

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <time.h>
5  #define epsilon 1e-12
6  void addmat(double *A, double *B, double *C, int m, int n)
7  { // sabiranje matrica A mXn + B mXn = C mXn
8      int i;
9
10     for(i=0;i<m*n;i++){
11         C[i] = A[i]+B[i];
12     }
13 }
14 void multmat(double *A, double *B, double *C,
15             int m, int p, int n)
16 { // mnozenje matrica A mXp * B pXn = C mXn
17     int i,j,k;
18
19     for(i=0;i<m;i++){
20         for(j=0;j<n;j++){
21             C[i*n+j] = 0;
22             for(k=0;k<p;k++){
23                 C[i*n+j] = C[i*n+j]+A[i*p+k]*B[k*n+j];
24             }
25         }
26     }
27 void multscal(double x, double *A, double *B, int m, int n)
28 { // mnozenje matrica x * A mXn = B mXn
29     int i;
30
31     for(i=0;i<m*n;i++){
32         B[i] = x*A[i];
33     }
34 void transpose(double *A, double *B, int m, int n)
35 { // transpose( A mXn ) = B nXm
36     int i,j;
37
38     for(i=0;i<m;i++){
39         for(j=0;j<n;j++){
40             B[j*m+i] = A[i*n+j];
41         }
42     }
43 }
44 int rowpivot(double *A, int n, int m)
45 { // pivotizuje vrstu m sa il, vraca -1 ako rang<n
46     int i,j,il=m;
47     double temp;
48     for(i=m+1;i<n;i++){
49         if(fabs(A[i*n+m])>fabs(A[il*n+m])){
50             il = i;
51         }
52     }
53     if(fabs(A[il*n+m])<epsilon)
54         return -1;
55     if(il!=m){
56         for(j=m;j<n;j++){
57             temp=A[il*n+j]; A[il*n+j]=A[m*n+j]; A[m*n+j]=temp;
58         }
59     }
60     return il;
61 }
62 int inverse(double *A, double *B, int n)
63 { // A^(-1) = B (menja A) daje n-rang(A). rang(A)<n => nema inv
64     int i,j,k,il;
65     double alpha;
66     for(i=0;i<n;i++){

```

```

67     for(j=0;j<n;j++){
68         B[i*n+j] = (double)(i==j);
69 // Gausove eliminacije
70     for(i=0;i<n-1;i++){
71         if((il=rowpivot(A,n,i))== -1){
72             return (n-i); // rang je i, A nema inverznu
73         }
74         for(j=0;j<n;j++){
75             alpha = B[i*n+j]; B[i*n+j] = B[il*n+j];
76             B[il*n+j] = alpha;
77         }
78         for(k=i+1;k<n;k++){
79             alpha = A[k*n+i]/A[i*n+i];
80             A[k*n+i] = 0;
81             for(j=i+1;j<n;j++){
82                 A[k*n+j] = A[k*n+j] - alpha*A[i*n+j];
83             }
84             for(j=0;j<n;j++){
85                 B[k*n+j] = B[k*n+j] - alpha*B[i*n+j];
86             }
87         }
88         if(fabs(A[n*n-1])<epsilon)
89             return 1; // rang je n-1, A nema inverznu
90 // Zordanove eliminacije
91     for(i=n-1;i>0;i--){
92         for(k=0;k<i;k++){
93             alpha = A[k*n+i]/A[i*n+i];
94             for(j=0;j<n;j++){
95                 B[k*n+j]=B[k*n+j]-alpha*B[i*n+j];
96             }
97         }
98         for(j=0;j<n;j++){
99             B[i*n+j]=B[i*n+j]/A[i*n+i];
100         }
101     }
102     for(j=0;j<n;j++){
103         B[j]=B[j]/A[i];
104     }
105     return 0; // A ima inverznu, nalazi se u B
106 }
107 int pivot(double *A, int n, int m)
108 { // pivotizuje vrstu il / kolonu j1 sa m / m, vraca +/-1
109     int i,j,il=m,j1=m,znak=1;
110     double temp;
111     for(i=m;i<n;i++){
112         for(j=m;j<n;j++){
113             if(fabs(A[i*n+j])>fabs(A[il*n+j1])){
114                 il = i;
115                 j1 = j;
116             }
117         }
118     }
119     if(fabs(A[il*n+j1])<epsilon)
120         return 0;
121     if(il!=m){
122         znak = znak*(-1);
123         for(j=m;j<n;j++){
124             temp=A[il*n+j]; A[il*n+j]=A[m*n+j]; A[m*n+j]=temp;
125         }
126     }
127     if(j1!=m){
128         znak = znak*(-1);
129         for(i=0;i<n;i++){
130             temp=A[i*n+m]; A[i*n+m]=A[i*n+j1]; A[i*n+j1]=temp;
131         }
132     }
133     return znak;
134 }
135 double det(double *A, int n)

```

```

133 { // racuna determinantu matrice A nXn, (menja elemente A)
134     int i,j,k,pm,znak=1;
135     double alpha;
136     for(i=0;i<n-1;i++){
137         if(!(pm=pivot(A,n,i)))
138             return 0.0;
139         znak = znak*pm;
140         for(k=i+1;k<n;k++){
141             alpha = A[k*n+i]/A[i*n+i];
142             A[k*n+i] = 0;
143             for(j=i+1;j<n;j++){
144                 A[k*n+j] = A[k*n+j] - alpha*A[i*n+j];
145             }
146         }
147         if(fabs(A[n*n-1])<epsilon)
148             return 0.0;
149         alpha = znak*A[0];
150         for(i=1;i<n;i++){
151             alpha = alpha * A[i*n+i];
152         }
153     }
154 void printmatrix(double *A, int m, int n)
155 { // stampa matricu A mXn, m,n < 10
156     int i,j;
157
158     printf("\n+-");
159     for(j=0;j<m;j++) printf("          ");
160     printf("-+\n");
161     for(i=0;i<m;i++){
162         printf("| ");
163         for(j=0;j<n;j++){
164             printf(" %6.2f ",A[i*n+j]);
165             printf(" |\n");
166         }
167         printf("+-");
168         for(j=0;j<m;j++) printf("          ");
169         printf("-+\n");
170     }

```