Aufgabe 1: Einkaufsliste

Ziel: Ein Array erstellen und mit Werten füllen.

Aufgaben:

1. Erstellen Sie ein `string`-Array namens `einkaufsliste` mit einer festen Größe von 4.
2. Weisen Sie den vier Plätzen im Array nacheinander die Werte "Milch", "Brot", "Butter" und "Eier" zu. Verwenden Sie dafür den Index-Zugriff (`einkaufsliste[0] = ...`).
3. Geben Sie das Element an der dritten Stelle (Index 2) auf der Konsole aus.

Note:

- Diese Aufgabe festigt die grundlegende Syntax von Deklaration, Initialisierung und Zugriff.
- Achten Sie darauf, dass die Studierenden den Unterschied zwischen der dritten *Stelle* und dem *Index* 2 verstehen.

Aufgabe 2: Lottozahlen

Ziel: Ein Array direkt bei der Erstellung mit Werten initialisieren.

Aufgaben:

1. Erstellen Sie ein `int`-Array namens `lottozahlen`.
2. Initialisieren Sie es direkt mit den folgenden 6 Zahlen: `4, 8, 15, 16, 23, 42`. Verwenden Sie die `{}`-Syntax.
3. Ändern Sie die zweite Zahl (Index 1) auf den Wert `9`.
4. Geben Sie die erste und die letzte Zahl des Arrays auf der Konsole aus.

Note:

- Hier wird die verkürzte Initialisierung geübt.
- Die Aufgabe, die letzte Zahl auszugeben, kann zu einer Diskussion über `lottozahlen.Length - 1` führen.

Aufgabe 3: Alle Personen begrüßen

Ziel: Ein Array mit einer `foreach`-Schleife durchlaufen.

Aufgaben:

1. Erstellen Sie ein `string`-Array `freunde` mit den Namen von drei Ihrer Freunde.
2. Schreiben Sie eine `foreach`-Schleife, die jeden Namen aus dem Array ausgibt.
3. Die Ausgabe sollte für jeden Freund "Hallo, [Name]!" lauten, wobei `[Name]` der jeweilige Name aus dem Array ist.

Note:

- Diese Aufgabe zeigt den Vorteil der `foreach`-Schleife für einfache Lese-Operationen.
- Sie ist ideal, um die Syntax von `foreach` zu verinnerlichen.

Aufgabe 4: Notendurchschnitt berechnen

Ziel: Mit einer `for`-Schleife die Summe der Elemente eines Arrays berechnen.

Aufgaben:

1. Erstellen Sie ein `int`-Array `testNoten` mit den Werten `1, 3, 2, 1, 4`.
2. Erstellen Sie eine `double`-Variable `summe` und initialisieren Sie sie mit `0`.
3. Schreiben Sie eine `for`-Schleife, die alle Noten im Array zu `summe` addiert.
4. Geben Sie am Ende die Gesamtsumme auf der Konsole aus.
5. **Bonus:** Berechnen und geben Sie auch den Durchschnitt aus (`summe / testNoten.Length`).

Note:

- Hier wird bewusst eine `for`-Schleife nahegelegt, obwohl `foreach` auch ginge.
- Die Aufgabe erfordert eine Hilfsvariable (`summe`), ein häufiges Muster.
- Der Bonus erfordert eine Typumwandlung (implizit, da `summe` ein `double` ist), was ein guter Diskussionspunkt ist.

Aufgabe 5: Tic-Tac-Toe-Feld (einfach)

Ziel: Ein 2D-Array erstellen und einen Wert darin platzieren.

Aufgaben:

1. Erstellen Sie ein `char` 2D-Array namens `spielfeld` mit 3 Zeilen und 3 Spalten.
2. Platzieren Sie das Zeichen 'X' in der Mitte des Feldes (Zeile 1, Spalte 1).
3. Platzieren Sie das Zeichen 'O' in der Ecke oben links (Zeile 0, Spalte 0).
4. Geben Sie das Gesamte Spielfeld auf der Console aus (mittels 2 verschachtelten Schleifen)

Note:

- Diese Aufgabe festigt die Syntax für die Erstellung und den Zugriff auf 2D-Arrays.
- Sie ist die direkte Vorbereitung für die finale, größere Aufgabe.

Aufgabe 6: Ampelsteuerung

Ziel: Einen `enum` für einen einfachen Zustand definieren und verwenden.

Aufgaben:

1. Definieren Sie einen `enum` namens `Ampelphase` mit den Werten `Rot`, `Gelb`, `Gruen`.
2. Erstellen Sie eine Variable `aktuellePhase` vom Typ `Ampelphase` und weisen Sie ihr den Wert `Ampelphase.Gruen` zu.
3. Schreiben Sie eine `if`-Anweisung, die prüft, ob `aktuellePhase` gleich `Ampelphase.Gruen` ist.
4. Wenn die Bedingung zutrifft, geben Sie "Du darfst gehen!" auf der Konsole aus.

Note:

- Eine sehr grundlegende Aufgabe, um die Enum-Syntax und den Vergleich zu festigen.
- Dies zeigt den praktischen Nutzen von Enums zur Darstellung von Zuständen.

Aufgabe 7: Interaktives Tic-Tac-Toe

Ziel: Eine primitive, partiell spielbare Version von Tic-Tac-Toe erstellen.

Aufgaben:

1. Erstellen Sie ein `char` 2D-Array `spielfeld` (3x3) und füllen Sie es mit Leerzeichen (`' '`).
2. Erstellen Sie eine `char` Variable `aktuellerSpieler` und setzen Sie sie auf `'X'`.
3. Starten Sie eine Endlosschleife (`while (true)`).
4. **Innerhalb der Schleife:** a. Geben Sie das aktuelle Spielfeld mit verschachtelten `for`-Schleifen aus. b. Fragen Sie den aktuellen Spieler nach seinem Zug (z.B. "Spieler X, gib Zeile und Spalte ein (z.B. 1 1):"). c. Lesen Sie die Eingabe des Benutzers mit `Console.ReadLine()`. d. Zerlegen Sie die Eingabe in zwei Zahlen für Zeile und Spalte. **Tipp:** `input.Split(' ')` und `int.Parse()`. Denken Sie daran, 1 abzuziehen, um auf den Array-Index (0-2) zu kommen! e. Setzen Sie das Zeichen von `aktuellerSpieler` an die eingegebene Position im `spielfeld`. f. Wechseln Sie den Spieler: Wenn `aktuellerSpieler` 'X' war, wird er 'O', ansonsten wird er wieder 'X'.