Arbeitsblatt 8. 2D-Felder

Aufgabe 8.1. 2D-Feld erzeugen, ausgeben, verarbeiten

a) Erzeugen Sie ein zweidimensionales Feld f mit beliebiger, durch Sie festgelegter Größe, z.B. 3 Zeilen, 5 Spalten

Befüllen Sie das Feld mit Zufallszahlen zwischen 0 und 9.

b) Schreiben Sie eine Funktion Ausgabe, die ein zweidimensionales Integer-Feld auf der Console ausgibt. Den Code dafür können Sie i.W. aus dem Handout zu mehrdimensionalen Arrays oder der Vorlesung übernehmen.

Der Funktionskopf sollte z.B. so aussehen:

```
    static void Ausgabe(int[,] f)
```

Lassen Sie damit das Array aus der ersten Teilaufgabe ausgeben.

c) Schreiben Sie eine Funktion "Transponiere", welche die Zeilen und Spalten eines beliebig großen zweidimensionalen Feldes vertauscht (transponiert), d.h. folgende Transformation soll vorgenommen werden:

$$\begin{bmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 21 & 31 \\ 12 & 22 & 32 \end{bmatrix}$$

Die Funktion soll ein zweidimensionales Array als Parameter bekommen und ein neues zweidimensionales Array erzeugen und als Ergebnis zurückgeben. Der Funktionskopf sollte also so aussehen:

```
    static int[,] Transponiere(int[,] f)
```

Testen Sie die Funktion mit dem Feld aus der ersten Aufgabe.

```
    int[,] f2 = Transponiere(f);
    Console.WriteLine("Transponiert:");
    Ausgabe(f2);
```

d) Ausgabe mit Spaltensumme

Stellen Sie sich vor, Ihr Feld würde die Zahlen in einem Excel-Arbeitsblatt Verkäufe oder Lagerbestände beinhalten. In Excel muss häufig eine Zeilen- oder Spaltensumme errechnet werden. Das machen wir jetzt auch.

Erweitern Sie dafür Ihre Ausgabefunktion von oben: Geben Sie in einer Summenzeile unterhalb des 2D-Feldes die jeweiligen Spaltensummen aus.

```
  3  5  6
  7  8  2
  4  5  3
  9  7  8
----------
 23 25 19
```

Damit die Spaltensumme nicht immer ausgegeben wird, ergänzen Sie in der Ausgabefunktion einen optionalen Bool-Parameter (default false).

Nur, wenn dieser explizit mit true angegeben wird, soll die Summe ausgegeben werden.

Aufgabe 8.2. Einheitsmatrix

Eine Einheits- oder Identitätsmatrix ist eine quadratische Matrix, die auf der Diagonale nur Einsen hat und ansonsten Nullen, z.B. so $\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$.

Schreiben Sie eine Funktion "IsIdentity()", die für eine beliebige Matrix ermittelt, ob sie eine Einheitsmatrix ist und testen Sie sie in einem Hauptprogramm mit geeigneten Testfällen, z.B.

```
int[,] m1 = { { 1, 0, 0 },
              { 0, 1, 0 },
              { 0, 0, 1 } };

int[,] m2 = { { 1, 0, 0 },
              { 0, 1, 0 },
              { 0, 1, 1 } };

int[,] m3 = { { 1, 0, 0 },
              { 0, 1, 0 },
              { 0, 0, 0 } };
```

Aufgabe 8.3. Werte-Matrix

Schreiben Sie ein Programm, das einen Wert n einliest und dann eine quadratische Werte-Matrix wie unten angegeben erzeugt und ausgibt. Nutzen Sie für die Ausgabe die Funktion aus der ersten Aufgabe.

```
n eingeben: 5
   0   1   2   3   4
  -1   0   1   2   3
  -2  -1   0   1   2
  -3  -2  -1   0   1
  -4  -3  -2  -1   0
```

Überlegen Sie sich, wie die Berechnungsvorschrift aussehen muss, um aus i und j den Zellenwert zu ermitteln.

Aufgabe 8.4. Für die Schnellen: Größtes Produkt in einer Matrix

Lösen Sie Problem 11 von Project Euler: https://projecteuler.net/problem=11

Die Lösung ist 70600674. Hier bzw. im Git unter "Euler11.cs" die Matrix zum Rauskopieren:

```
int[,] m = {{08, 02, 22, 97, 38, 15, 00, 40, 00, 75, 04, 05, 07, 78, 52, 12, 50, 77, 91, 08},
            {49, 49, 99, 40, 17, 81, 18, 57, 60, 87, 17, 40, 98, 43, 69, 48, 04, 56, 62, 00},
            {81, 49, 31, 73, 55, 79, 14, 29, 93, 71, 40, 67, 53, 88, 30, 03, 49, 13, 36, 65},
            {52, 70, 95, 23, 04, 60, 11, 42, 69, 24, 68, 56, 01, 32, 56, 71, 37, 02, 36, 91},
            {22, 31, 16, 71, 51, 67, 63, 89, 41, 92, 36, 54, 22, 40, 40, 28, 66, 33, 13, 80},
            {24, 47, 32, 60, 99, 03, 45, 02, 44, 75, 33, 53, 78, 36, 84, 20, 35, 17, 12, 50},
            {32, 98, 81, 28, 64, 23, 67, 10, 26, 38, 40, 67, 59, 54, 70, 66, 18, 38, 64, 70},
            {67, 26, 20, 68, 02, 62, 12, 20, 95, 63, 94, 39, 63, 08, 40, 91, 66, 49, 94, 21},
            {24, 55, 58, 05, 66, 73, 99, 26, 97, 17, 78, 78, 96, 83, 14, 88, 34, 89, 63, 72},
            {21, 36, 23, 09, 75, 00, 76, 44, 20, 45, 35, 14, 00, 61, 33, 97, 34, 31, 33, 95},
            {78, 17, 53, 28, 22, 75, 31, 67, 15, 94, 03, 80, 04, 62, 16, 14, 09, 53, 56, 92},
            {16, 39, 05, 42, 96, 35, 31, 47, 55, 58, 88, 24, 00, 17, 54, 24, 36, 29, 85, 57},
            {86, 56, 00, 48, 35, 71, 89, 07, 05, 44, 44, 37, 44, 60, 21, 58, 51, 54, 17, 58},
            {19, 80, 81, 68, 05, 94, 47, 69, 28, 73, 92, 13, 86, 52, 17, 77, 04, 89, 55, 40},
            {04, 52, 08, 83, 97, 35, 99, 16, 07, 97, 57, 32, 16, 26, 26, 79, 33, 27, 98, 66},
            {88, 36, 68, 87, 57, 62, 20, 72, 03, 46, 33, 67, 46, 55, 12, 32, 63, 93, 53, 69},
            {04, 42, 16, 73, 38, 25, 39, 11, 24, 94, 72, 18, 08, 46, 29, 32, 40, 62, 76, 36},
            {20, 69, 36, 41, 72, 30, 23, 88, 34, 62, 99, 69, 82, 67, 59, 85, 74, 04, 36, 16},
            {20, 73, 35, 29, 78, 31, 90, 01, 74, 31, 49, 71, 48, 86, 81, 16, 23, 57, 05, 54},
            {01, 70, 54, 71, 83, 51, 54, 69, 16, 92, 33, 48, 61, 43, 52, 01, 89, 19, 67, 48}};
```