a high-level set of instructions to control the behavior of a computer. Programming languages, like human languages, are defined through the use of syntactic and semantic rules to determine structure and meaning, respectively. Although at a high level compared with the electronics of computers, programming languages are still difficult to learn, and most people cannot write a computer program. (Twidale & Nichols, 2009; Frantzeskou, MacDonell, & Stamatatos, 2010)

Syntax, grammar, and symbols or words used to give instructions to a computer.

Development of Low-Level Languages

All computers operate by following machine language programs, a long sequence of instructions called machine code that is addressed to the hardware of the computer and is written in binary notation (see numeration), which uses only the digits 1 and 0. First-generation languages, called machine languages, required the writing of long strings of binary numbers to represent such operations as “add,” “subtract,” “and compare.” Later improvements allowed octal, decimal, or hexadecimal representation of the binary strings.

Because writing programs in machine language is impractical (it is tedious and error prone), symbolic, or assembly, languages—second-generation languages—were introduced in the early 1950s. They use simple mnemonics such as A for “add” or M for “multiply,” which are translated into machine language by a computer program called an assembler. The assembler then turns that program into a machine language program. An extension of such a language is the macro instruction, a mnemonic (such as “READ”) for which the assembler substitutes a series of simpler mnemonics. The resulting machine language programs, however, are specific to one type of computer and will usually not run on a computer with a different type of central processing unit (CPU).

Evolution of High-Level Languages

The lack of portability between different computers led to the development of high-level languages—so called because they permitted a programmer to ignore many low-level details of the computer's hardware. Further, it was recognized that the closer the syntax, rules, and mnemonics of the programming language could be to “natural language” the less likely it became that the programmer would inadvertently introduce errors (called “bugs”) into the program. Hence, in the mid-1950s a third generation of languages came into use. These algorithmic, or procedural, languages are designed for solving a particular type of problem. Unlike machine or symbolic languages, they vary little between computers. They must be translated into machine code by a program called a compiler or interpreter.

Early computers were used almost exclusively by scientists, and the first high-level language, Fortran [Formula translation], was developed (1953–57) for scientific and engineering applications by John Backus at the IBM Corp. A program that handled recursive algorithms better, LISP [LIST Processing], was developed by John McCarthy at the Massachusetts Institute of Technology in the early 1950s; implemented in 1959, it has become the standard language for the artificial intelligence community.

https://search.credoreference.com/content/topic/programming_language
COBOL [COmmon Business Oriented Language], the first language intended for commercial applications, is still widely used; it was developed by a committee of computer manufacturers and users under the leadership of Grace Hopper, a U.S. Navy programmer, in 1959. ALGOL [ALGOirthmic Language], developed in Europe about 1958, is used primarily in mathematics and science, as is APL [A Programming Language], published in the United States in 1962 by Kenneth Iverson. PL/1 [Programming Language 1], developed in the late 1960s by the IBM Corp., and ADA [for Ada Augusta, countess of Lovelace, biographer of Charles Babbage], developed in 1981 by the U.S. Dept. of Defense, are designed for both business and scientific use.

BASIC [Beginner's All-purpose Symbolic Instruction Code] was developed by two Dartmouth College professors, John Kemeny and Thomas Kurtz, as a teaching tool for undergraduates (1966); it subsequently became the primary language of the personal computer revolution. In 1971, Swiss professor Nicholas Wirth developed a more structured language for teaching that he named Pascal (for French mathematician Blaise Pascal, who built the first successful mechanical calculator). Modula 2, a Pascal-like language for commercial and mathematical applications, was introduced by Wirth in 1982. Ten years before that, to implement the UNIX operating system, Dennis Ritchie of Bell Laboratories produced a language that he called C; along with its extensions, called C++, developed by Bjarne Stroustrup of Bell Laboratories, it has perhaps become the most widely used general-purpose language among professional programmers because of its ability to deal with the rigors of object-oriented programming. Java is an object-oriented language similar to C++ but simplified to eliminate features that are prone to programming errors. Java was developed specifically as a network-oriented language, for writing programs that can be safely downloaded through the Internet and immediately run without fear of computer viruses. Using small Java programs called applets, World Wide Web pages can be developed that include a full range of multimedia functions.

Fourth-generation languages are nonprocedural—they specify what is to be accomplished without describing how. The first one, FORTH, developed in 1970 by American astronomer Charles Moore, is used in scientific and industrial control applications. Most fourth-generation languages are written for specific purposes. Fifth-generation languages, which are still in their infancy, are an outgrowth of artificial intelligence research. PROLOG [PROgramming LOGic], developed by French computer scientist Alain Colmerauer and logician Philippe Roussel in the early 1970s, is useful for programming logical processes and making deductions automatically.

Many other languages have been designed to meet specialized needs. GPSS [General Purpose System Simulator] is used for modeling physical and environmental events, and SNOBOL [String-Oriented Symbolic Language] is designed for pattern matching and list processing. LOGO, a version of LISP, was developed in the 1960s to help children learn about computers. PILOT [Programmed Instruction Learning, Or Testing] is used in writing instructional software, and Occam is a nonsequential language that optimizes the execution of a program's instructions in parallel-processing systems.

There are also procedural languages that operate solely within a larger program to customize it to a user's particular needs. These include the programming languages of several database and statistical programs, the scripting languages of communications programs, and the macro languages of word-processing programs.

**Compilers and Interpreters**

Once the program is written and has had any errors repaired (a process called debugging), it may be
executed in one of two ways, depending on the language. With some languages, such as C or Pascal, the program is turned into a separate machine language program by a compiler, which functions much as an assembler does. Other languages, such as LISP, do not have compilers but use an interpreter to read and interpret the program a line at a time and convert it into machine code. A few languages, such as BASIC, have both compilers and interpreters. Source code, the form in which a program is written in a high-level language, can easily be transferred from one type of computer to another, and a compiler or interpreter specific to the machine configuration can convert the source code to object, or machine, code.

Bibliography

T. W. Pratt; M. V. Zelkowitz, Programming Languages: Design and Implementation (3d ed. 1996);
C. Ghezzi; M. Jazayem, Programming Language Concepts (3d ed. 1997);

APA

Chicago

Harvard

MLA

APA

Chicago

Harvard

MLA