Folge 1 - Java und BlueJ

1.1
Informatiksysteme und Programmiersprachen

Lernziele

Wenn Sie diese Seite durchgearbeitet haben, sollten Sie wissen

- Was man unter einem **Informatiksystem** versteht,
- welche Aufgabe **Programmiersprachen** erfüllen, auch im Alltag
- was man unter einem **Programm** versteht,
- dass ein Programm mit einem **Kochrezept** vergleichbar ist,
- dass es neben linearen Programmen auch **verzweigte** Programme gibt.

Was ist ein Informatiksystem ?

Hier sehen wir uns einfach mal einen Ausschnitt aus dem Kernlehrplan des Landes NRW zum Thema an:

"Informatiksysteme sind heute weltweit miteinander vernetzt. Ein Informatiksystem ist eine spezifische Zusammenstellung von Hardware-, Software- und Netzwerkkomponenten zur Lösung eines Anwenderproblems. Gegenstand der Betrachtung ... ist schwerpunktmäßig der prinzipielle Aufbau singulärer und vernetzter Rechnersysteme und deren Interaktion untereinander und mit dem Benutzer."

Was ist eine Programmiersprache?

Mit Hilfe von **Programmiersprachen** können wir Handys, Tablets, Chips von Waschmaschinen, aber auch "richtige" Computer programmieren. Unter einem **Computerprogramm** versteht man eine Liste von **Anweisungen**, die das Endgerät ausführen soll. So ähnlich, wie ein **Rezept** eine Liste von Anweisungen ist, die der Koch ausführen soll.

Zutaten für 50 Plätzchen:

- 260 g Weizenmehl
- 1 Teelöffel Backpulver
- 90 g Zucker
- 2 Esslöffel Vanillezucker
- 130 g Pflanzenmargarine
- 2 Teelöffel abgeriebene Organgen-Schale
- 1 Hühnerei
- 50 g Kuvertüre zartbitter
- 50 g weiße Schokolade

Anleitung:

- Mehl, Backpulver, Zucker, Vanillezucker, Margarine, Orangenschale und Ei zu einem glatten Teig verarbeiten.
- Zartbitter-Kuvertüre hacken und unterkneten.
- Teig zu einer Rolle formen und in Folie gewickelt ca. 30 min. kalt stellen.
- Von der Rolle 50 ca. 5mm dicke Scheiben abschneiden und auf ein mit Backfolie ausgelegtes Backblech legen.
- Im vorgeheizten Backofen bei 180 Grad auf mittlerer Schiene ca. 15 – 20 min. goldgelb backen.
- Die weiße Schokolade im Wasserbad schmelzen.
- Die fertig gebackenen Plätzchen damit fadenartig überziehen.

Das Rezept informiert den Koch zunächst darüber, was er für die Herstellung von 50 Orangen-Plätzchen alles benötigt. Das Programm macht etwas Ähnliches, es sagt dem Computer, dass es Speicherplatz für drei reelle Zahlen benötigt. Innerhalb des Programms werden die drei Zahlen dann als **zahl1**, **zahl2** und **mittel** bezeichnet.

```
program Mittelwert;

var zahl1, zahl2, mittel : real;

begin
  read zahl1;
  read zahl2;
  mittel := (zahl1 + zahl2) / 2;
  write mittel;
end.
```

Das Rezept schreibt dann Schritt für Schritt vor, was der Koch machen soll. Das Programm macht genau das Gleiche. Zuerst soll ein Wert für die Variable **zahl1** eingelesen werden. Die Benutzerin des Programms wird dazu am Bildschirm aufgefordert, eine Ziffernfolge einzutippen. Die eingetippte Ziffernfolge, beispielsweise "123" wird dann als Zahl 123 in der Variable **zahl1** gespeichert.

Dann wird entsprechend die zweite Zahl eingelesen. Die Benutzerin des Programms gibt wieder eine Folge von Ziffern ein, zum Beispiel "543", und das Programm speichert die Zahl 543 in der Variable **zahl2**.

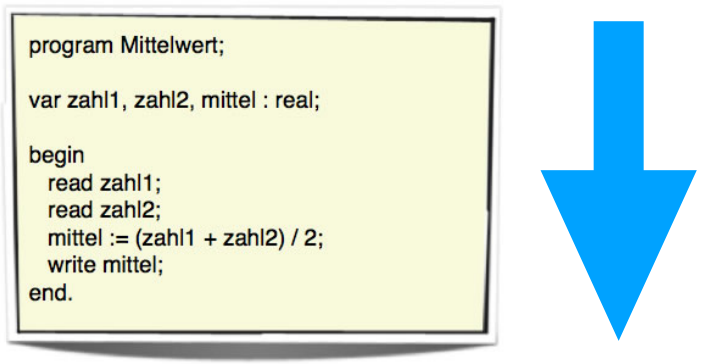
Dann addiert das Programm die Werte der beiden Variablen und dividiert das Ergebnis durch 2. Das Resultat dieser Berechnung wird dann in der Variable **mittel** gespeichert.

Der Befehl **write mittel** gibt schließlich die Zahl, die in der Variable **mittel** gespeichert ist, auf dem Bildschirm aus.

Dieses "Programm" ist so, wie es hier abgedruckt ist, nicht lauffähig. Es wurde in Anlehnung an die alte Computersprache **Pascal** geschrieben.

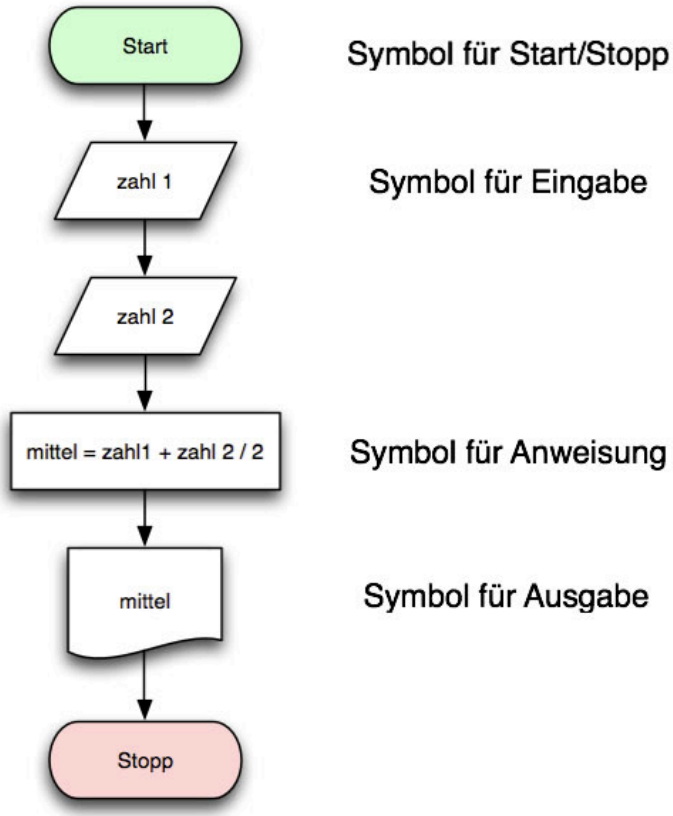
Lineare Programme

Ein Computerprogramm kann völlig **linear** verlaufen, so wie das Programm in unserem Beispiel. Graphisch könnte man den Ablauf eines solchen Programms durch einen großen Pfeil darstellen, der von oben nach unten geht:



Eine weitaus bessere Darstellung des Programmflusses erhält man durch ein **Flussdiagramm**. Hier wird der Ablauf des Programms, der so genannte **Programmfluss**, durch genormte Symbole dargestellt.

Flussdiagramm für das Programm



Dieses Flussdiagramm ist eine anschauliche graphische Darstellung des Programmflusses. Zunächst kommt das **Start-Symbol**, das Programm fängt hier also an.

Das Parallelogramm "zahl 1" ist ein **Eingabe-Symbol**. Eine Zahl soll jetzt eingelesen und in der Variable "zahl 1" gespeichert werden. Entsprechendes gilt für "zahl 2".

Das Rechteck "mittel = zahl 1 + zahl 2 / 2" ist ein **Anweisungs-Symbol**. Der aufmerksamen Betrachterin wird wohl auffallen, dass die hier abgedruckte Anweisung falsch ist. Wo steckt der Fehler? Kleiner Tipp: Sie kennen noch die alte Regel aus dem Mathe-Unterricht: "Punktrechnung geht vor Strichrechnung".

Das Symbol, das so aussieht wie ein abgerissenes Blatt Papier ist ein **Ausgabe-Symbol**. Der berechnete Mittelwert soll jetzt dem Benutzer ausgegeben oder angezeigt werden.

Ganz am Ende kommt ein **Stopp-Symbol**, das wieder genau das besagt, was es heißt, nämlich dass das Programm jetzt stoppt. In der Fachsprache würde man jetzt sagen: Das Programm **terminiert**.

Verzweigte Programme

Es gibt aber auch Computerprogramme, die sogenannte **Fallunterscheidungen** treffen können. Stellen wir uns ein einfaches Programm vor, dass den Wert eines Quotienten zweier Zahlen berechnen soll.

Zunächst werden wieder die erforderlichen Variablen deklariert (**zahl1**, **zahl2**, **ergebnis**), und dann kommen die Anweisungen des Programms. Die beiden Zahlen werden eingelesen, dann wird die erste Zahl durch die zweite dividiert und das Ergebnis gespeichert und anschließend ausgegeben.

Das Programm funktioniert auch wunderbar, bis jemand auf die Idee kommt, als **zahl2** eine Null einzutippen. Da man nicht durch Null dividieren kann, stürzt das Programm mit einem sogenannten **Laufzeitfehler** ab. Wenn man Pech hat, kann man anschließend den Rechner, das Handy, die Smartwatch oder was immer man auch gerade benutzt neu starten.

Wenn man also ein solches Programm schreibt, ist es ratsam, eine **Fallunterscheidung** einzubauen.

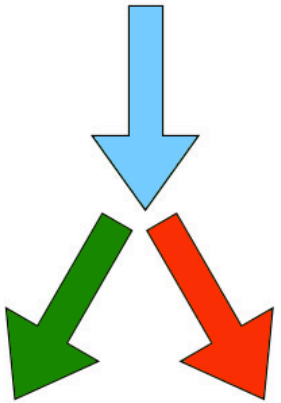
Fallunterscheidungen

In der Programmiersprache Pascal sähe eine solche Fallunterscheidung so aus:

```
program Teile;

var zahl1, zahl2, ergebnis : real;

begin
  read zahl1;
  read zahl2;
  if zahl2 <> 0
    then
      ergebnis = zahl1/zahl2;
      write ergebnis;
    else
      write "Man darf nicht durch 0 teilen!";
end.
```



Jetzt sollten Sie eine ungefähre Vorstellung davon haben, was der Unterschied zwischen Kochen / Backen einerseits und Programmieren andererseits ist.

Ausblick

Im Laufe dieses Kurses werden wir uns hauptsächlich mit der Programmiersprache **Java** beschäftigen, ohne die es heute wahrscheinlich keine Handys, Tablets und Smartwatches geben würde. Allerdings ist dieser Kurs für die Arbeit an "richtigen" Computern konzipiert. Eingesetzt wird dabei die Entwicklungsumgebung **BlueJ**, die eigens für den Anfängerunterricht an höheren Schulen und Universitäten entwickelt worden ist.

Aufgaben / Übungen

Aufgabe 1.1-1

Erstellen Sie eine Tabelle mit mindestens fünf "Informatiksystemen", die auch im Alltag benutzt werden. Listen Sie die Funktion dieser Systeme auf und geben Sie an, ob diese Systeme vernetzt sind oder "stand alone" (singulär, für sich, ohne Vernetzung) arbeiten können.

Aufgabe 1.1-2

Recherchieren Sie zu einem der folgenden Themen:

- Eingabegeräte
- Ausgabegeräte
- Externe Speichermedien
- Geschichte der Computer
- Aufbau eines Computers
- World Wide Web (WWW)
- Computer und Arbeitswelt
- Gegenüberstellung analog/digital an ausgewählten Beispielen
- Homeoffice früher und heute

und stellen Sie Ihre Recherche-Ergebnisse dem Kurs vor, indem Sie eine kleine Präsentation dazu anfertigen.

Aufgabe 1.1-3

Was macht folgendes Programm?

```
program geheim;
var z1, z2, z3, ergebnis : integer;

begin
   read z1;
   read z2;
   read z3;
   if z1 < z2 and z1 < z3
      then ergebnis = z1;
   else if z2 < z1 and z2 < z3
      then ergebnis = z2;
   else if z3 < z1 and z3 < z2
      then ergebnis = z3;
   write ergebnis;
end.
```

Beschreiben Sie, welches Ergebnis das Programm liefert und überlegen Sie, wie man das Programm noch optimieren könnte. Erläutern Sie diese Optimierung!