Übung - Arrays , Kontrollstrukturen & Methoden

Tic Tac Toe – Console Application

Aufgabe – Created by MostWanted38 | Lars – 15.07.2020

http://twitch.tv/mostwanted38

Bauen Sie das Spiel Tic-Tac-Toe nach. Als Spielfeldgrundlage dient ein 2D Array mit dem Namen „spielfeld“ und der Größe 3x3.

(0) **Bevor Sie beginnen… Hier einige Wiederholungsfragen:**

Welche Datentypen gibt es?

Benennen Sie die grundlegenden Datentypen & beschreiben Sie wofür diese jeweils stehen bzw. welche Werte diese annehmen können.
(Hinweis: Ganze Zahlen, Kommazahlen, Zeichen, Text, Wahrheitswerte)

Was ist ein Array?

Beschreiben Sie was ein Array ist, indem Sie etwas Vergleichbares aus der echten Welt nehmen. Beschreiben sie dabei die Eigenschaften und Besonderheiten eines Arrays.

Erweitern Sie Ihr Beispiel aus dem echten Leben von einem Ein-Dimensionalen Array auf ein 2D-Array (Hinweis: ein Array von Arrays) und beschreiben Sie was sich ändern würde.

Was ist eine Methode?

Benennen Sie die Elemente einer Methode & beschreiben Sie wofür welche Komponenten stehen?
(Hinweis: Name, Rückgabewert, Eingabewerte … ?)

Erklären Sie was die nachfolgenden 2 Methoden machen & zeigen Sie den Unterschied zwischen den beiden auf

```
public void Addiere1_value(int x) { x += 1; }

public void Addiere1_ref(ref int x) { x += 1; }
```

(Hinweis: Unterschied zwischen “Call by Value“ und „Call by Reference“)

Was ist eine Konstrollstruktur?

Welchen Arten von Anweisungen gibt es?

Wie funktioniert eine Anweisung?

Welche Arten von Schleifen gibt es?

Erklären Sie den logischen Ablauf einer Schleife!

Bauen Sie den logischen Ablauf einer for-schleife nach indem sie die Logik der for-schleife als while-Schleife darstellen. Was ändert sich dadurch?

Aufgaben:

(1) Entscheiden sie sich bei dem Array „spielfeld“ für einen geeigneten Datentyp
(Hinweis: Alle Datentypen, die mehr als 3 Werte annehmen können sind geeignet)

(2) Entscheiden Sie sich zu Beginn für 3 Werte innerhalb des in (1) gewählten Datentyps, die verwendet werden sollen um folgendes darzustellen:
Leeres Spielfeld
Spielfeld ist besetzt von Spieler 1
Spielfeld ist besetzt von Spieler 2

(3) Implementieren/Programmieren Sie eine Methode namens „**SpielfeldLesen**“ die das gesamte 2D Array „spielfeld“ auf der Console ausgibt. Überlegen Sie sich wie Sie das 3x3 Spielfeld auf der Console darstellen können

(4) Implementieren/Programmieren Sie eine Methode namens „**SpielfeldErzeugen**“ die das gesamte „spielfeld“ zurücksetzt (alle Felder leer). Überlegen Sie wie sie **ALLE Werte** im 2D Array „spielfeld“ auf „Leeres Spielfeld“ setzen

(5) Implementieren/Programmieren Sie eine Methode namens „**SpielfeldBereitsBesetzt**“ die an der Stelle/Position[X, Y] auf dem „spielfeld“ testet ob ein Feld bereits durch einen Spieler besetzt ist. Entscheiden Sie Sich für geeignete Methoden-Parameter!

(6) Implementieren/Programmieren Sie eine Methode namens „**SpielfeldBesetzen**“, die an der Stelle/Position [X, Y] auf dem „spielfeld“ den Wert des entsprechenden Spielers setzt.
   (a) Überlegen Sie welche Datentypen für diese Methode geeignet sind & Entscheiden Sie sich für geeignete Parameter & Rückgabewerte.
   (b) Fangen Sie ungültige Spielfeld-Positionen (X, Y) ab!
   Beispiele: [-1,-1], [-1,0], [4,4]
   (c) Beachten Sie dabei, dass Sie das Spielfeld nur dann besetzen können, wenn das Feld leer ist.
   (d) Bei Erfolg soll die Methode „Hat funktioniert“ und bei Scheitern „Hat nicht funktioniert“ zurückgeben.

(7) Implementieren/Programmieren Sie eine Funktion namens „**RundeIstBeendet**“, welche prüfen soll ob entweder:
   a) Das gesamte Spielfeld (alle 9 Felder) von Spielern belegt worden ist
   ODER
   b) Spieler 1 jetzt 3 Felder in einer Reihe (horizontal, vertikal oder diagonal) besetzt hat
   ODER
   c) Spieler 2 jetzt 3 Felder in einer Reihe (horizontal, vertikal oder diagonal) besetzt hat

   Entsprechend des zutreffenden Falles (a), (b) oder (c) soll die Methode einen passenden Wert zurückgeben:

   - Bei (a) - „Keiner hat Gewonnen“
   - Bei (b) - „Spieler 1 hat Gewonnen“
   - Bei (c) - „Spieler 2 hat Gewonnen“

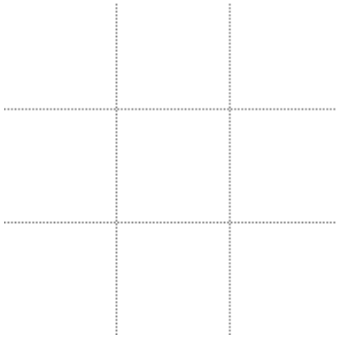
   Entscheiden Sie sich auch hier für einen geeigneten Datentyp!

   Hinweis: Sie können doppelten Code vermeiden indem Sie eine Hilfs-Methode „SpielerHatGewonnen“ für (7)(b) und (7)(c) schreiben, welche für den entsprechenden Spieler die jeweiligen benötigten Kombinationen testet & am Ende „Gewonnen“ oder „Nicht Gewonnen“ zurückgibt.

   Als Hilfe dient das Gedankenspiel auf der nachfolgenden Seite.

Gedanken:

- Wann hat ein Spieler gewonnen?
- In welchen Richtungen kann ein Spieler 3 Felder besetzen?
- Welche der 9 Felder müssen in Kombination von einem Spieler besetzt sein, damit dieser gewonnen hat?
- Wann hat ein Spieler nicht gewonnen?



Machen Sie sich Notizen, Gedanken & ggf. eine Zeichnung zur Hilfe!

Welche Kombinationen von Feldern können besetzt werden damit in (b) oder (c) gesagt wird „Spieler1“ oder „Spieler2“ hat gewonnen?

(8) Implementieren/Programmieren sie nun alles, was sie in (1) bis (7) entworfen haben, in die **public static void Main(string[] args)** um einen Spielablauf zu erzeugen.
Überlegen Sie Was, Wann & In Welcher Reihenfolge passieren muss, Implementieren Sie das Spiel & Testen Sie diese Spiellogik!

(9) Überlegen Sie Sich, wie sie das Spiel Neustarten können, wenn „**RundeIstBeendet**“ über das Ende eines Spiels Bescheid gibt.

Allgemeine Hinweise zum Spielablauf

(a) Zu Spielbeginn ist das Spielfeld komplett leer
(b) Immer wenn jemand ein Spielfeld besetzt muss das Spielfeld in der Console aktualisiert bzw neu ausgegeben werden.
(c) Immer wenn jemand ein Spielfeld besetzt wird muss überprüft werden ob die Runde beendet ist (mit oder ohne Gewinner)

Vertiefung

(10) Um ihren bisher erstellten Code nachzuvollziehen, nehmen sie sich Zeit und Visualisieren & Erklären sie die Logik ihrer Methoden mittels eines beliebigen Diagramms.
(Ziel: Erlernen Code-Ideen in Form eines Schaubilds darzustellen)

Empfehlenswert wäre ein
Programm-Ablauf-Plan (PAP) bzw.
Programm-Ablauf-Graph (PAG) bzw.
https://de.wikipedia.org/wiki/Programmablaufplan

Nutzen Sie für diese Aufgabe gerne Haltepunkte
(Visual Studio: F9 = Haltepunkt in gewählter Zeiler setzen)
um Schritt für Schritt einzeln
(Visual Studio: F10 = Zur Nächsten Anweisung gehen)
zu sehen wie das Programm abläuft.

Zum Experimentieren und Weiterlernen:

Wie kann dieses Projekt erweitert werden?

(11) Implementieren sie alles Nötige um die Anzahl der Spielzüge, die in der Runde benötigt wurden bis zum Spielende, abzuspeichern. Überlegen Sie Sich ob Sie weitere Variablen benötigen und schreiben sie eine Methode „**MerkeAnzahlSpielzuege**“ dafür.

(12) Implementieren sie alles Nötige um die Anzahl der gespielten Runden mitzuzählen & zu speichern. Überlegen Sie Sich ob Sie weitere Variablen benötigen und schreiben sie eine Methode namens „**MerkeAnzahlGespielterRunden**“

(13) Zu Spielbeginn wird vom Benutzer entschieden wie groß das Spielfeld ist. Zum Beispiel wünscht dieser sich nun ein Spielfeld der Breite 10 und Höhe 6.

Ändern Sie die in (4) erstellte Methode „**SpielfeldErzeugen**“ so ab, dass diese nun die Parameter „Breite“ und „Höhe“ annimmt und

a) Fangen Sie alle ungültigen Werte ab (Minimale Spielfeldgröße 3x3)
b) Das „spielfeld“ muss mit entsprechender Größe neu erzeugt werden
c) Alle Spielfelder im „spielfeld“ werden auf „Spielfeld leer“ gesetzt

Ändern Sie die in (3) erstellte Methode „**SpielfeldLesen**“ so ab, dass diese entsprechend das Spielfeld der Größe MxN auf der Console ausgeben kann.

(Hinweis: Was ist M? Was ist N?)

Überlegen Sie welche weiteren Methoden von (5) bis (9) entsprechend angepasst werden müssen

(14) Lassen Sie vor Spielbeginn den Benutzer entscheiden welchen Wert dieser für seine Spielfeldbelegung verwenden möchte.
Beachten sie dabei den in (1) und (2) gewählten Datentyp!
Wandeln Sie die Benutzereingabe entsprechend um!

Überlegen Sie welche Variablen benötigt werden.
Implementieren Sie alles Benötigte in Form einer neuen Methode mit dem Namen „**AendereSpielerDarstellung**“.
(Hinweis [C#]: Methode mit Referenzübergabe - ref)

(15) Denken Sie sich noch weitere Änderungen & Erweiterungen für dieses Projekt aus! Spielen Sie mit dem bisherigen Code herum!

Aufgabe – Created by MostWanted38 | Lars – 15.07.2020

http://twitch.tv/mostwanted38

Kommentar:

Fortgeschrittene Programmierer sollten sich als Obergrenze eine Zeit von maximal 60 oder 90 Minuten geben um die Aufgaben (1) bis (14) zu implementieren bzw. abzuarbeiten.

Einsteiger, Anfänger & Neulinge dürfen sich hier mehrere Stunden bzw. einen halben Tag Zeit in Anspruch nehmen um die Grundlagen dieser 3 Themen zu verstehen & zu vertiefen.

Ziel dieses Aufgabenblatts ist es, dass die Grundlagen verstanden werden & danach bevor man in einem neuen Projekt wieder eine Kontrollstruktur, Array oder Methode schreibt schon im Kopf eine Vorstellung hat was in welcher Reihenfolge passiert & überlegt was brauche ich, wofür brauche ich das & welche Werte muss ich dafür abfragen/anlegen