Determining The Barriers Faced By Novice Programmers

Pranay Kumar Sevella *pranay_kumar.sevella@students.tamuk.edu*
Dept. of Electrical Engineering and Computer Science
Texas A&M University–Kingsville
Kingsville, 78363, U.S.A

Young Lee *young.lee@tamuk.edu*
Dept. of Electrical Engineering and Computer Science
Texas A&M University–Kingsville
Kingsville, 78363, U.S.A

Jeong Yang *jeong.yang@tamuk.edu*
Dept. of Electrical Engineering and Computer Science
Texas A&M University–Kingsville
Kingsville, 78363, U.S.A

Abstract

Most of the novice programmers find glitches at various phases while trying to complete a program in their Computer Science programming course. These phases can be while constructing the code, finding errors in the code at the time of compilation of the program, debugging these errors while executing the program. Novice programmers are unable to understand some of the concepts in programming. Computer Science programming course instructors are experiencing difficulty in finding these barriers faced by the students. These barriers are forcing students to drop programming course from their degree plan and becoming a concern to the professors teaching programming course. In this research ActivePresenter software is used. This software recorded the full motion video with crystal clear quality and helped in capturing screen shots automatically with a click of a mouse or pressing any key on the keyboard of the students who are trying to complete a programming assignment. By analyzing all the recordings collected from different students, these barriers are determined.

Keywords: Novice Programmer, Programming Barrier, Programming Education.

1. INTRODUCTION

Computer Science education researchers have tried since the 1980s to find how the Computer Science professors can effectively serve their students. Computer Science students who are considered as novice programmers registering in introductory programming courses are facing difficulties in programming and dropping from the course after attending few lecture classes. Computer Science programming instructors are facing difficulty in recognizing these barriers.

Purpose

The main purpose of this research is to find the barriers faced by the novice programmers. These barriers are considered to be challenges to the computer science programming instructors.

Method

All the students are given a C programming assignment to complete in one hour in the computer lab. While students are trying to complete the given assignment, their computer screens are automatically recorded in video. These video files, which record student's activity on the assignments, are analyzed and the barriers are determined.

The method used for determining these barriers is categorized into three stages.

Stage 0: Before Experiment Conducted
- Identifying a problem
- Preparing Classified Barriers
- Preparing questions to be asked to students
- IRB approval
- Setup experiments
- Find and invite novice programmer students
- Setup video recording on the computer lab

Stage 1: At the time of experiment
- Give an instruction to the students get consent to attend this experiment
- Recording the screen of the students trying to complete programming assignment

Stage 2: After the experiment
- Analyzing the recordings of each student individually
- Finding the barriers
- Classifying each barrier

Result
By thoroughly analyzing the recorded data, all the barriers faced by the Novice programmers are determined. And even the concepts which the Novice programmers face difficulty in understanding are also determined.

1.1 About C Programming
C language is the most popular and widely used programming language. Dennis Ritchie developed C language between 1969 and 1973. Most of the other programming languages are derived directly or indirectly from C language. C is sometimes considered as “High-Level assembly language”.

1.2 Novice Programmer
Any person, who is a beginner in programming, is called a Novice Programmer. In this research, freshmen students who have knowledge about C programming which is one of the courses in their previous semester are considered to be novice programmers.

2. RELATED WORK AND SUMMARY OF REFERENCES
2.1 Background
Previously, several studies have been done to understand the behavior of a Novice programmer. One such study is [1], the authors estimate the final grade of the student based on how the students do their respective lab assignments and how they react to errors they face while programming [3]. Authors have recorded the data affective states and behavioral states of the students. Then, they applied the linear regression using each behavioral activity and affective activity. In this study, the teaching assistants collected the data by periodically observing the activities of each student.

The authors [5] have developed a syntax error preprocessing and integrated semantic system that helps novice programmers to decipher the compile-time error messages. By this system, novice programmers stress more on issues related to design rather than issues related to implementation. All the syntax errors that are checked are collected by a survey [6]–[7] of former and current teaching instructors. There are discrepancies observed between the errors identified by the instructors and the errors encountered by the students. In this paper, the authors have developed a real-time system with machine controlled error collection that records a hundred percent of java errors in a database. All the errors encountered by the students, faculty, and users using this IDE are collected in the database. Hence in this paper, it is concluded that there are

five errors .The instructors have identified that those errors are also collected by the automated system. The system also verified that there are discrepancies between error encountered by students and errors identified by instructors. And through the use of this system, most common errors faced by Novice programmers are also identified.

The authors [11] claim that there are many big gaps brought by present methods of programming education in understanding programming by novice programmers. Due to these big gaps, the novice programmers are losing confidence in continuing with the programming course. Hence to reduce these big gaps the authors have introduced an AtoP method. In this method, novice programmers are given a checklist along with the programming assignment. This checklist consists of small exercises, which help students to write source code for the assignment. Teaching assistants help students [12-15] to complete all the small exercises in the checklist along with the programming assignment. The authors have also developed a site called AtoP which records all the results from interactive assessments of Novice programmers. Novice programmers can retrieve their data at any time. With this AtoP method the big gaps for students are reduced.

2.2 Recording Programmers' Activity

For recording the activity of programmer in the computer, software for recording the screen of the student is required. Once software is started, it starts recording the screen of the students so that all the activities of students can be recorded. This data is saved in video format.

3. RESEARCH QUESTION

3.1 Problem

Many of the novice programmers take C programming as their Computer Science programming course. But some students are dropping this course after attending few lecture classes. Students who are continuing with the course are facing difficulties in the subject. Some students are easily able to complete their assignments on their own. On the other hand, some students are unable to complete given assignments on their own and instead, they take the help of their Instructor or Teaching Assistants to complete their assignments.

3.2 Goal

The goal is to find the specific phases or activities in the programming course where the novice programmers find difficulty in understanding and also implementing some C programming concepts in their program. Once the instructor finds the part or activity of the program where the novice programmer is facing problems, the instructor can guide the students to overcome this problem.

3.3 Hypothesis

The main hypothesis that is expected before the start of the research is that most of the barriers faced by the students are conceptual barriers rather than basic programming barriers.

Main concept oriented barriers can be as follows:

- Loops
  The barriers that can be faced in loops are understanding the concept of loops, understanding syntax, format, working, and control flow of program in *for*, *while* and *do-while* loops.

- Nested statements
  . The barriers that can be faced in nested statements are understanding the concept of nested statements *if*, *else-if* and *nested-if* statements. Understanding syntax, format, control flow of program in these statements. Novice programmers cannot understand when *if block* statements are executed and when the *else* block is executed.

- Switch concepts
  The barriers that Novice programmers can face in *switch* concept are the format of *switch* statement, the syntax, defining each case in switch statement, and understanding the control flow of program.

- Arrays concepts
  The barriers that Novice programmers can face in the concept of arrays are initialization of arrays, addressing any particular element in an array and conditions for applying basic arithmetic operations to arrays.

- Concept of functions
  The barriers faced by Novice programmers in the concept of functions are defining a function and if the function has arguments, passing arguments to that function and calling the function.

- Files concepts
  The barriers faced by Novice programmers in files concept can be initializing a variable to the file, accessing data present in the file, and reading or writing data from a file.

The basic programming barriers can be as follows:

- Header files declaration
  Declaring header files correctly can be a barrier to Novice programmers. This comes under basic programming barrier because in every programming code, header files have to be declared.

- Syntax errors
  Writing correct syntax is a necessity for all the concepts in C programming. Since every concept has particular syntax, writing correct syntax is a basic programming barrier for novice programmers.

- Basic programming format
  Writing basic format of programming can also be one of the barriers faced by Novice Programmer.

4. METHOD

This approach is implemented on Under-Graduate students of Electrical Engineering and Computer Science Department of Frank H. Dotterwhich College of Engineering at Texas A & M University - Kingsville who have taken Computer Science as their major studies.

In order to collect the screen recordings of students and analyze their recorded videos, an approval from Institutional Review Board (IRB) granted by the office of Research and Sponsored Programs at Texas A & M University – Kingsville is required because humans are considered as subjects in this research.

The students who participated in this approach were asked to sign and agree a consent form that was approved by IRB. Some of the important points from the consent form are as follows:

- Students do not get any benefits if they participate in this research
- Students can back out from participation at any moment
- Students agree to record their screen while they are completing their programming assignment
- Students agree to review their screen recording
- All the students participating in this experiment are above 18 years of age.

Twenty students came forward to participate in the experiment. Six different programming assignments were prepared and one randomly selected assignment among these six assignments was given to each student.

Six programming assignments cover different topics of C programming. The assignments were based on following C programming concepts:

- *if* and *nested-if* statements
- Iteration concepts (using *for* or *while* or *do-while*)
- *switch* statements
- files (manipulating data in files using C programming)
- functions (defining a function, calling that function and passing arguments to function)
- arrays (applying mathematical operations to arrays)

The categorization of each programming assignment distributed to each student in the respective programming concepts is explained in the Table 1.

Program	Programming Concepts
Program 1	*if* and *nested-if* statements
Program 2	Iteration concepts
Program 3	*switch* statements
Program 4	Files concepts
Program 5	Functions concepts
Program 6	Arrays concepts

TABLE 1: Categorization of Each Programming Assignment In The Respective Programming Concepts.

Students are given one hour to complete their programming assignment. Each student is assigned with one computer system in a one-to-one student to computer ratio in the Computer Laboratory.
The basic method of approach can be categorized into three different categories. They are as follows:

- Collecting data
- Method of collecting data
- Analyzing data

The control flow of these three categories can be represented as follows:

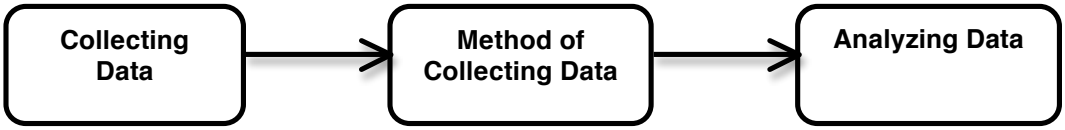

FIGURE 1: Control Flow of Method.

The detailed procedure of these three categories is explained below.

4. 1 Collecting Data
All the students from the Computer Science programming course who are interested in taking part in this research are given a C programming assignment. Students have to complete these programming assignments in the lab. Data is collected while the students are working on their

programming assignment. Data is collected individually from each novice programmer. The data collected is the screen recordings of the students.

4. 2 Method of Collecting Data

The data is collected by video recording and saved in video format. For Screen Recording no external hardware is required, as the software itself can record the screen of the novice programmers and the recorded data is saved in video format.

The software used to record the computer screen was ActivePresenter Free Edition Version 3.7.2 (Released 01.08.2013). It is a product of Atomi Systems, Inc. This software records the full motion video with crystal clear quality. This software also helps in capturing screenshots automatically with a click of mouse or by pressing any key on the keyboard. With this software, the collected data, which is in video format, can be exported to various other video formats like AVI, MP4, WMV, WebM.

Students use Dev C++ 4.9.9.2 version as their IDE (Integrated Developing Environment) for C programming. Dev C++ is used as IDE for both C and C++ Programming.

4.3 Analyzing Data

The assembled data is processed for analysis. The screen recording of each student is taken one at a time and properly analyzed.

Whenever any ambiguity is found, which is, when there is a long pause in student's programming activity, it means that student is facing some problem in writing further programming code. This pause can be starting from two minutes to any amount of time. This problem can be one of the barriers faced by the novice programmer.

Once the student completes programming code and tries to execute the program, compile time errors are found. When the student is unable to rectify these compile time errors, this can be considered as barrier faced by the novice programmer.

After the program is compiled, the student executes the program and gets unexpected output. This can also be considered as one of the barriers faced by the novice programmer.

Once all the barriers faced by the students are observed, then the programming code of the students is analyzed and the mistakes made by the student in their programming code are noted.

Later, how the student overcomes these barriers is observed. Students are allowed to utilize online resources by browsing the Internet and trying to correct their programming code.

This process is carried out for each individual student recording.

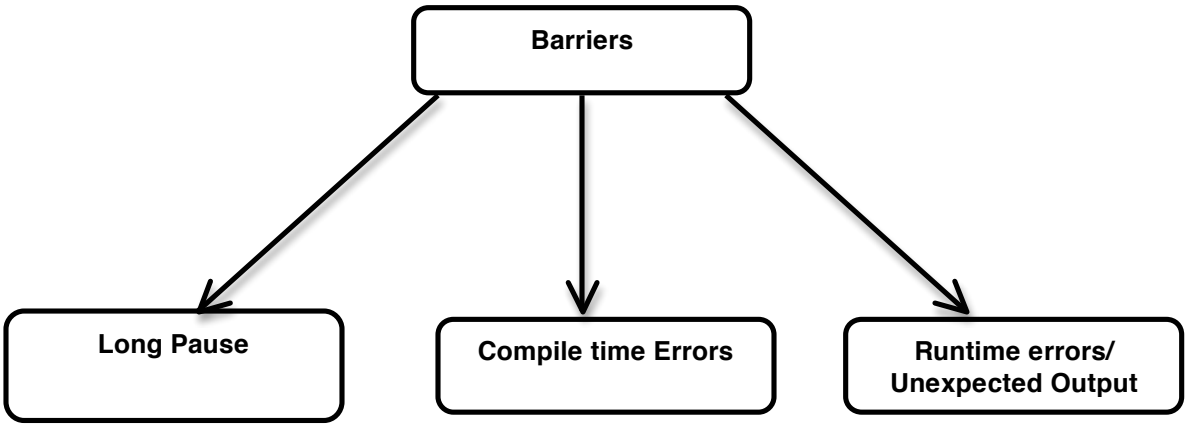

FIGURE 2: Classification of Barriers.

5. RESULTS

Out of twenty students who have participated in the experiment, screen recordings of only 16 students are collected. The rest of the four students recordings were not properly recorded and could not be used for the research purpose.

There are different problems that students faced while they were trying to complete their programming assignment. The problems faced by these novice programmers are mentioned in the following paragraphs.

Header Files

Header Files must be declared in all the programming code; hence the total number of students participated for this barrier are sixteen. And four students have made mistakes while declaring header files. Figure 3 indicates wrong declaration of header files.

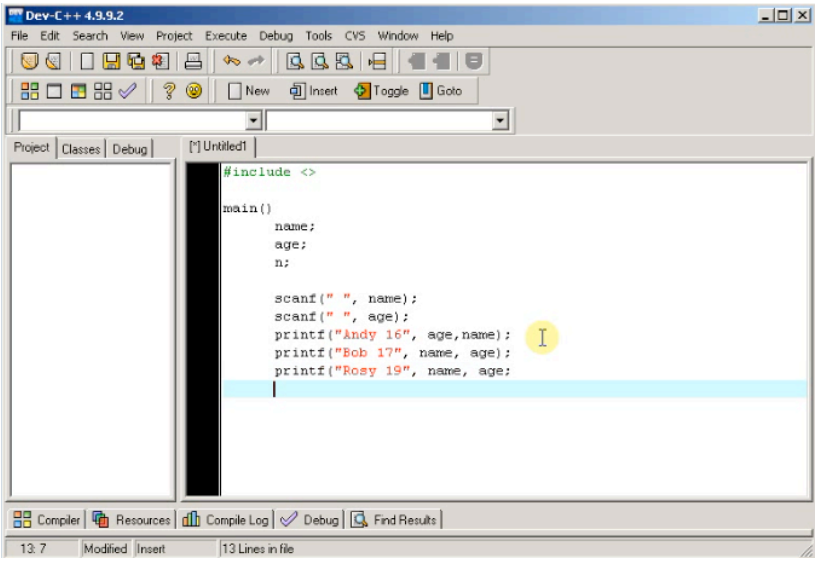

FIGURE 3: Screenshot 1.

Thinking for logic

Logic is required for writing any programming code; therefore, the total number of students participated in this barrier are sixteen. Four students have taken time to think logically for writing programming code.

Variables

Every programming code needs to have variables. These variables have to be declared with specific data types according to their usage in programming code; hence the total number of students participating in this barrier are sixteen. Three students declared wrong data types to the variables. One student did not assign data type to the variables. Another student is unable to use variables correctly in the program. Figure 3 indicates wrong declaration of variables.

printf* and *scanf

Every programming code that displays any output in the output screen has *printf* statements and every programming code that takes data from the user contains *scanf* statements. The total numbers of students participating in this barrier are sixteen. Two students were unable to write correct syntax for both *printf* and *scanf* statements. And four students were unable to write correct syntax for *scanf* statements. Out of four students, two students could not use *scanf* correctly. One student did not know how to take data given by the user and did not know the concept of *scanf*. Figure 3 indicates wrong declaration of *printf* and *scanf* .

Zero level

For this barrier, the knowledge of all the students is considered; hence the total number of students participating in this barrier are sixteen. Four students were unable to write basic programming code. Out of these four, two students tried to browse the Internet but were unable to understand the programming code found on the Internet.

Usage of Internet

All of the students were permitted to use the Internet for online resources to get basic syntax or any other programming related data from the Internet. Three students used online resources to get basic syntax and basic programming code. Out of these three, two students were able to complete their assignment successfully.

if* and nested *if

Three students got program1, which contained the concepts of *if* and *nested*-if statements. One student was unable to write conditional statements and could not use multiple *if* statements and put all *if* conditions in *printf* statements.

Iteration concept

Three students got program2, which contained the iteration concepts. All the students who got the programming assignment with iteration concept faced barriers. Two students faced difficulties in initializing the loop. One student got confused between the syntax of *for* loop and *while* loop and was unable to write looping statements. Another student completed his programming code but did not get the correct output and tried to manipulate code on variables in the code, but was unable to trace on which variable the operations were carried out. Figure 4 indicates wrong usage of while loop.

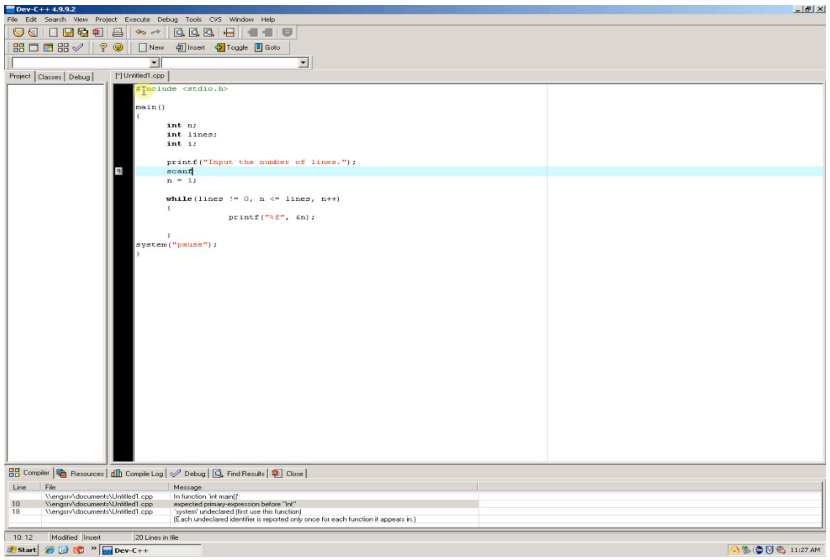

FIGURE 4: Screenshot 2.

***Switch* concept**

Three students got program3, which contained the *switch* concepts. All the three students were unable to define the *switch* statements and *switch* syntax and used Internet resources for format and syntax. Figure 5 indicates wrong declaration of *switch* statement.

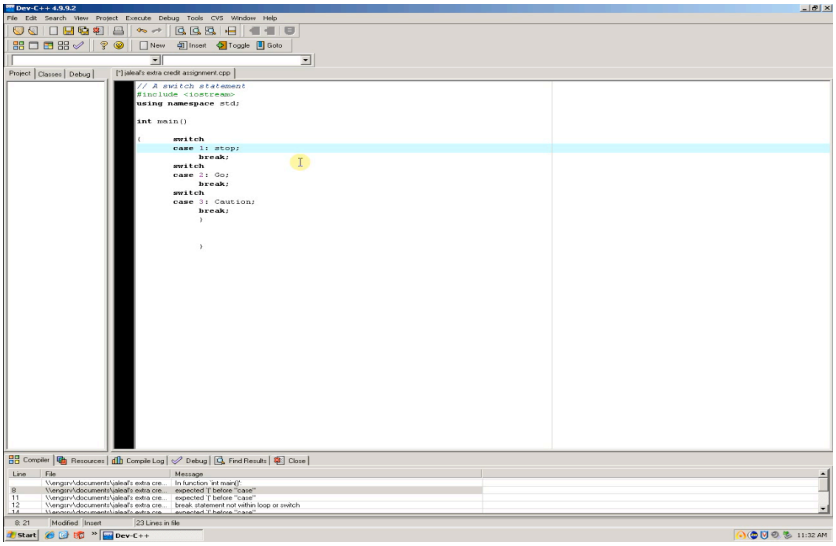

FIGURE 5: Screenshot 3.

Files concept

Two students got program4, which contained the concepts of files. There was only one student who faced difficulty in writing programming code and did not know the concept of files. Screen shot 4 indicates wrong usage of files concept.

Functions concept

Three students got program5, which contained the concepts of functions. Two students faced difficulties in function's concept. Out of these two, one student did not know the concept of functions and completed the program without using functions. The other student made mistakes while defining a function and was unable to call function with arguments.

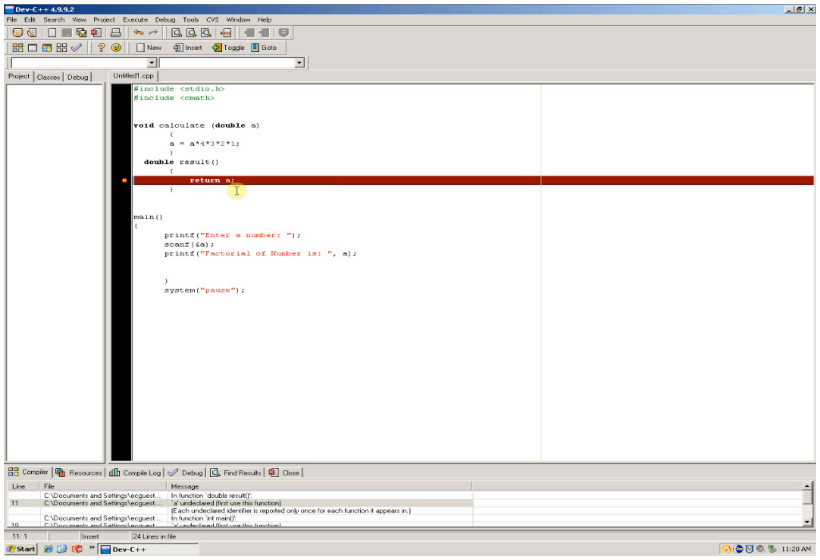

FIGURE 6: Screenshot 4.

Arrays concept

Two students got program6, which contained the concepts of arrays and both students were able to complete their programming assignment successfully . There were no barriers faced by the students in the arrays concept.

These were the barriers that were observed from the students programming recordings.

At the end of the experiment, it was observed that seven students out of sixteen students who participated in the experiment were able to complete their programming assignments successfully. Five students partially completed their assignment, and four students were in level zero and were unable to write the basic code.

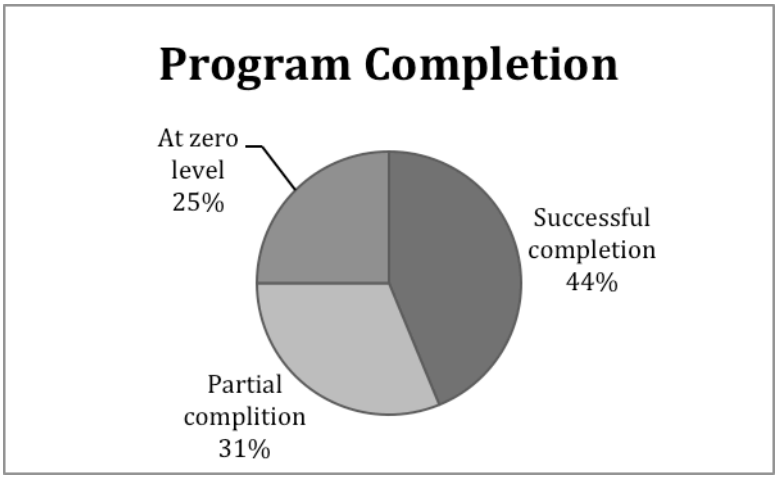

FIGURE 7: Pie Chart Representing Students to Program Completed.

There are 29 barriers that are observed in this research. Out of these 29, 19 barriers are basic programming based barriers and 10 are conceptual based barriers.

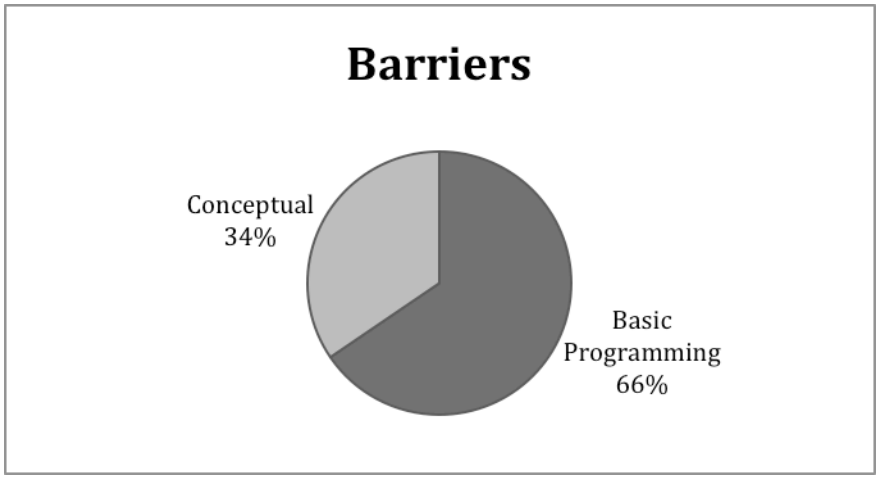

FIGURE 8: Pie chart of Barriers.

Table 2 represents the summarized list of all the barriers that are observed, the number of students who have faced these barriers, and the total number of students who were involved.

Barriers	Number of Students facing Barriers	Total Number of Students Participating
Header files	4	16
Thinking about logic	4	16
Variables	3	16
Issues with *printf* & *scanf*	8	16
Zero level	4	16
Usage of Internet	8	16
if and *nested-if*	1	3
Iteration concept	3	3
switch concept	3	3
Files concept	1	2
Functions concept	2	3

TABLE 2: Classification of All Barriers.

6. CONCLUSION

In this research, all the glitches that novice programmers face at different phases while completing a program in their Computer Science Programming course are determined. Also, the C programming concepts that novice programmers find difficulty in understanding and implementing in their C programming code are also determined.

After the successful completion of this research, it is found that the hypothesis predicted before the start of this is research is proved to be wrong. The predicted hypothesis at the start of this research is that most of the barriers faced by the novice programmers would be conceptual barriers than basic programming barriers. But after the research, it is proved that most of the barriers faced by novice programmers are basic programming barriers, rather than conceptual barriers.

The reason why most of the students drop from programming course can also be found out. This study even helps the programming instructors to teach C programming concepts in a more effective manner [16].

For future studies, the activities of the Novice Programmers while trying to complete the given assignment are also recorded with the help of an external video recorder. By this recording the facial gestures of the Novice Programmers is collected and, their behavioral states [1,18-21] can be determined. And with this recording, the other external resources that Novice Programmers use to overcome the barriers faced by them [17] are determined. These external resources can be referring to a Text Book or, referring to their classroom notes, or by taking help from their friend, or by taking help from Teaching Assistants.

7. REFERENCES

[1] Ma.Mercedes T. Rodrigo, Anna Christine M. Amarra, Sheryl Ann L.Lim, Ryan S. Baker, Thomas Dy, Sheila A. M. S. Pascua, Emily S. Tabanao, Matthew C. Jadud, Maria Beatriz V. Espejo-Lahoz, Jessica O. Sugay. *Affective and Behavioral Predictors of Novice Programmer Achievement*. In ITiCSE'09, July 6-9, 2009, Paris, France.

[2] Matthew A. Turk and Alex P. Pentland, *Face Recognition Using Eigenfaces.* In Computer Research and Development (ICCRD), 2011 3rd International Conference.

[3] Joni, S., Soloway, E., Goldman, R, and Ehrlich, K. 1983. *Just so stories: how the program got that bug*. SIGCUE Outlook 17,4(Sep. 1983), 13-26.

[4] Baker, R.S., Corbett, A.T., Koedinger, K.R., and Wagner, !A.Z. (2004) *Off-task behavior in the Cognitive Tutor classroom: When students "Game The System"*. ACM CHI 2004: Computer-Human Interaction, 383-390.

[5] James Jackson, Michael Cobb, Curtis Carver. 2004. *Identifying Top Java errors for Novice programmers*. 35th ASEE/IEEE Frontiers in Education Conference, Indianapolis, IN.

[6] Flowers, Thomas, Curtis Carver, and James Jackson. *Empowering Novice Programmers with Gauntlet*. Frontiers in Education, 2004.

[7] Hristova, Maria, Ananya Misra, Megan Rutter, and Rebecca Mercuri, *Identifying and Correcting Java Programming Errors for Introductory Computer Science Students*. ACM SIGCSE 2003. pp 19-23.

[8] Bruckman, Amy and Elizabeth Edwards. *Should we leverage natural- language knowledge? An Analysis of user errors in a natural-language- style programming language*. ACM SIGCHI 1999. pp 207-214.

[9] Chabert, Joan and T. F. Higginbotham, *An Investigation of Novice Programmer Errors in IBM 370 (OS) Assembly Language*. ACM 14th Annual Southeast regional Conference 1976. pp 319-323.

[10] Spohrer, James and Elliot Soloway, *Novice Mistakes: Are the folk Wisdoms correct?* Communications of the ACM 1986, pp 624-632.

[11] Dinh Dong Phuong, Yusuke Yokota, Fumiko Harada, Hiromitsu Shimakawa, (2010). *Graining and Filling Understanding Gaps for Novice Programmers*. 2010 International Conference on Education and Management Technology(ICEMT 2010)

[12] Paul Gross and Kris Powers, *Evaluating assessments of novice programming environments,* Proceedings of the first international workshop on Computing education research, USA , pages: 99 - 110 , October 2005.

[13] Charlie Daly, John Waldron, *Assessing the assessment of programming ability*, SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education, 2004, pages 210-213.

[14] Masoud Naghedolfeizi, Singli Garcia, Nabil Yousif, and Ramana M. Gosukonda, *Assessing long-term student performance in programming subjects*, December 2008, Journal of Computing Sciences in Colleges , Volume 24 Issue 2, page 241-247.

[15] Nghi Truong, Paul Roe and Peter Bancroft, Automated Feedback for “Fill in the Gap” Programming Exercises, January 2005, ACE '05: Proceedings of the 7th Australasian conference on Computing education - Volume 42 , pages 117-126.

[16] Douglas A. Kranch, Teaching the novice programmer: A study of instructional sequences and perception, May 2011, Springer Science+Business Media, LLC 2011

[17] Brad Myers and Andrew Ko, Studying Development and Debugging To Help Create a Better Programming Environment, 2003, CHI 2003 Workshop on Perspectives in End User Development.

[18] Matthew C. Jadud, An Exploration of Novice Compilation Behaviour in BlueJ, October 2006,

A thesis submitted to the University of Kent at Canterbury.

[19] Albert Lai and Gail C. Murphy, Behavioural Concern Modelling for Software Change Tasks, 2003, IEEE.

[20] Yuska P. C. Aguiar, Maria F. Q. Vieira, Edith Galy, ean-Marc Mercantini and Charles Santoni, Refining a User Behaviour Model Based on the Observation of Emotional States, 2011, COGNITIVE 2011 : The Third International Conference on Advanced Cognitive Technologies and Applications.

[21] Ben Shneiderman, Exploratory Experiments in Programmer Behavior, June 1975, Technical report No. 17 Submitted to Indiana University Bloomington.