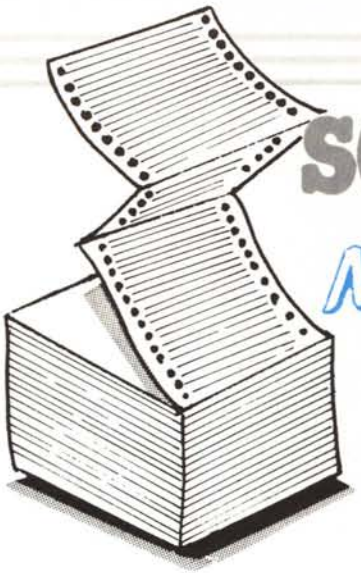

software MS-DOS

a cura di Valter Di Dio

Per questa puntata faremo contenti tutti, nel senso che pubblichiamo tre programmini, o più esattamente tre Tool, scritti nei tre linguaggi più usati del momento: il Basic, il Pascal e il C.

Il programma in Basic serve per listare i programmi in Basic in modo più leggibile. A tal proposito vorrei precisare al lettore che la differenza tra il Pascal, il C e il Basic non è solo nella chiarezza del listato; oltre al fatto che sono compilatori (e non è poco!) dove mettiamo le variabili locali, le label, la ricorsività e (non ultimo) un editor a tutto schermo?

Il programma in Pascal permette di generare delle scritte scorrevoli e lo pubblichiamo volentieri soprattutto per far vedere come vanno scritti gli INLINE per facilitare il controllo dei codici a chi li copia o semplicemente per far capire cosa fa. Più sostanziose le routine in C che forniscono una buona libreria di tool matematici e di gestione delle stringhe.

Pretty per Basic MS-DOS

di Pietro Vigone - Borgo Vercelli (VC)

Quei pochi sfortunati che, come il sottoscritto, non dispongono dei quattrini necessari per l'acquisto dei costosissimi compilatori C, Pascal, macroassembler e così via, di cui le pagine di MC mostrano interessantissimi esempi di utilizzo, non possono che rassegnarsi a scrivere i loro programmi con il vecchio e farraginoso Basic (ma come è poi possibile che abbiano tutti tanti soldi? mah!).

*Si prega il Sig. **Felice Sobrero**, autore del programma «Accesso a nuovi comandi» pubblicato sul numero 65 di MCmicrocomputer, di mettersi in contatto con la redazione.*

Uno dei difetti dell'umile e giustamente denigrato linguaggio di cui sopra è senza dubbio l'illeggibilità dei listati che produce: una sola linea di programma che occupa tre o quattro righe, parole troncate a metà, pagine dense di caratteri... Nessun paragone con l'aspetto leggero e «arioso» di un sorgente in linguaggio C.

Per sopperire come possibile a questa carenza, ho realizzato il programma Pretty, che legge un file Basic salvato su disco in formato ASCII e lo stampa dopo averlo formattato in modo che sia più «bello».

Precisamente, Pretty compie le seguenti operazioni:

— giustifica a destra i numeri di linea;

— stampa una sola istruzione per riga;

— evidenzia i commenti con caratteri in grassetto;

— indenta appropriatamente i cicli FOR, WHILE e le istruzioni IF: tutte le istruzioni comprese tra un FOR e un NEXT, o tra un WHILE e un WEND, sono stampate incolonnate quattro spazi più a destra; THEN e ELSE vengono scritti sotto all'IF corrispondente, spostati di due spazi, e se sono seguiti da istruzioni queste sono stampate una per linea e incolonnate quattro spazi a destra dell'IF. Se queste strutture sono nidificate, anche l'indentamento è realizzato congruentemente.

È fin d'ora evidente che se il programma da listare non è corretto, e contiene ad esempio istruzioni FOR senza il corrispondente NEXT, anche la stampa prodotta darà risultati diversi da quelli attesi; ciò non significa che Pretty serve solo per programmi funzionanti, infatti ha la sua utilità anche l'evidenziazione dell'errore; e il programmatore disattento pagherà la sua distrazione con il consumo di un po' di carta.

Per utilizzare Pretty, occorre salvare il programma da listare sul drive di default con il nome «P» e in formato ASCII [SAVE "P", A], poi dare RUN "PRETTY"; compare un prompt; dopo aver verificato che la stampante sia pronta, premere CR. Le linee del programma da listare compaiono sullo schermo (non formattate) e contemporaneamente vengono stampate.

Descrizione del programma

100-270: si eseguono varie inizializzazioni. Tutte le variabili numeriche sono dichiarate intere; sono definite le funzioni pseudo-booleane FNDIGIT e FNALPHA, che danno un valore vero se il primo carattere del loro argomento e rispettivamente una cifra o un carattere alfabetico maiuscolo, e la funzione stringa FNTL$ che restituisce il suo argomento, sempre di tipo stringa, privato del primo carattere. Il file P.BAS viene aperto come sequenziale, input.

270-300: si carica in LIN$ una linea del programma e la si stampa sul video.

310-340: stampa del numero di linea, giustificato a destra mediante una RSET.

360-830: ricerca delle parole chiave. Scartati eventuali spazi, mediante la subroutine WORD si isola una keyword.

390-470: REM (o equivalente apice semplice). COMMENT indica il carattere di LIN$ in cui comincia il commento: serve per il caso di un commento che segue le istruzioni, indicato da un apice semplice. Mediante la subroutine EMIT (1430) si stampano quindi prima la parte di LIN$ fino a COMMENT e poi, con carattere emphatized, la successiva.

480-490: WEND. Il contatore INDENT, che indica appunto il numero di spazi da lasciare bianchi, è decrementato di quattro.

500-550: NEXT. Rispetto al precedente vi è la complicazione che un NEXT con una lista di variabili separate da virgole chiude tanti cicli quante sono le variabili. Per aggiornare INDENT lo si decrementa di quattro per ogni virgola posta tra il NEXT e i due punti o la fine della linea.

560-570: stampa dell'indentamento. Si esegue a questo punto per far sì che il NEXT sia scritto in colonna al relativo FOR e così via: se fosse eseguito all'inizio, il NEXT sarebbe spostato ancora a destra del FOR, se fosse eseguito alla fine il FOR risulterebbe incolonnato alle istruzioni che lo seguono e spostato a destra rispetto al NEXT.

580-600: FOR e WHILE. INDENT è incrementato di quattro.

600-660: IF. Si incrementano i contatori INDENT e IFNESTING: quest'ultimo indica la nidificazione delle istruzioni condizionali ed è inizializzato a zero in linea 290. Si cerca la posizione di THEN o GOTO all'interno di LIN$; in linea 640 si evita il caso, pur sempre possibile, che nell'espressione dell'IF ci sia una stringa contenente qualche THEN o GOTO, evento questo che viene gestito dalla subroutine STRING

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. 265.

Pretty

```
100 REM ******
110 REM ****** PRETTY
120 REM ******
130 REM ****** pietro vigone 1987
140 REM ******
150 CLS:
    KEY OFF
160 LOCATE 10,28:
    PRINT "PRETTY"
170 LOCATE 20,12:
    PRINT "Accertarsi che la stampante sia pronta e premere ,return."
180 DEFINT A-Z
190 AP$=CHR$(34):
    EMPH$=CHR$(27)+"E":
    NORM$=CHR$(27)+"F"
200 DEF FNDIGIT(C$)=C$>="0" AND C$<":":
    DEF FNALPHA(C$)=C$>="A" AND C$<"["
210 DEF FNTL$(L$)=RIGHT$(L$,LEN(L$)-1)
220 OPEN "P.DAS" FOR INPUT AS #1
230 I$=INKEY$:
    IF I$<>CHR$(13)
      THEN 230
240 LOCATE 20:
    PRINT SPACE$(80)
250 INDENT=0
260 REM ******
270 REM ****** ciclo principale
280 LINE INPUT #1,LIN$:
    PRINT LIN$
290 IFNESTING=0
300 REM ******
310 REM ****** stampa numero di linea
320 GOSUB 1220
330 L$=SPACE$(5):
    RSET L$=W$
340 CONT=6:
    LPRINT L$;" ";
350 REM ******
360 REM ****** cerca parole chiave
370 IF LEFT$(LIN$,1)=" "
      THEN
        LIN$=FNTL$(LIN$):
        GOTO 370
380 L$=LIN$:
    GOSUB 1140
390 IF W$<>"REM" AND LEFT$(LIN$,1)<>"'"
      THEN 480
400 REM ****** commento
410 COMMENT=1
420 L$=LEFT$(LIN$,COMMENT-1):
    GOSUB 1430
430 LPRINT EMPH$;
440 L$=MID$(LIN$,COMMENT,255)
450 GOSUB 1430
460 LPRINT NORM$;
470 GOTO 970
480 REM ****** istruzione WEND
490 IF W$="WEND"
      THEN
        INDENT=INDENT-4:
        GOTO 560
500 IF W$<>"NEXT"
      THEN 560
510 REM ****** istruzione NEXT
520 COMMA=0
530 INDENT=INDENT-4
540 COLON=INSTR(LIN$+":",":"):
    COMMA=INSTR(COMMA+1,LIN$+",",",")
550 IF COMMA<COLON
      THEN 530
560 REM ****** stampa indentamento
570 CONT=CONT+INDENT:
    LPRINT SPC(INDENT);
580 REM ****** istruzioni FOR e WHILE
590 IF W$="FOR" OR W$="WHILE"
      THEN
        INDENT=INDENT+4:
        GOTO 640
600 IF W$<>"IF"
      THEN 640
610 REM ****** istruzione IF
620 IFNESTING=IFNESTING+1:
    INDENT=INDENT+4
630 TCASE=INSTR(LIN$+" THEN "," THEN "):
    TER=INSTR(LIN$+" GOTO "," GOTO "):
    IF TER<TCASE
      THEN
        TCASE=TER
640 APICES=INSTR(LIN$+AP$,AP$):
    IF APICES<TCASE
      THEN
        GOSUB 1050:
        GOTO 630
650 L$=LEFT$(LIN$,TCASE-1)
660 GOSUB 1430
670 REM ****** clausola THEN
680 PRECELSE=0
690 LPRINT :
    LPRINT SPC(4+INDENT);MID$(LIN$,TCASE+1,4);
700 LIN$=MID$(LIN$,TCASE+6,255)
710 IF LEFT$(LIN$,1)=" "
      THEN
        LIN$=FNTL$(LIN$):
        GOTO 710
720 IF NOT FNDIGIT(LIN$)
      THEN
        LPRINT:
        CONT=6:
        LPRINT SPC(CONT);:
        GOTO 360
730 GOSUB 1220:
    LPRINT " ";W$;:
    GOTO 840
740 REM ****** clausola ELSE
750 IF PRECELSE
      THEN
        IFNESTING=IFNESTING-1:
        INDENT=INDENT-4
      ELSE
        PRECELSE=1
760 L$=LEFT$(LIN$,FCASE-1)
770 GOSUB 1430
780 LPRINT :
    LPRINT SPC(4+INDENT);MID$(LIN$,FCASE,4);
790 LIN$=MID$(LIN$,FCASE+5,255)
800 IF LEFT$(LIN$,1)=" "
      THEN
        LIN$=FNTL$(LIN$):
        GOTO 800
810 IF NOT FNDIGIT(LIN$)
      THEN
        LPRINT:
        CONT=6:
        LPRINT SPC(CONT);:
        GOTO 360
820 GOSUB 1220:
    LPRINT " ";W$;
830 REM ******
840 REM ****** cerca separatori
850 COLON=INSTR(LIN$+":",":"):
    APICES=INSTR(LIN$+AP$,AP$):
    COMMENT=INSTR(LIN$+"'","'")
860 FCASE=1
870 FCASE=(INSTR(FCASE,LIN$+" ELSE ","ELSE ")):
    IF FCASE>1
      THEN
        IF FNALPHA(MID$(LIN$+" ELSE ",FCASE-1,1))
          THEN 870
880 IF COMMENT AND COMMENT<COLON AND COMMENT<APICES AND COMMENT<FCASE
      THEN 420
890 IF APICES AND APICES<COLON AND APICES<FCASE
      THEN
        GOSUB 1050:
        GOTO 840
900 IF FCASE<COLON
      THEN 740
910 REM ******
920 REM ****** stampa fino alla istruzione successiva
930 L$=LEFT$(LIN$,COLON):
    LIN$=MID$(LIN$,COLON+1,255)
940 GOSUB 1430
950 IF LIN$<>""
      THEN
        LPRINT :
        CONT=6:
        LPRINT SPC(CONT);:
        GOTO 360
960 REM ******
970 REM ****** fine istruzione
980 IF EOF(1)
      THEN 1010
990 LPRINT :
    INDENT=INDENT-4*IFNESTING:
    GOTO 270
1000 REM ******
1010 REM ****** fine listato
1020 CLOSE #1:
     LPRINT CHR$(12)
1030 END
1040 REM ******
1050 REM ****** subroutine STRING
1060 REM ****** stampa una stringa compresa tra apici
1070 L$=LEFT$(LIN$,APICES-1):
     LIN$=RIGHT$(LIN$,LEN(LIN$)-APICES+1)
1080 GOSUB 1430
1090 APICES=INSTR(2,LIN$,AP$)
1100 W$=LEFT$(LIN$,APICES):
     LIN$=MID$(LIN$,APICES+1,255)
1110 GOSUB 1360
1120 RETURN
1130 REM ******
1140 REM ****** subroutine WORD
1150 REM ****** isola una sequenza di car. alfabetici
1160 W$=""
1170 WHILE FNALPHA(L$)
1180      W$=W$+LEFT$(L$,1):
          L$=FNTL$(L$)
1190 WEND
1200 RETURN
1210 REM ******
1220 REM ****** subroutine LINENUM
1230 REM ****** isola un numero di linea
1240 W$=""
1250 WHILE FNDIGIT(LIN$)
1260      W$=W$+LEFT$(LIN$,1):
          LIN$=FNTL$(LIN$)
1270 WEND
1280 RETURN
1290 REM ******
1300 REM ****** subroutine LEXEME
1310 REM ****** isola una sequenza di car. fino al primo spazio
1320 SPACE=INSTR(L$+" "," ")
1330 W$=LEFT$(L$,SPACE):
     L$=MID$(L$,SPACE+1,255)
1340 RETURN
1350 REM ******
1360 REM ****** subroutine COMPOSE
1370 REM ****** stampa evitando troncamenti
1380 IF CONT+LEN(W$)<=80
       THEN 1400
1390 LPRINT :
     CONT=6+INDENT:
     LPRINT SPC(CONT);
1400 CONT=CONT+LEN(W$):
     LPRINT W$;
1410 RETURN
1420 REM ******
1430 REM ****** subroutine EMIT
1440 REM ****** stampa parte di uno statement
1450 WHILE L$<>""
1460      GOSUB 1300:
          GOSUB 1360
1470 WEND
1480 RETURN
```

(1430). In 650-660 la parte di LIN$ fino al THEN escluso è stampata.

670-730: THEN (o GOTO). Il THEN o GOTO è stampato sotto all'IF, spostato di due caratteri. Dopo aver scartato eventuali spazi, se seguono istruzioni si va a capo e si ritorna all'individuazione delle parole chiave, altrimenti segue un numero di linea e lo si stampa sulla stessa riga.

740-820: ELSE. Vi sono due casi: se l'ELSE è preceduto da un THEN, va scritto incolonnato a esso, se è preceduto da un altro ELSE, va scritto quattro spazi a sinistra. La discriminazione è eseguita in linea 750, in base alla variabile PRECELSE. FCASE indica il punto in cui si trova ELSE all'interno di LIN$; la parte precedente questo punto è stampata, poi si stampa l'ELSE: le linee 800-820 corrispondono alle 710-730 per il THEN.

750-900: ricerca dei separatori. Le istruzioni possono essere separate dai due punti oppure da un ELSE; però se i due punti o l'ELSE sono dopo un apice singolo o compresi tra doppi apici vanno trattati diversamente, inoltre l'ELSE deve essere seguito da uno spazio e preceduto da un carattere non alfabetico (infatti il Basic accetta anche costrutti del tipo "IF A=B THEN 20ELSE 30").

920-950: stampa fino ai due punti.

970-990: fine linea. Elimina gli indentamenti causati da istruzioni IF.

1010-1030: fine file da listare. Il programma produce un form feed e si ferma.

1050-1130: subroutine STRING. Stampa prima la parte di LIN$ fino al primo doppio apice, poi quella contenuta tra apici, inclusi questi. Si evita per quanto possibile di troncare la costante stringa.

1140-1210: subroutine WORD. Isola una sequenza di caratteri maiuscoli.

1220-1290: subroutine LINENUM. Isola un numero di LIN$.

1300-1350: subroutine LEXEME. Isola una sottostringa fino al primo spazio.

1360-1420: subroutine COMPOSE. Mediante la variabile CCNT, che conteggia i caratteri già stampati, si evitano troncamenti.

1430-1480: subroutine EMIT. Usa LEXEME e COMPOSE per stampare.

Scritte scorrevoli

di Giovanni Benintende - Leonforte

Questa routine, scritta in Turbo Pascal, permette di far passare una frase attraverso una riga qualsiasi del video. Il passaggio avviene pixel per pixel e non per caratteri interi. È ovvio che, per questo, è necessaria la presenza della scheda grafica. Osservando il listato, si può notare che è composto dal programma principale e dalla procedura Scorri.

All'interno di questa è presente la procedura Slitta. Quest'ultima contiene la routine che consente lo scorrimento di una immagine per pixel. Per ottenere una discreta velocità di scorrimento è stato necessario scrivere la routine in codice macchina e inserirla in una istruzione Inline.

La procedura Scorri accetta i parametri, stampa uno per uno i caratteri della frase, e calcola gli indirizzi da passare alla procedura Slitta.

È possibile isolare la procedura Scorri (con la Slitta al suo interno) dal programma principale per poi utilizzarla in qualsiasi altro programma scritto in Turbo Pascal, a patto che nel programma (o nella procedura) di appartenenza venga dichiarato il tipo Str255 = String[255].

Per usarla sarà poi sufficiente chiamarla con i parametri a proprio piacimento come dall'esempio, purché questi siano validi. Una curiosità: passando il valore 7 per NumPixel anziché 8, i caratteri verranno più compatti, permettendo di ottenere una densità di oltre 90 colonne per riga.

Compute

di Ernesto Ruffini - Milano

Vi propongo una routine, scritta in C, che ritengo utile per divertirsi a "giochicchiare" coi numeri, passatempo mio preferito.

Il problema era: come far definire dall'utente una funzione (nel senso matematico del termine) per poi sfruttarla? In Basic interpretato il problema è facilmente sormontabile con opportuno Merge, ma il C è per definizione da compilare, e quindi bisogna avere un piccolo interprete in grado di capire una funzione matematica. È questo lo scopo di Macro.C (non chiedetemi perché l'ho chiamato così: non lo so nemmeno io). L'idea mi è venuta da un libro di Lecarme-Nebut, «Pascal» edito dalla McGraw-Hill, di cui sconsiglio vivamente la lettura e che a tutt'oggi è l'unico libro di informatica di cui non ho capito il senso. Alle pagine 240 e seguenti si trova un «Calcolatore da tavolo», di per sé assolutamente incompleto e per di più errato in alcune cose, ma con due funzioni, espressione e valore (che da me si chiama fattore) che ho tradotto in C, fornendole di tutto il resto che serviva.

Spiegare come il tutto funziona non è per nulla agevole: basta dire che il tutto va in modo ricorsivo per cominciare a innervosirsi e a perdere la testa, ma andiamo avanti lo stesso: l'unica funzione che viene richiamata da main() o da chi per lei è calcola(), a cui vengono passati: la stringa che contiene la funzione (ad esempio "2*cos(1)/2.0+1"), il carattere dal quale si deve far partire il calcolo (di solito 0, ma se si mette 2, nel nostro esempio verrà calcolata solo "cos(1)/2.0+1") e infine un puntatore a carattere che conterrà il codice di un even-

Scritte scorrevoli

```
Program EsempioDiScorrimento;
Type Str255=String[255];

Procedure Scorri(Colonna,Riga,Larghezza,Durata: Integer; Frase: Str255);
Var Indice,Lunghezza,IndirizzoIniziale,IndirizzoFinale: Integer;

Procedure Slitta(NumPixel,Inizio,Fine,Tempo: Integer);
Begin
Inline                  { - Scivolamento di  NumPixel - }
($1E                    {       Push Ds                 }
/$B8/$00/$B0            {       Mov Ax,$B800            }
/$8E/$D8                {       Mov Ds,Ax               }
/$8B/$1E/Inizio         {       Mov Di,[Bp+Inizio]      }
/$8B/$9E/NumPixel       {       Mov Bx,[Bp+NumPixel]    }
/$53                    {  L6: Push Bx                  }
/$8B/$9E/Fine           {       Mov Bx,[Bp+Fine]        }
/$B4/$00/$00            {       Mov Dx,0                }
/$57                    {  L5: Push Bx                  }
/$B5/$04                {       Mov Ch,4                }
/$57                    {  L4: Push Bx                  }
/$B1/$04                {       Mov Cl,4                }
/$D1/$E2                {  L3: Shl Dx,1                 }
/$D0/$57/$FF            {       Rcl Byte Ptr [Bx-1],1   }
/$83/$D2/$00            {       Adc Dx,0                }
/$81/$C3/$00/$20        {       Add Bx,$2000            }
/$FE/$C9                {       Dec Cl                  }
/$75/$F0                {       Jnz L3                  }
/$8B/$9E/Tempo          {       Mov Bx,[Bp+Tempo]       }
/$57                    {  L2: Push Bx                  }
/$8B/$9E/Tempo          {       Mov Bx,[Bp+Tempo]       }
/$4B                    {  L1: Dec Bx                   }
/$75/$FC                {       Jnz L1                  }
/$5B                    {       Pop Bx                  }
/$4B                    {       Dec Bx                  }
/$75/$F4                {       Jnz L2                  }
/$5B                    {       Pop Bx                  }
/$83/$C3/$50            {       Add Bx,80               }
/$FE/$C0                {       Dec Ch                  }
/$75/$D5                {       Jnz L4                  }
/$5B                    {       Pop Bx                  }
/$4B                    {       Dec Bx                  }
/$3B/$DF                {       Cmp Bx,Di               }
/$75/$CC                {       Jnz L5                  }
/$5B                    {       Pop Bx                  }
/$4B                    {       Dec Bx                  }
/$75/$C0                {       Jnz L6                  }
/$1F)                   {       Pop Ds                  }
End;   {Procedure Slitta}

Begin   {Scorri}
For Indice:=1 to Larghezza Do Frase:=Frase+' ';
Lunghezza:=Length(Frase);
IndirizzoIniziale:=Colonna-1+(Riga-1)*320;
IndirizzoFinale:=Colonna+Larghezza-1+(Riga-1)*320;
For Indice:=1 to Lunghezza Do
    Begin
    GoToXY(Colonna+Larghezza-1,Riga);
    Write(Frase[Indice]);
    Slitta(8,IndirizzoIniziale,IndirizzoFinale,Durata)
    End;   {For}
End;   {Procedure Scorri}

Begin   {Programma Principale}
HiRes;
Scorri(21,10,40,6,'Nel mezzo del cammin di nostra vita  mi ritrovai per '+
                  'una selva oscura.  ch la dritta via era smarrita... ');
TextMode;
End.   {Programma EsempioDiScorrimento}
```

Compute - MACRO.C

```
/* funzione          tipo ritornato   azione                                  */
/* ------------------------------------------------------------------------- */
/* calcola(c[], i, *ch)  double       valuta la stringa s a partire          */
/*                                    dall'iesimo carattere (in genere 0)    */
/*                                    in caso di errore ch contiene il codice */


#ifndef MATHS
#include <maths.h>
#define MATHS 1
#endif

#ifndef CTYPE
#include <ctype.h>
#define CTYPE 1
#endif

extern double varvalore[];

static char err;

#define DIVZERO   1
#define PARENTH   2
#define UNKNOWN   3
#define ILLLOG    4
#define ILLSQRT   5
#define OVERFLOW  6


double calcola(st, i, c)
char st[];
int i;
char *c;
{

  int j;
  double valore=0.0;

  err = 0;
  for (j=0; st[j]!='\0'; j++)
    st[j]=toupper(st[j]);
  j=i;
  espressione(st, &valore, &j);
  *c=err;
  return(valore);

}


espressione(st, valore, i)
char *st;
double *valore;
int *i;
{

  double operandodestro;
  char   operatore;

  if (err)
    return;
  if ((st[*i]=='+') || (st[*i]=='-')) {
    operatore=st[*i];
    (*i)++;
  }
  else
    operatore='+';
  fattore(st, valore, i);
  if (operatore=='-')
    *valore=-*valore;
  while ((st[*i]=='+') || (st[*i]=='-')) {
    operatore=st[*i];
    (*i)++;
    fattore(st, &operandodestro, i);
    if (operatore=='+')
      *valore+=operandodestro;
    else
      *valore-=operandodestro;
  }

}


fattore(st, valore, i)
char st[];
double *valore;
int *i;
{

  double operandodestro;
  char   operatore;

  if (err)
    return;
  termine(st, valore, i);
  while ((st[*i]=='*') || (st[*i]=='/')) {
    operatore=st[*i];
    (*i)++;
    termine(st, &operandodestro, i);
    if (operatore=='*')
      *valore*=operandodestro;
    else
      if (operandodestro!=0)
        *valore/=operandodestro;
      else
        errore(DIVZERO);
  }


}


termine(st, valore, i)
char st[];
double *valore;
int *i;
{

  if (err)
    return;
  if (isalpha(st[*i]))
    controlla(st, valore, i);
  else
    if (isdigit(st[*i]))
      costante(st, valore, i);
  else
    if (st[*i]=='(') {
      (*i)++;
      espressione(st, valore, i);
      if (st[*i]!=')')
        errore(PARENTH);
      else
        (*i)++;
      }
  else
    errore(UNKNOWN);

}


costante(st, valore, i)
char st[];
double *valore;
int *i;
{

  double potenza, per();
  char   operatore;

  if (err)
    return;
  potenza=1.0;
  *valore=0.0;
  while (isdigit(st[*i]))
    *valore=*valore*10.0+st[(*i)++]-'0';
  if (st[*i]=='.') {
    (*i)++;
    for( ; isdigit(st[*i]); potenza*=10.0)
      *valore=*valore*10.0+st[(*i)++]-'0';
  }
  *valore/=potenza;
  if ((st[*i]=='E') || (st[*i]=='e')) {
    (*i)++;
    if ((st[*i]=='+') || (st[*i]=='-'))
      operatore=st[(*i)++];
    else
      operatore='+';
    potenza=0.0;
    while (isdigit(st[*i]))
      potenza=potenza*10.0+st[(*i)++]-'0';
    if (potenza>37.0)
      errore(OVERFLOW);
    else {
      if (operatore=='-')
        potenza=-potenza;
      *valore*=per(10.0, (int)potenza);
    }
  }

}


errore(num)
int  num;
{

  if (err)
    return;
  err = num;

}


compara(s1, punt, s2)
char s1[];
int punt;
char s2[];
{

  int i, lun;
  lun=strlen(s2);
  for (i=0; i<lun; i++)
    if (s1[i+punt]!=s2[i])
      return(0);
  return(1);

}


controlla(st, valore, i)
char st[];
double *valore;
int *i;
{

  double dd, ln(), sqrt(), sin(), cos(), exp();

  if (err)
    return;
  if (compara(st, *i, "LN(")) {
    *i+=3;
    espressione(st, &dd, i);
    if (dd<=0)
      errore(ILLLOG);
    else {
      *valore=ln(dd);
      if (st[*i]!=')')
        errore(PARENTH);
      else
        (*i)++;
    }
  }
```

(continua a pag. 248)

(segue da pagina 247)

```
else
  if (compara(st, *i, "SQRT(")) {
    *i+=5;
    espressione(st, &dd, i);
    if (dd<0)
      errore(ILLSQRT);
    else {
      *valore=sqrt(dd);
      if (st[*i]!=')')
        errore(PARENTH);
      else
        (*i)++;
    }
  }
else
  if (compara(st, *i, "SEN(") || compara(st, *i, "SIN("))
    *i+=4;
    espressione(st, &dd, i);
    *valore=sin(dd);
    if (st[*i]!=')')
      errore(PARENTH);
    else
      (*i)++;
  }
else
  if (compara(st, *i, "COS(")) {
    *i+=4;
    espressione(st, &dd, i);
    *valore=cos(dd);
    if (st[*i]!=')')
      errore(PARENTH);
    else
      (*i)++;
  }
else
  if (compara(st, *i, "EXP(")) {
    *i+=4;
    espressione(st, &dd, i);
    *valore=exp(dd);
    if (st[*i]!=')')
      errore(PARENTH);
    else
      (*i)++;
  }
else
  if ((st[*i]=='X') || (st[*i]=='E') || (st[*i]=='P')) {
    *valore=varvalore[st[*i]-'A'];
    (*i)++;
  }
else
  errore(UNKNOWN);
}
/*
 ERRORI:
 switch(num) {
   case DIVZERO:
     puts("Divisione per zero");
     break;
   case PARENTH:
     puts("Parentesi non bilanciate");
     break;
   case UNKNOWN:
     puts("Operando sconosciuto");
     break;
   case ILLLOG:
     puts("Logaritmo di argomento non positivo");
     break;
   case ILLSQRT:
     puts("Radice quadrata di argomento negativo");
     break;
   case OVERFLOW:
     puts("Overflow");
     break;
```

tuale errore occorso durante il calcolo dell'espressione (se ad esempio scriviamo "3+(4$6)"); il risultato sarà naturalmente il valore dell'espressione. La calcola() oltre ad azzerare errori precedenti e a mettere in maiuscolo la stringa passatale, richiama espressione(), che come quasi tutte le altre funzioni ha tre parametri: la stringa, il valore dell'espressione fino a quel momento e il numero del carattere a cui il calcolo è attualmente fermo, espressione() si preoccupa delle somme e sottrazioni, e chiama fattore() per sapere cosa deve sommare; fattore() si preoccupa di moltiplicare e dividere, e a sua volta chiama termine() per sapere cosa moltiplicare, termine() decide il tutto: se il carattere su cui si trova è un numero, chiede a costante() di esplicitarlo (costante() è una specie di atod(), per intenderci(!?)); se è un carattere scarica su controlla() il lavoro di traduzione; se è una parentesi aperta chiama espressione() e fa ricominciare tutto daccapo, altrimenti si arrende, controlla() si preoccupava dei caratteri, e infatti se c'è SIN(, chiama espressione() per calcolare il segno dell'argomento, se c'è COS(...; in più ha tre costanti: P (pi greco), E (la base dei ln), X (perché le funzioni hanno le variabili indipendenti); i valori di queste costanti li trova un extern double varvalore. Il tutto è naturalmente corredato dall'opportuna funzione errore() che pensa agli strafalcioni.

Anche se non è chiaro non mi interessa, io non lo ripeto con altre parole, al massimo posso ricopiare quello che ho scritto sopra, ma temo qualche attentato, quindi mi astengo.

Alla fine del file si trova, come commento, un pezzo di programma da mettere in main() o in chi per lui per gestire gli errori, che traduce cioè «errore 1» in «occhio che vuoi dividere per zero».

Per completare e dare un esempio di utilizzazione delle routine scritte, ecco Compute.C che calcola l'espressione sulla riga comando, basta cioè battere "COMPUTE 1+1" per avere in tutta risposta un sorprendente "2": non valeva forse la pena di fare tutta questa fatica per avere un risultato così gratificante?

Dimenticavo alcune cose: il Lattice C accetta i nomi di funzioni lunghi fino a 39 caratteri: se Integr.C dà dei problemi è perché ci sono funzioni, come trapezoideinfinito(), che hanno il nome lungo 18 caratteri, e magari non viene accettato da tutti i compilatori. Aggiunta finale: in Maths.H, oltre che le varie sin(), cos() eccetera, io ho aggiunto una per(a, b) che ritorna la b-esima potenza di a, con a double e b intero! MC

MATHS.H

```
/* funzione   tipo ritornato   azione                          */
/* ---------------------------------------------------------- */
/* per(d, i)  double           ritorna d^i, con i intero      */

double per(a, b)
double a;
int    b;
{
  double r=1.0;
  if (b>=0) {
    for( ; b-->0; r*=a)
      ;
    return(r);
  }
  else {
    for( ; b++<0; r/=a)
      ;
    return(r);
  }
}
```

COMPUTE.C

```
#define MAX 100
#include <ctype.h>
#define CTYPE 1
#include "macro.c"

double varvalore[26];


main(argc, argv)
int  argc;
char *argv[];
{
  double calcola(), ris;
  char c;

  varvalore['P'-'A']=3.141592654;
  varvalore['E'-'A']=2.718281828;
  if (argc!=2) {
    printf("Uso: compute espressione\n");
    exit(1);
  }
  ris=calcola(argv[1], 0, &c);
  if (c==0)
    printf("%lf\n",ris);
  else {
    printf("Errore occorso: numero %d\n", c);
    exit(1);
  }
}
```

Oggi c'è un po' di magia nella tecnologia. Provate ad aprire uno Z-183 ed una strana sensazione si impadronirà di voi. Sarà perché Zenith è il numero uno nell'emozionante mondo dei portatili (rapporto IDC, 1987)?

O le ragioni sono da ricercare nelle favolose capacità di quest'ultima sua creatura? Sarà per lo schermo superiore per definizione ad ogni altro, per la tastiera particolarmente funzionale, per il peso, per la praticità della sua impugnatura o per lo splendido design?

O forse il suo fascino si deve all'autonomia di cinque ore, alle interfacce di entrata-uscita integrate e alle possibilità di collegamento con stampante e mouse? E la totale compatibilità con gli altri PC, la possibilità di integrazione col mondo delle telecomunicazioni, il disco fisso e i dischetti da 3.5", il microprocessore 80C88 CMOS da 8 Mhz e la memoria estensibile a 1.4 Mb? Z-183: è un nome o una formula magica?

ZENITH SINCE 1918 | data systems

Apriti sesamo.

Per richiedere documentazione e informazioni sui prodotti ZENITH, inviare il tagliando oppure telefonare alla DATA MILL Viale Restelli 3/7 20124 Milano - Tel. 603041-2-3-4

Nome ..... Attività ..... Indirizzo .....

Cognome ..... Società ..... Telefono .....