Classification of Cognitive Difficulties of Students to Learn Computer Programming

Renumol V. G., Jayaprakash S., Janakiram D.
Department of Computer Science,
Indian Institute of Technology,
Madras, India.

Abstract

Teaching and learning of computer programming is mentioned as one of the grand challenges in Computing Education. A study has been conducted in this regard, to analyze the difficulties of students in programming education. This study analyses the data collected from a group of engineering undergraduates. The respondents have mentioned programming difficulties at various granularities and levels. It has been observed that they have problems in all the three domains of Bloom's Taxonomy. But majority of their difficulties are in cognitive domain. The affective domain and psychomotor domain problems listed were trivial and less in number. It has been further observed that students have cognitive difficulties at various levels, during programming and program comprehension, independent of the programming language used. These difficulties are mapped to various levels of Bloom's taxonomy in cognitive domain – Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. This mapping helped to identify some of the cognitive processes involved in programming. Strengthening of these cognitive skills in students may improve the effectiveness of programming education. This study is an initial step of an ongoing research to develop a cognitive model for programming education.

Keywords: *Computer Programming Education; Cognitive Difficulties; Bloom's Taxonomy.*

1. Introduction

Programming is the process of writing, testing and debugging of computer programs using different programming languages. For this, a programmer requires declarative and procedural knowledge. The former is the knowledge of programming language syntax and semantics which in turn needs memorization and comprehension abilities; whereas the latter is problem solving and program design skills which in turn needs additional skills like abstraction and logical thinking, and domain knowledge. So programming is a complex task and needs multiple skills and knowledge. Various publications in programming education show that the failure rate and drop-out rate of programming courses are relatively high [14][16] and overall effectiveness is poor.

The students in programming course can be categorized as effective and ineffective based on their effectiveness in programming. Effective students can write programs and they typically learn programming with moderate effort. Whereas, ineffective students cannot write correct programs and need more personal attention and cognitive support to learn programming. Since the failure rate is high, the ineffective category plays a significant role in the effectiveness of programming courses.

On the other hand, teachers also find it difficult to design an instructional system for programming courses so that it would be effective for a large group of heterogeneous students (in undergraduate courses, some students would have learned basics of programming during secondary school whereas some would be very new to computer). This issue of scalability and heterogeneity is a challenge for programming educators, which necessitates an effective and efficient teaching strategy along with learning materials and tools. But designing a better programming course needs an in-depth knowledge of the learning style of the students and the problems they face during programming. In this study, an attempt has been made through data collection and analysis to identify the issues in learning programming and the general pattern currently used to learn programming. For this, a questionnaire was designed with relevant questions and distributed for data collection among a group of engineering undergraduates. The data analysis shows that, though students have difficulties in cognitive, affective and psychomotor domains, the cognitive domain problems predominate. The result of this study will provide a starting point for further research on cognitive processes during programming and some directions to improve the computer programming education.

The organization of the paper is as follows. Section 2 provides a literature survey on programming education. Section 3 describes the conceptual framework. Section 4 deals with the methodology. Section 5 explains the results and section 6 is the discussion on the study. Section 7 concludes the paper with future work.

2. Literature Survey

Teaching and learning of programming is mentioned as one of the seven grand challenges in Computing Education [14]. Research has been going on in improving the effectiveness of Programming Education, but the results are not scientifically proven or globally accepted [4][5][7][12][13][17]. It is obvious that these anecdotal evidences are not pedagogically reasonable.

From various literatures [4][10][16], it is identified that the following are the main resources for novices to learn programming:

1. Classroom teaching
2. Lab session (Practice using computer)
3. Programming text books/Learning material

In class room teaching, the teacher spends a significant amount of time in teaching the program constructs of a specific language and gets less time to expose the actual programming process [1]. The problem with this method is that students with good comprehensive ability and logical thinking will survive but the other students find it difficult to bridge the gap between language constructs and program construction.

The results of psychological studies in computer programming expertise show that turning a novice into an expert is impossible in a four year undergraduate programme, but competence is possible by practice [20]. Thus lab sessions play a significant role because of the importance of practice in learning programming. But in lab sessions also, some teachers will start giving large programs as assignments to novices [4] rather than starting from small and simple programs. This non-systematic teaching will increase complexity in students' mind, since brain has a natural tendency to learn in an incremental way [5]. In addition to that, novices may not get sufficient individual-feedback during lab session. During the programming task learners receive relatively high levels of feedback on low level issues, such as syntax rules, but tend to receive low levels of feedback on conceptually more difficult issues [3].

The third main resource is programming text books. Since programming is a dynamic process, “program” text books being a static media are ill suited to expose the process of programming [1][10]. They can be used just for knowledge acquisition and they do not reveal the program construction process.

The literature shows that each of the three learning resources has its own drawbacks. Addressing them independently may not solve the issues. During programming one needs to solve the given problem and then translate the solution to a sequence of computer language instructions. Though programming is a human activity, many programming languages are designed from machine point-of-view, considering machine efficiency related issues rather than human-centered issues [18]. Therefore, transformation of a problem solution to a program takes more cognitive effort from the programmer. It requires different cognitive skills like comprehension, abstraction, analysis, problem solving, debugging etc. [8] [21]. This adds the difficulty for the novices to learn programming. Also, the learning curve for different programming languages varies based on the language and the cognitive characteristics of the learner [19].

From the literature study, it is clear that there is a need to improve the effectiveness of programming education. Existing literature has many anecdotal evidences of pedagogical strategies for programming education. They may add teaching load, time and content to the already loaded curriculum. Most of the time, programming is a self-discovered process for students. Even though programming is a cognitive process, the current teaching approaches are hardly ever based on any cognitive models. Investigating the issues from the learners’ perspective may lead to better results in programming education. A data analysis on students’ difficulties during programming and their learning styles may give an insight into the real issues.

3. Conceptual Framework

The earlier studies on programming difficulties have been carried out mainly on difficulties to learn specific programming language concepts [9][11][15] than general learning difficulties to program. The objective of this study is to understand the students in terms of learning difficulties and learning style in programming courses, independent of the programming languages. To achieve this, relevant data are collected from a sample of 137 undergraduate engineering students through suitable questionnaire. These data are subjected to analysis and various observations are made. The observations of the study reveal the role of cognitive science in programming education. The results can be used as a starting point to design an integrated and feasible teaching strategy along with proper learning material and tools.

4. Methodology

Data acquisition and analysis can be done in various ways based on the trade-off between required accuracy of results and the cost of resources. Questionnaire is an economic way to collect data from a relatively large group of respondents. The reliability of the data depends mainly on the design quality of the questionnaire and quality of its administration.

4.1. Questionnaire Design

A set of guidelines were followed to design the questionnaire. In order to make it valid, it should address the goals of the study precisely. The basic factors relevant to the programming education were identified and questions were framed based on these factors. The factors considered are listed below:

- The first programming language of each student
- Learners’ cognitive difficulties (in general) during programming

- Learners’ difficulties in program comprehension
- Assessment of teaching quality by learners
- Learner’s interest in programming
- Learning style/approach in programming course

The questions were framed straightforward, without any ambiguity. A question was framed to get an idea of various languages used as first programming language. The questions to list the general difficulties and comprehension difficulties were in open format, giving freedom to the students to list various difficulties they encounter, independent of the programming language. The open format increases the reliability of data, since there is no external trigger to answer in a particular way. Since teaching quality helps the students to learn the given content and is an important factor to decide the effectiveness of a course, question was framed to know the students’ view on it. A question was framed to assess the students’ interest in programming courses, since subject interest plays a vital role in learning outcome. It is well known that, if the teaching style is coherent with the learning style, then the effectiveness is more. Therefore a question was framed in closed format to know the learning resource usage preference of each student. This was given with six options: classroom, text book, lab session, discussion with friends, coaching centre and other resources.

Different questionnaires were prepared for effective and ineffective students. The focus of effective students’ questionnaire was to get data on their learning style in programming courses, the difficulties they had faced as novices and how they overcome them, while the focus for ineffective students was on their difficulties irrespective of the programming language, their interest to learn programming, their view on teaching quality and their expectations from a teacher. The questionnaires have been validated by few experts from the field.

4.2 Data Collection

The administration of the survey plays an important role in the quality and reliability of the collected data. Students at undergraduate level are the sample elements of this survey and typically students have the tendency to falsify their answers or leave the questions without answering. The main reasons for such behavior can be that they do not understand the relevance of the survey or sincere answers can be a threat to their privacy or do not understand the question. Hence to increase the reliability of the data, certain measures and decisions have been made as follows:

- Survey administrator should explain the objective of the survey and the importance of providing sincere answers
- Make questions on personal details as optional and do not compel the respondents to reveal their identity
- Put a promising statement which ensures that their responses will be kept confidential and will be used only for academic and research purpose
- Give reasonable time to answer the questions
- Ask the respondents to sit far apart so that they will not tend to copy the answer from a neighbor or discuss with each other
- Provide a comfortable environment. Neither the classroom teacher nor the survey administrator should overlook the answers during the survey administration
- Let the respondents participate willfully and sincerely without someone’s compulsion

Typically reliability of data depends on sample adequacy also. Therefore theoretical sampling has been used as the sampling method. It is a responsive approach which makes sampling open and flexible [6]. Based on the accessibility, four engineering colleges from south India were selected to collect data. The printed questionnaires were administered responsibly among undergraduate CS/IT students of these colleges. The respondents have been exposed to programming during their B. Tech. course either for one or two initial semesters. Responses were collected from a sample of 137 students which includes 46 effective and 91 ineffective students of different semesters. The categorization of students was done according to the concerned teachers' opinion, which was based on whether a student is able to write programs (effective) or not (ineffective), at academic level.

The various strategies mentioned above, and followed for questionnaire design, administration and sampling adequacy, ensure the quality and reliability of the collected data. These strategies address the scientific aspect of qualitative research also.

5. Analysis

The various data collected are presented duly in different graphical forms like pie chart, table and bar-chart and the observations made are explained in the following 6 subsections.

5.1 First programming language

The percentage of each language is presented as pie chart in Figure 1. Out of the 137 students, 74% used 'C' as the first programming language, 10% used 'C++' as the first PL and 9% used BASIC as the first PL. There were also very few students who had used other languages like Java, LOGO, FORTRAN etc.

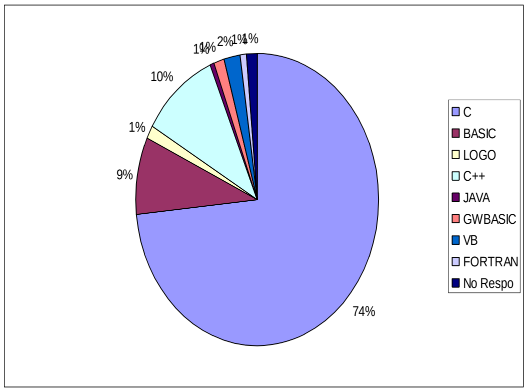


Figure1.Proportion of the first programming language

5.2 Programming difficulties

Survey respondents (both category) were asked to list at least five difficulties they faced during programming. Since the question was in open format, a wide variety of difficulties they encountered during programming were listed. These are classified as follows, mainly based on the programming process aspect:

i. Starting problems
ii. Language syntax related
iii. Logic related
iv. Debugging related
v. Lack of knowledge related to various aspects like OS, problem domain etc.

vi. Psychological problems
vii. Physiological problems

A complete list of the difficulties as mentioned by the students is given in appendix A.

Most of these difficulties are also classified using Bloom's Taxonomy (BT) [2] as the template. BT is a well-known taxonomy of educational objectives and explains 3 domains of learning – Cognitive, Affective and Psychomotor. Cognitive domain deals with knowledge and thinking, affective domain deals with attitudes and psychomotor domain deals with physical actions. It was primarily developed by Bloom for academic education and each domain has a well-defined hierarchy of levels. From the above list of seven problems, the first five problems can be mapped to Bloom's Taxonomy on cognitive domain, which is a well defined hierarchy of six cognitive levels -Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. The sixth and seventh problems can be mapped to affective domain and psychomotor domain respectively. Out of these three domains, cognitive domain problems seem to be predominant.

The respondents have cognitive problems at various levels. Some of them have problems with the very basic cognitive skill called memorization which is essential for knowledge acquisition. Some others have listed cognitive difficulties to comprehend a given problem and a given program, which needs comprehension and analysis skills. Yet another group finds it difficult to apply ('application' cognitive level) the concepts learned to solve problems. Majority finds difficulty to arrive a correct logic, to integrate modules to a working program and difficulty in algorithm design. These are synthesis problems. It is obvious that students also have difficulties in evaluation, which is the highest cognition level. The symptoms for these are their difficulties to justify/defend/describe a program logic. The classification of the problems to various cognitive levels is done based on cognitive-level-keyword mapping. BT in cognitive domain provides a set of keywords for each cognitive level.

Therefore it is observed that, programming students have serious difficulties in all the levels of Bloom's taxonomy in cognitive domain, independent of the programming language they have used. Some of the respondents claimed that the memorization problems may be because of the structure/features (syntax and semantics) of the programming language, where the morphology is different and complex than a natural language. Other higher level problems may be a manifestation of lack of various cognitive skills in students. A detailed analysis is needed to find out the real reasons for these difficulties and it is beyond the scope of this work. Table 1 shows the mapping of some of the difficulties to the various levels of BT in cognitive domain.

Table 1. Mapping of cognitive difficulties to various levels of Bloom's Taxonomy in cognitive domain

Bloom's Taxonomy Level	**Associated programming difficulties**
Knowledge: Recall data	• Difficulty to remember the syntax of a PL and making syntax errors • Lack of knowledge of useful library functions and header files • Lack of knowledge about system software • Forget to declare/initialize variables • Lack of problem-domain knowledge
Comprehension: Understand the meaning,	• Difficulty to understand the problem to

translation, interpolation, and interpretation of instructions and problems; state a problem in one's own words	solve • Unable to interpret the compiler generated error/warning messages • Difficulty in translating a logic to program • Difficulty in minimizing the number of steps • Difficulty to comprehend a given program
Application: Use a concept in a new situation or use an abstraction unprompted; apply what was learned in the classroom to novel situations in the workplace	• Unable to apply theoretical knowledge • Unable to solve a given problem • Difficulty in algorithm design • Difficulty in code optimization
Analysis: Separate material or concepts into component parts so that its organizational structure may be understood; distinguish between facts and inferences	• Difficulty to comprehend a program without comments • Difficulty to understand recursive functions • Difficulty to understand the logic of large/complex programs
Synthesis: Build a structure or pattern from diverse elements; put parts together to form a whole, with emphasis on creating a new meaning or structure	• Taking more time to find a correct logic • Difficulty in algorithm design • Difficulty to integrate different modules
Evaluation: Make judgments about the value of ideas or materials	• Unable to describe a program logic • Difficulty to justify and debug a program logic • Taking more time to debug a program

5.3 Comprehension difficulties

Comprehension of a given program is an important ability a programming student should acquire. It mainly requires two cognitive skills: comprehension and analysis. But students find it difficult to comprehend a given program for various reasons. It is observed that they have problems at various granularities, starting from understanding the purpose of a variable in a program to the logic of the program. A complete list of the comprehension difficulties as mentioned by the students is given in appendix B.

5.4 Teaching quality

This indicates the view and expectations of learners on teaching. It is observed that the majority of the ineffective respondents (81%) believe that they can improve their programming skills provided quality teaching of their expectation. 7% believe that they cannot improve even with quality teaching, 2% were uncertain on this and 10% did not respond to this question. The survey also sought the students' expectations from quality teaching. The points they have mentioned are summarized as follows:

- Explain basic concepts
- Teach the syntax and semantics of the programming language

- Provide examples
- Explain the logic of each program
- Provide more program assignments
    - Simple to complex order
    - Specify what output is needed from each program
- Give feedback on logic related errors
- Help to optimize the code
- Help to improve the thinking level of students
- Personal attention
- Assistance in difficulties

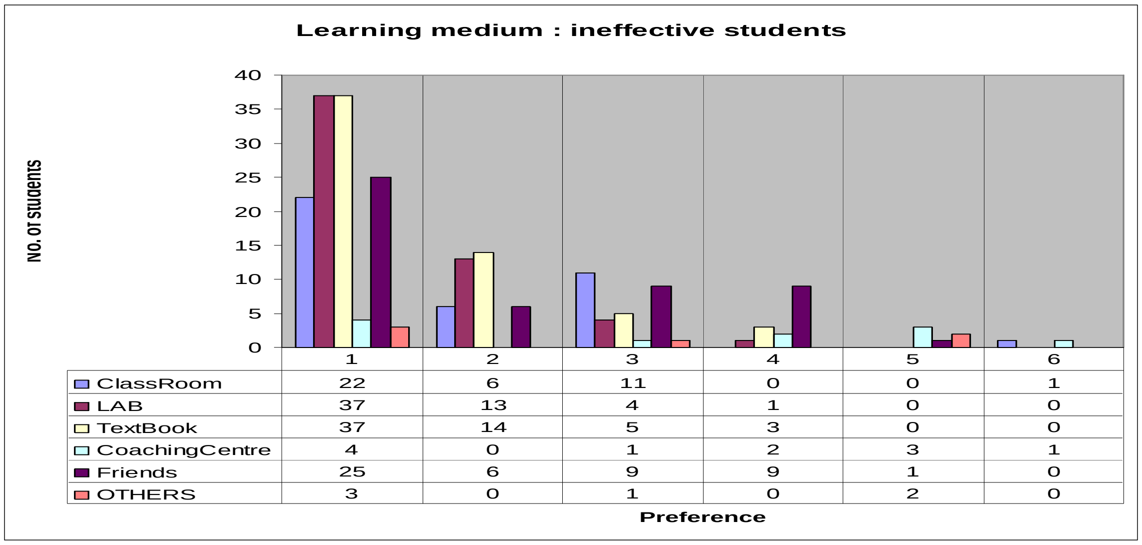
These demands can also be viewed as an indirect listing of their cognitive difficulties, which need to be addressed when a teaching strategy or learning material and tools are developed.

5.5 Students' interest to learn programming

Since interest of a student is the basic factor which influences learning, it was decided to know their interest in programming courses. This data was collected only from ineffective students to know whether this factor affects their learning effectiveness. Surprisingly, 92% of the respondents marked that they are interested to learn programming. The reasons for this interest may be different. 8% did not respond to this question. So one can observe that, lack of programming skills in ineffective students may be a manifestation of cognitive domain problems than affective domain problems. Conduction of experiments is needed to prove this and to improve the cognitive levels of ineffective students.

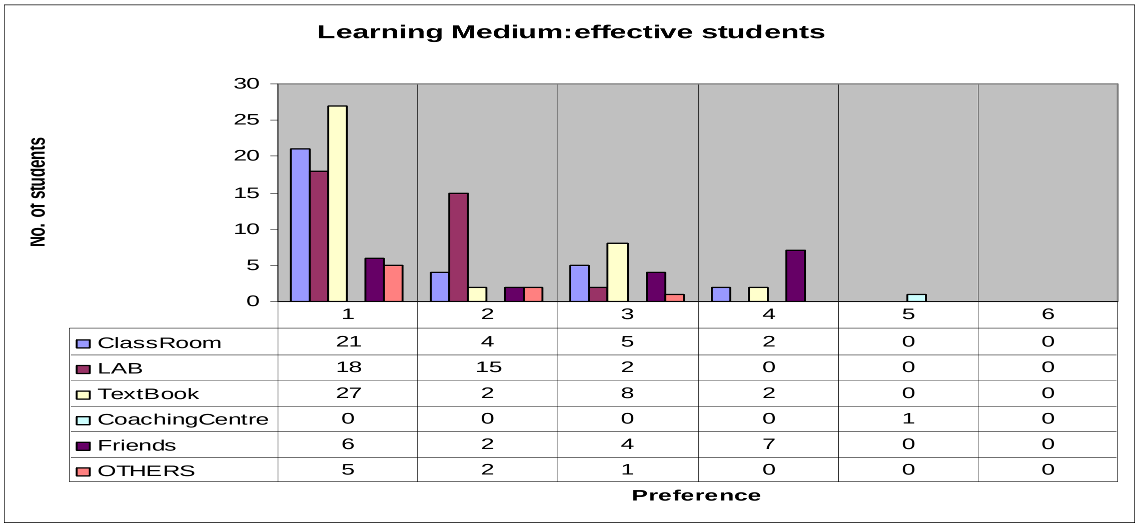
5.6 Learning style

The data on usage of learning resources, collected from effective and ineffective category are converted into bar chart by putting preference of learning resources on the x-axis and number of students on the y-axis. Out of the given six options of learning resources, both the category preferred more on classroom, lab and text books. The major difference observed is that the ineffective students prefer peer tutoring also along with the above specified three resources. This is represented in bar chart in Figure 2 and 3.



	1	2	3	4	5	6
ClassRoom	22	6	11	0	0	1
LAB	37	13	4	1	0	0
TextBook	37	14	5	3	0	0
CoachingCentre	4	0	1	2	3	1
Friends	25	6	9	9	1	0
OTHERS	3	0	1	0	2	0

Figure 2. Learning resource preference by ineffective students



	1	2	3	4	5	6
ClassRoom	21	4	5	2	0	0
LAB	18	15	2	0	0	0
TextBook	27	2	8	2	0	0
CoachingCentre	0	0	0	0	1	0
Friends	6	2	4	7	0	0
OTHERS	5	2	1	0	0	0

Figure 3. Learning resource preference by effective students

The effective students were also asked to specify their style of learning programming, for which they gave various answers. But a pattern could be found from these answers. They first acquire theoretical knowledge on programming from various resources like class room, text book etc. and then learn more by practicing in a computer. This shows that a programmer needs to acquire both declarative and procedural knowledge. This model works for effective students because they have enough cognitive skills to tailor to the programming needs. But the ineffective students have to first acquire enough cognitive skills to start with programming.

5.7 Discussion

The analysis of various data revealed that the cognitive domain of students plays a significant role in their effectiveness in programming process than any other attribute. They have cognitive difficulties at various levels, during programming and program comprehension, independent of the programming languages used. The programming students have serious difficulties in all the 6 levels of Bloom’s taxonomy in cognitive domain - Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. The list of expectations from quality teaching by ineffective students indirectly shows their cognitive difficulties and their necessity for more cognitive support. The response of ineffective students on their interest to learn programming shows that cognitive domain influences their effectiveness than affective domain. The preference for peer tutoring learning style by ineffective students indicates that they need cognitive support in learning. In summary, the difficulties listed by the respondents and other attributes are converging to cognitive domain. Therefore cognitive domain has high significance in programming education than affective or psychomotor domain or any other demographic attributes.

6. Conclusion and Future Work

In order to investigate on problems with programming education, this paper analyzed various data collected from a group of undergraduate engineering students on programming education. This study shows that students have cognitive difficulties at various levels, during programming and program comprehension, independent of the programming language they have used. It also shows that there is a distinct difference in the learning styles of effective and ineffective students. The latter category has a higher preference for peer tutoring along with classroom, lab and textbooks. Therefore, explicit focus and cognitive support is needed to enable the ineffective students to learn programming. The cost of not improving the effectiveness of computing education is an unacceptable failure rate in programming courses, which in turn will produce poor quality programmers from academic institutions. The impact of this on industry and society may be three dimensional – cost, quality and time. A breakthrough research is needed to make programming courses more effective, enjoyable and attractive.

The results of this study show that all the six levels of cognitive domain specified in Bloom's Taxonomy are highly significant in programming process. This high cognitive requirement for programming process can be the reason why many students experience it difficult to learn. The results presented in this study seem logical and applicable. They can provide some directions for the design of an instructional system, learning materials and tools, in order to make programming courses more effective. However, the issues like heterogeneity in students' cognitive behavior and scalability can increase the challenge to educators. A further study is under progress to corroborate the findings presented here and to design a comprehensive cognitive model for programming education.

Reference:

[1] J. Bennedsen, M.E Caspersen, "Exposing the Programming Process", Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, p 186-190, Feb 2005.

[2] B.S. Bloom et al., "Taxonomy of Educational Objectives – The classification of Educational Goals" Handbook I: Cognitive Domain, David McKay, New York 1956.

[3] M. Butler, M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", Proceedings of Australasian Society for Computers in Learning in Tertiary Education (ASCILITE) Singapore, 2007.

[4] W.Campbell, E.Bolker, "Teaching Programming by Immersion, Reading and Writing", 32nd ASEE/IEEE Frontiers in Education Conference, Nov. 2002.

[5] M.E. Caspersen, J. Bennedsen, "Instructional Design of a Programming Course – A Learning Theoretic Approach", Workshop on International computing Education Research, Sep 2007.

[6] J.M. Corbin, A.L. Strauss, "Basics of Qualitative Research", 3rd Ed., Sage Publications Ltd., U.K., 2008.

[7] V. Dagiene, J. Skupiene, "Learning by Competitions: Olympiads in Informatics as a Tool for Training High-Grade Skills in Programming", 2nd International Conference on I.T.: Research and Education, Jul 2004.

[8] Y. Gael, "A theory of Program Comprehension: Joining Vision Science and Program comprehension", GEODES, University of Montreal, Dec 2005.

[9] S. Garner, P. Haden, A. Robins, "My Program is correct But it Doesn't Run: A Preliminary Investigation of Novice Programmers' Problems", Australasian Computing Education conference, 2005.

[10] D. Gries, "What Have We Not Learned about Teaching Programming?", IEEE Computer, Vol. 39, 2006.

[11] E. Lahtinen et al. "A Study of the Difficulties of Novice Programmers", Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE) , Portugal, 2005.

[12] M.C. Linn, M.J. Clancy, "Can experts' explanations help students develop program design skills?", Int. Journal on Man-Machine Studies, Vol. 36, 1992.

[13] R.Lister, J. Leaney, "First Year Programming: Let All the Flowers Bloom", 5th Australasian Computing Education conference, 2003.

[14] A. McGettrick et al, "Grand challenges in Computing: Education – A Summary", The Computer Journal Vol. 48, 2005.

[15] I. Milne, G. Rowe, "Difficulties in Learning and Teaching Programming – Views of Students and Tutors", Journal of Education and Information Technologies 7:1, 55-66, 2002.

[16] A. Robins et al., “Learning and Teaching Programming: A Review and Discussion”, Computer Science Education Journal, Vol. 13, 2003.

[17] M. Shaw, “Software Engineering Education: A Roadmap”, Proceedings of 22nd International Conference on Software Engineering, 2000.

[18] B. Shneiderman, “Cognitive Psychology and Programming Language Design” ACM SIGPLAN, Jul 1975.

[19] G.L. White, M.P. Sivitanides, “A Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers’ Cognitive Characteristics”, Journal of Information Systems Education, Vol. 13(1), 2002.

[20] L.E. Winslow, “Programming Pedagogy - A Psychological Overview”, SIGCSE Bulletin, Vol. 28, Sep 1996.

[21] S. Xu, V. Rajlich, “Cognitive Process during Program Debugging”, Proceedings of the Third International Conference on Cognitive Informatics, 2004.

Appendix A
Programming difficulties

- Starting problem
  - Where to start
  - How to start
- Syntax related
  - Making Syntax mistakes
    - Confusion in usage of uppercase/lowercase in keywords/function names
    - Mistake in usage of operators
    - Mistake in passing parameters to function
    - Mistake in the usage of control structures
  - Difficulty to remember the syntax
    - Syntax varies from one language to another
    - Difficulty in the usage of array and pointers
  - Lack of knowledge of useful library functions and header files
- Logic related
  - To find the correct logic
  - Taking more time to find a correct logic
  - Difficulty in converting a logic to program
  - Difficulty in minimizing the number of steps
  - Difficulty in algorithm design
  - Code optimization and beautification
  - Integration of modules
- Debugging related
  - How to debug
  - Taking more time to debug
  - Unable to interpret the compiler warning messages
  - Exception handling
- Lack of knowledge about system software
  - Operating system
  - Program developing environment
    - Editor
    - Compiler
      - Difficulty to interpret compiler generated error messages
    - Other programming tools
- Other difficulties
  - Unable to apply theoretical knowledge
  - Forget to declare/initialize variables
  - Difficulty to understand the problem to solve
    - Communication language-related problem
    - Lack of problem-domain knowledge
  - Frustration
    - Due to lack of attention and help from teacher
    - When errors are more and unable to rectify them
  - Typing speed is low
  - Unable to sit too long in front of computer
  - Unreasonable time/Time constraint

Appendix B
Comprehension difficulties

- Due to lack of knowledge of programming language

- Difficulty to comprehend a program without comments
- Difficulties at various granularities
  - To understand the purpose of each variable
  - To understand the control structures/flow
  - To understand certain functions
    - Difficulty to understand recursive functions
  - To understand the logic of programs
  - To comprehend large/complex programs
- Other difficulties
  - Due to lack of practice
  - Due to lack of assistance