Functions in C

As you become more competent in programming, and your programs become more challenging, there emerges a need to be able to organize your programs in a more systematic manner which can both aid development of code as well as the reuse of code. One tool available in all major languages is the ability to write code in smaller blocks which are used to build up the overall program. Generically, these blocks of code can be referred to as subroutines, although depending on the language, they may be considered to be *methods* or *functions*. You have already seen how C uses functions, as every C program must contain the **main()** function in order to be run. Other operations in C, from the printf() and scanf() to rand() or pow(), are built in routines that make the job of you, the programmer, easier.

What Can Functions do For You?

Initially, when your programs are very small, it is difficult to see how functions can be anything more than a "make work" task. They can seem to create more work than they are worth. However, as your programs grow in complexity you will find that it becomes more and more difficult to keep track of what is happening in your code. When your program grows beyond a screen of code, even something as simple as keeping track of brackets becomes more frustrating. Also, once some aspect of your program is working properly, it is nice to not have to worry about that block, and instead focus of new code being added. In both these situations function use can be very helpful. This is not to say that using functions is trivial, but a bit of forethought in your program development can be extremely helpful in being able to use functions effectively. Another important aspect of functions is that if they are designed well, they can be very easily used in other programs which require similar operations.

Writing Functions

As with many aspects of programming, some parts of code are done for logic and some are done for style. It is important to remember that your need for your own style should be considered less important than the need to write clean, well structured code that can be easily read by other programmers. Some of the aspects of function creation will be style, so try to ensure you match the practices shown in the examples.

Functions in C always have three fundamental parts that make up their ***signature***. The function needs a **name**, which clearly indicates what it is going to do, an **argument list** which is the data it needs to run, and a **return value** which can be sent back to the point in code which called the function. This has been seen in all of your C programs so far:

int **main** (void)

Name: **main**

Argument list: **void** (meaning nothing is needed to start the program)

Return value: **int** corresponds to the **return 0** line at the end of code when it is done executing.