CS 106X, Lecture 1

Welcome to CS 106X!

reading:

Course Information handout

Programming Abstractions in C++, Ch. 1-2

Plan For Today

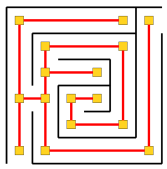
- Introduction
- Course Policies
- Getting Started with C++

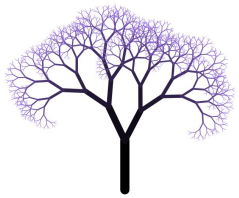
Plan For Today

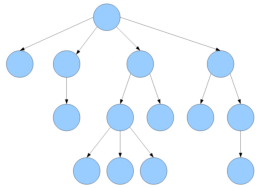
- Introduction
- Course Policies
- Getting Started with C++

What is CS 106X?

- **Programming Abstractions (Accelerated)**
  - **data**; complex data structures
  - Uses the **C++** programming language
  - **algorithm** analysis and algorithmic techniques such as recursion
  - programming **style** and software development practices
  - Accelerated pace vs. CS 106B







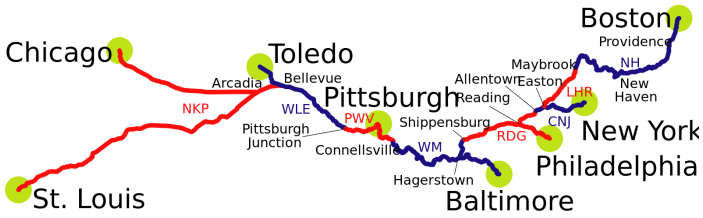
Course Overview

- Mastering **ADTs** (Collections)
- Understanding **recursion** and **recursive backtracking**
- Managing **memory** with **pointers**
- Implementing collections using **data structures** like **linked lists** and **trees**
- Learning about **graphs** and **graph algorithms**
- Analyzing algorithmic efficiency

You'll be able to...

- Use data structures to generate your own "Shakespearean" play
- Draw beautiful recursive geometric shapes
- Solve recursive problems like partitioning crisis resources or tending to patients
- Write your own compression program, like .zip files
- Use graphs to implement algorithms like those used in navigation apps such as Google Maps

... beyond ourselves in our opinions As it is common for the younger sort To lack discretion. Come, go we to the King. They have letters for him. Ere we were two days old at sea, a pirate of very warlike ...



CS 106A, B and X

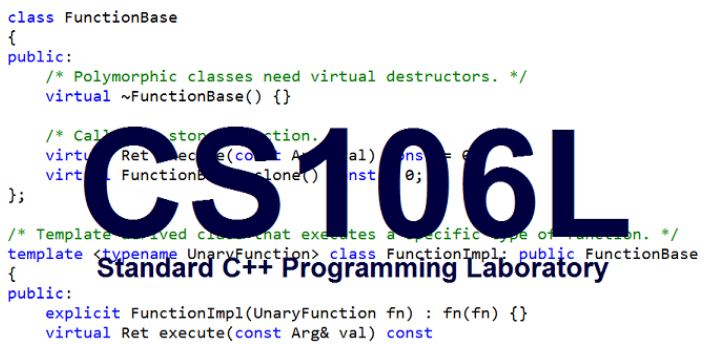
- **CS 106A**: Programming Methodology (Java, Python, JavaScript)
  Prereq: none
  - first course in programming, software development, coding style
  - text and graphics; basic data and algorithms; problem solving

- **CS 106B**: Programming Abstractions (C++) *Prereq: 106A*
  - **data**; complex data structures
  - **algorithm** analysis and algorithmic techniques such as recursion
  - programming **style** and software development practices

- **CS 106X**: Programming Abstractions (Accelerated) (C++)
  - similar content to CS 106B, but faster and more challenging
  - expects significant coding experience, ability to learn quickly

Companion Classes

- **CS 106L (Aut, Spr):** a one-unit course to learn and practice C++ programming in depth



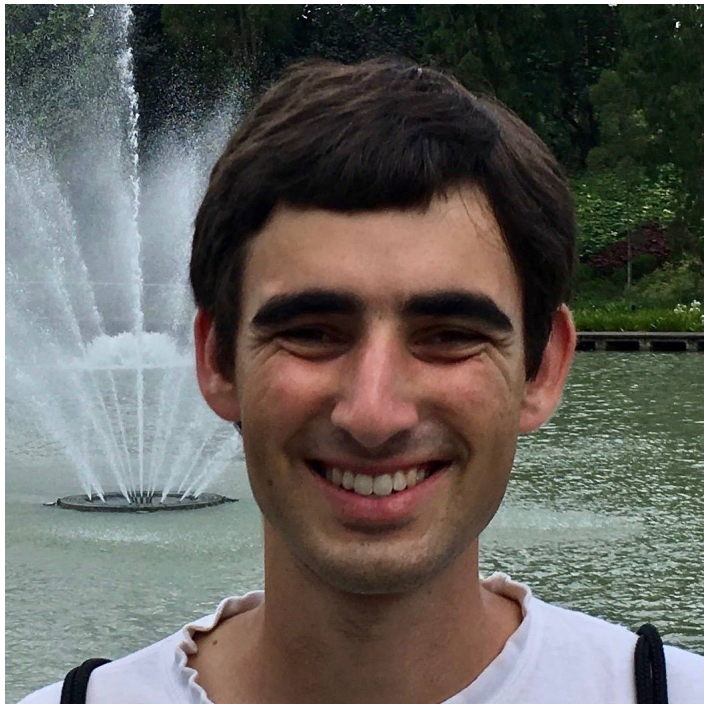
- **CS 106S (Aut, Win, Spr):** a one-unit course to explore the intersection of CS and social good

Course Website

cs106x.stanford.edu

Nice to meet you!

- Lecturer: Nick Troccoli
- OH: MWF 1:30-2:30PM
- troccoli@stanford.edu



- Head TA: Zachary Birnholz
- OH: Tuesday 2:30-4:30PM
- zacharyb@stanford.edu

Section Leaders

- Helpful undergraduate assistants who will:
  - run your discussion section each week
  - grade your homework assignments and exams
  - help you when you have questions
  - ... and much more

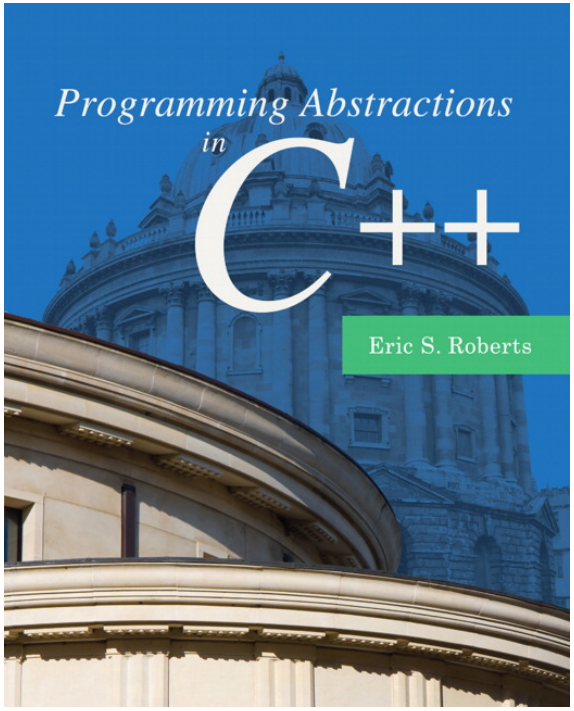
Plan For Today

- Introduction
- Course Policies
- Getting Started with C++

Textbook

- *Programming Abstractions in C++*, by Eric Roberts
  - written here at Stanford; tailored to this course
  - no problems directly assigned from it
  - usable on (closed-note) exams
  - on reserve at library

  - *suggested:* either buy a copy,
    or have access to one when you need it



Grading

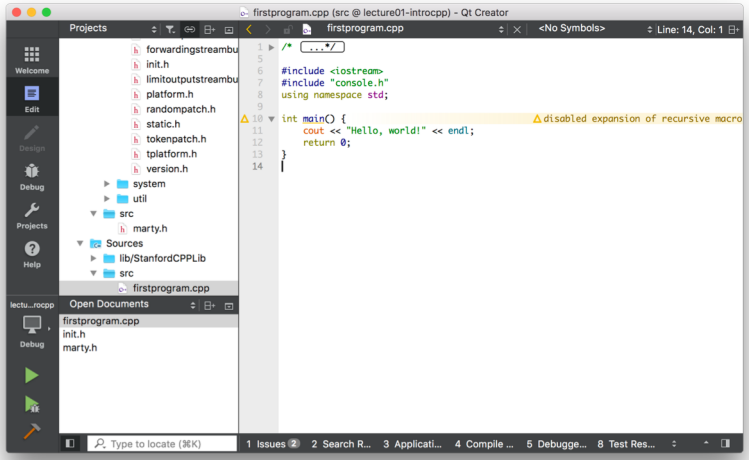
******	45%	Programming assignments
*	10%	Section Participation
**	20%	Midterm Exam
****	25%	Final Exam

Grading

******	45%	Programming assignments
*	10%	Section Participation
**	20%	Midterm Exam
****	25%	Final Exam

Programming Assignments

- ~7 programming assignments (some individual, some in **pairs**), completed using **Qt Creator**
  - Free software, available on course website
  - Troubleshooting session **Wed. 9/26 7-9PM in LaIR**
  - Please follow our installation instructions (special X ones)
  - We will give out starter projects for each assignment



- graded on **functionality** (behavior) and **style** (elegance)
  - Interactive grading sessions for every assignment
  - grading scale is divided into "buckets"

The Bucket System

√	satisfactory; meets requirements, maybe a few issues

The Bucket System

√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues

The Bucket System

√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements

The Bucket System

+	Exceeds expectations; often reflects additional work
√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements

The Bucket System

+	Exceeds expectations; often reflects additional work
√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements
-	Extremely serious problems, a little effort and understanding

The Bucket System

++	Absolutely fantastic submission (*very rare)*
+	Exceeds expectations; often reflects additional work
√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements
-	Extremely serious problems, a little effort and understanding

The Bucket System

++	Absolutely fantastic submission (*very rare)*
+	Exceeds expectations; often reflects additional work
√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements
-	Extremely serious problems, a little effort and understanding
--	Little effort

The Bucket System

++	Absolutely fantastic submission (*very rare)*
+	Exceeds expectations; often reflects additional work
√+	Well done; satisfies all assignment requirements
√	satisfactory; meets requirements, maybe a few issues
√-	Problems serious enough to fall short of assignment requirements
-	Extremely serious problems, a little effort and understanding
--	Little effort
0	No submission

Getting Help

- Visit the SLs in the **LaIR/CLaIR** (1st floor of Tresidder Union)
  - open Sun-Thursday, 7PM – 11PM, starting this Sunday 9/30
  - staffed with multiple section leaders to answer questions

- Other help resources:
  - Instructor/head TA office hours
  - Piazza discussion forum (for conceptual or logistics questions)
  - Email (only for private logistics or grading questions)

- See the "Course Communications" handout on the course website for the best ways to get help.

2 Minds are Better Than 1

- Some assignments may optionally be done in **pairs**
- Both partners receive the same grade
- A chance to brainstorm ideas and work with another programmer
- **MUST be in the same section!**
- More info in handout #1 and on the course website

Interactive Grading

- For each assignment (except for the last), you will get feedback via an **Interactive Grading** (IG) session, scheduled with your section leader.
- Go over assignment feedback, strengths, things to improve

Late Days

- **Start out with 3 “free late days”**: each late day allows you to submit an assignment one lecture day late without penalty.
- Hard deadline 3 lecture days after original due date
- 1-bucket deduction per day late after late days are exhausted
- Pair late days are assessed individually
- ”Pre-granted extensions” – additional extensions granted only in *very special* circumstances. **Head TA** must approve extensions.

Grading

******	45%	Programming assignments
*	10%	Section Participation
**	20%	Midterm Exam
****	25%	Final Exam

Discussion Sections

- Weekly 50-minute sections led by your section leader, starting *next week.*
- Go over lecture material, do practice problems, answer questions
- Graded on section attendance + participation (+IG attendance)
- Submit section preferences between **Thursday 9/27 5PM and Sunday 9/30 5PM**. Signups are *not* first-come-first-serve.

Grading

******	45%	Programming assignments
*	10%	Section Participation
**	20%	Midterm Exam
****	25%	Final Exam

Exams

- **Midterm exam** – Thursday, November 1st, 7-9PM
  - Contact me by *October 25* if you have an academic or University conflict
- **Final exam** – Monday, December 10th, 8:30-11:30AM
  - No alternate final! You ***MUST*** be able to take the final exam at the scheduled time.

- Both exams are *open-book, closed-notes, closed-electronic-device*. You will be provided with a syntax reference sheet.

Grading

******	45%	Programming assignments
*	10%	Section Participation
**	20%	Midterm Exam
****	25%	Final Exam

Stanford Honor Code

- The **Honor Code** is an undertaking of the students, individually and collectively:
  - that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  - that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

- The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

- While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

see also: http://honorcode.stanford.edu/

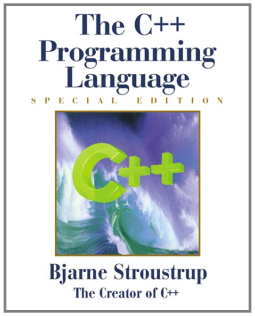
Honor Code and CS 106X

- Please help us ensure academic integrity:
  - Indicate any assistance received on HW (books, friends, etc.).
  - Do not look at other people's solution code *(outside of your pair)*.
  - Do not give your solution code to others, or post it on the web.
  - Report any inappropriate activity you see performed by others.
- Assignments are checked regularly for similarity with help of software tools.
- If you realize that you have made a mistake, you may retract your submission to any assignment at any time, no questions asked.
- If you need help, please contact us and we will help you.
  - We do not want you to feel any pressure to violate the Honor Code in order to succeed in this course.

Plan For Today

- Introduction
- Course Policies
- Getting Started with C++

What is C++ ? (1.2)



- **C++**: A programming language developed in 1983 by Bjarne Stroustrup.
  - one of the world's most widely used languages today
  - built for systems programming with high speed/efficiency
  - built on older C language by adding object-oriented programming
  - continues to be improved over time (latest version: C++17)

- C++ syntax has many similarities with Java and C
  - similar data types (`int`, `double`, `char`, `void`)
  - similar operators (+, -, *, /, %), keywords
  - use of { } braces for scope
  - comes equipped with a large standard library for you to use

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Program comments

Inline comments can be written as:

```
// comment
```

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Import statements

C++ libraries are written with angle brackets

Local (and Stanford) libraries have quotes:

```
#include "lib.h"
```

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Namespaces

Functions and variables are divided (scoped) by namespace

Normally would refer to them as **`namespace::symbol`**

The "using" keyword removes the need for the namespace (brings those symbols into the global program scope)

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Main function – entry point for the program
Should always return an integer (0 = success)
Functions do not need to be part of a class in c++

First C++ program (1.1)


```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

cout – prints output to the screen

Familiar syntax (1.5-1.8)

```
int x = 42 + 7 * -5;                    // variables, types
double pi = 3.14159;
char c = 'Q';                           /* two comment styles */
bool b = true;

for (int i = 0; i < 10; i++) {          // for loops
    if (i % 2 == 0) {                   // if statements
        x += i;
    }
}

while (x > 0 && c == 'Q' || b) {        // while loops, logic
    x = x / 2;
    if (x == 42) { return 0; }
}

fooBar(x, 17, c);                       // function call
barBaz("this is a string");             // string usage
```

Console output: cout

- `cout << expression << expression ...`

```
cout << "You are " << age << " years old!";
```

- `endl`
  - A variable that means "end of line"
  - Same as "\n", but more compatible with all operating systems

```
cout << "You are " << age << " years old!" << endl;
```

Getting Console Input

– Use the Stanford Library simpio: #include "simpio.h"

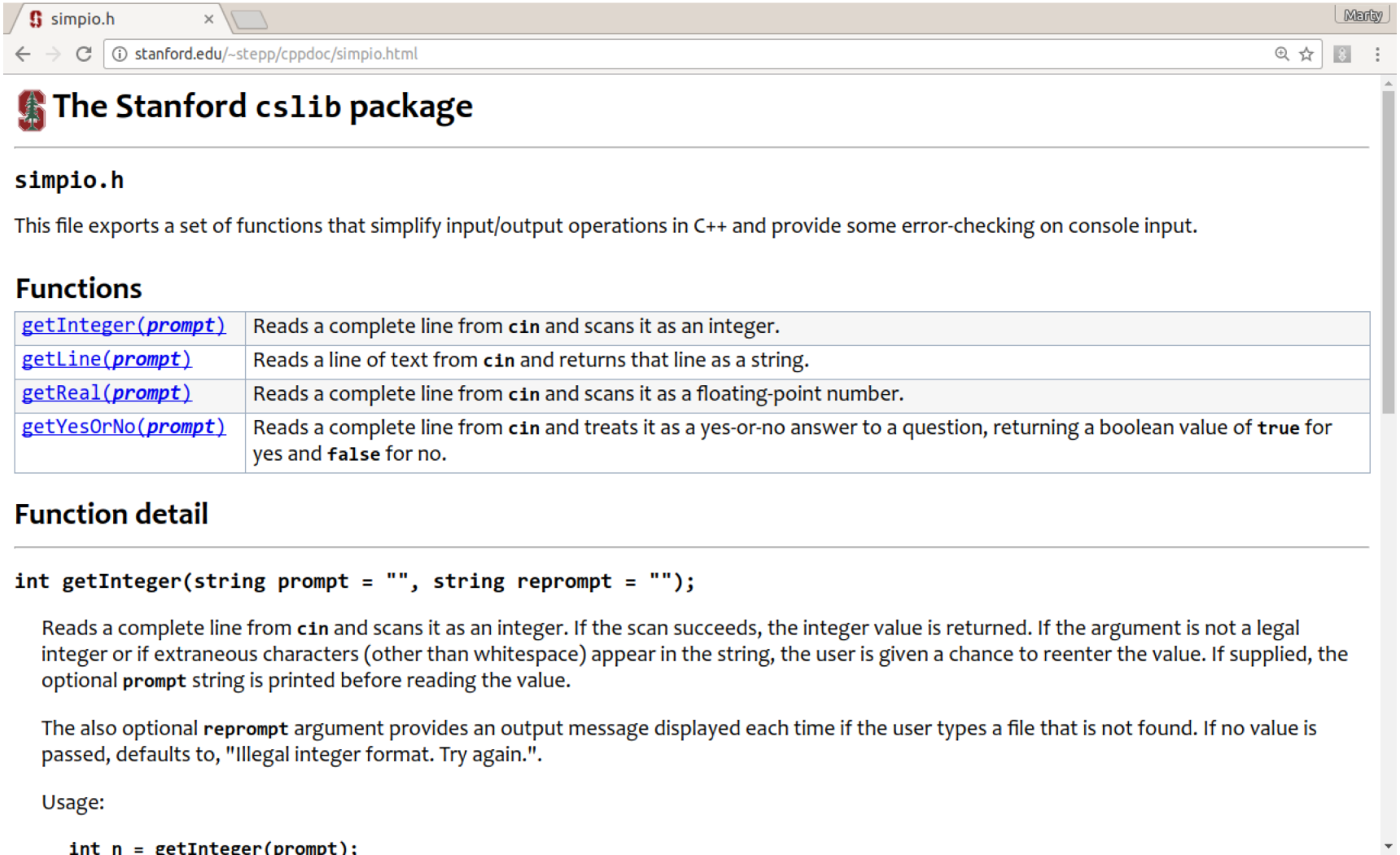
Function name	Description
`getInteger("prompt")`	repeatedly prompts until an integer is typed; returns it
`getReal("prompt")`	repeatedly prompts until double is typed; returns it
`getLine("prompt")`	prompts and reads/returns an entire line of text
`getYesOrNo("prompt")`	repeatedly prompts for a Yes/No answer; return it as a bool

```
string fullName = getLine("Student name? ");
int age = getInteger("How old are you? ");
double gpa = getReal("What's your GPA so far? ");
if (getYesOrNo("Destroy the universe?")) { ... }
```

– NOTE: cin is discouraged
  - Doesn't handle errors well or work with Stanford libraries
  - Difficult to get full lines of input

Stanford library

http://stanford.edu/~stepp/cppdoc/



The Stanford `cslib` package

`simpio.h`

This file exports a set of functions that simplify input/output operations in C++ and provide some error-checking on console input.

Functions

`getInteger(prompt)`	Reads a complete line from **`cin`** and scans it as an integer.
`getLine(prompt)`	Reads a line of text from **`cin`** and returns that line as a string.
`getReal(prompt)`	Reads a complete line from **`cin`** and scans it as a floating-point number.
`getYesOrNo(prompt)`	Reads a complete line from **`cin`** and treats it as a yes-or-no answer to a question, returning a boolean value of **`true`** for yes and **`false`** for no.

Function detail

```
int getInteger(string prompt = "", string reprompt = "");
```

Reads a complete line from **`cin`** and scans it as an integer. If the scan succeeds, the integer value is returned. If the argument is not a legal integer or if extraneous characters (other than whitespace) appear in the string, the user is given a chance to reenter the value. If supplied, the optional **`prompt`** string is printed before reading the value.

The also optional **`reprompt`** argument provides an output message displayed each time if the user types a file that is not found. If no value is passed, defaults to, "Illegal integer format. Try again.".

Usage:

```
int n = getInteger(prompt);
```

Exercise: Stanford vs Cal

- Write a program to compute who won the Stanford-Berkeley game.
  - Assume that the user enters valid integers.
  - Example output:

```
Stanford points scored? 87
Cal points scored? 3
Stanford won!
```

Stanford vs Cal Solution

```
/* This program prints a score of a football game. */
#include <iostream>
#include "console.h"
#include "simpio.h"
using namespace std;

int main() {
    int stanford = getInteger("Stanford points scored? ");
    int cal = getInteger("Cal points scored? ");
    if (stanford > cal) {
        cout << "Stanford won!" << endl;
    } else if (cal > stanford) {
        cout << "Cal won!" << endl;
    } else {
        cout << "A tie." << endl;
    }
    return 0;
}
```

Wrap-up

- Introduction √
- Course Policies √
- Getting Started with C++ √

Next time: diving deeper into C++

Overflow Slides

C++ programs/files (1.3)

- C++ source code lives in .cpp files
  - Additional declarations can be put in "header" .h files

- Source code is compiled into binary *object* files (.o)
- unlike a Java .class, C++ executables are *platform-dependent*