



Saudi Computer Society, King Saud University

Applied Computing and Informatics

(<http://computer.org.sa>)
www.ksu.edu.sa
www.sciencedirect.com



ORIGINAL ARTICLE

Multimedia application for educational purposes: Development of algorithmic thinking



Eva Milková *

Department of Informatics, Faculty of Science, University of Hradec Králové, Rokitanského 62, Czech Republic

Received 11 April 2014; revised 6 May 2014; accepted 8 May 2014

Available online 20 May 2014

KEYWORDS

Computer science
education;
Multimedia applica-
tion;
Algorithm;
Algorithmic thinking

Abstract This paper is based on many years' experience with multimedia applications supporting the area of computer science education and it could serve as an inspirational material directed to all educators developing students' algorithmic thinking. Education of subjects related with computer science is from the perspective of other for centuries taught subjects, still in its infancy. Even nowadays a teaching method aimed at developing algorithmic thinking of students is still the subject of extensive discussions and teachers are looking for different ways on how to access it to students. Next to the educational approach to this base of computer science it is also important to find a suitable support for students' self-learning. Multimedia applications give teachers an excellent chance to demonstrate and visualize the subject matter more clearly and comprehensibly, as well as also enabling them to prepare study material for students which optimizes their study habits. Along with large software products developed by a team of professionals there are also various smaller programs dealing with objects appropriate to course subject matter created on a script given by the teacher with regard to students' needs. In the paper such application prepared

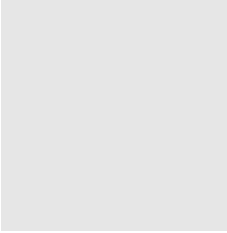
* Tel.: +420 493331337.

E-mail address: eva.milkova@uhk.cz

Peer review under responsibility of King Saud University.



Production and hosting by Elsevier



to intensify self-preparation of students in subjects developing algorithmic thinking is introduced and its benefit discussed. Animations useful to be used as an introductory complement to lectures are introduced as well. At the end advantages of the professional virtual learning environment containing such study material are mentioned.

© 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.

1. Introduction

The area of software development has passed a rapid expansion and this trend continues so far. Each developer has to learn constantly and master new technology. Crucial role is played by the basis which developer gains at the beginning of his–her career. Thus an essential part of studies at faculties preparing students in the area of computer science is the development of student’s ability to think algorithmically. Students must be able to create various algorithms solving given problems starting with easy ones and consecutively increase their algorithmic knowledge and shifts during studies till the level where they deeply understand much more complex algorithms.

Multimedia applications have substantially influenced education. They give teachers an excellent chance to demonstrate and visualize the subject matter more clearly and comprehensibly, as well as also enabling them to prepare study material for students which optimizes their study habits. Along with large software products dealing with a wide spectrum of objects developed by a team of professionals there are also various smaller programs dealing with objects appropriate to course subject matter created on a script given by the teacher with regard to students’ needs. The author of the paper has prepared with her students such multimedia applications for many years.

In the paper one application and animations prepared to intensify self-preparation of students studying the subject *Algorithms and Data Structure* are introduced and their benefits are discussed. At the end advantages of the professional virtual learning environment containing such study material are emphasized as well.

2. Algorithmic thinking development

There have been still long discussions regarding what kind of programming is suitable for beginners (cf. e.g. [Horák and Mitrovič, 2012](#); [Guniš and Šnajder, 2012](#)). Protagonists of object oriented languages argue that students beginning with structured programming acquire habits that cause big problems for them when using object oriented languages.

Our approach that we have been using for many years in the subject *Algorithms and Data Structures* is based on an imagination of a brick-box, where only several base elements are available from which children are able to create incredible

buildings. We do not use any programming language in the subject, students write algorithms on paper in Czech meta-language. The used Czech meta-language is nothing more than the Pascal programming language basic commands. (Remark: Pascal programming language was created by Nicklaus Wirth especially for educational purposes, see [Wirth, 1989](#).)

The mentioned process of algorithm design is reflected in the course structure. Thus, when we lead our students’ first steps in the creation of algorithms we explain to them that it is like building interesting objects out of just a few basic elements. In the subject *Algorithms and Data Structures* it means that we start our teaching with basic algorithmic structures (basic elements from the brick-box) and typical algorithmic structures (a few parts made out of these elements) written in Czech meta-language and then we let students get into the secrets of making whole algorithms (building whole constructions) written in Czech meta-language as well.

Let us describe our approach more precisely. We proceed in the following steps.

After explanation of the three simple commands (`read`, `write` and `assign:=`) we describe basic algorithmic structures (i.e., basic elements from the brick-box) – block of commands, incomplete and complete branching, and loop construction with the condition in the beginning, i.e.:

<i>blocks of commands</i>	<i>incomplete branching</i>	<i>complete branching</i>	<i>loop construction</i>
begin	if condition then	if condition then	while condition do
statement1;	statement	statement1	statement
statement2;		else statement2	
...			
statementN;			
end			

Using these basic structures students are soon able to understand typical algorithmic structures (e.g., in the given sequence to determine the number of elements with the given property, to calculate the sum and the product of some elements, to find the minimum and the maximum value, to find the last/first occurrence of the searched value, to remove value – see the following example, to add new value, etc.). Subsequently we let the students dive into the secrets of making algorithms (i.e., building whole constructions).

Example

Removal of *k-th term* of numerical sequence (a_1, a_2, \dots, a_n) , $1 \leq k \leq n$, saved in an array **a**.


```
begin
read(k);
for i from k to n-1 do
a[i]:= a[i + 1];
n:= n - 1;
end.
```

When explaining an algorithmic structure we always illustrate it on an example, e.g. in the following way.

Example

Let us imagine that the following sequence of integers is saved in the array **a** of length 10. We remove the sixth term, i.e. we remove the value saved in the term **a**[6] and shorten the array by one element.

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
124	56	88	0	0	65	584	12	0	6



a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
124	56	88	0	0	584	12	0	6	6

We explain and students practise all algorithmic structures, at first only those which use single variables. After the thorough practising structures on problems using single variables we proceed further and explain the data structure of one-dimensional and two-dimensional array. Using these data structures all previous matter is repeated together with a careful attention to the work with array indices.

During lessons students apply the acquired knowledge to a variety of tasks. They work in groups of two or three and each group is responsible for solving one of the given tasks. After some time when students prepare their solutions on a piece of paper, each task is illustrated and presented by two students (or three, depending on the number of groups, each group responsible for the task-solution deposes one student) on the blackboard and their solutions are compared and discussed by all students. *On the one hand this means that students are led to try and find more solutions to the given task, on the other hand when incorrect solutions occur among the presented solutions the teacher has an opportunity to discuss with students where the problem is.*

Dealing with the given topic we start to solve easier tasks and consecutively proceed to more difficult ones. We carefully discuss mutual relations among algorithms. Let us demonstrate it in the following two examples:

Example 2.1

Let us suppose that we are working with a finite numerical sequence (a_1, a_2, \dots, a_n) of n terms being already saved in an array **a** of the length n , $n \geq 1$. Let us create algorithms solving the following tasks.

- Remove the first three terms a_1, a_2, a_3 from the sequence.
- Remove the fifth till ninth terms, i.e. a_5, \dots, a_9 , from the sequence.
- From the sequence remove the terms a_p, \dots, a_q , supposing $p < q$.

Example 2.2

Compare the three algorithms solving the same task “Find out the gained amount of a gathering” and introduce an appropriate practical situation to each given algorithm.

(I)

```
begin
  read(p);
  sum := 0;
  i := 1;
  while i ≤ p do
    begin
      read(present);
      sum := sum + present;
      i := i + 1;
    end;
  write ("the gained amount = ", sum);
end.
```

(II)

```
begin
  sum := 0;
  read(present);
  while present ≥ 0 do
    begin
      sum := sum + present;
      read (present);
    end;
  write ("the gained amount = ", sum);
end.
```

(III)

```
begin
  sum := 0;
  read (continue);
  while continue < > 'NO' do
    begin
      read (present);
      sum := sum + present;
      read (continue);
    end;
  write ("the gained amount = ", sum);
end.
```

The subject *Algorithms and Data Structures* is placed into our curricula before other subjects which deal with algorithmic and programming skills.

3. Self-study and the feedback

There is an important question. How can students get feedback for their solutions written on paper in the Czech meta-language when studying at home? There are a lot of tasks that we give our students to solve. They solve not only the whole tasks

but we also let them complete prepared algorithms and determine values of variables similarly as you can see in the following two examples.

Example 3.1

Task: Complete the algorithm which calculates and writes out the following value of sum: If $x \leq y$ then it is the sum of integers $x, x + 1, \dots, y$, and if $x > y$ then it is the sum of integers $x, x - 1, \dots, y$.

```

begin
  read(x);
  read(y);
  sum := .....;
  number := x;
  if x ..... y then
    while number ..... y do
      begin
        sum := sum + .....;
        number := number + 1;
      end
    else
      while number ..... y do
        begin
          sum := sum + .....;
          number := number - 1;
        end;
      write ("The sum of integers from ‘,x,’ to ‘,y,’ is equal to”, sum, ‘.’’);
    end.

```

Solution

```

begin
  read(x);
  read(y);
  sum := 0;
  number := x;
  if x ≤ y then
    while number ≤ y do
      begin
        sum := sum + number;
        number := number + 1;
      end
    else
      while number ≥ y do
        begin
          sum := sum + number;
          number := number - 1;
        end;
      write ("The sum of integers from ‘,x,’ to ‘,y,’ is equal to”, sum, ‘.’’);
    end.

```

Example 3.2

Task There are integers saved in the two-dimensional array **a** (see the table **Starting position**). Determine the values in the array **a** after finishing the following algorithm. Write them to the table **Final position**.

```
begin
  m:= 3;
  n:= 4;
  for i:= 1 to m do
    for j:= 1 to n do
      read(a[i,j]);
  for i:= 1 to m do
    for j:= 1 to n do
      if a[i,j] > i + j then
        a[i,j] := a[i,j] - 1
      else
        a[i,j] := a[i,j] + 1;
end.
```

Starting position

	1	2	3	4
1	5	2	1	8
2	9	3	6	7
3	0	4	5	2

Final position

	1	2	3	4
1				
2				
3				

Solution (cf. [Fig. 1](#) thereafter)

Starting position

	1	2	3	4
1	5	2	1	8
2	9	3	6	7
3	0	4	5	2

Final position

	1	2	3	4
1	4	3	2	7
2	8	4	5	6
3	1	5	6	3

The answer to the question given at the beginning of this section is: Students can practise their knowledge using multimedia program *ALGORITHMS* developed by our student within his thesis ([Voborník, 2006](#)).

3.1. Program ALGORITHMS

The program *ALGORITHMS* is created in the Delphi environment and it really is an excellent tool for both beginners and advanced students in the field of algorithms.

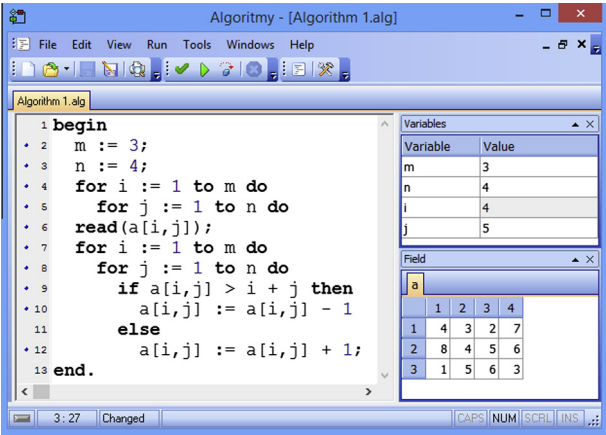


Fig. 1 Program Algorithms – Solution of the above given Example 3.2.

The program is user friendly and its functions are arranged to be intuitive and at the same time to remind professional editors and debuggers of well-known programming languages, which also facilitates their subsequent transition. Because many users are beginners the program is free of many unnecessary features which would rather complicate its use at this level. (Milková, 2011)

Using the program, students can place their solution of the given task, written in Czech meta-language, into the program and the program shows them step-by-step how their algorithm works, if it is correct or not.

The program also shows the actual values of used variables in each step of the algorithm’s process. In this way students can easily check their paper-results from tasks similar to the above introduced example 3.2, against the results of these tasks displayed in the program (see Fig. 1).

Using the *ALGORITHMS* program students can go at their own speed through each algorithm and better understand the explained matter.

The program can be downloaded from http://lide.uhk.cz/prf/ucitel/milkoev1/en_index.htm using page ALGDATS > Lecture – Students presentations.

Remark: The *ALGORITHMS* program can be localized into different languages. Hence, the program can theoretically be used in the users own mother tongue, including the possibility to define own keys of used meta-language. (Voborník, 2011)

3.2. Visualization within animations

The above *ALGORITHMS* program is not the only multimedia study material used within the subject *Algorithms and Data Structure*. It is essential material for student’s feedback but we also use several smaller digital objects as a support of lectures and students’ imagination.

As an example let us mention animations, created in Macromedia Flash environment, visualizing on six snowmen three prime sorting techniques: Selection Sort (see Fig. 2), Insertion Sort and Bubble Sort. These small presentations were created by a student through their optional project and they can be downloaded from the same page as the *ALGORITHMS* program - see above mentioned link.

The three prime sorting methods help us when we explain the term algorithm at the beginning of our lectures and emphasize the fact that the given task can be mostly solved by more than one algorithm. We provoke students to imagine the whole process (algorithm) of sorting things by size. We properly explain the idea of each prime algorithm and usually demonstrate it with the help of students. It is known that the main idea of Insertion Sort is usually described as the way many people sort a hand of playing cards. We also remark this illustration; however, we prefer to explain the main idea of each prime sorting algorithm as the way children rank in a line. In the lecture there are always several students willing to be ranked to illustrate the sorting processes. Students can later remind this “show of the appropriate sorting algorithm” using the animations sorting six snowmen. We go back to this topic later (almost at the end of the term) when discussing the formulation of each prime algorithm in the used Czech meta-language.

3.3. Textbook Algorithms – basic constructions and their visualization

The whole area explained within the subject *Algorithms and Data Structures* is introduced in the textbook [Milková et al. \(2010\)](#), where more than 150 problem assignments, questions and exercises are presented. The accuracy of a solution can be verified with the help of the program *ALGORITHMS* which is enclosed, together with solutions of all the textbook’s given tasks, on the CD attached to the textbook.

3.4. Virtual learning environment

Education at the University of Hradec Králové is supported in a virtual learning environment (the Blackboard learning system supports future engineers and managers studying at the Faculty of Informatics and Management, the Moodle learning platform supports future teachers studying at the Faculty of Science). The virtual learning environment is an opportunity for students to combine their occupation with studies. Relevant study material is accessible whenever and wherever.



Fig. 2 Macromedia flash animation – Selection Sort of six snowmen.

There are many tools each virtual learning environment offers (cf. [Hubálovský and Šedivý, 2011](#); [Kostolányová et al., 2011](#); [Salem, 2011](#)). However, virtual courses have to be clearly organized and should not contain redundant information. Here let us review in alphabetical order the tools that we use in the *ALGDS* course created for the subject *Algorithms and Data Structures* to support it and let us describe their functions in our *ALGDS* course. (cf. [Milková, 2011](#)).

- *Announcements*: This tool serves the teacher as a reminder of important dates (written credit tests, examination etc.) and offers necessary information to students.
- *Calendar*: The tool is used by students to remark information about important dates.
- *Course Content*: The tool serves as the main course page. Here, students find a detailed plan of lectures, the *ALGORITHM*S program, samples of credit and exam tests, and information concerning recommended literature and credit and exam conditions.
- *Discussion*: Using this tool, students have the opportunity to discuss their questions and problems concerning the appropriate subject with other students. The tool is displayed in our *ALGDS* course mainly as a means of communication among students. Thanks to the lectures and lessons, the *Discussion* tool is only seldom used by teachers.
- *Evaluation*: The lecturer and assistants insert evaluations of credit tests into the virtual environment thus every student can find evaluations of his/her work.
- *Modules*: This tool is used to assign all electronic study materials (additional text to the textbook [Milková et al. \(2010\)](#), problem statements of tasks solved in lesson, presentations, and animations) that correspond to a lecture to the appropriate module. The number of modules is the same as the number of lectures provided during a term. Moreover, students interested in an area explained within a subject can find here files and/or links to additional information outside the immediate framework of the subject.

Thanks to the textbook [Milková et al. \(2010\)](#) containing solved problem assignments and CD of the *ALGORITHM*S program and solutions of all the given tasks in the textbook, the *ALGDS* course serves as an optional complement to the subject *Algorithms and Data Structures*.

Although the course is optional, students attend the course at the start of the term to download the detailed plan of lectures (lectures closely correspond to the content in the textbook; the plan includes links to pages whose content will be discussed in the appropriate week) and information concerning recommended literature as well as credit and exam conditions and samples of credit and exam tests.

Students find it very handy to see all animations and presentations used in the lecture that are placed within appropriate module as well as the list of problem statements of all exercises solved in lessons.

4. Results and discussion

The engagement of students into a subject and their interest in the subject matter belongs to crucial elements of successful teaching and learning process. Students preferentially take in and process information in different ways: by seeing and hearing, reflecting and acting, reasoning logically and intuitively, analyzing and visualizing, steadily and in fits and starts. According [Felder and Silverman, 1988](#); there are four main two-poles-dimensions of learning styles; active/reflective, sensing/intuitive, visual/verbal, and sequential/global.

Balanced study material taking into account the above described learning dimensions and effecting systematical learning can influence engagement, interest and students' knowledge very much. On one hand it supports the preferred learning dimension pole of a student and on the other hand it enhances his/her ability to absorb subject matter also with regard to the opposite learning dimension pole. Hence, students' ability to take in and process information in various ways is increased and learning difficulty decreased.

4.1. Felder-Silverman learning style model

There are many publications devoted to learning styles. Recently we began to be interested in Felder-Silverman learning style model (cf. [El-Hmoudová, 2013](#)). Let us briefly introduce the main characterization of, in the model given, four [Felder and Silverman, 1988](#); learning style dimensions:

Active and reflective learners

- Active learners tend to retain and understand information best by doing something active with it – discussing or applying it or explaining it to others.
 - Reflective learners prefer to think about it quietly first.
- “Let’s try it out and see how it works” is an active learner’s phrase.
 - “Let’s think it through first” is the reflective learner’s response.
- Active learners tend to like group work more than reflective learners.
 - Reflective learners prefer working alone.

Sensing and intuitive learners

- Sensing learners tend to like learning facts,
 - Intuitive learners often prefer discovering possibilities and relationships.
- Sensing learners don’t like courses that have no apparent connection to the real world;
 - Intuitive learners don’t like “plug-and-chug” courses that involve a lot of memorization and routine calculations.

Visual and verbal learners

- Visual learners remember best what they see - pictures, diagrams, flow charts, time lines, films, and demonstrations.
 - o Verbal learners get more out of words - written and spoken explanations.

Sequential and global learners

- Sequential learners tend to gain understanding in linear steps, with each step following logically from the previous one.
 - o Global learners tend to learn in large jumps, absorbing material almost randomly without seeing connections, and then suddenly “getting it.”
- Sequential learners tend to follow logical stepwise paths in finding solutions.
 - o Global learners may be able to solve complex problems quickly or put things together in novel ways once they have grasped the big picture, but they may have difficulty explaining how they did it.

The preference for one category or the other may be strong, moderate, or mild.

4.2. Index of learning styles

Based on the above mentioned model of four learning style dimensions the Index of Learning Styles (ILS shortly) was developed by Barbara A. Soloman and Richard M. Felder. For more information see web-link to ([Felder and Soloman, 2013](#)).

ILS is a self-scoring questionnaire for assessing preferences on four dimensions of the Felder-Silverman model. ILS may be used at no cost for non-commercial purposes by individuals who wish to determine their own learning style profile and by educators who wish to use it for teaching, advising, or research.

In the academic year 2012/13 as well as this academic year 2013/14, the above mentioned ILS was implemented in the *ALGDS* course into the virtual learning environment and students are asked to fill in the ILS at the beginning of the semester.

From 382 responses, with respect to distinguish strong, moderate, or mild preference for one category or the other, we gained the following results.

According our premises, in each above-mentioned academic year there were about 98% students belonging to strong visual learners.

More students belong to moderate till strong sensing learners (about 76% in each year).

Concerning the other two dimensions, active/reflective and sequential/ global, most students belong to the mild level of these categories more of them to the “left side”, which means to the mild active and mild sequential dimension.

Hence, with respect to the paper topic, the result that 98% students belong to strong visual learners confirms usefulness of multimedia applications usage as an educational support. Everyone learns more when information is presented both visually and verbally.

5. Conclusion

In the paper we introduced our approach to the development of algorithmic thinking of beginners within the subject *Algorithms and Data Structures* using multimedia applications. Multimedia applications serve as a useful educational support that is very welcome and appreciated by students.

The paper follows the paper [Milková \(2012\)](#) and it is also intended as an inspiration for all educators developing students' algorithmic thinking. Sometimes it is not easy to find an appropriate content and approach to teaching a subject and each inspiration could be valuable to know and possibly to apply.

References

- El-Hmoudová, D., 2013. The Impact of Learning Style Dimensions on Computer-based Key Language Competences Testing. *Procedia - Social and Behavioral Sciences*. Elsevier, Amsterdam.
- Felder, R.M., Silverman, L.K., 1988. Learning and teaching styles in engineering education. *Eng. Educ.* 78 (7), 674–681.
- Felder, R.M., Soloman, B.A., 2013. Index of Learning Styles, retrieved September 23, 2013, from <<http://www.ncsu.edu/felder-public/ILSpa.html>>.
- Guniš, J., Šnajder, L., 2012. The model of algorithmic thinking – dimensions and levels. In: *Information and Communication Technology in Education (ICTE 2012)*, Ostrava: University of Ostrava, Rožnov pod Radhoštěm, Czech Republic, September 11–13, 2012, 69–78.
- Horák, O., Mitrovič, L., 2012. Description of the Basic Algorithm Blocks and Structures Representation in Courses of Algorithm Development, In: *Wseas Transactions on Advances in Engineering Education*, vol. 9(2), pp. 43–51.
- Hubálovský, Š., Šedivý, J., 2011. Education of student's project team cooperation using virtual communication supported by LMS system, In: *14th International Conference on Interactive Collaborative Learning (ICL2011) – 11th International Conference Virtual University (vu'11)*, Slovenská technická univerzita, Bratislava, Slovakia, pp. 456–459.
- Kostolányová, K., Šarmanová, J., Takács, O., 2011. Structure of Study Supports for Adaptable Instruction. *New Educ. Rev.* 25 (3), 235–247.
- Milková, E., 2012. Multimedia application supporting self preparation to develop algorithmic thinking. In: *Recent Advances in Computer Engineering Series, Proc. of 3rd International Conference on Applied Informatics and Computing Theory (AICT'12)*, WSEAS Press, Barcelona, Spain, October 17–19, pp. 155–159.
- Milková, E., 2011. Graph theory and algorithms: various approaches to utilization of virtual learning environment. In: *14th International Conference on Interactive Collaborative Learning (ICL2011) – 11th International Conference Virtual University (vu'11)*. Slovenská technická univerzita, Bratislava, Slovakia, pp. 637–642.
- Milková, E. et al, 2010. *Algoritmy – základní konstrukce v příkladech a jejich vizualizace* (in English: *Algorithms – basic constructions and their visualization*). Gaudeamus, Hradec Králové.
- Salem A.M., 2011. Intellectual E-Learning Systems, In: *Proc. Of the Annual International Conference on “Virtual and Augmented Reality in Education” (VARE 2011)*, Valmiera, Latvia, 16–23.
- Voborník, P., 2006. *Programovací jazyk pro podporu výuky algoritmů*, Hradec Králové: thesis.
- Voborník, P., 2011. Teaching algorithms using multimedia tools. In: *Proc. of the conference Efficiency and Responsibility in Education 2011*. Praha: FEM CULS, pp. 312–321.
- Wirth, N., 1989. *Algoritmy a datové štruktúry údajov*. Alfa, Bratislava.