Lab: Conditionals

CSC 161, “Imperative Problem Solving and Data Structures”
Department of Computer Science · Grinnell College
February 8, 2016

In this lab, we’ll practice constructing conditions and conditional statements in C, conditioning the motions of Scribbler2 robots on readings from their sensors.

Pairs

Here are the pairs for today’s lab. Drivers are on the left.

- Ella Nicolson and Lilly Webster
- Wyatt Heritage and Sophie Gaschott
- Erhaan Ahmad and Tyler Williams
- Ying Zhang and Saung Thuya
- Colin Greenman and Sanjay Sudhir
- Lex Martin and Yuyin Sun
- Dennis Chan and Faizaan Ali
- Tanner Tufto and Jae Hong Shin
- Eli Salm and Mattori Birnbaum
- Anna Blindermann and Josh Lavin
- Gemma Nash and Jong Hoon Bae
- Zachary Susag and Cory McCartan

The Scribbler2 Sensors

The Scribbler2 robots have several sensors at each end. On the front end (if you consider the “fluke” to be at the back), there are three light sensors, (pointed generally forwards and upwards, but in slightly different directions) and two obstacle sensors, one on each side, with an emitter of infrared light placed between them The obstacle sensors work by detecting infrared light reflected from objects that lie in the robot’s path.

You can use the rGetLightTxt and rGetIRText functions, as described in the MyroC.h header file that we walked through last week, to recover readings from the sensor. The first argument to either function should be a string literal that names the position of the sensor ("left", "middle", or "right" for the light sensors, just "left" or "right" for the obstacle sensors). The second obstacle should be a positive integer indicating the number of times in succession the sensor should be read. (It may be useful to read the sensor more than once, in case the readings are not very accurate.)

The rGetLightTxt function returns an integer approximately equal to the mean of the sensor readings indicating the amount of light that the sensor perceived on a scale from 0 (maximum brightness) to approximately 65000 (near-maximum darkness).

The rGetIRText function returns either 0 (“no obstacle detected”) or 1 (“obstacle detected”), depending on whether the mean of the sensor readings fell below or above some threshold (not specified in the available documentation).

Exercises

Exercise 000. Write, compile, and run a C program that (a) takes three successive readings from the middle light sensor at the scribbler end of the robot and prints out their mean, appropriately labeled, and (b) takes three successive readings from each of the obstacle sensors at the scribbler end of the robot and prints out the string "Blocked!", followed by a newline, if either sensor detects an obstacle (or if both do), or the string "Ready to roll!", followed by a newline, if neither sensor detects an obstacle.
Exercise 001. Write up a list of tests that you could do to determine whether the program you wrote in the preceding exercise produces answers that are correct (or at least plausible). For each test, state the result that you expect to obtain if your program is correct.

Exercise 010: Run the tests you devised in the preceding exercise, and record the results. If any of the tests yielded results different from the results you predicted, diagnose the problem. What additional tests could you do to determine what went wrong? How might you vary the program code to get additional results that might help you towards a diagnosis?

Exercise 011: Revise your code so that the robot issues a low-pitched beep (350 hertz for half a second) if its left obstacle sensor detects an obstacle and/or a high-pitched beep (700 hertz for half a second) of its right obstacle sensor detects an obstacle. Recompile and test the revised program.

Exercise 100: Write, compile, and test a C program that directs the robot to move as follows:

- If both of its obstacle sensors report the presence of obstacles, the robot should back up for one second.
- If its left sensor reports an obstacle but its right one does not, the robot should turn right ninety degrees and then move forward for one second. (You may have to experiment to find out how quickly the robot can turn itself ninety degrees.)
- Similarly, if its right sensor reports an obstacle but its left one does not, the robot should turn left ninety degrees and then move forward for one second.
- Finally, if neither sensor reports an obstacle, the robot should simply move forward for one second.

Exercise 101: Define a C function turn_towards_light that takes no arguments and returns no value, but directs the robot to turn left slightly if its left light sensor returns a value that is less than the value returned by its right light sensor, and to turn right slightly in the opposite case. Write, compile, and test a C program that invokes the turn_towards_light function fifteen times in succession.

Exercise 110: Add a call to the turn_towards_light function that causes the robot to beep briefly at a pitch somewhere in the range from 300 to 3300 hertz, depending on the reading from the middle light sensor: 300 hertz if the sensor reading is 65000 (or higher), 3300 hertz if it is zero. If the sensor reading has some intermediate value, select the appropriate pitch by linear interpolation. (In other words: find a linear function $f$ of the sensor reading such that $f(0) = 300$ and $f(65000) = 300$ and have the program apply $f$ to the sensor reading to select the pitch for the beep.)

There are also three obstacle sensors ("left", "middle", and "right") at the “fluke” end of the robot. The rGetObstacleText function samples one of these sensors and returns an integer value in the range from 0 to 6400, depending on the proximity of the obstacle: 0 indicates that any obstacle is too far away to be detected at all, while 6400 indicates an obstacle very close to the robot. (In addition, when the robot’s battery starts to lose power, the range of values shrinks. Fortunately, the MyroC library contains a function that returns the current voltage produced by the robot’s batteries.)

Exercise 111: Write, compile, and test a C program that prints out reports from the robot’s obstacle sensors, unless the battery voltage is less than 4.5 volts, in which case your program should ask the user to put new batteries in the robot and recharge the old ones.