DEVELOPMENT OF VARIATIONAL THINKING SKILLS IN PROGRAMMING TEACHING

Sava Grozdev, Todorka Terzieva

Abstract. *The paper presents an example of methodological approach to the development of variational thinking skills in teaching programming. Various ways in solving a given task are implemented for the purpose. One of the forms, through which the variational thinking is manifested, is related to trail practical actions. In the process of comprehension of the properties thus acquired, students are doing their own (correct or incorrect) conclusions for other, hidden properties and at the same time they discover possibilities for new ways of action and acquiring of new effects. The variability and the generalizing function of thinking are in a close interrelation, and their interaction to a great extend determines the dynamics of the cognitive activity of the student.*

Keywords: programming, critical thinking, variational thinking
2010 Mathematics Subject Classifications: 97Q60, 97Q30

The new and fast-changing content of Informatics teaching requires the development of methodologies which could provide not only reproduction of large volume of knowledge, but most of all methodologies, which could form and develop competences that would allow students to master the knowledge actively and to create skills for acquiring new knowledge independently and for comprehending it critically.

We shall emphasize a number of problems, which students encounter in teaching programming:

- difficulties in pointing out more than one way of solving a task;
- difficulties in analyzing their own solution or other suggested solution of a task;
- lack of skills to discover independently and rectify admitted mistakes;
- difficulties in finding the optimal solution.

In the studies of numerous authors there are different approaches to increase the efficiency of programming teaching [5,7,10] and information technologies [2,4,11] at all levels of the teaching process. The development and the implementation of a teaching process focusing on the development of programming skills through the use of different paradigms and methods from the modern programming languages are discussed in [1, 8, 9].

In order to overcome the problems arising at all stages of training, it is necessary to maintain and develop striving for unconventional thinking and

creative approach to each task [3]. Independence and critical thinking are essential for the formation of own style of learning activity, excluding templates, ready understandings and actions.

The term 'critical thinking' is long known from the works of psychologists like Piaget, Poya, Bruner, Vygotsky. Researchers like Bono, Dewey, Lipman, Halpern, Cluster consider the problem of forming of critical thinking. Criticality of thinking is necessary at all levels of the problem education process such as analysis, comparison, generalization and negation. Without a critical evaluation of a task it is impossible to formulate hypotheses. Moreover, critical thinking includes proving of the hypothesis or its disproval. Next are the stages of confirming the conclusions, search of the most rational solution of the problem, etc. Halpern notices that critical thinking does not mean negative reasoning and criticism, but a reasonable discussion of the variety of solutions of the problem aiming at finding a well-grounded solution [6].

One of the most important qualities of critical thinking is the so-called variational thinking. The specific functions of the variational thinking consist in the discovery of hidden properties, connections and correlations in the actual reality by transforming this reality into conditions, where the acquiring of new facts only through conceptual and visual-figurative thinking becomes difficult or impossible. One of the forms, through which the variational thinking is manifested, is related to trail practical actions. The actions alone are not directed at acquiring an immediate practical effect. Their function is rather cognitive and they create medium for studying a specific object or analyzing a specific situation, e.g. a specific task. Despite the huge amount of research on variational thinking, there are numerous problems which are studied insufficiently. Such a problem is the relationship between the variational influence on a specific task and the understanding of the achieved results. By using different means of influence, the student studies the specific tasks comprehensively. Each task presents specific information and therefore it is knowledge. The more versatile the influences are the bigger is the probability that a specific task is studied in the variety of its peculiarities. In the comprehension process of the properties thus acquired, the students are doing their own (correct or incorrect) conclusions for other, hidden properties and at the same time they discover possibilities for new ways of action and of acquiring new effects.

By variability we understand the skill of finding new and variational means for an actual transformation of the specific task. The trail actions have an important role in this process, aiming at the discovery of hidden internal connections. In this aspect, the variability is a significant property of the trail activity. The latter is achieved through external trail actions. The generalizing function of thinking is in the skill to comprehend the data obtained in the course of the actual transformations, to make conclusions about hidden properties, connections and correlations. Certainly, the availability of this skill allows the functioning of all thinking processes, and not only of the variational thinking. Variability also means variety of the trail actions. The variability and the generalizing function of thinking

are in close interrelation, and their interaction determines the dynamics of the cognitive activity of the student to a great extend.

Exactly considering the goal – training of critical thinking, the authors emphasize the development of skills for variational thinking, which is realized naturally through different ways and approaches in solving a specific task in programming teaching of first-year students of Informatics.

Students study successively the methods and means of the structured programming - basic control structures, abstraction of data (types and data structures and their presentation), compilation and analysis of algorithms for searching, sorting, recursion and iteration. Experience show that students have particular difficulties in studying the topic of strings and algorithms for their processing. So far they have knowledge on the basic control structures, realization of an abstract structure array, sorting and search algorithms, type pointer, dynamic variables and data processing by grouping in programming languages, subprograms - functions. The difficulty arises in the transmission of accumulated knowledge in the new situation, taking into account its special features. We offered the students a task for processing the string with an additional condition for the realization of more than one way of solution.

The task: Create a program, which counts the number of each character encountering in a keyboard input string.

The results analysis shows that most of the students reach declarative level of learning for basic concepts and algorithms. They prefer to avoid the use of functions, pointers and dynamic variables. According to the data received from the decisions, 67% of students only declared and initialized the string, 20% offer only one solution and 13% - more than one solution. This directs us to investigate the correlation between variational influence on a specific task and the understanding of the achieved results.

Different knowledge and approaches could be applied while choosing a solution of this task regarding:

- basic control structures in use;
- implementing of different sorting algorithms;
- use of standard and user functions;
- basic operations and standard functions for work with strings;
- realization on type pointer and dynamic variables.

The solution should be written in C++.

Initially, an analysis on the task is accomplished in the form of discussion. Emphasis is placed on the formalization and separation of the tasks into subtasks and formulation of major goals. The students investigate different approaches for problem solving with goal to searching for optimal solution.

Option 1. Each consecutive symbol will be checked whether it is included in the symbols input before. If the symbol is already found (f=1), this means that it has been counted and the consecutive symbol is skipped. If the symbol is not there (f=0), this means that the current symbol is not encountered yet and the counting starts from the respective position to the end of the string.

```
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;
void main()
{
char Str[100];
int S,i,j,f;
cout<<"Enter niz: ";
cin.getline(Str, 100);
for (i=0;i<strlen(Str);i++)
{ f=0;
  for (j=0;j<=i-1;j++)
   if (Str[i]==Str[j]) f=1;
   if (f==0) { S=0;
               for (j=0;j<strlen(Str);j++)
               if (Str[i]==Str[j])  S=S+1;
               cout<<Str[i]<<" - "<< S<<endl;
              }
} system("pause");
}
```

Option 2. The main problem of the task is to avoid recounting of already counted symbols. Therefore, the input symbols could be firstly sorted lexicographically and then counted. The students are given the possibility to use different sorting algorithms. In the sorted array it should be noticed the moment when the next compared symbol is different from the current one – at this moment the current symbol and its number are deduced. Then we move to the next symbol, etc.

```
void main()
{
char Str[300],ch;
int S,i,j,len;
cout<<"Enter niz: ";
cin.getline(Str,300);
len=strlen(Str);
for (i=0;i<len;i++)
for (j=0;j<len;j++)
   if (Str[i]<Str[j])
   {
       ch=Str[j];
       Str[j]=Str[i];
       Str[i]=ch;
   }
   ch=Str[0];
   S=0;
```

```
   for (i=0;i<len;i++)
   if (ch==Str[i]) S=S+1;
   else {
         cout<<ch<<" - "<< S<<endl;
         ch=Str[i];
         S=1;
        }
cout<<ch<<" - "<< S<<endl;
cout<<"Sortiran niz:"<<Str<<endl;
}
```

Option 3. The next goal, which is set is a realization of a second way by using standard function for sorting by the Standard Template Library: Algorithms and user-defined function for counting each character. Additional condition is defining the string using pointers. We will present a solution by using a standard sorting function. To this end it is necessary to add a directive for the use of the sorting function of C++ Standard Template Library.

```
#include <algorithm>
using namespace std;
void countchar(char *str,int len)
{ char ch=str[0];
  int S=0;
  for (int i=0;i<len;i++)
   if (ch==str[i]) S=S+1;
   else
     {  cout<<ch<<" - "<< S<<endl;
        ch=str[i];
        S=1;     }
   cout<<ch<<" - "<< S<<endl;
}
void main ()
{
int N;
cout<<"Enter string length:"; cin>>N;
cin.ignore();
char *str=new char[N];
cout<<"Enter string: ";
cin.getline(str,N);
int strLength=strlen(str);
sort(str, str+strLength);
countchar(str,strLength);
cout<<"Sortiran niz:"<<str<<endl;
}
```

Option 4. Add a condition to transfer all upper cases to lower cases and then count the different symbols. Create an user function for transforming the upper - lower case.

```
const int MAXLEN=200;
void toLowerCase(char* str)
{
  int strLength=strlen(str);
  for(int i=0; i<strLength; i++)
      str[i]=tolower(str[i]);
}
```

```
void main ()
{
char str[MAXLEN];
int count, i, j;
cout<<"Enter string: ";
cin.getline(str,MAXLEN);
toLowerCase(str);
int strLength=strlen(str);
for(i=0; i<strLength; i++)
 {   if(str[i])
       { count=1;
         for(j=i+1; j<strLength; j++)
             if(str[i]==str[j])
              {  count++;
                 str[j]=0; }
          cout<<"\""<<str[i]<<"\" "<<count<<"\n";}
}}
```

Option 5. This is a more unconventional approach. Use a variable which contains the ASCII code of each symbol. Each ordered symbol of the input string is compared with the respective code until match.

```
void main()
{
char Str[200];
int S,i,j;
cout<<"enter niz: ";
cin.getline(Str,200);
for (i=0; i<=255;i++)
 { S=0;
   for (j=0; j<strlen(Str);j++)
     if ((char)i == Str[j]) S=S+1;
   if (S!=0) cout<<(char)i<<" - "<< S <<endl;
}}
```

In our opinion and according to the feedback received from the students, there are many benefits gained from this methodological approach. Results from the study show, that the interest of students is stimulated; their attention and motivation are promoted for the implementation of accumulated theoretical knowledge for use in new context. Thus develop skills for finding different solutions of the specific problem and also build a critical attitude to the respective problematic situation. This creates conditions for the development of skills to read (implement) a finished algorithm, which is not less important than creating the algorithm itself. On the other hand, the unconventional decisions help to avoid automatic memorising and implementing of already solved tasks. The variational-critical position of the student and the teacher directs to searching and finding of more effective solutions of the task and reaching the optimal solution. In this way weaknesses and omissions in the algorithm are analysed, which could be used for perfecting of the algorithm.

According to our opinion, the purposeful training of critical thinking through the development of skills for variational thinking would allow students to overcome the reproductive level of comprehension of the taught material, thus

motivating a deeper penetration into the essence of the problems and leading to unconventional approaches to the task solution. The generalizing function of thinking is in the skill of comprehending the obtained data in the course of the actual transformations, of making conclusions about hidden properties, connections and correlations. This skill allows the functioning of all thinking processes, and not only of the variational thinking.

References

[1] Aneva, S., The Role of Basic Problems in Learning of Event-Driven Programming with Visual C# Environment in High School, Proceedings of the Anniversary International Conference, Synergetics and Reflection in Mathematics education, September 10-12, (2010), Bachinovo, Bulgaria, pp. 353-363 (in Bulgarian).

[2] Angelova, E., A. Rahnev, Boosting Teaching and Learning Efficiency in Training Teachers of Information Technology, Scientific Works, Plovdiv University, vol. 36, book 3, (2009), Mathematics, pp. 5-18.

[3] Chehlarova, T., Approaches for determination of the intelligence and its structure, Scientific Works, Plovdiv University, vol. 46, b.2, (2009), Methodology of Teaching, pp. 27-35 (in Bulgarian).

[4] Dicheva, T., Model for a System of Questions for Module “Databases” in Education in Information Technologies, Proceedings of the Anniversary International Conference, Synergetics and Reflection in Mathematics education, September 10-12, (2010), Bachinovo, Bulgaria., pp. 389-397 (in Bulgarian).

[5] Dureva, D., Problems of the methodology of training in Informatics and Information Technology, Publishing House "N. Rilski", Blagoevgrad, (2003) (in Bulgarian).

[6] Halpern, D., Thought and knowledge: an introduction to critical thinking, Hillsdale, New Jersy, Mahwah, (1996).

[7] Iliev, A., N. Kyurkchiev, Nontrivial Methods in Numerical Analysis. Selected Topics in Numerical Analysis, LAP LAMBERT Academic Publishing GmbH & Co. KG, (2010).

[8] Malinova, A., Developing Programming skills using “Mathematica”, Proceedings of the Anniversary International Conference, Synergetics and Reflection in Mathematics education, September 10-12, (2010), Bachinovo, Bulgaria, pp. 425-435 (in Bulgarian).

[9] McGill, T., Volet, A conceptual framework for analyzing students’ knowledge of programming, Journal of research on Computing in Education, (1997), 3, 276.

[10] Momcheva-Gardeva, G., S. Kapralov, Development of Combinatorial Skills for University Students in Computer Science, Proc. of the 6th Mediterranean Conference on Mathematics Education, Plovdiv, Bulgaria, (2009), pp. 293-304.

[11] Shotlekov, I., A. Rahnev, Evaluating the Quality of Student Web Design Projects, Mathematics and Education in Mathematics, Sofia, (2010), pp. 227-236.

Sava Grozdev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: sava.grozdev@gmail.com

Todorka Terzieva
Faculty of Mathematics and Informatics
Plovdiv University “Paisiy Hilendarski”
236, Bulgaria Blvd.
4003 Plovdiv, Bulgaria
e-mail: dora@uni-plovdiv.bg