BBP (block-based programming) toward TBP (text-based programming)

Leon Tynes and Kyungbin Kwon

Title: Guide students toward text-based programming through pseudocode activities

Summary: When students understand pseudocode in block-based programming exercises and projects, they will have an easier experience comprehending functions and syntax in text-based programming. This approach will make students' thinking visual for teachers' formative assessment.

Background:
"Pseudocode is a kind of structured english for describing algorithms. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax… It describe[s] the entire logic of the algorithm so that implementation becomes a rote mechanical task of translating line by line into source code" (California Polytechnic State University, n.d.). The process of decoding block-based programs to pseudocode will strengthen student understanding between specific block structure in Computational Thinking (CT) concepts, as well as emphasize how important pseudocode is to the effective development of program code in either format. However, the methodology of writing pseudocode and decoding block-based programs is scarce in coding lessons and activities.

There is a lack of explanation (learning experience) making connection between two different modes of programming. Students often feel huge gaps between block-based programs and text-based programs in K-12 education. The basis of student frustrations with text-based programs commonly stem from "pattern recognition and reproduction" directly related to syntax (Taggart, n.d.). Pseudocode instruction is not emphasized in K-12 classrooms and third party curriculum providers, and there are no decoding exercises for students to develop and understanding of the underlying text based code in block-based programming platforms. Therefore, students are not developing fluency in writing or understanding text based coding languages based on their block-based programming learning experience.

There are programs, such as Droplet, that reveal text-based code from block-based programming platforms. However, without specific lessons centered around pseudocode and the introduction of text-based programming syntax, students still have difficulty making the correlation between the function and programming concepts between the two platforms. We suggest that using pseudocode can be a stepping stone to understanding common principles (CT concepts) shared in them.

Research

Participants: This study (Kwon, 2017) focused on preservice teachers who have minimum learning experience in programming. Target students can navigate block-based program learning environments and have a basic conceptual understanding of CT concepts. They know the basic structure of text-based program syntax and can read simple codes, but they are not yet proficient in developing text-based programs.

Context of learning: The main goal of the course was to introduce computational thinking and to teach programming concepts. In order to emphasize the importance of conceptual understanding of programming, the instructor provided problems that students could solve without a computer, and asked them to develop pseudocode of the solutions. The researcher aimed to identify students' misconceptions of programming by analyzing their pseudocode.

Findings

Students' pseudocode that illustrated solutions of given problems revealed their mental models utilizing the concepts of programming and computational thinking skills. Findings suggested that students (1) liked to use a specific case to understand problem-solving process rather than consider a general solution at first; (2) omitted necessary specifications based on a false assumption that computers would be able to follow the instructions that were quite ambiguous; (3) did not express their intention that computer could execute.

So what?

Considering that pseudocode can reveal students' conceptual understanding of programming, teachers will be able to understand students' CT concepts through their pseudocodes and provide tailored guide based on that. By doing so, the study suggests pedagogical implementations that would make a smooth transition from block-based programming to text-based programming. Introducing more pseudocode writing opportunities and providing necessary scaffold to guide them in learning CT concepts are necessary (Odisho, Aziz, and Giacaman, 2016). In an effective CS curriculum or course, students need to have text-based programming experience based on core CT concepts that relate back to block-based programming. Coding platforms such as Microsoft MakeCode (with Wonder Workshop hardware devices) provide students opportunities to transition from BBP to Javascript within the same activity and learning objective. The Raspberry Pi hardware devices work with the Scratch BBP in its operating system, while providing opportunities to transition students to the Python TBP.

In both of these examples, the BBP becomes the pseudocode for students to effectively modify, then write accurate text-based code. Teachers must intentionally evaluate educational coding platforms with the development of pseudocode as an essential part of programming that will allow students to perform in either platform with an understanding of core concepts.

Reference:

California Polytechnic State University. (n.d.). Pseudocode standard. Retrieved from http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html

Kwon, K. (2017). Novice programmer's misconception of programming reflected on problem-solving plans. *International Journal of Computer Science Education in Schools, 1*(4), 14-24. doi:10.21585/ijcses.v1i4.19

Powers K, Ecott S, Hirshfield L M. 2007. Through the looking glass: teaching CS0 with Alice. *SIGCSE Bull. 39*, 1 (March 2007), 213-217.

Odisho, O. , Aziz, M. and Giacaman, N. (2016), Teaching and learning data structure concepts via Visual Kinesthetic Pseudocode with the aid of a constructively aligned app. *Comput Appl Eng Educ, 24*, 926-933. doi:10.1002/cae.21768

Taggart, M. (n.d.). *Bridging the Scratch Gap: From Blocks to Text Programming*. Retrieved from https://theforeverstudent.com/bridging-the-scratch-gap-from-blocks-to-text-programming-7b3db8356000

Zhen Xu, Albert D. Ritzhaupt, Fengchun Tian & Karthikeyan Umapathy (2019) Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study, *Computer Science Education*, DOI: 10.1080/08993408.2019.1565233