Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

Problem Solving and Interrelation of Concepts in Teaching Algorithmic Thinking and Programming

Zsanett Szabó

Eötvös Loránd University
szabo.zsanett@inf.elte.hu

Abstract

Developing problem-solving thinking should not be the task of only mathematics. It should be the task also of informatics. Students' problem-solving abilities can be greatly enhanced by solving consciously structured, interrelated programming tasks. To create a well-structured series of tasks it is necessary, that we understand and become aware of our own system of concepts. As a teacher, we need to be aware of the concepts that are really important and how they relate to each other. Our aim is to help our students to develop connections and correspondences within their concepts that will facilitate the integration of new concepts into their concept map. In this paper we deal with this concept map and the issues that arise with it.

Keywords: teaching informatics, teaching programming, problem solving, algorithmic thinking, concept map, concept system, beginning of teaching programming

MSC: 68

1. Introduction

William James and György Pólya also said [7], that problem solving is one of the most typical and most unique human activities. Therefore one of the most important task of education is to develop students' problem solving thinking. It is one of the keystones of teaching mathematics. According to Pólya, the main task of teaching mathematics in secondary school is to emphasize the systematic work of problem solving. Nowadays systematic problem-solving should play an important

role also in previous years, not only in high school and not just in mathematics, but also in other subjects such as computer science.

2. Problem solving thinking with programming

According to Pólya, problem-solving thinking is in fact purposeful thinking, the search for a means to achieve a set goal [7]. We can develop this purposeful thinking, the problem-solving ability of students through programming tasks in a similar way to mathematical tasks. When we teach problem solving thinking in informatics we teach attitudes and algorithmic thinking.

2.1. Problem solving like in Pólya's model

The steps of Pólya's problem-solving model [8] that is basically designed for solving mathematical problems, is also well applicable with a little addition and modification for solving programming problems. Steps of Polya's model are the following:

P1. Understand the problem
P2. Devise a plan
P3. Carry out the plan
P4. Look back

Problem solving in mathematics is very similar to algorithmic program solving. As proof of this we show one of the algorithm problem solving strategies [10] which is very similar to Pólya's problem solving model. Steps of this algorithm problem solving strategy are the following:

S1. Analyse the problem
S2. Restate the problem
S3. Write out examples (input-output)
S4. Break the problem into its component parts
S5. Make an algorithm
S6. Step through your example data with algorithm
S7. Code it up
S8. Make tests
S9. Refactor

It would be easy to compare these steps with the steps of Pólya's problem solving model. Their similarity can be seen very well if we compare the helping questions for each step that guide us through the problem-solving process.

2.2. Problem solving like in other models

Of course, not only Pólya, but also others have developed problem-solving methods and problem-solving models. György Kontra gave a detailed review of these [4]. In these models the same that the mathematical problem solving and algorithmic

problem solving can be parallelized in each of them in the same way as Pólya's model.

At the international level there are studies about the similarity and difference between algorithmic and mathematical problem-solving which have been previously published [2, 3, 10].

2.3. Difference between mathematical and algorithmic problem solving

Perhaps the most striking difference is that while in mathematics it is our job to perform calculations, in the case of algorithmic problem solving the machine does the calculations for us. The advantage of this is that we can solve complex tasks that require a lot of computing. However, compared to solving mathematical problems, we have a harder task with algorithmic problem solving, since after solving the problem we have to write out the solution for the users with correct syntax [6, 9, 11]. Moreover, in algorithmic problem solving, data collection and data conversion to a suitable form for the algorithm often precedes the actual problem solving.

2.4. Importance of teaching problem solving thinking

Problem solving methods used in programming are often used in real life. Knowing these problem solving methods and applying them to small problems will help us to have a toolbox that we can work on when we have to solve a bigger problem. It is similar to the reason why we learn and teach mathematics, but it focuses on other types of problem solving in mathematics than in programming. By programming, we can solve problems that we would not be able to calculate manually.

Designing a problem solving process, decomposing problems into sub-problems, applying small problems to larger cases, recursion, thinking backwards are all very important techniques that can we easily apply in our everyday practice. The easiest way to illustrate these techniques is to illustrate them through programming.

3. Effective transfer of knowledge

For teaching programming and problem-solving thinking correctly and efficiently, and for forming up of the suitable concept structure it is necessary to think over the concepts and knowledge of the topic.

György Pólya used the terminology of guided discovery to build the desired mathematical concept structure. He worked with consciously, purposefully selected and structured sets of exercises to help students to reach the required concepts and contexts [1, 8]. Because of the similarity of mathematical and programming problem solving, the teaching of algorithmic problem solving and programming could be made more efficient with such consciously structured sets of tasks. The

set of tasks which are created in this way could greatly help teachers and students to develop the basic concepts of algorithms and programming.

To create well-structured series of tasks it is necessary, that we understand and become aware of our own system of concepts. As a teacher, we need to be aware of the concepts that are really important and how they relate to each other. Our aim is to help our students to develop connections and correspondences within their concepts that will facilitate the integration of new concepts into their concept map.

3.1. Short overview of the integration of new concepts

One of the basic assumptions of cognitive psychology is that students largely "construct" new knowledge themselves. They not only add new information to their existing knowledge repository, but they also link new knowledge to their existing knowledge [1]. They integrate new knowledge into their existing knowledge structures by forming new relations between the structures.

From this point of view, informatics and programming are fortunate, because we can make some relations with programming tasks to many other prior knowledge. "In the Netherlands, informatics has been defined as a new generation discipline, because it is linked with Mathematics, Physics, Engineering, Linguistics, Philosophy, Psychology, Economy, Business, and Social Science in general" [9, 5]. In this way, programming knowledge can integrate into students' existing knowledge structures with many, many points of contact, if we consciously pay attention to those points of contact.

3.2. Development of the concept structure

To understand our own concept structure, we first need to gather concepts and then systematize them. When we collect the concepts, we must realize that the topic of making algorithms and programming uses many concepts, but in many cases the relations between the concepts are not definite. There is not, or only very difficult to determine the "correct" order of the teaching of the concepts.

Relationships between concepts, which appear below, do not represent the order in which they should be taught. They refer only to relationships between individual concepts. Because the "correct" order of teaching these concepts depends on many factors, and because the concepts are interrelated at many points to each other, so we present only a "general situation". Thus, it is possible that a connection is not marked.

The whole concept structure is quite complex, so we will first illustrate a simplified version of the whole structure and then its smaller units. After that we show the complete concept structure by denoting the relationships between the smaller units.

A simplified version of the concept structure is shown in Figure 1. Here, the closely related terms are denoted by a collective name.

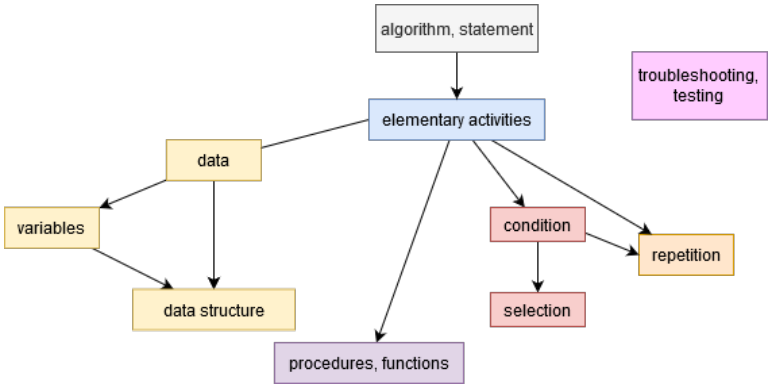

Figure 1: Simplified version of the concept structure

3.3. Initial concepts: algorithm, instruction

At the beginning of the topic making algorithms and programming, we almost always have to use concepts of instruction, command, algorithm, and some related concepts. They are illustrated in Figure 2.

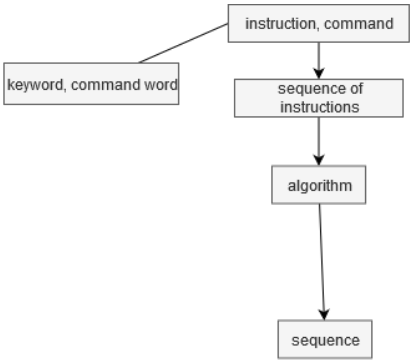

Figure 2: Initial concepts

The concept of sequence is a good example of the fact that not all concepts are named during the teaching process and thus in the students' own conceptual structure. Most programming students probably do not hear this concept, but they still need to know what it means.

3.4. Data, variables, elementary activities

When we solve algorithmic and programming tasks we will soon need to work with data and perform operations with that. To store data we have to use different types of variables and their declaration. Related to this we use input and output

operations. These concepts are interconnected at several points, as illustrated in Figure 3.

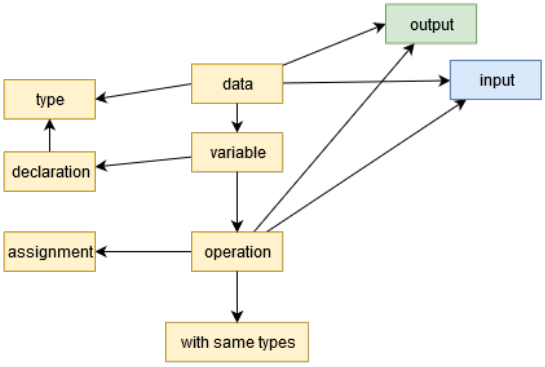

Figure 3: Concepts of data, variables and elementary activities

If we need to work with larger amounts or more complex data, we also need to use different data structures (Figure 4). In relation to these concepts the depth, the environment and the age of group in which these are taught is particularly important. Because it depends on these factors which data structures we teach and which we do not.

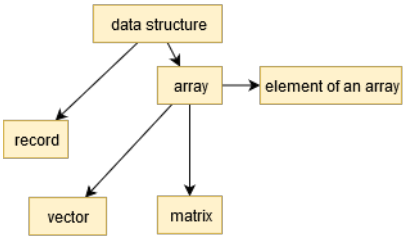

Figure 4: Concepts of data structures

3.5. Condition, selection, repetition

One of the most important parts of the concept structure is the concepts related to conditions, selection and repetition (Figure 5). Understanding these is a particularly important and indispensable part of learning and teaching this topic, because algorithms and programs are built mainly on these control structures.

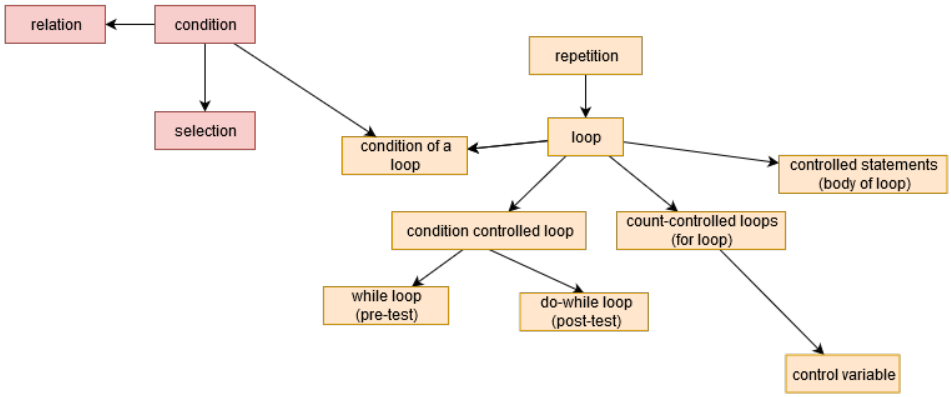

Figure 5: Concepts related to conditions, selection and repetition

3.6. Troubleshooting and testing

We also have to deal with troubleshooting and testing when we would like to check and run our algorithms and programs. Linking concepts (Figure 6) and debugging methods are highly dependent on the programming environment and the depth of teaching the topic. Nevertheless a minimal level of troubleshooting is required regardless of the environment. Debugging cannot really be linked to previous parts of the concept structure, because debugging and testing may be required at all levels of making algorithms and programming.

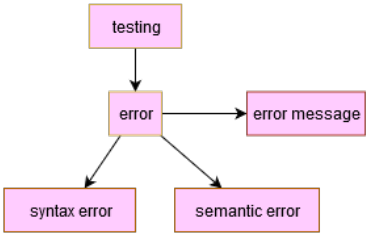

Figure 6: Concepts related to troubleshooting and testing

3.7. The whole concept structure

By combining the above mentioned concepts and the smaller parts of the concept structure, we can create the "complete" conceptual structure of the topic of algorithmic programming (Figure 7).

It contains a few concepts that have not yet been discussed. These concepts have been left out of the discussion of smaller sections because they are not so closely related to them, or because they are not necessarily discussed when we

teach the basics of the topic making algorithm and programming, or because they are not present in all frequently taught programming environments. Examples of this are operations with different types of variables, global and local variables, and recursion.

The concept of control structures also appears here for the first time. This can be omitted from the learning process because it is not important to know the concept of control structures but the goal is to make the students aware of the concept and usage of control structures.

3.8. Structure of the teaching process

Based on this "general version" of conceptual structure of the topic of making algorithms and programming, it can be seen that there are very complex relationships between the concepts.

Therefore there is not, or only very difficult to determine the "correct" order of the teaching of these concepts. This "correct" order depends on many factors. One of these is the context in which students are learning a programming language and where they meet with these concepts. Another important factor is the time, that we can spend on teaching the topic, the students' or group's previous knowledge, and their age-specific characteristics (such as how abstractly they can think). Similarly, learners' interests can play an important role, because it can influence which areas they can relate the new concepts to.

We are planning to develop several versions of the concept structure, taking into account the specialties of each programming environment and programming language.

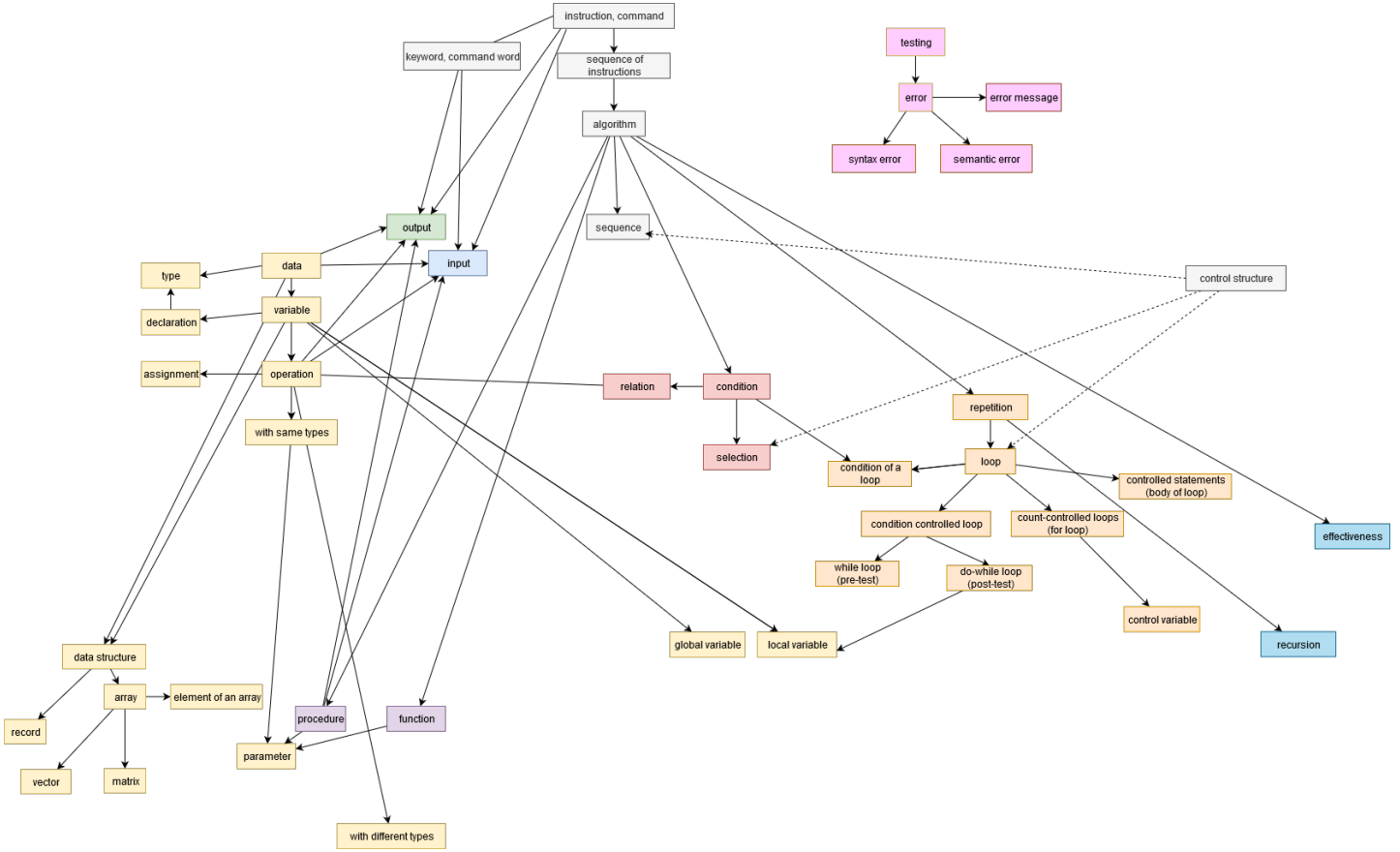

Figure 7: The "complete" conceptual structure of the topic of algorithmic programming

References

[1] Ambrus, A., Theoretical bases of teaching problem solving (the publication is in Hungarian: A problémamegoldás (feladatmegoldás) tanításának elméleti alapjai), *Pedagógiai Szemle* Vol. 10 (2002)

[2] Chaudhry, N.G., Rasool, G., A Case Study on Improving Problem Solving Skills of Undergraduate Computer Science Students, *World Applied Sciences Journal*, Vol. 20 (2012), 34–39.

[3] David, G., On Inductive Process in Algorithmi Problem Solving, *Olympiads in Informatis*, Vol. 8 (2014), 81–91.

[4] Kontra, Gy., Problem and problem solving thinking (the publication is in Hungarian: Probléma és problémamegoldó gondolkodás), *Magyar Pedagógia*, Vol. 96/4. (1996), 341–366.

[5] Mulder, F., Computer Science: from a BÉTA to a DELTA subject, *Informatica, Tinfon*, Vol. 11 (2002), 48.

[6] Papert, S., Mindstorms. Children, Computers and Powerful Ideas, *New York: Basic Books*, Inc. Publishers (1980)

[7] Pólya, Gy., Mathematical Discovery. On Understanding, Learning, and Teaching Problem Solving (in Hungarian: A problémamegoldás iskolája), *Tankönyvkiadó*, Budapest (1970)

[8] Pólya, Gy., How to Solve It, A New Aspect of Mathematical Method (in Hungarian: A gondolkodás iskolája), *Akkord Kiadó*, (2000)

[9] Saeli, M., Perrenet, J., M.G.Jochems, W., Zwaneveld, B., Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective, *Informatics in Education*, Vol. 10 (2011), 73–88.

[10] Schwepker, E., Algorithm Problem Solving Strategies, `https://dev.to/moresaltmorelemon/algorithm-problem-solving-strategies-21cp` (available: 28.01.2020.)

[11] Szlávi, P., Zsakó, L., Programming versus application, *Mittermeir R.T. (eds) Informatics Education – The Bridge between Using and Understanding Computers. ISSEP 2006*, Lecture Notes in Computer Science, 4226 (2006), 48–58.