Arbeitsblatt 7. Felder

Aufgabe 7.1. Feldübung

Schreiben Sie ohne Zuhilfenahme vordefinierter Standardfunktionen ein Programm, das

a) ein Feld f mit den Zahlen 5, 8, 6, 4, 3, 5, 9, 2, 6, 9 anlegt und den Inhalt des Feldes in einer Zeile ausgibt:

```
5 8 6 4 3 5 9 2 6 9
// können Sie alternativ auch diese Ausgabe erzeugen?
f = [5, 8, 6, 4, 3, 5, 9, 2, 6, 9]
```

b) die Summe und Durchschnitt der Zahlen ermittelt und ausgibt:

```
Summe: 57
Durchschnitt: 5,7
```

c) eine weitere Zahl einliest und alle Zahlen des Feldes f ausgibt, die größer als diese Zahl sind:

```
Zahl eingeben: 5

Größer sind: 8 6 9 6 9
```

d) die größte Zahl und kleinste Zahl im Feld f sucht und ausgibt:

```
Maximum: 9
Minimum: 2
```

e) Erstellen Sie für Aufgabe a) eine Funktion Ausgabe, die ein Integer-Feld als Parameter bekommt und den Feldinhalt auf der Console ausgibt. Passen Sie Ihr Hauptprogramm an.

f) Schreiben Sie eine Funktion FeldStatistik, die ein int-Feld als Eingabeparameter bekommt und Summe, Durchschnitt, Min und Max als Out-Parameter zurückgibt. Passen Sie das Hauptprogramm so an, dass diese Funktion statt b) und c) verwendet wird.

g) Optional: Unter Einbindung von System.Linq und über die Array-Klasse sind viele Funktionen für iterierbare Datentypen (Felder, Listen usw.) standardmäßig verfügbar. Probieren Sie sie aus bzw. googlen Sie deren Funktionalität: `f.Sum(), f.Min(), f.Max()`

Aufgabe 7.2. Feldwerte umkehren (Reverse)

a) Schreiben Sie eine Funktion Reverse(f), die ein Integer-Feld f als Eingabe-Parameter bekommt und die Reihenfolge der Elemente im Feld f umkehrt.
Es wird in der Funktion kein neues Feld erzeugt, es werden nur Elemente vertauscht!

Beispiel:

```
int[] f = { 6, 7, 4, 3, 2, 5, 9 };

Console.Write("Feld vorher: ");
Ausgabe(f);
//Aufruf für Reverse – soll genau so funktionieren!
Reverse(f);
Console.Write("Feld umgekehrt: ");
Ausgabe(f);
```

Liefert als Ausgabe:

```
Feld vorher: 6 7 4 3 2 5 9
Feld umgekehrt: 9 5 2 3 4 7 6
```

Wie gehen Sie vor? Brauchen Sie mehrere Schleifen oder reicht eine?

b) Die Funktion Reverse() bildet das Verhalten der Standard-Funktion Array.Reverse(f) nach. Verwenden Sie diese Funktion, um Ihr Feld wieder in den Ausgangszustand zu versetzen und lassen Sie es erneut ausgeben.

Aufgabe 7.3. Balken-Diagramm

Erstellen Sie eine Funktion, welche zu einem gegebenen eindimensionalen Integer-Feld beliebiger Länge ein Histogramm als Balken-Diagramm wie unten dargestellt ausgibt.

Sie können für die Balken einfach ein * als Zeichen nehmen oder bspw. das Unicode-Zeichen '\u2584', wie in dem Beispiel unten.

Beispiel:

```
int[] f = { 10, 20, 15, 2, 6 };
BalkenDiagramm(f);

10 ▄▄▄▄▄▄▄▄▄▄
20 ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
15 ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
 2 ▄▄
 6 ▄▄▄▄▄▄
```

Hinweis: Weitere Sonderzeichen finden Sie in der Unicode-Zeichentabelle, z.B. hier: https://www.key-shortcut.com/zeichentabellen/unicode-2000-2fff/#c1609.

Ggf. ist dieser Befehl vorab notwendig, um das Zeichen richtig darzustellen:

```
    Console.OutputEncoding = System.Text.Encoding.UTF8;
```

Aufgabe 7.4. Selection Sort

Ein einfacher Algorithmus für die Sortierung der Werte in einem Array ist Selection Sort. Die Grundidee ist folgende:

- Finde die kleinste Zahl und tausche sie mit der Zahl in der ersten Position
- Finde die nächstkleinste Zahl und tausche sie mit der Zahl in der zweiten Position
- usw. bis die letzte Position erreicht ist

Schreiben Sie eine Funktion **Sort**, die ein Integer-Feld als Parameter erhält und dieses Feld mit diesem Algorithmus sortiert (ohne ein neues anzulegen).

Überlegen Sie sich zuerst auf Papier, was dafür zu tun ist. Spielen Sie den Lösungsweg an einem einfachen Beispiel durch, z.B. diesem hier:

```
int[] a = {5, 8, 3, 2, 9, 8};
```

Aufgabe 7.5. Quersumme rekursiv

Die Quersumme einer positiven ganzen Zahl berechnet sich durch die Summe ihrer Ziffern, d.h.

Quersumme(3521) = 3+5+2+1

Schreiben Sie eine rekursive Funktion für die Berechnung der Quersumme:

$$q(n) = \begin{cases} n & falls\ n = 0 \\ n\%10 + q(n\ /\ 10) & sonst \end{cases}$$