Chapter 10: Input and Output Streams

Plan for today

- We will talk about:
  - The temperature readings program and I/O error handling
  - In-class practice from the previous meeting
    - `tellg()` function
    - `seekg()` function
  - User-defined output: `operator<<()`
  - User-defined input: `operator>>()`

User-defined output: operator<<()

- Usually pretty trivial. Recall our **`Date`** class.

User-defined output: operator<<()

- Usually pretty trivial. Recall our **`Date`** class.

```
ostream& operator<<(ostream& out, const Date& d) {
     out << MonthsNames[int(d.m) - 1] << " "
         << d.d << ", " << d.y;
     return out;
   }
```

User-defined output: operator<<()

- Usually pretty trivial. Recall our **`Date`** class.

```
ostream& operator<<(ostream& out, const Date& d) {
     out << MonthsNames[int(d.m) - 1] << " "
         << d.d << ", " << d.y;
     return out;
   }
```

- we often use several different ways of outputting a value

- we may announce it as a friend, or may use public methods

User-defined output: operator<<()

- Usually pretty trivial. Recall our **Date** class.

```
ostream& operator<<(ostream& out, const Date& d) {
   out << MonthsNames[int(d.m) - 1] << " "
          << d.d << ", " << d.y;
   return out;
   }
```

- Uses:

```
cout << d1;              // means operator<<(cout,d1);
cout << d1 << d2;    // means (cout << d1)  <<  d2;
                  // means (operator<<(cout,d1)) <<  d2;
        // means operator<<((operator<<(cout,d1)), d2);
```

User-defined input: `operator>>()`

- Defining the input `operator >>` for a given type and input format is basically an exercise in error handling, hence can be tricky.

User-defined input: operator>>()

- Look at the suggested version (by the textbook):

```
istream& operator>>(istream& is, Date& dd)
{  // read date in format: ( year , month , day )
   int y, d, m;
   char ch1, ch2, ch3, ch4;
   is >> ch1 >> y >> ch2 >> m >> ch3 >> d >> ch4;
   if (!is) return is;  // we didn't get our values, so just leave
   if (ch1!='(' || ch2!=',' || ch3!=',' || ch4!=')')
    {  // oops: format error
         is.clear(ios_base::failbit);
          // something wrong: set state to fail()
         return is;    // and leave
    }
```

User-defined input: `operator>>()`

```
...
    dd = Date{y,Month(m),d};  // update dd
    return is; // and leave with is in the good()   state
}
```

In-class practice

Let's talk about rational numbers that can be written as a fraction $\frac{a}{b}$
Where *a* and *b* are integer values, and *b* is not zero!
Assume that we have a definition of the class hidden somewhere,
but we know that **num** and **den** are data attributes of the class, representing *a* and *b* correspondingly.

You may assume that we either have public methods **getNum()** and **getDen()** or we stated that both, input and output operators are friends of class Rational.

Write the definitions of the **operator<<** and **operator>>** for Rational class objects.

Resources used for these slides

- slides provided by B. Stroustrup at https://www.stroustrup.com/PPP2slides.html

- Class textbook