

# Programming example for TURING MACHINE

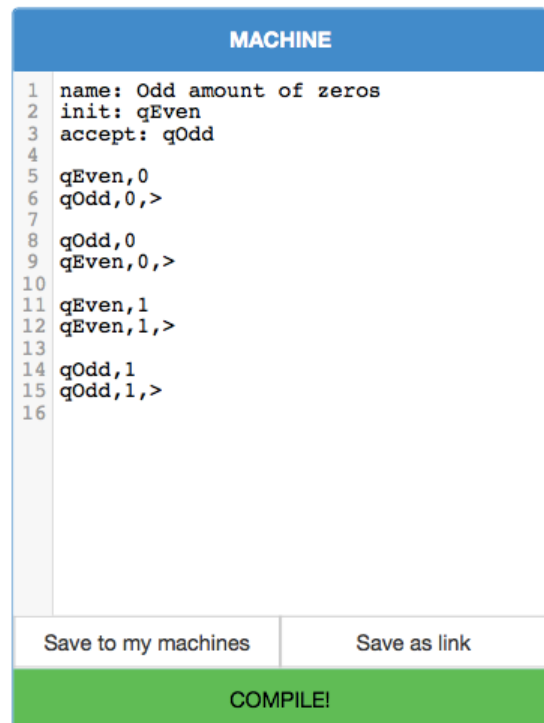
This section presents a programming example from coding to running. However, it does not explain the programming language, which can be learned in the [previous section](#). We will construct a machine for distinguishing strings with an odd amount of zeros. To simplify the construction, we will assume every input is a binary number (a number containing only 1s and 0s).

Our machine will have two states, `qEven` and `qOdd`. The machine will read the input from left to right, so every transition instruction will end with the symbol `>`. While reading the symbols, if the amount of zeros read so far is even, the machine will be in state `qEven`. Otherwise it will be in state `qOdd`. Based on this principle, and considering that at the beginning the machine hasn't read any zeros, the initial state is `qEven`.

Now let's think about the transitions. If the machine is in state `qEven`, it has read an even amount of zeros. Hence, if it reads a new zero it should change to state `qOdd`. Accordingly, the transition shown in lines 5 and 6 of Figure 1 is defined. Following the same argument, the rest of the transitions are defined between lines 8 and 15 in the code of Figure 1.

Once the machine has finished processing the number, it will be reading a blank cell. Since there are no transitions defined for the blank symbol, the running will end. In this case, the machine will be in state `qOdd` if and only if the number had an odd amount of zeros. Therefore the set of accepting states contains only state `qOdd`, as shown in Figure 1.

Now let's see a complete run of our Turing machine. Assume we already compiled the code and loaded the string '0100'. Figure 2 depicts The machine panel at the beginning of the run.



```
1 name: Odd amount of zeros
2 init: qEven
3 accept: qOdd
4
5 qEven,0
6 qOdd,0,>
7
8 qOdd,0
9 qEven,0,>
10
11 qEven,1
12 qEven,1,>
13
14 qOdd,1
15 qOdd,1,>
16
```

Figure 1



Figure 2

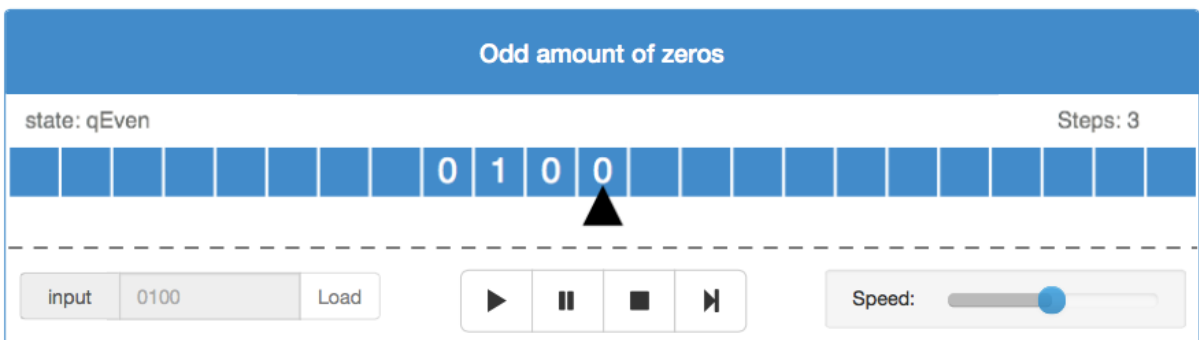
At this point the state is qEven and the head is reading a 0, the instruction of the first transition would be applied. After this, the machine would look as shown in the next Figure.



Now the machine is in state qOdd and reading a 1, so the last transition is applied and the machine changes to the configuration shown in Figure 4.



The machine is now in state qOdd and reading a 0. The second transition is applied and the head now points to the last symbol of the input.



Finally, the first transition is applied again.

The screenshot shows a simulation window for a Turing machine. At the top, a blue header reads "Odd amount of zeros". Below this, the current state is "qOdd" and the number of steps is "Steps: 4". The central part of the window is a tape represented by a row of blue cells. The first four cells contain the binary string "0100", and the rest are blank. A black triangle points to the first blank cell. Above the tape, the text "Word Accepted!" is displayed. At the bottom, there is a control panel with an "input" field containing "0100", a "Load" button, a set of playback controls (play, pause, stop, next), and a "Speed" slider.

Now the machine is in state  $q_{\text{Odd}}$  and the pointed cell is blank. Since there is no transition with condition ' $q_{\text{Odd}}, \_$ ', the process stops. Notice that the last state of the run was an accepting state ( $q_{\text{Odd}}$ ). Hence, we say that the machine *accepts* the input 0100. This is what we expected, as the machine was designed to accept every binary number with an odd amount of zeros. This concludes our example, but there is still a lot to be learned. To continue with Turing machines that have more than one tape read the [next section](#).