

Modern C++ Programming

1. INTRODUCTION

Federico Busato

University of Verona, Dept. of Computer Science
2018, v1.5



About Programming

“And programming computers was so fascinating. You create your own little universe, and then it does what you tell it to do”

Vint Cerf (TCP/IP co-inventor and Turing Award)

“Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program”

Linus Torvalds (principal developer of the Linux kernel)

“You might not think that programmers are artists, but programming is an extremely creative profession. It’s logic-based creativity”

John Romero (co-founder of id Software)

A Little History of C/C++ Programming Language

The Assembly Programming Language



A long time ago, in a galaxy far,
far away...there was **Assembly**

- Extremely simple instructions
- Requires lots of code to do simple tasks
- Can express anything your computer can do
- Hard to read, write
- ...redundant, boring programming, bugs proliferation

```
main:
.Lfunc_begin0:
    push rbp
.Lcfi0:
.Lcfi1:
    mov rbp, rsp
.Lcfi2:
    sub rsp, 16
    movabs rdi, .L.str
.Ltmp0:
    mov al, 0
    call printf
    xor ecx, ecx
    mov dword ptr [rbp - 4], eax
    mov eax, ecx
    add rsp, 16
    pop rbp
    ret
.Ltmp1:
.Lfunc_end0:
.L.str:
.asciz "Hello World\n"
```

In the 1969 **Dennis M. Ritchie** and **Ken Thompson** (AT&T, Bell Labs) worked on developing a operating system for a large computer that could be used by a thousand users. The new operating system was called **UNIX**.

The whole system was still written in assembly code. Besides assembler and Fortran, UNIX also had an interpreter for the **programming language B**. A high-level language like B made it possible to write many pages of code task in just a few lines of code. In this way the code could be produced much faster then in assembly.

A drawback of the B language was that it did not know data-types. (Everything was expressed in machine words). Another functionality that the B language did not provide was the use of “structures”. The lag of these things formed the reason for Dennis M. Ritchie to develop the **programming language C**. In 1988 they delivered the final standard definition ANSI C.



Dennis M. Ritchie, and Ken Thompson

```
#include "stdio.h"  
  
int main() {  
    printf("Hello World\n");  
}
```

Areas of Application:

- UNIX operating system
- Computer games
- Due to their power and ease of use, C were used in the programming of the special effects for Star Wars



Star Wars - The Empire Strikes Back

The **C++ programming language** (originally named “C with Classes”) was devised by **Bjarne Stroustrup** also an employee from Bell Labs (AT&T). Stroustrup started working on C with Classes in 1979. (The ++ is C language operator)

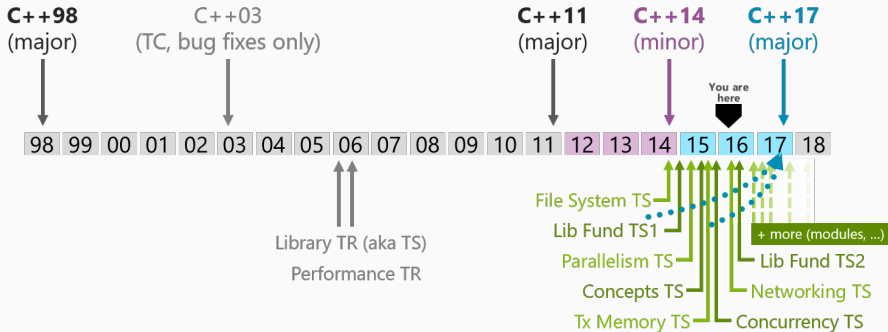
The first commercial release of the C++ language was in October of 1985



Bjarne Stroustrup

Areas of Application

- Operating systems: Windows, Android, OS X, Linux
- Image Editing Application: Adobe Premier, Photoshop and Illustrator
- Web browser: Firefox, Chrome, etc.
- High-Performance Computing (HPC)
- Embedded systems
- Multimedia
- Scientific applications: Machine Learning, Data analysis at CERN/NASA, SETI@home
- Google also use C++ for Indexing
- Database: MySQL
- Compilers: LLVM

























Modern C++ Evolution

C++

- 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
- Do not sacrifice **performance** except as a last resort
- Enforce **safety at compile time** whenever possible

Programming Languages Ranking (IEEE Spectrum - 2018)

Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	98.4
3. C	  	98.2
4. Java	  	97.5
5. C#	  	89.8
6. PHP		85.4
7. R		83.3
8. JavaScript	 	82.8
9. Go	 	76.7
10. Assembly		74.5

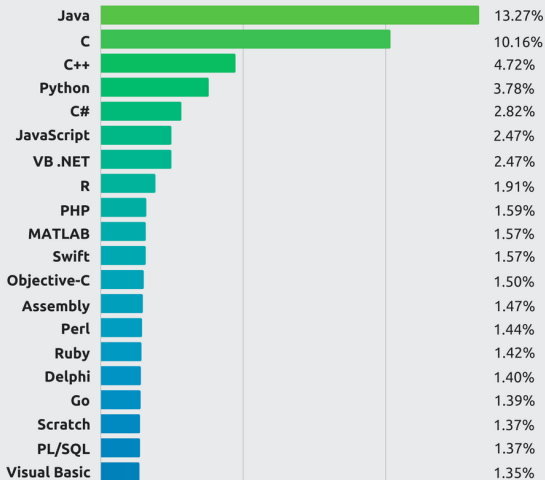
Link: [interactive-the-top-programming-languages-2018](#)

Source: Google Search/Threads, Twitter, Github, Stack Overflow, IEEE Xplore Digital Library

Most Popular Programming Languages (TIOBE - 2018)

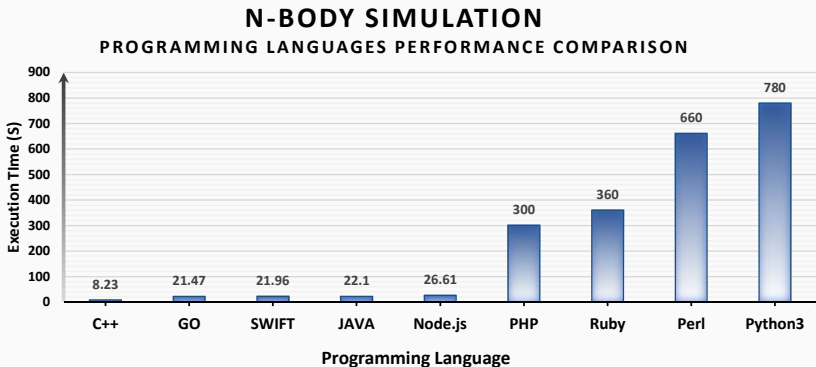
Top Programming Languages

Tiobe Index - December 2017



Why C++ is so popular?

- **Extreme performance** (theoretically enables highest performance)
- **Relatively easy to prove and test a C++ program**
- **Many support tools:** coverage, analysis, profiling, etc.
- **Low-level code:** drivers, kernels, etc.



Why C++ is so difficult?

- C++ is the hardest language for students to master
 - Huge set of features
 - Worry about memory management
 - Learn meta-programming
 - Distinguish compile-time from run-time
 - Low-level implementation details: pointer arithmetics, structure padding, etc.
-

“C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off”

Bjarne Stroustrup

“The problem with using C++...is that there's already a strong tendency in the language to require you to know everything before you can do anything”

Larry Wall (developer of the Perl language)

References

Unofficial C++ references:

en.cppreference.com

www.cplusplus.com/reference

IBM Knowledge Center

Tutorials:

www.learncpp.com

www.tutorialspoint.com/cplusplus

en.wikibooks.org/wiki/C++

yet another insignificant...programming notes

Other resources:

isocpp.org/faq

stackoverflow.com

The Course

The Course

The primary goal of the course is to guide the student, who has previous experience with C and object-oriented features, to a proficiency level of C++ programming

Organization:

- 12 lectures
- More than 600 slides

Roadmap:

- Review basic C concepts in C++ (built-in types, memory management, preprocessing, etc.)
- Introduce object-oriented and template concepts
- Present how to organize the code and the main conventions
- C++ related tools usage (debugger, static analysis, etc.)

What is/What is not

What the course **is not**:

- A theoretical course on programming
- A high-level concept description

What the course **is**:

- A “quite” advanced C++ programming language course
- A very practical course
- Do not focus on the concepts behind something but on what is the best way to use it
- Prefer examples instead long descriptions
- Present many language features

Prerequisites:

- Knowledges of C programming language
- Knowledges of object-oriented programming

Federico Busato, Ph.D.



- **Research interests:** Parallel/High-Performance Computing, Graph Theory, and Linear Algebra
- **Current Experience:** Senior Software Engineer at Nvidia (California, USA) | CUDA Mathematical Libraries
- **Courses:** Advanced Architectures (Master degree), Operating System (Bachelor degree)

Alessandro Danese, Ph.D.



- **Research interests:** Embedded System Verification
- **Previous Experience:** Software Engineer at Intel (Portland, Oregon, USA)
- **Courses:** Design Automation of Embedded Systems (Master degree), Operating System (Bachelor degree)

My rule at Nvidia: lead the cuSPARSE library (+ recruiting)

<https://docs.nvidia.com/cuda/cusparses/index.html>

The cuSPARSE library contains a set of basic linear algebra subroutines used for handling sparse matrices (matrix-matrix multiplication, triangular solver, etc.) on GPU devices

cuSPARSE is part of the CUDA Toolkit (8M downloads in 2018)

cuSPARSE users:

- Industrial (Google, Facebook, Microsoft, etc.)
- Academic (student/researchers/national laboratories)

cuSPARSE applications:

- High-performance numerical solver
- Physic/Simulation/EDA/CAD software
- Computer Graphics
- (recently) AI/Deep learning

The library:

- More than 300,000 lines of code
- Must provide high performance
- Must work correctly on main 32/64-bit OS (Windows, Android, Linux, Mac, etc.)
- Must work correctly on main CPU architectures (Intel, AMD, ARM, IBM, etc.)
- Must work correctly on all GPU architectures
- Comprises host (C/C++), device code (CUDA, C++ extension), and assembly code
- Support half-precision floating point, complex numbers, etc.

“The only way to learn a new programming language is by writing programs in it”

Dennis Ritchie