

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

Plan for Today

- Course Overview and Expectations
 - Course Staff introduction
 - Course Policies
- Introduction to C++
 - Syntax
 - Import statements
 - Console input/output
 - Our first programs

Course Staff



- Instructor: Tyler Conklin
- OH: M-Th 12:30-1:30PM
- tconklin@stanford.edu



- Head TA: Kate Rydberg
- OH: Tues 8-10PM
- rydbergk@stanford.edu

Section Leaders (SLs)

- Lead **required** 50-minute sections (5% of your grade)
 - If you need to miss a week, 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](#)
- Coding homework questions or Qt issues
 - LaIR or head TA/instructor OH
- Conceptual Questions (no code)
 - Piazza, 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**
- LaIR and CLaIR: 7-11PM, Sunday through Wednesday, in Tresidder
- **Textbook**: Eric Roberts' *Programming Abstractions in C++*
 - <http://web.stanford.edu/class/cs106b/textbooks.html>
- Homework Turn-in/Review: **Paperless**
 - cs198.stanford.edu/paperless
- Our IDE: **Qt Creator**
 - http://web.stanford.edu/dept/cs_edu/qt-creator/qt-creator.shtml

Homework

- 7 weekly homeworks (the schedule is on our website: <http://web.stanford.edu/class/cs106b/schedule/>)
- Cumulatively 45% 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
 - 7/24 7-9PM
- Final
 - 8/16 12:15-3:15PM
- 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 Kate and me

Honor Code and CS 106

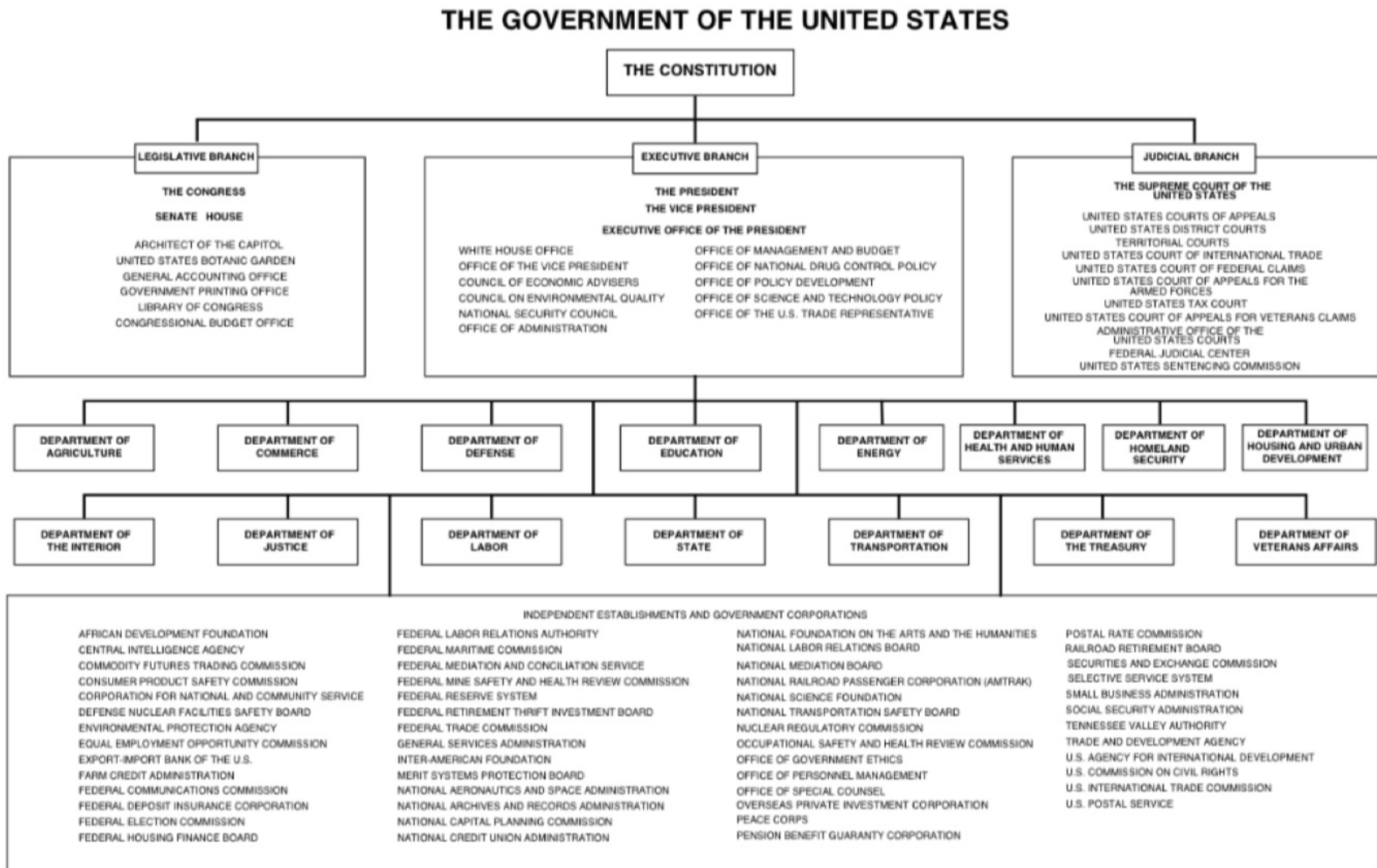
<http://honorcode.stanford.edu/>

- 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 Using **ADTs** (Collections)
- Understanding **recursion** and **recursive backtracking**
- Managing **memory** with **pointers**
- Implementing collections using **data structures** like **linked lists** and **trees**
- Learning about **graphs**, **hashing**, and **sorting**.
- Analyzing algorithmic efficiency

Modeling Hierarchies



Google Maps

University of California, Berkeley



Stanford University

Add destination

Leave now

OPTIONS

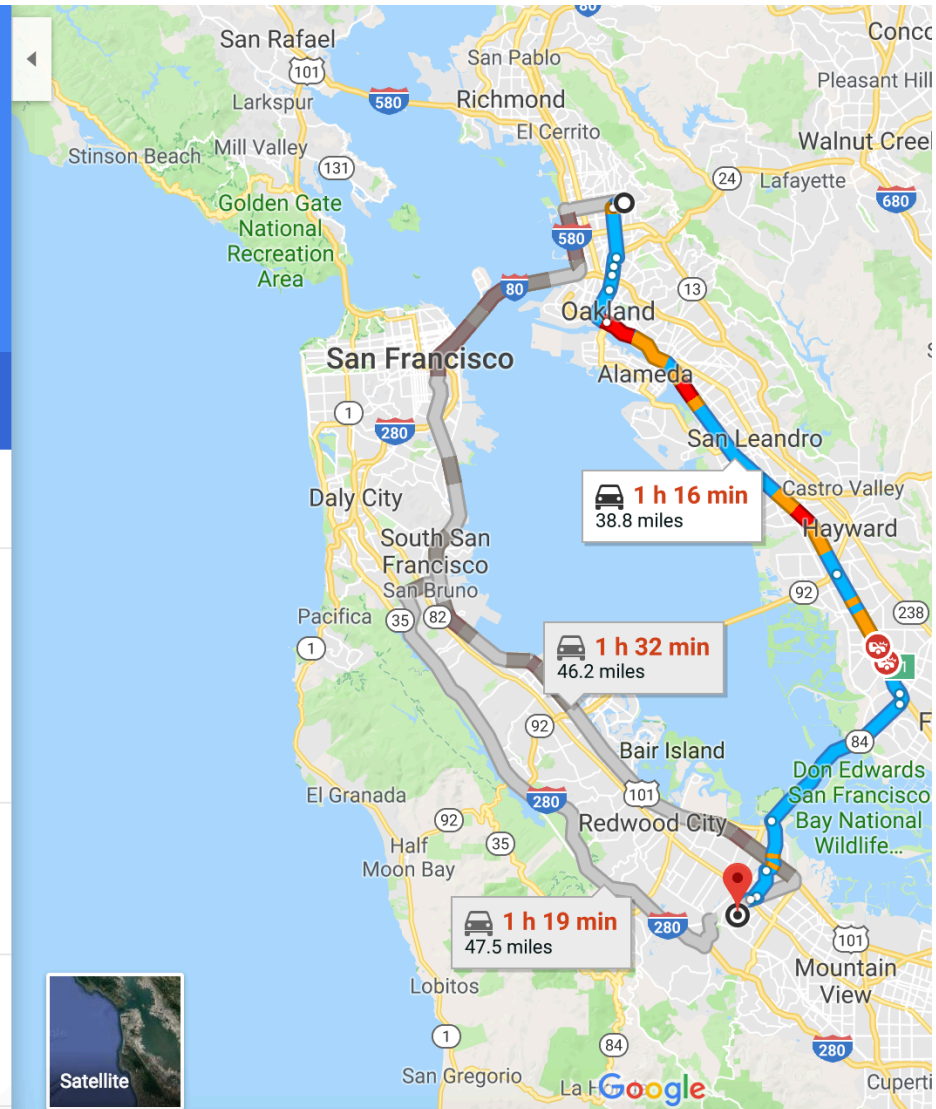
 [Send directions to your phone](#)

 **via I-880 S** **1 h 16 min**
Fastest route now, avoids slowdown on the Bay Bridge
38.8 miles
 This route has tolls.

[DETAILS](#)

 **via I-280 S** **1 h 19 min**
Heavy traffic, as usual
47.5 miles

 **via US-101 S** **1 h 32 min**
Heavy traffic, as usual
46.2 miles



What I Expect From You

- Start Early!
- Ask for help!

Plan for Today

- Course Overview and Expectations
 - Course Staff introduction
 - Course Policies
- Introduction to C++
 - Syntax
 - Import statements
 - Console input/output
 - Our first programs

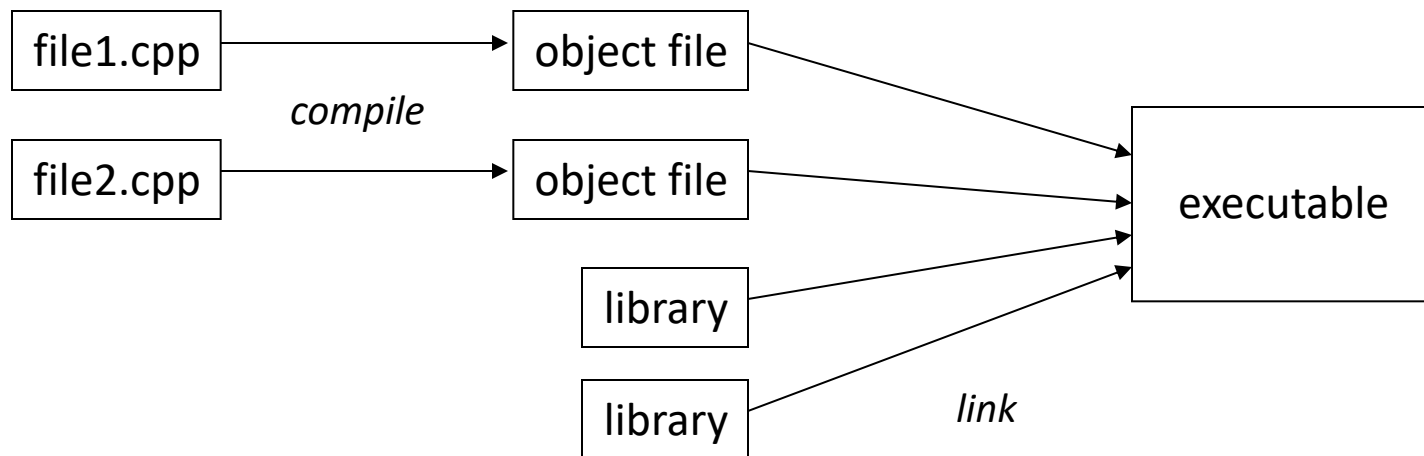
What is C++ ?

- **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 (`+`, `-`, `*`, `/`, `%`) and keywords
 - use of `{ }` braces for scope
 - comes equipped with a large standard library for you to use



C++ programs/files

- 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 file, C++ executables are *platform-dependent*



First C++ program

```
/*  
 * 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

```
/*  
 * 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

```
/*  
 * 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

```
/*  
 * 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

```
/*  
 * 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++

Familiar syntax

```
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

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
<code>getInteger("prompt")</code>	<i>repeatedly</i> prompts until an integer is typed; returns it
<code>getReal("prompt")</code>	<i>repeatedly</i> prompts until double is typed; returns it
<code>getLine("prompt")</code>	prompts and reads/returns an entire line of text
<code>getYesOrNo("prompt")</code>	<i>repeatedly</i> 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 <code>c1n</code> and scans it as an integer.
getLine(prompt)	Reads a line of text from <code>c1n</code> and returns that line as a string.
getReal(prompt)	Reads a complete line from <code>c1n</code> and scans it as a floating-point number.
getYesOrNo(prompt)	Reads a complete line from <code>c1n</code> 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 `c1n` 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 "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;
}
```

Look Ahead

- Sign up for a section! => <https://cs198.stanford.edu/cs198/>
 - Section signups close **Tuesday at 5PM**
 - Make sure you indicate the same preferences as your partner
- Assn. 0 is out (due **Thursday at 5PM**). Find it on the course website.
 - Try to install Qt Creator tonight!! If things don't go so well, stop by our **Qt Creator Installation help session** Wednesday from 8-10PM in the LaIR
- Visit the website to familiarize yourself with it => <http://web.stanford.edu/class/cs106b/>
- Add to the class playlist [here](#)! (Not mandatory).