Pascal o non Pascal?

In molte lettere recentemente giunte in redazione mi si accusa di non aver pubblicato programmi scritti in Pascal in questa rubrica. E questo è vero. Si insinua inoltre che non mi piaccia questo linguaggio. E questo è falso. Io lavoro regolarmente in Pascal (più esattamente in Turbo Pascal) sul mio M24 e lo trovo più comodo e più bello del Basic Microsoft (certo che il confronto andrebbe fatto col Basic strutturato del Mac).

Perché allora questa penuria di software Pascal per Apple? Il motivo va ricercato non nel Pascal (quello di Wirth) ma nell'UCSD Pascal (il Sistema Operativo!). Non è assolutamente possibile lavorare con un Editor non residente, con un Filer che gestisce i volumi in un modo così rigido e, soprattutto, con un sistema che va in blocco anche per errori di programma. Il Pascal UCSD è superato, è vecchio e decrepito; ed è ora che vada in pensione. A provvedere sarà probabilmente la versione Apple del Turbo Pascal che sta uscendo ora negli Stati Uniti e che deriva da quella sviluppata per il Mac. La gestione del video è a finestre: in una il listato (edit), in una il sistema operativo (comandi) e nell'ultima l'output del programma. Quando questo Pascal giungerà anche in Italia allora probabilmente aumenteranno gli utenti Pascal, e, di conseguenza, il software scritto in Pascal e inviato in redazione per la pubblicazione.

Nell'attesa 'sorbitevi' questo programma in Pascal che attiva una semplice utility per la gestione del cursore in fase di Input.

Per chi non conoscesse il Pascal ricordo alcune particolarità di questo linguaggio. Intanto è un compilatore, genera perciò un codice eseguibile, scritto in linguaggio macchina, che non necessita più, per girare, dei dischetti del Pascal. Come tutti i compilatori ogni variabile utilizzata deve essere stata prima definita; le variabili possono avere tipi diversi (nel Basic i tipi sono solo tre: intero, reale, stringa) in Pascal i tipi possono essere anche definiti da utente; ad esempio si può definire una variabile Colore co-

```
PROGRAM LEGGIX;
USES TRANSCEND;

type VETTORE = packed array [1..80] of char;

var CAR : VETTORE;
    COUNT : integer;
    VALORE : real;
    SIGN : boolean;

procedure EDIT;

var LENGTH,I : integer;
    BOL : boolean;
    TEMP : char;
    INSIEME : set of char;

  procedure LEFTARROW;

  begin
    if COUNT > 1  then
    begin
      write(TEMP);
      read(keyboard,TEMP);
      COUNT := COUNT - 1;
    end
    else
    begin
      write(chr(7));
      COUNT := 1;
      read(keyboard,TEMP)
    end
  end;

  procedure RIGHTARROW;

  begin
    if COUNT < LENGTH then
    begin
      write(CAR[COUNT]);
      read(keyboard,TEMP);
      COUNT := COUNT +1
    end
    else read(keyboard,TEMP);
    while ord(TEMP) =  8 do LEFTARROW
  end;

  procedure ER;
  begin
   COUNT := COUNT - 1;
   BOL := FALSE;
   write(chr(7))
  end;

  begin
    COUNT := 0; LENGTH := 0;
    INSIEME := ['0'..'9','.','+','-','E',chr(13),chr(32),chr(8),chr(21)];
      repeat
         BOL := TRUE;
         COUNT := COUNT + 1;
         read(keyboard,TEMP);
           while ord(TEMP) =  8 do LEFTARROW;            (* ord(8) = <--  *)
           while ord(TEMP) = 21 do RIGHTARROW;           (* ord(21) = --> *)
         if not (TEMP in INSIEME) then ER;
         if BOL = TRUE and not (ord(TEMP) in [8,21]) then
         begin
           if not (COUNT < LENGTH) then LENGTH := COUNT + 1;
           CAR[COUNT] := TEMP;
           write(CAR[COUNT])
         end;
      until eoln(keyboard);
end;      (* procedure EDIT *)

function POTEN (POWER : integer) : real;
(* questa funzione e' sostituibile,nel Pascal dell'Apple,con PWROFTEN *)

begin POTEN := exp(POWER * ln(10)) end;  (* funzione POTEN *)
```

me un insieme di tinte composte da Rosso, Verde, Blu, Giallo, ecc. (dove per insieme si intende un insieme vero, non un vettore!). Le variabili possono essere definite in testa al programma, e allora valgono in qualsiasi punto del programma, oppure all'interno delle subroutine (che in Pascal si chiamano più correttamente blocchi o procedure) e allora valgono solo all'interno di esse. Per il resto le istruzioni sono abbastanza normali, salvo il fatto che non esiste il Goto; o più esattamente: nella versione originale il Goto non era stato implementato, dopo molte pressioni da parte di programmatori poco 'strutturati' è stata inserita la possibilità di utilizzare il Goto e le Label, a patto però di dichiararlo espressamente in testa al programma (una sorta di mea culpa). Il programma è composto da un'unica istruzione che inizia con il primo Begin (io in genere lo scrivo tutto maiuscolo) e con l'ultimo End riconoscibile perché seguito dal punto (END.), questa istruzione contiene al suo interno altre istruzioni, che a loro volta ne contengono altre, e così via. Questo tipo di struttura del programma consente una eccellente leggibilità del codice sorgente (che si può scrivere con un qualsiasi Word Processor) e un semplice controllo delle singole procedure che compongono il programma. È altresì ovvio che programmare in Pascal non è solo utilizzare il linguaggio Pascal, ma anche e soprattutto pensare in pascal, pensare cioè strutturato. Altrimenti ne verranno fuori dei programmi che del Pascal hanno solo la sintassi, ma non la potenza!

Per chi invece del Pascal non ne vuol proprio sapere, presentiamo un brevissimo programma per lo sviluppo di istogrammi quasi 3D.

```
function DIGITS : real ;
var L,I : integer;
    DIG : real;

begin
 COUNT := 1; L:=0; DIG := 0; SIGN := false;
  while not
  ((CAR[COUNT] = '.') or (CAR[COUNT] = 'E') or (ord(CAR[COUNT]) = 32)) do
    begin
      COUNT := COUNT + 1;                          (* ASCII 32 = CR o spazio *)
      L := COUNT-2;
    end;
  case  CAR[1] of    '-' : begin SIGN := true; CAR[1] := '0' end;
                     '+' : CAR[1] := '0';
  end  (* case *);
  for I := 1 to COUNT-1 do
  begin
    DIG := DIG + (ord(CAR[I]) - 48) * POTEN(L);
    L := L-1
  end;
  DIGITS := DIG
end;   (* funzione DIGITS *)

function DECIMALS : real;
var L1 : integer;
    DEC : real;
    TEMP : char;

begin
  L1 := 0; DEC := 0;
  while not ((ord(CAR[COUNT]) = 32) or (CAR[COUNT] = 'E'))  do
  begin
    COUNT := COUNT +1;
    L1 := L1 + 1;
    TEMP := CAR[COUNT];
    if (TEMP = 'E') or (ord(TEMP) = 32) then TEMP := '0';
    DEC := DEC + (ord(TEMP) - 48) / POTEN(L1);
  end;
  DECIMALS := DEC
end;   (* funzione DECIMALS *)

function EXPONENT : real;
var L :integer;
    SN : real;

begin
  SN := 0;L := 1;
  COUNT := COUNT + 1;
  if CAR[COUNT] in ['0'..'9'] then
    begin
      SN:= POTEN(trunc((ord(CAR[COUNT])-48)*10 + (ord(CAR[COUNT+1])-48)));
      EXPONENT := SN
    end;

  if CAR[COUNT] = '+' then
    begin
      COUNT := COUNT + 1;
      SN:= POTEN(trunc((ord(CAR[COUNT])-48)*10 + (ord(CAR[COUNT+1])-48)));
      EXPONENT := SN
    end;

  if CAR[COUNT] = '-' then
    begin
      COUNT := COUNT+1;
      SN:= 1/POTEN(trunc((ord(CAR[COUNT])-48)*10 + (ord(CAR[COUNT+1])-48)));
      EXPONENT := SN
    end;
  if SN  = 0 then begin SN := 1;EXPONENT := SN end
end;   (* funzione EXPONENT *)

begin   (* main program *)
  write('Introdurre un valore : ');
  EDIT;
  VALORE := (DIGITS + DECIMALS) * EXPONENT;
  if SIGN = true then VALORE := -1*VALORE;
  writeln; write('valore = ',VALORE)
end.
```

Programma LEGGIX

di Daniel Liscia - Torino

Vi invio un programma in Pascal che potrà essere utile a quanti utilizzano il compilatore per programmi di statistica o di elaborazione di dati numerici. Esso ha lo scopo di ovviare ad un difetto del linguaggio che risiede nelle istruzioni READ e READLN. Queste istruzioni sono equivalenti a INPUT del Basic, esse assegnano un valore intero o reale ad una variabile (non prendo quindi in considerazione le variabili del tipo carattere o stringa, quest'ultima d'altronde non prevista dal Pascal standard).

Contrariamente al Basic non è possibile, spostando il cursore, tornare sui caratteri già battuti per correggerli, questo è fonte di grande frustrazione, soprattutto quando, dovendo introdurre 100 valori in un programma, si commette un errore nel 99esimo. Se si tenta una correzione il programma si interrompe ed è allora necessario reinizializzare il sistema, con la conseguente perdita di tutti i dati.

Il programma si compone di due parti:

la prima è una procedura di edit, una versione molto semplificata di un programma di trattamento testi. I simboli numerici vengono letti come caratteri (tipo CHAR) ed ogni singolo valore viene assegnato ad una variabile indicizzata di un array di caratteri, cioè una stringa.

Le sottoprocedure LEFTARROW e RIGHTARROW controllano lo spostamento del cursore e tengono conto (COUNT) della sua posizione nell'array consentendo di modificare i valori già battuti. Una piccola difficoltà di

programmazione è derivata dal fatto che il tasto <-- (codice ASCII = 8) ha come risposta lo spostamento effettivo del cursore mentre il tasto --> (codice ASCII = 21) è «muto» cioè la sua pressione non ha alcuna azione; la logica di questa strana codifica ASCII, se non altro peculiare, mi è ignota. La variabile LENGTH, da non confondersi con la funzione LENGTH della gestione delle variabili di tipo stringa nell'UCSD Pascal, memorizza il numero massimo di caratteri battuti e consente di non far superare al cursore l'ultimo carattere di destra.

Se vengono inavvertitamente battuti tasti che non corrispondono a cifre, i segni «+» «−», il punto decimale, la «E» (maiuscola) della notazione scientifica, si avrà un segnale di errore (CHR(7)). La seconda parte del programma è costituita da tre funzioni: DIGITS, DECIMALS, EXPONENT.

Queste compiono la scansione della stringa di caratteri individuando progressivamente il segno e le cifre prima della virgola, i numeri decimali, il segno e l'esponente di 10 (le due cifre che seguono la «E»).

Input corretti sono, ad esempio:

123, +123, −123, 123.45, −123.45E06, +123.45E+06

Non corretti sono:

123.45E6 (10 alla sesta deve essere battuto come «E06» o «E+06») 123.45E−6, 12.3.45.

Il numero di cifre significative rimane limitato a sette e l'intervallo di valori è compreso fra 1.0E+37 e 1.0E−37. Questi limiti possono variare a seconda del compilatore e del calcolatore usato.

Ho cercato il più possibile di rimanere fedele al Pascal Standard evitando di utilizzare le variabili di tipo stringa. La gestione delle stringhe di caratteri infatti varia in modo considerevole nei diversi compilatori, e d'altronde è comprensibile la riluttanza di N. Wirth ad inserire le stringhe nei suoi linguaggi (Modula-2 compreso) risulta infatti abbastanza semplice utilizzare gli array di caratteri e costruirsi le proprie funzioni.

L'unica eccezione, nel programma LEGGIX, è il file KEYBOARD che consente di leggere un carattere senza che questo appaia sullo schermo. L'assenza del file KEYBOARD nel Pascal Standard è spiegabile con il fatto che all'inizio degli anni '70, l'introduzione dei dati veniva effettuata principalmente con le schede perforate e non mediante terminale con monitor.

USES TRANSCEND è necessaria, nell'UCSD dell'Apple per la funzione EXP. Nel Pascal Standard può essere omesso.

Istogrammi 3-D

di Massimiliano Scordamaglia
Castelfranco Emilia

Il programma in questione è abbastanza corto (la parte di calcolo vera e propria si limita a poche righe) ed è sufficientemente comprensibile.

Il programma parte visualizzando i tasti comando e premendo A passa direttamente a chiedere tutti i dati utili per il proseguimento del calcolo, come l'intervallo (utile solo per la tabella), le modalità, ovvero il numero di frequenze che vogliamo visualizzare, la modalità di partenza anche questa utile solo per la tabella e infine viene chiesta la frequenza tante volte quante le modalità che abbiamo inserito.

Naturalmente non sono ammesse sia frequenze negative che modalità negative o maggiori di 100.

A questo punto comincia il calcolo trovando per prima la frequenza maggiore salvandola in MM, poi inizia il disegno con uno "scale" dell'asse X e Y.

L'asse X è dato da R, variabile, contenente il rapporto tra 280 e il numero di istogrammi più la metà della larghezza di un istogramma per permettere che lo spessore non esca dallo schermo.

Dalla lunghezza R viene poi ricavato L che è l'80 per cento dello spazio destinato a ciascun istogramma, e infine viene trovato S lo spessore che è il 50 per cento della larghezza dell'istogramma.

L'asse Y è semplicemente stato ricavato con la proporzione «280: MM = x:F(n)».

Finalmente si parte a disegnare il rettangolo principale e successivamente lo spessore.

Ci si può rendere conto che degli istogrammi così strutturati sono trasparenti, per cui ho eseguito un «fill» della parte frontale e dello spessore.

Finito il disegno, il computer aspetta qualche carattere e torna al menu.

I comandi sono:

T che visualizza la tabella

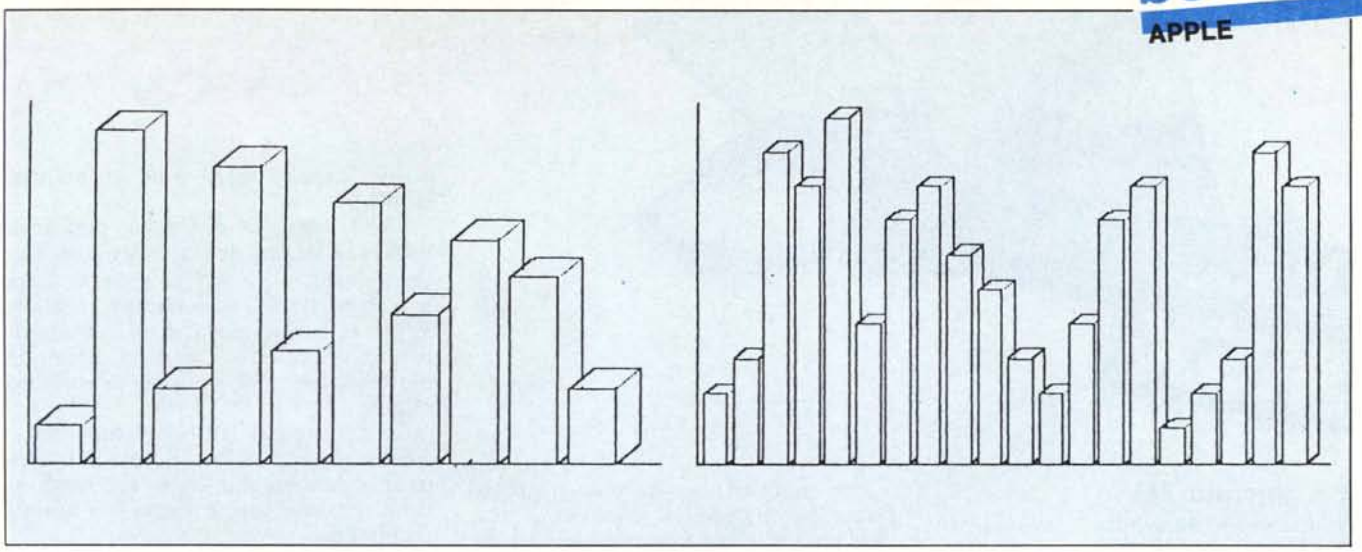
A che accetta nuovi dati

I che visualizza nuovamente gli istogrammi

Q che fa finire il programma.

Naturalmente c'è CTRL S che blocca lo scrolling della tabella. Come si vede il programma non presenta parti-

```
5  DIM F(100)
10  TEXT : HOME : POKE 34,2
20  INVERSE : PRINT " ISTOGRAMMI 3-D ": NORMAL : PRINT
30  PRINT "COMANDI :": PRINT "I : VISUALIZZA ISTOGRAMMI"
40  PRINT "T : VISUALIZZA TABELLA"
50  PRINT "CTRL S: FERMA E RIPRENDE SCROLL TABELLA"
60  PRINT "A : ALTRI ISTOGRAMMI"
70  PRINT "Q : FINE PROGRAMMA": PRINT
75  GOTO 410
80  INPUT "QUALE INTERVALLO ?";I
90  INPUT "QUANTE MODALITA' ?";M:M =  INT (M)
100  IF M <  = 0 OR M > 100 THEN 90
110  INPUT "MODALITA' DI PARTENZA ";MP
120  FOR C = 1 TO M
130  PRINT "FREQUENZA N.";C: INPUT F(C)
140  IF F(C) < 0 THEN  INVERSE : PRINT "NON SONO AMMESSE FREQUEN
     ZE NEGATIVE": NORMAL : GOTO 130
150  NEXT C
160 MM = F(1)
170  FOR C = 2 TO M
180  IF F(C) > MM THEN MM = F(C)
190  NEXT C
195  IF MM = 0 THEN 10
200  HGR2
210  HCOLOR= 7
220  HPLOT 0,0 TO 0,191 TO 279,191
230 R = (276 - (276 / M) / 2) / M:L = .8 * R:S = .5 * L
240  FOR C = 0 TO M - 1
250  HCOLOR= 7
260 Y = ((191 - S * 1.5) * F(C + 1)) / MM
270 X = R * C + 3
280  HPLOT X,191 TO X,191 - Y TO X + L,191 - Y TO X + L,191
290  HPLOT X,191 - Y TO X + S,191 - Y - S TO X + L + S,191 - Y -
     S TO X + L,191 - Y
300  HPLOT X + L,191 TO X + L + S,191 - S TO X + L + S,191 - Y -
     S
310  HCOLOR= 4
320  FOR E = 192 - Y TO 191
330  IF E = 192 THEN 350
340  HPLOT X + 1,E TO X + L - 1,E
350  NEXT E
360  FOR E = 192 - Y - S TO 190 - Y
370 D = 192 - Y - E
380  HPLOT X + 1 + D,E TO X + L - 2 + D,E
390  NEXT E
400  NEXT C
410  GET A$
420  IF A$ = "T" THEN 470
430  IF A$ = "A" THEN 80
440  IF A$ = "I" THEN  POKE  - 16304,0: POKE  - 16297,0: POKE  -
     16299,0: GOTO 410
450  IF A$ = "Q" THEN  TEXT : HOME : END
460  GOTO 10
470  REM
480  TEXT : HOME : PRINT "!     MODALITA'     !    FREQUENZE
      !----------------------------------------"
490  POKE 34,2
500 C = 0
510  FOR P = MP TO MP + M * I - I STEP I
520 C = C + 1
530  PRINT "!   ";P;" - ";P + I;
540  HTAB 20: PRINT "!        ";F(C)
550  NEXT P
560  GOTO 410
```



colari difficoltà di comprensione come d'altra parte non penso che contenga enormi deficienze visto che l'ho provato con moltissimi valori.

Lista delle variabili

F : vettore contenente le frequenze;
I : intervallo;
M : numero di modalità;
MP: modalità di partenza;
MM: variabile contenente la frequenza maggiore;
R : rapporto determinante lo spazio per ciascun istogramma;
L : larghezza dell'istogramma;
S : spessore dell'istogramma;
X : variabile contenente il valore dell'asse x;
Y : variabile contenente il valore dell'asse y;
A$: scelta dell'utente;
C,E,P: variabili usate per i cicli FOR/NEXT

Il programma

5: dimensionamento di F;
10-70: presentazione e stampa dei tasti da utilizzare;
80-150: input dei dati;
160-190: calcolo della frequenza maggiore;
170-400: disegno istogrammi;
410-460: input dei tasti;
470-560: visualizzazione tabella.

MC