# IMPROVING STUDENTS' FOCUS IN INTRODUCTORY PROGRAMMING COURSES

KONECKI, M.

***Abstract:*** *Introductory programming courses struggle with low passing rates and low quality of knowledge that students possess even at the very end of their introductory programming courses. Abstract nature of programming makes students unmotivated and skeptic towards programming. Very soon after the first lectures students start to be confused and unable to follow the rest of the lectures which makes them unwilling to deal with their homework assignments. In order to try to resolve this problem and to increase students' focus in introductory programming courses a more dynamic approach is proposed which includes the usage of PTG (Programming Tasks Generator). PTG is designed to dynamically generate appropriate tasks for every particular student. In this way lectures become less predictable and students are encouraged to stay focused and also to do their tasks on their own. In this paper the results of using PTG are presented and elaborated.*

***Key words:*** *introductory programming courses, problems, focus, motivation, programming tasks generator*

**Authors´ data:** Dr. Sc. **Konecki**, M[ario], University of Zagreb, Faculty of Organization and Informatics, Pavlinska 2, 42000 Varazdin, Croatia, mario.konecki@foi.hr

**This Publication has to be referred as:** Konecki, M[ario] (2015). Improving Students' Focus in Introductory Programming Courses, Chapter 15 in DAAAM International Scientific Book 2015, pp.165-172, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-05-1, ISSN 1726-9687, Vienna, Austria
DOI: 10.2507/daaam.scibook.2015.15

## 1. Introduction

To say that programming can be hard for students is nothing unfamiliar, especially when talking about the introductory programming courses. The fact is that in most cases programming courses and teachers involved in educational process report about low results of their students' tests. Students experience various difficulties when trying to learn programming and the causes of these difficulties are very different from one case to another. This problem needs to be resolved because of the importance of programming in a modern information society. Programming is a basis for almost all business systems which are supported by various programs and information systems that make everyday activities fast and accurate enough to meet the requirement of large and complex demands of modern markets. Finding the answers to problems in programming courses and increasing the number of high quality programming experts is an important task that deserves the attention.

Students differ in their preferable way of learning and they differ in the way they perceive presented matter. The reasons of difficulties that students experience are not the same for all students and there are many factors that influence success rates of programming courses. All these factors need to be addressed and improving the results of programming courses requires constant and joint research efforts that will deal with a wide variety of aspects that can aid students but also their teachers in getting better results. One of the aspects that are needed in order for students to learn programming is motivation. To learn effectively and efficiently students have to be focused on the lectures and presented concepts. In this paper the difficulties that students experience in their programming courses are presented and elaborated. The proposed solution for improving focus of students in programming courses is also given and discussed.

## 2. Difficulties in programming courses

When considering the difficulties and problems in programming courses, many can be stated. The question is how motivated and interested students are at the time of enrolment in their programming courses. Various researches report that only a portion of students have a genuine interest in programming in the first place. One research reports that only 22% of the first year students enrolled programming course because they had any actual interest in it. 40% of students enrolled programming course because they thought it will be good for their career and 35% of students enrolled programming course because it was obligatory to do so (Bergin & Reilly, 2005). This indicates that one of the problems of programming courses is the fact that many students are simply not interested enough in programming. Interest affects motivation and motivation, especially intrinsic motivation, has been recognized as an important factor of students' success in programming courses (Nikula et al., 2011).

One of the problems of programming courses is also the fear that one generation passes to another generation which makes new generations of students skeptic towards programming. Along with negative reputation programming is also considered to be very abstract, hard to understand and hard to learn (Hanks et al., 2004; Jenkins, 2002; Peng, 2010; Robins et al., 2003). Students are simply not used to learning abstract

syntax or abstract concepts. Most of courses that students have had experience with in their previous education are of more familiar and concrete nature and are mostly based on memorization of facts (Gomes & Mendes, 2007). Another problem that students experience is increasing difficulty of presented programming concepts. As their programming course advances, students become more anxious and less motivated to learn because they tend to lose focus due to the fact that they get lost in more and more complex constructs and syntax. There is also a question of choosing a proper programming language that would suit introductory programming courses. Some authors claim that this kind of language doesn't exist and cannot be found (Smith et al., 2000). Programming as a course also requires a little bit different way of learning that students have to be aware of. Programming must be observed as a skill, because it essential is a skill and as such it requires constant efforts and a lot of practice in a prolonged time, which is something that many students are not used to. As a result, many students try to learn programming in a short period of time by memorizing programming examples, which enables them to remember the syntax but does nothing for their problems-solving skills (Lister et al., 2004) which are required to produce valid programming solutions.

Gomes and Mendes (Gomes & Mendes, 2007) give several reasons for difficulties that are experienced by students in their programming courses:

- Programming demands a high abstraction level
- Programming needs a good level of both knowledge and practical problem-solving techniques
- Programming requires a very practical and intensive study, which is quite different from what is required in many other courses (which are more based on theoretical knowledge, implying extensive reading and some memorization)
- Usually teaching cannot be individualized, due to the common classes size
- Programming is mostly dynamic, but usually taught using static materials
- Teachers' methodologies many times don't take into consideration students' learning styles
- Programming languages have a very complex syntax with characteristics defined for professional use and not with pedagogical motivations

Hawi (Hawi, 2010) reports on students' perception about the most important causes of problems in programming courses. In this research he has identified 10 causal attributions of such problems (Hawi, 2010):

- Learning strategy
- Lack of study
- Lack of practice
- Subject difficulty
- Lack of effort
- Appropriate teaching method
- Exam anxiety
- Cheating
- Lack of time
- Unfair treatment

There is obviously not one clear reason for difficulties experienced by students in their programming courses but all these reasons result in decrease of students' motivation and focus. When considering efforts that have been made in aiding students to understand programming, most of them are based on various visualizations (Baldwin & Kuljis, 2001; Hu, 2004; Sorva et al., 2013) which are aimed at making abstract programming concepts easier to imagine and comprehend. There are also many other approaches and recommendations that have been developed in order to help students in their programming courses. Jenkins gives several recommendations for programming courses (Jenkins, 2002):

- Programming should never be taught before the second year of any course
- The language used should be chosen for its pedagogic suitability and not because it is popular in industry
- Programming should be taught by those who can teach programming and not by those who can program
- Programming courses should be designed to be flexible in order to allow different students to learn in different ways
- There should be no summative (continuous) assessment to ease pressure on students
- Departments should acknowledge that programming is difficult and supply adequate support to students

When considering programming as a skill, it can be divided into three parts that need to be comprehended in order for a student to program properly. These three parts of programming knowledge are syntactic, conceptual and strategic knowledge (Baldwin & Kuljis, 2001). Syntactic knowledge is similar to knowledge about words and grammatical rules of a spoken language and it denotes knowing how to express something in some programming language. Conceptual knowledge denotes the knowledge about programming concepts and various programming principles which are necessary in order for students to develop a valid programming mental models. Strategic knowledge denotes the knowledge about problem analysis and problem-solving skills which are needed in order for students to be able to construct valid algorithms and working computer programs.

Students of programming courses are encountered with a challenge of getting used to a new way of reasoning which is not intuitive or known to them. This point of view is supported by many of already mentioned research results. Another set of recommendations for improving the situation and results in programming courses is given by Konecki (Konecki, 2014a):

- Introduce additional programming course prior to introductory programming course that would promote algorithmic way of thinking
- Increase motivation of students for learning programming
- Explain to students that programming is a skill, not merely knowledge
- Introduce elements of constructivism into teaching process
- Introduce learning by example
- Introduce animation and other visualization techniques combined with interaction
- Introduce interactive visual simulations
- Include support for multiple learning styles

A special set of difficulties comes with the education of students with some kind of disabilities. Visually impaired students for example require quite different approach and a suitable aiding technology, since they are not able to rely on visualization to clarify abstract programming concepts to them. GUIDL (Graphical User Interface Description Language) system (Konecki, 2014b) is one of aiding technologies that enables visually impaired students to learn programming and basic programming concepts, such as objects or their properties, in a simpler and more intuitive manner through creation of simple user interfaces.

Difficulties in programming courses are present and they cause students to feel anxious and skeptic towards programming. This is especially true for students of introductory programming courses. One of the reasons for students experiencing lack of motivation is losing focus, because students get overwhelmed and become unable to follow presented matter in a clear and understandable way, which in the end leads to worsening of their test results.

## 3. Programming tasks generator

In order to improve students' focus on their programming tasks and to make them more active, a PTG (Programming Tasks Generator) has been proposed and developed. In many cases students tend to rely on the work of their colleagues when talking about homework tasks and the same applies in the case of graded students' tests. In order to reduce this practice and to make students more focused on their work, the PTG provides students with individually generated programming tasks that are part of their homework and their graded tests. Another aspect of PTG is that it was designed to facilitate a greater number of smaller programming tasks that would enable students to learn in small steps and to be motivated with every successful step and solved task, which is something that lacks in the case of a larger and more complex programming tasks which are frequently used in programming courses.

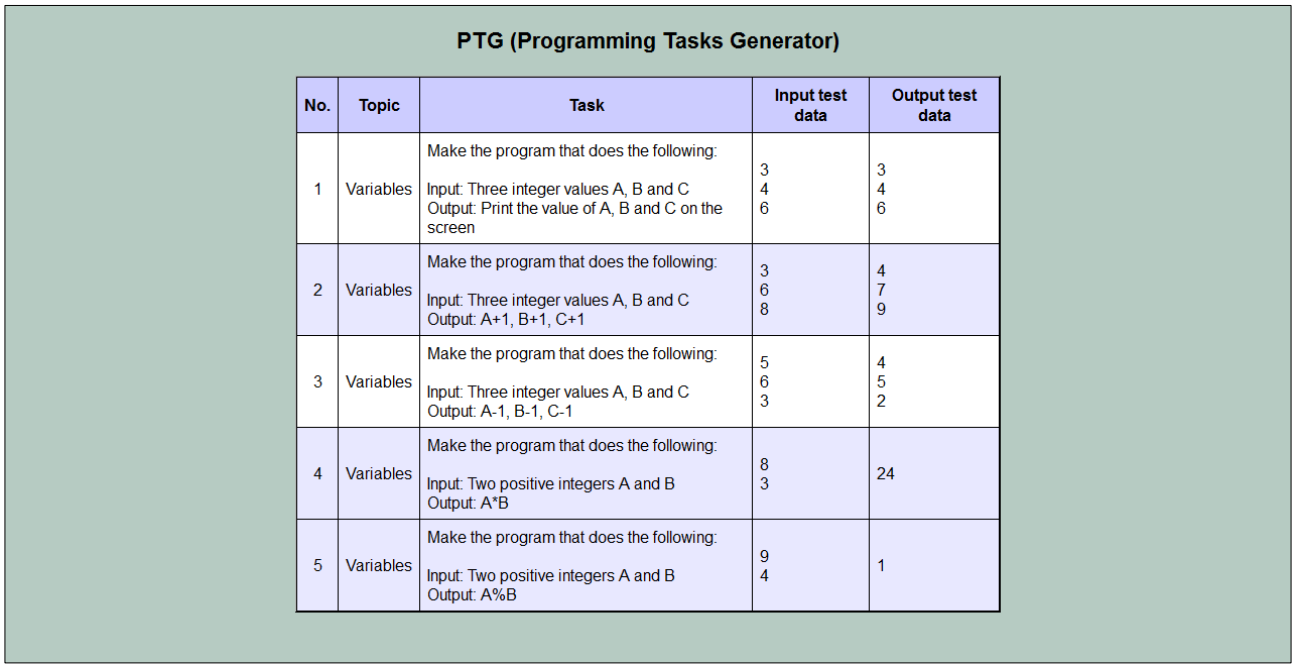
In this way students cannot get too relaxed because they are not able to depend on the work of other students. Instead, students are confronted with their own tasks which are different from the tasks of their peers and have to be solved by each of the students individually. This kind of approach makes students more involved in their work and as a result gives students more knowledge and a greater skill in the end.

PTG is a web-based system that is secured with several levels of protection which are needed in order for system to be reliable. For example, system has time-limited functionality and it is available only in a limited amount of time which is defined in the system before its use. These time intervals which are defined by date, start time and by end time ensure that the system will not be used outside of the classroom, in the case of graded tests. System also has disabled functionality of right mouse click, certain keys on a keyboard, automatic locking of the system after three false sign in attempts, regular backups, etc.

PTG is available only through a special container that is installed on a computer and has credentials that enable it to access PTG web service. In the background of a PTG is a database with various programming tasks which are grouped by topics. Teacher is able to define the date and time schedule of switching the topics so that

students get the appropriate programming tasks depending on the week of lectures in which they are in some point. Another feature of PTG is the limitation of time that is allowed for tasks from a particular topic. As the first step of using PTG, students are required to sign in with their student number and their full name.

After the students sign in they are presented with individual programming tasks which means that students are not able to rely on their surrounding colleagues but instead they have to do their own tasks alone. In this way students become more focused and more involved in their programming course. Generated tasks in PTG are shown in Fig. 1.



**PTG (Programming Tasks Generator)**

| No. | Topic | Task | Input test data | Output test data |
|---|---|---|---|---|
| 1 | Variables | Make the program that does the following:<br><br>Input: Three integer values A, B and C<br>Output: Print the value of A, B and C on the screen | 3<br>4<br>6 | 3<br>4<br>6 |
| 2 | Variables | Make the program that does the following:<br><br>Input: Three integer values A, B and C<br>Output: A+1, B+1, C+1 | 3<br>6<br>8 | 4<br>7<br>9 |
| 3 | Variables | Make the program that does the following:<br><br>Input: Three integer values A, B and C<br>Output: A-1, B-1, C-1 | 5<br>6<br>3 | 4<br>5<br>2 |
| 4 | Variables | Make the program that does the following:<br><br>Input: Two positive integers A and B<br>Output: A*B | 8<br>3 | 24 |
| 5 | Variables | Make the program that does the following:<br><br>Input: Two positive integers A and B<br>Output: A%B | 9<br>4 | 1 |

Fig. 1. Generated tasks in PTG

All generated tasks have their number, topic that represents a theoretical basis for the task, the text of the task and input as well as output test data. This data are very useful because they give the student a mean of being sure the written code produces valid results, according to the given task. PTG enables students to open a C++ coding window and to compile their programs. After finishing and testing their tasks, students are required to upload their solutions in the PTG system. PTG has been used and tested during one academic year in introductory programming course. 285 students have been using the system as a platform for their programming tasks during one semester. In order to assess the usefulness and effectiveness of PTG, 35 random students have been interviewed and asked to state their experience with PTG. Students have reported that they found PTG to be useful and that it has improved their focus since they simply had to deal with their programming tasks. Students have also stated that smaller and simpler tasks were much less overwhelming and that they found this way of learning to be more motivated compared to more complex and larger programming tasks. They also suggested that it would be even better to have programming tasks that are of more life-situations character, compared to logical and mathematical tasks that were in most cases given to students.

## 4. Conclusion

Programming is both important and challenging to learn and teach, especially in the case of introductory programming courses. Students are experiencing difficulties in their programming courses and this situation often results in decrease of their motivation to further deal with programming. Programming is complex and abstract and it is hard to learn but the motivation of students to learn or the lack of their motivation to learn is also an important factor that affects success rates of programming courses. Many approaches and tools, mostly based on various visualizations, have been developed as an aiding means to help students understand programming in an easier way. In practice, existing aiding tools usage depends on the teachers and institutions they belong to. Abstract nature of programming also causes students to lose focus as their programming courses advance.

In order to improve students focus in programming courses the PTG (Programming Tasks Generator) has been developed. PTG has been designed to provide students with individually generated programming tasks that are designed to improve their focus and increase the amount of their individual work. This kind of approach has been designed to improve the results of programming courses and to increase students' acquired programming skill. PTG has been tested, as a part of introductory programming course, and was perceived as a useful and effective aiding mean in students' programming education. Further development of PTG and more detailed research about the attitude and perception of students about PTG as well as detailed comparison of the usage of PTG with a more traditional approach in programming education will be a part of future research.

## 5. References

Baldwin, L. P. & Kuljis, J. (2001). Learning programming using program visualization techniques. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Sprague, R. H., Jr. (Ed.), pp. 1051-1058, ISBN 0-7695-0981-9, Maui, Hawaii, USA, January 3rd - 6th, IEEE, Washington, DC, USA

Bergin, S. & Reilly, R. (2005). The influence of motivation and comfort-level on learning to program. *Proceedings of the 17th Annual Workshop on the* Psychology *of Programming Interest Group*, Douce, C. (Ed.), pp. 293-304, University of Sussex, Brighton, UK, June 28th - July 1st, Psychology of Programming Interest Group, UK

Gomes, A. & Mendes, A. J. (2007). An environment to improve programming education. *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, Rachev, B.; Smrikarov, A. & Dimov, D. (Eds.), Art. No. 88, pp. 1-6, ISBN 978-954-9641-50-9, Rousse, Bulgaria, June 14th - 15th, ACM, New York, USA

Hanks, B.; McDowell, C.; Draper, D. & Krnjajic, M. (2004). Program quality with pair programming in CS1. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Boyle, R. (Ed.), pp. 176-

180, ISBN 1-58113-836-9, Leeds, United Kingdom, June 28th - 30th, ACM New York, NY, USA

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education*, Vol. 54, No. 4, May 2010, pp. 1127-1136, ISSN 0360-1315

Hu, M. (2004). Teaching novices programming with core language and dynamic visualization. *Proceedings of the 17th NACCQ*, Mann, S. & Clear, T. (Eds.), pp. 94-103, ISBN 0-476-00726-7, Christchurch, New Zealand, July 6th - 9th, NACCQ, Hillcrest, Hamiliton, New Zealand

Jenkins, T. (2002). On the difficulty of learning to program. *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, pp. 53-58, ISBN 0-9541927-1-0, Loughborough, UK, August 27th - 29th, ICS Subject Centre, Ulster, Ireland

Konecki, M. (2014a). Problems in programming education and means of their improvement, Chapter 37 in *DAAAM International Scientific Book 2014*, pp. 459-470, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-90150998-8, ISSN 1726-9687, Vienna, Austria

Konecki, M. (2014b). GUIDL as an Aiding Technology in Programming Education of Visually Impaired. *Journal of Computers*, Vol. 9, No. 12, December 2014, pp. 2816-2821, ISSN 1796-203X

Lister, R.; Adams, E. S.; Fitzgerald, S.; Fone, W.; Hamer, J.; Lindholm, M.; McCartney, R.; Moström, J. E.; Sanders, K.; Seppälä, O.; Simon, B. & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, Vol. 36, No. 4, December 2004, pp. 119-150, ISSN 0097-8418

Nikula, U.; Gotel, O. & Kasurinen, J. (2011). A motivation guided holistic rehabilitation of the first programming course. *ACM Transactions on Computing Education (TOCE)*, Vol. 11, No. 4, November 2011, Art. No. 24, ISSN 1946-6226

Peng, W. (2010). Practice and experience in the application of problem-based learning in computer programming course. *Proceedings of the International Conference on Educational and Information Technology (ICEIT)*, Yuting, L. (Ed.), Vol. 1, pp. 170-172, ISBN 978-1-4244-8033-3, Chongqing, China, September 17th - 19th, IEEE, Piscataway, New York, NY, USA

Robins, A.; Rountree, J. & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Journal of Computer Science Education*, Vol. 13, No. 2, April 2003, pp. 137-172, ISSN 0899-3408

Smith, D. C.; Cypher, A. & Tesler, L. (2000). Programming by example: novice programming comes of age. *Communications of the ACM*, Vol. 43, No. 3, March 2000, pp. 75-81, ISSN 0001-0782

Sorva, J.; Karavirta, V. & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, Vol. 13, No. 4, November 2013, Art. No. 15, ISSN 1946-6226