Computing for Engineers (EE108)

Assignment 4 – strings and advanced types

Objectives: This assignment aims to further develop the use of strings, and advanced types in C (on the Arduino platform).

Learning outcomes:

- Use of strings and a simple FIFO buffer for reading input
- Potentially, use of advanced types using struct and typedef

Assignment instructions

1. The assignment is for individuals working independently and builds on the lab exercises
2. Create a new sketches as required, naming them as specified in each section.
3. Create a single plain text submission file (.txt) for the assignment using a plain text editor (e.g. NotePad++). Copy all the sketches you write and any other answers required for the assignment into the submission text file. *Name the file "108_A4_firstname_surname.txt", using your actual name. Put your name and assignment number at the top of the submission file also. Clearly label everything in the file. Submissions without names or clearly labelled sections will have to be marked down or (in the worst case) not marked at all.*
4. Upload your sketches in a zip file, using the same naming convention as the text file, except using a .zip suffix.

Marking for lab/assignment

Most of the assignment will be marked in the following lab session. It is essential that you demonstrate all sketches you have completed or partially completed.

For all code sections, marks will be deducted for bad structure, communication and style (e.g. repetitive or highly inefficient code, inappropriate global variables, missing or mismatching comments, poor variable names, bad indentation, etc.), incorrect behaviour, and failure to compile.

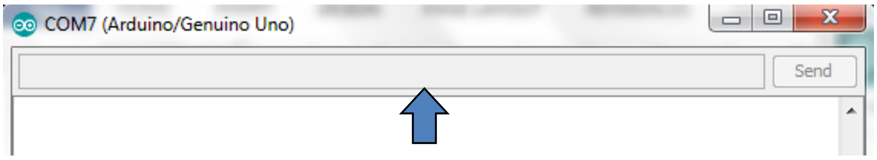
General marks will also be lost if the submission document requirements are not followed

NOTE: plagiarism and collusion will not be tolerated. You may be interviewed in detail on any work submitted and any evidence that the work is not your own will have the appropriate consequences.

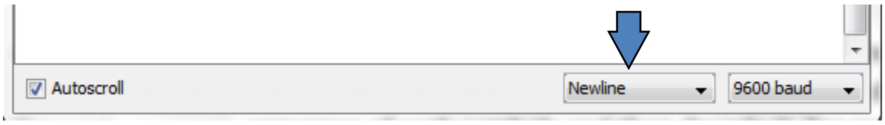
1 Read characters from serial input – save sketch as “A4_AverageDirection”

Background: This exercise uses concepts introduced in the notes “2.22 C strings and pointer const correctness”, “2.30 advanced types”

To input text via serial input we use the edit box at the top of the console window.



Normally, no characters are sent to the Arduino until you click the send button. We would prefer to send characters as soon as you press the enter key. To allow this we must change the line ending mode of the console window from “no line ending” to “newline” as in the following figure.



☑ Autoscroll Newline 9600 baud

Please review the SerialEcho example sketch as this will be useful to your solution. It demonstrates demonstrates the process of reading characters from serial input. However, the only thing it does with those characters is echo them back to serial output. It uses the function `Serial.available` and `Serial.read`. Note that it demonstrates concepts only and you will need to make various modifications for your solution.

The purpose of this assignment is to read direction keys (for which we’ll use J, K, L, and I), calculate the resultant direction of the most recent 8 keys, and match that to a one of a limited set of compass directions.

You will also need to develop and use a FIFO buffer/queue. You have already created and used a FIFO buffer as part of lab 6.

A4_AverageDirection requirements:

- Read characters from serial input until a space or newline is read and echo them
- Identify direction characters in the serial input. The characters J, K, L, and I (uppercase or lowercase) represent directions (left, down, right, and up respectively).
- Use a FIFO buffer to keep track of the 10 most recently entered direction characters. Ignore other characters. The buffer keeps is never emptied or cleared (even after newline and processing) and must remember its contents. Instead, each new input will simply add to the contents of the buffer. Once 10 characters have been added, each new addition causes the oldest (first in) character to be removed. We will call each direction character in the buffer a “direction input”

- Map from each direction input to a unit vector whose x and y coordinates can only be the integers -1, 0, or 1. For example L (right) is represented by the vector x=1, y=0 and K (down) by the vector x=0, y=-1. (You might need a struct for this, depending on your implementation)
- Calculate the resultant vector of all the direction inputs in the FIFO buffer, but only the inputs in the FIFO buffer. Just sum the components of the unit vectors corresponding to each direction input in the buffer.
  IMPORTANT NOTE: If you do not use a FIFO buffer and just try to sum up all inputs you receive before adding them to the buffer, your answer will not be correct as it may include less than or more than the most recent 10 inputs.
- Calculate the slope using fixed point math. Unlike floating point, fixed point basically uses integral values that have been scaled by a fixed amount. E.g. in integral math, 1/2 = 0, but 1*1000/2 = 500 and we know the answer is scaled by 1000 (so 500 should be interpreted as meaning 0.5). Use this technique to calculate the slope scaled by 1000 (see output below).
- Then determine the nearest matching compass direction from among the following: North, South, East, West, NorthEast, SouthEast, NorthWest, SouthWest.

  Hint: Avoid floating point math. Avoid very repetitive code or very large sets of if-else statements. The correct solution will only return the closest compass direction, e.g. "LLLII" would have a resultant (x=3,y=2) and would be closest to NorthEast whereas "LLLLI" would have a resultant (x=4,y=1) and would be closest to East. It isn't sufficient to simply check if x > y or vice versa. Consider the angles that distinguish between being closer to one direction than another. You could check against the corresponding slopes.
- Ensure that you print the contents of the buffer (and only the buffer contents – not any extra inputs), the resultant vector (x and y), the slope of the resultant vector (scaled by 1000), and the nearest matching compass direction. The output should appear something like

```
Type direction characters: J, K, L, or I (upper or lower case) and hit
return

  Received: 1234fghllksd
  Buffer contents: llk
  resultant: x=2, y=-1
  direction: RHS, 1000*slope=-500, compass=SouthEast

Type direction characters: J, K, L, or I (upper or lower case) and hit
return

  Received: iijkija
  Buffer contents: lkiijkij
  resultant: x=-1, y=1
  direction: LHS, 1000*slope=-1000, compass=NorthWest
```

✏ Copy the sketch into your answer document.

2 Reflection

Q1: What was the most important thing you learned in the assignment?

Q2: What was the hardest aspect of this assignment and why?