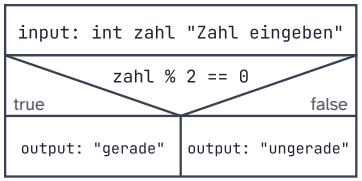
Struktogramme lesen und zeichnen

Spickzettel Struktogramm · Grundlagen der Programmierung

Grundbausteine · Ablauf · Unterprogramme · Hilfe

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

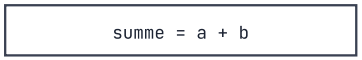
Algorithmus — Beispiel



Eine genaue Schritt-für-Schritt-Anleitung, mit der man eine Aufgabe löst. Jeder Schritt ist eindeutig, nach endlich vielen Schritten ist Schluss.

Ein **Struktogramm** zeigt einen Algorithmus als Bild aus Kästen und wird von oben nach unten gelesen (Nassi-Shneiderman, DIN 66261).

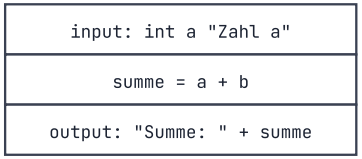
Anweisung — statement



Ein einzelner Schritt: eine Zuweisung, eine Rechnung, eine Eingabe oder eine Ausgabe.

```
summe = a + b;
```

Sequenz — sequence



Anweisungen laufen nacheinander von oben nach unten, jede genau einmal.

`input:` liest einen Wert ein, `output:` gibt etwas aus.

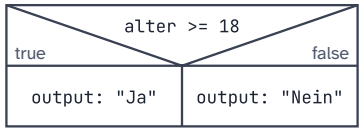
Aufruf — Unterprogramm



Startet ein Unterprogramm, also ein eigenes Struktogramm. In den Klammern stehen die Parameter. Eine Funktion gibt mit `return` einen Rückgabewert zurück.

```
int e = verdopple(5);
```

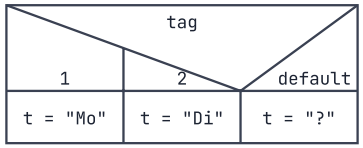
Verzweigung — if-else



Die Bedingung wird geprüft. Ist sie `true`, läuft der linke Zweig, sonst der rechte. Ein Zweig darf leer bleiben (ø).

```
if (alter >= 18) { … }
else { … }
```

Fallauswahl — switch



Vergleicht den Wert einer Variablen mit festen Fällen. Genau ein Fall wird ausgeführt. Passt keiner, läuft `default`.

```
switch (tag) {
  case 1 -> t = "Mo";
  case 2 -> t = "Di";
  default -> t = "?";
}
```

Platzhalter — leeres Element



Hier passiert nichts, zum Beispiel in einem leeren false-Zweig. In VisuStruct ziehst du ein Element darauf, dann wird der Platzhalter ersetzt.

Regeln für gute Struktogramme

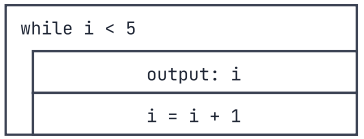
Jedes Element hat genau einen Eingang oben und einen Ausgang unten. Elemente werden nur **ineinander gesetzt**, nie mit Pfeilen verbunden. So entsteht kein Spaghetticode.

> Bedingungen als echten Vergleich schreiben: `zahl % 2 == 0` statt „zahl gerade?“. Dann kann die Simulation sie prüfen und der Code entsteht direkt.

Variablen mit sprechenden Namen, Zählvariable `i`, verschachtelt `j`.

> **Testen:** Simulation Schritt für Schritt oder Schreibtischtest mit einer Wertetabelle.

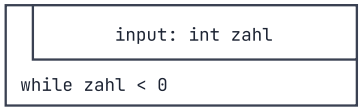
Kopfgesteuerte Schleife — while



Prüft die Bedingung **vor** jedem Durchlauf. Ist sie gleich zu Beginn false, läuft der Rumpf kein einziges Mal.

```
while (i < 5) { … }
```

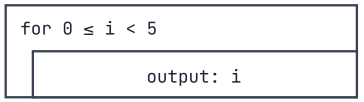
Fußgesteuerte Schleife — do-while



Führt den Rumpf zuerst aus und prüft **danach**. Läuft also mindestens einmal und wiederholt, solange die Bedingung true ist.

```
do { … } while (zahl < 0);
```

Zählschleife — for



Wiederholt eine feste Anzahl von Malen. Die Zählvariable `i` ändert sich bei jedem Durchlauf. Prüft vorher, ist also kopfgesteuert.

```
for (int i = 0; i < 5;
i++) { … }
```

Endlosschleife — endless



Hat keine Bedingung und hört nie von selbst auf, zum Beispiel beim Roboter, der immer weiterfährt. Verlassen nur mit Abbruch.

```
while (true) { … }
```

Abbruch — break



Verlässt die umgebende Schleife sofort. Es geht unter der Schleife weiter. Sparsam einsetzen.

```
break;
```