale COMMODORE, mentre HRDCOPY ha la stessa funzione nel caso di un normale schermo di dati.

Il comando SCRSV vi permette di memorizzare uno schermo a bassa risoluzione, mentre con SCRLD potete richiamare e visualizzare lo schermo precedentemente memorizzato.

Il capitolo 7 contiene inoltre i comandi per effettuare lo scroll di una determinata superficie dello schermo nella direzione desiderata. Ciò può anche essere effettuato con il "wrap round", cioè le lettere che scompaiono da un lato dello schermo ricompaiono dall'altro lato dello stesso, oppure senza, cioè le lettere che escono fuori dall'area di scroll non ricompaiono.

È bene notare che i comandi presentati e discussi in questo capitolo possono essere utilizzati sia direttamente da tastiera, che da programma.

7.2 BCKGNDS

FORMATO: BCKGNDS sc,b1,b2,b3

SCOPO: cambiare il colore dello sfondo di un carattere.

Quando un carattere è visualizzato sullo schermo, occupa una superficie quadrata di lato 8x8 pixel. Il colore del quadrato è normalmente quello del resto dello schermo (tranne nel caso in cui il carattere sia visualizzato in video inverso). Con il comando BCKGNDS potete cambiare il colore di questo quadrato, sia in video normale, che in video inverso. Si noti che solamente i caratteri indicati sulla superficie superiore di ogni tasto possono essere usati con il comando BCKGNDS. I simboli grafici NON possono essere utilizzati.

ESEMPIO: visualizzare un messaggio usando tre differenti colori dello sfondo dei caratteri (notare che nel seguente programma i caratteri in corsivo debbono essere battuti con il tasto SHIFT premuto).

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 BCKGNDS 1,3,5,6
30 PRINT"                                        ":PRINT
40 PRINT"< CTRL RVS ON > DEL BASIC SIMONS":PRINT
50 PRINT"< CTRL RVS ON >                         ":PRINT
```

BATTERE: RUN < RETURN >

RISULTATO: vengono visualizzate tre linee di testo, la prima con sfondo ciano, la seconda con sfondo verde e la terza con sfondo blu.

7.3 FLASH

FORMATO: FLASH colore, velocità

oppure: FLASH colore

SCOPO: far lampeggiare lo schermo di un determinato colore.

Il comando FLASH vi permette di far lampeggiare lo schermo di un dato colore tra la rappresentazione normale e quella in video inverso, una volta ogni quattro secondi, oppure ad una velocità a vostra scelta. La gamma di velocità varia tra 1 e 255. Ogni unità è circa equivalente ad 1/60 di secondo, ed una volta attivato questo comando, il lampeggio continua fino a quando non si usa il comando OFF (paragrafo seguente). Si noti che il comando FLASH non può essere utilizzato nei modi grafici high-resolution e multi-colour.

ESEMPIO: far lampeggiare ad una determinata velocità le zone dello schermo colorate in rosso.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 PRINT (AT(12,10)"PERCHÈ, < CTRL/3 > CIAO < CTRL/1 > LÀ"
30 FLASH 2,10
1000 GOTO 1000
```

BATTERE: RUN < RETURN >

RISULTATO: la parola "CIAO" lampeggia di continuo, alla frequenza di circa 1/6 di secondo.

ESEMPIO: far lampeggiare le zone dello schermo colorate in nero una volta ogni quattro secondi.

AZIONE: introdurre il programma dell'esempio precedente, e quindi effettuare il listing sullo schermo.

COMANDO: FLASH 0 < RETURN >

RISULTATO: il listing del programma lampeggia ogni quattro secondi.

7.4 OFF

FORMATO: OFF

SCOPO: disattivare il comando FLASH.

OFF disattiva il comando FLASH. Si noti che il colore in cui rimangono i caratteri dipende dal momento in cui viene eseguito il comando OFF.

ESEMPIO: disattivare il comando FLASH nel seguente programma:

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 PRINT AT(12,10)"PERCHÈ < CTRL/3 > CIAO < CTRL/1 > LÀ"
30 FLASH 2,10
40 PAUSE 10
50 OFF
60 PRINT AT(12,10)"PERCHÈ < CTRL/3 > CIAO < CTRL/1 > LÀ"
```

BATTERE: RUN < RETURN >

RISULTATO: la parola "CIAO" lampeggia per soli 10 secondi.

7.5 BFLASH

FORMATO: BFLASH velocità,c1,c2

oppure: BFLASH 0

SCOPO: far lampeggiare o disattivare il lampeggio del bordo dello schermo.

BFLASH vi permette di far lampeggiare i bordi dello schermo del vostro COMMODORE 64. Il primo parametro nel comando determina la velocità nella gamma da 1 a 255. Ogni unità equivale a circa 1/60 di secondo. I parametri c1 e c2 sono i numeri dei colori in cui si desidera che lampeggino i bordi. BFLASH 0 disattiva il lampeggio. Si noti che il colore in cui rimarranno i bordi a lampeggio disattivato dipende dal momento in cui viene eseguito il comando.

ESEMPIO: far lampeggiare i bordi in rosso e blu.

COMANDO: BFLASH 25,2,6 < RETURN >

RISULTATO: i bordi dello schermo lampeggiano continuamente, dapprima in rosso e quindi in blu, cambiando colore con una frequenza di circa 1/3 di secondo.

ESEMPIO: attivare il lampeggio dei bordi e quindi disattivare lo stesso.

INTRODURRE:
```
10 BFLASH 25,2,6
20 PAUSE 5
30 BFLASH 0
1000 GOTO 1000
```

BATTERE: RUN < RETURN >

RISULTATO: i bordi dello schermo lampeggiano in rosso e blu per soli 5 secondi.

7.6 FCHR

FORMATO: FCHR r,c,w,d,codice

SCOPO: riempire una superficie dello schermo con un carattere.

Il comando FCHR vi mette in grado di riempire un'area definita dello schermo con un carattere scelto da voi. I parametri r e c costituiscono le coordinate del punto di partenza del riempimento. Le righe sono numerate da 0 a 24 e le colonne da 0 a 39. I parametri w e d definiscono rispettivamente la larghezza e la profondità dell'area di schermo. La larghezza è misurata in caratteri mentre la profondità, in colonne. L'ultimo parametro del comando è il codice poke del carattere che desiderate visualizzare (un elenco completo dei codici poke si trova nella "Guida per l'utente" del vostro COMMODORE 64).

ESEMPIO: visualizzare un blocco di 'A'.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 FCHR 0,0,10,10,1
30 GOTO 20
```

BATTERE: RUN < RETURN >

RISULTATO: un blocco di 10x10 caratteri, nell'angolo in alto a sinistra dello schermo, viene riempito con delle 'A'.

7.7 FCOL

FORMATO: FCOL r,c,w,l,colore

SCOPO: cambiare il colore di un carattere.

Il comando FCOL cambia il colore di tutti i caratteri compresi in una determinata porzione dello schermo, in un altro colore specificato. Come nel caso del comando FCHR (paragrafo precedente), i primi quattro parametri del comando definiscono la porzione dello schermo che interessa, mentre il numero che segue indica il nuovo colore che si desidera utilizzare per ogni carattere che appare in quella porzione (l'elenco dei numeri che identificano i colori è riportato nel paragrafo 6.3).

ESEMPIO: cambiare il colore dei caratteri da nero in rosso.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 FCHR 12,15,10,10,1
30 FCOL 12,15,5,5,2
```

BATTERE: RUN < RETURN >

RISULTATO: viene visualizzato un blocco di 100 'A', delle quali, un quarto sono colorate in rosso e le rimanenti in nero.

7.8 FILL

FORMATO: FILL r,c,w,l,codice,colore

SCOPO: riempire una porzione definita dello schermo con un determinato carattere, in uno specificato colore.

FILL vi mette in grado di riempire una definita area dello schermo con caratteri di tipo e colore specificato. Come nel caso del comando FCHR (vedi paragrafo 7.6), i primi quattro parametri del comando definiscono la porzione dello schermo che interessa, il numero che segue è il codice poke del carattere che si desidera usare (un elenco completo dei codici poke si trova nella "Guida per l'utente" del vostro COMMODORE 64) ed infine, l'ultimo parametro rappresenta il numero relativo al nuovo colore (l'elenco dei numeri che identificano i colori riportato nel paragrafo 6.3).

ESEMPIO: visualizzare dei blocchi di caratteri di differenti colori.

INTRODURRE:
```
10 REM"*** ESEMPIO DI FILL ***
20 PRINT"< SHIFT CLR/HOME>"
30 X = INT(40 * RND(1)):Y = INT(25 * RND(1))
40 X1 = INT(20 * RND(1)) + 1:Y1 = INT(15 * RND(1)) + 1
50 IF X + X1 > 40 OR Y + 1 > 25 THEN 30
60 FILL Y,X,X1,Y1,160,INT(16 * RND(1)):GOTO 30
```

BATTERE: RUN < RETURN >

RISULTATO: sono visualizzati dei blocchi di caratteri di differenti colori in posizione casuale sullo schermo.

AZIONE: tenendo premuto il tasto RUN/STOP, premere RESTORE.

RISULTATO: appare nuovamente lo schermo normale.

7.9 MOVE

FORMATO: MOVE r,c,w,l,dr,dc

SCOPO: duplicare una porzione dello schermo.

MOVE vi permette di rivisualizzare una determinata porzione di schermo in qualunque altra parte dello stesso. I primi quattro parametri del comando definiscono la porzione che interessa (vedi paragrafo 7.6), mentre gli ultimi due parametri specificano le coordinate di riga e colonna dell'angolo in alto a sinistra della porzione di schermo dove volete portare l'informazione.

NOTA

L'altezza della porzione che si desidera duplicare, aggiunta al numero di righe della porzione in cui si vuole riprodurre l'informazione deve essere al massimo uguale a 25 righe, cioè l'altezza massima di tutto lo schermo. Nello stesso modo, la larghezza della zona da duplicare, addizionata a quella della porzione in cui si desidera effettuare la riproduzione, deve essere inferiore a 40 colonne, cioè alla larghezza massima dello schermo. Se eccedete da questi valori, appare il messaggio "BAD MODE" e dovrete in tal caso introdurre nuovamente il messaggio.

ESEMPIO: duplicare un blocco di testo.

INTRODURRE:
```
10 REM"*** ESEMPIO DI MOVE ***
20 PRINT"< SHIFT CLR/HOME > < CTRL/1 > PREMERE LA BARRA
   SPAZIATRICE"
30 PRINT"< CTRL/2 > PER SPOSTARE QUESTA AREA"
40 PRINT"< CTRL/3 > IN UN'ALTRA PARTE
50 PRINT"< CTRL/4 > DELLO SCHERMO."
60 GET A$:IF A$ <> "" THEN 60
70 MOVE 0,0,17,5,15,20
80 GOTO 80
```

BATTERE: RUN < RETURN >

AZIONE: premere la barra spaziatrice

RISULTATO: il messaggio nell'angolo in alto a sinistra dello schermo viene duplicato nell'angolo in basso a destra dello stesso.

7.10 INV

FORMATO: INV r,c,w,l

SCOPO: invertire la presentazione di una specificata porzione dello schermo.

Il comando INV vi permette di visualizzare in video inverso tutti i caratteri normali compresi in una porzione dello schermo precedentemente definita. Ovviamente, i caratteri già rappresentati in video inverso, verranno, dopo questo comando, presentati in video normale (si noti che il comando INV Pone ad 1 od a 0 il bit 7 di ogni carattere).

ESEMPIO: invertire la rappresentazione di uno specificato blocco di testo.

INTRODURRE:
```
10 REM"*** ESEMPIO DI INV ***
20 PRINT"< SHIFT CLR/HOME > PREMERE LA BARRA
   SPAZIATRICE"
30 PRINT"PER INVERTIRE LA RAPPRESENTAZIONE DI QUESTA"
40 PRINT"AREA DELLO SCHERMO!"
50 GET A$:IF A$ <> "" THEN 50
60 INV 0,0,16,4
70 GOTO 50
```

BATTERE: RUN < RETURN >

RISULTATO: il messaggio "PREMERE LA BARRA SPAZIATRICE PER INVERTIRE LA RAPPRESENTAZIONE DI QUESTA AREA DELLO SCHERMO" viene visualizzato nell'angolo in alto a sinistra dello schermo.

AZIONE: premere la barra spaziatrice.

RISULTATO: il messaggio viene ora mostrato in video inverso.

AZIONE: premere nuovamente la barra spaziatrice.

RISULTATO: il messaggio viene nuovamente visualizzato normalmente.

7.11 SCROLLING

FORMATO: direzione, W,sr,sc,ec,r

oppure: direzione B,sr,sc,ec,r

SCOPO: effettuare lo scroll di una definita porzione dello schermo.

Il SIMONS' BASIC mette a vostra disposizione un comando per effettuare lo scroll di aree specificate dello schermo in qualunque direzione desiderata. Il primo parametro del comando vi permette di specificare la direzione dello scroll che può essere: LEFT (sinistra), RIGHT (destra), UP (alto) e DOWN (basso).

Il secondo parametro del comando, W o B, indica lo scroll con "wrap round" oppure con "blanking". Nel primo caso, i caratteri che escono dall'area di scroll da un lato, entrano da quello opposto; nel secondo caso, invece, i caratteri che escono da un lato dell'area di scroll, non ricompaiono.

I parametri successivi, sr ed sc, definiscono le coordinate di riga e di colonna del punto in cui inizia l'area di scroll, mentre ec ed er definiscono il punto di fine della stessa. È possibile combinare assieme più comandi di scroll per effettuare contemporaneamente lo scroll di diverse aree, in differenti direzioni. Le dimensioni massime di ogni area di scroll non possono essere superiori a 24 righe per 23 colonne. Si noti anche che lo scroll non può essere effettuato nei modi grafici hi-rel e multi-col.

ESEMPIO: effettuare lo scroll di due porzioni di schermo in due direzioni differenti.

INTRODURRE:
```
10 PRINT "< SHIFT CLR/HOME >"
20 FOR X = 0 TO 39
30 Y = INT(10*SIN(X/π)) + 12
40 PRINT AT(X,y)"*"
50 NEXT
60 LEFTW 0,0,20,20,25
70 GOTO 60
```

BATTERE: RUN < RETURN >

RISULTATO: appare sullo schermo una curva che effettua lo scroll sia a destra che a sinistra, contemporaneamente.

7.12 MEMORIZZAZIONE E RICHIAMO DI UNO SCHERMO

7.12.1 SCRSV

FORMATO: SCRSV 2,8,2,"nome,S,W"

oppure: SCRSV 1,1,1,"nome"

SCOPO: memorizzare i dati presenti su uno schermo low-resolution.

Il comando SCRSV vi permette di memorizzare i dati presenti su uno schermo a bassa risoluzione, su cassetta o floppy disk. Il primo valore che segue il comando è il numero del file logico e comanda al COMMODORE 64 di aprire un canale sull'unità a floppy od a cassetta. Questo parametro vale 1 per la cassetta ed 8 per il floppy. Il terzo parametro è l'indirizzo secondario, cioè un'istruzione speciale che indica al computer come memorizzare l'informazione. Ad esempio, ponendo questo valore pari ad 1 per la cassetta, indica al computer di scrivere il file e quindi di porre il marker di end-of-file al termine del nastro, quando il file viene chiuso. Il parametro 'nome' è il nome con cui volete identificare quel particolare schermo di dati. Esso deve essere unico, per poter effettuare la la memorizzazione e la ricerca del file in modo determinato. Lo stesso nome dovrà essere poi usato con il comando SCRLD (paragrafo seguente) per richiamare e visualizzare quello schermo. Il parametro S indica che l'accesso al file è sequenziale, mentre il parametro W indica che il file deve essere scritto anzichè essere letto da. Ogni schermo memorizzato occupa circa 5 blocchi. Si noti che i parametri risultano separati da delle virgole, mentre il nome ed i parametri S e W sono racchiusi tra virgolette.

ESEMPIO: disegnare la bandiera italiana e memorizzarla su floppy disk.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 FILL 6,10,20,4,160,2
30 FILL 10,10,20,4,160,1
40 FILL 14,10,20,4,160,5
50 SCRSV 2,8,2,"TRICOLORE,S,W"
60 GOTO 60
```

BATTERE: RUN < RETURN >

RISULTATO: la bandiera italiana viene visualizzata e quindi memorizzata su floppy disk.

7.12.2 SCRLD

FORMATO: SCRLD 2,8,2,"nome"

oppure: SCRLD 1,1,0"nome"

SCOPO: richiamare uno schermo di dati memorizzato.

Il comando SCRLD vi permette di richiamare e di visualizzare uno schermo che era stato memorizzato con il comando SCRSV (paragrafo precedente). Il primo valore che segue il comando è il numero del file logico. Esso comanda al COMMODORE 64 di aprire un canale dati sull'unità a cassette o a floppy disk. Il secondo valore specifica il dispositivo sul quale si desidera memorizzare lo schermo, e vale 1 per l'unità a cassetta ed 8 per quella a floppy disk. Il terzo valore rappresenta l'indirizzo secondario, cioè una istruzione speciale che indica al computer che l'informazione deve essere caricata nella stessa area di memoria che occupava prima che fosse stata memorizzata. L'ultimo parametro è il nome con cui avevate identificato quello schermo durante la sua memorizzazione, e deve essere compreso tra virgolette.

ESEMPIO: richiamare e visualizzare sullo schermo il programma memorizzato su floppy nel paragrafo precedente.

BATTERE: SCRLD 2,8,2,"TRICOLORE" < RETURN >

RISULTATO: la bandiera italiana viene chiamata dal floppy e visualizzata nuovamente sullo schermo.

7.13 STAMPA DEI DATI PRESENTI SULLO SCHERMO

7.13.1 INTRODUZIONE

Il SIMONS' BASIC ha un potente set di comandi che permette di utilizzare una stampante seriale per riprodurre le informazioni presenti sia su schermo normale, che grafico. Questi comandi sono estremamente utili nella progettazione grafica di bozzetti e nella produzione di diagrammi ed istogrammi, ad esempio nella rappresentazione statistica di variabili.

7.13.2 COPY

FORMATO: COPY

SCOPO: riprodurre una stampa su carta di ciò che è rappresentato sullo schermo in modo grafico.

COPY invia il contenuto di uno schermo grafico su stampante seriale. Si noti che, se avete usato il comando CIRCLE (paragrafo 6.5.11) per tracciare delle circonferenze sullo schermo, dovete cambiare i raggi che avete utilizzato per ottenere le stesse circonferenze anche sulla stampante. Per stampare una circonferenza, il raggio x deve essere uguale al raggio y. Per visualizzare la circonferenza dello schermo, occorre riportare i raggi ai loro valori originali.

ESEMPIO: visualizzare un diagramma a torta ellittico su schermo grafico high-rel, e quindi stamparlo su carta con forma circolare.

INTRODURRE:
```
  10 HIRES 0,1:MULTI 5,4,6
  20 CIRCLE 80,100,78,78,1
  30 ANGL 80,100,120,78,78,1
  40 ANGL 80,100,160,78,78,1
  50 ANGL 80,100,220,78,78,1
  60 ANGL 80,100,330,78,78,1
  70 PAINT 90,35,1
  80 PAINT 60,60,3
  90 PAINT 90,120,2
 105 LOW COL 7,4,6
 110 PAINT 80,110,1
 120 COPY
1000 GOTO 1000
```

BATTERE: RUN < RETURN >

RISULTATO: viene dapprima visualizzato un diagramma a torta ellittico sullo schermo, e quindi lo stesso viene stampato perfettamente circolare.

7.13.3 HRDCPY

FORMATO: HRDCPY

SCOPO: Riprodurre una stampa su carta di ciò che è rappresentato sullo schermo in modo low-resolution.

HRDCPY vi permette di riprodurre uno schermo low-rel su una stampante seriale. Questo comando è particolarmente utile per stampare moduli preformati, fatture ed altri documenti di questo tipo.

ESEMPIO: visualizzare un messaggio sullo schermo e quindi riprodurlo con la stampante.

INTRODURRE:
```
10 PRINT"< SHIFT CLR/HOME >"
20 PRINT AT(5,8)"SIMONS' BASIC":PRINT
30 PRINT"LA POTENTE ESPANSIONE DEL BASIC"
40 HRDCPY
50 END
```

BATTERE: RUN < RETURN >

RISULTATO: viene visualizzato sullo schermo il messaggio:

SIMONS' BASIC
LA POTENTE ESPANSIONE DEL BASIC

e quindi lo stesso viene stampato con la stampante seriale COMMODORE.

CAPITOLO 8
SPRITE E SIMBOLI GRAFICI DEFINITI DALL'UTILIZZATORE

8.1 INTRODUZIONE

Il capitolo 8 contiene i comandi che vi permettono di generare ed animare gli "sprite" e di generare dei simboli grafici secondo le vostre specifiche. Il capitolo è diviso in due parti, ognuna riservata ad uno di questi argomenti.

8.2 SPRITE

8.2.1 INTRODUZIONE

Uno sprite è un oggetto programmabile che può assumere un vasta gamma di forme. Questo oggetto può muoversi su tutto lo schermo, dicendo semplicemente al computer dove volete che si trovi lo sprite (una descrizione più dettagliata degli sprite si trova nella "Guida per l'utente" del vostro COMMODORE 64).

Uno sprite è chiamato nell'ambito del SIMONS' BASIC 'moveable object block' o MOB. È possibile creare ed animare contemporaneamente sullo schermo fino ad 8 MOB; i MOB possono essere visualizzati sia in modo normale, che in modo grafico. Inoltre vi sono due tipi di MOB, uno in high-rel ed uno in multi-col: il primo tipo occupa 24 punti di larghezza x 21 di altezza ed ogni punto è, in questo caso, largo un pixel; il secondo tipo di MOB, invece, è largo 12 Punti x 21 di altezza ed ogni punto è, in questo caso, largo due pixel. Mentre un MOB high-resolution può essere di un solo colore, un MOB multicolour può essere di tre colori diversi, ovviamente tra i 16 colori disponibili per il COMMODORE 64.

Nel BAISC standard, la generazione e l'animazione degli sprite richiede numerose istruzioni POKE. Il SIMONS' BASIC sostituisce i comandi POKE con dei semplici comandi.

Il comando DESIGN vi permette di specificare la locazione della memoria del COMMODORE 64 dove sono memorizzati i dati di ogni MOB. Ogni MOB viene progettato su una griglia, all'interno del listing stesso del programma, in modo che possiate controllarne l'aspetto, prima che siano usati. I MOB possono essere utilizzati sia su schermo normale, che in unione a schermi grafici high-resolution e multi-colour. Il comando MOB SET sceglie un determinato MOB e gli attribuisce il suo colore fondamentale. CMOB, invece, permette di assegnare al MOB due colori supplementari quando si progettano MOB multi-colour. Il comando MMOB vi permette di visualizzare e/o muovere un determinato MOB in uno specificato punto dello schermo. RLOCMOB, invece, vi permette di muovere un MOB da una locazione all'altra dello schermo, con i comandi DETECT e CHECK potete determinare dove un MOB è entrato in collisione con un altro MOB, oppure con un elemento su schermo normale.

Si noti che i comandi descritti in questo capitolo possono essere utilizzati solo come parte di un programma, e non direttamente da tastiera.

Gli esempi utilizzati in questo capitolo del manuale, riuniti tutti assieme al termine dello stesso capitolo, costituiscono un programma completo per visualizzare due MOB sullo schermo. Per tale motivo, siete pregati di non usare il comando NEW tra gli esempi, nè di lanciare le parti di programma, finchè non vi diremo come fare.

8.2.2 DESIGN

FORMATO: DESIGN c,ad

oppure: DESIGN c,sa + gc

SCOPO: riservare in memoria dello spazio per un MOB.

Il comando DESIGN riserva nella memoria del COMMODORE 64 lo spazio sufficiente a contenere il MOB che desiderate creare. Ogni MOB utilizza 64 byte. Il primo parametro del comando DESIGN specifica se il MOB è in modo high-resolution o in modo multi-colour, che sono indicati rispettivamente dai valori Ø e 1. Il secondo parametro, ad, definisce l'indirizzo di partenza in memoria del primo byte del MOB. Questo numero deve essere un multiplo di 64, e deve essere compreso nella gamma 2048 e 16319; esso può essere introdotto sia in forma decimale, che esadecimale. Se il valore è esadecimale, deve essere preceduto dal carattere $. Se il MOB deve essere usato su schermo high-resolution, dovete aggiungere un valore costante pari a 49152 in base decimale, o $CØØØ, in base esadecimale.

Ogni area di 64 byte disponibile in memoria per un MOB è chiamata 'blocco'. Se dividete per 64 l'indirizzo di partenza di un MOB, ottenete il numero del blocco. Questo numero è usato nel comando MOB SET (paragrafo 8.2.5) per inizializzare il MOB.

NOTA

Il valore della costante grafica NON DEVE essere aggiunto all'indirizzo di partenza per calcolare il numero del blocco.

Le aree disponibili nella memoria del COMMODORE 64 per i MOB e i numeri di blocco associati sono elencati di seguito:

NUMERI DI BLOCCO	LOCAZIONE DI MEMORIA
32-63	2Ø48-4Ø95
128-255	8192-16383

Se avete usato il comando MEM (paragrafo 8.3.2), solamente i blocchi 192-255 sono disponibili per memorizzare i MOB.

NOTA

Potete memorizzare tanti MOB quanti ne contiene la memoria del vostro COMMODORE 64. Tuttavia, ne potete visualizzare contemporaneamente solo otto. Durante il corso del programma, potete sostituire uno dei MOB presenti sullo schermo con un altro presente in memoria, usando l'indirizzo di partenza della sua locazione nella memoria stessa.

ESEMPIO: riservare in memoria lo spazio per un MOB high-resolution, su schermo normale.

INTRODURRE: 9Ø DESIGN Ø,2Ø48

RISULTATO: quando viene creato il MOB hi-rel, i dati ad esso relativi verranno memorizzati a partire dalla locazione 2Ø48 in avanti, nel blocco 32, cioè 2Ø48 diviso 64.

ESEMPIO: riservare in memoria lo spazio per un MOB multi-colour, su schermo normale.

INTRODURRE: 32Ø DESIGN 1,2112

RISULTATO: quando viene creato il MOB lo-col, i dati ad esso relativi verranno memorizzati a partire dalla locazione 2112 in avanti, nel blocco 33, cioè 2112 diviso 64.

8.2.3 @

FORMATO: @.....

oppure: @.....

SCOPO: impostare la griglia su cui va disegnato il MOB.

Il comando @ vi permette di predisporre una griglia su cui disegnare il MOB. La griglia è larga 24 punti per MOB hi-rel e 12 punti per MOB locol. In entrambi i csi, è alta 21 linee.

NOTA

Assicuratevi che ogni numero di linea della griglia abbia lo stesso numero di cifre, cioè due, o tre o quattro. In tal modo, eviterete l'indent delle righe, rendendo più sempli il progetto del MOB.

Come spiegato nel paragrafo 8.1, i MOB hi-rel possono essere progettati in un solo colore, mentre quelli mu-col, con tre colori differenti. L'unico colore del MOB hi-rel ed il colore primario del MOB mu-col vengono definiti per mezzo del comando MOB SET (paragrafi successivi). Invece, i due colori supplementari del MOB mu-col sono assegnati con il comando CMOB (paragrafo 8.2.4). Per definire il colore di ogni punto delMOB mu-col si utilizza uno dei caratteri elencati nella tabella seguente:

MOB HIGH-RESOLUTION

CODICE COLORE	COLORE CORRISPONDENTE
B	colore assegnato con il comando MOB SET

MOB MULTI-COLOUR

CODICE COLORE	COLORE CORRISPONDENTE
B	colore 1 nel comando CMOB
C	colore assegnato con il comando MOB SET
D	colore 2 nel comando CMOB

Inoltre, in ognuno dei due tipi di MOB, potete lasciare il colore dello sfondo semplicemente non introducendo nessun carattere.

ESEMPIO: progettare un MOB hi-rel

INTRODURRE:

```
5 PRINT"<SHIFT CLR/HOME>"
10 REM"*** EXAMPLE OF MOBS ***
80 REM"*** DESIGN THE MOBS ***
90 DESIGN 0,2048
100     ........BBBBB.........
110     ........BB...BB........
120     .......BB.....BB.......
130     ........BB...BB........
140     ........BBBBB.........
150     ...........B...........
160     ....BBBBBBBBBBBBBBB....
170     ....BBBBBBB.BBBBBBB....
180     ....BBBBBB..BBBBBBB....
190     ...BBB..BBB.BBB..BBBB...
200     ...BBB..BBB.BBB..BBBB...
210     ...BBBBBBB...BBBBBBB...
220     ..BBBBBBBBBBBBBBBBBBB..
230     ..BBB...B...B..BBB.BBB..
240     ..BBBB.BB.B.B.B.BB.BBB..
250     .BBBBB.BB.B.B.B.BB.B.BBB.
260     .BBBBB.BB...B.BBB..BBBB.
270     .BBBBBBBBBBBBBBBBBBBBB.
280     .BBBBBBBBBBBBBBBBBBBBB.
290     .......................
300     .......................
```

RISULTATO: quando verrà lanciata questa parte del programma, il tracciato della griglia verrà memorizzato come MOB nel blocco di memoria 32.

ESEMPIO: progettare un MOB multi-colour.

INTRODURRE:

```
320 DESIGN 1,2112
400     ......BB....
410     .....BCDB...
420     ....BBCCCBBB
430     ...BBCCCB...
440     .....BCB....
450     .BB..BCB....
460     BCCB.BCB....
470     .BCB.BCB....
480     ..BB.BCB....
490     ...BBCCCB...
500     ..BDCDCDCB..
510     .BDCDCDCDCB.
520     .BCDCDCDCCB.
530     ..BCDCDCBB..
540     ...BBBBB....
550     ....B..B....
560.    ...B....B...
570     ..B......B..
580     ...B......B.
590     ..BBB....BBB
600     .....B.....B
```

RISULTATO: quando verrà lanciata questa parte di programma, il tracciato sulla griglia verrà memorizzato come MOB nel blocco di memoria 33.

8.2.4 CMOB

FORMATO: CMOB C1,c2

SCOPO: definire i colori di un MOB mu-col

Il comando CMOB vi permette di definire due colori addizionali per un MOB multi-colour, cioè i colori di quei punti che saranno indicati con le lettere B e D nella griglia del MOB stesso (paragrafo 8.2.3).

ESEMPIO: continuando con il programma precedente, assegnate i colori nero e verde al MOB mu-col.

INTRODURRE: 610 CMOB 0,5

RISULTATO: quando il MOB mu-col verrà visualizzato, tutti i punti indicati con B saranno neri, mentre quelli indicati con D saranno verdi.

8.2.5 MOB SET

FORMATO: MOB SET mb,blk,col,pr,res

SCOPO: inizializzare un MOB

Come dice il nome stesso, il comando MOB SET inizializza un MOB. Il parametro Mb specifica il numero del MOB desiderato e deve essere unicamente individuato per ogni MOB diverso. Il MOB con il numero più basso ha priorità più elevata su quelli con numero più alto: ciò significa che se due o più MOB si incontrano sullo schermo, quello a priorità maggiore (numero più basso) passa sopra a quello con priorità minore.

Il secondo parametro del MOB SET, blk, individua il blocco di memoria da cui si desidera prelevare il MOB (paragrafo 8.2.2). Il parametro successivo, col, definisce il colore principale del MOB, cioè il colore assegnato ad ogni punto del MOB indicato con B, in modo hi-rel, e con C, in modo mu-col.

Il parametro pr specifica la priorità del MOB relativamente allo schermo, cioè se il MOB passa SOPRA o SOTTO i caratteri presenti sullo schermo. Per pr = Ø, il MOB ha priorità sullo schermo, mentre per pr = 1, i dati sullo schermo hanno priorità sul MOB. L'ultimo parametro del comando MOB SET, res, indica se il MOB deve essere creato in modo hi-rel (Ø) o mu-col (1).

ESEMPIO: richiamare il MOB hi-rel memorizzato da programma nei paragrafi precedenti.

INTRODURRE: 7ØØ MOB SET Ø,32,Ø,1,Ø

RISULTATO: quando gira questa parte del programma, il MOB hi-rel numero Ø, memorizzato nel blocco 32, viene inizializzato. Quando viene visualizzato, il MOB è colorato in nero e passa sopra tutti i dati presenti sullo schermo.

ESEMPIO: richiamare il MOB multi-colour memorizzato da programma nei paragrafi precedenti.

INTRODURRE: 71Ø MOB SET 1,33,2,Ø,1

RISULTATO: quando gira questa parte del programma, il MOB mu-col numero 1 memorizzato nel blocco 33, viene inizializzato. Quando viene visualizzato, il MOB è colorato in rosso e passa sopra tutti i dati presenti sullo schermo.

8.2.6 MMOB

FORMATO: MMOB mn,x1,x2,y2,espansione, velocità

SCOPO: visualizzare e/o animare un MOB.

Il comando MMOB vi permette di visualizzare un MOB in un punto dello schermo e quindi, se lo desiderate, di spostarlo in un'altra posizione. Il primo parametro, mn, indica il numero del MOB che desiderate visualizzare ed animare.

I parametri x1 ed y1 sono le coordinate del punto dello schermo dove il MOB verrà visualizzato prima del movimento, mentre x2 ed y2 sono le coordinate del punto finale del movimento. Se non desiderate spostare il MOB, ponete semplicemente gli stessi valori per le due coppie di coordinate.

Il parametro "espansione" definisce le dimensioni con cui il MOB viene visualizzato. I valori che può assumere sono riportati nella sottostante tabella, accanto alle dimensioni da essi individuate.

VALORE DEL PARAMETRO	DIMENSIONI ASSUNTE
Ø	il MOB viene visualizzato con le sue dimensioni normali
1	il MOB viene espanso sull'asse x il doppio della sua dimensione normale
2	il MOB viene espanso suill'asse y il doppio della sua dimensione normale
3	il MOB viene espanso su entrambi gli assi il doppio delle sue dimensioni normali.

Il parametro "velocità" specifica la velocità con la quale si muove il MOB.

ESEMPIO: muovere i MOB relativi al programma precedente.

INTRODURRE: 8ØØ MMOB 1,Ø,Ø,2ØØ,2ØØ,2,2Ø
81Ø MMOB Ø,Ø,Ø,185,7Ø,3,2Ø

RISULTATO: quando questa parte del programma viene eseguita, il MOB hi-rel viene visualizzato espanso sull'asse y, nella parte alta dello schermo. Il MOB mu-col appare invece rappresentato con dimensioni doppie di quelle normali, nella parte inferiore dello schermo.

8.2.7 RLOCMOB

FORMATO:
RLOCMOB mn,x,y,espansione,velocità

SCOPO: spostare un MOB tra due differenti locazioni dello schermo.

RLOCMOB vi permette di far muovere un MOB già presente sullo schermo in una diversa posizione dello stesso. I parametri x ed y sono le coordinate sullo schermo del punto in cui volete far muovere il MOB. Gli altri parametri hanno lo stesso significato di quelli utilizzati nel comando MMOB (paragrafo precedente).

ESEMPIO: far muovere il MOB del programma precedente.

INTRODURRE:
```
820 FOR I = 1 TO 20:X = 150 ★ INT(RND (1)) + 50
830 RLOCMOB 1,X,200,2,10
840 RLOCMOB 0,X-15,70,3,10
850 NEXT
```

RISULTATO: quando viene eseguita questa parte del programma, i due MOB sembrano inseguirsi sullo schermo.

8.2.8 DETECT

FORMATO: DETECT n

SCOPO: attivare la rilevazione di collisioni fra MOB.

Il comando DETECT attiva la rilevazione di collisioni fra MOB. Assegnando ad n il valore 0, si comanda al COMMODORE 64 di rilevare le collisioni tra un MOB ed un altro; se invece n = 1, viene attivata la rilevazione tra un MOB ed i caratteri presenti sullo schermo. Si noti che il comando deve sempre essere usato due vole, la prima per cancellare l'area della memoria dove viene memorizzata l'avvenuta collisione (chiamata registro di collisione degli sprite) e la seconda volta per attivare la rilevazione stessa.

ESEMPIO: cancellare il registro di collisione degli sprite relativo al programma precedente.

INTRODURRE:
```
825 DETECT 0.
```


8.2.9 CHECK

FORMATO: IF CHECK (mn1,mn2) = 0 THEN azione

oppure: IF CHECK (0) = 0 THEN azione

SCOPO: verificare se sono avvenute delle collisioni, MOB.

Il comando CHECK vi permette di verificare se sono avvenute delle collisioni fra MOB, oppure fra un MOB e dei dati sullo schermo. I MOB sui quali desiderate effettuare il controllo sono identificati dai parametri racchiusi fra parentesi, che seguono il comando.

Il parametro 0 tra parentesi comanda al COMMODORE 64 di verificare se c'è stata collisione fra un MOB ed i dati sullo schermo.

In entrambi i casi, se si è verificata la collisione, viene compiuta l'azione specificata nel relativo parametro del comando.

ESEMPIO: effettuare lo scroll del MOB hi-rel verso il basso e verificare se c'è stata collisione con il MOB mu-col.

INTRODURRE:
```
858 FOR P = 70 TO 200
859 DETECT 0:IF CHECK(0,1) = 0 THEN 865
860 RLOCMOB 0,X-15,P,3,10:NEXT
```

RISULTATO: quando viene eseguita questa parte del programma, se un MOB entra in collisione con l'altro MOB, viene eseguito il comando specificato nella linea di programma 865 (paragrafo seguente).

8.2.10 MOB OFF

FORMATO: MOB OFF mn

SCOPO: cancellare un MOB dallo schermo.

Il comando MOB OFF fa sparire un MOB dallo schermo. Il parametro mn specifica il numero del MOB che si desidera far sparire.

ESEMPIO: completare il programma costruito con gli esempi di questo capitolo.

INTRODURRE:
```
855 PRINT AT(X/8 +2,20)"OH C..."
856 PAUSE 1
865 PRINT AT(X/8 + 2,20)"OH CENTRATO!!!"
870 PAUSE 1
875 MOB OFF 1:RLOCMOB 0,X-15,196,3,10
```

BATTERE: RUN < RETURN >

AZIONE: tenete d'occhio l'uccello che cerca di sfuggire al peso.

RISULTATO: Lo struzzo viene schiacciato dal peso di una tonnellata ed appare sullo schermo il messaggio:

OH CENTRATO!!!

(ci scusiamo con gli appartenenti alla Protezione Animali).

8.3 CREAZIONE DI SIMBOLI DEFINITI DALL'UTENTE

8.3.1 INTRODUZIONE

Il SIMONS' BASIC vi fornisce la possibilità di sostituire i caratteri della tastiera del vostro COMMODORE 64 con dei simboli che definite voi stessi, secondo le vostre esigenze.

Il set di caratteri del COMMODORE 64 è memorizzato su una ROM (Read Only Memory: memoria che può solo essere letta e non utilizzata dall'utente per memorizzare dati). Per ridefinire i caratteri, dovete prima trasferire una copia degli stessi dalla ROM su una RAM (Random Access Memory: memoria ad accesso casuale, sulla quale l'utente può scrivere e leggere i propri dati). Ciò è possibile per mezzo del comando MEM, mentre con DESIGN potete scegliere il carattere che desiderate ridefinire, in funzione del suo codice di poke (l'elenco completo dei codici di poke si trova sulla "Guida per l'utente" del COMMODORE 64). Ogni carattere viene disegnato su una griglia e ciò vi consente di vedere il carattere stesso mentre lo state creando. Si noti che i caratteri definiti dall'utente NON POSSONO essere utilizzati su schermo grafico.

8.3.2 MEM

FORMATO: MEM

SCOPO: trasferisce i caratteri della ROM alla RAM.

Il comando MEM trasferisce una copia del set di caratteri del COMMODORE 64 da ROM su RAM, sotto il controllo del sistema operativo Kernal. Lo schermo viene spostato alla locazione $CC00 e i dati relativi agli sprite possono essere inseriti solamente dalla locazione $F000, cioè dal blocco 192 in avanti (paragrafo 8.2.1). Per riportare il computer alle condizioni di partenza, cioè al set di caratteri originali, occorre semplicemente tenere premuto il tasto RUN/STOP e quindi premere RESTORE.

Le figure 8.1 ed 8.2 mostrano la configurazione della memoria del COMMODORE 64 rispettivamente prima e dopo aver usato il comando MEM.

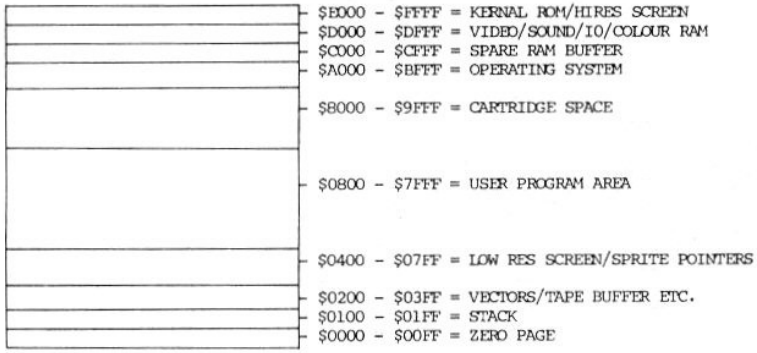

FIGURA 8-1 CONFIGURAZIONE DI MEMORIA PRIMA MEM

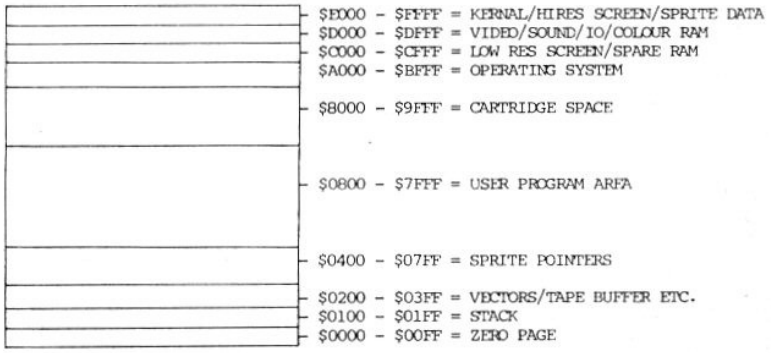

FIGURA 8-2 CONFIGURAZIONE DI MEMORIA DOPO MEM

NOTA
Il comando TRACE (paragrafo 2.11.1) non può essere usato se nel programma è presente il comando MEM.

ESEMPIO: copiare il set di caratteri da ROM su RAM.

INTRODURRE: 10 MEM

RISULTATO: quando viene eseguita questa parte del programma, il set di caratteri del COMMODORE 64 viene copiato su RAM per permettere la successiva ridefinizione.

8.3.3 DESIGN

FORMATO: DESIGN 2,$E000 + ch★8

SCOPO: specificare quale carattere deve essere sostituito dal carattere definito dell'utente.

Il comando DESIGN vi permette di specificare il carattere che deve essere sostituito da quello che desiderate definire. Notate che esiste anche un analogo comando DESIGN per la definizione degli sprite (paragrafo 8.2.2) che utilizza però un formato differente.

I caratteri definiti dall'utente possono essere usati solamente in modo low-resolution. La cifra 2 presente nel formato e che segue il comando stesso specifica al computer che i dati relativi al carattere definito dall'utente seguiranno immediatamente dopo. Ogni nuovo carattere occupa 8 byte in memoria ed il valore esadecimale $E000 è l'indirizzo di partenza dei dati relativi al carattere stesso. Il parametro ch è il codice poke del carattere esistente che desiderate ridefinire (un elenco completo dei codici poke è presente nella "Guida per l'utente" del vostro COMMODORE 64). Il nuovo carattere viene disegnato su una griglia di 8x8 punti (paragrafo seguente). Esso sarà visualizzato ogni volta che premerete il tasto relativo al carattere che avete ridefinito.

ESEMPIO: ridefinire il carattere "Z".

INTRODURRE: 20 DESIGN 2,$E000 + 26★8

RISULTATO: quando avete creato un nuovo carattere, esso verrà visualizzato ogni volta che premerete il tasto con la lettera "Z".

8.3.4 @

FORMATO: @.....

Il comando @ vi permette di impostare la griglia per disegnare un carattere ridefinito secondo le vostre esigenze. La griglia è larga 8 punti ed alta 8 linee. Il nuovo carattere viene costruito ponendo una lettera "B" sui punti desiderati della griglia.

NOTA
Assicuratevi che ogni numero di linea della griglia abbia la stessa lunghezza, cioè due o tre o quattro cifre. Facendo ciò eviterete di dover effettuare un indent di parte della griglia facilitando il disegno del carattere stesso.

ESEMPIO: ridefinire un carattere a forma di cappello a cilindro.

INTRODURRE:
```
DESIGN 2,$E000 + 26 * 8
30  ........
40  ........
50  ..BBBB..
60  ..BBBB..
70  ..BBBB..
80  BBBBBBBB
90  BBBBBBBB
95  ........
```

BATTERE: RUN < RETURN >

SCHERMO: READY

AZIONE: premere alcune volte il tasto Z.

RISULTATO: viene visualizzato il carattere ridefinito.

Per riottenere il normale set di carattere del COMMODORE 64, basta semplicemente tenere premuto il tasto RUN/STOP e quindi premere il tasto RESTORE.

CAPITOLO 9
PROGRAMMAZIONE STRUTTURATA

9.1 INTRODUZIONE

Uno dei principali problemi che si incontrano programmando il BASIC è la mancanza di un flusso strutturato per i programmi più complessi. L'uso di istruzioni GOTO e GOSUB spesso fa sì che anche un programma abbastanza semplice divenga pressochè incomprensibile perfino per chi lo ha scritto. Questo problema può in parte essere superato con l'uso di istruzioni REM che spiegano le funzioni delle varie routine, ma ciò, oltre che comportare una perdita di tempo, occupa anche notevoli porzioni della memoria del computer.

Il SIMONS' BASIC supera questi problemi grazie a dei particolari comandi di programmazione strutturata, che permettono di evitare istruzioni GOTO e GOSUB nei vostri programmi. Ad esempio, il comando PROC vi permette di etichettare ogni routine che usate (questa funzione è equivalente al comando chiamato Divisione di Procedura del COBOL). Le routine così individuate possono essere usate sia con il comando CALL, che con il comando EXEC.

Anche la struttura di loop FOR...NEXT è stata cambiata. Il comando REPEAT...UNTIL vi permette di eseguire una determinata procedura, il numero di volte che desiderate. Il comando LOOP...EXIT IF...END LOOP fornisce condizioni di test multiplo nell'ambito di un loop e la normale condizione di test IF...THEN è qui completata da ELSE, per cui è più semplice specificare (sulla stessa linea di programma) il percorso che deve seguire l'esecuzione del programma quando determinate condizioni vengono soddisfatte, oppure no. Infine, il comando RCOMP vi permette di riutilizzare l'ultimo comando IF...THEN...ELSE, senza dover ripetere il comando.

Notate che tutti i comandi descritti in questo capitolo possono essere usati solamente come parte di un programma.

9.2 CONDIZIONI DI TEST E LOOP DI PROGRAMMA

9.2.1 IF...THEN...ELSE

FORMATO: IF condizione THEN vera:ELSE:falsa

SCOPO: verificare il sussistere o no di una condizione e seguire una via se è vera ed un'altra se è falsa.

Il comando IF...THEN...ELSE funziona nello stesso modo del comando IF...THEN del BASIC standard. L'unica importante differenza consiste nel fatto che è possibile specificare, nella stessa riga, le azioni intraprese per entrambi i risultati del test (vero o falso). Notate che ELSE deve essere separato sia dal codice precedente, che da quello seguente, per mezzo del simbolo (:).

ESEMPIO: verificare la risposta ad una domanda.

INTRODURRE:
```
10 PRINT"POSSIEDI UN COMPUTER COMMODORE 64?"
20 PRINT"PER FAVORE, RISPONDI SÌ O NO (S/N)
30 FETCH"< CLR HOME >",1,A$
40 IF A$ = "S"THEN 60
50 IF A$ = "N"THEN 70:ELSE:GOTO 20
60 PRINT"CONGRATULAZIONI":END
70 PRINT"CONDOGLIANZE":END
```

BATTERE: RUN < RETURN >

RISULTATO: il messaggio appropriato viene visualizzato in funzione della risposta data SÌ oppure NO (S o N). Qualunque altra risposta viene ignorata.

9.2.2 REPEAT.....UNTIL

FORMATO: REPEAT loop UNTIL viene verificata la condizione.

SCOPO: continuare a far effettuare al programma un loop fino a quando non sia verificata una determinata condizione.

REPEAT...UNTIL permette di ottenere la stessa funzione del comando FOR...NEXT del BASIC standard, con la differenza che anzichè ripetere il loop un numero prefissato di volte, il loop viene ripetuto finchè non si verifica una determinata condizione. REPEAT avvia il loop ed UNTIL verifica se si è verificata la condizione, ad esempio se X < 10, che, se vera, provoca l'uscita del programma dal loop. Se, invece, la condizione non è verificata, il programma prosegue il loop.

NOTA
NON POTETE NIDIFICARE PIÙ DI NOVE LOOP.
SE EFFETTUATE CIÒ, APPARE IL MESSAGGIO:
? STACK TOO LARGE

ESEMPIO: stampare le lettere dell'alfabeto, da A a G.

INTRODURRE:
```
10 A = 65
20 REPEAT PRINT CHR$(A):A = A + 1:UNTIL A > 70
30 PRINT "FATTO!!!"
```


BATTERE: RUN < RETURN >

SCHERMO:
```
A

B
C
D
E
F
G
FATTO!!!
```

9.2.3 RCOMP

FORMATO: RCOMP:vero:ELSE:falso

SCOPO: rieseguire la condizione di test dell'ultima frase IF...THEN...ELSE effettuata nel programma.

RCOMP fa ripetere al programma l'ultima condizione di test di un comando IF...THEN...ELSE. In tal modo è evitata la necessità di riintrodurre il comando.

ESEMPIO: ripetere per tre volte la stessa condizione di test.

ESEMPIO: ripetere per tre volte la stessa condizione di test.

INTRODURRE:
```
10 INPUT A
20 IF A = 10 THEN PRINT "CIAO":ELSE:PRINT "CIAO"
30 RCOMP:PRINT "MIKE":ELSE:PRINT"JOHN"
40 RCOMP:PRINT "BENVENUTO":ELSE:PRINT "SPERO CHE CI RIVEDREMO"
50 GOTO 10
```

BATTERE: RUN < RETURN >

SCHERMO: ?

BATTERE: 10 < RETURN >

SCHERMO: CIAO MIKE BENVENUTO

BATTERE: 5 < RETURN >

SCHERMO: CIAO JOHN SPERO CHE CI RIVEDREMO

RISULTATO: quando viene introdotto un valore, lo stesso viene verificato tre volte e quindi viene visualizzato un messaggio che dipende dal valore stesso.

9.2.4 LOOP...EXIT IF...END LOOP

FORMATO: LOOP loop di programma EXIT IF condizione vera END LOOP

SCOPO: effettuare continuamente un loop finchè non viene verificata una determinata condizione.

LOOP...EXIT IF...END LOOP effettua un loop di programma in modo simile al comando REPEAT...UNTIL (paragrafo 9.2.2) con una differenza sostanziale: REPEAT...UNTIL effettua il test di verifica della condizione solamente sl termine delle istruzioni che compongono il loop, ed una volta sola. LOOP...EXIT IF...END LOOP, invece, permette di effettuare un numero qualunque di verifiche, anche all'interno del loop. Se una delle condizioni è soddisfatta, l'esecuzione del programma esce dal loop e continua dalla frase successiva a END LOOP. Se invece, nessuna delle condizioni e verificata, il programma prosegue nel loop.

NOTA
NON POTETE NIDIFICARE PIÙ DI NOVE LOOP.
SE EFFETTUATE CIÒ, APPARE IL MESSAGGIO:
? STACK TOO LARGE

ESEMPIO: visualizzare un carattere da tastiera, compreso tra A ed F.

BATTERE:
```
10 PRINT "INTRODURRE UNA LETTERA COMPRESA TRA A ED F"
20 LOOP
30 GET A$:IF A$ ="" THEN 30
40 EXIT IF ASC(A$) < 65
50 EXIT IF ASC(A$) > 70
60 PRINT A$
65 END LOOP
70 PRINT CHR$(13)"NON È NEL CAMPO": END
```

BATTERE: RUN < RETURN >

BATTERE: C < RETURN >

SCHERMO: C

BATTERE: B < RETURN >

SCHERMO: CB

BATTERE: S < RETURN >

SCHERMO: NON È NEL CAMPO

RISULTATO: il computer effettua una scansione della tastiera e tutte le lettere introdotte che appartengono al campo definito, vengono visualizzate sullo schermo; introducendo una lettera al di fuori di questo campo, il programma lascia il loop e viene stampato il messaggio.

9.3 PROCEDURE DI PROGRAMMA

9.3.1 INTRODUZIONE

Per facilitare la stesura di programmi in forma strutturata, il SIMONS' BASIC mette a vostra disposizione quattro comandi che vi permettono di etichettare le routine usate nei vostri programmi, in modo da chiamarle per nome, quando è richiesto il loro intervento, con il notevole vantaggio di evitare comandi GOTO E GOSUB. Questo tipo di routine viene chiamato "procedura" e, ogni procedura usata di frequente, può essere memorizzata in una biblioteca e quindi richiamata quando necessario. Il comando PROC (paragrafo 9.3.2) è usato per assegnare il nome alle procedure che poi possono essere eseguite con il comando CALL (paragrafo 9.3.4) oppure con il comando EXEC (paragrafo 9.3.5). CALL funziona come il comando GOTO del BASIC standard, cioè l'esecuzione del programma passa all'inizio della procedura e continua da quel punto, mentre il comando EXEC ha lo stesso comportamento del comando del BASIC standard GOSUB, cioè l'esecuzione passa alla procedura chiamata e, terminata quest'ultima, ritorna al programma principale, alla linea immediatamente successiva al comando EXEC. Il termine della procedura deve quindi essere sempre indicato per mezzo del comando END PROC (paragrafo 9.3.3) che ha lo stesso compito del comando RETURN in BASIC standard.

Gli esempi riportati in questo capitolo del manuale permettono, collegati assieme, di formare un programma completo. Per tale motivo, non usate il comando NEW e non fate eseguire il programma prima che vi indichiamo come fare.

NOTA
NON POTETE NIDIFICARE PIÙ DI NOVE LOOP. SE EFFETTUATE CIÒ, APPARE IL MESSAGGIO:
? STACK TOO LARGE

9.3.2 PROC

FORMATO: PROC nome

SCOPO: etichettare una routine.

PROC vi permette di etichettare una routine di programma e quindi, di richiamare la stessa con quel nome, quando necessario. Tutti i caratteri che seguono il comando PROC sulla stessa riga sono considerati parte del nome della procedura, quindi il comando PROC ed il nome della procedura non debbono essere seguiti da nessun altro codice di comando sulla stessa riga del programma.

ESEMPIO: assegnare l'etichetta "INTRODUZIONE DI NOMI" ad una routine di programma.

INTRODURRE: 100 PROC INTRODUZIONE DI NOMI

RISULTATO: quando il programma viene lanciato, questo comando assegna il nome INTRODUZIONE DI NOMI ad una procedura costituita dai comandi che seguono questa linea.

9.3.3 END PROC

FORMATO: END PROC

SCOPO: indicare il termine di una procedura.

END PROC indica la chiusura di una procedura che deve successivamente essere chiamata con il comando EXEC (paragrafo 9.3.5). Questo comando ha lo stesso effetto del comando in BASIC standard RETURN, cioè al termine della procedura, l'esecuzione del programma ritorna al programma principale, alla riga successiva al comando EXEC che ha chiamato la procedura.

ESEMPIO: preparare una procedura per introdurre il nome dell'utente come variabile.

INTRODURRE:
```
100 PROC INTRODUZIONE DI NOMI
110 PRINT "COME TI CHIAMI?"
120 FETCH "< CLR/HOME >",15,A$
130 END PROC
```

RISULTATO: quando viene lanciato il programma, la procedura INTRODUZIONE DI NOMI può essere richiamata con il comando EXEC (paragrafo 9.3.5).

9.3.4 CALL

FORMATO: CALL nome della procedura

SCOPO: trasferire l'esecuzione del programma ad una linea specificata.

Il comando CALL funziona come il GOTO del BASIC standard, tranne che viene usato il nome della procedura, anzichè il numero della linea di programma. Tutto ciò che segue il comando CALL sulla stessa linea di programma, viene interpretato come nome della procedura che deve essere chiamata. Per tale motivo, CALL ed il nome della procedura non debbono essere seguiti da nessun altro comando sulla stessa linea. La procedura chiamata deve essere, in questo caso, di tipo aperto, cioè non deve essere presente nessun comando END PROC.

ESEMPIO: inizio di un programma di sort.

INTRODURRE:
```
10 PRINT "SHIFT CLR/HOME >"
20 PRINT"QUANTI NOMI DESIDERI ORDINARE?"
30 PRINT"NON PIÙ DI 15 NOMI"
32 PRINT"E NON PIÙ DI DIECI CARATTERI"
34 PRINT"PER OGNI NOME"
40 FETCH"< CRSR/DOWN >",2,X
50 IF X < 16 THEN CALL INTRODUZIONE DI NOMI:ELSE:GOTO 10
60 PROC INTRODUZIONE DI NOMI
70 FOR I = 1 TO X
80 FETCH "< CLR/HOME >",10,A$(I):PRINT TAB (20) "VA BENE"
```

RISULTATO: se il numero dei nomi introdotti è inferiore a 16, il programma continua dalla procedura INTRODUZIONE DI NOMI alla linea 60.

9.3.5 EXEC

FORMATO: EXEC nome della procedura

SCOPO: chiamare una routine di programma e quindi ritornare alla linea successiva nel programma principale, dopo che la routine è stata completata.

EXEC ha la stessa funzione del comando GOSUB del BASIC standard, cioè l'esecuzione del programma passa ad una specificata routine e, dopo aver eseguito quest'ultima, ritorna alla linea del programma principale che viene immediatamente dopo la linea contenente il comando EXEC. Ciò avviene dopo che è stata raggiunta l'istruzione END PROC (vedi 9.3.3) nella routine. Ogni volta che viene trovato il comando EXEC, tutto ciò che lo segue viene interpretato come il nome della procedura da eseguire; per tale motivo, nessun altro codice di programma deve essere presente sulla stessa linea di programma.

ESEMPIO: completare il programma di sort dell'esempio precedente.

INTRODURRE:
```
 100 EXEC SORT
 110 PRINT "< CLR/HOMLE CRSR/DOWN >"
 120 FOR I = 1 TO 5:PRINT TAB(20)A$(I):NEXT
 130 END
 140 :
 150 :
1000 PROC SORT
1020 M = 1
1030 REPEAT
1040 T = 0:FOR I = 1 TO N - M
1050 IF A$(I) < A$(I + 1) THEN 1070
1060 W$ = A$(I):A$(I) = A$(I + 1):A$(I +1) = W$:T = 1
1070 NEXT I
1080 M = M + 1
1090 UNTIL T = 0
1110 END PROC
```

BATTERE: RUN < RETURN >

AZIONE: dopo l'avviso, introdurre un massimo di 15 nomi, premendo RETURN tra ognuno di essi.

RISULTATO: i nomi vengono ordinati alfabeticamente e quindi visualizzati.

9.4 VARIABILI DI PROGRAMMA

9.4.1 INTRODUZIONE

L'uso delle normali varibili del BASIC standard può diventare complicato soprattutto quando il loro numero è elevato. Il SIMONS' BASIC risolve questo problema mettendo a vostra disposizione due tipi di variabili, quelle LOCALI, da usarsi nell'ambito di una determinata routine, e quelle globali, da usarsi in tutto il programma. Ciò permette di ridurre il numero delle variabili necessarie e rende più facile la documentazione del programma stesso, risparmiando nel contempo memoria del calcolatore e, quindi, permettendo la stesura di programmi più lunghi.

Il valore di ogni variabile in un programma BASIC cambia in funzione di quando e di dove la variabile stessa è utilizzata. Il comando LOCAL vi permette di memorizzare i valori correntemente assegnati alle variabili, di cancellarli e quindi di utilizzare le stesse variabili in un'altra sezione del programma. Il comando GLOBAL assegna nuovamente quei valori che avevano le variabili prima che venisse eseguito il comando LOCAL.

9.4.2 LOCAL

FORMATO: LOCAL variabile 1, variabile 2, variabile 3.....

SCOPO: assegnare le variabili ad una specificata routine del programma.

Il comando LOCAL vi permette di cancellare i valori delle variabili precedentemente definite e di usare le stesse variabili, localmente in una determinata routine di programma. Il comando GLOBAL (paragrafo seguente) restituisce alle stesse variabili il loro valore originale.

NOTA

LE VARIABILI DEFINITE CON IL COMANDO LOCAL DEBBONO ESSERE STATE PRECEDENTEMENTE DICHIARATE. NON SEGUENDO QUESTA REGOLA, IL PROGRAMMA SI ARRESTA. PER SUPERARE CIÒ OCCORRE PREMERE IL TASTO RUN/STOP, SEGUITO DA RESTORE.

ESEMPIO: assegnare localmente delle variabili.

INTRODURRE:
```
10 REM "*** ESEMPIO DI LOCAL ***
20 PRINT"< SHIFT CLR/HOME >"
30 A$ = "VALORE INIZIALE":A% = 123:   A = 456.7
40 LOCAL A$,A%,A
50 A$ = "NUOVO VALORE              ":A% = 789:   A = 321.4
60 PRINT A$,A%,A
```


BATTERE: RUN < RETURN >

SCHERMO: NUOVO VALORE 789 321.4

RISULTATO: i valori originariamente assegnati alle varibili sono memorizzati. Vengono quindi assegnati nuovi valori alle variabili e questi valori sono stampati.

9.4.3 GLOBAL

FORMATO: GLOBAL

SCOPO: riassegnare i valori originari alle variabili definite locali.

Il comando GLOBAL provoca la cancellazione delle variabili usate come locali nell'ambito di una routine e la riassegnazione dei valori che avevano prima che fosse usato il comando LOCAL (paragrafo precedente).

ESEMPIO: usare il programma dell'esempio precedente, riassegnando i valori GLOBAL alle variabili definite LOCAL.

INTRODURRE:
```
10 REM"*** ESEMPIO DI LOCAL/GLOBAL ***
20 PRINT"< SHIFT CLR/HOME >"
30 A$ = "VALORE INIZIALE":A% = 123:   A = 456.7
40 LOCAL A$,A%,A
50 A$ = "NUOVO VALORE              ":A% = 789:   A = 321.4
60 PRINT A$,A%,A
70 GLOBAL
80 PRINT A$,A%,A
```

BATTERE: RUN < RETURN >

SCHERMO: NUOVO VALORE 789 321.4

VALORE INIZIALE 123 456.7

RISULTATO: i valori assegnati alle variabili prima del comando LOCAL sono stati nuovamente riassegnati alle stesse.

CAPITOLO 10
INTERCETTAZIONE DEGLI ERRORI

10.1 INTRODUZIONE

Il SIMONS' BASIC mette a vostra disposizione dei comandi che vi permettono di intercettare gli errori di programma, allo scopo di prevenire l'arresto (crash) degli stessi, portandone comunque a buon fine l'esecuzione. Il comando ON ERROR provoca il salto del programma ad un determinato punto dello stesso, quando si verifica un errore. Inoltre, vi sono due variabili, ERRLN ed ERRN. che contengono rispettivamente il numero della linea ed il numero dell'errore che eventualmente si è verificato. Controllando il valore in esse contenuto, è possibile prendere le azioni correttive appropriate e far visualizzare un messaggio di errore, secondo le vostre esigenze. OUT disattiva l'ultimo comando ON ERROR utilizzato, mentre NO ERROR riporta il COMMODORE 64 alle normali routine di gestione degli errori del BASIC.

10.2 ON ERROR

FORMATO: ON ERROR: GOTO numero di linea

SCOPO: intercettare gli errori del programma.

Il comando ON ERROR vi mette in grado di intercettare eventuali errori nei vostri programmi BASIC, impedendo il loro arresto indesiderato. Quando l'errore viene trovato, l'esecuzione passa alla linea indicata nel GOTO. Il numero dell'errore viene memorizzato nella variabile ERRN, mentre il numero della linea in cui si è verificato l'errore viene memorizzato nella variable ERRLN. Verificando il valore contenuto in ERRN, è possibile vedere se si è verificato un errore e quindi, in caso affermativo, prendere le dovute precauzioni, eventualmente facendo visualizzare dei messaggi di errore personalizzati.

NOTA

Dopo aver verificato la presenza di un errore ed aver preso le opportune precauzioni, dovete sempre usare il comando OUT (paragrafo seguente) prima di continuare l'esecuzione del programma. Questo comando deve essere usato anche se avete fermato il programma contenente il comando ON ERROR e desiderate effettuare l'editing dello stesso.

Gli errori che possono essere individuati con il SIMONS' BASIC sono elencati di seguito con i numeri ad essi associati:

NUMERO DI ERRORE	TIPO
1	Numero eccessivo di file
2	File aperto
3	File non aperto
4	File non trovato
5	Dispositivo non presente
10	Istruzione NEXT priva di FOR
11	Errore di sintassi
12	Istruzione RETURN senza GOSUB
13	Lettura dati
14	Valore non permesso
15	Overflow
16	Limiti di memoria raggiunti
17	Frase non definita
18	Pedice scorretto
19	Matrice ridimensionata
20	Divisione per zero
21	Direzione non permessa
22	Tipo non accettato
23	Stringa troppo lunga

Negli esempi che seguono, battete le informazioni esattamente come sono mostrate. Gli errori di battitura sono voluti per dimostrare come funzionano i comandi di intercettazione degli errori del SIMONS' BASIC.

ESEMPIO: intercettare un errore di SINTASSI e visualizzare un messaggio di errore definito dall'utente.

INTRODURRE:
```
  5 REM "*** ESEMPIO DI TRATTAMENTO DEGLI ERRORI ***
 10 ON ERROR: GOTO 100
 15 PRINT"< SHIFT CLR/HOME >"
 20 READ B
 25 PRINT B: GOTO 20
 30 DATA 1,2,3,4,5
100 IF ERRN = 11 THEN PRINT "ERRORE DI BATTITURA NELLA
    LINEA";ERRLN
```

BATTERE: RUN < RETURN >

SCHERMO: ERRORE DI BATTITURA NELLA LINEA 15

RISULTATO: poichè la battitura della parola in BASIC PRINT è risultata errata, il programma è passato ad eseguire la routine di trattamento dell'errore alla linea 100. Il programma ha cercato un errore di sintassi, cioè con il numero 11 e, poichè esso si è verificato, ha visualizzato il messaggio di errore definito dall'utente alla linea 100.

AZIONE: battere il tasto RUN/STOP.

SCHERMO: READY

ESEMPIO: intercettare un errore di tipo DATI.

BATTERE: OUT < RETURN >

(il comando OUT sarà descritto nel paragrafo seguente)

AZIONE: correggere l'errore nella linea 15 dell'esempio precedente e quindi introdurre la seguente linea di programma:

```
110 IF ERRN = THEN PRINT"NON VI SONO INFORMAZIONI SUFFI
CIENTI SULLA LINEA";ERRLN
```

BATTERE: RUN < RETURN >

SCHERMO:
```
1
2
3
4
5
NON VI SONO INFORMAZIONI SUFFICIENTI SULLA LINEA 20
```

RISULTATO: poichè vi sono solamente 5 elementi DATI, il programma salta alla routine di trattamento dell'errore alla linea 100 quando cerca di leggere il dato numero sei. La linea 110 verifica l'errore di tipo 13 e, poichè un errore di tale tipo è stato riscontrato, visualizza il messaggio di errore definito dall'utente nella linea stessa.

10.3 OUT

FORMATO: OUT

SCOPO: disattivare l'ultimo comando ON ERROR

OUT disattiva l'ultimo comando ON ERROR usato. Questo comando deve essere sempre usato per riportare il COMMODORE 64 alle normali routine di trattamento dell'errore, dopo aver usato il comando ON ERROR.

ESEMPIO: usando l'esempio di programma del paragrafo precedente, disattivare il comando ON ERROR.

AZIONE: introdurre la seguente linea corretta:

```
20 READ B: J = J + 1: IF J = 5 THEN OUT
```

BATTERE: RUN < RETURN >

SCHERMO:
```
1
2
3
4
5
?OUT OF DATA ERROR
```

RISULTATO: poichè vi sono solamente quattro elementi DATI, il programma subisce un crash quando cerca di leggere il quindi elemento. Il COMMODORE 64 visualizza pertanto il messaggio standard per questo tipo di errore.

10.4 NO ERROR

FORMATO: NO ERROR

SCOPO: ristabilire le normali routine di trattamento dell'errore di COMMODORE 64.

Il comando NO ERROR disattiva tutti i comandi di intercettazione degli errori del SIMONS' BASIC e riporta il controllo alle routine di trattamento degli errori del COMMODORE 64.

ESEMPIO: usando il programma dell'esempio precedente, riportare il controllo dell'errore al COMMODORE 64.

AZIONE:
```
 15 PRINT"< SHIFT CLR/HOME >"
100 NO ERROR:IF ERRN = 11 THEN PRINT"ERRORE DI BATTITURA
    NELLA LINEA"; ERRLN
```

AZIONE: battere RUN < RETURN >

SCHERMO: SYNTANX ERROR IN LINE 100

CAPITOLO 11
EFFETTI SONORI E MUSICALI CON IL SIMONS' BASIC

11.1 INTRODUZIONE

Tra le caratteristiche più interessanti del COMMODORE 64 è certamente la possibilità di sintetizzazione musicale. Con un po' di pratica è possibile riprodurre con il vostro computer il suono di numerosi strumenti musicali. Le possibilità del COMMODORE 64 sono molto estese e il SIMONS' BASIC non è stato progettato per sfruttare completamente dette possibilità: lo scopo dei comandi presenti nella cartuccia è essenzialmente quello di introdurvi alla programmazione dei suoni. Se desiderate saperne di più sulle possibilità musicali del COMMODORE 64, rivolgetevi al vostro rivenditore di fiducia che saprà consigliarvi sul software e sui libri più adatti a sviluppare in particolare questo argomento.

11.1.1 FORMAZIONE DEI SUONI

Suoni differenti sono prodotti fra frequenze differenti. Ad una frequenza più elevata, corrisponde una nota più acuta. Molti personal computer hanno possibilità audio, ma a differenza della maggior parte, il COMMODORE 64 ha la capacità di dare forma al segnale sonoro. In tal modo è possibile dire al computer come deve essere suonata ogni parte di un segnale a frequenza data. Il volume di una nota musicale o di un suono cambia dal primo momento in cui l'udite fino a quando non l'udite più. Tutti i suoni sono generati in quattro fasi: ATTACK, DECAY, SUSTAIN e RELEASE. Queste fasi messe assieme formano quello che si chiama inviluppo del suono. Ecco descritta la funzione di ognuna di queste fasi in un suono.

ATTACK

è la fase in cui il segnale passa dal volume nullo al volume massimo.

DECAY

è la fase in cui il segnale scende dal massimo del volume ad un valore circa metà.

SUSTAIN

è la fase in cui il volume rimane costante al suo valore medio.

RELEASE

è la fase in cui il segnale passa dal volume medio della fase precedente al volume nullo.

Per maggior chiarezza, la figura 11-1 rappresenta l'inviluppo di un suono di questo tipo.

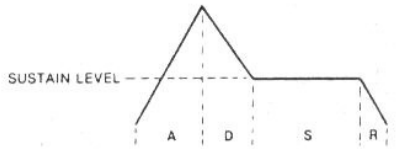

FIGURA 11-1 - INVILUPPO DI UN SUONO

11.1.2 TIPO DI ONDE

Il COMMODORE 64 vi permette di scegliere il tipo di onda sonora che desiderate usare per suonare la musica o riprodurre gli effetti sonori che avete creato. Ogni tipo di onda sonora produce effetti differenti. Le forme d'onda e gli effetti da esse prodotti sono descritti nel seguito. Viene anche illustrato un diagramma per ogni diversa forma d'onda.

TRIANGOLARE

Questa forma d'onda ha un basso contenuto in armoniche ed ha un suono tipo flauto.

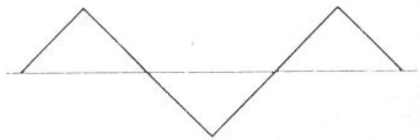

FIGURA 11-3 - TRIANGOLARE

A DENTE DI SEGA

Questa forma d'onda è ricca in armoniche ed ha un suono brillante, di ottoni.

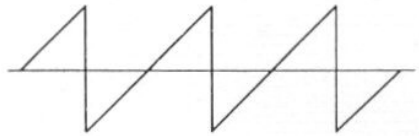

FIGURA 11-3 - ONDA SONORA A DENTE DI SEGA

IMPULSO/QUADRA

Questa forma d'onda vi permette di riprodurre numerosi toni diversi, da suoni brillanti a suoni cupi, da suoni nasali a suoni acuti.

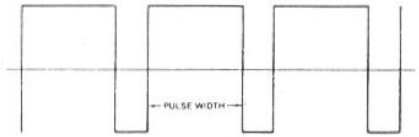

FIG. 11-4 - ONDA SONORA IMPULSO/QUADRA

RUMORE

Come il nome stesso suggerisce, questa forma d'onda permette di ottenere diversi tipi di rumore, per creare vari effetti sonori.

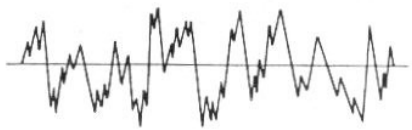

FIGURA 11-5 - ONDA DI RUMORE

11.1.3 PROGRAMMAZIONE DI EFFETTI SONORI

Nel BASIC standard, per suonare musica o creare degli effetti sonori con il COMMODORE 64 occorre usare numerosi comandi POKE e ciò è noioso e porta via molto tempo. Il SIMONS' BASIC risolve questo problema. Gli speciali comandi presenti nella cartuccia eliminano la necessità di accedere direttamente alla memoria con le istruzioni POKE.

Il comando VOL vi permette di definire il volume degli effetti sonori che desiderate riprodurre, mentre invece con WAVE potete scegliere il tipo dell'onda; il comand ENVELOPE vi permette invece di scegliere la durata delle varie fasi che compongono il segnale. Il comando MUSIC è usato per comporre la musica che poi potete riprodurre con il comando PLAY. Notate che questi comandi possono essere utilizzati solamente come parte di un programma, e non direttamente da tastiera.

Il paragrafo che segue descrive il formato e lo scopo dei comandi musicali del SIMONS' BASIC. Inoltre, viene anche riportato un breve esempio di ogni comando.

Gli esempi utilizzati nel paragrafo, riuniti assieme, formano un programma che suona un motivo. Non utilizzate quindi il comando NEW e non lanciate il programma prima che lo stesso sia stato terminato.

11.2 COMANDI MUSICALI

11.2.1 VOL

FORMATO: VOL n

SCOPO: scegliere il volume del suono

Il comando VOL vi permette di definire il livello del volume con cui saranno eseguiti i suoni o i brani musicali che seguono. Il livello va da Ø a 15, dove 15 rappresenta il suono a volume più elevato, mentre con un valore pari a Ø il volume del suono è nullo. Il valore di VOL rimane fissato finch non si utilizza un nuovo comando VOL.

ESEMPIO: impostare un volume pari a 15.

INTRODURRE: 1Ø VOL 15

RISULTATO: qualunque suono che segue questo comando viene riprodotto con il volume massimo.

11.2.2 WAVE

FORMATO: WAVE numero della "voce", numero binario.

SCOPO: scegliere la forma d'onda.

WAVE vi permette di scegliere la forma d'onda che desiderate usare per suonare la vostra musica o riprodurre gli effetti sonori (paragrafo 11.1.2). Il primo parametro del comando Wave specifica il generatore con cui sarà riprodotto il suono: il COMMODORE 64 ha tre generatori disponibili, ognuno dotato di nove ottave, che sono numerati da 1 a 3. È possibile comporre un suono con un generatore e quindi effettuare il mixaggio con un suono proveniente da un'altro genertatore.

Il secondo parametro del comando WAVE è un numero binario (notate che con il comando WAVE, questo numero non deve essere preceduto dal simbolo %. Questo numero dice al COMMODORE 64 come riprodurre ogni suono. Ognuno degli otto bit di cui è composto, ha una determinata funzione: per far eseguire queste ultime, occorre porre il bit relativo pari ad 1. I bit sono numerati da Ø a 7, procedendo da destra (Ø) a sinistra (7). Nella tabella che segue sono illustrate le funzioni relative ad ogni bit.

NUMERO DEL BIT	FUNZIONE
Ø	pone ad 1 bit di gate (non usato)
1	attiva la sincronizzazione
2	attiva la modulazione ad anello
3	pone ad 1 bit il test (mantenere a Ø)
4	attiva un'onda triangolare
5	attiva un'onda a dente di sega
6	attiva un'onda impulso/quadra
7	attiva un'onda di rumore

Queste funzioni sono descritte nel seguito con maggiori dettagli.

BIT Ø - BIT DI GATE

Quando questo bi è posto ad 1 su un COMMODORE 64 privo di SIMONS' BASIC, viene attivato il generatore di inviluppo e si dà il via alla produzione delle quattro fasi che compongono il suono. Tuttavia, poichè il SIMONS' BASIC lo pone automaticamente ad 1 eseguendo il comando PLAY (paragrafo 11.2.5), lasciate sempre questo bit a Ø.

BIT 1 - SINCRONIZZAZIONE

Ponendo ad 1 il bit 1, si sincronizza il suono di uno dei 3 generatori con quello di un altro. Ottenendo da un generatore una nota continua a frequenza costante e producendo un suono a frequenza variabile con il secondo, è possibile produrre una vasta gamma di armonie complesse. Per ottenere i migliori risultati, il suono a frequenza costante deve avere frequenza inferiore al valore minimo assunto dalla frequenza del suono variabile. Il generatore che scegliete per produrre la frequenza variabile determina il generatore che potete utilizzare come frequenza fissa. Le relazioni sono riportate nella tabella che segue:

GENERATORE A FREQUENZA VARIABILE	GENERATORE A FREQUENZA FISSA
1	3
2	1
3	2

Il numero del generatore viene specificato per mezzo del primo parametro del comando WAVE (vedi sopra).

BIT 2 - MODULAZIONE AD ANELLO

Ponendo ad 1 il bit 2, si attiva la modulazione ad anello. Questo effetto è simile alla sincronizzazione (paragrafo precedente), tranne per il fatto che in questo caso, sia la frequenza che l'ampiezza (cioè il volume) del suono possono essere variati contemporaneamente, ottenendo l'effetto "waw-waw". Variando la frequenza di un generatore e tenendo costante quella di un altro, si possono produrre in questo caso strutture sonore non armoniche, creando suoni di strumenti a percussione ed effetti speciali.

Per rendere udibile la modulazione ad anello, occorre impostare per il generatore a frequenza variabile un'onda triangolare, che viene poi sostituita da suono del secondo generatore modulato con il primo. Come nel caso della sincronizzazione, il generatore scelto con frequenza variabile determina quello a frequenza fissa. Le relazioni sono riportate nella tabella che segue:

GENERATORE A FREQUENZA VARIABILE	GENERATORE A FREQUENZA FISSA
1	3
2	1
3	2

BIT 3 - BIT DI TEST

Questo bit non viene utilizzato dal SIMONS' BASIC, per cui deve sempre essere tenuto a Ø.

BIT 4 - FORMA D'ONDA TRIANGOLARE

Ponendo ad 1 il bit 4, si ottiene una forma d'onda triangolare.

BIT 5 - FORMA D'ONDA A DENTE DI SEGA

Ponendo ad 1 il bit 5, si ottiene una forma d'onda a dente di sega.

BIT 6 - FORMA D'ONDA IMPULSO/QUADRA

Ponendo ad 1 il bit 6, si ottiene una forma d'onda costituita da impulsi quadri.

BIT 7 - RUMORE

Ponendo ad 1 il bit 7, si ottiene una forma d'onda di rumore.

NOTA
Ponendo ad 1 uno qualunque dei bit da 4 a 7, occorre tenere a Ø gli altri bit compresi in questo campo.

ESEMPIO: impostare un'onda triangolare per il generatore di suoni 1.

INTRODURRE: 20 WAWE 1,000,10000

RISULTATO: quando questa parte di programma viene eseguita, il pezzo musicale che segue il comando viene riprodotto usando una forma d'onda triangolare.

11.2.3 ENVELOPE

ENVELOPE: vn,a,d,s,r

SCOPO: definire le fasi di un suono.

Come spiegato nel paragrafo 11.1.1, il COMMODORE 64 vi permette di definire l'inviluppo del suono che desiderate riprodurre, per mezzo del comando ENVELOPE. Il parametro vn rappresenta il numero del generatore che desiderate utilizzare per riprodurre il suono. I parametri a,d ed r specificano rispettivamente la durata delle fasi di ATTACK, DECAY e RELEASE dei suoni che si desiderano riprodurre. La durata di queste fasi è specificata per mezzo di numeri variabili da 0 a 15, ai quali corrispondono i tempi elencati in tabella, espressi in millisecondi:

VALORE (TEMPO/CICLO)	DURATA ATTACK	DURATA DECAY/RELEASE (TEMPO DI CICLO)
0	2	6
1	8	24
2	16	48
3	24	72
4	38	114
5	56	168
6	68	204
7	80	240
8	100	300
9	250	750
10	500	1500
11	800	2400
12	1000	3000
13	3000	9000
14	5000	15000
15	8000	24000

Dopo aver terminato di introdurre Ogni stringa musicale, tenendo premuto il tasto SHIFT, premete CLR/HOME e battete la lettera G. Con ciò, otterrete il sincronismo della fase di RELEASE (paragrafo 11.1.1).

Il parametro s relativo alla fase di SUSTAIN è compreso tra 0 e 15 e definisce il livello di volume intermedio a cui sarà mantenuto il suono. Ciò ha lo stesso effetto di una variazione di volume (paragrafo 11.2.1) mentre la nota selezionata viene riprodotta; tuttavia, questa variazione è effettuata solamente per quel particolare generatore, mentre il volume degli altri rimane inalterato.

ESEMPIO: creare un inviluppo per il suono proveniente dal generatore 1.

INTRODURRE: 30 ENVELOPE 1,8,8,8,0

RISULTATO: tutte le note musicali che seguono questo comando saranno riprodotte dal generatore 1 con tempi delle fasi di ATTACK, DECAY e RELEASE corrispondenti ai valori in tabella e con un volume intermedio di SUSTAIN par ad 8.

11.2.4 MUSIC

FORMATO: MUSIC n, "stringa di note musicali"

oppure: MUSIC n, variabile + variabile + variabile....

SCOPO: comporre o creare degli effetti sonori.

Il comando MUSIC vi permette di comporre e successivamente di suonare della musica o di creare e riprodurre degli effetti sonori. Il primo parametro del comando determina la durata di una singola battuta e può variare tra 1 e 255, dove 1 rappresenta la durata massima e 255 quella minima. Dopo questo parametro, occorre impostare la stringa di note musicali desiderata, che può essere composta al massimo da 255 caratteri. All'inizio della stringa si specifica il numero del generatore che si desidera utilizzare con quella stringa stessa. Per fare ciò, occorre tenere premuto il tasto SHIFT, premere il tasto CLR/HOME (appare sullo schermo un simbolo a forma di cuore in campo inverso) e quindi introdurre il numero del generatore principale. Notate che, con ogni stringa di note, è possibile utilizzare solo un generatore musicale.

L'impostazione delle note avviene per mezzo delle lettere maiuscole da A a G, secondo la notazione musicale anglosassone; C è la prima nota di ogni ottava. Nella tabella che segue è riportata l'equivalenza tra la notazione aretina (comunemente usata nei paesi latini e che risale a Guido d'Arezzo) e la notazione anglosassone.

NOTAZIONE ANGLOSASSONE	NOTAZIONE ARETINA
C	DO
D	RE
E	MI
F	FA
G	SOL
A	LA
B	SI

Gli accidenti (bemolle e diesis) vengono impostati come segue: per introdurre una nota in diesis occorre tenere premuto il tasto SHIFT e quindi premere la lettera relativa alla nota stessa; invece, per introdurre, una nota in bemolle, basta introdurre la nota che la precedente nella scala musicale in diesis: ad esempio, un MI (E) bemolle viene introdotto come RE (D) diesis ed un SI (8) bemolle, come LA (A) diesis.

Le pause musicali sono introdotte con la lettera Z.

Per specificare l'ottava (tra le nove disponibili) a cui appartiene la nota, si introduce un numero compreso tra Ø ed 8, DOPO la nota stessa (ovviamente, le pause non hanno questo numero). La figura (durata) di ogni nota è determinata da un carattere di controllo che segue il numero che specifica l'ottava. Questo carattere viene introdotto per mezzo di un tasto funzione (da F1 ad F8). Nella tabella che segue sono riportate le figure relative alle note ed i tasti funzioni ad esse associate.

TASTO FUNZIONE	FIGURA DI NOTA
F1	semicroma (1/16 di battuta)
F3	croma (1/8 di battuta)
F5	semiminima (1/4 di battuta)
F7	minima (1/2 di battuta)
F2	semibreve (1 battuta)
F4	semibreve (2 battute)
F6	semibreve (4 battute)
F8	semibreve (8 battute)

Dopo aver specificato la figura dell'ultima nota della stringa, tenendo premuto il tasto SHIFT, premete CLR/HOME e battete la lettera G. In tal modo, otterrete il sincronismo nella fase di RELEASE (paragrafo 11.1.1).

ESEMPIO: comporre un motivo.

INTRODURRE:
```
40 A$ = "<SHIFT CLR/HOME>1Z<F1>C5<F1>E5<F1>F5<F1>"
50 A2$ = "G5<F7>C5<F1>E5<F1>F5<F1>G5<F7>C5<F1>E5
   <F1>F5<F1>G5<F3>E5<F3>C5<F3>E5<F3>D5<F5>E5<F1>
   E5<F1>D5<F1>C5<F7>C5<F1>"
60 A2$ = A2$ + "E5<F3>G5<F3>G5<F1>F5<F5>F5<F3>E5
   <F1>F5<F1>G5<F3>E5<F3>C5<F3>D5<F3>C5<F3>C5<F1>
   C5<F1>E5<F1>F5<F1>"
70 A3$ = "C5<F7>C5<F1><SHIFT CLR/HOME>G"
80 MUSIC 8,A$ + A2$ + A2$ + A3$
```

RISULTATO: quando questa parte del programma viene eseguita, le sequenze musicali vengono memorizzate nelle variabili di stringa A$,A2$ ed A3$.

11.2.5 PLAY

FORMATO: PLAY n

SCOPO: suonare la musica precedentemente composta, o riprodurre un effetto sonoro.

Il comando PLAY, come suggerisce il nome, vi permette di suonare la musica che avete composto. Il parametro che segue il comando, indica come deve essere suonata la musica nell'ambito del programma. Ponendo n=Ø la riproduzione viene disattivata; con n=1, il computer suona il pezzo e, quando lo stesso è terminato, riprnde l'esecuzione del programma, mentre per n=2, viene suonato il pezzo e, contemporaneamente, prosegue l'esecuzione del programma.

ESEMPIO: suonare il brano che è stato composto nei paragrafi precedenti, e quindi riprendere l'esecuzione del programma.

INTRODURRE:
```
 9Ø PLAY 2
1ØØ GOTO 1ØØ
```
AZIONE: battere RUN < RETURN >

RISULTATO: il computer suona il motivo "When The Saints Go Marching In"

ESEMPIO: creare un effetto sonoro.

INTRODURRE:
```
1Ø VOL 15
2Ø WAVE 1,1ØØØØØØØ
3Ø ENVELOPE 1,Ø,1Ø,Ø,Ø
4Ø MUSIC 5,"< SHIFT CLR/HOME > 1 C5 < F2 >"
45 REPEAT
5Ø PLAY 1
55 A = A + 1: UNTIL A= 5
```
AZIONE: battere RUN < RETURN >

RISULTATO: sono sparati cinque colpi di arma da fuoco

Notate che PLAY 2 NON PUÒ essere usato assieme ai modi grafici high-resolution e multi colour.

CAPITOLO 12
FUNZIONI READ

12.1 INTRODUZIONE

Il capitolo 12 illustra le quattro funzioni per la lettura di coordinate presenti nella cartuccia del SIMONS' BASIC. Se, nei vostri programmi, utilizzate dispositivi come la penna ottica, il joystick o le paddle, queste funzioni vi permettono di determinare le loro posizioni.

NOTA

La penna ottica può essere inserita SOLO nella porta N. 1, cioè quella più lontana dall'interruttore di accensione. Il joystick, invece, può essere inserito in una qualunque delle porte (vedi la "Guida per l'utente" del COMMODORE 64).

I comandi PENX e PENY vi permettono di determinare sullo schermo la posizione in cui si trova la penna ottica. Il comando POT determina sullo schermo la posizione delle paddle, mentre JOY determina in quale direzione è puntato il joystick.

12.2 PENX

FORMATO: variabile = PENX

SCOPO: leggere la coordinata x della penna luminosa.

Il comando PENX legge la posizione della penna ottica, rispetto al bordo sinistro dello schermo (asse y). Il valore può variare tra Ø e 32Ø. Nel paragrafo che segue, è riportato l'esempio relativo a questo comando. Notate che il valore PENX deve sempre essere letto prima di quello di PENY.

12.3 PENY

FORMATO: variabile = PENY

SCOPO: leggere la coordinata y della penna luminosa.

Il comando PENY legge la posizione della penna ottica, rispetto al bordo superiore dello schermo (asse x). Il valore può variare tra 0 e 200.

ESEMPIO: tracciare uno schizzo sullo schermo con la penna luminosa e quindi visualizzarlo sullo stesso.

AZIONE: introdurre il seguente programma e quindi lanciare lo stesso con RUN. Le istruzioni sono presentate sullo schermo.

```
10 REM:***LIGHT PEN PROGRAM***
20 HIRES0,1
30 TEXT10,10,"AFTER YOU HAVE TYPED IN 'RUN'",1,1,8
40 TEXT10,25,"YOU HAVE 15 SECONDS TO INSERT",1,1,8
50 TEXT10,40,"THE LIGHTPEN IN 'CONTROL PORT 1'",1,1,8
60 TEXT10,55,"WHEN YOUR DRAWING IS COMPLETE",1,1,8
70 TEXT10,70,"REMOVE LIGHTPEN FROM USER PORT",1,1,8
80 TEXT10,85,"AND PRESS SPACE-BAR FOR PRINT-OUT",1,1,8
90 PAUSE 15
100 HIRES0,1
110 LINE10,10,300,10,1:LINE300,10,300,180,1
120 LINE300,180,10,180,1:LINE10,180,10,10,1
130 GETA$:IFA$=""THEN130
140 IFA$=" "THEN160
150 PLOT (PENX+51)AND255,(PENY-50)AND255,1:GOTO 150
160 COPY:END
```


12.4 POT

FORMATO: variabile = POT(0)

oppure: variabile = POT(1)

SCOPO: determinare la posizione di una delle paddle.

Il comando POT vi permette di determinare di quanto è stata ruotata una delle paddle. Il valore è compreso nel campo 1-255. Il numero che segue il comando vi permette di specificare la paddle a cui si fa riferimento.

ESEMPIO: spostare un sprite usando una paddle.

INTRODURRE:

```
10 REM:****PADDLE PROGRAM****
20 REM:******BY K MORRIS*****
30 HIRES0,1
40 TEXT10,10,"BY MOVING THE PADDLES YOU CAN",1,1,8
50 TEXT10,25,"GET THE SPRITES TO MEET.",1,1,8
60 TEXT10,40,"WHY DON'T YOU TRY ?",1,1,8
65 TEXT10,55,"(SET PADDLE X=0 AND PADDLE Y=255)",1,1,8
70 DESIGN 0,64*32+49152
80 @.......................B
90 @.......................B
100 @.......................B
110 @...........BBBBBBBBBBBB
120 @...........B...........B
130 @...........B...........B
140 @...........B...........B
150 @...........B.......BBBBB
160 @...........B.......B...B
170 @...........B.......B...B
180 @.BBBBBBBBBBBBBBBBBBBBBB
190 @...........B.......B...B
200 @...........B.......B...B
210 @...........B.......BBBBB
220 @...........B...........B
230 @...........B...........B
240 @...........B...........B
250 @...........BBBBBBBBBBBB
260 @.......................B
270 @.......................B
280 @.......................B
290 DESIGN 0,64*33+49152
300 @B.......................
310 @B.......................
```

```
320 @B.......................
330 @BBBBBBBBBBBBB...........
340 @B...........B...........
350 @B...........B...........
360 @B...........B...........
370 @BBBBB.......B...........
380 @B...B.......B...........
390 @B...B.......B...........
400 @BBBBBBBBBBBBBBBBBBBBBBB.
410 @B...B.......B...........
420 @B...B.......B...........
430 @BBBBB.......B...........
440 @B...........B...........
450 @B...........B...........
460 @B...........B...........
470 @BBBBBBBBBBBBB...........
480 @B.......................
490 @B.......................
500 @B.......................
510 MOB SET 0,32,2,0,0
520 MOB SET 1,33,2,0,0
530 Z=10:W=245
535 MMOB0,Z,170,Z,170,3,50
536 MMOB1,W,170,W,170,3,50
540 X=INT(POT(0))
550 FORL=ZTOX
555 IFL=100THEN615
560 RLOCMOB0,L,170,3,50
580 NEXTL
590 Z=X:GOTO540
615 Y=INT(POT(1))
620 FORB=WTOY
625 IFB=145THEN700
630 RLOCMOB1,B,170,3,50
640 NEXT B
650 W=Y:GOTO615
700 TEXT105,170,"FIRE",1,3,8:PAUSE1
703 MOB OFF 0:MOB OFF 1
705 TEXT105,170,"FIRE",0,3,8
710 TEXT75,145,"BOOM !!!",1,5,20
716 BFLASH10,0,7
720 VOL15
730 WAVE 1,10000000
740 ENVELOPE 1,3,0,15,0
750 A$="□1Z■35■"
760 MUSIC 8,A$
770 PLAY 1
820 VOL0:BFLASH0:HIRES0,1:PRINT"□
```


12.5 JOY

FORMATO: variabile = JOYY

SCOPO: determinare il valore associato alla posizione che il joystick.

Il comando JOY vi permette di verificare la direzione in cui è puntato il joystick o se il pulsante "fire" è premuto. I valori associati a queste situazioni sono riportati in figura 12.1.

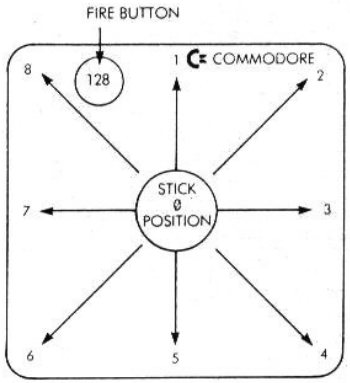

FIGURA 12-1 - VALORI ASSOCIATI ALLE POSIZIONI DEL JOYSTICK

ESEMPIO: disegnare una figura con il joystick, e quindi colorarla.

AZIONE: introdurre il seguente programma e quindi lanciare lo stesso con RUN. Le istruzioni sono presentate sullo schermo.

```
10 REM:********JOY STICK PROGRAM*****
11 HIRES0,1
12 LINE10,20,310,20,1:LINE310,20,310,160,1
13 LINE310,160,10,160,1:LINE10,160,10,20,1
15 TEXT88,30,"THE MOZOSKETCH",1,2,10
16 TEXT90,43,"──────────────",1,2,8
17 TEXT18,60,"DRAW A SHAPE MAKING SURE THAT",1,2,10
18 TEXT26,80,"THERE ARE NO GAPS,AND KEEP",1,2,10
19 TEXT60,100,"WITHIN THE BOUNDARIES.",1,2,10
20 TEXT105,140,"BY K J MORRIS",1,2,8:PAUSE 3
30 HIRES0,1
35 TEXT93,5,"THE MOZOSKETCH",1,1,8
40 TEXT4,18,"DRAW A SHAPE AND PRESS THE FIRE BUTTON",1,1,8
45 TEXT73,31,"TO SEE WHAT HAPPENS",1,1,8
50 LINE10,50,310,50,1:LINE310,50,310,150,1
55 LINE310,150,10,150,1:LINE10,150,10,50,1
80 X=160:Y=100
90 PLOTX,Y,1
100 IF JOY=1 THEN Y=Y-1:GOTO200
110 IF JOY=2 THEN Y=Y-1:X=X+1:GOTO200
120 IF JOY=3 THEN X=X+1:GOTO200
130 IF JOY=4 THEN X=X+1:Y=Y+1:GOTO200
140 IF JOY=5 THEN Y=Y+1:GOTO200
150 IF JOY=6 THEN Y=Y+1:X=X-1:GOTO200
160 IF JOY=7 THEN X=X-1:GOTO200
170 IF JOY=8 THEN X=X-1:Y=Y-1:GOTO200
180 IF JOY=128 THEN TEXT27,170,"WELL DONE PICASSO",1,3,16
190 PAUSE2:GOTO340
200 IFX<20THENX=20:GOTO 100
300 IFX>300THENX=300:GOTO100
310 IFY<60THENY=60:GOTO100
320 IFY>140THENY=140:GOTO100
330 PLOTX,Y,1:GOTO100
340 LOW COL 2,1,1
350 PAINTX+1,Y+1,1:PAUSE5
360 HIRES0,1
370 TEXT50,50,"PRESS SPACE BAR FOR ANOTHER GO",1,2,8
380 GET A$:IFA$=""THEN380
390 IFA$=" "THEN 30
400 GOTO380
```


CAPITOLO 13
ALCUNI ESEMPI DI PROGRAMMI IN SIMONS' BASIC

13.1 INTRODUZIONE

Il capitolo 13 contiene quattro esempi di programma per meglio illustrare che cosa si può fare con la cartuccia del SIMONS' BASIC. Introducete semplicemente i programmi e battete RUN. Buon divertimento.

13.2 PROGRAMMA 1 - DISEGNAMO UN POLIEDRO

Il seguente programma traccia una figura con molti lati e con dimensioni via via decrescenti.

```
10 PRINT"□"
20 CENTRE "SIMONS BASIC POLYHEDRA":PRINT:PRINT
30 CENTRE "BY S BEATS":PRINT:PRINT
40 PRINT "NUMBER OF SIDES  ";:FETCH "■",2,N
50 EXEC SPIRAL
60 TEXT 10,10,"PRESS A KEY",1,1,8
70 GETA$:IFA$=""THEN70
80 CSET0:GOTO10
996 :
997 :
998 :
999 :
1000 PROC SPIRAL
1010 MP=100:HIRES 0,1
1020 FORJ = 0 TO 2.5*π STEPπ/20
1030 FORK = 0+J TO 2*π+J+.1STEP2*π/N
1040 X=INT(MP*1.3*SIN(K)+160)
1050 Y=INT(MP*COS(K)+100)
1060 IFK>0+J THEN LINE X1,Y1,X,Y,1
1070 X1=X:Y1=Y:NEXT
1080 MP=MP-2:NEXT
1090 END PROC
```

13.3 PROGRAMMA 2 - RICERCA DI UNA PAROLA

Il programma sotto riportato vi permette di introdurre un massimo di 20 parole a vostra scelta. Successivamente le mescola in una matrice. Voi dovete trovare le coordinate del punto della griglia dove inizia ogni parola. Potete anche scegliere di stampare la griglia, in modo da effettuare il gioco senza computer.

```
10 REM ***************************
20 REM *                         *
30 REM *  WORDSEARCH             *
40 REM *                         *
50 REM *    BY STEVE BEATS       *
60 REM *                         *
70 REM ***************************
80 :
90 :
100 EXEC SETUP
110 EXEC GETWORDS
120 EXEC SCREEN
130 EXEC SORTLENGTHS
140 EXEC PLACEWORDS
150 EXEC PRINTGRID
160 EXEC GAME
170 CALL FINISH
180 :
181 :
3000 PROC GAME
3002 PRINT AT(28,1) "PRINT(Y/N)"
3004 GET@$:IF@$<>"Y"AND@$<>"N"THEN3004
3005 TI$="000000":TU=0
3006 PRINT AT(28,1) "              "
3008 IF@$="N"THEN 3010
3009 HRDCPY
3010 WF=0:REPEAT
3011 PRINT AT(28,1) "ROW ";:FETCH "W",3,RO$
3012 IF TI$>"001000"THEN TU=1:END PROC
3020 PRINT AT(28,1) "COLUMN ";:FETCH "W",3,CO$
3030 PRINT AT(28,1) "              "
3040 RO=VAL(RO$):CO=VAL(CO$)
3050 IF RO>0ANDRO<21ANDCO>0ANDCO<21THEN3070
3060 PRINT AT(28,1) "ERROR":PAUSE 5000:PRINT AT(28,1) "      ":GOTO3011
3070 F=0:FORI=1TONW:IFRO=PY(I)ANDCO=PX(I)THENF=1:XI=I
3080 NEXT:IFF=1THEN3100
3090 PRINT AT(28,1) "WRONG":PAUSE5000:PRINT AT(28,1) "      ":GOTO3011
3100 FORI=0TOLEN(W$(XI))-1
3110 PRINT AT(3+PX(XI)+I*DX(TW(XI)),2+PY(XI)+I*DY(TW(XI))) "■";
3115 PRINTMID$(W$(XI),I+1,1)
3120 NEXT:PRINT AT(25,2+XI) W$(XI):WF=WF+1
3130 UNTIL WF=NW
```



```
3140 END PROC
3150 :
3160 :
4000 PROC PRINTGRID
4010 PRINT"■":FORY=1TO20:FORX=1TO20
4020 IFA$(X,Y)=""THENA$(X,Y)=MID$(W$(NW*RND(1)+1),5*RND(1)+1,1)
4040 PRINT AT(3+X,2+Y) A$(X,Y)
4050 NEXT:NEXT
4060 END PROC
4070 :
4080 :
5000 PROC PLACEWORDS
5010 PW=0:REPEAT:PW=PW+1
5020 PX(PW)=INT(20*RND(1)+1)
5030 PY(PW)=INT(20*RND(1)+1)
5040 DR=INT(8*RND(1)+1):TW(PW)=DR
5041 CX=PX(PW)+LEN(W$(PW))*DX(DR):CY=PY(PW)+LEN(W$(PW))*DY(DR)
5050 IFCX<1ORCX>20ORCY<1ORCY>20THEN5020
5051 REM IT FITS THE GRID SO CHECK LETTERS
5060 F=0:FORCK=0TOLEN(W$(PW))-1
5070 Z1$=MID$(W$(PW),CK+1,1):Z2$=A$(PX(PW)+CK*DX(DR),PY(PW)+CK*DY(DR))
5080 IFZ2$<>""ANDZ1$<>Z2$THENF=1
5090 NEXT:IFF=1THEN5020
5091 REM IT FITS SO SLOT IT IN
5100 FORCK=0TOLEN(W$(PW))-1
5110 Z1$=MID$(W$(PW),CK+1,1):A$(PX(PW)+CK*DX(DR),PY(PW)+CK*DY(DR))=Z1$
5120 PRINT AT(25,2+PW) W$(PW):NEXT
5130 UNTIL PW=NW
5140 END PROC
5150 :
5160 :
6000 PROC SORTLENGTHS
6010 PRINT AT(25,1) "■SORTING"
6020 F=0:FORI=1TONW-1
6030 IFLEN(W$(I))<LEN(W$(I+1))THENT$=W$(I+1):W$(I+1)=W$(I):W$(I)=T$:F=1
6040 NEXT:IFF=1THEN6020
6050 PRINT AT(25,1) "■          "
6060 END PROC
6070 :
6080 :
7000 PROC SCREEN
7010 PRINT"■■■■■■■■";:COLOUR 5,0
7020 PRINT"/";:FORI=1TO20:PRINT"-";:NEXT
7030 PRINT"\■■";:FORI=1TO20:PRINT"|■■";:NEXT
7040 PRINT"/■■";:FORI=1TO20:PRINT"-■■";:NEXT
7050 PRINT"\■■";:FORI=1TO20:PRINT"|■■";:NEXT
7060 FORI=1TO20:NU$=STR$(I):NU$=MID$(NU$,2):IFLEN(NU$)=1THENNU$=" "+NU$
7070 PRINT AT(1,I+2) NU$:PRINT AT(I+3,0) LEFT$(NU$,1)
7080 PRINT AT(I+3,1) RIGHT$(NU$,1):NEXT
7090 END PROC
7100 :
```

```
7110 :
8000 PROC GETWORDS
8010 COLOUR10,0:PRINT"[CLR][DOWN]HOW MANY WORDS (MAX 20)?"
8020 FETCH "%",3,NW$
8030 PRINT:NW=VAL(NW$):IFNW<1ORNW>20THENS010
8040 PRINT"[DOWN]        NOW TYPE IN THE WORDS[DOWN]"
8050 PRINT" THEY MUST BE BETWEEN 5 AND 15 LETTERS[DOWN]"
8060 FORI=1TONW
8070 PRINTI"[RVS]) ";
8080 FETCH "$",15,W$(I)
8090 IFLEN(W$(I))<5THENPRINTCHR$(13)"[UP]":GOTO8070
8100 PRINT:NEXT
8110 END PROC
8998 :
8999 :
9000 PROC SETUP
9010 COLOUR 5,0:PRINT"[CLR]SETTING UP, PLEASE WAIT....."
9020 DIMA$(20,20),W$(20),PX(23),PY(20),DX(8),DY(8),TW(20)
9030 FORI=1TO20:FORJ=1TO20:A$(I,J)="":NEXT:NEXT
9040 RESTORE:FORI=1TO8:READDX(I),DY(I):NEXT
9050 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0,-1,-1
9060 FORI=1TO20:W$(I)="":PX(I)=0:PY(I)=0:NEXT
9070 END PROC
9998 :
9999 :
10000 PROC FINISH
10010 IFTW=1THEN10100
10020 PRINT AT(28,1) "AGAIN(Y/N)"
10030 GET A$:IFA$<>"Y"ANDA$<>"N"THEN10030
10040 IF A$="Y"THENCLR:RUN
10050 PRINT"[CLR]":END
10100 FORX1=1TONW
10110 FORI=0TOLEN(W$(X1))-1
10120 PRINT AT(3+PX(X1)+I*DX(TW(X1)),2+PY(X1)+I*DY(TW(X1))) "[RVS]";
10130 PRINTMID$(W$(X1),I+1,1)
10140 NEXT:PRINT AT(25,2+X1) W$(X1):NEXT
10150 GOTO10020
```


13.4 PROGRAMMA 3 - QUADRATO MAGICO ALFABETICO

Questo programma mescola casualmente le lettere dell'alfabeto da A ad O, e le dispone in un quadrato di lato 4. Nel quadrato vi è uno spazio vuoto. Voi dovete riordinare alfabeticamente le lettere facendo scorrere le stesse nel quadrato, usufruendo dello spazio lasciato vuoto, come nel classico quadrato magico con i numeri.

```
1 REM ***** LETTER SLIDER GAME *****
2 REM *****                   *****
3 REM *****   BY STEVE BEATS  *****
4 :
5 EXEC INSTRUCTIONS
10 DIMA$(4,4),B$(4,4):MN=1
20 S$="[RVS] [RVS]  [RVS] ↑ [RVS]  "
30 C$="[RVS] [RVS]  [RVS]"
40 L$="ABCDEFGHIJKLMNO "
50 PT=1:FORY=1TO4:FORX=1TO4
60 I1$=MID$(C$,PT,1):I2$=MID$(L$,PT,1):PT=PT+1
70 W$= INST(I1$,S$,1)
80 W$= INST(I2$,W$,10)
90 A$(X,Y)=W$
100 NEXT:NEXT
110 A$(4,4)="[RVS]   [RVS]  [RVS]   "
120 PT=1:FORY=1TO4:FORX=1TO4
130 B$(X,Y)=MID$(L$,PT,1):PT=PT+1
140 NEXT:NEXT:PX=4:PY=4
150 PRINT"[CLR]":COLOUR5,0
160 PRINT AT(12,5) "[RVS]┌──────────┐";
170 FORI=1TO12:PRINT"[RVS]│";:NEXT:PRINT"[RVS]┘";
180 FORI=1TO12:PRINT"─[RVS]";:NEXT:PRINT" [UP]";
190 FORI=1TO12:PRINT"│[RVS]";:NEXT
200 EXEC LETTERS
205 EXEC SHUFFLE
206 REPEAT
210 GETR$:IFR$<"A"ORR$>"O"THEN210
215 PRINT AT(12,3) "                "
220 EXEC CHECKIT
230 PRINT AT(12,3) MS$
240 EXEC FINISHED
250 UNTIL WI=1
260 RUN 10
998 :
999 :
1000 PROC LETTERS
1010 FORY=1TO4:FORX=1TO4
1020 PRINT AT(3*X+10,3*Y+3) A$(X,Y)
1030 NEXT:NEXT
1040 END PROC
1050 :
1060 :
2000 PROC SHUFFLE
2010 RESTORE:FORI=1TO4:READXD(I),YD(I):NEXT
2015 PRINT AT(12,3) "SHUFFLING...."
2020 DATA -1,0,0,1,1,0,0,-1
2025 I=0:REPEAT:I=I+1
2030 D=INT(8*RND(RND(1))+1)
```

```
2040 IFPX+XD(D)<1ORPX+XD(D)>4ORPY+YD(D)<1ORPY+YD(D)>4THEN2030
2050 X1=PX+XD(D):Y1=PY+YD(D):IFX2=X1ANDY2=Y1THEN2030
2060 T$=A$(X1,Y1):A$(X1,Y1)=A$(PX,PY):A$(PX,PY)=T$
2070 T$=B$(X1,Y1):B$(X1,Y1)=B$(PX,PY):B$(PX,PY)=T$
2080 X2=PX:Y2=PY:PX=X1:PY=Y1
2090 EXEC LETTERS
2100 UNTIL I>100
2110 PRINT AT(12,3) "
2120 END PROC
2130 :
2140 :
3000 PROC CHECKIT
3005 PRINT AT(1,1) "■MOVE NUMBER"MN
3010 FL=0:FORI=1TO4
3020 IFPX+XD(I)<1ORPX+XD(I)>4ORPY+YD(I)<1ORPY+YD(I)>4THEN3040
3030 IFB$(PX+XD(I),PY+YD(I))=R$THENFL=1:TE=I
3040 NEXT:IFFL=0THENMS$="ILLEGAL MOVE !!!":GOTO3100
3050 X1=PX+XD(TE):Y1=PY+YD(TE)
3060 T$=A$(X1,Y1):A$(X1,Y1)=A$(PX,PY):A$(PX,PY)=T$
3070 T$=B$(X1,Y1):B$(X1,Y1)=B$(PX,PY):B$(PX,PY)=T$
3080 EXEC LETTERS
3090 MS$="MOVE OK !":PX=X1:PY=Y1:MN=MN+1
3100 END PROC
3110 :
3120 :
4000 PROC FINISHED
4010 FI$="":FORY=1TO4:FORX=1TO4
4020 FI$=FI$+B$(X,Y):NEXT:NEXT
4030 IFFI$=L$THEN4050
4040 END PROC
4050 PRINT AT(12,3) "A WINNER !!!":PAUSE10:WI=1:GOTO4040
4060 :
4070 :
5000 PROC INSTRUCTIONS
5005 COLOUR10,0
5010 PRINT"■DO YOU REQUIRE INSTRUCTIONS (Y/N)?"
5020 GETA$:IFA$<>"Y"ANDA$<>"N"THEN5020
5030 IF A$="N"THEN END PROC
5040 PRINT"■■■    THE OBJECT OF THE GAME IS TO GET
5050 PRINT"■    ALL OF THE LETTERS IN THE CORRECT"
5060 PRINT"■    ORDER....."
5070 PRINT"■■                  A B C D "
5080 PRINT"■                   E F G H "
5090 PRINT"■                   I J K L "
5100 PRINT"■                   M N O ■ "
5110 PRINT"■■■    TO MOVE A LETTER INTO THE VACANT"
5120 PRINT"■    SPACE, JUST TYPE THAT LETTER ON"
5130 PRINT"■    THE KEYBOARD."
5140 PRINT"■■                  TRY IT NOW"
5150 GET A$:IFA$=""THEN5150
5160 END PROC
```


13.5 PROGRAMMA 4 - L'AUTO D'EPOCA

Questo programma disegna un'auto d'epoca su schermo grafico multi-colour.

```
1 COLOUR10,5
10 HIRES 0,1:MULTI 0,2,1
20 CIRCLE 30,150,10,11,1
30 CIRCLE 30,150,13,15,1
31 PAINT 28,150,2
32 PAINT 19,150,1
35 ARC 30,150,270,90,10,17,21,3
36 LINE 15,150,17,150,3
37 LINE 43,150,45,150,3
38 PAINT 16,148,3
40 BLOCK 3,105,13,150,1
41 LINE 3,105,6,105,0
42 LINE 3,105,3,107,0:LINE 10,105,13,105,0:LINE13,105,13,107,0
43 LINE 3,150,6,150,0
44 LINE 3,148,3,150,0
45 LINE 3,115,0,125,1
46 LINE 3,140,0,125,1
50 LINE 16,139,16,105,1
60 LINE 16,105,30,105,1
70 LINE 30,105,30,129,1
71 PAINT 16,107,1
80 LINE 30,105,70,105,2
90 LINE 70,105,70,145,2
91 LINE 70,145,47,145,2
92 PAINT 33,107,2
93 LINE 43,152,56,152,1
94 BLOCK 56,152,74,154,1
95 LINE 74,152,80,152,1
96 LINE 80,152,80,144,1
98 BLOCK 72,105,94,145,2:PLOT 71,105,1
99 PAINT 75,151,1
100 BLOCK 82,146,108,153,1
101 BLOCK 110,144,120,152,1
102 BLOCK 95,105,108,146,1
103 BLOCK 108,105,114,141,1
105 BLOCK 31,70,36,104,1
106 LINE 31,70,95,70,1
```

```
107 BLOCK 70,70,72,104,2
108 BLOCK 31,67,95,69,2
109 LINE 69,71,69,104,1
110 LINE 72,71,72,104,1
112 LINE 95,70,95,104,1
114 BLOCK 35,102,69,104,1
115 BLOCK 73,102,95,104,1
116 LINE 88,102,91,96,1
120 LINE31,66,76,66,1
121 LINE31,65,46,65,1
122 LINE31,64,36,64,1
132 LINE97,70,99,73,1
133 LINE97,67,99,73,1
134 LINE97,68,99,73,1
135 LINE97,69,99,73,1
220 CIRCLE 140,150,10,11,1
230 CIRCLE 140,150,13,15,1
231 PAINT 138,150,2
232 PAINT 129,150,1
235 ARC 140,150,270,90,10,17,21,3
236 LINE 125,150,127,150,3
237 LINE 153,150,155,150,3
238 PAINT 126,148,3
300 LINE 128,152,114,152,1
310 LINE 114,144,124,144,1
320 PAINT 127,151,1
330 LINE 125,141,116,141,1
340 LINE 116,141,116,115,1
341 LINE 116,115,136,115,1
342 LINE  136,115,136,130,1
345 PAINT 133,127,1
350 BLOCK 116,105,136,112,1
360 LINE 137,105,141,105,2
370 LINE 141,105,141,129,2
380 LINE 137,105,137,115,2
390 PAINT 139,107,2
391 PLOT 141,105,1:FORT=128TO120STEP-2
392 LINE T,118,T,127,3:NEXT
400 LOW COL 0,2,6+8 :FORT=0TO159:IFTEST(T,98)=0THENPLOT T,98,3
410 NEXT
420 FORT=0TO62STEP8:LINE 0,T,160,T,3:NEXT:BLOCK 0,0,160,62,3
430 PAINT 2,64,3
500 FORT=0TO159STEP5:IFTEST(T,96)=0THENPAINTT,96,3
510 NEXT
600 LINE 78,107,76,107,1:PLOT 76,108,1
610 LOW COL 7,2,6+8
620 ARC 0,0,90,180,10,20,20,1
630 PAINT 0,0,1
700 FORT=90TO180STEP10:ANGL 0,0,T,33,33,1:NEXT
999 PAUSE 4:NRM:MEM:COLOUR 0,0:PRINT"□[illegible]SEE..!!"
1000 POKE648,4:PRINT"□":A$="D2":LOAD"LOADER",8
```


APPENDICE A
MESSAGGI DI ERRORE

Quando usate il SIMONS' BASIC, puo apparire sullo schermo un messaggio di errore. Questi messaggi sono esclusivi della cartuccia del SIMONS' BASIC e sono elencati in questa appendice, con il loro significato e la probabile causa di errore.

? BAD MODE

si verifica quando avete introdotto un qualunque parametro di un qualunque comando al di fuori della gamma di valori accettati.

? NOT HEX CHARACTER

avete tentato di convertire un numero non in base esadecimale nel suo equivalente in base decimale, usando il comando $.

? NOT BINARY CHARACTER

avete tentato di convertire un numero non in base binaria nel suo equivalente in base decimale, usando il comando %.

? UNTIL WITHOUT REPEAT

avete usato il comando UNTIL senza alcuna precedente dichiarazione REPEAT.

? END LOOP WITHOUT LOOP

avete usato il comando END LOOP senza alcuna precedente dichiarazione di LOOP.

? END PROC WITHOUT EXEC

avete usato il comando END PROC senza aver definito in precedenza nessuna procedura.

? PROC NOT FOUND

avete tentato di usare una procedura non esistente.

? NOT ENOUGH LINES

non avete riservato sufficienti linee per la griglia di un MOB.

? BAD CHAR FOR A MOB IN LINE n

un parametro nella progettazione di un MOB cade fuori della gamma definita. Il numero della linea in cui viene segnalato l'errore è sempre quello in cui viene eseguito il comando DESIGN, anche se ciò non significa necessariamente che l'errore si trova in quella linea.

? STACK TOO LARGE

avete cercato di nidificare più di nove procedure in un loop di programma.

GLOSSARIO

È qui riportato un elenco di termini tecnici utilizzati in questo manuale, con accanto il loro significato.

BIT
acronimo di Binary Digit. Rappresenta la più piccola quantità di informazione e di memoria del computer.

COORDINATA (COORDINATE)
la distanza di un punto di una griglia dagli assi x ed y.

CRASH DI PROGRAMMA
un arresto non desiderato nell'esecuzione del programma.

DATO (DATA)
un'informazione mantenuta nella memoria del computer o in un dispositivo di memoria di massa.

DEBUGGING
ricerca e correzione degli errori di programma.

EDITING
scrittura del programma

ENTRY
introduzione del programma.

FLOATING POINT
rappresentazione di un numero in forma esponenziale.

INDIRIZZO DI PARTENZA (START ADDRESS)
è la locazione della memoria del computer a partire dalla quale viene memorizzato un dato.

INDIRIZZO SECONDARIO (SECONDARY ADDRESS)
instruzione di memorizzazione di un programma o di un file.

INTERO (INTEGER)
un numero intero.

KERNAL
il sistema operativo del COMMODORE 64.

LIBRERIA (LIBRARY)
un insieme di programmi e/o di sub-routine.

LISTING (LISTATO)
indica l'elenco delle istruzioni che compongono un programma.

MOB
acronimo di Moveable Object Block. Indica un insieme grafico definito dall'utente e con movimento programmabile. Vedi anche SPRITE.

OTTAVA (OCTAVE)
indica un insieme di otto note musicali in cui le singole frequenze stanno in determinato rapporto.

ORIGINE (ORIGIN)
è il punto dello schermo dove inizia un tracciato.

PERIFERICA (PERIPHERAL)
è un dispositivo esterno collegato al computer.

PIXEL
è la più piccola porzione indirizzabile dello schermo.

REGISTRO (REGISTER)
area riservata nella memoria del computer.

SALTO (BRANCH)
trasferisce l'esecuzione del programma da una linea ad un'altra.

SCROLLING
spostamento dello schermo o di una sua porzione in direzione verticale od orizzontale.

SPRITE
oggetto grafico programmabile. Vedi anche MOB.

TEMPO DEL CICLO (TIME CYCLE)
durata di una componente della forma d'onda, misurata in millesimi di secondo.

INDICE ANALITICO