Arrays

- An array is very much like a matrix.
- In the C language, an array is a collection of variables, **all of the same type** (e.g. char, int, float, double).
- Let's look at a 1-D array as a simple example.
- Imagine that I have six children (I really only have four, but I am counting my two dogs) and I wish to store their ages in a computer program, to print out when necessary (in case I get Alzheimer's or something!)
Declaring Arrays

- In this case I will declare the array, and also assign the values to each element.
- How do we declare an array? Almost just like we declare any single variable, but with an important difference.
- If I want to declare an int variable, I use:
  - int age
- If I want to declare an array I use
  - int age[6]
- The square brackets indicate that "age" is an array.
- The six indicates that the array has six values.

Addressing Arrays

- Arrays have an index variable, which starts at 0, and goes to the (maximum number of elements –1)
- e.g. int my_array[5] has members
  - my_array[0] to my_array[4]
- To print out the first member use:
  - Printf("Here is the first value in the array, %d \m", my_array[0] );
Reading Data into an Array

- Very like reading data into any other variable, but the addressing is a bit different
- Recall that the statement
  - `scanf("%d", &my_variable);`
- will read whatever is typed in at the keyboard into the variable `my_variable`, which in this case must be declared to be an integer. The & is the operator that returns the address of `my_variable`
- The array is similar, but you don’t need the & as we always refer to an array by its address: `my_array[0]` is the address of the first element.

Arrays of Strings

- Recall that in C there is no special string type. Instead we declare an array of char variables:
  - `char my_variable[20];` is a variable which can hold up to 20 characters.
- For an array, you need a second specification, which is the number of 20 character strings you want to store:
  - `char my_variable[20][10];` is an array which can hold up to 10 strings of 20 characters each.
The IF statement

- The IF statement is great for comparing things. To use it you need to know the C language comparison operators.
- These are
  - `==` "is equal to" (note this is different to the assignment `=`, where you place a value in a variable or do a mathematical operation: e.g. `number = 5`)
  - `<` "less than"
  - `>` "greater then"
  - `<=` "less than or equal to"
  - `>=` "greater then or equal to"
  - `!=` "not equal to"

The IF statement has the following syntax:

- `if(compare one thing with another)`
- `{`
  - `Do these things if comparison is true;`
- `}`

- notice that there is no semicolon at the end of the brackets of the IF statement.
The break keyword

- break is a very handy keyword. The break keyword allows you to get out of any C loop (e.g. for or while), which can be useful if you have accidentally sent the computer into an infinite loop. Of course, you can always get out by pressing control-C, but that terminates the whole program.