Lecture 1_C

Terminal Commands

1. Basic commands to create a c file and compile it

```
code hello.c
// creates a c file (only works in VScode)
make hello
// compiles the file and creates an executable file named hello
// note that no .extension is needed for the executable file
./hello
// execute the file hello
// dot . indicates the current directory
// the slash / separates the current directory and the file name
```

2. `make`

```
    make file_name
    // make finds file_name.c and compiles it to executable file file_name
```

Header Files

1. `Library`
    - A *library* is a collection of code created by someone. Libraries are collections of pre-written code and functions that others have written in the past that we can utilize in our code.
    - To use the library, we need to use the command

```
#include <stdio.h>
```

```
This command tells the compiler that you want to use the capabilities of
a library called stdio.h, a header file. This allows you to utilize
the printf function.
```

Basic C knowledge

1. Function

```c
void meow(void)
{
    printf("meow\n");
}
```

The initial `void` means that the function does not return any values. The `(void)` means that no values are being provided to the function.
2. For counters, the convention is to count from 0 to < target using for loop

```c
    for (int i = 0; i < 3; i++)
    {
        printf("meow\n");
    }
```

3. Escape characters

```
\n  create a new line
\r  return to the start of a line
\"  print a double quote
\`  print a single quote
\\  print a backslash
```

4. Single quotes are utilized for single characters.

```c
    if (c == 'y')
    // when refering to a single char, use ''
    {
        printf("Agreed.\n");
    }
```

5. Prototype
   When calling a funtion A within funtion B, and A is defined below B, we need to provide a prototype at the beginning of the code(It is also a good practice to use another head file)
6. Constant

```c
    const int n = 3;
    // n is now a constant. It can never be changed.
    // This prevents malicious changes of n
```

7. Abstraction
   To write more elegant code, it is always better to write each small step as a function and call functions within other functions instead of writing redundant code.
8. Types
   - `bool` , a Boolean expression of either true or false
   - `char` , a single character like a or 2
   - `double` , a floating-point value with more digits than a float
   - `float` , a floating-point value, or a real number with a decimal value
   - `int` , integers up to a certain size, or number of bits
   - `long` , integers with more bits, so they can count higher than an int
   - `string` , a string of characters
9. Truncation: throw away any digits after the decimal because an integer divided by an integer will always result in an integer in C