Year	Language	Developed by	Features	Firsts
1936-45	Plankalkul	Konrad Zuse	Designed for theoretical machine that was never built. 2's complement integers, floating point, arrays, records, nested records, iteration, expressions resembling assertions. Notation was awkward. Developed in isolation in Bavaria during the war, was unknown to the Allies, little was published about it until 1970.	First high level language.
1949	Short Code	John Mauchly	No complete description published but programming manual has survived. Codes were byte-pair values (UNIVAC had 12 6-bit bytes per word). Implemented as pure interpreter. Approx. 50 times slower than machine code.	labeling of memory locations (named variables)
1951-53	A-0, A-1, A-2 for UNIVAC	Grace Hopper	No program code has survived. Did macro expansion of text into blocks of machine code.	
1952	FORTRAN 0	? Alick Glennie? ? Laning & Zirerler?	Translated arithmetic expressions. Because of the high cost of computing, object code speed was a primary design consideration	First compiled high level language. First implementation of arrays & subroutines
1954	Speedcoding	John Backus	Allowed IBM 701 to serve as a 3-address FP calculator.	Automatically incremented address registers
1957	FORTRAN I	John Backus		I/O formatting, 6-character variable names, user-defined subroutines, if, do.
1957	FLOW-MATIC	Grace Hopper		First attempt to write data processing statements in English-like constructs
1958	LISP	John McCarthy	Lists. Code & data homoiconic (represented same way), meaning a program could write code that could then be interpreted. Dominated AI computing for 20+ years	First functional programming language. First JIT compiler demonstrated 1968
1958	FORTRAN II		Fixed bugs in FORTRAN I.	First independent compilation of subroutines.

Year	Language	Developed by	Features	Firsts
1958	ALGOL 58		Presented as report, met with enthusiasm, never actually implemented. Assignments were originally of form expression >= variable, later changed (mostly at insistence of the Americans) to variable := expression.	First internationally developed language. First language designed to be cross-platform. Formalized ideas of data type, compound statements.
1960	ALGOL 60		Recursion. Quickly became standard method of describing algorithms. Most imperative languages owe a debt to it. Never became a huge success on its own as a programming language.	First use of block structure. Pass by value or reference. Stack-dynamic arrays. First use of BNF.
1960	COBOL	Grace Hopper	Designed for ease of use so that managers could read programs, even at the cost of execution speed. Separate statements for data description & execution, in different parts of the program.	First language designed specifically for business (several companies had languages under development; COBOL got there first). First high-level macro language. First implementation of hierarchial data structures. 30-character names including hyphens. First language whose use was mandated by DoD.
1960	APL	Iverson	Optimized for matrix operations. Used through IBM printing terminals,which had additional symbols available. Most commands are one character. Extremely terse, compact, fast, near-perfect orthogonality. Nearly impossible to maintain or modify; regarded as best used for throwaway or write-once programming	

Year	Language	Developed by	Features	Firsts
1960	JOVIAL	Schwartz	Developed by Jules Schwartz, SDC Corp., for programming electronics in military aircraft. IAL stood for International Algorithmic Language, the name originally proposed for ALGOL 58. The name was originally OVIAL, which was opposed for various reasons; someone proposed JOVIAL, with no particular meaning. A committee member joked that it stood for Jules' Own Version of the International Algorithmic Language, and that's what stuck. JOVIAL was the primary programming language of the US Air Force for years.	
1964	PL/I	IBM	Recursion could be disabled for more efficient linkage if a program had no recursion. Many firsts were poorly implemented; it enjoyed some success in the 1970s but generally regarded now as a brilliant, beautiful failure.	First attempt at a language suitable for multiple purposes —scientific AND business AND logic/AI programming. First concurrent execution of subprograms (but poorly implemented). First systematic use of exceptions —23 types of exception & runtime errors. First use of pointers as a data type. First implementation of array slicing.
1964	SIMULA I	Nygaard & Dahl		First language designed explicitly for simulations
1966	FORTRAN IV (III was never released)		Explicit type declarations for variables	Passing subprograms as parameters to other subprograms

Year	Language	Developed by	Features	Firsts
1966	ALGOL-W	Wirth & Hoare	Developed at Stanford as teaching language; forerunner of Pascal	Use of case statement for multiple selection. Passing parameters by value-result.
1967	SIMULA 67	Nygaard & Dahl		First use of class (data) abstraction. First use of coroutines (subroutines that can return, then resume where they left off)
1968	ALGOL 68		Described using a new formal grammar, much more complex than BNF. Designers invented several new words to describe the language. This made the language hard to understand & learn, and limited its acceptance.	First user-defined data types. Implicit heap-dynamic arrays.
1970	B	Thompson	Based on Basic Combined Programming Language (BCPL). Not typed.	First high level language implemented on UNIX.
1970	FORTH	Moore	Stack based. User can extend language by adding new commands. Postfix notation. Base system relatively easy to port to new hardware. Used mostly in embedded systems, noted for fast compact code.	First resident development environment for Intel 8086 (1978) and Apple Macintosh (1984).
mid-1970s	Scheme	MIT	LISP descendant. Functions as 1st class entities. Static scoping.	
1971	BASIC	Kement & Kurtz	Designed at Dartmouth for ease of use by non-science majors.	First timesharing system. First language where programmer time was explicitly considered more valuable than computer time.
1971	SNOBOL	Farber, Griswold, & Polonsky	Designed for text processing. Powerful string pattern matching built in.	

Year	Language	Developed by	Features	Firsts
1971	Pascal	Wirth	Quickly became most widely used teaching language. Because it was designed as a teaching language, it lacked several key features (such as taking a variable-length array as a function parameter, random file I/O, or separate compilation). Borland's Turbo Pascal addressed these & was briefly popular as a production language.	
1972	C	Ritchie	Typed, but type checking not consistently enforced. Flexible but not very secure. Primary design goal was compact & fast-executing object code. Availability with UNIX made it widely adopted & it quickly became a widespread general development language. Standardized by ANSI in 1989 with updates in 1999.	
1972	Prolog	Roussel	Logic programming based on predicate calculus, using resolution proofs as inference technique	Formal logic as programming technique
1972	Smalltalk	Kay	Borrowed a few ideas from SIMULA 67 and ran with them. Syntax based on message passing between objects.	First fully OO language. First GUI as fundamental part of the language. More sophisticated memory management by JIT compiler
1975	CLU	Liskov	Introduce CLUsters, a fore-runner of objects. Classes had constructors & methods, but not inheritance. Exceptions considered part of normal program flow, so an efficient typesafe way to break out of methods if a function should return a value "except when" certain conditions apply.	Abstract data types, call by sharing, iterators, multiple return values, multiple assignment (A,B = X,Y)
1977	FORTRAN 77		Added string handling, logical loop control, if with optional else	
1977	awk	Aho, Kernighan, & Weinberger	Developed as report-generation and scripting language;later extensions made it more general purpose	
1977	sh	Bourne	Standard shell language for UNIX. Most commands are calls to system programs.	

Year	Language	Developed by	Features	Firsts
1979	Ada		History's largest design effort, instigated & pushed by DoD. Much more extensive exception handling than previous languages (reliability was major design goal). Delayed by difficulty writing compiler.	Packages for data encapsulation. Generic program units (e.g. sort routine that leaves data type unspecified, works for anything with < operator). Concurrency via rendezvous method.
1983	C With Classes	Stroustrup	Added function parameter type checking, first use of classes, public/private access control, constructor/destructor methods, friend classes, inline functions, default parameters, overloaded assignment operator. Goals were the ability to write OO code as with Smalltalk, with little performance penalty from C.	
1984	C++	Stroustrup	Virtual methods, method name & operator overloading, references as a type. As design goals included usability anywhere C could be used, few features from C were removed, even those considered unsafe.	
Early 1980s	Objective C	Cox & Love	C + objects using Smalltalk syntax. Licensed by Steve Jobs after he left Apple to write NeXT software, later used it for OS X. Standard language of iPhone development. Strictly a superset of C, so all of C's insecurities are retained.	
1985	Eiffel	Meyer	Based on principles of Eiffel programming method: design by contract, command-query separation, uniform-access principle, single-choice principle, option-operand separation	Design by contract drives essential language features
1986	Common LISP		Attempt to standardize LISP dialects. Dynamic scoping as an option. Records, arrays, complex numbers, strings.	
1985ish	ML	Robin Milner	Primarily functional but supports imperative programming; syntax closer to Java/C++.	
1986	Miranda	David Turner	Functional, based partly on ML partly on K&R C	

Year	Language	Developed by	Features	Firsts
1987	Perl	Wall	Originally conceived as a combination of sh and awk. Naming conventions include type. Dynamic length arrays that can be sparse (gaps between elements). Associative arrays (hashes).	
1989	C++ 2.0	Stroustrup	Multiple inheritance, abstract classes	
1990	Fortran 90		Added dynamic arrays, records, pointers, multiple selection, modules. Removed some features from earlier versions of FORTRAN. Dropped requirement for fixed line format. Also dropped ALL CAPS convention for identifiers	
1990	C++ 3.0	Stroustrup	Templates, exception handling	
1990	Java	Gosling et al	Designed for reliability in consumer devices. Quickly adopted for Web programming via applets.	
1992	Haskell	Hudak & Fasel	Purely functional; no variables, no assignments. Lazy evaluation (expressions not evaluated until they're needed)	List comprehensions
1992	Visual BASIC		Popular for writing GUI interfaces. Became standard interface language for MS Office. Supports OO programming.	
1993	Lua	Ierusalimschy, Celes, & Figruieriedo	Extensibility is primary design goal. Supports closures; functions are 1st class values. Noted for use in gaming industry.	
1994	Prolog++	Moss	Prolog, with objects	
1994	PHP	Lerdorf	Server-side scripting, originated as Personal Home Page Tools. Server runs PHP, which produces HTML as output.	
1995	Ada 95		More flexible type derivation allowed inheritance; dynamic dispatch implemented, allowing polymorphism. Improved data sharing between concurrent threads.	
1995	Javascript	Eich	Primary use is dynamically creating & modifying HTML documents (web pages)	
1995	Python	van Rossum	Strong dynamic typing. Includes lists, immutable lists (tuples), associative arrays (dictionaries). List comprehensions.	
1996	Ruby	Matsumoto	Pure OO language. All classes, predefined or user defined, can be subclassed.	
1997	Fortran 95		Few changes from F90.	

Year	Language	Developed by	Features	Firsts
1998	Caml	Cousineau et al	Based mostly on ML and Haskell	
1998	Alice (software)	Pausch etal.	Drag & drop educational tool to produce animations without writing code (though adding Java code is supported in more advanced sections). Not to be confused with the Alice programming language, a variant of ML	
2000	Delphi	Hejlsberg	Pascal, with objects	
2001	D	Bright	Started as re-engineering of C++. Allows imperative, OO, functional programming; concurrency support is good. Has some features to improve memory safety compared with C++.	
2002	Managed C++ (MC++)		Extends C++ to .NET. Adds properties, delegates, interfaces, reference type for garbage collected objects. Multiple inheritance not supported.	
2002	C#	Hejlsberg et al	Translated to Common Intermediate Language shared by all .NET languages; .NET uses a JIT compiler. Improved data type safety over C++ (but function pointers can still be dangerous). Has undergone rapid evolution since initial release, with additions of dynamic typing, anonymous types	Delegate type (OO, type-safe references to subprograms)
2004	Fortran 2003		Added support for objects, procedure pointers, interoperability with C	DO CONCURRENT
2005	Ada 2005		Interfaces (similar to Java), better synchronization	
2006	Ocaml		Caml, with objects	
2009	Go	Pike, Thompson, & Griesemer	Designed for fast compilation. Variable name precedes type names; type inference; functions can return multiple values. No inheritance or generics. Support for concurrency using goroutines	
2010	Fortran 2008		Blocks to define local scopes, co-arrays	
2010	F#	Syme et al	Functional, .NET platform. Also supports procedural & OO programming	
2010	Rust	Hoare et al.	Focus on memory safety & efficient but safe concurrency	

Year	Language	Developed by	Features	Firsts
2011	Scratch	Resnick	Designed to teach children programming by drag/drop code blocks & 'remixing' programs to produce animations w/ multimedia	
2014	Swift		Developed by Apple; uses Objective C's runtime library, so Swift & Objective C code can easily coexist	
2020	Raku	Wall	Perl 6 announced 2000; renamed to Raku in 2019 when backwards compatibility with Perl abandoned as design goal. Provided better OO features, more regular RE evaluation, improved consistency in features & implementation.	