Journal of Computer Science

Original Research Paper

Novice Programmer = (Sourcecode) (Pseudocode) Algorithm

[1]Budi Yulianto, [2]Harjanto Prabowo, [3]Raymond Kosala and [4]Manik Hapsara

[1]Computer Science Department, School of Computer Science,
Bina Nusantara University, Jakarta, Indonesia 11480
[2]Management Department, BINUS Business School Undergraduate Program,
Bina Nusantara University, Jakarta, Indonesia 11480
[3]Computer Science Department, Faculty of Computing and Media,
Bina Nusantara University, Jakarta, Indonesia 11480
[4]Computer Science Department, BINUS Graduate Program - Doctor of Computer Science,
Bina Nusantara University, Jakarta, Indonesia 11480

Article history
Received: 7-11-2017
Revised: 14-01-2018
Accepted: 9-04-2018

Corresponding Author:
Budi Yulianto
Computer Science Department,
School of Computer Science,
Bina Nusantara University,
Jakarta, Indonesia 11480
Email: jibril_budi@yahoo.com

Abstract: Difficulties in learning programming often hinder new students as novice programmers. One of the difficulties is to transform algorithm in mind into syntactical solution (sourcecode). This study proposes an application to help students in transform their algorithm (logic) into sourcecode. The proposed application can be used to write down students' algorithm (logic) as block of pseudocode and then transform it into selected programming language sourcecode. Students can learn and modify the sourcecode and then try to execute it (learning by doing). Proposed application can improve 17% score and 14% passing rate of novice programmers (new students) in learning programming.

Keywords: Algorithm, Pseudocode, Novice Programmer, Programming Language

Introduction

Programming language is a language used by programmers to write commands (syntax and semantics) that can be understood by a computer to create a program (TechTerms, 2011). The development of today's technology has encouraged the public interest to learn programming that becomes the prospect and business opportunity (Microsoft, 2015). Based on Developers Survey in 2015 (Stack Overflow, 2015), most of them study programming languages by self-learning or autodidact (41.8%), bachelor in CS (37.7%), magister in CS (18.4%), work training (36.7%), industry certification (6.1%), boot camp (3.5%), doctoral in CS (2.2%), mentorship program (1.0%) and others (4.3%).

Garner research shows that there are 3 obstacles or difficulties in learning programming (Shuhidan *et al*., 2011). First is errors in writing syntax like missing semicolon and curly bracket. Second is difficulties in understanding and designing a program. Third is difficulties in understanding the basic structure of a program. These three difficulties often hinder new student as a novice programmer to learn programming (Layona *et al*., 2017).

Novice programmers need some tools to overcome those difficulties (Yulianto *et al*., 2016a; Yulianto and Prabowo, 2017). This also applies to early semester students in some universities that are new to programming and have not mastered it. In addition, some universities (especially in rural areas or with limited budget) do not have tools that can help their new students in learning programming (Yulianto *et al*., 2016b).

Previous researches shown that there are several educational tools to support students in starting learning programming (Yulianto *et al*., 2013). All are operated in offline, online, or both ways (Brandão *et al*., 2012). Some popular applications are Scratch, Alice, Blockly and Pencil Code (Ebrahimi *et al*., 2013; Bau *et al*., 2015). Scratch can help novice programmers to learn algorithm by using animation or game (Brandão *et al*., 2012; Ebrahimi *et al*., 2013). Alice can improve novice programmers' motivation in learning programming by using 3D concept and its interactivity. Blockly and Pencil Code implement block of pseudocode to help novice programmers in transforming their algorithm into pseudocode (Bau *et al*., 2015). Block of pseudocode is a concept of presenting pseudocode in a block-based rather than in text-based (Weintrop, 2015). Unfortunately, those applications don't generate the pseudocode into sourcecode to let novice programmers try to modify and learn from it.

Pears and Jordine apply game tool named RoboCode to help their students in sharpening logic (Pears *et al*., 2007; Jordine *et al*., 2014). RoboCode is a Java and .NET programming game for developing a robot tank to

battle against other tanks. The robot battles are running in real-time and on-screen (http://robowiki.net). Hundhausen and Vivian use LightBot to let their students solve a puzzle by using programming logic (Hundhausen *et al*., 2009; Vivian *et al*., 2014).

Other tools have been applied to help their students, such as Jeroo (Pears *et al*., 2007), Kodu (Sentance and Schwiderski-Grosche, 2012), Pencil Code (Vivian *et al*., 2014), Cargo-Bot and Move the Turtle (Grantham, 2011). Jeroo is an effective classroom-tested tool that helps novices learn fundamental concepts of object-oriented programming (www.jeroo.org and http://home.cc.gatech.edu/dorn/jeroo). Kodu lets students to create games on PC and Xbox via a simple visual programming language (https://www.kodugamelab.com). Pencil Code is used for learning professional programming languages by using an editor that lets students work in blocks (https://pencilcode.net/). Cargo-Bot is a puzzle game to teach a robot how to move crates (https://twolivesleft.com/CargoBot/). Move the Turtle teaches students the basics of programming (http://movetheturtle.com/). Unfortunately, those tools don't provide a feature of conversion to programming language source code to lets students modify and execute it in real environment (learning by doing).

Based on that problems, this study proposes an approach by using application for new students to convert their algorithm in mind into block of pseudocode, then generate it into a selected programming language (sourcecode). After that, students can try to modify and execute the generated sourcecode (learning by doing) (Yulianto *et al*., 2017). By this method, students can overcome the difficulties in transforming their algorithm into programming language.

Converting a data type to another in a programming language is called 'type casting' and symbolized with round brackets '()'. That's why this study is titled "Novice Programmer = (Sourcecode) (Pseudocode) Algorithm", which means converting algorithm in mind into pseudocode and then into sourcecode for novice programmers.

Data Collection

Software development method used in this study is Rational Unified Process (RUP). This study is not focusing in RUP discussion since it's a common method and easily to be learned. Data collection is gathered by distributing questionnaire to 129 students. Questionnaire results will be a foundation in developing the purposed application. Based on the results, most respondents learn C or C++ in early semester (Table 1). After early semester, most respondents learn PHP, PL/SQL, HTML/JavaScript, Java, or C# in next semesters. Most respondents recommend C, C++ and Java as first programming languages to be learned by new students (Fig. 1). This application provides C, C++ and Java to be selected by students when converting pseudocode into a programming language. Students can select which programming language to be used.

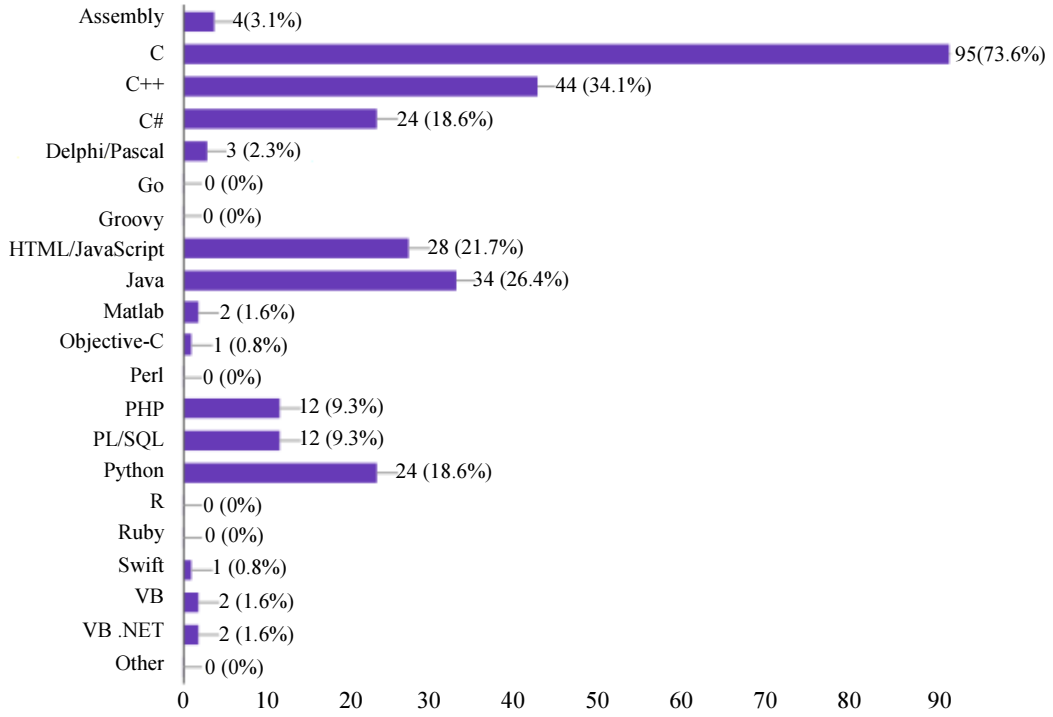


Fig. 1: First programming languages recommendation

Table 1: Programming language learned in early and next semester

Prog. Lang.	1	2	3	4	5
C	50	64	2	11	2
C++	45	38	9	34	3
C#	32	5	41	29	22
Delphi / Pascal	7	15	6	11	90
HTML / JavaScript	29	8	65	16	11
Java	27	16	42	35	9
PHP	12	4	80	8	25
PL/SQL	14	7	78	12	18
Python	9	15	23	18	64
VB	11	11	17	18	72
VB.NET	5	6	29	12	77

[1]Learn in the early semester and continue
[2]Learn in the early semester but not continue
[3]Learn in next semester (after first programming language) and continue
[4]Learn in next semester (after first programming language) and not continue
[5]Never learn

Table 2: Common mistakes in writing sourcecode

Mistake	1	2	3	4
Statement operator *e.g., { } ( ) [ ] ; .*	62	13	54	0
Assignment and arithmetic operator *eg., = + - / * % ++ --*	57	8	64	0
Relational Operator *e.g., <<= >>= == !=*	68	13	48	0
Logical operator *e.g., && || ^ !*	81	18	30	0
Data type and variable	66	14	49	0
Input/output	57	17	55	0
Selection *e.g., if-else, switch case*	62	15	52	0
Repetition *e.g., for, do-while, while*	70	20	39	0

[1]Often happens at the beginning, but less in next
[2]Often happens at the beginning and next
[3]Rarely happens at the beginning
[4]Never happen

Table 3: Features to be developed

Features	Respondents
Save and open (pseudocode) project	62
Copy, edit, delete the sequence or logic	47
Undo/redo	56
Error message	87
Convert pseudocode to sourcecode	86
Pseudocode templates	71
Others	9

Half of respondents often make more errors at the beginning of learning programming, but make less in the next learning. Another half respondents make less errors at the beginning of learning (Table 2). This means that they can overcome the errors along the time of learning. However, this application will provide features to help student in minimizing those errors.

Table 4: Chapter number of topic in text book

Topics	1	2	3	4	5	X
Input/Output	2	2	3	2	2	2.2
Variable	2	2	3	2	2	2.2
Assignment	2	2	3	2	2	2.2
Data type	3	2	4	3	2	2.8
In.& decrement	4	3	5	4	2	3.6
Logical operator	4	3	5	4	2	3.6
Selection	4	3	5	4	4	4.0
Repetition	5	4	6	5	4	4.8
Function	6	5	7	6	6	6.0
Array	7	6	7	7	3	6.0
Pointer	8	7	-	-	-	7.5

Some features that are needed by novice programmers to help them in learning programming are also listed on Table 3. Topics are sorted based on chapters in text books (Table 4). Basic topics (top 5 from average, symbolized with 'X') will be converted as application features in creating block of pseudocode. This average method is used to rank which ones to be basic topics based on smaller score.

Proposed System

System is proposed as a web-based application. Main window is divided in 4 sections (Fig. 2). First section (top left) provides students to create new project or open existing one (on Project tab). Students can also choose a provided pseudocode template, so they don't need to start from blank. On Program Control tab, students can declare variable, add input/output, selection, repetition and assignment statement. Second section (top right) provides tutorial (PDF type) for students. So, they can choose and read the tutorial during creating the block of pseudocode. Third section (bottom left) displays the results of block of pseudocode. Students can also edit, move and delete block of pseudocode and generate it into sourcecode of a selected programming language. Last section (bottom right) will display generated source code and students can download it. There are also options to generate block of pseudocode into textual pseudocode form (downloadable as a text file) or flowchart image (downloadable as a JPG on Flowchart tab on third section).

When students create a repetition (for) statement, a pop-up window will be displayed for step 1 (variable source). Student selects a variable source (from existing code, or creating new one), data type, variable name and value (Fig. 3). Student can select variable value from existing variable or insert new value (custom). On step 2 (condition), student selects a variable to be compared to another variable or a value. Student can also add additional conditions in this step. On step 3 (increment/decrement), student selects a variable and set as increment or decrement. Student can go back to previous step. After all steps done, block of pseudocode is displayed. Student can generate it into a programming

language. It is also able to create a 'do-while' or 'while' statement besides 'for'.

When student creates a text (output) statement, a pop-up window will be displayed (Fig. 4). Student inputs a text to be displayed and can be combined with ASCII code or escape sequence. After that, block of pseudocode is displayed and student can generate it into a programming language.

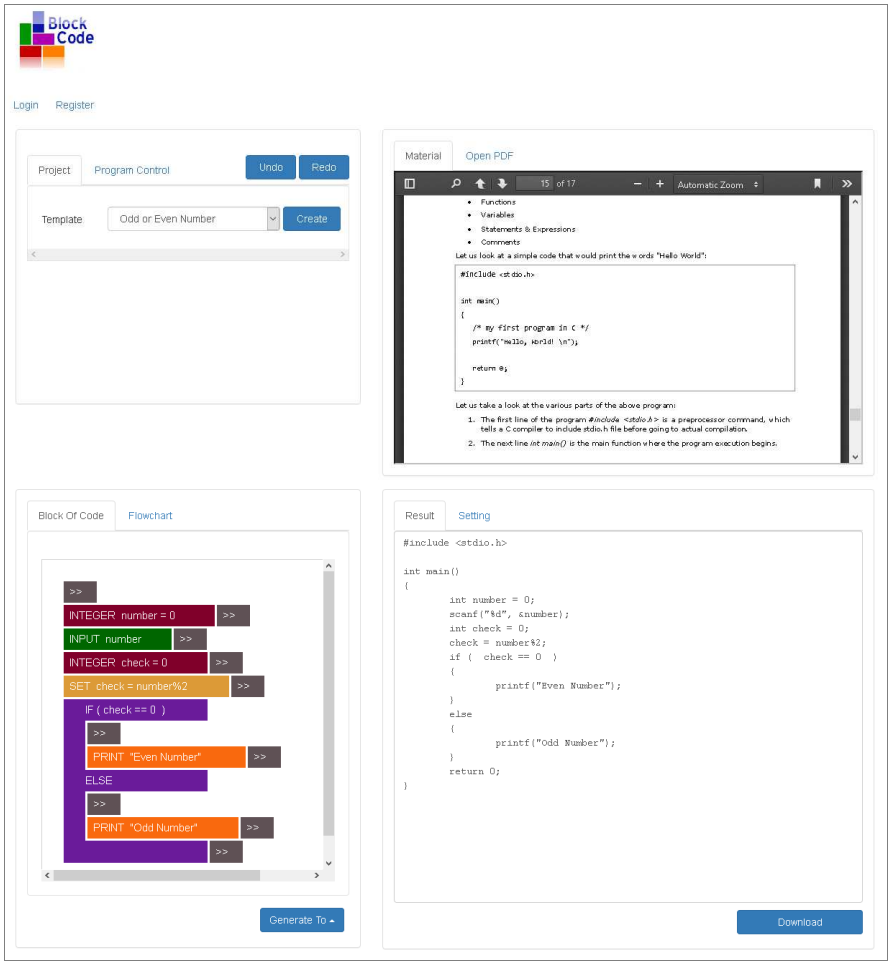
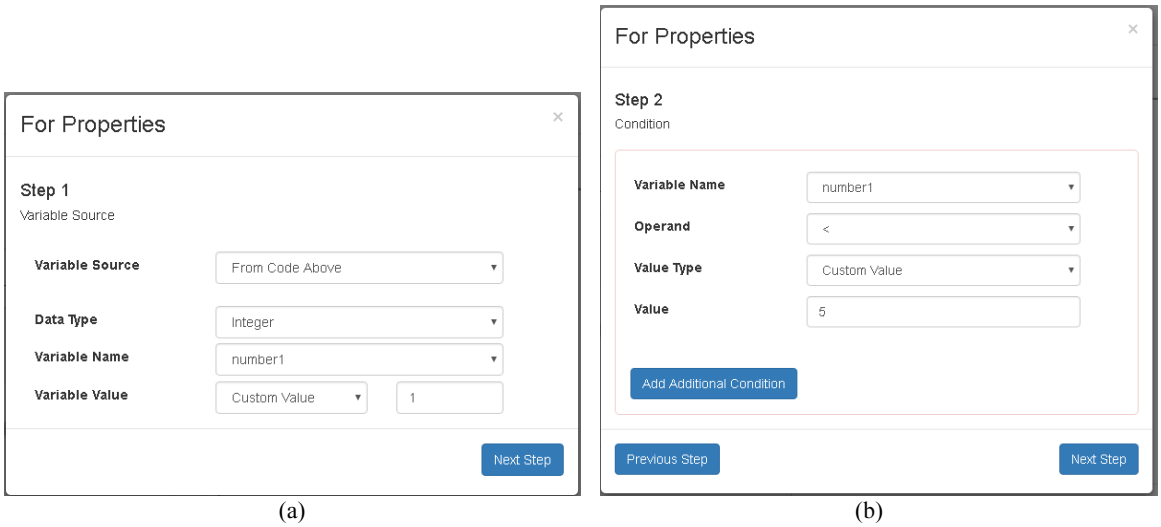
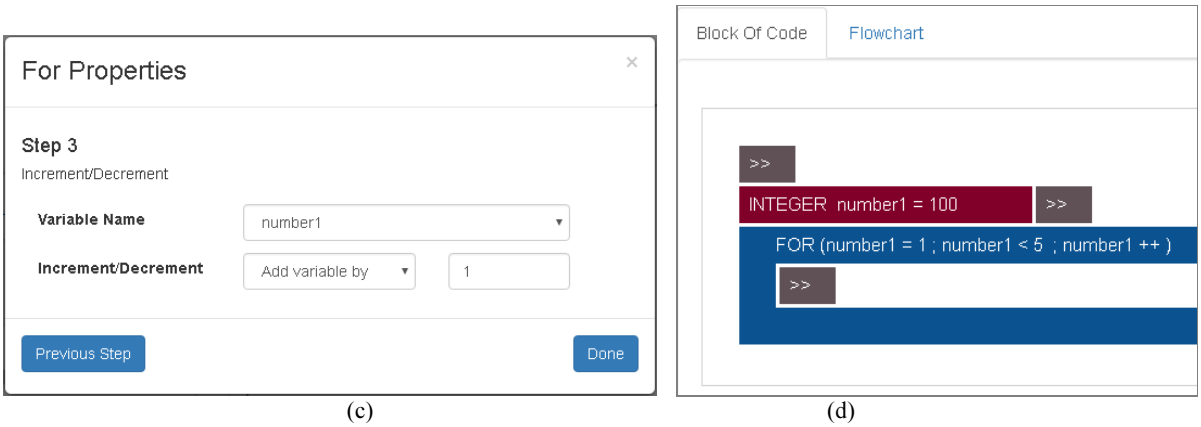


Fig. 2: Main window (4 sections)



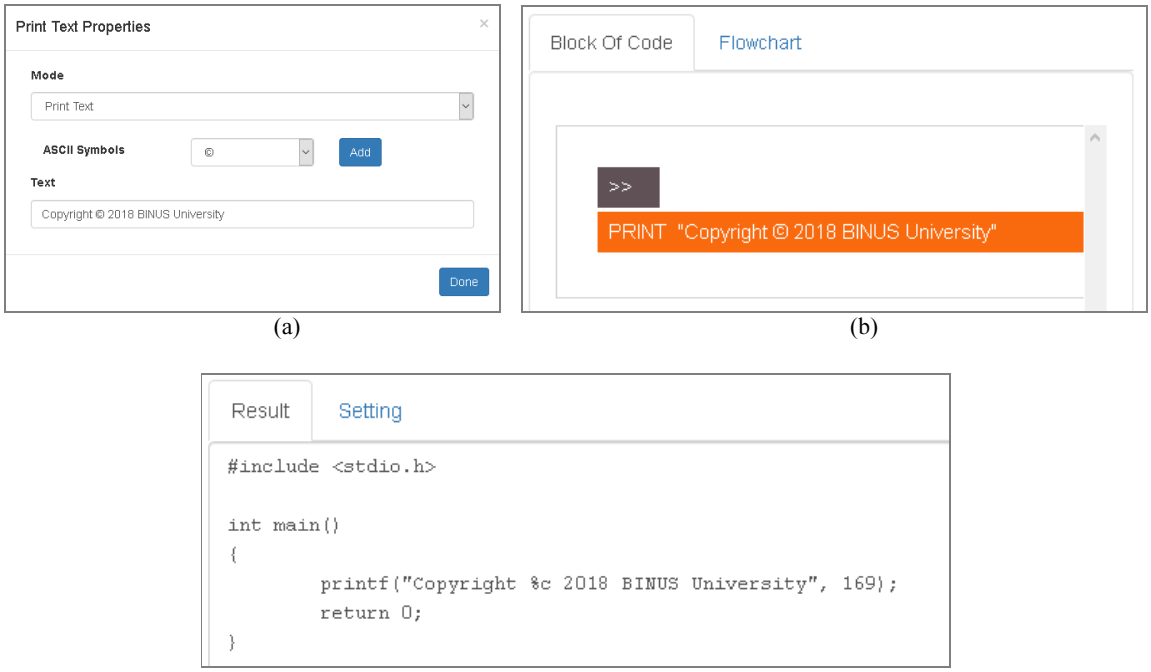
(a) (b)



(c) (d)

(e)

Fig. 3: Creating 'for' (repetition) statement: (a) parameter 1, (b) parameter 2, (c) parameter 3, (d) block of code, (e) sourcecode



(a) (b)

(c)

Fig. 4: Creating text (output) statement: (a) text to be printed (b) block of code (c) sourcecode

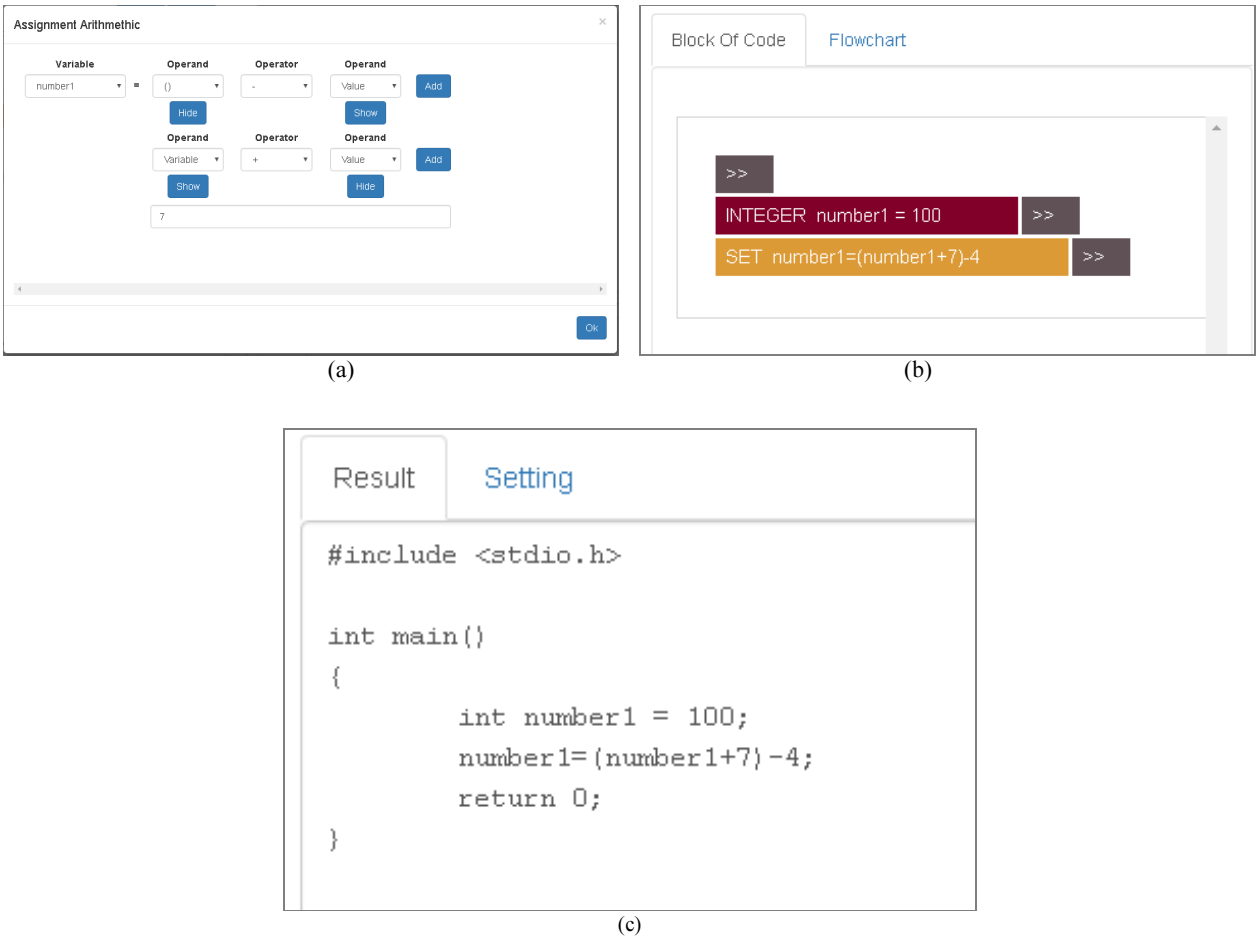
When a student creates an arithmetic statement, a pop-up window will be displayed (Fig. 5). Student selects an existing variable, operand 1, operator and operand 2. Each operand can be dropped down into a sub arithmetic expression (also consists of operand 1, operator and operand 2). After that, block of pseudocode is displayed and student can generate it into a programming language.

Evaluation

Evaluation is conducted by doing an experiment to 260 first semester students in computer laboratory (divided in 2 groups of 130 students). All are new students who have learned basic Java programming theory (I/O, variable, data type) and introductory of selection and repetition structure. Group 1 is for traditional learning without using this application and Group 2 is using. By giving same pretest (an arithmetic case), both groups are statistically indicated as normal and homogeneous population and have same average of score. Average pretest score for Group 1 is 53.53 and Group 2 is 54.24.

Due to limitation of computer laboratory capacity, each group is divided in 4 class sessions of 32-33 students. Instructor (same for all sessions) provides a demo of using the application for Group 2 and then the students try.

Next, instructor gives same posttest (arithmetic case, different from pretest). All students for both groups are given time for thinking the solution (algorithm) in 5 min and write on white paper. After that, students in Group 1 are writing the code directly by using TextPad. Students in Group 2 start to transform their algorithm into block of pseudocode by using the application and then generate it into Java programming languages (source code). The source code is downloaded and then compiled by using TextPad to check its accuracy for both error free and algorithm correctness. Both groups are given the same duration of 30 min to finish the posttest. After the session, questionnaire is distributed to students in Group 2 to evaluate the application subjectively. Results are shown on Table 5.



(a) (b)

(c)

Fig. 5: Creating aritmethic statement: (a) operand and operator (b) block of code (c) sourcecode

Table 5: Results of Evaluation

Subject	Yes (%)	No (%)
Application is easy to use	92.3	7.7
Application UI is attractive	69.2	30.8
Application can help in learning programming	97.7	2.3
Interest to use the application to start learning other programming language	94.6	5.4
Generating sourcecode is quite fast	73.8	26.2
Generated sourcecode is accurate (no error)	100.0	0.0

At the end, instructor collects all source code files from both groups and compile it one by one. All source code files from Group 2 are error free (in line with the questionnaire results). After that, scoring of all students from both groups is done by checking syntax, I/O, data type, conditional expression, selection, etc. Average score of Group 1 is 70.95 from 100 (with lowest score is 50 and highest is 88). Average score of Group 2 is 87.16 (with lowest score is 59 and highest is 100).

Conclusion

Based on the experiment conducted on this study, it can be concluded that this proposed application can help novice programmers in writing their logic (algorithm) in the form of block of pseudocode, then generate it into source code. Generated source code can be downloaded according to the provided programming language and then students can modify and learn from it (learning by doing). Additionally, this application is also accessible on various devices, both PC and mobile.

This study has limitation of comparing the duration of finishing the posttest (how many students finish the posttest earlier). Further research can include the time calculation, effectiveness in learning or improving programming or problem-solving skills and easiness of learning or migrating the new programming languages.

Acknowledgment

This research is funded by the Ministry of Research, Technology and Higher Education of the Republic of Indonesia through Doctoral Dissertation Grant (contract number of 039A/VR.RTT/VI/2017); and supported by Doctor of Computer Science, Bina Nusantara University. Thanks to Everald Kevin Setiadi, Henry Febryan and Calvyn Julian in developing the prototype.

Author's Contributions

Budi Yulianto: Lead research project, coordinate developer, doing experiment and write the manuscript.

Harjanto Prabowo: Advise research project, design the experiment, data analysis and write manuscript.

Raymond Kosala: Advise research project, design the application, data analysis, write manuscript and proof reading.

Manik Hapsara: Advise research project, design the research, methodology, data analysis, write manuscript and proof reading.

Ethics

Authors confirm that this manuscript has not been published elsewhere and that no ethical issues are involved.

References

Bau, D., D.A. Bau, M. Dawson and C. Pickens, 2015. Pencil code: Block code for a text world. Proceedings of the 14th International Conference on Interaction Design and Childre, Jun. 21-24, ACM New York, NY, USA, pp: 445-448. DOI: 10.1145/2771839.2771875

Brandão, L.D., R. da Silva Ribeiro and A.A. Brandão, 2012. A system to help teaching and learning algorithms. Proceedings of the Frontiers in Education Conference, Oct. 3-6, IEEE Xplore Press, Seattle, WA, USA, pp: 1-6. DOI: 10.1109/FIE.2012.6462374

Ebrahimi, A., S. Geranzeli, T. Shokouhi and E.R. Tee, 2013. Programming for children; "Alice and scratch analysis. Proceedings of the 3rd International Conference on Emerging Trends of Computer and Information Technology, Nov. 6-7, ResearchGate GmbH, Singapore, pp: 106-115.

Grantham, N., 2011. 9 sites that make programming for kids fun. Fractus Learning.

Hundhausen, C.D., S.F. Farley and J.L. Brown, 2009. Can direct manipulation lower the barriers to computer programming and promote transfer of training? An experimental study. ACM Trans. Computer-Human Interact., 16: 13-13. DOI: 10.1145/1592440.1592442

Jordine, T., Y. Liang and E. Ihler, 2014. A mobile-device based serious gaming approach for teaching and learning Java programming. Proceedings of the Frontiers in Education Conference, Oct. 22-25, IEEE Xplore Press, Madrid, Spain, pp: 1-5. DOI: 10.1109/FIE.2014.7044206

Layona, R., B. Yulianto and Y. Tunardi, 2017. Authoring tool for interactive video content for learning programming. Proc. Comput. Sci., 116: 37-44. DOI: 10.1016/j.procs.2017.10.006

Microsoft, 2015. Three out of four students in Asia Pacific want coding as a core subject in school, reveals Microsoft study.

Pears, A., S. Seidman, L. Malmi, L. Mannila and E. Adams *et al*., 2007. A survey of literature on the teaching of introductory programming. ACM SIGCSE Bull., 39: 204-223. DOI: 10.1145/1345443.1345441

Sentance, S. and S. Schwiderski-Grosche, 2012. Challenge and creativity: Using .NET gadgeteer in schools. Proceedings of the 7th Workshop in Primary and Secondary Computing Education, Nov. 08-09, ACM New York, USA, Hamburg, Germany, pp: 90-100. DOI: 10.1145/2481449.2481473

Shuhidan, S.M., M. Hamilton and D. D'Souza, 2011. Understanding novice programmer difficulties via guided learning. Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, Jun. 27-29, ACM New York, NY, USA, Darmstadt, Germany, pp: 213-217. DOI: 10.1145/1999747.1999808

Stack Overflow, 2015. Developer survey 2015. http://stackoverflow.com/research/developer-survey-2015

TechTerms, 2011. Programming Language. https://techterms.com/definition/programming_language

Vivian, R., K. Falkner and C. Szabo, 2014. Can everybody learn to code?: Computer science community perceptions about learning the fundamentals of programming. Proceedings of the 14th Koli Calling International Conference on Computing Education Research, Nov. 20-23, ACM New York, NY, USA, Koli, Finland, pp: 41-50. DOI: 10.1145/2674683.2674695

Weintrop, D., 2015. Minding the gap between blocks-based and text-based programming. Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Mar. 04-07, ACM New York, USA, Kansas City, Missouri, pp: 720-720. DOI: 10.1145/2676723.2693622

Yulianto, B. and H. Prabowo, 2017. Effective digital contents for computer programming learning: A systematic literature review. Adv. Sci. Lett., 23: 4733-4737. DOI: 10.1166/asl.2017.8877

Yulianto, B., E. Heriyanni, R.E. Sembiring, R. Amalia and R. Fridian, 2013 Aplikasi pembelajaran algoritma dasar interaktif berbasiskan computer assisted instruction. ComTech: Comput., Math. Eng. Applic., 4: 1255-1266. DOI: 10.21512/comtech.v4i2.2611

Yulianto, B., H. Prabowo and R. Kosala, 2016a. Comparing the effectiveness of digital contents for improving learning outcomes in computer programming for autodidact students. J. e-Learn. Knowl. Society.

Yulianto, B., H. Prabowo, R. Kosala, amd M. Hapsara, 2016b. MOOC architecture model for computer programming courses. Proceedings of the International Conference on Information Management and Technology, Nov. 16-18 IEEE Xplore Press, Bandung, Indonesia, pp: 35-40. DOI: 10.1109/ICIMTech.2016.7930298

Yulianto, B., H. Prabowo, R. Kosala and M. Hapsara, 2017. Harmonik = ++(Web IDE). Proc. Comput. Sci., 116: 222-231. DOI: 10.1016/j.procs.2017.10.044