C++ Today

The Beast is Back

C++ Today

The Beast is Back

Jon Kalb

O'REILLY®

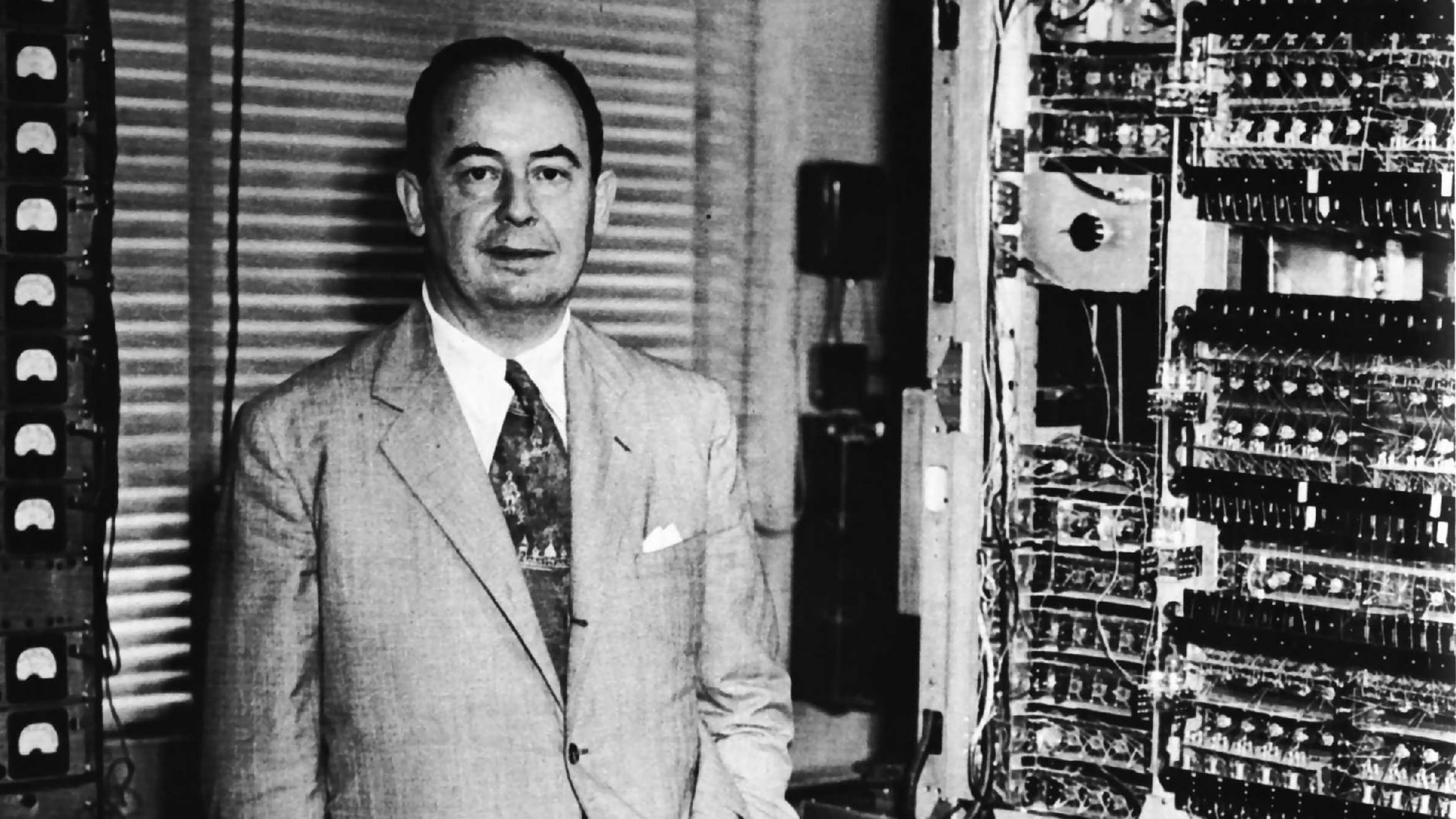
C++ Today

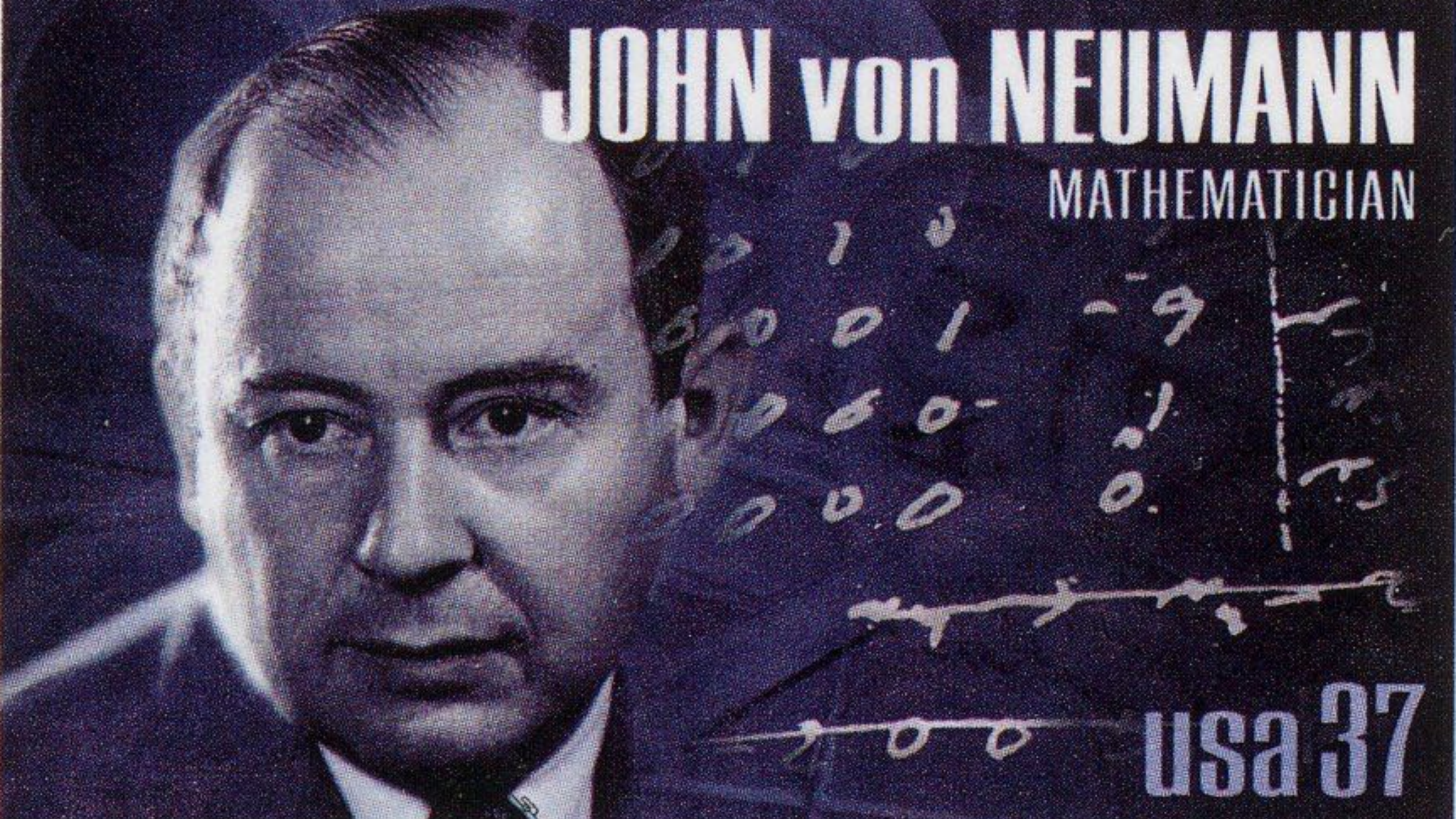
The Beast is Back

Jon Kalb & Gašper Ažman

Free Download
(with email address)

O'Reilly Media:
j.mp/Cpp_Today



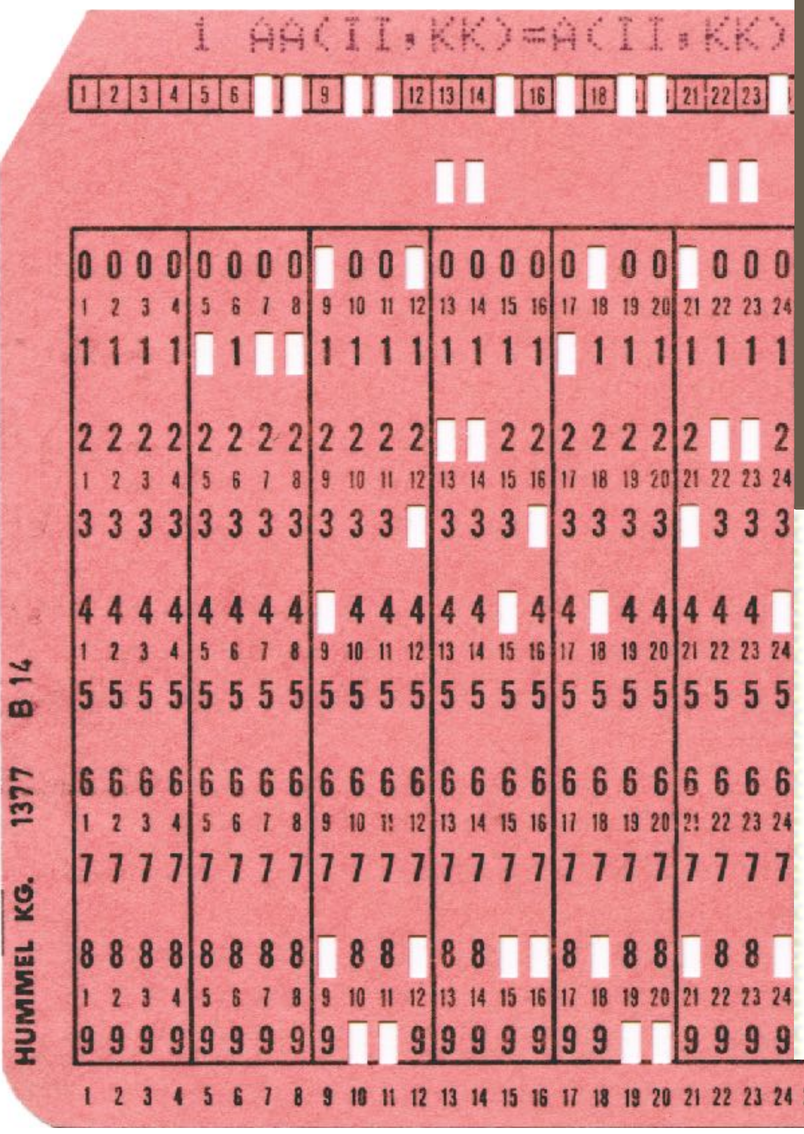

JOHN von NEUMANN
MATHEMATICIAN
usa 37

```
38:A1 DE     LDA ($DE,X)  } (ANY2+A)
3A:91 DE     STA ($DE),Y  } =(ANY2)
3C:A9 DE     LDA #$DE     } ANY2=ANY2
3E:20 10 0E  JSR DECZPG   } -1
41:A5 DE     LDA $DE
43:C5 DB     CMP $DB
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:[illegible]88   [illegible] DEY
[illegible]

52:A9 20     LDA #$20
54:91 DB     STA ($DB),Y
56:[illegible]C0 00
```

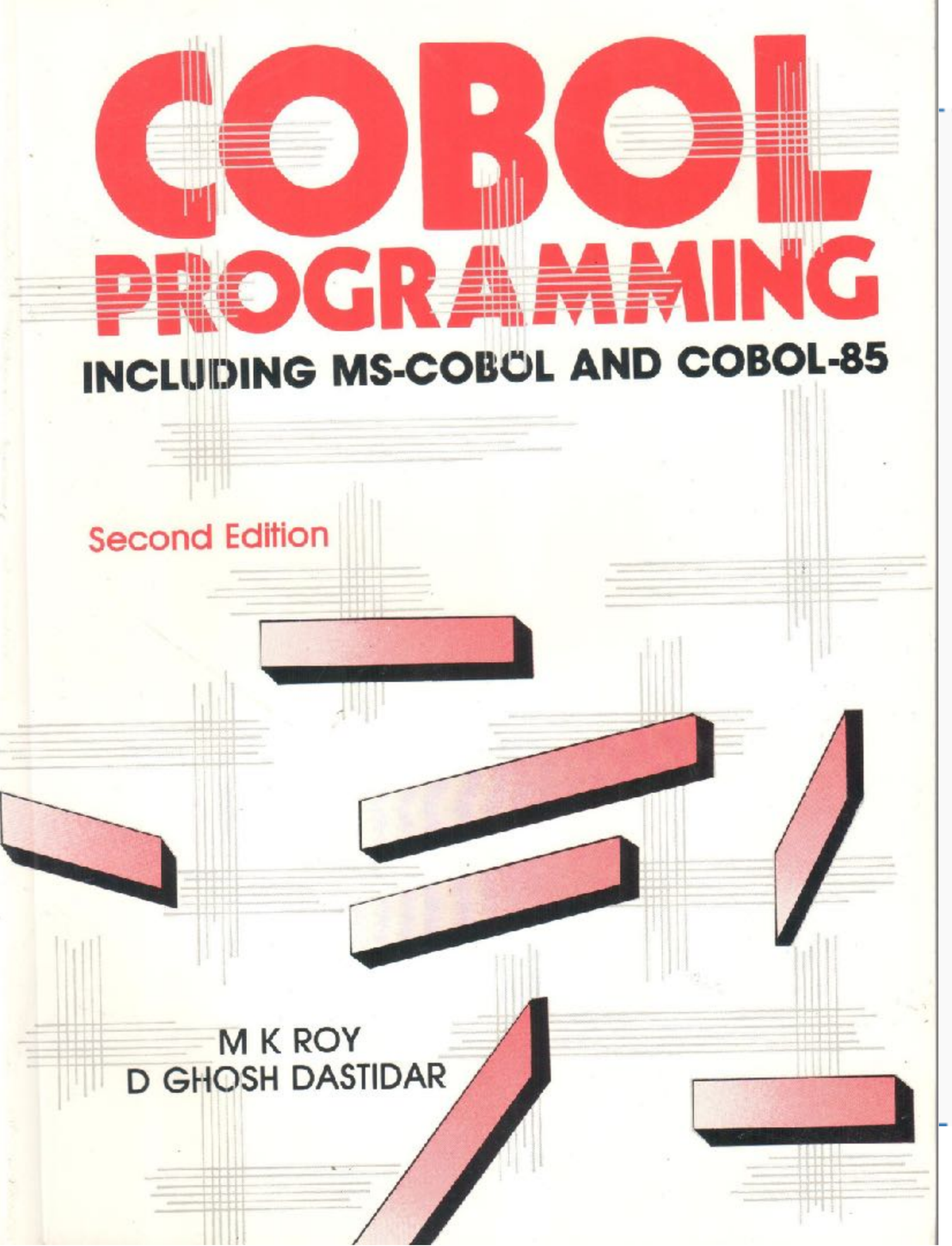
FOR-NEXT MEMORY MOVING ROUTINE.

Stores spaces ($20) in ne



Fortran

```
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/
```




```
//COBUCLG  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//HELOWRLD EXEC COBUCLG,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 DSNAME=SYS1.COBLIB,DISP=SHR
//            DD DSNAME=SYS1.LINKLIB,DISP=SHR
//GO.SYSPRINT DD SYSOUT=A
//
```


PROBE INTERFACE
PRINTER / 360 / CALCOMP
USE IN LEFT SIDE
DATA DISP
MOTOROLA PRINTER
BRIGHTEN
MUSIC
Scotch


1960s


IBM
System 360

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.

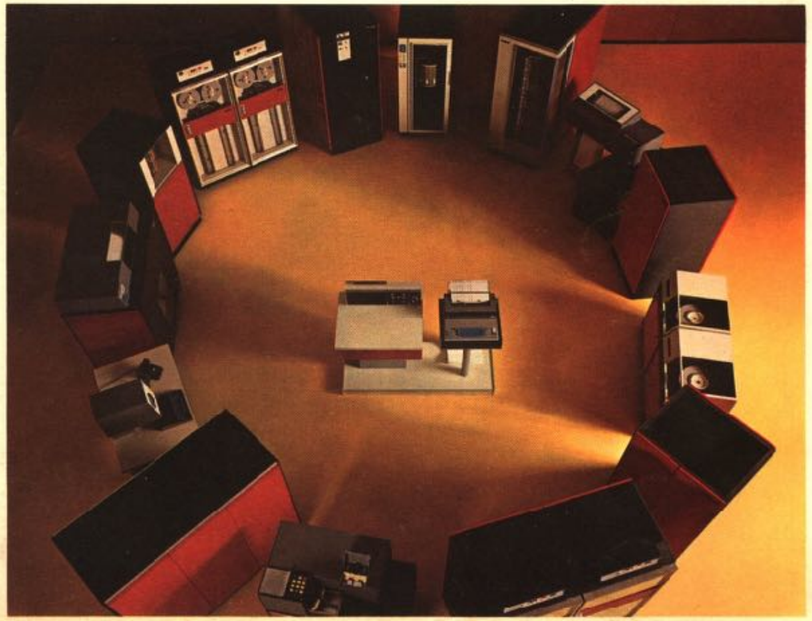
Same goes for input and output devices. Any printer, tape, storage unit, reader or terminal that works in a small configuration works in a larger one. You choose what you need now. You add new components when you need them.

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

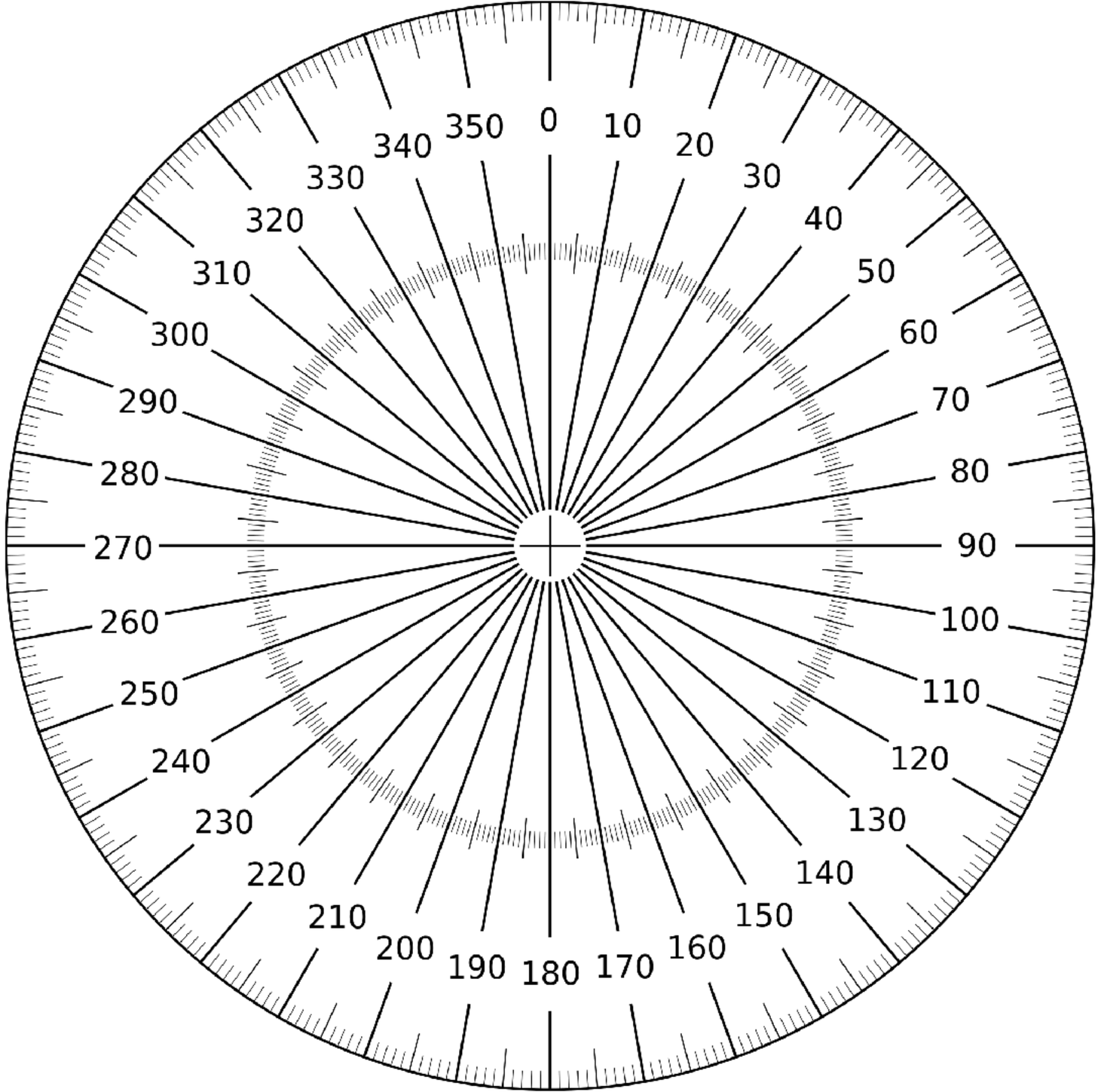
SYSTEM/360 solves today's problems. And it expands to solve tomorrow's problems, too.

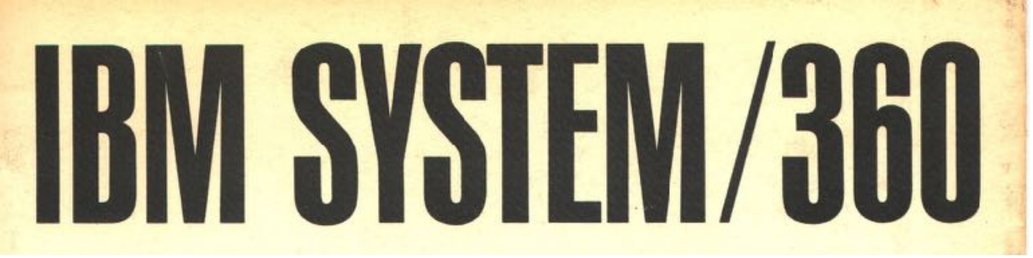
It cuts today's costs....and it will also cut tomorrow's. There's never been a system quite like it.

IBM® DATA PROCESSING

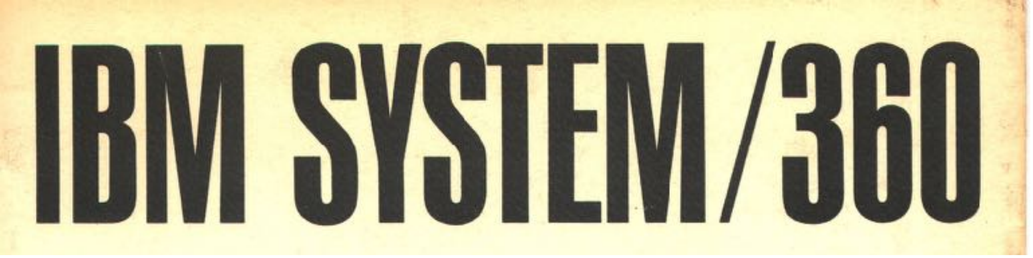


Litho in U.S.A. 528-0958





- 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

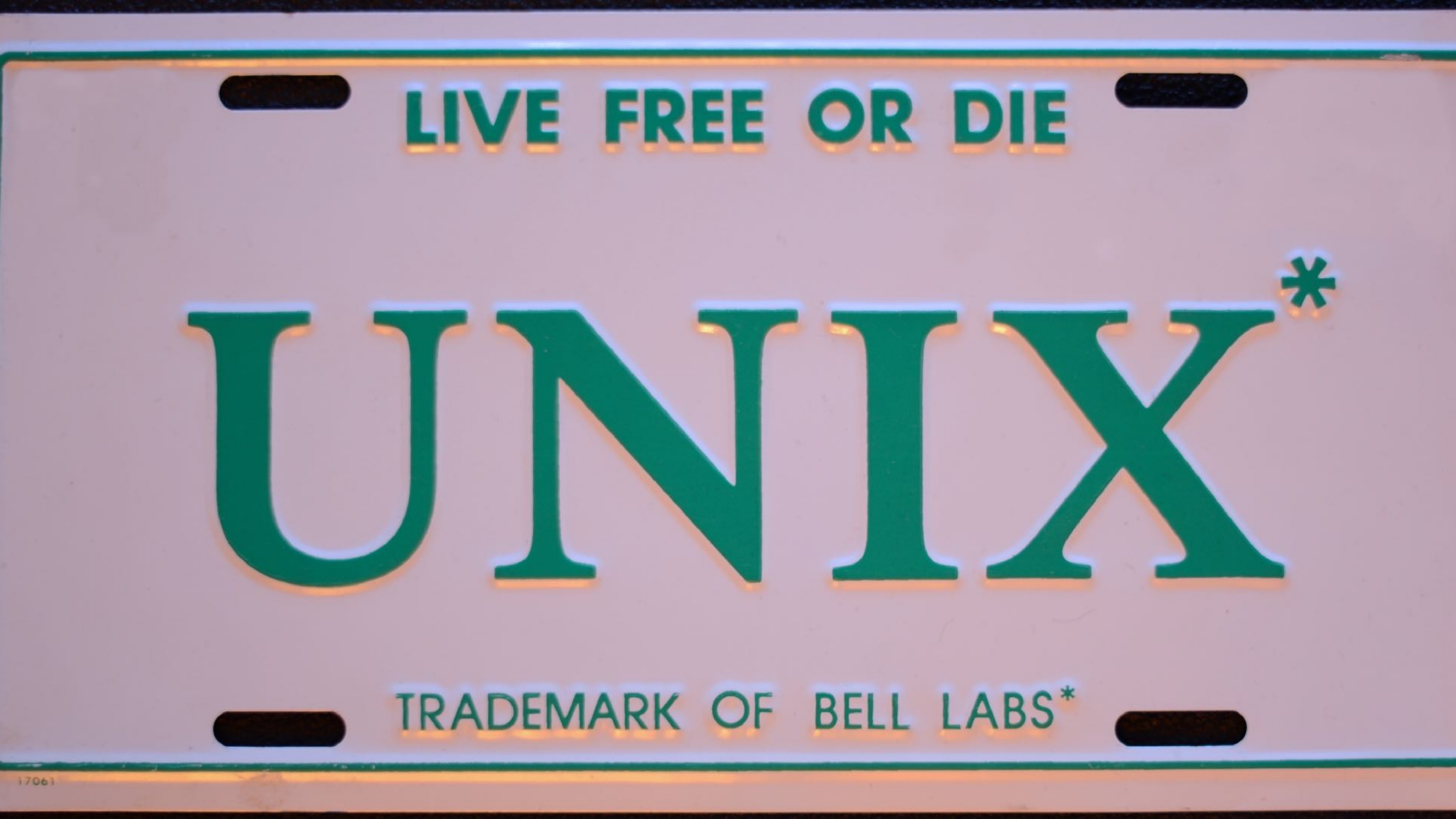


- Designed to be a machine for science and business
- Featured binary, decimal (BCD), and hexadecimal floating-point calculations
- First instruction set implemented in microcode
- 8-bit byte addressing

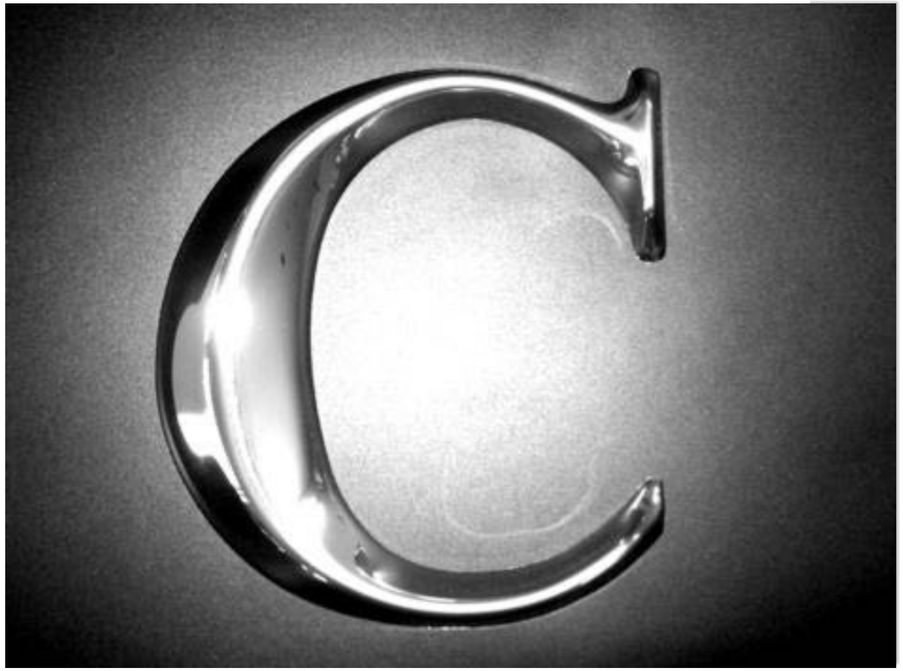
Established the “C machine” architecture a decade before “C.”


1970s


DIGITAL EQUIPMENT CORPORATION
PDP-7
JANUS 71
SYSTEM SOFTWARE ONLY


LIVE FREE OR DIE
UNIX*
TRADEMARK OF BELL LABS*
17061

THE C PROGRAMMING LANGUAGE



C

PROGRAMMING

LANGUAGE

CREATOR

DENNIS RITCHIE

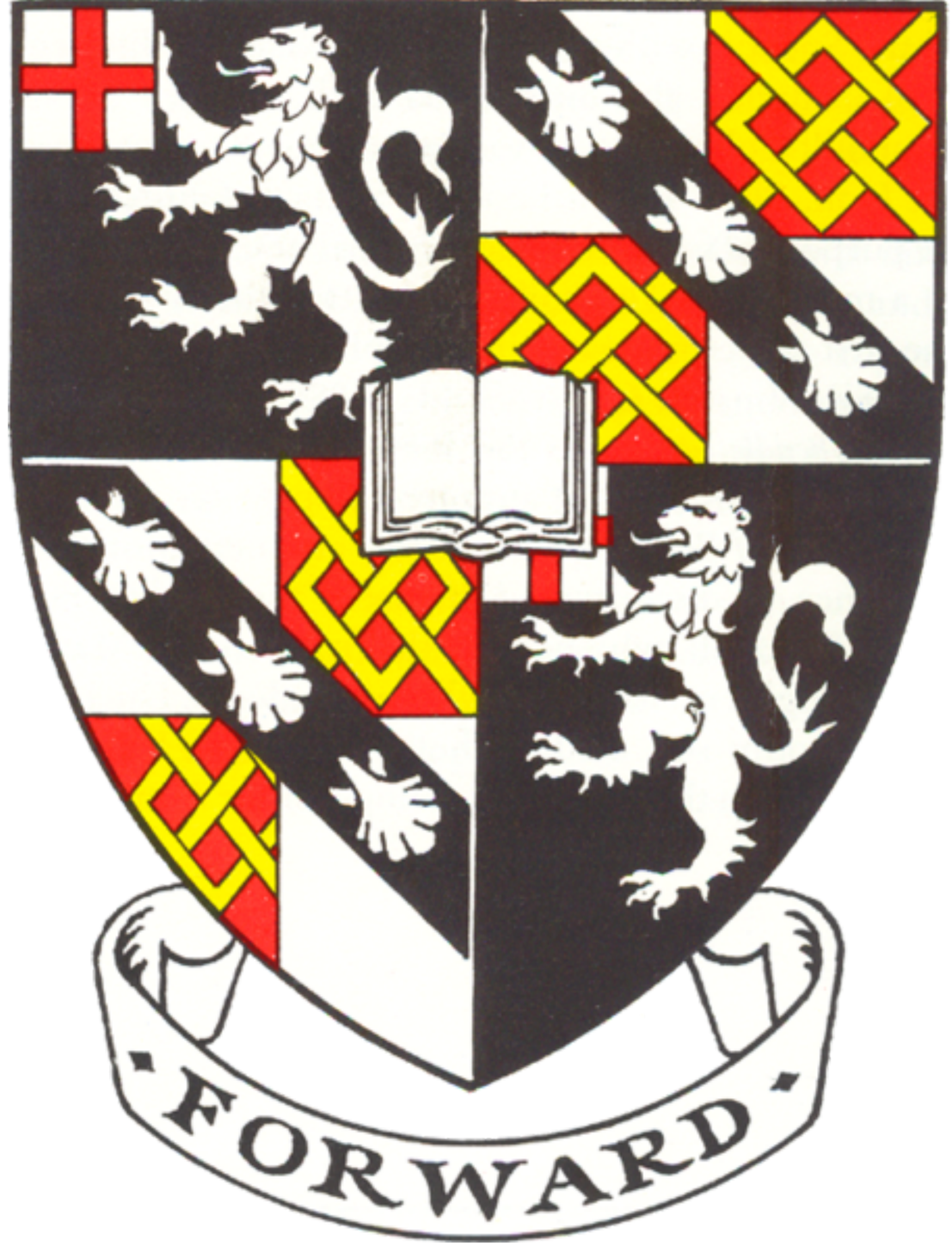
1941 - 2011


THE 1980s

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


FORWARD

“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 any thing that can be done with a fundamental type, can be done with a user-defined type
  - for example: a programmer can defined a type that functions as a
    - fundamental arithmetic type (operator overload support mathematical operations)
    - object pointer (smart pointer)
    - function pointer (functor)

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

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

boost
C++ LIBRARIES

2000 2001 2002 2003 2004 2005

2000 - 2010

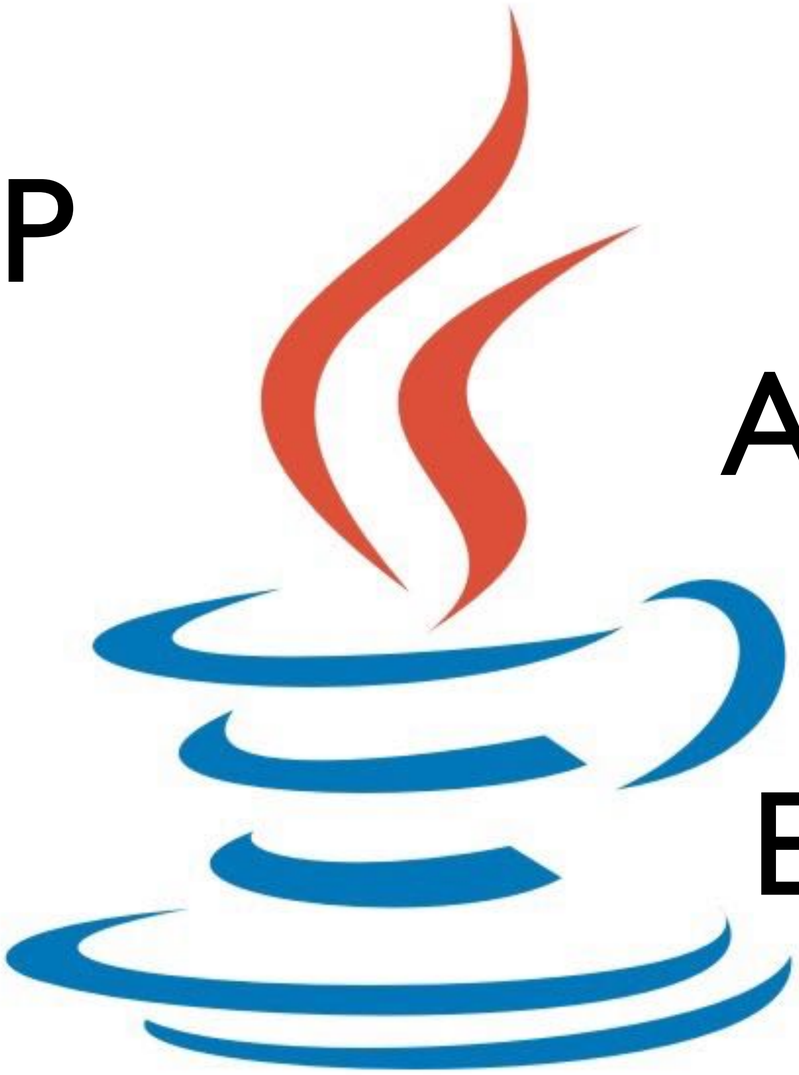
A Decade To Forget

2006 2007 2008 2009 2010

Supports OOP

Language of the Web

Language of the Future



Almost as fast

Easier to learn and teach

Java™

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

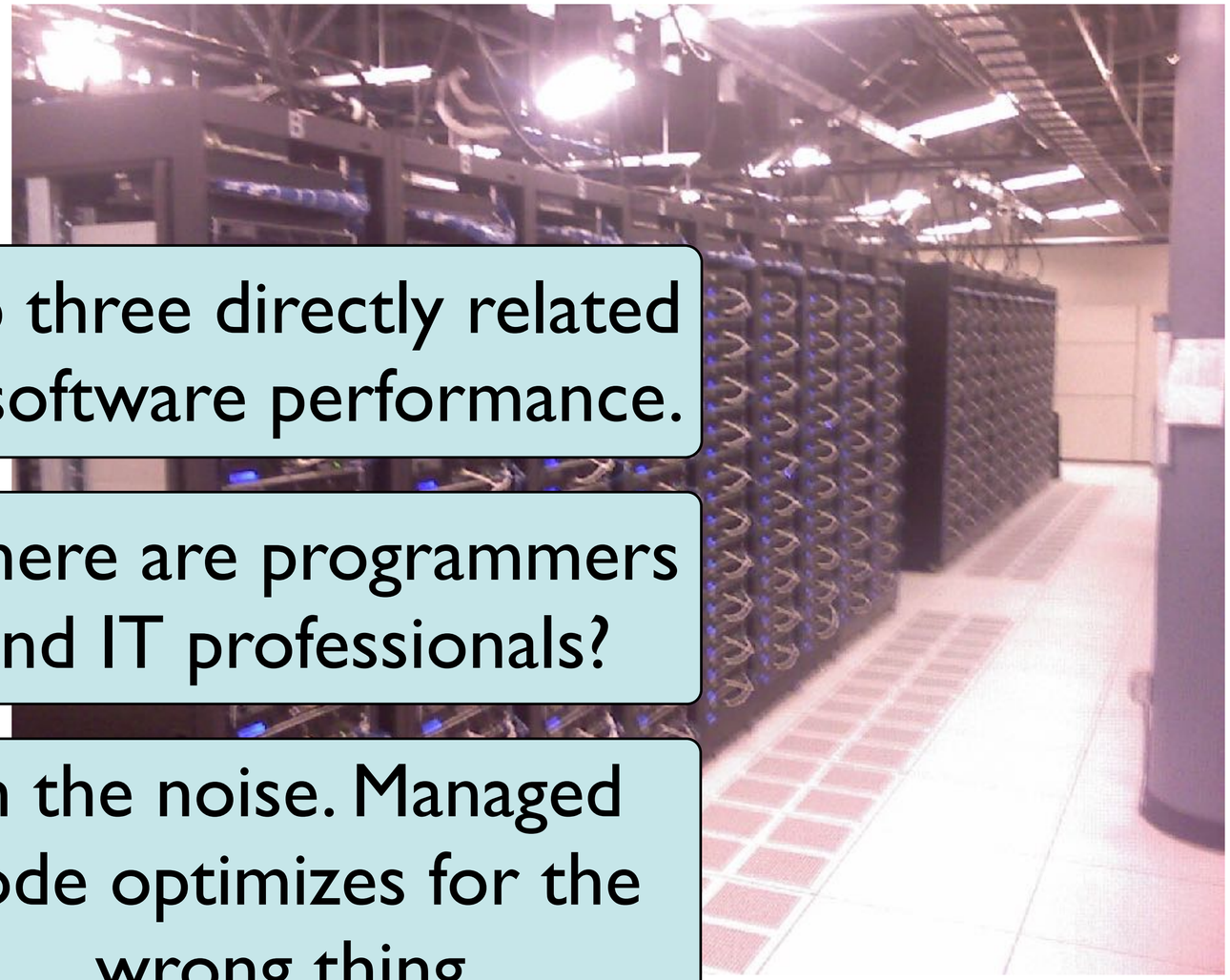
International Organization for Standardization

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!

C++11

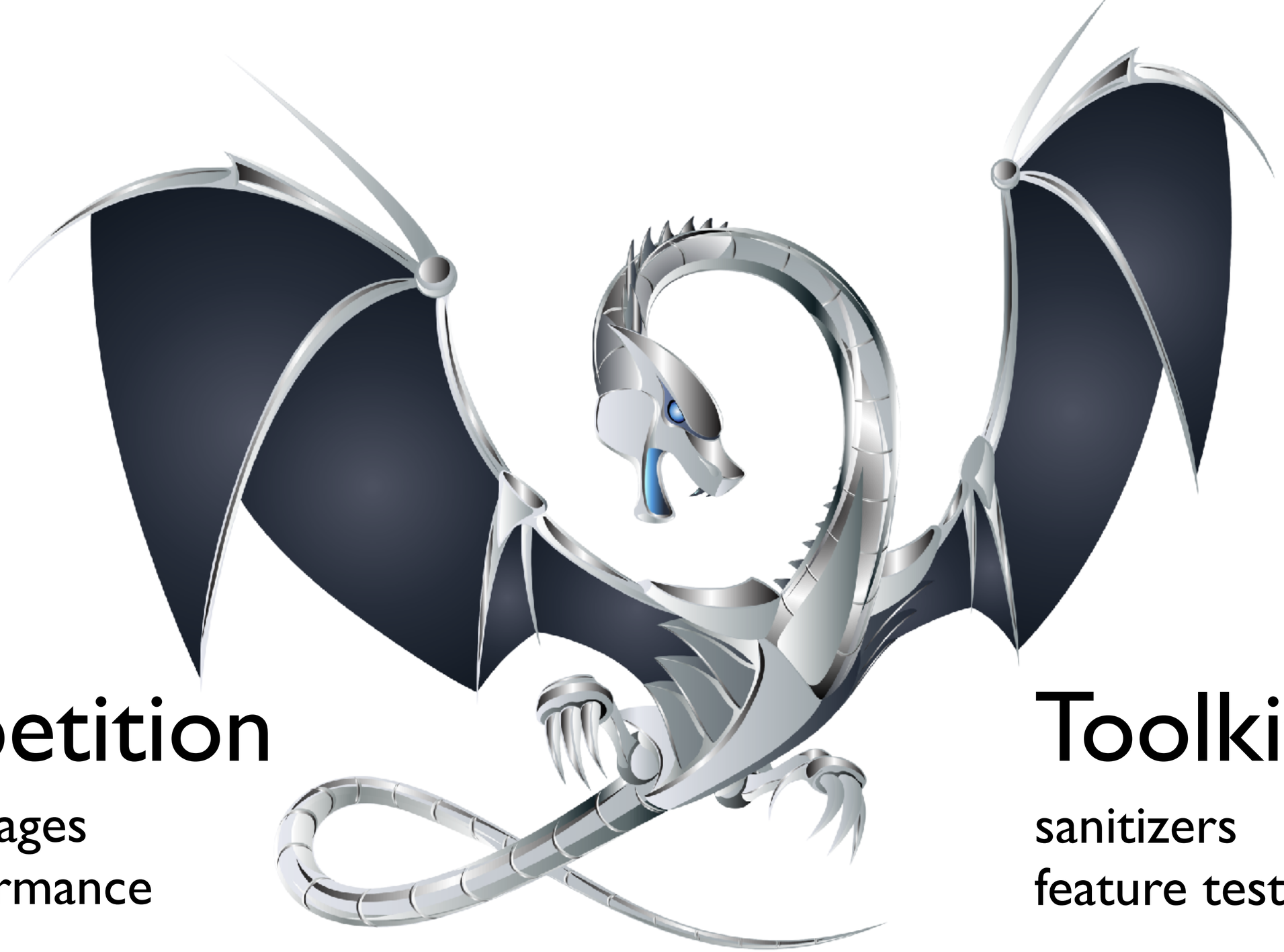
- page count
  - C++03: 776
  - C++11: 1353
- “simplifying” by adding
  - auto
  - range-base for loops
  - >>
  - enums more consistent
  - uniform initialization

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



Competition

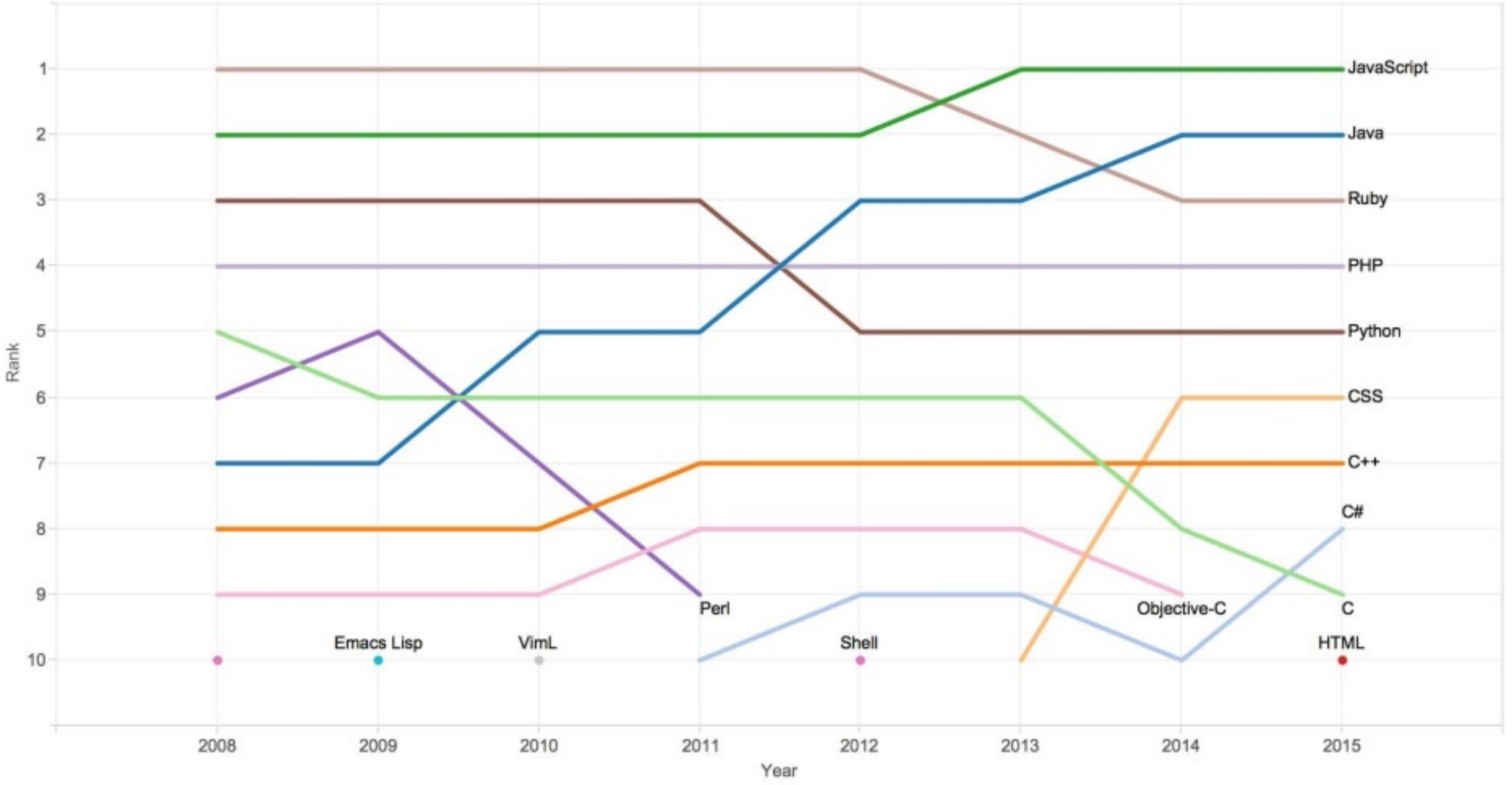
error messages
build performance

Toolkit

sanitizers
feature testing


I WANT YOU
TO OPEN SOURCE!
WWW.MIL-OSS.ORG

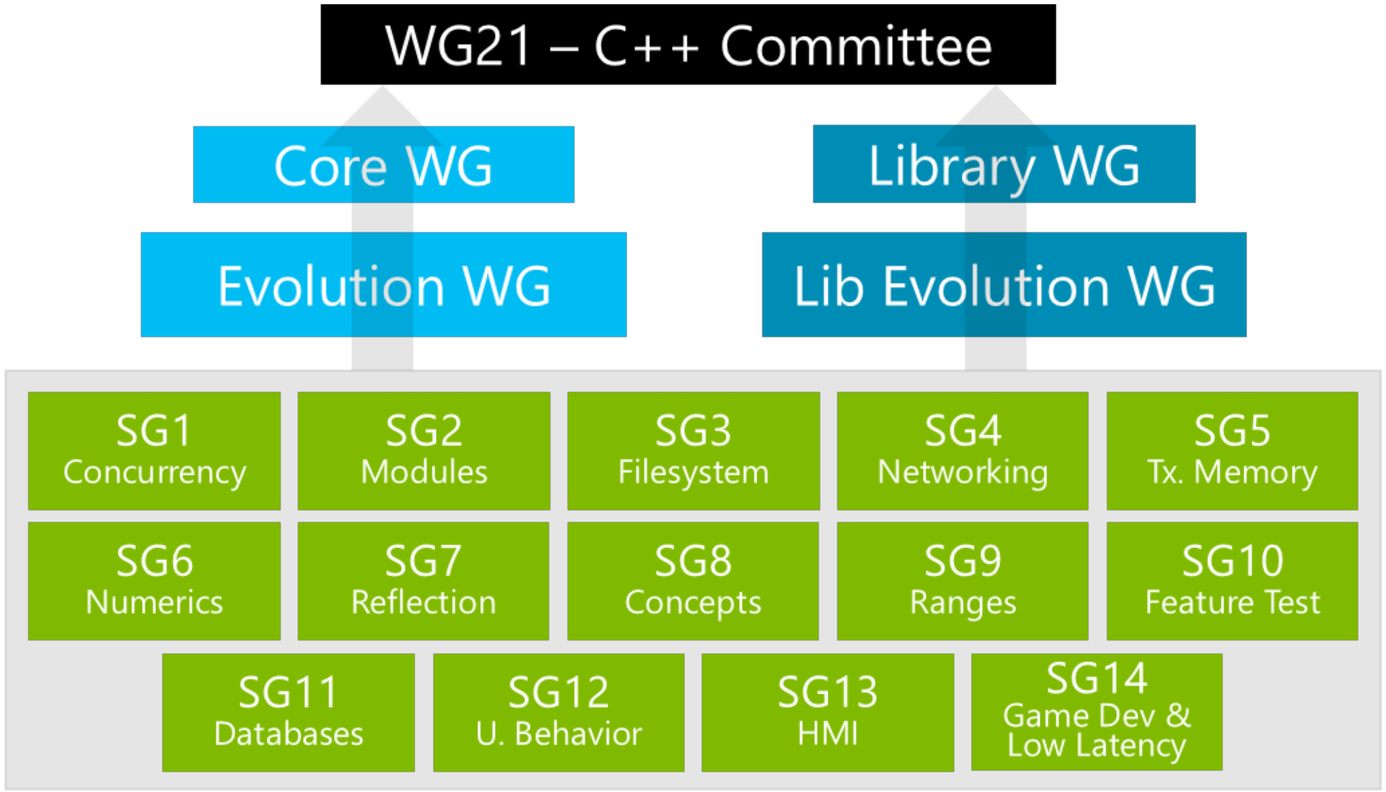
Rank of top languages on GitHub.com over time



Source: GitHub.com

International Organization for Standardization

International Organization for Standardization




Sign In /
Get Started!
Tour
Core Guidelines
Super-FAQ
Standardization
About
News, Status & Discussion about Standard C++
FEATURES
Working Draft of the next standard
github
Follow All Posts
The home of Standard C++ on the web — news, status and discussion about the C++ standard on all compilers and platforms.
Recent Highlights
Selected Recent C++ Questions
Get Started!
Tour
Core Guidelines
Super-FAQ
Standardization
About
Wiki Home > C++ FAQ
View
C++ FAQ
Save to: Instapaper Pocket Readability
Welcome to the C++ Super-FAQ!
What's "Super" about this FAQ? In part it's because this is a merger of two great FAQs: Marshall Cline's C++ FAQs, and Bjarne Stroustrup's C++ FAQ. And in part it's because this is a wiki being continuously updated for modern C++. There are some FAQ topics not yet updated; if you spot one, suggest an improvement using the link on the bar for that FAQ.
We would like to gratefully acknowledge Pearson Education (Addison-Wesley), Marshall Cline, Bjarne Stroustrup, Herb Sutter, and Andrei Alexandrescu for launching this FAQ by freely contributing the contents of Marshall Cline's online C++ FAQ Lite, Bjarne Stroustrup's online C++ and C++11 FAQs, and in the near future material from their books C++ FAQs Second Edition by Marshall Cline, Greg Lomow, and Mike Girou, and C++ Coding Standards by Herb Sutter and Andrei Alexandrescu.

CLion

Cross-platform C/C++ IDE
by JetBrains

Just Released

facts

JetBRAINS 2015

we learned before bringing you

cross-platform IDE for C and C++

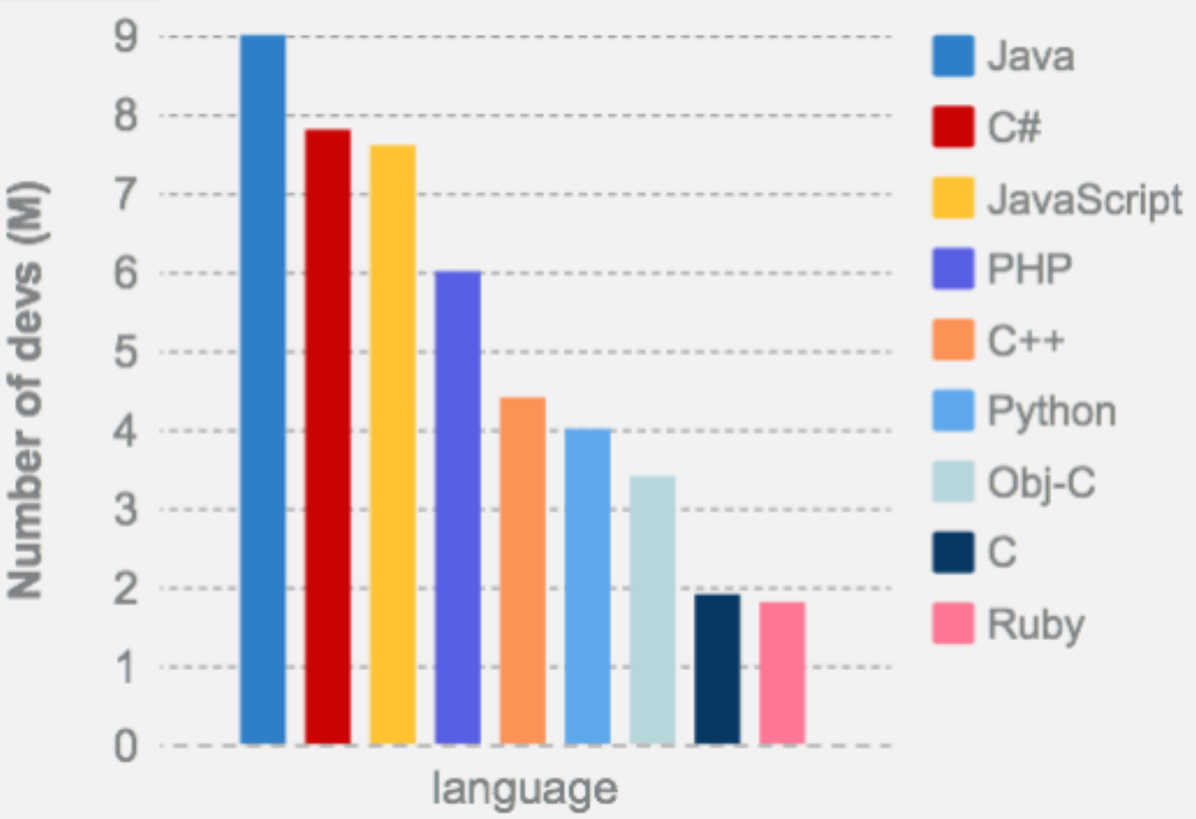
CLion

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.



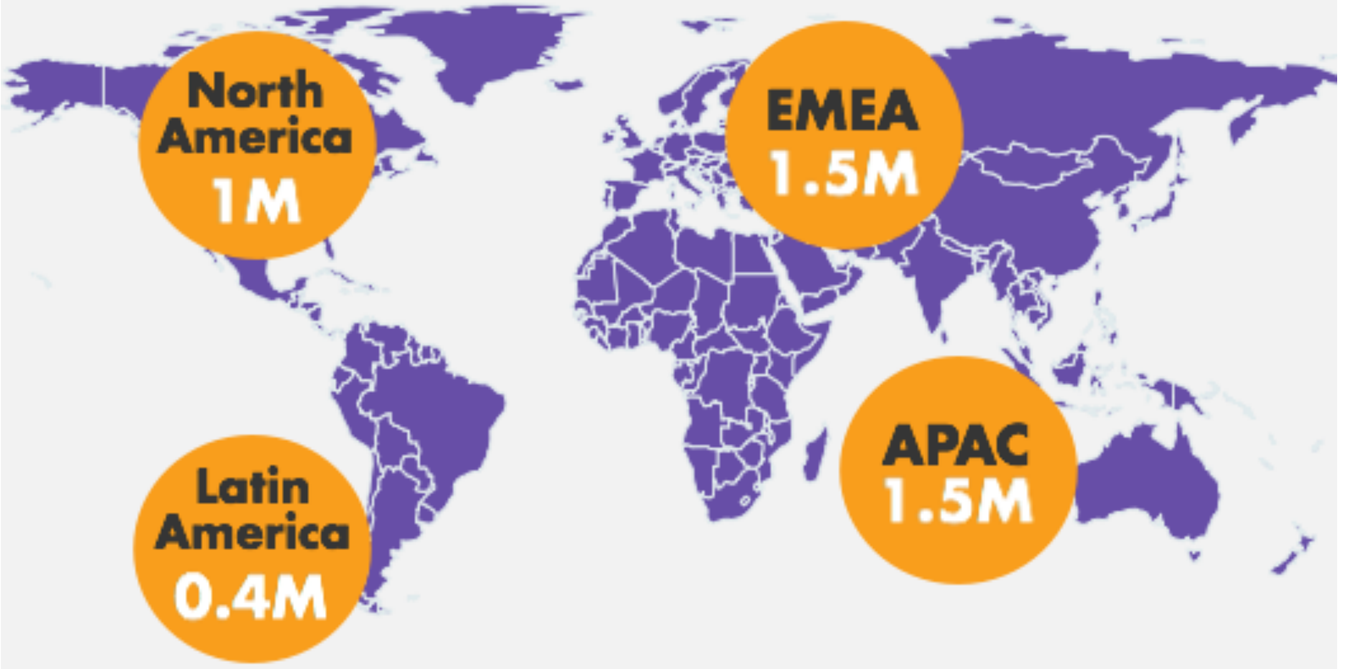
As many C++ developers as Python developers

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.

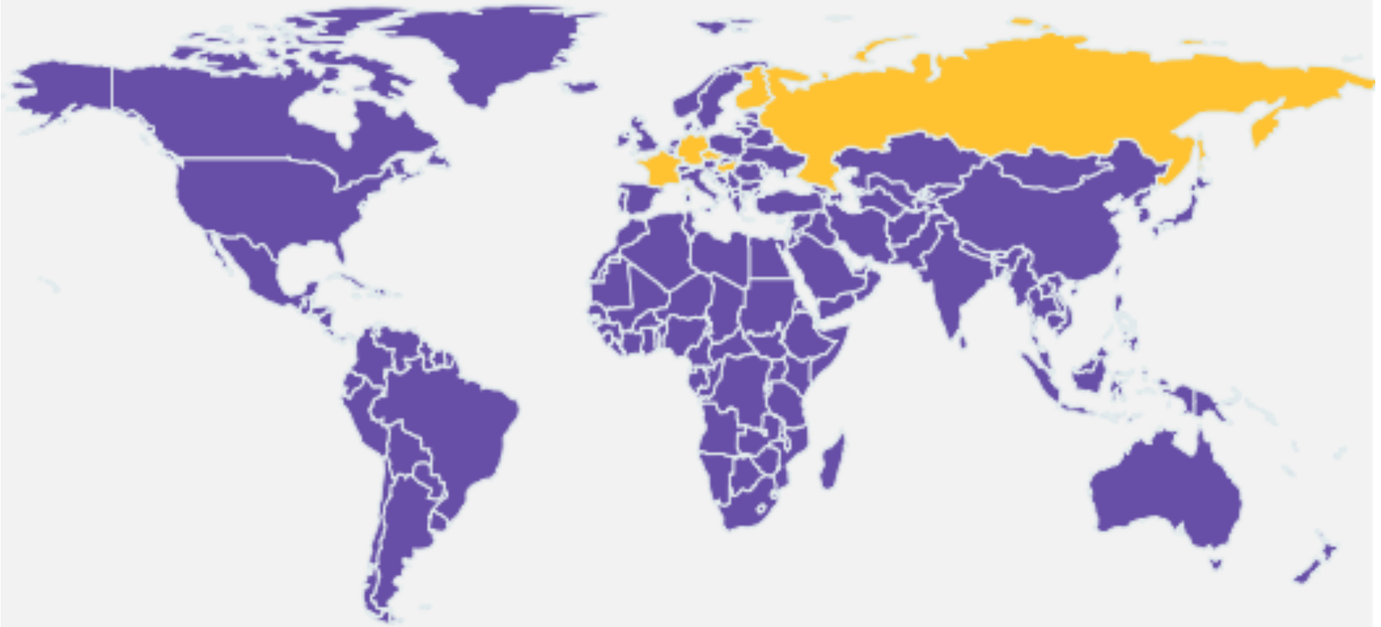
C++ developers by world region

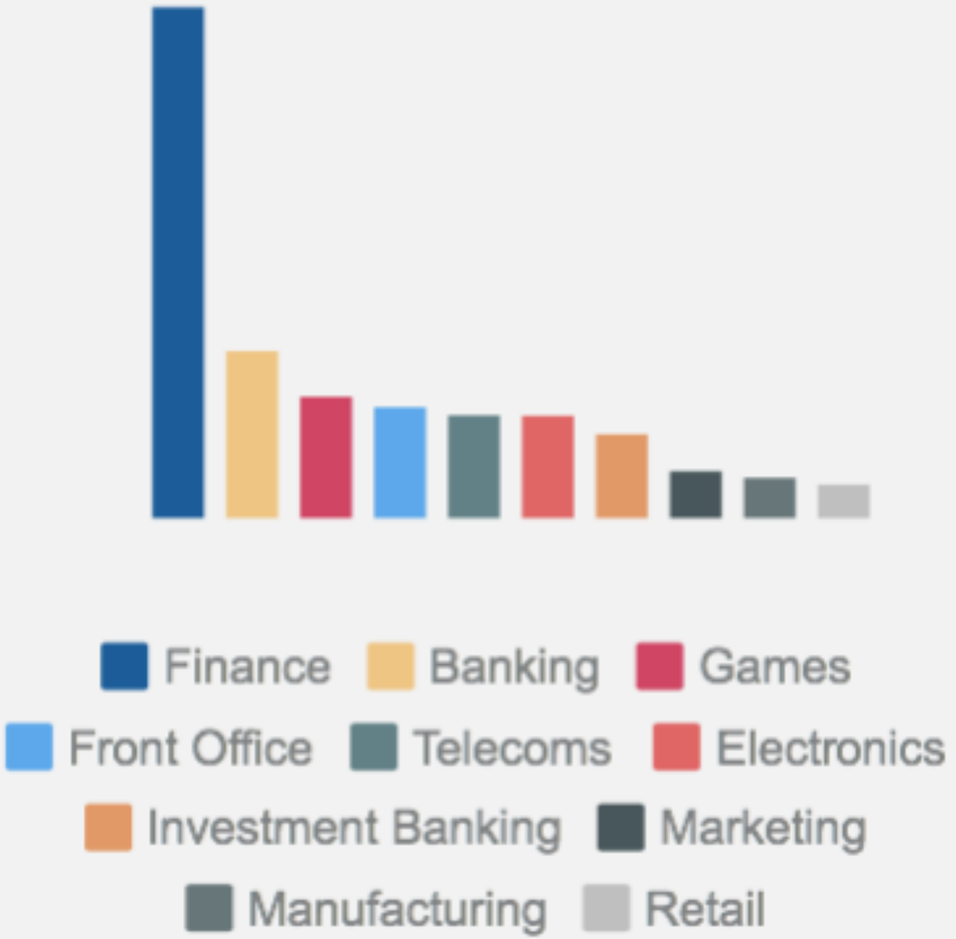
EMEA and Asia-Pacific are the most densely populated regions with regard to developers in general and C++ developers in particular.



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.





Industry distribution

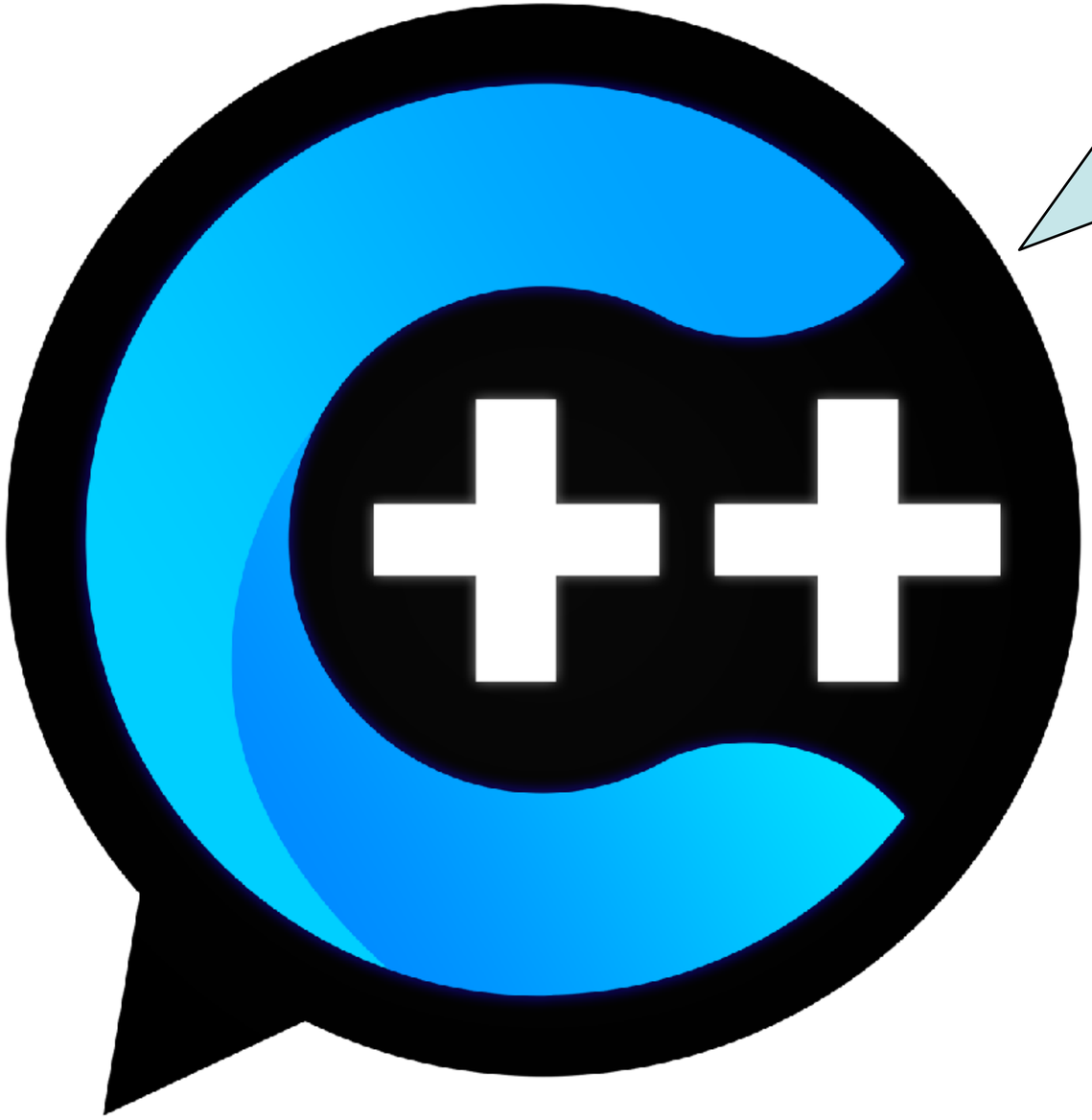
Based on the C/C++ job ads we analyzed, C++ is most used in industries like Finance and Banking.

Episode 145!

Over three years.

CppCast

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



Cpp.chat

reddit CPP

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 15, 2016	/r/cpp hits 40K subscribers
30K	Jul 22, 2015	/r/cpp hits 30K subscribers
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 we started tracking
Creation	May 26, 2008	/r/cpp is born

stackoverflow

c++ × 536057

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

249 asked today, 1485 this week

cppreference.com

Page Discussion

C++ reference

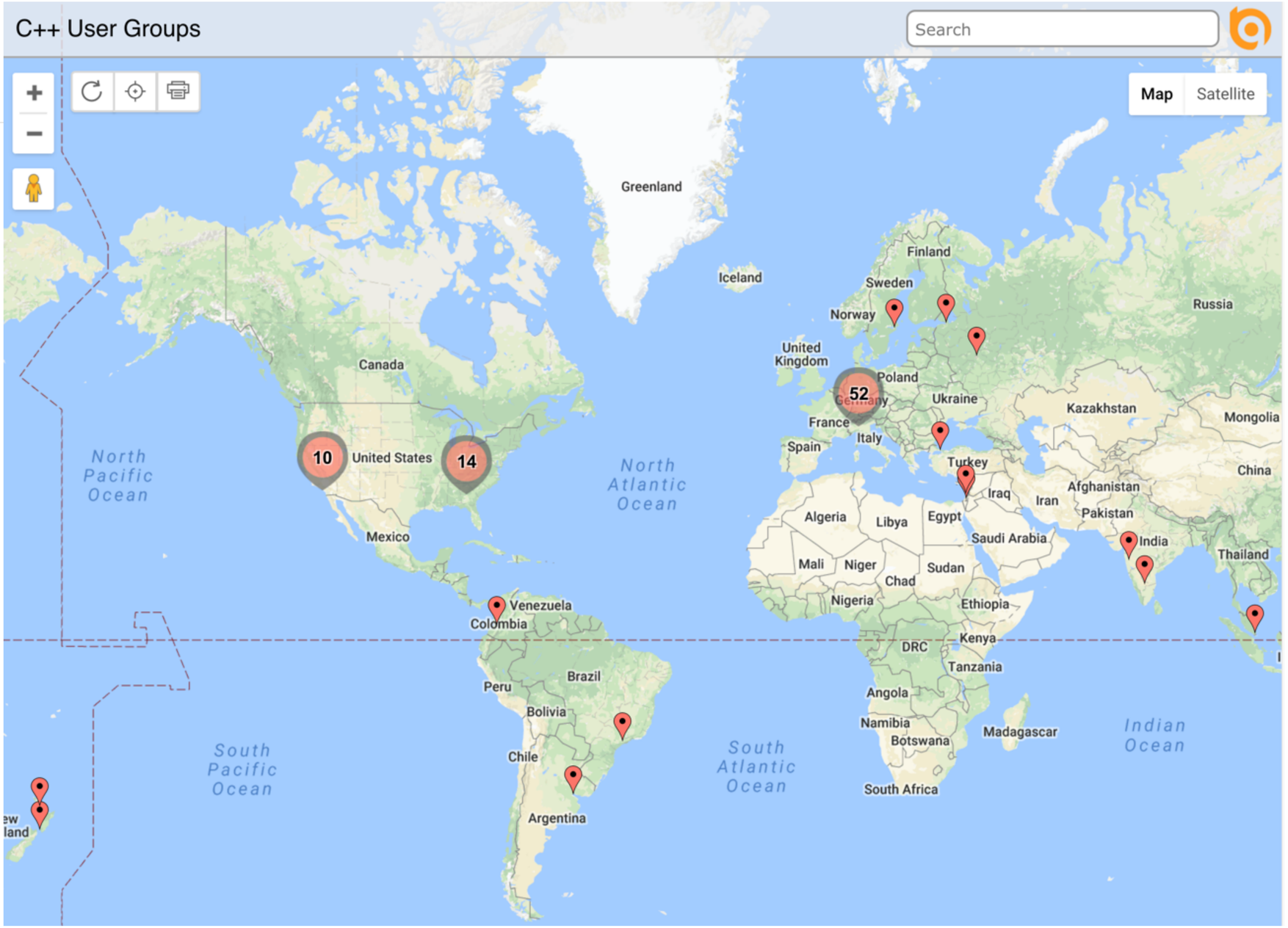
C++98, C++03, C++11, C++14, C++17, C++20

Cprogramming.com

Your resource for C and C++

User Groups Worldwide

Contents of this section:



Get Started! Tour Core Guidelines Super-FAQ Standardization About

FEATURES

Working Draft of the next standard

github

Current ISO C++ status

Upcoming ISO C++ meetings

Compiler conformance status

NAVIGATION

FAQ Home

FAQ RSS Feed

FAQ Help

SEARCH THIS WIKI

Go

Wiki Home > Conferences Worldwide

Conferences Worldwide

Contents of this section:

- / Pacific
- Europe
- North Amer

Get your visa invitations now!

FAQ Asia / P

C++ and System Software Summit - China

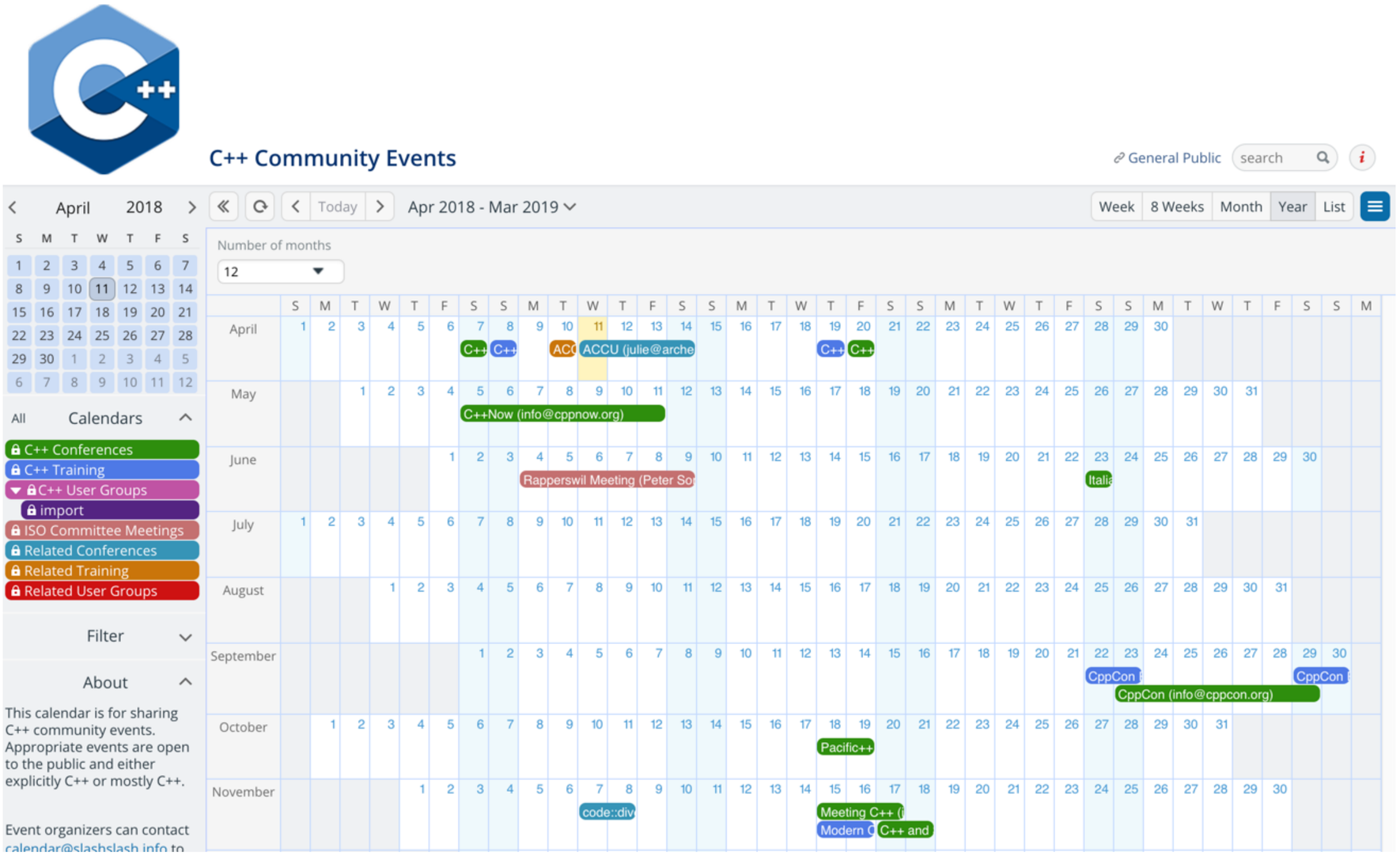
CppSiberia - Russia

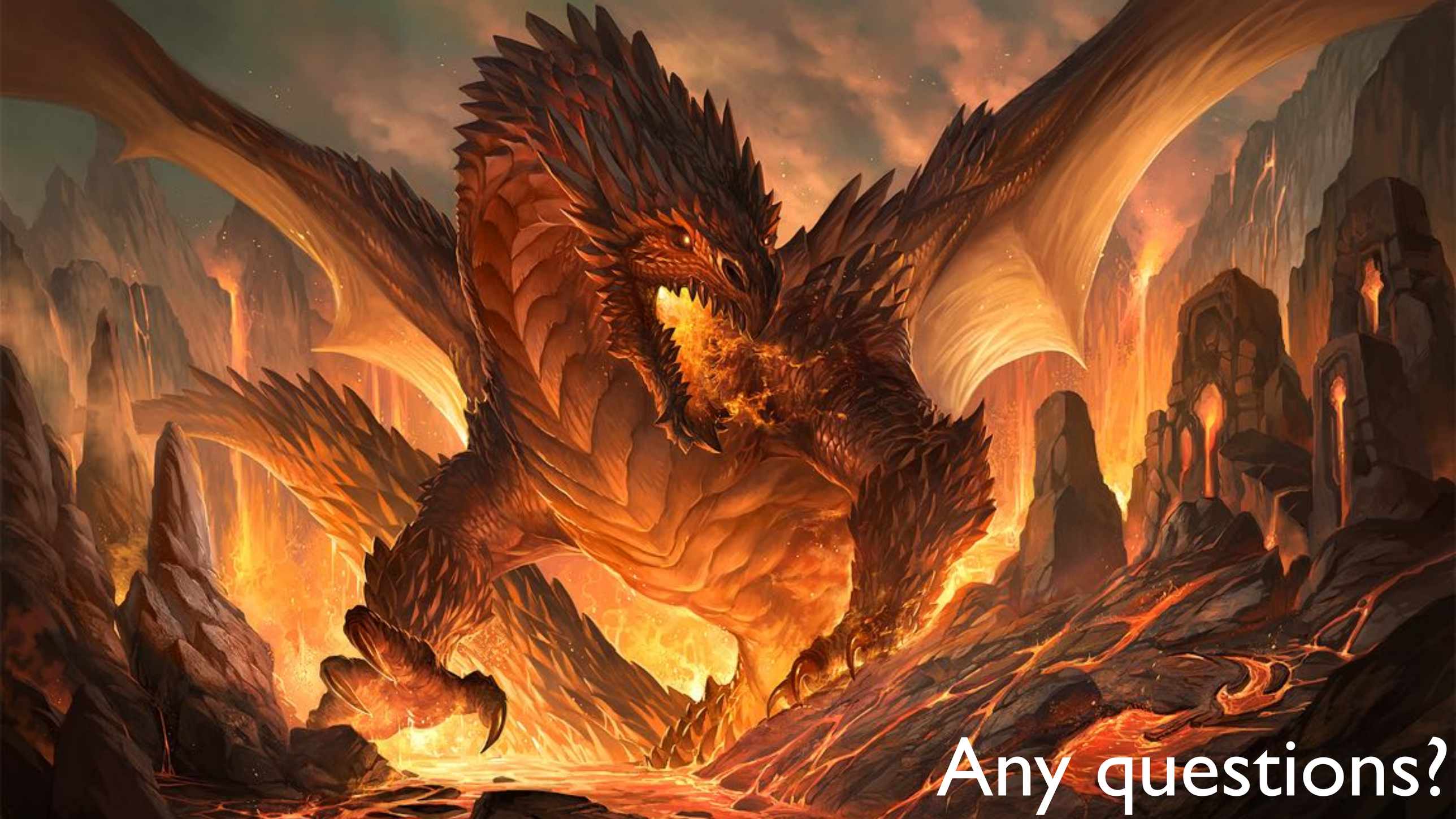
PacifiC++ - Australia/New Zealand/Asia

FAQ Europe

ACCU - UK


http://cppvap.wikidot.com/


C++ Community Events
General Public
search
April 2018
Today
Apr 2018 - Mar 2019
Week
8 Weeks
Month
Year
List
Number of months
12
All 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
April
May
June
July
August
September
October
November
C++Now (info@cppnow.org)
Rapperswil Meeting (Peter So
ACCU (julie@arche
CppCon
CppCon (info@cppcon.org)
Pacific++
code::div
C++ and


Any questions?