AlgoWeb: A Web-Based Environment for Learning Introductory Programming

Ricardo Vargas Dorneles – CCTI/UCS – rvdornel@ucs.br,

Delcino Picinin Jr. – CCTI/UCS – dpicinij@ucs.br

André Gustavo Adami – CCTI/UCS – agadami@ucs.br

Abstract *- This paper describes a web environment named AlgoWeb for learning introductory programming. AlgoWeb has a syntax highlighting editor for structured Portuguese, performs step-by-step processing, and provides supports for breakpoints and monitoring variable values. In addition, AlgoWeb has a set of algorithmic problems that encompasses the concepts and techniques taught in introductory programming courses. AlgoWeb can also check the correctness of the solution (with respect to previously defined input and output data) submitted by its users for the problems available in the environment. All the interactions between the users and the environment are registered, providing a valuable tool to evaluate the progress of each student or class.*

Keywords: *Introductory programming, algorithms, learning tools, web-based learning*

Resumo *- Esse artigo descreve um ambiente de web chamado AlgoWeb para o aprendizado de algoritmos. O Algoweb possui um editor orientado à sintaxe de algoritmos em Português estruturado, executa processamento passo-a-passo, e suporta pontos de parada e monitoramento de variáveis. Ele também possui um banco de problemas algorítmicos que cobrem os conceitos e técnicas ensinados nas disciplinas de introdução a programação. O ambiente realiza a verificação automática da correção dos algoritmos submetidos para cada problema (conforme os dados de entrada e saída definidos previamente). Todas as interações entre os usuários e o ambiente são registradas, fornecendo uma ferramenta valiosa para avaliar o progresso de cada estudante ou da turma.*

Palavras-chave: *introdução à programação, algoritmos, ferramentas de aprendizagem, aprendizado baseado na web.*

1 Introduction

The choice of a language to teach the first steps in programming is still an open issue (Giangrande, 2007). It is very common in Brazilian Undergraduate Schools to use structured Portuguese to teach introductory programming, instead of regular programming languages, such as C or Pascal. Structured Portuguese is a simpler version of a programming language because it has few data types and data flow control structures, and it uses easy-to-remember Portuguese terms. This way, the student can focus more on the problem analysis and solution identification rather than in the syntax details of a given language to describe the solution. Furthermore, in programming courses, students tend to spend more time correcting syntax errors or understanding error messages from the compiler.

During the period that we have been teaching introductory programming, which is more than 20 years, we have observed several difficulties faced by the students in learning introductory programming:

1. The students do not understand the problem;

2. The students understand the problem, but they do not know its solution;
3. The students understand the problem and know the solution, but they are not able to identify the steps needed in the solution;
4. The students can identify the steps of the solution, but they do not know how to write it in an algorithmic form;
5. The students can write an algorithm, but they do not know how to test it.

Each one of the listed difficulties corresponds to the steps for program problem solving described by Winslow (1996).

The learning difficulties in the beginning of computer programming courses become a great obstacle for the students. Such difficulties discourage the students to the point of being one of the main causes for evasion in the beginning of Computer Science undergraduate courses. Another consequence of such difficulties is the high failure rate of students (Bennedsen and Caspersen, 2007). Therefore, it is imperative to develop tools and methods that can help students to learn how to program.

In this paper, we describe a web-based environment for learning introductory programming that provides several resources to improve the autonomy of the students in learning how to program. This way the environment significantly reduces the student dependence on teacher intervention. In addition, the environment provides tools for educators to follow the students' progress. Since 2009, the environment is being used by students from five courses in the University of Caxias do Sul.

2 Introductory Programming Learning Environments

The use of environments for learning introductory programming has been investigated for several years. Several environments have been proposed for structured Portuguese, such as, Visualg (Souza, 2009), webPortugol (Hostins and Raabe, 2007) and Portugol IDE (Manso et al., 2009). VisuAlg[1] is a Windows® program developed by Apoio Informática. It has an algorithm editor and several execution modes with possibility of monitoring variables value. Portugol IDE[2], developed by the Instituto Politécnico de Tomar, Portugal, is very similar to VisuAlg, but it allows the user to use flowcharts to represent an algorithm. WebPortugol[3], developed by the University of Vale do Itajaí, is an applet that allows users to check their algorithm using test data stored in a XML file. However, such feature is not well documented.

The referred environments provide the means to write and execute algorithms written in structured Portuguese. In fact, until recently, VisuAlg was the environment used by the students in the University of Caxias do Sul. However, none of them provides to students a set of problems to be solved and examples of input and output data for testing their solutions. Despite the limited documentation, only WebPortugol provides a tool for checking a solution for a problem. In addition, no environment offers an interface to track the students' progress.

3 The AlgoWeb Environment

AlgoWeb[4] is a web-based environment composed of four parts: a set of algorithmic problems, what we referred to problem portfolio, an algorithm editor/interpreter, a solution assessment tool, and a teacher interface. Figure 1 shows a screen of the AlgoWeb environment.

[1] http://www.apoioinformatica.inf.br/visualg
[2] http://orion.ipt.pt/~manso/Portugol/
[3] http://www.univali.br/webportugol
[4] https://vposeidon9.ucs.br

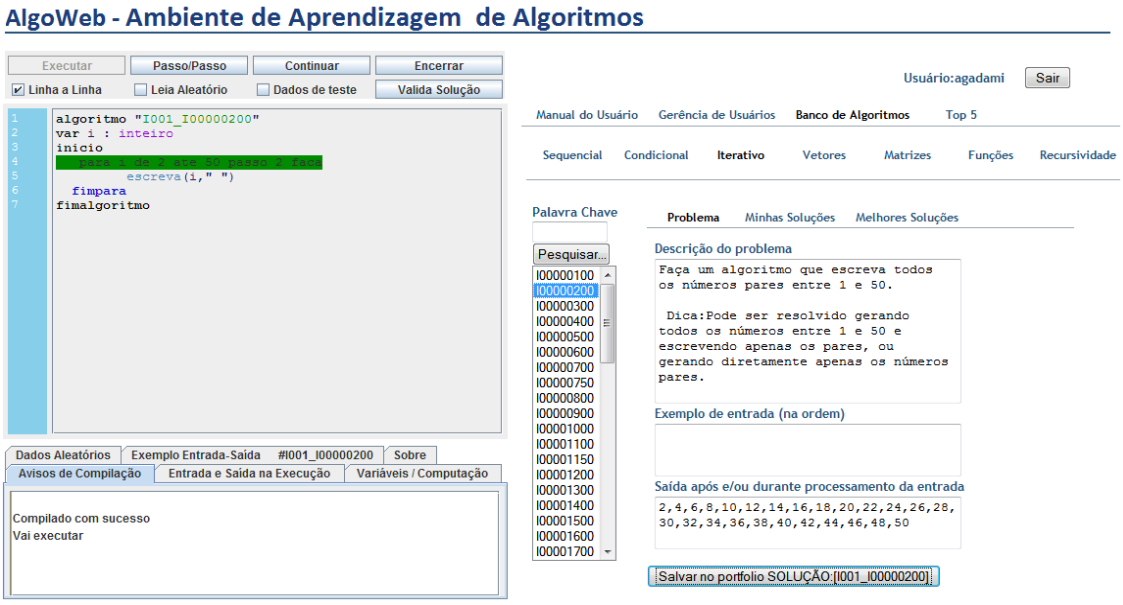

Figure 1. The AlgoWeb Environment. The algorithm interpreter is on the left and the algorithm portfolio and user management are on the right. An iterative problem is shown with the input/output examples.

AlgoWeb was developed using different languages according to their purpose in a client/server architecture. On the server side, the environment was developed using Python with Django framework. Django was used because of its rapid web development through the use of the Model-View-Controller (MVC) pattern. On the client side, the editor/interpreter is an applet developed using Java.

3.1 Problem Portfolio

The problem portfolio has a wide range of algorithmic problems that covers the entire syllabus of an introductory programming course. Each problem comes with some examples of input and output. Note that such information does not guarantee that the solution covered all the possibilities, but provides a small set of output data (and respective input data, when it is necessary) which the algorithm must produce after its processing.

The set of problems is divided into seven groups, according to the data/flow-control structures required to solve them:

1. 23 problems that can be solved without conditional or iterative structures;
2. 57 problems that require the use of conditional structures;
3. 80 problems that require the use of iterative structures;
4. 74 problems that require the use of one-dimensional array data structures;
5. 14 problems that require the use of two-dimensional array data structures;
6. 6 problems that require the use of functions or procedures;
7. 11 problems to be solved using recursive methods.

Inside each group, the problems are arranged from the simplest to the most complex solution. The set of problems is stored in a MySQL database.

Once a problem is selected, the environment presents a description of the problem, an example of input data (when required) and respective output data. In order to avoid solutions that produce only the output data presented to the user, each problem has a hidden set of input/output pairs that is used by the interpreter to check whether the algorithm is working.

3.2 Algorithm Editor/Interpreter

AlgoWeb has a syntax highlighting editor for structured Portuguese. While the algorithm is written, the font color is changed according to their function (commands become light blue, flow control structures become dark blue, data type become purple, and so on). This feature helps the users to identify mistyping or invalid name construction.

Once the algorithm has been written, the user can test it using the interpreter provided by the environment. Before running the code, the interpreter makes sure that the algorithm is lexically and syntactically correct. Error messages are issued according to the error found in the code.

There are four ways of running a code: direct, step-by-step, with breakpoints, or direct with timer. In the direct mode, the algorithm is fully executed without interruptions. In the step-by-step, the user controls the execution by executing each instruction (which is highlighted) every time a step button is pressed. In the execution with breakpoints, the user can place breakpoints (double clicking on the line) in the code that tells to the interpreter to stop processing at that point. In the direct with timer, each instruction is highlighted for half second during its execution. During certain modes of execution (step-by-step, breakpoints and direct with timer), the user can see the content of each variable in the algorithm through the Variable Watch tab.

AlgoWeb allows the user to generate data automatically for testing the solution by two ways: random data and test set. In the first way, the interpreter produces random data for each input required by the algorithm. Integer and real numbers can have their range configured in the environment. Strings are generated using a set of names from movies and cartoons. In the second way, the user can type the input data once and test several times without entering the data every run.

Since an algorithm can take some time to be written (or corrected), AlgoWeb allows registered users to save their ongoing work on the server. The authentication and account management interfaces were developed using HTML and Javascript.

The algorithm editor/interpreter was developed in Java. The reasons for choosing such language are:

1. Since an applet is executed in the client terminal, any problem in the algorithm (e.g., an infinite loop) does not compromise the server performance.
2. Applet can communicate with HTML (language used to develop the problem portfolio) through Javascript. Therefore, the communication between the interpreter and the problem portfolio is straightforward.

3.3 Solution Assessment

One of the major problems faced by the students is to identify a complete set of test data that can check the correctness of the algorithm. AlgoWeb provides the means for reducing such problem by making available some examples of input and output data for each problem, and a tool for checking whether the solution solves a problem from the problem portfolio. Such tool already exists for programming languages like Java (e.g., RoboProf (Daly, 1999)) and Scheme (e.g., BOSS (Joy and Luck, 1998)).

The user has to submit his or her solution of a given problem to be checked by the environment. First, the user must select a problem from the problem portfolio. Then, the user can write his or her solution for the problem. Finally, the user can submit his or her solution to the interpreter check it against the outputs from the problem portfolio for the respective problem. Since, all algorithms produce numerical outputs, the interpreter extracts the numbers from the algorithm output and compare to the correct outputs. This way, non-numerical symbols (like spaces, commas, letters and so on) do not interfere in the checking process. In case of success, the algorithm is ranked by the total number of operations (assignments, comparisons, and so on) for each one of the output examples.

A rank of the most efficient solutions (according to the number of operations) and their respective users are saved for each problem. The goal of this information is to motivate the students to improve the algorithm efficiency besides the efficacy.

The solution assessment was developed using Python with Django, a high-level web framework. The reason for using such framework is the rapid development with less code with an easy design.

3.4 Teacher Interface

Given that students use the environment, it is important that teachers can track student learning progress or can assess the problems faced by the students when writing algorithms. AlgoWeb provides a tool that allows the teacher to perform such task through the Teacher Interface, as shown in Figure 2. This interface (available only to teachers for obvious reasons) provides the following information: a list of users that solved (or just saved without checking the solution) a given problem, a list of problems solved (or just saved) by a given user, every saved version of an algorithm (with the respective number of executed operations, when checked as correct by the environment).

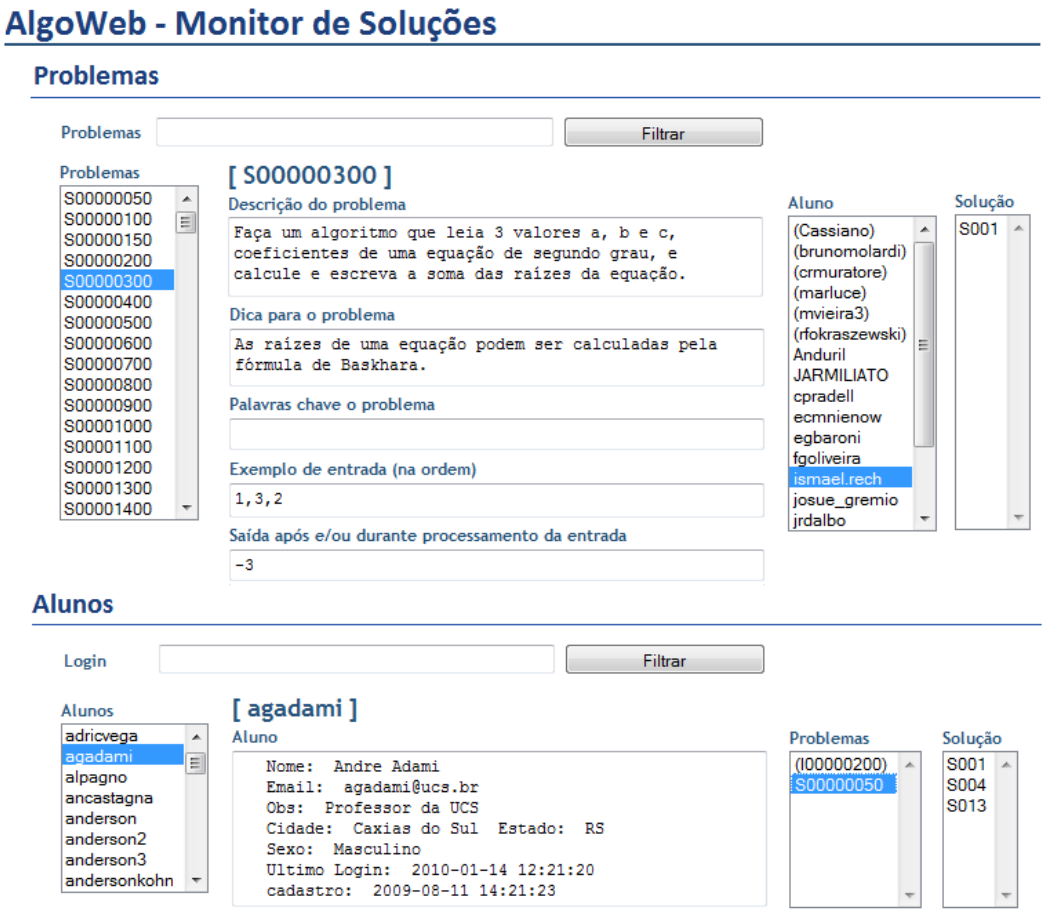

Figure 2. The Teacher Interface shows a list of users that solved a given problem and which problems were solved by a given user. In both cases, the interface can show the algorithms saved/submitted by the users.

4 Experimental Setup and Results

AlgoWeb has been used since 2009 in five introductory programming courses in the following undergraduate programs: Computer Science, Information Systems, and Production Engineering. In each course, AlgoWeb was used as a supporting tool for

distance learning, but the students could choose to use it or not during the take-home exercises. At the beginning of a semester, all courses had an introductory class on how to use the environment. The environment was not used in the remaining classes (algorithms were built using only pen and paper). Currently, there are about 40,000 algorithms saved in the database by 650 registered users.

Despite the lack of an actual measurement, all the involved teachers claimed that the students solved more problems with the help of AlgoWeb. Usually, four or five problems are solved per class, which results in around sixty problems worked in class in a 15-week semester. In addition, a couple of problems are given as homework. All the teachers reported that the students turned in more exercises than previous semesters.

The students reported that they were more motivated in solving the problems because of the feedback given by the environment. The easy to understand error messages, the execution capability, variable content monitoring and the possibility of checking the solution were the main features of the environment that helped them to successfully write correct algorithm.

The identification of the most efficient algorithms became an important motivational tool in the environment. Students were compelled to search for more efficient algorithms to beat the best ranked solution (according to the number of operations). This was noticed by several submissions of the same algorithm with decreasing number of executed operations (provided by the Teacher Interface).

Besides the identification of the most efficient solution for each problem, the environment presents also the rank of the students that solved correctly more problems in each category. Some students feel strongly motivated to appear in this rank. At this moment, the leader in this rank solved correctly 219 problems in all categories. This rank has been used to select candidates to be teaching assistants for the introductory programming courses. Furthermore, it makes easier to follow the progress of these students during the following classes (e.g. Programming, Object-oriented Programming). All students that we have monitored in the following semesters have had good performances in programming.

Until now, only one experiment was conducted using the environment as a support during exams. In that occasion, the students could use only the editor/interpreter to test their solutions, without accessing the database. The use of the editor/interpreter was optional, and in a class of 15 students, only one of them preferred not to use it.

5 Final Considerations

This paper described a web-based environment for learning introductory programming, called AlgoWeb. The environment has a syntax highlighting editor for structured Portuguese and a set of algorithmic problems that encompasses the concepts and techniques taught in introductory programming courses. Users can write, save, and execute in several ways their algorithms. In addition, AlgoWeb can check whether an algorithm solves a given problem.

The goal of such tool is to support distance learning and to reduce the student dependency on the teacher's feedback. Each interaction with the environment is logged by AlgoWeb, providing important information (e.g., number of solved exercises, last solved exercise, number of submissions per problem, the algorithm itself) about the student progress in class.

AlgoWeb was tested in five introductory programming courses in the following undergraduate programs: Computer Science, Information Systems, and Production Engineering. Students reported that they were more motivated to write algorithms because the environment provided a feedback when the teacher was not available. In fact, teachers claimed that the students turned in more exercises. This motivation could be verified by the considerable number of submitted algorithms (about 40,000 algorithms).

Indeed, learning tools such as AlgoWeb are a great supporting tool for introductory programming teachers. It is impractical for a teacher to manually check every algorithm and provide a feedback for classes with large number of students. Such burden can be reduced by the capability of checking automatically an algorithmic solution. This way, teachers can focus more on the real problems that the students are facing and students can work on many more problems because of the environment feedback.

Acknowledgment

We would like to thank all the supporting work done and valuable feedback to Vânius Gava. We also want to thank Cristian Koliver and Marcos Casa for sharing this tool with their students for evaluation.

References

GIANGRANDE, E. CS2 Programming Language Options, **Journal of Computing Sciences in Colleges**, v. 22, n.3, pp. 153-160, 2007.

WINSLOW, L. E. Programming Pedagogy - A Psychological Overview, **ACM SIGCSE Bulletin**, v. 28, n. 3, pp 17-25, 1996.

BENNEDSEN, J.; CASPERSEN, M.E., Failure Rates in Introductory Programming, **ACM SIGCSE Bulletin**, v. 39, n. 2, pp 32-36, Jun. 2007.

SOUZA, C.M. VisuAlg - Ferramenta de Apoio ao Ensino de Programação, **Revista TECEN**, vol. 2, , pp. 1-9, Sep. 2009.

HOSTINS, H.; RAABE, A. Auxiliando a Aprendizagem de Algoritmos com a Ferramenta WebPortugol, In: **Congresso da SBC - XV Workshop de Educação em Computação**, 23., 2007, pp. 96 – 105.

MANSOA.; OLIVEIRA, L.; MARQUES, C. Portugol IDE – Uma Ferramenta para o Ensino de Programação, In: **PAEE'2009 - Project Approaches in Engineering Education**. Guimarães, Portugal, 2009.

DALY, C. RoboProf and an Introductory Computer Programming Course, In: **Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education,** 4., 1999, Cracow, Poland. , 1999, pp 155-158.

JOY, M.; LUCK, M. Effective Electronic Marking for on-line Assessment, In: **Annual Conference on the Teaching of Computing**, 6., 1998, Dublin City University, Ireland, 1998, pp 134-138.