# 13 Stream Input/Output: A Deeper Look

## Objectives

In this chapter you'll:

- Use C++ object-oriented stream input/output.
- Perform input and output of individual characters.
- Use unformatted I/O for high performance.
- Use stream manipulators to display integers in octal and hexadecimal formats.
- Specify precision for both input and output.
- Display floating-point values in both scientific and fixed-point notation.
- Set and restore the format state.
- Control justification and padding.
- Determine the success or failure of input/output operations.
- Tie output streams to input streams.

## Self-Review Exercises

**13.1** *(Fill in the Blanks)* Answer each of the following:

a) Input/output in C++ occurs as _____ of bytes.

**ANS:** streams.

b) The stream manipulators for justification are _____, _____ and _____.

**ANS:** `left`, `right` and `internal`.

c) Member function _____ can be used to set and reset format state.

**ANS:** `flags`.

d) Most C++ programs that do I/O should include the _____ header that contains the declarations required for all stream-I/O operations.

**ANS:** `<iostream>`.

e) When using parameterized manipulators, the header _____ must be included.

**ANS:** `<iomanip>`.

f) The _____ stream manipulator causes positive numbers to display with a plus sign.

**ANS:** `showpos`.

g) The `ostream` member function _____ is used to perform unformatted output.

**ANS:** `write`.

h) Input operations are supported by class _____.

**ANS:** `istream`.

i) Standard error stream outputs are directed to the stream objects _____ or _____.

**ANS:** `cerr` or `clog`.

j) Output operations are supported by class _____.

**ANS:** `ostream`.

k) The symbol for the stream insertion operator is _____.

**ANS:** `<<`.

l) The four objects that correspond to the standard devices on the system include _____, _____, _____ and _____.

**ANS:** `cin`, `cout`, `cerr` and `clog`.

m) The symbol for the stream extraction operator is _____.

**ANS:** `>>`.

n) The stream manipulators _____, _____ and _____ specify that integers should be displayed in octal, hexadecimal and decimal formats, respectively.

**ANS:** `oct`, `hex` and `dec`.

**13.2** *(True or False)* State whether each of the following is *true* or *false*. If the answer is *false*, explain why.

a) The stream member function `flags` with a `long` argument sets the `flags` state variable to its argument and returns its previous value.

**ANS:** False. The stream member function `flags` with a `fmtflags` argument sets the `flags` state variable to its argument and returns the prior state settings.

b) The stream insertion operator `<<` and the stream extraction operator `>>` are overloaded to handle all standard data types—including strings and memory addresses (stream insertion only)—and all user-defined data types.

**ANS:** False. The stream insertion and stream extraction operators are not overloaded for all user-defined types. You must specifically provide the overloaded operator functions to overload the stream operators for use with each user-defined type you create.

c) The stream member function `flags` with no arguments resets the stream's format state.

**ANS:** False. The stream member function `flags` with no arguments returns the current format settings as a `fmtflags` data type, which represents the format state.

d) Input with the stream extraction operator >> always skips leading white-space characters in the input stream, by default.

**ANS:** True.

e) The stream member function `rdstate` returns the current state of the stream.

**ANS:** True.

f) The `cout` stream normally is connected to the display screen.

**ANS:** True.

g) The stream member function `good` returns `true` if the `bad`, `fail` and `eof` member functions all return `false`.

**ANS:** True.

h) The `cin` stream normally is connected to the display screen.

**ANS:** False. The `cin` stream is connected to the standard input of the computer, which normally is the keyboard.

i) If a nonrecoverable error occurs during a stream operation, the `bad` member function will return `true`.

**ANS:** True.

j) Output to `cerr` is unbuffered and output to `clog` is buffered.

**ANS:** True.

k) Stream manipulator `showpoint` forces floating-point values to print with the default six digits of precision unless the precision value has been changed, in which case floating-point values print with the specified precision.

**ANS:** True.

l) The `ostream` member function `put` outputs the specified number of characters.

**ANS:** False. The `ostream` member function `put` outputs its single-character argument.

m) The stream manipulators `dec`, `oct` and `hex` affect only the next integer output operation.

**ANS:** False. The stream manipulators `dec`, `oct` and `hex` set the output format state for integers to the specified base until the base is changed again or the program terminates.

**13.3** ***(Write a C++ Statement)*** For each of the following, write a single statement that performs the indicated task.

a) Output the string `"Enter your name: "`.

**ANS:**
```
cout << "Enter your name: ";
```

b) Use a stream manipulator that causes the exponent in scientific notation and the letters in hexadecimal values to print in capital letters.

**ANS:**
```
cout << uppercase;
```

c) Output the address of the variable `myString` of type `char *`.

**ANS:**
```
cout << static_cast<void*>(myString);
```

d) Use a stream manipulator to ensure that floating-point values print in scientific notation.

**ANS:**
```
cout << scientific;
```

e) Output the address in variable `integerPtr` of type `int *`.

**ANS:**
```
cout << integerPtr;
```

f) Use a stream manipulator such that, when integer values are output, the integer base for octal and hexadecimal values is displayed.

**ANS:**
```
cout << showbase;
```

g) Output the value pointed to by `floatPtr` of type `float *`.

**ANS:**
```
cout << *floatPtr;
```

h) Use a stream member function to set the fill character to `'*'` for printing in field widths larger than the values being output. Repeat this statement with a stream manipulator.

**ANS:**
```
cout.fill('*');
cout << setfill('*');
```

i) Output the characters `'O'` and `'K'` in one statement with `ostream` function `put`.

**ANS:**
```
cout.put('O').put('K');
```

j) Get the value of the next character to input without extracting it from the stream.

**ANS:**
```
cin.peek();
```

k) Input a single character into variable `charValue` of type `char`, using the `istream` member function `get` in two different ways.

**ANS:**
```
charValue = cin.get();
cin.get(charValue);
```

l) Input and discard the next six characters in the input stream.

**ANS:**
```
cin.ignore(6);
```

m) Use `istream` member function `read` to input 50 characters into `char` array `line`.

**ANS:**
```
cin.read(line, 50);
```

n) Read 10 characters into character array `name`. Stop reading characters if the `'.'` delimiter is encountered. Do not remove the delimiter from the input stream. Write another statement that performs this task and removes the delimiter from the input.

**ANS:**
```
cin.get(name, 10, '.');
cin.getline(name, 10, '.');
```

o) Use the `istream` member function `gcount` to determine the number of characters input into character array `line` by the last call to `istream` member function `read`, and output that number of characters, using `ostream` member function `write`.

**ANS:**
```
cout.write(line, cin.gcount());
```

p) Output `124`, `18.376`, `'Z'`, `1000000` and `"String"`, separated by spaces.

**ANS:**
```
cout << 124 << ' ' << 18.376 << " Z " << 1000000 << " String";
```

q) Display `cout`'s current precision setting.

**ANS:**
```
cout << cout.precision();
```

r) Input an integer value into `int` variable `months` and a floating-point value into `float` variable `percentageRate`.

**ANS:**
```
cin >> months >> percentageRate;
```

s) Print `1.92`, `1.925` and `1.9258` separated by tabs and with `3` digits of precision, using a stream manipulator.

**ANS:**
```
cout << setprecision(3) << 1.92 << '\t' << 1.925 << '\t' << 1.9258;
```

t) Print integer `100` in octal, hexadecimal and decimal, using stream manipulators and separated by tabs.

**ANS:**
```
cout << oct << 100 << '\t' << hex << 100 << '\t' << dec << 100;
```

u) Print integer `100` in decimal, octal and hexadecimal separated by tabs, using a stream manipulator to change the base.

**ANS:**
```
cout << 100 << '\t' << setbase(8) << 100 << '\t' << setbase(16) << 100;
```

v) Print `1234` right justified in a `10`-digit field.

**ANS:**
```
cout << setw(10) << 1234;
```

w) Read characters into character array `line` until the character `'z'` is encountered, up to a limit of `20` characters (including a terminating null character). Do not extract the delimiter character from the stream.

**ANS:**
```
cin.get(line, 20, 'z');
```

x) Use integer variables `x` and `y` to specify the field width and precision used to display the `double` value `87.4573`, and display the value.

**ANS:**
```
cout << setw(x) << setprecision(y) << 87.4573;
```

**13.4** ***(Find and Correct Code Errors)*** Identify the error in each of the following statements and explain how to correct it.

a) `cout << "Value of x <= y is: " << x <= y;`

**ANS:** *Error:* The precedence of the << operator is higher than that of <=, which causes the statement to be evaluated improperly and also causes a compiler error.
*Correction:* Place parentheses around the expression `x <= y`.

b) The following statement should print the integer value of `'c'`.
`cout << 'c';`

**ANS:** *Error:* In C++, characters are not treated as small integers, as they are in C.
*Correction:* To print the numerical value for a character in the computer's character set, the character must be cast to an integer value, as in the following:
```
cout << static_cast<int>('c');
```

c) `cout << ""A string in quotes"";`

**ANS:** *Error:* Quote characters cannot be printed in a string unless an escape sequence is used.
*Correction:* Print the string:
```
cout << "\"A string in quotes\"";
```

**13.5** *(Show Outputs)* For each of the following, show the output.

a)
```
cout << "12345\n";
cout.width(5);
cout.fill('*');
cout << 123 << "\n" << 123;
```
**ANS:**
```
12345
**123
123
```

b) `cout << setw(10) << setfill('$') << 10000;`

**ANS:** `$$$$$10000`

c) `cout << setw(8) << setprecision(3) << 1024.987654;`

**ANS:** `1024.988`

d) `cout << showbase << oct << 99 << "\n" << hex << 99;`

**ANS:**
```
0143
0x63
```

e) `cout << 100000 << "\n" << showpos << 100000;`

**ANS:**
```
100000
+100000
```

f) `cout << setw(10) << setprecision(2) << scientific << 444.93738;`

**ANS:** `4.45e+002`

## Exercises

*NOTE: Solutions to the programming exercises are located in the* **`ch13solutions`** *folder.*

**13.6** *(Write C++ Statements)* Write a statement for each of the following:

a) Print integer `40000` left justified in a 15-digit field.

**ANS:** `cout << left << setw(15) << 40000 << '\n';`

b) Read a string into character array variable `state`.

**ANS:** `cin >> state;`

c) Print `200` with and without a sign.

**ANS:** `cout << showpos << 200 << setw(4) << '\n' << noshowpos << 200 << '\n';`

d) Print the decimal value `100` in hexadecimal form preceded by `0x`.

**ANS:** `cout << showbase << hex << 100 << '\n';`

e) Read characters into array `charArray` until the character `'p'` is encountered, up to a limit of 10 characters (including the terminating null character). Extract the delimiter from the input stream, and discard it.

**ANS:**
```
cin.getline(charArray, 10, 'p');
```

f) Print `1.234` in a 9-digit field with preceding zeros.

**ANS:**
```
cout << fixed << showpoint  << setw(9)
        << setfill('0') << internal << 1.234 << '\n';
```