# Tutorial for TURING_MACHINE

As explained in the previous section, how a Turing Machine processes the input depends on a set of **transitions**, on the **input word**, and on the **initial state**. Once the processing finishes, to know if at the end of an execution the machine accepted the input, we must specify a set of **accepting states**.

Easy as it sounds, to program a TURING_MACHINE you must provide the transitions, the initial state and the accepting states. Before explaining how to define these three objects, we will show how to use the controls of the machine. Assume you already programmed your TURING_MACHINE. Probably your code looks something similar to Figure 1.

Once your machine is programmed, you must click the green COMPILE button. If your code has any errors, the compiler will let you know one by one the lines you must change. If not, the machine panel will come up (see Figure 2).

As the first section explained, Turing machines are intended to process inputs. Hence, before running your machine you must provide an input word in the bottom left corner of the machine panel and click the Load button. Now you are ready to start using the play, pause, stop and step buttons to run the machine. If you want to have a try, use the code from one of the examples on the website.


Figure 1

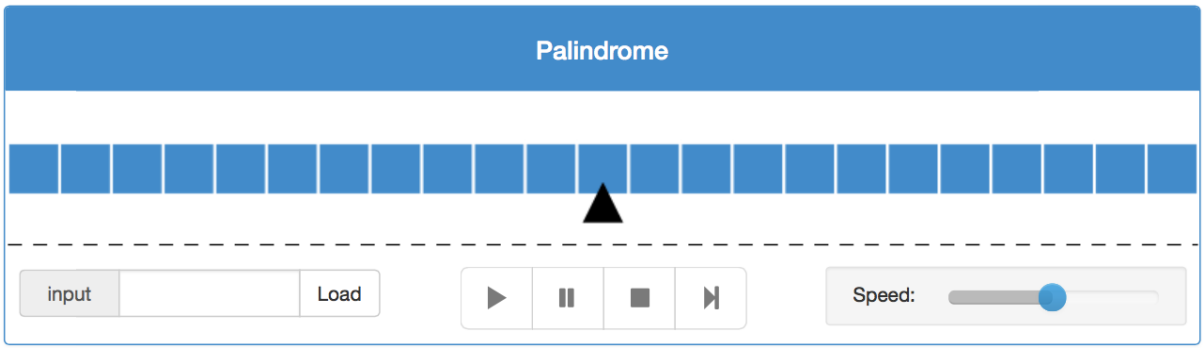
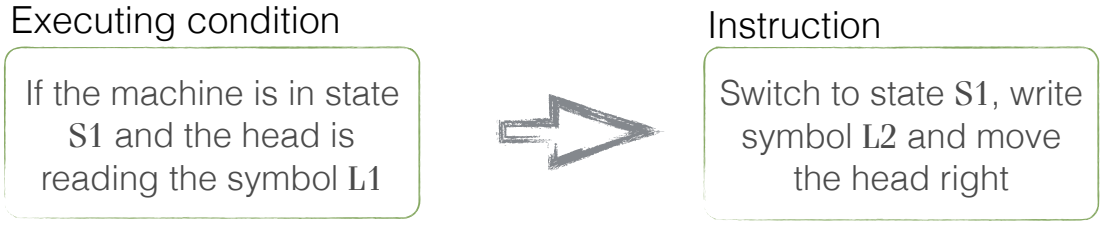



Figure 2

Now let's see how to program a TURING_MACHINE. As already mentioned, everything you must provide is the set of **transitions**, the **initial state** and the **accepting states**. Providing the initial and accepting states is straight-forward. For example, the machine programmed in Figure 1 has initial state **q0**, and accepting states **q10** and **q11**. Figure 1 also shows how to give a name to your machine.

Next we must provide the transitions. Recall from the first section that a transition of a Turing machine looks similar to this:

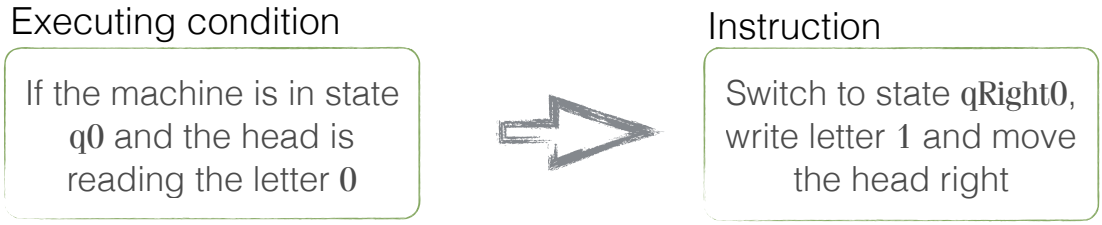



States S1 and S2 can be any state of the machine, and symbols L1 and L2 can be practically any symbol. The head can move one cell right, one cell left, or stay where it is. To encode this we use the symbols shown in Table 1.

| Movement | Representation |
|---|---|
| Stay | - |
| Move right | > |
| Move left | < |

Table 1

To write one transition we use two lines. One for representing the executing condition and the other to represent the instruction. To encode the executing condition just write the name of the state S1 and the symbol L1 separated by a comma. To encode the instruction, write the name of the state S1, the name of the letter L2 and the symbol representing the head's move. For example, the transition coded in lines 5 and 6 of Figure 1 is read as next:




A final comment on the programming language is that we use underscore (_) to encode blank cells. For example if you want to write a transition which's condition is 'If the machine is in state q, reading a blank cell', the first line of that transition would be 'q,_'.

To see a complete programming example visit the next section.