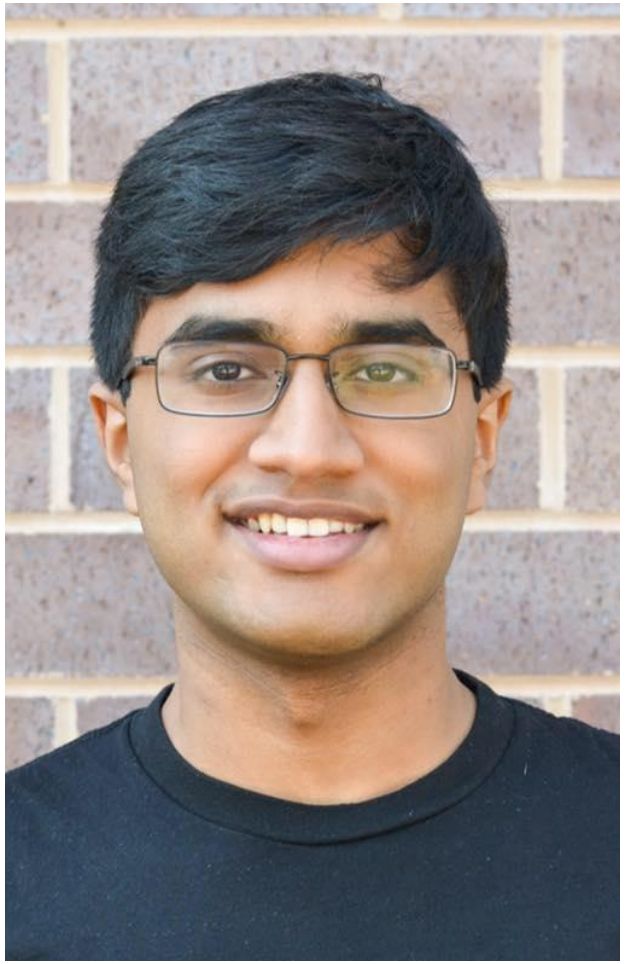
Welcome to CS 106L!

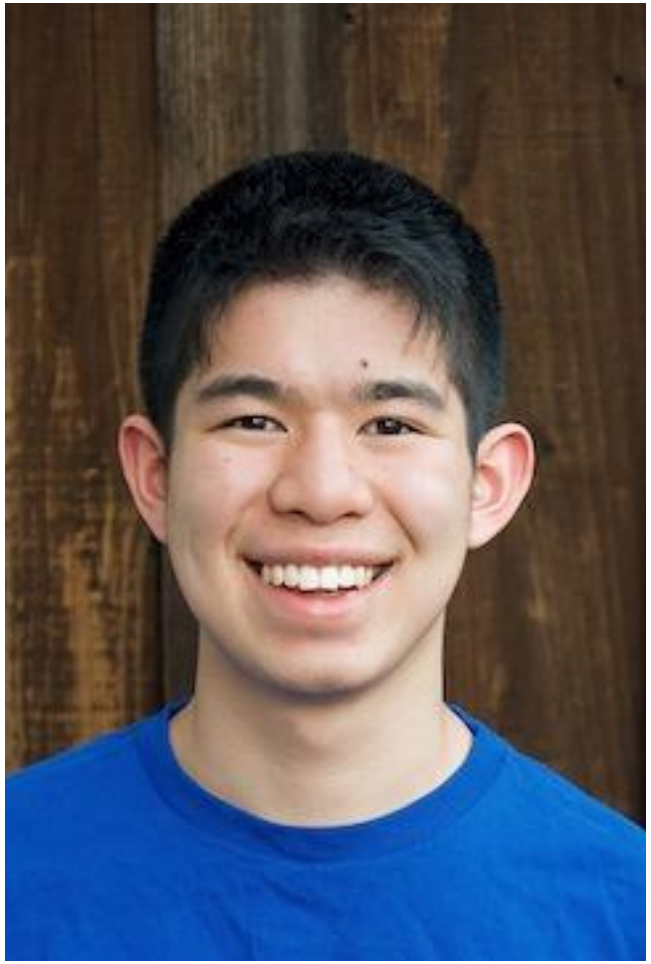
Today's Agenda

- 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%

Take that, Python!

Programming language popularity: C++ bounces back at Python's expense

Broader compiler support is driving a resurgence in interest in the nearly 35-year-old C++ programming language, which replaces Python in Tiobe's top 3.

By Liam Tung | April 8, 2019 -- 12:43 GMT (20:43 GMT+08:00) | Topic: Enterprise Software

5

Python has seen the largest rise of any

MORE FROM LIAM TUNG

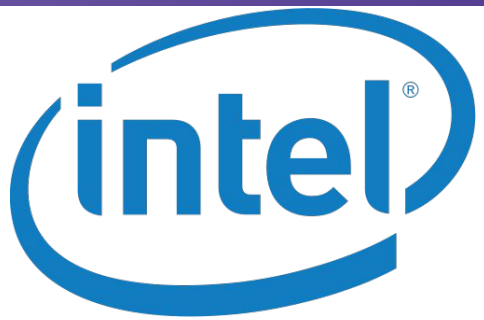
Google

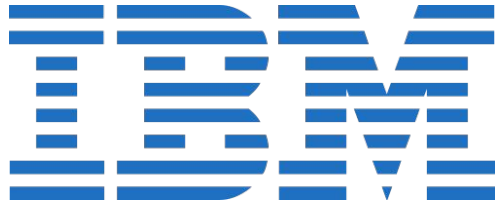
Google: We've changed search rankings to reward 'original news reporting'

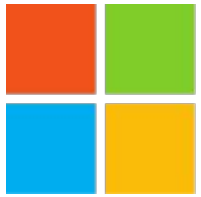
Classes that use C++

- **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++



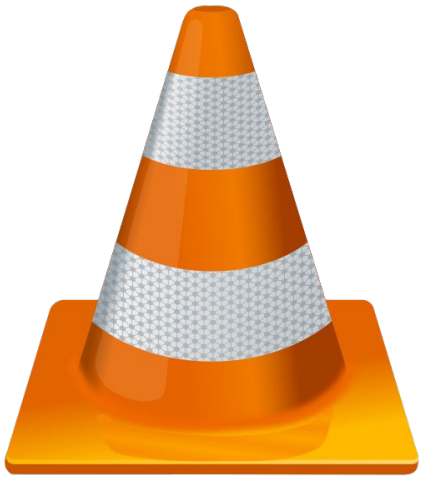




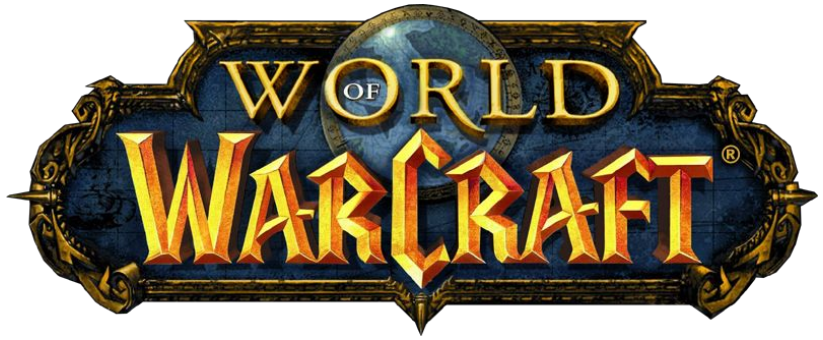
Microsoft

Browsers written in C++

Software written in C++



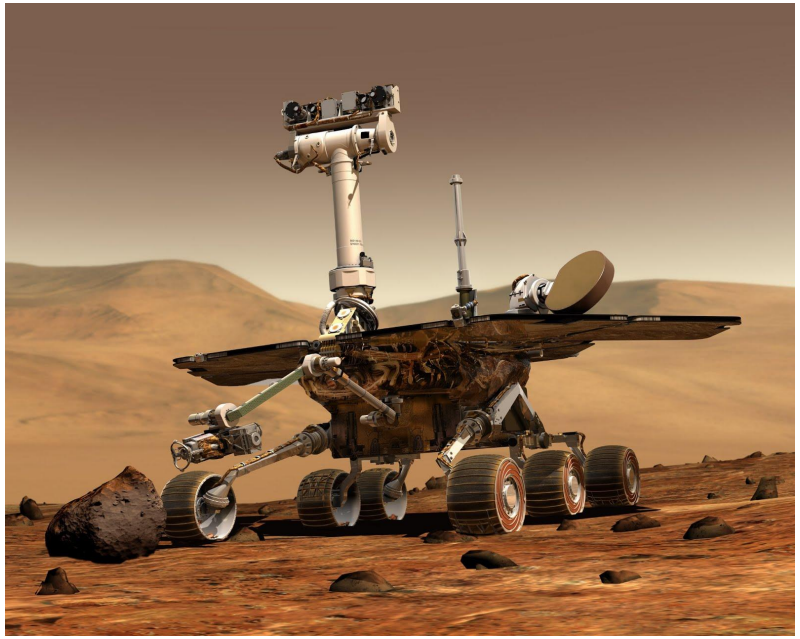
Games written in C++



Other cool stuff written in C++

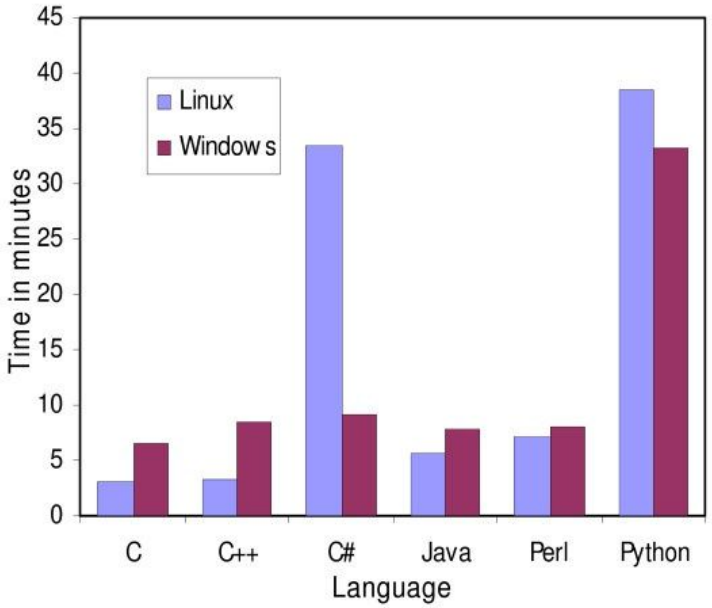
The F-35 Lightning II (Joint Strike Fighter) relies extensively on C++

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

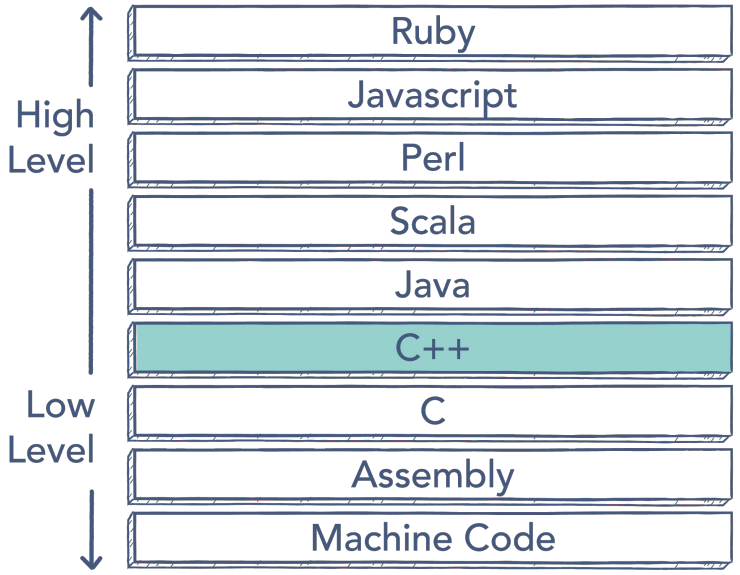


Why C++?

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

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());`

Rough Outline of Topics

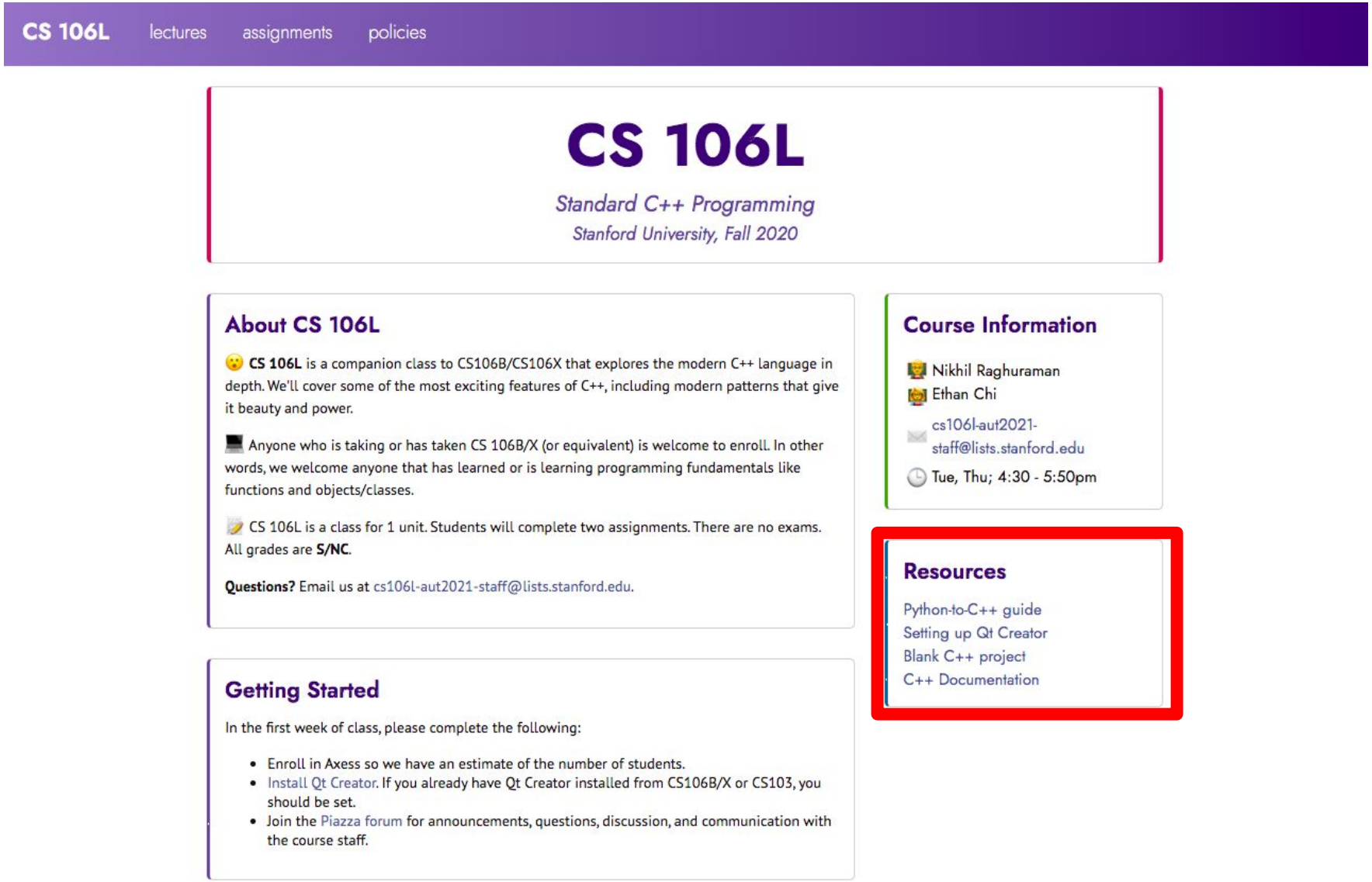
- Basics Week 1: Introduction and Structures
- Basics Week 2: References and Streams
- STL Week 3: Containers and Iterators
- STL Week 4-5: Templates and Algorithms
- Templates Week 5: Template Classes
- Class Design Week 6: Const Correctness and Operators
- Class Design Week 7: Special Member Functions
- Class Design Week 8: RAII
- Bonus Topics Week 8-9: Multithreading and Overflow

Logistics

Logistics

Lecture:	T/Th 4:30-5:50 (usually ends @ 5:20) 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/fall2020/cs106l/home


CS 106L
lectures
assignments
policies
CS 106L
Standard C++ Programming
Stanford University, Fall 2020
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-aut2021-staff@lists.stanford.edu.
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.
Course Information
Nikhil Raghuraman
Ethan Chi
cs106l-aut2021-staff@lists.stanford.edu
Tue, Thu; 4:30 - 5:50pm
Resources
Python-to-C++ guide
Setting up Qt Creator
Blank C++ project
C++ Documentation


Questions?

Survey

https://forms.gle/Ye6wp3Ziz5kxJ1mm7

= +1 late day!

History of C++

Some C++ Code

```cpp
#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:

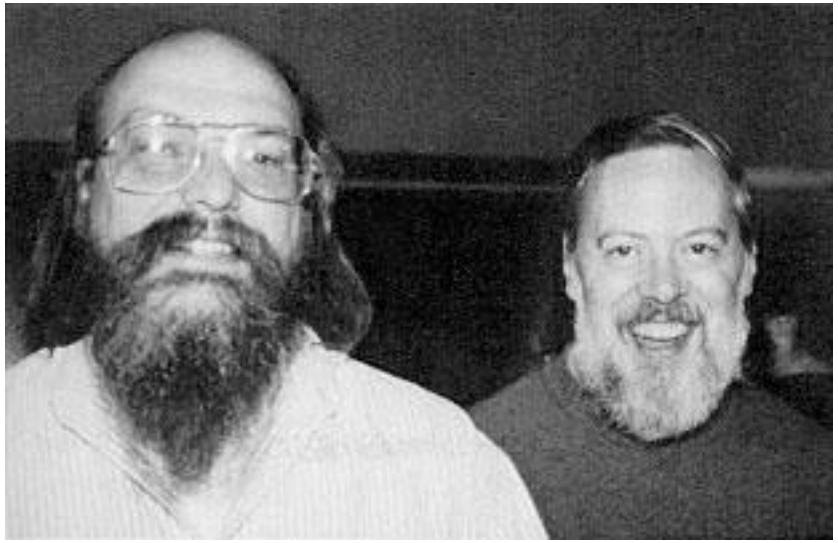
- A lot of code to do simple tasks
- Very hard to understand
- Extremely unportable (hard to make work across all systems)

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!



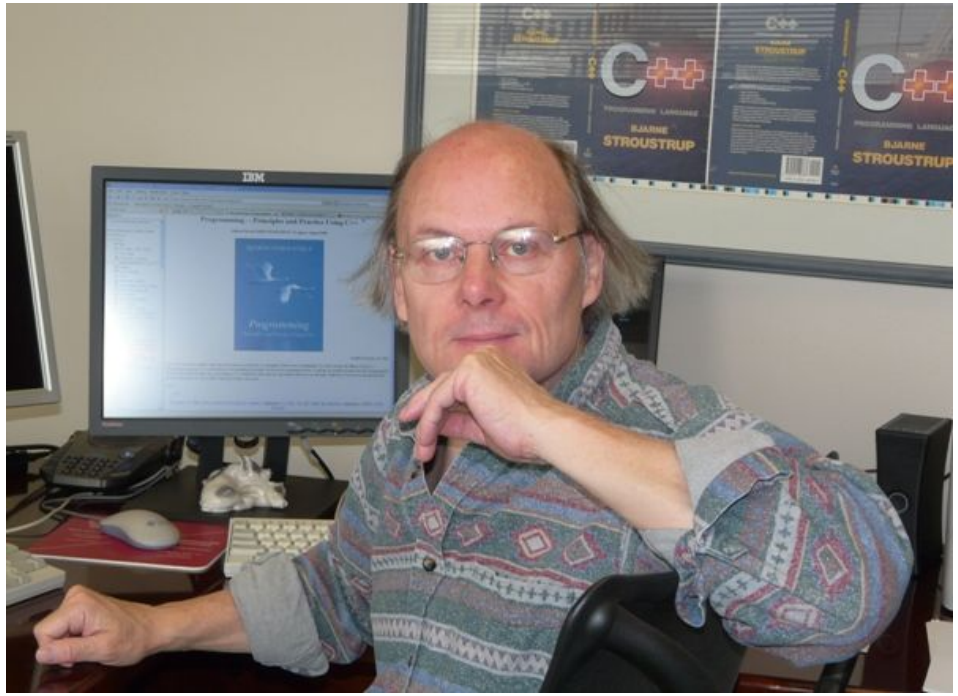
Ken Thompson and Dennis Ritchie, creators of the C language.

C++ History: Invention of C

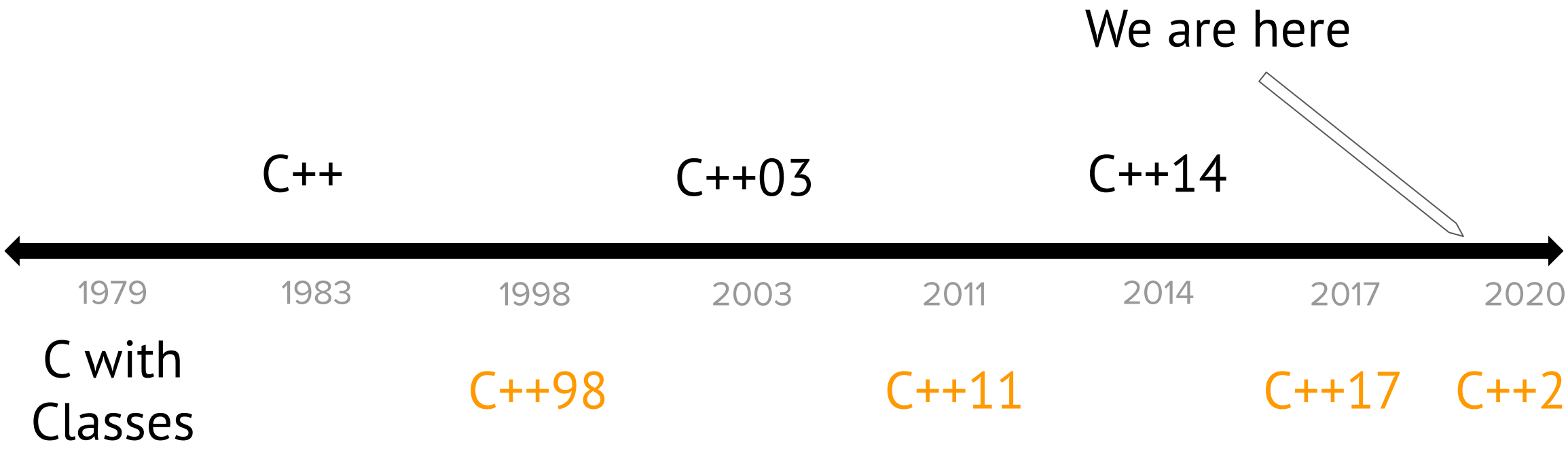
- C was popular since it was simple.
- This was also its weakness:
  - No **objects** or **classes**
  - Difficult to write code that worked **generically**
  - Tedious when writing **large** programs

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++

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

Design Philosophy of C++

- Multi-paradigm 多范式
- Express ideas and intent directly in code.
- Safety
- Efficiency
- Abstraction

什么叫做多范式编程?
编程范式是指编程时的指导思想。放在编程语言里，则代表了这个语言的设计方向，即语言是为了便于遵循某种，或某些思想编程而设计的。多范式编程语言中的多范式，是指这个语言支持使用者采用多种不同的编程范式来撰写程序。C语言是过程式编程语言；Java、C#是面向对象式编程语言；Haskell是函数式编程语言；而C++则是多范式的编程语言。


Questions?

Live Code Demo:
Our First C++ Program!

SUPPLEMENTAL CONTENT: Command-line Compilation (Alternative to Qt Creator)

CL Compilation

- For our assignments and in CS106B, you'll use QT Creator to compile your code. However, QT Creator isn't the only way to compile C++ code!
- Today we will briefly cover how to do this in the terminal.
- First we should understand how C++ compilation works.

CL Compilation

1. Preprocessor - Deals with `#include`, `#define`, etc directives
2. Compiler - Converts C++ source code into assembly
3. Assembler - Turns assembled code into object code (.o files)
4. Linker - Object files are linked together to make an executable program

Preprocessor

Responsible for everything starting with a #

```
#include

#define

#ifndef

#pragma
```

Compilers

- Converts each .cpp source file into **assembly**.
- This process is localised to each file.
- Outputs .s files

Assembler

- Turns previously generated assembly code into **object code**.
- Outputs .o files.
- Still no intercommunication between separate cpp files.

Linker

- Combines all the separate object files into one **executable** file.
- In previous phases we only looked at one file at a time.
- The linker is the first place where files are **combined**.

Linker

- Linker checks that every declared function has an implementation.
- That's why you might see errors like this:

```
Linker error: symbols not found for
architecture x86
Linker error: duplicate symbols found
for architecture x86
```

Let's try it ourselves!

- We will use g++ as our compiler.
  - Macs should have g++ automatically. On Windows, see this link to download.
- Basic usage:

```
g++ main.cpp otherFile.cpp -o execFileName
```

Let’s try it ourselves

- We will use three common compiler flags:
- `-std=c++14`
  - Enable C++14 support
- `-g`
  - Add debugging information to the output
- `-Wall`
  - Turn on most compiler warnings

Recap

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