42. Grundlagen der Programmierung - ohne LUA

Autor: Mover
Quelle: EEP-Freunde-Forum

Grundlagen der Programmierung – die Theorie

Wer jetzt denkt hier geht es um die neue Skriptsprache LUA, der wird bei diesem Thema enttäuscht sein. Ich werde nur die theoretischen Grundlagen der Programmierlogik und damit der Programmierung abhandeln.

In der Programmierung gibt es zwei verschiedene Philosophien, die sich im Lauf der Entwicklung der Programmierung ergeben haben. Die ältere der beiden ist die sogenannte strukturierte Programmierung. Bei der anderen um die objektorientierte Programmierung, welche alle neueren Programmiersprachen unterstützen. Windows selbst ist komplett objektorientiert aufgebaut.

Wenn wir mit LUA arbeiten wollen geht es um die strukturierte Programmierung. Deswegen werde ich hier nur diese betrachten.

Wie in der gesamten EDV gilt auch hier das Grundprinzip der EDV und das ist das sogenannte EVA-Prinzip. Nein jetzt wird es nicht biblisch und ich schreibe nicht weiter über Adam und Eva. Die drei Buchstaben sind Abkürzungen.

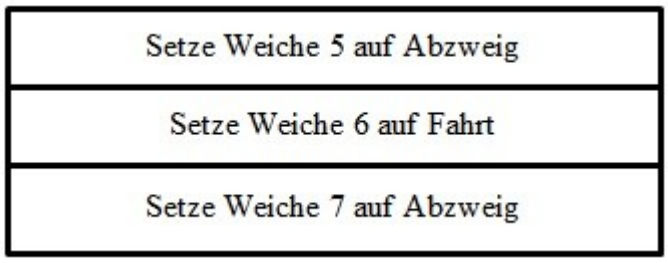
E = Eingabe
V = Verarbeitung
A = Ausgabe

Um als Programmierer eine Aufgabe zu lösen muss ich wissen was das Programm liefern soll. Also das was am Ende ausgegeben werden soll (in welcher Form auch immer). Als nächstes muss ich mir überlegen welche Formeln ich brauche bzw. welche Parameter benötigt werden. Danach ergibt sich dann was eingegeben werden muss (und das ist nicht immer über die Tastatur).

Um das dann umzusetzen gibt es drei wesentliche Strukturen, deren ich mich dann bediene. Welche sind das?

Als erstes gibt es die Folge (der Fachmann sagt Sequenz dazu). Hierbei handelt es sich um eine Folge von Anweisungen, welche immer von oben nach unten abgearbeitet werden. Dies erfolgt genau einmal, da es kein Zurück dabei gibt. Wo könnten wir dies in EEP gebrauchen? Nehmen wir dazu einfach das Schalten einer Fahrstraße.

Als grafisches Hilfsmittel gibt es zur Visualisierung das Struktogramm. Hier sieht die Folge dann so aus (Beispiel).

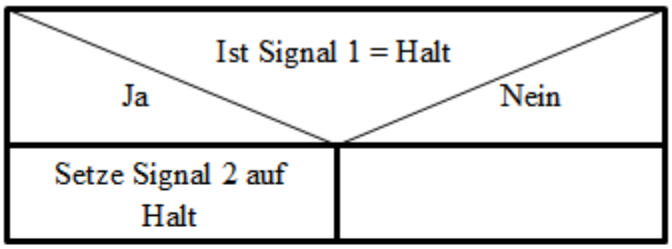


Ich habe hier bewusst die Anweisungen umgangssprachlich gemacht. Das dürfte auch für die meisten von euch, die noch nie programmiert haben einfacher sein. Außerdem hat es den Vorteil, dass die Logik nicht von der Programmiersprache abhängig ist. Kommt morgen satt LUA dann LEO, ändern sich lediglich die entsprechenden Befehle. Die eigentliche Logik bleibt aber die selbe.

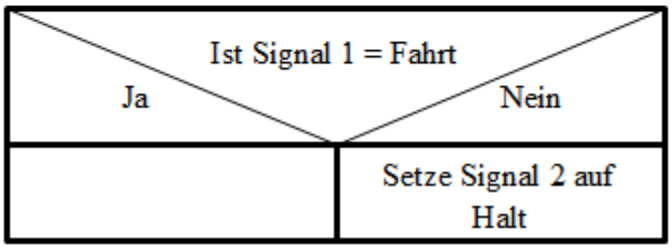
Die zweite Struktur, die wir brauchen ist die Verzweigung (fachmännisch Selektion genannt). Das Programm überprüft eine Bedingung und führt je nach Ergebnis die für diesen Fall vorgesehenen Anweisungen aus. Man unterscheidet dabei 3 Arten von Verzweigungen.

Die erste Art ist die einseitige Verzweigung. Hier sind nach Prüfung der Bedingung nur in einem Zweig (Ja oder Nein) Anweisungen vorgesehen. Auf dem anderen Zweig passiert nichts. Beispiel wäre in EEP das Abfragen eines Signals. Ist das zum Beispiel auf Halt, könnte ein anderes Signal auch auf Halt gestellt werden. Ist das erste Signal nicht auf Halt, soll nichts passieren.

Grafische Darstellung:



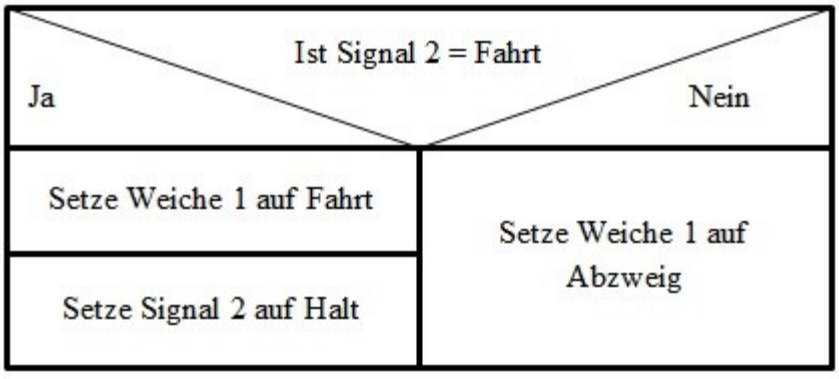
Ich könnte die Bedingung auch anders formulieren.



Merkt ihr es? Zwei unterschiedliche Denkweisen führen zum selben Ergebnis. Es gibt unterschiedliche Wege, die zum Ziel führen. Wie die aussehen, entscheidet letztendlich der, der vor dem Bildschirm sitzt. Es muss für ihn logisch sein, denn er muss sein Programm auch noch in einem halben Jahr verstehen und das ist manchmal schon schwer genug.

Wenn es eine einseitige Verzweigung gibt, dann gibt es auch mit Sicherheit eine zweiseitige Verzweigung. So ist es auch. Hier prüft das Programm die Bedingung. Trifft diese zu werden die Anweisungen im Ja-Zweig ausgeführt, sonst die Anweisungen im Nein-Zweig.

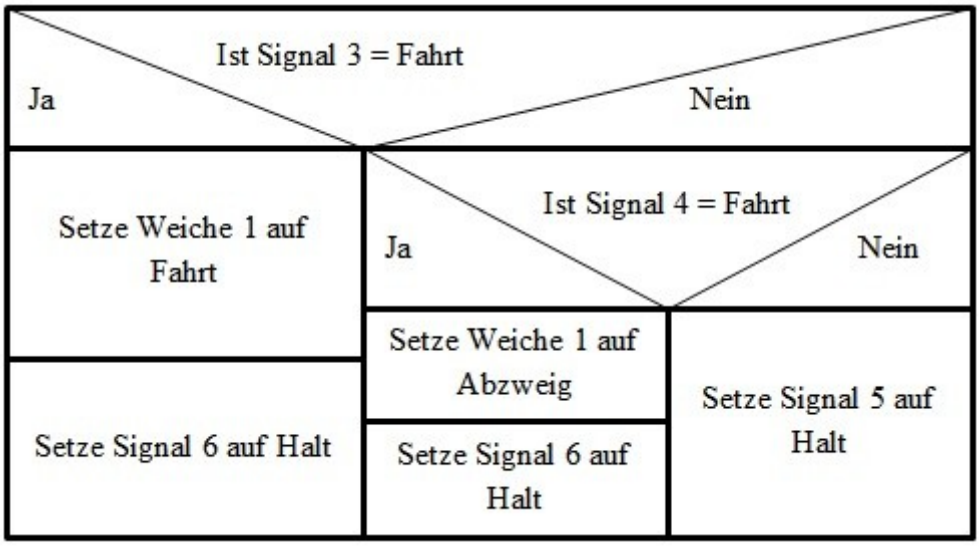
Nehmen wir mal als Beispiel einen eingleisigen Industrieabzweig. Ist dort schon ein Zug an der Rampe, soll der Zug welcher dort hin will, auf ein Wartegleis fahren. Dazu dient ein Signal (2), welches dies signalisiert. Die Weiche (1) wird auf Fahrt gestellt wenn der Abzweig frei ist, auf Abzweig wenn nicht (Wartegleis).



Die letzte Form ist eigentlich eine Mischung aus den beiden vorherigen Verzweigungen und ist die

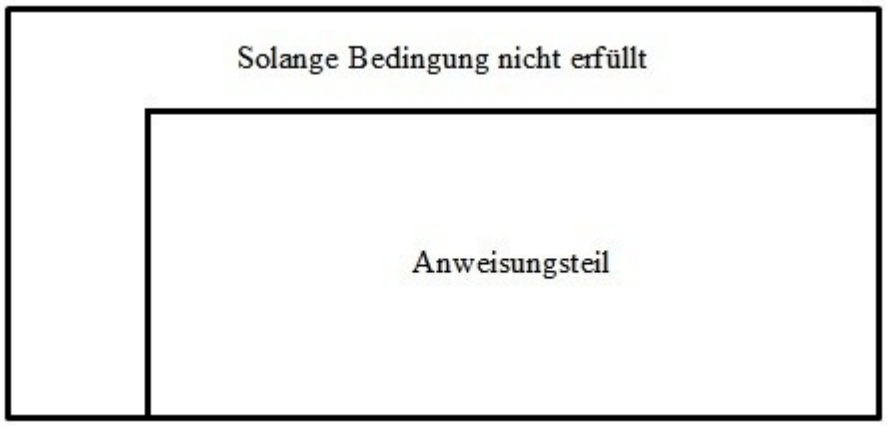
mehrseitige Verzweigung oder auch verschachtelte Verzweigung genannt.

Stellt euch einen zweigleisigen Bahnhof vor, wo wir prüfen wollen welches oder ob überhaupt ein Gleis frei ist. Dazu haben wir 2 unsichtbare Signale (3 und 4), welche sich merken ob ein Gleis frei (Fahrt) ist oder nicht (Halt). Ist kein Gleis frei, wird Signal 5 auf Halt gestellt. Jetzt visualisieren wir das ganze mal. Die Einfahrt wird durch Signal 6 signalisiert (damit nichts in die Richtung losfahren kann).



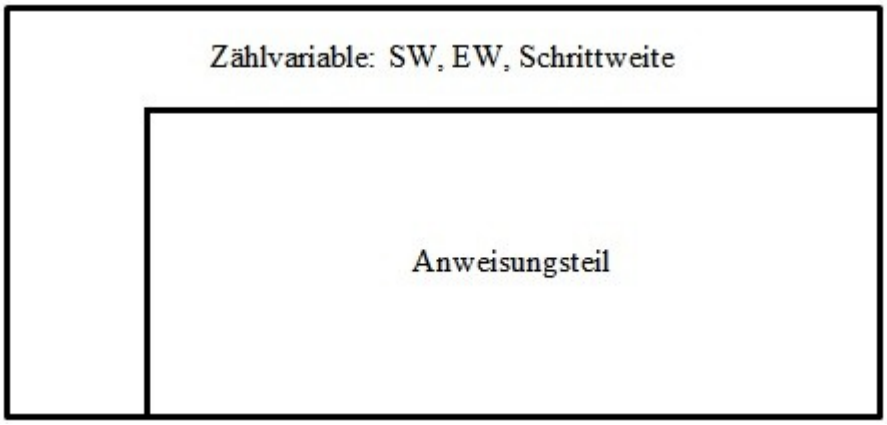
Die letzte Ablaufstruktur ist die Schleife (Wiederholung oder auch Iteration genannt). Hier gibt es 3 verschiedene Arten. LUA unterstützt diese. Ich selbst habe aber in einem LUA-Skript noch keine Schleife verwendet bzw. verwenden müssen. Deshalb werde ich die Schleifenarten lediglich ohne konkrete Beispiele vorstellen.

Die erste ist die kopfgesteuerte Schleife. Kopfgesteuert deshalb, weil die Bedingung oben (Kopf) definiert wird. Das Programm prüft also zuerst ob die Bedingung erfüllt ist. Ist dies gleich der Fall, dann wird der Anweisungsteil der Schleife gar nicht durchlaufen und das Programm wird mit der ersten Anweisung nach der Schleife fortgesetzt.



Als nächstes gibt es noch die fußgesteuerte Schleife. Die Bedingung wird unten (Fuß) nach dem Anweisungsteil überprüft. Das Programm durchläuft erst den Anweisungsteil der Schleife und prüft dann ob die Abbruchbedingung erfüllt ist. Diese Schleife wird also mindestens einmal durchlaufen. Sollte es darauf ankommen, dass dies ausgeschlossen sein muss, wäre die kopfgesteuerte Schleife die richtige Wahl.

Kommen wir nun zur letzten Schleifenart: der Zählschleife. Hier wird eine Zählvariable beginnend mit einem Startwert (SW) bis zu einem Endwert (EW) um eine Schrittweite erhöht bis der Endwert erreicht ist. Der Anweisungsteil wird bis dahin ausgeführt. Grafisch sieht dies so aus:



Das war es auch schon in diesem zugegeben sehr theoretischen Thema. Ich hoffe aber, dass ich dies doch verständlich rüberbringen konnte.

Nun noch ein paar gutgemeinte Ratschläge für alle die noch nie programmiert haben. Macht euch zu einem Problem eine kleine Skizze ähnlich der grafischen Beispiele. Überlegt im Vorfeld welche Anweisungen im einzelnen nötig sind. Schreibt euch solange ihr noch nicht geübt seid nicht gleich EEPSetSignal(1,1) auf sondern stattdessen "Setze Signal 1 auf Fahrt" bis ihr die Funktionen in LUA kennt und in der Sprache sicher seit. So kriegt ihr den Programmablauf auch ohne LUA hin und müsst euch erst mal nur darauf konzentrieren.

Mit zunehmender Erfahrung könnt ihr dann immer mehr darauf verzichten. Wie gesagt, übt erst mal einfache und vom Umfang her kleine Sachen. Die komplizierteren kommen dann von ganz allein, je sicherer ihr werdet.

Verseht eure Quelltexte mit Kommentaren. Es erleichtert ungemein das Verstehen des Programms, wenn man es nach längerer Zeit wieder angeht oder etwas ändern möchte.

Im nächsten Teil geht es dann darum, visualisierte Programmabläufe dann in LUA umzusetzen. Bis dahin viel Spaß mit EEP.