THINK INDIA JOURNAL

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Challenges in Teaching and Learning Process for Programming Language Courses – A Survey

[1]Jayamary.A, [2]Thaddeus.S
[1]Assistant Professor
[2]Associate Professor
Don Bosco College, Yelagiri Hills

Abstract: Programming is a skill that students of Computer Science need to learn yet struggle to learn. If students are to master the skill, the root cause for the difficulties in learning must be addressed. In this context, a systematic literature survey is done to study the work carried out by researchers to find out the difficulties faced by students and faculty while learning or teaching programming. Focusing on the problems from the teacher's perspective and analyzing the literature, the authors have ascertained the most common problems as curriculum, selection of introductory programming language, pedagogy, literature and tools for teaching programming. The authors also propose an intelligent tutoring system that will support students in providing resources and exercises to learn programming and will ensure faculties teach programming effectively.
Keywords: Problem solving methods, programming language teaching, critical thinking, rote learning, comprehension

1. INTRODUCTION

Learning to solve problems using programming languages is difficult for majority of Indian rural students. Having been in the field of teaching for more than fifteen years, the authors observe that students' interest in solving problems using programming languages is decreasing. In the analysis done, it was found to be true especially among women students of rural areas in Vellore district of Tamil Nadu, India. Getting into the routine, facing the challenges in learning and applying programming language, many students do not succeed. Trying to analyze the root cause of it, it is observed that the problem is not with students but the pedagogy and method of study. The faculty is not able to attend to a large class `of fifty students at a time because of which students resort to learning without understanding. Rote learning is more applied than comprehension and critical thinking. Researches are being conducted to uncover the reasons for being unable to solve problems using any programming language and the increase in the number of dropouts.

This article is divided into four sections. The first section is the introduction. The second section deals with the context and earlier studies that discuss the challenges in learning programming languages. Section 3 deals with the deployed approaches proposed by researchers, Section 4 deals with Discussion and Section 5 deals with the conclusion.

2. THE CONTEXT AND THE EARLIER STUDIES

Computer Science teachers have always been challenged with the task of teaching programming languages interestingly. Unlike the teaching of theory subjects, programming language teaching involves teaching the concept, making the students understand the concept and go to the extent of applying the concept to problem solving. Often it is found that students understand the concept but are unable to apply it to new problems. Exploring the reasons for this, we find many studies are undertaken to unearth the learning difficulties among students. Pears classifies the difficulties as Curricula, pedagogy, tools for teaching and language choice, while the study by Lahtinen, Robins, Milne and Rowe, Goldman and others include concept-related difficulties. Based on these classifications, few problems are discussed below

2.1 Curriculum and Selection of Language

Today there is a wide variety of programming languages to solve problems. Many programming languages such as C, C++, Java and Visual Basic are used as introductory programming courses. Constant debates among computing educators exist over the usage of Structured versus Object-Oriented driven curriculum. The analysis done by P.

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Channa Reddy [1] exemplifies that though procedure-oriented programming languages appeal directly to human thought process as the human beings are used to solving problems in a procedural manner, Object oriented programming is more suitable for solving complicated problems. Azad Ali and David Smith[2] propose an introductory course that must not only be a pre-requisite but also prepare the students for advanced courses of programming or any other courses. This introductory course should be open to students of other departments as well. At times writing a program with the proper syntax can become a melancholy as students who are new to programming need to write the code with proper spelling, correct number of braces, quotes or semicolons, which if not done correctly may lead to syntax errors. The authors in their study quote four reasons for this melancholy namely strict syntax, unknown structure, time required to develop a program, and working individually. To overcome this, the authors introduced Alice programming language as the environment helps students to code without having to worry about semicolons or curly braces and allows the students to focus on the concepts. The software provides a multimedia environment that makes the students feel comfortable and easy to create, use and manipulate objects. Alice helped students to view the results anytime during development and the visual output seen on the screen made the students enthusiastic about their learning. When the student views the objects moving on the screen, they even forget that they are working individually. But the difficulty with this language was that it was purely a learning tool. It was not used to develop applications like payroll or any other. So the entire study duration was divided into two phases. In the first phase, Alice was taught to students during the time of which students liked the idea of programming and developed interest in coding. In the second half of their study, students transited from Alice to learning Java by using the interface of Alice. The result showed that the interest of students in learning coding was still evident when shifting from Alice to Java, which may have been missed if Java was taught as an introductory programming language.

A research by Natalya Prokofyevaa et al.[3] reveals that many of the institutions still use the traditional structural programming (Pascal / C) as introductory languages proceeded by object-oriented programming (Java, C++) programming language. Though procedural languages are a good start, one has to keep in mind that computer science graduates would be working in the IT industry in three or four years from their first year and hence cannot be taught a language which is no more used in the industry. A reasonable selection is the Object-oriented approach, which could help in the Internet application development. The authors ascertain that this Object oriented programming can either be Pascal (with Delphi, its visual programming environment or Lazarus, the free environment) or Java – a fully object-oriented language that support in developing platform-independent applications with the help of integrated development environment Eclipse. The authors conclude that the initial programming language could be Java or C++. Raina Mason Graham Cooper and Michael de Raadt [4] in their study show that there are a number of languages available but the selection of it by students has variations down the years. Java had been selected by 43.9% of students in 2001, while it had decreased to 39% in 2010. However Python which was 0% in 2001 had been increased to 19.5%. The reason for change was the higher use of the language in the industry and the benefits of language. The languages were selected because of its extendibility, libraries, online and community help available. These factors led to better teaching of the language, which in turn increased the number of students who opted for the course.

Theodora koulouri et al. [5] throw insight on the language to be used as introductory language based on (i) the basic programming constructs such as loops, conditional statements or libraries and (ii) the bugs that could fall under the category of semantic or syntax errors. Python was introduced as the first language as it was found to be simple, expressive and popular in the industry with a lower overhead. When a program is written in Java, we needed access modifiers, classes, methods, keyword static, type and array but these were not required in Python. When a student had written a piece of code, the faculty had immediately given a formative feedback on the code along with a reference to the lecture material regarding the concept. This was followed by the problem-solving training sessions for students. This method was used with two groups. The group that was exposed to Python first learnt programming better than the other group.

Linda Mannila Turku and Michael de Raadt [6] make a comparative study of various factors that are to be considered while introducing a language. The authors articulate that a language is to be checked for its suitability of teaching, representation for physical correlation, offering of a framework, design and environment, its interactive and rapid code development environment, its support for user community, its availability for free of cost and its extensibility beyond education and reliance and efficiency. Based on these factors, the authors go on to prove that

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Eiffel and Python score higher (15 points) when compared to other languages like Java(14) and Visual Basic (6). Based on the usefulness of the language in industry, Python is proved to be the most preferred introductory language. Kasurinen J., and Nikula, U [7] in their study on the reasons for drop out, prove that Python suits the learning needs of novice programmers better than any other language.

Another study made by D. Krpan, I. Bilobrk [8] on the difficulties faced by students while learning programming found that the choice of language played a vital role in determining the learning of programming. The study conducted on students of Croatian Universities, shows that one group of students learnt C as the introductory language while another group learnt QBasic and another group Python. A comparison was done that concluded that students felt QBasic was much simpler when compared to C or Python with regard to Data types, input-output, loops, arrays, functions and files. In their study, groups were given 3 questions and were asked to choose the language of choice. It was found the group that had learnt QBasic and Python had solved the problems at a success rate of 75% while C had 35% success rate. The study concludes that though Python is a promising language, QBasic was much preferred by students as it was simpler and more effective. The study by Katrin Becker[9] reports the return of the computer science department to Pascal after experiencing problems teaching C and C++ in the introductory programming course. The study shows how the language chosen for an introductory programming course does not necessarily have to be one of those currently most widely used. Programming language selection should be based on suitability and purpose and not just the language's advantages and disadvantages. Whatever be the language being taught, the curriculum should be student-centric and not teacher-centric [10] i.e the outcome of learning a language should clearly focus on students and what the student is capable of after having learnt the language.

2.2. Pedagogy

The traditional classroom teaching involving chalk and talk is replaced with audio-visual presentations. Teachers in general use visual presentations to explain a concept, often missing out the descriptive walk-through on the board. This makes it difficult for slow learners to understand how a program function. In one of the study by M.Sivasakthi and R. Rajendran [11], most students claimed that they could learn programming more effectively in practical Laboratory sessions rather than theory. Discussion with others might be handy when the exercise is to be done within a small group. Practicing could be effective in learning programming as students rated that working alone on programming course work is better than lectures. The study also shows that many first-year students of colleges in India find it difficult to learn Java programming because until their secondary education, medium of instruction has been vernacular language. Switching to English made it difficult for them to learn the language. The study also ascertains that students found certain topics as Concurrent programming, User Interface components with Swing and Generic programming to be the most difficult topics to learn. It was also observed that students felt that lesser real-world examples, the teaching pedagogies used by faculty, lack of motivation on the part of students to learn Java, less conducive environment were other factors that made learning of language difficult. As SitiRosminah and Ahmad ZamzuriMohamadAli [12] points out the world is moving towards the use of computers in all fields and this has paved the way for the need of technical experts be it software developers, testers or database administrations. In order to prepare the students for such jobs, the educational institutions (assuming) have the required personnel to teach the students. But the question is "Are the students being taught the skill set required for the industry?" The authors have made a study, based on the work already done by T. Phit-Huan, T. Choo-Yee [13] and I. Milne and G. Rowe[14]. The survey is done in two parts. The first part takes into account the students' general information and programming experience, while part 2 takes into account the students' programming experience that includes level of understanding of different topics, type of problems encountered, tools used for teaching programming and factors that could increase the interest of programming. The study proved that majority of students had a moderate level of understanding of basic concepts, but exhibited difficulties in concepts of multidimensional arrays and data structures and were unable to apply their knowledge on problems. The students felt that lesser practical examples were given by faculty and that the learning environment was not conducive as many a times the systems were not functional. A study by Gomes et al. [15] depicts that teachers are more focused on teaching the syntax rather than problem solving skills. Teachers teach dynamic concepts using static materials, which makes it difficult for students to know the dynamics of programming. The authors also observe that the pedagogies are not personalized and the strategies don't support all learners. The study done by Antony Robins and Janet Rountree [16] focuses on the problems faced by faculty teaching programming languages. The authors categorize programmers as novices and experts. The

ISSN:0971-1260

Vol-22-Issue-10-November-2019

review found that novice programmers viewed programming as knowledge-driven subject while the experts viewed it as a skill. The expert programmers viewed code line by line and ran it except for unusual programs which they learnt in long intermittent runs. Novices read both conventional and new programs in the same way. The focus of experts was on refined knowledge depiction and problem-solving strategies that could be applied for any problem to be solved. Experts were good at recognizing, adapting and using blueprints of structure but this was not found in novices. The authors conclude that the faculty has to first identify novice learners and in turn categorize them as effective and ineffective novices, then formulate pedagogies to turn ineffective novices to effective ones. This can be done only by modifying teaching pedagogies based on the type of programmer he/ she is.

2.3. Student's capabilities and behavior

Students generally apply the same studying methods for both theory and practices. Perkins et al. [17] classified the beginners in programming into 3 categories namely stoppers, movers and extreme movers. Stoppers are students who lack focus and when challenged with a problem, they just give up, while movers are group of students who use feedbacks about errors to move on with programming making better progress in their work. The third category is extreme movers, who make very little changes and like the stopper group, show no progress in their work. Applying the categories in today's context, we find the ratio as 6: 3:1. Majority of the students [Ratio of 6] do not know to solve problems. The reason behind this could be because of deficiency of mathematical and analytical skills as showed by M.de Raadt and M. Toleman [18] or because the staff would have focused on teaching their programming concepts rather than developing problem solving abilities[15]. In the study made by Robins and Rountree [16], novices had superficial knowledge where the focus was on the syntax. The attitude of the novices was more of "Quick-Fix" where they spent little time testing code and attempted only small fixtures rather than reforming programs. The review also related the programmers to the type of languages used. It ascertains that Object oriented programming was easier for expert, but the nature of control flow and function made the novices assume that programming was difficult.

D. Adair and M. Jaeger [19] categorize the general difficulties faced by students as cognitive difficulties, programming difficulties and image difficulties. The cognitive difficulties include the teaching strategies such as lectures, tutorials or lab exercises, learning style as understanding or memorizing, motivational factors such as genuine interest, career prospects and family pressure. The image difficulties include the interest to learn, the reputation of the organization where student is studying and the pace of instruction. The programming difficulties included certain topics such as exceptions, GUIs and files that were found to be difficult by majority of the students. Yet another study made by Jenkins [20], proves that the pedagogies of a faculty may not be supportive to all students because every student is unique and the methods that appeal to each of them could be different. Another study was made by Josephat O. Oroma, Herbert Wanga [21] on the challenges faced by students in learning programming languages. The study observes that Programming is a new ground in developing countries like Tanzania and the students are exposed to it only when they join the university. He has also observed that the students needed extrinsic motivation and had lower intrinsic motivation. The students also lack self-efficacy. Students with self-efficacy may undertake challenging tasks, take extra efforts to achieve their goals and persevere towards it in spite of difficulties. The method used by students had only been memorizing which forced them to either by heart or copy programs. Students lacked analytic skills to analyze problems and they had also been transferred the fear to program, making them believe that programming was individualistic and not collaborative. Due to this, they get disappointed and eventually give up programming.

2.4. Course concept difficulties

Du Boulay[22] identifies five major problems with the novices. The first is the problem of orientation. The students lack the understanding of what a program is and why it could be used. The second problem is the notional machine. Students lack the understanding of the general features of computer and the relation between the physical and notional machine. Students lack the understanding of the actual execution of a program. The author suggests that students should be given a model of description of machine. The third problem is the notation of formal programming language which deals with the syntax and semantics. The students need help in writing the syntax correctly especially during the learning of introductory programming. The fourth problem is that of Acquiring structures ie students have difficulty in certain difficult concepts such as Recursion [23], [24] Arrays [25], Advanced

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Data structures [26], Schulte and Bennedsen[27], Algorithms [28],[29]. The fifth problem is that of the pragmatics of Programming i.e designing a program is a challenge to many of the students. Many a time students don't understand what is expected of them. At times even when they understood the problem, some were not able to convert that understanding to an algorithm [25]. Another general problem that Robins et al. [24] speaks of is the lack of analytical and problem-solving skills among the students. This proof has been supported by many other researchers [15], [30], [31]. Table1 showcases a few problems along with the solutions proposed.

Table 1 Problems and solutions

References	Problem	Solution
Robins et al.[24]	Superficial knowledge and lack hypothesis	Provide simple learning materials
Du Boulay [22]	Problem of orientation and program's syntax and semantics	
Sandra Milena[31]	Selection of introductory programming language	Begin with Object oriented programming
Koulouri et al. [5]	Introductory Language Selection	Python could be used as the introductory language
Goldman et al. [23] Lahtinen et al. [26] Robins et al. [24] Schulte and Bennedsen [27]	Course-related difficulties - Recursion	More practical exercises could be given for learning. Since the difficulties are linked to advanced individual entities, learning resources, to develop program creation, updating or debugging, could be provided Engage students in problem -solving activities to learn few concepts at a time
Goldman et al. [23] Schulte and Bennedsen [27]	Inheritance	
Milne and Rowe [14] Schulte and Bennedsen [27]	Polymorphism	
Lahtinen et al. [26] Schulte and Bennedsen [27]	Advanced data structures	
Milne and Rowe [14] Lahtinen et al. [26] Milne and Rowe [14]	Operator overloading Pointers	
Garner et al.[25], Meisalo, Suhonen et al.[41], Robins et al. [23]	Arrays	
Garner et al. 2005; Milne and Rowe 2002; Robins et al. 2003)	Constructors	
Seppälä et al. [28] Xinogalos et al. [29] Williams et al. [42]	Algorithms Program design – understanding the task and working out an Algorithm / solution to solve the	Practical Problem-solving tasks could be given to students paired or collaboratively

ISSN:0971-1260

Vol-22-Issue-10-November-2019

	same	
	Lack of Mathematical and analytical skills	Introduce a preliminary curriculum based on mathematical foundations such as Predicate calculus, Boolean Algebra and establishing formal proofs of program correctness. (Dijkstra [44]) Teach problem solving skills rather than focusing on syntax Introduce algorithmic thinking without a computer. Students must be coached in analyzing their intuitions and connecting them to designated tasks (Weigend [43])

3. SOLUTION APPROACH

Several approaches address the issues at different levels of the taxonomy. Brooks was one of the predecessors who proposed a model of program comprehension [32],[33]. His model is set in the perspective of three domains namely the problem domain where the problem is proposed, followed by the intermediate domain where the problem is analyzed and converted to structures and values and finally the programming domain where the problem is symbolized as data structures and algorithms. Another model of comprehension [34],[35] and [36] contended that the subjects' model of a program can be propelled by various tasks, like for example, manipulating a program rather than responding to queries relating to it.

Rist [37] offers another complete model of program generation. In this model, Knowledge is represented as points in internal memory (regular, intermittent, and semantic) or external memory (the program design, transcripts or the program itself). A point translates an "action" that may be represented as a line of code, or chunks of code like loops, to one or more methods of arbitrary size. Points are indexed using a record of the format (function, target, instance), for example, a loop to read could be indicated as (read, stream,-). Points also have four "ports", : practice, create, conform and govern, which permit them to be interconnected with respect to the flow of control and flow of data. He remarks that there is very little association between the capability to script a program and the capability to read a program. Since both are important, both need to be imparted.

A few others have proposed solutions based on models and processes conceived. Writing a program involves maintaining different kinds of “Mental models”. Innumerable researches have observed the dominant role played by a model of the computer, often referred to as a "notional machine" [22]. This notional machine is an abstract computer whose properties are inferred from the concepts used by the programming language. Du Boulay [22] conceives a replica of a program based on the script of the program. For example, looking at an illustration, a student will be able to understand how a steam engine functions. Soloway and Spohrer [34], [35] also suggest the use of graphical representations to describe control flow and the states of the program better. Jerome S Bruner [10] points out that the teachers should establish a link between himself and students. Learning should be spiral shaped. They should be able to learn basic concepts, move from basic to advanced concepts and from advanced to new things in life. Students should be helped with simple learning resources which could later be extended when needed [24]. Hromkovic [38] views programming as a skill to commune a set of directives to a mindless computer. To ensure this is done, basic programming could be imparted through simple structures so that Students are able to concentrate on semantics and thereby make lesser mistakes. Another way is to use practical examples. The teachers can start with simple and then move to complex programs.

ISSN:0971-1260

Vol-22-Issue-10-November-2019

To overcome difficulties linked to course-related concepts, learning resources to develop program creation, updating or debugging could be provided. Practical Problem-solving tasks could be given to students paired or collaboratively [42]. Teachers should focus on imparting problem-solving skills rather than syntax [15]. The students should be engaged in problem -solving activities to learn few concepts at a time. Students should be taught to convert their intuitions to programming design [43].

Programming cannot be taught today using only the traditional chalk and talk method. Programming tools are specifically designed to assist the learners learn programming languages. Teachers use digital learning resources to combine text, image, video and audio. However not all tools can cent percent be relied upon. As observed by Pears [39] there are tools for visualization, programming environment tools and automatic assessment tools. He observes that the difficulty with this is students use the dynamic visualization tools but without understanding the in-depth meaning of the algorithm. There are other tools to analyze program features. Luck and Joy[40] describe the use of a tool BOSS used in the University of Warwick, which verifies the accuracy of the program execution by equating the output of a student's program with that of the solution model defined by the teacher. There are also other tools that evaluate internal data representations [22] or testing of program [33],[34]. Teachers use many tools for assessing the students' program, which the students feel, help them, in formative assessment and learn better. However they feel that immediate feedback could be very helpful in learning programming.

E-learning has become popular among the students, learning is not restricted to classroom. There exists many websites which support students in learning programming. One such site is Codecombat.com which helps students to learn programming by first constructing a game and then playing the same. Codeacademy is another website that supports students with collaborative lessons submitted by program learners. Another tool is Cloud9 which is an open source integrated development environment (IDE). It provides attributes for both program writing and integrating the programs of students with that of the staff so that they find out how each student does the given task. Colaboratory is another E-learning tool for learning programming which provides interface to learn python programming.

4. DISCUSSION

The goal of this article is to study the problems related to teaching or learning of programming languages. The search for articles was made based on keywords as "Challenges in teaching / learning of programming languages", "Challenges in software Engineering Education", "Difficulties in teaching programming languages", "difficulties in learning programming languages" and "Problems and solution approaches for teaching programming languages". As Table2 depicts, of the 94 articles considered, 73 were appropriate to the topic. The authors categorized them into articles dealing with problems and solutions and review articles. Around 20 articles were on literature review while 43 discussed the problem with solution approaches.

Table 2 - Articles reviewed

Area	Considered	Reviewed(Appropriate)	Included
Curriculum	25	15	10
Pedagogy	20	18	06
Students' Capabilities and Skills	05	10	04
Teaching tools	20	15	14
Course-related problems	23	15	13
Total	93	73	47

ISSN:0971-1260

Vol-22-Issue-10-November-2019

Reviewing the articles, it is found that problems arise from not just a single factor but a combination of factors. This article reports results that are consistent with findings identified in the reviewed literature. Elementary problems relating to basic program syntax or advanced concepts like inheritance, pointers, constructors and data structures do exist. Experience shows that students should be taught programming with multiple exercises and activities that stimulate thinking skills. The teaching faculties need to focus on problem solving skills rather than the language syntax. Teachers must be encouraged to teach programming with personalized pedagogies to support all learners. This involves a lot of work on the part of the faculty and not every staff may be passionate about spending time for designing different pedagogies for different types of students.

One study of exposing students to a user-friendly programming environment is good but this cannot be applied to all university-affiliated institutions in India as the curriculum is decided by the university and the time period of 90 working days (One semester) may not be enough to make students learn a language and then shift to another one. Another observation on the variations in the selection of languages by students does not hold good in today's context. According to Tiobe index-2019 (Tiobe uses 25 search engines to calculate the ratings of the programming languages across the world), Java still occupies 1st or 2nd place in the list of programming languages, followed by C, Python, C++, C#, Visual Basic.Net, JavaScript and Ruby as shown in Table 3. Again students' selecting the language as per their choice is possible if "choice-based credit system" is implemented strictly.

Table 3 - Ratings of Programming languages

Rank	Programming Language	Ratings
1.	Java	16.884 %
2.	C	16.180 %
3.	Python	9.089 %
4.	C++	6.229 %
5.	C#	3.860 %
6.	VisualBasic.Net	3.745 %

More than the programming language, the studies prove that students can perform better if they are taught problem solving skills than focusing on the syntax. At the beginning, the teacher starts to teach problem solving techniques but towards the end due to lack of time, they hasten to complete the syllabus. Students learn programming better in practical laboratory sessions than in theory sessions. It is observed that still in many areas of rural Tamil Nadu, most students use the programming laboratory sessions as typing sessions, executing the plagiarized code while only a few students really use it for trying out different logic of the code.

5. CONCLUSION

This article attempts to give an overview of research issues relating to the teaching and learning of programming languages. The authors discuss problems from the factor-specific research in each of these areas. It is found that there are different approaches to teach programming and one can never conclude that a particular approach is cent percent suitable for teaching programming. The authors attempt to inspire the staff in designing the course with suitable use of tools and technologies. Staff must be virtually present 24 X 7 to guide a student in learning programming and this is possible only with the development of a student-centric intelligent tutoring system supported by audio visual aids and auto assessment. Efforts are being taken to develop an intelligent tutoring system that will help students to learn problem solving using programming languages effectively and easily.

References

[1] Chenna Reddy, "Analysis of Teaching Computer Programming in Indian Context", Journal of Engineering Education Transformations, 2015

ISSN:0971-1260

Vol-22-Issue-10-November-2019

[2] Azad Ali and David Smith, "Teaching an Introductory Programming Language in a General Education Course", Journal of Information Technology Education: Innovations in Practice, 2014

[3] Prokofyeva, N., Uhanova, M., Katalnikova, S., Synytsya, K., &Jurenoks, A. (2017). Introductory Programming Training of First Year Students. Procedia Computer Science, 104, 286–293. doi:10.1016/j.procs.2017.01.137

[4] Raina Mason Graham Cooper, Michael de Raadt, "Trends in Introductory Programming Courses in Australian Universities – Languages, Environments and Pedagogy", Australasian Computing Education Conference (ACE2012)

[5] Theodora koulouri "Teaching Introductory Programming: a Quantitative Evaluation of Different Approaches", ACM Transactions on Computing Education (TOCE), February 2015

[6] Linda Mannila Turku and Michael de Raadt, "An Objective Comparison of Languages for Teaching Introductory Programming", Baltic Sea Conference on Computing Education Research, Koli Calling

[7] Kasurinen, Jussi & Nikula, Uolevi, "Lower dropout rates and better grades through revised course infrastructure", International Conference on Computers and Advanced Technology in Education, 2007

[8] Krpan D, I. Bilobrk, "Introductory Programming Languages in Higher Education", MIPRO International convention, 2011

[9] Katrin Becker, 'Back to Pascal: Retro but not backwards', Journal of Computing in Small Colleges, 2002

[10] Jerome S Bruner, "The process of Education", Harvard University press, 1999

[11] Sivasakthi M and R. Rajendran, "Learning difficulties of 'object-oriented programming paradigm using Java': students' perspective", 2011

[12] SitiRosminah and Ahmad ZamzuriMohamad Ali, "Difficulties in learning programming: Views of students", International Conference on Current Issues in Education, ICCIE 2012

[13] T. Phit-Huan, T. Choo-Yee, and L. Siew-Woei, "Learning difficulties in programming courses: Undergraduates' perspective and perception,",International Conference on Computer Technology and Development, vol. 2, pp. 42–46, 2009.

[14] Iain Milne And Glenn Rowe, "Difficulties in Learning and Teaching Programming—Views of Students and Tutors", Education and Information Technologies 7:1, 55–66, 2002

[15] Gomes, Anabela, Mendes, A. J. "Learning to program - difficulties and solutions", 2007, Coimbra, Portugal, International Conference on Engineering Education(ICEE) 2007, Coimbra, Portugal

[16] Anthony Robins, Janet Rountree, and Nathan Rountree, "Learning and Teaching Programming: a review and discussion", Computer Science Education, 2003

[17] D. N. Perkins, Chris Hancock, Renee Hobbs, Fay Martin and Rebecca Simmons, "Conditions of learning in novice programmers", doi: 10.2190/GUJT-JCBJ-Q6QU-Q9PL
http://baywood.com

[18] Perkins, D.N. Hancock, C.Hobbs, Martin F and Simons R, "M.de Raadt, R. Watson, and M. Toleman. "Language Trends in Introductory Programming". In Proceedings of Informing Science and IT Education Conference. Informing Science.org, June 2002

[19] D. Adair and M. Jaeger, "Difficulties in Teaching and Learning the Java Programming Language", International Conference on Engineering Education (ICEE) 2011

[20] Tony Jenkins, "The Motivation of Students of Programming", Proceedings of Conference on Innovation & Technology in Computer Science Education 2001, pp 53-56

[21] Josephat O. Oroma, Herbert Wanga, Fredrick Ngumbuke, "Challenges of teaching and learning computer programming in developing countries", Barcelona conference (EDULEARN-2012)

[22] Benedict Du Boulay, "Some difficulties of learning to program" , doi: 10.2190/3LFX-9RRF-67T8-UVK9, http://baywood.com

[23] Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M. C., & Zilles, C. (2008), "*Identifying important and difficult concepts in introductory computing courses using a Delphi process*". *ACM SIGCSE Bulletin, 40(1), 256*. doi:10.1145/1352322.1352226

[24] Anthony Robins, Patricia Haden, Sandy Garner, "Problem Distributions in a CS1 course", Australian Computer Society, Inc. Eighth Australasian Computing Education Conference (ACE2006), Hobart, Tasmania, Australia, 2006.

[25] Sandy Garner, Patricia Haden, Anthony Robins, "My Program is Correct But it Doesn't Run: A Preliminary Investigation of Novice Programmers' Problems", Australasian Computing Education Conference 2005, Newcastle, Australia

[26] Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education - ITiCSE '05. doi:10.1145/1067445.1067453

[27] Schulte, C., & Bennedsen, J. (2006). "*What do teachers teach in introductory programming"? Proceedings of the 2006 International Workshop on Computing Education Research - ICER '06*. doi:10.1145/1151588.1151593

[28] Seppälä, Otto & Malmi, Lauri & Korhonen, Ari. (2005). Observations on student errors in algorithm simulation exercises. 81-86., Proceedings of the 5th Annual Finnish / Baltic Sea Conference on Computer Science Education

[29] Xinogalos, S., Satratzemi, M., and Dagdilelis, V. (2006). "An introduction to object oriented programming with a didactic microworld: Object Karel", https://doi.org/10.1016/j.compedu.2004.09.005

[30] Romeike R. (2008) What's My Challenge? The Forgotten Part of Problem Solving in Computer Science Education. In: Mittermeir R.T., Sysło M.M. (eds) Informatics Education - Supporting Computational Thinking. ISSEP 2008

[31] Saeli, Mara & Perrenet, Jacob & Jochems, Wim & Zwaneveld, Bert. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. Informatics in Education. 10. 73-88.

[32] Brooks R.E, "Towards a theory of the comprehension of computer programs, 1983

[33] Brooks R.E, " Categories of programming knowledge and their applications", 1990

[34] Soloway E and Spohrer J C, "Studying the novice programmer", 1989

[35] Spohrer and Soloway E, "Novice mistakes: Are the folk wisdoms correct", Communications of the ACM, July1989, Volume 29, Number 7

ISSN:0971-1260

Vol-22-Issue-10-November-2019

[36] Letovsky S, "Cognitive processes in program comprehension", Journal of systems and software, 325-327,1986

[37] Rist R.S, "Program structure and design: Cognitive science", 1995

[38] Hromkovič J. (2006) Contributing to General Education by Teaching Informatics. In: Mittermeir R.T. (eds) Informatics Education – The Bridge between Using and Understanding Computers. ISSEP 2006. Lecture Notes in Computer Science, vol 4226. Springer, Berlin, Heidelberg - DOI https://doi.org/10.1007/11915355_3

[39] Arnold Pears and Stephen Seidman, "A Survey of Literature on the Teaching of Introductory Programming",Conference on Innovation & Technology in Computer Science Education 2007

[40] Mike Joy, Nathan Griths and Russell Boyatt, "The BOSS Online Submission and Assessment system", ACM Journal on Educational Resources in Computing, Vol. 5, No. 3, September 2005, Article 2.

[41] Meisalo, V., Suhonen, J., Torvinen, S., & Sutinen, E. (2002). *Formative evaluation scheme for a web-based course design. Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE'02*. doi:10.1145/544414.544454

[42] Williams, Laurie & Yang, Kai & Wiebe, Eric & Ferzli, Miriam & Miller, Carol. (2002), "Pair Programming in an Introductory Computer Science Course: Initial."

[43] Michael Weigend, " Michael Weigend", International conference on Informatics in Secondary Schools - Evolution and Perspectives: The Bridge between using and understanding Computers, Pages 117-126, 2006