Date de intrare:

* n – tip int, citit de la tastatură, ordinul matricilor
* A[i,j], i=0, n-1 ; j=0,n-1 – elemente de tip double, citită de la tastatură, matricea coeficienților
* B[i], i=0, n-1 – elemente de tip double, citită de la tastatură, matricea unidimensională a termenilor liberi

Date intermediare:

* Matrix G[] - listă de matrici G (matrici de rotație)
* int k – numărul de matrici G (echivalent cu numărul de elemente “0” de sub diagonala principală)
* Matrix Q – matricea ortogonală care satisface relația QRx = B, unde R s-a calculat ca fiind matricea triunghiulară superior și Q s-a calculat ca fiind matricile G transpuse si înmulțite

Date de ieșire:

* X[i], i=0, n-1 – tip double, matricea valorilor necunoscutelor

Observații:

* ordinul matricii se poate deduce în timpul citirii, deci nu este necesar să-l primim ca date de intrare în mod explicit

Algoritmul:

Observații:

* algoritmul în C# verifică și dacă matricea A este pătratică și dacă B (matricea termenilor liberi) are același număr de linii cu matricea A
* algoritmul în C# abstractizează o matrice în tipul *Matrix* în care s-au implementat operațiile necesare pe matrici (transpunere, produs de două matrici) de aceea implementarile funcțiilor produsMatrici(X, Y) și transpuneMatrice(X) nu sunt prezentate.
* produsMatrici(X, Y) returnează o matrice care reprezintă rezultatul înmulțirii lui X și Y
* transpuneMatrice(X) transpune X-ul

Algoritmul în pseudocod:

Variabile globale:

k = 0;

Matrix G[];

Matrix Q, A, B, X;

Funcția “main”:

pentru j = 0, j < n – 1, j++, execută:

pentru i = n-1, i > j, i--, execută:  
 dacă 0 != Ai,j atunci:  
 G[k] = generateG(i, j);   
 A = produsMatrici(G[k], A);

k++;  
 sfârșit dacă.  
 sfârșit pentru.  
sfârșit pentru.  
  
dacă k > 0 atunci:   
 computeQ();   
altfel   
 pentru i=0, i<n, i++, execută:   
 Qi,i = 1;  
 sfârșit pentru.  
sfârșit dacă.  
   
transpuneMatrice(Q);  
  
Matrix Qb = produsMatrici(Q, B);  
   
pentru i=n–1, i>=0, i--, execută:  
 double S = 0;  
  
 pentru j=i+1, j<n, j++, execută:  
 S += Xj \* Ai,j

sfârșit pentru.  
   
 Xi =;  
sfârșit pentru.

Funcția “generateG” (i, j parametrii, poziția pe linie și pe coloană a elementului de făcut “0” în A):

double a = Ai-1,j;  
double b = Ai,j;  
double r = ;  
double c = ;  
double s = ;  
Matrix g;  
  
pentru p=0, p<n, p++, execută:  
 g[p, p] = 1;  
sfârșit pentru.  
  
g[i, i] = g[i-1, i-1] = c;  
g[i, i-1] = s;  
g[i-1, i] = -s;  
  
returnează g;

Funcția “computeQ”:  
transpuneMatrice(G0);  
Q = G0;  
pentru i=1, i<k, i++, execută:  
 transpuneMatrice(Gi);  
 Q = Q \* Gi;  
sfârșit pentru.