History and Evolution of Programming Languages

From Machine Code to Present Day

The First Generation

- 1945: ENIAC Machine Code
- Level of processor
- Processor executable binary commands
- Processor understandable binary data
- Programming in 1s and 0s
- Pure machine language

The Second Generation

- 1952: IBM 701 Assembler
- Level of processor architecture
- Humanized processor command symbols
- Processor data in human readable form
- No more machine language
- No more processor executable and understandable
- Must be translated to machine code to execute

The Third Generation

- 1957: FORTRAN
- High (human) level
- Commands and data are highly processor independent
- Commands and data are abstract and complex
- Language of mathematicians and engineers
- No more generations, but programming paradigms

Procedural Programming

- 1958: FORTRAN (1957)
- Code organized in callable procedures
- Reusable code

Structured Programming

- 1960: ALGOL
- Code organized in blocks inside procedures
- Rich control structures of code blocks
- No jumps in code
- Simpler execution flow
- Complex structured data

Modular Programming

- 1974: COBOL (1959)
- Module as set of procedures
- Single aspect of functionality
- Independent
- Interchangeable
- Reusable modules of functionality

Object Oriented Programming

- 1967: Simula
- Object as structured data with code
- Encapsulation
- Inheritance
- Polymorphism
- Description of real world objects and GUI

Imperative Programming

- Machine code
- Assembler
- Procedural programming
- Structured programming
- Modular programming
- Object oriented programming
- Statements (imperatives) changing state (mutating data)
- Control structures
- Abstraction of processor commands

Functional Programming

- 1958: LISP
- Immutable data
- Functions transforming data into new values
- No state
- The same values for the same data
- No side effects
- Thread-safe
- Parallel execution of functions with no data dependency
- Multithreaded applications