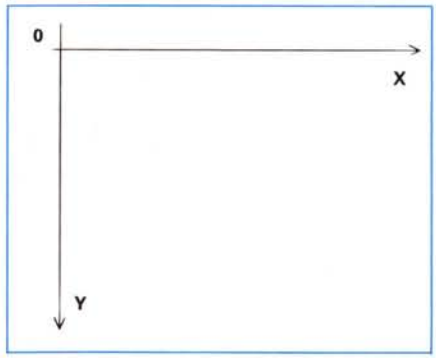
Il merito principale del primo programma di questo mese è la documentazione! Arrivano infatti in redazione decine di programmi ben fatti, e spesso anche utili, ma con solo due righe di commento; visto che in genere non si pubblicano i listati, diventa indispensabile una buona documentazione. Riassumo quindi come corredare il software inviato per la pubblicazione: innanzitutto il dischetto con tutti i programmi, sia i sorgenti che gli eseguibili, possibilmente anche un piccolo demo, poi la descrizione delle principali funzioni del programma, la spiegazione di quali difficoltà si siano incontrate e come sono state risolte e, se non sono lunghissime, le istruzioni d'uso. Non servono i listati su carta e tutta la documentazione può essere messa direttamente sul disco sia in formato ASCII (solo testo o non-document per intenderci) sia in formato WS o WORD. Affinché poi sia possibile remunerare i lavori servono i dati fiscali completi: nome, cognome, indirizzo, luogo di nascita e codice fiscale.
Il secondo programma converte in stringa un numero, ne pubblicammo una versione per Apple II diversi anni or sono, e altre versioni sono state inviate nel frattempo, ma questa è l'unica che non mette quasi limiti alla grandezza dell'importo e che non scrive cose tipo «unomilionidieciottomila...»; non ho ben capito l'utilità della valuta, visto che se devo scrivere un importo in dollari la routine dovrebbe gestire anche i centesimi, cosa che non fa!

Hard-Copy per Hercules

di Antonio Martinelli - Bari

Sappiamo che lo schermo grafico della scheda grafica Hercules, corrisponde,



nel piano cartesiano, al quarto quadrante:
con gli estremi nei punti (719,0) e (0,347). Volendo stampare la schermata grafica a piena pagina, dovrò far sì che il carrello della testina si sposti scandendo l'asse Y da 347 a 0, mentre il rullo avanza al crescere delle X, (vedi fig. 1).

Allora, supposto che si conosca come una stampante a 9 aghi gestisce la grafica, ogni linea che in questa modalità andremo a stampare sarà costituita da una stringa di 348 caratteri ASCII e corrispondentemente sullo schermo, da un blocco di 348x8 pixel.

A questo punto è chiaro che non c'è assolutamente alcun problema se si vuole ottenre un'hardcopy in singola densità del video. Il problema è solo nel risultato qualitativo. Infatti, per quanto una stampa di questo tipo sia veloce, in taluni casi, visto che le due cose sono più o meno inversamente proporzionali, si può desiderare una stampa più precisa, più delineata: insomma una stampa migliore. Come fare però?

L'idea più semplice, è quella di utilizzare una densità di stampa che non sia la singola, ottenendo qualcosa del tipo in figura 2, che è già qualcosa, ma evidentemente lascia il problema dello spazio tra dot delle linee diagonali (v. fig. 2 a), di quelle verticali (b) e oltretutto genera il problema dell'ispessimento orizzontale dei contorni (c) (tutte cose, comunque, più o meno visibili a seconda della qualità della testina e del nastro).

Come migliorare ancora la stampa?

Il suggerimento è immediato. Nume-

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. 295.

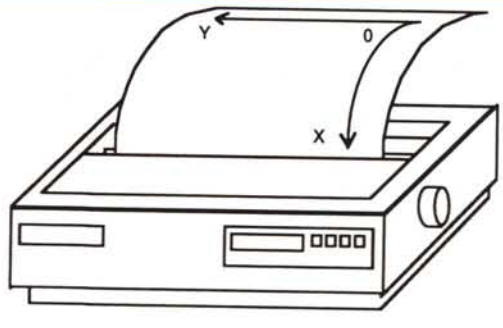


Figura 1

rosissime stampanti consentono uno step del rullo inferiore al dot, più precisamente di 1/216 di pollice o equivalentemente di 1/3 di dot (vedi sequenza ESCAPE CHR$(27) "J" CHR$(n)).

Una capacità che se utilizzata congiuntamente all'idea del ripasso, ci permette di ottenere ragionevoli risultati.

In definitiva si sfrutterà, in modo banale come si vede in figura 2 la doppia densità per migliorare la risoluzione orizzontale, ed un ripasso «ad hoc» shiftato verso il basso di 1/3 di dot, per sopperire all'insufficiente risoluzione verticale ed obliqua; in pratica anziché stampare utilizzando per ogni linea grafica in un unico passaggio una stringa di 348 caratteri ASCII (singola densità), si stamperà utilizzando una stringa di 348x2 caratteri (doppia densità) per il primo passaggio (v. var RIPASSO1), ed ancora una seconda stringa, anche questa di 348x2 caratteri, per il secondo passaggio shiftato (v. var RIPASSO2), raggiungendo una potenziale risoluzione visibile in figura 3.

Ma adesso vediamo come funziona la procedura. Saltando il blocco che resetta la stampante, si incontra un primo ciclo, CICLO1, che è quello che individua i blocchi di 8x348 pixel di cui ho parlato all'inizio. Subito dopo, il primo dei due cicli annidati nel precedente; quello che ingloba i blocchi B1 e B2.

Questi ultimi, (con una condizione IF CICLO1=0 THEN... che evita un inutile ricalcolo di L [CICLO2] a partire dal secondo passaggio di CICLO2), servono a costruire i vettori L e L_SUCC, che contengono la codifica base dei bit, relativi, rispettivamente, alla linea che poi verrà stampata (nel secondo ciclo CICLO2) e di quella immediatamente successiva (che serve per le elaborazioni del blocco B3).

Il secondo ed ultimo ciclo CICLO2, è quello che contiene il blocco B3, cuore della procedura, la stampa vera e propria della linea e del suo ripasso (RIPASSO1 e RIPASSO2).

A questo punto non ci resta altro che scoprire come funziona il blocco B3.

A tale scopo ritengo che sia meglio avvalerci della simulazione di un passaggio di quest'ultimo.

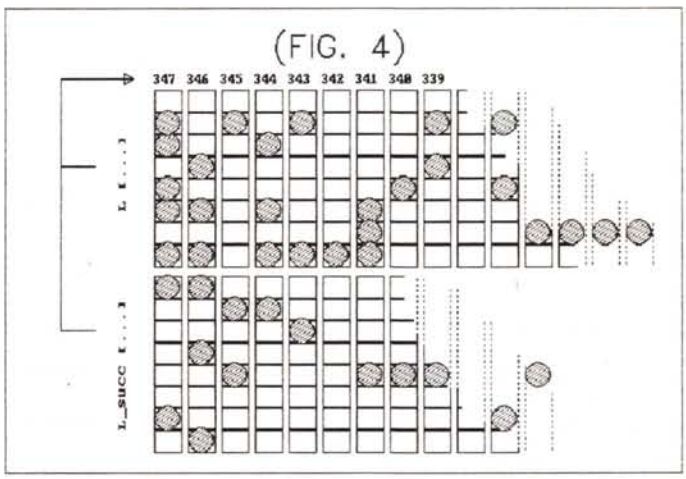
Supponiamo quindi di avere a disposizione L e L_SUCC tali che le configurazioni dei dot corrispondenti siano per esempio quelle in figura 4, e consideriamo l'effetto che produce il primo passaggio del secondo CICLO2.

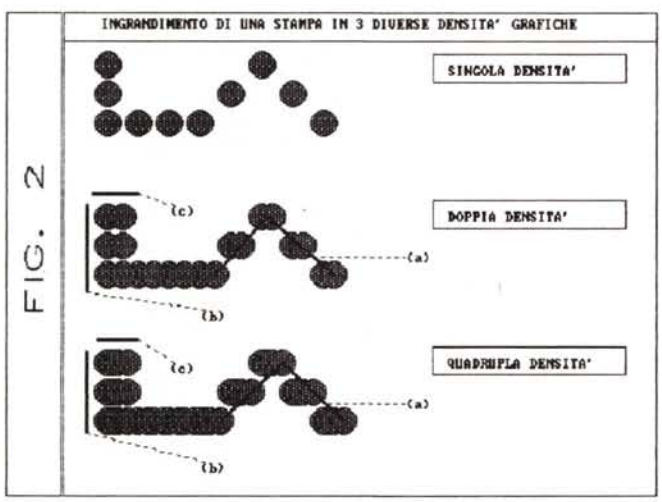
La configurazione in analisi è quella relativa a L [347] (fig. 4a) e i risultati dei

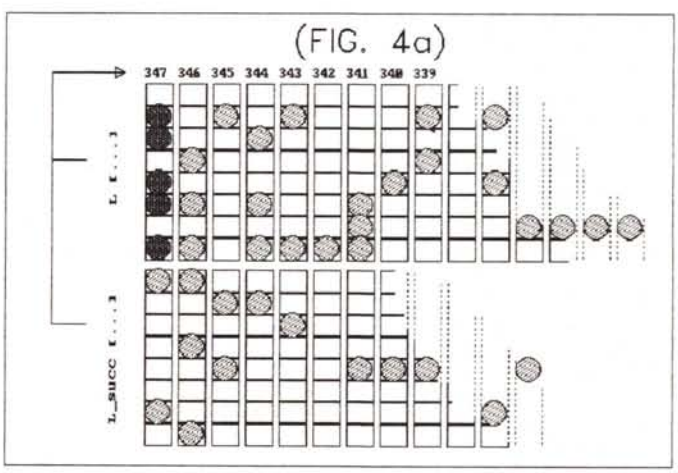
```
{//////////////////////////////////////////////////////////////////
 // PROCEDURA DI HARDCOPY PER SCREEN GRAFICO SCHEDA HERCULES      //
 //         Ver         : 1.00  Aprile 90                          //
 //         Compilatore : TURBO PASCAL 5.00                        //
 //         Uso         : chiamarla  quando  è  ancora  attiva  la //
 //                       pagina grafica   che  interessa stampare //
 //         Note        : fra le  USES  includere  necessariamente //
 //                       GRAPH e PRINTER                          //
 //         Caratterist.: buona qualità/velocità di stampa         //
 //////////////////////////////////////////////////////////////////
 // Autore : MARTINELLI ANTONIO                                   //
 //////////////////////////////////////////////////////////////////}


PROCEDURE HCOPY;
TYPE  RIGA           = ARRAY [-1..1920] OF CHAR;
VAR   L              : ARRAY [-1..1920] OF BYTE;
      L_SUCC         : ARRAY [-1..1920] OF BYTE;
      RIPASSO1       : RIGA;
      RIPASSO2       : RIGA;
      CAR            : CHAR;
      CICLO1,CICLO2  : INTEGER;
      ASCII          : BYTE;
      XMAX,YMAX      : INTEGER;
      N1,N2          : INTEGER;

BEGIN
    YMAX:=347;                       {******** COORDINATE MASSIME ********}
    XMAX:=719;                       {********    DELLA SCHEDA    ********}

    CAR:=CHR(27);WRITE (LST,CAR);    {*  MASTER RESET **}
    CAR:='@'    ;WRITE (LST,CAR);    {**** STAMPANTE ***}

    L     [-1]:=0;                   {** E' NECESSARIO CHE QUESTE VARIABILI **}
    L_SUCC[-1]:=0;                   {** SIANO AZZERATE PER IL BLOCCO B3    **}

    FOR CICLO1:=0 TO (XMAX DIV 8) DO

    BEGIN
      FOR CICLO2:=YMAX DOWNTO 0 DO
      BEGIN
         IF CICLO1=0 THEN BEGIN
                        ASCII:=0;                                              {**************}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)  ,CICLO2)*128);     {**          **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+1,CICLO2)*64 );     {**          **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+2,CICLO2)*32 );     {**          **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+3,CICLO2)*16 );     {**  BLOCCO  **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+4,CICLO2)*8  );     {**    B1    **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+5,CICLO2)*4  );     {**          **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+6,CICLO2)*2  );     {**          **}
                        ASCII:=ASCII+(GETPIXEL ((CICLO1*8)+7,CICLO2)*1  );     {**          **}
                        L[CICLO2]:=ASCII;                                      {**************}
                      END
         ELSE L[CICLO2]:=L_SUCC[CICLO2];
         ASCII:=0;                                                  {**************}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)  ,CICLO2)*128);    {**          **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+1,CICLO2)*64 );    {**          **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+2,CICLO2)*32 );    {**          **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+3,CICLO2)*16 );    {**  BLOCCO  **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+4,CICLO2)*8  );    {**    B2    **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+5,CICLO2)*4  );    {**          **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+6,CICLO2)*2  );    {**          **}
         ASCII:=ASCII+(GETPIXEL (( (CICLO1+1)*8)+7,CICLO2)*1  );    {**          **}
         L_SUCC [CICLO2]:=ASCII;                                    {**************}
      END;
      FOR CICLO2:=YMAX DOWNTO 0 DO
      BEGIN                                                                                                  {**************}
                                                                                                             {**          **}
         ASCII:=(L[CICLO2]);                                                                                 {**          **}
         IF ASCII=26 THEN ASCII:=27;RIPASSO1[CICLO2*2  ]:=CHR(ASCII);                                        {**          **}
                                                                                                             {**          **}
         ASCII:=(L[CICLO2] AND L[CICLO2-1]);                                                                 {**          **}
         IF ASCII=26 THEN ASCII:=27;RIPASSO1[CICLO2*2-1]:=CHR(ASCII);                                        {**  BLOCCO  **}
                                                                                                             {**          **}
         ASCII:=(L[CICLO2] AND ( (L[CICLO2] SHL 1)+(L_SUCC[CICLO2] SHR 7) ));                                {**    B3    **}
         IF ASCII=26 THEN ASCII:=27;RIPASSO2[CICLO2*2  ]:=CHR(ASCII);                                        {**          **}
                                                                                                             {**          **}
         ASCII:=(L[CICLO2] AND (L[CICLO2-1] SHR 1) );                                                        {**          **}
         ASCII:=(ASCII shl 1) OR    ( ((L[CICLO2-1] SHL 1)+(L_SUCC[CICLO2-1] SHR 7) ) AND L[CICLO2] );       {**          **}
         IF ASCII=26 THEN ASCII:=27;RIPASSO2[CICLO2*2-1]:=CHR(ASCII);                                        {**          **}
      END;                                                                                                   {**************}

      CAR:=CHR(27)       ;WRITE (LST,CAR);                   {*******************************}
      CAR:='L'           ;WRITE (LST,CAR);                   {** PREPARA LA STAMPANTE A    **}
      N2:=(348*2      ) DIV 256;                             {** RICEVERE LA LINEA GRAFICA **}
      N1:=(348*2      )-(N2*256);                            {** DEL 1° PASSAGGIO IN       **}
      CAR:=CHR(N1)       ;WRITE (LST,CAR);                   {** MODALITA' DOPPIA DENSITA' **}
      CAR:=CHR(N2)       ;WRITE (LST,CAR);                   {*******************************}
      FOR CICLO2:=YMAX*2 DOWNTO -1 DO WRITE (LST,RIPASSO1[CICLO2]);   {**************************************}
      WRITE (LST,CHR(13));                                   {**  STAMPA LA LINEA DI BASE, MANDA **}
      CAR:=CHR(27)       ;WRITE (LST,CAR);                   {**  UN "CR" ED  AVANZA IL RULLO    **}
      CAR:='J'           ;WRITE (LST,CAR);                   {**  DI 1/216 DI POLLICE            **}
      CAR:=CHR(01)       ;WRITE (LST,CAR);                   {**************************************}

      CAR:=CHR(27)       ;WRITE (LST,CAR);                   {*******************************}
      CAR:='L'           ;WRITE (LST,CAR);                   {** PREPARA LA STAMPANTE A    **}
      N2:=(348*2      ) DIV 256;                             {** RICEVERE LA LINEA GRAFICA **}
      N1:=(348*2      )-(N2*256);                            {** DEL 1° PASSAGGIO IN       **}
      CAR:=CHR(N1)       ;WRITE (LST,CAR);                   {** MODALITA' DOPPIA DENSITA' **}
      CAR:=CHR(N2)       ;WRITE (LST,CAR);                   {*******************************}
      FOR CICLO2:=YMAX*2 DOWNTO -1 DO WRITE (LST,RIPASSO2[CICLO2]);   {**************************************}
      WRITE (LST,CHR(13));                                   {** STAMPA LA LINEA DI RIPASSO, MANDA **}
      CAR:=CHR(27)       ;WRITE (LST,CAR);                   {** UN "CR" E DAVANZA IL RULLO    DI **}
      CAR:='J'           ;WRITE (LST,CAR);                   {** 23/216 DI POLLICE                **}
      CAR:=CHR(23)       ;WRITE (LST,CAR);                   {**************************************}


    END;
END;


   {//////////////////////////////////////////////////////////////////
    // QUALORA INTERESSI CENTRARE LA STAMPA OCCORRE:                 //
    //                                                               //
       MODIFICARE ( ) IN : N2:=(348*2+120   ) DIV 256;               //
                           N1:=(348*2+120   )-(N2*256);              //
       ED AGGIUNGERE ( ) : FOR CICLO2:=1 TO 120 DO WRITE (LST,CHR(0)); //
    //////////////////////////////////////////////////////////////////}
```

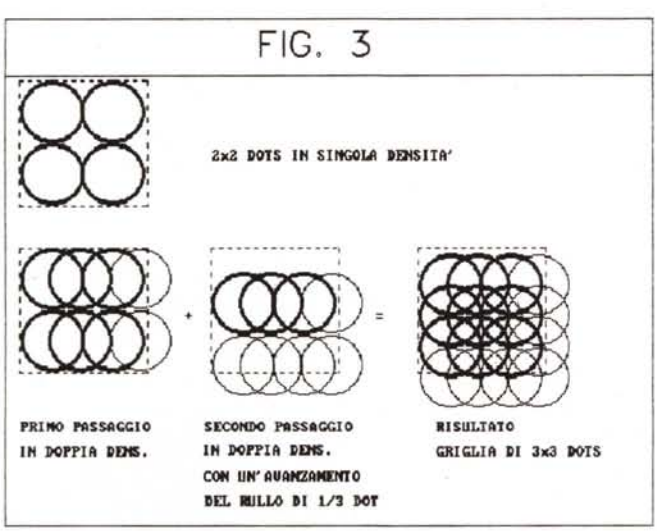
4 sottoblocchi distinguibili in B3 sono visibili in ordine in figura 4 b, c, d, e, con il risultato finale ben visibile in figura 4f.

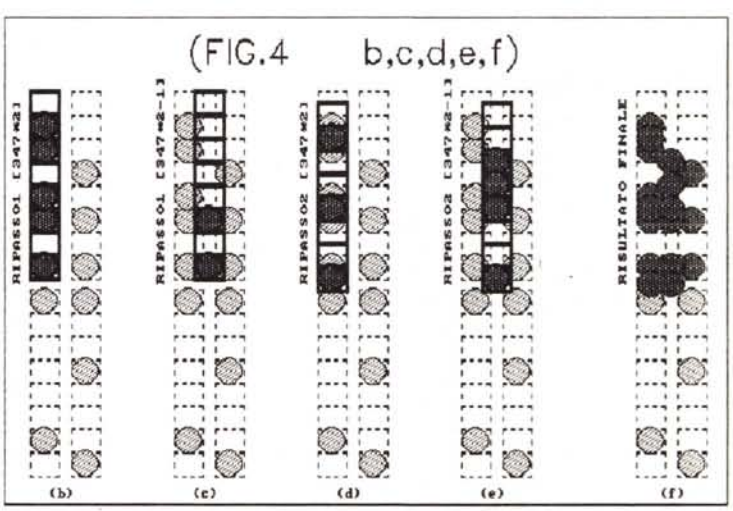
Come commento finale ritengo che, aldilà di un possibile ulteriore perfezionamento della qualità di stampa (incremento della risoluzione utilizzando la quadrupla densità ed un terzo ripasso) e di un facile adattamento per tutte le schede grafiche gestite dal T.PASCAL 5, l'interesse per questa procedura deve andare oltre lo specifico utilizzo di libreria per programmi personali. Infatti ritengo che un utilizzo interessante potrebbe essere quello di sfruttarla in un programmino che traduca insoddisfacenti stampe su disco, in stampe un po' più accurate dal lato estetico.











Convert

di Marco Cardin - Padova

Il programma Convert, per computer MS-DOS, scritto in Turbo Pascal (vs. 4.0) è una semplice, ma utile, routine che permette di trasformare un numero intero nella sua espressione in lettere corrispondenti.

Ad esempio, dando in input 1234, si avrà come output «milleduecentotrentaquattro».

Il range degli interi ammessi varia da 0 a 999.999.999.999.999. È inoltre possibile specificare una «unità monetaria» e da ciò mi sembra sia abbastanza chiaro l'utilizzo di questa routine.

Ritengo che sia molto utile nel caso di compilazione mediante personal di ricevute, conti correnti o di qualsiasi altro documento che richieda l'esplicitazione anche in forma letterale di una determinata cifra. È il caso, ad esempio, delle banche quando compilano gli assegni.

L'idea di realizzare questo programma è nata quando ho dovuto compilare circa un centinaio di ricevute per l'associazione di volontariato presso la quale opero (l'A.N.P.Ha, Associazione Nuoto Portatori Handicaps): arrivato all'ottantesima iniziavo ad avere visioni mistiche alla «Fantozzi» e alla fine non avevo assolutamente la certezza di ciò che avevo scritto. Ripetere per cento volte di seguito espressioni del tipo «unmilionequattrocentosessantaseimilacinquecentoquarantacinque» è veramente alienante, figuriamoci se le ricevute fossero state il doppio o il triplo!

Da qui quindi l'idea di computerizzare il procedimento in modo da evitare probabili esaurimenti nervosi e di ottenere risultati senza dubbio più professionali dal punto di vista estetico e, soprattutto, esenti dai sempre possibili errori di trascrizione.

Il programma

L'algoritmo su cui è strutturato il programma si basa sulla scomposizione del numero inserito, interpretato come stringa, in blocchi di tre elementi ciascuno. Ogni blocco viene analizzato per individuare le centinaia, le decine e le unità.

Il valore in input è interpretato come stringa per permettere l'utilizzo di punti o virgole per la separazione delle migliaia, possibilità non data dal software in possesso a molte banche. Verranno, quindi, interpretati correttamente, ad esempio, sia 1.234.567 che 1,234,567 oltre a 1234567.

Una prima parte del programma si occupa di riscontrare l'eventuale presen-

Convert

```
program convert (input,output);

 uses
  crt;

 const
  valori=500;

 var
  cf:string[2];
  cl:string[6];
  nc:string[19];
  mu,um,vl:string;
  b,c,e,h,i,j,k,l,m,n,p,nf,nn,ps,cod,cont:integer;
  mi:array[0..3] of string[8];
  ml:array[2..9] of string[9];
  nl:array[1..19] of string[11];
  nr:array[1..valori] of string[19];

 procedure scan (d,ps:integer);          (* converte una cifra del blocco *)
  begin                                  (* nell'integer corrispondente   *)
   cf:=copy(nc,d,ps);
   val(cf,nf,cod);
  end;

 procedure lung (n:integer);             (* determina il numero di blocchi *)
  begin                                  (* da tre elementi ciascuno oltre *)
   m:=n div 3;                           (* a quello eventuale iniziale da *)
   p:=n-m*3;                             (* uno o due elementi             *)
   case p of
    0:k:=0;
    1:k:=2;
    2:k:=1;
   end;
   if (p > 0) then h:=m+1
    else h:=m;
  end;


 begin

  nl[1]:='uno'; nl[2]:='due'; nl[3]:='tre'; nl[4]:='quattro'; nl[5]:='cinque';
  nl[6]:='sei'; nl[7]:='sette'; nl[8]:='otto'; nl[9]:='nove'; nl[10]:='dieci';
  nl[11]:='undici'; nl[12]:='dodici'; nl[13]:='tredici'; nl[14]:='quattordici';
  nl[15]:='quindici'; nl[16]:='sedici'; nl[17]:='diciasette';
  nl[18]:='diciotto'; nl[19]:='diciannove';
  ml[2]:='venti'; ml[3]:='trenta'; ml[4]:='quaranta'; ml[5]:='cinquanta';
  ml[6]:='sessanta';ml[7]:='settanta';ml[8]:='ottanta';ml[9]:='novanta';
  mi[0]:='mila'; mi[1]:='miliardi'; mi[2]:='milioni'; mi[3]:='mila';

  clrscr;
  writeln('inserire i numeri  [max ',valori,']:');
  writeln;
  j:=0;
  repeat                                 (* input dei valori *)
   j:=j+1;
   write('nr[',j,']=');
   readln(nr[j]);
  until nr[j]='';
  writeln;
  write('unità monetaria: ');
  readln(um);
  mu:=um;
  clrscr;
  for cont:=1 to j-1 do            (* inizio ciclo per tutti i valori input *)
   begin
    nc:=nr[cont];
    n:=length(nc);
    lung (n);
    if (n > 3) then                      (* inizio controllo presenza di *)
     begin                               (* punti o virgole              *)
      nn:=n;
      scan (nn-3,1);
      while cod <> 0 do
       begin
        delete(nc,nn-3,1);
        n:=n-1;
        nn:=nn-4;
        scan (nn-3,1);
       end;
      if (nn <> n) then lung (n);
     end;                                (* fine controllo *)
    vl:='';
    m:=h;
    l:=1;
    c:=0;
    for i:=1 to h do                (* inizio ciclo analisi di ogni blocco *)
     begin                          (* del singolo valore di input         *)
      b:=0;
      scan (l,1);
      if (k=0) then                      (* k=0 -> il blocco ha 3 cifre *)
       begin
        if (nf=1) then vl:=concat(vl,'cento')
         else
          if (nf <> 0) then vl:=concat(vl,nl[nf],'cento')
```

```
        else if (nf=0) then b:=b+1;
      scan (l+1,1);
      if (nf=0) then
       begin
        b:=b+1;
        scan (l+2,1);
        if (nf <> 0) then
         begin
          if (nf=1) and (b=2) then
           begin
            if (m=2) or (m=5) then
             begin
              vl:=concat(vl,'mille');
              b:=3;
              e:=1;
             end;
            if (m=3) then
             begin
              vl:=concat(vl,'unmilione');
              b:=3;
              e:=1;
             end;
            if (e=0) then vl:=concat(vl,nl[nf]);
           end
          else vl:=concat(vl,nl[nf]);
         end
        else b:=b+1;
       end
      else c:=1;
     end;
    if (k=1) or (c=1) then              (* k=1 -> il primo blocco ha 2 cifre *)
     if (nf=1) then
      begin
       scan (l+c,2);
       vl:=concat(vl,nl[nf]);
      end
     else
      begin
       scan (l+c,1);
       vl:=concat(vl,ml[nf]);
       scan (l+c+1,1);
       if (nf <> 0) then
        begin
         if (nf=1) or (nf=8) then
          begin
           n:=length(vl);
           delete(vl,n,1);
          end;
         vl:=concat(vl,nl[nf]);
        end;
      end;
    if (k=2) then                       (* k=2 -> il primo blocco ha 1 cifra *)
     if (nf=0) then vl:='zero'
      else vl:=nl[nf];
    l:=l+3-k;
    c:=0;
    e:=0;
    k:=0;
    if (m > 1) and (b <> 3) then vl:=concat(vl,mi[5-m]);
    if (m=4) and (b=3) then vl:=concat(vl,mi[1]);
    m:=m-1;
   end;                                 (* fine ciclo analisi del singolo valore *)
  cl:=copy(vl,1,6);                       (* serie di controlli per una migliore *)
  if (cl='unomil') then                   (* conversione del valore analizzato   *)
   begin
    if (h=2) or (h=5) then
     begin
      delete(vl,1,7);
      vl:=concat('mille',vl);
     end;
    if (h=3) then
     begin
      delete(vl,1,10);
      vl:=concat('unmilione',vl);
     end;
    if (h=4) then
     begin
      delete(vl,1,11);
      vl:=concat('unmiliardo',vl);
     end;
   end;
  if (um <> '') then
   begin
    n:=length(vl);
    cl:=copy(vl,n-1,2);
    if (cl='ne') or (cl='ni') or (cl='do') or (cl='di')
     then um:=concat(' ','di',' ',mu)
      else um:=concat(' ',mu);
   end;
  writeln(nr[cont],' ==> ',vl,um);    (* output valori in forma letterale *)
  writeln;
 end;                                 (* fine ciclo per tutti i valori di input *)

end.
```

za di punti o virgole, per passare alla procedura LUNG il corretto valore della lunghezza della stringa di input, in modo da individuare con precisione il numero di blocchi costituiti da tre elementi ciascuno.

La procedura SCAN è una sorta di «scandaglio» che analizza ogni elemento del singolo blocco convertendolo nel corrispondente valore integer tramite l'istruzione VAL. Ciò permette di associare, ad esempio, al valore «9» stringa l'integer 9 e da qui la corrispondente espressione letterale «nove».

Un ciclo FOR iniziale ripete il procedimento sin qui visto per il numero di blocchi calcolato, ponendo alla fine di ogni blocco analizzato e trasformato in lettere la giusta desinenza «mila», «milioni», «miliardi». Tanto per fare un esempio: 21.234.567 è scomposto in 3 blocchi. Nel primo c'è 21, nel secondo 234 e nel terzo 567: alla fine del primo la desinenza sarà «milioni», alla fine del secondo sarà «mila».

Da notare che sono eliminate automaticamente cacofonie del tipo «trentauno», «unomilioni», «ottantaotto», ecc., sostituendole con le più corrette «trentuno», «unmilione», «ottantotto», ecc. È possibile inserire circa 3000 valori senza oltrepassare il limite dei 64 Kbyte dato dal fatto che in Tubo Pascal non è possibile, se non con variabili dinamiche, creare variabili per più, appunto, di 64 Kbyte.

Utilizzo

È chiaro che questa routine presa isolatamente non dimostra l'utilità che ha in realtà. Ritengo pertanto che anche il programmatore meno esperto non troverà difficoltà ad adattarla per un utilizzo più concreto quando l'output fosse destinato a ricevute, assegni, conti correnti o a qualsiasi altro documento che necessiti dell'esplicitazione letterale di una determinata cifra.

Penso che questo programma possa risultare utile a liberi professionisti, associazioni, cooperative e a quanti vogliano velocizzare e migliorare questa parte della loro attività con un tipo di software che per quanto so è disponibile per gli istituti di credito e non certo per i molti potenziali utilizzatori.

Per quanto mi riguarda, l'ho inserito nel più generale contesto di compilazione di ricevute (la cui maschera può anche essere facilmente creata con qualsiasi word processor), passando da circa un'ora e mezza necessaria alla compilazione manuale ai pochi minuti richiesti per la parte di input e di output del programma, con risultati senza dubbio migliori. MC