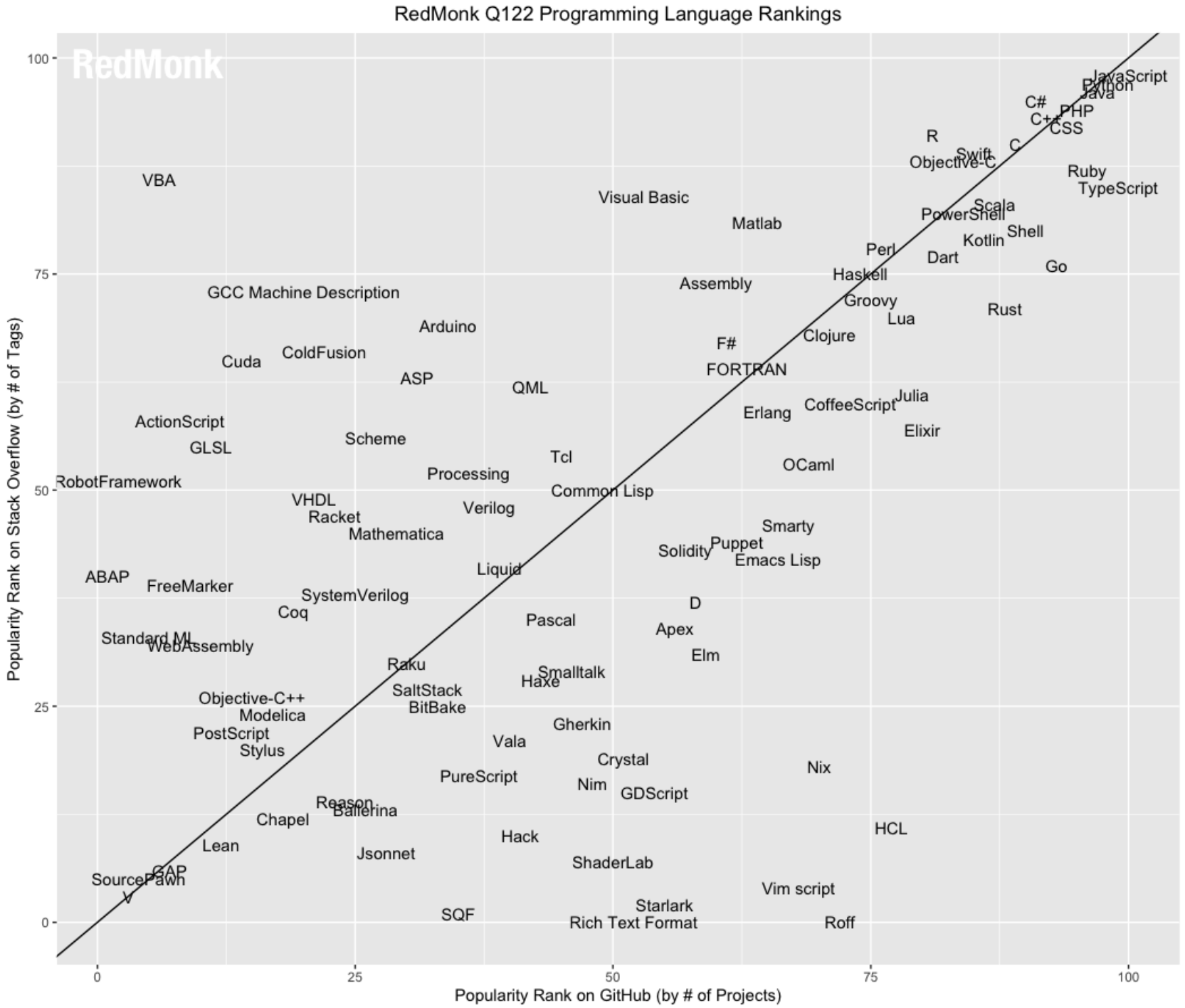
Programming in C++

Why and How

Andreas Füglistaler


RedMonk Q122 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
JavaScript
Python
Java
C#
PHP
C++
CSS
R
C
Swift
Objective-C
Ruby
TypeScript
Scala
PowerShell
Shell
Kotlin
Perl
Dart
Go
Haskell
Assembly
Groovy
Lua
Rust
Clojure
F#
FORTRAN
Julia
CoffeeScript
Erlang
Elixir
OCaml
Tcl
Processing
Common Lisp
Verilog
VHDL
Racket
Mathematica
Smarty
Puppet
Solidity
Emacs Lisp
Liquid
SystemVerilog
Coq
D
Pascal
Apex
Elm
Smalltalk
Haxe
Raku
SaltStack
BitBake
Objective-C++
Modelica
PostScript
Stylus
Gherkin
Vala
Crystal
Nix
PureScript
Nim
GDScript
Reason
Ballerina
Chapel
Hack
HCL
Lean
Jsonnet
ShaderLab
GAP
SourcePawn
V
Vim script
Starlark
SQF
Rich Text Format
Roff
VBA
Visual Basic
Matlab
GCC Machine Description
Arduino
ColdFusion
Cuda
ASP
QML
ActionScript
Scheme
GLSL
RobotFramework
ABAP
FreeMarker
Standard ML
WebAssembly

Top 10 Programming Languages

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

Top 10 Programming Languages

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

Interpreted

✔ No compilation step

x Interpreter must be available

✔ Fast development

x Slow execution

✔ Few security issues

x Limited hardware access

✔ Automatic memory management

✔ Buffer overflow check

x Slow/unpredictable

Top 10 Programming Languages

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

Compile to VM

✔ Compile once, run everywhere
x Limited hardware optimization

✔ Few security issues
x Limited hardware access

✔ Automatic memory management
✔ Buffer overflow check
x Slow/unpredictable

➤ Reasonable fast code
➤ Reasonable fast compilation

Top 10 Programming Languages

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

Procedural programming

✔ No hidden costs
x No hidden safety-nets

✔ Few language constructs
x No code reuse (generics, inheritance)

✔ Fast memory and buffer management
x Prone to errors

✔ Favored by the best (Torwalds, Thompson)
x Too dangerous for mere mortals

Top 10 Programming Languages

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

Modern Systems Languages

There are many (Rust, Go, D, Nim, Zig, Pony)

✔ Learned from past errors
✔ Fast compilation
x Will it still exist in 10 years?

✔ Memory and overflow strategy
x Either Boilerplate or Runtime cost

- Single- or Multi-Paradigm
- Lots or no hidden costs
- Flexible or static

Top 10 Programming Languages

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

Legacy Systems Programming

✔ Adopts/steals successful features
x Slow compilation due to header files
x Lots of non-orthogonal features

✔ Multi-paradigms
x Complicated language

✔ Zero-Cost abstractions
x Mostly a sales pitch

✔ Different memory and overflow strategies
x Default is manual

Future

It's difficult to make predictions, especially about the future

Use two languages

- Python/R/Lua/Scheme for convenience
- Rewrite the slowest part in C/C++/FORTRAN
- numpy, PyTorch, TensorFlow, LAMMPS, pyRosetta

Solving the "Two-language problem"

- Fast compilation:
  Swift, D, Go
- Fast interpretation (JIT):
  Julia, PyPy, luaJIT

Why C++

Runtime Speed $\gg$ Development Speed

Runtime Speed $\approx$ Safety

Runtime Speed $\ll$ Scalability

✔ Industrial standard

✔ Scales well in code- and team-size

✔ Modern Paradigms

Live C++ Example

`TVector.cpp`

C++ Tools

Compiler

gcc, clang, VC++

✔ Warnings are your friends

IDE

🖱 CLion, Eclipse, Visual Studio Code

⌨ emacs, vim

✔ Find files, grep in project

✔ On the fly compile errors

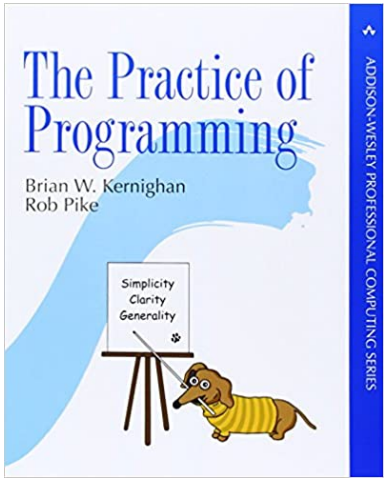
✔ Goto definition/declaration, find references

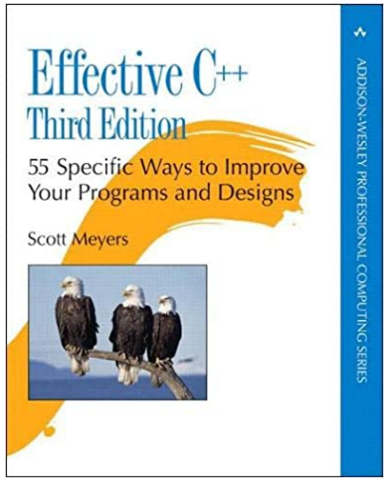
✔ Code formation, auto-complete

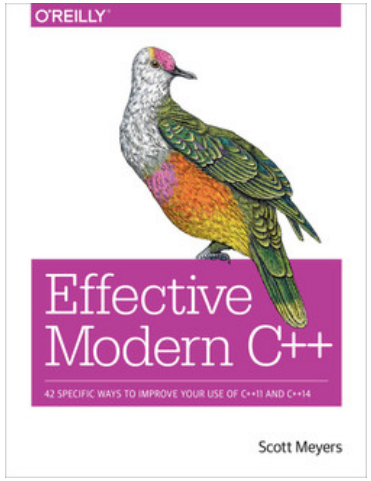
C++ Resources

cppreference.com

godbolt.org







C++ Talks

`https://www.youtube.com/user/CppCon`

`https://www.youtube.com/c/NDCConferences`

Philosophy: Bjarne Stroustrup, Herb Sutter

Basics: Scott Meyers, Jason Turner, Kate Gregory

Advanced: Andrei Alexandrescu, Chandler Carruth

C++ Core Guidelines

Initiated by Bjarne Stroustrup and Herb Sutter

`isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`

Aims

- ✔ Less error-prone and more maintainable
- ✔ Faster/easier initial development
- ✔ Zero-overhead principle
- ✔ Guidelines, not rules


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.