Proceedings of the Pakistan Academy of Sciences: A: Physical and Computational Sciences 59(1): 59-70 (2022)

ISSN (Print): 2518-4245; ISSN (Online): 2518-4253
https://doi.org/10.53560/PPASA(59-1)710

Pakistan Academy of Sciences

Research Article

Two-Phase CS0 for Introductory Programming

Muhammad Shumail Naveed[1*], and Muhammad Sarim[2]

[1]Department of Computer Science & Information Technology, University of Balochistan, Quetta, Pakistan
[2]Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Karachi, Pakistan

Abstract: Computer programming is the heart of computer science and thereby an important skill of the students. However, comprehending programming is extremely hard and introductory courses on programming are notorious to cause issues and challenges for learners, which affect their motivation and consequently cause high dropouts and low retention. This paper introduced CS0 as a precursor programming course to teach beginners, the fundamental notions of programming before the first course on programming. The proposed CS0 is grounded on two-phase learning strategy and equipped with a collaboration strategy. The results of the initial evaluation of proposed course are reasonably encouraging to motivate and prepare the novices for the first course on programming. The statistical significance of a proposed course is observed in improving the academic outcomes of novices in a first programming course. The normality of data was checked using the Kolmogorov-Smirnov and Shapiro-Wilk tests. The findings were analyzed with one-way ANOVA test and Kruskal–Wallis H test, indicating that the suggested course is statistically significant in enhancing the academic achievements of beginners in their first programming course.

Keywords: Programming, introductory programming, pair programming, CS0, motivation.

1. INTRODUCTION

MComputer Programming is one of the important abilities of computer science students [1, 2]. However, comprehending programming is extremely difficult for novice students [3,4], and therefore introductory courses on programming are generally very difficult. Appropriate problem-solving skills are necessary for programming [5], but the syntax and denotations of computer languages are mostly hard for novices [6]. It is commonly believed that complicated syntax, the span of time to construct a computer program and isolation during coding and learning are the prime factors for the sophistication to learn and understand the programming [7]. In [8], it is opined that the inherent difficulty of the programming and a lack of motivation in novices are the core reasons of their failure.

The first course on computer programming is usually called a CS1 [9]. The first programming language is immensely important for students since it forms the tone for all the successive classes of computer education [10]. However, the requirement of problem-solving abilities, complicated syntax, strange semantic and dense logic disturbs students' learning curve in a first course on programming [11]. These actualities have caused novice students to scare from programming and form wrong judgment regarding their education in programming [12].

An initial disappointment of learning fundamental concepts affects the confidence and increase dropout of students in CS1. Educators and mentors are being asked to determine the beneficial intermingle of technology and pedagogy to aid students in CS1 [13]. Different studies are organized to control these problems yet the failure rates and dropout in introductory courses on programming are still very high [8, 14, 15, 16, 17].

Received: April 2020; Accepted: March 2022
*Corresponding Author: Muhammad Shumail Naveed <mshumailn@gmail.com>

Several studies have attempted to analyze and determine the aspects that predict success of students in a first course on programming. In [18], it is argued that the performance of students is strongly affected by their learning strategy. Motivation is extremely important for successful instruction [19]. Settle et al. [14], identified the positive correlation between the motivation and success to learn the programming.

Anderson et al. [20] described that learning in programming is affected by students' low motivation. Likwise, Nikula et al. [21] argued that lack of motivation increased the failure rates in a first programming course. Similarly, it is widely believed and reported that prior experience of programming strongly affects students' outcome in the introductory courses of programming [22].

The rest article is organized as follows. Related work is presented in the next section. Section 3 described the research approach and methods. Section 4 includes the evaluation and discussed the initial results. Finally, the conclusion is described in section 5.

2. RELATED WORK

Several programming environments and approaches have been introduced to control the difficulties of introductory programming.

Visual programming language is a programming language that helps students in learning programming concepts by allowing figures and graphical objects to write the programs [23]. These languages allow the drag-and-drop features to develop a program. Scratch is a language developed to teach programming to novice students. It supports to develop interactive stories, computer animation, interactive stories, and other multimedia projects. Blockly is a client-side library developed by Google for JavaScript. It eases the programming by supporting drag-and-drop of components to develop the programs. Sub-clauses of instructions and placeholders for identifiers are supported in Blockly [24]. With Blockly novices concentrating on algorithm semantics and relaxed from the rigid syntax of a language.

Seraj et al [25], compared and analysed the impact of Scratch and Blockly in increasing the programming capabilities of students. The study reported the significance of Scratch over the Blockly. However, Blockly is found useful in increasing the interest of students in future programming.

Alice is a programming language for educational purpose. It follows an object-oriented style of programming and developed by Carnegie Mellon University. Alice supports beginners in learning elementary concepts like variables, arrays, objects & classes, and recursionbeginners in elementary concepts like variables, arrays, objects & classes, recursion, and inheritance. In [26], the four-component instructional model and Alice are proposed for initial programming learning and results suggested that the use of Alice in programming learning has the positive effects.

The transition from visual to textual languages is another main concern of introductory programming [27]. In a landmark study [28], Alice is critically analysed. The study described that drag and drop feature in Alice separates the learning of syntax from comprehending the semantics. It is also identified that Alice rises the confidence, but this confidence vanished when students shifted from Alice to the traditional text-based languages. It is widely believed that visual tools help novice students in the initial stages, but these tools should not be assumed as a universal panacea [29].

Bakar et al. [30], proposed a VJava module which includes submodules comprised of multiple sections that cover the elementary areas of programming. MJava library is included in a module that permits the computer programs to construct visual outputs. The contents, design and usability aspects of VJava are corroborated with the experts' validation process.

In [31], the impacts of exercises-only approach with lectures combined with exercises in introductory programming was compared. The study reported that both methodologies are fruitful in introductory programming, but the exercise-only approach is more effective than the other approach. The study also identified that prior programming knowledge and grade expectation are important antecedents of learning performance in programming.

Malik et al. [32], developed an application called PROBSOL to increase the skills of beginners in the introductory courses of programming. PROBSOL is centered on pseudo code techniques and supports web application and mobile app which is available from Google store. The application of PROBSOL showed that their use promotes students' engagement, logic and problem solving skills. However, the mobile based framework is recommended by novices over the web based dialect.

The CS0 is a doable methodology to prepare the novices for the introductory courses in programming. The CS0 courses provide a system for novices to get a background which is essential for CS1 [33]. The CS0 course is offered as a pre-programming course with no explicit prerequisites. The majority of CS0 courses use programming tools to introduce the novice students to the introductory concepts of computer programming.

In [34], a CS0 course for students has introduced. The course embraces GameMaker and C# to introduce programming and covers elementary topics like sequences, operators, conditions, loops, arrays and functions. The prime goal was to attract the students and provide them with an elementary knowledge of computer programming. The views received from learners reported that pedagogy is helpful to prepare students for real programming.

Dyne and Braun [35] define and evaluated a course that supports methods and techniques for resolving problems and critical reasoning to aid beginners with the essentials required for successfully completing CS1. The preliminary results of inducting the course are very reasonable in improving students' academic outcomes.

Uludag proposed a course that used web application development environment, Lego Mindstorms with block-based language and aimed to support the students in programming courses and initially found very effective in learning the introductory programming [36].

Haungs et al. [37] introduced a CS0 course that covers multiple tracks that students can select (for example, gaming, mobile apps and music, robotics). This allows the beginners to comprehend the fundamental of programming and teamwork.

Gudmundsen et al. [38], introduced a CS0 course for introductory programming. The course covers Visual Logic in first stage and then switched to Python. The initial results are reasonably encouraging in increasing the performance and commitment of students.

Naveed and Sarim [39] defined an algorithmic style programming language and introduced before the CS1 to illuminate the beginners about the fundamental concepts of novice programming. The language is very similar to natural language based algorithmic description. It also supports the generation of equivalent source programs in Python, Java, C++ and C from the input source program. The initial application of LPL is very encouraging in increasing students' performance and retention rate.

Dawson et al. [40], developed a CS0.5 course to improve students' attitude, satisfaction and achievement in CS1. The course was offered as a unified and reduced-face interval course. The students' were assigned pre-class work comprising of reading material and problem solving using new notions. Students' feedback revealed that the designed course successfully improved students' outcomes. Similarly, Wood et al. [41] conducted a very comprehensive study by comparing different pedagogical approaches of precursor courses and reported that CS0 has a productive impact on students' attitudes and performance.

3. DESIGN AND METHODS

The novice students without some previous understanding of programming need to equipped with methods to resolve the problem and employ a computer programming language as a framework to attempt the problem concurrently [42]. These manifold preconditions naturally increase the cognitive load [43] and for that reason, the first course on programming frequently necessities to introduce platform for novice students.

In this research, a CS0 course for novice students of CS1 has proposed and its effectiveness is investigated as it is widely accepted that CS0 course

is a crucial element of education [44]. Principally, the proposed CS0 does not refute the prevailing precursor courses, but solely mingle their coherent elements in an innovative fashion. The central aim of the proposed course is to support novice students by providing prior knowledge of programming and improve their academic outcomes in a first course on programming. The proposed CS0 is based on two-phase learning and equipped with a collaboration strategy. In two-phase learning the fundamentals of programming are introduced to students in two stages.

In the first phase of the proposed course, the novices are introduced to elementary theories of foundational programming without compelling them to concentrate on the rigid structure of languages because adjourning actual in introductory level would provide fair learning opportunity to students [45]. This phase merely presents the plain concepts of novice programming. So in the first phase, the fundamental concepts of programming are introduced through the visual language which is unanimously recognized as a feasible approach to engage and support the students.

The visual language offered in the first phase of a proposed CS0 presents the basics of programming by representing the difficult theories with graphical support and fostering their enthusiasm but presenting no actual familiarity with concrete programming. Klassen [46] defined that graphical language like Alice in CS0 is not enough to prepare the students for CS1. So CS0 must involve actual programming and should follow a meticulous approach.

The second phase of proposed CS0 presents the elementary concepts with a text-oriented language. The text-based platform is recognized as an operational podium for novice students to explicitly deal with actual computer programming [47]. In [48], it is identified that the algorithmic approach is desirable to foundational programming. So, the second phase is designed to cover the programming with easy, clear and self-explanatory statements like in Python.

Students commonly encounter nervousness while learning programming; so in a proposed CS0, the novices are persuaded to work in a pair as it is positive for students and may increase their academic outcomes. Pair programming has been found very effective in introductory programming courses [49, 50] and in other academic courses [51].

The extensive examination of the literature revealed that several ways have been introduced to address the difficulties of introductory programming. Table 1 compares some of the most notable methods for introductory programming courses.

Several solutions have been suggested to overcome the challenges of beginning programming, according to the discussion in Table 1. Each method provided is focused on a certain domain and programming language. The significance of beginning programming cannot be overlooked, as it is a requirement for pursuing higher level courses [59].

4. RESULTS

In order to ascertain the actual effectiveness of proposed CS0 course a small study is conducted. The study analyzed the impact of using the two-phase CS0 course in improving the academic achievement of novices in the first course on programming.

During the study 148 students have participated which are randomly clustered into four groups. To the first group, no CS0 course is offered before offering the first course on programming. Python is introduced as a CS0 to a second group, Blockly is offered to a third group, whereas the proposed CS0 is offered to the fourth group of a study. Blockly is used in the first phase, Blockly is a visual block language that enables the rapid creation of programs. In introductory programming environments, block-based programming has proven popular for teaching programming to young students. Blockly has been recognized as a valuable tool for improving students' coding skills and changing their attitudes about programming. Python is used in the second phase of a proposed CS0 course

The study was performed on the undergraduate students of computer science in 2018 and the subjects voluntarily contributed in the study. All the learners declare that they have no previous acquaintance of actual programming.
Later, a course on programming is offered as a CS1

Table 1. Comparison of Programming Environments & Approaches

Sr. No.	Environment + Approaches	Merits & Demerits
1	Visual Languages	A visual programming language is one that allows to create programs using graphical elements and images. Visual languages increased learners' knowledge of basic programming principles while also increasing their self-efficacy [52,53]. In visual languages the learners are relieved from the burden of difficult syntax. When a learner transitions from a visual language to a real language with intricate syntax, their experience with visual languages frequently diminishes.
2	Learners Programming Language	The complexity of programming languages is exacerbated by the complicated and peculiar syntax of programming languages. A simple syntax-based programming language has been introduced that supports algorithmic type constructs for program development. The majority of them are still in the early stages of development and are only helpful in learning a limited number of programming languages [39].
3	Blended Learning	Blended learning is a form of teaching that blends conventional place-based classroom methods with online educational materials and chances for online participation. Several studies have sought to use blended skills to develop their students' performance in beginning programming courses [54]. Although the technique has proven to be quite efficient, it does necessitate regular hardware support for utilizing online teaching content, which is almost impossible in every educational system.
4	Didactic Strategies	To make programming easier for novices, several didactic strategies are introduced. The strategies often included approaches and techniques such as preceding courses, use of doodles, and pair programming [55]. These tactics are usually developed for a specific sort of programming course and a specific group of students or learning environment.
5	Intelligent Tutoring System	Intelligent tutoring is a smart learning method designed to improve student scores, pass rates, and teacher efficiency. Over multiple academic years, a set of technology-based instructional practices was adopted in an introductory programming course, and the data demonstrate that the grade distributions have significantly improved [56]. However, most of the experimental studies on intelligent tutoring systems are conducted on a small number of subjects. It is widely observed that supplemental assistance in intelligent tutoring simply increases learners' cognitive load.
6	Descriptive Errors Presentation	The difficulty of debugging source programs has an impact on a novice's ability to learn programming. In development environments, descriptive error messages are used to make debugging easier. However, it was discovered that the simplicity of error messages has a significant impact on debugging score, but there is no association between debugging and programming scores [57].
7	Gamification	Different studies have demonstrated the educational benefits of digital gaming. The adaptation of game characteristics to non-game environments is referred to as gamification. Various studies have demonstrated that when used correctly, gamification may establish a learning environment and lead to significant improvements in students' interest in programming. However, according to a study on the impact of gamification on programming, gamification does not improve students' performance while increase their engagement [58].

by a same team of instructors to all the groups of study and C language is covered during the course. In all groups the same teaching methodology, material and programming environments are followed.

After the completion of CS1 all subjects are internally evaluated and results are shown in Table 2.

High mean marks are observed in the fourth group (pass rate = 61%), followed by a third group (pass rate = 49%), third second group (pass rate= 41%) and first group (pass rate= 36%), respectively. The mean score of each group in a study can be analyzed with a mean plot shown in Fig. 1.

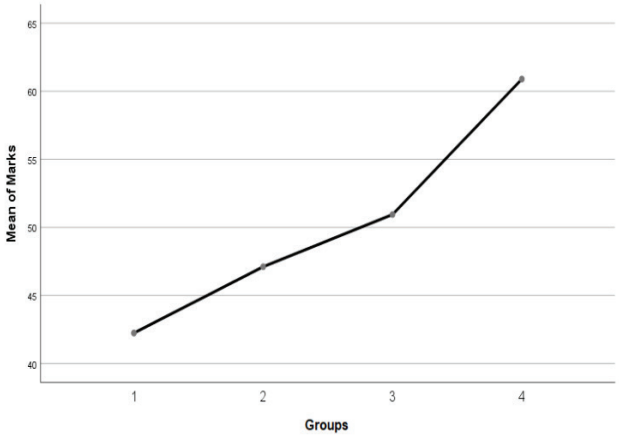

Fig 1. Mean plot of marks

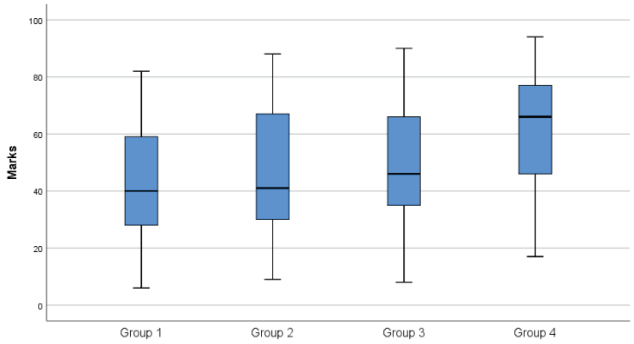

Fig 2. Box plot of marks

The mean plot shown in Fig. 1 illustrates how the mean of score varies across different groups in a study and the least mean score is observed in the first group and the highest mean score in the fourth group. Fig. 2 shows the boxplot of the marks obtained by the groups involved in the study.

The median, minimum and maximum marks illustrated in the box plot declare the fourth group as highest and the first group being the last in securing the marks in the first programming course and the second and third groups' lies among them.

Table 2. Score of students in CS1

Groups	Mean	Std. Deviation	Std. Error	Min	Max
1	42.24	20.55	3.38	6	82
2	47.11	21.29	3.50	9	88
3	50.95	20.76	3.41	8	90
4	60.89	20.62	3.39	17	94

For further analysis, the normality tests were conducted on the score of students and results are shown in Table 3.

Kolmogorov-Smirnov test indicates that the mark secured by all groups of study follows a normal distribution. Similarly, the Shapiro-Wilk test shows that secured scores of all the involved groups in a study follows a normal distribution.

For better illustration a normal probability (Q-Q) plot of the marks is shown Fig. 3 and depicted the distribution of data against the expected normal distribution.

Table 3. Normality test on the score of students in CS1

	Kolmogorov-Smirnov			Shapiro-Wilk		
Groups	Statistic	df	Sig.	Statistic	df	Sig.
1	0.090	37	.200	0.974	37	0.542
2	0.126	37	0.143	0.954	37	0.127
3	0.108	37	.200	0.975	37	0.573
4	0.138	37	0.071	0.954	37	0.125

A one-factor analysis of variance (ANOVA) was conducted on all groups of study and the results showed that there was a significant difference between study groups in a first programming course remembered at the $p < 0.05$ for the conditions $[(3,144) = 5.347, p = 0.002]$.

Concerning the students' retention, the study assessed their responses to the following questions: "In next semester semester, I am willing in another course on programming". Subjects replied on 5-item Likert scale and the results are illustrated in Fig. 4.

The study revealed that high interest in another course on programming is observed in those students who have attended a two-phase CS0 course before the CS1. For further analysis a Kruskal–Wallis H test is conducted on the feedback of subjects and results are shown in Table 4.

The fourth group is identified as top with highest mean value, followed by group two, three and one respectively. The significant differences

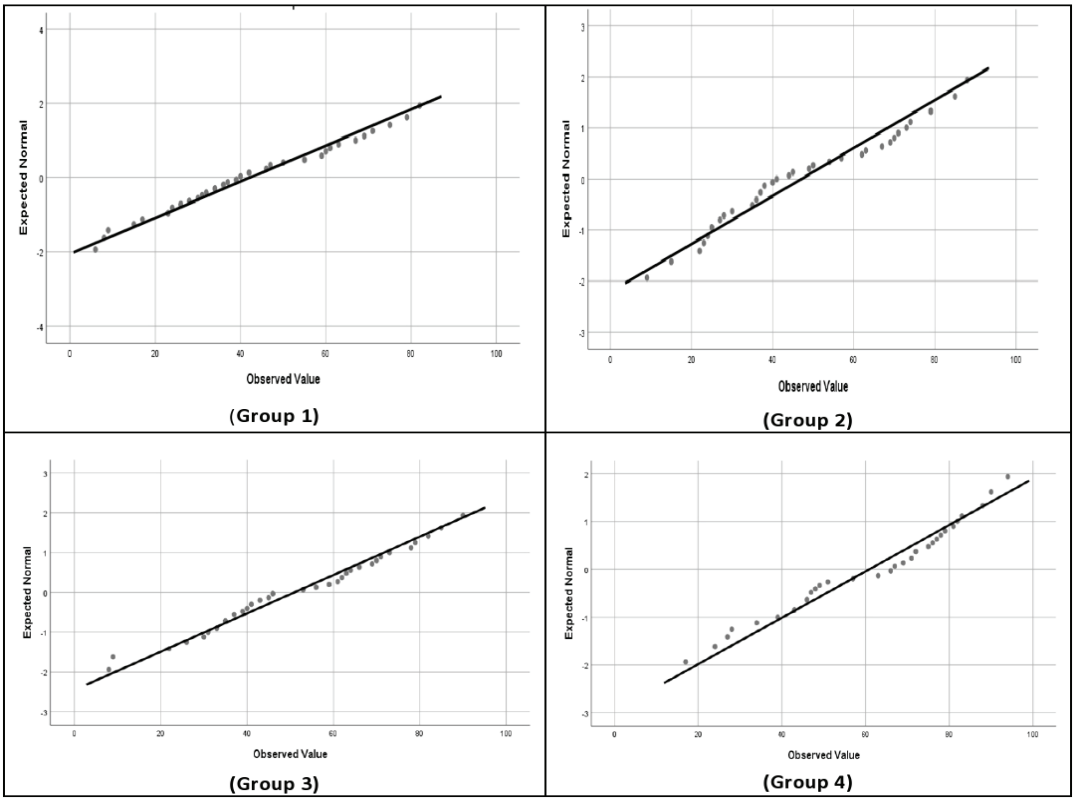

Fig 3. Normal Q-Q Plot of Groups

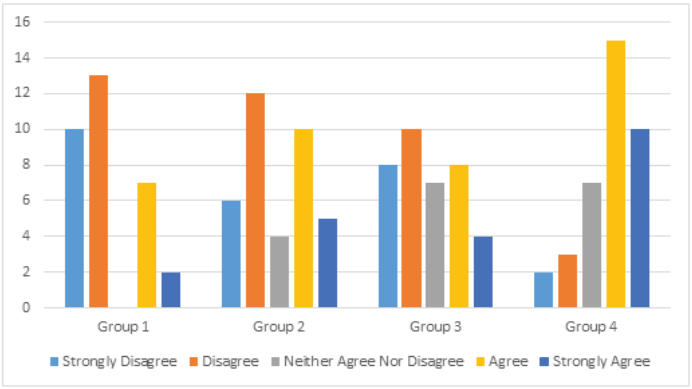

Fig 4. Students' willingness in another course on programming

Table 4. Ranks of groups in study

Group	Size	Mean Rank
1	37	57.70
2	37	72.86
3	37	67.76
4	37	99.68

Table 5. Comparison of learning strategies for CS1

S.No	Study	Method	Results
1	Seraj et al. [25]	Comparative analysis of Scratch and Blockly	Scratch is more helpful than Blockly, $F(1,22) = 28.02$, $p < 0.001$
2	Bakar et al. [30]	Expert validation Process.	VJava module is confirmed suitable, with score ranged from 3.67 to 5.00 on 5-point Likert scale.
3	Zhang et al. [31]	Comparative analysis of two teaching approaches	The exercise only approach is more useful than lectures combined with exercises in introductory programming, $df = 34$, $t = 2.320$, $p < 0.05$
4	Malik et al. 2019 [32]	Development and application of pseudocode-based application with two modes, web-based and mobile-based	3% of positive improvement in higher category as well as in medium achiever category
5	Panitz et al. [34]	Survey and feedback from students	GameMaker and C# based CS0 is effective in attracting and preparing students for programming
6	Naveed et al. [39]	Comparative analysis of different precursor courses	Textual programming language with the support of high-level source code generation is supportive to prepare novices for CS1
7	Dawson et al. [40]	Survey and appreciative inquiry	CS 0.5 is helpful to improve the students' outcome in CS1 and increased their personal interest: $t(515.0) = 8.96$, $p < 0.001$
8	Wood et al. [41]	Comparative study of pedagogical methods in CS0	CS0 course is helpful in CS1 yet no pedagogical method in CS0 is better than other methods, p-value = 0.449
9	Parham-Mocello et al. [45]	Comparative analysis of code-first versus using stories	Using stories is useful in introductory programming, especially for females, with 95% confidence, $\alpha <= .05$
10	Proposed CS0	Comparative analysis of different pattern of CS0 courses.	Two-phase CS0 is more helpful than single language based CS0 and also fruitful in preparing students for CS1, $F = 5.35$, $df = 3$, $p = .002$

(Kruskal-Wallis = 20.404, $p < .05$, $df = 3$) were found among the four groups of study.

5. DISCUSSIONS

An important aspect of introductory programming courses is the development of a good way of thinking [60]. This article presents a two-phase learning technique that is accompanied by a cooperation strategy.

These results suggested that the amalgamation of graphical and textual environment in CS0 is better thanusing a single environment-based CS0 course. Moreover, the two-phase CS0 course is more useful in improving the performance of students in CS1 as well as by increasing their retention in the other course of programming.

The results obtained from the initial evaluation of two-phase CS0 are further compared with other studies and the results are shown in Table 5.

The results illustrated in Table 4 opined that the proposed CS0 course is quite fruitful and supportive in increasing the academic outcomes of novices

in a first course on programming and therefore comparable with other topical strategies.

The novelty of proposed CS0 resides in an amalgamated approach that unified the two modes of programming systems in a single precursor course with an agile collaboration strategy to prepare the novices for the first course on programming. The statistical significance of initial evaluation suggests the viability of the proposed course. To our knowledge, none of any study has been conducted or at least reported on the subjects of Pakistan.

6. CONCLUSIONS

The first course on programming is infamously complex for novice students. These complexities are typically exhibited in the form of weak performance and low rates in retention. In this article two-phase CS0 is introduced to boost the academic achievements of novice students in a first course on programming.

The proposed course is initially evaluated and preliminary results suggested that the lack of previous knowledge of programming is one of a main reason behind the hardness of introductory programming. Likewise, a CS0 course that introduced elementary programming without forcing the students to understand the grammar of a specific programming language and then introduce the concepts of programming with a simple textual programming is a suitable methodology to prepare students for a first course on programming and ultimately improve their performance and retention. However, in current form, the study has several limitations. First, the sample size is not very large and the subjects in each group of the study are randomly selected. Second, the impact of CS0 is evaluated on the performance of students in CS1, but not in the subsequent programming courses. Third, the demographic parameters of students are not considered during the analysis.

7. ACKNOWLEDGEMENTS

The authors wish to thank all the students who participated in the study. Special thanks to Muhammad Tahaam and Muhammad Aayaan for their suggestions and support in improving the manuscript.

8. REFERENCES

1. J. O'Kelly, and J. P. Gibson. RoboCode & Problem-based Learning: A Non-prescriptive Approach to Teaching Programming. *ACM SIGCSE Bulletin* 38(3): 217-221 (2006).
2. M. V. McCracken., D. D. Almstrum., M. Guzdial., D. Hagan., Y. B. Kolikant., C. Laxer., L. Thomas., I. Utting, and T. Wilusz. A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First year CS Students. *ACM SIGCSE Bulletin* 33(4):125-180 (2001).
3. B. Sabitzer, and S. Pasterk. Brain-based programming continued: Effective teaching in programming courses. *Proceedings of IEEE Frontiers in Education Conference*: 1-6 (2014).
4. C. Ott. Decoding Feedback: Improving feedback practices for students in introductory programming courses. *PhD thesis*, University of Otago, Dunedin, New Zealand: (2014).
5. S. M. Taheri., M. Sasaki, and H. T. Ngetha. Evaluating the Effectiveness of Problem Solving Techniques and Tools in Programming. *Proceedings of Science and Information Conference:* 928-932 (2015).
6. A. Sen. Using Code Analysis Tool in Introductory Programming Class. *Issues in Information Systems* 15(1): 1-10 (2014).
7. A. Ali, and D. Smith. Teaching an Introductory Programming Language in a General Education Course. *Journal of Information Technology Education: Innovations in Practice* 13: 57-67 (2014).
8. G. Gárcia-Mateos, and J. L. Fernández-Alemán. A Course on Algorithms and Data structures using On-line Judging. *ACM SIGCSE Bulletin* 41(3):45-49 (2009).
9. M. Brown., C. Hu., C. Burch, and M. Nooner. CS0: Why, What, and How?: Panel Discussion. *Journal of Computing Sciences in Colleges* 25 (5):79-81 (2010).
10. K. Howell. First Computer Languages. *Journal of Computing Sciences in Colleges* 18(4):317-331 (2003).
11. A. Riker. Natural Language in Programming An English Syntax-based Approach for Reducing the Difficulty of First Programming Language Acquisition. *Master's thesis,* Department of Computer Science, Graduate School of Arts and Sciences, Brandeis University: (2010).
12. M. Konecki. Introductory Programming Education

for Visually Impaired. *International Journal of Research in Engineering and Technology* 3(17): 65-70 (2014).

13. K. Price, and S. Smith. Improving Student Performance in CS1. *Journal of Computing Sciences in Colleges* 30(2): 157-163 (2014).
14. A. Settle., A. Vihavainen, and J. Sorva. Three Views on Motivation and Programming. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*: 321-322 (2014).
15. A. Robins. Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education* 20(1): 37-71 (2010).
16. R. H. Sloan, and P. Troy. CS 0.5: A Better Approach to Introductory Computer Science for Majors. *ACM SIGCSE Bulletin* 40(1): 271-275 (2008).
17. A. Robins. Novice programmers and introductory programming, *S. A. Fincher & A. V. Robins (Eds.), the Cambridge Handbook of Computing Education Research. Cambridge, UK: Cambridge University Press*: 327-376 (2019).
18. P. Y. Amoako., K. A. Sarpong., J. K. Arthur, and C. Adjetey. Performance of Students in Computer Programming: Background, Field of Study and Learning Approach Paradigm. *International Journal of Computer Applications* 77(12): 17-21 (2013).
19. L. M. Serrano-Camara., M. Paredes-Velasco., C. Alcover, and J. A. Velazquez-Iturbide. An evaluation of students motivation in computer supported collaborative learning of programming concepts. *Computers in Human Behavio*r 31: 499-508 (2014).
20. R. Anderson., M. D. Ernst., R. Ordonez., P. Pham, and S. A. Wolfman. Introductory Programming Meets the Real World: Using Real Problems and Data in CS1. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education, pages*: 465-466 (2014).
21. U. Nikula., O. Gotel, and J. Kasurinen. A Motivation Guided Holistic Rehabilitation of the First Programming Course. *ACM Transactions on Computing Education* 11(4): 24:1-24:38 (2011).
22. A. Tafliovich., J. Campbell, and A. Petersen. A Student Perspective on Prior Experience in CS1. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*: 239-244 (2013).
23. K. Al-Tahat. The Impact of a 3D Visual Programming Tool on Students' Performance and Attitude in Computer Programming: A Case Study in Jordan. *Journal of Cases on Information Technology* 21(1): 52-64 (2019).
24. A. Marron., G. Weiss, and G. Wiener. A Decentralized Approach for Programming Interactive Applications with JavaScript and Blockly. *Proceedings of the 2nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions*: 59-70 (2012).
25. M. Seraj., E. Katterfeldt., K. Bub., S. Autexier, and R. Drechsler. Scratch and Google Blockly: How Girls' Programming Skills and Attitudes are Influenced. *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*: 1-10 (2019).
26. J. M. Costa, and G. L. Miranda. Using Alice Software with 4C-ID Model: Effects in Programming Knowledge and Logical Reasoning, *Informatics in Education* 18(1): 1-15 (2019).
27. Y. Matsuzawa., T. Ohata., M. Sugiura, and S. Saka. Language Migration in non-CS Introductory Programming Through Mutual Language Translation Environment. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education:* 185-190 (2015).
28. K. Powers., S. Ecott, and L. M. Hirshfield. Through the Looking Glass: Teaching CS0 with Alice. *ACM SIGCSE Bulletin* 39(1):213-217 (2007).
29. A. Stefik, and S. Siebert. An Empirical Investigation into Programming Language Syntax. *ACM Transactions on Computing Education* 13(4): 19:1-19:40 (2013).
30. M. A. Bakar., M. Mukhtar, and F. Khalid, The Development of a Visual Output Approach for Programming via the Application of Cognitive Load Theory and Constructivism. *International Journal of Advanced Computer Science and Applications* 10(11): 305-312 (2019).
31. X. Zhang., C. Zhang., T. F. Stafford, and Ping Zhang. Teaching Introductory Programming to IS Students: The Impact of Teaching Approaches on Learning Performance. *Journal of Information Systems Education* 24(2): 147-155 (2013).
32. S. I. Malik., R. Mathew., R. Al-Nuaimi., A. Al-Sideiri, and J. Coldwell-Neilson. Learning problem solving skills: Comparison of E-learning and M-learning in an introductory programming course. *Education and Information Technologie*s 24: 2779-2796 (2019).
33. C. Farrell. Predicting (and Creating) Success in CS1. *Issues in Information Systems* 7(1): 259-263 (2006).
34. M. Panitz., K. Sung, and R. Rosenberg. Game Programming in CS0: A Scaffolded Approach. *Journal of Computing Sciences in Colleges* 26(1):

126-132 (2010).

35. M. V. Dyne, and J. Braun. Effectiveness of a Computational Thinking (CS0) Course on Student Analytical Skills. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*: 133-138 (2014).
36. S. Uludag., M. Karakus, and S. W. Turner. Implementing IT0/CS0 with Scratch, App Inventor For android, and Lego Mindstorms. *Proceedings of the 2011 Conference on Information Technology Education*: 183-190 (2011).
37. M. Haungs., C. Clark., J. Clements, and D. Janzen. Improving First-year Success and Retention Through Interest-based CS0 Courses. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*: 589-594 (2012).
38. D. Gudmundsen., L. Olivieri, and N. Sarawagi. Using Visual Logic: Three Different Approaches in Different Courses - General Education, CS0, and CS1. *Journal of Computing Sciences in Colleges* 26(6): 23-29 (2011).
39. 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* 35(3): 347-358 (2016).
40. J. Q. Dawson., M. Allen., A. Campbell, and A. Valair. Designing an Introductory Programming Course to Improve Non-Majors' Experiences. *Proceeding of 49th ACM Technical Symposium on Computer Science Education*: 26-31 (2018).
41. Z. J. Wood., J. Clements., Z. Peterson., D. Janzen., H. Smith., M. Haungs., J. Workman., J. Bellardo, and B. DeBruhl. Mixed Approaches to CS0: Exploring Topic and Pedagogy Variance After Six Years of CS0. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*: 20-25 (2018).
42. S. Mishra., S. Balan., S. Iyer, and S. Murthy. Effect of a 2-week Scratch Intervention in CS1 on Learners with Varying Prior Knowledge. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*: 45-50 (2014).
43. J. J. Merrienboer, and J. Sweller. Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review* 17(2): 147-177 (2005).
44. T. Babbitt., C. Schooler, and Kyle King. Punch Cards to Python: A Case Study of a CS0 Core Course. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*: 811-817 (2019).
45. J. Parham-Mocello., M. Erwig, and E. Dominguez. To Code or Not to Code? Programming in Introductory CS Courses. *Proceeding of IEEE Symposium on Visual Languages and Human-Centric Computing*: 187-191 (2019).
46. M. Klassen. Visual Approach for Teaching Programming Concepts. *Proceedings of the 9th International Conference on Engineering Education*: M4H-5-M4H-10 (2006).
47. V. Isomoottonen., A. Lakanen, and V. Lappalainen. K-12 Game Programming Course Concept Using Textual Programming. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*: 459-464 (2011).
48. R. Garlick, and E. C. Cankaya. Using Alice in CS1: A Quantitative Experiment. *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*:165-168 (2010).
49. L. Jarratt., N. A. Bowman., K. C. Culver, and A. M. Segre. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Educatio*n: 176-181 (2019).
50. M. Ayub., O. Karnalim., R. Risal., W. F. Senjaya, and M. C. Wijanto. Utilising pair programming to enhance the performance of slow-paced students on introductory programming. *Journal of Technology and Science Education* 9(3): 357-367 (2019).
51. M. S. Naveed, and M. Sarim. Didactic Strategy for Learning Theory of Automata & Formal Languages, Pakistan Academy of Sciences, *A. Physical and Computational Sciences* 55 (2): 55–67 (2018).
52. C. Tsai. Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy, *Computers in Human Behavior* 95: 224-232 (2019).
53. M. Noone, and A. Mooney. Visual and textual programming languages: a systematic review of the literature, *Journal of Computers in Education* 5: 149-174 (2018).
54. M. N. Demaidi, M. Qamhieh, and A. Afeefi. Applying Blended Learning in Programming Courses, *IEEE Access* vol. 7: 156824-156833, (2019).
55. M. S. Naveed, M. Sarim, and A. Nadeem. C in CS1: Snags and Viable Solution, *Mehran University Research Journal of Engineering & Technology* 37(1): 1-14 (2018).
56. J. Figueiredo, and F. J. García-Peñalvo. Intelligent Tutoring Systems approach to Introductory

Programming Courses. *Proceedings of the Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*: 34-39 (2020).

57. M. S. Naveed, and M. Sarim. Analyzing the Effects of Error Messages Presentation on Debugging and Programming, *Sukkur IBA Journal of Computing and Mathematical Sciences* 4(2): 38-48 (2020).
58. S. Papadakis, and M. Kalogiannakis. Using Gamification for Supporting an Introductory Programming Course. The Case of ClassCraft in a Secondary Education Classroom, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* 229: 366-375 (2018).
59. U. Omer, M. S. Farooq, and A. Abid. Introductory programming course: review and future implications, *PeerJ Computer Science* 7: e647 (2021).
60. T. Teodosiev, and A. Nachev, Some Pitfalls in Introductory Programming Courses. *Informatics in Education* 11(2): 241-255 (2012).