Bedingte Anweisung (if)

Description

Eines der wichtigsten Elemente der Programmierung ist die bedingte Anweisung. Hierzu muss eine Bedingung überprüft und die Überprüfung bestehen, bevor eine Anweisung ausgeführt wird. Dieses Einsteiger-Tutorial erklärt, wie das geht.

Kaum ein Programm würde funktionieren, wenn es nicht in der Lage wäre, Bedingungen zu prüfen und anhand des Ergebnisses eine Anweisung auszuführen. Die if-Anweisung gehört zur elementarsten Form und sollte von jedem angehenden Programmierer beherrscht werden.

Was macht if?

Mit if prüfen wir, allgemein gesagt, eine Gegebenheit. Wenn diese unseren Vorgaben entspricht, oder nicht entspricht, folgt eine Anweisung. Das nennt man in der Programmierung eine Bedingte Anweisung.

Aufbau

Vereinfacht dargestellt, funktioniert eine if-Abfrage so:

```
wenn Bedingung dann
     Anweisungen
```

Ein konkretes Beispiel würde so aussehen:

```
if (variable1 > variable2)
{
     variable1 = variable2;
}
```

In diesem Beispiel prüfen wir, ob die erste Variable größer ist als die zweite. Wenn dem so ist, wird die

erste Variable auf den Wert der zweiten Variable gestellt. Wenn also Variabl1 den Wert 2 hat und Variable2 den Wert 1, wird aus der Variable1 eine 1.

Bitte beachte, dass die Bedingung in runden Klammern geprüft wird, die Anweisung sich aber in geschweiften Klammern befindet.

Wenn-Dann-Sonst

Das obere Beispiel sagt nur aus: „Wenn eine Bedingung erfüllt ist, dann mache folgendes…". Nun wollen wir aber eine Anweisung, die auch beinhaltet, was passieren soll, wenn die Bedingung nicht erfüllt ist. Dazu brauchen wir das *else*, was „sonst" bedeutet.

```
if (variable1 > variable2)
{
     variable1 = variable2;
} else {
     variable2 = variable1;
}
```

Der Anfang bleibt gleich. Wenn die Variable1 größer ist als Variable2, nimmt die Variable1 den Wert von Variable2 an. Dann kommt die Neuerung! Wenn die Bedingung nicht erfüllt wurde, dann wird nimmt die Variable2 den Wert von Variable1 an.

Damit kann man schon sehr viel anfangen. Nehmen wir an, wir haben ein Spiel für zwei Spieler. Nun können wir prüfen, ob Spieler 1 mehr Punkte hat als Spieler 2. Wenn ja, hat Spieler 1 gewonnen, wenn nein, hat Spieler 2 gewonnen, sofern kein Unentschieden möglich ist.

Doch was ist, wenn wir drei Möglichkeiten haben? Dazu gibt es Elfe if.

Else if / elseif

Gleich mal vorab: Manche Programmiersprachen wie GML verwenden else if, andere, wie PHP, elseif. Im Beispiel werde ich es getrennt schreiben.

```
if (scorePlayer1 > scorePlayer2)
{
     winner = 1;

} else if (scorePlayer2 > scorePlayer1) {
     winner = 2;
} else {
     winner = 3;
}
```

Was haben wir hier gemacht? Wir prüfen, ob die Punktzahl von Player1 größer ist als die von Player2. Wenn ja, bekommt die Variable *winner* den Wert 1. Wenn dem nicht so ist, wird geprüft, ob Player2 mehr Punkte hat als Player1. Wenn das zutrifft, bekommt die Variable *winner* den Wert 2. Trifft das auch nicht zu, bleibt nur die Möglichkeit, dass die Werte gleich sind. Dann bekommt die Variable *winner* den Wert 3 und es ist unentschieden. 0 habe ich in dem Fall dafür reserviert, dass noch kein Gewinner

feststeht.

Wahr oder falsch?

Damit kann man schon viele Sachen abfragen. Sehr oft müssen wir nur wissen, ob eine Variable wahr ist, oder falsch. Dazu gibt es mehrere Möglichkeiten:

```
if (variable == 1)… // Für wahr
if (variable == 0)… // Für falsch
```

Das == ist eine Boolesche Variable, die nur wahr oder falsch sein kann. In einigen Programmiersprachen ist auch das = erlaubt, aber da weiß man als Leser des Codes nicht auf Anhieb, dass es eine Boolesche Variable sein soll. 1 oder 0 könnten auch beliebige Zahlen sein. Deshalb kann man in einigen Programmiersprachen, wie GML, wie folgt abfragen:

```
if (variable = true)… // Für wahr
if (variable = false)… // Für falsch
```

Da gibt es ebenfalls keine Missverständnisse. Das Ganze kann man aber auch verkürzt darstellen:

```
if (variable)… // Für wahr
if (!variable)… // Für falsch
```

Die Variable in Klammern wird hier nur auf ihre Richtigkeit geprüft. Das Ausrufezeichen davor bedeutet so viel wie „ist nicht”, in dem Zusammenhang also „ist nicht wahr”. Dazu kommen wir gleich noch einmal.

Verschiedene Ausdrücke

Mit dem bisher gezeigten kann man schon einiges anfangen, aber das reicht noch nicht. Wenn man nur dann eine Anweisung ausführen will, wenn eine Bedingung nicht zutrifft, wird es schon schwierig. Ebenso, wenn mehrere Bedingungen zutreffen sollen. Das kann man zwar mit ifs verschachteln, aber irgendwann kann das kein Mensch mehr lesen. Um solche und andere Probleme zu lösen, gibt es verschieden Ausdrücke, die man verwenden kann. Die folgenden Ausdrücke sind nicht nur für if-Anweisungen gedacht, können aber von Programmiersprache zu Programmiersprache unterschiedlich ausfallen.

Ausdruck	Bedeutung	Beispiel
=	ist gleich	Variable = 2
!	ist nicht	!Variable

Ausdruck	Bedeutung	Beispiel
==	Boolesche Variable	Variable == true
&&	und	(Variable1 = 1) && (Variable2 = 2)
||	oder	(Variable1 = 1) || (Variable2 = 2)
^^	xor / Exklusiv-ODER	(Variable1 = 1) ^^ (Variable2 = 2)
<	kleiner	Variable1 < Variable2
<=	kleiner oder gleich	Variable1 <= Variable2
>	größer	Variable1 > Variable2
>=	größer oder gleich	Variable1 >= Variable2
!=	ungleich	Variable1 != Variable2
+	plus	Variable + 2
–	minus	Variable – 2
*	mal	Variable * 2
/	geteilt	Variable / 2
++	addiert einen Wert hinzu	i++
—	subtrahiert einen Wert ab	i–

Ausdruck	Bedeutung	Beispiel
+=	Wert addieren	Variable += 2
-=	Wert subtrahieren	Variable -= 2
div	Ganzzahldivision	Variable div 2
mod	zieht den Restwert multipliziert mit dem Teiler vom Anfangswert ab	Variable mod 2

Damit kann man schon sehr viel machen, nicht nur bei if-Abfragen.

Beispiele für Kombinationen

```
if (variable1) && (variable2)
// Heißt: Wenn Variable1 und Variable2 wahr sind. Hier müssen beide Bedingunge

if (variable1) || (variable2)
// Heißt: Wenn Variable1 oder Variable2 wahr ist. Hier muss nur eine Bedingung

if (variable1) && (variable2 >= 100)
// Heißt: Wenn Variable1 wahr ist und der Wert von Variable2 100 oder größer a

if (!variable1) && (variable2) && (variable3 != 10)
//Heißt: Wenn Variable1 falsch ist, Variable2 wahr ist und Variable3 ungleich
```

Das letzte Beispiel wirkt für Anfänger vielleicht etwas an den Haaren herbeigezogen, doch kommen solche Fälle immer wieder vor. Variable1 kann Gegner sein, der nicht mehr existieren darf. Variable2 kann der Spieler sein, der noch lebt und Variable3 können Lebenspunkte sein, die nur beim Levelstart auf 10 stehen. Nur in dieser Kombination hat der Spieler das Level geschafft. Wie gesagt, solche Kombinationen sind nicht ungewöhnlich.

Es gibt natürlich noch mehr Kombinationsmöglichkeiten, aber diese sollten sich anhand der Tabelle und den Beispielen ableiten lassen. Wie alles bei der Programmierung braucht auch der korrekte, effiziente Einsatz von if-Abfragen, etwas Übung. Manchmal hilft es, das Problem erst einmal als Klartext zu formulieren um dann die korrekte Abfrage zu erstellen. Als **Hausaufgabe** könnte man aus folgendem Text eine Abfrage mit Anweisungen basteln. Wer die Lösung hat, darf sie gerne in die Kommentare schreiben.

“Wenn Wert1 größer 0 ist und Wert2 wahr, dann soll die Variable *i* um 1 erhöht werden. Wenn Wert1 0 oder kleiner ist und Wert2 wahr, dann soll *i* um zwei erhöht werden. Wenn überhaupt nichts zutrifft, wird aus *i* eine Null.”

Date Created
10. Oktober 2016

Author
sven