unit uMNK;

interface

uses

math, Generics.Collections ;

type

d2array = array of array of double;

// матрицы записываются по столбцам, т.е. Vcol1 ,..Vcoln где вектор компонентов состоящий из элементов row m[0] = колонка V0

// Транспонирование

// каждая строка становится строкой

function transpondmatrix(m:d2array):d2array;

// Детерминант. Определен только для квадратной матрицы

function DeterminantGauss(a:d2array):double;

function Invers(m:d2array):d2array;

procedure MultMatrix(m1,m2:d2array; var m3:d2array);

// n - число точек в сигнале m - порядок интерполяции

procedure Gram (n,m:integer; const x,f:array of double; var a:d2array);overload;

procedure Gram (n,m:integer; const x,f:olevariant; var a:d2array);overload;

procedure Gram (n,m:integer; const f:olevariant; x0,dx:double;var a:d2array);overload;

// m - степень полинома; c - массив коэффициентов полинома; x, y входные данные

procedure buildMNK(m:integer; const x, y: array of double; var сi:array of double);overload;

// size - число точек в массиве

procedure buildMNK(m:integer; x, y: olevariant; size:integer; var сi:array of double);overload;

procedure buildMNK(m:integer; y: olevariant; x0,dx:double; size:integer; var сi:array of double);overload;

//procedure buildMNK(m:integer; const x, y: olevariant; var сi:array of double);

// посчитать значение полинома в точке x

function fi (m:integer; var c:array of double; x:real):real;overload;

function fi (var c:array of double; x:real):real;overload;

implementation

// a11 = m1\_row1\*m2\_col1

procedure MultMatrix(m1,m2:d2array; var m3:d2array);

var

i, j, k,

// размерность матриц

m1d1, m1d2, m2d1, m2d2:integer;

begin

m1d1:=length(m1);

m1d2:=length(m1[0]);

m2d1:=length(m2);

m2d2:=length(m2[0]);

// у результата число строк равно числу строк первой матрицы и числу колонок второй

setlength(m3, m2d1, m1d2);

// цикл по строкам первой матрицы

for i:=0 to m1d2-1 do

begin

// цикл по колонкам второй матрицы

for j:=0 to m2d1-1 do

begin

m3[j,i]:=0;

for k:=0 to m1d1-1 do

begin

m3[j,i]:=m3[j,i] + m1[k,i] \* m2[j,k];

end;

end;

end;

end;

function Invers(m:d2array):d2array;

var

n,k, i, j: integer;

b, obr: d2array;

begin

n:=Length(m);

setlength(b,n,n);

setlength(obr,n,n);

for I := 0 to n - 1 do

begin

for j := 0 to n - 1 do

begin

obr[i,j]:=m[i,j];

end;

end;

for k:=0 to n - 1 do

begin

for i:=0 to n - 1 do

for j:=0 to n - 1 do

begin

if (i=k) and (j=k) then

b[i,j] := 1/obr[i,j];

if (i=k) and (j<>k) then

b[i,j] := -obr[i,j]/obr[k,k];

if (i<>k) and (j=k) then

b[i,j] := obr[i,k]/obr[k,k];

if (i<>k) and (j<>k) then

b[i,j] := obr[i,j] - obr[k,j] \* obr[i,k]/obr[k,k];

end;

for i:= 0 to n - 1 do

for j:= 0 to n - 1 do obr[i, j]:= b[i, j];

end;

result:=obr;

end;

procedure Swap(k,n:integer;var a:d2array; var p:integer);

//перестановка строк, если главный элемент=0

var

z:Real;j,i:integer;

begin

z:=abs(a[k,k]);i:=k;p:=0;

for j:=k+1 to n-1 do

begin

if abs(a[j,k])>z then

begin

z:=abs(a[j,k]);i:=j;

p:=p+1;//счетчик перестановок

end;

end;

if i>k then

begin

for j:=k to n-1 do

begin

z:=a[i,j];

a[i,j]:=a[k,j];

a[k,j]:=z;

end;

end;

end;

function Znak(p:integer):integer;

//при перестановке меняется знак определителя, надо его отследить

begin

if p mod 2=0 then

result:=1

else

result:=-1;

end;

function DeterminantGauss(a:d2array):double;

var

k,i,j,p,n:integer;r:real;

buf:d2array;

begin

result:=1.0;

n:=length(a);

setlength(buf,n,n);

for I := 0 to n - 1 do

begin

for j := 0 to n - 1 do

begin

buf[i,j]:=a[i,j];

end;

end;

for k:=0 to n-1 do

begin

if buf[k,k]=0 then

swap(k,n,buf,p);//перестановка строк

result:=znak(p)\*result\*buf[k,k];//вычисление определителя

for j:=k+1 to n-1 do //пересчет коэффициентов

begin

r:=buf[j,k]/buf[k,k];

for i:=k to n-1 do

buf[j,i]:=buf[j,i]-r\*buf[k,i];

end;

end;

end;

function transpondmatrix(m:d2array):d2array;

var

r, c,dim1,dim2: Integer;

begin

dim1:=length(m);

dim2:=length(m[0]);

setlength(result, dim2, dim1);

for r := 0 to dim1-1 do

begin

for c := 0 to dim2 - 1 do

begin

result[c,r]:=m[r,c];

end;

end;

end;

// показательная функция для формирования матрицы Грамма

// нужна для решения системы уравнений

function ex (a:real; n:integer):real;

var i:integer;

e:real;

begin

e:=1;

for i:=1 to n do

e:=e\*a;

ex:=e;

end;

procedure Gram (n,m:integer; const x,f:olevariant; var a:d2array);

var

i,j:integer;

p,q,r,s:real;

begin

for j:=0 to m do

begin

s:=0; r:=0; q:=0;

for i:=0 to n-1 do

begin

p:=ex(x[i],j);

s:=s+p;

r:=r+p\*f[i];

// сумма x

q:=q+p\*ex(x[i],m);

end;

a[0,j]:=s;

a[j,m]:=q;

a[j,m+1]:=r;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

end;

procedure Gram (n,m:integer; const x,f:array of double; var a:d2array);

var

i,j:integer;

p,q,r,s:real;

begin

for j:=0 to m do

begin

s:=0; r:=0; q:=0;

for i:=0 to n-1 do

begin

p:=ex(x[i],j);

s:=s+p;

r:=r+p\*f[i];

// сумма x

q:=q+p\*ex(x[i],m);

end;

a[0,j]:=s;

a[j,m]:=q;

a[j,m+1]:=r;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

end;

procedure Gram (n,m:integer; const f:olevariant; x0,dx:double;var a:d2array);overload;

var

i,j:integer;

x,p,q,r,s:real;

begin

for j:=0 to m do

begin

s:=0; r:=0; q:=0;

for i:=0 to n-1 do

begin

// сумма x

x:=x0+dx\*i;

p:=ex(x,j);

s:=s+p;

r:=r+p\*f[i];

q:=q+p\*ex(x,m);

end;

a[0,j]:=s;

a[j,m]:=q;

a[j,m+1]:=r;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

for i:=1 to m do

begin

for j:=0 to m-1 do

begin

a[i,j]:=a[i-1,j+1];

end;

end;

end;

// решение симметричной матрицы методом гаусса. x - решение вектор 0..n

// a - матрица грамма, n - степень полинома

procedure Gauss(n:integer; var a:d2array; var x:array of double);

var

i,j,k,l,k1,n1:integer;

r,s:real;

begin

n1:=n+1;

for k:=0 to n do

begin

k1:=k+1;

s:=a[k,k];

for j:=k1 to n1 do a[k,j]:=a[k,j]/s;

for i:=k1 to n do

begin

r:=a[i,k];

for j:=k1 to n1 do

a[i,j]:=a[i,j]-a[k,j]\*r;

end;

end;

for i:=n downto 0 do

begin

s:=a[i,n1];

for j:=i+1 to n do

s:=s-a[i,j]\*x[j];

x[i]:=s;

end;

end;

// значение полинома c - значения полинома x1 - x вычисляемого узла

// m - степень полинома

function fi (m:integer; var c:array of double; x:real):real;

var i:integer; p:real;

begin

p:=c[m];

for i:=m-1 downto 0 do

p:=c[i]+x\*p;

fi:=p;

end;

function fi (var c:array of double; x:real):real;

var

m:integer;

begin

m:=length(c);

result:=fi(m,c,x);

end;

procedure buildMNK(m:integer; const x, y: array of double; var сi:array of double);

var

// загадочный член обозначающий число узлов (ИНТЕРПОЛЯЦИИ????!!!!)

n:integer;

// матрица к грамма

a:d2array;

I, j: Integer;

begin

n:=length(x);

// непонятно почему но поставил +2 а не +1 с запасом. иначе вываливается инвалид поинтер на домашней делфи

// на работе норм работает с +1

setlength(a,m+2,m+2);

// n - длина массива

Gram (n, m, x, y, a);

// заполнение своб коэф.

for I := 0 to m do

begin

for j := 0 to n - 1 do

begin

сi[i]:=сi[i]+Power(x[j],i)\*y[j];

end;

end;

Gauss (m,a,сi);

end;

procedure buildMNK(m:integer; x, y: olevariant; size:integer; var сi:array of double);

var

// размер сигнала

n:integer;

// матрица к грамма

a:d2array;

I, j: Integer;

begin

n:=size;

setlength(a,m+2,m+2);

Gram (n, m, x, y, a);

// заполнение своб коэф.

for I := 0 to m do

begin

for j := 0 to n - 1 do

begin

сi[i]:=сi[i]+Power(x[j],i)\*y[j];

end;

end;

Gauss (m,a,сi);

end;

procedure buildMNK(m:integer; y: olevariant; x0,dx:double; size:integer; var сi:array of double);overload;

var

// размер сигнала

n:integer;

// матрица к грамма

a:d2array;

I, j: Integer;

begin

n:=size;

setlength(a,m+2,m+2);

Gram (n, m, x0,dx, y, a);

// заполнение своб коэф.

for I := 0 to m do

begin

for j := 0 to n - 1 do

begin

сi[i]:=сi[i]+Power((x0+dx\*j),i)\*y[j];

end;

end;

Gauss (m,a,сi);

end;

end.