int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
    // guaranteed to be random.
}
Functions

Functions are a useful way of grouping together code in a logical manner (also allows easier re-usability)

The returned value replaces the function where it was called/(used)

This means that when you use a function, your original function suspends and waits for the function to end before resuming
Announcements

HW1 due today

Office hours: Cheng Peng and Nishad Trivedi are no longer holding office hours

Quiz next Wednesday (will cover up to if/else)
Functions

(See maze.cpp)
Functions

It is important to note that the code will resume after the function call where it was used.

For example, \( \text{sqrt}(4) \) will return the value 2.0 where it was used and the rest of your code will continue.

Where does the maze code return to?
Functions

Multiple function uses/calls create a “stack” much like pancakes: every time you use a function, it will add another pancake.

When you return, the top pancake is removed.

main() is the bottom pancake.
Functions

How to make the person run?

RUN FOREST, RUN!!!

(See: runForest.cpp)
Functions

You can also use functions that return bool types in an if statement or loop.

This is commonly used if you have complex logic as it is normally easier to write a function that have a very complex bool expression.

(See: findPrime.cpp)
scope

(See: SillySwap.cpp)

Typically the value of variables is copied and not given access to the real value.

This is similar to moodle, the score you see for grades cannot change the score I give you!
Blocks (inside `{ }`) of code can only see variables from their parent blocks.

You can also make global variables outside of all blocks (almost as if your whole program has a start and end brace around it).

(See: globalVariable.cpp)
We will talk more about the difference between a variable's memory location and value later.

For now, a memory location (or reference) will give a function full access to modify the value.
You can give away your memory location by using “call by reference” with functions.

This will share the variable between the two functions, namely the function that is using the references (&) can modify the value.

(See: callByReference.cpp)
Memory

Memory:

Code:
Memory (declaration)

Memory:
#0 (int) x

Code:
int x;
Memory (initialization)

Memory:

#0 (int) x

2

Code:

```c
int x;
x = 2;
```
Memory (re-assign)

Memory:
#0 (int) x

9

Code:

```c
int x;
x = 2;
x = 9;
```
Memory (copy value)

Memory:

#0 (int) x
9

#1 (double) y
9.00 * 10^0

Code:

```c
int x;
x = 2;
x = 9;
double y = x;
```
References

When memory does not actually hold the value of an object, but instead holds information about the actual location...

... this is called a reference  (See: myInt.java)
If you use a normal function (call by value) then you will essentially make a photo copy of the variables

(makes 2 variables, does not effect other)

If you use call-by-reference, you have only one variable, but share it between you two

This is similar to a website link, if two people follow the link they end up in the same page
Debugging

- Test small pieces of code at a time
- Add cout statements to see values in loops (and to localize error in general)
- Test code on inputs you know the answer