C in CS1: Snags and Viable Solution

SHUMAIL NAVEED*, MUHAMMAD SARIM*, AND ADNAN NADEEM*

RECEIVED ON 04.12.2015 ACCEPTED ON 16.02.2016

ABSTRACT

Programming is one of the career rewarding skills; however, learning programming skill is extremely hard and arduous as supported by several studies. The first programming language has an everlasting impact on the programmer's program's development abilities. In most of the universities the imperative paradigm is used for introductory programming courses and generally C language is used as a base language of a first programming course. The C language is a leading programming language and extensively utilized for commercial applications. The majority of the programming languages are highly motivated from the C language, yet its intrinsic complexities and non-pedagogic origin evidently makes it hard and a complex choice for a first programming course. This paper proposed a rational and realizable solution that can make a C language a suitable choice for a first the course of programming.

Key Words: Introductory Programming Courses, Doodles, Pre-Programming, Teaching Methodology, Pair Programming.

1. INTRODUCTION

Programming skill is an essential element of computer science courses and one of many skills that students of computer science programs are presumed to master [1]. One of the most difficult aspects of computer science curriculums involves facilitating students in comprehending the concepts of computer programming. Almost in all computer science academic programs, it is required to take an introductory programming course. Programming is an important skill and a rewarding career. It is usually recognized that it takes around ten years of experience to be an professional programmer [2]. However, learning to program is a difficult task for beginners [3,4]. Hagan and Markham claim that for all computing courses the programming is the least interesting and most difficult subject by most first year students [5]. Programming is inherently complex and beginning students naturally lack the knowledge and dexterity of programming experts. Their knowledge is not general but context specific [6]. Beginning students are restricted to the surface knowledge, deficient in comprehensive mental models, unsuccessful to employ pertinent knowledge, and visualize the programming line-by-line rather than utilizing consequential program components or formations.

The introductory programming course is a "gate-keeper" to triumph and success in computer science/ computing education [7]. The significance of an appropriately defined first course in programming (usually called CS1) can not be overstated. The first programming course leaves the beginning students with good programming practices [8]. A poor experience may have a worse impact on student attitude, and may result in a change in majors.

Corresponding Author (E-Mail: shumail.naveed@fuuast.edu.pk)
* Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Karachi.

The first programming language of student has a strong effect on program development capabilities as significant as the impact of our native language on our thought [9]; in fact the first programming language will set the tone for all subsequent classes [10].

Selection of an appropriate programming language for the first course on programming is an important issue [11,12]. There have been very substantial studies and discussion on selection of programming language for introductory courses of programming [13-16]. However, the introductory programming courses are widely recognized to pose challenges and problems for both students and instructors and high dropout rates of 20-60% are reported in different academic institutions [17-20]. Despite of an extensive study in introductory programming, the pass rates were not observed to have considerably differed over time [21].

The appropriate paradigm and language chosen for the introductory programming course may overcome the intrinsic complexity of introductory programming. There are around 2000 to 3000 famous programming languages documented on the Web [22], consequently it is intricate to select any programming language for the first programming course. Over the years there has been a substantial debate and study about which language is appropriate for the introductory programming course; hitherto there is no final consensus on any programming language. Such as in [23], it is argued that Java is better than C++, and C++ is not a good choice for introductory programming. Similarly, Wallace and Martin considered that Java is better that other programming language [24]. However, Biddle and Tempero [25] categorized the Java a reasonable, but not an efficacious choice for introductory programming courses. In [26], Python is selected for the introductory programming course.

Raina et. al. conducted a landmark study of the 44 introductory programming courses in 28 universities in Australia and compared the results with previously conducted censuses during 2001 and 2003 [27]. According to that study, 54.5% of instructors are using procedural paradigm, 25.0% are using the object-oriented paradigm, 18.2% are following the mix of paradigm and 2.3% are using a functional paradigm. The study also revealed that Java, Python, C, C# and Visual Basic are the widely used programming languages.

In 2013, the same pattern of study is conducted and surveyed the 38 introductory programming courses in New Zealand and Australian universities [28]. According to that study the imperative (procedural) paradigm is the preferred paradigm used for teaching as shown in Fig. 1.

The study also reported that Python, Java, JavaScript, C, C++ and C# are the popular languages. The programming paradigm and languages followed in the teaching institutions of Pakistan are somewhat different from other countries. At the time of writing, there are around 164 universities/degree awarding institutions in Pakistan [29], and according to the available information, the majority of the universities/institutions are formally running the pure undergraduate computer science programs and nearly all are using the imperative paradigm for the first programming course (usually called CS1). It is worth noting that around 90% of these universities in Pakistan are using the C language in CS1.

C is the general purpose programming language and extremely useful for different types of applications [30]. It is the core language of the UNIX operating system. C is equipped with powerful data-structures, useful

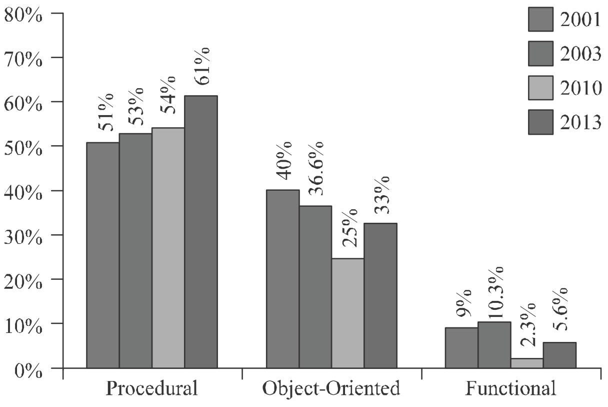


FIG. 1. TRENDS IN PARADIGM TAUGHT [28]

expressions, control structures and a large variety of data types. Its tiny size and generalization make it suitable for several domains.

The TIOBE Programming Community index [31], acknowledges the C language as a popular programming language. Table 1 shows the rating of the top 10 languages of March 2015.

As a part of a study of the motivation, delineation and anatomy of precursor and pedagogical programming languages, we have made the discussion and interview with 154 experienced teachers/academic instructors in the teaching institutions of Pakistan. Instructors were asked to indicate the appropriate programming paradigm for CS1.

TABLE 1. TIOBE INDEX FOR MARCH 2015

Popularity	Programming Language	Ratings (%)
1	C	16.642
2	Java	15.580
3	Objective-C	6.688
4	C++	6.636
5	C#	4.923
6	PHP	3.997
7	JavaScript	3.629
8	Python	2.614
9	Visual Basic.Net	2.326
10	Visual Basic	1.949

Among all the instructors, 106 recommended the imperative paradigm, 24 endorsed the mix of paradigms and 18 recommended the object-oriented paradigm, it is surprising that some instructors, 4 from 154 recommended the logic paradigm, and 2 recommended the functional paradigm. Fig. 2 shows the overall recommendation in numeric proportions:

The response about the recommended paradigms clearly indicates that imperative paradigm is one of a most advocated method in Pakistan. The instructors were also asked to recommend the appropriate programming language for the first imperative programming course. Fig. 3 shows the response of recommended programming languages.

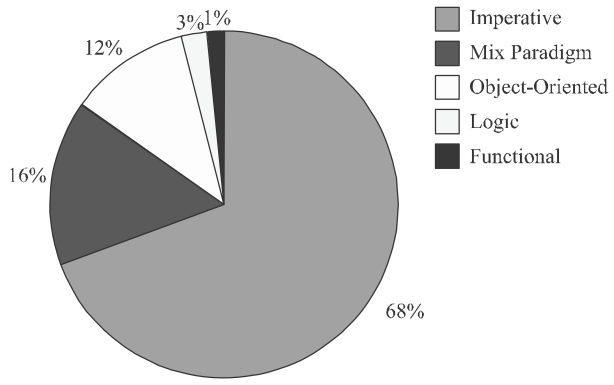


FIG. 2. RECOMMENDED PARADIGMS

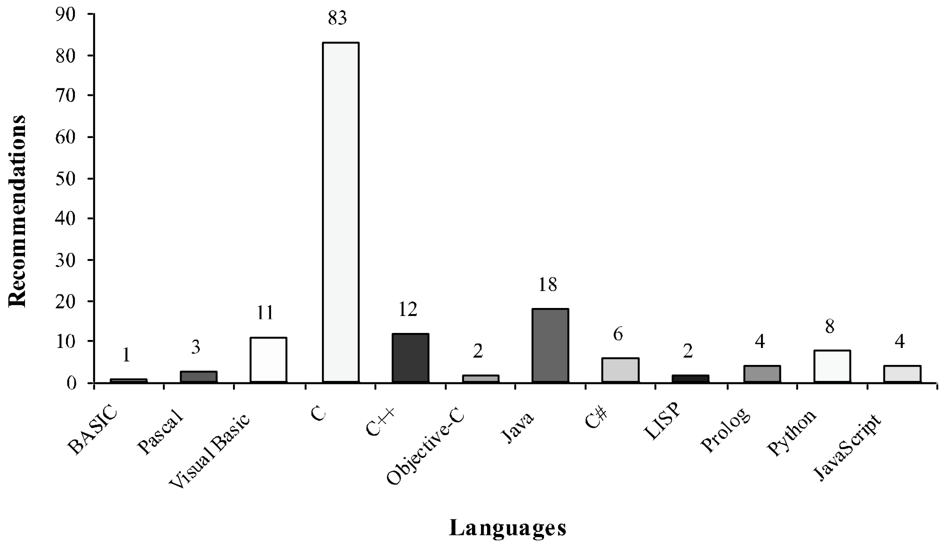


FIG. 3. RECOMMENDED PROGRAMMING LANGUAGES

The response of recommended programming languages indicates that for majority of instructors in Pakistan, the C language is still a favorite choice for the first programming course.

The C is widely recognized language, yet there have been a wide debate about its educational appropriateness and usefulness.

The pedagogical application of a C language is explicitly opposed in that been designed especially for educational purposes [32]. This paper introduced a plain technique that can make C language a suitable option for the CS1 in controlling its inherent complexity and intricacies.

The rest of the paper is structured as follows. The major intricacies and difficulties association with the application of the C language in introductory courses of programming are described in the second section, while the prior work is briefly illustrated in a third section. The proposed solution is introduced in a fourth section. The initial results obtained after the primary assessment of the proposed techniques are described in the fifth section and the last section includes the conclusion.

2. CRITICISM AND PEDAGOGICAL ISSUES

The C language has wide applications in commercial and scientific areas. However, it is explicitly criticized and opposed to education in the introductory courses of programming. Following is the brief element of main appraisal and educational concerns of the C language.

(1) The C is not outsized, yet too complex for novice students [33], and it is unfeasible to introduce entire C language in the CS1.

(2) It is openly described that C language was designed by Dennis Ritchie, who was not non-educationist but a mathematician and physicist. So rarely considered the educational aspects during the development of a C language, and ultimately it is unenviable to exercise C language for introductory courses of programming.

(3) C is not a natural language and its syntax and facets like program organization, arrays and control structures are difficult for novice students [34]. In [35], Petrov claims that unexpected and unfamiliar syntax of C language creates the confusion and raise the psychological tension.

(4) The unsigned data types in C are virtually useless, in that it is safe and simple to use the signed data. Similarly the formatting symbol "%" is difficult to rationalize.

(5) The loops and other control structures are quite difficult to identify. C programming language has weak typing. For instance, the ASCII characters can be used as an integer, and this type of provisions may lead to confusion.

(6) Most of the operators are confusing, orthogonal and contradictory with the logical and mathematical notations. For instance, multiple ways are available for arithmetic assignment:

```
a += 1;

a = a + 1;
```

The abbreviation of several arithmetic operations seems useless and overwhelming.

(7) The equality statement is extremely difficult and mostly muddled with assignment statement and consequently causes strenuous errors.

(8) The grammar of C language controverts to the human language and grammar [35], and C itself does not follow the conventional mathematical notation.

(9) The subscripting of arrays in C language starts from zero, and it also does not support bound checking, for that reasons novice students face severe problems in the learning and implementing arrays based programs.

(10) The majority of the statements of C language is permitted to be defined on a single line, while numerous statements entails a unique line. For novice students of CS1, it is mostly very complex to comprehend and implement these dissimilarities. The case sensitivity of a C language is another major issue that makes the language more difficult for beginners.

(11) The Boolean-data type is not available in C language and educationally it is a main problem [36], and the corresponding code for the realization of Boolean-data type is usually very odd and complex in the framework of learning and generally an error prone.

(12) Programmer-defined functions, arguments and the classes of arguments are generally difficult for novice students.

(13) The pointer is an essential constituent of a C program, but it is extremely complex for beginners, especially the arithmetic of C language is exceptionally complex to comprehend. Egan and McDonald claim that pointers and manual memory management can be difficult to comprehend [37]. The pointers are not only difficult for beginners, but also hard for experienced programmers [35].

(14) The I/O operations are always very interrelated, and their equivalent constructs should be very symmetrical, however the ‘printf’ and ‘scanf’ are highly nonsymmetrical. The ampersand sign “&” is mandatory for ‘scanf’, but not for ‘printf’. The conventional input function ‘scanf’ in the C language, is naturally difficult to use because it involves a pointer to the variable for storing the input value [38].

(15) The union is one of a useful feature of the C language; though, it is abnormally strenuous to understand and implement.

(16) The shift-wise operators and bit-manipulation operators are very difficult to learn and implement.

(17) The very strange notion of logical operators is used in C language. The exclamation mark “!” is used as an operator in the C language, whereas it is customarily used for different purpose, and these differences make it confusing for beginners.

(18) Conditional compilation is an imperative element of the C language, but typically very difficult for novice students. Most of the available compilers of the C language are not equipped with powerful debuggers, and due to this fact, it is very strenuous to debug the C programs.

(19) In [39], it is described that the C language is an industry standard and for that reason it should not utilized as an introductory language because the beginners will become biased for future programming languages. It is also described that C is a magnificent programming language for terrorists as it is excellent for negative programming.

3. SIGNIFICANT SOLUTIONS

Several solutions have been developed to define the C language viable for introductory courses in programming. In [34], the notion of using HTML, EXCEL and MATLAB before introducing the C language is proposed. These applications help the novice beginners in learning the difficult features of a C language. For instance, the HTML can be helpful in comprehending the layout of the program; EXCEL may be fruitful in learning the input/output and arrays. Likewise, MATLAB can be productive in introducing the condition and repetition control structures.

Egan and McDonald [37] introduced the notion of natural language description of program behavior and program visualization to help the beginners in increasing their knowledge, understanding and aid in program debugging. A system based on that proposed method is formalized and used for C. Similarly, in [40], the use of Robotic Invention System 2.0 for introducing the C language is introduced.

In [36], it is argued that there are some troubles in employing the C language at the initial level; nonetheless it is possible to lessen the difficulties by utilizing the benefits of conventional software engineering techniques.

4. METHOD AND DISCUSSION

The C is a very prevailing language, but it is difficult to conventionally teach C language to novice students than it is to introduce the pure educational programming language, but the indelible paybacks are more important and necessary than momentary outlays. Practically, it is possible to diminish the complexities associated with using the C language in introductory courses of programming by utilizing the proposed method. The proposed technique is based on 4 pillars.

4.1 Prior Knowledge

The lack of prior knowledge of programming is a major cause for the intricacy of C language. The novices without prior experience of programming suffer severe problems in learning process [41,42]. However, the beginners with prior acquaintance of programming work better, particularly in the CS1 [43,44]. It is widely believed that the C language is extremely logical and this perception demotivated the novice students.

The first principle of the proposed technqiue is to introduce a foundation course before introducing the actual C language. This foundation course can provide the basic knowledge of algorithmic thinking, problem solving and essential of elementary programming. Fortunately, several preprogramming tools are available.

GRAIL is preprogramming language designed to offer before the first course of programming and helps the beginners to understand the programming concepts [45]. GRAIL is a small programming language designed to support the imperative programming concepts. On the whole the statements of GRAIL are almost similar to the equivalent statements of contemporary high level programming languages. Each construct in GRAIL has a single syntax and it's a viable choice to introduce before the C language.

In [46] a CS0 course has introduced to help the beginners who have no previous knowledge of programming. It covers the fundamental topics like sequences, operators, selection statements, repetitive statements, functions and arrays. The course first uses GameMaker [46] to cover the fundamental topics of elementary programming. The same pattern course can be useful before starting the C language.

In [48], Dyne and Braun described and the CS0 course called Computational Thinking to develop and improve the analytical problem solving of those students who are not prepared for introductory programming. The results of using CS0 indicate a positive impact on improving the analytical problem solving skills of students.

A preliminary course has designed by using Alice [48] to prepare the students for comprehending the concepts of programming and preparing them for a first programming course [50,51]. The results indicate the aptness of Alice in improving the performance of students and similarly the same leverage can be obtained by introducing Alice before the C language.

In [52], CS 0.5 course is introduced to before the first programming course. The intention of CS 0.5 is to give beginners a programming maturity before diving into CS1. The gentle revised version of Guzdial's media computation [53] is used in CS 0.5.

Sometimes it is undesirable to initiate a foundation language before starting the C language; in that situation, it is possible to introduce the educational dialects of C language (like Educational C [33]), and the dedicated programming environment (like Thetis [54]) before starting the C language.

Apart from the other tools there are myriad many tools like RAPTOR and FLOWGORITHM that can be used to introduce the fundamental of elementary programming before the C language.

4.2 Use of Doodles

Several elements of the C language like input/output, arrays, pointers, functions and types of arguments are obviously difficult for novice students, and the key reasons from their intricacy are the complexity of topics and the unusual syntax of the C language. The novice students are generally forced to learn these notions and the strange syntax of the language at the same time. The unknown and mysterious taxonomies and complex syntax of C makes the control structures, arrays, pointers and functions more difficult for beginners; however the reasonable exercise of doodles can conquer these problems.

Utmost employment of doodles in introducing and learning the C language is the second principle of the proposed technique. Doodle is any kind of an illustration or diagram drawn to comprehend the purpose of the programs [55]. Doodle is exceptionally helpful in the preliminary stages of programming [56]. With the different types of doodles [55], it is significantly straightforward to comprehend the hard concepts without considering the complicated syntax of the C language. Through pictorial illustrations and doodles it is quite simple to learn these fundamental concepts.

4.3 Less is More

Nearly all the contemporary programming languages, including the C is radically very large, and sensibly it is unreasonable and unfair to teach entire C language in the CS1, but only the necessary topics are advantageous in the CS1, where complex topics like union, bitwise-operators, shift-wise operators and conditional compilation can be offered in the later programming courses or may be introduced in the later stages of the CS1 and this is the fundamental idea of the third principle of the proposed technique.

4.4 Pair Programming

The C language is completely diverse from programming languages and therefore requires a radically special approach. Similarly, the time, efforts, adroitness and deliberation need to learn the C language is appreciably more than the other programming languages.

It is widely accepted that the pair programming can overcome the several pedagogical problems. Pair programming is a pivotal practice of Extreme Programming and a way in which two developers work together at one system on the same problem [57]. The programmer writing the program is called the driver and the other teammate is called navigator. Pair programming has been largely applied in education because of the leverage it brings to students. The pair programming has been fruitful in introductory programming courses.

In order to strengthen our thesis the instructors were asked to indicate whether the pair programming can surmount the several problems associated with the use of the C language in a first programming course. Fig. 4 indicates the response of instructors.

Most of the instructors recognized that the effective teaching methodology can be a useful panacea for the several pedagogical issues of C languages. At last, it is worth here to state that in the teaching of C language programming, design, some problems exists on both sides, students and instructors [58]. So the instructors should revolutionize their teaching philosophy and students should instinctively learn and ameliorate their learning motivation.

5. PRELIMINARY TEST AND RESULTS

In order to ascertain whether the proposed method is really effective in making the C language a viable choice for the CS1, a small study is conducted. As a part of the study, 82 students of the undergraduate computer science program are selected and randomly divided in two groups (called control group and treatment group). The C based first programming course is offered by the same instructor to the both groups, and the conventional teaching approach is followed for the control group, whereas the proposed method is followed for the treatment group. For the realization of a first principle of the proposed method, we have used the RAPTOR environment to introduce the algorithmic and preprogramming knowledge to the treatment group. The flowchart-based notation has been extensively used in the context of introducing novices to programming [59]. RAPTOR is a programming environment, developed exclusively to aid novice students in comprehending the programming and the algorithms. RAPTOR is successfully used to introduce programming to novice students with flowcharts [60-62]. Fig. 5 shows the environment of RAPTOR.

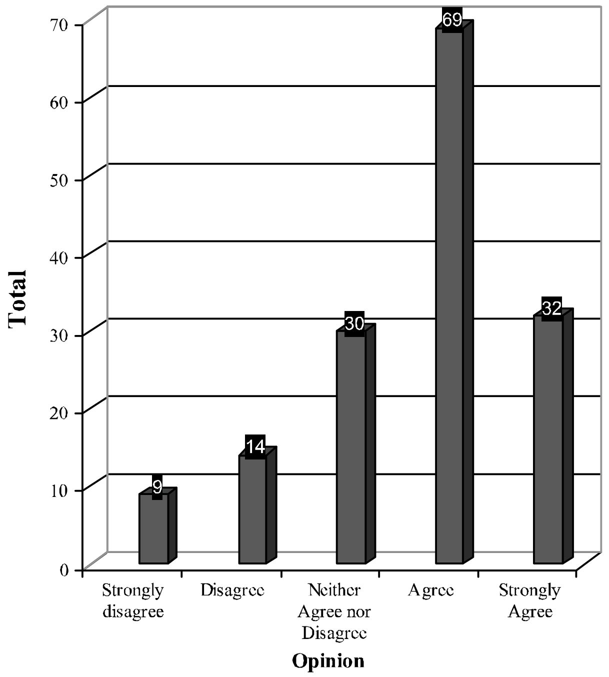


FIG. 4. IMPACT OF PAIR PROGRAMMING

The preprogramming environment/language is virtually introduced to increase the motivation level by providing the prior knowledge before starting the actual massive course.

Throughout the duration of course, the students of treatment group are softly encouraged and motivated to doodle the programs. The instructor itself doodles the major programs and concepts introduced during the course. With the extensive use of doodles it is clearly realized that several logical concepts of the C language are extremely easy to comprehend and implement. Fig. 6 shows the sample doodle for the arrays.

Several concepts of C language like functions, types of arguments and pointers are intrinsically hard to teach and comprehend, whereas the feedback received from the instructor illustrate that the use of doodle surprisingly reduced their complexities. As an illustration, consider the Fig. 7 for the sample doodle of user defined function.

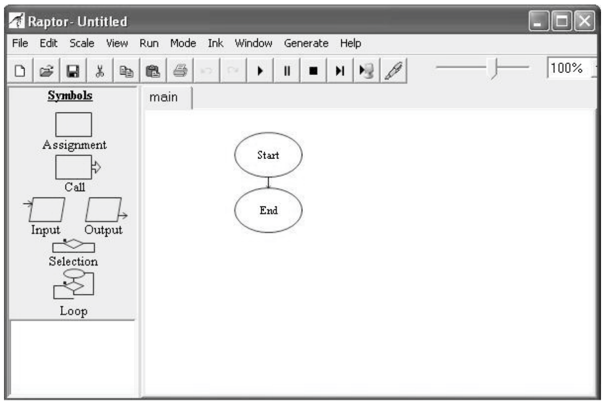
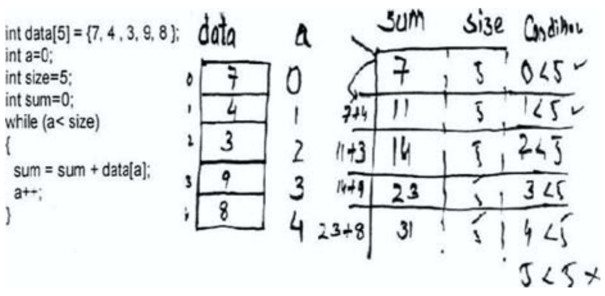


FIG. 5. RAPTOR ENVIRONMENT



```
int data[5] = {7, 4 , 3, 9, 8 };
int a=0;
int size=5;
int sum=0;
while (a< size)
{
  sum = sum + data[a];
  a++;
}
```

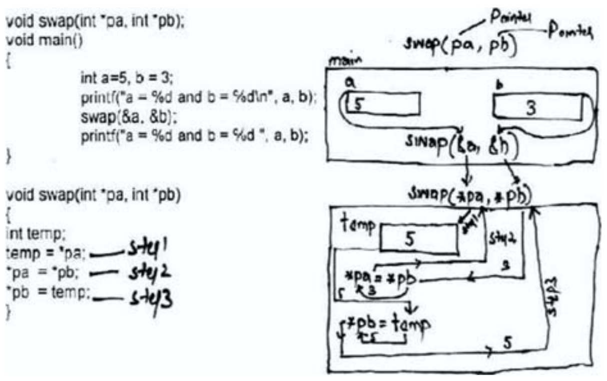
FIG. 6. DOODLE OF ARRAYS

The curriculum of the C based first programming course for the both groups was same and included the elementary topics, but also included some complex topics like the bitwise operators, union and conditional compilation. As the realization of the third principle of the proposed method, these complex topics are introduced in the later lectures, and introduce with the extensive use of doodles. For treatment group the instructor principally follows the imperative-first approach, but also consider the concept-first approach.

After the completion of course, both groups are evaluated with pen and paper exam. Fig. 8 shows the results.

Fig. 9 shows the box plot of the marks obtained by the students of the control group.

The box plot of the marks obtained by the students of the treatment group is shown in Fig. 10.



```
void swap(int *pa, int *pb);
void main()
{
        int a=5, b = 3;
        printf("a = %d and b = %d\n", a, b);
        swap(&a, &b);
        printf("a = %d and b = %d ", a, b);
}

void swap(int *pa, int *pb)
{
int temp;
temp = *pa;  —— step1
*pa  = *pb;  —— step2
*pb  = temp; —— step3
}
```

FIG. 7. DOODLE FOR THE USER DEFINED FUNCTION

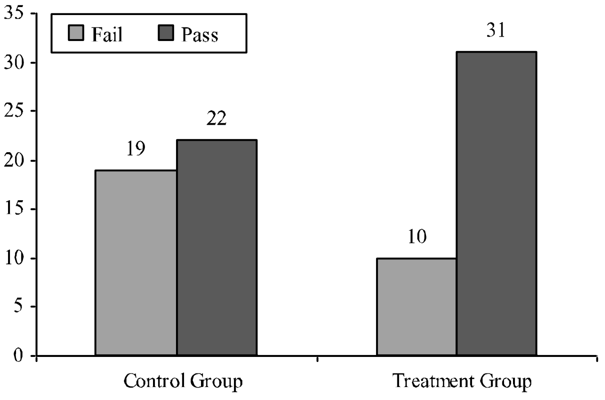


FIG. 8. RESULTS OF EVALUATION

The results obtained from the initial assessment of the proposed method reveals that the proposed method is quite helpful in diminishing the complexity of using the C language in a first programming course. In the control group the pass rate was 53.56 where 75.61 in the treatment group. The mean marks for the control group was 49.80 (standard deviation = 20.85) whereas the mean marks for the treatment group was 57.83 (standard deviation = 19.51). Independent sample t-test with a t-value = 2.38 and p-value < 0.01 signified that students of treatment group did significantly better in C based first programming course than the students of the control group in the same C based first programming course.

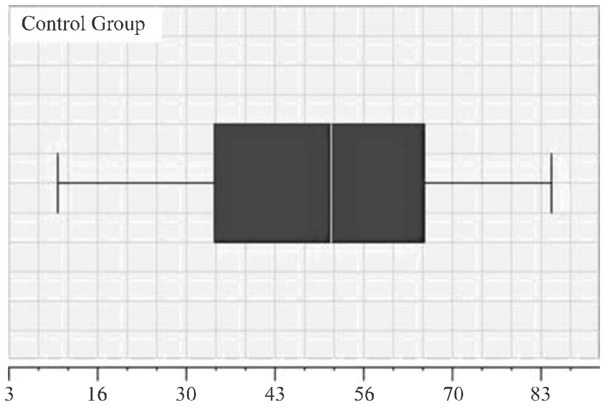


FIG. 9. BOX PLOTS OF THE MARKS OF STUDENTS IN CONTROL GROUP

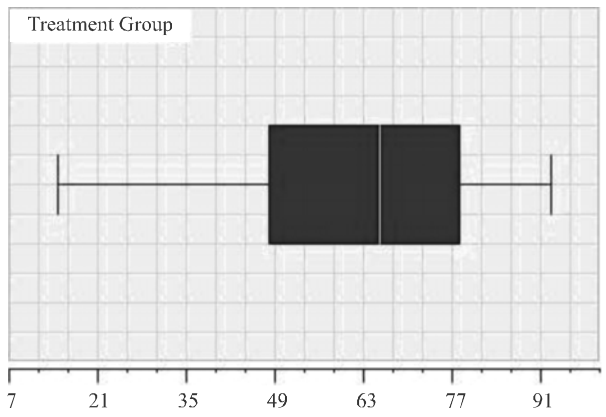


FIG. 10. BOX PLOTS OF THE MARKS OF STUDENTS IN TREATMENT GROUP

The grade wise distribution of marks of students of both groups is shown in Fig. 11.

From current results and analysis, it can be inferred that the proposed method can also improve the motivation level and may be useful in improving retention rate and controlling the dropout rates. In order to obtain, the significance of the proposed method in increasing the retention rate, students from both groups were interviewed and asked "whether they are willing to study another programming course in the next semester". The feedback received from the students is shown in Fig. 12.

The feedback received from the students indicated that students in the treatment group are highly motivated and their retention rate is much higher the students of the control group.

At last is important to realize that the pedagogical success of any programming language and improvement of retention pivot on several factors, albeit the feedback received from the both groups clearly indicates that the proposed method may be fruitful in reducing the inherent complexities of C language.

6. CONCLUSION

The C language is of a widely used programming language; however, its non-educational source makes it disputed and problematic for the CS1. In this paper a simple method is proposed that can make the C language a realizable choice for the CS1 by reducing it intrinsic complexities. The proposed method is based on four main principles which are simple, feasible and applicable. We have applied the proposed method on a small group of students and statistical results and knowledge of using the proposed technique suggest that it can be useful in controlling the intrinsic intricacies of C language and can ultimately make the C language a viable choice for the CS1.

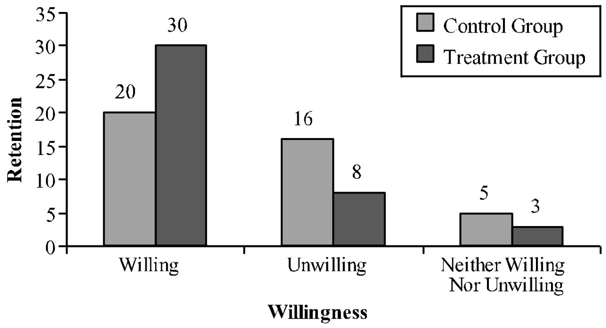
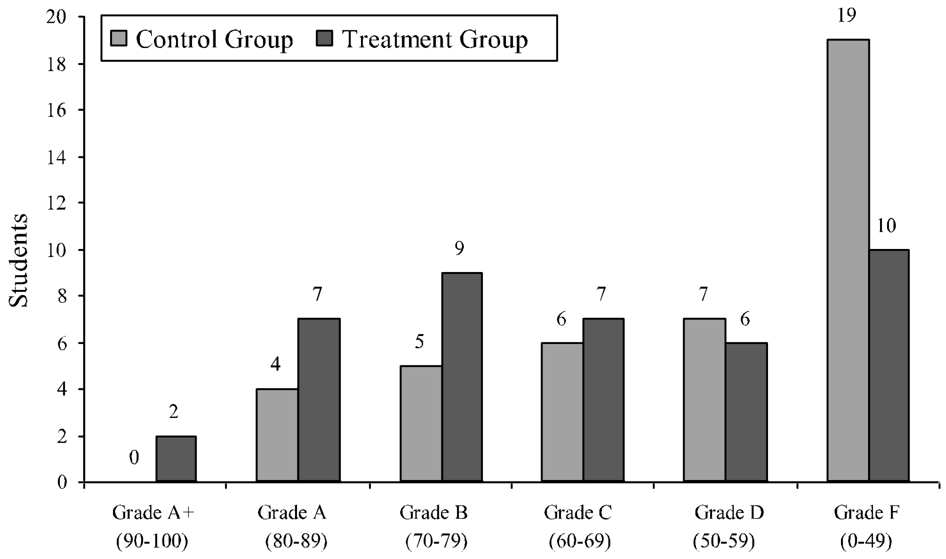


FIG. 12. STUDENTS WILLINGNESS IN NEXT PROGRAMMING COURSE



FI. 11. GRADE-WISE RESULT

ACKNOWLEDGEMENTS

The support provided by the Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Karachi, Pakistan, is appreciatively acknowledged. Authors would also like to express thanks Kashif Raffat, for his precious suggestions and criticisms. In addition, authors would like to thank all of the students who participated in evaluating the proposed method.

REFERENCES

[1] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I., and Wilusz, T., “A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students”, Working Group Reports from Innovation and Technology in Computer Science Education, pp.125-180, USA, 2001.

[2] Winslow, L.E., “Programming Pedagogy - A Psychological Overview”, ACM SIGCSE Bulletin, Volume 28, No. 3, pp.17-22, USA, 1996.

[3] Anna, E., Laakso, M., Lopez, M., and Sarkar, A., “Relationship between Text and Action Conceptions of Programming: A Phenomenographic and Quantitative Perspective”, Proceedings of 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, pp. 33-37, Germany, 2011.

[4] Sabitzer, B., and Pasterk, S., “Brain-Based Programming Continued: Effective Teaching in Programming Courses”, IEEE Frontiers in Education Conference, pp. 1-6, Spain, 2014.

[5] Hagan, D., and Markham, S., “Does it Help to Have Some Programming Experience Before Beginning a Computing Degree Program?”, ACM SIGCSE Bulletin, Volume 32, No. 3, pp. 25-28, USA, 2000.

[6] Byrne, P., and Lyons, G., “The Effect of Student Attributes on Success in Programming”, Proceedings of 6th Annual Conference on Innovation and Technology in Computer Science Education, pp. 49-52, USA, 2001.

[7] Selby, L., Ryba, K., and Young, A., “Women in Computing: What does the Data Show?”, ACM SIGCSE Bulletin, Volume 30, No. 4, pp. 62-67, USA, 1998.

[8] Pendergast, M.O., “Teaching Introductory Programming to IS Students: Java Problems and Pitfalls”, Journal of Information Technology Education, Volume 5, USA, 2006.

[9] Wexelblat, R.L., “The Consequences of One’s First Programming Language”, Proceedings of 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems, pp. 52-55, USA, 1980.

[10] Howell, K., “First Computer Languages”, Journal of Computing Sciences in Colleges, Volume 18, No. 4, pp. 317-331, USA, 2003.

[11] Laakso, M., Kaila, E., Rajala, T., and Salakoski, T., “Define and Visualize Your First Programming Language”, 8th IEEE International Conference on Advanced Learning Technologies, pp. 324-326, Spain, 2008.

[12] Minor, J.T., and Gewali, L.P., “Pedagogical Issues in Programming Languages”, International Conference on Information Technology: Coding and Computing, Volume 1, pp. 562-565, 2004.

[13] Ali, A., and Smith, D., “Teaching an Introductory Programming Language in a General Education Course”, Journal of Information Technology Education: Innovations in Practice, Volume 13, 2014.

[14] Ivanovic, M., Budimac, Z., and Paunic, D., “Educational Influences of Choice of First Programming Language”, Proceedings of International Conference on Numerical Analysis and Applied Mathematics, pp. 310010-1–310010-4, Greece, 2014.

[15] Goosen, L., “A Brief History of Choosing First Programming Languages”, History of Computing and Education, Volume 3, pp. 167-170, USA, 2008.

[16] Parker, K.R., Chao, J.T., Ottaway, T.A., and Chang, J., “A Formal Language Selection Process for Introductory Programming Courses”, Journal of Information Technology Education: Research, Volume 5, No. 1, pp. 133-151, USA, 2006.

[17] Herrmann, N., Popyack J.L., Char, B., Zoski, P., Cera C.D., Lass, R.N., and Nanjappa, A., "Redesigning Introductory Computer Programming Using Multi-Level Online Modules for a Mixed Audience", ACM SIGCSE Bulletin, Volume 35, No. 1, pp. 196-200, USA, 2003.

[18] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S., "Improving the CS1 Experience with Pair Programming", ACM SIGCSE Bulletin, Volume 35, No. 1, pp. 359-362, USA, 2003.

[19] Rich, L., Perry, H., and Guzdial, M., "A CS1 Course Designed to Address Interests of Women", ACM SIGCSE Bulletin, Volume 36, No. 1, pp. 190-194, USA, 2004.

[20] Sloan, R.H., and Troy, P., "CS 0.5: A Better Approach to Introductory Computer Science for Majors", ACM SIGCSE Bulletin, Volume 40, No. 1, pp. 271-275, USA, 2008.

[21] Watson, C., and Li, F.W.B., "Failure Rates in Introductory Programming Revisited", Proceedings of Conference on Innovation & Technology in Computer Science Education, pp. 39-44, USA, 2014.

[22] Kaplan, R.M., "Choosing a First Programming Language", Proceedings of ACM Conference on Information Technology Education, pp. 63-164, USA, 2010.

[23] Jab³onowski, J., "A Case Study in Introductory Programming", International Conference on Computer Systems and Technologies, Volume 82, pp. 1-7, Bulgaria 2007.

[24] Wallace, C., and Martin, P., "Not Whether Java but How Java", Proceedings of International Computer Science Conference on Asia-Pacific Software Engineering, pp. 517-518, Hong Kong, 1997.

[25] Biddle, R., and Tempero, E., "Java Pitfalls for Beginners", ACM SIGCSE Bulletin, Volume 30, No. 2, pp. 48-52, USA, 1998.

[26] Grandell, L., Peltomki, M., Back, R., and Salakoski, T., "Why Complicate Things?: Introducing Programming in High School Using Python", Proceedings of 8th Australasian Conference on Computing Education, Volume 52, pp. 71-80, Australia, 2006.

[27] Mason, R., Cooper, G., and deRaadt, M., "Trends in Introductory Programming Courses in Australian Universities: Languages, Environments and Pedagogy", Proceedings of 14th Australasian Conference on Computing Education, Volume 123, pp. 33-42, 2012.

[28] Mason, R., and Cooper, G., "Introductory Programming Courses in Australia and New Zealand in 2013 - Trends and Reasons", Proceedings of the 16th Australasian Conference on Computing Education, Volume 148, pp. 139-147, New Zealand, 2014.

[29] http://www.hec.gov.pk/Ourinstitutes/pages/Default.aspx (Last Accessed: 2nd April, 2015).

[30] Ritchie, D.M., Johnson, S.C., Lesk, M.E., and Kernighan, B.W., "UNIX Time-Sharing System: The C Programming Language", Bell System Technical Journal, Volume 57, No. 6, pp. 1991-2019, USA, 1978.

[31] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. (Last Accessed: 6th April, 2015).

[32] Pears, A., Seidman, S., and Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J., "A Survey of Literature on the Teaching of Introductory Programming", ACM SIGCSE Bulletin, Volume 39, No. 4, pp. 204-223, USA, 2007.

[33] Ruckert, M., and Halpern, R., "Educational C", ACM SIGCSE Bulletin, Volume 25, No. 1, pp. 6-9, USA, 1993.

[34] Budny, D., Lund, L., Vipperman, J., and Patzer, J. L.II, "Four Steps to Teaching C Programming", 32nd Annual Frontiers in Education, Volume 2, pp. 18-22, 2002.

[35] Petrov, P.T., "New Evaluation of the Language C for Educational", International Scientific Conference on Engineering and Scientific Purposes, pp. 353-361, 2010.

[36] Roberts, E.S., "Using C in CS1: Evaluating the Stanford Experience", ACM SIGCSE Bulletin, Volume 25, No. 1, pp. 117-121, 1993.

[37] Egan, M.H., and McDonald, C., "Program Visualization and Explanation for Novice C Programmers", Proceedings of 16th Australasian Conference on Computing Education, Volume 148, pp. 51-57, New Zealand, 2014.

[38] Rosenberg, J., and Kölling, M., "I/O Considered Harmful (At Least for the First Few Weeks)", Proceedings of 2nd Australasian Conference on Computer Science Education, pp. 216-223, Australia, 1997.

[39] Johnson, L.F., "C in the First Course Considered Harmful", Communications of the ACM, Volume 38, No. 5, pp. 99-101, USA, 1995.

[40] Kim, S.H., and Jeon, J.W., "Educating C Language Using LEGO Mindstorms Robotic Invention System 2.0", Proceedings of IEEE International Conference on Robotics and Automation, pp. 715-720, USA, 2006.

[41] Davy, J.R., Audin, K., Barkham, M., and Joyner, C., "Student Well-Being in a Computing Department", ACM SIGCSE Bulletin, Volume 32, No. 3, pp. 136-139, USA, 2000.

[42] Morrison, M., and Newman, T.S., "A Study of the Impact of Student Background and Preparedness on Outcomes in CS I", ACM SIGCSE Bulletin, Volume 33, No. 1, pp. 179-183, USA, 2001.

[43] Holden, E., and Weeden, E., "The Impact of Prior Experience in an Information Technology Programming Course Sequence", Proceedings of 4th Conference on Information Technology Curriculum, pp.41-46, USA, 2003.

[44] Tafliovich, A., Campbell, J., and Petersen, A., "A Student Perspective on Prior Experience in CS1", Proceedings of 44th ACM Technical Symposium on Computer Science Education, pp. 239-244, USA, 2013.

[45] McIver, L., and Conway, D., "GRAIL: A Zeroth Programming Language", Proceedings of 7th International Conference on Computing in Education, pp. 43-50, 1999.

[46] Panitz, M., Sung, K., and Rosenberg, R., "Game Programming in CS0: A Scaffolded Approach", Journal of Computing Sciences in Colleges, Volume 26, No. 1, pp. 126-132, USA, 2010.

[47] Hernandez, C.C., Silva, L., Segura, R. A., Schimiguel, J., Ledón, M.F.P., Bezerra, L.N.M., and Silveira, I.F., "Teaching Programming Principles through a Game Engine", CLEI Electronic Journal, Volume 13, No. 2, pp. 1-8, 2010.

[48] Dyne, M.V., and Braun, J., "Effectiveness of a Computational Thinking (CS0) Course on Student Analytical Skills", Proceedings of 45th ACM Technical Symposium on Computer Science Education, pp. 133-138, USA, 2014.

[49] Cooper, S., "The Design of Alice", ACM Transactions on Computing Education, Volume 10, No. 4, pp. 1-16, USA, 2010.

[50] Moskal, B., Lurie, D., and Cooper, S., "Evaluating the Effectiveness of a New Instructional Approach", ACM SIGCSE Bulletin, Volume 36, No. 1, pp. 75-79, USA, 2004.

[51] Cooper, S., Dann, W., and Pausch, R., "Alice: A 3D Tool for Introductory Programming Concepts", Journal of Computing Sciences in Colleges, Volume 15, No. 5, pp. 107-116, USA, 2000.

[52] Sloan, R.H., and Troy, P., "CS 0.5: A Better Approach to Introductory Computer Science for Majors", ACM SIGCSE Bulletin, Volume 40, No. 1, pp. 271-275, USA, 2008.

[53] Guzdial, M., "A Media Computation Course for Non-majors", ACM SIGCSE Bulletin- Proceedings of 8th Annual Conference on Innovation and Technology in Computer Science Education, Volume 35, No. 3, pp. 104-108, USA, 2003.

[54] Freund, S.N., and Roberts, E.S., "Thetis: An ANSI C Programming Environment Designed for Introductory Use", ACM SIGCSE Bulletin, Volume 96, pp. 300-304, USA, 1996.

[55] Lister, R., Adams, Elizabeth S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostrom, J.E., Sanders, K., and Seppala, O., Simon, B., and Thomas, L., "A Multi-National Study of Reading and Tracing Skills in Novice Programmers", ACM SIGCSE Bulletin, Volume 36, No. 4, pp. 119-150, USA, 2004.

[56] DeLeon, M., Espejo-Lahoz, M.B., and Rodrigo, M.M.T., "An Analysis of Novice Programmer Doodles and Student Achievement in an Introductory Programming Class", Philippine Information Technology Journal, Volume 1, No. 2, pp. 17-21, Philippine, 2008.

[57] Hulkko, H., and Abrahamsson, P., "A Multiple Case Study on the Impact of Pair Programming on Product Quality", Proceedings of 27th International Conference on Software Engineering, pp. 495-504, USA, 2005.

[58] Gao, H., Qiu, Z., Wu, D., and Gao, L., "Research and Reflection on Teaching of C Programming Language Design", Intelligent Computation in Big Data Era, Communications in Computer and Information Science, Volume 503, pp. 370-377, USA, 2015.

[59] Xinogalos, S., "Using Flowchart-Based Programming Environments for Simplifying Programming and Software Engineering Processes", IEEE Conference on Global Engineering Education, pp. 1313-1322, Germany, 2013.

[60] Carlisle, M.C., Wilson, T.A., Humphries, J.W., and Hadfield, S.M, "RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving", Proceedings of 36th SIGCSE Technical Symposium on Computer Science Education, pp. 176-180, USA, 2005.

[61] Carlisle, M.C., Wilson, T.A., Humphries, J.W., and Hadfield, S.M., "RAPTOR: Introducing Programming to Non-Majors with Flowcharts", Journal of Computing Sciences in Colleges, Volume 19, No. 4, pp. 52-60, USA, 2004.

[62] Carlisle, M.C., "Raptor: A Visual Programming Environment for Teaching Object-Oriented Programming", Journal of Computing Sciences in Colleges", Volume 24, No. 4, pp. 275-281, USA, 2009.