3 Overloading << and >>

- The insertion operator << is a binary operator
  - The first operand is the output stream
  - The second operand is the value following <<

```
cout << "Hello out there.\n";
// Operand1(cout) Operand2(<<) Operand3("Hello out there.\n")
```

3.1 Replacing Function output

- Overloading the << **operator** allows us to use << instead of **TimeOfDay**'s **output** function
  - Given the declaration:

```
time1.output();
```

can become

```
cout << time1 << endl;
```

3.2 What Does << Return?

- Because << is a binary operator
  cout << "I get up " << time1 << " everyday.";
  seems as if it could be grouped as
  ((cout << "I get up ") << time1) << " everyday.";
- To provide **cout** as an argument for << **time1**,
  (cout << "I get up ") must return **cout**

3.3 Overloaded << Declaration

- Based on the previous example, << should return its first argument, the output stream
  - This leads to a declaration of the overloaded << operator for the **TimeOfDay** class:

```
class TimeOfDay
{
        public:
        :
        :
                friend ostream& operator << (ostream& strm, const TimeOfDay& t1);
```

3.4 Overloaded << Definition

- The following defines the << **operator**
  - It is almost same function body as **TimeOfDay::output** of old version.

```
std::ostream& operator << (std::ostream &strm, const TimeOfDay& t1)
{
    strm << std::setfill('0') << std::setw(2) << t1.hours << ':'
    << std::setfill('0') << std::setw(2) << t1.minutes;
    return strm;
}
```

3.4.1 Return ostream& ?

- The **&** means a **reference** is returned
  - So far all our functions have returned values
- The value of a stream object is not so simple to return

  - The value of a stream might be an entire file, the keyboard, or the screen!
- We want to return the stream itself, not the value of the stream
- The **&** means that we want to return the stream, not its value

3.5 Overloading >>

- Overloading the >> **operator** for input is very similar to overloading the << for output
  - >> could be declare this way for the **TimeOfDay** class

```
class TimeOfDay
{
        public:
        :
        :
                friend std::istream& operator >> (std::istream &strm, TimeOfDay& t1);
```

  - >> could be defined this way for the **TimeOfDay** class

```
std::istream& operator >> (std::istream &strm, TimeOfDay& t1)
{
    strm >> t1.hours >> t1.minutes;
    return strm;
}
```