

First Generation Programming Languages

- Introduced in the 1940's.
- Sometimes referred as Binary Language, Machine Language, Very Low Level Language, Machine Code or Object Code.
- It is a language made up of entirely 1s and 0s.
- Programmers have to design their code by hand then transfer it to a computer by using a punch card, punch tape or flicking switches.
- It is the only language a computer is capable of understanding without using a translation program.
- Close to machines.
- Modern day programmers still occasionally use machine level code, especially when programming lower level functions of the system, such as drivers, interfaces with firmware and hardware devices.

First Generation Programming Languages

Advantages...

- Machine language makes fast and efficient use of the computer.
- It requires no translator to translate the code. It is directly understood by the computer.

Disadvantages...

- All operation codes have to be remembered.
- All memory addresses have to be remembered.
- It is hard to amend or find errors in a program written in the machine language.

Second Generation Programming Languages

- Introduced in the 1950's
- Sometimes referred as Assembly Language or Low Level Language.
- Programmes are in the form of Alphanumeric Symbols (or Mnemonic Codes) instead of 0's and 1's. These alphanumeric symbols can have maximum up to 5 letter combinations e.g. ADD for addition, SUB for subtraction, START LABEL etc. because of this feature it is also known as "Symbolic Programming Language".
- Close to machine
- Programmes are translated into machine language using Assemblers.
- Not portable
- Used in kernels and hardware driver, but more often find use in extremely intensive processing such as games, video editing, graphic manipulation/rendering.

Second Generation Programming Languages

Advantages...

- Assembly language is easier to understand and use as compared to machine language.
- It is easy to locate and correct errors.
- It is easily modified.

Disadvantages...

- Like machine language, it is also machine dependent/specific.
- Since it is machine dependent, the programmer also needs to understand the hardware.

Second Generation Programming Languages

Examples which are commonly in use today are:

- RISC(Reduced Instruction Set Computer)
- CISC(Complex Instruction Set Computer)
- x86 as that is what our embedded systems and desktop computers use.

Third Generation Programming Languages

- Introduced in the 1950's.
- Purpose of developing High-Level Languages was to enable people to write programs easily, in their own native language environment (English).
- These are Symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes.
- Close to humans.
- Compiler (which converts the language into machine code automatically) was developed
- First compiled high level programming language : in 1952 for the Mark 1 computer at the University of Manchester.
- In 1954, FORTRAN was invented at IBM by John Backus. It was the first widely used high level general purpose programming language to have a functional implementation.

Third Generation Programming Languages

Advantages...

- Easier to learn and understand than an assembly language as instructions (statements) that resemble human language or the standard notation of mathematics.
- Have less-rigid rules, forms, and syntaxes, so the potential for error is reduced.
- Are machine-independent programs therefore programs written in a high-level language do not have to be reprogrammed when a new computer is installed.
- Programmers do not have to learn a new language for each computer they program.

Disadvantages...

- Less efficient than assembler language programs and require a greater amount of computer time for translation into machine instructions.

Third Generation Programming Languages

Types of High Level Language :

1) Algebraic Formula-Type Processing :-

These languages are oriented towards the computational procedures for solving mathematical and statistical problems.

Examples include:

- BASIC (Beginners All Purpose Symbolic Instruction Code)
- FORTRAN (Formula Translation)
- PL/I (Programming Language, Version 1)
- ALGOL (Algorithmic Language)
- APL (A Programming Language)

2) Business Data Processing :-

These languages are best able to maintain data processing procedures and problems involved in handling files.

Some examples include:

- COBOL (Common Business Oriented Language)
- RPG (Report Program Generator)

3) String and List Processing :-

These are used for string manipulation, including search patterns and inserting and deleting characters.

Examples are:

- LISP (List Processing)
- Prolog (Program in Logic)

4) Programming Languages (Approx 50 types in wikipedia) :-

In OOP, the computer program is divided into objects.

Examples are:

- C++
- Java
- Objective C (develop software for Mac OS X and iOS) or C# (develop software for Windows (web,desktop,mobile)).

5) Visual Programming Language :-

These programming languages are designed for building Windows-based applications.

Examples are:

- Visual Basic
- Visual Java
- Visual C

Third Generation Programming Languages

Programming Paradigm :






- Many paradigms(Approx 20-30 paradigms) were developed between 1960's to late 1970's.
- A programming paradigm is a style or "way" of programming (Never say programming language paradigms, its wrong).
- There are a lot of programming paradigms, main are :
 1. Imperative (or procedural)
 2. Applicative (or functional)
 3. Declarative (or logical or Rule-based)
 4. Structured
 5. Object oriented

Third Generation Programming Languages

Programming Paradigm :

- Many paradigms(Approx 20-30 paradigms) were developed between 1960's to late 1970's.
- A programming paradigm is a style or "way" of programming (Never say programming language paradigms, its wrong).
- There are a lot of programming paradigms, main are :
 1. Imperative (or procedural)
 2. Applicative (or functional)
 3. Declarative (or logical or Rule-based)
 4. Structured
 5. Object oriented

TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Chief Developer, Company
	1957	Fortran	Team led by John W. Backus at IBM
	1958	ALGOL	ACM/GAMM (Association for Computing Machinery/German Association for Applied Mathematics & Mechanics)
	1958	LISP	John McCarthy
	1960	COBOL	The CODASYL Committee
	1964	BASIC	John George Kemeny and Thomas Eugene Kurtz at Dartmouth College

TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Cheif Developer, Company
	1968	Simula	Ole-Johan Dahl, Bjørn Myhrhaug, Kristen Nygaard at Norsk Regnesentral
	1970	Pascal	Niklaus Wirth, Kathleen Jensen
	1972	C	Dennis Ritchie
	1972	Smalltalk	Alan Kay, Adele Goldberg, Dan Ingalls, Xerox PARC
	1972	Prolog	Alain Colmerauer

TIMELINE OF PROGRAMMING LANGUAGES



Year

1974

Language

SQL

Cheif Developer, Company

IBM



1978

MATLAB

Cleve Moler at the University of New Mexico



1980

C with classes(c++)

Bjarne Stroustrup



1983

Objective-C

Brad Cox








1983






C++

Bjarne Stroustrup






TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Cheif Developer, Company
	1987	Perl	Larry Wall
	1991	HTML	Tim Berners-Lee
	1991	Python	Guido van Rossum
	1991	Visual Basic	Alan Cooper, sold to Microsoft
	1993	R	Robert Gentleman and Ross Ihaka

TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Chief Developer, Company
	1995	Java	James Gosling at Sun Microsystems (now Oracle)
	1995	PHP	Rasmus Lerdorf
	1995	Ruby	Yukihiro Matsumoto
	1995	JavaScript(LiveScript)	Brendan Eich at Netscape
	1996	VBScript	Microsoft

TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Cheif Developer, Company
	1998	Standard C++	ANSI/ISO Standard C++
	2000	C#	Anders Hejlsberg, Microsoft (ECMA)
	2001	Visual Basic .NET	Microsoft
	2003	Scala	Martin Odersky
	2006	Windows PowerShell	Microsoft

TIMELINE OF PROGRAMMING LANGUAGES

	Year	Language	Cheif Developer, Company
	2009	Go	Google
	2011	Kotlin	JetBrains
	2014	Hack	Facebook
	2014	Swift	Apple Inc.

Fourth Generation Programming Languages

- 1970s through the 1990s
- Also known as Very High Level Language or Non-Procedural Language.
- It is Application Specific.
- Close to natural language.
- Closer to the domain, Further from the machine.
- Fourth generation languages need approximately one tenth the number of statements that a high level languages needs to achieve the same results..
- Non-computer professionals can develop software.

Fourth Generation Programming Languages

Examples are :

1. Query languages (SQL)
2. Report Programmer Generators (RPG by IBM) : created for punched card machines.
3. Applications generators
4. MATLAB
5. Some minicomputer applications eg PowerBuilder, FOCUS, Infotrieve-4GL, Progress 4GL etc

Fifth Generation Programming Languages

- It is based on solving using constraints given to the program rather than using an algorithm written by a programmer.
- Introduced around 1990's.
- Very closely resembles human speech.
One could word a statement in several ways perhaps even misspelling some words or changing the order of the words and get the same result.
- Examples are : Prolog, Mercury, OPS5, AI etc
- These languages are also designed to make the computer "smarter".
- Mainly used in artificial intelligence research.
- Natural languages already available for microcomputers include Clout, Q&A, and Savvy Retriever (for use with databases) and HAL (Human Access Language).