

C++ Today

The Beast is Back

C++ Today

The Beast is Back

Jon Kalb

O'REILLY®

C++ Today

The Beast is Back



Jon Kalb & Gašper Ažman

O'REILLY®

C++ Today

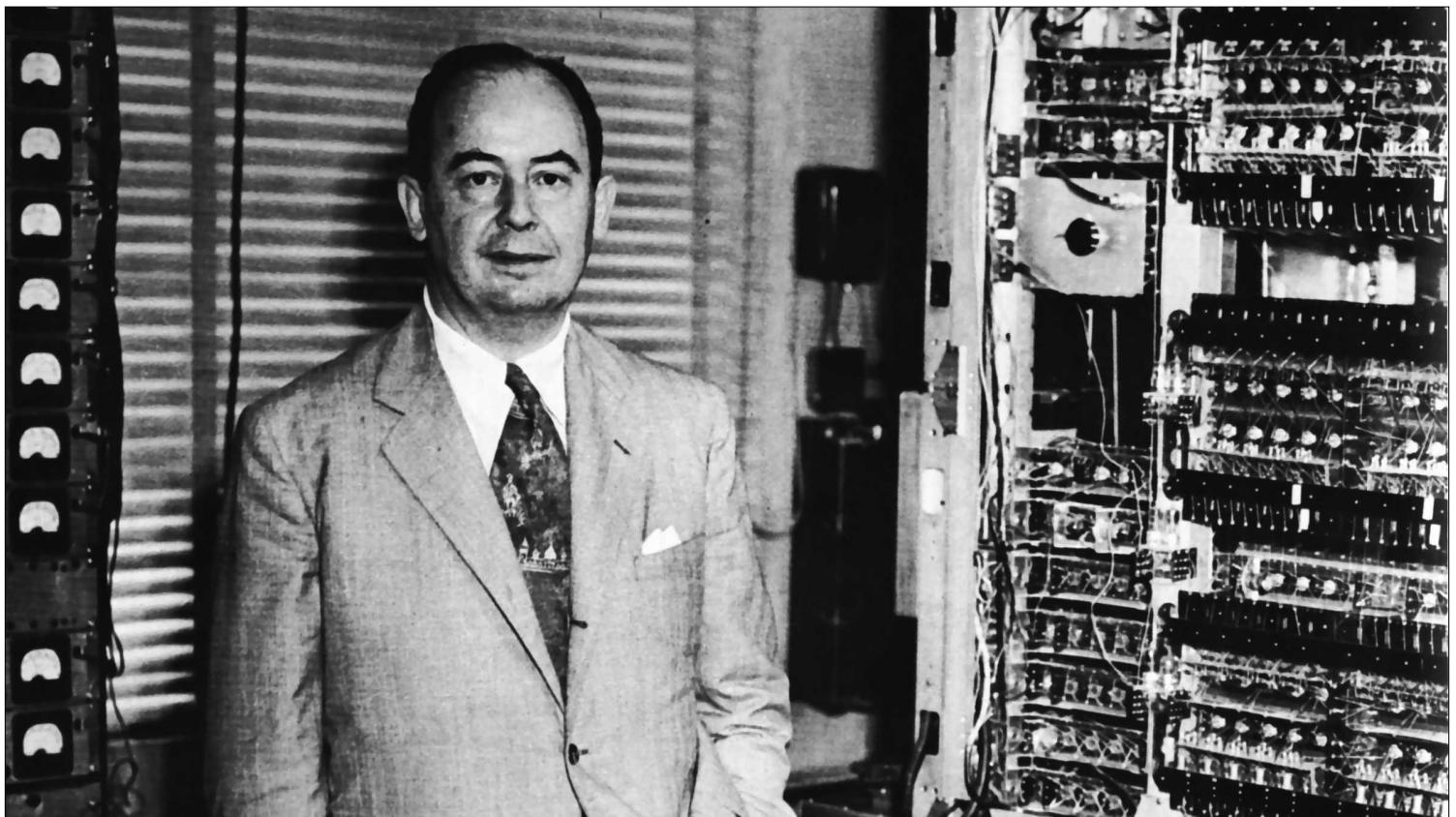
The Beast is Back



Jon Kalb & Gašper Ažman

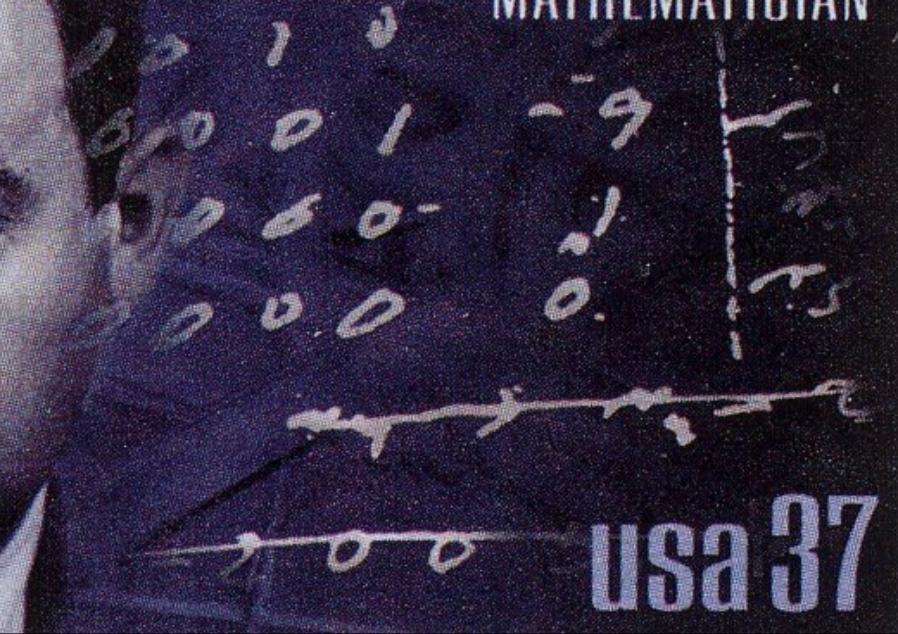
Free Download
(with email address)

O'Reilly Media:
j.mp/Cpp_Today

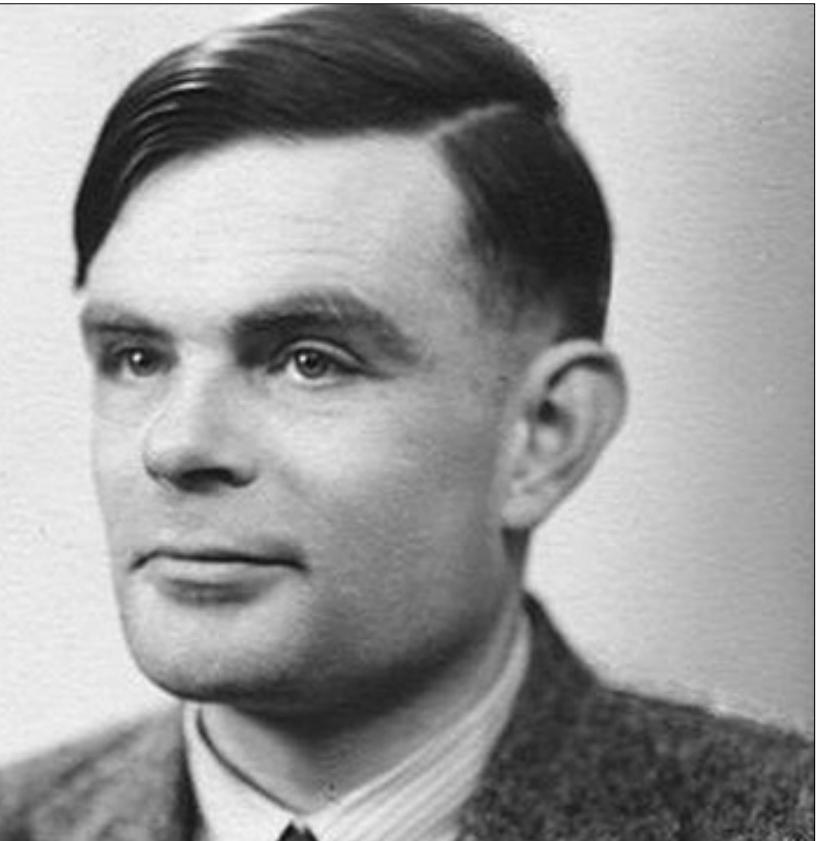


JOHN von NEUMANN

MATHEMATICIAN



38:A1 DE LDA (\$DE,X) } (ANY2+A)
3A:91 DE STA (\$DE),Y } = (ANY2)
3C:A9 DE
3E:20 10 0E LDA #DE } ANY2=ANY2
 JSR DECZPG } -1 } FOR-NEXT
41:A5 DE LDA \$DE MEMORY MOVING
43:C5 DB CMP \$DB ROUTINE.
45:D0 F1 BNE \$0E38
47:A5 DF LDA \$DF
49:C5 DC CMP \$DC
4B:D0 EB BNE \$0E38
4D:A1 DE LDA (\$DE,X)
4F:91 DE STA (\$DE),Y
51:~~10000000~~ 88
52:~~00000000~~ DEY
53:~~00000000~~ DEY
55:C0 00
56:A9 20 LDA #\$20
58:91 DB STA (\$DB),Y } Stores spaces (\$20) in ne
59:C0 00



Fortran

i AAC(I,KK)=AC(I,KK)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

HUMMEL KG. 1377 B 14

```

C Uniform Deviates
C p. 199
C Return Uniform Random Number Between 0.0 AND 1.0
C Set IDUM to any negative value to initialize the sequence
REAL FUNCTION RAN3 (IDUM) RESULT (R)
INTEGER*2 IDUM, I, II, J, K
REAL R, FAC
SAVE
INTEGER*4 MBIG, MSEED, MZ, MA(55), IFF, MJ, MK, INEXT, INEXTP
PARAMETER (MBIG=1000000000, MSEED=161803398, MZ=0, FAC=1.0/MBIG)
DATA IFF /0/

```



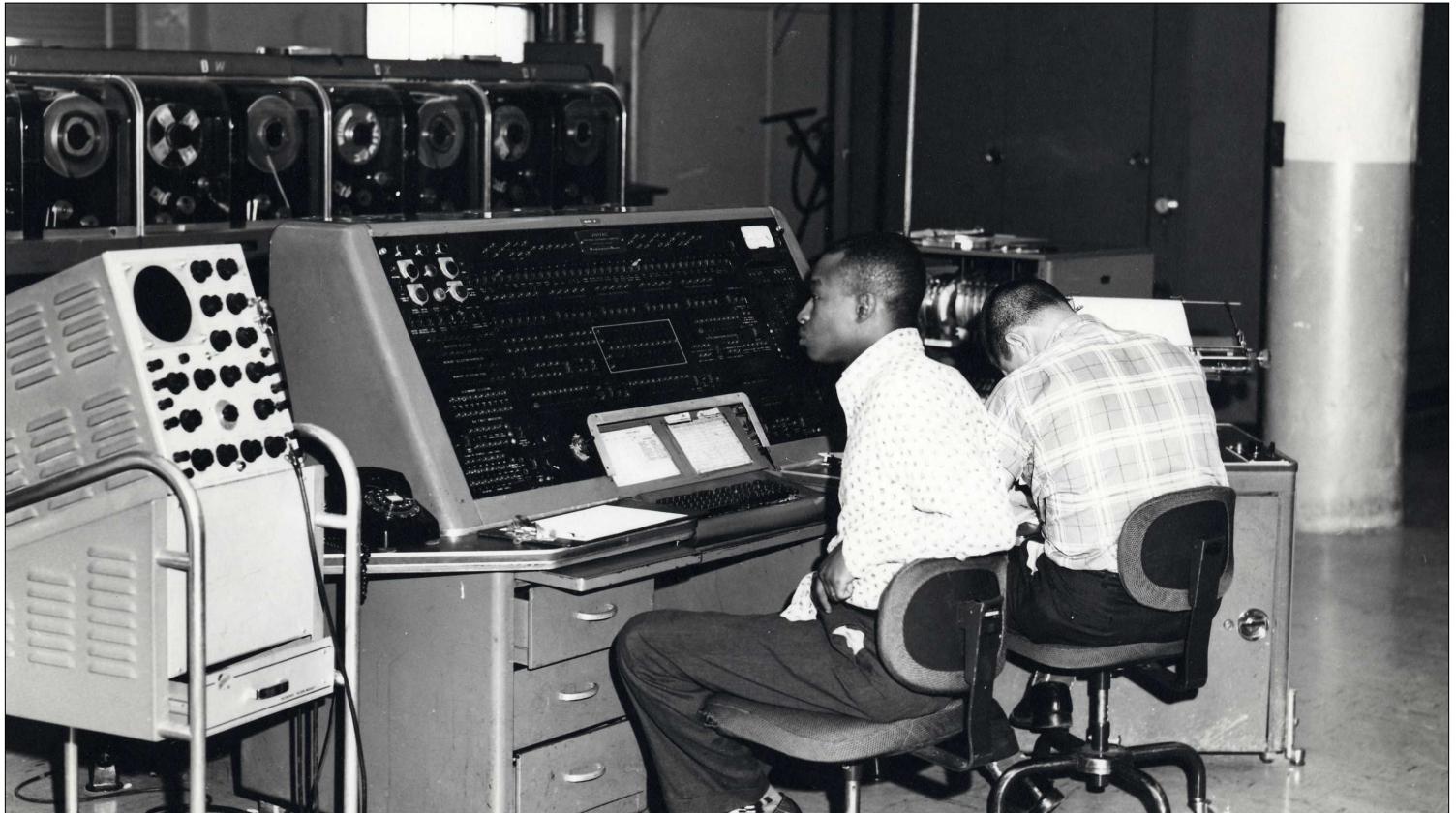
COBOL PROGRAMMING

INCLUDING MS-COBOL AND COBOL-85

Second Edition

M K ROY
D GHOSH DASTIDAR

```
//COBULG JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//HELOWRLD EXEC COBULG,PARM.COB='MAP,LIST,LET'
//COB.SYSIN DD *
 001  IDENTIFICATION DIVISION.
 002  PROGRAM-ID.  'HELLO'.
 003  ENVIRONMENT DIVISION.
 004  CONFIGURATION SECTION.
 005  SOURCE-COMPUTER.  IBM-360.
 006  OBJECT-COMPUTER.  IBM-360.
 0065  SPECIAL-NAMES.
 0066      CONSOLE IS CNSL.
 007  DATA DIVISION.
 008  WORKING-STORAGE SECTION.
 009    77 HELLO-CONST  PIC X(12) VALUE 'HELLO, WORLD'.
 075  PROCEDURE DIVISION.
 090    000-DISPLAY.
 100        DISPLAY HELLO-CONST UPON CNSL.
 110        STOP RUN.
//LKED.SYSLIB DD DSNNAME=SYS1.COBLIB,DISP=SHR
//                      DD DSNNAME=SYS1.LINKLIB,DISP=SHR
//GO.SYSPRINT DD SYSOUT=A
//
```







IBM SYSTEM/360

Now one new computer fills all your data processing needs

You can easily increase the size of SYSTEM/360 when your business grows or you want to add new applications.

You don't have to revise most of your programs. You don't have to switch to new input and output devices.

Any program that works on the smallest configuration can work on the largest.

Same goes for the programming systems. The simplest operating system, the simplest language translator or object program can work on any SYSTEM/360.

Same goes for input and output devices. Any printer, tape storage unit, reader or terminal that works in a small configuration works in a larger one. You choose what you need now. You add new components when you need them.

This is true from the smallest configuration to the largest configuration.

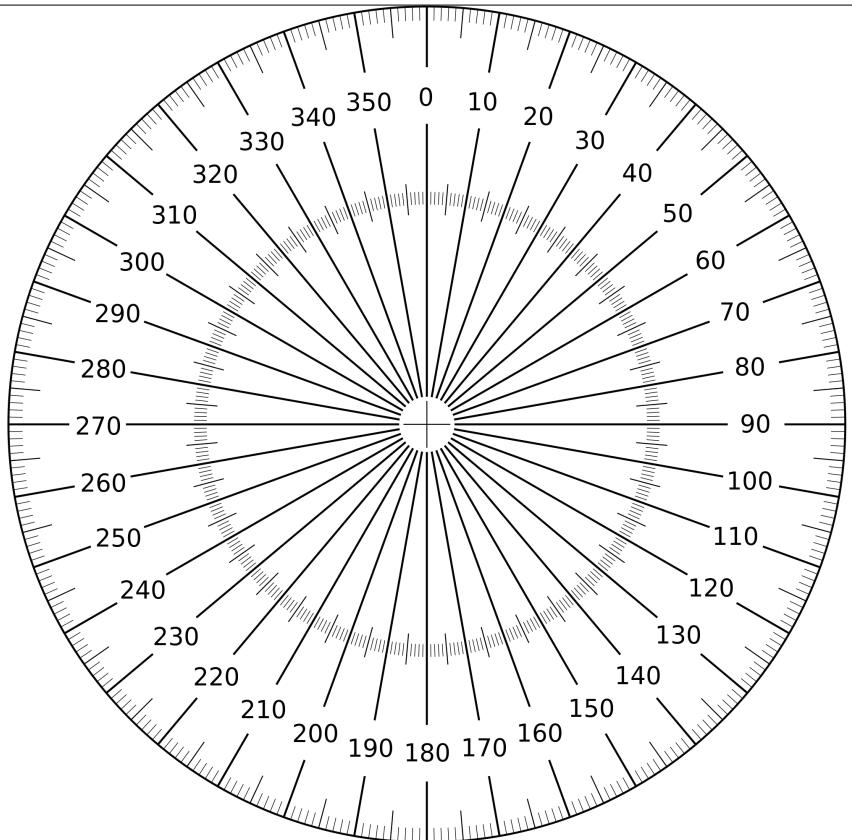
SYSTEM/360 solves today's problems. And it expands to solve tomorrow's problems, too.

It cuts today's costs...and it will also cut tomorrow's. There's never been a system quite like it.

IBM
DATA PROCESSING



Little in U.S.A. 510-0958

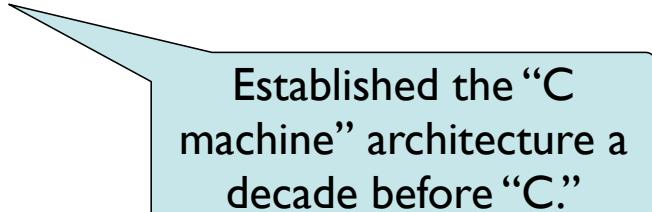


IBM SYSTEM/360

- Designed to be a machine for science and business
- Featured binary, decimal (BCD), and hexadecimal floating-point calculations
- First instruction set implemented in microcode
- 8-bit byte addressing

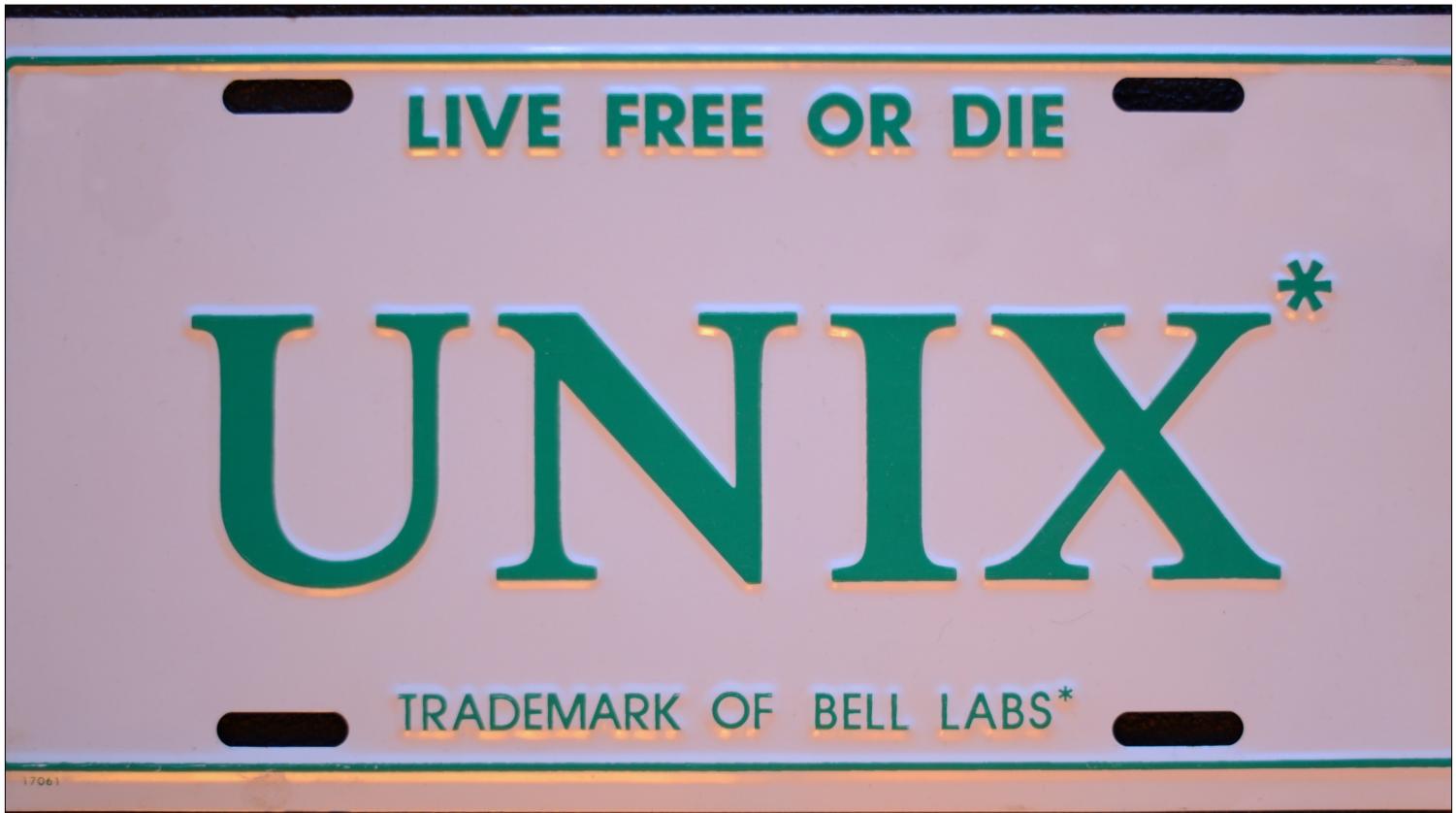
IBM SYSTEM/360

- Designed to be a machine for science and business
- Featured binary, decimal (BCD), and hexadecimal floating-point calculations
- First instruction set implemented in microcode
- 8-bit byte addressing



Established the “C machine” architecture a decade before “C.”



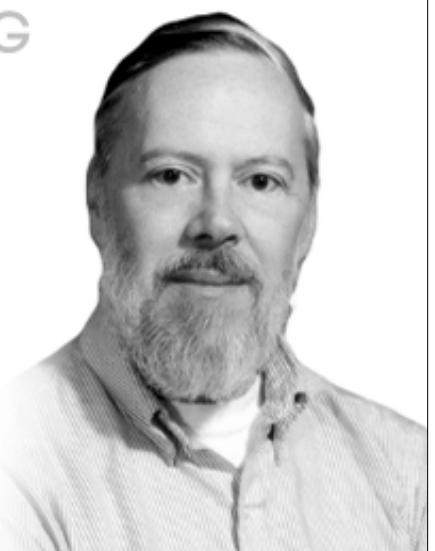


THE C PROGRAMMING LANGUAGE



C
PROGRAMMING
LANGUAGE
CREATOR

DENNIS RITCHIE
1941 - 2011





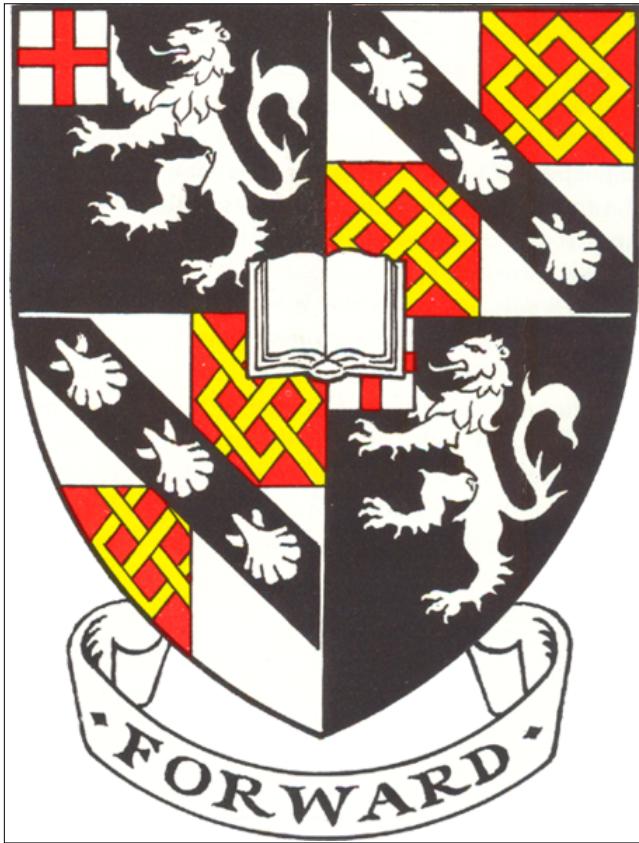
Turing Award

For ideas fundamental to the emergence of object-oriented programming, through their design of the programming languages Simula I and Simula 67



Dahl and Nygaard at the time of Simula's development



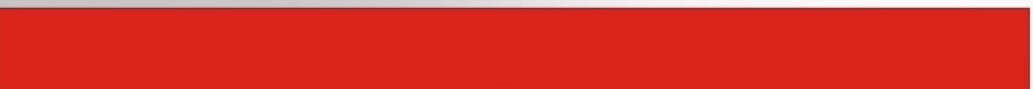


“So I rewrote my simulator from Simula into BCPL and all of the high-level structure disappeared. All of the nice organization that had helped me debug and helped me design disappeared. But the resulting BCPL, once I had debugged it and lost half of my hair in the process, ran really fast. It could use all of the resources of the machine. It could communicate with anything on the machine and I got my data and I got my PhD...

“I came away with the opinion that I would never again want to attack a problem with tools that [were] fundamentally unsuitable. And, in particular, I don't want to make the choice between elegant, which Simula was for this problem, and efficient, which BCPL was. I want both. And that has been sort of one of my guiding lights. If you give people the choice of writing good code or fast code there's something wrong. Good code should be fast.”



C / C++



THE
C++

PROGRAMMING LANGUAGE

Why C++?

- High-Level Abstractions at Low Cost
- Low-Level Access When You Need It
- Wide Range of Applicability
- Highly Portable
- Better Resource Management

High-Level Abstractions at Low Cost

- user-defined types
- type templates
- generic algorithms
- type aliases
- type inference
- compile-time introspection
- runtime polymorphism
- exceptions
- deterministic destruction...

High-Level Abstractions at Low Cost

- user-defined types
 - power and expressiveness of built-in types
 - almost anything that can be done with a fundamental type, can be done with a user-defined type
 - for example: a programmer can define a type that functions as a
 - fundamental arithmetic type (operator overload supports mathematical operations)
 - object pointer (smart pointer)
 - function pointer (function object)

High-Level Abstractions at Low Cost

- No programming paradigm is forced on the user.
- Instead there is support for:
 - procedural
 - object-based
 - object-oriented
 - generic
 - functional
 - value semantics
- freely mixing any of these paradigms

High-Level Abstractions at Low Cost

- Libraries!
- C++ can be thought of a language for making libraries
- Libraries can be created that have:
 - power
 - efficiency
 - safety
 - natural syntax
 - type-specific optimizations
 - automatic resource management
 - generic features that are as efficient as custom code

High-Level Abstractions at Low Cost

- Little or no abstraction penalty
- Bjarne Stroustrup refers to his goal as:

“the zero-overhead principle”

If you don't use the feature, you pay nothing.

If you do use it, you pay no more than you would if you coded it by hand.

Low-Level Access When You Need It

- systems-programming language:
 - low-level hardware control
 - responding to hardware interrupts
 - device drivers
 - manipulate memory in arbitrary ways — down to the bit level
 - on par with assembler, but if you really need it, allows inline assembly code
- all of this comes free with being a superset of “C”

Low-Level Access When You Need It

- management of user-defined type
- most languages create objects by
 - allocating memory from the heap
 - running construction function
- C++ can construct
 - on the heap
 - static memory
 - stack memory
 - arbitrary memory

Low-Level Access When You Need It

- can be cache-optimized
 - control necessary to exploit caches
 - avoid false sharing
- most managed language containers
 - don't hold memory in contiguous memory
 - can't exploit look-ahead buffers
- C++ containers can do this
 - arrays, vectors, deques,
 - user-defined containers

Wide Range of Applicability

- don't solve the specific problem, solve the general one
- scale
 - software engineers are all about scale
 - algorithms, but also languages
- low end
 - embedded app, device driver, CGIs, mobile apps
 - "you only pay for what you use"
 - low-memory footprint
- high end
 - projects with
 - hundreds of engineers
 - scores of modules
 - separate module compilation

Highly Portable

- “C machine” model
 - minimalistic requirements
 - universally supported by hardware
- one or more C++ toolchains on almost all platforms
- C++ is the only high-level language alternative available on all of the top mobile platforms
- C++ can be written to support all these platforms without rewriting

Highly Portable

- even a “perfect” language has no value if we can’t build it on our target platform
- factors outside the language
 - tool chains
 - analyzers / non-build tools
 - experienced engineers
 - software libraries
 - books / instructional materials
 - trouble shooting support
 - training opportunities
- large installed base / industry support

Better Resource Management

- garbage collection
- the good...
 - no leaks or double dispose
 - very, very annoying problems
- ... and the bad
 - memory not released until unspecified time later
 - ... if at all
 - not under the control of the programmer
 - may result in “freezing”
 - requires a large buffer of unused memory

Better Resource Management

- C++ requires a better solution
- must generalize
 - memory is only one resource
 - other resources need to be similarly managed
 - release must happen immediately
- deterministic destruction
- RAII
 - resource acquisition is initialization
 - responsibility acquisition is initialization
 - every responsibility that must be discharged is
 - an object with appropriate lifetime (usually on the stack)
 - discharged when the object goes out of scope
 - “finally” violates DRY

Michael Caisse

λ ciere



THE 1990s



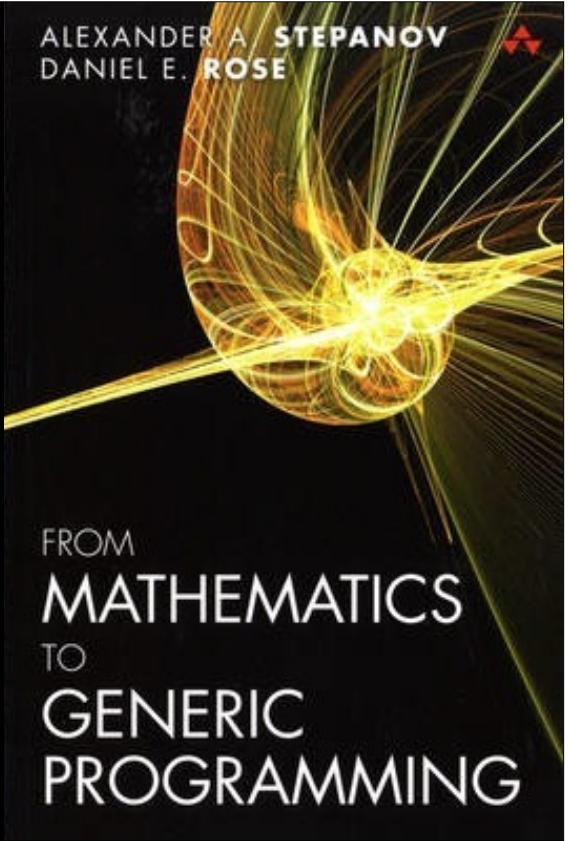
O Object
O Oriented
P Programming



International
Organization for
Standardization



ALEXANDER A. STEPANOV
DANIEL E. ROSE



FROM
MATHEMATICS
TO
GENERIC
PROGRAMMING





The C++ Standard Template Library

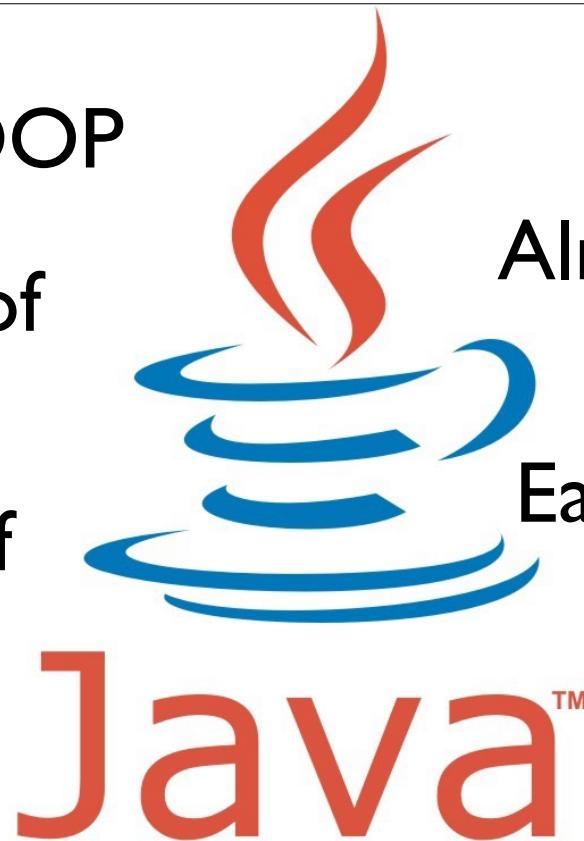




Supports OOP

Language of
the Web

Language of
the Future



Almost as fast

Easier to learn
and teach

Almost as fast

- desktop computers in the 2000s were so powerful that the “need for speed” seemed to be sated
- Why write in a much more complicated language to gain only a little improvement in speed?
- C++ came with
 - a lot baggage from C syntax
 - a lot of complexity to get a little more power
 - templates
 - generic programming
 - operator overloads
 - pointers are hard!
 - managed languages were seen as “fast enough”

Managed Languages

- Two virtual machines
- Java Virtual Machine
 - Java, Scala, Jython, Jruby, Clojure, Groovy
- Microsoft's Common Language Interface
 - C#, F#, IronPython, IronRuby, C++/CLI
- colleges learned that it is easier to teach managed languages than “C machine” languages
- pointers are hard!



International
Organization for
Standardization



International
Organization for
Standardization



Performance Matters

- mobile devices
 - performance with less expensive process
 - better battery performance
- cloud computing
 - performance per dollar of hardware
 - performance per watt of power
 - performance per watt of cooling



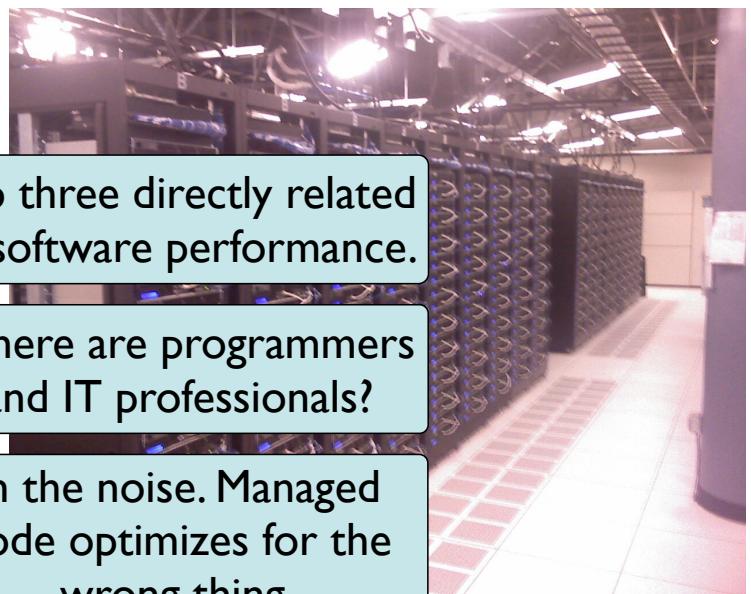
Performance Matters

- James Hamilton of AWS
 - costs of modern high-scale data centers:
 - servers
 - power distribution and cooling
 - power
 - networking equipment
 - other infrastructure

Top three directly related to software performance.

Where are programmers and IT professionals?

In the noise. Managed code optimizes for the wrong thing.

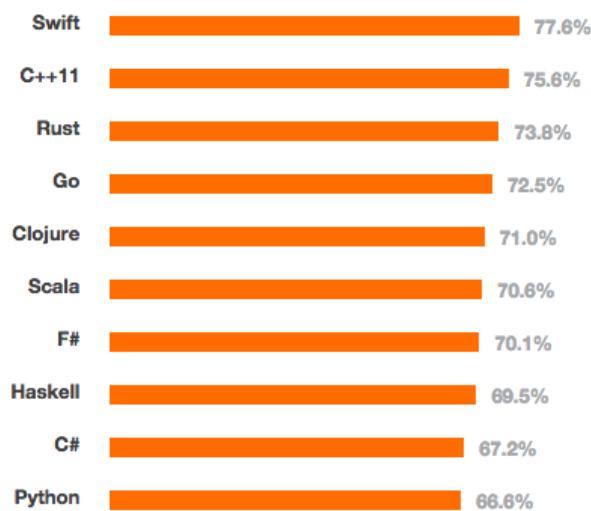




Feels like a new language!

II. MOST LOVED, DREADED, AND WANTED

Most Loved Most Dreaded Most Wanted



C++11

- page count
 - C++03: 776
 - C++11: 1353
- “simplifying” by adding
 - auto
 - range-base for loops
 - >>
 - enums more consistent
 - uniform initialization



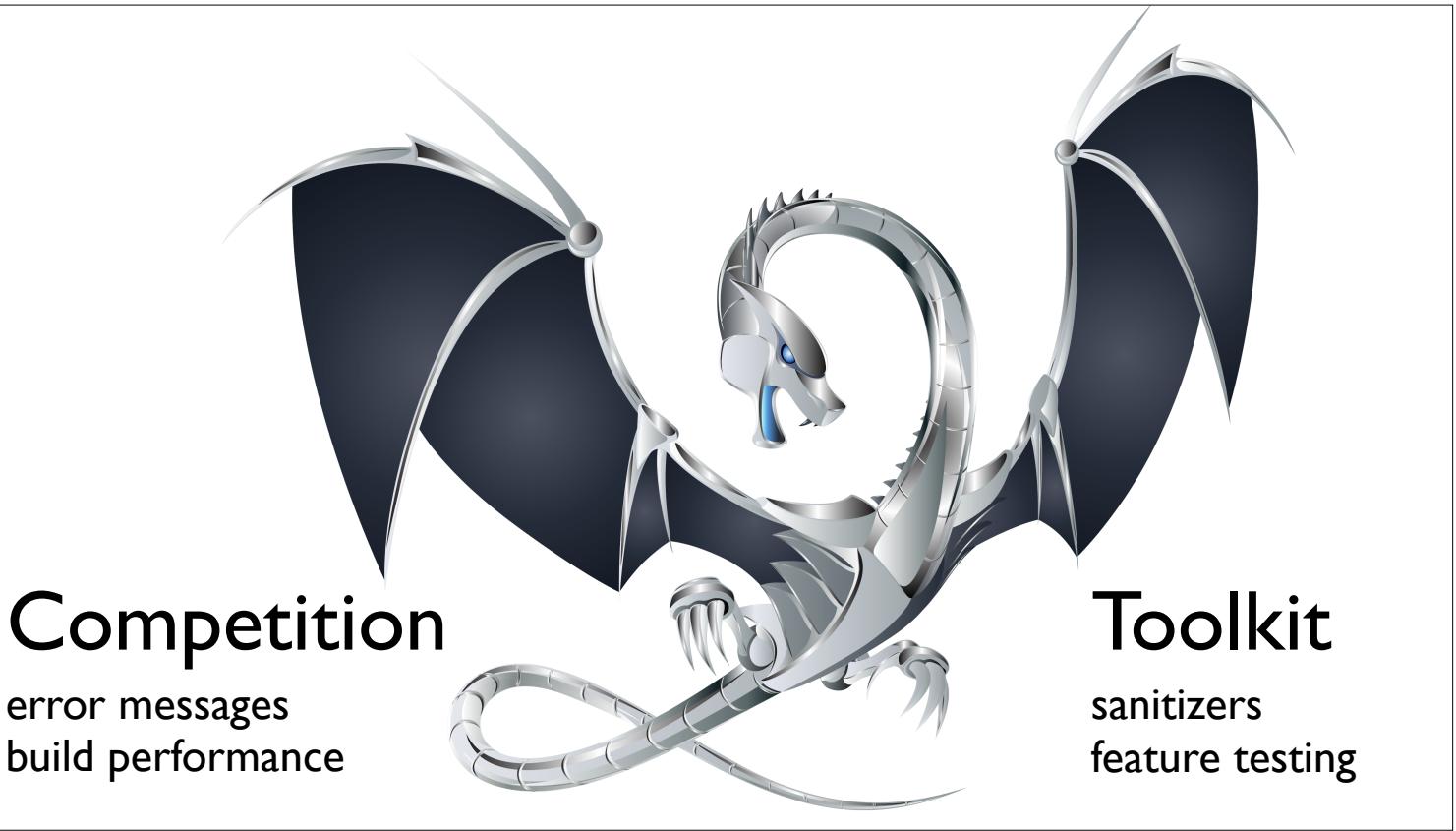
C++11

- lambda expressions
 - simplifying
 - leveraging generic algorithm
 - more functional
- improve support for characters
 - character sets / Unicode
 - custom literals
- “TR1” libraries
 - tuples
 - smart pointers



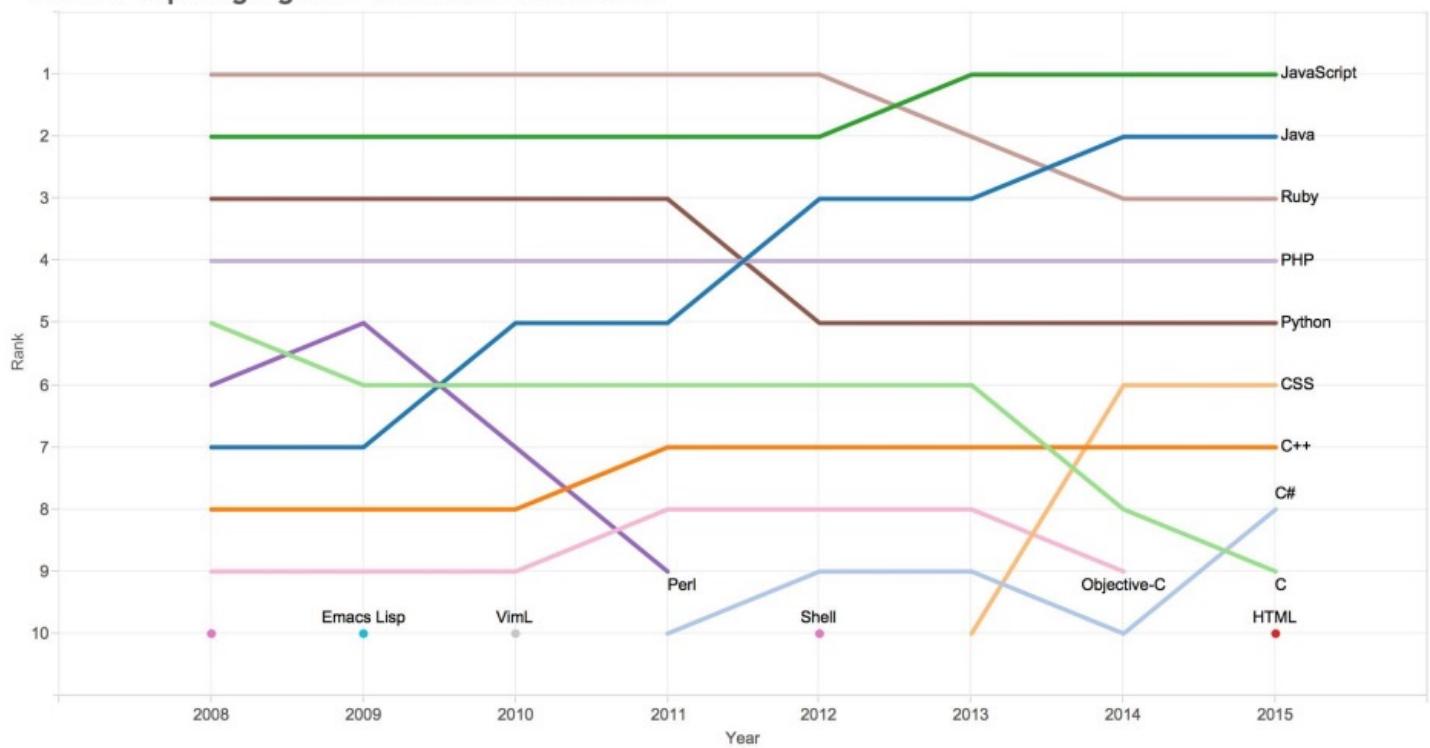
C++11

- better support for library authors
 - better introspection of types
 - variadic templates
 - perfect forwarding
- big new areas
 - multithreading
 - move semantics





Rank of top languages on GitHub.com over time



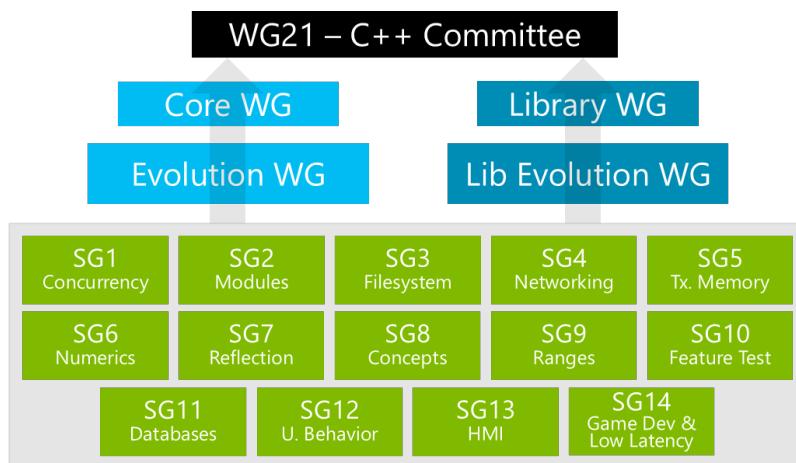
Source: GitHub.com



International Organization for Standardization



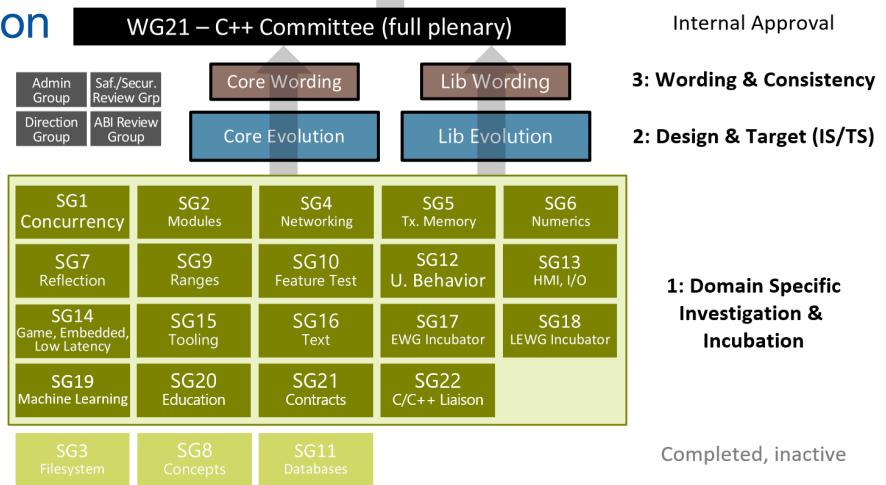
International
Organization for
Standardization





International Organization for Standardization

3-stage pipeline



Completed, inactive

1: Domain Specific Investigation & Incubation

3: Wording & Consistency

2: Design & Target (IS/TS)

(F)DIS Approval

CD & PDTs Approval

Internal Approval

The screenshot shows the homepage of the C++ Super-FAQ website, which is a merger of Marshall Cline's C++ FAQs and Bjarne Stroustrup's C++ FAQ. The site features a large C++ logo at the top, followed by navigation links for Get Started!, Tour, Core Guidelines, Super-FAQ, Standardization, and About. Below the navigation is a section titled "News, Status & Discussion about Standard C++" with a "Follow All Posts" button and a "Recent Highlights" section. A central search bar is present. At the bottom, there is a "C++ FAQ" section with a "Welcome to the C++ Super-FAQ!" message and a note about the merger. The footer includes links for Instapaper, Pocket, and Readability, along with a "View" link.

Just Released

CLion

Cross-platform C/C++ IDE
by JetBrains



C/C++ facts

we learned before bringing you
cross-platform IDE for C and C++

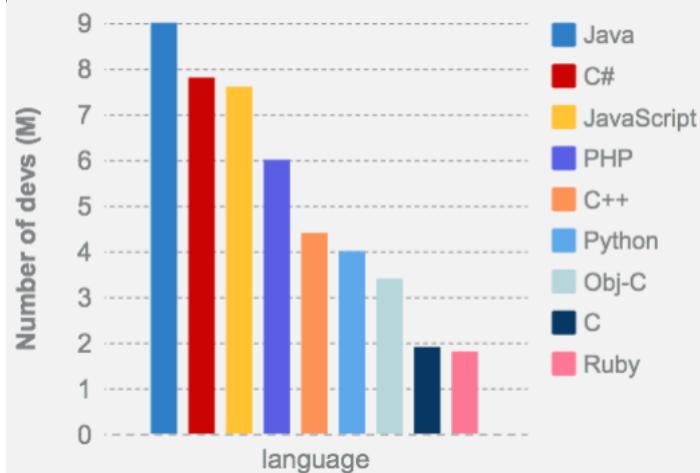
CLion

Sources used:

1. Our user Survey
2. Stackoverflow
3. Job ads: Indeed.com
4. TIOBE index
5. GitHub
6. Google Trends
7. Reddit
8. External reports

**~4.4 million C++ devs
~1.9 million C devs**

There are 4.4 million C++ developers and nearly 2 million C developers in the world.



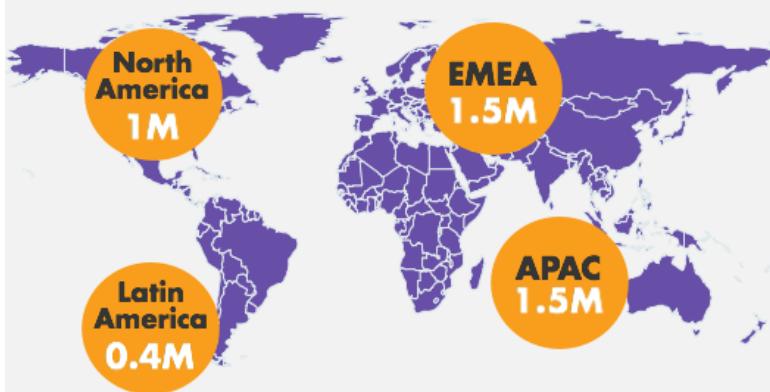
As many C++ developers as Python developers

We've analyzed a range of sources to estimate the number of worldwide developers using the most popular languages.

C++ is on par with Python, while the adoption of C is similar to that of Ruby.

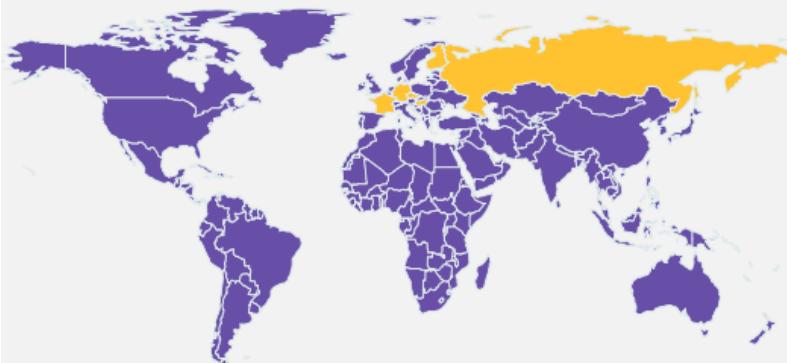
C++ developers by world region

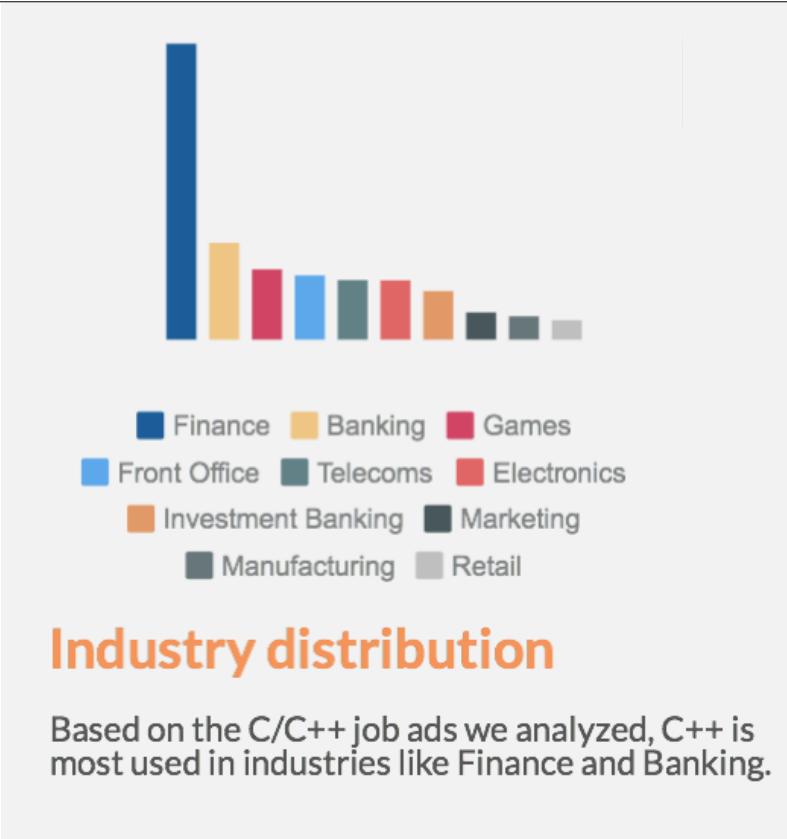
EMEA and Asia-Pacific are the most densely populated regions with regard to developers in general and C++ developers in particular.



Where C++ is relatively ahead of other languages

C++ is relatively more popular than other languages and technologies in Russia, Czech Republic, Hungary, France, Singapore, Finland, Israel and Germany.

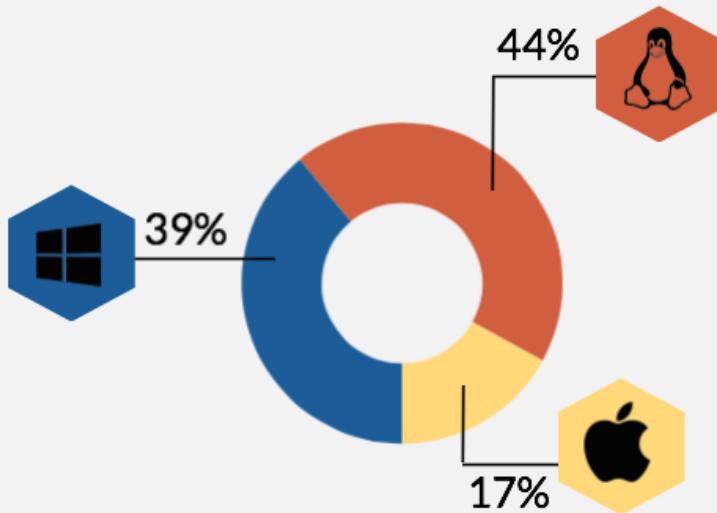




A collage of various programming language names, including SQL, Java, C, JavaScript, Perl, Python, Ruby, C#, PHP, VB, Objective-C, and Objective-C. The words are arranged in a scattered, overlapping manner.

Languages used with C++

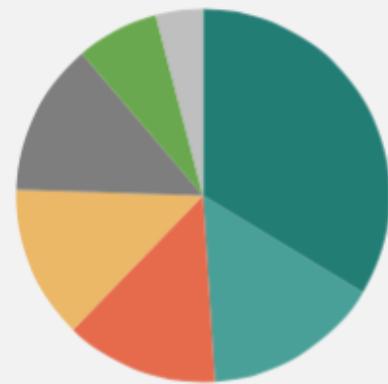
Judging from job ads, the languages that are most typically used together with C++ include Java, C, C#, Python, SQL and JavaScript.



C++ developers by platform

The most popular platform is Linux, used by 44% of C/C++ developers, followed by Windows (39%) and OS X (17%).

- C++11 (34%)
- C++03 (15%)
- C99 (13%)
- ANSI (13%)
- C++98 (13%)
- C11 (7%)
- Embedded (4%)



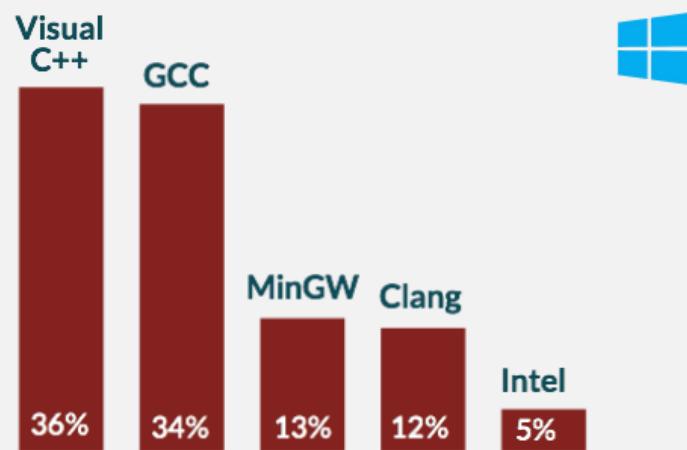
C++ versions

The most popular C++ version is currently C++11, with a share of 34%.



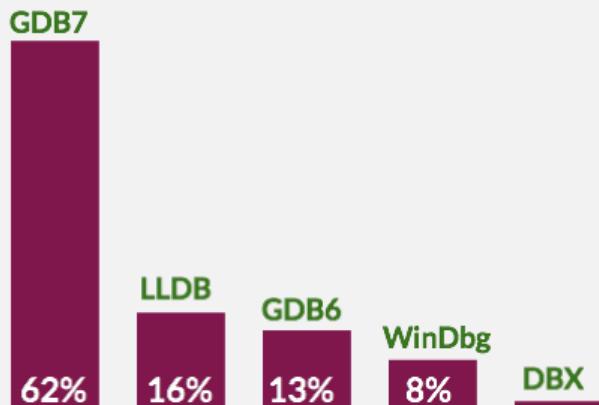
Popular C++ compilers

GCC is by far the most popular compiler with 65%, followed by Clang with 20%. Together they cover 85% of all C++ developers.



C++ compilers on Windows

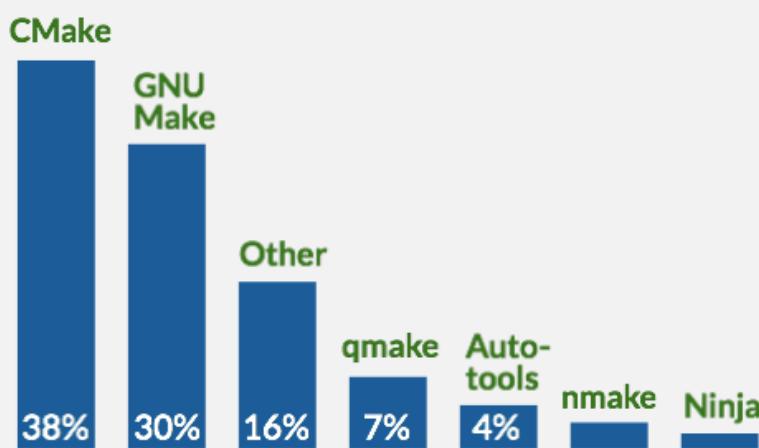
The title of the most popular C++ compiler on Windows is a virtual tie between Visual C++ (36%) and GCC (34%). MinGW (with no exact compiler name) and Clang are also tied with ~12% each. Intel comes in last at 5%.



Popular C++ debuggers

The runaway leader among debuggers used by C++ developers is GDB7 (62%), with LLDB and GDB6 trailing with 16% and 13%, respectively.

The distribution on OS X is noticeably different: LLDB is ahead at 39% with GDB7 a close second at 32%.



Popular C++ build systems

CMake and GNU Make build systems are the two close leaders, together accounting for 68% of all C++ developers.

Compiler + Build tool + Debugger

GCC + CMake + GDB7 toolchain takes the top spot for all C++ developers.

On OS X, however, that honor goes to Clang + CMake + LLDB.



Over five years.



CppCast

```
auto CppCast = pod_cast<C++>("http://cppcast.com");
```



Cpp.chat

No Diagnostic Required

The podcast and YouTube show
for C++ Annotated

JET
BRAINS

Algorithms +
Data
Structures =
Programs

<https://cpplang-inviter.cppalliance.org/>



Milestone	Date	Note
60K	Feb 19, 2018	/r/cpp hits 60K subscribers
TREND	Jan 10, 2018	/r/cpp is trending – Trend thread
50K	Jun 15, 2017	/r/cpp hits 50K subscribers
40K	Sep	
30K	Jul 2	
20K	Apr 17, 2014	/r/cpp hits 20K subscribers
TOP 1K	Nov 30, 2012	/r/cpp enters TOP 1K subreddits
10K	Oct 30, 2012	/r/cpp had 10K subscribers when v0.1 was released
Creation	May 26, 2008	/r/cpp is born

192K users



C++ × 536057

a general-purpose programming language. It was originally developed as an extension to C, and keeps a similar syntax. It is now a

700 K ???

177 channels

20K members

~ 1K weekly active users

[cppreference.com](#)

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

[C++ reference](#)

C++98, C++03, C++11, C++14, C++17, C++20, C++23 |



<https://cpplang-inviter.cppalliance.org/>



+



Join **CppLang** on Slack.

18108 users are registered so far.

you@yourdomain.com



I'm not a robot



GET MY INVITE

or [sign in](#).

<https://cpplang.now.sh/>



Join **Cpplang** on Slack.

136 users online now of **7092** registered.

you@yourdomain.com

I'm not a robot



GET MY INVITE

or [sign in](#).

Cpplang Jon Kalb Unread Mentions

deleteme
documentation
eastconst
embedded
embo
events
future_standard
general
include
ip_law
jobs
local_groups
meetingcpp
meta_programming
meta_slack
pacifcpp
plug_worthy
poetry
purecpp
sg15_tooling
sg20_education
sg6_numerics
sharing
show-runners
speakerscorner
students
teaching
test
ug_ca_toronto
ug_us_atlanta_private
ug-ca-montreal
ug-de-berlin
ug-de-pottcpp
ug-es-madrid
ug-fr-paris-confrue

More Unreads

ug_ca_toronto

You created this channel yesterday. This is the very beginning of the # ug_ca_toronto channel. Purpose: Discussion of and about the Toronto local user group. ([edit](#))

+ Add an app [Invite others to this channel](#)

Yesterday

Jon Kalb 13:44 joined #ug_ca_toronto.

Jon Kalb 13:44 set the channel purpose: Discussion of and about the Toronto local user group.

Michael Daum 13:44 joined #ug_ca_toronto by invitation from Jon Kalb, along with 2 others.

Jon Kalb 13:45 @Cajoon, @Jason Walter I'm looking forward to meeting you.

gregcons I'm looking forward to seeing you again!

Kate Gregory 13:46 Me too!

Jason Walter 13:47 Likewise! Excited for the event. Please send me the address you are staying and I can pick you up

Jon Kalb 16:26 @Jason Walter If you could stop on Yorkville Ave right at the corner with Avenue Road at 17:45, I'll be waiting. Does that work?

Jason Walter 20:50 Yes. See you then!

Today

Michael Daum 13:03 Oh hi!

Thanks for making this channel, @jonkalb. Sorry it took me so long to get here.

Message #ug_ca_toronto @



Get Started! Tour Core Guidelines Super-FAQ Standardization About

Wiki Home > User Groups Worldwide

User Groups Worldwide

Contents of this section:

- World Map
- How can I start a local user group?
- How can I present at a local user group?
- Argentina
- Australia
- Austria
- Belarus
- Belgium
- Brazil
- Bulgaria
- Canada
- Czech Republic
- China
- Colombia
- Denmark
- France
- Germany
- Hungary
- India
- Ireland
- Israel
- Italy
- Luxembourg
- Latin America
- Macedonia
- Netherlands
- New Zealand
- Norway
- Poland

github

Working Draft of the next standard

Current ISO C++ status

Upcoming ISO C++ meetings

Compiler conformance status

NAVIGATION

FAQ Home

FAQ RSS Feed

FAQ Help

SEARCH THIS WIKI

GO TO PAGE

UPCOMING EVENTS

ACCU 2018
April 11-14, Bristol, UK

C++ Russia 2018
April 19-20, Saint-Petersburg, Russia

C++ Now 2018
May 6-11, Aspen, CO, USA

ADC++ 2018
May 14-16, Burghausen, Germany

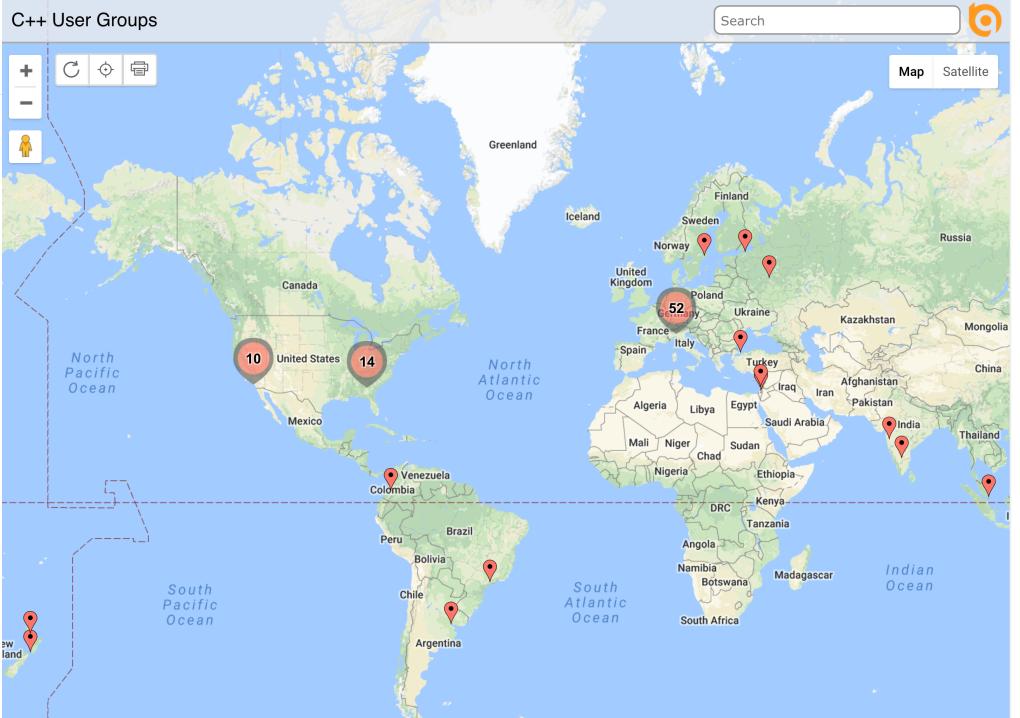
Summer ISO C++ Meeting
Jun 4-9, Rapperswil, Switzerland

Italian C++ Conference 2018

C++ User Groups

Search

Map Satellite



The map displays the global distribution of C++ User Groups. Major concentrations are shown in North America (USA and Canada), Europe (Germany has 52 groups), and Asia (India has 14 groups). Other notable locations include Australia, Brazil, and several countries in Africa and the Middle East.



C++ Community Events

General Public search

Number of months: 12

Month	Day	Event
April	1	C++ Conf
April	2	C++ Conf
April	3	C++ Conf
April	4	C++ Conf
April	5	C++ Conf
April	6	C++ Conf
April	7	C++ Conf
April	8	C++ Conf
April	9	C++ Conf
April	10	C++ Conf
April	11	ACCU (julie@arche)
April	12	ACCU (julie@arche)
April	13	ACCU (julie@arche)
April	14	ACCU (julie@arche)
April	15	ACCU (julie@arche)
April	16	ACCU (julie@arche)
April	17	ACCU (julie@arche)
April	18	ACCU (julie@arche)
April	19	ACCU (julie@arche)
April	20	ACCU (julie@arche)
April	21	ACCU (julie@arche)
April	22	ACCU (julie@arche)
April	23	ACCU (julie@arche)
April	24	ACCU (julie@arche)
April	25	ACCU (julie@arche)
April	26	ACCU (julie@arche)
April	27	ACCU (julie@arche)
April	28	ACCU (julie@arche)
April	29	ACCU (julie@arche)
April	30	ACCU (julie@arche)
May	1	C++Now (info@cppnow.org)
May	2	C++Now (info@cppnow.org)
May	3	C++Now (info@cppnow.org)
May	4	C++Now (info@cppnow.org)
May	5	C++Now (info@cppnow.org)
May	6	C++Now (info@cppnow.org)
May	7	C++Now (info@cppnow.org)
May	8	C++Now (info@cppnow.org)
May	9	C++Now (info@cppnow.org)
May	10	C++Now (info@cppnow.org)
May	11	C++Now (info@cppnow.org)
May	12	C++Now (info@cppnow.org)
May	13	C++Now (info@cppnow.org)
May	14	C++Now (info@cppnow.org)
May	15	C++Now (info@cppnow.org)
May	16	C++Now (info@cppnow.org)
May	17	C++Now (info@cppnow.org)
May	18	C++Now (info@cppnow.org)
May	19	C++Now (info@cppnow.org)
May	20	C++Now (info@cppnow.org)
May	21	C++Now (info@cppnow.org)
May	22	C++Now (info@cppnow.org)
May	23	C++Now (info@cppnow.org)
May	24	C++Now (info@cppnow.org)
May	25	C++Now (info@cppnow.org)
May	26	C++Now (info@cppnow.org)
May	27	C++Now (info@cppnow.org)
May	28	C++Now (info@cppnow.org)
May	29	C++Now (info@cppnow.org)
May	30	C++Now (info@cppnow.org)
June	1	Rapperswil Meeting (Peter Sp)
June	2	Rapperswil Meeting (Peter Sp)
June	3	Rapperswil Meeting (Peter Sp)
June	4	Rapperswil Meeting (Peter Sp)
June	5	Rapperswil Meeting (Peter Sp)
June	6	Rapperswil Meeting (Peter Sp)
June	7	Rapperswil Meeting (Peter Sp)
June	8	Rapperswil Meeting (Peter Sp)
June	9	Rapperswil Meeting (Peter Sp)
June	10	Rapperswil Meeting (Peter Sp)
June	11	Rapperswil Meeting (Peter Sp)
June	12	Rapperswil Meeting (Peter Sp)
June	13	Rapperswil Meeting (Peter Sp)
June	14	Rapperswil Meeting (Peter Sp)
June	15	Rapperswil Meeting (Peter Sp)
June	16	Rapperswil Meeting (Peter Sp)
June	17	Rapperswil Meeting (Peter Sp)
June	18	Rapperswil Meeting (Peter Sp)
June	19	Rapperswil Meeting (Peter Sp)
June	20	Rapperswil Meeting (Peter Sp)
June	21	Rapperswil Meeting (Peter Sp)
June	22	Rapperswil Meeting (Peter Sp)
June	23	Rapperswil Meeting (Peter Sp)
June	24	Rapperswil Meeting (Peter Sp)
June	25	Rapperswil Meeting (Peter Sp)
June	26	Rapperswil Meeting (Peter Sp)
June	27	Rapperswil Meeting (Peter Sp)
June	28	Rapperswil Meeting (Peter Sp)
June	29	Rapperswil Meeting (Peter Sp)
June	30	Rapperswil Meeting (Peter Sp)
July	1	talk
July	2	talk
July	3	talk
July	4	talk
July	5	talk
July	6	talk
July	7	talk
July	8	talk
July	9	talk
July	10	talk
July	11	talk
July	12	talk
July	13	talk
July	14	talk
July	15	talk
July	16	talk
July	17	talk
July	18	talk
July	19	talk
July	20	talk
July	21	talk
July	22	talk
July	23	talk
July	24	talk
July	25	talk
July	26	talk
July	27	talk
July	28	talk
July	29	talk
July	30	talk
July	31	talk
August	1	talk
August	2	talk
August	3	talk
August	4	talk
August	5	talk
August	6	talk
August	7	talk
August	8	talk
August	9	talk
August	10	talk
August	11	talk
August	12	talk
August	13	talk
August	14	talk
August	15	talk
August	16	talk
August	17	talk
August	18	talk
August	19	talk
August	20	talk
August	21	talk
August	22	talk
August	23	talk
August	24	talk
August	25	talk
August	26	talk
August	27	talk
August	28	talk
August	29	talk
August	30	talk
September	1	Pacific++
September	2	Pacific++
September	3	Pacific++
September	4	Pacific++
September	5	Pacific++
September	6	Pacific++
September	7	Pacific++
September	8	Pacific++
September	9	Pacific++
September	10	Pacific++
September	11	Pacific++
September	12	Pacific++
September	13	Pacific++
September	14	Pacific++
September	15	Pacific++
September	16	Pacific++
September	17	Pacific++
September	18	Pacific++
September	19	Pacific++
September	20	Pacific++
September	21	Pacific++
September	22	Pacific++
September	23	Pacific++
September	24	Pacific++
September	25	Pacific++
September	26	Pacific++
September	27	Pacific++
September	28	Pacific++
September	29	Pacific++
September	30	Pacific++
October	1	CppCon (info@cppcon.org)
October	2	CppCon (info@cppcon.org)
October	3	CppCon (info@cppcon.org)
October	4	CppCon (info@cppcon.org)
October	5	CppCon (info@cppcon.org)
October	6	CppCon (info@cppcon.org)
October	7	CppCon (info@cppcon.org)
October	8	CppCon (info@cppcon.org)
October	9	CppCon (info@cppcon.org)
October	10	CppCon (info@cppcon.org)
October	11	CppCon (info@cppcon.org)
October	12	CppCon (info@cppcon.org)
October	13	CppCon (info@cppcon.org)
October	14	CppCon (info@cppcon.org)
October	15	CppCon (info@cppcon.org)
October	16	CppCon (info@cppcon.org)
October	17	CppCon (info@cppcon.org)
October	18	CppCon (info@cppcon.org)
October	19	CppCon (info@cppcon.org)
October	20	CppCon (info@cppcon.org)
October	21	CppCon (info@cppcon.org)
October	22	CppCon (info@cppcon.org)
October	23	CppCon (info@cppcon.org)
October	24	CppCon (info@cppcon.org)
October	25	CppCon (info@cppcon.org)
October	26	CppCon (info@cppcon.org)
October	27	CppCon (info@cppcon.org)
October	28	CppCon (info@cppcon.org)
October	29	CppCon (info@cppcon.org)
October	30	CppCon (info@cppcon.org)
November	1	Meeting C++/Modern C++ and
November	2	Meeting C++/Modern C++ and
November	3	Meeting C++/Modern C++ and
November	4	Meeting C++/Modern C++ and
November	5	Meeting C++/Modern C++ and
November	6	Meeting C++/Modern C++ and
November	7	Meeting C++/Modern C++ and
November	8	Meeting C++/Modern C++ and
November	9	Meeting C++/Modern C++ and
November	10	Meeting C++/Modern C++ and
November	11	Meeting C++/Modern C++ and
November	12	Meeting C++/Modern C++ and
November	13	Meeting C++/Modern C++ and
November	14	Meeting C++/Modern C++ and
November	15	Meeting C++/Modern C++ and
November	16	Meeting C++/Modern C++ and
November	17	Meeting C++/Modern C++ and
November	18	Meeting C++/Modern C++ and
November	19	Meeting C++/Modern C++ and
November	20	Meeting C++/Modern C++ and
November	21	Meeting C++/Modern C++ and
November	22	Meeting C++/Modern C++ and
November	23	Meeting C++/Modern C++ and
November	24	Meeting C++/Modern C++ and
November	25	Meeting C++/Modern C++ and
November	26	Meeting C++/Modern C++ and
November	27	Meeting C++/Modern C++ and
November	28	Meeting C++/Modern C++ and
November	29	Meeting C++/Modern C++ and
November	30	Meeting C++/Modern C++ and

General Public search

Number of months: 12

Week 8 Weeks Month Year List

Calendars

C++ Conferences

C++ Training

C++ User Groups

Import

ISO Committee Meetings

Related Conferences

Related Training

Related User Groups

Filter

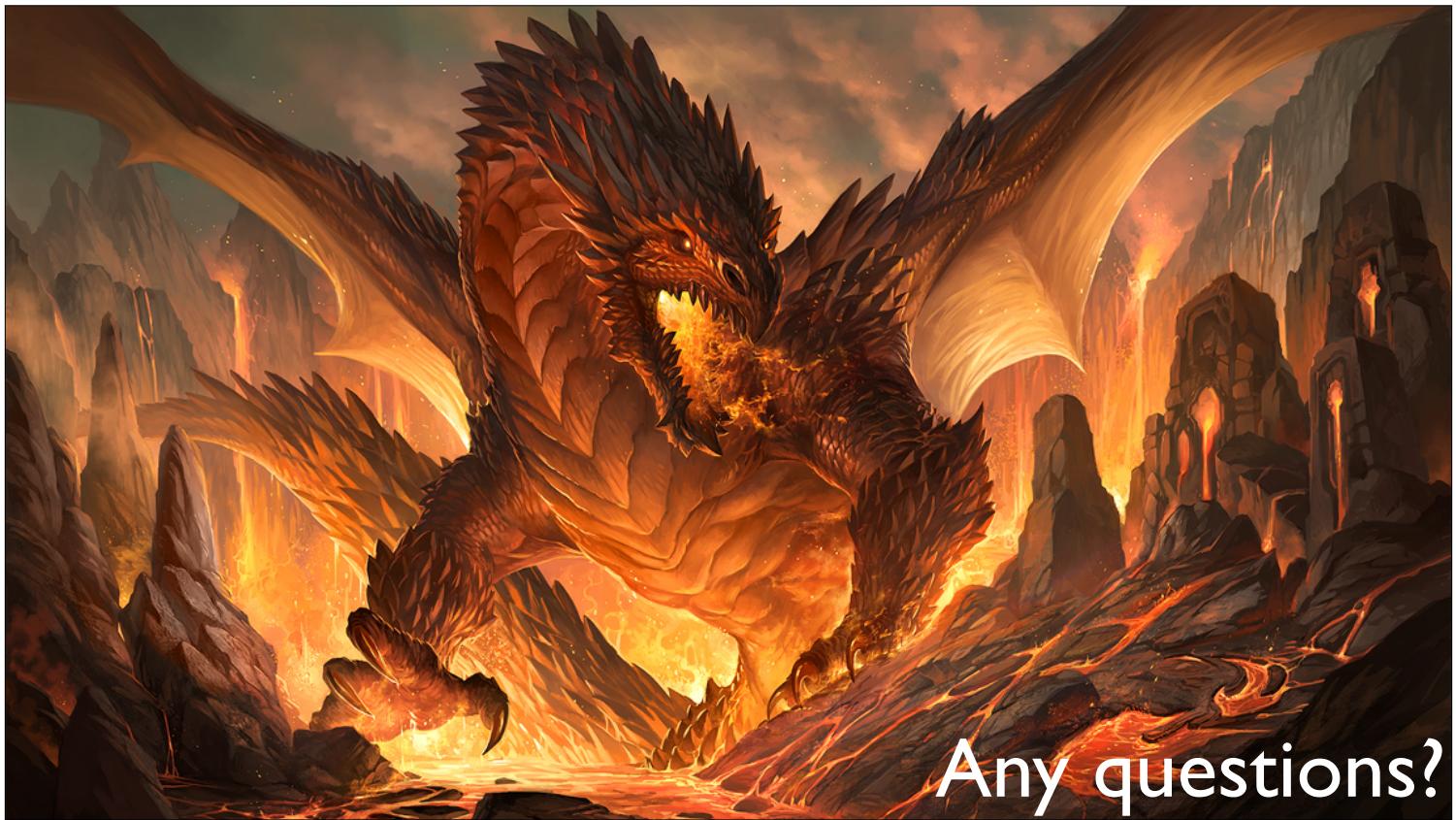
About

This calendar is for sharing C++ community events. Appropriate events are open to the public and either explicitly C++ or mostly C++.

Event organizers can contact calendar@cachechach.com

The screenshot shows the homepage of the C++ FAQ. On the left sidebar, there are decorative icons: a green and yellow flower-like logo, a large blue hexagonal logo with a white 'C' and '++', and a yellow 'C' on a black background. The main content area has a dark header with navigation links: Get Started!, Tour, Core Guidelines, Super-FAQ, Standardization, and About. Below this is a breadcrumb trail: Wiki Home > Conferences Worldwide. The main title is "Conferences Worldwide". A section titled "Contents of this section:" lists three categories: Asia / Pacific, Europe, and North America. Under "FAQ Asia / Pacific", there are links to "C++ and System Software Summit - China" and "CppSiberia - Russia". Under "FAQ Europe", there is a link to "ACCU - UK". On the right side, there is a graphic for "C++ NORTH" featuring a red maple leaf and a stylized 'C' made of orange and yellow squares above wavy blue lines. There is also a circular logo for "C++ COREHARD" with "COREHARD" in the center and "C++ CONFERENCE" around the border.





Any questions?