University of Sindh Journal of Information and Communication Technology (USJICT)
Volume 5, Issue 4, December 2021

ISSN-E: 2523-1235, ISSN-P: 2521-5582 © Published by University of Sindh, Jamshoro
Website: http://sujo.usindh.edu.pk/index.php/USJICT/

Measuring the Programming Complexity of C and C++ using Halstead Metrics

Muhammad Shumail Naveed

Department of Computer Science & Information Technology, University of Balochistan
mshumailn@gmail.com

Abstract: Computer algorithm is the core of computer science and important prerequisite of computer science professionals. However, its hard and abstract nature makes it difficult to understand. Pedagogical issues in learning of algorithms are generally resolved through elaborating the algorithms with their implementation in some programming language. As there are many programming languages, the selection of appropriate programming language for effective implementation of algorithms remains a challenging issue. In this article, common algorithms of data structures are measured by analyzing their implementation in C and C++ through Halstead complexity metrics. For statistical analysis Shapiro-Wilk and Kolmogorov-Smirnov were used to test whether the results are well-modelled by a normal distribution. The results of study were analyzed with Mann-Whitney U test which identified that as compare to C++ the less effort (3.99% difference), time (3.90%) and bugs (10%) are involved in C for the implementation of algorithms, whereas C++ involves less difficulty (10.51%) during the implementation of sampled algorithms. The work stated in this article provide a novel aspect to relate and evaluate other programming languages.

Keywords: Halstead Metrics; Computer Algorithms; Programming languages; C, C++;

I. INTRODUCTION

The digital era has transformed the world and workplace, making computing technologies essential part of life. The need of computer science professionals has increased expeditiously. Analogous to this increasing requirement for computer science personnel is the constant change in the kind of expertise that are brought about by inventions in cutting edge technologies. Requirement of computationally knowledgeable workers are increasing [1], and projected to grow 12% from 2018 to 2028 [2].

In that connection, it becomes essential to have a clear understanding of computer science. Virtually, computer science is not a single discipline, but a combination of several areas, including algorithms & data structures, programming, computer architecture, networks and artificial intelligence.

The algorithm is a central part of computer science [3], and a key pillar of software development [4]. Algorithms play a vital role in computing education. Beginners naturally need to be familiar with different algorithms and their corresponding data structures. Similarly, the students are not only expected to learn the functionality of algorithms, but also how a problem should be resolved by particular algorithm. Similarly, computer professionals are expected to have proficiency in the design and optimization of algorithms [5].

Learning algorithms and gaining algorithmic thinking is extremely arduous [6, 7], and challenging for both beginners and instructors. Pedagogical issues of comprehending algorithms are usually attempted through the didactic strategies like the use of visualization that explains algorithms for beginners by illustrating the complex process.

Algorithm visualization used graphics and animation [8], dynamically demonstrate the process of algorithm by providing a step-by-step illustration of operations. The use of visualization increased the motivation of students and aid them to concentrate on the actual process, but visualization tools does not address technical aspect [9], so they are not pedagogically productive in every case.

Conventionally, the performance of the algorithms is measured either by calculating time or space complexity. This kind of analysis is usually language independent. However, the algorithms are generally designed to be implemented as programs [10]. It is widely recommended [11], to use real programming languages for the courses on algorithms.

A programming language provides a collection of primitives, rules and operators. There are about thousands of programming languages which are categorized in different paradigms. Programming in general is a hard subject to comprehend [12]. It entails several abstract notions and beginners rarely receive adequate level of personal instructions. Although thousands of programming languages have been developed, but all of them never survive, only few like C and C++ are alive due to potent attributes and salient features.

C and C++ are the descendent of Classic C. Over the years, these programming languages have grown in different dimensions and paces. Resultantly, each language delivers the support of Classic C programming in somewhat different styles. The syntax of C and C++ are very similar yet both of them belongs to a different programming paradigm. C is procedural whereas C++ is object-oriented.

C is a widely used programming language and a foundational technology for contemporary computing [13]. It is one of a favorite choice for introductory programming [14], and suitable for engineering applications. Since its origin thousands of programs have been written in it. Even after the development of many other languages, the C language is still popular.

It is widely recognized that object-oriented paradigm can enhance code reusability and maintenance. Object-oriented programming language, typically the C++ has been replacing procedural languages in several domains. C++ is a general-purpose programming language designed by Bjarne Stroustrup. It is frequently used when performance and resources are more important [15]. Handling of memory at low-level is extensively supported by C++.

As the area of computer algorithms progressed, many challenges are being presented in the selection of appropriate programming language for the efficient implementation of algorithms. The selection of appropriate language is indispensable both from technical and educational aspects. In this article a novel approach is presented that compared C and C++ by evaluating the implementation of conventional computer algorithms which are offered during the course on data structures and algorithms. To the best of our study, no analysis of such form has been reported for C and C++.

The article is organized as follows. Section 2 presents the previous work on evaluation of C and C++. Research design and results are described in section 3. Discussion is included in section 4 and followed by a conclusion.

II. Literature REview

Increasing the performance of computer programs has remained an active area of research. Both compiler optimization and hardware architecture related efforts have been made to improve performance of programs. Calder et al. [16] analyzed the behavior differences of C and C++ by examining the optimization techniques. The study also recognized the behavioral attributes of C++ programs that suggested optimization that would be functional in those programs. The results descried that C++ programs are significantly different than C programs.

Weixing et al. [17], analyzed the corpus of C and C++ on ARM7TDMI by comparing the usage of instruction set through the dynamic behavioral measurement. The study was conducted on embedded processors and result described that the size of C corpus is smaller than C++ programs. The study also observed that the function size of C programs is larger than C++ programs. Similarly, more memory instructions and control transfers are identified in C++ programs.

Studies on comparisons of programming paradigms are not very new [18]. Among all paradigms, object-oriented and procedural paradigms and their languages are much studied [19]. In the same vein Myrtveit and Stensrud [20], analyzed C and C++ by evaluating their software development productivity. The study used the data from real software projects. The studied found no experimental evidence that C is less productive than C++.

Bhattacharya and Neamtiu [21], compared C and C++ by analyzing the effect of programming language on software quality and productivity. During study open software projects are investigated. The study revealed that C++ code is less complex and entails less effort to maintain the code. Similarly, C++ is less prone to errors. The study also identified that code bases are transitioning from C to C++.

Prechelt [22], analyzed the common programming languages including C and C++. The study analyzed the reliability and runtime performance. During the study, collection of requirements implemented in the same programs are compared. Results declared the C and C++ to be fast and memory efficient, but less reliable than other languages in a study.

Zhu et al. [23] conducted a statement frequency analysis on the corpus of C, C++ and Java code. A large corpus of source code is used during the study and more than fifty-four million lines of code are analyzed. The results described that statement use frequency in selected languages is similar.

IEEE Spectrum is a flagship magazine and website of Institute of Electrical and Electronics Engineers. IEEE ranked the contemporary programming languages according to their popularity. In IEEE Spectrum 2019 [24], C is ranked on third with the score of 94.4 whereas C++ is on fourth with a score of 87.5. The ranking is defined by synthesizing different software metrics from different sources, including Google search, GitHub, IEEE Xplore Digital Library, Twitter, Stack Overflow and Dice.com.

TIOBE Company measures the popularity of programming languages by creating and maintaining the programming index. Queries passed to different search engines are used to calculate the index of programming languages. TIOBE also provides a cumulative rank of programming languages over different years (Fig. 1).

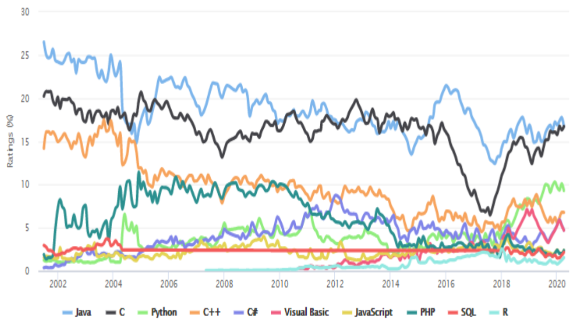

Figure 1. TIOBE Programming Community Index [25]

In latest TIOBE programming community index, C is top-ranked, whereas C++ is on fourth position.

III. Design & method

The study aims to examine the complexity involved in the implementation of conventional computer algorithms in C and C++. In order to accomplish the desired objective of the article, the following research methodology (Fig. 2) is defined.

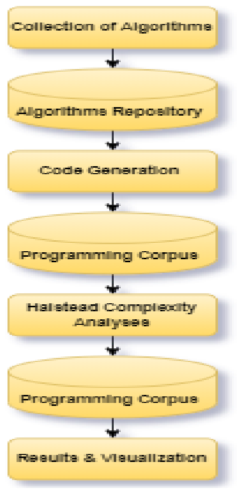

Figure 2. Research Methodology

As a part of the study, 225 algorithms are selected from online sources and books. The algorithms which being are offered in the courses of data structures and algorithms are selected for the study. These algorithms are also covered in elementary courses on computer programming. The detail of selected algorithms and topic coverage is shown in Fig. 3.

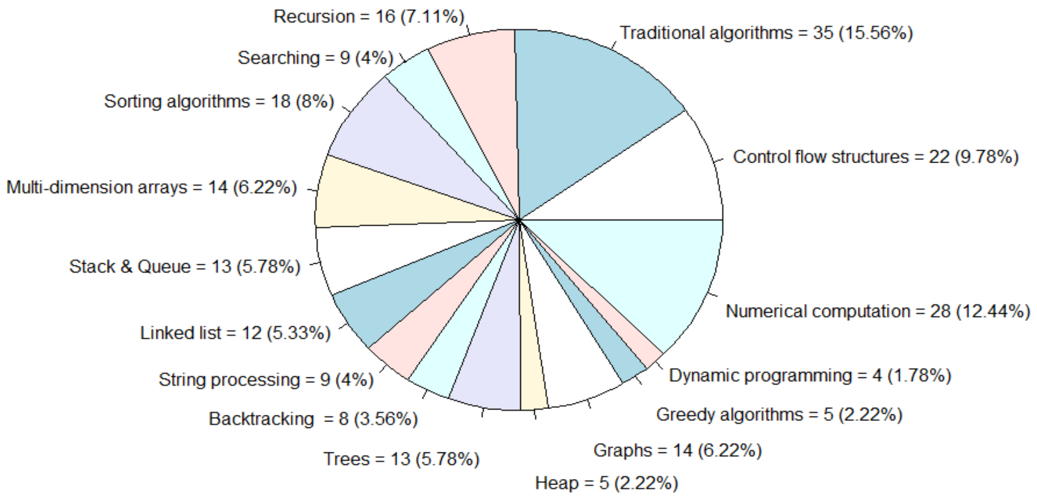

Figure 3. Detail of Selected Algorithms

.

For several algorithms the equivalent high-level codes of C and C++ were already present. However, for remaining algorithms the high-level code generator [26] was used that constituted the source code of selected algorithms. After elementary preprocessing of programming corpus, the programs were analyzed with Halstead complexity metrics which is an important technique to measure the complexity of program code [27, 28].

Halstead complexity is a suite of software metrics which is frequently used in automatic software complexity tool [29, 30]. There are many software quantification metrics, but Halstead complexity was chosen for this study because it provides several ways for analyzing program complexity in terms of difficulty, effort, time, and bugs.

The Halstead complexity was checked with Metric tool which is freely available on SourceForge. During the analysis of programming corpus, primitive attributes of programs was collected and results are shown in Table I.

Table I. PRIMITIVE ATTRIBUTES OF PROGRAMS

Attribute	Language	Mean	Median	Variance	Min	Max	Range	Total	Kurtosis
Operators	C	77.11	72.00	1028.49	26.00	225.00	199.00	17350	3.18
	C++	79.24	74.00	998.92	29.00	230.00	201.00	17828	3.68
Distinct Operators	C	25.11	25.00	15.69	13.00	35.00	22.00	5649	-0.30
	C++	28.49	29.00	15.00	18.00	39.00	21.00	6411	0.06
Operands	C	40.79	36.00	475.85	9.00	159.00	150.00	9178	5.68
	C++	40.93	36.00	475.08	8.00	157.00	149.00	9210	5.77
Distinct Operands	C	13.00	12.00	20.14	5.00	33.00	28.00	2926	1.92
	C++	13.17	12.00	23.11	1.00	42.00	41.00	2963	4.92
Program Vocabulary	C	38.14	38.00	54.29	24.00	63.00	39.00	8582	0.00
	C++	41.66	42.00	56.39	24.00	80.00	56.00	9374	2.21
Program Length	C	117.90	106.00	2847.52	35.00	374.00	339.00	26528	4.20
	C++	120.17	111.00	2814.65	37.00	374.00	337.00	27038	4.52
Estimated Program Length	C	166.38	163.34	1220.09	98.02	313.67	215.65	37445.60	0.65
	C++	188.07	185.26	1480.92	109.72	425.90	316.18	42384.50	5.56
Volume	C	627.72	567.90	97985.04	162.54	2142.91	1980.37	141236.51	3.79
	C++	653.81	593.15	100626.72	73.92	2209.18	2135.26	147107.32	4.22

The primitive attributes of programming corpus were identified through lexical analysis that recognized the lexical elements of source programs, categorized them as operators or operands. The frequencies of these operators and operands were used to calculate the program vocabulary, length, estimated length and volume. The primitive attributes were merely computed since these are used in determining the difficulty, effort, time and bugs. After the computation of primitive attributes, the main measures of programming corpus were calculated and descriptive statistics are shown in Table II.

Table II. CALCULATED MEASURES OF PROGRAMMING CORPUS

Measures	Language	Mean	Median	Total	Min	Max	Range	Skewness	Kurtosis
Difficulty	C	22.74	20.50	5115.50	4.50	79.50	75.00	1.95	5.89
	C++	20.47	18.00	4605.00	4.00	78.50	74.50	1.90	5.77
Effort	C	16118.60	9845.87	3626684.97	731.41	156036.63	155305.22	3.93	21.47
	C++	16775.98	10485.36	3774594.92	695.67	159060.75	158365.08	3.90	20.34
Time	C	896.22	546.99	201649.19	40.63	8668.70	8628.07	3.94	21.49
	C++	931.85	582.52	209666.37	38.65	8836.71	8798.06	3.90	20.34
Bugs	C	0.19	0.15	43.14	0.03	0.97	0.94	2.56	9.90
	C++	0.21	0.17	44.40	0.03	0.98	0.95	2.32	7.43

The difficulty of implementing conventional computer algorithms in C is higher than the C++. However, C is better in respect of time, effort and bugs. For better illustration, the calculated measures are shown with bean plots (Fig. 4).

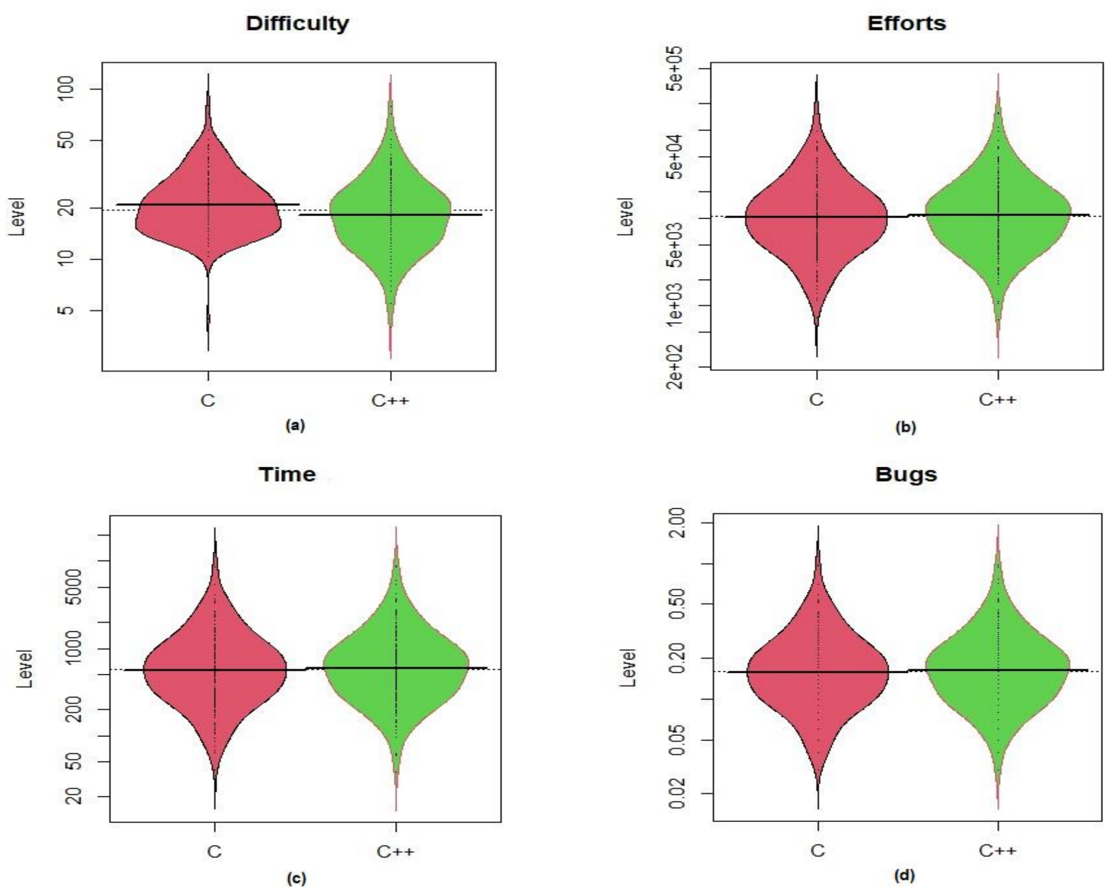

Figure 4. Bean Plots of Calculated Attributes

The interquartile range and median depicted in boxplots descried that difficulty of implementing algorithms in C++ is lower than C. However, C requires less effort than C++ to implement the underlying algorithms. Likewise, C requires less time and bugs than the C++. The statistical tests are conducted with SPSS 25 and for all tests a common threshold of 0.05 was selected. The results are initially evaluated for normality with Shapiro–Wilk test and Kolmogorov–Smirnov test. Kolmogorov–Smirnov is a test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution, or to compare two samples. The Shapiro-Wilk test is a commonly used statistical technique for determining if a continuous variable follows a normal distribution. The results obtained with normality tests are given in Table III.

Table III. RESULT OF NORMALITY TESTS

Measures	**Language**	**Shapiro-Wilk**			**Kolmogorov-Smirnov**		
		Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Difficulty	C	0.84	225.00	< 0.05	0.15	225.00	< 0.05
	C++	0.86	225.00	< 0.05	0.12	225.00	< 0.05
Effort	C	0.61	225.00	< 0.05	0.22	225.00	< 0.05
	C++	0.61	225.00	< 0.05	0.22	225.00	< 0.05
Time	C	0.61	225.00	< 0.05	0.22	225.00	< 0.05
	C++	0.61	225.00	< 0.05	0.22	225.00	< 0.05
Bugs	C	0.78	225.00	< 0.05	0.16	225.00	< 0.05
	C++	0.79	166.00	0.00	0.17	166.00	0.00

Different sorts of algorithms are selected for study, so their equivalent codes in C and C++ varies in size and structure. Consequently, the lexical specification of programs in the corpus are extremely diverse and resultantly the analyzed attributes are not normally distributed. Non-normality was observed in the calculated difficulty, effort, time and bugs. So, the non-parametric test was applied to identify the statistical differences between C and C++ and results are exhibited in Table IV.

Table IV. RESULTS OF U-TEST

Parameter	Language	Mean Ranks	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)
Difficulty	C	244.651	21003.500	46428.500	-3.125	0.002
	C++	206.349				
Effort	C	221.607	24436.500	49861.500	-0.635	0.525
	C++	229.393				
Time	C	221.767	24472.500	49897.500	-0.609	0.543
	C++	229.233				
Bugs	C	189.740	17266.500	42691.500	-1.276	0.202
	C++	204.485				

The Mann-Whitney U test delineated that difference between C and C++ in respect of effort, time and bugs are statistically significant. Though, in view of difficulty no statistical difference was observed between the C and C++.

IV. Discussion

The traditional ways of living and working have been altered by information and communication technologies [31]. The backbone of information and communication technology is programming. Programming has always been exclusive, as it has been associated with computer professionals and IT experts [32]. There are hundreds of programming language but C and C++ are the most popular and widely studied programming languages.

The work presented in this article examined the 225 conventional computer algorithms by analyzing their implementation in C and C++. Halstead complexity metrics is used to analyze the implementation of collected algorithms. In C corpus, 17350 operators are identified, whereas 17828 in C++. The slight difference of 2.72% narrates that C required less operators than C++. Likewise, 5649 distinct operators are identified in C corpus, whereas 6411 in C++ and this defines a difference of 12.64% that evince the aptness of C.

In C corpus 9178 operands are found in which 2926 operands were unique. Correspondingly, in C++ corpus 9210 operands are recognized in which 2963 were unique. This signifies that implementation of conventional algorithms in C requires less operands than C++.

The average score of program length for C corpus (38.14) is lower than the average score of C++ (41.66). Likewise, the estimated program length of the C corpus (166.38) is lower than the average score of C++ (188.07). So, from the perspectives of program length and estimated program length, C is much better than C++.

The volume in Halstead complexity metrics represents the number of mental comparisons required to develop a program. The mean score of volume for C corpus is 627.72 which is lesser than the score of C++ (653.81). The difference between the volume of C and C++ is about 4.1%, which explicate that C requires fewer mental comparisons than C++ for the implementation of the conventional computer algorithm.

The notion of difficulty in Halstead complexity represents the hardness of a program to write or understand. The mean score of difficulty for C corpus is 244.65 which is higher than the mean score of C++ (206.35), that suggests that C++ involves lower difficulty in the implementation of the algorithm. Withal the Mann-Whitney U test conducted on difficulty of programming corpus identified a significant difference between C and C++.

The effort in Halstead complexity metrics represents the elementary mental discriminations required to generate a program. The mean score of calculated effort for C corpus is 16118.60 while 16775.98 in C++ corpus which expound that C is finer than C++. Howbeit, the Mann-Whitney U test conducted on calculated effort described that difference between C and C++ is not statistically significant.

The mean score of time of C corpus is 896.22 whereas 931.85 for C++. Similarly, the cumulative time of C corpus is 201649.19 while 209666.37 for C++. So, C is found more effectual than C++ in the matter of time required to implement the conventional algorithms. Though, the Mann-Whitney U test conducted on calculated time described that difference between C and C++ is not statistically significant.

The mean score of delivered bugs for C corpus is 43.14 and 44.39 for the corpus of C++. The cumulative delivered bugs for C corpus are 43.14 whereas 44.39 for C++. The Mann-Whitney U test conducted on delivered bugs described that difference between C and C++ is not statistically significant yet the implementation of algorithms in C++ involves more bugs than C.

All in all, the results stated that for the implementation of conventional algorithms, C involves less effort, time and bugs whereas lower difficulty is observed in C++.

In statistical terms, the size of the corpus is relatively small, similarly the size of analyzed program is quite compact. So, on the basis of presented results a definite claim about the volume, difficulty, effort and time about C and C++ cannot be defined and the results are subjected to the considered algorithms and their implementation.

V. Conclusion

The learning of algorithms is essential for computer professionals because it is the core component of computer science. Computer algorithms are implemented as computer programs and thereby selection of appropriate language for an efficient implementation of the algorithm is a challenging task. In this article, C and C++ are compared by examining the implementation of conventional algorithms of computer science. The study suggests that despite of procedural paradigm, C involves lower effort to implement the conventional algorithms. Similarly, less time is required in C than C++ to implement the algorithms. Likewise, C is better than C++ in respect of delivered bugs. However, the C++ entails less difficulty while the implementation of conventional algorithms. The study implicitly suggests that despite of procedural paradigm, the C language is comparable to C++ and even in several measure it is more effective for the implementation of conventional algorithms. Topics for future work include i) examining the same problem on large sample of programs ii) use of data from real software projects iii) incorporating other software metrics like Chidamber & Kemerer metrics and cognitive complexity metrics for more kinds of analyses iv) comparative analysis of the implementation of bioinformatic algorithms in topical programming languages.

Acknowledgment

The author owes a great debt to Muhammad Tahaam for supporting the study by creating the programming corpus. This study would not be possible without his help. Special thanks to Muhammad Aayaan for valuable discussion, motivation and encouragement.

References

[1] S. B. Fee, A. M. Holland-Minkley, and T. E. Lombardi, "Re-envisioning Computing Across Disciplines,", Springer: New Directions for Computing Education, pp. 1-11, 2017.

[2] https://www.bls.gov/ooh/computer-and-information-technology/home.htm (Last Access: 8th May, 2020).

[3] V. Karavirta, and C. A. Shaffer, "Creating Engaging Online Learning Material with the JSAV JavaScript Algorithm Visualization Library," IEEE Transactions on Learning Technologies, vol. 9, no. 2, pp. 171-183, 2016.

[4] P. Moraes, and L. Teixeira, "Willow: A Tool for Interactive Programming Visualization to Help in the Data Structures and Algorithms Teaching-Learning Process," Proc. XXXIII Brazilian Symposium on Software Engineering, pp. 553-558, 2019.

[5] S. Combéfis, S. A. Barry, M. Crappe, M. David, G. D. Moffarts, H. Hachez, and J. Kessels, "Learning and Teaching Algorithm Design and Optimisation using Contests Tasks," Olympiads in Informatics, vol. 11, pp. 19–28, 2017.

[6] L. Végh, "JavaScript Library for Developing Interactive Micro-Level Animations for Teaching and Learning Algorithms on One-Dimensional Arrays," Acta Didactica Napocensia, vol. 9, no. 2, pp. 23-32, 2016.

[7] T. M. Celinski, B. A. Dijkstra, L. G. Ribeiro, M. A. de Souza, and V. G. Celinski, "Development of Learning Objects and their Application in Teaching and Learning Data Structures and their Algorithms," Iberoamerican Journal of Applied Computing, vol. 7, no. 2, pp. 23-32, 2017.

[8] A. T. Avancena, A. Nishihara, and C. Kondo, "Developing an Algorithm Learning Tool for High School Introductory Computer Science," Education Research International, pp. 1-11, 2015.

[9] R. A. Nathasya, O. Karnalim, and M. Ayub, "Integrating Program and Algorithm Visualisation for Learning Data Structure Implementation", Egyptian Informatics Journal, vol. 20, no. 3, pp. 193-204, 2019.

[10] L. Manelli, "Implementation of Algorithms in the C Programming Language," In: Introducing Algorithms in C. Apress, Berkeley, 2020.

[11] A. Laaksonen, "A Competitive Programming Approach to a University Introductory Algorithms Course," Olympiads in Informatics, vol. 11, pp. 87-92, 2017.

[12] W. Debabi, and T. Bensebaa, Using Serious Game to enhance algorithmic learning and teaching, "Journal of e-Learning and Knowledge Society," vol. 12, no. 2, pp. 127-140, 2016.

[13] A. Moss, R. Schluntz, and P. A. Buhr "C∀: Adding modern programming language features to C," Practice and Experience, vol. 48, pp. 2111–2146, 2018.

[14] M. S. Naveed, M. Sarim, and A. Nadeem, "C in CS1: Snags and Viable Solution," Mehran University Research Journal of Engineering & Technology, vol. 37, no. 1, pp. 1-14, 2018.

[15] C. Sanderson, and R. Curtin, "Armadillo: a template-based C++ library for linear algebra," Journal of Open Source Software, vol. 1, no. 2:26, 2016.

[16] B. Calder, D. Grunwald, and B. Zorn, "Quantifying Behavioral Differences Between C and C++ Programs," Journal of Programming languages, vol. 2, no. 4, pp. 313-351, 1994.

[17] J. Weixing, S. Feng, and Q. Baojun, "Execution Characteristics of C++ and C Programs on Embedded Processor ARM7TDMI," Proc. 5th WSEAS International Conference on Applied Computer Science, Hangzhou, pp. 1033-1038, 2006.

[18] M. E. Hayder, C. S. Ierotheou, and D. E. Keyes, "Three Parallel Programming Paradigms: Comparisons on an Archetypal PDE computation," Parallel and Distributed Computing Practices, vol. 2, pp. 35-53, 2000.

[19] G. White, and M. Sivitanides, "Cognitive differences between Procedural Programming and Object-oriented Programming", Information Technology and Management, vol. 6, no. 4, pp. 333-350, 2005.

[20] I. Myrtveit, and E. Stensrud, "An Empirical Study of Software Development Productivity in C and C++," Proc. of Norsk Informatikkonferanse, 2008.

[21] P. Bhattacharya and I. Neamtiu, "Assessing Programming Language Impact on Development and Maintenance: A Study on C and C++," Proc. 33rd International Conference on Software Engineering, pp. 171-180, 2011.

[22] L. Prechelt, "An Empirical Comparison of Seven Programming Languages," Computer, vol. 33, no. 10, pp. 23-29, 2000.

[23] X. Zhu, E. J. Whitehead, C. Sadowski, and Q. Song, "An Analysis of Programming Language Statement Frequency in C, C++, and Java Source code," SOFTWARE: Practice and Experience, vol. 45, pp. 1479-1495, 2015.

[24] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019 (Last Access: 27th August, 2021).

[25] https://www.tiobe.com/tiobe-index/ (Last Access: 1st October, 2021).

[26] 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.

[27] M. S. Al-Batah, N. Alhindawi, R. Malkawi, and A. A. Zuraiqi, "Hybrid Technique for Complexity Analysis for Java Code," International Journal of Software Innovation, vol. 7, no. 3, pp. 118-133, 2019.

[28] Q. Liping, L, Jing, and S. Yaqing, "Research on the Complexity Measurement Technology of Software Structure Based on AST," University Politehnica of Bucharest Scientific Bulletin Series C-Electrical Engineering and Computer Science, vol. 80, no. 1, pp. 39-50, 2018.

[29] B. A. Sanusi, S. O. Olabiyisi, A . O. Afolabi, and A. O. Olowoye, "Development of an Enhanced Automated Software Complexity Measurement System," Journal of Advances in Computational Intelligence Theory, vol. 1, no. 3, pp. 1-11, 2019.

[30] M. S. Naveed, "Comparison of C++ and Java in Implementing Introductory Programming Algorithms," Quest Research Journal, vol. 19, no.1, pp. 95-103, 2021.

[31] S. Chandio, M. S. A. Seman, S. Samsuri, A. Kanwal, and A. Shah, "Assessing ICT Implementation and Acceptance at Public Sector Universities in Pakistan," University of Sindh Journal of Information and Communication Technology, vol. 2, no. 1, pp. 52-56, 2018.

[32] M. A. Rahman, R. S. U. Riaz, and T. Rana, "PFE: A Visual Programming Frame Work for Teaching Programming to Dummies or beginners," University of Sindh Journal of Information and Communication Technology, vol. 4, no. 3, pp. 194-198, 2020.