Aufgaben Arrays:

Aufgabe 1: Array füllen

Schreiben Sie ein Programm, das ein Array mit 10 zufälligen Integer-Werten zwischen 1 und 100 füllt. Geben Sie anschließend die 10 Werte auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
37, 2, 93, 45, 81, 89, 12, 19, 66, 20
```

Aufgabe 2: Quadratzahlen

Schreiben Sie ein Programm, das die Quadratzahlen von 1 bis 10 in einem Array speichert. Geben Sie die Zahlen anschließend in umgekehrter Reihenfolge wieder auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
100, 81, 64, 49, 36, 25, 16, 9, 4, 1
```

Aufgabe 3: Vor- und Nachname

Schreiben Sie ein Programm, das zwei String-Arrays gleicher Größe erstellt, die Größe soll der Benutzer festlegen. Der Benutzer soll anschließend in einer Schleife Vor- und Nachnamen von Personen eingeben, wobei der Vorname im ersten Array und der Nachname im zweiten Array gespeichert wird. Es dürfen nur genau so viele Namen eingegeben werden, wie in die Arrays passen. Geben Sie anschließend die vollständigen Namen in einer Schleife auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
Anzahl Personen: 3
Eingabe Vorname: Max
Eingabe Nachname: Mustermann
Eingabe Vorname: Paul
Eingabe Nachname: Panzer
Eingabe Vorname: James
Eingabe Nachname: Bond

Max Mustermann
Paul Panzer
James Bond
```

Aufgabe 4: Statistik

Schreiben Sie ein Programm, das ein Array mit 10 zufälligen Zahlen zwischen 1 und 99 füllt. Geben Sie danach auf der Konsole die größte Zahl, die kleinste Zahl, den Durchschnitt der Zahlen und die Summe der Zahlen aus.

Beispiel-Bildschirmausgabe:

```
37, 2, 93, 45, 81, 89, 12, 19, 66, 20
Minimum: 2
Maximum: 93
Durchschnitt: 46,4
Summe: 464
```

Aufgabe 5: Lottozahlen 1

Schreiben Sie ein Programm, das 6 zufällige Zahlen zwischen 1 und 49 wählt. Speichern Sie diese aber nicht in einem Array der Größe 6, sondern erstellen Sie einen Boolean-Array der Größe 49 (oder besser 50?). Markieren Sie dort die gezogenen Zahlen und sorgen Sie dafür, dass jede Zahl nur einmal gezogen wird. Geben Sie anschließend die Zahlen auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
6, 16, 17, 26, 30, 33
```

Aufgabe 6: Binärzahlen 1

Schreiben Sie ein Programm, das eine maximal 8 Bit große Dezimalzahl in eine Binärzahl umrechnet. Legen Sie dazu ein Integer-Array der Größe 8 an und wenden Sie dann das Divisionsverfahren an, wobei Sie jeweils den Rest im Array speichern. Geben Sie anschließend die Binärzahl korrekt auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
Eingabe Dezimalzahl: 168
Ergebnis Binärzahl: 10101000
```

Aufgabe 7: Array-Suche 1

Schreiben Sie ein Programm, das ein Array mit 50 zufälligen Zahlen im Bereich von 1 bis 99 füllt. Fragen Sie anschließend den Benutzer nach einer Zahl aus diesem Bereich, und prüfen Sie, ob die Zahl im Array vorhanden ist. Geben Sie die erste Position der Zahl im Array aus. Zur Kontrolle können Sie auch zusätzlich das gesamte Array auf der Konsole ausgeben.

Beispiel-Bildschirmausgabe:

```
Eingabe Zahl: 52
Gefunden an Position 11
```

```
Eingabe Zahl: 99
Nicht gefunden
```

Aufgabe 8: Spielpaarungen

Schreiben Sie ein Programm, das alle einmaligen Spielpaarungen von 10 Mannschaften ausgibt. Die Mannschaften sollen die Bezeichnung A bis J bekommen. Speichern Sie die Mannschaften in einem Array und geben Sie die Paarungen über zwei verschachtelte Schleifen aus. Einmalige Spielpaarung bedeutet, dass „A gegen B“ identisch ist mit „B gegen A“ und daher nur einmal auftauchen darf. Und die Mannschaften spielen natürlich auch nicht gegen sich selbst.

Beispiel-Bildschirmausgabe:

```
A gegen B
A gegen C
A gegen D
…
B gegen C
…
I gegen J
```

Aufgabe 9: Array-Suche 2

Verbessern Sie das Programm Array-Suche 1, sodass nun alle Positionen ausgegeben werden, an denen die Zahl im Array vorkommt.

Beispiel-Bildschirmausgabe:

```
Eingabe Zahl: 52
Gefunden an Position 3, 17, 49
```

Aufgabe 10: Lottozahlen 2

Schreiben Sie ein Programm, das 6 zufällige Zahlen zwischen 1 und 49 wählt, und diese in einem Integer-Array der Größe 6 speichert. Stellen Sie sicher, dass jede Zahl in der Ziehung nur einmal gezogen werden kann. Geben Sie anschließend die Zahlen auf der Konsole aus.

Beispiel-Bildschirmausgabe:

```
26, 33, 17, 6, 30, 16
```

Aufgabe 11: Mittelwert ohne Min und Max

Schreiben Sie ein Programm, das ein Integer-Array mit zufälligen Zahlen zwischen 1 und 99 füllt. Die Größe des Arrays soll der Benutzer festlegen. Ermitteln Sie dann den Mittelwert der Zahlen, aber schließen Sie dabei die größte und die kleinste Zahl aus.

Beispiel-Bildschirmausgabe:

```
Anzahl Zahlen: 10

37, 2, 93, 45, 81, 89, 12, 19, 66, 20

Mittelwert ohne Min und Max: 46,125
```

Aufgabe 12: Summensuche

Schreiben Sie ein Programm, das ein Integer-Array der Länge 10 mit zufälligen Zahlen im Bereich von 1 bis 99 füllt. Das Programm soll dann prüfen, ob zwei Zahlen im Array zusammen eine bestimmte Summe ergeben. Die Summe soll vom Benutzer eingegeben werden.

Beispiel-Bildschirmausgabe:

```
37, 2, 93, 45, 81, 89, 12, 19, 66, 20

Eingabe Summe: 57

Index 0 und 9 ergeben zusammen 57.

Index 3 und 6 ergeben zusammen 57.
```

Aufgabe 13: Reverse

Schreiben Sie ein Programm, das eine eingegebene Zeichenkette umdreht. Dabei sollen in einer Schleife die sich gegenüber liegenden Zeichen im Array vertauscht werden. Die Verwendung von speziellen Array-Methoden wie z.B. Reverse() ist natürlich nicht Sinn der Sache.

Beispiel-Bildschirmausgabe:

```
Eingabe: Lager
Ausgabe: regaL
```

Aufgabe 14: Palindrom

Schreiben Sie ein Programm, das eine eingegebene Zeichenkette daraufhin prüft, ob es sich um ein Palindrom handelt. Ein Palindrom ist ein Text, der von rechts und von links gelesen das Gleiche ergibt. Zum Beispiel „Regallager“ oder „MAOAM“ oder „0123210“. Die Groß- und Kleinschreibung soll dabei nicht beachtet werden.

Beispiel-Bildschirmausgabe:

```
Eingabe: Regallager
Ist ein Palindrom.
Eingabe: Regal
Ist kein Palindrom.
```

Aufgabe 15: Rotationen

Schreiben Sie ein Programm, das zu einer eingegebenen Zeichenkette alle Rotationen dieser Zeichenkette auf der Konsole ausgibt.

Beispiel-Bildschirmausgabe:

```
Eingabe: Regal
egalR
galRe
alReg
lRega
Regal
```

Aufgabe 16: Duplikatsuche

Schreiben Sie ein Programm, das nach Duplikaten in einem Integer-Array sucht. Füllen Sie ein Integer-Array mit zufälligen Zahlen (oder geben Sie ein fest definiertes Array vor). Suchen Sie dann im Array nach doppelten Zahlen und geben Sie diese auf der Konsole aus:

Beispiel-Bildschirmausgabe:

```
37, 2, 93, 45, 81, 89, 12, 19, 81, 66, 20, 2

Duplikat gefunden: 2

Duplikat gefunden: 81
```

Aufgabe 17: Luhn-Algorithmus

Schreiben Sie ein Programm, das vom Benutzer die Eingabe einer Kreditkartennummer verlangt. Eine Kreditkartennummer ist 12- bis 16-stellig, wobei die letzte Ziffer eine Prüfziffer nach dem Luhn-Algorithmus ist.

Der Luhn-Algorithmus ist folgendermaßen definiert:

1. `Durchlaufe die Nummer (mit der Prüfziffer) ziffernweise von rechts nach links und bilde die Summe der Ziffern. Aber: Verdoppele dabei jede zweite Ziffer, und wenn dabei ein Wert größer als 9 heraus-kommt, subtrahiere 9.`
2. `Wenn die Summe als letzte Ziffer eine 0 hat (Modulo 10 = 0), erken-ne die Nummer als gültig an und sonst nicht.`

Beispiel:

Ziffer:	4	6	8	3	4	5	7	8	2	9	3	7	6	5	2	2	
Doppelt:	8		16		8		14		4		6		12		4		
Einstellig:			7				5						3				
Addieren:	8	6	7	3	8	5	5	8	4	9	6	7	3	5	4	2	= 90

Beispiel-Bildschirmausgabe:

```
Kreditkartennummer: 4683457829376522
Die Karte ist gültig.

Kreditkartennummer: 4683457829376525
Die Karte ist nicht gültig.
```

Aufgabe 18: Binärsuche

Schreiben Sie ein Programm, das eine Zahl in einem vorgegebenen, sortierten Array sucht. Das Array soll dabei wie folgt definiert sein: `1, 3, 4, 7, 9, 13, 17, 18, 19, 23, 26, 30`

Gehen Sie bei der Suche nach folgendem Algorithmus vor:

- Bilden Sie ein Intervall über das Array. Die untere Grenze ist zu Beginn 0, die obere Grenze ist zu Beginn das letzte Element des Arrays.
- Schauen Sie sich das Element an, das in der Mitte des Intervalls liegt ((UG + OG) / 2). Vergleichen Sie das Element an dieser Position mit dem Suchelement.
  - Haben Sie das Element gefunden, ist der Algorithmus beendet.
  - Ist das Suchelement kleiner, als das Element dieser Position, suchen Sie in der unteren Hälfte weiter. Die untere Grenze bleibt gleich, obere Grenze ist die aktuelle Position -1.
  - Ist das Suchelement größer, als das Element dieser Position, suchen Sie in der oberen Hälfte weiter. Die obere Grenze bleibt gleich, untere Grenze ist die aktuelle Position +1.

Wiederholen Sie die obigen Schritte so lange, bis Sie entweder das Element gefunden haben, oder aber die untere Grenze größer oder gleich der oberen Grenze ist.