Module	CS4012 – Representation and Modelling
Day	11
Lab	8
Topic	Arrays

Summary

(Data structure): A specific way of organizing data in a computer for efficient use.

An “array” is a type of data structure that groups together values called elements into a contiguous block of memory. This memory is divided into indexed slots that each holds a value. In Lua, arrays are represented using the “table” data structure. The elements of such an array can be of any data type, and can be indexed with any data value except “nil”. The following examines conventional arrays which are indexed numerically.

Exercise 1: Creating and populating arrays

In Lua, arrays are created and initialized using constructors. The most basic constructor, which creates an empty array, is represented by a pair of curly brackets. For example, the following code snippet creates an empty array and populates it with string values:

```
seasons  = { }                -- creates an empty array
seasons[1] =  “spring”        -- adds the value “spring” at index 1
seasons[2] =  “summer”        -- adds the value “summer” at index 2 …
seasons[3] =  “winter”
seasons[4] =  “autumn”
```

Notice that the empty array grows as necessary to accommodate additional elements. Adding a comma separated list of values between these curly brackets initializes an array on creation. The same array as “seasons” can be created in one line using the so-called list constructor:

```
seasons = {“spring”, “summer”, “winter”, “autumn”}
```

Tasks:

1. Write a program that creates an array initialized with the days of the week as string values, and then outputs the contents of the array.

2. Write a program that initializes an array with this week’s lotto numbers: 6, 7, 15, 17, 20, 45, and the bonus number 19. The program should then output winning numbers as follows:

“The winning numbers were, 6, 7, 15, 17, 20, 45, and the bonus number was 19”

3. Write a program that creates an array containing the first ten square numbers and then outputs the contents of the array. The output should look something like:

“1, 4, 9, 16, 25, 36, 49, 64, 81, 100”

Hint: Consider using a loop to go through an array. The length operator, “#” can be used to find the length of an array e.g. for the array “seasons” above, the expression “#seasons” would evaluate to 4.

Exercise 2: Copying an array

Arrays are “objects” in memory. A variable to which an array has been assigned holds its address in memory, not its value. Assigning the same array to two different variables does not result in a new copy:

```
original = {5, 10, 15, 10, 25}

copy = original              -- “copy” points to the same table as ”original”.
```

Tasks:

1. Starting with the code fragment above, change the first element of the array “copy” and output the contents “original”. What can you conclude from the output?

2. Write a program that copies the array “original” to another array called “copy”. Change the first element of “copy” to the number “73”, and output the contents of both “original” and “copy”. The output should show “original” unchanged by the modification to “copy”.

Exercise 3: Counting and searching

Grouping data together into one place can make finding and counting data much easier. Any element in array can be accessed simply by shifting the index and pointing to that value. For this reason arrays are tremendously useful for the searching and sorting of data.

Tasks:

Given the following array of numbers:

```
numbers = { 8, 58, 18, 33, 59, 42, 48, 93, 22, 40}
```

1. Write a program that counts the number of values smaller than the value 33.

Hint: Use a count variable to hold the number of matches.

2. Write a program that finds the smallest value in the array.

Hint: Pick a starting candidate index. Compare the value at the candidate index with remaining elements. Update the candidate index if the element being compared with is smaller.