Students’ difficulties in and opinions about designing algorithms according to different instructional applications

Ozgen Korkmaz*
Mevlana University, Education Faculty, Department of Computer and Instructional Technology, Konya, Turkey.

Received: 21 October 2011; accepted: 18 December 2011

Abstract

The present study aims to reveal whether the difficulties experienced when designing algorithms differ for students who were directly taught a programming language and those who were only taught algorithm, and students’ opinions about these difficulties. It is a pretest-posttest control group semi-experimental study with both quantitative and qualitative characters. The sample group consists of 45 students. The study data were collected by using an Interview Form, and an Algorithm Skill Test. The obtained quantitative data were subjected to arithmetic mean, standard deviation and t-test (p<0.05). As a result: The students have insufficient levels of algorithm designing skills, and a great majority of them experience difficulties in the process of designing algorithms. The students’ opinions revealed that the group simply taught algorithms have less difficulty in planning how to solve a problem, and solving problems that require mathematical operations when compared to the group directly taught a programming language.

Keywords: Programming and programming languages, Algorithm design skill, Teaching/learning strategies, Difficulties

1. Introduction

Individuals who are to work in fields that require knowledge about information technologies, computer training and programming may need sufficient knowledge about computer programming. Since knowledge of programming languages is of great importance for programming skills, programmers should arguably possess a maximum level of knowledge about the characteristics of programming languages. Therefore, it is very important to teach programming skills [1] as these skills are regarded as among the fundamentals of both computer sciences, and computer teacher training programs. Possessing programming skills is also important for a career in the information industry. Thus, although the popularity of the courses of introduction to programming has been on the rise in higher education, failure and disappointment are quite common as acquiring this skill is hard and requires a lot of effort [2, 3]. In fact, Gomes and Mendes [4] state that problems are experienced in the process of teaching programming skills in terms of academic achievement and students’ satisfaction levels. One of the main reasons behind students’ failures in programming courses may be the traditional methods commonly used to teach programming languages, and the instruction tools that disregarded students’ characteristics. In addition, the general lack of knowledge among students about the problem solving process, and about how to design a program or algorithm [5–7], may also lead to failures. Boulay [8] underlined the importance of problem solving skills in the programming process. Similarly, Spohrer and Soloway [9], Winslow [3] and Soloway et al. [10] stress the fact that many students experience problems in understanding the algorithmic structure in the programming process.

Students’ problems in learning programming skills can certainly be attributed to many factors. Chiefly among them is the inherent complexity of the programming process, the tendency to overlook the fact that this process requires special teaching approaches, the need for strenuous efforts to learn, and the lack of skills to use mathematical knowledge in the programming process [5]. Jenkins [6] listed the main teaching methods that might cause problems in teaching programming skills as follows: 1. The instruction process based on the overall classroom environment, and the lack of one-to-one teaching with students. 2. Instructors who disregard their students’ learning styles, and overlook individual learning preferences. 3. The passive attitude in learning environments of students who simply watch presentations or follow what is written on the board [11]. The disregard for students’ learning styles in the learning process may result in learning difficulties for many students [12, 13], for these students usually prefer active and interactive environments. Furthermore, most teachers tend to base their teaching on memorising verbal presentation of information and details, instead of establishing, and emphasising, a link between algorithms, and problem solving in the process of teaching programming skills, encouraging students to be active, and presenting them with up-to-date examples.

Programming skills can mainly be examined in two categories, which are (a) the perspective of professional software engineering that requires experience and teamwork, and usually aims at large-scale software projects, and (b) the educational-psychological perspective that is rather based on individual learning, and requires program writing skills for a problem with defined limits [14, 15]. The present study deals with the problems and solutions in individual programming skills or the educational-psychological perspective which influences one’s possession of programming skills to solve any defined problem. Research in the field of education frequently addresses the problems experienced in learning abstract concepts. Such difficulties usually arise since abstract concepts are not adequately associated with real life or are not made concrete enough [16]. These problems are also frequently experienced in teaching programming skills, and are always found in the programming process which requires multiple abstract operations [17]. Various solutions have been offered to overcome the above mentioned learning difficulties experienced by students in acquiring programming skills. One solution for students is to take an introductory course to programming to learn effective algorithm design skills before directly taking a programming course, which could be an effective solution for the problems experienced in teaching programming skills. In other words, prior learning of algorithm design skills is argued to be important for better learning of programming skills [18]. However, although the literature lacks adequate evidence of whether teaching algorithm skills prior to programming training contributes to programming skills, and if it does, what the extent of such contributions is, it could be argued that algorithm designing skills could be important for developing programming skills and those students’ possible difficulties in the process of algorithm designing may adversely affect their programming skills. On the basis of this assumption, the present study aims to reveal whether the difficulties experienced when designing algorithms differ for students who were directly taught a programming language, and those who were only taught algorithm, and students’ opinions about these difficulties.

Sub-Problems

1. Prior to the application, are groups equivalent in terms of designing algorithm skills?
2. What are the difficulties experienced by students when designing algorithms?
3. Do the difficulties differ for students who were directly taught a programming language, and those who were only taught algorithm logic?
4. What do students think about the difficulties they have in the process of designing algorithms?

2. Method

2.1. Research Model

The present study is a pretest-posttest control group semi-experimental study with both quantitative and qualitative characters.

*Corresponding author: Tel: +90-505-319-2785; fax: +90-332-241-1111. E-mail address: ozgenkorkmaz@gmail.com or okorkmaz@mevlana.edu.tr (O. Korkmaz).
2.2. Study Group

Study group of research consists of a total of 45 students in their first year in the Department of Computer and Instruction Technologies in an Education Faculty, of whom 23 are female and 22 are male. The students were randomly grouped into two experiment groups and equal numbers in terms of gender. Experiment Groups I and II were also randomly selected. Table 1 summarises the gender distribution of the students in the study groups.

2.3. Experimental procedures

For five hours a week for four weeks, experiment group I was taught the concepts of program, command, algorithm, flowchart, constant, condition, cycle and basic commands in Visual Basic. Next, example questions were analysed about how to develop a program piece. The students were presented with explanations and examples about what strategies they should follow in the problem solving process. At every stage of the instruction, the students were made to develop algorithms to solve different exemplary problems and were provided with instant feedback and corrections for their mistakes. At the end of each session, they were given problem statements as coruscations, which they were asked to solve for the next session. The assignments were checked at the beginning of each session and appropriate feedback and corrections were provided.

Similarly, for five hours a week for four weeks, experiment group II was also taught the concepts of variable, Visual Basic editor, program, command, algorithm, draw chart, constant, condition, cycle and basic commands. The students were provided with instant feedback and corrections for their mistakes. At the end of each session, they were given problem statements as coruscations, which they were asked to solve for the next session. The assignments were checked at the beginning of each session and appropriate feedback and corrections were provided.

The exemplary problems and extracurricular activities used in both groups are the same. However, solutions were formulated as algorithms in experiment group I and as pieces of Visual Basic program in experiment group II. As a result, experiment group I used algorithms to solve a problem, while experiment group II used Visual Basic commands.

2.4. Data collection instrument

In order to identify students’ opinions about the kinds of problems they experience in algorithm designing process, an interview form was developed and the students were asked to respond in writing to the questions in this form. During the process of formulating the questions, we considered that the process of algorithm developments takes place in two stages, which are designing a solution by analyzing the problem and applying the designed solution. As a matter of fact, the literature contains evidence that these stages are the basic stages in problem solving process [19, 20, 21]. In this framework, the students were asked the following questions:

1. What are the problems you experience in the process of problem analysis and solution planning when you are to write an algorithm to solve a problem?
2. What are the problems you experience when writing a solution you design in the form of an algorithm?
3. What do you think is the primary problem of first priority that you are faced with during the entire process of developing an algorithm?

In order to measure algorithm development skills, the students were administered a test form containing four open-ended algorithm questions. The students were asked to write down the solution they project to solve the problem in each question, to write the algorithm and to draw the flowchart for their solutions, and to prepare check tables for the algorithm. These questions aiming to measure students’ algorithm skills were examined by two field experts and they were used after being corrected in accordance with the opinions of these experts. The test contains the following questions:

1. Write down the algorithm that divides the greater by the smaller of two given integers simply by using subtraction, and give quotient and remainder.
2. Write down the algorithm that calculates the value of function depending on a given variable n:
   \[ \sum_{x=1}^{n} (2/3)x + (1/2)x \]
3. Write down the algorithm that inverts a list of a given number of students including their ID numbers and names (that replaces the first student by the last entry in the index).
4. Write down the algorithm that gives the monthly profit rates of N number of companies for a year, and find for each company average annual profit and the months with maximum and maximum profit.

2.5. Data analysis

The data obtained through the algorithm skill test were evaluated using a rubric. The rubric includes three categories: ‘problem analysis’, ‘solution design’, and ‘mathematical transfer’. Analysis and solution design are among the stages of the problem solving process [19, 20, 21]. Mathematical transfer refers to the mathematical expression of a solution. One cannot algorithmize a solution that cannot be mathematically expressed. Therefore, it may be useful to consider in this mathematical transfer process skill along with the stages of problem-solving skill. Skills that could be included under each category were identified by a literature review, and with the help of four field experts including a computer engineer, a software engineer and two education technologists.

The skills under each category were separately examined by the researchers. Each skill was rated with “0” in cases where students failed to respond or gave fully wrong responses, with “1” even there were incomplete responses or small errors, and with “2” when the response was correct. Thus, each question was evaluated for ten different skill levels, and with a rating range between 0 and 20. The total score for the four algorithms in the test was calculated to be between 0 and 80. Subsequently, separate evaluations of the researchers were compared to arrive at a common evaluation. Each category and total scores were then compared to the student scores ranging between zero and 100. Below is a presentation of the examples of the skills under each category and the examples rated with 0 and 1 for these skills. Since cases of perfect correctness do not involve relativity, no examples are provided for such cases.

a. Problem Analysis: The following skills under the problem analysis category were analyzed by examining the students’ expressions of their designed solutions. Basic skills falling under this category could be summarized as follows:

1. The ability to express a problem as a whole: Rather than thinking about a problem expressed in the problem statement on a particular example, it refers to perceiving it as a whole, being able to consider all given circumstances and mentally designing the given information to solve a problem, the values and the operations of whom 23 are female and 22 are male. Students who totally failed to express their ideas or analyze a problem as a whole were rated with “0” for the skill in question. The following example is provided for the responses found to be totally incorrect since there was no mention of how to arrive at the solution: “We identify N number of index variables for profit rates, and I arrive at solutions by doing calculations for each company”. An example for the responses rated with “1” for minor errors or incompleteness is as follows: “We continuously identify N number of index variables for profit rates, and I arrive at solutions by doing calculations for each company”.

b. Solution Design: The following skills under the solution design category were analyzed by examining the students’ solutions. Basic skills falling under this category could be summarized as follows:

1. The ability to verbally express a solution: This represents the ability to verbally express a solution or to express it in a totally incorrect way rated with “0” for the skill in question. The following example is provided for the responses found to be totally incorrect since the student did not mention the way to

---

Table 1. Gender Distributions of the Study Groups

<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th>Female</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Experimental Group</td>
<td>11</td>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>II. Experimental Group</td>
<td>11</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
<td>23</td>
<td>45</td>
</tr>
</tbody>
</table>

---

Special Issue               (2012)
arrive at a solution: “We enter an arbitrary number of companies. In the cycle, we increase the cycle by one if the cycle value is smaller than the number of months, while we enter the number of months by increasing it by one if it is smaller than 12. Next, we take the value as 0 in a different cycle, assign the greatest value for maximum profit, and thus calculate the minimum profit”. An example for the responses rated with “1” for incomplete solution expressions is as follows: “Two numerical entries will be done separately. Of these numbers, we subtract the smaller one from the greater one until it is less than the smaller, and increase the quotient by one.” This solution was considered as incomplete since the student did not express the cycle operation.

3. The ability to correctly design a solution: It refers to the level of logical correctness for a solution verbally expressed by a student. Students who totally failed to verbally express the solution or expressed it in a totally incorrect way were rated with “0” for the skill in question. The following example is provided for the responses found to be totally incorrect since the expected solution involves reversing a string rather than writing a string by an inverse cycle: “I use the sequence a(x) while taking the ID numbers, and names of X number of students, and write down the entry values in a cycle. Each entry number is placed in the sequence and finally we write the sequence by performing an inverse cycle”. An example for the responses rated with “1” for incomplete solution expressions or minor errors is as follows: “In solving this problem, we should first find out which number of the two is greater. Next, we continuously subtract the smaller number from the greater one until the subtrahend is less than the minuend”. This solution ignored the number of subtraction operations to be performed in order to arrive at the quotient.

b. Solution Design: The following skills under the development category were analysed by examining the students’ algorithms, and flowcharts.

1. Ability to algorithmically express a solution: It refers to the ability to correctly express a solution whose logic is mentally designed in the form of an algorithm. The main indicator of an algorithm’s accuracy is the extent to which one can attain at the correct solution for each value using the same logic. The students whose algorithms were totally incorrect or fairly incomplete were rated with “0” for the skill, and the following examples could be provided for the responses rated with “1” for small errors or incompleteness in the algorithms:

```
A [nn]

n= "Please enter the sequence number."
i=1, j=1
Loop: If i<n
  j= "Input Student’ Name."
  j= "Input Student’ Number"
  j=+1, Go Loop 2
  j=i+1, Go Loop
End: If i>n

j=0

j=i+1, Go Loop
```

1. pt: Algorithm not finalized by expressing the solution

2. Ability to correctly use the variables: It refers to the ability to distinguish between the name of a variable and its value and to solve a problem by using the least possible number of variables in an algorithm. Students whose variables were totally incorrect or incomplete were rated with “0” for this skill, while those with small errors or missing parts in the variables were rated with “1”. The following examples are provided for such evaluations:

```
A [nın]

n= "Please enter the sequence number."
i=1, j=1
Loop: If i<n
  j= "Input Student’ Name."
  j= "Input Student’ Number"
  j=+1, Go Loop 2
  j=i+1, Go Loop
End: If i>n

j=0

j=i+1, Go Loop
```

1. pt: No mention of inverse cycle

3. Accuracy of cycles: It refers to the ability to use cycles to easily perform repeated operations, to express the beginning, and end of a cycle when designing cycles, to avoid cases of infinite cycles, and to use multiple cycles without overlapping. Students whose cycles were totally incorrect or incomplete were rated with “0”, and those with incompleteness or small errors in their cycles were rated with “1”, for which the following example could be presented:

```
S=0, K=0
Input A, Input B
If A>B K=A-B
If A=B K=K-A, S=S+1
If K=0 Go 6
S=K+1
If K<K=K+B, print K, S
```

1. pt: The name and indices of an index variable not distinguished.

4. Accuracy of Condition Statements: It refers to the ability to correctly express appropriate condition statements that could control cycle or general conditions. The students whose condition statements were totally wrong or incomplete were rated with “0” for this skill, while those with small errors or incompleteness in their conditions statements were rated with “1”, which could be exemplified as follows:

```
Read N; n=1, mount=1
Profit= ("Enter the monthly profit")
Year_Profit=Profit*12
mouns= mouns+1
enlays= Year_Profit, enlays= Year_Profit
If moun<13
```

1. pt: Cycle blocks, and conditions not mentioned.

```
Small=0, Number1= "Input Number1"
Big =0, Result=0, Numbers=0
 Remainder=0, Number2= "Input Number2"
 Input Number1, Number2
 Loop: If Number1>Number2
  Number2=Small, Number1=Big
  Go Loop 2
 Else Result=Remainder
  Number1=Small, Number2=Big
  Go Loop2
```

1. pt: The end of the 1” cycle block is unidentified.

```
Input A, Input B
If A>B K=A-B
If A=B K=K-A, S=S+1
If K=0 Go 6
S=K+1
If K<K=K+B, print K, S
```

1. pt: No need for definitions by taking N’s value as 1 at Step 7.

```
```
```
```
5. Brevity of the algorithm: It refers to the ability to design the shortest solution. None of the students wrote algorithms that were too long. However, the students who failed to write the algorithm were rated with “0” for the skill. Below are examples for the responses rated with “1” since their algorithms were partially long:

```
division=0, divisor=0, remainder=0
Input x, y
If x>y remainder=x
Label1: Loop i=1 to n, i=i+1
x=x remainder
If remainder>y Go Label1
Else is=division, y=divisor
Else
Print division, divisor, remainder
```

The student attempted to control the cases that do not require controlling, writing an unnecessarily long algorithm.

e. Mathematical Transfer: The following skills under the mathematical transfer category were analysed by examining the mathematical expressions used by the students in their algorithms.

1. Accuracy of the Mathematical Operators Used: It refers to the ability to correctly use operators, and to consider operation sequence while performing mathematical operations. The students who used the mathematical operators totally incorrectly in their algorithms were rated with “0” for this skill, while those with smaller errors concerning mathematical operators were rated with “1”, examples for whose responses are given below:

```
Input a, b
Biggest=a
If a>b then Biggest=a
Label 1: a=a-b
If a>b then Go Label1
Go Label1
Else print count=count+1
```

```
Input number (a)
Label 1: Loop n=1 to a
Total=0
Input number (a)
Label 1: Loop n=1 to a
Total=0
n=n+1
go Label1
Print total
```

2. Mathematical expression of the solution: It refers to the ability to mathematically express the logical solutions. The students who completely fail to mathematically express the solutions they designed were rated with “0” for the skill and “1” for this point was given for smaller errors in mathematical expression of solutions, for which examples are given below:

```
array[n,2], say=0
Loop 1: If j<n array[j,1]="Input name"
Array[2]= “Input number”
Count=Count+1, j=j+1
Loop 2: If j<n List (j,1)=reverse list [j], j=j+1
Go loop1
print cont
```

```
If x>y remainder=x
Label 2: Loop i=1 to n
y=x remainder
If remainder>y Go Label2
Else
Print division, divisor, remainder
```

2. Mathematical expression of the solution: It refers to the ability to mathematically express logical solutions. The students who completely fail to mathematically express the logical solutions were rated with “0” for the skill and “1” for this point was given for smaller errors in mathematical expression of solutions, for which examples are given below:

```
Array[2]= “Input number”
Count=Count+1, j=j+1
Loop 2: If j<n List (j,1)=reverse list [j], j=j+1
Go loop1
```

The student attempted to control the cases that do not require controlling, writing an unnecessarily long algorithm.

The interview form used to identify the students' difficulties

Table 2. Differentiation between the Students’ Difficulties according to Categories in terms of Pretest Scores

<table>
<thead>
<tr>
<th>Skills according to Categories</th>
<th>Groups</th>
<th>N</th>
<th>X</th>
<th>ss.</th>
<th>t</th>
<th>sd</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Analysis</td>
<td>I. Exp. Gr.</td>
<td>22</td>
<td>19.32</td>
<td>14.86</td>
<td>.610</td>
<td>43</td>
<td>0.545</td>
</tr>
<tr>
<td></td>
<td>II. Exp. Gr.</td>
<td>23</td>
<td>16.85</td>
<td>12.23</td>
<td>1.176</td>
<td>43</td>
<td>0.246</td>
</tr>
<tr>
<td>Solution Design</td>
<td>I. Exp. Gr.</td>
<td>22</td>
<td>37.50</td>
<td>14.94</td>
<td>1.256</td>
<td>43</td>
<td>0.216</td>
</tr>
<tr>
<td></td>
<td>II. Exp. Gr.</td>
<td>23</td>
<td>33.15</td>
<td>9.36</td>
<td>1.176</td>
<td>43</td>
<td>0.246</td>
</tr>
<tr>
<td>Mathematical Transfer</td>
<td>I. Exp. Gr.</td>
<td>22</td>
<td>27.84</td>
<td>13.42</td>
<td>1.122</td>
<td>43</td>
<td>0.268</td>
</tr>
<tr>
<td></td>
<td>II. Exp. Gr.</td>
<td>23</td>
<td>23.97</td>
<td>9.49</td>
<td>1.122</td>
<td>43</td>
<td>0.268</td>
</tr>
</tbody>
</table>

As is clear from Table 2, according to pretest there is no significant difference between the problems experienced in the algorithm development process by the students who were directly taught a programming language and those who were only taught algorithm in terms of the categories for basic algorithm skills. Thus, it could be argued that the terms of categories of groups prior to the application of basic skills can be said to be equivalent.

3. 2. Difficulties challenging students during algorithm design

Table 3 summarises the students’ means and standard deviations converted into standard posttest scores in the categories concerning algorithm designing skills.

Table 3. Posttest Score Means for Basic Algorithm Skills according to Categories

<table>
<thead>
<tr>
<th>Skills according to Categories</th>
<th>N</th>
<th>X</th>
<th>ss.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mathematical Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As shown in Table 3, according to posttest scores the students have a moderate level of algorithm designing skills ($\bar{X} = 62.84$). An independent examination of the categories reveals that the students had the greatest difficulty during the algorithm designing process in the Development category ($\bar{X} = 60.76$), while they had the least difficulty in Mathematical Transfer ($\bar{X} = 65.67$). So it could be argued that the students had greater difficulty at the level of algorithm development to solve the problems they analysed during the process of algorithm development. Table 4 summarises the arithmetic means and standard deviations for the basic skills under these categories.

| Table 4. Basic Skills in the Process of Algorithm Development |
|-----------------|-----------|-----------|
| Skills                     | N | $\bar{X}$ | ss. |
| 1. The ability to express a problem as a whole     | 5.38 | 1.850 |
| 2. The ability to verbally express a solution      | 5.07 | 1.827 |
| 3. The ability to correctly design a solution      | 5.00 | 1.895 |
| 4. Ability to algorithmically express a solution   | 4.62 | 1.542 |
| 5. Ability to correctly use the variables          | 4.73 | 1.698 |
| 6. Accuracy of cycles                              | 45  | 4.93    | 2.060 |
| 7. Accuracy of Condition Statements                | 5.36 | 1.861 |
| 8. Brevity of the algorithm                        | 4.56 | 1.673 |
| 9. Accuracy of the Mathematical Operators Used     | 5.67 | 1.732 |
| 10. Mathematical expression of the solution        | 4.82 | 2.026 |
| Total                                            | 5.01 | 1.410 |

As seen in Table 4, the students’ competence regarding their skill in the algorithm development process ranges between $\bar{X} = 4.56$ and 5.67 (out of a total of eight points). The overall mean is $\bar{X} = 5.01$. Thus, the students have a moderate level of algorithm development skills. An examination of the means reveal that the students had greater difficulty in the process of algorithm development to solve a problem in the following skills: The ability to express an algorithm in a shorter way ($\bar{X} = 4.56$), Expressing a solution in the form of an algorithm ($\bar{X} = 4.62$), The ability to correctly use the variables ($\bar{X} = 4.73$), and Mathematical expression of a solution ($\bar{X} = 4.82$). On the other hand, they had less difficulty in the skills of the use of mathematical operators ($\bar{X} = 5.67$), the use of condition statements ($\bar{X} = 5.36$), and the ability to analyse the problem as a whole ($\bar{X} = 5.38$). Therefore, the students arguably experienced less problems in the algorithm development process at the levels of problem analysis and the use of mathematical operators, and condition statements, while they had more problems at the levels of the ability to express a solution as shortly as possible and in the form of an algorithm and correctly using variables.

3. 3. Differences between the problems experienced by the students who were directly taught a programming language and those who were only taught algorithm

Table 5 summarises the posttest results concerning the differences between the programming skills of the students who were directly taught a programming language and those who were only taught algorithm according to the categories for algorithm development skills.

| Table 5. Differentiation between the students’ difficulties according to categories in terms of posttest scores |
|---------------------------------------------------------------|-----------|-----------|-----------|
| Skills according to Categories/Groups                     | N | $\bar{X}$ | ss. | t | sd | P |
| Problem Analysis                                           | 22 | 66.55    | 26.92 | 0.593 | 43 | 0.556 |
| I. Exp. Gr.                                                | 23 | 62.52    | 17.88 | 0.593 | 43 | 0.556 |
| II. Exp. Gr.                                               | 23 | 61.73    | 22.49 | 0.320 | 43 | 0.751 |
| Solution Design                                            | 22 | 61.73    | 22.49 | 0.320 | 43 | 0.751 |
| I. Exp. Gr.                                                | 23 | 59.83    | 17.19 | 0.320 | 43 | 0.751 |
| II. Exp. Gr.                                               | 23 | 65.73    | 26.37 | 0.018 | 43 | 0.986 |
| Mathematical Transfer                                      | 22 | 65.61    | 16.56 | 0.442 | 43 | 0.661 |
| I. Exp. Gr.                                                | 23 | 64.05    | 20.69 | 0.442 | 43 | 0.661 |
| II. Exp. Gr.                                               | 23 | 61.70    | 14.72 | 0.442 | 43 | 0.661 |

As is clear from Table 5, according to posttest there is no significant difference between the problems experienced in the algorithm development process by the students who were directly taught a programming language and those who were only taught algorithm in terms of the categories for basic algorithm skills. Thus, it could be argued that direct teaching of a programming language and simply teaching algorithms do not have any effect on the problems that the students experience in algorithm development according to the categories.

3. 4. Students’ opinions about the difficulties they experienced in algorithm development process

Table 6 summarizes student responses to the following question: “What are the difficulties you have in the process of problem analysis and solution planning when you are asked to write an algorithm to solve a problem?”

| Table 6. The Difficulties in Problem Analysis, and Solution Planning for Students |
|---------------------------------------------------------------|-----------|-----------|
| Difficulties Experienced in Problem Analysis                  | I. Exp. Gr | II. Exp. Gr |
| f | % | f | % |
|---------------------------------------------------------------|-----------|-----------|
| 1. I do not have any difficulty in planning a solution for a problem. I understand a problem, and find out what operations are required. | 9 | 40.9 | 6 | 20.7 |
| 2. I understand the problem but cannot design the solution. | 8 | 36.4 | 10 | 34.5 |
| 3. When writing an algorithm, I cannot decide on the sequence of operations at the stage of problem analysis. | 2 | 9.1 | 4 | 13.8 |
| 4. Since my knowledge of mathematics is inadequate, I have much difficulty in finding out the solution in cases where complex mathematical operations are required. | 3 | 13.6 | 9 | 31.0 |
| Total                                                          | 22 | 100 | 29 | 100 |

As is clear from Table 6, concerning the difficulties experienced in problem analysis process, 40.9% of the students who were simply taught algorithms stated that they had no problems in the process; 36.4% experienced problems about mentally designing the solution; 13.60% had difficulties in finding out the solution in cases that require complex mathematical operations since their knowledge of mathematics is inadequate; and 9.1% stated that they had difficulties in determining the sequence of operations needed. As for the students who were directly taught a programming language, 34.5% stated that they had problems in mentally designing the solution; 31.8% had difficulties in finding out the solution in cases that require complex mathematical operations since their knowledge of mathematics is inadequate; 20.7% had no problems in the process; and 13.8% stated that they had difficulties in determining the sequence of operations needed. It could be concluded that the group who were simply taught algorithms had less difficulty in planning a solution for a problem and solving problems that require mathematical operations when compared to the group directly taught a programming language. On the other hand, a significant part of neither the students who were directly taught a programming language nor those who were simply taught algorithms had any difficulty in understanding a problem but both had...
difficulties about solutions. Table 7 summarises student responses to the question “What are the problems you experience when writing a solution you mentally design in the form of an algorithm?”

Table 7. Difficulties in Expressing Solutions as Algorithms according to the Students

<table>
<thead>
<tr>
<th>Difficulties Experienced in Developing Algorithms</th>
<th>I. Exp. Gr</th>
<th>II. Exp. Gr</th>
</tr>
</thead>
<tbody>
<tr>
<td>I do not have much difficulty in writing a solution I mentally design as an algorithm.</td>
<td>2</td>
<td>7.1</td>
</tr>
<tr>
<td>I cannot express the steps I mentally design in the form of an algorithm (I do not know how and where to start).</td>
<td>9</td>
<td>52.13</td>
</tr>
<tr>
<td>I experience problems with cycles.</td>
<td>6</td>
<td>21.31</td>
</tr>
<tr>
<td>I have problems with condition statements.</td>
<td>6</td>
<td>21.34</td>
</tr>
<tr>
<td>I have difficulty in identifying and using variables (particularly index variables).</td>
<td>5</td>
<td>17.90</td>
</tr>
<tr>
<td>I can easily write codes using Visual Basic but have difficulty in converting them into algorithms.</td>
<td>4</td>
<td>4.51</td>
</tr>
<tr>
<td>Total</td>
<td>28</td>
<td>100</td>
</tr>
</tbody>
</table>

As seen in Table 7 concerning the difficulties experienced in the algorithm development process, 32% of the students simply taught algorithms had difficulties in expressing the steps they mentally design in the form of algorithms; 21.4% had difficulties both in cycles and condition statements; and 17.9% had troubles in the use of variables. As for the students directly taught a programming language, 50% stated that they had difficulty in expressing the steps they mentally design in the form of algorithms; 11.5% had difficulty in both cycles and the use of variables, while 15.4% stated that they did not have any difficulty in writing codes using a programming language but have trouble in expressing these solutions in the form of algorithms. Furthermore, a very small part of both student groups (7.1% of those simply taught algorithms and 7.7% of those directly taught a programming language) stated that they did not have any difficulty in developing algorithms. Thus, a majority of both student groups arguably had problems in the process of designing algorithms. On the other hand, the students simply taught algorithms had the greatest difficulty in expressing the solutions they design in the form of algorithms, in cycles, condition statements and the use of variables, while those directly taught a programming language usually had difficulty in expressing solutions as algorithms.

Table 8 summarizes the student responses to the question “What do you think is the primary problem of first priority that you are faced with during the entire process of developing an algorithm?”

Table 8. Primary problems of first priority in the process of algorithm development for the students

<table>
<thead>
<tr>
<th>Primary Difficulties of First Priority Experienced during the Entire Process of Algorithm Development</th>
<th>I. Exp. Gr</th>
<th>II. Exp. Gr</th>
</tr>
</thead>
<tbody>
<tr>
<td>I have difficulty in determining where and how to use variables</td>
<td>4</td>
<td>18.25</td>
</tr>
<tr>
<td>I cannot be sure about how to design cycles.</td>
<td>4</td>
<td>18.25</td>
</tr>
<tr>
<td>My primary problem does exactly not know about the concept of algorithm, and where to use algorithms.</td>
<td>2</td>
<td>9.04</td>
</tr>
<tr>
<td>Failure to comprehend the logic of algorithms</td>
<td>3</td>
<td>13.64</td>
</tr>
<tr>
<td>I cannot put algorithms down on paper.</td>
<td>2</td>
<td>9.11</td>
</tr>
<tr>
<td>I have trouble in establishing links between mathematical logic and algorithmic logic.</td>
<td>1</td>
<td>4.50</td>
</tr>
<tr>
<td>Analyzing and envisioning problems.</td>
<td>4</td>
<td>18.20</td>
</tr>
<tr>
<td>Lack of experience</td>
<td>2</td>
<td>9.15</td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
<td>100</td>
</tr>
</tbody>
</table>

As is clear from Table 8, the aspects about which the students had the greatest problems in the algorithm development process include variables, cycles, problem analysis and envisioning solutions, respectively, for the group simply taught algorithms (18.2%), and problem analysis and envisioning solutions for the group directly taught a programming language (44%). On the other hand, 20% of the students in this group attribute their problems to their lack of experience. Therefore, it could be argued that more problems are experienced in variables, cycles and problem analysis by the students simply taught algorithms, and in problem analysis and envisioning solutions by those directly taught a programming language when compared to other skills.

4. Conclusion and suggestions

Students’ algorithm development skills are not at a sufficient level and the great majority of them have problems in the algorithm designing process. As frequently noted in the literature, along with students’ low learning motivation both in algorithm development and teaching programming skills, problems are usually experienced in making students acquire these skills in terms of the cognitive domain [1, 9, 32, 23, 24, 25]. If the learning process does not take into account the students’ characteristics and individual learning needs, such subjects could be highly difficult particularly for students outside the fields of computer engineering or programming. As a matter of fact, the sample group of the present study consists of pre-service computer teachers. Undoubtedly, research on different study groups with regard to the teaching of these skills may present different results. Therefore, it would be appropriate to conduct various studies as to the differences in teaching students to which various skills such as programming, critical thinking and problem solving acquired by pre-service teachers during the courses they took throughout their high school education can affect and contribute to higher education.

Both in the experiment and control groups, the students’ difficulties in the algorithm development process were less at the levels of problem analysis, and the use of mathematical operators, and condition statement, while they had more problems in expressing a solution as shortly as possible, and in the form of an algorithm as well as in the correct use of variables. As revealed by an examination of the categories concerning algorithm skills, students experience greater troubles when developing an algorithm to solve the problems they analysed. From this result it could be concluded that students at higher education levels possess basic skills such as problem analysis, even though they lack adequate introductory behavior. In other words, even though they are inadequate, the student is not aware or does not use them very often; most students arguably possess passive programming knowledge that has not yet been revealed or fully manifested.

Direct teaching of a programming language and simply teaching algorithmic logic do not influence the difficulties experienced by the students in the process of algorithm development neither in terms of basic skills nor the categories concerning these skills. As suggested by this result, although there is a belief underlined in the literature that the skills of designing and developing algorithms are often necessary to teach programming skills, it may not be indispensable as demonstrated in the present study [26, 27, 28]. To put it differently, no linear relationship exists between teaching algorithm skills and teaching programming skills. Thus, it could be argued that it is a waste of time to offer students the algorithm course as a prerequisite in teaching programming skills and to include the algorithm course in the curriculum.

According to the student opinions, the group simply taught algorithms have less difficulty in planning a problem solution and solving problems that require mathematical operations when compared to the group directly taught a programming language. Each programming language has its peculiar writing rules. On the basis of such writing rules, mathematical statements differ from standard statements used in daily life. For instance, “a+a=2” is used to increase a variable’s value in Visual Basic, whereas statement “++a” is used in C# for the same operation. Such statements are used in the standard mathematical statements while teaching algorithmic logic. Thus, a lack of understanding about the mathematical statements peculiar to programming language among the students directly taught a programming language might have resulted in greater difficulties for them in finding out solutions involving complex mathematical operations when compared to the students simply taught the algorithmic logic. The troubles experienced by mathematical statements by the students directly taught a programming language might have complicated their problem analysis process. On the other hand, a significant part of both students directly taught a programming language and those simply taught algorithmic logic did not have any difficulty in understanding the problems but had troubles about solutions. This could be attributed to underdeveloped algorithmic thinking, critical thinking, mathematical thinking or problem-solving skills on the part of students.

The greatest difficulties are experienced in expressing a design solution in the form of algorithms and the use of cycles, condition statements and variables by the students simply taught algorithmic logic, and in expressing the solution in the form of an algorithm by the students directly taught a programming language [28-31]. The students directly taught a programming language can easily see the results of their programs, and condition commands on the editor, which could have resulted in less difficulty for them when compared to the students simply taught algorithms. In fact, some of the students directly taught a programming language stated that they were able to write code pieces using Visual Basic commands but failed to express the same code piece in the form of an algorithm. This might also have led to the greater troubles for these students in expressing the solutions they design in the form of algorithms when compared to
those simply taught algorithmic logic. In fact, these students can easily follow the results of the Visual Basic codes they write using the editor, and, when necessary, use the trial-and-error method to solve the problems for which they have difficulty in designing solutions.

Acknowledgements: A part of this study is presented at ICITS2010 and IETC2011.

References