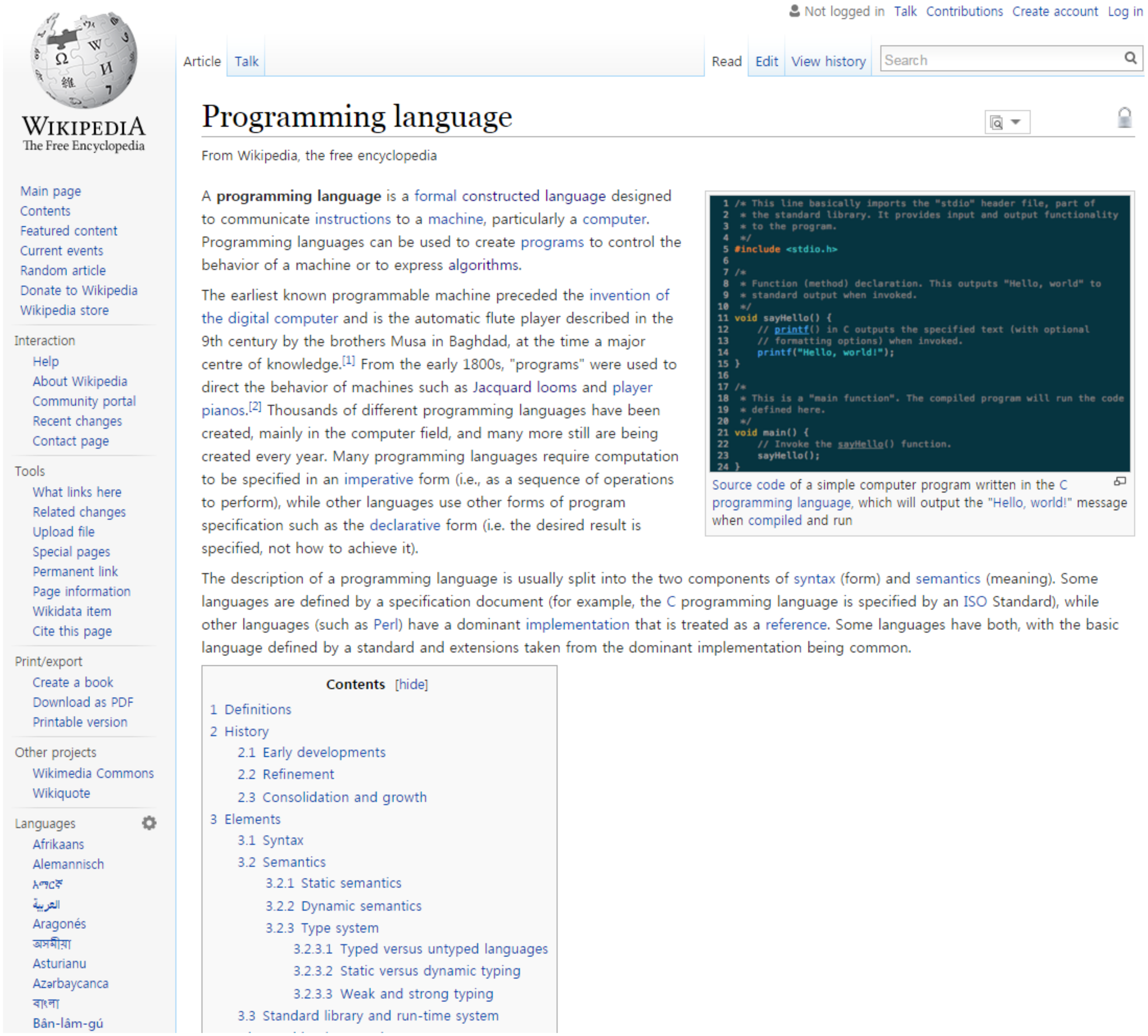
Evolution of Programming Languages

국민대학교 컴퓨터공학부

강 승 식



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Other projects
Wikimedia Commons
Wikiquote

Languages
Afrikaans
Alemannisch
አማርኛ
العربية
Aragonés
অসমীয়া
Asturianu
Azərbaycanca
বাংলা
Bân-lâm-gú

Programming language

From Wikipedia, the free encyclopedia

A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.

The earliest known programmable machine preceded the invention of the digital computer and is the automatic flute player described in the 9th century by the brothers Musa in Baghdad, at the time a major centre of knowledge.[1] From the early 1800s, "programs" were used to direct the behavior of machines such as Jacquard looms and player pianos.[2] Thousands of different programming languages have been created, mainly in the computer field, and many more still are being created every year. Many programming languages require computation to be specified in an imperative form (i.e., as a sequence of operations to perform), while other languages use other forms of program specification such as the declarative form (i.e. the desired result is specified, not how to achieve it).

```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world" to
 * standard output when invoked.
 */
void sayHello() {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
void main() {
    // Invoke the sayHello() function.
    sayHello();
}
```

Source code of a simple computer program written in the C programming language, which will output the "Hello, world!" message when compiled and run

The description of a programming language is usually split into the two components of syntax (form) and semantics (meaning). Some languages are defined by a specification document (for example, the C programming language is specified by an ISO Standard), while other languages (such as Perl) have a dominant implementation that is treated as a reference. Some languages have both, with the basic language defined by a standard and extensions taken from the dominant implementation being common.

Contents [hide]

What is a programming language?

- Formal constructed language
  - Designed to communicate instructions to a machine
  - Used to create programs to express algorithms
- Computation is specified in
  - Imperative form
    - as a sequence of operations to perform
  - Declarative form
    - The desired result is specified, not how to achieve it
- Description of P.L.
  - Syntax (form)
  - Semantics (meaning)

P.L. Implementation

- Interpretation
- Compilation
    - code generation → assembling → linking → loading and execution
- Cross-compiler 안드로이드 (다른 기계에서 사용 컴파일은 내컴퓨터에서)
- Source-to-source translation
- Compilation into bytecode – Smalltalk, Java
    - Interpreted by a virtual machine
    - JIT(Just-In-Time) or AOT(Ahead-Of-Time) compilation

First-generation programming language(1GL)

- Machine-level programming languages
- The instructions in 1GL are
  - made of binary numbers, represented by 1s and 0s.
  - instructions were entered through the front panel switches
- Advantage
  - run very fast and very efficiently, precisely
  - instructions are executed directly by the CPU
- Disadvantages
  - an error is not as easy to fix.

Second-generation programming language(2GL)

- Assembly languages
- The code can be read and written by a programmer.
    - It must be converted into a machine readable form, a process called assembly.
- Sometimes used in kernels and device drivers
    - but more often find use in extremely intensive processing such as games, video editing, graphic manipulation/rendering.
    - One method for creating such code is by allowing a compiler to generate a machine-optimized assembly language version of a particular function. This code is then hand-tuned.

Third-generation programming language(3GL)

- First high-level programming languages written in the 1950s.

  - Plankalkul, Konrad Zuse, 1943~1945
  - Short Code, John Mauchly, 1949
  - Autocode, Alick Gennie, early 1950s (Mark I)
  - FORTRAN, John Backus, 1954 (IBM)
  - FLOW-MATIC, Grace Hopper, 1955 (UNIVAC I)

1960s to 1970s: development of the major language paradigms

- APL
  - introduced array programming and influenced functional programming.
- ALGOL
  - refined both structured procedural programming and the discipline of language specification; the "Revised Report on the Algorithmic Language ALGOL 60" became a model for how later language specifications were written.
- Lisp
  - implemented in 1958, was the first dynamically typed functional programming language
- Simula
  - was the first language designed to support object-oriented programming in the 1960s;
  - in the mid-1970s, Smalltalk followed with the first "purely" object-oriented language.
- C
  - was developed between 1969 and 1973 as a system programming language for the Unix operating system, and remains popular.
- Prolog
  - designed in 1972, was the first logic programming language.
- ML
  - built a polymorphic type system on top of Lisp, pioneering statically typed functional programming languages, in 1978.

GOTO-less programming

- Edsger Dijkstra, 1968
  - GOTO statements should be eliminated from all "higher level" programming languages
  - Published in Communications of the ACM

Consolidation and growth

- C++
- Ada
- Pascal
- ML
- Modula-2
- Perl
- Java

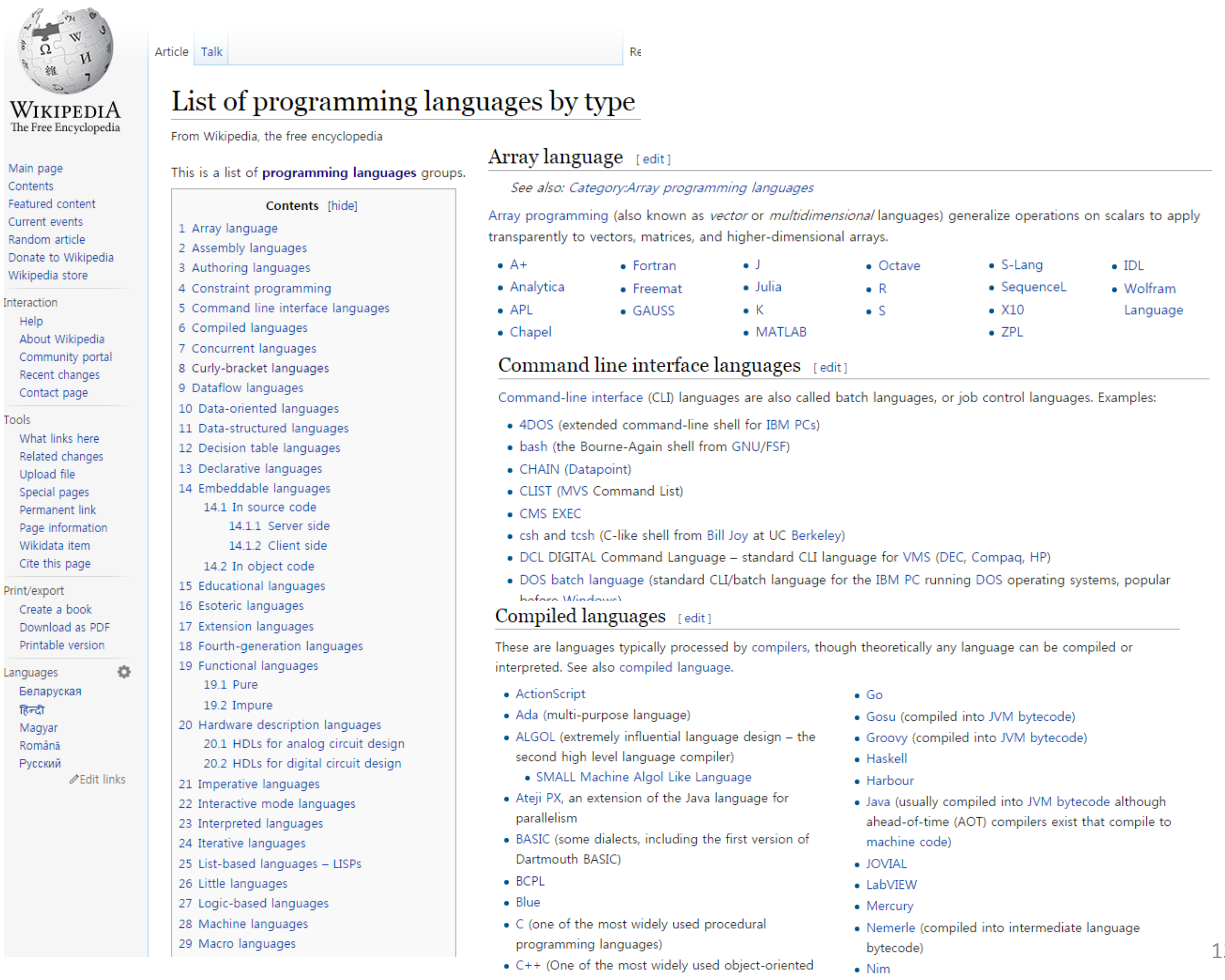
- Programming language evolution continues, in both industry and research. Current directions include security and reliability verification, new kinds of modularity (mixins, delegates, aspects), and database integration such as Microsoft's LINQ.
- Fourth-generation programming languages (4GL)
- Fifth generation programming languages (5GL)

Fourth-generation programming languages (4GL)

- higher level of abstraction
- making the language more programmer-friendly, powerful and versatile
- support for database management, report generation, mathematical optimization, GUI development, or web development.
- Some researchers state that 4GLs are a subset of domain-specific languages (DSLs).
  - Table-driven (codeless) programming: PowerBuilder
  - Report-generator programming languages
  - Forms generators: CASE tools
  - Data management 4GLs such as SAS, SPSS and Stata

The 1980s: fifth-generation languages (5GL)

- based on solving problems using constraints given to the program, rather than using an algorithm written by a programmer.
- constraint-based and logic programming languages and some declarative languages
- make the computer solve a given problem without the programmer.
- the programmer only needs to worry about what problems need to be solved and what conditions need to be met, without worrying about how to implement a routine or algorithm to solve them.
- Lisp, KL-ONE, Prolog, OPS5, Mercury
- However, given a set of constraints defining a particular problem, deriving an efficient algorithm to solve it is a very difficult problem in itself.



Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Беларуская
हिन्दी
Magyar
Română
Русский
Edit links

List of programming languages by type

From Wikipedia, the free encyclopedia

This is a list of **programming languages** groups.

Contents [hide]

Array language [edit]

See also: Category:Array programming languages

Array programming (also known as *vector* or *multidimensional* languages) generalize operations on scalars to apply transparently to vectors, matrices, and higher-dimensional arrays.

- A+
- Analytica
- APL
- Chapel
- Fortran
- Freemat
- GAUSS
- J
- Julia
- K
- MATLAB
- Octave
- R
- S
- S-Lang
- SequenceL
- X10
- ZPL
- IDL
- Wolfram Language

Command line interface languages [edit]

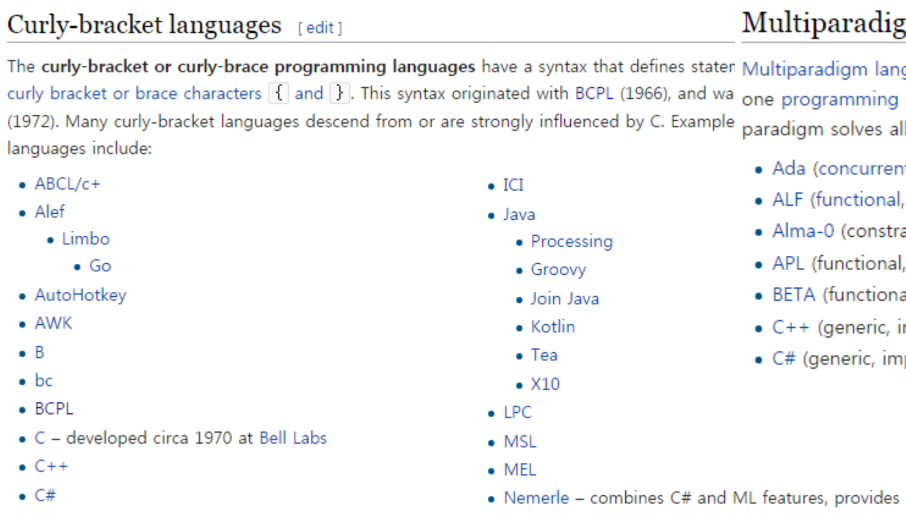
Command-line interface (CLI) languages are also called batch languages, or job control languages. Examples:

- 4DOS (extended command-line shell for IBM PCs)
- bash (the Bourne-Again shell from GNU/FSF)
- CHAIN (Datapoint)
- CLIST (MVS Command List)
- CMS EXEC
- csh and tcsh (C-like shell from Bill Joy at UC Berkeley)
- DCL DIGITAL Command Language – standard CLI language for VMS (DEC, Compaq, HP)
- DOS batch language (standard CLI/batch language for the IBM PC running DOS operating systems, popular before Windows)

Compiled languages [edit]

These are languages typically processed by compilers, though theoretically any language can be compiled or interpreted. See also compiled language.

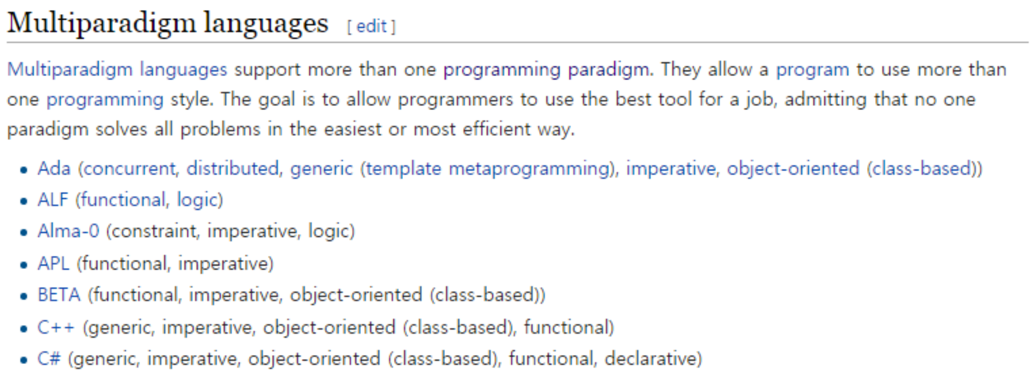
- ActionScript
- Ada (multi-purpose language)
- ALGOL (extremely influential language design – the second high level language compiler)
  - SMALL Machine Algol Like Language
- Ateji PX, an extension of the Java language for parallelism
- BASIC (some dialects, including the first version of Dartmouth BASIC)
- BCPL
- Blue
- C (one of the most widely used procedural programming languages)
- C++ (One of the most widely used object-oriented
- Go
- Gosu (compiled into JVM bytecode)
- Groovy (compiled into JVM bytecode)
- Haskell
- Harbour
- Java (usually compiled into JVM bytecode although ahead-of-time (AOT) compilers exist that compile to machine code)
- JOVIAL
- LabVIEW
- Mercury
- Nemerle (compiled into intermediate language bytecode)
- Nim



Curly-bracket languages [edit]

The **curly-bracket or curly-brace programming languages** have a syntax that defines stater curly bracket or brace characters { and }. This syntax originated with BCPL (1966), and wa (1972). Many curly-bracket languages descend from or are strongly influenced by C. Example languages include:

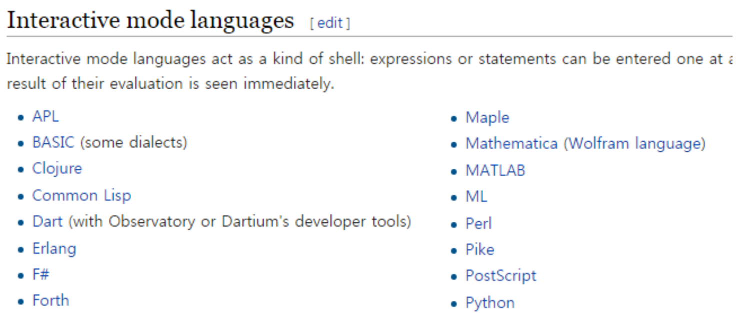
- ABCL/c+
- Alef
  - Limbo
    - Go
- AutoHotkey
- AWK
- B
- bc
- BCPL
- C – developed circa 1970 at Bell Labs
- C++
- C#
- ICI
- Java
  - Processing
  - Groovy
  - Join Java
  - Kotlin
  - Tea
  - X10
- LPC
- MSL
- MEL
- Nemerle – combines C# and ML features, provides



Multiparadigm languages [edit]

Multiparadigm languages support more than one programming paradigm. They allow a program to use more than one programming style. The goal is to allow programmers to use the best tool for a job, admitting that no one paradigm solves all problems in the easiest or most efficient way.

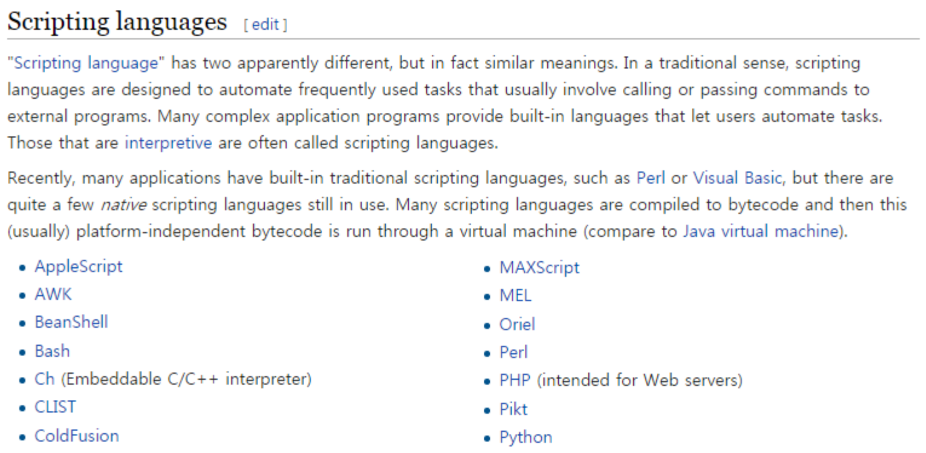
- Ada (concurrent, distributed, generic (template metaprogramming), imperative, object-oriented (class-based))
- ALF (functional, logic)
- Alma-0 (constraint, imperative, logic)
- APL (functional, imperative)
- BETA (functional, imperative, object-oriented (class-based))
- C++ (generic, imperative, object-oriented (class-based), functional)
- C# (generic, imperative, object-oriented (class-based), functional, declarative)



Interactive mode languages [edit]

Interactive mode languages act as a kind of shell: expressions or statements can be entered one at a result of their evaluation is seen immediately.

- APL
- BASIC (some dialects)
- Clojure
- Common Lisp
- Dart (with Observatory or Dartium's developer tools)
- Erlang
- F#
- Forth
- Maple
- Mathematica (Wolfram language)
- MATLAB
- ML
- Perl
- Pike
- PostScript
- Python



Scripting languages [edit]

"Scripting language" has two apparently different, but in fact similar meanings. In a traditional sense, scripting languages are designed to automate frequently used tasks that usually involve calling or passing commands to external programs. Many complex application programs provide built-in languages that let users automate tasks. Those that are interpretive are often called scripting languages.

Recently, many applications have built-in traditional scripting languages, such as Perl or Visual Basic, but there are quite a few *native* scripting languages still in use. Many scripting languages are compiled to bytecode and then this (usually) platform-independent bytecode is run through a virtual machine (compare to Java virtual machine).

- AppleScript
- AWK
- BeanShell
- Bash
- Ch (Embeddable C/C++ interpreter)
- CLIST
- ColdFusion
- MAXScript
- MEL
- Oriel
- Perl
- PHP (intended for Web servers)
- Pikt
- Python