Session 2 - The Cequel

Welcome to the Cequel!

See what I did there

I hope you enjoyed Session 1 and are now comfortable with using variables, for loops, and if statements to create very basic C programs (including commenting and being able to compile your code), and are familiar with undefined behaviour and arrays - we can now start to explore

It's getting to the harder stuff ...

Good luck and happy coding!

- CHANGE THE README

Contents

Pointers

We are about to learn arguably the most fundamental part of C: pointers. They may seem scary but we'll soon see that they're not *actually* that bad!

Memory Addresses: How Variables Are Stored

We know that variables store values for us; but what does this actually look like, in the memory of our computers?

Let's think about the following variable initialisations:

```c
char a = 5;
char b = 14;
```

Memory Address	Value Stored
6422039	5
6422040	14

We can see that the variable **a** has been stored at 6422039, and that variable **b** has been stored at 6422040 (the next **memory address**). These **memory addresses** tell you where you'd find the variables in all of the RAM your computer is currently using so they tend to be fairly large values.

Memory addresses are actually in **binary** (which uses only 2 symbols: `0, 1`) and converting to **denary** (our numbering system which uses 10 symbols: `0, 1, 2, 3, 4, 5, 6, 7, 8, 9`) is a bit of a pain, so we don't usually represent addresses with denary. However, because of the large values that memory addresses tend to be, we can't exactly read them in their binary form since you get values like this: `11000011111111000010111` (`6422039` in binary).

Therefore, we typically represent addresses using **hexadecimal** (which uses 16 symbols: `0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F`) which is both more compact than binary and, unlike denary, can be easily converted to and from binary.

- Mention different types need different n bytes
- May have HUGE amount of memory; what type is a memory address?
- The NULL pointer (for failed one)

Functions

Next Session...

Well done on completing Session 2! You're almost through to the end; just one more session to go...

If you're not tired of C by then, then remember there's a bonus session on (INSERT DATE)

Optional Exercises

Not sure what to put here yet

Acknowledgements

Thanks to ...
Originally created by Edward Denton.