Functions

1325-Object-Oriented Programming

Fadiah Qudah

University of Texas at Arlington

Lecture Overview

- Lecture
  - Functions (Foundations)
    - What is a Function?
      - Math
      - Programming
    - Motivation for Functions
    - Anatomy of a Function
    - Keeping our Functions
    - C++ Standard Library

LECTURE

What is a Function?

Motivation for Functions

Anatomy of a Function

Keeping our Functions

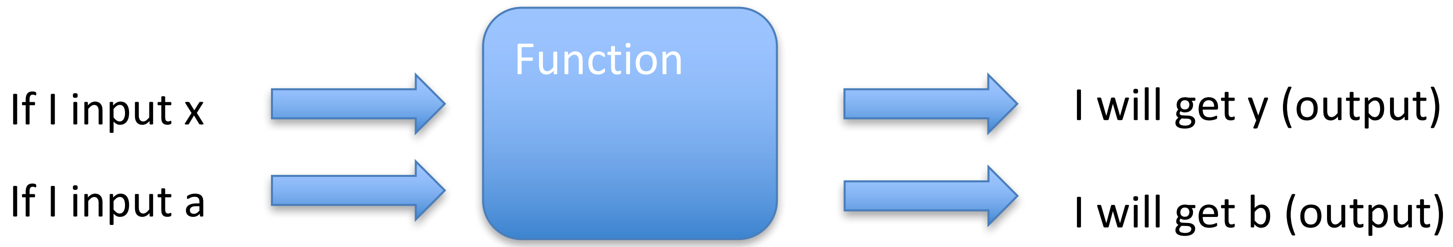
C++ Standard Library

What is a Function in Math?

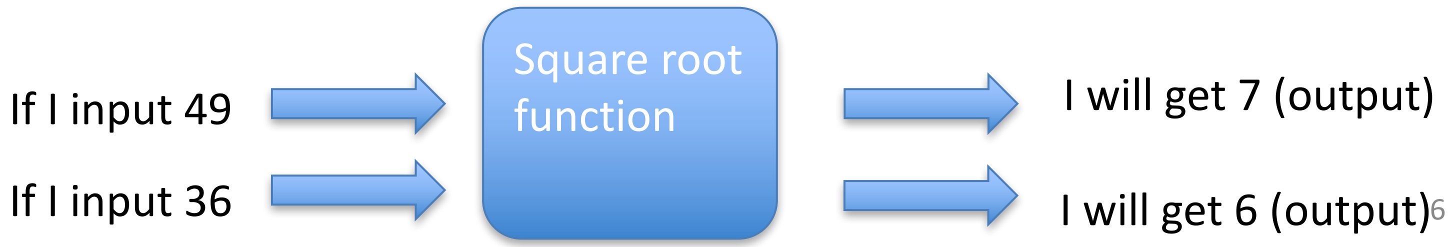
- Before we begin, I want to mention we will start learning about classes in the C++ language in the next lecture
  - We will be learning about making objects from classes and accessing functions that way

What is a Function in Math?

- In math, a function is the relationship between a set of inputs and outputs
  - Formal definition: http://mathworld.wolfram.com/Function.html
  - You can think of it like this:

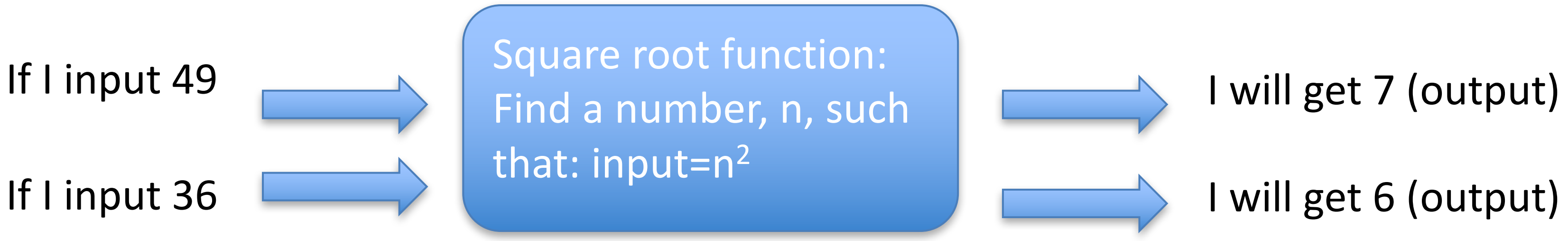


Example:



What is a Function in Math?

- When using a function, the input has certain actions done on it to produce the output
  - This simply means:
    - Use the input to help you arrive at the output



What is a Function in Math?

- You will hear phrases like:
  - y is a function of x
    - This simply means that the value of y relies on the value of x
      - **If x=9, then y=3. If x=16, y=4.**
  - z is a function of a and b
    - This simply means the value of z relies on the value of a and b
      - **If a=1 and b=2, then z=4. If a=2 and b=3, then z=6.**

What is a Function in Math?

- Remember this when we talk about function declarations in C++ later on in this lecture:

y is the output x is the input

- y is a function of x y=f(x)

z is the output a and b are inputs

- z is a function of a and b z=g(a, b)

English **Math equivalent**

What is a Function in Programming?

- In programming, a function is a collection of programming statements (each themselves doing a task) that are combined to perform a specific task
  - Sometimes you need to give an input (or inputs) to your function in order for it to work (called ***parameters***)
  - Your function can have an output which we signify with a ***return type*** (void means nothing is returned)
  - In 1310, using Java, we called this concept ***methods***

What is a Function?

Motivation for Functions

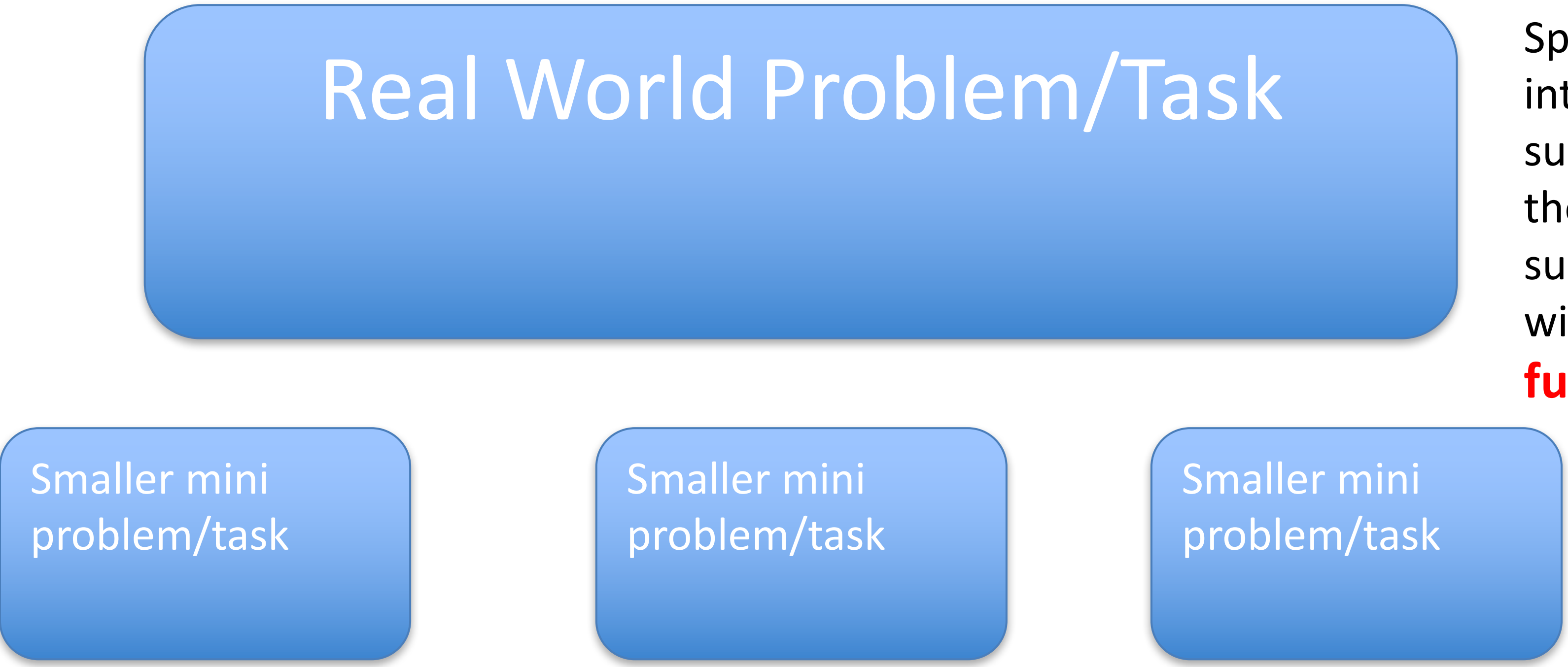
Anatomy of a Function

Keeping our Functions

C++ Standard Library

Motivation for Functions

- Why do we even have functions?
  - When not using an object-oriented paradigm (like when we use C), functions are used in the following way:


Real World Problem/Task
Smaller mini problem/task
Smaller mini problem/task
Smaller mini problem/task

Split the problem into smaller subtasks. Solve those smaller subtasks. These will be our **functions.**

Motivation for Functions

- When using the functions in the object-oriented paradigm, they will act as functionality for an object (we will discuss this more in depth next class)
- For example, if we want to represent a restaurant customer in a program:
  - We make a customer object
  - Give it the functionality of ordering (in the form of a function)
  - Don't worry about fully getting this concept right now- just introducing it

What is a Function?

Motivation for Functions

Anatomy of a Function

Keeping our Functions

C++ Standard Library

Anatomy of a Function

- Let’s discuss the general anatomy of a function
  - Declarations
  - Definitions
  - Parameters
  - Arguments
  - Return Types

```
#include <iostream>
```

Return Type-What type of value the function returns

```
int foo_function(int r1, int r2)
{
	int i=0;
	int answer=1;

	for(i=0;i<r2;i++)
	{
		answer=answer*r1;
	}

	return answer;
}
```

Parameter -part of the function definition. Think of them as reminders to the user that they need to actually give values when using the function.

A quick way to know if it is a parameter is whether or not a variable type (such as int) is present before a variable name.

```
int main (int argc, char **argv)
{
	int c=foo_function(3, 4);
}
```

Argument-when you actually use the function and put values in-
in this example, 3 and 4 are arguments for the first function call
5 and 3 are arguments in the second function call

```
#include <iostream>



int foo_function(int r1, int r2)
{
        int i=0;
        int answer=1;

        for(i=0;i<r2;i++)
        {
                answer=answer*r1;
        }

        return answer;
}
```

__Remember:__ main is also a function-it has a return type (int) and parameters.   Sometimes, you will see main written without a return type or parameters:

```
int main (int argc, char **argv)
{
        int c=foo_function(3, 4);
}
```

Anatomy of a Function

- Finally, notice how the function declaration is laid out the same way we saw with functions in math:

output

inputs

y=f(a, b)

type
of output
(return type)

inputs (parameters)

```
int function_name(int a, int b);
```

(Function declaration)

What is a Function?
Motivation for Functions
Anatomy of a Function
Keeping our Functions
C++ Standard Library

Keeping Our Functions

- We will be keeping our functions in something called a class (starting next lecture)
  - We will be using objects to access these functions
- We can keep classes libraries
  - We can include headers in our programs to access classes we made
- Note that we could also make individual functions (like what we did in C)
  - That would not be utilizing the object-oriented features available to us in the C++ language

What is a Function?

Motivation for Functions

Anatomy of a Function

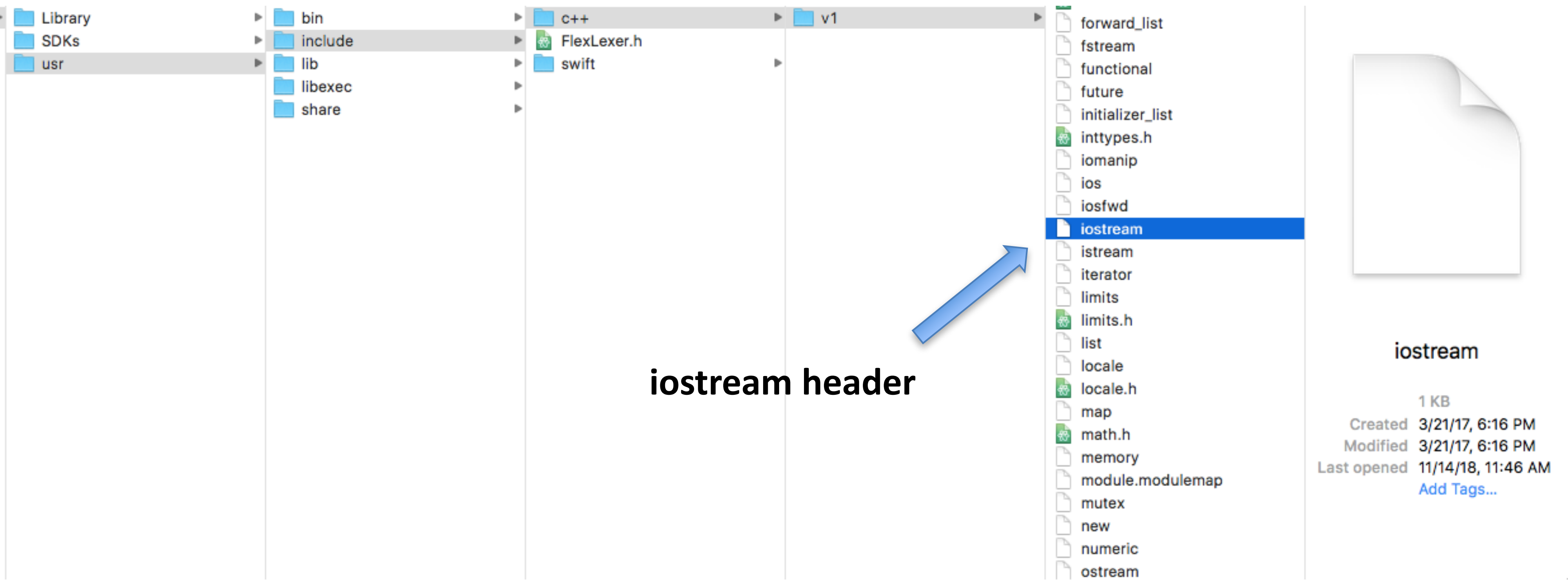
Keeping our Functions

C++ Standard Library

C++ Standard Library

- The C++ Standard Library is a collection of classes (we will learn about these next lecture) and functions
  - We can use it when programming (we don’t have to re-write code-someone already did it for us)
  - Notice the iostream header on the next page (what we use to do input and output)
  - Note there are other things in the library we can also use (I won’t mention it for now)

C++ Standard Library



Library
SDKs
usr

bin
include
lib
libexec
share

c++
FlexLexer.h
swift

v1

forward_list
fstream
functional
future
initializer_list
inttypes.h
iomanip
ios
iosfwd
iostream
istream
iterator
limits
limits.h
list
locale
locale.h
map
math.h
memory
module.modulemap
mutex
new
numeric
ostream

iostream

1 KB
Created 3/21/17, 6:16 PM
Modified 3/21/17, 6:16 PM
Last opened 11/14/18, 11:46 AM
Add Tags...

iostream header

C++ standard library
(screenshot from my computer)