# Welcome to CS106L

Reid Watson
(rawatson@stanford.edu)

# Lecture 0: The Weird One

- Administrivia
- Why CS106L
  - Why do we have a special class about C++?
- C++
  - What does the universe of programming languages look like?
  - What role does C++ fill?

# Lecture 0: The Weird One

- Administrivia
- Why CS106L
  - Why do we have a special class about C++?
- C++
  - What does the universe of programming languages look like?
  - What role does C++ fill?

# Administrivia

- Staff
  - Instructor: Reid Watson (rawatson@stanford.edu)
  - Don't be afraid to get in touch, I love hearing from students!
  - TAs are still being determined
- Office hours
  - Tuesday, Thursday: 5:05 - 6:00 PM
  - There will also be CS106L specific LaIR hours
- Prerequisites
  - You should either be enrolled in CS106B/X right now, or have successfully completed it in an earlier quarter

# Administrivia

- One unit, Credit/No Credit
- Assignments
  - 3 programming assignments, no exams/sections
- Website
  - cs106l.stanford.edu (L as in lima)
  - Everything is there
- Late Policy
  - You will get three 24 hour late days this quarter
  - You can use one late day per assignment
- Honor Code
  - Same as CS106B/X

# Lecture 0: The Weird One

- Administrivia
- Why CS106L
  - Why do we have a special class about C++?
- C++
  - What does the universe of programming languages look like?
  - What role does C++ fill?

# What Exactly is CS106L?

- CS106B/X: Learn how to model and solve complex problems with computers
  - Explore common abstractions for representing problems
  - Harness recursion and think about problems recursively
  - Quantitatively analyze different approaches for solving problems
- CS106B/X uses C++ as a means to an end

# What Exactly is CS106L?

- CS106L: Learn how and why to write clear, efficient, and powerful C++ code
  - Understand industry standard C++ coding practices
  - Become a better programmer by practicing these skills
  - Understand the world of programming a bit better by studying C++

# "Industry Standard C++"

- C++ allows us to solve the same problem many different ways
- Understanding why you should write code a certain way is key to being a good C++ programmer
- To understand why this matters, let's look at how we write "hello world" in C++

# Hello World in C++

```
#include <iostream>
using namespace std;
int main() {
   cout << "Hello World!" << endl;
}
```

# Hello World in C++

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   puts("Hello world");
   return EXIT_SUCCESS;
}
```

# Why C++?

- Why does CS106B/X use C++?
- Why don't we learn {language X} instead?
- Why do programmers use C++?
- Is C++ going to be useful to me?
- Can I get a job using C++?

To answer these questions, we'll need to understand C++ a little bit better...

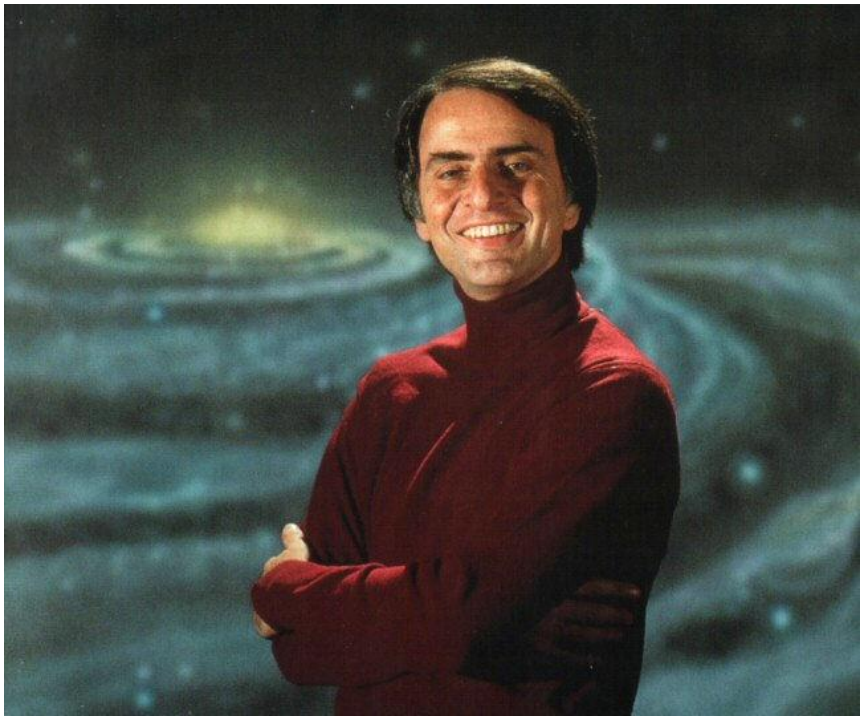
# Lecture 0: The Weird One

- Administrivia
- Why CS106L
  - Why do we have a special class about C++?
- C++
  - What does the universe of programming languages look like?
  - What role does C++ fill?

# The Universe of Programming

"If you wish to make an apple pie from scratch, you must first invent the universe"

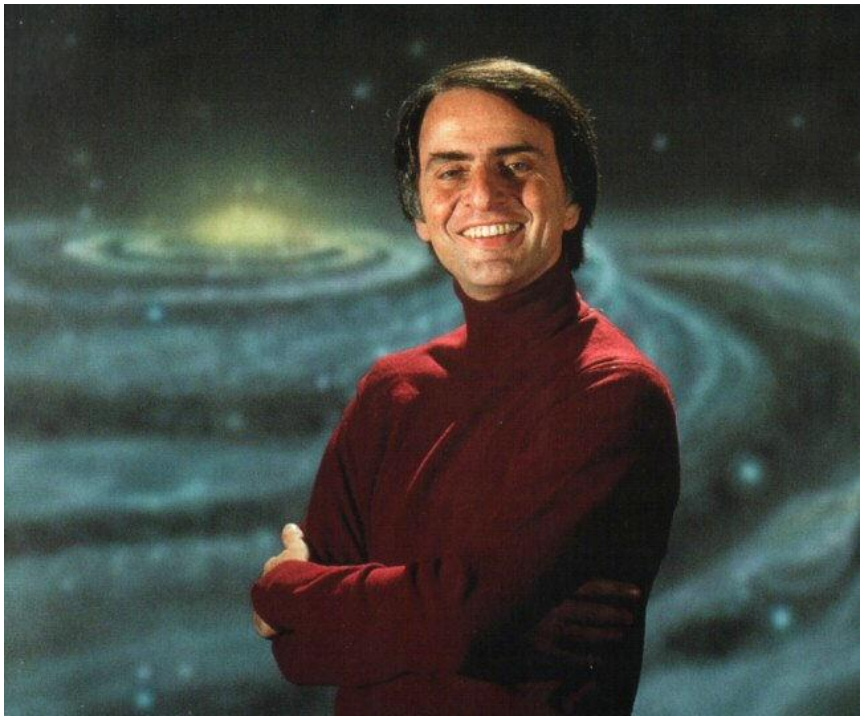
-- Carl Sagan



# The Universe of Programming

"If you wish to understand C++, you must first invent the universe"

-- Carl Sagan



# The Universe of Programming

To understand the universe of programming, we'll start by looking at the most basic programming language used, assembly.

# Assembly

Here's an example of a "hello world" program in 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
```

# Assembly

- Computers operate by executing a series of **instructions**.
  - Instructions are things like "add two numbers", or "store this value into memory"
- **Assembly language** describes exactly which instruction the computer should execute
- When you write assembly, you have complete control over your program
- You'll learn more about assembly in CS107

# Assembly

- Well written assembly is extremely fast
  - Any construct which the computer can execute can be expressed in assembly

If all computer programs **can** be written in assembly, why don't we just learn assembly?

# Assembly

- Assembly code can be extremely difficult to understand
  - Specifying **exactly** what the computer should do can take a very long time!
  - A 5 line C++ program might involve hundreds of lines of assembly code
- Assembly code written for one type of computer might not work on another
  - Laptops typically use a set of instructions called **x86**, while phones typically use **ARM**

# Higher Level Programming

- Writing assembly is too difficult
- **Source code** is written in a **programming language** like Java or C++.
- **Compilers** transform source code into assembly

# Higher Level Programming

**Source Code**

```
#include <iostream>
using namespace std;
int main() {
     cout << "Hello
World!";
     cout << endl;
}
```

**Compiler**

**Assembly**

```
section     .text
global      _start

_start:
    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80
    mov     eax,1
    int     0x80

section     .data
msg     db  'Hello World!',
0xa
len     equ $ - msg
```

# Higher Level Programming

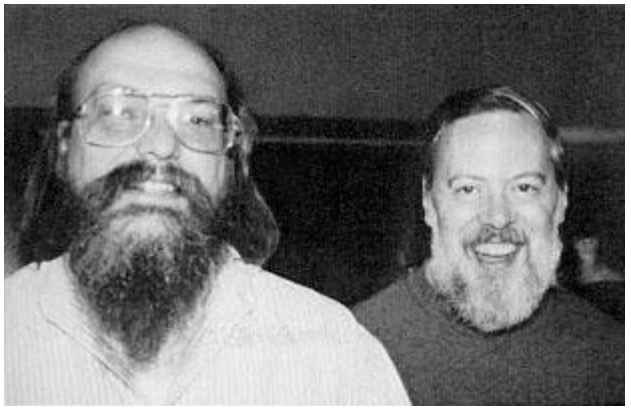
- What should source code look like?
  - How should the idea of a function be represented in source code?
  - Should a programming language offer a large standard library, or a small one?
  - What syntax should we use to represent various operations

# Higher Level Programming

- There are a lot of programming languages because people have very different ideas what source code should look like
  - Some languages are better at producing code which executes quickly
  - Some languages try and make it easy to express complex ideas in a simple manner
  - Some languages are designed with a specific application in mind

# History of C++: C

- The C programming language was released in 1972
- C was a big hit -- it made writing efficient code which worked on different machines very easy
- Take CS107 to learn more!



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

# Hello World in C

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   puts("Hello world");
   return EXIT_SUCCESS;
}
```

# History of C++: C

- C was popular in part because it was a "simple" language
- C  wasn't that much more complicated than assembly language
  - Writing good C code doesn't require memorizing a lot of unusual syntax
  - Creating a C compiler for a system wasn't too complicated (relatively speaking)
  - C code would work on any system which had a C compiler

# History of C++: C

- C was criticized in part because it was a "simple" language
  - The C language doesn't provide things like maps and objects.
  - Some parts of C (C strings especially) are difficult to work with and C doesn't provide any alternatives

# Introducing C++

- Bjarne Stroustrup created C++ in 1983 to add new features to C
- C++ was also a massive hit, and quickly became one of the most popular languages

Bjarne Stroustrup, living large and creating C++

# What is C++ Anyway?

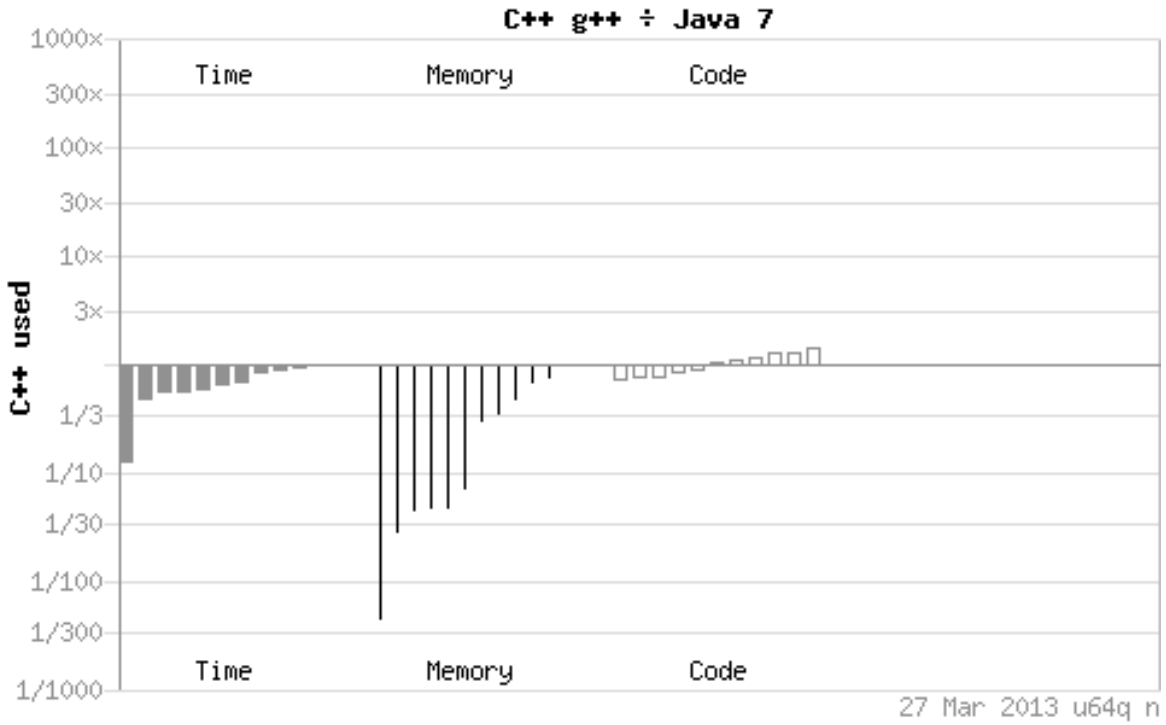
C++ is a programming language which simplifies complex tasks without sacrificing performance

# What is C++ Anyway?

- (Almost) all C code is valid C++
- What constitutes valid C++ code is defined by the ~1400 page C++ standard
  - C++ is definitely not a "simple" language!
- We're going to focus on the important parts of C++ in CS106L
- But first, let's answer the original question:
  - Why is C++ so popular / used in CS106B / important enough to deserve its own class?

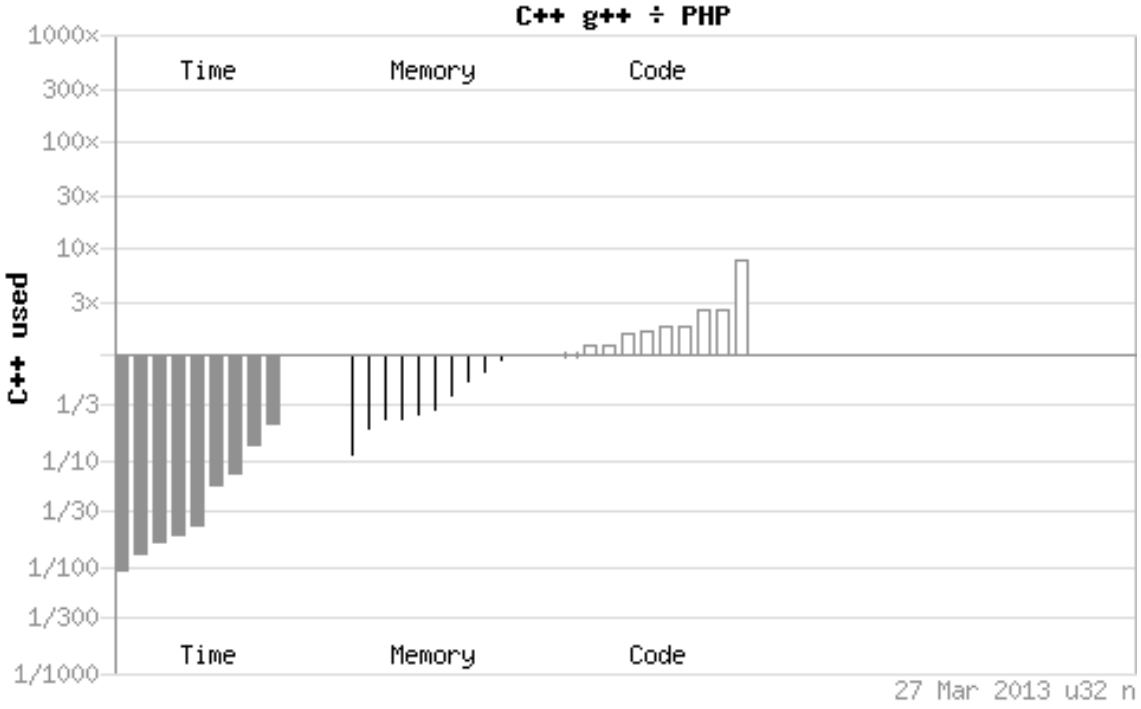
# Why C++: Performance

Well written C++ offers some of the highest performance of any language currently used



# Why C++: Performance

Facebook began using C++ and C++ based tools to help them handle the huge amounts of traffic they get every day



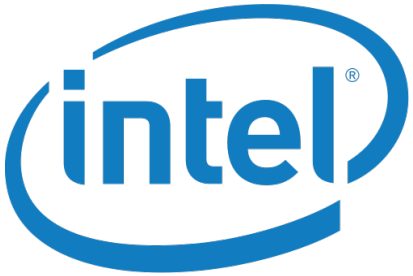
# Why C++: Users

C++ is one of the most widely used languages in the world.

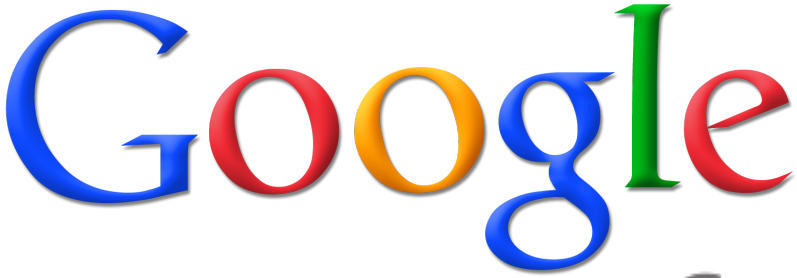
Let's peek at some examples...

# Why C++: Users (companies)



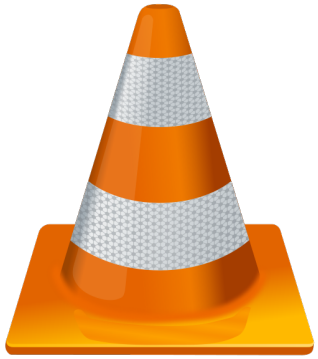





# Why C++: Users (web browsers)

# Why C++: Users (software)






# Why C++: Users (games)

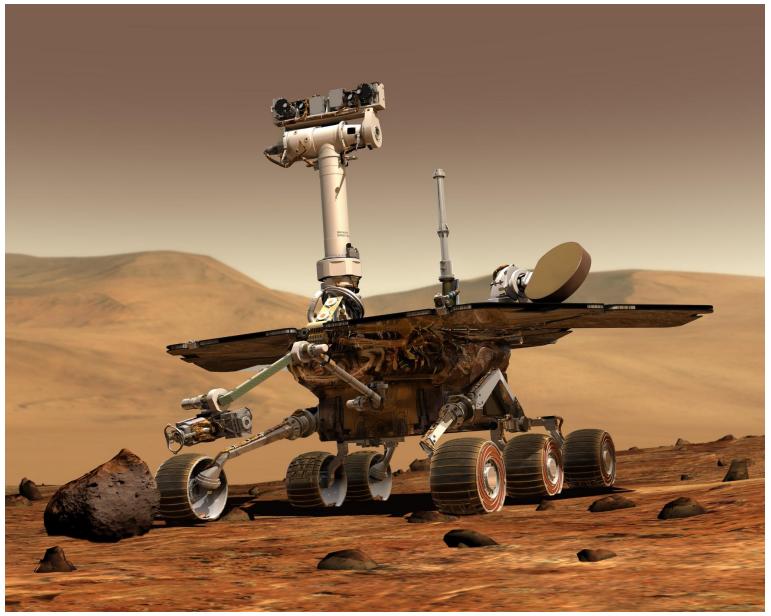





# Why C++: Users (cool things)

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++: Users (Java)



The most widely used Java runtime is written in C++

# Why C++: The Big Picture

C++ is a great way to understand how to model complex ideas in a straightforward manner

You can learn a lot about computer programming by studying why things are the way they are in C++

# Tomorrow

- We start writing real code!
- C++ streams and stream manipulation
- Check the website for more information