WIKIPEDIA

Comparison of programming languages

Programming languages are used for controlling the behavior of a machine (often a computer). Like natural languages, programming languages follow the rules for syntax and semantics.

There are thousands of programming languages[1] and new ones are created every year. Few languages ever become sufficiently popular that they are used by more than a few people, but professional programmers may use dozens of languages in a career.

Most programming languages are not standardized by an international (or national) standard, even widely used ones, such as Perl or Standard ML (despite the name). Notable standardized programming languages include ALGOL, C, C++, JavaScript (under the name ECMAScript), Smalltalk, Prolog, Common Lisp, Scheme (IEEE standard), Ada, Fortran, COBOL, SQL and XQuery.

Contents

General comparison

The following table compares general and technical information for a selection of commonly used programming languages. See the individual languages' articles for further information. Please note that the following table may be missing some information.

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
1C:Enterprise	Application, RAD, business, general, web, mobile	Yes		Yes	Yes	Yes	Yes	Yes	Object-based, Prototype-based programming	No
ActionScript 3.0	Application, client-side, web	Yes	Yes	Yes				Yes		1996, ECMA
Ada	Application, embedded, realtime, system	Yes	Yes[2]		Yes[3]	Yes[4]			concurrent,[5] distributed,[6]	1983, 2005, 2012, ANSI, ISO, GOST 27831-88[7]
Aldor	Highly domain-specific, symbolic computing	Yes	Yes	Yes						No
ALGOL 58	Application	Yes								No
ALGOL 60	Application	Yes			Yes	Yes				1960, IFIP WG 2.1, ISO[8]
ALGOL 68	Application	Yes		Yes	Yes	Yes			concurrent	1968, IFIP WG 2.1, GOST 27974-88,[9]
Ateji PX	Parallel application		Yes						pi calculus	No
APL	Application, data processing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	array-oriented, tacit	1989, ISO
Assembly language	General	Yes							any, syntax is usually highly specific, related to the target processor	IEEE 694-1985[10]
AutoHotkey	GUI automation (macros), highly domain-specific	Yes	Yes[11]		Yes			Yes		No
AutoIt	GUI automation (macros), highly domain-specific	Yes			Yes			Yes		No
Ballerina	Integration, agile, server-side, general	Yes	Yes	Yes	Yes			Yes	concurrent, transactional, statically and strongly typed programming, diagrammatic / visual programming	2018 De facto standard via Ballerina Language Specification[12]
Bash	Shell, scripting	Yes			Yes					No, but optionally POSIX.2 [13]
BASIC	Application, education	Yes			Yes					1983, ANSI, ISO, ECMA

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
BeanShell	Application, scripting	Yes	Yes	Yes			Yes			In progress, JCP[14]
BLISS	System				Yes					No
BlitzMax	Application, game	Yes	Yes		Yes		Yes			No
Boo	Application, game scripting		Yes							No
Bro	domain-specific, application	Yes						Yes		No
C	Application, system,[15] general purpose, low-level operations	Yes			Yes					1989, ANSI C89, ISO C90, ISO C99, ISO C11, ISO C18[16]
C++	Application, system	Yes	Yes	Yes	Yes	Yes				1998. ISO/IEC 2003, ISO/IEC 2011,ISO/IEC 2014,ISO/IEC 2017[17]
C#	Application, RAD, business, client-side, general, server-side, web	Yes	Yes	Yes[18]	Yes	Yes	Yes	Yes	structured, concurrent	2000, ECMA, ISO[19]
Clarion	General, business, web	Yes	Yes	Yes[20]						Unknown
Clean	General			Yes		Yes				No
Clojure	General			Yes					concurrent	No
CLU	General	Yes	Yes		Yes	Yes				No
COBOL	Application, business	Yes	Yes		Yes					ANSI X3.23 1968, 1974, 1985; ISO/IEC 1989:1985, 2002, 2014
Cobra	Application, business, general, web	Yes	Yes	Yes		Yes	Yes			No
ColdFusion (CFML)	Web		Yes		Yes					No

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
Common Lisp	General	Yes	Yes	Yes	Yes	Yes	Yes	Yes	extensible syntax, Array-oriented (https://github.com/equwal/CLAPL), syntactic macros, multiple dispatch, concurrent	1994, ANSI
COMAL 80	Education	Yes			Yes					No
Crystal	General purpose	Yes	Yes[21]	Yes	Yes				alpha stage[22]	No
Curry	Application			Yes		Yes			lazy evaluation, non-determinism	*De facto* standard via Curry Language Report
Cython	Application, general, numerical computing	Yes	Yes	Yes			Yes		aspect-oriented	No
D	Application, system	Yes	Yes	Yes	Yes	Yes	Yes		generative, concurrent	No
Dart	Application, web, server-side, mobile, IoT	Yes	Yes	Yes					structured	Ecma-408 standard
Dylan	Application		Yes	Yes						No
Eiffel	General, application, business, client-side, server-side, web (EWF)	Yes	Yes	Yes[23][24]		Yes	Yes Erl-G (http://se.inf.ethz.ch/old/people/leitner/erl_g)	Yes Agents	distributed SCOOP (https://docs.eiffel.com/book/solutions/concurrent-eiffel-scoop:), Void-safe (https://docs.eiffel.com/book/method/void-safe-programming-eiffel:)	2005, ECMA, ISO[25]
Elixir	Application, distributed			Yes				Yes	concurrent, distributed	No
Erlang	Application, distributed			Yes				Yes	concurrent, distributed	No
Euphoria	Application				Yes		Yes			No
Factor	General	Yes		can be viewed as		Yes	Yes		stack-oriented	No
FP				Yes						No
F#	Application	Yes	Yes	Yes	Yes	Yes	Yes	Yes		No
Forth	General	Yes		can be viewed as					stack-oriented	1994, ANSI

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
Fortran	Application, numerical computing	Yes	Yes	Yes	Yes	Yes			array-based, vectorized, concurrent, native distributed/shared-memory parallelism	1966, ANSI 66, ANSI 77, MIL-STD-1753, ISO 90, ISO 95, ISO 2003, ISO/IEC 1539-1:2010 (2008), ISO/IEC JTC1/SC22/WG5 N2145 (2018)
FreeBASIC	Application, numerical computing	Yes	Yes		Yes	Yes				No
Gambas	Application	Yes	Yes					Yes		No
Game Maker Language	Application, games	Yes	Yes					Yes		No
GLBasic	Application, games	Yes	Yes		Yes				simple object-oriented	No
Go	Application, web, server-side	Yes	[26]		Yes		Yes	Yes	concurrent	*De facto* standard via Go Language Specification
Gosu	Application, general, scripting, web	Yes	Yes			Yes	Yes			No
GraphTalk	Application		Yes						logic	No
Groovy	Application, general, scripting, web	Yes	Yes	Yes	Yes	Yes	Yes	Yes	meta-programming	In progress, JCP[27]
Harbour	Application, business, data processing, general, web	Yes	Yes	Yes	Yes	Yes	Yes		declarative	No
Haskell	Application			Yes		Yes			lazy evaluation	2010, Haskell 2010[28]
Haxe	Application, general, web	Yes	Yes	Yes		Yes	Yes			No
HyperNext	Application, education				Yes			Yes	weakly typed	No
HyperTalk	Application, RAD, general		Yes					Yes	weakly typed	Unknown
Io	Application, host-driven scripting	Yes	Yes							No
IPL	General			Yes						Unknown

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
ISLISP	General	Yes	Yes	Yes		Yes				1997, ISO
J	Data processing								array-oriented, function-level, tacit	No
JADE	Application, distributed	Yes	Yes							No
Java	Application, business, client-side, general, mobile development, server-side, web	Yes	Yes	Yes	Yes	Yes	Yes	Yes	concurrent	*De facto* standard via Java Language Specification
JavaScript	Client-side, server-side, web	Yes	Yes	Yes	Yes		Yes	Yes	prototype-based	1997, ECMA
Joy	Research			Yes					stack-oriented	No
Julia	General, technical computing	Yes	Yes	Yes	Yes	Yes	Yes		multiple dispatch, meta, scalar and array-oriented, parallel, concurrent, distributed ("cloud")	No
K	Data processing, business								array-oriented, tacit	Unknown
Kotlin	Application, mobile development, server-side, client-side, web	Yes	Yes	Yes	Yes	Yes	Yes[29]	Yes		No
Ksh	Shell, scripting	Yes	Yes		Yes				several variants, custom programmable, dynamic loadable modules	1992, POSIX.2 [30]
LabVIEW (G)	Application, industrial instrumentation-automation	Yes	Yes	Yes				Yes	dataflow, visual	No
Lisp	General			Yes						Unknown
LiveCode	Application, RAD, general		Yes					Yes	weakly typed	No
Logtalk	Artificial intelligence, application		Yes				Yes	Yes	logic	No

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
LSL	Virtual worlds content scripting and animation	Yes			Yes			Yes	Scripts exist in in-world objects	Maybe[31]
Lua	Application, embedded scripting	Yes	Yes[32]	Yes	Yes		Yes		aspect-oriented, prototype-based	No[33]
Maple	Symbolic computation, numerical computing	Yes	Yes	Yes	Yes				distributed	No
Mathematica	Symbolic language	Yes	Yes	Yes	Yes	Yes	Yes	Yes	logic, distributed	No
MATLAB	Highly domain-specific, numerical computing	Yes	Yes		Yes					No
Modula-2	Application, system	Yes				Yes				1996, ISO[34]
Modula-3	Application	Yes	Yes			Yes				No
MUMPS (M)	Application, databases	Yes			Yes				concurrent, multi-user, NoSQL, transaction processing	1977, ANSI
Nim	Application, general, web, scripting, system	Yes	Yes	Yes	Yes	Yes	Yes		multiple dispatch, Concurrent, meta	No
Oberon	Application, system	Yes	Yes							No
Object Pascal	Application, general, mobile app, web	Yes	Yes		Yes	Yes	Yes	Yes	structured	No
Objective-C	Application, general	Yes	Yes		Yes		Yes		concurrent	No
OCaml	Application, general	Yes	Yes	Yes	Yes	Yes				No
Occam	General	Yes			Yes				concurrent, process-oriented	No
Opa	Web applications	Yes		Yes		Yes			distributed	No
OpenLisp	General, Embedded Lisp Engine	Yes	Yes	Yes		Yes				Supersedes ISLISP, ISO

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
Oxygene	Application	Yes	Yes			Yes				No
Oz-Mozart	Application, distribution, education	Yes	Yes	Yes					concurrent, logic	No
Pascal	Application, education	Yes			Yes					1983, ISO[35]
Perl	Application, scripting, text processing, Web	Yes	Yes	Yes	Yes	Yes	Yes			No
PHP	Server-side, web application, web	Yes	Yes[36]	Yes[37]	Yes		Yes			"De facto" standard via language specification and Requests for Comments (RFCs)
PL/I	Application	Yes	Yes		Yes					1969, ECMA-50 (1976)
Plus	Application, system development	Yes			Yes					No
PostScript	Graphics, page description	Yes			Yes				concatenative, stack-oriented	Yes, as the PostScript Reference Manual[38]
PowerShell	Administration, application, general, scripting	Yes	Yes	Yes	Yes		Yes		pipeline	No
Prolog	Application, artificial intelligence			Yes	Yes		Yes		logic, declarative	1995, ISO/IEC 13211-1:1995, TC1 2007, TC2 2012, TC3 2017
PureBasic	Application				Yes					No
Python	Application, general, web, scripting, artificial intelligence, scientific computing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	aspect-oriented	"De facto" standard via Python Enhancement Proposals (PEPs)
R	Application, statistics	Yes	Yes	Yes	Yes		Yes			No

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
Racket	Education, general, scripting		Yes	Yes	Yes		Yes		modular, logic, meta	No
Raku	Scripting, text processing, glue	Yes	Yes	Yes	Yes	Yes	Yes		aspect-oriented, array, lazy evaluation, multiple dispatch, metaprogramming	Yes
REALbasic	Application				Yes					Unknown
Rebol	Distributed	Yes	Yes	Yes	Yes		Yes	Yes	dialected	No
REXX	Scripting	Yes	Yes (NetRexx and Object REXX dialects)	No	Yes		No	No		1996 (ANSI X3.274-1996)
RPG	Application, system	Yes			Yes					No
Ruby	Application, scripting, web	Yes	Yes	Yes			Yes		aspect-oriented	2011(JIS X 3017), 2012(ISO/IEC 30170)
Rust	Application, server-side, system, web	Yes	Yes[39]	Yes	Yes	Yes		Yes	concurrent	No
S	Application, statistics	Yes	Yes	Yes	Yes					No
S-Lang	Application, numerical, scripting	Yes			Yes					No
Scala	Application, distributed, web	Yes	Yes	Yes		Yes	Yes	Yes		*De facto* standard via Scala Language Specification (SLS)
Scheme	Education, general			Yes					extensible syntax	1998, R^6RS
Seed7	Application, general, scripting, web	Yes	Yes			Yes	Yes		multi-paradigm, extensible, structured	No
Simula	Education, general	Yes	Yes					Yes	discrete event simulation, multi-threaded (quasi-parallel) program execution	1968

Language	Intended use	Imperative	Object-oriented	Functional	Procedural	Generic	Reflective	Event-driven	Other paradigm(s)	Standardized?
Small Basic	Application, education, games	Yes						Yes	component-oriented	No
Smalltalk	Application, general, business, artificial intelligence, education, web	Yes	Yes	Yes	Yes		Yes	Yes	concurrent, declarative	1998, ANSI
SNOBOL	Text processing									Unknown
Standard ML	Application	Yes		Yes		Yes				1997, SML '97[40]
Swift	Application, general	Yes	Yes	Yes	Yes	Yes	Yes	Yes	concurrent, declarative, protocol-oriented	No
Tcl	Application, scripting, web	Yes	Yes	Yes	Yes		Yes	Yes		No
Visual Basic	Application, RAD, education, business, general, (Includes VBA), office automation	Yes	Yes			Yes		Yes	component-oriented	No
Visual Basic .NET	Application, RAD, education, web, business, general	Yes	Yes	Yes	Yes	Yes	Yes	Yes	structured, concurrent	No
Visual FoxPro	Application		Yes						data-centric, logic	No
Visual Prolog	Application	Yes	Yes	Yes				Yes	declarative, logic	No
Wolfram Language	Symbolic language	Yes	Yes	Yes	Yes	Yes	Yes	Yes	logic, distributed	No
XL		Yes	Yes						concept programming	No
Xojo	Application, RAD, general, web	Yes	Yes		Yes		Yes	Yes		No
XPath/XQuery	Databases, data processing, scripting			Yes					tree-oriented	W3C 1999 XPath 1, 2010 XQuery 1, 2014 XPath/XQuery 3.0
Zsh	Shell, scripting	Yes			Yes				loadable modules	No

Type systems

Failsafe I/O and system calls

Most programming languages will print an error message or throw an exception if an input/output operation or other system call (e.g., chmod, kill) fails, unless the programmer has explicitly arranged for different handling of these events. Thus, these languages fail safely in this regard.

Some (mostly older) languages require that the programmer explicitly add checks for these kinds of errors. Psychologically, different cognitive biases (e.g., optimism bias) may affect novice and experts alike and these omissions can lead to erroneous behavior.

Language	Failsafe I/O
1C:Enterprise	Yes
Ada	Yes (exceptions)
ALGOL	Yes (exceptions or return value depending on function)
AutoHotkey	No (global ErrorLevel must be explicitly checked)
Bash	Optional[FSIO 1]
Ballerina	Yes
Bro	Yes
C	No[FSIO 2]
C++	Some (STL iostreams throw on failure but C APIs like stdio or POSIX do not)[FSIO 2]
C#	Yes
COBOL	No
Common Lisp	Yes ("conditions and restarts" system)
Curry	Yes
D	Yes (throwing on failure) [FSIO 3]
Eiffel	No – It actually depends on the library and it is not defined by the language
Erlang	Yes
Fortran	Yes
GLBasic	No – Will generally cause program to crash
Go	Yes (unless result explicitly ignored)
Gosu	Yes
Harbour	Yes
Haskell	Yes
ISLISP	Yes
Java	Yes
Julia	Yes
Kotlin	Yes
LabVIEW	Yes
Lua	No (some functions do not warn or throw exceptions)
Mathematica	Yes
Object Pascal	Some
Objective-C	Yes (exceptions)

Language	Failsafe I/O

Language	Failsafe I/O
OCaml	Yes (exceptions)
OpenLisp	Yes
Perl	No[FSIO 4]
PHP	Yes
Python	Yes
Raku	Yes
Rebol	Yes
Rexx	Yes (with optional **signal on**... trap handling)
RPG	No
Ruby	Yes
Rust	Yes (unless result explicitly ignored)
S	Unknown
Smalltalk	Yes
Scala	Yes[FSIO 5]
Standard ML	Yes
Swift ≥ 2.0	Yes (exceptions)
Tcl	Yes
Visual Basic	Yes
Visual Basic .NET	Yes
Visual Prolog	Yes
Wolfram Language	Yes
Xojo	Yes
XPath/XQuery	Yes (exceptions)
Language	Failsafe I/O

1. `set -e` enables termination if any unchecked exit status is nonzero.
2. gcc can warn on unchecked errno. Newer versions of Visual Studio usually throw exceptions on failed I/O when using stdio.
3. https://dlang.org/phobos/std_stdio.html
4. Considerable error checking can be enabled optionally, but by default Perl is not failsafe.
5. Scala runs on the Java Virtual Machine from which it inherits the runtime exception handling.

Expressiveness

The literature on programming languages contains an abundance of informal claims about their relative expressive power, but there is no framework for formalizing such statements nor for deriving interesting consequences.[43] This table provides two measures of expressiveness from two different sources. An additional measure of expressiveness, in GZip bytes, can be found on the Computer Language Benchmarks Game.[44]

Language	Statements ratio[41]	Lines ratio[42]
C	1	1
C++	2.5	1
Fortran	2	0.8
Java	2.5	1.5
Perl	6	6
Smalltalk	6	6.25
Python	6	6.5

Benchmarks

Benchmarks are designed to mimic a particular type of workload on a component or system. The computer programs used for compiling some of the benchmark data in this section may not have been fully optimized, and the relevance of the data is disputed. The most accurate benchmarks are those that are customized to your particular situation. Other people's benchmark data may have some value to others, but proper interpretation brings many challenges. The Computer Language Benchmarks Game site warns against over-generalizing from benchmark data, but contains a large number of micro-benchmarks of reader-contributed code snippets, with an interface that generates various charts and tables comparing specific programming languages and types of tests.[45]

Timeline of specific language comparisons

- 1974 – Comparative Notes on Algol 68 and PL/I[46] – S. H. Valentine – November 1974
- 1976 – Evaluation of ALGOL 68, JOVIAL J3B, Pascal, Simula 67, and TACPOL Versus TINMAN – Requirements for a Common High Order Programming Language.
- 1977 – A comparison of PASCAL and ALGOL 68[47] – Andrew S. Tanenbaum – June 1977.
- 1993 – Five Little Languages and How They Grew – BLISS, Pascal, ALGOL 68, BCPL & C – Dennis M. Ritchie – April 1993.
- 2009 – On Go – oh, go on – How well will Google's Go stand up against Brand X programming language? – David Given – November 2009

See also

To display all pages, subcategories and images click on the "►":

▼ Lists of programming languages (19 P)
- List of programming languages
- List of programming languages by type
- Lists of programming languages
- List of programming languages for artificial intelligence
- List of audio programming languages
- List of BASIC dialects
- List of C-family programming languages
- List of CLI languages

List of concurrent and parallel programming languages
List of educational programming languages
Generational list of programming languages
List of JVM languages
List of Lisp-family programming languages
Non-English-based programming languages
List of object-oriented programming languages
Comparison of open-source programming language licensing
List of reflective programming languages and platforms
Timeline of programming languages
Unisys MCP programming languages

- Comparison of basic instructions of programming languages
- Comparison of programming languages (syntax)
- Comparison of programming paradigms
- Comparison of integrated development environments
- Comparison of multi-paradigm programming languages
- Measuring programming language popularity
- TIOBE index

References

1. As of May 2006 Diarmuid Pigott's Encyclopedia of Computer Languages (http://hopl.murdoch.edu.au/) Archived (https://web.archive.org/web/20110220044217/http://hopl.murdoch.edu.au/) 2011-02-20 at the Wayback Machine hosted at Murdoch University, Australia lists 8512 computer languages.
2. Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3 (http://www.adaic.org/standards/05rm/html/RM-TTL.html), 3.9 Tagged Types and Type Extensions (http://www.adaic.org/standards/05rm/html/RM-3-9.html)
3. Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3 (http://www.adaic.org/standards/05rm/html/RM-TTL.html), Section 6: Subprograms (http://www.adaic.org/standards/05rm/html/RM-6.html)
4. Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3 (http://www.adaic.org/standards/05rm/html/RM-TTL.html), Section 12: Generic Units (http://www.adaic.org/standards/05rm/html/RM-12.html)
5. Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3 (http://www.adaic.org/standards/05rm/html/RM-TTL.html), Section 9: Tasks and Synchronization (http://www.adaic.org/standards/05rm/html/RM-9.html)
6. Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3 (http://www.adaic.org/standards/05rm/html/RM-TTL.html) Annex E: Distributed Systems (http://www.adaic.org/standards/05rm/html/RM-E.html)
7. "Vak.ru" (https://web.archive.org/web/20170330020459/http://vak.ru/lib/exe/fetch.php/book/gost/pdf/gost-27831-88.pdf) (PDF). Archived from the original (http://vak.ru/lib/exe/fetch.php/book/gost/pdf/gost-27831-88.pdf) (PDF) on 2017-03-30. Retrieved 2008-08-09.
8. ISO 1538:1984 (http://www.open-std.org/jtc1/sc22/docs/oldwgs/wg6.html)
9. "Vak.ru" (https://web.archive.org/web/20170324231641/http://vak.ru/lib/exe/fetch.php/book/gost/pdf/gost-27974-88.pdf) (PDF). Archived from the original (http://vak.ru/lib/exe/fetch.php/book/gost/pdf/gost-27974-88.pdf) (PDF) on 2017-03-24. Retrieved 2008-08-09.
10. IEEE 694-1985 (https://standards.ieee.org/standard/694-1985.html)
11. Objects - Definition & Usage (https://www.autohotkey.com/docs/Objects.htm)
12. "Ballerina Language Specification" (https://ballerina.io/res/Ballerina-Language-Specification-WD-2015-05-01.pdf) (PDF). WSO2. 2018-05-01. Retrieved 2018-05-03.
13. POSIX.2, Shell and Utilities, Command Interpreter (IEEE Std 1003.2-1992.)
14. JSR 274 (http://jcp.org/en/jsr/detail?id=274)
15. bell-labs.com (https://www.bell-labs.com/usr/dmr/www/chist.html)
16. ANSI C89, ISO/IEC 9899:1990, 1999, 2011, 2018 (http://www.open-std.org/JTC1/SC22/WG14/)
17. ISO/IEC 14882:1998, 2003, 2011, 2014, 2017 (http://www.open-std.org/JTC1/SC22/WG21/)

18. Codeproject.com: Functional Programming in C# 3.0 using Lambda Expression (http://www.codeproject.com/KB/cs/intro_functional_csharp.aspx)
19. ECMA-334; ISO/IEC 23270:2006
20. Softvelocity.com (http://www.softvelocity.com)
21. https://github.com/crystal-lang/crystal#why
22. https://github.com/crystal-lang/crystal#status
23. Basic Eiffel language mechanisms (http://se.ethz.ch/~meyer/publications/online/eiffel/basic.html#Agents:)
24. Closure (computer programming)
25. ECMA-367; ISO/IEC 25436:2006
26. The Go Programming Language (FAQ) (https://golang.org/doc/faq#Is_Go_an_object-oriented_language)
27. JSR 241 (http://jcp.org/en/jsr/detail?id=241)
28. "The Haskell 2010 Language Report" (http://www.haskell.org/onlinereport/haskell2010/). Retrieved 2011-12-07. Most Haskell implementations extend the Haskell 2010 standard.
29. "M8 is out!" (http://blog.jetbrains.com/kotlin/2014/07/m8-is-out/). "As a first peek into the future reflective capabilities of Kotlin, you can now access properties as first-class objects in Kotlin"
30. POSIX.2, Shell and Utilities, Command Interpreter (IEEE Std 1003.2-1992.)
31. "De facto" reference is the Second Life implementation of LSL. Halcyon (Inworldz) and Open Sims propose compatible implementations with additionnal functions
32. Lua doesn't have explicit "object" type (more general type of "table" is used for object definition), but does have explicit syntax for object method calling
33. Version releases are accompanied with a definitive Lua Reference Manual showing full syntax and semantics; a reference implementation, and a test suite. These are used to generate other Lua VM implementations and compilers such as Kahlua and LLVM-Lua.
34. ISO/IEC 10514-1:1996
35. ISO 7185
36. PHP Manual (http://php.net/manual/en/index.php), Chapter 19. Classes and Objects (PHP 5) (http://php.net/manual/en/language.oop5.php),
37. PHP Manual (http://php.net/manual/en/index.php), Chapter 17. Functions (http://php.net/manual/en/language.functions.php)
38. "PostScript Language Reference Manual" (https://web.archive.org/web/20170218093716/https://www.adobe.com/products/postscript/pdfs/PLRM.pdf) (PDF). Archived from the original (https://www.adobe.com/products/postscript/pdfs/PLRM.pdf) (PDF) on 2017-02-18. Retrieved 2017-02-18.
39. Is Rust an Object-Oriented Programming Language? (https://doc.rust-lang.org/book/ch17-00-oop.html)
40. SMLNJ.org (http://www.smlnj.org/sml97.html)
41. Data from *Code Complete* (https://books.google.com/books?id=3JfE7TGUwvgC&pg=PT100). p. 100. The *Statements ratio* column "shows typical ratios of source statements in several high-level languages to the equivalent code in C. A higher ratio means that each line of code in the language listed accomplishes more than does each line of code in C.
42. The ratio of line count tests won by each language to the number won by C when using the *Compare to* feature at Jon McLoone (November 14, 2012). "Code Length Measured in 14 Languages" (https://web.archive.org/web/20121119043607/https://blog.wolfram.com/2012/11/14/code-length-measured-in-14-languages/). Archived from the original (https://blog.wolfram.com/2012/11/14/code-length-measured-in-14-languages/) on 2012-11-19. C gcc was used for C, C++ g++ was used for C++, FORTRAN G95 was used for FORTRAN, Java JDK Server was used for Java, and Smalltalk GST was used for Smalltalk.
43. Felleisen, Matthias. *On the Expressive Power of Programming Languages*. ESOP '90 3rd European Symposium on Programming. CiteSeerX 10.1.1.51.4656 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.4656).
44. "How programs are measured | Computer Language Benchmarks Game" (https://benchmarksgame-team.pages.debian.net/benchmarksgame/how-programs-are-measured.html#source-code). *benchmarksgame-team.pages.debian.net/benchmarksgame*. Retrieved 2018-05-29.
45. "The Ultimate Benchmark | The Computer Language Benchmarks Game" (https://benchmarksgame-team.pages.debian.net/benchmarksgame/dont-jump-to-conclusions.html). *benchmarksgame-team.pages.debian.net/benchmarksgame*. Retrieved 2018-05-29.
46. Valentine, S. H. (November 1974). "Comparative Notes on Algol 68 and PL/I" (https://doi.org/10.1093%2Fcomjnl%2F17.4.325). *The Computer Journal*. **17** (4): 325–331. doi:10.1093/comjnl/17.4.325 (https://doi.org/10.1093%2Fcomjnl%2F17.4.325).
47. http://dare.ubvu.vu.nl/bitstream/1871/2609/1/11054.pdf

Further reading

- Cezzar, Ruknet (1995). *A Guide to Programming Languages: Overview and Comparison* (https://archive.org/details/authenticationsy0000oppl). ISBN 978-0-89006-812-0.