An Introduction to Programming/C++ - Week 2

Ben Goska
Oregon State University

October 17, 2009

Why Functions?

Functions are useful for two reasons:

- Stop code copying
- Break code into simple chunks

Recall from math that standard functions look like $f : x \rightarrow y$, in C++ (or any programming language) the idea is similar. A function has a name, some input and some output. We can perform any operation in a function.

Defining Functions

A function definition looks something like:

```
output_type function_name(inputs...)
```

An example of this could look similar to our main function from hello world.

```
int main(void) {
    //things to do in here
    //...
    return 0;
}
```

This function returns an integer (as signified by 'int') is called 'main', and takes nothing as an input (ie 'void').

Example

Time to show some examples, try to follow along (the code can be found in the SVN if you want to review it later).

- Basic functions (w2BasicFunc.cpp)
- Scope example (w2Scope.cpp)
- Recursion (w2Recursion.cpp)

Header Files

It's important to remember that functions must be defined before being used. They can be defined in a seperate file, remember the include that we have done many times:

```
#include <iostream>
```

Now, if we want to define a function in another file (say myHeader.h) we would do:

```
#include "myHeader.h"
```

Lets look at an example. (w2Header.cpp).

Multiple Files

Now, the problem with header files is that if you include them into multiple cpp files that built into a single program the compiler gets very angry. This is because we defined something with the same name multiple times.
To fix this we will use another techinque. We can break a program into multiple cpp files and only define the actual functions. See the following example (w2Multi.cpp, w2MultiHeader.h, w2MultiHeader.cpp).

Defines

The preprocessor can do more for us than just include files. We can use it to name constants. This is can make code more readable, constants are defined in the following manner:

```
#define PI 3.1416
```

Then later we can just use the constant by name, such as:

```
std::cout << PI << std::endl;
```

Macros

Macros are a mix between constants and functions. They are useful for defining simple functions, there are a few differences between functions and macros, but they are fairly significant.

```
#define SQUARE_VALUE(x) (x*x)
```

And we can use it by:

```
std::cout << SQUARE_VALUE(10) << std::endl;
```

Preprocessor Magic

The preprocessor has a bit more magic in store for us, in fact we can use it to change code layout on compile. This is illistrated through the code in (w2Preprocessor.cpp).