# Lecture 1: Welcome to CS106L!

CS106L, Fall 2025

Rachel Fernandez, Thomas Poimenidis

# Today’s Agenda

- Introductions!
- The Pitch
- Course Logistics

# Introductions

**Thomas Poimenidis**

**Rachel Fernandez**

# Now you can meet (some of) each other!

- Turn to the people next to you and introduce yourselves!
- **Potential Conversation Topics:**
  - What's something you're into (or not into)?
  - Why do you want to take this class?

# The Pitch

# Why C++?

# “The invisible foundation of everything”

# Valorant

[source]


CS:GO/CS2
Yellow
Blue
Orange
Purple
[source]

# ...and many more!






# High Frequency Trading


[source]

# Self Driving


[source]

# GPU Programming


[source]

# Arduino


[source]

# And much, much more!

- Databases (MySQL, MongoDB)
- Web Browsers (Chrome, Safari, Edge)
- Virtual Reality (Quest)
- Low level ML (PyTorch, TensorFlow)
- Compilers, virtual machines (JVM, LLVM, GCC)
- Operating Systems (Windows, MacOS, Linux)

# “The invisible foundation of everything”

# C++ is great for...

- Handling lots of data
- And handling it very efficiently
- And doing it in an elegant, readable way

# C++ in Industry



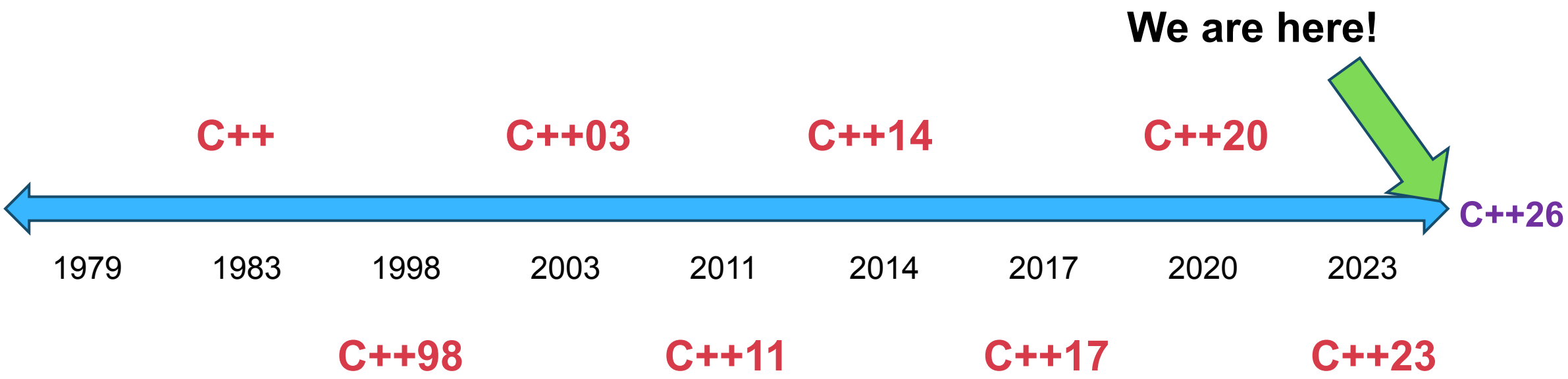
# C++ was first released in 1985, still #2!

| Mar 2025 | Programming Language | Ratings | Change |
|---|---|---|---|
| 1 | Python | 23.85% | +8.22% |
| **2** | **C++** | **11.08%** | **+0.37%** |
| 3 | Java | 10.36% | +1.41% |
| 4 | C | 9.53% | -1.64% |
| 5 | C# | 4.87% | -2.67% |

[TIOBE Index, March 2025]

# The C++ Community

- C++ has a **MASSIVE** user base
- C++ Standard continues to be revised every three years
  - A standards committee decides new changes to the language

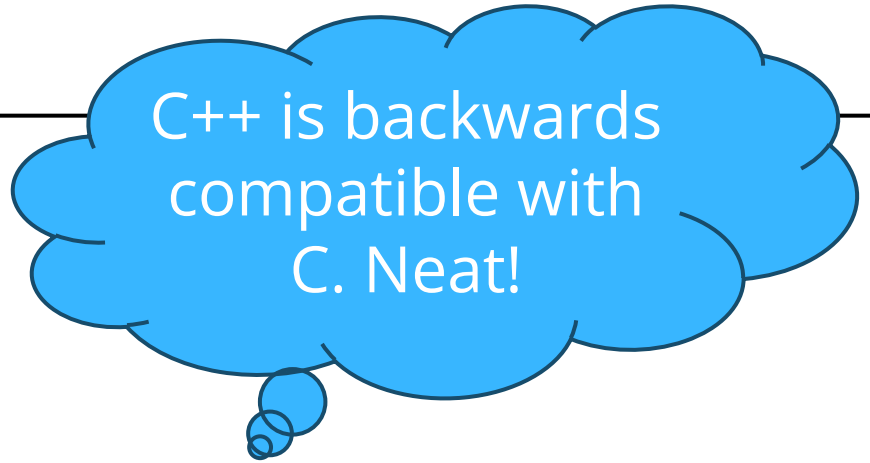


# What is C++?

# A valid C++ program

```
#include <iostream>
#include <string>

int main() {
    auto str = std::make_unique<std::string>("Hello World!");
    std::cout << *str << std::endl;
    return 0;
}

// Prints "Hello World!"
```

# Also a valid C++ program




```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 a valid C++ program

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

int main(int argc, char *argv) {
        asm(".LC0:\n\t"
                        ".string \"Hello, world!\"\n\t"
                "main:\n\t"
                        "push rbp\n\t"
                        "mov rbp, rsp\n\t"
                        "sub rsp, 16\n\t"
                        "mov DWORD PTR [rbp-4], edi\n\t"
                        "mov QWORD PTR [rbp-16], rsi\n\t"
                        "mov edi, OFFSET FLAT:.LC0\n\t"
                        "call puts\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?

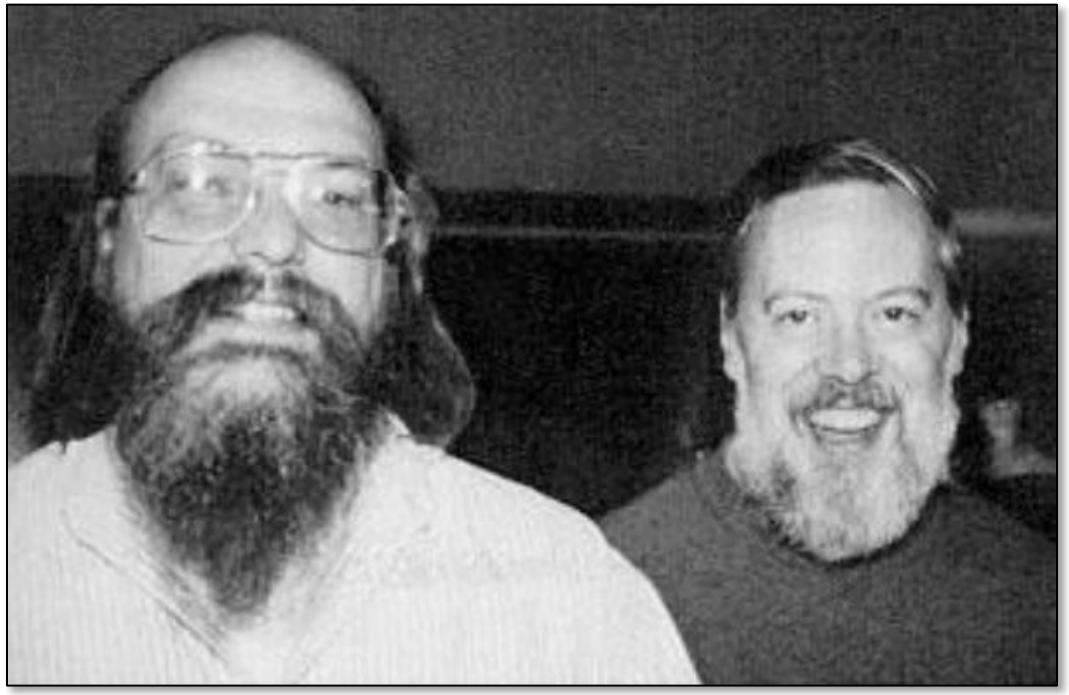
# C++ History: Assembly

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

- ❌ A **lot of code** (even for simple tasks)
- ❌ Very **hard to understand**
- ❌ Extremely **unportable**

# C++ History: Invention of C

- Dennis Ritchie created C in 1972 to much praise.
- C made it easy to write code that was:
  - Fast
  - Simple
  - Cross platform
    - **Compilers!** Source Code → Assembly
- Learn to love it in **CS107**!



Dennis Ritchie (left), alongside Ken Thompson (right)

# C++ History: Invention of C

- C was popular because it was simple
  - *“When I read C I know what the output Assembly is going to look like”*

    —Linus Torvalds, creator of Linux
- However, C has some weaknesses:
  - No **objects** or classes
  - Difficult to write **generic or templated** code
  - **Tedious** to write large programs

# C++ History: Welcome to C++!

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

Bjarne Stroustrup, the man himself ☺

# C++ Design Philosophy

- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
- Do not waste time or space.
- Compartmentalize messy constructs.
- Allow the programmer full control, responsibility, and choice.

"Code should be elegant **and** efficient; I hate to have to choose between those"

—Bjarne Stroustrup

# C++ Design Philosophy (Summarized)

- Readable
- Safety
- Efficiency
- Abstraction
- Programmer Choice

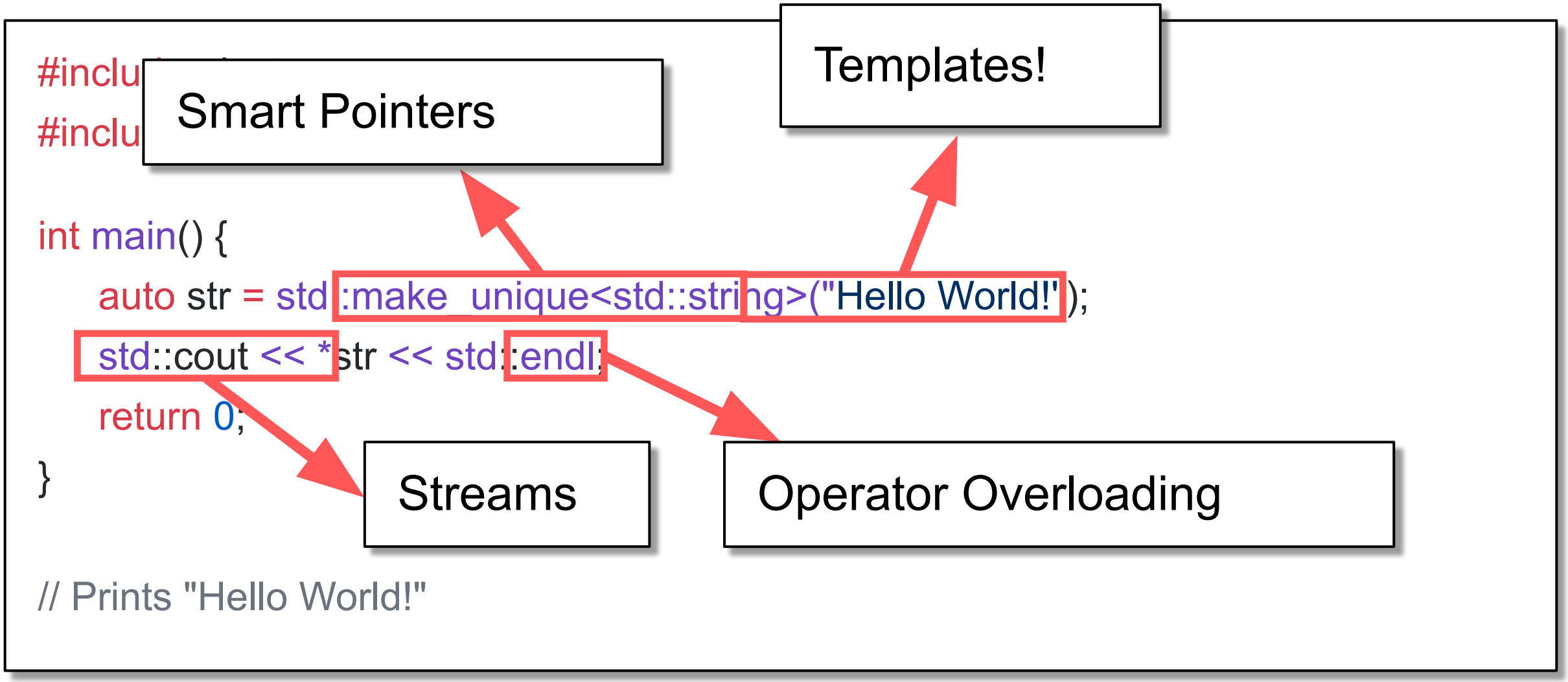
# A valid C++ program (Assembly style)

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

int main(int argc, char *argv) {
        asm(".LC0:\n\t"
                        ".string \"Hello, world!\"\n\t"
                "main:\n\t"
                        "push rbp\n\t"
                        "mov rbp, rsp\n\t"
                        "sub rsp, 16\n\t"
                        "mov DWORD PTR [rbp-4], edi\n\t"
                        "mov QWORD PTR [rbp-16], rsi\n\t"
                        "mov edi, OFFSET FLAT:.LC0\n\t"
                        "call puts\n\t");
        return EXIT_SUCCESS;
}
```

# A valid C++ program (C style)

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

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

# A valid C++ program (C++ style!)



# Why take CS106L?

# Topics We'll Cover

| | |
|---|---|
| Welcome | Types & Structs |
| Initialization & References | Streams |
| Containers | Iterators & Pointers |
| Classes | Advanced Classes |
| Templates | Advanced Templates |
| Functions & Lambdas | Operator Overloading |
| Special Member Functions | Move Semantics |
| std::optional and Type Safety | RAII, Smart Pointers, C++ Projects |

# Other Classes (e.g. CS106B) vs. CS106L

- Focus on **concepts** like abstractions, recursion, pointers etc.
- **Bare minimum** C++ in order to use these concepts
- **Heavy** time commitment

- 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
- Relaxed **1 unit** class!

# You will probably use C++...

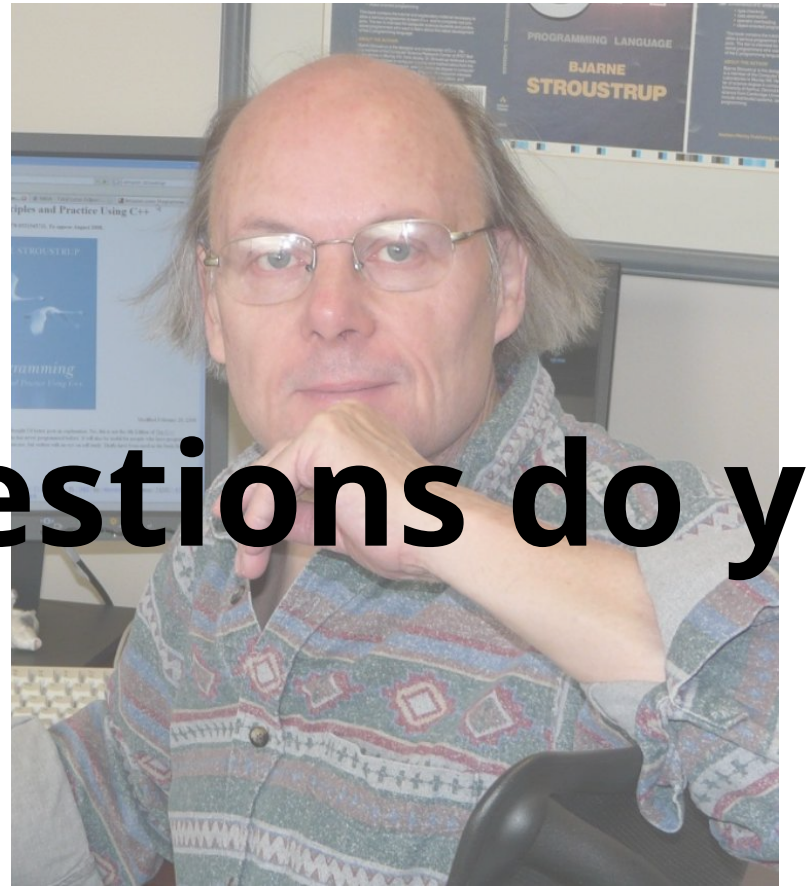
- In at least one of Stanford's classes
  - **CS 111:** Operating Systems Principles
  - **CME 213:** Introduction to parallel computing using MPI, openMP, and CUDA
  - **CS 143:** Compilers
  - **CS 144:** Introduction to Computer Networking
  - **CS 248A:** Computer Graphics: Rendering, Geometry, and Image Manipulation
  - **MUSIC 256A:** Music, Computing, Design: The Art of Design
  - ...and more!
- And in real life!

# C++ helps develop good coding hygiene

- Am I using objects the way they're meant to be used?
  - Type checking, type safety
- Am I using memory efficiently?
  - Reference/copy semantics, move semantics
- Am I modifying something I'm not supposed to?
  - const and const correctness
- Other languages relax these restrictions

“Nobody should call themselves a professional if they only know one language” —Bjarne Stroustrup

# What questions do you have?

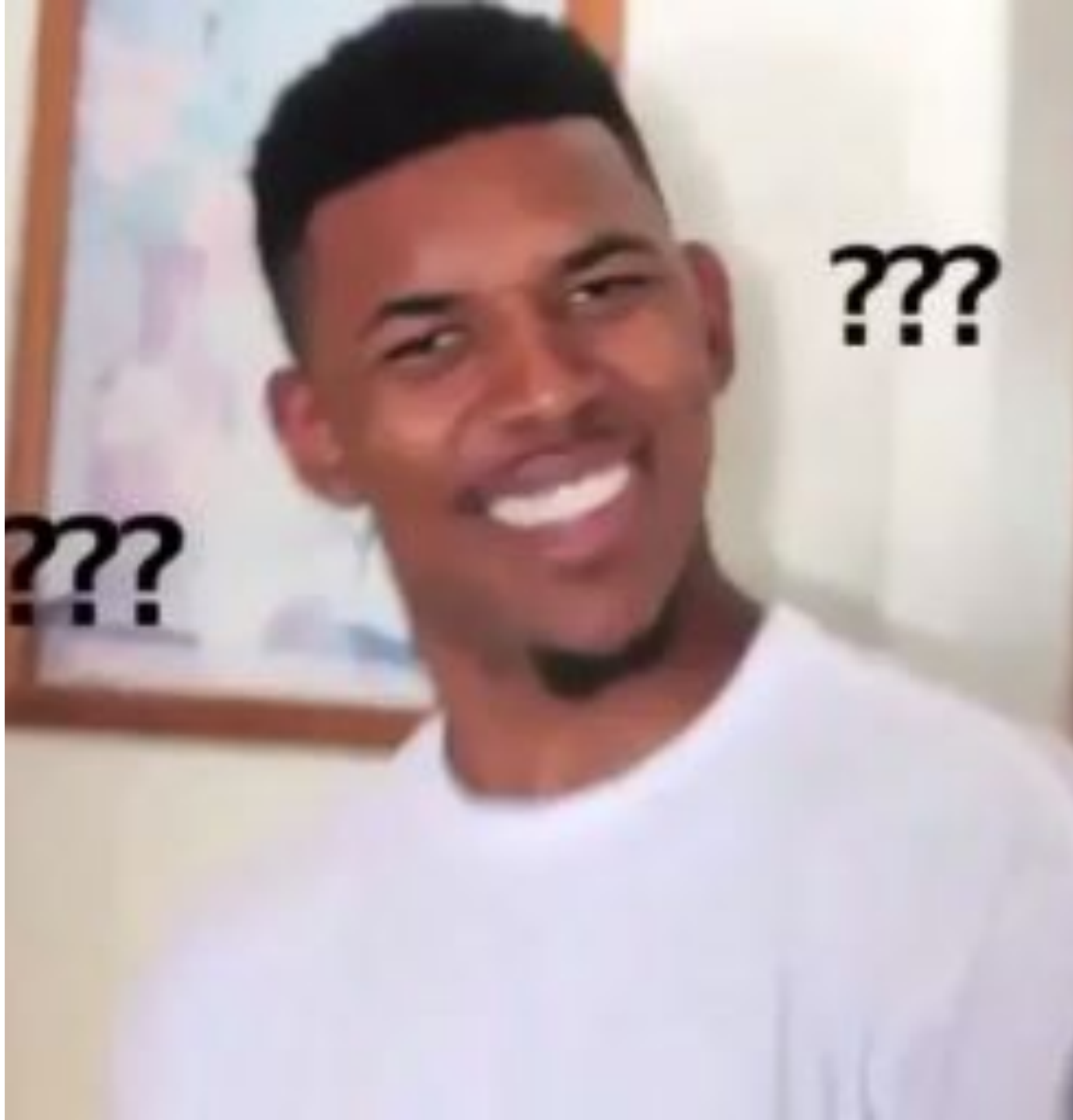



bjarne_about_to_raise_hand

# Course Logistics

# Asking Questions

- We welcome questions!
- Feel free to raise your hand at ***<u>any time</u>*** with a question
- We'll also pause periodically to solicit questions and check understanding



# Access and Accommodations

- Students with disabilities 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 to you.

- Don't be shy about 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!
- Recognized we are all in-process (humility, question posing, avoid perfectionism)

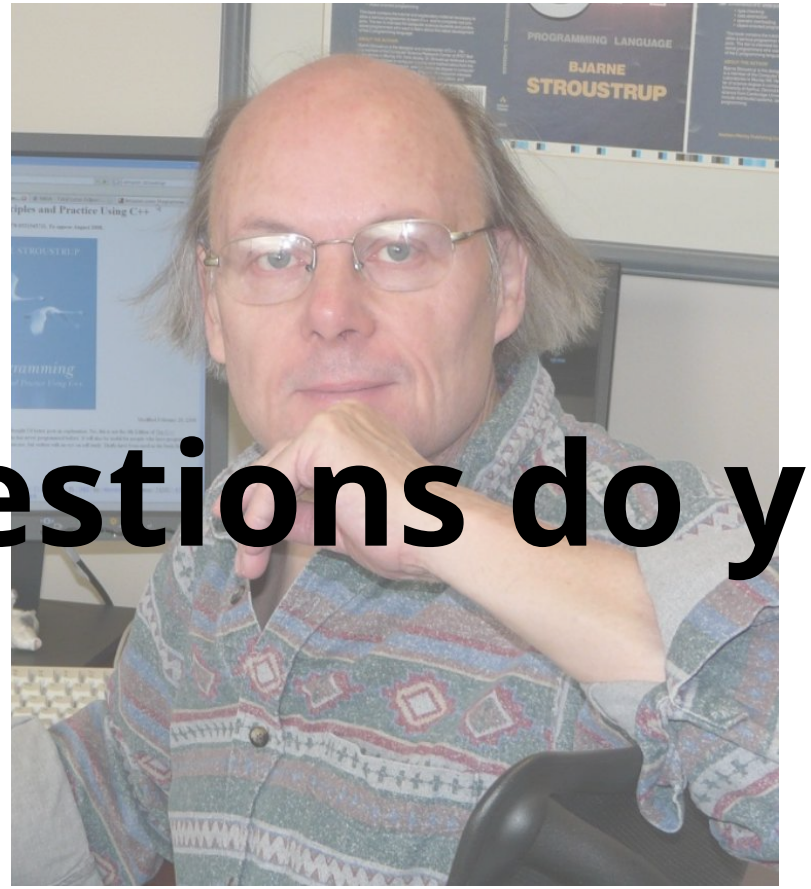
# Guiding Principles

- 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 :)

# What questions do you have?




bjarne_about_to_raise_hand

# Lecture

- Held **Tuesdays and Thursdays** from **3:00PM – 4:20PM** in **Thornton 110**

- Lecture is not recorded.

- **Attendance is required.** Short participations quizzes (1-2 questions) will be given at the beginning of lecture starting in week 2. **All students are given 2 free absences**.

# Illness

- If you 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 times are TBD and will be in person
  - Will be settled by week 2 (before the first assignment)
  - Will be posted on Ed

- We want to talk to you! Come talk!

- Extra OH weeks 9 – 10!

- Watch the course website and Ed for more info.

# CS106L

Standard C++ Programming

Stanford University
Fall 2025



## About CS106L

💸 **CS106L** is a 1-unit class that explores the modern C++ language in depth. We'll cover some of the most exciting features of C++, including modern patterns (up through C++26) that give it beauty and power.

🪖 Anyone who is taking or has taken CS106B/X (or equivalent) is welcome to enroll. In other words, we welcome anyone that has learned or is learning programming fundamentals like functions and objects/classes.

🥕 CS106L is a class for **1 unit**. Students will complete 8 ***very short*** weekly assignments. These are not meant to be too challenging but instead function as some hands-on practice with a few of the concepts we discuss in class the previous week. There are ***no exams or papers***. All grades are S/NC. Class will finish in week 8 to give you time for finals.

🏙️ CS106L is built for you! Even if you're not taking the class, you're welcome to come to our in-person office hours **(starting week 2)**. **Thomas:** TBD, **Rachel:** TBD

## Course Info


- 🐱 Thomas Poimenidis
- 🦜 Rachel Fernandez
- 📪 cs106l-aut2526-staff@lists.stanford.edu
- 🕰️ TTh 3:00 - 4:20pm, Thornton 110


## Quick Links

- 💯 See My Grades
- Dicussion Forum
- Paperless (Submit Code)
- C++ Documentation
- 🐍 Python to C++ Guide
- Previous Quarter's Page

## Schedule

| Week | Tuesday | Thursday |
|---|---|---|
| 1 | September 23<br>1. Welcome! | September 25<br>2. Types & Structs |
| 2 | September 30<br>3. Initialization & References | October 2<br>4. Streams |
| 3 | October 7<br>5. Containers | October 9<br>6. Iterators & Pointers |
| 4 | October 14<br>7. Classes | October 16<br>8. Inheritance |

# Assignments

- There will be 8 short weekly assignments (1-2 hours each)
  - Submissions will be on paperless as directed on the assignment handout!

- Assignments will be released on Fridays and due in one week (the following Friday)
  - All students have three free late days.

# Grading

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

- **How do you get an S?**
  - Attend **12 of the 14** lectures between Week 2 and Week 9
  - Successful completion of **6 out of 8** weekly assignments

# Get in touch with us!

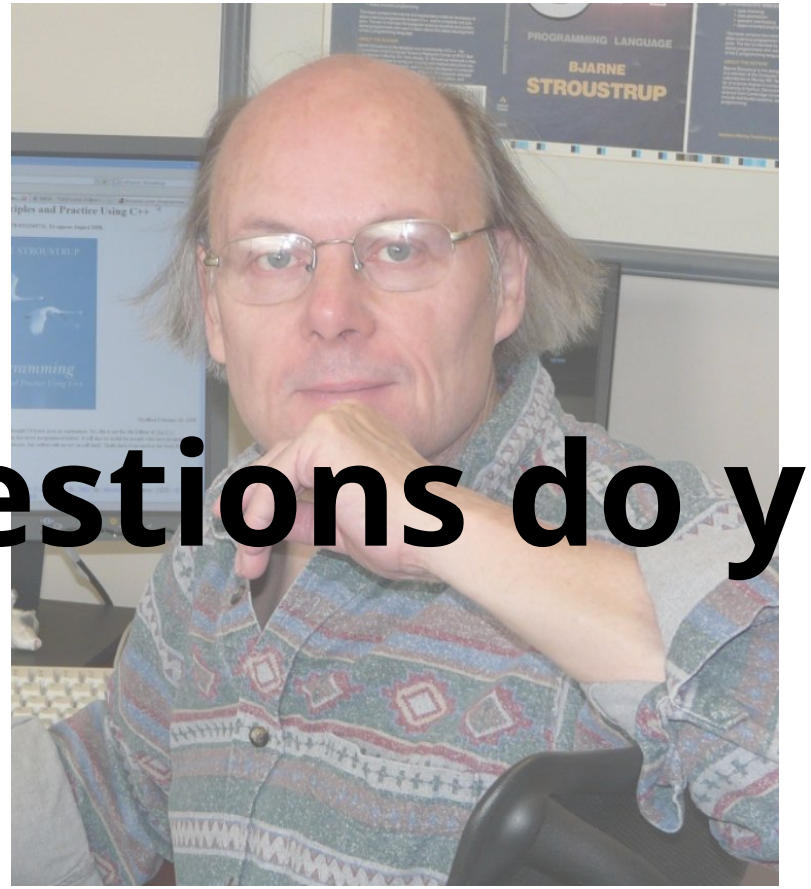
- Here are the best ways to communicate with us!

- Email us: **cs106l-aut2526-staff@lists.stanford.edu**
  - Please use this email and not our individual emails so we both receive the message!

- Public or private post on Ed

- After class or in our office hours

# What questions do you have?




bjarne_about_to_raise_hand