4.0.2. Grundlagen der Programmierung

4.0.2.1. Variablen und Schleifen

Ein gewöhnlicher Computer wie unser Mikrocontroller ist eine Maschine, die nicht (mit-)denken bzw. lernen kann, und daher genaue Anweisungen benötigt, um zu funktionieren. Ihre Stärke liegt darin, einfache Aufgaben in beliebig vielen Wiederholungen ohne Ermüdung auszuführen.

Beispiel:
Eine CNC (computerized numerical control)-Maschine soll so programmiert werden, dass sie z.B. nach **Eingabe** der **Werte** 3, 15 und 100 genau n = 100 Bolzen mit dem Radius r = 3 mm und der Länge l = 15 mm herstellt.

Um die Bedeutung der drei eingegebenen **Werte** erkennen zu können, verwendet man wie in der Mathematik die **Variablen** Radius r, Länge l und Stückzahl n.

Diese müssen zunächst **deklariert** werden, d.h., der Maschine wird mitgeteilt, wie die Variablen **heißen** und welche **Werte** (**Datentypen**) für die Eingabe in Frage kommen. Dies können u.a. ganze Zahlen (**integers**), Dezimalzahlen mit beliebiger Kommaposition mit 6 Stellen (**Gleitkommazahlen** bzw. **floating point numbers**), Zahlengruppen (**arrays**), einzelne Buchstaben (**characters**) oder Zeichenketten (**strings**) z.B. für Beschriftungen sein.

Damit die Maschine bei fehlender Eingabe nicht einfach das Programm vom letzten Mal wiederholt, werden die Variablen zu Beginn auf Null gesetzt (**initialisiert**).

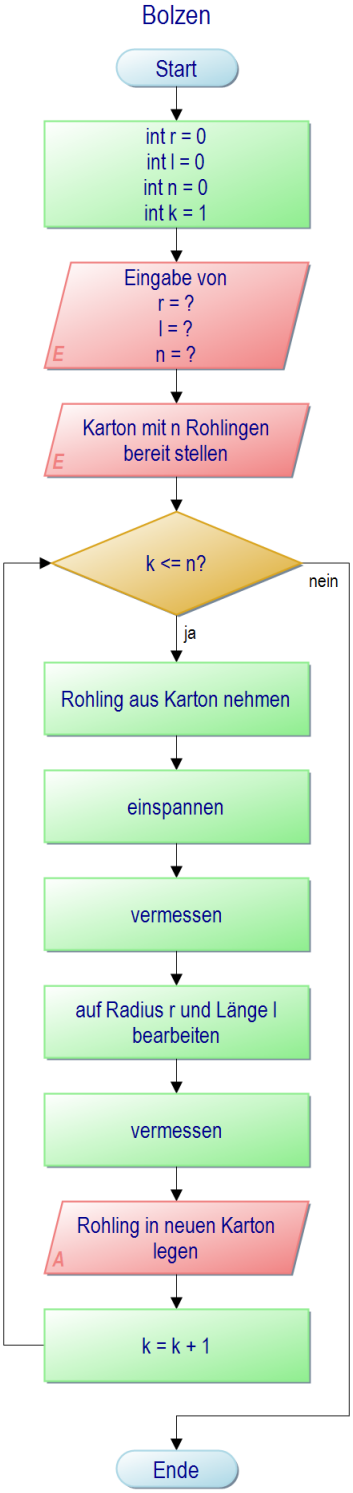
Auf dem Display der Maschine erscheint dann zu Beginn z.B. die **Abfrage**
Radius r = ?
Länge l = ?
Stückzahl n = ?

Nach der Eingabe der Werte führt die Maschine n = 100 mal die gleiche Tätigkeit aus: Rohling aus dem Karton nehmen, einspannen, vermessen, bearbeiten, wieder vermessen und in einen anderen Karton legen.

Zum Abzählen der Durchläufe dieser **Schleife** benötigt die Maschine eine **Zählvariable** k, die zu Beginn auf den Wert k = 1 gesetzt (**initialisiert**) wird und bei jedem Durchlauf um einen Schritt hoch gesetzt wird. Bei k = 100 verlässt die Maschine die Schleife und beendet das Programm.

Anders als in der Mathematik beschreibt das Zeichen = in der Informatik eine **Zuweisung**. k = k + 1 bedeutet also, dass der Wert von k um 1 vermehrt wird. Entsprechend würde k = 2*k eine Verdopplung des Wertes anzeigen.

Der rechts abgebildete **Ablaufplan** kann mit der Java-Freeware **PaPDesigner** des Georg Simon Ohm-Berufskollegs in Köln-Deutz leicht erstellt werden. Download z.B. unter http://www.PaPDesigner.softonic.de



Übungen: Aufgaben zur Programmierung Nr. 1 und 2

4.0.2.2. EVA-Prinzip und Funktionen
Wie oben beschrieben, verarbeitet ein Computer die Variablen bzw. ihre eingegebene Werte nach genau festgelegten Anweisungen zu einem eindeutigen Ergebnis. Dieses Prinzip nennt man kurz **Eingabe-Verarbeitung-Ausgabe** (EVA). In der Mathematik entspricht es dem **Funktionsbegriff**.

Z.B. ordnet eine Funktion f jedem rationalen Wert des Argumentes x den ebenfalls rationalen Funktionswert f(x) = 2x + 3 zu. In der Programmiersprache C definiert man diese Funktion ganz ähnlich: `float f(float x) = 2*x+3`. Der vorangestellte Datentyp `float` besagt einfach, dass sowohl die Argumente (**Eingabe**) x als auch die Funktionswerte (**Ausgabe**) f(x) 6-stellige Gleitkommazahlen sein dürfen. Der Compiler für die Arduino-Entwicklungsumgebung versteht nur einen kleinen Teil der Programmiersprache C und erlaubt daher keine eigene Definition von Funktionen. Die wichtigsten Taschenrechnerfunktionen stehen aber zur Verfügung, z.B. die **Quadratwurzel** (square root) `float sqrt(float x)` und die **Sinusfunktion** `float sin(float x)`.

Die beiden wichtigsten Funktionen eines Arduino-Programms (**sketches**) dienen der internen Organisation. Sie benötigen keine direkte Eingabe und liefern auch keine direkte Ausgabe. Ihr Datentyp ist daher **void** (nichts) und ihre Argumentklammer ist leer (muss aber trotzdem immer eingegeben werden!).

Jeder Sketch enthält zunächst die Funktion `void setup()`, welche die **Zuordnung der Ein- und Ausgänge** festlegt.

Beispiel:

```
int Wert = 0            // Die ganzzahlige Variable Wert wird deklariert und auf 0 //
                        // gesetzt
void setup()
{                       // Beginn der Zuordnung
pinMode(7, INPUT);      // Pin 7 wird als Eingang festgelegt
pinMode(13, OUPUT);     // Pin 13 wird als Ausgang festgelegt
}                       // Ende der Zuordnung
```

Drei wichtige Trennzeichen:
- Die **geschweiften Klammern** {} geben Anfang und Ende der Abschnittes an
- Die **Semikolons** geben das Ende einer Einzelanweisung an.
- Die **doppelten Schrägstriche** geben den Beginn eines einzeiligen **Kommentars** an, der vom Compiler ignoriert wird

Jeder Sketch enthält außerdem die Funktion `void loop()` für eine **Endlosschleife**. Alle nachfolgenden Befehle werden solange wiederholt, bis man den Stecker zieht.

Beispiel:

```
void loop()
{                       // Beginn der Endlosschleife
Wert = digitalRead(7);  // Die Eingabe (1 oder 0) von Pin 7 wird an die Variable Wert
                        // übergeben
digitalWrite(13, Wert); // Der Inhalt von Wert wird an Pin 13 übergeben
}                       // Ende der Endlosschleife
```

Wenn z.B. an Pin 7 ein Tastschalter und an Pin 13 eine LED angeschlossen sind, so leuchtet die LED auf, wenn der Schalter gedrückt wird. Die Endlosschleife bewirkt, dass man den Schalter so oft drücken kann wie man will.

Übungen: Aufgaben zur Programmierung Nr. 3 und 4