IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

The Evolution of Computer Programming Languages

Prathamesh Pratap Rane
Institute of Distance and Open Learning, Mumbai, Maharashtra, India

Abstract: *This research paper aims to study the progression and evolution of Computer Languages. How have evolved over time and also examines the evidence that newer programming languages are powerful than older ones, take the best of what older languages have to offer, so that older languages weaknesses can be eliminated and add the new strengths. The paper studies the progression of programming languages by examining the features of the most prominent languages and how these features have been adopted by newer languages. Finally, the paper presents a list of languages that may become the most popular programming languages.*

Keywords: Programming language progression, classifications, future of languages

I. INTRODUCTION

A language that used to set instructions for computer is known as programming language. It is a set of instructions that tells the computer what to do. Programming languages are used to create software applications, websites, and other computer programs.

The first programming languages were developed in the 1940s and 1950s. They were very simple and inefficient, but they were a necessary step in the evolution of computer programming.

In the 1940s, programmers used machine-specific assembly languages to write code. These languages were difficult to use and had many errors.The first modern programming language likeFORTRAN, was created in 1955. It was followed by COBOL, LISP, and ALGOL. These four languages are the foundation of all modern programming languages.

There are also many applications that are still written in these languages and are still sustain from1964, BASIC short for beginner's all-purpose Symbolic Instruction Codewas created in 1964, and C in 1969. C is a robust that is used to write operating systems and its functions, like C is a programming language that is widely used in the development of operating systems. UNIX and Linux are two operating systems that have a lot of code written in C

II. EVOLUTION OF COMPUTER LANGUAGES

The progression of programming languages can be divided into four main phases:

- The machine code era (1940s-1950s)
- The assembly language era (1950s-1960s)
- The high-level language era (1960s-present)
- The modern era (1990s-present)

2.1 THE MACHINE CODE AGE

The era of machine code marked the beginning of programming languages' development. The language that the hardware of the computer can understand directly is called machine code. It is a very basic language that is challenging to write and debug.

The first programming-languages was created using machine code. These languages were very simple and not so powerful, but they were a necessary step in the progression of computer programming.

2.2 THE ASSEMBLY LANGUAGE AGE

The assembly language era followed the machine code period. Assembly language is a higher-level language than machine code, but it is still a lower level language. Assembly language uses to represent machine code instructions.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

This makes assembly language code more user-friendly and efficient. High-level languages are becoming more popular, while assembly language is becoming less so.

2.3 THE HIGH-LEVEL LANGUAGE AGE

From 1960s the high-level language periods began. High level languages are easy to write and debug compare to machine code or assembly language. High level programming languages are more easily movable than assembly, meaning that they can be used without rewritten. This is because high level languages are not specific to any architecture, while assembly language is specific.

Some of the most famous high level languages include:

- FORTRAN short forFormula Translation
- COBOL which is extended form Common Business Oriented Language
- BASIC short for Beginners All-purpose Symbolic Instruction Code
- PASCAL
- C
- C++
- JAVA
- PYTHON
- JavaScript

2.4 THE MODERN AGE

The modern age of programming languages started from 1990s. This era is marked by the creation of new programming approaches, such as object-oriented and functional programming.

These languages are used to develop a wide range of applications, from web apps to mobile apps to desktop apps to scientific computing apps.

The field of programming languages is rapidly changing, and it's thrilling to imagine what new possibilities the future holds.

Some of the most famous programming languages in the modern ageare:

- JAVA
- PYTHON
- JAVASCRIPT
- C#
- RUBY
- SWIFT
- KOTLIN
- GO
- RUST
- JULIA

TABLE I. Programming Languages with its birth year.

NAME OF PROGRAMMING LANGUAGE	YEAR
REGIONAL ASSEMBLY LANGUAGE	1951
AUTO CODE	1952
IPL (FORERUNNER TO LISP)	1954
FLOW-MATIC (LED TO COBOL)	1955
FORTRAN (FIRST COMPILER)	1957
COMTRAN (PRECURSOR TO COBOL)	1957
LISP	1958

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

ALGOL 58	1958
FACT (FORERUNNER TO COBOL)	1959
COBOL	1959
RPG	1959
ALGOL 60	1960
APL	1962
SIMULA	1962
SNOBOL	1962
CPL (FORERUNNER TO C)	1963
SPEAKEASY	1964
BASIC	1964
PL/I	1964
JOSS	1966
MUMPS	1966
BCPL (FORERUNNER TO C)	1967
LOGO (AN EDUCATIONAL LANGUAGE THAT LATER INFLUENCED SMALLTALK AND SCRATCH).	1967
BCPL (FORERUNNER TO B)	1967
LOGO	1968
B (FORERUNNER TO C)	1969
PASCAL	1970
FORTH	1970
C	1972
SMALLTALK	1972
PROLOG	1972
ML	1973
SCHEME	1975
SQL (A QUERY LANGUAGE, LATER EXTENDED)	1978
C++ (AS C WITH CLASSES, RENAMED IN 1983)	1980
ADA	1983
COMMON LISP	1984
MATLAB	1984
DBASE III, DBASE III PLUS (CLIPPER AND FOXPRO AS FOXBASE)	1984
EIFFEL	1985
OBJECTIVE-C	1986
LABVIEW (VISUAL PROGRAMMING LANGUAGE)	1986
ERLANG	1986
PERL	1987
PIC (MARKUP LANGUAGE)	1988
TCL	1988
WOLFRAM LANGUAGE (AS PART OF MATHEMATICA, ONLY GOT A SEPARATE NAME IN JUNE 2013)	1988
FL (BACKUS)	1989

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

HASKELL	1990
PYTHON	1990
VISUAL BASIC	1991
LUA	1993
R	1993
CLOS (PART OF ANSI COMMON LISP)	1994
RUBY	1995
ADA 95	1995
JAVA	1995
DELPHI (OBJECT PASCAL)	1995
JAVASCRIPT	1995
PHP	1995
OCAML	1996
REBOL	1997
ACTIONSCRIPT	2000
C#	2001
D	2001
SCRATCH	2002
GROOVY	2003
SCALA	2003
F#	2005
HOLYC	2005
POWERSHELL	2006
CLOJURE	2007
NIM	2008
GO	2009
DART	2011
KOTLIN	2011
JULIA	2012
TYPESCRIPT	2012
ELM	2012
ELIXIR	2012
SWIFT	2014
RUST	2015
RAKU	2015
BOSQUE	2019
MICROSOFT POWER FX	2021

TABLE II. NUMBER OF NEW PROGRAMMING LANGUAGES BORN, INYEAR-BANDS

S.No.	Year	No. of languages
1	1951 to 1960	12
2	1961 to 1970	16
3	1971 to 1980	07

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

4	1981 to 1990	15
5	1991 to 2000	13
6	2001 to 2010	11
7	2010 to 2020	10
8	2020 to Present	01

III. TYPES OF PROGRAMMING LANGUAGES

1. **By level of abstraction:** Programming languages can be divided into below three levels of abstraction:

- Low-level languages: Low-level languages are used to communicate directly with the hardware since they are near to the machine. Although they are challenging to understand and apply, they are incredibly effective. Machine level language and assembly level language are a few of examples of low-level languages.
- Mid-level languages: A middle ground between low-level and high-level languages is represented by mid-level languages. Compared to low-level languages, they are simpler to learn and use, but they are less effective. C and C++ are two examples of mid-level programming languages.
- High-level languages: The most abstract languages, known as high-level languages, are used to extract algorithms and data structures in a manner that is similar to human language. Although they are the simplest to pick up and use, they are less effective than low-level languages. High-level languages include Python, Java, and JavaScript, as examples.

2. **By paradigm:** The programming paradigm used by a language can be applecable to categorize it. A programming paradigm is a type of programming that establishes the organization and composition of programs. Some popular programming paradigms include:

- Object-oriented programming (OOP): Data and the methods that use it are viewed as objects in OOP languages. Programmers can write reusable code and arrange programs in OOP languages in a way that is simple to comprehend and maintain. OOP languages include Java, C++, and Python as examples.
- Functional programming: Functions are heavily emphasized in functional programming languages. Functional languages use recursion rather than variables to solve issues. Applications in scientific computing and artificial intelligence frequently use functional programming. Lisp and Haskell are two examples of functional programming languages.
- Procedural programming: The foundation of procedural programming languages is the idea of procedures, which are collections of instructions that are carried out one at a time. Although procedural languages are simple to learn and use, maintaining and debugging them can be challenging. COBOL and BASIC are a couple of examples of procedural programming languages.

3. By application: Programming languages can also be classified by their application domain. Some popular application domains include:

- Desktop applications: Software applications that operate on desktop computers are called desktop applications. Languages used for desktop applications include Python, C++, and Java.
- Web development: Websites and web applications are made using web development languages. The languages used in web development include HTML, CSS, and JavaScript.
- Mobile applications: Software applications known as "mobile applications" run on tablets and smartphones. Swift, Kotlin, and Java are a few examples of languages used for mobile applications.
- Artificial intelligence: Agents with artificial intelligence are developed using artificial intelligence languages. Lisp and Prolog are two examples of artificial intelligence languages.
- Scientific computing: To resolve scientific and mathematical issues, scientific computing languages are employed. Python, MATLAB, and R are a few examples of scientific programming languages.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

IV. EXECUTION PROGRAMMING LANGUAGES

Some of the terms for execution of languages are:

- Processor- processor is a computer program which processes other computer programs.
- Compiler- A Compiler is "source code" - the code written in a programming language—into "object code," the code that a computer can run.
- Interpreter- A computer processor that accepts source code programs, transforms them into easily executable formats, and then executes them under supervision. Object code may or may not be the easily executable form.
- Translator- A processor which converts one source code into another.

1. Structures

The computer languages contain structures at the word level, the sentence level and the meaning associated with sentence.

- Lexical structures: This is world level structures and decides about the token in the language. This is identify by abstract machines and called finite automata.
- Syntax structures: These are sentence level structures, expressed by given by parse-tree, and identify by the abstract machine push down automata.
- Semantic structures: It is given by association of nodes in parse-tree.
- Ambiguity in languages: If there is an situation where there is more than one syntax trees for the same sentence then the language and the grammar is misleading.

2. Compiler Writing

This is the process of creating a compiler, which is a program that converts source code written in a high level into machine code that a computer can execute.

A compiler is often divided into three phases:

- Lexical analysis: During this phase, the source code is broken down into tokens, which are the fundamental building elements of the language.
- Syntax analysis: This phase involves checking the source code for syntax correctness.
- Semantic analysis: This step looks for semantic flaws in the source code, such as type mistakes and undefined variables.

After analysing the source code, the compiler generates machine codes that the computer execute.

Compiler writing is a complex and difficult task, but it is also gratifying. Compilers are fundamental for the generation of modern software and playcrucial role in computer science.

Below are the difficulties associated with compiler development:

- The Language: The compiler must grasp the syntax of the computer language being compiled.
- The optimization: The compiler must be able to optimize the generated machine code in order to increase its performance.
- The target machine: The compiler must be capable of producing machine code compatible with the target machine.
- The error handling: The compiler must be capable of detecting and reporting problems in the source code.

V. SCOPE OF PROGRAMMING LANGUAGES

Are the existing programming languages not enough?

There are so many programming languages already available to choose and use and build a computer program.

Still, new languages born year by year, but why?

The requirement for another programming language is determined by a variety of reasons, including the developers' specialized needs and the target platform.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

Developer requirements: Some developers may believe that the existing programming languages do not suit their requirements. For example, they may require a more expressive language or one that is more suited to a specific sort of application.

- The target platform: Some programming languages are better suited to specific target platforms than others. C++, for example, is a strong choice for developing programs for embedded devices, whereas Python is a solid choice for producing online applications.
- The availability of resources: The creation of a new programming language necessitates a large investment of time and resources. If there are already existing languages that satisfy the needs of the developers and the target platform, it may not be worth the expense to build a new language.

Ultimately, the decision of whether to develop a new programming language is a complex one that should be made case wise.

Some reasons why new programming languages are still being created are as follows:

- Handling new issues: As technology grows, new challenges came up that need to the development of new programming languages to address
- Enhancing performance: new programming languages can be develop to enhance the performance of current languages like Rust is a new programming language that aims to be quicker and safer than C++.
- Adding additional features: New programming languages aredevelop to provide new features not found in existing languages. A new programming language that is intended to be simple to learn and use.
- Testing new ideas: new programming languages might be created to test new concepts in programming language design. Haskell, for example, is a new programming language that is intended to be fully functional.

Below are some specific programming languages that are expected to be popular in the future:

- PYTHON: Python was invented by Guido van Rossum in 1991. It is a general purpose language known for its readability and simplicity. Python is used for many different things, including web development; data research, and machine learning.
- JavaScript: JavaScript was developed by Brendan Eich in 1995. It is a scripting language that is used to make web pages more interactive. JavaScript is also becoming more popular for the development of mobile and desktop applications.
- Go: Robert Griesemer, Rob Pike, and Ken Thompson developed Go in 2009. It is a newer language that is intended to be simple, efficient, and scalable. Go is becoming popular for creating cloud-native applications.
- Rust: Graydon Hoare designed the Rust programming language in 2010. It is a new language that is intended to be secure, quick, and expressive. Rust is becoming increasingly popular for creating high-performance applications.
- Julia: Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman, and others created Julia in 2012. It is a new language created for numerical computing. Julia is becoming well-known for her work on scientific and financial applications.
- TypeScript: TypeScriptwas developed by Anders Hejlsberg in 2012. It is a JavaScript typed superset. TypeScript is a language that is used to bring type safety to JavaScript code.
- Swift: Apple Inc. created Swift in 2014. It is a general-purpose programming language for macOS, iOS, watchOS, and tvOS. Swift is well-known for its dependability and performance.
- Kotlin: JetBrains produced Kotlin in 2011. It is a general-purpose language created for the JVM. Kotlin is well-known for its clarity and safety.
- Dart: Dart was created by Google in 2011. It is a general-purpose language created for the web. Dart is well-known for its scalability and performance.
- Elm: Evan Czaplicki designed Elm in 2012. It is a functional programming language intended for web development.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 1, October 2023

VI. CONCLUSION

This paper has presented the evolution of programming languages, Classification of Programming languages, future of programming languages also we see some specific programming languages that are expected to be popular in the future. Computer programming languages have changed over time to meet programmers' needs as well as business expectations. Early languages were basic and ineffective, but they progressively evolved into more complex and powerful languages. There are numerous programming languages accessible today, each with its own unique benefits and drawbacks. The programming language used is determined by the needs of the programmer and the application being built

REFERENCES

[1]. https://www.semion.io/doc/on-the-evolution-of-programming-languages.
[2]. https://en.wikipedia.org/wiki/History_of_programming_languages
[3]. https://www.extremetech.com/computing/91572-the-evolution-of-computer-languages-infographic