Practical Data Structures for non-IT programs

Shridhar T. Doddamani
Asst. Prof, Dept. of Automation and Robotics
BVBCET, Hubli. Karnataka
shridhar_d@bvb.edu

Arunkumar C. Giriyapur
Associate Professor, Dept. of Mechanical Engineering
BVBCET, Hubli.Karnataka, India
aaron@bvb.edu

***Abstract*-Teaching software related subjects to students pursuing an engineering degree other than computer science and aiming to join non-IT industries is really a challenging task. In order to teach subjects like Data structures certain useful activities can be planned which will help the students to understand the concepts quickly and apply them according to their requirements. This in turn will help students aiming to join multidisciplinary industries. A structured approach has been used to map topics covered in theory to the laboratory exercises. Further the laboratory exercises can be categorized to address the ABET criteria. The programming concepts were taught using visual tools and it has been observed that the student involvement has increased along with their performance.**

Keywords*- *Data structures; multidisciplinar;, ABET criteria; Non-IT majors;

I. INTRODUCTION:

In the increasingly competitive world, programming language is considered a basic and necessary skill for any student pursuing a career in an IT industry. Computer science and information Science curricula provide rigorous training in computer programming while non-IT programs such as Automation and Robotics, Electronics and Communications, etc. place at best a secondary or least importance to programming. Teaching programming courses in such programs leads to several challenges.

The objective of the course is to expose students with computational\ logical thinking and fill the gap between computer science fundamentals and programming skills. A student survey was carried out and it was noticed that student's interest in learning the subject was not high because of the following perceptions:

1. Computer science is the only program where knowledge of programming is essential.
2. The course content is highly logical and abstract; thus making it difficult to understand and reproduce.
3. Awareness of the course requirement with respect to the program.
4. If required the course can be learnt in a couple of weeks after completing the degree through external training programs.

The course learning objectives were framed keeping these perceptions in mind and asking the question "is there a unified approach to teach programming to non-IT students".

II. RELATED WORKS

There are many practices in teaching computer programming to non-major students [1,2,3]. 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 problems using a computer, and how to write algorithms, 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 fairly mathematical computer graphics algorithms to teach an advanced programming 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.

III. COURSE DESIGN

It is essential to link the theory with laboratory and hence the laboratory exercises were organized looking at the course topics. The course is developed with following learning objectives:

At the end of the course the student should be able to:

1. Store and organize data in a computer so that it can be used efficiently.
2. Implement programs that are efficient and fast to execute and also structure the data.
3. Perform basic operations on data Structures like stacks, queue, linked list, trees and graphs.
4. Implement operations on stacks, queues, linked list, trees and graphs.
5. Select Data Structure for a given application.

Since the students are non-majors, great care has been taken to present the ideas in a more intuitive and interesting manner. Further the laboratory exercises as categorized into demonstrations, exercises, structured enquiry and open-ended enquiry. Table 1 shows how the lecture topics and related laboratories are organized.

Table 1. Mapping of theory topics with laboratory exercises.

Week	Topics in theory	Exercises in Laboratory	Category
1,2	Basics of C programming	Write a Program to sort given 'n' numbers using Bubble sort, Selection sort, and Insertion sort. Write a program to search an element from a group of available elements using binary search and Quick search.	Demonstration
3,4	Functions, Files and pointers	Write a program to perform the operations of str_cmp and str_cat. Write a program Using files and pointer to create student data base.	
5,6	Stacks	Write a program to simulate the arrangement of components from one container to another by robot using the concept of stack. Write a program to convert given valid parenthesized infix expression to postfix.	Exercise
7-8	Queues	Write a program to simulate the collection of goods from conveyor belt and arrange in container using the principle of FIFO. Write a program to simulate execution of assigned tasks in a circular manner.	
9	Linked lists	Write a program using dynamic variables and pointers to handle student information.	
10-11	Implementation of stack and queues using linked lists	Write a program using stack to simulate the car parking system in front of a mall. Write a program using queue to simulate cleaning of domestic equipments by robot.	Structured enquiry
12	Circular and doubly linked list	Write a program using linked list to simulate the table cleaning in hotel by generating navigation path and following the same path.	
13	Trees	Write a program to traverse binary tree in in-order, pre-order and post-order.	Open ended
14	Graphs	Write a program to identify the shortest path and simulate the operation of pick and place.	

Students came up with real world problems and presented a synopsis on the same. Some of the problems selected were:

(i) Election commission and voting.
(ii) Simulation of clock room operations.
(iii) Movie ticket booking
(iv) Bus controller
(v) Mobile log management
(vi) Finding shortest path for a robot
(vii) ATM transactions
(viii) Shortest path finding robot
(ix) Path planning robot

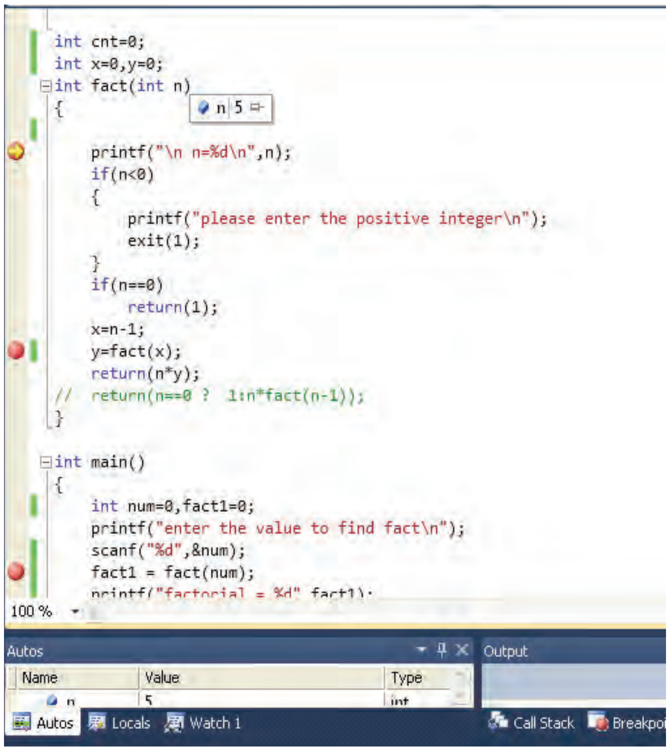
IV. TEACHING METHODOLOGY

The Visual studio programming environment was used to explain the code in which programmer can set the break points, step into the program and explain the flow of code and data visually. Animations like selection sort, quick sort, binary search etc, have helped student to grasp the concept easily. Activities like team formation, problem statement assignment, conceptualization and problem solving encouraged student involvement. Developing quick solutions to new problems using solutions for from earlier problems has increased confidence to solve any kind of problems. The detail descriptions of pedagogical practices are as follows:

1. Getting familiar with program logic: The 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 MS Visual studio as programming tool is used to explain the code, the visual studio has a provision to pause the execution of program for a given break point and execute step by step showing how the data is flowing in the program. This technique will help the student to understand program logic and data flow in program easily, it also helps in identifying the faults in the program. Visual studio also helps in understanding different 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.

Figure 1 shows the how a variable value can be watched even while program is still executing. This technique will help programmers to identify the faults during run time and fix them.

2. Animations: A number of animations available on internet to teach algorithms like quick sort, selection sort, sequential search, binary search and animations for stacks, queues, linked lists and trees were used to help the students understand the concepts.



```
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);
    printf("factorial = %d",fact1);
```

Name	Value	Type
n	5	int

Figure 1 visual studio program simulation

3. Class activities: A class activity increases the student involvement in class. The class activity is performed 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 conceptualize come up with a 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 to 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. 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 in 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 in the form of synopsis. The 11 synopses are received each with an innovative 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.

V. Analysis:

At the end of the course feedback was collected from the students. The questions asked were:

1. The laboratory has reinforced the concepts which are taught in theory classes.
2. Visual programming has helped a lot in understanding the data flow and logic of the program.
3. The concepts taught will help in projects.
4. The exercises done will help you to identify the real world problems and solve.
5. The exercises will help you to work in a team and communicate effectively.
6. The presentations given by students have helped to stay motivated towards subject.
7. Learning algorithms with animation has led in easy understanding of algorithms.
8. The instructor has succeeded in solving your problems.
9. You can solve the given problem in limited time with available resources

The response from the students is plotted in Figure 2

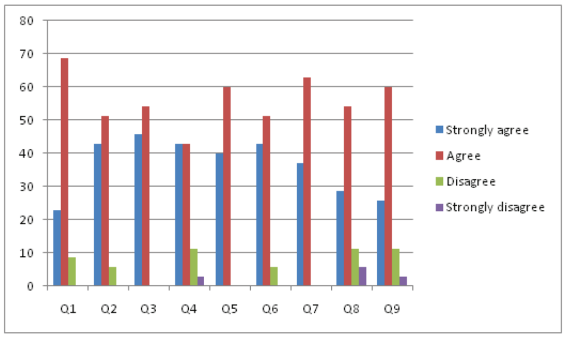


Figure 2: feedback analysis

Role of visual studio in understanding the program: Student opined that this technique was very effective in explaining 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.

Class activities: The class activity helped students interact with each other, share their ideas and come up with effective coding.

Referral programming: Student could refer to the solutions developed for previous problems and use the logic and code. This helped them not to reinvent the wheel and quickly develop solutions for new situations.

VI. Conclusion and future works:

The main objective when teaching this course was to look at the student perceptions so as to make programming more interesting which would further lead to better performance in class and deeper understanding and appreciation of the subject. Interactions in the classroom, along with the average performance of the students in the assignments, provide encouraging evidence that the approach worked well in these aspects. Student's survey shows that the course fulfilled its objectives.

A few improvements have been suggested to improve the student performance. First, all the teams can be given a single problem statement instead of several to generate a variety of solutions to the same problem. Second, students should type the programs immediately block by block to eliminate syntax errors and develop good programming skill. Third, image processing concepts can be used to teach programming [5].

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: a first step. 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