PROGRAMMING LANGUAGES

ENGLISH PRESENTATION

27 NOVEMBRE 2023

GROUP 3
LSI – INFO -3

SUMMARY

INTRODUCTION

Hello everyone. Today, we are going to talk to you about a topic that concerns all computer scientists, but also anyone who is interested in how machines execute the instructions that we give them: programming languages. What is a programming language? How did they appear and how did they evolve over time? What are the different types and generations of programming languages? What are the advantages and disadvantages of each language? What are the most used and popular languages today? These are some of the questions that we will try to answer in this presentation.

You probably know that programming languages are ways of communicating with machines to give them instructions to execute. But do you know that the first programming languages date back to the 19th century, with inventions like the Jacquard loom or the analytical engine of Babbage? Do you know

that programming languages diversified and became more complex in the 20th century, to adapt to the needs and capabilities of the machines? Do you know that programming languages are classified into four generations, according to their level of abstraction and their proximity to natural language? Do you know that programming languages are based on different paradigms, such as imperative programming, functional programming, object-oriented programming, concurrent programming, etc.? Do you know that programming languages have impacts on the productivity, reliability, portability, performance, etc., of the programs?

Our goal in this presentation is to make you discover or rediscover the history and evolution of programming languages, to present you the main characteristics and examples of each type and generation of programming languages, and to make you think about the challenges and issues that programming languages pose in the current world.

we hope this presentation will interest you and teach you new things. we thank you for your attention and we invite you to ask me questions at the end of our presentation.

HYSTORY AND EVOLUTION OF PROGRAMMING LANGUAGES

Programming languages are ways of communicating with machines to give them instructions to execute. They started to appear in the 19th century, with inventions like the Jacquard loom, which used punched cards to control the weaving of patterns, or the analytical engine of Charles Babbage, which was designed to perform mathematical calculations from programs written by Ada Lovelace, considered as the first programmer in history[1].

In the 20th century, with the development of electronics and computing, programming languages diversified and became more complex, to adapt to the needs and capabilities of the machines. We can generally distinguish four generations of programming languages:

- The **first-generation languages**, or machine languages, are the most basic and the closest to the functioning of the machines. They use binary codes, composed of 0 and 1, to represent the instructions and the data. They are hard to read and write for humans, and depend on the type of machine used. Examples: binary code, hexadecimal code.

- The **second-generation languages**, or assembly languages, are more abstract languages than machine languages, but still related to the hardware. They use mnemonic symbols, like letters or

numbers, to represent the instructions and the data. They are easier to read and write than machine languages, but require a translator, called an assembler, to be converted into machine language. Examples: Assembler, COBOL, FORTRAN.

- The **third-generation languages**, or high-level languages, are more distant from the hardware and closer to the natural language. They use keywords, control structures, variables, functions, etc., to express the instructions and the data. They are easier to read and write than assembly languages, but require a compiler or an interpreter to be converted into machine language. Examples: C, Java, Python, Ruby.

- The **fourth-generation languages**, or very high-level languages, are even more abstract and more oriented towards the problem to be solved. They use expressions, queries, rules, etc., to specify the instructions and the data. They are easier to read and write than high-level languages, but require a specific execution environment to be executed. Examples: SQL, Prolog, MATLAB, R.

The evolution of programming languages continues today, with the emergence of new paradigms, such as functional programming, object-oriented programming, concurrent programming, etc., which aim to improve the productivity, reliability, portability, performance, etc., of the programs. Examples: Haskell, Scala, Erlang, Go.

CARACTERISTICS AND CLASSIFICATION OF PROGRAMMING LANGUAGES

Programming languages are essential tools for software development. They allow developers to communicate with computers and provide instructions to perform specific tasks. These languages are numerous and diverse, each with its own characteristics and classifications. This presentation aims to explore these different aspects according to several criteria.

Abstraction Level:

- Low-level languages: They are close to the machine language and offer little abstraction. Assembler is a typical example of a low-level language.
- High-level languages: They are closer to human language and offer greater abstraction. Languages like Python, Java, and C# are considered high-level.

Programming Paradigm:

- Imperative: Based on a sequence of instructions that change the program's state. C and Python can be used in an imperative manner.
- Functional: Emphasizes functions and data processing. Haskell and Erlang are examples of functional languages.
- Object-oriented: Centers design around objects and their interactions. Java and C++ are object-oriented languages.
- Logic: Uses formal logic to solve problems. Prolog is a logic programming language.

Syntax:

- Prefixed: Operators precede operands. Lisp uses prefixed syntax.
- Infix: Operators are placed between operands. Most languages like C, Java, and Python use infix syntax.
- Postfixed: Operators follow operands. Forth is an example of a language using postfixed syntax.

Typing:

- Static: Types are checked before program execution. C and Java use static typing.
- Dynamic: Types are checked during program execution. Python and JavaScript are dynamically typed languages.
- Strong: Type conversions are restricted and must be explicit. Haskell is known for its strong typing.
- Weak: Type conversions are more liberal and can be implicit. C can be considered weakly typed.

Portability:

- Compiled: The source code is transformed into platform-specific machine language. C and Go are compiled languages.
- Interpreted: The source code is executed line by line by an interpreter. Python and Ruby are interpreted languages.
- Hybrid: Combine aspects of compiled and interpreted languages. Java uses a hybrid approach with the Java Virtual Machine (JVM).

- Performance:

- Execution time: Compiled languages tend to be faster as the code is optimized for the target platform.
- Memory consumption: Languages that offer more abstraction and high-level features may consume more memory.

In conclusion, programming languages are diverse and designed with different goals in mind. The choice of a language will often depend on the problem to be solved, the developer's preferences, and the constraints of the project.

USE OF PROGRAMMING LANGUAGES

Programming languages are used to develop all the applications that we use in our computers, telephones and almost all of our electronic equipment.... It is through programming that all technological developments in the world occur. Since different programming languages have the same syntax, this facilitates group work, which further accelerates technological development.

The disadvantage of programming languages is that they evolve regularly and programmers must be learning all the time to avoid falling behind.

FUTURE OF PROGRAMMING LANGUAGES

Programming languages continually evolve to adapt to the changing needs of the IT industry. For instance, Python, with its flexibility and specialized libraries, remains at the forefront for applications related to AI and machine learning. This evolution demonstrates the ability of languages to embrace new technologies and address emerging challenges.

AI and Its Influence:

The advent of AI has a significant impact on the programming landscape. Programming languages used in the AI domain, such as Python and R, have become essential. Programmers need to acquire specific AI-related skills to remain competitive, highlighting the necessity for constant adaptation to technological advancements.

Perspectives for Programming Languages:

Programming languages will not become obsolete but rather be tailored to new challenges. Specialized languages might emerge to meet the specific needs of AI, cybersecurity, or other growing fields. Versatile languages like Python will likely continue to dominate due to their adaptability.

Impact on the Programming Profession:

Programmers in the era of AI must broaden their skill set to remain relevant. Emphasizing machine learning, data analysis, and other AI-related skills becomes crucial. However, the demand for skilled programmers will remain strong, evolving toward more specialized roles in the development of AI and other emerging technologies.

In conclusion, programming languages will continue to play a crucial role in the computing field, adapting to current challenges, including the rise of AI. Programmers, by embracing these changes, will have the opportunity to stay at the forefront of technological innovation and contribute significantly to a constantly evolving digital future.