Function Prototypes

- Like variables, functions must be declared before they are used
  - That's why we've been putting them at the top of the file, before `main`
  - This simultaneously *declares* and *defines* the function
  - In practice, we like to put `main` first, so we separate the function *declaration* from the function *definition*

Function Prototypes

- Functions are *declared* with a prototype
  - Looks just like the function heading as a statement (with ;)

```
double pow( double x, double y );
```

- As long as the declaration is before the function is used, you can put the *definition* anywhere
  - The definition is unchanged, still has heading and body
  - Convention is to put all function declarations together, followed by all function definitions (with `main` first)

Prototypes and Organization

```
// prototype
double get_side();

int main()
{
   double x;
   // call
   x = get_side();
   return 0;
}

// definition
double get_side()
{
   double input;
   cout << "Please enter a side of the triangle: ";
   cin >> input;
   return input;
}
```

Exercise 1

- Write the prototype for a function that takes in a string and a number *n* and returns the *nth* word in the string
  - Name: `nth_word`
  - Parameters: `1 string, 1 number`
  - Return value: `1 string`

Exercise 2

- Complete the code below:

```
string sentence = "They switched from the Swingline to the Boston
   stapler, but I kept my Swingline stapler because it didn't bind
   up as much, and I kept the staples for the Swingline stapler
   and it's not okay because if they take my stapler then I'll set
   the building on fire…";
string word;

// use the nth_word function to get the 14th word out of
//  sentence and store it in word
```

Using Multiple Files

- It can be convenient to organize code into more than one file
  - Particularly for reusing code (libraries)
- Standard libraries are included with:

```
#include <name> (e.g. iostream, string, cmath, etc)
```

- Files in a project are included with:

```
#include "name" (e.g. main.h, functions.cpp, etc)
```

- Including a file merely inserts that file in place of the include directive
- Useful for putting functions in their own file