프로그래밍 경험에 따른 Computational Thinking의 차이에 관한 연구

성정숙*, 김민자*, 김현철*
*고려대학교 컴퓨터교육과
e-mail:hkim64@gmail.com

A Study on Difference of Computational Thinking by Programming Experience

Jung Sook Sung*, Minja Kim*, Hyeoncheol Kim*
*Dept of Computer Education, Korea University

요 약

최근 프로그래밍 교육이 새로운 방식의 사고력을 길러준다는 점에서 컴퓨터 공학 전공자뿐 아니라 일반인에게도 그 중요성이 강조되고 있다. 이러한 추세에 따라 프로그래밍 교육을 통해 길러지는 새로운 사고 과정인 computational thinking 관련 연구가 다양하게 시도되고 있다. 그러나 대부분의 연구가 특정 프로그래밍 언어에 의존적인 평가도구를 개발하여 측정하고 있으며 프로그래밍 경험 자체가 일상생활에서 일어나는 현상의 문제 해결을 위한 computational thinking에 영향을 주었는지를 연구한 결과는 아직 명확히 알려진 바가 없다. 따라서 본 연구는 프로그래밍 경험 유무에 따라 computational thinking에 차이가 있는가를 알아보았으며 그 결과 프로그래밍 경험이 있는 학생이 일상에서 일어나는 복잡한 문제를 더 논리적이고 상세하게 추상화하고, 구조화를 통해 더 명확한 모델링을 하였음을 알 수 있었다. 결론적으로 프로그래밍 경험 유무에 따라 computational thinking의 차이가 있음을 알 수 있었으며, computational thinking의 함양을 위해 프로그래밍 교육이 도움이 될 것으로 기대할 수 있다.

1. 서론

최근 프로그래밍 교육이 새로운 방식의 사고력을 길러준다는 점에서 컴퓨터 공학 전공자뿐 아니라 일반인에게도 프로그래밍 교육의 중요성이 강조되고 있다. 빌게이츠, 빌 클린턴, 팀 오라일리, 에릭 슈미트 등의 유명 인사들은 code.org에서 게시한 영상에 출연하여 프로그래밍 교육은 새로운 시대가 요구하는 새로운 사고 과정을 기르기 위해 모두에게 필수적인 교육이라고 주장하였다[1]. 이러한 세계적 추세에 따라 프로그래밍 교육을 통해 길러지는 새로운 사고 과정인 computational thinking과 관련한 연구가 최근에 시도되고 있다. 그러나 교육용 프로그램인 스크래치를 활용한 프로그래밍 과정 속에서 computational thinking 요소를 찾아 진단하는 방식이 대부분이며, 프로그래밍 경험 자체가 일상생활에서 일어나는 현상에 대해 문제 해결을 위한 computational thinking에 영향을 주었는지를 연구한 결과는 아직 명확히 알려진 바가 없다. 이 관계를 이해하는 첫 단계로 본 연구에서는 K대학의 핵심교양과목 수강생을 대상으로 학생들의 프로그래밍 경험에 따라 computational thinking에 차이가 있는지를 알아보았다.

2. Computational Thinking과 추상화

Computational thinking은 컴퓨터 과학의 기초적 개념을 기반으로 문제를 해결하고 시스템을 디자인하거나 인간 행동을 이해하는 접근 방법을 의미한다[2]. 이는 문제와 해결방법을 정보 처리 매개체가 효율적으로 수행할 수 있는 형태로 체계화하는 사고 과정을 말한다[3]. 몇 몇 연구에서는 computational thinking을 논리적 사고, 알고리즘적 사고, 과학적 사고 등 여러 사고 능력의 집합으로 정의하고 있다[4, 5]. 또한 computational thinking은 컴퓨터 과학에 기초하지만 컴퓨터 과학뿐 아니라 인문, 사회 분야 등 영역에 관계없이 영향을 미치고 있다[6]. 따라서, computational thinking은 여러 영역에서 새로운 발견과 혁신의 수단이 될 것으로 전망된다[7].

그러므로 computational thinking은 컴퓨터 과학을 공부한 사람만을 위한 것이 아닌 모든 사람들이 갖추어야 할 기본 기술이며[2] 전공에 상관없이 교육해야 한다.

이렇게 중요성이 부각되고 있는 computational thinking 함양 교육을 위해 측정 방법에 대한 연구가 시도되고 있다. 하지만 대부분의 연구에서 특정 프로그래밍 언어에 의존적인 평가도구를 개발하여 측정하고 있어 [8, 9] '모든 상황과 모든 사람을 위한 computational thinking

(computational thinking everywhere for everyone)'[7]을 측정하는 방법으로는 한계가 있는 것으로 보인다. 따라서 본 연구에서는 실생활에서 만날 수 있는 문제를 학생들에게 제시하고 그 결과물을 computational thinking의 핵심 요소인 '추상화(abstraction)'에 근거하여 분석하는 방식으로 computational thinking을 측정하였다. Computational thinking의 핵심 요소인 추상화는[7], 비단 computational thinking에서 뿐 아니라 컴퓨터 과학 교육에서도 학생들의 학업성취도를 높이는 핵심 역량으로 인식되고 있다[10].

Computational thinking을 처음 고안한 Wing(2008)은 computational thinking의 추상화를 1) 정의, 2) 계층 구조화, 3) 계층 간의 관계 이해의 3가지로 보았다. 먼저, 정의는 어떤 사항을 강조해야 하는지 혹은 무시할 수 있는지를 파악하는 것을 의미한다. 계층 구조화는 추상화 과정이 여러 계층으로 이루어지며 이를 명확히 하는 것을 말한다. 마지막으로 계층 간의 관계를 이해하는 것은 각 계층 간의 기능과 관계에 대한 이해를 의미한다.

본 연구에서는 Wing(2008)이 정의한 추상화에 근거하여 제시한 과제에 맞도록 고안된 기준을 사용하였다. 이 때, 세번째 요소인 계층간 관계의 이해 부분은 과제와 적절하지 않다고 판단되어 제외하고 첫 번째와 두 번째 요소를 중심으로 기준을 정하였다.

3. 연구방법

본 연구는 프로그래밍 경험 유무에 따라 복잡한 문제를 더 논리적이고 상세하게 추상화하고 구조화하여 해결해 나가는 computational thinking에 차이를 보이는가를 알아보았다. 이를 위해 K대학의 2013-2학기 핵심교양수업 수강생을 대상으로 약 5분 정도 시간을 주고 '자동으로 라면을 끓여주는 기계를 만들기 위해 기계가 이해할 수 있도록 각 단계를 작성해보라'는 퀴즈를 제시하였다. 이후 작성된 결과물을 즉시 수거하여 연구자들에 의해 고안된 세 가지 추상화 평가요소를 기준으로 채점하고 분석하였다. 또한, 설문을 통해 수합한 학생들의 프로그래밍 경험 유무와 분석된 추상화 요소들의 점수를 연구문제에 답하기 위해 두 독립표본 T검정으로 통계 분석하였다.

라면 끓이는 기계 설계 퀴즈는 라면을 끓이는 과정을 자동화하기 위해 그 과정을 분해하고 재구성하여 글로 작성해보는 것이다. 이를 통해 일상생활에 일어날 수 있는 문제를 학생들이 어떻게 추상화하고 구조화 하여 해결하는지를 알 수 있다.

추상화는 필수단계 포함 정도와 계층별 세분화 정도로, 구조화는 상태변화에 따른 조건문 사용 정도를 기준으로 세 가지 요소를 각각 평가하여 분석하였다.

첫 번째 요소는 라면을 끓이는 과정을 논리적으로 무결하게 완성하기 위해 필수적인 단계(5단계)가 얼마나 포함되었는지를 알아보기 위한 것으로 평가는 표현된 필수단계의 수로 계산하였다.

두 번째 요소는 '라면을 끓인다'라는 막연한 주제를 얼마나 많이 세분화했는지를 세분화 단계별로 계층을 만들어 평가하였다. 예를 들어 '물을 끓인다'라는 단계를 그 하부 수준인 '냄비에 물이 있는지 확인한다', '냄비를 불에 올린다', '가열한다' 등으로 표현한 경우 각각의 세분화 단계마다 점수를 주어 평가하였다. 본 연구주제의 경우, 최소 1단계에서 최대 15단계까지 세부적으로 표현될 수 있었으며 총 5개의 계층으로 분류될 수 있었다.

마지막으로 구조화 정도를 보기 위해 상태변화에 따라 조건문을 적절히 사용했는지 여부로 점수를 주었다. 예를 들어, '물을 끓이고 라면을 넣는다'라고 하면 조건문을 사용하지 않은 것으로 보이지만, '물이 끓으면 라면을 넣는다'라고 표현하거나 이를 순서도와 같은 도식으로 표현한 경우 조건문을 사용한 것으로 간주하였다.

4. 연구결과

전체 수강생 115명 중 63명이 퀴즈에 답안을 제시하였으며 설문에 응답하였다. 응답자 중 프로그래밍 경험이 있는 집단은 33명, 프로그래밍 경험이 없는 집단 30명으로 약 50%의 비율을 나타내었다.

프로그래밍 경험이 있는 학생들 중 약 85% 정도인 28명은 대학교 때 처음 프로그래밍을 배웠다고 응답했다. 이는 사실상 대부분의 학생들이 대학교에 와서야 처음으로 프로그래밍을 접하게 됨을 알 수 있다. 처음 배우게 된 프로그래밍 언어로는 C와 같은 구조/절차적 언어가 프로그래밍 유경험자의 약 58%(19명)로 가장 많았고, Java나 C++와 같은 객체지향언어가 약 36%(12명)로 그 뒤를 이었다.

다음의 <표 1>에서 프로그래밍 경험 여부에 따라 추상화의 요소인 필수단계 포함 정도와 계층별 세분화 정도, 그리고 구조화의 요소인 조건문 사용정도를 두 독립표본 T검정으로 분석한 결과, 모든 요소에서 유의미한 점수 차이를 보였다.

<표 1> 프로그래밍 경험에 따른 추상화 및 구조화 요소들의 차이검정을 위한 두 독립표본 T검정 결과

	경험 여부	평균	표준 편차	t	p
필수단계	Y(n=33)	4.31	.80	45.70	.047*
	N(n=30)	3.92	1.31		
계층별 세분화	Y(n=33)	5.13	1.45	2.63	.011*
	N(n=30)	4.56	1.28		
조건문 사용	Y(n=33)	1.41	.71	2.59	.012*
	N(n=30)	.94	.71		

먼저, 필수단계 포함 정도에서 프로그래밍 경험이 있는 학생 집단은 5점 만점에 평균 4.31로 프로그래밍 경험이 없는 학생 집단의 평균(m=3.92)보다 높게 나타났다 (t=45.70, p=0.047). 두 번째로, 계층별 세분화 정도에서도

프로그래밍 경험이 있는 학생 집단의 평균(m=5.13)이 경험이 없는 학생 집단의 평균(m=4.56)보다 높은 결과가 도출되었다(t=2.63, p=0.011). 마지막으로, 구조화 표현정도에서도 프로그래밍 경험이 있는 집단이(m=1.41) 경험이 없는 집단(m=0.94) 보다 유의미하게 조건문을 사용한 것으로 나타났다(t=2.59, p=0.012).

결과적으로 추상화와 구조화의 정도를 알아보기 위한 3가지 요소에서 모두 통계적으로 유의미하게 프로그래밍 경험이 있는 집단이 없는 집단보다 높게 나타났다. 즉, 프로그래밍 경험이 있는 집단이 없는 집단보다 추상화와 구조화를 하는 정도가 높다고 할 수 있다.

5. 결론

프로그래밍 경험이 있는 학생이 일상에서 일어나는 복잡한 문제를 더 논리적이고 상세하게 추상화하고 구조화하여 더 명확한 모델링을 하였음을 알 수 있었다. 결론적으로 프로그래밍 경험 유무에 따라 computational thinking의 차이가 있음을 알 수 있었으며, computational thinking의 함양을 위해 프로그래밍 교육이 도움이 될 것으로 기대할 수 있다.

ACKNOWLEDGMENT

본 연구는 문화체육관광부 및 한국 콘텐츠진흥원의 2013년도 문화콘텐츠산업기술지원사업의 연구결과로 수행되었음.

참고문헌

[1] http://www.code.org

[2] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

[3] Cuny, J., Snyder, L., & Wing, J. (2010). Demystifying Computational Thinking for Non-Computer Scientists, Work in progress, retrieved at http://www.cs.cmu.edu/~CompThink/.

[4] CS4FN: http://www.cs4fn.org/computationalthinking/

[5] 유중현, 김종혜 (2008). 문제 해결과정에서의 정보과학적 사고 능력에 대한 개념적 고찰. 정보창의교육논문지, 2(2), 15-24.

[6] Bundy, A. (2007). Computational thinking is pervasive. Journal of Scientific and Practical Computing. 1, 67-69.

[7]Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transacations of the Royal Society A, 366, 3717-3725.

[8] Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The Fairy Performance Assessment: Measuring computational thinking in middle school. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 215-220). ACM.

[9] Bort, H., & Brylow, D. (2013, March). CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 427-432). ACM.

[10] Bennedssen, J., & Caspersen, M. E. (2008, September). Abstraction ability as an indicator of success for learning computing science?. In Proceedings of the Fourth international Workshop on Computing Education Research (pp. 15-26). ACM.