



A linguagem de programação C++ e por que você deveria aprendê-la

ISO/IEC 14882:2020
(C++ 20)

Fabio Galuppo, M.Sc.

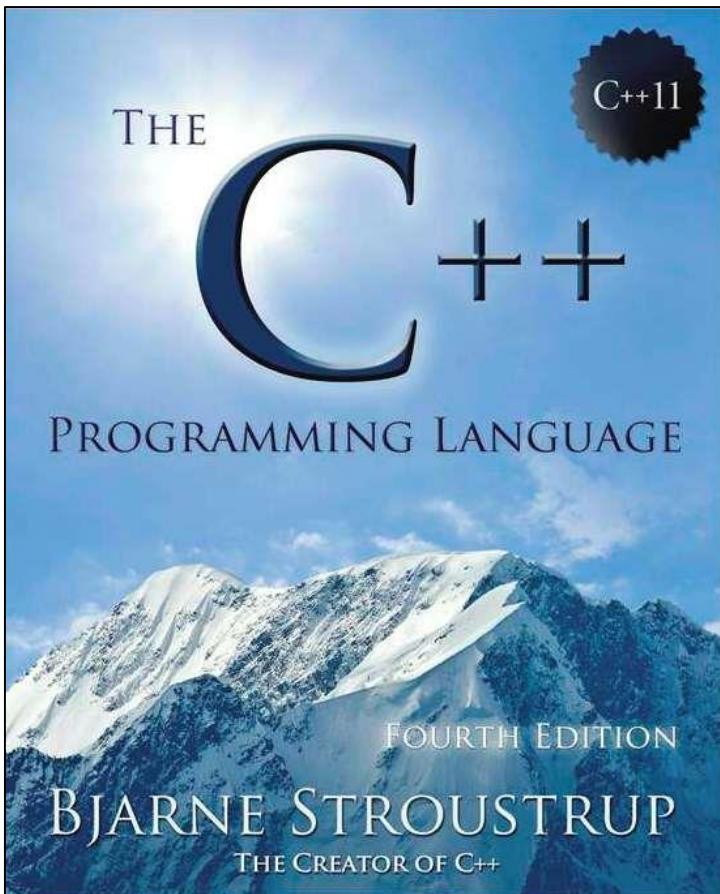
fabiogaluppo@acm.org

<https://bit.ly/AprenderCpp2024>

INTRODUÇÃO

C++

- É uma linguagem de programação
 - +40 anos (C com Classes em 1979)



NÚMEROS

TIOBE Index for March 2024

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular web sites Google, Amazon, Wikipedia, Bing and more than 20 others are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found [here](#).

Mar 2024	Mar 2023	Change	Programming Language	Ratings	Change
1	1		 Python	15.63%	+0.80%
2	2		 C	11.17%	-3.56%
3	4		 C++	10.70%	-2.59%
4	3		 Java	8.95%	-4.61%
5	5		 C#	7.54%	+0.37%
6	7		 JavaScript	3.38%	+1.21%
7	8		 SQL	1.92%	-0.04%
8	10		 Go	1.56%	+0.32%
9	14		 Scratch	1.46%	+0.45%
10	6		 Visual Basic	1.42%	-3.33%

<https://www.tiobe.com/tiobe-index/>

TIOBE Index for August 2017

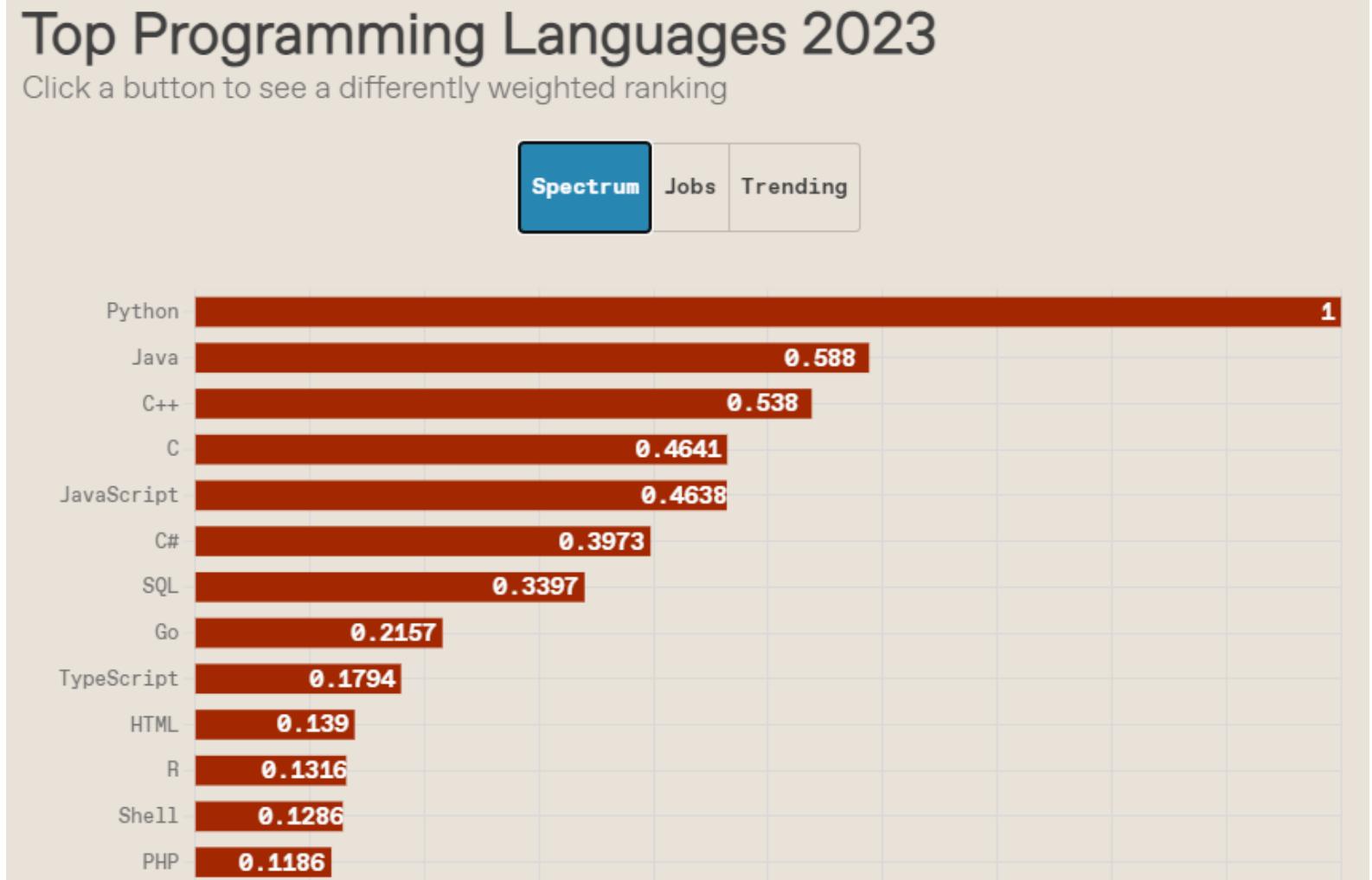
The top programming languages are in a long term decline: both Java and C have all time low scores in the TIOBE index. And almost all of the other top 10 languages are going down as well year to year. So what languages are taking advantage of this? It is all happening down in the charts around position 40. A new set of languages is gaining ground, notably Crystal (#32), Kotlin (#41), Clojure (#42), Hack (#43) and Julia (#46). Especially Crystal with its jump from position 60 to 32 in one month is doing very well. The Crystal programming language is a statically typed Ruby variant. Since it is compiled it is superfast and has a small memory footprint without losing the feeling of being easy to use. It seems worthwhile to give it a try.

Aug 2017	Aug 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.961%	-6.05%
2	2		C	6.477%	-4.83%
3	3		C++	5.550%	-0.25%
4	4		C#	4.195%	-0.71%
5	5		Python	3.692%	-0.71%
6	8	▲	Visual Basic .NET	2.569%	+0.05%
7	6	▼	PHP	2.293%	-0.88%
8	7	▼	JavaScript	2.098%	-0.61%
9	9		Perl	1.995%	-0.52%
10	12	▲	Ruby	1.965%	-0.31%
11	14	▲	Swift	1.825%	-0.16%
12	11	▼	Delphi/Object Pascal	1.825%	-0.45%
13	13		Visual Basic	1.809%	-0.24%
14	10	▼	Assembly language	1.805%	-0.56%
15	17	▲	R	1.766%	+0.16%
16	20	▲	Go	1.645%	+0.37%

<https://www.tiobe.com/tiobe-index/>

IEEE Spectrum The 2023 Top Programming Languages

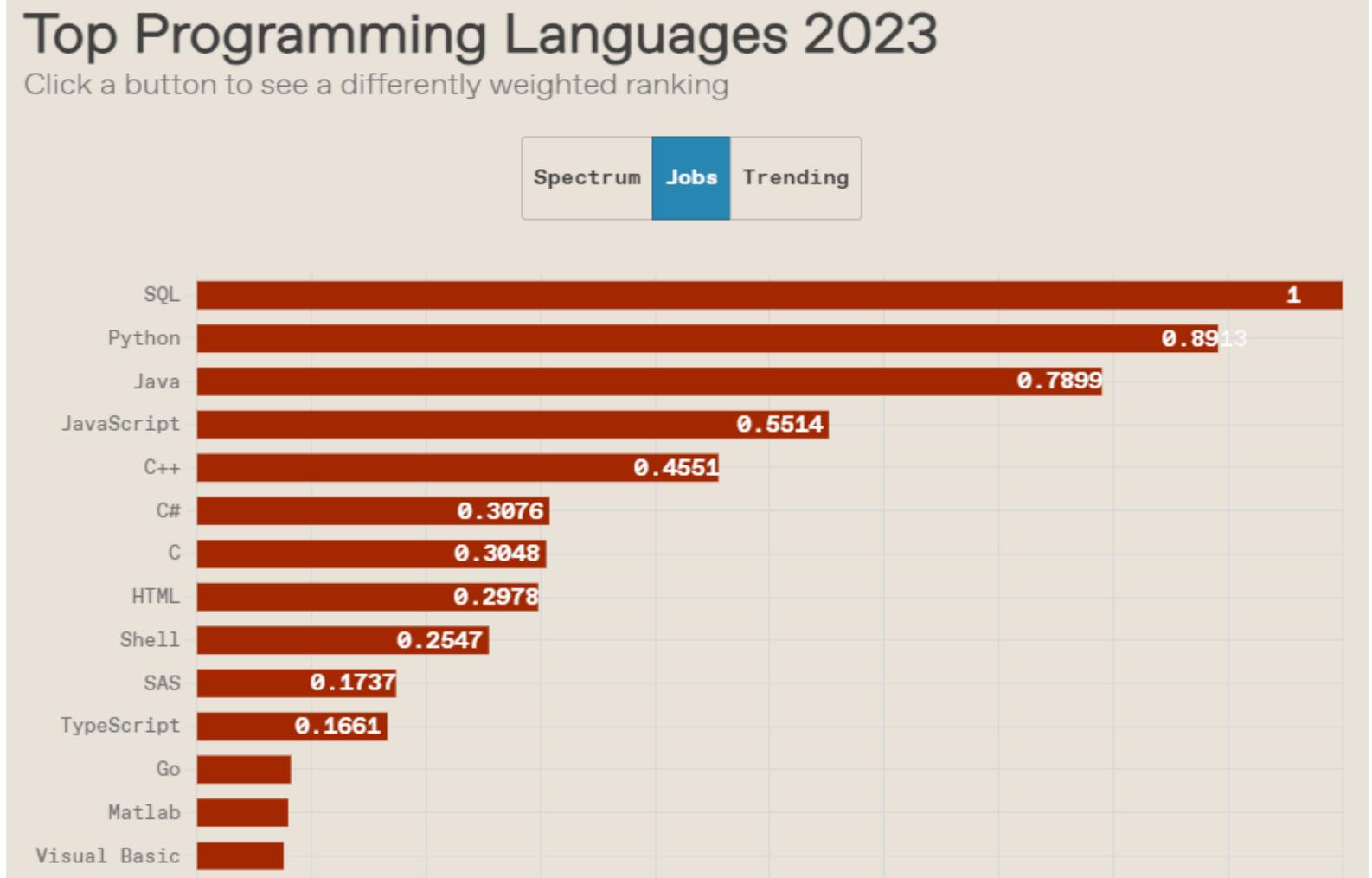
Python and SQL are on top, but old languages shouldn't be forgotten



<https://spectrum.ieee.org/the-top-programming-languages-2023>

IEEE Spectrum The 2023 Top Programming Languages

Python and SQL are on top, but old languages shouldn't be forgotten

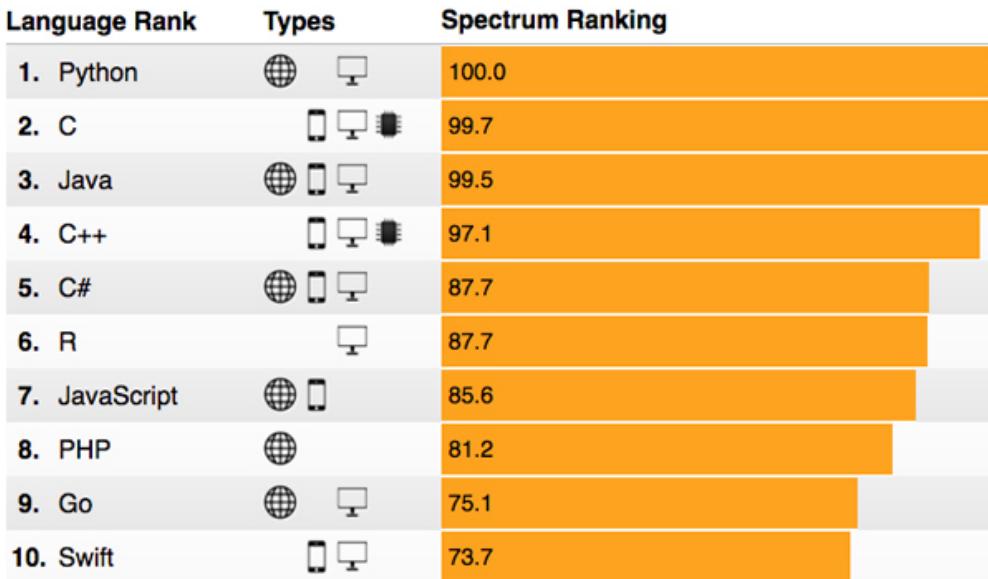


<https://spectrum.ieee.org/the-top-programming-languages-2023>

IEEE Spectrum The 2017 Top Programming Languages

Python jumps to No. 1, and Swift enters the Top Ten

So what are the Top Ten Languages for the typical *Spectrum* reader?



Python has continued its upward trajectory from last year and jumped two places to the No. 1 slot, though the top four—Python, C, Java, and C++—all remain very close in popularity. Indeed, in Diakopoulos's analysis of what the underlying metrics have to say about the languages currently in demand by recruiting companies, C comes out ahead of Python by a good margin.

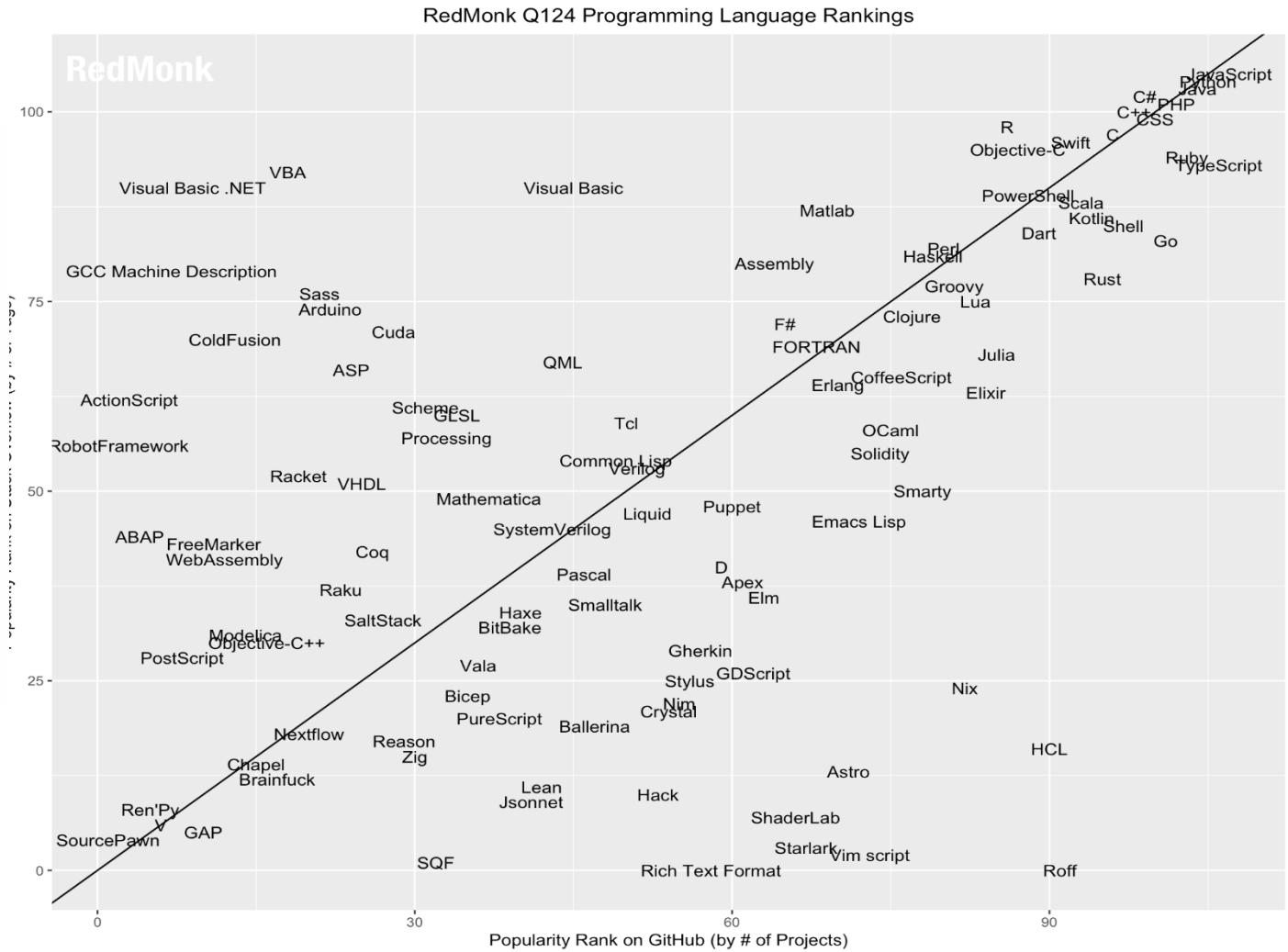
C# has reentered the top five, taking back the place it lost to R last year. Ruby has fallen all the way down to 12th position, but in doing so it has given Apple's Swift the chance to join Google's Go in the Top Ten. This is impressive, as Swift debuted on the rankings just two years ago. (Outside the

<http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>

The RedMonk Programming Language Rankings

January 2024

- 1 JavaScript
- 2 Python
- 3 Java
- 4 PHP
- 5 C#
- 6 TypeScript
- 6 CSS
- 8 C++
- 9 Ruby
- 10 C

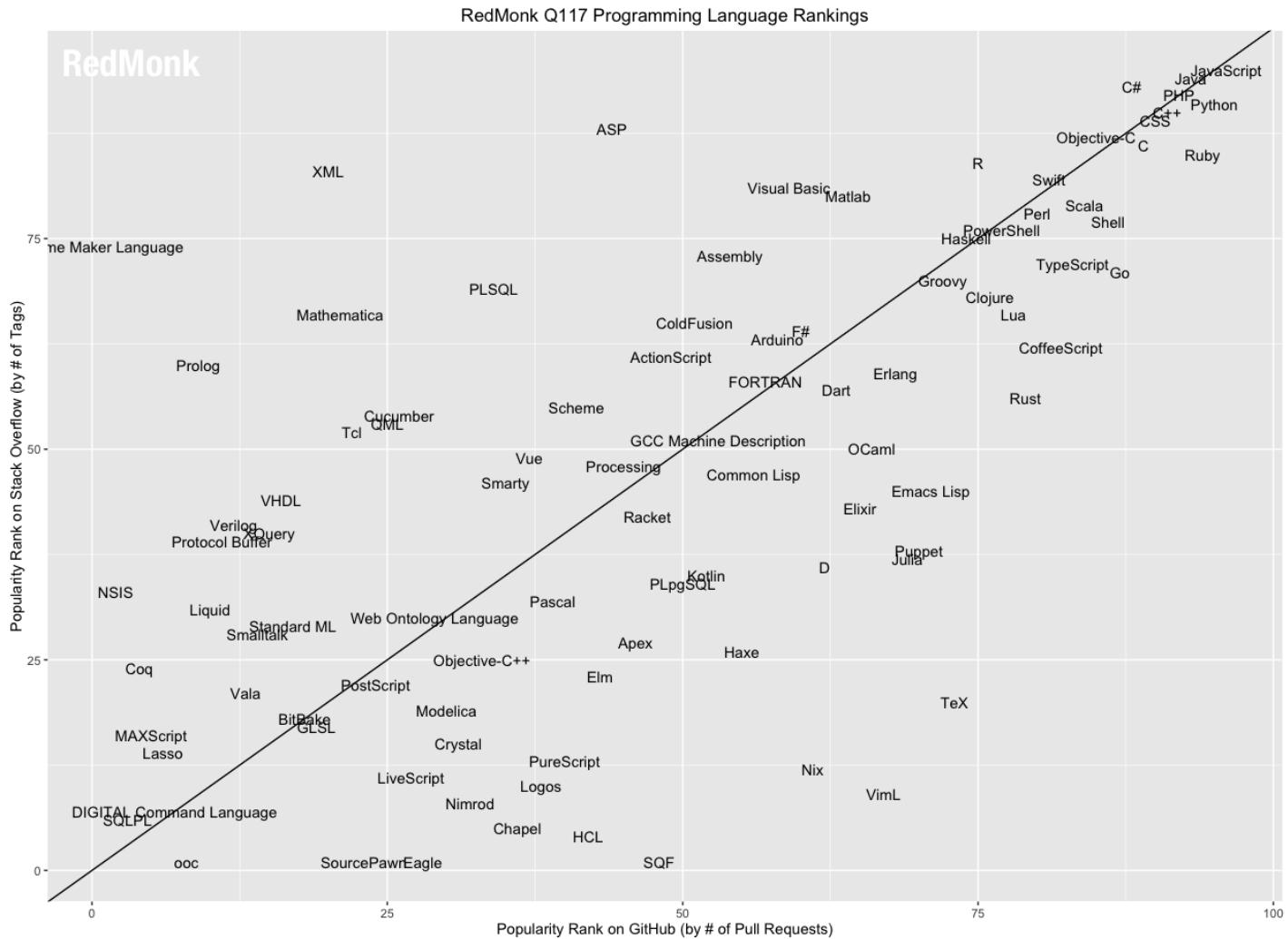


<https://redmonk.com/sogrady/2024/03/08/language-rankings-1-24/>

The RedMonk Programming Language Rankings

January 2017

- 1 JavaScript
- 2 Java
- 3 Python
- 4 PHP
- 5 C#
- 5 C++
- 7 CSS
- 7 Ruby
- 9 C
- 10 Objective-C



<http://redmonk.com/sogrady/2017/03/17/language-rankings-1-17/>

The Computer Language Benchmarks Game

C++ g++ programs versus C gcc

n-body

source	secs	mem	gz	cpu	secs
<u>C gcc #9</u>	2.08	11,392	1633		2.08
<u>C++ g++ #0</u>	2.14	10,960	1927		2.14

fasta

source	secs	mem	gz	cpu	secs
<u>C gcc #9</u>	0.78	11,284	1463		0.78
<u>C gcc #5</u>	1.30	11,392	1281		1.30
<u>C++ g++ #8</u>	0.77	10,944	2751		1.52

mandelbrot

source	secs	mem	gz	cpu	secs
<u>C++ g++</u>	0.89	34,012	1791		3.46
<u>C++ g++ #4</u>	0.88	34,664	3542		3.48
<u>C++ g++ #6</u>	0.98	34,768	1002		3.88
<u>C gcc #6</u>	1.30	32,992	1135		5.17

reverse-complement

source	secs	mem	gz	cpu	secs
<u>C gcc #7</u>	0.43	499,200	1895		0.62
<u>C++ g++ #2</u>	0.68	499,568	2093		0.68

binary-trees

source	secs	mem	gz	cpu	secs
<u>C++ g++ #7</u>	0.94	201,600	890		3.31
<u>C++ g++ #5</u>	1.13	200,428	885		4.05
<u>C++ g++ #9</u>	1.45	135,832	809		4.10
<u>C gcc #2</u>	1.57	168,448	809		4.11

The Computer Language Benchmarks Game

C++ g++ programs versus Java and Rust

n-body	secs	mem	gz	cpu secs
C++	2.14	10,960	1927	2.14
Rust	2.81	11,068	1874	2.81
Java	6.77	40,632	1429	6.82
mandelbrot				
C++	0.89	34,012	1791	3.46
Rust	1.01	32,380	1295	3.99
Java	4.39	101,236	660	16.12
pidigits				
Rust	0.71	11,068	799	0.71
C++	0.87	10,956	798	0.87
Java	0.92	38,380	764	0.95
fasta				
C++	0.77	10,944	2751	1.52
Rust	0.76	11,068	2529	1.52
Java	3.17	40,400	1524	3.27
reverse-complement				
C++	0.68	499,568	2093	0.68
Rust	0.47	498,844	3040	0.77
Java	3.02	2,027,152	752	3.69
binary-trees				
C++	0.94	201,600	890	3.31
Rust	1.08	198,984	765	3.82
Java	4.47	2,116,292	840	5.15

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/gpp-java.html>

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/gpp-rust.html>

Database of Databases

Implementation	
C++	173
Java	146
C	114
Go	86
Rust	53
Python	41
C#	33
JavaScript	28
Erlang	13
Scala	12



O QUE É C++?

O que é C++?

Smarter computing
Texas A&M University

What is C++?

Template
meta-programming!

Buffer
overflows

Classes

Too big!

An object-oriented
programming language

A hybrid language

Class hierarchies

A multi-paradigm
programming language

It's C!

Embedded systems
programming language

Low level!

Generic programming

A random collection
of features



Stroustrup - ACCU'13

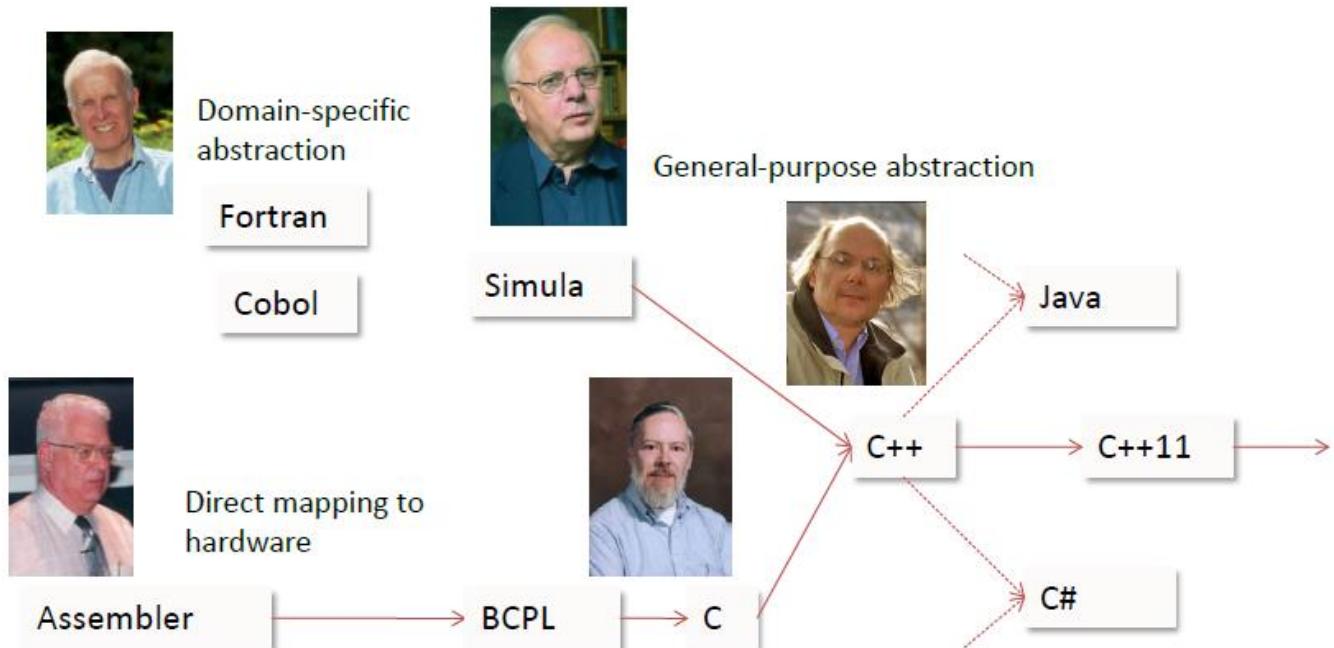
The C++ Programming Language

C++ is a general purpose programming language with a bias towards systems programming that

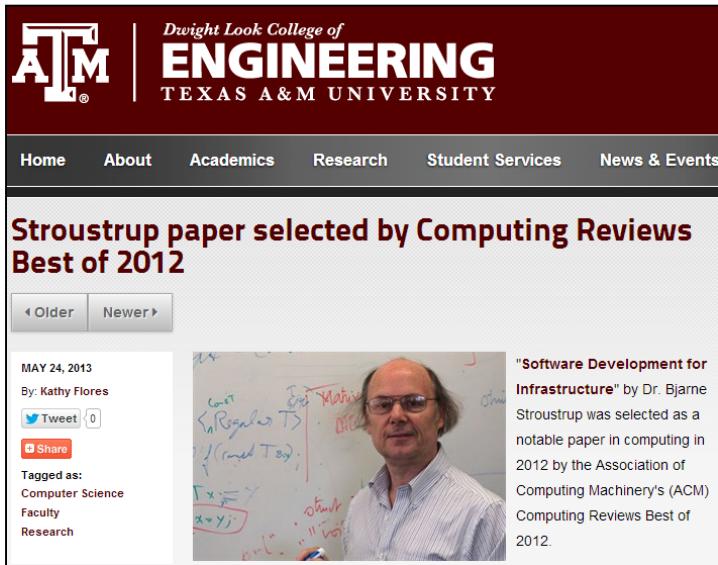
- is [a better C](#)
- supports [data abstraction](#)
- supports [object-oriented programming](#)
- supports [generic programming](#).

“C++ is a general purpose programming language designed to make programming more enjoyable for the serious programmer.”

<https://www.stroustrup.com/quotes.html>



C++ System Programming



The screenshot shows the Texas A&M University website for the Dwight Look College of Engineering. The header features the college's logo and name. Below the header, a news article is displayed about a paper selected by Computing Reviews Best of 2012. The article includes a photo of Bjarne Stroustrup, the author of the paper, standing next to a whiteboard with handwritten code.

Stroustrup paper selected by Computing Reviews Best of 2012

By Kathy Flores

May 24, 2013

Tweet 0 Share

Tagged as: Computer Science Faculty Research

"Software Development for Infrastructure" by Dr. Bjarne Stroustrup was selected as a notable paper in computing in 2012 by the Association of Computing Machinery's (ACM) Computing Reviews Best of 2012.



The image shows the cover of IEEE Computer Magazine, Volume 45, Issue 1. The cover features a large pipe with a sunset scene visible through its opening, symbolizing infrastructure. The title of the cover feature is "Software Development for Infrastructure" by Bjarne Stroustrup. The sidebar on the right lists awards: "Computing Reviews Best of 2012" and "Notable Paper". The ACM ThinkLoud logo is also present.

COVER FEATURE

Software Development for Infrastructure

Bjarne Stroustrup, Texas A&M University

Infrastructure software needs more stringent correctness, reliability, efficiency, and maintainability requirements than non-essential applications. This implies greater emphasis on up-front design, static structure enforced by a type system, compact data structures, simplified code structure, and improved tool support. Education for infrastructure and application developers should differ to reflect that emphasis.

The implication puts a new emphasis on the old challenge to software developers: deliver code that is both correct and efficient. The benefits achievable with better system design and implementation are huge. If we can double the efficiency of server software, we can make do with one server farm instead of two (or more). If we can double the efficiency of critical software in a smartphone, its battery life almost doubles. If we halve the bug count in safety-critical software, we can naively expect only half as many people to sustain injuries.

This view contrasts with the common opinion that human effort is the factor to optimize in system development.

IEEE Computer Magazine Volume:45, Issue: 1
<http://dx.doi.org/10.1109/MC.2011.353>

“A light-weight abstraction programming language

Key strengths:

- **software infrastructure**
- resource-constrained applications”

C++ 11
The Future is here

-- <http://www.infoq.com/presentations/Cplusplus-11-Bjarne-Stroustrup>

O início do C++

- 1979 (C com Classes)
- 1985 (Primeira versão comercial)

1979 April

Work on C with Classes began

1979 October

First C with Classes (Cpre) running

1983 August

First C++ in use at Bell Labs

1984

C++ named

1985 February

Cfront Release E (first external C++ release)

1985 October

Cfront Release 1.0 (first commercial release)

The C++ Programming Language

The screenshot shows a web browser window with two tabs open. The active tab is titled 'C++ Historical Sources Archive — So...' and the URL is www.softwarepreservation.org/projects/c_plus_plus. The second tab is titled 'Cfront - Wikipedia, the free encyclopedia'.

The main content area displays the 'C++ Historical Sources Archive' page. It includes a sidebar navigation menu with links to Home, Projects, ALGOL, C++, Applications, Cfront releases, Libraries, Papers and articles, and FORTRAN and. The 'C++' link is highlighted. The main content area features an 'Abstract' section describing the archive as a collection of design documents, source code, and other materials concerning the birth, development, standardization, and use of the C++ programming language. It also lists 'Contents' such as Chronology and Cfront releases. At the bottom of the page, there is a link to <http://www.mcjones.org/dustydecks/>.

O caminho para padronização do C++

- 1990 (ANSI C++ Standard – baseado no “ARM”)
- 1998 (1^a versão do padrão ISO C++)
<https://www.iso.org/standard/25845.html>
- 2003 (2^a versão do padrão ISO C++)
<https://www.iso.org/standard/38110.html>
- 2011 (3^a versão do padrão ISO C++)
<https://www.iso.org/standard/50372.html>
- 2014 (4^a versão do padrão ISO C++)
<https://www.iso.org/standard/64029.html>
- 2017 (5^a versão do padrão ISO C++)
<https://www.iso.org/standard/68564.html>
- 2020 (6^a versão do padrão ISO C++)
<https://www.iso.org/standard/79358.html>
- **2023 (7^a versão do padrão ISO C++)**
<https://www.iso.org/standard/83626.html> (em andamento)
<https://open-std.org/jtc1/sc22/wg21/docs/papers/2023/n4944.pdf> (*draft* atual)



ISO C++ (WG 21)



Recent milestones: C++23 done, out for final ballot; C++26 work has begun

C++ Padrão = Linguagem + Bibliotecas

C++ 23: <https://github.com/cplusplus/draft>

Document Number: N4944
Date: 2023-03-19
Revises: N4928
Reply to: Thomas Köppe
Google DeepMind
cxxeditor@gmail.com

Working Draft, Standard for Programming Language C++

acesso em: 29 de Março de 2024

Table 5: Keywords [tab:lex.key]

alignas	constinit	false	public	true
alignof	const_cast	float	register	try
asm	continue	for	reinterpret_cast	typedef
auto	co_await	friend	requires	typeid
bool	co_return	goto	return	typename
break	co_yield	if	short	union
case	decltype	inline	signed	unsigned
catch	default	int	sizeof	using
char	delete	long	static	virtual
char8_t	do	mutable	static_assert	void
char16_t	double	namespace	static_cast	volatile
char32_t	dynamic_cast	new	struct	wchar_t
class	else	noexcept	switch	while
concept	enum	nullptr	template	
const	explicit	operator	this	
constexpr	export	private	thread_local	
constexpr	extern	protected	throw	

C++ 20: ISO/IEC 14882:2020

This document specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements

and relaxations of the first requirement appear at various places within this document.

C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2018 *Programming languages — C* (hereinafter referred to as the *C standard*). C++ provides many facilities beyond those provided by C, including additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators,

and additional library facilities.

Table 23: Library categories [tab:library.categories]

Clause	Category
Clause 17	Language support library
Clause 18	Concepts library
Clause 19	Diagnostics library
Clause 20	Memory management library
Clause 21	Metaprogramming library
Clause 22	General utilities library
Clause 23	Strings library
Clause 24	Containers library
Clause 25	Iterators library
Clause 26	Ranges library
Clause 27	Algorithms library
Clause 28	Numerics library
Clause 29	Time library
Clause 30	Localization library
Clause 31	Input/output library
Clause 32	Regular expressions library
Clause 33	Concurrency support library

FERRAMENTAS

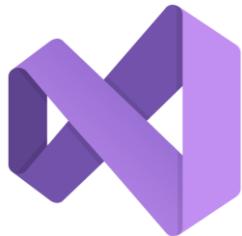
Compiladores



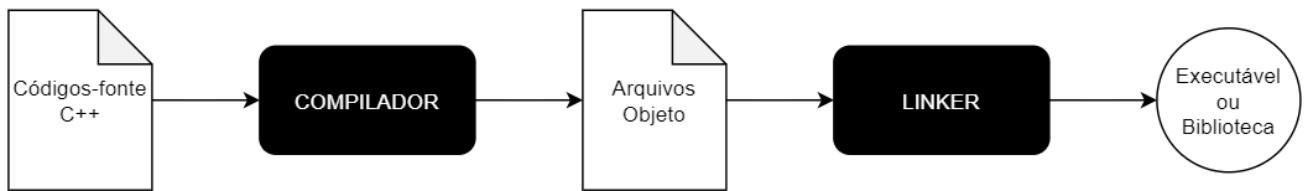
Clang



GCC



**Microsoft
Visual C++**



- GCC
- Clang
- MSVC
- Apple Clang
- EDG eccp
- Intel C++
- IBM XL C++
- IBM Open XL C++ for AIX
- IBM Open XL C++ for z/OS
- Sun/Oracle C++
- Embarcadero C++ Builder
- Cray
- Nvidia HPC C++ (ex PGI)*
- Nvidia nvcc

Compiladores



Clang



GCC



**Microsoft
Visual C++**



Microsoft STL

Standard library implementation

cl.exe

Creates object files (.obj)

lib.exe

Creates static libraries (.lib)

link.exe

Creates executables (.exe) and dynamic libraries (.dll)

Visual Studio Debugger

Attaches to processes and lets developer "step" through code

Compiladores



Clang



GCC



**Microsoft
Visual C++**



gcc / g++

Creates object files (.o)

ar

Creates static libraries (.a)

ld

Creates executables and shared libraries (.so)

gdb

Attaches to processes and lets developer "step" through code

Compiladores



Clang



GCC



**Microsoft
Visual C++**



libc++

Standard library implementation

**clang /
clang++**

Creates object
files (.o)

ar

Creates static
libraries (.a)

lld

Creates executables
and shared libraries
(.so)

lldb

Attaches to processes
and lets developer
"step" through code

Compiladores



Clang



GCC



Microsoft
Visual C++

Clang/LLVM
Ubuntu (via apt-get):
sudo apt-get install clang
macOS (via MacPorts - XCode e XCode Command Line Tools devem estar instalados):
sudo port install clang-17
Window (via MSYS2):
pacman -S mingw-w64-x86_64-clang

GCC (GNU Compiler Collection)
Ubuntu (via apt-get):
sudo apt-get install build-essential
macOS (via MacPorts - XCode e XCode Command Line Tools devem estar instalados):
sudo port install gcc
Window (via MSYS2):
pacman -S --needed base-devel mingw-w64- ucrt-x86_64-toolchain

Windows (3) —

<input checked="" type="checkbox"/> Universal Windows Platform development Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.
<input type="checkbox"/> .NET desktop development Build WPF, Windows Forms and console applications using the .NET Framework.
<input type="checkbox"/> Desktop development with C++ Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like MFC and...

(Via <https://visualstudio.microsoft.com/vs/community/>)

Compiler Explorer

The screenshot shows the Compiler Explorer interface. On the left, a C++ source code editor displays the following code:

```
#include <cstdint>

std::uint64_t pow(std::uint32_t base, std::uint16_t exponent) {
    std::uint64_t result = 1;
    while(exponent-- > 0) {
        result *= base;
    }
    return result;
}
```

On the right, an assembly output window shows the generated assembly code for the `x86-64 clang 18.1.0` compiler with optimization level `-O0`. The assembly code is as follows:

```
1  pow(unsigned int, unsigned short);                                # @pow(unsigned int, unsigned short)
2      push   rbp
3      mov    rbp, rsp
4      mov    ax, si
5      mov    dword ptr [rbp - 4], edi
6      mov    word ptr [rbp - 6], ax
7      mov    qword ptr [rbp - 16], 1
8 .LBB0_1:                                                               # =>This Inner Loop Header: Depth=1
9      mov    ax, word ptr [rbp - 6]
10     mov    cx, ax
11     add    cx, -1
12     mov    word ptr [rbp - 6], cx
13     movzx  eax, ax
14     cmp    eax, 0
15     jle    .LBB0_3
16     mov    eax, dword ptr [rbp - 4]
17     imul  rax, qword ptr [rbp - 16]
18     mov    qword ptr [rbp - 16], rax
19     jmp    .LBB0_1
20 .LBB0_3:
21     mov    rax, qword ptr [rbp - 16]
22     pop    rbp
23     ret
```

```
#include <cstdint>

std::uint64_t pow(std::uint32_t base, std::uint16_t exponent) {
    std::uint64_t result = 1;
    while(exponent-- > 0) {
        result *= base;
    }
    return result;
}
```

<https://godbolt.org/>

Quick C++ Benchmark

Quick C++ Benchmark

Run Quick Bench locally Support Quick Bench Suite ▾ More ▾

```
1 static void StringCreation(benchmark::State& state) {
2     // Code inside this loop is measured repeatedly
3     for (auto _ : state) {
4         std::string created_string("hello");
5         // Make sure the variable is not optimized away by compiler
6         benchmark::DoNotOptimize(created_string);
7     }
8 }
9 // Register the function as a benchmark
10 BENCHMARK(StringCreation);

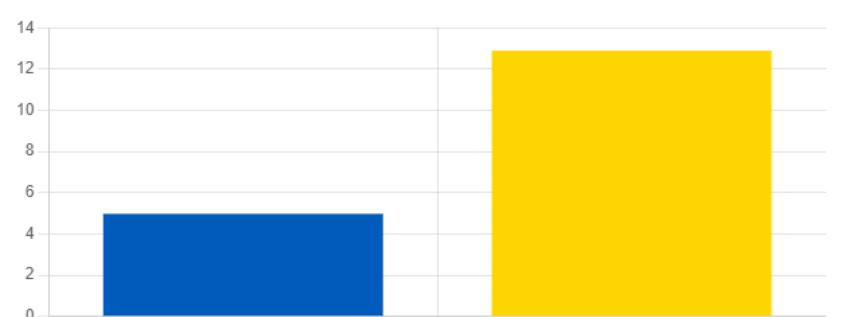
11 static void StringCopy(benchmark::State& state) {
12     // Code before the loop is not measured
13     std::string x = "hello";
14     for (auto _ : state) {
15         std::string copy(x);
16     }
17 }
18 BENCHMARK(StringCopy);
19
20
```

compiler = Clang 17.0 ▾ std = c++20 ▾ optim = O3 ▾
STL = libstdc++(GNU) ▾ Other flags ▾

Run Benchmark Disassembly = AT&T ▾ Clear cached results

Charts Assembly



Operation	ratio (CPU time / Noop time)
StringCreation	~5
StringCopy	~13

ratio (CPU time / Noop time)
Lower is faster

Show Noop bar

C++ Insights

The screenshot shows the C++ Insights interface. At the top, there are several icons: a magnifying glass, a play button, a save icon, a download icon, a star, a gear, and a person. To the right of these is the text "[C++ Standard: C++ 20]". Below the icons are two code panes: "Source:" and "Insight:". The "Source:" pane contains the following C++ code:

```
1 #include <ranges>
2 #include <algorithm>
3 #include <array>
4 #include <cstdio>
5
6 int main() {
7     std::array<int, 5> ints = { 81, 72, 63, 54, 45 };
8     std::ranges::sort(ints);
9     for (auto i : ints)
10        std::printf("%d\n", i);
11    return 0;
12 }
```

The "Insight:" pane shows the generated assembly-like code corresponding to the source code above:

```
1 #include <ranges>
2 #include <algorithm>
3 #include <array>
4 #include <cstdio>
5
6 int main()
7 {
8     std::array<int, 5> ints = {{81, 72, 63, 54, 45}};
9     std::ranges::sort.operator()(ints, {}, {});
10 {
11     std::array<int, 5> & __range1 = ints;
12     int * __begin1 = __range1.begin();
13     int * __end1 = __range1.end();
14     for(; __begin1 != __end1; ++__begin1) {
15         int i = *__begin1;
16         printf("%d\n", i);
17     }
18 }
19 }
20 return 0;
21 }
22 }
```

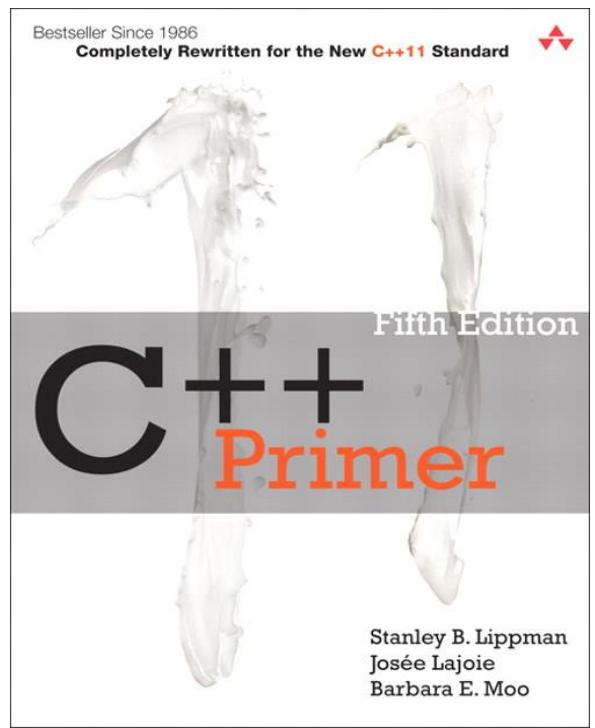
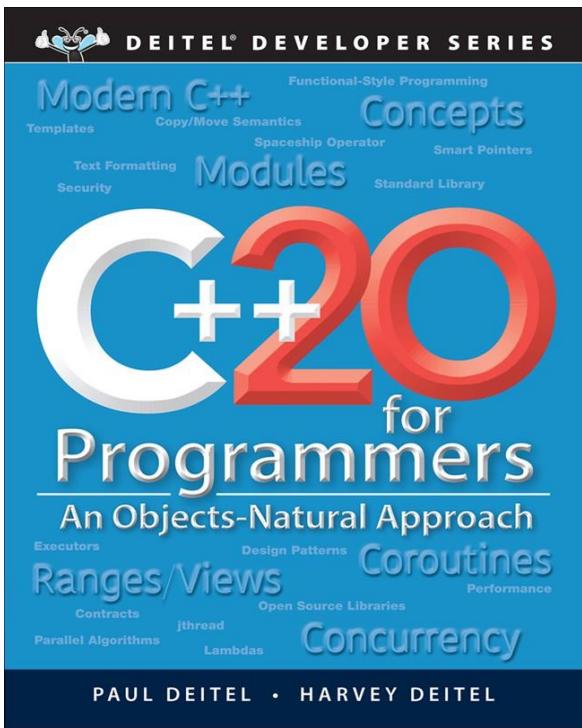
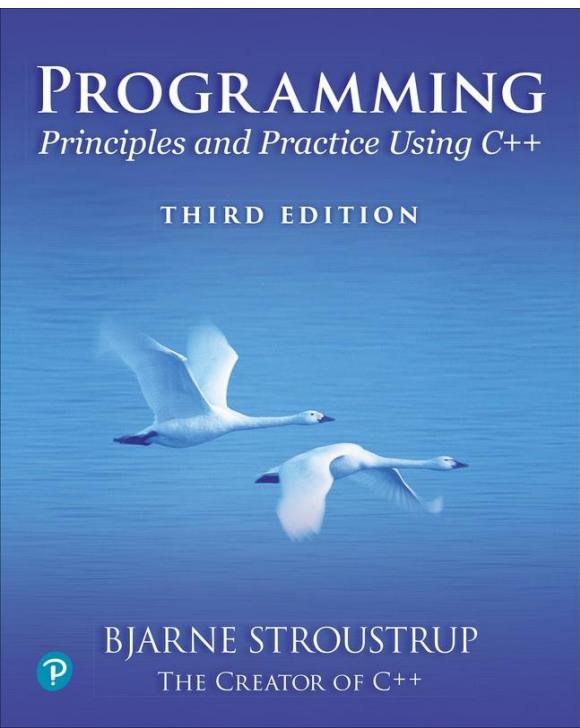
Outras ferramentas

- IDEs
 - Visual Studio, Visual Studio Code, Qt Creator, CLion, ...
- Build
 - CMake
 - make, msbuild, ninja
- Gerenciador de Pacotes
 - Vcpkg, Conan
- Análise Estática
 - Microsoft C/C++ Code Analysis, Clang Static Analyzer, clang-tidy, CppCheck, PVS-Studio
- Análise Dinâmica
 - Sanitizers
 - Address (ASan), Leak (LSan), Thread (TSan), UndefinedBehavior (UBSan)
 - <https://github.com/google/sanitizers/>

**EU QUERO APRENDER C++!
POR ONDE COMEÇAR?
O QUE PRECISO SABER?**

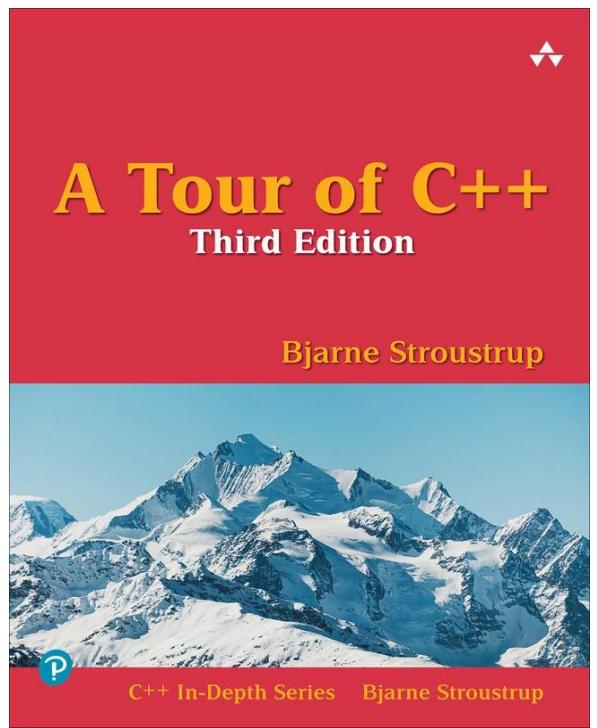
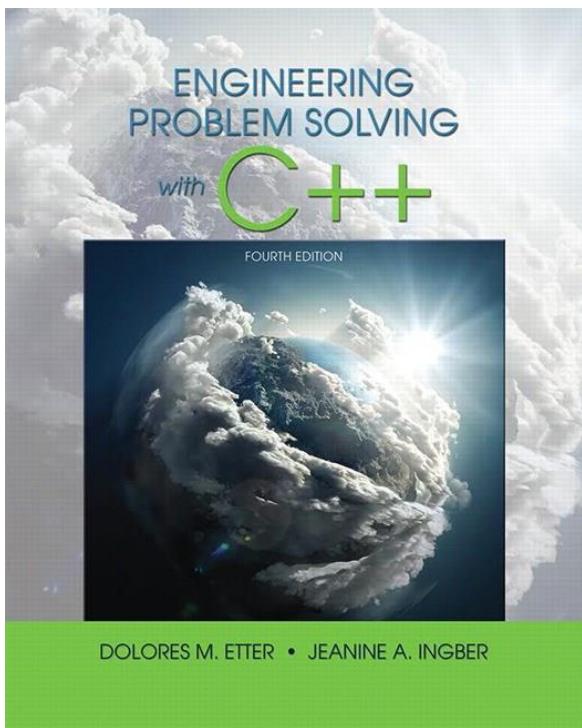
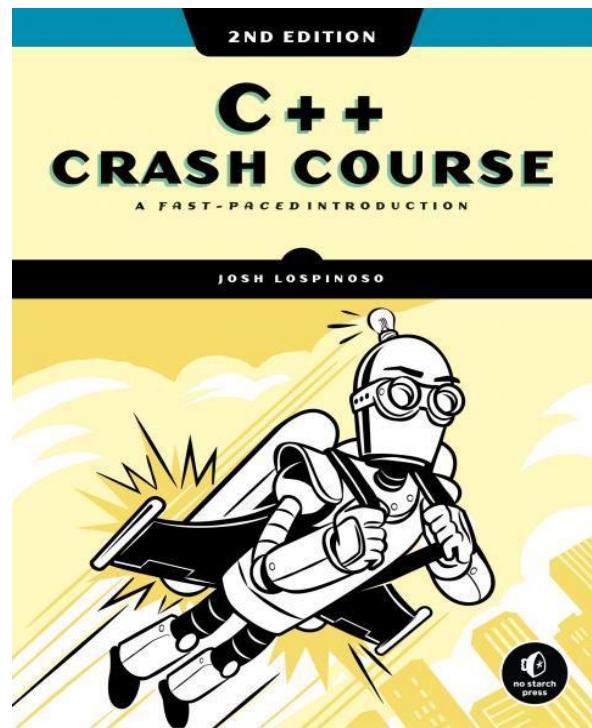
Eu quero aprender C++!

Por onde começar?



Eu quero aprender C++!

Por onde começar?



Eu quero aprender C++!

Por onde começar?

Updated for C++ 20

Discovering Modern C++

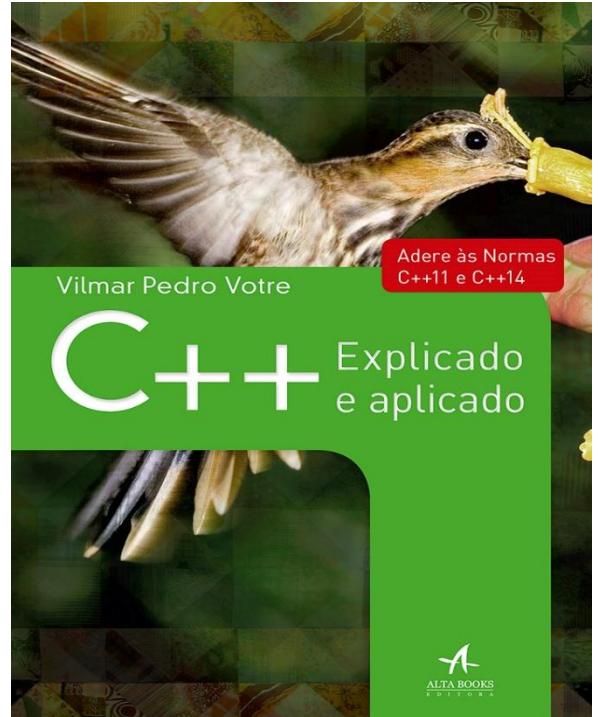
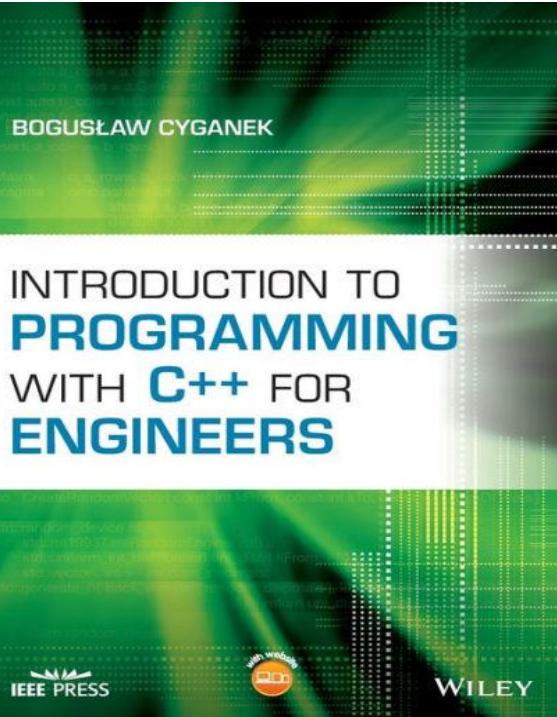
An Intensive Course for Scientists, Engineers, and Programmers

Peter Gottschling

SECOND EDITION

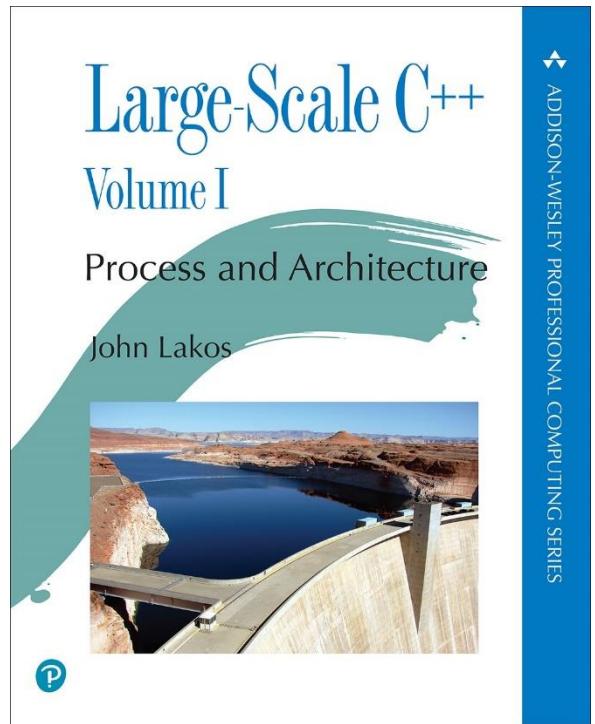
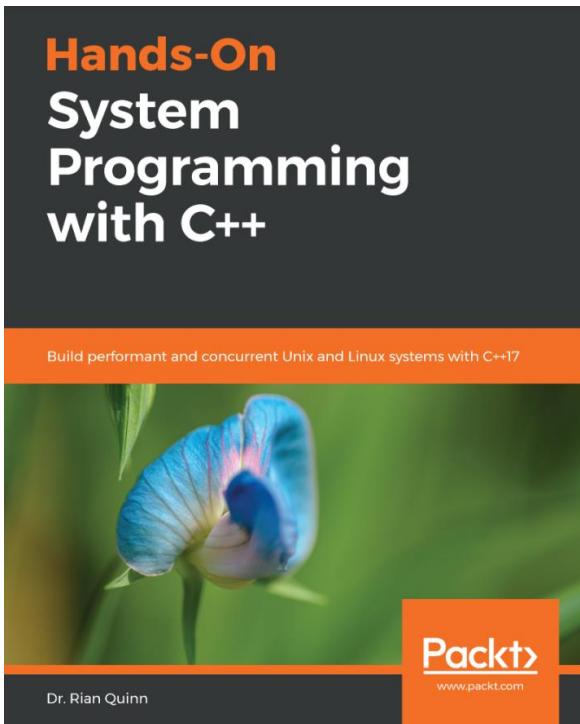
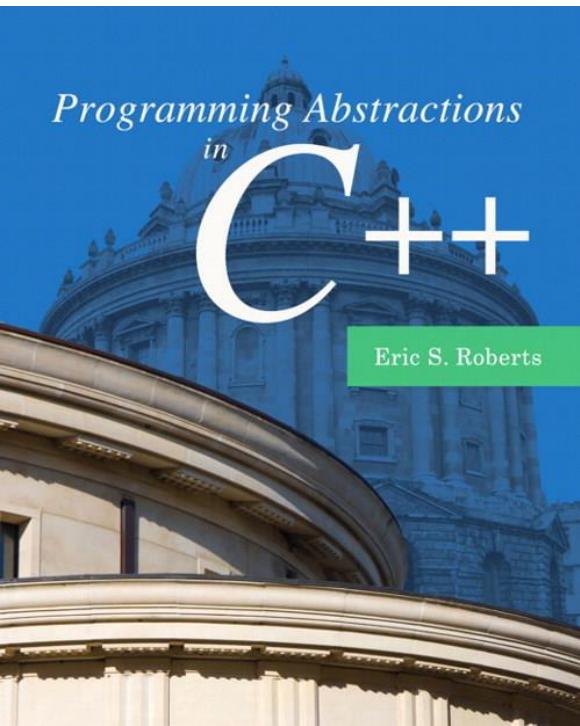


C++ In-Depth Series Bjarne Stroustrup



Eu quero aprender C++!

Por onde começar?

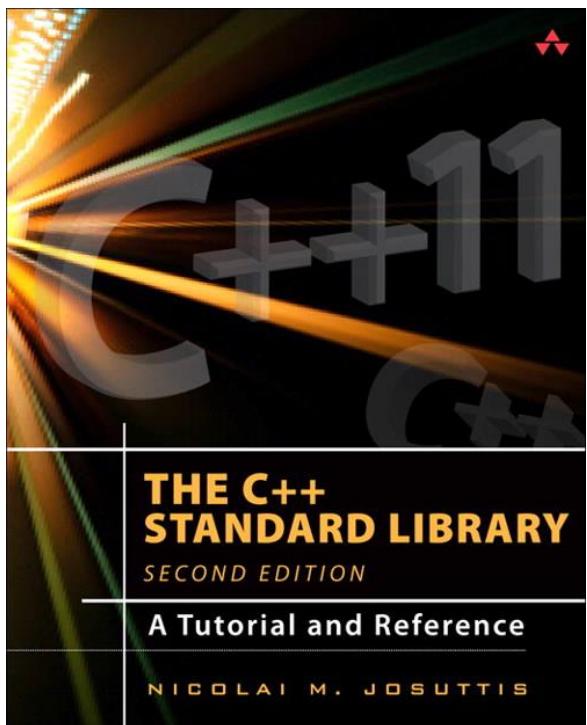


ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Eu quero aprender C++!

O que preciso saber?

- Standard C++ Library



http://www.cplusplus.com/reference/ Reference - C++ Reference

Reference

Standard C++ Library reference

C Library

The elements of the C language library aspects, from general utility functions:

Selected Standard Library Headers	Header	ISO Standard
<algorithm>	copy(), find(), sort()	§iso.25
<array>	array	§iso.23.3.2
<chrono>	duration, time_point	§iso.20.11.2
<cmath>	sqrt(), pow()	§iso.26.8
<complex>	complex, sqrt(), pow()	§iso.26.8
<forward_list>	forward_list	§iso.23.3.4
<iostream>	fstream, ifstream, ofstream	§iso.27.9.1
<future>	future, promise	§iso.30.6
<ios>	hex, dec, scientific, fixed, defaultfloat	§iso.27.5
<iostream>	istream, ostream, cin, cout	§iso.27.4
<map>	map, multimap	§iso.23.4.4
<memory>	unique_ptr, shared_ptr, allocator	§iso.20.6
<random>	default_random_engine, normal_distribution	§iso.26.5
<regex>	regex, smatch	§iso.28.8
<string>	string, basic_string	§iso.21.3
<set>	set, multiset	§iso.23.4.6
<sstream>	istrstream, ostrstream	§iso.27.8
<stdexcept>	length_error, out_of_range, runtime_error	§iso.19.2
<thread>	thread	§iso.30.3
<unordered_map>	unordered_map, unordered_multimap	§iso.23.5.4
<utility>	move(), swap(), pair	§iso.20.1
<vector>	vector	§iso.23.3.6

<http://www.cplusplus.com>

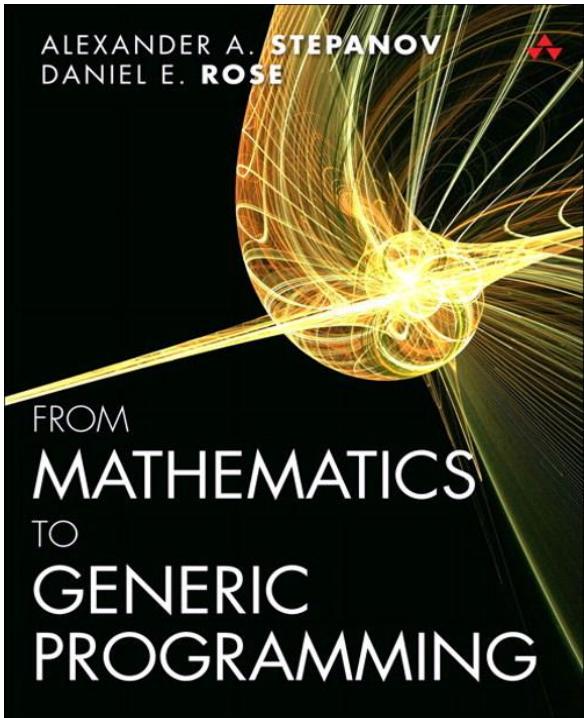
Eu quero aprender C++!

O que preciso saber?

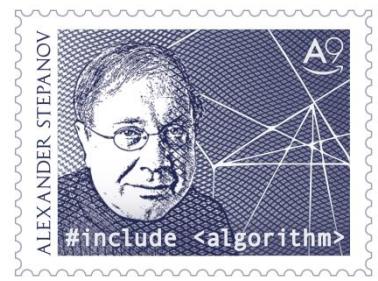
- **Standard Template Library (STL)**

The Standard Template Library, or *STL*, is a C++ library of **container classes**, **algorithms**, and **iterators**; it provides many of the basic algorithms and data structures of computer science.

The STL is a *generic* library, meaning that its components are heavily parameterized: almost every component in the STL is a template. https://www.boost.org/sgi/stl/stl_introduction.html



- Generic Programming* is an
- approach to programming...
 - focused on designing algorithms and data structures so that they work in the most general setting...
 - without loss of efficiency



Eu quero aprender C++!

O que preciso saber?

- `#include <algorithm>`

Standard Template Library: Algorithms

The header `<algorithm>` defines a collection of functions especially designed to be used on ranges of elements.

A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers. Notice though, that algorithms operate through iterators directly on the values, not affecting in any way the structure of any possible container (it never affects the size or storage allocation of the container).

```
template<ForwardIterator I, typename T, Compare C>
I lower_bound(I first, I last, const T& val, C compare)
{
    using std::distance;
    using std::advance;
    using D = DifferenceType<I>;
    D count = distance<>(first, last);
    while (count > D(0))
    {
        I it = first;
        D step = half<>(count);
        advance<>(it, step);
        if (compare(it, val))
        {
            first = it;
            ++first;
            count = count - step - D(1);
        }
        else
        {
            count = step;
        }
    }
    return first;
}
```

std::binary_search

```
template <class ForwardIterator, class T>
default (1)   bool binary_search (ForwardIterator first, ForwardIterator last,
                                const T& val);
```

```
1 template <class ForwardIterator, class T>
2     bool binary_search (ForwardIterator first, ForwardIterator last, const T& val)
3 {
4     first = std::lower_bound(first, last, val);
5     return (first!=last && !(val<*first));
6 }
```

Eu quero aprender C++!

O que preciso saber?

- Programação Concorrente



THE MICROSOFT JOURNAL FOR DEVELOPERS MARCH 2012 VOL 27 NO 3

Building the Internet of Things Torsten Grabs and Colin Miller	30
Develop Hybrid Native and Mobile Web Apps Shane Church	40
Create a Continuous Client Using Portable Class Libraries David Kean	48
New Concurrency Features in Visual C++ 11 Diego Dagum	56
Windows Phone Data Binding Jesse Liberty	62

COLUMNS

THE CUTTING EDGE
Build a Progress Bar with SignalR
Dino Esposito, page 6

DATA POINTS
Entity Framework Code First
and DbContext FAQs
Julie Lerman, page 14

FORECAST: CLOUDY
Exploring Cloud Architecture
Joseph Fultz, page 18

**THE WORKING
PROGRAMMER**
Talk to Me, Part 2: ELIZA
Ted Neward, page 74

Headers	
<atomic>	Atomic (header)
<thread>	Thread (header)
<mutex>	Mutex (header)
<condition_variable>	Condition variable (header)
<future>	Future (header)

This article discusses Visual C++ 11, a prerelease technology.
All related information is subject to change.

This article discusses:

- Parallel execution
- Asynchronous tasks
- Threads
- Variables and exceptions
- Synchronization
- Atomic types
- Mutexes and locks
- Condition variables

Technologies discussed:

Visual C++ 11

Code download available at:

code.msdn.microsoft.com/mag201203CPP

DIEGO DAGUM is a software developer with more than 20 years of experience. He's currently a Visual C++ community program manager with Microsoft.

THANKS to the following technical experts for reviewing this article:
David Cravey, Alon Fliess, Fabio Galuppo and Marc Gregoire

Eu quero aprender C++!

O que preciso saber?

The screenshot shows the cplusplus.com website's Reference section. The left sidebar has a 'C++' tab selected, with 'Reference' highlighted. Below it is a 'Reference' section containing links to C library, Containers, Input/Output, Multi-threading, and Other topics. The main content area is titled 'Reference' and 'Standard C++ Library reference'. It includes a table listing various C headers and their descriptions.

Header	Description
<cassert> (assert.h)	C Diagnostics Library (header)
<cctype> (ctype.h)	Character handling functions (header)
<cerrno> (errno.h)	C Errors (header)
<cfenv> (fenv.h)	Floating-point environment (header)
<cfloat> (float.h)	Characteristics of floating-point types (header)
<cinttypes> (inttypes.h)	C integer types (header)
<ciso646> (iso646.h)	ISO 646 Alternative operator spellings (header)
<climits> (limits.h)	Sizes of integral types (header)
<clocale> (locale.h)	C localization library (header)
<cmath> (math.h)	C numerics library (header)
<csetjmp> (setjmp.h)	Non local jumps (header)
<csignal> (signal.h)	C library to handle signals (header)
<cstdarg> (stdarg.h)	Variable arguments handling (header)
<cstdbool> (stdbool.h)	Boolean type (header)
<cstddef> (stddef.h)	C Standard definitions (header)

<https://cplusplus.com/reference>

Eu quero aprender C++!

O que preciso saber?

cppreference.com

 Log in

Search

[Page](#) [Discussion](#)

Standard revision: [Diff](#)

[View](#) [View source](#) [History](#)

C++ reference

[C++11](#), [C++14](#), [C++17](#), [C++20](#), [C++23](#), [C++26](#) | Compiler support [C++11](#), [C++14](#), [C++17](#), [C++20](#), [C++23](#), [C++26](#)

Language

Keywords – Preprocessor
ASCII chart
Basic concepts
Comments
Names (lookup)
Types (fundamental types)
The main function
Expressions
Value categories
Evaluation order
Operators (precedence)
Conversions – Literals
Statements
if – switch
for – range-for ([C++11](#))
while – do-while
Declarations – Initialization
Functions – Overloading
Classes (unions)
Templates – Exceptions
Freestanding implementations
Standard library (headers)
Named requirements
Feature test macros ([C++20](#))
Language support library
Program utilities
`source_location` ([C++20](#))
Coroutine support ([C++20](#))
Three-way comparison ([C++20](#))
Type support
`numeric_limits` – `type_info`
`initializer_list` ([C++11](#))
Concepts library ([C++20](#))
Diagnostics library
exception – System error
`basic_stacktrace` ([C++23](#))

Memory management library

`unique_ptr` ([C++11](#))
`shared_ptr` ([C++11](#))
`weak_ptr` ([C++11](#))
Memory resources ([C++17](#))
Allocators – Low level management
Metaprogramming library ([C++11](#))
Type traits – `ratio`
`integer_sequence` ([C++14](#))
General utilities library
Function objects – `hash` ([C++11](#))
Swap – Type operations ([C++11](#))
Integer comparison ([C++20](#))
`pair` – tuple ([C++11](#))
`optional` ([C++17](#))
`expected` ([C++23](#))
`variant` ([C++17](#)) – any ([C++17](#))
String conversions ([C++17](#))
Formatting ([C++20](#))
`bitset` – Bit manipulation ([C++20](#))
Debugging support ([C++26](#))
Strings library
`basic_string` – `char_traits`
`basic_string_view` ([C++17](#))
Null-terminated strings:
byte – multibyte – wide
Containers library
`vector` – `deque` – array ([C++11](#))
`list` – `forward_list` ([C++11](#))
`map` – `multimap` – set – `multiset`
`unordered_map` ([C++11](#))
`unordered_multimap` ([C++11](#))
`unordered_set` ([C++11](#))
`unordered_multiset` ([C++11](#))
Container adaptors
`span` ([C++20](#)) – `mdspan` ([C++23](#))

Iterators library

Ranges library ([C++20](#))
Algorithms library

Execution policies ([C++17](#))
Constrained algorithms ([C++20](#))

Numerics library

Common math functions
Mathematical special functions ([C++17](#))
Mathematical constants ([C++20](#))
Basic linear algebra algorithms ([C++26](#))
Numeric algorithms
Pseudo-random number generation
Floating-point environment ([C++11](#))
complex – valarray

Date and time library

Calendar ([C++20](#)) – Time zone ([C++20](#))

Localization library

locale – Character classification
`text_encoding` ([C++26](#))

Input/output library

Print functions ([C++23](#))
Stream-based I/O – I/O manipulators
`basic_istream` – `basic_ostream`
Synchronized output ([C++20](#))
File systems ([C++17](#))

Regular expressions library

`basic_regex` – Algorithms

Default regular expression grammar

Concurrency support library

thread – `jthread` ([C++20](#))
atomic – `atomic_flag`
`atomic_ref` ([C++20](#)) – `memory_order`
Mutual exclusion – Semaphores ([C++20](#))
Condition variables – Futures
latch ([C++20](#)) – barrier ([C++20](#))
Safe Reclamation ([C++26](#))

Technical specifications

Standard library extensions

(library fundamentals TS)
`resource_adaptor` – `invocation_type`

Standard library extensions v2

(library fundamentals TS v2)
`propagate_const` – `ostream_joiner` – `randint`
`observer_ptr` – Detection idiom

Standard library extensions v3

(library fundamentals TS v3)
`scope_exit` – `scope_fail` – `scope_success` – `unique_resource`

Parallelism library extensions v2

(parallelism TS v2)
`simd`

Concurrency library extensions

(concurrency TS)

Transactional Memory

(TMTS)

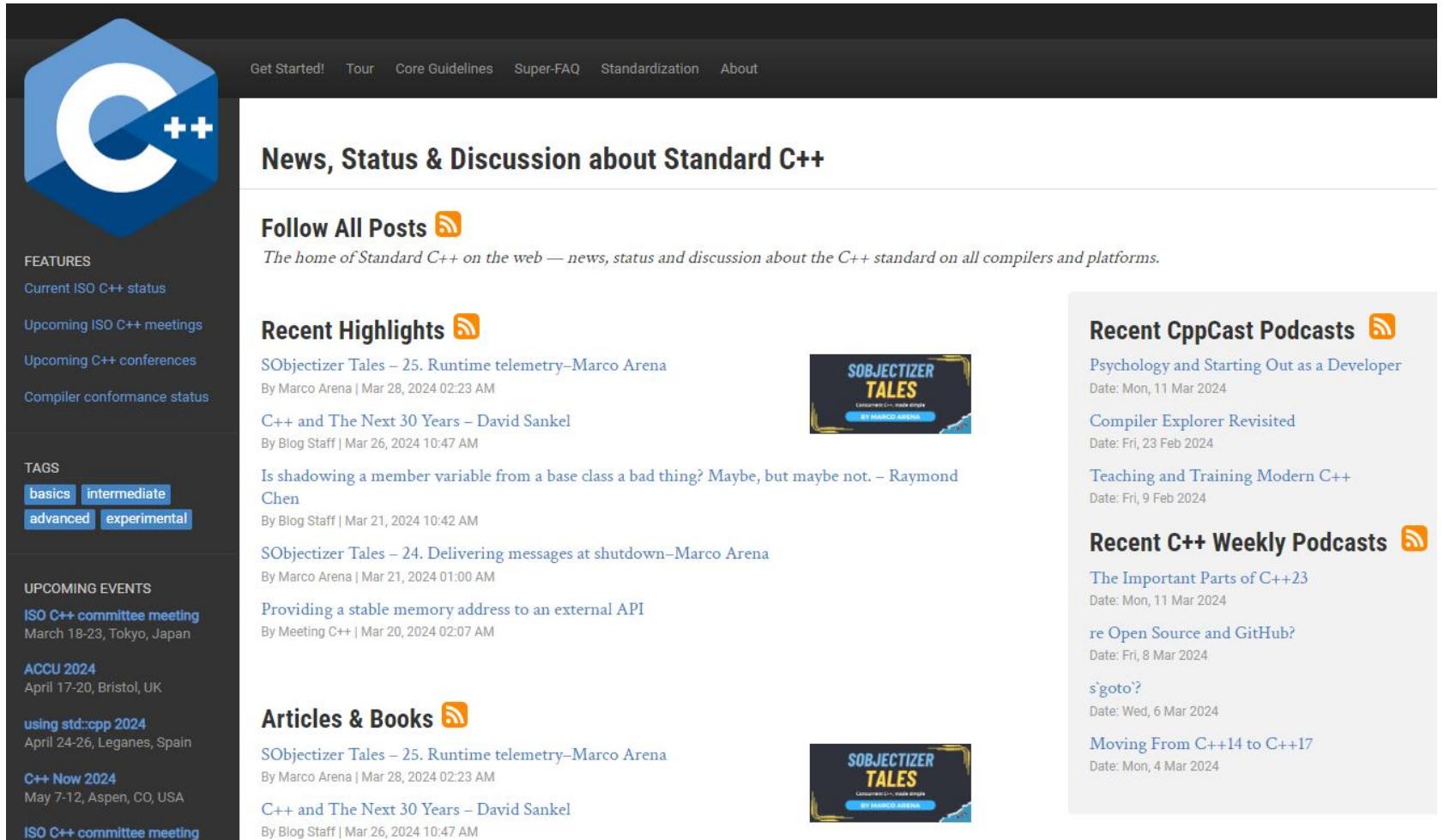
Reflection

(reflection TS)

External Links – Non-ANSI/ISO Libraries – Index – std Symbol Index

Eu quero aprender C++!

O que preciso saber?



The screenshot shows the ISO C++ website homepage. At the top, there's a large blue hexagonal logo with a white 'C++' symbol. Below it, a navigation bar includes links for 'Get Started!', 'Tour', 'Core Guidelines', 'Super-FAQ', 'Standardization', and 'About'. On the left sidebar, there are sections for 'FEATURES' (Current ISO C++ status, Upcoming ISO C++ meetings, Upcoming C++ conferences, Compiler conformance status), 'TAGS' (basics, intermediate, advanced, experimental), and 'UPCOMING EVENTS' (ISO C++ committee meeting, ACCU 2024, using std::cpp 2024, C++ Now 2024, ISO C++ committee meeting). The main content area features a section titled 'News, Status & Discussion about Standard C++' with a heading 'Follow All Posts' and an RSS icon. It includes a brief description: 'The home of Standard C++ on the web — news, status and discussion about the C++ standard on all compilers and platforms.' Below this, there are three columns of recent posts: 'Recent Highlights' (SObjectizer Tales - 25. Runtime telemetry by Marco Arena, C++ and The Next 30 Years by David Sankel, Is shadowing a member variable from a base class a bad thing? by Raymond Chen), 'Recent CppCast Podcasts' (Psychology and Starting Out as a Developer by Marco Arena, Compiler Explorer Revisited by Marco Arena, Teaching and Training Modern C++ by Marco Arena), and 'Recent C++ Weekly Podcasts' (The Important Parts of C++23 by Marco Arena, re Open Source and GitHub? by Marco Arena, s'goto? by Marco Arena, Moving From C++14 to C++17 by Marco Arena).

<https://isocpp.org/>

Eu quero aprender C++!

O que preciso saber?

- Core Guidelines

The image shows a screenshot of the C++ Core Guidelines website on the left and two book covers on the right.

C++ Core Guidelines Website:

- Logo:** A blue hexagon containing a white 'C' with a green arrow pointing to it, followed by '++'.
- Header:** 'C++ Core Guidelines' and 'February 15, 2024'.
- Editors:** Bjarne Stroustrup and Herb Sutter.
- Description:** This is a living document (code) project, this work of derivative works from this project requires LICENSE file for details, modify, and derive from.
- Comments:** Comments and suggestions extend this document.
- Available libraries:** Available libraries implemented outlines our aims and
- Problems:** The sets of rules or enforceability.

Books:

- Best Practices for Modern C++** by Rainer Grimm (Author), Bjarne Stroustrup (Foreword). The cover features a large 'C++' logo and the text 'CORE GUIDELINES EXPLAINED'.
- BEAUTIFUL C++** by J. Guy Davidson / Kate Gregory. The cover features a colorful, abstract graphic of overlapping shapes and the text 'BEAUTIFUL C++'.

Eu quero aprender C++!

O que preciso saber?



CppCon

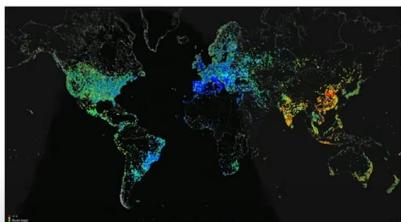
@CppCon • 145K subscribers • 1.5K videos

Visit cppcon.org for details on next year's conference. CppCon sponsors have made it pos...more



C++20: C++ at 40

stability and evolution



Bjarne Stroustrup

Morgan Stanley, Columbia University

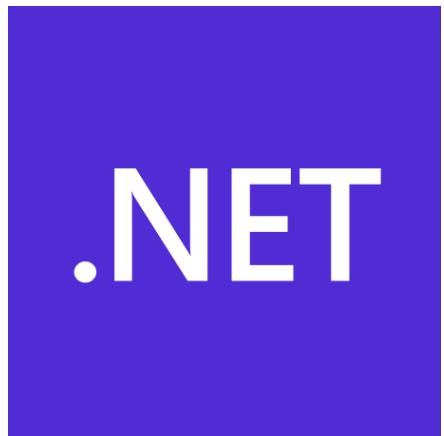
www.stroustrup.com

<https://www.youtube.com/@CppCon>

TECNOLOGIAS QUE UTILIZAM C++

Quem depende de C++?

- Máquina Virtual



runtime / src / coreclr / vm /	
amd64	Remove support for non-indirect code headers (#100351)
arm	Remove support for non-indirect code headers (#100351)
arm64	Remove support for non-indirect code headers (#100351)
eventing	Remove strtok, strtok_s, wcstok_s PAL APIs (#98008)
i386	Remove support for non-indirect code headers (#100351)
loongarch64	Remove support for non-indirect code headers (#100351)
ppc64le	Upstream coreclr and pal changes to support powerpc (ppc64le) a...
riscv64	Remove support for non-indirect code headers (#100351)
s390x	Build support for s390x: clr.iltools and cir.paltests (#53289)
wks	Replaced the '_obj' CMake objects and their INTERFACE's, with just...
CMakeLists.txt	Move last P/Invoke error from native Thread and delete C/C++ Sa...
ClrEtwAll.man	Add EventSource event for WaitHandle waits (#94737)
ClrEtwAllMeta.lst	Add EventSource event for WaitHandle waits (#94737)
amsi.cpp	December infra rollout - remove duplicated 'src' from coreclr subr...
amsi.h	December infra rollout - remove duplicated 'src' from coreclr subr...
appdomain.cpp	Move last P/Invoke error from native Thread and delete C/C++ Sa...
appdomain.hpp	Move various thread-local objects that were used in self-containe...
appdomain.inl	December infra rollout - remove duplicated 'src' from coreclr subr...
appdomainnative.cpp	Convert HELPER_METHOD_FRAME to QCalls (5/N) (#96526)
appdomainnative.hpp	Convert HELPER_METHOD_FRAME to QCalls (5/N) (#96526)

<https://github.com/dotnet/runtime/tree/main/src/coreclr/vm>

Quem depende de C++?

- Máquina Virtual



Files

master Go to file

src

- demo/share
- hotspot
- cpu
 - aarch64
 - arm
 - ppc
 - riscv
 - s390
 - x86
 - gc
 - abstractInterpreter_x86.cpp
 - assembler_x86.cpp
 - assembler_x86.hpp
 - assembler_x86.inline.hpp
 - bytes_x86.hpp
 - c1_CodeStubs_x86.cpp
 - c1_Defs_x86.hpp
 - c1_FpuStackSim_x86.cpp
 - c1_FpuStackSim_x86.hpp
 - c1_FrameMap_x86.cpp
 - c1_FrameMap_x86.hpp
 - c1_LIRAssembler_x86.cpp

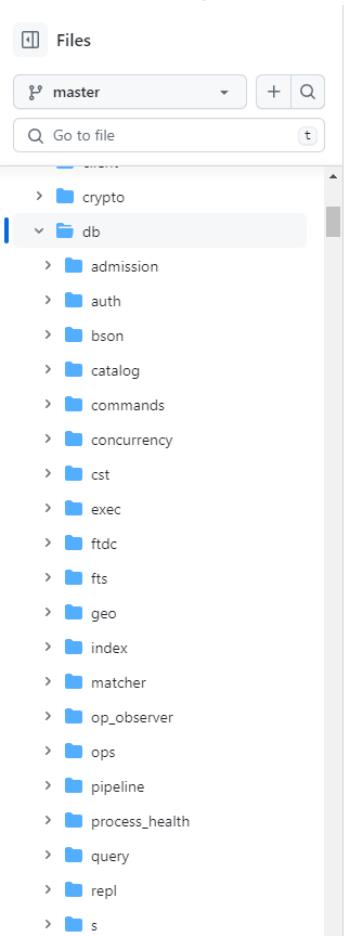
jdk / src / hotspot / cpu / x86 /	
stubGenerator_x86_64_pow.cpp	8304828: Lots of constant static data not declared static const in c...
stubGenerator_x86_64_sin.cpp	8304828: Lots of constant static data not declared static const in c...
stubGenerator_x86_64_tan.cpp	8304828: Lots of constant static data not declared static const in c...
stubRoutines_x86.cpp	8252136: Several methods in hotspot are missing "static"
stubRoutines_x86.hpp	8322768: Optimize non-subword vector compress and expand API...
stubRoutines_x86_32.cpp	8320272: Make method_entry_barrier address shared
stubRoutines_x86_64.cpp	8322768: Optimize non-subword vector compress and expand API...
templateInterpreterGenerator_x86.cpp	8301997: Move method resolution information out of the cpCache
templateInterpreterGenerator_x86_32.cpp	8301498: Replace NULL with nullptr in cpu/x86
templateInterpreterGenerator_x86_64.cpp	8301498: Replace NULL with nullptr in cpu/x86
templateTable_x86.cpp	8320276: Improve class initialization barrier in TemplateTable::new
templateTable_x86.hpp	8301997: Move method resolution information out of the cpCache
upcallLinker_x86_32.cpp	8254693: Add Panama feature to pass heap segments to native co...
upcallLinker_x86_64.cpp	8322962: Upcall stub might go undetected when freezing frames
vmStructs_x86.hpp	8265403: consolidate definition of CPU features
vm_version_x86.cpp	8327999: Remove copy of unused registers for cpu features check ...
vm_version_x86.hpp	8325991: Accelerate Poly1305 on x86_64 using AVX2 instructions
vmreg_x86.cpp	8292153: x86: Represent Registers as values
vmreg_x86.hpp	8292153: x86: Represent Registers as values
vmreg_x86.inline.hpp	8292153: x86: Represent Registers as values

<https://github.com/openjdk/jdk/tree/master/src/hotspot>



Quem depende de C++?

- Banco de Dados (orientado a documentos)



A screenshot of a GitHub repository interface. On the left, there's a sidebar with a 'Files' tab, a dropdown set to 'master', and a search bar. Below these are buttons for '+' and 'Go to file'. The main area shows a tree view of files under the 'db' directory. The 'db' folder is expanded, showing sub-directories like 'admission', 'auth', 'bson', 'catalog', 'commands', 'concurrency', 'cst', 'exec', 'ftdc', 'fts', 'geo', 'index', 'matcher', 'op_observer', 'ops', 'pipeline', 'process_health', 'query', 'repl', and 's'. To the right of the tree view is a list of files with their corresponding GitHub pull requests:

File	Pull Request
client.cpp	SERVER-87015 Optimize registration of operations in `MozJS...`
client.h	SERVER-84765 Return the correct port at host field when embed...
client_context_test.cpp	SERVER-81830 Create Clients with Service, not ServiceContext
client_out_of_line_executor.cpp	SERVER-87974 Enable modernize-use-override and apply fixes (#2...
client_out_of_line_executor.h	SERVER-87974 Enable modernize-use-override and apply fixes (#2...
client_out_of_line_executor_test.cpp	SERVER-82100 Remove Client::unspecifiedService and force all call...
client_strand.cpp	SERVER-81059 bulk-revert of include of boost iif.hpp
client_strand.h	SERVER-77047 SERVER-77048 IWYU automated changes up to en...
client_strand_test.cpp	SERVER-81830 Create Clients with Service, not ServiceContext
clientcursor.cpp	SERVER-88271 Make QueryKnobConfigurations accessible via Expr...
clientcursor.h	SERVER-88271 Make QueryKnobConfigurations accessible via Expr...
cloner.cpp	SERVER-87805 movePrimary should not preserve UUID's of unshar...
cloner.h	SERVER-87805 movePrimary should not preserve UUID's of unshar...
cloner.idl	SERVER-86977: Make redact a required field for set parameters an...
cluster_auth_mode_option.idl	SERVER-41412 Introduce a search_beta_auth suite.
cluster_command_translations.h	SERVER-81266: MultiUpdateCoordinator sends cluster update to s...
cluster_role.cpp	SERVER-82303 separate MetricsTree per Service (#17183)
cluster_role.h	SERVER-85770 Enable internal routing capabilities by default
cluster_transaction_api.cpp	SERVER-84183 Support FLE2 in embedded router mode (#18323)
cluster_transaction_api.h	SERVER-87974 Enable modernize-use-override and apply fixes (#2...

<https://github.com/mongodb/mongo/tree/master/src/mongo/db>

Quem depende de C++?

- Ambiente de tempo de execução JavaScript



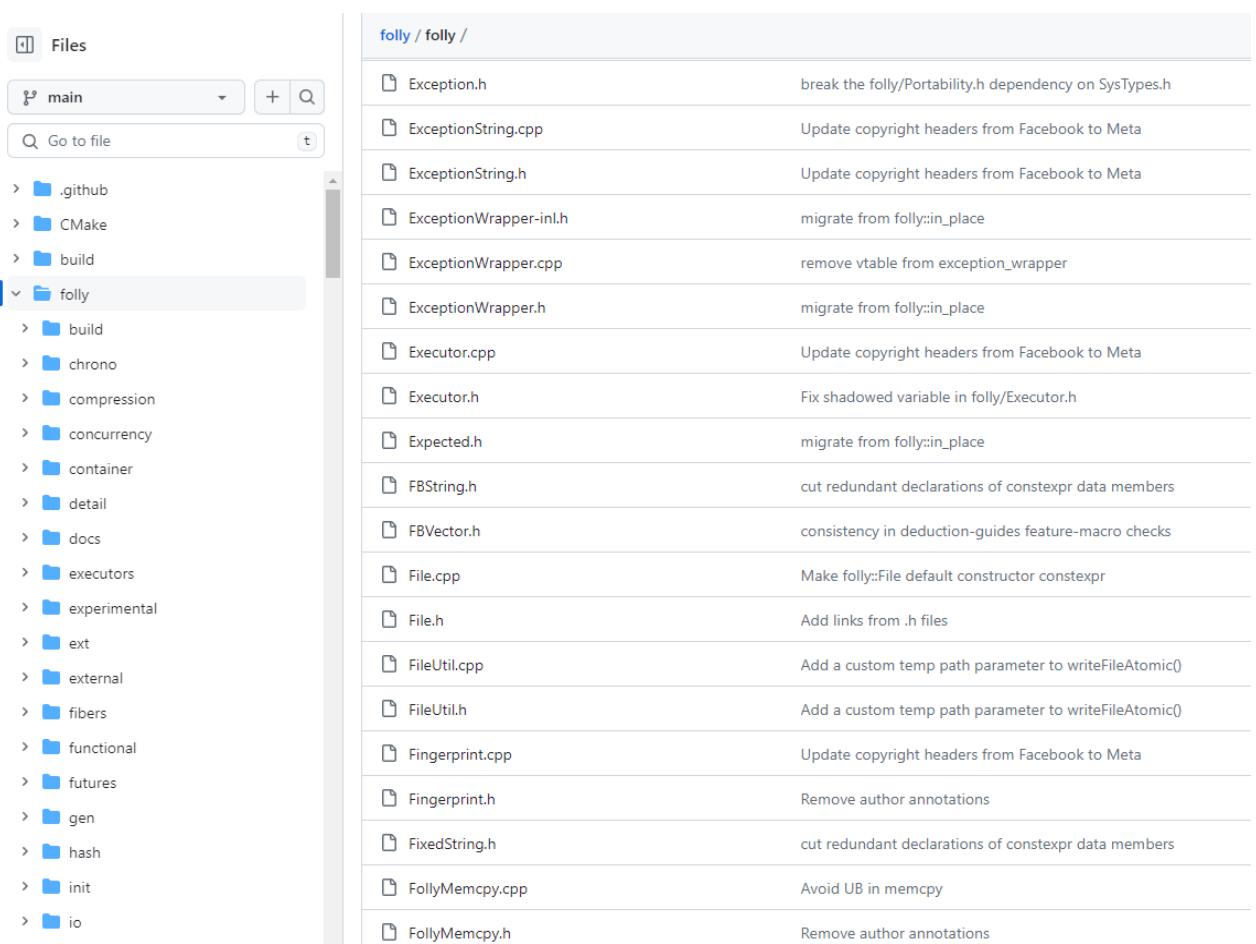
node / src /	
dataqueue	quic: further implementation details
inspector	inspector: add NodeRuntime.waitingForDebugger event
large_pages	src: large page attributing an id on Linux
permission	src: avoid shadowed string in fs_permission
quic	quic: various additional cleanups, fixes in Endpoint
res	build: remove dtrace & etw support
tracing	src: trace threadpool event
.clang-tidy	tools: add clang-tidy rule in src
README.md	src: remove ContextEmbedderIndex:kBindingDataStoreIndex
acorn_version.h	deps: update acorn to 8.11.3
aliased_buffer-inl.h	src: remove aliased buffer weak callback
aliased_buffer.h	src: remove aliased buffer weak callback
aliased_struct-inl.h	src: add AliasedStruct utility
aliased_struct.h	src: add AliasedStruct utility
async_wrap-inl.h	src: add worker per-isolate binding initialization
async_wrap.cc	module: bootstrap module loaders in shadow realm
async_wrap.h	async_hooks,inspector: implement inspector api without async_wrap
base64-inl.h	src: remove base64_select_table and base64_table
base64.h	src: remove base64_select_table and base64_table
base64_version.h	deps: update base64 to 0.5.2

<https://github.com/mongodb/mongo/tree/master/src/mongo/db>

Quem depende de C++?

- Rede Social

 Meta
facebook



The screenshot shows a GitHub repository interface. On the left, there's a file browser with a sidebar containing links like 'main', '.github', 'CMake', 'build', and 'folly'. The 'folly' folder is expanded, showing sub-directories: build, chrono, compression, concurrency, container, detail, docs, executors, experimental, ext, external, fibers, functional, futures, gen, hash, init, and io. To the right, a list of commits in the 'folly/folly' directory is displayed, each with a file icon, the file name, and a brief description.

File	Description
Exception.h	break the folly/Portability.h dependency on SysTypes.h
ExceptionString.cpp	Update copyright headers from Facebook to Meta
ExceptionString.h	Update copyright headers from Facebook to Meta
ExceptionWrapper-inl.h	migrate from folly::in_place
ExceptionWrapper.cpp	remove vtable from exception_wrapper
ExceptionWrapper.h	migrate from folly::in_place
Executor.cpp	Update copyright headers from Facebook to Meta
Executor.h	Fix shadowed variable in folly/Executor.h
Expected.h	migrate from folly::in_place
FBString.h	cut redundant declarations of constexpr data members
FBVector.h	consistency in deduction-guides feature-macro checks
File.cpp	Make folly::File default constructor constexpr
File.h	Add links from .h files
FileUtil.cpp	Add a custom temp path parameter to writeFileAtomic()
FileUtil.h	Add a custom temp path parameter to writeFileAtomic()
Fingerprint.cpp	Update copyright headers from Facebook to Meta
Fingerprint.h	Remove author annotations
FixedString.h	cut redundant declarations of constexpr data members
FollyMemcpy.cpp	Avoid UB in memcpy
FollyMemcpy.h	Remove author annotations

<https://github.com/facebook/folly/tree/main/folly>

Quem depende de C++?

- Navegador Web



chromium / chrome / browser / background /

background_application_list_model.h	Update copyright headers in chrome
background_application_list_model_unittest.cc	[Extensions] Create c/b/extensions/permissions directory
background_contents.cc	Remove WebContents::CreateParams::is_never_visible
background_contents.h	[mparch] Clean up WCD::DidNavigateMainFramePostCom...
background_contents_service.cc	[Extensions cleanup] Update GetExtensionById() in background/
background_contents_service.h	Use FlakyDanglingUntriaged for previously marked raw_ptrs.
background_contents_service_factory.cc	Implement the non-deprecated builder.
background_contents_service_factory.h	Implement the non-deprecated builder.
background_contents_service_observer.h	Update copyright headers in chrome
background_contents_service_unittest.cc	Delete unused notification includes
background_mode_manager.cc	Removed DCHECK from BackgroundModeManager::OnProfileWill...
background_mode_manager.h	Release keep-alive for force installed extensions in background m...
background_mode_manager_aura.cc	Update copyright headers in chrome
background_mode_manager_chromeos.cc	Update copyright headers in chrome
background_mode_manager_mac.cc	Don't add Chromium as a login item
background_mode_manager_unittest.cc	Release keep-alive for force installed extensions in background m...
background_mode_manager_win.cc	Rename chrome/app/ branding strings
background_mode_optimizer.cc	Clean up some BackgroundModeOptimizer code
background_mode_optimizer.h	Clean up some BackgroundModeOptimizer code
background_mode_optimizer_unittest.cc	Clean up some BackgroundModeOptimizer code

<https://github.com/chromium/chromium/tree/main/chrome>

Quem depende de C++?

- Motor de jogos



**UNREAL
ENGINE**

Files

release + 🔎

Go to file ⚙

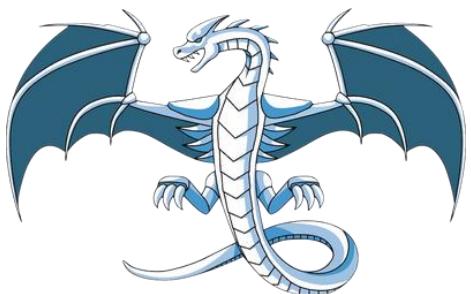
- Engine
- Binaries
- Build
- Config
- Documentation
- Extras
- Plugins
- Programs
- Shaders
- Source
 - Developer
 - Editor
 - Programs
 - Runtime
 - AIModule
 - AVEncoder
 - AVIWriter
 - AdvancedWidgets
 - Advertising
 - Analytics
 - Android
 - AnimGraphRuntime

UnrealEngine / Engine / Source / Runtime / RenderCore / Private /	
RayTracingValidationShaders.cpp	Deprecated non-command list RHI methods.
RenderCaptureInterface.cpp	Ran IWYU on RHI and RenderCore, private only.
RenderCore.cpp	Add GNearClippingPlane_RenderThread to be accessed on only o...
RenderGraphAllocator.cpp	Unshelved from pending changelist '24339224':
RenderGraphBlackboard.cpp	Ran IWYU on RHI and RenderCore, private only.
RenderGraphBuilder.cpp	Deprecated non-command list RHI methods.
RenderGraphEvent.cpp	Deprecated InitRHI() in favor of InitRHI(FRHICommandListBase&).
RenderGraphPass.cpp	Ran IWYU on RHI and RenderCore, private only.
RenderGraphPrivate.cpp	Disable AsyncCompute and ParallelExecute when FlushGPU is ena...
RenderGraphPrivate.h	Fixes AsyncCompute passes not showing up in ProfileGPU
RenderGraphResourcePool.cpp	Fixed validation error in transient allocator due to transient allo...
RenderGraphResourcePool.h	Fixed validation error in transient allocator due to transient allo...
RenderGraphResources.cpp	Deprecated non-command list RHI methods.
RenderGraphTrace.cpp	Added parallel setup for RDG passes
RenderGraphUtils.cpp	Bindless: consider BindlessResourceParameters array when clearin...
RenderGraphValidation.cpp	Added CVar to RDG to control whether to enable expensive RDG v...
RenderResource.cpp	Deprecated non-command list RHI methods.
RenderTargetPool.cpp	Removing uses of ForceIntoNonStreamingMemoryTracking while ...
RenderTransform.cpp	Initialize FRenderTransform::Identity with explicit values, since FMa...
RenderUtils.cpp	Deprecated InitRHI() in favor of InitRHI(FRHICommandListBase&).

<https://github.com/EpicGames/UnrealEngine/tree/release/Engine>

Quem depende de C++?

- Infraestrutura de compiladores



Files

main Go to file

llvm benchmarks bindings cmake docs examples include llvm-c llvm ADT

ADL.h APFixedPoint.h APFloat.h APInt.h APSInt.h AddressRanges.h AllocatorList.h Any.h ArrayRef.h BitVector.h Bitfields.h BitmaskEnum.h Bitset.h

SCCIterator.h STLExtras.h STLForwardCompat.h STLFunctionalExtras.h ScopeExit.h ScopedHashTable.h Sequence.h SetOperations.h SetVector.h SmallBitVector.h SmallPtrSet.h SmallSet.h SmallString.h SmallVector.h SmallVectorExtras.h SparseBitVector.h SparseMultiSet.h SparseSet.h StableHashing.h Statistic.h

[SCCIterator] Union MST node by rank correctly (#86389)
[AIX][TOC] Add -mtodata/-mno-todata options on AIX (#67999)
[ADT] Backport std::to_underlying from C++23 (#70681)
Revert "[ADT] Apply fixes from modernize-type-trait (NFC)"
Use std::decay_t (NFC)
[ADT] Use Empty Base Optimization for Allocators
[ADT] Support iterating size-based integer ranges.
New SetOperations and unit testing for all SetOperations
[ADT] Use llvm::is_contained (NFC)
Use llvm::count(lr)_{zero,one} (NFC)
[ADT] Use a constexpr version of llvm::bit_ceil (NFC) (#79709)
[llvm][CycleInfo] Quick look-up for block in cycle.
Revert "[ADT] Add std::string_view conversion to SmallString (#833...
[LLVM][ADT] Explicitly convert size_t values to SmallVector's size ty...
[ADT] Allow specifying the size of resulting SmallVector in 'map_to...'
[ADT] Include bit.h instead of MathExtras.h (NFC)
[llvm] Use std::is_unsigned instead of std::numeric_limits (NFC)
[ADT] Remove an extraneous ternary operator (NFC)
[NFC] Move StableHashing.h from CodeGen to ADT (#67704)
[Support] Drop unnecessary const from a return type (NFC)

<https://github.com/llvm/llvm-project/tree/main/llvm>

BIBLIOTECAS PARA C++

Awesome C++

A curated list of awesome C/C++ frameworks, libraries, resources, and shiny things

<https://cpp.libhunt.com/>

All Categories

Artificial Intelligence	20	Inter-process communication	12
Asynchronous Event Loop	11	JSON	31
Audio	13	Logging	21
Biology	4	Machine Learning	21
BitTorrent	3	Math	31
CLI	17	Miscellaneous	60
Compression	21	Multimedia	10
Concurrency	38	Networking	57
Containers	19	PDF	5
Cryptography	20	Physics	10
Encryption	1	Reliability Engineering	1
CSV	6	Robotics	10
Database	30	Scientific Computing	13
Data Structures	6	Scripting	24
Debug	28	Serialization	19
Frameworks	36	Standard Libraries	7
Game Engine	28	Testing	7
Graphics	22	Text Processing	7
GUI	49	Video	6
Hooking	2	Virtual Machines	5
IDE	2	Web Application Framework	17
Image Processing	24	XML	8
Internationalization	5		

Awesome C++ / All Categories / Inter-Process Communication

ZeroMQ

ZeroMQ core engine in C++, implements ZMQ/3.1

[zeromq.org](#) [Source Code](#) [Changelog](#) [Suggest Changes](#)

Popularity ★ 9.2

Activity 7.6

Stars 9,198

Watchers 408

Forks 2,311

Last Commit 3 days ago

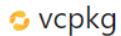
↔ Popular Comparisons

ZeroMQ vs gRPC	ZeroMQ vs Cap'n Proto
ZeroMQ vs nanomsg	ZeroMQ vs WAMP
ZeroMQ vs Apache Thrift	

Vcpkg

Gerenciador de Pacotes

<https://vcpkg.io/en/packages>



Get Started Packages Documentation GitHub Blog

Showing 1-10 of 2396 packages

3fd | Version: 2.6.3

C++ Framework For Fast Development

Compatibility: ✓ arm-uwp ✓ arm64-windows ✓ x64-linux ✓ x64-osx ✓ x64-uwp ✓ x64-windows ✓ x64-windows-static ✓ x86-windows
[View Details](#)

7zip | Version: 23.01

Library for archiving file with a high compression ratio.

Compatibility: ✓ arm-uwp ✓ arm64-windows ✓ x64-linux ✓ x64-osx ✓ x64-uwp ✓ x64-windows ✓ x64-windows-static ✓ x86-windows
[View Details](#)

ableton | Version: 3.0.6

Meta-package for transitioning to abletonlink port.

Compatibility: ✓ arm-uwp ✓ arm64-windows ✓ x64-linux ✓ x64-osx ✓ x64-uwp ✓ x64-windows ✓ x64-windows-static ✓ x86-windows
[View Details](#)

ableton-link | Version: 3.1.0

Ableton Link, a technology that synchronizes musical beat, tempo, and phase across multiple applications running on one or more devices.

Compatibility: ✓ arm-uwp ✓ arm64-windows ✓ x64-linux ✓ x64-osx ✓ x64-uwp ✓ x64-windows ✓ x64-windows-static ✓ x86-windows
[View Details](#)

abseil | Version: 20230802.1



13612

an open-source collection designed to augment the C++ standard library. Abseil is an open-source collection of C++ library code designed to augment the C++ standard library. The Abseil library code is...

Compatibility: ✓ arm-uwp ✓ arm64-windows ✓ x64-linux ✓ x64-osx ✓ x64-uwp ✓ x64-windows ✓ x64-windows-static ✓ x86-windows
[View Details](#)

Boost

<http://boost.org/>



Boost provides free peer-reviewed portable C++ source **libraries**.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The [Boost license](#) encourages the use of Boost libraries for all users with minimal restrictions.

I. RAII and Memory Management

- 1. Boost.SmartPointers
- 2. Boost.PointerContainer
- 3. Boost.ScopeExit
- 4. Boost.Pool

II. String Handling

- 5. Boost.StringAlgorithms
- 6. Boost.LexicalCast
- 7. Boost.Format
- 8. Boost.Regex
- 9. Boost.Xpressive
- 10. Boost.Tokenizer
- 11. Boost.Spirit

V. Algorithms

- 29. Boost.Algorithm
- 30. Boost.Range
- 31. Boost.Graph

III. Containers

- 12. Boost.Multimap
- 13. Boost.Bimap
- 14. Boost.Array
- 15. Boost.Unordered
- 16. Boost.CircularBuffer
- 17. Boost.Heap
- 18. Boost.Intrusive
- 19. Boost.MultiArray
- 20. Boost.Container

IV. Data Structures

- 21. Boost.Optional
- 22. Boost.Tuple
- 23. Boost.Any
- 24. Boost.Variant
- 25. Boost.PropertyTree
- 26. Boost.DynamicBitset
- 27. Boost.Tribool
- 28. Boost.CompressedPair

VI. Communication

- 32. Boost.Asio
- 33. Boost.Interprocess
- VII. Streams and Files
- 34. Boost.IOSStreams
- 35. Boost.Filesystem

VIII. Time

- 36. Boost.DateTime
- 37. Boost.Chrono
- 38. Boost.Timer

IX. Functional Programming

- 39. Boost.Phoenix
- 40. Boost.Function
- 41. Boost.Bind
- 42. Boost.Ref
- 43. Boost.Lambda

X. Parallel Programming

- 44. Boost.Thread
- 45. Boost.Atomic
- 46. Boost.Lockfree
- 47. Boost.MPI

XI. Generic Programming

- 48. Boost.TypeTraits
- 49. Boost.EnableIf
- 50. Boost.Fusion

XII. Language Extensions

- 51. Boost.Coroutine
- 52. Boost.Foreach
- 53. Boost.Parameter
- 54. Boost.Conversion

XIII. Error Handling

- 55. Boost.System
- 56. Boost.Exception

XIV. Number Handling

- 57. Boost.Integer
- 58. Boost.Accumulators
- 59. Boost.MinMax
- 60. Boost.Random
- 61. Boost.NumericConversion

XV. Application Libraries

- 62. Boost.Log
- 63. Boost.ProgramOptions
- 64. Boost.Serialization
- 65. Boost.Uuid

XVI. Design Patterns

- 66. Boost.Flyweight
- 67. Boost.Signals2
- 68. Boost.MetaStateMachine

XVII. Other Libraries

- 69. Boost.Utility
- 70. Boost.Assign
- 71. Boost.Swap
- 72. Boost.Operators

<https://theboostcpplibraries.com/>

[https://www.boost.org/users/history/version 1 84 0.html](https://www.boost.org/users/history/version_1_84_0.html)

ZeroMQ

<http://zeromq.org>

- *Intelligent socket library for messaging*
 - Variedade nos padrões de comunicação
 - Request-Reply, Publisher-Subscriber, Push-Pull, Dealer-Router, ...
 - Suporta: *inproc, IPC, TCP, TIPC, multicast*
- Modelo de concorrência baseado em atores (*Erlang-style*)
- *Open Source* (escrita em C++)
- Multiplas plataformas
- Integração com diversas linguagens (mais de 30)
 - C, C++, Java, C#, Python, ...
- *Deploy simples* (uma *library*)
- Alta performance
 - <http://zeromq.org/results:multicore-tests>
 - ~6 milhões de mensagens por segundo



EA Standard Template Library (EASTL)

<https://github.com/electronicarts/EASTL>



EASTL stands for Electronic Arts Standard Template Library. It is a C++ template library of containers, algorithms, and iterators useful for runtime and tool development across multiple platforms. It is a fairly extensive and robust implementation of such a library and has an emphasis on high performance above all other considerations.

EASTL Public

EASTL stands for Electronic Arts Standard Template Library. It is an extensive and robust implementation that has an emphasis on high performance.

● C++ ⭐ 7,633 📄 BSD-3-Clause 896 63 9 Updated 4 days ago

c-plus-plus games c-plus-plus-11
modern-cpp stl c-plus-plus-14
c-plus-plus-17 eastl c-plus-plus-20



Julian Manolov
@_jju_

[Follow](#)

The EASTL we've been using in Battlefield and all other Frostbite games is now open SOURCE:



electronicarts/EASTL

EASTL stands for Electronic Arts Standard Template Library. It is an extensive and robust implementation that has an emphasis on high performance.

[github.com](https://github.com/electronicarts/EASTL)

POCO C++ Libraries

<https://pocoproject.org/index.html>

SIMPLIFY C++ DEVELOPMENT

The **POCO C++ Libraries** are powerful cross-platform C++ libraries for building network- and internet-based applications that run on desktop, server, mobile, IoT, and embedded systems.



EMBEDDED DEVICES

Create software for connected embedded devices running Linux, Windows Embedded or QNX.



MOBILE APPS

Create cross-platform backends in C++ for iOS and Android applications and combine it with a native or HTML5-based user interface.



INTERNET OF THINGS

Create software for IoT devices that talk to cloud backends over HTTP REST APIs. See [macchina.io](#) for an IoT platform built with POCO.



SERVER APPLICATIONS

Build application servers in C++ that talk to SQL databases, MongoDB or Redis.



CLOUD & MICROSERVICES

Build high-performance microservices with REST APIs for data analytics or machine learning in C++.



DESKTOP APPS

Build desktop applications that talk to REST APIs or SQL databases.

Anatomy of an Option

- > An option has a
- > full name
- > an optional short name (one character)
- > information used for printing, or usage statement.
- > an optional argument name.
- > An option can be optional or required.
- > An option can be repeatable, which means that it can be given more than once on the command line.



```
Excluded "MacchinaServer" class has static functions to determine generic environment information at run time.
> env::value("MACCHINA")
    Returns the value of an environment variable. Throws a std::invalid_argument if the variable is undefined.

> env::exists("MACCHINA")
    Check whether an environment variable is defined.

> env::erase("MACCHINA")
    Set the value of an environment variable.

> env::clear()
    Set the value of an environment variable.
```

System Information at Run Time

The `MacchinaServer` class has static functions to determine generic environment information at run time.

- > `env::value("MACCHINA")`
- > `env::exists("MACCHINA")`
- > `env::erase("MACCHINA")`
- > `env::clear()`
- > `env::value("MACCHINA")`

Applications

Building Command-Line and Server Applications.

Network Programming

Writing network and internet applications.

Memory Management

Reference counting, shared pointers, buffer management, and more.

Strings, Text and Formatting

Working with strings, formatting, tokenizing, regular expressions and text encodings.

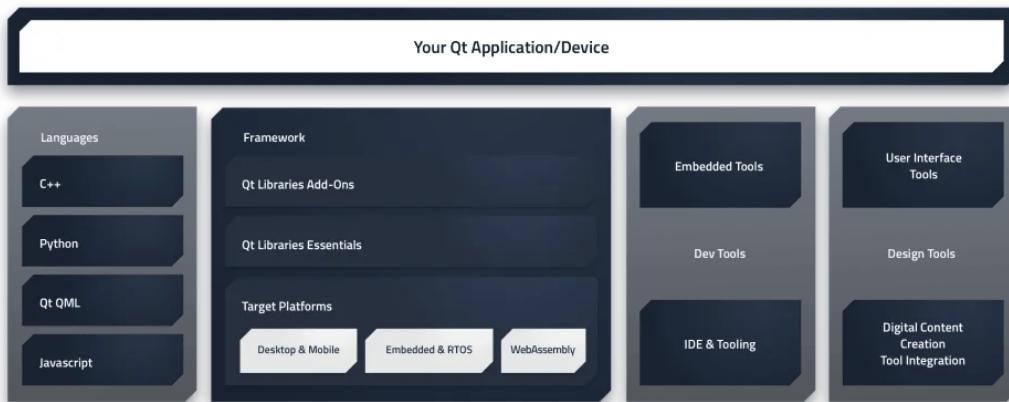
Platform and Environment

Getting information about the system you're running on.

Qt

<https://www.qt.io/>

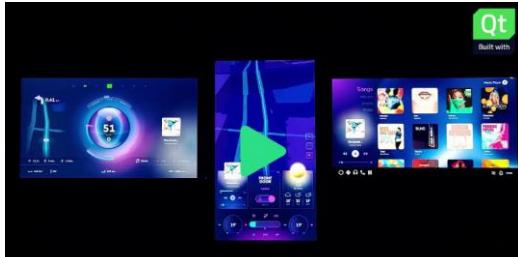
1+ Billion Devices and Applications Powered and Assured by Qt



Qt Framework

Use Qt's libraries and APIs to develop software with native C++ performance for mobile, desktop, and embedded systems.

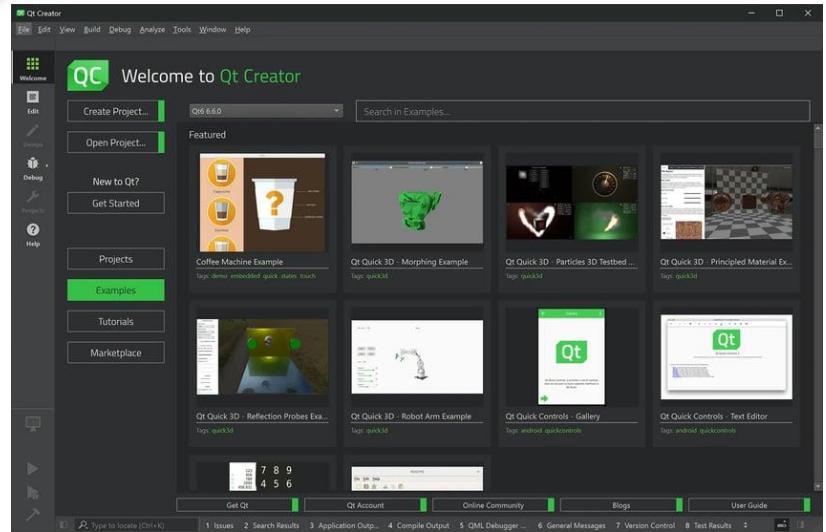
<https://www.qt.io/resources/videos/digital-cockpit-on-multiple-screens-and-operating-systems?hsLang=en>



Qt Creator

The AI-Enabled Cross-Platform IDE

Qt Creator is a cross-platform integrated development environment (IDE) tailored for maximum developer productivity. Qt Creator supports the use of coding assistants like GitHub Copilot during programming. It aids developers in creating software for desktop, mobile, and embedded platforms.



Cinder C++

<http://libcinder.org/>



Cinder is a *C++ library* for programming with aesthetic intent - the sort of development often called *creative coding*. This includes domains like graphics, audio, video, and computational geometry. Cinder is cross-platform, with official support for macOS, Windows, Linux, iOS, and Windows UWP.

Cinder is production-proven, powerful enough to be the primary tool for professionals, but still suitable for learning and experimentation.



Dia Lights

by Kollision, Transform, Martin Professional, Danish Industry

As Denmark's biggest permanent interactive media facade, Dia Lights comprises more than 80.000 individual LEDs and covers an area of 190 x 20 meters on the recently rebuilt HQ of the Confederation of Danish Industry in the heart of Copenhagen.



Planetary iPad App

by Robert Hodgin, Bloom Studio

Visualize your music collection on your iPad as a galaxy of stars, planets and moons. This project became Cooper Hewitt National Design Museum's first digital acquisition.

<https://libcinder.org/gallery>

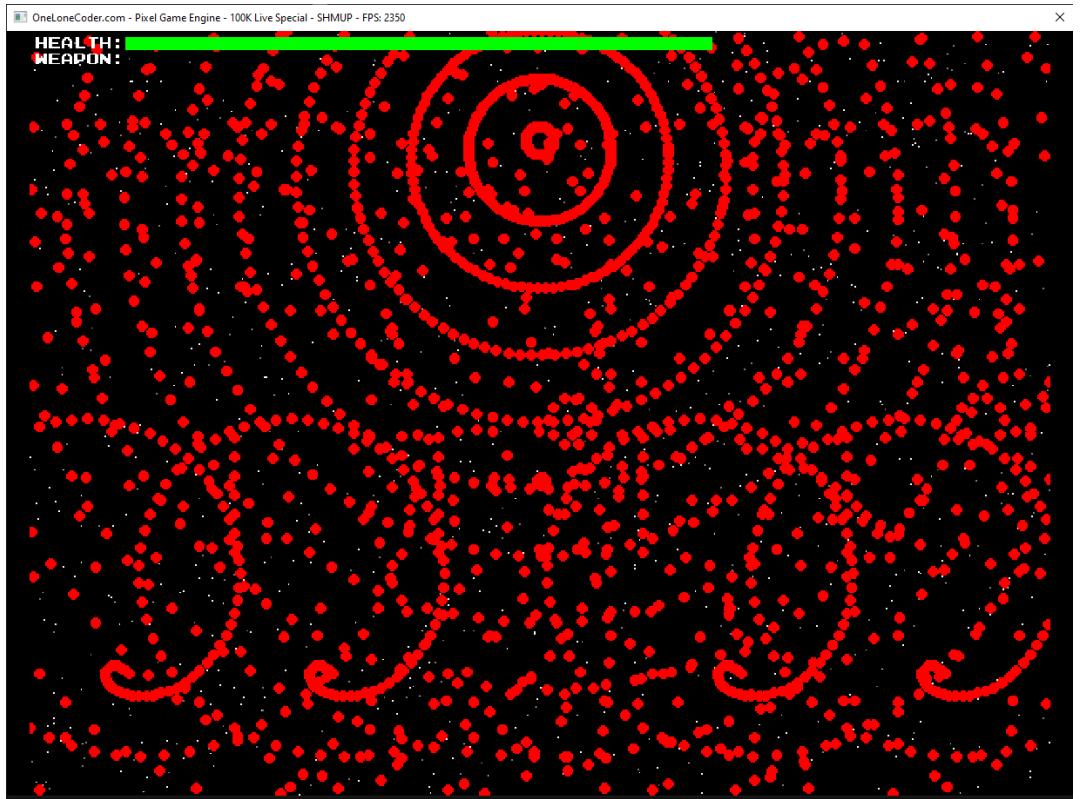
olcPixelGameEngine

<https://github.com/OneLoneCoder/olcPixelGameEngine>

Javidx9 / PixelGameEngine / SmallerProjects / OneLoneCoder_PGE_ShootEmUp.cpp

Code Blame 484 lines (393 loc) · 15.1 KB ⚡ Code 55% faster with GitHub Copilot

```
69  class Example : public olc::PixelGameEngine
314      bool OnUserUpdate(float fElapsedTime) override
433          // GRAPHICS
434          Clear(olc::BLACK);
435
436          // Update & Draw Stars
437          for (size_t i=0; i<aryStars.size(); i++)
438          {
439              auto& s = aryStars[i];
440              s.y += fScrollSpeed * fElapsedTime * ((aryStars.size() >> 2) ? 0.8f : 1.0f);
441              if (s.y >= (float)ScreenHeight())
442                  s = { (float)(rand() % ScreenWidth()), 0.0f };
443
444              Draw(s, (i < aryStars.size() >> 2) ? olc::DARK_GREY : olc::WHITE);
445
446
447          SetPixelMode(olc::Pixel::MASK);
448
449          // Draw Enemies
450          for (auto& e : listEnemies)    DrawSprite(e.pos, sprEnemy[e.def.nSpriteID]);
451
452          // Draw Player
453          DrawSprite(vPlayerPos, sprPlayer);
454
455          SetPixelMode(olc::Pixel::NORMAL);
456
457          // Draw Enemy Bullets
458          for (auto& b : listEnemyBullets) FillCircle(b.pos, 3, olc::RED);
459
460
461          // Draw Player Bullets
462          for (auto& b : listPlayerBullets) FillCircle(b.pos, 3, olc::CYAN);
463
464          // Draw Fragments
465          for (auto& b : listFragments) Draw(b.pos, olc::YELLOW);
466
467          // Draw Player Health Bar
468          DrawString(4, 4, "HEALTH:");
469          FillRect(60, 4, (fPlayerHealth / 1000.0f * 576.0f), 8, olc::GREEN);
470
471          DrawString(4, 14, "WEAPON:");
472          FillRect(60, 14, (fPlayerGunTemp / 100.0f * 576.0f), 8, olc::YELLOW);
```



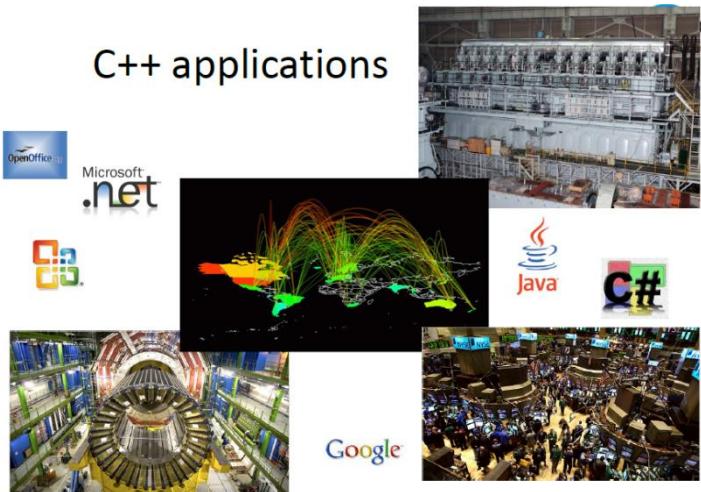
https://github.com/OneLoneCoder/Javidx9/blob/master/PixelGameEngine/SmallerProjects/OneLoneCoder_PGE_ShootEmUp.cpp

ENTÃO, POR QUE APRENDER C++?

WRAP-UP

Mais aplicações de C++

C++ applications



C++ Applications



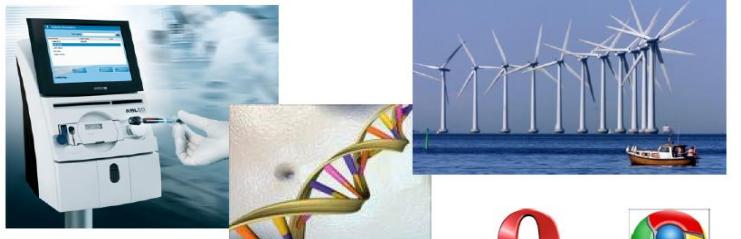
www.lextrait.com/vincent/implementations.html

AMADEUS
Your technology partner

Stroustrup - ACCU'13

C++ Applications

- www.research.att.com/~bs/applications.html



Stroustrup - ACCU'13

<http://www.stroustrup.com/applications.html>

<http://www.lextrait.com/vincent/implementations.html>

<http://www.infoq.com/presentations/Cplusplus-11-Bjarne-Stroustrup>

It's all about Polyglot Programming!



C++ supports systems programming. This implies that C++ code is able to effectively interoperate with software written in other languages on a system. The idea of writing all software in a single language is a fantasy. From the beginning, C++ was designed to interoperate simply and efficiently with C, assembler, and Fortran. By that, I meant that a C++, C, assembler, or Fortran function could call functions in the other languages without extra overhead or conversion of data structures passed among them.



"Nobody should call themselves a professional if they only knew one language."

...**C++**, of course; **Java**; maybe **Python** for mainline work... And if you know those, you can't help know sort of a little bit about **Ruby** and **JavaScript**, you can't help knowing **C** because that's what fills out the domain and of course **C#**. But again, **these languages create a cluster** so that **if you knew either five of the ones that I said, you would actually know the others...**

Conclusão

ou

por que você eu deveria aprendê-la?

- *Software Infrastructure*
 - *System Programming*
- Desempenho e Controle
 - *Zero-cost abstraction*
- Eficiente em termos de consumo de recursos
 - Bateria, Memória, ...
- Interoperável
- Relevante
 - Diversas aplicações com dependência direta ou indireta
 - Bem posicionada no mercado
- Multiparadigma
- Melhorando constantemente
 - próxima versão ISO C++ em 2026



A linguagem de programação C++ e por que você deveria aprendê-la

ISO/IEC 14882:2020
(C++ 20)

Fabio Galuppo, M.Sc.

fabiogaluppo@acm.org

<https://bit.ly/AprenderCpp2024>