Objectives: This laboratory aims to practice using chars, strings, and advanced types (mainly struct) in C (with the Arduino platform).

Learning outcomes:

- Declare and use strings
- Define and use structure types

Lab instructions

1. The lab is for individuals working alone and must be completed during the lab session
2. Create a new sketch for each major section in the lab.
3. Before you leave the lab, call the lab demonstrator to check what you have done for all the sections (this is why separate sketches are important). *Anything not checked by the demonstrator during lab will have to be assigned zero marks.*
4. Create a single plain text submission file (.txt) for the lab using a plain text editor (e.g. NotePad++) and submit only this text file. Copy all the sketches you write and any other answers required for the lab into the submission text file. *Name the file "108_Lab8_firstname_surname.txt", using your actual name. Include your name and lab number at the top of the submission file also and clearly label everything in the file. Unclear submissions will have to be marked down or (in the worst case) not marked at all.*

Marking for lab/assignment

Most of the lab and assignment will be marked during lab sessions. It is essential that you get the demonstrator to confirm your progress during the lab (for the lab portion) and at the start of the next lab (for the assignment portion).

For all code sections, marks will be deducted for bad communication and style (e.g. missing or mismatching comments, poor variable names, bad indentation, inappropriate use of global variables, unnecessary code repetition, etc.) and incorrect behaviour, or failure to follow the requirements of the question.

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

1 Basic Struct – save sketch as “Lab8_StructLineSlope”

Background: The purpose of the second exercise is to practice using structure variables and accessing structure fields. Refer to notes “2.30 Advanced Types” for more information about structures. You will be accessing structure variable members directly (using the dot notation) and indirectly via a pointer (using the arrow notation).

Lab8_StructLineSlope requirements:

- Download the starter sketch, Lab8_StructLineSlopeStarter if you have not already done so and extract it into your sketchbook. Save the starter code as the correct sketch name for part 1 of the lab. This starter code defines a new structure type called struct Point2d which represents a point in 2 dimensional space using Cartesian coordinates.
- Add a typedef so that you can use the type name `Point2d` as an alias for `struct Point2d`. Use this typedef’d alias throughout the rest of your programme.
- In the loop function declare two variables, `point1` and `point2`, whose type is `Point2d`. (The variables will need remember their values from one function call to the next).
    - point1 represents the start point of a line segment and should be initialized so that its x and y members are 1500 and 500 respectively.
    - point2 represents the end point of the line segment and should be initialized so that both its members are zero.
- Whenever SW1 is normally clicked, print the current values of start and end point structure variables, calculate the slope of the line segment and print that also
    - Define a function to print a single Point2d and call this to print each point. Remember to pass in the struct to be printed using a pointer (to avoid unnecessary copies).
    - Define a function to calculate the slope given two points. Remember to pass in the structs using appropriate pointers. Print the returned slope value – don’t print inside the function.
    - The calculated slope should be a floating point number and you will probably need to use a float cast in the expression to ensure that the result is not integer truncated
    - Be careful not to divide by zero in the slope calculation (infinite slope) – check it first and return 99999.99 instead in this case
- Whenever SW1 is long clicked
    - Directly inside the loop function (i.e. without calling a helper function), replace both the point2 variable’s x and y member values with new random values between -9999 and 9999 inclusive
    - Print the new value of point2
- The output should look something like

```
[click] point1 {x=1500, y=500}, point2 {x=0, y=0}, slope=0.333
[long click] new point2 {x=2000, y=1000}
[click] point1 {x=1500, y=500}, point2 {x=2000, y=1000}, slope=1
```

✎ Copy the sketch into your answer document and demonstrate it.

2 Basic string manipulation – save sketch as "Lab8_PasswordGen"

Background: This exercise is based on notes "2.22 String Pointers and Const". You will develop a password generator that can create a new random alphabetic word whenever a button is long-clicked and save it as the current password. Whenever the button is normally clicked the programme will print the current password without changing it.

To solve the exercise you will manipulate a modifiable string in an array and perform some basic arithmetic with char type variables or array elements and ASCII codes.

Lab8_PasswordGen requirements:

- First download the starter code, Lab8_PasswordGenStarter from moodle and extract it into your sketchbook. Save the starter code as the correct sketch name for part 1 of the lab.
- Declare a local string (character array) variable inside the loop function that can hold passwords with up to 10 characters (excluding the nul terminator).
  - Initialize the array so that it initially contains the string: PASSWORD
  - Ensure the array remembers its value from one execution of the loop function to the next
- Whenever SW1 is clicked, print the current value of the password (as stored in the password array). Because a string in C is just an array of characters that includes the nul terminator character you can just call Serial.print to print the contents of the password array as a string.
- Whenever SW1 is long clicked:
  - Choose a random number, `len`, between 4 and the maximum password length (inclusive). This will be the new password length.
  - Iterate over the first `len` characters of the password array, replacing them with random alphabetic characters between 'A' and 'Z' inclusive.

    Hint: you can use arithmetic with characters. For example, 'A' + 7 is 'H'. You can use this to convert from a random number to a random letter.
  - Insert the nul terminator, `'\0'`, at the array element just after the last valid character in the new password to make sure it is a valid terminated string
  - Finally print the string length and new contents of the password string
- The program output should be something like:

```
[SW1 Click] current password is: PASSWORD
[SW1 Long Click] new password (len = 4): RFKQ
[SW1 Click] current password is: RFKQ
```

Reminder: Put a comment at the top of each sketch as in lab 1. Do this for every sketch in the lab. Don't forget to use good coding practice, naming conventions, etc.

Copy the sketch into your answer document and demonstrate it.

3 Extra credit – save sketch as "Lab8_PasswordGenExtra"

Modify the sketch in part 2 so that the password may contain uppercase and lowercase letters, at least 1 number, and 1 punctuation symbol. The positions of the different character types in the string must not be fixed. It must be possible for the character types to appear in any order, i.e. it must be equally likely that the punctuation character appears in any legal position in the string.

✏ Copy the sketch into your answer document and demonstrate it.