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

ISO/IEC 14882:2020

(C++ 20)

Fabio Galuppo, M.Sc.

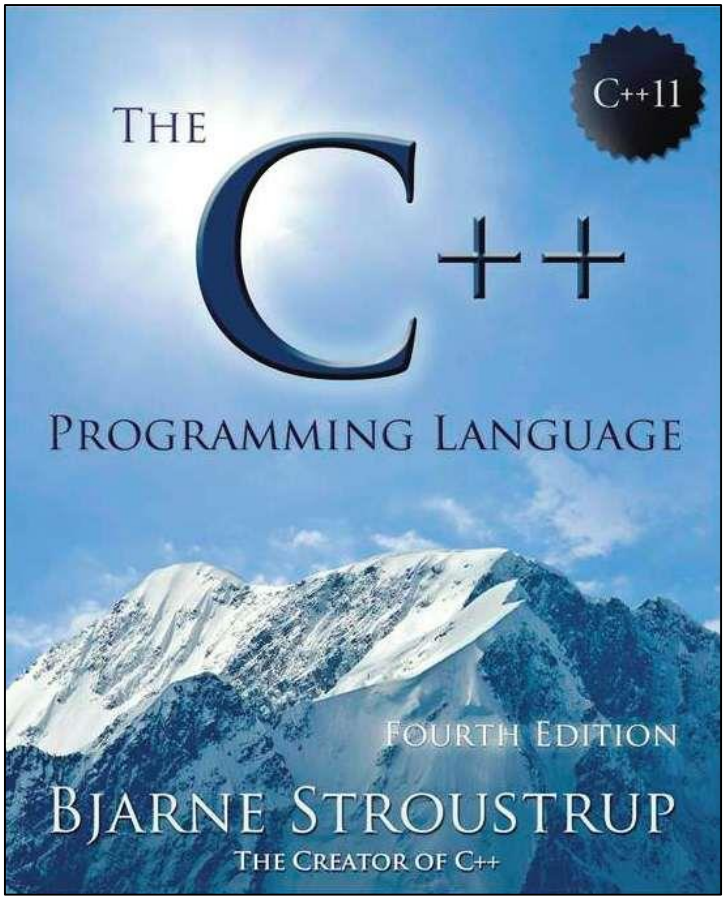
fabiogaluppo@acm.org

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

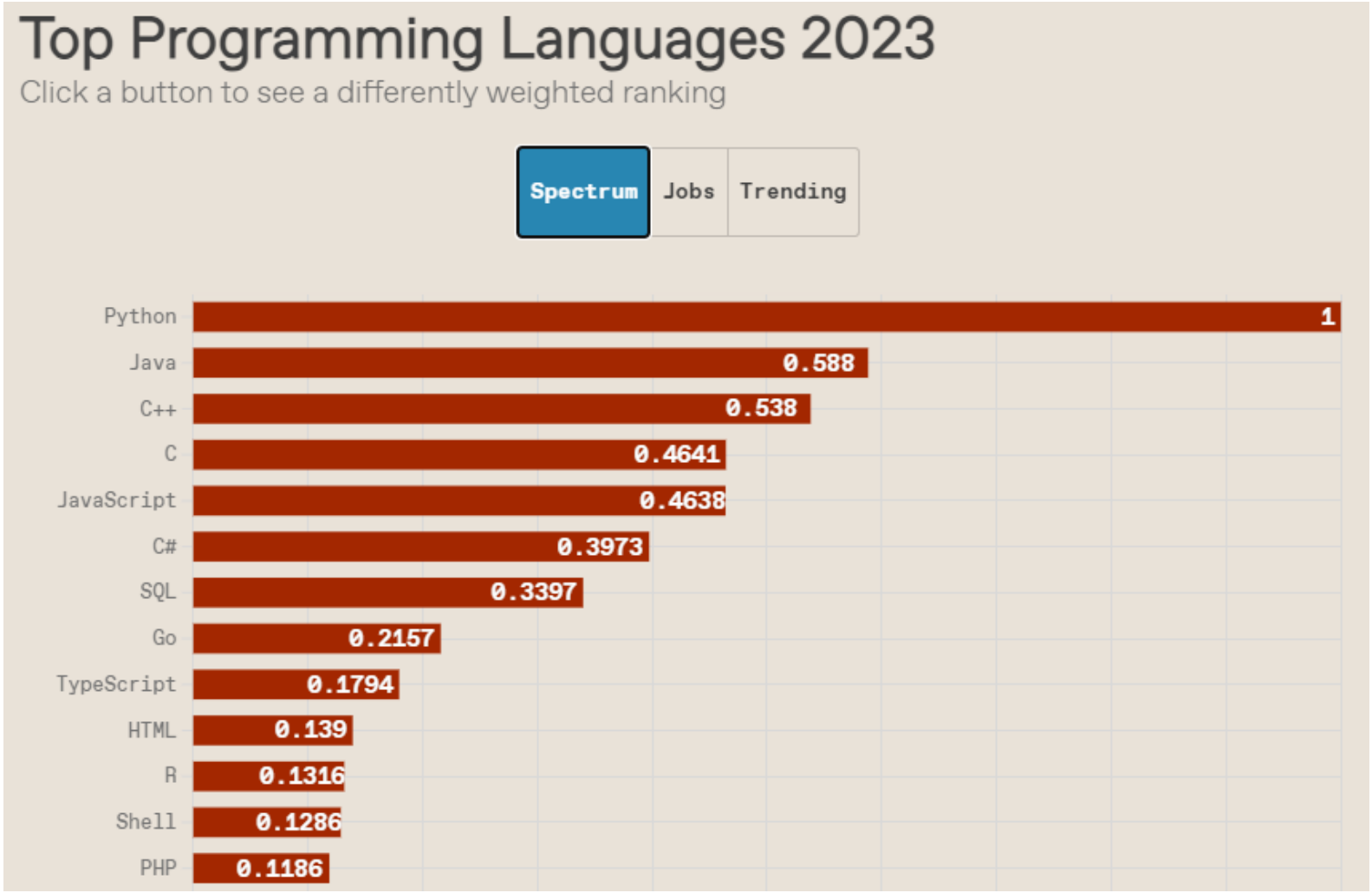
The top programming languages are in a long term decline: both Java and C have all time low scores in the TIOBE index. And almost all of the other top 10 languages are going down as well year to year. So what languages are taking advantage of this? It is all happening down in the charts around position 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

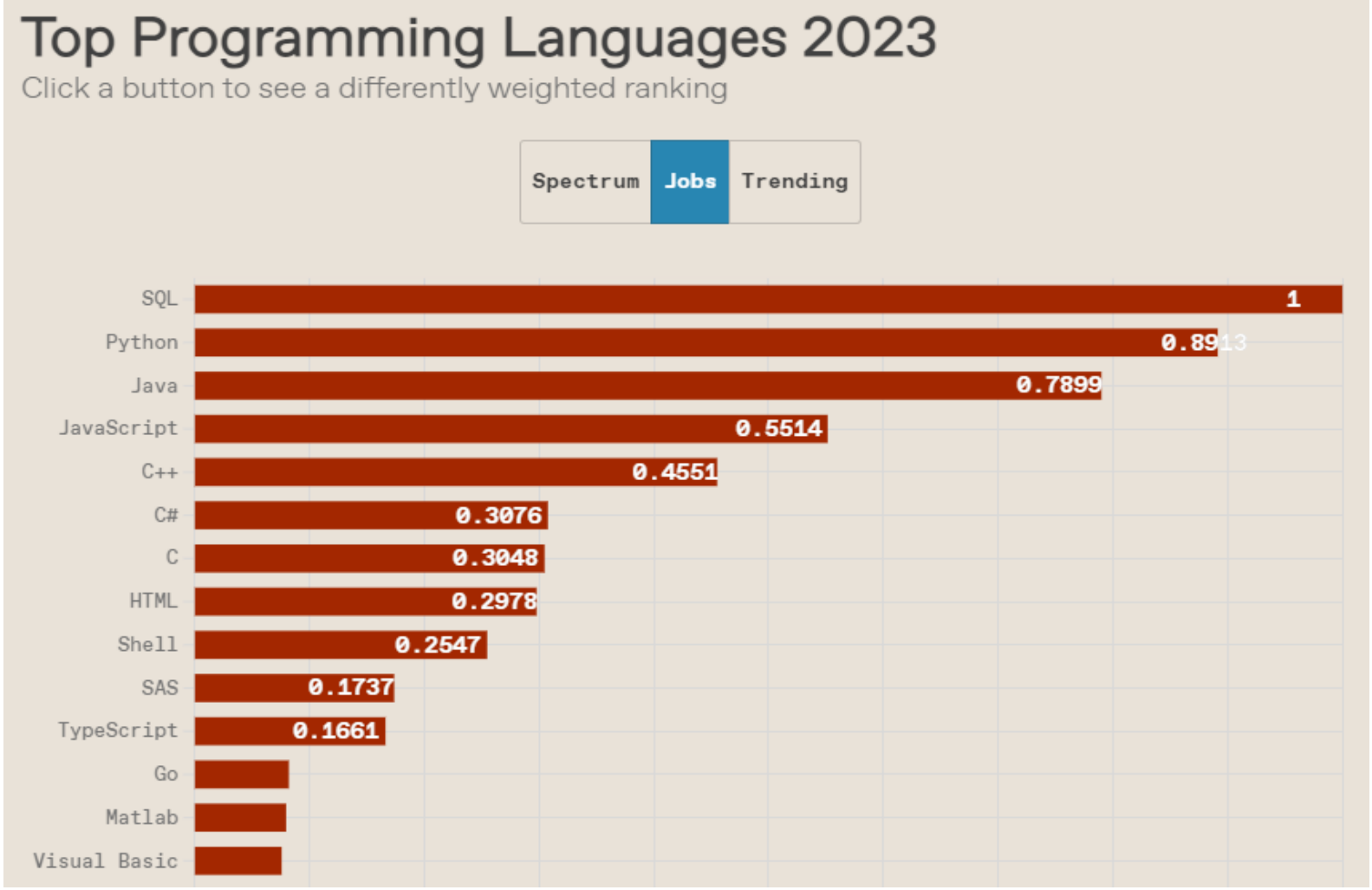
Python and SQL are on top, but old languages shouldn't be forgotten



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

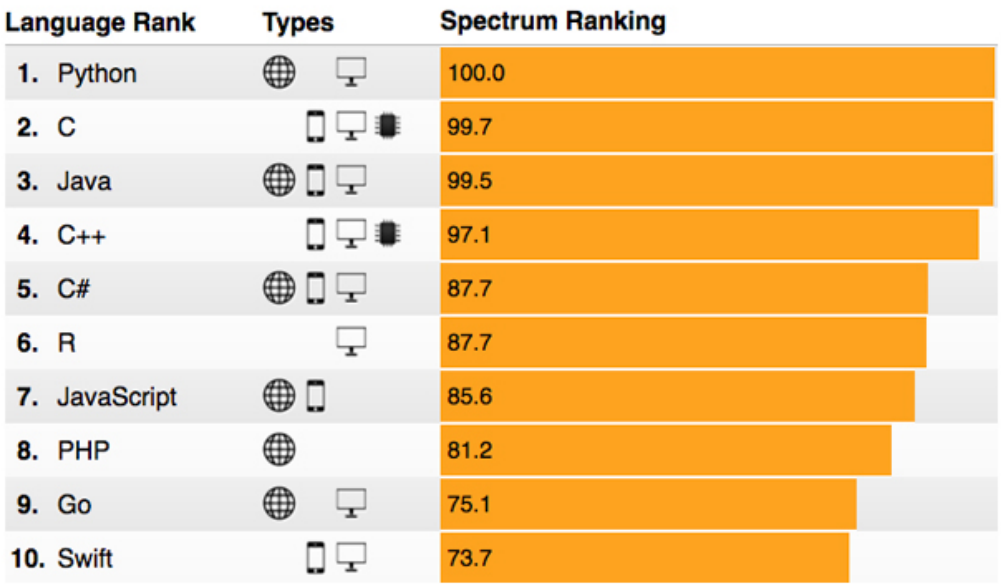


https://spectrum.ieee.org/the-top-programming-languages-2023

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?



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

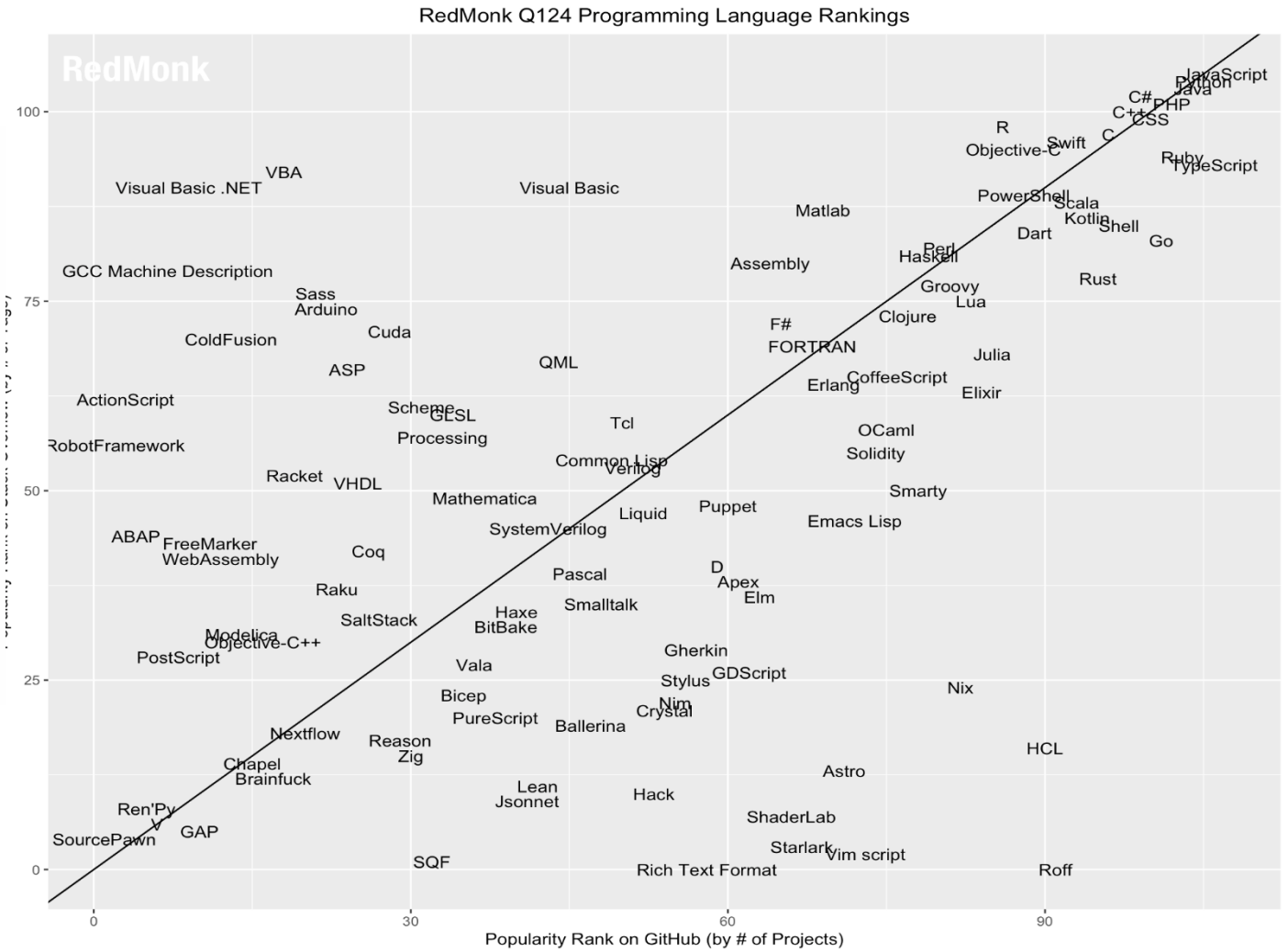
Python has continued its upward trajectory from last year and jumped two places to the No. 1 slot, though the top four—Python, C, Java, and C++—all remain very close in popularity. Indeed, in Diakopoulos's analysis of what the underlying metrics have to say about the languages currently in demand by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

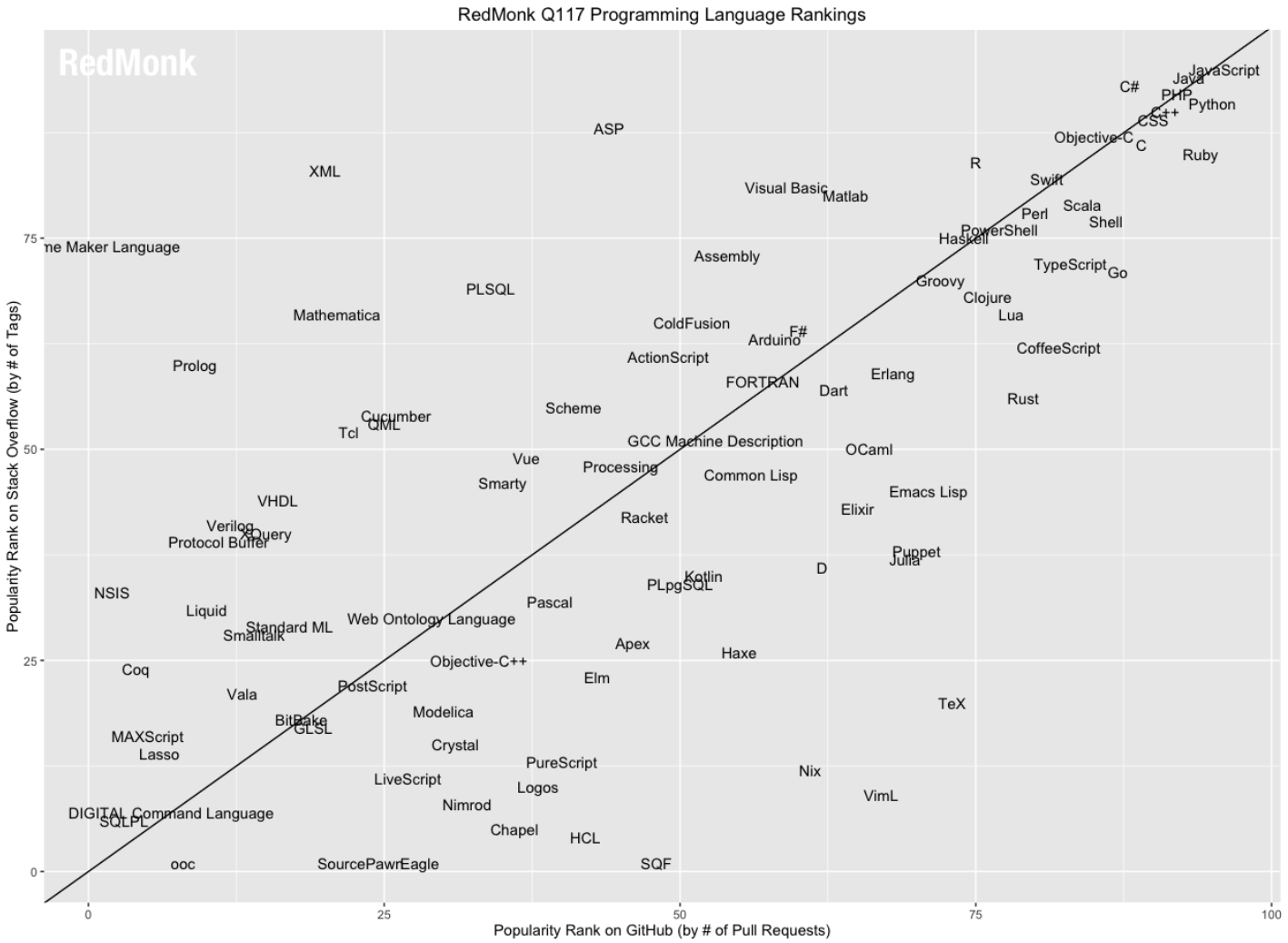
1 JavaScript
2 Python
3 Java
4 PHP
5 C#
6 TypeScript
6 CSS
8 C++
9 Ruby
10 C



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
C gcc #9	2.08	11,392	1633	2.08
C++ g++ #0	2.14	10,960	1927	2.14

mandelbrot

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

pidigits

source	secs	mem	gz	cpu secs
C gcc #2	0.81	11,392	416	0.81
C++ g++ #5	0.87	10,956	798	0.87

fasta

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

reverse-complement

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

binary-trees

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

https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/cpp.html

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

ORACLE
Oracle RDBMS
Last updated June 26, 2022, 10:58 p.m.

MySQL
MySQL
Last updated Sept. 24, 2023, 6:13 p.m.

Microsoft SQL Server
SQL Server
Last updated June 26, 2022, 10:58 p.m.

MongoDB.
MongoDB
Last updated Feb. 14, 2024, 8:34 a.m.

SCYLLA.
Scylla
Last updated Nov. 10, 2022, 3:40 p.m.

https://dbdb.io/stats

O QUE É C++?

O que é C++?

What is C++?

Smarter computin
Texas A&M Univer

Template meta-programming!

Class hierarchies

A hybrid language

Buffer overflows

A multi-paradigm programming language

Classes

It's C!

Too big!

Embedded systems programming language

Low level!

Generic programming

An object-oriented programming language

A random collection of features

Stroustrup - ACCU'13

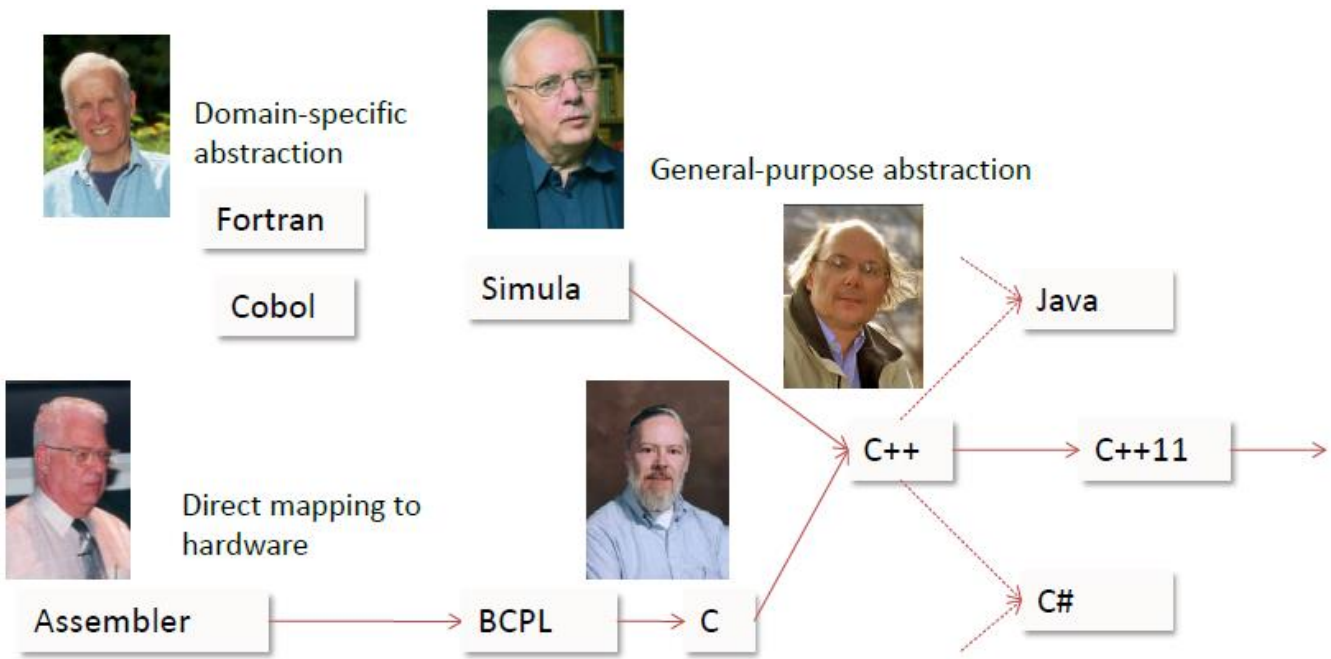
https://www.stroustrup.com/

The C++ Programming Language

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

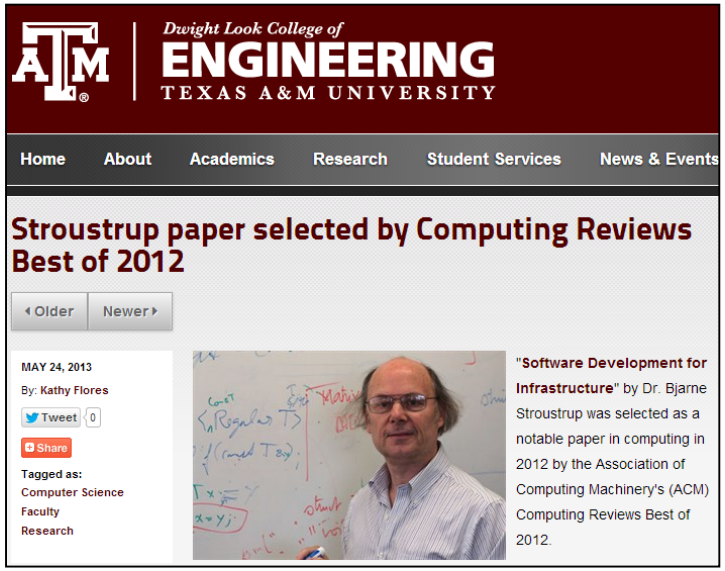
- is a better C
- supports data abstraction
- supports object-oriented programming
- supports generic programming.

"C++ is a general purpose programming language designed to make programming more enjoyable for the serious programmer."
https://www.stroustrup.com/quotes.html



https://www.stroustrup.com/4th.html

C++ System Programming



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

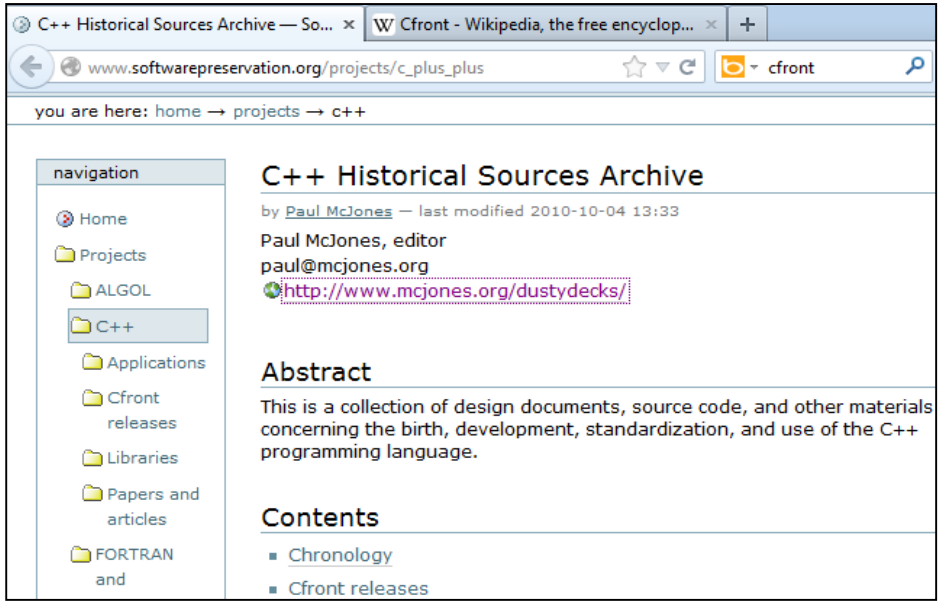
1979 October
First C with Classes (Cpre) running

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



https://www.softwarepreservation.org/projects/c_plus_plus

O caminho para padronização do C++

- 1990 (ANSI C++ Standard – baseado no “ARM”)
- 1998 (1ª versão do padrão ISO C++)
  https://www.iso.org/standard/25845.html
- 2003 (2ª versão do padrão ISO C++)
  https://www.iso.org/standard/38110.html
- 2011 (3ª versão do padrão ISO C++)
  https://www.iso.org/standard/50372.html
- 2014 (4ª versão do padrão ISO C++)
  https://www.iso.org/standard/64029.html
- 2017 (5ª versão do padrão ISO C++)
  https://www.iso.org/standard/68564.html
- 2020 (6ª versão do padrão ISO C++)
  https://www.iso.org/standard/79358.html
- **2023** (7ª 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 International Organization for Standardization

ISO C++ (WG 21)

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

https://isocpp.org/std/status

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

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 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	
consteval	export	private	thread_local	
constexpr	extern	protected	throw	

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

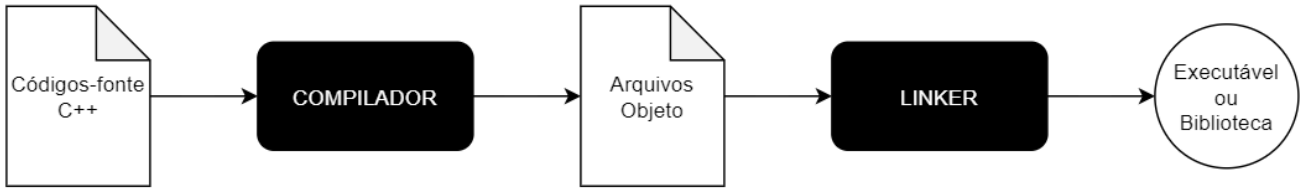
FERRAMENTAS

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

https://en.cppreference.com/w/cpp/compiler_support

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

https://github.com/CppCon/CppCon2023/blob/main/Presentations/B2B_Getting_Started_with_Cpp.pdf

Compiladores

Clang

GCC

libstdc++

Standard library implementation

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

https://github.com/CppCon/CppCon2023/blob/main/Presentations/B2B_Getting_Started_with_Cpp.pdf

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

https://github.com/CppCon/CppCon2023/blob/main/Presentations/B2B_Getting_Started_with_Cpp.pdf

Compiladores

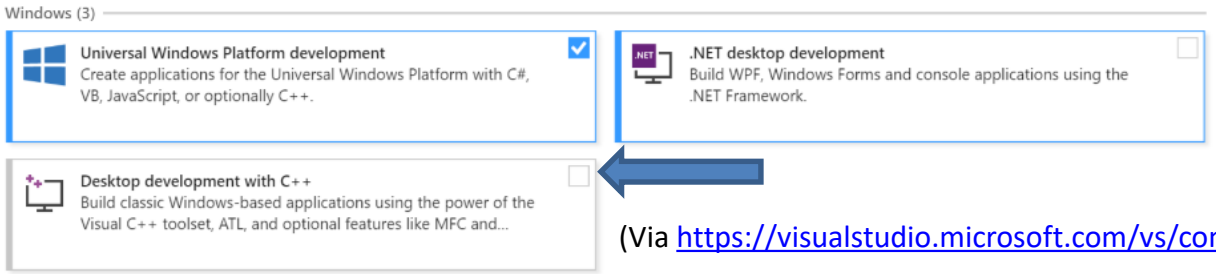
Clang

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**

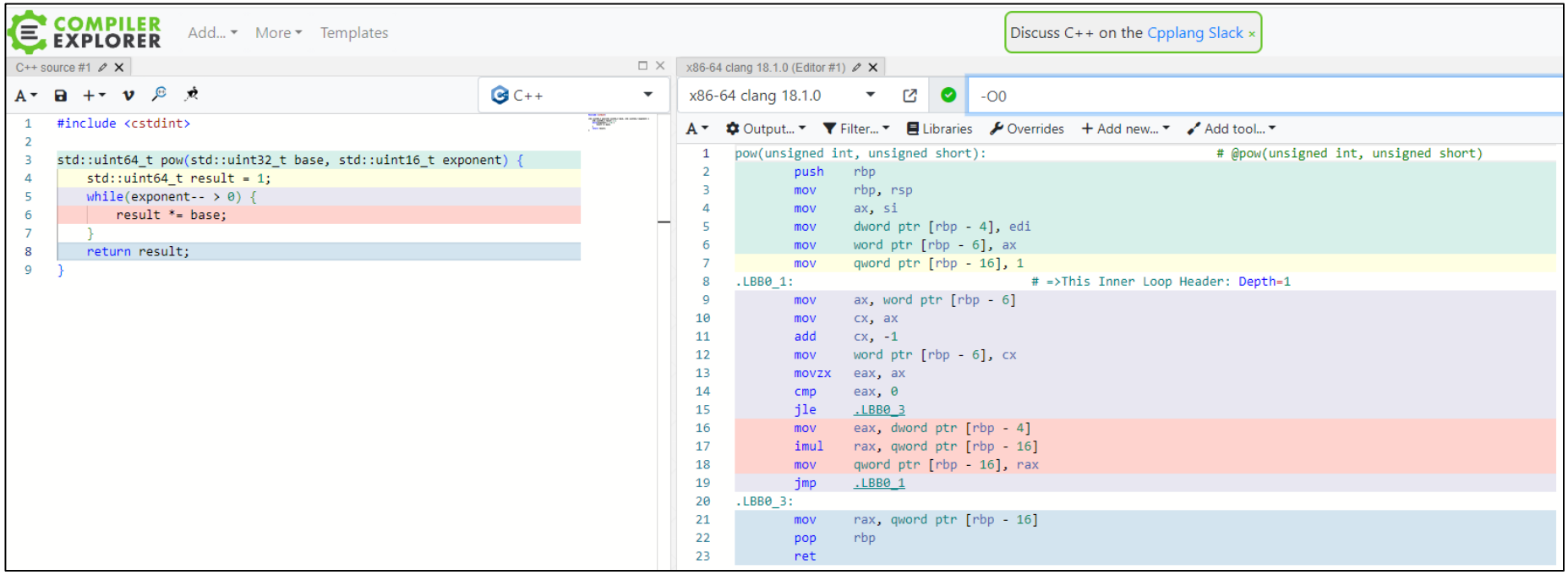
GCC (GNU Compiler Collection)
Ubuntu (via apt-get): **sudo apt-get install build-essential**
macOS (via MacPorts - XCode e XCode Command Line Tools devem estar instalados): **sudo port install gcc**
Window (via MSYS2): **pacman -S --needed base-devel mingw-w64- ucrt-x86_64-toolchain**



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

https://github.com/CppCon/CppCon2023/blob/main/Presentations/B2B_Getting_Started_with_Cpp.pdf

Compiler Explorer

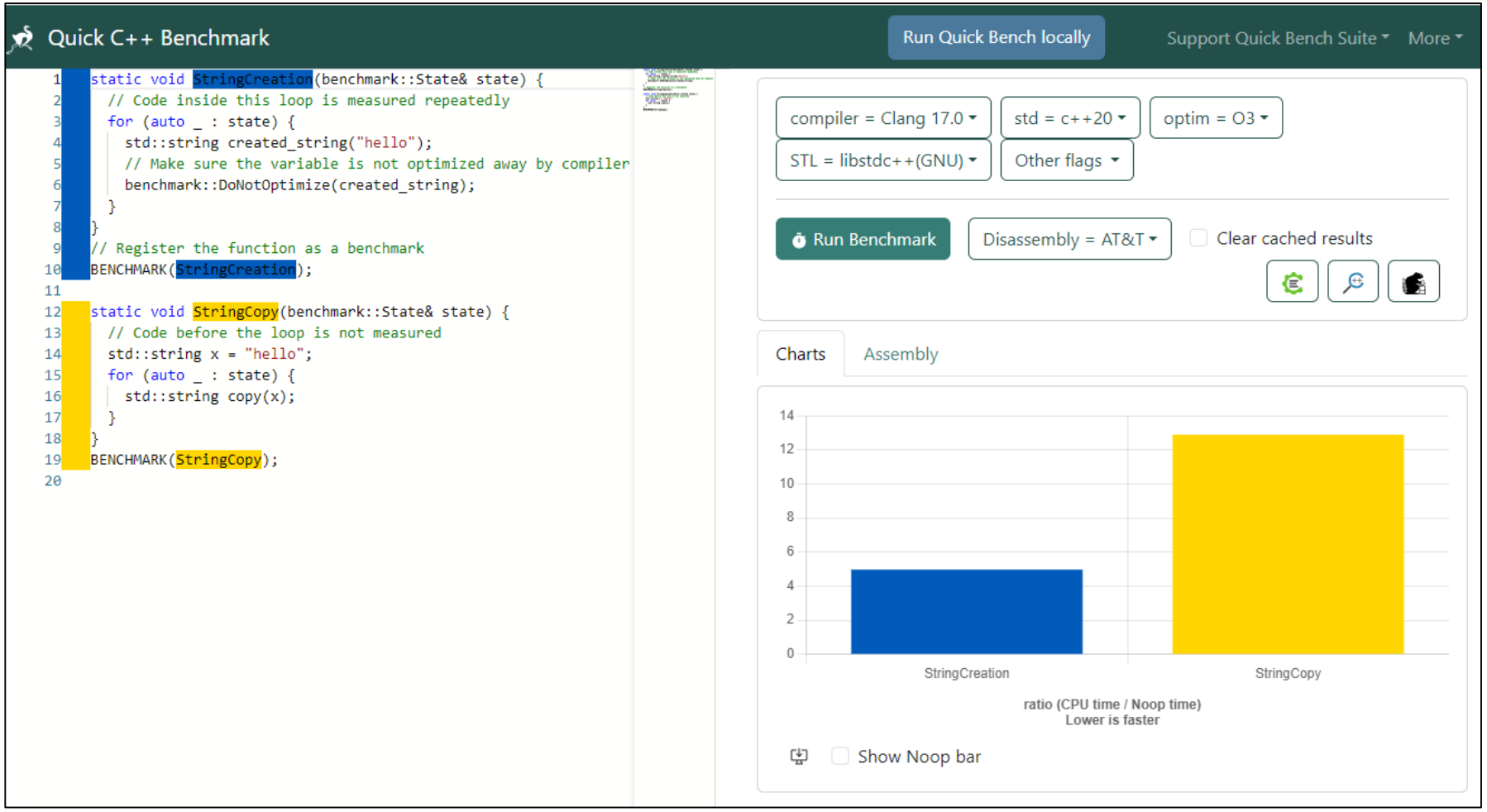



```cpp
#include <cstdint>

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

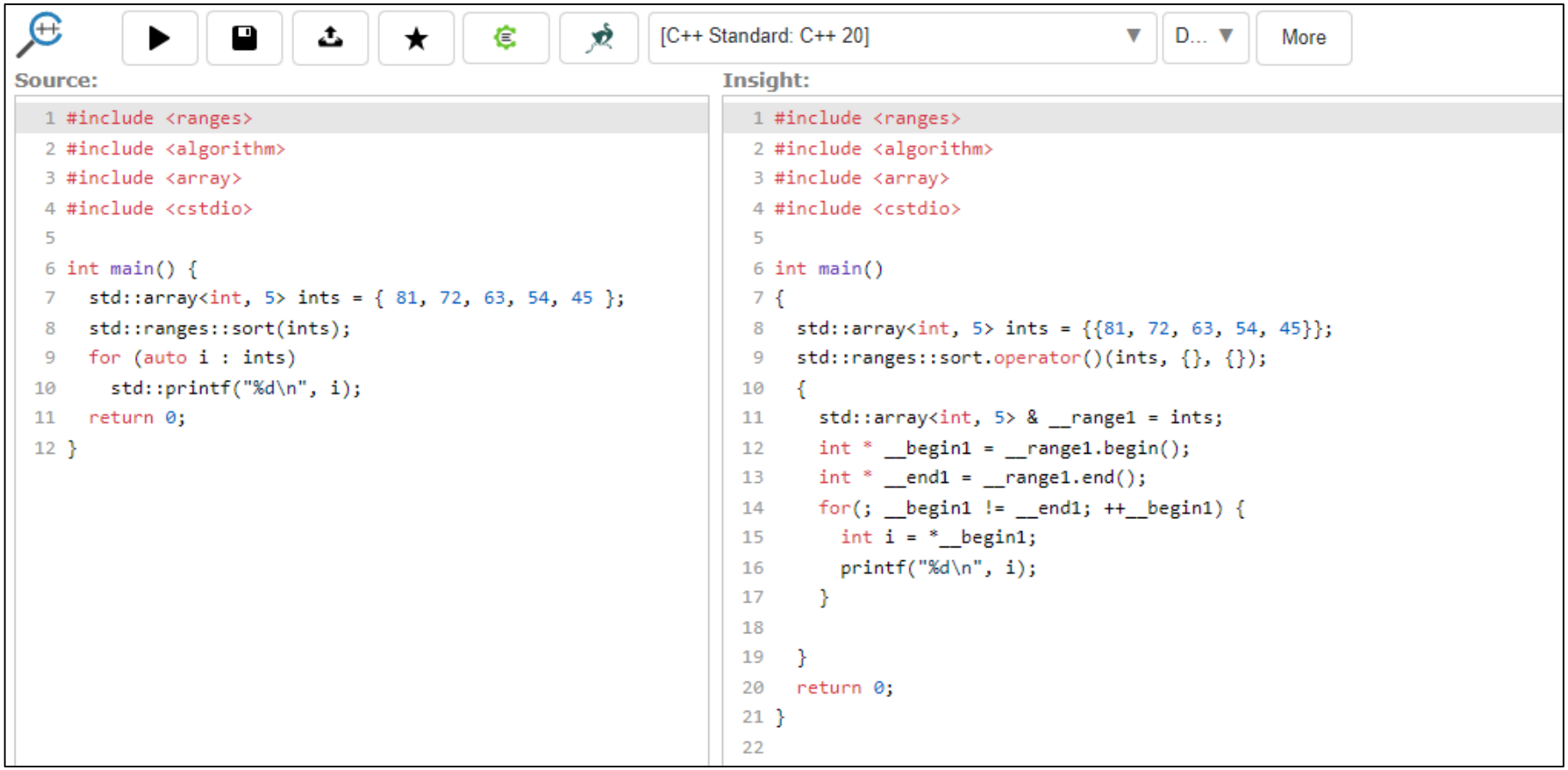
https://godbolt.org/

Quick C++ Benchmark



https://quick-bench.com/

C++ Insights




```
#include <ranges>
#include <algorithm>
#include <array>
#include <cstdio>

int main() {
  std::array<int, 5> ints = { 81, 72, 63, 54, 45 };
  std::ranges::sort(ints);
  for (auto i : ints)
    std::printf("%d\n", i);
  return 0;
}
```

```
#include <ranges>
#include <algorithm>
#include <array>
#include <cstdio>

int main()
{
  std::array<int, 5> ints = {{81, 72, 63, 54, 45}};
  std::ranges::sort.operator()(ints, {}, {});
  {
    std::array<int, 5> & __range1 = ints;
    int * __begin1 = __range1.begin();
    int * __end1 = __range1.end();
    for(; __begin1 != __end1; ++__begin1) {
      int i = *__begin1;
      printf("%d\n", i);
    }

  }
  return 0;
}

```

https://cppinsights.io/

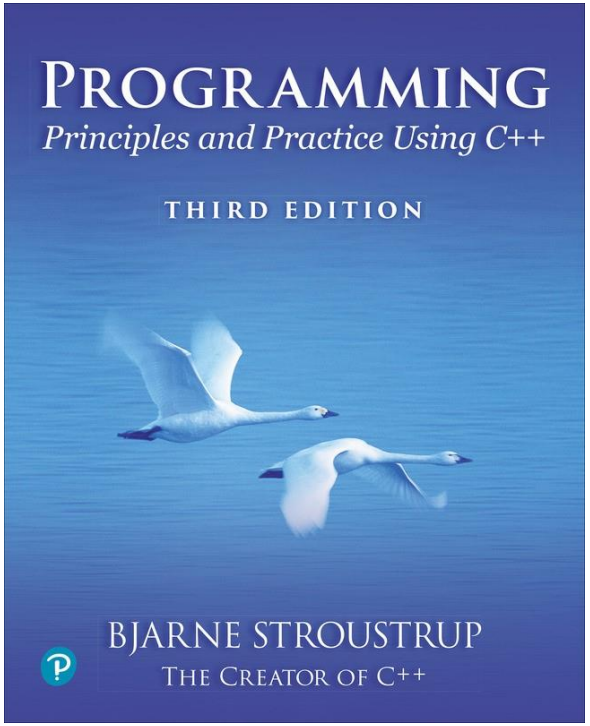
Outras ferramentas

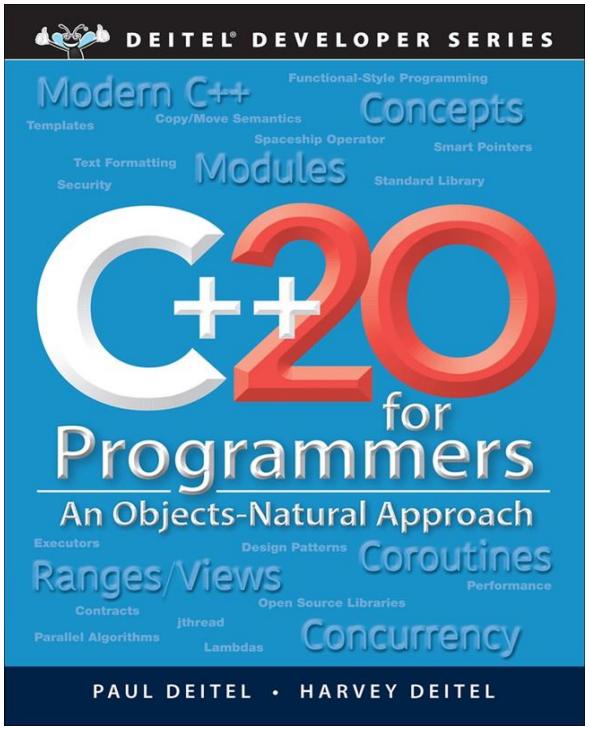
- IDEs
  - Visual Studio, Visual Studio Code, Qt Creator, CLion, …
- Build
  - CMake
    - make, msbuild, ninja
- Gerenciador de Pacotes
  - Vcpkg, Conan
- Análise Estática
  - Microsoft C/C++ Code Analysis, Clang Static Analyzer, clang-tidy, CppCheck, PVS-Studio
- Análise Dinâmica
  - Sanitizers
  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

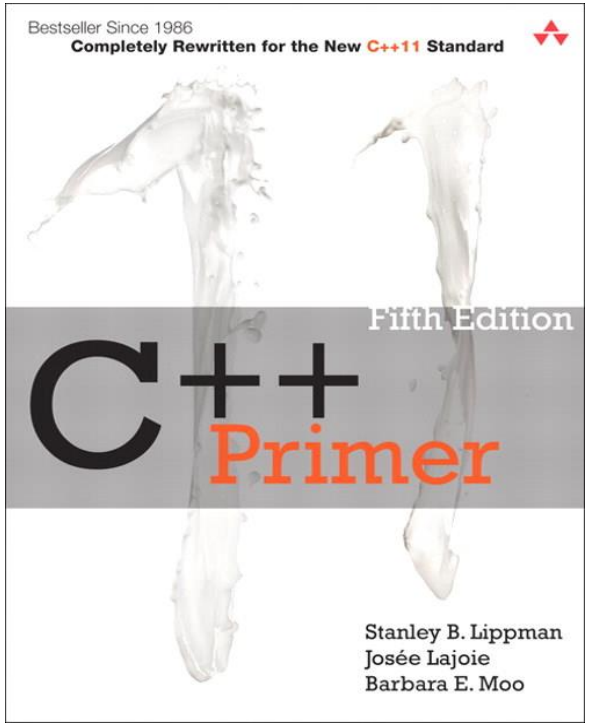
Getting Started with C++ - Michael Price - CppCon 2023: https://www.youtube.com/watch?v=NReDubvNjRg

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

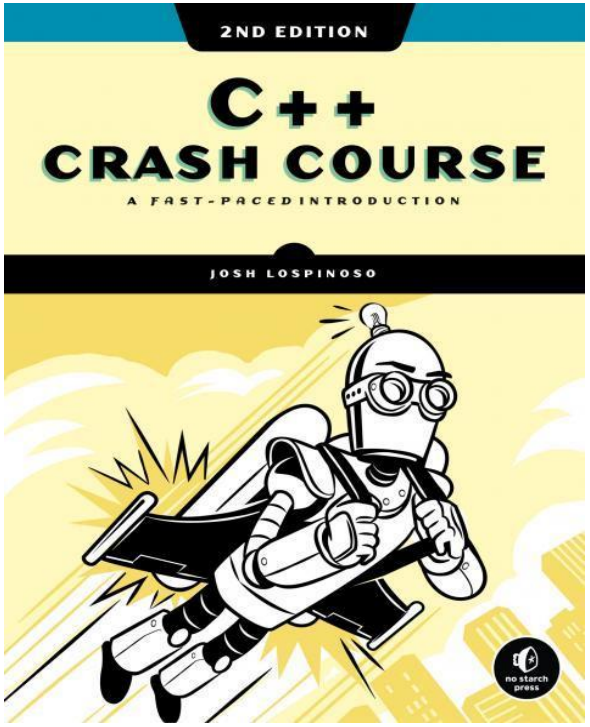
Eu quero aprender C++!
Por onde começar?

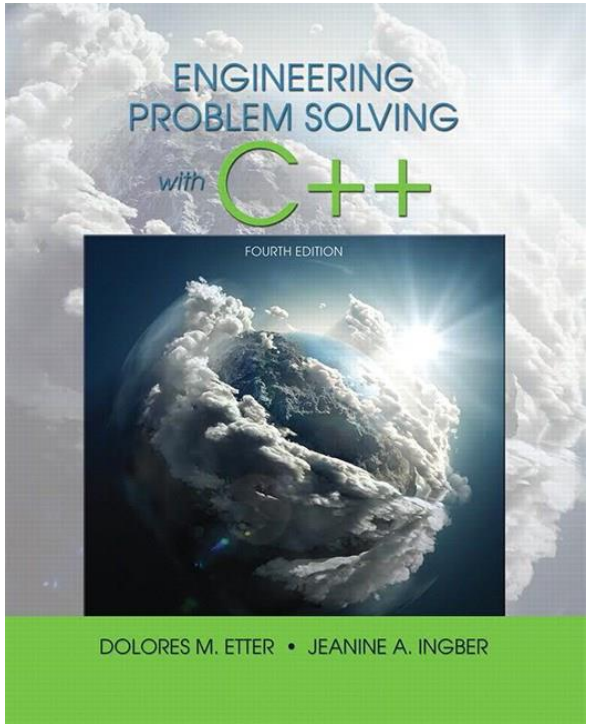


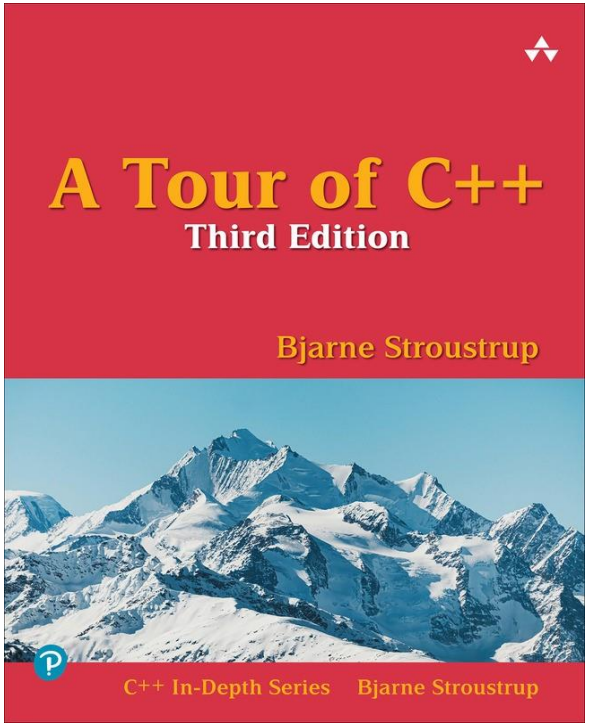




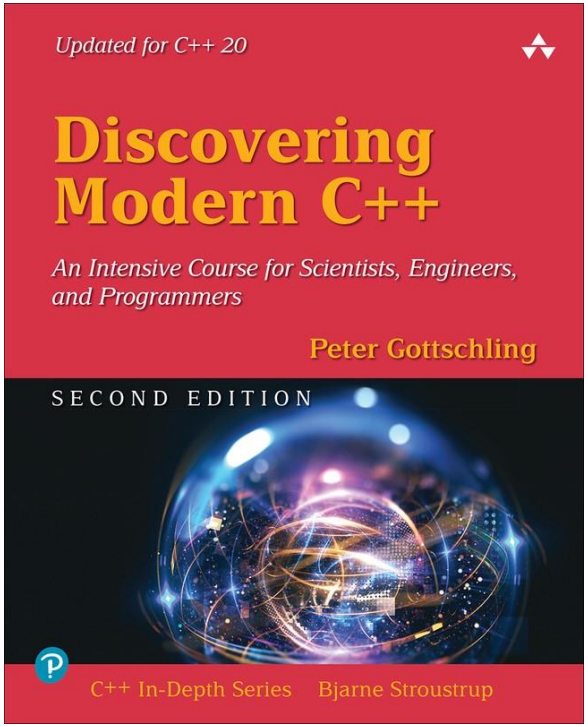
Eu quero aprender C++!
Por onde começar?

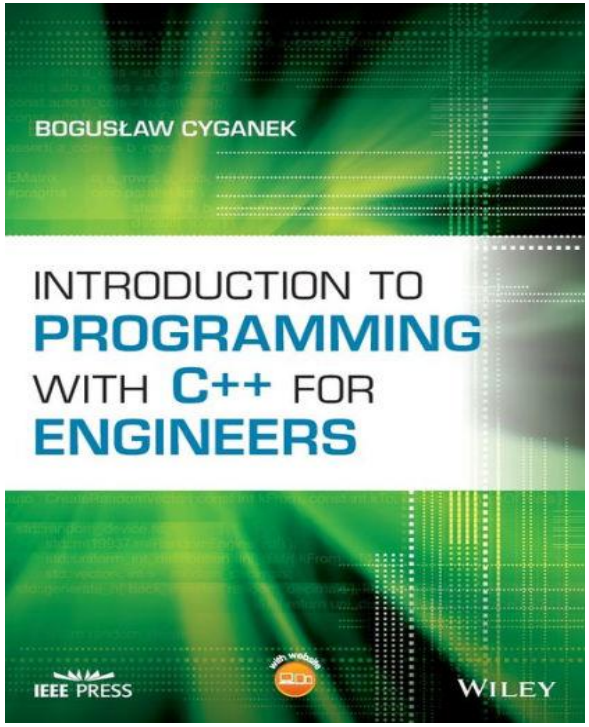


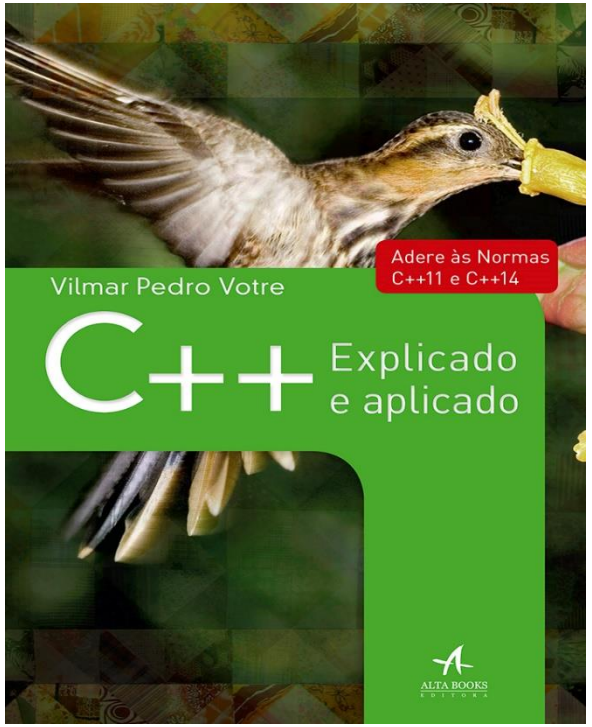




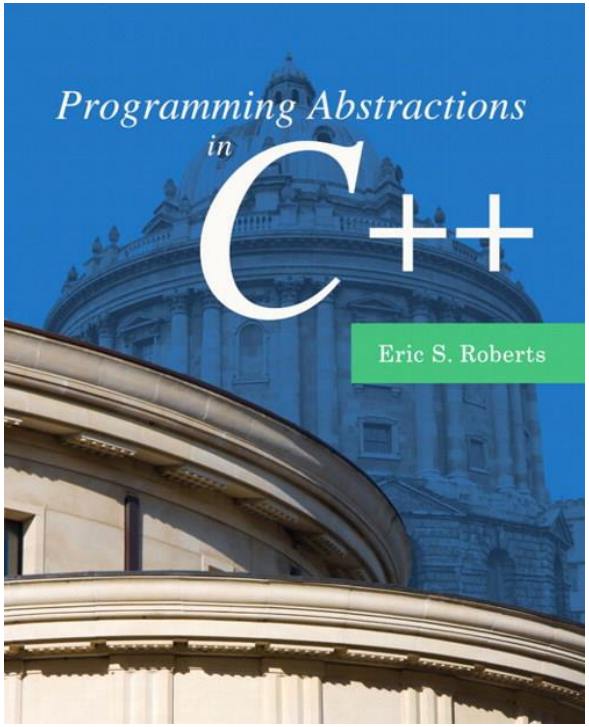
Eu quero aprender C++!
Por onde começar?

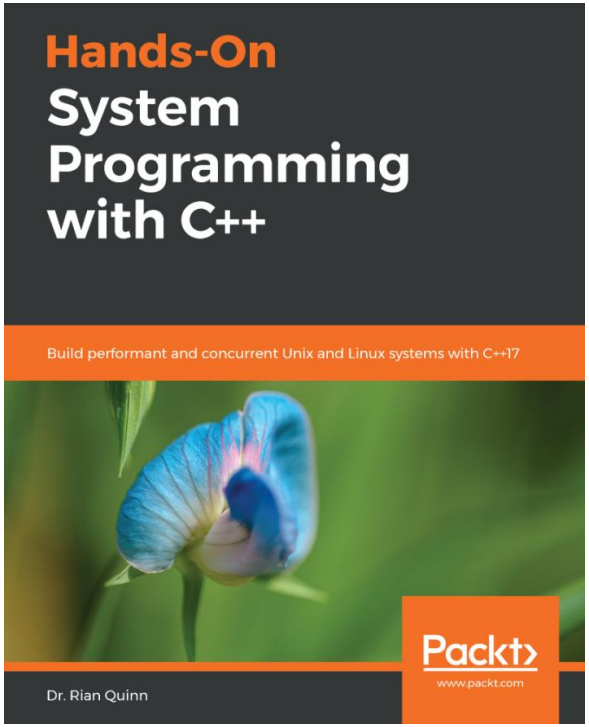


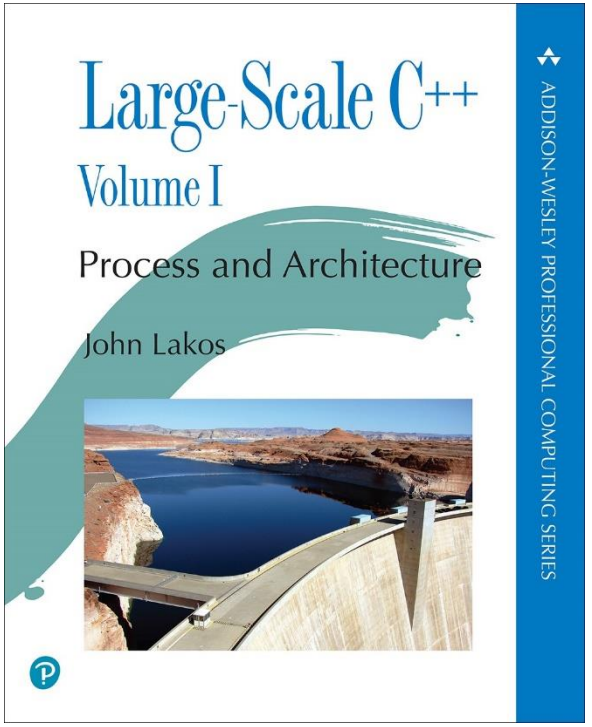




Eu quero aprender C++!
Por onde começar?

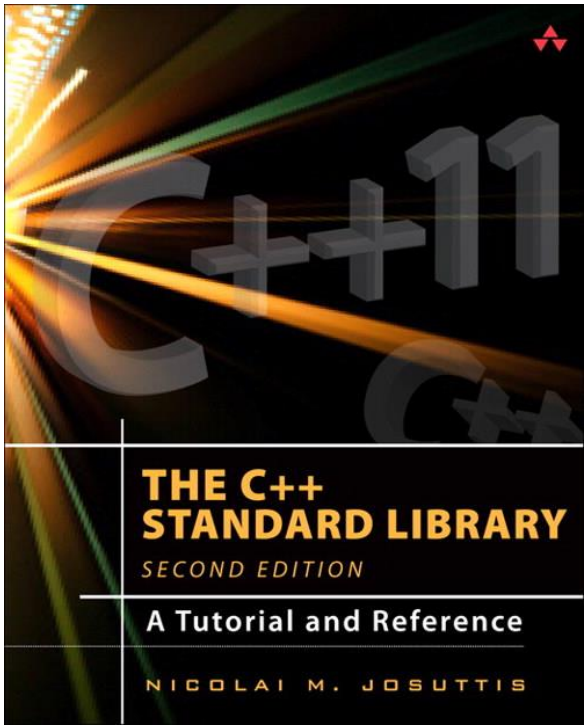


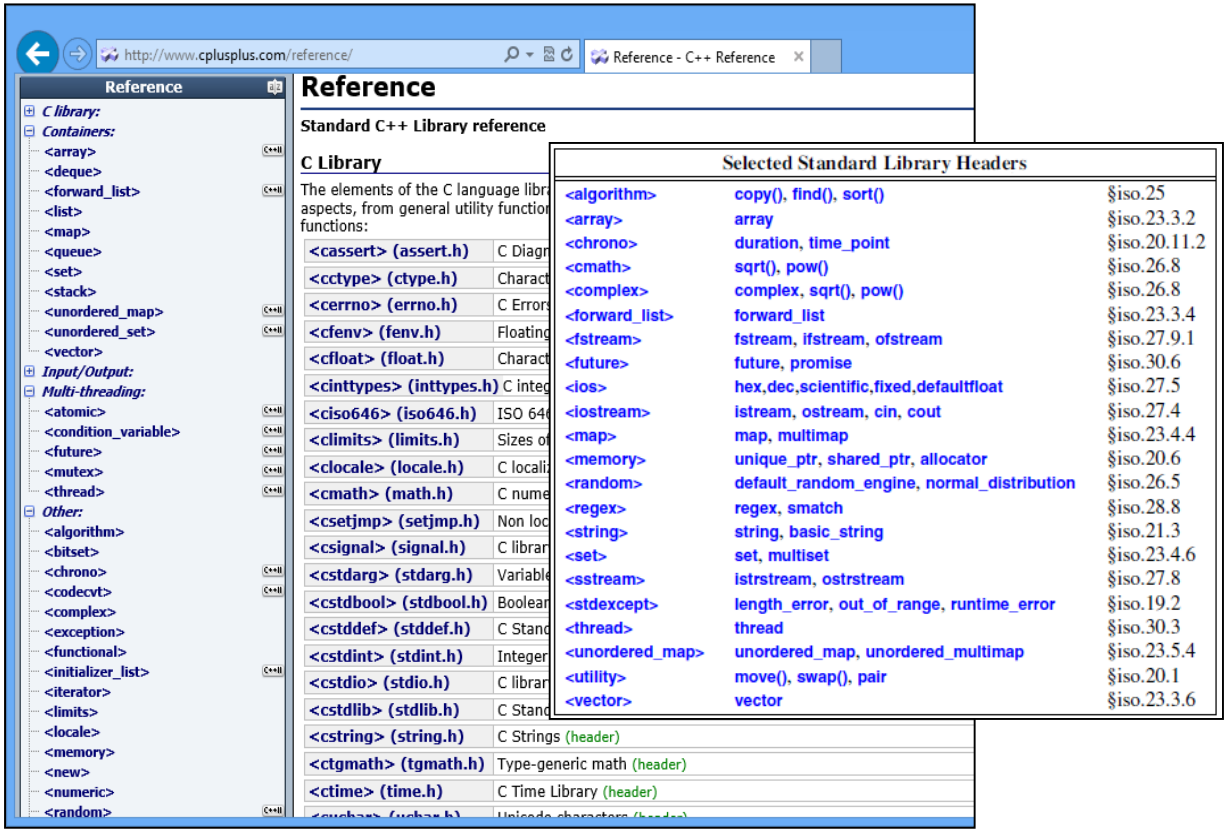




Eu quero aprender C++!
O que preciso saber?

- Standard C++ Library





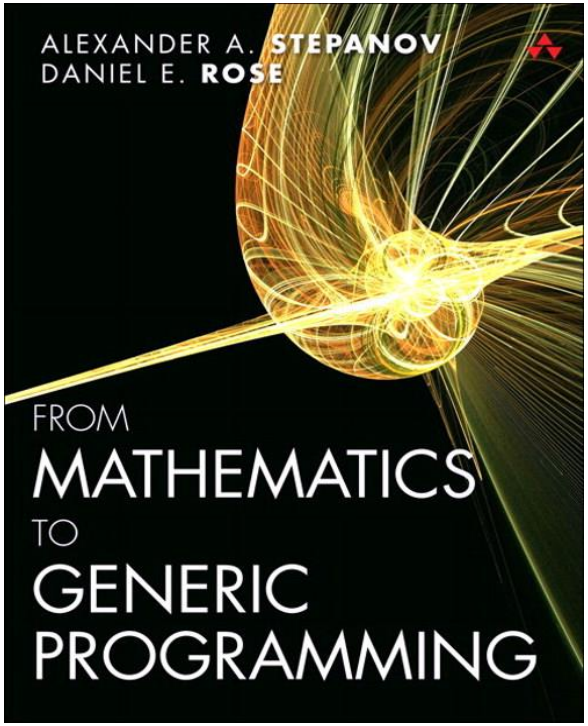
http://www.cplusplus.com

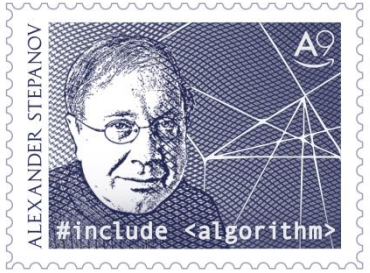
Eu quero aprender C++!
O que preciso saber?

- 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

http://www.fm2gp.com/slides/FM2GP_Course_Slides_Pt1.pdf

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

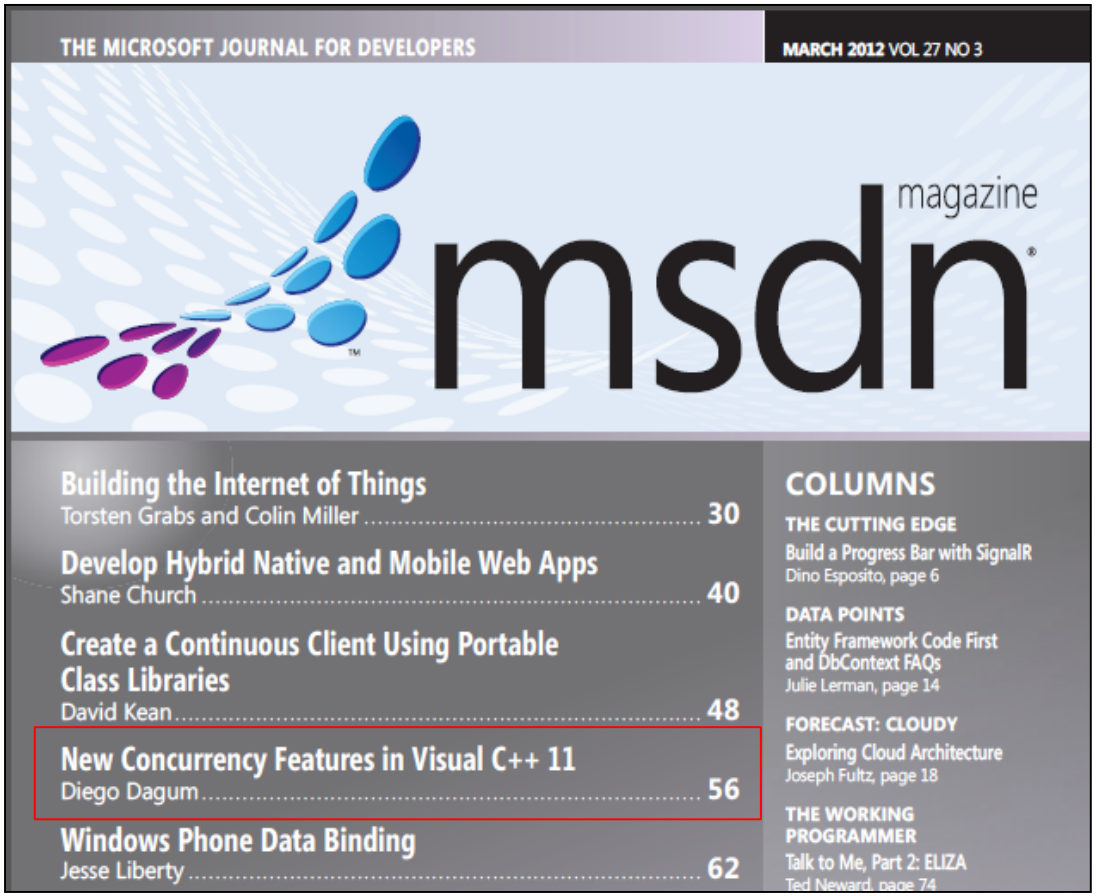
```
template <class ForwardIterator, class T>
  bool binary_search (ForwardIterator first, ForwardIterator last,
                      const T& val);
```
default (1)

```
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 }
```

http://www.cplusplus.com/reference/algorithm/

Eu quero aprender C++!
O que preciso saber?

- Programação Concorrente



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:

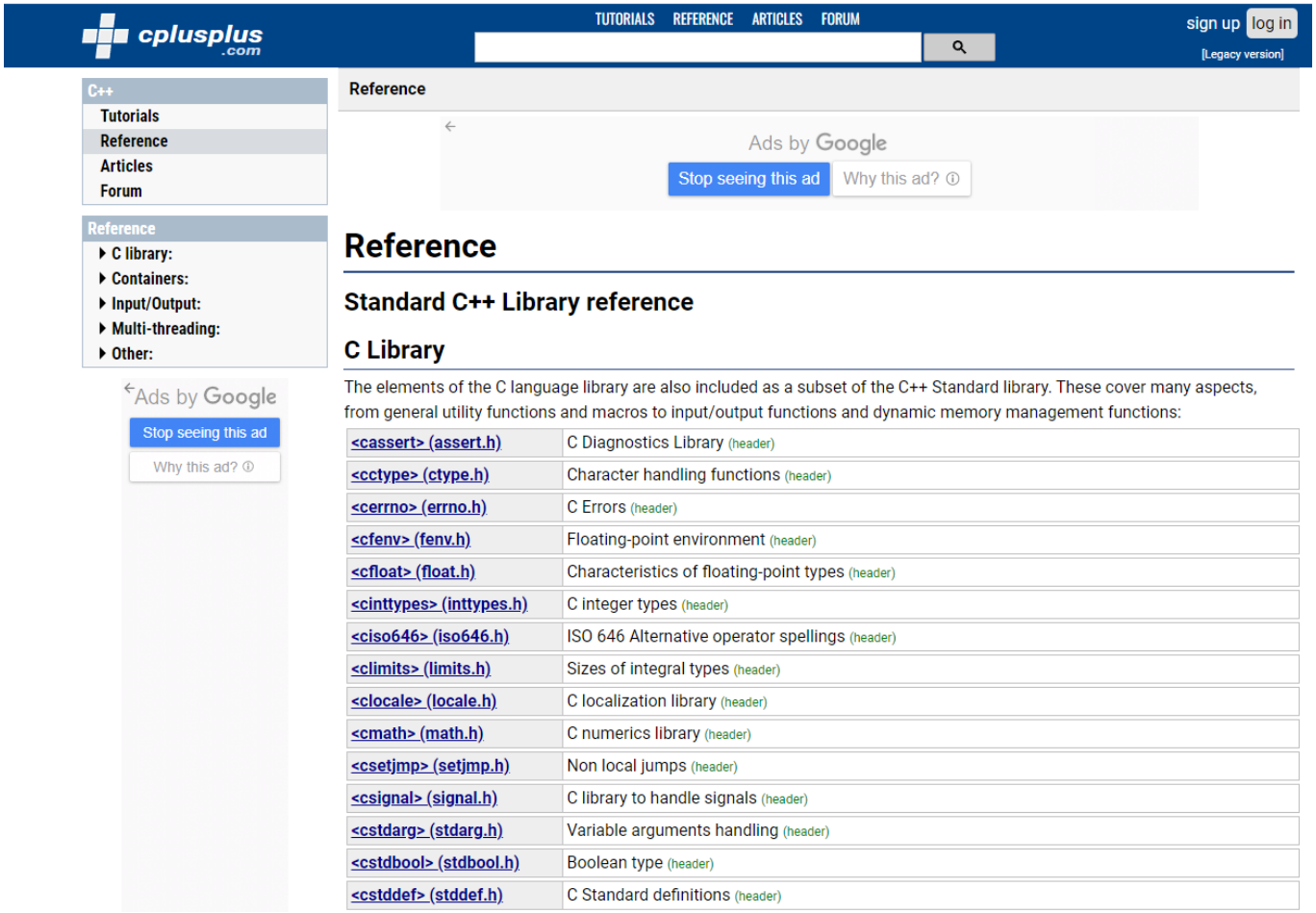
code.msdn.microsoft.com/mag201203CPP

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*

http://msdn.microsoft.com/en-us/magazine/hh852594.aspx

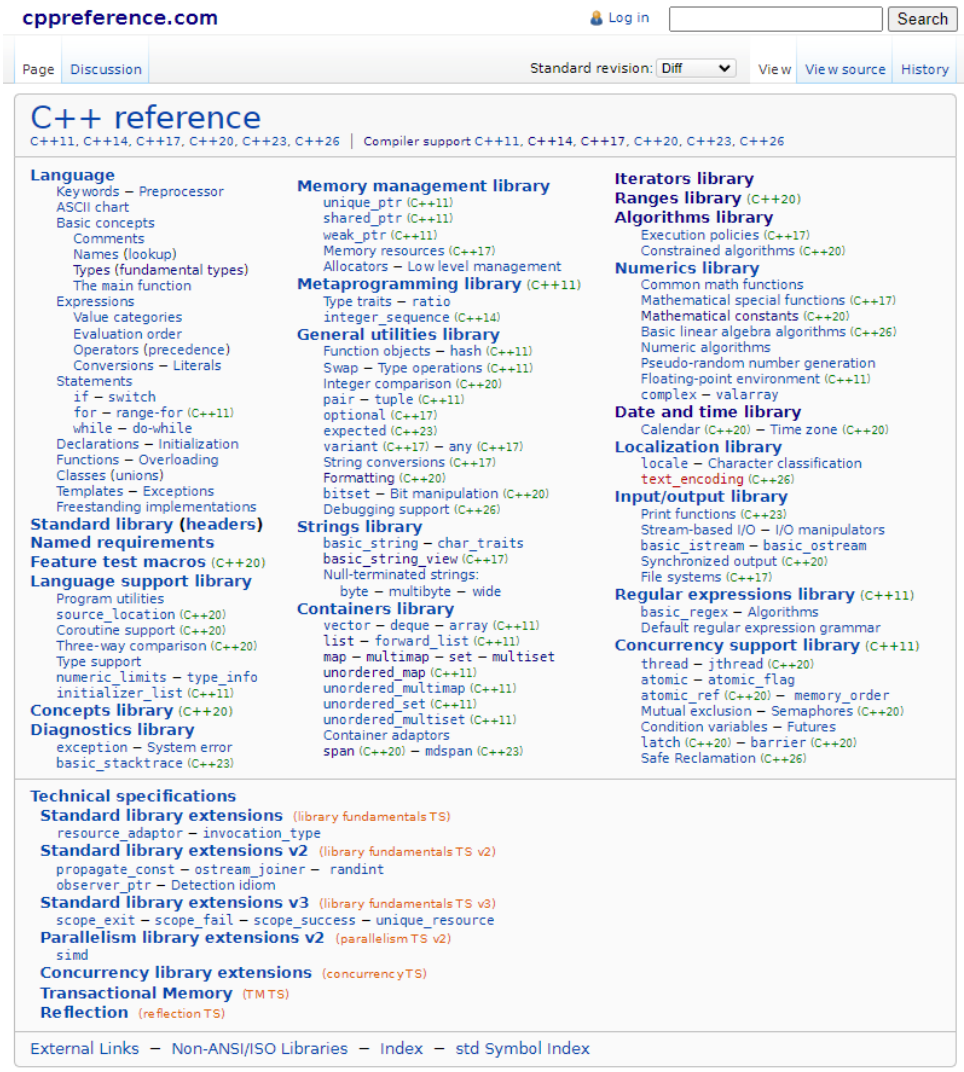
Eu quero aprender C++!
O que preciso saber?



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 (C++11)
- basic_regex – Algorithms
- Default regular expression grammar

Concurrency support library (C++11)
- 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)

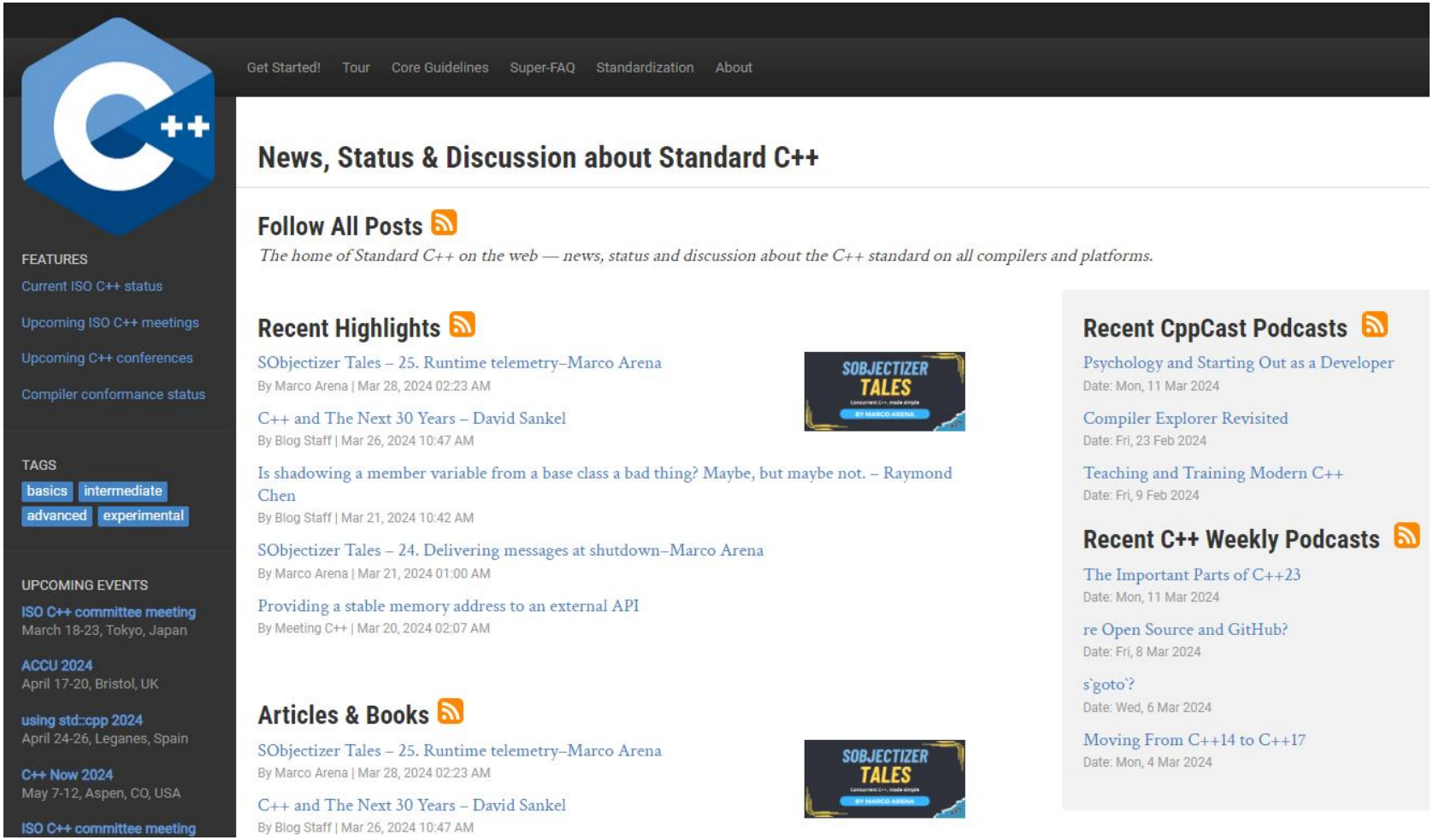
Transactional Memory (TM TS)

Reflection (reflection TS)

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

https://en.cppreference.com

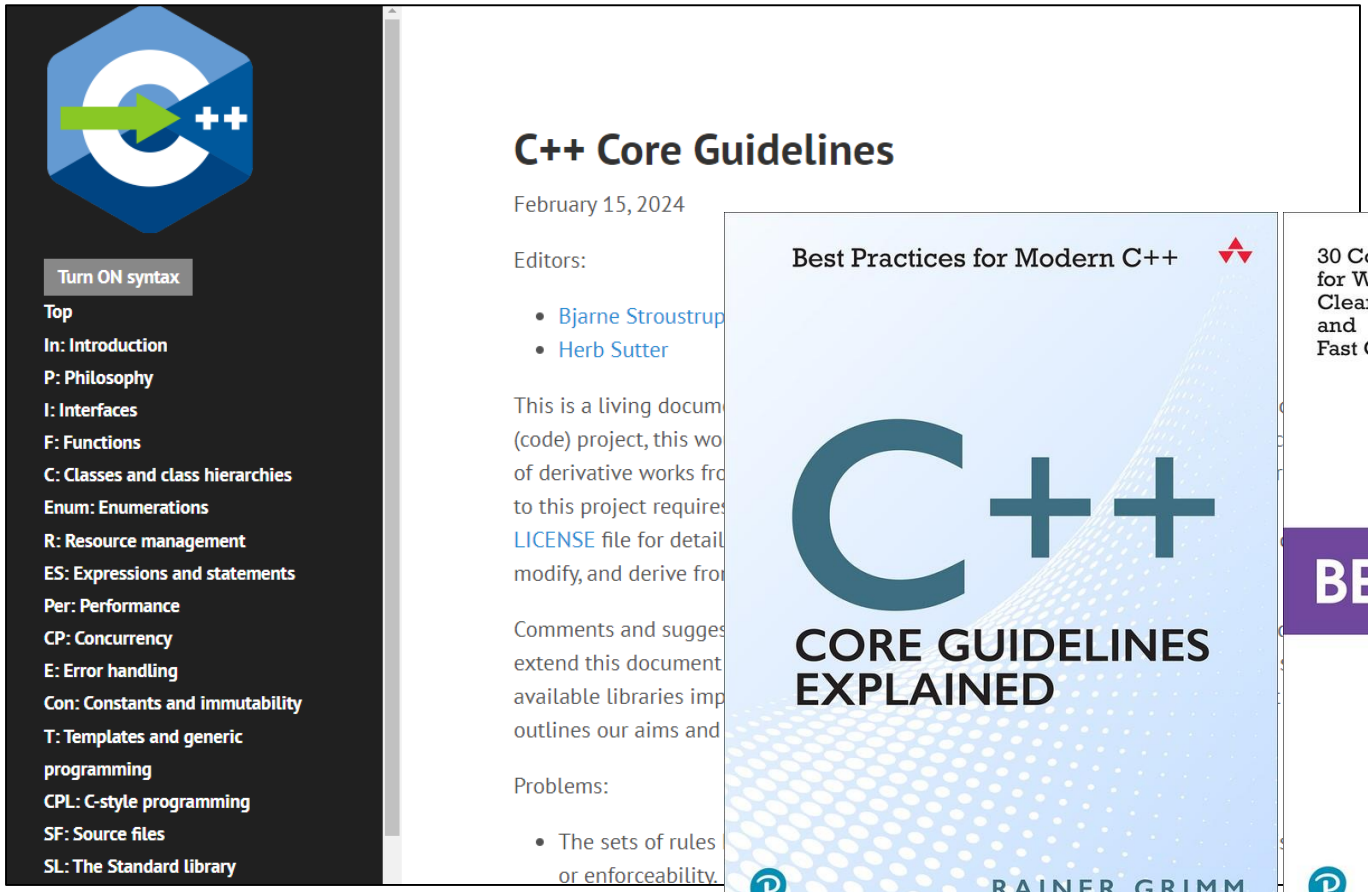
Eu quero aprender C++!
O que preciso saber?

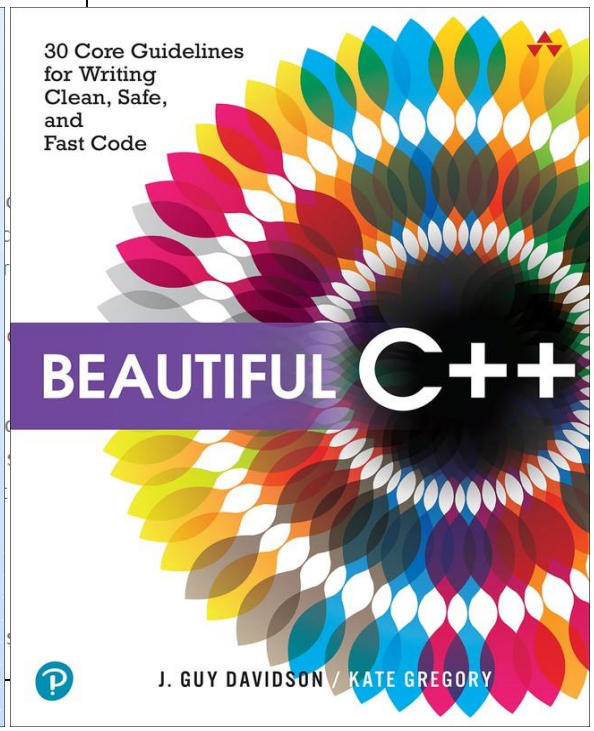


https://isocpp.org/

Eu quero aprender C++!
O que preciso saber?

- Core Guidelines





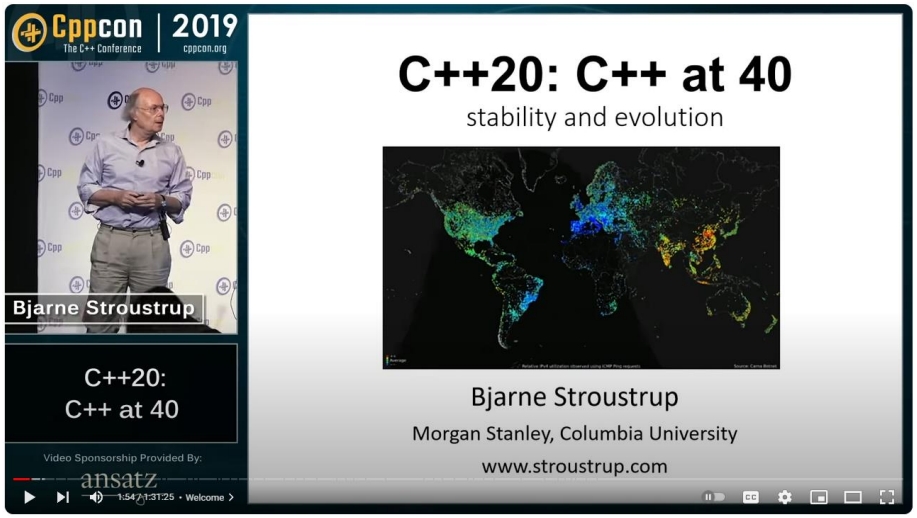
https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

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 po ...more



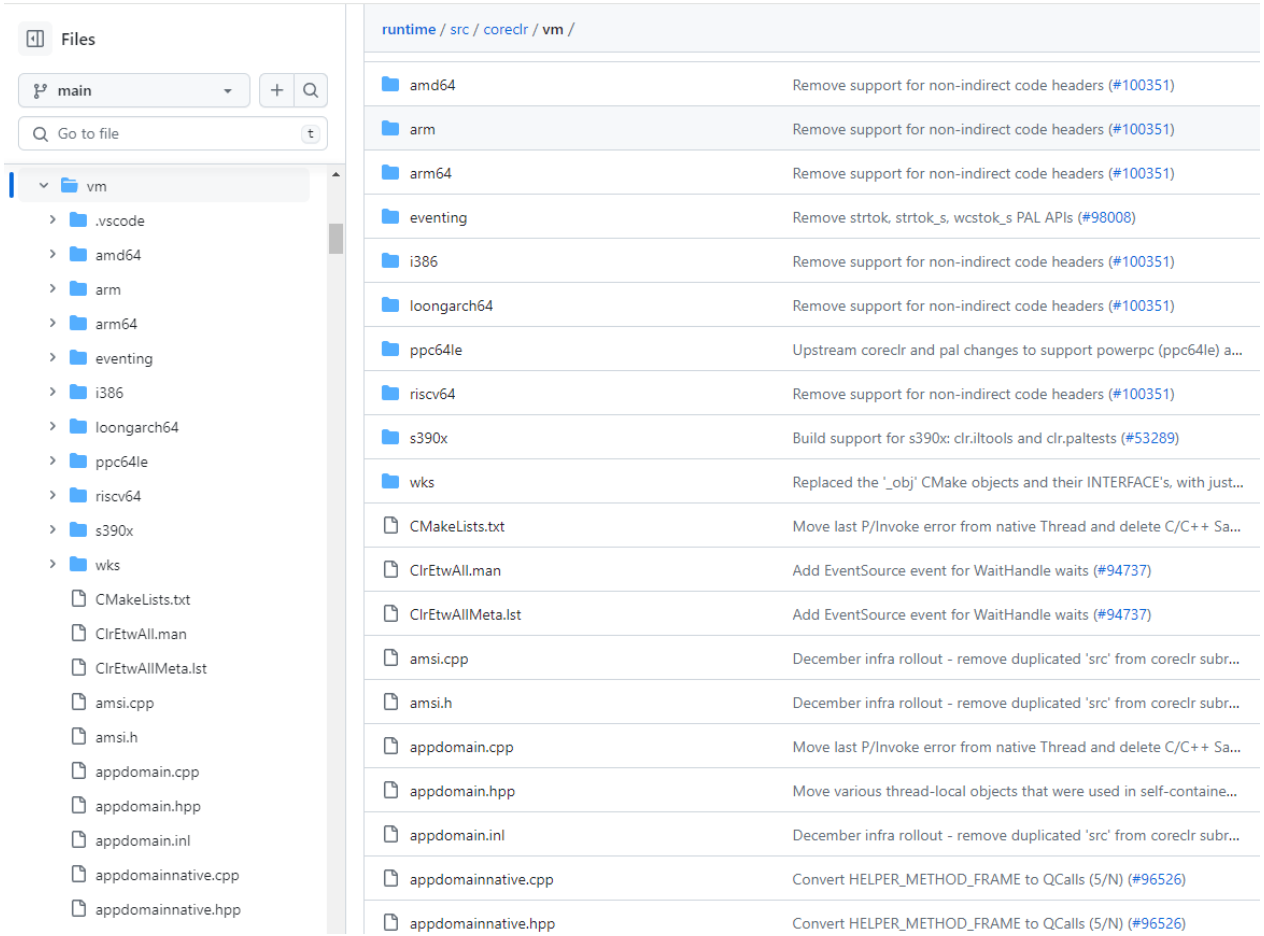
https://www.youtube.com/@CppCon

TECNOLOGIAS QUE UTILIZAM C++

Quem depende de C++?

- Máquina Virtual

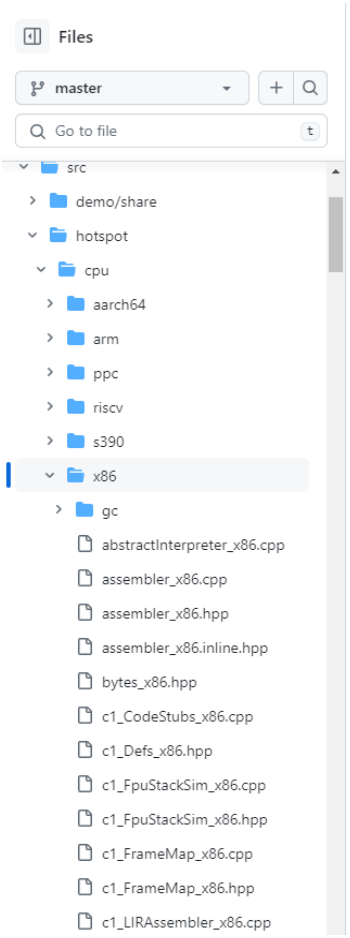
.NET

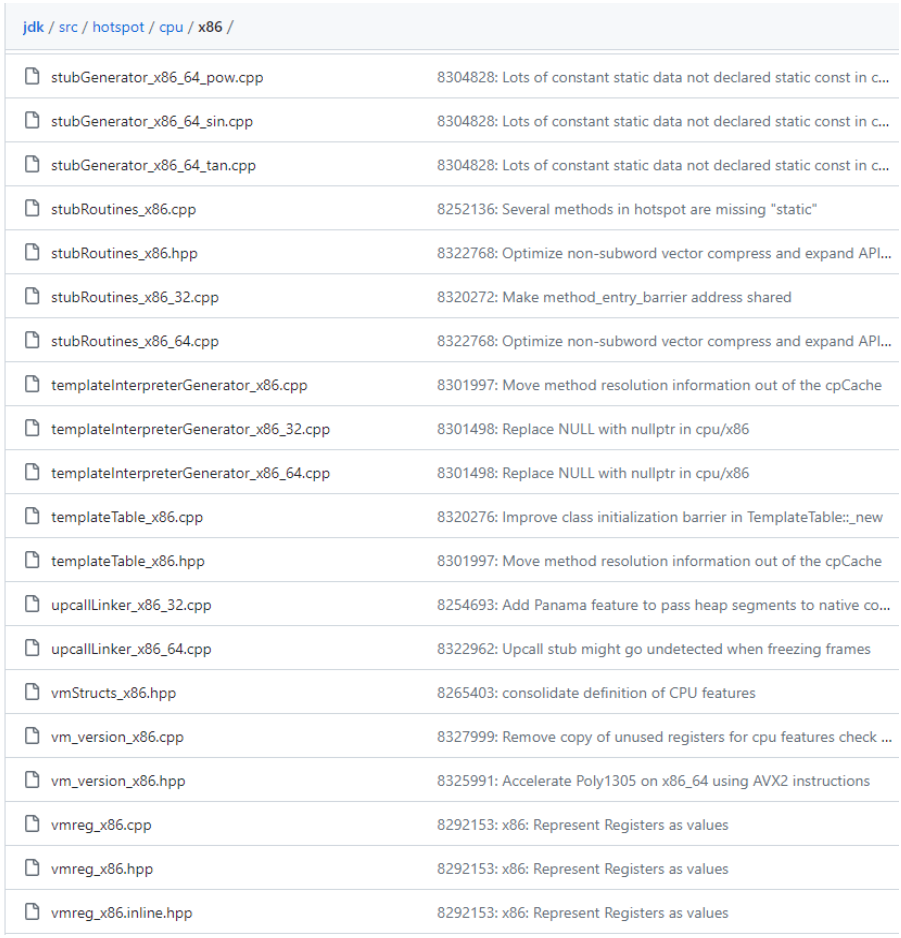


https://github.com/dotnet/runtime/tree/main/src/coreclr/vm

Quem depende de C++?

- Máquina Virtual



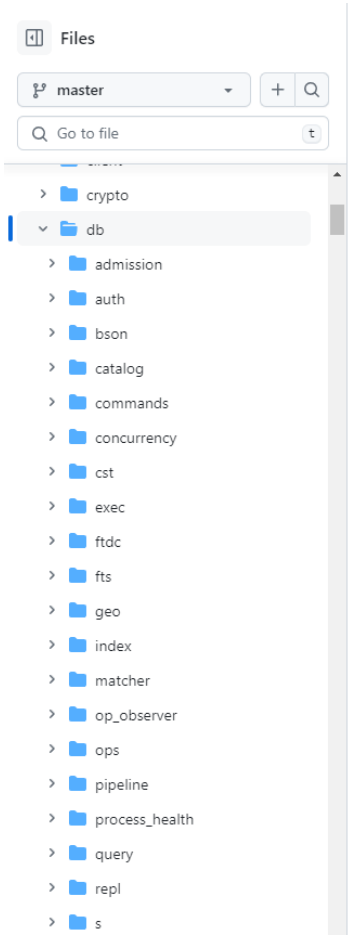


https://github.com/openjdk/jdk/tree/master/src/hotspot

Quem depende de C++?

- Banco de Dados (orientado a documentos)

MongoDB®



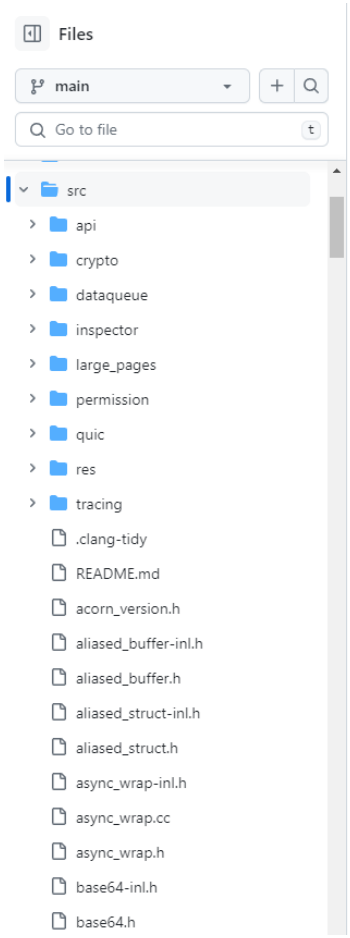
mongo / src / mongo / db /

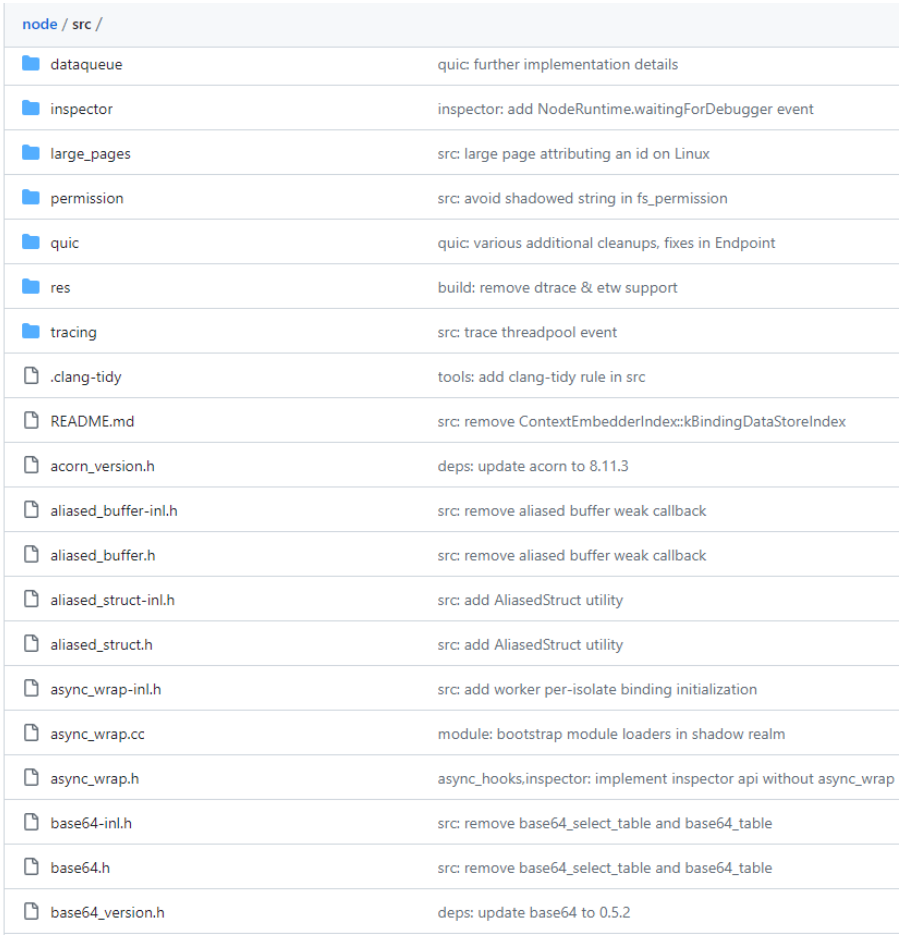
client.cpp	SERVER-87015 Optimize registration of operations in `MozJSScript...
client.h	SERVER-84765 Return the correct port at host field when embedd...
client_context_test.cpp	SERVER-81830 Create Clients with Service, not ServiceContext
client_out_of_line_executor.cpp	SERVER-87974 Enable modernize-use-override 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



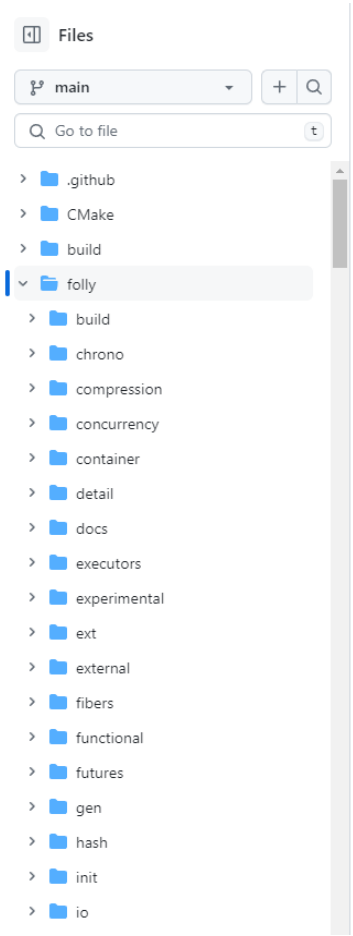


https://github.com/mongodb/mongo/tree/master/src/mongo/db

Quem depende de C++?

- Rede Social

Meta
facebook



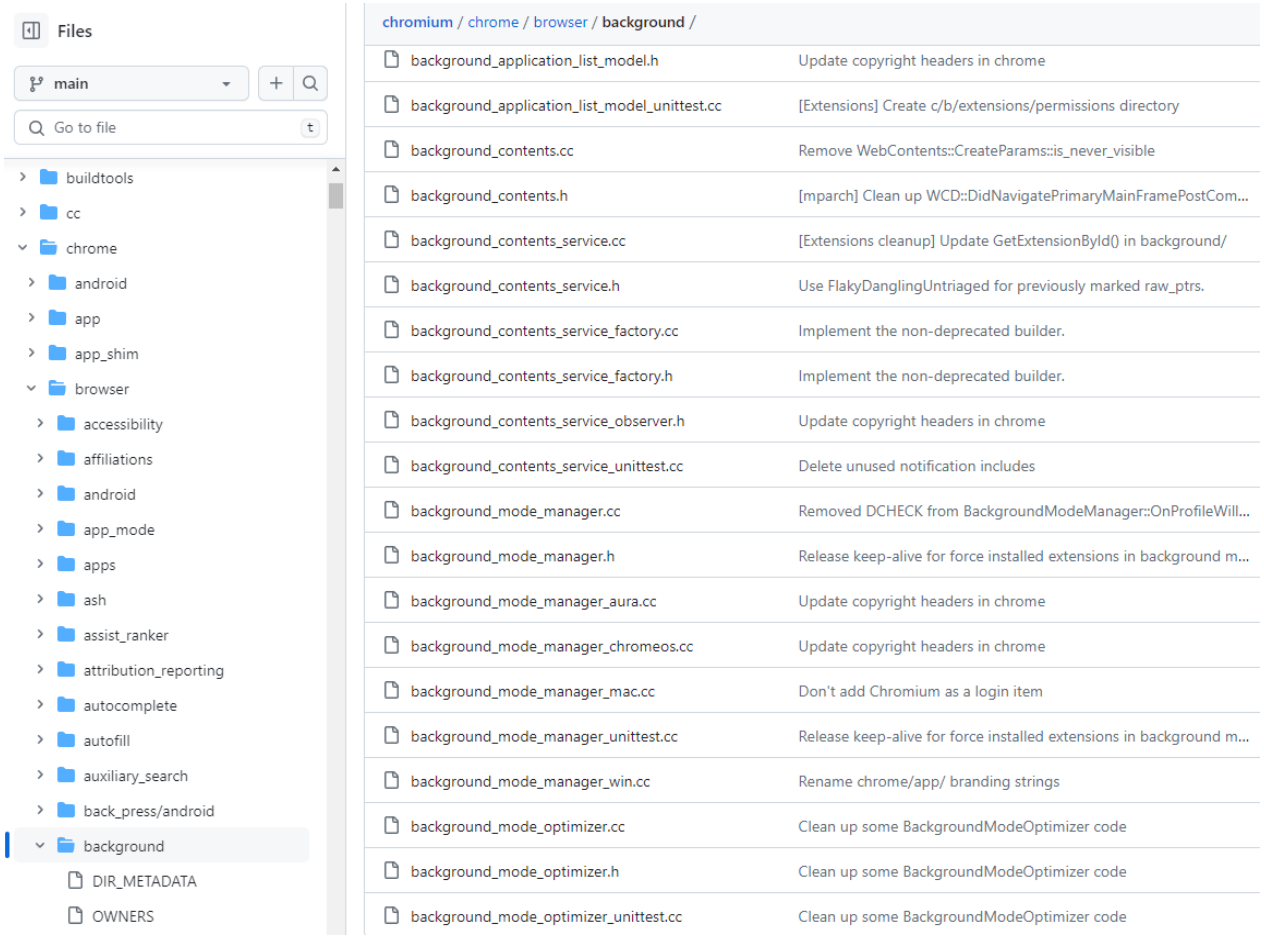
folly / folly /

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++?

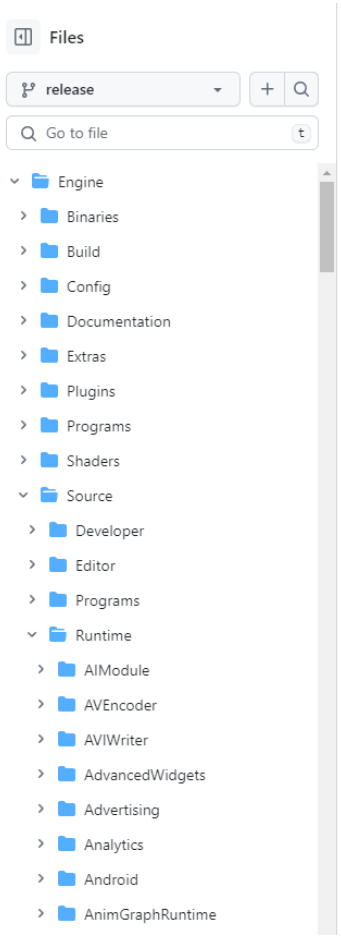
- Navegador Web

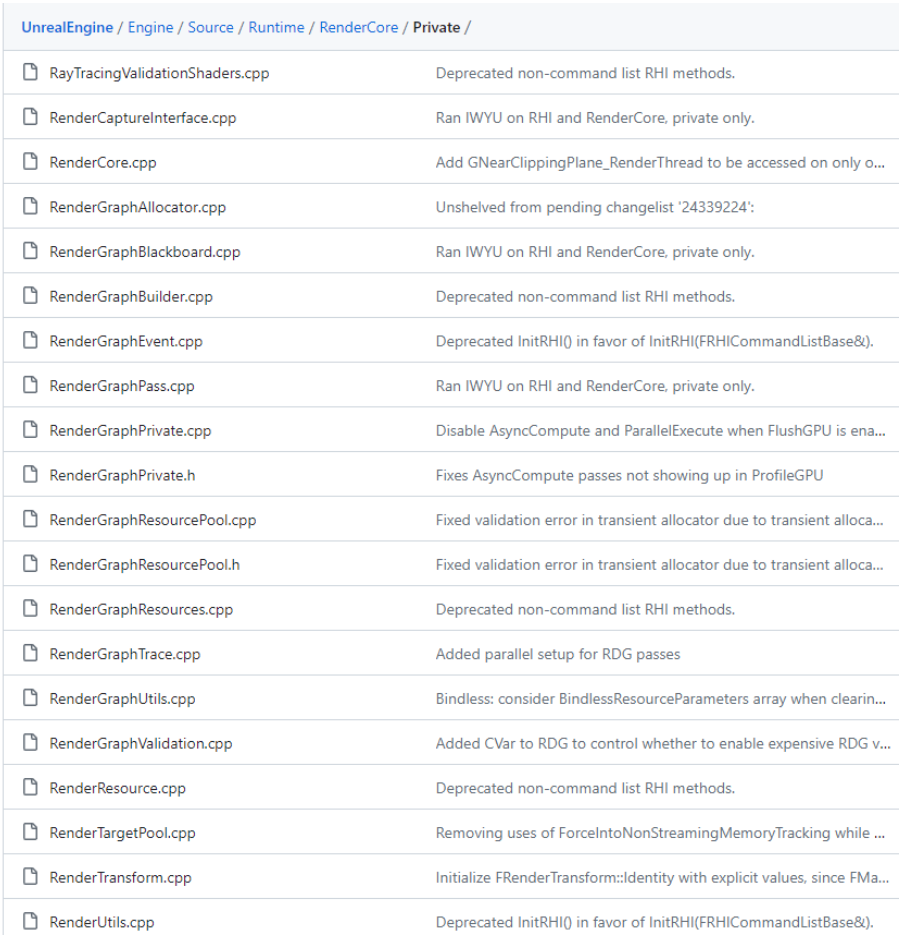


https://github.com/chromium/chromium/tree/main/chrome

Quem depende de C++?

- Motor de jogos

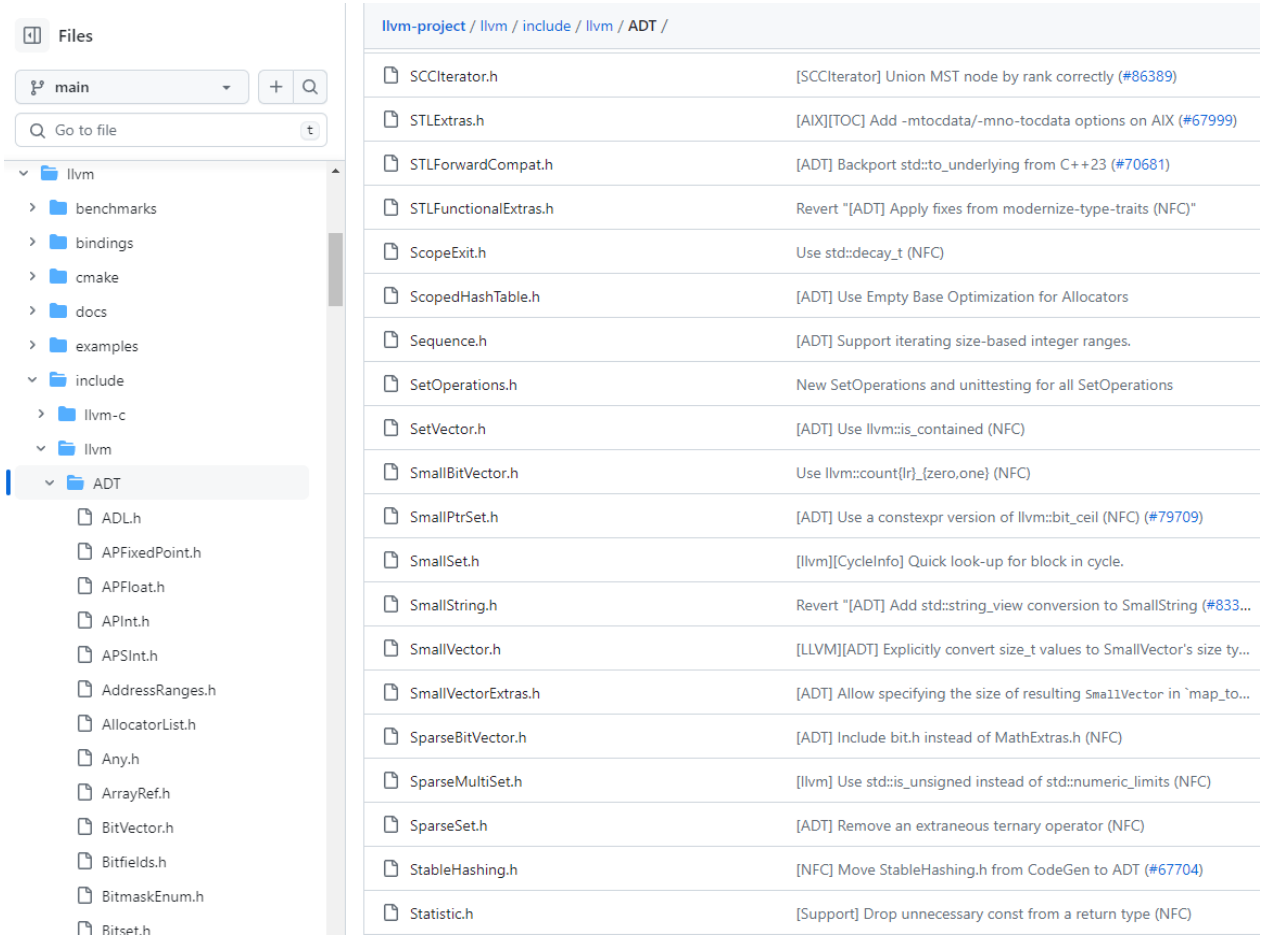




https://github.com/EpicGames/UnrealEngine/tree/release/Engine

Quem depende de C++?

- Infraestrutura de compiladores



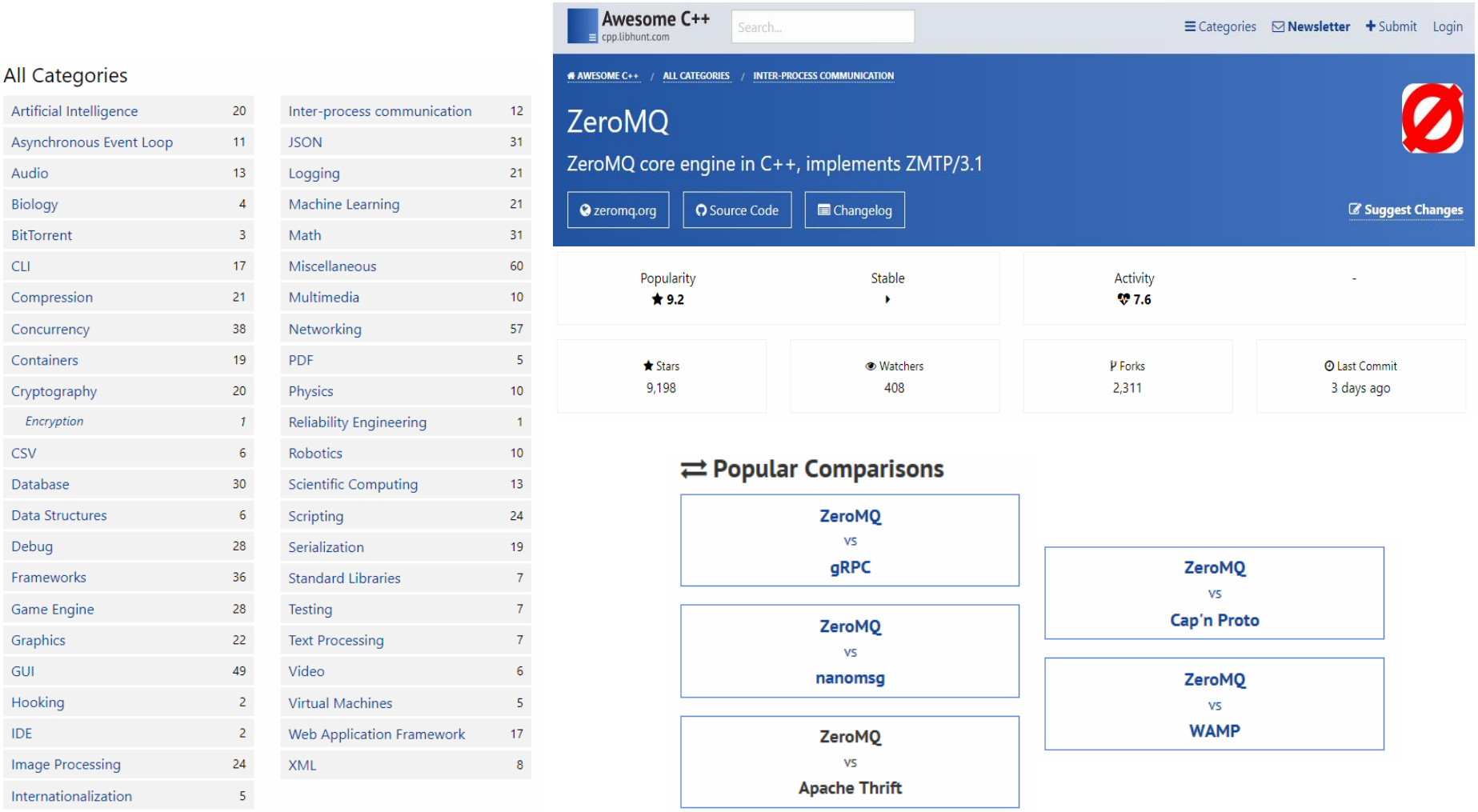
https://github.com/llvm/llvm-project/tree/main/llvm

BIBLIOTECAS PARA C++

Awesome C++

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

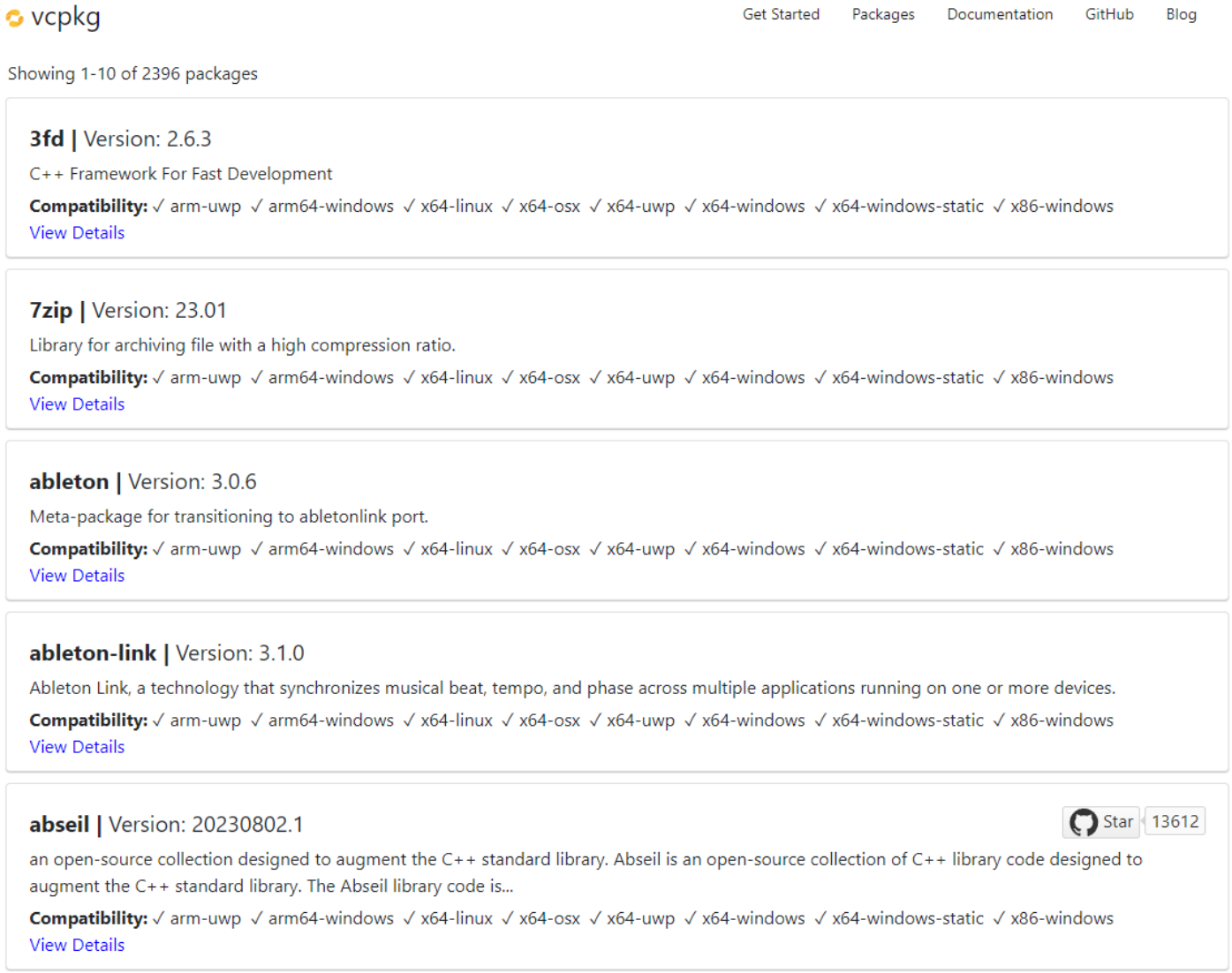
https://cpp.libhunt.com/



Vcpkg

Gerenciador de Pacotes

https://vcpkg.io/en/packages



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.MultiIndex
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.IOStreams
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

ZeroMQ

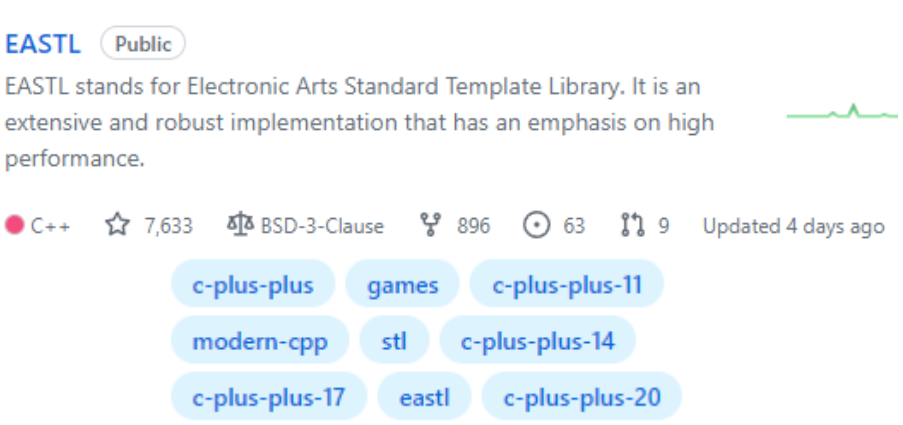
http://zeromq.org

- *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.



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

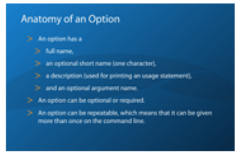
Build application servers in C++ that talk to SQL databases, MongoDB or Redis.

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.



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 Creator

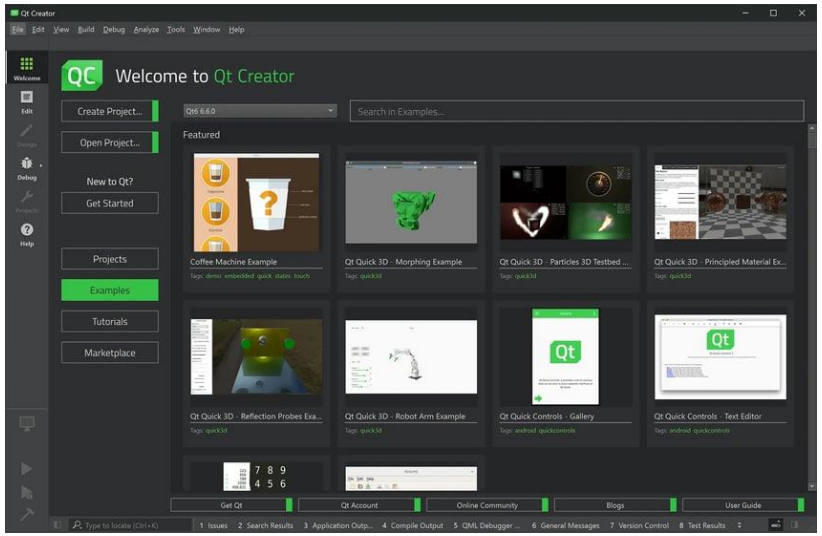
The AI-Enabled Cross-Platform IDE

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.

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



Cinder C++

http://libcinder.org/

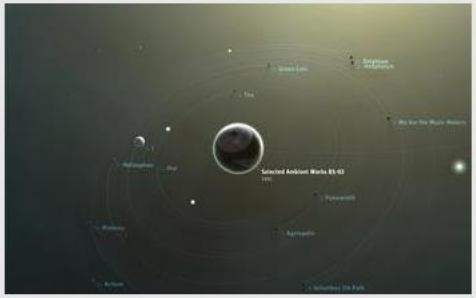
Cinder is a C++ library for programming with aesthetic intent - the sort of development often called *creative coding*. This includes domains like graphics, audio, video, and computational geometry. Cinder is cross-platform, with official support for macOS, Windows, Linux, iOS, and Windows UWP.

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

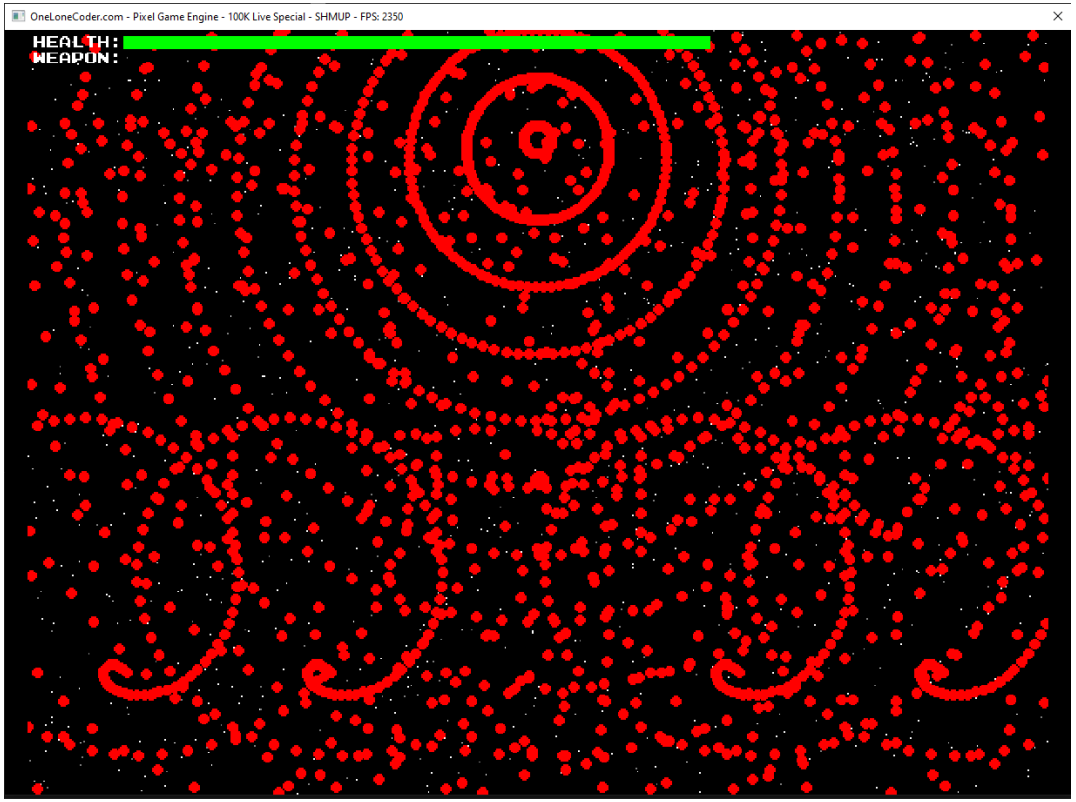
olcPixelGameEngine

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           {
434                   // GRAPHICS
435                   Clear(olc::BLACK);
436
437                   // Update & Draw Stars
438                   for (size_t i=0; i<aryStars.size(); i++)
439                   {
440                           auto& s = aryStars[i];
441                           s.y += fScrollSpeed * fElapsedTime * ((i<aryStars.size() >> 2) ? 0.8f : 1.0f);
442                           if (s.y >= (float)ScreenHeight())
443                                   s = { (float)(rand() % ScreenWidth()), 0.0f };
444
445                           Draw(s, (i < aryStars.size() >> 2) ? olc::DARK_GREY : olc::WHITE);
446                   }
447
448                   SetPixelMode(olc::Pixel::MASK);
449
450                   // Draw Enemies
451                   for (auto& e : listEnemies)     DrawSprite(e.pos, sprEnemy[e.def.nSpriteID]);
452
453                   // Draw Player
454                   DrawSprite(vPlayerPos, sprPlayer);
455
456                   SetPixelMode(olc::Pixel::NORMAL);
457
458                   // Draw Enemy Bullets
459                   for (auto& b : listEnemyBullets) FillCircle(b.pos, 3, olc::RED);
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++

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.”

Bjarne Stroustrup: The 5 Programming Languages You Need to Know

...**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**...

http://www.youtube.com/watch?v=NvWTnIoQZj4

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