STUDENTS’ CRITICAL THINKING SKILLS IMPROVEMENT VIA ALGORITHMIC PROBLEMS SOLVING

G. Atanasova
University of Ruse (BULGARIA)

Abstract

The modern society requires creative thinking personalities with adequate thinking in rapidly changing conditions. Due to the rapid evolutionary nature of computer science, lifelong learning becomes even more important. Equipping students with critical and creative thinking skills is very important for computer science area and can make learning more effective. Both critical and creative thinking skills can be taught by offering explicit courses on such topics or by development important skills as part of the content of various courses in computer science or computer engineering programs. Teaching critical skills along with course content may be more appropriate than transferring the subject knowledge. Some topics may provide a very natural way to teach critical thinking skills.

Many strategic educational programs conclude that it is important to give a high priority in any educational system to the Improving the quality of students’ thinking. The aim of the entire Computer science and engineering degree program is to teach the involved students to think critically. Every course in computer science field demands students to clarify their thoughts sufficiently so that those thoughts to be expressed in a form that a computer can carry them out. The various programming courses comprehend different algorithms knowledge and mental model making abilities. In this paper, we describe our efforts through algorithmic problems solving on a topic from an introductory course in a typical computer science graduate program to infuse critical thinking skills into the course contents. The critical thinking analysis led us to propose a set of algorithm problems corresponding to the relevant element of the critical thinking structure. With this paper, we introduce some algorithmic tasks so that force the students in computer sciences to broaden their critical thinking skills and mental model making abilities.

Keywords: Computer Science Education, Critical Thinking, Algorithms, Learning Sequence, Problem-Solving Skills.

1 INTRODUCTION

In our conventional way of teaching computer science, emphasis is on covering the contents in ways the instructors consider appropriate. Generally no or very little explicit emphasis is given to the thinking skills. Critical thinking skills place emphasis on making thinking process explicit. Combining the two approaches have the potential to enhance the learning abilities of the students. The approach can teach thinking about thinking while thinking in order to make thinking better [1].

There is a significant amount of literature available ([2],[3],[4],[5]) on infusing critical thinking skills into course content in medicine, nursing, psychology, engineering and pure sciences like physics and chemistry. Very little literature is available on such efforts in computer science [6].

Algorithms are recipes for solving a problem. They are fundamental to computer science and software engineering. Algorithms are the formal foundation of computer programming but also exist independently of computers as systematic problem-solving procedures. Knowledge on a high level is compulsory foundation of computer science education and acquiring algorithms’ ability increases students’ critical thinking skills.

Section 2 of the paper discusses critical thinking. Some algorithms’ feathers and its influence on students’ thinking skills and awareness are discussed in section 3. How the design of instruction perspectives of the algorithmic problems impacts on students’ knowledge acquiring and some groups of learning problems are described in section 4. Section 5 presents for example three problems for students’ critical thinking skills improvement. Conclusions and future intentions of the author are summarized in section 6.
2 CRITICAL THINKING

We begin this section with a quotation from an article by Facione ([6], [7]) to highlight the importance of critical thinking skills.

We understand critical thinking to be purposeful, self-regulatory judgment which results in interpretation, analysis, evaluation, and inference, as well as explanation of the evidential, conceptual, methodological, criteriological, or contextual considerations upon which that judgment is based. Critical thinking is essential as a tool of inquiry. As such, critical thinking is a liberating force in education and a powerful resource in one’s personal and civic life. While not synonymous with good thinking, this kind of thinking is a pervasive and self-rectifying human phenomenon. The ideal critical thinker is habitually inquisitive, well-informed, trustful of reason, open-minded, flexible, fair-minded in evaluation, honest in facing personal biases, prudent in making judgments, willing to reconsider, clear about issues, orderly in complex matters, diligent in seeking relevant information, reasonable in the selection of criteria, focused in inquiry, and persistent in seeking results which are as precise as the subject and the circumstances of inquiry permit. Thus, educating good critical thinkers means working toward this ideal.

Recently there has been an increasing emphasis on lifelong learning and one is expected to learn new things even long after finishing formal education. The mere transfer of knowledge from teacher to students is considered inadequate. To facilitate learning outside and after formal education, many good thinking skills such as critical and creative thinking should form part of classroom teaching [8]. The best way to achieve this is to infuse these skills into regular course content [1]. This exercise also gives an opportunity to use active learning techniques in the class room. In this paper, we describe some of our efforts in infusing critical thinking skills into a course on introduction to computer programming algorithms.

The purpose of specifically teaching critical thinking in computer science or any other discipline is to improve the thinking skills of students and thus better prepare them to succeed in the world. But, one may ask, don't we automatically teach critical thinking when we teach our subjects, especially engineering disciplines which need rational thinking in optimizing the design while balancing many (often conflicting) requirements through judicious trade-offs? The answer to this question is often ‘no’ for the following reasons. All education consists of transmitting to student two different things:

1. the subject matter or discipline content of the course ("what to think")
2. the correct way to understand and evaluate this subject matter ("how to think").

We do an excellent job of transmitting the content of our respective academic disciplines, but we often fail to teach students how to think effectively about the subject matter, that is, how to properly understand and evaluate it. This second ability is termed critical thinking. Due to various constraints (time being the main constraint), majority of us approach content, not as a mode of thinking or as a system of thought, but rather as a sequence of stuff to be routinely covered and committed to memory. When content is approached in this lower order way, there is no basis for intellectual growth as there are no deep structures of knowledge formed and no basis for long term grasp and control. Critical thinking, in contrast, approaches all content explicitly as thinking and weaves new thinking into old. It is thinking about thinking while thinking in order to make thinking better [1].

As we mentioned in section 1 there is a significant amount of literature available on infusing critical thinking skills into course content in medicine, nursing, psychology, engineering and pure sciences like physics and chemistry, very little literature is available on such efforts in computer science.

While many university administrations encourage introduction of good thinking skills and other interpersonal skills, the initial reaction of faculty is an apprehension that introduction of these (perceived to be) extra skills eat into their classroom time. On the contrary, our experience shows that explicit introduction of these important skills stimulate the students thinking and enhance their learning skills.

3 CRITICAL THINKING AND ALGORITHMS

Algorithms are recipes for solving a problem. They are fundamental to computer science and software engineering. Algorithms are the formal foundation of computer programming but also exist independently of computers as systematic problem-solving procedures. The introductory programming courses must introduce algorithmics, the study of algorithms. It is not about programming and coding
but rather about understanding and analyzing algorithms and about algorithmic problem-solving, i.e. the design of systematic problem-solving procedures. Students develop algorithms to solve a wide variety of different problems and in this way promote its own critical thinking skills. There is an implicit perception that algorithmic problem-solving skills are acquired while learning to program. Programming assignments in the introductory programming course exercise the use of programming structures, but they also require the development of algorithms. Experience shows that while students handle well the basic algorithmic constructions, they often demonstrate poor algorithmic problem-solving skills; difficulties arise with regard to problem decomposition, developing an efficient solution, or using previously seen solutions even for elementary problems ([9], [10], [11], [12]). One of the main obstacles in learning problem solving while learning programming is the tendency to put a lot of attention on syntax rules while missing the abstract ideas. We believe that targeting learning abstract ideas and algorithmic problem-solving principles within the introductory courses may enhance the development of students' skills in computer science and support students' learning.

Problems as educational units afford rich opportunities for teaching, learning and developing personable skills. The algorithmic problems in introductory computer science courses are directed to basic algorithm development and knowledge skills formulation. The aimed result is professional competence which range over: concepts acquiring, ideas, facts, principles, methods, possible operations and inner relations, basic terms and symbols. Algorithmic problems have significant influence on students' thinking skills development and their intellectual growth. Each step of an algorithm problem solving requires appropriate critical thinking activities (table 1). The problem solving process involves students' thinking abilities and formulates their personality awareness. The problems may be directed at different makings development such as working, collaborating, self-adjusting and etc. The problem solving process exerts a significant influence on students' education.

<table>
<thead>
<tr>
<th>Algorithmic Problem-Solving Skills</th>
<th>Critical Thinking Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithmic Problem’s comprehension</strong></td>
<td>Reformulation of the algorithmic problem in terms of inputs, outputs, data processing, assumptions, constraints and required results.</td>
</tr>
<tr>
<td><strong>Problem’s decomposition</strong></td>
<td>Identifying, naming and listing steps and subtasks.</td>
</tr>
<tr>
<td><strong>Analogical reasoning</strong></td>
<td>Identifying similarities between other early formulated problems and ready algorithmic decisions. Raising awareness of common mistakes caused by referencing to unsuitable problems.</td>
</tr>
<tr>
<td><strong>Generalization and abstraction</strong></td>
<td>Mental model development. Extracting prototypes of problems from analogical problems in different contexts.</td>
</tr>
<tr>
<td><strong>Identifying problem’s prototype</strong></td>
<td>Systematic and well-structured introduction of algorithmic patterns. Problems’ categorization.</td>
</tr>
<tr>
<td><strong>Algorithmic Problem’s structure identification</strong></td>
<td>Identifying the relation between subtasks. Schematizing a problem’s structure, using flow chart.</td>
</tr>
<tr>
<td><strong>Evaluation and appreciation of efficiency and elegance</strong></td>
<td>Comparing solutions with regard to efficiency and elegancy.</td>
</tr>
<tr>
<td><strong>Verbalization of ideas</strong></td>
<td>Formulation of a precise idea, differentiating between an algorithm and its implementation.</td>
</tr>
</tbody>
</table>
4 DESIGN OF INSTRUCTION PERSPECTIVES OF THE ALGORITHMIC PROBLEMS

We stated that the problems-solving contributed to the construction of students’ knowledge in a meaningful way. The organization of knowledge helps in categorizing a problem according to its prototype. One aspect of problems’ design that students mostly benefited from is grouping problems according to their prototypes. The way the problems were organized has a definite influence on learning problem-solving and critical thinking skills. Dividing the problems into levels and seeing examples for each type of problem helps in pulling out from the toolbox the right way to solve a new problem. It is important to exercise the same type of problem several times, so a student better understands the main common idea in those problems. Otherwise, students cannot make links between them and cannot assimilate what is taught. The more students see examples and ideas, the more they are able to connect between similar problems. When solving the same problem several times, each time phrased in a different manner, it develops the skill of comparison between problems. Another aspect of problems’ design that students benefited from is the gradual increase in problem complexity which may enhance the skill of problem structure identification. Learning types of small problems helps in visualizing a big problem. Often problems contain smaller problems similar to those that students had learned previously. This method showed how important it is to decompose a task into subtasks, and to handle each subtask separately.

The problems would be assiduously and purposefully selected in systematic sequence. We may differentiate the following groups of problems [13]:

- **Problems for new learning knowledge introduction.** This kind of problems bringing in the new training units and ensure positive learning motivation. The introduction of difficult and unknown problems raises students’ motivation and learning willingness. The problems for new knowledge introduction would not be solved by acquired skills. These problems have the essence of the new learning ideas.

- **Problems for knowledge reassertion.** This kind of problems contributes to aware and permanent learning units’ adoption. By solving problems for knowledge reassertion the students concentrate its attention on the essence of the matter. It is important to keep the relevant proportion between theory and practice by formulating the problems on this stage. These problems have to offer the introduced new learning units reenactment only.

- **Problems for knowledge application.** We can consider that knowledge is acquired if students know not only the definition of the units, methods, algorithms and etc. We consider that students have acquired steady skills if they can apply the acquired knowledge. The problems are picked out. They must include all the various applications of the taught material. On the other hand the knowledge application may be realized on different levels – low and high complexity applications and applications with non-standard solutions. The problems that require non-standard solutions build students’ professional awareness and form their critical thinking skills.

5 PROBLEMS FOR STUDENTS’ CRITICAL THINKING SKILLS IMPROVEMENT

The most significant teaching element is the appropriate selection of the problems’ combination. It predestines the learning process progress and students’ skills development.

For acquiring the basic algorithmic conceptions we need to follow a systematic way and therefore we defined a sequence of learning topics (table 2). The topics are consistent with the above described problems’ groups.
The relationship between the topic sequence and the described above three kinds of problems we demonstrate with the following problems, implemented in C/C++.

Problem #1 presents an example for new knowledge introduction related to the topic 8.

**Problem #1:** Give an algorithm for the sum and number of the digits of a natural number computation.

**Problem #1 decision in C++:**
```cpp
#include<iostream.h>
int main()
{
    int c, a, br=0, sum=0;
    cin>>a; //input a natural number
    while(a>0){
        //while in the input number there are digits
        c=a%10; //take the last digit
        br++; //increment the number of digits
        sum+=c; //sum the new sum of the digits
        a/=10; //remove the processed digit
    }
    cout<<sum<<endl; //output the sum of the digits
    cout<<br<<endl; //output the number of the digits
}
```

On topic 11 we consider that the Problem #2 is suitable for knowledge reassertion.

**Problem #2:** Give an algorithm for some tree-digits natural number input and output the largest natural numbers that may be composed from the digits for each input.
The students can use the taught algorithm (problem #1) on topic 8 for that problem decision. They learn to identify, apply and exercise early mastered knowledge by this mean. The trainees have to define the common elements and the differences in the discussed algorithmic problems. The analysing and summarization of taught algorithms in a new one is the process that form and improve students' critical thinking skills.

Problem #2 decision in C++:

```c++
#include <iostream>

int main()
{
    int k, a, b, c, max, n;
    //enter the number of inputs
    cin>>n;
    for(i=0;i<n;i++)
    { //input the i-th number
        cin>>k;
        //calculate the last digit
        a=k%10;
        //calculate the second digit
        b=k/10%10;
        //calculate the first digit
        c=k/100;
        /* order the digits so that ‘a’ to contain the minimum digit and ‘c’ - maximum digit */
        if (b<a) swap(a,b);
        if (c<a) swap(a,c);
        if (c<b) swap(c,b);
        /* make the maximum number that is possible to be composed from the digits */
        max=c*100+b*10+a;
        cout<< max <<endl;
    }
}
```

The Problem #3 is related to topic 11 and it is suitable for knowledge application.

Problem #3: In the numerology from a person’s birth date is calculated the so called “personal number” that shows the person’s makings and destiny. This number is calculated as sum of all the birth date digits and so on while the sum becomes one-digit number.

Example: 29.09.1989 -> 2+9+0+9+1+9+8+9 = 47 -> 4+7 = 11 -> 1+1 = 2

Personal number of the birth date 26.09.1989 is 2.

The problem is to suggest an algorithm that computes the personal digit from a given birth date.

This problem uses the knowledge acquired from the taught in Problem #1and reasserted in Problem #2 algorithm for the sum and number of the digits of a natural number computation. Problem #3 grades over the students’ critical thinking skill as they have to search the right way of non-conventional knowledge application. We must to point out that we educate student for real life and the real problems need good critical thinkers with high knowledge application awareness. So problem #3 and similarities of it are the end aim of the computer science bachelor course education.

Problem #3 decision in C:

```c
#include <stdio.h>
#include <string.h>

int numero(char *data)
/* Here we use a subalgorithm for the sum of the digits calculation. The birth date, called in that case 'data' is input parameter */
{
    int i, sum = 0, n = strlen(data);
    //inner variables initialization
    for (i=0;i<n;i++)
```
//for each symbol of the input
if (data[i]!='.')
//if the symbol is differ from .
    sum+=(data[i]-'0');
//increase the sum with the current digit
while (sum>10)
//while the sum is not single-digit
    sum = (sum /10) + (sum%10);
//separate the last digit and increase the sum
return sum;
}

int main(){
    char data[32];
    gets(data);
    //input a person’s birth date
    printf("%s-%d\n",data,numero(data));
    /* invoke the subalgorithm for the person’s number calculation */
}

6 CONCLUSIONS

Critical thinking abilities are significant point of academic maturity and a trademark of a well-educated person. Our experience as assistants in computer science department, where critical thinking is an explicitly desired outcome, has led us to carefully explore the relationship between computer science and critical thinking skills.

Critical thinking skills can be cultivated in many ways besides algorithmic exercises, and at many points in the curriculum besides the introductory programming courses. Algorithmic problem solving suggests opportunities for basic algorithm knowledge acquire and parallel critical thinking skills development. Of course, there is needed a systematic approach with proper chosen problems. Therefore we developed and suggested a learning topic sequence. In this paper we presented three kinds of problems for solving and a concrete problem of each kind related to some of topics.

The connections between algorithm learning and critical thinking are rich exploring furthers. How might we assess the effect of an increased emphasis on critical thinking skills? How can we develop algorithms and the learning sequence that promote critical thinking? What the students’ opinion is? We look forward to further work in this area.

REFERENCES


