Spiel/ Simulation mit Excel-makro programmieren

Um ein Makro erstellen zu können, muss man diese erstmal aktivieren. Nachdem wir eine neue Excel Mappe erstellt haben, klicken wir auf "Datei", auf "Optionen" und dann auf "Menüband anpassen". Unter den Hauptregisterkarten setzen wir bei Entwicklertools den Haken. Das Optionen-fenster können wir dann schließen.

In der Navigationsleiste haben wie jetzt den Reiter "Entwicklertools". Hier können wir nun nach Belieben Makros erstellen und bearbeiten. Nach einem Klick auf "Makro aufzchn" und Bestätigung des Namens vom Makro wird alles aufgezeichnet, was wir in der Tabelle machen. Nachdem wir n der Zelle A1 eine 1 eintragen können wir auf "Aufzeichnung beenden" klicken, um die Aufnahme zu beenden.

Mit einem Klick auf "Makros" werden alle erstellten Makros angezeigt. Wenn wir unser erstelltes Makro auswählen und auf "bearbeiten" klicken, wird der Code des Makros angezeigt. Wenn wir in dem Makro/Code Fenster auf den Playbutton (grün) klicken, so wird das Makro ausgeführt.

Wir können auch Haltepunkte setzen, um genau zusehen was im geschriebenen Code passiert. Um einen Haltepunkt in einer Zeile zu erstellen klicken wir links von der Zeile auf den schmalen grauen Rand. Dort erschein ein roter Punkt und auch die Zeile wird rot. Das Programm hält dann an diesem Punkt. Mit F8 können wir du einen Schritt weiter gehen, mit F5 springen wir zum nächsten Haltepunkt bzw. zum Ende des Codes und beenden ihn.

Die 1 wird in im Feld A1 durch die Zeilen

Range("A1").Select ActiveCell.FormulaR1C1 = "1"

realisiert. Durch das Range("A1").Select wird A1 zu aktiven Zelle. Mit ActiveCell.FormulaR1C1= "1" wird in A1 eine 1 eingetragen.

Das ist aber nicht die einzige Möglichkeit eine Zelle zu befüllen.

Mit dem Befehl Cells(1,1) = 1 passiert genau das gleiche. Der Cells-Befehl ist wie folgt aufgebaut:

Cells(Y-Position, X-Position) = Wert

Cells(1,2) = 3 schreibt also in das Feld B1 eine 3. Wenn wir statt einer Zahl ein Wort/ bzw. Text schreiben wollen, müssen wir vor und hinter diesem Anführungszeichen schreiben also:

Cells(4, 5) = "Test"

Statt 1 und Test zweimal zu schreiben, können wir diese in Variablen speichern. Zum Speichern von unserer 1 benutzen wir einen Integer, der nur ganze Zahlen speichern kann. Für unseren Test benutzen wir einen String, der Texte und Zahlen speichern kann.

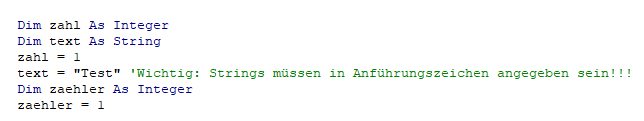
Die Variablen erstellen wir wie folgt:

Dim name as Variable

Um den erstellten Variablen Werte zuzuweisen, machen wir folgendes:

name = Wert

In unserem Beispiel also:



Wir können jetzt also einfach folgendes schreiben, um 1 oder Test auszugeben:

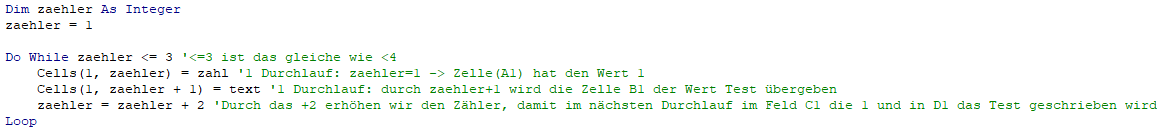


Statt viermal den Cells Befehl zu nutzen, können wir auch eine Schleife benutzen. Die Do While Schleife führt etwas aus, solange eine Bedingung erfüllt ist. Wir brauchen also eine Bedingung, damit 1 und Test 2-mal abwechselnd ausgegeben werden. Wir können also einfach einen weiteren Integer erzeugen und diesen in der Schleife hochzählen. Die Syntax der Do While Schleife sieht wie folgt aus:

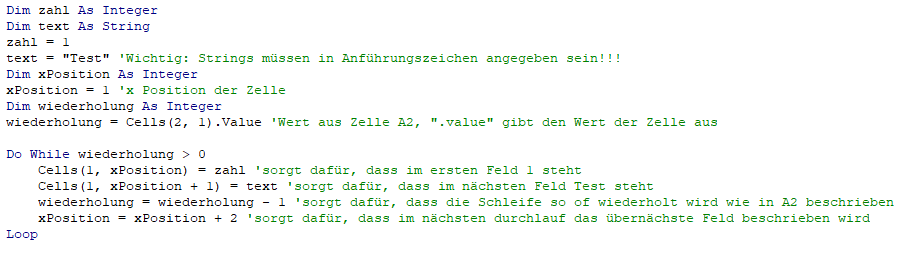
Do While Bedingung

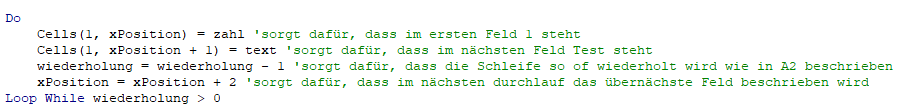
mach was

Loop

In unserem Beispiel also:

Statt festzulegen, dass die1 und der Text insgesamt viermal, können wir in ein bestimmtes Feld eintragen, wie oft beide zusammen vorkommen. Hierzu müssen wir unseren eingetragenen Wert speichern. Wir wissen, dass es eine Zahl ist. Wir brauchen also einen weiteren Integer. Um den Wert eines Feldes in dem Integer zu speichern setzen wir unsere Variable mit dem Feld gleich.

In unserem Beispiel also:

Wir nutzen hier eine kopfgesteuerte Schleife, das heißt wir überprüfen erst die Bedingung, bevor wir in die Schleife gehen. Wenn wir aber die Bedingung an das Ende setzen führen wir auf jeden Fall die Schleife mindestens einmal aus. Das ist dann eine Fußgesteuerte Schleife.

In unserem Bespiel also:

Wenn die Variable "Wiederholung" Null ist wird trotzdem 1 und Text jeweils einmal ausgegeben, obwohl wir das nicht wollten.

Jetzt wollen wir 2 übergebene Zahlen miteinander vergleichen und die größere Zahl ausgeben. Unseren momentanen Code können wir einfach kommentieren, indem wir vor jede Zeile ein einfaches Anführungszeichen (') schreiben. Kommentierte Zeilen werden beim ausführen des Makros ignoriert.

Wir erstellen uns 2 Integer denen wir die Werte aus 2 Zellen übergeben. Um zu überprüfen, welche Zahl größer ist, benutzen wir eine If-Abfrage. Die Syntax einer If-Abfrage sieht wie folgt aus:

If Bedingung Then

mach was

Elseif Bedinung Then

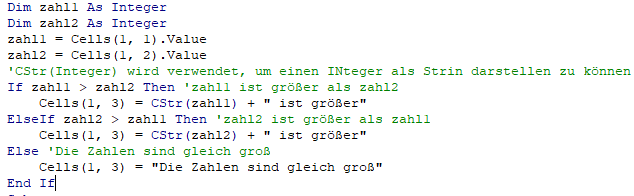
mach was

Else

mach was

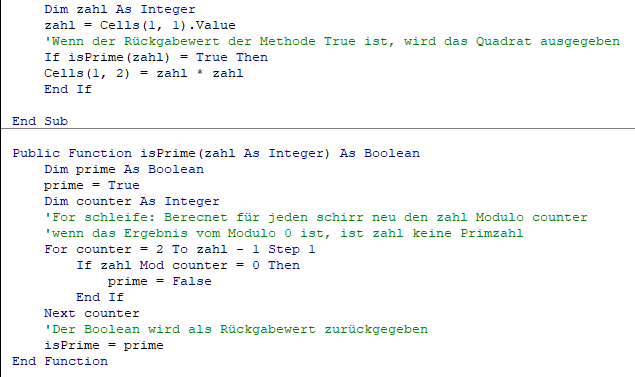
End If

Das Elseif ist ein weiterer Fall, der abgefragt werden kann. Das Else beschreibt, was passiert wenn keiner der oberen Fälle stimmt. In unserem Fall brauchen wir alle 3, denn entweder ist eins größer als das andere, oder die Zahlen sind gleich groß. Die If-Abfrage sieht dann so aus:

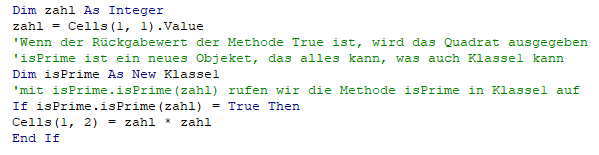


Jetzt geben wie nur eine Zahl an und nur wenn diese Zahl eine Primzahl ist, berechnen wir von ihr das Quadrat und geben dieses aus. Wir brauchen also eine Prüfung die erkennt, ob die Zahl eine Primzahl ist. Wenn ja soll von ihr das Quadrat berechnet werden.

Für die Prüfung brauchen wir einen weiteren Datentypen der entweder "wahr" oder "falsch" ist. Je nachdem, ob die Zahl eine Primzahl ist oder nicht. Dieser Datentyp heißt Boolean und ist entweder true oder false. Die Überprüfung der Zahl können wir zudem auslagern, falls wir sie später nochmal brauchen. Für die Überprüfung eignet sich am besten eine For-schleife, da man mit ihr in jedem Schritt den Einzelwert berechnen kann. Das sieht dann so aus:



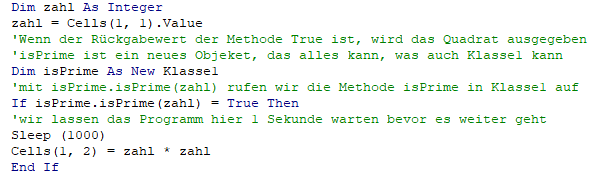
Um von einem anderen Makro auf die istPrimzahl Methode zugreifen zu können, müssen wir die Methode in eine eigene Klasse umziehen. Hierzu erstellen wir zunächst ein neues Klassenmodul (Rechtsklick auf VBAProject->einfügen->Klassenmodul). In das Klassenmodul kopieren wir die Funktion und löschen sie aus unserem Makro. Um die Funktion in unserem Makro nutzen zu können, müssen wir uns ein neues Objekt als Klasse1 erstellen.



Jetzt wollen wir 1 Sekunde warten, bevor wir das Quadrat ausgeben. Hierfür brauchen wir folgende Zeile, die wir vor unser Makro schreiben:

Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Die Sub können wir jetzt einfach aufrufen:



Ansonsten steht hier noch mehr:

https://de.wikibooks.org/wiki/VBA\_in\_Excel