A Procedural Way of Teaching Procedural Programming Language

M. ABDULLAH-AL-WADUD
Department of Software Engineering
King Saud University
Diriyah, Riyadh
KINGDOM OF SAUDI ARABIA
mwadud@ksu.edu.sa

Abstract: - A procedural programming language such as C and Pascal is usually the first programming language the most the students come across. However, in most curriculums, the concept of 'procedure' is introduced to the students after almost half of the course is covered. Many students are then found to be reluctant to get the fruit of procedural style of programming. They tend to write big programs without making use of procedures, functions or subroutines. Thus they do not feel the charm of programming as it should be encountered to solve a problem. This paper proposes a guideline of designing the curriculum of procedural programming language courses where thinking in a procedural manner is emphasized so that the learners can follow the art of procedural languages more comprehensively that the traditional style of curriculum design.

Key-Words: - Procedural programming language, teaching, procedure, function, subroutine, top-down approach

1 Introduction

Procedural programming is a computer programming technique where a program is divided into different modules such as function, subroutine and procedure. Most undergraduate programs related to computer science and engineering introduce students to a procedural language such as C or Pascal to start learning computer programming. However, many students are not found to perform to the level of expectation on such courses. Many researchers have been trying to find the cause of the problems faced by the students, and make suggestions to improve the teaching and learning of such courses [1-3]. Comprehensive literature reviews on such research works can be found in [2] and [3].

Traditionally, a bottom-up approach is followed to teach the students programming. It starts with introducing the very basic syntax of the programing language which does not give much idea about the relation with the problems that might need computer programming to be solved. At this point, many students, especially those from the non-IT departments, feel lack of motivations to study the course. Traditional bottom-up approaches cannot attract the students to feel the charm of such courses.

Another important aspect of such programming course is to teach the procedural way of programming trough making procedures, subroutines or functions. However, in traditional curriculums, when the writing of procedure is introduced after teaching most of the basic syntax, conditions, loops, etc., most students do not feel the motivation to use the procedural techniques unless it is posted as a condition.

This paper proposes a guideline to design the curriculum of procedural language courses that will motivate the students to learn and apply the procedural style of programming languages. Application of this style of teaching has also been proved fruitful to eight sections of C programming language in past few years.

The rest of the paper of organized as follows. Section 2 introduces the proposed style of teaching procedural programming language, while Section 3 presents the effectiveness of following this way of teaching in several sections. Finally, Section 4 concludes the paper.

2 The Proposed Method of Teaching

The main focus of the prosed method of teaching is to give a comprehensive introduction to procedural way of programming so that students feel more comfortable to work with procedures in program design and have the hands on experience of solving different problems following a procedural way of problem solving approach.

The proposed way of teaching suggests to first introduce the syntax of very basic operations such as displaying outputs on screen and getting simple inputs. Afterwards, instead of focusing on detail input and output commands, different expressions, conditional operations, loops, etc., as done in the traditional way of teaching, the proposed method directly moves into the procedural approach. As for example, students can be introduced to the simple mathematical functions first as shown in the C programming code in Fig. 1. This gives a very simple but easy-to-follow example for the student to understand how to use a function/procedure by just calling its name without thinking about the detail of the calculations to be done.

```
#include<stdio.h>
#include<math.h>

void main(){
      int a,b;
      scanf("%d",&a);

      b = abs(a);

      printf("%d",b);
}
```

Fig. 1. The first program using function.

After the introduction to a simple program using function as shown in Fig. 1, students can then be introduced to making of another simple program using a function. For example, let us consider a problem to input an integer and calculate the square of it. To the students, it can be explained as a three step process: input a number, calculate the square and display the number. These three steps can be coded as shown in Fig. 2.

```
#include<stdio.h>

void main(){
      int a,b;
      scanf("%d",&a);

      b = square(a);

      printf("%d",b);
}
```

Fig. 2. Solving a problem in a procedure-oriented way.

Note that the proposed method does not suggest to explain how the square is calculated at this point. It prefers to focus on the overall design in terms what are needed to be done; not thinking too much about how is every step done. Thus a large problem is broken into some sub-problems. The detail of the calculation of a sub-problem is to be thought during the design of that particular procedure/function only.

After designing the program, the different functions/procedures can be solved one after another; following a top-down fashion. Fig. 3 shows an implementation of the `square()` function. At this point, different syntax such as parameters and return value can be explained.

```
int square(int a){
      int s;
      s = a*a;
      return s;
}
```

Fig. 3. The detail implementation of the function.

In such a way, students can be encouraged to think in a procedural way of solving any problem, be it a simple or a complex problem.

After the introduction to a simple program as in Fig. 2, we can move forward to a bigger problems. Let us take, as an example, a problem of showing the integers from 1 to 10 in one line and then showing the integers from 21 to 30 in the next line. A traditional way of teaching loops usually encourages to use two loops to solve this problem. However, the proposed procedural approach suggests to teach solving it in a procedural way that involves the program design as presented in Fig. 4, which designs the solutions in two steps: to print 1 to 10, and then to print 21 to 20. Note also that the same function is used twice as the purpose is the same while the only differences are in the parameters. Such a program shows the power of procedure where a single implementation of a function can serve various purposes depending on the parameters. Such an example can highly motivate the students at the very beginning to make use of procedures whenever possible.

```
#include<stdio.h>

void main(){
      printIntegers(1,10);

      printIntegers(21,30);
}
```

Fig. 4. The first program using function.

Let us consider another problem. We need to calculate the average of 5 integers stored in an array. So, what will be the program design? A procedural approach may have two steps, calculate the average and print the result, as shown in Fig. 5.

```
#include<stdio.h>

void main(){
 int a[5] = {23, 56, 34, 655, 39};

 float avg = average(a);

 printf("%f",avg);
}
```

Fig. 5. The first program using function.

```
float average(int a[]){
      int sum;
      sum = summation(a);

      float avg = sum/5.0;

      return avg;
}
```

Fig. 6. The first program using function.

The next step is to implement the `average()` function. It also involves few steps such as calculating the summation and calculates the average. Fig. 6 shows an implementation. Thus a function/procedure can be thought of a smaller problem, which can again be designed in a procedural approach if it seems to be complex. However, while designing a program/procedure, the details of the other procedures, which are called from it, need not to be focused on. Our proposed method of teaching encourages thinking a function as a black box, something similar to the blocks-based programming [4-5], which can 'somehow' perform a command. Such a high level thinking enhances capabilities of the students in designing big and complex programming solutions.

When the students are taught to approach every problem or sub-problem in a procedural way, they find it a very methodical way to follow. And after few tries, they can catch the power of it, and apply to different problems

3 Performance Evaluation

In the department of Industrial and Management Engineering in Hankuk University of Foreign Studies, South Korea, the traditional curriculum was followed while teaching the C programming language during two years, 2009 and 2010, in 4 sections. Afterwards, the proposed approach was applied in the following three years in 6 sections. Table 1 presents the performances of the students in different years. It shows that the performances of the students got significantly improved after the adoption of the procedural teaching style in 2011. It was also found that student had become more interactive with the instructor afterwards. This shows the increased level of motivations that was geared by the proposed teaching technique.

Table 1 The performances of the students in the C programming course

Year	**Average marks** (out of 100)
2009	78.052
2010	79.013
2011	82.568
2012	85.984
2013	85.560

4 Conclusion

This paper present a procedural style of teaching procedural languages such as C and Pascal. The teaching method fosters the introduction of procedures to be done at a very early stage of the course, and suggests that all the problems should be encouraged to be solved using procedures/functions. By doing so, students can get enough hands on experiences of procedural program design and implementations. It also shows a problem solving idea to the students where they can call the different procedures confidently and effectively when needed. Experiments on 10 different sections have also proven the fruitfulness of this teaching style. This style of teaching has a great potential to be applied on other technical courses too.

References:

[1] Anthony Robins, Learning edge momentum: A new account of outcomes in CS1, *Computer Science Education*, Vol. 20, No. 1, 2010.

[2] A. Robins, J. Rountree and N. Rountree, Learning and teaching programming: A review and discussion, *Computer Science Education*, Vol. 13, No. 2, 2003, pp. 137 - 172.

[3] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin and J. Paterson, A survey of literature on the teaching of introductory programming. *SIGCSE Bulletin*. Vol. 39, No. 4, 2007, pp. 204–223

[4] David Werntrop, Wilensky Uri, The challenges of studying blocks-based programming environments, in the *Proceedings of IEEE*

Blocks and Beyond Workshop (Blocks and Beyond), 2015.

[5] D. Weintrop and U. Wilensky, To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming, in the *Proceedings of the 14th International Conference on Interaction Design and Children*, New York, NY, USA, 2015, pp. 199–208.