Principles of Programming Languages

A **Programming Language** is an artificial (made-up) language that can be used to control the behavior of a machine, particularly a computer. Programming languages allow us to translate the 1s and 0s into something that humans can understand and write.
A set of commands, instructions, and symbols that humans can manipulate in order to communicate with computers. It's a high-level set of instructions to control the behavior of a computer. Although high-level compared with the electronics of computers programming languages are still difficult to learn and most people cannot write a computer program.

What makes a language successful?

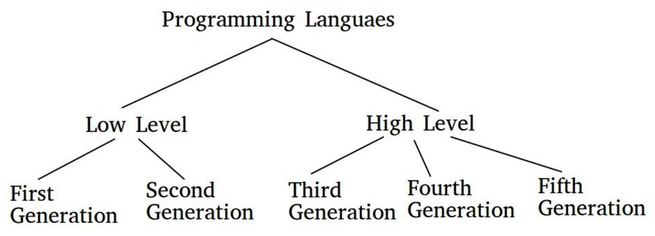
The language which is easy to learn, easy to express things, easy use once fluent and "powerful". Besides, easy to implement, possible to compile to very good (fast/small) code.
A programming language's features include orthogonality or simplicity, available control structures, data types and data structures, syntax design, support for abstraction, expressiveness, type equivalence, and strong versus weak type checking, exception handling, and restricted aliasing.

The following key characteristics:

- Simplicity and readability
- Clarity about binding
- Reliability
- Support
- Abstraction
- Orthogonality
- Efficient implementation

Read More:

1. Generations of Programming languages with examples.
2. Low level and High level languages with examples.



Structured Programming

Kenneth Leroy Busbee and Dave Braunschweig

Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines in contrast to using simple tests and jumps such as the go to statement, which can lead to "**spaghetti code**" that is potentially difficult to follow and maintain.

One of the most important concepts of programming is the ability to control a program so that different lines of code are executed or that some lines of code are executed many times. The mechanisms that allow us to control the flow of execution are called **control structures**. Flowcharting is a method of documenting (charting) the flow (or paths) that a program would execute. There are three main categories of control structures:

- **Sequence** – Very boring. Simply do one instruction then the next and the next. Just do them in a given sequence or in the order listed. Most lines of code are this.
- **Selection** – This is where you select or choose between two or more flows. The choice is decided by asking some sort of question. The answer determines the path (or which lines of code) will be executed.
- **Iteration** – Also known as repetition, it allows some code (one to many lines) to be executed (or repeated) several times. The code might not be executed at all (repeat it zero times), executed a fixed number of times or executed indefinitely until some condition has been met. Also known as looping because the flowcharting shows the flow looping back to repeat the task.

A fourth category describes unstructured code.

- **Branching** – An uncontrolled structure that allows the flow of execution to jump to a different part of the program. This category is rarely used in modular structured programming.

All high-level programming languages have control structures. All languages have the first three categories of control structures (sequence, selection, and iteration). Most have the if then else structure (which belongs to the selection category) and the while structure (which belongs to the iteration category). After these two basic structures, there are usually language variations.

The concept of **structured programming** started in the late 1960's with an article by Edsger Dijkstra. He proposed a "go to less" method of planning programming logic that eliminated the need for the branching category of control structures. The topic was debated for about 20 years. But ultimately – "By the end of the 20th century nearly all computer scientists were convinced that it is useful to learn and apply the concepts of structured programming."

Key Terms

branching	An uncontrolled structure that allows the flow of execution to jump to a different part of the program.
control structures	Mechanisms that allow us to control the flow of execution within a program.
iteration	A control structure that allows some lines of code to be executed many times.
selection	A control structure where the program chooses between two or more options.
sequence	A control structure where the program executes the items in the order listed.
spaghetti code	A pejorative phrase for unstructured and difficult to maintain source code.
structured programming	A method of planning programs that avoids the branching category of control structures.

References

- cnx.org: Programming Fundamentals – A Modular Structured Approach using C++

Discussion:

1. Advantages and Disadvantages of structured programming?
2. Basic concepts of Structured Programming