Influence of Analytical Approach in Logic Building for Freshman Computer Science Students: A Study

Meenu Khurana[1], Archana Mantri[2], Viney Khurana[3]

[1,2]Chitkara University Institute of Engineering and Technology
Chitkara University, Punjab, India
[3]CEO, Paint The Slate SoftSkills, Chandigarh, India
[1]meenu.khurana@chitkara.edu.in
[2]archana.mantri@chitkara.edu.in
[3]vk.softskills@gmail.com

Abstract: Evidence of a knee-jerk reaction of despair among freshman computer engineering students is not uncommon. It is not the programming environment like the coding language itself that weighs down on them, but rather the shift in their thoughts that they need to bring in for solving a problem. Some people are born problem-solvers, some are not. But most of us lie in between, which means that most of us tend to solve issues and cross hurdles, sometimes in one go and sometimes by hit and trial. The only way to do this is to consciously think logically. But is it as easy as turning a key and opening the lock? How do you open the complex box of your thinking abilities? Engineering graduates must think from a different angle to the solutions to make themselves ready for the tough competitions. These freshman students therefore find themselves in a spot when it comes to forming and then implementing logic in their coding. A study has been conducted to see if this gap between school curriculum and learning programming subject in their first year can be filled in a conscious phased manner to improve their logic building. Various logic building techniques were put to practical use to study if the outcomes can be affected and the programming skills of students enhanced. In this paper, we see the methodology used to bring a group of students to a level where they are comfortable with logic building and are no longer wary of the solution finding.

Keywords: Logical Thinking, Analytical Approach, Reasoning Skills, Problem Solving, Logical Exercises, Programming Fear.

1. INTRODUCTION

Whereas the competition is increasing multiple-fold due to supply-demand imbalance and the equilibrium tilting more towards automation, studies in the recent past are showing heavy degradation in the quality of engineering graduates being produced. The struggle of computer science engineering students with programming continues to be a major factor for their failure in classrooms as well as the industry later on. The need for an innovative methodology for teaching as depicted in the quantitative analysis and research (Kannan et al., 2018) is very much agreed upon.

The teaching methodology in their case, as observed, is mainly statement oriented which means a programming language is merely considered as any other conventional language and is seen as a set of statements or elements to be taught in a specific order.
For example, in teaching C programming language, the typical curriculum goes like this in almost all universities:

Introduction to C
The C Character Set
Writing First Program of C “Hello World Program”
Identifiers, Keywords
Datatypes, Constants,...and so on..

Along the way, coding rules and conventions are tutored and then the executable programming code with perfect syntax is prayed to appear as if by some magic. The tragedy in all this is that everybody is putting their best efforts and yet nobody is wiser by the end of the day.

Our quest to find out the villain led us to get into Root Cause Analysis (RCA) as our scientific approach. This approach was consistent with the teaching reforms as advocated in another research (Gao, 2011) where they formed the bridge between training and learning abilities. Through countless prodding, umpteen hours and days of talking to students, teachers, guides, mentors, counsellors and management and deliberating the curriculum exhaustively, we were hell bent on putting our finger on the one problem area we could address. We had decided to strip down the problems through which we could reach to new conclusions via querying attitude. Our motivation was more intrinsic in nature. The love of the work itself was giving us the meaning and purpose to our research.

And soon enough we were rewarded by our quest. We found out our villain responsible for the decay of the quality of coding in programming students. To our surprise

the protagonist was the antagonist. What an irony. The hero of a programming language, Logic, turns out to be the villain, if only by way of absence. All the while, students were not being able to comprehend coding because their hero was missing. Without logic, coding is just a set of instructions which are written in a certain manner with some rules and then that is it. Similar findings can be found in other research papers as well (Lokare, et al., 2018). Then the praying starts in a desperate hope that the program solution will present itself. Due to this many students really hated programming courses.

Now that we knew our disease, we needed to develop a cure via the tools available like Instructional Strategies, Assessment Techniques, Collaborative Technique, etc as highlighted by a research (Snyder, Synder, 2008). In the process of observation and analysing the problem at hand, we hit a bigger wall. It is quite intricate to make someone 'un-learn' something and then make them learn it with a different approach. Therefore, if at all we are to build their base on logics, we need to catch them at the beginning that is their first year. The second year students would be much more difficult to turn to into logical thinking for efficient coding. Third year or Final year students would be near to an impossibility to change their learning patterns (Dharwadkar, Shingan, 2017).

Broadly we had one Critical Approach to reach our destination of resolving our problem and then we had the Analytical Approach. We needed to find the correct approach to it. If at all our experience in teaching has taught us, it is the fact that the Critical Approach to solving an issue is very effective in a professional or say, in a job environment. But when it comes to `classroom learning, it does not give as promising results as we would like to believe. So the research question now became if the Analytical Approach towards solving a problem can enhance the much needed logical skills in these lost students. We were in for a surprise.

2. METHODOLOGY

Begin with broader question and keep narrowing down till we find the basic building block of the problem which is the root cause. And then start building a solution in the reverse process, going from smaller solutions to larger ones. This in itself is the Analytical approach. After some deliberations, we thought of doing an in-depth study of both the freshman batches and second year students of computer science engineering. That is where the various Analytical tools came in handy.
So we set upon taking on our problem phase wise –

A. Problem-definition phase

Cognitive abilities play an important role in shaping of a successful programmer. High algorithmic and logical reasoning skills are likely to be the most important skills required in learning programming (Jin, 2010). Ideally, with these skills, students will be able to analyse the given problems logically and later provide the best solutions. Previous studies have also revealed that students who are lacking the required cognitive skills will fail to grasp the basic concepts of programming and will eventually becoming less interested and demotivated as referred to in the paper (Sujatha, et al., 2012). This meant that we needed to identify and possibly grade students' capabilities in providing solutions through problem-solving strategies and techniques. This would give us a fair assessment of students' logical thinking and reasoning skill levels. And so we put 400 computer science engineering students (200 from First Year engineering and 200 from Second Year engineering) to test with a special common question paper designed to assess these skills in them. The results were accumulated and groupings were done according to the level of their logical thinking abilities. The students' with marks more than 80% were left out of our research parameters for obvious reason. Rest of them were grouped as shown in Table1.

Table1. GROUPING OF STUDENTS

Test Marks	Year	Grade	Group	No. of Students
0 to 20%	First Year	A	I	26
	Second Year		II	20
21% to 40%	First Year	B	III	47
	Second Year		IV	52
41% to 60%	First Year	C	V	102
	Second Year		VI	89
61% to 80%	First Year	D	VII	16
	Second Year		VIII	22

Table2 depicts the division of these groups in accordance with their logical reasoning skills. Group numbers I and II were identified to be NLR (Nil Logical Reasoning), Group numbers III and IV to be LLR (Low Logical Reasoning), Group V and VI to be ILR (Intermediate Logical Reasoning) and Group numbers VII and VIII to be MLR (Medium Logical Reasoning). The HLR (High Logical Reasoning) Group has not been included in this study.

Table2. INTERPRETATION OF RESULTS

Logical Standing	Grade	No. of Students
NLR (Nil Logical Reasoning)	A	46
LLR (Low Logical Reasoning)	B	99
ILR (Intermediate Logical Reasoning)	C	191
MLR (Medium Logical Reasoning)	D	38

The average marks obtained in this initial test are also recorded in Table3. It was now qualitatively and quantitatively possible to know the extent of problems with these tested students.

Table3. INITIAL TEST MARKS OBTAINED RECORD*

Grade	Year	Average Marks**	Avg Grade Marks**
A	First Year	12	14
	Second Year	16	
B	First Year	22	29
	Second Year	35	
C	First Year	49	54
	Second Year	58	
D	First Year	72	76
	Second Year	79	

*All marks taken into account are rounded off to nearest digit
**Maximum Marks=100

B. Procedures-design phase

Now that groupings were done comprehensively and their logical and reasoning skill levels were transparent, we could focus on designing programs for them. Naturally, since these programs had to deal with different levels, they had to be formulated with relevant and conscious differences in their course curriculums. In order to keep the strength of the classes to be effective, sections were made with students not more than 50 in each section. Table4 shows the difference of the program and Table5 elaborates the methodology of the course curriculum. Here, it may be noted that these sessions were designed in such a way so as not to interfere with the regular academic schedules of the students.

Table4. COURSE DIFFERENCES

Groups	No. of Students	No. of Sections	Duration
Grade A (NLR)	46	01	64 hours
Grade B (LLR)	99	02	48 hours
Grade C (ILR)	191	04	24 hours
Grade D (MLR)	38	01	12 hours

Table5. COURSE DESIGNS

Total Duration	Aptitude Training	Logic Building	Logic Application (practise with coding)
64 hours (16 weeks)	4 hours	24 hours	36 hours
48 hours (12 weeks)	4 hours	20 hours	24 hours
24 hours (6 weeks)	4 hours	10 hours	10 hours
12 hours (3 weeks)	4 hours	5 hours	3 hours

In theory and understanding, the course curriculum was kept the same in all sections and in all groups.
The difference in duration was due to the practising part. For example, both the sections of Grade B and the four sections of Grade C were taught the same topic 'Seven Steps To Improve Analytical Thinking Skills' under 'Logic Building' of Table4, which in itself was 2 hours. But Grade B was given 4 hours of practising for this topic with examples and Grade C students were given only 2 hours to practise. Grade A students were made to practise the same for 6 hours.

The key topics in the course covered were:

- Understanding the Necessity of a Logical Approach
- Different Approaches to Logic Building
- Analytical Approach for Logic Building
- Reasoning and Aptitude Building
- Essentials of Analytical Skills
- Streamlining of Logical Thoughts
- Seven Steps to Improve Analytical Thinking Skills
- Logical Problem-Solving Techniques
- Identification and Removal of Logical Fallacies
- Analytical Approach Adaption in Coding

C. Observation phase

Carefully designed tests were conducted every week to ascertain the progress of all batches. But it was made sure that the results of these tests do not influence the ongoing course, otherwise a concrete conclusion would have been difficult to arrive at, considering on-the-fly improvisations. What needs to be remembered is that in this research, we are not assessing the students on their coding skills like programming or their syntax or how efficient they are in writing a program or running it on compiler. We are here to see how efficient they can get if their logical reasoning and analytical skills can be improved. How can we make them better in their programming skills through the analytical approach of logic building. And does it really help them?

The test results were duly recorded, examined with each aspect and interpreted.
Figure1 and Figure2 shows the average marks results of the week-wise tests conducted for First Year and Second Year students respectively for all categories of NLR, LLR, ILR and MLR.

3. RESULTS AND DISCUSSION

As it can be seen from Figure1, the growth in initial one to two weeks is not much in all Grades. This is primarily due to induction and grasping the new concepts of identifying reasoning inabilities in own self and talking a step towards logical thinking. This is applicable significantly more in cases of NLR and LLR.

Till about 70~75% classes or sessions, the growth is good in all cases which means till about 12 weeks in NLR, 9 weeks in LLR, 4 weeks in ILR and 2 weeks in MLR. After that it is not considerable, which means the effectiveness of the program lessens in the last stretch of 25~30% sessions. It shows that once Logic Reasoning has begun with the student and he/she is able to sustain it with his/her coding skills, he/she is ready to take on any programming course further on. This also means around 75% of our program is sufficient to kick start a students' Analytical Skills for Logic Building and beyond that the regular learning courses will take on the momentum.

The duration of the courses we designed for the research were very well balanced as they proved to bring all the Grades of students at almost same level. It would be appropriate to say here that the students of NLR, who were disheartened and frustrated because of the problems with building logic for their programming, were now confident of their problem-solving skills.

From the Figure2 which shows the data of Second Year students' week-wise test marks, it is pretty clear that the same pattern is emerging as in case of First Year students. Similar dull growth is observed in the starting weeks of the program and similar slump is there in the last 25~30% of the program.

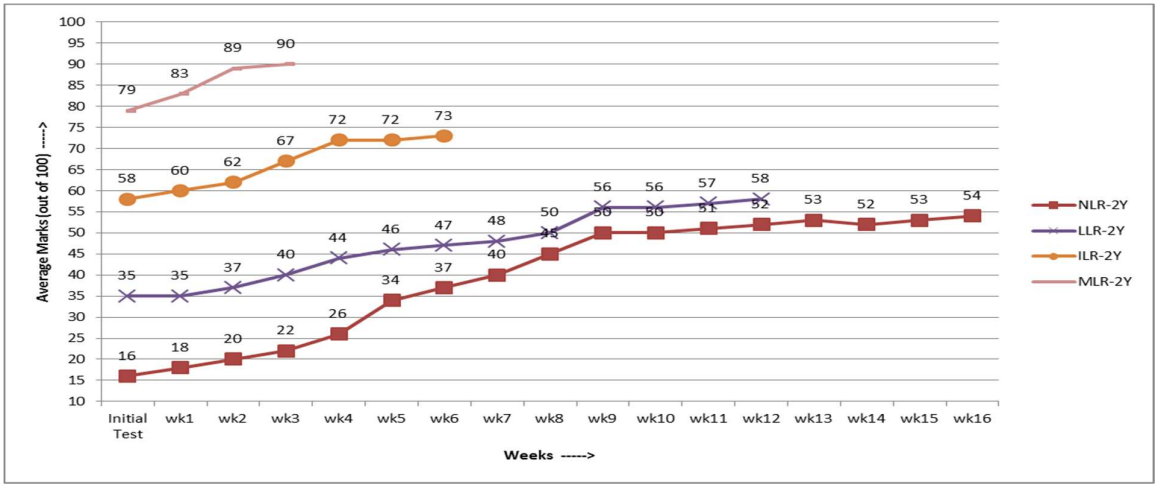


Figure1: Test Results of First Year Students (All Batches)

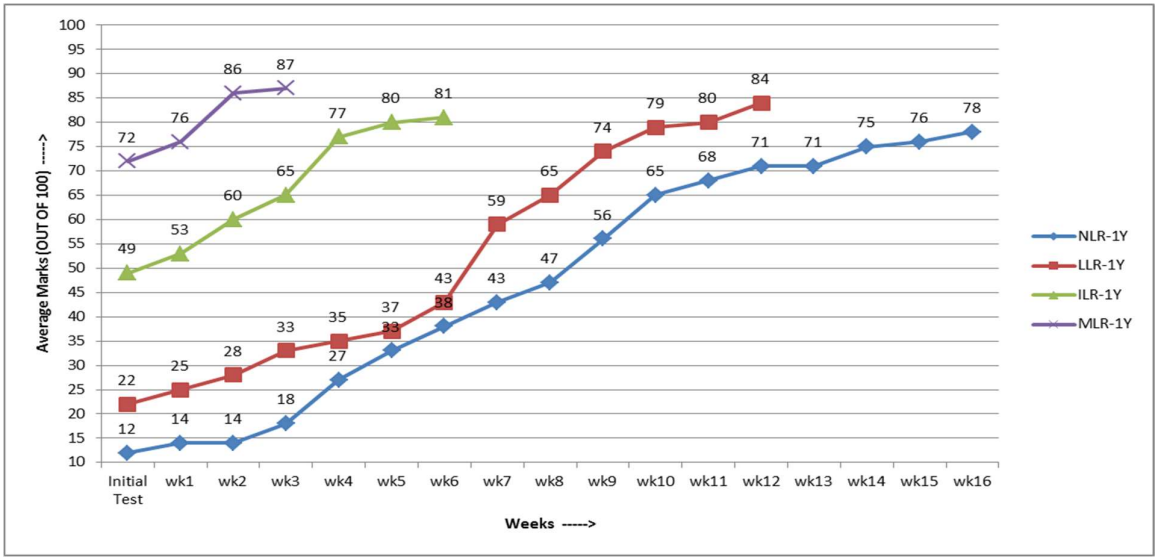
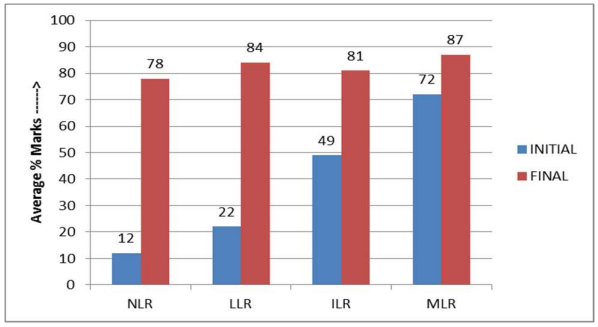


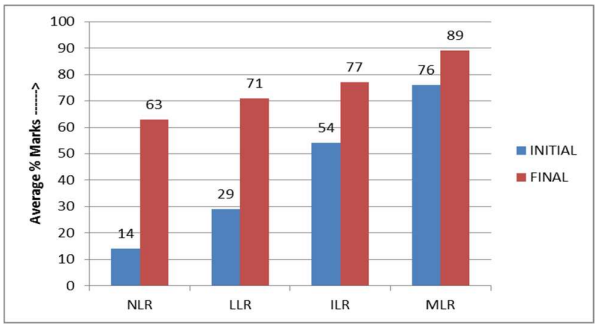
Figure2: Test Results of Second Year Students (All Batches)

The Graph1 shows tremendous growth of 558% in case of NLR, 277% improvement in LLR students, 65% growth in ILR and 22% in MLR students. The lower growth in MLR can only be partially attributed to the duration of their program as their initial rating itself was better in comparison to that of NLR. That being said, it is also clear at the same time that the growth is directly proportional to the duration of practise of Logical exercises.

From the Graph2, it can be seen that the growth in NLR is the maximum (350%), LLR registering145%, ILR with 43% and MLR with 17% improvement. Although the growth is there but still we do not see them coming up either with good logical skills or at least at par with others in the same year.



Graph1: Net Growth Data for First Year Students



Graph2: Net Growth Data for Second Year Students

From here we can see that First Year students' response in learning the Logical Skills was way much better than that of the Second Year ones. The reason behind this is simple – the notion these students had already formed about how to do the programming with the general language learning skill we talked about earlier, would not let them break the glass ceiling and go for the basic logic of the problem given. It should not be forgotten that these students have been learning coding from a different perspective and not analytically.

4. CAUSE AND EFFECT

The main strength of this kind of experimental research is the natural articulation between the two variables – technique and its outcome, determined by a cause and effect relationship. By causing systematic manipulation of teaching techniques, the effects of these approaches were recorded to be leading to logical results. Table6 depicts the stages of the different approaches and their concomitant. The period of implementation of these techniques was different for all groups depending on their program length, however keeping the percentage as same for all.

Table6. TECHNIQUES AND THEIR OUTCOMES

Approach / Technique	% of total hours of program	Effect on Students
Establish the prerequisite through group discussions of real life examples and a case study	Initial 5%	Yearning to learn the logical way generated
Reasoning and aptitude building through exercises and practise problems	Next 20%	Started to think in problem solving mode
Streamlining logical thoughts via an analytical approach through brain games like Sudoku	Next 20%	Understood the importance of steps in a logical solution
Step-wise analytical thinking skill development through practice problems and importance of flowcharts	Next 25%	Improvement in logical thinking
Implementing the above technique in coding of examples/ problems in C language	Next 30%	Improvement of logic building in coding

CONCLUSIONS

The objective of this research, which was to ascertain that the Analytical Approach to Logic Building can really help computer science engineering students overcome their fear of programming and thus make coding easier for them, was successfully achieved.

The study also glaringly highlights that since it is very difficult to 'un-learn' the half-learnt things and then 're-learn' it with different perspective, freshman computer science students should be made to learn building their logical skills right from the beginning, lest it is too late.

This study also shows that students be tested before they enrol in such guiding program so that appropriate duration of courses can be identified with.

ACKNOWLEDGEMENT

With gratitude, we thank the management of Chitkara University in encouraging us to conduct this research.

REFERENCES

Kannan, S. Sumathi, D. and Prabakaran, T. (2018) A Study on Challenges and Opportunities in Teaching Programming Subject to first Year Computer Science and Engineering Students:In the Perspective of Faculty and Student, Journal of Engineering Education Transformations, 31(3), 74-78.

Gao, R. (2011) doi: 10.1109/ICCSE.2011.6028863

Lokare, V. T. Jadhav, P. M. and Patil, S. S. (2018) An Integrated Approach for Teaching Object Oriented Programming (C++) Course, Journal of Engineering Education Transformations, 31(3), 17-23.

Snyder, L. G. and Synder, M. J. (2008) Teaching Critical Thinking and Problem Solving Skills, The Delta Pi Epsilon Journal 1(2).

Dharwadkar, N. V. and Shingan, G. G. (2017) Student Quality Circle: Skillful Learning Environment, Journal of Engineering Education Transformations, 31(2), 57-67.

Jin, Wang (2010) doi: 10.1109/ICETC.2010.5529249.

Sujatha, C. Jayalaxmi, G. N. and Suvarna, G. K. (2012) doi: 10.1109/AICERA.2012.6306734.