Teaching Algorithms for Visually Impaired and Blind Students using Physical Flowcharts and Screen Readers

Rodolfo M. Pereira*†, Felippe Fernandes da Silva‡, Carlos N. Silla Jr.*
*Intelligent Systems Laboratory – Pontifical Catholic University of Parana – Curitiba, PR, Brazil 80215–901
‡Federal Institute of Education, Science and Technology of Parana – Pinhais, PR, Brazil 83330–200
†Department of Informatics – State University of Maringa – Maringa, PR, Brazil 87020–900
Email: rodolfomp123@gmail.com, felippefernandes10@yahoo.com.br, carlos.sillajr@gmail.com

***Abstract*—This Innovative Practice Full Paper is grounded in the theme of teaching algorithms and programming to students with Special Educational Needs and Disabilities (SENDs) which is an increasingly difficult task for educators. The inclusion of SENDs in this context, makes the teaching task even more challenging, since in addition to the students' physical limitations, there is also the non-preparation of the teachers for specialized education. Among the SENDs, the students with visual impairment present one of the biggest obstacles in teaching algorithms. Considering the previously described context, this work proposes a method that uses Physical Flowcharts and Screen Readers in order to assist teaching algorithms to Visually Impaired students. To investigate the effectiveness of our method, a study was carried out in a Visually Impaired group of five students using the proposed method. The results of this study indicate that our method is promising and may be used in the future to help students with this kind of special needs.**

Keywords*—*Visually Impaired Students; Blind Students; Programming; Flowcharts.

I. Introduction

According to Loksa et al. [1], with the demand for software developers in the whole world, programming has become 21st century literacy. Together with the rise of coding needs, there is also a rise of the desire for learning how to code. Some countries around the world are even beginning to require coding classes in the high school *curriculum*. Loksa et al. [1] state that programming is the trend ability.

Teaching algorithms and programming is a difficult task for educators. In general, freshmen students have many difficulties to assimilate algorithms courses. According to Koliver et al. [2], one of the reasons for this problem is the students' unpreparedness to face a discipline whose main objective is to design logical solutions to generic problems. Wilson [3] emphasizes that the difficulty of adaptation may be because the students are used to "memorize" the contents of the classes.

When students with Special Educational Needs and Disabilities (SENDs) are included in these universities, the task of teaching algorithms becomes even more challenging. In addition to the students' physical limitations, there is also the problem of non-preparation of the professors for specialized education to students with these special conditions.

Among the SENDs, the visually impaired and blind students present one of the biggest obstacles of teaching algorithms. The evolution of computer accessibility over the past years produced several accessible mechanisms and applications to visually impaired and blind students, such as shown in [4], [5] and [6], notwithstanding teaching computer programming is still a challenging task.

Regarding the techniques to teach algorithms, the use of flowcharts is a method that presents series of advantages, mainly due to its potential to visualize the solution with figures, as stated in [7]. Nevertheless, since it is a purely visual method, the application of its potential to visually impaired and blind students is very limited.

Furthermore, there are a lot of screen readers software available, which were developed, in general, to help visually impaired and blind users to use ordinary computer tools. However, a large part of these screen readers are not able to correctly handle a programming IDE, since it is a tool that provides specific functionalities, such as code syntax coloring and step-by-step debugging.

This paper proposes an education method to assist the teaching of logic of algorithms and programming to visually impaired and blind students. The proposed method is divided into two mechanisms: (1) The use of physical flowcharts and (2) Screen Readers. The first part is focused on teaching the basic concepts of algorithms based mainly on the theory of flowcharts, in which its pieces were physically made, so the visually impaired/blind students can touch the pieces and identify the algorithms logic. The second part proposes the analysis and selection of a particular Screen Reader in conjunction with a programming IDE to assist the students to write and debug programs on the computer.

In order to investigate the effectiveness of our method, an experimental study was carried out in a visually impaired/blind group of five students. With the consent of the students, they were introduced to six two hours classes. Before each class, the students were submitted to an oral test with fourteen questions. These questions approached all the content that would be taught in the six classes. However, we expected that the students could answer correctly at least the questions which the content was taught in the previous classes.

The remainder of this paper is structured as follows. Section 2 presents the accessibility challenge. In Section 3 we present a discussion about the related work. Section 4 presents the method proposed in this paper. Section 5 shows the results and discussions related to the application of the

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

proposed method. And finally, in Section 6, we present the final conclusions and directions for future work.

II. The Accessibility Challenge

Accessibility is a dynamic process, associated not only with technological development, but mainly with the development of the society. The word accessibility can be defined as a concept that involves both aspects of the physical space: The space in which we live, and the digital space [8], [9].

According to the World Wide Web Consortium (W3C) [10], the Web is fundamentally designed to work for all people, whatever their hardware, software, language, culture, location, physical or mental ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive abilities. Thus the impact of disability is radically changed on the Web because the Web removes barriers to communication and interaction that many people face in the physical world. However, when websites, web technologies or web tools are badly designed, they can create barriers that exclude people from using the Web.

Regarding the accessibility groups, a group that deserves attention is the Visually Impaired. There are two types of visual deficits: Low vision and blindness. The classification has been made through visual acuity, in which blind people are the ones who have 20/200 vision in the best eye, and low vision people bearer who have 20/70 vision in the same conditions [11].

In addition to the speed in which new approaches emerge, a difficulty ensuring accessibility within the new paradigm is simply that there is a lack of experience concerning that paradigm. By definition, a new paradigm brings new challenges. Until people with disabilities use new technologies, and there is an opportunity to analyze what the barriers are and to gain experience with different solutions, it is unclear what the best solutions may be [12].

Concerning the technology field, there are algorithms and programming. During a computer science undergraduate course, for example, it is very difficult to think how to teach a totally visual discipline to a person with special visual needs. Here is the great challenge of this work: The teaching of algorithms for the visually impaired.

III. Related Work

Tools such as VisuAlg [13] and Tepequém [14] were implemented in order to develop logical reasoning and to teach algorithms in an undergraduate course, facilitating students comprehension. These tools can be adapted for teaching the visually impaired by reading software or screen magnifiers such as JAWS [15] and Window-Eyes [16]. What is observed is that these tools were not developed properly to assist the visually impaired. For this reason, adaptations must be made so that the software continues with its purpose.

Calder et al. [4] conducted a research that sought to help blind students to understand data structures and algorithms. The research consisted in the development of an extension to an earlier work on presenting graphs to users who are blind. The proposed system is called Plumb Extra and transmits the animation of algorithms using audio cues and synthesized speech. Although the framework was finished, the authors did not completed formal trials, having only a blind senior student of Computer Science to evaluate the system. According to the authors, the student successfully completed the animation with minimal prompting from the developers and informed he felt that the model was natural to use.

Capovilla et al. [5] proposes a method to introduce algorithmic thinking to Visually Impaired students using a haptic model[1]. The main idea is to use LEGO^{TM} pieces to teach algorithmic thinking by recreating situations and remaking the steps to find a solution. The method was evaluated in a group of five blind students, teaching them three basic search algorithms: linear search, binary search, and lookup in a binary search tree. According to the authors, the students were able to solve the proposed problems. The major limitation of the experiment is the small students' group size.

Papazafiropulos et al. [6] describes an approach to teach algorithms and data structures using haptic objects, based on the use of 3D printed models. In particular, the authors developed a form to represent data structures, i.e., haptic models for reproducing arrays and their application to algorithms. A short evaluation was conducted with a group of three blind students, who declared that the haptic models made the understanding of arrays and sorting algorithms much easier. The major limitation of the evaluation is also the small group of students.

According to Junior et al. [18], podcast is an alternative technology to support accessibility and teaching that allows the teacher to provide classes and didactic material in audio format, which can be heard at any place and time. This technology can also be useful for SENDs, such as blind and visual impaired students.

Ferreira et al. [19] also developed a series of podcasts called Podcast Algorithms and used them in an undergraduate class of Information Systems, specifically in the subject of Algorithms and Data Structures I. At the end of each class, the students were asked to listen to the podcasts, which intend to motivate them in the studies. The results indicate that 15% of the students strongly agreed that their motivation to study algorithms increased, 57.5% agreed that their motivation was improved in some way, while 22.5% declared themselves indifferent and 5% Disagreed with the motivations.

Considering the related work context, it is possible to verify that some research has already been developed to make teaching accessible to SENDs. Haptic and auditory methods were developed in order to include visually impaired students in the classes.

IV. Proposed Method

The developed method is divided into two parts: The use of physical flowcharts (1) and screen readers (2). The first part is focused on teaching the basic concepts of algorithms and is based mainly on the theory of flowcharts. The second proposes the use of screen readers to assist visually impaired students to write and debug programs on the computer. In this section we make a detailed explanation concerning both parts of the method.

[1]Haptic technology recreates the sense of touch by applying forces, vibrations or motions to the user [17].

A. Physical Flowcharts

According to Ascencio and Campos [20], flowcharts present a great advantage to teach algorithms due to its visualization potential of solutions using figures. By its turn, Kumar [7] describes a list of six advantages of flowcharts:

1) *Communication*: Flowcharts are the better way of communicating the logic of a system to all concerned.
2) *Effective Analysis*: With the help of flowcharts, problems may be analyzed in a more effective way.
3) *Proper Documentation*: A program flowchart may be used as a documentation, which is needed for various purposes.
4) *Efficient Coding*: The flowchart act as a guide or blueprint during the systems analysis and program development phase.
5) *Proper Debugging*: The flowchart helps in the debugging process.
6) *Efficient Program Maintenance*: It helps the programmer to put efforts more efficiently on a specific part of the problem/solution.

However, because it is a purely visual method, the previous described potentials cannot be applied to visually impaired and blind students.

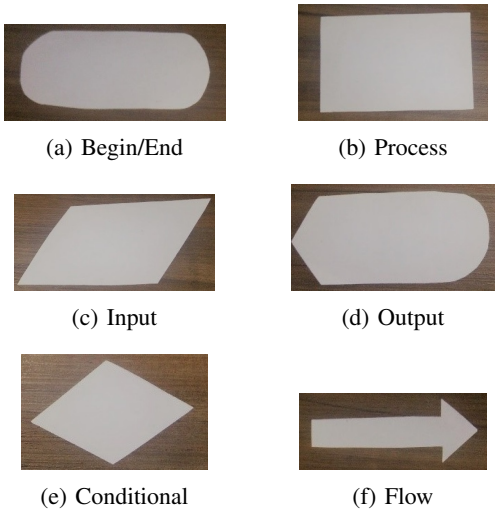
Aiming at the students' other senses, the flowchart symbols were adapted in physical pieces, so the students can use their touch to recognize them. A cardboard table with the pieces was created, in which the students could distinguish each of the flowchart symbols through the touch. The pieces had a small braille writing, indicating which algorithm symbol they represented.

Figure 1 shows the crafted pieces for the physical flowchart. To facilitate the students' sensitivity, the pieces were made in paperboard, which is thicker in relation to regular paper.

Algorithm 1 presents a pseudocode solution to resolve the problem of calculate the arithmetic average between two grades, printing "Pass" if the value is greater than 6.0, or "Fail" otherwise. In order to compare which one is easier to understand, Figure 2 shows a physical flowchart solution for the same problem. We can observe that with physical flowchart the solution is "viewable" and, in the case of the visual impairment students, it can be understood through touch.

Algorithm 1 Arithmetic Average

```
write "Enter the first grade:"
read grade_1
write "Enter the second grade:"
read grade_2
average ← (grade_1 + grade_2) / 2
if (average ≥ 6.0) then
    write "Student Passed"
else
    write "Student Fail"
end if
```



(a) Begin/End (b) Process (c) Input (d) Output (e) Conditional (f) Flow

Fig. 1: Symbols of physical flowchart developed.

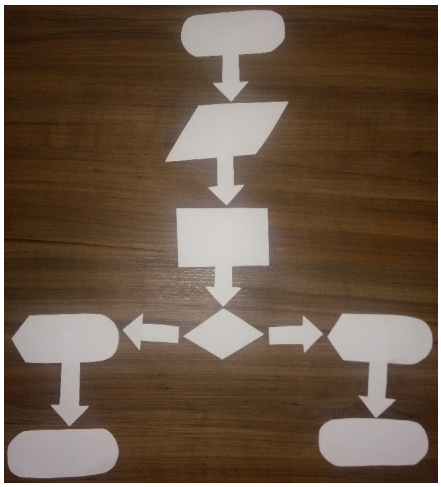


Fig. 2: Solution with physical flowchart.

B. Screen Readers

Following the sequence of steps in order to learn programming, after learn how the basic logic of algorithms works, the students need to apply these knowledge in real computer programs. Thus, this part of our research was aimed to choose a way to assist the visual impairment and blind students to write computer programs.

Simple modifications can be made to the computer display by accessing "Accessibility Options" in most of the Operating Systems (OS). In these options usually the mouse pointer and icons can be enlarged and the blink rate can be slowed down. Besides, computer monitor can be tilted to reduce visual glare. However, these modifications do not meet the needs of all students with visual impairments. The accessibility options do not modify the programs. For this reason a software, such as a Screen Reader, may be used to help the students to use other programs [21].

The screen readers can describe to users, through a speech synthesizer, the content of the screen that is displayed on

the monitor. In this kind of software it is possible for the user to choose different voice tones and different speeds, guiding the visually impaired according to their need. To use these softwares the students need to be instructed in how to turn voice on/off, make adjustments (speed, pitch, volume), navigate within a document, read (by characters, words, lines, sentences and paragraphs), and access the help file/manual to troubleshoot [21].

The Screen Readers DOSVOX [22], JAWS [15] and NVDA [23] were evaluated in order to find the best Screen Reader to be used together with a programming framework.

The first reader, DOSVOX, works fine with file manipulation. If the user wants to check a text document on the computer, it will open the document and describe what is on the screen. JAWS and NVDA describe the interface in details, allowing the manipulation of external softwares. All analyzed screen readers works in the background of the operating system.

In order to implement the algorithms structures, the programming language chosen was Pascal, since it was designed as a teaching language. Pascal has a Portuguese compiler named Pascalzim [24]. Since the screen reader was used together with the compiler, Pascalzim was chosen in order to provide to the visually impaired students a first contact with programming in their native language. Figure 3 shows the main screen of Pascalzim IDE with a Pascal solution to the problem of calculate the arithmetic average between two grades.

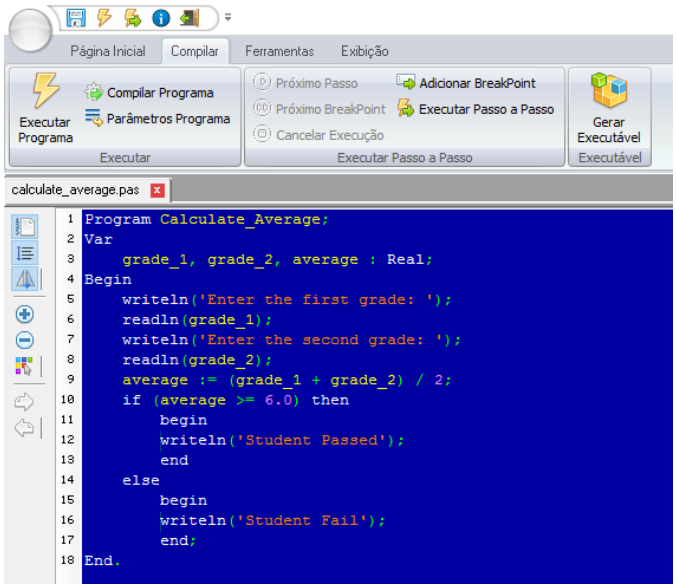


Fig. 3: Main screen of the Pascalzim IDE.

Analyzing the screen readers, we observe that DOSVOX does not access the Pascalzim interface, i.e., it is only possible to access the raw text document direct from the Operating System. For this reason the use of DOSVOX was discarded. We verified that both JAWS and NVDA had the same efficiency in terms of access to Pascalzim and, therefore, the students were able to choose between the two readers. Due to its friendly synthesizer voice pack, all the students preferred to work with JAWS.

Figure 4 shows the main screen of JAWS tool with its options and utilities menus. The software has a configuration that enables the adjustment of the speed in which the voice is reproduced. To use the screen reader, it is necessary that the student directs the arrows of the keyboard, guiding the reader through the text and, by default, JAWS reads the line where the cursor is positioned.

To run the program on Pascalzim IDE it is necessary to press the 'F9' key, which opens the console with the results and describes them. However, for a better functioning of JAWS, it is necessary to insert a *readkey()* function at the end of the programs. This function makes the program wait for an user input before return to Pascalzim's code edit screen. If this function is not added, at the end of the program, the code edit screen is opened and JAWS capture the compiler's version data rather than inform the results of a program.

V. Validation

To validate our proposed method we present the following content: The profile of the students that participated in the research will be discussed, explaining the criteria used to make the students selection (A); The evaluation method that was applied to the students to test their learning rate (B); and the results obtained from the tests followed by discussions concerning the students' performance (C).

A. Students Profile

Due to the fact that algorithms courses require a relatively high logical and mathematical knowledge of the students, for the experimental classes, we looked for students that completed or were senior in high school. We did not stipulate a minimum or maximum age for the students, and it is important to note that the students were aged between eighteen and fifty years.

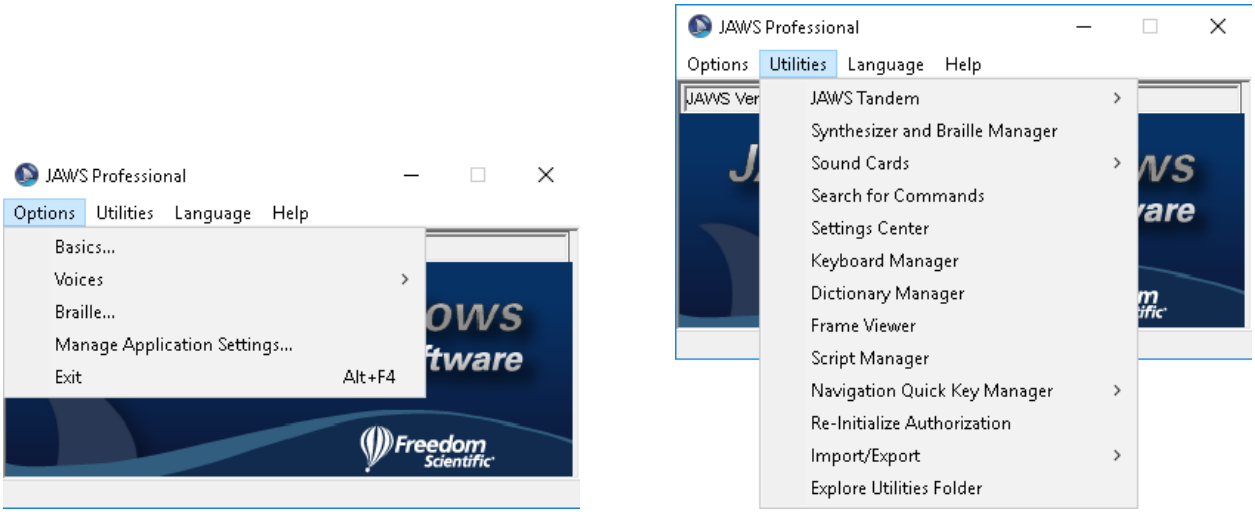
The degree of student's visual impairment was also not defined. In the beginning of our research, we were able to find a total of ten students that were interested to participate. Among these students, four of them have some level of low vision and six have total blindness.

Unfortunately, from the ten students who have committed themselves to contribute to the research, only five of them show up in any of the classes: three with total blindness and two with low vision. Most of the absent students had locomotion or health problems.

B. Evaluation Method

In order to evaluate the proposed method, we planned six 2 hours classes, totaling 12 hours of instruction. Table I presents the topics taught in the classes. The initial class was aimed at introducing the main concepts of hardware and software, while the following classes focused on the teaching of algorithms and programming using our method.

To evaluate the students' knowledge acquisition, a questionnaire test was applied on the beginning of each class, aiming to evaluate the content learned in previous class. Following the study of Butler [25], in which repeated testing produces superior transfer of learning than repeated studying, the same questionnaire was applied in all the six classes. These questions were answered by the students without consulting



(a) Options Menu.

(b) Utilities Menu.

Fig. 4: Main screen of JAWS.

TABLE I: Classes Cronogram.

Class Number	Subjects/Topics
1	Hardware and Software Concepts. Numerical Sets and Numbering Systems. Introduction to Programming Languages.
2	Basic Concepts of Algorithms. Definition of Data and Variables.
3	Flowchart Concepts. Concepts of Conditional and Repetitive Structures.
4	Examples of Programming with Physical Flowcharts. Introduction to Pascal Language.
5	Introduction to Softwares Screen Readers. Developments of exercises with conditional structures on the computer.
6	Development of exercises with repetitive structures on the computer.

any material or support. Furthermore, this evaluation method verifies whether the student's understanding is increasing as the classes are taught.

The questions can be seen in Table II. The test was applied orally and there are theoretical and practical tasks. While one of the practical tasks involves the recognition of physical flowchart pieces, other questions have been proposed in order to evaluate the students' potential to develop algorithms. In the Table, practical questions have the symbol (*) at the end of the question. The programming questions were applied on the computer. It is important to note that the physical flowchart pieces used in the recognition did not have the braille identification.

C. Results and Discussion

In each test, it was expected that the number of correct answers was greater than the previous test, since the students supposedly absorbed more knowledge from the class. It is important to note that in the beginning of the first class the questionnaire was also applied to the students. Unfortunately,

TABLE II: Questionnaire applied to the students.

Number	Question Description
1	What is Hardware?
2	What is Software?
3	What are Algorithms?
4	What are data input and output?
5	Which data types exists?
6	What is a flowchart?
7	Inspect and explain the flowchart pieces. Explain the operation of these pieces in a flowchart. (*)
8	Explain the IF/THEN selection structure.
9	Explain the FOR/WHILE repeat structure.
10	What is "Pascal Language"?
11	Cite some reserved words from Pascal Language.
12	Make an algorithm that calculates the sum of two numbers. (*)
13	Make an algorithm with a selection structure that, given a number, identify if it is even or odd. (*)
14	Make an algorithm with a repeat structure that, given two numbers A and B, while A is less than B, the algorithm adds 1 to the value of A. Check A to find out if it is smaller than B. (*)

one of the five students that were present in the first class did not showed up in any of the next classes, leading the experiment to a total of four students.

In the following we present Table III, which shows a rating of the students' answers. The values in the Table represent the correctness of each question per test, in which the responses can range from "0.0" (completely incorrect) to "1" (complements correct). Besides, the "Total" percentage, related to each test, is composed by the number of points achieved by the student divided by the total of possible points (14.0). With the purpose of presenting a different vision of the data from Table III, we also present two figures plotting the results: Figure 5, which presents a graphic that summarize the average success of the students per test, and Figure 7, which shows the students' success per test/question.

First of all, to sum up, we may note through Figure 5 that three of the students had a satisfactory performance, achieving around 80% of success in the last test. The only exception was student 4, which got only 54% of success. However as student 4 did not attended to classes 2 and 4, this result may also be considered as satisfactory.

In order to investigate the effectiveness of the proposed method, we have to mainly observe particular tests and questions. Concerning the understanding of flowcharts concepts and basic algorithms structures, we may observe the students' accuracies for questions 6 and 7 from test 4 and questions 8 and 9 from test 5. For the purpose of analyzing the capability to develop basic problems solutions using a Screen Reader, we may observe the students' answers for questions 10, 11 and 12 from test 5 and questions 13 and 14 from test 6.

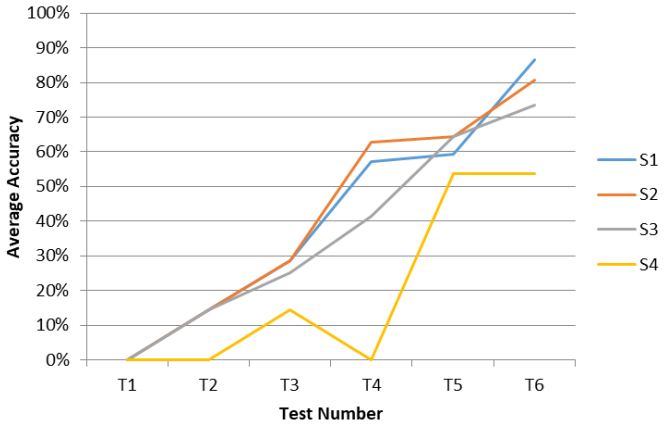


Fig. 5: Average accuracy per student/test.

Observing the results achieved by student #1, we can note his unfamiliarity concerning the subject in the first three tests. As expected, in the first test the student got 0% of correct answers, which also occurs with all the other students. In the second test, the student had enough knowledge to answer correctly the first two questions, totaling 14% of success rate. In the following classes, the level of difficulty increased, which led to a growth of only 1% of success rate between tests 4 and 5. However, analyzing the last test, we can observe that the student was able to absorb 81% of all the contents taught, which is a satisfactory result. It is interesting to note that, even though the student achieved satisfactory results in the questions concerning the basic algorithms structures using flowcharts and basic problems solutions using a Screen Reader, in test 5 the student answered incorrectly the questions 8 and 9, which he had fully scored in the previous test.

Looking at the results of student #2 and comparing his percentage of correctness with student #1, we can observe that both have very similar performances, obtaining the same results in the first three evaluations. Again, there is a leap of learning between classes 2 and 3, which does not happen between classes 4 and 5. However, it is noteworthy that student #2 obtained the best percentage of correct answers in the last test among all students, reaching 86% of success rate. It is also interesting to observe that, such as student #1, this student incorrectly answered questions 8 and 9 in test 5, even though they were fully scored in test 4.

Observing the results of student #3, we can note that the student did not show up in two classes, which affected his performance in tests 3, 5 and 6. Nevertheless, the student was able to answer correctly the first two questions in the third test, making him get a success rate of 14%. The same happened in the fifth test, the student did not attend the fourth class, but he was able to correctly answer the questions concerning the content of the third class, which made he get a success rate of 54%. These results can be justified by the fact that in every class, after the test, we made a revision about the content taught in the last class. Considering the student's frequency (66%), this student got reasonable results in the last test.

Student #4 got similar results compared to students #1 and #2. However, it is interesting to note that, unlike the other students, between the fourth and fifth tests, student #4 got a good learning rate, raising its performance in 23%. In the last test the student presented some difficulties in the development of the algorithms and was only able to solve exercises partially, which made his final accuracy a little bit lower than students #1 and #2. It is interesting to observe that, even though the student showed a learning growth between tests, he was unable to correctly answer question 10 in any of the tests, indicating that he did not learn about the programming language.

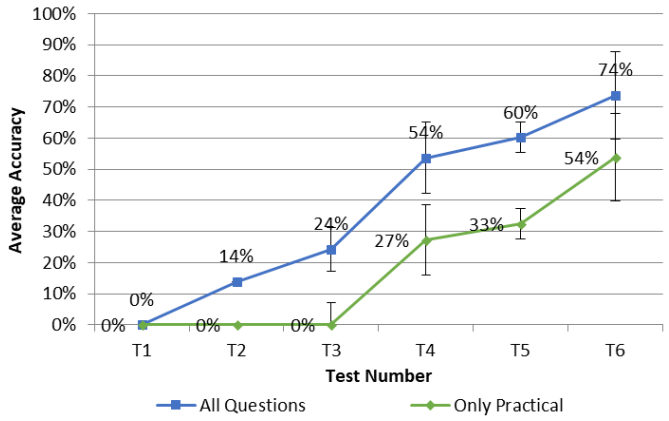


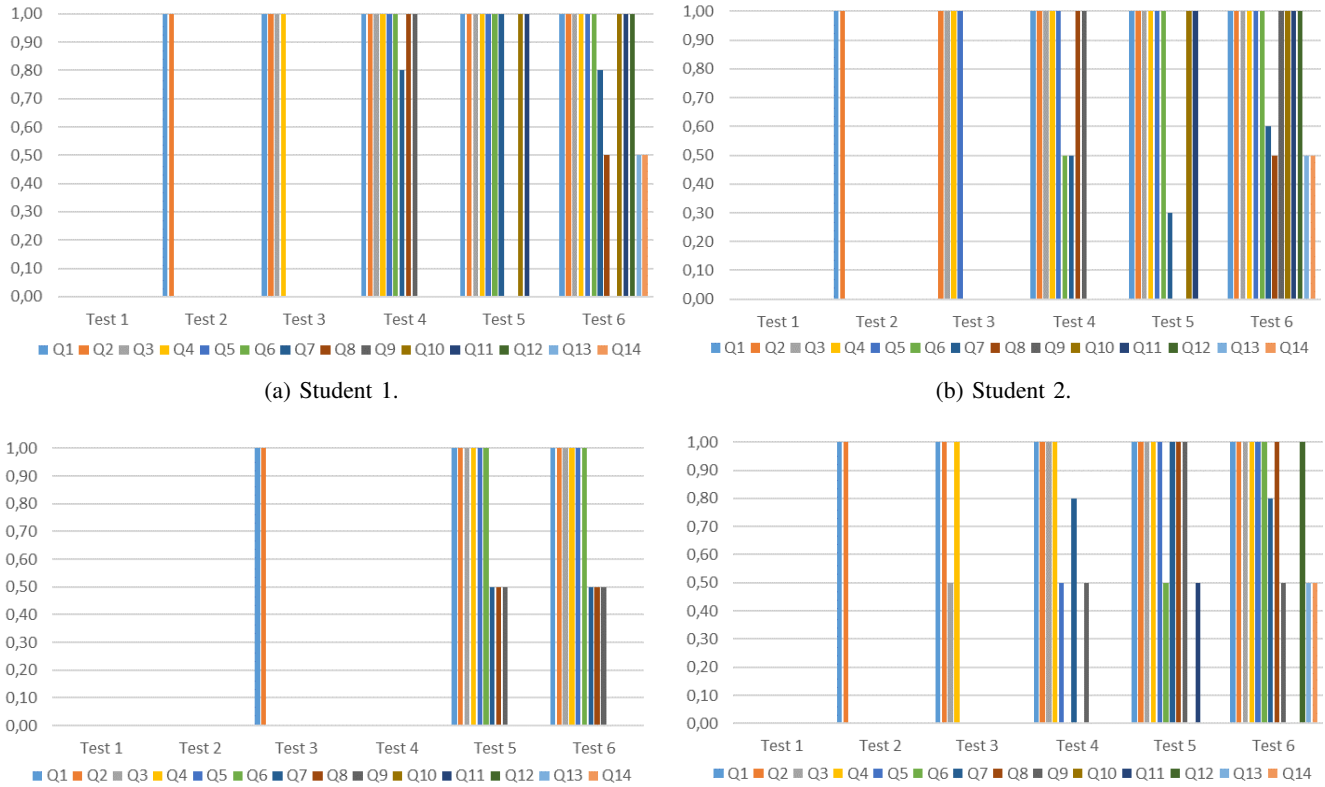
Fig. 6: Average accuracy of the students per test: All questions *versus* Only practical.

In order to analyze the impact of our method in the hole learning process and with respect to the students' understanding of the creation of algorithms, in Figure 6 we present the average accuracy of the students per test in two scenarios: All questions (1 to 14) and considering only the practical questions (7 to 14). It is important to observe that these graphics consider only the results of students who showed up in the particular class in which the test was applied, which means, for example, that the average accuracy of tests 2 and 4 does not consider student #4, and so on. We may observe that in both cases (all questions and only practical ones) we have almost a linear growth of the learning rate, which indicate that the students were in fact learning thought the classes. However, we may also note that this learning rate is considerable smaller in relation to the practical questions, since we have a learning growth up to 54% in the practical questions compared to 74% for all questions. These results indicate that our method is promising, insofar as the students are learning.

TABLE III: Individual students' success per test.

	Test 1				**Test 2**				**Test 3**				**Test 4**				**Test 5**				**Test 6**			
	S1	**S2**	**S3**	**S4**	**S1**	**S2**	**S3**	**S4**	**S1**	**S2**	**S3**	**S4**	**S1**	**S2**	**S3**	**S4**	**S1**	**S2**	**S3**	**S4**	**S1**	**S2**	**S3**	**S4**
Q1	-	-	-	-	1	1	N/A	1	1	-	1	1	1	1	N/A	1	1	1	1	1	1	1	1	1
Q2	-	-	-	-	1	1	N/A	1	1	1	1	1	1	1	N/A	1	1	1	1	1	1	1	1	1
Q3	-	-	-	-	-	-	N/A	-	1	1	-	.5	1	1	N/A	1	1	1	1	1	1	1	1	1
Q4	-	-	-	-	-	-	N/A	-	1	1	-	1	1	1	N/A	1	1	1	1	1	1	1	1	1
Q5	-	-	-	-	-	-	N/A	-	-	1	-	-	1	1	N/A	.5	1	1	1	1	1	1	1	1
Q6	-	-	-	-	-	-	N/A	-	-	-	-	-	1	.5	N/A	-	1	1	1	.5	1	1	1	1
Q7	-	-	-	-	-	-	N/A	-	-	-	-	-	.8	.5	N/A	.8	1	.3	.5	1	.8	.6	.5	.8
Q8	-	-	-	-	-	-	N/A	-	-	-	-	-	1	1	N/A	-	-	-	.5	1	.5	.5	.5	1
Q9	-	-	-	-	-	-	N/A	-	-	-	-	-	1	1	N/A	.5	-	-	.5	1	-	1	.5	.5
Q10	-	-	-	-	-	-	N/A	-	-	-	-	-	-	-	N/A	-	1	1	-	-	1	1	-	-
Q11	-	-	-	-	-	-	N/A	-	-	-	-	-	-	-	N/A	-	1	1	-	.5	1	1	-	-
Q12	-	-	-	-	-	-	N/A	-	-	-	-	-	-	-	N/A	-	-	-	-	-	1	1	-	1
Q13	-	-	-	-	-	-	N/A	-	-	-	-	-	-	-	N/A	-	-	-	-	-	.5	.5	-	.5
Q14	-	-	-	-	-	-	N/A	-	-	-	-	-	-	-	N/A	-	-	-	-	-	.5	.5	-	.5
Total	**0%**	**0%**	**0%**	**0%**	**14%**	**14%**	**N/A**	**14%**	**29%**	**29%**	**14%**	**25%**	**63%**	**57%**	**N/A**	**41%**	**64%**	**59%**	**54%**	**64%**	**81%**	**86%**	**54%**	**74%**

(*) S1, S2, S3 and S4 refer to the students' numbers, such as Q1, Q2, ..., Q14 refers to the questions.
(**) N/A is a short to "Not Applied" and it is used in the cases where the student did not showed up in the class.
(***) The "-" mark indicate that the student incorrectly answer the question.



(a) Student 1.

(b) Student 2.

(c) Student 3.

(d) Student 4.

Fig. 7: Individual students' success per test.

VI. Conclusions and Future Work

This work deals with the challenge of teaching of algorithms for the visually impaired and blind students. A teaching method was developed in which physical flowcharts and screen readers were used to present the content of algorithms and to improve the understanding and comprehension of the students. The proposed method was applied to a total of five students. After analyzing the results, we could observe that the students achieved an average score of 74% in the last test, indicating that the proposed method is promising to be used in the future. This satisfactory result may have been influenced by the use of repeated testing, which usually offers additional benefits in the learning process.

According to the students' reports, the method is didactic and dynamic, since it introduces the student to the reality of teaching algorithms using tangible symbols and screen readers, which significantly improves the understanding of the content being taught.

Some difficulties were found in the development of this work. We can consider as the main difficulty the frequency of students in the classes. This problem was verified early in the meetings, in which ten students were expected to show up, but only five attended the first class.

As limitations of the proposed method, we can cite the difficulty of transporting the pieces used to teach the physical flowcharts, since these pieces need to be carried out by the teacher in the first classes of the course. Another limitation is related to the screen readers, in which a software version upgrade can lead to compatibility problems when using the screen reader together with the programming framework.

As future work we suggest the application of the proposed method in classes with a larger number of students, so the results obtained can be better consolidated. In addition, physical flowchart pieces with different colors can be used to help students with low vision in the pieces recognition.

Acknowledgment

We thank the Brazilian Research Support Agencies: CAPES - Coordination for the Improvement of Higher Education Personnel, CNPq - National Council for Scientific and Technological Development and FA - Araucaria Foundation for their financial support. We would also like to thank the educational centers from Maringa-PR that helped us to look for students to complete our experimental classes.

References

[1] D. Loksa, A. J. Ko, W. Jernigan, A. Oleson, C. J. Mendez, and M. M. Burnett, "Programming, problem solving, and self-awareness: effects of explicit guidance," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 1449–1461.

[2] C. Koliver, R. V. Dorneles, and M. E. Casa, "Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos," in *Anais do XII Workshop de Educação em Computação*, 2004.

[3] G. Wilson, *How to Teach Programming (And Other Things)*, 1st ed. Lulu.com, 2017.

[4] M. Calder, R. Cohen, J. Lanzoni, N. Landry, and J. Skaff, "Teaching data structures to students who are blind," *ACM Special Interest Group on Computer Science Education Bulletin*, vol. 39, no. 3, pp. 87–90, 2007.

[5] D. Capovilla, J. Krugel, and P. Hubwieser, "Teaching algorithmic thinking using haptic models for visually impaired students," in *Proceedings of the Learning and Teaching in Computing and Engineering Conference*, 2013, pp. 167–171.

[6] N. Papazafiropulos, L. Fanucci, B. Leporini, S. Pelagatti, and R. Roncella, "Haptic models of arrays through 3d printing for computer science education," in *Proceedings of the International Conference on Computers Helping People with Special Needs*, 2016, pp. 491–498.

[7] A. Kumar, *Computer Basics with Office Automation*, 1st ed. I. K. International Publishing House, 2010.

[8] C. S. Fichten, J. V. Asuncion, M. Barile, V. Ferraro, and J. Wolforth, "Accessibility of e-learning and computer and information technologies for students with visual impairments in postsecondary education," *Journal of Visual Impairment & Blindness*, vol. 103, no. 9, p. 543, 2009.

[9] D. Weiss, A. Nelson, H. Gibson, W. Temperley, S. Peedell, A. Lieber, M. Hancher, E. Poyart, S. Belchior, and N. Fullman, "A global map of travel time to cities to assess inequalities in accessibility in 2015," *Nature*, vol. 553, pp. 333–336, 2018.

[10] S. Abou-Zahra, A. Arch, A. Chuter, S. Duchateau, J. Welsh, W. Loughborough, C. Roy, S. Rush, and Y. Yesilada, "Accessibility," W3C, 2018, access Date: 06 June, 2018. [Online]. Available: https://www.w3.org/standards/webdesign/accessibility

[11] M. Oliver, "Defining impairment and disability," *Disability and Equality Law*, p. 3, 2017.

[12] M. Cooper, "Accessibility of emerging rich web technologies: web 2.0 and the semantic web," in *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, 2007, pp. 93–98.

[13] C. M. de Souza, "Visualg-ferramenta de apoio ao ensino de programação," *Revista Eletrônica TECCEN*, vol. 2, no. 2, 2016.

[14] O. H. Junior, "Tepequém: uma nova ferramenta para o ensino de algoritmos nos cursos superiores em computação," in *Anais do XVII Workshop sobre Educação em Informática*, 2009.

[15] "Blindness solutions: Jaws," Freedom Scientific Incorporation, 2018, access Date: 06 June, 2018. [Online]. Available: http://www.freedomscientific.com/Products/Blindness/JAWS

[16] "Gw micro: Window-eyes," Ai Squared, 2018, access Date: 06 June, 2018. [Online]. Available: http://www.gwmicro.com/Window-Eyes/

[17] M. Page, "Haptic holography/touching the ethereal," in *Journal of Physics: Conference Series*, vol. 415, no. 1, 2013, pp. 12–41.

[18] B. Junior, J. Batista, Coutinho, and C. Pereira, "Podcast: uma ferramenta tecnológica para auxílio ao ensino de deficientes visuais," in *Proceedings of the Lusocom Conference*, 2009, pp. 2114–2126.

[19] C. Ferreira, J. Anjos, J. Normando, M. Castro, V. Odakura, R. Sacchi, and C. Barvinski, "Uso de podcast para apoio a aprendizagem de algoritmos em curso de graduação em computação," in *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, vol. 5, no. 1, 2016, p. 1208.

[20] A. F. G. Ascencio and E. A. V. Campos, *Fundamentos da programação de computadores: algoritmos, Pascal e C/C++*, 3rd ed. Pearson, 2012.

[21] "Screen magnification & readers," Carmen Willings, 2018, access Date: 06 June, 2018. [Online]. Available: http://www.teachingvisuallyimpaired.com/screen-enlargement–readers.html

[22] "Dosvox project," Federal University of Rio de Janeiro, 2018, access Date: 06 June, 2018. [Online]. Available: http://intervox.nce.ufrj.br/dosvox/

[23] "Nvda: Nonvisual desktop access," NV Access, 2018, access Date: 06 June, 2018. [Online]. Available: https://www.nvaccess.org/

[24] "Pascalzim: A pascal compiler," University of Brasilia, 2018, access Date: 06 June, 2018. [Online]. Available: http://pascalzimbr.blogspot.com.br/

[25] A. C. Butler, "Repeated testing produces superior transfer of learning relative to repeated studying." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 36, no. 5, p. 1118, 2010.