Chapter 3
Using Free Functions

3rd Edition

Computing Fundamentals with C++

Rick Mercer
Franklin, Beedle & Associates

Goals

- Evaluate some mathematical and trigonometric functions
- Use arguments in function calls
- Appreciate why programmers divide software into functions
- Read function headings so you can use existing functions

cmath functions

- C++ defines a large collection of standard math and trig functions such as

```
sqrt(x) // return the square root of x
fabs(x) // return the absolute value of x
```

- Functions are called by specifying the function name followed by the argument(s) in parentheses:

```
cout << sqrt(4.0) << " " << fabs(-2.34);
```

- To use mathematical functions, you must

```
#include <cmath> // For ceil, floor, pow
```

The pow function

- The pow function returns the first argument to the second argument's power

  `pow(2.0, 3.0)` returns 2 to the 3rd power ($2^3 = 8.0$)

```
#include <cmath>  // For the cmath functions
#include <iostream>
using namespace std;
int main() {
  double base, power;
  base = 2.0;
  power = 4.0;
  cout << pow(base, power) << endl;
  cout << pow(-2, 3) << endl;
  return 0;
}
```

Function Headings

- Can understand how to use a function if you see the function heading
- General form

  type functionName (*type arg1* , type arg2 , …)
- One function heading

```
double pow(double base, double power)
```
- Comments example function calls also help

```
pow(5.0, 3.0)  // evaluates to 125
```

Some `cmath` functions

double ceil(double x)	Smallest integer >= x	ceil(2.1)	3.0
double cos(double x)	Cosine of x radians	cos(1.0)	0.5403
double fabs(double x)	Absolute value of x	fabs(-1.5)	1.5
double floor(double x)	Largest integer <= x	floor(2.9)	2.0
double round(double x)	Nearest integer to x	round(1.5)	2
double sin(double x)	Sine of x radians	sin(1.0)	0.84147
double sqrt(double x)	Square root of x	sqrt(4.0)	2.0

Evaluate some Function Calls

- Different arguments cause different return values

```
ceil(0.1)_____      sqrt(16.0)_____
ceil(1.1)_____      sqrt(sqrt(16))_____
pow(2.0, 3)_____    fabs(-1.2)_____
sqrt(4.0)_____      floor(3.99)_____
```

Rounding to n decimals

- Code that rounds `x` to `n` decimal places using the `pow` and `floor` functions

```
x = 456.789;
n = 2;
x = x * pow(10, n);   // x _________
x = x + 0.5;          // x _________
x = floor(x);         // x _________
x = x / pow(10, n);   // x _________
```

Calling Documented Functions

- C++ has many free functions available
- Programmers can write their own free functions to develop programs
- Functions are more easily understood when documented with comment
- Functions may also include pre- and post-conditions *next slide*

Preconditions and Postconditions

- C++ comments that represents a contract between the implementers of a function and the user (client) of that function
- *Precondition* What the function requires to be true when called
- *Postcondition* What the function will do if the precondition(s) is/are met

Pre: and Post: conditions

- The preconditions are the circumstances that must be true in order for the function to successfully fulfill the postconditions
- Example *Precondition abbreviates to* `pre:`

```
double sqrt(double x)
// pre: x >= 0
// post: Returns square root of x
```

- `sqrt(-1.0)` evaluates to `nan` (not a number)

Function Headings

- A *function heading* is the complete declaration of a function without its implementation (sometimes called the function's *signature).*
- For example, this signature tells us how to call the function, but not how it works:

```
double sqrt(double x)
```

Function Headings

- General form of a function heading:

  return-type function-name (*parameter-list*)

  return-type is any C++ type e.g. `double int string`

  function-name is any valid C++ identifier that is not reserved

  parameter-list is a set of 0 or more parameters

- General form for declaring parameters:

  class-name identifier

- Examples

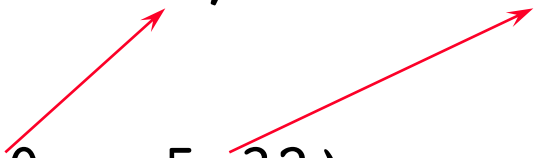
```
double f(double x)
int max(int j, int k)
string duplicate(string str, int n)
```

Argument/Parameter Associations

- Example call to `max` shows that arguments match parameters by position

```
double max(double x, double y)


cout << max(3.0, -5.32);
```



- The value of the 1st argument is copied to the 1st parameter, the value of the 2nd argument to the 2nd parameter, and so on, like assignments:

```
x = 3.0;
y = -5.32;
```

`char` and `bool` types

- C++ has two other primitive types that are used as the return types or parameters: `char` and `bool`
- `bool` type functions return 0 if the function returns false or a non-zero for true (usually 1)

```
#include <cctype> // isdigit isalpha
#include <iostream>
using namespace std;
int main() {
   cout << isdigit('9') << endl; // 1
   cout << isdigit('X') << endl; // 0
   cout << isalpha('A') << endl; // 1
   cout << isalpha('<') << endl; // 0
  return 0;
}
```

`char` and `int`

- The `toupper` and `tolower` functions convert a `char` to its lower case or upper case equivalent
- Because the return type for both is `int` instead of `char`, the functions need to be cast to `char` with the code `(char)` to see the character

```
toupper('a')          // 65
(char) toupper('a')   // A
tolower('A')          // 97
(char) tolower('A')   // a
```

Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
64	40	100	&#064;	@	96	60	140	&#096;	`
65	41	101	&#065;	A	97	61	141	&#097;	a
66	42	102	&#066;	B	98	62	142	&#098;	b
67	43	103	&#067;	C	99	63	143	&#099;	c

Summary

- Documented function headings provide the following information:
  - The type of value returned by the function
  - The function name
  - The number of arguments to use in a call
  - The type of arguments required in the function call
  - Pre- and post-conditions tell us what the function will do if the preconditions are met