Welcome to CS 106L!

We’re so glad you’re here!

Haven Whitney and Fabio Ibanez

Spring 2024
CONTENTS

01. Introductions
02. Course Logistics
03. The ✨Pitch✨
04. C++ Basics
01. Introductions

02. Course Logistics

03. The ✨ Pitch ✨

04. C++ Basics
Welcome and Logistics

http://web.stanford.edu/class/cs106l/

I’m Haven!

I’m Fabio!
Now you all can meet (some of) each other!

First: Introduce yourself to the person on your right

Second: Introduce yourself to the person on your left

Potential Conversation Topics:

- What’s something you’re into and not into?
- Why do you want to take this class?
CONTENTS

01. Introductions

02. Course Logistics

03. The ✨ Pitch ✨

04. C++ Basics
Asking Questions

We welcome questions!

- Feel free to raise your hand at any time with a question.
- We’ll also pause periodically to solicit questions and check understanding.
Asking Questions

We welcome questions!

- Feel free to raise your hand at any time with a question.
- We’ll also pause periodically to solicit questions and check understanding.
Access and Accommodations

- Disabled students are a valued and essential part of the Stanford community. We welcome you to our class.
- Please work with OAE but also let us know if there's anything we can do to make the course more accessible for you.
- Don't be shy asking for accommodations if problems arise. We're very reasonable people and will do whatever we can to help.
Community Norms

- Shame-free zone
- Treat your peers and instructors with kindness and respect
- Be curious
- Communication is key!
- Recognize we are all in-process (humility, question posing, avoid perfectionism)
Guiding Principles

We will do everything we can to support you. We want to provide flexibility to the best of our ability!

- We want to hear your feedback so we can ensure the class is going as smoothly as possible for everyone
- Please communicate with us if any personal circumstances or issues arise! We are here to support you.
Questions?
Lecture

- Held **Tuesdays** and **Thursdays** 4:30pm-5:50pm in Thornton 110
- No lecture after week 9!
- Lecture is not recorded.
- **Attendance is required**. Short participation quizzes (1-2 questions) will be given at the beginning of lecture starting in week 2. All students are given 5 free absences.
Lecture

CS106L is an enrichment course to 106B! As such, we want to cover new and fun material that will be helpful in your C++ journey.

- C++ is a huge language. We want you to get practice with some things, exposure to others, and a lot is not covered.
Illness

If you feel ill or are sick, for the wellbeing of yourself and others please stay home, take care of yourself, and reach out to us - we never want you to feel that you must attend class if you are not feeling well!

Similarly, if you have an emergency or exceptional circumstance, please reach out to us so that we can help!
Office Hours

● OH time TBD and will be in person.
  ○ These will be settled by week 3 (before first assignment)

● We want to talk to you! Come talk!

● Extra office hours weeks 9-10!

● Watch the website (cs106l.stanford.edu) and Ed for more info.
All class information can be found at:

cs106l.stanford.edu
Assignments

There will be 7 short weekly assignments (typically takes 1 hour at most depending on experience).

- Submissions will be on Paperless or as directed on the assignment handout!

Assignments will be released on Thursdays and due in one week (the following Thursday)

- All students have three free late days.
Grading

Grading is S/NC. We expect everyone to get a S!

How to get an S?

- Attend at least 8 of the 13 required lectures between Week 2 and Week 9
- Successful completion of 5 out of 7 weekly assignments
Get in touch with us!

Here are the best ways to communicate with us, in no particular order:

- Email us: cs106l-spr2324-staff@lists.stanford.edu
  - Please use this email not our individual emails so we both receive the message!
- Public or Private Post on Ed
- After class or in our office hours
Questions?
CONTENTS

01. Introductions

02. Course Logistics

03. The ✨ Pitch ✨

04. C++ Basics
## Course Content

<table>
<thead>
<tr>
<th>Week</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Admin, Brief Intro to C++ feature</td>
</tr>
<tr>
<td>2</td>
<td>Initialization + References, Streams</td>
</tr>
<tr>
<td>3</td>
<td>Containers, Iterators, Pointers</td>
</tr>
<tr>
<td>4</td>
<td>Classes, Template Classes, Const</td>
</tr>
<tr>
<td>5</td>
<td>Template Functions, Functions, Lambdas</td>
</tr>
<tr>
<td>6</td>
<td>Operators, Special Member Functions</td>
</tr>
<tr>
<td>7</td>
<td>Move Semantics, Type safety</td>
</tr>
<tr>
<td>8</td>
<td>Bonus Topics + MORE OFFICE HOURS</td>
</tr>
<tr>
<td>9</td>
<td>NO CLASS MORE OFFICE HOURS</td>
</tr>
<tr>
<td>10</td>
<td>NO CLASS MORE OFFICE HOURS</td>
</tr>
</tbody>
</table>
Why CS106L?
Welcome and Logistics

- **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></td>
</tr>
</tbody>
</table>

Tiobe Index, 2021
We use it in classes...

- CS 111: Operating Systems Principles
- 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

... and more!
...and in real life!

amazon.com

Chrome

Google

Among Us
**Why C++?**

**FAST**

- **Lower-level control**
  - Ruby
  - Javascript
  - Python
  - Java
  - C++
  - C
  - Assembly
  - Machine Code

**Graph showing time in minutes for different languages:**

- C
- C++
- C#
- Java
- Perl
- Python

Legend:
- Linux
- Windows
What is C++?
This is some C++ code...

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```
This is also some C++ code! (?)

```c++
#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 C++ code!!

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

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

```c
#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++ is backwards compatible with lower level languages! Neat!
C++ History: Assembly

section .text
global _start

_start:

section .data

msg db 'Hello, world!', 0xa
len equ $ - msg
C++ History: Assembly

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

**Why don’t we always use assembly?**
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)
C++ History: Invention of C

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

**Idea:**
- Source code can be written in a more intuitive language for humans.
- 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

Ken Thompson and Dennis Ritchie 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!**
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**
C++ History: Evolution of C++

- 1979: C++ foundation
- 1983: C++03
- 1998: C++98
- 2003: C++11
- 2011: C++14
- 2014: C++17
- 2017: C++20
- 2020: We are here!
- 2023: C++23

C with Classes
- 1979: C++ foundation
- 1983: C++98
- 1998: C++11
- 2003: C++17
- 2011: C++23
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
Questions?
But... what is C++?
We’ll talk about it Thursday!

Thanks for coming! Next up: Types and Structs!