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

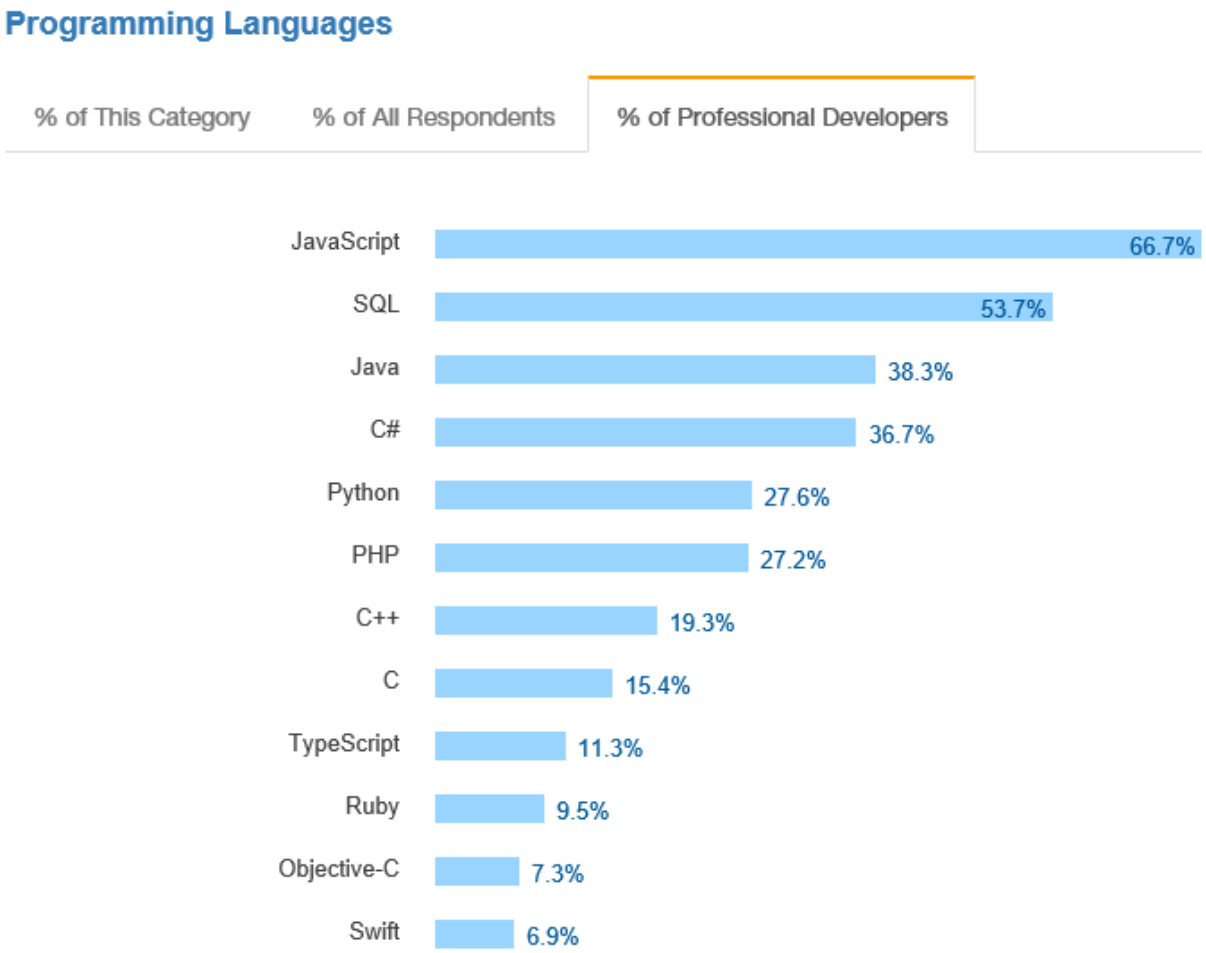
Microsoft MVP Visual Studio and Development Technologies

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

http://bit.ly/XV_Jornada_UNESP_Bauru

Stack Overflow

Developer Survey Results 2017



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

TIOBE Index for May 2017

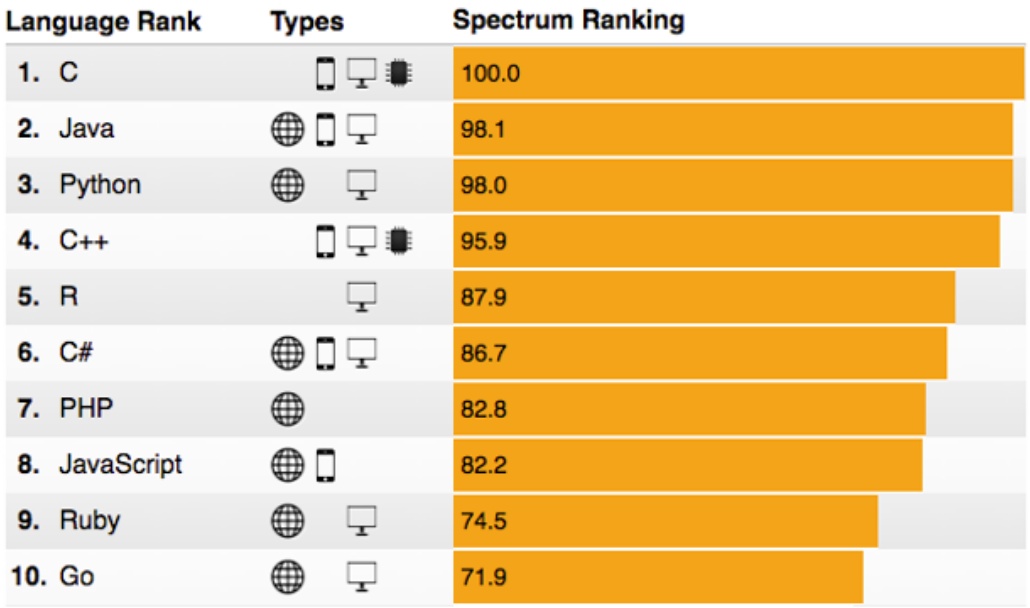
Java and C are in a heavy downward trend since the beginning of 2016. Both languages have lost more than 6% if compared to last year. So which programming languages are taking advantage of this drop? Well, actually all the other languages. Since software is adopted by more and more domains nowadays, 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	v	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	v	PHP	2.693%	-0.30%
10	9	v	Perl	2.602%	+0.28%
11	8	v	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	v	Objective-C	2.101%	+0.50%
16	42	^^	Go	2.080%	+1.83%

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?



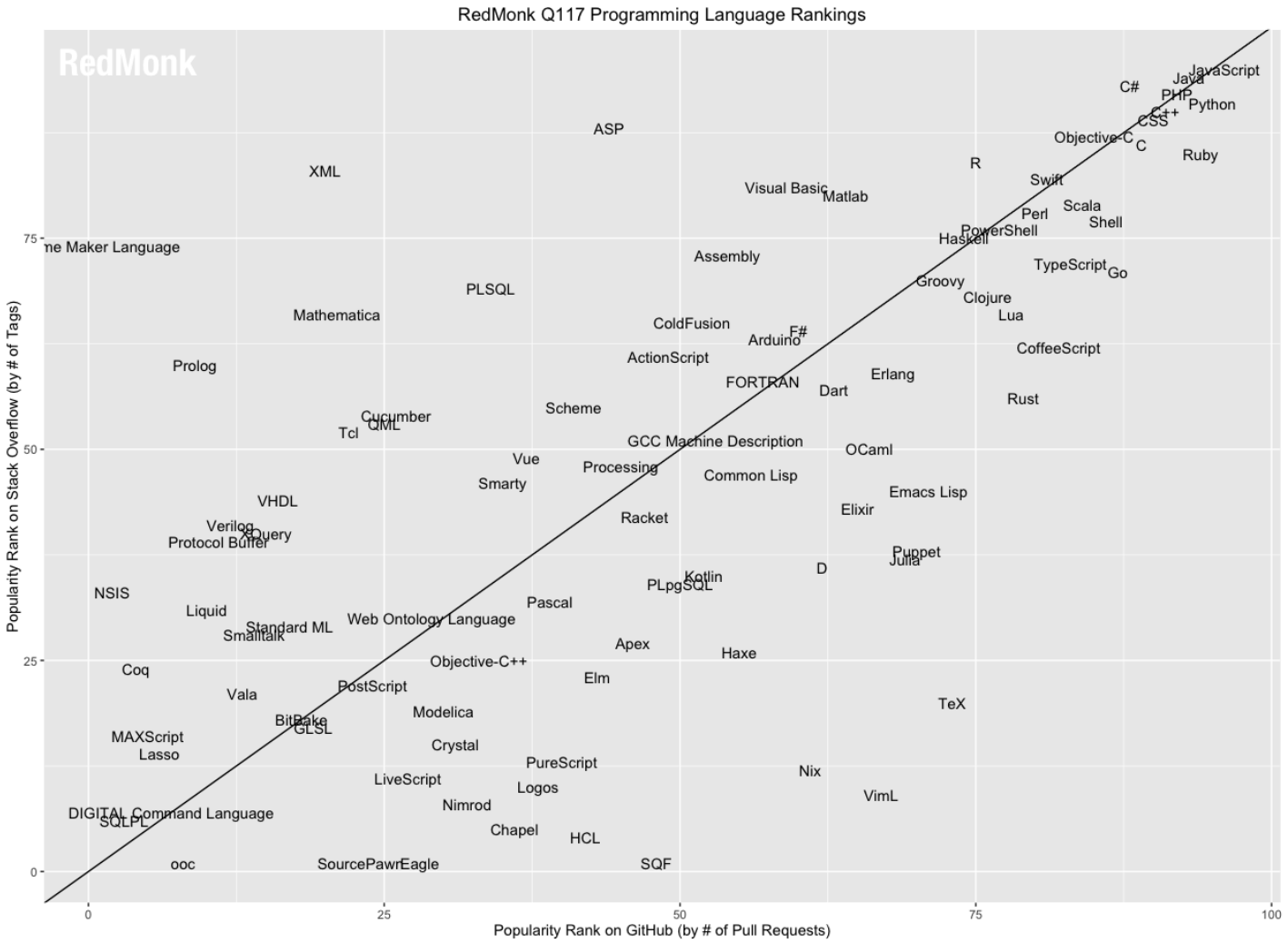
Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

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

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	cpu load
C++ g++	**9.31**	1,052	1544	9.30	1% 1% 1% 100%
C gcc	9.96	1,016	1490	9.95	1% 1% 1% 100%

mandelbrot

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

binary-trees

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

pidigits

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

spectral-norm

source	secs	mem	gz	cpu	cpu load
C++ g++	**1.99**	1,880	1044	7.89	100% 99% 99% 99%
C gcc	1.99	1,824	1139	7.88	99% 99% 99% 100%

fasta

source	secs	mem	gz	cpu	cpu load
C++ g++	1.48	4,176	2313	5.25	89% 90% 88% 89%
C gcc	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
C++ g++	**2.36**	114,404	809	7.69	86% 71% 87% 85%
Java	11.26	593,156	835	39.02	85% 88% 90% 88%

mandelbrot

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

n-body

source	secs	mem	gz	cpu	cpu load
C++ g++	**9.31**	1,052	1544	9.30	1% 1% 1% 100%
Java	21.54	27,092	1489	21.56	1% 1% 100% 1%

spectral-norm

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

pidigits

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

fasta

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

http://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=gpp&lang2=java

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

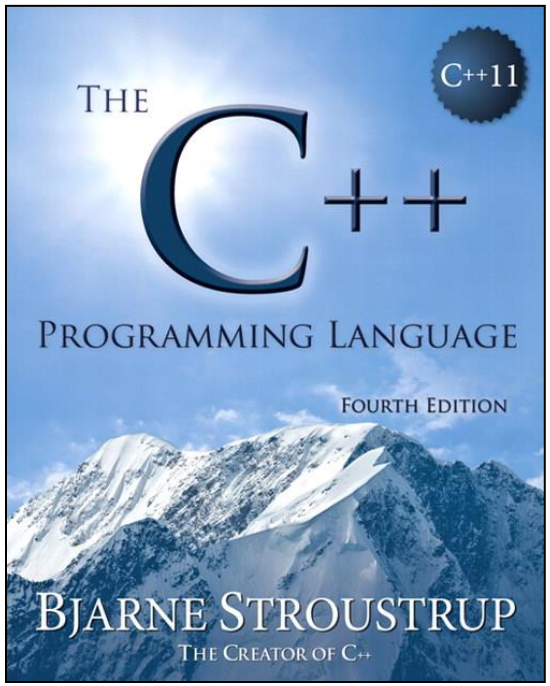
Fonte: http://www.stroustrup.com/

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.



Domain-specific abstraction

Fortran

Cobol

General-purpose abstraction

Simula

Java

C++

C++11

C#

Direct mapping to hardware

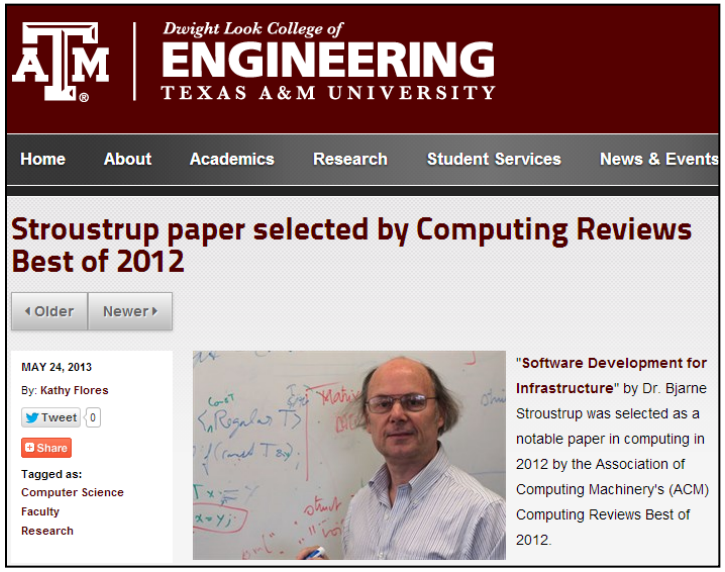
Assembler

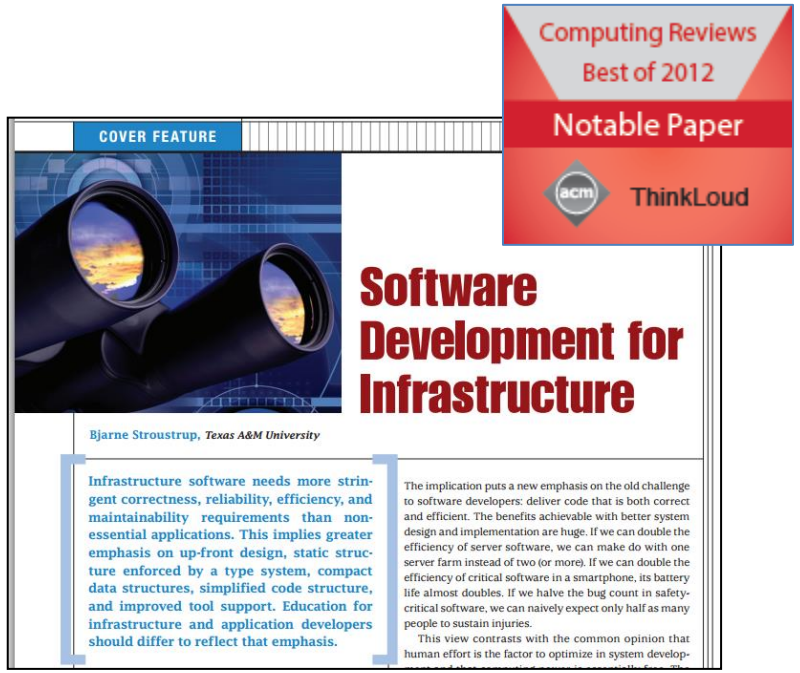
BCPL

C

Fonte: http://www.stroustrup.com/

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"

C++ 11
The Future is here

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

C++: o início

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

1979 April
Work on C with Classes began

1979 October
First C with Classes (Cpre) running

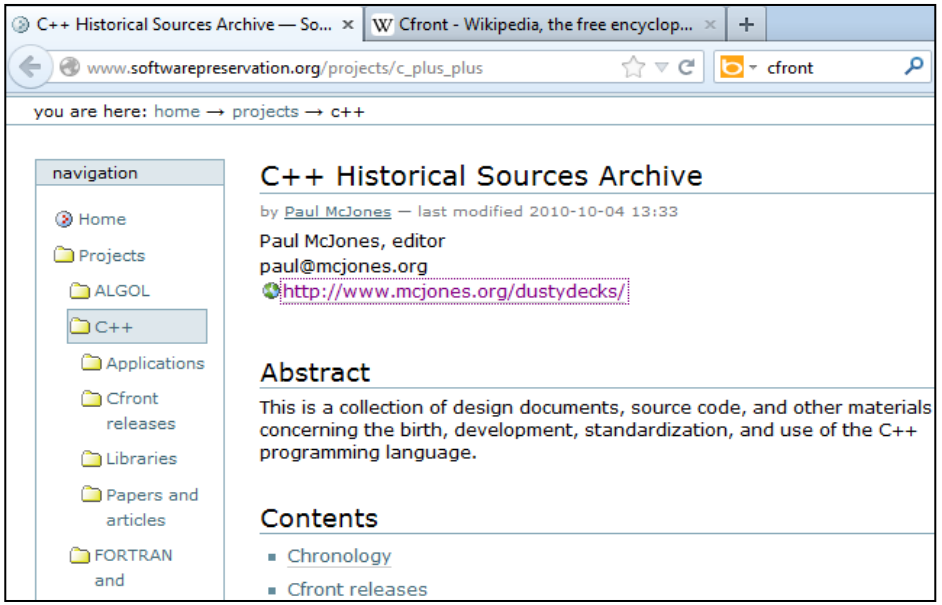
1983 August
First C++ in use at Bell Labs

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

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



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)

Linguagem + Biblioteca

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

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

C++ 14: ISO/IEC 14882:2014

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

!“Hello, World!”

```cpp
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!”

limit_cpp17.cpp

CppPlayground (Global Scope)

```
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
```

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
```

CppPlayground Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuratio

- 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

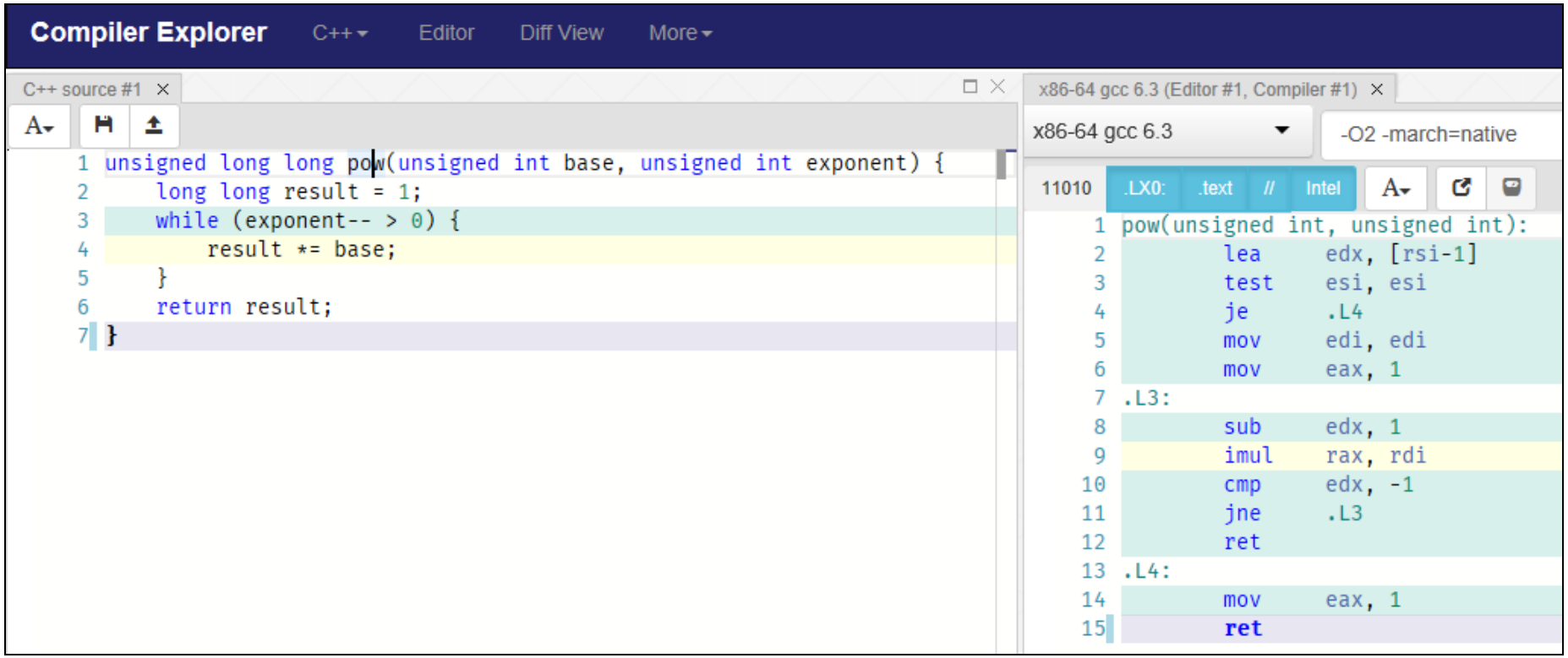
Disable Language Extensions	No
Treat WChar_t As Built in Type	Yes (/Zc:wchar_t)
Force Conformance in For Loop Scope	Yes (/Zc:forScope)
Remove unreferenced code and data	Yes (/Zc:inline)
Enforce type conversion rules	
Enable Run-Time Type Information	
Open MP Support	
C++ Language Standard	ISO C++ Latest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

C++ Language Standard
Determines the C++ language standard. (/std:c++14, /std:c++latest)

OK Cancel

https://www.visualstudio.com/

C++ Explorer



https://godbolt.org/

Eu quero aprender C++!
Por onde começar?

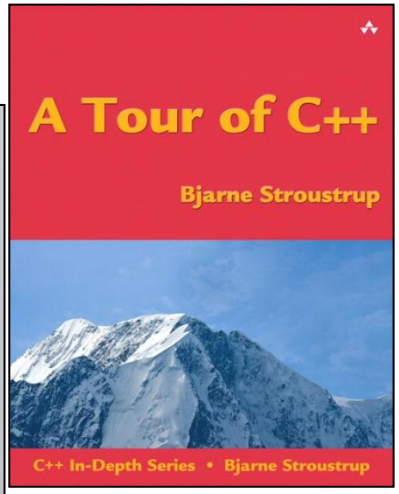
Bjarne Stroustrup
The Essence of C++: With Examples in C++84, C++98, C++11, and C++14

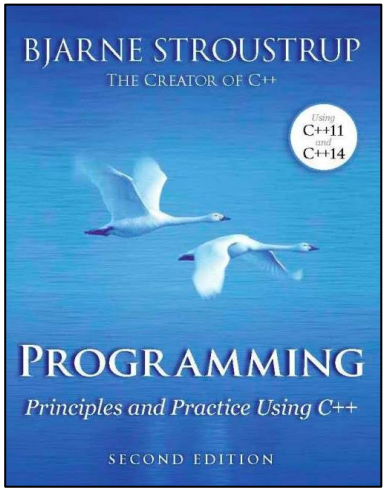
Abstract

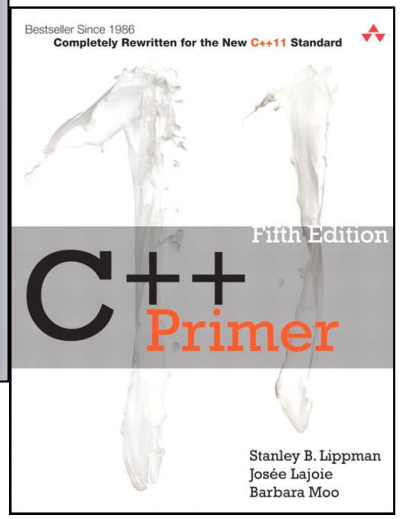
- C++11 is being deployed and the shape of C++14 is becoming clear. This talk examines the foundations of C++. What is essential? What sets C++ apart from other languages? How does new and old features support (or distract from) design and programming relying on this essence.
- I focus on the abstraction 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 (?)

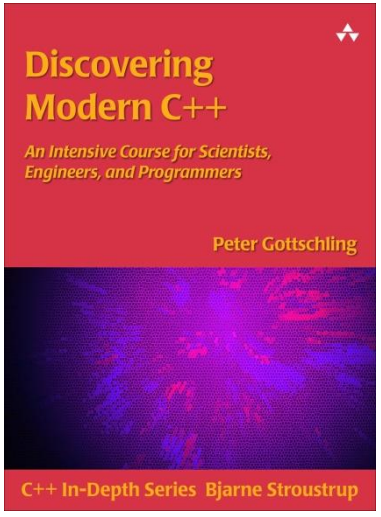
Stroustrup - Essence - Going Native'13 2

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

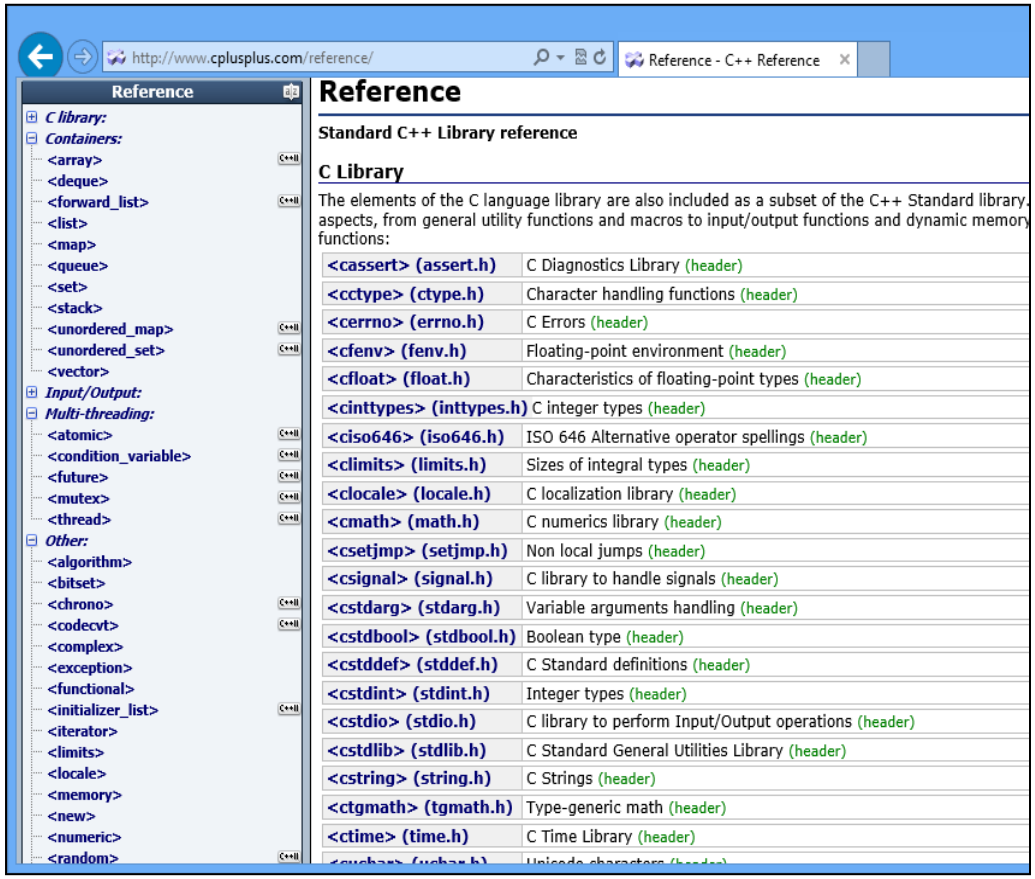






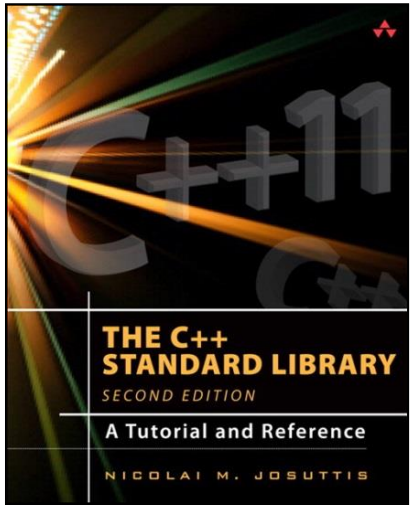


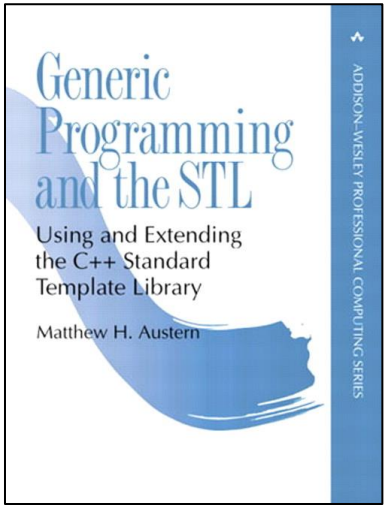
Standard Template Library (STL)



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
<ios>	hex,dec,scientific,fixed,defaultfloat	§iso.27.5
<iostream>	istream, ostream, cin, cout	§iso.27.4
<map>	map, multimap	§iso.23.4.4
<memory>	unique_ptr, shared_ptr, allocator	§iso.20.6
<random>	default_random_engine, normal_distribution	§iso.26.5
<regex>	regex, smatch	§iso.28.8
<string>	string, basic_string	§iso.21.3
<set>	set, multiset	§iso.23.4.6
<sstream>	istrstream, ostrstream	§iso.27.8
<stdexcept>	length_error, out_of_range, runtime_error	§iso.19.2
<thread>	thread	§iso.30.3
<unordered_map>	unordered_map, unordered_multimap	§iso.23.5.4
<utility>	move(), swap(), pair	§iso.20.1
<vector>	vector	§iso.23.3.6



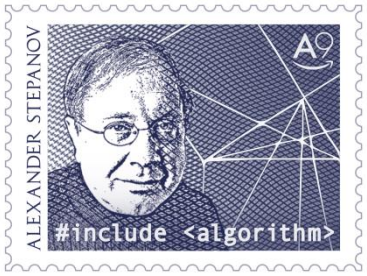


Stepanov, STL e Programação Genérica

- 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.

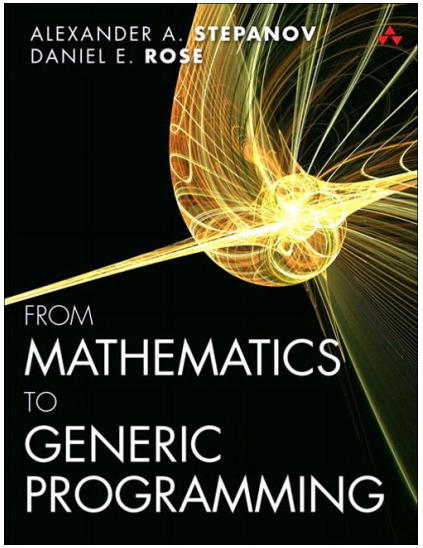
https://www.sgi.com/tech/stl/stl_introduction.html

- 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

#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>
bool binary_search (ForwardIterator first, ForwardIterator last,
                    const T& val);
```

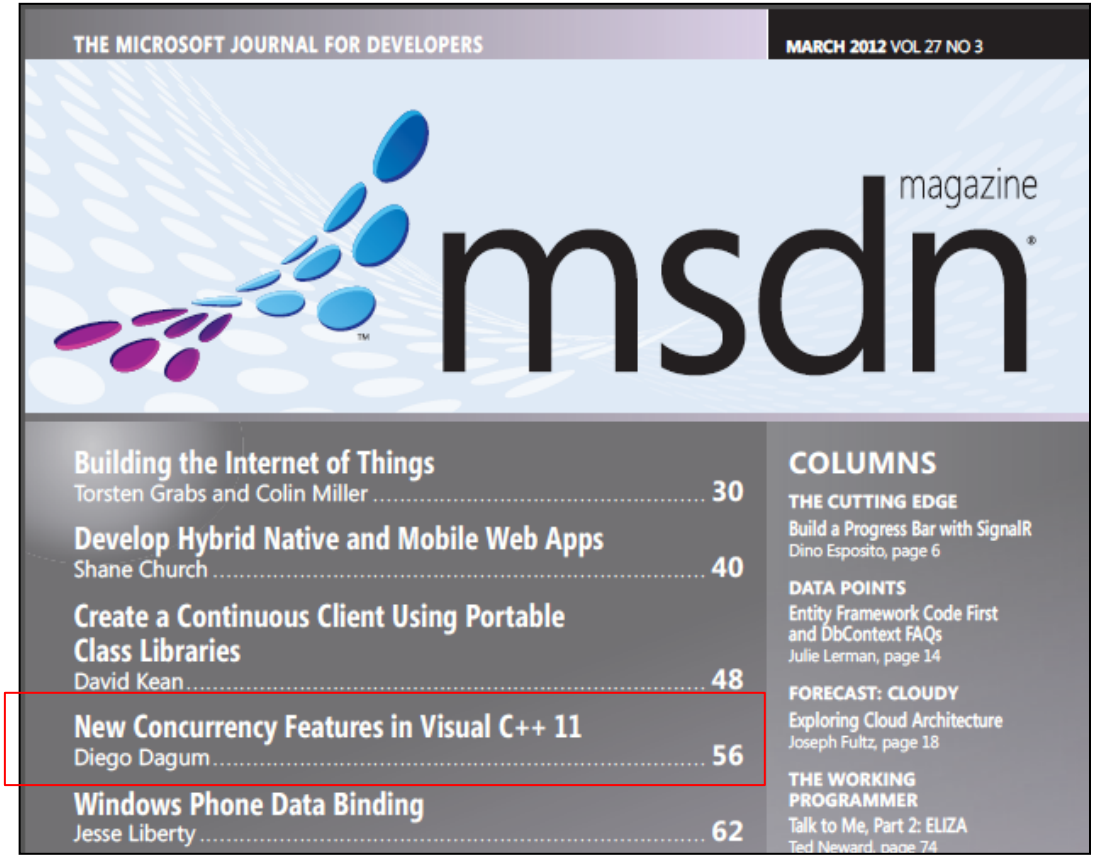
default (1)

```
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/

Programação Concorrente com C++

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



library
Multi-threading

Atomic and thread support
Support for atomics and threads:

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

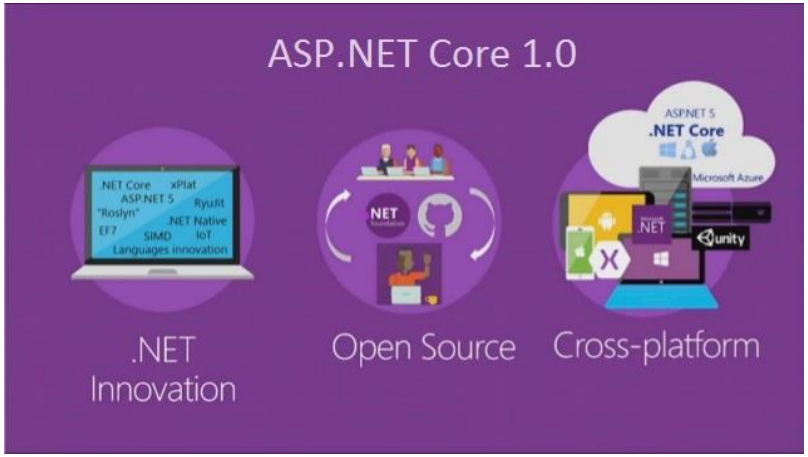
DIEGO DAGUM *is a software developer with more than 20 years of experience. He's currently a Visual C++ community program manager with Microsoft.*

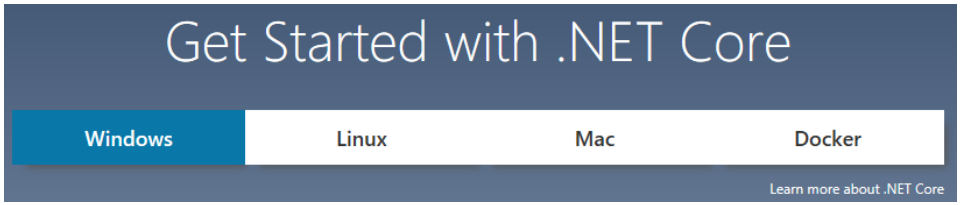
THANKS to the following technical experts for reviewing this article: *David Cravey, Alon Fliess, Fabio Galuppo and Marc Gregoire*

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++?

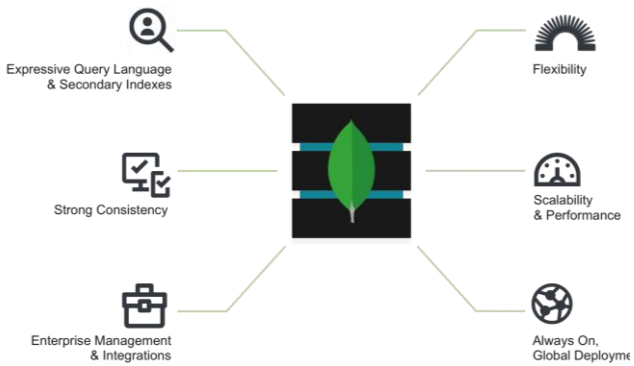




amd64	[x64/Linux] Use correct argument registers in InterpreterStub (#12002)
arm	Finish deleting dead CAS code from CoreLib (#11436)
arm64	[Arm64/Win] Revise JIT_MemSet (#11420)
coreclr	Initial change to support System.Private.CoreLib.dll as Core Library.
crossgen	Finish deleting dead CAS code from CoreLib (#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	Finish deleting dead CAS code from CoreLib (#11436)
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++?

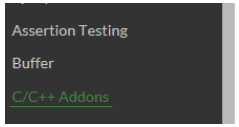


background.cpp	SERVER-23971 Clang-Format code	a year ago
background.h	SERVER-23971 Clang-Format code	a year ago
client.cpp	SERVER-28201 Allow detatching of the current Client	a month ago
client.h	SERVER-28201 Allow detatching of the current Client	a month ago
clientcursor.cpp	SERVER-28327 Revamp cursor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

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



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.

> 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++?

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

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

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

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

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/

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

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

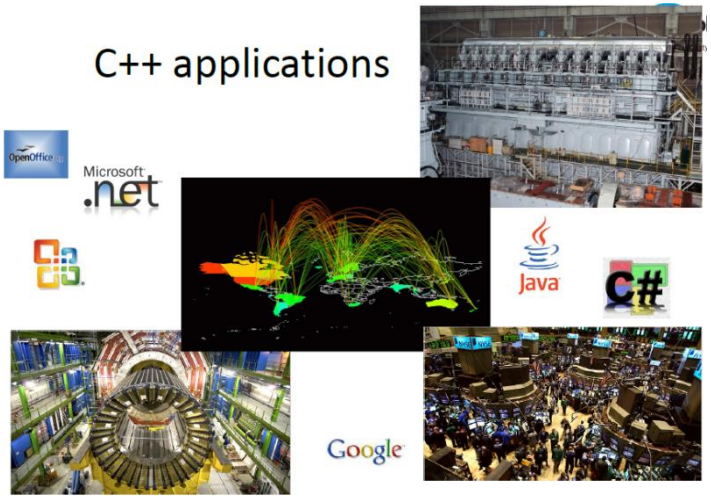
by Robert Hodgin, Bloom Studio

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

TÓPICOS ADICIONAIS SOBRE C++

Mais aplicações de C++



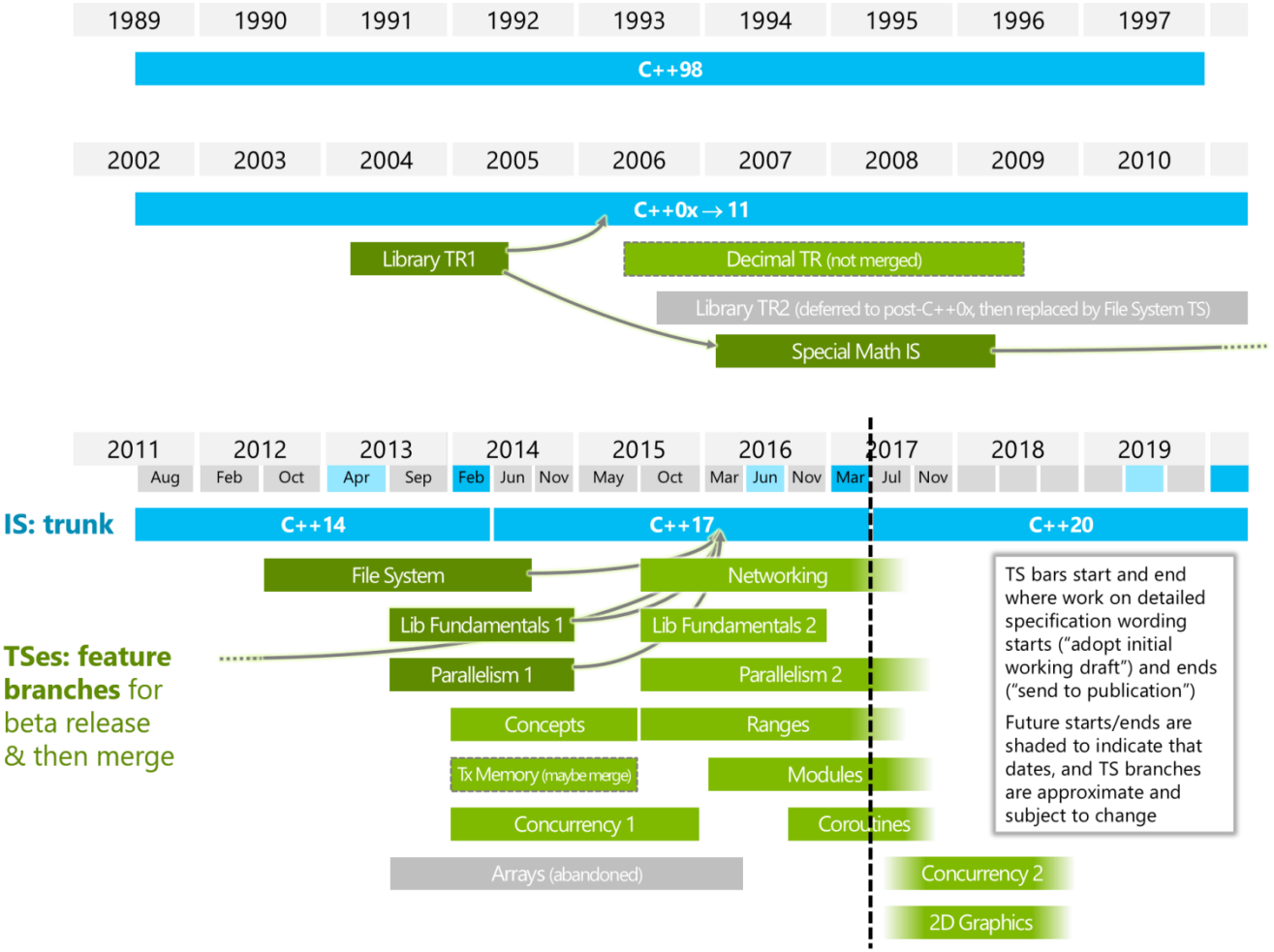
http://www.stroustrup.com/applications.html

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

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

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

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=NvWTnIoQZj4

Bjarne Stroustrup: The 5 Programming Languages You Need to Know

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

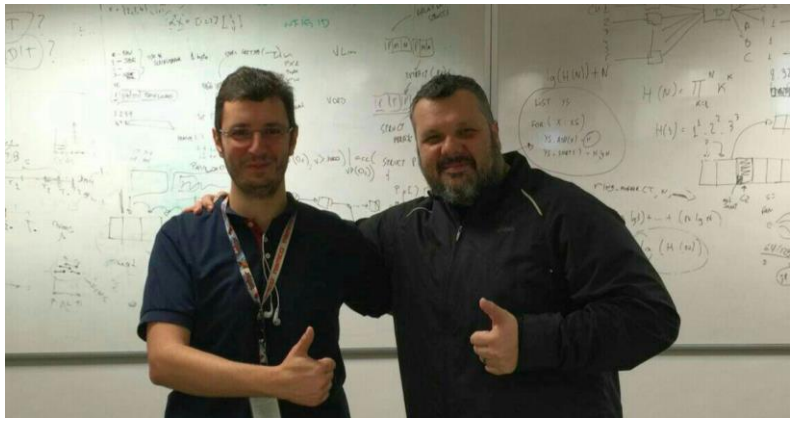
...**C++**, of course; **Java**; maybe **Python** for mainline work... And if you know those, you can't help know sort of a little bit about **Ruby** and **JavaScript**, you can't help knowing **C** because that's what fills out the domain and of course **C#**. But again, **these languages create a cluster**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

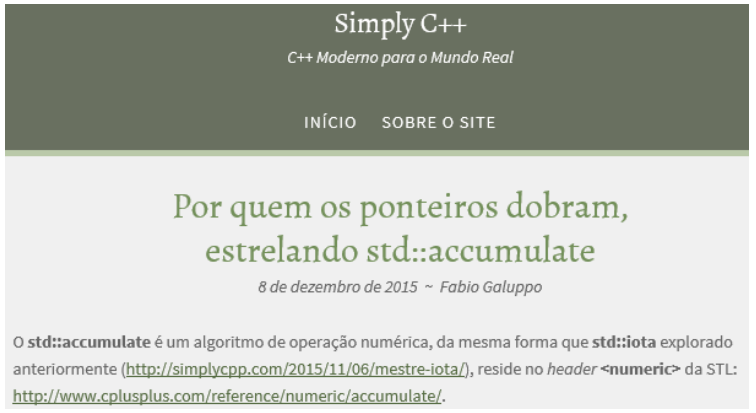
Se quiser saber mais sobre:

C++
Programação Genérica
Iterators
Policy-based Design
Algoritmos
...

visite:

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/XV_Jornada_UNESP_Bauru