```csharp
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalkulator_macierze
{
    class Program
    {

        static void Main(string[] args)
        {
            double[,] arr1 = BuildMatrix();
            //double[,] arr2 = BuildMatrix();
            //MatrixMultiplication(arr1,arr2);
            InvertedMatrix(arr1);
            Console.ReadLine();
        }

        private static double[,] BuildMatrix() //Buduj macierz o w wierszach i k kolumnach
        {
            Console.Write("Wiersze: ");
            int w = int.Parse(Console.ReadLine());
            Console.Write("Kolumny: ");
            int k = int.Parse(Console.ReadLine());
            double[,] matrix = new double[w, k];
            Console.WriteLine("Jak chcesz uzupełnić macierz");
            Console.WriteLine("1. Automatycznie (Losowe wartości od 1-10)");
            Console.WriteLine("2. Macierz jednostkowa");
            Console.WriteLine("3. Samodzielnie");
            string rodzaj = Console.ReadLine();
            switch (int.Parse(rodzaj))
            {
                case 1: /*losowe*/
                    for (int i = 0; i < w; i++)
                    {
                        for (int j = 0; j < k; j++)
                        {
                            Random rand = new Random();
                            for (int z = 0; z < 2000000; z++)
                            {
                                matrix[i, j] = rand.Next(1, 10);
                            }
                            Console.Write(matrix[i, j] + " ");

                        }
                        Console.WriteLine("");
                    }
                    return matrix;
                case 2: /*jednostkowa*/
                    for (int i = 0; i < w; i++)
                    {
                        matrix[i, i] = 1.0;
                    }
                    return matrix;
                case 3:
                    for (int i = 0; i < matrix.GetLength(0); i++)
                    {
                        Console.WriteLine("Wiersz nr " + (i+1) + ": ");
                        for (int j = 0; j < matrix.GetLength(1); j++)
                        {
                            Console.WriteLine("Wartość nr " + (j+1) + ": ");
                            matrix[i, j] = double.Parse(Console.ReadLine());
                        }
                    }
                    return matrix;
            }

            for (int i = 0; i < w; i++)
            {
                int x = i;
                for (int j = 0; j < k; j++)
                {
                    Random rand = new Random();
                    for (int z = 0; z < 200000; z++)
                    {
                        matrix[i, j] = rand.Next(1, 10);
                    }
                    Console.Write(matrix[i, j] + " ");
                    /*
                    matrix[i, j] = x++ + 1;
                    Console.Write(matrix[i, j] + " ");
                    */
                }
                Console.WriteLine("");
            }
            return matrix;
        }
        private static void MatrixMultiplication(double[,] arr1, double[,] arr2) //Mnożenie macierzy
        {
            if (arr1.GetLength(1) != arr2.GetLength(0)) //Sprawdz czy dzialanie jest mozliwe
            {
                Console.Write("Blad, dzialanie niemozliwe, liczba kolumn
                  macierzy A nie jest rowna liczbie wierszy macierzy B");
                return;
            }
            Console.WriteLine("");
            double[,] matrix = new double[arr1.GetLength(1), arr2.GetLength(0)];
            for (int k = 0; k < matrix.GetLength(1); k++)
            {
                for (int i = 0; i < arr1.GetLength(0); i++)
                {
                    double sum = 0;
                    for (int j = 0; j < arr1.GetLength(1); j++)
                    {
                        sum += arr1[i, j] * arr2[j, k];
                    }
                    matrix[i, k] = sum;
                }
            }
            DisplayMatrix(matrix);
        }
        private static void AddMatrix(double[,] arr1, double[,] arr2) // Dodawanie macierzy
        {
            if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) != arr2.GetLength(1))
            {
                Console.WriteLine("Nie mozna dodac macierzy o roznych wymiarach");
                return;
            }
            double[,] matrix = new double[arr1.GetLength(0), arr1.GetLength(1)];
            for (int i = 0; i < arr1.GetLength(0); i++)
            {
                for (int j = 0; j < arr2.GetLength(1); j++)
                {
                    matrix[i, j] = arr1[i, j] + arr2[i, j];
                }
            }
            DisplayMatrix(matrix);
        }
        private static void SubtractMatrix(double[,] arr1, double[,] arr2) // Odejmowanie macierzy
        {
            if (arr1.GetLength(0) != arr2.GetLength(0) || arr1.GetLength(1) != arr2.GetLength(1))
            {
                Console.WriteLine("Nie mozna odejmowac macierzy o roznych wymiarach");
                return;
            }
            double[,] matrix = new double[arr1.GetLength(0), arr1.GetLength(1)];
            for (int i = 0; i < arr1.GetLength(0); i++)
            {
                for (int j = 0; j < arr2.GetLength(1); j++)
                {
                    matrix[i, j] = arr1[i, j] - arr2[i, j];
                }
            }
            DisplayMatrix(matrix);
        }
        private static bool MatrixDeterminant(double[,] matrix) //Wyznacznik macierzy
        {
            int matrixsize = matrix.GetLength(1);
            for (int j = 0; j < matrixsize; j++)
            {
                double x = matrix[j, j]; //element listy na przekątnej
                if (x == 0) //jeśli element na przękątnej jest równy zero wyznacznik jest równy 0, można więc przerwać obliczenia
                {
                    break;
                }
                for (int i = j + 1; i < matrixsize; i++) //... dla każdego następnego wiersza
                {
                    double y = matrix[i, j] / x;
                    for (int k = 0; k < matrixsize; k++)
                    {
                        matrix[i, k] -= (matrix[j, k] * y); // y = matrix[i,j] / x
                    }
                }
                DisplayMatrix(matrix);
            }
            double wyznacznik = 1;
            for (int a = 0; a < matrixsize; a++)
            {
                wyznacznik *= matrix[a, a];
            }
            Console.WriteLine("Wyznacznik macierzy: " + Math.Round(wyznacznik));
            if (wyznacznik != 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        private static void InvertedMatrix(double[,] matrix) //Macierz odwrotna
        {
            double[,] matrixClone = new double[matrix.GetLength(0),
              matrix.GetLength(1)];
            matrixClone = (double[,])matrix.Clone();
            bool isItPossible = MatrixDeterminant(matrixClone);
            if (isItPossible)
            {
                int matrixsize = matrix.GetLength(1);
                double[,] identitymatrix = new double[matrixsize,matrixsize]; //
                  stworzenie macierzy jednostkowej
                for (int i = 0; i < matrixsize; i++)
                {
                    identitymatrix[i, i] = 1.0; //wypełnienie jej jedynkami na
                    przekątnej
                }
                for (int j = 0; j < matrixsize; j++)
                {
                    double x = matrix[j, j];
                    for (int i = 0; i < matrixsize; i++)
                    {
                        if (i == j) continue;
                        double y = matrix[i, j]/x;
                        for (int k = 0; k < matrixsize; k++)
                        {
                            identitymatrix[i, k] = identitymatrix[i, k] -
                    (identitymatrix[j, k] * y);
                            matrix[i, k] = matrix[i, k] - (matrix[j, k] * y);
                        }
                    }
                    for (int i = 0; i < matrixsize; i++)
                    {
                        identitymatrix[j, i] = (identitymatrix[j, i] / x);
                        matrix[j, i] = (matrix[j, i] / x);
                    }
                    DisplayMatrix(matrix);
                    DisplayMatrix(identitymatrix);
                }
                double[,] testmatrix = new double[matrixsize,matrixsize];
                for (int i = 0; i < matrixsize; i++)
                {
                    for (int j = 0; j < matrixsize; j++)
                    {
                        testmatrix[i, j] = Math.Round(identitymatrix[i, j],3);
                    }
                }
                Console.WriteLine("Macierz jednostkowa po zaokrągleniu:");
                DisplayMatrix(testmatrix);
            }
        }
        private static void DisplayMatrix(double[,] arr) //Wyświetl macierz
        {
            Console.WriteLine();
            for (int i = 0; i < arr.GetLength(0); i++)
            {
                for (int j = 0; j < arr.GetLength(1); j++)
                {
                    Console.Write(arr[i, j] + " ");
                }
                Console.WriteLine("");
            }
        }
        private static void TransposeMatrix(double[,] matrix) //Macierz transponowana
        {
            double[,] TranMatrix = new double[matrix.GetLength(0),matrix.GetLength(1)];
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    TranMatrix[j, i] = matrix[i, j];
                }
            }
            DisplayMatrix(TranMatrix);
        }

    }
}
```