# Optimizer DBase III

*di Daniele Bufarini - Rieti*

*Il DB III è ormai diventato lo strumento principale per chi si occupa di gestione degli archivi o di programmi commerciali vari. Non è raro ormai trovare delle contabilità sviluppate in DB III e poi magari compilate.*
*Proprio per chi usa assiduamente questo linguaggio di quarta generazione ecco due routine mandateci da due lettori quasi contemporaneamente, scritte con un linguaggio simile, il Quick Basic Microsoft e il Turbo Basic Borland, e che sono perfettamente complementari.*
*Una serve per compattare i programmi in DB III per renderli più veloci, l'altra permette invece di riportarli al loro stato naturale perché siano facilmente leggibili dagli umani.*
*Va inoltre ricordato che, dal momento che i programmi in DB III sono dei normali file ASCII, si possono utilizzare queste routine anche per altri scopi.*

È un programma scritto in Turbo Basic, ed è stato da me ideato per uno scopo ben preciso: rendere un programma scritto in Dbase III più veloce!

Mi spiego meglio: tutti sappiamo che il DBase III è un interprete, e perciò è costretto a scandire ad una ad una tutte le righe del programma, perdendo un grande tempo ad interpretare righe dove vi sono molti spazi bianchi (programmi cioè scritti secondo la programmazione Top-Down, o strutturata), e istruzioni scritte per intero. Leggendo il libro «DBase III: Tecniche di programmazione avanzate» di Joseph David Carrabis edito dalla Jackson, ho scoperto che, come scritto alle pagine 16-17, eliminando gli spazi bianchi di cui sopra, e riducendo tutte le istruzioni alla lunghezza di quattro caratteri si ha un incremento notevole nei tempi di esecuzione del programma (ovviamente tutto questo non ha senso se avete un compilatore con cui appunto compilare il programma!).

Come avrete già capito quindi, il programma non fa altro che leggere il

```
token$ = FNGetToken$("token string, a short one", " ,")
 while (token$ <> "")
   print token$
   token$ = FNGetToken$("", " ,")
 wend
```

*Sequenza per usare correttamente la funzione FNGet-ToKen$.*

```
1
' Optimizer DBase III
' (C) Copyright 1988 By Daniele Bufarini

%True = -1
%False = 0
Num = 1
Sep$ = " ": word$ = "": InputFile$ = "": OutputFile$ = ""
dim Stat$(13)

Def FNStrSpn(InString$, Separater$)
static LenInString, LenSeparater, StartFound, IndexSeparater, IndexInString
static ChTemp$
      LenInString = Len(InString$)
      LenSeparater = Len(Separater$)
      for IndexInString = 1 to LenInString
            ChTemp$ = Mid$(InString$, IndexInString, 1)
            StartFound = %False
            for IndexSeparater = 1 to LenSeparater
                  if (ChTemp$ = Mid$(Separater$, IndexSeparater, 1)) then
                        goto NextChar
                  end if
            next IndexSeparater
            StartFound = %True
            goto EndStrSpn
NextChar:
      Next IndexInString
EndStrSpn:
      if (StartFound) then
            FNStrSpn = IndexInString
      else
            FNStrSpn = %False
      end if
End Def

Def FNStrBrk(String1$, String2$)
static LenString1, IndexString1, StartFound
static ChTemp$
      LenString1 = Len(String1$)
      for IndexString1 = 1 to LenString1
            ChTemp$ = Mid$(String1$, IndexString1, 1)
            if (instr(String2$, ChTemp$)) then
                  StartFound = %True
                  FNStrBrk = IndexString1
                  Exit Def
            end if
      Next IndexString1
      FnStrBrk = %False
End Def

def FNGetToken$(Search$, InSep$)
static TokenIndex1
      if (Search$ = "") then
            Search$ = TokenString$
      else
            TokenIndex2 = 1
            TokenString$ = Search$
      end if
      if (TokenIndex2 >= len(Search$)) then
            FNGetToken$ = ""
            Exit Def
      end if
      TokenIndex1 = FNStrSpn(mid$(Search$, TokenIndex2, len(Search$)), InSep$)
      if (TokenIndex1 = 0) then
            TokenIndex1 = len(Search$)
      else
            TokenIndex1 = TokenIndex1 + TokenIndex2 - 1
      end if
      TokenIndex2 = FNStrBrk(mid$(Search$, TokenIndex1, len(Search$)), InSep$)
      if (TokenIndex2 = 0) then
            TokenIndex2 = len(Search$) + 1
      else
            TokenIndex2 = TokenIndex2 + TokenIndex2 - 1
      end if
      FNGetToken$ = mid$(Search$,TokenIndex1,TokenIndex2 - TokenIndex1)
end def

def FNIsChar(Char$)
static CharAsc
      CharAsc = asc(Char$)
```

**2**

```
        FNIsChar = ((CharAsc >= asc("A")) and (CharAsc <= asc("Z"))) or _
                   ((CharAsc >= asc("a")) and (CharAsc <= asc("z")))
end def

sub GetFileNames(InputFile$,OutputFile$,Seps$)
        cls
        print " Optimizer Dbase III"
        print " Copyright (c) 1988  BY DANIELE BUFARINI"
        print " "
        if (Command$ = "") then
                Input " Input Filename  (return per terminare) ";InputFile$
                If inputfile$ = "" Then Exit Sub
                Input " Output Filename (per default e' LPT1:) ";OutputFile$
                cls
        else
                InputFile$ = FNGetToken$(Command$, Seps$)
                OutputFile$ = FNGetToken$("", "  ")
                print " Input  Filename: ";InputFile$," Output Filename: ";OutputFile$
        end if
ExitGet:
                if Instr(InputFile$,".") = 0 then InputFile$ = InputFile$ + ".prg"
                if (OutputFile$ = "") then OutputFile$ = "LPT1:"
end sub
Call GetFileNames(InputFile$,OutputFile$," "): if InputFile$ = "" then end
open InputFile$ for input as #1
open OutputFile$ for output as #2
 while not eof(1)
   line input #1, word$
   word1$ = UCASE$(FNGetToken$(word$,Seps$))
   while (word1$ <> "")
     if FNIsChar(word1$) = %True then
          Select Case UCASE$(left$(word1$,1))
            Case "A"
              restore AKey
            Case "B"
              restore BKey
            Case "C"
              restore CKey
            Case "D"
              restore DKey
            Case "E"
              restore EKey
            Case "F"
              restore FKey
            Case "G"
              restore GKey
            Case "H"
              restore HKey
            Case "I"
              restore IKey
            Case "J"
              restore JKey
            Case "K"
              restore KKey
            Case "L"
              restore LKey
            Case "M"
              restore MKey
            Case "N"
              restore NKey
            Case "O"
              restore OKey
            Case "P"
              restore PKey
            Case "Q"
              restore QKey
            Case "R"
              restore RKey
            Case "S"
              restore SKey
            Case "T"
              restore TKey
            Case "U"
              restore UKey
            Case "V"
              restore VKey
            Case "W"
              restore WKey
            Case "X"
              restore XKey
```

**3**

```
            Case "Y"
              restore YKey
            Case "Z"
              restore ZKey
          End Select
          read Num
          for i=1 to Num: read Stat$(i)
           if word1$ = Stat$(i) then word1$ = left$(Stat$(i),4)
          next
        end if
        print #2, word1$;" ";: print word1$;" ";
        word1$ = UCASE$(FNGetToken$("",Seps$))
      wend
      print #2, chr$(13): print chr$(13)
    wend
  close

AKey:
      Data 5, ACCEPT, APPEND, ASSIST, AVERAGE, ALTERNATE, ""
BKey:
      Data 1, BROWSE, ""
CKey:
      Data 13, CANCEL, CHANGE, CLEAR, CLOSE, CONTINUE, COUNT, CREATE, CARRY,_
               COLOR, CONFIRM, CONSOLE, COMMAND, CMONTH, ""
DKey:
      Data 8, DELETE, DISPLAY, DEBUG, DECIMALS, DEFAULT, DELETED,DELIMITERS,_
              DEVICE, ""
EKey:
      Data 8, EJECT, ERASE, ESCAPE, EXACT, EXTENDED, ENDCASE, ENDDO, ENDIF, ""
FKey:
      Data 4, FILTER, FIXED, FORMAT, FUNCTION, ""
GKey:
      Data 1, ""
HKey:
      Data 1, HEADING, ""
IKey:
      Data 4, INDEX, INPUT, INSERT, INTENSITY, ""
JKey:
      Data 1, ""
KKey:
      Data 1, ""
LKey:
      Data 3, LABEL, LOCATE, LOWER, ""
MKey:
      Data 4, MODIFY, MEMORY, MARGIN, MONTH, ""
NKey:
      Data 1, ""
OKey:
      Data 1, OTHERWISE, ""
PKey:
      Data 7, PARAMETERS, PRIVATE, PROCEDURE, PUBLIC, PRINT,PROCEDURE,PICTURE,""
QKey:
      Data 1, ""
RKey:
      Data 12, RECALL, REINDEX, RELEASE, RENAME, REPLACE, REPORT, RESTORE,_
               RETURN, RELATION, RECNO, ROUND, RANGE, ""
SKey:
      Data 9, SELECT, STORE, SAFETY, SCOREBOARD, STEP, STRUCTURE, STATUS,_
              SPACE, SUBSTR, ""
TKey:
      Data 1, TOTAL, ""
UKey:
      Data 3, UPDATE, UNIQUE, UPPER, ""
VKey:
      Data 1, ""
WKey:
      Data 1, WHILE, ""

XKey:
      Data 1, ""
YKey:
      Data 1, ""
ZKey:
      Data 1, ""
```

sorgente del programma in DBase III, riscrivendolo con le modifiche opportune.

Due parole sul programma vero e proprio: il suo funzionamento è molto semplice, e l'unica cosa di rilievo da notare, e che potrà essere riutilizzata anche in altri programmi, è la funzione FNGet-Token$, che, data una stringa in input, restituisce di volta in volta, le singole parole che compongono la stringa in output.

La sequenza per usare correttamente la funzione (presa dall'utility PPRINT che la Microsoft distribuisce insieme al suo QuickBasic) è pubblicata nella pagina precedente.

Ho fatto molte prove, anche su programmi che avevo scritto appositamente per degli amici, ed ho notato che Optimizer DBase III si comporta molto bene, dando dei buoni risultati in quanto a velocità di esecuzione. Ritengo inoltre che Optimizer DBase III sia un buon esempio delle potenzialità di quello stupendo linguaggio che è il Turbo Basic (io programmo anche in C, ed ho visto che il Turbo Basic regge molto bene il confronto).

**N.B.** Optimizer DBase III può essere usato in due modi: sia invocando Optimize, e dando poi i nomi del sorgente DBase e del programma in output; sia usando il modo command-line, cioè con la forma Optimize [d:]\percorso\nome-del-file [d:]\percorso\nome-del-file, dove il primo nome è quello del sorgente in input, il secondo quello del file in output.

# Indent DBase

*di Marco Tinari - Roma*

Il programma INDENTDB ha come scopo principale quello di indentare in maniera corretta e veloce applicazioni scritte in DBASE III plus. Per «indentazione» ci si riferisce a quella tecnica mediante la quale la struttura di un programma viene evidenziata attraverso opportuni incolonnamenti — progressivamente spostati rispetto al margine sinistro — delle istruzioni al fine di: aumentare la leggibilità del programma stesso, di semplificare nel corso del progetto la comprensibilità delle scelte fatte, di velocizzare l'individuazione degli errori e, non ultimo, di agevolare —

*È disponibile, presso la redazione, il disco con i programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 249*

**1**

```
*
*     dBASE III indenter
*     By Marco Tinari 1988
*          ver 1.0
*
*
* definizione variabili iniziali
colon = 0                 && numero di spazi iniziali
cuca = 0                  && numero di DO..CASE attivi
nuca = 0                  && numero di DO..CASE totali
cuwh = 0                  && numero di DO..WHILE attivi
nuwh = 0                  && numero di DO..WHILE totali
cuif = 0                  && numero di IF..[ELSE]..ENDIF attivi
nuif = 0                  && numero di IF..[ELSE]..ENDIF totali
linee = 0                 && numero di linee totali
nucar = 0                 && numero caratteri nel file
* stampa intestazione iniziale
PRINT "  ********************************"
PRINT ""
PRINT "      dBASE III indenter"
PRINT "   By Marco Tinari 1988"
PRINT "        ver 1.0"
PRINT ""
PRINT "  ********************************"
* prendi il drive
drive$ = ""
PRINT : PRINT
INPUT "Quale drive (ad es. A) "; drive$
IF LEN(drive$) < 1 THEN BEEP: PRINT "Drive non permesso !": GOTO drive
drive$ = drive$ + ":"
* prendi il nome del file da indentare
PRINT : PRINT
INPUT "Introdurre nome del file (senza .PRG) ", nomefile$
IF LEN(nomefile$) > 8 THEN nomefile$ = LEFT$(nomefile$, 8)
* chiede gli spazi per rientro
rientro$ = ""
PRINT : PRINT
INPUT "Quanti spazi per rientro "; rient
IF rient < 0 THEN PRINT "Introdurre un numero positivo o 0 !": GOTO rientro
* apre files di lavoro ed inizia la procedura
OPEN drive$ + nomefile$ + ".prg" FOR INPUT AS 1
OPEN drive$ + "temp.tmp" FOR OUTPUT AS 2
```

**3**

```
    END IF
    linee = linee + 1
WEND
CLOSE 1, 2
*
* controlla che tutto sia O.K.
IF cuca > 0 THEN
    BEEP: PRINT
    PRINT "ATTENZIONE...una struttura DO CASE non e' stata chiusa !"
END IF
IF cuwh > 0 THEN
    BEEP: PRINT
    PRINT "ATTENZIONE...una struttura DO WHILE non e' stata chiusa !"
END IF
IF cuif > 0 THEN
    BEEP: PRINT
    PRINT "ATTENZIONE...una struttura IF non e' stata chiusa !"
END IF
IF cuca > 0 OR cuif > 0 OR cuwh > 0 THEN GOTO shutdown
*
* ristabilisci nome e togli file di lavoro
KILL drive$ + nomefile$ + ".prg"
NAME drive$ + "temp.tmp" AS drive$ + nomefile$ + ".prg"
*
*stampa statistica sul file
PRINT : PRINT
PRINT "Nel file "; nomefile$; " sono presenti:"
PRINT linee; "linee di programma"
IF nuca > 0 THEN PRINT nuca; "DO CASE"
IF nuwh > 0 THEN PRINT nuwh; "DO WHILE"
IF nuif > 0 THEN PRINT nuif; "IF"
IF nucar > 4096 THEN
    BEEP: PRINT
    PRINT "ATTENZIONE! il file ha superato i 4096 caratteri"
    PRINT "e non e' piu' gestibile con l'editor del dBASE III."
END IF
END
*
errl:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una CASE senza aver dichiarato DO CASE"
GOTO shutdown
*
*
err2:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una OTHERWISE senza aver dichiarato DO CASE"
GOTO shutdown
```

**2**

```
WHILE NOT EOF(1)
        flag = 0
        LINE INPUT #1, linea$
        CALL ltrim(linea$)
        CALL rtrim(linea$)
        b$ = linea$                     ' b$ stringa di lavoro
        CALL ucase(b$)
        IF MID$(b$, 1, 4) = "CASE" THEN
                flag = 1
                IF cuca < 1 THEN GOTO err1
                linea$ = SPACE$(colon - rient) + linea$
        END IF
        IF MID$(b$, 1, 9) = "OTHERWISE" THEN
                flag = 1
                IF cuca < 1 THEN GOTO err2
                linea$ = SPACE$(colon - rient) + linea$
        END IF
        IF MID$(b$, 1, 4) = "ELSE" THEN
                flag = 1
                IF cuif < 1 THEN GOTO err3
                linea$ = SPACE$(colon - rient) + linea$
        END IF
        IF MID$(b$, 1, 7) = "ENDCASE" THEN
                flag = 1
                IF cuca < 1 THEN GOTO err4
                colon = colon - rient
                linea$ = SPACE$(colon) + linea$
                cuca = cuca - 1
        END IF
        IF MID$(b$, 1, 5) = "ENDDO" THEN
                flag = 1
                IF cuwh < 1 THEN GOTO err5
                colon = colon - rient
                linea$ = SPACE$(colon) + linea$
                cuwh = cuwh - 1
        END IF
        IF MID$(b$, 1, 5) = "ENDIF" THEN
                flag = 1
                IF cuif < 1 THEN GOTO err6
                colon = colon - rient
                linea$ = SPACE$(colon) + linea$
                cuif = cuif - 1
        END IF
        IF MID$(b$, 1, 3) = "IF " THEN
                flag = 1
                linea$ = SPACE$(colon) + linea$
                colon = colon + rient
                cuif = cuif + 1
                nuif = nuif + 1
        END IF
        IF MID$(b$, 1, 3) = "DO " THEN
                p = 3
                WHILE MID$(b$, p, 1) = " "
                        p = p + 1
                WEND
                IF MID$(b$, p, 4) = "CASE" THEN
                        flag = 1
                        linea$ = SPACE$(colon) + linea$
                        colon = colon + rient
                        cuca = cuca + 1
                        nuca = nuca + 1
                END IF
                IF MID$(b$, p, 5) = "WHILE" THEN
                        flag = 1
                        linea$ = SPACE$(colon) + linea$
                        colon = colon + rient
                        cuwh = cuwh + 1
                        nuwh = nuwh + 1
                END IF
        END IF
        IF flag = 1 THEN
                nucar = nucar + LEN(linea$)
                PRINT linea$
                PRINT #2, linea$
        ELSE
                nucar = nucar + LEN(linea$) + colon
                PRINT #2, SPACE$(colon); linea$
                PRINT SPACE$(colon) + linea$
```

**4**

```
err3:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una ELSE senza aver dichiarato IF"
GOTO shutdown

err4:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una ENDCASE senza aver dichiarato DO CASE"
GOTO shutdown

err5:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una ENDDO senza aver dichiarato DO WHILE"
GOTO shutdown

err6:
PRINT : PRINT : PRINT
BEEP
PRINT "E' stata trovata una ENDIF senza aver dichiarato IF"
GOTO shutdown

shutdown:
CLOSE 1, 2
PRINT : PRINT : PRINT
PRINT "A causa dell'errore occorso l'originale e' stato lasciato inalterato"
PRINT "Premere un tasto per terminare."
KILL drive$ + "temp.tmp"
WHILE INKEY$ = ""
WEND
END

' subroutine per levare gli spazi iniziali
SUB ltrim (x$) STATIC
        WHILE LEFT$(x$, 1) = " "
                x$ = RIGHT$(x$, LEN(x$) - 1)
        WEND
END SUB

' subroutine per levare gli spazi in coda
SUB rtrim (x$) STATIC
        WHILE RIGHT$(x$, 1) = " "
                x$ = LEFT$(x$, LEN(x$) - 1)
        WEND
END SUB

' subroutine per trasformare i caratteri in maiuscoli
SUB ucase (x$) STATIC
        lenght = LEN(x$)
        IF lenght > 0 THEN
                FOR i = 1 TO lenght
                        ch = ASC(MID$(x$, i, 1))
                        IF ch > 96 AND ch < 127 THEN MID$(x$, i, 1) = CHR$(ch - 32)
                NEXT i
        END IF
END SUB
```

quando occorre — la manutenzione della vostra applicazione.

Tutto ciò è di fondamentale importanza per una corretta programmazione strutturata e di assoluta necessità per chi (come il sottoscritto) un po' per hobby, un po' per lavoro è costretto a passare ore davanti a strutture IF, DO CASE, DO WHILE, etc. tipiche del DBASE III plus; con tale utility, infatti, individuare un ENDIF od un ENDCASE mancante è questione di secondi, come pure il sapere da quante linee è composta la vostra applicazione ed il numero ed il tipo delle strutture presenti in essa.

Per ottenere tutto ciò è sufficiente, dopo aver compilato il programma in un file di tipo EXE, digitare:

INDENTDB

se vi trovate in DOS, oppure:

!INDENTDB

se vi trovate in ambiente DBASE III plus. Dopo pochi secondi vi compare l'intestazione e la richiesta del drive nel quale risiede l'applicazione da indentare; potete quindi battere la lettera corrispondente (A,B,...) e, una volta digitato RETURN, dovete immettere il nome della vostra applicazione omettendo il suffisso .PRG. A tale domanda non è possibile immettere, oltre al nome subdirectory. Come ultima cosa dovete informare il programma di quante colonne deve rientrare ogni qualvolta si trova davanti ad una nuova struttura (ottimi risultati si ottengono rispondendo 5). Subito dopo vedrete scorrere sul video le linee della vostra applicazione, contemporaneamente memorizzate sul floppy precedentemente indicato.

Se tutto è andato OK e se la vostra applicazione è scritta in maniera corretta (se cioè ad ogni IF corrisponde un ENDIF, ad ogni DO WHILE un ENDDO, etc.) il programma terminerà dandovi delle informazioni statistiche sul risultato finale, come: il numero delle linee e delle strutture presenti, dicendovi anche il tipo, e se, il file eccede i fatidici 4096 caratteri, sarete avvisati dell'impossibilità di usare l'editor del DBASE; qualora vi trovaste in tale possibilità dovrete ricorrere ad altri strumenti per l'editing tipo il Kedit, Wordstar (opzione N) oppure, se vi trovate in WINDOWS, il Notepad.

Se invece il programma trova una struttura non chiusa od altre scorrettezze (ad esempio una OTHERWISE senza prima aver trovato DO CASE) vi avvertirà con un sonoro beep e relativo messaggio sul video, informandovi anche che la vostra applicazione non è stata indentata a causa dell'errore trovato.

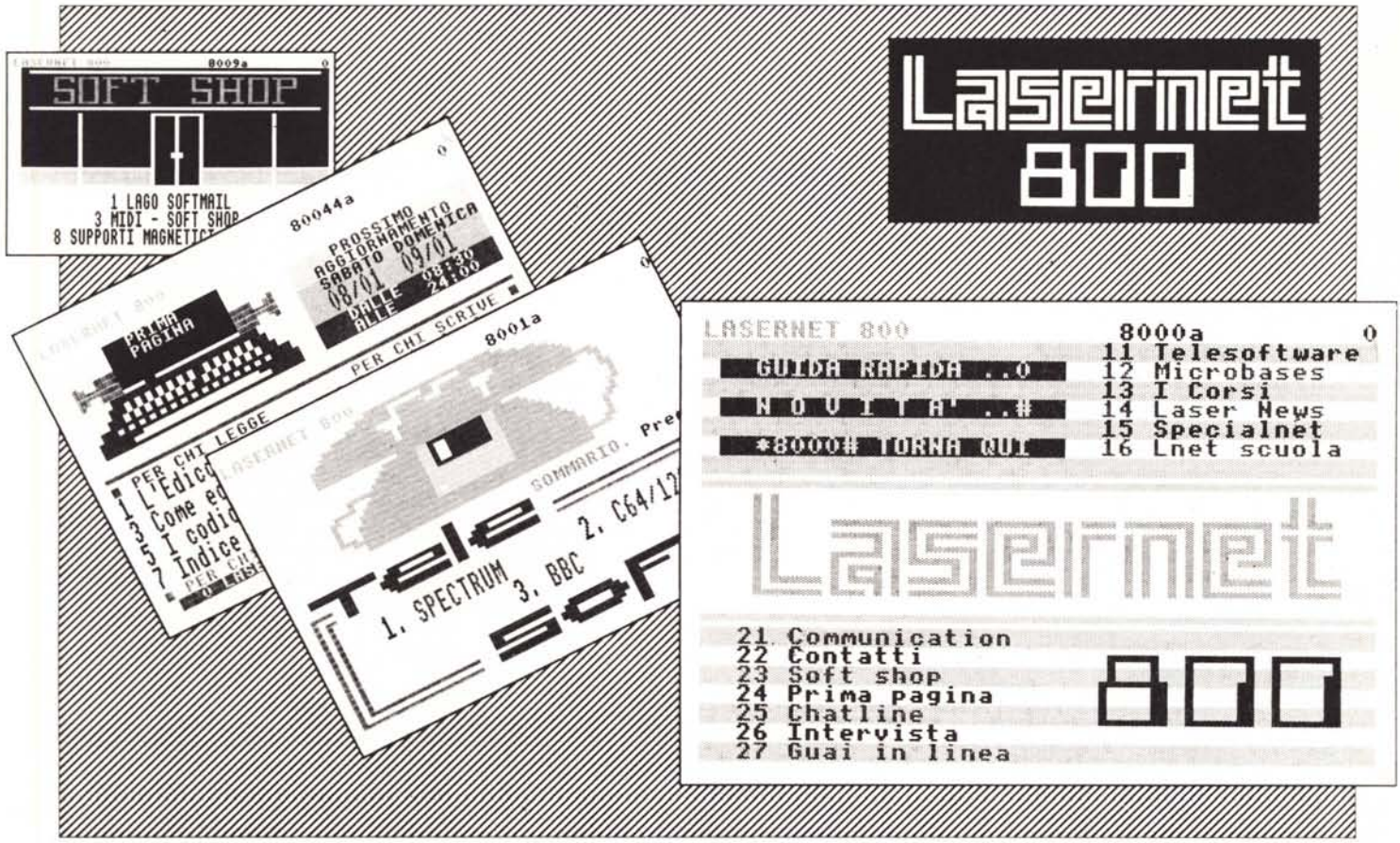
Per concludere vorrei far notare che il programma è stato sviluppato in Quick Basic 2.0, che è sprovvisto dell'istruzione SELECT CASE, e quindi sono stato costretto ad usare una serie di IF... ENDIF piuttosto che una sola struttura SELECT CASE. Chi ha buona volontà può apportare la miglioria e magari farmi sapere i risultati attraverso MC LINK (il mio codice è MC3712). MC

**Bibliografia**

Microsoft Quick Basic Compiler 2.0, Microsoft Corp. Batini - Aiello, Metodologie di analisi e progetto di programmi Facoltà di Ingegneria Roma

Lavorare in dBASE III PLUS, Ashton-Tate
Programmare in dBASE III PLUS, Ashton-Tate
Manuale di Infomatica, Calderini.

# UN'EMOZIONE DA 1200 BIT AL SECONDO




- La potenza di una banca dati, la dinamica di un quotidiano.
- L'unico servizio telematico italiano con le notizie in tempo reale sul mondo dell'informatica.
- Il solo accessibile tramite la rete nazionale Videotel presente in più di 67 distretti telefonici (oltre 1000 comuni!).
- Con LASERNET 800 potrai caricare programmi in TELESOFTWARE, chiacchierare in diretta con tutta Italia sulle CHATLINES, editare un tuo spazio personale su PRIMA PAGINA, leggere le notizie più interessanti di LASER NEWS e migliorare la tua programmazione con i nostri corsi.
- Oltre 5000 pagine consultabili 24 ore su 24.
- Il nostro servizio ti costa ogni giorno meno della metà di un quotidiano!

# PROVALA!

Per avere maggiori informazioni sul servizio compila il tagliando e spediscilo a:
LASERNET 800 - Via G.Modena, 9
20129 Milano - Tel. 02/200.201

**Desidero ricevere maggiori informazioni su LASERNET 800** MC

**Cognome**..... **Nome**.....
**Via**.....
**Città**..... **Prov**.....
**CAP**..... **Tel**.....
**Data di nascita**...../...../.....
**Il mio computer é un:**

| | | | |
|---|---|---|---|
| Commodore | ☐ 64 | ☐ 128 | ☐ Amiga |
| ☐ MSX | ☐ BBC | ☐ Atari ST | ☐ PC |
| Spectrum | ☐ 48K | ☐ Plus | ☐ 128 |

☐ **Ho già un adattatore telematico**