Credo Reference

High-level language

including: problem-oriented language, third-generation language (3GL), fourth-generation language (4GL), application generation language

is designed to help a programmer express a computer program in a way that reflects the problem that is being solved, rather than the details of how the computer will produce the solution. These languages are often described as problem-oriented languages. The programmer will certainly be allowed to use long descriptive names for the variables, and to structure the program into subroutines or functions to help keep the logic of the solution clearly visible. Certain mathematical notation will be permitted, allowing calculations to be specified in the same way as in written mathematics. When the language is translated, the compiler (see page 176) or interpreter (see page 177) will take care of the details of the many machine-code instructions necessary to cause the computer to execute the program. Compare this with the definition of low-level language, page 243.

Languages that were developed at the time of third-generation computers (see page 288) are known as third-generation languages (3GL), and many are still in current use.

- Fourth-generation language (4GL) is used to describe languages aiming at end-users rather than specialist computer practitioners. A characteristic of these languages was the recognition that computing power was becoming more freely and more cheaply available, and that popular software reduced the time taken to develop users' programs. 4GLs are also known as application generation languages.

Some comparative information about some significant high-level languages appears in Table C7.1 (see page 239) and Table C7.2 (see page 241).

Table C7.1 Early programming languages

Language	Date	Derivation of name	Translation	Characteristics	Notes
Ada	1975-83	after Countess Lovelace	compiled	imperative	mainly used for large military systems, sponsored by US Dept of Defense

Generally no longer used for program development although some are still in use.

The date is that of initial development; most languages have been continuously improved since invention.

Translation indicates the most usual method, not necessarily the only one.

Characteristics indicates:

- whether the language is used by giving instructions on how to solve the problem (an algorithm, or conventional program) 'imperative';
- whether it is based on functions or procedures applied to sets of data functional';
- or whether its basic structure is that of an object-oriented programming language.

Language	Date	Derivation of name	Translation	Characteristics	Notes
ALGOL	1958-68	ALGOrithmic Language	compiled	imperative	originally for (paper) description of algorithms an influential language with several distinct versions (Algol-60, Algol-68)
ALGOL68	1968	ALGOrithmic Language	compiled	imperative	developed from Algol with more structure
APL	1957-68	A Programming Language	interpreted	functional	easier to write than to read; it uses symbols not always found on regular keyboards requires large memory. Operates on vectors
COBOL	1959-60	COmmon Business Oriented Language	compiled	imperative	easier to read than to write; large memory requirements, hence a mainframe language; highly structured data suited to business use
FORTH	late 1960s	pun on ‘Fourth’	mixed	unique, functional	very different from other languages: used in control and graphics applications
Pascal	1968-71	after Blaise Pascal	compiled	imperative	mainstream general-purpose structured language of the 1970s and 1980s
PL/1	1963-64	Programming Language 1	compiled	imperative	union of COBOL and FORTRAN, introduced new simple concepts but never gained popularity
POP-2	late 1960s	author: Dr R. J. Popplestone	interpreted	functional	artificial intelligence uses

Generally no longer used for program development although some are still in use.

The date is that of initial development; most languages have been continuously improved since invention.

Translation indicates the most usual method, not necessarily the only one.

Characteristics indicates:

- whether the language is used by giving instructions on how to solve the problem (an algorithm, or conventional program) ‘imperative’;
- whether it is based on functions or procedures applied to sets of data functional’;
- or whether its basic structure is that of an object-oriented programming language.

Language	Date	Derivation of name	Translation	Characteristics	Notes
RPG	1964	Report Program Generator	compiled	imperative	almost an applications package for business reports
Simula	1965	from simulation	compiled	object-oriented	Simulation. First object-oriented language
SNOBOL	1962-68	Stri/Vg Oriented SymBOlic Language	compiled	functional	string manipulation language

Generally no longer used for program development although some are still in use.

The date is that of initial development; most languages have been continuously improved since invention.

Translation indicates the most usual method, not necessarily the only one.

Characteristics indicates:

- whether the language is used by giving instructions on how to solve the problem (an algorithm, or conventional program) 'imperative';
- whether it is based on functions or procedures applied to sets of data functional';
- or whether its basic structure is that of an object-oriented programming language.

Table C7.2 Programming languages

Language	Date	Derivation of name	Translation	Characteristics	Notes
Ada 95	1995				superseded Ada, see Table C7.1, page 239
BASIC	1964	Beginners /Hlpurpose Symbolic/ nstruction Code	interpreted	imperative	easy to learn and (in modernised versions) the world's most frequently encountered programming language, especially in schools
C	1972	see notes	compiled	imperative	systems programming language, derived from 'B', which derived in turn from BCPL, which derived from CPL
C++	1979-83	C plus a bit more	compiled	object-oriented	object-oriented features added to C

Language	Date	Derivation of name	Translation	Characteristics	Notes
c#	2001		compiled	object-oriented	client-server distributed applications pronounced C sharp
Delphi	1995		compiled	object-oriented	Object Pascal extension to Pascal by Borland
FORTRAN	1954-57	FORmula TR/\ A/slation	compiled	imperative	mainstream scientific programming language; language still reflects punched-card input too early to be a structured programming language
FORTRAN 2003					there have been different generations
Java	1991-95	Its 'exotic, exciting, adrenalinepumping connotations' linked to Java coffee beans	compiled to intermediate code then compiled dynamically	object-oriented	platform-independent related to C++ for the internet use of applets, and bytecode for distributed applications across the internet
Lisp	1959	List Processing	interpreted	functional	artificial intelligence uses an influential language, in that many other languages have been derived from it
Logo	1966-68	Greek for 'thought'	interpreted	procedural, list processing	based on list-processing features, but noted for its 'turtle graphics' subset; popular in education as it is considered to teach thinking as well as programming
Prolog	1972	PROgramming in LOG ic	interpreted	declarative	artificial intelligence uses a very different language
Smalltalk	1972-80	to emphasis nature of language interface	interpreted	object-oriented	forerunner of most graphics interfaces
Visual Basic®	early 1990s	GUI-based extension of BASIC	interpreted or compiled	imperative	provides quick and easy means of creating interface applications for Microsoft® Windows®
VBA		Visual Basic® for Applications		imperative	later development of Visual Basic®

Language	Date	Derivation of name	Translation	Characteristics	Notes
Perl	1987	Originally Pearl	interpreted	object-oriented procedural	general purpose, used for text processing, web development, GUI development
JavaScript	1995		interpreted	object-oriented procedural	scripting language, superset of ECMAScript standard
Python	1994		interpreted	object-oriented procedural	

Typically used today for program development.

The date is that of initial development; most languages have been continuously improved since invention.

Translation indicates the most usual method, not necessarily the only one.

Characteristics indicates:

- whether the language is used by giving instructions on how to solve the problem (an algorithm, or conventional program) ‘imperative’;
- whether it is based on functions or procedures applied to sets of data ‘functional’;
- or whether its basic structure is that of an object-oriented programming language.

Prolog is rather an unusual language and is described on page 246. Java, and some other languages, may be encoded into an intermediate code that can then be interpreted on other systems.

Citation Information:

‘High-level language’ (2016) in Burdett, A. and Bowen, D., *BCS Glossary of Computing and ICT*. 14th edn. BCS, The Chartered Institute for IT. Available at: https://search.credoreference.com/articles/Qm9va0FydGljbGU6NDU2MDc3Nw==?aid=279777 (Accessed: 6 February 2024).