```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Security.AccessControl;
 8  using System.Text;
 9  using System.Threading.Tasks;
10  using System.Windows.Forms;
11  using System.Windows.Forms.VisualStyles;
12
13  namespace WindowsFormsApp2
14  {
15      public partial class Form1 : Form
16      {
17          public Form1()
18          {
19              InitializeComponent();
20          }
21          private void GenerateMatrix(string w, string k, Control box, int
              choice) //Stwórz pola typu Textbox dla macierzy
22          {
23              box.Controls.Clear();
24              if (w == "" || k == "")
25              {
26                  label11.Visible = true;
27                  label11.Text = "Błąd, nie wpisano wielkości macierzy";
28                  return;
29              }
30              box.Size = new Size(int.Parse(w)*60, int.Parse(k)*40);
31              for (int i = 0; i < int.Parse(k); i++)
32              {
33                  GroupBox row = new GroupBox();
34                  row.Margin = new Padding(0);
35                  row.Size = new Size(int.Parse(w)*60, 30);
36                  box.Controls.Add(row);
37                  for (int j = 0; j < int.Parse(w); j++)
38                  {
39                      TextBox text = new TextBox();
40                      text.Location = new Point(j*(50+10),0);
41                      text.Size = new Size(50,30);
42                      if (choice == 1)
43                      {
44                          Random rand = new Random();
45                          for (int z = 0; z < 200000; z++)
46                          {
47                              text.Text = rand.Next(1, 10).ToString();
48                          }
```

```
49                      }
50                      else if (choice == 2)
51                      {
52                          if (int.Parse(w) != int.Parse(k))
53                          {
54                              label11.Text = "Macierz nie jest kwadratowa";
55                              label11.Visible = true;
56                              return;
57                          }
58                          if (i==j){
59                              text.Text = "1";
60                          }else{
61                              text.Text = "0";
62                          }
63                      }else{
64                          text.Text = "";
65                      }
66                      row.Controls.Add(text);
67                  }
68              }
69          }
70          private double[,] Read(Control box) //Czytaj wartości z pól
71          {
72              int i = 0;
73              if (box.Controls.Count == 0)
74              {
75                  double[,] tmpmat = new double[0,0];
76                  return tmpmat;
77              }
78              double[,] matrix = new double[box.Controls.Count, box.Controls
                  [0].Controls.Count];
79              foreach (Control control in box.Controls)
80              {
81                  int j = 0;
82                  foreach (TextBox text in control.Controls)
83                  {
84                      if (text.Text == "")
85                      {
86                          double[,] tmpmat = new double[0, 0];
87                          return tmpmat;
88
89                      }
90                      matrix[i, j] = double.Parse(text.Text);
91                      j++;
92                  }
93                  i++;
94              }
95              return matrix;
96          }
```

```
97          private void DisplayMatrix(double[,] matrix) //Wyświetl wartości w
              polach
98          {
99              GenerateMatrix(matrix.GetLength(1).ToString(), matrix.GetLength
                  (0).ToString(), flowLayoutPanel3,0);
100             int i = 0;
101
102             foreach (Control control in flowLayoutPanel3.Controls)
103             {
104                 int j = 0;
105                 foreach (TextBox text in control.Controls)
106                 {
107                     text.Text = matrix[i, j].ToString();
108                     j++;
109                 }
110             i++;
111             }
112         }
113         private void MatrixMultiplication(double[,] matrixA, double[,]
              matrixB) //Mnożenie macierzy
114         {
115             if (matrixA.GetLength(1) != matrixB.GetLength(0)) //Sprawdz czy
                  dzialanie jest mozliwe
116             {
117                 label11.Visible = true;
118                 label11.Text = "Blad, dzialanie niemozliwe, liczba kolumn
                      macierzy A nie jest rowna liczbie wierszy macierzy B";
119                 return;
120             }
121             double[,] matrix = new double[matrixA.GetLength(0),
                  matrixB.GetLength(1)];
122             for (int k = 0; k < matrix.GetLength(1); k++)
123             {
124                 for (int i = 0; i < matrixA.GetLength(0); i++)
125                 {
126                     double sum = 0;
127                     for (int j = 0; j < matrixA.GetLength(1); j++)
128                     {
129                         sum += matrixA[i, j] * matrixB[j,k];
130                     }
131                     matrix[i, k] = sum;
132                 }
133             }
134             DisplayMatrix(matrix);
135         }
136         private void DivideMatrix(double[,] matrixA, double[,] matrixB) //
              Dzielenie macierzy
137         {
138             double[,] InvMatrixB = InvertedMatrix(matrixB);
```

```
139             MatrixMultiplication(matrixA, InvMatrixB);
140         }
141         private void AddMatrix(double[,] arr1, double[,] arr2) //Dodawanie
              macierzy
142         {
143             if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) !=
                  arr2.GetLength(1))
144             {
145                 label11.Visible = true;
146                 label11.Text="Nie mozna dodac macierzy o roznych wymiarach";
147                 return;
148             }
149             for (int i = 0; i < arr1.GetLength(0); i++)
150             {
151                 for (int j = 0; j < arr2.GetLength(1); j++)
152                 {
153                     arr1[i, j] += arr2[i, j];
154                 }
155             }
156             DisplayMatrix(arr1);
157         }
158         private void SubtractMatrix(double[,] arr1, double[,] arr2) //
              Odejmowanie macierzy
159         {
160             if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) !=
                  arr2.GetLength(1))
161             {
162                 label11.Visible = true;
163                 label11.Text="Nie mozna odejmowac macierzy o roznych wymiarach";
164                 return;
165             }
166             for (int i = 0; i < arr1.GetLength(0); i++)
167             {
168                 for (int j = 0; j < arr2.GetLength(1); j++)
169                 {
170                     arr1[i, j] -= arr2[i, j];
171                 }
172             }
173             DisplayMatrix(arr1);
174         }
175         private int MatrixDeterminant(double[,] matrix) //Wyznacznik macierzy
176         {
177             if (matrix.GetLength(0)!=matrix.GetLength(1)) return 0;
178             int matrixsize = matrix.GetLength(1);
179             for (int j = 0; j < matrixsize; j++)
180             {
181                 double x = matrix[j, j]; //element listy na przekątnej
182                 if (x == 0) return 0; //jeśli element na przękątnej jest równy
                      zero wyznacznik jest równy 0, można więc przerwać obliczenia
```

```
183                 for (int i = j + 1; i < matrixsize; i++)
184                 {
185                     double y = matrix[i, j] / x; //liczba przez jaką trzeba
                        pomnożyć wartość z wiersza j do wyzerowania wartości z
                        wierszy
186                     for (int k = 0; k < matrixsize; k++) //... element w wierszu
                         "i" i kolumnie "j"
187                     {
188                         matrix[i, k] = matrix[i, k] - (matrix[j, k] * y); //
                        odejmowanie wartości z wiersza i wartości z wiersza j
                        pomnożonego przez wartość y
189                     }
190                 }
191             }
192             double determinant = 1;
193             for (int a = 0; a < matrixsize; a++)
194             {
195                 determinant *= matrix[a, a];
196             }
197             //return (int)determinant;
198             return (int)Math.Round(determinant);
199         }
200         private double[,] InvertedMatrix(double[,] matrix) //Macierz odwrotna
201         {
202             double[,] matrixClone = new double[matrix.GetLength(0),
                  matrix.GetLength(1)];
203             matrixClone = (double[,])matrix.Clone();
204             int determinant = MatrixDeterminant(matrixClone);
205             if (determinant != 0)
206             {
207                 int matrixsize = matrix.GetLength(1);
208                 double[,] identitymatrix = new double[matrixsize,
                      matrixsize]; //stworzenie macierzy jednostkowej
209                 for (int i = 0; i < matrixsize; i++)
210                 {
211                     identitymatrix[i, i] = 1.0; //wypełnienie jej jedynkami na
                        przekątnej
212                 }
213                 for (int j = 0; j < matrixsize; j++)
214                 {
215                     double x = matrix[j, j];
216                     for (int i = 0; i < matrixsize; i++) //zerowanie kolumn pod
                        przekątną i odjęcie x od reszty wartości w wierszach
217                     {
218                         if (i == j) continue;
219                         double y = matrix[i, j] / x;
220                         for (int k = 0; k < matrixsize; k++)
221                         {
222                             identitymatrix[i, k] = identitymatrix[i, k] -
```

```
                    (identitymatrix[j, k] * y);
223                         matrix[i, k] = matrix[i, k] - (matrix[j, k] * y);
224                     }
225                 }
226                 for (int i = 0; i < matrixsize; i++) //uzyskanie 1 na przekątnej
227                 {
228                     identitymatrix[j, i] = (identitymatrix[j, i] / x);
229                     matrix[j, i] = (matrix[j, i] / x);
230                 }
231             }
232             for (int i = 0; i < matrixsize; i++)
233             {
234                 for (int j = 0; j < matrixsize; j++)
235                 {
236                     identitymatrix[i, j] = Math.Round(identitymatrix[i, j], 3); //zaokrąglanie wyników
237                 }
238             }
239             return identitymatrix;
240         }
241         else
242         {
243             label11.Visible = true;
244             label11.Text = "Macierz nie posiada macierzy odwrotnej";
245             return matrix;
246         }
247     }
248     private void TransposeMatrix(double[,] matrix) //Macierz transponowana
249     {
250         double[,] TranMatrix = new double[matrix.GetLength(1), matrix.GetLength(0)];
251         for (int i = 0; i < matrix.GetLength(0); i++)
252         {
253             for (int j = 0; j < matrix.GetLength(1); j++)
254             {
255                 TranMatrix[j, i] = matrix[i, j];
256             }
257         }
258         DisplayMatrix(TranMatrix);
259     }
260     private void Check(Control text)
261     {
262         if (text.Text != "")
263         {
264             if (int.Parse(text.Text) > 8)
265             {
266                 text.Text = "8";
267             }
```

```
268
269             }
270         }
271         private void button1_Click(object sender, EventArgs e)
272         {
273             if (radioButton1.Checked == true)
274             {
275                 GenerateMatrix(textBox1.Text, textBox2.Text,
                      flowLayoutPanel1,1);
276             }
277             else if(radioButton2.Checked == true)
278             {
279                 GenerateMatrix(textBox1.Text, textBox2.Text,
                      flowLayoutPanel1,2);
280             }
281             else
282             {
283                 GenerateMatrix(textBox1.Text, textBox2.Text,
                      flowLayoutPanel1,0);
284             }
285         }
286         private void textBox1_TextChanged(object sender, EventArgs e)
287         {
288             Check(textBox1);
289         }
290         private void button2_Click(object sender, EventArgs e)
291         {
292             if (radioButton4.Checked == true)
293             {
294                 GenerateMatrix(textBox4.Text, textBox3.Text, flowLayoutPanel2,
                      1);
295             }
296             else if (radioButton3.Checked == true)
297             {
298                 GenerateMatrix(textBox4.Text, textBox3.Text, flowLayoutPanel2,
                      2);
299             }
300             else
301             {
302                 GenerateMatrix(textBox4.Text, textBox3.Text, flowLayoutPanel2,
                      0);
303             }
304         }
305         private void textBox2_TextChanged(object sender, EventArgs e)
306         {
307             Check(textBox2);
308         }
309         private void textBox4_TextChanged(object sender, EventArgs e)
310         {
```

```
311             Check(textBox4);
312         }
313         private void textBox3_TextChanged(object sender, EventArgs e)
314         {
315             Check(textBox3);
316         }
317
318         private void checkBox1_CheckedChanged(object sender, EventArgs e)
319         {
320             if (checkBox1.Checked)
321             {
322                 groupBox4.Enabled = true;
323                 groupBox4.Visible = true;
324                 groupBox1.Enabled = false;
325                 groupBox5.Enabled = true;
326             }
327             else
328             {
329                 groupBox4.Enabled = false;
330                 groupBox4.Visible= false;
331                 groupBox1.Enabled = true;
332                 groupBox5.Enabled = false;
333             }
334         }
335
336         private void button3_Click(object sender, EventArgs e)
337         {
338             label11.Text = "";
339             groupBox8.Visible = false;
340             if (checkBox1.Checked == false)
341             {
342                 double[,] matrix = Read(flowLayoutPanel1);
343                 if (radioButton10.Checked == true)
344                 {
345                     TransposeMatrix(matrix);
346
347                 }else if (radioButton9.Checked == true)
348                 {
349                     double [,] invertedMatrix = InvertedMatrix(matrix);
350                     DisplayMatrix(invertedMatrix);
351                 }else if (radioButton11.Checked == true)
352                 {
353                     int determinant = MatrixDeterminant(matrix);
354                     label14.Text = determinant.ToString();
355                     groupBox8.Visible = true;
356                 }
357                 else
358                 {
359                     label11.Visible = true;
```

```
360                     label11.Text = "Nie wybrano żadnego działania";
361                 }
362             }
363             else
364             {
365                 double[,] matrix1 = Read(flowLayoutPanel1);
366                 double[,] matrix2 = Read(flowLayoutPanel2);
367                 if (radioButton5.Checked == true)
368                 {
369                     AddMatrix(matrix1, matrix2);
370                 }else if (radioButton6.Checked == true)
371                 {
372                     SubtractMatrix(matrix1, matrix2);
373                 }
374                 else if(radioButton8.Checked == true)
375                 {
376                    MatrixMultiplication(matrix1, matrix2);
377                 }
378                 else if (radioButton7.Checked == true)
379                 {
380                     DivideMatrix(matrix2, matrix1);
381                 }
382                 else
383                 {
384                     label11.Visible = true;
385                     label11.Text = "Nie wybrano żadnego działania";
386                 }
387             }
388         }
389     }
390
391 }
392
```