Comparison of C++ and Java in Implementing Introductory Programming Algorithms

Muhammad Shumail Naveed

Department of Computer Science Information Technology, University of Balochistan, Quetta, Pakistan

Author's Email: mshumailn@gmail.com

Abstract

Computer programming is the heart of computer science and C++ and Java are the most popular and widely used languages. The performance of C++ and Java programming languages has been benchmarked using robotics, numerical, genetic and evolutionary algorithms, but not using introductory programming algorithms. This article examined C++ and Java by analysing the implementation of novice algorithms of introductory programming courses by evaluating the difficulty, effort, time and delivered bugs of programs with Halstead software metrics. The study identified that Java is more difficult than C++ in the implementation of selected algorithms. Similarly, the efforts required to translate the novice algorithms in Java are higher than C++. The time involved in translating selected algorithms in C++ is lower than Java. Similarly, the number of delivered bugs in C++ is lower than Java. The study suggests that C++ is more suitable than Java for the implementation of introductory programming algorithms and reasonably more suitable for introductory programming courses. The study emboldens the programming linguists to do further analyses in contemplating other programming languages with Halstead software metrics and other kinds of algorithms.

Keywords—C++, Halstead Complexity Metrics, Introductory Programming Courses, Java, Programming Languages

1 Introduction

TECHNOLOGY pervades nearly all areas of society [1]. In the new global environment, the information technology is becoming widespread and seems inevitable. Information technology has successfully integrated in different areas including healthcare [2], education [3], defence [4], marketing [5], and sports [6]. The computer is an essential element of information technology and software is a main part of any digital computer. Software is developed through a formal system called programming languages. Programming is the heart of computer science [7] and a carrier awarding discipline [8]. The job opportunities of programmers are estimated to grow 8% from 2012 to 2022 [9]. Similarly, according to Dice [10], the opportunities for software developers will increase by 2026. The rapid increase in job opportunities of programmers naturally amplifies the challenges in the design of programming languages. Notable achievements have been made and

a plethora of programming languages have been developed. Thousands of programming languages have been developed, but these languages do not endure forever. Indeed, many powerful languages become obsolete some time. Only few programming languages such as C++ and Java are surviving and are popular due to their dynamic features and rapid improvement.

C++ is a general-purpose and high-level language developed by Bjarne Stroustrup. It was designed to provide flexibility and efficiency of C language with program organization facilities of Simula [11]. C++ is a compiler-based language, so it translates the source program into a machine native instruction through a compiler [12], and uses a linker to attach the pre-compiled libraries with generated program of a compiler. Categorically, C++ is an imperative language and treats the program as a collection of statements that change the state of a program. Apart from imperative nature, C++ is a procedural language and thereby supports the notion of procedures and subroutines. It also supports structured programming and is statically typed with smart support of object-oriented programming and generic programming. C++ has an

ISSN: 2523-0379 (Online), ISSN: 1605-8607 (Print)

DOI: <https://doi.org/10.52584/QRJ.1901.14>

This is an open access article published by Quaid-e-Awam University of Engineering Science & Technology, Nawabshah, Pakistan under CC BY 4.0 International License.

extensive support to manipulate memory at low level. Pre-processors are also available in C++ [13] as a text-substitution mechanism.

Java is a successful and powerful programming language developed by James Gosling [14]. Java is strongly and statically typed [15]. Its syntax is inspired from C++ and it is a compiled and interpreted language. Platform independence and security are its unique features. Java is extensively used in big data applications [16] and provides a strong solution to develop robust and efficient object-oriented software [17]. Pre-processors and linkers are not available in Java, and packages are loaded or imported dynamically at runtime. Java does not support operator overloading. C++ and Java are not only used in commercial applications, but also in the pedagogy of introductory programming. In the introductory course on programming (CS1), the conventional algorithms of programming are offered to learners with suitable programming languages. It has been widely observed that learning to program is challenging and hard for beginners [18–19], and normally manifested in the form of high dropout and failure [20].

The intrinsic complexity of introductory programming courses naturally challenge the researchers and consequently a choice of programming language for introductory programming courses has been an area of study for decades [21]. Still there is no consensus on the choice of programming language for an introductory programming course. Many instructors agree to use C++ or Java in CS1; whereas, several opposed their induction in introductory courses on programming, for instance, Agarwal and Agarwal [22] has identified several issues of teaching C++ in CS1, and similarly Nasrawt and Lam has mentioned the drawbacks of using Java in introductory programming [23].

The selection of appropriate language for introductory programming courses is usually made on an empirical basis by experimenting on the groups of learners. Invention of programming languages initiated the best and ongoing research on the pros and cons of programming languages through the comparative evaluation. As a matter of fact, the comparative evaluation of programming languages has gained some attention from early on. Most studies devoted to analyse the runtime performance, integrity, compiled size, reliability, memory utilization [24] and vulnerabilities, yet very less focus has been concentrated on evaluation of the implementation of introductory programming algorithms in topical programming languages. The prospective evaluation is essentially important both from the technical and pedagogical aspects.

The central objective of present study is to com-

pare C++ and Java by analysing the implementation of introductory programming algorithms. The novelty of this work lies in two affirmations: i) the use of Halstead complexity metrics in the comparison of programming languages, and ii) the endorsement of suitable language for introductory programming courses through quantitative analysis rather than the empirical examination. To the best of our knowledge, no study of such pattern has yet been conducted. The proposed study would provide a quantitative knowledge about the suitability of C++ and Java in introductory programming courses along with numerous metrics.

The organization of this article is as follows. Section 2 discusses the related work on comparison of C++ and Java. Section 3 describes the research method and results. Section 4 presents the discussion followed by a conclusion.

2 Related Work

There have been many studies on comparative analysis of C++ and Java. Phipps [25] conducted a study to analyse C++ and Java in terms of defect rates and productivity. Personal Software Process, which is self-improvement process [26], is used to collect data from development projects. The study identified no difference between C++ and Java while analysing the defects against development time.

In last many decades, the evolutionary algorithms have shown to be a useful solution for optimization problems [27]. Merelo-Guervo et. al. [28] analysed evolutionary algorithms over OneMax function, bitflip mutation and crossover, and evaluated the performance of interpreted and compiled languages including C, C++, Java and Python. The study reported Java as fastest, but pointed that interpreter-based languages should not be much slower than compiled languages.

Robots' applications are the complex class of software. Compiler-based languages like C and C++ are normally considered more suitable to implement robotics software. Gherardi et. al. [29] analysed the runtime performance of Java and C++ in the implementation of robotics algorithms. The study reported that on Windows platform, Java is around 1.09 to 1.51 times slower than C++ and around 1.21 to 1.91 times slower on Linux platform.

elPrep is a framework for processing Binary Alignment in sequencing pipelines. Memory management is a major challenge in elPrep. Costanza et. al. [30] implemented nontrivial subset of elPrep in C++, Java and Go, and analysed their memory utilisation and execution performance. The descriptive results reported

that C++ and Java are not reasonable to implement elPrep.

Generic programming is a pivotal element that supports the development of reusable components in software construction [31]. Saiedian and Hill [32] analysed the support of generic programming in C++ and Java. Qualitative as well as some quantitative attributes including performance, memory usage at runtime and size are analysed in the study. The evaluation described that support of generic programming in C++ is higher and better than the one provided by Java Development Kit. Likewise, in another study [33], it is opined that C++ has an extra expressive strength of generic programming; whereas, generic in Java is a syntactic sugar and merely promotes casting from coding level to the bytecode level.

Multiple inheritance is an essential feature of object-oriented programming in which a class inherits the features from single or multiple parents. Albaloooshi and Mahmood [34] studied the implementation of multiple inheritance in C++, Java and Python and analysed the complexity and reusability with Chidamber and Kemerer metrics. The study suggested that C++ and Python support better reusability with multiple inheritance; whereas, Java has poor structure of objects in the implementation of multiple inheritance.

Fourment and Gillings [35] benchmarked the performance of programming languages through the analysis of implementing the Bioinformatics algorithms in different languages including C++ and Java. The study analysed the execution speed and memory utilisation by evaluating the algorithms of parsing BLAST file results, Neighbor-Joining tree construction algorithm and Sellers algorithms. The study concluded that C++ is faster and uses less memory than Java in the implementation of Bioinformatics algorithms.

TIOBE Company analysed the popularity of programming languages by maintaining a TIOBE programming community index and update the index once a month [36]. The index is based on the courses, third-party vendors and software engineers worldwide. Popular search engines including Google, Amazon, YouTube, Bing, Wikipedia and Yahoo are used to calculate the programming index.

A comprehensive popularity index of programming languages over different years provided by TIOBE is shown in Figure 1. According to TIOBE index, the popularity of Java remains higher than C++, yet a variation in popularity of both languages is observed in different span of time. PYPL (Popularity of Programming Language) is another forum for ranking topical programming languages. It is based on Google trends and like TIOBE it also provides a comprehensive pop-

Fig. 1: TIOBE programming community index

Fig. 2: PYPL index of C++ and Java

ularity index of programming languages of different years. The logarithmic scale of C++ and Java is shown Figure 2 [37]. IEEE Spectrum, in its sixth annual ranking of the top programming languages, listed Java on number 2 with the score of 96.3, while C++ had a score of 87.5 [38].

In present study, a pragmatic research methodology is defined and followed during the accomplishment of work (Figure 3).

3 Design & Methods

The study virtually aims to compare C++ and Java by evaluating the implementation of introductory programming algorithms with software metrics. During the study, two hundred algorithms are selected from different books and online sources. The threshold concepts covered by the selected algorithms are shown in Table 1. For selected algorithms, the source code of C++ and Java are developed with high-level code generator [7]. The current study selected those algorithms which are covered in CS1. The introductory programming courses conventionally used imperative and object-oriented programming language and therefore C++ and Java are selected for the present study. Python is a popular language and widely used in medical imaging [39], Solar Photovoltaic Systems [40] and text analysis [41]. Categorically, Python is a multi-paradigm language and its pedagogical significance in introductory programming courses is not very significant. Alzahrani et. al. [42] compared Python with C++ and identified that learning of Python required

Fig. 3: Research methodology

S. No	Topics	Algorithms
1	Control structures	25
2	Recursion	11
3	Conventional problems	35
4	Searching algorithms	8
5	Sorting algorithms	20
6	String processing	9
7	Stack & Queue	7
8	Linked list	11
9	Matrix processing	15
10	Tree	10
11	Heap	4
12	Graphs	13
13	Greedy algorithms	5
14	Dynamic programming	4
15	Numerical computation	23

TABLE 1: Description of the selected algorithms

higher struggle rates than C++. Similarly, Johnson et. al. [43] demonstrated that single Python is inadequate as an introductory programming language. The present study analysed the implementation of those algorithms which are offered in introductory programming. Since Python is not suitable for introductory programming, it is not considered for the current study.

The collected programs, formalized as a programming corpus, are analysed with Halstead complexity metrics (HCM) which is a useful way to analyse the complexity of code [44]. HCM was specially developed to quantify program complexity from source code by using the detail of operators and operands as an elementary information. The elementary information of operators and operands are based on following four variables.

η_1 = number of distinct operators.

η_2 = number of distinct operands.

N_1 = total number of occurrences of operators.

N_2 = total number of occurrences of operands.

The Halstead complexity metrics use the elementary information of operands and operators and relate them with simple mathematical equations to calculate the program difficulty, efforts, times and delivered bugs. The difficulty measure is associated to the difficulty of

the program to write or understand. The formula for calculating difficulty is given as follows.

$$Difficulty = \left(\frac{\eta_1}{2}\right) \times \left(\frac{N_2}{\eta_2}\right)$$

The effort is defined as a measure of the mental activity required to reduce a preconceived algorithm to a program. It can be viewed as an elementary mental discrimination required to generate a program. The formula for calculating effort is given as follows.

$$Effort = \left(\frac{\eta_1}{2}\right) \times \left(\frac{N_2}{\eta_2}\right) \times ((N_1 + N_2) \times \log_2(\frac{\eta_1}{\eta_2}))$$

The time (in seconds) required to translate the algorithm into implementation in the specified program language is calculated as follows.

$$Time = \left(\frac{\eta_1}{2}\right) \times \left(\frac{N_2}{\eta_2}\right) \times ((N_1 + N_2) \times \frac{\log_2(\eta_1 + \eta_2)}{18})$$

The delivered bugs represent an estimate for the number of errors in the implementation of algorithm. The formula for calculating delivered bugs is as follows.

$$Bugs = \left(\frac{\eta_1}{2}\right) \times \left(\frac{N_2}{\eta_2}\right) \times \frac{(N_1 + N_2) \times \log_2(\eta_1 + \eta_2)}{1500}$$

During the study, the programming corpus is analysed with Halstead Metric tool which is a software metrics analyser for several programming languages and freely available on SourceForge¹. The programs are virtually analysed in two stages. In the first stage, the basic attributes are identified which are shown in Table 2. During first phase, programs are lexically analysed. Operators and reserved words are categorised as operators; whereas, identifiers and constants are recognized as operands. Overall, 14446 operators are identified in the programming corpus of C++; whereas, 17103 operators are recognized in the corpus of Java. Similarly, 5674 unique operators are identified in corpus of C++; whereas, 6215 in Java. Moreover, 7116 operands and 2309 unique operands are recognized in corpus of C++; whereas 7140 operands and 2476

1. <https://sourceforge.net>

Attribute	Language	Mean	Median	Variance	Std. Dev.	Min	Max	Range	Skewness	
Operators	C++	72.23	65.5	570.25	23.88	34	150	116	0.98	0.66
	Java	85.52	80	756.38	27.5	42	175	133	0.89	0.64
Distinct Operators	C++	28.37	29	16.62	4.08	18	43	25	0.1	0.09
	Java	31.08	31	21.41	4.63	19	46	27	0	-0.05
Operands	C++	35.58	30	245.6	15.67	13	92	79	1.15	0.99
	Java	35.7	32	223.82	14.96	16	92	76	1.2	1.23
Distinct Operands	C++	11.55	11	14.16	3.76	4	25	21	0.66	0.21
	Java	12.38	12	12.61	3.55	4	26	22	0.85	1.22
Program Length	C++	107.81	97	1533.27	39.16	48	242	194	1.06	0.82
	Java	121.22	113	1675.57	40.93	58	267	209	0.98	0.77
Estimated Program Length	C++	179.13	178.94	1064.07	32.62	103.59	257.33	153.74	0.07	-0.47
	Java	200.42	198.33	1207.6	34.75	125.55	303.46	177.91	0.15	-0.02

TABLE 2: Result of first phase analysis

unique operands are in the corpus of Java. This analogously described that C++ required less operands than Java while the implementation of the algorithms. The statistics collected during the first stage are further used in the second stage. Volume, difficulty, effort, time and bugs are calculated which are shown in Table 3. The volume in Halstead complexity represents a relevant measure for the size of an implementation of any algorithm. The corpus of C++ analysed during the study has a cumulative volume of 115806.94, and 133034.48 in the case of Java. Similarly, the cumulative difficulty of C++ for the analysed program is 3558.00; whereas, 3570.00 for the corpus of Java. The programming efforts in Halstead complexity represents the mental activities required to implement the algorithm into a program. The cumulative programming efforts of C++ corpus are 2419831.27 and 2720431.84 for the Java. Likewise, the cumulative time required to implement the selected algorithms in C++ is 134435.07 seconds; whereas 151135.10 seconds in Java. Moreover, the cumulative delivered bugs for implementing selected algorithms in C++ are 32.69; whereas 35.67 in Java. For the visualization of calculated measures, the box plots, shown in Figure 4, are created with R (version 4.0.3). The density curves illustrated in a box plot described that for each measure, the rank of C++ is better than Java. For further analysis, the calculated values of volume, difficulty, effort, time and bugs are further checked for normality with SPSS (version 25) and results are shown in Table 4. The selected algorithms in the study are of different sizes and functionalities, therefore, the developed programs vary in terms of size and functionality. Hence, the calculated measures are not normal distributed. Mann-Whitney U test is conducted on the calculated measures for statistical analysis and results are shown in Table 5. The Mann-Whitney U tests conducted on calculated measures show that in terms of volume, efforts, time and bugs, the differences between C++ and Java are

statistically significant, however, only in the case of difficulty, the difference between C++ and Java is not statistically significant, yet the difficulty level of C++ is lower than Java.

4 Discussion

The study reported in this article comparatively evaluate C++ and Java by using Halstead complexity metrics. The implementation of 200 algorithms in C++ and Java is analysed. In the programming corpus of C++, 14446 operators are identified, whereas 17103 operators are recognized in the corpus of Java for same algorithms. The difference defined that Java required 18.93% more operators than C++. Similarly, in Java 9.53% unique operators are more involved than in C++. The average program length of C++ corpus is 107.81, whereas 121.22 in Java. This described that difference between both languages in terms of program length is 11.71% and evidently C++ is much finer than Java in terms of program length. The same behaviour is observed in estimated program length, in which C++ is found better than Java with the difference of 12.21%.

The mean value of volume is 579.03 and 665.17 for C++ and Java respectively, which identify the difference of 18.85%. The Mann-Whitney U test applied to the volume described that the difference between C++ and Java is statistically significant. This signifies that C++ is more suitable than Java in terms of size of the implementation of the algorithms. The mean score for the difficulty of implementing selected algorithms in C++ is 17.79, whereas 17.85 in Java. The Mann-Whitney U test conducted on calculated difficulty described that difference between C++ and Java is not statistically significant, yet the implementation of algorithms in Java is more complex than C++.

The average score of efforts required to translate the algorithms into C++ is 12099.16, whereas

Parameters	Volume		Difficulty		Efforts		Time		Bugs	
	C++	Java	C++	Java	C++	Java	C++	Java	C++	Java
Mean	579.03	665.17	17.79	17.85	12099.16	13602.16	672.18	755.68	0.16	0.18
Median	514.44	619.38	15.00	16.00	7697.79	9449.88	427.65	524.99	0.13	0.15
Std. Dev.	234.90	250.18	7.84	7.48	10982.76	11558.47	610.15	642.14	0.09	0.10
Minimum	237.80	287.34	6.50	8.00	1642.32	2298.75	91.24	127.71	0.05	0.06
Maximum	1392.71	1577.14	46.00	46.00	63145.44	72548.43	3508.08	4030.47	0.53	0.58
Range	1154.91	1289.80	39.50	38.00	61503.12	70249.68	3416.84	3902.76	0.48	0.52
Skewness	1.11	1.04	1.15	1.20	2.06	2.13	2.06	2.13	1.43	1.45
Kurtosis	1.03	1.05	0.99	1.23	4.79	5.64	4.79	5.64	2.00	2.32
Huber's M-Estimator	537.91	627.88	16.27	16.45	8980.81	10477.83	0.14	0.16	0.14	0.16
Tukey's Biweight	520.83	618.06	15.66	15.95	7924.67	9378.47	0.14	0.15	0.14	0.15
Hampel's M-Estimator	541.83	633.58	16.43	16.60	8798.12	10372.56	0.14	0.16	0.14	0.16
Andrews' Wave	520.74	617.69	15.65	15.95	7912.60	9347.21	0.14	0.15	0.14	0.15

TABLE 3: Result of second phase analysis

Fig. 4: Bean plots of the calculated measures

Measures	Kolmogorov-Smirnov			Shapiro-Wilk		
	Statistic	Df	Sig.	Statistic	Df	Sig.
Volume	0.121	200	0.00	0.912	200	0.00
	0.118	200	0.00	0.929	200	0.00
Difficulty	0.149	200	0.00	0.902	200	0.00
	0.133	200	0.00	0.896	200	0.00
Effort	0.171	200	0.00	0.772	200	0.00
	0.171	200	0.00	0.779	200	0.00
Time	0.171	200	0.00	0.772	200	0.00
	0.171	200	0.00	0.779	200	0.00
Bugs	0.154	200	0.00	0.865	200	0.00
	0.142	200	0.00	0.871	200	0.00

TABLE 4: Result of normality tests

Parameter	Language	Mean Ranks	Mann-Whitney U	Z	Asymp. Sig. (2-tailed)
Volume	C++	178.20	15539.50	-3.86	0.00
	Java	222.80			
Difficulty	C++	198.55	19610.50	-0.34	0.74
	Java	202.45			
Effort	C++	188.10	17520.00	-2.15	0.03
	Java	212.90			
Time	C++	188.10	17520.00	-2.15	0.03
	Java	212.90			
Bugs	C++	187.98	17496.00	-2.17	0.03
	Java	213.02			

TABLE 5: Result of Mann-Whitney U test

13602.16 in Java. This shows that the difference between both languages in terms of efforts is 11.69%. The Mann-Whitney U test conducted on the calculated efforts shows that the difference between C++ and Java is statistically significant. Therefore, as compare to Java, less elementary mental discrimination is required in C++ to generate the programs for conventional algorithms. The mean score of time involved to implement the selected algorithms in C++ is 672.18, whereas 755.68 in Java. Therefore, the time difference between both C++ and Java is 11.70%. The Mann-Whitney U test applied on the calculated time showed that the difference between C++ and Java is statistically significant. Therefore, C++ is found to be more efficient than Java in terms of time required to implement the algorithms.

The average score of the delivered bugs in corpus of C++ is 0.16, whereas 0.18 in Java. Therefore, the difference is 11.76%. The Mann-Whitney U test conducted on delivered bugs described that difference between C++ and Java is statistically significant. Hence, C++ is relatively better than Java on the subject of delivered bugs.

All the tests lead to a conclusion that C++ is better than Java in implementing the introductory programming algorithms. The results obtained in the study are further compared with the findings of other studies and are shown in Table 6. The detail included in Table 6 clearly delineate that the existing studies on C++ and Java do zilch in comparing and examining the difficulty, time and efforts required in the implementation of any sort of algorithms. The purview and justification for the accomplishment of this work lies in the formation of new knowledge and a novel pattern of study that has quantitatively analysed C++ and Java in different dimension by using Halstead complexity for determining the difficulty, time, efforts and bugs involved in the realization of introductory programming algorithms.

5 Conclusion

Programming is a core element of computer science and comparison of programming languages is an active area of programming languages. The popular languages like C++ and Java are being compared in assorted dimensions; however, no notable study has compared these languages in implementing the fundamental algorithms being offered in introductory programming courses. The selection of an appropriate programming language for introductory programming is conventionally decided on empirical corroborations. This study used Halstead complexity metrics and quantitatively analysed the C++ and Java by analysing the implementation of 200 conventional computer algorithms of introductory programming courses. The results are statistically analysed which showed that for the selected algorithms, the performance of C++ is much higher than that of Java in that it entails less difficulty to write and understand the programs. Similarly, C++ involves less mental discrimination in the generation of programs. Likewise, C++ required less time to translate and debug the programs and thereby is more suitable for introductory programming courses. This study presents a new tool for comparing and selecting felicitous language for introductory programming courses.

As future research, new algorithms will be added for more investigation and other techniques like MOOD metrics and Chidamber and Kemerer metrics for more detailed analysis.

Acknowledgement

The author would like to thank Dr. Muhammad Tahaam for his constructive suggestions in data collection and analysis. Special thanks to Dr. Muhammad Aayaan for giving valuable feedback in writing and improving the article.

S. No	Study	Methods	Results
1	Phipps [19]	Defect rates in C++ and Java are compared	C++ and Java have no difference in defect rates
2	Merelo-Guervo's et al. [22]	Analyzed the implementation of evolutionary algorithms in compiled and interpreted programming languages	Java is top-ranked
3	Gherardi et. al. [23]	Performance of C++ and Java is analysed through the implementation of robotics algorithms	C++ is faster than Java
4	Costanza et. al. [24]	C++, Java and Go are analysed in the implementation of elPrep multithreaded framework.	C++ and Java are less reasonable in term of memory utilization
5	Saiedian and Hill [26]	Support of generic programming in C++ is compared with Java by analyzing the size, performance and memory usage	C++ is better than Java in the support of generic programming
6	Albalooshi and Mahmood [28]	Multiple inheritance in C++, Java and Python is analyzed with Chidamber and Kemerer metrics	C++ and Python provide better multiple inheritance over Java
7	Fourment and Gillings [29]	Performance of C++ and Java is evaluated through the implementation of bioinformatics algorithms	C++ is faster and use less memory than Java
8	TIOBE Index [30]	Programming languages, including C++ and Java rank on popularity scale	Java is top-ranked, whereas C++ is on fourth rank
9	PYPL [31]	Programming languages including C++ and Java are ranked on popularity scale	Java is ranked at second, whereas C++ is on the sixth rank
10	IEEE Spectrum [32]	Rank the programming languages by using different metrics and data sources	The second position is assigned to Java and fourth to C++
11	Present Study	C++ and Java are comparatively evaluated by analyzing the volume, difficult, effort, time and bugs.	C++ is better than Java in implementing computer algorithms

TABLE 6: Comparison of studies on C++ and Java

References

- [1] R. Scherer, F. Siddiq and J. Tondeur, "The technology acceptance model (TAM): A meta-analytic structural equation modeling approach to explaining teachers' adoption of digital technology in education", *Computers Education*, vol. 128, pp. 13-35, 2019.
- [2] A. Meri, M. Hasan, M. Danaee, M. Jaber, M. Jarrar, N. Safei, M. Dauwed, S. K. Abd and M. Al-bsheish, "Modelling the utilization of cloud health information systems in the Iraqi public healthcare sector", *Telematics and Informatics*, vol. 36, pp. 132-146, 2019.
- [3] D. Farjon, A. Smits and J. Voogt, "Technology integration of pre-service teachers explained by attitudes and beliefs, competency, access, and experience", *Computers Education*, vol. 130, pp. 81-93, 2019.
- [4] C. Wilson, "Artificial Intelligence and Warfare", 21st Century Prometheus, Springer, pp. 125-140, 2020.
- [5] L. Busca and L. Bertrandias, "A Framework for Digital Marketing Research: Investigating the Four Cultural Eras of Digital Marketing", *Journal of Interactive Marketing*, vol. 49, pp. 1-19, 2020.
- [6] T. Kim and W. Chiu, "Consumer acceptance of sports wearable technology: the role of technology readiness. International Journal of Sports Marketing and Sponsorship", *International Journal of Sports Marketing and Sponsorship*, vol. 20, no. 1, pp. 109-126, 2019.
- [7] M. S. Naveed, M. Sarim and K. Ahsan, "Learners Programming Language a Helping System for Introductory Programming Courses", *Mehran University Research Journal of Engineering Technology*, vol. 35, no. 3, pp. 347-358, 2016.
- [8] M. S. Naveed, M. Sarim and Adnan Nadeem, "C in CS1: Snags and Viable Solution", *Mehran University Research Journal of Engineering Technology*, vol. 37, no. 1, pp. 1-14, 2018.
- [9] M. S. Naveed and M. Sarim, "Pedagogical Significance of Natural Language Programming in Introductory Programming." *Journal of Basic and Applied Sciences*, vol. 14, pp. 62-71, 2018.
- [10] <https://insights.dice.com/2019/01/03/software-developer-jobs-increase-2026/> Accessed: 25th February, 2021
- [11] B. Stroustrup, "Evolving a language in and for the real world: C++ 1991-2006", In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pp. 4-14-59. 2007.
- [12] D. Rassokhin, "The C++ programming language in cheminformatics and computational chemistry", *Journal of Cheminformatics*, 12:10, 2020.
- [13] M. Olsson, "Preprocessor", In *C++17 Quick Syntax Reference*, Apress, 2018.
- [14] G. O'Regan, "Java Programming Language", *The Innovation in Computing Companion*, Springer, pp. 171-174, 2018.
- [15] J. Gosling, B. Joy, G. Steele, G. Bracha and A. Buckley, "The Java Language Specification", *Java SE 8th Edition*, Addison-Wesley Professional, 2014.
- [16] G. Lebanon and M. El-Geish, "Learning Java", *Computing with Data*, Springer, pp. 169-190, 2018.
- [17] I. Lagartos, J. Manuel Redondo and F. Ortin, "Efficient runtime metaprogramming services for Java", *Journal of Systems and Software*, pp. 220-237, 2019.

- [18] S. I. Malik, M. Shakir, A. Eldow and M. W. Ashfaq, "Promoting Algorithmic Thinking in an Introductory Programming Course", *International Journal of Emerging Technologies in Learning*, vol. 14, no. 1, pp. 84-94, 2019.
- [19] M. S. Naveed, M. Sarim, "Analyzing the Effects of Error Messages Presentation on Debugging and Programming", *Sukkur IBA Journal of Computing and Mathematical Sciences*, vol. 4, no. 2, pp. 38-48, 2020.
- [20] E. D. Canedo, G. A. Santos, L. L. Leite, "An Assessment of the Teaching-Learning Methodologies Used in the Introductory Programming Courses at a Brazilian University", *Informatics in Education*, vol. 17, no. 1, pp. 45-59, 2018.
- [21] Simon, R. Mason, T. Crick, J. H. Davenport and E. Murphy, "Language Choice in Introductory Programming Courses at Australasian and UK Universities", In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 852-857, 2018.
- [22] A. Agarwal and K. Agarwal, "Some Deficiencies of C++ in Teaching CS1 and CS2", *ACM SIGPLAN Notices*, vol. 38, no. 6, pp. 9-13, 2006.
- [23] Z. O. Nasrawt and M. O. Lam, "Less-Java, More Learning: Language Design for Introductory Programming", *The Journal of Computing Sciences in Colleges*, vol. 34, no. 4, pp. 64-72, 2019.
- [24] L. Prechelt, "An Empirical Comparison of Seven Programming Languages", *Computer*, vol. 33, no. 10, pp. 23-29, 2000.
- [25] G. Phipps, "Comparing Observed Bug and Productivity Rates for Java and C++", *Software: Practice and Experience*, vol. 29, no. 4, pp. 345-358, 1999.
- [26] W. Jirapanthong, "Personal Software Process with Automatic Requirements Traceability to Support Startups", *Journal of Reviews on Global Economics*, vol. 6, pp. 367-374, 2017.
- [27] L. M. Antonio and C. A. C. Coello, "Coevolutionary Multi-objective Evolutionary Algorithms: Survey of the State-of-the-Art", *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 851-865, 2018.
- [28] J. Merelo-Guervos, I. B. Alvarez, P. A. Castillo and G. Romero, "A comparison of implementations of basic evolutionary algorithm operations in different languages", *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1602-1609, 2016.
- [29] L. Gherardi, D. Bruggi and D. Comotti, "A Java vs. C++ performance evaluation: a 3D Modeling benchmark", In *Proceeding of International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pp. 161-172, 2012.
- [30] P. Costanza, C. Herzeel and W. Verachtert, "Comparing Ease of Programming in C++, Go, and Java for Implementing a Next-Generation Sequencing Tool", *Evolutionary Bioinformatics*, vol. 15, pp. 1-5, 2019.
- [31] L. Chen, D. Wu, W. Ma, Y. Zhou, B. Xu and H. Leung, "How C++ Templates Are Used for Generic Programming: An Empirical Study on 50 Open Source Systems", *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 1, pp. 1-49, 2020.
- [32] H. Saiedian and S. Hill, "A comparative evaluation of generic programming in Java and C++", *Software: Practice and Experience*, vol. 33, pp. 121-142, 2003.
- [33] D. Ghosh, "Generics in Java and C++ a comparative model", *ACM SIGPLAN Notices*, vol. 39, no. 5, pp.40-47, 2004.
- [34] F. Albaloooshi and A. Mahmood, "Comparative Study on the Effect of Multiple Inheritance Mechanism in Java, C++, and Python on Complexity and Reusability of Code", *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, pp. 109-116, 2017.
- [35] M. Fourment and M. R. Gillings, "A comparison of common programming languages used in bioinformatics", *BMC Bioinformatics* 9:82, 2008.
- [36] <https://www.tiobe.com/tiobe-index/> (Accessed: 10th March, 2021)
- [37] <https://pypl.github.io/PYPL.html> (Accessed: 18th May, 2021)
- [38] <https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019> (Accessed: 10th March, 2021).
- [39] A. Jungo, O. Scheidegger, M. Reyes and F. Balsiger, "pymia: A Python package for data handling and evaluation in deep learning-based medical image analysis", *Computer Methods and Programs in Biomedicine*, vol. 198, 2021.
- [40] R. Sivapriyan, D. Elangovan, K. S. N. Lekhana, "Review of Python for Solar Photovoltaic Systems", *Evolutionary Computing and Mobile Sustainable Networks, Lecture Notes on Data Engineering and Communications Technologies*, vol. 53, 2021.
- [41] M. Anandarajan, C. Hill, T. Nolan, "Latent Semantic Analysis (LSA) in Python", *Practical Text Analytics*, vol. 2, pp. 221-242, 2019.
- [42] N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen and R. Lysecky, "Python Versus C++: An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses", In *Proceedings of the 49th ACM Technical Symposium on Computer Science*, pp. 86-91, 2018.
- [43] F. Johnson, S. McQuistin and J. O'Donnell, "Analysis of Student Misconceptions using Python as an Introductory Programming Language", In *Proceedings of the 4th Conference on Computing Education Practice*, pp. 1-4, 2020.
- [44] M. S. Al-Batah, N. Alhindawi, R. Malkawi and A. Al-Zuraiqi, "Hybrid Technique for Complexity Analysis for Java Code", *International Journal of Software Innovation*, vol. 7, no. 3, pp. 118-133, 2019.