Journal of Engineering Education Transformations ,
Volume 31 , No. 3, January 2018, ISSN 2349-2473, eISSN 2394-1707

Project Based Learning of Programming Subject: Case study on Data Structures

Shridhar T Doddamani

Department of Automation and Robotics, BVBCET, Hubli
Shridhar_d@bvb.edu

Abstract: With the advisement of information technology, students are inclined to learn computer programming based subjects. This would help students to incorporate the advancements in the information technology to the applications of their field of engineering. But teaching computer science subjects to the students pursuing an non-IT degree is challenging task. Since the students are aiming to join industries with different domains, teaching the subjects according to their requirements or that suits their domain requirement will increase the interest of students and will motivate them to learn the subject. In laboratory we divided the experimental programs in different categories like demonstration, exercise, structured query, open ended and course project. The programming concepts were taught using visual effects that helped them to understand logic, data flow of program and find the faults during execution time. The effectiveness of learning programming language and developing programming skills was significantly improved with the investigated approach.

Keywords: Data Structures, non-software, categorizations of experiments

Shridhar T Doddamani
Department of Automation and Robotics,
BVBCET, Hubli
Shridhar_d@bvb.edu

1. Introduction

In growing competitive world, Computer programming is considered a basic and important skill for any student pursuing a career in information technology. While computer science (CS) and information Science (IS) curricula provide rigorous training in computer programming, other IT-related majors such as Automation, Electronics, Electricals, IT, Mech. Etc, can place at best a secondary or least importance to programming. Teaching programming courses in such majors leads to several challenges.

The objective of the course is to expose students with computational thinking and fill the gap between computer science fundamentals and programming skills. These objectives present several challenges. To identify the possible challenges that can arise in the course, several surveys are carries out on students and noticed that student's interest in learning the subject is not high because of the following aspects:

1. The incorrect perception that computer science is the only IT degree where knowledge of programming is needed.

2. The course content is highly logical and abstract; students have some difficulties at the beginning of learning, so that they feel the course is difficult.

3. The course requires relatively high both in theories and in practices, even though theory is

easy to understand, but one may encounter other difficulties in practice. Some students reflected that they understood in the class, but fail to write programs.

4. For non-computer-science students, their instinctive feeling is the course is not their major's courses, they will not think highly of it.

5. They do not know the relation between this course and their major.

In our practice, we decided to address above challenges by offering a unified approach to teach computer programming and logically thinking to solve computer problems for all students in their programming course. Keeping the objectives and challenges in mind, we explored to answer the question "is there a unified approach to teach non-major and to what extent and in what aspects they could be taught same as CS major to solve real life problems that are like to encounter".

There are many practices in teaching computer programming to non-major students [1, 2, and 3]. Most of them are embedded in at least one of the computer science courses, and are combined with programming techniques for CS students. Also there are some courses which focus only on computational thinking itself. For example, in many universities the programming concepts are taught by using python, simulation and visualization [4]. They focus on how to describe and solve problem using a computer, and how to write algorithm, manipulate information and design programs using Python.

On the other hand, High-level application development for non-computer science major using image processing[5] Algorithm development and visualization environment for novice learners[6], visual computing in the form of computer graphics[7] has been used in programming courses. Leutenegger et al. [8] use games as tools to teach programming for computer science students, but using multimedia-focused languages. Duchowski et al. [9] use fairlymathe- matical computer graphics algorithms to teach an advanced pro- gramming course. Jordi et al. [10] discuss teaching computer graphics specifically related to information management. But their goal is to make computer graphics relevant to information systems and not basic programming.

2. Details of experiments

The Data Structures with C course is designed from view of what is computation and how do you perform it, how do you write efficient code to make any system as automated and solve real-world problems. The principles of selection of topics and teaching methods are the basics of computer programming that should be introduced with enough details to practice computing with computers. The course is developed with following objectives:

1. Storing and organizing data in a computer so that it can be used efficiently: Students could have used computer for storing and organizing data manually but could have never done the same with programs, here the main goal is to expose the students about storing and organizing data through computer programs.

2. Implement programs that are efficient and fast to execute and also structure the data: Computational thinking is a new concept need to be clearly introduced, including the basic concepts, what it is and is not. Given the problem statement students should solve it by writing the program but the written program has to be efficient and fast to execute.

3. Basic operations on data Structures like stacks, queue, linked list, trees and graphs: Since this is a data structures lab, students need to understand the fundamental data structures and able to explain them without any difficulties.

4. Implementation of operations on stacks, queues, linked list, trees and graphs: Understanding the basic concept theoretically is not a challenging task, but the challenge is to apply and solve the real-world problems with respect to their requirements.

5. Selection of Data Structures for a given application: The criterion for selecting data structure for student practice is that the concepts are not only easy to be mastered, but also powerful enough to practice core conceptions and skills of computational thinking. Finally the students should be able to choose the appropriate method and technique in solving the real-world problem.

The course consists of 60-minutes lectures and 2 hours lab section per week for 14 weeks. Table1 shows how the lecture topics and related laboratories are organized. It can be concluded from table that 80% efforts are devoted to introduce computational thinking.

Table 1. Mapping of theory topics with laboratory exercises.

Week	Topics in theory	Exercises in Library
1,2	Basics of C programming	Programs on sorting and searching.
3	Functions and pointers	Program to create function instead of using inbuilt functions. Program to manage memory, files and pointers.
4,5	Stacks, Queues	Program to perform basic operations (PUSH and POP) on stack. (conversion of expressions). Program to simulate arrangement of manufactured goods in a box by robot on first come first served basisplace. Program to assign jobs for a robot and make it to perform these jobs one after the other in circular manner.
6	Linked lists using stack and queue.	Program using dynamic variables and pointers to simulate the robot for cleanig domestic equipments.
7	Circular and doubly linked list	Program a robot that collects the files from one person to other person either by following a pre-defined path or an automatically generated path..
8-9	Trees and Graphs	Program a robot that visits each person and collects the information that he has. Program a robot that identifies the shortest path of objects and pick and place the object from one place to specified location.
10	Use the concepts of stack ,queues and lists to solve real-world problem	Open
11	Use the concept of trees and graphs to simulate real-world system	Open
12	Use any of the data structure concept to develop a system.	Identify real world problem and implement the same.

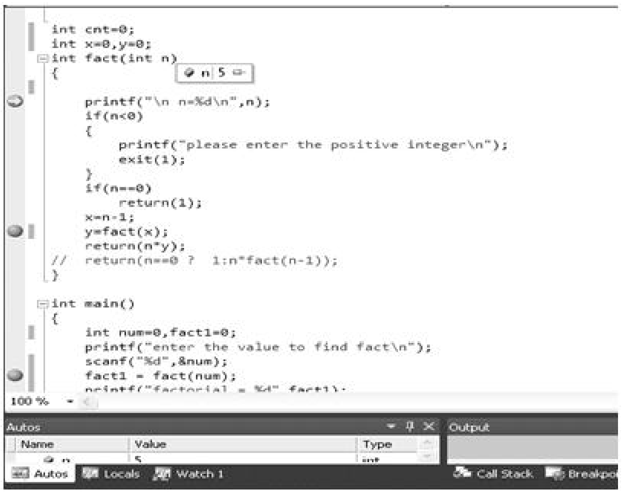
The course is organized based on the books Data Structures using C by Aaron M. Tenenbaum and Introduction to The Design & Analysis of Algorithms by Anany Levitin. Since our students are non-majors, great care has been taken to present the ideas in a more intuitive and interesting manner. This is very important because we just want to explore up to what extent could non-majors taught same as CS-major, and not to teach them to be computer scientists.

A good deal of this course deals with how computations are expressed as algorithm and how the correctness of them is assured. Also, students see how computational thinking applies to various disciplines to solve large scale problems.

1. Teaching Methodology

"What You See Is What You Code", a well known English proverb in Computer science majors that insists how the programming has to be done with respect to given problem statement. We have used few approaches in order to attain the goals of having the students to gain competence in Data structures, algorithms and in C programming. The Visual studio 2010 is used to explain the code in which we can set the break points and make program run step by step and explain the flow of data and values visually. Animations like selection sort, quick sort, binary search etc, will help student grasp the concept very easily. Making teams and giving problem statement to each team and make them to discuss among themselves and come with solution and then again sit separately and writing programs encouraged student involvement in class. Quiz's in the class to increase the competency. Make them to solve new problems with respect to known solved problems has increased confidence to solve any kind of problems. The in detail description of pedagogical practices are as follows:

1.Getting familiar with program logic: The foremost important thing in programming is to understand the code and its logic i.e. flow of data with respect to logic of program. To make students understand the concepts clearly and to give some visual effects we use visual studio to explain the code, the visual studio has a provision to stop the execution of program for a given break point and execute step by step by showing how the data is flowing in the program. This technique will help the student to understand program logic and data flow in program very easily, it also helps in identifying the faults in the program. Visual studio also helps in understanding difficult concepts like recursion, stack, queues, linked list etc. For example recursion is the method which calls itself again and again until specified condition is met; this recursive concept can be demonstrated by visual effect in which the compiler calls the same function more than once by changing the argument values as shown in Figure1



```
int cnt=0;
int x=0,y=0;
int fact(int n)
{
    printf("\n n=%d\n",n);
    if(n<0)
    {
        printf("please enter the positive integer\n");
        exit(1);
    }
    if(n==0)
        return(1);
    x=n-1;
    y=fact(x);
    return(n*y);
//  return(n==0 ?  1:n*fact(n-1));
}

int main()
{
    int num=0,fact1=0;
    printf("enter the value to find fact\n");
    scanf("%d",&num);
    fact1 = fact(num);
```

Figure1.

2. Animations: There are number of animations available on net to teach algorithms like selection sort, quick sort, binary search, sequential search and animations for stacks, queues, linked lists and trees helps student understand the concepts quickly and easily.

3.Class activities: A class activity increases the student involvement in class. In class we have performed an activity in which the class is divided into teams each of four members and different problem statement is given to each team. Then the team is asked to discuss among them self and get the solution. Once all teams get the solution, students are asked to sit separately and write the program. This helps students to share the program logic among others and motivates them to learn subject.

4.Referral programming: This is a common technique in which students solve the new problem with reference to known solution for the similar problem. While working with the real world problems, most of the solutions are difficult so students are asked to refer the known solution and modify them according current required solution.

5.Continuous evaluation: Lab test will be conducted after completion of every level(category) of programs and students will be evaluated for pre-specified marks and depending on the score in the test they will be decided whether they are eligible for next test or not. For example, after completion of demonstration exercises students need to write a test for 10 marks, if a student fail then they are not eligible for next test until they prove that they are fit for second test by writing the first test again.

6. Programming Project: 20% of a student's grade is based on a course project. Students are divided into teams; each team consists of 4 students. This team formation is done by students by themselves. Each team has to select a problem statement and submit it in the form of synopsis. Synopsis has to include problem statement, flowchart, and function/methods used and expected results. We got 11 synopses each with an interesting problem statements like: Demonstration of ATM transactions, online shopping, mobile phone log, vending machine, Election commission, Hospital management, rural one etc. Students were successfully completed the course projects.

4. Mapping of CLOs with PO's:

1. Storing and organizing data in a computer so that it can be used efficiently: At the end of the course students are able to write the programs store and organize the data effectively.

2. Implement programs that are efficient and fast to execute and also structure the data: At the end of the course student are able to write the programs using functions, pointers and dynamic memory allocation techniques that are efficient and fast in execution.

3. Basic operations on data Structures like stacks, queue, linked list, trees and graphs: At the end of the course students demonstrated the different operations on stack, queues, linked lists, trees and graphs.

4. Implementation of operations on stacks, queues, linked list, trees and graphs: Students were able to use the dynamic memory allocation and linked list concepts to implement real-world applications of stack, queues and linked lists.

5. Selection of Data Structures for a given application: As a course project students

successfully identified the problem and data structures involved in it and solved the problem.

5. Analysis:

The objective of the course is providing a foundation of programming principles and data structures that student can and will apply to discipline study. Our goal is to involve student actively in using programming in their discipline and seek a unified approach to teach various non-majors. To evaluate the effectiveness of our approach, during the course we interacted with student personally and took the response about various aspects regarding the course.

Student Reaction: The class consists of 46 students where 30% of the students are from diploma back ground.90% of the students were interested to learn from basic and requested to teach basics of C programming in first week of the course and more than 80% of the students were motivated to learn the course in depth and wanted to learn write effective programs. Following are the feedbacks from the student with respect to our teaching methodologies:

Role of visual studio in understanding the program: Student reacted that this technique was very effective and they can easily understand the logic and data flow of the program. Many of the students started debugging the code using break points which helped them to trace the fault during run time and solve it.

Animations: Students were interested to see the animations and understand how the algorithms work. Many of the students given feedback that they want to learn about creating the animations.

Class activities: The class activity helped students interact with each other, share their ideas and come up with effective coding. Students reacted that they want more of these activities in future.

Referral programming: When the problem definitions like Bus/ flight reservation, Toll tax booth system, cylinder booking systems were given students searched in the internet but they are ended with not getting the solution or with lengthy solutions. Therefore are asked to refer logic and write their own programs which helped them to get solution as well as understand the logic and concepts.

6. Conclusion and future works:

Future work:

There are some more improvements which can be brought in the class that can improve the performance of student. First, class organization can be improved and instead of giving different problem statement single problem statement can be given to analyze the performance of the students. Second, in class exercises were students should write the programs in the system instead of writing in book. This will help student over come syntax errors and develop good programming skill. Third, image processing concepts can be used to teach programming [5].

Conclusion:

The main objective in mind when teaching this course is to make programming more interesting to students who do not view it so, assuming that greater student interest would lead to better performance in class and deeper understanding and appreciation of the subject. Introductory programming courses often teach such basic concepts that it is difficult to design assignments that are simultaneously simple, challenging and interesting. We feel that our teaching methodology provides all three aspects. Our interactions in the classroom, along with the average performance of the students in the assignments, provide encouraging evidence that our approach worked well in these aspects.

We believe that for non-majors with no or less prior background, it is no problem to teach them to implement various data structures and algorithms, to think algorithmically, which are just same as teaching CS-majors. We think we are heading on the right direction. Student's survey shows that the course fulfilled its objectives.

References:

[1] Daniel D. Garcia, colleen M.Lewis, John P. Dougherty, and Mattew C. Jadud. If, you might be a computational thinker!.In proceedings of the 41st ACM SIGCSE. ACM NewYork, USA, 2010; 263-264.

[2] Eric Andew Freudenthal, Mary K. Roy, Alexandria Nicole Ogrey, Tanja Magoc"MCPT: media propelled computational thinking" in

proceedings of 41st ACM SIGCSE. ACM, Newyork. USA,2010; 37-41.

[3]Honh Qin. Teaching computational thinking through bioinformatics to biology students. In Proceedings of the 40th ACM SIGCSE. ACM New York, USA,2009;188-191.

[4] Susanne Hambrusch, Christoph Hoffmann, John T. Korb, Mark Haugan, Antony L. Hosking. A multidisciplinary approach towards computational thinking for science majors. SIGCSE Bull. 2009; 41(1): 183-187.

[5] Amit Shesh, " High-level application development for non-computer science majors using image processing",2012.170-177.

[6] Christopher D. Hundhausen?, Jonathan L. Brown "What You See Is What You Code: A "live" algorithm development and visualization environment for novice learners",2006,22-47.

[7]Davis TA, Geist R, Matzko S, Westall J. tewnZ: visual computing in the form of computer graphics. In: SIGCSE; 2004. p. 125–9.

[8] Leutenegger S, Edgington J. A games first approach to teaching introductory programming. In: Proceedings of SIGCSE; 2007. p. 115–8.

[9] Davis T. Teaching data structures and algorithms through graphics. In: Proceedings of Eurographics—education papers; 2007. p. 33–40.

[10] Jordi L, Esparaza J. Computer graphics for information system programmers. In: Proceedings of Eurographics—education papers; 2010. p. 57–62