



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

초
등
수
학

교
과
서
상
의

C
o
m
p
u
t
a
t
i
o
n
a
l

T
h
i
n
k
i
n
g

장
면

분
석

고
혜
영

2
0
1
8

석사학위논문

초등수학 교과서상의
Computational Thinking 장면 분석

Analysis of Computational Thinking
in Elementary Math Textbooks

제주대학교 교육대학원

초등수학교육전공

고혜영

2018년 8월

석사학위논문

초등수학 교과서상의
Computational Thinking 장면 분석

Analysis of Computational Thinking
in Elementary Math Textbooks

제주대학교 교육대학원

초등수학교육전공

고혜영

2018년 8월

초등수학 교과서상의
Computational Thinking 장면 분석

Analysis of Computational Thinking
in Elementary Math Textbooks

지도교수 최 근 배

이 논문을 교육학 석사학위 논문으로 제출함

제주대학교 교육대학원


초등수학교육전공

고 혜 영


2018년 5월

고 혜 영의

교육학 석사학위 논문을 인준함

심사위원장 김 해 규 

심사위원 현 종 익 

심사위원 최 근 배 

제주대학교 교육대학원

2018년 6월

목 차

국문 초록	iv
I. 서론	1
1. 연구의 필요성	1
2. 용어의 정의	2
II. 이론적 배경	4
1. Computational Thinking	4
2. Computational Thinking 교육 현황	7
3. 수학적 사고	10
III. 연구 방법 및 분석	18
1. 연구 내용 및 방법	18
2. 초등 수학 교과서 분석	19
IV. 결론 및 제언	50
1. 요약 및 결론	50
2. 제언	53
참고 문헌	54
ABSTRACT	56

표 목 차

〈표 II-1〉 CT의 구성요소	16
-------------------------	----

그림 목 차

[그림 II-1] CT의 6가지 특성	5
[그림 II-2] Wing의 CT퓨터 사고(CT)의 주요 구조	6
[그림 II-3] 컴퓨터 사고의 핵심 아이디어	7
[그림 II-4] 수학적 문제 해결력을 신장시키기 위한 교수· 학습에서의 유의사항	11
[그림 II-5] 세가지 관점에 따라 분류한 수학적 사고	12
[그림 II-6] Computational Thinking과 수학적 사고	17
[그림 III-1] 추상적 사고 장면(수 개념 형성)	19
[그림 III-2] 추상적 사고 장면(도형 개념 형성)	20
[그림 III-3] 추상적 사고 장면(두 수 사이의 대응 관계 확인)	20
[그림 III-4] 추상적 사고 장면(짝수, 홀수 개념 형성)	21
[그림 III-5] 추상적 사고 장면	22
[그림 III-6] 알고리즘 사고 장면	23
[그림 III-7] 알고리즘 사고 장면(발문 1, 2, 3)	24
[그림 III-8] 논리적 사고 장면	26
[그림 III-9] 논리적 사고 장면(도형 성질 추론 과정)	26
[그림 III-10] 분석적 사고 장면	27
[그림 III-11] 분석적 사고 장면(소수의 자릿값 확인)	28
[그림 III-12] 여러 사고들과 복합적으로 사용되는 분석적 사고 장면	29
[그림 III-13] 비판적 사고 장면	30
[그림 III-14] 재귀적 사고 장면	31

[그림 III-15] 문제 해결 전략의 하위 사고 장면	32
[그림 III-16] 2학년, 4학년 교과서에 나오는 수 크기 비교 문제	35
[그림 III-17] 단순한 수 크기 비교 Python 코드와 그 결과	36
[그림 III-18] 자릿값 비교를 적용한 수 크기 비교와 Python 코드와 그 결과	36
[그림 III-19] 짝수와 Python 코드	37
[그림 III-20] 다양한 짝수 Python 코드	37
[그림 III-21] 덧셈 첨가유형과 Python 코드	38
[그림 III-22] 덧셈 합병유형과 Python 코드	38
[그림 III-23] 뺄셈 구산유형과 Python 코드	39
[그림 III-24] 뺄셈 비교유형과 Python 코드	39
[그림 III-25] 곱셈 Python 코드	40
[그림 III-26] 동수누가 곱셈 Python 코드	41
[그림 III-27] 포함제 나눗셈과 Python 코드	42
[그림 III-28] 등분제 나눗셈과 Python 코드	42
[그림 III-29] 약수와 Python 코드	43
[그림 III-30] 배수와 Python 코드	43
[그림 III-31] 최대공약수와 Python 코드	44
[그림 III-32] 최소공배수와 Python 코드	46

[그림 Ⅲ-33] 정다각형과 Python 코드	47
[그림 Ⅲ-34] 선분으로 여러 가지 모양 만들기 와 Python 코드	49

국 문 초 록

초등수학 교과서상의 Computational Thinking 장면 분석

고 해 영

제주대학교 교육대학원 초등수학교육전공
지도교수 최 근 배

본 연구는 초등 수학 교과서에서 Computational Thinking과 관련 지을 수 있는 장면들을 찾아보고 분석하여 일반적이고 보편적인 SW 교육의 일환으로 CT교육에 대한 접근 가능성을 알아보는 데 그 목적이 있다.

이를 위하여 2009, 2015 개정 교육과정과 교과서를 살펴보며 수학 교과서에 반영된 CT 장면을 찾아 수학적 사고와 CT의 공통 영역에서 어느 부분에 해당되는지 분석하였다. 또한, 수학 교과서에 제시되어있는 문제를 컴퓨터 프로그래밍 언어인 Python으로 코딩하여 수학 교과 지도 시 CT를 염두하여 지도하였을 때 학습자들이 얻을 수 있는 교육적 효과를 탐색하였다.

초등 수학 교과서 내에서 추상적 사고 장면, 알고리즘적 사고 장면, 논리적 사고 장면, 분석적 사고 장면, 비판적 사고 장면, 재귀적 사고 장면, 문제 해결 전략의 하위 사고(동시적, 선행적, 전략적, 절차적, 재귀적) 장면 등 CT 장면을 찾을 수 있었다. CT 교육을 수학 교과서와 접목하여 학습할 수 있기 때문에 CT능력 및 수학적 사고를 동시에 함양 할 수 있으며, 간단히 읽고 지나갔던 이야기 마당에서 CT 능력 및 수학적 사고 함양이 크게 이루어질 수 있는 가능성을 발견하였다. 또한, 초등 수학 교과서에 프로그래밍 코드로 작성할 수

있는 수많은 소재들이 존재함을 확인하였고 학생들이 프로그래밍 코드를 작성하는 과정을 통해 학습 내용 복습 및 관계적 이해까지 도달할 수 있다. 즉, 현재 교육과정에서 다루어지는 수학 교과서를 CT 교육 소재로 삼고 CT교육을 자연스럽게 진행할 수 있음을 알 수 있다.

다만, 본 연구에서는 현장 적용 연구가 이루어지지 않았기 때문에 학생들에게 직접 적용해보고 그 학습 결과를 확인해보는 연구가 필요하며 학생들의 CT 능력 향상 정도를 측정하는 검사 도구에 대한 연구도 필요하다. 앞에서 언급한 두 가지의 후속 연구가 지속적이며 깊이 있게 이루어진다면 별도로 진행되는 CT 교육이 아닌 학생들이 학교 교육과정을 통해 접하고 이를 통해 자연스럽게 CT 능력이 함양될 수 있을 것이라 기대한다.

주요어 : Computational Thinking, 수학적 사고, 초등 수학 교과서

I. 서 론

1. 연구의 필요성

오늘날 급격히 발달하는 정보통신기와 기술의 보편화로 인해 우리의 삶은 큰 변화를 겪고 있다. 변화의 흐름이 빨라 더 이상 기존에 습득한 지식으로 미래를 대비할 수 없다는 생각이 만연해지고 이에 따라 교육의 패러다임도 바뀌고 있다. 이전 세대에서는 학생들이 학교에서 정해진 교과내용을 잘 습득하고 미래에 활용할 수 있기를 바랐다면, 현재 21세기에서는 변화하는 미래에 대비한 인재를 길러내기 위해서는 과거의 지식 기반 사회와는 다른 새로운 교육 방법이 요구되고 있다. 즉, ‘물고기를 잡아다가 먹여주는 것’이 아니라 ‘물고기 잡는 방법’을 가르쳐야 한다 것이다. 한 가지 상황에서만 통용되는 고정된 지식을 배우는 것이 아니라 다양한 상황에서 적용 가능한 방법을 배워 변화무쌍한 상황 속에서 적합하게 변형하여 활용할 수 있게 해야 한다.

이를 위해 우리나라에서도 21세기를 이끌어 갈 창의적 인재를 양성하기 위한 많은 노력을 기울이고 있다. 교육부에서는 21세기가 요구하는 학습자 역량을 강화하기 위해 스마트교육을 도입하였다.(교육과학기술부, 2011) 그러나 이러한 스마트교육은 21세기 학습자 역량을 지닌 창의적 인재를 양성한다는 본래 취지와는 다르게, 태블릿 PC와 같은 스마트기기 보급에만 치중되어 왔다. 이에 개정된 교육과정에서는 우리나라에서도 세계적 추세에 발맞춰 Computational Thinking의 개념을 포함한 소프트웨어 교육을 강조하고 있다.(미래창조과학부, 2013) Computational Thinking에 대한 다양한 정의가 있지만, Wing(2006)에 따르면 "Computational Thinking은 문제를 공식화하고 정보처리 주체가 해결책을 효과적으로 수행할 수 있는 형태로 표현하기 위한 관련된 모든 사고과정이다."라고 정의한다. 즉, Computational Thinking은 실생활에서의 복잡한 문제를 컴퓨터의 과학적 원리를 활용해서 단순화시켜 해결하는 사고 과정을 말한다.

미래창조과학부에서 SW교육과 Computational Thinking 능력 습득을 중요시 여겨 강조를 하고 있지만, 현재 학교 현장에서는 Computational Thinking 능력을 기르기 위한 수업 콘텐츠의 보완이 필요한 상황이다. 이는 교육 과정에서 컴퓨터는 정규 교과가 아닌 선택 교과이므로, Computational Thinking이 중심이 된 교과 수업이 이루어지기가 어려운 실정이며, 당장 내년부터 초등학교 5, 6학

년 실과 과목에서 이루어져야 할 SW교육을 성공적으로 실시하기 위해 가르칠 만한 전문 교사나 교사 방법에 대한 혼란이 있기 때문이다.

우리나라에서는 Computational Thinking이 컴퓨팅 사고로 번역되면서, Computational Thinking이 컴퓨터 교과 관련 사고일 뿐 수학 교과와의 관련성이나 문제해결로서의 Computational Thinking 자체는 그리 주목을 받고 있지 못하다. 이것은 Computational Thinking을 '수행 주체와 관계없이' 계산 과정의 역량과 한계를 기반으로 하는 사고라는 Wing의 설명과는 배치되는 입장이다. 우리나라에서도 SW 교육의 2018년 정식교과 도입을 결정하고 SW 인재 양성 계획을 실행하고 있지만, 현대시대의 SW 교육의 목적은 일반적이고 보편적인 교육으로서의 접근이 이루어져야 한다. 보편 교육에서 SW교육의 내용과 목표는 기본 사고력, Computational Thinking의 향상에 초점이 맞추어져야 함을 의미한다. 즉, Computational Thinking이 일반적 문제해결 뿐 아니라 전통적인 교과 영역에서의 문제해결에 적용될 수 있으며, 학교교육에서 Computational Thinking 개념은 교과와 통합적으로 접근되어야 한다. 5, 6 학년 교육과정에 배정된 제한된 시간에만 Computational Thinking 교육을 하는 것이 아니라 기존에 있는 정규 교과에서 지속적으로 Computational Thinking 교육을 실현해야 한다. 새롭게 Computational Thinking 교육을 진행하려고 하니 어떤 내용을 가르쳐야 할지, 어떻게 가르쳐야 할지 등 교육과정 구성 자체가 막막하지만, 초등학교 교육과정에서 이미 가르치고 있는 타교과 내 교육내용에서 Computational Thinking 교육을 접목시킨다면 Computational Thinking 교육이 훨씬 수월해질 것이다.

따라서 본 연구에서는 초등학교 수학 교육과정에 반영된 Computational Thinking 장면을 찾고 분석하여 Computational Thinking과 어떻게 연결 지을 수 있는지, 또 이를 통해 학생들이 습득 가능한 다양한 기능 및 교육적 이점은 무엇이 있는지 연구하고 분석해 보고자 한다.

2. 용어의 정의

Computational Thinking에 대한 정의는 다양하고 학자마다 조금씩 다르다. 국내에서 'CT'을 '계산적 사고', '절차적 사고', '컴퓨터적 사고', '컴퓨터 과학적 사고' 등으로 번역해서 사용하고 있으며, 이 중 가장 널리 사용되고 있는 용어는 '계산적 사고'이다. 하지만 계산적 사고라고 번역하여 Computational

Thinking을 접하였을 때 번역 용어가 주는 기능에만 한정하여 Computational Thinking을 생각하는 경향이 있다. CT는 절차나 알고리즘에 관한 것만도 아니고 계산하고만 관련된 기능도 아니다. 또한 ‘컴퓨터적 사고’라고 번역하였을 때 컴퓨터하고만 관련된 사고라고 생각하기 쉽기 때문에 본 연구에서는 번역 용어를 사용하지 않고 Computational Thinking(이하 CT)이라는 용어를 그대로 사용하여 CT의 포괄적 기능 및 사고를 모두 포함하여 전달하고자 한다.

Ⅱ. 이론적 배경

1. Computational Thinking

가. Computational Thinking의 역사

CT는 새로운 것이 아니다. CT는 역사적으로 적어도 1960년대로 거슬러 올라간다. Alan Perlis는 프로그래밍과 계산의 이론은 모든 분야의 대학생들이 배울 필요가 있다고 주장했다.(Grover, Pea, 2013) 이는 대학생들을 대상으로 한 CT 교육에 대한 언급이었고, 대학교육 이전 K-12교육의 맥락에서 CT는 1980년대에 Seymour Papert가 MIT 논문 ‘Mindstorms: Children, Computers, and Powerful Ideas(마인드스톰: 어린이, 컴퓨터, 강력한 아이디어)’을 내면서 인지되기 시작했다. Papert는 어린이를 대상으로 로고 프로그래밍 언어를 만들었고 이를 통해 절차적 사고를 개발하는 부분을 개척했다(Papert, 1980,1991). Papert가 ‘프로그래밍’이라는 컴퓨터용 언어를 좀 더 강조하였다면, 현재 많이 언급되고 전세계적으로 교육과정에 포함시키고자 하는 CT에 대한 언급 및 개념 확산은 2006년 Jeannette Wing에 의해서 이루어졌다. Wing은 “모두가 기본적으로 갖추어야 할 기초 소양인 읽기(Reading), 쓰기(wRiting), 산수(aRithmetics)처럼 CT도 선택이 아니라 모두가 갖추어야 할 기본 소양이다.”라고 ACM기사에 주장하며 ‘CT’라는 용어가 대중화되기 시작했다.(Wing. 2006)

나. Computational Thinking 정의

Wing은 CT를 “정보처리 주체가 효과적으로 해결책을 얻기 위해 문제를 공식화 및 수행할 수 있는 형태로 표현하기 위한 관련된 모든 사고과정이다.”라고 정의하였다.(Wing, 2006) 위키백과에서는 컴퓨터 과학자들이 사용하는 기법을 사용해서 문제를 해결하는 방법(위키백과, 2018)이라고 정의하였다. 그 외에도 Conery는 CT를 문제와 그 해결 방법을 형식화하는 사고의 과정(thought process)이라고 정의하였고(Conery 외, 2010), Gerald Sussman(2010)은 CT는 일을 처리하는 정확한 방법을 공식화하는 과정이라고 보았다. 또한, Peter Lee(2010)는 CT는 인간 지능을 확대하여 실제적인 적용을 할 수 있는 인간 지능의 메커니즘에 대한 연구이며 이것은 인간의 정신적 능력에 관한 복잡도를 관리하거나 일을 자동적으로 처리하도록 하는 추상화 도구를 통해 확장하는 것이라고 하였다. Andrew McGettrick(2010)은 비슷한 논조로 CT는 생각의 과정

과 함께 그것을 실제적으로 구현할 수 있는 능력도 포함되어야 한다고 주장하였으며 김태영은 CT란 교과를 초월한 실생활의 문제 상황에서 추상화, 자동화 등의 정보과학적 원리를 바탕으로 한 자료구조, 알고리즘, 프로그래밍 등의 컴퓨터로 구현 가능한 방법으로 창의적으로 문제를 해결하는 방식이라고 정의하였다.(김태영, 2014)

다. Computational Thinking의 특징

CT는 큰 복잡한 작업을 공격하거나 큰 복잡한 시스템을 설계할 때 추상화와 분해를 사용하는 것이다. 이는 모든 세부 사항을 이해하지 않고도 크고 복잡한 시스템을 안전하게 사용하고, 수정하고, 영향력을 행사할 수 있게 한다. 다음과 같이 CT는 6가지 특성을 지닌다.(Wing, 2006)

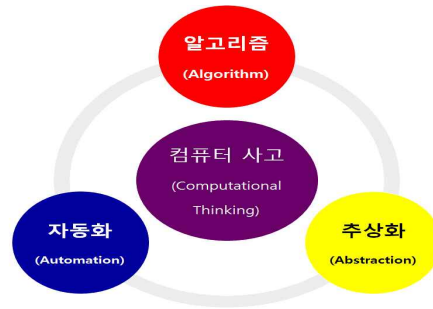


[그림 II-1] CT의 6가지 특성

라. Computational Thinking의 주요 구조(3A)

CT는 문제와 해결책의 설계와 분석에 사용되며, 21세기 새로운 읽고 쓸 줄 아는 기초 소양 능력이다. CT의 본질은 어떤 문제에 직면했을 때 컴퓨터 과학

자처럼 생각하는 것이며, 이는 컴퓨터 과학자들만을 위한 것이 아니라 모든 사람을 위한 근본적인 기술이다.(Wing, 2006) Wing(2006)은 컴퓨터 사고가 세계의 주요 구조, 즉 알고리즘(Algorithm), 추상화(Abstraction) 및 자동화(Automation)를 포함하고 있다고 주장했다.



[그림 II-2] Wing의 컴퓨터 사고(CT)의 주요 구조

알고리즘(Algorithm)은 단계별 지침의 집합이다. 추상화(Abstraction)는 문제 해결 프로세스를 일반화하고 유사한 문제로 전환하는 작업이다.(Barr, Stephenson, 2011) 추상화는 한 문제를 일반화하고 다른 유사한 문제들로 해결책을 옮기는 능력을 포함한다.(Yadav, Hong, Stephenson, 2016) 이는 병렬 처리 방식과 연결된다. 병렬 처리 방식은 코드를 데이터로 해석하고 데이터를 코드로 해석하는 것이다. CT는 알고리즘 사고와 병렬적 사고를 포함하고 있으며, 이는 구성적 추론, 패턴, 매칭, 절차적 사고, 재귀적 사고와 같은 다른 종류의 사고 과정과 결합하여 논리적 사고와 절차적 사고와 겹치는 부분이 있다. 마지막으로 자동화(Automation)는 디지털 및 시뮬레이션 도구를 사용하여 문제 해결책을 기계화하는 작업이 포함된다. CT의 3A 중 알고리즘과 추상화는 교실에 포함시킬 수 있는 핵심적인 컴퓨터 사고 아이디어이다.(Wing, 2006)

마. Computational Thinking의 핵심 개념

컴퓨터 사고를 K-12에 더 적용 가능하게하기 위한 노력으로, 컴퓨터 과학 교사 협회(CSTA)와 국제 교육 기술 협회(ISTE)는 9개의 핵심 CT개념과 2011년 기준을 포함하는 컴퓨터 사고의 운영 정의를 개발했다. 이러한 계산적 사고의 핵심적인 아이디어에는 데이터 수집(data collection), 데이터 분석(data analysis), 데이터 표현(data representation), 문제 해결(problem decomposition), 추상화(abstraction), 알고리즘 및 절차(algorithms & procedures), 자동화(automation), 병렬화(parallelization) 와 시뮬레이션(simulation)이 포함된

다.(Yadav, Hong, Stephenson, 2016)



[그림 II-3] 컴퓨터 사고의 핵심 아이디어
(컴퓨터 과학 교사 협회(CSTA)와 국제 교육 기술 협회(ISTE))

2. Computational Thinking 교육 현황

가. 국외 Computational Thinking 교육

CT는 학생들이 이미 배우고 있는 과목 영역의 맥락에서 아이디어와 원리를 컴퓨터로 이해할 수 있도록 해 주는 포괄적인 접근법을 제공한다.(장경운, 2017) CT의 본질은 복잡한 문제를 더 친숙한 문제로 세분화하고(문제 해결)컴퓨터 문제를 효율적으로 해결하고(알고리즘) 이와 유사한 단계를 사용하여 문제를 해결하고, 해결하는 방법을 검토하는 것이다.(Yadav, Hong, Stephenson, 2016) 미국과 해외의 많은 대학들은 컴퓨터 과학과 관련한 학부 과정을 재검토하고 있다. 많은 사람들이 프로그래밍이 아닌 컴퓨터와 관련된 기본적인 원리와 개념을 다루기 위해 컴퓨터 공학 과정을 바꾸고 있으며, 최근 카네기 멜론 대학에서는 1학년 과정을 수정하여 비전공자들에 대한 CT를 장려하고 있다.(Wing, 2010)

Wing(2010)은 컴퓨터 과학 전공자뿐만 아니라 비전문가도 CT를 이용할 수 있게 해야 하며, 학생들을 컴퓨터로 계산하는 방법과 모델에 노출시켜야 한다고 주장했다. 또한, CT가 모든 사람들에게 당연한 사고로 받아들여지는 것을 목표로 컴퓨터 과학의 즐거움, 경외, 힘을 지속적으로 전파해야 한다고 주장하였다.(Wing, 2006)

컴퓨터 사고를 둘러싼 흥미와 흥분은 연구와 학부 교육 수준을 뛰어넘어 성장해 왔다. CT가 광범위하게 사람들에게 전파할 수 있는 잠재력은 최근 여러 가지 프로젝트에 동기를 부여했다. 프로젝트 대부분은 유치원부터 고등학교까지의 과정에 주력하고 있다. CT 프로젝트를 후원하는 후원자들은 전문 기관, 정부, 학계, 업계에 걸쳐 있고 계산적 사고는 전 세계적으로 확산되고 있다.(Wing, 2010)

2006년부터 마이크로 소프트의 도움으로 카네기 멜론 고등학교는 컴퓨터 프로그래밍보다 CT가 학생들에게 줄 수 있는 교육적 시사점이 더 많다는 것을 교사들에게 이해시키고 공론화하기 위해 CS.S 여름 워크숍을 열었다. CS.S는 2007년에 UCLA와 워싱턴 대학으로 확산되었다. 2010년까지, 구글의 후원 하에 CS.S는 미국에서는 20개의 학교로, 유럽, 중동, 아프리카에서는 14개의 학교로 확산되었다.(Wing, 2010) 또한, 최근 CT가 K-12 교육 분야에 걸쳐 주도권을 행사하고 있으며 많은 나라에서 CT와 관련하여 교육개혁이 진행되고 있다.(Tedre, Denning, 2016) 교육개혁의 핵심은 모든 학생들을 위한 21세기의 핵심 기술로서 CT에 초점을 맞춘 것으로 K-12교실에 CT를 포함시키려는 많은 교과과정으로 이어졌다.(Yadav, Hong, Stephenson, 2016)

오랫동안 모범적이고 의무적인 고등학교 CS교과 과정을 자랑해 온 이스라엘과 러시아, 남아프리카, 뉴질랜드, 호주와 같은 국가들은 이미 K-12교과 과정에서 CT에 필요한 공간을 마련했다. NSF(미국의 전국 과학 재단)는 K-14학생 및 교사들의 CT 능력 개발을 위해 21세기 컴퓨팅 교육 프로그램을 시작했다.(Wing, 2006) 좀 더 최근에 영국은 왕립 협회로부터 대담한 2012년 정책 현장에 따라 모든 학생들에게 컴퓨터 사용을 가르치기 위한 프로그램들을 분류했다.(Grover, Pea, 2013)

학생들이 컴퓨터 과학 교과 과정과 프로그래밍 환경의 맥락 안에서 CT를 배우는 것도 중요하지만, K-12 강의실의 제약으로 인해 모든 학교가 독립적인 컴퓨팅 과정에 접근하는 것이 불가능할 수 있다. 그러나 컴퓨터 공학·기계는 교차 학문이고 초, 중등 분야에 걸쳐 다룰 수 있다는 점과 NGSS(차세대 과학 표준) 및 Common Core등의 현재의 교육 개혁은 학생들이 K-12 커리큘럼에서 CT에

노출되어야 함을 강조하고 있다는 점에 초점을 맞추어 교육개혁을 한다면 실패 확률이 줄어들고 성과 가능성을 높일 수 있다. CSTA3ISO/IEC 프레임워크에서 언급된 CT의 9개 핵심 개념과 기능은 CT를 K-12핵심 콘텐츠 영역에 포함시키기 시작할 수 있는 좋은 실마리를 제공한다. 예를 들어, 컴퓨터 사고의 주요 구성 요소 중 하나는 알고리즘을 사용하는 것인데, 이는 문제를 해결하거나 작업을 완료하기 위해 순차적인 단계를 사용하는 것을 나타낸다. 컴퓨터 사고가 문제 해결에 초점을 맞추고 학생들을 자동화할 수 있는 설계 과정에 참여시키는 것을 감안할 때, K-12 교사와 관리자들이 CT 아이디어를 교과 과정과 실습에 포함시키는 방법을 탐색하는 것이 필수적으로 필요하다. 오늘날 교과 과정의 요구 사항을 충족하는 데 따른 어려움을 감안할 때, 컴퓨터를 이용한 사고 아이디어를 교사들이 교실에서 이미 하고 있는 수업과 연결시키는 것이 최선의 접근법이다.(Yadav, Hong, Stephenson, 2016)

컴퓨터로 생각하는 것에 대한 교사들의 이해력을 계발하고 그들의 교과 과정 맥락과의 연결을 강조하는 것이 성공적으로 K-12 교실에서 CT를 포함시키는 열쇠이다. 이를 위해 교사 교육 프로그램에 교사들을 위한 CT 사고를 도입할 필요성에 대한 주장도 나오고 있다. 교사 교육 커리큘럼 내에서, 교사들은 핵심 과정(예: 학습 이론 및 교육 기술 과정)에서 컴퓨터 사고에 대해 배운 다음, 강의 과정을 통해 특정 과목에 컴퓨터 사고가 적용되는 방법에 대한 이해를 넓힐 수 있고 이를 토대로 교사들이 이미 하고 있는 수업에 컴퓨터 사고를 적용시켜 활용할 수 있는 기회의 폭을 넓힐 수 있다. (Yadav, Hong, Stephenson, 2016)

나. 국내 Computational Thinking 교육

교육적 맥락에서 CT에 대한 정의는 매우 다양한데, '컴퓨터 교과'에서 CT를 소개할 때 '문제해결을 위해 컴퓨터를 사용하기 전에 문제 자체와 해결방식을 이해해야 하는데 이 과제에 도움이 되는 기법이 CT다.'라고 언급하고 있다.(장경윤, 2017) 우리나라에서 CT가 컴퓨팅 사고로 번역되면서, CT는 컴퓨터 교과 관련 사고일 뿐 수학 교과와의 관련성이나 문제해결로서의 CT 자체는 그리 주목을 받고 있지 못하다.(장경윤, 2017) 이것은 CT를 '수행 주체와 관계없이' 계산 과정의 역량과 한계를 기반으로 하는 사고라는 Wing의 설명과는 배치되는 입장이다.(Wing, 2006)

그러므로 CT교육에서 특정 프로그래밍 언어를 깊숙이 숙달시키려는 시도는 바람직하지 않다. CT 교육이 전문적인 소프트웨어 개발자를 양성시키려는 교육이 아닌 만큼 프로그래밍 교육용 도구를 적절히 사용하는 것이 바람직하다.(양

단회, 2016) 또한, CT가 일반적 문제해결 뿐 아니라 전통적인 교과 영역에서의 문제해결에 적용될 수 있으며, 학교교육에서 CT 개념은 교과와 통합적으로 접근되어야 한다는 주장이 점차 제기되고 있다.(장경윤, 2017)

우리나라에서도 2018년부터 중학교에서 SW 교육의 정식교과 도입을 결정하고 SW 인재 양성 계획을 실행하고 있다. 현대시대의 SW 교육의 목적은 일반적이고 보편적인 교육으로서의 접근이 이루어지고 있다. 보편 교육에서 SW교육의 내용과 목표는 기본 사고력, CT의 향상에 초점이 맞추어져야 함을 의미한다.(김수환, 2015) 이는 CT가 더 이상 컴퓨터 교과라는 독자적인 과목에서만 길러질 수 있는 것이 아니라 다른 교과에서도 학습을 통해 자연스럽게 길러지고 향상될 수 있음을 시사한다. ‘기본 사고력 신장’과 ‘CT 능력 향상’이라는 보편 교육의 목표를 달성하기 위해 장경윤(2017)은 CT의 핵심 초석으로 분해, 패턴 인식, 추상화, 알고리즘이라고 정리하였다. 김수환(2015)은 CT를 기르는 데 필수적으로 가르쳐야 하는 영역은 알고리즘과 프로그래밍이라고 주장하였다. 또한, 문교식(2013)은 수학교육에서 CT의 사용이나 개발 가능성은 없는가에 관한 문제의식으로부터 CT의 초등교육 활용 방향에 관한 논문을 썼는데 이 논문에서는 33명의 교사를 대상으로 초등교육에서 CT의 활용 방향을 모색하기 위하여 설문조사를 하였다. 설문조사 결과 거의 모든 응답자들이(90.9%) 정규교육에서 컴퓨터 사고 교육의 필요성에 동의하였고, 과반수이상(63.6%)의 교사들이 계산사고의 ‘문제해결을 위한 알고리즘적 분석과 표현’부분이 수학 교과에서 가장 큰 도움을 줄 수 있다고 응답하였다.(문교식, 2013) 장경윤, 김수환, 문교식 논문을 통해 수학에서 문제해결을 위해 자연스럽게 사용하는 알고리즘이 CT 향상에 도움이 되며 수학 교과 지도 과정에서 자연스럽게 CT를 노출시키고 습득할 수 있음을 알 수 있다.

3. 수학적 사고

수학 교육과정 개정은 지속적인 학습량 감소로 이어져 왔으며, 그 배경에는 수학교육에서 지식의 습득보다 수학적 소양과 문제해결을 갖춘 인재 양성에 대한 국가적 관심이 있었다.(장경윤, 2017) 초등 수학 교과의 목표는 초등학교 수학적 지식과 기능을 습득하고 수학적으로 관찰하고 해석하는 능력을 길러 생활 주변에서 일어나는 문제를 합리적으로 해결하며 수학에 대한 긍정적인 태도를

기르는 데 있다. 여기서 수학적 문제 해결 능력은 수학적 지식과 사고력을 이용하여 적극적인 도전 의식을 가지고 스스로 주어진 문제의 해를 찾아내는 능력을 말하며, 1980년대 이후 수학 교육에서 지속적으로 강조되어온 중요한 수학적 능력이다. 수학적 문제 해결력을 신장시키기 위한 유의사항을 살펴보면 문제 해결 능력을 신장시키기 위해 수학적 사고가 필요하고, 수학적 사고는 초등학교 수학 교육이 나아갈 방향에서 강조하고 있을 정도로 중요하다.(교사용 지도서)

- 가. 문제 해결은 전 영역에서 지속적으로 지도한다.
- 나. 학생 스스로 문제 상황을 탐색하고 수학적 지식과 사고 방법을 토대로 해결 방법을 적절히 활용하여 문제를 해결하게 한다.
- 다. 문제 해결의 결과뿐만 아니라 문제 해결 방법과 과정, 문제를 만들어 보는 활동도 중시한다.
- 라. 생활 주변 현상, 사회 현상, 자연 현상 등의 여러 가지 현상에서 파악된 문제를 해결하면서 수학적 개념, 원리, 법칙을 탐구하고, 이를 일반화하게 한다.

[그림 II-4] 수학적 문제 해결력을 신장시키기 위한 교수·학습에서의 유의사항

가. 수학적 사고의 정의

수학적 사고라는 것은 문자 그대로 “수학에 관한 사고”이다. 지금까지 수학적 사고에 대한 연구는 많으나 이를 명쾌하게 정의한 것은 찾기가 어렵다. 강옥기(1990)은 “논리와 직관이 긴밀한 상호작용을 통해 문제를 해결해가는 체계적인 정신 활동”이라고 정의하였고, 강시중(1971)은 “대상을 수학적으로 보고 생각하며 수학을 만들고 다듬어 가는데 근원이 되는 생각”이라 정의하며 수학적 사고를 수학적 문제를 해결해 가는 체계적인 정신 활동이라고 했다. 또, 김응태·박한식·우정호(2007)는 “직관과 논리가 긴밀하게 결합된 사고 작용을 통해 수학적 문제를 해결해 가는 체계적인 정신 활동”이라고 정의하였다. 이러한 다양한 정의를 바탕으로 2009 개정 수학과 교육과정에서는 수학적 사고를 “주어진 문제 상황을 해결하기 위하여 사고하는 능력”이라고 정의하여 교사용 지도서에 “수학적 사고는 수학을 하는 데 있어서 필요하거나 사용될 수 있는 모든 형태의 사고”라고 정리하였다.

나. 수학적 사고의 유형

강시중(1971)은 수학적 사고를 3가지의 관점으로 나누었는데, 그 중 수학의 특성이거나 수학적 방법 면에서 생각되는 수학적 사고로 수학적 특성면에 해당되는 추상화, 형식화, 일반화, 특수화, 계통화, 직관성, 논리성 7가지와 방법면에 해당되는 귀납적 사고, 연역적 사고, 유추적 사고, 공리론적 방법, 구조화의 방법, 확장적 사고 6가지로 총 13가지 유형으로 정리하였다.

- ① 수학교육의 목표 면에서 생각되는 수학적 사고
 - 자주적으로 수리화 하는 일
 - 수리화를 통한 수학의 기초적인 개념·원리·법칙을 이해하는 일
 - 대상을 간결·명확·통합적으로 처리하는 일
 - 논리적으로 사고하는 기능과 태도를 기르는 일
 - 대상을 합리적으로 처리하는 기능을 기르는 일

- ② 수학의 특성이거나 수학적 방법 면에서 생각되는 수학적 사고
 - 수학의 특성면 : 추상화, 형식화, 일반화, 특수화, 계통화, 직관성, 논리성
 - 수학적 방법면 : 귀납적 사고, 연역적 사고, 유추적 사고, 공리론적 방법, 구조화의 방법, 확장적 사고

- ③ 수학의 내용 면에서 생각되는 수학적 사고
수와 연산, 도형, 측정, 확률과 통계, 문자와 식, 규칙성과 함수의 6개 영역에서 본 사고

[그림 II-5] 세가지 관점에 따라 분류한 수학적 사고

강완·백석윤(2003)은 수학적 사고를 수학의 기능적 측면에서 구별할 때 논리적 사고, 추상화, 일반화, 연역적 사고, 귀납적 사고, 유비적 사고 6가지로 정리하였고, 이광호(2007)는 수학적 방법과 관련된 수학적 사고로 귀납적 사고, 연역적 사고, 유추적 사고, 통합적 사고, 발전적 사고, 단순화의 사고, 기호화의 사고, 추상화의 사고, 일반화의 사고, 특수화의 사고로 총 10가지의 사고 유형으로 정리하였다. 또한, 정승연(2008)은 수학적 사고의 사고 작용의 측면에서 논리적 사고, 추상화의 사고, 일반화의 사고, 연역적 사고, 귀납적 사고, 유추적 사고 6가지로 정리하고, 김지은(2012)은 수학적 사고를 기능적 측면에서 귀납적 사고,

연역적 사고, 유추적 사고, 비판적 사고로 4가지 유형으로 정리하였다.

2009 개정 초등학교 교사용 지도서에서도 수학적 사고를 내용과 기능적 측면에 따라 분류하여 정리하였는데, 내용 특성과 관련하여 강시중(1971)이 수학의 내용 면에서 생각되는 수학적 사고와 비슷하게 집합적 사고, 함수적 사고, 대수적 사고, 도형적 사고, 통계적 사고 5가지로 정리하고, 기능적 특성으로 추상화, 일반화, 연역, 귀납, 유비 추리 5가지로 구별하여 총 10가지의 수학적 사고로 정리하였다.

그 중 CT와 연관 지을 수 있는 수학적 사고로 귀납적 사고, 연역적 사고, 유추적 사고, 추상화 사고, 기호화 사고, 논리적 사고, 비판적 사고, 총 7가지로 정하여 아래와 같이 정리하였다.

1) 귀납적 사고

귀납적 사고란 관찰된 개개의 사례를 총괄하고 그 사례의 규정이 필연적으로 거기에서 도출될 일반적인 주장의 판단을 확립하는 추리로 어떤 부류에 속하는 개개의 사례를 모두 열거하여 그것에 대해 각기 단독으로 주장하는 추리이다. 즉, 일반적인 명제나 보편적 원리, 법칙을 전제로 보다 특수하고 개별적인 명제나 또는 특수 원리, 법칙을 이끌어 내는 사고 방법이다. 예를 들면, 삼각형의 내각의 합이 180° 임을 연역적 방법으로 알아보기 위하여 평행한 두 직선과 다른 직선이 만날 때 엇각이나 동위각이 같다는 사실을 활용하여 삼각형의 내각의 합이 180° 임을 보이는 것이다.(지도서)

2) 연역적 사고

연역적 사고란 전체에 관한 지식이나 어느 경우에도 공통인 보편적 법칙으로부터 특수한 사례에 대해서 그 보편적 법칙을 추론하는 과정을 말한다. 즉, 일반적인 명제나 보편적 원리, 법칙을 전제로 보다 특수하고 개별적인 명제나 또는 특수 원리, 법칙을 이끌어 내는 사고 방법이다. 따라서 귀납적 사고와는 역방향으로 활동하는 사고 방법이다. 예를 들면, 삼각형의 내각의 합이 180° 임을 연역적 방법으로 알아보기 위하여 평행한 두 직선과 다른 직선이 만날 때 엇각이나 동위각이 같다는 사실을 활용하여 삼각형의 내각의 합이 180° 임을 보이는 것이다.(지도서)

3) 유추적 사고

유추는 유비 추리의 준말인데 몇 개의 유사점을 기초로 특정한 사실에서 그와 유사한 다른 특수한 사실의 성질을 추론하는 방법으로 수학 학습 활동에서 자주 이용되는 사고 방법이다. 예를 들면, 삼각형의 넓이 공식은 평행사변형의 넓이 공식을 통하여 구하고, 평행사변형의 넓이 공식은 직사각형의 넓이 공식에서 유도된다. 또한 사다리꼴의 넓이 공식은 평행사변형의 넓이 공식을 활용하여 유도됨을 알 수 있다.(지도서)

4) 추상화 사고

추상화 사고란 몇 개의 사물, 현상이나 장면을 모아서 이들 각각의 필요 없는 속성을 제외하고 공통으로 갖고 있는 성질을 추출하는 것을 말한다. 예를 들면, 직사각형이라는 개념을 학습할 때 유리창, 책받침, 책상 등을 보고 어떤 공통의 모양을 확인하여 직사각형이라는 개념을 형성한다. 이와 같은 추상화 사고는 개념형성에서 뿐만 아니라 문제해결에서도 이용된다. 일상의 문제 장면에서 성질을 추상하고 그 의미를 명확히 하여 수학적 문제를 만들거나 그 조건을 이상화하거나 명확히 하여 수학적 처리의 대상이 되는 문제로 다듬어 가는데 이용된다.(이광호, 2007)

5) 기호화 사고

기호화 사고란 문제의 상황을 그림으로 나타내거나 문자 또는 기호를 사용하여 모델링 하는 것을 말한다. 수학적 개념을 명확하고 간결하게, 전체적으로 표현하거나 사고할 수 있게 함으로써 수학의 발전에 지대한 영향을 끼친 수학적 사고이다.

6) 논리적 사고

논리적 사고는 문제해결의 각 단계에서 요구되는 주어진 사건, 사상, 사태들 간의 관계에서 관계를 파악하여 어떤 규칙을 알아내는 경우에 적용되는 능력을 말한다.(안상철,2012)

7) 비판적 사고

주어진 문제 상황에 관하여 그 문제를 이해하고 관련된 정보를 인식, 수집, 조직, 기억 또는 분석하는 활동을 통해 주어진 자료로부터 적절한 결론을 이끌어 내고, 그 자료에서 모순성과 불일치를 판단하는 사고를 말한다.

다. Computational Thinking 과 수학적 사고

초등학교 수학 교육이 나아갈 방향으로 수학적 연결성 추구가 있다. 즉, 보다 적극적으로 실제 생활이나 다른 교과에서 다루는 내용과 연결하여 수학적 지식을 다룸으로써 초등학생의 수학적 안목을 통합적으로 발전시킬 필요가 있다는 것이다. 이는 CT를 독립된 교과와 학습내용으로 가르치기 보다는 수학적 지식과 연결하여 자연스럽게 접하고 향상시킬 수 있다는 점을 시사하는 바이기도 하다. 수학적 연결성 추구를 통하여 수학적 지식을 단지 수학의 테두리 안에서만, 계산의 기능이나 수학적으로 제한된 사고 활동에서만 다루는 것이 아니라 실제 생활 속에서 부딪히게 되는 여러 현상을 수학적 시각이나 방식으로 이해 보는 기회를 얻을 수 있다. 수학적 연결성 추구로 초등 수학의 각 내용 영역마다 다른 교과와 직간접적으로 연결되는 부분은 얼마든지 찾아볼 수 있다. 예를 들어, 백분율을 비롯한 비율 관련 부분은 사회, 과학 교과와 직접적으로 연결된다. 꺾은선 그래프를 비롯한 각종 그래프 관련 내용 역시 사회 현상, 과학 현상을 설명하고 예측하는 데 중요한 수단으로서 기능할 수 있다. 이때, 초등수학과 교수 학습 방법에 나와 있는 교육 기자재 및 컴퓨터 활용에 대한 언급은 다음과 같다.

“계산 능력 배양을 목표로 하지 않는 경우의 복잡한 계산 수행, 수학의 개념, 원리, 법칙의 이해, 문제 해결력 향상 등을 위하여 계산기, 컴퓨터, 교육용 소프트웨어 등의 공학적 도구와 다양한 교구를 활용한다.”

수학의 개념, 원리, 법칙의 이해, 문제 해결력 향상을 위해 사용할 수 있는 컴퓨터가 실질적으로는 복잡한 계산 수행에 도움을 주는 도구적 기계로밖에 사용되지 않았다. 이는 컴퓨터라는 기계에 초점을 맞추어 생각하기 때문이다. 2012 개정 교육과정에서 강조하는 SW교육과 연계하여 CT 능력 향상에 초점을 맞춘다면 ‘CT’ 와 ‘수학적 사고’ 사이에 존재하는 연관성을 찾아 생각해 볼 필요가 있다.

Wing(2006)은 CT의 주요 구성요소로 알고리즘, 추상화, 자동화를 제시하였

다. 이에 유중현(2008)은 논리적 사고를 CT의 하위 사고양식으로 추가하였다. 이은경(2009)은 문제해결과정에 사용되는 CT의 구성 요소를 다음의 표와 같이 체계적으로 제시하였다.

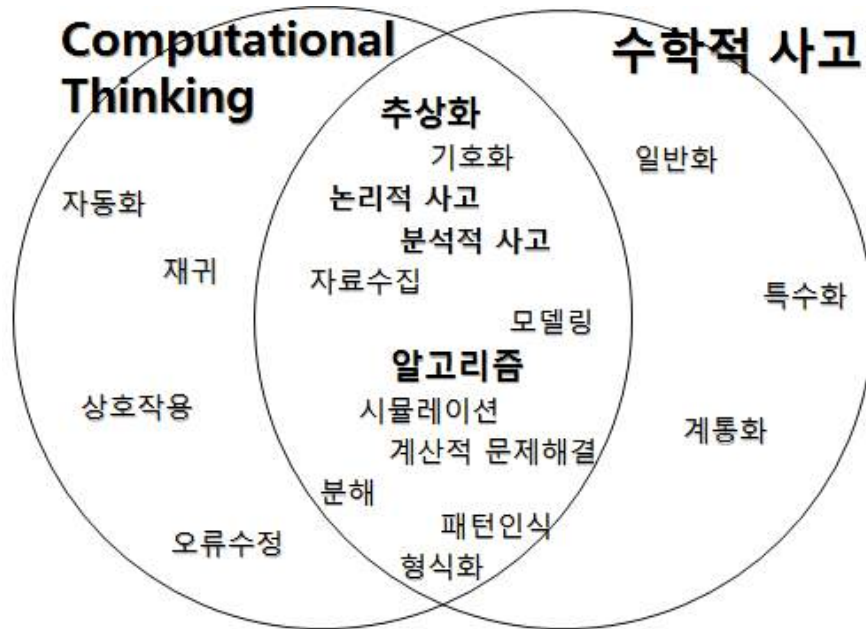
CT				
추상화 능력			자동화 능력	
문제 발견	문제 분석 및 표현	문제 해결 전략	도구 선택	도구 사용
논리적 사고 분석적 사고	분석적 사고 추상적 사고	동시적 사고 선행적 사고 전략적 사고 절차적 사고 재귀적 사고		

[표 II-1] CT의 구성요소

이은경(2009)이 제시한 CT의 구성 요소 중 문제 해결 전략에 들어있는 동시적 사고, 선행적 사고, 전략적 사고, 절차적 사고, 재귀적 사고는 수학적 문제 해결의 사고 과정과 직접적으로 연관되어 있다. Polya는 수학적 문제 해결 과정을 4단계로 나누어 제시하고 각 단계의 유용한 발문과 권고를 정리하였다. 문제의 이해 단계에서 사용하는 대표적인 발문인 “미지는 무엇인가? 자료는 무엇인가, 조건은 무엇인가?”라는 발문은 동시적 사고와 연결 가능하다. 해결 계획의 수립 단계에서 사용되는 “전에 그 문제를 본 일이 있는가? 그렇지 않으면 약간의 다른 문제를 본 일이 있는가?”, “관련된 문제를 알고 있는가?”, “유용하게 쓰일 수 있는 정리를 알고 있는가?”는 선행적 사고에 해당된다. 또한, 선행적 사고를 통해 떠올린 문제 해결 계획은 전략적 사고에 해당된다. 계획의 실행 단계에서 풀이 계획을 실행하고 매 단계를 점검하는 부분은 절차적 사고이다. 마지막 반성 단계를 통해 결과를 점검하고, 결과나 방법을 다른 문제에 활용하면서 4단계를 전체적 혹은 부분적으로 반복함으로써 재귀적 사고와 연결 가능하다.

CT의 구성 요소 중 문제 발견, 문제 분석 및 표현에 포함되어있는 논리적 사고, 분석적 사고, 추상적 사고 역시 수학 교과에서 발견이 가능하다. CT, 수학

적 사고는 서로 연관되며 이들 사이에 공통 영역과 각기 고유한 영역이 있다. [그림 II-7]은 이를 나타낸 것으로 수학적 사고 영역에서 CT 영역과 관련하여 시사점을 얻을 수 있는 부분에 주목하고자 한다.



[그림 II-6] Computational Thinking과 수학적 사고

Ⅲ. 연구 방법 및 분석

1. 연구 내용 및 방법

2009개정 및 2015개정 수학 교과서에서 반영된 CT 장면을 찾아 수학적 사고와 CT의 공통 영역에서 어느 부분에 해당되는지 분석한다. 또한, 수학 교과서에 제시되어있는 문제를 컴퓨터 프로그래밍 언어인 Python으로 코딩하여 초등학교 수학 교수·학습 활동에서 활용하고 고려해볼만한 CT의 접목 가능성을 알아보고자 한다.

여기서 파이썬(Python)이란, 1990년 암스테르담의 귀도 반 로섬이 개발한 인터프리터 언어이다. 우리나라에서는 아직 대중적으로 사용되고 있지 않지만 외국에서는 교육 목적뿐 아니라 실무에서도 많이 사용되고 있다. 그 대표적인 예가 바로 구글이다. 구글에서 만들어진 소프트웨어의 50%이상이 파이썬으로 만들어졌다. 파이썬 프로그램은 공동 작업과 유지 보수가 매우 쉽고 편리하다. 그 때문에 이미 다른 언어로 작성된 많은 프로그램과 모듈들이 파이썬으로 재구성되고 있다.

파이썬의 특징으로는 4가지를 꼽을 수 있다. 첫 번째, 파이썬은 인간다운 언어이다. 사람이 생각하는 방식을 그래도 표현할 수 있는 언어이다. 따라서 프로그래머는 굳이 컴퓨터의 사고 체계에 맞추어서 프로그래밍을 하려고 애쓸 필요가 없다. 두 번째, 파이썬은 문법이 쉬워 빠르게 배울 수 있다. 파이썬은 문법 자체가 아주 쉽고 간결하며 사람의 사고 체계와 매우 닮아 있다. 배우기 쉬운 언어, 활용하기 쉬운 언어이다. 세 번째, 파이썬은 간결하다. 다른 사람이 작업한 소스 코드도 한눈에 들어와 이해하기 쉽기 때문에 공동 작업과 유지 보수가 아주 쉽고 편하다. 네 번째, 파이썬은 오픈 소스로 무료이다. 파이썬의 특징 4가지를 살펴보면 수학 교과서에 나온 문제를 프로그래밍할 언어로 파이썬이 적합하다.

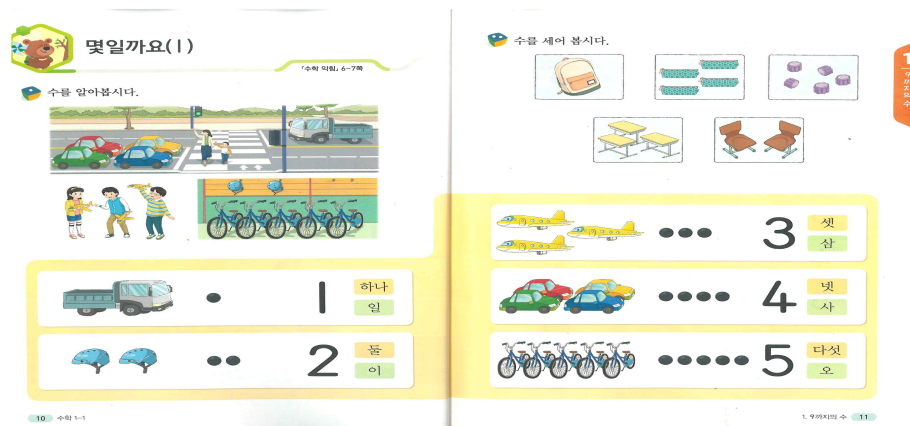
2. 초등 수학 교과서 분석

가. 교과서상의 CT 장면 분석

1) 추상적 사고 장면

초등수학 교과서에서 발견할 수 있는 CT의 구성요소 중에서 추상적 사고는 개념학습을 할 때 찾을 수 있다. 초등학교 1학년 수학에서는 1에서 9까지의 수를 배울 때 여러 가지 구체물을 통해 숫자를 인식하게 한다. 그 후 구체물이 가지고 있는 속성 중 색깔, 크기, 모양 등 서로 다른 요소는 제거하고, 공통적인 성질만을 추출하여 숫자를 익히기 때문에 추상화 사고 장면에 부합된다. 교과서의 장면을 보면 자연수 3은 연필 3자루, 색종이 3장, 사과 3개 등과 같은 구체물의 모임에서 구체물의 속성인 모양, 크기, 색깔 등은 버리고 공통적인 속성을 뽑아 만든 표상을 이상화하여 자연수 '3'(셋, 삼)의 개념을 형성한다.

이와 비슷하게 도형을 학습할 때에도 추상화 사고를 확인할 수 있다. 풀 뚜껑, 동전, 음료수 캔, 컵과 같은 구체물 중에서 모양이라는 속성만 추출하여 원을 학습하고, 삼각형은 삼각자, 옷걸이, 삼각 교통표지판을 통해, 사각형은 액자, 자, 칠판, 창틀, 책과 같은 구체물 중에서 모양과 각에 관심을 두어 학습하는 부분이 추상화 사고에 해당된다.



[그림 III-1] 추상적 사고 장면(수 개념 형성)

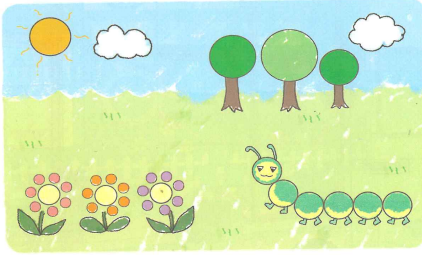


○을 알아볼까요

『수학 익힘』 20~21쪽

○을 알아봅시다.

- 모양을 찾아보세요.



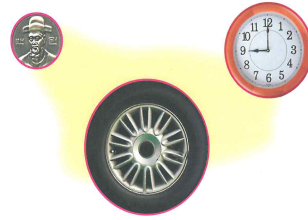
• 찾은 모양을 무엇이라고 할까요?

그림과 같은 모양의 도형을 원이라고 합니다.



30 수학 2-1

• 주변에서 원 모양을 찾아보세요.



[그림 III-2] 추상적 사고 장면(도형 개념 형성)

규칙이 있는 두 수 사이의 대응 관계를 확인하고 식으로 나타내는 장면에서도 추상적 사고를 찾을 수 있다.

생활 속에서 규칙을 찾아 식으로 나타낼 수 있어요
의미적 111~112쪽

예제 1 사람들이 영화를 보고 있습니다. 대응 관계를 찾아 식으로 나타내어 봅시다.

예제 2 팔린 팝콘의 수와 판매 금액 사이에는 어떤 대응 관계가 있는지 알아보시오.
 • 표를 완성하고 팔린 팝콘의 수와 판매 금액 사이에는 어떤 대응 관계가 있는지 설명해 보세요.

팔린 팝콘의 수	1	2	3	4	5	6
판매 금액	3000	6000	9000			

• 팔린 팝콘의 수를 □, 판매 금액을 △라 할 때 □와 △ 사이의 대응 관계를 식으로 나타내어 보세요.

예제 3 팝콘이 80통 팔렸다면 판매 금액은 얼마일까요?

180 수학 4-2

예제 4 영화관의 성인 입장료와 성인 입장객의 수 사이에는 어떤 대응 관계가 있는지 알아보시오.

• 표를 보고 성인 입장료와 성인 입장객의 수 사이에는 어떤 대응 관계가 있는지 설명해 보세요.

성인 입장료	8000	16000	24000	32000	40000	48000
성인 입장객의 수	1	2	3	4	5	6

• 성인 입장료를 □, 성인 입장객의 수를 △라 할 때 □와 △ 사이의 대응 관계를 식으로 나타내어 보세요.

예제 5 성인 입장료가 64000원이었다면 성인 입장객은 모두 몇 명일까요?

예제 6 주변에서 규칙을 찾아 보고 대응 관계를 식으로 나타내어 보세요.

181 6. 규칙과 대응

[그림 III-3] 추상적 사고 장면(두 수 사이의 대응 관계 확인)

영화관의 의자의 수와 팔걸이의 수 사이의 대응 관계를 표로 만들어 정리하면 팔걸이의 수는 의자의 수보다 항상 한 개가 더 많다는 규칙을 발견하여 기호를 이용하여 다음과 같이 식으로 표현할 수 있다.

$$\square = \triangle + 1 \quad (\square = \text{팔걸이의 수}, \triangle = \text{의자의 수})$$

이는 각 구체물에 있는 여러 가지 속성 중 공통적인 속성인 개수의 차이를 추출함으로써 추상적 사고에 해당된다.

CT의 추상적 사고는 수학적 사고의 추상화에만 국한되지 않고 더 포괄적인 의미를 담고 있다. Yadav · Hong · Stephenson(2016)에 따르면 CT의 추상화는 한 문제를 일반화하고 다른 유사한 문제들로 해결책을 옮기는 능력을 포함한다. 이는 수학적 사고 중 일반화와 연결이 가능하다. 수학적 사고의 일반화에 해당하는 추상적 사고는 짝수와 홀수를 학습할 때 확인 가능하다. 수를 학습하면서 둘씩 짝을 지을 수 있는 수를 알아봄으로써 모든 짝수는 2로 나누어 떨어지는 수라는 일반화를 이끌어 낼 수 있다.



[그림 III-4] 추상적 사고 장면(짝수, 홀수 개념 형성)

또한, 삼각형의 세 각의 크기를 각도기로 재어서 세 각의 크기의 합을 구한 후 모든 삼각형 내각의 합은 180° 라는 사실을 일반화할 수 있다. 더 나아가 사각형의 내각의 합은 360° 임을 비슷한 방법으로 이끌어낼 수 있고 이를 활용하여 사각형에서 세 각이 주어지고 주어지지 않은 나머지 한 각의 크기를 알아내는 문제를 해결할 수 있다.

삼각형의 세 각의 크기의 합을 알아봅시다.

- 모든 친구들과 서로 다른 삼각형을 그려 보세요.

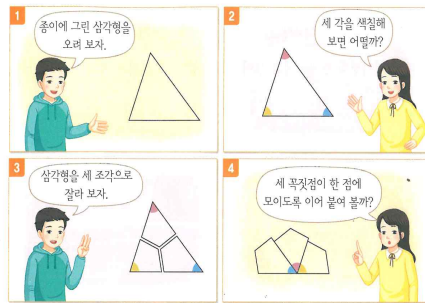
종이에 삼각형을 그려 보세요.



- 삼각형의 세 각의 크기를 각도기로 각각 재어 보세요.

- 삼각형의 세 각의 크기의 합을 구해 보세요.

자신이 그린 삼각형을 이용하여 세 각의 크기의 합을 알아봅시다.



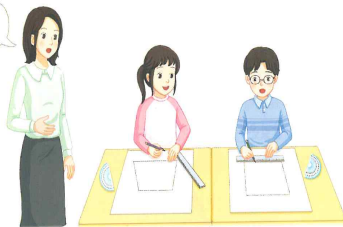
삼각형의 세 각의 크기의 합은 180° 입니다.

▲삼각형 세 내각의 합

사각형의 네 각의 크기의 합을 알아봅시다.

- 모든 친구들과 서로 다른 사각형을 그려 보세요.

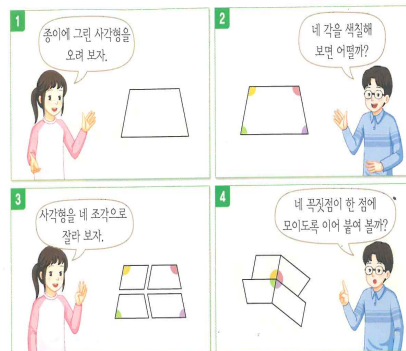
종이에 사각형을 그려 보세요.



- 사각형의 네 각의 크기를 각도기로 각각 재어 보세요.

- 사각형의 네 각의 크기의 합을 구해 보세요.

자신이 그린 사각형을 이용하여 네 각의 크기의 합을 알아봅시다.



사각형의 네 각의 크기의 합은 360° 입니다.

▲사각형 네 내각의 합

[그림 III-5] 추상적 사고 장면

2) 알고리즘적 사고 장면

초등수학 교과서에서 발견할 수 있는 CT의 구성요소 중에서 알고리즘 사고는 덧셈, 뺄셈, 곱셈, 나눗셈 등 계산과정을 형식화하는 부분에서 찾을 수 있다. 알고리즘 사고는 어떤 문제를 해결하는 데 있어서 필요한 절차를 나열하여 정리하는 것이기 때문에 [그림 III-6]과 같은 과정은 알고리즘 사고에 부합된다.

23+19를 어떻게 계산하는지 알아보십시오.

Three ten-frame diagrams illustrating the dot method for 23+19. The first shows 23 and 19. The second shows moving one dot from 19 to 23 to make a ten. The third shows the final sum 42.

▲ 덧셈하는 방법

57×23을 어떻게 계산하면 좋을지 이야기해 보시오.

A sequence of five diagrams showing the chunking method for 57×23. It breaks down 23 into 10, 10, and 3, and calculates 57×10, 57×10, and 57×3 separately.

▲ 곱셈하는 방법

98÷4를 어떻게 계산하면 좋을지 이야기해 보시오.

Three diagrams showing the two-digit number division method for 98÷4. It shows 98 as 80+18, then 80÷4=20 and 18÷4=4 remainder 2.

▲ 나머지가 있는 두자리 수 나눗셈하는 방법

42-17을 어떻게 계산하는지 알아보십시오.

Four diagrams showing the ten method for 42-17. It shows 42 as 30+10, then 30-10=20 and 10-7=3, so 20+3=23.

▲ 뺄셈하는 방법

36÷3을 어떻게 계산하면 좋을지 이야기해 보시오.

Three diagrams showing the no remainder division method for 36÷3. It shows 36 as 30+6, then 30÷3=10 and 6÷3=2, so 10+2=12.

▲ 나머지가 없는 나눗셈하는 방법

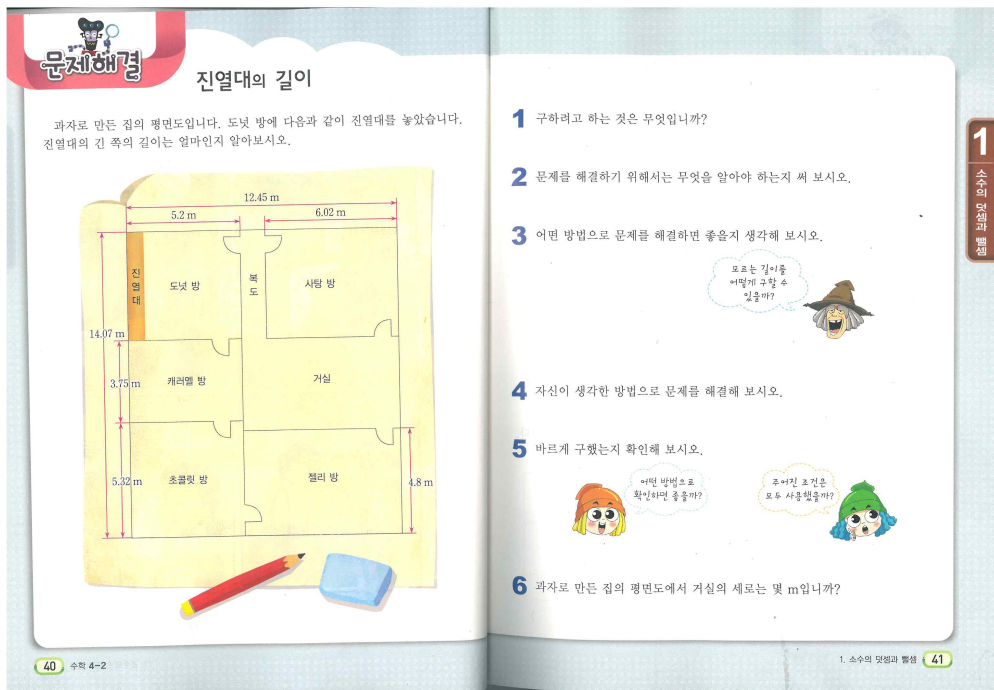
685÷27을 계산하는 방법을 알아보십시오.

Two diagrams showing the three-digit number division method for 685÷27. The first shows 685 as 540+145, then 540÷27=20 and 145÷27=5 remainder 10.

▲ 나머지가 있는 세자리 수 나눗셈하는 방법

[그림 III-6] 알고리즘 사고 장면

알고리즘 사고는 문제를 해결하는 방법에서도 찾을 수 있다. 특히 문장으로 된 문제를 해결할 때 학생들의 문제해결 과정을 돕기 위해 나온 발문들을 확인하면 알고리즘 사고를 명확하게 확인할 수 있다.



[그림 III-7] 알고리즘 사고 장면(발문1, 2, 3)

문제에서 구하려고 하는 것이 무엇인지, 문제를 해결하기 위해 무엇이 필요한지 확인하고 그에 맞는 계산식을 세우는 것은 어떤 문제를 해결하는 데 있어서 필요한 절차를 따른 것이므로 알고리즘 사고에 부합된다고 볼 수 있다.

3) 논리적 사고 장면

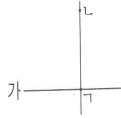
논리적 사고는 이론의 근거를 분명하고 정확히 하는 것으로 가정에서 결론으로 이끌어 가는 과정이 분석적이고 단계적인 특징을 갖고 있다. 초등수학 교과서에서 발견할 수 있는 CT의 구성요소 중에서 논리적 사고는 왜 그렇게 생각하는지 이유를 물어보는 발문 장면에서 확인할 수 있다.

아동 상호 간의 논의나 토론을 통해 논리적 사고력을 육성할 수 있는데, 이미 알고 있는 개념을 설명할 때 근거를 정확하고 분명하게 들어 결론을 이끌어 내

기 때문에 논리적 사고에 부합한다.

● 점 g 과 점 h 을 직선으로 이어 보시오.

● 그은 직선이 직선 g 에 대한 수선인 이유를 이야기해 보시오.



교실이나 운동장에서 평행선을 찾아 보시오.

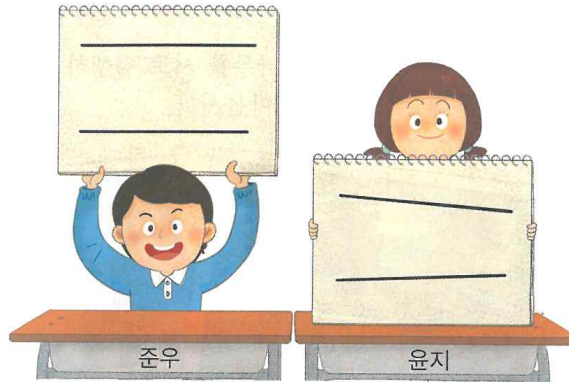
● 평행한 부분을 찾아 보시오.

● 찾은 부분이 평행선인 이유를 이야기해 보시오.

58 수학 4-2



준우와 윤지가 직선을 2개씩 그었습니다. 누가 그은 두 직선이 평행선인지 알아보시오.



- 준우와 윤지의 직선에 각각 수선을 그어 보시오.
- 그은 수선과 두 직선이 서로 수직입니까?
- 누가 그은 두 직선이 평행선입니까?
- 왜 그렇게 생각하는지 이야기해 보시오.

마무리 원기둥을 찾아 보시오.

가 나 다 라

- 원기둥을 찾아 써 보시오.
나
- 원기둥이라고 생각한 이유는 무엇입니까?
① 두 밑면이 2개이고 서로 평행, 합동이다.
② 옆면은 굽은면이고 1개 있다
- 원기둥이 아닌 것은 왜 원기둥이 아닙니까?
가 두 밑면이 원이 아닌 사각형인 사각기둥이다
다 두 밑면이 원이 아닌 오각형인 오각기둥이다 3. 원기둥, 원뿔, 구

73

[그림 III-8] 논리적 사고 장면

또한, 논리적 사고는 주어진 정보를 사용하여 새로운 정보를 찾아내는 장면인 도형의 성질을 추론하는 과정에서도 찾을 수 있다. 평행사변형, 마름모, 직사각형에서 변의 길이나 변의 위치, 각의 크기 등을 분석하여 도형의 성질을 발견하고 있으므로 논리적 사고에 부합된다.

평행사변형의 성질을 알 수 있어요 익힘책 55~56쪽

85 평행사변형 마음 사각형들은 마주 보는 두 쌍의 변이 서로 평행합니다. 평행사변형들의 다른 공통점을 찾아 이야기해 봅시다.

활동 1 평행사변형의 여러 가지 성질을 예상해 보시오.

- 평행사변형은 어떤 성질을 가지고 있을지 예상해 보시오.
- 내 예상과 친구의 예상을 비교해 보시오.

88 수학 4-2

활동 2 평행사변형의 여러 가지 성질을 확인해 보시오.

89 85에서 예상한 평행사변형의 성질을 여러 가지 방법으로 확인해 보시오.

예상한 성질	확인 방법	확인 결과

● 평행사변형의 여러 가지 성질을 이야기해 보시오.

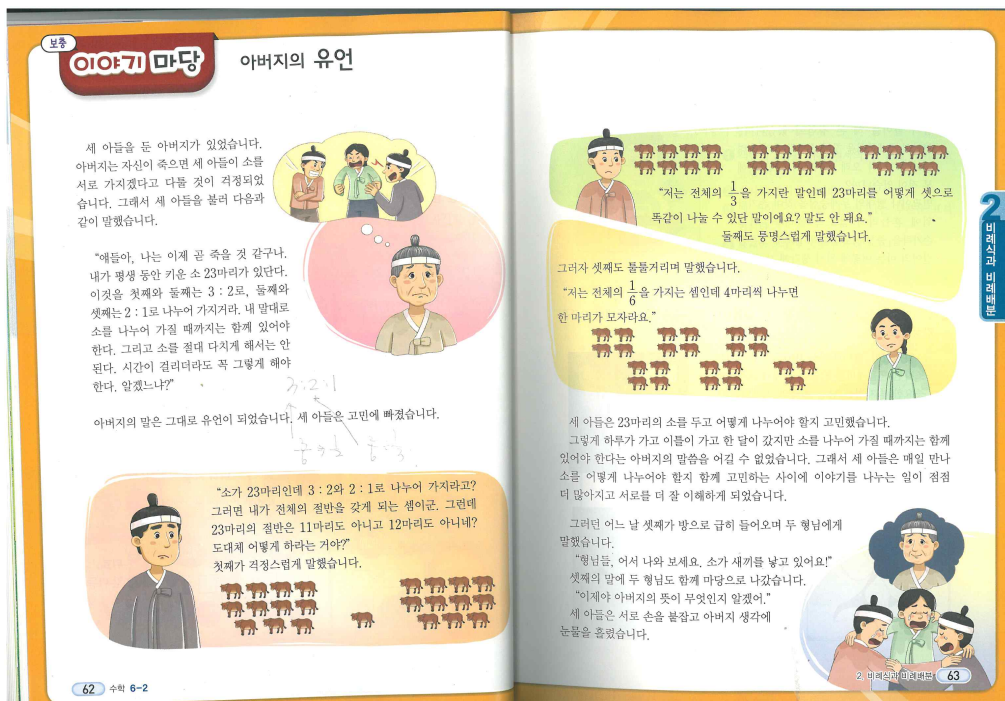
활동 3 평행사변형입니다. □ 안에 알맞은 수를 써 넣으시오.

89 3. 다각형

[그림 III-9] 논리적 사고 장면(도형 성질 추론 과정)

4) 분석적 사고 장면

분석적 사고는 문제해결을 위하여 주어진 단위 요소로서의 자료나 정보간의 논리적 관계를 확인함으로써 기존의 정보를 명료화하는 사고를 말한다. 초등수학 교과서에서 분석적 사고는 문제 해결 과정 전 부분에서 찾을 수 있다. 특히 문장제 문제를 보면 이야기 속 곳곳에 존재하는 문제 해결을 위한 정보들을 파악하고 주어진 조건들 간의 논리적인 관계를 생각해야하므로 분석적 사고를 쉽게 확인할 수 있다.



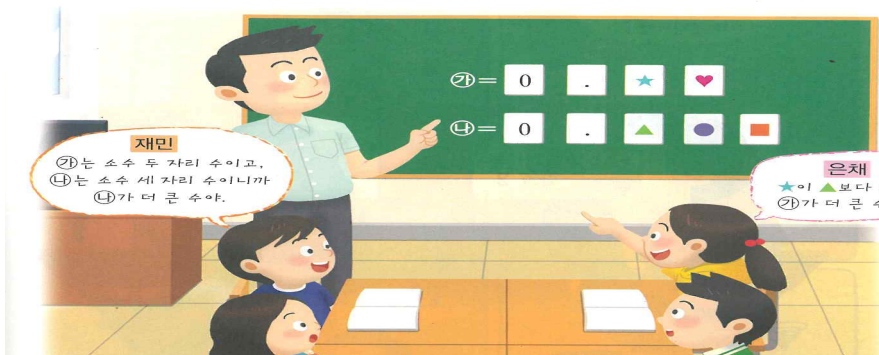
[그림 III-10] 분석적 사고 장면

다음 장면은 6학년 2학기 비례식과 비례배분에 나오는 내용으로 비례배분 문제를 이야기로 만든 것이다. 23마리의 소를 3:2:1로 비례배분하여 나눠 갖기 위해 어떻게 해야 하는지에 대하여 첫째 아들, 둘째 아들, 셋째 아들의 생각 및 말하는 부분에서 문제를 분석하여 해결하는 장면이 담겨있다.

문장제 문제가 아닌 [그림 III-11]과 같은 문제에서도 각 기호마다 소수 몇 번째 자리인지 확인하고 소수 두 자리 수와 소수 세 자리 수의 논리적 관계를 확인하고 있으므로 분석적 사고라 볼 수 있다. 소수 두 자리 수와 소수 세 자리 수를 각각 기호로 나타내어 학생들이 말하는 기호가 소수 몇 번째 자리수인지 확인하고 자릿수에 맞춰 크기 비교를 함으로써 단순 수 크기 비교가 아닌 분석적 사고가 필요하다.

3 선생님께서 칠판에 카드를 붙여 소수를 만드셨습니다. 소수의 크기를 비교하는 방법을 바르게 말한 학생의 이름을 모두 써 보시오.

1 소수의 덧셈과 뺄셈



재민
 ㉔는 소수 두 자리 수이고,
 ㉕는 소수 세 자리 수이니까
 ㉕가 더 큰 수야.

은채
 ★이 ▲보다 크면
 ㉔가 더 큰 수지.

재민
 ㉔는 소수 두 자리 수이고,
 ㉕는 소수 세 자리 수이니까
 ㉕가 더 큰 수야.

정희
 ★과 ▲가 같으면
 ㉔와 ㉕의 크기는
 같은 거네.

은채
 ★이 ▲보다 크면
 ㉔가 더 큰 수지.

준수
 ★과 ▲가 같으면
 소수 둘째 자리 수인 ♥와
 ●를 비교해 봐야 해.

[그림 III-11] 분석적 사고 장면(소수의 자릿값 확인)

분석적 사고는 독립적으로 실현되는 것이 아니라 다른 여러 사고들과 복합적으로 사용되며, 수학 교과서에 나와 있는 문제들을 해결할 때 항상 필요하고 작

용하는 사고이다. 그 예로 [그림 III-12]에서는 분석적 사고, 논리적 사고, 알고리즘적 사고가 동시다발적으로 활용되고 있음을 확인할 수 있다.

- 4 선우는 소수의 덧셈을 하는 데 어려움을 겪고 있습니다. 선우의 계산에서 잘못된 곳을 찾아 바르게 계산하고, 선우에게 풀이 과정을 설명해 보시오.



[그림 III-12] 여러 사고들과 복합적으로 사용되는 분석적 사고 장면

5) 비판적 사고 장면

비판적 사고는 주어진 문제 상황에 관하여 그 문제를 이해하고 관련된 정보를 인식, 수집, 조직, 기억 또는 분석하는 활동을 통해 주어진 자료로부터 적절한 결론을 이끌어 내고, 그 자료에서 모순성과 불일치를 판단하는 사고를 말한다. 초등 수학교과서에서는 Polya는 수학적 문제 해결 과정이 적용된 발문 중 문제에 대한 이해를 하는 첫 번째 단계에 해당이 된다. Polya의 문제 해결 4단계 중 첫 번째 단계에서는 구하고자 하는 것은 무엇인지, 알고 있는 것은 무엇인지 확인하면서 문제 해결과 관련된 정보를 인식하고 적절한 결론을 이끌어 내기 위해 필요한 정보와 그렇지 않은 정보를 구분하고 판단한다.

문제 해결 어떻게 쌓을까

✖ 윤지와 준호는 쌓기나무로 놀이를 하고 있습니다. 준호가 쌓기나무를 쌓을 수 있는 방법이 모두 몇 가지인지 알아보시오.

5개의 빈칸에 쌓기나무 9개를 쌓아 보. 각 칸마다 적어도 한 개의 쌓기나무를 쌓아야 하고, 4층으로 쌓은 칸이 있어야 해.

5개의 빈칸에 쌓기나무 9개?

- 구하려고 하는 것은 무엇입니까?
 - 주어진 조건은 무엇일까요?
 - 각 칸마다 적어도 한 개를 쌓아야 한다는 말은 무엇을 의미합니까?
- 어떤 방법으로 문제를 해결하면 좋을지 생각해 보시오.

어떤 문제를 나타내는 것일까?

간단하게 구할 수 있는 방법이 무엇일까?

26 수학 6-2

- 조건에 맞게 쌓는 방법이 몇 가지인지 빈칸에 나타내어 보시오.

도양을 돌릴 때 같은 모양은 한 가지로 생각해야 돼.
- 조건을 바꾸어 새로운 문제를 만들어 보시오.

어떤 조건을 바꿀 수 있을까?

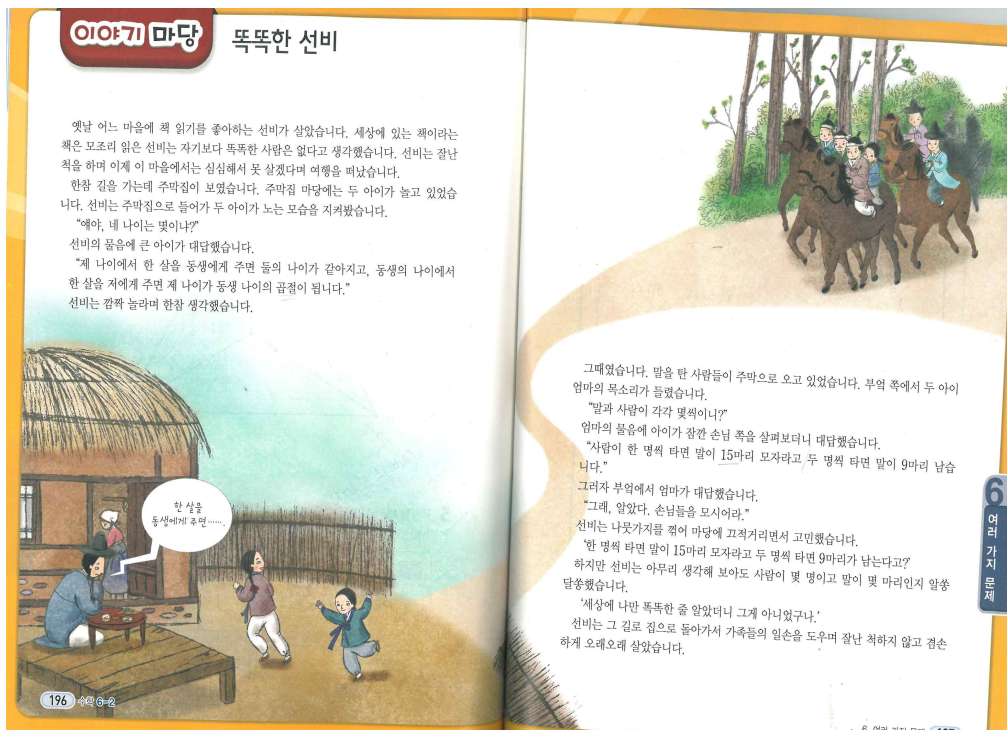
1. 쌓기나무 27

[그림 III-13] 비관적 사고 장면

6) 재귀적 사고 장면

[그림 II-6]에서 CT와 수학적 사고의 관계를 살펴볼 때 ‘재귀’는 CT와 수학적 사고의 교집합에 해당되지 않고 CT의 영역으로 분류한 바 있다. 이는 재귀가 원래의 자리로 되돌아가거나 되돌아온다는 뜻으로 재귀적 사고는 단순히 같은 행위를 반복적으로 하는 행동이 아니라 정해진 절차나 알고리즘을 시행하면서 필요한 부분으로 선택적으로 다시 돌아가 순서의 흐름에 맞게 되돌아가거나 되돌아오는 사고이다. 초등 수학교과서에서는 안내된 절차에 따른 단순 반복 즉, 예를 들면 곱셈구구를 학습할 때 동수누가에 따라 수가 커지는 것을 표로 작성하거나 규칙성 찾기 문제 영화관의 의자의 수와 팔걸이 수의 관계를 표로

작성하는 활동들은 재귀적 사고로 볼 수 없었다. 그리하여 CT와 수학적 사고의 관계에서는 ‘재귀’는 주로 CT의 영역에서 찾아볼 수 있어 수학적 사고와 연관성을 두지 않았다. 초등 수학 교과서에서는 학생들이 문제를 좀 더 쉽게 해결할 수 있게 하기 위해 재귀적 사고가 아닌 단순 알고리즘과 절차를 따라가도록 제시하여 쉽게 찾아보기 힘든 사고이다.

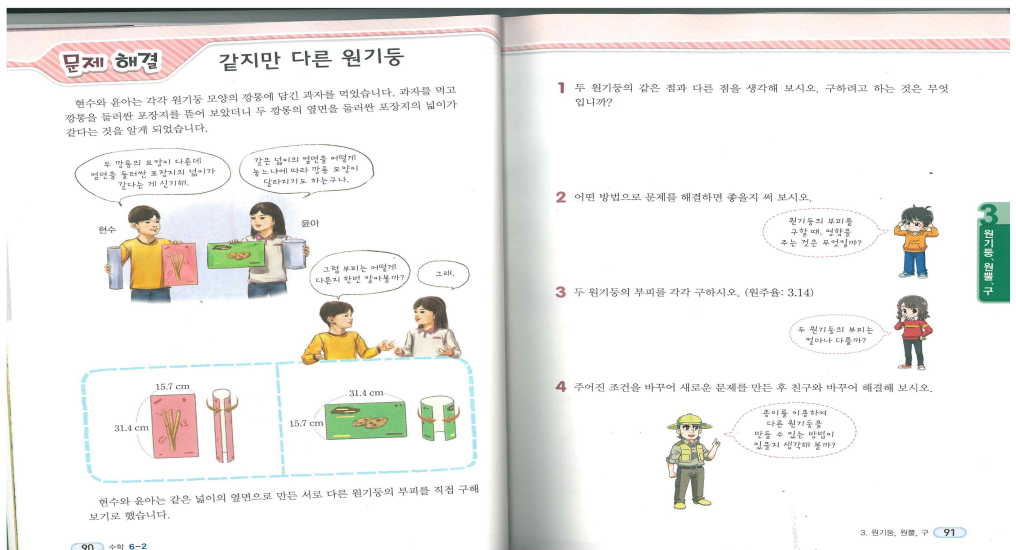
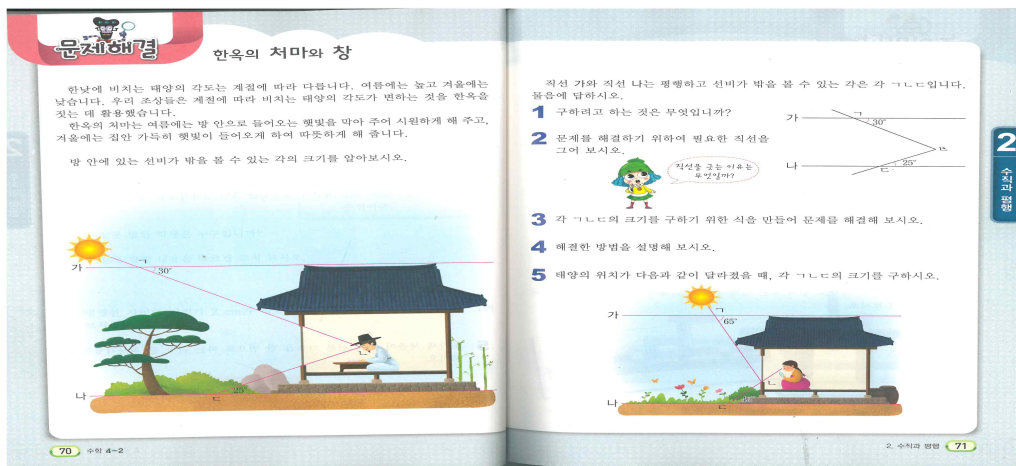


[그림 III-14] 재귀적 사고 장면

[그림 III-14]는 이야기 마당에 제시된 이야기로 학생들이 문제를 해결하지 않고 간단히 읽고 넘어갈 수 있는 내용이다. 형과 동생의 나이를 알기 위해 Polya의 문제해결 전략 중 ‘예상하고 확인하기’ 방법을 사용할 수 있다. 이때 예상의 기준이 되는 형(동생)의 나이를 토대로 동생(형)의 나이를 가늠하고, 형의 나이에서 한 살을 동생에게 주었을 때 둘의 나이가 같은지, 동생의 나이에서 한 살을 형에게 주었을 때 형의 나이가 동생의 나이의 곱절이 되는지 확인하면서 ‘예상하고 확인하기’를 부분적 또는 전체적으로 반복하므로 재귀적 사고에 부합된다. 여기서 반복은 동수누가의 개념으로 곱셈을 학습할 때 반복적으로 묶어세기

를 한다거나 교과서에 주어진 대로 표에 수를 적어 넣는 반복과는 차원이 다른
을 인지하여야 한다.

7) 문제 해결 전략의 하위 사고(동시적, 선행적, 전략적, 절차적, 재귀적) 장면
이은경(2009)이 제시한 CT의 구성 요소 중 문제 해결 전략에 들어있는 동시
적 사고, 선행적 사고, 전략적 사고, 절차적 사고, 재귀적 사고는 수학적 문제
해결의 사고 과정과 직접적으로 연관되어 있다.



[그림 III-15] 문제 해결 전략의 하위 사고 장면
(동시적, 선행적, 전략적, 절차적, 재귀적)

Polya는 문제 해결 과정을 4단계는 문제 이해, 계획 수립, 계획 실행, 반성 단계로 구성되어 있다. 문제의 이해 단계(1단계)에서 비판적 사고가 활성화되면서 동시에 여러 가지 조건들과 자료를 확인해야 하기 때문에 동시적 사고와 부합된다고 볼 수 있다. 해결 계획의 수립 단계(2단계)에서는 문제를 해결하기 위해 이전에 유사한 문제를 해결한 적이 있는지, 문제 해결을 위해 유용한 정리가 있는지 생각하므로 선행적 사고와 부합되며, 선행적 사고를 통해 떠올린 문제 해결 계획은 전략적 사고도 확인할 수 있다. 계획의 실행 단계(3단계)에서는 풀이 계획을 실행하고 매 단계를 점검하는 부분이므로 절차적 사고와 부합되며 마지막 반성 단계(4단계)를 통해 결과를 점검하고, 결과나 방법을 다른 문제에 활용하면서 4단계를 전체적 혹은 부분적으로 되돌아가 반복함으로써 재귀적 사고와 연결 가능하다.


나. '수와 연산', '도형' 영역에서의 Python 프로그래밍 코드

1) 수 크기 비교

초등 수학 교과서에서는 매 학년마다 자릿값의 개념을 확장하며 수를 도입한다. 1에서부터 10까지의 수, 11에서 100까지의 수, 세 자리 수, 네 자리 수 등을 학습할 때, 수를 읽고 쓴 후 수 크기 비교하는 학습이 필수적으로 포함되어 있다.

어느 수가 더 클까요
『수학 익힘』 16~17쪽

검은 콩은 2347개, 노란 콩은 3125개 있습니다. 콩의 수를 비교해 봅시다.



- 어느 콩이 더 많을까요?
- 어떻게 비교했는지 말해 보세요.

2347과 3125를 수 모형으로 나타내어 봅시다. 준비물 4

	천 모형	백 모형	십 모형	일 모형
2347				
3125				

- 2347과 3125 중에서 어느 수가 더 클까요?
- 왜 그렇게 생각했는지 말해 보세요.

22 수학 2-2

어느 수가 더 큰지 비교해 봅시다.

	천의 자리	백의 자리	십의 자리	일의 자리
7312	7	3	1	2
7298	7			8

- 두 수의 크기를 비교하여 $>$ 안에 $>$ 또는 $<$ 를 알맞게 써넣으세요.

천의 자리 수가 같아. $7312 < 7298$ 백의 자리 수는 다른데.....

- 어떻게 비교했는지 말해 보세요.

알맞은 수를 써넣어 봅시다.

	천의 자리	백의 자리	십의 자리	일의 자리
6100	6	1	0	0
5999				
6074				

- 가장 큰 수는 입니다.
- 가장 작은 수는 입니다.

1. 네 자리 수 23



어느 수가 더 클까요

수학 익힘, 16-17쪽

연수네 모듬은 줄넘기를 268번, 준기네 모듬은 312번 했습니다. 어느 모듬이 더 많이 했는지 알아봅시다.



어느 모듬이 줄넘기를 더 많이 했는지 말해 보세요.

268과 312 중 어느 수가 더 큰지 비교해 봅시다.

268과 312를 수 모형으로 나타내어 보세요. **준비물!**

	백 모형	십 모형	일 모형
268			
312			

두 수를 어떻게 비교했는지 말해 보세요.

347과 356 중 어느 수가 더 큰지 비교해 봅시다.

	백의 자리	십의 자리	일의 자리
347	3	4	7
356			

두 수의 크기를 비교하여 안에 > 또는 <를 알맞게 써넣으세요.



두 수를 어떻게 비교했는지 말해 보세요.

안에 알맞은 수를 써넣어 봅시다.

	백의 자리	십의 자리	일의 자리
610	6	1	0
594			
612			

가장 큰 수는 입니다.

가장 작은 수는 입니다.



수의 크기를 비교해 볼까요

수학 익힘, 16-17쪽

2016년 세계 여러 나라의 이동 전화 가입자 수를 살펴봅시다. 일본과 터키의 이동 전화 가입자 수를 비교해 봅시다.

일본과 터키 중 어느 나라의 이동 전화 가입자 수가 더 많을까요?

일본	1억 6426만 명 0
터키	7506만 명 0

일본과 터키의 이동 전화 가입자 수를 나타내어 보세요.

	억	천만	백만	십만	만	천	백	십	일
1억 6426만		6		2		0	0	0	0
7506만			5		6	0	0	0	0

어느 나라의 이동 전화 가입자 수가 더 많을까요?

어떻게 비교했는지 말해 보세요.

독일과 이집트의 이동 전화 가입자 수를 비교해 봅시다.

(출처: 국제기구 통계 이동 전화 가입자 수, 통계청, 2016.)

세계 여러 나라의 이동 전화 가입자 수

독일	9443만 명 0
이집트	9779만 명 0

독일과 이집트 중 어느 나라의 이동 전화 가입자 수가 더 적을까요?

독일과 이집트의 이동 전화 가입자 수를 나타내어 보세요.

	천만	백만	십만	만	천	백	십	일
9443만				3	0	0	0	0
9779만		7	7		0	0	0	0

어느 나라의 이동 전화 가입자 수가 더 적을까요?

어떻게 비교했는지 말해 보세요.



두 수의 크기를 비교해 봅시다.

- 두 수를 안에 써넣으세요.

4	0	0	0	0	0
2	0	0	0	0	0
6	0	0	0	0	0
<input type="text"/>					

4	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
<input type="text"/>					

- 위에 써넣은 두 수를 수직선에 나타내어 보세요.



- 두 수의 크기를 비교할 수 있는 방법을 말해 보세요.

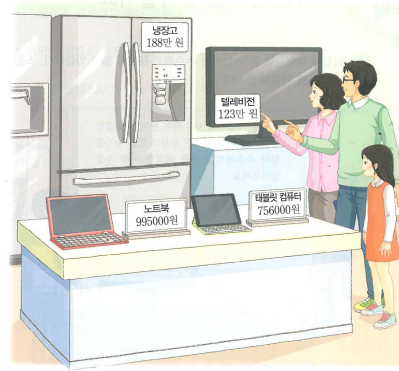
두 수의 크기를 비교하여 안에 >, =, <를 알맞게 써넣어 봅시다.

5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
<input type="text"/>				

6	0	0	0	0
5	0	0	0	0
7	0	0	0	0
<input type="text"/>				

1235460000 896540000

전자 제품의 가격을 비교해 봅시다.



1 큰 수

- 가격이 낮은 전자 제품부터 순서대로 써 보세요.

- 어떻게 비교했는지 말해 보세요.

[그림 III-16] 2학년, 4학년 교과서에 나오는 수 크기 비교 문제

수의 크기를 입력하고 단순하게 비교하는 것을 Python으로 코딩하면 [그림 III-17]과 같다. [그림 III-17]과 같은 코딩을 통해서 초등학교에서 수 크기 비교를 할 때 사용하는 알고리즘인 '자릿값이 큰 수부터 비교하여 수 크기를 확인한다.'를 알 수 없다. 학생들이 수 크기 비교 절차를 알 수 있도록 코딩을 하면 수 크기 비교 알고리즘은 [그림 III-18]과 같이 나올 수 있다. 프로그래밍 언어를 익히고 코딩하는 것이 주목적이 아니기 때문에 [그림 III-18]을 코딩하거나 코딩되어있는 것을 보고 이해하는 과정을 통해 학생들은 절차적 사고와 알고리즘을 이해할 수 있을 것이다.

```

x = input("크기를 비교할 첫번째 숫자를 입력하십시오")
a = int(x)

x = input("크기를 비교할 두번째 숫자를 입력하십시오")
b = int(x)

if a == b:
    print ("%d = %d"%(a, b))
elif a > b:
    print ("%d > %d"%(a, b))
else:
    print ("%d < %d"%(a, b))

```

```

===== RESTART: F:\ct#\파이썬\크기 비교_1.py =====
>>>
크기를 비교할 첫번째 숫자를 입력하십시오72
크기를 비교할 두번째 숫자를 입력하십시오72
72 = 72
>>>
===== RESTART: F:\ct#\파이썬\크기 비교_1.py =====
>>>
크기를 비교할 첫번째 숫자를 입력하십시오72
크기를 비교할 두번째 숫자를 입력하십시오75
72 < 75
>>>
===== RESTART: F:\ct#\파이썬\크기 비교_1.py =====
>>>
크기를 비교할 첫번째 숫자를 입력하십시오75
크기를 비교할 두번째 숫자를 입력하십시오72
75 > 72
>>>

```

[그림 III-17] 단순한 수 크기 비교 Python 코드와 그 결과

```

def compare(n1, n2):
    List1 = str(n1)
    List2 = str(n2)

    if len(List1) < len(List2):
        print('%d는 %d보다 크다.'%(n2, n1))
    elif len(List1) > len(List2):
        print('%d는 %d보다 크다.'%(n1, n2))
    else:
        if n1 == n2:
            print('%d과 %d는 같다.'%(n1, n2)) # print('{0}과 {1}는 같다.'.format(n1, n2))
        else:
            for i in range(0, len(List1)):
                if List1[i] < List2[i]:
                    print('%d는 %d보다 크다.'%(n2, n1))
                    break
                elif List1[i] > List2[i]:
                    print('%d는 %d보다 크다.'%(n1, n2))
                    break

```

[그림 III-18] 자릿값 비교를 적용한 수 크기 비교와 Python 코드

2) 짝수, 홀수

초등 수학 교과서에서 짝수는 둘씩 짝 지어지는 수로 학습한다. 나눗셈을 배운 학생들은 2로 나누어 나머지가 없는 수가 짝수이지만 아직 나눗셈을 배우지 않은 1학년 학생들에게는 새로운 수의 이름으로 짝수를 배우므로 짝수를 Python으로 코딩하면 [그림 III-19]와 같다.

```

print("나에게 숫자를 주세요.")
num = int(input())

minus = 0
while minus < num:
    minus = minus + 2

if num - minus == 0:
    print("%d은(는) 짝수이다."%num)

else:
    print("%d은(는) 홀수이다."%num)

```

[그림 III-19] 짝수와 Python 코드

[그림 III-19]는 주어진 수를 2씩 빼면서 값이 완전히 0이 될 경우는 짝수로, 아닌 경우는 홀수로 인식하는 코드이다. 이는 [그림 III-4]에 나와 있는 초등학교 1학년 교과서에 나와 있는 짝수 개념과 상통하다.

[그림 III-20]의 왼쪽 Python 코드는 짝수가 2로 나누었을 때 나머지가 없는 수임을 알고 코드화 시킨 것이고, 오른쪽 Python 코드는 짝수의 일의 자리 수가 2, 4, 6, 8로 반복되는 규칙을 찾고 일의 자리 수만 확인하여 짝수인지 홀수인지 확인하는 코드이다.

<pre> print("나에게 숫자를 주세요.") num = int(input()) if num % 2 == 0: print("%d은(는) 짝수이다."%num) else: print("%d은(는) 홀수이다."%num) </pre>	<pre> print("나에게 숫자를 주세요.") num = int(input()) even = ['0', '2', '4', '6', '8'] if str(num)[-1] in even: print("%d은(는) 짝수이다."%num) else: print("%d은(는) 홀수이다."%num) </pre>
--	---

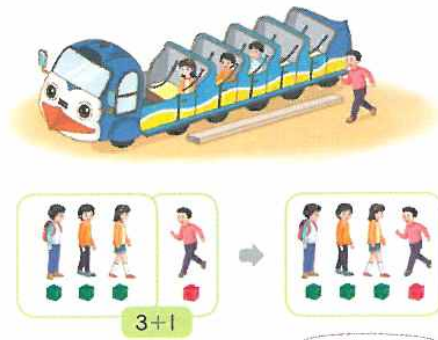
[그림 III-20] 다양한 짝수 Python 코드

3) 덧셈

덧셈 상황은 크게 첨가, 합병 상황으로 구분할 수 있다. 첨가 상황은 처음 있던 양이 증가하도록 보태는 것과 같은 변화를 일으키는 행위나 시간에 따른 변화에 관련되는 상황이며, 합병 상황은 변화를 일으키는 행위나 시간에 따른 변화와는 관련이 없고 전체 집합과 그 부분 집합의 관계에 관련되는 상황이다.(교

사용 지도서) 초등 수학 교과서에는 첨가 상황과 합병 상황이 제시되는데 그에 따라 Python 코드를 짜면 다음과 같다.

몇 명인지 알아봅시다.



```
def addi(a, b):
    list = []
    for i in range(0,b):
        list.append('o')

    return a + len(list)
```

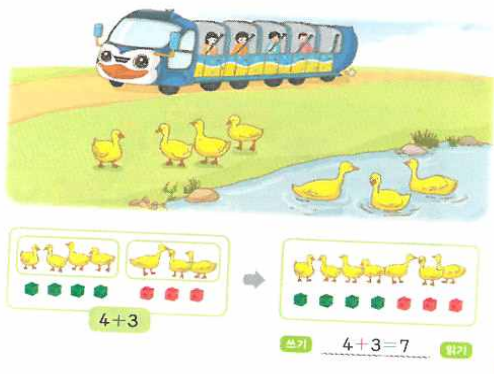
```
>>> addi(3, 1)
4
```

[그림 III-21] 덧셈 첨가유형과 Python 코드

[그림 III-21]의 왼쪽에 나와 있는 교과서 장면은 버스에 타고 있던 학생 3명과 새로 타는 학생 1명을 더하는 상황으로 덧셈 첨가유형이며 기존에 입력된 a에 0부터 b-1까지 추가하는 Python 코드를 만들 수 있다.

[그림 III-22]는 덧셈의 합병 유형으로 잔디 위에 있는 오리들과 물 속에 들어가는 오리들을 합하는 장면이다. a와 b에 입력한 수만큼 ○로 인식하여 총 합을 구하는 Python 코드를 만들 수 있다.

오리가 몇 마리인지 알아봅시다.



```
def add(a, b):
    list1 = []
    list2 = []
    for i in range(0,a):
        list1.append('o')
    for j in range(0,b):
        list2.append('o')

    print(list1 + list2)
    return len(list1 + list2)
```

```
>>> add(4, 3)
['o', 'o', 'o', 'o', 'o', 'o', 'o']
7
```

[그림 III-22] 덧셈 합병유형과 Python 코드

4) 뺄셈

뺄셈 상황은 크게 제거(구잔), 비교 상황으로 구분할 수 있다. 제거(구잔) 상

황은 처음 있던 양이 감소하도록 덜어 내는 것과 같은 변화를 일으키는 행위나 시간에 따른 변화에 관련되는 상황이며, 비교 상황은 서로소인 두 집합의 크기를 비교하는 것과 관련되는 상황이다.(교사용 지도서) 초등 수학 교과서에는 제거(구산)와 비교 상황이 제시되는데 그에 따라 Python 코드를 짜면 다음과 같다.

남은 풍선은 몇 개인지 알아봅시다.



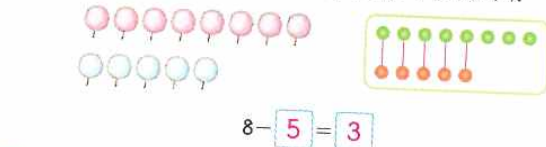
```
def sub(a,b):
    list = []
    for i in range(0,a):
        list.append('o')
    del list[0:b]
    print(list)
    return len(list)
```

```
>>> sub(6,1)
['o', 'o', 'o', 'o', 'o']
5
```

[그림 III-23] 뺄셈 구산유형과 Python 코드

[그림 III-23]의 왼쪽에 나와 있는 교과서 장면은 풍선 6개 중 1개의 풍선이 날아가는 상황으로 입력받은 a의 ○수에서 b만큼 제거해나가서 남은 ○의 수를 확인하는 Python 코드를 만들 수 있다.

분홍 솜사탕은 파란 솜사탕보다 몇 개 더 많은지 알아봅시다.



```
def subt(a,b):
    list1 = []
    list2 = []
    dif = []
    for i in range(1,a+1):
        list1.append(i)
    for j in range(1,b+1):
        list2.append(j)
    for i in list1:
        if i not in list2:
            dif.append(i)
    print(dif)
    return len(dif)
```

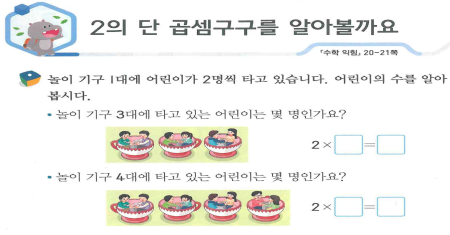
```
>>> subt(8,5)
[6, 7, 8]
3
```

[그림 III-24] 뺄셈 비교유형과 Python 코드

[그림 III-24]는 뽕셈의 비교 유형으로 분홍 솜사탕과 파란 솜사탕을 1대 1로 비교하여 몇 개 더 많은지 알아보는 상황이다. 이를 Python 코드로 만들게 되면 입력받은 a와 b를 list1과 list2로 인식하여 두 집합의 크기를 비교하여 list1과 list2이 원소의 차이를 알아내고 있다.

5) 곱셈

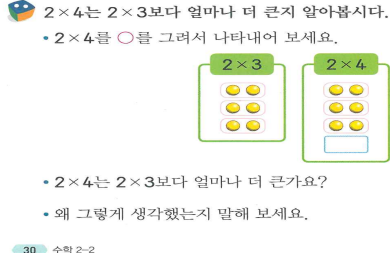
초등 수학 교과서에서 곱셈이 처음 도입되는 학년은 2학년이다. 2학년 1학기에 동수누가 개념으로 곱셈을 배우기 시작하고, 곱셈 구구단은 2학년 2학기부터 활용된다. 곱셈의 의미를 이미 알고 있는 학생들이 단순한 곱셈 기호만은 활용하여 만들 수 있는 곱셈 Python 코드는 [그림 III-25]와 같다.



2의 단 곱셈구구를 알아볼까요

높이 기구 1대에 어린이가 2명씩 타고 있습니다. 어린이의 수를 알아봅시다.

- 높이 기구 3대에 타고 있는 어린이는 몇 명인가요?
 $2 \times \square = \square$
- 높이 기구 4대에 타고 있는 어린이는 몇 명인가요?
 $2 \times \square = \square$



2×4 는 2×3 보다 얼마나 더 큰지 알아보시다.

- 2×4 를 ○를 그려서 나타내어 보세요.


2×3 2×4

- 2×4 는 2×3 보다 얼마나 더 큰가요?
- 왜 그렇게 생각했는지 말해 보세요.

• 2의 단 곱셈구구를 완성해 보세요.

$2 \times 1 = 2$
$2 \times 2 = 4$
$2 \times \square = 6$
$2 \times 4 = \square$
$2 \times 5 = \square$
$2 \times \square = 12$
$2 \times 7 = \square$
$2 \times \square = 16$
$2 \times 9 = \square$

2의 단 곱셈구구에서 곱하는 수가 1씩 커져요. 그러면 곱은 얼마씩 커지나요?



```

x = input("나에게 곱셈할 숫자를 주세요.")
a = int(x)

x = input("곱셈할 나머지 숫자를 주세요.")
b = int(x)

print("%d 곱하기 %d 은(는) %d 이다"%(a, b, a))

```

```

---
나에게 곱셈할 숫자를 주세요.5
곱셈할 나머지 숫자를 주세요.3
5 곱하기 3 은(는) 15 이다
>>>

```

[그림 III-25] 곱셈 Python 코드와 결과

동수누가의 개념을 적용하여 곱셈 코드를 만들게 되면 Python 코드는 [그림 III-26]과 같이 달라지게 된다.

```
def multi(a, b):  
    list = []  
    for i in range(0, b):  
        list.append(a)  
    return sum(list)
```

[그림 III-26] 동수누가 곱셈 Python 코드

6) 나눗셈 알고리즘

초등 수학교과서에서는 나눗셈 알고리즘을 두 가지 상황으로 도입하고 있다. 먼저, [그림 III-27]은 ‘포함제’의 관점, 즉 동수누가의 상황으로 활동을 한 후 ‘몫’과 ‘나머지’ 개념을 설명하고 있다. 이를 컴퓨터 프로그래밍 언어인 Python으로 코딩하면 [그림 III-27]의 오른쪽과 같다. ‘ $19 \div 5 = 3 \cdots 4$ ’를 직관적으로 19에서 단번에 $5 \times 3 = 15$ 를 빼면 4가 남으므로 몫이 3, 나머지가 4인 것으로 계산하지만, 이를 몫과 나머지를 구하는 프로그래밍을 작성하려면, 19에서 나머지가 5보다 작게 나올 때까지 5를 계속 빼면서 뺀 횟수와 그때 나머지를 기록하는 과정으로 알고리즘을 분해하고 이해하는 것이 필요하다. 알고리즘은 문제해결 방식을 단계적으로 분석해서 나오기 때문에 직관적으로 알고 있던 것도 알고리즘으로 코딩화 시키고 생각하는 시간은 학생들이 당연하게 알고 있던 개념을 분석하고 각 요소마다 관계들을 생각할 수 있어 논리적 사고에 도움이 될 것이다.

한편, 등분제 상황의 나눗셈 알고리즘은 [그림 III-28]과 같다. 교과서에서 두 가지의 나눗셈 알고리즘을 사용하고 있지만 실질적으로 주어진 상황만 보고 어느 것이 포함제 나눗셈인지 등분제 나눗셈인지 알지 못하고 단순 공식을 활용하여 도구적 이해로만 그치는 경우가 많다. 도구적 이해에 이미 숙달된 학생들에게 두 가지 나눗셈 상황에 따라 달라지는 코딩을 보면서 문장제 곱셈 문제를 분석해 볼 수 있는 기회가 될 것이다. 이를 통해 분석적 사고를 기르고 학생들이 가지고 있는 수학적 개념을 도구적으로만 활용하는데 그치지 않고 관계적 이해로까지 확장시킬 수 있을 것이다.

약수의 수학적 개념을 그대로 Python으로 코딩하면 [그림 III-29]와 같다.

수의 이름

어떤 수를 나누어떨어지게 하는 수를 그 수의 **약수**라고 합니다.
8을 1, 2, 4, 8로 나누면 나누어떨어집니다.
1, 2, 4, 8은 8의 약수입니다.

```
def divisor(a):  
    D = []  
  
    for x in range(1, a+1):  
        if a % x == 0:  
            D.append(x)  
    return D  
  
print(divisor(8))
```

[그림 III-29] 약수와 Python 코드

8) 배수

초등 수학 교과서에서는 배수의 개념을 ‘배(비)의 관점’으로 활동을 한 후, [그림 III-30]과 같이 배수를 도입하고 있다. 배수의 개념을 Python으로 코딩하면 [그림 III-30]과 같다.

수의 이름

어떤 수를 1배, 2배, 3배……한 수를 그 수의 **배수**라고 합니다.
3, 6, 9……는 3의 배수입니다.

```
def multiple(a, n):  
    M = []  
  
    for x in range(a, a + (n+1), a):  
        M.append(x)  
    return M  
  
print(multiple(8, 4))
```

[그림 III-30] 배수와 Python 코드

9) 최대공약수

초등 수학 교과서에서 어떤 두 자연수를 각각 나누어떨어지게 하는 약수 중에서 공통인 약수로서 공약수를 이해하고 구하며, 공약수 중에서 가장 큰 수를 최대공약수를 구하게 한다. 즉, 초등에서 최대공약수를 구하는 방법은 [그림 III-31]에서와 같이 ‘각 수의 약수를 구하고, 그 후 두 수의 공약수를 찾고, 끝으로 공약수 중에서 가장 큰 것’을 구하는 방법을 사용하고 있다. 이것을 Python으로 코딩하면 [그림 III-31]과 같다.

수의 이름

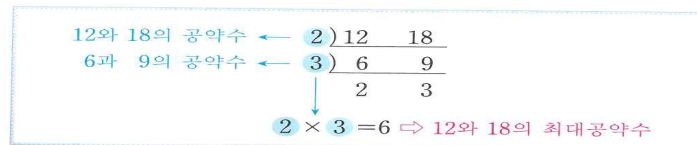
두 수의 공통인 약수를 두 수의 **공약수**라고 합니다.
 두 수의 공약수 중에서 가장 큰 수를 두 수의 **최대공약수**라고 합니다.
 1, 2, 3, 6은 12와 18의 공약수입니다.
 6은 12와 18의 최대공약수입니다.

활동 3

12와 18을 공약수로 나누어 보면서 최대공약수를 구하시오.

- 12와 18의 공약수를 구하시오.

- 12와 18의 공약수를 이용하여 12와 18의 최대공약수를 구할 수 있습니다.



```
def gcd(a, b):
    if a == 0:
        return b
    elif b == 0:
        return a
    else:
        A = []
        B = []

        for i in range(1, a+1):
            if a % i == 0:
                A.append(i)
        for j in range(1, b+1):
            if b % j == 0:
                B.append(j)

        return max(set(A) & set(B))

print(gcd(15, 5))
print(gcd(100, 50))
```

[그림 III-31] 최대공약수와 Python 코드

10) 최소공배수

초등 수학 교과서에서 어떤 두 자연수의 몇 배에 해당하는 배수 중에서 공통인 배수로서 공배수를 이해하고 구하며, 공배수 중에서 가장 작은 수를 최소공배수라 하고 최소공배수를 구하게 한다. 초등에서 최소공배수를 구하는 방법은 [그림 III-32]와 같이 '각 수의 배수를 구하고, 그 후 두 수의 공배수를 찾고, 끝으로 공배수 중 가장 작은 것'을 구하는 방법을 사용하고 있다. 이것을 Python으로 코딩하면[그림 III-32]와 같다.

수의 이름

두 수의 공통인 배수를 두 수의 **공배수**라고 합니다.
 두 수의 공배수 중에서 가장 작은 수를 두 수의 **최소공배수**라고 합니다.
 12, 24, 36 ……은 4와 6의 공배수입니다.
 12는 4와 6의 최소공배수입니다.

활동 3

12와 30을 공약수로 나누어 보면서 최소공배수를 구하시오.

- 12와 30의 공약수를 구하시오.
- 12와 30의 공약수를 이용하여 12와 30의 최소공배수를 구할 수 있습니다.

12와 30의 공약수 ←	2)	12	30	
6과 15의 공약수 ←	3)	6	15	
			2	5	
			↓	↓	
			2×3	2×5	
			$= 60 \Rightarrow$		12와 30의 최소공배수


```

def lcm(a, b):
    if (a == 0) or (b == 0):
        return 0

    else:
        A = []
        B = []

        for i in range(a, a + b + 1, a):
            A.append(i)
        for j in range(b, a + b + 1, b):
            B.append(j)

        return min(set(A) & set(B))

print(lcm(15,5))
print(lcm(100,50))
print(lcm(3,7))

```

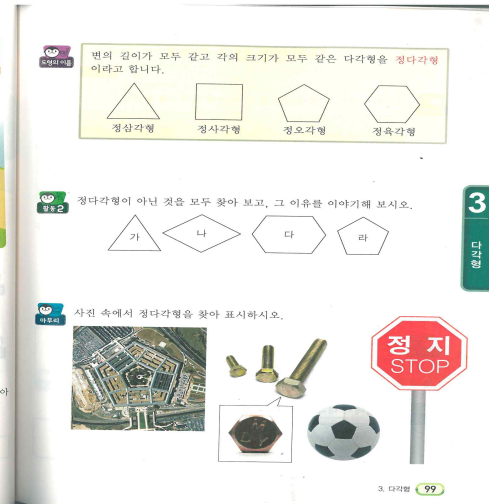
[그림 III-32] 최소공배수와 Python 코드

11) 정다각형

정다각형은 변의 길이가 모두 같고 각의 크기가 모두 같은 다각형을 정다각형이라고 한다. 정다각형은 변의 수에 따라 변이 3개이면 정삼각형, 변이 4개이면 정사각형, 변이 5개이면 정오각형, 변이 6개이면 정육각형으로 부른다. 초등수학 교과서에 나오는 정다각형은 정삼각형, 정사각형, 정오각형, 정육각형이며 이것을 Python으로 코딩하면 [그림 III-33]과 같다.

[그림 III-33]의 왼쪽 Python 코드는 삼각형이 세 변으로 구성되어있는 것을 활용하여 3회 반복기능을 사용한다. 이러한 Python 코드를 생각해내기 위해서는 학생들이 이미 삼각형의 한 내각의 크기와 반복횟수를 알고 있어야 하므로 처음부터 학생들 스스로 생각하게 하기 보다는 삼각형 그리기를 예시로 주고 사각형 그리기를 혼자 생각하여 Python 코드 짜기를 실습시키면 학생들의 CT 사고를 향상시킬 수 있을 것이다.

[그림 III-33]의 오른쪽 Python 코드는 n각형의 한 내각의 크기를 $360/n$ 으로 그리고 n번 반복하게 하여 여러 가지 다각형을 그릴 수 있는 Python 코드이다.



```
import turtle as t
#삼각형 그리기
for x in range(3):
    t.forward(100)
    t.left(120)

#사각형 그리기
for x in range(4):
    t.forward(100)
    t.left(90)

#원 그리기
t.circle(50)
```

```
import turtle as t

def polygon(n):
    for x in range(n):
        t.forward(50)
        t.left(360/n)

def polygon2(n, a):
    for x in range(n):
        t.forward(a)
        t.left(360/n)

polygon(3)
polygon(5)

t.up()
t.forward(100)
t.down()

polygon2(3, 75)
polygon2(5, 100)
```

[그림 III-33] 정다각형과 Python 코드

12) 선분으로 여러 가지 모양 만들기

6학년 2학기 수학 교과서 마지막 단원의 제목은 ‘여러 가지 문제’이다. 6단원 여러 가지 문제에서 규칙에 따라 선분으로 여러 가지 모양을 만드는 차시가 있다. 이 차시를 학습할 때 Python 코드를 활용하여 그 결과물을 직접 눈으로 확인하고 간단히 숫자와 색상을 바꿔가며 다양한 모양을 만들어갈 수 있다.

규칙에 따라 선분으로 여러 가지 모양을 만들 수 있어요

특별하기 직선이 만들어 내는 아름다운 장면입니다. 규칙에 따라 직선을 많이 모으면 어떤 모양이 만들어질지 생각해 봅시다.

182 수학 6-2

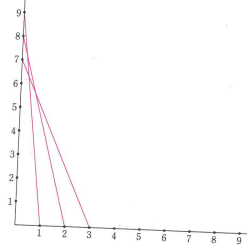
활동 1 규칙을 찾아 표를 완성하고 규칙에 따라 선분을 그으면 어떤 모양이 만들어질지 알아보시오.

- 규칙을 찾아 표를 완성하고 규칙을 써 보시오.

가로	1	2	3	4	5	6	7	8	9
세로	9	8	7						

규칙

- 규칙에 따라 두 수를 선분으로 이어 보시오.



- 완성된 모양을 보고 어떤 모양인지 이야기해 보시오.

6. 여러 가지 문제 183

6. 여러 가지 문제

활동 2 규칙을 정하고 규칙에 따라 선분을 그려 모양을 만들어 보시오.

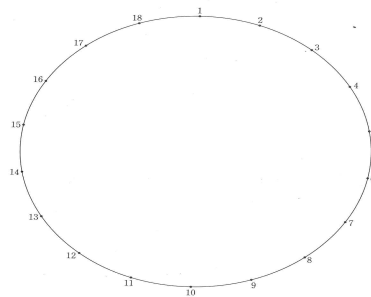
규칙

친구가 만든 모양과 비교해 보고 같은 점과 다른 점을 이야기해 보시오.

184 수학 6-2

활동 2 규칙을 정하고 규칙에 따라 선분을 그려 모양을 만들어 보시오.

규칙

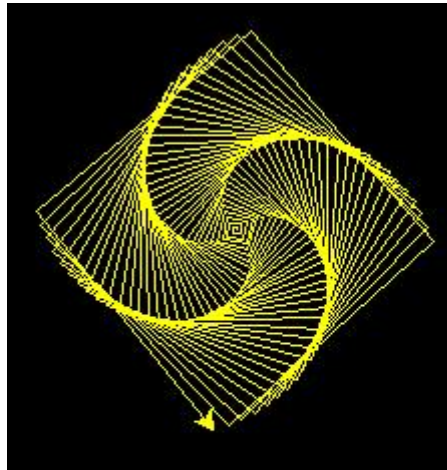


6. 여러 가지 문제

6. 여러 가지 문제 185

```
import turtle as t

t.bgcolor("black")
t.color("yellow")
t.speed(0)
for x in range(200):
    t.forward(x)
    t.left(79)
```



[그림 III-34] 선분으로 여러 가지 모양 만들기와 Python 코드

IV. 결론 및 제언

1. 요약 및 결론

급격히 발달하는 정보통신기기와 기술의 보편화로 오늘날 사회는 예측할 수 없게 다양한 변화를 겪고 있고 미래 사회를 예측하는 것은 더 어려운 일이 되었다. 이러한 미래 사회를 이끌어갈 창의적 미래 인재를 기르기 위해 교육부에서는 스마트 교육을 도입하였다. 그러나 교육과정에 도입된 스마트 교육은 21세기 미래 인재를 양성한다는 본래 취지와는 다르게 스마트 기기 보급에만 치중되어 왔다. SW교육이 본래 취지를 살리기 위해서는 실생활에서 복잡하고 다양한 문제를 컴퓨터가 문제를 해결해 나가는 과정인 CT의 개념을 포함한 소프트웨어 교육이 필요하다. Jeannette Wing이 주장한 것처럼 모두가 기본적으로 갖춰야 하는 3R(읽기, 쓰기, 산수)에 CT가 동등한 자격으로 같이 거론되기 위해서는 CT능력이 어느 특정하고 한정된 영역에서만 길러지는 것이 아니라 삶 전반적인 장면에서 다루어져야 한다. 그러므로 CT교육 역시 독자적인 영역이 아니라 모든 학생들에게 자연스럽게 노출되는 여러 교과에서 CT의 요소들을 찾고 CT능력 함양 교육이 이루어져야 한다.

본 연구는 이러한 귀인에서 출발하여 CT와 수학적 사고에서 공통적으로 들어가는 사고와 하위기능을 선정하였다. CT와 관련된 수학적 사고에는 귀납적 사고, 연역적 사고, 유추적 사고, 추상화 사고, 기호화 사고, 논리적 사고, 비판적 사고 총 7가지가 있고, CT의 구성요소 중 논리적 사고, 분석적 사고, 추상적 사고, 문제 해결 전략 하위 사고와 더 나아가 비판적 사고, 재귀적 사고를 포함한 총 6가지 CT 구성요소를 초등 수학 교과서 내에서 찾아보고 분석하였다. 이는 초등학교 수학 교육이 나아갈 방향으로 ‘수학적 연결성 추구’가 있으며 이를 CT와 연결하여 CT를 독립된 교과와 학습 내용으로 다루기보다 수학적 지식과 수학 교과 내용과 연결하여 자연스럽게 접하고 CT 능력을 향상할 수 있을거라 생각하였기 때문이다.

본 연구는 초등 수학 교과서에서 CT와 관련지을 수 있는 장면들을 찾아보고 분석하여 Python 코드로 만들어봄으로써 일반적이고 보편적인 SW교육의 일환으로 CT교육에 대한 접근 가능성을 알아보는 데 그 목적이 있다.

이를 위하여 우선 2009, 2015 개정 교육과정과 교과서 분석을 통해 수학 교과서에 반영된 CT장면을 찾아 수학적 사고와 CT의 공통 영역에서 어느 부분에

해당되는지 분석하였다. 또한, 수학 교과서에 제시되어있는 문제를 컴퓨터 프로그래밍 언어인 Python으로 코딩하여 수학 교과 지도 시 CT를 염두하여 지도하였을 때 학습자들이 얻을 수 있는 교육적 효과를 탐색하였다. 수학 교과서에 반영된 CT장면과 이를 분석한 결과는 다음과 같다.

첫째, CT교육은 독자적으로 학습 자료를 따로 구상하여 만들 필요 없이 수학 교과서 내에서 찾아 볼 수 있다. 초등 수학 교과서 상의 CT 장면에는 추상적 사고 장면, 알고리즘적 사고 장면, 논리적 사고 장면, 분석적 사고 장면, 비판적 사고 장면, 재귀적 사고 장면, 문제 해결 전략의 하위 사고(동시적, 선행적, 전략적, 절차적, 재귀적) 장면이 있다. 추상적 사고 장면은 숫자나 도형과 같이 개념을 학습할 때 찾아 볼 수 있으며, 규칙이 있는 두 수 사이의 대응 관계를 확인하고 이를 식으로 나타내는 장면에서도 찾아볼 수 있다. 또한, CT에서의 ‘추상화’란 개념은 한 문제를 일반화하고 다른 유사한 문제들로 해결책을 옮기는 능력도 포함하기 때문에 하나의 사실을 알아내고 일반화를 이끌어 내는 장면에서도 추상적 사고를 발견할 수 있다. 알고리즘적 사고 장면에는 덧셈, 뺄셈, 곱셈, 나눗셈 등 계산과정을 형식화하는 부분과 문제를 해결하는 과정을 돕기 위해 나오는 발문들에서 발견할 수 있다. 논리적 사고 장면은 왜 그렇게 생각하는지 이유를 묻는 발문과 아동 상호 간의 논의나 토론을 제시하는 발문, 평행사변형, 마름모, 직사각형에서 변의 길이나 변의 위치, 각의 크기 등을 분석하여 도형의 성질을 발견하는 장면에서 찾아볼 수 있다. 분석적 사고와 비판적 사고는 문제를 해결하기 위하여 주어진 단위 요소로서의 자료나 정보간의 논리적 관계를 확인하고 주어진 자료에서 모순성과 불일치를 판단하는 것이기 때문에 문제 해결 전 과정에서 찾아볼 수 있다. 재귀적 사고 장면은 이야기 마당에 나오는 수학 이야기 문제를 해결하기 위해 해결 방법을 학생 스스로 생각해보고 문제 해결을 위한 절차나 알고리즘을 반복적으로 시행하면서 필요한 부분으로 다시 돌아가 순서의 흐름에 맞게 되돌아가거나 되돌아와야 하는 문제에서 찾을 수 있다.

둘째, 초등 수학 교과서에서 찾아볼 수 있는 CT 장면만 잘 다루어도 학생들이 얻을 수 있는 교육적 이점이 크다는 점이다. 학생들이 문제를 좀 더 쉽게 해결할 수 있게 단순 알고리즘 절차를 따라가도록 제시하는 초등 수학 교과서의 특성상 재귀적 사고 장면은 쉽게 찾아볼 수 없지만 문제 해결 방법 및 절차를 스스로 생각하고 그 과정에서 재귀적 사고를 적용한다면 CT 및 수학적 사고

함양 가능성은 커진다. 특히, 수학 교과서에서 가볍게 읽고 넘어가는 이야기 마당에 담겨있는 수학 문제가 있는 이야기를 읽고 안내된 절차 및 알고리즘이 아닌 학생 스스로 해결 방법을 생각하고 시도해보면서 수학적 사고를 기를 수 있고 그 과정에서 자연스럽게 CT능력을 발휘하고 함양할 수 있다.

셋째, 초등 수학 교과서에는 프로그래밍 코드를 작성할 수 있는 수많은 소재들이 존재한다. 자릿값이 큰 수부터 비교하여 두 수의 크기를 비교하는 문제, 둘로 묶을 수 있는 짝수와 둘로 묶을 수 없는 홀수 문제, 첨가 문제 유형과 합병 문제 유형이 있는 덧셈, 제거(구잔) 문제 유형과 비교 문제 유형이 있는 뺄셈 문제, 동수누감을 이해할 수 있는 곱셈 문제, 등분제와 포함제를 Python 코드로 비교할 수 있는 나눗셈 문제, 약수와 배수 문제, 최대공약수와 최소공배수 모두 수와 연산 영역에서 프로그래밍 코드를 작성해 볼 수 있는 소재들이다. 위의 수학적 개념들은 해당 학년 학생들이 프로그래밍 코드를 작성하기 보다는 이미 그 개념을 배운 학생들이 복습 및 더 나아가는 방향으로 분석적이고 논리적으로 생각하는 보충 문제로 제시할 수 있다. 이를 통해 학생들은 해당 수학적 개념의 바탕이 되는 기본 개념을 확인하고 적용할 수 있다. 또한, 도형 영역에서는 거북이를 움직여 간단한 정다각형을 그릴 수 있는 것부터 시작하여 선분으로 여러 가지 모양 만들기가 프로그래밍 코드 소재로 사용할 수 있다.

넷째, 초등 교과서에 나와 있는 문제 상황을 보고 프로그래밍 코드를 작성하면 수학적 사고와 CT 능력을 동시에 함양할 수 있다. 수학 문제를 단순히 맞추기 위해서가 아니라 자신의 생각을 정리하고 코드로 작성해 봄으로써 기존에 알고 있던 해결법뿐만 아니라 여러 가지 다양하고 확산된 사고를 함으로써 프로그래밍을 직접 구상하고 실행하고 그 결과를 즉각적으로 확인해 볼 수 있는 기회는 학생들의 사고력을 향상시킬 것이다. 수 크기 비교 알고리즘을 생각하더라도 프로그래밍 언어를 짜면서 자릿값이 큰 것부터 비교해야 한다는 점을 생각할 수 있는 기회를 얻을 수 있다. 학생들이 자기도 모르게 수 크기를 비교하지만 자신이 알고 있는 내용을 한번 더 정리하고 내면화시킬 수 있다. 짝수 코드를 작성하는 부분에서는 동수누감, 나눗셈을 하여 나머지가 없는 수, 일의 자리 수가 2, 4, 6, 8로 반복되는 규칙 등 하나의 개념을 접근할 때 그와 관련된 다양한 개념들을 생각하고 여러 가지 결과가 나올 수 있다. 덧셈과 뺄셈 Python 코드를 작성할 때 코드를 통해 첨가, 합병, 구잔, 비교 유형을 알 수 있으며, 곱셈의 경우 Python 코드를 해석하다보면 `append`(덧붙이다, 첨부하다)를

활용하여 똑같은 수를 계속 더하는 동수누가의 개념을 이해할 수 있다. 나눗셈을 Python으로 코딩하는 경우 나눗셈을 기계적으로 계산 할 수 있는 학생들에게 나눗셈의 기본바탕은 뺄셈임을 다시 한 번 상기시켜줄 수 있는 좋은 기회가 된다. 최대공약수와 최소공배수 코드를 작성할 때에는 수학적 개념에 가장 바탕이 되는 기저 개념을 확인하고 인지할 수 있다. 도형 영역에서는 자신의 생각과 아이디어를 마음껏 펼쳐볼 수 있는 기회를 얻을 수 있다. 특히, 선분으로 여러 가지 모양 만들기 부분에서 Python 프로그램을 활용한다면 자신이 작성한 코드가 바로 눈앞에서 멋진 결과물로 나오고 코드 안에서 숫자와 몇 개의 조건을 바꿔가면서 다양한 모양을 만들 수 있다.

또한, 자신이 스스로 코드를 작성함으로써 자아 효능감도 긍정적으로 변할 것이며, 자신이 생각해 낸 프로그래밍 결과가 옳은지 아닌지 판단은 컴퓨터 프로그램을 통해 즉각적으로 확인 가능하기 때문에 실패에 대한 두려움이 없어질 것이다.

2. 제언

본 연구의 제한점 및 후속 연구에 대한 제언은 다음과 같다.

첫째, 본 연구에서 분석한 CT 접목 가능한 교과서 장면을 직접 학생들에게 적용하고 실험해보는 연구가 필요하다. 본 연구에서는 교과서상에서 CT 장면을 찾고 분석하는 데 그쳐 학생들이 교과 장면을 접할 때 실질적으로 CT 능력이 향상되는지에 대한 현장 적용 연구는 이루어지지 않았다.

둘째, 학생들의 CT 능력 향상 정도를 측정하는 검사 도구에 대한 연구가 필요하다. 복잡하고 다양한 문제를 컴퓨터가 문제를 해결해 나가는 과정인 CT를 학생들이 구현해 내고 있는지 어떻게 알아볼 것이며 그 향상 정도를 어떤 도구로 어떻게 측정할 것인지에 대하여 고민하고 체계적인 검사 도구 개발이 이루어질 필요가 있다.

참 고 문 헌

- 교육과학기술부, 수학 1-1, (주)천재교육, 2017a
- 교육과학기술부, 수학 1-2, (주)천재교육, 2016a
- 교육과학기술부, 수학 2-1, (주)천재교육, 2017b
- 교육과학기술부, 수학 2-2, (주)천재교육, 2016b
- 교육과학기술부, 수학 3-1, (주)천재교육, 2017c
- 교육과학기술부, 수학 3-2, (주)천재교육, 2016c
- 교육과학기술부, 수학 4-1, (주)천재교육, 2017d
- 교육과학기술부, 수학 4-2, (주)천재교육, 2016d
- 교육과학기술부, 수학 5-1, (주)천재교육, 2017e
- 교육과학기술부, 수학 6-2, (주)천재교육, 2016e
- 이은경. (2009). **Computational Thinking** 능력 향상을 위한 로봇 프로그래밍 교수 학습 모형, 한국교원대학교 대학원 박사학위논문. 12
- 양단희. (2016). **SW 교육의 올바른 방향에 대한 고찰 - Computational Thinking vs. Coding -**, 인터넷정보학회지. 55-58
- 장경윤. (2017). **수학 교과에서 계산적 사고(Computational Thinking) 교육**, 대한수학교육학회지. 560-563
- 문교식. (2013). **Computational Thinking의 초등교육 활용 방향**, 한국콘텐츠학회논문지. 525
- 김지은. (2012). **수학적 사고력 신장을 위한 지도 방안**, 충남대학교 교육대학원 석사학위논문. 5-7
- 정승연. (2008). **수학적 사고력 신장을 위한 지도방안**, 한양대학교 교육대학원 석사학위논문. 4
- 이광호. (2008). **수학적 사고의 적용 현황에 관한 연구 : 중학교 2학년 학생을 중심으로**, 단국대학교 석사학위논문. 5-14
- 남충모 · 김종우. (2011). **Computational Thinking 기반 초등수학과 교수·학습활동 분석**, 교육과학연구. 48-49
- 김수환. (2015). **Computational Thinking 증진을 위한 학습자 중심의 교수학습 전략의 효과**, 정보교육학회논문지. 324
- 강옥기. (1990). **수학적 사고력 신장 프로그램 개발을 위한 방안탐색 : 국민학교 산수과를 중심으로**, 한국교육개발원.

- 강시중. (1971). **수학적 사고력을 신장시키는 학습지도 방안**, 전주교육대학.
- 김응태 · 박한식 · 우정호. (2007). **수학교육학개론**, 서울대학교출판부.
- 강완 · 백석윤. (2003). **초등수학교육론**, 동명사.
- 안상철. (2012). **수학적 사고능력과 논리적 사고능력과의 상관관계**, 중앙대학교 교육대학원 석사학위논문.
- 김태영. (2014). **CT융합영재교육과정 운영 안내서**, 한국교원대학교 영재교육원.
- 이승찬. (2017). **모두의 알고리즘 with 파이썬**, 길벗.
- Yadav, A. & Hong, H. & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms, TechTrends. 565-568
- David Weintrop & Elham Beheshti & Michael Horn & Kai Orton & Kemi Jona & Laura Trouille & Uri Wilensky. (2016). Defining Computational Thinking for Mathematics and Science Classrooms, J Sci Educ Technol. 129
- Matti Tedre & Peter J. Denning. (2016). The Long Quest for Computational Thinking, Proceedings of the 16th Koli Calling Conference on Computing Education Research. 120
- Jeannette M. Wing. (2006). Computational Thinking, Communications of the ACM. 33-35
- Jeannette M. Wing. (2010). Computational Thinking: What and Why?, Communications of the ACM. 1-5
- Shuchi Grover & Roy Pea, (2013). Computational Thinking in K-12: A Review of the State of the Field, Educational Researcher. 38-40
- Shuchi Grover & Roy Pea. (2013). Computational Thinking in K-12: A Review of the State of the Field, Educational Researcher. Report of a workshop on the scope and nature of Computational Thinking, National Research Council.
- Papert. (1980). **Mindstorms: Children, Computers, and Powerful Ideas**, Harvester Studies in Cognitive Science.

A B S T R A C T *

Analysis of Computational Thinking in Elementary Math Textbooks

Ko, Hye Young

Major in Elementary Mathematics Education
Graduate School of Education
Jeju National University

Supervised by Professor Choi, keunbae

The purpose of this study was to examine the potential of CT education in primary math textbooks for identifying and analyzing scenes that may be associated with computational thinking.

For this purpose, I searched the 2009 and 2015 revision curriculum and textbooks to find the CT scenes that were reflected in the math textbooks, and analyzed the areas that fall into the common areas of mathematical accidents and CT. In addition, the problems presented in the mathematics textbook were coded into Python, a computer programming language, to explore the educational effects that learners would gain if they mapped CT in the mathematics curriculum.

* A thesis submitted to the committee of Graduate School of Education, [Jeju National University](#) in partial fulfillment of the requirements for the degree of Master of Education conferred in February, 2011.

Within the elementary mathematics textbook, there was a abstract thinking scene, an allergistic accident scene, a logical accident scene, a critical accident scene, a reflexive scene, and a sub-incidence of problem-solving strategy. As CT education can be combined with mathematics textbooks, it is possible to cultivate CT capabilities and mathematical thinking at the same time, and to develop CT capabilities and mathematical thinking in the context of a short read and a story. It also confirmed that there are a number of materials in elementary mathematics textbooks that can be written using programming codes, and it is understood that students will be able to review and gain a relevant understanding of their studies through the process of writing programming codes. In other words, it is shown that the mathematics textbooks currently covered in the curriculum can be used as CT materials and CT education can be conducted naturally.

However, since no field application was performed in this study, it is necessary to apply the results of the study directly to the students and to check the results of the study, and to measure the degree of improvement in the CT capabilities of the students. If the above two further studies are performed in a continuous and in-depth way, the CT capacity is naturally enhanced as students other than CT education are exposed to it through the school curriculum.