LEGO NXT Robots using NXC – Part 2

Adding comments

To make your program even more readable, it is good to add some comment to it. Whenever you put `//` on a line, the rest of that line is ignored and can be used for comments. A long comment can be put between `/*` and `*/`. Comments are syntax highlighted in the BricxCC. The full program could look as follows:

```
/* 10 SQUARES
   This program make the robot run 10 squares
*/
#define MOVE_TIME 500 // Time for a straight move
#define TURN_TIME 360 // Time for turning 90 degrees

task main()
{
       repeat(10) // Make 10 squares
       {
              repeat(4)
              {
                     OnFwd(OUT_AC, 75);
                     Wait(MOVE_TIME);
                     OnRev(OUT_C, 75);
                     Wait(TURN_TIME);
              }
       }
       Off(OUT_AC); // Now turn the motors off
}
```

Using variables

Variables form a very important aspect of every programming language. Variables are memory locations in which we can store a value. We can use that value at different places and we can change it. Let us describe the use of variables using an example.

```
#define TURN_TIME 360
int move_time; // define a variable
task main()
{
       move_time = 200; // set the initial value
       repeat(50)
       {
              OnFwd(OUT_AC, 75);
              Wait(move_time); // use the variable for sleeping
              OnRev(OUT_C, 75);
              Wait(TURN_TIME);
              move_time += 200; // increase the variable
       }
       Off(OUT_AC);
}
```

Random numbers

In all the above programs we defined exactly what the robot was supposed to do. But things get a lot more interesting when the robot is going to do things that we don't know. We want some randomness in the motions. In NXC you can create random numbers. The following program uses this to let the robot drive around in a random way. It constantly drives forwards for a random amount of time and then makes a random turn.

```
int move_time, turn_time;
task main()
{
    while(true)
    {
        move_time = Random(600);
        turn_time = Random(400);
        OnFwd(OUT_AC, 75);
        Wait(move_time);
        OnRev(OUT_A, 75);
        Wait(turn_time);
    }
}
```

The if statement

Sometimes you want that a particular part of your program is only executed in certain situations. In this case the if statement is used. Let me give an example. We will again change the program we have been working with so far, but with a new twist. We want the robot to drive along a straight line and then either make a left or a right turn. To do this we need random numbers again. We pick a random number that is either positive or negative. If the number is less than 0 we make a right turn; otherwise we make a left turn. Here is the program:

```
#define MOVE_TIME 500
#define TURN_TIME 360
task main()
{
    while(true)
    {
        OnFwd(OUT_AC, 75);
        Wait(MOVE_TIME);
        if (Random() >= 0)
        {
            OnRev(OUT_C, 75);
        }
        else
        {
            OnRev(OUT_A, 75);
        }
        Wait(TURN_TIME);
    }
}
```

The **if** statement looks a bit like the while statement. If the condition between the parentheses is true the part between the brackets is executed. Otherwise, the part between the brackets after the word **`else`** is executed. Let us look a bit better at the condition we use. It reads `Random() >= 0`. This means that `Random()` must be greater-than or equal to 0 to make the condition true. You can compare values in different ways. Here are the most important ones:

==	equal to
<	smaller than
<=	smaller than or equal to
>	larger than
>=	larger than or equal to
!=	not equal to

You can combine conditions use `&&`, which means "and", or `||`, which means "or". Here are some examples of conditions:

`true`	always true
`false`	never true
`ttt != 3`	true when ttt is not equal to 3
`(ttt >= 5) && (ttt <= 10)`	true when ttt lies between 5 and 10
`(aaa == 10) || (bbb == 10)`	true if either aaa or bbb (or both) are equal to 10

The do statement

There is another control structure, the **do** statement. It has the following form:

```
do
{
       statements;
}
while (condition);
```

The statements between the brackets after the **do** part are executed as long as the condition is true. The condition has the same form as in the **if** statement described above. Here is an example of a program. The robot runs around randomly for 20 seconds and then stops.

```
int move_time, turn_time, total_time;
task main()
{
       total_time = 0;
       do
       {
              move_time = Random(1000);
              turn_time = Random(1000);
              OnFwd(OUT_AC, 75);
              Wait(move_time);
              OnRev(OUT_C, 75);
              Wait(turn_time);
              total_time += move_time;
              total_time += turn_time;
       }
       while (total_time < 20000);
       Off(OUT_AC);
}
```