Welcome to CS 106L!

We're so glad you're here!

Today

01 **Introductions**

02 Course Logistics

03 The ✨Pitch✨

04 C++ Basics

Sarah

Into:

- Eating
- Walking
- Talking (esp while walking)
- Gaming (cards, board, video)
- Teaching
- Cleaning
- Snorkeling

Sarah

Not Into:

- 3D Video Games
- Strawberries
- Cliff Jumping

Sarah

Where I Came From:

- Transferred to Stanford from a community college
- Coded for the first time ~2.66 years ago (almost to the day)
- Never thought CS was for me
- Formerly chemistry major and pre-med

Haven

Into:

- Teaching
- Traveling
- CS + Robotics
- Spanish
- Skydiving
- Sidewalks
- The perfect stern brunch waffle

Haven

Not Into:

- Math 🤢
- Walking Slowly
- When the volume is on an odd number

Haven

Where I came from:

- FLI student from a small town in the South
- Never thought CS could be for someone like me
- Super exciting proving myself wrong and showing others they can do it too!

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 the story behind your name?
  - What’s something you’re into and not into?
  - Why do you want to take this class?

Today

~~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

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
- We're not going to do audience "questions" in lecture that are just showing off that you know some jargon or advanced topic

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)

Our 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.

Lecture

- Held Tuesdays and Thursdays 4:30-5:50pm in Thornton 110
- We will usually try to keep lectures closer to an hour+ish
- No lecture week 10 or week 6!
- Lecture is not recorded
- Attendance is required. Short participation questions will be given at the beginning of lecture starting in week 2. **Given 5 free absences**

Lecture

- CS106L is an enrichment course to 106B
- C++ is a huge language. We want you to get practice with some things, exposure to others, and a lot is not covered.

Lecture

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 use so that we can help

Office Hours

- OH time TBD, will be in person and virtual
- We want to talk to you! Come talk!
- Extra office hours week 6 and 10 when assignments are due!
- Watch the website (cs106l.stanford.edu) and Ed for more info

Where all class information can be found

cs106l.stanford.edu

Assignments

- There will be 2 ***short*** assignments (typically takes 2-4 hrs depending on experience)
- Pairs are allowed!
- **Assignment 1 Due Week 6:** Friday, May 12th @ 11:59pm (Late deadline: Sunday, May 14th @ 11:59pm)
- **Assignment 2 Due Week 10:** Wednesday, June 7th @ 11:59pm **(Firm Deadline)**

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
  - Submit both assignments without build errors

How to Communicate with Us

Notice that it's **spr** NOT **spring**

In no particular order:

- Email us: cs106l-spr2223-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

Course Overview

Week	Topics
1	Admin, Brief Intro to C++ feature
2	Initialization + References, Streams
3	Containers, Iterators, Pointers
4	Classes, Template Classes, Const
5	Template Functions, Functions, Lambdas
6	No class, extra office hours, **Assn 1 Due Friday**
7	Operators, Special Member Functions
8	Move Semantics, Type safety
9	Bonus Topics + MORE OFFICE HOURS
10	NO CLASS MORE OFFICE HOURS, **Assn 2 Due Wednesday**

Learning Outcomes

- Practice using industry standard coding tools such as ssh and VSCode
- Gain familiarity with powerful features of the stl
- Practice reading documentation to learn how to use a new language feature
- Exposure to standard c++ syntax and norms
- Learn a few “advanced” features of classes

Questions?

Today

~~01 Introductions~~

~~02 Course Logistics~~

03 The ✨Pitch✨

04 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

May 2021	Programming Language	Ratings	Chart Ratings
1	C	13.38%	
2	Python	11.87%	
3	Java	11.74%	
4	C++	7.81%	
5	C#	4.41%	
6	Visual Basic	4.02%	

Tiobe Index, 2021

Classes that use C++

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

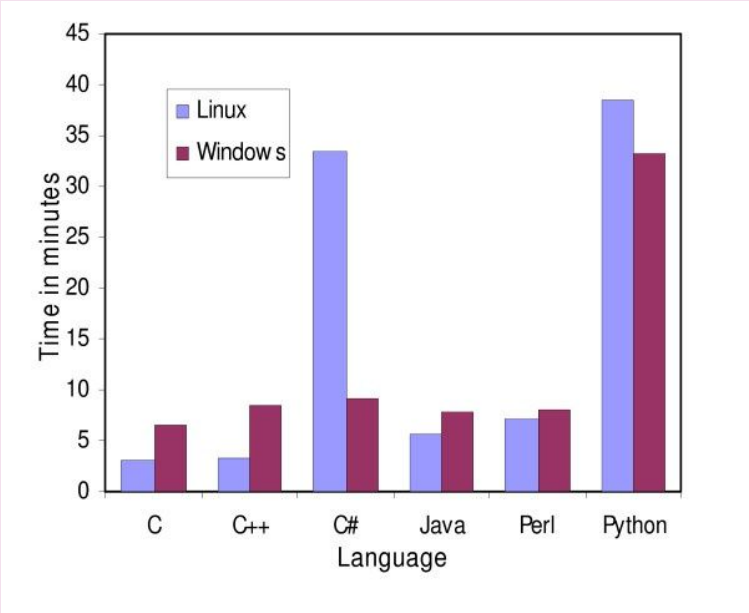
Many Cool Things Use/Were Made with C++

amazon.com®

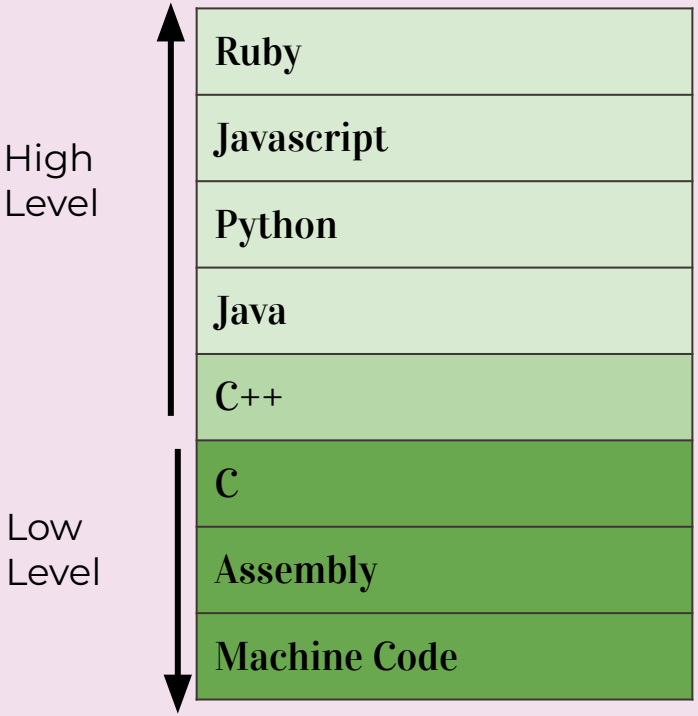
Google

Why C++?

FAST



Lower-level control



What is C++?

Some C++ Code

```
#include <iostream>

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

Also some C++ Code

```
#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++

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

int main(int argc, char *argv) {
   asm(  "sub    $0x20,%rsp\n\t"                    // assembly code!
         "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                  ;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

- 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)

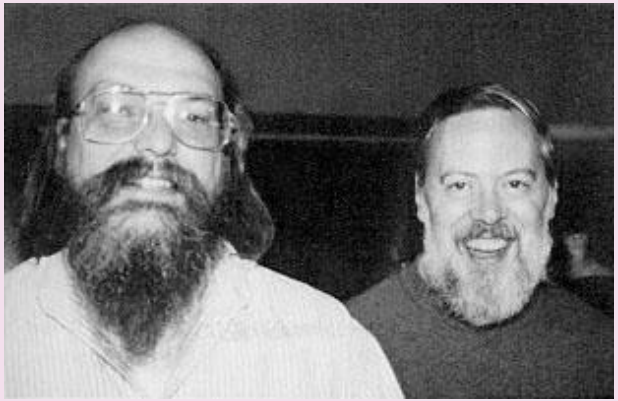
Next in C++ History: Invention of C

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!



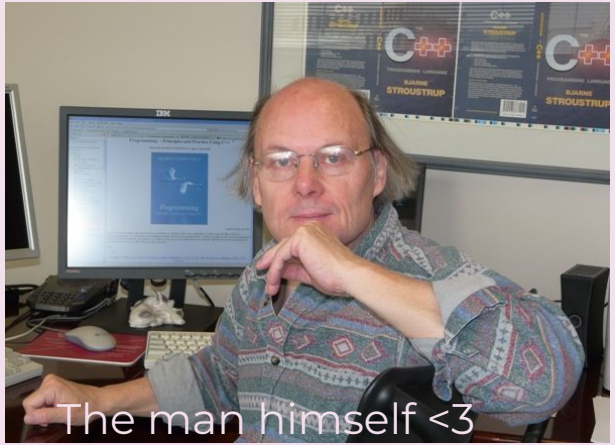
Ken Thompson and Dennis Ritchie, creators of the C language.

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 <3

C++ History: Evolution of C++

We are here

C++ C++03 C++14 C++20

1979 1983 1998 2003 2011 2014 2017 2020 ?

C with Classes C++98 C++11 C++17 C++23

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**

Questions?

But...What is C++?

Today

~~01~~ ~~Introductions~~

~~02~~ ~~Course Logistics~~

~~03~~ ~~The ✨Pitch✨~~

04 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

Basic

- Ser
- Prir
  dou
- Bas

The STL

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

Thank you for coming!
See you Thursday!