Multiple Programming Languages for Improving Computational Thinking in CS1

Dongeun Lee
Texas A&M University-Commerce
Commerce, Texas, USA
dongeun.lee@tamuc.edu

Kaoning Hu
Texas A&M University-Commerce
Commerce, Texas, USA
kaoning.hu@tamuc.edu

Omar El Ariss
Texas A&M University-Commerce
Commerce, Texas, USA
omar.el.ariss@tamuc.edu

Kibum Kwon
Texas A&M University-Commerce
Commerce, Texas, USA
kibum.kwon@tamuc.edu

ABSTRACT

Computational thinking can be deemed as thinking in algorithmic way, with which one can transpose given problems into computer algorithms. Since computational thinking requires abstract reasoning, it should not depend on particular programming languages. Unfortunately, introductory programming courses (CS1) often give students false impression that their goals are to teach a particular programming language. This study shares the design of new pedagogy for CS1 that removes dependency on a particular language and promotes computational thinking by teaching multiple programming languages simultaneously. Specifically, chosen programming languages range from low-level to high-level to expose students to different levels of abstraction from the details of computer architecture. Initial student survey responses from both trial and control groups show that there are significant improvements for the trial groups.

1 INTRODUCTION / PROBLEM

Traditionally, introductory programming courses (CS1) focused on the syntax and semantics of a particular programming language, where students have to learn syntactic rules first and then understand meaning of these rules. In this environment, the course objective naturally gravitates toward mastering a programming language itself and it is difficult to emphasize computational thinking that involves the process of abstract reasoning.

2 BACKGROUND / METHODS

Children are encouraged by parents to learn multiple languages. They excel in learning all of them at the same time. We believe that similar learning process can occur for novice learners in CS1. When properly encouraged to do so, students may be able to learn multiple languages at the same time, which in turn will provide the solid foundation for computational thinking.

We enforce computational thinking in CS1 by removing dependency on a particular programming language. We use multiple programming languages in CS1, ranging from high-level to low-level, simultaneously to broaden students' horizons beyond the syntax and semantics of programming languages. Although there have been few efforts where students are taught multiple programming languages in CS1 [1], those approaches were far from promoting abstract reasoning by multiple languages. Our focus is on teaching students computational thinking where high-level programs in any language are mapped to the same low-level representations. Students learn how different aspects of programming (e.g. data types and control structures) can be represented with different levels of abstraction.

In particular, students are first exposed to similar, but different syntax of Python (higher level) and Java (high level). They then understand these two codes are eventually processed the same way by our computer architecture with the help of Java bytecode (low level). Students build their confidence in programming by reading three different codes at the same time and realizing they are all semantically the same with different levels of abstraction. Here the focus is not on contrasting the syntax of one language to others, but on forming a general notion by abstracting from particulars.

3 RESULTS

We offered CS1 in both new pedagogy (trial group) and traditional pedagogy (control group) where a single language is taught, for Fall 2021 and Spring 2022 semesters. The proposed new pedagogy turns out to be more effective than traditional pedagogy, which is supported by student survey results regarding confidence, interest, and career in computer science. We also collected open-ended responses via the survey and identified students realize similarity in different languages despite the difference in syntax by constant practice of converting one language to others.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2021446.

REFERENCES

[1] Gongbing Hong, Jenq-Foung Yao, Chris Michael, and Lisa Phillips. 2018. A multilingual and comparative approach to teaching introductory computer programming. *Journal of Computing Sciences in Colleges* 33, 4 (2018), 4–12.

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9433-8/23/03.
https://doi.org/10.1145/3545947.3576322