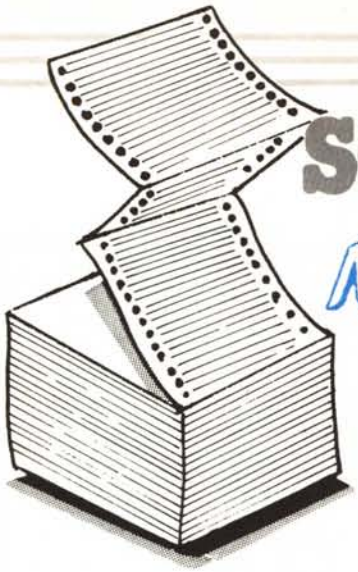


a cura di Valter Di Dio

Mini libreria grafica per Turbo Pascal

Una delle cose piacevoli della programmazione in Pascal, e in genere dei linguaggi più recenti, è la possibilità di crearsi delle librerie proprie da includere direttamente dentro al sorgente. Se oltretutto queste nuove procedure hanno lo stesso nome di una delle procedure standard del Turbo Pascal, ne prendono automaticamente il posto, consentendoci così di modificare anche parte del linguaggio di base.

D'altro canto però il Turbo Pascal non consente di avere librerie precompilate da linkare al nostro programma. Questo ci eviterebbe l'attesa della compilazione di moduli che già sappiamo corretti e che oltretutto si devono trovare per forza in testa alle nostre routine.

Chi infatti utilizza il Turbo Graphics Tool Box si deve sorbire ogni volta la ricompilazione del Typedef.SYS, del Graphic.SYS, del Kernel.SYS e se vuole usare anche le finestre del Window.SYS. Tutto ciò anche solo per disegnare un quadratino nella pagina grafica, e a poco serve il Dummy.SYS che, se accorcia il tempo necessario ad accorgersi di un errore di sintassi nel nostro programma, nulla può contro gli errori logici. Eppure non ci vorrebbe molto a permettere il salvataggio di una sorta di C-Code che contenga in pratica tutto il lavoro fatto dal compilatore fino alla fine degli Include File, compresa la lista dei riferimenti esterni; vedrete comunque che, prima o poi, qualcuno troverà il modo.

Nel frattempo l'unico modo è di scrivere degli include molto piccoli e di capacità limitate: meglio dover scrivere venti {$I file X} la volta che ci servono tutte le routine che aspettare 10 minuti per la compilazione di un modulo potentissimo, ma inutile.

Routine grafiche per Turbo Pascal

di Massimo Ponzoni - Parma

Questo manuale fornisce le istruzioni per l'utilizzo della micro-library GBASE.H. Questa libreria è stata creata per poter fare del plotting scientifico (grafici in x-y) con relativa semplicità.

È vero che esiste già il modulo Turbo Graphic Tool box che permette di fare grafica in modo eccellente, ma sovente non vale la pena di tirarsi dietro tutto il sistema grafico del Tool Box solo per fare un paio di linee.

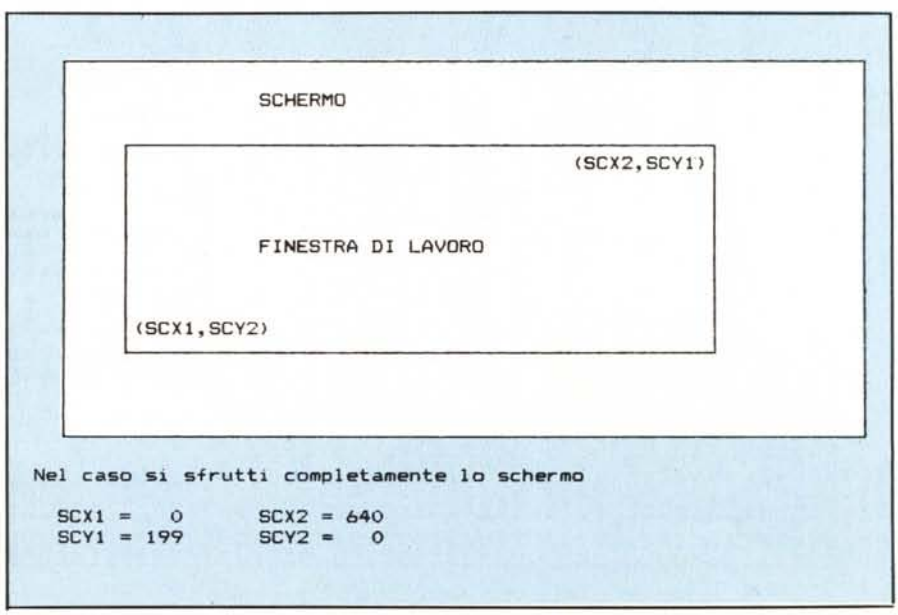
Questa libreria consente, con pochi comandi, di dimensionare automaticamente lo schermo in modo di poter contenere tutti i dati di un certo set oppure di dimensionarlo a piacimento e di disegnare assi, linee e punti in tutta comodità.

Come usare la libreria

Per poter avere a disposizione le procedure della libreria bisogna includere quest'ultima, in fase di stesura del programma, in testa al programma (appena dopo l'intestazione) tramite l'istruzione al compilatore:

```
{$I GBASE.H}
```

Così facendo si attivano le procedure descritte in seguito:



```
program Regressione;  (* --- Regressione dei minimi quadrati --- *)
                      (*  V.Massimo Ponzoni        Settembre '86 *)

{$i gbase.h }

Var
vc :dati;
i,n:integer;
x,y:dati;
xmed,ymed,a,b,r,rQuadro :real;
xBase,xTetto,yBase,yTetto, xplot,yplot : real;

(*--------------------------- calcola la media del vettore -------*)

function media(t:dati;n:integer):real;
Var
k:integer;
med:real;

   begin
   med:=0;
   for k:=1 to n do
       begin
       med:=med+t[k];
       end;
   media:=med/n;
   end;

(*-----------------calcola la somma del prodotto degli elementi --- *)

function sommaprod(t,s:dati;n:integer):real;
Var
k:integer;
sp:real;

   begin
   sp:=0;
   for k:=1 to n do
       begin
       sp:=sp+t[k]*s[k];
       end;
   sommaprod:=sp/n;
   end;


(* ---------------------------- Corpo principale del programma -------------*)

begin
ClrScr;
Writeln('');
Writeln('-------------------------------------------------------------------------------');
Writeln('+    Regressione dati a retta dei Minimi Quadrati  ( max 100 coppie di dati ) +');
Writeln('-------------------------------------------------------------------------------');
Writeln('');
Write('Immetti il numero dei dati  ');read(n);writeln('');Writeln('');
for i:=1 to n do
    begin
    write('X(',i,') = ');read(x[i]);
    Write('    Y(',i,') = ');read(y[i]);
    Writeln('');
    Writeln(LST,'X(',i,') = ',x[i],'Y(',i,') = ',y[i]);

    end;
xmed:=media(x,n);
ymed:=media(y,n);
a:=(sommaprod(x,y,n)-xmed*ymed)/(sommaprod(x,x,n)-xmed*xmed);
b:=ymed-a*xmed;
r:=(sommaprod(x,y,n)-xmed*ymed)/(Sqrt((sommaprod(x,x,n)-xmed*xmed)*(sommaprod(y,y,n)-ymed*ymed)));
rQuadro:=r*r;
writeln(LST,'a= ',a);
Writeln(LST,'b= ',b);
Writeln(LST,'r²= ',rQuadro:8:5);
(*--------- fine calcolo parametri statistici ----------------------*)

(*----------Inizio calcolo e disegno del grafico --------------------*)
IniGraf;
Ottimassi(x,y,n,True);
xBase:=BASE;
xTetto:=TETTO;
for i:=1 to n do
    begin
    Quad(x[i],y[i],2);
    end;
for i:=1 to 500 do
    begin
    xplot:=xBase+i*(xTetto-xBase)/500;
    yplot:=a*xplot+b;
    Punto(xplot,yplot);
    end;
repeat until Keypressed;
end.
```

procedure INIGRAF
{non necessita di parametri}

Inizializza lo schermo grafico (in alta risoluzione 640 × 200, ma è possibile con modifiche minime settare il tutto in bassa risoluzione per l'utilizzo dei colori) e fissa gli estremi (SCX1, SCX2, SCY1 e SCY2) della finestra che coincidono con gli estremi massimi dello schermo. Si possono sempre diminuire le dimensioni dello schermo assegnando opportuni parametri alle variabili SCX1, SCX2, SCY1 e SCY2 (vedi la figura in prima pagina per riferimento). Procedure SISTEMA (xb, yb, xt, yt: real).

Crea una corrispondenza biunivoca tra un certo dominio rettangolare

R = (x,y) tali che xb < x < xt
e yb < y < yt

e la finestra definita da Inigraf (o dall'utente). Ad esempio se si volesse disegnare una parabola

$y = x^2$ con $-2 < x < 2$

si opererebbe con

Sistema (−2,0,2,4)

Inoltre tramite questa procedura viene fissato lo step ottimale per i loop onde evitare che vengano calcolati valori non plottabili per i limiti del computer (risulterebbe inutile calcolare 2000 punti x-y per disegnare una retta, quando si dispone al massimo di 640 punti indirizzabili !!!).

Procedure DELTA
(x: dati; n: integer)

Determina un range in cui gli n elementi del vettore di reali x stanno «comodi». Inoltre il range calcolato sempre un multiplo di una «unità» comoda da leggere in fase di disegno del grafico (ad esempio 2,5 * 10^ −2 invece che 0,02576589).

In questa procedura vengono assegnati alcuni valori a determinate variabili globali:

PW = contiene la caratteristica logaritmica di Xmax-Xmin che viene usata per poter ridurre i numeri alla forma X.XX * 10^ PW.

BASE = contiene la base del range ottimale calcolato col criterio di cui sopra.

TETTO = contiene il valore massimo del range calcolato.

Procedure OTTIMASSI
(x,y: dati; n: integer; auto: boolean)

Se auto=true calcola, tramite al procedure delta, e disegna un sistema di assi ottimizzato per plottare il set di dati contenuto nei vettori x e y.

```
(*************************************************************************)
(*          Routine grafiche di base per plotting scientifico            *)
(*                  Max Ponzoni  Settembre  1986                         *)
(*************************************************************************)

Type
   dati=array[1..100] of real;

Var
   A_X,A_Y,B_X,B_Y :real;
   H_X,H_Y:real;
   X_BOT,X_TOP,Y_TOP,Y_BOT :real;
   PW,PASSO,BASE,TETTO :real;
   SCY1,SCY2,SCX1,SCX2 :integer;

(*************************************************************************)

Procedure IniGraf;                    {---------------------------------------}
begin                                 { inizializza lo schermo grafico e setta }
  HiRes;                              { i parametri per l'utilizzo pieno dello }
  HiResColor(14);                     { schermo.Questi possono comunque essere }
  SCY1:=0;                            { corretti ( vedi procedure  Ottimassi ) }
  SCY2:=199;                          {---------------------------------------}
  SCX1:=0;
  SCX2:=639;
end;

(*************************************************************************)

Procedure Sistema(xb,yb,xt,yt:real);          {-----------------------------------}
begin                                         { crea i parametri di conversione }
  X_BOT:=xb;                                  { da coordinate desiderate alle   }
  X_TOP:=xt;                                  { coordinate assolute del computer}
  Y_BOT:=yb;                                  { i fattori A_.... sono i coeff.  }
  Y_TOP:=yt;                                  { angolari. I B_... le intercette }
  A_X:=(SCX2-SCX1)/(X_TOP-X_BOT);             { Gli H_... sono gli incrementi   }
  B_X:=X_BOT*(SCX1-SCX2)/(X_TOP-X_BOT)+SCX1;  { ottimizzati per i loop          }
  A_Y:=(SCY2-SCY1)/(Y_BOT-Y_TOP);             {-----------------------------------}
  B_Y:=Y_TOP*(SCY2-SCY1)/(Y_TOP-Y_BOT)+SCY1;
  H_X:=(X_TOP-X_BOT)/(SCX2-SCX1);
  H_Y:=(Y_TOP-Y_BOT)/(SCY2-SCY1);
end;

(*************************************************************************)

Procedure Punto(x,y:real);            {---------------------------------------}
Var                                   { accende il pixel alle coordinate      }
   ics,ipsilon :integer;              { definite dalla procedure Sistema      }
begin                                 {---------------------------------------}
  ics:=round(A_X*x+B_X);
  ipsilon:=round(A_Y*y+B_Y);
  plot(ics,ipsilon,7);
end;

(*************************************************************************)

Procedure Linea(xi,yi,xf,yf:real);    {---------------------------------------}
                                      { Tira una linea dal punto (xi,yi) al   }
var                                   { punto (xf,yf).Entrambe le coppie sono }
   iniX,iniY,fineX,fineY :integer;    { riferite alle coordinate Sistema      }
begin                                 {---------------------------------------}
  iniX:=round(A_X*xi+B_X);
  iniY:=round(A_Y*yi+B_Y);
  fineX:=round(A_X*xf+B_X);
  fineY:=round(A_Y*yf+B_Y);
  Draw(iniX,iniY,fineX,fineY,7);
end;

(*************************************************************************)

Procedure Quad(x,y:real;scala:integer);   {---------------------------------}
                                          { Disegna un quadratino alle coor- }
var                                       { dinate x,y (definite da sistema) }
   ics,ips :integer;                      { La grandezza << SCALA >> defini- }
                                          { sce l'ampiezza del quadratino    }
begin                                     {---------------------------------}
  ics:=round(A_X*x+B_X);
  ips:=round(A_Y*y+B_Y);
  Draw(ics-scala*2,ips-scala,ics+scala*2,ips-scala,1);
  Draw(ics+scala*2,ips-scala,ics+scala*2,ips+scala,1);
  Draw(ics+scala*2,ips+scala,ics-scala*2,ips+scala,1);
  Draw(ics-scala*2,ips+scala,ics-scala*2,ips-scala,1);
end;

(*************************************************************************)

Procedure Rombo(x,y:real;scala:integer);  {---------------------------------}
                                          { Disegna   un rombo   alle coor- }
var                                       { dinate x,y (definite da sistema) }
   ics,ips :integer;                      { La grandezza << SCALA >> defini- }
                                          { sce l'ampiezza del rombo         }
begin                                     {---------------------------------}
  ics:=round(A_X*x+B_X);
  ips:=round(A_Y*y+B_Y);
  Draw(ics-scala*2,ips,ics,ips+scala,1);
  Draw(ics,ips+scala,ics+scala*2,ips,1);
  Draw(ics+scala*2,ips,ics,ips-scala,1);
  Draw(ics,ips-scala,ics-scala*2,ips,1);
end;
```

```
{*****************************************************************************}

Procedure Croce(x,y:real;scala:integer);  {-----------------------------------}
                                          { Disegna una croce     alle coor- }
var                                       { dinate x,y (definite da sistema) }
   ics,ips :integer;                      { La grandezza << SCALA >> defini- }
                                          { sce l'ampiezza della croce       }
begin                                     {-----------------------------------}
  ics:=round(A_X*x+B_X);
  ips:=round(A_Y*y+B_Y);
  Draw(ics+scala*2,ips,ics-scala*2,ips,1);
  Draw(ics,ips+scala,ics,ips-scala,1);
end;

{*****************************************************************************}

procedure delta(x:dati;n:integer);        {-----------------------------------}
                                          { Determina il migliore passo per una }
Var                                       { suddivisione degli assi in 10 step, }
                                          { in modo che l'intervallo dei dati   }
k :integer;                               { stia nel range (RANGE =10 * PASSO)  }
diffe,klog :real;                         { Inoltre viene determinata la caratte- }
xMax,xMin :real;                          { ristica logaritmica per ridurre i   }
                                          { numeri alla forma x.xx * 10^yyy     }
begin                                     {-----------------------------------}
  klog:=Ln(10.0);
  xMax:=x[n];
  xMin:=x[1];
  for k:=1 to n do begin
                     if xmax<x[k] then xmax:=x[k];
                     if xmin>x[k] then xmin:=x[k];
                   end;
  diffe:=xmax-xmin;
  PW:=int(Ln(diffe)/klog);
  PASSO:=int((diffe*Exp((klog*(1-PW)))/9+0.5))*Exp(klog*(PW-1));
  if (xMin>0) then BASE:=int(xMin*Exp(klog*(1-PW)))*Exp(klog*(PW-1))
              else BASE:=-1.0*int((0.1+Abs(xMin))*Exp(klog*(1-PW)))*Exp(klog*(PW-1));
  TETTO:=BASE+10.0*PASSO;
end;
{*****************************************************************************}

Procedure Ottimassi(x,y:dati;n:integer;auto:boolean);

var
   i,xc,yc:integer;
   pwx,passox,basex,valorex,      {-----------------------------------------}
   pwy,passoy,basey,valorey,      { Ottimizza (se richiesto) il range dei }
   klog :real;                    { dati tramite la procedure delta e     }
                                  { dimensiona lo schermo alle coordinate }
begin                             { ottimali .Inoltre disegna gli assi e  }
  klog:=Ln(10.0);                 { fa in modo che le divisioni siano fra-}
  SCX1:=40;                       { zioni decimali.In altenativa : se si  }
  SCX2:=600;                      { mette False nella variabile auto, il  }
  SCY1:=23;                       { range è contenuto negli ultimi due    }
  SCY2:=183;                      { elementi del vettore type dati. Questo}
  Draw(40,183,600,183,1);         { serve quando si vuole imporre un certo}
  Draw(40,183,40,23,1);           { range che non corrrisponde a quello   }
  Draw(600,183,590,180,1);        { ottimale calcolato, ad esempio quando }
  Draw(600,183,590,186,1);        { si vogliono confrontare grafici i cui }
  Draw(40,23,32,30,1);            { range calcolati sono diversi.         }
  Draw(40,23,48,30,1);            {-----------------------------------------}
  if auto then begin
                 delta(y,n);
                 pwy:=PW;
                 basey:=BASE;
                 passoy:=PASSO;
                 delta(x,n);
                 pwx:=PW;
                 basex:=BASE;
                 passox:=PASSO;
               end
          else begin
                 basex:=(x[n]-x[n-1]);
                 passox:=basex/10;
                 pwx:=int(Ln(basex)/klog);
                 basex:=x[n-1];
                 BASE:=basex; TETTO:=BASE+10.0*passox;
                 basey:=(y[n]-y[n-1]);
                 passoy:=basey/10;
                 pwy:=int(Ln(basey)/klog);
                 basey:=y[n-1];
               end;
  Sistema(basex,basey,basex+10*passox,basey+10*passoy);
  for i:=0 to 9 do begin
                     xc:=40+56*i;
                     Draw(xc,183,xc,186,1);
                     GotoXY(4+i*7,25);
                     valorex:=(basex+i*passox)/Exp(klog*pwx);
                     Write(valorex:4:2);
                     GotoXY(1,23-i*2);
                     valorey:=(basey+i*passoy)/Exp(klog*pwy);
                     Write(valorey:4:2);
                     yc:=183-16*i;
                     Draw(40,yc,36,yc,1)
                   end;
  yc:=trunc(pwy);
  GotoXY(1,2);Write('x10');
  GotoXY(4,1);Write(yc);
  xc:=trunc(pwx);
  GotoXY(76,25);Write('x10');
  GotoXY(79,24);Write(xc);
end;
```

In alternativa (se auto = false) disegna un sistema di assi dettato dall'utente in cui Xmax, Xmin, Ymax Ymin non vengono calcolati da delta, bensì sono contenuti negli ultimi due elementi dei rispettivi vettori x e y (type dati).

Questa opzione si rende necessaria qualora si vogliano confrontare due grafici i cui range calcolati non coincidano.

Esempio:

Ottimassi (lcs, lps, 15, True)
disegna un sistema di assi ottimale per le 15 coppie di dati (lcs, lps)

Ottimassi (lcs, lps, 2, False)
disegna un sistema di assi i cui range sono:
lcs[1] < x < lcs[2] e
lps[1] < y < lps[2]

Procedure
PUNTO (x,y: real)
QUAD (x,y: real; scala: integer)
ROMBO (x,y: real; scala: integer)
CROCE (x,y: real; scala: integer)

Disegnano un punto (o un quadratino o una croce o un rombo) alle coordinate (x,y). Le coordinate sono espresse nel sistema definito dalla Procedure Sistema per cui l'uso di queste Procedure deve essere successivo alla chiamata della Procedure Sistema.

Nel caso di Quad, Rombo e Croce la variabile Scala amplifica la grandezza dei disegni.

Esempio:

Quad (2,3,4) produce un quadratino alle coordinate (2,3)

mentre

Quad (2,3,8) produce un quadratino alle stesse coordinate, ma grande il doppio.

Procedure LINEA (xi,yi,xf,yf: real)

Tira una linea dal punto (xi,yi) al punto (xf,yf). Le coordinate sono espresse nel sistema definito nella Procedure Sistema.

Hardware e software necessario:

IBM PC o compatibile con almeno un floppy disk drive;

Scheda grafica CGA (con altre si deve modificare la library);

Stampante (opzionale);

MS-DOS 2.00 o successivi;

TURBO PASCAL.

Il listato di pagina 237 contiene REGRESS.PAS: un programma dimostrativo che sfrutta la libreria grafica per disegnare la regressione dei minimi quadrati su un set massimo di cento coppie di dati (x-y).

MC