Ekaba Bisong
Programming in C++
University of Calabar

Lesson Note #19
May 20, 2015

Adapted from a handout by Eric Roberts – Stanford University & lecture slide by Geza Kovacs – MIT

Functions

The general form of a function definition looks essentially the same as it does in Java:

```
type  name (parameter list)  {
      statements in the function body
}
```

Where type indicates what type the function returns, name is the name of the function, and parameter list is a list of variable declarations used to hold the values of each argument.

All functions need to be declared before they are called by specifying a prototype consisting of the header line followed by a semicolon.

Why define your own functions?

- Readability: sqrt(5) is clearer than copy-pasting in an algorithm to compute the square root
- Maintainability: To change the algorithm, just change the function (vs changing it everywhere you ever used it)
- Code reuse: Lets other people use algorithms you've implemented

C++ Enhancements to Functions

Functions can be overloaded, which means that you can define several different functions with the same name as long as the correct version can be determined by looking at the number and types of the arguments. The pattern of arguments required for a particular function is called its signature.

Functions can specify optional parameters by including an initializer after the variable name. For example, the function prototype

```
void setValue (int value = 13);
```

Indicates that setValue takes an optional argument that defaults to 13.