**19. August 2024**

**C# Anwendungsmöglichkeiten:**

* **Geschäftsanwendungen:** Daten erfassen, analysieren, verarbeiten.
* **Webanwendungen:** Dynamische Inhalte im Webbrowser.
* **Spiele:** 2D- und 3D-Games.
* **Finanz-/Wissenschaftliche Anwendungen:** Berechnungen, Analysen.
* **Cloudbasierte Anwendungen:** Nutzung von Cloud-Diensten.
* **Mobile Anwendungen:** Apps für Smartphones und Tablets.

**Begriffe erklärt:**

* **Konsole:** Ein Textfenster, das Eingaben annimmt und Ausgaben anzeigt.
* **Syntax:** Die Regeln und Struktur, die bestimmen, wie Code geschrieben werden muss.

1. **.NET-Editor:**
   * **Editor im Browser**: Erlaubt das Schreiben und Ausführen von C#-Code direkt im Browser.
2. **Codekommentare:**
   * **Kommentar erstellen:** Zwei Schrägstriche // vor eine Codezeile setzen.
   * **Zweck:** Vorübergehendes Deaktivieren von Code oder Notizen im Code hinzufügen.
3. **Unterschied zwischen Console.WriteLine() und Console.Write():**
   * **Console.WriteLine():** Gibt Text aus und springt in die nächste Zeile.
   * **Console.Write():** Gibt Text aus, ohne in die nächste Zeile zu springen.

**Begriffe erklärt:**

* **Compiler:** Übersetzt den Code in ein für den Computer ausführbares Format.
* **Konsole:** Ein Textfenster für Eingaben und Ausgaben.
* **Zeilenumbruch:** Bewegung des Cursors in die nächste Zeile (ähnlich wie die "Enter"-Taste).
  + **Quellcode:** Der von Entwicklern geschriebene Code, der später in eine vom Computer verständliche Form umgewandelt wird.

1. **Kompilierung:**
   * **Definition:** Prozess, bei dem der Quellcode in Maschinencode umgewandelt wird, den die CPU ausführen kann.
   * **Compiler:** Spezielles Programm, das diese Umwandlung durchführt.
2. **Syntax:**
   * **Definition:** Regeln für das Schreiben von Code, z.B. Zeichensetzung und Struktur.
   * **Syntaxhervorhebung:** Visuelles Hilfsmittel im Editor, um Fehler in der Code-Syntax leichter zu erkennen.
3. **Codeanalyse:**
   * **Console.WriteLine("Hello World!");**
     + **Console:** Klasse, die Methoden zur Konsolenausgabe enthält.
     + **WriteLine():** Methode zur Ausgabe einer Textzeile in die Konsole.
     + **Punkt (.):** Operator zum Zugriff auf Methoden innerhalb einer Klasse.
     + **Semikolon (;):** Beendet eine Anweisung in C#.

-Ein Literalwert ist ein konstanter und unveränderlicher Wert.

* **'b'**: Einzelnes Zeichen (char), verwendet einfache Anführungszeichen.
* **"b"**: Zeichenkette (string), verwendet doppelte Anführungszeichen.

**Erklärung:**

* **Einzelnes Zeichen (char):** Ein **char** speichert genau ein Zeichen, wie z.B. 'b', 'A', '1', oder '@'.
* **Zeichenkette (string):** Ein **string** ist eine Folge von Zeichen, z.B. "Hallo", "123", oder "b". Strings können aus einem oder mehreren Zeichen bestehen.

-**int**=integer(ganze Zahl) - Benötigt keinen **string** oder **char**

* **float**: Genauigkeit von ca. 6-9 Dezimalstellen.
* **double**: Genauigkeit von ca. 15-17 Dezimalstellen.
* **decimal**: Genauigkeit von ca. 28-29 Dezimalstellen.

**Erklärung:**

* **Gleitkommazahlen**: Zahlen mit Dezimalstellen, z.B. 3,14159.
* **Genauigkeit**: Anzahl der Ziffern nach dem Dezimaltrennzeichen, die der jeweilige Datentyp korrekt darstellen kann.

Float=“f“ am schluss der zahl (literalsuffix)

Double=Standardmäßiger Datentyp für Dezimalzahlen ohne Suffix

Decimal=literalsuffix „m“

**Boolesche Literale (bool)** in C#:

* **true**: Wahr
* **false**: Falsch

**Verwendung:** Boolesche Literale werden verwendet, um Bedingungen in Programmen zu überprüfen, z.B. in **if-Abfragen** oder Schleifen, um zu entscheiden, ob ein bestimmter Codeblock ausgeführt wird.

**Funktionen der Datentypen:**

* **string**: Wird für Texte, Wörter und alphanumerische Daten verwendet, die nicht für Berechnungen gedacht sind (z.B. Telefonnummern, Postleitzahlen).
* **char**: Speichert ein einzelnes Zeichen.
* **int**: Verwendet für ganze Zahlen, die in Berechnungen verwendet werden.
* **decimal**: Wird für Dezimalzahlen verwendet, insbesondere wenn hohe Präzision erforderlich ist (z.B. in finanziellen Berechnungen).
* **bool**: Repräsentiert einen Wahrheitswert (true oder false), z.B. für logische Entscheidungen.

**Richtige Wahl des Datentyps:**

* Wählen Sie den Datentyp basierend auf dem Verwendungszweck:
  + **Darstellung:** Verwenden Sie string oder char.
  + **Berechnungen:** Verwenden Sie int oder decimal.
  + **Logik:** Verwenden Sie bool.

**Was ist eine Variable?**

* **Definition:** Ein Container, der einen Wert speichert, der sich während der Programmausführung ändern kann.
* **Zweck:** Speichern, Lesen und Ändern von Werten im Code.

**Deklaration einer Variable:**

* **Syntax:** Datentyp Variablenname;
* **Beispiel:** string firstName; – Erstellt eine string-Variable mit dem Namen firstName, die Zeichenfolgen speichert.

**Bsp. Deklarieren mit variabel:**

string firstName;

firstName = "Bob";

Console.WriteLine(firstName);

**var** steht in C# für eine **implizite Typdeklaration**. Der Compiler bestimmt den Datentyp der Variable automatisch basierend auf dem zugewiesenen Wert.

var number = 42; // `number` wird als `int` erkannt

var name = "Alice"; // `name` wird als `string` erkannt

var pi = 3.14; // `pi` wird als `double` erkannt

string firstName = "Bob";

int messageCount = 3;

double temperatureCelsius = 34.4;

Console.WriteLine($"Hallo, {firstName} du hast {messageCount} neue Nachrichten im Postfach. Es sind {temperatureCelsius} Grad Celsius draussen!");

$ macht das man in einer linie mit den klammern arbeiten kann. **IMMER GANZ VORNE IN KLAMMER**

Man kann mit **{}** in einem satz die variabeln einfügen.

\n =Neue Zeile

\t =Tastopp