Welcome to CS 106L!

Reminder: class starts at 3:30 PM
Game Plan

- Welcome
- Why C++ and CS 106L?
- Logistics
- History and Philosophy of C++
- C++ Basics
Introduction
Instructors
Why C++?
C++ is still a very popular language.

<table>
<thead>
<tr>
<th>Sep 2019</th>
<th>Sep 2018</th>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Java</td>
<td>16.661%</td>
<td>-0.78%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>C</td>
<td>15.205%</td>
<td>-0.24%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td>Python</td>
<td>9.874%</td>
<td>+2.22%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>C++</td>
<td>5.635%</td>
<td>-1.76%</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>^</td>
<td>C#</td>
<td>3.399%</td>
<td>+0.10%</td>
</tr>
</tbody>
</table>
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

Python has seen the largest rise of any...
Classes that use C++

BIOE 215: Physics-Based Simulation of Biological Structure
CME 213: Introduction to parallel computing using MPI
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++
Browsers written in C++
Software written in C++
Games written in C++
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.
“One of the things I really like about programming languages is that it's the perfect excuse to stick your nose into any field. So if you're interested in high energy physics and the structure of the universe, being a programmer is one of the best ways to get in there. It's probably easier than becoming a theoretical physicist”

-Bjarne Stroustrup
Why CS 106L?
Comparison of 106B vs. 106L

**CS 106B**
Programming Abstractions

**CS 106L**
Standard C++ Programming
Comparison of 106B vs. 106L

**CS 106B**  
Applicable to all programming languages

**CS 106L**  
Applicable mostly to C++ (perhaps others)
## Comparison of 106B vs. 106L

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CS 106B</strong></td>
<td>Barely enough C++ to learn programming abstractions.</td>
</tr>
<tr>
<td><strong>CS 106L</strong></td>
<td>Enough C++ for a job, internship, or research.</td>
</tr>
</tbody>
</table>
Comparison of 106B vs. 106L

CS 106B: C++98*

CS 106L: C++17
   (sneak peek into C++20)

*exception: range-based for loop.
## Comparison of 106B vs. 106L

<table>
<thead>
<tr>
<th>CS 106B</th>
<th>Stanford libraries abstract away messy details.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 106L</td>
<td>Deep dive into messy C++ details.</td>
</tr>
</tbody>
</table>
Comparison of 106B vs. 106L

**CS 106B**

“Just use getInteger”

**CS 106L**

“Let’s tame cin”
Goals of CS 106L

1. Learn what features are out there in C++ and why they exist.

2. Become comfortable with reading C++ documentation.

3. Become familiar with the design philosophy of modern C++.

**NOT:** memorize the syntax of C++. 
C++ documentation is not beginner friendly.

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

- **default (1)**
  - `vector()`: explicit vector (const allocator_type& alloc);
  - `vector (size_type n, const allocator_type& alloc = allocator_type());`
  - `vector (size_type n, const value_type& val,  
    const allocator_type& alloc = allocator_type());`

- **fill (2)**
  - `vector (const vector& x);`
  - `vector (const vector& x, const allocator_type& alloc);`

- **copy (4)**
  - `vector (vector&& x);`
  - `vector (vector&& x, const allocator_type& alloc);`

- **move (5)**
  - `vector (initializer_list<value_type> il,  
    const allocator_type& alloc = allocator_type());`
Logistics
Logistics

Lecture: T/Th 3:30-4:20 in 380-380C, weeks 1-9
Website: https://cs106l.stanford.edu
Getting Help: Office Hours, Piazza, do not use LaIR
Assignments: 3 total, complete 2 satisfactorily for credit
Late Days: Two 24-hour late days
Development: Qt Creator (from CS 106B)
Honor Code: Don’t cheat. Same rules as CS 106B.

Piazza: https://piazza.com/stanford/fall2019/cs106l
QT Creator Setup

Troubleshooting session:
Thursday, 9/26, 7:30 - 9:30 pm at LaIR
Comparison of 106B vs. 106L

CS 106B  3 lectures + section / week

CS 106L  2 lectures / week
Comparison of 106B vs. 106L

**CS 106B**
undergrad 5, grad 3-5 units
letter grade or C/NC

**CS 106L**
1 unit S/NC
auditors welcome!
Comparison of 106B vs. 106L

**CS 106B**  
7 assignments + section + exams

**CS 106L**  
3 assignments, choose 2
History of C++
Some C++ Code

```cpp
#include <iostream>

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

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

int main(int argc, char *argv) {
    printf("%s", "Hello, world!\n");
    return EXIT_SUCCESS;
}
```
...Also (Technically) Some C++ Code

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

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

```
section .text
global _start

_start:
    ; must be declared for linker (ld)
    ; 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
len    equ $ - msg
        ; our dear string
        ; 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?
C++ History: Assembly

section .text
global _start

_start:
    ;must be declared for linker (ld)
    ;tell linker entry point
    mov edx, len
    mov ecx, msg
    mov ebx, 1
    mov eax, 4
    int 0x80
    int 0x80

section .data

msg db 'Hello, world!',0xa
len equ $ - msg

;message length
;message to write
;file descriptor (stdout)
;system call number (sys_write)
;call kernel
;system call number (sys_exit)
;call kernel

;our dear string
;length of our dear string
C++ History: Assembly

FAST, but...

Simple instructions?

→ Requires lots of code to do simple tasks

→ Hard to understand other people’s code

Complete control over program?

→ Extremely unportable
C++ History: Invention of C
C++ History: Invention of C

Writing assembly was too difficult but computers only understood assembly.

Idea:

- Source code can be written in a more intuitive language
- An additional program can convert it into assembly

This is called a compiler!
C++ History: Invention of C

K&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 first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

- Fast
- Simple to Use
- Cross-platform
- Had high level features
In 1983, the first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

- Fast
- Simple to Use
- Cross-platform
- Had high level features
In 1983, the first vestiges 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
- 1983: C with Classes
- 1998: C++98
- 2003: C++03
- 2011: C++11
- 2014: C++14
- 2017: C++17
- 2020: C++20

We are here
Design Philosophy of C++
Design Philosophy of C++
Design Philosophy of C++

• Allow the programmer full control, responsibility, and choice if they want it.
• 
• 
• 
•
Design Philosophy of C++

• Allow the programmer full control, responsibility, and choice if they want it.
• Express ideas and intent directly in code.
Finding the sum of a vector of ints.

```cpp
vector<int> vec = {1, 2, 3};
int sum = 0;
for (auto val : vec) {
    sum += val;
}
```

```cpp
vector<int> vec = {1, 2, 3};
int sum = 0;
for (const auto& val : vec) {
    sum += val;
}
```

```cpp
vector<int> vec = {1, 2, 3};
int sum = std::accumulate(vec.begin(), vec.end(), 0);
```
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.
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.
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++

- 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.
Example

Our first C++ program
Survey

https://bit.ly/2mLb60Y

= +1 late day!
Next time

Streams
Looking Ahead

- streams abstraction
- stringstream
- state bits
- i/o streams and buffering
- [CS 106B - Friday] file streams and the Stanford library
- types: type inference, structures, initialization
- error-checking and implementing simpio.h
- manipulators
- overloading stream << and >>
- sequence containers