



# ***Sequenzen: Lineare Programme***

# Algorithmus

Ein **Algorithmus** ist eine **exakte und eindeutige** Beschreibung eines **Lösungsverfahrens** als Abfolge von **einzelnen Schritten**.

Beispiel:

- Kochen nach Rezept
- Navi nutzen
- Tanzen
- Bedienen nach Betriebsanleitung
- Suchen von Extrem- und Wendepunkte

Ein **Programm** ist ein Algorithmus, der in einer **Computersprache** formuliert wird.

# Das EVA-Prinzip

**Eingabe**



**Verarbeitung**



**Ausgabe**

Rezept, Zutaten

Topf, Ofen (Hardware)  
Kochen (Software)

Lecker



# ***Ein- und Ausgabe***

# Ausgabe in C#

**Tipp für Visual Studio:**

**CW <Tab><Tab>**

**->**

**Console.WriteLine**

- `Console.WriteLine("Hallo"); // Ausgabe von Hallo und`  
`Console.WriteLine("Welt"); // springt in die nächste Zeile`
  - Ausgabe:  
Hallo  
Welt
- `Console.Write("Hallo"); // Ausgabe von Hallo und`  
`Console.WriteLine("Welt"); // bleibt in der Zeile Zeile`
  - Ausgabe: HalloWelt
- `string text="Hallo";`  
`int zahl=5;`  
`Console.WriteLine(text+"Welt"+zahl+ "! ");`
  - Ausgabe: HalloWelt5!

## Zusatz: Ausgabe in C#

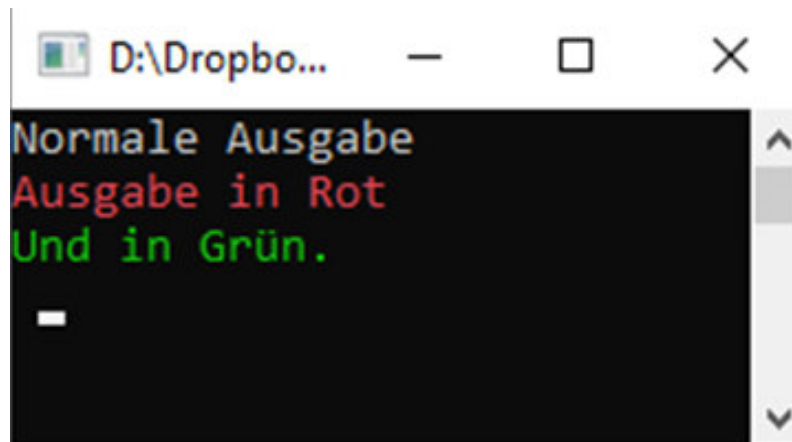
### Formatierter Ausgabestring

- // Platzhalter {1} {2} {3} ...  
int x=1,y=2,z=3;  
Console.WriteLine("x={0} y={1} z={2}", x, y, z);
- // Platzhalter \$"...{var}..."  
Console.WriteLine(\$"x={x} y={y} z={z}");
  - Ausgabe: x=1 y=2 z=3
- // Ausgabe über mehrere Zeilen  
Console.WriteLine(@"Mehrere Zeilen  
mit  
Sonderzeichen: "a"");
  - Ausgabe:  
Mehrere Zeilen  
mit  
Sonderzeichen: "a"

## Zusatz: Ausgabe in C#

### Farbige Ausgabe

```
Console.WriteLine("Normale Ausgabe");  
Console.ForegroundColor = ConsoleColor.Red;  
Console.WriteLine("Ausgabe in Rot");  
Console.ForegroundColor = ConsoleColor.Green;  
Console.WriteLine("Und in Grün.");  
Console.ForegroundColor = ConsoleColor.White;
```

A screenshot of a Windows console window. The title bar shows the file path "D:\Dropbo...". The console output displays three lines of text: "Normale Ausgabe" in white, "Ausgabe in Rot" in red, and "Und in Grün." in green. A white cursor is visible on the line following the last output.

```
D:\Dropbo...  
Normale Ausgabe  
Ausgabe in Rot  
Und in Grün.  
_
```

# Eingabe in C#: Console.ReadLine()

```
class Program {  
    0 Verweise  
    static void Main(string[] args) {  
  
        Console.Write("Bitte Namen eingeben: ");  
        string name = Console.ReadLine();  
        Console.WriteLine("Guten Tag " + name);  
        Console.Write("Bitte Alter eingeben: ");  
        int alter = Convert.ToInt32(Console.ReadLine());  
        int alterNächstesJahr = alter + 1;  
        Console.WriteLine("Nächstes Jahr wirst Du " + alterNächstesJahr);  
  
        Console.ReadKey();  
    }  
}
```

**Einlesen eines Strings  
über die Tastatur**

**String in int  
konvertieren**



## Zusatz Eingabe in C#: Console.ReadKey(true) Warten auf bestimmten Tastendruck

```
Console.WriteLine("Weiter mit Taste 'c'...");  
ConsoleKeyInfo keyInfo = Console.ReadKey(true);  
if (keyInfo.KeyChar == 'c') { Console.WriteLine("Gut gemacht!"); }  
//Eingabetaste: keyInfo.Key != ConsoleKey.Enter  
//Cursortaste: keyInfo.Key==ConsoleKey.LeftArrow
```



# ***Struktogramme***

## Beispiel Lineares Programm: Brutto berechnen

### Brutto

netto einlesen

prozent einlesen

$\text{steuern} = \text{netto} * \text{prozent} / 100$

$\text{brutto} = \text{netto} + \text{steuern}$

brutto ausgeben

## Arbeitsauftrag



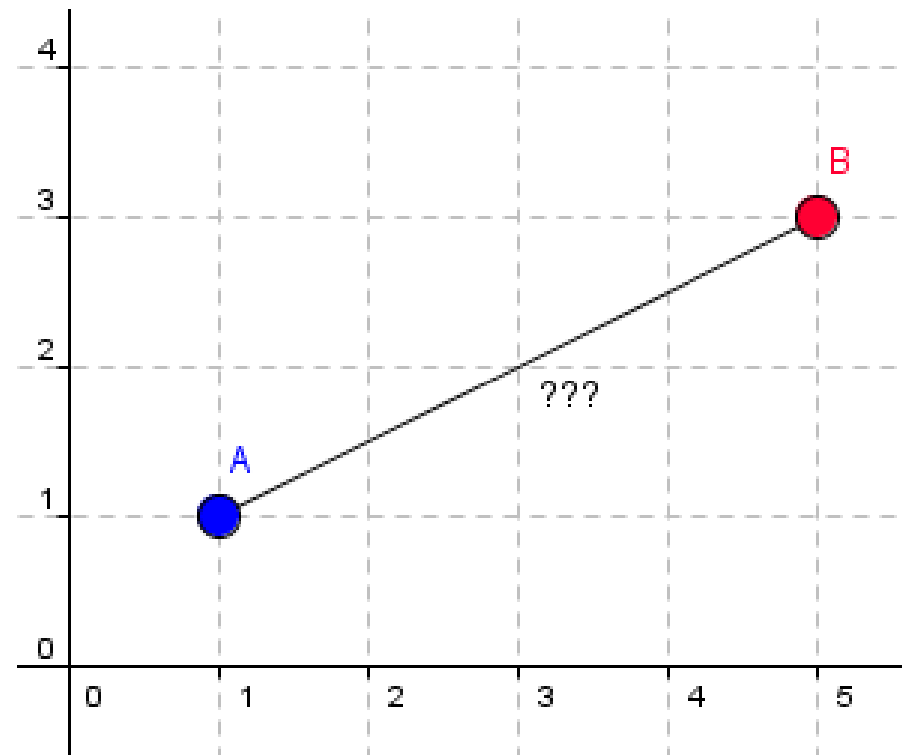
**Schreiben Sie das C#-Programm  
zum Struktogramm „Brutto“**

## Arbeitsauftrag



**Entwickeln Sie ein Struktogramm „Netto“  
(aus Brutto und Prozent wird Netto  
ausgerechnet)**

# Arbeitsauftrag



**Entwickeln Sie ein Struktogramm,  
das den Abstand von zwei  
Punkten in der Ebene ermittelt.**

# Arbeitsauftrag



## Würfel-Simulation schreiben

```
Random random = new Random();  
int wuerfel = random.Next(0, 2);
```



# ***Typumwandlungen (Typecasting)***



# Implizites und explizites Typecasting

- int → long
  - int iZahl=1000;  
long loZahl=1000000;  
loZahl=iZahl; // funktioniert implizit
- long → int
  - iZahl = (int) loZahl; // explizit durch den Typecast-Operator
- float → double
  - float fZahl=3.14f;  
double dZahl=77.7;  
dZahl = fZahl; // funktioniert implizit
- double → float
  - fZahl = (float) dZahl; // explizit durch den Typecast-Operator

# Explizites Typcasting zwischen verschiedenen Datentypen

- `int`  $\leftrightarrow$  `float`
  - `int iZahl1=5;`  
`float fZahl1=10.8f;`  
  
`fZahl1 = (float) iZahl1;`  
`iZahl1 = (int) fZahl1; // danach hat iZahl1 den Wert 10`
- `string`  $\leftrightarrow$  `int`, `string`  $\leftrightarrow$  `float`
  - `string sZahl1 = "56";`  
`int iZahl1 = 5;`  
`double dZahl1 = 10.8;`  
`iZahl1 = Convert.ToInt32(sZahl1);`  
`dZahl1 = Convert.ToDouble(sZahl1);`  
`sZahl1 = Convert.ToString(iZahl1);`

## Zusatz Typecasting

### Begriffe boxing und unboxing

- `int i = 123;`
- `object o = i;` // **Boxing**: Spezieller Typ wird verallgemeinert
- `int j = (int)o;` // **Unboxing**: Allgemeiner Typ wird spezialisiert

## Typecasting Beispiele 1

- `iZahl = (int) fZahl1 * (int) fZahl2;`  
oder  
`iZahl = (int) (fZahl1 * fZahl2);` // Klammern sind hier wichtig!
- `dZahl2 = fZahl1;` // geht implizit
- `fZahl = (float)(iZahl1 + iZahl2);` // expliziter Typecast-Operator
- `sZahl1 = Convert.ToString(loZahl1 + (long) iZahl1);`
- `iZahl1 = 1;`  
`fZahl1 = iZahl1 / 3;` //Vorsicht: fZahl1=0!! da Integer-Division  
`fZahl1 = iZahl1 / 3f;` // fZahl1=0.3333

## Typecasting Beispiele 2

- Kommazahlen abschneiden

```
fZahl1= 45.765f;
```

```
iZahl1= (int) fZahl1; // erhält nur den ganzzahligen Anteil, also 45
```

- Runden einer float-Zahl

```
float fZahl1=45.49f;
```

```
int iZahl1 = (int) (fZahl1+0.5f);
```

- Runden auf 3 Kommastellen

```
float fZahl1=45.123456f;
```

```
float fZahl3 = (int)(fZahl1*1000+0.5f)/1000f;
```