# Versuchen Sie immer, Ihren Code zu kommentieren!

## Aufgabe 1

Erstelle ein Array vom Typ int mit 5 Elementen und initialisiere es mit Werten deiner Wahl (Statische Initialisierung). Gib dann alle Elemente des Arrays auf der Konsole aus.

## Aufgabe 2

Erstelle ein Array vom Typ float mit 5 Elementen (Dynamische Initialisierung) und fülle es innerhalb einer Schleife mit dem Wert des Index. Gib dann alle Elemente des Arrays auf der Konsole aus.

## Aufgabe 3

Erstelle ein Array vom Typ int mit 10 Elementen und fülle es mithilfe von 2 Schleifen. Die erste Schleife soll alle Elemente mit geraden Indizes belegen, und die zweite Schleife soll alle Elemente mit ungeraden Indizes belegen. Gib dann alle Elemente des Arrays in einer dritten Schleife auf der Konsole aus.

## Aufgabe 4

Schreibe ein Programm, das die Länge eines Arrays dynamisch vom Nutzer abfragt und dann ein Array dieser Länge erstellt. Fülle es mit zufälligen Zahlen von 1 bis 100 (System.Random) und gib die Summe der Elemente aus.

## Aufgabe 5

Erstelle ein Array mit Zahlen von 0 bis 5 und ersetze jede Zahl durch die Summe aus sich selbst und 2 mithilfe einer Schleife (value + value + 2). Gib alle Elemente auf der Konsole aus.

## Aufgabe 6

Erstelle ein Array von booleschen Werten und negiere jeden Wert (setze true auf false und umgekehrt) mithilfe einer Schleife. Gib alle Elemente vor und nach der Schleife aus.

## Aufgabe 7

Erstelle ein Array von mindestens 10 Zahlen und berechne die Summe, den Durchschnitt, die Varianz und die Standardabweichung der Werte mithilfe einer Schleife. Gib die Ergebnisse aus.

## Aufgabe 8

Erstelle ein Array von Zahlen und finde das Maximum und Minimum mithilfe einer Schleife. Gib die Werte anschließend auf der Konsole aus.

## Aufgabe 9

Erstelle ein Array von Zahlen und gib diese in umgekehrter Reihenfolge auf der Konsole aus. Beispiel: 123456 => 654321

## Aufgabe 10

Erstelle ein Array von Zahlen und drehe die Reihenfolge der Elemente mithilfe einer Schleife um. Setze den ersten Wert an die letzte Stelle und den letzten Wert an die erste Stelle usw. Baue deine Schleife so, dass du kein weiteres Array benötigst.

## Aufgabe 11

Erstelle zwei Arrays von Zahlen gleicher Länge. Addiere die entsprechenden Elemente beider Arrays und speichere das Ergebnis in einem neuen Array. Gib alle Arrays und das Ergebnis aus.

Beispiel:
Array 1: 0 1 2 3
Array 2: 1 2 3 4
Array 3: 1 3 5 7

## Aufgabe 12

Erstelle ein Array von Zahlen. Ersetze alle geraden Zahlen durch ihre Hälfte und alle ungeraden Zahlen durch das Doppelte mithilfe einer Schleife. Gib das veränderte Array aus.

## Aufgabe 13

Erstelle ein Array von mindestens 10 Zahlen und prüfe, ob es aufsteigend sortiert ist. Gib eine entsprechende Meldung aus.

Beispiel: 1 2 3 4 5 6 7 8 9 10

## Aufgabe 14

Erstelle ein Array von mindestens 10 Zahlen und prüfe, ob es absteigend sortiert ist. Gib eine entsprechende Meldung aus.

Beispiel: 10 9 8 7 6 5 4 3 2 1

## Aufgabe 15

Erstelle ein Array von mindestens 10 Zahlen und prüfe, ob es auf- oder absteigend sortiert ist. Gib eine entsprechende Meldung aus.

## Aufgabe 16

Erstelle ein Array von ganzen Zahlen und implementiere einen Algorithmus, der das Array um eine Position nach rechts rotiert. Der letzte Wert wird zum ersten, und die übrigen Werte verschieben sich entsprechend. Gib das veränderte Array aus.

## Aufgabe 17

Schreibe eine Konsolenanwendung, die das Array args[] auswertet, um verschiedene Befehle und Parameter zu verarbeiten. Folgende Anforderungen sollen erfüllt werden:

1. –h: Zeigt eine Hilfe-Übersicht mit gültigen Farben für den Vordergrund und Hintergrund an:

   ----- Hilfe -----
   Farben die erlaubt sind:
   Red
   Blue
   Yellow

2. –foreground_color <farbe>: Legt die Farbe des Textes in der Konsole fest. Wenn die Farbe ungültig ist, wird sie ignoriert und eine Fehlermeldung ausgegeben. Die Textfarbe kann mit *Console.ForegroundColor = ConsoleColor.Red;* geändert werden
3. --background_color <farbe>: Legt die Farbe des Hintergrundes der Konsole fest. Wenn die Farbe ungültig ist, wird sie ignoriert und eine Fehlermeldung ausgegeben. Die Textfarbe kann mit *Console.BackgroundColor = ConsoleColor.Red;* geändert werden
4. Die Anwendung soll mit Parametern aufgerufen werden.

   Beispiel: console.exe –foreground_color Red –background_color Blue

   Zum Debuggen können die Parameter über die Debugeigenschaften eingestellt werden

## Aufgabe 18

Schreibe ein Programm, das eine 3x3 Matrix (zweidimensionales Array) erstellt und initialisiert. Gib die Matrix anschließend in der Konsole aus.

Beispiel:
3 2 3
1 5 2
3 5 7

## Aufgabe 19

Schreibe ein Programm, das die Summe aller Elemente in einer 5x5 Matrix berechnet. Gib die Summe anschließend auf der Konsole aus.

Beispiel:
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7

Summe: 60

## Aufgabe 20

Schreibe ein Programm, das die transponierte Form einer 2x3 Matrix auf der Konsole ausgibt. Die transponierte Matrix vertauscht die Zeilen und Spalten.

Beispiel:
1 2 3
2 4 6

1 2
2 4
3 6

## Aufgabe 21

Schreibe ein Programm, das die Diagonalelemente einer quadratischen 3x3 Matrix ausgibt.
Beispiel:

1 2 1
3 4 5
7 4 9

Ausgabe: 1 4 9

## Aufgabe 22

Schreibe ein Programm, das eine 3x3x3 Matrix mit den folgenden Werten anlegt und anschließend auf der Konsole ausgibt.

1 2 3  2 3 4  3 4 5
2 3 4  3 4 5  4 5 6
3 4 5  4 5 6  5 6 7

## Aufgabe 23

Schreibe ein Programm, das ein Jagged Array der Länge 3 erstellt. Für jedes verschachtelte Array soll die Länge vom Benutzer eingegeben werden. Die verschachtelten Arrays sollen mit vom Benutzer eingegebenen Zahlen gefüllt und dann korrekt formatiert ausgegeben werden.

Beispiel:

Jagged Array Länge 3
Array 1 Länge 3
Array 2 Länge 2
Array 3 Länge 4

Ausgabe:
1 2 3
1 3
1 2 3 4

## Aufgabe 22 (Selection Sort)

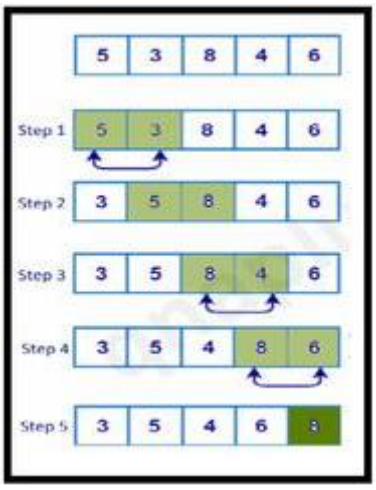
Schreibe ein Programm, das den Selection-Sort-Algorithmus nutzt, um ein Array zu sortieren. Bei diesem Algorithmus wird das kleinste Element im Array gefunden und an den Anfang verschoben, und dieser Prozess wird fortgesetzt, bis das gesamte Array sortiert ist.
Bei jeder Iteration der inneren Schleife wird das kleinste Element durch das Startelement in jeder Schleife ersetzt. Nach dem Ende jeder Schleife erhöhen wir die Startposition um 1 und führen sie bis zum vorletzten Element im Array aus. Wenn wir dies tun, haben wir am Ende der äußeren Schleife ein sortiertes Array.

Das folgende Bild erklärt die Iteration des Selection Sort-Algorithmus.


## Aufgabe 23 (Bubble Sort)

Schreibe ein Programm, das den Bubble Sort Algorithmus verwendet, um ein Array von Ganzzahlen aufsteigend zu sortieren. Das Program soll den Nutzer nach der Anzahl der Zahlen und anschließend den Zahlen die sortiert werden sollen fragen.

Erklärung:
In jeder Iteration der äußeren Schleife wird das größte Element gefunden und mit dem letzten Element in der Schleife vertauscht. In der inneren Schleife vertauschen wir zwei aufeinanderfolgende Elemente paarweise. In jeder inneren Schleife gehen wir vom ersten Element zum einen Element weniger, das wir in der vorherigen Schleife durchgegangen sind. Das Bild unten zeigt die 1. Iteration der inneren Schleife im Bubble Sort-Algorithmus.



## Aufgabe 24 (Linear Search)

Schreibe ein Programm, das ein bestimmtes Element in einem Array sucht, indem es jeden einzelnen Wert sequenziell durchgeht. Das Programm sollte eine Meldung ausgeben, ob das Element vorhanden ist, und falls ja, an welcher Position es sich befindet.

## Aufgabe 25 (Binary Search)

Entwickle ein Programm, das die Binary Search Algorithmus verwendet, um ein bestimmtes Element in einem **sortierten** Array zu finden. Dieser Algorithmus halbiert den Suchbereich in jedem Schritt, bis das Element gefunden ist. Das Programm sollte den Index des gesuchten Elements ausgeben.

Erklärung:

Eingabe:

- Das zu durchsuchende Array, von links nach rechts mit aufsteigend **sortierten** Werten gefüllt. Die gleichen Werte dürfen mehrfach vorkommen. (Das Array kann auch vorgegeben sein, muss nicht vom Nutzer eingelesen werden)
- Der Suchwert

Ausgabe:

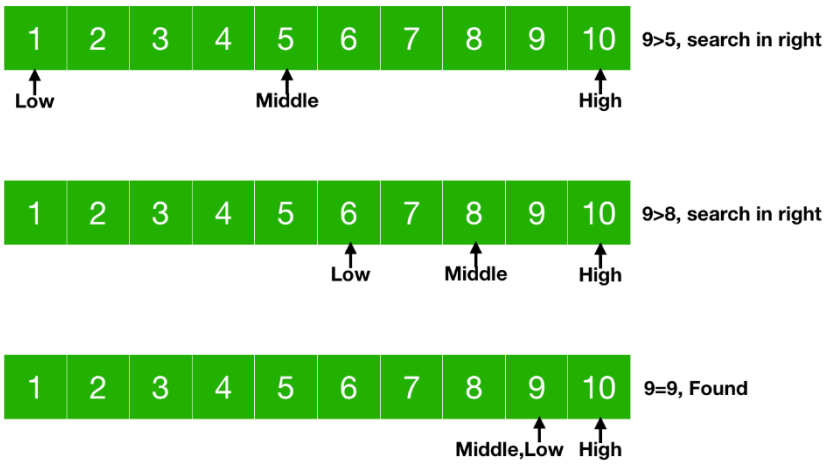
- Wenn der Suchwert gefunden wurde, die Position des Suchwertes im Array.
- Wenn der Suchwert mehrmals im Array vorkommt, ist nicht festgelegt, welche der möglichen Positionen ausgegeben wird.
- Wenn der Suchwert nicht gefunden wurde, ein entsprechender Fehlercode, möglicherweise ergänzt durch die Position, an der der Suchwert stehen würde.

Ablauf:

1. Der Suchbereich umfasst anfangs das komplette Array.
2. Ist der Suchbereich leer, wurde der Suchwert nicht gefunden. Die Suche ist erfolglos beendet.
3. Die Mitte des Suchbereichs wird berechnet und der dort befindliche Wert mit dem Suchwert verglichen.
4. Ist der Suchwert gleich dem Vergleichswert, ist die Suche erfolgreich beendet. Die Position der Mitte wird ausgegeben.
5. Ist der Suchwert kleiner als der Vergleichswert, kann sich der Suchwert nur in der linken Hälfte befinden. Der rechte Rand des Suchbereichs wird auf die Position links von der Mitte eingeschränkt.
6. Ist der Suchwert größer als der Vergleichswert, kann sich der Suchwert nur in der rechten Hälfte befinden. Der linke Rand des Suchbereich wird auf die Position rechts von der Mitte eingeschränkt.
7. Die Suche wird mit dem verkleinerten Suchbereich ab Schritt 2 wiederholt.

Bei diesem Ablauf wird die Länge des Suchbereichs in jedem Schritt halbiert. Spätestens wenn der Suchbereich auf ein einzelnes Element geschrumpft ist, ist die Suche beendet. Dieses eine Element ist entweder das gesuchte Element, oder das gesuchte Element kommt nicht vor.

Search for 9


1 2 3 4 5 6 7 8 9 10
9>5, search in right
Low
Middle
High
1 2 3 4 5 6 7 8 9 10
9>8, search in right
Low
Middle
High
1 2 3 4 5 6 7 8 9 10
9=9, Found
Middle,Low
High