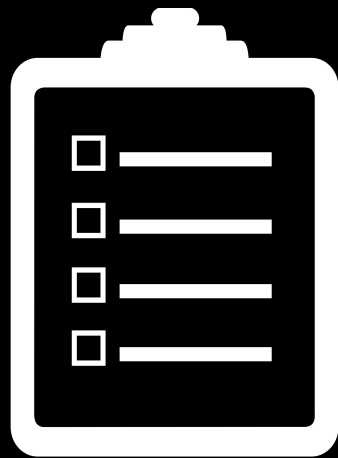


Welcome to CS 106L!

Ali Malik

malikali@stanford.edu

Game Plan



Welcome

Why CS106L?

Logistics

History and Philosophy of C++

C++ Basics

Welcome!

Instructor

Ali Malik

malikali@stanford.edu



Instructor

Ali Malik

malikali@stanford.edu

Email me whenever you have questions or even if you just want to talk. I'd love to hear from all of you!

Tell me about the things you are working on, or anything you find interesting.

You

You

MCS

Petroleum Eng.

Civil and
Environmental Eng.

Undeclared

MS&E

Biology

Statistics

Symbolic Systems

Physics

Music

Aero/Astro

Computer Science

Mathematics

Education

Economics

Energy Resource Eng.

Material Science

Electrical Eng.

MBA

Mechanical Eng.

Why CS106L?

What is CS106L?

CS106B/X:

- Focus is on teaching **concepts** like abstractions, recursion, pointers etc.
- Only teaches you enough C++ to practice these concepts.

CS106L:

- Learn how to write **powerful** and **elegant** code.
- Write **actual** C++ - no Stanford libraries!
- Understand the design decisions that lead to “good” code

Good C++ Code

```
#include <iostream>

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

“Good” C++ Code

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

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

“Good” C++ Code

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

int main(int argc, char *argv) {
    asm( "sub    $0x20,%rsp\n\t"
        "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;
}
```

~~“Good”~~ Terrible C++ Code

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

int main(int argc, char *argv) {
    asm( "sub    $0x20,%rsp\n\t"
        "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;
}
```



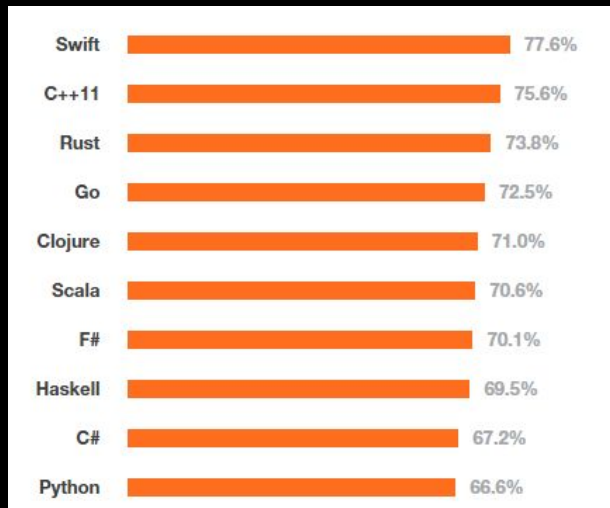
Why Learn C++?

Why C++: Popularity

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	2		C	7.742%	-6.86%
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%

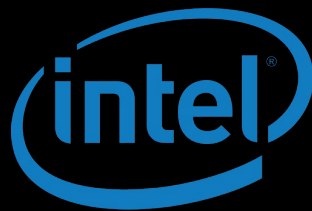
TIOBE Index, March 2017

Why C++: Popularity

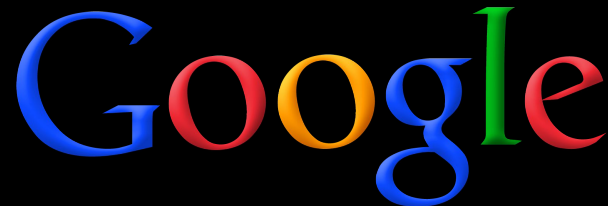


Most loved languages - StackOverflow 2015

Why C++: Users (companies)



facebook®



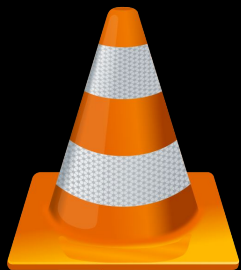
Microsoft



Why C++: Users (browsers)



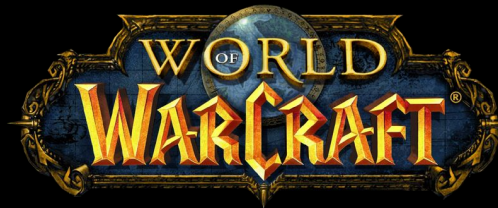
Why C++: Users (software)



Why C++: Users (games)



CALL^{OF}DUTY[®]



HALO 

MASS[™]
EFFECT[™]

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



Logistics

Logistics

Lecture T/Th 1:30 - 2:20pm in Shriram 104.

Website cs106l.stanford.edu

Office Hours TBA. Email if any help needed till then.

Assignments 3 in total - you need to do at least 2 to pass.

Late Days Three 24-hr late days (max two per assignment).

Development We will be using QT Creator. Get this set up quickly!

Honor Code Don't cheat, repercussions are severe.

Logistics

Lecture T/Th 1:30 - 2:20pm in 200-030.

Website cs106l.stanford.edu

Office Hours TBA. Email if any help needed till then.


Assignments 3 in total - you need to do at least 2 to pass.

Late Days Three 24-hr late days (max two per assignment).

Development We will be using QT Creator. Get this set up quickly!

Honor Code Don't cheat, repercussions are severe.

QT Help hours
this Thursday

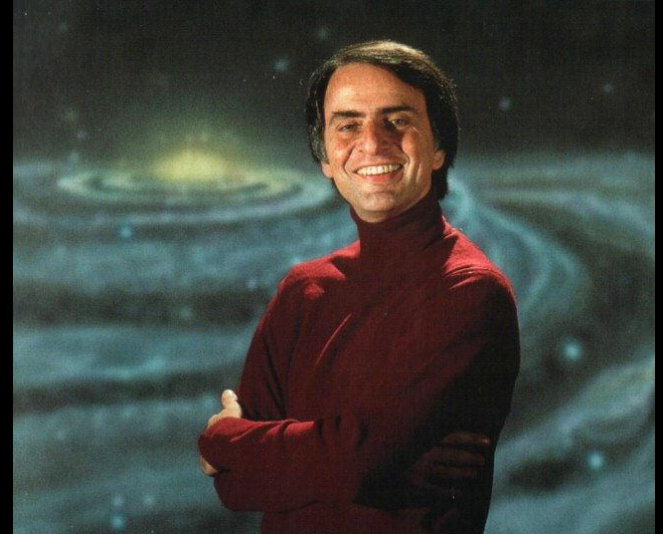


What is C++?

C++ History

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

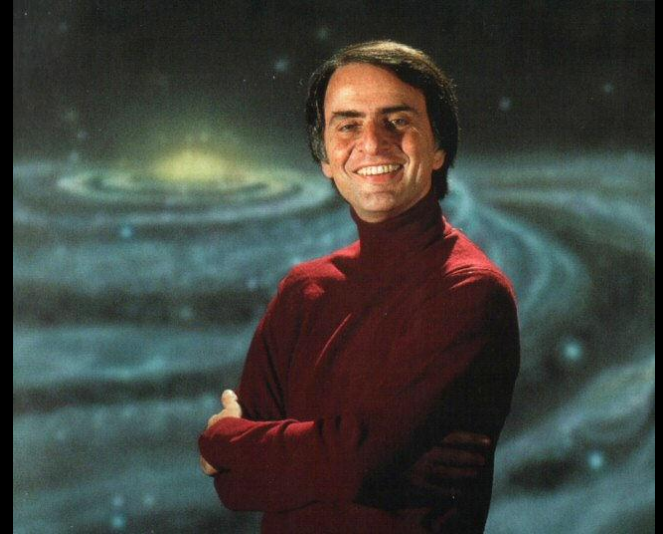
- *Carl Sagan*



C++ History

“If you wish to **understand C++** from scratch, you must first invent the universe”

- *(not) Carl Sagan*



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 (move bits around, add, subtract).

Well written assembly is **extremely** fast.

Gives you complete control over your program.

Why don't we always use assembly?

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

Requires lots of code to do simple tasks.

Hard to understand other people's code

Extremely unportable

C++ History: Moving Forward

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:

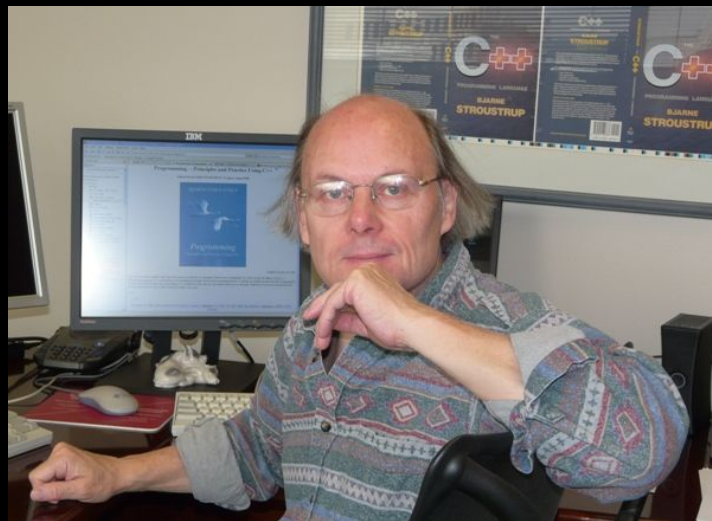
- No **objects** or **classes** (think cmap/cvec)
- Difficult to write code that worked **generically**
- Tedious when writing **large** programs

C++ History: Bjarne Stroustrup

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

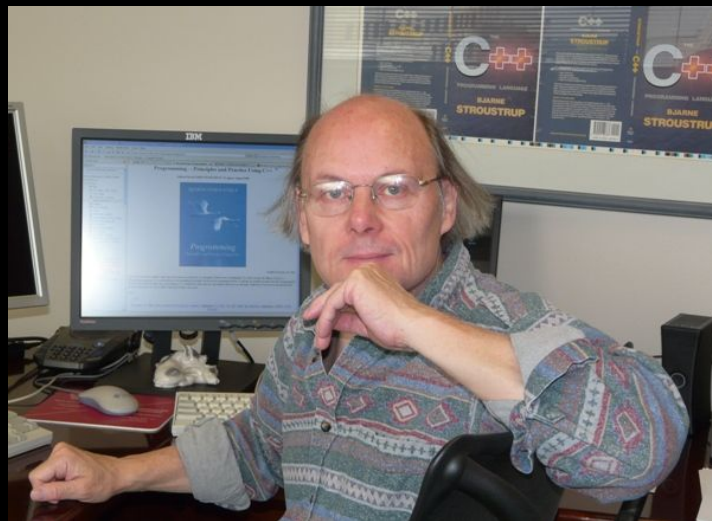


C++ History: Bjarne Stroustrup

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

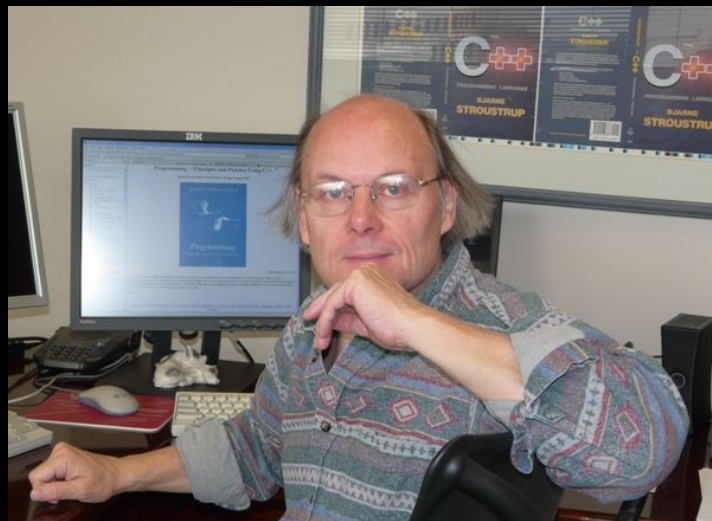


C++ History: Bjarne Stroustrup

In 1983, the first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

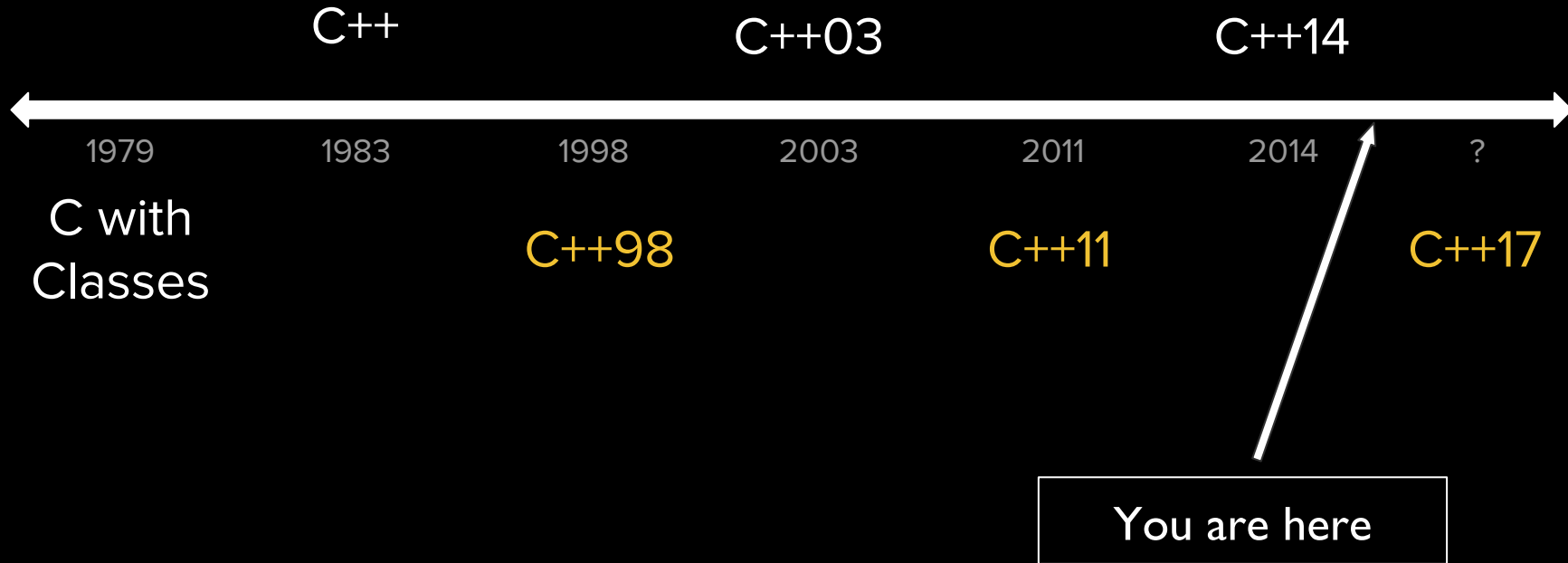
- Fast
- Simple to Use
- Cross-platform
- Had high level features



C++ History: Evolution of C++



C++ History: Evolution of C++



The C++ Philosophy

Only add features if they solve an actual problem

Programmers should be free to choose their own style

Compartmentalization is key

Allow the programmer full control if they want it

Don't sacrifice performance except as a last resort

Enforce safety at compile time whenever possible

The C++ Philosophy

Only add features if they solve an actual problem

Programmers should be free to choose their own style

Compartmentalization is key

Allow the programmer full control if they want it

Don't sacrifice performance except as a last resort

Enforce safety at compile time whenever possible

The C++ Philosophy

Only add features if they solve an actual problem

Programmers should be free to choose their own style

Compartmentalization is key

Allow the programmer full control if they want it

Don't sacrifice performance except as a last resort

Enforce safety at compile time whenever possible

The C++ Philosophy

Only add features if they solve an actual problem

Programmers should be free to choose their own style

Compartmentalization is key

Allow the programmer full control if they want it

Don't sacrifice performance except as a last resort

Enforce safety at compile time whenever possible

Our first C++ Program

hello.cpp

```
#include <iostream>

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

Next Time

Streams

