Journal of Korea Multimedia Society Vol. 20, No. 2, February 2017(pp. 382-395)
https://doi.org/10.9717/kmms.2017.20.2.382

컴퓨팅 사고력이 중요한 프로그래밍 교육에서 'code.org'를 활용한 교수방안

임 화 경†

A Study on Teaching using Website 'Code.org' in Programming Education based on Computational Thinking

Hwakyung Rim†

ABSTRACT

Learning computational thinking is very important in programming education. Computational thinking refers to the problem solving ability based on the theories of computer science, indicating the importance of algorithm thinking. That is the reason for focusing on promoting creativity and improving the problem solving ability of the students in programming education. This paper commented the elements to consider for teachers when teaching computational thinking to elementary school students with online coding education website 'code.org' that helps beginners have easy programming experiences based on the characteristics of the website, and proposed the appropriate teaching methods.

Key words: Computational Thinking, Algorithm, Creativity, Problem-solving Ability, Programming Education, Software Education, Code.org Site

1. 서 론

정보통신 기술은 사회의 패러다임을 급격히 변화시키고 있으며, 급속한 기술 혁신으로 현재, 4차 산업혁명 시대를 앞두고 있다. 단순한 지식경제 사회를 벗어나 창의융합형 인재가 필요한 융복합 기술이 주축이 되는 시대로 변화하고 있다[1-3]. 이러한 변화는 사회의 다양한 분야의 패러다임에 영향을 주고 있으며 특히, 교육에도 급격한 변화를 요구하고 있다. 최근, 교육 분야에서도 이러한 변화에 따라 창의융합형 인재 육성이 시급함을 깨닫고 교육부에서 2015년도 개정 교육과정을 통해 다양한 교육정책을 제시하고 있다[4-6]. 특히 정보통신 기술의 핵심인 소프트웨어 교육의 필요성에 대한 내용을 살펴보면 사용자(consumer) 입장의 컴퓨터 과학을 생산자(programmer) 입장의 컴퓨터 과학으로 교육적 시각이 변하기 시작하였음을 볼 수 있다. 즉, 소프트웨어를 사용하는 소비자가 아닌 개발하기 위해 필요한 인재육성이 시급함을 알 수 있으며, 이는 컴퓨터 과학적 원리와 기술을 기반으로 창의력(creativity)과 사고력을 통해 문제해결 능력(problem-solving ability)을 갖춘 인재를 육성하는 것에 초점을 두고 있음을 알 수 있다[7-9].

소프트웨어 교육에서 컴퓨팅 사고력(computational thinking)은 학습자(또는 개발자)의 창의력과 사고력을 향상시키고 신장하는데 매우 중요한 영향을 미친다. 그렇기 때문에 개발자의 창의력과 문제해결 방법을 프로그램화하여 표현한 것인 소프트웨어의

※ Corresponding Author : Hwakyung Rim, Address: (47503) Gyodae-ro 24, Yeonje-gu, Busan, Korea, TEL : +82-51-500-7325, FAX : +82-51- 500-7321, E-mail : rim@bnue.ac.kr
Receipt date : Jan. 4, 2017, Revision date : Jan. 9, 2017
Approval date : Jan. 11, 2017
† Dept. of Computer Education, Busan National University of Education
※ This study was supported by the research funding of Busan National University of Education in Korea(2015)

생산물(product)이 개발자의 저작권으로 독창성을 인정하는 이유이기도 하다.

위와 같은 이유로 교육부에서는 초·중등학교를 위한 2015년도 개정 교육과정에 소프트웨어 교육을 실시하기 위한 운영지침을 발표하였다[5]. 특히, 초등학교에서는 최소 연간 17시간이상을 확보하여 소프트웨어 교육을 실시하도록 세부지침을 발표하였으며, 실과교과와 연계하고 창의적 체험활동을 통하여 운영하도록 하였다. 그 내용은 크게 두 영역으로 나누어 '소프트웨어로 인한 생활의 변화와 정보사회에 필요한 윤리 의식'에 대한 내용과 '알고리즘과 프로그래밍'에 대한 내용으로 구성하고 있다. 2015년도 개정 전 교육과정과 크게 달라진 부분은 후자인 알고리즘과 프로그래밍 영역으로 컴퓨팅 사고력에 대한 내용을 다루도록 되어있다.

가장 중요한 것은 '알고리즘과 프로그래밍을 어떻게 가르칠 것인가?' 에 있다. 소프트웨어 교육 운영지침에서 제시한 교수학습 계획 및 전략을 살펴보면 "컴퓨팅 사고력을 기반으로 문제해결과정을 설계하고 이를 프로그래밍을 통해 실현할 수 있도록 시간을 편성한다" 와 "응용 소프트웨어의 사용법이나 프로그래밍 언어의 문법학습을 최소화하고 문제해결에 필요한 프로그래밍을 통해 컴퓨팅 사고력을 신장하는데 초점을 둔다"라고 제시하고 있다[5]. 이는 프로그래밍 언어를 선택하여 프로그래밍하는 과정에 초점을 두면 명령어와 문법에 집중하게 되고 학습자의 직관적인 코딩으로 문제해결 방법보다는 실행유무에만 관심을 갖게 되어 컴퓨팅 사고를 간과할 수 있기 때문에, 반드시 프로그래밍하기 전에 문제에 대하여 학습자 스스로 문제해결 방법을 구상하도록 하는 컴퓨팅 사고력에 초점을 둔 교수방법을 강조하고 있다.

이러한 사회적 요구에 따라 정부산하 기관 및 각 시도 교육청에서는 소프트웨어 교육에 대한 내용체계, 교수방법 및 평가에 대한 교육 자료를 제작 배포하고, 워크샵 또는 교사연수 프로그램 및 다양한 현장적용 프로그램 등을 개발하여 현재 활발하게 확산하고 있다[4,6]. 특히, 초등학생을 대상으로 하는 프로그래밍 교육은 초보자들도 쉽게 프로그래밍을 할 수 있도록 블록(block)으로 명령어를 지원하는 교육용 프로그래밍 언어(educational programming language)를 사용하여 교육하도록 안내하고 있다. 또한 유아를 포함한 청소년들이 쉽게 컴퓨팅 사고를 스스로 학습할 수 있도록 온라인으로 지원하는 미국의 Infosys 재단에서 지원하는 코딩(또는 프로그래밍) 교육 사이트인 'code.org'[10] 의 'Hour of code' 도 활용하도록 안내하고 있다.

따라서, 본 논문에서는 프로그래밍 교육에서 가장 중요한 요소인 컴퓨팅 사고(또는 알고리즘)를 어떻게 가르칠 것인가에 초점을 두고, 교수자로서 프로그래밍에 대해 초보자인 초등학생에게 온라인 코딩교육 사이트인 'code.org'를 활용하여 지도할 때 고려해야 할 사항을 'Hour of code'의 특징 분석을 통하여 설명하고, 이에 적합한 교수방안에 대해 제언하였다. 이 논문의 제언은 교육용 프로그래밍 언어를 사용하여 프로그래밍 교육을 지도하는 경우와 달리 온라인 학습 사이트를 활용하여 초등학생에게 컴퓨팅 사고를 교육하고자 하는 교수자에게 적합한 교수활동을 제언함으로써 프로그래밍 교육의 효과를 높이는데 그 목적이 있다. 논문에서 사용하고 있는 '프로그래밍 교육'이라는 용어는 소프트웨어를 생성하기까지의 전 과정을 포함하는 의미로 사용하였으며, '코딩' 또는 '프로그래밍' 용어는 프로그래밍 언어를 사용하여 프로그램화하여 실행하는 과정을 뜻하는 것으로 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구배경이 되는 소프트웨어 교육과 컴퓨팅 사고력에 대해 살펴보고, 교육부에서 제시한 알고리즘과 프로그래밍 영역의 성취기준과 프로그램 구현 프로세스와의 관계에 대하여 설명하였다. 3장에서는 알고리즘의 다양성과 효율성을 강조한 교수지도 방안에 대해 제시하였으며, 4장에서는 3장의 내용을 토대로 'code.org' 사이트를 활용하여 컴퓨팅 사고를 교육할 때 고려해야 할 요소들을 분석하고 이에 적합한 교수방안에 대해 제언하였다. 그리고 5장에서는 결론을 내렸다.

2. 소프트웨어 교육과 컴퓨팅 사고력

이 장에서는 연구의 배경이 되는 컴퓨팅 사고력에 대한 의미를 소프트웨어 개발과정과 비교하여 설명하였고, 초등학교에서 교육해야 할 소프트웨어 교육 운영지침에서 제시한 알고리즘과 프로그래밍 영역의 성취기준과 비교하여 설명하였다. 초등학생을 대상으로 하는 프로그래밍 교육이 단순히 프로그래밍 언어를 사용하여 직관적으로 프로그램화하는 것(코

딩) 또는 직관적인 코딩을 통해 피지컬 컴퓨팅을 하는 것으로 인식될 수 있기 때문에, 학습자의 창의력과 문제해결력을 고려한 프로그래밍 교육을 위해 교수자가 수업을 설계할 때 반드시 고려해야하는 프로세스를 설명하였다.

2.1 소프트웨어 개발 과정과 컴퓨팅 사고력

"컴퓨팅 사고력(computational thinking)"이라는 용어는 미국의 Wing 교수에 의해 언급이 시작되었으며, 이후 많은 학자들이 이 용어에 대하여 다양한 각도에서 정의를 내리고 정리하였다[11,12]. 2015년도 교육부에서 발표한 개정교육과정의 소프트웨어 교육 운영지침에도 이를 인용하여 "컴퓨팅의 기본적 개념과 원리를 기반으로 문제를 효율적으로 해결할 수 있는 사고능력" 이라고 명명하고 있다[5]. 이 용어는 소프트웨어 교육 즉, 프로그래밍 교육에서 알고리즘(algorithm)적 사고가 중요함을 강조한 것으로 학습자의 창의력, 문제해결력, 절차적 사고(procedure thinking)가 직접적인 영향을 주는 요소이기 때문이다. 위의 내용들은 소프트웨어를 생성하는 과정의 전체 프로세스를 학습에 반영해야 함을 의미하고 있다. 본 논문에서는 이 의미를 설명하기 위해 소프트웨어의 전체 프로세스를 거시적 관점에서 설명한 후, 상대적으로 하나의 소프트웨어를 프로그래밍 하는 과정을 미시적 관점으로 간주하여 설명하였다.

먼저, 거시적 관점에서 소프트웨어의 생명주기를 관리하는 분야인 소프트웨어 공학(software engineering)에 대한 정의를 살펴본다. 소프트웨어 공학은 소프트웨어의 요구사항 및 분석, 설계, 개발, 실행, 운용 및 유지보수 하는 전체과정을 체계적으로 관리 운용하는 것을 의미한다[13,14]. 이것은 공장에서 하나의 상품을 개발하고 상품화하여 판매한 후 소비자의 의견이나 불만사항에 대해 지속적으로 관리하는 공정과정과 유사하다. 이 과정을 상용화하기 위한 소프트웨어를 개발할 때 수행되는 일련의 과정에 유사하게 적용하고, 구체화한 공정 및 관리과정이 소프트웨어 공학이라고 볼 수 있다. 즉, 개발하고자 하는 소프트웨어에 대한 요구사항을 토대로 개발 가능성과 타당성에 대해 분석을 먼저 수행한다. 다음으로 요구사항을 만족시키는 전체적인 레이아웃과 세부적인 요소에 대하여 구체적으로 설계를 진행한다. 이 부분이 설계자의 독창성에 따라 여러 가지의 다양한 설계가 나올 수 있는 단계이다. 설계가 완성되면 프로그래밍 언어로 프로그래밍 작업을 수행하고 컴퓨터 시스템에서 실행될 수 있도록 컴파일 또는 인터프리터 작업을 거쳐 개발하고자하는 소프트웨어의 생산물(product)을 출력하게 된다. 이 후 오류수정 과정을 거쳐 상품화한 후에 주기적으로 발생하는 요구사항들을 고려하여 해당 과정을 유지보수하면서 업그레이드시키는 과정을 관리하게 된다.

하나의 소프트웨어는 생산될 때 까지 장기간 여러 과정을 거치면서 숙고하여 생산되기 때문에 오류가 발생할 때 폐기하고 새로운 것을 개발하는 것이 아니라 문제점을 주기적으로 수정하면서 업그레이드하여 성능을 향상시킨다. 따라서 이러한 생명주기 과정을 유지보수 관리하는 것이 매우 중요하다. 또한, 이 전체 과정은 하나의 소프트웨어 생산물을 생성하기 위해 반드시 필요한 단계들로 하나의 단계도 생략되면 고품질의 상품적 가치를 갖는 결과물을 생성하기가 어렵다.

그리고 하나의 소프트웨어가 생성되려면 많은 프로그램 객체들 간의 상호작용을 통해 하나의 생산물로 구현되는 과정을 거쳐야 한다. 여기서 하나의 객체인 프로그램 유닛이 생성되는 과정을 컴퓨팅 사고와 연계하여 미시적 관점으로 설명한다. 이 과정을 Fig. 1로 나타냈으며 각각의 단계에서 수행하는 작업과 컴퓨팅 사고력과의 관계를 중심으로 설명하였다. 또한 이 내용이 교수자가 프로그래밍 교육을 위해 교수설계를 할 때 직접적으로 고려해야 하는 부분이기도 하다.

Fig. 1의 프로그램 구현 프로세스(program implementation process)의 첫 번째 단계는 문제 정의(problem definition) 단계로 구현하고자 하는 프로그램의 주제 또는 문제를 설정하는 단계이다. 문제가

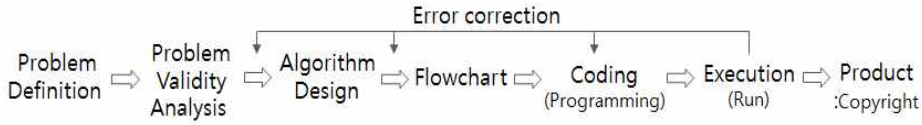


Fig. 1. Program implementation process.

설정되면 두 번째 단계로 프로그램화 할 가능성에 대한 타당성을 분석(problem validity analysis)한다. 문제를 해결하기 위해 필요한 요구사항이나 환경 또는 조건들을 고려하여 구현가능한지 그 여부에 대해 분석을 세밀하게 수행한다. 분석한 결과가 타당성이 있다면 세 번째 단계인 알고리즘 설계(algorithm design)를 수행한다. 이 단계는 개발자의 창의력과 문제를 해결하는 사고에 따라 알고리즘이 다양하게 설계될 수 있기 때문에 개발자의 문제해결력에 따라 프로그램의 효율성에도 영향을 주게 된다. 따라서 동일한 문제에 대하여 개발자의 창의력과 문제 해결력에 따라 다양한 알고리즘으로 설계될 수 있으므로 설계된 알고리즘의 효율성에 대해 분석하는 작업도 매우 중요한 과정이다. 이 과정에서 프로그램의 효율성(time complexity, space complexity)이 결정되기 때문이다.

이 세 단계가 프로그램 개발과정에서 가장 중요한 단계이다. 개발자의 창의성과 문제해결력에 의해 프로그램의 독창성과 효율성이 결정되기 때문이며, 이 부분이 컴퓨팅 사고와 직접 연관된 것으로 교수설계를 할 때에도 가장 핵심이 된다.

다음 단계로, 설계한 알고리즘을 공통 기호를 사용하여 순서도(flowchart)로 표현한다. 여기서, 알고리즘의 설계와 순서도의 표현은 분리된 작업이 아니라 대부분 알고리즘을 설계하면서 병행하여 수행한다. 이것을 토대로 프로그래밍 언어를 선택하여 코딩(coding)을 수행한 후 실행과정(run)을 거쳐 결과를 출력한다. 이 과정에서 오류가 발생하면 Fig. 1에서 보는 바와 같이 해당 단계로 회귀하여 오류가 발생하지 않을 때까지 수정을 거치는 작업을 반복한다. 더 이상의 오류가 발생하지 않으면 최종적으로 결과물(product)을 출력한다.

이러한 프로세스를 통해서 나온 결과물은 개발자의 아이디어로 설계하고 구현된 것이기 때문에 다양한 형태의 결과물로 표현되며 저작권 또한 중요성을 갖게 된다.

지금까지 설명한 바와 같이 컴퓨팅 사고력은 Fig. 1에서 표현하는 프로그램 구현 프로세스를 의미하는 것으로 프로그래밍 교육을 할 경우에 이 모든 프로세스를 통해 교수지도가 이루어져야 한다.

2.2 알고리즘과 프로그래밍 영역의 성취기준과 프로그램 구현 프로세스

2.1절에서 프로그램 개발의 전 과정이 컴퓨팅 사고력에 포함됨을 설명하였다. 즉, 어느 한 단계라도 생략하여 진행될 수 없음을 강조하였다. 이 장에서는 2015년도 개정 교육과정의 소프트웨어 교육 운영지침에서 제시한 초등학교에서 다루어야 할 알고리즘과 프로그래밍 영역의 성취기준과 2.1절에서 설명한 내용이 동일 맥락임을 설명하였다. Table 1은 소프트웨어 교육 운영지침에서 제시한 초등학교에서 다루어야 할 알고리즘과 프로그래밍 영역의 성취기준을 나타내고 있다.

중 영역은 크게 3가지로 나누고 있다. 첫째 영역인 '문제해결 과정의 체험(experience of the problem solving process)' 영역은 내용요소(component)로 '문제의 이해와 구조화(understanding and structuralization of the problem)' 그리고 '문제해결 방법 탐색(search for the problem solving methods)'으로 구성하고 있다. 이 부분은 프로그램 구현 프로세스에서 문제정의 단계부터 알고리즘 설계 이전단계까지에 해당된다. 학습자가 문제를 올바로 이해하고 있는가에 대한 사고력과 이를 기반으로 스스로 찾아낸 문제를 해결하는 방법에 대해 설명하고 문제점이 있다면 문제점을 해결하기 위해 개선점을 스스로 찾아낼 수 있는 사고 능력에 성취기준을 두고 있다. 이는 학습자의 창의력과 문제해결력을 신장시키는 것에 초점을 두고 있다는 것을 알 수 있다.

둘째 영역인 '알고리즘 체험(experience of understanding of algorithm)' 영역의 내용요소는 '알고리즘의 개념(the concept of algorithm)'과 '알고리즘의 체험(experience of algorithm)'으로 구성하고 있으며 알고리즘의 개념을 이해하고 간단한 알고리즘을 체험활동을 통한 이해력에 초점을 두고 있다. 즉, 첫째 영역의 학습을 통해 학습자가 고안한 문제해결 방법을 기호를 사용하여 순차적으로 순서도로 표현할 수 있는 능력을 신장시키는 것에 성취기준을 두고 있다. 이는 프로그램 구현 프로세스의 세 번째와 네 번째 단계에 해당된다.

셋째 영역인 '프로그래밍의 체험(experience of programming)' 영역의 내용요소는 '프로그래밍의 이해(understanding of programming)'와 '프로그래밍의 체험(experience of programming)'으로 구성하

Table 1. Achievement standards for 'algorithm and programming' in elementary school[5]

Mid Category	Component	Achievement standards
Experience of the problem solving process	Understanding and structuralization of the problem	Students can understand the proposed problems.
		Students can simplify the proposed problems.
	Search for the problem solving methods	Students can explain the problem-solving method according to the sequence.
		Students can explain issues and improvement methods of the proposed problem-solving method.
Experience of understanding of algorithm	The concept of algorithm	Students can understand the concept of algorithm.
	Experience of algorithm	Students can express the problem-solving procedure in symbols using sequence, selection and repetition.
		The students can understand simple algorithms (sort and search) through the experience activities.
Experience of programming	Understanding of programming	Students can understand the basic instructions of the programming language.
	Experience of programming	Students can make the same program with the given program.
		Students can make an original program by modifying the given program.
		Students can make their own simple program.

고 있다. 이는 순서도로 표현한 알고리즘을 시스템에서 실행하여 결과를 출력하는 과정을 프로그래밍 체험을 통하여 성취하도록 제시하고 있다. 프로그램 구현 프로세스의 다섯 번째 단계 이후에 해당되며, 둘째 영역까지의 학습을 통해 학습자가 설계한 기호화된 알고리즘을 바탕으로 적합한 프로그래밍 언어를 선택하여 코딩하고 실행하는 체험을 통해 프로그램을 생성하는 과정을 학습하는 것을 의미한다.

따라서, 해결해야 할 문제에 대해 학습자의 분석을 통하여 자신만의 문제해결 방법을 고안한 것이 알고리즘으로, 이것을 프로그램으로 표현하는 과정이 코딩이다. 만일 학습자 스스로 찾아낸 알고리즘을 설계하는 단계를 생략하고 코딩만 집중하여 교육한다면 직관적인 프로그래밍밖에 될 수 없기 때문에 컴퓨팅 사고를 기초한 교육이라고 보기 어렵다. 그리고 출력형태를 소프트웨어가 아닌 장치를 통해 출력하는 피지컬 컴퓨팅의 경우에도 출력실행 유무에 초점을 둔 학습이 아닌 알고리즘의 설계에 따라 출력이 어떻게 다르게 표현되는지에 초점을 두어야 컴퓨팅 사고에 의한 교육으로 볼 수 있다. 학습자의 창의력과 문제해결력은 문제분석과 알고리즘 설계에 의해 판단할 수 있기 때문이다. 단순히 직관적인 코딩을 위한 교육은 지양하고 프로그램을 구현하는 전 과정을 포함하여 학습하는 것으로 컴퓨팅 사고력을 해석해야 한다. 즉, Fig. 1에서 표현하는 전체 프로세스를 통해 교수지도가 이루어져야만 한다.

3. 알고리즘의 다양성과 효율성에 중점을 둔 교수지도

컴퓨팅 사고력은 학습자의 창의력과 문제해결력을 촉진하고 신장시키는 것이 매우 중요한 목적으로 이를 성취시키기 위해서 알고리즘 교육을 위한 교수지도 방법 또한 매우 중요하다. 이 장에서는 Fig. 1의 프로그램을 생성하는 전체 프로세스 중에서 코딩단계 이전까지 단계인 알고리즘 설계에 대해 지도할 때 교수자로서 고려해야 할 요소에 대해 설명하였다.

알고리즘을 교육할 때 교수지도 방법이 중요한 이유는 프로그래밍 교육은 생성된 결과물이 문제에서 요구하는 것을 정확하게 맞추고 있는가를 평가하는 것이 아니기 때문이다. 즉, 학습자가 어떻게 문제를 해결했는가를 관찰하여 평가하는 것이 중요하기 때문이다. 따라서 코딩은 학습자가 생각해낸 문제해결 방법을 절차적으로 표현해 놓은 알고리즘을 시스템

에서 실행시키기 위해 프로그래밍 작업을 하는 것이기 때문에 알고리즘 설계 없이 수행하는 코딩작업은 효율적인 알고리즘을 구현하기 어렵다고 볼 수 있다.

프로그래밍 교육을 처음 접하는 초등학생들에게 알고리즘에 대한 교수단계는 크게 세 단계로 나누어 실행되어야 한다. 이 때 교수자는 학습자가 문제를 해결하기 위해 노력하도록 창의력과 문제해결력을 자극시키거나 향상시키는 코칭역할을 하는 것이 매우 중요하다. 교수단계는 Table 2로 정리하였으며 다음과 같다. 이 교수단계는 알고리즘의 특징인 다양성과 효율성에 두고 있음을 볼 수 있다.

- 1단계 : 알고리즘의 개념과 알고리즘에 대한 변별력(the concept and discrimination capacity of algorithm)
- 2단계 : 알고리즘 설계와 다양성에 대한 학습(learning on design and the diversity of algorithm)
- 3단계 : 알고리즘의 효율성에 대한 이해능력과 효율적 알고리즘 설계(The capacity of understanding of the efficiency of algorithm and design of the efficient algorithms)

초등학생들에게 컴퓨팅 개념에 대한 학습은 난이도가 높을 수 있어서 실생활에서 컴퓨터와 관련한 사례를 통하여 쉽게 접근하는 것이 효과적이다. 먼저 알고리즘인 것과 아닌 것에 대한 변별력(discrimination:yes/no case) 학습을 통해 알고리즘의 개념을 자연스럽게 익히도록 한다. 이 때 교수자는 알고리즘을 제시하는 것이 아니라 학습자가 절차적 사고를 표현 하는 것에 익숙해지도록 스스로 생각해 낸 해결방법을 순차적으로 표현하도록 지도하고, 학습자들이 표현한 것들을 예시로 하여 예(yes case)와 비예(no case)를 변별할 수 있도록 코칭역할을 해야 한다. 그리고 교수자는 예와 비예에 대해 학습자 스스로 이유를 설명하도록 코칭하여야 한다. Fig. 2는 순차(sequence)만 고려한 알고리즘의 예와 비예를 구분하는 예제로 인터넷 사이트에 가입하는 절차의 일부예를 나타내고 있다. (b)가 비예이지만 서로 다른 세 가지의 알고리즘에 대해 학습자 스스로 판단하여 설명하도록 지도하여야 한다. 이러한 지도를 통하여 스스로 원인을 설명하도록 함으로써 복합적으로 알고리즘의 개념과 알고리즘을 구분할 수 있는 능력을 촉진시키도록 한다.

두 번째 단계에서는 알고리즘에 대한 변별력이 생긴 학습자이므로 실제 컴퓨터에서 해결해야 할 문제를 제시하고 학습자 스스로 문제해결 과정을 알고리즘으로 나타내도록 코칭한다. 이때 동일 문제에 대해 다양(diversity)한 알고리즘이 있을 수 있다는 것을

Table 2. Phases of teaching and roles of teachers in algorithm design

Phase	Component	Roles of teachers
Phase1	The concept and discrimination capacity of algorithm	• Teachers help students to develop procedural thinking through practice with the problems of the practical cases in the computational environment. • Teachers provide the concept learning and learning with discrimination using the yes/no cases. • Teachers provide repetitive learning through various cases so that students are able to have discrimination(yes case/no case) capacity.
Phase2	Learning on design and the diversity of algorithm	• Teachers help students to propose their problem-solving method in a way of procedural thinking about the given problem. • Teachers make students understand that various algorithms can be created on the same problem. • Teachers help students to explain the validity of each designed problem-solving method and mutually use their problem-solving method as a reference.
Phase3	The capacity of understanding of the efficiency of algorithm and design of the efficient algorithms	• Teachers help students to recognize the diversity of algorithm on the same problem and distinguish the efficient problem- solving method among the solutions. • Teachers teach students to determine the efficiency of algorithm depending on the conditions of the problems through students' algorithm. • Teachers help students to modify their algorithm to efficient ones.

(a) (b) (c)

Fig. 2. An example for phase 1, (a) and (c) yes case, (b) no case.

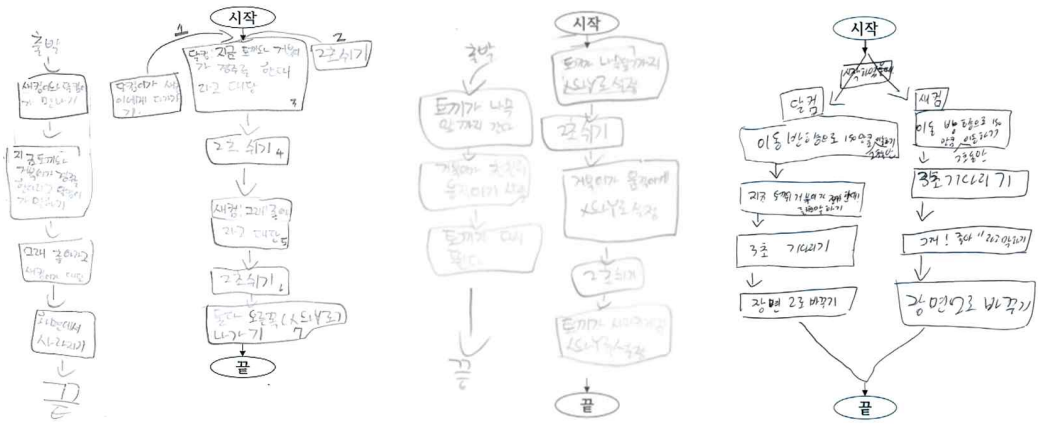
지도하여 학습자가 정답을 도출하려는 생각을 지양하도록 해야 한다. 즉, 학습자 각각 생각해 낸 문제해결 방법이 중요하다는 것과 다양한 알고리즘이 생성될 수 있다는 것에 불안감을 가질 수 있기 때문에 자신감을 넣어주어야 한다. 따라서 학습자가 생각해 낸 문제해결 과정을 본인만의 것으로 표현하는 것이 중요함을 강조해야한다. Fig. 3은 동일한 문제에 대해 다양한 방법으로 문제를 해결할 수 있음을 나타내는 난이도가 낮은 예이며, Fig. 4는 프로그래밍에 대해 초보자인 5학년 학생들에게 알고리즘 설계에 대해 교육한 사례의 일부를 나타낸 것이다. 문제의 주제는 두 객체가 주인공이 되어 문제를 해결하는 과정을 스토리텔링으로 구현한 프로그램으로 첫 장면의 동일한 스토리에 대해 다양한 알고리즘을 작성한 예이다.

이 때, 다양한 알고리즘에 대해 학습자들 간에 문제해결 방법을 경청하도록 지도해야한다. 이유는 본인과 다른 문제해결 방법에 대하여 이해하는 과정이 본인의 문제해결 방법을 수정하는데 영향을 미칠 수 있기 때문이다. 이는 보다 더 나은 알고리즘으로 업그레이드할 수 있는 능력을 기를 수 있기 때문에 다른 학습자의 생각을 듣는 토의 또는 토론과정이 매우 중요하며 이때 교수자는 코칭과 비계(scaffolding)역할을 해야 한다.

세 번째 단계에서는 이전단계까지의 사례와 연계하여 효율적인 알고리즘의 개념에 대해 지도한다. 이 학습은 학습자가 설계한 알고리즘을 효율적으로 수정할 수 있도록 지도하기 위한 알고리즘 교육의 마지막 단계로 볼 수 있다. 알고리즘의 효율성을 교육하기 위해서는 알고리즘을 표현하는 명령어인 순차(sequence), 조건(condition), 반복(repeat)에 대한 원리 학습이 반드시 이루어져야 한다. 이 부분은 기본

(a) (b) (c)

Fig. 3. An example of the diversity of algorithm on the same problem.



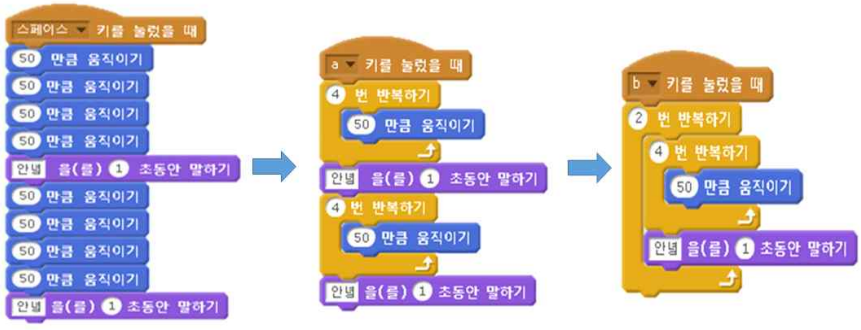
(a) (b) (c)

Fig. 4. An example of algorithm written by an elementary school students.

적으로 전체 단계에서 학습이 이루어져야 하지만 세 번째 단계에서 알고리즘의 효율성을 언급하기 위해서는 반드시 각 명령어에 대한 원리를 숙지시켜야 한다. 초등학생들에게 이 부분의 교육이 가장 어려운 부분이다. 따라서 각 명령어를 교육할 때는 사례를 들어 명령어를 익히도록 하는 것 보다는 학습자 스스로 설계한 알고리즘의 문제점 또는 불필요한 설계에 대해 설명하도록 하고 이에 따라 해당 명령어의 필요성을 설명할 수 있도록 코칭 해야 한다. 이 단계에서는 모든 학습자의 토의 또는 토론이 매우 중요하다. 직관적으로 명령어를 사용하여 알고리즘을 수정하는 것이 아니라 문제점과 해결점을 동료들의 의견을 상호교환하면서 원인을 찾아 스스로 문제를 해결하도록 유도해야 한다.

Fig. 5는 효율적인 알고리즘으로 변형하기 위해 명령어 선택이 중요함을 알려주는 학습사례의 예이다. 제시한 세 가지의 예가 모두 동일한 결과를 수행하지만 (a)는 순차적으로 처리한 명령어로 설계했기 때문에 동일한 명령어가 중복되는 문제점을 갖고 있다. 이를 해결하기 위해 반복문으로 해결한 것이 (b)이다. (a) 보다는 효율적이지만 여전히 반복문 2개가 중복되는 문제점을 갖고 있다. 이를 해결하기 위해 반복문을 중첩하여 사용함으로써 해결한 것이 (c) 이다. 이와 같이 학습자가 설계한 알고리즘을 분석하면서 한 단계씩 문제점을 해결하기 위해 학습자 스스로 판단하고 명령어를 선택하고 수정할 수 있도록 교수자는 코칭역할만 해야 한다. 이러한 코칭은 더욱 복잡한 문제를 해결해야 할 때 효율적인 알고리즘을 설계할 수 있도록 사고력을 신장시키는데 매우 중요한 역할을 하게된다.

또한, 효율성은 이 이외에도 Fig. 2와 같은 예처럼 조건에 따라 알고리즘의 효율성을 결정할 수 있는



(a) (b) (c)

Fig. 5. An Example of the efficiency of Algorithm (a) sequenced instruction (b) repeat loop (c) nested repeat loop.

것도 존재하기 때문에 다양한 조건에 따른 효율성 분석도 함께 병행하여야 한다. 결론적으로 학습자의 컴퓨팅 사고력을 향상시키려면 교수자의 코칭역할이 매우 중요하다는 것이다.

4. code.org 활용한 컴퓨팅 사고력 교육에 대한 지도방안

지금까지 알고리즘 교육의 중요성과 교수지도에 대해 제언하였다. 이장에서는 이 점에 초점을 두고 온라인 코딩교육 사이트인 'code.org'사이트의 'Hour of code'를 사용하여 초등학생들에게 프로그래밍 교육을 지도하고자 하는 교수자들에게 적합한 교수지도 방안에 대해 제언하였다.

본 논문에서 이 사이트를 선택한 이유는 다음과 같다. 대부분 초등학생을 대상으로 하는 프로그래밍 교육에서 코딩을 위해 초보자들도 쉽게 프로그래밍을 할 수 있도록 블록으로 명령어를 지원하는 교육용 프로그래밍 언어를 사용하고 있다. 이와 병행하여 프로그래밍에 대하여 스스로 학습할 수 있도록 온라인 코딩교육 사이트를 보조적 역할로 활용하거나 초보자에게 쉽게 프로그래밍에 익숙해지도록 이 사이트만 활용하여 지도하는 경우도 있다. 이 사이트를 활용할 경우에는 교육용 프로그래밍 언어를 사용하여 프로그래밍을 하는 경우와 교육환경이 다른 점을 파악하고 이에 적합한 교수지도를 해야 한다. 즉, 컴퓨팅 사고력을 교육하기 위해서는 적합한 언어 선택뿐만 아니라 온라인 사이트선택이 매우 중요하며 선택한 것에 따라 교수지도 방법이 다르게 적용되어야 하기 때문이다.

4.1 교육용 프로그래밍 언어와 code.org 사이트의 특징

교육용 프로그래밍 언어는 전문적인 프로그래밍 언어를 초보자들도 쉽게 프로그래밍 할 수 있도록 명령어를 블록으로 시각화하여 드래그 앤 드롭(drag & drop)기능을 통하여 코딩하고 실행하여 즉시 출력물을 확인할 수 있도록 시각화한 플랫폼을 지원하는 언어들을 의미한다[15]. 단순히 난이도가 높은 일반 프로그래밍 언어를 초중등학생의 수준에 적합하게 개발한 언어이기 때문에 프로그램 구현 프로세스(Fig. 1) 중에서 코딩단계 이후에 직접적으로 사용된다. 프로그래밍 언어는 다양하기 때문에 언어의 선택은 해결하고자 하는 문제에 적합한 것을 선택하는 것이 매우 중요하다. 초등학교에서는 특정분야를 구현하는데 한정된 언어가 아닌 범용 프로그래밍언어(general programming languages)인 스크래치(scratch)[16]또는 앤트리(entry)[17]언어를 사용하도록 권장하고 있다.

프로그래밍은 학습자가 설계한 알고리즘을 바탕으로 시스템에서 실행될 수 있도록 수행하는 작업이기 때문에 프로그램 구현 프로세스를 모두 수행하는 가운데 실행이 이루어져야 한다. 알고리즘 설계부분을 간과하고 코딩에만 집중한다면 직관적인 코딩으로 학습자의 창의력과 문제해결력은 의미가 없어지기 때문에 반드시 전체 프로세스를 통해 결과물을 생성하도록 지도해야 한다.

반면, 'code.org'사이트는 미국의 Infosys 재단에서 지원하는 온라인 코딩(또는 프로그래밍)교육 사이트로 유아를 포함한 청소년들뿐만 아니라 나이에 상관없이 어른도 쉽게 컴퓨팅 사고를 스스로 학습할 수 있도록 교육환경을 지원하는 곳이다. 이 사이트는 컴퓨터 과학의 중요성을 알리기 위해 컴퓨팅 사고를 일반인들에게 쉽게 접할 수 있도록 하기 위하여 현재 유명한 애니메이션, 만화, 영화, 게임 등의 캐릭터를 사용하여 친숙하게 프로그래밍을 할 수 있도록 지원하고, 자주 업데이트를 실시하고 있다. Fig. 6에서 보는바와 같이 (a)는 학습자를 위한 교육프로그램을 (b)는 교수자가 온라인으로 학습자의 온라인상 학습상태를 관리할 수 있는 학생관리 플랫폼을 (c)는 코딩 플랫폼을 지원하는 것으로 구성되어 있다.

특히 (c)의 구조는 교육용 프로그래밍 언어의 플랫폼과 유사하게 구성되어 있지만, 다른 점은 혼자서 학습할 수 있도록 단순한 문제부터 복잡한 문제 순으로 단계별로 문제가 주어지며, 이 문제를 해결하기 위한 명령어가 'blocks'에 제시된다. 각 단계의 문제는 이전단계까지 수행한 명령어를 그대로 내포하면서 효율적인 프로그램을 완성하도록 고급 명령어들을 제시하여 프로그래밍 능력을 향상시키는데 초점을 두고 있다. 제한된 명령어를 사용하여 코딩을 해야 하는 단점이 있지만 초보자에게 문제를 쉽게 해결할 수 있도록 한다는 점에선 의미가 있다고 볼 수 있다. 또한, 코딩을 한 후 실행 버튼을 누르면 명령어가 실행되는 과정을 시각적으로 활성화 상태를 보여

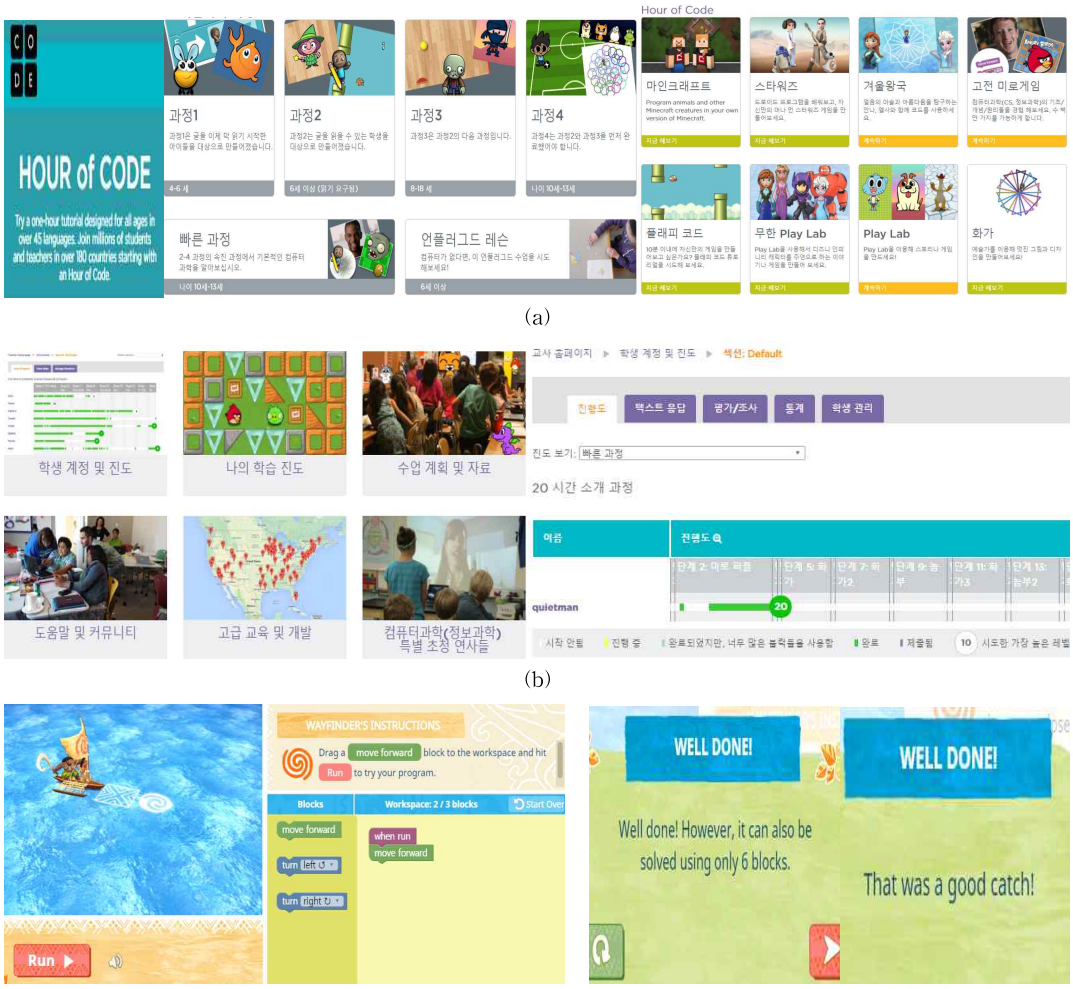


Fig. 6. 'code.org' website (a) for students (b) for teacher (c) platform for coding (d) comments.

주어 현재 어느 명령어를 수행하고 있는지 파악하기가 쉽다. 그리고 문제를 해결하는데 필요한 블럭 수를 제시하고 필요이상의 블록으로 코딩한 경우 경고 문구를 나타냄으로써 학습자가 혼자서 스스로 최적의 코딩을 할 수 있도록 지원하고 있다(Fig. 6의(d)).

4.2 컴퓨팅 사고력 교육과 교수지도 방안

4.2.1 'Hour of code'의 코딩환경의 특징

교수자가 초보자인 초등학생들에게 컴퓨팅 사고력을 촉진시키기 위한 관점에서 'Hour of code'를 활용하여 교육을 할 때 숙지해야 할 사항에 대해 설명하였다.

첫째, 교육용 프로그래밍 언어와 'Hour of code'와 가장 다른 점은 교육용 프로그래밍 언어는 알고리즘을 설계한 것을 시스템에서 실행하기 위해 코딩을 지원하는 언어로서 역할을 한다. 이 의미는 학습자 스스로 필요한 명령어를 찾아 코딩하도록 명령어를 제공하는 측면에서 설명한 것이다. 이에 반해 'Hour of Code'는 Fig. 7의 예에서 보는 바와 같이 주어진 문제를 해결하기 위해 필요한 명령어를 선별하여 'blocks'에 제공하고 학습자는 그 명령어 중에서 적합한 것을 선택하여 작업 영역에서 기본 틀로 제공하는 프로그램을 완성하도록 되어있다. 알고리즘의 다양성보다는 주어진 명령어를 사용하여 효율적으로 코딩을 작성할 수 있도록 안내하는 것에 초점을 둔 것이 가장 큰 차이점이다. 이는 이 사이트의 목적인 온

Fig. 7. An example of traditional maze game in 'Hour of code'.

라인상에서 학습자 혼자 스스로 명령어를 익히면서 코딩을 할 수 있도록 환경을 제공해야하기 때문이다.

둘째, 컴퓨팅 사고력은 창의성과 문제해결력에 초점을 두고 있다. 즉, 창의성은 비정형화된 생각으로부터 출발한다. 즉, 학습자의 아이디어로 문제를 해결하는 방법을 찾아내는 것으로부터 출발한다. 이렇게 찾아낸 방법들은 다양할 수밖에 없기 때문에 프로그램 구현 프로세스 과정을 거치면서 효율적인 알고리즘으로 진화하도록 하는 것이다. 그렇기 때문에 알고리즘의 다양성은 학습자의 창의적 사고에서부터 출발하는 것으로 매우 중요한 부분이다. Fig. 4에서 보면 동일한 문제에 대해 세 가지의 알고리즘을 나타내고 있지만 이 문제를 해결하기 위해서는 이외의 다양한 알고리즘이 존재할 수 있다. 이러한 다양한 알고리즘은 프로그래밍가능 해야 한다.

위의 관점에서 'Hour of code'의 특징을 살펴본다. Fig. 8은 '안나, 엘사와 함께하는 코드'에서 제공하는 문제 중에서 난이도가 낮은 사각형을 그리는 문제이다. (a)의 작업영역에서 제시하고 있는 기본 프로그램은 객체의 이동방향을 앞으로 100 만큼 이동한 후 오른쪽으로 90도 방향 돌기하는 명령어를 시작부분의 블록으로 미리 나타내고 있다. 학습자는 그 이후의 알고리즘을 완성하여 실행하면 된다. (b)는 제시한 블록에 따라 코딩한 것으로 가장 올바른 코딩으로 실행 성공을 알려준다. (c)는 제시한 명령어 순서를 사용하지 않고 아래쪽으로 먼저 이동하여 사각형을 그리는 방법도 있기 때문에 이에 따라 코딩한 것으로 기본적으로 제공하는 코드는 삭제되지 않고 그대로 남아있지만 실행 성공을 알려준다. 기본 코드가 작업영역에 남아있는 이유는 이 사이트는 효율적인 알고리즘을 코딩하도록 안내하기 위한 것으로 해석된다.

나머지 (c)와 (d)는 문제에서 사각형의 모습을 정해놓지 않았을 경우에 그릴 수 있는 이외의 다양한 경우의 알고리즘 중 2개를 코딩한 예이다. 이 두 예는 실행은 되었지만 주어진 문제를 해결한 코드가 아니므로 실패한 코딩으로 간주하였다. 이 예를 코딩한 이유는 이 사이트는 주어진 문제를 해결하기 위해 가장 효율적인 알고리즘을 작성하면서 명령어를 익히도록 하는 것이 주목적으로 학습자의 창의적인 알고리즘을 모두 코딩해서 실행하기 위한 학습에 사용하기에는 극히 제한적임을 보이기 위해서이다.

Fig. 9 는 복잡한 알고리즘을 코딩하는 문제에서 제시하는 블록의 예, 세 가지를 나타내고 있다. 이 문제들은 반드시 주어진 명령어에 의해서만 코딩을 해야 하는 제한점을 갖고 있다. 난이도가 낮은 문제는 순차(sequence statement) 또는 간단한 반복(repeat statement) 정도를 사용하기 때문에 코딩을 변형하여도 순서나 반복횟수, 방향등 정도의 약간의 변형을 주어도 다음단계를 도달하는데 크게 영향을 주지 않는다. 그러나 조건(conditional statement)이나 반복(repeat statement)을 복합적으로 사용하여 알고리즘을 설계하는 경우에는 어떻게 알고리즘을 설계하느냐에 따라 다양한 알고리즘으로 설계할 수 있다. 그런데 예에서 보면 조건문의 유형이나 반복의 조건이 이미 정해져 있어서 제시하지 않은 조건문을 사용하여 창의적으로 코딩할 수 없는 제한점을 갖는다. 그러나 여기서 제한하는 명령어를 사용하여 코딩은 완성할 수 있지만 그 알고리즘이 최적의 해결방법이라고 볼 수 없다. 이유는 복잡한 알고리즘일수록 명령어의 구성이 보다 복잡해짐으로 그 효율성은 다

Fig. 8. An example of Frozen animation game in 'Hour of code' (a)Drawing a quadrangle (b)coding 1 (c)coding 2 (d)coding 3.

각적인 요소를 고려해서 결정해야하기 때문이다.

셋째, 이 사이트에서 코딩하는 과정은 먼저 문제 제시와 문제를 해결하기 위한 명령어와 힌트를 제시하고 학습자가 주어진 명령어를 사용하여 코딩을 완성한 후 실행하도록 되어있다. 만일 실행 오류가 발생하면 오류에 대한 코멘트를 제시하면서 코딩을 수월하게 할 수 있도록 도와주는 과정으로 이루어져있다. 이 때, 학습자는 주어진 명령어를 작업영역에서 제공하는 기본 틀에 직관적으로 순서를 생각하면서 코딩을 완성하게 된다. 난이도가 낮은 경우에는 직관적으로 코딩을 완성할 수 있지만 문제의 난이도가 높아질수록 명령어를 다양하게 사용하여 코딩을 완성해야하기 때문에 직관적인 코딩보다는 알고리즘 설계가 더욱 더 중요하게 된다. 하지만 학습자는 온라인으로 혼자 학습하는 환경이기 때문에 코딩이 완성될 때 까지 직관적인 코딩을 반복하게 된다. 즉, 알고리즘 설계보다는 프로그램의 실행 여부에만 집중하게 되고 최종 단계에 도달하는 것에만 관심을 둘 수 있다는 것이다.

Fig. 9. An example of blocks.

4.2.2 교수지도에 대한 제언

컴퓨팅 사고력은 프로그래밍하는 자체를 의미하는 것이 아니라 프로그램 구현 프로세스 전체과정을 의미함을 앞에서 이미 설명하였다. 따라서 교수자는 이 프로세스를 모두 포함하는 교수설계를 해야만 한다. 이 프로세스에 따라 초등학생들에게 프로그래밍 교육을 하려면 주어진 문제에 대해 학습자 스스로 생각한 문제해결 과정을 순차적으로 표현할 수 있도록 개념과 원리 그리고 절차를 지도하는 것이 매우 중요하다. 이것을 프로그램으로 표현하는 것이기 때문에 프로그래밍 언어를 익히는 직관적인 코딩교육

은 학습자의 창의성이나 문제해결력을 간과한 교육으로 볼 수 있으며 지양해야 한다.

또한, 교수자는 알고리즘을 프로그래밍 언어로 표현하는 과정에서 특정영역에 국한된 전용 프로그래밍 언어보다는 범용언어를 선택해야 한다. 그 이유는 일부 특정 교육용 프로그래밍 언어 중에서는 기본 프로그래밍 구조의 명령어를 지원하지 않는 언어도 있기 때문이다. 그리고 출력형태를 다른 기기들의 동작으로 표현하는 피지컬 컴퓨팅을 위해 어떤 장치를 선택할 것인가 보다는 문제를 어떤 방법으로 해결해서 표현했는가에 초점을 두어야 한다.

같은 맥락에서 'Hour of code' 는 초보자인 초등학생에게 알고리즘에 대한 교육을 하지 않고 바로 투입하는 것은 지양해야 한다. 알고리즘에 대한 개념이 형성되지 않은 상태에서 투입하게 되면 직관적인 코딩과 각 단계에 도달하는 것에만 관심을 갖게 되어 프로그래밍 교육에 대한 오 개념을 갖게 할 수 있기 때문이다. 특히, 초등학생들에게 인기 있는 캐릭터를 사용하여 흥미롭게 구성하였기 때문에 컴퓨팅 사고력을 스스로 깨우치는 것은 무리가 있다.

만일, 알고리즘에 대한 교육을 했다고 하더라도 이 사이트는 효율적인 코딩을 지향하기 때문에 주어진 문제를 학습자의 다양한 알고리즘으로 표현하기에는 제한점이 있다. 오히려 이 사이트에서 제공하는 코딩이 정답인 것으로 생각할 수 있기 때문이다. 이 또한 오 개념을 갖게 할 수 있다. 따라서 알고리즘 설계와 교육용 프로그래밍 언어를 통해 기본적인 학습을 진행하면서 학습한 명령어 또는 문제해결 방법을 잘 이해하고 있는지를 확인하기 위한 매개체로 병행하여 이 사이트를 활용하는 것이 보다 효과적일 것이다. 다시 말해서 프로그램 구현 프로세스 중에서 알고리즘의 원리와 프로그래밍 구조를 이해하고 있는지 확인하는 보조적 도구로 활용하는 것이 가장 효과적일 것이다.

5. 결 론

프로그래밍 교육에 대해 초보자인 초등학생들에게 컴퓨팅 사고에 대한 교육은 교수자가 선택한 프로그래밍 환경과 교수방법에 따라 교육효과에 영향을 주게 된다. 본 논문에서는 두 가지로 나누어 제언하였다. 첫 번째는 컴퓨팅 사고력은 학습자가 갖고 있는 창의력을 촉진시키고 문제 해결력을 향상시키기 위해 매우 중요한 요소이며, 이를 신장시키기 위해서는 반드시 프로그램 구현 프로세스에 의해 교육이 이루어져야 함을 강조하였다. 두 번째는 앞에서 제언한 내용과 연계하여 온라인코딩교육 환경을 선택하여 프로그래밍 교육을 하는 교수자에게 적합한 교수지도 방안에 대해 제언하였다. 이 사이트는 학습자 혼자 온라인을 통하여 코딩에 대해 쉽게 배울 수 있는 것을 목적으로 하고 있기 때문에 알고리즘의 다양성보다는 제시된 명령어를 활용하여 주어진 기본 프로그램을 완성하여 효율적인 알고리즘 틀을 완성하도록 지원하고 있다. 이 환경은 초보자인 초등학생 혼자 학습하면서 컴퓨팅 사고력을 스스로 깨우치기 어렵기 때문에 자칫 직관적인 코딩을 통해 실행 여부에만 관심을 갖게 되어 프로그래밍 교육에 대한 오 개념을 형성시킬 수 있다. 따라서 이 사이트는 프로그램 구현 프로세스를 통한 학습과정에서 알고리즘의 개념 또는 원리와 프로그램의 구조에 대한 이해를 확인하는 보조적인 도구로써 활용하는 것이 효과적임을 사이트의 특징 분석을 통하여 제언하였다. 이 제언은 이 사이트를 활용하여 프로그래밍 교육을 시도하는 교수자에게 수업을 설계할 때 고려해야 할 지침으로 기여하는데 의미가 있을 것이다.

REFERENCE

[1] Y. Choi, T. Han, and S. Lim, *Understanding Computer & Information Technology*, Life and Power Press, 2016. Kyounggi-do

[2] T. Lee and H. Choi, *Informatic Education*, Hanbit Academy Ins. 2015. Seoul

[3] H. Kwon, "A Study on Convergence Education of IT & Design for Training Creative Talent," *Journal of Korea Multimedia Society*, Vol. 17, No. 11, pp. 1354-1362, 2014.

[4] M. Goldweber, J. Barr, and E. Patitsas, "Computer Science Education for Social Good," *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, pp. 15-16, 2013.

[5] V. Bennett, K. Koh, and A. Repenning, "Computing Creativity: Divergence in Computational Thinking," *Proceeding of the 44th ACM Technical Symposium on Computer Science*

Education, pp. 359-364, 2013

[6] H.C. Webb and M.B. Rosson, "Using Scaffolded Examples to Teach Computational Thinking Concepts," *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, pp. 95-100, 2013.

[7] Korea Education and Research Information Service, "Education Information," *Proceeding of KERIS Education Information Symposium*, pp23-53, 2014.

[8] Ministry of Education, *Software Education Operating Guideline*, 2015.

[9] Ministry of Education and Korea Education and Research Information Service, 2015 *Training Program of Teacher Training Institutes*, 2015.

[10] Code(2016) http://code.org (accessed Jan., 1, 2016).

[11] J.M. Wing, CT and Thinking about CT, Csrnegie Mellon University, https://www.cs.cmu.edu/afs/cs/usr/wing/www/talks/ct-and-tc-long.pdf (accessed Jul., 1, 2016).

[12] Computer Science Teachers Association, Computational Thinking Leadership Toolkit 1st Edition, http://csta.acm.org/(accessed Jul., 1, 2016).

[13] R.S. Pressman, *Software Engineering*, Mcgraw Hill, 2013. Singapore

[14] S. Ian, *Software Engineering 10th ed.*, pearson, 2015. Singapore

[15] Y. Jung, J. You, J. Lim, and Y. Son, *Software Education*, Cimass, 2015. Seoul

[16] Scratch(2016) https://scratch.mit.edu/ (accessed Jul., 1, 2016).

[17] Entry(2016) https://playentry.org/ (accessed Jul., 1, 2016).

임 화 경

1998년 서강대학교대학원 컴퓨터공학과 공학박사

2017년 현재 부산교육대학교 컴퓨터교육과 교수

관심분야 : 소프트웨어 교육 및 교수법, 컴퓨터 교수방법 및 설계, 창의성과 컴퓨팅적 사고, 디지털융합 교육