L2&3 - Fundamental Concepts

November 1, 2019

1 Introduction to C: Fundamental Concepts

1.0.1 PASS

Sean Kirkby - SA402 Tuesday 6:00 to 7:30pm and PA218 Thursday 6:00 - 7:30, office hours Monday 3:30-4:30 in the 4th floor of the library

1.0.2 Midterm

Saturday November 2 2019, 1:00-2:30 (tentative)

1.0.3 Lab Schedule

Attendance at lab session is not mandatory but may be the difference between *marginal* and *unsat*. Do not attend lab sections that are not your own.

1.1 Hello World in C

In [1]:
```
/* Hello World in C */

        #include <stdio.h>
        #include <stdlib.h>

        int main(void){
            printf("Hello World\n");
            return EXIT_SUCCESS; /* */
        }
```

```
Hello World
```

Let's unpack that a bit **stdio.h** - contains I/O functions like *printf*
EXIT_SUCCESS - a reserved word, defined in **stdlib.h**. To indicate failure, return **EXIT_FAILURE**.

Every standard C program has exactly one function named *main*, which is the program's entry point. Passing *void* means that the function has no *parameters*; that is, the function takes no other information to run.

printf - send *formatted* output to the standard output stream (the console)

The returned value from *main* is the program's exit status; in this case, a macro from stdlib.

1.2 Fahrenheit to Celsius

In [1]:
```
#include <stdio.h>
#include <stdlib.h>

int main(void){
    int lower, upper, step;
    float fahr, cels;

    // Set up the iteration limits and step size
    lower = -100;
    upper = 220;
    step = 20;
    fahr = lower;

    while (fahr <= upper){
        cels = (fahr - 32) * 5. / 9.;
        printf("%4.0f %6.1f \n", fahr, cels); // note float formatting
        fahr = fahr + step;
    }
}
```

```
-100  -73.3
 -80  -62.2
 -60  -51.1
 -40  -40.0
 -20  -28.9
   0  -17.8
  20   -6.7
  40    4.4
  60   15.6
  80   26.7
 100   37.8
 120   48.9
 140   60.0
 160   71.1
 180   82.2
 200   93.3
 220  104.4
```

Let's unpack *that* a bit **=** - the *assignment* operator. The expression on the right hand side is stored into the variable on the left hand side.
- - subtraction
/ - division
** * ** - multiplication
Parentheses operate algebraically; code inside the parentheses executes first.