Admin

- Sections meet this week
  - Section assignments e-mailed to you tomorrow
- Assign 1 out
- Handouts 5 & 7 come in two flavors
  - Take ONE version (Mac or Windows) depending your platform
- Today's topics
  - Libraries, C++ string and stream classes
- Reading
  - Handout 4, Reader Ch. 3 (today & next)

C++ libraries

- Groups related operations
  - Header file provides function prototypes and usage comments
  - Compiled library contains implementation
- C++ standard libraries
  - e.g. string, iostream, fstream
  - `#include <iostream>`
  - Terse, lowercase names: `cout getline substr`
- CS106 libraries
  - e.g. simpio, random, graphics
  - `#include "random.h"`
  - Capitalized verbose names: `GetInteger RandomChance DrawLine`

CS106 random.h

- Library of functions to provide randomness
  - Support for shuffling, dice-rolling, coin-flipping, etc.
  - Free functions
- `void Randomize()`
  - Call once at start to initialize new random sequence
- `int RandomInteger(int low, int high)`
  - Returns int chosen from at random from range low-high inclusive
- `double RandomReal(double low, double high)`
  - Same, but for real values
- `bool RandomChance(double probability)`
  - Returns true with odds of probability, false otherwise
- Coherent, convenient, complete

C++ string

- Models a sequence of characters
- string defines a class, strings are objects
  - many operations are member functions that operate on receiver string
- Simple operations
  - member function .length returns number of chars
  - square brackets to access individual chars
  - C++ strings are mutable! (unlike Java)

```
int main()
{
  string s;

  s = "cs106";
  for (int i = 0; i < s.length(); i++)
     s[i] = toupper(s[i]);
```

Operators on strings

- Assign using =, makes new copy
- Compare with relational ops (<, ==, >=, ...)
  - lexicographic ordering
- + is overloaded to do concatenation
  - operands must be chars or strings only

```
int main()
{
  string s, t = "hello";

  s = t;
  t[0] = 'j';
  s = s + ' ';
  if (s != t)
      t += t;
```

string member functions

- Invoke member functions using dot notation

  `str.function(args)`
- Sample member functions:

  `int length()`

  `int find(char ch, int pos = 0)`

  `int find(string pattern, int pos = 0)`
  - returns index of first occurrence or string::npos if not found

  `string substr(int pos, int len)`
  - returns new string, copies len characters starting from pos

  `void insert(int pos, string txt)`
  - changes receiver, inserts txt at pos

  `void replace(int pos, int len, string txt)`
  - changes receiver, removes len chars start at pos, replace with txt

  `void erase(int pos, int len)`
  - changes receiver, removes len chars starting at pos

CS106 strutils.h

- Few convenience free functions for string
- Converting between case
  - `string ConvertToLowerCase(string s)`
    `string ConvertToUpperCase(string s)`
- Converting numbers to string and back
  - `int StringToInteger(string s)`
    `string IntegerToString(int num)`

    `double StringToReal(string s)`
    `string RealToString(double num)`

C++ string vs C-string

- C++ inherits legacy of old-style C-string
  - (pointer to character array, null-terminated)
  - String literals are actually C-strings
- Converting C-string to C++ string
  - Happens automatically in most contexts
  - Can force using **`string("abc")`**
- Converting C++ string to C-string
  - Using member function **`a.c_str()`**
- Why do you care?
  - Some older functionality requires use of C-string
  - C-string not quite compatible with concatenation

Concatenation pitfall

- If one operand is true C++ string, all good

```
string str = "abc";
str + "def";
str + 'd';
str + ch;
```

- If both operands are C-string/char, bad times

```
"abc" + "def";     // won't compile
"abc" + 'd';      // compiles, but doesn't work
"abc" + ch;       //  same
```

- Can force conversion if needed

```
string("abc") + ch;
```

C++ console I/O

- Stream objects cout/cin
  - cout is the console output stream, cin for console input
  - << is stream insertion, >> is stream extraction

```
#include <iostream>

int main()
{
     int x,y;
     cout << "Enter two numbers: ";
     cin >> x  >> y;
     cout << "You said: " << x << " and " << y << endl;
```

- Safer, easier read from console using our simpio.h

```
#include "simpio.h"

int main()
{
   int x = GetInteger();
   string answer = GetLine();
```