

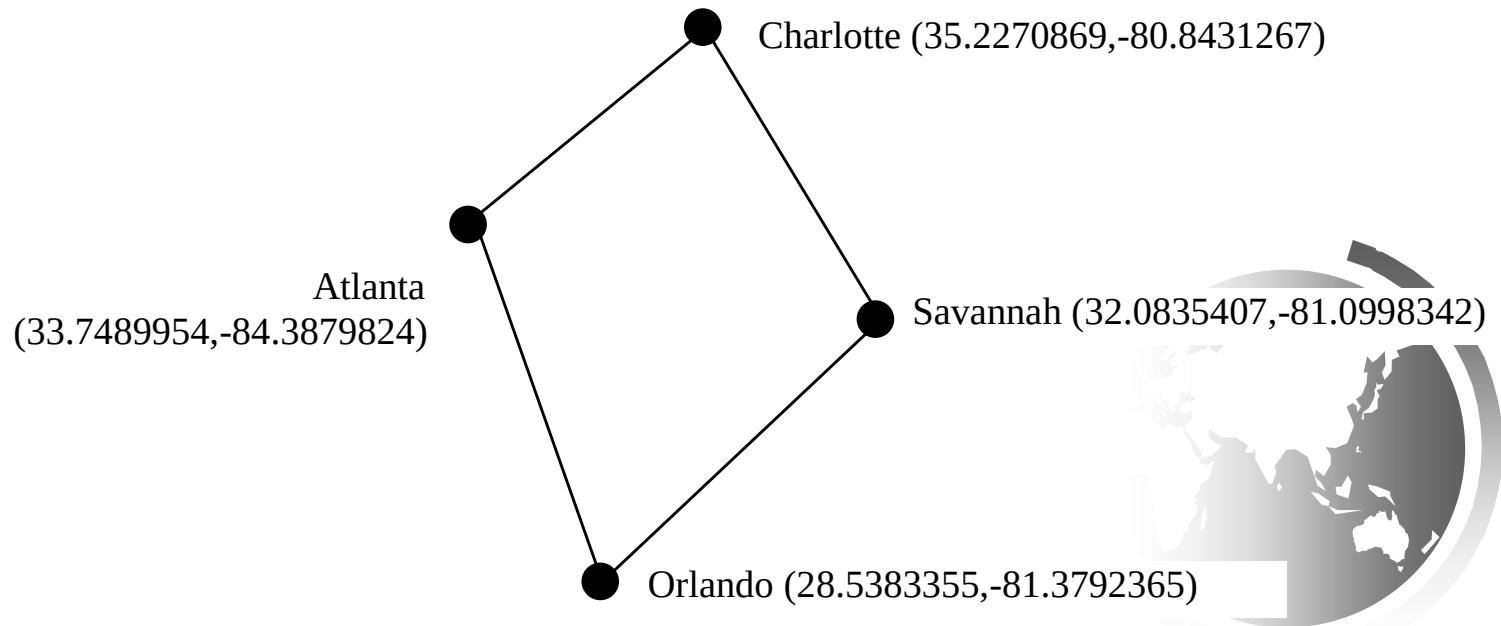


Chapter 4

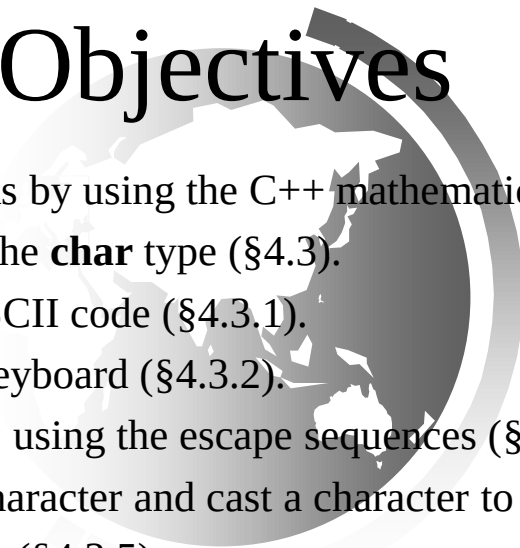
Mathematical Functions, Characters, and Strings

Motivations

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.



Objectives



- ☞ To solve mathematics problems by using the C++ mathematical functions (§4.2).
- ☞ To represent characters using the **char** type (§4.3).
- ☞ To encode characters using ASCII code (§4.3.1).
- ☞ To read a character from the keyboard (§4.3.2).
- ☞ To represent special characters using the escape sequences (§4.3.3).
- ☞ To cast a numeric value to a character and cast a character to an integer (§4.3.4).
- ☞ To compare and test characters (§4.3.5).
- ☞ To program using characters (**DisplayRandomCharacter**, **GuessBirthday**) (§§4.4-4.5).
- ☞ To test and convert characters using the C++ character functions (§4.6).
- ☞ To convert a hexadecimal character to a decimal value (**HexDigit2Dec**) (§4.7).
- ☞ To represent strings using the **string** type and introduce objects and instance functions (§4.8).
- ☞ To use the subscript operator for accessing and modifying characters in a string (§4.8.1).
- ☞ To use the + operator to concatenate strings (§4.8.2).
- ☞ To compare strings using the relational operators (§4.8.3).
- ☞ To read strings from the keyboard (§4.8.4).
- ☞ To revise the lottery program using strings (**LotteryUsingStrings**) (§4.9).
- ☞ To format output using stream manipulators (§4.10).
- ☞ To read and write data from/to a file (§4.11).

Mathematical Functions

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



Trigonometric Functions

Function	Description
<code>sin(radians)</code>	Returns the trigonometric sine of an angle in radians.
<code>cos(radians)</code>	Returns the trigonometric cosine of an angle in radians.
<code>tan(radians)</code>	Returns the trigonometric tangent of an angle in radians.
<code>asin(a)</code>	Returns the angle in radians for the inverse of sine.
<code>acos(a)</code>	Returns the angle in radians for the inverse of cosine.
<code>atan(a)</code>	Returns the angle in radians for the inverse of tangent.



Exponent Functions

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



Rounding Functions

Function	Description
<code>ceil(x)</code>	x is rounded up to its nearest integer. This integer is returned as a double value.
<code>floor(x)</code>	x is rounded down to its nearest integer. This integer is returned as a double value.

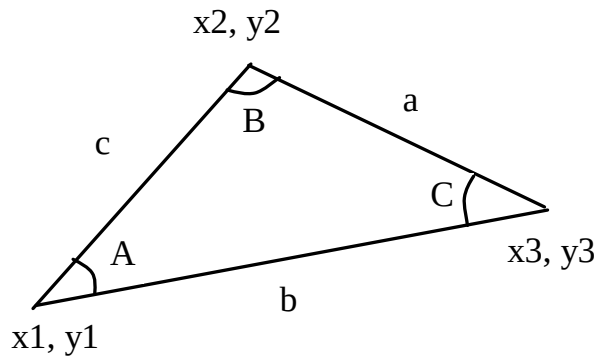


The min, max, and abs Functions

- ➡ `max(2, 3)` returns **3**
- ➡ `max(2.5, 3.0)` returns **3.0**
- ➡ `min(2.5, 4.6)` returns **2.5**
- ➡ `abs(-2)` returns **2**
- ➡ `abs(-2.1)` returns **2.1**



Case Study: Computing Angles of a Triangle



$$A = \arccos((a * a - b * b - c * c) / (-2 * b * c))$$

$$B = \arccos((b * b - a * a - c * c) / (-2 * a * c))$$

$$C = \arccos((c * c - b * b - a * a) / (-2 * a * b))$$

Write 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

ComputeAngles

```

#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));

    // Obtain three angles in degrees
    const double PI = 3.14159;
    double A = acos((a * a - b * b - c * c) / (-2 * b * c)) * 180 / PI;
    double B = acos((b * b - a * a - c * c) / (-2 * a * c)) * 180 / PI;
    double C = acos((c * c - b * b - a * a) / (-2 * a * b)) * 180 / PI;

    // Display the angles in degrees
    cout << "The three angles are " << round(A * 100) / 100.0 << " "
         << round(B * 100) / 100.0 << " " << round(C * 100) / 100.0 << endl;

    return 0;
}

```



Character Data Type

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

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

NOTE: 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;
```



Read Characters

To read a character from the keyboard, use

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



Escape Sequences for Special Characters

Character	Escape Sequence	Name	ASCII Code
\b		Backspace	8
\t		Tab	9
\n		Linefeed	10
\f		Formfeed	12
\r		Carriage Return	13
\\		Backslash	92
\'		Single Quote	39
\"		Double Quote	34

Appendix B: ASCII Character Set

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

TABLE B.1 ASCII Character Set in the Decimal Index

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		

ASCII Character Set, cont.

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

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	“	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	del



Casting between char and Numeric Types

```
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. A char operand is automatically cast into a number if the other operand is a number or a character. For example, the following statements

```
int i = '2' + '3'; // (int)'2' is 50 and (int)'3' is 51
cout << "i is " << i << endl; // i is decimal 101
int j = 2 + 'a'; // (int)'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



Note

It is worthwhile to note that 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. Furthermore, the ASCII code for 'a' is greater than the code for 'A'. So 'a' - 'A' is the same as 'b' - 'B'. For a lowercase letter *ch*, its corresponding uppercase letter is `static_cast<char>('A' + (ch - 'a'))`.



Problem: Converting a Lowercase to Uppercase

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

ToUppercase

ToUppercase



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

int main()
{
    char lowercaseLetter;
    cout << "Enter a lowercase letter: ";
    cin >> lowercaseLetter;

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

    cout << "The corresponding uppercase letter is "
        << uppercaseLetter << endl;

    return 0;
}
```



Comparing and Testing Characters

Two characters can be compared using the comparison operators just like comparing two numbers. This is done by comparing the ASCII codes of the two characters.

'a' < 'b' is true because the ASCII code for **'a'** (**97**) is less than the ASCII code for **'b'** (**98**).

'a' < 'A' is false because the ASCII code for **'a'** (**97**) is less than the ASCII code for **'A'** (**65**).

'1' < '8' is true because the ASCII code for **'1'** (**49**) is less than the ASCII code for **'8'** (**56**).



Case Study: Generating Random Characters

Computer programs process numerical data and characters. You have seen many examples that involve numerical data. It is also important to understand characters and how to process them.

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. You learned how to generate a random number in §3.8. Recall that you can use the srand(seed) function to set a seed and use rand() to return a random integer. You can use it to write a simple expression to generate random numbers in any range. For example,

Case Study: Generating Random Characters, cont.

`rand() % 10`



Returns a random integer between 0 and 9.

`50 + rand() % 50`



Returns a random integer between 50 and 99.

`a + rand() % b`



Returns a random number between a and a + b, excluding a + b.

DisplayRandomCharacter

DisplayRandomCharacter



```

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    cout << "Enter a starting character: ";
    char startChar;
    cin >> startChar;

    cout << "Enter an ending character: ";
    char endChar;
    cin >> endChar;

    // 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;

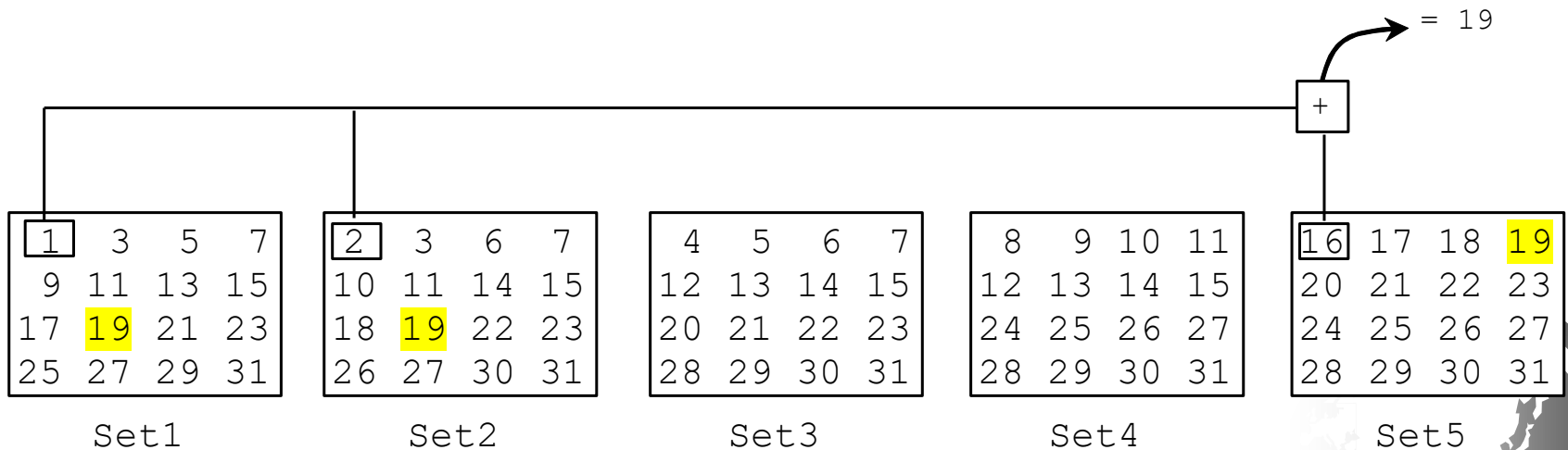
    return 0;
}

```



Case Study: Guessing Birthdays

This section uses the if statements to write an interesting game program. 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:



GuessBirthday

GuessBirthday

Character Functions

Function	Description
<code>isdigit(ch)</code>	Returns true if the specified character is a digit.
<code>isalpha(ch)</code>	Returns true if the specified character is a letter.
<code>isalnum(ch)</code>	Returns true if the specified character is a letter or digit.
<code>islower(ch)</code>	Returns true if the specified character is a lowercase letter.
<code>isupper(ch)</code>	Returns true if the specified character is an uppercase letter.
<code>isspace(ch)</code>	Returns true if the specified character is a whitespace character.
<code>tolower(ch)</code>	Returns the lowercase of the specified character.
<code>toupper(ch)</code>	Returns the uppercase of the specified character.

CharacterFunctions

CharacterFunctions



```

#include <iostream>
#include <cctype>
using namespace std;

int main()
{
    cout << "Enter a character: ";
    char ch;
    cin >> ch;

    cout << "You entered " << ch << endl;

    if (islower(ch))
    {
        cout << "It is a lowercase letter " << endl;
        cout << "Its equivalent uppercase letter is " <<
            static_cast<char>(toupper(ch)) << endl;
    }
    else if (isupper(ch))
    {
        cout << "It is an uppercase letter " << endl;
        cout << "Its equivalent lowercase letter is " <<
            static_cast<char>(tolower(ch)) << endl;
    }
    else if (isdigit(ch))
    {
        cout << "It is a digit character " << endl;
    }

    return 0;
}

```



Case Study: Converting a Hexadecimal Digit to a Decimal Value

Write a program that converts a hexadecimal digit into a decimal value.

HexDigit2Dec

HexDigit2Dec



```

#include <iostream>
#include <cctype>
using namespace std;

int main()
{
    cout << "Enter a hex digit: ";
    char hexDigit;
    cin >> hexDigit;

    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;
    }
    else
    {
        cout << hexDigit << " is an invalid input" << endl;
    }

    return 0;
}

```

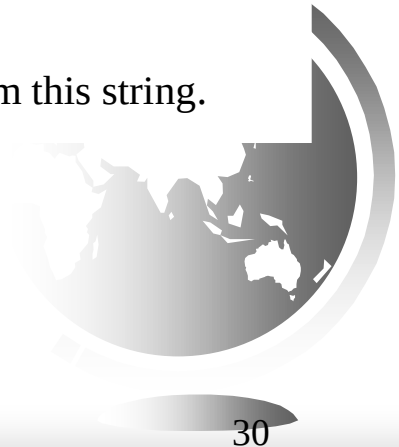


The string Type

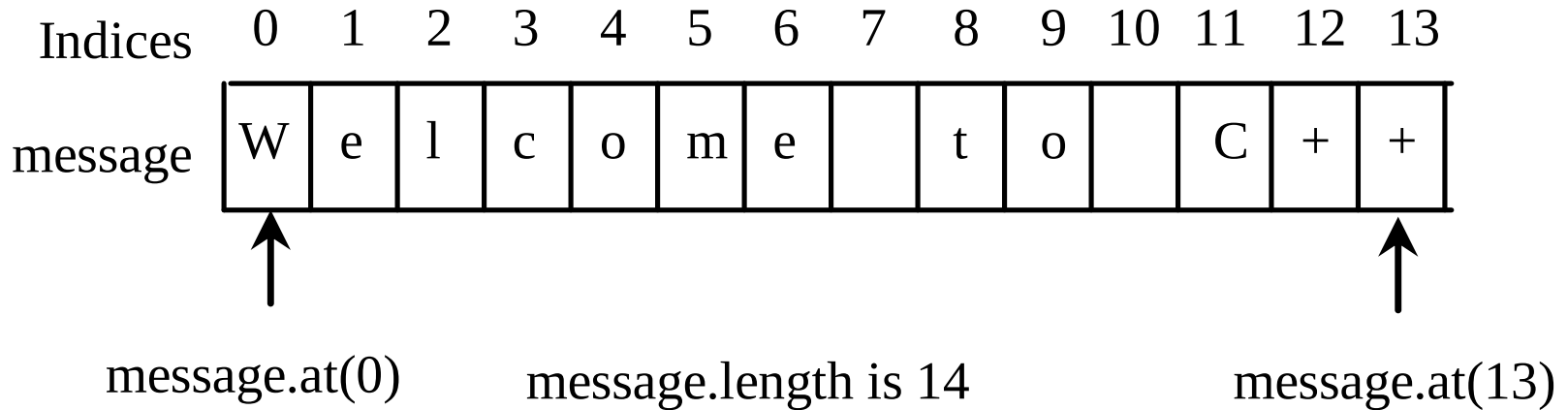
```
string s;
```

```
string message = "Programming is fun";
```

Function	Description
<code>length()</code>	Returns the number of characters in this string.
<code>size()</code>	Same as <code>length()</code> ;
<code>at(index)</code>	Returns the character at the specified index from this string.



String Subscript Operator



For convenience, C++ provides the subscript operator for accessing the character at a specified index in a string using the syntax **stringName[index]**. You can use this syntax to retrieve and modify the character in a string.

String Subscript Operator

```
string s = "ABCD";  
s[0] = 'P';  
cout << s[0] << endl;
```



Concatenating Strings

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

```
message += " and programming is fun";
```



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

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

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



Example

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

OrderTwoCities

OrderTwoCities



```
#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;
}
```



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.

LotteryUsingStrings

LotteryUsingStrings



```

#include <iostream>
#include <string> // for using strings
#include <ctime> // for time function
#include <cstdlib> // for rand and srand functions
using namespace std;

int main()
{
    string lottery;
    srand(time(0));
    int digit = rand() % 10; // Generate first digit
    lottery += static_cast<char>(digit + '0');
    digit = rand() % 10; // Generate second digit
    lottery += static_cast<char>(digit + '0');

    // Prompt the user to enter a guess
    cout << "Enter your lottery pick (two digits): ";
    string guess;
    cin >> guess;

    cout << "The lottery number is " << lottery << endl;

    // 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;

    return 0;
}

```



Formatting Console Output

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



setprecision(n) Manipulator

```
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

```
double monthlyPayment = 345.4567;  
double totalPayment = 78676.887234;  
cout << fixed << setprecision(2)  
      << monthlyPayment << endl  
      << totalPayment << endl;
```

displays

345.46

78676.89



showpoint Manipulator

```
cout << setprecision(6);
```

```
cout << 1.23 << endl;
```

```
cout << showpoint << 1.23 << endl;
```

```
cout << showpoint << 123.0 << endl;
```

displays

1.23

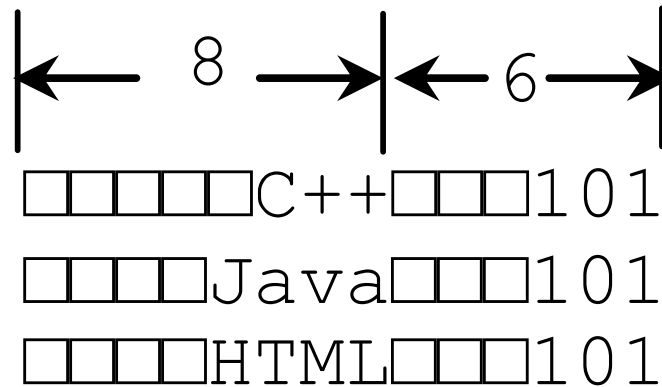
1.230000

123.0000



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
```



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□□



Example: Display a Table

Degrees	Radians	Sine	Cosine	Tangent
30	0.5236	0.5000	0.8660	0.5773
60	1.0472	0.8660	0.5000	1.7320

FormatDemo

FormatDemo



```

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

int main()
{
    // Display the header of the table
    cout << left << setw(10) << "Degrees" << setw(10) << "Radians"
         << setw(10) << "Sine" << setw(10) << "Cosine" << setw(10)
         << "Tangent" << endl;

    // Display values for 30 degrees
    const double PI = 3.14159;
    double degrees = 30;
    double radians = degrees * (PI / 180);
    cout << setw(10) << degrees << setw(10) << fixed
         << setprecision(4) << radians << setw(10) << sin(radians)
         << setw(10) << cos(radians) << setw(10) << tan(radians) << endl;

    // Display values for 60 degrees
    degrees = 60;
    radians = degrees * (PI / 180);
    cout << setw(10) << setprecision(0) << degrees << setw(10)
         << setprecision(4) << radians << setw(10)
         << sin(radians) << setw(10) << cos(radians) << setw(10)
         << tan(radians) << endl;

    return 0;
}

```



Simple File Output

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



SimpleFileOutput

SimpleFileOutput

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

int main()
{
    ofstream output;

    // Create a file
    output.open("numbers.txt");

    // Write numbers
    output << 95 << " " << 56 << " " << 34;

    // Close file
    output.close();

    cout << "Done" << endl;

    return 0;
}
```



Simple File Input

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

```
ifstream input;
```

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

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

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

```
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;
```



SimpleFileInput

SimpleFileInput

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

int main()
{
    ifstream input;

    // Open a file
    input.open("numbers.txt");

    int score1, score2, score3;

    // Read data
    input >> score1;
    input >> score2;
    input >> score3;

    cout << "Total score is " << score1 + score2 + score3 << endl;

    // Close file
    input.close();

    cout << "Done" << endl;
    return 0;
}
```

