Grundlagen der Programmierung in C# – Ein- und Ausgabe

Die Ein- und Ausgabe wird bei Konsolenanwendungen grundsätzlich über folgende Methoden gesteuert:

- Ausgabe:
  - **Console.Write();** – schreibt nach dem Cursor den auszugebenden Text
  - **Console.WriteLine();** – schreibt nach dem Cursor den auszugebenden Text und wechselt dann mit dem Cursor in die nächste Zeile
- Eingabe:
  - **Console.Read();** – liest das nächste Zeichen ein
  - **Console.ReadLine()** – liest die gesamte Zeile ein

Ausgabe

Prinzipiell sind die Ausgabemethoden so gestrickt, dass sie alle gängigen Datentypen ausgeben können. Man kann also in die Klammern sowohl eine Zeichenkette als auch eine Zahl schreiben und dieses ausgeben lassen.

```
1 Console.WriteLine("Hallo Welt"); //Ausgabe: Hallo Welt!
2 Console:WriteLine(123); //Ausgabe: 123
3
4 double zahl = 3.14;
5 Console.WriteLine(zahl); //Ausgabe: 3,14
```

Achtung Intern verlangt C# (weil die Sprache in Amerika entwickelt wurde) bei Dezimalzahlen einen Dezimalpunkt statt eines Kommas. D. h. bei der Initialisierung von Dezimalen muss der **Punkt** statt des Kommas geschrieben werden. Bei der Eingabe über die Konsole jedoch muss aber das Komma benutzt werden, da sich diese an den Einstellungen des Betriebssystems orientiert (wir befinden uns in Deutschland, wo das Komma üblich ist).

Natürlich können auch verschiedene Variablen und Zeichen in der Ausgabe kombiniert werden. Dazu nutzt man das **+**. So *verkettet* man die einzelnen Bestandteile miteinander:

```
1 string ausgabe = "Heute ist";
2
3 Console.WriteLine(ausgabe + " " + "Dienstag" + 3); //Ausgabe: Heute ist Dienstag3
```

Eingabe

Die Eingabemethoden geben jeweils einen Wert zurück: das eingelesene Zeichen.

Achtung Die Methode *Console.Read()* gibt einen *int*-Wert zurück, d. h. das eingelesene Zeichen wird in ihren ASCII-Code umgerechnet. Wird z. B. ein A eingelesen, wird intern der Wert 65 hinterlegt. Deshalb ist die Funktion *Console.ReadLine()* zu bevorzugen, da diese eine Zeichenkette, also einen *string* zurück gibt.

Die eingelesenen Zeichen können natürlich gespeichert werden:

```
1 string eingabe = Console.ReadLine();
```

Aufgabe 1. Schreibe ein kleines Programm, in dem der Nutzer einen Namen eingeben soll. Dieser wird dann in einer Variable **name** mit dem Datentyp *string* gespeichert. Anschließend soll das eigegebene Wort im folgenden Satz ausgegeben werden: „Hallo, meine Nam eist . . . . Ich wünsche dir einen schönen Tag.“ Statt der drei Punkte soll der eigegebene Name stehen.

Lösung

```
string name;

Console.Write("Gibt einen Namen für mich ein: ");
name = Console.ReadLine();

Console.WriteLine("Hallo, mein Name ist " + name
      + ". Ich wünsche dir einen schönen Tag");
```

Man möchte aber nicht nur Zeichenketten einlesen. Manchmal soll der Nutzer z. B. eine Zahl eingeben. Leider führt der folgende Code zu einer Fehlermeldung:

```
Console.WriteLine("Gib eine ganze Zahl ein!");
int eingabe = Console.ReadLine();
```

Die Methode *Console.ReadLine()* gibt immer eine Zeichenkette zurück. Diese muss nun in eine ganze Zahl *umgerechnet* werden. In der Informartik bezeichnet man dies als **Konvertieren**. C# stellt fast für jede Konvertierung eine Funktion zur Verfügung:

```
Console.WriteLine("Gib eine ganze Zahl ein!");
string eingabe = Console.ReadLine();

int zahl = Convert.ToInt32(eingabe);
```

Beispiele:

```
Convert.ToInt32("-123"); //wandelt den String in eine ganze Zahl um
Convert.ToDouble("3,14"); //wandslet den String in eine Dezimlazahl um
Convert.ToString(123); //wandelt die ganze Zahl in eine Zeichenkette um
Convert.ToBoolean(1); //Wandelt die 1 in den Wahrheitswert true um
```

Aufgabe 2. Schreibe ein Programm, in dem der Nutzer eine ganze Zahl eingeben soll. Das Programm soll dann das Vierfache der Zahl ausgeben.

Lösung:

```
int zahl;
string eingabe;

Console.Write("Gib eine ganze Zahl ein: ");
eingabe = Console.ReadLine();
zahl = Convert.ToInt32(eingabe);

zahl = zahl * 4;
Console.WriteLine("Das Vierfache deiner Zahl lautet " + zahl);
```