

V. Paola - G. Romano

C=128

FILES

MAPPE DI MEMORIA

SISTEMA OPERATIVO



libreria dario flaccovio editrice

V. Paola - G. Romano

C 128

**FILES
MAPPE DI MEMORIA
SISTEMA OPERATIVO**



libreria dario flaccovio editrice

PREFAZIONE

I FILES PER IL '128' ESCE IN UN MOMENTO IN CUI IL COMPUTER E' DIFFUSO IN MANIERA CAPILLARE; RECENTI STATISTICHE HANNO MESSO IN LUCE UNA REALTA' STRABILIANTE: UNA FAMIGLIA SU TRE POSSIEDE UN PERSONAL COMPUTER O UN HOME COMPUTER.

QUESTO VUOL DIRE CHE IL COMPUTER E' CONSIDERATO AL PARI DI UN QUALSIASI ELETTRODOMESTICO.

LE STESSE STATISTICHE METTONO PERO' IN LUCE IL FATTO CHE SONO MOLTO POCHI COLORO CHE USANO ATTIVAMENTE IL COMPUTER CIOE' PROGRAMMANDOLO PER LE LORO ESIGENZE, PER I LORO PROBLEMI. LA MAGGIORE FETTA DI UTENZA PREFERISCE COMPRARE PROGRAMMI GIA' PRONTI OPPURE USARE IL COMPUTER COME SOFISTICATA CONSOLE PER VIDEOGAMES.

TUTTO QUESTO SMINUISCE QUELLO CHE E' 'IL COMPUTER', LO AVVILISCE: UNA MACCHINA CREATA DALL' UOMO PER RISOLVERE I PROBLEMI DELL' UOMO, VIOLENTATA AD ESEGUIRE SOLO E SOLTANTO GIOCHI.

QUESTO LIBRO NON HA LA PRETESA DI VOLERE RIVOLUZIONARE QUESTO STATO DI FATTO, VUOLE SOLTANTO INSERIRSI IN QUELLA FASCIA DI UTENZA DI CUI PARLAVO PRIMA, QUELLA DEI 'PIGRI' CHE PUR AVENDO LE CAPACITA' PER SCRIVERE I PROPRI PROGRAMMI NON LO FANNO PERCHE' PENSANO ALLE MILLE DIFFICOLTA' CHE NORMALMENTE SI INCONTRANO NELLA STESURA DI UN PROGRAMMA, DIMOSTRANDO LORO CHE PROGRAMMARE E FACILE E INTERESSANTE.

PERCHE' PROPRIO I FILES ? MA PERCHE' SONO I PIU' DIFFICILI DA TRATTARE, A DETTA DI MOLTI, E NOI SIAMI QUI, CON QUESTO LIBRO, PER SMENTIRE QUESTE VOCI, PER DIMOSTRARE CHE PROGRAMMARE E' FACILE, BASTA VOLERLO, E CHE LA PROGRAMMAZIONE E' DA CONSIDERARSI COME UN GIUOCO INTELLIGENTE, UNA SERIE DI REGOLE PER ARRIVARE AD UN RISULTATO CONCRETO.

PERCHE' NON PROVARE ? E' INDESCRIVIBILE LA SODDISFAZIONE CHE SI PROVA QUANDO DOPO GIORNI DI FATICA SI VEDE APPARIRE SULLO SCHERMO IL RISULTATO DEL PROPRIO LAVORO, ED E' UNA GIOIA CONTINUA MIGLIORARE CIO' CHE SI E' FATTO.

I FILES RAPPRESENTANO UNA DELLE APPLICAZIONI PER CUI I

COMPUTER SONO STATI INVENTATI E SERVONO A REALIZZARE IN POCHI MINUTI DELLE APPLICAZIONI CHE AD UN UOMO PORTEREBBERO VIA GIORNI DI TEMPO. MA NON E' FANTASTICO ? AL LAVORO DUNQUE.

UNA DOVEROSA PRECISAZIONE: COME LETTORE DI DISCHI SI E' FATTO RIFERIMENTO ALLA UNITA' 1541, PER INTENDERCI QUELLA DEL 'VECCHIO'COMMODORE 64 DATO CHE AL MOMENTO NON E' ANCORA DISPONIBILE LA NUOVA UNITA' SPECIFICA PER IL '128'. IN APPENDICE AL LIBRO E' POSSIBILE TROVARE LE MAPPE DI MEMORIA DEL 'C 128' IN INGLESE, MA SONO CERTO CHE FARANNO MOLTO COMODO AL PROGRAMMATORE EVOLUTO, PER SFRUTTARE AL MEGLIO LE NOTEVOLI CARATTERISTICHE DI QUESTO COMPUTER.

UN RINGRAZIAMENTO A TUTTI COLORO CHE HANNO CONTRIBUITO ALLA STESURA DI QUESTO VOLUME, ED IN PARTICOLARE VORREI RINGRAZIARE MIA MOGLIE GIUSEPPINA, A CUI DEDICO QUESTO LAVORO, PER LA SUA SAPIENTE OPERA DI PROGRAMMATRICE CHE HA RISOLTO NON POCHI PROBLEMI.

L' AUTORE
VITTORIO PAOLA

C O M A N D I ' 1 2 8 '

COMANDI 128

PRIMA DI ADDENTRARCI NELLA TRADUZIONE DEL PROGETTO DELL' ARCHIVIO IN CODICE BASIC, BISOGNA CONOSCERE QUALI SONO I COMANDI CHE IL '128' USA PER MANIPOLARE I FILES IN GENERE. SARANNO ESAMINATI SOLO I NUOVI COMANDI, IN QUANTO I VECCHI, PER INTENDERCI QUELLI CHE VANNO BENE PER IL '64', SONO RIMASTI GLI STESSI.

DOPEN :

QUESTO COMANDO E' UTILIZZATO PER APRIRE UN FILE SEQUENZIALE O AD ACCESSO CASUALE, PURCHE' CON RECORDS DI LUNGHEZZA COSTANTE.

LA SINTASSI ESATTA E' LA SEGUENTE:

DOPEN #A,"NOME",DX,LR

DOVE:

A ---> NUMERO DEL FILE LOGICO APERTO

X ---> NUMERO DEL DRIVE USATO

R ---> LUNGHEZZA DEL RECORD

PER 'NOME' SI INTENDE, IL NOME DEL FILE CHE PUO' ESSERE AL MASSIMO DI 16 CARATTERI.

IL DOPEN SOSTITUISCE, AD ESEMPIO IL CLASSICO:

OPEN 1,8,0,"NOME"

CLOSE :

QUESTO COMANDO E' UTILIZZATO PER CHIUDERE UN FILE, PRECEDENTAMENTE APERTO CON DOPEN.

LA SINTASSI ESATTA E' :

DOPEN A

DOVE:

A ---> IL NUMERO DEL FILE LOGICO

RECORD :

QUESTO POTENTE COMANDO SERVE A PUNTARE LA LETTURA O LA SCRITTURA IN UN FILE AD ACCESSO CASUALE AD UN RECORD PRECISO, E POI, INDIVIDUATO IL RECORD, SI PUO' PUNTARE AD UN CAMPO PARTICOLARE.

VEDIAMO LA SINTASSI ESATTA:

RECORD #A,R,B

DOVE:

A ---> IL NUMERO DEL FILE LOGICO

R ---> IL RECORD DA SELEZIONARE

B ---> POSIZIONAMENTO ALL' INTERNO DEL RECORD

CON QUESTO COMANDO SARA' POSSIBILE GESTIRE IN MANIERA EFFICIENTE E COMPLETA I FILES AD ACCESSO CASUALE.

PER USARE I FILES OCCORE CONOSCERE ALTRI COMANDI, MA QUESTI SONO EGUALI A QUELLI USATI PER IL COMMODORE '64', ECCOLI:

PRINT #A

INPUT #A

GET #A

DOVE A RAPPRESENTA SEMPRE IL FILE LOGICO.
NON RESTA CHE PASSARE AGLI ESEMPI.

C O S A S O N O

I F I L E S ?

COSA SONO I FILES

UNA DELLE APPLICAZIONI PIU' IMPORTANTI DEL PERSONAL COMPUTER, E' QUELLA DI MEMORIZZARE GRANDI QUANTITA' DI DATI PER POI ESEGUIRE SU DI ESSI ELEBORAZIONI DI VARIO TIPO.

AD ESEMPIO, CON IL COMPUTER E' SEMPLICISSIMO FARE DELLE PROIEZIONI STATISTICHE AVENDO A DISPOSIZIONE UN CAMPIONE DI DATI, OPPURE DISPONENDO DEI DATI ADATTI, REDIGERE DEI TABULATI DOVE SIANO SEGNATI TUTTI GLI STUDENTI UNIVERSITARI CON I RELATIVI ESAMI SOSTENUTI, O ANCORA TENERE, CON SISTEMI MOLTO GROSSI, LA SITUAZIONE ANAGRAFICA DI UNA CITTA'.

CON IL NOSTRO '128' NON POSSIAMO FARE QUESTE GRANDI COSE PERO' NE POSSIAMO FARE TANTE ALTRE ALTRETTANTO INTERESSANTI. DUNQUE I FILES RAPPRESENTANO UNA SERIE DI INFORMAZIONI REGISTRATE IN MANIERA PERMANENTE SU DI UN SUPPORTO MAGNETICO, E CHE ABBIANO UN CERTO ORDINE, O CHE SIANO CLASSIFICATE IN UN CERTO MODO PER POI GARANTIRNE IL RITROVAMENTO.

I DATI SI POSSONO ORGANIZZARE IN DIVERSI MODI OGNUNO DEI QUALI HA I SUOI PRO E CONTRO, NOI ESAMINEREMO DUE DI QUESTI MODI, CHE RAPPRESENTANO, OGGI, LA MANIERA MIGLIORE DI GESTIRE GRANDI QUANTITA' DI DATI:

- I FILES SEQUENZIALI
- I FILES AD ACCESSO DIRETTO

I FILES SEQUENZIALI SONO ORGANIZZATI IN MANIERA CHE PER ACCEDERE AD UN DETERMINATO BLOCCO DI DATI, DEVO NECESSARIAMENTE LEGGERE TUTTO IL CONTENUTO DEL FILE, LEGGERE UN FILE SIGNIFICA, TRASPORTARE TUTTE LE INFORMAZIONI IN ESSO CONTENUTE DALLA MEMORIA DI MASSA, RAPPRESENTATA DAL SUPPORTO MAGNETICO SU CUI LE INFORMAZIONI SONO REGISTRATE, ALLA MEMORIA CENTRALE DEL SISTEMA ADOPERATO; COSI' OPERANDO, SI VERIFICA SPESSO CHE PER LEGGERE POCHI BYTES DI DATI SI E' COSTRETTI A PORTARE IN MEMORIA CENTRALE DIVERSE MIGLIAIA DI BYTES, CIOE' TUTTO IL FILES SEQUENZIALE.

CON I FILES AD ACCESSO DIRETTO, INVECE TUTTO QUESTO NON ESISTE: VI E', INFATTI, LA POSSIBILITA' DI ACCEDERE SOLO AL BLOCCO DI INFORMAZIONI CHE CI INTERESSA, IN PRATICA LA LETTURA, CIOE' TRASFERIMENTO IN MEMORIA CENTRALE, VIENE EFFETTUATA SUL FILE, SOLO NELLA PARTE CHE CI INTERESSA, LASCIANDO TUTTO IL RESTO SUL SUPPORTO MAGNETICO. IN QUESTA MANIERA L' OCCUPAZIONE DI MEMORIA CENTRALE E' RIDOTTA AL MINIMO INDISPENSABILE ED IL TRATTAMENTO DEI DATI E' MOLTO EFFICIENTE.

QUANDO CONVIENE USARE I FILES SEQUENZIALI E QUANDO QUELLI AD ACCESSO DIRETTO ?

LA SCELTA E' QUASI SEMPRE LEGATA AL TIPO DI SUPPORTO MAGNETICO ADOPERATO VEDIAMO DI CAPIRE PERCHE':

I SUPPORTI MAGNETICI PIU' DIFFUSI SONO IL NASTRO MAGNETICO E I DISCHI MAGNETICI. IL COMPUTER GESTISCE IN MANIERA DIVERSA QUESTI DUE TIPI DI SUPPORTI, PRECISAMENTE SU NASTRO NON E' CAPACE DI TROVARE DETERMINATE INFORMAZIONI E DEVE ESSERE CURA DEL PROGRAMMATORE POSIZIONARE LA TESTINA DI LETTURA ALL' INIZIO DEL FILE CONTENENTE IL DATO CERCATO.

SU DISCO, INVECE TUTTA LA GESTIONE DELLA RICERCA DEI FILES, E' PILOTATA DAL CALCOLATORE, AL PROGRAMMATORE BASTERA' SPECIFICARE QUALE FILE RICERCARE E IL COMPUTER AUTOMATICAMENTE TROVERA' LE INFORMAZIONI RICHIESTE.

DA QUESTA SEMPLICE DESCRIZIONE POSSIAMO TRARRE ALCUNE CONCLUSIONI E CIOE' CHE SU NASTRO SI POSSONO GESTIRE SOLTANTO FILES DI TIPO SEQUENZIALE MENTRE SU DISCO SI POSSONO GESTIRE FILES DI OGNI TIPO.

MA VEDIAMO PIU' DA VICINO, CON UN ESEMPIO PRATICO COME E' LOGICAMENTE ORGANIZZATO UN FILE AD ACCESSO DIRETTO:

IMMAGINIAMO UN COMUNE SCHEDARIO DI UNA AZIENDA CHE CONTIENE I DATI PRINCIPALI DI TUTTI I DIPENDENTI, OGNI DIPENDENTE AVRA' UNA SCHEDA E SU DI ESSA SARANNO ANNOTATI IL NOME IL COGNOME LA DATA DI NASCITA, LA DATA DI ASSUNZIONE, LA QUALIFICA ED EVENTUALMENTE DELLE NOTE PERSONALI.

L' INSIEME DELLE SCHEDE, RAPPRESENTA UN FILE DI INFORMAZIONI, OGNI SCHEDA RAPPRESENTA UN RECORD DI INFORMAZIONI, E OGNI CONTENUTO DELLA SCHEDA RAPPRESENTA UN CAMPO DI INFORMAZIONI.

IL FILE E', DUNQUE, UNA SERIE DI INFORMAZIONI ORGANIZZATE ED E' COSTITUITO DA TANTI RECORDS OGNUNO DEI QUALI E' DIVISO IN

CAMPI.
RIASSUMENDO:

FILE -----> DIPENDENTI
RECORD -----> MARIO ROSSI
CAMPO 1-----> NOME
CAMPO 2-----> COGNOME
CAMPO 3-----> DATA DI NASCITA
CAMPO 4-----> DATA DI ASSUNZIONE
CAMPO 5-----> QUALIFICA
CAMPO 6-----> NOTE PERSONALI

CHE VUOL DIRE:

NEL FILE 'DIPENDENTI' ESISTE IL NOMINATIVO 'MARIO ROSSI' CON I SEGUENTI DATI CARATTERISTICI:

NOME..... MARIO
COGNOME..... ROSSI
DATA DI NASCITA..... 31/10/1958
DATA DI ASSUNZIONE... 02/09/1982
QUALIFICA..... OPERAIO
NOTE PERSONALI..... OTTIMO ELEMENTO

L' INSIEME DEI DATI RELATIVI A MARIO ROSSI COSTITUIRA' UN RECORD DEL FILE 'DIPENDENTI', NATURALMENTE ESISTERA' IL RECORD GIULIO BIANCHI, IL RECORD MASSIMO ANTONI E COSI' VIA. UN FILE DI QUESTO TIPO PUO' ESSERE LUNGO QUANTO SI VUOLE, SI E' SOLO LIMITATI DALLA CAPIENZA DEL DISCO.

CON QUESTA LOGICA E' POSSIBILE RAPPRESENTARE QUALSIASI INFORMAZIONE, DALLA RUBRICA TELEFONICA, ALL' ORGANIZZAZIONE DI UNA BIBLIOTECA, ALL' INDICE DI UNA ENCICLOPEDIA E TANTE ALTRE COSE SI POSSONO FARE CON UN CORRETTO USO DEI FILES.

V E D I A M O I L D I S C O
P I U ' D A V I C I N O

VEDIAMO IL DISCO PIU' DA VICINO

ESISTONO DIVERSI TIPI DI DISCHETTI PER CALCOLATORI, NOI FAREMO RIFERIMENTO A QUELLI NORMALMENTE USATI PER IL COMMODORE '128' CHE MISURANO 5 POLLICI E 1/4.

QUESTI DISCHI VENGONO VENDUTI IN UNA CUSTODIA DI CARTA DOVE, NEL RETRO, SONO STAMPATE TUTTE LE PRECAUZIONI DA ADOTTARE PER EVITARE DANNI ALLA SUPERFICIE MAGNETICA CON INEVITABILE PERDITA DEI DATI SU DI ESSA REGISTRATI.

LE PRECAUZIONI DA ADOTTARE SONO:

- TENERE SEMPRE IL DISCO NELLA APPOSITA CUSTODIA PROTETTIVA;

- NON TOCCARE IL DISCO IN CORRISPONDENZA DELLE FESSURE ESPOSTE;

- NON ESPORRE IL DISCO A SBALZI DI TEMPERATURA, PRESTARE PARTICOLARE ATTENZIONE A NON SOTTOPORRE IL DISCO A FONTI DI CALORE;

- INSERIRE CON CURA IL DISCO NELL' APPOSITA FERITOIA DEL LETTORE (DRIVE DISK);

- NON PIEGARE IL DISCO;

- NON SOTTOPORLO A CAMPI MAGNETICI, VI RICORDO CHE SONO GENERATORI DI FORTI CAMPI MAGNETICI:

A) I TELEVISORI

B) I MONITOR

C) I MOTORI IN GENERE (PHON, MOTORI ELETTRICI, ECC. ECC.)

D) I TELEFONI QUANDO SUONANO

E) IL CAMPANELLO DI CASA

PRESTATE PARTICOLARE ATTENZIONE A QUESTA NORMA PERCHE' SI PERDONO VERAMENTE I DATI IN MANIERA IRREVERSIBILE.

ATTENZIONE PURE A NON FUMARE, QUANDO SI MANEGGIANO I DISCHI E A NON ROVESCIARE NIENTE SULLA SUPERFICIE MAGNETICA (BIBITE, MOLLICHE, ECC. ECC.

QUANDO SI COMPRA UN DISCO, QUESTO PUO' ANDARE BENE PER QUALUNQUE CALCOLATORE CHE USI QUEL FORMATO DI DISCHI, PER ADATTARLO NELLA FATTISPECIE, AL NOSTRO CALCOLATORE, BISOGNA SOTTOPORLO AD UNA OPERAZIONE CHE LO RENDERA' COMPATIBILE SOLO CON LA MARCA E IL TIPO DI CALCOLATORE ADOPERATO, QUESTA OPERAZIONE PRENDE IL NOME DI FORMATTAZIONE.

PER FARE UN ESEMPIO UN DISCO FORMATTATO CON IL COMMODORE '128' POTRA' SOLO ESSERE USATO DA UN COMMODORE '128' ED EVENTUALI COMPUTER COMPATIBILI.

IL COMANDO DA IMPARTIRE AL COMPUTER PER ESEGUIRE LA FORMATTAZIONE E' IL SEGUENTE:

OPEN 15,8,15,"N0:NOME,ID"

DOVE:

N ----> E' L' ABBREVIAZIONE DI NEW

0: ----> IDENTIFICA IL DRIVE

NOME ----> E' IL NOME DA DARE AL DISCO

ID ----> DUE CARATTERI DI IDENTITA'

SONO COMANDI VALIDI:

OPEN 15,8,15,"N0:FILES 128,00"

OPEN 15,8,15,"N0:DATI 12/12/78,55"

PER FORMATTARE UN DISCO SI PUO' USARE UN NUOVO COMANDO DISPONIBILE SUL '128', IL COMANDO HEADER, ECCO LA SINTASSI:

HEADER"NOME",DX,IYY

DOVE:

NOME ----> E' IL NOME DA DARE AL DISCO

X ----> E' IL NUMERO DEL DRIVE

YY ----> DUE CARATTERI DI IDENTITA'

SONO COMANDI VALIDI:

HEADER "CIAO",D0,I00

HEADER "DATAFILE",D0,I86

UNA VOLTA FORMATTATO IL DISCO RISULTERA' DIVISO IN TRACCE E SETTORI.

LE TRACCE SONO CONCENTRICHE E NUMERATE DALL' ESTERNO VERSO L' INTERNO E SONO DIVISE IN SETTORI, OGNI TRACCIA CONTIENE UN NUMERO DI SETTORI CHE VARIA IN FUNZIONE DELL' AMPIEZZA DELLA TRACCIA (LE PIU' INTERNE SONO PIU' PICCOLE DELLE ESTERNE).

LE INFORMAZIONI SONO COSI' MEMORIZZATE IN TRACCE E SETTORI E IL COMPUTER LE IDENTIFICA COME IN UN SISTEMA DI ASSI CARTESIANI, CON DELLE COORDINATE, APPUNTO, TRACCIA E SETTORE.

OGNI SETTORE CONTIENE 256 BYTES.

LA TRACCIA 18 E' RISERVATA ALLA DIRECTORY, CHE CONTIENE L' ELENCO DEI PROGRAMMI CONTENUTI SUL DISCO, A ALLA BAM (BLOCK AVAILABLE MAP) CIOE' UNA MAPPA DOVE SONO DESCRITTI TUTTI I SETTORI DEL DISCO E SE QUESTI SONO LIBERI O OCCUPATI.

I SETTORI IN UN DISCO FORMATTATI CON IL DRIVE 1541 (COMPATIBILE CON IL COMMODORE 64) SONO 664.

ORGANIZZIAMO
UN ARCHIVIO

ORGANIZZIAMO UN ARCHIVIO

ORA, PER ESEMPIO PROGETTEREMO UN ARCHIVIO E PER ESSERE PIU' PRECISI, UN ARCHIVIO DI LIBRI.

LA PROGETTAZIONE DI QUESTO ARCHIVIO CONSISTE NEI SEGUENTI PASSI:

- DEFINIZIONI DELLE INFORMAZIONI DA CONSERVARE PER OGNI SINGOLO LIBRO DELLA BIBLIOTECA, QUINDI DEFINIZIONE DEI CAMPI;

- DEFINIZIONE DEI CAMPI CHIAVE, CIOE' QUEI CAMPI CHE SERVIRANNO PER CERCARE LE INFORMAZIONI ALL' INTERNO DEL FILE,

- VALUTAZIONE DELLA LUNGHEZZA DI UN RECORD, COME SOMMA DELLE LUNGHEZZE DI TUTTI I CAMPI

- PREVISIONE, ANCHE APPROSSIMATIVA DEL NUMERO DI RECORD CHE COSTITUIRA' IL NOSTRO FILES;

- VALUTAZIONE DELL' IMPEGNO DI MEMORIA OCCUPATA DAL FILE SU DISCO E IN MEMORIA CENTRALE.

PRIMA FASE:

LE INFORMAZIONI DA MEMORIZZARE PER OGNI LIBRO COSTITUENTE LA BIBLIOTECA SONO, AD ESEMPIO, LE SEGUENTI:

A) TITOLO DEL LIBRO	----->	20 CARATTERI
B) AUTORE DEL LIBRO	----->	20 CARATTERI
C) ARGOMENTO	----->	20 CARATTERI
D) SCAFFALE	----->	5 CARATTERI
E) NOTE	----->	30 CARATTERI

TOTALE	95

QUESTO PRIMO CALCOLO SERVE A STABILIRE QUANTO SARA' LUNGO IL RECORD CHE RIPETENDOSI PER N VOLTE COSTITUIRA' IL FILE. CIO' E' MOLTO IMPORTANTE, IN QUANTO IN FASE DI IMMISSIONE DEI DATI BISOGNERA' ATTENERSI SCRUPolosAMENTE A TALI DIMENSIONI, PERCHE' IN CASO CONTRARIO, SI RISCHIA DI INVADERE IL RECORD SUCCESSIVO PROVOCANDO SOVRAPPOSIZIONI DI INFORMAZIONI.

SECONDA FASE

ORA SI DEVE STABILIRE IL CRITERIO CON CUI SI DOVRA' RICERCARE UN LIBRO ALL' INTERNO DEL FILE. BISOGNA DECIDERE SE UN LIBRO LO SI VUOLE CERCARE PER TITOLO O PER AUTORE O IN TUTTI E DUE I MODI. I CAMPI CHE SEVIRANNO PER LA RICERCA DELL' INTERO RECORD PRENDONO IL NOME DI CAMPI CHIAVE. NEL NOSTRO FILE NOI CERCHEREMO I LIBRI PER TITOLO E PER AUTORE, SEMBRA CHE SIA UNO DEI MODI MIGLIORI.

TERZA FASE

LA VALUTAZIONE DELLA LUNGHEZZA DEL RECORD E' STATA GIA' FATTA: OGNI RECORD SARA' LUNGO 95 CARATTERI.

QUARTA FASE

UNA BIBLIOTECA CASALINGA, DIFFICILMENTE SUPERA I 100 LIBRI, COMUNQUE NOI PREVEDEREMO CHE LA NOSTRA BIBLIOTECA SIA COSTITUITA DA 500 LIBRI, UN BEL NUMERO (POTREBBERO ESSERE MOLTI DI PIU').

QUINTA FASE

SE UN RECORD E' LUNGO 95 CARATTERI E IL FILE SARA' COMPOSTO DA 500 RECORD, TUTTO IL FILE SARA' COMPOSTO DA:

$CARATTERI = 95 \times 500 = 47500$

PER VALUTARE L' IMPEGNO DI MEMORIA IN KBYTES, BASTA SAPERE

CHE 1 KBYTES E' COMPOSTO DA 1024 BYTES (O CARATTERI) PER CUI AVREMO CHE:

$KBYTES=47500/1024=46.38$

IL FILE 'BIBLIOTECA, APPROSSIMANDO PER ECCESSO, SU DISCO OCCUPERA' 47 KBYTES.

QUESTA E' LA PARTE ANALITICA DELL' ARCHIVIO, A QUESTO PUNTO SI METTE MANO ALLA TASTIERA E SI TRADUCE IN CODICE QUANTO DETTO FINO AD ORA.

ALLA FINE CI RITROVEREMO FRA LE MANI UN PROGRAMMA IN GRADO DI GESTIRE LA NOSTRA BIBLIOTECA IN MANIERA EFFICIENTE.

QUESTO PROGRAMMA PUO' IN SEGUITO ESSERE MIGLIORATO, AD ESEMPIO, SI POTRA' PREVEDERE LA POSSIBILITA' DI OTTENERE DELLE STAMPE (REPORT) CON TUTTI I DATI INTRODOTTI, ORDINATI E RIASSUNTI IN MODO DA AVERE UN QUADRO COMPLETO DELLO STATO DI FATTO DELLA BIBLIOTECA.

AFFINCHE' UN ARCHIVIO SIA EFFICIENTE, NON BASTA INTRODURRE DATI E RICERCARLI, BISOGNERA' PREPARARE ROUTINE DI AGGIORNAMENTO.

L' AGGIORNAMENTO DI UN ARCHIVIO CONSISTE IN QUESTE FASI:

- VARIAZIONE DEI DATI GIA' INTRODOTTI;
- CANCELLAZIONE DI RECORD;
- AGGIUNTA DI RECORD;

FATTO QUESTO POSSIAMO RITENERE IL PROGRAMMA COMPLETO, FRA POCO VEDREMO CON ESEMPI PRATICI COME CREARE TUTTO QUESTO.

D A T A B A S E

DATA BASE

UN PROGRAMMA CHE CONSENTA DI ORGANIZZARE GROSSI FILES DI DATI IN UNA QUALCHE MANIERA, TALE DA RENDERE POSSIBILE LA CONSULTAZIONE, L' AGGIORNAMENTO, ED ALTRE OPERAZIONI CARATTERISTICHE, PRENDE IL NOME DI BASE DI DATI O DATA BASE.

IN UN DATA BASE CHE DEVE GESTIRE MOLTE INFORMAZIONI, ANCHE SUDDIVISE IN PIU' FILES VENGONO RICHIESTI ALCUNI REQUISITI FONDAMENTALI, VEDIAMOLI:

UN DATA BASE DEVE ESSERE

- FACILE DA USARE
- VELOCE NELLA RICERCA
- RELAZIONALE
- PROGRAMMABILE
- POCO O PER NIENTE RIDONDANTE
- AFFIDABILE
- SICURO

NATURALMENTE TUTTE QUESTE QUALITA' RACCOLTE IN UN UNICO DATA BASE, NE FANNO UN MODELLO IDEALE, IN PRATICA CI SI CONTENTA DI ALCUNE DI QUESTE QUALITA' IN RELAZIONE ALL' USO CHE SI DEVE FARE DEL PROGRAMMA, E ALLA QUALITA' DEL PRODOTTO ADOPERATO.

FACILE DA USARE, PERCHE' NON TUTTI SIAMO ESPERTI DI COMPUTER, IL PROGRAMMA PUO' ANDARE IN MANO ALLA SEGRETERIA O AL FATTORINO DI UNA DITTA, O AD UN MANAGER, CHE NON ABBAIA DIMESTICHEZZA CON QUESTO GENERE DI PROGRAMMI, ED IN QUESTO CASO NON SERVE A NULLA AVERE UN DATA BASE EFFICIENTISSIMO MA CHE PER ESSERE USATO RICHIEDE UN CORSO DI TRE MESI.

VELOCE NELLA RICERCA DEI DATI, PERCHE' NON E' AMMISSIBILE CHE PER LA RICERCA DI UN DATO IL COMPUTER PERDA DEL TEMPO PREZIOSO INUTILMENTE, ESISTE UN PARAMETRO DI CONFRONTO FRA IL TEMPO DI RICERCA IDEALE E QUELLO REALMENTE OCCORRENTE PER IL REPERIMENTO DI UN DATO , QUESTO PARAMETRO ESPRIME CON UN NUMERO, GENERALMENTE SECONDI QUANTO E' LO SCARTO FRA IL

TEMPO TEORICO, CALCOLATO IN BASE AD ALTRI PARAMETRI E IL TEMPO CHE IL PROGRAMMA IMPIEGA REALMENTE A REPERIRE UNA INFORMAZIONE.

TANTO PIU' QUESTA DIFFERENZA DI TEMPO SI AVVICINA ALLO ZERO (PARAMETRO IDEALE) TANTO MIGLIORE E' IL DATA BASE.

RELAZIONALE VUOL DIRE CHE SE PER ESEMPIO HO MEMORIZZATO TUTTI I DOTTORI IN MEDICINA CHE LAVORANO NEGLI OSPEDALI ITALIANI CON TUTTE LE LORO NOTE CARATTERISTICHE, IL PROGRAMMA DEVE ESSERE IN GRADO DI RELAZIONARE ALL' OPERATORE CHE INTERROGA IL DATA BASE, SU QUALUNQUE INFORMAZIONE CHE RIGUARDI I DATI COMPRENSIVI LA BASE DI DATI.

PER ESEMPIO IL DATA BASE DEVE SAPERE FORNIRE, I NOMINATIVI DEI MEDICI CHE LAVORANO IN OSPEDALI SITI IN COMUNI CON PIU' DI 500000 ABITANTI CHE SIANO SPECIALIZZATI IN CHIRURGIA E CHE ABBIANO FATTO ALMENO 3 ANNI DI INTERNATO.

LA RISPOSTA A QUESTA COMPLESSA DOMANDA IMPLICA UNA RICERCA FRA I DATI CON DELLE CONDIZIONI BEN PRECISE, SCARTANDO TUTTI I RECORD DEI MEDICI CHE NON CORRISPONDONO ALLE RICHIESTE DELL' OPERATORE E ACCETTANDO, INVECE, TUTTI QUEI RECORD CHE RISPONDONO AI REQUISITI RICHIESTI.

QUESTO TIPO DI RICERCA, PREVEDE DUE FASI, UNA PRIMA DENOMINATA 'PARSER' CHE SERVE A DECIFRARE L' INTERROGAZIONE E A PORRE LE CONDIZIONI DI LETTURA, ED UNA SECONDA, CHE EFFETTUI UNA SELEZIONE DEI RECORD IN BASE AI PARAMETRI ELABORATI DAL PARSER; COME SI PUO' DEDURRE UNA ORGANIZZAZIONE DEI DATI MOLTO COMPLESSA, DI TUTTO CIO' FAREMO QUALCHE ESEMPIO NEI PROSSIMI CAPITOLI.

UN DATABASE PROGRAMMABILE CONSENTE ALL' OPERATORE DI STABILIRE DETERMINATE PROCEDURE, PERSONALIZZANDO L'USO DEL PROGRAMMA STESSO, CIO' NATURALMENTE RICHIEDE UNO STUDIO DELLE FUNZIONI DI PROGRAMMAZIONE (IN GENERE UN VERO E PROPRIO LINGUAGGIO ALLE VOLTE ANCHE COMPLESSO) CHE IL DATA BASE METTE A DISPOSIZIONE DELL' UTENTE.

QUESTA E' UNA FUNZIONE DEL DATABASE RIVOLTA SOLO AGLI ADDETTI AI LAVORI E CONSENTE LA CREAZIONE DI PROCEDURE MOLTO INTERESSANTI.

LA RIDONDANZA E' UN PARAMETRO MOLTO IMPORTANTE DA CONSIDERARE IN UN DATABASE, E CONSISTE NEL FATTO CHE ALCUNI PROGRAMMI A CAUSA DELLA LORO COMPLESSA ORGANIZZAZIONE REGISTRINO IN PIU' FILES LA STESSA INFORMAZIONE, CIO'

NATURALMENTE PRODUCE UNO SPRECO DI MEMORIA, CHE, SPECIALMENTE NEI PICCOLI SISTEMI, SI TRADUCE IN UN RALLENTAMENTO DI TUTTE LE ATTIVITA' DEL CALCOLATORE.

PURTROPPO LA RIDONDANZA E' QUASI INEVITABILE NELLA GESTIONE DI FILES COMPLESSI, PERO' CON OPPORTUNI MEZZI, LA SI PUO' RIDURRE DRASTICAMENTE.

ANCHE QUESTO ARGOMENTO DI STUDIO SARA' OGGETTO DEI PROSSIMI CAPITOLI.

AFFIDABILE NEL SENSO, CHE LE INFORMAZIONI MEMORIZZATE NON DEVONO ANDARE PERDUTE PER QUALCHE RAGIONE, O CHE NEL CASO DI INCIDENTI DI VARIO TIPO (CADUTA DELLA TENSIONE, ERRATO USO DEL PROGRAMMA), LE PROCEDURE PER RIPRISTINARE IL FILE SIANO LE PIU' SEMPLICI POSSIBILI.

INFINE, SICURO, CIOE' CHE GARANTISCA UN BUON LIVELLO DI SICUREZZA DEI DATI INTRODOTTI CONTRO:

- A) MANOMISSIONI DATI DA PARTE DI ESTRANEI
- B) PERDITE DI CORRENTE DEL SISTEMA
- C) MANOVRE ERRATE DELL' OPERATORE

TUTTO QUESTO RAPPRESENTA UN DATA BASE IDEALE, PRATICAMENTE INESISTENTE MA IN BASE AD ATTENTA VALUTAZIONE DELLE PROPRIE ESIGENZE SI PUO' SCEGLIERE, UN BUON DATA BASE O , ADDIRITTURA, PROGRAMMARSENE UNO DA SOLI.

SCRIVERE UN BUON PROGRAMMA DI DATABASE NON E' LA COSA PIU' SEMPLICE DI QUESTO MONDO, PER QUESTO MOLTO SPESSO SI RICORRE ALL' ACQUISTO DI PACCHETTI GIA' PRONTI.

QUESTI PACCHETTI CONSENTONO DI DEFINIRE IL PROPRIO FILE COME DELLE SCHEDE, SIMULANDO UN ARCHIVIO CARTACEO TIPO SCHEDARIO, NATURALMENTE CON UNA ORGANIZZAZIONE DEI DATI PIU' EFFICIENTE.

QUESTI PROGRAMMI SONO VOLUTAMENTE SCRITTI, IN MANIERA GENERICA, CIOE' ADATTABILI A MOLTEPLICI SITUAZIONI.

POSSONO SERVIRE:

- AL DENTISTA PER IL PROPRIO ARCHIVIO PAZIENTI
 - AL CONSULENTE PER RICORDARGLI LE SCADENZE DEI SUOI ASSISTITI
 - AL MAGAZZINIERE PER ORGANIZZARE IL PROPRIO LAVORO
 - AL BIBLIOTECAIO PER RASSETTARE I LIBRI DELLA BIBLIOTECA
 - A NOI PER LA RUBRICA TELEFONICA
- E MILLE ALTRE APPLICAZIONI.

SCRIVERE UN PROGRAMMA COSI', E' VERAMENTE DIFFICILE E LUNGI DA ME L' INTENZIONE DI INSEGNARVELO, OCCORREREBERO 10 LIBRI COME QUESTO PER FARLO.

E' FACILE, E VEDREMO QUANTO, SCRIVER UN PROGRAMMA PERSONALIZZATO, CIOE' ORIENTATO ALLE PROPRIE ESIGENZE, CHE SVOLGA SOLO IL LAVORO CHE NOI VOGLIAMO.

NON CI IMPORTI DI CREARE UN PROGRAMMA CHE POTREBBE ANDARE BENE PER TUTTE LE PERSONE CHE ABBIAMO VISTO, MA SOLO PER NOI, PER LE NOSTRE ESIGENZE: QUESTO E' QUANTO FAREMO NEI PROSSIMI CAPITOLI.

Q U A L C H E S E M P L I C E
E S E M P I O

QUALCHE SEMPLICE ESEMPIO

VEDIAMO DI FARE QUALCHE ESEMPIO DI QUANTO DETTO, VEDREMO
COME SI SCRIVE E SI LEGGE, SU E DA UN FILES.

ECCO IL PRIMO PROBLEMA:

SCRIVERE UN PROGRAMMA CHE REGISTRI SU UN FILE LA PAROLA
'PROVA' SEGUITA DAL NUMERO DI RECORD PER 100 VOLTE.

```

1 REM *****
2 REM ***
3 REM *** QUESTO PROGRAMMA SCRIVE ***
4 REM ***
5 REM *** LA PAROLA 'PROVA' PER ***
6 REM ***
7 REM *** 100 VOLTE IN UN FILES. ***
8 REM ***
9 REM *****
10 PRINT CHR$(147):REC%=1
20 DOOPEN #1,"PROVA.DAT",D0,L20
30 A$="PROVA+STR$(REC%)
40 PRINT A$,REC%
50 RECORD #1,1,(REC%)
60 PRINT #1,A$
70 REC%=REC%+1
90 IF REC%>100 THEN DCLOSE:END
90 GOTO 30

```

COMMENTO

- LA RIGA 10 SERVE A CANCELLARE LO SCHERMO, CON IL
CHR\$(142), NATURALMENTE SI POTRA' USARE IL CUORICINO REVERSE
DEL COMMODORE. LA VARIABILE INTERA REC% INDICA CHE BISOGNA
LEGGERE DAL PRIMO RECORD DEL FILE, SERVE IN PRATICA COME
CONTATORE DI RECORD.

- LA RIGA 20 APRE IL FILE 'PROVA.DAT' SUL DRIVE 0 CON FILE
LOGICO 1 E CON RECORD TUTTI EGUALI DI 20 CARATTERI

- LA RIGA 30 DEFINISCE LA VARIABILE A\$ CON LA STRINGA

'PROVA' SOMMATA AL NUMERO DEL RECORD CORRENTE, CIOE' IN QUESTO MODO: PROVA 1, PROVA 2, PROVA 3, ECC. ECC.

- LA RIGA 40 PRODUCE UNA SCRITTA SULLO SCHERMO DI QUELLO CHE VERRA' SCRITTO NEL FILE (A\$) E DEL NUMERO DI RECORD CORRISPONDENTE (REC%)

- LA RIGA 50 SERVE A COMUNICARE AL DRIVE CHE SI HA INTENZIONE DI SCRIVERE SUL RECORD REC% DAL PRIMO CARATTERE DEI 20 A DISPOSIZIONE, TUTTO NATURALMENTE SUL FILE LOGICO 1.

- LA RIGA 50 SCRIVE SUL RECORD GIA' PREDISPOSTO L' INFORMAZIONE A\$ PRECEDENTEMENTE PREPARATA.

- LA RIGA 60 SERVE AD INCREMENTARE LA VARIABILE REC% PREDISPONENDOSI, COSI', A SCRIVERE NEL RECORD SUCCESSIVO.

- LA RIGA 70 CONTROLLA SE SIAMO ARRIVATI AL 100 ESIMO RECORDS, SE SI' VIENE CHIUSO IL FILE (DCLOSE) E IL PROGRAMMA TERMINA, SE NO SI PASSA ALLA RIGA SUCCESSIVA.

- LA RIGA 80 SERVE A SALTARE ALLA RIGA 30 PER RICOMINCIARE.

RICAPITOLANDO:

- SI APRE IL FILES
- SI PREPARA L' INFORMAZIONE
- SI PREDISPONE IL RECORD
- SI SCRIVE SUL FILES
- SI INCREMENTA IL RECORD
- SI CONTROLLA SE E' L' ULTIMO, SE SI CHIDERE IL FILE, SE NO AL PUNTO SUCCESSIVO
- RICOMINCIARE

SEMPLICE NO ? AVETE CREATO IL VOSTRO PRIMO FILES AD ACCESSO CASUALE SU DISCO; PER SINCERARVENE CHIAMATE LA DIRECTORY DEL

DISCO, DOVRESTE TROVARE, INSIEME ALLE ALTRE, LA SEGUENTE INFORMAZIONE:

4 "PROVA.DAT" REL

CIOE' IL FILE DA VOI CREATO ESISTE SU DISCO, OCCUPA 4 BLOCCHI CIOE' CIRCA 1 KBYTE ED E' UN FILE DI TIPO REL (RELATIVO).

DURANTE LA REGISTRAZIONE IL LED ROSSO DEL DRIVE RIMANE COSTANTEMENTE ACCESO, ED ALLA FINE DELL' OPERAZIONE DEVE SPEGNERE; SE DOVESSE, INVECE, LAMPEGGIARE E' CHIARO CHE QUALCHE COSA NON E' ANDATA PER BENE, IN QUESTO CASO CONTROLLATE IL LISTATO BATTUTO.

NATURALMENTE DOPO AVERE SCRITTO SI DEVE ESSERE IN GRADO DI LEGGERE LE INFORMAZIONI MEMORIZZATE.

ECCO IL SECONDO PROBLEMA:

SCRIVERE UN PROGRAMMA CHE LEGGA E MOSTRI SUL VIDEO I DATI MEMORIZZATI CON IL PRECEDENTE ESEMPIO.

```

1 REM *****
2 REM ***                                     ***
3 REM *** QUESTO PROGRAMMA SERVE A ***
4 REM ***                                     ***
5 REM *** LEGGERE I CONTENUTI DEL ***
6 REM ***                                     ***
7 REM *** FILE 'PROVA.DAT' ***
8 REM ***                                     ***
9 REM *****
10 PRINT CHR$(147):REC%=1
20 DOPEN #1,"PROVA.DAT",D0,L20
30 RECORD #1,(REC%)
40 INPUT #1,A$
50 PRINT A$
60 REC%=REC%+1
70 IF ASC(A$)=255 THEN DCLOSE:END
80 GOTO 30

```

COMMENTO

- LA RIGA 10 CANCELLA LO SCHERMO E SETTA LA VARIABILE REC%

AD UNO PER CONTARE I RECORDS

- LA RIGA 20 APRE IL FILE 'PROVA.DAT' CON GLI STESSI PARAMETRI DEL PROGRAMMA CHE HA GENERATO IL FILE SUL DISCO

- LA RIGA 30 SI POSIZIONA SUL RECORD, E PREDISPONE ALLA LETTURA

- LA RIGA 40 LEGGE FISICAMENTE L' INFORMAZIONE SELEZIONATA

- LA RIGA 50 SCRIVE IL DATO SULLO SCHERMO

- LA RIGA 60 INCREMENTA IL NUMERO DI RECORD, IN ALTRE PAROLE CI SI PREPARA A LEGGERE IL RECORD SUCCESSIVO

- LA RIGA 70 SERVE A CONTROLLARE SE SIAMO ALLA FINE DEL FILE, INFATTI IL CODICE ASCII DOPO L' ULTIMO RECORD E', APPUNTO, 255.

SE SIAMO ALLA FINE IL FILE VIENE CHIUSO, ALTRIMENTI SI PASSA AL PUNTO SUCCESSIVO

- LA RIGA 80 SERVE A RICOMINCIARE

RICAPITOLANDO:

- SI APRE IL FILE

- SI POSIZIONA IL RECORD DA LEGGERE

- SI LEGGE L' INFORMAZIONE

- SI CONTROLLA SE E' L' ULTIMO RECORD, SE SI' SI CHIUDE ALTRIMENTI SI PASSA AL PUNTO SUCCESSIVO

- SI INCREMENTA IL RECORD PER LA LETTURA SUCCESSIVA

- SI RICOMINCIA

UNO DEI PROBLEMI CHE SI PRESENTA IN FASE DI LETTURA E' CHE NON E' POSSIBILE, PENA MANDARE IN ERRORE IL DRIVE, LEGGERE UN FILE PIU' IN LA DELL' ULTIMO RECORD FISICAMENTE SCRITTO. QUESTO VUOL DIRE CHE, SE UN FILE E' COMPOSTO DA 100 RECORD

ED IO TENTO DI LEGGERNE IL 101 ESIMO, CADO IN UNA SITUAZIONE DI ERRORE CON CONSEGUENZE IMPREVEDIBILI.

SI EVINCE CHE IN LETTURA E' DI FONDAMENTALE IMPORTANZA CONOSCERE QUANTI RECORDS CONTIENE IL FILE CHE CI APPRESTIAMO A LEGGERE.

ESISTONO MOLTI METODI PER RISOLVERE QUESTO PROBLEMA, VEDIAMONE ALCUNI:

A) ALCUNI DIALETTI BASIC CONTENGONO UNA FUNZIONE DI FINE FILE (EOF, END OF FILE) CHE INDICA QUANDO IL FILE E' FINITO:

```
10 INPUT #1,A$
20 IF (EOF) THEN END
30 GOTO 10
```

CHE VUOL DIRE LEGGI FINO A QUANDO NON E' FINITO IL FILES. QUESTO COMANDO NON ESISTE NEL '128'.

B) IN GENERE IL SISTEMA OPERATIVO RESTITUISCE UN CARATTERE PARTICOLARE QUANDO SI TENTA DI LEGGERE PIU' IN LA' DELL' ULTIMO RECORDS, IL '128' GENERA UN CODICE ASCII PARI A 255 CHIAMATO TERMINATORE DI FILES.

IN QUESTA MANIERA E' STATO LETTO IL FILE NELL' ESEMPIO PRECEDENTE:

```
50 IF ASC(A$)=255 THEN 'DCLOSE:END
```

CHE VUOL DIRE, SE IL CODICE ASCII DEL CARATTERE LETTO E' 255 SIGNIFICA CHE SIAMO OLTRE L' ULTIMO RECORD, PER CUI CHIUDI IL FILES E TERMINA IL PROGRAMMA.

C) SI MEMORIZZA IL NUMERO DEI RECORDS SCRITTI NEL PRIMO RECORD DEL FILE.

COSI' FACENDO, IL PRIMO RECORD NON CONTERRA' INFORMAZIONI COME GLI ALTRI RECORD, MA SOLTANTO UN NUMERO CHE INDICA DI QUANTI RECORDS E' COMPOSTO L' INTERO FILES.

QUANDO SI VA' A LEGGERE, SI LEGGE IL PRIMO RECORD E POI SI VISUALIZZANO I DATI DAL SECONDO RECORD IN POI FINO AL MASSIMO NUMERO LETTO SUL PRIMO RECORD.

QUESTI METODI SARANNO USATI IN ESEMPI SUCCESSIVI UN PO' PIU' COMPLESSI.

PER LEGGERE UN FILE , ESISTE UN' ALTRA TECNICA PIU' PRECISA E PIU' SICURA.

QUESTA TECNICA CONSISTE NEL LEGGERE IL FILE UN CARATTERE ALLA VOLTA, TRAMITE LA FUNZIONE ' GET '.

IL PROGRAMMA CHE SEGUE LEGGE I DATI DEL FILE 'PROVA.DAT' PRECEDENTEMENTE CREATO.

```

1 REM *****
2 REM ***                               ***
3 REM *** QUESTO PROGRAMMA LEGGE IL ***
4 REM ***                               ***
5 REM *** FILE 'PROVA.DAT' CON LA ***
6 REM ***                               ***
7 REM *** FUNZIONE GET. ***
8 REM ***                               ***
9 REM *****
10 PRINT CHR$(147):REC%=1
20 DOPEN #1,"PROVA.DAT",D0,L20
30 RECORD #1,REC%
40 FOR C=1 TO 8:GET #1,A$:B$=B$+A$:NEXT
50 PRINT B$
60 REC%=REC%+1:B$=""
70 GOTO 30

```

COMMENTO

- LA RIGA 10 CANCELLA LO SCHERMO E PREPARA LA VARIABILE REC%

- LA RIGA 20 APRE IL FILE CON GLI STESSI PARAMETRI DEL PROGRAMMA CHE LO HA SCRITTO

- LA RIGA 30 SI POSIZIONA SUL RECORD

- LA RIGA 40 RAPPRESENTA LA NOVITA', INFATTI QUESTA RIGA CONSISTE IN UN CICLO FOR NEXT CHE, TRAMITE LA GET LEGGE IL RECORD, O MEGLIO 8 CARATTERI DEL RECORD, UN CARATTERE ALLA VOLTA, E SEMPRE NELLA STESSA RIGA, CARATTERE PER CARATTERE VIENE COSTRUITO IL DATO NELLA VARIABILE B\$.

FACCIAMO UN ESEMPIO PRATICO.

SUPPONIAMO DI VOLERE LEGGERE LA PAROLA 'PROVA', LA ROUTINE FUNZIONA COSI':

GET#1,A\$	B\$
P	P
R	PR
O	PRO
V	PROV
A	PROVA

IN ALTRE PAROLE LA GET LEGGE UN CARATTER E QUESTO VIENE CONSERVATO, SOMMATO AI PRECEDENTI, NELLA VARIABILE B\$.

- LA RIGA 50 SCRIVE IL LA SOMMA DI CARATTERI, CHE RAPPRESENTA IL DATO COMPLETO, RICOSTRUITO, SUL VIDEO

- LA RIGA 60 INCREMENTA IL RECORD E AZZERA LA VARIABILE B\$, QUESTO SERVE A PREDISPORSI PER LA LETTURA SUCCESSIVA

- LA RIGA 70 FA RICOMINCIARE IL LA LETTURA DAL PROSSIMO RECORDS.

IN QUESTO PROGRAMMA NON SI E TENUTO CONTO DELLA FINE DEL FILE, INFATTI UNA VOLTA LANCIATO, ESSO, LEGGERA' TUTTI I RECORDS CORRETTAMENTE SCRITTI, DOPO DI CHE' ANDRA' IN ERRORE (SPIA ROSSA DEL DRIVE CHE LAMPEGGIA).

LA LETTURA, IN TUTTI GLI ESEMPI SVOLTI E' STATA FATTA IN MANIERA SEQUENZIALE, CIOE' UN RECORD DOPO L' ALTRO, PER TUTTO IL FILE.

SAPPIAMO PERO', CHE L' ACCESSO DIRETTO CI CONSENTE DI ACCEDERE A QUALUNQUE TIPO DI RECORDS APPARTENENTE AL FILE, SENZA DOVERE PER QUESTO LEGGERE TUTTI GLI ALTRI PRECEDENTI O SUCCESSIVI CHE SIANO.

VEDIAMO COME FARE A REALIZZARE UN PROGRAMMINO IN GRADO DI LEGGERE RECORDS DEL FILE 'PROVA.DAT' IN MANIERA DIRETTA, SFRUTTANDO, COSI' LA POTENZA DI QUESTO TIPO DI FILES.

IL PROGRAMMA CHE SEGUE LEGGE QUALUNQUE RECORD COMPRESO NEL FILE TRAMITE IL NUMERO PROGRESSIVO DEL RECORD STESSO. IN ALTRE PAROLE IL POSTO CHE IL RECORD CERCATO OCCUPA NEL FILE SARA' LA CHIAVE DI LETTURA, PIU' AVANTI VEDREMO COME USARE DELLE CHIAVI PIU' COMODE COME AD ESEMPIO UN CAMPO STESSO DEL RECORD IN MODO DA POTER LEGGERE UN RECORD INDICANDONE IL NOME E NON IL NUMERO.

USARE COME CHIAVE DI LETTURA IL NUMERO PROGRESSIVO DEL RECORD E' IL METODO DI RICERCA PIU' FACILE IN QUANTO INTERAMENTE GESTITO DAL SISTEMA OPERATIVO DEL '128', BASTA DIRGLI DI LEGGERE IL 40 ESIMO RECORD E LUI OBBEDIRA' AI NOSTRI COMANDI, PER LEGGERE CON CHIAVI DIVERSE, DEVE ESSERE IL PROGRAMMATORE A GESTIRE TUTTO, QUINDI LA COSA SI COMPLICA UN TANTINO, MA VEDREMO IN SEGUITO COME RISOLVERE QUESTI PROBLEMI, MA VEDIAMO IL PROGRAMMA:

```

1 REM *****
2 REM *** QUESTO PROGRAMMA LEGGE IL ***
3 REM *** CONTENUTO DEL FILE IN ***
4 REM *** MODO RELATIVO, CIOE' CON ***
5 REM *** RECORD SELEZIONATI DAL ***
6 REM *** PROGRAMMATORE, SI SUPPONE ***
7 REM *** CHE IL FILE SIA FORMATO ***
8 REM *** DA 100 RECORDS ***
9 REM *****
10 PRINT CHR$(147)
20 DOPEN #1,"PROVA.DAT",D0,L20
30 INPUT"DAMMI IL NUMERO DEL RECORD 0 PER FINIRE ";REC%
35 IF REC%=0 THEN DCLOSE:END
40 IF REC%>100 THEN 30
50 RECORD #1,REC%
60 INPUT#1 A$
70 PRINT A$
80 GOTO 30

```

COMMENTO

- LA RIGA 10 COME AL SOLITO SERVE A CANCELLARE LO SCHERMO

- LA RIGA 20 APRE IL FILE PROVA.DAT, CON GLI STESSI PARAMETRI DEL PROGRAMMA CHE LO HA CREATO

- LA RIGA 30 CHIEDE ALL' OPERATORE IL NUMERO DEL RECORD CHE INTENDE CERCARE NEL FILE

- LA RIGA 35 CONTROLLA SE REC% E' UGUALE A ZERO, IN QUESTO CASO CHIUDE IL FILE E TERMINA IL PROGRAMMA, TUTTO QUESTO SERVE A CREARE UN MODO PER POTERE USCIRE DAL PROGRAMMA

- LA RIGA 40 CONTROLLA SE IL NUMERO DI RECORD SELEZIONATO SUPERA L' AMPIEZZA DEL FILE, APPUNTO 100 RECORD, SE SI, IL DATO VIENE RIFIUTATO, ALTRIMENTI VIENE ACCETTATO

- LA RIGA 50 SERVE A POSIZIONARE LA LETTURA SUL RECORD VOLUTO DALL' OPERATORE

- LA RIGA 60 LEGGE FISICAMENTE IL RECORD NEL FILE DI DATI

- LA RIGA 70 SCRIVE IL RISULTATO DELLA LETTURA SUL VIDEO

- LA RIGA 80 RIMANDA DI NUOVO ALL' INIZIO DEL PROGRAMMA, PRONTI PER UN' ALTRA RICHIESTA.

SE TUTTO E' ANDATO PER IL MEGLIO, ADESSO, SAPETE SCRIVERE UN FILE RELATIVO E POI LEGGERLO SEQUENZIALMENTE O PER NUMERO DI RECORD.

PIU' AVANTI IMPAREREMO A LEGGERE UN RECORD CON CHIAVI DI ACCESSO DIVERSE PER UNA PIU' FUNZIONALE LETTURA.

Q U A L C H E E S E M P I O
P I U ' C O M P L E S S O

QUALCHE ESEMPIO PIU' COMPLESSO

PROVIAMO A SCRIVERE UN PICCOLO PROGRAMMA CHE CREI SU DISCO UNA AGENDA TELEFONICA IN GRADO DI MEMORIZZARE I NOMI DEI NOSTRI AMICI CON I RISPETTIVI NUMERI DI TELEFONO. LA CREAZIONE DI UN ARCHIVIO DEL GENERE, MA SOPRATTUTTO LA SUA GESTIONE PREVEDONO ALCUNE FASI FONDAMENTALI:

A) SCRITTURA NEL FILE CON POSSIBILITA' DI ACCODARE EVENTUALI NOMI NUOVI (APPEND)

B) LETTURA SEQUENZIALE DI TUTTI I NOMI CHE COSTITUISCONO IL FILE

C) CANCELLAZIONE DI NOMINATIVI NEL FILE

D) RICERCA SELETTIVA DI NOMINATIVI, SFRUTTANDO APPIENO, L' ACCESSO DIRETTO.

POCO ALLA VOLTA COSTRUIREMO TUTTO CIO'.

COMINCIAMO DALLA PRIMA FASE LA SCRITTURA: IN QUESTO MOMENTO IL PROGRAMMATORE DEVE PENSARE AD ALCUNI CONCETTI FONDAMENTALI PER ORGANIZZARE IL PROPRIO ARCHIVIO, ESATTAMENTE:

A) DEFINIZIONE DEI CAMPI E QUINDI LUNGHEZZA DEL RECORD

B) MODO DI CONSERVARE IL NUMERO DEI RECORD GIA' SCRITTI.

NOI COSTRUIREMO UN ARCHIVIO CON LE SEGUENTI CARATTERISTICHE:

CAMPI:

1) NOME	CARATTERI	15
2) COGNOME	CARATTERI	20
3) TELEFONO	CARATTERI	15

T O T A L E	CARATTERI	50

IL NUMERO DEI RECORD SCRITTI SARANNO CONSERVATI NEL PRIMO RECORD DEL FILE.

ESAMINIAMO LA SEQUENZA LOGICA DEL PROGRAMMA:

- PREPARA LO SCHERMO
- APRI IL CANALE
- CHIEDI IN INPUT IL NOME
- CONTROLLA SE IL NOME E' PIU' LUNGO DI 15 CARATTERI SE SI RIFIUTA IL DATO E PASSA AL PUNTO PRECEDENTE
- SE IL NOME E' FORMATO DA MENO DI 15 CARATTERI, RIEMPI I RIMANENTI SPAZI CON UN BLANK (BARRA SPAZIATRICE)
- CHIEDI IN INPUT IL COGNOME
- CONTROLLA SE IL COGNOME E' PIU' LUNGO DI 20 CARATTERI, SE SI RIFIUTA IL DATO E PASSA AL PUNTO PRECEDENTE
- SE IL COGNOME E' FORMATO DA MENO DI 20 CARATTERI, RIEMPI I RIMANENTI SPAZI CON BLANK (BARRA SPAZIATRICE)
- CHIEDI IN INPUT IL TELEFONO
- CONTROLLA SE IL TELEFONO E' PIU' LUNGO DI 15 CARATTERI, SE SI RIFIUTA IL DATO E PASSA AL PUNTO PRECEDENTE
- DE IL TELEFONO E' FORMATO DA MENO DI 15 CARATTERI, RIEMPI I RIMANENTI SPAZI CON BLANK (BARRA SPAZIATRICE)
- COSTRUISCI IL RECORD, COME SOMMA DEI SINGOLI CAMPI
- PREDISPONI UN CONTROLLO PER POTERE USCIRE CORRETTAMENTE DAL PROGRAMMA
- POSIZIONATI SUL GIUSTO RECORD
- SCRIVI NEL RECORD VOLUTO

- INCREMENTA IL NUMERO DI RECORD

- RITORNA ALL' INIZIO

FACILE NO ? IN QUESTA SEQUENZA LOGICA VI SONO ALCUNI CONCETTI NUOVI COME AD ESEMPIO, IL RIEMPIMENTO DEL CAMPO CON BLANK FINO AL COMPLETAMENTO DELLA LUNGHEZZA MASSIMA PREVISTA, VEDIAMO PERCHE':

VITTORIO	*PAOLA	*321190	*
----------	--------	---------	---

SUPPONIAMO CHE QUESTO SIA UN RECORD E CHE IL CARATTERE * RAPPRESENTI IL CONFINE DI UN CAMPO RISPETTO ALL' ALTRO.

E' ESTREMAMENTE IMPORTANTE CHE NON AVVENGANO SCONFINAMENTI TRA CAMPI, O PEGGIO FRA RECORD ADIACENTI, SE CIO' DOVESSE AVVENIRE SAREBBE COMPROMESSA LA SUCCESSIVA FASE DI LETTURA DEI DATI E QUINDI LA RICERCA STESSA DELLE INFORMAZIONI.

E', ALTRETTANTO, IMPORTANTE SAPERE, PER UNA CORRETTA LETTURA SAPERE QUANTO E' LUNGO UN CAMPO ECCO PERCHE' IL CAMPO SI COMPLETA SEMPRE CON DEGLI SPAZI BIANCHI FINO A COMPLETAMENTO DELLA LUNGHEZZA MASSIMA.

VEDIAMO IL CODICE BASIC CORRISPONDENTE IN VERSIONE '128':

```

1 REM *****
2 REM ***
3 REM *** PROGRAMMA CHE SCRIVE IN ***
4 REM ***
5 REM *** UN FILE UNA AGENDA ***
6 REM ***
7 REM *** TELEFONICA ***
8 REM ***
9 REM *****
10 SCNCLR:REC%=1
20 DOPEN #1,"NOMI.DAT",D0,55
30 INPUT "NOME ";N$:N=LEN(N$)
40 IF N>15 THEN 30
45 FOR C=1 TO 15-N:N$=N$+" ":NEXT C
50 INPUT "COGNOME ";C$:B=LEN(C$)
60 IF B>20 THEN 50

```

```

65 FOR C=1 TO 20-B:C$=C$+" ":NEXT C
70 INPUT "TELEFONO ";T$:T=LEN(T$)
80 IF T>15 THEN 70
85 FOR C=1 TO 15-T:T$=T$+" ":NEXT C
90 PRINT "PREMI 'E' PER FINIRE"
100 FIN$=N$+C$+T$
110 GET A$:IF A$="" THEN 110
120 IF A$="E" THEN DCLOSE:END
130 RECORD #1,(REC%)
140 PRINT #1,FIN$
150 REC%=REC%+1
160 GOTO 30

```

COMMENTO

- LA RIGA 10 SERVE A CANCELLARE LO SCHERMO E A SETTARE A 1 LA VARIABILE CHE CONTA IL NUMERO DI RECORD.
DA NOTARE LA NUOVA ISTRUZIONE DEL '128' CHE SERVE A CANCELLARE LO SCHERMO.

- LA RIGA 20 SERVE AD APRIRE IL FILE DI NOME 'NOMI.DAT' ALLA SOLITA MANIERA.

- LA RIGA 30 SERVE A CHIEDERE IL NOMINATIVO ALL' UTENTE E A MEMORIZZARLO NELLA VARIABILE N\$.
IN N VIENE MESSO IL NUMERO DI CARATTERI CHE COMPONGONO LA VARIABILE N\$, SERVIRA' IN SEGUITO

- LA RIGA 40 SERVE A CONTROLLARE, ATTRAVERSO LA VARIABILE N SE IL DATO N\$ E' COM

- LA RIGA 45 SERVE, COME GIA' DETTO, A RIEMPIRE IL CAMPO DI SPAZI BIANCHI, FINO AL COMPLETAMENTO DELLA MASSIMA LUNGHEZZA

- DALLA RIGA 50 ALLA 85 VENGONO EFFETTUATE LE STESSE OPERAZIONI PER I RIMANENTI DUE CAMPI, COGNOME E TELEFONO

- LA RIGA 90 STAMPA UN MESSAGGIO SULLO SCHERMO

- LA RIGA 100 COMPONE LA VARIABILE FIN\$ COME SOMMA DEI SINGOLI CAMPI, PREPARA, INSOMMA, IL RECORD PRONTO PER ESSERE SCRITTO NEL FILE

- LA RIGA 110 TRAMITE LA FUNZIONE GET PRENDE UN CARATTERE DALLA CODA DEL BUFFER DI TASTIERA SE QUESTO CARATTERE E' NULLO SI RIMANE SULLA STESSA LINEA. QUESTA LINEA, IN PRATICA, CONSENTE L' ARRESTO DEL PROGRAMMA FINO A CHE UN TASTO QUALSIASI NON VENGA PREMUTO.

- LA RIGA 120 CONTROLLA SE PER CASO IL CARATTERE RILEVATO DALLA GET NON SIA UNA E, NEL QUAL CASO VIENE CHIUSO IL FILE E IL PROGRAMMA TERMINA

- LA RIGA 130 SERVE A POSIZIONARE LA SCRITTURA SUL RECORD VOLUTO, IMPOSTATO DALLA VARIABILE REC%

- LA RIGA 140 SCRIVE SUL RECORD SELEZIONATO

- LA RIGA 150 INCREMENTA IL NUMERO DI RECORD

- LA RIGA 160 RIMANDA ALL' INIZIO DEL PROGRAMMA

IL PROGRAMMA IN QUESTA VERSIONE SCRIVE NEL FILE TUTTA UNA SERIE DI DATI, MA ANCORA NON E' RISOLTO IL PROBLEMA DI MEMORIZZARE IL NUMERO DEI RECORD SCRITTI.

MODIFICHIAMO IL PROGRAMMA IN MODO CHE ALLA FINE REGISTRI SU DI UN FILE RELATIVO IL NUMERO DEI RECORD PRESENTI NEL FILE. LA PRIMA OPERAZIONE CHE IL PROGRAMMA DEVE FARE E' QUELLA DI ANDARE A LEGGERE QUANTI RECORD CI SONO SCRITTI NEL FILE E, IN CASO DI AGGIUNTE DI DATI, COMINCIARE AD AGGIUNGERE DAL RECORD SUCCESSIVO ALL' ULTIMO SCRITTO, ONDE EVITARE SPIACEVOLI SOVRAPPOSIZIONI DI DATI. ECCO COME MODIFICARE IL PROGRAMMA:

```
10 SCNCLR:GOSUB 500
```

LE RIGHE DI PROGRAMMA DALLA 20 ALLA 110 NON SI CAMBIANO

```
120 IF A$="E" THEN GOSUB 600:DCLOSE:END
```

LE RIGHE FINO ALLA FINE RIMANGONO UGUALI
AGGIUNGERE LE SEGUENTI LINEE DI PROGRAMMA:

```

500 REM LEGGE IL FILE RELATIVO 'NREC.NUM'
510 DOPEN #1,"NREC.NUM",D0,L5
520 RECORD #1,1
530 INPUT #1,REC$
540 REC%=VAL(REC$)
550 DCLOSE
560 IF REC%=0 THEN REC%=1
570 RETURN
600 REM SCRIVE IL NUMERO DI RECORDS
610 DOPEN #1,"NREC.NUM",D0,L5
620 RECORD #1,1
640 REC$=STR$(REC%)
650 PRINT #1,REC$
660 DCLOSE
670 RETURN

```

CON QUESTE MODIFICHE, IL PROGRAMMA COME PRIMA COSA VA A LEGGERE NEL FILE IL NUMERO DI RECORD SCRITTI, SE QUESTO NUMERO E' 0, CIOE' E' LA PRIMA VOLTA CHE SI SCRIVE NEL FILE, LA VARIABILE REC% VIENE POSTA AD 1.

UNA EVENTUALE AGGIUNTA DI NUOVI DATI VIENE ACCODATA AI PRECEDENTI, PROPRIO PERCHE' IL PROGRAMMA CONOSCE IL NUMERO DEI RECORD GIA' SCRITTI.

QUANDO SI DECIDE DI USCIRE DAL PROGRAMMA, SI SALTA ALLA SOUBROUTINE CHE PARTE DALLA RIGA 600 PER AGGIORNARE IL NUMERO DI RECORD SCRITTI NEL FILE, POI SI ESCE.

SE TUTTO E' ANDATO PER IL MEGLIO, CHIAMANDO LA DIRECTORY DEL DISCO, DEVONO TROVARSI DUE NUOVI FILES CHIAMATI:

```

XXX      "RECORD.DAT"          REL
XXX      "NREC.NUM"           REL

```

DOVE XXX RAPPRESENTA L' OCCUPAZIONE DEL DISCO ESPRESSA IN BLOCCHI, RICORDO CHE 4 BLOCCHI CORRISPONDONO AD 1 KBYTES DI MEMORIA IMPEGNATA.

SCRIVIAMO ADESSO UN PROGRAMMA IN GRADO DI LEGGERE IL FILE SCRITTO IN PRECEDENZA, VEDIAMO IL CODICE BASIC:

```

1 REM *****
2 REM ***                               ***

```

```

3 REM *** QUESTO PROGRAMMA E' IN ***
4 REM *** ***
5 REM *** GRADO DI LEGGERE QUANTO ***
6 REM *** ***
7 REM *** SCRITTO IN PRECEDENZA ***
8 REM *** ***
9 REM *****
10 SCNCLR:GOSUB 500
20 DOPEN #1,"NOMI.DAT",D0,L55          30 RECORD #1,REC%
40 INPUT #1,FIN$
50 PRINT FIN$
60 REC%=REC%+1
65 IF REC%>RE% THEN DCLOSE:END
70 GOTO 30
500 REM LEGGE IL NUMERO DI RECORD SCRITTI
510 DOPEN #1,"NREC.NUM",D0,L5
520 RECORD #1,1
530 INPUT REC$
540 RE%=VAL(REC$)
550 IF RE%=0 THEN PRINT "FILE VUOTO":REC%=1:RETURN
560 REC%=RE%:RETURN

```

QUESTO PROGRAMMA LEGGE IN MANIERA SEQUENZIALE, CIOE' UN RECORD DOPO L' ALTRO, TUTTO IL FILE.

LA RIGA 65 LEGGE LA FINE DEL FILE, QUESTO PUO' ESSERE FATTO ATTRAVERSO L' ANALISI DELLA VARIABILE DI STATO SD, LA QUALE RIPORTA, NEL CASO SE NE VERIFICHINO, IL CODICE DI ERRORE. QUANTO SI TENTA DI LEGGERE PIU' DEL DOVUTO UN FILE, VIENE GENERATO UN ERRORE CHE LA VARIABILE SD RIPORTA CON IL NUMERO 50. PERTANTO LA FINE DEL FILE SI PUO' ANCHE TESTARE CON LA SEGUENTE LINEA DI PROGRAMMA:

```
65 IF SD>50 THEN DCLOSE:END
```

E' MOLTO IMPORTANTE RILEVARE IL CODICE DI ERRORE SUBITO DOPO IL POSIZIONAMENTO EFFETTUATO CON L' ISTRUZIONE RECORD #. ECCO IL COMMENTO COMPLETO AL PROGRAMMA:

- LA RIGA 10 CANCELLA LO SCHERMO E RIMANDA ALLA SOUBROUTINE POSTA ALLA RIGA 500 INCARICATA DI LEGGERE IL NUMERO DI RECORDS SCRITTI, QUESTO NUMERO VIENE POSTO NELLA VARIABILE

RF

- LA RIGA 20 APRE IL FILE 'NOMI.DAT' CON LE SOLITE MODALITA'
- LA RIGA 30 SERVE AL POSIZIONAMENTO AL RECORD VOLUTO INDICATO DALLA VARIABILE REC%
- LA RIGA 40 LEGGE IL RECORD E LO PONE NELLA VARIABILE FIN%
- LA RIGA 50 SCRIVE IL RECORD SULLO SCHERMO
- LA RIGA 60 INCREMENTA IL NUMERO DI RECORD
- LA RIGA 65, GIA' COMMENTATA, SERVE A VEDERE SE SI TRATTA DELL' ULTIMO RECORD
- LA RIGA 70 RIMANDA ALL' INIZIO

IL PEZZO DI PROGRAMMA DALLA RIGA 500 IN GIU' SI COMMENTA DA
SE' E SERVE A LEGGERE DALL' APPOSITO FILE, IL NUMERO DI
RECORD SCRITTI.

NATURALMENTE E' POSSIBILE RAGGRUPPARE I DUE PROGRAMMI IN UNO UNICO CON UN MENU' CHE CONSENTA DI SELEZIONARE LA LETTURA O LA SCRITTURA DEI DATI.

ECCO, ORA, IL LISTATO COMPLETO DI UN PROGRAMMA CHE SCRIVE SU
DI UN FILE (ANCHE AGGIUNGENDO DATI AD ALTRI GIÀ PRESENTI,
E FORNISCE A RICHIESTA UN REPORT SULLA STAMPANTE.
PER REPORT SI INTENDE UNA STAMPA ORDINATA DEI CONTENUTI DI
UN FILE.

```

1 REM *****
2 REM ***
3 REM *** PROGRAMMA COMPLETO DI ***
4 REM *** LETTURA E SCRITTURA SU ***
5 REM *** UN FILE CON OUTPUT SU ***
6 REM *** VIDEO O SU STAMPANTE ***
7 REM *** COMPLETO DI MENU ***
8 REM ***
9 REM *****
10 SCNCCLR:CLR:SW=0

```

```

20 PRINT:PRINT:PRINT
30 PRINT" PREMI (1) PER SCRIVERE NEL FILE":PRINT
40 PRINT" PREMI (2) PER LEGGERE IL FILE":PRINT
50 PRINT" PREMI (3) PER REPORT SU STAMPANTE ":PRINT
60 PRINT" PREMI (4) PER FINE LAVORO "
70 GET A$: IF A$="" THEN 70
80 ON VAL(A$) GOTO 100,500,3000,5000
90 GOTO 70
100 SCNCLR:GOSUB 6000:REC%=REC%+1
110 DOPEN #1,"NOMI.DAT",D0,L55
120 INPUT"NOME ";N$:N=LEN(N$)
130 IF N>15 THEN120
140 INPUT"COGNOME ";C$:B=VAL(C$)
150 IF B>20 THEN 140
160 INPUT"TELEFONO ";T$:T=VAL(T$)
170 IF T>15 THEN 160
180 FOR C=1 TO 15-N:N$=N$+" ":NEXT C
190 FOR C=1 TO 20-B:C$=C$+" ":NEXT C
200 FOR C=1 TO 15-T:T$=T$+" ":NEXT C
210 PRINT " PREMI 'E' PER FINIRE "
220 FIN$=N$+C$+T$
230 GET A$: IF A$="" THEN 230
240 IF A$="E" THEN GOSUB 6100:DCLOSE:GOTO 10
250 RECORD #1,(REC%)
260 PRINT #1,FIN$
270 REC%=REC%+1
280 GOTO 120
500 SCNCLR:REC%=1
510 DOPEN #1,"NOMI.DAT",D0,L55
520 RECORD #1,(REC%)
522 IF DS=50 AND SW=1 THEN DCLOSE:PRINT #4:CLOSE #4:GOTO 10
525 IF DS=50 THEN DCLOSE:GOTO 900
530 INPUT #1,FIN$: IF ASC(FIN$)=255 THEN 560
540
N$=LEFT$(FIN$,15):C$=MID$(FIN$,15,20):T$=RIGHT$(FIN$,15):GOS
UB 1000
545 IF SW=1 THEN PRINT #4,N1$,C1$,T1$:GOTO 560
550 PRINT N1$,C1$,T1$
560 REC%=REC%+1:N1$="":C1$="":T1$=""
570 GOTO 520
900 PRINT:PRINT:PRINT "FINE DATI PREMI UN TASTO "

```

```

910 GET A$:IF A$="" THEN 910
920 GOTO 10
1000 FOR C=1 TO 15:D$=MID$(N$,C,1)
1010 IF D$=CHR$(32) THEN 1030
1020 N1$=N1$+D$
1030 NEXT C:D$=""
1040 FOR C=1 TO 20:D$=MID$(C$,C,1)
1050 IF D$=CHR$(32) THEN 1070
1060 C1$=C1$+D$
1070 NEXT C:D$=""
1080 FOR C=1 TO 15:D$=MID$(T$,C,1)
1090 IF D$=CHR$(32) THEN 2010
2000 T1$=T1$+D$
2010 NEXT C:D$=""
2020 RETURN
3000 OPEN#4:SW=1:GOTO 500
5000 DCLOSE:SCNCLR:PRINT "FINE LAVORO":END
6000 DOPEN #2,"NREC",D0,L5
6010 RECORD #2,1
6020 INPUT #2,REC$
6030 REC%=VAL(REC$)
6040 DCLOSE #2
6050 RETURN
6100 DOPEN #2,"NREC",D0,L5
6110 RECORD #2,1
6120 REC%=STR$(REC%)
6130 PRINT #2,REC$
6140 DCLOSE #2
6150 RETURN

```

COMMENTO

- LINEE 10-90, IN QUESTE LINEE VIENE DEFINITO IL MENU' CHE CI CONSENTE DI LA SCELTA FRA PIU' OPZIONI.
 L' ESTETICA DI QUESTO MENU' NON E' DELLE MIGLIORI, ANZI E' PROPRIO TRASCURATA, MA LO SCOPO DI QUESTO LIBRO E' QUELLO DI IMPARARE AD USARE I FILES, COMUNQUE AFFIDO ALLA VOSTRA FANTASIA, OGNI POSSIBILE MIGLIORIA ALLA SCHERMATA INIZIALE.
 L' INPUT DELLA RISPOSTA E' CONTROLLATO, E GESTITO IN MANIERA TALE DA ACCETTARE SOLO I NUMERI PREVISTI NEL MENU', OGNI ALTRO TASTO E' RIFIUTATO.

E' BUONA NORMA CURARE PARTICOLARMENTE QUESTO ASPETTO DEI PROGRAMMI PER EVITARE ALL' UTENTE FINALE OGNI POSSIBILE ERRORE.

- LINEE 100-280, IN QUESTE LINEE E' CODIFICATA LA ROUTINE CHE SI OCCUPA DELLA SCRITTURA NEL FILE.

NESSUNA NOVITA' RISPETTO AI PRECEDENTI ESEMPI.

LA ROUTINE LEGGE PRIMA DI OGNI COSA, IL NUMERO DEI RECORD SCRITTI E COMINCIA A SCRIVERE DAL SUCCESSIVO, EVITANDO, COSI', SPIACEVOLI SOVRAPPOSIZIONI DI DATI.

- LINEE 500-2020, IN QUESTE LINEE E' CODIFICATA LA ROUTINE DI LETTURA DEI DATI E LA LORO PRESENTAZIONE O SU VIDEO O SU STAMPANTE.

IN QUESTA ROUTINE UNA NOVITA', LA VARIABILE SW, CON IL SUO STATO FA DA DEVIATORE VERSO LA STAMPANTE O VERSO IL VIDEO: SE SW=1 ALLORA L' OUTPUT DEI DATI E' DIRETTO ALLA STAMPANTE, SE SW=0 L' OUTPUT E' RIVOLTO AL VIDEO.

IN QUESTE RIGHE E' COMPRESA LA ROUTINE CHE ESTRAE, CON DEI CICLI FOR...NEXT E USANDO L' ISTRUZIONE BASIC MID\$, LE INFORMAZIONI DAL RECORD, ELIMINANDO GLI SPAZI BIANCHI.

LE ROUTINE PRESENTATE SONO ANCORA 'GREZZE' MA UNA VOLTA CAPITO IL PRINCIPIO DI FUNZIONAMENTO, SARA' UN OTTIMO ESERCIZIO CERCARE DI APPORTARE DELLE MIGLIORIE AL PROGRAMMA.

- LINEA 3000, QUI VIENE APERTO IL CANALE DELLA STAMPANTE E VIENE SETTATA LA VARIABILE SW AD 1 PER INDIRIZZARE L' OUTPUT DEI DATI SULLA STAMPANTE.

- LA LINEA 5000, ASSOLVE AL FINE LAVORO, CHIUDENDO TUTTI I FILES ED EMETTENDO SU VIDEO UN MESSAGGIO DI SALUTO.

- DALLA LINEA 6000 IN POI SONO LOCATE ALCUNE ROUTINE CHE SERVONO ALLA LETTURA ED AGGIORNAMENTO DEL FILE CHE CONTIENE IL NUMERO DEI RECORD SCRITTI.

QUESTO PROGRAMMA RAPPRESENTA UN VALIDO ESEMPIO DI COME SI PUO' ORGANIZZARE UN FILE (NEL NOSTRO CASO UNA RUBRICA TELEFONICA) E COME ORGANIZZARE UNA LETTURA DEI DATI O UN REPORT RIEPILOGATIVO.

UN ASPETTO ANCORA NON CURATO E' LA RICERCA DI UN RECORD

SPECIFICO ALL' INTERNO DEL FILE, SFRUTTANDO QUELLE CHE SONO LE CARATTERISTICHE PECULIARI DELL' ACCESSO DIRETTO. VEDREMO CON ALTRI ESEMPI, COME CODIFICARE TUTTO CIO' E COME CREARE DEI SOTTOPROGRAMMI, CHE CONSENTANO DI SELEZIONARE, SECONDO ALCUNI CRITERI, DEI RECORD APPARTENENTI AL FILE.

C O M E R I C E R C A R E
I D A T I

COME RICERCARE I DATI

UN OPERATORE NON CONOSCE NECESSARIAMENTE LA POSIZIONE DEL RECORD (PER POSIZIONE SI INTENDE IL POSTO CHE OCCUPA ALL' INTERNO DEL FILE) IN CUI SONO MEMORIZZARE LE INFORMAZIONI RELATIVE AD UN LIBRO, O A UN CLIENTE OPPURE AD UNA FATTURA.

IL PROBLEMA DIVENTA: COME REPERIRE IN UN FILE, FORMATO DA MOLTI RECORD, UNA INFORMAZIONE SPECIFICA ?.

IN GENERE QUESTO PROBLEMA VIENE SINTETIZZATO CON UNA ESPRESSIONE 'ACCESSO PER CHIAVE'.

PER CHIAVE SI INTENDE, UNA QUALCHE MANIERA CHE SERVA A SELEZIONARE FRA MOLTI RECORD, QUELLO CHE CONTIENE LE INFORMAZIONI CHE INTERESSANO.

QUANDO SCRIVIAMO I DATI IL SISTEMA OPERATIVO DELLA MACCHINA ATTRIBUISCE AD OGNI RECORD REGISTRATO UN NUMERO PROGRESSIVO ALL' INTERNO DEL FILE, QUESTA NUMERAZIONE RAPPRESENTA UNA RUDIMENTALE CHIAVE DI ACCESSO E' POSSIBILE ACCEDERE AL RECORD NUMERO 20, IL QUALE IN QUESTA MANIERA SARA' SELEZIONATO FRA MOLTI ALTRI.

QUESTO MODO DI REPERIRE LE INFORMAZIONI E' MOLTO RUDIMENTALE MA E' L' UNICO CHE LA MACCHINA GESTISCE AUTONOMAMENTE, QUALUNQUE ALTRO SISTEMA PIU' EVOLUTO DEVE ESSERE GESTITO DA PROGRAMMA.

E' CERTAMENTE SCOMODO USARE COME CHIAVE IL NUMERO DEL RECORD PERCHE' NON SAPPIAMO ESATTAMENTE COSA CI SARA' SCRITTO NEL RECORD SELEZIONATO.

PER VOLERE FARE UN ESEMPIO, E' MOLTO PIU' COMODO DIRE ALLA MACCHINA DI REPERIRE IL RECORD RELATIVO A MARIO ROSSI CHE NON PROCEDERE A TENTATIVI CON I NUMERI PROGRESSIVI: DOVE SARA' MEMORIZZATO IL RECORD MARIO ROSSI AL 20 ESIMO POSTO OPPURE AL 156 ESIMO POSTO, O ANCORA AL 45 ESIMO POSTO ? DIFFICILE A DIRSI, MOLTO MEGLIO DIRE CERCA IL RECORD ' MARIO ROSSI ', SENZA PREOCCUPARSI QUALE SARA' IL NUMERO PROGRESSIVO.

TUTTO QUESTO PRESUPPONE UNA GESTIONE VIA SOFTWARE DA PARTE DEL PROGRAMMATTORE, GESTIONE CHE DEVE ESSERE IL PIU' RAZIONALE POSSIBILE, PER NON APPESANTIRE TROPPO IL PROGRAMMA. ESISTONO TRE METODI VALIDI PER EFFETTUARE L' ACCESSO PER CHIAVE, ESSI SONO:

- 1) RICERCA SEQUENZIALE
- 2) RICERCA TRAMITE UNA TABELLA
- 3) RICERCA CON CODICE HASH

RICERCA SEQUENZIALE:

QUESTO METODO E' IL PIU' LENTO DEI TRE, MA E' QUELLO CHE GARANTISCE UNA MIGLIORE AFFIDABILITA', UNA GESTIONE PIU' LEGGERA ED UNA SEMPLICITA' VERAMENTE UNICA.

IL METODO CONSISTE NELL' IDENTIFICARE L' INFORMAZIONE DA RICERCARE, E POI LEGGERE TUTTI I RECORDS UNO ALLA VOLTA CONFRONTANDO L' INFORMAZIONE CERCATA (CHIAVE) CON I CONTENUTI DEL RECORD O DI UN CAMPO DEL RECORD.

PER ESEMPIO SE SI VUOLE RICERCARE IL NOME 'ROSSI' SI OPERA NELLA SEGUENTE MANIERA:

- SI LEGGE IL RECORD
- SI ESTRAE IL CAMPO CHE CONTIENE IL NOME
- E' 'ROSSI ?' SE NO INCREMENTA IL RECORD E TORNA AL PRIMO PUNTO, SE SI, TROVATO

SI GENERA UN LOOP DAL QUALE SI ESCE SOLO DOPO AVERE TROVATO IL DATO CHE INTERESSA.

SE ALLA FINE DEL FILE IL NOME NON E' STATO RINTRACCIATO E' EVIDENTE CHE NON APPARTIENE AL FILE, IN QUESTO CASO SI VISUALIZZA UN MESSAGGIO DI ERRORE E SI TERMINA LA RICERCA. VEDIAMO IN PRATICA COME REALIZZARE TUTTO CIO' IN BASIC.

```

1 REM *****
2 REM *** QUESTO PROGRAMMA CERCA UN ***
3 REM *** NOME VOLUTO ALL' INTERNO ***
4 REM *** DEL FILE CON IL METODO ***
5 REM *** DELLA LETTURA SEQUENZIALE ***
6 REM *** DI TUTTI I RECORD. ***
7 REM *** PER FILE DI PICCOLE ***
8 REM *** DIMENSIONI E' VELOCE. ***
9 REM *****
10 SCNCLR:GOSUB 600:FLAG=0
15 INPUT "IMMETTI NOME DA CERCARE";N$
18 L=LEN(N$)
20 DOPEN#1,"NOMI.DAT",D0,L55
30 FOR CC=1 TO REC%
40 RECORD#1,(CC)
50 INPUT#1,REC$
60 DATO$=LEFT$(REC$,15)
70 A$=LEFT$(DATO$,L)
80 IF A$=N$ THEN GOSUB 500
90 NEXT CC
100 IF FLAG=0 THEN PRINT"NOME NON ESISTENTE":END
110 END
500 FLAG=1
510 PRINT REC$
520 RETURN
600 DOPEN#2,"NREC",D0,L5
610 RECORD#2,1
620 INPUT#2,LP$
630 REC%=VAL(LP$)
640 DCLOSE#2
650 RETURN

```

COMMENTO

- LA RIGA 10 CANCELLA LO SCHERMO E RIMANDA ALLA SOUBROUTINE ALLOCATA A PARTIRE DALLA RIGA 600, CHE SERVE A LEGGERE DALL' APPOSITO FILE IL NUMERO DI RECORD SCRITTI.

NELLA STESSA LINEA VIENE POSTA A ZERO LA VARIABILE FLAG, QUESTA VARIABILE SEVIRA' AD INDICARE SE ALLA FINE DEL CICLO, ALMENO UN NOME E' STATO TROVATO.

- LA RIGA 15 PROVVEDE A CHIEDERE ALL' UTENTE IL NOME DA CERCARE NEL FILE.

PER NOME SI INTENDE PROPRIO IL NOME E NON NOME E COGNOME INSIEME O ALTRE STRANE RICHIESTE, QUESTO PERCHE' LA RICERCA E' IMPOSTATA SOLO SULLA RICERCA DEL PRIMO CAMPO DI OGNI RECORD DEL FILE.

- LA RIGA 18 METTE IN L LA LUNGHEZZA DEL NOME DA CERCARE, PER MARIO L ASSUME IL VALORE 5.

- LA RIGA 20 APRE IL FILE 'NOMI.DAT' CHE CONTIENE I DATI

- LA RIGA 30 COMINCIA IL CICLO DI LETTURA E CONFRONTO DELL' INTERO FILE CON IL NOME CERCATO

- LA RIGA 40 SERVE A POSIZIONARE LA LETTURA SUL RECORD INDICATO DALLA VARIABILE CC CHE ASSUME TUTTI I VALORI DA 1 A REC% CHE INDICA L' ULTIMO RECORD

- LA RIGA 50 PROVVEDE A LEGGERE FISICAMENTE IL RECORD IN PRECEDENZA PUNTATO E TRASFERIRNE I CONTENUTI NELLA VARIABILE REC\$

- LA RIGA 60 SERVE AD ESTRARRE DALL' INTERO RECORD IL CAMPO CHE CONTIENE IL NOME CHE E' IL PRIMO E SI ESTENDE PER 15 CARATTERI, LA VARIABILE DATO\$ PER IL NOME MARIO ASSUME IL SEGUENTE CONTENUTO:

MARIO*****

DOVE IL CARATTERE * RAPPRESENTA DEI CARATTERI BLANK
(CHR*(32)) CORRISPONDENTI ALLA PRESSIONE DELLA BARRA
SPAZIATRICE.

- LA RIGA 70 E' DI FONDAMENTALE IMPORTANZA PERCHE' PREPARA
IL CAMPO NOME LETTO DAL DISCO PER IL CONFRONTO.

E' CHIARO CHE IL CONFRONTO FRA IL NOME RICHIESTO AD ESEMPIO
'MARIO' E IL CAMPO LETTO 'MARIO*****' NON DARA' NAI
ESITO POSITIVO PERCHE' 'MARIO' NON E' CERTAMENTE UGUALE A
'MARIO*****'.

LA RIGA 70 PROVEDE A RENDERE COMPATIBILI IL CAMPO FORNITO
IN INPUT CON QUELLO LETTO DAL DISCO.

- LA RIGA 80 EFFETTUA IL CONFRONTO VERO E PROPRIO CON I
CAMPI ORMAI RESI COMPATIBILI DALLE ISTRUZIONI PRECEDENTI.

IL RESTO DEL PROGRAMMA E' IDENTICO AD ALTRE ROUTINE GIA'
VISTE IN ESEMPI PRECEDENTI.

RICERCA TRAMITE TABELLA INDICE

QUESTO METODO E' SICURAMENTE PIU' VELOCE DEL PRECEDENTE, MA
PIU' DIFFICILE DA GESTIRE, E PER GESTIONE INTENDO TUTTE LE
ROUTINE CHE SERVONO A MANTENERE ATTIVA UNA TABELLA IN
MEMORIA.

MA COSA SI INTENDE ESATTAMENTE PER TABELLA ?

L' UNICO MODO DI CERCARE LE INFORMAZIONI ALL' INTERNO DI UN
FILE CHE IL '128' SA GESTIRE AUTONOMAMENTE E' QUELLO DELLA
NUMERAZIONE PROGRESSIVA DEI RECORDS, CIOE' TUTTI I RECORDS
SONO NUMERATI SECONDO UN ORDINE CHE IL CALCOLATORE E' IN
GRADO DI RICONOSCERE.

SE AD ESEMPIO DICIAMO AL CALCOLATORE DI ANDARE A LEGGERE IL
RECORD 121, SARA' SICURAMENTE IN GRADO DI COMPIERE TALE
OPERAZIONE, QUESTE SONO LE ISTRUZIONI:

RECORD#1,121

INPUT#1,REC#

MA NON ESISTONO COMANDI PER COMUNICARE AL CALCOLATORE DI ANDARE A LEGGERE IL RECORD RELATIVO A 'MARIO ROSSI'. QUESTO RECORD PUO' ESSERE RINTRACCIATO CON UNA SOLA LETTURA DEL DISCO, SOLO SE CONOSCIAMO L' ESATTA POSIZIONE DEL RECORD RELATIVO A 'MARIO ROSSI'.

MA, ALLORA DOVREMMO TENERE A MENTE TUTTE LE CORRISPONDENZE TRA NOMI E NUMERI DI RECORD, CIOE' UNA TABELLA DEL GENERE:

POSTO	NOME
1	MARIO
2	UGO
3	MARCO
4	PEPPE
5	FRANCO
6	ANNA
7	LORNA
8	ELENA

MA E' SICURAMENTE PIU' EFFICACE FARE CONSERVARE IN MEMORIA AL CALCOLATORE UNA SIMILE TABELLA.

COSI' QUANDO SI CERCHERA IL NOME 'MARIO' SI SAPRA' SUBITO CHE IL RECORD NEL FILE MASTER, DOVE SONO CONTENUTI TUTTI I DATI RELATIVI AL NOME RICHiesto E' IL PRIMO, E COSI' PER TUTTI GLI ALTRI NOMI.

LA RICERCA IN QUESTO MODO SARA VELOCE, PERCHE' LA TABELLA RISIEME IN MEMORIA CENTRALE E LA CONSULTAZIONE NON PREVEDE ACCESSI AL DISCO.

NATURALMENTE LA TABELLA SI PUO' ORGANIZZARE CON I COGNOMI O I NUMERI DI TELEFONO, TUTTO DIPENDE QUALE E' LA CHIAVE DI ACCESSO AL FILE, CIOE' CON QUALE INDIZIO SI VUOLE RICERCARE NEL FILE MASTER.

ECCO ORA GLI ESEMPI, COMINCIAMO A VEDERE COME SI GENERA IN

MEMORIA CENTRALE LA TABELLA, LEGGENDO IL FILE MASTER:

```

1 REM *****
2 REM *** QUESTO PROGRAMMA GENERA ***
3 REM *** UNA TABELLA IN IN$(REC%,2)***
4 REM *** LEGGENDO I CONTENUTI DEL ***
5 REM *** FILE MASTER CHE CONTIENE ***
6 REM *** I DATI. ***
7 REM *** LA TABELLA SI CHIAMA FILE ***
8 REM *** INDICE ***
9 REM *****
10 SCNCLR:GOSUB 900
20 DIM IN$(REC%,2)
30 DOPEN#1,"NOMI.DAT",D0,L55
40 FOR CC=1 TO REC%
50 RECORD#1,(CC)
60 INPUT#1,REC$
70 A$=LEFT$(REC$,15)
80 FOR R=1 TO 15
100 D$=MID$(A$,R,1)
110 IF D$=CHR$(32) THEN 130
120 T1$=T1$+D$
130 NEXT R:D$=""
140 IN$(CC,1)=T1$
150 IN$(CC,2)=STR$(CC):T1$=""
155 NEXT CC
160 PRINT "FINITO":DCLOSE:END
900 DOPEN#2,"NREC",D0,L5
910 RECORD#2,1
920 INPUT#1,REC$
930 REC%=VAL(REC$)
940 DCLOSE#2
950 RETURN

```

COMMENTO

- LA RIGA 10 CANCELLA LO SCHERMO E TRAMITE LA SOUBROUTINE POSTA DALLA RIGA 900, LEGGE IL NUMERO DI RECORD SCRITTO,

PONENDO TALE VALORE NELLA VARIABILE INTERA REC%.

- PRIMA DI PARLARE DELLA RIGA 20 CHE DIMENSIONA LA TABELLA DI CONSULTAZIONE, VEDIAMO COME E' FATTA UNA TABELLA.

```

DIM A$(6,2)      1      2
1 |-----|
  | M A R I O | 1 |
  |-----|
2 |-----|
  | U G O   | 2 |
  |-----|
3 |-----|
  | G I N O | 3 |
  |-----|
4 |-----|
  | P A O L O | 4 |
  |-----|
5 |-----|
  | F R A N C O | 5 |
  |-----|
6 |-----|
  | E L I O   | 6 |
  |-----|

```

IN QUESTA TABELLA AVREMO CHE:

```

A$(1,1)="MARIO"    A$(1,2)="1"
A$(2,1)="UGO"      A$(2,2)="2"
A$(3,1)="GINO"     A$(3,2)="3"
A$(4,1)="PAOLO"    A$(4,2)="4"
A$(5,1)="FRANCO"   A$(5,2)="5"
A$(6,1)="ELIO"     A$(6,2)="6"

```

CHIARO COME FUNZIONA UNA TABELLA ?.

LA RIGA 20, DUNQUE DIMENSIONA QUESTA TABELLA E LO FA CON LA VARIABILE REC% CHE INDICA QUANTI RECORD SONO STATI SCRITTI.

- LA RIGA 30 APRE IL FILE MASTER, PER INTENDERCI, QUELLO CHE

CONTIENE I DATI

- LA RIGA 40 APRE IL CICLO FOR...NEXT ALL' INTERNO DEL QUALE CI SONO PRATICAMENTE TUTTE LE OPERAZIONI DI RIEMPIMENTO E COSTRUZIONE DELLA TABELLA

- LA RIGA 50 POSIZIONA LA LETTURA DEL RECORD INDICATO DALLA VARIABILE CC

- LA RIGA 60 LEGGE I CONTENUTI DEL RECORD PUNTATO IN PRECEDENZA

- LA RIGA 70 SERVE AD ESTRARRE IL CAMPO CHE CONTIENE IL NOME.

QUESTA ESTRAZIONE VIENE FATTA IN QUESTO MODO PERCHE' IL CAMPO INTERESSATO E' IL PRIMO ED E' LUNGO 15 CARATTERI, SE IL CAMPO FOSSE L' ULTIMO SI SAREBBE USATA LA FUNZIONE MID\$(A\$,S,D) DOVE 'S' E 'D' RAPPRESENTANO I CONFINI DEI CAMPI ADIACENTI.

SE, INVECE IL CAMPO INTERESSATO FOSSE STATO L' ULTIMO SI SAREBBE USATA LA FUNZIONE RIGHT\$(A\$,G) DOVE 'G' RAPPRESENTA LA LUNGHEZZA DEL CAMPO DA ESTRARRE.

- DALLA RIGA 90 ALLA RIGA 130 E' POSIZIONATA UNA ROUTINE CHE ELIMINA GLI SPAZI BIANCHI DAL CAMPO, ELIMINANDO L' INCONVENIENTE GIA VISTO DEI CARATTERI BLANK (CHR\$(32)) CHE COMPLETANO IL CAMPO FINO AL MASSIMO DELLA SUA LUNGHEZZA

- NELLE RIGHE 140 E 150 AVVIENE LA VERA ASSEGNAZIONE DELLE VARIABILI PREPARATE IN PRECEDENZA, NELLA TABELLA

- DA 300 IN POI C'E' LA SOLITA ROUTINE CHE LEGGE IL NUMERO DI RECORD GIA' SCRITTI DAL FILE APPOSITO

LA TABELLA COSI' COSTRUITA PUO' ESSERE USATA PER LA CONSULTAZIONE NEL SEGUENTE MODO:

```
1 REM *****
2 REM *** QUESTA ROUTINE SERVE A ***
3 REM *** CONSULTARE UNA TABELLA ***
```

```

4 REM *** COSTRUITA CON IL PROGRAMMA***
5 REM *** PRECEDENTE, RESTITUENDO ***
6 REM *** IL NUMERO DEL RECORD DOVE ***
7 REM *** REPERIRE I DATI NEL FILE ***
8 REM *** MASTER ***
9 REM *****
10 SCNCLR:GOSUB 900:SW=0
20 INPUT "NOME DA CERCARE ";N$:L=LEN(N$)
30 FOR CC=1 TO REC%
40 IF N$=LEFT$(IN$(CC,1) THEN AA=VAL(IN$(CC,2):GOSUB 100
50 NEXT CC
60 IF SW=0 THEN PRINT "IL NOMINATIVO RICHIESTO NON ESISTE"
70 END
100 SW=1:PRINT "TROVATO AL ";AA;" RECORD":RETURN

```

QUESTA ROUTINE NON FA ALTRO CHE CERCARE ALL' INTERNO DELLA TABELLA IL NOME CERCATO (DA NOTARE CHE GRAZIE AL CONFRONTO DELLA RIGA 60, BASTA FORNIRE IN INPUT UN INDIZIO DEL NOME AD ESEMPIO CES AL POSTO DI CESARE, NATURALMENTE SARANNO TROVATI TUTTI QUEI NOMI CHE INIZIANO CON 'CES').

IL CONFRONTO FRA IL NOME CERCATO E GLI ELEMENTI DELLA TABELLA E' MOLTO VELOCE PERCHE' LA TABELLA RISIIDE IN MEMORIA CENTRALE E NON SI DEVE FARE NESSUN ACCESSO AL DISCO PER CONSULTARLA.

NELLA VARIABILE 'AA' VIENE DEPOSITATO IL NUMERO DI RECORD CORRISPONDENTE NEL FILE MASTER, IN PRATICA IL POSTO DOVE REPERIRE IL RESTO DELLE INFORMAZIONI PER IL NOME SELEZIONATO.

QUESTO METODO E' MOLTO EFFICACE MA LA SUA GESTIONE, LABORIOSA, INFATTI BASTI PENSARE AL FATTO CHE LA TABELLA DEVE ESSERE RICOSTRUITA OGNI VOLTA CHE SI EFFETTUA UNA VARIAZIONE NEL FILE MASTER (CANCELLAZIONE O VARIAZIONE).

VOLENDO LA TABELLA GENERATA CON LA ROUTINE OPPORTUNA, A FINE LAVORO PUO' ESSERE CONSERVATA SU DISCO IN UN FILE SEQUENZIALE CHIAMATO 'INDICE' PER COMODITA', COSI' ALLA SESSIONE DI LAVORO SUCCESSIVA NON CI SARA' BISOGNO DI GENERARE LA TABELLA INDICE MA BASTERA' LEGGERLA DAL FILE SEQUENZIALE.

VEDIAMO COME SI PUO' REGISTRARE LA TABELLA INDICE.

```

1 REM *****
2 REM *** QUESTO PROGRAMMA SALVA LA ***
3 REM *** TABELLA CHE COSTITUISCE ***
4 REM *** IL FILE INDICE, SU DISCO ***
5 REM *** IN UN FILE SEQUENZIALE ***
6 REM *** DENOMINATO 'INDICE'. ***
7 REM *** SI SUPPONE CHE IL NUMERO ***
8 REM *** DI RECORD SIA '100'. ***
9 REM *****
10 GONCLR:RECK=100
20 OPEN I,S,I,"INDICE,S,W"
30 FOR CC=1 TO 100
40 FOR CA=1 TO 2
50 PRINT#I,IN$(CC,CA)
60 NEXT CA
70 NEXT CC

```

CON QUESTA SEMPLICE ROUTINE TUTTA LA TABELLA VIENE RIVERSATA SU DISCO.

CHIAMANDO LA DIRECTORY SI DEVE TROVARE LA SEGUENTE INFORMAZIONE:

```

      XXX      "INDICE"      SEQ

```

DOVE XXX RAPPRESENTANO I BLOCCHI OCCUPATI SUL DISCO DAL FILE SEQUENZIALE.

NOTARE COME I DUE CICLI FOR...NEXT, SIANO CORRETTAMENTE NIDIFICATI, PRIMA VIENE CHIUSO IL PIU' INTERNO E POI L' ESTERNO.

NATURALMENTE QUANDO IL PROGRAMMA PRINCIPALE (QUELLO CHE GESTISCE L' ARCHIVIO), SARA' LANCIATO, BISOGNERA' RILEGGERE LA TABELLA CONSERVATA SU DISCO, VEDIAMO COME:

```

1 REM *****
2 REM *** QUESTA ROUTINE LEGGE LA ***
3 REM *** TABELLA DAL FILE 'INDICE' ***
4 REM *** E RICOSTRUISCE IN MEMORIA ***
5 REM *** IL FILE INDICE. ***
6 REM *** SI SUPPONE CHE I RECORDS ***

```

```

7 REM *** SIANO '100'.          ***
8 REM ***                      ***
9 REM *****
10 SCNCLR:REC%=100:DIM IN$(REC%,2)
20 OPEN 1,8,0,"INDICE,S,R"
30 FOR CC=1 TO 100
40 FOR CA=1 TO 2
50 INPUT#1,IN$(CC,CA)
60 NEXT CA
70 NEXT CC

```

LA TABELLA E', ORA, COSTITUITA IN MEMORIA ED E' PRONTA PER ESSERE CONSULTATA.

METODO HASH:

QUESTO E' UN METODO ESTREMAMENTE VELOCE, IL PIU' VELOCE, MA RICCO DI INCONVENIENTI E PROBLEMI, I QUALI UNA VOLTA RISOLTI, GARANTISCONO UNA GESTIONE DELLE INFORMAZIONI ULTRAVELOCE.

IL METODO CONSISTE NEL RICAVARE IL POSTO DEL RECORD NEL FILE MASTER DALLA CHIAVE STESSA, SENZA NESSUN CONFRONTO, SENZA INDICI.

NATURALMENTE, LA GESTIONE DI QUESTO METODO DEVE ESSERE PREVENTIVATA, E MESSA IN OPERA FINO DALLA SCRITTURA DELLE INFORMAZIONI SUL FILE MASTER.

IL METODO HASH CONSISTE NELLA MESSA A PUNTO DI UN ALGORITMO MATEMATICO CHE CONSENTA SIA IN FASE DI SCRITTURA CHE IN FASE DI LETTURA, DI ESTRAPOLARE LE INFORMAZIONI DALLA CHIAVE DI LETTURA.

ESAMINIAMO UN SEMPLICE (E POCO VALIDO) ALGORITMO HASH:

```

1 REM *****
2 REM *** QUESTA ROUTINE RAPPRESENTA***
3 REM *** UN SEMPLICE, MA POCO      ***
4 REM *** VALIDO, ESEMPIO DI COME   ***
5 REM *** SI PUO' ESTRAPOLARE IL    ***
6 REM *** NUMERO DEL RECORD DOVE    ***

```

```

7 REM *** LEGGERE O SCRIVERE DATI ***
8 REM *** DALLA CHIAVE DI ACCESSO ***
9 REM ****
10 SCNCLR
20 INPUT "NOME ";N$
30 A=ASC(LEFT$(N$,1))
40 B=ASC(RIGHT$(N$,1))
50 C=INT(A+B/10)
60 IF C<1 THEN PRINT "RECORD NON VALIDO":GOTO 10
70 PRINT C;" E' IL RECORD DOVE I DATI SONO SCRITTI O LETTI"

```

DALLA ROUTINE SI EVINCE CHE, IL NUMERO DI RECORD VIENE RICAVATO ESTRAENDO IL PRIMO (RIGA 30) E L' ULTIMO (RIGA 40) CARATTERE DEL NOME (N\$) E SOMMANDO I RISPETTIVI CODICI ASCII E DIVIDENDO IL TUTTO PER 10.

SEGUIAMO IL FUNZIONAMENTO DEL PROGRAMMA RIGA PER RIGA: SUPPONIAMO CHE IL NOME INERESSATO SIA 'VITTORIO', LA PRIMA LETTERA E' LA 'A' CON CODICE ASCII 65, L' ULTIMA LETTERA E' LA 'O' CODICE ASCII 79.

LA SOMMA DEI DUE CODICI ASCII E' 144 CHE DIVISO 10 FA 14.4 IL CUI INTERO E' 14, QUESTO E' IL RECORD DOVE SCRIVERE O LEGGERE LE INFORMAZIONI RELATIVE A 'VITTORIO'.

VELOCE NO ?

LA FUNZIONALITA' DEL METODO DIPENDE ESSENZIALMENTE DALL' ALGORITMO HASH USATO, NEL NOSTRO CASO, INFATTI, VENGONO GENERATI GLI STESSI RECORD CON MOLTI NOMI COME:

- VITO
- VINCENZO
- VENANZIO

E CON TUTTI QUEI NOMI CHE INIZIANO PER 'V' E FINISCONO PER 'O'.

QUESTI INCONVENIENTI SI CHIAMANO 'COLLISIONI'.

UN' ALTRO INCONVENIENTE DEL METODO HASH E' CHE ALCUNI RECORD NON VENGONO MAI SCRITTI, PERCHE' IL LORO NUMERO NON VIENE MAI FUORI DALL' ALGORITMO.

CONCLUDENDO, METODO HASH SI' MA A PATTO CHE L' ALGORITMO SIA BUONO, TENTEREMO QUALCHE ESEMPIO.

SONO ALGORITMI DECENTI:

1) LETTURA DELLA PRIMA LETTERA DEL NOME DELL' ULTIMA E DI QUELLA DI MEZZO, SOMMA DEI RISPETTIVI CODICI ASCII E DIVISIONE PER 10 DEL RISULTATO.

2) LETTURA DI TUTTE LE LETTERE DEL NOME SOMMA DEI RISPETTIVI CODICI ASCII E DIVISIONE DEL RISULTATO PER 1 SE QUESTO E' >10 OPPURE PER 10 SE QUESTO E' MAGGIORE DI 100.

QUESTO METODO E' MOLTO BELLO E NUOVO, INTRODOTTO DA POCO NELLA PRATICA DELLA PROGRAMMAZIONE, QUINDI METTETEVI AL LAVORO PER TROVARE ALTRI ALTRI ALGORITMI SUFFICIENTEMENTE PRECISI.

R I O R D I N O D E I D A T I

RIORDINO DEI DATI

UNO DEI TANTI PROBLEMI DA RISOLVERE QUANDO SI PROGETTA UN PROGRAMMA DI ARCHIVIAZIONE DATI, E' QUELLO DI ORDINARE LE INFORMAZIONI SECONDO TALUNI CRITERI.

AD ESEMPIO, RISULTA MOLTO COMODO PER ALCUNI TIPI DI RICERCA CHE I DATI SIANO ORDINATI ALFABETICAMENTE.

I CRITERI DI ORDINAMENTO CHE IN GENERE SI SEGUONO IN PROGRAMMI DI QUESTO TIPO SONO I SEGUENTI.

- ORDINE ALFABETICO ASCENDENTE
- ORDINE ALFABETICO DISCENDENTE
- ORDINE PER LETTERA
- ORDINE PER CATEGORIE

PER QUANTO RIGUARDA LE VARIABILI LETTERALI, E:

- ORDINE NUMERICO CRESCENTE
- ORDINE NUMERICO DECRESCENTE
- ORDINE PER INSIEMI

PER QUEL CHE RIGUARDA LE VARIABILI NUMERICHE.

COME DICEVO, E' MOLTO UTILE AVERE UN FILE INDICE (VEDI TABELLA INDICE) ORDINATO IN SEQUENZE ALFABETICHE, AI FINI DI UNA RICERCA ULTERIORMENTE MIGLIORATA DA UN PUNTO DI VISTA VELOCITA'.

UN TIPO DI GESTIONE MOLTO EFFICACE E' QUELLA CHE SFRUTTA IL RIORDINO A GRUPPI DI LETTERE, DEI DATI , PER POI, IN FASE DI RICERCA, ANZICHE' CONSULTARE TUTTO IL FILE INDICE DAL PRIMO CAMPO, SI PROCEDE A FARE I CONFRONTI DALLA LETTERA CHE INTERESSA, LA QUALE VIENE ESTRATTA DAL CAMPO CHE SI DA IN INPUT PER LA RICERCA

SE AD ESEMPIO, SI DA' IN INPUT DA CERCARE IL NOME UGO, SI ESTRAE LA PRIMA LETTERA CHE E' LA 'U' E SI VA A CERCARE IN QUELLA PARTE DEL FILE INDICE DOVE RISIEDONO TUTTI I NOMI CHE COMINCIANO PER 'U'.

E' CHIARO CHE IN QUESTO MODO SI RIDUCONO DRASTICAMENTE IL NUMERO DI CONFRONTI DA EFFETTUARE E DI CONSEGUENZA SI RIDUCE IL TEMPO NECESSARIO ALLA RICERCA.

VOLENDO REALIZZARE IN PRATICA TALE PROCEDURA SI OPERA NELLA SEGUENTE MANIERA:

- SI COSTITUISCE IN MEMORIA LA TABELLA INDICE , SI METTONO IN UN VETTORE (MATRICE AD UNA SOLA DIMENSIONE) I CONTENUTI DEL CAMPO DA ORDINARE (NOMI, TELEFONO, ECC.ECC)

- TRAMITE UN PROGRAMMA SPECIALE DETTO 'SORT' SI METTONO IN ORDINE ALFABETICO I CONTENUTI DEL VETTORE

- SI COSTITUISCE UNA NUOVA TABELLA (SOTTOTABELLA DEL FILE INDICE) IN CUI SI MEMORIZZANO I LIMITI DOVE INIZIA E FINISCE IL CAMPO DI UNA LETTERA (ES. 'A' DAL 1 AL 20 POSTO 'B' DAL 21 AL 45 POSTO ECC. ECC), E IL GIOCO E' FATTO. QUANDO SI ANDRA' A FARE UNA RICERCA SI SAPRA' SUBITO IN QUALE INERVALLO DEL FILE CERCARE LE INFORMAZIONI DESIDERATE.

VEDIAMO COME APPARE UN VETTORE DISORDINATO.

MARIO	---	RECORD 1
UGO	---	RECORD 2
FRANCO	---	RECORD 3
CESARE	---	RECORD 4
		ECC. ECC.

NOTATE I RECORD CORRISPONDENTI A FIANCO DELL' INFORMAZIONE MEMORIZZATA NEL VETTORE.

ED ECCO COME APPARE IL VETTORE DOPO ESSERE STATO TRATTATO CON UN APPOSITO PROGRAMMA DI SORT:

CESARE	---	RECORD 4
FRANCO	---	RECORD 3
MARIO	---	RECORD 1
UGO	---	RECORD 2
		ECC. ECC.

DA NOTARE CHE I NOMI SONO ORDINATI ALFABETICAMENTE, E SI PORTANO DIETRO IL RECORD, COSI' SARA' SEMPRE POSSIBILE REPERIRE LE INFORMAZIONI, ALL' INTERNO DEL FILE MASTER. E' DI FONDAMENTALE IMPORTANZA CHE DOPO IL RIORDINO CI SIA SEMPRE UNA CORRISPONDENZA FRA CAMPO CHIAVE (NEL NOSTRO CASO IL NOME) E IL RECORD CORRISPONDENTE SUL FILE MASTER, E' EVIDENTE CHE SE VIENE PERSA QUESTA CORRISPONDENZA NON SARA' POSSIBILE REPERIRE I DATI RELATIVI AD UNA CHIAVE DI ACCESSO. QUESTO PROCEDIMENTO DI RIORDINO DEL FILE INDICE EQUIVALE AL RIORDINO ALFABETICO DEL FILE MASTER, MA CON UN NOTEVOLE RISPARMIO, SIA DI TEMPO DI ESECUZIONE, CHE DI CODICE BASIC. OSSERVIAMO UN SEMPLICE PROGRAMMA DI SORT E CERCHIAMO DI CAPIRE IL FUNZIONAMENTO.

METODO DI ORDINAMENTO RIPPLE

QUESTO METODO CONSISTE NEL CONFRONTARE A DUE A DUE TUTTI GLI ELEMENTI DI UNA TABELLA DISORDINATA, E SE NECESSARIO SCAMBIARE TRA LORO, GLI ELEMENTI CHE DAL CONFRONTO RISULTASSERO ESSERE NON DISPOSTI NELL' ORDINE DESIDERATO. OGNI VOLTA CHE VIENE EFFETTUATO UNO SCAMBIO (SWAP) SI RICOMINCIA DAL PRIMO ELEMENTO DELLA TABELLA. DOPO N PASSI DI ORDINAMENTO SI OTTERRA' LA TABELLA ORDINATA ALFABETICAMENTE, VEDIAMO IL CODICE:

```
1 REM *****
2 REM *** PROGRAMMA DI SORT CHE      ***
3 REM *** USA IL METODO RIPPLE      ***
4 REM *** O DEI CONFRONTI          ***
```

```

5 REM *** SUCCESSIVI. ***
6 REM *** IL VETTORE DA ORDINARE E' ***
7 REM *** A$(N). ***
8 REM *** ***
9 REM *****
10 SCNCLR: DIM A$(50): C=1
20 INPUT "NOME (PREMI E PER FINE)"; A$(C)
30 IF C=50 OR A$(C)="E" THEN 50
40 C=C+1: GOTO 20
50 PRINT "ECCO I NOMINATIVI"
60 SLEEP 5
70 FOR D=1 TO C
80 PRINT A$(D)
90 NEXT D
100 PRINT "STO RIORDINANDO"
110 K=C
120 INV=0
130 FOR I=1 TO K-1
140 IF LEFT$(A$(I+1),3)<LEFT$(A$(I),3) THEN GOSUB 500
150 NEXT I
160 IF INV<>0 THEN K=K-1: GOTO 120
170 SCNCLR
180 PRINT "ECCO I NOMI ORDINATI "
190 SLEEP 5
200 FOR D=1 TO C
210 PRINT A$(D)
220 NEXT D
230 END
500 REM ESEGUE SCAMBIO
510 T$=A$(I): U$=A$(I+1)
520 A$(I+1)=T$: A$(I)=U$
530 INV=1: RETURN

```

ECCO COME FUNZIONA IL PROGRAMMA:

TABELLA INIZIALE

VITTORIO	1 CONFRONTO :
<hr/>	
CESARE	VITTORIO ' E ' ORDINATO
<hr/>	RISPETTO A CESARE ?
	RISPOSTA : NO
ALDO	ALLORA SCAMBIO
<hr/>	
ZINA	
<hr/>	

DOPO IL PRIMO SCAMBIO

CESARE	2 CONFRONTO
<hr/>	
VITTORIO	' CESARE E ' ORDINATO
<hr/>	RISPETTO A VITTORIO ?
	RISPOSTA : SI
	NESSUNO SCAMBIO
ALDO	
<hr/>	3 CONFRONTO
ZINA	' VITTORIO E ' ORDINATO
<hr/>	RISPETTO A ALDO ?
	RISPOSTA : NO
	ALLORA SCAMBIO

DOPO IL SECONDO SCAMBIO

	4 CONFRONTO
CESARE	
	CESARE E' ORDINATO
	RISPETTO AD ALDO ?
ALDO	RISPOSTA : NO
	ALLORA SCAMBIO

VITTORIO

ZINA

DOPO IL TERZO SCAMBIO

	IL VETTORE, COME SI PUO'
ALDO	OSSERVARE, E' ORDINATO
	ALFABETICAMENTE
	IL PROGRAMMA SEGUE
CESARE	QUESTO PROCEDIMENTO
	CHE IN SINTESI
	PRENDE IL NOME
VITTORIO	DI METODO 'RIPPLE'

ZINA

COMMENTO AL PROGRAMMA

- LA RIGA 10 CANCELLA LO SCHERMO E DIMENSIONA IL VETTORE A\$(N) CHE DOVRA' CONTENERE I NOMI DA ORDINARE PER UN MASSIMO DI 50 (POSSONO ESSERE ANCHE DI PIU').
NELLO STESSO MOMENTO VIENE SETTATA AD UNO LA VARIABILE 'C', CHE SERVIRA' COME CONTATORE DURANTE LA FASE DI INPUT.

- LA RIGA 20 ACCETTA UN NOME E LO DEPOSITA NEL VETTORE CON INDICE 'C', QUESTO PROCESSO SI RIPETE UN MASSIMO DI 50 VOLTE, O COMUNQUE QUANDO VIENE DATA IN INPUT LA LETTERA 'E'.

- LA RIGA 30 SI OCCUPA DI CONTROLLARE SE SIAMO ARRIVATI AL 50 ESIMO NOME O SE E' STATO PREMUTO IL TASTO 'E' DA NOTARE L' USO DELL' OPERATORE LOGICO 'OR' IL QUALE FA SI' CHE AL VERIFICARSI DI UNA SOLA DELLE DUE CONDIZIONI SPECIFICATE NEL 'IF', SI SALTI ALLA ROUTINE DI ORDINAMENTO RIPLE

- LA RIGA 40 INCREMENTA IL CONTATORE 'C' E DIRAMA IL PROGRAMMA ALLA RIGA 20 DOVE SI RIPETE L' INPUT

- LA RIGA 50 STAMPA UN MESSAGGIO

LA RIGA 60 USA UN COMANDO DEL '128' CHE SERVE A GENERARE RITARDI L' ISTRUZIONE 'SLEEP N' DOVE N RAPPRESENTA IL NUMERO DI SECONDI DI STASI DEL PROGRAMMA.

- LE RIGHE 70-90 SERVONO A STAMPARE TUTTI NOMINATIVI CONTENUTI NEL VETTORE A\$(N), TRAMITE UN CICLO FOR...NEXT

- LA RIGA 100 STAMPA UN MESSAGGIO

- DALLA RIGA 110 ALLA RIGA 160 E' LOCATA LA ROUTINE DI SORT

CHE FUNZIONA COME SI E' GIA' VISTO.

I METODI PER ORDINARE SONO DIVERSI, A TITOLO DI ESEMPIO ECCO UNA ROUTINE DI SORT RIDOTTA ALL' ESSENZIALE, CHE FUNZIONA MOLTO BENE, SOLTANTO UN PO' LENTAMENTE:

```

1 REM *****
2 REM *** ROUTINE DI SORT PER      ***
3 REM *** LETTERE , METODO        ***
4 REM *** BUBBLE-SORT              ***
5 REM *** VETTORE DA ORDINARE A$(N) ***
6 REM *** CON N MASSIMO 50.        ***
7 REM ***                          ***
8 REM ***                          ***
9 REM *****
10 FOR I=1 TO N-1
11 FOR J=I+1 TO N
12 IF LEFT$(A$(J),3)<LEFT$(A$(I),3) THEN GOSUB 100
13 NEXT J
14 NEXT I
15 END
100 T$=A$(J):U$=A$(I)
110 A$(I)=T$:A$(J)=U$
120 RETURN

```

IL PROGRAMMA SI COMMENTA DA SE' E PUO' FUNZIONARE SIA PER LETTERE CHE PER NUMERI, IN QUEST' ULTIMO CASO BISOGNA SOSTITUIRE IL VETTORE LETTERALE CON UNO NUMERICO E LE VARIABILI LETTERALI CON VARIABILI NUMERICHE. IL PROGRAMMA DIVENTA COSI':

```

1 REM *****
2 REM *** ROUTINE DI SORT PER      ***

```

```

3 REM *** NUMERI METODO BUBBLE-SORT ***
4 REM *** VETTORE DA ORDINARE A(N) ***
5 REM *** DOVE N MASSIMO 50. ***
6 REM *** ***
7 REM *** ***
8 REM *** ***
9 REM *****
10 FOR I=1 TO N-1
20 FOR J=I+1 TO N
30 IF A(J)<A(I) THEN GOSUB 100
40 NEXT J
50 NEXT I
60 END
100 T=A(J):U=A(I)
110 A(I)=T:A(J)=U
120 RETURN

```

DA TENERE PRESENTE CHE PER IL RIORDINO DI VETTORI MOLTO AMPI, IL TEMPO DI ESECUZIONE DIVENTA CONSIDEREVOLE, ANCHE PER LA PRESENZA DEL FENOMENO 'GARBAGE COLLECTION', LETTERALMENTE RACCOLTA DI SPAZZATURA.

QUESTO FENOMENO CONSISTE IN UN RIEMPIMENTO DELLA MEMORIA DA PARTE DI VARIABILI INUTILI DURANTE LA FASE DI SCAMBIO.

UN RIMEDIO A QUESTO INCONVENIENTE, CONSISTE NEL RICHIAMO PERIODICO DELLA FUNZIONE 'FRE(0)', INFATTI, IL '128' PER IL CALCOLO DELLA MEMORIA LIBERA AL MOMENTO, RIORGANIZZA TUTTE LE VARIABILI ATTIVE, ELIMINANDO QUELLE NON IN USO.

LA RIGA DA INSERIRE E' LA SEGUENTE:

```
X=FRE(0)
```

LA SI PUO' INSERIRE DENTRO LA SOUBROUTINE DI SCAMBIO COSI' OGNI VOLTA CHE NE VIENE EFFETTUATO UNO, SI PULISCE LA MEMORIA DALLE VARIABILI INUTILI.

PER IL RIORDINO DI GROSSE MOLI DI DATI CONVIENE USARE UN ALGORITMO DI SORT PIU' COMPLESSO MA MOLTO VELOCE, E CHE PRENDE IL NOME DEI PROGRAMMATORI CHE LO HANNO MESSO A PUNTO PER PRIMI, MI RIFERISCO AL METODO SHELL-METZNER.

QUESTO SORT E' MOLTO VELOCE PERCHE' ESEGUE POCHI CONFRONTI E FRA ELEMENTI LONTANI, E NON ADIACENTI, COME I METODI ESAMINATI IN PRECEDENZA.

```

1 REM *****
2 REM *** ORDINAMENTO DI UN VETTORE ***
3 REM *** NUMERICO A(N) CON IL      ***
4 REM *** METODO SHELL-METZNER.      ***
5 REM *** IL METODO SI PUO'          ***
6 REM *** APPLICARE ANCHE A VETTORI ***
7 REM *** ALFANUMERICI.              ***
8 REM ***                             ***
9 REM *****
100 SC=N
110 SC=INT(SC/2):IF SC<1 THEN STOP
120 J=1:K=N-SC
130 I=J
140 M=I+SC
150 IF A(I)<A(M) THEN 180
160 GOSUB 500
170 I=I-SC:IF I<1 THEN 180 ELSE GOTO 140
180 J=J+1:IF J>K THEN 110 ELSE GOTO 130

```

DA NOTARE L' USO DELLA FUNZIONE 'ELSE' CHE CONSENTE UN ACCOSTAMENTO AI PRINCIPI DELLA PROGRAMMAZIONE STRUTTURATA, PROPRIA DI LINGUAGGI PIU' EVOLUTI COME IL PASCAL E IL C. CREDO CHE OGNI COMMENTO SIA SUPERFLUO, IL PROGRAMMA FUNZIONA SEMPRE PER CONFRONTI, SOLO CHE QUESTI SONO RIDOTTI AL MINIMO, PER VELOCIZZARE L' ESECUZIONE. CON QUESTA PANORAMICA SUI PROGRAMMI DI SORT CONCLUDO L' ARGOMENTO LASCIANDO AL LETTORE IL COMPITO DI METTERE IN PRATICA QUESTI SUGGERIMENTI.

A P P E N D I C I

M A P P E D I M E M O R I A

MAPPE DI MEMORIA

IL COMMODORE '128' E' USCITO DA QUASI 6 MESI ED ANCORA NON ESISTE DOCUMENTAZIONE PER QUANTO RIGUARDA MAPPE DI MEMORIA, RIFERIMENTI AL SISTEMA OPERATIVO, TRUCCHI, DETTAGLI SULLE NUOVE ISTRUZIONI.

INSOMMA CI SI RENDE CONTO DI AVERE FRA LE MANI UNA MACCHINA CON DELLE POTENZIALITA' ENORMI, MA NON SI RIESCE A VEDERE NESSUN RISULTATO CONCRETO.

IL FATTO DI INTRODURRE IN APPENDICE ALCUNE NOTIZIE CHE CON I FILES HANNO POCO A CHE VEDERE, NON VUOL ESSERE UNA TROVATA PUBBLICITARIA, MA SOLO UN CHIAMIAMOLO 'REGALO' FATTO AI LETTORI, VISTA LA DIFFICOLTA CHE SI INCONTRA A REPERIRE TALI INFORMAZIONI.

LE MAPPE PRESENTATE IN QUESTO LIBRO, CREDO COME NOVITA' ASSOLUTA, SONO IN LINGUA ORIGINALE (INGLESE) E NON SONO STATE VOLUTAMENTE TRADOTTE, PROPRIO PER CONSERVARNE 'L' ORIGINALITA'.

SONO DETTAGLiate IN ALCUNI TRATTI MENTRE SONO SOMMARIE IN ALTRI, MA COMUNQUE RAPPRESENTANO UN VALIDO STRUMENTO DI LAVORO PER IL PROGRAMMATORE PIU' EVOLUTO.

LE MAPPE SONO DIVISE IN DUE SEZIONI, LA PRIMA DESCRIVE LA P.A.M. CON PARTICOLARE ATTENZIONE RIVOLTA AL BASIC E ALLE LOCAZIONE AD ESSO DEDICATE; LA SECONDA DEDICATA ALLE ROUTINE DI INPUT/OUTPUT RESIDENTI IN R.O.M., ROUTINE VERAMENTE INTERESSANTI, CHE POSSONO ESSERE SFRUTTATE DA SOLE CON ECCELLENTI RISULTATI.

CREDO CHE CONOSCERE QUESTE INFORMAZIONI POSSA ESSERE UTILE PER USARE AL MEGLIO LA MACCHINA, ED IN ATTESA DI NOTIZIE PIU' DETTAGLiate DALLA FONTE UFFICIALE, CONTENTIAMOCI.

```

!
!
! -----
! C-128 RAM MAP
! -----
!
!

```

```

!0002-0009 BASIC Zero Page
! 0002-0009 Temp. Storage-BANK; <PC;
! >PC; ST; A; X; Y; P from CPU
! or code for switch to 64
! 0009 CHARAC ;SEARCH CHARACTER
! 000A ENDCHR ;FLAG-SCAN FOR QUOTE
! 000B TRMPOS ;SCR COL / LAST TAB
! 000C VERCHK ;FLAG- 0=LOAD 1=VER
! 000D COUNT ;INPUT BUF.PTR/ # OF
! SUBSCRIPTS
! 000E DIMFLAG ;FLAG- DFLT ARR DIM
! 000F VALTYP ;DATA TYPE-
! $FF=STR $00=NUM
! 0010 INTFLAG ;DATA TYPE-
! $00=FL.PT. $80=INT
! 0011 GARBFL ;FLAG- DATA SCAN /
! LIST QUOTE/GARB.COL
! 0011 DORES
! 0012 SUBFLG ;FLAG-SUBSCRIPT REF.
! / USER FUNC. CALL
! 0013 INPFLG ;FLAG- $00=INPUT;
! $40=GET; $38=READ
! 0014 DOMASK
! 0014 TANSGN ;FLAG- TAN SIGN /
! COMPARISON RESULT
! 0015 ;CHANNEL POKER
! 0016 LINNUM ;TEMP INTEGER VALUE
! 0018 TEMPOT ;PNTR-TEMP STR STACK
! 0019 LASTPT ;LAST TEMP STR ADDR
! 0019 TEMPST ;STACK FOR TEMP STRS
! 0024 INDEX ;UTIL. POINTER AREA
! 0028 RESHO ;FL.PT. PROD OF MULT
! 0029 RESMOH
! 002A ADDEND

```

```

! 002A  RESMO
! 002B  RESLO
! 002D-003C  BASIC memory vectors
! 002D      Start of BASIC Pgm RAM(0)
! 002F      Start of Var.$0400 RAM(1)
! 0031      Start of Arrays  RAM(1)
! 0033      End of Arrays + 1  RAM(1)
! 0035      Bottom of Strings RAM(1)
! 0037      Lowest String vctr RAM(1)
! 0039      Top of Strings  RAM(1)
! 003B  CURLIN  ;CURR. BASIC LINE #
! 003D  TXTPTR  ;PNTR TO BASIC TXT
! 003F  FORM    ;USED BY PRINT USING
! 003F  FNDPNT  ;POINTER TO ITEM
!           FOUND BY SEARCH
! 0041  DATLIN  ;CURRENT DATA LINE#
! 0043  DATAPTR ;CURRENT DATA ADDR
! 0045  INPPTR  ;VECTOR-INPUT ROUT.
! 0047  VARNAM  ;CURR BASIC VAR NAME
! 0049  FDECPT
! 0049  VARPNT  ;POINTER- CURRENT
!           BASIC VARIABLE DATA
! 004B  LSTPNT
! 004B  ANDMSK
! 004B  FORPNT  ;POINTER- INDEX
!           VARIABLE FOR/NEXT
! 004C  EORMSK  =FORPNT+1
! 004D  VARTXT
! 004D  OPPTR
! 004F  OPFmask
! 0050  GRBPNT
! 0050  TEMPF3
! 0050  DEFPNT
! 0052  DSCPNT
! 0055  HELPER
! 0056  JMPER
! 0058  OLDOV
! 0059  TEMPF1
! PTARG1  =TEMPF1 ;MULTIPLY
!           DEFINED FOR INSTR

```

```

!          PTARG2      =TEMPF1+2
!          STR1        =TEMPF1+4
!          STR2        =TEMPF1+7
!          POSITN      =TEMPF1+10
!          MATCH       =TEMPF1+11
! 005A  ARYPNT
! 005A  HIGHDS
! 005C  HIGHTR
! 005E  TEMPF2
! 005F  DECCNT
! 0061  GRBTOP
! 0061  DPTFLG
! 0061  LOWTR
! 0062  EXPSGN
!          TENEXP      =DECCNT+1
! 0063  FAC
! 0063  DSCTMP
! 0063  LEFTFLAG      ;PAINT-LEFT FLAG.
! 0063  FACEXP        ;FAC#1 EXPONET
! 0064  RIGHTFLAG     ;PAINT-RIGHT FLAG
! 0065  FACMOH
! 0066  INDICE
! 0066  FACMOH
! 0067  FACMO
! 0068  FACSGN        ;POINTER-
!                      SERIES-EVAL. CONST.
! 0069  DEGREE
! 0069  SGNFLG        ;POINTER-
!                      SERIES-EVAL. CONST.
! 006A  ARGEXP        ;FAC#2 EXPONET
! 006B  ARGHO         ;FAC#2 MANTISSA
! 006C  ARGMOH
! 006D  ARGMO
! 006E  ARGLO
! 006F  ARGSGN        ;FAC#2 SIGN
! 0070  STRNG1
! 0070  ARISGN        ;SIGN COMPARISON
!                      RESOUL:FAC#1 VS #2
! 0071  FACOV         ;FAC#1 LOW-ORDER
!                      (ROUNDING)

```

```

! 0072  STRNG2
! 0072  POLYPT
! 0072  CURTOL
! 0072  FBUFPT      ;PNTR; CASS. BUFFER
! 0074  AUTINC      ;INC. VAL FOR AUTO
!                   (0=OFF)
! 0076  MVDFLG      ;FLAG IF 10K HIRES
! 0077  NOZE        ;USING LEAD ZERO CT
! 0077  SPRNUM      ;MOVSPR&SPRITE TEMP
! 0077  KEYNUM
! 0078  HULP        ;COUNTER
! 0078  KEYSIZ
! 0079  SYNTMP      ;TEMP FOR IND LOADS
! 007A  DSDEC       ;DESCRIPTOR FOR DS#
! 007D  TOS         ;TOP / RUNTIME STACK
! 007F  RUNMOD      ;FLAG-RUN/DIR. MODE
! 0080  PARSTS      ;DOS PARSER ST WORD
! 0080  POINT       ;USING PNTR/DEC.PT.
! 0081  PARSTX
! 0082  QLOSTK      ;GRAPHIC ZP STORAGE
! 0083  COLSEL      ;CURRENT COLOR
! 0084  MULTICOLOR1
! 0085  MULTICOLOR2
! 0086  FOREGROUND
! 0087  SCALEX      ;SCALE FACTOR IN X
! 0089  SCALEY      ;SCALE FACTOR IN Y
! 008B  STOPNB      ;STOP PAINT IF NOT
!                   B.G./SAME COLOR
! 008C  GRAPNT
! 008E  VTEMP1
! 008F  VTEMP2
!0090-00FF  KERNAL Zero Page
! 0090  STATUS      ;I/O OPER STAT BYTE
! 0091  Flag; STOP Key
! 0092  SVXT        ;TAPE TEMPORARY
! 0093  VERCK       ;LOAD OR VERIFY FLAG
! 0094  C3P0        ;SER. BUFFERED CHAR
! 0095  BSOUR       ;CHAR BUF FOR SERIAL
! 0096  SYNO        ;CASS SYNC BUFFER
! 0097  XSAV        ;TEMP FOR BASIN

```

```

! 0098          No. of open files
! 0099          Default Input Device(0)
! 009A          Default Output Device(3)
! 009B  PRTY      ;CASSETTE PARITY
! 009C  DPSW      ;CASS.DIPOLE SWITCH
! 009D  MSGFLG    ;OS MESSAGE FLAG
! 009E  PTR1      ;CASS. ERROR PASS 1
! 009E  T1        ;TEMP 1
! 009F  PTR2      ;CASS. ERROR PASS 2
! 009F  T2        ;TEMP 2
! 00A0-00A2      Jiffy Clock
! 00A3  R2D2      ;SERIAL BUSS USAGE
! 00A3  PCNTR     ;CASSETTE STUFF
! 00A4  BSOUR1    ;TEMP/SERIAL ROUTINE
! 00A4  FIRT      ;TEMP/SERIAL ROUTINE
! 00A5  COUNT     ;TEMP/SERIAL ROUTINE
! 00A5  CNTDN     ;CASS.SYNC COUNTDOWN
! 00A6  BUFPT     ;CASS.BUFFER POINTER
! 00A7  INBIT     ;RS232 RCVR IN. BIT
! 00A7  SHCN1     ;CASS. SHORT COUNT
! 00A8  BITC1     ;RS232 RCVR BIT CNT
! 00A8  RER       ;CASSETTE READ ERROR
! 00A9  RINONE    ;RS-232 RCVR FL FOR
!                START BIT CHECK
! 00A9  REZ       ;CASS READING ZEROS
! 00AA  RIDATA    ;RS232 RCVR BYTE BUF
! 00AA  RDFLG     ;CASSETTE READ MODE
! 00AB  RIPRTY    ;RS232 REVR PARITY
! 00AB  SHCNH     ;CASS. SHORT COUNT
! 00AC  SAL       ;POINTER - TAPE
!                BUF./SCREEN SCROLL
! 00AD  SAH       ;
! 00AE  EAL       ;TAPE END ADDRESSES/
!                END OF PROGRAM
! 00AF  EAH       ;
! 00B0  CMP0      ;TAPE TIME CONST.
! 00B1  TEMP      ;
! 00B2-00B3      Tape Buffer vector
! 00B4  BITTS     ;RS-232 TRNS BIT CNT
! 00B4  SNSWL     ;

```

```

! 00B5 NXTBIT ;RS232 TRNS NEXT BIT
! 00B5 DIFF
! 00B6 RODATA ;RS232 TRNS BYTE BUF
! 00B6 PRP ;
! 00B7 LEN of current filename
! 00B8 Current File No.
! 00B9 Current Sec. Addr.
! 00BA Current Device No.
! 00BB-00BC Filename address
! 00BD ROPRTY ;RS232 TRNS PARITY
! 00BD OCHAR ;
! 00BE FSBLK ;CASS READ BLOCK CNT
! 00BF DRIVE
! 00BF MYCH ;SERIAL WORD BUFFER
! 00C0 CASL ;CASSETTE
! MANUAL/CONTROLLED SWITCH (IRQ)
! 00C1 TRACK
! 00C1 STAL ;IO START ADDR (<LB)
! 00C2 STAH ;IO START ADDR (>HB)
! 00C3-4 MEMUSS ;CASS LOAD TEMPS
! 00C3 TMP2
! 00C5 DATA ;TAPE READ WRITE DATA
! 00C6 BA ;BANK FOR CURRENT
! 00C7 FNBANK ;BANK OF FILENAME
! 00C8 RIBUF ;RS232 INPUT BUF PTR
! 00CA ROBUF ;RS232 OUTPUT BUF PTR
! 00CC KEYTAB ;KEYSCAN TABLE PTR
! 00CE IMPARM ;PRIM UTIL STRG CNTR
! 00D0 NDX ;INDEX/KEYBOARD QUE
! 00D1 KYNDX ;PENDING FUNCTION KEY
! 00D2 KEYIDX ;INDEX TO PENDING
! FUNCTION KEY STRING
! 00D3 SHFLAG ;SHIFT KEY STATUS
! 00D4 SFDX ;CURRENT KEY INDEX
! 00D5 LTSX ;LAST KEY INDEX
! 00D6 CRSM ;(CR) INPUT FLAG
! 00D7 MODE ;40/80 COLUMN MODE
! 00D8 GRAPHM ;TEXT GRAPHIC MODE
! 00D9 CHAREN ;RAM/ROM VIC CHAR
! FETCH FLAG

```

```

!;THE FOLLOWING LOCATIONS ARE SHARED BY
! SEVERAL ROUTINES
!;00DA SEDSAL ;PNTR FOR MOVELINE
! 00DC SEDEAL ;
! 00DE SEDT1 ;SAVPOS
! 00DF SEDT2 ;
!;00DA KEYSIZ ;PROG KEY VAR
! 00DB KEYLEN ;
! 00DC KEYNUM ;
! 00DD KEYNXT ;
! 00DE KEYBNK ;
! 00DF KEYTMP ;
!;00DA BITMSK ;TEMP/TAB&LINE WRAP
! 00DB SAVER ;YET ANOTHER TEMP
! 00DE-00DF ;LOCAL SCR EDIT VAR
! ;(40/80 MODE CHANGE)
!;00E0 ;PNTRS FOR SCR ED
!;00FF ;END OF SCR ED VAR
!0100-01FF CPU Stack
! 0100-010F F BUFFER
! 0100 ;BASIC DOS INTRFC VAR
! 0110-0149 BASIC DOS Using
! 0124 ;END OF DOS VARS
! 0125-0138 ;SPACE FOR PRT USING
!0200-02A1 BASIC and MON input buf
!02A2-02FB KERNAL RAM Code
! 02A2-02AE LDA ($ZP);y for MMU=x
! 02AA Zero-Page indirect addr
! 02AF-02BD STA ($ZP);y for MMU=x
! 02B9 Zero-Page indirect addr
! 02BE-02CC CMP ($ZP);y for MMU=x
! 02CB Zero-Page indirect addr
! 02CD-02CF JSR $02E3
! 02D0-02E2 Store CPU reg. in $02-09
! 02E3-02FB RTI based on $02-09
!02FC-033B Indirect vectors-unknown
! 0300 Print BASIC Message 403F
! 0302 BASIC Warm Start 40C6
! 0304 Tokenize BASIC Text 430D
! 0306 BASIC Text LIST 5151

```

```

! 0308      BASIC Char Dispatch 4AA2
! 030A      BASIC Token Eval.   78DA
! 030C-0310 Unknown vectors
! 0314      IRQ Hardware int.   FA65
! 0316      BRK interrupt       B003
! 0318      NMI Non-Mask. Int.  FA40
! 031A      OPEN                 EFBD
! 031C      CLOSE                F188
! 031E      CHKIN                F106
! 0320      CHKOUT               F14C
! 0322      CLRCHN              F226
! 0324      CHRIN               EF06
! 0326      CHROUT              EF79
! 0328      STOP                F66E
! 032A      GETIN               EEED
! 032C      CLALL               F222
! 032E      User-Defined        B006
! 0330      LOAD                F26C
! 0332      SAVE                F54E
! 0334-0348 KERNAL vectors
! 0334      CTLVAC ;EDIT;PRINT'CNTRL'IND
! 0336      SHFVAC ;ED;PRINT'SHIFTED'IND
! 0338      ESCVAC ;ED;PRINT'ESCAPE'IND
! 033A      KEYVAC ;ED;KEYSCAN LOGIC IND
!033C-037F KERNAL Tables
!0380-03FF BASIC RAM Code
! 0380-039E BASIC CHRGET routine
! 039F-03D1 misc. LDA routines
!0400-04FF VIC Text Screen (VM #1)
!0800-09FF BASIC Run-Time Stack
!0A00-0AFF MON & KERNAL Abs. Var.
! 0A00-0A01 BASIC Cold Start 4000/3
! 0A02      RAM Init. if = $A5
! 0A04      BASIC init. if bit 0 set
! 0A20-      ;GLOBAL ABS SCR DECL
! 0A80-0AB4  ;MONITORS DOMAIN
! 0AC0      Number of Int./Ext. ROM's
! 0AC1-0AC4 Active ROM flags
!0B00-0BBF  Cassette Buffer
!0BC0-0BFF  (Disk Boot Page)

```

!0C00-0CFF	RS-232 Input Buffer
!0D00-0DFF	RS-232 Output Buffer
!0E00-0FFF	Sprite Definition Area
!1000-10FF	Function Key Buffer
! 1000-1009	# of bytes per Key
! 100A-	Function Key strings
!1100-1107	CP/M Reset Code
!1108-11FF	BASIC DOS/VSP Variables
!1200-12FF	BASIC Absolute Variables
! 117A-122A	Misc. vectors
! 1210	End of BASIC Program
!1300-13FF	?
!1400-17FF	Resv./Foreign Lang. Sys.
!1800-1BFF	Resv./Function Key ML
!1C00-FEFF	BASIC Program Area or
! 1C00-1FFF	VIC BIT-MAP Color (VM #2)
! 2000-3FFF	VIC BIT-MAP Screen
! 4000-FEFF	BASIC Program Area(con't)
!FF05-FF44	KERNAL Dispatch Code
!FFD0-	CP/M and KERNAL RAM Code
!FFFA-FFFF	NMI;RST; and IRQ vectors
!	
!****RAM(1)****	
!	
!0000-03FF	Common with RAM(0)
!0400-FEFF	BASIC Variable Storage
!FF05-FF44	KERNAL Int. Dispatch Code
!FFF5-FFF9	"cbm" and RST addr \$E224
!FFFA-FFFF	NMI;RST; and IRQ vectors

```

!          -----
!          C-128 ROM-I/O MAP
!          -----
!
!0000*      8502 On-Chip D-D Register
!0001*      8502 On-Chip I/O Register
!!!!!!!!!!!!
!4000-7FFF  BASIC Low ROM or
!           Internal Low ROM or
!           External Low ROM.
! 4000-      BASIC Power-up JMP $4023
! 4003-      BASIC RESET JMP $4009
! 4023-4044  BASIC Power-up
! 4045-410F  Init. BASIC registers
! 4112-4179  Init. BASIC Abs. Var.
! 417A-418C  Initialize D501-4
! 419B-41BA  Print RESET Screen
! 41BB-4250  RESET Screen Header Char.
! 4251-4278  Init. 0300-11;02FC
! 4279-43DD  BASIC CHRGET ML
! 43DD-      Tokenize BASIC Text
! 4417-4515  BASIC 2.0 Keywords
! 4516-46F3  BASIC 7.0 Keywords
! 484B-4A81  BASIC Error Messages
! 4AA2-      BASIC Char. Dispatch
! 4D3F-      Print BASIC Message
! 4DC6-      BASIC Warm Start
! 5151-      BASIC Text LIST
! 5109-5261  Init. BASIC Pgm vectors
! 6EB2-6EDA  Init. BASIC Abs. Var.
! 73DA-      BASIC Token Eval.
! 7E82-7FFD  Blank
!!!!!!!!!!!!
!8000-BFFF  BASIC Mid ROM or
!           Internal ROM or
!           External ROM.
! 9251-9298  BASIC to KERNAL JMP Table
! A845-A84C  To BANK(15)
! AA6E-AE62  Blank

```

```

! AFA8-AFFF      Blank
! B000-BFFF      Monitor ROM
!   B000-        Monitor RST code
!   B003-        BRK
!   BB72-BFFD    Blank
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!C000-CFFF      Editor High ROM or
!               Internal ROM or
!               External ROM.
!   C000-        CINT -JMP $C07B
!   C00F-        SCREEN
!   C012-        SCNKEY
!   C018-        PLOT
!   C07B-        CINT
!   CEAB-CEF4    Function Key Init. values
!   CEF5-CFFD    Blank
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!D000-DFFF      I/O Space
!   D500-D50B    MMU I/O chip (switchable)
!   D500         MMU Config. Reg. (<$FF00)
!   D501-D504    MMU Preconfig. Registers
!   D505         bit 7 - 40/80 switch 1=40
!               bit 6 - 128/64 mode 1=64
!               bit 5 - GAME line; bi-dir
!               bit 4 - EXROM line; bi-d
!               bit 3 - Fast Serial I/O
!               bit 1-2 - reserved
!               bit 0 -8502/280 CPU 0=280
!   D506         RAM Config. Register
!               bits 0-1 -K of Common RAM
!                   00-1K      01-4K
!                   10-8K      11-16K
!               bits 2-3 -Loc./Common RAM
!                   00-none     01-Bottom
!                   10-Top      11-Both
!               bits 4-5 -reserved
!               bit 6 -RAM # used by VIC
!               bit 7 -reserved
!   D507         Page for CPU Zero Page
!   D508         LSB-RAM # for CPU ZP

```

```

! D509      Page for CPU stack
! D50A      LSB-RAM # for CPU stack
! D50B      System Version Register
!           bits 0-3 -MMU chip vers.
!           bits 4-7 -Code for K/RAM
!           0010- 128K
!           0000- 256K
! D50C-D5FF Blank
! D600-D601 Access to 16K display RAM
! D600      Register addresses
! D601      Data
! DC00-DCFF CIA #1 (Keyboard; etc.)
! DD00-DDFF CIA #2 (Serial; etc.)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!E000-FFFF  KERNAL High ROM
! E000-     KERNAL ROM Code
! E000-E048 RST Code
! E048-E055 $D500 Init. values
! E056-     RESTORE
! E05B-     VECTOR
! E093-     RAMTAS
! E0CD-E108 Init.$FF05- on RAM(0-3) &
!           KERNAL RAM Code on RAM(0)
! E109-E1EF IOINIT routine
! E1F0-E223 If $FFF5-9 on RAM(1)=cbm,
!           then JMP ($FFF8);else...
! E224-E241 Init. $FFF5-9 on RAM(1)
! E242-E2BF Switch to 64 mode if D505
!           bits 4-5 not set; check
!           Int./Ext. ROM and JMP
! E24B-E26A Switch to 64 mode
! E33B-     TALK
! E33E-     LISTEN
! E43E-     ACPTR
! E4D2-     SECOND
! E4E0-     TKSA
! E503-     CIOUT
! E515-     UNTLK
! E526-     UNLSN
! F23D-F264 Close all files on dev=ac

```

```

! F265-      LOAD
! F53E-      SAVE
! F5F8-      UDTIM
! F63D-F65D  Check for CTRL R/S or C=
! F65E-      RDTIM
! F665-      SETTIM
! F6B1-F71D  KERNAL Messages
! F731-      SETNAM
! F738-      SETLFS
! F744-      READST
! F75C-      SETMSG
! F75F-      SETTMO
! F763-      MEMTOP
! F772-      MEMBOT
! F781-      IOBASE
! F7D0-F7D9  Calc KERNAL RAM Code 02A2
!             to LDA byte from addr;y
!             at Zero-Page vector in
!             acc for BANK(x)
! F7EC-F7EF  LDA MMU value for BANK(x)
! F7F0-F7FF  MMU values for BANK(0-15)
! F800-F859  KERNAL RAM Code $02A2-FB
! F85A-F866  KERNAL RAM Code $03F0-FC
! F867-F888  Check for ROM not used
!             on RST ; Load Boot Page
!             and execute
! F9D5-F9FA  Load page and STA($ACC)
! FA00-      Editor Tables
! FA17-FA3F  Print bytes after JSR
! FA40-      NMI
! FA65-      IRQ
! FC3B-FC7F  Blank
! FC80-FEFF  Foreign Lang. Sys.-Blank
!FF00*      MMU Config. Register
!FF01-FF04*  MMU Load Config. Regs.
! FF05-FF44  KERNAL Int. Dispatch Code
! FF3D-FF44  RST Code JMP $E000
! FF47-FF80  KERNAL Hardware JMP Table
! FF81-FFF5  KERNAL User Jump Table
! FFFA-FFFF  NMI;RST; and IRQ vectors

```

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!* These are I/O registers and take the
! place of RAM or ROM always.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!MMU Configuration Register $D500/FF00
! bits      values      purpose
! 6-7  RAM control-
!           00  RAM(0)
!           01  RAM(1)
!           10  RAM(2)-not present
!           11  RAM(3)-not present
! 4-5  ROM control-
!           00  KERNAL and EDIT ROM
!           01  Internal High ROM
!           10  External High ROM
!           11  None
! 2-3  Mid ROM control- $8000-BFFF
!           00  BASIC ROM
!           01  Internal ROM
!           10  External ROM
!           11  None
! 1    Low ROM Control- $4000-7FFF
!           0=ROM      1=RAM
! 0    I/O-ROM Control- $0000-DFFF
!           0=I/O      1=CHAR2000

```


A L C U N E R O U T I N E
D E L S I S T E M A
O P E R A T I V O

ALCUNE ROUTINE DEL SISTEMA OPERATIVO

PREMETTO CHE LE NOTIZIE CHE SEGUONO SUL SISTEMA OPERATIVO DEL COMMODORE '128' SONO STATE RICAVATE IN MASSIMA PARTE DA SPERIMENTAZIONI SULLA MACCHINA DELL' AUTORE, MA ANCHE DA RIVISTE DEL SETTORE CHE, SIA PURE MOLTO LENTAMENTE, SI COMINCIANO AD INTERESSARE A QUESTO CALCOLATORE PUBBLICANDO PROGRAMMI E NOTIZIE UTILI.

SEGUE UN ELENCO DI ROUTINE DEL SISTEMA OPERATIVO.

- INDIRIZZO \$FFD4 65357

QUESTA ROUTINE SERVE A PASSARE DAL MODO '128' AL MODO '64'. VENGONO ESEGUITE ALCUNE OPERAZIONI FONDAMENTALI PER GARANTIRE LA COMPATIBILITA' CON TUTTI IL SOFTWARE DEL '64' VECCHIA MANIERA, A TAL PROPOSITO L' AUTORE HA RISCONTRATO IN QUALCHE PROGRAMMA COMMERCIALE (POCHI PER LA VERITA') INCOMPATIBILTA' FRA IL '128' IN MODO '64' E' IL '64' VECCHI STILE, AD UN PIU' ATTENTO ESAME DELLA MAPPA DI MEMORIA SI E' RISCONTRATO LA DIFFERENZA DI CONTENUTO DI ALCUNE LOCAZIONI DI MEMORIA.

LA DIFFERENZA CHE PIU' FA PREOCCUPARE E' QUELLA LEGATA ALLA LOCAZIONE 1 CHE HA CONTENUTI DIVERSI PER LE DUE MACCHINE. COME SI SAPRA' QUESTA LOCAZIONE FUNZIONA DA PORTA DI COMMUTAZIONE DI ALCUNI BANCHI DI MEMORIA, IL FATTO CHE SUL '128' IN MODO '64' SIA DIVERSA DA QUELLA DEL 'VECCHIO 64' POTREBBE PORTARE QUALCHE INCONVENIENTE, TALE DA PREGIUDICARE LA COMPATIBILTA' TANTO DECANTATA.

- INDIRIZZO \$FF50 65360

QUESTA ROUTINE INIZIALIZZA LA MEMORIA R.A.M.
SERVE ANCHE A RICONFIGURARE IL SISTEMA QUANDO SIANO PRESENTI
DELLE ESPANSIONI DI MEMORIA.

- INDIRIZZO \$FF53 65363

QUESTA E' LA ROUTINE CHE ESEGUE IL BOOT DAL DISCO QUANDO
VIENE INCONTRATO NELLE PRIME TRACCE DEL DISCO UN FILE
OPPORTUNO, A QUESTO PROPOSITO VALE LA PENA DI DARE QUALCHE
NOTIZIA IN PIU':

ALL' ACCENSIONE DEL COMPUTER IL SISTEMA OPERATIVO, CONTROLLA
SE E' COLLEGATO AL SISTEMA UN DRIVE FUNZIONANTE.

SE IL DRIVE E' COLLEGATO VIENE LETTA LA PRIMA TRACCIA
SETTORE 0

VENGONO ESAMINATI I PRIMI CARATTERI E SE QUESTI RISULTANO
UGUALI A 'CBM' VIENE ESEGUITA QUALSIASI ROUTINE IN
LINGUAGGIO MACCHINA CHE SI TROVI SCRITTA SULLO STESSO
BLOCCO.

QUESTO ASPETTO DELLA FACCENDA E' SENZA DUBBIO INTERESSANTE
IN QUANTO SI POSSONO PRODURRE PROCEDURE DI AUTOSTART DEI
DISCHETTI VERAMENTE EFFICACI (AVETE PRESENTE IL DISCO DEL
CPM ?).

- INDIRIZZO \$FF56 65366

QUESTA ROUTINE ESEGUE LA PARTENZA A FREDDO DEL SISTEMA,
CONTEMPORANEAMENTE VENGONO RILEVATI ALCUNI PARAMETRI, COME
LA PRESENZA DI UNA CARTUCCIA NELLA PORTA O LA PRESENZA DI
QUALCHE ESPANSIONE.

- INDIRIZZO \$FF59 65369

QUESTA ROUTINE SERVE A MANIPOLARE I FILES, IN PARTICOLARE
SERVE ALLA RICERCA DI UN FILE LOGICO.
NON HA NESSUNA UTILITA' PRATICA PER LA GESTIONE DEI FILES.

- INDIRIZZO \$FF5C 65372

QUESTA ROUTINE RICERCA I PARAMETRI DI UN FILE BASANDOSI SUL
CARATTERE SECONDARIO

- INDIRIZZO \$FF5F 65375

QUESTA ROUTINE SERVE AD INVERTIRE IL MODO 40-80 COLONNE DI
TESTO

- INDIRIZZO \$FF62 65378

QUESTA ROUTINE VIENE USATA DAL SISTEMA OPERATIVO PER LA
GESTIONE DELLE 80 COLONNE E SERVE IN PARTICOLARE AD USARE IL
SET DI CARATTERI APPROPRIATO AL FORMATO DELLO SCHERMO
ADOPERATO.

- INDIRIZZO \$FF65 65381

QUESTA ROUTINE SERVE A CAMBIARE I TASTI FUNZIONE.
VOLENDO PUO' ESSERE ADOPERATA DA LINGUAGGIO MACCHINA,

PONENDO NELL' ACCUMULATORE L' INDIRIZZO CHE FA DA PUNTATORE ALLA PAROLA ASSEGNATA AL TASTO FUNZIONE, IL NUMERO DEL QUALE DEVE ESSERE DEPOSITATO NEL REGISTRO X.

- INDIRIZZO \$FF68 65384

QUESTA ROUTINE DEFINISCE SU QUALE BANCO DI MEMORIA DEVONO ESSERE RIVOLTE LE OPERAZIONI DI INPUT/OUTPUT.

- INDIRIZZO \$FF6B 65387

QUESTA ROUTINE PRELEVA IL COSIDDETTO 'BYTE DI CONFIGURAZIONE' CIOE' IL SISTEMA SI RENDE CONTO SE HA ESPANSIONI DI MEMORIA COLLEGATE O PERIFERICHE ECC. ECC.

- INDIRIZZO \$FF6E 65390

QUESTA ROUTINE SERVE AD ADATTARE I SALTII QUANDO SONO PRESENTI DELLE ESPANSIONI DI MEMORIA CHE COINVOLGONO BANCHI DI MEMORIA DIVERSI DA QUELLI STANDARD

- INDIRIZZO \$FF71 65393

QUESTA ROUTINE TRASFERISCE IL CONTROLLO A PROGRAMMI CHE RISIEDONO IN BANCHI DIVERSI DA QUELLO STANDARD

- INDIRIZZO \$FF74 65396

QUESTA ROUTINE PRELEVA UN BYTE DA QUALSIASI BANCO DI MEMORIA

- INDIRIZZO \$FF77 65399

QUESTA ROUTINE DEPOSITA L' ACCUMULATORE IN UNA LOCAZIONE APPARTENENTE A QUALSIASI BANCO DI MEMORIA

- INDIRIZZO \$FF7D 65402

QUESTA ROUTINE CONSEGNA UNA SERIE DI BYTE ASSEGNATI AD UNA PERIFERICA

- INDIRIZZO \$FFA5 65445

QUESTA ROUTINE ACCETTA DATI DAL BUS SERIALE.
SERVE A GESTIRE TUTTE LE COMUNICAZIONI DA E VERSO IL DRIVE E
COMUNQUE CON TUTTI I DISPOSITIVI CONNESSI CON QUESTO BUS

- INDIRIZZO \$FFC6 65478

QUESTA ROUTINE APRE UN CANALE DI INPUT VERSO QUALUNQUE PERIFERICA CONNESSA, ANCHE CON LA TASTIERA, NATURALMENTE NON CON LA STAMPANTE.

- INDIRIZZO \$FFC9 65481

QUESTA ROUTINE APRE UN CANALE DI OUTPUT.
NATURALMENTE LA PERIFERICA INTERESSATA DEVE ESSERE IN GRADO DI RICEVERE IL FLUSSO DI INFORMAZIONI GENERATE DALLA UNITA' CENTRALE.

- INDIRIZZO \$FFCF 65487

QUESTA ROUTINE ACCETTA UN DATO DAL CANALE DI INPUT PRECEDENTEMENTE APERTO.

SE NESSUN CANALE E' STATO PRECEDENTEMENTE APERTO L' INPUT VIENE INTERPRETATO COME PROVENIENTE DALLA TASTIERA.

- INDIRIZZO \$FFD2 65490

QUESTA ROUTINE INVIA UN DATO SU DI UN CANALE APERTO CON LA ROUTINE VISTA IN PRECEDENZA.

SE NESSUN CANALE E' STATO PRECEDENTEMENTE APERTO I DATI SARANNO INVIATI AL VIDEO.

- INDIRIZZO \$FFA8 65448

QUESTA ROUTINE SERVE AD INVIARE INFORMAZIONI SUL BUS SERIALE, A PATTO CHE LA PERIFERICA INTERESSATA SIA IN STATO DI ASCOLTO.

- INDIRIZZO \$FFC3 65475

ROUTINE CHE CHIUDE UN FILE LOGICO SPECIFICO, APERTO IN PRECEDENZA CON L' APPOSITA ROUTINE.

- INDIRIZZO \$FFCC 65484

QUESTA ROUTINE PULISCE I BUFFER DATI DI TUTTI I CANALI APERTI, PER EVITARE IN TRASMISSIONI SUCCESSIVE, PERICOLOSE SOVRAPPOSIZIONI DI DATI

- INDIRIZZO \$FFE7 65511

ROUTINE EQUIVALENTE A QUELLA DEL COMMODORE '64', SERVE A CHIUDERE TUTTI I FILE APERTI

- INDIRIZZO \$FFE4 65508

QUESTA ROUTINE RICEVE UN CARATTERE DA UNA PERIFERICA. IN PARTICOLARE, SE QUESTA PERIFERICA E' LA TASTIERA, VIENE LETTA LA CODA DEL BUFFER CORRISPONDENTE. RICORDO CHE IL BUFFER DI TASTIERA PUO' CONTENERE MASSIMO 10 CARATTERI

- INDIRIZZO \$FFD5 65493

ANCHE QUESTA ROUTINE E' IDENTICA A QUELLA DEL '64', E SERVE A CARICARE DATI DA UNA PERIFERICA NELLA R.A.M. UTENTE. QUESTA ROUTINE PUO' ESSERE ANCHE USATA PER VERIFICARE QUANTO SCRITTO SU DISCO O SU NASTRO. PER COMPIERE UN LOAD O UN VERIFY, L' ACCUMULATORE DEVE ASSUMERE PARTICOLARI VALORI:

1 -----> PER VERIFICA

0 -----> PER LOAD (CARICAMENTO DATI)

SI POSSONO FARE CARICAMENTI NORMALI O CARICAMENTI RILOCATI. NON E' POSSIBILE ESEGUIRE UN LOAD DA TASTIERA O DA SCHERMO (NATURALMENTE)

- INDIRIZZO \$FF9C 65436

QUESTA ROUTINE SERVE A FISSARE IL LIMITE BASSO DELLA MEMORIA.

TALE OPERAZIONE RISULTA UTILE QUANDO SI VOGLIONO PROTEGGERE DALLE POSSIBILI SOVRAPPOSIZIONI DEI PROGRAMMI BASIC, AREE DI MEMORIA DESTINATE A ROUTINE IN LINGUAGGIO MACCHINA O PAGINE GRAFICHE ECC. ECC.

- INDIRIZZO \$FF99 65433

QUESTA ROUTINE FUNZIONA COME LA PRECEDENTE E SERVE A COMPIERE LE STESSSE OPERAZIONI, SOLO CHE QUESTA VOLTA IMPOSTA IL TOP DELLA MEMORIA PRESERVANDO AREE AL DI SOPRA DEI NORMALI PROGRAMMI BASIC

- INDIRIZZO \$FFF0 65520

QUESTA ROUTINE SERVE A POSIZIONARE IL CURSORE IN UN PUNTO DELLO SCHERMO O A RILEVARE LE COORDINATE DEL CURSORE.

LE DUE FUNZIONI VENGONO SETTATE DEA VALORE CHE ASSUME UN BIT PARTICOLARE DEL REGISTRO DI STATO DENOMINATO 'BIT DI CARRY' LETTERALMENTE 'TRASPORTO'.

SE IL CARRY E':

1 -----> VIENE RILEVATA LA POSIZIONE DEL CURSORE

0 -----> VIENE POSIZIONATO IL CURSORE IN UN PUNTO DELLO SCHERMO.

IN OGNI CASO I REGISTRI X ED Y DEL MICROPROCESSORE, INDICANO LE COORDINATE DEL CURSORE.

- INDIRIZZO \$FF87 65415

QUESTA ROUTINE VIENE USATA ALL' ACCENSIONE DEL SISTEMA E

OLTRE A FISSARE I PUNTATORI DELLA R.A.M. ESEGUE UNA DIAGNOSTICA SULLA MEMORIA STESSA RIPORTANDO DEI CODICI DI ERRORE SE QUALCHE COSA NON HA FUNZIONATO A DOVERE. DA NOTARE CHE ALLA FINE DEL TEST LA ROUTINE IN QUESTIONE PULISCE QUASI TUTTE LE LOCAZIONI DEL BASIC

- INDIRIZZO \$FFDE 65502

QUESTA ROUTINE LEGGE L' OROLOGIO DI SISTEMA CON LA RILEVAZIONE DI UN TEMPO MINIMO PARI A 1/60 ESIMO DI SECONDO.

- INDIRIZZO \$FFB7 65463

QUESTA ROUTINE VIENE IN GENERE CHIAMATA DOPO UN QUALUNQUE COLLOQUIO CON UNA PERIFERICA, E RIPORTA LA CONDIZIONE CORRENTE, IN ALTRE PAROLE RIPORTA UN EVENTUALE CODICE DI ERRORE SE LA COMUNICAZIONE E' AVVENUTA CON DEGLI ERRORI

- INDIRIZZO \$FF8A 65418

QUESTA ROUTINE RICOPIA ALL' ACCENSIONE DEL SISTEMA TUTTI I VALORI DEI PUNTATORI E DELLE LOCAZIONI DI MEMORIA USATE DAL SISTEMA DALLA R.O.M ALLA R.A.M CORRISPONDENTE

- INDIRIZZO \$FFD8 65496

QUESTA ROUTINE SERVE A SALVARE I CONTENUTI DELLA R.A.M. SU DI UNA PERIFERICA.

- INDIRIZZO \$FF9F 65439

QUESTA ROUTINE CONTROLLA SE CI SONO TASTI PREMUTI, ESEGUE UNA SCANSIONE COMPLETA DELLA TASTIERA. SE UN TASTO E' PREMUTO, IL CORRISPONDENTE CODICE ASCII VIENE DEPOSTO IN CODA AL BUFFER DI TASTIERA, PER ELABORAZIONI SUCCESSIVE

- INDIRIZZO \$FFE1 65505

QUESTA ROUTINE ESEGUE LA SCANSIONE DELLA TASTIERA COME LA PRECEDENTE, MA SI OCCUPA DI RILEVARE SOLTANTO L' AVVENUTA PRESSIONE DEL TASTO DI 'STOP'

- INDIRIZZO \$FFEA 65514

QUESTA ROUTINE INCREMENTA L' OROLOGIO DI SISTEMA ED E' , PER QUESTO MOTIVO, INSERITA NELLA LINEA DI INTERRUPT DEL CICLO MACCHINA

L E T T U R A F I L E
S E Q U E N Z I A L I

LETTURA FILE SEQUENZIALI

IL LISTATO CHE SEGUE E' UN PROGRAMMA SCRITTO DALL' AUTORE APPPOSITAMENTE PER IL COMMODORE '128'.

SERVE A COMPIERE OPERAZIONI SU FILE SEQUENZIALI COME, LETTURA, SCRITTURA, CANCELLAZIONE, VISIONE DELLA DIRECTORY, ECC.ECC, ED E' COMPATIBILE CON MOLTI PROGRAMMI TIPO WORD-PROCESSOR O DATA BASE CHE GENERANO COME TRANSITO DEI DATI, APPUNTO, FILE SEQUENZIALI.

IL PROGRAMMA ALL' INIZIO RIDEFINISCE TUTTI I TASTI FUNZIONE CON LE NUOVE ISTRUZIONI DEL '128' PER USI INTERNI DEL PROGRAMMA, CONSENTENDO ALL' OPERATORE DI ACCEDERE ALLE FUNZIONI TRAMITE LA PRESSIONE DI UN TASTO.

ALLA FINE DELL' ESECUZIONE VENGONO RIPRISTINATI, PER I TASTI FUNZIONE I NORMALI USI.

IL PROGRAMMA PUO' FUNZIONARE SIA A 40 CHE A 80 COLONNE, BASTA RISPONDERE ALLA DOMANDA CHE VIENE POSTA ALL' UTENTE DOPO IL LANCIO.

INUTILE PRECISARE CHE PER FUNZIONARE AD 80 COLONNE IL '128' DEVE ESSERE CONNESSO AD UN MONITOR DI TIPO RGB E NON AL CONSUETO MONITOR COMPOSITO O PEGGIO AL NORMALE TELEVISORE.

BISOGNA PRESTARE PARTICOLARE ATTENZIONE A TUTTE LE RIGHE DI PROGRAMMA CHE CONTENGONO ISTRUZIONI DI TIPO 'PRINT', IN QUANTO I NORMALI CARATTERI DI CONTROLLO ADOPERATI DAL '128' PER POSIZIONARE IL CURSORE O PER OTTENERE DELLE FUNZIONI SPECIALI, (CHE POI SONO QUELLE DEL NORMALE COMMODORE '64') NON SONO RICONOSCIUTI DAL WORD-PROCESSOR ADOPERATO PER COMPORRE QUESTO LIBRO, E QUINDI, SONO STATI STAMPATI IN MANIERA ANOMALA.

PER RIMEDIARE A QUESTO INCONVENIENTE, QUI DI SEGUITO VIENE PROPOSTA UNA TABELLA DI CONVERSIONE TRA I SEGNI CHE COMPAGNONO SUL LISTATO, E COME IN REALTA' SI DEVONO OTTENERE DURANTE LA DIGITAZIONE; QUESTO DOVREBBE RIMETTERE LE COSE A POSTO.

SUL LISTATO	SULLA TASTIERA	EFFETTO
Q	CRSR GIU'	BASSO
J	CRSR DESTRA	DESTRA
R	CONTROL+9	REVERSE

```

10
KEY1,CHR$(133):KEY3,CHR$(134):KEY5,CHR$(135):KEY7,CHR$(136):
KEY2,CHR$(137):KEY4,CHR$(138):KEY6,CHR$(139):KEY8,CHR$(140)
20 PRINTCHR$(14)"sQQQQQ"TAB(12)"FILE SEQUENZIALE"
30 PRINTTAB(12)"QULETTURA/ESTAMPA":PRINTTAB(18)"QQDI"
40 PRINTTAB(13)"QVITTORIO PAOLA"
42 PRINTTAB(12)"QQQPER C '123' "
43 PRINTTAB(14)"40/80 COL"
44 FORX=1TO2000:NEXT
45 GOSUB800
50 PRINT"s"CHR$(14)"OPZIONI":PRINT"QF1 = DIRECTORY"
60 PRINT"QF3 = SOLO SCHERMO":PRINT"QF5 = HARDCOPY"
70 PRINT"QF7 = CANCELLA FILE"
80 GETKEY K$:IFK$(CHR$(133)OR"K")>CHR$(136)THEN30
90 IFK$=CHR$(136)THEN570
100 IFK$>CHR$(133)THEN310
110 PRINT"sRDIRECTORYrQQ":PRINT"F1 = SOLO SCHERMO"
120 PRINT"QF3 = HARDCOPY":PRINT"QF5 = FUNZIONI PER
CANCELLARE":P=0
130 GETKEY K$:IFK$(CHR$(133)OR"K")>CHR$(135)THEN130
140 IFK$=CHR$(135)THEN50
150 IFK$=CHR$(133)THEN260
160
OPEN4,4:PRINT#4:PRINT#4,CHR$(14)"DIRECTORY"CHR$(15):PRINT#4:
PRINT#4:P=1
170
PRINT"sRDIRECTORYr":PRINT:PRINT:OPEN1,S,0,"#0":GET#1,A$,B$:N
$=CHR$(0)
180 GET#1,A$,B$:IFB$=""THEN280
190 GET#1,A$,B$:PRINTASC(A$+N$)+256*ASC(B$+N$);
200 IFP=1THENPRINT#4,ASC(A$+N$)+256*ASC(B$+N$);
210 GET#1,A$:IFA$=""THEN240
220 IFP=1THENPRINT#4,A$;
230 PRINTA$;GOTO210
240 PRINT:IFP=1THENPRINT#4
250 GOTO180

```

```

260 DIRECTORY
270 PRINT"QQPREMI UN TASTO":GETKEY K$:GOTO50
290 IFP=1THENFORX=1TO7:PRINT#4:NEXTX:P=0
290 PRINT:PRINT"PREMI UN TASTO PER CONTINUARE":CLOSE1:CLOSE4
300 GETKEYK$
310 OPEN1,4,7:L=1:P=0:IFK$=CHR$(135)THENP=1
320 PRINT"sRNome DEL FILE :r ":INPUTF$
330 OPEN8,8,8,F$+",S,R":GOSUB720
340 PRINT"sRPREMI UN TASTO PER PAUSA r"
345 PRINT"QQPREMI 'no scroll' PER PAUSA
"
250 PRINT:PRINT:PRINTTAB(10)"R"F$r":PRINT:PRINT
360
IFP=1THENPRINT#1,CHR$(145)CHR$(14)F$CHR$(15)CHR$(17):PRINT#1
:PRINT#1:L=4
370 GET#8,CH$
390 PRINTCH$:IFST<>0THEN510
400 IFP=1THENPRINT#1,CH$:IFCH$=CHR$(13)THENL=L+1:GOSUB750
410 GETK$:IFK$=""THEN370
420 PRINT:FORX=1TO39:PRINT"*":NEXTX:PRINT:PRINT"RPAUSA IN
CORSO"
430 PRINT"OPTIONS:":PRINT"F1 = STAMPANTEON":PRINT"F3 =
STAMPANTE OFF"
440 PRINT"F5 = OPERAZIONE ABORTITA":PRINT"F7 = CONTINUA
"
450 GETKEY K$:IFK$<CHR$(123)OR K$>CHR$(135)THEN450
460
IFK$=CHR$(133)THENP=1:PRINT#1,CHR$(17):PRINT#1:PRINT#1:L=3:G
OTO490
470 IFK$=CHR$(134)THENP=0:GOTO490
480 IFK$=CHR$(135)THEN500
490
PRINT"RCONTINUINGr":FORX=1TO39:PRINT"*":NEXT:PRINT:GOTO370
500 PRINT"sQQQQQ"TAB(11)CHR$(142)"OPERAZIONE ABORTITA"
510
IFP=1THENPRINT#1:FORX=L+2TO66:PRINT#1:NEXTX:PRINT#1,CHR$(145
)
520 PRINTDS$
530 CLOSE1:CLOSE8:PRINT:PRINTCHR$(14)"QQVUOI LEGGERE UN
":PRINT"ALTRO FILE ? (Y/N)"

```

```

540 GETKEY K$: IFK$="Y" THEN CLR: GOT050
550 IFK$(">") "N" THEN 540
560 GOT0900
570 PRINT CHR$(142): GOSUB 770
580 PRINT "QQQNome DEL FILE DA CANCELLARE";
590 PRINT "0 PREMI RETURN PER ANNULLARE"
600 PRINT "RNome FILE: r"; INPUT F$
610 IFF$=" " THEN 660
620 GOSUB 770: PRINT "QQ"; PRINT "QQVUOI VERAMENTE CANCELLARE ";
630 PRINT "QIL R "F$" r FILE? (Y/N)"
640 GETKEY K$: IFK$=" " THEN 640
650 IFK$="Y" THEN 670
660 PRINT CHR$(14): GOT0500
670 GOSUB 770: PRINT "QQSTO CANCELLANDO "F$
680 CLOSE 15: OPEN 15, 8, 15: PRINT #15, "S0:" F$
690 FOR X=1 TO 2000: NEXT X: CLOSE 15
700 GOSUB 770: PRINT "QQ"F$" IS NOW DEAD!!"
710 FOR X=1 TO 2000: NEXT X: PRINT CHR$(14): GOT050
720 IF DS>0 THEN PRINT: PRINT "ERRORE" SUL DISCO
": PRINT DS$: GOT0530
730 IF DS>0 THEN PRINT: PRINT "ERRORE" SUL DISCO
": PRINT DS$: GOT0530
740 RETURN
750 IFL=60 THEN L=1: FOR X=1 TO 7: PRINT #1: NEXT X
760 RETURN
770 PRINT "sQQQQQ" TAB(8) "RFUNZIONE PER CANCELLARE ATTIVATA"
780 RETURN
800 PRINT "sQQQQQ11111111R4r0 COLONNE 0 R2r0 COLONNE?"
805 PRINT "QQ11111111111111SCEGLI R4r 0 R2r"
810 GETKEY K$
820 IFK$="4" THEN PRINT "sQQ": FOR X=1 TO 10: PRINT "*****" 40
COLONNE *****q": NEXT X: FOR X=1 TO 1500: NEXT X: RETURN
830 IFK$(">") "8" THEN 810
840 PRINT "s ": FAST
850 FOR X=1 TO 20
860 PRINT "80 COLONNE***** 80 COLONNE***** 80
COLONNE***** 80 COLONNE*****q"
870 NEXT
880 FOR X=1 TO 4000: NEXT

```

```
890 RETURN
```

```
900
```

```
KEY1,"GRAPHIC":KEY2,"DLOAD"+CHR$(34):KEY3,"DIRECTORY"+CHR$(1  
3):KEY4,"SCNCLR"+CHR$(13):KEY5,"DSAVE"+CHR$(34):KEY6,"RUN"+C  
HR$(13):KEY7,"LIST"+CHR$(13)
```

```
910 KEY8,"MONITOR"+CHR$(13):SLOW
```

C O D I C I D I E R R O R E

MESSAGGI DI ERRORE

DURANTE LA SCRITTURA USANDO FILE RELATIVI, SI PUO' INCORRERE IN LAMPEGGIAMENTI DELLA SPIA ROSSA DEL DRIVE, CIO' INDICA UNA CONDIZIONE DI ERRORE, MA IN QUESTO CASO E' PERFETTAMENTE NORMALE.

IL FATTO AVVIENE PERCHE' IL COMPUTER CERCA DI SCRIVERE SU UN RECORD DOVE NON E' MEMORIZZATA NESSUNA INFORMAZIONE, PERTANTO NON LO RICONOSCE E VA IN ERRORE.

COMUNQUE NELLA VARIABILE DI STATO DS\$ E' SEMPRE CONTENUTO UN MESSAGGIO CHE SINTETIZZA LA CAUSA DI EVENTUALI ERRORI GENERATI DURANTE IL FUNZIONAMENTO DEI PROGRAMMI.

QUI DI SEGUITO UNA DESCRIZIONE DEI MESSAGGI DI ERRORE CHE SI POSSONO OTTENERE CON:

PRINT DS\$

-TOO MANY FILES

QUESTO ERRORE VIENE GENERATO QUANDO SI TENTANO DI APRIRE PIU' DI 10 FILES CONTEMPORANEAMENTE.

- FILE OPEN

QUESTO ERRORE INSORGE QUANDO SI TENTA DI APRIRE UN FILE CHE A QUEL MOMENTO, RISULTA ATTIVO.

- FILE NOT OPEN

QUESTO ERRORE SI VERIFICA QUANDO SI TENTA UN ACCESSO AL FILE, SENZA CHE QUESTO SIA STATO APERTO IN PRECEDENZA.

- FILE NOT FOUND

IL NOME DEL FILE CERCATO NON ESISTE NELLA DIRECTORY CORRENTE.

NEL CASO I TRATTI DI FILE SEQUENZIALI CONTROLLARE L' INDIRIZZA SECONDARIO CHE DEVE ESSERE 0 PER LETTURE E 1 PER SCRITTURE.

- DEVICE NOT PRESENT

IL DRIVE, O COMUNQUE LA PERIFERICA A CUI E' RIVOLTO IL COMANDO, E' SPENTA O NON CORRETTAMENTE COLLEGATA AL SISTEMA.

- NOT INPUT FILE

SI E' TENTATO DI ESEGUIRE UNA LETTURA IN UN FILE APERTO PER SCRITTURA.

NEL CASO DI FILE SEQUENZIALI, CONTROLLARE L' INDIRIZZO SECONDARIO.

- NOT OUTPUT FILE

SI E' TENTATO DI ESEGUIRE UNA SCRITTURA SU UN FILE APERTO PER LA LETTURA, ANCHE IN QUESTO CASI CONTROLLARE L' INDIRIZZO SECONDARIO.

- MISSING FILE NAME

IN UN COMANDO DI LETTURA O SCRITTURA E' STATO OMESSO IL NOME DEL FILE SU CUI INDIRIZZARE LE OPERAZIONI.

- ILLEGAL DEVICE NUMBER

E' STATO EFFETTUATO UN TENTATIVO DI USARE UNA PERIFERICA ILLEGALE.

QUESTO ERRORE INSORGE, SE SI TENTA DI USARE IL SECONDO DRIVE SENZA CHE QUESTO SIA CONNESSO O SE SI TENTA DI SALVARE DELLE INFORMAZIONI CON CODICE 0, CIOE' SULLO SCHERMO ECC. ECC.

- BAD DISK

E' STATA TENTATA UNA FORMATTAZIONE CON IL COMANDO 'HEADER' SU DI UN DISCO DIFETTOSO.

B I B L I O G R A F I A

BIBLIOGRAFIA

- JAUQUES BOISGONTIER, IL BASIC E LA GESTIONE DEI FILES, VOL 2, JACKSON.
- AUTORI VARI, GUIDA DI RIFERIMENTO AL C 128, EVM.
- VITTORIO PAOLA E GAETANO ROMANO, TECNICHE AVANZATE DI PROTEZIONE E SPROTEZIONE DEL SOFTWARE, LIBRERIA DARIO FLACCOVIO EDITRICE.
- DANIEL JEAN DAVID, PROGRAMMARE IN PASCAL, JACKSON.
- ROBERTO DORETTI, DATABASE CONCETTI E DISEGNO, JACKSON.
- DANIEL MARTIN, BANCA DATI APPLICAZIONI SU PICCOLI E GRANDI CALCOLATORI, TECNICHE NUOVE
- A. BRUSAMOLIN MANTOVANI, INFORMATICA, EDIZIONI CEDAM PADOVA.
- GLORIANO ROSSI, UTILITY PER IL COMMODORE '64', EDIZIONI ACANTHUS.
- MANUALI VARI COMMODORE

INDICE

Prefazione	Pag.	5
Comandi 128	»	7
Cosa sono i files?	»	11
Vediamo il disco più da vicino	»	17
Organizziamo un archivio	»	23
Data base	»	29
Qualche semplice esempio	»	35
Qualche esempio più complesso	»	47
Come ricercare i dati	»	61
Riordino dei dati	»	77
Mappe di memoria	»	93
Alcune routine del sistema operativo	»	111
Lettura file sequenziali	»	123
Codici di errore	»	131
Bibliografia	»	137

Finito di stampare
dalla Tipolito Priulla
per conto della
Libreria Dario Flaccovio Editrice
Palermo, Giugno 1986

