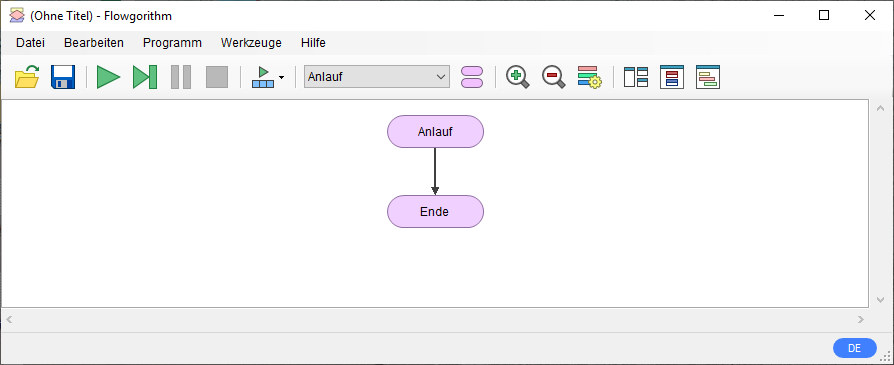
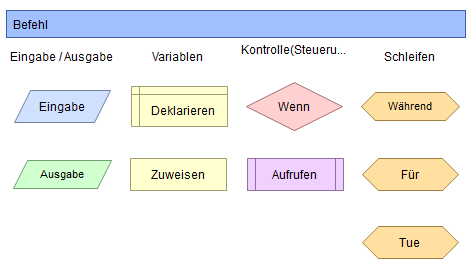
# Programmieren lernen mit der Simulation FLOWGORITHM



**Flowgorithm** ( <http://www.flowgorithm.org/download/> ) ist ein nettes freies Werkzeug, um Anfängern die Grundlagen des Programmierens näherzubringen, da es erlaubt, einfache Abläufe als **Flussdiagramme** zu beschreiben. Man findet sich schnell mit den elementaren Bausteinen für Ein- und Ausgabe zurecht, erkennt schnell die Notwendigkeit des Speicherns von Informationen in Variablen, und lernt, die Kontrolle über den Programmablauf über Verzweigungen und Schleifen übersichtlich darzustellen. Komplexere Teilaufgaben können schließlich als eigene Unterprogramme ausgeführt werden.

Ein eigenes Fenster zeigt bei Bedarf die Veränderungen im Speicher eines Programmes an, was besonders beim Simulieren im Schritt-für-Schritt Modus lehrreich ist!

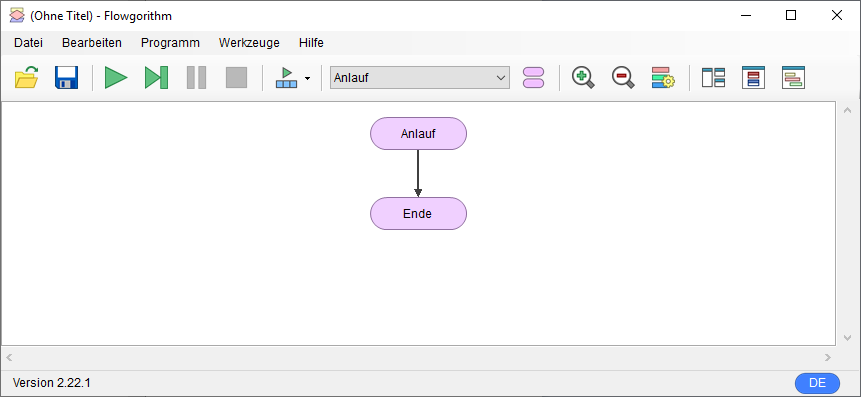
Einmal heruntergeladen und installiert, kann das Programm auch einfach auf einen USB-Stick kopiert und somit mitgenommen werden, um portabel verwendet zu werden.

Wer einen erfolgreichen Einstieg geschafft hat, kann einige fertige Musterprogramme von deren Homepage herunterladen.

# Das erste Programm

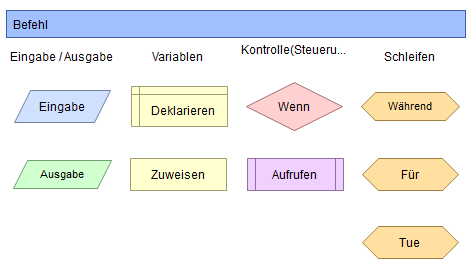
Wohl seit der der Erfindung der frühesten Programmiersprachen gibt es ein Musterbeispiel, für welche das erste Programm geschrieben wird: die Ausgabe von „Hallo, Welt“. Diese Herausforderung soll unser Einstieg sein!

Nach dem Start von **Flowgorithm** finden wir folgenden Startbildschirm vor:

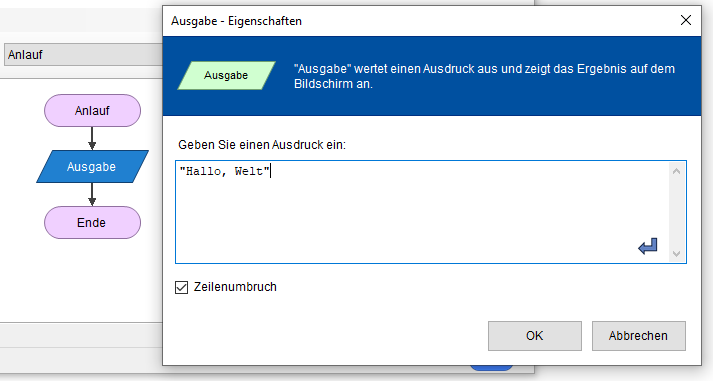
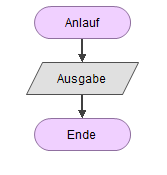


Jedes Programm besitzt sowohl einen definierten Anfang (hier als „Anlauf“ bezeichnet), also auch ein definiertes Ende. Über den blauen „Speichern“ Button können wir unsere Arbeit für später bei Bedarf speichern. Wenn wir das noch nicht gemacht haben, machen wir das nun! Dazu wählen wir einen möglichst klingenden Namen, z.B. **HalloWelt**

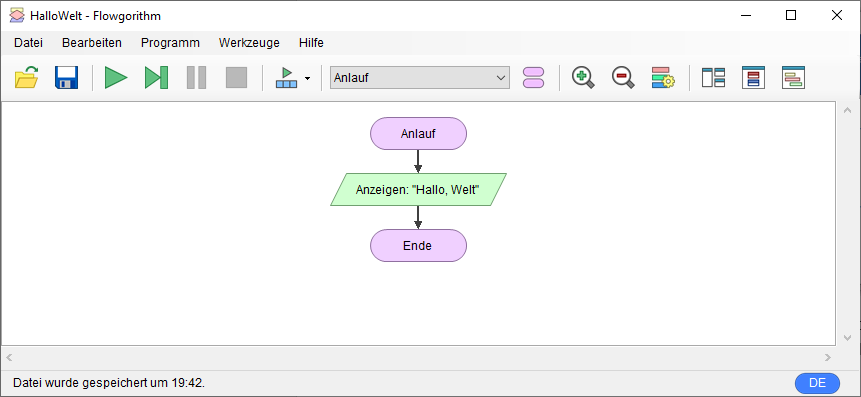
Als nächsten Schritt klicken wir mit der rechten Maustaste auf jenen Pfeil, der vom Anlauf zum Ende führt. Wir sehen nun eine Menge an Objekten, die eingefügt werden können:

****

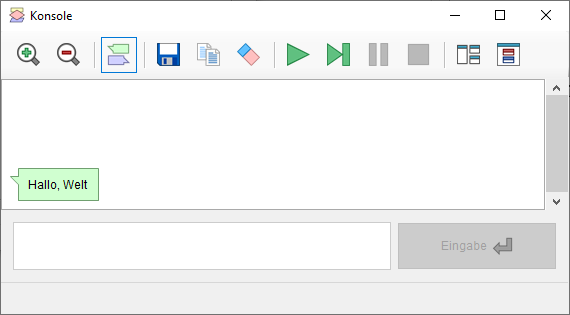
Ein einfacher Klick auf das grüne Ausgabe-Symbol erzeugt nun eine vorerst graue Ausgabe-Box. Nachdem wir einen Doppelklick darauf machen, können wir festlegen, was ausgegeben werden soll.



Wir haben vorhin besprochen, **Hallo Welt** einzugeben: achten wir dabei, nicht die doppelten Hochkommas am Beginn und am Ende zu vergessen. Ein Klick auf OK zeigt uns nun das fertige Programm:



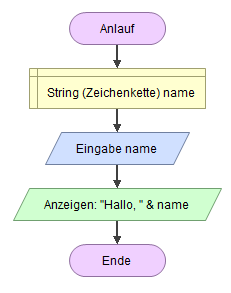
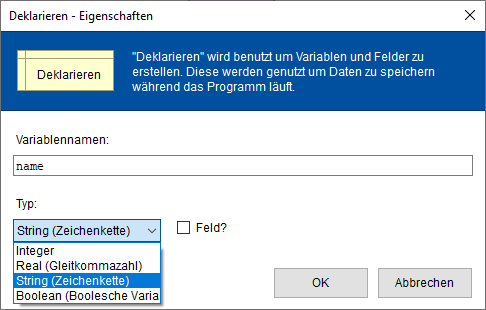
Nun wollen wir das Programm „in Action“ sehen: wir bewegen die Maus vorsichtig zum linken grünen Pfeil und sehen, dass wir damit das Programm ausführen können. Es öffnet sich ein Konsolenfenster, welches in einer (leider eckigen) grünen Sprechblase **Hallo Welt** darstellt:



# Eingabe und Ausgabe von Werten

Wir wollen nun unser erstes „echtes Programm“ schreiben – ein Programm, welches einen Nutzer nach dem Namen fragt, und letztendlich diesen begrüßt!

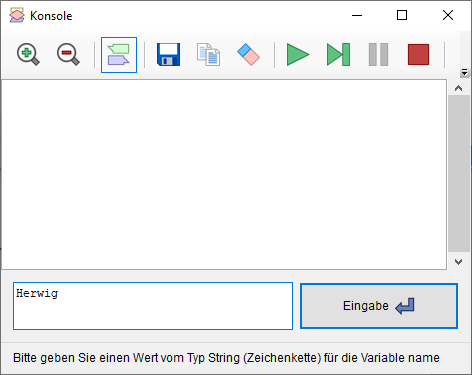
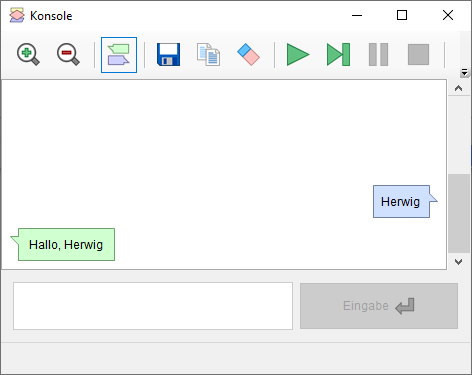
Dazu brauchen wir 2 neue „Befehle“: einen zum Speichern, und einen weiteren zum Eingeben des Namens. Das Festlegen eines Speichers wird im Übrigen „Deklarieren“ genannt!

Wir fügen die entsprechenden Symbole in ein (neues) Flowgorithm Programm ein. Beim Deklarieren des Namens (Doppelklick auf „Deklarieren“) müssen wir noch festlegen, welche Art von Speicher verwendet werden soll. Es werden derzeit 4 Arten von Speicher angeboten: **Ganze Zahlen** (Integer), **Kommazahlen** (Real), **Zeichenketten** (String) und **Wahrheitswerte** (Boolean).

Zur Ausgabe verbinden wir den Text „Hallo“ mit der Variablen *name* über ein kaufmännisches *&-*Symbol.

Wenn wir nun über das grüne Dreieck den Programmablauf starten, geht das Konsolen Fenster auf:

Zur Hilfestellung sehen wir unter einem Eingabefeld „Bitte geben Sie…“.

Nach Eingabe des Namens und abschließendem Drücken der Enter-Taste sehen wir rechts eine blaue Sprechblase mit der getätigten Eingabe, und links, wie gewohnt, die Ausgabe … das war’s!

Wir speichern unser Programm, unter einem neuen Namen.

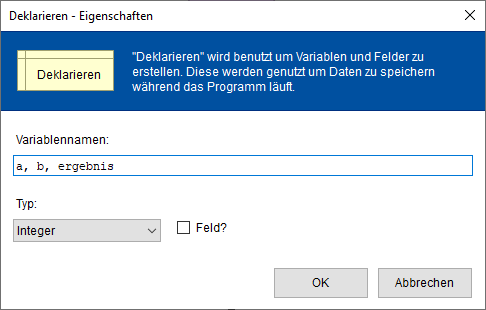
Zur Übung versuchen wir, nun, die Ausgabe zu löschen (einfach draufklicken und die Entfernen Taste drücken)! Danach markieren wir mit der Maus einen Bereich, der knapp größer als die Deklaration und die Eingabe sind… nach einem Klick mit der rechten Maustaste können wir nun beide Symbole auf einmal löschen. **Bitte** **NICHT SPEICHERN** 😊 aber trotzdem das Programm schließen!

Sehr spektakulär war dieses erste Programm noch nicht… steigern wir den Schwierigkeitsgrad etwas.

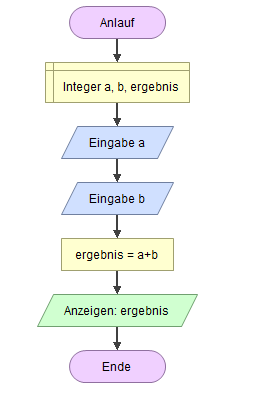
# Das EVA-Prinzip – **E**ingabe, **V**erarbeitung und **A**usgabe

Wer kurz nach „**eva prinzip**“ googeln will – gerne! Fazit: Jedes einigermaßen sinnvolle Programm braucht eine oder mehrere Eingaben, es verarbeitet Daten und gibt sie letztendlich wieder etwas aus.

Wir wollen nun ein einfaches neues Programm erstellen, welches zwei Zahlen einliest, diese beiden addiert und das Ergebnis ausgibt. Klingt nicht wirklich schwierig!

Welchen Speicher (welche „**Variablen**“) brauchen wir dafür? Genau: 2 Zahlen zur Eingabe und eine zum Berechnen des Ergebnisses. Das können wir sogar in einem Schritt erledigen, in dem wir oben ein Symbol „Deklaration“ einfügen, und 3 Variablen, mit Beistrich getrennt, eingeben.

Das funktioniert nur, weil alle 3 gleichen Datentyps sind, nämlich **Integer**.

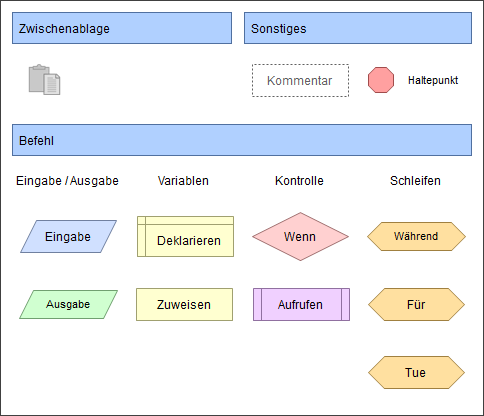
Das fertige Programm sieht letztendlich (hoffentlich) folgendermaßen aus:

Die Eingabe der Zahl a und der Zahl b erfolgt in eigenen Schritten.

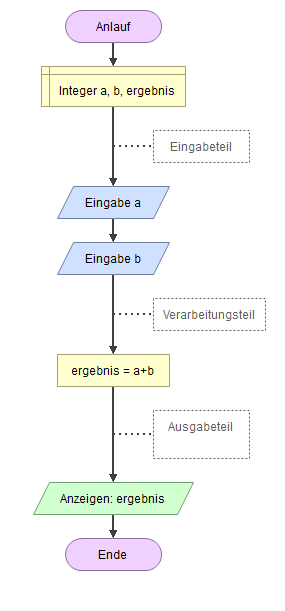
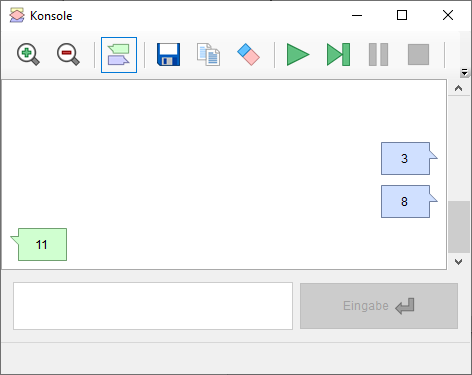
Die Verarbeitung besteht in der Berechnung des Ergebnisses

Die Ausgabe ist der letzte Schritt.

# Programme kommentieren

Wenn Programme umfangreicher werden, kann es nicht schaden, Anmerkungen anzubringen, die den Ablauf nicht beeinflussen, jedoch dabei helfen, uns später wieder zurechtzufinden. Wenn man mit der rechten Maus ein Symbol einfügt, gibt es rechts oben unter „Sonstiges“ den Kommentar. Mit Doppelklick kann dieser entsprechend mit einem Text befüllt werden!

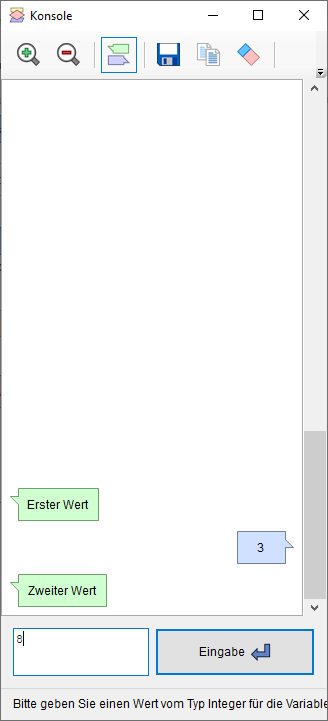
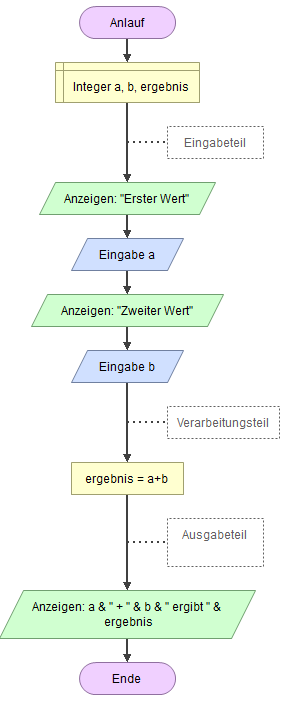
Wir kommentieren nun unser Programm,   
um das EVA-Prinzip zu zeigen:

 3 + 8 = 11

# Das EVA-Prinzip in der Realität

Unser Programm zeigt noch Mängel in der praktischen Anwendbarkeit: Den Hinweis zur Eingabe muss man einmal finden ☹

Besser wäre es, wenn unser Programm eine aktive Benutzerführung hätte. Dazu machen wir vor jede Eingabe eine Ausgabe!

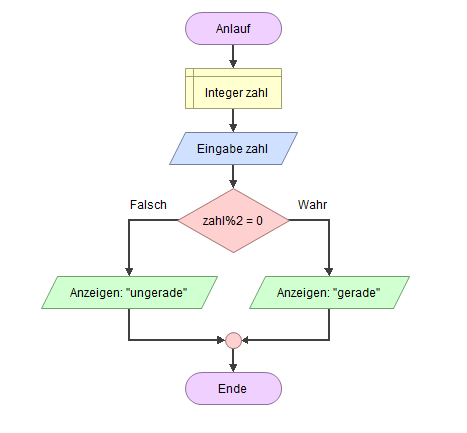


Und wieder ist das Ergebnis besser geworden:

# Den Programmfluss steuern

Dass Programme geradlinig vom Anfang bis zum Ende ablaufen, ist natürlich in der Praxis kaum der Fall. Oft treten Situationen auf, in denen einmal dies, ein anderes Mal etwas anderes durchlaufen werden soll. Somit brauchen wir einen Mechanismus zur Kontrolle des Ablaufes.

Wir stellen uns die Aufgabe, dass ein Programm erkennt, ob eine gerade oder eine ungerade Zahl eingegeben worden ist.

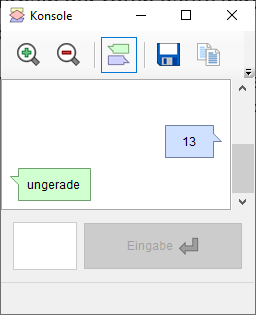
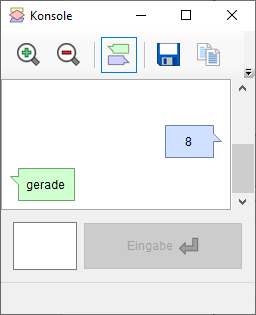
Die Kontrolle der Verzweigung in die Fälle „gerade“ und „ungerade“ übernimmt dazu in Flowgorithm der „WENN“ Befehl. Man wählt dazu die rötliche Raute aus dem Werkzeugkasten und konfiguriert sie entsprechend. Nun existieren 2 Wege – nach rechts „wahr“, nach links „falsch“.

Zur Überprüfung, ob eine Zahl gerade ist, wir der sogenannte Restwertoperator verwendet. Neben dem bereits bekannten Rechenoperator + gibt es auch noch - \* / (Division) und eben % (den Divisionsrest).

Nachhilfe zum Thema Dividieren/Divisionsrest gibt’s beim Mathe-Lehrer 😉

Innerhalb der roten Raute wird die Rechnung *zahl % 2* durchgeführt. Dabei wird *zahl* durch 2 dividiert und der Divisionsrest betrachtet. Dieser Divisionsrest kann nur 2 Werte annehmen: 0 und 1.

Danach wird verglichen, ob dieser berechnete Wert gleich 0 ist. Wenn ja, wird der „Wahr“-Abschnitt durchlaufen. Ansonsten der „Falsch“-Abschnitt.

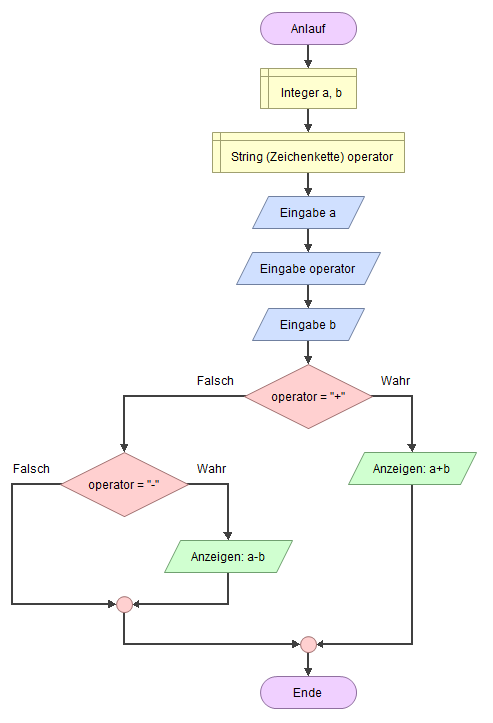
 

# Mehrere Verzweigungen

Eine einfache Verzweigung, wie im vorigen Beispiel, wird auch nur selten ausreichen, um komplexe Aufgaben zu lösen.

Wir wollen in diesem Abschnitt das Programm zum Addieren von Zahlen zu einem Taschenrechner erweitern, der gängige Rechenoperationen beherrscht.

Anmerkung: bei den meisten nun folgenden Programmen wird auf eine elegante Benutzerführung bewusst verzichtet, damit die abgedruckten Diagramme einfach verständlich bleiben, und auf keinen Fall überladen wirken. Wer diese Aufgaben selber nachprogrammiert, soll selbstverständlich eine möglichst elegante Benutzerführung in die Ablaufdiagramme einbauen!

Zur Umsetzung:

Die Idee ist einfach, zusätzlich einen Rechenoperator einzugeben. Ist dieser „+“, dann wird eine Addition durchgeführt. Wenn nicht, wird überprüft, ob ein „-“ vorliegt, dann kann eine Subtraktion durchgeführt werden.

Wie geht’s dann weiter?

Ganz eifrige bauen gar noch Multiplikation und Division ein! Oder überlegen, bei der Division sowohl das Ergebnis als auch den Rest zu berechnen!

Man sieht, dass die Anzeige so flexibel ist, eine Rechnung durchzuführen und das Ergebnis anzuzeigen. Man spart dadurch eine Variable (ergebnis) ein.

Außerdem erfolgen Verarbeitung (Berechnung) und Ausgabe hier sogar in einem Schritt.

Soviel zu Theorie und Praxis in Bezug auf das   
EVA-Prinzip 😉

Sinnvoll wäre auch noch eine Strategie, bei Eingabe einer ungültigen Rechenoperation eine Fehlermeldung auszugeben…

# Komplexere Verzweigungen

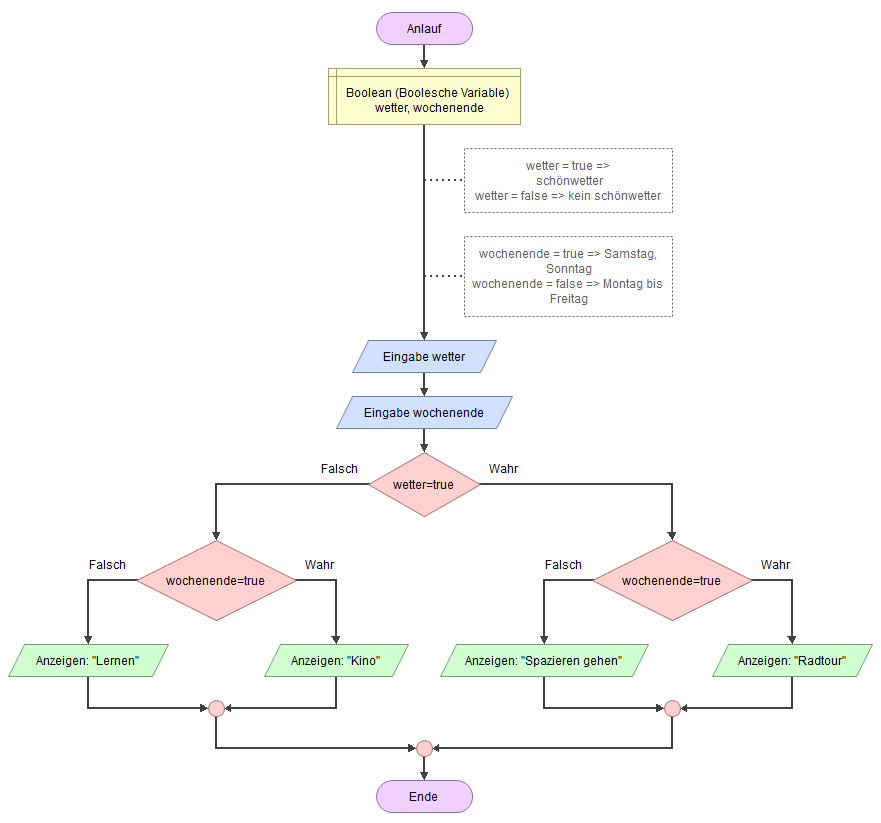
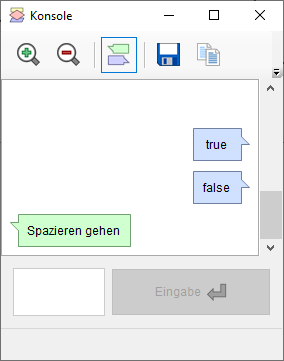
Stellen wir uns folgendes vor: Abhängig vom Wetter und der Tatsache, ob es Wochenende ist, wollen wir verschiedene Aktionen durchführen:

Ist es schön am Wochenende, dann machen wir eine Radtour  
Ist es nicht schön am Wochenende, dann gehen wir ins Kino  
Ist es schön unter der Woche, dann gehen wir (kurz) spazieren  
Ist es nicht schön unter der Woche, dann lernen wir für die Schularbeit nächste Woche

Wir stellen das tabellarisch dar:

|  |  |  |
| --- | --- | --- |
| Wetter schön / Wochenende | Ja | nein |
| Ja | Radtour machen | Kurz spazieren gehen |
| Nein | Kinobesuch | Für die Schularbeit lernen |

So eine Situation kann man den **Variablen** für **Wahrheitswerte** elegant abbilden – da braucht man nur zwei: Eine für das Wetter, eine für das Wochenende!

Hinweis: Bitte um Benutzerführung ergänzen