Object Oriented Programming by C++

Functions (1/2)

Basic: Using and Writing Functions

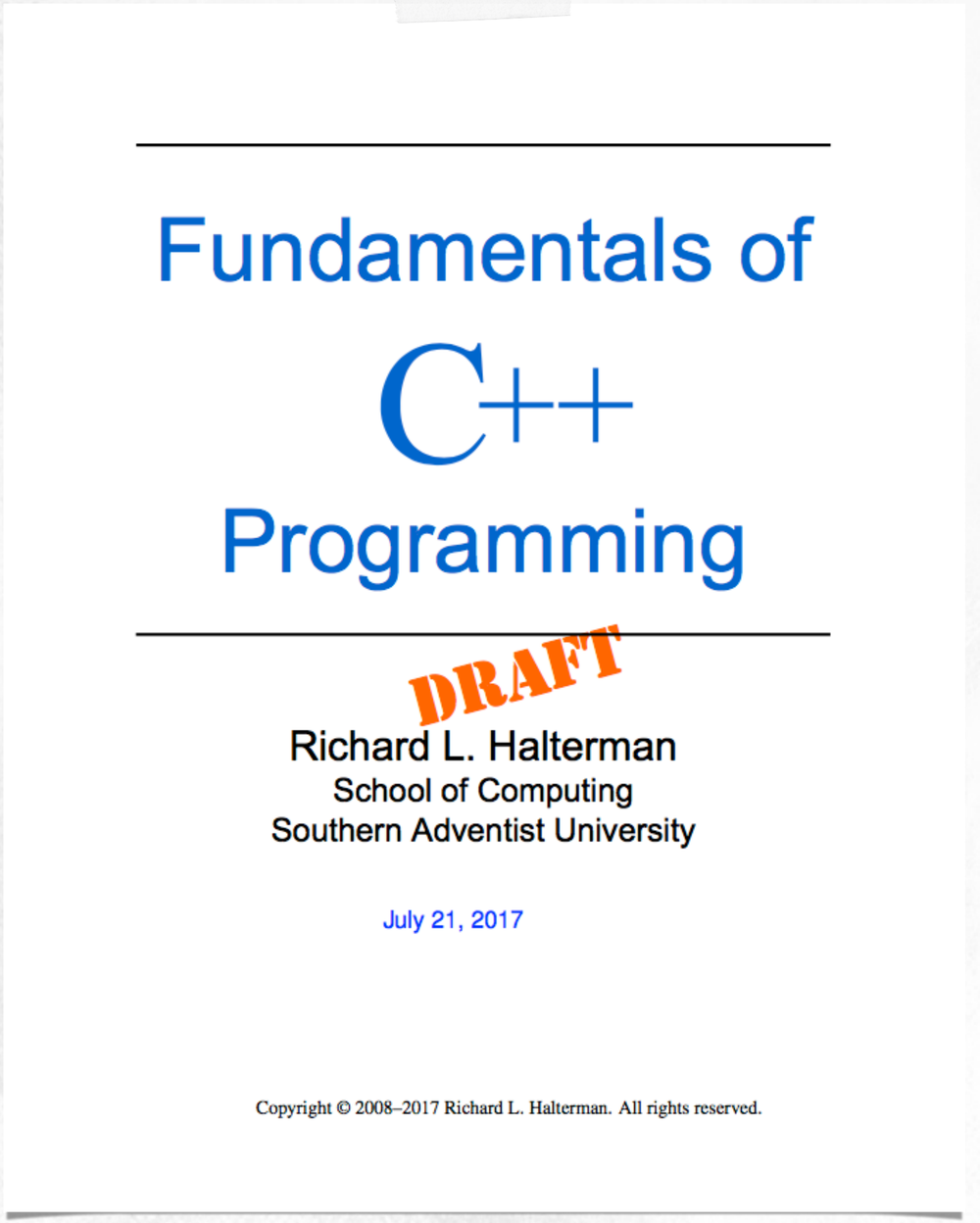
2017. 8.

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr
Web: http://mobilelab.khu.ac.kr/

Textbook & Copyright

- Textbook: http://python.cs.southern.edu/cppbook/progcpp.pdf
- Sample Codes: https://github.com/halterman/CppBook-SourceCode



Fundamentals of

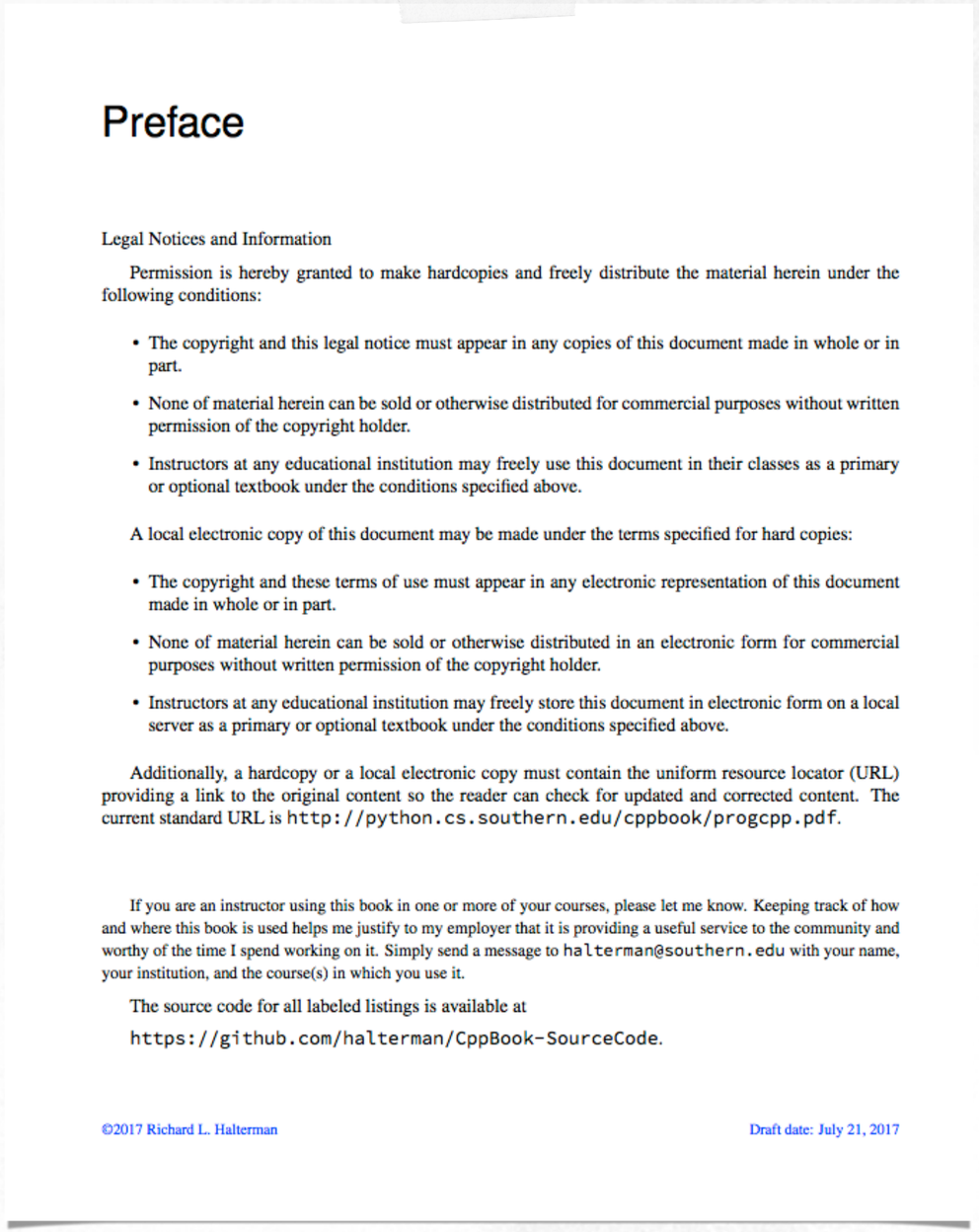
C++

Programming

DRAFT

Richard L. Halterman
School of Computing
Southern Adventist University

July 21, 2017



Preface

If you are an instructor using this book in one or more of your courses, please let me know. Keeping track of how and where this book is used helps me justify to my employer that it is providing a useful service to the community and worthy of the time I spend working on it. Simply send a message to `halterman@southern.edu` with your name, your institution, and the course(s) in which you use it.

The source code for all labeled listings is available at

`https://github.com/halterman/CppBook-SourceCode`.

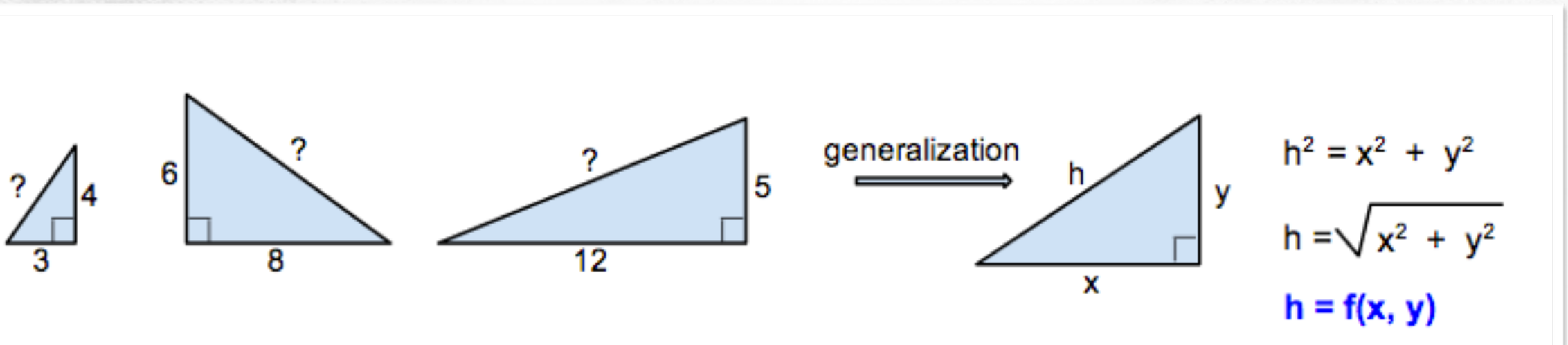
 Draft date: July 21, 2017

Contents

- Black-box Model
- Understanding Function
- Standard Functions and Libraries
- Writing Function

Black-box Model

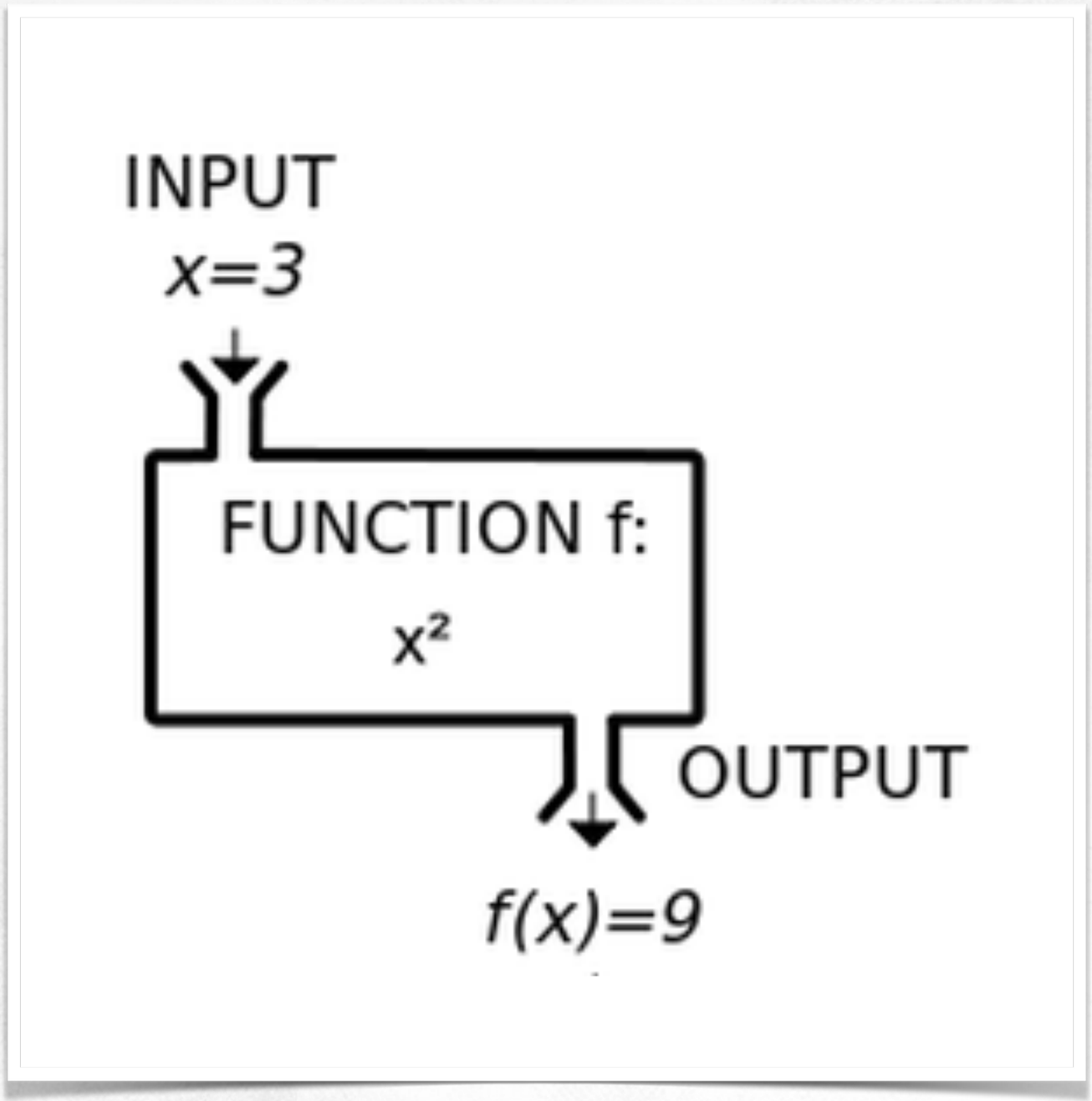
Functions in Math



Reference: http://cs.wellesley.edu/~cs115/readings/more-functions.html

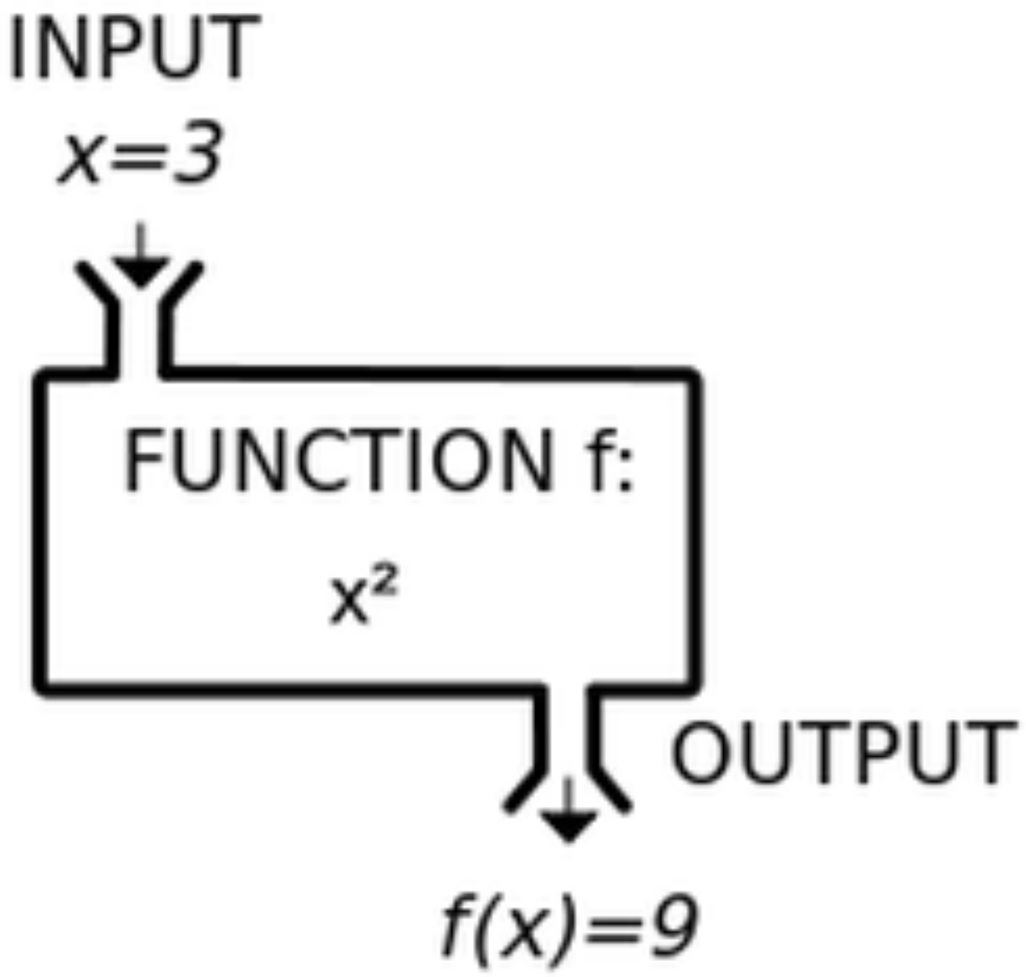
Black-box Model

Function in Math = Naming for Promised Calcs



 Reference: http://cs.wellesley.edu/~cs115/readings/more-functions.html

Function in C++ = Naming for Promised OPs




```
int iSqrt(int p)
{
    return p * p;
}
```

```
x = 3;

y = iSqrt( x );
```

 Reference: http://cs.wellesley.edu/~cs115/readings/more-functions.html

Declare, Define, and Call

- Function Declaring: introduces the function name and its type

```
int iSqrt(int);
```

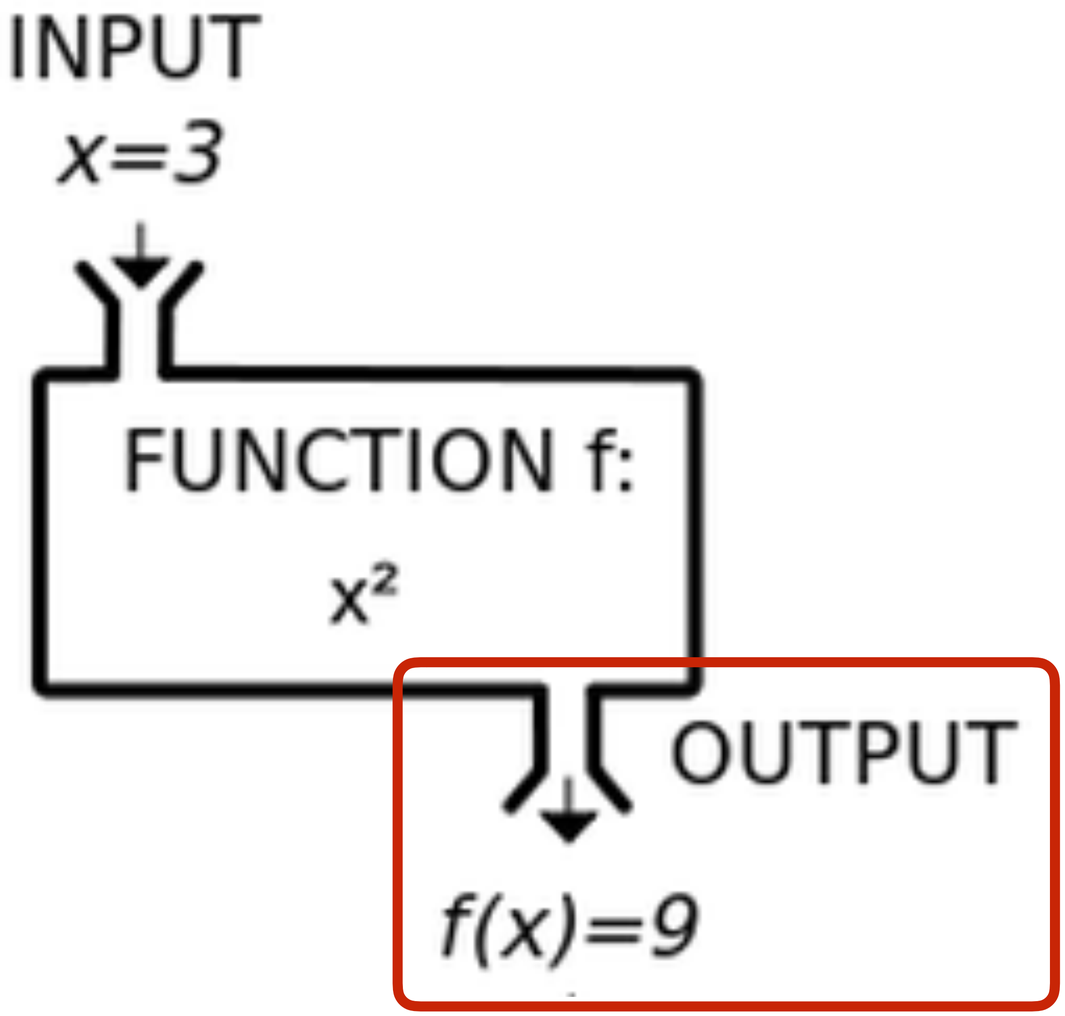
- Function Defining: associates the function name/type with the function body

```
int iSqrt(int p)
{
    return p * p;
}
```

- Function Calling: calls the pre-defined function to execute encapsulated operations

```
y = iSqrt( x );
```

Return Value (= Result)



Type of Result

```
int iSqrt(int p)
{
    return p * p;
}
```

Returning Result

```
x = 3;

y = iSqrt( x );
```

Storing Result

Library: cmath

- Collection of mathematical functions in standard C++ language
- ***sqrt*** function is in cmath library
  - A library is also a collection of implementations of behavior (= function),
  - written in terms of a language, that has a well-defined interface by which the behavior is invoked

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
     double input;
     // Get value from the user
     cout << "Enter number: ";
     cin >> input;
     // Compute the square root
     double root = sqrt(input);
     // Report result
     cout << "Square root of " << input << " = " << root << '\n';
 }
```

Library: cmath

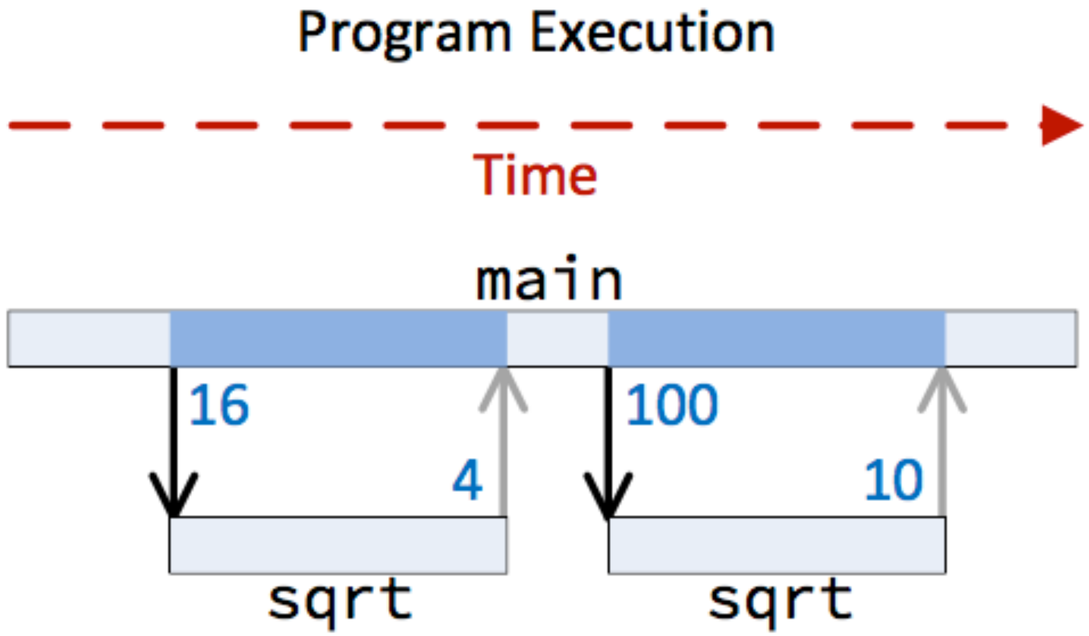
- Collection of mathematical functions in standard C++ language
- ***sqrt*** function is among them

```
int main() {
    double value;

    //  Assign variable
    value = 16;

    //  Compute s square root
    double root = sqrt(value);

    //  Compute another
    root = sqrt(100);
}
```



Library: functions in cmath

mathfunctions Module
`double sqrt(double x)` Computes the square root of a number: `sqrt(`x`)` $= \sqrt{x}$
`double exp(double x)` Computes e raised a power: `exp(`x`)` $= e^x$
`double log(double x)` Computes the natural logarithm of a number: `log(`x`)` $= \log_e x = \ln x$
`double log10(double x)` Computes the common logarithm of a number: `log(`x`)` $= \log_{10} x$
`double cos(double)` Computes the cosine of a value specified in radians: `cos(`x`)` $= \cos x$; other trigonometric functions include sine, tangent, arc cosine, arc sine, arc tangent, hyperbolic cosine, hyperbolic sine, and hyperbolic tangent
`double pow(double x, double y)` Raises one number to a power of another: `pow(`x, y`)` $= x^y$
`double fabs(double x)` Computes the absolute value of a number: `fabs(`x`)` $= \lvert x \rvert$

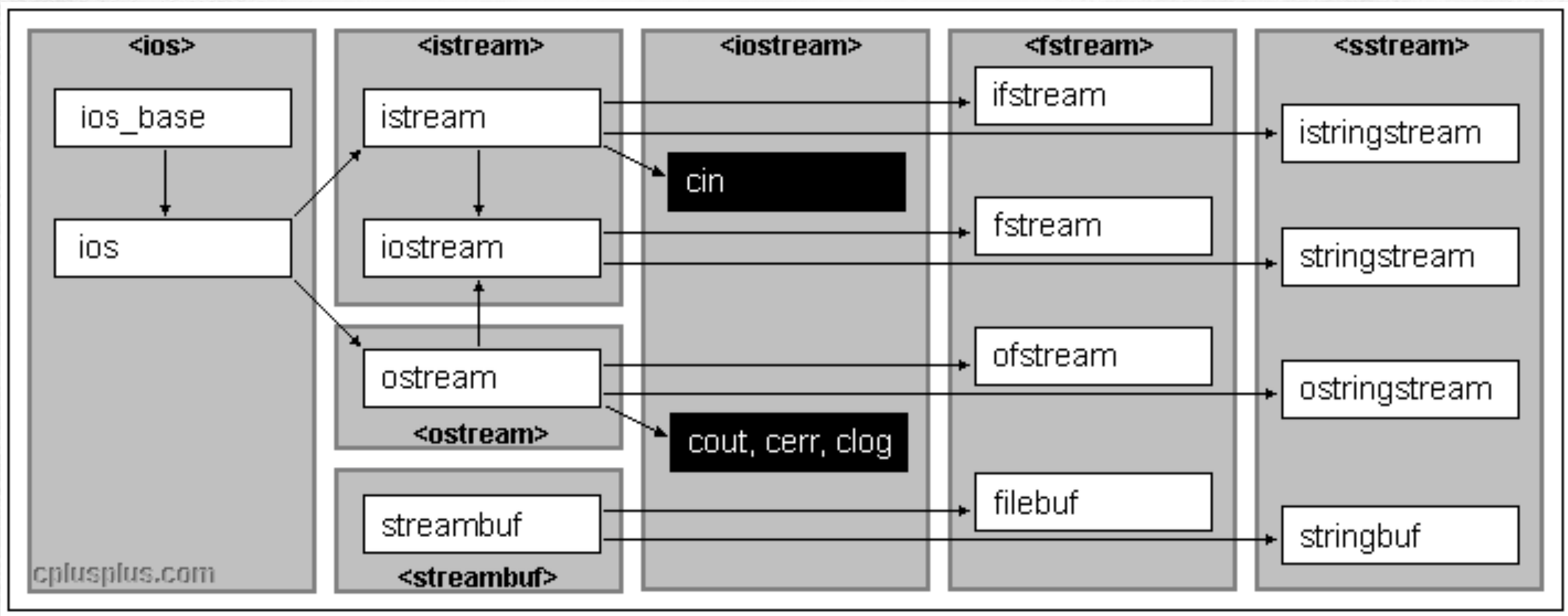
Library: C Libraries

<cassert> (assert.h)	C Diagnostics Library (header)
<cctype> (ctype.h)	Character handling functions (header)
<cerrno> (errno.h)	C Errors (header)
<cfenv> (fenv.h)	Floating-point environment (header)
<cfloat> (float.h)	Characteristics of floating-point types (header)
<cinttypes> (inttypes.h)	C integer types (header)
<ciso646> (iso646.h)	ISO 646 Alternative operator spellings (header)
<climits> (limits.h)	Sizes of integral types (header)
<clocale> (locale.h)	C localization library (header)
<cmath> (math.h)	C numerics library (header)
<csetjmp> (setjmp.h)	Non local jumps (header)
<csignal> (signal.h)	C library to handle signals (header)
<cstdarg> (stdarg.h)	Variable arguments handling (header)
<cstdbool> (stdbool.h)	Boolean type (header)
<cstddef> (stddef.h)	C Standard definitions (header)
<cstdint> (stdint.h)	Integer types (header)
<cstdio> (stdio.h)	C library to perform Input/Output operations (header)
<cstdlib> (stdlib.h)	C Standard General Utilities Library (header)
<cstring> (string.h)	C Strings (header)
<ctgmath> (tgmath.h)	Type-generic math (header)
<ctime> (time.h)	C Time Library (header)
<cuchar> (uchar.h)	Unicode characters (header)
<cwchar> (wchar.h)	Wide characters (header)
<cwctype> (wctype.h)	Wide character type (header)

Library: Container Libraries

<array>	Array header (header)
<bitset>	Bitset header (header)
<deque>	Deque header (header)
<forward_list>	Forward list (header)
<list>	List header (header)
<map>	Map header (header)
<queue>	Queue header (header)
<set>	Set header (header)
<stack>	Stack header (header)
<unordered_map>	Unordered map header (header)
<unordered_set>	Unordered set header (header)
<vector>	Vector header (header)

Library: Standard I/O Libraries



Function Definition Dissection

Type of value the function computes

Name of function

Type of value the function requires the caller to provide

Body of function

The name the function uses for the value provided by the caller

```
double square_root(double x) {
    double diff;
    //  Compute a provisional square root
    double root = 1.0;

    do { //  Loop until the p
        //  is close enough t
        root = (root + x/root) / 2.0;
        //  How bad is the approximation?
        diff = root * root - x;
    } while (diff > 0.0001 || diff < -0.0001);
    return root;
}
```

Function without Input & Return Value

Listing 9.2: simplefunction.cpp

```
#include <iostream>

// Definition of the prompt function
void prompt() {
    std::cout << "Please enter an integer value: ";
}

int main() {
    int value1, value2, sum;
    std::cout << "This program adds together two integers.\n";
    prompt();    // Call the function
    std::cin >> value1;
    prompt();    // Call the function again
    std::cin >> value2;
    sum = value1 + value2;
    std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Function with Return Value

Listing 9.6: betterprompt.cpp

```
#include <iostream>

// Definition of the prompt function
int prompt() {
    int result;
    std::cout << "Please enter an integer value: ";
    std::cin >> result;
    return result;
}

int main() {
    int value1, value2, sum;
    std::cout << "This program adds together two integers.\n";
    value1 = prompt();    // Call the function
    value2 = prompt();    // Call the function again
    sum = value1 + value2;
    std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Function with Input & Return Value

Listing 9.7: evenbetterprompt.cpp

```
#include <iostream>

// Definition of the prompt function
int prompt(int n) {
    int result;
    std::cout << "Please enter integer #" << n << ": ";
    std::cin >> result;
    return result;
}

int main() {
    int value1, value2, sum;
    std::cout << "This program adds together two integers.\n";
    value1 = prompt(1);    // Call the function
    value2 = prompt(2);    // Call the function again
    sum = value1 + value2;
    std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Default Arguments

- Function Declaring with Default Arguments
  - Allows a function to be called without providing one or more trailing arguments.

```
void point(int x = 3, int y = 4);
```

- Function Calling

```
point(1,2); // calls point(1,2)
point(1);   // calls point(1,4)
point();    // calls point(3,4)
```


Reference: http://en.cppreference.com/w/cpp/language/default_arguments

Object Oriented Programming by C++

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr

Web: http://mobilelab.khu.ac.kr/