CS 106B, Lecture 1
Introduction to C++

reading:
Programming Abstractions in C++, Chapters 1 & 2
Plan for Today

• Course Overview and Expectations
  – Course Staff introduction
  – Course Policies

• Introduction to C++
  – Syntax
  – Import statements
  – Console input/output
  – Our first programs

• Assignment 0 - NameHash
Course Staff

• Instructor: Ashley Taylor
• OH: 11:20-12:15PM M-Th
• ataylor4@stanford.edu

• Head TA: Shreya Shankar
• OH: 1:30-3:30PM Tuesday
• shreya@cs.stanford.edu
Section Leaders (SLs)

- Lead **required** 50-minute sections (5% of your grade)
  - If you need to miss a week, just attend a different section. Full list at cs198.stanford.edu
- Grade homework
- Hold office hours (**LaIR**) from 7-11PM, Sunday-Wednesday, in the first floor of Tresidder
- Sign up for section before **5PM on Tuesday** at cs198.stanford.edu
Whom to Contact?

- **Non-coding** homework or course logistics questions
  - Piazza ([https://piazza.com/class/jgo9q8mwgn26s8](https://piazza.com/class/jgo9q8mwgn26s8))

- Coding homework questions or Qt issues
  - LaIR or head TA/instructor OH

- Conceptual Questions (no code)
  - CLaIR (same time and place as LaIR) or head TA/instructor OH

- Homework grading questions
  - Email your SL

- Alternate exam scheduling, assignment regrade requests, switching section to work with a partner, extension requests
  - Email the head TA

- Honor Code Questions or Course Feedback
  - Email the instructor or attend instructor OH
Course Tools

• Course website (important announcements, handouts, etc.)
  – http://web.stanford.edu/class/cs106b

• Course Forum: Piazza
  – https://piazza.com/class/jgo9q8mwgn26s8

• LaIR and CLaIR: 7-11PM, Sunday through Wednesday, in Tresidder

• Lecture Videos: canvas.stanford.edu or mvideox.stanford.edu

• Textbook: Eric Roberts’ Programming Abstractions in C++
  – http://web.stanford.edu/class/cs106b/textbooks.html

• Homework Turn-in: Paperless
  – cs198.stanford.edu/paperless

• Our IDE: Qt Creator

• Course Forum: Piazza
  – https://piazza.com/class/jgo9q8mwgn26s8
Homework

• 6.5 near-weekly homeworks (the schedule is on our website: http://web.stanford.edu/class/cs106b/schedule/)
• Cumulatively 35% of your grade
• Graded on functionality and style
• Use a “bucket-system”: most grades are a check-plus or a check
• Pair Programming: pairs must be in the same section, work together on an assignment

• Late assignments
  – Everyone gets **three** free 24-hour late days for the quarter
  – May turn in an assignment no more than 48-hours late; the last assignment will not be accepted late
  – After late days are used, each additional 24-hour period is one bucket deduction

• **Hint:** always read (and re-read) the homework spec
Exams

• Midterm
  – Wednesday, July 25, 7-9PM in Hewlett 200

• Final
  – Saturday, August 18, 8:30-11:30AM. Location TBA
  – The final will be cumulative with more emphasis on material covered in the second half of the quarter

• All exams are closed-book, closed-note though you may bring one 8.5x11” double-sided sheet of notes with you

• Please fill out the exam form (on the course website) before Friday (part of Homework 0)

• Students with accommodations should send their accommodations letter to Shreya and me
Honor Code and CS 106

http://honorcode.stanford.edu/

• Please help us ensure academic integrity:
  – 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.
  – Indicate any assistance received on HW (books, web sites, friends).
  – Report any inappropriate activity you see performed by others.

• Assignments are checked 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.

• See Honor Code handout on course web site
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
Phylogenetic Trees

Fractal Graphics

Source: https://upload.wikimedia.org/wikipedia/commons/0/0e/Terragen_render.jpg
Google Maps

Source: https://www.google.com/maps
• Assignment 0 released today, due Friday
  – Google form with course logistic information
  – Fill out the exam conflict survey by **5PM on Friday**
  – Try to install Qt Creator tonight, and stop by our **Qt Creator Installation help session** tomorrow from 1:30-3:30PM in Gates B02

• Sign up for section at cs198.stanford.edu
  – Section signups close **today at 5PM**
  – Make sure you sign up for the same sections as your partner (if you have one)
  – You'll be assigned a section before Wednesday's lecture
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
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

![Diagram showing the compilation and linking process of C++ programs/files]

file1.cpp → object file

file2.cpp → object file

object file → executable

library

library

compile

link
/*
 * 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"
/*
 * 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)
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */

#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
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
User Input and Output

reading:

*Programming Abstractions in C++, Chapter 2, 4*
• `cout << expression << expression ...`

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

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

```cpp
cout << "You are " << age << " years old!" << endl;
```
Getting Console Input

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

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getInteger(&quot;prompt&quot;)</code></td>
<td>repeatedly prompts until an integer is typed; returns it</td>
</tr>
<tr>
<td><code>getReal(&quot;prompt&quot;)</code></td>
<td>repeatedly prompts until double is typed; returns it</td>
</tr>
<tr>
<td><code>.getLine(&quot;prompt&quot;)</code></td>
<td>prompts and reads/returns an entire line of text</td>
</tr>
<tr>
<td><code>getYesOrNo(&quot;prompt&quot;)</code></td>
<td>repeatedly prompts for a Yes/No answer; return it as a bool (optionally pass in values to treat as &quot;yes&quot; and &quot;no&quot;)</td>
</tr>
<tr>
<td><code>getYesOrNo(&quot;prompt&quot;, &quot;y&quot;, &quot;n&quot;)</code></td>
<td>repeatedly prompts for a Yes/No answer; return it as a bool (optionally pass in values to treat as &quot;yes&quot; and &quot;no&quot;)</td>
</tr>
</tbody>
</table>

```cpp
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
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getInteger(prompt)</code></td>
<td>Reads a complete line from <code>cin</code> and scans it as an integer.</td>
</tr>
<tr>
<td><code>getline(prompt)</code></td>
<td>Reads a line of text from <code>cin</code> and returns that line as a string.</td>
</tr>
<tr>
<td><code>getReal(prompt)</code></td>
<td>Reads a complete line from <code>cin</code> and scans it as a floating-point number.</td>
</tr>
<tr>
<td><code>getYesOrNo(prompt)</code></td>
<td>Reads a complete line from <code>cin</code> and treats it as a yes-or-no answer to a question, returning a boolean value of <code>true</code> for yes and <code>false</code> for no.</td>
</tr>
</tbody>
</table>

Function detail

```cpp
int getInteger(string prompt = "", string reprompt = "") {
    // Implementation details...
}
```

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:

```cpp
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!
/* This program prints a score of a football game. */
#include <iostream>
#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;
}
Overflow (extra) slides
Which is not an error?

```cpp
#include <iostream> // A.
using namespace std;

public static int main() { // B.
    int age = getInteger("How old are you?"); // C.

    cout << "You are " + age + ", wow!" + endl; // D.
}
```