

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

ISO/IEC 14882:2017  
(C++ 17)

Fabio Galuppo, M.Sc.

<http://fabiogaluppo.com> e <http://simplycpp.com/>

fabiogaluppo@acm.org

@FabioGaluppo

Microsoft MVP Visual Studio and Development Technologies

<https://mvp.microsoft.com/en-us/PublicProfile/9529>

<http://bit.ly/devxp2017>

# Stack Overflow

## Developer Survey Results 2017

### Programming Languages



<https://insights.stackoverflow.com/survey/2017#most-popular-technologies>

# TIOBE Index for May 2017

Java and C are in a heavy downward trend since the beginning of 2016. Both languages have lost more than 6% if compared to last year. So which programming languages are taking advantage of this drop? Well, actually all the other languages. Since software is adopted by more and more domains nowadays, C (low level software development) and Java (high level software development) apparently don't suffice any more. To illustrate this point, a rating of 0.6% was sufficient to reach the top 20 in 2012. Nowadays this would put you at position 33.

May 2017	May 2016	Change	Programming Language	Ratings	Change
1	1		Java	14.639%	-6.32%
2	2		C	7.002%	-6.22%
3	3		C++	4.751%	-1.95%
4	5	⬆	Python	3.548%	-0.24%
5	4	⬇	C#	3.457%	-1.02%
6	10	⬆	Visual Basic .NET	3.391%	+1.07%
7	7		JavaScript	3.071%	+0.73%
8	12	⬆	Assembly language	2.859%	+0.98%
9	6	⬇	PHP	2.693%	-0.30%
10	9	⬇	Perl	2.602%	+0.28%
11	8	⬇	Ruby	2.429%	+0.09%
12	13	⬆	Visual Basic	2.347%	+0.52%
13	15	⬆	Swift	2.274%	+0.68%
14	16	⬆	R	2.192%	+0.86%
15	14	⬇	Objective-C	2.101%	+0.50%
16	42	⬆	Go	2.080%	+1.83%

<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 2016 Top Programming Languages

## C is No. 1, but big data is still the big winner

The default preset is intended to echo the interests of the average IEEE member. So what are *Spectrum's* Top Ten Languages for 2016?

Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9























After two years in second place, C has finally edged out Java for the top spot. Staying in the top five, Python has swapped places with C++ to take the No. 3 position, and C# has fallen out of the top five to be replaced with R. R is following its momentum from previous years, as part of a positive trend in general for modern big-data languages that Diakopoulos analyses in more detail here.

<http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>

# 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

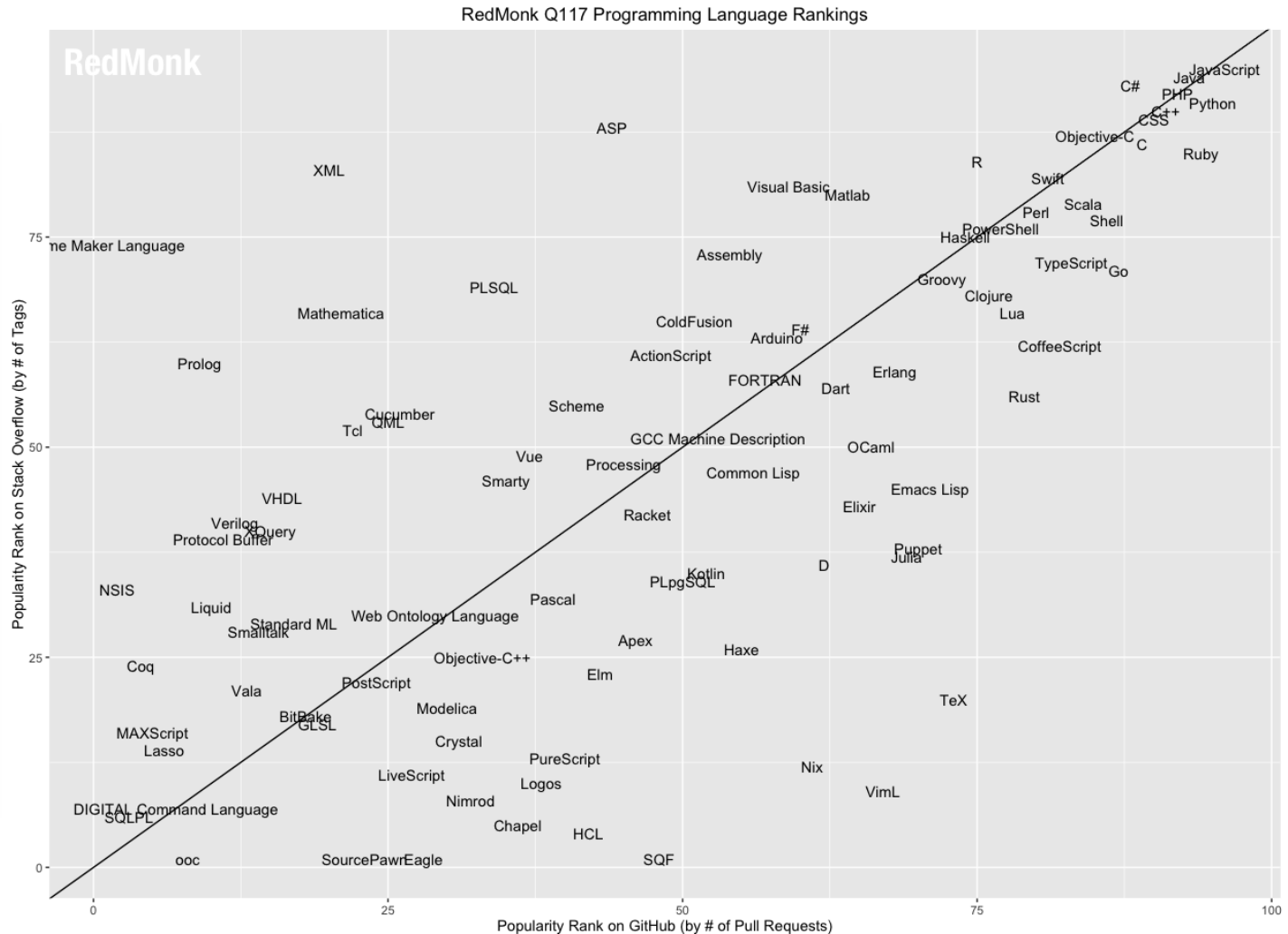
Python has continued its upward trajectory from last year and jumped two places to the No. 1 slot, though the top four—Python, C, Java, and C++—all remain very close in popularity. Indeed, in Diakopoulos's analysis of what the underlying metrics have to say about the languages currently in demand by 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 2017

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



# The Computer Language Benchmarks Game

## C++ g++ programs versus C gcc

### n-body

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>9.31</b>	1,052	1544	9.30	1%	1%	1%	100%
<u>C gcc</u>	9.96	1,016	1490	9.95	1%	1%	1%	100%

### mandelbrot

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	1.73	34,064	1002	6.80	98%	98%	98%	99%
<u>C gcc</u>	1.65	32,684	1135	6.53	99%	99%	100%	99%

### binary-trees

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>2.36</b>	114,404	809	7.69	86%	71%	87%	85%
<u>C gcc</u>	2.38	131,728	836	7.70	98%	87%	73%	68%

### pidigits

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	1.89	3,868	508	1.89	100%	1%	2%	1%
<u>C gcc</u>	1.73	2,116	448	1.73	1%	99%	1%	0%

### spectral-norm

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>1.99</b>	1,880	1044	7.89	100%	99%	99%	99%
<u>C gcc</u>	1.99	1,824	1139	7.88	99%	99%	99%	100%

### fasta

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	1.48	4,176	2313	5.25	89%	90%	88%	89%
<u>C gcc</u>	1.33	2,856	2249	5.28	100%	99%	100%	99%

<http://benchmarksgame.alioth.debian.org/u64q/cpp.html>



# The Computer Language Benchmarks Game

## C++ g++ programs versus Java

### binary-trees

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>2.36</b>	114,404	809	7.69	86%	71%	87%	85%
<u>Java</u>	11.26	593,156	835	39.02	85%	88%	90%	88%

### spectral-norm

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>1.99</b>	1,880	1044	7.89	100%	99%	99%	99%
<u>Java</u>	4.29	29,884	950	16.56	96%	97%	99%	95%

### mandelbrot

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>1.73</b>	34,064	1002	6.80	98%	98%	98%	99%
<u>Java</u>	7.10	90,588	796	27.92	99%	99%	98%	98%

### pidigits

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>1.89</b>	3,868	508	1.89	100%	1%	2%	1%
<u>Java</u>	3.06	31,760	938	3.16	6%	3%	97%	1%

### n-body

source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>9.31</b>	1,052	1544	9.30	1%	1%	1%	100%
<u>Java</u>	21.54	27,092	1489	21.56	1%	1%	100%	1%

### fasta

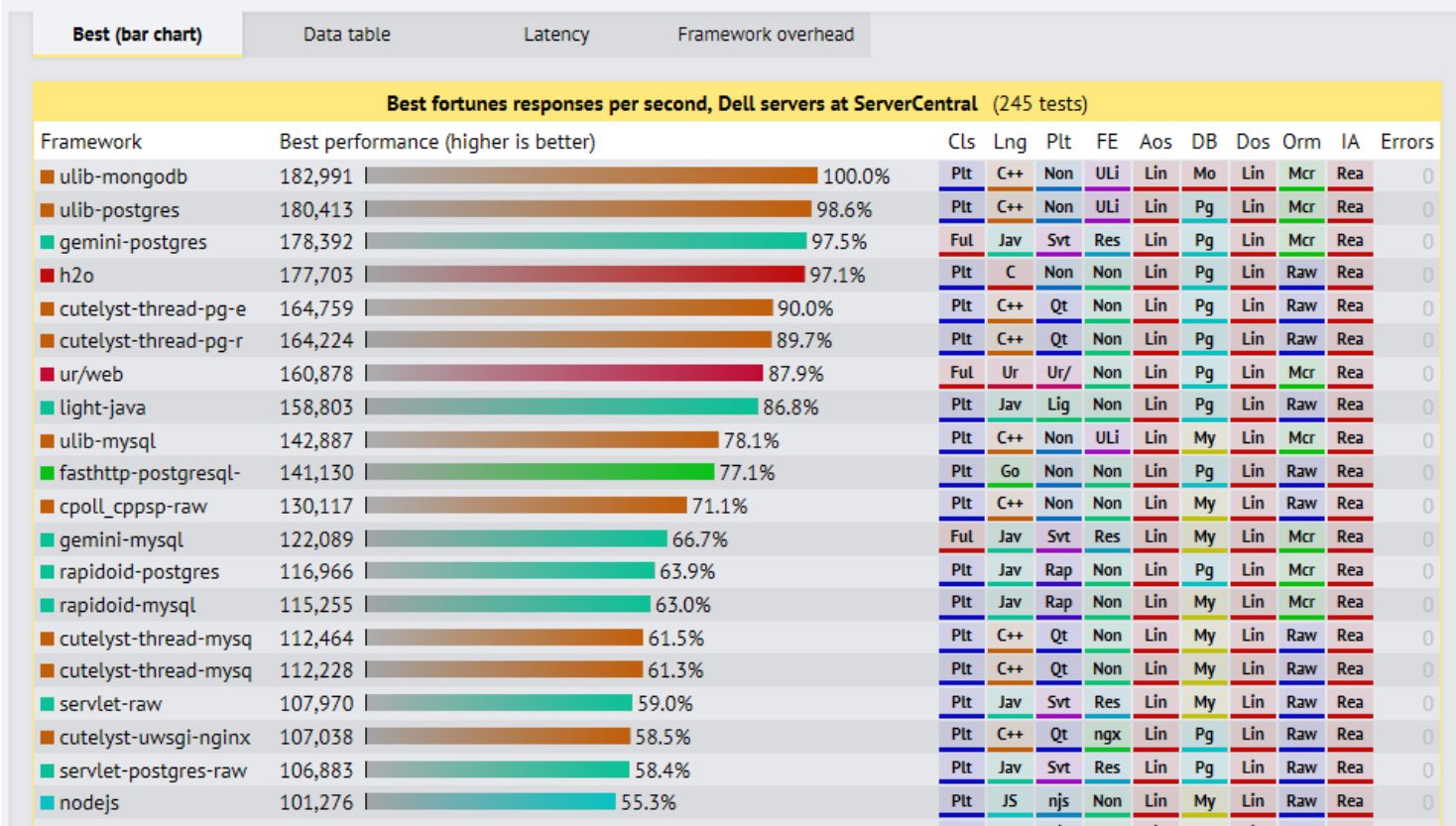
source	secs	mem	gz	cpu	cpu load			
<u>C++ g++</u>	<b>1.48</b>	4,176	2313	5.25	89%	90%	88%	89%
<u>Java</u>	2.13	36,036	2457	5.66	94%	58%	59%	60%

# Web Framework Benchmarks

## Test type 4: Fortunes

This test exercises the ORM, database connectivity, dynamic-size collections, sorting, server-side templates, XSS countermeasures, and character encoding.

## Results



<https://github.com/stefanocasazza/ULib>

ULib is a mature and well-considered C++ framework for developing high-performance applications that includes a blazingly-fast async HTTP server--more batteries included.

<https://www.techempower.com/benchmarks/#section=data-r14&hw=ph&test=fortune>

**C++**

# O que é C++?

## What is C++?

Smarter computing  
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

An object-oriented  
programming language

Generic programming

Low level!

A random collection  
of features

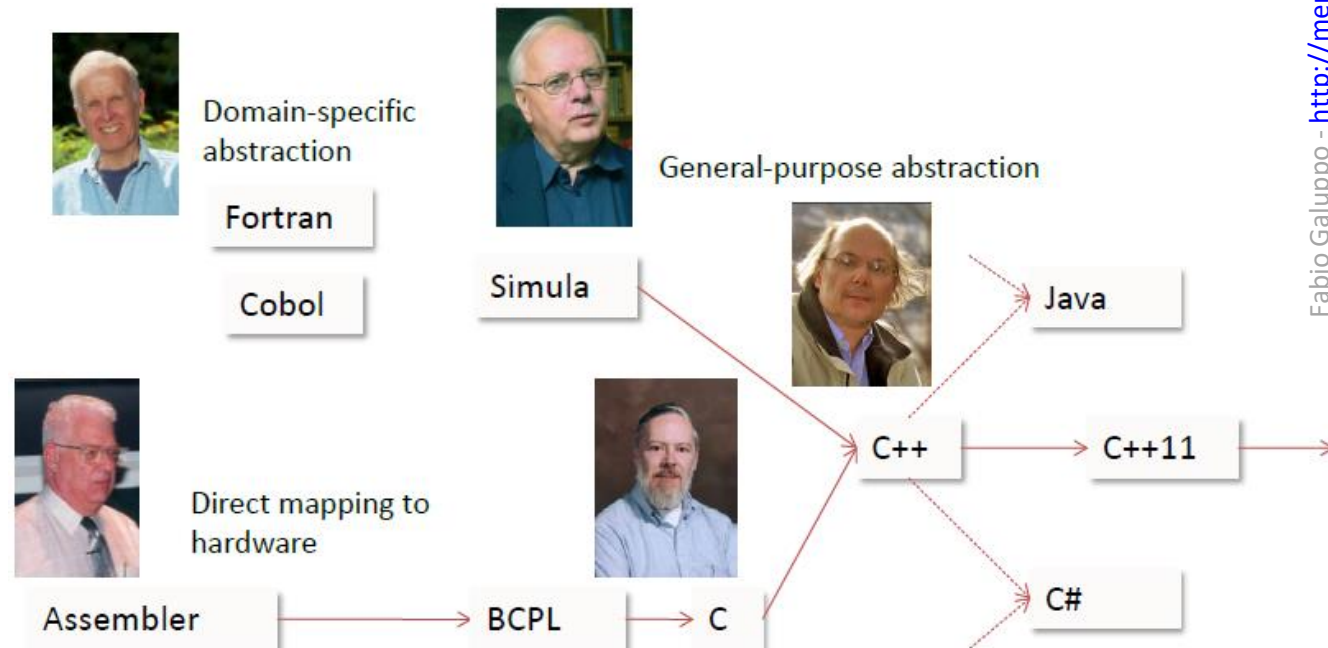
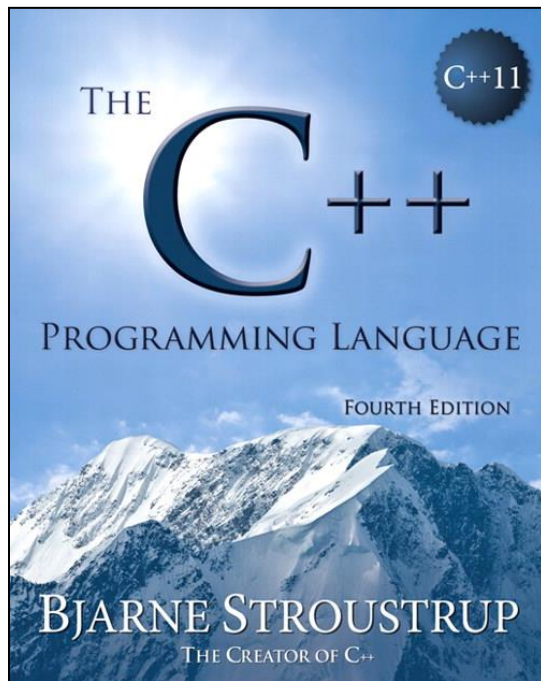
Stroustrup - ACCU'13

# The C++ Programming Language

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

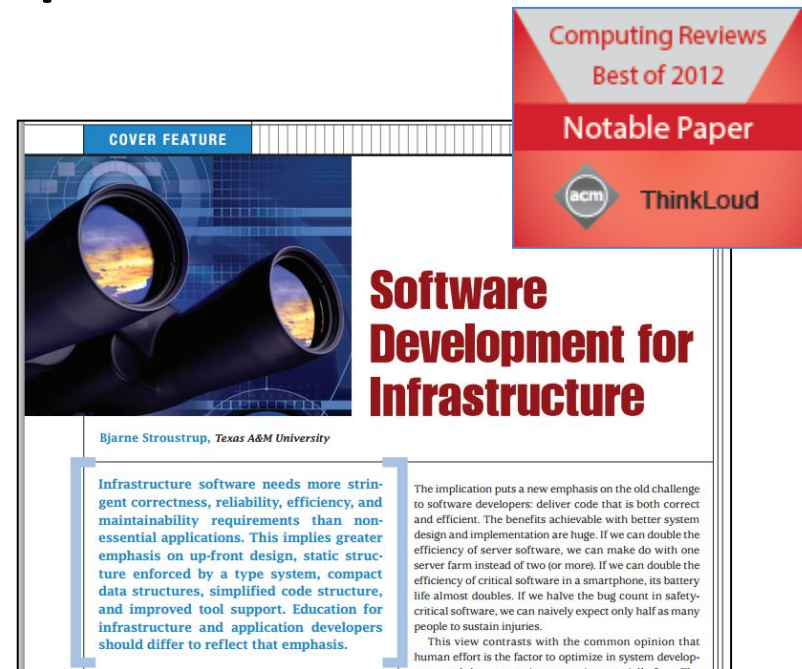
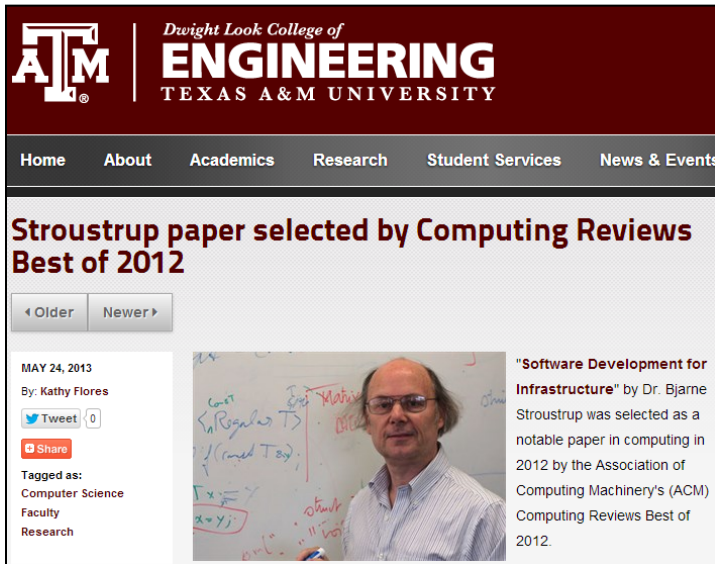
- is a better C
- supports data abstraction
- supports object-oriented programming
- supports generic programming.

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





# C++



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”

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

C++ 11

The Future is here

# C++: o início

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

[http://www.softwarepreservation.org/projects/c\\_plus\\_plus](http://www.softwarepreservation.org/projects/c_plus_plus)

## 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*

C++ Historical Sources Archive

by [Paul McJones](#) — last modified 2010-10-04 13:33

Paul McJones, editor  
[paul@mcjones.org](mailto:paul@mcjones.org)  
<http://www.mcjones.org/dustydecks/>

### Abstract

This is a collection of design documents, source code, and other materials concerning the birth, development, standardization, and use of the C++ programming language.

### Contents

- Chronology
- Cfront releases

[http://www.softwarepreservation.org/projects/c\\_plus\\_plus/cfront/release\\_1.0/src/cfront/](http://www.softwarepreservation.org/projects/c_plus_plus/cfront/release_1.0/src/cfront/)

## Release 1.0

Cfront 1.0, in October 1985, was the first commercial release.

## Source Code

- Release 1.0, AT&T Technologies, Inc. Files timestamped February 7, 1986.

# C++: o caminho para o ISO C++

- 1990 (ANSI C++ Standard – baseado no “ARM”)
- 1998 (Primeira versão do padrão ISO C++)  
<https://www.iso.org/standard/25845.html>
- 2003 (Segunda versão do padrão ISO C++)  
<https://www.iso.org/standard/38110.html>
- 2011 (Terceira versão do padrão ISO C++)  
<https://www.iso.org/standard/50372.html>
- 2014 (Quarta versão do padrão ISO C++)  
<https://www.iso.org/standard/64029.html>
- 2017 (Quinta versão do padrão ISO C++)  
<https://www.iso.org/standard/68564.html> (em andamento)  
<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4659.pdf> (draft atual)

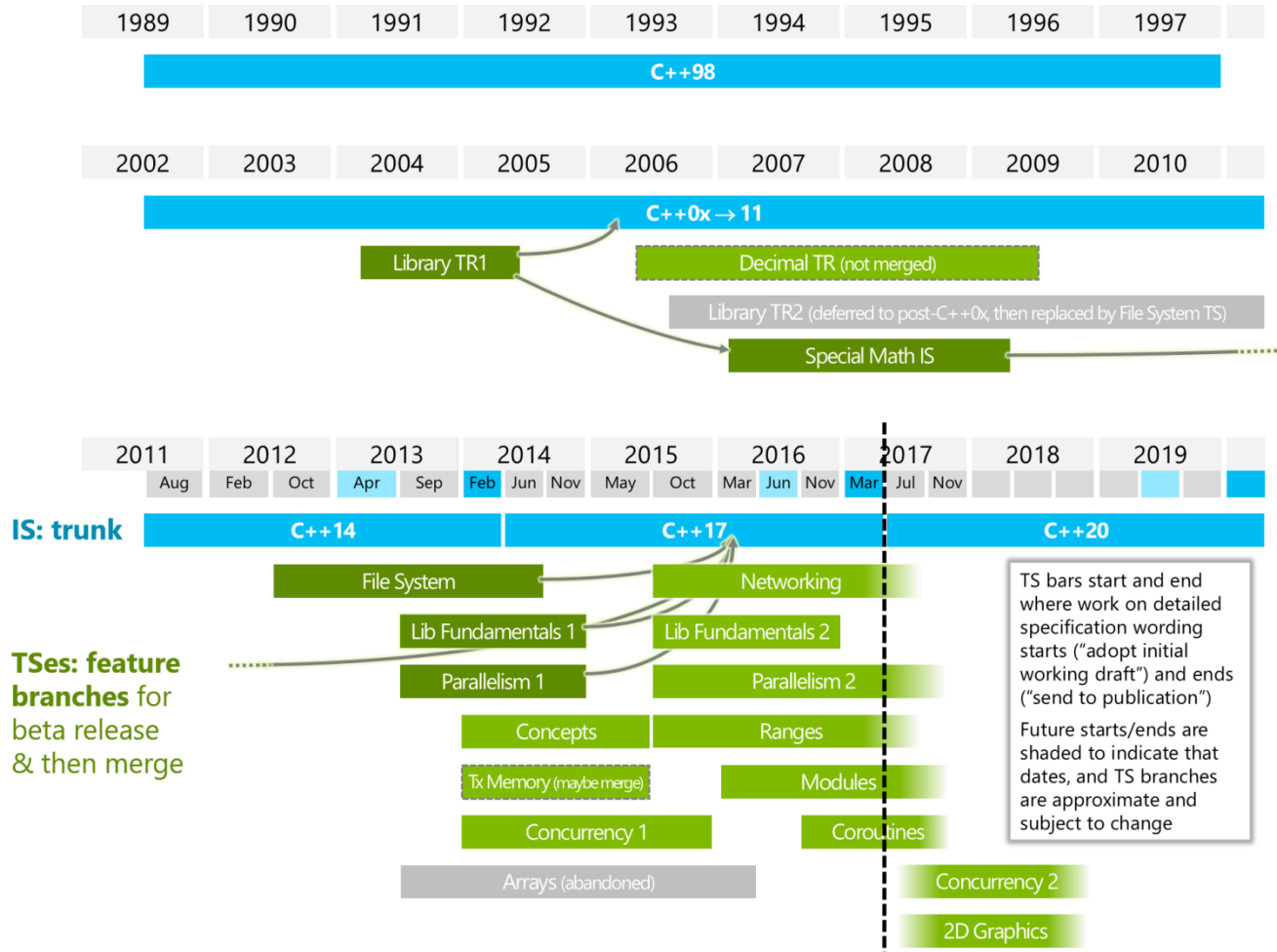




International  
Organization for  
Standardization

# ISO C++ (WG 21)

<http://isocpp.org/>



The committee has completed work on C++17, which is now in its final ISO balloting process, and aims to begin work on C++20 in July.

<https://isocpp.org/std/status>

# Linguagem + Biblioteca

C++ 17: <https://github.com/cplusplus/draft>

C++ 14: ISO/IEC 14882:2014

Document Number: N4659  
Date: 2017-03-21  
Revises: N4640  
Reply to: Richard Smith  
Google Inc  
cxxeditor@gmail.com

## Working Draft, Standard for Programming Language C++

acesso em: 01 de Junho de 2017

ISO/IEC 14882:2014 specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, and so this International Standard also defines C++. Other requirements and relaxations of the first requirement appear at various places within this International Standard.

C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:1999 *Programming languages ? C* (hereinafter referred to as the *C standard*). In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

Table 5 — Keywords

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

5.11 Keywords [lex.key] [5. Lexical conventions [lex]]

## Compiladores



[http://en.cppreference.com/w/cpp/compiler\\_support](http://en.cppreference.com/w/cpp/compiler_support)

# !“Hello, World!”

```
unsigned long long pow(unsigned int base, unsigned int exponent) {  
    long long result = 1;  
    while (exponent-- > 0) {  
        result *= base;  
    }  
    return result;  
}  
  
#include <iostream>  
  
int main() {  
    std::cout << "2^4 = " << pow(2, 4) << "\n";  
    std::cout << "3^3 = " << pow(3, 3) << "\n";  
    std::cout << "4^3 = " << pow(4, 3) << "\n";  
    std::cout << "5^2 = " << pow(5, 2) << "\n";  
    return 0;  
}
```

MinGW with g++ 5.3.0 and boost 1.60.0

```
C:\__repo\cpp>g++ -pipe -Wall -O2 -std=c++14 -S pow_test.cpp
```

```
C:\__repo\cpp>a
```

```
2^4 = 16
```

```
3^3 = 27
```

```
4^3 = 64
```

```
5^2 = 25
```

# !“Hello, World!”

The image shows a CppPlayground window with a C++ program and its configuration properties.

**Code in limit\_cpp17.cpp:**

```
1 #include <functional>
2 #include <cmath>
3 #include <optional>
4
5 std::optional<float> limit(std::function<float(float)> f, float x0) {
6     const float threshold = 0.0001f;
7     auto left = f(x0 - threshold);
8     auto right = f(x0 + threshold);
9     auto estimate = (left + right) / 2.f;
10    if (std::abs(left - right) < 0.01f)
11        return estimate;
12    return std::nullopt;
13 }
14
```

**CppPlayground Property Pages:**

- Configuration: Active(Debug)
- Platform: Active(Win32)
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - All Options
    - Command Line
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

Property	Value
Disable Language Extensions	No
Treat WChar_t As Built in Type	Yes (/Zc:wchar_t)
Force Conformance in For Loop Scope	Yes (/Zc:forScope)
Remove unreferenced code and data	Yes (/Zc:inline)
Enforce type conversion rules	
Enable Run-Time Type Information	
Open MP Support	
C++ Language Standard	ISO C++ Latest Draft Standard (/std:c++latest)

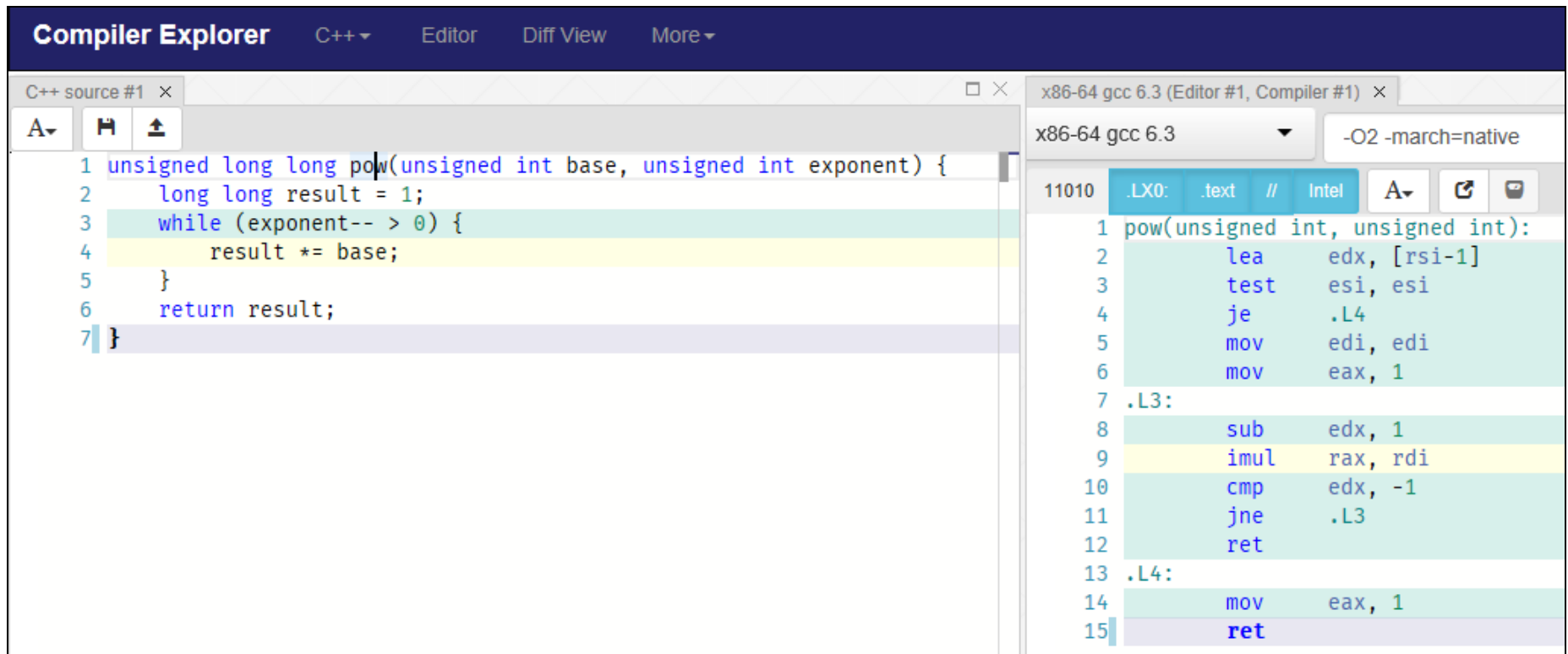
**Command Prompt Output:**

```
C:\WINDOWS\system32\cmd.exe
f: lim(3 * x^2), x->2
The limit of f when x approaches 2.000000 is 12.000000
f: lim((2 * x + 1) / (x^2)), x->3
The limit of f when x approaches 3.000000 is 0.777778
f: lim((x^2 - 1) / (x - 1)), x->1
The limit of f when x approaches 1.000000 is 2.000000
f: lim(sin(5 / (x - 1))), x->1
The limit does not exist
f: lim(e^x), x->0
The limit of f when x approaches 0.000000 is 1.000000
f: lim(sin(x) / x), x->0
The limit of f when x approaches 0.000000 is 1.000000
```

<https://www.visualstudio.com/>

# C++ Explorer

## Assembly At Your Fingertips



The screenshot displays the Compiler Explorer interface. The left pane shows the C++ source code for a `pow` function. The right pane shows the generated assembly code for x86-64 gcc 6.3. The assembly code includes labels `.L3` and `.L4` and instructions like `lea`, `test`, `je`, `mov`, `imul`, `cmp`, `jne`, and `ret`.

```
Compiler Explorer C++ Editor Diff View More
```

```
C++ source #1 x
```

```
1 unsigned long long pow(unsigned int base, unsigned int exponent) {
2     long long result = 1;
3     while (exponent-- > 0) {
4         result *= base;
5     }
6     return result;
7 }
```

```
x86-64 gcc 6.3 (Editor #1, Compiler #1) x
```

```
x86-64 gcc 6.3 -O2 -march=native
```

```
11010 .LX0: .text // Intel A
```

```
1 pow(unsigned int, unsigned int):
2     lea     edx, [rsi-1]
3     test   esi, esi
4     je     .L4
5     mov     edi, edi
6     mov     eax, 1
7 .L3:
8     sub     edx, 1
9     imul    rax, rdi
10    cmp     edx, -1
11    jne     .L3
12    ret
13 .L4:
14    mov     eax, 1
15    ret
```

<https://godbolt.org/>

# Eu quero aprender C++!

## Por onde começar?

Bjarne Stroustrup

The Essence of C++: With Examples in C++84, C++98, C++11, and C++14

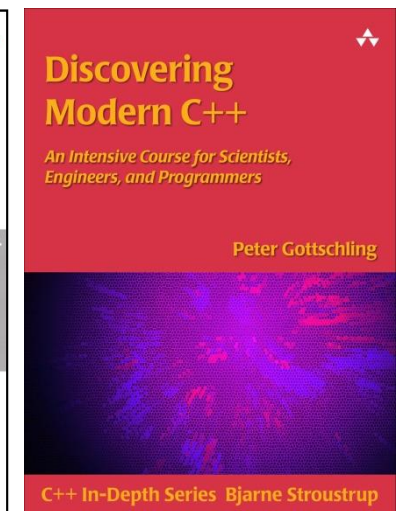
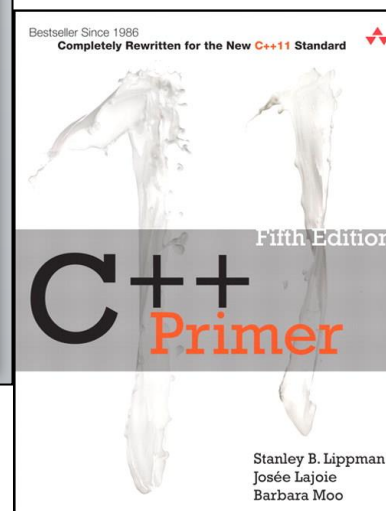
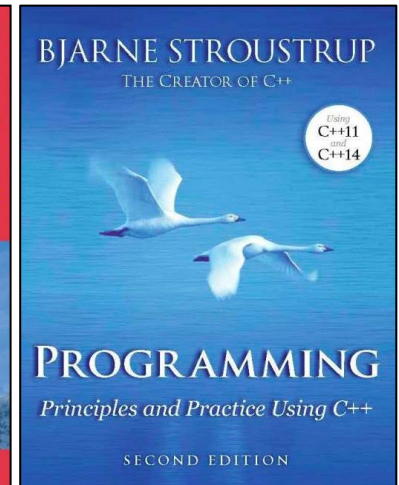
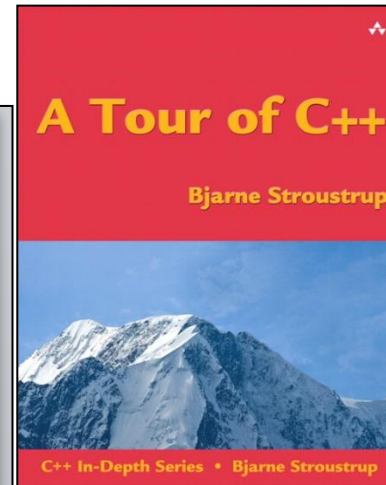
### Abstract

- C++11 is being deployed and the shape of C++14 is becoming clear. This talk examines the foundations of C++. What is essential? What sets C++ apart from other languages? How does new and old features support (or distract from) design and programming relying on this essence.
- I focus on the abstraction mechanisms (as opposed to the mapping to the machine): Classes and templates. Fundamentally, if you understand vector, you understand C++.
- Type safety and resource safety are key design aims for a program. These aims must be met without limiting the range of applications and without imposing significant run-time or space overheads. I address issues of resource management (garbage collection is not an ideal answer and pointers should not be used as resource handles), generic programming (we must make it simpler and safer), compile-time computation (how and when?), and type safety (casts belongs in the lowest-level hardware interface). I will touch upon move semantics, exceptions, concepts, type aliases, and more. My aim is not so much to present novel features and technique, but to explore how C++'s feature set supports a new and more effective design and programming style.
- Primary audience
  - Experienced programmers with weak C++ understanding
  - Academics/Teachers/Mentors
  - Architects (?)

Stroustrup - Essence - Going Native'13

2

<http://channel9.msdn.com/Events/GoingNative/2013/Opening-Keynote-Bjarne-Stroustrup>





# Standard Template Library (STL)

Reference - C++ Reference

Reference

Standard C++ Library reference

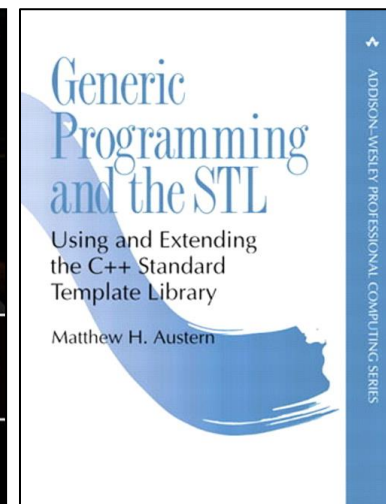
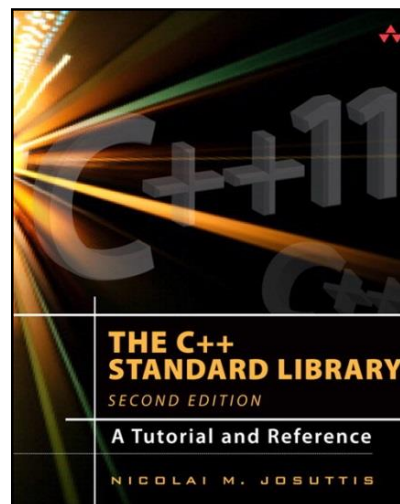
C Library

The elements of the C language library are also included as a subset of the C++ Standard library, aspects, from general utility functions and macros to input/output functions and dynamic memory functions:

<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)
<locale> (locale.h)	C localization library (header)
<cmath> (math.h)	C numerics library (header)
<setjmp> (setjmp.h)	Non local jumps (header)
<csignal> (signal.h)	C library to handle signals (header)
<cstdlib> (stdlib.h)	Variable arguments handling (header)
<stdbool> (stdbool.h)	Boolean type (header)
<stddef> (stddef.h)	C Standard definitions (header)
<stdint> (stdint.h)	Integer types (header)
<stdio> (stdio.h)	C library to perform Input/Output operations (header)
<stdlib> (stdlib.h)	C Standard General Utilities Library (header)
<string> (string.h)	C Strings (header)
<tgmath> (tgmath.h)	Type-generic math (header)
<time> (time.h)	C Time Library (header)
<wchar> (wchar.h)	Unicode characters (header)

<http://www.cplusplus.com>

Selected Standard Library Headers		
<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
<fstream>	fstream, ifstream, ofstream	§iso.27.9.1
<future>	future, promise	§iso.30.6
<iostream>	hex,dec,scientific,fixed,defaultfloat	§iso.27.5
<istream>	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>	istringstream, ostringstream	§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



# Stepanov, STL e Programação Genérica

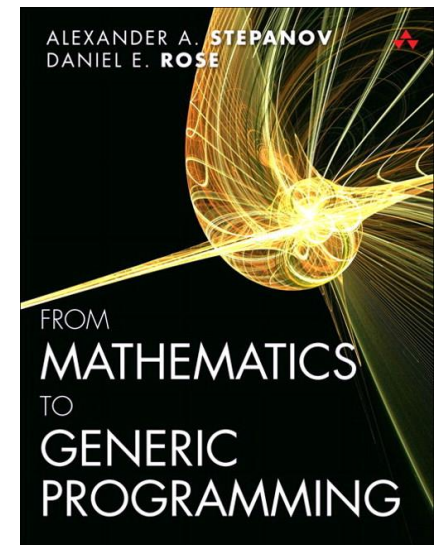
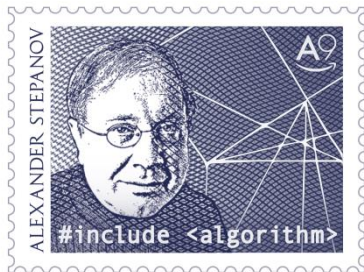
- The Standard Template Library, or *STL*, is a C++ library of **container classes**, **algorithms**, and **iterators**; it provides many of the basic algorithms and data structures of computer science. The STL is a *generic* library, meaning that its components are heavily parameterized: almost every component in the STL is a template. You should make sure that you understand how templates work in C++ before you use the STL.

[https://www.sgi.com/tech/stl/stl\\_introduction.html](https://www.sgi.com/tech/stl/stl_introduction.html)

- Alexander Stepanov

*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](http://www.fm2gp.com/slides/FM2GP_Course_Slides_Pt1.pdf)



# #include <algorithm>

## <algorithm>

<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 }
```

# Programação Concorrente com C++

```
#include <thread>
#include <future>
#include <atomic>
#include <mutex>
#include <condition_variable>
```



library	
<b>Multi-threading</b> 	
<b>Atomic and thread support</b>	
Support for atomics and threads:	
● <b>Headers</b>	
<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](http://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.

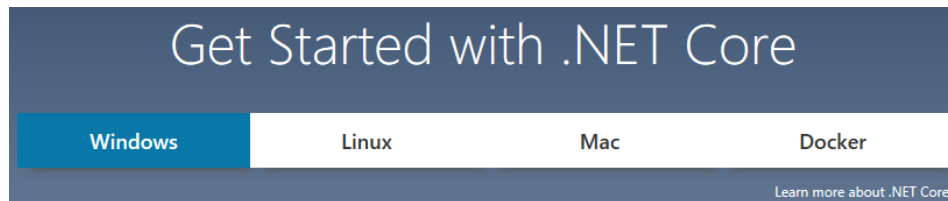
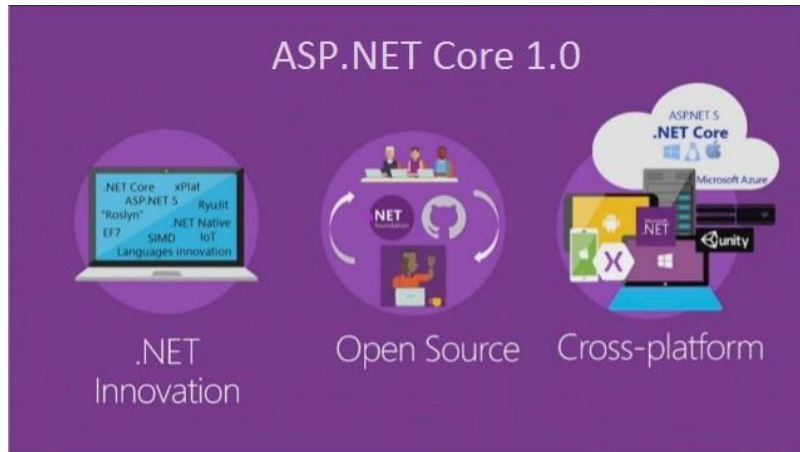
**THANKS** to the following technical experts for reviewing this article:  
David Cravey, Alon Fliess, **Fabio Galuppo** and Marc Gregoire

artigo online: <http://msdn.microsoft.com/en-us/magazine/hh852594.aspx>

código fonte: <http://archive.msdn.microsoft.com/mag201203CPP>

**TECNOLOGIAS QUE UTILIZAM C++**

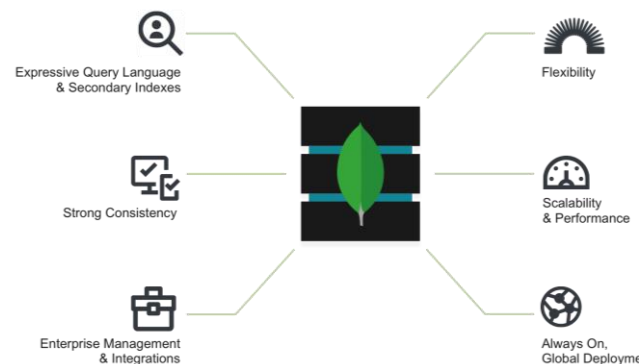
# Quem depende de C++?



amd64	[x64/Linux] Use correct argument registers in InterpreterStub (#12002)
arm	Finish deleting dead CAS code from CoreLib (#11436)
arm64	[Arm64/Win] Revise JIT_MemSet (#11420)
coreclr	Initial change to support System.Private.CoreLib.dll as Core Library.
crossgen	Finish deleting dead CAS code from CoreLib (#11436)
crossgen_mscorlib	Remove files related to legacy build system (#8723)
dac	Remove files related to legacy build system (#8723)
i386	Linux/x86: Fix clang 4.0 build (#11610)
wks	Fix AsmConstants.inc to be incrementally rebuilt as necessary
.gitmirror	Initial commit to populate CoreCLR repo
CMakeLists.txt	Finish deleting dead CAS code from CoreLib (#11436)
ClrEtwAll.man	Fix Externally reported issue with are ETW logging messages (#11722)
ClrEtwAllMeta.lst	adding more context
appdomain.cpp	Finish deleting dead CAS code from CoreLib (#11436)
appdomain.hpp	<a href="#">Finish deleting dead CAS code from CoreLib (#11436)</a>
appdomain.inl	Finish deleting dead CAS code from CoreLib (#11436)
appdomainconfigfactory.hpp	Update license headers
appdomainnative.cpp	Finish deleting dead CAS code from CoreLib (#11436)
appdomainnative.hpp	Finish deleting dead CAS code from CoreLib (#11436)
appxutil.cpp	Remove hosting methods that always return false (#9930)
appxutil.h	Delete dead code
argdestination.h	For Reflection invoke, use ArgLocDesc
argslot.h	Update license headers
armsinglestepper.h	Update license headers
array.cpp	Refactor MethodTable::ContainsStackPtr (#5754)
array.h	Update license headers
assembly.cpp	Finish deleting dead CAS code from CoreLib (#11436)

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

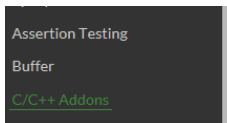
# Quem depende de C++?



<a href="#">background.cpp</a>	SERVER-23971 Clang-Format code	a year ago
<a href="#">background.h</a>	SERVER-23971 Clang-Format code	a year ago
<a href="#">client.cpp</a>	SERVER-28201 Allow detaching of the current Client	a month ago
<a href="#">client.h</a>	SERVER-28201 Allow detaching of the current Client	a month ago
<a href="#">clientcursor.cpp</a>	SERVER-28327 Revamp cursor timeout logic.	29 days ago
<a href="#">clientcursor.h</a>	SERVER-21754 Partition CursorManager's structures	6 days ago
<a href="#">cloner.cpp</a>	SERVER-29088 Cache uuid in Collection class to fix perf regression	24 days ago
<a href="#">cloner.h</a>	SERVER-27938 Rename all OperationContext variables to opCtx	3 months ago
<a href="#">collection_index_usage_tracker.cpp</a>	SERVER-20520 Include index key in \$indexStats return documents	2 years ago
<a href="#">collection_index_usage_tracker.h</a>	SERVER-20520 Include index key in \$indexStats return documents	2 years ago
<a href="#">collection_index_usage_tracker_test...</a>	SERVER-24508 BSONObj::ComparatorInterface	10 months ago
<a href="#">commands.cpp</a>	SERVER-29264 Replace RequestInterface with OpMsgRequest in mongod com...	2 days ago
<a href="#">commands.h</a>	SERVER-29264 Replace RequestInterface with OpMsgRequest in mongod com...	2 days ago
<a href="#">commands_test.cpp</a>	SERVER-27773 add operationTime field to the command response	3 months ago
<a href="#">conn_pool_options.cpp</a>	SERVER-20096: ExportedServerParameter<T> is not thread-safe for param...	2 years ago
<a href="#">conn_pool_options.h</a>	SERVER-18579: Clang-Format - reformat code, no comment reflow	2 years ago
<a href="#">curop.cpp</a>	SERVER-28575 Profile entry for update/delete should contain entire ra...	9 days ago
<a href="#">curop.h</a>	SERVER-28575 Profile entry for update/delete should contain entire ra...	9 days ago
<a href="#">curop_metrics.cpp</a>	SERVER-23971 Clang-Format code	a year ago
<a href="#">curop_metrics.h</a>	SERVER-18579: Clang-Format - reformat code, no comment reflow	2 years ago
<a href="#">cursor_id.h</a>	SERVER-18766 Pull definition of CursorId out into its own header	2 years ago
<a href="#">db.cpp</a>	SERVER-29254 Moved MinValid into ReplicationProcess	2 days ago
<a href="#">db.h</a>	SERVER-27987 Create and persist UUIDs for newly created collections	3 months ago

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

# Quem depende de C++?



## Addons

Node.js Addons are dynamically-linked shared objects, written in C or C++, that can be loaded into Node.js using the `require()` function, C/C++ libraries.

<a href="#">inspector_io.cc</a>	inspector: bind to random port with --inspect=0	2 days ago
<a href="#">inspector_io.h</a>	inspector: bind to random port with --inspect=0	2 days ago
<a href="#">inspector_socket.cc</a>	inspector: handle socket close before close frame	20 days ago
<a href="#">inspector_socket.h</a>	inspector: zero out structure members	8 months ago
<a href="#">inspector_socket_server.cc</a>	inspector: fix process_debugEnd() for inspector	10 days ago
<a href="#">inspector_socket_server.h</a>	inspector: fix process_debugEnd() for inspector	10 days ago
<a href="#">js_stream.cc</a>	src: implement native changes for async_hooks	22 days ago
<a href="#">js_stream.h</a>	src: implement native changes for async_hooks	22 days ago
<a href="#">node.cc</a>	inspector: bind to random port with --inspect=0	2 days ago
<a href="#">node.d</a>	dtrace: add missing translator	5 years ago
<a href="#">node.h</a>	async_hooks: implement C++ embedder API	5 days ago
<a href="#">node.stp</a>	meta: restore original copyright header	3 months ago
<a href="#">node_api.cc</a>	n-api: enable napi_wrap() to work with any object	3 hours ago
<a href="#">node_api.h</a>	n-api: add napi_get_version	6 days ago
<a href="#">node_api_types.h</a>	n-api: implement async helper methods	2 months ago
<a href="#">node_buffer.cc</a>	lib,src: refactor buffer out of range index	5 days ago
<a href="#">node_buffer.h</a>	meta: restore original copyright header	3 months ago
<a href="#">node_config.cc</a>	src: add --pending-deprecation and NODE_PENDING_DEPRECATION	a month ago
<a href="#">node_constants.cc</a>	src: reduce number of exported symbols	2 months ago
<a href="#">node_constants.h</a>	crypto: add sign/verify support for RSASSA-PSS	2 months ago
<a href="#">node_contextify.cc</a>	vm: fix race condition with timeout param	9 days ago
<a href="#">node_counters.cc</a>	meta: restore original copyright header	3 months ago

Sounds pretty well right but, why should I write C++ code since I'm very comfortable with Javascript and the last time I saw C++ code was in the university... the answer is nothing else than **Performance!**

<https://medium.com/developers-writing/how-to-get-a-performance-boost-using-node-js-native-addons-fd3a24719c85>

<https://github.com/nodejs/node/tree/master/src>

# Quem depende de C++?



<a href="#">Executor.cpp</a>	Add keepAlive() mechanism	4 months ago
<a href="#">Executor.h</a>	Add keepAlive() mechanism	4 months ago
<a href="#">Expected.h</a>	Suppress more warnings for MSVC	2 months ago
<a href="#">FBString.h</a>	Enable -Wimplicit-fallthrough	11 days ago
<a href="#">FBVector.h</a>	Don't use macros for FBVector::insert	2 months ago
<a href="#">File.cpp</a>	Helper utility to construct, returns an Expected<.,>	a month ago
<a href="#">File.h</a>	Helper utility to construct, returns an Expected<.,>	a month ago
<a href="#">FileUtil.cpp</a>	2017	5 months ago
<a href="#">FileUtil.h</a>	2017	5 months ago
<a href="#">Fingerprint.h</a>	2017	5 months ago
<a href="#">FixedString.h</a>	FixedString gets comparisons with folly::Range and hence with std::st...	28 days ago
<a href="#">Foreach.h</a>	folly/Foreach.h: avoid shadowing warnings	3 months ago
<a href="#">Format-inl.h</a>	Fix FBString with MSVC	24 days ago
<a href="#">Format.cpp</a>	add FOLLY_FALLTHROUGH throughout to satisfy internal linter	22 days ago

## Folly: Facebook Open-source Library

### What is `folly` ?

Folly (acronymed loosely after Facebook Open Source Library) is a library of C++11 components designed with practicality and efficiency in mind. **Folly contains a variety of core library components used extensively at Facebook.** In particular, it's often a dependency of Facebook's other open source C++ efforts and place where those projects can share code.

It complements (as opposed to competing against) offerings such as Boost and of course `std`. In fact, we embark on defining our own component only when something we need is either not available, or does not meet the needed performance profile. We endeavor to remove things from folly if or when `std` or Boost obsoletes them.

Performance concerns permeate much of Folly, sometimes leading to designs that are more idiosyncratic than they would otherwise be (see e.g. `PackedSyncPtr.h`, `SmallLocks.h`). Good performance at large scale is a unifying theme in all of Folly.

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

# Quem depende de C++?



<a href="#">PackedNormal.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">RenderCommandFence.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">RenderCore.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">RenderResource.h</a>	Copying //UE4/Dev-Core to //UE4/Dev-Main (Source: //UE4/Dev-Core @ 33...	3 months ago
<a href="#">RenderUtils.h</a>	Copying //UE4/Dev-Rendering to //UE4/Dev-Main (Source: //UE4/Dev-Rend...	6 months ago
<a href="#">RendererInterface.h</a>	Copying //UE4/Dev-Rendering to //UE4/Dev-Main (Source: //UE4/Dev-Rend...	2 months ago
<a href="#">RenderingThread.h</a>	Copying //UE4/Dev-Core to //UE4/Dev-Main (Source: //UE4/Dev-Core @ 33...	3 months ago
<a href="#">TickableObjectRenderThread.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">UniformBuffer.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago

<a href="#">AtmosphereTextureParameters.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">DecalRenderingCommon.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">DrawingPolicy.h</a>	Copying //UE4/Dev-Rendering to //UE4/Dev-Main (Source: //UE4/Dev-Rend...	2 months ago
<a href="#">GlobalDistanceFieldParameters.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">HdrCustomResolveShaders.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">MaterialShader.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">MeshMaterialShader.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">PostProcessParameters.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">PrimitiveSceneInfo.h</a>	Updating copyright notices to 2017 (copying from //Tasks/UE4/Dev-Copy...	6 months ago
<a href="#">SceneRenderTargetParameters.h</a>	Require explicit material domain when setting deferred params safety ...	13 days ago
<a href="#">VolumeRendering.h</a>	Copying //UE4/Dev-Rendering to //UE4/Dev-Main (Source: //UE4/Dev-Rend...	2 months ago

<https://www.unrealengine.com/ue4-on-github>



# **BIBLIOTECAS PARA C++**

# Awesome C++

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

## All Categories

Artificial Intelligence	15
Asynchronous Event Loop	5
Audio	11
Biology	4
BitTorrent	3
CLI	7
Compression	20
Concurrency	23
Containers	11
Cryptography	15
CSV	2
Database	23
Data Structures	1
Debug	25
Frameworks	30
Game Engine	19
Graphics	32
GUI	17

⋮

The screenshot shows the 'Awesome C++' website interface. At the top, there's a navigation bar with 'Awesome C++', a search bar, and links for 'Newsletter', 'Submit', 'Categories', and 'Login'. Below this, the main header for 'ZeroMQ' is displayed, including its description 'High-speed, modular asynchronous communication library. [LGPL]' and buttons for 'zeromq.org' and 'Source Code'. On the right, there are 'Recommend' and 'Suggest Changes' buttons. The main content area has two tabs: 'Overview' and 'Resources'. Under 'Overview', there are several metrics: 'Popularity' (★ 9.2), 'Stable' (with a right arrow), 'Activity' (❤️ 9.4), 'Growing' (with an upward arrow), '★ Stars' (3,593), '👁 Watchers' (320), '🔗 Forks' (1,116), and '🕒 Last Commit' (3 days ago). Below this, there's a section titled '⇄ Popular Comparisons' with five boxes showing 'ZeroMQ' compared to 'gRPC', 'nanomsg', 'Apache Thrift', 'Cap'n Proto', and 'WAMP'.

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

# 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++)
- Múltiplas 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



# 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 both commercial and non-commercial use.

## 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/>

[http://www.boost.org/doc/libs/1\\_64\\_0/](http://www.boost.org/doc/libs/1_64_0/)

# 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

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

c-plus-plus

games

c-plus-plus-11

modern-cpp

stl

● C++ ★ 2,641 🍷 255 Updated 7 days ago



**Julian Manolov**

@jjju\_

 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)

# 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*

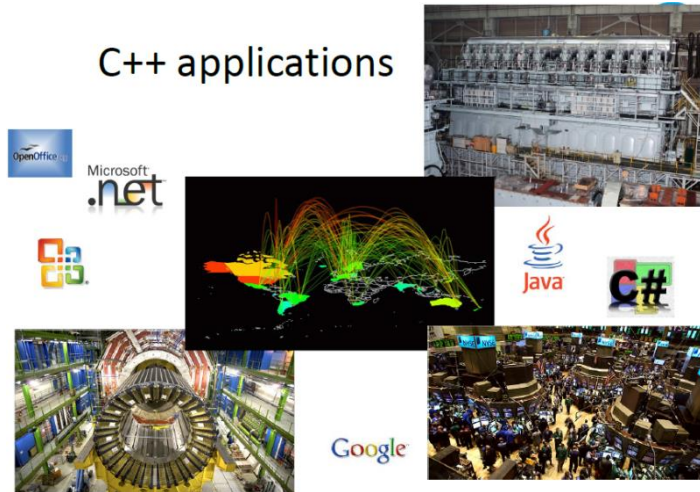
Visualize your music collection on your iPad as a galaxy of stars, planets and moons. This project became Cooper Hewitt National Design Museum's first digital acquisition.

<https://libcinder.org/gallery>

**PORQUE APRENDER C++?**

# Mais aplicações de C++

## C++ applications



## C++ Applications

- [www.research.att.com/~bs/applications.html](http://www.research.att.com/~bs/applications.html)



## C++ Applications

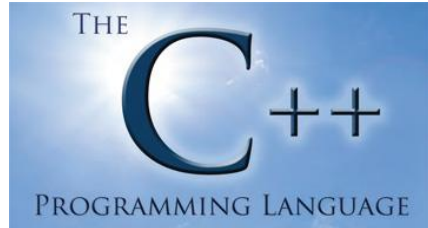


<http://www.stoustrup.com/applications.html>

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



# 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.

<http://www.youtube.com/watch?v=NvWTnIoQZi4>



Bjame Stroustrup: The 5 Programming Languages You Need to Know

“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*
- 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
- Desafiador

# Se quiser saber mais sobre:

C++  
Programação Genérica  
Iterators  
Policy-based Design  
Algoritmos

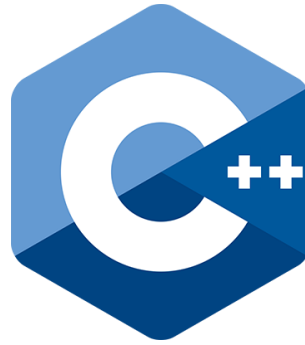
...

visite:

[www.simplycpp.com](http://www.simplycpp.com)



<http://www.simplycpp.com>



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

ISO/IEC 14882:2017  
(C++ 17)

Fabio Galuppo, M.Sc.

<http://fabiogaluppo.com> e <http://simplycpp.com/>

fabiogaluppo@acm.org

@FabioGaluppo

Microsoft MVP Visual Studio and Development Technologies

<https://mvp.microsoft.com/en-us/PublicProfile/9529>

<http://bit.ly/devxp2017>