# Unsere ersten Anweisungen

- Kommunikation mit dem Benutzer über die Konsole
  - User Interface
- Ausgabe

```
System.Console.WriteLine(text);
```

  - Gibt den Text auf den Bildschirm aus und positioniert Cursor an den Beginn der nächsten Zeile (Zeilenvorschub)

```
System.Console.Write(text);
```

  - Gibt den Text auf den Bildschirm aus und positioniert Cursor an das Ende der ausgegebenen Zeile
- Eingabe von Texten

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

  - Liest eine Zeile (bis zum Drücken der Enter-Taste) ein und speichert den Text in der Stringvariablen `eingabe`

# Einlesen von Zahlen

```
Console.Write("Bitte geben sie eine Zahl ein: ");
string userInput = Console.ReadLine();

int value = Convert.ToInt32(userInput);

double decimalValue = Convert.ToDouble(userInput);
```

- Alternative

```
int value = int.Parse(userInput);
double decimalValue = double.Parse (userInput);
```

# Bsp: Einlesen eines Gleitkomma-Bruches

```
Console.Write("Enter numerator: ");
string userInput = Console.RealdLine();
double numerator = Convert.ToDouble(userInput);

Console.Write("Enter denominator: ");
userInput = Console.RealdLine();
double denominator = Convert.ToDouble(userInput);

double decimalValue = numerator / denominator;

Console.WriteLine($"{numerator}/{denominator}={decimalValue}");
```

# Ausgabe von Variablen mit Platzhaltern

```
Console.Write($"Hello {name}");
Console.WriteLine($"{x} = {y}");
```

**Platzhalter-Syntax**

"{" variable ["," width] [":" format [precision]] "}"

| | |
|---|---|
| *variable* | Variablenname |
| *width* | Feldbreite (wenn zu klein, wird sie überschritten)<br>positiv = rechtsbündig, negativ = linksbündig |
| *format* | Formatierungscode (z.B. d, f, e, x, ...) |
| *precision* | Anzahl der Nachkommastellen (machmal Anzahl der Ziffern) |

Beispiel: {floatingPointNumber,10:f2}

# Ausgabe von Variablen mit Platzhaltern 2

```
Console.Write("Hello {0}", name);
Console.WriteLine("{0} = {1}", x, y);
```

**Platzhalter-Syntax**

"{" n ["," width] [":" format [precision]] "}"

| | |
|---|---|
| *n* | Argumentnummer (beginnend bei 0) |
| *width* | Feldbreite (wenn zu klein, wird sie überschritten)<br>positiv = rechtsbündig, negativ = linksbündig |
| *format* | Formatierungscode (z.B. d, f, e, x, ...) |
| *precision* | Anzahl der Nachkommastellen (machmal Anzahl der Ziffern) |

Beispiel: {0,10:f2}

# Formatierungscode für Zahlen

| | | |
|---|---|---|
| d, D | Dezimalformat (ganze Zahl mit führenden Nullen)<br>precision = Anz. Ziffern | -xxxxx |
| f, F | Fixpunktformat<br>precision = Anz. Nachkommastellen (Default = 2) | -xxxxx.xx |
| n, N | Nummernformat (mit Tausender-Trennstrich)<br>precision = Anz. Nachkommastellen (Default = 2) | -xx,xxx.xx |
| e, E | Exponentialformat (groß/klein signifikant)<br>precision = Anz.Nachkommastellen | -x.xxxE+xxx |
| c, C | Currency-Format<br>precision = Anz. Nachkommastellen (Default = 2)<br>Negative Werte werden in Klammern gesetzt | $xx,xxx.xx<br><br>($xx,xxx.xx) |
| x, X | Hexadezimalformat (groß/klein signifikant)<br>precision = Anzahl Hex-Ziffern (evtl. führende 0) | xxx |
| g, G | General (kompaktestes Format für gegebenen Wert; Default) | |

# Zahlenformatierung: Beispiele

```
int x = 17;

Console.WriteLine($"{x}");          17
Console.WriteLine($"{x,5}");           17

Console.WriteLine($"{x:d}");        17
Console.WriteLine($"{x,5:d3}");        017

Console.WriteLine($"{x:f}");        17.00
Console.WriteLine($"{x:f1}");       17.0

Console.WriteLine($"{x:E}");        1.700000E+001
Console.WriteLine($"{x:e1}");       1.7e+001

Console.WriteLine($"{x:x}");        11
Console.WriteLine($"{x:x4}");       0011
```