Arbeitsblatt 12 - Funktionen

Was sind Funktionen?

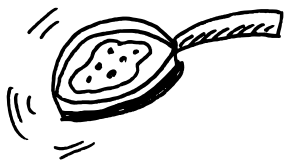
Definition der Funktion
backeEinenPfannkuchen ()

```
void backeEinenPfannkuchen () {
    fettePfanneEin();
    mixeTeig();
    tuePortionInPfanne();
    warteBisBraun();
    drehePfannkuchenUm();
    warteBisBraun();
    serviereDenPfannkuchen();
}
```

Aufrufe anderer Funktionen

Mittels *Funktionen* können Programme in einzelne *Teilbereiche/Teilprobleme* gegliedert werden. Jede Funktion übernimmt eine bestimmte Aufgabe und kann selber auch wieder in verschiedene Teilbereiche zerlegt werden. Die Programme werden so strukturierter und übersichtlicher, zusätzlich ersparen Funktionen jede Menge Tipparbeit!

Pfannkuchen



Mal angenommen, wir wollen einen Pfannkuchen backen, dann müssen wir einzelne **Teilprobleme** lösen: Wir müssen zuerst die *Pfanne einfetten*, dann den *Teig mixen*, eine *Portion Teig in die Pfanne tun*, kurz *warten*, bis der Teig braun ist, den *Pfannkuchen wenden*, wieder kurz *warten* und dann können wir den *Pfannkuchen servieren*! Für jedes dieser Teilprobleme definieren wir uns eine eigene Funktion, z.B. die Funktion warteBisBraun ():

```
void backeEinenPfannkuchen () {
    fettePfanneEin();
    mixeTeig();
    tuePortionInPfanne();
    warteBisBraun();
    drehePfannkuchenUm();
    warteBisBraun();
    serviereDenPfannkuchen();
}
```

Aufrufe der Funktion
warteBisBraun ()

```
void warteBisBraun () {
    while (guckUnterPfannkuchen () != braun) {
        warte (1 min);
    }
}
```

Definition der Funktion
warteBisBraun ()

Unser Hauptproblem besteht darin, einen Pfannkuchen zu backen. Dies wird von der Funktion **backeEinenPfannkuchen()** erledigt. Die Funktion hat also die Aufgabe, einen Pfannkuchen zu backen und ruft dafür einige andere Funktionen auf, um diese Aufgabe zu lösen. Die Aufgabe wird also in **Teilbereiche** zerlegt, die jeweils von einer eigenen Funktion gelöst werden, z.B. bewirkt der Aufruf der Funktion **warteBisBraun()**, dass unter den Pfannkuchen geguckt wird und solange dieser noch nicht braun ist, wird eine Minute gewartet. Diese Funktion rufen wir zweimal auf, damit der Pfannkuchen von beiden Seiten braun wird! Die Definition der Funktion, also die Festlegung was genau die Funktion machen soll, muss nur einmal implementiert werden. Danach kann man die Funktion beliebig oft im Programm verwenden.

Funktionen mit Parametern

Wenn man einer Funktion beim Aufruf **bestimmte Informationen** übergeben möchte, dann schreibt man in der Definition innerhalb der runden Klammern den oder die gewünschten **Parameter**. Das kann z.B. so aussehen:

Parameter

```
void backeVielePfannkuchen(int anzahl) {
    for (int i = 0; i < anzahl; i++) {
        backeEinenPfannkuchen();
    }
}
```

Eine Funktion kann *einen* oder auch *mehrere* Parameter übergeben bekommen, in unserem Beispiel bekommt die Funktion backeVielePfannkuchen **(int anzahl)** einen Parameter namens **anzahl** vom Typ Integer übergeben. Das bedeutet, dass man beim Aufruf der Funktion eine ganze Zahl mit angeben muss, die bestimmt wie viele Pfannkuchen gebacken werden sollen. Wenn du zum Beispiel alle deine Freunde zum Pfannkuchenessen einladen möchtest:

```
void bereiteAllesVor() {
    aufraeumen();
    saubermachen();
    tueDiesUndDas();
    ...
    backeVielePfannkuchen(200);
    ...
}
```

Durch diesen Aufruf wird jetzt automatisch 200 mal die Funktion backeEinenPfannkuchen() aufgerufen!

Wie oft wird dann insgesamt die Funktion warteBisBraun() aufgerufen? Weißt du's?

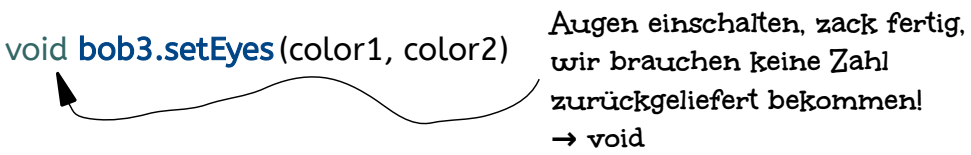
Wenn wir also 200 Pfannkuchen herstellen möchten, dann müssen wir nur noch `backeVielePfannkuchen (200);` schreiben. Wir müssen uns an dieser Stelle nicht um den Teig, das Mixen, das Abwarten usw. kümmern, dies wird alles von unseren anderen Funktionen erledigt!

Funktionen mit Rückgabewert

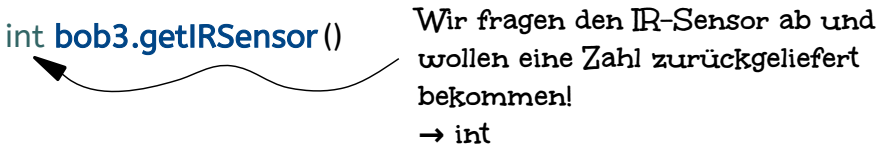
Ein **Rückgabewert** ist der Wert (z.B. eine Zahl), den eine Funktion zurückliefert, wenn sie aufgerufen wird. Manche Funktionen sind **mit** Rückgabewert und manche sind **ohne** Rückgabewert definiert. Das hängt ganz davon ab, was die Funktion machen soll!

Wenn man eine neue Funktion definiert, schreibt man vor den Namen der Funktion ein Schlüsselwort um anzuzeigen, ob die Funktion einen **Rückgabewert** liefert oder nicht:

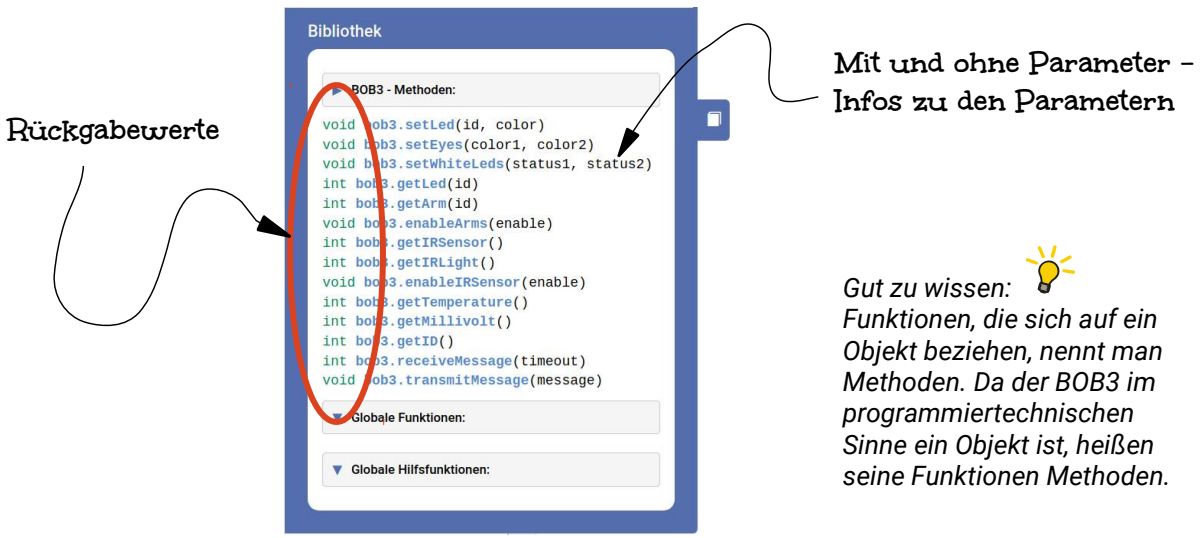
Für Funktionen **ohne Rückgabewert** schreibt man `void`, z.B. hier:



Für Funktionen **mit Rückgabewert** schreibt man den **Typ** des Rückgabewerts, z.B. `int`:



In der Bibliothek von BOB3 kannst du nachschauen, welche vordefinierten Funktionen und Methoden es für den Bob gibt, und wie diese definiert sind:



Gut zu wissen: Funktionen, die sich auf ein Objekt beziehen, nennt man Methoden. Da der BOB3 im programmiertechnischen Sinne ein Objekt ist, heißen seine Funktionen Methoden.

Aufgabe 1: Beschreibe, was eine Funktion ist und welche Vorteile sie bietet.

Aufgabe 2: Beschreibe, was die Funktion `warteBisBraun ()` von Blatt 1 macht.

Aufgabe 3: Beschreibe den Unterschied von Funktionen mit und Funktionen ohne Rückgabewert. Nenne jeweils ein Beispiel!

Aufgabe 4: Welche der folgenden Funktionen/Methoden haben einen Rückgabewert?

a) int `bob3.getIRLight ()`

b) void `bob3.transmitMessage (message)`

c) void `delay (milliseconds)`

d) void `bob3.setLed (id, color)`

e) int `min (zahl1, zahl2)`

f) int `bob3.getArm (id)`

g) int `rgb (red, green, blue)`

h) int `recall ()`

i) void `bob3.setEyes (color1, color2)`

j) void `remember (value)`

Aufgabe 5: Ordne die folgenden Funktionen/Methoden **aufsteigend** anhand der Anzahl ihrer Parameter.

a) int bob3.getIRLight ()

b) void bob3.transmitMessage (message)

c) void delay (milliseconds)

d) void bob3.setLed (id, color)

e) int min (zahl1, zahl2)

f) int bob3.getArm (id)

g) int rgb (red, green, blue)

h) int recall ()

i) void bob3.setEyes (color1, color2)

j) void remember (value)

Aufgabe 6: In unserer Funktion backeEinenPfannkuchen () von Blatt 1 wird unter anderem die Funktion mixeTeig () aufgerufen. Schreibe die Definition der Funktion mixeTeig (), verwende dazu folgende Funktionen, aber nur die passenden!!

stelleSchuesselInDenOfen ()

fuegeMehlHinzu ()

stelleSchuesselBereit ()

quirlen ()

einePriseSalzHinein ()

fuegeEierHinzu ()

hackeKnoblauch ()

lassDieKatzeProbieren ()

fuegeSchokoladeHinzu ()

fuegeOSaftHinzu ()

mahleKaffeebohnen ()

fuegeZuckerHinzu ()

einSchussMineralwasserHinein ()

stelleKeksdoseBereit ()

fuegeMilchHinzu ()