```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Globalization;
 4  using System.Linq;
 5  using System.Text;
 6  using System.Threading.Tasks;
 7
 8  namespace kalkulator_macierze
 9  {
10      class Program
11      {
12
13          static void Main(string[] args)
14          {
15              double[,] arr1 = BuildMatrix();
16              //double[,] arr2 = BuildMatrix();
17              //MatrixMultiplication(arr1,arr2);
18              InvertedMatrix(arr1);
19              Console.ReadLine();
20          }
21
22          private static double[,] BuildMatrix() //Buduj macierz o w wierszach i k
                kolumnach
23          {
24              Console.Write("Wiersze: ");
25              int w = int.Parse(Console.ReadLine());
26              Console.Write("Kolumny: ");
27              int k = int.Parse(Console.ReadLine());
28              double[,] matrix = new double[w, k];
29              Console.WriteLine("Jak chcesz uzupełnić macierz");
30              Console.WriteLine("1. Automatycznie (Losowe wartości od 1-10)");
31              Console.WriteLine("2. Macierz jednostkowa");
32              Console.WriteLine("3. Samodzielnie");
33              string rodzaj = Console.ReadLine();
34              switch (int.Parse(rodzaj))
35              {
36                  case 1: /*losowe*/
37                      for (int i = 0; i < w; i++)
38                      {
39                          for (int j = 0; j < k; j++)
40                          {
41                              Random rand = new Random();
42                              for (int z = 0; z < 2000000; z++)
43                              {
44                                  matrix[i, j] = rand.Next(1, 10);
45                              }
46                              Console.Write(matrix[i, j] + " ");
47
48                          }
```

```
49                          Console.WriteLine("");
50                      }
51                      return matrix;
52                  case 2: /*jednostkowa*/
53                      for (int i = 0; i < w; i++)
54                      {
55                          matrix[i, i] = 1.0;
56                      }
57                      return matrix;
58                  case 3:
59                      for (int i = 0; i < matrix.GetLength(0); i++)
60                      {
61                          Console.WriteLine("Wiersz nr " + (i+1) + ": ");
62                          for (int j = 0; j < matrix.GetLength(1); j++)
63                          {
64                              Console.WriteLine("Wartość nr " + (j+1) + ": ");
65                              matrix[i, j] = double.Parse(Console.ReadLine());
66                          }
67                      }
68                      return matrix;
69              }
70
71              for (int i = 0; i < w; i++)
72              {
73                  int x = i;
74                  for (int j = 0; j < k; j++)
75                  {
76                      Random rand = new Random();
77                      for (int z = 0; z < 200000; z++)
78                      {
79                          matrix[i, j] = rand.Next(1, 10);
80                      }
81                      Console.Write(matrix[i, j] + " ");
82                      /*
83                      matrix[i, j] = x++ + 1;
84                      Console.Write(matrix[i, j] + " ");
85                      */
86                  }
87                  Console.WriteLine("");
88              }
89              return matrix;
90          }
91          private static void MatrixMultiplication(double[,] arr1, double[,]
              arr2) //Mnożenie macierzy
92          {
93              if (arr1.GetLength(1) != arr2.GetLength(0)) //Sprawdz czy dzialanie
                  jest mozliwe
94              {
95                  Console.Write("Blad, dzialanie niemozliwe, liczba kolumn
```

```
                    macierzy A nie jest rowna liczbie wierszy macierzy B");
 96                 return;
 97             }
 98             Console.WriteLine("");
 99             double[,] matrix = new double[arr1.GetLength(1), arr2.GetLength(0)];
100             for (int k = 0; k < matrix.GetLength(1); k++)
101             {
102                 for (int i = 0; i < arr1.GetLength(0); i++)
103                 {
104                     double sum = 0;
105                     for (int j = 0; j < arr1.GetLength(1); j++)
106                     {
107                         sum += arr1[i, j] * arr2[j, k];
108                     }
109                     matrix[i, k] = sum;
110                 }
111             }
112             DisplayMatrix(matrix);
113         }
114         private static void AddMatrix(double[,] arr1, double[,] arr2) //
              Dodawanie macierzy
115         {
116             if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) !=
                arr2.GetLength(1))
117             {
118                 Console.WriteLine("Nie mozna dodac macierzy o roznych
                    wymiarach");
119                 return;
120             }
121             double[,] matrix = new double[arr1.GetLength(0), arr1.GetLength(1)];
122             for (int i = 0; i < arr1.GetLength(0); i++)
123             {
124                 for (int j = 0; j < arr2.GetLength(1); j++)
125                 {
126                     matrix[i, j] = arr1[i, j] + arr2[i, j];
127                 }
128             }
129             DisplayMatrix(matrix);
130         }
131         private static void SubtractMatrix(double[,] arr1, double[,] arr2) //
              Odejmowanie macierzy
132         {
133             if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) !=
                arr2.GetLength(1))
134             {
135                 Console.WriteLine("Nie mozna odejmowac macierzy o roznych
                    wymiarach");
136                 return;
137             }
```

```
138             double[,] matrix = new double[arr1.GetLength(0), arr1.GetLength(1)];
139             for (int i = 0; i < arr1.GetLength(0); i++)
140             {
141                 for (int j = 0; j < arr2.GetLength(1); j++)
142                 {
143                     matrix[i, j] = arr1[i, j] - arr2[i, j];
144                 }
145             }
146             DisplayMatrix(matrix);
147         }
148         private static bool MatrixDeterminant(double[,] matrix) //Wyznacznik
              macierzy
149         {
150             int matrixsize = matrix.GetLength(1);
151             for (int j = 0; j < matrixsize; j++)
152             {
153                 double x = matrix[j, j]; //element listy na przekątnej
154                 if (x == 0) //jeśli element na przękątnej jest równy zero
                      wyznacznik jest równy 0, można więc przerwać obliczenia
155                 {
156                     break;
157                 }
158                 for (int i = j + 1; i < matrixsize; i++) //... dla każdego
                      następnego wiersza
159                 {
160                     double y = matrix[i, j] / x;
161                     for (int k = 0; k < matrixsize; k++)
162                     {
163                         matrix[i, k] -= (matrix[j, k] * y); // y = matrix[i,j] /
                           x
164                     }
165                 }
166                 DisplayMatrix(matrix);
167             }
168             double wyznacznik = 1;
169             for (int a = 0; a < matrixsize; a++)
170             {
171                 wyznacznik *= matrix[a, a];
172             }
173             Console.WriteLine("Wyznacznik macierzy: " + Math.Round(wyznacznik));
174             if (wyznacznik != 0)
175             {
176                 return true;
177             }
178             else
179             {
180                 return false;
181             }
182         }
```

```
183         private static void InvertedMatrix(double[,] matrix) //Macierz odwrotna
184         {
185             double[,] matrixClone = new double[matrix.GetLength(0),
                  matrix.GetLength(1)];
186             matrixClone = (double[,])matrix.Clone();
187             bool isItPossible = MatrixDeterminant(matrixClone);
188             if (isItPossible)
189             {
190                 int matrixsize = matrix.GetLength(1);
191                 double[,] identitymatrix = new double[matrixsize,matrixsize]; //
                    stworzenie macierzy jednostkowej
192                 for (int i = 0; i < matrixsize; i++)
193                 {
194                     identitymatrix[i, i] = 1.0; //wypełnienie jej jedynkami na
                        przekątnej
195                 }
196                 for (int j = 0; j < matrixsize; j++)
197                 {
198                     double x = matrix[j, j];
199                     for (int i = 0; i < matrixsize; i++)
200                     {
201                         if (i == j) continue;
202                         double y = matrix[i, j]/x;
203                         for (int k = 0; k < matrixsize; k++)
204                         {
205                             identitymatrix[i, k] = identitymatrix[i, k] -
                        (identitymatrix[j, k] * y);
206                             matrix[i, k] = matrix[i, k] - (matrix[j, k] * y);
207                         }
208                     }
209                     for (int i = 0; i < matrixsize; i++)
210                     {
211                         identitymatrix[j, i] = (identitymatrix[j, i] / x);
212                         matrix[j, i] = (matrix[j, i] / x);
213                     }
214                     DisplayMatrix(matrix);
215                     DisplayMatrix(identitymatrix);
216                 }
217                 double[,] testmatrix = new double[matrixsize,matrixsize];
218                 for (int i = 0; i < matrixsize; i++)
219                 {
220                     for (int j = 0; j < matrixsize; j++)
221                     {
222                         testmatrix[i, j] = Math.Round(identitymatrix[i, j],3);
223                     }
224                 }
225                 Console.WriteLine("Macierz jednostkowa po zaokrągleniu:");
226                 DisplayMatrix(testmatrix);
227             }
```

```
228         }
229         private static void DisplayMatrix(double[,] arr) //Wyświetl macierz
230         {
231             Console.WriteLine();
232             for (int i = 0; i < arr.GetLength(0); i++)
233             {
234                 for (int j = 0; j < arr.GetLength(1); j++)
235                 {
236                     Console.Write(arr[i, j] + " ");
237                 }
238                 Console.WriteLine("");
239             }
240         }
241         private static void TransposeMatrix(double[,] matrix) //Macierz
              transponowana
242         {
243             double[,] TranMatrix = new double[matrix.GetLength
                (0),matrix.GetLength(1)];
244             for (int i = 0; i < matrix.GetLength(0); i++)
245             {
246                 for (int j = 0; j < matrix.GetLength(1); j++)
247                 {
248                     TranMatrix[j, i] = matrix[i, j];
249                 }
250             }
251             DisplayMatrix(TranMatrix);
252         }
253
254     }
255 }
256
```