

Course goals

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

Course goals

Write complex, correct programs in C and C++

Leverage programming ecosystem

- Linux
- Compiler (gcc, g++)
- Debugger (gdb)
- Build system (make)

Learn basic principles of software design and engineering

Prerequisites

Experience with Java, Python or similar

C/C++ experience *not* a prerequisite

Learn C

- Language features
- Pointers & dynamic memory allocation
- “Low-level” programming

Learn C++

- How is it different from C?
- Object-oriented programming
- Generic programming

Gain proficiency in Linux & related programming tools

- Basic command-line tools
- Compilers, debuggers, profilers

Grow as a programmer & software engineer

Why C/C++?

Why C/C++?

- Ubiquitous
- Efficient
- Mature

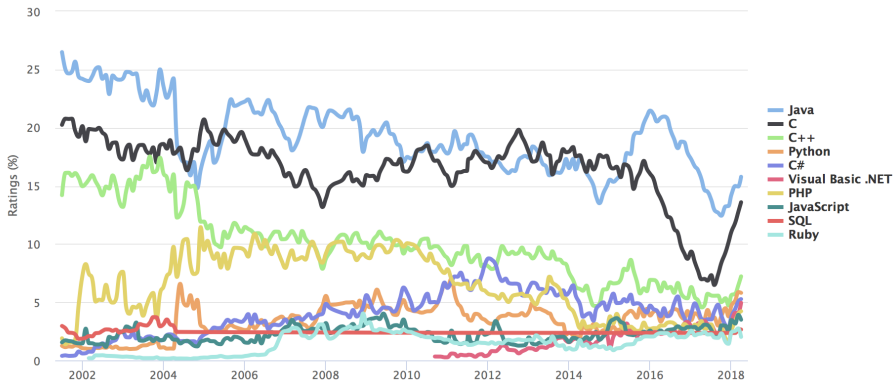
Much of the world's crucial software is in C

- Used Java? JVM is written in C++, as are many libraries
- Used Python? CPython interpreter written in C
- Used the Internet? Network stacks, routers, web servers, ...
- Like science?
 - <https://github.com/collections/software-in-science>
 - My lab members & I program in C/C++ a lot

Ubiquitous

TIOBE Programming Community Index

Source: www.tiobe.com



www.tiobe.com/tiobe-index/

Based on search engine hits for "<language> programming"

Higher-level languages like Java & Python present a trade-off:

- High-level languages “take care of things” for you
 - Source code is more concise, abstract
 - Harder to make mistakes
- ... but also hide things from you
 - How variables are laid out in memory
 - When memory is allocated and de-allocated
 - Hardware features, especially non-portable features

Around since the 1970s (C) and 80s (C++)

Undergraduates have learned it for decades

Software jobs often require it; "we need someone who. . .

- . . . can make something really fast if needed"
- . . . knows how to program all kinds of weird hardware"
- . . . knows how to interact with the operating system"
- . . . can handle our large codebase, written in C"

Why C/C++?

Newer “systems languages” aim for a similar level of efficiency as C/C++

- But with simpler language
- Less “burdened” by long history & by need to stay backward-compatible

Examples

- Swift – developer.apple.com/swift
- Go – golang.org
- Rust – rust-lang.org

Why C/C++?

On efficiency, they approach C/C++

But they do not approach C/C++ in maturity/ubiquity

- Some, like Swift, are associated with (& tied to) particular companies

Why C/C++?

Apr 2018	Apr 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.777%	+0.21%
2	2		C	13.589%	+6.62%
3	3		C++	7.218%	+2.66%
4	5	▲	Python	5.803%	+2.35%
5	4	▼	C#	5.265%	+1.69%
6	7	▲	Visual Basic .NET	4.947%	+1.70%
7	6	▼	PHP	4.218%	+0.84%
8	8		JavaScript	3.492%	+0.64%
9	-	▲	SQL	2.650%	+2.65%
10	11	▲	Ruby	2.018%	-0.29%
11	9	▼	Delphi/Object Pascal	1.961%	-0.86%
12	15	▲	R	1.806%	-0.33%
13	16	▲	Visual Basic	1.798%	-0.26%
14	13	▼	Assembly language	1.655%	-0.51%
15	12	▼	Swift	1.534%	-0.75%
16	10	▼	Perl	1.527%	-0.89%
17	17		MATLAB	1.457%	-0.59%
18	14	▼	Objective-C	1.250%	-0.91%
19	18	▼	Go	1.180%	-0.79%
20	20		PL/SQL	1.173%	-0.45%