Input/Output in C++

Chapter 14

Objectives

- **Use the iostream library for input and output for a**
- **Do formatted I/O with iostreams.**
- **Outline the inheritance hierarchy of the principal streams classes.**
- **Do file I/O using streams.**
- **Overload operators >> and << to do I/O in your   own classes.**
- **Gain experience through code walk-throughs and lab exercises.**
    - The example programs are in the **chapter directory.**
    - Labs located in Labs/Lab14

Input/Output in C++

- **The C++ language itself does not define input/output.**
- **Input/output is implemented in a standard library called the *iostream* library.**
- **The iostream library provides a set of operations for reading and writing of the built-in data types.**
- **The programmer can extend certain of these operations to do input and output of class types.**
- **To use the iostream library include the header file <*iostream*> .**

Built-in Stream Objects

- **Input stream object, belonging to class *istream*:**

cin	**standard input**

- **Output stream objects, belonging to class *ostream*:**

cout	**standard output**
cerr	**standard error (unbuffered)**
clog	**standard error (buffered)**

Output Operator <<

- **Output is performed by the insertion operator <<.**
- **Built-in types such as int, char, char* are supported:**

```
cout  <<  97;
cout  <<  'A';
cout  <<  "Hello";
```

- **Insertion operations can be concatenated into a single statement:**

```
cout << 97 << 'A' << "Hello";
```

- **endl can be used for newline, has the effect of flushing the buffer :**

```
cout  <<  "Hello, world"  <<  endl;
```

Input Operator >>

- Input is performed by the extraction operator >>.
- You do not need to use & as with C scanf function.

```
int num;
cin  >>  num;
```

- Note direction suggested by << and >>:

```
cout  <<  num       (num ---> output)
cin  >>  num        (input ---> num).
```

- Extraction operator, like insertion operator, can be concatenated:

```
cin  >>  num1  >>  num2;
```

Character Input

- **Extraction operator >> skips over white space.**

```
char  ch;
while  (cin >> ch)        // false at EOF
```

- **To read individual characters including blanks, use member function get.**

```
char  ch;
while  (cin.get(ch))
```

- **To read a string or a line use getline (see echostr as an example)**

String Input

- **Extraction operator >> applied to a char * variable** reads a string delimited by white space.
  - A null byte is appended to string.

```
char  buf[80];
for  (int i = 0; i < 3; ++i){
    cin  >>  buf;
    cout  <<  buf  << endl;
}
```

Formatted I/O

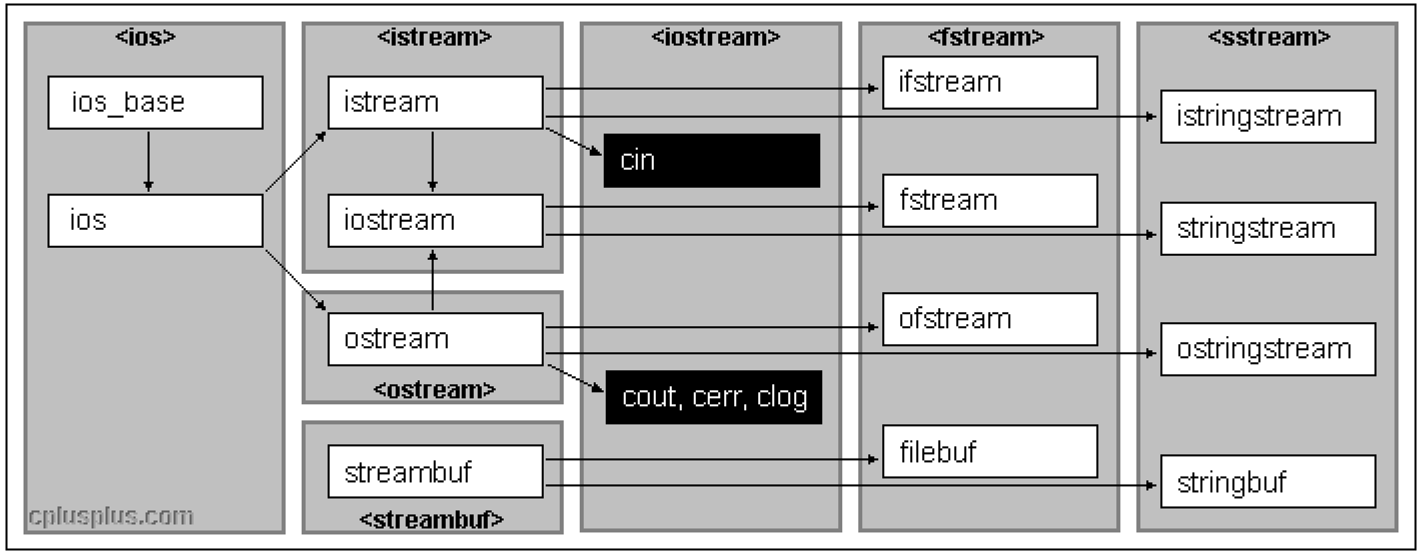
- **The iostream library supports a very extensive set of formatting facilities.**
    - One programming example is presented on the next page.
- **A workable strategy for C programmers is to use the C functions *sprintf* and *sscanf* for formatting to/from strings, and to use the iostream library for the actual I/O:**

```
float  x;
char  buf[80];
sprintf(buf, "%10.4f", x);
cout  <<  buf;
```

Formatted I/O Demo

- **Review and run the example in the format folder.**

Streams Hierarchy (Simplified)



File I/O

- **Include header file <fstream>**
- **Output file stream class ofstream**
    - **Constructor**
    - **<<, other operators**
    - **open, close**
- **Input file stream class ifstream**
    - **Constructors**
    - **>>, other operations**
    - **open, close**

File Opening

- Constructor can both create a stream and open file:

```
ofstream  out("file1.out");
```

- Constructor can create stream and subsequently connect stream to a file:

```
ifstream  in;
in.open("file1.in");
in.close();
```

File Opening (continued)

- **Test for success of open operation by checking for non-zero stream:**

```
ofstream  in("nofile.xxx");
if ( ! in ) { /* error in opening file */ }
```

- **Non-default file opening modes can be specified by an optional argument using enumeration constants in class ios:**

```
ofstream  out("file.out", ios::app);// opens in append mode
```

File Copy demos

- **Review and run the example in the intcopy folder.**
- **Review and run the example in the charcopy folder.**

Demo Overloading Stream Operators

- **C++ is able to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator <<.**
- **The stream insertion and stream extraction operators also can be overloaded to perform input and output for user-defined types like an object.**
- **It is important to make operator overloading function a friend of the class because it would be called without creating an object.**
- **Review and run the sample program in the StringIO folder.**

Summary

- **Input/output is implemented in C++ in a standard library called the the *iostream* library.**
- **Built in streams *cin, cout, cerr* are available.**
- **File I/O can be performed by defining new streams via a constructor.**
- **Output can be performed by insertion operator << and input by the extraction operator >>.**
- **Additional I/O operations include *get, put, getline*.**
- **I/O is supported for standard data types such as *char, int, char **, etc.**
- **The I/O operations can be overloaded to be used with user defined types.**