**Intro to Programming**

**COEN 10 -- Prof. Silvia Figueira**

**What is programming?**

Programming is about telling the computer what to do. The computer can execute a **limited set of actions**, which need to be combined to enable it to accomplish sophisticated tasks. A program is simply a combination of these actions.

Example 1:   
Imagine that you are programming robot R to move from point A to point B, as shown in Figure 1. Robot R has very limited actions. It can walk one step ahead, turn left, and turn right. To simplify the program, we will call these actions ahead, left, and right, respectively. Assuming, it takes robot R 20 steps to complete each segment of the path, your program may look like Program 1 below.

Figure 1

| |

| |

| |

------------------- ----------------

R

------------------- ----------------

A | |

| |

| |

| |

| |

| |

| |B

Program 1

1. repeat 20 times: ahead
2. right
3. repeat 20 times: ahead

To make robot R move from point A to point B, we combined the actions to create a program in which robot R follows the specified **sequence**: it walks 20 steps ahead, by repeating action ahead 20 times in a **loop**, then it turns right, and then it walks 20 steps again by repeating action ahead 20 times in another **loop**. By following the **execution path**provided by the program, robot R is able to go from point A to point B.

Note that, if we had made a mistake and told robot R to walk 15 steps, robot R would not be able to turn right. Also, if we had told robot R to turn left instead of right, it might never have reached point B. Therefore, it is crucial that we give the computer **correct instructions**, or it may not be able to accomplish the right task.

In Example 1, robot R had a very limited set of actions. Let's improve robot R, by adding a sensor which enables robot R to identify turning points and the final destination. This new capability will enable us to write more sophisticated programs.

Example 2:   
Imagine that you are programming robot R to move from point A to point B, as shown in Figure 2. Robot R has limited actions. It can walk one step ahead, turn left, turn right, identify a turning point, and identify the final destination. To simplify the program, we will call the actions ahead, left, and right, and we will call the **conditions** at turning point and at final destination. The number of steps required in each segment of the path is not known beforehand. Your program may look like Program 2 below.

Figure 2

| |

| |

| |

------------------- ----------------

R

------------------- ----------------

A | |

| |

| |

| |B

Program 2

1. repeat if not at final destination
   * if at turning point, right
   * otherwise, ahead

To make robot R move from point A to point B, we combined the actions and conditions to create a program in which robot R follows the specified sequence inside a loop that executes while robot R is not at the final destination. Inside the loop, at each **iteration**, robot R checks a **condition**, i.e., the turning point. If it is at a turning point, robot R turns right. Otherwise, it walks one step ahead.

Note that Program 2 is more powerful than Program 1, since it enables robot R to move in routes for which the lenght of each segment is not known beforehand. In this case, all the programmer needs to know beforehand is the direction of the turns.

Note also that the loop in Program 2 is different from the loop in Program 1. In program 1 the loop executes a number of times, i.e., 20 times, while in Program 2, the loops executes while a condition is true, i.e., while robot R is not at the final destination.

Example 3:   
As an exercise, consider the scenario in Figure 3. Write a program to take robot R from point A to point B, assuming you do not know the lenght of each segment of the path, but robot R can sense turning points X and Y. Write a second program to take robot R from point A to point B, assuming you know the lenght of each segment of the path, which are represented by S1, S2, S3, and S4, as shown in the picture. Write the same programs to take robot R from point A to point C.

Figure 3

| |

| |

| |

------------------- ----------------

R S1 S2

------------------- ----------------

A X| | B

| |

| | S3

| |

| |

------------------- ------------------------------

S3 S4

------------------- ------------------------------

Y| | C

| |

| |

Computers nowadays are powerful and able to execute more than one instruction at a time. By using this capability, we can program robot R to do multiple things at the same time. For example, robot R might be able to walk and do something else at the same time.

Both correctness and efficiency are key in programming. Correctness is of course imperative, and does not necessarily imply effectiveness. An effective program will execute faster and/or use less resources, depending on the effectiveness goal. In some situations, an ineffective program, although correct, may be useless. For example, in real time systems (e.g., games, data streaming, and surgical devices), the computer needs to be extremely responsive, requiring high efficiency.

Example 4: Imagine that you are writing a program to help robot R get from point A to point B in Figure 4. Depending on the program, the robot may take different paths, as shown in Program 4, a and b, below.

Figure 4

| | | |

| | | |

| | | |

------------------- ---------------------------------------------- --------------------

R

------------------- ---------------------------------------------- --------------------

A W| | Y| |

| | | |

| | | |

| | | |

| | | |

------------------- ---------------------------------------------- ---------------------

------------------- ---------------------------------------------- ---------------------

B X| | Z| |

| | | |

| | | |

Program 4a:

1. repeat if not at final destination
   * if at turning point W or X, right
   * otherwise, ahead

Program 4b:

1. repeat if not at final destination
   * if at turning point Y or Z, right
   * otherwise, ahead

Both programs are right, as both will take robot R from A to B. However, robot R will arrive earlier if it executes Program 4a, since the path chosen is shorter.

These examples show that programming requires precision and good planning. A bad program may lead the computer to do wrong things or do the right thing but slower than necessary.