CS118 – Programming Fundamentals

Lecture # 17 Monday, October 21, 2019 FALL 2019 FAST – NUCES, Faisalabad Campus

Zain Iqbal

Objectives

In this week/classes, you will:

- Learn about standard (predefined) functions and discover how to use them in a program
- Learn about user-defined functions
- Examine value-returning functions, including actual and formal parameters
- Explore how to construct and use a value-returning, user-defined function in a program

Modular Programming

- Modular programming: Breaking a program up into smaller, manageable functions or modules
- Function: A collection of statements to perform a task
- Motivation for modular programming
 - Improves maintainability of programs
 - Simplifies the process of writing programs

Introduction

Boss to worker analogy

■ A boss (the calling function or caller) asks a worker (the called function) to perform a task and return (i.e., report back) the results when the task is done

Introduction

- Functions are like building blocks
- They allow complicated programs to be divided into manageable pieces
- Some advantages of functions:
 - A programmer can focus on just that part of the program and construct it, debug it, and perfect it
 - Different people can work on different functions simultaneously
 - Can be re-used (even in different programs)
 - Enhance program readability

Introduction (cont'd.)

Functions

- Called modules
- Like miniature programs
- Can be put together to form a larger program

Predefined Functions

- In algebra, a function is defined as a rule or correspondence between values, called the function's arguments, and the unique value of the function associated with the arguments
 - \blacksquare If f(x) = 2x + 5, then f(1) = 7, f(2) = 9, and f(3) = 11
 - 1, 2, and 3 are arguments
 - 7, 9, and 11 are the corresponding values

- Some of the predefined mathematical functions are:
 - sqrt(x);
 - sqrt(4);
 - \rightarrow sqrt(3 6x);
 - pow(x, y);
 - floor(x);
- Predefined functions are organized into separate libraries
- I/O functions are in **iostream** header
- Math functions are in cmath/math header

- pow(x,y) calculates x^y
 - \rightarrow pow(2, 3) = 8.0
 - Returns a value of type double
 - x and y are the parameters (or arguments)
 - This function has two parameters
- sqrt(x) calculates the nonnegative square root of x, for x >= 0.0
 - \rightarrow sqrt(2.25) is 1.5
 - Type double

- The floor function floor(x) calculates largest whole number not greater than x
 - floor(48.79) is 48.0
 - Type double
 - Has only one parameter

TABLE 6-1 Predefined Functions

| Function | Header File | Purpose | Parameter(s) Type | Result |
|----------------------------|---------------------|---|----------------------|--------|
| abs(x) | <cstdlib></cstdlib> | Returns the absolute value of its argument: $abs(-7) = 7$ | int | int |
| ceil(x) | <cmath></cmath> | Returns the smallest whole number that is not less than x: ceil(56.34) = 57.0 | double | double |
| cos(x) | <cmath></cmath> | Returns the cosine of angle x: cos(0.0) = 1.0 | double (radians) | double |
| exp(x) | <cmath></cmath> | Returns e^x , where $e = 2.718$: exp(1.0) = 2.71828 | double | double |
| fabs (x) CS118 - FALL 2019 | <cmath></cmath> | Returns the absolute value of its argument: fabs (-5.67) = 5.67 | double | double |

TABLE 6-1 Predefined Functions (continued)

| Function | Header File | Purpose | Parameter(s) Type | Result |
|------------------------------------|-------------------|--|----------------------|--------|
| floor(x) | <cmath></cmath> | Returns the largest whole number that is not greater than x:floor (45.67) = 45.00 | double | double |
| islower(x) | <cctype></cctype> | Returns true if x is a lowercase letter; otherwise, it returns false; islower ('h') is true | int | int |
| isupper(x) | <cctype></cctype> | Returns true if x is a uppercase letter; otherwise, it returns false; isupper ('K') is true | int | int |
| pow(x, y) | <cmath></cmath> | Returns x ^y ; if x is negative, y must be a whole number: pow (0.16, 0.5) = 0.4 | double | double |
| sqrt(x) | <cmath></cmath> | Returns the nonnegative square root of x; x must be nonnegative: sqrt (4.0) = 2.0 | double | double |
| tolower(x) | <cctype></cctype> | Returns the lowercase value of x if x is uppercase; otherwise, it returns x | int | int |
| toupper (x) CS118-FALL 2019 | <cctype></cctype> | Returns the uppercase value of x if x is lowercase; otherwise, it returns x | int | int |

```
Example 6-1
//How to use predefined function
#include <iostream>
#include <cmath>
#include <cctype>
#include <cstdlib>
using namespace std;
int main(){
      int x;
      double u, v;
      cout << "Line 1: Uppercase a is "</pre>
             <k static_cast<char> (toupper('a'))
            << endl;
                                      //Line 1
      u = 4.2;
                               //Line 2
      v = 3.0;
                               //Line 3
      cout << "Line 4: " << u << " to the power of "</pre>
            << v << " = " << pow(u,v) << endl; //Line 4
      cout << "Line 5 : 5.0 to the power of 4 = "</pre>
      << pow(5.0,4) << endl;
                               //Line 5
      u = u + pow(3.0, 3); //Line 6
      cout << "Line 7: u = " << u << endl; //Line 7</pre>
                               //Line 8
      x = -15;
      cout << "Line 9: Absolute value of " << x</pre>
            << " = " << abs(x) << endl; //Line 9
      return 0;
```

Example 6-1 sample run:

```
Line 1: Uppercase a is A
Line 4: 4.2 to the power of 3 = 74.088
Line 5: 5.0 to the power of 4 = 625
Line 7: u = 31.2
Line 9: Absolute value of -15 = 15

Process exited after 0.08357 seconds with return value 0
Press any key to continue . . . _
```

User-Defined Functions

- Value-returning functions: have a return type
 - Return a value of a specific data type using the return statement function, called the type of the function
 - You need to add the following items:
 - The name of the function
 - The number of parameters, if any
 - The data type of each parameter
 - The data type of the value computed (that is, the value returned) by the Function
- Void functions: do not have a return type
 - Do not use a return statement to return a value

Function Return Type

If a function returns a value, the type of the value must be indicated

```
int main()
```

If a function does not return a value, its return type is void

```
void printHeading()
{
    cout << "\tMonthly Sales\n";
}</pre>
```

Defining and Calling Functions

- Function call: Statement that causes a function to execute
- Function definition: Statements that make up a function

```
int abs(int number);
Similarly the function abs might have the following definition:
int abs(int number)
{
   if(number < 0)
      number = -number;
   return number;
}</pre>
```

Function Definition

- Definition includes
 - return type: Data type of the value the function returns to the part of the program that called it
 - name: Name of the function. Function names follow same rules as variable names
 - parameter list: Variables that hold the values passed to the function
 - **body:** Statements that perform the function's task

Syntax: Value-Returning function

```
functionType functionName(formal parameter list)
{
    statements
}
```

Syntax: Formal Parameter List

```
dataType identifier, dataType identifier, ...
```

```
Function Return Type Function name

Formal Heading

int abs(int number)

formal parameter list

if (number<0)

number=-number;

return number;
```

Syntax: Actual Parameter List

expression or variable, expression or variable, ...

functionType functionName ()

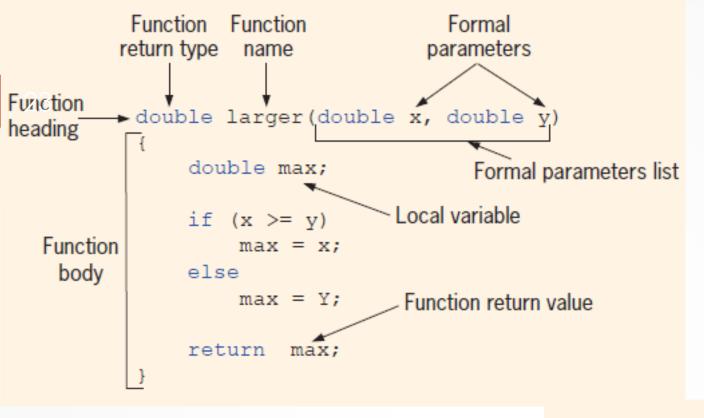
Can be zero parameter

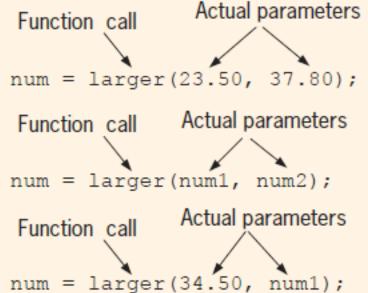
Actual Parameter Vs Formal 22 Parameter

```
Suppose that the heading of the function pow is:
double pow (double base, double exponent)
From the heading of the function pow, it follows that the formal parameters of pow are
base and exponent. Consider the following statements:
double u = 2.5;
double v = 3.0;
double x, y;
                                                                    //Line 1
x = pow(u, v);
y = pow(2.0, 3.2) + 5.1;
                                                                    //Line 2
cout << u << " to the power of 7 = " << pow(u, 7) << endl; //Line 3
```

Formal Parameter: A variable declared in the function heading.

Actual Parameter: A variable or expression listed in a call to a function.





Function Header

- The function header consists of
 - the function return type
 - the function name
 - the function parameter list
- Example:

int main()

Calling a Function

■ To call a function, use the function name followed by () and;

printHeading();

- When a function is called, the program executes the body of the function
- After the function terminates, execution resumes in the calling function at the point of call
- main() is automatically called when the program starts
- main() can call any number of functions
- Functions can call other functions

Function Call

functionName(actual parameter list)

Questions

