How to Teach Recursion: A Formula to Transform an Iterative-Based to a Recursive-Based Method

Christian Servin
cservin1@epcc.edu
El Paso Community College
El Paso, Texas, USA

ABSTRACT

Recursion is one of the essential concepts in computer programming courses and one of the most challenging ideas to deliver in the fundamentals of computer programming courses effectively. Although several previous approaches aid students identify base and recursive cases when they write recursive-based methods, no algorithm has been proposed to help students understand the translation process between iterative to recursive. We proposed a teaching technique that allows students to translate components from loop-based programming to a recursive one. We demonstrate the method in several well-known problems.

CCS CONCEPTS

• **Applied computing**; • **Education**; • **Computer-managed instruction**;

KEYWORDS

CS 1; CS 2; Recursion; Fundamentals in CS; Community College

1 PROBLEM AND MOTIVATION

Recursion is a fundamental concept in computer science that is extremely useful for any implementation that requires divide-and-conquer thinking such as linked lists, binary search trees, or sorting algorithms. Recursion is also known for its elegance in programming style, despite its space complexity. It is known that the notion of recursion is not trivial to understand one introduced in the fundamentals of CS. However, there are several studies to improve recursion education in computer science education. There is work that helps to recognize base cases and recursive calls, as stated in [1, 4]. Some studies helped understand these concepts through visualization techniques [2], by identifying and handling base case attributes [4], or by providing explicit guidance to students on how to choose control-flow structures [3]. Although these techniques have helped understand the idea of recursion and to define base cases, they do not provide a systematic process to help students translating iterative-based concepts into recursion.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE '21, March 13–20, 2021, Virtual Event, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8062-1/21/03.
https://doi.org/10.1145/3408877.3439670

2 OVERVIEW

The dissemination of the recursion concept takes place in two courses. In CS 1, right after the topic of methods and more extensively in CS 2 before introducing data structures such as linked lists and trees. Both courses use Java as the programming language. The algorithm proposed consists of five steps: (1) while loop translation; (2) control variables identification; (3) return data type identification; (4) boolean condition exchange; (5) parameters exchange. The technique proposed has been implemented in a community college for the past four years in CS 2. Students registered in this course are Calculus 1 ready, where a few students take Discrete Math concurrently with CS 2. We compared two different sections of CS 2, where one section used the technique proposed, and the other uses regular lecture notes and book descriptions. An assignment and a quiz were used as instruments to measure the efficacy of the method. The results showed that students who used the translation technique wrote recursive-based methods more effectively by establishing the correct base and recursive cases (assuming the iterative-based method was correct). Students who did not use the translation technique provided solutions to a problem; however, some solutions lack correct base cases and had syntax errors.

3 CONTRIBUTION AND FUTURE WORK

This work's primary contribution is an algorithm's formalism to translate an iterative-based into a recursive-based method for three different types of recursion: linear, tail, and binary. The work shows that students find a comfortable way to handle recursion, despite the fact of memorizing the algorithm steps. It is intended to evaluate student's recursive-based skills in CS 3 using different instruments.

REFERENCES

[1] Kim B. Bruce, Andrea Danyluk, and Thomas Murtagh. 2005. Why Structural Recursion Should Be Taught before Arrays in CS 1. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA) *(SIGCSE '05)*. Association for Computing Machinery, New York, NY, USA, 246–250. https://doi.org/10.1145/1047344.1047430

[2] Wanda Dann, Stephen Cooper, and Randy Pausch. 2001. Using Visualization to Teach Novices Recursion. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education* (Canterbury, United Kingdom) *(ITiCSE '01)*. Association for Computing Machinery, New York, NY, USA, 109–112. https://doi.org/10.1145/377435.377507

[3] Ramy Esteero, Mohammed Khan, Mohamed Mohamed, Larry Yueli Zhang, and Daniel Zingaro. 2018. Recursion or Iteration: Does It Matter What Students Choose?. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1011–1016. https://doi.org/10.1145/3159450.3159455

[4] Michael Wirth. 2008. Introducing Recursion by Parking Cars. *SIGCSE Bull.* 40, 4 (Nov. 2008), 52–55. https://doi.org/10.1145/1473195.1473219