A Web-based approach to support initial algorithmic procedural programming learning

Marcelino, M. J.
University of Coimbra
Dep. Engª Informática, Polo II
3030-290 Coimbra - Portugal
+351 239 790067
zemar@dei.uc.pt

Tosheva, I.
University of Rousse
Dep. of Computer Systems
7017 Rousse, Bulgaria
+359898921763
reni_20_kzl@mail.bg

Dobrodzhaliev, Y.
University of Rousse
Dep. of Computer Systems
7017 Rousse, Bulgaria
+359898584619
Janysoft@mail.bg

Gomes, A.
Polytechnic Institute of Coimbra
Dep. Engª Informática e Sistemas
3030-199 Coimbra - Portugal
+351 239 790 350
anabela@isec.pt

Mendes, A. J.
University of Coimbra
Dep. Engª Informática, Polo II
3030-290 Coimbra - Portugal
+351 239 790000
toze@dei.uc.pt

ABSTRACT
Student difficulties in programming learning are not easily overcome, especially in initial learning stages. To minimize these difficulties, it is important that students have an active role in their learning process. The availability of interactive tools that can support students' work, both in classes and autonomously, can have a positive impact in their learning performance. In this paper we describe SICAS-W, a Web-based system to support students in initial algorithmic procedural programming learning. With this environment students can build, simulate, test and compare algorithms for proposed problems and submit and test their solutions online.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education - *Computer science education*.

General Terms
Algorithms, Languages.

Keywords
Algorithmic learning, problem-solving techniques, programming learning.

1. INTRODUCTION
Programming learning is not easy for beginners and brings them a lot of difficulties, commonly reported in many research studies [1-2]. This is particularly important in early learning stages, as many students fail to be approved in initial programming courses, leading also to difficulties in other courses that expect them to have already developed good programming skills. This situation can degrade student confidence and self motivation and in some cases lead to abandon.

Mainly, to learn programming is to learn another way to solve problems (many times previously known from other disciplines) Students must develop capacities like abstraction, generalization, critical thinking, transfer, etc., and that is not simple to many freshmen.

Over the years, several approaches and software tools have been developed to support programming learning. Antunes in [3] overviews several of these approaches quite comprehensively. They include:

- Simpler Programming Languages or mini-languages,
- Controlled development environments,
- Micro-worlds,
- Tools to test solutions,
- Tools to algorithm, or program, animation and simulation,
- Online courses, e-books, dedicated sites.

Examples of the first category are BASIC language (designed to be simpler to use than traditional general-purpose languages) and MiniJava (a mini-language or a subset of Java) [4-5].

Controlled development environments are simpler than professional IDEs and designed with educational purposes. Available options are limited to reduce the usual complexity of their professional counterparts. An example is BlueJ [6].

Micro-worlds are environments where the user has to ask some character to perform specific tasks in a simulated world, usually more concrete and close to the student context than other typical programming environment. Karel the Robot is a well-known example of this approach [7].

Tools to test student solutions to problems usually compare the results of a student program with several provided input/output datasets. Among them we can mention WebToTeach [8] and ELP [9], systems developed for the Web.

Among tools to algorithm or program animation and simulation we can distinguish between specific tools, that only allow the animation of predefined examples (usually built by the teacher), and more general tools, that allow the animation and simulation of any solution (algorithm or program) developed by the student. In the first subgroup we can mention JHAVÉ [10] and in the second JELIOT [11] and SICAS [12-13].

We have been using SICAS for some years to support the initial learning stages in our procedural programming course at the Informatics Engineering Department of the University of Coimbra. Although we believe that SICAS has been useful to many of our students during classes, it was not used by many of them in their autonomous work. Several reasons may be pointed for this, but mostly students say they did not use it because they did not install it in their computers and they did not have sufficient problems to solve. Also teachers point the fact that SICAS does not give them information about student's performance and difficulties, making it unfeasible for them to follow students' work.

2. A WEB-BASED SYSTEM TO SUPPORT INITIAL PROGRAMMING LEARNING

To minimize the above problems we decided to develop a Web based version of SICAS, that we called SICAS-W. This version introduces four main changes relatively to the standalone version:

- Availability,
- Diversified sets of problems for the students to solve (organized by difficulty level),
- Information about student performance and
- Algorithm validation.

There is an extra advantage, since the same SICAS-W installation can be shared by several teachers, even from different institutions, so that each one can contribute with a small set of problems to a bigger database. This allows the students to have a larger number of problems to solve without imposing a heavy work on the teachers.

Because SICAS-W is based on SICAS, before describing the system main characteristics, we will start, in the next section, by describing SICAS standalone main features.

2.1 SICAS standalone

SICAS was designed to support learning of basic procedural programming concepts, such as selection and repetition. It is language independent and oriented to the design and implementation of algorithms. Using SICAS students are encouraged to develop their capacities through problem solving. They can create solutions to problems, simulate them and see if they work as expected. The simulator can be used to detect and correct errors, but also to look for alternative ways to solve the problems and to compare them. SICAS supports a constructivist approach to learning, as each student assumes an active role, learning at his own pace and progressively constructing his own knowledge.

In SICAS, algorithm design is supported by an iconic environment where the student builds a flowchart based on typical graphical symbols that represents the solution (see Figure 1 for an example). Flowcharts were chosen to express algorithms, instead of pseudo-code, because many studies reinforce the idea that they are more appellative, facilitate understanding, and are simpler and probably less prone to errors than pseudo-code [14].

Algorithms developed with SICAS can include assignments, input/output, repetition and selection instructions. They can use numeric and string variables. Functions can also be defined and used. These elements can be introduced in a flowchart by clicking (in the toolbar icons) and pointing (in the design area). When that happens, a dialog box automatically opens asking the student to specify the element details (e.g. the condition in a selection). This option helps students to avoid common novice programmer syntax errors. Lines connecting components are automatically inserted, avoiding inconsistencies in the flowchart. The environment also includes the ability to delete, modify or copy any component.

Any algorithm created with SICAS is automatically translated to pseudo-code, C and JAVA code. These alternatives show that a well designed algorithm can be easily translated into several programming languages and that the most important factor in algorithm design is its conception, not the programming language in which it is coded.

After creating an algorithm, it is possible to see its animated simulation. The student can control the speed at which the simulation progresses (step-by-step, slow or fast), pause the simulation and go back to repeat any part of the execution. This allows a deeper analysis of available data and/or a discussion with the teacher or other learners.

2.2 SICAS-W

SICAS-W allows two types of registered users:

- students and
- teachers.

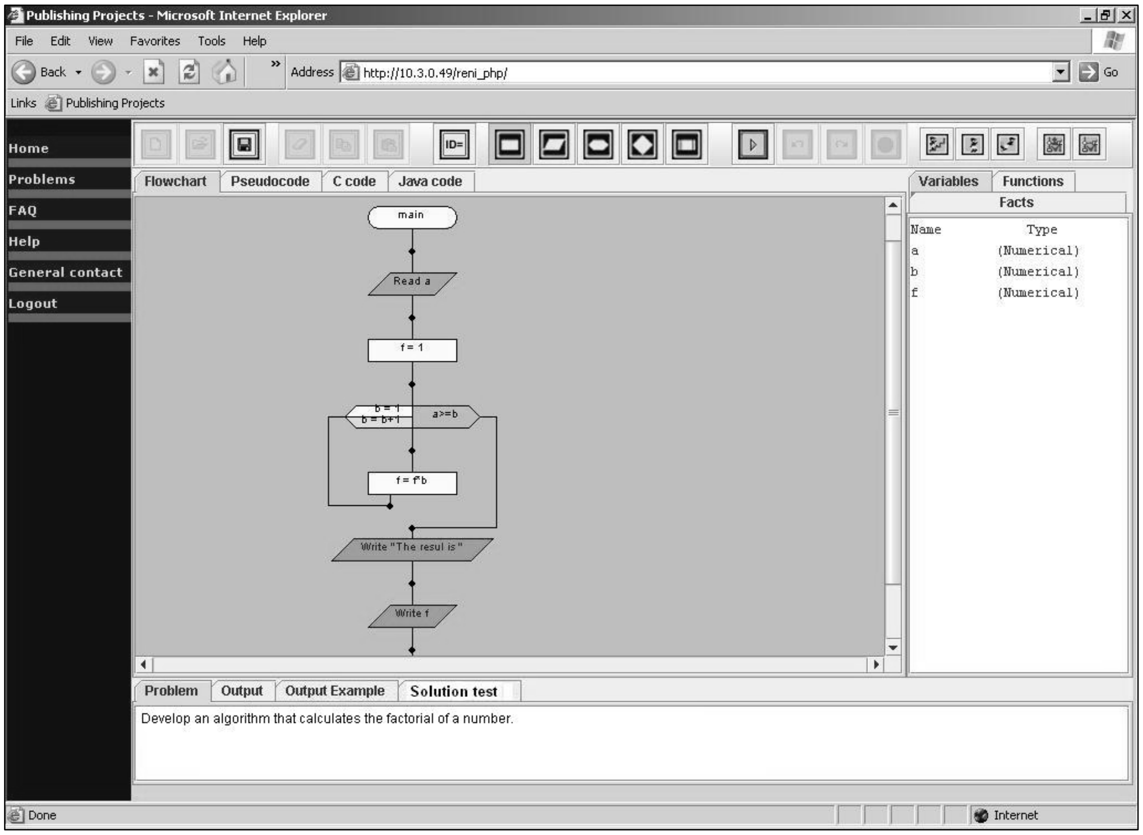
Of course, to have access to all the environment functionalities both students and teachers must have an account and must login.

A main job for the teacher in this system is to provide problems for students to solve. A teacher can insert, delete and update problems in the environment and also monitor the students' evolution and the students' global response to problems (if many students are able to solve a particular problem, or not, etc.).

As we can see in Figure 1, the main page of the system is organized in two frames or sections.

In the left frame, there is a menu of options. After login, the menu changes accordingly to the type of user that has logged in.

In the right frame what appears depends also on the option that has been selected. If it is a student and he has selected a problem to solve, the Web version of SICAS appears (see Figure 1).



Name	Type
a	(Numerical)
b	(Numerical)
f	(Numerical)

Develop an algorithm that calculates the factorial of a number.

Figure 1. Main page of SICAS-W in student mode showing a problem to calculate the factorial of a number.

Problems are organized by level of difficulty. At present, there are three levels of difficulty:

- Easy,
- Medium and
- Hard.

In each level there are several problems that can be solved. Before selecting a specific problem the user has to select the level he wants to work.

When a problem is selected, the problem formulation appears in the Problem area and the student has at his disposal a tool where he can build an algorithm that solves the problem (see Figure 1 for an example).

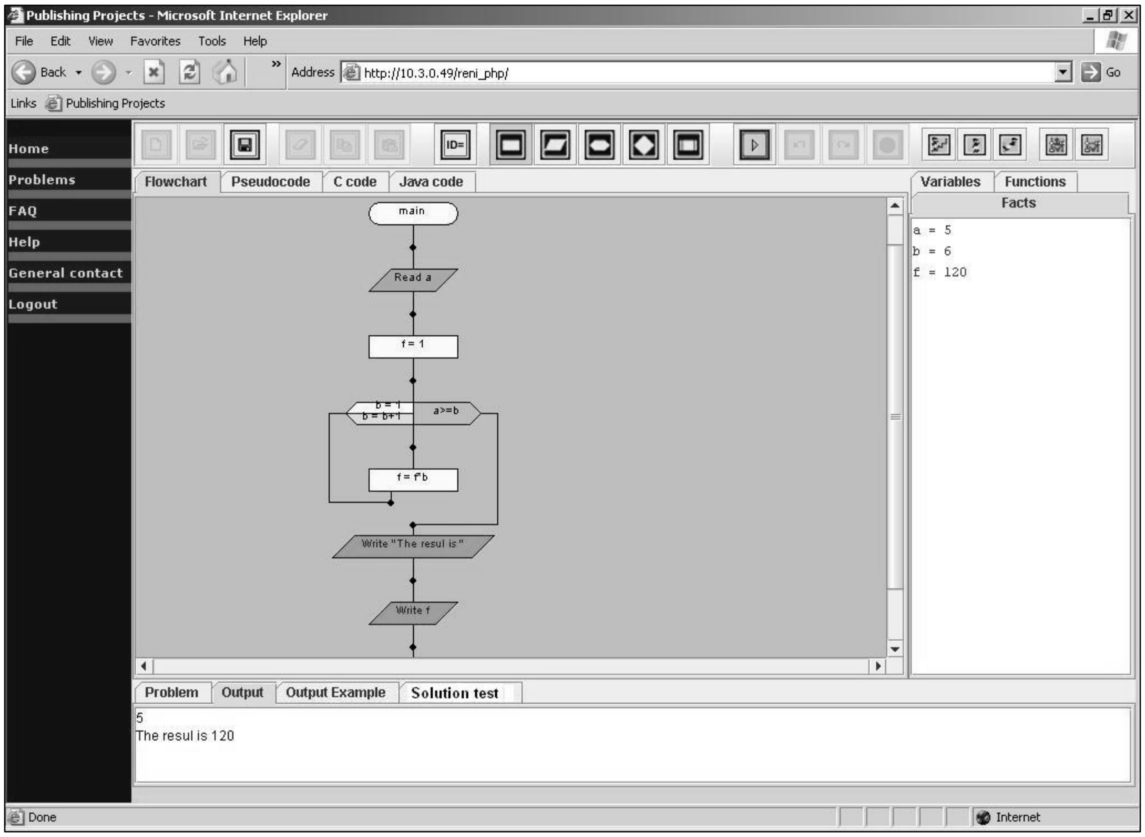
After, a student can simulate the algorithm, with several control speeds, and watch the result of his solution (see Figure 2 for the final result of the simulation of the previous example).

A problem has a formulation (a text), one or more algorithms that solve it and, eventually, some data test sets. The fact that a problem can be solved using several algorithms caters for different styles and forms of understanding, allowing the student to compare them and find out which resolution is more adequate for a given problem.

If a set of problem test data (inputs and outputs) was provided by the teacher, the student can compare his solution with the teacher's and validate it. This is particularly useful for special cases (e.g. invalid data or data that will likely produce erroneous situations), since many students tend to get satisfied if their solutions work for the average case, without caring to see if they work in any case.

After selecting and solving a problem students can submit the final solution to the system. The system them automatically analyses the solution by testing it against a set of case tests introduced by the teacher and informs the student of the result. For the moment an algorithm can only pass or fail such a test. If it passes it is probably rightly designed. If it fails it must have at least some type of error, but the system does not give the student any information about it. However, if a teacher has not provided the test data set the algorithm can not be tested. In that case the system will inform the student with an adequate message (in the "Solution test" area). Still, we encourage every teacher to do it as it can be quite beneficial for a student to have some feedback on the quality of his solution.

This idea of automatic assessment of programs developed by students is not completely new, but usually it is used in more advanced stages of programming learning and not in the very beginning with algorithms and their animation [15-17].



```
a = 5
b = 6
f = 120
```

```
5
The resul is 120
```

Figure 2. SICAS-W in student mode showing the final result of the factorial problem.

One important feature of both SICAS and SICAS-W is that the interface language can be changed easily as the tools were designed in such a way that they are adaptable to the user language. This includes also pseudo-code and problem formulation that should be also in the user mother language. As one of the tool intentions is to broad its audience and share resources, this is a very important aspect. By default everything shown to the user is in his language, but if he changes the language the whole system adapts, including the problem text (if provided when introduced into the system by the teacher).

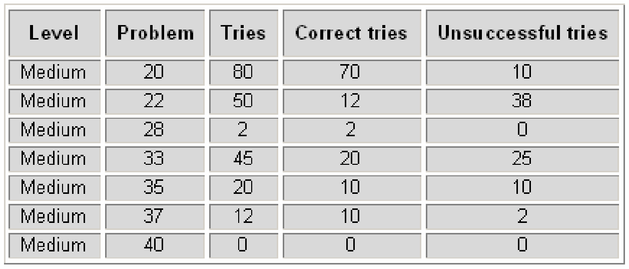
In what concerns tools for the teacher, the system has some statistical tools too to monitor/evaluate students' performance. It is possible to obtain from the database information about:

- a particular student,
- a particular problem,
- a particular level of problems,
- from a certain date onwards,
- for solved problems of a certain level,
- for unsolved problems of a certain level.

In the next table an example of a query done (by level) is shown.

edium level problems, appears. We can observe each problem number, how many times each has been tried, successfully solved and attempted (unsuccessfully tried).

Table 1. Result of a query made by a teacher to the system by problem level.



Level	Problem	Tries	Correct tries	Unsuccessful tries
Medium	20	80	70	10
Medium	22	50	12	38
Medium	28	2	2	0
Medium	33	45	20	25
Medium	35	20	10	10
Medium	37	12	10	2
Medium	40	0	0	0

From this information we can extract several relevant data. For instance, problems number 22 and 33 seems problematic as they were tried more times without success than with success. This is an easy way to identify difficult problems that may need an extra clarification or reformulation, including level reclassification, etc. Likewise, information can be obtained on the other items.

All the information used in the system is stored in a database. This database deals with student and teacher accounts, proposed problems, solved problems, students' performance, etc.

The technologies and languages used to build the site and the supporting database are HTML, the Java programming language, MySQL and PHP. These were chosen essentially because they are free technologies and languages.

3. CONCLUSIONS AND FUTURE WORK

In this paper we described a Web-based system to support initial procedural programming students' learning using algorithms as the first 'programming language'. Although the algorithms are expressed using flowcharts, they can be visualized as pseudo-code, Java code or C++ code. This allows us to use the system not only in the initial stage of learning, but also at a later stage, as a student can devise an algorithm first with a flowchart and ask the system to convert it automatically to one of these languages in order to run it in his current IDE.

The system works in two modes, one for teachers, to provide problems and monitor students' work, and another for students to design, simulate and automatically test their solutions to selected or proposed problems in the form of computer algorithms.

Soon it will be finished in order to make some preliminary usability tests with a restricted number of users (teachers and students), so that we can use it in a broader context in the next academic year to support the discipline of "Princípios de Programação Procedimental" (Principles of Procedural Programming) of the Informatics Engineering and Communications and Multimedia Degrees of the University of Coimbra. During this course we will conduct a more rigorous evaluation study where we will observe several students using the tool, pass a questionnaire and interview some of them in the end.

However, we can say that for the student SICAS-W will offer:

- An auto-testing mechanism that can enhance his own conscience about his learning,
- feedback about errors made,
- a challenge when trying to reach higher proficiency levels,
- a healthy competition among the class or classes and courses involved.

For the teachers it can offer an effective way to:

- provide more problems to students,
- know their students evolution,
- identify students' difficulties,
- improve student support,
- share experiences with other teachers.

4. REFERENCES

[1] Milne, I. and Rowe, G. Difficulties in Learning and teaching programming – views of students and tutors. *Education and Information Technologies*, 7, 1 (2002), 55-66.

[2] Johnson, D. Algorithmics and programming in the school mathematics curriculum: support is waning – is there still a case to be made? *Education and Information Technologies*, 5, 3 (2000), 201-214.

[3] Antunes, R. *Ambiente de apoio à aprendizgem de programação Web utilizando PHP*. Master Thesis, University of Coimbra, 2005.

[4] Kurtz, T. *History of Programming Languages*. Academic Press, New York, 1981.

[5] Roberts, E. An overview of MiniJava. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE'01)* (Charlotte, USA, February, 2001), 1-5.

[6] Kolling, M., Quig, B., Patterson, A. and Rosenberg, J. The BlueJ system and its pedagogy. *Journal of Computing Science Education, Special Issue of Learning and Teaching Object Technology*, 12, 4 (2003), 249-268.

[7] Pattis, R. *Karel the Robot: A Gentle Introduction t the Art of Programming*. John Wiley & Sons, 1981.

[8] Arnow, D. and Barshay, O. WebToTeach: An Interactive Focused Programming Exercise System. In *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference* (San Juan, Puerto Rico, November, 1999), 39-44.

[9] Truong, N., Bancroft, P. and Roe, P. A web based environment for learning to program. In *Proceedings of the 26th Australasian Computer Science Conference on Research and Practice in Information Technology* CRIPTS'03 (Adelaide, Australia, 2003), 255-264.

[10] Naps, T., Eagan, J. and Norton, L. JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualizations. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, 2000.

[11] Ben-Ari, M., Myller, N., Sutinen, E., Tarhio, J. Perspectives on Program Animation with Jeliot. *Lecture Notes in Computer Science*, 2269 (2002), Springer-Verlag, 31-45.

[12] Gomes, A. and Mendes, A. A animação na aprendizagem de conceitos básicos de programação, *Revista de Enseñanza y Tecnología*, 13 (1999), 22-32.

[13] Rebelo, B. Marcelino, M. and Mendes, A. Evaluation and Utilization of SICAS - A System to Support Algorithm Learning. In *Proceedings of the 8th IASTED International Conference on Computers and Advanced Technology in Education CATE 2005* (Aruba, August, 2005), 153-158.

[14] Scanlan, D. Structured Flowcharts Outperform Pseudocode: An Experimental Comparison, *IEEE Software*, 6, 5 (1989), 28-36.

[15] Wu, S., Tsai, S. and Yang, P. JAVALAB – A Java Tutorial and Programming Laboratory System. In *Proceedings of Exploring Innovation in Education and Research*, Taiwan, 2005.

[16] Foubister, S., Michaelson, G. and Tomes, N. Automatic assessment of elementary standard programs using Ceilidh, *Journal of Computer Assisted Learning*, 13 81997) 99-108.

[17] Korhonen, A., Malmi, L., Nikander, J. and Tenhunen, P. Interaction and Feedback in Automatically Assessed Algorithm Simulation Exercises. *Journal of Information Technology Education*, 2 (2003), 241-255.