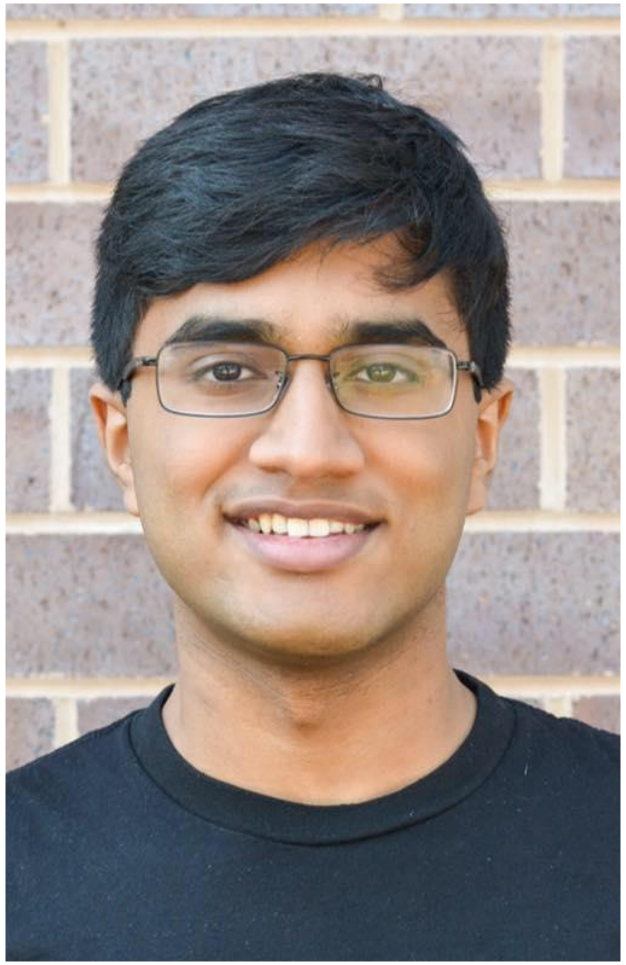
Welcome to CS 106L!

Today’s Agenda

- Introductions
- Logistics
- History and philosophy of C++
- C++ basics
- (Supplemental material) Command-line compilation

Introduction

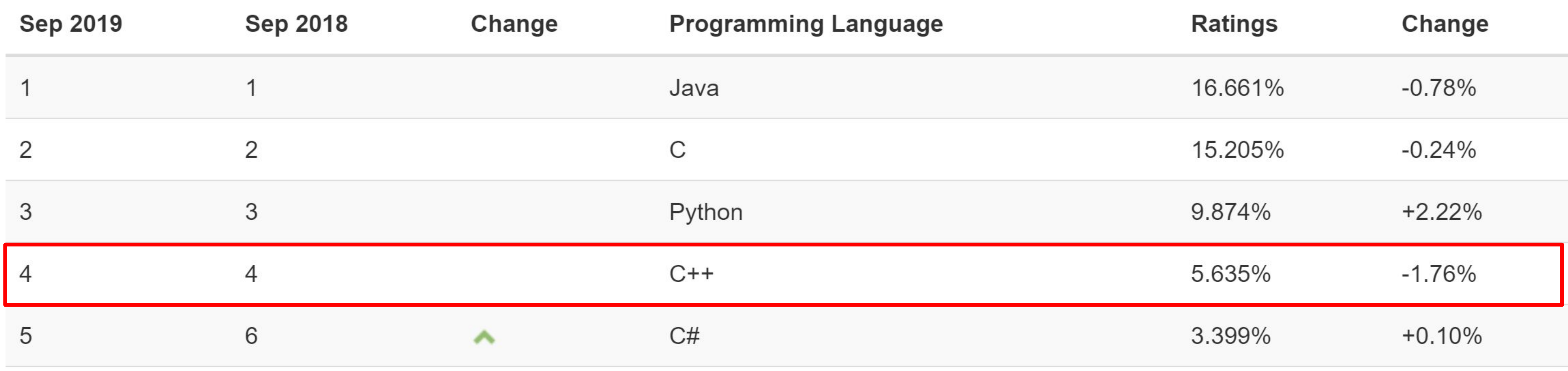


Nikhil Raghuraman
(Tuesdays)

Ethan A. Chi
(Thursdays)

Why C++?

C++ is still a very popular language

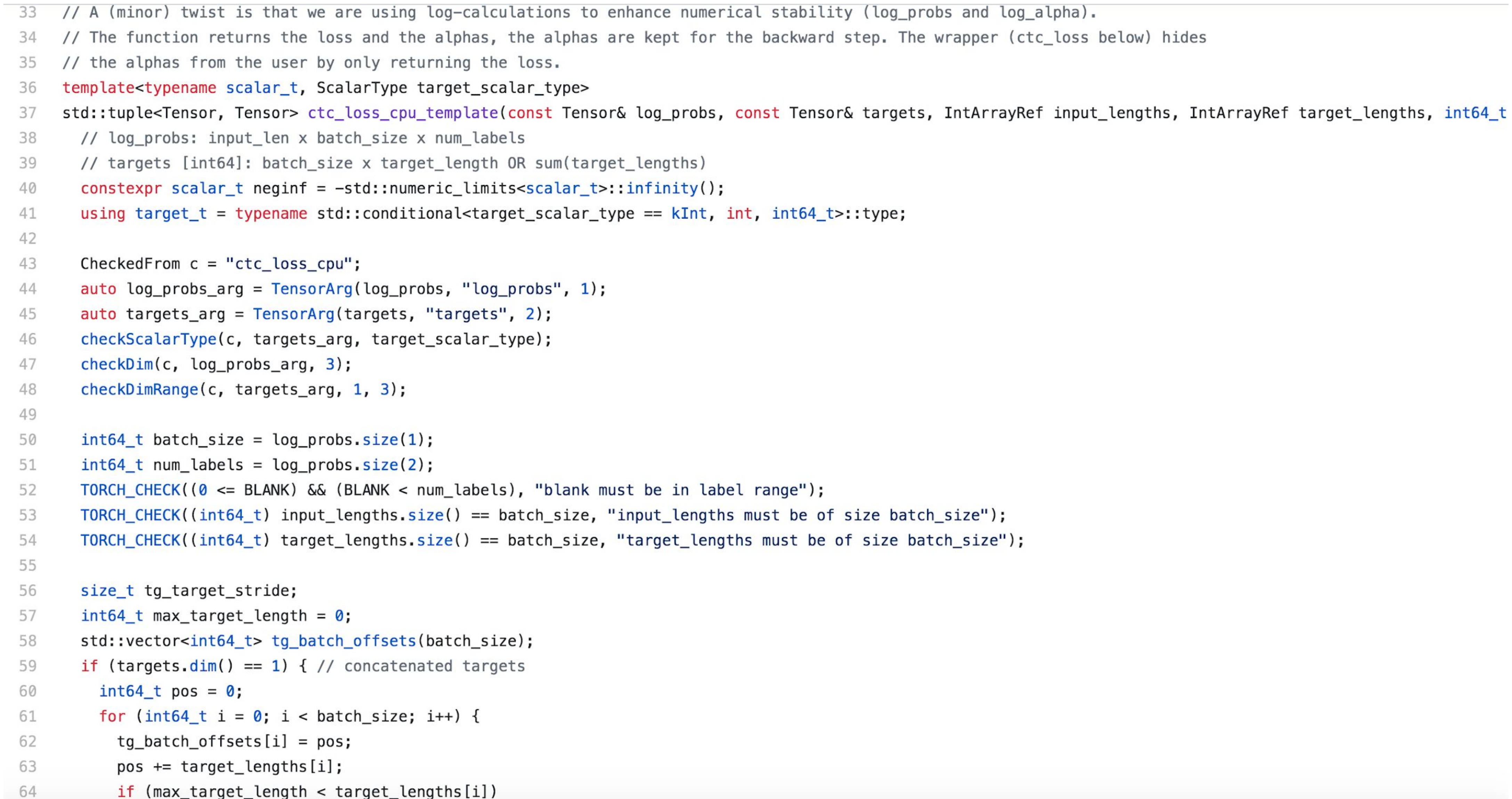


Sep 2019	Sep 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	▲	C#	3.399%	+0.10%

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

Deep learning frameworks are built on C++



```cpp
// A (minor) twist is that we are using log-calculations to enhance numerical stability (log_probs and log_alpha).
// The function returns the loss and the alphas, the alphas are kept for the backward step. The wrapper (ctc_loss below) hides
// the alphas from the user by only returning the loss.
template<typename scalar_t, ScalarType target_scalar_type>
std::tuple<Tensor, Tensor> ctc_loss_cpu_template(const Tensor& log_probs, const Tensor& targets, IntArrayRef input_lengths, IntArrayRef target_lengths, int64_t
  // log_probs: input_len x batch_size x num_labels
  // targets [int64]: batch_size x target_length OR sum(target_lengths)
  constexpr scalar_t neginf = -std::numeric_limits<scalar_t>::infinity();
  using target_t = typename std::conditional<target_scalar_type == kInt, int, int64_t>::type;

  CheckedFrom c = "ctc_loss_cpu";
  auto log_probs_arg = TensorArg(log_probs, "log_probs", 1);
  auto targets_arg = TensorArg(targets, "targets", 2);
  checkScalarType(c, targets_arg, target_scalar_type);
  checkDim(c, log_probs_arg, 3);
  checkDimRange(c, targets_arg, 1, 3);

  int64_t batch_size = log_probs.size(1);
  int64_t num_labels = log_probs.size(2);
  TORCH_CHECK((0 <= BLANK) && (BLANK < num_labels), "blank must be in label range");
  TORCH_CHECK((int64_t) input_lengths.size() == batch_size, "input_lengths must be of size batch_size");
  TORCH_CHECK((int64_t) target_lengths.size() == batch_size, "target_lengths must be of size batch_size");

  size_t tg_target_stride;
  int64_t max_target_length = 0;
  std::vector<int64_t> tg_batch_offsets(batch_size);
  if (targets.dim() == 1) { // concatenated targets
    int64_t pos = 0;
    for (int64_t i = 0; i < batch_size; i++) {
      tg_batch_offsets[i] = pos;
      pos += target_lengths[i];
      if (max_target_length < target_lengths[i])
```

Companies that use C++

amazon.com®

intel®

IBM

facebook®

Google

Microsoft

Adobe®

Browsers written in C++

Software written in C++

Java™

Microsoft® Office™

Ps

Windows

YouTube

Games written in C++

ASSASSIN'S CREED™

CALL OF DUTY®

PORTAL 2™

WORLD OF WARCRAFT®

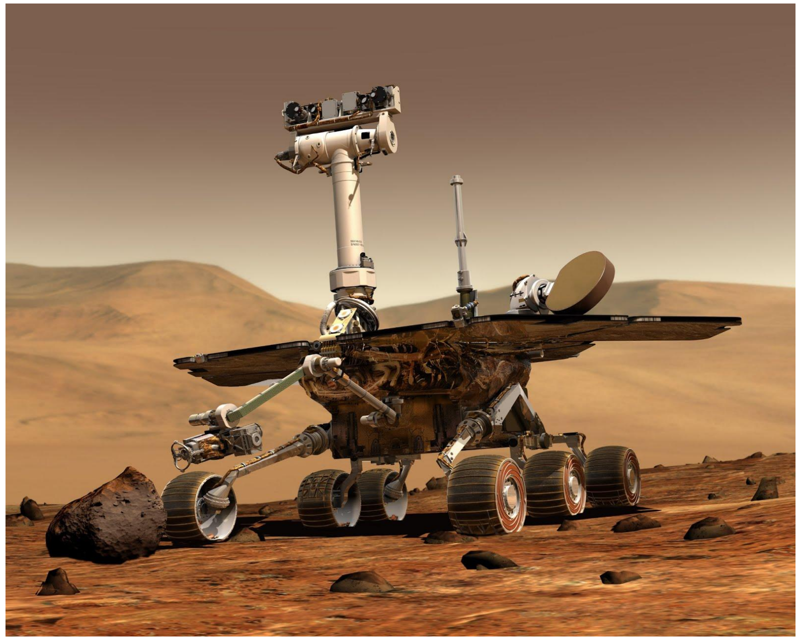
STARCRAFT II

HALO

Other cool stuff written in C++

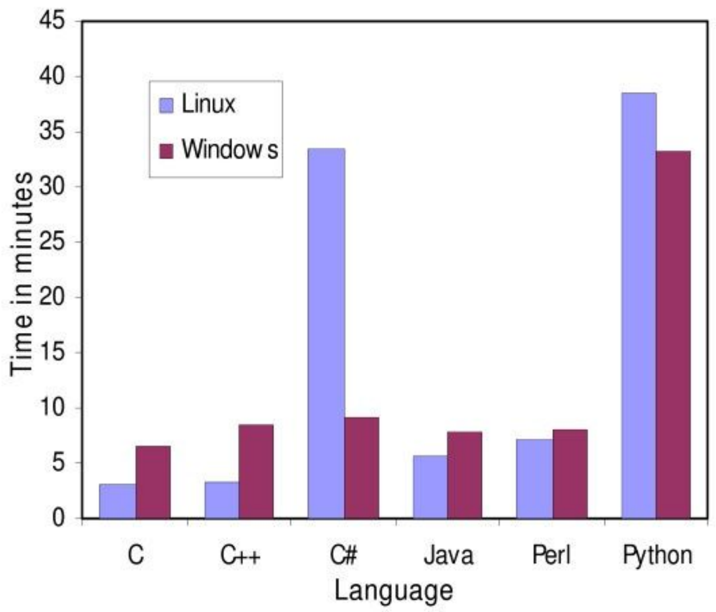
The F-35 Lightning II (Joint Strike Fighter) relies extensively on C++

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

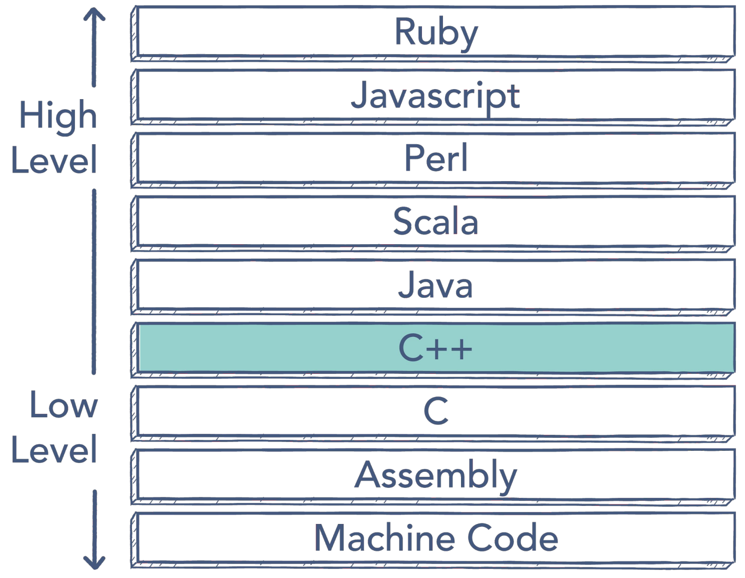


Why C++?

Fast



Lower-level control



Why CS 106L?

Goals of CS 106L

- Learn what features are out there in C++ and why they exist
- Become comfortable reading C++ documentation
- Become familiar with the design philosophy of modern C++

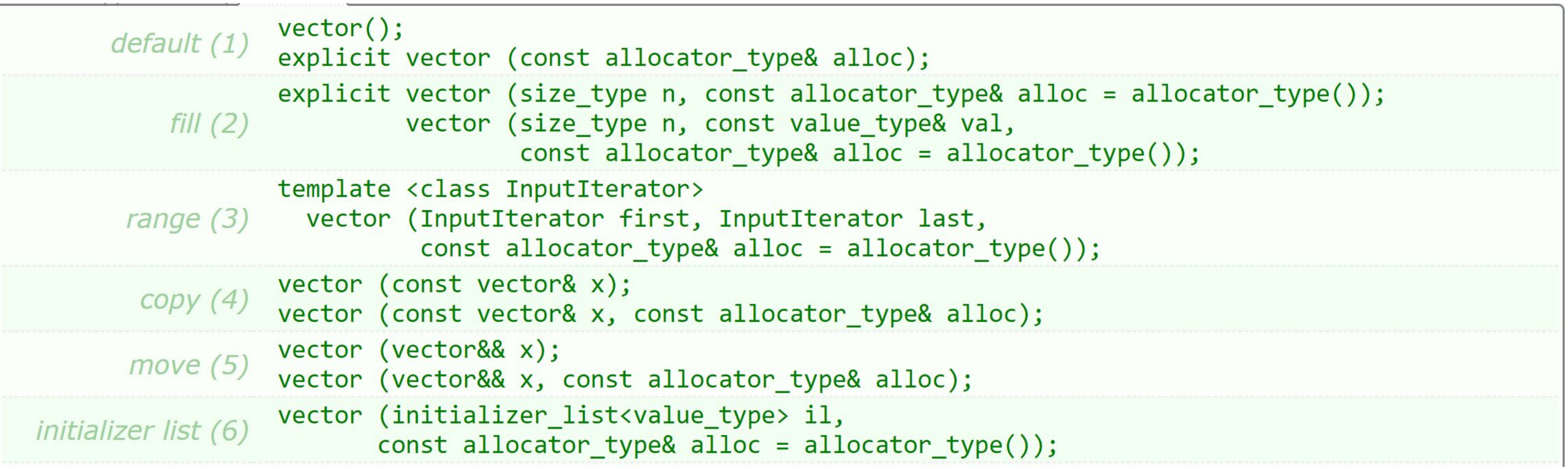
NOT memorize C++ syntax

Rough Outline of Topics

Week	Topic	Lectures
1	**Intro to C++**	Structs
2		References + Streams
3	**STL**	Containers + Iterators
4-5		Template Functions, Functions, STL
6	**Classes**	Template Classes, Const-Correctness
7		Operator Overloading, SMFs
8		Move Semantics, RAII
9	**Wrap-up**	Guest lecture?

C++ documentation is "expert-friendly"

```
vector<int> nums; // the first default constructor
```



default (1)

```
vector();
explicit vector (const allocator_type& alloc);
```

fill (2)

```
explicit vector (size_type n, const allocator_type& alloc = allocator_type());
         vector (size_type n, const value_type& val,
                 const allocator_type& alloc = allocator_type());
```

range (3)

```
template <class InputIterator>
  vector (InputIterator first, InputIterator last,
          const allocator_type& alloc = allocator_type());
```

copy (4)

```
vector (const vector& x);
vector (const vector& x, const allocator_type& alloc);
```

move (5)

```
vector (vector&& x);
vector (vector&& x, const allocator_type& alloc);
```

initializer list (6)

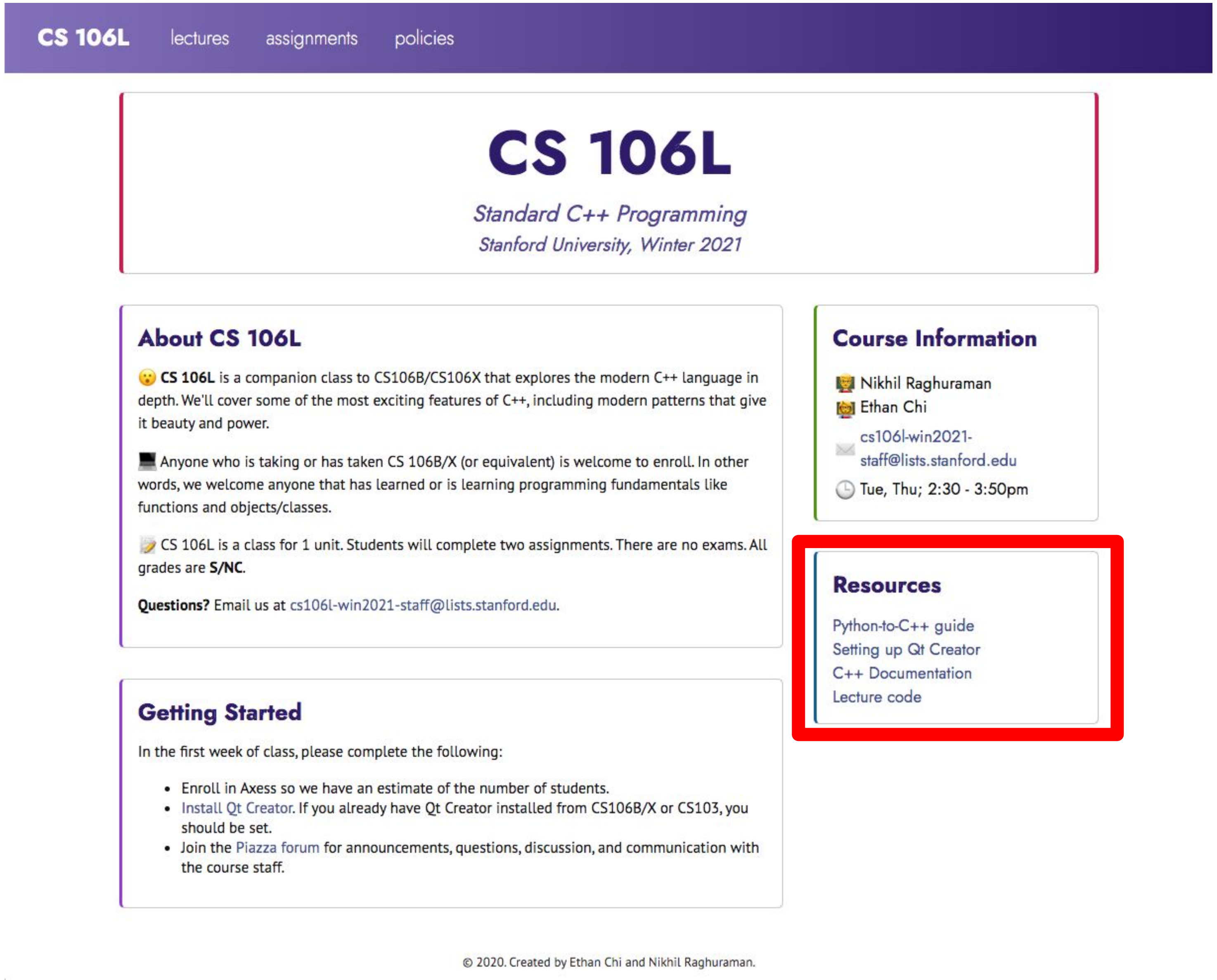
```
vector (initializer_list<value_type> il,
       const allocator_type& alloc = allocator_type());
```

Logistics

Logistics

Lecture:	T/Th 2:30-3:50 (usually ends @ 3:20) on Zoom, weeks 1-9
Website:	https://cs106l.stanford.edu
Getting Help:	Office Hours, Piazza, do not use LaIR
Assignments:	2 assignments, submit both for credit on Paperless
Late Days:	Earn 24-hour late days through surveys
Development:	Qt Creator (from CS 106B)
Honor Code:	Don’t cheat. Same rules as CS 106B.

piazza: https://piazza.com/stanford/winter2020/cs106l/home



CS 106L lectures assignments policies

CS 106L

Standard C++ Programming
Stanford University, Winter 2021

About CS 106L

😮 **CS 106L** is a companion class to CS106B/CS106X that explores the modern C++ language in depth. We'll cover some of the most exciting features of C++, including modern patterns that give it beauty and power.

💻 Anyone who is taking or has taken CS 106B/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.

📝 CS 106L is a class for 1 unit. Students will complete two assignments. There are no exams. All grades are **S/NC**.

Questions? Email us at cs106l-win2021-staff@lists.stanford.edu.

Course Information

Nikhil Raghuraman
Ethan Chi
✉ cs106l-win2021-staff@lists.stanford.edu
🕒 Tue, Thu; 2:30 - 3:50pm

Resources

Python-to-C++ guide
Setting up Qt Creator
C++ Documentation
Lecture code

Getting Started

In the first week of class, please complete the following:

- Enroll in Axess so we have an estimate of the number of students.
- Install Qt Creator. If you already have Qt Creator installed from CS106B/X or CS103, you should be set.
- Join the Piazza forum for announcements, questions, discussion, and communication with the course staff.

🤔 Questions? 🤔

Survey

https://forms.gle/cNFH2YMXyzQBqr5b9

= +1 late day!

History of 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++ code

```
#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

Benefits:

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

Why don’t we always use Assembly?

📝 Answer in the chat.

C++ History: Assembly

Drawbacks:

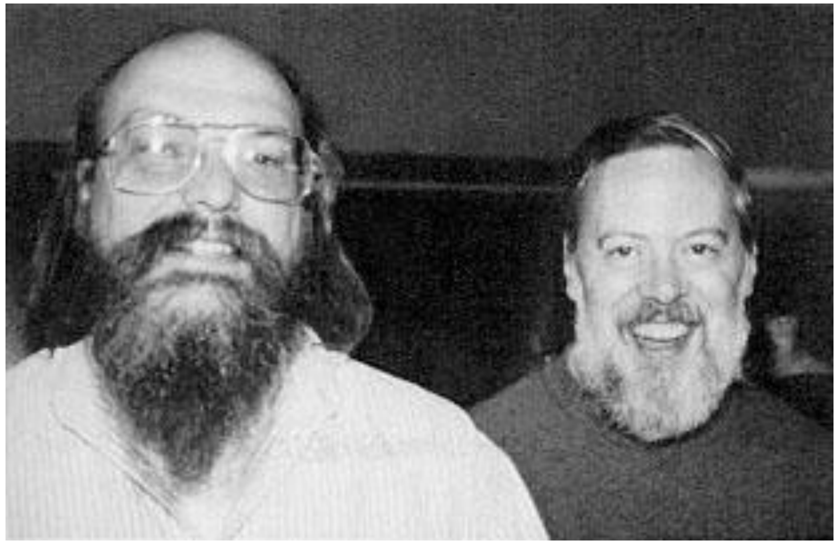
- A lot of code to do simple tasks
- Very hard to understand
- Extremely unportable (hard to make work across all systems)

C++ History: Invention of C

- Problem: computers can only understand assembly!
- Idea:
  - Source code can be written in a more intuitive language
  - An additional program can convert it into assembly
    - This additional program is called a compiler!

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.

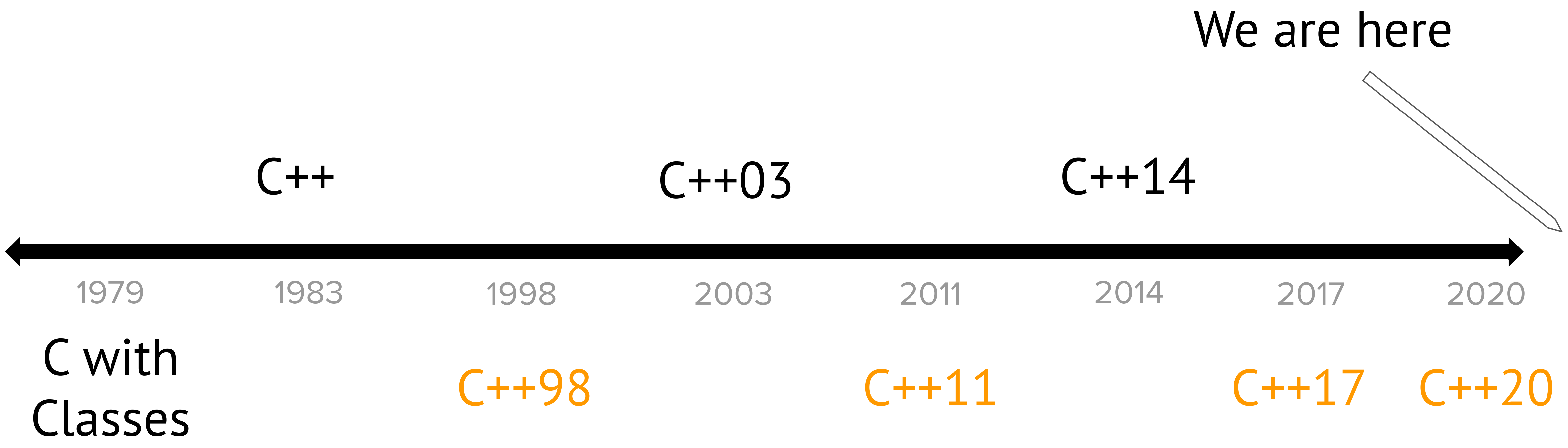
C++ History: Invention of C

- C was popular since it was simple.
- This was also its weakness:
  - No **objects** or **classes**
  - Difficult to write code that worked **generically**
  - 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++



Design Philosophy of C++

Design Philosophy of C++

Give programmers more choice

(at the cost of more responsibility).

Example: **low-level memory access**

Design Philosophy of C++

Catch errors at compile-time

(at the cost of being slightly more verbose).

Example: **mandatory-ish typing**

Design Philosophy of C++

Compartmentalize messy constructs.

Example: **smart pointers**

C++ is...

- Multi-paradigm
- Efficient
- Supports abstraction

~~🤔~~ Questions? ~~🤔~~

Live Code Demo:
Our First C++ Program!

Recap

- C++ is an extremely ubiquitous and important language
- C++ is all about efficiency and transparency of intent
- **Next time:** Structures