International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Comparative Analysis of the Efficiency in Programming Languages: A Case of Study

Jesús A. Román[1], María-Luisa Pérez-Delgado[2]

[1]University of Salamanca, E.P.S. of Zamora,
Av. Cardenal Cisneros, 34, 49022, Zamora, Spain
zjarg[at]usal.es

[2]University of Salamanca, E.P.S. of Zamora,
Av. Cardenal Cisneros, 34, 49022, Zamora, Spain
mlperez[at]usal.es

Abstract: ***Software development is conditioned in many cases by the programming language chosen. Nowadays there are many programming languages, each with different characteristics, so they are different from each other in terms of their performance. A comparison of the computational efficiency of these languages will allow us to have an objective point of view when choosing one of them, and it will also allow us to know the characteristics of the languages that are not obvious in many cases. To test them, a series of programming languages and several algorithms have been selected. The algorithms have been implemented using these languages and the execution time of each one has been determined, which makes it possible to make a comparison that allows us to characterize them.***

Keywords: programming languages, algorithm, efficiency, benchmarking

1. Introduction

When a programming language is selected to develop any application, there are preferences or prejudices that may be taken over a particular language prevail in many cases. However, when choosing a specific language we must go further and put on value its computational efficiency, its learning curve, its speed in terms of development and its internal characteristics.

Due to the number of programming languages currently available [1], and the characteristics of each one, it is necessary to compare their main strengths and weaknesses, as well as the speed of execution when the programs developed in these languages are executed. In this way, there are many organizations that are specialized in a particular programming language for their developments. However, the most prefer to choose the creation of expert work teams in different languages [2].

In this research, 5 different programming languages have been selected to perform the different benchmarking tests. Four of these programming languages are among the ten most used [3], and the last one is a historical programming language. The following paragraphs show a set of characteristics of each proposed programming language.

C: Its popularity is due to the ease it presents when writing compact and very simple code, since it only has functions and lacks procedures. All this is achieved through a syntactic economy and, therefore, through simple structures, a flow control and a large set of operators. It is powerful, its learning curve is fast and it can be applied in an infinite number of projects [4].

Java: It is mandatory to interpret this language through the java virtual machine (JVM), which shows points in favor of this language, such as greater security and stability, less version problems, and less complexity that some of the languages that preceded it, such as C or C ++. It includes a "garbage collector" (automatic memory optimization) and the code goes through some verified bytecodes to ensure the correctness of the code and its secure nature [5].

Pascal: It is easy and fast to learn this language, which uses different syntax for assignments and comparisons. It allows defining assignments within expressions, which avoids errors caused by variables used incorrectly due to an unknown type and also makes Hungarian notation unnecessary (prefixes assigned to the variable names to identify their type) [6].

PHP: It is very simple, but offers a large number of advanced features for professional developers. Its use is widespread and it is completely free. It has a great capacity for adaptation, because in addition to developing web pages it offers other functionalities, such as database communication, sending cookies, evaluating data modules, etc. It is updated with a high frequency, which controls vulnerabilities [7].

Python: It has experienced great growth in recent decades due to its rapid development and simplicity, as well as its libraries, data types and built-in functions. This language is not only limited to Unix, it also presents the option of being able to complete with Windows, Mac, OS / 2, etc. It is also free, which encourages its use [8].

This article is organized as follows. Section 2 presents a theoretical framework on programming languages and an introduction of the proposed programming languages. In section 3 the proposed algorithms for benchmarking the selected languages are selected. Next, section 4 analyzes the results obtained when the algorithms written using each language are executed. Finally section 5 presents the

conclusions obtained in this work.

2. Theoretical Framework

This section presents a general introduction to the types of programming languages and their history. In addition, a brief description is made of each of the programming languages proposed above that allows them to be situated and characterized in a descriptive way.

The programming languages have experienced an evolution from their creation until the present, so we have defined five different generations [9], [10]:

- **1st Generation:** it includes the machine languages, which use the binary system (combinations of zeros and ones) to give orders to a specific processor for its correct operation. Despite being faster than high-level languages, the syntax is complex and makes it difficult to find errors in the source code, which slows down the improvements.

- **2nd Generation:** it includes the assembly languages, which are heirs of the machine languages. Their instructions are defined by abbreviations that combine numbers and letters, called mnemonics. Their source codes are shorter and, therefore, occupy less memory. However, their learning curve is very slow and they are difficult to maintain and execute. These languages gave rise to translator programs, which could translate assembly language into machine language.

- **3rd Generation:** it includes the high-level languages. Their algorithms are adapted to the capacity of intellect and cognition of the human being, and they are fully prepared for interaction with machines.

- **4th Generation:** it includes the RAD ("Rapid Application Development") languages. As the name implies, they are fast development applications that produce codes by themselves. These are languages oriented to the administration and processing of databases.

- **5th Generation:** it includes languages whose fundamental objective is to apply them to artificial intelligence, although the latter are still in the making.

The following paragraphs introduce the programming languages selected to perform the tests that will allow us to compare their results and obtain the conclusions of this research.

First, it is necessary to differentiate a compiler and an interpreter. A compiler is software capable of translating and transforming source language (high-level language) into object language (machine language). On the other hand, an interpreter is software that allows the translation and execution of programs written in a source programming language. Both translators also differ in that a compiler translates the entire program at once while an interpreter translates and executes the program line by line.

C (1972) was created by Dennis Ritchie. It is a general-purpose language, linked to Unix (one of the first operating systems in history and predecessor of others like Linux), although it really does not depend on any system or machine. Although it is mainly used to write compilers and operating systems, it can also be used to develop applications. It differs in its ability to handle data easily manageable by the hardware of most specific computers (numbers, characters, addresses, etc.).

Java (1995) was created by James Gosling and Sun Microsystems. It is a general, concurrent language (simultaneity in the execution of several interactive tasks) and object-oriented. It not only provides software, but has also developed hardware for its execution. According to Sun, Java could be described by a series of key features such as: simplicity, aesthetics, distribution, interpretation, robustness, security, neutral architecture, multithreading (it performs several threads at the same time), portability, high performance and dynamism.

Pascal (1970) was created by Niklaus Wirth and owes its name to the French mathematician Blaise Pascal. It is a general-purpose language, very structured and strongly typed, that is, its code is divided into easily readable portions called functions or procedures and the data type of all variables must be declared beforehand to enable its use. All programs created with this language have two distinct parts: declarative part and operations part, so that everything that will be used in the second part appears in the first one.

PHP (1994) was created by Rasmus Lerdorf and Zeev Suraski. It is a scripting language ("script language" means configured to be executed by an interpreter). It works on the server side to make dynamic content and web pages that produce html, which will then be sent to the client, which will execute the appropriate script.

Python (1991) was created by Guido Van Rossum. It was born as a complement to C within Unix, inspired by the previous ABC language. It is a general purpose scripting language (despite not being designed exclusively for the web, it allows the development of web pages in a satisfactory way), interactive, with dynamic typing, interpreted (without compilers), clear and multi-paradigm (since it is object-oriented but in turn is compatible with both imperative and functional programming).

3. Benchmarking Algorithms

This section describes the algorithms that have been used to compare the different programming languages selected. This comparison is carried out based on the execution time of the algorithms. The proposed algorithms are the following:

- Display in the screen N integer numbers.

N= {100,000; 500,000; 1,000,000}.

Table 1: Algorithm 1

```
for (i = 1 to i = N by 1) do
 write i
end for
```

- Access a local file N times to read from the file and display content on the screen. N={1,000; 2,500; 5,000}.

Table 2: Algorithm 2

```
for (i = 1 to i = N by 1) do
 open file
 read file
 write content
 close file
end for
```

- Access a local file N times to write to it and display content on the screen. N={1,000; 2,500; 5,000}.

Table 3: Algorithm 3

```
for (i = 1 to i = N by 1) do
 open file
 write file
 write content
 close file
end for
```

- Enter N integer numbers in an array and then go through it to write the values on the screen (1,000; 2,500; 5,000).

Table 4: Algorithm 4

```
variable = matrix ()
for (i = 1 to i = N by 1) do
 variable [i] = i
end for

for (i = 1 to i = N by 1) do
 write variable [i]
end for
```

The choice of these algorithms is not random. It has been taken into account that their programming is standard and common to all languages, and that they do not use complex data types that could generate differences external to the desired performance.

4. Benchmarking Tests and Results

This section presents the results of the tests performed with each programming language. Its evaluation was carried out on an HP ENVY 17-j100ns laptop with an Intel Core i7-4710MQ processor (6 MB cache) with four cores of 2.50 GHz each one, RAM / HDD memory: 12 GB RAM DDR3 / 1 TB (5400 RPM), Windows 10 and 6-cell battery.

The environment used to run the programs was the one shown in Table 5.

Table 5: Development environments

Language	*Environment*
C	Dev C++ 5.11 [11]
Java	BlueJ 3.1.7 (JDK 8) [12]
Pascal	Free Pascal 3.0.0 [13]
PHP	PHP for Windows 5.7 [14]
Python	Python 3.5.2 [15]

To measure the runtime, a timestamp of difference has been introduced in the code, so the accuracy is complete. In addition, 10 tests have been performed for each algorithm and language and the average value has been calculated.

Table 6: Execution time for Algorithm 1

Operations	*Execution Time (ms)*				
	C	*Java*	*Pascal*	*PHP*	*Python*
100,000	5,250	27,500	18,305	1,270	38,930
500,000	25,600	52,600	91,407	19,80	203,630
1,000,000	51,30	69,10	183,05	75,15	401,58

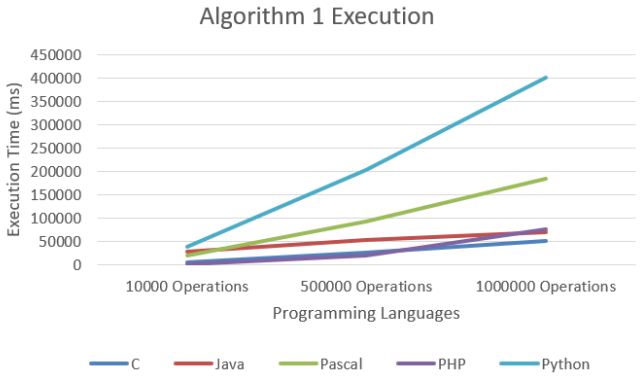

Figure 1: Benchmarking for Algorithm 1

Table 6 and *Figure 1* show the results obtained in the tests performed with algorithm 1. The most efficient programming languages are C, Java and PHP. It is observed that the execution time increases approximately linearly with the number of operations for almost all languages (mainly for Java and C). However, for PHP this linearity begins to be lost when the number of operations increases considerably. The slowest execution corresponds to the Python language with clearly visible differences.

Table 7: Execution time for Algorithm 2

Operations	*Execution Time (ms)*				
	C	*Java*	*Pascal*	*PHP*	*Python*
1,000	321	240	313	126	743
2,500	581	454	783	230	1,769
5,000	1,240	867	1,570	592	3,319

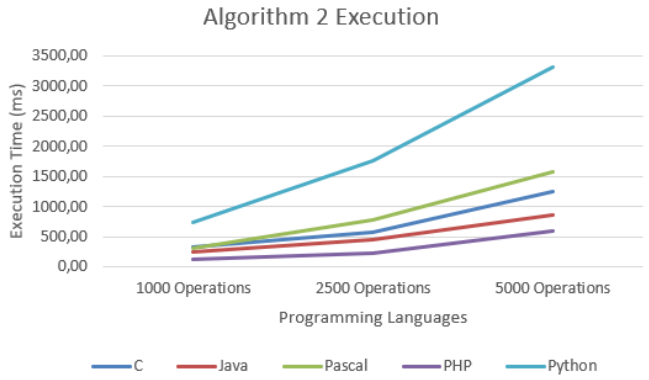

Figure 2: Benchmarking for Algorithm 2

Table 7 and *Figure 2* show the results obtained in the tests performed with algorithm 2. The most efficient programming languages are again C, Java and PHP. In this case, linearity based on the increase of the operations is lost for all of them. The slower execution corresponds again to the Python language, with clearly visible differences.

Table 8: Execution time for Algorithm 3

Operations	*Execution Time (ms)*				
	C	*Java*	*Pascal*	*PHP*	*Python*
1,000	400	577	1,005	427	1,308
2,500	1,360	1,190	1,510	1,75	3,276
5,000	4,320	2,600	2,013	3,14	6,844

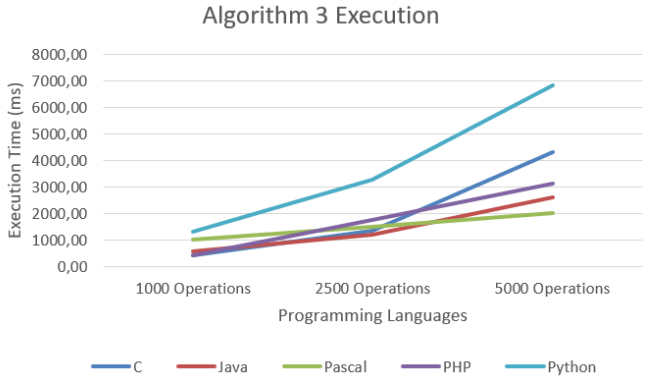

Figure 3: Benchmarking for Algorithm 3

The results of the execution of algorithm 3, which are included in *Table 8* and *Figure 3*, show a variation of the trend followed in the previous algorithms. Python is still the slowest language; however, C becomes the second slowest language as the number of operations increases. In addition, Pascal becomes the second fastest programming language.

Table 9: Execution time for Algorithm 4

Operations	*Execution Time (ms)*				
	C	*Java*	*Pascal*	*PHP*	*Python*
1,000	5.00	0.02	300	1.34	0.99
2,500	5.29	0.04	600	3.28	0.99
5,000	7.78	0.09	1,000	5.98	2.00

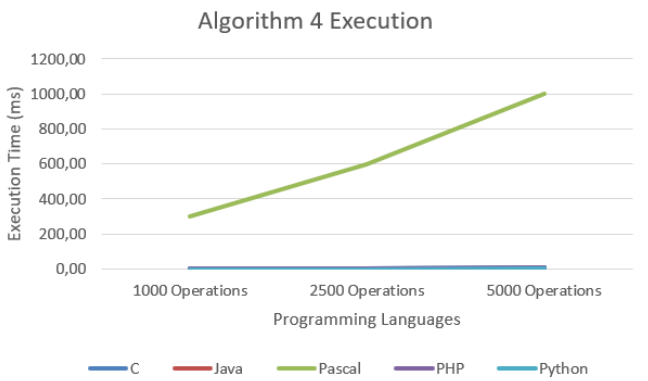

Figure 4: Benchmarking for Algorithm 4

Figure 4 shows an important difference between Pascal and the other programming languages. However, it is in *Table 9* where these differences can be observed in detail. While it is true that the difference in time is very small, considering that this algorithm works with internal memory, Java is the language that behaves in the most efficient way. All programming languages show linearity, except Pascal when the number of operations increases.

5. Conclusions

Throughout this work, several programming languages have been used to implement four algorithms and thus check their computational efficiency. The proposed languages are in the ranking of the most used, except one of them, which is considered one of the historical languages to learn to program. These algorithms have been selected to work with internal memory, data input-output, and standard display output. For this, a simple programming has been sought in terms of standard typology, data and procedures that can be implemented without any problem using the proposed programming languages.

The results obtained in the different tests performed show disparity in the execution of the different algorithms. Although the results of each algorithm are more favorable to a different programming language, it is true that the languages that provide the best overall results for the set of algorithms are Java and PHP.

It cannot be concluded that one programming language is better than another because each one has different characteristics that make it more suitable for specific cases. However, the set of tests carried out shows differences in runtime in the programs that implement the proposed algorithms. This fact indicates that there are programming languages that manage some features better than others; such features include data input-output, internal memory or standard output.

References

[1] Yaofei Chen, R. Dios, A. Mili, Lan Wu and Kefei Wang, "An empirical study of programming language trends," in IEEE Software, vol. 22, no. 3, pp. 72-79, May-June 2005.
[2] P. Kraft. Programmers and managers: The routinization of computer programming in the United States. Springer Science & Business Media, 2012.
[3] TIOBE, "TIOBE Index October 2019". [Online]. Available: https://www.tiobe.com [Accessed: Oct. 05, 2019].
[4] Al Kelley and I. Pohl. A book on C; Programming in C. Benjamin-Cummings Publishing Co., Inc., 1994.
[5] A. Ken, J. Gosling and D. Holmes. The Java programming language. Addison Wesley Professional, 2005.
[6] N. Wirth. "The programming language Pascal." Acta Informatica 1.1 (1971): 35-63.
[7] K. Tatroe, P. MacIntyre and R. Lerdorf. Programming PHP: Creating Dynamic Web Pages. O'Reilly Media, Inc. 2013.
[8] M. Guzdial, and B. Ericson. Introduction to computing and programming in python. Pearson, 2016.
[9] G. O'Regan. Introduction to Programming Languages. In: World of Computing. Springer, Cham, 2018.
[10] S. Valverde, and R. V. Solé. "Punctuated equilibrium in the large-scale evolution of programming languages." Journal of The Royal Society Interface 12.107 (2015): 20150249.
[11] DEV C++, "C, C++ Compiler". Available: https://sourceforge.net/projects/orwelldevcpp/ [Accessed: Oct. 06, 2019]. (General Internet site)
[12] BlueJ, "Java IDE". Available: https://www.bluej.org/ [Accessed: Oct. 06, 2019].
[13] Free Pascal, "Pascal Compiler". Available: https://www.freepascal.org/ [Accessed: Oct. 06, 2019].
[14] PHP, "PHP 5.7 Interpreter". Available: https://www.php.net/ [Accessed: Oct. 06, 2019].
[15] Python, "Python 3.5.2 Interpreter". Available: https://www.python.org/downloads/release/python-352/ [Accessed: Oct. 06, 2019].

Author Profile

Jesús A. Román is a professor in the Computer Science and Automatics Department at the University os Salamanca, Spain. He got his Computer Science Engineering degree from the Pontificial University of Salamanca in 2006, Mcs in Intelligent Systems in 2009, and PhD degree in 2016 from the University of Salamanca. His research interests focus ok the areas of Intelligent Systems, Artificial Intelligence and Computer Science. He has published several research articles and chapters in books related to these areas.

María-Luisa Pérez-Delgado is a Professor in the Computer Science and Automatics Department at the University of Salamanca, Spain. After getting her Computer Science Engineering degree from the University of Valladolid, Spain, she received her PhD degree from the University of Salamanca. Her research interests focus on the areas of artificial intelligence, optimization, graph theory and data mining. She has published several research articles and books related to these areas