Welcome to CS 106L!

... Stick around!
Today

- Introductions
- Course Logistics
- The Pitch
- C++ Basics
Frankie

Into:
- Outside
- My Toyota Sienna
- Crosswords
- Programming
- Language Theory
- Magic
- Brandi (Oct 7 @ frost)
Frankie

Not Into:
- Inside
- People who drop 106L
Sathya

Into:
- EE + Physics
- Computational Physics
- Violin
- Climbing
- LoL/Valo
Sathya

Not Into:
- Blisters
- Leetcode
Today

- Introductions
- Course Logistics
- The Pitch
- C++ Basics
Lecture

- Held Tuesdays and Thursdays 3:15-4:45pm in 380-380C (here and now)
- We will usually try to keep lectures closer to an hour, just wanted to give ourselves time to get into the cool stuff!
- No lecture week 10!
- You can email us to request a screen recording of a lecture but we want you to come in person!
Office Hours

- Sathya: Thursday 12:30-1:30 in person, 5-6 online!
  - **both online this week!
- Frankie: Wednesdays, 3:10ish-4:10ish in person, 6-7pm online!
- We want to talk to you! Come talk!
- Extra office hours when assignments are due!
- Stay tuned for more (will Frankie’s be outside?)
- Watch the website (cs106l.stanford.edu) for more info
Assignments

- There will be 3 *very short* assignments
- You only need to do 2 to pass the class
- Pairs are allowed! (Not at all necessary)
- 3 late days, more if you fill out feedback forms!
- Email us to work out any extensions
- Check out the assignment setup page ASAP!
Questions?
Today

- Introductions
- Course Logistics
- The Pitch
- C++ Basics
Why CS106L?
CS106B
- Focus is on **concepts** like abstractions, recursion, pointers etc.
- Bare minimum C++ in order to use these concepts

CS106L
- Focus is on **code**: what makes it good, what **powerful** and **elegant** code looks like
- The real deal: No Stanford libraries, only STL
- Understand **how and why** C++ was made
Why C++?
C++ is still a very popular language

<table>
<thead>
<tr>
<th>May 2021</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Chart Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>13.38%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Python</td>
<td>11.87%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Java</td>
<td>11.74%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C++</td>
<td>7.81%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>C#</td>
<td>4.41%</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Visual Basic</td>
<td>4.02%</td>
<td>16</td>
</tr>
</tbody>
</table>

Tiobe Index, 2021
Classes that use C++

BIOE 215: Physics-Based Simulation of Biological Structure
CME 253: Introduction to CUDA (deep learning)
CS 144: Introduction to Computer Networking
CS 231N: Convolutional Neural Networks for Visual Recognition
GENE 222: Parallel Computing for Healthcare
ME 328: Medical Robotics
MUSIC 256A: Music, Computing, Design I
MUSIC 420A: Signal Processing Models in Musical Acoustics
Companies that use C++

amazon.com

intel

IBM

facebook

Google

Apple

Microsoft

Adobe
Browsers written in C++
Software written in C++
Games written in C++

ASSASSIN'S CREED

CALL OF DUTY

MASS EFFECT

STAR CRAFT

WORLD OF WARCRAFT

HALO
Other cool stuff written in C++

The F-35 Lightning II (Joint Strike Fighter) relies extensively on C++

The Spirit rover was operational for over 6 years when the mission was only planned to run for around 3 months
Why C++?

- Fast

Lower-level control

- C
- Assembly
- Machine Code
- C++
- Java
- Scala
- Perl
- Javascript
- Ruby
What is C++?
```cpp
#include <iostream>

int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```
Also some C++ Code

```cpp
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
    printf("%s", "Hello, world!\n");
    // ^a C function!
    return EXIT_SUCCESS;
}
```
Also (technically) some C++ code

```cpp
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
    asm(
        "sub    $0x20,%rsp\n\t"
        "movabs $0x77202c6f6c6c6548,%rax\n\t"
        "mov    %rax,(%rsp)\n\t"
        "movl   $0x646c726f, 0x8(%rsp)\n\t"
        "movw   $0x21, 0xc(%rsp)\n\t"
        "movb   $0x0,0xd(%rsp)\n\t"
        "leaq    (%rsp),%rax\n\t"
        "mov    %rax,%rdi\n\t"
        "call  __Z6myputsPc\n\t"
        "add    $0x20, %rsp\n\t"
    );
    return EXIT_SUCCESS;
}
```
C++ History: Assembly

section .text

global _start

_start:

    ;tell linker entry point

    mov edx, len
    ;message length

    mov ecx, msg
    ;message to write

    mov ebx, 1
    ;file descriptor (stdout)

    mov eax, 4
    ;system call number (sys_write)

    int 0x80
    ;call kernel

    mov eax, 1
    ;system call number (sys_exit)

    int 0x80
    ;call kernel

section .data

msg db 'Hello, world!', 0xa
;our dear string

len equ $ - msg
;length of our dear string
C++ History: Assembly

Benefits:
- Unbelievably simple instructions
- Extremely fast (when well-written)
- Complete control over your program

Why don’t we always use Assembly?
Assembly looks like this

section .text

global _start ;must be declared for linker (ld)

_start: ;tell linker entry point

    mov edx, len ;message length
    mov ecx, msg ;message to write
    mov ebx, 1 ;file descriptor (stdout)
    mov eax, 4 ;system call number (sys_write)
    int  0x80 ;call kernel
    mov eax, 1 ;system call number (sys_exit)
    int  0x80 ;call kernel

section .data

msg    db 'Hello, world!', 0xa ;our dear string
len     equ $ - msg ;length of our dear string
C++ History: Assembly

Drawbacks:

- A lot of code to do simple tasks
- Very hard to understand
- Extremely unportable (hard to make work across all systems)
Next in C++ History: Invention of C

**Problem:** computers can only understand assembly!

- **Idea:**
  - Source code can be written in a more intuitive language
  - An additional program can convert it into assembly
    - This additional program is called a **compiler**!
  - Take **CS143** to learn more!
C++ History: Invention of C

- T&R created C in 1972, to much praise
- C made it easy to write code that was
  - Fast
  - Simple
  - Cross-platform
- Learn to love it in CS107!

Ken Thompson and Dennis Ritchie, creators of the C language.
C++ History: Invention of C

- C was popular because it was simple.
- This was also its weakness:
  - No objects or classes
  - Difficult to write generic code
  - Tedious when writing large programs
C++ History: Welcome to C++!

- In 1983, the beginnings of C++ were created by Bjarne Stroustrup.
- He wanted a language that was:
  - Fast
  - Simple to use
  - Cross-platform
  - Had high-level features

The man himself 😊
C++ History: Evolution of C++

- **C++**: 1979
- **C++ with Classes**: 1983
- **C++98**: 1998
- **C++03**: 2003
- **C++11**: 2011
- **C++14**: 2014
- **C++17**: 2017
- **C++20**: 2020
- **C++23**: ?

We are here.
But...What is C++?
Today

- Introductions
- Course Logistics
- The Pitch
- C++ Basics
C++: Basic Syntax + the STL

Basic syntax
- Semicolons at EOL
- Primitive types (ints, doubles etc)
- Basic grammar rules

The STL
- Tons of general functionality
- Built in classes like maps, sets, vectors
- Accessed through the namespace std::
Standard C++: Basic Syntax + std library

The STL

- Tons of general functionality
- Built in classes like maps, sets, vectors
- Accesses through the namespace std::
  - Extremely powerful and well-maintained
CS106B

- Stanford libraries abstract away messy details of C++
- C++98*
- “Use this function we made for you called getInteger”
- “““style”””

*plus range-based for-loops

CS106L

- All the messy details
- C++17 (sneak peak at 20)
- Learn how cin is used to make getInteger
- Learn how to abstract away messy details for others

NOT: memorize c++ syntax
Design Philosophy of C++
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- **Compartmentalization is key**
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Design Philosophy of C++

- Only add features if they solve an actual problem
- Programmers should be free to choose their own style
- Compartmentalization is key
- Allow the programmer full control if they want it
- Don’t sacrifice performance except as a last resort
- Enforce safety at compile time whenever possible
Our first C++ program: HelloWorld.cpp