# Chapter 4: Mathematical Functions, Characters, and Strings

## Sections 4.1–4.11

Textbooks: Y. Daniel Liang, Introduction to Programming with C++, 3rd Edition

These slides were adapted by Prof. Gheith Abandah from the Computer Engineering Department of the University of Jordan for the Course: Computer Skills for Engineers (0907101)

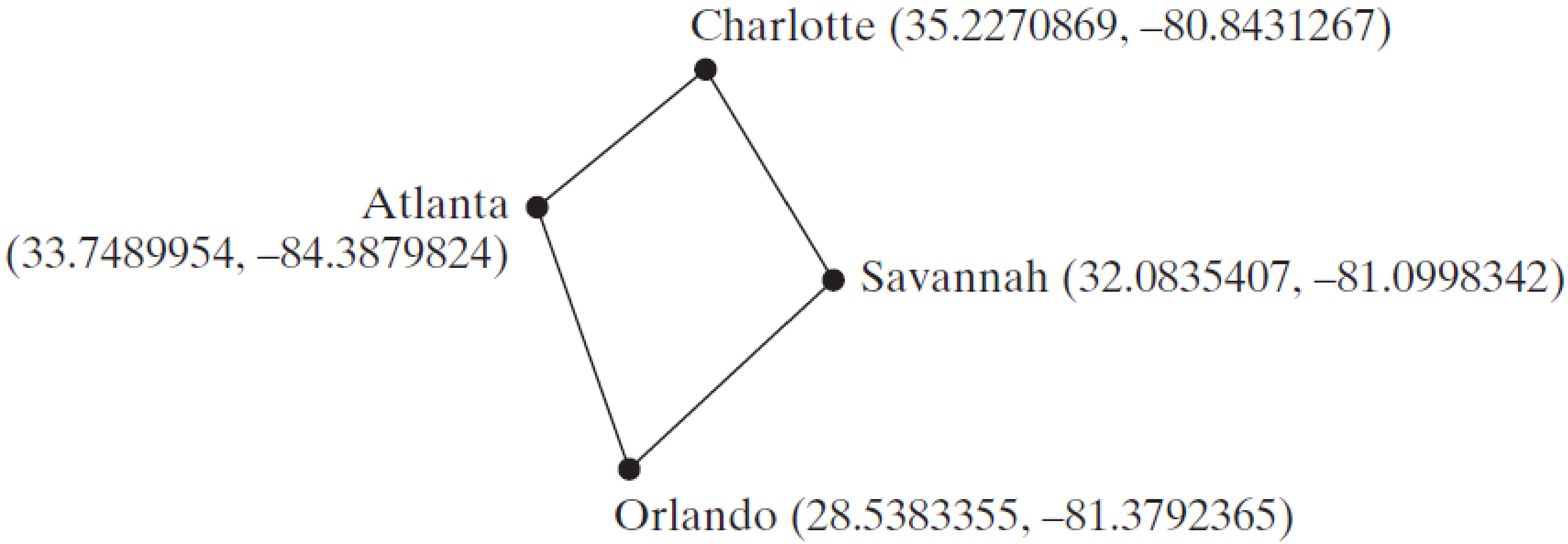
Updated by Dr. Ashraf Suyyagh (Spring 2021)

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Introduction

Suppose you need to estimate the area enclosed by four cities, given the GPS locations (latitude and longitude) of these cities, as shown in the following diagram. How would you write a program to solve this problem? You will be able to write such a program after completing this chapter.


Charlotte (35.2270869, –80.8431267)
Atlanta (33.7489954, –84.3879824)
Savannah (32.0835407, –81.0998342)
Orlando (28.5383355, –81.3792365)

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Mathematical Functions

C++ provides many useful functions in the **cmath** header for performing common mathematical functions.

1. Trigonometric functions
2. Exponent functions
3. Service functions

To use them, you need to include:

```
#include <cmath>
```

# Trigonometric Functions

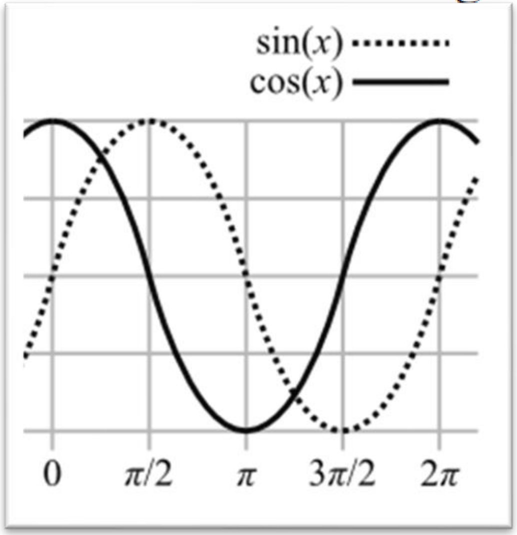
| *Function* | *Description* |
|---|---|
| `sin(radians)` | Returns the trigonometric sine of an angle in radians. |
| `cos(radians)` | Returns the trigonometric cosine of an angle in radians |
| `tan(radians)` | Returns the trigonometric tangent of an angle in radians. |
| `asin(a)` | Returns the angle in radians for the inverse of sine. |
| `acos(a)` | Returns the angle in radians for the inverse of cosine. |
| `atan(a)` | Returns the angle in radians for the inverse of tangent. |

`sin(0)` returns `0.0`

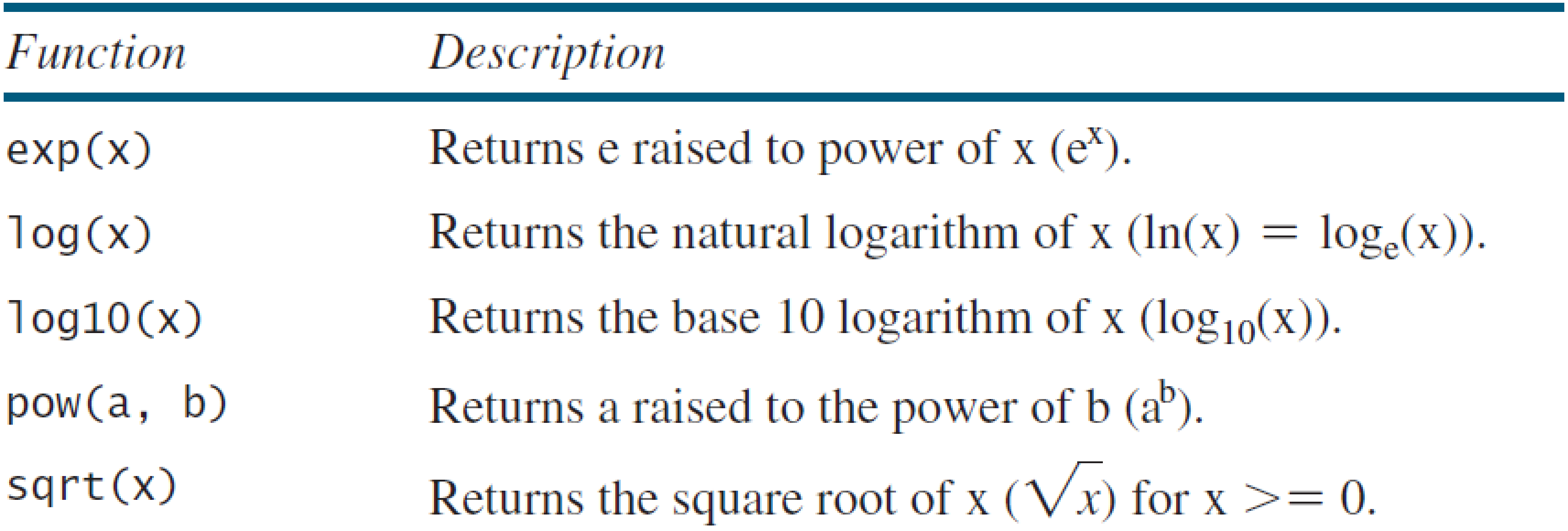
`sin(PI / 2)` returns `1.0`

`cos(0)` returns `1.0`

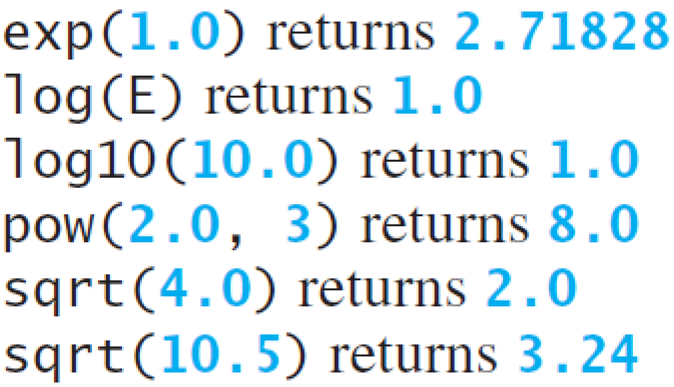
`atan(1.0)` returns `0.785398` (same as π/4)



# Exponent Functions



| *Function* | *Description* |
|---|---|
| `exp(x)` | Returns e raised to power of x ($e^x$). |
| `log(x)` | Returns the natural logarithm of x ($\ln(x) = \log_e(x)$). |
| `log10(x)` | Returns the base 10 logarithm of x ($\log_{10}(x)$). |
| `pow(a, b)` | Returns a raised to the power of b ($a^b$). |
| `sqrt(x)` | Returns the square root of x ($\sqrt{x}$) for x >= 0. |

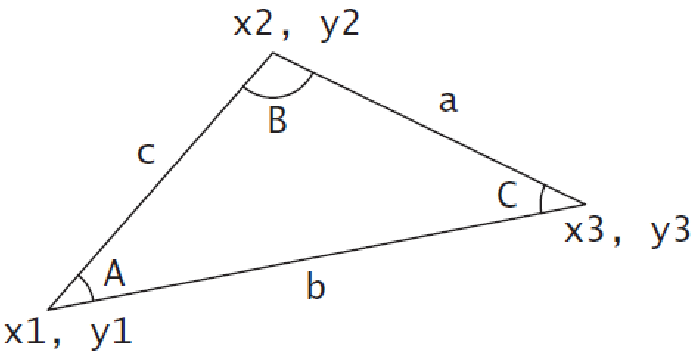


```
exp(1.0) returns 2.71828
log(E) returns 1.0
log10(10.0) returns 1.0
pow(2.0, 3) returns 8.0
sqrt(4.0) returns 2.0
sqrt(10.5) returns 3.24
```

# Service Functions

| Function | Description | Example |
|---|---|---|
| ceil(x) | x is rounded up to its nearest integer. This integer is returned as a double value. | ceil(2.1) returns 3.0<br>ceil(-2.1) returns -2.0 |
| floor(x) | x is rounded down to its nearest integer. This integer is returned as a double value. | floor(2.1) returns 2.0<br>floor(-2.1) returns -3.0 |
| min(x, y) | Returns the minimum of x and y. | max(2, 3) returns 3 |
| max(x, y) | Returns the maximum of x and y. | min(2.5, 4.6) returns 2.5 |
| abs(x) | Returns the absolute value of x. | abs(-2.1) returns 2.1 |

# Case Study: Computing Angles of a Triangle



```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

A program that prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the triangle’s angles.

ComputeAngles

Run

# ComputeAngles.cpp 1/2

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    // Prompt the user to enter three points
    cout << "Enter three points: ";
    double x1, y1, x2, y2, x3, y3;
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

    // Compute three sides
    double a = sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
    double b = sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3));
    double c = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
```

# ComputeAngles.cpp 2/2

```
    // Obtain three angles in radians
    double A = acos((a * a - b * b - c * c) / (-2 * b * c));
    double B = acos((b * b - a * a - c * c) / (-2 * a * c));
    double C = acos((c * c - b * b - a * a) / (-2 * a * b));

    // Display the angles in degress
    const double PI = 3.14159;
    cout << "The three angles are " << A * 180 / PI << " "
        << B * 180 / PI << " " << C * 180 / PI << endl;

    return 0;
}
```

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Character Data Type

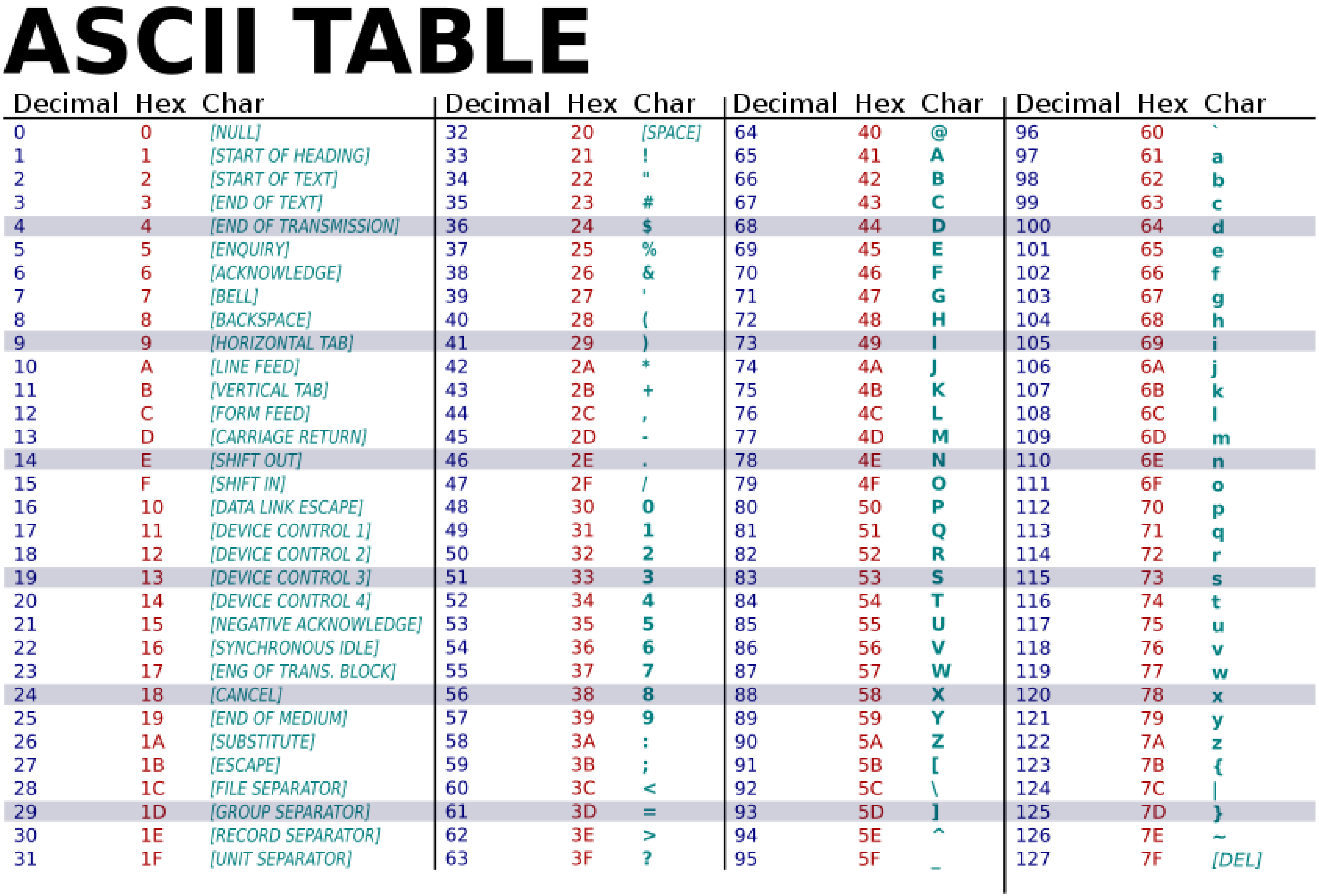
- *A character data type represents a single character.*

  ```
  char letter = 'A'; (ASCII)
  char numChar = '4'; (ASCII)
  ```

- The increment and decrement operators can also be used on **`char`** variables to get the next or preceding character. For example, the following statements display character **`b`**.

  ```
  char ch = 'a';
  cout << ++ch;
  ```

- The characters are encoded into numbers using the ASCII code.



# ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

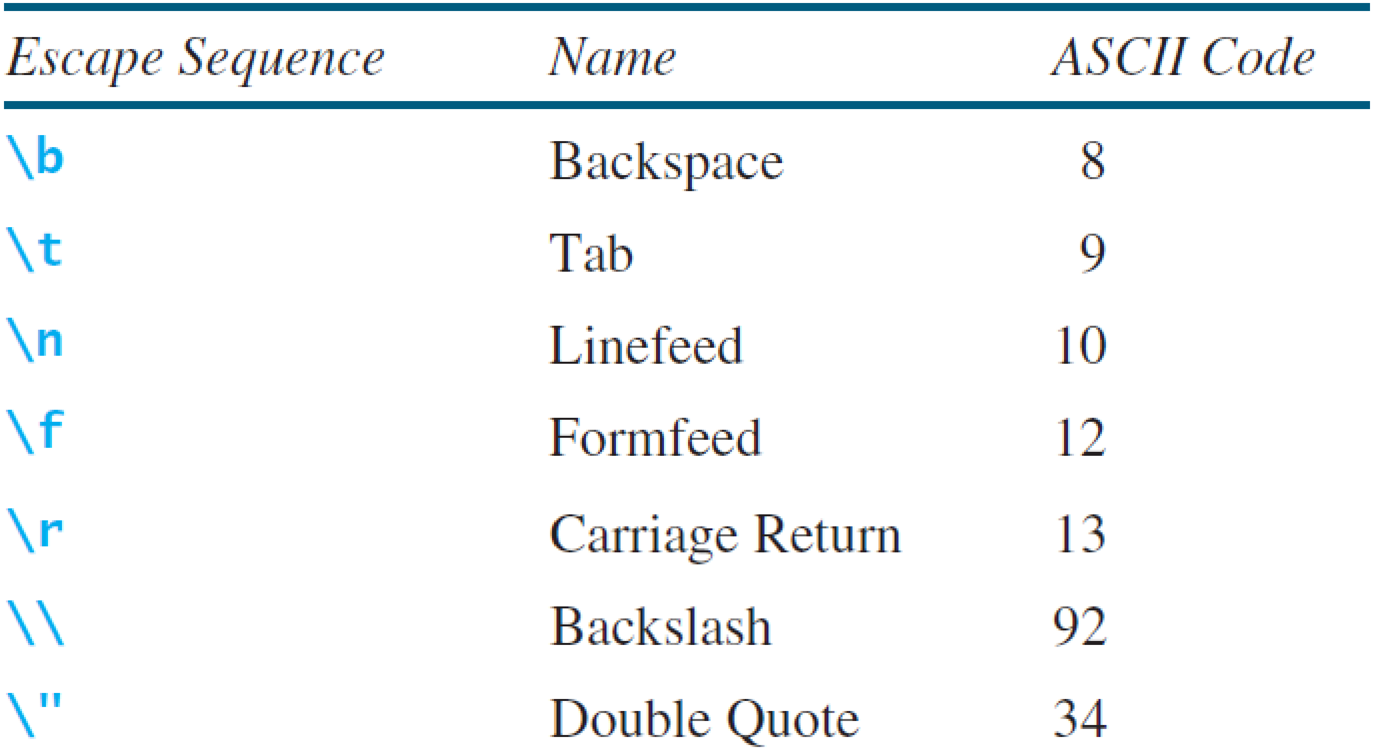
# Read Characters

To read a character from the keyboard, use

```
cout << "Enter a character: ";
char ch;
cin >> ch; // Read a character
```

# Escape Sequences

C++ uses a special notation to represent special character.



| *Escape Sequence* | *Name* | *ASCII Code* |
|---|---|---|
| `\b` | Backspace | 8 |
| `\t` | Tab | 9 |
| `\n` | Linefeed | 10 |
| `\f` | Formfeed | 12 |
| `\r` | Carriage Return | 13 |
| `\\` | Backslash | 92 |
| `\"` | Double Quote | 34 |

```
cout << "He said \"Hi\".\n";
```

The output is: `He said "Hi".`

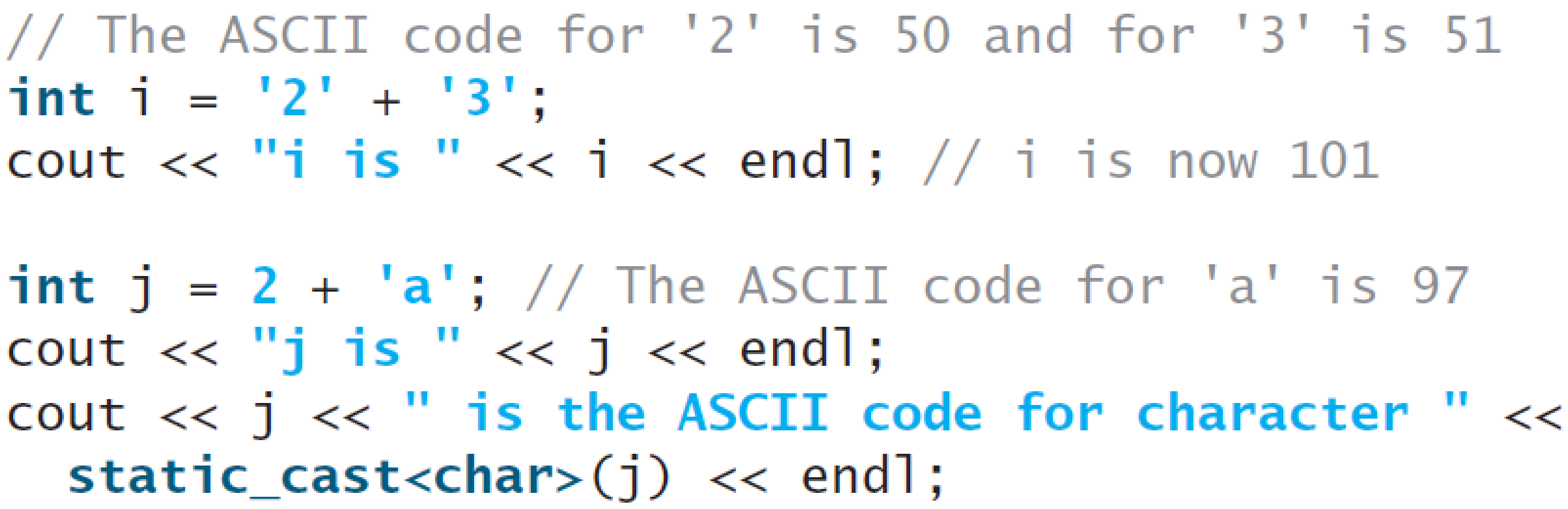
# Casting between `char` and Numeric Types

- A **`char`** can be cast into any numeric type, and vice versa.
- When an integer is cast into a char, only its lower 8 bits of data are used; the other part is ignored.

```
int i = 'a';
// Same as int i = static_cast<int>('a');

char c = 97;
// Same as char c = static_cast<char>(97);
```

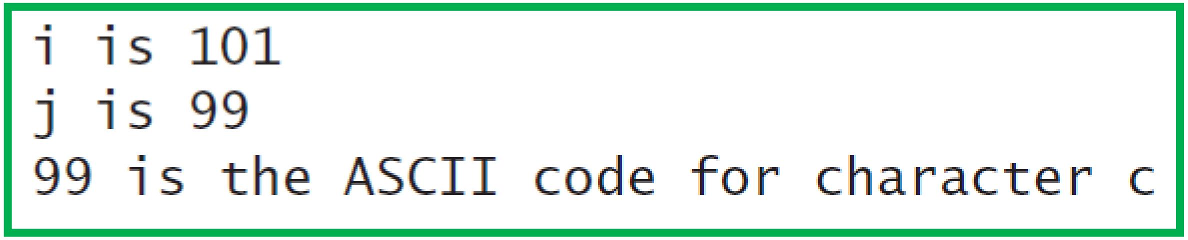
# Numeric Operators on Characters

The **char** type is treated as if it is an integer of the byte size. All numeric operators can be applied to **char** operands.



```
// The ASCII code for '2' is 50 and for '3' is 51
int i = '2' + '3';
cout << "i is " << i << endl; // i is now 101

int j = 2 + 'a'; // The ASCII code for 'a' is 97
cout << "j is " << j << endl;
cout << j << " is the ASCII code for character " <<
  static_cast<char>(j) << endl;
```

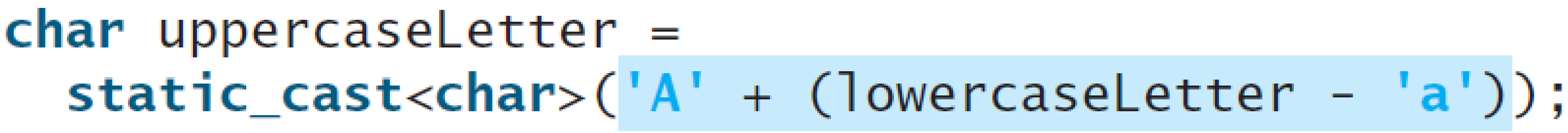
Display



```
i is 101
j is 99
99 is the ASCII code for character c
```

# Example: Converting a Lowercase to Uppercase

A program that prompts the user to enter a lowercase letter and finds its corresponding uppercase letter.



```
char uppercaseLetter =
  static_cast<char>('A' + (lowercaseLetter - 'a'));
```

ToUppercase

Run

# Comparing and Testing Characters

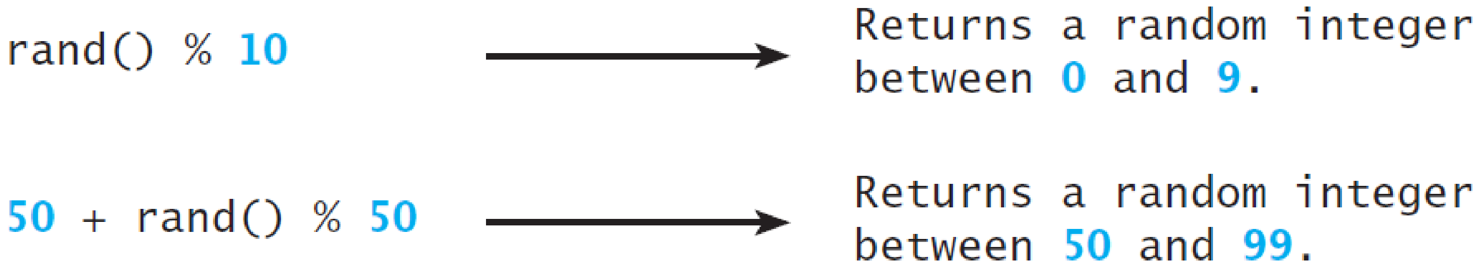
- The ASCII for lowercase letters are consecutive integers starting from the code for 'a', then for 'b', 'c', ..., and 'z'. The same is true for the uppercase letters.
- The lower case of a letter is larger than its upper case by 32.
- Two characters can be compared using the comparison operators just like comparing two numbers.
- `'a' < 'b'` is true because the ASCII code for `'a'` (97) is less than the ASCII code for 'b' (98).
- `'a' < 'A'` is false.
- `'1' < '8'` is true.

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Case Study: Generating Random Characters

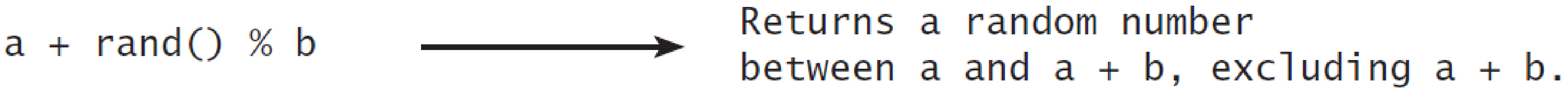
The **rand()** function returns a random integer. You can use it to write a simple expression to generate random numbers in any range.



rand() % 10 → Returns a random integer between 0 and 9.

50 + rand() % 50 → Returns a random integer between 50 and 99.

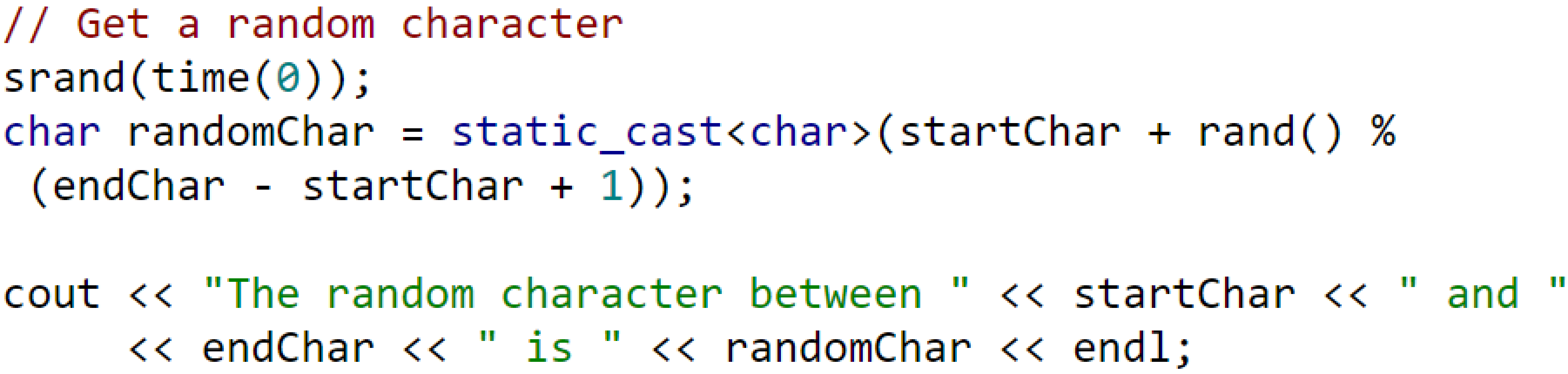
In general,



a + rand() % b → Returns a random number between a and a + b, excluding a + b.

# Case Study: Generating Random Characters, cont.

Every character has a unique ASCII code between 0 and 127. To generate a random character is to generate a random integer between 0 and 127. The **srand(seed)** function is used to set a seed.



```
// Get a random character
srand(time(0));
char randomChar = static_cast<char>(startChar + rand() %
 (endChar - startChar + 1));

cout << "The random character between " << startChar << " and "
     << endChar << " is " << randomChar << endl;
```

DisplayRandomCharacter

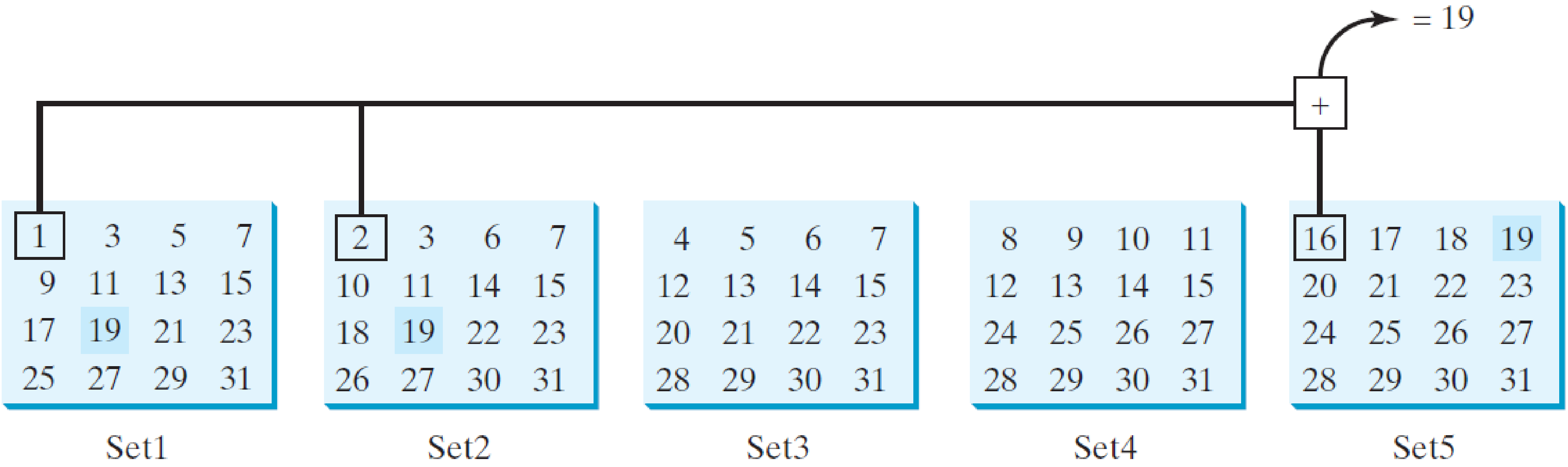
Run

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Case Study: Guessing Birthdays

- The program can find your birth date. The program prompts you to answer whether your birth date is in the following five sets of numbers:



| | | | |
|---|---|---|---|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |
| 25 | 27 | 29 | 31 |

Set1

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 7 |
| 10 | 11 | 14 | 15 |
| 18 | 19 | 22 | 23 |
| 26 | 27 | 30 | 31 |

Set2

| | | | |
|---|---|---|---|
| 4 | 5 | 6 | 7 |
| 12 | 13 | 14 | 15 |
| 20 | 21 | 22 | 23 |
| 28 | 29 | 30 | 31 |

Set3

| | | | |
|---|---|---|---|
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |

Set4

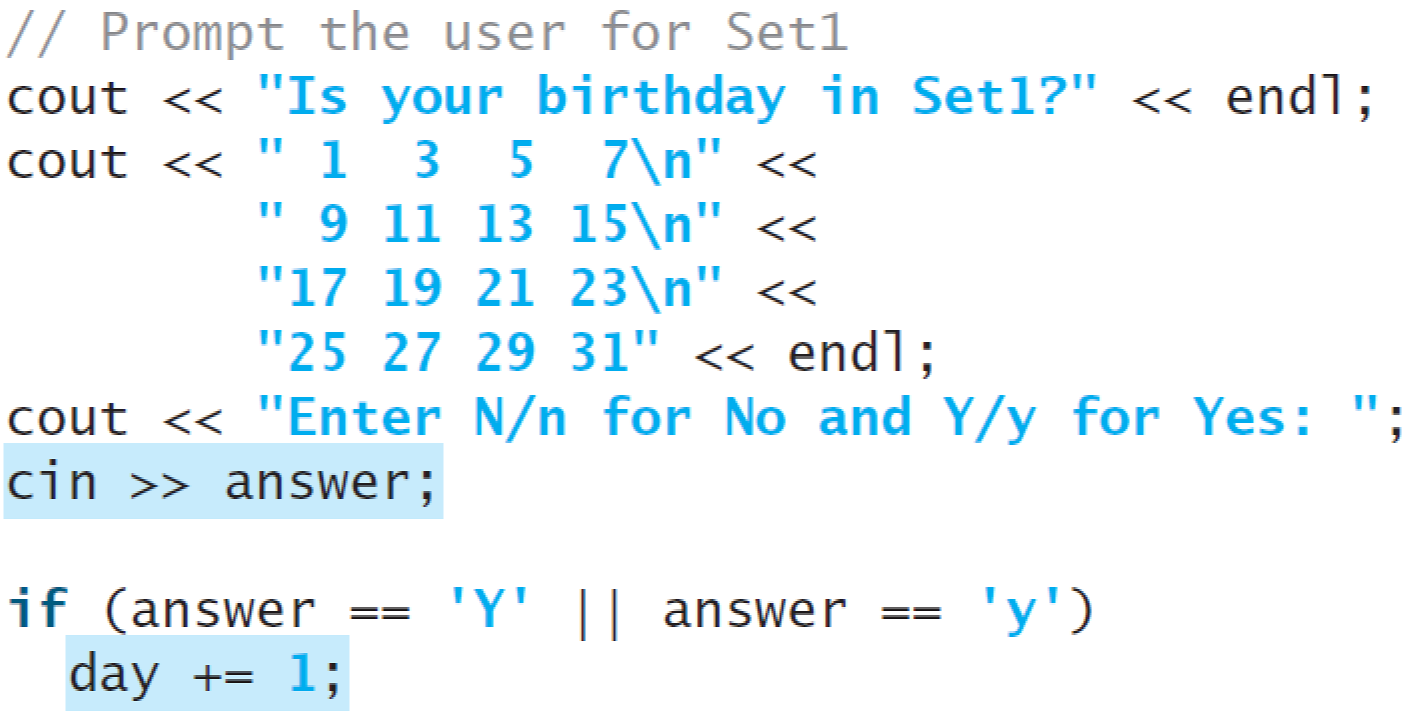
| | | | |
|---|---|---|---|
| 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |

Set5

GuessBirthday

Run

# Case Study: Guessing Birthdays



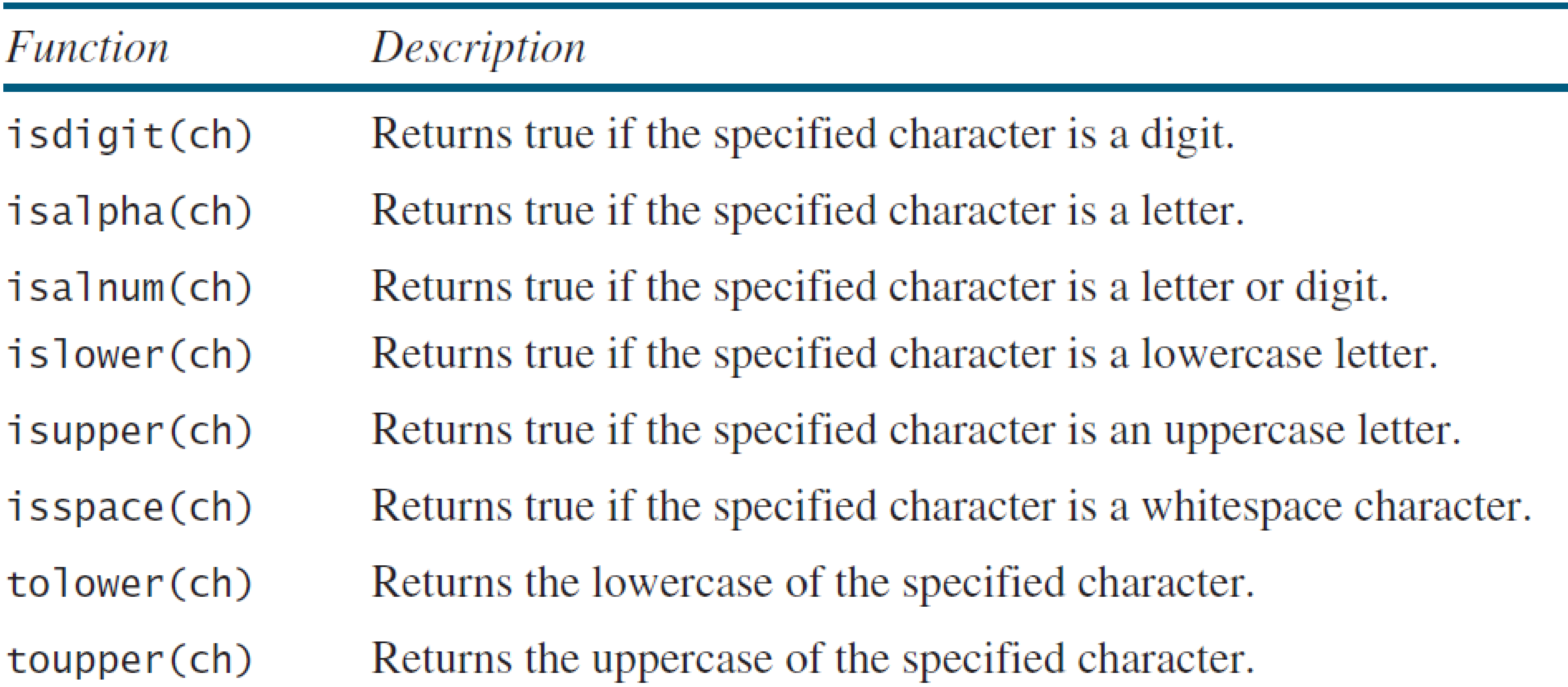
```
// Prompt the user for Set1
cout << "Is your birthday in Set1?" << endl;
cout << " 1  3  5  7\n" <<
        " 9 11 13 15\n" <<
        "17 19 21 23\n" <<
        "25 27 29 31" << endl;
cout << "Enter N/n for No and Y/y for Yes: ";
cin >> answer;

if (answer == 'Y' || answer == 'y')
  day += 1;
```

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

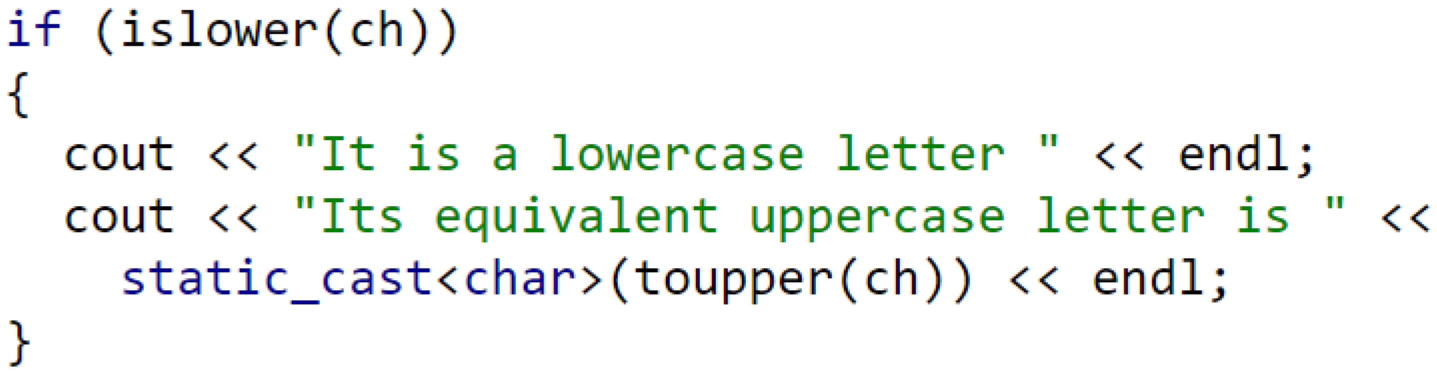
# Character Functions

*C++ contains functions for working with characters.*



| *Function* | *Description* |
|---|---|
| `isdigit(ch)` | Returns true if the specified character is a digit. |
| `isalpha(ch)` | Returns true if the specified character is a letter. |
| `isalnum(ch)` | Returns true if the specified character is a letter or digit. |
| `islower(ch)` | Returns true if the specified character is a lowercase letter. |
| `isupper(ch)` | Returns true if the specified character is an uppercase letter. |
| `isspace(ch)` | Returns true if the specified character is a whitespace character. |
| `tolower(ch)` | Returns the lowercase of the specified character. |
| `toupper(ch)` | Returns the uppercase of the specified character. |

# Example using Character Functions

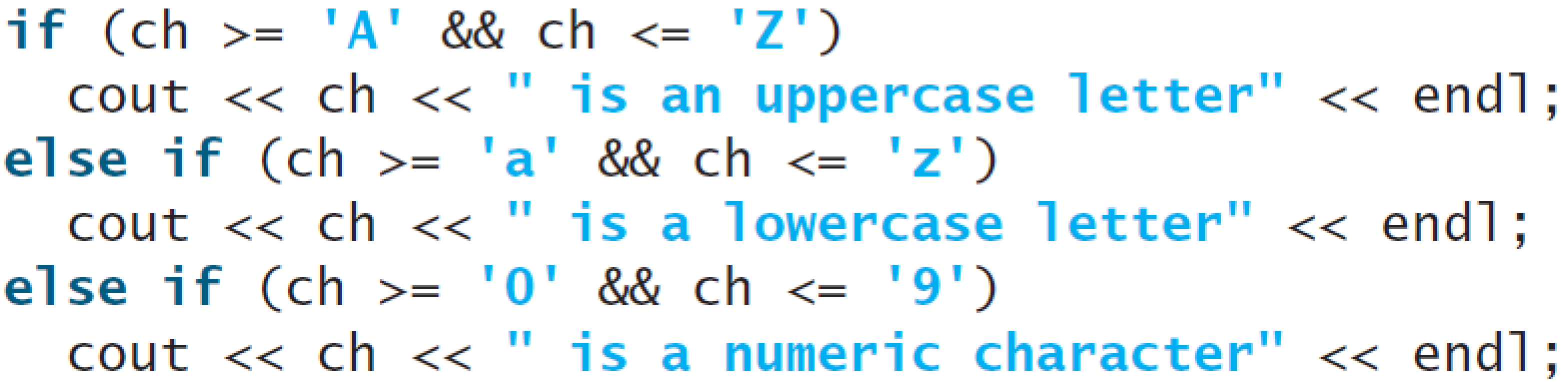


```
if (islower(ch))
{
  cout << "It is a lowercase letter " << endl;
  cout << "Its equivalent uppercase letter is " <<
    static_cast<char>(toupper(ch)) << endl;
}
```

CharacterFunctions Run

# Character Functions

- You can use **`isupper()`**, **`islower()`** and **`isdigit()`** in the code below.



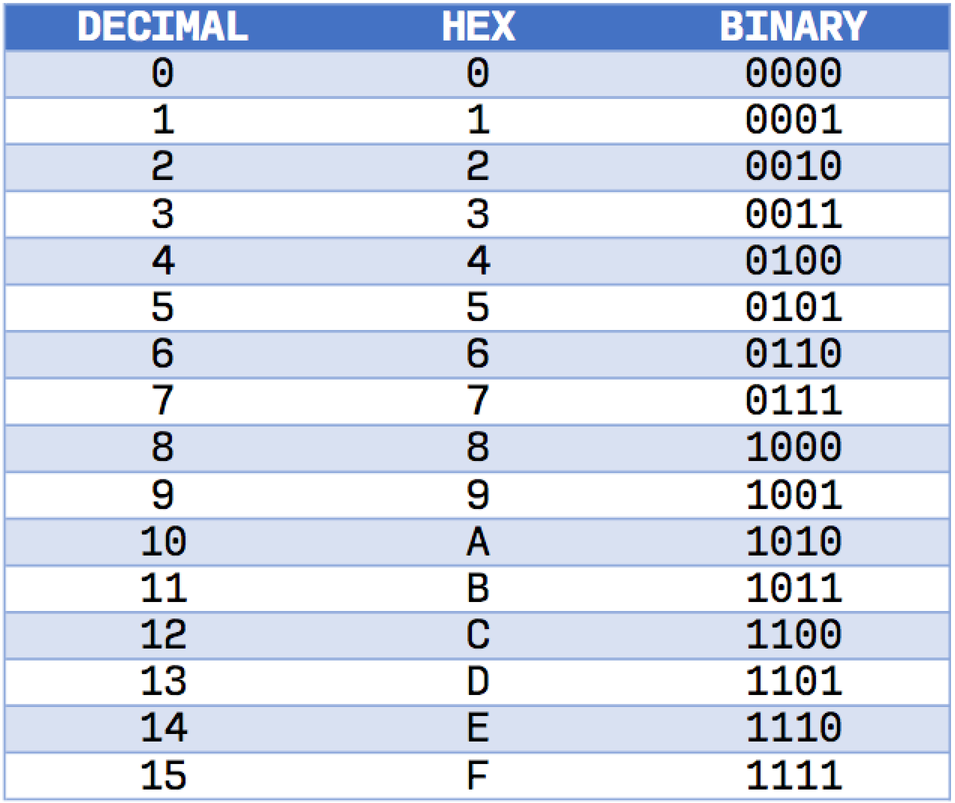
```
if (ch >= 'A' && ch <= 'Z')
  cout << ch << " is an uppercase letter" << endl;
else if (ch >= 'a' && ch <= 'z')
  cout << ch << " is a lowercase letter" << endl;
else if (ch >= '0' && ch <= '9')
  cout << ch << " is a numeric character" << endl;
```

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Case Study: Converting a Hexadecimal Digit to a Decimal Value

A program that converts a hexadecimal digit to decimal.

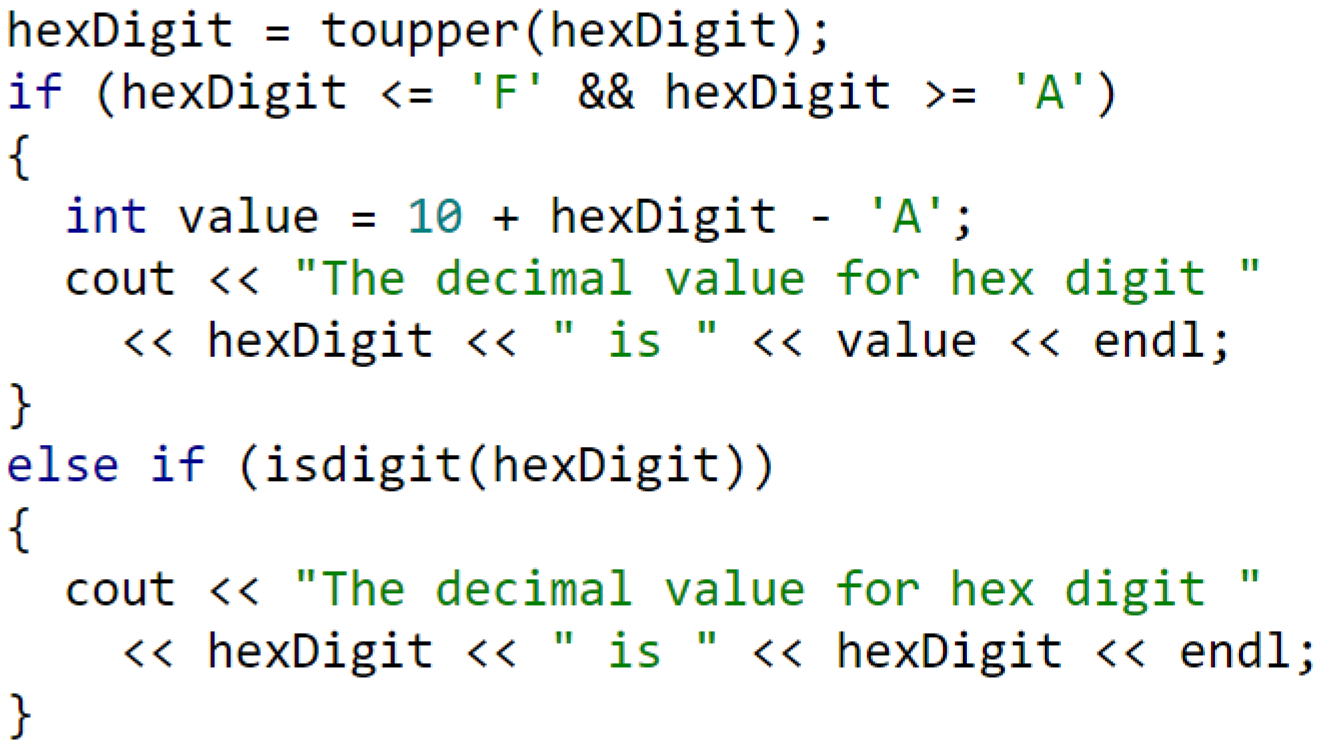


| DECIMAL | HEX | BINARY |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

HexDigit2Dec

Run

# HexDigit2Dec.cpp



```
hexDigit = toupper(hexDigit);
if (hexDigit <= 'F' && hexDigit >= 'A')
{
  int value = 10 + hexDigit - 'A';
  cout << "The decimal value for hex digit "
    << hexDigit << " is " << value << endl;
}
else if (isdigit(hexDigit))
{
  cout << "The decimal value for hex digit "
    << hexDigit << " is " << hexDigit << endl;
}
```

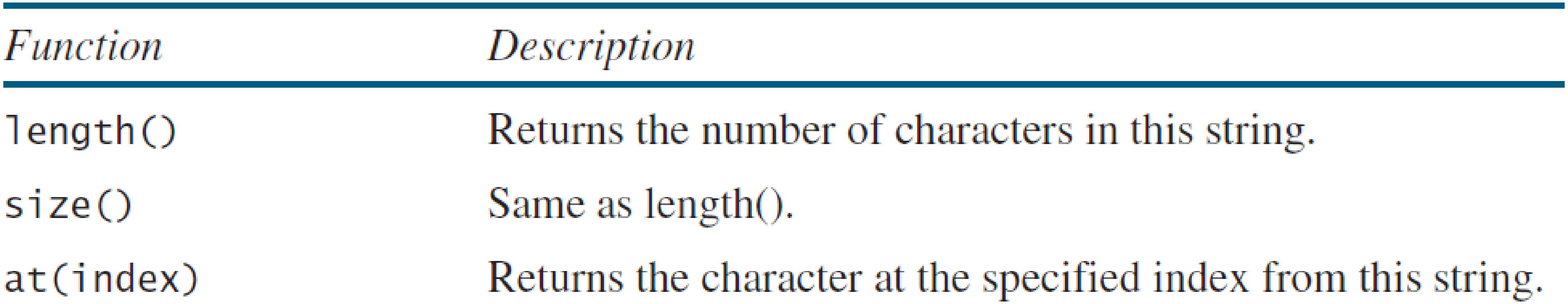
# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# The `string` Type

*A string is a sequence of characters.*

```
#include <string>
string s;
string message = "Programming is fun";
```



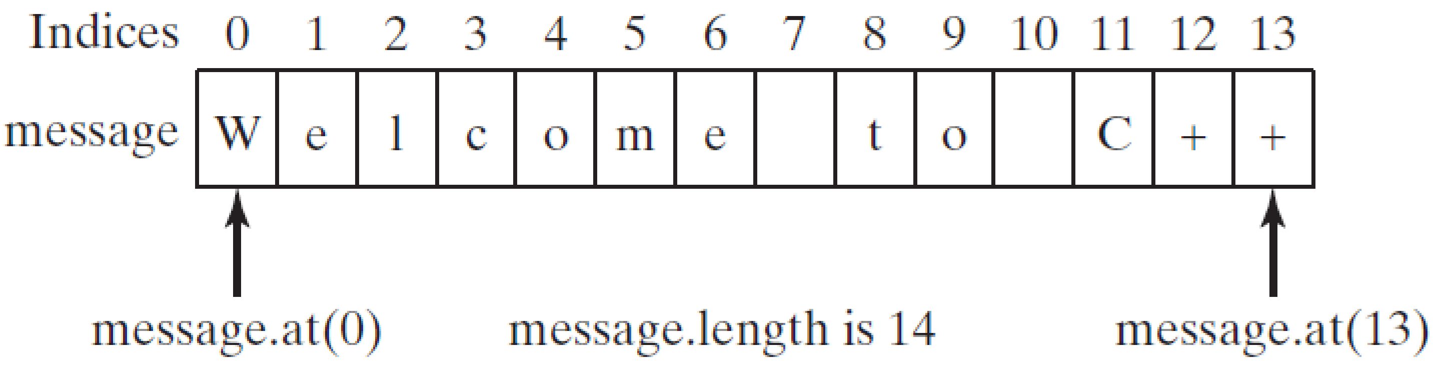
| *Function* | *Description* |
|---|---|
| `length()` | Returns the number of characters in this string. |
| `size()` | Same as length(). |
| `at(index)` | Returns the character at the specified index from this string. |

# String Subscript Operator

C++ provides the subscript operator for accessing the character at a specified index in a string using the syntax **stringName[index]**.

```
string s = "welcome to C++";
s.at(0) = 'W';
cout << s.length() << s[0] << endl;
```

```
14W
```



| Indices | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| message | W | e | l | c | o | m | e |  | t | o |  | C | + | + |

message.at(0)

message.length is 14

message.at(13)

# Concatenating Strings

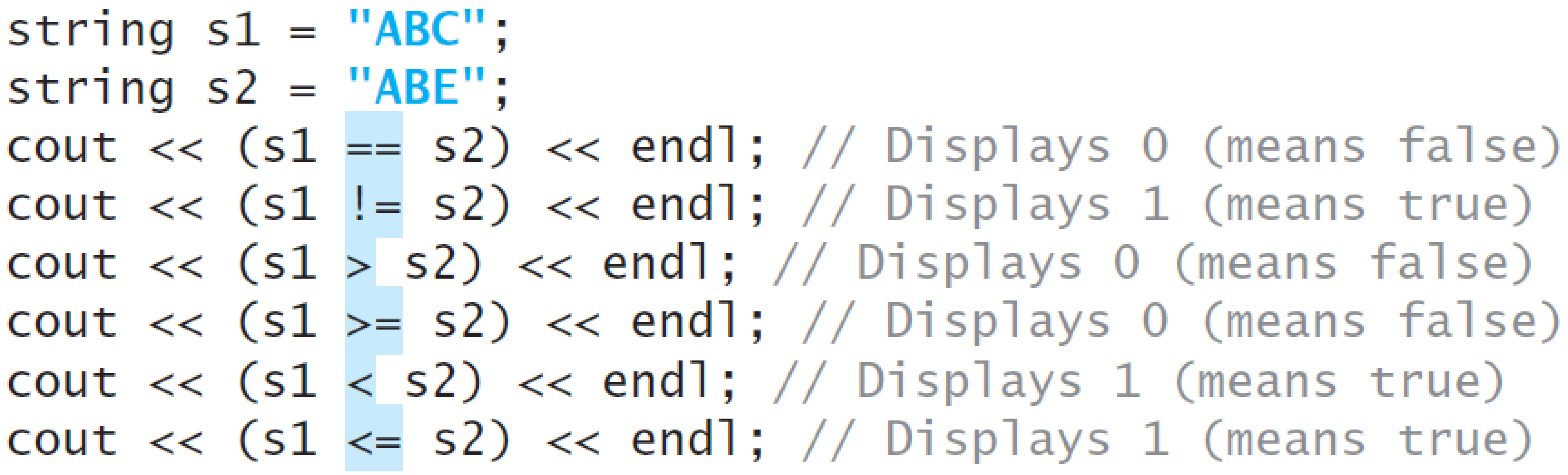
C++ provides the + operator for concatenating two strings.

```
string s3 = s1 + s2;
```

```
string m = "Good";
m += " morning";
m += '!';
cout << m << endl;
Good morning!
```

# Comparing Strings

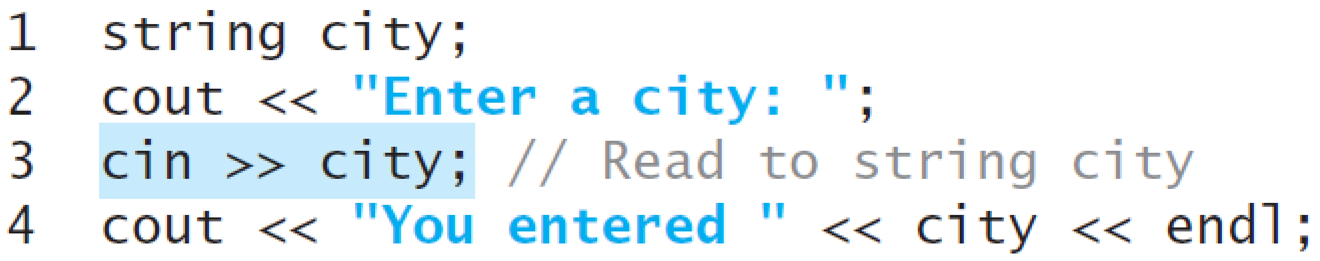
You can use the relational operators **==, !=, <, <=, >, >=** to compare two strings. This is done by comparing their corresponding characters on by one from left to right. For example,



```
string s1 = "ABC";
string s2 = "ABE";
cout << (s1 == s2) << endl; // Displays 0 (means false)
cout << (s1 != s2) << endl; // Displays 1 (means true)
cout << (s1 > s2) << endl; // Displays 0 (means false)
cout << (s1 >= s2) << endl; // Displays 0 (means false)
cout << (s1 < s2) << endl; // Displays 1 (means true)
cout << (s1 <= s2) << endl; // Displays 1 (means true)
```

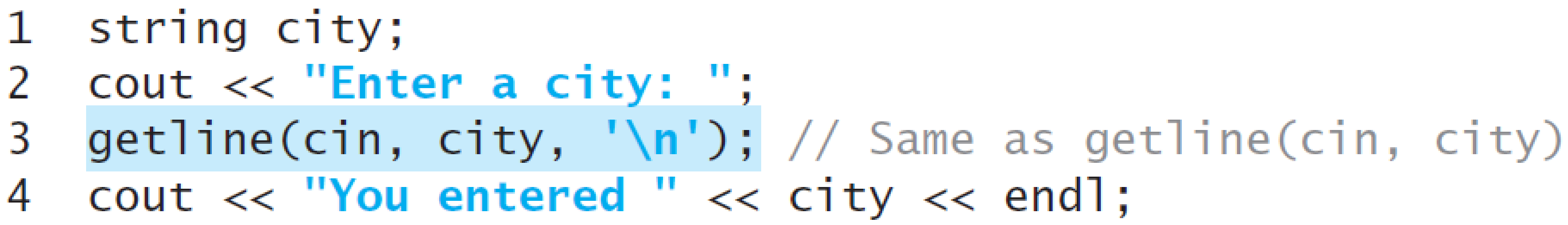
# Reading Strings

Reading a word:



```
string city;
cout << "Enter a city: ";
cin >> city; // Read to string city
cout << "You entered " << city << endl;
```

Reading a line using **`getline(cin, s, delimitCharacter)`**:



```
string city;
cout << "Enter a city: ";
getline(cin, city, '\n'); // Same as getline(cin, city)
cout << "You entered " << city << endl;
```

# Example: Order Two Cities

A program that prompts the user to enter two cities and displays them in alphabetical order.

OrderTwoCities

Run

# OrderTwoCities.cpp

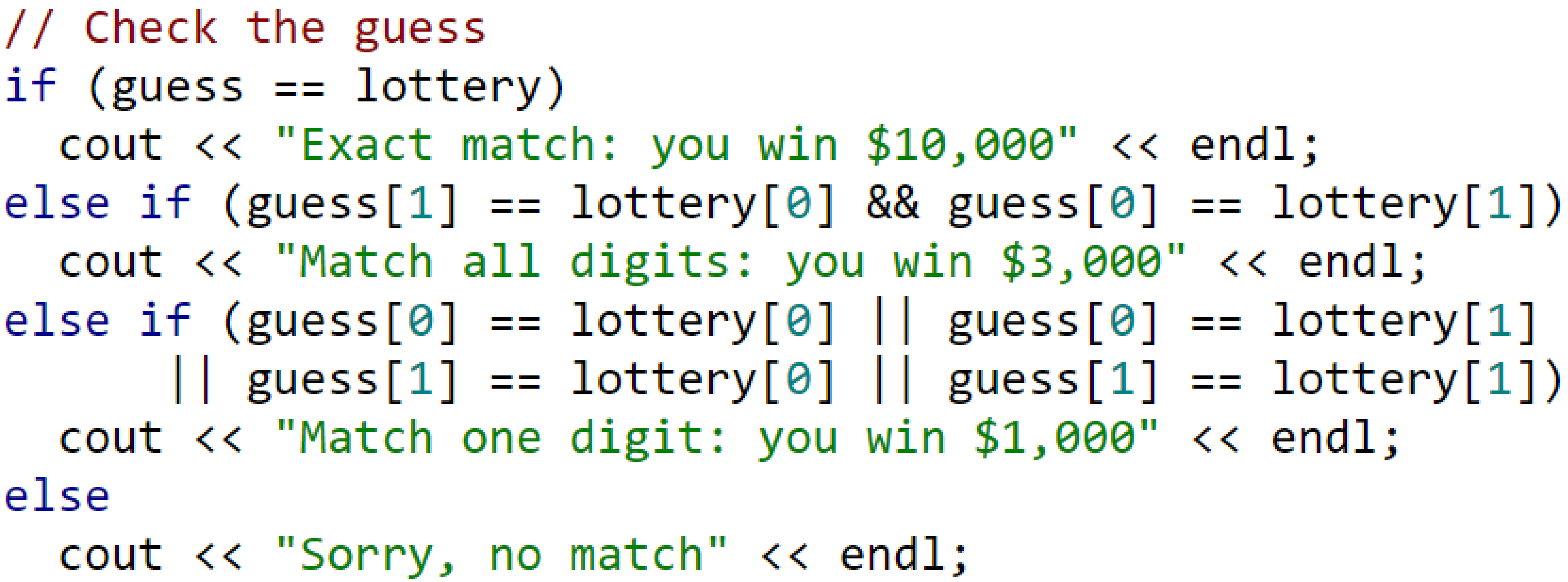
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string city1, city2;
    cout << "Enter the first city: ";
    getline(cin, city1);
    cout << "Enter the second city: ";
    getline(cin, city2);

    cout << "The cities in alphabetical order are ";
    if (city1 < city2)
        cout << city1 << " " << city2 << endl;
    else
        cout << city2 << " " << city1 << endl;

    return 0;
}
```

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Case Study: Revising the Lottery Program Using Strings

A problem can be solved using many different approaches. This section rewrites the lottery program in Listing 3.7 using strings. Using strings simplifies this program.



```
// Check the guess
if (guess == lottery)
  cout << "Exact match: you win $10,000" << endl;
else if (guess[1] == lottery[0] && guess[0] == lottery[1])
  cout << "Match all digits: you win $3,000" << endl;
else if (guess[0] == lottery[0] || guess[0] == lottery[1]
      || guess[1] == lottery[0] || guess[1] == lottery[1])
  cout << "Match one digit: you win $1,000" << endl;
else
  cout << "Sorry, no match" << endl;
```

LotteryUsingStrings

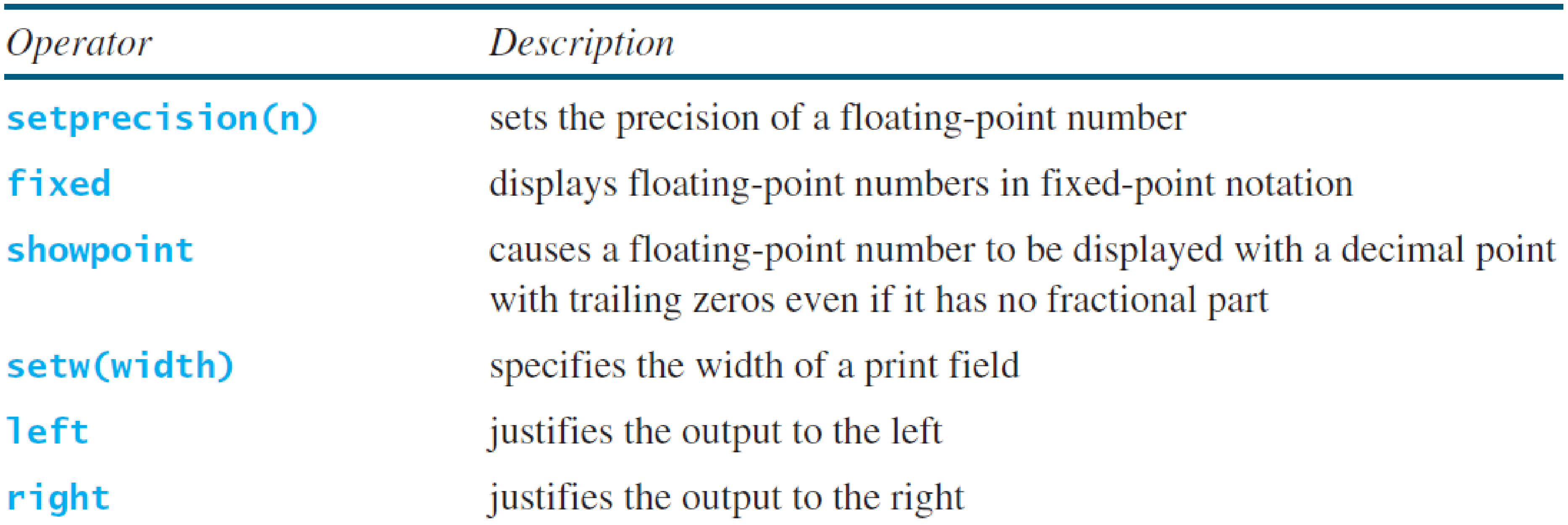
Run

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

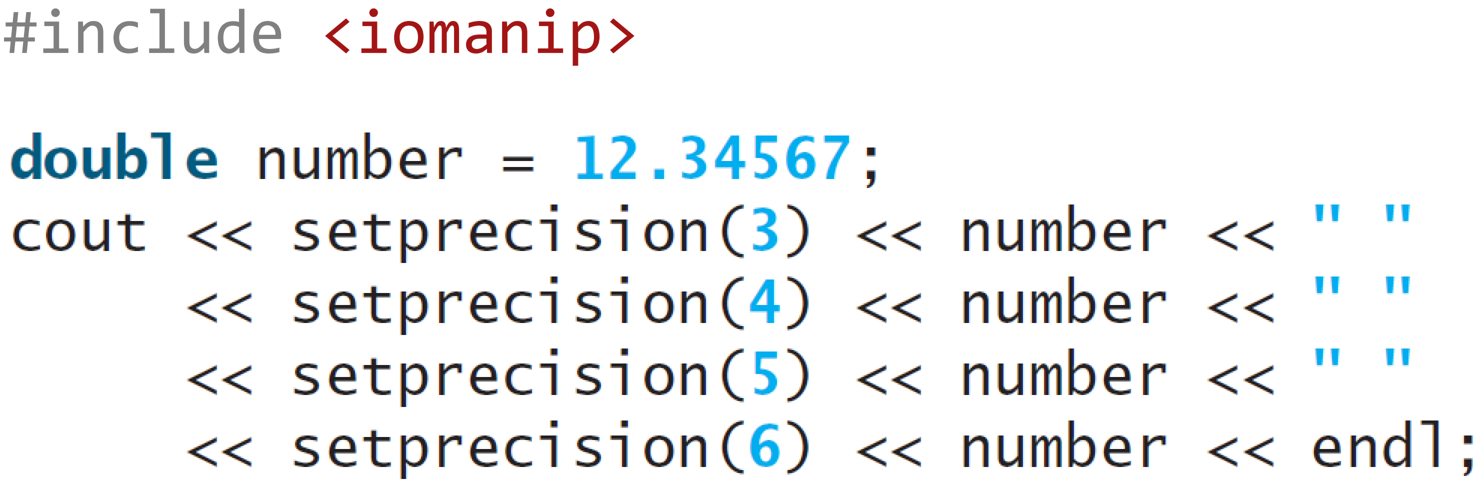
# Formatting Console Output

*You can use the stream manipulators to display formatted output on the console.*



| *Operator* | *Description* |
|---|---|
| `setprecision(n)` | sets the precision of a floating-point number |
| `fixed` | displays floating-point numbers in fixed-point notation |
| `showpoint` | causes a floating-point number to be displayed with a decimal point with trailing zeros even if it has no fractional part |
| `setw(width)` | specifies the width of a print field |
| `left` | justifies the output to the left |
| `right` | justifies the output to the right |

# setprecision(n) Manipulator



```
#include <iomanip>

double number = 12.34567;
cout << setprecision(3) << number << " "
     << setprecision(4) << number << " "
     << setprecision(5) << number << " "
     << setprecision(6) << number << endl;
```

displays

```
12.3 12.35 12.346 12.3457
```

# `fixed` Manipulator

```
cout << 232123434.357;
```

displays

```
2.32123e+08
```

```
cout << fixed << 232123434.357;
```

displays

```
232123434.357000
```

```
cout << fixed << setprecision(2)
     << 232123434.357;
```

displays

```
232123434.36
```

# showpoint Manipulator

```
cout << setprecision(6);
cout << 1.23 << endl;
cout << showpoint << 1.23 << endl;
cout << showpoint << 123.0 << endl;
```

displays

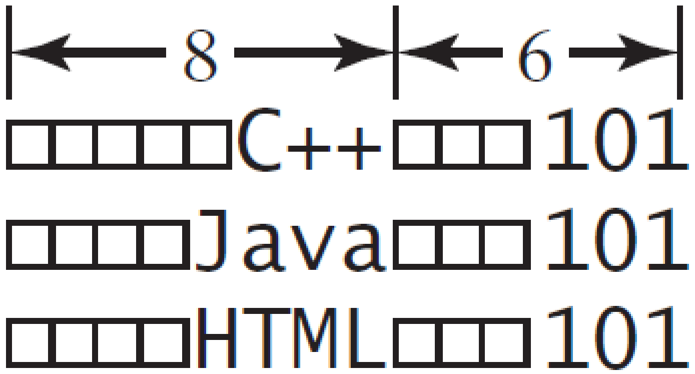
```
1.23
1.23000
123.000
```

# setw(width) Manipulator

```
cout << setw(8) << "C++" << setw(6) << 101 << endl;
cout << setw(8) << "Java" << setw(6) << 101 << endl;
cout << setw(8) << "HTML" << setw(6) << 101 << endl;
```

displays

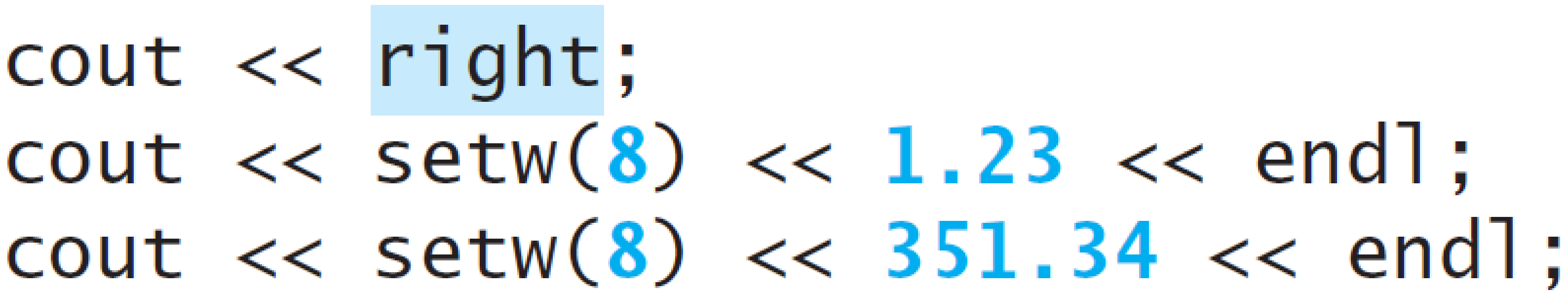
```
     C++   101
    Java   101
    HTML   101
```



```
cout << setw(8) << "Programming" << "#" << setw(2) << 101;
```

```
Prgramming#101
```

# `left` and `right` Manipulators

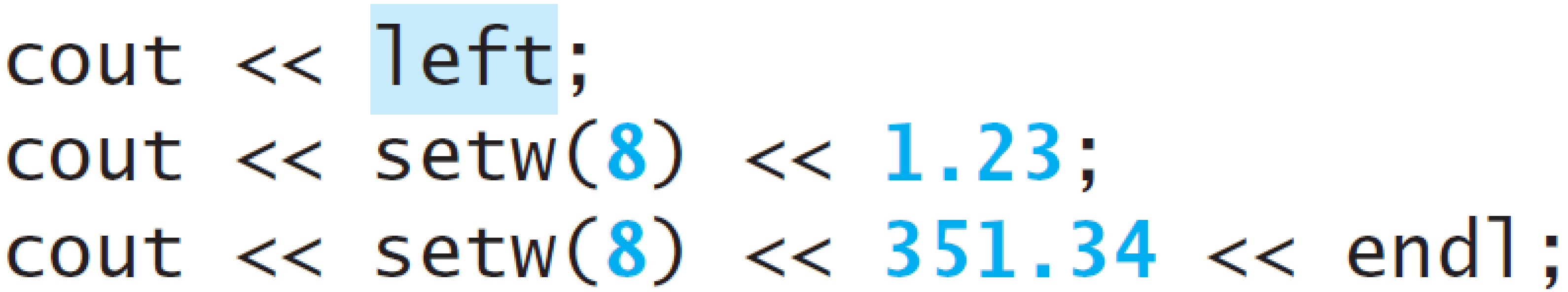


```
cout << right;
cout << setw(8) << 1.23 << endl;
cout << setw(8) << 351.34 << endl;
```

displays

```
□□□□1.23
□□351.34
```

# left and right Manipulators



```
cout << left;
cout << setw(8) << 1.23;
cout << setw(8) << 351.34 << endl;
```

displays

```
1.23□□□□351.34□□
```

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output

# Simple File Output

To write data to a file, first declare a variable of the **ofstream** type:

```
#include <fstream>
ofstream output;
```

To specify a file, invoke the **open** function from **output** object as follows:

```
output.open("numbers.txt");
```

Optionally, you can create a file output object and open the file in one statement like this:

```
ofstream output("numbers.txt");
```

To write data, use the stream insertion operator (**<<**) in the same way that you send data to the **cout** object. For example,

```
output << 95 << " " << 56 << " " << 34 << endl;
```

Finally:

```
output.close();
```

SimpleFileOutput

Run

# Simple File Input

To read data from a file, first declare a variable of the **`ifstream`** type:

```
#include <fstream>
ifstream input;
```

To specify a file, invoke the **`open`** function from **`input`** as follows:

```
input.open("numbers.txt");
```

Or:

```
ifstream input("numbers.txt");
```

To read data, use the stream extraction operator (**>>**) in the same way that you read data from the cin object. For example,

```
input >> score1 >> score2 >> score3;
```

Finally:

```
input.close();
```

SimpleFileInput

Run

# Outline

- Introduction
- Mathematical Functions
- Character Data Type and Operations
- Case Study: Generating Random Characters
- Case Study: Guessing Birthdays
- Character Functions
- Case Study: Converting Hexadecimal Decimal
- The string Type
- Case Study: Revising the Lottery Program Using Strings
- Formatting Console Output
- Simple File Input and Output