Writing your own Functions

Modularization

We have seen functions

- printf
- scanf
- getchar
- sqrt
- The C Libraries have 100’s of functions

Why have functions?

- Carry out repetitive tasks.
- Create a logical structure to your program.
- Easier to debug, get one piece of code working in isolation of the rest of the code.
- Divide the work among people.
- General purpose functions can be used by anyone.

Example of Organizing Code.

```
int main()     {
   getUserInput();
   processData();
   displayOutput();
}
```

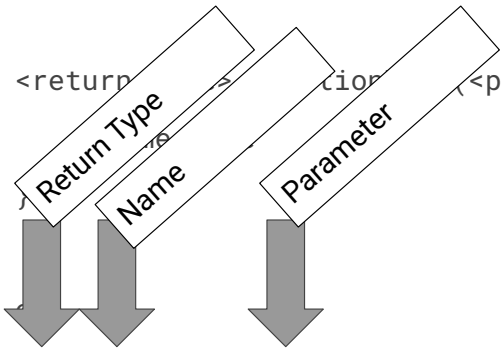
This would be a perfectly legit main function.

What every function looks like

```
<return type> functionName(<parameter(s)>) {
    //Some code
}

ex:
int sqrt(double inputVal) {
    //Fancy math
    return answer;
}
```

What every function looks like




```
int sqrt(double inputVal) {
    //Fancy math
    return answer;
}
```

Every program has at least one

- main is just a function that we start the program with.
- All functions are written the same way, with a type (e.g. int) and perhaps some parameters to pass information back and forth.
- Note: There is void type if you don't want the program to return anything.

Simple function

```
void putPrompt() {
    printf("Press any key to end program");
}
```

- Functions like this can still be useful to isolate repetitive code.
- Functions like this can still be useful to isolate repetitive code.

Function that returns something

```
int diceRoll() {
    return(rand() % 6 + 1)
}
```

- Still useful. Can make program more readable and isolate repetitive code.
- In this “diceRoll” has a value, which is returned to the main program
- e.g. while (diceRoll() + diceRoll() != 7)

return()

- Just like we use return(0); in main, we use the return keyword to return values to the calling program.
- The compiler will warn you if you are not returning a value and the function isn’t void.

Prototypes

- I like to see the main() function FIRST in a program with the functions declared BELOW it.
- However, the compiler needs to recognize the function before you use it.
- So that the compiler knows what we are talking about when it sees a reference to a function in main, we mention the function above main, followed by a ; instead of the code. This is called prototyping.

Example

```
#include <stdio.h>
//prototypes
int diceRoll();

// main function
int main() {
      Code for main ;
}

// functions
int diceRoll() {
    code for dice roll;
}
```