```
{ Program pentru generarea de grafice de functii in fisiere text. }
program Grafice_de_functii_in_fisiere_text;

uses Crt;

const car_linie = 40;

type
    { Definim un tip procedural pentru functii matematice.
      Vom lucra doar cu functii matematice care au un singur argument
      real si care returneaza o valoare reala. }
    FunctiiMatematice = function(x:real) : real;

    { Definim si un tip de date vector de functii matematice.
      Vom folosi o variabila de acest tip pentru a pastra o lista
      de functii ale caror grafice vrem sa le generam. }
    ListaFunctii = array[1..10] of FunctiiMatematice;

var
   { Numarul functiilor la care le desenam graficele. }
   NrFunctii : Integer;

   { O lista de functii la care le desenam graficele. Aceasta variabila
     este de fapt un vector de un tip procedural. Fiecare intrare din
     acest vector va retine adresa unei functii din programul nostru,
     functie al carei grafic va fi generat intr-un fisier text pe disc. }
   functii : ListaFunctii;

   { Variabila fisier pe care o folosim la scrierea in fisiere pe disc.
     Lucram cu o singura variabila, dar vom vedea ca se vor genera mai
     multe grafice si ca fiecare grafic va fi generat in alt fisier pe
     disc. Ca urmare aceasta unica variabila va fi refolosita de la un
     fisier la altul. }
   f : text;

   { In acest vector pastram numele fisierelor in care se vor genera
     graficele. }
   nume_fisiere : array[1..100] of string[20];

   { Limitele intervalului pe care se reprezinta graficele. Acestea
     vor fi [0, 2*PI]. }
   lim_inf, lim_sup : real;

   { Numarul de pasi cu care se reprezinta graficul. Acesta se citeste
     de la tastatura. Cu cat este mai mare, cu atat mai precis va
     fi reprezentat graficul. }
   pasi : integer;

   { Variabila folosita pentru iteratii. }
   i : integer;


{ Urmeaza definirea functiilor ale caror grafice le vom reprezenta.
  Observati ca functiile sunt include intre directivele de compilare
  SF+...$F-, deoarece adresele lor vor fi atribuite unor variabile
  procedurale. Mai exact adresele acestor functii vor fi pastrate in
  vectorul "functii". }
{$F+}
function MySin(x : Real) : Real;
begin
     MySin := sin(x);
```

```
end;

function MyCos(x : Real) : Real;
begin
     MyCos := cos(x);
end;

function MySinCos(x : Real) : Real;
begin
     MySinCos := sin(x) * cos(x);
end;
{$F-}


{ Subprogram de initializare. Aici setam numarul de functii, pastram
  adresele functiilor in vectorul "functii", stabilim numele fisierelor
  in care sa se genereze graficele, stabilim limitele intervalului
  pe care desenam graficul. }
procedure Initializare;
begin
     ClrScr;

     { Avem trei functii la care vrem sa le reprezentam graficele. }
     NrFunctii := 3;

     { Prima functie este "MySin" si graficul se va scrie in fisierul
       "sin.txt". }
     functii[1] := MySin; nume_fisiere[1] := 'sin.txt';

     { A doua functie este "MyCos" si graficul se va scrie in fisierul
       "cos.txt". }
     functii[2] := MyCos; nume_fisiere[2] := 'cos.txt';

     { Iar a treia functie este "MySinCos" si graficul ei se va scrie
       in fisierul "sincos.txt". }
     functii[3] := MySinCos; nume_fisiere[3] := 'sincos.txt';

     { Limitele intervalului vor fi [0, 2*PI]. }
     lim_inf := 0;
     lim_sup := 2 * Pi;

     { Si citim de la tastatura numarul de pasi (adica precizia) cu
       care se deseneaza graficele. }
     Write('Introduceti numarul de pasi:'); ReadLn(pasi);
end;


{ Pentru o anumita functie matematica (transmisa prin parametrul
  "fnc" ca variabila procedurala), se calculeaza valoarea minima
  si valoarea maxima pe intervalul dat de variabilele globale
  "lim_inf" si "lim_sup". Avem nevoie de minim si de maxim pentru
  a putea reprezenta graficul. }
procedure CalculMinMax(fnc : FunctiiMatematice; var min, max : real);
var i : integer;
    val : real;
    x : real;
begin
     min := fnc(lim_inf);
     max := fnc(lim_sup);
     for i := 1 to pasi do
         begin
```

```
              x := lim_inf + (lim_sup-lim_inf) * i / pasi;
              val := fnc(x);
              if min > val then
                 min := val;
              if max < val then
                 max := val;
         end;
end;


procedure ScrieLinie(var fout : text; val_aprox : integer);
var i : Integer;
begin
     for i:= 0 to val_aprox-1 do write(fout, '.');
     write(fout, '*');
     for i:= val_aprox +1 to car_linie do write(fout, '.');
     writeln(fout);
end;


{ Pentru o functie matematica transmisa prin parametrul "fnc",
  genereaza graficul ei in fisierul text transmis prin parametrul
  "fout". }
procedure Grafic(var fout:text; fnc : FunctiiMatematice);
var min, max : real;
    i : integer;
    x : real;
    val : real;
    val_aprox : integer;
begin
     { Intai calculam minumul si maximul functiei. }
     CalculMinMax(fnc, min, max);

     { Apoi pentru fiecare pas... }
     for i:=0 to pasi do
         begin
              { Calculam valoarea din intervalul "lim_inf" si "lim_sup"
                care corespunde pasului curent. }
              x := lim_inf + (lim_sup-lim_inf) * i / pasi;

              { Calculam valoarea functiei in acest punct. }
              val := fnc(x);

              { Fiecare valoare a functiei o vom reprezenta in fisierul
                text pe cate o linie. O linie are latimea de "car_linie"
                caractere, definita ca si constanta in program.
                Trebuie sa calculam al catelea caracter din linie
                ar corespunde valorii functiei in acest punct. }
              val_aprox := round((val-min) * car_linie / (max-min));

              { Dupa ce am gasit al catelea caracter corespunde valorii
                functiei, scriem linia in fisier. }
              ScrieLinie(fout, val_aprox);
         end;

end;


{ Programul principal. }
begin
     { Facem initializarile. }
```

```
     Initializare;

     { Apoi luam rand pe rand fiecare functie din vectorul "functii". }
     for i := 1 to NrFunctii do
         begin
              { Cream pe disc fisierul corespunzator. }
              Assign(f, nume_fisiere[i]);
              Rewrite(f);

              { Reprezentam graficul in fisierul ce l-am creat. }
              Grafic(f, functii[i]);

              { Dupa care inchidem fisierul. Dupa cum se observa,
                o singura variabila fisier este refolosita pentru
                generarea tuturor graficelor. }
              Close(f);
              WriteLn('Fisierul ''', nume_fisiere[i], ''' a fost scris.');
         end;
end.
```