OVERLOADED operator<< AND operator>>

Input and Output functions

Delroy A. Brinkerhoff

REVIEWING FUNCTION OVERLOADING

- Overloaded functions must have unique argument lists:
  - `void f(int x);`
  - `void f(int x, int y);`
  - `int f(int x);`
  - `double f(int x);`
  - `double f(double x);`
  - `void print(ostream out, int x);`
  - `void print(ostream out, double x);`

<iostream>

class ostream

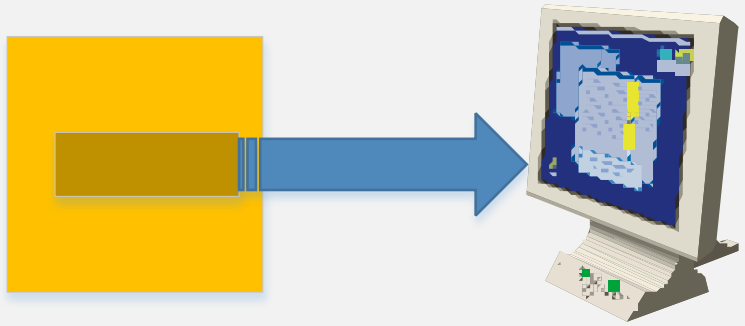
```
friend ostream& operator<<(ostream&, char);
friend ostream& operator<<(ostream&, char*);
friend ostream& operator<<(ostream&, short);
friend ostream& operator<<(ostream&, int);
friend ostream& operator<<(ostream&, long);
friend ostream& operator<<(ostream&, float);
friend ostream& operator<<(ostream&, double);
```

class istream

```
friend istream& operator>>(istream&, char&);
friend istream& operator>>(istream&, char*);
friend istream& operator>>(istream&, short&);
friend istream& operator>>(istream&, int&);
friend istream& operator>>(istream&, long&);
friend istream& operator>>(istream&, float&);
friend istream& operator>>(istream&, double&);
```

operator<<
THE INSERTER

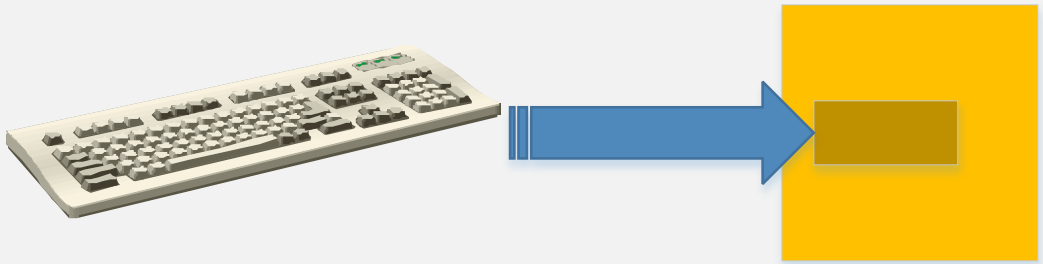
- ALWAYS a friend function
- ALWAYS follows the same pattern:
  - returns ostream reference
  - first argument ostream reference
  - second argument reference to the friending class



```
friend ostream& operator<<(ostream& out, foo& me)
{
    out << me.field << endl;
    return out;
}
```

operator>>
THE EXTRACTOR

- ALWAYS a friend function
- ALWAYS follows the same pattern:
  - returns istream reference
  - first argument istream reference
  - second argument reference to the friending class



```
friend istream& operator>>(istream& in, foo& me)
{
    in >> me.field;
    return in;
}
```