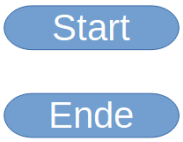
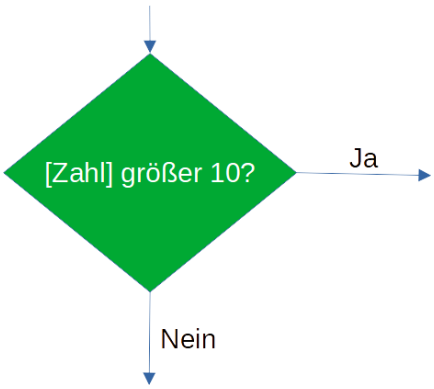
Wozu braucht man Flussdiagramme?

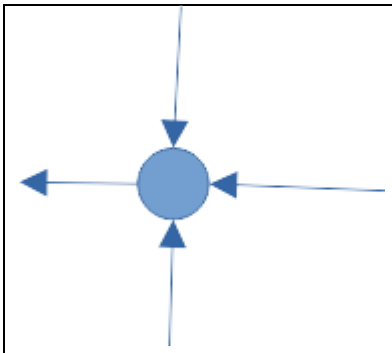
Programme werden oft sehr lang, und dann kann es schwierig sein, auf Anhieb zu erkennen, was ein Programm genau macht. Hier helfen Flussdiagramme. Das sind Zeichnungen, auf denen man den Aufbau eines Programms sehr gut erkennen kann.

Flussdiagramme sind auch sehr gute "Baupläne" für richtige Programme. Da sie nicht von einer bestimmten Programmiersprache abhängen, kann man sie für beliebige Programmierumgebungen verwenden.

Einige wichtige Symbole für Flussdiagramme

Hier einige Symbole, die in Flussdiagrammen verwendet werden. Es gibt noch viel mehr davon, aber die hier genügen uns erst einmal für einfache Programme.

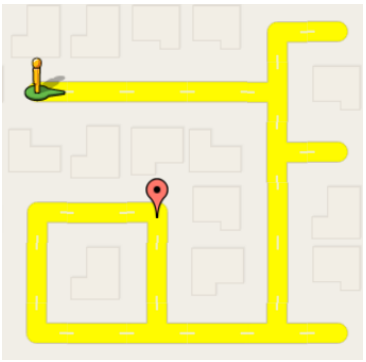
	Programmfluss-Pfeil Ein Pfeil zeigt an, wo das Programm entlang gehen soll. Man kann nie gegen die Pfeilrichtung gehen.
	**Start oder Ende** An diesen Blöcken beginnt oder endet ein Programm. Ein Programm hat meistens genau einen Start- und einen Ende Block.
	Anweisung Diese Blöcke geben Befehle: "dreh dich", "gehe vorwärts", "zähle eins dazu", und so weiter. Der Befehl steht als Text im Block.
	**Entscheidung** In diesen Blöcken stellt das Programm eine Ja/Nein Frage. Wenn die Antwort "Ja" ist, geht das Programm in eine Richtung weiter, wenn sie "nein" ist, in eine andere. Pfeile beginnen und enden immer an den Spitzen des grünen Parallelogramms.

	**Verbindung, "Knoten"** An diesen Punkten können getrennte Wege wieder zusammenkommen. Es können beliebig viele Pfeile hineinlaufen, aber nur einer kommt wieder heraus. Dort geht das Programm weiter.

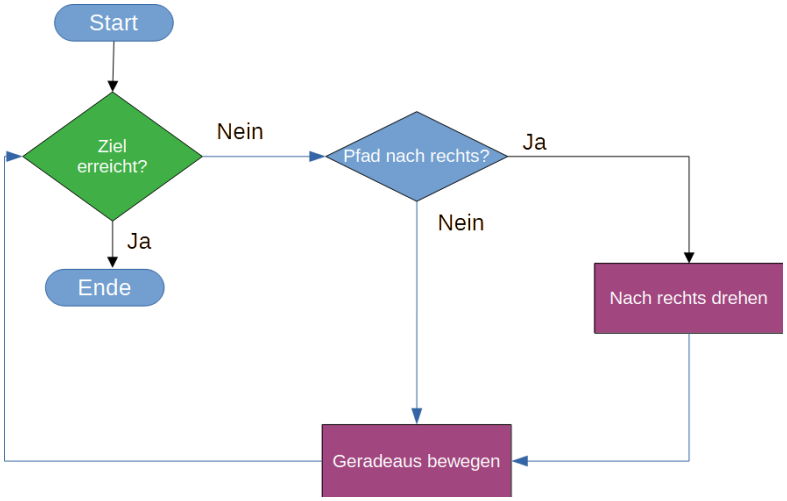
Flussdiagramme sind allgemeine Anleitungen, wie man ein bestimmtes Problem lösen kann. Sie sind nicht abhängig von einer bestimmten Programmiersprache.

Ein einfaches Flussdiagramm

In der Programmier-Spielecke "Blockly" gibt es als Aufgabe folgendes Labyrinth zu lösen:



Die Aufgabe besteht darin, ein Programm zu entwickeln, das die Spielfigur zum Zielpunkt laufen lässt. Folgendes Flussdiagramm kann helfen, das Programm zu verstehen:



Probiere das Flussdiagramm im Kopf oder auf einem Blatt Papier aus, und denke darüber nach, ob es ein "gutes" Programm ist. Erreicht die Spielfigur das Ziel? Funktioniert es auch dann, wenn du die Spielfigur wo anders hinstellst, oder wenn du das Ziel irgendwo anders machst?

Das Beispiel stammt von der Webseite "blockly.games" (Labyrinth Level 7).

In Blockly "programmiert" man, indem man farbige Blöcke zusammensteckt. So schaut das Blockly-Programm, das man aus dem Flussdiagramm entwickelt, aus:

Wenn du im Kopf darüber nachdenkst, was die Blöcke mit der Spielfigur tun, siehst du, dass es genau das selbe ist wie im Flussdiagramm. Ich finde aber, dass das Flussdiagramm wesentlich einfacher zu lesen ist.

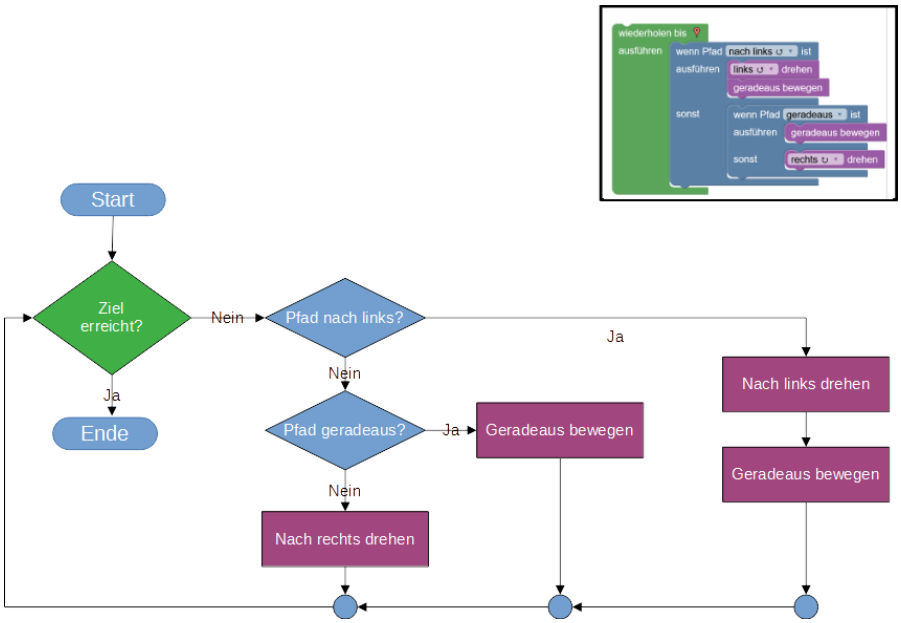
Und in einer "richtigen" Programmiersprache, z.B. "Javascript", schaut das Programm so aus:

```
while (notDone()) {
  if (isPathRight()) {
    turnRight();
  }
  moveForward();
}
```

Das Programm ist nicht sehr gut. Es gibt viele Orte im Labyrinth, wo die Spielfigur das Ziel nicht finden würde. Wenn du ein richtig gutes Programm sehen möchtest, wo du tatsächlich die Spielfigur und das Ziel an eine beliebige Stelle setzen kannst, schau dir doch das folgende Beispiel an.

Ein komplizierteres Flussdiagramm

Hier kommt ein Beispiel für ein etwas umfangreicheres Flussdiagramm. Es beschreibt ein einfaches Programm, das eine Figur durch ein beliebiges Labyrinth führen kann. Du kannst es gerne ausprobieren auf https://blockly.games/maze?lang=de&level=10&&skin=0

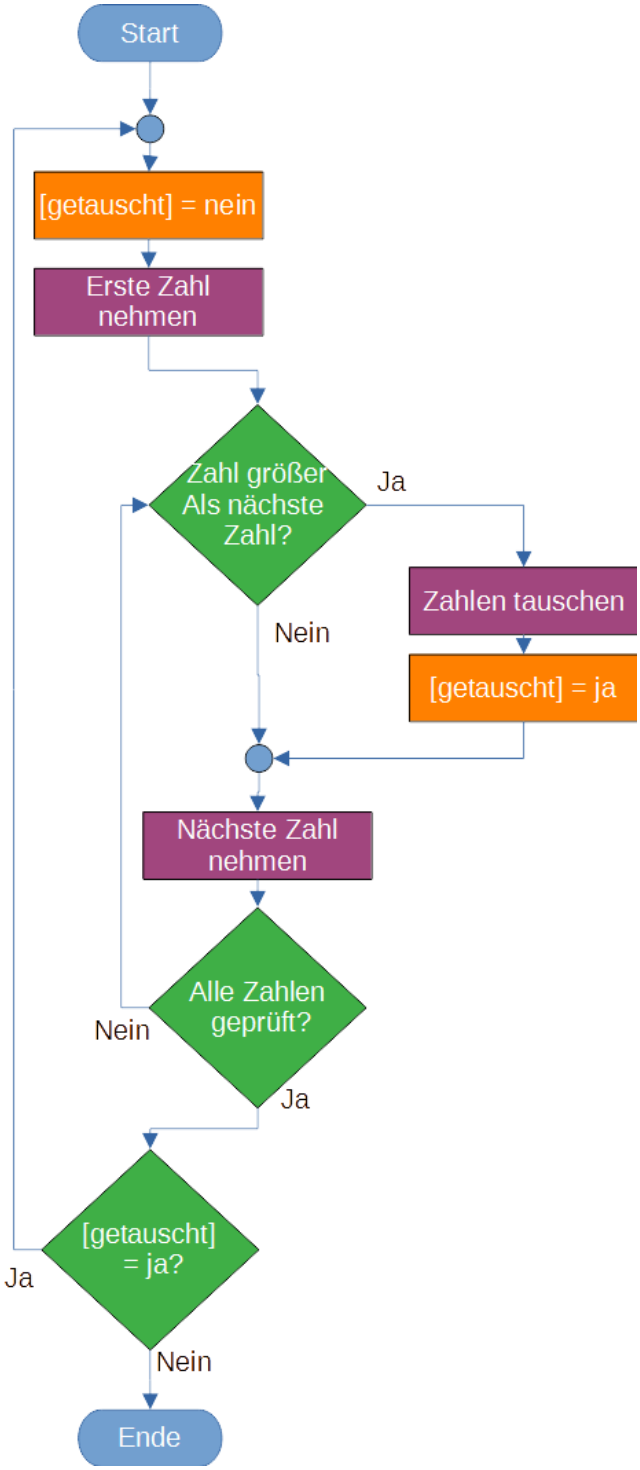


Rechts oben siehst du die Lösung für Blockly.

Man kann Flussdiagramme auch gut verwenden, um Programme "im Kopf" zu testen. Oft nimmt man dabei auch noch Papier und Bleistift zu Hilfe. Im Flussdiagramm ist es viel einfacher, dem Programm Schritt für Schritt zu folgen.

Übung

Versuche herauszufinden, was folgendes Flussdiagramm macht, wenn man eine Liste von zufälligen Zahlen hindurchschickt.



Und am Ende noch eine Frage für zukünftige Programmierer:

kann man dieses einfache Programm noch "optimieren"? "Optimieren" bedeutet, dass man vielleicht noch eine bessere Lösung finden könnte. Eine die schneller läuft beispielsweise, oder eine die weniger Blöcke braucht. Hinweis: ja, man kann es wesentlich schneller machen, und außerdem würde es in einer bestimmten Situation überhaupt nicht funktionieren. Findest du sie?