COMP1511 18s2

Week-04 Tutorial Exercises

The tutorial will start with a code review.
Your tutor has asked a lab pair to present their week 3 work.

Discuss the good, the bad and the ugly aspects of their code.

Please be gentle in any criticism - we are all learning!

1. What is an array?

2. Give an expression that sums the first and third element of an array called `numbers`

3. If an array is declared as `int numbers[20];` and your program assigns a value to each element in the array, what is the problem with the statement `x = numbers[20];` ?

4. How would you declare a variable **squares** to be an array of integers with 15 elements?
   Write a C code fragment to store, in each element of this array, the square of the index of that element, e.g., **squares[5]** would contain the value 25.

5. Write a C function that calculates the dot-product of two vectors of integers. Your function should use the following prototype

   ```
   int dot_product(int vector1[VECTOR_LENGTH], int vector2[VECTOR_LENGTH])
   ```

   Reminder: you calculate the dot product by multiplying corresponding elements and summing the result.

   So the dot-product of `1 3 1 3 2` and `2 1 2 1 2` is `14`.

6. Write a C function that returns 1 if an array of ints is in increasing order, 0 otherwise,
   It should have this prototype:

   ```
   int increasing(double array[], int length)
   ```

---

7. `struct`s allow us to *combine* multiple variables and bundle them together. One example where we may want to do this is where we have values that have multiple *components*.

   One example of *values* that have multiple *components* are `point`s in 2D or 3D space.

   ```
   typedef struct _point2d {
       double x;
       double y;
   } point2d;

   typedef struct _point3d {
       double x;
       double y;
       double z;
   } point2d;
   ```

   Another example is **Complex Numbers**. If you are not familiar with *complex numbers* , watch the videos available on the lab activity page for complexAbs. Complex numbers have a *real* and *imaginary* component. Mathematical operations on *complex numbers* work differently, due to how they are defined.

   Walk through the complexAbs which implements the mathematical `abs` function for complex types.

8. BMP Images: We can represent images in many *file formats*. One common format that used to be popular was **BMP**. Please read the following "COMP1511 in Colour!"

   - COMP1511 in Colour!

   For a computer to be able to display an image, it has to know the colour for each pixel in the image as well as other information about how the pixels are arranged.

   The information about the image is called the *metadata* and contains information such as

   - how wide the image is,
   - how high the image is,
   - how many pixels there are, and
   - where in the file to find the pixels.

   The pixel colour information is stored as a 2-dimensional array in the file. Colour is represented as a mix of red, blue, and green light. Each pixel represents its colour by representing the amount of red, green, and blue light to use by specifying a value between 0 (no light) and 255 (maximum light) and uses a single byte for each value.

`struct`s, like all other types, can be placed into arrays.

Walk through the Chessboard activity during your tutorial.

Revision questions

The remaining tutorial questions are primarily intended for revision - either this week or later in session.
Your tutor may still choose to cover some of the questions time permitting.

9. Write a C program `occur.c` which reads 6 numbers then reads another number and prints how many times that number occurred in the first 6. For example:

```
$ ./occur
Enter 6 numbers: 1 3 1 3 1 9
Enter a number: 1
1 occurred 3 times in the 6 numbers read
```

Make sure you make you make it very easy to change how many numbers the program reads.

10. Write a C program which reads numbers until end-of-input is reached and then prints the middle number.
So if 9 numbers were entered the program should print the 5th number entered. And if 27 numbers were entered the program should print the 14th number entered.

If an even number of numbers is entered there are two middle numbers, print the second of them.

For example if 8 numbers are entered, prints the 5th.

You can assume at most 10,000 numbers will be entered.

11. Write a C function that takes in an array of integers, and returns the difference between the smallest and largest numbers in it.

```
double max_difference(int numbers[], int num_elements);
```