CS 368 Announcements
Wednesday, April 24, 2013

Program p5 – due Wednesday, 5/8, at 10:00 pm – 10%

Last Time

- continue Ch. 9 (Input and Output)
- file I/O - *fileIO.cpp*
- string class
- C strings

Today

- finish Ch. 9
- manipulators
- start Ch. 7 (Templates)
- templated functions
- templated classes

Next Time

- finish Ch. 7, start Ch. 10 (Collections: The STL)
- compiling with templates
- more template features
- start STL (containers)

Formatting Output Using Manipulators

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main() {
    int x = 1234;
    double y = 8763.1415;
    string str = "cs368";

    cout << "x is " << x << ", y is " << y <<", str is "
         << str << endl;





    cout << left << setw(20) << x << " "
         << right << setw(11) << setprecision(6)
         << fixed << y << " "
         << str << endl;





    cout << left << setw(20) << 97215 << " "
         << right << setw(11) << setprecision(6)
         << fixed << 12.34 << " "
         << "November" << endl;

    cout << "Enter a number and a string: ";
    cin >> x >> str;
    cout << "x is " << x << ", str is " << str << '\n';
}
```

Templated Functions

A function template is not a function but a pattern for what could become a function

How to write a function template:

```
template < generic_type_list >
// rest of function definition using types
// listed in the generic_type_list
```

where *`generic_type_list`* is a comma-separated list of

```
class name            or
typename name
```

Examples of Templated Functions

Example: `minimum`

```
int minimum( int x, int y ) {
    return (x < y) ? x : y;
}
```

Example: `swapIt`

Using Templated Functions

```
int x1 = 5, y1 = 9;
double x2 = 3.2, y2 = 9.7;
string s1("hello"), s2("goodbye");

cout << minimum(x1, y1) << endl;

cout << minimum<double>(x2, y2) << endl;

cout << minimum(s1, s2) << endl;


swapIt(x1, y1);

swapIt(x2, y2);

swapIt(s1, s2);

cout << "After swapIt, x1 = " << x1
     << ", y1 = " << y1 << endl;
cout << "After swapIt, x2 = " << x2
     << ", y2 = " << y2 << endl;
cout << "After swapIt, s1 = " << s1
     << ", s2 = " << s2 << endl;
```

Templated Classes

```
template <typename Object>
class ObjectWrapper {

  public:

    ObjectWrapper(const Object & initValue = Object() ) :
      value(initValue) { }

    const Object & getValue() const {
        return value;
    }

    void setValue( const Object & newValue );

  private:

    Object value;
};


template <typename Object>
void ObjectWrapper<Object>::setValue(
                             const Object & newValue ) {
    value = newValue;
}
```