박사학위논문

계산적 사고력 신장을 위한 PPS기반 프로그래밍 교육 프로그램

제주대학교 대학원

과학교육학부 컴퓨터교육전공

김병수

2014년 2월

계산적 사고력 신장을 위한
PPS기반 프로그래밍 교육 프로그램

指導敎授 金 鍾 勳

金 柄 秀

이 論文을 敎育學 博士學位 論文으로 提出함

2013年 12月

金柄秀의 敎育學 博士學位 論文을 認准함

審査委員長 金 漢 鎰

委 員 朴 贊 晶

委 員 金 成 植

委 員 李 載 昊

委 員 金 鍾 勳

濟州大學校 大學院

2013年 12月

Programming Education Program based on PPS to Improve Computational Thinking Ability

ByeongSu Kim
(Supervised by professor JongHoon Kim)

A thesis submitted in partial fulfillment of the requirement for the degree of Doctor of Philosophy in Education

2013. 12.

This thesis has been examined and approved.

Major of Computer Education
Faculty of Science Education
GRADUATE SCHOOL
JEJU NATIONAL UNIVERSITY

목 차

표 목 차

그 림 목 차

<국문초록>

계산적 사고력 신장을 위한 PPS기반 프로그래밍 교육 프로그램

김 병 수

제주대학교 대학원 과학교육학부 컴퓨터교육전공

지도교수 김 종 훈

본 연구는 PPS기반의 프로그래밍 교육 프로그램을 개발하고 이를 적용하여 학습자의 계산적 사고력을 증진시킬 수 있는지를 검증하는 데 목적이 있다. PPS란, Paper-and-pencil Programming Strategy의 약자로 “다이어그램, 문장, 코드, 순서도, 표 등의 의미 있는 기호와 상징을 이용하여 계산모델을 스스로 정의하고, 문제에 대한 해결 방법(solution)이나 아이디어를 추상화하여 종이에 직접 쓰거나 그리면서 논리적으로 표현하고자 하는 문제 해결의 방법을 모델링하는 전략”을 의미한다.

본 연구의 목적은 PPS기반 프로그래밍 교육 프로그램에 의해 학습자의 계산적 사고력이 증진될 수 있는지를 검증하는 것이다.

이를 위해 본 연구는 다음과 같은 절차로 진행되었다.

먼저, 관련 연구 동향 파악을 통해 계산적 사고를 구성하는 6개의 사고력(추상적 사고, 비판적 사고, 논리적 사고, 창의적 사고, 재귀적 사고, 알고리즘적 사고)를 선정하고 이들의 관계를 분석적으로 접근하고자 이들을 도식화하여 제시하였다. 이러한 분석은 교육 프로그램의 개발과 검사 도구의 문항 개발에 토대가 되었다.

본격적인 교육 프로그램 개발에 있어서는 가장 먼저 수업 모형을 선정하였다. CPS(Creative Problem Solving, 창의적 문제해결) 모형을 수정·보완하여 수업의 과정을 '계산모델의 분석', '계산화', '문제의 분석·탐색 및 아이디어 발견', '해결 발견', '수용 발견'으로 설정하였다. 수업 내용에 있어서도 프로그래밍을 중심으로 한 컴퓨터과학의 핵심 개념(아이디어)을 10가지(순차구조, 조건분기, 반복, 동시성, 변수, 난수, 알고리즘, 객체, 함수, 재귀)로 추출하였다. 이렇게 선정된 수업 모형과 수업 내용을 바탕으로 각 문제 해결의 설계 단계에 PPS 활동을 적극적으로 활용할 수 있도록 교육 프로그램을 개발하였다.

학습자의 계산적 사고력 신장을 위한 측정은 계산적 창의성과 계산적 인지력의 측정으로 나누어 실시하였다. 계산적 창의성 측정은 계산적 사고력의 구성 요인 중 창의적 사고(창의성)의 측정을 의미하며 보편적으로 널리 사용되는 Torrance의 TTCT 검사를 사용하였다. 계산적 인지력의 측정은 계산적 사고를 이루는 구성 요인 중 창의성을 제외한 추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고를 통합적으로 측정하는 것을 의미하며 본 연구에서 이러한 검사 도구를 직접 개발하여야 할 필요가 있었다.

계산적 인지력 검사 도구의 개발은 평가 문항의 제작, 타당도, 변별도, 난이도, 신뢰도 검증을 통해 문항을 수정·보완하여 동형검사지 A형, B형을 개발하였다. 개발된 검사 도구는 논리적 사고력 검사인 GALT와 높은 상관도를 보였다. 계산적 인지력에 논리적 사고가 포함되었기 때문에 이들의 높은 상관관계는 개발된 계산적 인지력 검사 도구의 타당성을 더욱 높인다고 볼 수 있다. 또한, 계산적 인지력 검사 도구인 A형, B형 검사지는 동형 검사 신뢰도가 검증되었다. 이렇게 개발된 계산적 인지력 검사 도구와 TTCT 창의성 검사 도구가 본 연구의 교육 프로그램 투입의 사전·사후에 사용되어 학습자의 계산적 사고력이 측정되었다.

이러한 검사 도구를 사용하여 본 연구에서는 다음과 같은 결과가 도출되었다.

첫째, PPS기반 프로그래밍 교육 프로그램은 계산적 창의성에 긍정적인 영향을 미친다.

둘째, PPS기반 프로그래밍 교육 프로그램은 계산적 인지력에 긍정적인 영향

을 미친다. 즉, 이러한 결과로 PPS기반 프로그래밍 교육 프로그램은 계산적 사고력에 긍정적인 영향을 미친다고 볼 수 있겠다.

이러한 결과로 본 연구는 컴퓨터과학 교육에서 다음과 같은 기여를 할 수 있다고 판단된다.

첫째, PPS 활동을 기반으로 한 프로그래밍 교육은 계산적 인지력과 계산적 창의성을 모두 증진시킬 수 있다는 가능성의 시사이다. 단순히 프로그래밍 학습 이전에 이루어지는 선행학습으로서의 언플러그드 학습 방법이 아닌, 프로그래밍에서의 문제해결 설계 단계의 학습에 초점을 맞추어 교육을 한다는 것에 더 큰 의의를 찾아야 할 것이다. 즉, 본 연구에서 개발된 PPS기반 프로그래밍 교육 프로그램과 같이 PPS 활동은 다양한 프로그래밍 교육에서 문제 해결의 과정에서 적극적으로 활용되어질 필요가 있다는 것이다.

둘째, 계산적 사고력을 이루는 구성 요인들 간의 관계 분석이다. 본 연구에서 제시하는 구성 요인들의 종류와 그것들의 관계에 대한 논의는 계속적으로 이루어질 필요가 있겠다. 하지만, 본 연구의 결과가 본 연구에서 제시한 각 구성 요인들의 관계 도식에 근거를 마련하였다고 할 수 있다.

셋째, 본 연구에서 개발된 계산적 사고력 검사 도구는 관련 연구에서 계속적으로 활용될 수 있고 재생산성이 높은 연구 결과물이라고 말할 수 있을 것이다.

본 연구의 제한점은 다음과 같다.

첫째, 본 연구의 비교집단은 처치가 없도록 설계되었다. 이는 본 연구의 결과만으로는 PPS 활동 자체의 여부에 따른 프로그래밍의 교육적 효과를 검증할 수는 없다는 의미이다.

둘째, 본 연구에서 사용된 창의성 검사 도구인 TTCT 언어 검사는 창의성의 핵심 구성요인 중 '정교성'을 측정하지는 못하고 있다.

덧붙여, 본 연구의 결과를 확고히 다지기 위해서는 다음의 사항들에 대한 지속적인 검증이 필요할 것이라고 본다.

첫째, 전문가 검증을 통해 계산적 사고력 검사 도구의 타당도를 확보할 필요가 있으며, 신뢰도 확보를 위해 다수의 학생들에게 검사를 투입할 필요가 있다.

둘째, 본 연구의 실험집단은 17명이다. 물론 이는 실험-비교 연구에 미흡한 인원은 아니나, 상관연구를 수행하기에는 부족한 편이다. 즉, 학습자로부터 얻은 계산적 창의성과 계산적 인지력의 상관 연구를 위해서는 추후에 더 많은 연구 참여자를 확보하여 연구를 적용할 필요가 있다.

주요어: PPS. Paper-and-pencil Programming Strategy, 계산적 사고력, 계산적 창의성, 계산적 인지력, 프로그래밍

I. 서론

1. 연구의 필요성

급변하는 현대 사회에서 학문의 영역은 더욱 세분화 되어 가고 있으며 그에 따라 각 분야의 전문 정보의 양은 기하급수적으로 늘어나고 있다. 이러한 시대에 미래의 주역이 될 청소년들에게 무엇을 어떻게 교육해야 할지에 대한 비전(vision) 제시는 현대의 많은 교육자들의 최대 관심사이다. 학문이 세분화 되어 질수록 학생들에게 필요한 능력은 전문 분야의 지식이 아니라 그 지식 안에 있는 핵심적 가치와 원리이다. 또한 각각의 전문 영역에서 발휘해야 하는 능력의 핵심에 자리 잡고 있는 인간의 능력을 파악하고 학습 활동을 통해 이러한 능력을 신장시킬 수 있도록 해야 함에 동의하고 있다(박남기, 2011; Craft, 2010; Pring et al., 2012; Stouffer, Russell & Oliva, 2004).

Wing(2006, 2008)은 인간의 기본·기초적인 능력인 3R(reading, writing, arithmetic)에 계산적 사고(CT: Computational Thinking)를 덧붙여 미래를 살아갈 인간에게 꼭 필요한 능력으로 제시하였다. 계산적 사고에 대한 개념 정립, 응용 및 교육적 목적과 방법에 대한 논의는 컴퓨터과학 분야의 많은 연구자들과 교육자들에게 관심을 불러 일으켰을 뿐만 아니라 컴퓨터과학 분야를 넘어 융합된 미래사회를 위해 비전을 제시하고자 하는 다양한 분야의 학자들도 주목하고 있는 이슈(issue)이다(NRC, 2010, 2011).

또한, 미래학자 앨빈 토플러는 '부의 미래'에서 "미래사회는 조직력에서 창의성으로, 물질적 자산에서 지식자산으로, 그리고 제조 중심에서 서비스 중심으로 다시 개편될 것이며, 한 개인의 창의성이 전 세계를 뒤흔들 수 있는 유망사업으로 성공할 수 있다."라고 하였고, 역사학자 토인비는 "역사의 변화는 언제나 창조적 소수에 의해 주도된다."라는 말을 남겼다(이수원, 2011). 이 학자들은 미래 사회에서 가장 주목받아야 할 인간의 능력은 창의성이라고 얘기하고 있다. 이러한 관점은 많은 학자들의 지지를 받으며 관련 연구가 계속되고 있다

(김영정, 2005; 삼광현, 노명우, 강정석, 2012; Rowe, 2004).

계산적 사고력과 창의성의 관계에 대한 정의와 실증적 상관 연구는 추후의 과제로 남겨두더라도, 이러한 각각의 능력들과 프로그래밍과 관련된 연구는 활발히 수행되어져 왔다. 많은 학자들은 컴퓨터과학 분야에서 계산적 사고력 증진을 위한 가장 효과적인 활동으로 프로그래밍을 손꼽고 있다(Cutts, Esper & Simon, 2011; Felleisen & Krishnamurthi, 2009; Kafai & Burke, 2013; Orr, 2009; Resnick et al., 2009). 이러한 주장을 뒷받침하는 실증적인 연구가 최근 활발히 이루어지고 있으며 특히, 계산적 사고력의 평가 도구 개발과 이를 증진하기 위한 프로그래밍 활동에 대한 연구도 최근 많아지고 있다(김병수, 2010; 김병수, 김종훈, 2012a, 2012b, 2012c; 이은경, 2013; Brennan & Resnick, 2012; Koh, Basawapatna, Bennett & Repenning, 2010; Seiter & Foreman, 2013).

창의성 증진을 위한 프로그래밍 교육에 대한 연구 또한 활발하며, 많은 학자들은 프로그래밍이 창의성을 증진시키기 위해 좋은 활동임을 주장하고 있고(김갑수, 2010; 전성균, 서영민, 이영준, 2011; Gallardo, Julia & Jorda, 2008), 이러한 주장은 관련 연구들을 통해 검증되고 있다(유정수, 이민희, 2009; 전성균, 이영준, 2012; Knobelsdorf & Romeike, 2008; Lewandowski, Johnson, & Goldweber, 2005; Long, 2007; Romeike, 2007a, 2007b).

계산적 사고 또는 창의성 증진을 위한 기존 연구들을 평가함에 있어서 주목할 것은, '학습 활동이나 교수·학습 과정에서 그러한 능력을 증진시키기 위한 구체적인 방법이나 전략적 특징을 얼마나 가지고 있는가?'라는 것이며 '그 내용과 방법 또는 전략이 얼마나 그러한 능력과 연관있는가?'를 밝히고 있지 못하고 있다는 것이다.

초기 계산적 사고력의 관련 연구에서는 창의성에 대한 언급은 활발하지 않았지만 최근 들어서 계산적 사고력 내에 창의성이 포함되어야 한다고 주장하고 있다. 이는 계산적 창의성(computational creativity)이라는 이름으로 연구되어 지고 있다(.Morris & Secretan, 2009; Settles, 2010). 본 연구에서도 계산적 사고력 내에 창의적 사고(창의성)가 내재되어 있다고 보며 다른 구성 요인과의 관계에서도 분석하고자 하였다.

프로그래밍 활동이 하나의 소프트웨어 설계와 구현이라는 문제 해결 과정이

라는 관점에서 바라보았을 때, 설계의 단계에서의 계획과 구상은 매우 중요하다(신우창, 2003; Booch, 2005; Fraser, Kurnar & Vaishnavi, 1994; Jacobson, 1999). 이러한 관점에서 본 연구에서는 학습자가 문제를 해결하는 과정의 설계 및 구현 단계에서, 종이와 연필을 이용하여 다이어그램, 문장, 코드, 순서도, 표 등의 의미있는 기호와 상징을 이용하여 자신의 해결책(solution)을 논리적으로 구상하고 표현하는 PPS(Paper-and-pencil Programming Strategy)[1] 활동을 통한 프로그래밍 학습이 계산적 사고력이 증진될 수 있음을 보이고자 하였다. 계산적 사고의 핵심 기능인 추상화가 컴퓨터과학 전반에 걸친 가장 핵심적인 아이디어라는 주장을 바탕으로(Bennedssen & Caspersen, 2008; Koppelman & van Dijk, 2010; Kramer, 2007), PPS는 계산적 사고의 추상화 기능을 적극적으로 활용한 전략이라고 할 수 있으며 이를 통해 문제 해결 과정에서의 계산적 사고력의 증진을 이끌어 낼 수 있을 것이라고 본다.

2. 연구의 내용과 절차

본 연구에서의 연구 내용은 다음과 같다.

첫째, 컴퓨터과학 분야에서 프로그래밍 교육을 통해 계산적 사고와 창의성 증진을 이끈 기존의 실증적인 실험 연구들에 대해 조사하여 그 특징들을 비교·분석하여 제시한다.

둘째, 계산적 사고의 분석적 접근을 통해 계산적 사고의 구성 요인을 제시한다. 창의적 사고(창의성)를 포함하여 추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고들이 어떻게 계산적 사고를 구성하는지에 대한 논의가 이루어진다.

셋째, 계산적 사고력 증진을 위한 PPS기반의 프로그래밍 교육 프로그램을

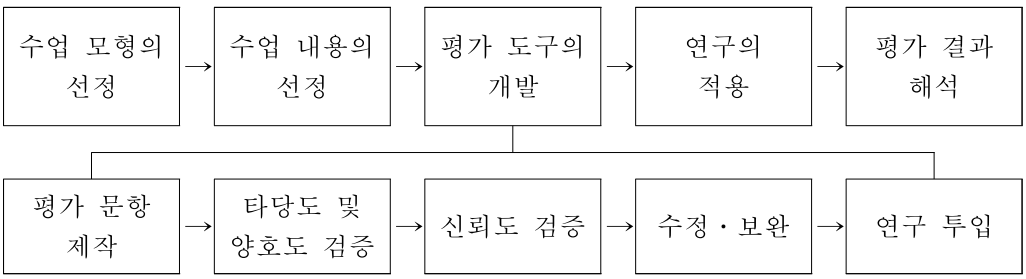
1) PPS라는 용어를 한국어 번역으로는 '손코딩'으로 바꾸어 쓸 수 있을 것이다. 그러나 본 연구에서는 '코딩'이라는 의미가 설계의 측면의 문제가 아닌 단순히 코드(code)를 옮겨 쓰는 활동 자체를 일컫는 용어로 인식되어 질 수 있다는 점에서 해당 용어를 사용하지 않았다. PPS는 코딩의 범위를 넘어 새로운 계산모델을 정의하고, 문제를 계산으로 표현하며 문제의 해결책을 다양한 방법으로 쓰거나 그릴 수 있는 프로그래밍 또는 모델링 전략을 일컫는다.

설계한다.

넷째, 계산적 인지력[2]을 측정할 수 있는 검사 도구를 개발하고 이를 활용하여 학습자의 계산적 사고력을 평가한다[3].

다섯째, 연구의 적용 후 학습자의 계산적 사고력의 변화를 검사 도구를 이용하여 수집하고 이를 통해 연구의 결과를 도출·분석하여 효과를 검증한다.

PPS기반 프로그래밍 교육 프로그램의 개발 설계 및 적용과 관련된 본 연구의 연구 절차는 [그림 Ⅰ-1]과 같다.



[그림 Ⅰ-1] 연구 절차

3. 연구의 기대 효과

본 연구의 기대 효과는 다음과 같다.

첫째, 문제 해결 자체에 집중하기 위해 설계 단계에서의 구상을 강조하는 전략의 프로그래밍 학습이 학습자의 계산적 사고력을 증진시킬 수 있다는 결과를 보여준다면, 이는 기존의 전통적 프로그래밍 언어 학습이나 로봇 프로그래밍 등과 같이 학습자의 동기유발을 위한 다른 교육적 수단을 사용하지 않고서라도 일반화하기 쉽다는 장점과 더불어 프로그래밍 학습의 질을 높일 수 있을 것이라고 기대할 수 있다.

---

2) 계산적 인지력이란 계산적 사고력의 구성 요인 중 창의성을 제외한 추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고를 일컫는 융합적 사고를 지칭한다.

3) 본 연구에서는 창의성을 계산적 창의성으로 명명하여 계산적 인지력과 구분하여 사용하고, 검사 도구로는 보편적으로 널리 사용되고 있는 Torrance의 TTCT 검사를 사용하였다. 따라서 계산적 창의성 검사도구의 개발은 연구 내용의 범위에서 배제하였다.

둘째, 프로그래밍과 계산적 사고력의 관계를 정의함으로써 구체적인 프로그래밍 교육 방법과 전략을 제시할 수 있으며, 이는 이후의 관련 연구에 도움이 될 수 있을 것이라 기대한다.

셋째, 본 연구에서 개발한 계산적 인지력 검사 도구를 활용하여 이후의 관련 연구에서도 학습자의 계산적 인지력을 측정·평가 할 수 있을 것으로 기대한다.

넷째, PPS기반 프로그래밍 교육 프로그램은 하나의 프로그래밍 언어로만 가능한 교육 프로그램이 아니기 때문에 다양한 프로그래밍 언어를 이용하거나, 언플러그드식의 컴퓨터과학 문제 해결 학습에도 적용되어 긍정적인 효과를 기대해 볼 수 있을 것이다.

4. 연구의 제한점

본 연구가 갖는 연구의 제한점은 다음과 같다.

첫째, 비교집단의 처치 설정에 관한 것이다. 본 연구에서의 비교집단은 처치가 없다. 즉 실험집단과 비교하였을 때 'PPS기반 프로그래밍 교육 프로그램' 자체의 교육적 효과를 알아볼 수 있다는 것이다. 연구 설계에서 하나의 비교집단을 더 투입하여 같은 주제로 프로그래밍 학습을 투입하여 PPS 활동의 여부에 따른 교육적 효과도 검증할 수 있었을 것이다. 본 연구에서는 현실적인 제한으로 이러한 연구 설계는 이루어지지 못했다.

둘째, 창의성 검사 도구의 측정 변인에 관한 것이다. 본 연구에서 사용한 창의성 측정 도구인 TTCT 언어 검사는 창의성의 구성 요인 중 '유창성', '융통성', '독창성'을 측정할 수 있다. 다시 말하면, 프로그래밍 활동에서 요구되는 창의성 요인 중 '정교성'에 대한 측정은 검사 도구의 한계로 측정이 불가했다.

셋째, 창의성의 보편성과 특수성에 관한 것이다. 본 연구에서는 계산적 사고에 필요한 창의성을 계산적 창의성이라 명명하였지만 이는 계산적 인지력과의 구분을 위한 것이지 창의성의 영역 특수성 측면을 강조하고자 한 것은 아니다.

물론 일반적인 창의성과 컴퓨터과학에서의 창의성의 상관관계에 대한 논의가 더 필요하겠지만, 본 연구에서는 보편적인 창의성과 컴퓨터과학에서의 창의성이 어느 정도의 상관관계가 있다고 보고, 보편적 창의성의 측정을 통해 계산적 창의성의 증진 여부를 가늠해 보고자 한다. 이는 하나의 가정이므로 이러한 가정 자체가 연구의 제한점이 될 수 있다고 본다.

II. 관련연구 동향

1. 계산적 사고(Computational Thinking)

Denning(2003)은 다익스트라(Dijkstra)가 “천문학이 망원경을 연구하는 것이 아닌 것처럼 컴퓨팅은 더 이상 컴퓨터를 의미하는 것이 아니다”라고 말한 것처럼 컴퓨터는 컴퓨터과학의 목적이 아닌 이 학문 영역에서의 연구를 위한 하나의 도구일 뿐이라고 주장하였다. 다시 말하면, 컴퓨터과학은 컴퓨터를 어떻게 활용하는지에 대해 가르치는 것이 아니라 계산과 컴퓨팅에 관한 학문임을 강조하고 있는 것이다. 즉, 계산적 사고에 대해 언급하기 전에 먼저 계산 자체에 대한 학문적 접근이 필요하리라고 본다.

1) 계산(Computation)

“인지과학(cognitive science)과 언어 철학(linguistic philosophy)에 대한 심도 있는 이해 없이는 기계 지능(machine intelligence)에 대해 논의할 수 없다”는 Denning(2003)의 주장은 하나의 학문으로서 컴퓨터과학에 대한 탄생 배경을 언급하고 있다.

인지과학이란 인간의 지식의 본질은 물론 컴퓨터를 포함하는 각종 인공물들(artifacts)에 편재된 지식체계의 본질을 연구하는 종합과학이다(박창호 외, 2007). 인지과학 측면에서 컴퓨터의 등장은 인간의 마음(mind)에 대한 연구가 불가능하다는 회의론을 잠재웠다. 컴퓨터가 기호로 이루어진 프로그램에 의해 어떠한 유령적인 현상 없이 작동되어지는 것처럼 인간의 뇌도 마음에 의해 작동되어지는 것이라고 생각할 수 있게 되었다. 인간이 정보를 처리하는 과정에 대한 인지과학의 연구는 컴퓨팅 머신이 상황을 지각하고, 통신하고, 정보를 처리하는 방법에 대한 컴퓨터과학의 연구에도 많은 영향을 끼쳤다(Johnson-Laird, 1988).

컴퓨터과학에서 인지과학에 대해 관심을 기울일 필요가 있는 이유는 바로 '계산'에 대한 논의가 시작되었기 때문이다. 1930년대의 Kurt Gödel(Gödel, Kleene & Rosser, 1934), Alonzo Church(Church, 1936), Alan Turing(Turing, 1936) 등에 의한 계산에 대한 접근은 컴퓨터과학 분야의 발전에 많은 영향을 끼쳤다. 인지과학자 Johnson_Laird는 "계산은 우리들로 하여금 해결되어야 할 필요가 있는 문제들을 어느 정도의 정확성을 가지고 설정할 수 있도록 하였다"라고 그의 저서에 옮겼다. Conery(2010a, 2010b)는 "계산이란 문제해결을 이끄는 간결하고 구조적으로 정의된 단계의 순차적 나열을 의미한다. 문제 자체는 모호함이 없이 정확하게 정의되어야 하며, 문제를 해결하는 계산의 각 단계는 매우 구체적인 용어로 묘사되어야 한다"고 정의하였다. 이러한 정의에서 '문제'는 문제 자체와 그 해결책(solution)에 대한 형식적 정의가 기호 형식으로 부호화되어야 하며, '단계'는 하나의 기호 집합을 새로운 기호 집합으로 변환하는 기호 처리 조작을 의미한다. 이러한 그의 정의는 무엇을 계산이라 할 수 있으며 무엇을 계산이라 할 수 없는지에 대한 정확한 분류를 돕는데 기여했다.

앞서 인지과학에서의 마음과 뇌의 관계를 컴퓨터과학에서는 프로그램과 기계[4]와의 관계로 언급했다. 그렇다면 '뇌에서 일어나는 사고[5](thought)를 계산이라고 할 수 있는가?' 또는 '모든 계산은 사고인가?'라는 질문을 던져 볼 수 있을 것이다. 즉 '문제'와 '단계'에 대한 정의와 함께 계산에 대한 정의를 통해 이러한 계산 가능성(computability)에 대한 문제들의 분류가 가능해졌다. 즉, 컴퓨터과학의 측면에서 자연과 인공물에 편재된 모든 문제에 대해서 계산할 수 있다고 보는 것은 아니나 구조적으로 잘 정의만 할 수 있다면 많은 문제들을 표현하고 이를 계산할 수 있다는 것이다. Denning(2007, 2010)은 계산의 범위를 단순히 수학, 공학적 문제의 범위를 벗어나 자연에서부터 인간의 일상생활, 물리적 생활공간과 사회과학까지를 포함하여야 한다고 하였다. 이러한 이유로 그는 컴퓨터과학이 자연과학(natural science)임을 역설하였다.

김형철(2011)의 연구에서는 이러한 관점들이 잘 나타나게 '계산'에 대해 정의하였다. 그는 "계산은 그것이 기반하고 있는 계산모델(computational model)의

4) 프로그램과 기계의 관계를 일반적으로 소프트웨어와 하드웨어의 관계로 생각할 수 있다.
5) 뇌에서 일어나는 사고의 일부 또는 전체

관점에서 정의되는 처리 과정이다"라고 하였다. 본 연구에서는 이를 구체적으로 정의하여, 계산이란 "정의가 분명한 유한의 명령어 집합(또는 규칙)을 이용하여 입력 표현(input representation)을 출력 표현(output representation)으로 사상(mapping)하는 과정을 정의한 상태들의 순차[6]"를 의미한다고 정의한다.

2) 계산모델(Computation Model)

계산은 그것이 기반하고 있는 계산모델의 관점에서 정의되는 처리과정이다. 계산모델은 컴퓨팅 시스템에 대한 수학적 추상체[7]로, 계산에서 사용될 수 있는 연산들의 집합과 각 연산의 비용을 정의한다. 따라서 어떤 계산에 대해 정확히 이해하려면 그것이 기반하고 있는 계산 모델에 대해 올바르게 알아야 한다(김형철, 2011). 이러한 계산모델에 대한 연구가 중요한 이유는 문제 상태의 유형에 따라 적정 계산모델을 선택하거나 변형함으로써 원하는 계산을 생성해주는 알고리즘을 보다 효과적으로 찾게 해준다.

(1) 튜링기계(TM: Turing Machine)

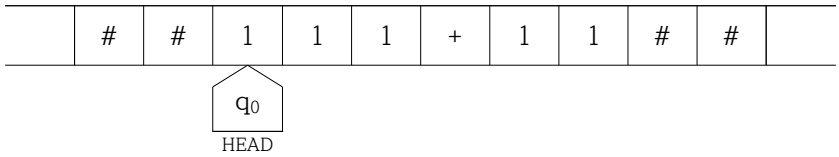
튜링기계란 영국의 수학자이며 논리학자인 튜링(Alan M. Turing)이 추상적으로 고안한 계산 기계이다(Davis, 1958). 하나의 튜링기계는 다음의 구성 요소들을 갖는다(김재권, 1997).

- 양쪽 방향으로 계속 이어지는 '네모 칸들'로 이루어진 하나의 테이프
- 임의의 주어진 시점에 그 테이프의 네모 칸들 중의 한 곳에 위치해 있는 스캐너-프린터('헤드')
- 유한한 집합의 내적 상태들 혹은 배열들, q_0, …, q_n
- 기호들을 구성하는 유한한 수의 알파벳, b_1, …, b_m

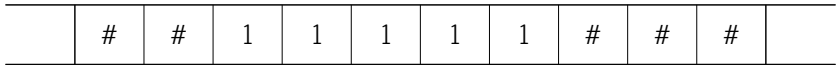
이러한 튜링기계에서 2와 3을 더하는 문제가 있다면 이에 대한 덧셈 과정을 정의해주는 지침들의 완전한 집합인 기계표를 정의할 수 있다.

6) 정의된 상태들의 순차라 함은 하나 이상의 연속된 순차의 경우도 모두 포함한다.
7) 본 연구에서의 계산모델은 수학적 추상체뿐 아니라 언어적 추상체를 포함한다.

<초기 상태>



<목적 상태>



<기계표>

	q_0	q_1
1	$1Rq_0$	# Halt
+	$1Rq_0$	
#	$\#Lq_1$	

[그림 Ⅱ-1] 2+3 연산을 위한 튜링기계의 상태와 기계표

(2) 유한상태기계(FSM: Finite State Machine)

유한상태기계는 유한한 메모리를 가진 기계에 대한 추상적 모델이다. 특정 시점에 유한상태기계의 상태는 그 당시 메모리에 저장되어 있는 값들의 총체로 정의될 수 있으며, 메모리 용량이 유한해 서로 다른 상태의 개수가 유한하기 때문에 유한상태기계라 한다(김형철, 2011).

유한상태기계는 아래와 같이 구성되며 $M=(S, I, O, f, g, s_0)$라고 나타낸다(함미옥, 홍영진, 2008).

- 상태의 유한집합 S
- 입력 심볼의 유한집합 I
- 출력 심볼의 유한집합 O
- 추이함수(transition function) $f: S \times I \rightarrow S$
- 출력함수(output function) $g: S \times I \rightarrow O$
- 초기상태 s_0

예를 들어 유한상태기계 $M=(S, I, O, f, g, s_0)$가 $S=\{s_0, s_1\}$, $I=\{a, b\}$, $O=\{0, 1\}$이라고 할 때 아래의 표에 의한 함수 $f: S \times I \rightarrow S$와 $g: S \times I \rightarrow O$에 대한 정의는 아래와 같다.

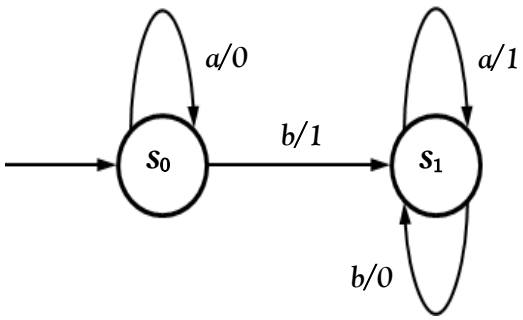
<표 Ⅱ-1> 유한상태기계 정의의 예

	f		g	
S \ I	a	b	a	b
s_0	s_0	s_1	0	1
s_1	s_1	s_1	1	0

$f\{s_0, a\}=s_0$, $f\{s_0, b\}=s_1$, $f\{s_1, a\}=s_1$, $f\{s_1, b\}=s_1$

$g\{s_0, a\}=0$, $g\{s_0, b\}=1$, $g\{s_1, a\}=1$, $g\{s_1, b\}=0$

이 유한상태기계에 대한 상태전이도표는 아래와 같다.



[그림 Ⅱ-2] 상태전이도표

(3) 유한상태 오토마타(FSA: Finite State Automata)

유한상태 오토마타는 출력 기호가 0과 1 중 하나이고, 현재 상태가 직전 출력을 결정짓는 유한상태기계를 말한다. 즉 어느 한 상태로 전이할 때 출력되는 기호 모두가 0으로 통일되어 있거나 1로 통일되어 있다는 의미이다(김형철, 2011). 유한상태 오토마타는 아래와 같이 구성되며 $M=(S, I, O, f, s_0, A)$라고 나타낸다(함미옥, 홍영진, 2008).

· 상태의 유한집합 S

· 입력 심볼의 유한집합 I

· 추이함수 f: $S \times I \rightarrow S$

· 초기상태 s_0

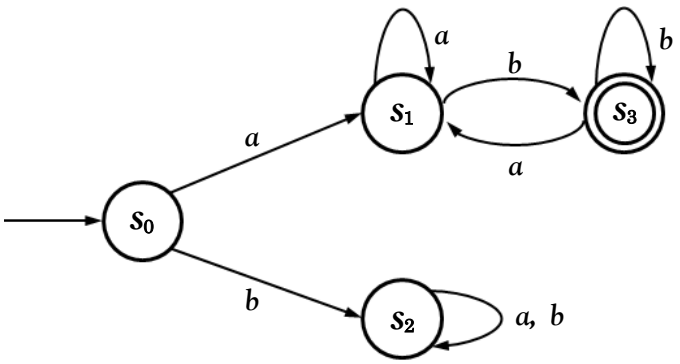
· S의 부분집합인 수용상태 A

유한상태 오토마타 $M=(S, I, O, f, s_0, A)$에서 $S=\{s_0, s_1, s_2, s_3\}$, $I=\{a, b\}$, $A=\{s_3\}$일 때 상태표는 다음과 같다.

<표 Ⅱ-2> 유한상태 오토마타의 상태표

	f	
S \ I	a	b
s_0	s_1	s_2
s_1	s_1	s_3
s_2	s_2	s_2
s_3	s_1	s_3

이 유한상태 오토마타의 상태전이도표는 아래의 그림과 같다.



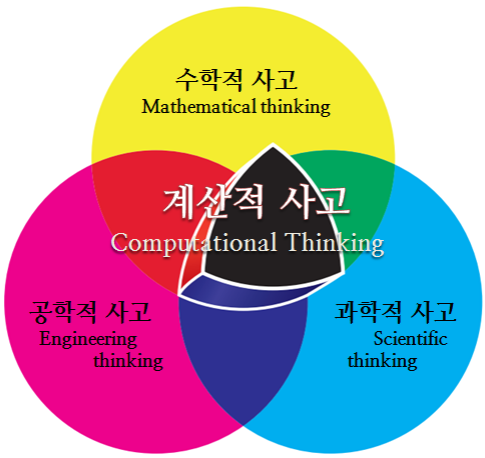
[그림 Ⅱ-3] 유한상태 오토마타의 상태전이도표

3) 계산적 사고의 정의와 구성 요인

(1) 계산적 사고의 정의

계산적 사고는 Wing(2006)에 의해 처음 그 개념이 정의되어 소개되었다. Wing(2006, 2008)은 “계산적 사고는 컴퓨팅의 핵심 개념을 기반으로 문제를

해결하고 시스템을 디자인하고 인간의 행동양식을 이해하려는 접근 방법"이라고 정의하였다. "계산적 사고는 하나의 분석적 사고이다. 문제를 해결하는 일반적인 방법에서는 수학적 사고를 공유하며, 실세계의 제약 안에서 작동되는 크고 복잡한 시스템을 설계하고 평가하는 접근에는 공학적 사고를 공유하고, 인간의 행동, 마음, 지능, 계산가능성에 대한 이해를 위한 접근에서는 과학적 사고를 공유한다"고 설명하였다.



[그림 Ⅱ-4] 계산적 사고와 공유되는 사고들

Denning(2009, 2010)은 "계산적 사고는 원래 1950, 1960년대 Newell, Perlis, Siman에 의해 연구된 알고리즘적 사고(algorithmic thinking)로 불리우던 개념이 현대에서는 계산적 사고라 불리운다."고 하였다. 많은 학자들은 Wing의 계산적 사고에 대한 정의가 실제 인간의 어떠한 사고 능력을 말하는 것인지에 대한 구체적인 예가 더욱 필요함을 언급하였다(NRC, 2010, 2011).

계산적 사고에 대한 정의를 내리기 위해 계산적 사고에 대한 국내·외의 관련 연구를 <표 Ⅱ-3>과 같이 정리하였다.

<표 Ⅱ-3> 계산적 사고 관련 국내·외 관련 연구

연구자 (발행연도)	연 구 내 용	
Wing (2006, 2008)	제목	Computational thinking
	정의	Computational thinking is taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing. (계산적 사고는 컴퓨팅의 기본 개념과 원리에 기반하여 문제를 해결하고, 시스템을 설계하고, 인간의 행동양식을 이해하고자 하는 접근 방법이다.)
CSTA Standards Task Force (2011)	제목	CSTA K-12 Computer Science Standards (revised 2011)
	정의	Computational thinking is an approach to solving problems in a way that can be implemented with a computer. (계산적 사고는 컴퓨터를 활용해 구현할 수 있는 방법으로 문제를 해결해 나가는 접근 방법이다.)
NRC (2010)	제목	Report of a workshop on the scope and nature of computational thinking
	정의	상당히 많은 학자들이 자신 나름대로의 계산적 사고에 대한 정의를 내리고 있다. 그 중 몇가지 주목할 만한 정의는 다음과 같다 (p.11～12). • Peter Lee – Computational thinking is fundamentally about expanding human mental capabilities through abstract tools that help mangage complexity and allow for automation of tasks. (계산적 사고란 근본적으로, 작업의 자동화를 감안하고 복잡도 관리를 돕는 추상적 도구들을 통한 인간의 정신적 능력의 확장이다.) • Bill Wulf – Computational thinking focuses on processes and abstract phenomena that enable processes. (계산적 사고란 처리를 가능하게 하는 추상적 현상과 처리에 초점을 맞춘다.) • Sussman – Computational thinking has an underlying lingustic structure. (계산적 사고는 언어학적 구조 기반을 갖고 있다) • Wing and Sussman – Computational thinking could be seen as a bridge between science and engineering. (계산적 사고는 과학과 공학 사이의 다리처럼 보여질 수 있다,)
NRC (2011)	제목	Report of a workshop on the pedagogical aspects of computational thinking
	정의	계산적 사고에 대한 정의가 첫 번째 워크샵의 주제였다면, 두 번째 워크샵은 계산적 사고에 대한 교육학적 관점들에 대한 논의가 이루어졌다. (p.6～7) • Peter Henderson – Computational thinking as generalized problem solving with constraints. (제약을 가진 일반화된 문제 해결로서의 계산적 사고)

이은경[1] (2009)	제목	Computational Thinking 능력 향상을 위한 로봇 프로그래밍 교수 학습 모형
	정의	21세기를 살아가는 모든 사람이 갖추어야 할 기본적인 사고 능력으로 컴퓨터 과학의 기본 개념과 원리에 따른 문제 해결, 시스템 설계, 인간 행동의 이해를 포함하는 추상적 사고 능력이다. 이는 문제 해결을 위해 적합한 추상적 개념을 선택하고 구성하기 위한 추상화 능력과 추상적 개념을 자동화하기 위해 적합한 컴퓨팅 도구를 선택하고 사용하기 위한 능력을 포함한다.
	평가	자체개발 16개 문항(문항당 1점, 총점 16점).
유중현, 김종혜[3] (2008)	제목	문제 해결과정에서의 계산적 사고력에 대한 개념적 고찰
	정의	Wing(2006)의 정의를 인용함.
김종혜[1] (2009)	제목	계산적 사고 기반의 문제 해결 능력 향상을 위한 중등 교육 프로그램
	정의	Wing(2006)의 정의를 인용함.
	평가	자체개발 15개 문항
서성원[2] (2010)	제목	TPL과 VPL을 활용한 로봇 프로그래밍 교육이 계산적 사고력에 미치는 영향
	정의	이은경(2009)의 정의를 인용함.
	평가	이은경(2009)의 평가지를 수정・보완하여 사용 중.
김형철[2] (2011)	제목	컴퓨터과학 교육용 계산 원리 학습도구의 기능요소 고찰
	정의	계산적 사고는, 좁은 의미로는 계산 시스템(computational system)을 활용해 효과적으로 작업하기 위해 습득해야 할 사고방식이나 태도이며, 넓은 의미로는 세상을 이해하는 양식(단순한 방법을 초월한 양식, 광범위한 인간노력에 두루 접목 가능한 양식)이다.
송정미[2] (2011)	제목	스크래치를 활용한 계산적 사고 기반 퍼즐교육
	정의	Wing(2006)의 정의를 인용함.
권대용[1] (2011)	제목	Algorithmic brick based tangible robot and hybrid programming environments for enhancing computational thinking
	정의	계산적 사고는 21세기 중반까지 모든 사람들이 사용하게 될 필수적 기술(skill)로써, 컴퓨터과학의 기본 개념과 원리에 따른 문제 해결, 시스템 설계, 인간 행동의 이해를 포함하며 컴퓨터과학의 다양한 영역을 반영하는 정신적 도구의 범위를 포함한다.
	평가	논리적 사고력 검사(GALT)
박지연[2] (2012)	제목	Computational Thinking 능력과 유아놀이와의 연관성 분석
	정의	Wing(2006)의 정의를 인용함.
이시내[2] (2013)	제목	계산적 사고 기반 문제해결활동의 상호작용 분석을 위한 프레임워크 개발
	정의	Wing(2006)의 정의를 인용함.

[1] 박사학위 논문, [2] 석사학위 논문, [3] 학회지 논문 또는 학술대회 발표 논문

대부분의 연구가 계산적 사고의 정의를 Wing(2006)의 정의대로 사용하고 있

었지만 계산적 사고를 이루고 있는 그 구성 요인에 대한 분석은 연구마다 달랐다. 본 연구에서 정의하는 계산적 사고의 개념은 다음과 같다.

계산적 사고는 좁은 의미로는 “계산 시스템(computational system)을 활용해 효과적으로 작업하기 위해 습득해야 할 사고방식이나 태도”이며, 넓은 의미로는 “세상을 이해하는 양식[8]”으로(김형철, 2011), “실세계에서부터 디지털세계를 망라하여 자연적으로 또는 인간 사회에 의해 존재하는 다양한 현상 속의 계산을 계산 대행자(computational agent)를 이용하여 발견하거나 새롭게 창조하기 위해 습득해야 할 인간의 사고 양식과 태도”를 의미한다.

(2) 계산적 사고의 구성 요인

계산적 사고의 정의뿐 아니라 구성 요인이 무엇인가에 대한 접근은 연구마다 다양하다. 계산적 사고와 관련된 국내·외 연구들에서 계산적 사고의 구성 요인을 제시한 연구들을 정리하면 <표 Ⅱ-4>와 같다.

<표 Ⅱ-4> 계산적 사고 관련 연구에서의 구성 요인 분석

구 분	Wing (2006)	유중현 김종혜 (2008)	이은경 (2009)	Den -ning (2010)	송정미 (2011)	권대용 (2011)	Selby (2012)	Werner et al. (2012)	Walden et al. (2013)
추상적 사고	○		○				○	○	
재귀적 사고	○	○	○						
분해적 사고	○						○		
예방/보호/복구적 사고	○								
경험적 추론	○								
논리적 사고		○	○		○				
비판적 사고		○	○						○
동시적 사고			○						
선행적 사고			○						
전략적 사고			○						
절차적 사고			○						
알고리즘적 사고		○		○	○	○	○	○	
효율적인 솔루션					○				
과학적 사고					○				
혁신적 사고					○				
문제해결력									○
일반화							○		
모델링 능력							○	○	

8) 단순한 방법을 초월한 양식, 광범위한 인간노력에 두루 접목 가능한 양식을 말한다.

관련 연구들에서 제시한 계산적 사고를 구성하고 있는 다양한 요인들에서 주로 공통적인 것들은 추상화 사고, 재귀적 사고, 논리적 사고, 비판적 사고, 알고리즘적 사고라고 할 수 있다. 각 연구에서 구성 요인으로 제시되는 사고 또는 능력들을 지칭하는 용어들이 실제로 독립적인지에 대해서는 모호한 부분이 있다. 예를 들어, 일반화(generalization)와 모델링(modeling)을 위한 능력은 상당히 비슷한 개념일 수도 있으며, 분해적 사고와 분석적·비판적 사고의 개념도 어느 정도 공통적인 부분이 있다고 할 수 있겠다.

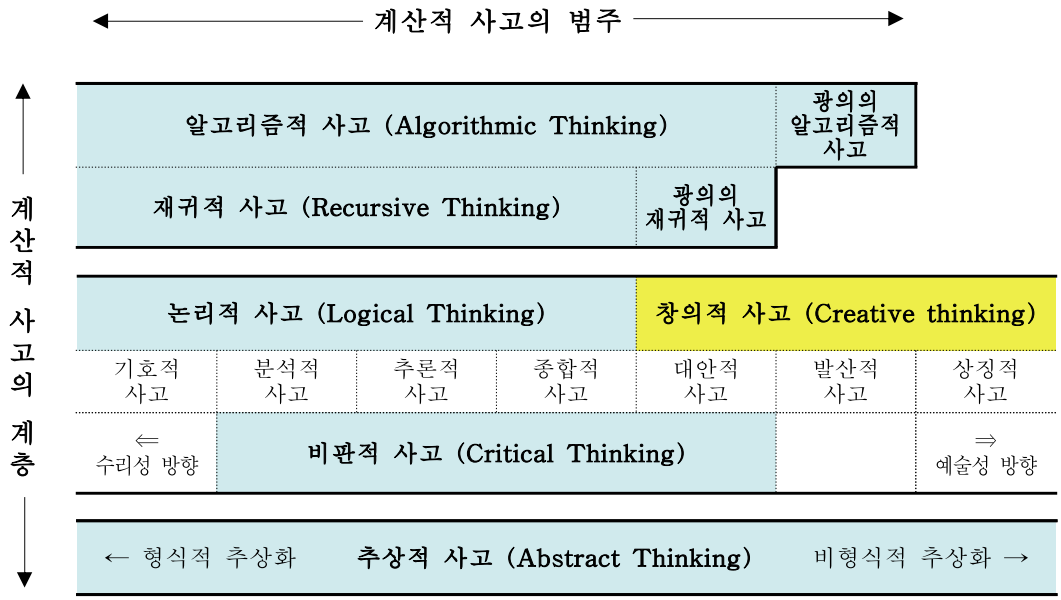
본 연구에서는 기존 연구에서 제시한 다양한 구성 요인들을 분석한 결과, 공통적이며 핵심적인 5가지의 구성 요인을 중심으로 계산적 사고의 구성 요인으로 제시하고자 하였으며, 최근 계산적 사고에서 주목하고 있는 창의성과 계산적 사고와의 관계를 분석하고자 하였다.

이러한 분석에 가장 기준이 된 연구는 김영정(2004)의 '비판적 사고: 비판적 사고와 공학교육'이다. 김영정은 비판적 사고에 필요한 고차적 사고력을 총 7범주로 나누어 도식화하여 제시하였다(그림 Ⅱ-5).

<table>
<tr><td>⇐
수리성 방향</td><td colspan="4">비판적 사고</td><td></td><td>⇒
예술성 방향</td></tr>
<tr><td>Foraml Symbolic Thinking
기호적 사고</td><td>Analytical Thinking
분석적 사고</td><td>Inferential Thinking
추론적 사고</td><td>Synthetical Thinking
종합적 사고</td><td>Alternative Thinking
대안적 사고</td><td>Divergent Thinking
발산적 사고</td><td>Material Symbolic Thinking
상징적 사고</td></tr>
<tr><td></td><td>개념적 분석
텍스트 분석</td><td>분석적 추론
: 연역
종합적 추론
: 귀납귀추</td><td>논리 퍼즐
의사 결정
상황 추리
민감성</td><td>관점/발상전환
대안 창안,
시야/시계확장
시각/지평전환
재정의</td><td>유창성
융통성
독창성
정교성</td><td></td></tr>
<tr><td colspan="4">논리적 사고</td><td colspan="3">창의적 사고</td></tr>
<tr><td colspan="5">광의의 논리적 사고</td><td colspan="2">협의의 창의적 사고</td></tr>
<tr><td colspan="3">협의의 논리적 사고</td><td colspan="4">광의의 창의적 사고</td></tr>
</table>

[그림 Ⅱ-5] 사고력의 7범주 (김영정, 2004)

김영정이 제시한 비판적 사고, 논리적 사고, 창의적 사고를 기반으로 추상적 사고, 재귀적 사고, 알고리즘적 사고와의 관계를 [그림 Ⅱ-6]과 같이 제시하고자 한다.



※ 계산적 인지력 (computational cognition)
: 추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고
※ 계산적 창의성 (computational creativity): 창의적 사고

[그림 Ⅱ-6] 계산적 사고의 범주와 계층

[그림 Ⅱ-6]에서 계산적 사고의 구성 요인간의 관계와 더불어 창의적 사고(창의성)와의 관계에 대해서 나타내고자 했다. 이는 본 연구에서 계산적 사고를 바라보는 관점을 나타내고자 하는 시도이며, 하나의 제안으로 제시하고자 한다.

① 비판적 사고(Critical thinking)

비판적 사고에 대한 개념과 하위 요소들에 정의는 매우 다양하다(이강범, 2008). 비판적 사고와 관련한 연구 중에서 김영정(2004)의 비판적 사고론은 연구의 성과와 그 영향성 모두 인정을 받고 있다(최훈, 2010). 비판적 사고는 때때로 '비판적-창의적 사고(critico-creative thinking)'라고 불린다. 첫 번째 이유

는 '비판적 사고'라는 용어 자체가 부정적으로 들린다는 이유이고, 두 번째 이유는 비판적 사고가 다른 사람의 논증이나 아이디어들을 잘 평가하기 위해서는 종종 매우 상상력이 풍부하고, 다른 가능성들, 대안적인 고려들을 창의적으로 찾아내야 하기 때문이다. 하지만 비판적-창의적 사고라는 용어는 다소 길고 어색하기 때문에 비판적 사고라는 용어가 일반적으로 널리 사용되고 있다. 즉, 비판적 사고라는 의미는 긍정적이고 상상적인 측면까지를 포괄하고 있다(김영정, 2004).

김영정(2004)은 단순 기억 능력이나 상기(recall) 능력과 같은 저차원적 사고 능력을 제외한 고차적 사고 능력을 7범주로 구별하여 제시하였다([그림 Ⅱ-5] 참조). 수리성 방향의 최고 능력으로 기호적 사고(Formal Symbolic Thinking)가 있으며, 예술성 방향의 최고 능력으로 상징적 사고(Material Symbolic Thinking)가 있다. 기호적 사고는 수렴적 사고/수직적 사고의 최고봉이라 할 수 있으며, 상징적 사고는 발산적 사고/수평적 사고의 최고봉이라고 할 수 있다. 이런 점에서 이 두 사고능력은 매우 중요한 사고능력이기는 하나, 보다 보편적이고 일반적 사고 능력인 비판적 사고에 관한 논의를 할 때는 그 차지하는 비중이나 관련성이 상대적으로 작다. 사고력의 7범주 중 기호적 사고와 상징적 사고의 2개 범주를 뺀 나머지 5개 범주는 보다 보편적이고 일반적인 수준의 사고 능력이라 할 수 있을 것이다(김영정, 2004).

② 논리적 사고(Logical thinking)

논리적 사고란 대상이나 사건들의 인과적 관계에 의하여 합리적인 결론을 추론하는 능력이다(손은경, 1995; Gopnik & Sobel, 2000).

논리적 사고의 개념은 그 적용 범위가 하나로 확정되어 있는 것은 아니다. 표준적 의미 이회에도 넓은 의미와 좁은 의미의 개념들이 존재한다는 것이다. 사실 영어의 critical thinking(비판적 사고)와 가장 가까운 우리말 용어를 들어보라면, 아마도 논리적 사고(또는 합리적 사고)가 될 것이다. 즉 일반적으로 비판적 사고와 논리적 사고는 동일한 개념을 지칭하고 혼용되어 사용되기도 한다. 하지만 논리적 사고는 주로 형식논리/수리논리적인 뉘앙스가 강한 반면, 비판적 사고는 비형식논리학을 기반으로 한 일상언어/비형식논리적인 뉘앙스가 강한 용어이다. 즉, 논리적 사고는 타당성(validity)을 그 준거로 가지는 사고인

반면, 비판적 사고는 건전성(soundness)을 그 준거로 가지는 사고라고 구분될 수도 있을 것이다. 이것은 논리적 사고는 구체적인 상황이나 맥락을 가급적 배제하는 추상화의 길을 지향해온 반면, 비판적 사고는 판단이나 추리에 있어 상황이나 맥락의 중요성을 강조하고 상황이나 맥락을 적극적으로 고려하는 구체화의 길을 지향해온 것과도 통한다. 그러나 논리적 사고와 비판적 사고 모두 수리논리적 측면이나 언어논리적 측면 어느 한 측면에 대한 강조를 넘어서서 양자 모두 통합된 적용력을 가진 사고 유형에 대한 연구와 훈련을 궁극적으로 추구하고 있다는 점에서 그 최종 지향점은 같다고 봐야 할 것이다. 따라서 비판적 사고와 논리적 사고를 혼용하여 사용하는 데에는 큰 무리가 없으며, 두 사고의 핵심은 모두 수렴적 사고(convergent thinking) 또는 수직적 사고(vertical thinking)에 있다고 본다. 단, 가장 좁은 의미의 논리적 사고는 기호적 사고와 더불어 분석적 사고와 추론적 사고가 포함되며, 표준적 의미의 논리적 사고는 여기에 종합적 사고가 더 추가되며, 넓은 의미의 논리적 사고는 여기에 대안적 사고가 더 추가된다. 아무리 논리적 사고의 개념을 넓게 잡아도 여기에는 발산적 사고나 상징적 사고의 개념은 포괄되지 않는다(김영정, 2004).

③ 추상적 사고(Abstract thinking)

추상적 사고를 단어 자체로만 볼 때, 단어들 간의 관계에 의해 의미가 표상되는 의미망 체계(Minsky & Minsky, 1968)에서의 인간의 인식을 의미하는 것처럼 보이지만 '요약'이라는 의미를 강조할 필요가 있다.

Trope과 Liberman(2003)은 "추상화의 차원(dimension)이라는 것은 스펙트럼의 구체적 부분에서의 낮은 수준의 지역적 사고에서 추상적 부분에서의 높은 수준의 전체적 사고에 의해 특징지어진 연속체"라고 정의하였다. 예를 들어, 추상적 사고는 '숲'에 대한 생각인 반면, 구체적 사고는 '나무'에 관한 생각이라고 제시하였다. Smith와 Trope(2006)는 "추상적 사고란 자극을 그보다 높은 차원으로 분류할 수 있을 뿐 아니라, 자극의 주요 측면에 초점을 두고 패턴과 핵심을 추출하기 위한 구조와 패턴을 감지하는 사고"라고 정의하였다.

본 연구에서는 추상적 사고를 비판적 사고의 기반이 되는 개념으로 보고 있다. 그렇다고 단순 기억 능력이나 상기(recall) 능력과 같은 저차원적 사고 능력 보다는 상대적으로 높은 수준의 사고를 요구한다. 예를 들어 [그림 Ⅱ-6]을

보며 사람임을 인식하는 과정은 그다지 높은 수준의 사고를 요구하지 않는다. 하지만 추상화의 수준이 점점 높아질수록 7개 사고들을 자극할 수 있다.

예를 들어 프로그래밍 코드는 추상적 사고를 기반으로 하여 분석적 사고, 기호적 사고를 요구하며(형식적 추상화 측면), 피카소의 그림은 추상적 사고를 기반으로 하여 발산적 사고, 상징적 사고를 요구하게 된다(비형식적 추상화 측면). 즉 추상화의 단계가 높아질수록 그보다 높은 고차적 사고력을 요구하게 된다는 의미이다.

④ 재귀적 사고(Recursive thinking)

인지공학의 관점에서, 재귀적 사고란 '자기 자신 또는 타인의 사고에 관한 사고'를 의미한다(Kurdek, 1977; Landry & Lyons-Ruth, 1980; Oppenheimer, 1978). 예를 들자면 '나는 아빠를 생각하는 엄마를 생각한다'와 같은 사고가 재귀적 사고라 할 수 있다.

유중현과 김종혜는 "재귀적 사고란 인간이 영아기 때 사용하는 원초적인 사고과정에서 시작하여 인간이 커가면서 논리적 사고가 접목되어 논리를 바탕으로 하여 조건이 만들어지고 조건을 변경하면서 문제가 해결될 때까지 반복적으로 사고하는 과정"이라 정의하였다.

Wing(2006)은 재귀적 사고란 병렬적 처리를 의미한다고 하였다. 코드를 데이터로, 데이터를 코드로 해석하는 것, 하나의 문제의 장·단점을 인식하는 것, 어떤 이에게 또는 사물에게 하나 이상의 이름을 부여하는 것이 그 예가 될 수 있다고 하였다. 프로그램의 정확성과 효율성만을 평가하는 것이 아니라 간결함과 단순성을 고려한 심미적 측면에서의 시스템 설계도 재귀적 사고라 하였다. 이러한 정의가 재귀적 사고의 전체를 정의한다는 것보다는, 컴퓨터과학에서의 재귀적 사고의 정의라고 할 수 있겠다.

이러한 정의들이 어떠한 영역에서의 재귀적 사고를 의미하든, 일반적으로 재귀적 사고는 논리적 사고, 종합적 사고를 기반으로 한다는 점에는 변함이 없다. 더 넓게 생각해서는 Wing(2006)의 정의처럼 '하나의 사물이나 사람에게 하나 이상의 이름을 부여'하는 데에 필요한 대안적 사고가 필요하다.

본 연구에서 재귀적 사고와 알고리즘적 사고를 김영정(2004)이 정의한 7개의 고차적 사고의 범주와는 다른 차원의 사고 양식(패러다임)으로 보고 있다. 예

를 들어 논리적 사고력이 높은 사람도 재귀적 사고를 전혀 접해보지 않았다면 논리적 사고력이 상대적으로 낮은 사람과 그 수준이 별다를 게 없기 때문이다. 물론 논리적 사고력의 기반 위에서 재귀적 사고가 일어나기 때문에 이들의 상관관계는 높으나, 사고의 양식은 다르다.

⑤ 알고리즘적 사고(Algorithmic thinking)

알고리즘적 사고는 어떠한 문제를 해결하기 위한 문제해결 절차를 만들어내는 능력이다(유중현, 김종혜, 2010; Jenkins, 2002). Futschek(2006)은 알고리즘적 사고와 창의성이 매우 강한 상관관계를 갖는 능력이라고 정의하며 알고리즘적 사고를 문제를 해결하기 위해 특정한 해결 방법을 습득하는 사고양식이라고 하였다. 유중현과 김종혜(2010)는 문제를 해결하기 위해서는 직관적 사고가 필요하며, 직관적 사고에 의해 나온 해결책들을 통해 결과를 예측해 보는 추론적·논리적 사고가 필요하며 각각의 해결책들의 장단점을 파악하는 비판적 사고가 이미 포함되어 있다고 말하고 있다. 알고리즘적 사고에 의해 도출된 결과가 항상 논리적으로 옳고 최적의 알고리즘이라는 것은 아니며 문제를 한 번에 해결하는 '통찰'을 의미하는 것도 아니다. 하나의 문제를 해결하는데 필요한 최적의 계산을 논리적 사고의 기반 위에서 하려는 사고 과정이라는 의미이다.

그러나 알고리즘적 사고의 양식은 재귀적 사고처럼 추상적 사고나 비판적 또는 논리적 사고와는 독립적이라고 할 수 있다. 즉 논리적 사고와 상관은 있지만 이와는 다른 사고과정의 패턴을 요구하는 능력이다. 이러한 관계는 논리적 사고력만을 신장시킨다고 알고리즘적 사고가 신장되는 것은 아니지만, 알고리즘적 사고를 촉진시키기 위한 학습이 논리적 사고를 신장시킬 수는 있을 것이라는 경험적 측면에서 이해해 볼 수도 있다. 계산적 사고력의 계층적인 측면을 볼 때 알고리즘적 사고는 논리적 사고의 기반 위에서 수행되며, 넓은 의미의 알고리즘적 사고는 발산적 사고 또한 필요로 한다.

재귀적 사고 자체를 알고리즘적 사고라고 할 수는 있으나 '모든 재귀적 사고는 알고리즘적 사고인가?'에 대한 물음에서 넓은 의미의 재귀적 사고를 고려해 볼 때에는 알고리즘적 사고의 일부라고 할 수 있을 것이다. 이러한 이유로 이 두 개의 사고를 완전한 포함관계로 표현하지 않고 구분 지을 필요가 있다고

판단된다.

⑥ 창의적 사고(창의성)과의 관계

비판적 사고는 창의적 사고의 핵심인 발산적 사고(divergent thinking) 또는 수평적 사고(측면적 사고, lateral thinking)를 포함하고 있지 않다. Guilford(1956)는 발산적 사고의 범주에 유창성(fluency), 융통성(flexibility), 독창성(originality), 정교성(elaborativeness) 등의 요소가 포함된다고 했다. Torrance(1974)는 이 발산적 사고의 네 요소에 결점과 문제에 대한 민감성(sensitivity) 및 재정의 하기(redefinition)와 같은 능력을 추가하기도 하였다. 그는 창의적 사고와 발산적 사고가 같은 것이 아니라고 결론 내린다. 그것은 창의적 사고에는 발산적 사고에는 없는 민감성과 재정의 능력이 포함된다고 믿었기 때문이다. 즉 비판적 사고와 창의성(창의적 사고)의 차이의 핵심 발산적 사고의 포함 여부에 있다고 할 수 있다(김영정, 2004).

창의성에 대한 구체적인 논의는 이후에 계속되며, 여기에서는 [그림 Ⅱ-6]에서 창의성이 계층과 범주를 살펴볼 필요가 있다. 창의성은 추상적 사고를 기반으로 할 수 있으며, 논리적 사고력이 창의성보다는 수리적인 반면, 창의성은 예술적인 성향을 가지고 있다. 프로그래밍 자체가 하나의 예술임은 오래전부터 주장되어 오기도 하였다(Pattis, 1981; Springer & Friedman, 1990).

4) 추상화와 자동화

Wing(2008)은 계산적 사고의 핵심적인 두 가지 원리를 추상화와 자동화로 제시하고 있다.

(1) 추상화

계산적 사고에 대한 개념의 본질은 추상화에 있다(Wing, 2008). 컴퓨터과학에서 추상화에 대한 중요성에 대한 언급은 계속되어 왔다(Devlin, 2003; Hazzan, 1999; Kramer, 2007). 추상화라는 의미는 크게 2가지 측면의 의미를 갖는다.

- '무엇을 단순화하고 관심을 집중시키기 위해 세부적인 것을 제거하는 과정'이며,
- '그것들의 공통적인 핵심과 본질을 찾기 위해 이를 일반화하는 과정'이다.

[그림 Ⅱ-7] Henri Matisse의 그림(좌), Harry Beck의 런던 지하도(우) (Kramer, 2007)

추상화는 미술이나 음악과 같이 컴퓨터과학 외의 다른 학문에서도 널리 사용되는 개념이다. Henri Matisse의 [그림 Ⅱ-7]에서는 간결한 선들과 조각 그림만으로도 여성의 누드화임을 알 수 있게 해줄 만큼 추상화가 잘 된 작품이다. 째즈 뮤지션은 "간단한 음악을 복잡하게 만드는 것은 쉬운 반면, 복잡한 음악을 간단한 음악으로 만드는 것은 더 어렵다"라고 추상화의 어려움을 토로한다. Harry Beck의 1928과 1933년도 제작인 런던 지하도 그림도 추상화의 개념을 이해할 수 있는 좋은 예이다. 1933년도의 지하도 그림은 그 전의 지하도 그림에 있던 곡선들을 수평선, 수직선, 대각선으로만 표현하였다. 이러한 도식적인 표현은 사람들에게 여러 방면으로 많은 편리함을 주기도 하지만 실제상의 거리나 위치에 대한 정보는 손실한 듯 보인다. 이는 추상화의 특징이며 추상화 단계에서 주의해야 할 부분이다. 과도한 추상화는 불충분한 정보를 가지게 되며 그렇다고 너무 자세해진다면 혼란스럽고 이해하기가 힘들어진다는 것이다. 즉, 추상화의 수준, 추상화로 인한 이점 등은 모두 해당 추상화의 목적에 달려있다는 의미이다(Kramer, 2007).

추상화는 컴퓨터과학 전반에 걸쳐 가장 핵심이라고 할 수 있다(Bennedssen

& Caspersen, 2008; Koppelman & van Dijk, 2010; Kramer, 2007). 추상화의 기술은 인간이 이미 습득한 능력이며 많은 영역에서 활용되고 있다. 인간은 태어나서 성장과정에서 수 세기, 색 이름의 사용, 사람과의 관계에 대한 명칭 사용 등에서 보편적인 추상화에 대한 반복적인 연습을 하게 된다. 수학 분야에서 이러한 추상화에 대한 연구와 학습이 많이 이루어지기는 하지만 그 영역은 수학적 사고에서 크게 벗어나지 못한다. 컴퓨터과학 분야에서 추상화의 범위는 하드웨어, 소프트웨어, 또는 추상적 개념뿐 아니라 문제 상황 자체, 그리고 그것의 해결과정 등의 영역에서 이루어지며 이 과정에서의 추상적 개념의 추상화, 그리고 이것들의 반복으로 인해 생기는 추상화 레벨(level)에 대한 이해도 필요하게 된다.

(2) 자동화

Wing(2008)은 추상화를 정신적 도구(mental tool)라고 한다면 이러한 추상화의 기능은 기계적 도구(metal tool)인 자동화를 통해 더 증폭되어진다고 말한다. 자동화란 추상개념과 추상 레이어, 이들 사이의 관계를 기계화하여 작동시킨다. 즉 이해하기 복잡하고 까다로운 추상개념과 추상레이어를 정교하게 해석할 수 있도록 기계적으로 자동화할 수 있다는 것은 컴퓨터 발전 역사의 핵심이라고 할 수 있다. 컴퓨팅 머신의 빠른 처리 능력, 저장 공간, 통신 능력이 있었기에 이러한 발전이 가능했다. 하지만 인간이 정보를 처리하고 계산을 한다는 관점에서 컴퓨터란 컴퓨팅 머신만이 아닌 인간을 지칭하기도 함을 알아야 한다. 다른 말로는 계산적 사고가 컴퓨팅 머신을 반드시 필요로 하는 것은 아니라는 의미이다. 또한 컴퓨터의 처리 능력과 함께 인간의 정보 처리능력을 결합한 새로운 영역의 능력을 개발할 수도 있을 것이다. 예를 들자면 아직까지도 어떠한 이미지를 해석하고 분석하는 능력은 인간이 컴퓨팅 머신보다 훨씬 나으며, 방대한 정보를 관리하고 다루는 데에는 컴퓨팅 머신이 훨씬 빠르고 정확하다고 할 수 있겠다.

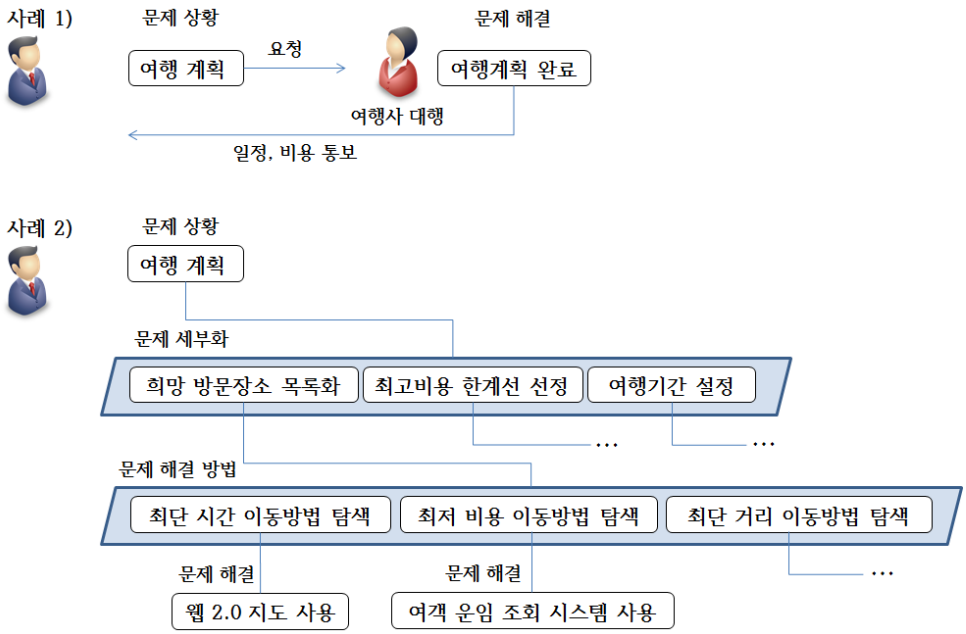
어떠한 문제를 해결하는 데에 우리가 먼저 고민해야 할 일은 추상개념을 어떻게 정의하고 계산의 대행자를 컴퓨팅 머신으로 해야할지 또는 인간이 해야할지, 아니면 이들의 조합으로 할지를 결정해야 한다는 것이다.

5) 계산적 사고의 사례

(1) 일상생활에서의 계산적 사고

우리의 일상 생활에서 추상화와 자동화를 이용한 계산적 사고의 사례에는 어떤 것이 있을까? 유럽으로 배낭여행을 가기 위해 여행 계획을 세운다고 생각해보자. 어떤 이는 여행사에게 자신의 일정과 경비를 모두 맡기기도 하며 어떤 이는 여행 일정 모두를 스스로 계획하기 위해 문제를 여러 단계로 쪼개어 해결하려고 할 것이다.

전자는 문제의 상황을 큰 덩어리로 보고(추상화) 이를 바로 여행사로부터 해답을 찾고자 하였으며(자동화), 후자는 문제를 더 작은 덩어리로 쪼개어 해결하려고 하고 있다(추상화).



[그림 Ⅱ-8] 계산적 사고의 사례

후자의 경우를 좀 더 세밀히 살펴보자. 여행자의 입장에 따라 배낭여행 계획 세우기라는 문제가 쪼개어지는 기준은 다양할 수 있다. 최대 지출 경비에 맞추어 일정을 계획할 수도 있으며 희망 방문장소를 우선적으로 선정하여 이에 맞추어 일정을 계획할 수도 있을 것이다. 만약 희망 방문장소를 목록화하여 이를 여행기간 내에 방문할 것을 목표로 한다면 이 문제는 다시 최단 시간 이동, 최

저 비용 이동, 최단 거리로 이동 방법 등으로 해결 방법을 탐색할 수 있을 것이다. 현대 사회에서 이러한 방법들 중 어떤 방법을 선택하더라도 우리는 인터넷을 이용하는 것이 가장 현명한 전략임을 알고 있다. 인터넷을 통해 제공되는 웹 지도, 여객 운임 조회/예약 시스템은 이러한 이동방법에 대한 답을 제공해 줄 수 있다. 사실 이러한 프로그램들은 인간에 의해 개발되었고 해당 문제에 대한 계산을 잘 정의하여 처리하고 있기 때문에 제대로 작동하고 있는 것이다. 하지만 일반 사용자들은 그 내부의 어떠한 복잡한 계산에 대해 알아야 할 필요는 없다. 이렇게 컴퓨팅 머신을 사용하지 않고 여행자는 직접 수많은 여객사를 방문하거나 전화하여 다양한 정보를 알아 볼 수도 있는 일이겠지만 매우 반복적이고 기계적인 일이다. 이는 많은 시간과 노력이 필요한 일이기에 자동화된 컴퓨팅 머신이 대신 계산하여 준다면 이를 이용하지 않을 이유가 없다. 이렇게 일상 생활의 문제 해결의 과정에서 휴먼 컴퓨터의 능력과 컴퓨팅 머신의 능력의 조합이 매우 훌륭한 역할을 해 나가고 있음을 많은 사람들이 경험적으로 알고 있다.

이런 과정으로 문제가 해결되어 여행계획이 완료되었다고 가정해보자. 사실 이러한 일련의 과정은 여행사 직원이 하는 일 자체이다. 즉 전자의 예에서 여행사에서는 후자의 예에서 여행자가 직접 했던 문제 해결 과정을 자동화하여 일정을 계획하는 프로세스, 즉 정해진 계산에 의해 작업을 했던 것이다. 이 문제를 해결해 나가는 과정에서 계산이란 컴퓨팅 머신으로 최저 비용, 최단 시간 또는 최단 거리 탐색 알고리즘에 의한 계산뿐 아니라 여행자의 요구를 분석하고 희망 방문 장소의 가중치를 정하고 유한한 자원을 가장 효율적으로 이용해야 하는 조건 안에서 해답을 이끌어내는 계산까지를 포함할 수 있다. 물론 그러한 계산 모두를 정형화하여 컴퓨팅 머신이 대신하면 더욱 효율적일 수 있으나 위 문제에서는 인간의 심리와 관련된 다양한 변수들의 존재로 현재까지는 휴먼 컴퓨터와의 조합이 문제를 해결하는 것이 합리적이라고 생각된다.

전자의 예에서는 여행자의 입장에서 여행사(여행사 직원)가 컴퓨팅 주체가 되어 자동화의 기능을 해 주었다고 볼 수 있으며, 후자의 예에서는 전자의 예에 비해 추상화 레이어 사용이 많았으며 컴퓨팅 머신이 자동화의 기능을 수행해 주었다고 볼 수 있겠다. 이처럼 하나의 문제에서 어떤 모델을 이용할 것인

지, 문제를 바라보는 관점에 따라 추상화, 추상 레이어, 자동화의 사용은 확연히 달라질 수 있다.

위의 예에서도 볼 수 있듯이 우리는 문제를 효율적으로 해결하기 위해서 자동화된 컴퓨터를 이용하게 된다. 이 때 중요한 것은 어떠한 컴퓨터에 어떠한 문제를 주어 자동화를 통해 문제를 쉽게 해결하느냐는 것이다. 하나의 문제를 어떻게 추상화시켜 어떠한 컴퓨터에 던져 주느냐가 문제를 해결해 나가는 핵심인 것이다. 즉 추상화가 문제 해결을 위한 계산적 사고의 키(key)인 것이다 (Bennedssen & Caspersen, 2008; Hazzan, 1999; Koppelman & van Dijk, 2010; Kramer, 2007).

이와 같이 우리의 일상생활에서 계산적 사고가 필요한 문제들은 쉽게 찾아볼 수 있으며 우리는 각자 해당 상황에서 알맞은 해법을 찾기 위한 계산을 자연스럽게 하고 있다.

- 가나다 순의 명단표에서 이름 찾기
  - 선형탐색: 앞에서부터 찾기
  - 이진탐색: 중간 지점에서부터 찾기
- 슈퍼마켓 계산대 또는 공항 검색대 줄서기
  - 작업 스케쥴링 분석
- 최소의 시간과 움직임으로 집 청소하기
  - 병렬 처리(빨래를 돌려 놓고 방 청소하기), 최단 경로 구하기
- 책장 또는 냉장고 정리하기
  - 가상 메모리 기술(자주 사용하는 것은 가까이에 두고 공간이 부족할 때에는 무엇을 버릴지 나름대로의 기준을 정하여 결정함)
- 심부름 하기
  - 스케줄링 기법(데드라인이 빠른 것부터 / 현재 하는 일과 관련된 일부터 등)
- 잃어버린 물건 찾기
  - 백트래킹(왔던 길을 순서대로 되돌아가며/회상하며 물건을 찾기)

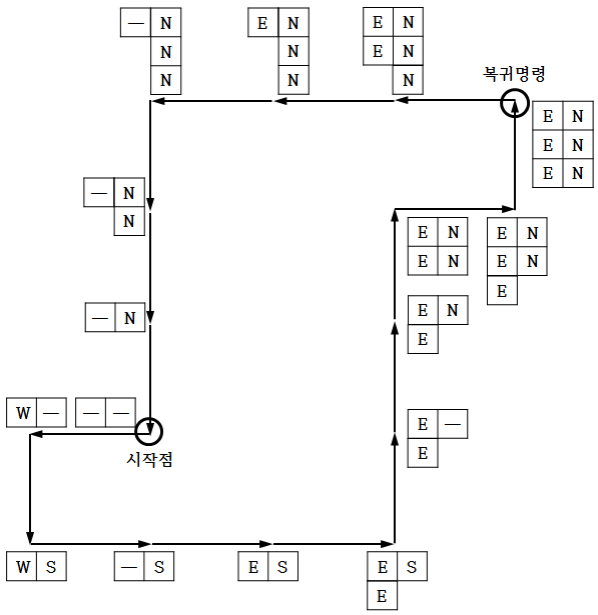
위의 일상의 예제들에서 계산의 주체는 인간이다. 컴퓨터 과학에서 사용되는 많은 문제 해결 전략이 인간의 생활에서도 사용되고 있다. 이는 달리 생각해보

면 컴퓨터 과학의 문제 해결 전략을 인간의 문제 해결 과정에서 찾고자 했던 것이라고 생각해볼 수 있을 것이다.

(2) 인공물에서의 계산적 사고

앞서 계산적 사고는 수학적사고뿐 아니라 과학적, 공학적 사고를 함께 필요로 하는 추상화라고 했으며 추상화 레벨이 존재한다고 하였다. 이러한 개념들을 쉽게 이해할 수 있는 간단하고 좋은 예를 Jonh-Laird의 저서에서 찾아볼 수 있다.

John-Laird(1988)는 박테리아처럼 자신의 세계를 탐험하는 로봇을 [그림 Ⅱ-9]와 같이 상상하며 다음과 같은 시스템을 제안하였다.



[그림 Ⅱ-9] John-Laird의 로봇 경로 프로그램

① 로봇은 앞이나 뒤로만 움직일 수 있으며 로봇의 경로는 처음 시작점에서 항상 끝을 내게 된다. 이를 위해서는 앞으로 걷는 발걸음 수와 뒤로 걷는 발걸음 수가 같아야 할 것이다.

② 이러한 로봇의 기능을 위해서는 로봇 자체가 자신의 발걸음을 셈하는 프로그램을 장착시킬 필요가 있다. 이러한 기억력을 위해 로봇은 가장 단순한 형태 중 하나인 스택(stack) 구조를 사용하기로 한다.

③ 로봇이 앞으로 한걸음 나가는 것을 <forward>라는 상징으로, 뒤로 한걸음 가는 것을 <backward>라는 상징으로 표시할 것이다.

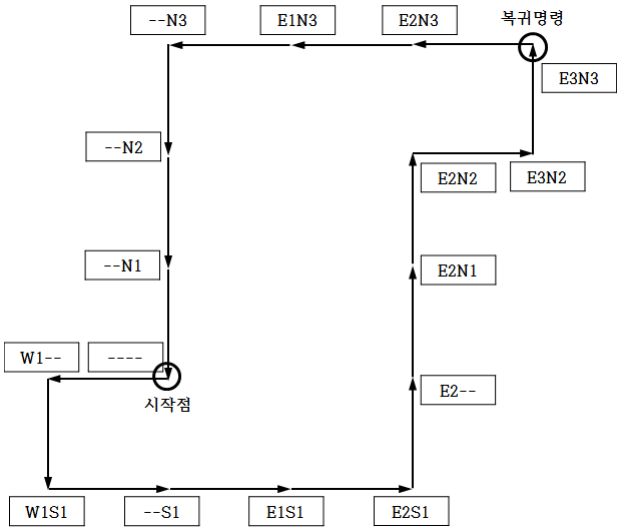
④ 로봇의 프로그램은 스택이 비어있는 상태에서 출발한다.

⑤ 일반적으로 로봇이 한 발걸음 떼어 놓을 때 그 프로그램은 그 발걸음에 대응하는 상징을 스택의 맨 위에 놓는다.

⑥ 하나의 예외가 있다면 한 방향에서 한걸음 걸은 것은 반대방향으로의 한 걸음에 의해 삭제된다. 따라서 그러한 연속절차가 일어날 때마다 스택의 맨 위에 있는 상징은 제거된다. 예를 들어 만약에 로봇이 뒷걸음을 치고, 더미의 맨 위에 <forward>라는 상징이 있다면 프로그램은 그것을 제거할 것이다.

이러한 규칙에 의해 로봇은 그것의 스택이 처음처럼 비어 있을 때 시작점으로 돌아오게 되는 시스템 환경을 갖추게 된다.

만약 이러한 방법으로 동-서, 남-북의 두 차원에서 로봇 경로 프로그램을 만들기 위해서는 두 개의 스택이 필요할 것이다. 하나의 스택은 북쪽과 남쪽을 향한 발걸음을 기록하고 다른 하나는 동쪽과 서쪽을 기록한다. 로봇 경로에서의 임의의 한 지점에 있어서 시작점으로부터의 위치는 로봇이 해당 지점을 지날 때 스택에 저장된 내용으로 알 수 있다.



[그림 Ⅱ-10] 수정된 로봇 경로 프로그램

위에서 언급했던 것처럼 John-Laird는 로봇 경로 프로그램을 만들며 6가지 정도의 조건과 규칙을 논리적으로 정리하였다. 이러한 추상화의 단계에서 우리는 수학적 사고뿐 아니라 과학적, 공학적 사고가 필요함을 쉽게 이해할 수 있을 것이다. 이 예에서 사용되는 두 개의 스택을 이용한 추상개념은 이 문제를 해결하기 위한 유일한 해는 아니다. 그리고 스택이라는 개념을 사용하지 않더라도 해결할 수 있다. 다음의 [그림 Ⅱ-10]처럼 하나의 상징만을 이용한 해결 방법도 있을 것이다.

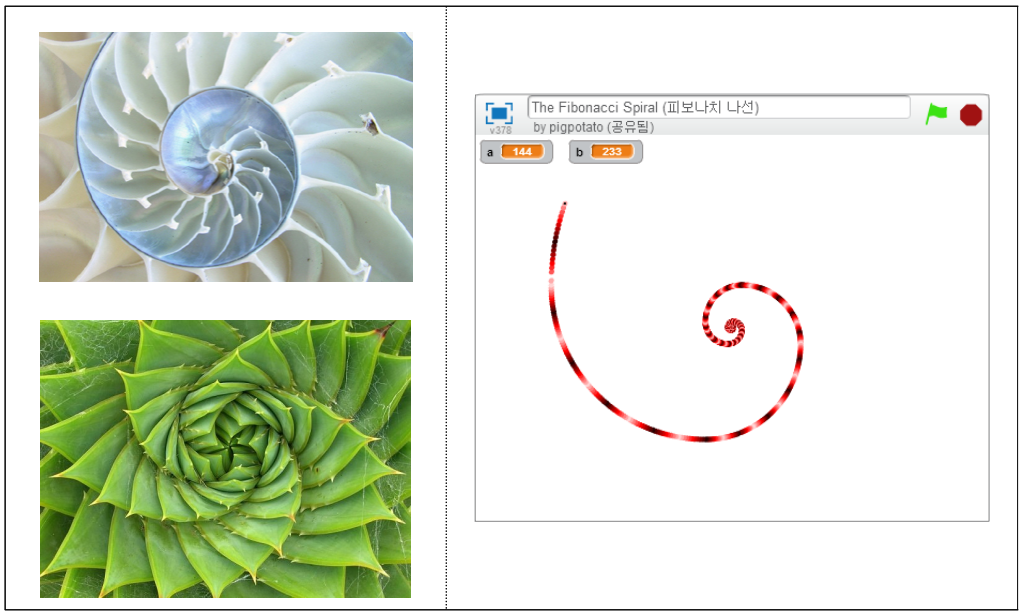
[그림 Ⅱ-10]에서처럼 [E1N3]이라는 상징은 시작점에서부터 동쪽으로 한 걸음, 북쪽으로 세 걸음 떨어진 위치에 로봇이 위치하고 있다는 의미이다. 그리고 또한 시작점을 돌아가기 위해서는 서쪽으로 한 걸음, 남쪽으로 세 걸음 더 가야 한다는 의미이기도 하다.

만약 위의 로봇 시스템 자체 내에서 실제 두 개의 스택이 사용되지만 [E1N3]라는 상징만으로도 사용자에게 전달된다면 사용자는 스택이라는 용어 자체를 알지 못하더라도 이러한 상징이 무엇을 뜻하는지 알 수 있다. 이렇게 개념의 분할로 인한 편이성이 추상 레이어가 주는 이점이라고 할 수 있다. 하나의 추상 레이어 안에서의 추상 개념들은 동일한 계산을 통해 그보다 상위, 또는 하위의 추상 레이어의 추상 개념으로 변환 되어질 수 있다.

잘 정의된 추상 레이어의 사용은 복잡하고 거대한 시스템을 설계하고 구축할 수 있도록 해준다(Wing, 2008).

(3) 자연에서의 계산적 사고

컴퓨터 과학에서는 자연의 많은 현상을 시뮬레이션으로 구현하고자 하는 시도가 계속되고 있다. 컴퓨팅 머신으로 자연의 현상을 구현하고자 한다는 것은 자연의 어떠한 상태의 변화들 안에서 존재하는 계산을 정형화하여 표현하려는 시도라 할 수 있겠다. 자연속에서도 이러한 계산이 존재함을 많은 연구를 통해 알려졌고 컴퓨팅 머신의 계산 능력이 이러한 발견을 돕기도 하였다. 가장 쉬운 예로는 조개의 나선형, 식물의 가지 분리, 자연에서의 꽃잎 수, 꽃씨의 배열, 꿀벌의 가계도 등의 자연현상에서 피보나치 수열과 관련된 계산이 발견되었다(김현진, 2008; 박숙, 2000; Koshy, 2001).



[그림 Ⅱ-11] 자연 속 계산(피보나치 수열)의 존재[9]와 계산 구현[10]의 예

6) 계산적 사고력 검사 도구

계산적 사고력이라는 용어가 사용된 이래 계산적 사고가 무엇인지에 대한 논의가 계속되었지만 이것을 어떻게 측정하고 평가할 수 있을 것인가에 대한 연구는 미흡했고 최근에 들어서야 몇몇 연구들이 시작되었다.

계산적 사고의 평가 도구 개발에 대한 국내·외의 연구는 <표 Ⅱ-5>와 같다.

<표 Ⅱ-5> 계산적 사고력 측정 도구 개발 관련 연구

연구자	연구명	측정 도구	검사 대상
이은경 (2009)	Computational Thinking 능력 향상을 위한 로봇 프로그래밍 교수 학습 모형	자체개발 (검사지)	K 대학 및 S 대학의 컴퓨터과학 관련 전공 1학년 학생 (실험집단 18명, 비교집단 25명)
김종혜 (2009)	계산적 사고 기반의 문제 해결 능력 향상을 위한 중등 교육 프로그램	자체개발 (검사지)	(적용해보지 않음)
서성원 (2010)	TPL과 VPL을 활용한 로봇 프로그래밍 교육이 계산적 사고력에 미치는 영향	이은경(2009)의 평가 도구 사용	M 고등학교 자연계열 2학년 2개 학급 70명 (실험집단 35명, 비교집단 35명)

9) 출처: http://thinkingwritingcreating.edublogs.org
10) 출처: http://scratch.mit.edu/projects/11712050/

연구자	연구명	측정 도구	검사 대상
Koh, et al. (2010)	Towards the automatic recognition of computational thinking for adaptive visual language learning.	자체개발 (Computational Thinking Pattern Graph)	제작된 프로그램 자체의 코드를 분석하여 계산적 사고 패턴을 추출하여 그래프로 보여줌.
박지연 (2012)	Computational Thinking 능력과 유아놀이와의 연관성 분석	자체개발 (검사지)	(적용해보지 않음)
Gouws et al. (2013)	First year student performance in a test for computational thinking.	Computer Olympiad의 Talent Search 테스트	Rhodes 대학의 컴퓨터과학 학부생 83명

이은경(2009)의 연구에서 개발된 검사 도구는 PISA의 문제를 바탕으로 만들어진 문항들로 계산적 사고의 구성요인에 대해서만 분류하였고 문항 내용에 대한 분류나 분석이 없다. 또한 문제해결 과정의 각 영역에 사용되는 계산적 사고의 구성요인을 제시하였지만 이를 신뢰할만한 근거는 없다. 예를 들자면 문제 발견에는 논리적/분석적 사고만이 필요하며, 문제 해결 전략에는 동시적/선행적/전략적/절차적/재귀적 사고만이 필요하다는 근거는 제시되고 있지 못하다.

서성원(2010)의 연구에서 개발된 검사 도구는 이은경(2009)의 연구에서 개발된 평가 도구를 수정 · 보완하여 사용하였기에 검사의 총점으로 평가를 하였다. 즉, 검사 도구에 대한 분석적 접근이 이루어지지 않았다.

김종혜(2009)의 연구에서 개발된 검사 도구는 계산적 사고 기반의 문제 해결 능력의 단계를 나누고, 주제를 '도서관, 수에 의한 디자인, 교육과정, 환승체계, 에너지, 영화감상, 관개, 전화통화, 냉동고, 휴가'로 제시하였다. 이러한 평가 문항의 주제는 컴퓨터과학의 핵심 주제에 어떠한 관계를 갖고 있는지에 대해 제시를 못하고 있다.

본 연구에서는 기존의 관련 연구를 분석하여 계산적 사고력을 측정하기 위한 검사 도구의 개발을 연구의 범위에 포함하고자 하였다. 기존의 검사 도구와

비교하였을 때 본 연구에서 개발한 검사도구가 갖고 있는 특징은 크게 두 부분으로 나눌 수 있다.

첫째, 기존 연구에서 가장 미흡했던 부분은 문항 자체의 내용(주제)에 대한 선정 기준이었다. 본 연구에서는 프로그래밍을 중심으로 한 컴퓨터과학의 핵심 주제와 아이디어에 대한 관련 연구(Schwill, 1994, 1997; Zendler A & Klaudt, 2012; Zendler & Spannagel, 2008; Zendler, Spannagel & Klaudt, 2008)와 소프트웨어의 인지복잡도의 측정(Gu & Tong, 2004; Minsky & Minsky, 1968; Misra, 2006; Shao & Wang, 2003)에서 주제를 도출하였다. 도출된 10가지의 주제는 순차구조, 조건분기, 반복, 동시성(병렬처리), 변수, 난수, 알고리즘, 객체, 함수, 재귀이다. 이러한 주제를 실생활 문제와 연계하여 특정한 지식이 없이도 그 개념이나 아이디어에 대한 문제를 해결할 수 있도록 문항을 제작하였다

둘째, 검사 문항의 수준을 기존 검사 도구의 대상자보다 어린 학생들에게 투입할 수 있도록 낮추었다. 이를 위해 문항의 설명을 줄이고 직관적으로 문제를 해결할 수 있도록 그림을 많이 이용하였으며 예시 설명을 추가하기도 하였다. 이는 문장제 문제가 언어형 문제의 이해능력에 따라 문제해결 과정에 끼치는 부정적인 영향을 최소한으로 줄이기 위함이다(김선희, 2012).

2. 창의성

‘창의성’이라는 용어는 다양한 의미를 가지며 처음 심리학 영역에서 논쟁적으로 토론이 이루어졌다. 창의성에 대한 연구는 창의성에 대한 정의, 평가 및 증진에 관해 다양한 관점에 초점을 맞추고 있다. 또한, 창의적 인간, 창의적 과정, 환경 요소의 영향, 창의적 생산물이 무엇인가에 대한 탐구가 관련 연구에서 계속되고 있다(Runco, 2007). 본 연구에서도 창의성에 대한 정의가 필요하며 창의성의 실체를 알기 위해 그 구성 요인들을 파악하고 검사를 위해 활용되어질 필요가 있다고 본다.

1) 창의성의 정의와 구성 요인

창의성에 대한 정의는 학자마다 조금씩 다르다. 창의성과 관련된 기존의 연구들을 통해 창의성에 대한 정의를 살펴볼 필요가 있다.

<표 Ⅱ-6> 창의성 관련 연구

연구자 (발행연도)	연 구 내 용	
Torrance (1957)	제목	Psychology of survival
	정의	창의성이란 문제, 지식의 부족, 부족한 요소들, 부조화 등에 대한 민감성, 문제와 장애를 규명하는 것, 문제해결책을 찾는 것, 문제나 해결되어야 할 결점에 대한 추측, 가설을 형성하고 이러한 가설을 검증하고 수정, 재검토하여 결과를 얻어내는 것이다.
Guilford (1967)	제목	The nature of human intelligence
	정의	창의성이란 "인간의 지적, 정의적 요인을 모두 포함하고 있으며, 지적요인은 지능 검사로 측정되기 어려우나 지능의 한 중요한 측면이면서 인간의 보편적인 잠재력이고, 창의적 요인은 창의적 행동을 발휘하는 개인의 인성적, 기질적 특성이며, 학교 교육에서 훈련을 통해 개발할 수 있는 것"이라고 하면서 인간의 지적 능력 중 창의성의 분명한 지침이 되는 사고 유형을 확산적 사고(divergent thinking)라고 하였으며, 이것을 측정함으로써 창의성을 측정할 수 있다고 하였다.
Runco (1989)	제목	The creativity of children's art
	정의	창의성은 문제의 정의 또는 발견, 그리고/또는 그것의 해결책으로 묘사된다. 창의성이란 이 해결책이 다른 가능한 해결책과 비교하였을 때 확실히 확산적이거나 독특해야 한다
김영채 (1999)	제목	창의적 문제해결: 창의력의 이론, 개발과 수업
	정의	창의성을 협의, 광의, 과정으로서의 창의성으로 나누어 정의하였다. 협의의 창의성은 확산적 사고로 어떤 문제에 대한 반응의 수가 많고, 다양하고, 독특한 것일수록 창의적으로 보았다. 광의의 창의성은 새롭고 유용한 어떤 것을 생산해 내는 행동 또는 정신과정을 창의성이라고 하였다. 과정으로서의 창의성은 기존의 정보들을 특정한 요구조건에 맞거나 유용하도록 새롭게 조합시킨 것이라고 보았다.
김혜숙 (1999)	제목	창의성 진단 측정도구의 개발 및 타당화
	정의	창의성이란 "새롭고 가치있는 유용한 것을 만들어 내는 능력(힘)으로서 이는 개인의 정의적 성향과 인지적 능력, 환경(상황) 및 과제와의 상호작용을 통해 결정되는 것"이다.
Csikszent -mihalyi (2003)	제목	창의성의 즐거움
	정의	창의성은 특별한 사람들의 정신적 활동이 아니라 "사람의 사고와 사회·문화적 맥락의 상호작용에서 나오는 새롭고 가치가 있는 아이디어나 행동"이다.

Guilford(1950)는 창의성이란 개인에 따라 개인차가 있을 뿐 모든 사람들이

가지고 있는 능력이라고 주장하며, 새로운 시각에서 창의성 연구에 관심을 가질 것을 촉구한 이후로 창의성에 관한 연구는 심리 측정적 접근을 통하여 구체적이고 과학적인 양상을 띄게 되었다. 그는 유창성, 융통성, 독창성을 기본 구성 요인으로 하며 정답이나 오답이 없고, 상상력을 발휘하여 주어진 문제에 대해 다양하고 보다 많은 해결책을 산출해 내도록 하는 확산적 사고를 창의성의 핵심요인으로 보았다(김윤미, 2013, 재인용).

본 연구에서는 김영채(1999)의 협의의 창의성의 개념을 기반으로 “창의성이란 특정한 사회적 맥락 속에서 어떤 문제에 대한 반응이 다양하고 독특한 해결책을 탐색하고 고안할 수 있는 확산적 사고와 수렴적 사고를 포함하는 인지적 능력”으로 정의한다.

여기에서 말하는 확산적 사고와 수렴적 사고에 대한 정의를 내리자면, 확산적 사고란 자발적, 독자적, 유동적 그리고 주도적인 것으로 이전에 가지고 있던 생각을 정교화 하고 내재된 뜻을 끌어내어 새로운 자료로 일반화시키도록 하는 것이다. 주어진 정보로부터 다른 정보를 생성하는 것으로 질적으로 우수하고 양적으로 많은 반응을 말한다. 즉 사고를 명료・확장・발전시키는 것이다(정은아, 2005). Amabile(1983)에 의하면 과제 동기, 영역관련 지식 및 창의성 기능의 수렴이 창의성이라 설명을 하고 있으며, Treffinger, Sortore와 Firestien(1983)은 수렴적 사고를 주어진 질문에 하나의 완벽한 답을 도출하는 능력으로 보고, 창의성의 비판적인 구성요인으로 보았다. 즉, 수렴적 사고가 없이는 어떠한 행위도 의사결정도 이루어질 수 없으며, 의사결정에 초점을 둘 때가 바로 확산적 사고를 그만 두고 수렴적 사고를 해야할 때라는 것이다.

창의성을 분석적으로 접근하기 위해, 이러한 창의성의 구성 요인들에 대한 관련 연구를 조사하여 인지적 영역에서의 각 학자마다 주장하는 창의성의 구성 요인에 대해 <표 Ⅱ-7>에 정리하였다.

<표 Ⅱ-7> 창의성 관련 연구에서의 창의성 구성 요인

련 연구 / 구성요인	Guilford (1956)	Torrance (1974)	Gold (1981)	Feldhusen (1983)	허경철 외. (1991)	이영덕, 정원식 (1995)	전경원 (2000) K-CCTYC
유창성	○	○	○	○	○	○	○
융통성	○	○	○	○	○	○	○
독창성	○	○	○	○	○	○	○
정교성	○	○		○	○		
민감성		○			○		
재정의 및 재구성력		○					
조직성						○	
지각적 개방성						○	
성격적 요인						○	
상상력							○

관련 연구의 조사에서 볼 수 있듯이, ‘유창성’, ‘융통성’, ‘독창성’, ‘정교성’은 창의성의 구성 요인으로 대부분 공통적으로 인정하고 있는 구성 요인이다. 본 연구에서는 창의성 검사 도구인 TTCT 언어 검사를 통해 이 중 3가지 요소인 ‘유창성’, ‘융통성’, ‘독창성’을 측정하고자 한다. TTCT 언어 검사를 창의성 평가 도구로 사용하게 되는 이유는 앞서 언급했듯이 보편적이고 신뢰성 있는 검사 도구라는 점과 컴퓨터과학에서의 프로그래밍 활동에 필요한 사고 양식이 매우 언어적인 것이라는 점 때문이다.

이미정(2011)의 연구에 따르면 유창성, 융통성, 독창성은 다음과 같은 개념이라고 할 수 있다.

유창성이란 특정한 문제 상황에서 가능한 많은 양의 아이디어를 산출해 내는 능력이다. 흔히 정확하고 훌륭한 하나의 답을 얻기 위해 오랫동안 머리를 짜내며 고민한다. 이는 가장 훌륭한 아이디어를 얻으려는 강박관념 때문에 생기는 현상이나 유창성이란 질보다는 양이 중요한 개념이다.

융통성은 고정적인 사고방식이나 시각 자체를 전환시켜 다양한 아이디어나 문제해결 방법을 산출하는 능력이다. 즉 어떤 문제를 해결하거나 아이디어를

떠올릴 때, 한 가지 방법만을 고집하지 않고 다양한 측면에서 여러 가지 방법으로 접근하려고 하는 능력이다.

독창성이란 새롭고 참신함의 의미이며, 창의성의 가장 핵심적인 요소이다. 흔히 사람들이 얘기하는 창의성이 바로 독창성을 말하는 것이기도 하다. 독창성은 기존의 사고에서 벗어나 독특하면서도 참신한 아이디어를 생성하는 능력으로 창의적 사고의 궁극적인 목표라고도 할 수 있다.

2) 창의성의 영역 보편성(일반성)과 한계

창의성의 '영역 보편성(일반성)'이란, 창의성이 지능의 'g' 요인과 같이 개인의 모든 영역에 기저를 이루며 다양한 영역에 고르게 영향을 미치는 보편적이며 공통적인 능력을 의미한다고 보는 것이다. 즉, 어느 한 영역에서 창의적 수행 수준이 높은 개인은 다른 영역에서도 유사한 창의성을 발휘한다는 것을 전제로 한다. 지금까지의 창의성 연구에서 확산적 사고는 창의성의 모든 영역에 적용될 수 있는 일반적이고 보편적인 것으로 인식되어 왔다. Torrance(1992)는 '창의적인 사고능력이란 창의적인 성취를 할 때 작용한다고 생각되는 일반화된 정신능력의 집합'이라고 정의하고, 이에 기초해 창의적 사고능력을 측정할 수 있는 TTCT 검사를 개발하였다. 그는 TTCT와 같은 확산적 사고검사에서 높은 점수를 받은 사람은 창의적으로 행동할 가능성이 높다고 주장하였다.

이와는 달리, 어느 한 영역에서의 성공적인 수행이 다른 영역과는 무관하다는 창의성의 '영역 특수성'은 Gardner(1983, 1993)의 다중지능이론에 그 기초를 삼고 있다. 다중지능이론에서 상호독립적인 인지적 영역이나 지능들이 있다는 증거가 창의성은 영역특수성이라는 것을 증명해준다는 것이다.

Ennis(1989)에 따르면 창의성의 영역 특수성은 적어도 다음과 같은 두 가지의 특징적인 원리를 가지고 있다고 한다. 첫째, 어떤 영역에서 창의적 사고를 하기 위해서는 배경지식이 필요하다는 것이고, 둘째, 창의적 사고력과 태도는 한 영역에서 다른 영역으로 무조건적으로 전이하지는 않는다고 본다. 즉 전이는 다양한 영역에서 훈련을 충분히 하고, 그리고 전이에 초점을 두고 수업을 할 때만 일어날 수 있다고 말한다.

그러나 관련 영역의 배경지식이 창의적 사고의 충분 조건이라고 추리하는

것은 오류이다. 충분한 지식과 정보가 쉽게 활성화 될 수 있는 기능적인 것이 아니거나, 전혀 다른 영역간의 활동이거나 새로운 아이디어를 생산해내는 기능이 충분히 연습되지 않았다면 창의적 성취는 기대하기 어렵다.

김영채(2012)의 연구에서 영역 보편성의 한계는 주로 4가지로 요약될 수 있다고 했다.

첫째, 창의성의 영역 특수성에 대한 경험적 증거들은 주로 피험자들이 창의해 내는 실제 창의가 영역 간에 거의 상관이 없음을 보여주는 연구에서 나온 것들이다. 어떤 경우에 보다 더 창의적인 이야기를 썼던 사람은 나중에 다시 보다 더 창의적인 이야기를 쓸 가능성이 크지만 이들이 창의적 꼴라쥬나 수학 퍼즐을 만들거나 해결해 낼 가능성은 거의 없다고 한다(Baer, 1993; Runco, 1989) 영역 특수성 연구는 주로 개인의 실제적 창의적 수행을 이용하는 반면, 영역 보편성의 증거는 주로 심리 측정적 연구에서 얻고 있는 경향이 있다 (Plucker, 1998).

둘째, 창의성은 실천적 측면을 강조한다. '보이기에는' 개념적, 분석적으로 보편적인 것 같은 것도 실제의 수행에서 보면 영역 특수적인 것일 수 있음을 강조하고 있다. 예를 들어 브레인스토밍 기능을 훈련한다고 하여 발산적 사고 기능이 다른 영역, 다른 과제로 쉽게 전이하지 않을 수 있음을 지적하였다.

셋째, 창의성의 영역 특수성은 창의성의 수준에 따라서도 다를 수 있음을 보여준다. 앞서 말한 배경지식은 전문 지식이고 이것은 매우 영역 특수적인 것이다. 필요한 지식과 훈련의 정도는 과제마다에서 다를 수 있지만 충분한 지식과 훈련이 없는 창의적 생산을 불가능하다.

넷째, 창의성을 측정하기 위해 평가 도구를 개발했던 Torrance 자신도 TTCT를 개발하며 두 가지 검사지를 개발하였다. TTCT에는 언어검사와 도형검사의 두 가지가 있고, Torrance 자신도 '언어' 과제와 '도형' 과제에서 피검자의 창의성 요인분석의 결과는 아주 상이하게 나왔고, 그래서 그는 하나는 '언어 검사', 다른 하나는 '도형 검사'라 부르게 되었다. 그는 요인분석 결과가 언어과제와 도형과제에서 다르게 나오게 된 것은 '사고의 양식'이 다르기 때문에 놀라운 일이 아니라고 말한다[11].

11) 하나는 언어적인 것이고 다른 하나는 시각적인 것임.

3) 창의성의 합류 이론(confluence theory)

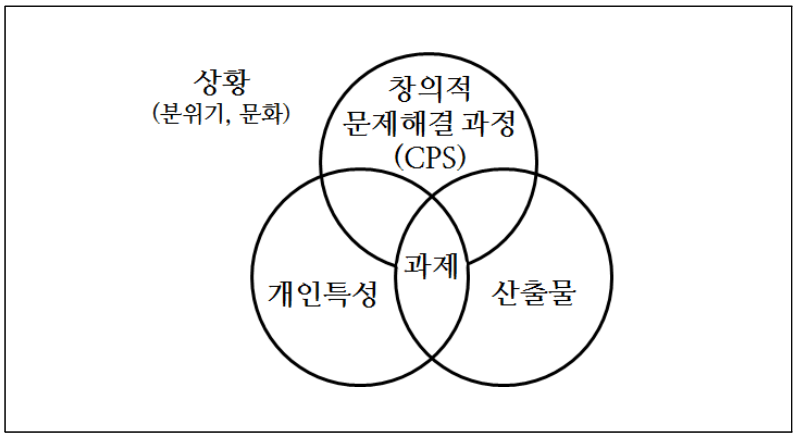
합류이론은 1990년대 이후의 창의성 연구에서 주류를 이루고 있다(박성익, 이규민, 2004; Isaksen, Puccio, & Treffinger, 1993).

합류 이론은 개인적 요소를 강조하거나 환경의 영향을 더 강조하는 등에 따라 차이가 있지만, 다음과 같은 의의와 공통점을 가진다(Lubart, 1999).

첫째, 창의성 발현을 위해서는 창의성을 구성하는 인지적·정의적·환경적 요소가 함께 수렴되어야 한다는 포괄적인 이론을 제시하고 있다. 둘째, 창의성의 개념을 복합적이고 다면적인 현상으로 설명한다. 셋째, 그 자체가 영역특수적인 이론은 아니지만 여러 영역에서의 창의성을 설명해 줄 수 있다. 넷째, 탁월한 수준의 창의적 행동뿐만 아니라 평범한 성인과 아동들의 창의적 행동도 설명해 줄 수 있다.

본 연구에서는 창의성에 대한 관점을 합류 이론의 생태학적 관점(Ecological view)으로 바라보고 있다. 생태학적 관점은 Isaksen, Puccio와 Treffinger(1993)가 제안한 것으로 개인 특성, 상황, 과정, 과제, 산출물 등과 같은 구성 요인들에 기초하여 창의성을 설명하고 있다. 생태학적 관점은 창의적 산물을 산출하는 출처들 간의 자연적인 상호작용을 이해함으로써 창의성의 다면적인 본질에 접근하고 있다.

생태학적 관점에서는 '개인', '상황', '과제', '창의적 문제해결(CPS: Creative Problem Solving) 과정' 및 '산출물'과 같은 다양한 차원의 변인들이 창의성에 영향을 미친다고 설명한다. 개인 차원에는 인지 양식, 능숙함, 동기, 개인적 기질과 성격, 지식 기반과 전문성 등이 포함되며, 상황 차원에는 심리적 환경, 문화적 가치와 규범, 집단 및 조직의 풍토, 지도자 유형과 행동, 보상 체계의 특징, 자신의 일에 대한 개념과 인식 등이 포함된다. 과제(task)는 과제의 중요도, 모호함, 특이성, 복잡성 정도가 창의성에 영향을 미치고, 과정 차원에서는 창의적 문제해결(CPS) 과정에서 거치게 되는 단계 및 기법들이 요구된다.



[그림 Ⅱ-12] 생태학적 관점의 창의성 구성 요소(윤선희, 2011)

박성익과 이규민(2004)은 과제의 특성에 따라 개인이 느끼는 호기심과 신기성의 정도가 달라지므로 과제의 특성에 따른 개인의 관심, 열정, 해결 수준 등이 달라진다고 설명한다. 따라서 개개인의 창의성을 높이기 위해서는 개별 학생에게 맞는 과제가 어떤 것인지를 알고, 개인의 특성이 반영된 과제를 중심으로 교육 프로그램을 제공하여, 개인이 특히 잘하는 특수 분야에서의 창의성을 높일 수 있는 구체적 전략들을 개발해야 한다고 주장한다.

본 연구에서는 비슷한 지적·창의적 수준의 그룹의 학생들에게 흥미롭고 이해하기 쉬운 내용과 방법을 선정하고 개발하여 이를 창의적 문제해결의 과정으로 제시한다면, 창의성 증진에 효과가 있을 것이라 기대하고 있다.

4) 창의성 검사 도구

2001년에서부터 2012년까지의 국내 창의성 관련 연구에서 창의성 검사도구로 TTCT가 가장 많이 사용되었다(정미인, 정혜인, 정세영, 김영채, 2013). 이러한 통계에 대한 결과는 컴퓨터과학 분야에서도 다르지 않다. <표 Ⅱ-8>은 최근 컴퓨터과학 분야에서 창의성과 관련된 논문들에서 사용되는 창의성 측정 검사 도구를 정리한 것이다.

<표 Ⅱ-8> 프로그래밍과 창의성 관련 연구

연구자	연구명 (학위논문)	평가 도구
이태옥 (2006)	마이크로 로봇 교육을 통한 초등학교 창의성 계발에 대한 연구	TTCT (도형) 검사
이점순 (2008)	LOGO프로그래밍 언어가 초등학생의 창의성 발달에 미치는 영향	TTCT (도형) 검사
이민희 (2009)	두리틀을 이용한 프로그래밍 수업이 창의성, 문제해결력, 프로그래밍 흥미도 향상에 미치는 영향	TTCT (도형) 검사
서영민 (2010)	초등정보영재의 창의성 신장을 위한 교과 통합 로봇 프로그래밍 수업 모형	강충열(2001)의 '창의적 성향 검사'와 자체개발한 '창의적 인지 능력 검사'
김성훈 (2010)	초등학생의 창의성 신장을 위한 스크래치 프로그래밍 교재 개발 연구	TTCT (도형) 검사
박경재 (2010)	EPL과 로봇 프로그램 교육의 창의성 신장 효과 분석	이경화(2003)의 '객관형 초등 창의성 검사'
김종진 (2011)	EPL을 이용한 창의성 증진 교육 프로그램 개발 및 적용에 관한 연구 - 로고와 스크래치를 중심으로	TTCT (도형) 검사

컴퓨터 교육 연구 분야에서도 TTCT 검사가 주로 활용되고 있으나 대부분이 TTCT 도형검사이다. 하지만 이 연구들에서 프로그래밍 활동이 TTCT 도형검사의 과제활동과 '사고의 양식'이 같기 때문임을 설명하고 있지는 못하고 있다(이민희, 2009; 이태옥, 2006).

이에 반해, 장승권(1996)은 영어로 프로그램은(program)의 어원적인 의미는 '문서를 짜기'(pro-gram), 즉 '미리 써보기'(writing-beforehand)란 의미이며 프로그램의 개념고 실천은 바로 글쓰기이라고 주장하였다. 실제로 프로그램이란 종이 위에 쓴 것이 옮겨지든, 처음부터 컴퓨터에 쓰여지든 '글쓰기'라는 행위와 분리해서는 생각할 수 없다. 다시 말해 프로그램이란 말의 의미도 글쓰기이고 그 실천 행위도 글쓰기라는 의미라고 주장하고 있다.

일반적으로 프로그래밍 자체는 문제를 해결할 수 있는 다양한 대안을 산출하고 최적의 해를 탐구하고 적용하는 과정에서 수렴적 사고를 요구하게 된다. 역설적으로 프로그래머는 이러한 과정에서 문제를 해결하기 위해 비판적·논리적 사고, 창의적 접근, 오류수정을 위한 다양한 시도를 해야 한다. 즉 프로그

래밍은 수렴적 사고뿐 아니라 확산적 사고를 요구하는 특징을 가지고 있다.

본 연구의 관점도 이런 의미에서 프로그래밍은 창의적인 글쓰기이며 논리적 문장, 기호 또는 상징을 질서화하는 행위로써 확산적 사고와 수렴적 사고를 동시에 요구하는 창의적 언어 사고의 양식이라고 바라보고 있다(전성균, 서영민, 이영준, 2011).

이러한 관점은 프로그래밍에 국한 된 것이 아니라 계산적 사고 자체에서도 찾아볼 수 있다는 의견도 있다. 2010년에 있었던 a workshop on the scope and nature of computational thinking에서 Sussman은 “계산적 사고는 언어학적 구조 기반을 갖고 있다”라고 언급하였으며(NRC, 2010), 2011년에 있었던 a workshop on the pedagogical aspects of computational thinking에서 Ursula Wolz는 계산적 사고의 개념이 저널리즘(journalism)에 스며들어 있다고 언급하였다. 그는 “언어는 저널리즘에서 찾을 수 있는 것처럼 매우 자연적일 수 있거나 컴퓨터과학에서 찾을 수 있는 것처럼 형식적(formal)일 수 있다. 형식적 비형식적 언어 모두 정보의 접근, 데이터의 집합, 정보의 통합을 포함한다. 신뢰성, 개인성, 정확성, 논리적 일관성의 개념 모두 형식적, 비형식적 언어에 필수적이며 이들은 지식의 표현과 사례의 추상화를 포함한다.”고 설명하였다(NRC, 2011). Wolz 외(2011)의 실제적인 연구가 이러한 주장을 뒷받침해주었다. Wolz의 연구는 글쓰기 활동을 통해 계산적 사고를 능동적으로 활용할 수 있도록 설계하고 적용하였고 학생의 설문결과 중 스크래치 프로그램 제작에 가장 필요한 기술은 수학적, 예술적 능력보다 ‘글쓰기’라는 답변을 얻었다.

5) 계산적 사고와 프로그래밍

프로그래밍이 컴퓨터과학의 핵심 개념을 가르치기에 가장 적합한 활동이라는 것은 앞서 많은 학자의 생각을 빗대어 언급했다. 최근 계산적 사고에 대한 관심이 높아지면서 계산적 사고와 프로그래밍간의 관계에 대한 연구가 보고되고 있다. 프로그래밍 활동 또한 하나의 학습이며 각 단계마다 사용되는 인지영역이 있다면 Anderson과 Krathwohl(2002)에 의해 다음과 같은 수정된 블룸(Bloom)의 분류를 사용하여 구분하였다.

<표 Ⅱ-9> 프로그래밍에서의 계산적 사고의 발현

<table>
<tr><th rowspan="2">프로그래밍 단계</th><th rowspan="2">Bloom의 인지영역 분류 (Anderson, & Krathwohl, 2002)</th><th colspan="3">계산적 사고의 요구 기술(skill) 또는 구성 요인</th></tr>
<tr><th>유종현, 김종혜 (2008)</th><th>Selby (2012)</th><th>L'Heureux et al. (2012)</th></tr>
<tr><td>기본 활용 익히기</td><td>기억하기</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>요구사항 이해</td><td>이해하기</td><td>비판적 사고</td><td>분해적 사고</td><td>논리적 사고</td></tr>
<tr><td>해결방안 탐색, 설계</td><td>응용하기</td><td>논리적, 알고리즘적, 비판적, 재귀적 사고</td><td>추상화</td><td>전략화, 추상적 사고, 절차적 사고</td></tr>
<tr><td>모듈화, 코딩, 오류 수정</td><td>분석하기</td><td rowspan="3">논리적, 알고리즘적, 비판적, 재귀적 사고</td><td>일반화</td><td>최적화</td></tr>
<tr><td>테스팅 및 보완</td><td>평가하기</td><td>모델링 /시뮬레이션</td><td rowspan="2">반복적 개선</td></tr>
<tr><td>독창적 프로그램 완성</td><td>창조하기</td><td>알고리즘 설계</td></tr>
</table>

관련 연구들을 통해 볼 때, 많은 연구에서 프로그래밍 활동의 각 단계가 계산적 사고의 어떠한 구성 요인들을 요구하고 있음을 나타내고자 하고 있다. 즉, 많은 학자들은 프로그래밍 활동이 계산적 사고 자체가 발현되는 활동임을 인식하고 이를 분석적으로 제시하여 나타내고자 하고 있다.

6) 창의성과 프로그래밍

창의성 증진을 위한 프로그래밍 교육과 관련된 연구에 대해서 전성균, 서영민, 이영준(2011)은 지금까지의 관련 연구들을 크게 두가지 관점으로 분류하였다. 첫 번째 분류의 연구들은 프로그래밍 교육을 시행했고 그 결과가 문제해결력, 논리적 사고력뿐 아니라 부수적으로 창의성도 증진되었기 때문에 창의성 교육도 가능할 것이라는 시사를 해준 연구들이다. 두 번째 분류의 연구들은 로봇 등의 새로운 학습도구를 이용하여 학생들의 흥미를 돕고 학습하니 창의성 교육이 가능했다는 것인데 이는 일반화가 어려운 단점이 있다는 것이다.

최근 몇몇의 실증적인 연구들을 통해 컴퓨터를 활용한 다양한 학습이나 활동으로 인해 창의성을 증진시킬 수 있다는 연구 결과들이 보고 되었다(서영민,

이영준, 2010; 최성규, 정남용, 2003; Lewandowski, Johnson, & Goldweber, 2005; Li & Chen, 2007; Romeike, 2007, 2008). 특히, 컴퓨터과학 분야 중에서도 프로그래밍을 학습을 통한 창의성 신장 연구가 관심을 최근 많이 받고 있다.(김갑수, 2010; 전성균, 서영민, 이영준, 2011; Gallardo, Julia & Jorda, 2008). 또한, 최근의 연구에서는 프로그래밍과 창의성에 대한 관계에 대한 고찰을 심도 있게 하기 시작하였다.

<표 Ⅱ-10> 프로그래밍과 창의성 관련 연구

연구자 (발행연도)	연 구 내 용	
유인환, 김태완[3] (2006)	제목	MINDSTORMS을 이용한 프로그래밍 학습이 창의력에 미치는 효과
	핵심 내용	로봇 프로그래밍에 있어 창의력 관련 구인들을 유창성, 독창성, 융통성, 정교성으로 정의하고 MINDSTORMS을 이용하여 학습자의 사전 사후 검사를 통해 창의성의 구성 요인별로 분석하였고 융통성과 정교성이 유의미한 차이가 있었다. 창의성 검사는 TTCT 도형 2개 활동, 언어 4개 활동을 활용하였다.
	시사점	학습 활동 중 창의성이 요구되는 활동이 매우 많고 다양하여 어떠한 변인이 창의성에 영향을 미쳤는지에 대한 분석을 하기가 어렵다.
이은경[3] (2008)	제목	로봇 활용 프로그래밍 학습이 창의적 문제해결성향에 미치는 영향
	핵심 내용	이은경의 경우 MINDSTORMS를 이용하여 창의적 문제 해결성향을 인지적, 정의적, 지식구성 요인으로 구분하여 다변량 분석을 통해 유의미한 상관관계를 도출하였다. 창의적 문제해결성향 검사는 강정하와 최인수의 투자이론검사를 사용하였다.
	시사점	로봇 프로그래밍을 통해 창의성 증진에 대한 긍정적 영향을 검증하였다.
전성균, 서영민, 이영준[3] (2011)	제목	창의성과 프로그래밍 교육에 관한 고찰
	핵심 내용	・창의성의 하위요인: 수렴적 사고, 발산적 사고 ・프로그래밍 교육의 특징: 수렴적 사고, 발산적 사고 프로그래밍을 통해 프로그램을 구현한다는 것은 결국 최적의 코드를 선정하기 위해 수렴적 사고가 많이 강조되는 측면이 있지만, 프로그래밍 교육에 확산적 사고가 포함되어 있다는 것(문제해결을 위해 다양한 방법을 찾아보고, 오류 발생 시 오류수정을 위해 다양한 생각의 시도)은 프로그래밍 교육을 통한 창의성 교육이 가능하다는 의미를 지닌다. 하지만 프로그래밍 교육을 통해 창의성 교육이 가능하다는 명제가 항상 성립하는지는 의문이다. 최근 창의성 연구에서 수렴적 사고의 중요성을 재인식하고는 있지만, 프로그래밍 교육 특성상 수렴적 사고의 비중이 더 크기 때문에 확산적 사고를 촉진할 수 있는 방향으로 연구가 진행되어야만 창의성 증진에 긍정적인 영향을 미칠 수 있을 것이다.

연구자 (발행연도)	연 구 내 용	
	시사점	창의성과 프로그래밍이 수렴적 사고와 발산적 사고라는 공통의 요인을 요구하는 능력임을 구체적인 활동 예를 들어 논리적으로 설명하고 있다.
윤선희, 김영식[3] (2011)	제목	정보과학 창의성의 구성 요소 탐색
	핵심 내용	· 창의성의 영역특수성: 학문 영역마다 고유의 창의성이 존재하고 이는 다른 영역으로 전이되지 않는다는 개념 · 창의성의 합류 이론: 일반적인 창의적 사고 기술에 더불어 영역특수적 기술, 환경의 촉진이나 방해 등이 창의성 발현에 모두 요구된다는 개념 영역특수성과 합류 이론에 기반하여 컴퓨터과학 교과에 적합한 창의성의 구성 요소를 델파이 방법으로 탐색하였다. 결과적으로, 인지적 영역 · 정의적 영역 · 환경적 영역에서 창의성과 관련된 요소들을 선정하였다.
	시사점	프로그래밍과 창의성이 어떠한 관계로 서로 상관이 있는지에 대한 언급은 없으나 정보영재 교사, 프로그래머 및 연구원 등의 다수의 다양한 전문가의 참여로 창의성에 대한 요소에 대한 합의적(델파이 기법) 정의를 내릴 수 있었음.

[1] 박사학위 논문, [2] 석사학위 논문, [3] 학회지 논문 또는 학술대회 발표 논문

전성균, 서영민, 이영준(2011)의 연구에 따르면 프로그래밍 교육과 창의성의 구성 요인이 모두 수렴적 사고와 발산적 사고를 포함하고 있기 때문에 프로그래밍 교육을 통한 창의성 신장은 가능하다고 보고 있다. 예를 들자면, 프로그래밍 자체는 다양한 가능한 대안 중에서 최적의 해를 찾아 검토하고 적용하는 과정에서 비판적 사고 및 논리적 사고가 활발히 이루어지고 이는 수렴적 사고의 특징을 지닌다. 또한 문제를 인식하여 문제해결을 위한 다양한 방법을 찾아보고, 또한 오류 발생 시 오류수정을 위해 다양한 생각을 해봄으로써 확산적 사고의 특징을 지닌다고 할 수 있다는 것이다.

이러한 주장은 관련 연구들을 통해 실제적으로 검증되고 있다(유정수, 이민희, 2009; 전성균, 이영준, 2012; Knobelsdorf & Romeike, 2008; Lewandowski, Johnson, & Goldweber, 2005; Long, 2007; Romeike, 2008).

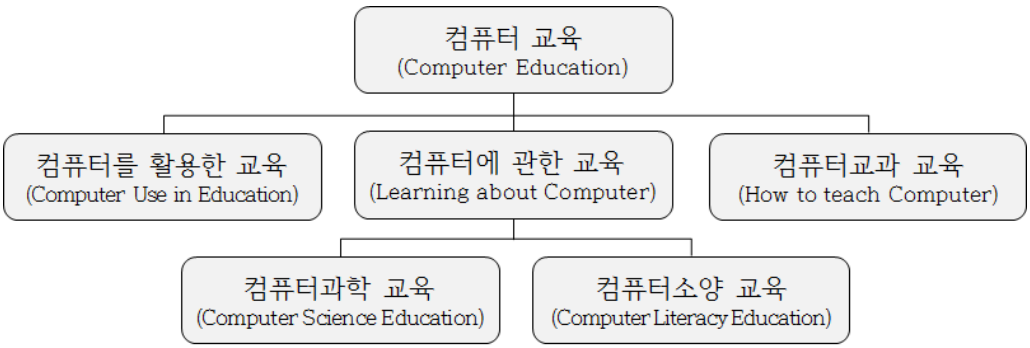
3. 컴퓨터과학과 프로그래밍

1) 컴퓨터과학 교육

IT에 미래 사회의 국가 경쟁력이 달려 있다는 말을 부정할 수 있는 사람은 없을 것이다. 이런 맥락에서 공교육에서의 컴퓨터과학 교육의 필요성이 그 어느 때보다 강조되고 있는 시점이다(이원규, 정효숙, 2004).

하나의 학문이 타 학문과 구분될 수 있는 중요한 기준은 학문에서 다루는 내용이 독립적이어야 할 것이다. 컴퓨터과학의 학문적 내용은 문제해결 과정 자체와 사고에 초점이 맞추어져 있다는 점에서 다른 학문과 구분된다(Tucker et al., 2011).

컴퓨터 교육의 영역은 크게 컴퓨터를 활용한 교육, 컴퓨터에 관한 교육, 그리고 컴퓨터교과 교육으로 나눌 수 있다.



[그림 Ⅱ-13] 컴퓨터교육의 영역

컴퓨터를 활용한 교육(Computer Use in Education)은 컴퓨터를 교수·학습의 도구로 하여 학습과정에 활용하는 것을 말한다. 컴퓨터에 관한 교육(Learning about Computer)은 컴퓨터 자체를 학습의 대상으로 하는 것으로써 컴퓨터의 작동 원리, 구조 및 응용 프로그램 등 컴퓨터를 실제 이용하기 위한 컴퓨터에 대한 지식을 가르치는 것이다. 컴퓨터 교과교육(How to Teach Computer)은 컴퓨터를 어떻게 가르칠 것인가에 관계된 컴퓨터 교육론, 컴퓨터 교재 연구, 컴퓨터 교육평가, 컴퓨터 지도 및 방법론 등에 관한 것이다. 컴퓨터

에 관한 교육은 컴퓨터과학 교육(Computer Science Education)과 컴퓨터소양 교육(Computer Literacy Education)이 있다. 컴퓨터소양 교육은 컴퓨터 문맹퇴치를 목적으로 컴퓨터 사용법을 교육하는 것으로 워드, 엑셀 등의 응용소프트웨어 사용법을 익히고, 인터넷을 통한 정보 획득과정을 교육한다. 컴퓨터과학 교육은 컴퓨터과학의 핵심을 교육하는 것이다.

즉, 컴퓨터과학 교육에서 교육자로서 생각해야 할 점은 무엇이 컴퓨터과학의 핵심이며 그 목적이 무엇인가에 대한 정립이 필요하다는 것이다.

컴퓨터과학은 문제해결 과정 자체에 대한 사고 능력을 기르게 한다(Tucker et al., 2011). 현대 사회의 컴퓨터 과학자(computer scientist)들은 사업가, 예술가, 과학자, 각 분야의 전문가들과 밀접하게 활동하고 있다. 그들이 하고 있는 일은 주어진 문제를 명쾌하고 분명하게 서술하여 계산으로 정의하고 이를 컴퓨터로 표현하는 것이다. 컴퓨터 과학자와 다른 분야의 전문가들이 이러한 관계는 미래 사회에서 더욱 밀접해질 것으로 예측된다. 사실 컴퓨터 과학자들에게 예술과 생물학, 건축, 경제를 모두 가르치는 것보다 모든 이가 컴퓨터과학의 핵심을 이해하는 것이 미래 사회에서 이들 간의 의사소통에 더 큰 도움이 되리라 생각이 든다. 이러한 이유에서 Wing(2006)은 컴퓨터과학 교육이 3R처럼 필수적이여야 한다는 주장을 하고 있는 것이다.

<표 Ⅱ-11> CSTA K-12 컴퓨터과학 교육 기준안의 목표

The goals of CSTA K-12 Computer Science Standards
1. 컴퓨터과학의 본질과 현대 사회에서의 위상을 이해할 수 있도록 한다.
2. 개념과 기술이 조합된 컴퓨터과학을 이해할 수 있도록 한다.
3. 다른 교과의 문제 해결 활동을 통해 컴퓨터과학 기술들(특히 계산적 사고)을 사용할 수 있도록 한다.
4. 본 기준안은 학생들의 소속 학교의 IT 교육과 AP(Advanced Placement) 컴퓨터과학 교육 교육과정을 보완할 수 있도록 한다.

계산적 사고력의 신장이 컴퓨터과학 교육 목표의 핵심이 되어야 한다는 이슈(issue)는 미국의 CSTA K-12 컴퓨터과학 교육 기준안에서부터 제시되었다.

이 기준안에서 컴퓨터과학 교육의 목표를 4가지로 정의하였다(Tucker et al., 2011).

현행 우리나라의 2009 개정 교육과정 초·중등학교 총론(교육과학기술부, 2012)에서 컴퓨터과학 교육의 목적에 대한 언급은 중학교 선택교과의 '정보' 교과에서 잘 찾아볼 수 있다. 물론 고등학교 공업 전문 교과의 '정보 기술과 활용' 과목이나 상업 정보 계열 전문 교과의 '자료 구조', '모바일 콘텐츠', '프로그래밍 실무', '미디어 콘텐츠 일반', '미디어 콘텐츠 실무', '웹 프로그래밍', '컴퓨터 일반' 과목에서도 컴퓨터과학 교육의 목적에 대한 내용은 엿볼 수 있으나 이들 과목에서의 목표는 비교적 세부적인 내용에 대한 목표이다.

<표 Ⅱ-12> 중학교 '정보' 교과의 목표

중학교 「정보」 교과의 목표
정보는 정보 과학 기술의 기본 개념과 원리를 이해하고, 실생활의 다양한 문제를 계산적 사고(computational thinking)로 관찰하고 해결하는 능력과 정보 윤리적 소양을 기르는데 중점을 둔다.
가. 정보 과학 기술의 기본 개념과 원리를 습득하고, 계산적 사고력을 익혀 창의적이고 효율적인 문제해결 능력을 갖춘다.
나. 미래 정보사회의 일원으로 갖추어야 할 소양인 정보 윤리 및 정보 보호, 정보기술 및 기기에 대한 이해를 바탕으로, 이를 올바르게 활용하고 실천할 수 있는 태도를 기른다.
다. 정보과학의 논리적, 절차적 사고를 통해 일상생활 문제를 효율적인 알고리즘으로 해결하고, 이런 사고를 실생활과 정보 기기에 적용하는 능력을 기른다.
라. 다른 학문들과 통합되어 새로운 형태로 확장, 발전시켜나가는 융합 학문 분야를 개척할 수 있는 능력과 태도를 기른다.

2009 개정 교육과정의 중학교 '정보' 교과의 목표는 CSTA의 교육과정 목표보다 잘 정의되어 있다고 판단된다. 특히 계산적 사고 또는 계산적 사고로 번역될 수 있는 'Computational Thinking'이 본 교과의 핵심임을 엿볼 수 있다. 하위 목표 4가지를 분석해보자면 ① 창의적 사고력을 통한 문제해결 능력, ② 정보윤리·보호·기술 및 기기에 대한 이해와 올바른 활용 및 태도, ③ 일상생활 문제의 알고리즘적 해결과 적용 능력, ④ 융합 학문 분야의 개척 능력과 태

도를 신장시키기 위해 '정보' 교과를 학생들에게 가르칠 필요가 있다는 것이다.

본 연구에서 컴퓨터과학 교육의 목표를 '정보' 교과의 목표를 참고하여 재정의 하자면 <표 Ⅱ-13>과 같다.

<표 Ⅱ-13> 수정된 목표

컴퓨터과학 교육의 목표 (본 연구에서의 재정의)
컴퓨터과학 기술의 기본 개념과 원리를 이해하고, 실생활의 다양한 문제를 계산적 사고(computational thinking)로 관찰하고 **표현하여 창의적 문제해결 능력**과 정보윤리적 소양을 기르는데 중점을 둔다.

본 연구에서 컴퓨터과학 교육의 목표를 재정의함에 있어 강조하고자 하는 바는 학생들 스스로 실생활의 문제를 추상화 능력을 통해 계산을 도출하여 표현할 수 있느냐는 것이다. 특히 여기에서의 계산은 궁극적으로 컴퓨터의 자동화를 위한 것이므로 이를 염두하여 정형화되고 논리적이여야 한다는 점이다. 또한 컴퓨터과학의 여러 문제해결 방법을 가르치고 학습하는 이유는 기존의 문제해결 방법을 익혀 똑같이 해결해나가고자 하는 것이 전부가 아니다. 더 근본적인 것은 이를 통해 처음 접해보는 다른 문제들을 창의적으로 해결할 수 있기를 기대하기 때문이라고 할 수 있기에 '창의적 문제해결 능력'이라는 단어를 삽입하였다.

2) 컴퓨터과학에서의 프로그래밍

창의성과 계산적 사고의 증진이라는 본 연구의 목적을 보았을 때, 컴퓨터과학의 여러 분야 중 프로그래밍을 학습 활동으로 선택하게 된 이유를 크게 두 가지로 말할 수 있다.

첫째 이유는 많은 학자들의 컴퓨터과학에서의 프로그래밍에 대한 중요성을 강조하고 있다는 점이다(Resnick et al., 2009; Tucker et al., 2011). 컴퓨터과학의 핵심 개념의 대부분을 실제적인 활동을 통해서 학습할 수 있는 매우 효과적인 활동이라는 것이 많은 학자들의 공통적인 의견이다.

둘째 이유는 프로그래밍과 관련된 많은 연구들에 의한 계산적 사고 또는 창의성과 관련된 실증적인 연구 결과들 때문이다(김병수, 김종훈, 2012b; 유정수, 이민희, 2009; 이은경, 2013; 전성균, 이영준, 2012; Bennett & Repenning, 2010; Brennan & Resnick, 2012; Knobelsdorf & Romeike, 2008; Koh, Basawapatna, Seiter & Foreman, 2013; Lewandowski, Johnson, & Goldweber, 2005; Long, 2007, Romeike, 2008). 즉 프로그래밍이 컴퓨터과학 학습에 매우 중요하다는 학자들의 주장을 실증적인 연구 결과로 입증하였다.

본 연구에서도 컴퓨터과학의 많은 학습 활동 중 프로그래밍을 선택하게 된 이유도 여기에 있다.

3) 스크래치 프로그래밍 언어

비쥬얼 프로그래밍 언어(visual programming languages)는 사용자가 프로그램을 어떻게 작성하는지에 대한 방법을 효과적으로 가르쳐 줄 수 있는 도구이다. 전통적인 프로그래밍 방법과 확실한 차이는 사용자의 동기 부여가 더 크다는 점이다(Koh, Basawapatna, Bennett & Repenning, 2010).

본 연구에서 적용한 프로그래밍 언어는 연구 대상자의 수준에 맞춰 스크래치(http://scratch.mit.edu)로 제시하였으나 본 연구에서 개발된 교육 프로그램이 스크래치에만 적용되는 것은 아니다. 하나의 프로그래밍 언어를 교육하기 위해서는 해당 프로그래밍 언어의 특징을 파악하고 프로그래밍 활동 중에 오개념이 생길만한 소지에 대한 내용 분석을 확실히 할 필요가 있다. 그러한 의미에서 본 연구의 프로그래밍 학습 도구로 사용된 '스크래치'의 특징 및 명령처리에 관한 내용을 심도있게 분석할 필요가 있다고 본다.

(1) 스크래치 관련 연구

스크래치는 2007년 5월 미국 MIT Media Lab에서 10대 청소년들과 초보 프로그래머들이 쉽게 프로그래밍을 이해할 수 있도록 만든 교육용 프로그래밍 언어이다. 현재 스크래치 웹 사이트(http://scratch.mit.edu)를 이용하여 프로젝트를 공유하는 핵심 사용자의 연령은 8세에서 16세이며, 12세가 가장 많은 편이다. 스크래치는 수학 및 컴퓨터과학의 개념뿐 아니라 창의적이며 조직적인 사고와 협력적 작업 능력을 증진시킬 수 있다(Resnick et al., 2009)

스크래치의 특징은 학습자가 애니메이션, 게임, 인터액티브 아트(interactive art)를 쉽게 만들 수 있게 해주는 '미디어 기반 프로그래밍 환경'이다. 레고나 퍼즐조각처럼 블록 모양의 명령어를 끼워 맞추는 방법으로 프로그래밍을 처음 접하는 초보자들이 조작하기에 매우 쉬운 환경을 제공하고 있다. <표 Ⅱ-14>에서 볼 수 있듯이 스크래치를 활용한 학습은 프로그래밍 교육이 갖는 장점을 살리면서 학습자의 동기를 자극해 흥미와 관심을 불러일으켜 학습 성취도를 높일 수 있을 뿐만 아니라 창의적인 문제해결능력과 논리적 사고력 및 알고리즘 사고력을 향상시키고 있다.

<표 Ⅱ-14> 스크래치 프로그래밍 관련 연구

연구자 (발행연도)	연 구 내 용	
Resnick, M. (2007)	제목	All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten
	내용	'Imagine, Create, Play, Share, Reflect and back to Imagine' 이라는 나선형 학습 접근 방법에 스크래치와 같은 새로운 기술을 도입한 학습 방법에 대해 기술하였다.
Maloney et al. (2008)	제목	Programming by choice: urban youth learning programming with scratch
	내용	남중부 로스앤젤레스의 클럽하우스(방과후학교 센터)에서 8세에서 18세까지의 학생들에게 스크래치가 소개된 후의 2년 동안 536개의 프로젝트가 제작되었다. 이 프로젝트 스크립트를 분석한 결과, 프로그래밍의 핵심 개념인 반복(Loops), 통신과 동기화(Communications and Synch.), 논리연산(Boolean logic), 변수(Variables), 난수(Random Numbers)의 사용이 1년차보다 2년차에 유의미하게 늘어났음을 보여주었다.
Romeike, R. (2007)	제목	Applying creativity in CS high school education: criteria, teaching example and evaluation
	내용	고등학교에서의 컴퓨터과학 학습에서 창의성에 대한 평가 준거를 설정한 프로그래밍 입문 학습에서 학습자들의 동기유발과 흥미, 학습 성취도가 증진되었다.
송정범, 조성환, 이태욱[3] (2008)	제목	스크래치 프로그래밍 학습이 학습자의 동기와 문제해결력에 미치는 영향
	내용	초등학교 6학년 학생들의 재량활동 시간을 활용하여 스크래치 프로그래밍 학습한 결과 학습자의 내재적 동기와 문제해결력 향상에 효과가 있는 것으로 나타났다.
	검사도구	작업 선호도 검사(WPI)와 문제해결력 검사(PISA 문제)
조성환, 송정범,	제목	CPS에 기반한 스크래치 EPL이 문제해결력과 프로그래밍 태도에 미치는 효과

연구자 (발행연도)	연 구 내 용	
김성식, 이경화[3] (2008)	내용	중학교 남녀 학습자에게 CPS 모형 기반의 스크래치 프로그래밍 학습이 문제해결력 향상과 프로그래밍 교육에 대한 태도에 긍정적인 영향을 미쳤다.
	검사도구	PISA 2003 문제해결 영역 평가 문항, 우영애가 개발한 프로그래밍 교과의 정의적 태도 영역 검사지
김종진, 현동림, 김승완, 김종훈, 원유헌 (2010)	제목	교육용 프로그래밍 언어인 로고와 스크래치 교재 개발 및 비교 실험
	내용	초등학생을 대상으로 로고와 스크래치 프로그래밍 학습 후, 창의성 검사 결과를 교차 분석하여 비교한 결과, 두 언어 모두 창의성에 긍정적 영향을 미쳤다. 특히 로고는 창의성 영역의 유창성, 스크래치는 추상성과 저항 영역에 긍정적인 영향을 주었다.
	검사도구	TTCT 창의성 검사(도형)
한선관, 김수환, 서정보[3] (2010)	제목	스크래치 프로그래밍을 활용한 게임중독 치료 프로그램의 개발
	내용	초등학교 게임중독 고위험군을 대상으로 게임중독 치료 대안활동과 함께 스크래치 교육 프로그램을 적용한 결과, 게임중독 치료에 긍정적인 답변을 얻고, 휴식 시간 게임 접속 시간이 감소된 것으로 분석되었다. 관찰과 면접을 통합 분석 결과 자아존중감과 창작의 기쁨, 자아 제어 능력에도 긍정적인 영향을 미쳤다.
안경미, 손원성, 최윤철[3] (2011)	제목	스크래치 프로그래밍 교육이 초등학생의 학습 몰입과 프로그래밍 능력에 미치는 효과
	내용	초등학교 고학년을 대상으로 스크래치 프로그래밍 학습 집단과 순서도 학습 집단을 비교한 결과, 스크래치 프로그래밍 학습이 상대적으로 학습몰입 수준과 프로그래밍 능력에 긍정적인 영향을 미침을 보였다.
	검사도구	학습 몰입 수준 검사지(Csikszentmihalyi의 검사지를 수정하여 사용)와 자체개발한 프로그래밍능력 평가 문항
박용철, 이수정[3] (2011)	제목	스크래치 프로그래밍 교육이 초등학생의 자기 주도적 학습 능력에 미치는 효과
	내용	초등학교 6학년 학생들에게 스크래치 프로그래밍을 실시한 결과, ICT 소양 프로그래밍 수업에 비해 자기주도적 학습 능력의 하위요소인 개방성, 내재적 동기, 자율성 영역의 신장에 유의한 차이를 보였다. 특히 ICT 활용 능력이 우수한 학생들에게 그러한 신장이 크게 나타났다.
	검사도구	자기주도적학습 검사 (Gueliemino의 자기주도적 학습 준비도 검사 수정본)
함성진, 양창모[3] (2011)	제목	스크래치를 이용한 초등학교 컴퓨터 교육과정 설계
	내용	프로그래밍 교육을 위한 교육과정을 설계하여 전문가 집단으로부터 학습 대상, 교육 요소, 학습 내용, 학습량, 교수학습지도안의 적정성 등의 거의 모든 영역에서 적절하다는 평가를 받았다.
류충구[2] (2012)	제목	Scratch가 초등 영재들의 창의적 문제해결력에 미치는 효과

연구자 (발행연도)	연 구 내 용	
	내용	초등학교 영재학생들을 대상으로 스크래치 프로그래밍 학습이 비주얼베이직 프로그래밍 수업보다 창의적 문제해결력에 긍정적인 효과를 미친다.
	검사도구	창의적 문제해결력 검사 (서울대 심리연구실 MI연구팀 개발)
조준필[2] (2012)[1]	제목	기술교육에서 중학생의 논리적 사고력 함양을 위한 스크래치 학습 프로그램 개발
	내용	중학생들의 기술교육에 있어서 논리적 사고력을 함양시키기 위한 스크래치 학습 프로그램을 전문가의 검증을 통해 개발하였음.
박정신, 조석봉[3] (2012)[1]	제목	프로그래밍입문 수업에서 스크래치 활용 효과분석
	내용	전문대학의 컴퓨터 전공 학생들이 프로그래밍언어 수업에서 스크래치를 활용하여 비활용반과의 비교에서 중간고사와 기발고사의 점수를 비교하여 알고리즘 구상과 프로그래밍 작성에 효과가 있음을 보였다.
	검사도구	자체 개발한 평가 도구 (개념 평가 질문 유형과 코드 작성 문제 유형)

[1] 박사학위 논문, [2] 석사학위 논문, [3] 학회지 논문 또는 학술대회 발표 논문

<표 Ⅱ-14>에서 보듯이 스크래치 프로그래밍과 관련된 연구들의 목적들은 논리적 사고력 신장, 몰입도 향상, 자기주도적 학습 능력 향상, 문제해결력 향상, 창의성 향상, 게임 중독 치료 등으로 매우 다양하였다. 본 연구에서 밝히고자 하는 프로그래밍 학습과 계산적 사고력의 증진에 대한 상관연구 및 비교실험 연구는 미비했으며 창의성에 대한 연구 또한 많지 않은 편이라고 할 수 있겠다.

(2) 스크래치 프로그래밍 언어의 특징

① 블록 조립형 프로그래밍

프로그래밍을 처음 접하는 사람들도 쉽고 재미있게 프로그래밍의 개념을 익힐 수 있도록 레고 블록처럼 블록을 가져다 쓰는 개념으로 스크립트를 작성할 수 있다. 이는 사용자가 해당 프로그래밍 언어의 키워드와 문법을 외워야 하는 인지적 부담을 줄여준다. '스크래치'라는 이름의 유래에서도 이러한 개념을 이해할 수 있는데, '스크래치'의 의미는 레코드판을 턴테이블에 가지고 와 DJ(디스크자키)의 취향대로 음악을 믹싱(mixing)하는 작업을 의미한다.

② 객체지향 프로그래밍(OOP: Object-oriented programming)

스크래치는 전통적인 객체지향 프로그래밍의 개념을 어느정도 가지고 있다고 말할 수 있다. 객체지향 프로그래밍의 핵심 개념과 주요 특징을 중심으로 스크래치는 어느 정도 객체지향 프로그래밍의 속성을 가지고 있는지 자세히 알아볼 필요가 있다.

<표 Ⅱ-15> 객체지향 프로그래밍 관점에서 본 스크래치의 특징

개 념	스크래치 1.4	스크래치 2.0
객 체	○	○
클래스	×	▲
추상화	×	▲
캡슐화	▲	▲
상속성	×	▲
다형성	×	▲
메시지 전달	○	○

(○: 프로그램 내 지원 / ▲: 일부 지원 및 구현 가능 / ×: 지원 및 구현 불가능)

- 객체: 객체는 독립적으로 자신을 표현할 수 있고, 동작 가능한 가장 작은 단위이며 구성요소(속성과 기능)를 가지고 있다는 측면에서 스크래치의 하나의 스프라이트는 하나의 객체라고 할 수 있다. 사용자가 의식하지 않더라도 스크래치의 스프라이트는 다양한 속성(x・y좌표, 방향, 회전방식, 그래픽 효과, 크기 등)을 지니게 된다.
- 클래스와 추상화: 클래스란 동일한 속성과 메소드를 가진 객체를 생성하기 위한 틀이라고 정의할 수 있다. 스크래치에서는 클래스를 이용하도록 지원하지는 않는다. 하지만 기능적 측면에서 클래스처럼 각 객체의 공통적 속성을 정의하여 객체를 생성하여 사용할 수 있다. 이러한 측면에서 추상화의 특징을 가지고 있다고 할 수 있다. 예를 들어 움직이는 다양한 색깔의 구슬에 사용자가 마우

스를 가져가 대면 구슬이 사라지게 하려는 기능을 구현한다고 해 보자. 이를 위해 <부록 1>과 같은 스크립트를 작성할 수 있다. 이 프로그램(프로젝트)이 실행되면 'ball' 스프라이트가 가지고 있는 색깔 속성이 0으로 세팅된다. 총 10회의 자기 복제가 일어나는데 이 때마다 색깔 속성은 10만큼 바뀌게 된다. 물론 스크래치의 모든 스프라이트는 공통된 속성들을 가지고 있기 때문에 위의 스크립트를 클래스 정의라고 하기에는 부족하다고 볼 수도 있겠지만 어디까지나 클래스의 개념을 이용할 수는 있다. 위 스크립트에서 복제되는 스프라이트들은 복제 당시 부모 객체가 가지고 있던 색깔 속성의 값을 그대로 물려받게 된다.

- 캡슐화: 클래스의 어떠한 부분은 외부에 접근할 수 있도록 공개하고 그 외에는 접근하지 못하도록 접근을 차단하는 정보은닉의 개념이다. 스크래치에서는 이러한 캡슐화가 클래스 측면이 아니라 객체(스프라이트)의 변수를 통해 구현된다고 할 수 있다.
- 상속성: 복제의 기능이 도입된 스크래치 2.0 버전에서 객체(스프라이트) 레벨에서 스프라이트 변수를 생성하여 사용할 수 있지만 복제 시에 이 변수까지 복제되는 것은 아니다. 앞서 말했듯이 복제 시 상속되는 것은 부모 객체의 속성뿐이다. 복제된 스프라이트가 다시 복제될 때에도 이러한 규칙은 계속된다.
- 다형성: 프로그램에서 메소드의 인자를 달리하여 동일한 이름의 함수 호출로도 다른 기능을 구현해 나가는 것이다. 스크래치 2.0 버전에서는 '추가 블록' 그룹이 새롭게 생성되어 블록을 사용자가 직접 만들 수 있으며 이 때 같은 이름의 블록을 만들 수 있으며 인자들을 달리할 수 있다. 하나의 스프라이트 또는 복제된 스프라이트 안에서만 가능하다.
- 메시지 전달: 스크래치에서 다른 객체의 속성을 참조할 수는 있지만 접근하여 수정할 수는 없다. 객체 자신의 속성은 내부에서만 수정이 가능하다. 객체간 통신의 방법은 '방송하기'라는 방법으로 제공한다. 만약 다른 객체의 속성의 변경이라든지 기능을 필요로

할 때에는 '방송하기'를 통해 해당 객체가 직접 그것을 할 수 있도록 해야 한다.

③ 다중 프로그래밍(multiprogramming)

다중 프로그래밍은 여러 개의 프로그램들이 단일처리기(processor) 상에서 동시에 실행되는 것이다. 단일처리기이기 때문에 실제로 여러 개의 프로그램이 동시에 수행된다고는 볼 수 없다. 그런 이유로 의도치 않은 오류가 생길 수 있다. 스크래치를 이용하여 교수·학습 시에는 이러한 오류가 스크래치의 오류가 아닌 다중 프로그래밍 기법의 오류임을 인식하고 지도하여야 학생들에게 오개념이 생기지 않을 것이다. 예를 들어 <부록 2>의 스크래치 프로젝트처럼 '선수1'과 '선수2'의 스프라이트를 생성하고 출발선으로부터 같은 거리에서 동시에 출발시켰을 때 누가 먼저 도착할까?

<표 Ⅱ-16> 스크래치 스크립트의 처리 순서

	스크래치 1.4 버전	스크래치 2.0 버전
스프라이트 내	• 최근 스크립트 창에 등록된 제어블록 모듈 또는 그 전체가 새로운 위치로 움직여진 모듈의 스크립트가 먼저 실행된다.	• 최근 스크립트 창에 등록된 이벤트 블록 모듈의 스크립트가 가장 나중에 실행된다.
	• 반복문 내 스크립트 명령문은 1차례씩 돌아가며 실행되지만, 순차적으로 연결된 스크립트 명령문은 끝날 때까지 처리된다.	
스프라이트 간	• 최근 생성된 스프라이트의 스크립트가 먼저 실행된다. • 스프라이트 내의 모든 스크립트 실행 후 모두 종료되거나 반복문만 남았다면 다음 스프라이트로 처리 순서를 넘긴다.	

이러한 스프라이트 내 또는 스프라이트 간의 스크립트 처리 우선 순위를 생각한다면 <부록 3>과 같은 최악의 경우도 생각해 볼 수 있다.

<부록 2>에서 사용한 '선수1' 스프라이트를 삭제하여 다시 만들고 다음과 같은 스크립트를 작성할 경우 결과는 예상외가 될 수 있다. 이렇게 만든 <부록 3>의 프로젝트 결과는 '선수1' 스프라이트만 결승점에 도착하게 된다. 그 이유를 추정해보면 다음과 같다.

조건) 스프라이트 등록 순서는 '선수2', '결승선', '선수1'이다.

처리1) ‘선수1’ 스프라이트가 가장 먼저 등록되었기 때문에 스크립트가 가장 먼저 실행된다. 이 때 모듈 2가 가장 최근에 작성된 것이라면 조건문(만약 결승선에 닿기라면)이 실행된다. 시작 위치에서 결승선을 만나지 않기 때문에 처리하지 않고 넘어간다. 모듈 1을 실행하여 앞으로 10만큼 움직이기를 수차례 수행한다. 우연히 결승선에서 멈추었다. 더 이상 처리할 스크립트가 없어서 다음 스프라이트로 처리 순서를 넘긴다.

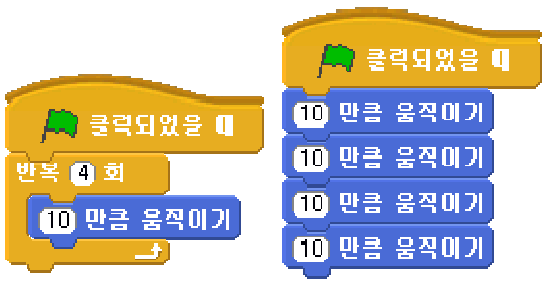
처리2) ‘결승선’의 스크립트가 실행된다. ‘선수1’과 ‘선수2’가 동시에 결승선에 닿지 않았으므로 처리하지 않고 다음 스프라이트로 처리 순서를 넘긴다.

처리3) ‘선수2’ 스프라이트는 반복문으로 앞으로 10만큼 1회 움직인다. 결승선에 닿지 않았으므로 처리하지 않고 다음 스프라이트로 처리 순서를 넘긴다.

처리4) ‘선수1’ 스프라이트의 모듈 2가 수행된다. 현재 ‘선수1’ 스프라이트는 결승선에 닿았기 때문에 조건문 내의 스크립트가 실행된다. ‘우승자’ 리스트에 ‘선수1’이라는 단어가 저장되고 프로젝트가 종료된다.

이러한 결과들이 시사하는 점은 다음과 같다.

- 강의자는 스크립트의 처리순서에 대해 알고 있어야 한다. 우선순위를 암기하여 알아둘 필요는 없지만, 이러한 처리순서의 개념이 존재한다는 것을 알아둘 필요가 있다. 아래의 두 스크립트의 의미는 같다고 할 수 있지만 실제 처리순서는 다르다. 반복문은 1회의 반복 때마다 처리 순서를 다음 스크립트 모듈 또는 다른 스프라이트로 넘겨주지만 순차적 명령의 연속인 경우는 한번에 모두 처리된다.



[그림 Ⅱ-14] 의미는 서로 같으나,
처리 결과가 다른 두 스크립트 모듈

· 전체 프로그램의 원활한 병행 처리를 위해 각 모듈의 크기의 균형적 조정할 필요가 있다. 원활한 병행 처리를 위해서는 반복문의 구조화가 스크립트 간 균형적으로 조정되어야 할 필요가 있다는 것이다.

· 공유자원 접근 처리를 위한 원자적 연산(atomic operation)이 가능하다.
  - 순차적으로 연결된 블록은 한 번에 모두 처리된다는 점을 이용하여 공유자원 접근 처리를 충돌 없이 구현할 수 있다.

4. PPS기반 컴퓨터과학 학습

1) PPS(Paper-and-pencil Programming Strategy)의 개념

컴퓨터과학은 우리의 일상과 사회 전반에 대한 혁신을 이끌고 있음에는 누구도 부정하지 못하고 있음에도 이 학문 자체에 대한 대중의 관심은 계속 줄고 있다(Carter, 2006; Denning & NcGettrick, 2005; Nwana, 1997). 많은 연구자들은 이러한 주된 이유를 '프로그래밍이 컴퓨팅의 전부이다'라는 식의 전통적인 교육적 접근과 교수학습 방법에 있다고 얘기하고 있다(Deek, Kimmel, & McHugh, 1998; Foster, 2005; Patterson, 2005; Zendler & Klaudt, 2012). 대중과 학생들의 관심을 다시 얻고자 많은 교육방법과 전략이 생겨났다. 예를 들어 언플러그드 학습(Bell, Alexander, Freeman, & Grimley, 2009; Taub, Armoni, & Ben-Ari, 2012), 피지컬 프로그래밍(Montemayor et al., 2002), 로봇 프로그래밍(Flowers & Gossett, 2002), 문제/퍼즐/게임 기반 학습(Greitzer, Kuchar, & Huston, 2007; Koschmann, Feltovich, Myers, & Barrows, 1992; Merrick, 2010)으로 컴퓨터과학의 핵심을 가르치려는 시도가 있었다.

이러한 접근 방법 자체도 하나의 방법이나 전략일 뿐 교육적 목표에 대한 비전을 동반하지는 않는다. 그러한 관점에서 계산적 사고는 교육적 패러다임을 바꿀 수 있는 측면을 제시하고 있다. 즉 컴퓨터과학 전공자가 아닌 비전공자들과 일반인, 학생들을 위해 컴퓨터과학을 가르칠 때 그들에게 프로그래밍이 아닌 그보다 핵심적인 계산적 사고를 키우게 하는 것이 더 우선이 되어야 한다

는 것이다. 프로그래밍 언어는 인간이 어떠한 문제를 해결해 나가기 위해 선택된 도구중 하나일 뿐 이 학문의 학습 목적이 될 수는 없을 것이다.

이러한 목적을 설정하고 난 후에는 다음과 같은 세부적인 사항들에 대한 고민이 필요하다.

질문 1) 학습자들이 계산적 사고의 개념을 이해하고 이를 직접적으로 활용하며 가르칠 수 있는 교육적 방법의 모델은 어떤 것인가?

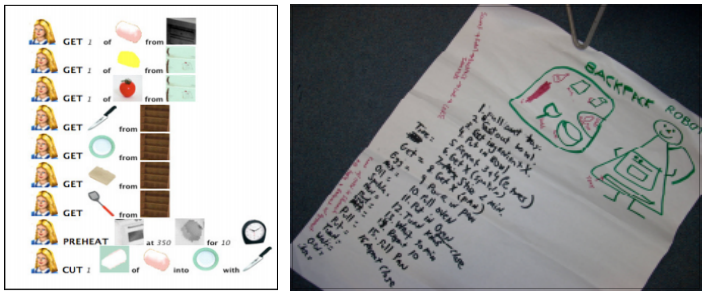
질문 2) 학습 내용으로 선정될 컴퓨터과학의 핵심은 무엇인가?

질문 3) 학습자들의 계산적 사고의 변화를 어떻게 측정할 것인가?

이러한 질문에 대한 답을 찾기 위해 먼저 Schwill(1994, 1997)의 3가지 컴퓨터과학의 핵심 아이디어와 Denning(2010)의 7개의 컴퓨팅 원리를 기반으로 하여 학습 내용을 선정하였다. 그리고 언플러그드 학습 방법의 일환으로 PPS(Paper-and-Pencil Programming Strategy) 교육방법을 개발하여 적용하였다. 또한 학습자들의 계산적 사고의 변화를 측정하기 위해 계산적 사고의 일부라고 할 수 있는 논리적 사고력을 사전·사후에 측정하고자 하였다. 또한 학습자들이 실제로 계산적 사고의 개념을 얼마나 이해하고 있으며 이러한 교육방법이 기존의 전통적인 프로그래밍 수업 방법에 비해 어떠한 효과를 갖는지를 알아보고자 하였다.

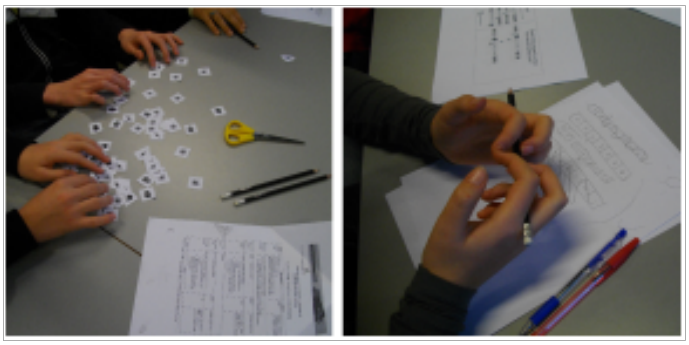
PPS는 “다이어그램, 문장, 코드, 순서도, 표 등의 의미 있는 기호와 상징을 이용하여 계산모델을 스스로 정의하고, 문제에 대한 해결 방법(solution)이나 아이디어를 추상화하여 종이에 직접 쓰거나 그리면서 논리적으로 표현하고자 하는 문제 해결의 방법을 모델링하는 전략”을 의미한다.

PPS가 언플러그드식의 방법을 추구한다는 측면에서는 전혀 새로운 것은 아니다. 예를 들자면 이전에도 이러한 언플러그드식의 학습 방법은 많이 연구되어져 왔다. 예를 들어 Tarkan 외(2010)의 연구에서는 요리를 하는 과정을 하나의 절차로 문장을 이용하여 프로그램화 하도록 하는 방법을 Toque라 소개하고 그 활동들을 제시하였다.



[그림 Ⅱ-15] 아이콘 기반의 Toque 언어(좌), 아이디어 디자인(우) (Tarkan et al., 2010)

또한 Valente와 Marchetti(2011)의 연구에서도 튜링머신을 만들기 위해 제작한 종이 템플릿을 제공하고 직접 종이와 연필로 튜링머신을 제작하도록 하기도 하였다.

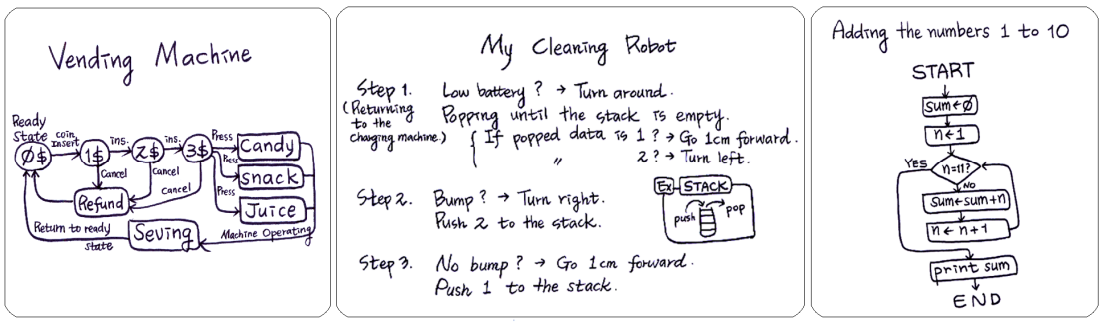


[그림 Ⅱ-16] TM(튜링머신)의 제작(좌), 계산 과정 추론(우) (Valente & Marchetti, 2011)

이러한 기존의 전략과 PPS와의 차이점은 문제의 유형이나 계산모델이 정해져 있지 않다는 점이다. 위 연구에서 Toque는 문제의 유형이 음식만들기에 국한되었다는 점, 튜링머신은 계산모델이 이미 정해져 있기 때문에 문제유형에도 제한이 있다는 점이 PPS와는 다르다.

오히려 PPS는 이러한 교육 방법들을 개발하는 초기단계의 아이디어와 비슷하다고 할 수 있다. 즉 어떠한 문제가 주어졌을 때 그 문제를 해결하기 위한 가장 효율적인 계산모델을 직접 정의하고 문제를 추상화하여 계산으로 나타내며 해결책을 모델링하고자 하는 방법이다.

예를 들어, 문제 상황이 '카레를 만드는 방법'이라면 PPS는 다양한 모델을 제시해 줄 수 있다. 그것은 일련의 문장들이 될 수도 있고 다이어그램이나 순서도가 될 수도 있으며 학습자 스스로가 만든 전혀 새로운 것일 수도 있다. 즉, 그 표현 방법은 무엇이건 상관이 없다. 그것보다 중요한 것은 그것을 보고 다른 이에게 내가 설명하고자 하는 카레 요리법을 정확히 전달할 수 있느냐는 것이다. 이러한 설계의 단계를 추상화라고 한다면 요리법을 보고 요리를 하는 수행 과정을 자동화라고 설명할 수 있다.



[그림 Ⅱ-17] PPS를 활용한 다양한 계산모델

2) PPS의 특징

PPS가 PBL(Problem-based learning)이나 프로그래밍 언어 교육, 순서도를 활용한 교육, 브레인스토밍 기법 등과 구별될 수 있는 차이점에는 다음과 같은 것들이 있다.

- 문제를 해결하는 것 뿐만 아니라 추상화와 자동화를 활용하여 그 해결 방법을 논리적이고 효율적으로 표현하고자 하는 것을 배우는 학습이다.
- 프로그래밍 언어 자체에 대한 학습이 아니라 시스템(문제 해결을 위한 전체 조직)과 그 시스템의 사용 언어를 설계한다.
- 브레인 스토밍과 같은 발산적 사고를 요구함과 동시에 자신의 알고리즘이나 솔루션의 디버깅을 위한 수렴적 사고를 요구한다.
- 자연과 일상의 존재하는 문제에 대하여도 탐색하고 표현하여 다루어질 수 있다.
- 컴퓨팅 머신 없이도 종이와 연필을 활용하여 컴퓨터과학자처럼 창의적

으로 사고하는 방법을 학습할 수 있다.

3) 연구의 적용과 결과

비전공자들을 위해 PPS기반의 계산적 사고 중심 컴퓨터과학 학습을 설계하여 적용해 보았다.

(1) 참여자

연구 참여자들은 제주에 위치한 대학교 132명의 2학년 학생들이며 이들 모두가 컴퓨터과학 교과를 교양과목으로 필수 이수하여야 했다. 이 들 중 9명의 학생들만이 계산적 사고에 대해 들어봤으며 11명의 학생만이 초·중등 교육과정 상에서 플래시 액션스크립트, 비주얼 베이직 또는 로고와 같은 프로그래밍 학습을 받아본 적이 있는 비전공자들이었다.

(2) 연구기간

한 학기 수업에서 총 15주의 강의가 본 연구에서 개발한 프로그램대로 진행되었으며 한 강의는 50분 수업 2차시로 이루어진다. 결석없이 학습에 참여한 PPS반 55명(3개반)과 Logo반 55명(3개반)에 대해서만 사전·사후의 논리적 사고력 GALT 결과, 계산적 사고의 이해도와 컴퓨터과학의 흥미도에 대한 설문결과를 활용하였다.

(3) 학습 내용의 선정

Denning(2010)은 1990년대 이전까지도 많은 컴퓨터 과학자들이 이 분야의 알고리즘, 데이터구조, 운영체제, 네트워크, 데이터베이스, 그래픽, 인공지능, 소프트웨어 공학 등을 중심으로 컴퓨터과학을 정의한 것에 대해 기술적 해석이라고 생각하고 이러한 기술적 측면을 제한한 과학적 해석으로 컴퓨터과학 내부의 핵심적인 원리들을 강조하고자 하였다. 이는 계산, 통신, 협력, 기억, 자동화, 평가, 설계이다. 이러한 원리 중심의 학습은 그 내용이 세부적이며 수준이 높은 편이라 전공자 위주의 학습이 될 수 있다.

Schwill(1994, 1997)의 아이디어 중심 접근 방법에서 컴퓨터과학의 핵심을 선정하기 위한 4가지 기준을 정하였다. 이는 수평적 기준, 수직적 기준, 시간적 기준, 감각적 기준이다. 이러한 기준은 쉽게 다음과 같이 풀이할 수 있다. '컴퓨터과학 이외의 많은 타 영역에도 다양한 방법으로 관찰, 적용될 수 있는가?'

(수평적 기준), '지적 수준이 다양한 학생들에게 가르칠 수 있는 내용인가?'(수직적 기준), '컴퓨터과학의 발전사에 지속적으로 관찰되었던 핵심 내용인가?'(시간적 기준), '이론적이기 보다는 일상 생활에서도 의미를 가질 수 있는 내용인가?'(감각적 기준)가 그것이다. 이러한 기준에 의한 핵심 아이디어를 3가지로 선별하였는데 이는 알고리즘화(algorithmization), 구조적 해체(dissection), 언어(language)이다.

Schwill의 3가지 핵심 아이디어를 학습 내용으로 선정하기에는 추상적이며 세부적이지 못하다. 이러한 아이디어를 학습 내용으로 세부화하기 위해 소프트웨어의 기본 제어 구조(BCSs: Basic control structures)와 프로그래밍에서 사용되는 컴퓨터과학의 개념을 추출하였다. 이렇게 선정된 내용을 실험집단에는 PPS로, 비교집단에는 Logo 프로그래밍 학습으로 나누어 같은 주제이지만 해당 전략에 맞추어 세부 내용을 일부 수정하였다.

<표 Ⅱ-17> PPS 활용 컴퓨터과학 학습의 내용

주	PPS 학습반 주제	Logo 학습반 주제
1~2	· 계산적 사고의 개념 소개 · 문제해결 과정에서의 프로그래밍 - 추상화와 자동화의 의미	
3~6	· PPS 소개 - 계산 모델 소개 - 변수, 순차적 처리, 반복, 조건문 - 튜링머신 제작 활동 - 다양한 문제에서의 PPS 활용	· Logo 프로그래밍 소개 - 변수, 순차적 처리, 반복, 조건문 - 인터페이스 소개, 대화형 프로그래밍 작성 · 도형 그리기 · 프로시저 - 기능적 추상화를 위한 프로시저 사용
7~11	· 수학적 사고 요구의 문제 해결 - 1~10까지의 합 구하기, 수열, 최대값 찾기, 구구단 출력 - 정렬, 소수구하기, 최대 (다양한 형태의 flowchart 제시)	· 수학적 사고 요구의 문제 해결 - 1~10까지의 합 구하기, 수열, 최대값 찾기, 구구단 출력 - 자료구조의 이용 - 정렬, 소수구하기, 최대
12~15	· 일상의 문제들의 접근 - 알고리즘을 활용한 일상의 문제를 PPS를 적용하여 해결	· 알고리즘 활용 문제 - 퍼즐관련 문제 - 재귀호출을 이용한 도형 그리기

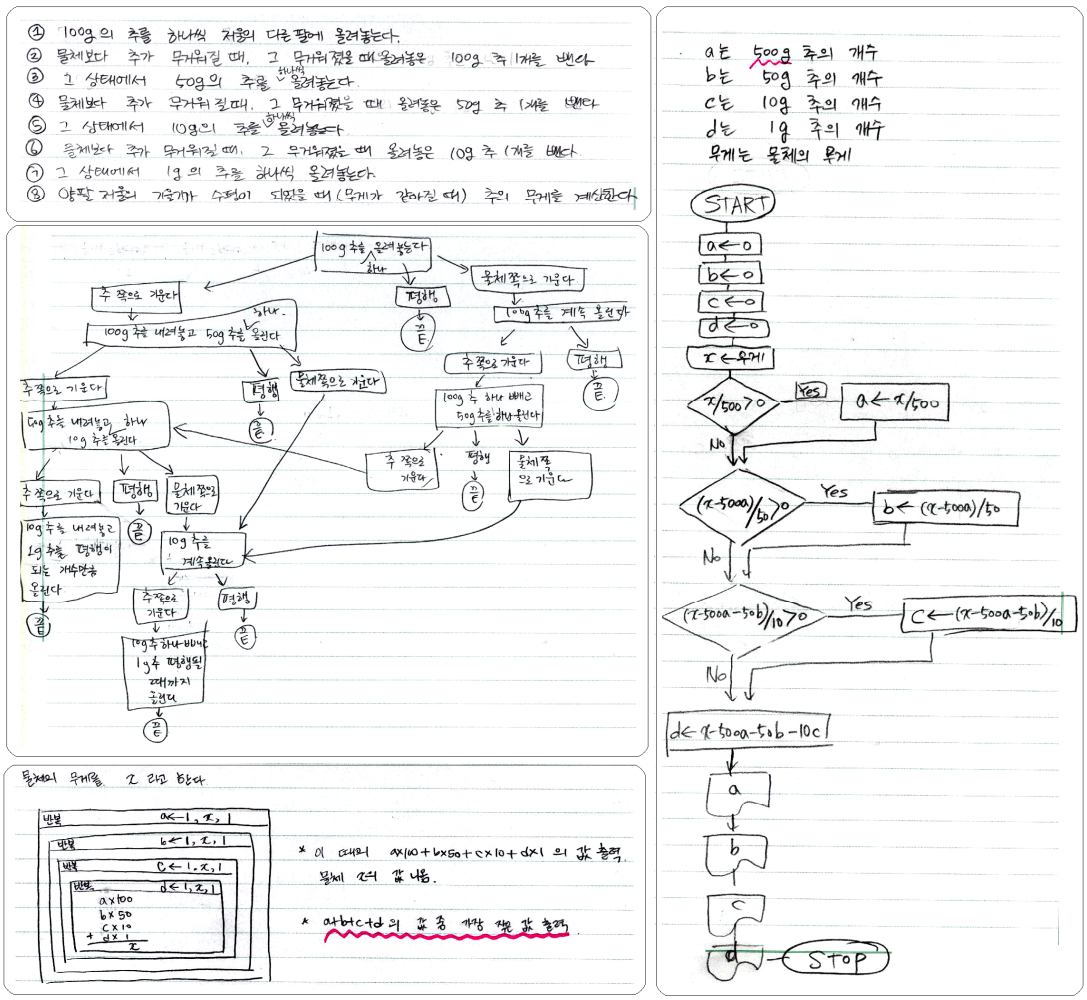
(4) 학습의 실제

PPS 학습의 초반에 튜링머신에 대한 개념을 소개하고 관련된 문제들을 해결하여 추상화와 자동화에 대한 개념을 익히도록 하였다. <부록 4>와 같이 몇 가지 조건을 요구하는 튜링머신을 제작하는 활동을 통해 많은 학생들은 자신만의 솔루션을 개발하였다. 하지만 많은 학생들 대부분이 최소 1개 이상의 오류를 갖고 있거나 요구조건을 모두 만족하지는 못하는 솔루션을 개발하였다. 일부분의 학생들만이 요구조건을 만족하는 자신만의 솔루션을 개발하였다.

[그림 Ⅱ-18] 같은 문제에 대한 학생들의 서로 다른 해결책

오답들을 제외한 학생들의 솔루션 중 요구조건을 만족하는 해답 중에 똑같은 것은 한 쌍도 없었다. 같은 아이디어라고 하더라도 세부적인 코드가 다르거나 더 효율적인 방법이 존재했다. 처음 이러한 활동을 접해본 학생들의 반응은 다양했으며 몇몇의 학생들은 매우 흥미롭다는 반응을 보이기도 했다.

PPS의 학습의 마지막 몇 시수는 일상생활의 문제에 대해 다루었다. 예를 들어 양팔(접시)저울의 오른쪽에 물건을 올려 무게를 재려는 방법에 대한 솔루션을 디자인하기로 하였다. 참고로 1g, 10g, 50g, 100g의 분동(추)이 무한개 존재하며 물건의 무게는 양의 정수라고 가정한다.



[그림 Ⅱ-19] '저울로 무게를 재는 방법'이라는 문제에 대한 다양한 해답 설계

이에 대한 학생들의 솔루션은 다양했다. 다양한 자신들만의 계산모델을 정의했고 알고리즘을 개발하였다. PPS 활동의 취지는 이러한 솔루션들들이 얼마나 오류가 없는 훌륭한 알고리즘인가를 측정하는 것이 아니라, 학습자가 자신만의 솔루션을 추상화를 통해 표현할 수 있다는 것과 자동화를 통해 이를 확인하고 디버깅하는 과정에 대한 학습의 기회를 주고자 하는 것이다. 궁극적으로 이를 통해 계산적 사고의 개념을 이해하고 활용할 수 있는 힘을 기르게 하려는 것이다.

(5) 학습 결과의 평가

계산적 사고에 대한 정의가 학계에서 확실히 내려지지 않은 것처럼 이에 대

한 측정 방법에 대한 연구는 미비하다. Wing(2006)은 계산적 사고의 구성 요인으로 알고리즘적 사고, 재귀적 사고, 비판적 사고를 제시하였다. 유중현과 김종혜(2008)는 Wing이 논리적 사고에 대해서 거론하지는 않았지만 계산적 사고는 예방, 보호, 과잉의 행동 및 최악의 상황으로부터의 보호, 에러 수정 등을 말하며 이러한 문제의 해결법을 발견하기 위해서 귀납적 추론을 사용한다고 하였는데 여기서 귀납적 추론이 논리적 사고라고 볼 수 있다고 하였다. 또한 문제해결을 실패하였을 경우 잘못된 부분을 찾아 그 부분부터 새로운 해결책을 제시하는 과정도 논리적 사고의 일부라 할 수 있고, 해결책을 발견한 후 해결책을 계속적으로 성공할 때까지 적용하는 재귀적 사고의 구조도 논리적 사고의 일부라고 볼 수 있다. 즉 계산적 사고에 논리적 사고도 하위 사고 양식임을 확인할 수 있다는 주장인 것이다.

추가적으로 CS4FN(http://www.cs4fn.com)은 21세기 살아갈 인간에게 매우 기본적인 사고방법으로 문제해결 능력, 비판적 사고능력을 배우는 것을 포함하였다. 이는 모든 사람에게 매우 필요한 능력이며 그러한 문제 해결 능력에는 계산적 사고가 필요한데 그것의 하위 개념으로 알고리즘적 사고, 논리적 사고, 재귀적 사고를 제시하였다.

이렇듯 계산적 사고에 대한 정의가 다양함에 따라 아직까지 계산적 사고를 측정할만한 일반화된 평가는 없다. 하지만 계산적 사고의 하위 개념인 논리적 사고력으로 계산적 사고의 증진을 알아보고자 한다.

① 논리적 사고력의 측정

가장 일반적이며 널리 사용되는 논리적 사고력 평가지로는 GALT(Group Assessment of Logical Thinking)가 있다. 이 평가는 Roadrangka, Yeany, Padilla(1983)가 개발한 선다형 지면평가로, 총 21문제로 구성되어 있으며 이는 4문제의 보존논리, 6문제의 비례논리, 4문제의 변인통제논리, 2문제의 확률논리, 2문제의 상관논리, 3문제의 조합논리로 분류된다. 각 문제는 2개의 하위 문제로 구성되며 첫 번째 문제는 현상의 결과를, 두 번째 문제는 그 결과에 대한 이유를 묻는 문제이며 두 문제를 모두 맞아야 해당 문제를 맞혔다고 보며 1점을 획득하게 된다. Roadangak 외(1983)의 연구자들은 Piaget와 Inhelder(1969)의 인지발달 단계에 맞추어 GALT 평가 후 0~8점은 초보적인 수준, 즉 감각

운동기나 전조작기, 또는 구체적 조작기의 시작단계라 할 수 있으며, 9~16점은 구체적 조작기와 형식적 조작기 중간 단계, 17~21점은 형식적 조작기에 이르렀다고 판단할 수 있다고 보았다.

② 계산적 사고의 이해도 및 컴퓨터과학의 흥미도 측정

본 연구에서 PPS를 통한 컴퓨터과학의 학습이 프로그래밍 언어를 이용한 학습보다 계산적 사고에 대한 이해도를 높일 수 있는지를 설문을 통해 알아보려고 했다. 또한 이러한 학습이 궁극적으로 비전공자인 학생들에게 컴퓨터과학이라는 학문에 대한 흥미도를 높였는지에 대한 설문도 진행하였다. 이러한 설문은 1점(매우 아니다)에서 5점(매우 그렇다)까지의 리커드 척도의 설문이며 학습의 사전·사후에 이루어졌다.

(6) 연구의 결과와 분석

① 논리적 사고력

사전·사후의 PPS반과 Logo반을 서로 집단간의 독립표본 t검정을 실시하였다. 이에 대한 결과는 <부록 5>에 첨부하였다. 사전·사후 PPS반과 Logo반의 논리적 사고력의 집단간 차이는 없었다. 이것으로 사전과 사후의 두 집단은 동질 집단임을 알 수 있다. 이 결과만으로는 집단 내에서의 어떠한 학습 효과가 있었는지는 알 수 없을 것이다. 다시 말하면, 두 집단 모두 연구 적용 전에 비해 논리적 사고력이 증진되었는지, 아니면 아무런 효과가 없었는지는 알 수 없다. 따라서 이러한 효과적 측면을 알아보기 위해 각 집단 내에서 대응표본 t검정을 실시하였고 그 결과는 <부록 6>과 같다.

집단 내 사전·사후에 대한 대응표본 t검정 결과를 보면 실제 집단 내에서의 논리적 사고력의 각 하위 요소에 대한 변화를 알 수 있다. Logo반의 경우에서는 상관, 조합을 포함하여 논리적 사고력 합계에 대해서 유의미한 증가를 보였고, PPS반에서는 보존, 비례, 상관을 포함하여 논리적 사고력 합계에 대해서 유의미한 증가를 보였다.

이러한 결과를 해석해본다면 두 집단 모두 몇몇 하위 요소와 논리적 사고력 합계에 대해 사전보다 유의미한 증가를 보였으며 어느 한 집단이 더 높은 증가를 보인 것은 아니다. 이를 다르게 해석해본다면 논리적 사고력을 높이는 데에는 Logo 프로그래밍 활동만큼 PPS 활동도 효과가 있었다는 의미이며, 그

목적이 계산적 사고의 증진이라면 PPS 활동 또한 일반적인 프로그래밍 활동만큼 가치가 있을 수 있다고 말할 수 있겠다.

② 계산적 사고의 이해도

계산적 사고의 이해도를 설문하기 위해 “나는 계산적 사고를 이해하고 있다.”, “프로그래밍에서의 계산적 사고를 설명할 수 있다.”라는 2가지 질문을 사전과 사후에 각각 이용하였다. 1점(매우 아니다)에서 5점(매우 그렇다)까지의 리커드 척도를 이용한 설문이며 이 두가지 질문에 대한 신뢰도로 Cronbach's α는 사전에 .84, 사후에 .85를 보였다. <부록 7>은 각 집단에서의 사전·사후의 계산적 사고의 이해도의 두 독립표본 t검증 결과를 보여주고 있다.

사전 설문에서 Logo반의 평균은 1.675, 편차는 .729이였으며 PPS반의 평균은 1.587, 편차는 .469였다. 이 두 집단에 대한 독립표본 t검정의 결과는 .825로 유의미한 차이는 없었다. 사후 설문에서 Logo반의 평균은 2.750, 편차는 .846였으며 PPS반의 평균은 3.232, 편차는 .793였다. 사후 결과에 대한 두 독립표본 t검정의 결과는 -3.336으로 유의확률 $p<.05$ 내에서 유의미한 차이를 보였다.

두 집단 모두 계산적 사고라는 주제를 통해 다양한 문제를 해결하는 활동을 하였다. 하지만 프로그래밍 언어를 직접 사용한 Logo반의 학생들보다 PPS 활동을 접한 학생들의 계산적 사고의 이해도가 유의미하게 높았다는 의미이다. 이는 PPS 활동이 계산적 사고의 개념과 이해를 돕는데 매우 효율적이라고 할 수 있겠다.

③ 컴퓨터과학의 흥미도

컴퓨터과학의 흥미도를 설문하기 위해 “나는 컴퓨터과학을 배우는데 흥미를 느낀다.”, “나는 컴퓨터의 원리를 배우는 것을 즐긴다.”라는 2가지 질문을 사전과 사후에 각각 이용하였다. 1점(매우 아니다)에서 5점(매우 그렇다)까지의 리커드 척도를 이용한 설문이며 이 두가지 질문에 대한 신뢰도로 Cronbach's α는 사전에 .84, 사후에 .81을 보였다. <부록 8>는 각 집단에서의 사전·사후의 컴퓨터과학의 흥미도의 두 독립표본 t검증 결과를 보여주고 있다.

사전 설문에서 Logo반의 평균은 3.125, 편차는 .734이였으며 PPS반의 평균은 3.130, 편차는 .793였다. 이 두 집단에 대한 독립표본 t검정의 결과는 -.040로 유의미한 차이는 없었다. 사후 설문에서 Logo반의 평균은 3.733, 편차는 .627였

으며 PPS반의 평균은 3.957, 편차는 .610였다. 사후 결과에 대한 두 독립표본 t 검정의 결과는 -2.044로 유의확률 $p<.05$ 내에서 유의미한 차이를 보였다.

Logo반과 PPS반 모두 사전에 비해 사후의 결과가 높게 나타났다. 하지만 PPS반 학생들의 사후 설문 결과가 유의미하게 높아진 이유에 대해 분석해 볼 필요가 있을 것이다. 이는 Logo와 같이 실제 프로그래밍 언어의 문법을 익히는 과정에서의 인지적 부담을 줄일 수 있다는 점과 PPS 활동이 수학적 사고를 요구하는 문제에서부터 일상 생활에 대한 문제까지의 해결 과정에서 컴퓨터과학이 쉽고 유용하다는 점이 학습자의 흥미도를 높였다고 분석된다.

III. PPS기반 프로그래밍 교육 프로그램의 개발

1. 학습모형의 선정

본 연구에서 개발하고자 하는 학습 모형의 기반을 CPS(Creative Problem Solving, 창의적 문제해결) 모형에서 찾고자 하였다.

CPS는 Creative Problem Solving의 약자로 '창의적 문제해결' 또는 '창의적 문제해결 접근법'을 의미한다. 현대의 많은 학자들이 창의성을 '창의적 문제해결 과정'으로 바라보며 문제해결과 창의성을 함께 논의하고 있다. 일부 학자들은 창의성을 문제를 해결하는 과정에서 새롭게 나타나는 능력이라고 정의하고 있으며 대부분의 문제 해결에 관한 모델들은 문제해결이 창의성과 밀접과 관계가 있음을 보여주고 있다.

CPS 모형은 Wallas가 창의성을 사고과정으로 이해하고 창의적 사고를 준비, 부화, 조명, 검증의 4단계로 이뤄진다고 한 것에서부터 시작되어 여러 학자들에 의해 연구되어져 왔다(김나라, 2012).

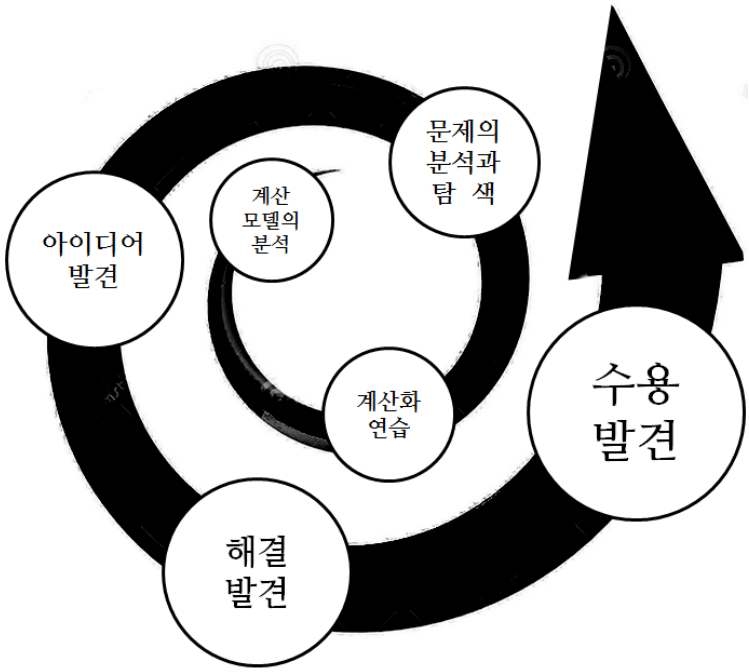
창의적 문제해결 과정의 단계에 대한 연구 중에는 Hayes의 '완전한 문제해결자'와 Bransford와 Stein의 IDEAL이 있으며(Brandsford & Stein, 1984; Hayes, 1981), 관련 연구들 가운데 가장 대표적인 것이 Osborn과 Parnes의 CPS이다(Osborn, 1957; Parnes, 1967). Osborn과 Parnes의 CPS 모형의 5단계는 사실발견, 문제발견, 아이디어 발견, 해결발견, 수용발견으로 구성되어 있다.

<표 Ⅲ-1> Osborn과 Parnes의 CPS 5단계

단 계	내 용
사실 발견	문제를 파악하기 위한 정보와 자료를 수집하고 분석하는 단계로 객관적이고 정확한 역할이 필요하며 문제 상황에 대한 사실을 찾아야 한다.
문제 발견	다양한 관점으로 문제를 재정의하는 단계로, 해결하고자 하는 문제의 범위 안에서 핵심 문제 요소를 찾아 이를 간결하게 만든다.

단 계	내 용
아이디어 발견	문제를 해결할 수 있는 다양한 대안들을 생각해 내는 단계로 가능한 한 많은 아이디어를 산출할 수 있도록 확산적 사고가 발휘될 수 있도록 돕는다.
해결 발견	아이디어 발견 단계에서 산출된 다양한 아이디어 중에서 최적의 아이디어를 고를 수 있도록 비교하고 평가한다.
수용 발견	최적의 아이디어를 처음의 문제 상황에 적용하고, 실제 적용 시에 발생되는 방해 요인이나 제한점들에 대한 극복 방안도 함께 구안한다.

본 연구에서 개발하는 학습 프로그램 또한 이러한 CPS 모형에 맞게 창의적 문제해결을 요구하는 문제를 제시하고 이를 해결해나가는 과정에서 계산적 사고력과 창의성을 증진시켜 나가고자 하여 다음과 같이 수정된 CPS 6단계를 사용하였다. 수정된 CPS 6단계에서의 학습의 절차는 Bruner(1966)가 제시한 나선형 교육과정에 따라 활동이 진행된다.



[그림 Ⅲ-1] 나선형 교육과정에 따른 프로그래밍 학습 단계

나선형 교육과정은 학습자의 지식수준과 교육 목적에 따라 지식이 점점 크

게 돌아 나오고 순환하여 그 지식의 폭과 심도가 더해지는 모양을 나타낸 것이다. 따라서 (A+B)+(A+B+C)+(B+C+D)+(A+D)+…N과 같이 나타낼 수 있다. 다시 말해 간단한 것에서 복잡한 것, 자주 쓰이는 것도 더 연습하는 것, 어려운 것을 배우기 전에 관련된 쉬운 것을 복습하는 것 순으로 교육 내용을 조직하는 것이 나선형 교육과정이다(김재희, 2008). 이에 따라 본 연구에서 개발한 PPS기반 프로그래밍 교육 프로그램의 단계는 <표 Ⅲ-2>와 같다.

<표 Ⅲ-2> PPS기반 프로그래밍 교육 프로그램 단계

단계명	핵심 활동	특 징	기대 효과
계산모델의 분석	프로그래밍 언어 학습	학습자는 사용할 프로그래밍 언어 중 학습에 필요한 문법과 기능을 학습한다. 이러한 학습은 계산모델의 속성을 익히게 되고 계산모델을 분석적으로 이해할 수 있게 한다.	비판적, 논리적 사고의 활용
계산화	계산화 문제(소문제)를 통한 계산화 연습	계산화 단계에서는 언플러그드식의 학습으로 프로그래밍 언어와 비슷한 계산모델을 정의하여 이용하거나, 직접 프로그래밍 언어를 사용한다. 최종적으로 해결하려는 문제보다 상대적으로 작은 크기의 문제를 PPS로 해결하는 연습을 통해 프로그래밍의 핵심 개념과 처리과정을 학습하게 된다. 이 단계는 문제의 크기가 너무 작거나, 간단히 해결될 수 있는 문제에 대해서는 필요에 따라 생략되어질 수 있으며 계산모델의 분석보다 이전에 제시될 수도 있다.	계산적 인지력의 활용, 확산적 사고를 통한 창의성 발현
문제의 분석·탐색 및 아이디어 발견	프로그래밍 문제의 분 석	다양한 관점으로 문제를 재정의하는 단계로, 해결하고자 하는 문제의 범위 안에서 핵심 문제 요소를 찾아 이를 간결하게 만든다.	계산적 사고의 활용
	다양한 아이디어 고안	제시된 문제를 해결할 수 있는 다양한 대안들을 생각해 내는 단계로 가능한 한 많은 아이디어를 산출할 수 있도록 확산적 사고를 유도한다. 또한 이 아이디어를 PPS를 통하여 설계하여 표현해본다.	계산적 인지력의 활용, 확산적 사고를 통한 창의성 발현

단계명	핵심 활동	특 징	기대 효과
해결 발견	최선 또는 최적의 해결책 선택	아이디어 탐색에서 산출된 다양한 대안 중에서 최적의 해결책을 고를 수 있도록 실제 프로그래밍을 통해 비교하고 평가한다. PPS로 표현한 설계를 실제 프로그래밍을 통해 구현 후 비교하며 최선 또는 최적의 해결책을 선택한다.	논리적 사고, 알고리즘적 사고, 재귀적 사고 중심의 활용
수용 발견	해결책 구현, 적용	최적의 해결책을 처음의 문제 상황에 적용하고, 실제 적용 시에 발생되는 방해 요인이나 제한점들에 대한 극복 방안도 함께 구안한다.	총체적 계산적 사고의 활용

‘계산모델의 분석’ 단계에서는 문제를 해결하기 위한 최소한의 프로그래밍 언어의 문법과 기능을 학습하게 된다. 학습자는 문제를 해결하기 위한 도구로써 프로그래밍 언어 시스템을 계산 모델로 인식하고 이를 분석적으로 이해할 수 있게 된다.

‘계산화’ 단계에서는 컴퓨팅 머신 상에서 프로그래밍 언어를 직접적으로 사용하지 않고, 언플러그드 형식의 학습이 이루어진다. 해당 프로그래밍 언어와 비슷한 유형의 계산모델을 정의하고 실제 전체 학습에서 해결하고자 하는 문제보다 작은 크기의 문제를 제시하게 된다. 학습자는 이 문제를 PPS로 자신의 해결책을 모델링하여 표현한다. 이 단계는 필요에 따라 ‘계산모델의 분석’ 단계 전에 제시될 수도 있다.

기존의 CPS 모형에서 나뉘어졌던 ‘문제 발견’, ‘아이디어 발견’을 ‘문제의 분석·탐색과 아이디어 발견’ 단계로 통합하고, 이 단계에서는 문제를 다양한 관점으로 재정의하며, 해결할 수 있는 다양한 대안들을 생각해 내는 단계이다. 이러한 아이디어는 PPS를 통해 해결책을 모델링하여 표현하여 둔다.

‘해결 발견’ 단계에서는 아이디어 발견 단계에서 산출된 다양한 대안 중에서 최적의 해결책을 고를 수 있도록 실제 프로그래밍을 통해 비교하여 평가한다. PPS로 표현한 설계를 실제 프로그래밍을 통해 구현해보며 다른 해결책과의 장·단점을 파악해보며 최선 또는 최적의 해결책을 선택한다.

‘수용 발견’ 단계에서는 최적의 해결책을 처음의 문제 상황에 적용하고, 실제

적용 시에 발생되는 방해 요인이나 제한점들에 대한 극복 방안도 함께 구안하게 된다. 즉 모델링 되었던 해결책의 설계가 실제 코드(code)나 스크립트(script)로 바뀔 때 프로그래밍 언어 자체가 가지는 특징이나 제한점, 자료구조나 데이터 처리 방법에 의한 제한점을 고려하며 프로그래밍 하게 된다는 의미이다.

2. 학습 내용의 구성

본 연구에서 개발하고자 하는 학습 프로그램의 목적은 언어 자체에 대한 습득이 아닌, 컴퓨터과학의 핵심 내용들을 학습하며 창의성 신장을 할 수 있도록 하는 것이다. 대다수의 교재와 많은 스크래치 언어 학습 프로그램의 학습 순서를 보자면, 몇 차시의 초반 수업에서 블록들의 종류와 기능을 모두 익히고 이를 토대로 중·후반의 수업에서는 좀 더 어려운 프로젝트를 수행하는 것이 일반적인 접근이다. 본 연구에서는 학습 내용 또한 나선형 교육과정 방법에 맞추어 쉽고 간단한 프로그래밍 언어의 문법이나 기능만을 가지고 해결할 수 있는 문제를 통해 계산적 사고력과 창의성을 증대하고자 했다.

본 연구에서는 스크래치 프로그래밍 언어를 이용하여 구체적인 학습 내용을 구성하도록 했다. 하지만 본 연구에서 개발되는 교육 프로그램이 특정한 프로그래밍 언어에만 부합하는 것은 아니며 하나의 모델적인 구성이다.

<표 Ⅲ-3> PPS기반 프로그래밍 학습의 순서

순서	프로그래밍 핵심 개념	교재 구성	스크래치에서 학습할 핵심 블록
1	순차적 구조	1장	동작 블록
2	반복	2장	펜블록, 제어블록
3	조건, 분기		관찰블록, 형태블록
4	변수, 난수	3장	데이터블록, 연산블록
5	동시성과 병렬처리	4장	이벤트 '방송하기' 블록
6	객체, 시뮬레이션		스프라이트 변수
7	함수, 재귀		이벤트 '방송하기' 블록

3. 학습의 설계

본 연구에서 수정된 5단계의 CPS 모형을 1세트라고 한다면, 1세트 내에서 학습하게 되는 활동은 크게 '계산모델 학습', '계산화 연습', '프로그래밍 문제해결'로 분류될 수 있다. '계산모델의 분석'과 '계산화' 단계의 순서는 전·후 관계가 바뀔 수 있다. 또한 '문제의 분석·탐색' 단계에서부터 '수용 발견' 단계는 프로그래밍 문제해결 활동을 통해 이루어지게 된다. 이 때 PPS의 활용은 계산화 연습 활동과 프로그래밍 문제해결 활동에서 적극적으로 활용된다.

<표 Ⅲ-4> 수업 모형의 단계에 따른 활동과 PPS 활용

수업 모형의 단계	활동명 (교재)	PPS 활용
계산모델의 분석	계산모델 학습	-
계산화	계산화 연습	○
문제의 분석· 탐색	프로그래밍 문제해결	○
아이디어 발견		
해결 발견		
수용 발견		

PPS를 활용한 이러한 수업모형과 활동을 기반으로 개발된 교육 프로그램의 세부 내용은 <표 Ⅲ-5>와 같다.

<표 Ⅲ-5> 교육 프로그램의 세부 내용

순서	차시	활동명 (교재)	세부 내용
1장. 동작과 순서	(2)	계산화 연습	• 생활에서 알아보기 (짜장면 배달의 달인) • 창의적으로 생각하기 #1, #2, #3
		계산모델 학습	• 스크래치 화면 구성 알기 • 무대와 좌표, 동작 블록 익히기
		프로그래밍 문제해결	• 버스 운전하기 #1 #2

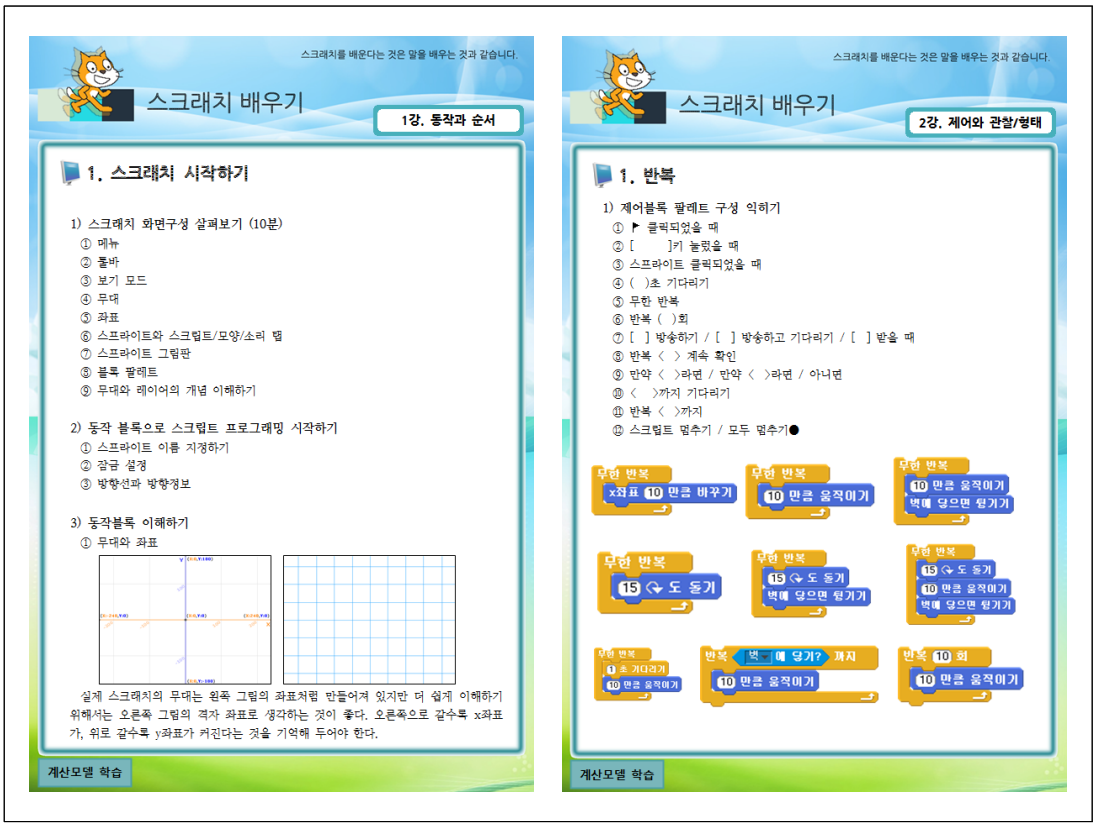
순서	차시	활동명 (교재)	세부 내용
	(1)	계산화 연습	• 당신의 컴퓨터 IQ는? (생각해보기 #1~#4)
		계산모델 학습	• 펜 블록 익히기
		프로그래밍 문제해결	• 한 붓 그리기 #1#2 • 우유 배달하기 • 외계인의 미스테리 서클 만들기
2장. 제어와 관찰, 형태	(3)	계산모델 학습	• 반복 개념 익히기
		계산화 연습	• 퀴즈1: Catch Me If You Can • 퀴즈2: 이어 달리기
		프로그래밍 문제해결	• 반복문의 위력 • 생각하고 생각하고 생각하라. 생각의 반복 • 반복의 반복의 반복의 반복의 반복
	(2)	계산모델 학습	• 조건, 관찰/형태 개념 익히기
		계산화 연습	• 퀴즈3: 숫자를 알아맞혀라. • 퀴즈4: 기하학 미디어 아트
3장. 변수, 연산과 동시 처리	(2)	계산모델 학습	• 변수의 개념 이해하기
		계산화 연습	• 퀴즈1: 계산기 디자인하기 • 퀴즈2: 변수 값 교환하기 • 퀴즈3: 3의 배수이면서 동시에 5의 배수인 수 • 퀴즈4: 3의 배수이거나 또는 5의 배수인 수 • 퀴즈5: 변수를 이용하여 도형 만들기
	(3)	계산모델 학습	• 난수의 개념 이해하기
		계산화 연습	• 퀴즈 6: 숫자 맞히기 열고개 게임 만들기
		프로그래밍 문제해결	• 인공지능 로봇의 시작 • 시계 만들기
	(1)	계산모델 학습	• 동시처리 때문에 일어나는 일들
		프로그래밍 문제해결	• 과녁 맞추기 게임 만들기
4장. 객체 지향, 인공 지능 그리고 시뮬	(4)	계산모델 학습	• 객체지향, 인공지능, 시뮬레이션의 개념 이해하기
		계산화 연습	• 퀴즈 1: 10번 말하기 반복하는 고양이
		프로그래밍 문제해결	• 땅을 파는 인공지능 지렁이 • 벌레의 움직임 • 청소로봇 #1, #2

순서	차시	활동명 (교재)	세부 내용
레이션	(2)	계산모델 학습	• 함수와 재귀의 의미 이해하기 • 이벤트의 '방송' 블록 익히기
		프로그래밍 문제해결	• 박테리아와 백신 • 감기의 전염
5장. 게임과 문제 해결	(4)	계산모델 학습	• 게임의 요소를 위한 기능 이해
		프로그래밍 문제해결	• 애플리케이션 만들기 • 양치기 소년 • 잠수함 슈팅 • 기억력 테스트 • 한자리 수 암산 • 점프 게임

실제 각 활동에서 제시되는 문제들이 프로그래밍의 핵심 개념을 어떻게 다루고 있는지를 보며, 각 활동을 통해 계산적 사고력과 창의성의 어떠한 구성요인을 증진시키려 하는지를 교재의 구성이나 실제 문제를 통해 살펴볼 필요가 있을 것이다. 실제 제작된 교재의 전체적인 구성은 <부록 9>를 참고하여 살펴볼 수 있다.

1) 계산모델 학습

'계산모델 학습' 활동에서는 해당 프로그래밍 언어에서 문제 해결과 관련된 문법과 기능을 학습하게 된다[그림 Ⅲ-2]. 실제로 프로그래밍 언어를 다루며 문법적인 규칙이나 기능들에 대해 학습하게 된다. 한꺼번에 너무 많은 양의 문법과 기능을 학습하게 되면 인지적 부담을 느끼므로 적당량을 선정하여 학습하는데, 본 연구에서 선정한 스크래치 프로그래밍 언어와 같은 경우는 하나의 블록 그룹에 속한 블록들을 중심으로 학습을 진행할 수 있도록 교재를 구성하였다. 교재의 전체적은 구성은 문법과 기능들에 대한 소개는 아니며, 학습 흐름에 대한 참고가 될 수 있도록 구성되었다.



[그림 Ⅲ-2] 계산모델 학습의 교재 구성

학습자가 각각의 기능들에 대한 확실한 이해가 필요하며 이를 위해 충분한 설명이 필요하다. 잘못 생성된 오개념은 이후의 프로그래밍 문제해결에서 빚어지는 오류를 예방할 수 있다.

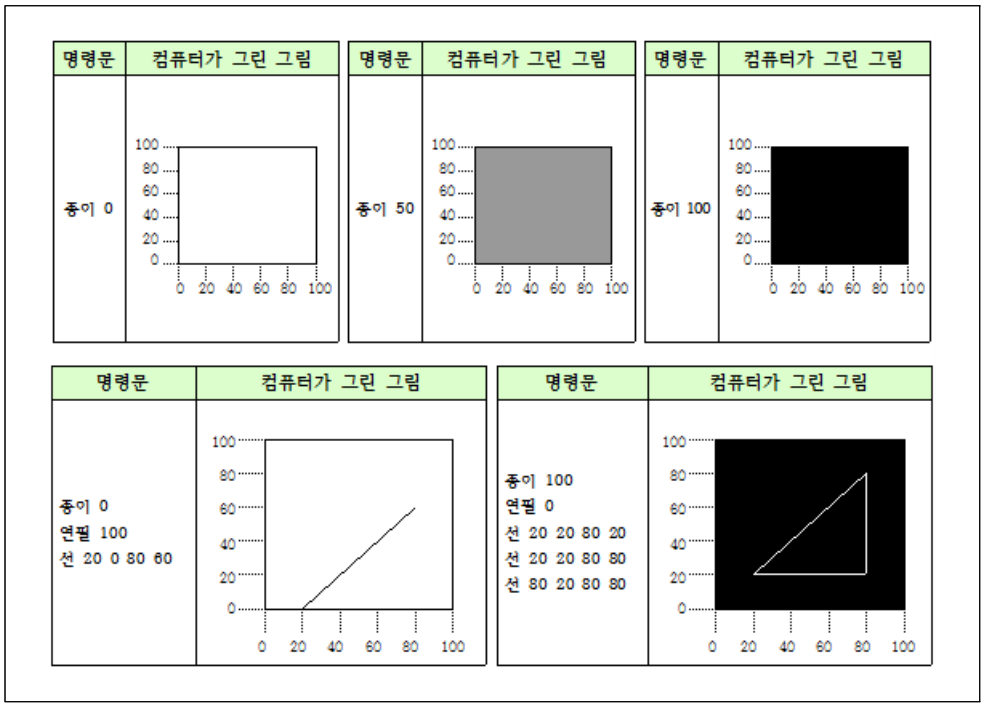
2) 계산화 연습

계산화 연습 활동은 말 그대로 '계산'을 만드는 연습을 하는 활동을 일컫는다. 이러한 계산은 주어지는 문제에서 학습자가 발견하거나 스스로 새롭게 만들어야 한다. 교재의 구성도 이러한 학습이 활발히 이루어질 수 있도록 학습지 형태로 구성되었다. 학습지에서는 PPS를 활용하여 자신의 해결책을 형식에 구애받지 않고 표현할 수 있도록 하였다.

이러한 계산화 연습은 필요에 따라 계산모델 학습 이전에도 이루어질 수 있다.

(1) 계산모델 학습 이전의 계산화 연습 활동

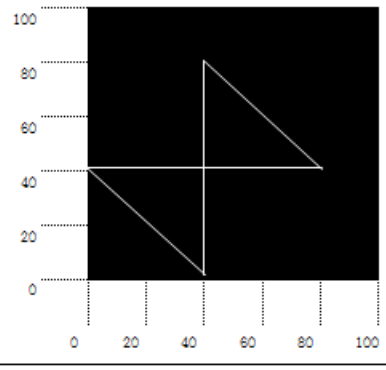
1장의 '동작과 순서'에서 좌표에 대한 개념과 좌표를 이용한 활동의 이해를 위해 계산화 연습이 계산모델 학습 활동인 펜 블록 학습보다 먼저 제시되었다[그림 Ⅲ-3]. 펜 블록과 관련된 계산화 연습에서는 좌표의 개념, 색칠하기 등으로 그래픽 적인 요소들에 대한 프로그래밍 감각을 익히기 위해서, 컴퓨터가 그림을 그리는데 명령어를 어떻게 처리하는지에 대한 규칙을 스스로 발견할 수 있도록 몇 가지 예시를 제시하였다.



[그림 Ⅲ-3] 계산화 학습의 예 (2장)

학습자는 이러한 예시를 통해 이러한 현상 안에 있는 '계산'을 스스로 발견할 수 있어야 한다. 이러한 '계산'에 대한 이해를 토대로 비슷한 유형의 문제에 대한 계산을 스스로 만들 수 있을 것이며, 이를 위해 PPS를 활용한 문제를 제시하였다[그림 Ⅲ-4].

3. 한 붓 그리기 방법으로 컴퓨터로 아래와 같은 도형을 그리려고 합니다.
명령문을 가장 적게 사용하여 빈칸에 써보세요.

명령문	컴퓨터가 그린 그림
	

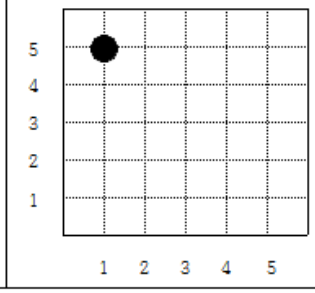
[그림 Ⅲ-4] 계산화 학습에서의 PPS 활동 (2장)

이와 같은 활동은 이후에 이루어질 계산모델의 학습에서 펜 블록의 기능과 문법적 요소를 이해하는데 훨씬 도움이 될 것이라고 기대한다.

이러한 학습은 단순히 문법 규칙을 암기하기 위한 선행 학습의 차원을 넘을 수 있다. 비쥬얼 프로그래밍 언어를 사용하는 학습자들의 성향을 경험적으로 보았을 때, learning by doing식의 학습이 흥미를 고취시킬 수는 있지만 실제적으로 깊은 사고를 필요로 하는 시간동안 그저 이것저것 해보는 것으로 시간을 보내고 만다. 즉, 이러한 학습은 계산모델의 학습 이전에 이미 핵심적으로 다룰 '계산'에 대한 개념 형성을 유도할 수 있기 때문에 이후의 학습에 매우 유용할 것이라고 기대할 수 있다. [그림 Ⅲ-5]와 같이 알고리즘적 사고가 필요한 계산화 연습도 사전에 이루어질 수 있다.

생각해보기 #4

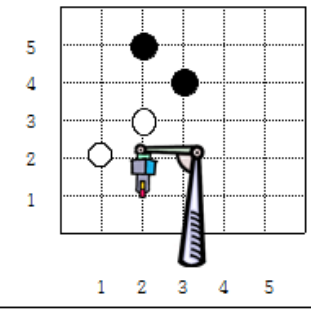
컴퓨터가 바둑을 둘 수 있게 만든 프로그램이 있다. 기본적인 명령어는 아래와 같다.

명령문	컴퓨터가 그린 그림
검 {1,5}	

생각해 봅시다.

3. 만약 우리가 만든 명령문에 따라 로봇팔이 바둑판 위를 공중으로 움직이며 바둑판에 바둑돌을 그려 넣는다고 상상해봅시다.

로봇팔이 움직이는 시간이 가장 짧게 아래의 바둑돌을 그려 넣는 명령문을 작성해 봅시다. (로봇팔의 현재 위치는 {2,1}입니다.)

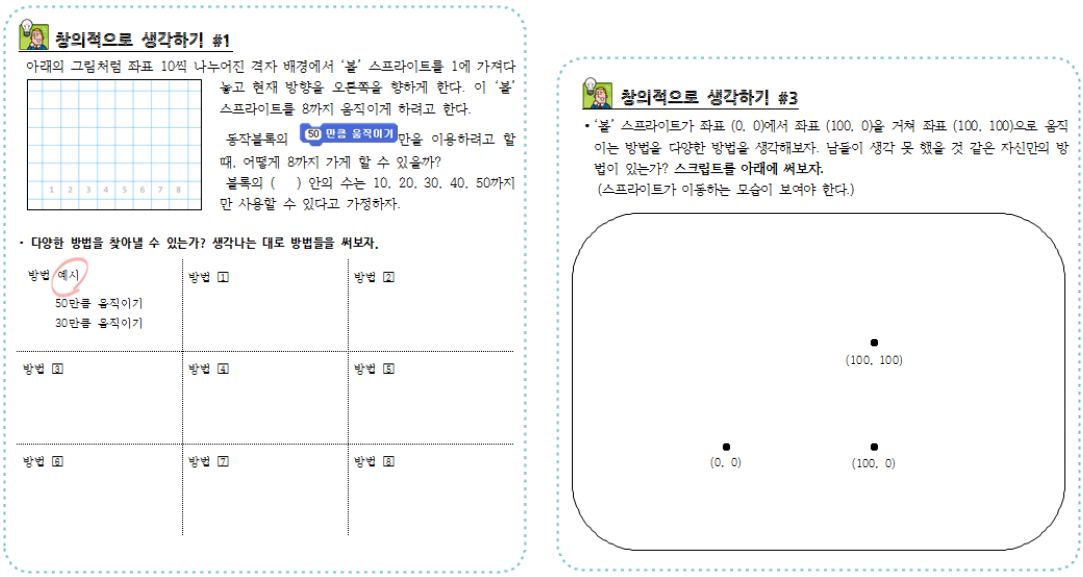
명령문	컴퓨터가 그린 그림
	

[그림 Ⅲ-5] 알고리즘적 사고를 요구하는 계산화 연습 활동(2장)

(2) 계산모델 학습 이후의 계산화 연습 활동

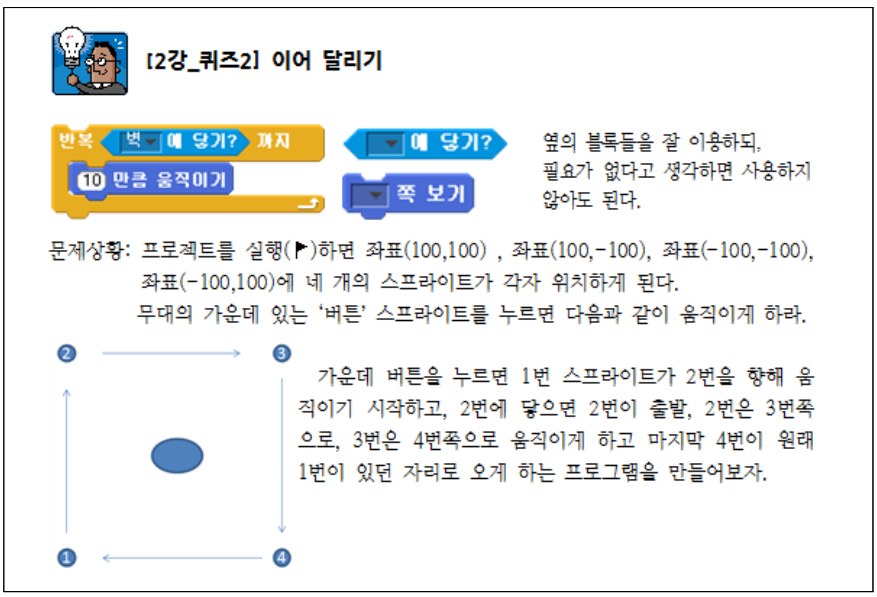
계산모델 학습 이후에 이루어지는 계산화 연습 활동의 목적은 계산모델에 대한 더욱 익숙해지기 위한 것이기도 하며, 한편으로는 그 익숙함에서 탈피하여 창의적 생각을 고취하고자 하는데 있다.

1장의 '동작과 순서'에서 계산모델의 학습 이후에 이루어지는 계산화 학습에는 창의성의 유창성과 독창성을 요구하는 활동이 포함되어 있다[그림 Ⅲ-6].



[그림 Ⅲ-6] 유창성과 독창성을 요구하는 계산화 연습 활동 (1장)

계산모델 학습 이후의 계산화 연습 활동은 교재에서는 퀴즈 형식으로도 문제를 제시하였다[그림 Ⅲ-7]. 이 문제는 다음에 이루어질 프로그래밍 문제해결 활동에서 다루어지는 상대적으로 큰 문제를 해결하기 위한 비교적 작은 문제로 다루어지며, 프로그래밍 활동으로 직접 문제를 해결할 수 있도록 하되 PPS를 활용하여 문제해결에 대한 설계를 할 수 있도록 하였다.

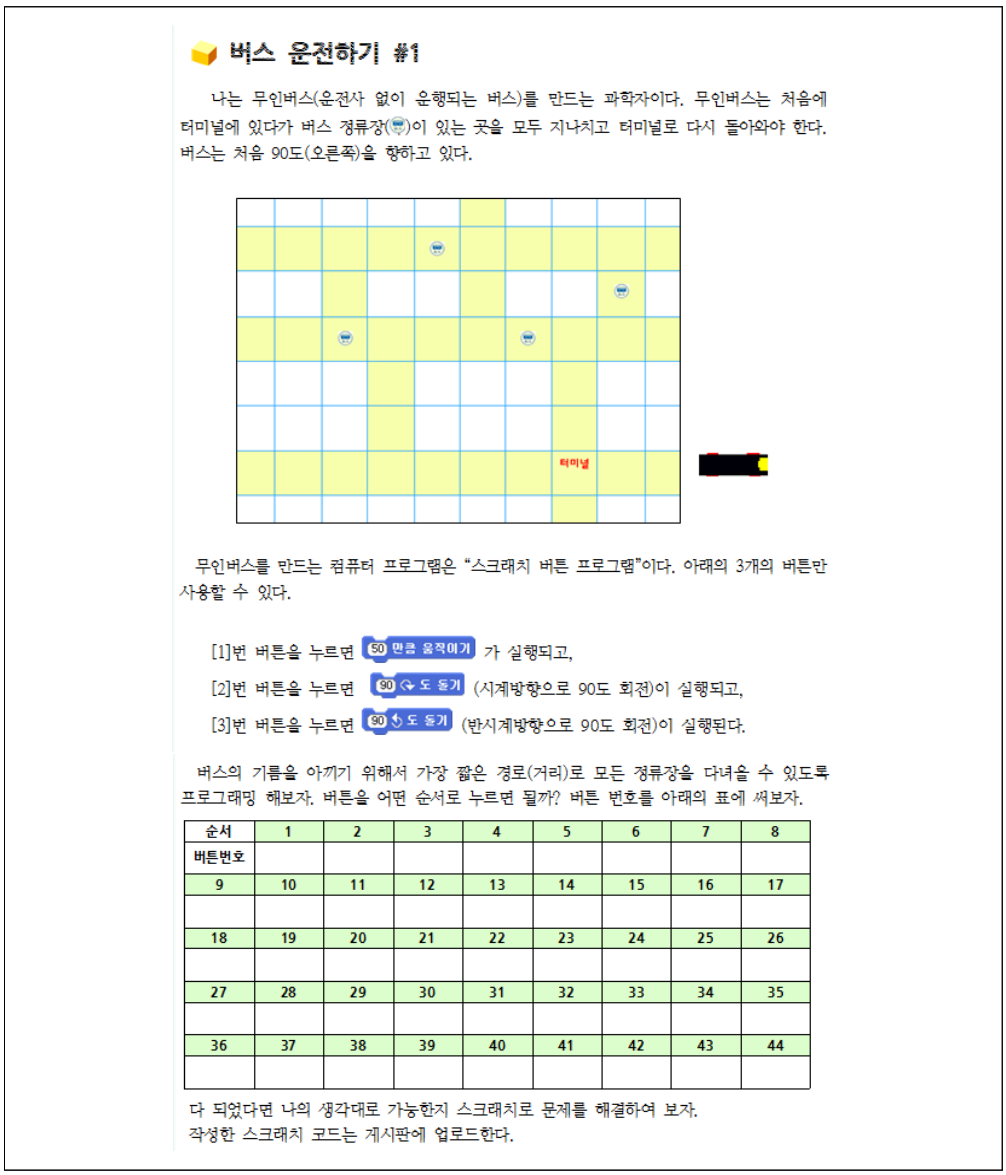


[그림 Ⅲ-7] 퀴즈형식의 계산화 연습 활동

3) 프로그래밍 문제해결

프로그래밍 문제해결은 각 장에서 배운 학습 내용을 토대로 가장 큰 프로래밍 문제에 대해 다루게 된다. 1장에서 5장으로 갈수록 그 수준은 높아져가며 이전 장에서 배운 내용이 기반이 되어야 해결할 수 있는 문제들이기도 하다.

1장의 '동작과 순서'에서는 3개 정도의 블록만을 사용하여 다음과 같이 알고리즘적 사고를 요구하는 문제를 제시하여 문제해결 학습 자체에 몰입할 수 있도록 학습지를 구성하였다[그림 Ⅲ-8].



버스 운전하기 #1

나는 무인버스(운전사 없이 운행되는 버스)를 만드는 과학자이다. 무인버스는 처음에 터미널에 있다가 버스 정류장(🚌)이 있는 곳을 모두 지나치고 터미널로 다시 돌아와야 한다. 버스는 처음 90도(오른쪽)을 향하고 있다.

무인버스를 만드는 컴퓨터 프로그램은 "스크래치 버튼 프로그램"이다. 아래의 3개의 버튼만 사용할 수 있다.

[1]번 버튼을 누르면 50 만큼 움직이기 가 실행되고,

[2]번 버튼을 누르면 90 도 돌기 (시계방향으로 90도 회전)이 실행되고,

[3]번 버튼을 누르면 90 도 돌기 (반시계방향으로 90도 회전)이 실행된다.

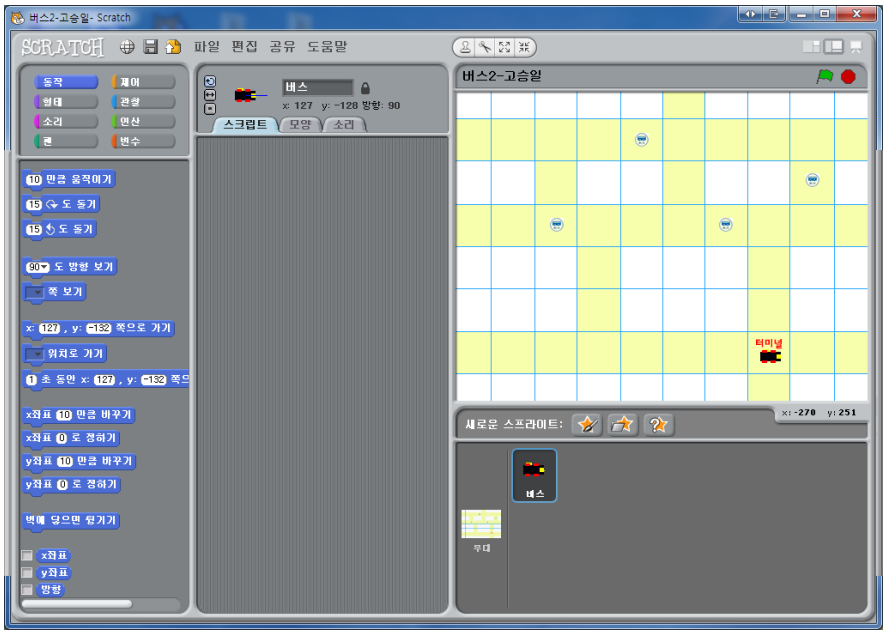
버스의 기름을 아끼기 위해서 가장 짧은 경로(거리)로 모든 정류장을 다녀올 수 있도록 프로그래밍 해보자. 버튼을 어떤 순서로 누르면 될까? 버튼 번호를 아래의 표에 써보자.

순서	1	2	3	4	5	6	7	8
버튼번호								
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44

다 되었다면 나의 생각대로 가능한지 스크래치로 문제를 해결하여 보자.
작성한 스크래치 코드는 게시판에 업로드한다.

[그림 Ⅲ-8] 알고리즘적 사고를 요구하는 프로그래밍 문제

언플러그드식의 학습지 활동이 끝난 후에는 학습지에 제시된 문제와 같은 문제가 그대로 스크래치 프로젝트로 제시된다[그림 Ⅲ-9]. 사실 '계산'에 대한 문제가 해결되며 프로그래밍 언어가 무엇이건간에 그 계산을 코드(code)로 옮겨 놓는 작업은 기계적인 일이다. 앞서 예제로 든 문제 또한 자신의 해결책을 코드로 옮겨 놓는 것은 고차원적인 사고를 요구하는 것은 아니다. 하지만 이러한 활동을 통해 하고자 하는 것은 자신의 해결책이 실제로 어떠한 결과를 낳는지는 확인해보기 위한 것이다.

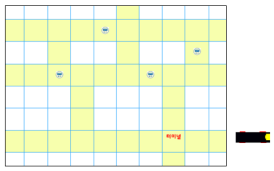
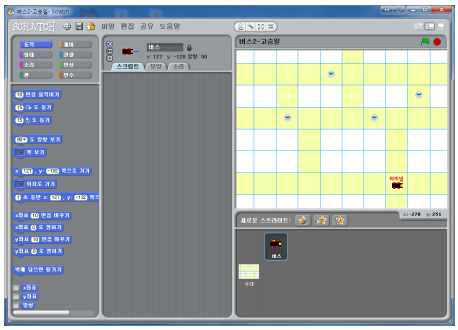
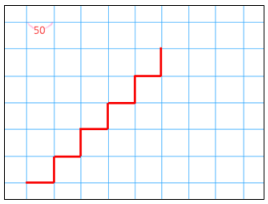
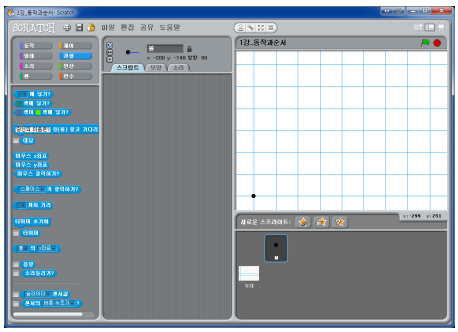
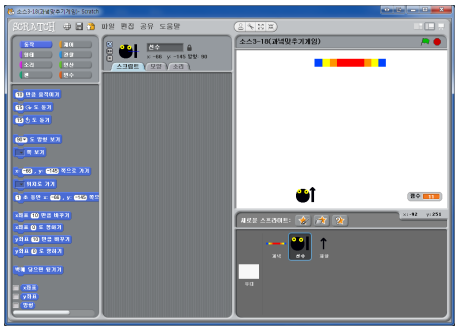
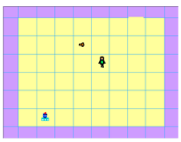
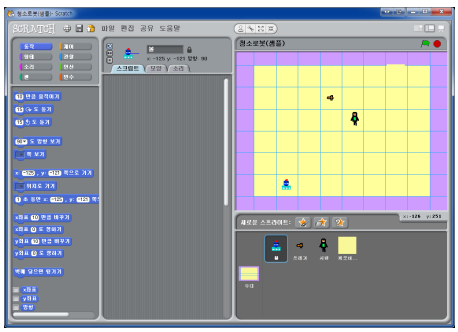


[그림 Ⅲ-9] 스크래치 프로젝트로 제시되는 프로그래밍 문제

만약 학습자 자신의 해결책이 문제의 해답이 아닐 경우에는 그 오류를 찾아내야 한다. 이러한 과정을 거쳐 학습자는 문제가 요구하는 해답을 찾게 되고, 각 장에서 가르치고자 하는 프로그래밍의 핵심 개념을 학습하게 된다.

프로그래밍 문제해결 활동에서의 학습지 활동과 실제 학습자에게 제공되는 프로젝트 중 몇 가지를 예시로 들자면 <표 Ⅲ-6>와 같다.

<표 Ⅲ-6> 교재의 문제와 스크래치 프로젝트의 문제

구분	교재의 문제 구성	스크래치 프로젝트의 문제 구성
1장. 동작과 순서	**버스 운전하기 #1** 나는 무인버스(운전사 없이 운행되는 버스)를 만드는 과학자이다. 무인버스는 처음에 터미널에 있다가 버스 정류장(▭)이 있는 곳을 모두 지나치고 터미널로 다시 들어와야 한다. 버스는 처음 90도(오른쪽)을 향하고 있다.  무인버스를 만드는 컴퓨터 프로그램은 "스크래치 버튼 프로그램"이다. 아래의 3개의 버튼만 사용할 수 있다. [1]번 버튼을 누르면 [10만큼 움직이기] 가 실행되고, [2]번 버튼을 누르면 [90 ↻ 도 돌기] (시계방향으로 90도 회전)이 실행되고, [3]번 버튼을 누르면 [90 ↺ 도 돌기] (반시계방향으로 90도 회전)이 실행된다.	
2장. 제어와 관찰, 형태	**반복문의 위력?**  50격자로 나뉘어진 바탕무대에서 스프라이트가 움직이며 그림을 그렸다. **문제를 해결해 봅시다.** 위의 경로처럼 스프라이트가 움직이며 선을 그리는 프로그램을 만드시오. 단, 사용할 수 있는 블록은 펜 팔레트의 모든 블록과 아래의 3가지 제어 블록이다. [x좌표 10 만큼 바꾸기] [y좌표 10 만큼 바꾸기] [반복 5 회]	
3장. 변수, 연산과 동시 처리	**과녁 맞추기 게임** 하나의 게임을 만드려고 한다. 스페이스바를 누르면 화살을 쏘게 되고 과녁의 어느 부분에 맞느냐에 따라 점수가 계산된다.  **문제를 해결해 봅시다.** 게임을 디자인해봅시다. 화면구성을 그리거나 아이디어를 쓰세요.	
4장. 객체 지향, 인공 지능 그리고 시뮬레이션	**청소로봇 #1**  방에 사람, 로봇이 있다. 1. 사람과 로봇은 1초마다 무작위(위/아래/오른쪽/왼쪽)로 움직이지만 벽으로 이동할 수는 없다. 2. 사람은 50%의 확률로 현재 위치에 쓰레기를 버린다. (스탬프 기능 사용) 3. 로봇은 현재 위치에 쓰레기가 있다면 이를 줍는다. (스탬프 기능 사용)	

4. 검사 도구의 개발 및 검증

기존의 많은 계산적 사고와 관련된 연구에서도 마찬가지로 계산적 사고의 검사 도구의 개발 연구는 매우 미흡하다. 본 연구에서는 계산적 사고의 구성 요인을 계산적 인지력(추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고)과 계산적 창의성(창의적 사고)로 구분하여 설정하였다.

계산적 창의성의 검사 도구는 Torrance의 TTCT 언어 검사를 사용하였으며, 계산적 인지력 검사 도구의 개발은 본 연구에서 직접 수행되었다. 검사 도구의 개발 과정은 관련 연구 및 문헌 조사에서부터 문항 제작, 내용 선정, 내용 타당도 검증, 양호도 및 신뢰도 측정, 동형검사 신뢰도 측정, GALT와의 상관분석의 절차를 거쳤으며, 총 제작기간은 3개월이 소요되었다.

1) 평가 문항 제작

본 연구에서 측정하고자 하는 계산적 사고력은 일반적인 문제 해결 상황에 적용 가능한 전략적 지식에 해당하며, 컴퓨터과학 분야의 암기적 지식에서 벗어나 실제적인 문제 해결 과정에 적용할 수 있는 능력을 측정할 수 있어야 할 것이다. 이를 위해 평가 도구 검사지의 개발을 위한 틀을 이은경(2009)의 연구에서의 틀을 수정하여 <표 Ⅲ-7>과 같이 선정하였다.

<표 Ⅲ-7> 평가 도구 검사지 개발 틀

구성 요소	개발 방향
문제 유형	다양한 인지도구 및 표현양식을 활용하도록 요구하는 복합 과제 형태로 구성
문제 상황	실생활 맥락적인 문제 상황 제시
학문 영역	문제에서 평가하고자 하는 내용은 영역 특수적일 수 있으나 이를 해결하는 사고에서는 영역 특수적 지식의 영향을 최소화 할 수 있는 문항 구성
문제 해결 과정	문제 해결 과정에서 학습자가 활용하는 절차와 전략에 대한 정보 수집이 가능한 형태로 구성

이에 따라 <부록 10>, <부록 11>과 같은 계산적 사고력 측정 검사지 A와

이와 동형 검사지 형태인 B를 만들었다. 이 검사들은 <표 Ⅲ-8>과 같이 각각 총 10가지의 주제로 총 25개의 문항을 제작되었는데 이는 소프트웨어의 인지복잡도의 측정(Gu & Tong, 2004; Minsky & Minsky, 1968; Misra, 2006; Shao & Wang, 2003)과 프로그래밍을 중심으로 한 컴퓨터과학에서 핵심 주제/아이디어에 대한 관련 연구(Schwill, 1994, 1997; Zendler A & Klaudt, 2012; Zendler & Spannagel, 2008; Zendler, Spannagel & Klaudt, 2008)에서 추출하였다.

<표 Ⅲ-8> 주제와 문항수

문제번호	주 제	문항수	주제별 문항수
1	순차구조	1	2
2		1	
3	조건분기	2	3
4		1	
5	반복	2	3
6		1	
7	동시성 (병렬처리)	2	3
8		1	
9	변수	2	5
10		3	
11	난수	2	2
12	알고리즘	1	1
13	객체	2	2
14	함수	2	2
15	재귀	2	2
			총 25 문항

2) 타당도 검증

본 연구에서 개발한 문항들에 대한 타당도를 알아보기 위해 전문가 집단에게 설문을 실시하였다. 컴퓨터교육 분야에서 경력이 7~10년인 현직 초등교사 6명으로 전문가 집단이 구성되었다. 전문가 각자에게 개발된 문항들을 무작위 순서로 모두 제시하고, 해당 문항 검사가 학습자의 문제 해결 과정에서 어떠한 단계에 해당되며 컴퓨터과학의 어떠한 핵심 내용을 다루고 있는지에 대한 의견을 설문을 통해 조사하였다.

<표 Ⅲ-9> 각 문항에 대한 전문가 집단의 분석

문항 번호	평가 개념 영역	빈도수 /비율	문제 해결의 단계					평가 개념 영역(컴퓨터과학 및 프로그래밍 개념)									
			문제이해 및 분석	문제해결 방안탐색	문제해결 방안설계	해결 및 구현	평가	순차 구조	조건 분기	반복	동시성	변수	난수	알고 리즘	객체	함수	재귀
1	순차 구조	빈도	6	4	1	2	0	4	3	3	0	3	0	2	0	0	1
		%	100	67	17	33	0	67	50	50	0	50	0	33	0	0	17
2	순차 구조	빈도	2	4	4	1	2	3	2	0	0	0	0	4	0	1	0
		%	33	67	67	17	33	50	33	0	0	0	0	67	0	17	0
3	조건 분기	빈도	5	1	4	3	0	2	6	0	0	0	0	1	2	1	1
		%	83	17	67	50	0	33	100	0	0	0	0	17	33	17	17
4	조건 분기	빈도	6	2	1	1	2	2	5	1	0	3	0	1	3	1	0
		%	100	33	17	17	33	33	83	17	0	50	0	17	50	17	0
5	반복	빈도	5	2	1	4	0	3	0	6	0	0	0	0	0	1	0
		%	83	33	17	67	0	50	0	100	0	0	0	0	0	17	0
6	반복	빈도	4	3	1	3	0	6	2	3	0	1	0	0	0	0	0
		%	67	50	17	50	0	100	33	50	0	17	0	0	0	0	0
7	동시성	빈도	4	3	0	3	2	5	5	4	4	0	0	0	0	1	0
		%	67	50	0	50	33	83	83	67	67	0	0	0	0	17	0
8	동시성	빈도	5	4	0	3	0	5	3	0	3	0	0	1	0	2	0
		%	83	67	0	50	0	83	50	0	50	0	0	17	0	33	0
9	변수	빈도	5	3	1	2	2	5	2	0	0	3	0	1	0	4	1
		%	83	50	17	33	33	83	33	0	0	50	0	17	0	67	17
10	변수	빈도	4	1	1	4	3	6	1	0	0	5	0	1	0	3	0
		%	67	17	17	67	50	100	17	0	0	83	0	17	0	50	0
11	난수	빈도	3	4	1	4	2	3	0	0	2	5	5	0	0	2	1
		%	50	67	17	67	33	50	0	0	33	83	83	0	0	33	17
12	알고 리즘	빈도	5	2	4	5	1	6	4	3	0	1	4	4	0	3	2
		%	83	33	67	83	17	100	67	50	0	17	67	67	0	50	33
13	객체	빈도	4	4	2	3	0	1	1	1	0	4	0	0	5	1	0
		%	67	67	33	50	0	17	17	17	0	67	0	0	83	17	0
14	함수	빈도	4	2	1	4	2	3	1	0	0	3	0	0	0	5	0
		%	67	33	17	67	33	50	17	0	0	50	0	0	0	83	0
15	재귀	빈도	4	2	4	4	2	2	1	1	0	3	0	1	2	5	5
		%	67	33	67	67	33	33	17	17	0	50	0	17	33	83	83
전체		빈도	66	41	26	46	18	56	36	22	9	31	9	31	11	12	15
		%	73	46	29	51	20	62	40	24	10	34	10	34	12	13	17

전체 문항에 대해 전문가들의 73%는 문제 해결의 단계의 ‘문제이해 및 분석’ 과정이 포함된다고 판단하였으며, ‘문제 해결 방안 탐색’ 과정이 46%, ‘문제 해결 방안 설계’ 과정이 29%, ‘해결 및 구현’ 과정이 51%, ‘평가’ 과정이 20%로 포함된다고 판단하였다.

개발된 문항의 컴퓨터과학 및 프로그래밍 학습의 소재에 대한 타당도 설문조사에서는 대부분의 문항에 대해 전문가 집단은 ‘순차구조’, ‘조건분기’가 포함되었다고 판단하였다. 모든 문항을 전체적으로 봤을 때 ‘순차구조’는 전문가의 62%가, ‘조건분기’가 40%가 포함되었다고 판단하였다. 결과적으로 평가 제작시 평가하고자 하는 개념에 대한 전문가 집단의 선택 비율보다 더 높은 선택을 받은 평가 문항도 있었다. 이러한 평가 문항에 대해서는 다음과 같은 원리로 평가 개념 영역명을 수정하였다.

- 원리: 하나의 문항에서 의도한 평가 개념 항목의 전문가 선택 비율 이상인 항목이 존재하며 이것이 50% 이상이라면 평가 주제명에 해당 평가 개념을 추가한다. 만약 의도한 평가 개념이 전문가들로부터 50%의 선택을 받지 못하면 50% 이상 받은 평가 개념으로 주제명을 수정한다. 어떠한 평가 개념으로도 50% 이상의 선택을 받지 못한다면 평가 문항 자체를 삭제한다.

<표 Ⅲ-10> 수정된 문항 구성

문제번호	기존 평가 주제	추가된 평가 주제	문항수
1	순차구조		1
2		알고리즘	1
3	조건분기		2
4			1
5	반복		2
6		순차구조	1
7	동시성 (병렬처리)	순차구조, 조건분기, 반복	2
8		순차구조, 조건분기	1
9	변수	순차구조, 함수	2
10		순차구조	3
11	난수	변수	2
12	알고리즘	난수	1
13	객체		2
14	함수		2
15	재귀	함수	2

이러한 원리로 2번 문항은 '알고리즘', 6번 문항은 '순차구조', 7번 문항은 '순차구조', '조건분기', '반복', 8번 문항은 '순차구조', '조건분기', 9번 문항은 '순차구조', '함수', 10번 문항은 '순차구조', 11번 문항은 '변수'. 12번 문항은 '순차구조', '조건분기', '난수', 15번 문항은 '함수' 영역이 각 문항 주제명에 추가 시켰다. 이러한 전문가 집단의 타당도 검사에서 알 수 있는 것은 모든 문제가 컴퓨터과학의 내용을 소재로 한 기반의 문제이며 문제 해결에서의 복합적인 단계를 요구하는 수준의 문제로 분석되어졌다.

3) 양호도 검증과 신뢰도 측정

<표 Ⅲ-11> 문항별 난이도, 변별도, 신뢰도 측정표

문제번호		주제	난이도		변별도 지수		신뢰도 Cronbach's α
			A형	B형	A형	B형	
1		순차구조	0.09	0.06	0.21	0.17	.526
2		순차구조,알고리즘	0.18	0.56	0.42	0.44	.569
3	①	조건분기	0.74	0.71	0.41	0.53	.330
	②		0.71	0.94	0.30	0.12	.480
4			0.74	0.71	0.43	0.54	.778
5	①	반복	0.71	0.79	0.36	0.56	.769
	②		0.24	0.18	0.37	0.54	.446
6		순차구조, 반복	0.47	0.62	0.59	0.61	.755
7	①	순차구조, 조건분기, 반복, 동시성	0.62	0.74	0.50	0.27	.767
	②		0.41	0.50	0.10	-0.17	.214
8		순차구조, 조건분기, 동시성	0.68	0.62	0.58	0.56	.765
9	①	순차구조, 변수, 함수	0.76	0.85	0.67	0.56	.520
	②		0.53	0.50	0.53	0.73	.522
10	①	순차구조, 변수	0.82	0.53	0.56	0.46	.299
	②		0.47	0.50	0.57	0.74	.641
	③		0.38	0.41	0.42	0.71	.820
11	①	변수, 난수	0.59	0.35	0.49	0.54	.364
	②		0.38	0.21	0.48	0.67	.652
12		난수, 알고리즘	0.29	0.15	0.22	0.46	.618
13	①	객체	0.50	0.56	0.38	0.33	.695
	②		0.41	0.59	0.59	0.57	.627
14	①	함수	0.41	0.50	0.69	0.56	.835
	②		0.41	0.41	0.67	0.67	.679
15	①	함수, 재귀	0.32	0.47	0.59	0.71	.845
	②		0.35	0.44	0.69	0.67	.828
전체			0.48	0.51	-	-	

계산적 인지력 평가 25문항에 대한 양호도(난이도와 변별력) 측정 및 신뢰도 측정을 위해 제주시 초등학교에 재학 중인 4학년 21명과 서귀포시 초등학교에 재학 중인 6학년 13명, 총 34명의 초등학생들에게 동형검사를 무작위 문항 순으로 시간차 없이 투입하여 데이터를 수집하였다. 다음의 표는 총 25문항에 대한 난이도, 변별력, 동형검사 문항간의 문항내적일관성 신뢰도 측정의 결과이다.

난이도는 고전검사이론(한국교육평가학회, 2004)에 입각하여 <표 Ⅲ-11>과 같이 각 문항의 답을 맞춘 피험자수를 총 피험자 수로 나누어 측정하였고, Cangelosi(1990)의 문항 난이도에 의한 문항평가가 이루어졌으며, A형 전체의 난이도가 0.48, B형의 전체 난이도가 0.51로 매우 적절하게 난이도가 분포되었다.

<표 Ⅲ-12> Cangelosi(1990)의 문항 난이도에 의한 문항평가

문항난이도	문항평가	문항번호				문항수
.75이상	쉬운 문항	A9-①	A10-①			A형: 2
		B3-②	B5-①	B9-①		B형: 3
.25이하	어려운 문항	A1	A2	A5-②		A형: 3
		B1	B5-②	B11-②	B12	B형: 4
.25~.75	적절한 문항	그 외 문제들				A형: 20 B형: 18

문항 변별력은 상관계수(r)에 의해 측정하였고(성태제, 2010), 그 결과를 <표 Ⅲ-13>과 같이 Ebel(1965)의 기준으로 상관계수가 -1 이상 +1 이하의 값을 가지며, 변별도지수에 따라 문항변별도를 평가하게 된다. 변별력을 인정받을 수 없는 변별도 지수가 .30 이하의 문제는 A형과 B형 검사의 문항 1번, 문항 7-②번과 A형 검사의 문항 12번, B형 검사의 문항 3-②번, 문항 7-①번이 변별력이 낮은 것으로 측정되었다.

<표 Ⅲ-13> Ebel(1990)의 문항 변별력 기준에 의한 문항평가

변별도지수	문항평가	문항번호				문항수
.10미만	변별력이 없는 문항	B7-②				A형: 0 B형: 1
.10이상~.20미만	변별력이 매우 낮은 문항	A7-② B1	B3-②			A형: 1 B형: 2
.20이상~.30미만	변별력이 낮은 문항	A1 B7-①	A12			A형: 2 B형: 1
.30이상~.40미만	변별력이 있는 문항	A3-② B13-①	A5-①	A5-②	A13-①	A형: 4 B형: 1
.40이상	변별력이 높은 문항	그 외 문제들				A형: 18 B형: 20

신뢰도 측정은 문항내적일관성에 대해 측정하였고 Cronbach's α값으로 신뢰도를 평가하였다. Nunnally(1978)의 연구에 따라 <표 Ⅲ-14>와 같이 신뢰도 평가의 Cronbach's α 계수의 최소치를 .5로 정하고 이에 따라 각 문항들의 신뢰도를 분류하였다.

Cronbach's α에 의한 상관계수가 .50 이하의 문항은 3-①, 3-②, 5-②, 7-②, 10-①, 11-①로 신뢰도 수준이 낮았다.

<표 Ⅲ-14> Nunnally의 Cronbach's α에 의한 문항 신뢰도 평가

Cronbach's α	신뢰도 평가	문항번호				문항수
.5 미만	신뢰할 수 없음	3-① 10-①	3-② 11-①	5-②	7-②	6
.5이상~.6미만	최소의 기준 (.5이상)	1	2	9-①	9-②	4
.6이상~.7미만	허용할 만함	10-② 13-②	11-① 14-②	12	13-①	6
.7이상	신뢰할 만함	4 8 15-②	5-① 10-③	6 14-①	7-① 15-①	9

4) 문항의 수정・보완

개발한 문항의 변별도 측정에 의해 A, B형에서 모두 변별력이 없는 검사의 문항인 7-②번을 삭제하였다. 문항 삭제에 의해 변별도가 다시 측정되기 때문에 삭제 문항수를 최소화 하였다. 또한 신뢰도 측정에 의해 신뢰할 수 없는 문항 3-①, 3-②, 5-②, 7-②, 10-①, 11-①을 삭제하였다. 이에 따라 수정된 문제는 다음의 표와 같으며 수정된 문제만을 갖고 다시 변별도를 측정하였다. 변별도는 각 개인의 전체 점수와 해당 문항의 정답 여부의 비율로 계산되기 때문에 문제를 삭제할 경우 변별도 지수는 달라진다. 또한 A형, B형의 전체 난이도도 달라지기 때문에 <표 Ⅲ-15>와 같이 재측정하여 제시하였다.

<표 Ⅲ-15> 수정・확정된 문항 구성

문제번호		주제	난이도		변별도지수		신뢰도 Cronbach's α
			A형	B형	A형	B형	
1		순차구조	0.09	0.06	0.24	0.14	.526
2		순차구조,알고리즘	0.18	0.56	0.41	0.41	.569
4		조건분기	0.74	0.71	0.43	0.56	.778
5	①	반복	0.71	0.79	0.38	0.57	.769
6		순차구조, 반복	0.47	0.62	0.59	0.61	.755
7	①	순차구조, 조건분기, 반복, 동시성	0.62	0.74	0.50	0.29	.767
8		순차구조, 조건분기, 동시성	0.68	0.62	0.61	0.61	.765
9	①	순차구조, 변수, 함수	0.76	0.85	0.67	0.52	.520
	②		0.53	0.50	0.52	0.77	.522
10	②	순차구조, 변수	0.47	0.50	0.59	0.76	.641
	③		0.38	0.41	0.41	0.70	.820
11	②	변수, 난수	0.38	0.21	0.54	0.70	.652
12		난수, 알고리즘	0.29	0.15	0.20	0.45	.618
13	①	객체	0.50	0.56	0.37	0.31	.695
	②		0.41	0.59	0.61	0.61	.627
14	①	함수	0.41	0.50	0.73	0.58	.835
	②		0.41	0.41	0.66	0.65	.679
15	①	함수, 재귀	0.32	0.47	0.61	0.72	.845
	②		0.35	0.44	0.74	0.69	.828
전체		총 19문제	0.46	0.53	-	-	

※ 삭제된 문제: 총 6문제 (문항 3-①번, 3-②번, 5-②번, 7-②번, 10-①번, 11-①번)

5) 동형검사 신뢰도 측정

개발 후 선정된 총 19문제에 대하여 Spearman의 상관계수를 이용하여 동형검사 신뢰도를 측정하였고 <표 Ⅲ-16>, <표 Ⅲ-17>은 이 결과를 나타낸 것이다.

<표 Ⅲ-16> 동형검사의 기술통계

	평균	표준 편차	N
A형 검사	8.71	4.726	34
B형 검사	9.68	5.062	34

<표 Ⅲ-17> 동형검사의 상관분석

		A형 검사	B형 검사
A형 검사	Pearson 상관계수	1	.931**
	Sig. (2-tailed)		.000
	N	34	34

$p^{**} < .01$

<표 Ⅲ-17>에서 A형 검사, B형 검사간 계산적 인지력 지수(각 문항의 총점)의 상관계수는 .931로 매우 높은 상관을 나타내고 있으며 유의수준 .01(양쪽)에서 유의하였다. 즉 두 검사지 A형, B형은 동형검사로써 신뢰도를 확보하였다고 할 수 있겠다.

6) GALT 검사와의 상관분석 및 타당성 검증

본 연구에서 계산적 인지력이 추상적 사고, 비판적 사고, 논리적 사고, 재귀적 사고, 알고리즘적 사고로 이루어져 있고, 문제 해결 과정에서 대부분이 논리적 사고를 기반으로 한다면 계산적 인지력과 논리적 사고력은 상관관계가 있다고 할 수 있을 것이다. 이러한 이유로 본 연구에서 개발한 계산적 인지력 검사를 기존에 논리적 사고력 검사로 가장 일반적으로 사용되는 GALT 검사(Roadrangka, Yeany, & Padilla, 1983)와의 상관관계를 분석하여 보았다. 검사의 방법은 동형검사를 실시한 표본 집단에게 하루의 간격을 두고 GALT를 투

입하여 검사하였다. 각 학생이 얻은 계산적 인지력 검사 A형과 B형의 총점과 GALT에서 얻은 총점과 하위 요인들의 점수들에 관한 상관을 분석하고자 하였다. <표 Ⅲ-18>은 이 두 검사의 기술통계이고, <표 Ⅲ-19>는 이 두 검사의 상관관계에 대한 결과이다.

<표 Ⅲ-18> 계산적 인지력 검사와 GALT 검사의 기술통계

검사도구 유형 (만점)			평균	표준편차
계산적 인지력 검사 A형 (19)			8.71	4.726
계산적 인지력 검사 B형 (19)			9.68	5.062
논리적 사고력	총 점 (21)		5.76	3.144
	하위요인	보존 (4)	2.85	1.019
		비례 (6)	.74	1.286
		변인통제 (4)	.88	1.149
		확률 (2)	.15	.500
		상관 (2)	.03	.171
		조합 (3)	1.12	.808

(N=34)

<표 Ⅲ-19> 계산적 인지력 검사 A형, B형과 논리적 사고력(GALT)와의 상관분석

		A형 총점	B형 총점	논리적 사고력(GALT)						
				총점	하위요인					
					보존	비례	변인통제	확률	상관	조합
A형 총점	Pearson 상관계수	1	.931**	**.615****	.349*	.480**	.395*	.352*	.086	.390*
	Sig. (2-tailed)		.000	.000	.043	.004	.021	.041	.630	.022
B형 총점	Pearson 상관계수	.931**	1	**.631****	.378*	.424*	.442**	.402*	.116	.402*
	Sig. (2-tailed)	.000		.000	.027	.012	.009	.018	.513	.018

**. 상관계수는 0.01 수준(양쪽)에서 유의함.
*. 상관계수는 0.05 수준(양쪽)에서 유의함.

Pearson의 상관계수의 해석을 위해 성태제(2007)는 <표 Ⅲ-20>과 같이 그 일반적인 기준을 제시하였고 이 기준에 따라 본 연구에서도 결과를 해석을 하였다.

<표 Ⅲ-20> Pearson 상관계수의 해석 기준

상관계수의 범위	상관관계의 해석
±.00 ~ .20	상관이 매우 낮다.
±.20 ~ .40	상관이 낮다.
±.40 ~ .60	상관이 있다.
±.60 ~ .80	상관이 높다.
±.80 ~ 1.00	상관이 매우 높다.

<표 Ⅲ-19>를 살펴보면, 계산적 인지력 검사 A형과 GALT 상관 분석의 상관계수가 .615이고, 검사 B형과 GALT의 상관계수는 .631이다. 이는 <표 Ⅲ-18>의 상관계수 해석 기준으로 볼 때, 두 종류의 검사가 상관이 높음을 나타내고 있다는 것을 알 수 있다. 계산적 인지력 검사 A형은 GALT 검사의 논리적 사고력의 하위 요인인 보존논리와는 .349, 비례논리와는 .480, 변인통제논리와는 .295, 확률논리와는 .352, 조합논리와는 .390의 상관계수를 나타내고 있다. 즉 이러한 다수의 하위 요인들과의 낮은 상관관계들이 논리적 사고력의 총점과 계산적 인지력 A형의 총점과의 높은 상관을 나타내게 된 이유로 분석된다.

계산적 인지력 검사 B형은 GALT 검사의 논리적 사고력의 하위 요인인 보존논리와는 .378, 비례논리와는 .424, 변인통제논리와는 .442, 확률논리와는 .402, 조합논리와는 .402의 상관계수를 나타내고 있다. 같은 관점으로 바라볼 때, 이러한 하위요인과의 다수의 낮은 상관관계가 논리적 사고력의 총점과 계산적 인지력 검사 B형과의 높은 상관관계를 나타내게 된 것으로 분석된다.

결과적으로 본 연구에서 개발한 계산적 인지력 검사 A형과 B형은 모두 논리적 사고를 측정하기 위한 GALT와도 상관도가 높은 것으로 나타났으며, 이로써 계산적 인지력 검사가 최소 논리적 사고력을 측정할 수 있다는 검사의 타당성을 일부 확보할 수 있다고 할 수 있을 것이다.

IV. 연구의 적용 및 결과 분석

1. 연구 방법

1) 연구 가설

본 연구의 가설은 2가지이며 다음과 같다.

① <가설 1>

- 영가설: PPS기반 프로그래밍 교육 프로그램에 의한 학습자의 계산적 인지력에는 차이가 없다.
- 대립가설: PPS기반 프로그래밍 교육 프로그램에 의한 학습자의 계산적 인지력에는 차이가 있다.

② <가설 2>

- 영가설: PPS기반 프로그래밍 교육 프로그램에 의한 학습자의 창의성에는 차이가 없다.
- 대립가설: PPS기반 프로그래밍 교육 프로그램에 의한 학습자의 창의성에는 차이가 있다.

2) 연구 대상

본 연구에서 개발된 참여자는 지원자 표집(교육기부 프로그램)에 의한 지원자 표본(volunteer sample)이다. 제주도내 초등학교 4, 5학년 학생들 중 선착순 지원자 17명(4학년 12명, 5학년 5명)을 대상으로 전체 프로그램의 오리엔테이션, 사전·사후의 TTCT 언어 검사를 제외하여 총 6일 동안 24차시 수업으로 진행되었다. 강사 1인이 전체 학습을 진행하였고 보조 강사 1인은 PPS 활동 및 프로그래밍 문제해결 활동에 도움을 주었다. 모든 학생들은 데스크 탑 컴퓨터를 각자 사용하였고 윈도우용 스크래치 버전 1.4를 사용하였다.

성태제와 시기자(2006)는 실험 연구에서 표본 집단에 대한 크기에 대해 다음과 같이 언급하였다. “표본의 크기를 어느 정도로 하는 것이 좋은가에 대한 절대적인 기준은 존재하지 않지만 Gall, Gall과 Borg(2003)는 상관연구에서는 전

통적으로 최소한 30명 이상의 피험자수를 사용해야 하고 비교-실험연구의 경우 비교되는 각 집단마다 최소한 15명 이상의 피험자가 있어야 한다고 제시하였다"고 언급하였다. 이에 따라 본 연구에서도 17명의 참여자로 학습 프로그램에 대한 비교-실험연구에 무리가 없을 것으로 판단하고 진행하였다.

계산적 인지력을 비교하기 위한 비교집단은 제주도 동지역 초등학교 4학년 7명, 읍·면 지역 초등학교 5학년 4명, 6학년 8명으로 총 18명의 지원자를 대상으로 실험집단과 같이 사전·사후에 계산적 인지력 검사지 A형과 B형을 투입하였다.

<표 Ⅳ-1> 연구대상자

구분	실험집단 (명)			비교집단 (명)		
성별	남	여	소계	남	여	소계
4학년	8	4	12	3	4	7
5학년	3	2	5	-	4	4
6학년	-	-	-	6	2	8
소계	11	6	**총 17명**	9	10	**총 19명**

실험집단의 참여자 중 1명은 창의성 검사는 실시하였으나 계산적 인지력 검사는 시행하지 못하여 결측값으로 처리하여 통계를 실시하였다.

3) 연구 설계

본 연구에서 개발한 학습 프로그램은 실험집단에게 집중이수제 형식으로 총 6일간의 집합 강의 및 실습으로 이루어졌다. 시작일과 최종일을 포함하면 총 8일이며, 시작일에는 전체 학습의 소개와 사전평가(창의성 검사, 계산적 인지력 검사)가 이루어졌고 최종일에는 사후평가와 학습자들이 만든 프로젝트를 동료 학생들과 학부모에게 발표하는 시간을 가졌다. 매일 오전 9시부터 오후 1시까지 40분의 수업 4차시와 3회의 10~15분 휴식, 1회의 간식 시간이 주어졌다. 2일차와 3일차 사이에는 주말(토요일, 일요일)로 인해 프로그램을 운영하지 않았다.

<표 Ⅳ-2> 연구 투입 기간

구분	시작일 (수)	1일차 (목)	2일차 (금)	3일차 (월)	4일차 (화)	5일차 (수)	6일차 (목)	최종일 (금)
1차시	소개							사후평가
2차시								
3차시	사전평가							발표회
4차시								

4) 검사 도구

(1) 계산적 인지력 검사

계산적 인지력 측정을 위해서 사전사후검사 통제집단설계(pretest-posttest control group design)를 사용하였다. 검사 도구로는 본 연구에서 개발한 계산적 인지력 검사를 사용하였다.

<표 Ⅳ-3> 계산적 인지력 측정의 연구 설계

	사전검사	처치	사후검사
비교집단 (N=21)	O_1		O_2
실험집단 (N=17)	O_3	X	O_4

X : 프로그래밍 학습

O_1O_3 : 사전검사(계산적 인지력 검사) - 두 독립표본 t검정

O_2O_4 : 사후검사(계산적 인지력 검사) - 두 독립표본 t검정

O_3O_4 : 사전·사후검사(계산적 인지력 검사) - 두 대응(종속)표본 t검정

(2) 창의성 검사

창의성 검사에 대해서는 단일집단 사후검사설계(one-group posttest-only design)를 사용한다. 이는 표준화검사 자료나 국가 단위의 믿을 만한 자료가 있을 때 사용할 수 있는 방법으로 단일표본 t검정을 사용한다(성태제, 시기자, 2006).

<표 Ⅳ-4> 창의성 측정의 단일집단 사후검사 설계

처치	사후검사 (모집단과 비교)
X	O

X : 프로그래밍 학습

O : 사후검사(TTCT 검사) - 정규성 검증 후 단일표본 t검정

본 연구에서 사용한 창의성 검사 도구는 Torrance가 개발한 창의성 검사의 한국어 번역판으로 '규준표' 작성을 위해 초등 1학년에서 고등학교 3학년 이상

(12+학년)까지 A형은 6,918명, B형은 5,638명의 표본에 의해 측정된 타당도와 신뢰도를 갖는 표준화 검사이다(김영채, 2010).

또한 집단내의 처치 효과를 명확히 알기 위해서 단일집단 사전사후검사설계(one-group pretest-posttest design)로 실험집단 내의 사전·사후 창의성 검사 결과를 비교하였다.

<표 Ⅳ-5> 창의성 측정의 단일집단 사전사후검사 설계

사전검사	처치	사후검사 (집단내 비교)
O_1	X	O_2

X : 프로그래밍 학습

O_1O_2 : 사전·사후검사(TTCT 검사) - 정규성 검증 후 두 대응(종속)표본 t검정

2. 연구의 적용

앞서 제시한 5단계의 학습과정에 따른 세부 학습 내용에 대한 전개에 대한 실제 학습은 과정은 다음과 같이 진행된다.

1) 계산모델의 분석 단계

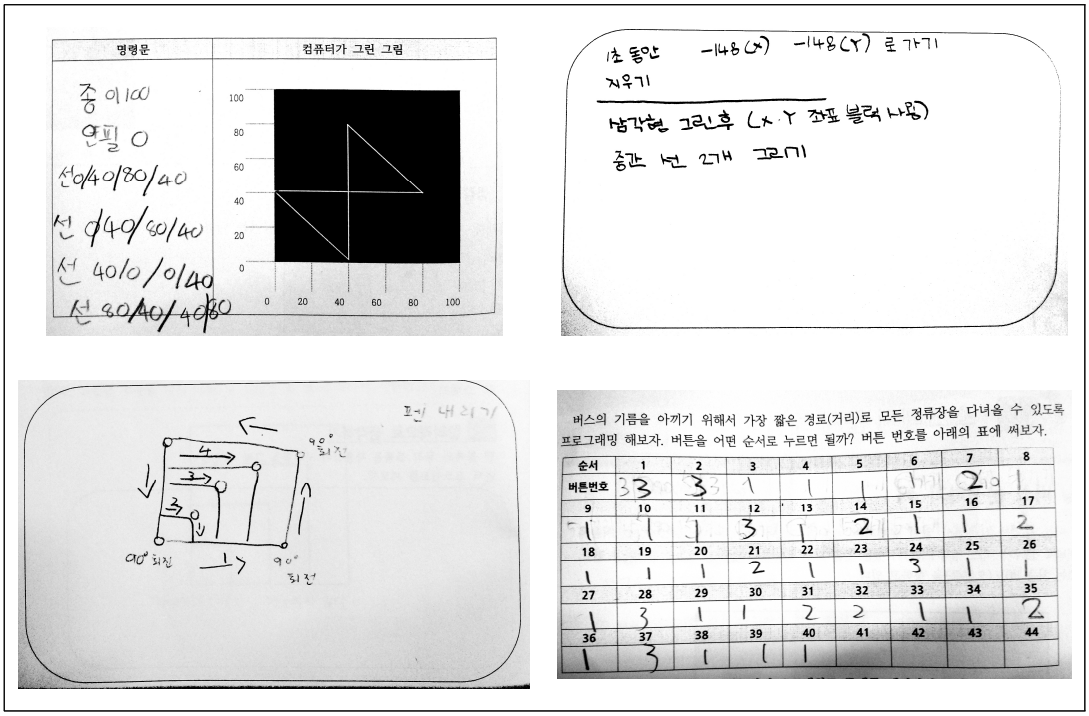
계산모델의 분석 단계에서는 학습자는 사용할 프로그래밍 언어 중 학습에 필요한 문법과 기능을 학습한다. 이러한 학습은 계산모델의 속성을 익히게 되고 계산모델을 분석적으로 이해할 수 있게 한다. 실제 프로그래밍 언어를 접하고 문법의 규칙과 기능을 학습하게 된다.

2) 계산화 단계

계산화 단계에서는 언플러그드식의 학습으로 프로그래밍 언어와 비슷한 계산 모델을 정의하여 문제를 해결하거나, 직접 프로그래밍 언어를 사용하여 문제(퀴즈 형식)를 해결할 수 있도록 한다. 어떠한 형식으로 문제를 해결하든

PPS 활동을 빠트리지 않고 자신의 해결책에 대한 아이디어를 모델링할 수 있는 기회를 갖게 하는 것이 중요하다.

이 단계의 문제는 각 장에서 해결하려는 프로그래밍 해결문제의 크기보다 상대적으로 작은 크기의 문제로 프로그래밍의 핵심 개념과 처리과정을 학습하게 된다.



[그림 Ⅳ-1] 계산화 단계의 PPS 활동 결과물

이 단계의 학습의 PPS 활동은 다음과 같은 교육적 효과를 기대할 수 있다.

- **학습자의 계산적 사고의 능동적 활용을 유도한다.** 학습자는 자신의 해결책을 나타내는데 자신만 알아보기 위한 메모 수준이 아니라 좀 더 논리적이고 일반적으로 나타낼 수 있도록 안내한다. 이러한 활동 자체로도 학습자에게 비판적·논리적·추상적·재귀적·알고리즘 사고의 활용을 기대할 수 있다.
- **창의성 발휘를 유도한다.** 제시된 문제에 대한 해답은 상당히 다양할 수도 있고, 아니면 오로지 하나일 수도 있다. 최적의 또는 최선의 해답

을 찾는 과정에서 학습자는 기존에 자신이 갖고 있던 지식을 활용하여 창의성의 유창성과 융통성, 독창성을 발휘하게 되는 기회를 갖게 된다.

- **구현(자동화)을 고려한 설계가 이루어진다.** 학습자는 학습의 초반에는 이러한 활동들이 어떠한 의미를 갖는지 모르지만 학습과정을 거치면서 이러한 활동이 더 큰 문제를 해결하고 이를 프로그래밍으로 구현하기 위한 설계 및 기초적 학습 단계임을 깨닫게 되고, 구현(자동화)의 과정을 염두에 두어 설계를 하는 습관을 갖게 할 것이다.

3) 문제의 분석·탐색 및 아이디어 발견 단계

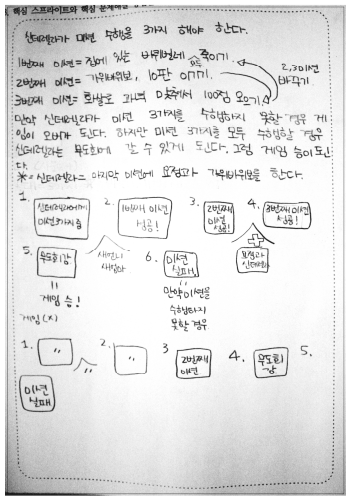
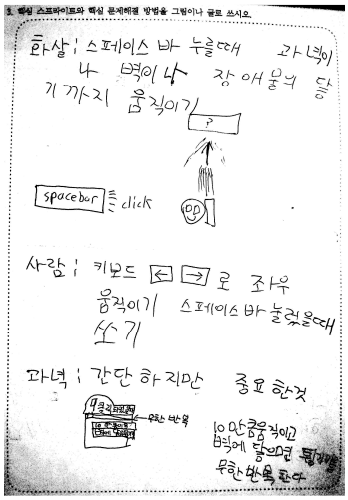
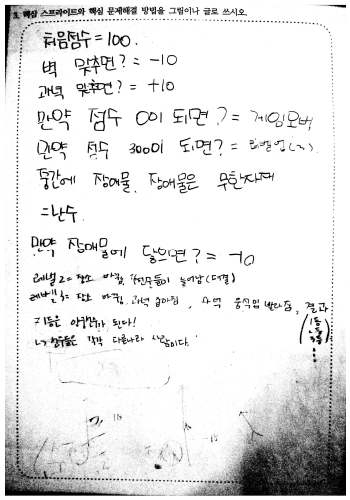
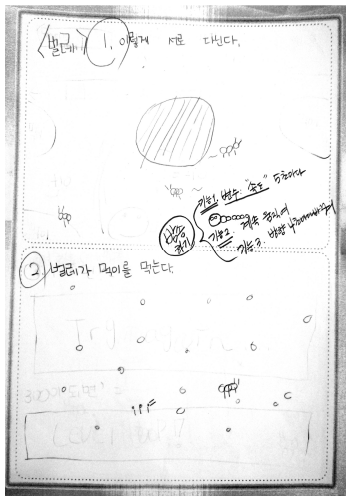
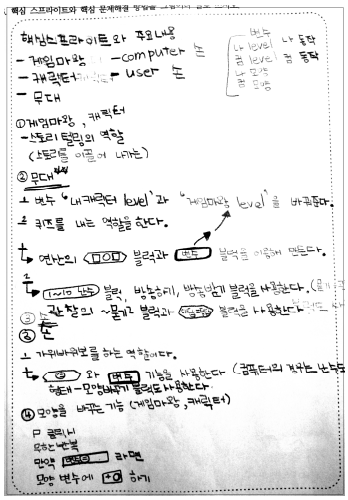
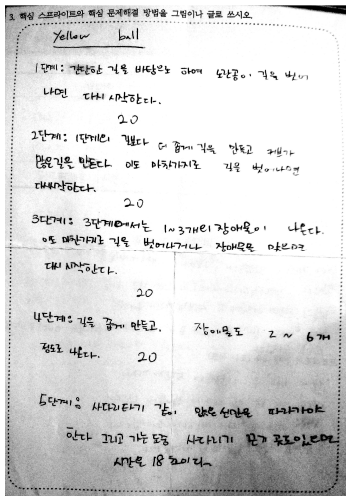
이 단계에서는 학습자에게 제시된 계산화 문제와 유사한 프로그래밍 문제가 제시된다. 이 문제는 상대적으로 계산화 문제보다는 복잡하고 사고의 깊이를 더 요구하는 문제이다. 이 단계에서는 문제가 요구하는 것이 무엇인지를 파악하고 이를 해결하기 위한 대안을 탐색하고 발견하게 된다. 학습자 자신의 대안을 여러 가지로 생각할 수 있으며 PPS 활동을 통해 해결책을 모델링할 수 있다. 교사는 이 단계의 활동 중에 순회하며 PPS 활동이 단순히 화면 스케치가 아닌 학생들 나름의 논리를 갖고 문제를 해결해 나갈 수 있는 해결책의 모델링 설계가 될 수 있도록 첨삭 지도가 필요하다. 문제 자체가 교재의 학습지 형태로 제공되지만 학습자가 프로그래밍 언어에서 문제 파일을 열어보고 아이디어를 얻고자 할 때에는 이를 수용하였다.

학생들이 프로그래밍 문제해결에 있어서 PPS를 활용하고, 그 결과물들을 보면 [그림 Ⅳ-2]와 같이 크게 세 가지 유형으로 분류할 수 있었다.

첫 번째 유형은 스토리 중심의 모델링으로 시간의 흐름에 따라 프로젝트가 어떻게 진행되는지를 표현하는 방법이다. 두 번째 유형은 객체 중심 모델링으로 객체(스프라이트)가 하는 일을 중심으로 표현하는 방법이다. 이는 스크래치가 객체를 사용함으로 인해 학습자가 '객체중심 프로그래밍'의 개념을 학습하지 않고도 객체 중심의 프로그래밍 사고를 습득하게 된 것으로 보아진다. 세 번째 유형은 이벤트 중심의 모델링으로 전체 프로젝트에서 어떠한 사건(이벤트)이 생겼을 때 어떠한 변화가 생기는지를 표현한 방법이다. 학습자들은 자신의 프로젝트에서 점수의 획득이나 손실, 단계의 상승, 어디와 부딪혔을 때 등

의 사건을 중심으로 전체 프로젝트의 흐름을 표현하기도 하였다.

물론 제시된 문제의 유형에 따라 적합한 모델링 방법이 있을 수 있겠지만, 같은 문제를 가지고도 여러 학습자가 자신들만의 해결책을 다양한 모델링 방법으로 표현하였다.

스토리 중심 모델링	객체 중심 모델링	이벤트 중심 모델링
		
		

[그림 Ⅳ-2] 문제의 분석·탐색 및 아이디어 발견 단계의 PPS 활동 결과물

이 단계는 다음과 같은 프로그래밍 교육에 있어서 다음과 같은 교육적 효과를 기대해 볼 수 있다.

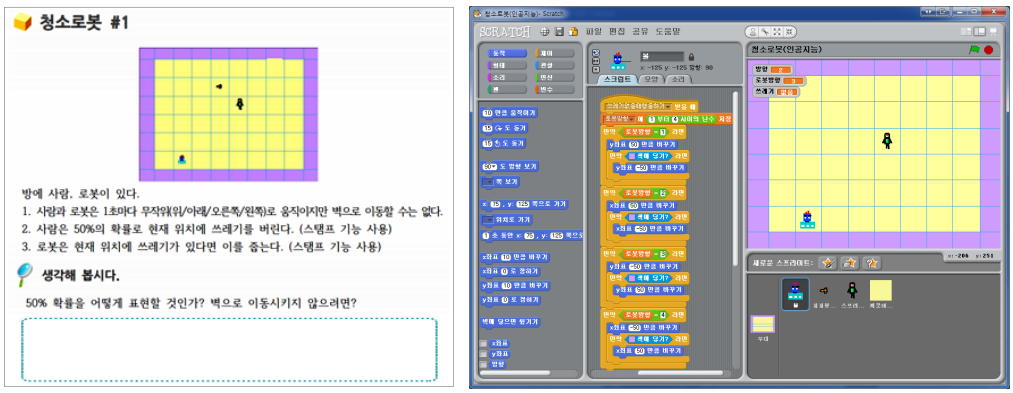
· **기존의 지식을 바탕으로 재귀적 사고, 알고리즘적 사고의 능동적 활**

용을 촉진한다. 계산모델의 학습이나 계산화 연습 활동을 통해 배웠던 프로그래밍의 핵심 개념과 기술을 이용하여 좀 더 높은 수준의 사고를 요구하는 문제의 해결책을 탐색한다. 이 과정에서 학습자의 재귀적 사고 및 알고리즘적 사고의 활용이 요구된다.

- **문제해결의 아이디어 탐색을 통해 유창성과 독창성을 요구한다.** 다양한 아이디어를 발견하는 활동을 통해 학습자의 창의성의 구성요인 중 유창성과 독창성에 대한 촉진을 기대할 수 있다.
- PPS의 적극적인 활용과 그 기술의 습득은 **프로그래밍의 문제뿐 아니라 다양한 형태의 실세계 문제를 해결하는 데에도 적극적으로 활용할 수 있다.**

4) 해결 발견 단계

이 단계에서는 전 단계에서 모델링 해두었던 몇 가지 대안들을 실제 프로그래밍 언어로 구현 및 테스트해보며 최적의 해답을 찾게 된다. 프로그래밍 언어의 문제 파일은 전 단계에서 학습지로 제시된 문제와 똑같은 형태로 제공된다.



[그림 Ⅳ-3] 학습지의 문제(좌)와 실제 프로그래밍 문제(우)

이 단계의 학습은 다음과 같은 교육적 효과를 기대할 수 있다.

- **프로그래밍 설계의 중요성을 알게 된다.** 학습지의 문제와 똑같은 형식의 프로그래밍 언어의 문제를 제공함으로 해서 학습자는 자신의 모델링 계획을 그대로 코드(code)로 옮기는 일에만 집중할 수 있다. 설계 단

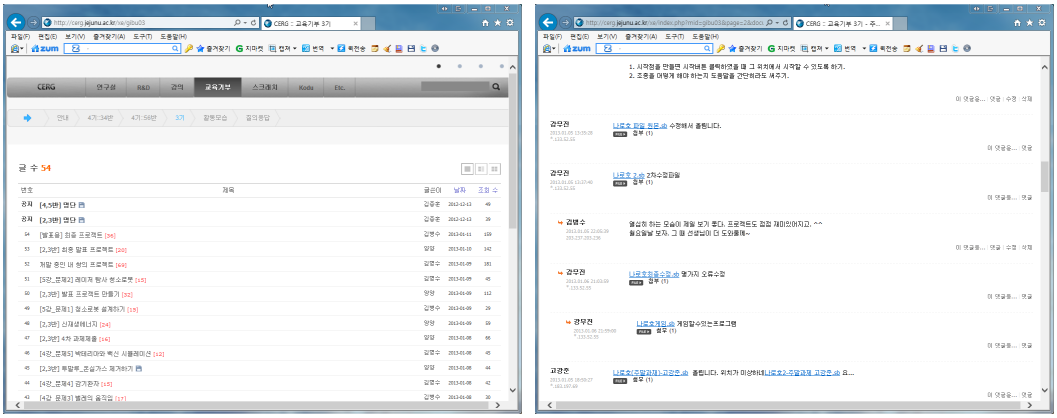
계에서 매우 구체적인 코드까지는 계획되지 않더라도 설계가 잘 된 해결책일수록 프로그래밍이 쉽다는 경험을 갖게 되며, 이는 설계의 중요성을 인식하게 하는 기회가 된다.

- **문제 자체에 몰입하는 시간을 확대하여 준다.** 물론 그리기 활동이 창의성에 긍정적인 영향을 줄 수는 있겠지만, 경험적인 측면으로 보았을 때 나이가 어리고, 비주얼 프로그래밍을 처음 접하는 학생일수록 스크래치 프로그래밍 활동시 그리기 활동에 몰입하는 시간이 비효율적으로 상당히 많은 편이었다. 즉, 문제 자체를 프로젝트 파일로 제공함은 학습자에게 문제 자체에 몰입하는 시간을 더 확대하여 주게 된다.
- **문제 해결을 위한 동기를 부여한다.** 잘 만들어진 문제형식의 프로젝트는 학생들로 하여금 문제를 풀고 싶은 동기를 유발한다. 주어진 문제 자체가 흥미로운 도전이라면, 이는 문제 상황 자체에 몰입하고 문제 해결을 위한 동기를 부여하게 된다.
- **학습 전이가 효과적으로 일어날 것이다.** 앞서 거쳐 온 일련의 학습 과정 속에서 학습자는 배운 내용을 활용하고자 할 것이며 제시되는 문제들이 서로 연관성이 많다면 학습의 전이가 효과적으로 일어날 것이다.

5) 수용 발견 단계

강의자는 예상되는 해답에 대해 샘플 예제를 미리 만들었으며 문제를 해결하는 데 어려움을 겪는 학습자를 대상으로 샘플 예제의 해답과 비슷하도록 문제를 해결할 수 있도록 유도하였다. 하지만 학습자 스스로 제시한 아이디어에 대해서는 최대한 수용하고 그러한 방식대로 해결해 나갈 수 있도록 수용하였다. PPS 활동에서는 특정한 프로그래밍 언어가 제공하는 기능의 한계를 고려하지 않았으나, 실제 '해결 발견' 단계에서의 프로그래밍은 이러한 괴리를 가져올 수 있다. 이 단계에서는 이러한 방해 요인이나 제한점을 극복하기 위한 방법을 찾고, 오류와 버그를 찾아 자신의 해결책을 완성해 나가도록 한다.

학습자가 완성한 프로젝트 파일은 게시판을 이용하여 공유하고 교사 또는 학습자 상호간 피드백을 주고받기도 하였다.



[그림 Ⅳ-4] 프로젝트 공유를 위한 게시판 활용

이 단계에서의 교육적 효과로 다음과 같은 것들을 기대해 볼 수 있을 것이다.

- **문제해결에 대한 접근을 비교할 수 있다.** 게시판을 이용한 프로젝트의 공유로 학습자는 같은 문제임에도 불구하고 자신과는 다른 방법으로 문제를 해결한 많은 프로젝트를 볼 수 있다. 이는 문제를 바라보는 관점의 차이로 고정된 관점에서 벗어날 수 있는 학습자의 통찰력을 촉진한다.
- **프로젝트의 완성도를 높일 수 있다.** 게시판으로 공유되는 프로젝트는 공개되기 때문에 학습자는 프로젝트 완성에 몰두하고 오류 수정을 지속적으로 하는 효과가 관찰되었다.

3. 연구 결과와 분석

1) 계산적 인지력의 검사

연구의 참여자 중 1명은 본 연구의 최종일에 결석을 하여 사후 검사를 실시하지 못하였고 이 원자료는 결측값으로 처리하여 계산적 인지력 검사 결과를 분석하였다.

(1) 정규성 검정

실험집단과 비교집단의 사전 계산적 인지력 검사의 결과를 이용하여 두 집단에 대해 각각 정규성 검정을 실시하였다. 정규성 검정은 콜모고로프-스미르노프 (Kolmogorov-Smironv) 검정을 계산적 인지력 지수에 대해서 실시하였다.

<표 Ⅳ-6> 정규성 검정

	기술통계				귀무가설	Sig.	귀무가설 결정
	평 균	표준편차	최대	최소			
실험집단 (N=16)	13.38	2.363	16	9	계산적 인지력 지수(각 문항의 총점)의 평균이 13.38이고 표준편차가 2.36인 정규 분포이다.	.683	채택
비교집단 (N=19)	12.58	2.714	17	8	계산적 인지력 지수(각 문항의 총점)의 평균이 12.58이고 표준편차가 2.71인 정규 분포이다.	.846	채택

두 집단에 대해 각각 콜모고르프-스미르노프 검정을 실시한 결과, 먼저 실험집단은 유의확률 .683으로 유의수준 .05에서 귀무가설을 기각하지 못하였다. 즉 '계산적 인지력 지수의 평균이 13.38이고 표준편차가 2.36인 정규 분포이다'라는 귀무가설을 채택하였으며 실험집단이 정규분포를 가짐을 확인하였다.

비교집단에 대해서도 유의확률 .846로 유의수준 .05에서 귀무가설을 기각하지 못하였다. 즉 '계산적 인지력 지수의 평균이 12.58이고 표준편차가 2.71인 정규 분포이다'라는 귀무가설을 채택하였으며 비교집단이 정규분포를 가짐을 확인하였다. 정규성이 검증된 두 집단에 대해 다음의 분석들은 모수통계를 사용하였다.

(2) 사전 검사의 두 독립표본 t검정

계산적 인지력 검사에 의해 측정된 총점을 계산적 인지력 지수라고 할 때, 사전 검사의 계산적 인지력 지수를 이용하여 두 독립표본 t검정을 실시하였다.

<표 Ⅳ-7> 사전 검사의 두 독립표본 t검정

		N	평균	표준 편차	t	Sig. (2-tailed)
사전검사의 계산적 인지력 지수	실험집단	16	13.38	2.363	.916	.366
	비교집단	19	12.58	2.714		

<표 Ⅳ-7>의 두 독립표본 t검정의 결과를 살펴보면, 사전검사의 실험집단 평균은 13.38, 표준편차는 2.363이고 비교집단의 평균은 12.58, 표준편차는 2.714이다. 두 독립표본 t검정 결과의 t 통계값은 .916이고 유의확률은 .366로 유의수준 .05에서 두 집단은 차이가 없었다.

(3) 사후 검사의 두 독립표본 t검정

본 연구의 학습이 이루어진 후, 사후 검사를 실시하고 사후 검사의 계산적 인지력 지수를 이용하여 <표 Ⅳ-8>과 같이 두 독립표본 t검정을 실시하였다.

<표 Ⅳ-8> 사후 검사의 두 독립표본 t검정

		N	평균	표준 편차	t	Sig. (2-tailed)
사후검사의 계산적 인지력 지수	실험집단	16	15.13	1.996	2.252	.031
	비교집단	19	13.05	3.188		

사후검사의 실험집단 평균은 15.13, 표준편차는 1.996이고 비교집단의 평균은 13.05, 표준편차는 3.188이다. <표 Ⅳ-8>의 두 독립표본 t검정 결과를 살펴보면, t 통계값은 2.072이고 유의확률 .031로 유의수준 .05에서 두 집단은 유의미한 차이를 보였다.

(4) 집단내 사전·사후 검사의 대응표본 t검정

두 집단내의 사전·사후 검사의 대응표본 t검정으로 계산적 인지력 지수의 변화를 확실하게 알아보고자 하였고 <표 Ⅳ-9>는 이러한 통계에 대한 결과이

다.

<표 Ⅳ-9> 집단내 사전·사후 검사의 대응표본 t검정

	N	사전검사		사후검사		사전-사후 t	Sig. (2-tailed)
		평균	표준 편차	평균	표준 편차		
실험집단	16	13.38	2.363	15.13	1.996	-2.842	.012
비교집단	19	12.58	2.714	13.05	3.188	-1.531	.143

<표 Ⅳ-9>의 통계 결과를 살펴보면, 실험집단의 사전·사후 대응표본 t검정의 t 통계값은 -2.842이고 유의확률 .012로 유의수준 .05에서 유의미한 차이를 보였다. 비교집단은 사전·사후 대응표본 t검정의 t 통계값은 -1.531이고 유의확률 .143으로 유의수준 .05에서 차이가 없었다.

(5) 계산적 인지력 검사 결과의 분석

실험집단과 비교집단은 정규분포를 갖춘 표본임이 검증되었으며 사전 계산적 인지력 검사의 두 독립표본 t검정에서 두 집단은 차이가 없었다. 본 연구에서 개발한 프로그래밍 학습을 적용한 후 사후 계산적 인지력 검사의 두 독립표본 t검정에서 두 집단은 유의미한 차이를 보였다. 실험집단내 사전·사후의 대응표본 t검정의 결과 유의미한 차이를 보였고, 비교집단은 차이가 없었다. 이로써 <가설 1>은 기각되어 대립가설이 받아들여져 본 연구에서 개발된 프로그래밍 학습에 의한 학습자의 계산적 인지력에는 차이가 있음을 입증하였다.

2) 계산적 창의성의 검사

본 연구에서의 계산적 창의성의 검사 도구는 Torrance가 개발한 TTCT 언어 검사(김영채의 한국 번역판)로 사전검사에 A형, 사후검사에 동형 검사지인 B형을 투입하였다. TTCT의 언어검사는 6개의 과제(활동)로 구성되어 있고 이를 통해 창의성의 3가지 구성 요인(유창성, 융통성, 독창성)을 측정할 수 있다(김영채, 2010). 채점은 창의력 한국 FPSP가 인정하는 TTCT 채점 전문가 자

격을 취득 후 연구자가 직접 채점하였다.

TTCT는 많은 사례의 표본 검사를 통해 요인별, 학년별 통계치가 이미 제시된 타당도와 신뢰도가 높은 검사이다(김영채, 2010). 본 검사의 규준표의 표준점수들은 평균치가 100, 표준편차가 20인 분포를 이루고 있기 때문에 본 연구에서 얻은 검사 결과는 비교집단 없이 단일 표본 검증을 실시하였다. 단, 사전 단계에서 본 연구의 참여자들이 모집단을 대표하는지에 대한 적합성에 대한 검증을 실시하였다.

(1) 정규성 검정

먼저 참여자들이 사전 검사 결과에 따라 정규성 검정을 실시하였다. 사전 검사 데이터에 대한 비모수/모수 통계를 결정하기 위해 정규성 검정의 방법으로 콜모고로프-스미르노프 (Kolmogorov-Smironv) 검정을 실시하였고 <표 Ⅳ-10>은 그 결과를 나타낸 것이다.

<표 Ⅳ-10> 정규성 검정

요 인	기술통계 (N=17)				귀무가설	Sig.	귀무가설 결정
	평 균	표준편차	최대	최소			
유창성	98.71	13.665	128	77	평균100, 표준편차가 20인 정규분포이다.	.619	채택
융통성	101.88	14.628	130	72		.603	채택
독창성	103.00	15.540	139	85		.318	채택
창의성 지수	101.24	14.087	132	78		.409	채택

참여자의 TTCT 사전 검사(언어)에서 얻은 유창성, 융통성, 독창성, 그리고 이 지수의 총합인 창의성 지수의 유의확률은 .619, .603, .318, .409로 유의수준 .05에서 모집단과 같은 정규 분포를 갖고 있음을 검증하였다. 이에 따라 이후의 통계는 모수통계의 방법을 사용하였다.

(2) 사후 실험집단과 모집단의 비교

본 연구에서 개발한 학습을 적용한 후에 단일 표본 t검정을 실시하였고 검정값은 100으로 설정(TTCT의 표준점수의 평균)하였다. 이는 학습 후 실험집단이 모집단과의 평균과 분포에 차이가 있는지를 비교하기 위한 것이다.

<표 Ⅳ-11> 사후 단일 표본 t검정

	평균	표준편차	t	Sig. (2-tailed)
유창성	111.71	13.499	3.575	.003
융통성	114.06	11.239	5.158	.000
독창성	111.71	13.499	3.575	.003
창의성 지수	111.82	12.300	3.964	.001

<표 Ⅳ-11>의 통계 결과를 살펴보면, 참여자의 TTCT 사후 검사(언어)에서 얻은 유창성 t 통계값은 3.575, 융통성은 5.158, 독창성은 3.575, 그리고 이 지수의 총합인 창의성 지수의 t 통계값은 3.964로 유의수준 .05에서 유의확률이 각각 003, .000, .003, .001로 모든 요인에 대해서 모집단과의 차이를 보였다.

(3) 사전·사후 대응 표본 t검정

사전·사후의 창의성 요인 및 창의성 지수에 대한 대응 표본 t검정을 통해 실험집단의 참여자들의 창의성의 어떠한 하위요인에 변화가 생겼는지를 알아보고자 하였다. <표 Ⅳ-12>는 이러한 통계의 결과이다.

<표 Ⅳ-12> 사전·사후 대응 표본 t검정

사전-사후	대응 상관관계 (Sig.)	사전-사후		t	Sig. (2-tailed)
		평균	표준편차		
유창성	.693 (.002)	-11.118	10.629	-4.312	.001
융통성	.543 (.024)	-12.176	12.724	-3.946	.001
독창성	.643 (.005)	-8.706	12.414	-2.892	.011
창의성 지수	.658 (.004)	-10.588	11.029	-3.958	.001

<표 Ⅳ-12>의 통계 결과를 살펴보면, 사전·사후 대응 표본 t검정의 결과로 유창성, 융통성, 독창성 및 창의성 지수의 t 통계값은 -4.312, -3.946, -2.892, -3.958로서 유의수준 .05에서 모두 유의미한 차이가 있는 것으로 분석되었다.

(4) 창의성 검사 결과의 분석

사전검사에서 모집단과 같은 정규성을 가진 실험집단은 사후검사에서 모집단과의 단일표본 t검정 비교에 의해 유의미한 차이가 생겼음을 보여주었다. 이로써 <가설-1>은 기각되어 대립가설이 받아들여져 본 연구에서 개발된 프로그래밍 교육 프로그램에 의한 학습자의 창의성에는 차이가 있음이 입증되었다.

또한, 사전·사후 대응 표본 t검정에 의해서는 실험 참여자들의 창의성의 모든 하위 요인들과 이들의 총합인 창의성 지수가 증진됨을 밝혔다. 이로써 본 연구에서 개발한 프로그래밍 학습 프로그램은 학습자들의 창의성의 모든 요인들을 증진시키는 데에 긍정적인 영향을 끼칠 수 있음을 밝혔다.

3) 분석

본 연구에서는 계산적 사고력을 계산적 창의성과 계산적 인지력으로 나누고 각 해당 영역을 측정하였다. 본 연구에서 개발한 교육 프로그램을 통해 학습자의 계산적 창의성과 계산적 인지력이 모두 증진되었기 때문에 계산적 사고력이 증진되었다고 말할 수 있을 것이다.

V. 결론 및 제언

컴퓨터과학 교육이라는 학문 영역에서 무엇을, 어떻게, 왜 가르쳐야 하는가에 대한 고민은 학문 자체에 대한 가치와 목적을 정립하는 것과 같은 맥락일 것이다.

먼저, 컴퓨터과학에서 무엇을 가르쳐야 하는가에 대한 질문을 생각해 볼 때, 컴퓨터과학의 중심 연구 대상에는 '계산(computation)'이 있으며 이러한 계산에 대한 원리 학습이 컴퓨터과학의 핵심을 학습하게 되는 것이라고 생각할 수 있다.

이러한 학습은 21세기 모든 인간에게 필요하게 될 문제 해결 능력을 갖추게 하기 위함으로 수학적, 과학적, 공학적 사고를 수반하는 '계산적 사고(computational thinking)'를 갖추게 하기 위함이고 이것이 컴퓨터과학을 가르치는 목적이 되어야 할 것이다. 또한 계산적 사고에 의한 문제 해결은 문제에 따른 해법의 기계적인 반복 적용이 아닌 상당한 창의성을 요구하는 일이며 이 또한 컴퓨터과학 교육이 이루어지는 이유가 될 것이다.

마지막으로, 창의성을 포함한 계산적 사고력을 기르기 위해 컴퓨터과학의 핵심을 가르치기 위해 컴퓨터과학 교육을 어떻게 해야 하는가에 대한 질문의 해답은 '계산 원리의 학습'이다. 계산 원리의 학습을 가장 효율적으로 학습할 수 있는 활동이 프로그래밍이다. 프로그래밍 활동 자체가 계산에 대한 이해와 계산 원리의 학습 과정이라고 해도 과언이 아닐 정도이다. 계산적 사고의 개념에서 프로그래밍은 프로그래밍 언어의 학습이 아닌 컴퓨팅의 기능인 추상화와 자동화에 대한 활동이며 이는 컴퓨팅 머신 상에서만 이루어지는 것이 아니라는 점을 이해해야 할 것이다.

본 연구에서는 이러한 기본적인 교육적 관점에서 계산 원리의 학습으로써 언플러그드식의 PPS(Paper-and-pencil programming strategy) 활동을 제안하여 비전공자를 대상으로 교육적 효과를 검증하였고, PPS기반 프로그래밍 학습을 통하여 초등학생들의 계산적 사고력(계산적 인지력과 계산적 창의성) 증진에 도움을 주고 있음을 입증하였다.

본 연구의 결과가 기존의 연구와 어떠한 차이점을 갖고, 그 차이점으로 인해 어떠한 시사점을 찾아볼 수 있는지에 대해 알아볼 필요가 있을 것이다.

첫째, 기존 연구에서 언급하고 있는 계산적 사고에 대한 개념을 확대하여 창의성을 포함한 계산적 사고력을 정의하고, 이들의 구성 요인들을 분석하고 관계를 도식화하여 제시하였다. 계산적 사고, 창의성과 관련한 기존의 연구에서는 계산적 사고의 구성요인을 설정하고 이들을 단순히 나열하여 제시하는데 그쳤다. 본 연구에서는 여기에서 한 단계 더 나아가, 계산적 사고의 구성요인을 5가지로 추출하여 인간의 사고 구조에서 이들의 관계를 도식화하였다. 이 관계 중에는 창의성과의 관계도 포함되어 있다. 이러한 과정에서 인지과학, 공학교육의 연구 결과를 참조하였으며, 이를 통해 계산적 사고는 인간의 고차적 사고의 수준에서 다양한 사고의 범주와 차원 안에서의 관계임을 나타내고자 하였다. 이러한 시도는 추후의 관련 연구에서 수정・보완되더라도 계산적 사고의 실체를 이해하고자 하는 연구의 시작점이 될 수 있다는 의의를 갖는다.

둘째, 프로그래밍 활동에서 PPS의 활용을 차이점으로 둘 수 있다. PPS를 하나의 언플러그드 학습 방법이라고 여길 경우, 기존의 언플러그드 학습의 경우는 컴퓨터과학 관련 영역 또는 다른 학문의 영역에서의 높은 학업 성취도를 보인다는 주장의 연구가 주류였다(박윤성, 2009; 서인숙, 2011; 전현석, 2010; 조현하, 2011). 그러나 본 연구에서의 결과는 기존의 연구들과는 다른 측면을 가지고 있다. 언플러그드식의 학습으로서 PPS를 활용하고 이를 실제 프로그래밍 활동과 연계함으로써, 프로그래밍에서 '설계의 학습'에 초점이 맞추고 있다는 점에서 기존 연구와의 차이점을 찾을 수 있다. 컴퓨터과학의 학습에서 PPS 활동의 가장 큰 장점은 학습자 스스로가 계산적 사고의 실체를 정확히 인식하고 이를 활용할 수 있는 능력을 지니게 될 수 있다는 것이다. 또한, PPS를 활용하는 과정에서 요구되는 발산적 사고와 수렴적 사고가 계산적 사고와 창의성을 촉진시키는 역할을 하게 되는 것이다. 이는 본 연구의 가장 핵심적인 결과라고도 할 수 있다.

셋째, 계산적 사고력 중 계산적 인지력 검사 도구의 개발이다. 앞서 언급하였듯이 계산적 사고력 검사 도구의 개발은 아직 매우 미흡한 상태이다. 이는 계산적 사고에 대한 정의에서부터 구성요인의 설정, 해당 구성요인의 측정을

위한 문항 개발 등의 절차가 쉽지 않고 오랜 기간이 걸리기 때문으로 파악된다. 본 연구에서는 초등학교 학생 수준의 계산적 사고력 검사 도구를 개발하였기 때문에 추후의 연구에서도 수정·보완하여 사용할 수 있을 것으로 기대된다.

본 연구의 결과를 확고히 다지기 위해서는 크게 3가지 부분에서 지속적인 검증이 필요하다.

첫째, 전문가 검증을 통한 계산적 인지력 검사지의 타당도와 신뢰도의 확보이다. 계산적 사고력 중 계산적 인지력으로 구분된 구성 요인을 측정하기 위해 알맞은 내용인지, 검사 방법과 결과는 신뢰할 수 있는지에 대한 지속적인 연구가 필요하다. 본 연구에서의 동형검사신뢰도를 확보에 추가하여 검사 도구의 검사-재검사신뢰도(test-retest reliability)를 확보할 수 있는 연구가 필요하다고 본다.

둘째, 학습 결과 자체에 대한 신뢰도 확보이다. 본 연구의 실험 집단은 편의 표집에 의한 지원자들이다. 지원자 표본(vonlunteer sample)에 의한 참여자들은 비지원자들에 의한 연구보다 상대적으로 학력, 사회계층, 지적 수준이 높을 수 있다(Rosenthal & Rosnow, 1975)는 주장이 있기 때문에 본 연구의 결과가 학습자의 인지적 능력에 의한 것이 아님을 입증하기 위해서는 다수의 실험집단으로 추가 연구가 필요하다고 본다.

셋째, 본 연구에서 계산적 사고력의 구성 요인들 간의 관계에 대한 상관 분석이 필요하다. 본 연구에서는 상관연구에 필요한 30명 이상의 참여자가 되지 않아 이러한 분석을 할 수 없었다. 추후의 연구에서는 다수의 참여자로 상관분석을 통해 이들의 상관관계를 분석할 필요도 있다고 생각된다.

우리는 누구나 일상생활에서 문제에 부딪히고, 그 문제를 해결해 나가며 살아간다. 우리는 그 문제를 해결해 나가는 과정에서 가장 좋은(효율적) 방법을 탐색하거나 새로운 방법을 시도하기도 한다. 대부분의 사람들에게 이렇게 하여 얻어진 문제 해결의 방법의 경험은 극히 개인적인 것으로 남고 공유되지 못한다. 그런 반면, 컴퓨터과학은 하나의 학문으로서 다른 학문과 차별성을 갖게 되는 주요한 특징 중 하나는 '문제 해결 방법' 자체에 대해 연구하는 학문 영역이라는 점이다. 컴퓨터과학 교육의 목적은 모든 인간을 프로그래머나 컴퓨터

과학자로 만들려는 것이 아니라 컴퓨팅 원리가 사회를 지배할 미래 사회에서 모든 인간의 문제 해결에 대한 의사소통을 돕고자 하는데 있을 것이다.

본 논문은 이러한 컴퓨터과학 교육의 목적 달성에 기여하고자 한, 프로그래밍 학습 방법에 대한 실험 연구로 지속적으로 본 연구의 결과를 입증하기 위해 보완·발전시켜 나갈 것이다.

참 고 문 헌

교육과학기술부 (2012. 12). *중학교 선택 교과 교육과정* (교육과학기술부 고시 제 2011-361호 [별책 18]). 교육과학기술부.

권대용 (2011). *Algorithmic brick based tangible robot and hybrid programming environments for enhancing computational thinking*. 박사 학위논문, 고려대학교 대학원.

김갑수 (2010). 초등학생들의 창의력과 논리력 향상을 위한 프로그래밍 언어 교수전략에 관한 연구. *정보교육학회논문지, 14*(1), 89-97.

김광수 (2002). 비판적 사고론. *철학연구, 58*, 5-42.

김나라 (2012). *CPS기반의 PBL수업이 중등정보영재의 창의적 문제해결성향에 미치는 영향*. 석사 학위논문, 한국교원대학교 교육대학원.

김미진 (2010). *동화를 활용한 문제해결 활동 경험이 유아의 창의성 증진에 미치는 영향*. 석사 학위논문, 중앙대학교 교육대학원.

김병수 (2010). *한국정보올림피아드 경시부문 전국본선 교재 개발 연구: 초등부를 중심으로*. 석사 학위논문, 제주대학교 교육대학원.

김병수, 김종훈 (2012a). 정보교과교육: 계산적 사고 기반 정보과학 알고리즘 학습의 설계. *한국컴퓨터교육학회 학술발표대회논문집, 16*(2), 85-90.

김병수, 김종훈 (2012b). 초등 예비교사의 컴퓨터과학에 대한 인식 변화를 위한 계산적 사고 기반 알고리즘 학습의 설계 및 적용. *수산해양교육연구, 24*(4), 528-542.

김병수, 김종훈 (2012c). 한국정보올림피아드 초등부 경시부문 문제해결을 통한 알고리즘 교재 개발 및 적용. *정보교육학회논문지, 16*(1), 11-20.

김선희 (2012). *의사소통 능력 신장 프로그램이 수학 문장제 해결능력에 미치는 영향*. 석사 학위논문, 한국교원대학교 교육대학원.

김수환, 이원규, 김현철 (2009). 개정된 정보교육과정에서 교육용프로그래밍언어의 교육적 적용방안. *컴퓨터교육학회논문지, 12*(2). 23-31.

김양은 (2010). 초등학생 '게임 제작'과정을 통한 게임 이해 과정에 대한 질적 고찰. *한국방송학보, 24*(4), 7-46.

김영옥 (2011). *학습 몰입 이론을 적용한 게임작성 프로그래밍 수업 모형 개발 및 적용*. 석사 학위논문, 한국교원대학교 교육대학원.

김영정 (2004). 비판적 사고: 비판적 사고와 공학교육. *공학교육, 11*(2), 94-101.

김영정 (2005). *한국의 미래를 위한 기반 교육으로서의 창의성과 비판적 사고 교육*. 한국연구재단(NRF) 연구성과물.

김영채 (1999). *창의적 문제해결: 창의력의 이론*, 개발과 수업. 서울: 교육과학사.

김영채 (2007). 교수·학습의 과정과 창의력 교육. *사고개발, 3*(2), 1-35.

김영채 (2010). *검사요강: Torrance TTCT (언어) 검사 A & B형*. 대구: 창의력 한국FPSP.

김영채 (2012). 창의력의 영역 보편성과 특수성: 쟁점과 TTCT 창의력 검사의 분석. *대한사고개발학회지, 8*(1), 1-29.

김윤미 (2013). *CPS 활용 미술프로그램이 유아의 창의성과 조형미술 표현력에 미치는 효과*. 석사 학위논문, 숭실대학교 교육대학원.

김재권 (1997). *심리철학*(하종호 외 번역). 철학과현실사.

김정아, 김병수, 이지훤, 김종훈 (2011). 융합형 인재 양성을 위한 IT 기반 STEAM 교수·학습방안 연구. *수산해양교육연구, 23*(3), 445-460.

김종진, 현동림, 김승완, 김종훈, 원유헌 (2010). 교육용 프로그래밍 언어인 로고와 스크래치 교재 개발 및 비교 실험. *한국콘텐츠학회, 10*(7), 459-469.

김종혜 (2009). *정보과학적 사고 기반의 문제 해결 능력 향상을 위한 중등 교육 프로그램*. 박사 학위논문, 고려대학교 대학원.

김종훈, 김종진 (2008). *프로그래밍 언어론*. 서울: 한빛미디어.

김현진 (2008). *생활 속에서 발견된 피보나치 수열에 대하여*. 석사 학위논문, 조선대학교 교육대학원.

김형석, 김종혜, 이원규 (2009). 컴퓨터 과학 교육 1: 프로그래밍 교육과 창의성의 인지적 요소와의 상관관계 분석. *컴퓨터교육학회논문지 학술발표대회논문집, 13*(1), 175-179.

김형철 (2011). *컴퓨터과학 교육용 계산 원리 학습도구의 기능요소 고찰*. 석사 학위논문, 제주대학교 교육대학원.

김혜숙 (1999). 창의성 진단 측정도구의 개발 및 타당화. *교육심리연구, 13*(4), 269-303.

류미애 (2005). *게임고등학교의 교육체계에 관한 연구: 컴퓨터 게임제작 과정을 중심으로*. 석사 학위논문, 중부대학교 대학원.

류충구 (2012). Scratch가 초등 영재들의 창의적 문제해결력에 미치는 효과. 석사 학위논문, 경인교육대학교 교육대학원.

문외식 (2005). 초등학생의 논리적 사고력 및 문제 해결 능력 향상을 위한 컴퓨터 프로그래밍 교육과정 모델. *정보교육학회논문지, 9*(4). 595-605.

박남기 (2011). 초등교육 미래 비전에 비추어본 초등교원 양성 교육 개편 방향. *초등교육연구, 24*(3), 325-349.

박숙 (2000). 피보나치 수열에 대하여: 수열의 역사와 응용, 그리고 현대적 해석과 현장수업을 중심으로. 석사 학위논문, 한남대학교 교육대학원.

박용철, 이수정 (2011). 스크래치 프로그래밍 교육이 초등학생의 자기 주도적 학습 능력에 미치는 효과. 정보*교육학회논문지, 15*(1), 93-100.

박윤성 (2009). *초등학교 컴퓨터교육에서 언플러그드 학습 방법을 활용한 정보표현 영역 교수·학습에 관한 연구*. 석사 학위논문, 진주교육대학교 교육대학원.

박은진, 김희정 (2004). *비판적 사고를 위한 논리*. 서울: 아카넷.

박정신, 조석봉 (2012). 프로그래밍입문 수업에서 스크래치 활용 효과분석. *디지털정책연구, 10*(9), 449-456.

박지연 (2012). *Computational Thinking 능력과 유아놀이와의 연관성 분석*. 석사 학위논문, 고려대학교 대학원.

백성현 (2009). *게임 프로그래밍 교육을 통한 초등학생의 게임중독 개선*. 석사 학위논문, 경인교육대학교 교육대학원.

백영균 (2006). 게임기반학습(Game Based Learning) 활성화의 전제조건에 대한 고찰. *한국정보과학회, 24*(2), 45-50.

박창호, 곽호완, 김성일, 김영진, 김진우, 이건효, 이재식, 이종구, 한광희, 황상민 (2007). *인지공학심리학: 인간-시스템 상호작용의 이해*. 서울: 시그마프레스.

삼광현, 노명우, 강정석 (2012). *미래교육의 열쇠 창의적 문화교육: 협력적 다중지능적 창의적 발달을 위한 새로운 교육학*. 살림터.

서성원 (2010). *TPL과 VPL을 활용한 로봇 프로그래밍 교육이 정보과학적 사고 능력에 미치는 영향*. 석사 학위논문, 한국교원대학교 교육대학원.

서영민, 이영준 (2010). 초등정보영재의 창의성 신장을 위한 교과 통합 로봇 프로그래밍 수업 모형. *컴퓨터교육학회논문지, 13*(1). 19-26.

서인숙 (2011). *언플러그드 협동학습이 초등 정보기기 교육의 학습동기 및 학업성취도에 미치는 효과*. 석사 학위논문, 한국교원대학교 대학원.

성태제 (2004). *문항제작 및 분석의 이론과 실제*. 서울: 학지사.

성태제, 시기자 (2006). *연구방법론*. 서울: 학지사.

성태제. (2007). *SPSS/AMOS를 이용한 알기 쉬운 통계분석 - 기술통계에서 구조방정식 모형까지*. 서울: 학지사.

손은경 (1995). *취학전 아동의 연령, 과제제시방법 및 과제유형에 따른 인과성 발달 연구.* 박사 학위논문, 동아대학교 대학원.

송정미 (2011). *스크래치를 활용한 정보과학적 사고 기반 퍼즐교육.* 석사 학위논문, 고려대학교 교육대학원.

송정범, 조성환, 이태욱, (2008). 스크래치 프로그래밍 학습이 학습자의 동기와 문제해결력에 미치는 영향. *정보교육학회논문지, 12*(3), 323-332.

신우창 (2003). 객체지향 소프트웨어 설계의 재사용을 위한 설계패턴 명세언어. 박사 학위논문, 서울대학교 대학원.

안경미, 손원성, 최윤철 (2011). 스크래치 프로그래밍 교육이 초등학생의 학습 몰입과 프로그래밍 능력에 미치는 효과. *정보교육학회논문지, 15*(1), 1-10.

안정현 (2010). *중학생의 특성을 고려한 스크래치 프로그래밍 수업모형.* 석사 학위논문, 한국교원대학교 대학원.

유인환, 김태완 (2006). MINDSTORMS을 이용한 프로그래밍 학습이 창의력에 미치는 효과. *컴퓨터교육학회논문지, 9*(1), 1-11.

유정수, 이민희 (2009). 두리틀을 이용한 프로그래밍 수업이 창의성, 문제해결력, 프로그래밍 흥미도 향상에 미치는 영향. *정보교육학회논문지, 13*(4), 443-450.

유중현, 김종혜 (2010). 문제 해결과정에서의 정보과학적 사고 능력에 대한 개념적 고찰. *정보창의교육, 2*(2), 15-24.

윤선희, 김영식 (2011). 정보과학 창의성의 구성 요소 탐색. *컴퓨터교육학회논문지, 14*(1), 45-54.

윤혜진 (2012). *유아의 논리적 사고의 발달: 추론상황과 삼단논법을 중심으로.* 박사 학위논문, 계명대학교 대학원.

이강범 (2008). *'비판적 사고'에 대한 고찰과 그 활용: Delphi report를 중심으로.* 석사 학위논문, 연세대학교 교육대학원.

이미정 (2011). *창의성 향상을 위한 글쓰기 교육 연구: 동화 바꾸어 쓰기를 중심으로.* 석사 학위논문, 한남대학교 대학원.

이민희 (2009). 두리틀을 이용한 프로그래밍 수업이 창의성, 문제해결력, 프로그래밍 흥미도 향상에 미치는 영향. 석사학위 논문, 전주교육대학교 교육대학원.

이송희 (2009). *창의성의 개인적 및 사회문화적 결정요인에 관한 연구.* 박사 학위논문, 전북대학교 대학원.

이수원 (2011). *다양성과 창의성에 우리의 미래가 있다.* 월간 과학창의 2011년 2월호(통권

제161호). Retrieved from http://www.kofac.re.kr/ebook/monthly/201102/default1.html

이원규, 유헌창, 김현철, 정순영 (2003). *컴퓨터 교육론*. 서울: 홍릉과학출판사.

이원규, 정효숙 (2004). 초·중등과정에서의 컴퓨터과학교육의 역할과 필요성. *한국정보과학지, 22*(5). 31-34.

이은경 (2008). 로봇 활용 프로그래밍 학습이 창의적 문제해결성향에 미치는 영향. *대한공업교육학회, 33*(2), 120-136.

이은경 (2009). *Computational Thinking 능력 향상을 위한 로봇 프로그래밍 교수 학습 모형*. 박사 학위논문, 한국교원대학교.

이은경 (2013). 컴퓨터교과교육: 계산적 사고 향상을 위한 창의적 스크래치 프로그래밍 학습. *컴퓨터교육학회논문지, 16*(1), 1-9.

이재분, 현주, 류덕엽, 조성인 (2002). *초·중학생의 지적·정의적 발달수준 분석연구(III)*. 한국교육개발원.

이주호 (2013). *게이미피케이션 개념을 기반으로 한 미디어테크 계획*. 석사 학위논문, 홍익대학교 대학원.

이태옥 (2006). *마이크로 로봇 교육을 통한 초등학교 창의성 계발에 대한 연구*. 석사 학위논문, 제주교육대학교 교육대학원.

임화경, 조용남 (2011). Kodu 비주얼 프로그래밍 언어를 사용한 초등학생의 창의적 3D 게임프로그래밍 학습. *한국컴퓨터정보학회, 17*(11), 53-61.

장승권 (1996). 정보기술의 발전과 글쓰기로서의 컴퓨터 프로그래밍. *문화과학, 9*, 51-68.

전경원 (2000). *유아 종합 창의성 검사*. 서울: 학지사

전성균, 이영준 (2012). 초등학생의 확산적 사고 촉진을 위한 CPS 프로그래밍 수업의 효과 분석. *컴퓨터교육학회논문지, 15*(2), 1-8.

전성균, 서영민, 이영준 (2011). 컴퓨터교과교육과 컴퓨터과학: 창의성과 프로그래밍 교육에 관한 고찰. *한국컴퓨터교육학회 학술발표대회논문집, 15*(1), 73-77.

전현석 (2010). *언플러그드 알고리즘 학습이 영재 학생의 학업성취도에 미치는 영향*. 석사 학위논문, 한국교원대학교 교육대학원.

정미인, 정혜인, 정세영, 김영채 (2013). 2001-2012년 창의력관련 연구의 통합적 분석. *사고개발, 9*(1), 1-26.

정은아 (2005). *동화를 이용한 문제해결경험에서의 확산적 사고가 유아의 창의성에 미치는 영향*. 석사 학위논문, 단국대학교 교육대학원.

조문현 (2012). *스크래치 프로그래밍 교육에서 교수자 역할에 따른 메타인지 전략 지원의*

효과. 석사 학위논문, 경인교육대학교 교육대학원.

조성환, 송정범, 김성식, 이경화 (2008). CPS에 기반한 스크래치 EPL이 문제해결력과 프로그래밍 태도에 미치는 효과. *정보교육학회논문지, 12*(1), 77-88.

조준필 (2012). *기술교육에서 중학생의 논리적 사고력 함양을 위한 스크래치 학습 프로그램 개발*. 석사 학위논문, 한국교원대학교 대학원.

조현하 (2011). *고등학교 '정보'교과 교육에서 언플러그드 수업이 학업성취도에 미치는 영향: 전문계 고등학생을 대상으로*. 석사 학위논문, 고려대학교 교육대학원.

최성규, 정남용 (2003). 컴퓨터 애니메이션을 이용한 수업이 초등학생의 창의성 신장에 미치는 효과. *한국실과교육학회지, 16*(3), 35-54.

최훈 (2010). 김영정 교수의 비판적 사고론. *논리연구, 13*(2), 1-26.

한국교육평가학회 (2004). *교육평가용어사전*. 서울: 학지사.

한선관, 김수환, 서정보 (2010). 스크래치 프로그래밍을 활용한 게임중독 치료 프로그램의 개발. *정보교육학회논문지, 14*(1). 61-68.

함미옥, 홍영진 (2008). *이산수학*. 서울: 한빛미디어.

함성진, 양창모 (2011). 스크래치를 이용한 초등학교 컴퓨터 교육과정 설계. *정보교육학회논문지, 15*(3), 413-423.

허경철, 김홍원, 임선하, 김명숙, 양미경 (1991). *사고력 신장을 위한 프로그램 개발 연구 (V)*. 서울: 한국교육개발원.

Amabile, T. M. (1983). *The social psychology of creativity.* New York: Springer-Verlag.

Anderson, L. & Krathwohl, D. A. (2001). *Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives.* New York: Longman.

Baer (1993). *Divergent thinking and creativity: A task-specific approach* New Jersey: Lawrence Erlbaum Associates.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20-29.

Bennedssen, J., & Caspersen, M. E. (2008, September). Abstraction ability as an indicator of success for learning computing science?. *Proceedings of the Fourth international Workshop on Computing Education Research* (pp. 15-26). ACM. doi: 10.1145/1404520.1404523

Bennett, V. E., Koh, K., & Repenning, A. (2013, March). Computing creativity: divergence in computational thinking. *Proceedings of the 44th ACM technical symposium on Computer science education* (pp. 359-364). ACM. doi: 10.1145/2445196.2445302

Booch, G. (2005). The Unified Modeling Language User Guide (2nd Ed.). Pearson Education India.

Böszörmenyi, L. (2005). Teaching: People to people-about people: A plea for the historic and human view. In R. T. Mittermeir (Ed.), From computer literacy to informatics fundamentals, *Proceedings of the International Conference on Informatics in Secondary Schools-Evolution and Perspectives, ISSEP 2005* (pp. 93-102). New York: Springer. doi: 10.1007/978-3-540-31958-0_13

Bransford, G. D., & Stein, B. S. (1984). *The ideal problem solver.* New York: Freeman.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada.

Bruner, J. S. (1996). *The culture of education.* Harvard University Press.

Cangelosi, J. S. (1990). *Designing tests for evaluating student achievement.* New York: Longman.

Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bulletin, 38*(1), 27-31. doi: 10.1145/1124706.1121352

Clements, D. H. (1995). Teaching creativity with computers. *Educational Psychology Review, 7*(2), 141-161. doi: 10.1007/BF02212491

Cohen, A., & Haberman, B. (2007). Computer science: a language of technology. *ACM SIGCSE Bulletin, 39*(4), 65-69. doi: 10.1145/1345375.1345417

Conery, J. S. (2010a). Ubiquity symposium 'What is computation?': Computation is symbol manipulation. *Ubiquity, 2010(November)*, 4. doi: 10.1145/1880066.1889839

Conery, J. S. (2010b). *Explorations in Computing: An Introduction to Computer Science (Chapman & Hall/CRC Textbooks in Computing).* CRC Press.

Craft, A. (2010). *Creativity and Education Futures: Learning in a Digital Age.*

Trentham Books Ltd. Staffordshire: UK. doi: 10.1080/01411926.2011.605110

Cropley, A. J. (2001). *Creativity in education and learning: A guide for teachers and educators*. Psychology Press.

Csikszentmihalyi, M. (1997). *Finding flow: The psychology of engagement with everyday life*. Basic Books.

Cutts, Q., Esper, S., & Simon, B. (2011, August). Computing as the 4th R: a general education approach to computing education. *Proceedings of the seventh international workshop on Computing education research* (pp. 133-138). ACM. doi: 10.1145/2016911.2016938

Davis, M. (1958). *Computability and Unsolvability*. New York: McGraw-Hill.

Deek, F. P., Kimmel, H., & McHugh, J. A. (1998). Pedagogical changes in the delivery of the first-course in computer science: Problem solving, then programming. *Journal of Engineering Education, 87*(3), 313-320. doi: 10.1002/j.2168-9830.1998.tb00359.x

Denning, P. J. (2003). Great principles of computing. *Communications of the ACM, 46*(11), 15-20. doi: 10.1145/948383.948400

Denning, P. J. (2005). Is computer science science?. *Communications of the ACM, 48*(4), 27-31. doi: 10.1145/1053291.1053309

Denning, P. J. (2007). Computing is a natural science. *Communications of the ACM, 50*(7), 13-18. doi: 10.1145/1272516.1272529

Denning, P. J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30. doi: 10.1145/1516046.1516054

Denning, P. J. (2010). The great principles of computing. *American Scientist, 98*(5), 369-372. Retrieved from http://www.americanscientist.org/libraries/documents/20108101750328103-2010-09Denning-ComputingScience.pdf

Denning, P. J. (2010). Ubiquity symposium 'What is computation?': Opening statement. *Ubiquity, 2010(November)*, 1. doi: 10.1145/1880066.1880067

Denning, P. J., & McGettrick, A. (2005). Recentering computer science. *Communications of the ACM, 48*(11), 15-19. doi: 10.1145/1096000.1096018

Devlin, K. (2003, September). Why universities require computer science students to take math. *Communications of the ACM, 46*(9), 37-39.

Douadi, B., Tahar, B., & Hamid, S. (2012). Smart edutainment game for algorithmic thinking. *Procedia-Social and Behavioral Sciences, 31*, 454-458. doi: 10.1016/j.sbspro.2011.12.085

Ebel, R. L. (1965). Measuring educational achievement (pp. 421-424). Englewood Cliffs, New Jersey: Prentice-hall.

Ennis, R. H. (1989). *Critical thinking and subject specificity: Clarification and needed research.* Educational Researcher, 18(3), 4-11. doi: 10.3102/0013189X018003004

Fadi, P. D., & McHugh, J. A. (2001). Prototype software development tools for beginning programming. *Journal of Computer Science Education, 14*, 14-20.

Feldhusen, J. F. (1983). The Purdue Creative Thinking Program. In I. S. Sato (Ed.), *Creativity research and educational planing* (pp.41-46). Louisiana: National State Leadership Training Institute for the Gifted and Talented.

Felleisen, M., & Krishnamurthi, S. (2009). Viewpoint Why computer science doesn't matter. *Communications of the ACM, 52*(7), 37-40. doi: 10.1145/1538788.1538803

Flowers, T. R., & Gossett, K. A. (2002). Teaching problem solving, computing, and information technology with robots. *Journal of Computing Sciences in Colleges, 17*(6), 45-55.

Foster, A. L. (2005). Student interest in computer science plummets. The Chronicle of Higher Education, 51(38), A31-A32. Retrieved from http://search.proquest.com /docview/214686068?accountid=10066

Fraser, M. D., Kumar, K., & Vaishnavi, V. K. (1994). Strategies for incorporating formal specifications in software development. *Communications of the ACM, 37*(10), 74-86. doi: 10.1145/194313.194399

Gal-Ezer, J., & Zeldes, A. (2000). Teaching software designing skills. *Computer Science Education, 10*(1), 25-38.

Gall, M. D., Gall, J. P., & Borg, W. (2003). Educational research (7th ed.). New York: Longman.

Gallardo, D., Julia, C. F., & Jorda, S. (2008, October). TurTan: A tangible programming language for creative exploration. In Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. *3rd IEEE International Workshop on* (pp. 89-92). IEEE. doi: 10.1109/TABLETOP.2008.4660189

Gardner, H. (1983). *The Frames of Mind.The theory of multiple intelligence*. New York: BasicBooks.

Gardner, H. (1993). *Multiple intelligences: The theory in practice*. New York: BasicBooks.

Glass, R. L. (2006). *Software Creativity 2.0*. Atlanta: Developer.* Books.

Gold, J. B. (1981). *Developing the creative problem solving skills of intermediate age educable mentally retarded student*. Doctoral dissertation, Fordham University.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013, October). First year student performance in a test for computational thinking. 알고리즘적 사고의 양식이 *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference* (pp. 271-277). ACM. doi: 10.1145/2513456.2513484

Greitzer, F. L., Kuchar, O. A., & Huston, K. (2007). Cognitive science implications for enhancing training effectiveness in a serious gaming context. *Journal on Educational Resources in Computing, 7*(3), 2. doi: 10.1145/1281320.1281322

Grier, D. A. (2005). *When computers were human (Vol. 316)*. Princeton: Princeton University Press.

Gruhn, V., & Laue, R. (2007, August). On experiments for measuring cognitive weights for software control structures. *In Cognitive Informatics, 6th IEEE International Conference on* (pp. 116-119). IEEE.

Gu, M., & Tong, X. (2004). Towards hypotheses on creativity in software development. *In Product Focused Software Process Improvement* (pp. 47-61). Springer Berlin Heidelberg.

Guilford, J. P. (1950). Creativity. *American Psychologist, 5*(9), 444-454.

Guilford, J. P. (1956). Structure of intellect, *Psychological Bulletin, 53*, 267-293.

Guilford, J. P. (1967). *The nature of human intelligence*. New York: McGraw-Hill.

Guilford, J. P. (1967). *The nature of human intelligence*. New York: McGraw-Hill.

Haberman, B. (2004). High-school students' attitudes regarding procedural abstraction. *Education and Information Technologies, 9*(2), 131-145.

Hayes, G. R. (1981). *The complete problem solver*. Philadelphia: Franklin Institute Press.

Hazzan, O. (1999). Reducing abstraction level when learning abstract algebra concepts. Educational Studies in Mathematics, 40(1), 71-90. doi: 10.1023/A:1003780613628

Isaken, S. G., Puccio, G. J., & Treffinger, D. J. (1993). An ecological approach to creativity research: profiling for creative problem solving. *The Journal of Creativity Behavior, 27*(3), 149-170.

Jacobson, I. (1999). *The unified software development process*. Addison-Wesley Object Technology Series.

Jenkins, T. (2002). On the Difficulty of Learning to Program. *Proceedings of 3rd LTSN-ICS Conference*, 53-58.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science*. Harvard University Press.

Kafai, Y. B., & Burke, Q. (2013, March). The social turn in K-12 programming: moving from computational thinking to computational participation. *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 603-608). ACM. doi: 10.1145/2445196.2445373

Kelley, D. (1995). *Automata and Formal Languages: an introduction*. Prentice-Hall.

Knobelsdorf, M., & Romeike, R. (2008). Creativity as a pathway to computer science. *ACM SIGCSE Bulletin, 40*(3), 286-290. doi: 10.1145/1597849.1384347

Koh, K. H. (2011, September). Computing indicators of creativity. *In Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on* (pp. 231-232). IEEE. doi: 10.1109/VLHCC.2011.6070407

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. *In Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on* (pp. 59-66). IEEE. doi: 10.1109/VLHCC.2010.17

Koppelman, H., & van Dijk, B. (2010, June). Teaching abstraction in introductory courses. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 174-178). ACM. doi: 10.1145/1822090.1822140

Koschmann, T. D., Feltovich, P. J., Myers, A. C., & Barrows, H. S. (1992). Implications of CSCL for problem-based learning. *ACM SIGCUE Outlook, 21*(3), 32-35. doi: 10.1145/130893.130902

Koshy, T. (2001). *Fibonacci and lucas numbers with applications*. New York: John Wiley & Sons.

Kramer, J. (2007). Is abstraction the key to computing?. Communications of the ACM, 50(4), 36-42. doi: 10.1145/1232743.1232745

Kurdek, L. A. (1977). Structural components and intellectual correlates of cognitive perspective taking in first- through fourth-grade children. *Child Development, 48*, 1503-1511.

Landry, M., & Lyons-Ruth, K. (1980). Recursive structure in cognitive perspective taking. *Child Development, 51*, 386-394.

Law, E., & Ahn, L. V. (2011). Human computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning, 5*(3), 1-121. doi: 10.2200/S00371ED1V01Y201107AIM013

Lewandowski, G., Johnson, E. & Goldweber, M. (2005). Fostering a Creative Interest in Computer Science. *Proceedings of the 36th SIGCSE technical symposium on Computer science education* (pp. 535-539). ACM. doi: 10.1145/1047344.1047512

L'Heureux, J., Boisvert, D., Cohen, R., & Sanghera, K. (2012, October). IT problem solving: an implementation of computational thinking in information technology. *Proceedings of the 13th annual conference on Information technology education* (pp. 183-188). ACM. doi: 10.1145/2380552.2380606

LI, D. N., & CHEN, H. M. (2007). Discussion and Practice on Fostering Science Creativity of Undergraduates in Computer Science Subject. *Journal of Beijing Institute of Technology (Social Sciences Edition), S1*. 3-4.

Long, J. (2007). Just for fun: Using programming games in software programming training and education-A field study of IBM robocode community. *Journal of Information Technology Education, 6*, 279-290.

Lubart, T. M. (1999). *Componential Models*. In M. A. Runco, & S. R. Pritzker. (Eds.). Encyclopedia of Creativity: Vol. 1 (pp. 295-300). San Diego, CA: Academic Press.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. ACM SIGCSE Bulletin, 40(1), 367-371. doi: 10.1145/1352322.1352260

Means, H. W. (1991). Using Literature in a Computer Science Service Course: Improving Abstract/Critical Thinking Skills. *Journal of Computing Sciences in Colleges, 6*(5), 30-34.

Merrick, K. E. (2010). An empirical evaluation of puzzle-based learning as an interest approach for teaching introductory computer science. *Education, IEEE Transactions on, 53*(4), 677-680. doi: 10.1109/TE.2009.2039217

Minsky, M. L., & Minsky, M. (1968). *Semantic information processing* (Vol. 142). Cambridge, MA: MIT press.

Misra, S. (2006). A complexity measure based on cognitive weights. *International Journal of Theoretical and Applied Computer Sciences, 1*(1). 1-10.

Misra, S., & Misra, A. K. (2004, August). Evaluating cognitive complexity measure with Weyuker properties. *Proceedings of the Third IEEE International Conference on* (pp. 103-108). IEEE.

Montemayor, J., Druin, A., Farber, A., Simms, S., Churaman, W., & D'Amour, A. (2002). Physical programming: Designing tools for children to create physical interactive environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 299-306). doi: 10.1145/503376.503430

Morris, D., & Secretan, J. (2009, April). Computational creativity support: Using algorithms and machine learning to help people be more creative. *In CHI'09 Extended Abstracts on Human Factors in Computing Systems* (pp. 4733-4736). ACM.

National Research Council (NRC) (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: The National Academies Press. Retrieved from http://www.nap.edu/catalog.php?record_id=12840

National Research Council (NRC) (2011). *Report of a workshop on the pedagogical aspects of computational thinking*. Washington, DC: The National Academies Press. Retrieved from http://www.nap.edu/catalog.php?record_id=13170

Navlakha, S., & Bar-Joseph, Z. (2011). Algorithms in nature: the convergence of

systems biology and computational thinking. *Molecular systems biology, 7*(1). doi: 10.1038/msb.2011.78

Nunnally, J. C. (1978). *Psychometric theory* (2nd Ed.). New York: McGraw Hill.

Nwana, H. S. (1997). Is computer science education in crisis? *ACM Computing Surveys (CSUR), 29*(4), 322-324. doi: 10.1145/267580.267582

Oppenheimer, L. (1978). The development of the processing of social perspectives: A cognitive model. *International Journal of Behavioral Development, 1*(2), 149-171. doi: 10.1177/016502547800100204

Orr, G. (2009, August). Computational thinking through programming and algorithmic art. *In SIGGRAPH 2009: Talks* (p. 31). ACM. doi: 10.1145/1597990.1598021

Osborn, A.F. (1957). Applied Imagination. principles and procedures of Creative Thinking (2nd Ed.). New York: Scribner.

Parnes, S. J. (1967). Creative behavior guidebook. New York: Scribner.

Patterson, D. A. (2005). Restoring the popularity of computer science. *Communications of the ACM, 48(9), 25-28*. doi: 10.1145/1081992.1082011

Pattis, R. E. (1981). *Karel the robot: a gentle introduction to the art of programming*. John Wiley & Sons, Inc..

Piaget, J., & Inhelder, B. (1969). *The psychology of the child*. New Yorks: BasicBooks.

Plucker, J. A. (1998). Beware of simple conclusions: The case for content generality of creativity. Creativity Research Journal, 11(2), 179-182. doi: 10.1207/s15326934crj1102_8

Runco, M. A. (2001). Introduction to the special issue: Commemorating Guilford's 1950 presidential address. *Creativity Research Journal, 13*(3-4), 245. doi: 10.1207/S15326934CRJ1334_01

Prensky, M. (2004). Proposal for educational software development sites: an open source tool to create the learning software we need. *On the Horizon, 12*(1), 41-44. doi: 10.1108/10748120410699585

Pring, R. et al. (2012). *Education for All: The Future of Education and Training for 14-19 Year-Olds*. Routledge.

Resnick, M. (2007, June). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of the 6th*

ACM SIGCHI conference on Creativity & cognition (pp. 1-6). ACM. doi: 10.1145/1254960.1254961

Resnick et al. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67. doi: 10.1145/1592761.1592779

Roadrangka, V., Yeany, R. H., & Padilla, M. J. (1983, April). The construction and validation of Group Assessment of Logical Thinking (GALT). *Paper presented at the annual meeting of the National Association for Research in Science Teaching*, Dallas, Texas.

Romeike, R. (2006, February). Creative students: what can we learn from them for teaching computer science?. *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006* (pp. 149-150). ACM. doi: 10.1145/1315803.1315835

Romeike, R. (2007a, June). Three drivers for creativity in computer science education. *Proceedings of the IFIP-Conference on Informatics, Mathematics and ICT: a golden triangle*.

Romeike, R. (2007b, November). Applying creativity in CS high school education: criteria, teaching example and evaluation. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88* (pp. 87-96). Australian Computer Society, Inc..

Rosenthal, R., & Rosnow, F. L. (1975) *The volunteer subject*. New York: Wiley.

Rowe, A. J. (2004). *Creative intelligence*. Financial Times/Prentice Hall.

Runco, M. A. (1989). The creativity of children's art. *Child Study Journal, 19*, 177-190.

Runco, M. A. (2007). *Creativity: Theories and Themes; Research, Development, and Practice*. Amsterdam: Elsevier.

Runco, M. A. & Okuda, S. M. (1988). Problem discovery, divergent thinking, and the creative process. J*ournal of Youth and Adolescence, 17*(3), 211-220.

Schwill, A. (1994). Fundamental ideas of computer science. Bulletin-European Association for Theoretical Computer Science, 53, 274-274.

Schwill, A. (1997). Computer science education based on fundamental ideas. *Samways, Brain (Hrsg.): Information Technology - Supporting change through teacher*

education. London: Chapman & Hall, 285-291.

Seiter, L., & Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59-66). ACM. doi: 10.1145/2493394.2493403

Selby, C. C. (2012, November). Promoting computational thinking with programming. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 74-77). ACM. doi: 10.1145/2481449.2481466

Settles, B. (2010, June). Computational creativity tools for songwriters. *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity* (pp. 49-57). Association for Computational Linguistics.

Shaeffer, E. M. (2009). *Shifting Perspectives: Point of view in visual images affects abstract and concrete thinking.* Doctoral dissertation, Fordham University.

Shao, J., & Wang, Y. (2003). A new measure of software complexity based on cognitive weights. *Electrical and Computer Engineering, Canadian Journal of, 28*(2), 69-74.

Shneiderman, B. (2000). Creating creativity: user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction (TOCHI), 7*(1), 114-138. doi: 10.1145/344949.345077

Smith, P. K., & Trope, Y. (2006). You focus on the forest when you're in charge of the trees: power priming and abstract information processing. *Journal of personality and social psychology, 90*(4), 578.

Springer, G., & Friedman, D. P. (1990). *Scheme and the Art of Programming*. McGraw-Hill, Inc..

Stouffer, W. B., Russell, J. S., & Oliva, M. G. (2004). Making the strange familiar: Creativity and the future of engineering education. *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition* (pp. 20-23). Retrieved from https://www.engr.wisc.edu/cee/faculty/russell_jeffrey/004.pdf

Tarkan, S. et al. (2010, April). Toque: designing a cooking-based programming language for and with children. *Proceedings of the SIGCHI Conference on*

Human Factors in Computing Systems (pp. 2417-2426). ACM. doi: 10.1145/1753326.1753692

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education (TOCE), 12*(2), 8. doi: 10.1145/2160547.2160551

Thomas, J. C., Lee, A., & Danis, C. (2002). Enhancing Creative Design via Software Tools. *Communications of the ACM, 45*(10), 112-115. doi: 10.1145 /570907.570944

Torrance, E. P. (1957). *Psychology of survival*. Unpublished manuscript, Air Force Personnel Research Center, Lackland Air Force Base, Texas.

Torrance, E. P. (1966). *Torrance tests of creative thinking: Norms-technical manual*. New Jersey: Personnel Press.

Torrance, E. P. (1974). *Torrence tests of creativity thinking: Norms, technical manual*. Princeton, New Jersey: Personnel Press/Ginn.

Torrance, E. P. (1995). *Why fly?: A philosophy of creativity*. New Jersey: Ablex publishing Corporation.

Touretzky, D. S., Marghitu, D., Ludi, S., Bernstein, D., & Ni, L. (2013, March). Accelerating K-12 computational thinking using scaffolding, staging, and abstraction. *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 609-614). ACM. doi: 10.1145/2445196.2445374

Treffinger, D. J., Sortore, M. R., & Firestien, R. L. (1983). Theoretical perspective on creative and its facilitation: An overview. *The Journal of Creative Behavior, 17*(1), 9-17. Retrieved from http://www.cpsb.com/research/articles/creative-problem-solving /Theoretical-Perspectives-Creative-Learning-Facilitation.pdf

Trope, Y., & Liberman, N. (2003). Temporal construal. *Psychological Review, 110*(3), 403-421.

Tucker, A., Seehorn, D., Carey, S., Moix, D., Fuschetto, B., Lee, I., O'Grady-Cuniff, D., Stephenson, C., & Verno, A. (2011). *CSTA K-12 Computer Science Standards. Revised 2011*. CSTA Standards Task Force. Retrieved from http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society, 42*(2),

230-265. Retrieved from http://classes.soe.ucsc.edu/cmps210/Winter11/Papers/turing-1936.pdf

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433-460.

Valente, A., & Marchetti, E. (2011, July). Programming Turing Machines as a Game for Technology Sense-Making. *In Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on* (pp. 428-430). IEEE. doi: 10.1109/ICALT.2011.134

Walden, J., Doyle, M., Garns, R., & Hart, Z. (2013, July). An informatics perspective on computational thinking. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 4-9). ACM. doi: 10.1145/2462476.2483797

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The Fairy Performance Assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). ACM. doi: 10.1145/2157136.2157200

Wilson, E. O. (1999). *Consilience: The unity of knowledge* (No. 31). Random House Digital, Inc..

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3). 33-35. doi: 10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881). 3717-3725. doi: 10.1098/rsta.2008.0118

Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational Thinking and Expository Writing in the Middle School. *ACM Transactions on Computing Education (TOCE), 11*(2), 9. doi: 10.1145/1993069.1993073

Zendler, A., & Klaudt, D. (2012). Central computer science concepts to research-based teacher training in computer science: An experimental study. *Journal of Educational Computing Research, 46*(2), 153-172. doi: 10.2190/EC.46.2.c

Zendler, A., & Spannagel, C. (2008) Empirical foundation of central concepts for computer science education. *ACM Journal on Educational Resources in*

Computing, 8(2), 6. doi: 10.1145/1362787.1362790

Zendler, A., Spannagel, C., & Klaudt, D. (2008). Process as content in computer science education: Empirical determination of central processes. *Computer Science Education, 18*(4), 231-245. doi: 10.1080/08993400802390553

<ABSTRACT>

Programming Education Program based on PPS to Improve Computational Thinking Ability

ByeongSul Kim

Major of Computer Education, Faculty of Science Education

Graduate School, Jeju National University

Supervised by professor JongHoon Kim

This study aimed to develop the programming education program based on PPS and verify whether it can improve learners' computational thinking ability. Paper-and-pencil Programming Strategy(PPS) is a modeling strategy of representing an idea logically by diagrams, sentences, codes, flowcharts, tables or any other meaningful representation that can be created using paper and pencil.

The purpose of this study is verifying whether the developed programming education program based on PPS can improve learners' computational thinking ability.

For this purpose, this study conducted following as below sequences.

First, 6 fundamental factors of computational thinking — Abstract thinking, Critical thinking, Logical thinking, Creative thinking, Recursive thinking, Algorithmic thinking — were extracted from the relative researches. In this study, the diagram about the relation of computational thinking and creativity, and the relationship of computational thinking's factors was proposed. This analysis could be the base for developing the education program and questions in assessment tool.

In the process of developing the education program, first, models of instruction

were considered. In this education program, the sequence of the instruction is 'Analysis the computation model', 'Make computation', 'Find and analysis Problem, and find idea', 'Find solution', and 'Find acceptance', which was developed by modifying and supplementing the Creative Problem Solving (CPS) model. In the process of developing learning contents, 10 core concepts of computer science focusing programming — Sequence, Condition/Branch, Iteration, Concurrency, Variable, Random, Algorithm, Object, Function, Recursive — were extracted. And then, based on these developed instruction model and learning contents, the programming education program was developed for utilizing PPS activity actively in the step of solution design of problem-solving.

For the assessment of learners' computational thinking ability, it needed to be divided two parts - computational creativity and computational cognition. For the assessment of computational creativity, Torrance's TTCT test was used. For the assessment of computational cognition(abstract thinking, critical thinking, logical thinking, recursive thinking and algorithimic thinking), a new test of computational cognition ability is needed to develop in this study. Its parallel-form assessment tool (A and B type) was developed in order of making questions, verifying validity, discrimination, difficulty and reliability, modifying question, and verifying parallel-form reliability. The developed assessment tool has high correlation with A and B type, and existing Group Assessment of Logical Thinking (GALT). The learners' computational thinking ability was measured using this assessment tool in this study.

The results of this study were summarized as follows:

First, the programming educational program affect learners' computational cognition ability positively.

Second, the programming educational program affect learners' computational creativity positively.

This study could have positive contribution to computer science education as follows:

First, the programming education using PPS activity has the possibility of improving learners' computational cognition ability and computational creativity. It's not only a method of unplugged learning as a prerequisite learning. It proposes that we need to focus on the solution design more in problem-solving of programming. And, PPS activity could be used actively in various problem-solving activities.

Second, the result of this study could be a evidence for the diagram about the relation of computational thinking and its factors.

Third, the developed assessment tool of computational thinking could be the reproductive product for the further relative researches.

This study has some limitations.

First, the comparative group was designed without any treatment. It means that the result of this study could not verify the only educational effect of PPS activity in programming.

Second, TTCT (Verbal) could not measure the 'Elaboration' of creativity.

In addition, it need to verify the result again to establish the conclusion of this study.

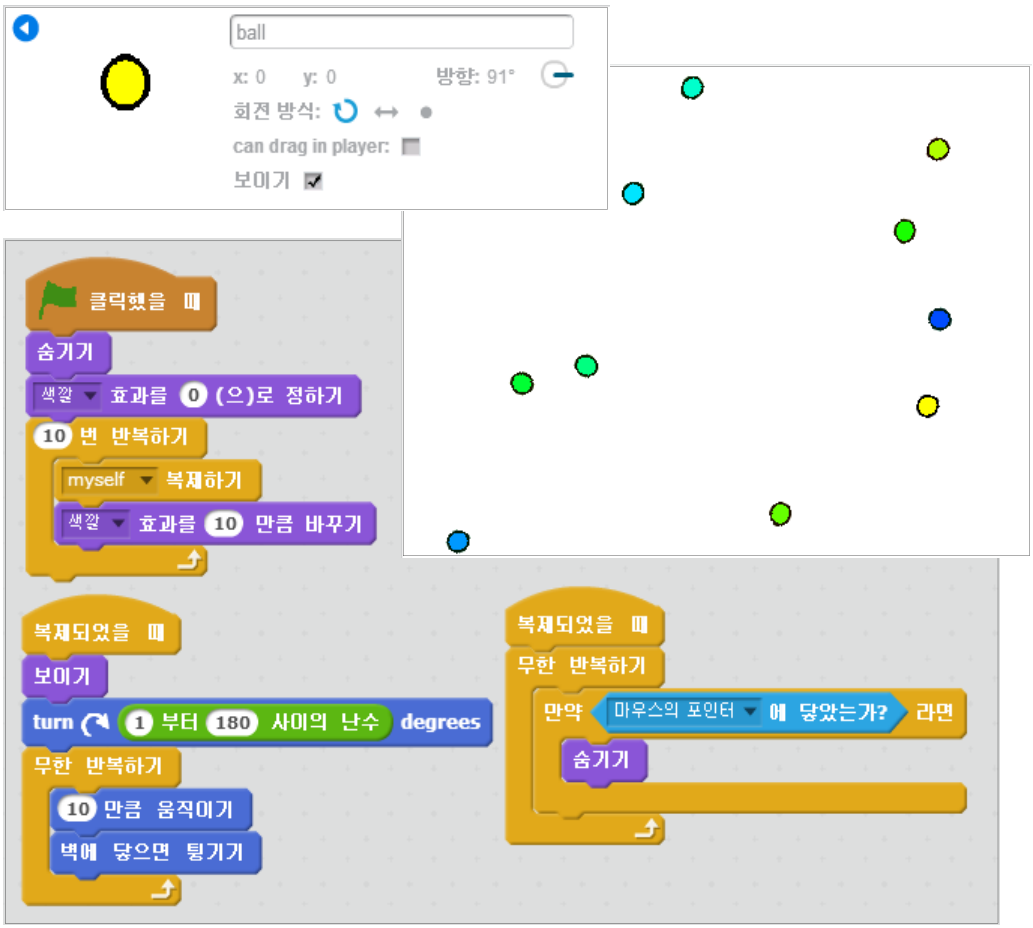
First, the assessment of computational thinking ability need more experts group's validity. And it need to be applied to a large testers group.

Second, the experiment group has 17 participants. It's sufficient for the comparison experiment research, but not sufficient for the correlate research. For finding the relation of computational thinkgin and creativity, it need to be applied to a large participants group.

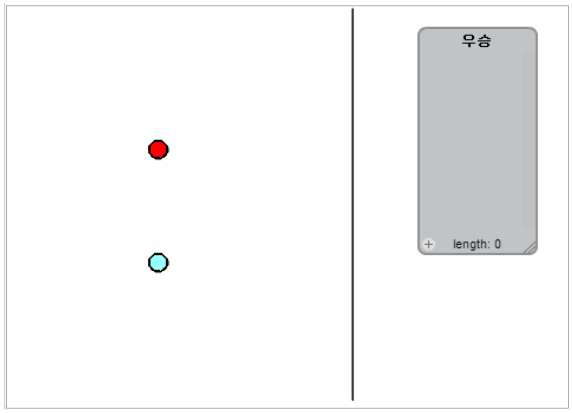
Keyword: PPS, Paper-and-pencil Programming Strategy, Computational thinking, Computational cognition ability, Computational creativity, Programming

부 록

<부록 1> 스크래치 스크립트



<부록 2> 스크래치 스크립트

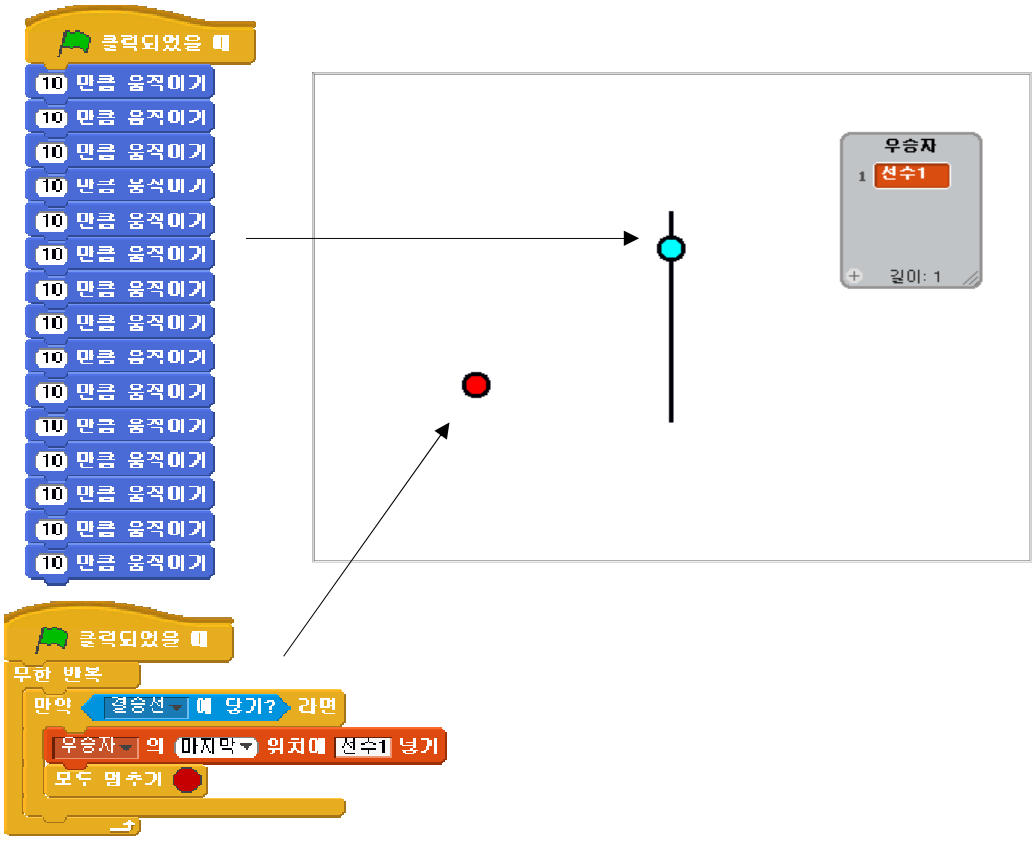


선수1

선수2

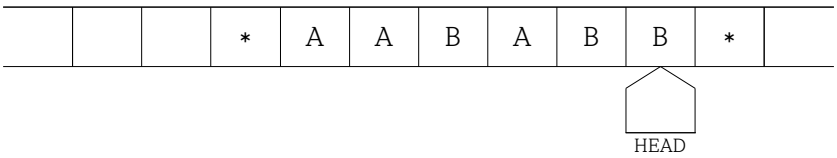
결승선

<부록 3> 스크래치 스크립트



<부록 4> 튜링머신 문제

초기 상태



튜링 머신의 테이프에는 3가지 기호(A, B, *(데이터 없음))가 쓰여져 있다. 우리는 다음의 요구사항에 맞는 시스템을 개발하려고 한다.

• 튜링머신의 헤더가 왼쪽으로 움직일 때 현재 셀과 같은 알파벳이 나오면 이들을 모두 기호 S로 바꿔어야 한다. 하지만, 현재 셀과 다르다면 현재의 셀은 기호 P로 바꿔어야 한다.

• 헤더가 어떠한 셀에서 기호 *를 만나면 시스템 전체의 작동은 멈추게 된다.

• 튜링머신의 헤더의 시작점의 오른쪽에는 기호 *가 있으며 왼쪽으로 이동하며 시작된다.

예를 들어 상태 1의 각 기호들이 각 셀에 있다고 하나씩 있다고 가정할 때 이는 상태 2처럼 바꿔어져야 한다(_는 헤더의 위치를 나타낸다).

상태 1: * A A B A B B *

상태 2: * S S P P S S *

같은 원리로 상태 3은 상태 4로 바꿔어져야 할 것이다.

상태 3: * A B B B A A B *

상태 4: * P S S S S S P *

<부록 5> 사전·사후 집단 간 독립표본 t검정

사전·사후 집단 간 독립표본 t검정

하위요소	기간	집단	M	SD	t	Sig. (2-tailed)
보존	사전	LOGO	3.60	.627	1.615	.109
		PPS	3.38	.782		
	사후	LOGO	3.67	.579	.880	.381
		PPS	3.56	.714		
비례	사전	LOGO	5.09	.986	–.090	.928
		PPS	5.11	1.117		
	사후	LOGO	5.38	.871	–.589	.557
		PPS	5.47	.742		
변인통제	사전	LOGO	3.53	.604	–.301	.764
		PPS	3.56	.660		
	사후	LOGO	3.58	.738	–.573	.568
		PPS	3.65	.584		
확률	사전	LOGO	1.65	.552	–.864	.389
		PPS	1.75	.552		
	사후	LOGO	1.73	.525	.000	1.000
		PPS	1.73	.525		
상관	사전	LOGO	.33	.610	–1.123	.264
		PPS	.47	.742		
	사후	LOGO	.58	.786	–1.052	.295
		PPS	.75	.844		
조합	사전	LOGO	2.65	.584	–1.045	.298
		PPS	2.76	.508		
	사후	LOGO	2.85	.356	–.465	.643
		PPS	2.89	.458		
전체	사전	LOGO	16.85	1.890	–.463	.645
		PPS	17.04	2.219		
	사후	LOGO	17.80	1.909	–.776	.439
		PPS	18.05	1.508		

N = 55, df = 108

<부록 6> 사전·사후 집단 내 대응표본 t검정

사전·사후 집단 내 대응표본 *t*검정

집단	대응 사전 - 사후	대응차		t
		M	SD	
LOGO	보존	–.073	.466	–1.158
	비례	–.291	1.227	–1.758
	변인통제	–.055	.848	–.477
	확률	–.073	.690	–.782
	상관	–.255	.775	–2.436*
	조합	–.200	.650	–2.283*
	합계	–.945	1.671	–4.195***
PPS	보존	–.182	.641	–2.104*
	비례	–.364	1.060	–2.543*
	변인통제	–.091	.752	–.896
	확률	.018	.561	.240
	상관	–.273	.781	–2.591*
	조합	–.127	.546	–1.728
	합계	–1.018	2.415	–3.127**

*$p < .05$ (two-tailed), **$p < .01$ (two-tailed), ***$p < .001$ (two-tailed), N = 55, $df = 54$.

<부록 7> 계산적 사고의 이해도 비교

계산적 사고의 이해도 비교

구분	LOGO	PPS	*t*
	평균 (편차)	평균 (편차)	
사전설문	1.655 (.732)	1.582 (.469)	.620
사후설문	2.673 (.783)	3.300 (.785)	–4.195**

**$p < .01$ (two-tailed), $df = 108$.

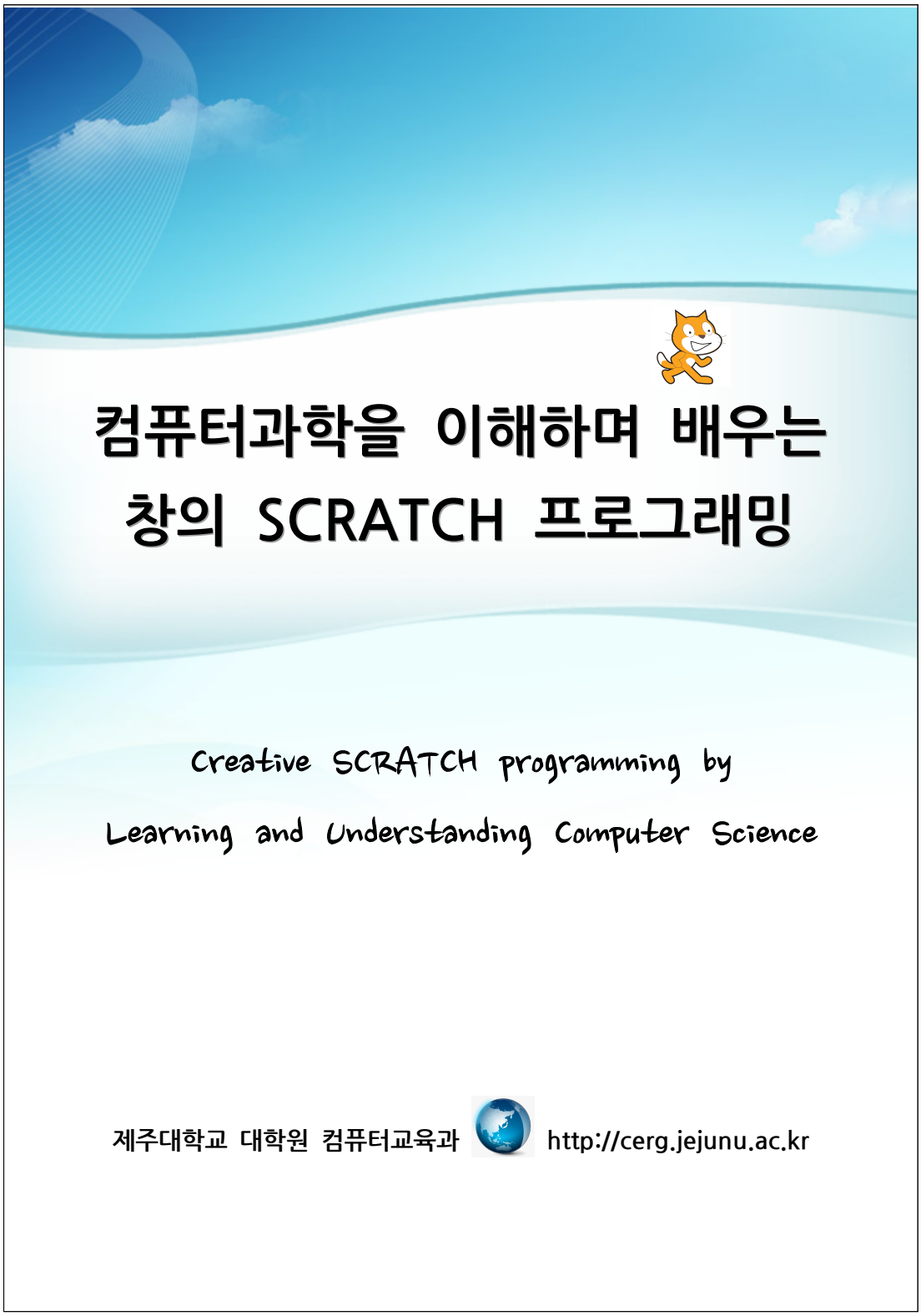
<부록 8> 컴퓨터과학의 흥미도 비교

컴퓨터과학의 흥미도 비교

구분	LOGO	PPS	*t*
	평균 (편차)	평균 (편차)	
사전설문	3.182 (.709)	3.191 (.778)	–.064
사후설문	3.655 (.576)	3.955 (.618)	–2.632*

*$p < .05$ (two-tailed), $df = 108$.

<부록 9> 교재



자장면 배달의 달인

자장면 배달을 하려고 한다. 짜장면 가게는 아래의 그림에서 ◉에 위치하고 있다.

자장면 배달은 걸어서 가도 되고, 자전거나 오토바이를 이용할 수 있다.

문제를 해결해 봅시다.

한 명의 배달원이 (가), (나), (다), (라)의 집에 자장면들을 배달하고자 할 때, 배달 순서에 상관없이 가장 빠르게 다녀오기 위한 방법을 표현해보세요. 어떤 이동수단을 사용할지 어떤 순서로 이동할지를 다른 사람이 이해할 수 있도록 표현해보세요.

1. 스크래치 시작하기

1) 스크래치 화면구성 살펴보기 (10분)

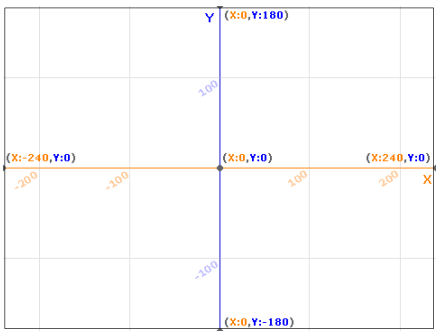
① 메뉴 ② 툴바 ③ 보기 모드
④ 무대 ⑤ 좌표
⑥ 스프라이트와 스크립트/모양/소리 탭
⑦ 스프라이트 그림판
⑧ 블록 팔레트
⑨ 무대와 레이어의 개념 이해하기

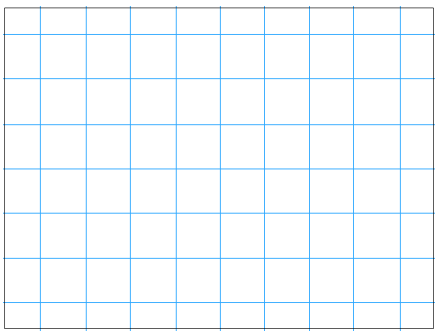
2) 동작 블록으로 스크립트 프로그래밍 시작하기

① 스프라이트 이름 지정하기
② 잠금 설정
③ 방향선과 방향정보

3) 동작블록 이해하기

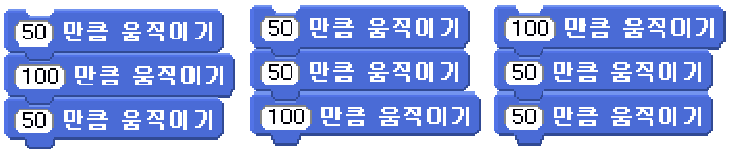
① 무대와 좌표





실제 스크래치의 무대는 왼쪽 그림의 좌표처럼 만들어져 있지만 더 쉽게 이해하기 위해서는 오른쪽 그림의 격자 좌표로 생각하는 것이 좋다. 오른쪽으로 갈수록 x좌표가, 위로 갈수록 y좌표가 커진다는 것을 기억해 두어야 한다.

② 블록 조합하기 연습하기 (‘볼’ 스프라이트 사용하기)

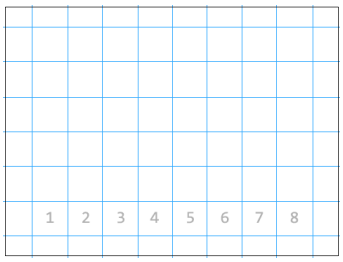


③ 클릭-실행 기법을 통해 동작과 순서의 개념 이해하기

(두가지 방법 모두 같은 결과를 보여준다.)

④ 블록 복사하여 이용하기 / 스프라이트 복사하기 / 스크립트 복사하기

창의적으로 생각하기 #1



옆의 그림처럼 좌표 50씩 나누어진 격자 배경에서 ‘볼’ 스프라이트를 1의 위치에 가져다 놓고 현재 방향을 오른쪽을 향하게 한다. 이 ‘볼’ 스프라이트를 8의 위치까지 움직이게 하려고 한다.

동작블록의 

만을 이용하려고 할 때, 어떻게 8까지 가게 할 수 있을까?

블록의 ( ) 안의 수는 50, 100, 150, 200까지만 사용할 수 있다고 가정하자. 다양한 방법을 찾아낼 수 있는가? 생각나는 대로 방법들을 써보자.

방법 예시 200만큼 움직이기 200만큼 움직이기	**방법 1**	**방법 2**
방법 3	**방법 4**	**방법 5**
방법 6	**방법 7**	**방법 8**

⑤ 

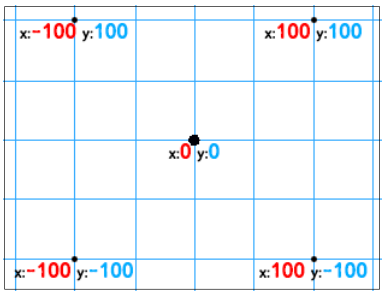
이용하기

좌표를 굳이 사용하지 않아도 된다. 대신 기준점(0,0)을 잘 활용하자.

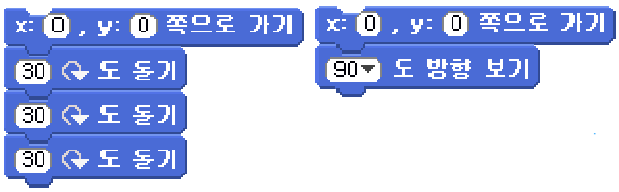
⑥ ( )초 동안 좌표로 움직이기

창의적으로 생각하기 #2

•'볼' 스프라이트가 아래의 좌표 배경에서 움직이며 모든 점들을 지나 원위치(0,0)로 돌아오는 스크립트를 만들어보자. '( )초 동안 x:( ), y:( )쪽으로 움직이기'를 활용하여라. 자신의 스크립트를 아래에 써보자.



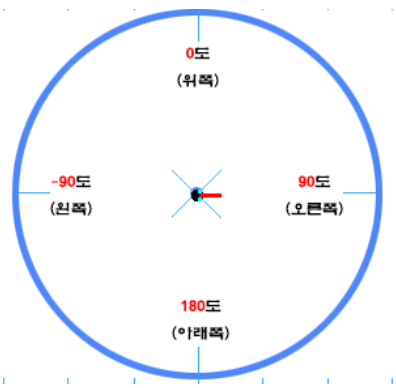
⑦ 회전이용하기 / 방향보기



- '회전하기'와 '( )도 방향보기'의 차이점 알기

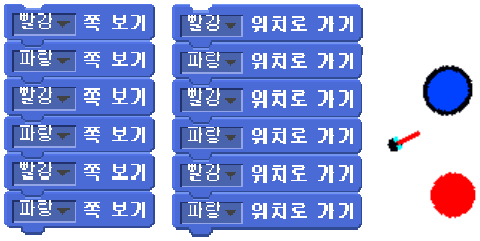
(결과는 같지만 실행단계의 차이를 보인다.)

- 단계별 실행을 설정하여 차례대로 실행되는 모습을 확인하기)



⑧ [ ]쪽 보기, [ ]위치로 가기

파랑, 빨강 스프라이트를 만든다.



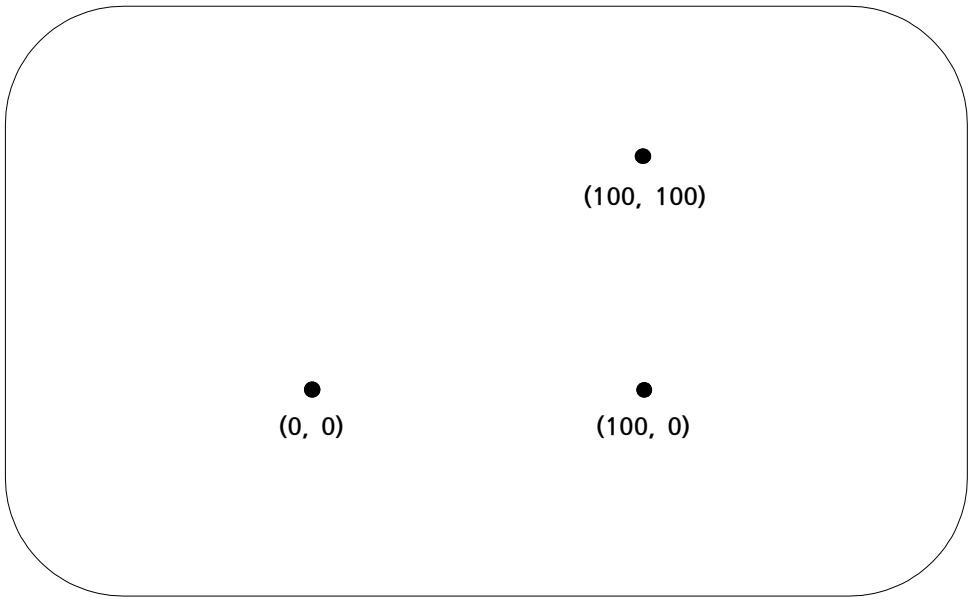
※ 마우스 포인터쪽 보기 활용

⑨ 좌표 정하기와 좌표 바꾸기의 차이 알기

창의적으로 생각하기 #3

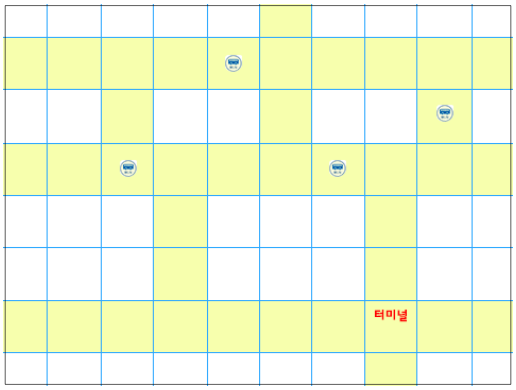
•'볼' 스프라이트가 좌표 (0, 0)에서 좌표 (100, 0)을 거쳐 좌표 (100, 100)으로 움직이는 방법을 다양한 방법을 생각해보자. 남들이 생각 못 했을 것 같은 자신만의 방법이 있는가? **스크립트를 아래에 써보자.**

(스프라이트가 이동하는 모습이 보여야 한다.)



버스 운전하기 #1

나는 무인버스(운전사 없이 운행되는 버스)를 만드는 과학자이다. 무인버스는 처음에 터미널에 있다가 버스 정류장() 이 있는 곳을 모두 지나치고 터미널로 다시 돌아와야 한다. 버스는 처음 90도(오른쪽)을 향하고 있다.



무인버스를 만드는 컴퓨터 프로그램은 "스크래치 버튼 프로그램"이다. 아래의 3개의 버튼만 사용할 수 있다.

[1]번 버튼을 누르면 

가 실행되고,

[2]번 버튼을 누르면 

(시계방향으로 90도 회전)이 실행되고,

[3]번 버튼을 누르면 

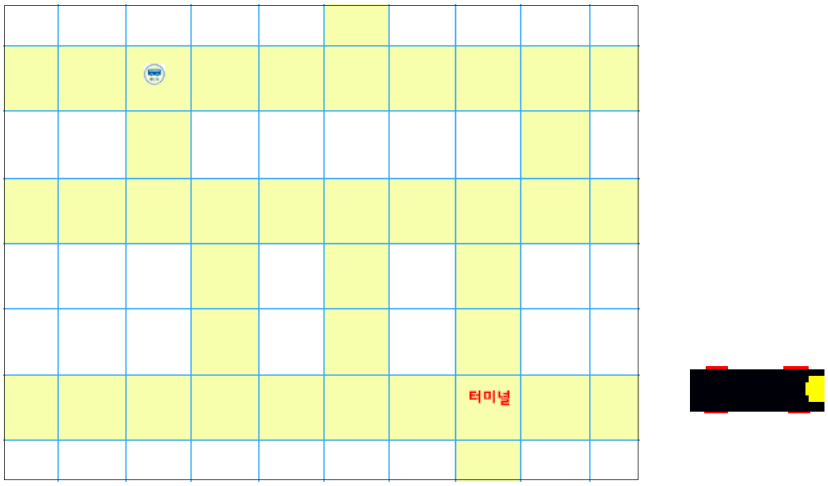
(반시계방향으로 90도 회전)이 실행된다.

버스의 기름을 아끼기 위해서 가장 짧은 경로(거리)로 모든 정류장을 다녀올 수 있도록 프로그래밍 해보자. 버튼을 어떤 순서로 누르면 될까? 버튼 번호를 아래의 표에 써보자.

순서	1	2	3	4	5	6	7	8
번호								
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44

버스 운전하기 #2

터미널에 있는 버스가 정류장까지 가려고 한다. 버스는 현재 터미널에 위치하고 있고 90도(오른쪽)을 향해 있다.



문제 해결에 사용할 수 있는 블록은 총 2개이다.

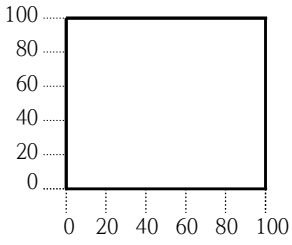
을 사용하되 ( )안의 숫자를 바꿀 수 있다.

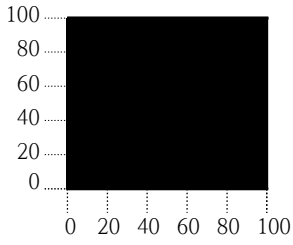
가장 적은 개수의 블록을 사용하여 정류장까지 도착하는 방법을 스크래치로 만들어보자.

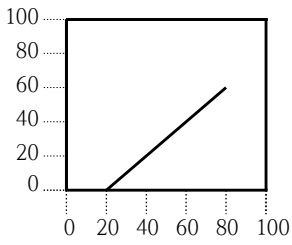
(노란 길 위에서만 버스는 다닐 수 있고, 한 칸씩 이동하지 않아도 된다.)

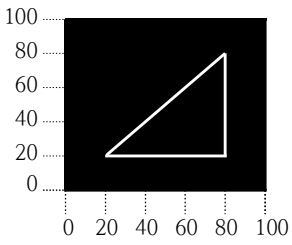
생각해보기 #1

수에 의한 디자인은 컴퓨터에서 그래픽을 만들어 내는 디자인 도구이다. 사람이 명령어들을 만들어 줌으로써 컴퓨터는 명령어에 의해서 그림들을 그려낸다. 문제에 답하기 전에 아래에 예로 제시한 명령어와 그림을 주의 깊게 살펴보자.

명령문	컴퓨터가 그린 그림
종이 0	

명령문	컴퓨터가 그린 그림
종이 100	

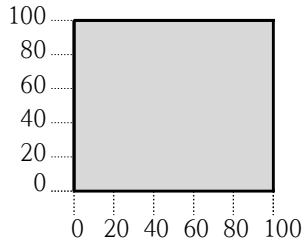
명령문	컴퓨터가 그린 그림
종이 0 연필 100 선 20 0 80 60	

명령문	컴퓨터가 그린 그림
종이 100 연필 0 선 20 20 80 20 선 20 20 80 80 선 80 20 80 80	

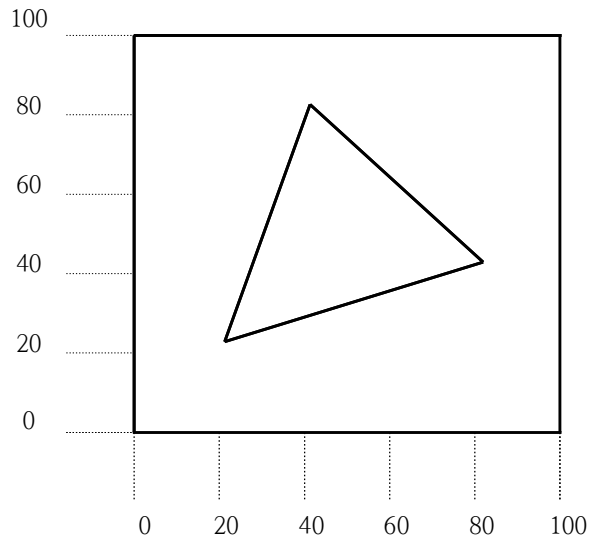
생각해 봅시다.

1. 아래에 제시된 그림을 그리기 위한 명령어는 다음 중 어느 것인가?

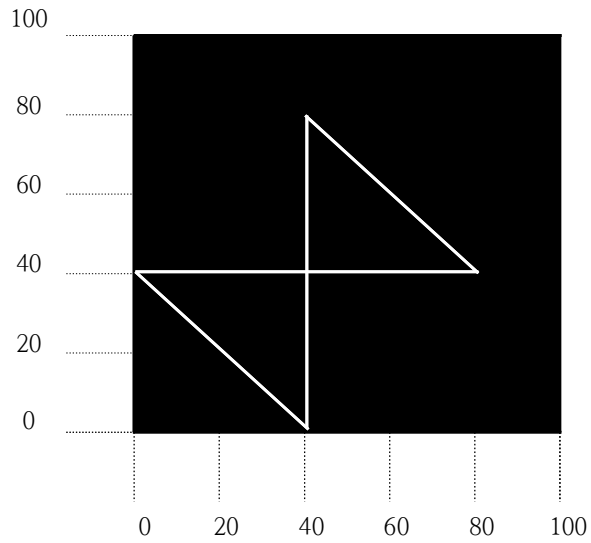
① 종이 0
② 선 50
③ 종이 50
④ 연필 50
⑤ 연필 100



2. 한 붓 그리기 방법으로 컴퓨터로 아래와 같은 삼각형을 그리려고 합니다. 빈칸에 명령문을 써보세요.

명령문	컴퓨터가 그린 그림
	

3. 한 붓 그리기 방법으로 컴퓨터로 아래와 같은 도형을 그리려고 합니다. **명령문을 가장 적게(짧게) 사용하여 빈칸에 써보세요.**

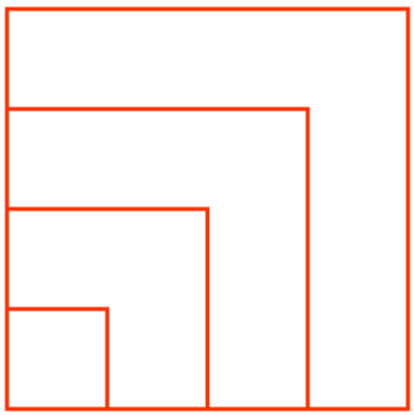
명령문	컴퓨터가 그린 그림
	

생각해보기 #2

성냥개비 16개로 가장 큰 정사각형을 만들려고 한다.
아래의 성냥개비 1개를 시작으로 그려보아라.

생각해보기 #3

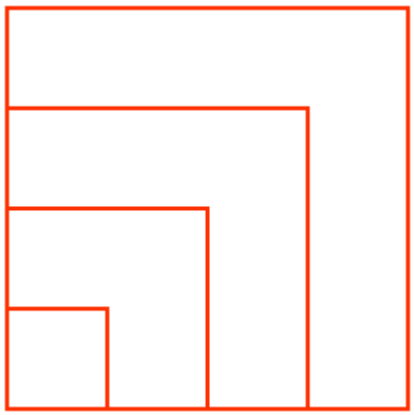
그렸던 부분을 다시 그릴 수 없는 한 붓 그리기 놀이를 하고 있다.
다음 도형을 한 붓 그리기 할 수 있는가?



가능하다 ( ), 가능하지 않다 ( )

생각해 봅시다.

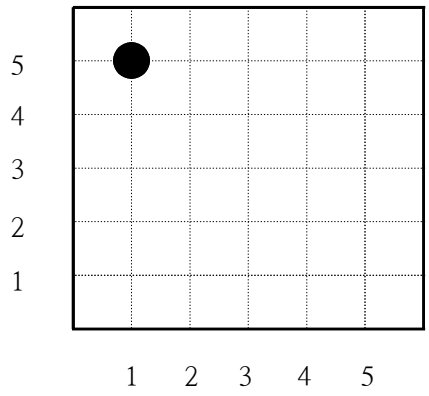
만약, 그렸던 부분을 다시 그릴 수 있다면 한 붓 그리기가 가능한가?



가능하다 ( ), 가능하지 않다 ( )

생각해보기 #4

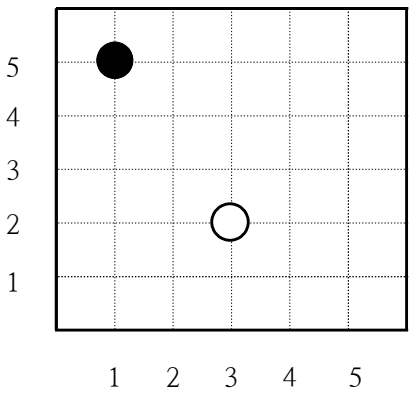
컴퓨터가 바둑을 둘 수 있게 만든 프로그램이 있다. 기본적인 명령어는 아래와 같다.

명령문	컴퓨터가 그린 그림
검 {1,5}	

생각해 봅시다.

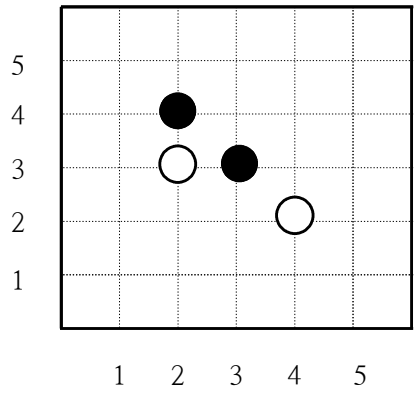
만약, 그렸던 부분을 다시 그릴 수 있다면 한 붓 그리기가 가능한가?

1. 아래에 제시된 바둑돌을 놓는 명령어는 다음 중 어느 것인가?

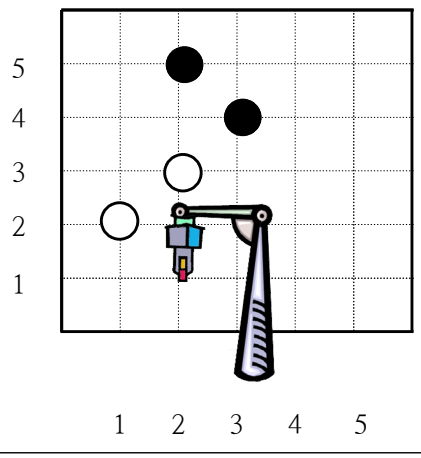


① 검 {1,5}
② 검 {1,5} 흰 {2,3}
③ 검 {1,5} 흰 {3,2}
④ 흰 {3,2}
⑤ 검 {1,5} 흰 {3,3}

2. 아래에 제시된 바둑돌을 놓는 명령어를 쓰시오.
(순서는 상관없습니다.)

명령문	컴퓨터가 그린 그림
	

3. 만약 우리가 만든 명령문에 따라 로봇팔이 바둑판 위를 공중으로 움직이며 바둑판에 바둑돌을 그려 넣는다고 상상해봅시다. **로봇팔이 움직이는 시간이 가장 짧게** 아래의 바둑돌을 그려 넣는 명령문을 작성해 봅시다. (로봇팔의 현재 위치는 {2,1}입니다.)

명령문	컴퓨터가 그린 그림
	

2. 펜으로 시작하기

1) 펜 팔레트 블록 구성 익히기

① 지우기 ② 펜 내리기 ③ 펜 올리기

④ 펜의 색 □로 정하기

⑤ 펜의 색 ( )만큼 바꾸기

- 펜의 색: 총 200가지 / 0 (빨간색), 70 (초록색), 130 (파랑), 170 (분홍색)
- 펜색의 좌우로 움직인다고 생각하기

⑥ 펜의 색 ( )로 정하기

⑦ 펜의 그림자 ( )만큼 바꾸기 // 명암 바꾸기

- 색의 어둡고 밝은 정도: 0~100 / 0 (가장 어두움), 50 (적당함), 100 (매우 밝음)
- 펜색의 위, 아래로 움직인다고 생각하기

⑧ 펜의 그림자 ( )로 정하기 ⑨ 펜의 크기 ( )만큼 바꾸기

⑩ 펜의 크기 ( )로 정하기

⑪ 스템프

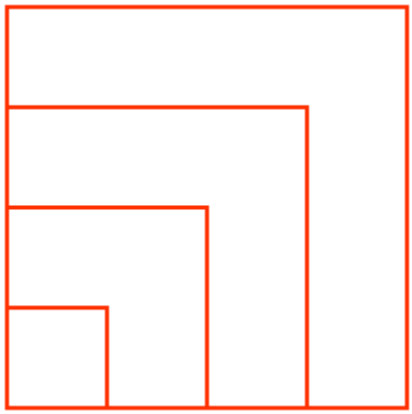
창의적으로 생각하기

펜 블록과 동작 블록을 사용하여 사각형을 그릴 수 있을까?
먼저 스크립트를 써보자.

한 붓 그리기 #1

생각해 봅시다.

펜 블록과 동작 블록만을 사용하여 다음의 도형을 그릴 수 있을까?



자신의 아이디어를 아래의 칸에 써보자.

(스크립트를 써도 좋습니다.)

한 붓 그리기 #2

창의적으로 생각해 봅시다.

펜 블록과 동작 블록을 이용하여 아래와 같은 도형을 그려봅시다.

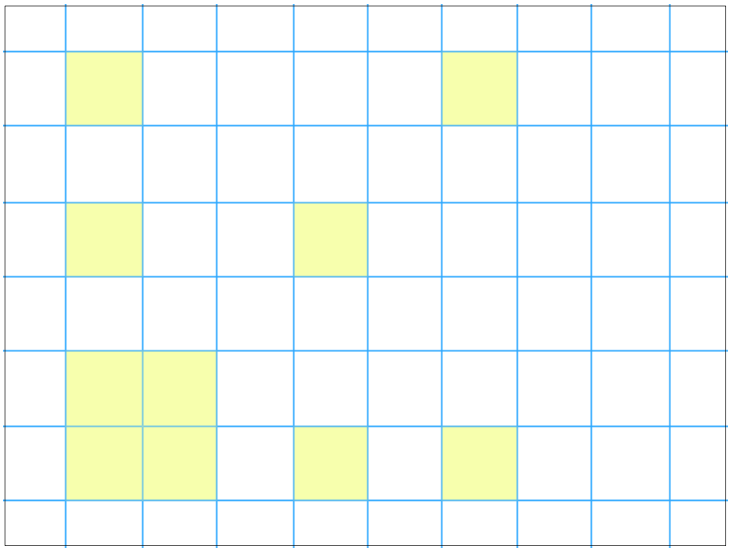
정답이 없으므로 다양한 방법을 생각해보세요.

자신의 아이디어를 아래의 칸에 써보자.

(스크립트를 써도 좋습니다.)

우유 배달하기

노란 부분은 우유 배달을 받는 집이다.



이 노란 부분에 모두 우유 배달을 하고자 한다.
아래의 그림처럼 우유 배달을 했다는 증거로 도장(●)을 찍고 오고자 한다. 자신의 아이디어를 아래의 칸에 써보자.

(스크립트를 써도 좋습니다.)

외계인의 미스테리 서클?

동작 블록과 펜 블록만 사용하고도 다양한 도형만들기가 가능하다.
아마 똑똑한 외계인은 이런 방법으로 지구에 미스테리 서클(Mystery Circle: 이상하고 신비한 원 모양)을 만들고 있지 모른다.

아래의 미스테리 서클을 만들 수 있을까?

자신의 아이디어를 아래의 칸에 써보자.

(스크립트를 써도 좋습니다.)

1. 반복

1) 제어블록 팔레트 구성 익히기

① ▶ 클릭되었을 때 ② [ ]키 눌렀을 때
③ 스프라이트 클릭되었을 때 ④ ( )초 기다리기
⑤ 무한 반복 ⑥ 반복 ( )회
⑦ [ ] 방송하기 / [ ] 방송하고 기다리기 / [ ] 받을 때
⑧ 반복 〈 〉 계속 확인
⑨ 만약 〈 〉라면 / 만약 〈 〉라면 / 아니면
⑩ 〈 〉까지 기다리기 ⑪ 반복 〈 〉까지
⑫ 스크립트 멈추기 / 모두 멈추기●

[2강_퀴즈1] Catch Me If You Can (날 잡아봐라!)

문제상황: 똑같은 공모양의 스프라이트가 5개 있다. 이 스프라이트들을 프로젝트가 실행되면 직선으로 이동하기 시작하고 벽에 부딪히면 튕기면서 계속 움직인다. 사용자가 마우스로 클릭하면 스프라이트는 움직임을 멈춘다.

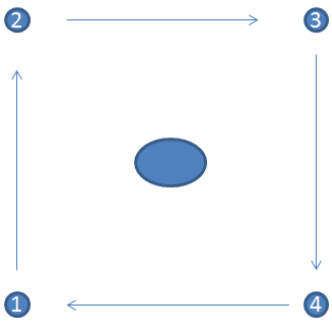
프로젝트를 만들고 테스트 해보시오.

[2강_퀴즈2] 이어 달리기

아래의 블록들을 잘 이용하되, 필요가 없다고 생각하면 사용하지 않아도 된다.

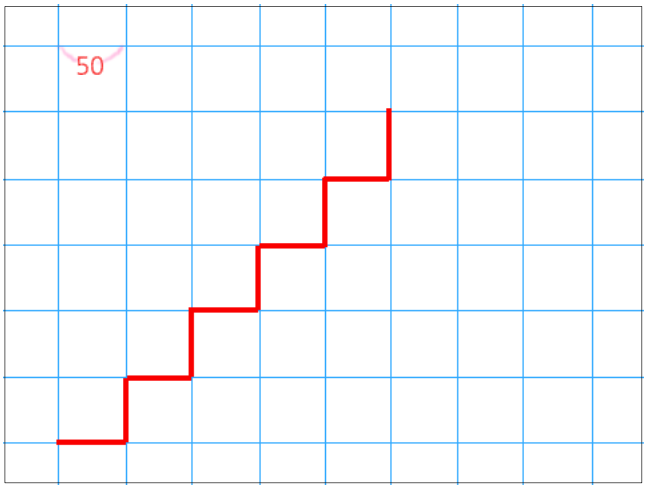
문제상황: 프로젝트를 실행(⚑)하면 좌표(100,100), 좌표(100,-100), 좌표(-100,-100), 좌표(-100,100)에 네 개의 스프라이트가 각자 위치하게 된다.

무대의 가운데 있는 '버튼' 스프라이트를 누르면 다음과 같이 움직이게 하라.



가운데 버튼을 누르면 1번 스프라이트가 2번을 향해 움직이기 시작하고, 2번에 닿으면 2번이 출발, 2번은 3번쪽으로, 3번은 4번쪽으로 움직이게 하고 마지막 4번이 원래 1번이 있던 자리로 오게 하는 프로그램을 만들어보자.

반복문의 위력?



50격자로 나뉘어진 바탕무대에서 스프라이트가 움직이며 그림을 그렸다.

문제를 해결해 봅시다.

위의 경로처럼 스프라이트가 움직이며 선을 그리는 프로그램을 만드시오. 단, 사용할 수 있는 블록은 펜 팔레트의 모든 블록과 아래의 3가지 제어 블록이다.

( )안의 숫자들은 바꿀 수 없다.

생각하고 생각하고 생각하라. 생각의 반복.

수학과 컴퓨팅(또는 프로그래밍)은 매우 밀접하게 관련되어 있다.

다음의 그림은 정사각형이다.

문제를 해결해 봅시다.

위의 정사각형을 그리는 프로그램을 만들고 프로그램을 게시판에 업로드한다. 블록 사용에 제한은 없으며 가장 적은 개수의 블록을 사용해라.

반복의 반복의 반복의 반복.

1번 명령: '안녕'을 두 번 하시오.

2번 명령: 1번 명령을 두 번 수행하시오.

3번 명령: (또 2번을 반복한다면?)

앞쪽의 문제에서 난이도를 조금 높여보자.

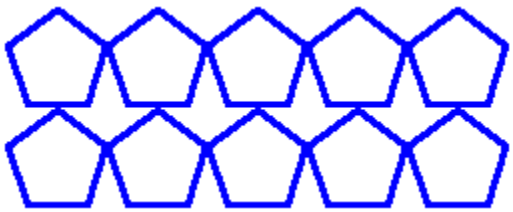
문제를 해결해 봅시다.

1. 정□각형의 내각의 합을 구하는 식이 180 * (□-2) 라고 할 때 아래의 정다각형들을 그릴 수 있을까?

(모든 평면도형의 외각의 합은 360도이다. 계산기를 이용해도 좋다.)

설명을 잠시 듣고 프로그램을 만들어보자.

2. 아래의 그림처럼 반복해서 같은 정다각형을 그리려면 어떻게 해야 할까?



3. 보지 않고서는 믿을 수 없다!

정팔각형, 정십각형, 정십이각형, 정이십각형이 존재할까요?

계산은 계산기로~!

2. 조건, 관찰과 형태

1) [프로젝트] 만약 ~에 닿기 라면 / [ ] 쪽 보기를 이용하기

① 화살표로 달리는 고양이 만들기

② 고양이를 쫓아다니는 박쥐떼 만들기

③ 타이머 달기

2) 마법봉을 쫓아 다니는 불빛

3) (스프라이트)의 ( ) / ( )까지 거리을 이용해보자.

① 왼쪽으로 움직일 때 숨기기 / 오른쪽으로 움직일 때는 보이기

② 두 스프라이트의 거리가 100일 때 숨기기 / 말하기

(컴퓨터는 우리보다 똑똑할까? 거리가 100인 것을 계산할 것인가?)

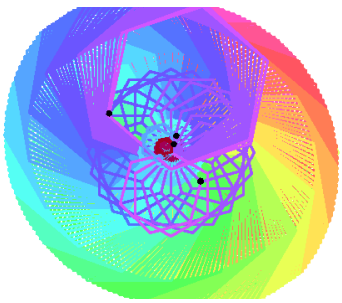
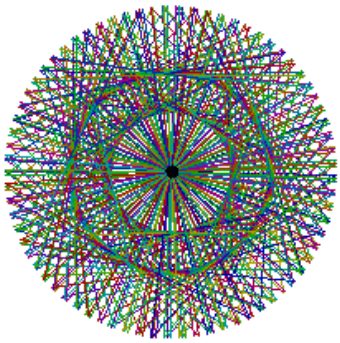
[2강_퀴즈3] 숫자를 알아맞혀라.

(변수, 난수의 개념은 아직 사용하지 않는다.)

① [ ]을 묻고 기다리기 / 대답 사용하기
② 1부터 10까지의 수 중 하나를 마음속으로 생각하자.
③ 다섯고개가 되게 하기 위해서는 어떻게 해야 할까?

[2강_퀴즈4] 기하학 미디어 아트

원 / 육각형 / 별을 만드는 방법을 익히고 반복문을 몇 개 중첩시킨다면 다음의 기하학적인 미디어 아트를 만들 수 있다. 도전해 보시오.

1. 변수

1) 변수란? 변수의 개념 알기

2) '대답' 변수란 무엇인가? [소스3-1]

① 비밀번호 물어보기 // 고양이가 당신의 이름을 말한다.
② '대답'이라는 변수 사용하기
③ 비밀번호가 맞을 때까지 계속 물어보기

3) '타이머'를 변수로 이용하기 [소스3-2]

① 타이머 초기화
② '시간' 변수 만들기
③ 시간에 타이머 저장하기 / 1초 기다리기

4) 계산기 만들기

① 계산 기능 구현하기 [소스3-3]
② '시간' 변수 만들기
③ 시간에 타이머 저장하기 / 1초 기다리기

[3강_퀴즈1] 계산기 디자인하기 [소스3-4]

[3강_퀴즈2] 변수 값 교환하기

[3강_퀴즈3] 3의 배수이면서 동시에 5의 배수인 수

문제상황

: 사용자에게 숫자를 입력받으면 3의 배수이면서 동시에 5의 배수인지를 알아보는 프로그램을 만들어보자.

예를 들어, 13을 입력 받으면 “3의 배수이면서 5의 배수가 아닙니다.” 출력 15를 입력 받으면 “3의 배수이면서 5의 배수입니다.”

[소스3-5]

출력

〈 〈(대답)나누기(3)의 나머지=0〉 그리고 〈(대답)나누기 5의 나머지=0〉 〉

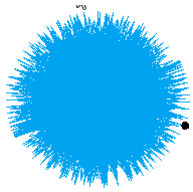
[3강_퀴즈4] 3의 배수이거나 또는 5의 배수인 수

5) 1부터 10까지 순서대로 말하는 로봇

[소스3-6]

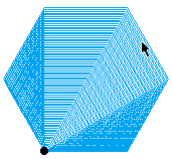
① ‘누적하기’의 의미 이해하기

[3강_퀴즈5] 변수를 이용하여 아래의 도형들을 만들어 보자.



변수를 이용하여 회전 / 움직이기
[소스 3-8]

변수를 이용한 정사각형 그리기
[소스 3-9]



변수를 이용한 정육각형 그리기
[소스3-10]

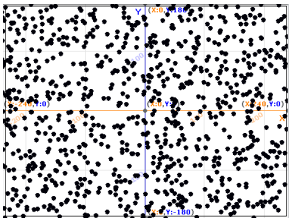
변수를 이용하여 별그리기
[소스 3-11]

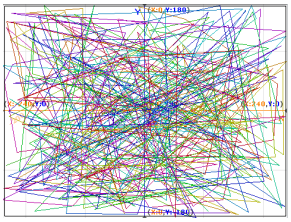
2. 난수

1) 난수 이해하기

① X좌표: -240 ~ 240 / Y좌표: -180 ~ 180

[소스3-11]





[3강_퀴즈6] 숫자 맞히기 열고개

[소스3-12]

문제상황

: 컴퓨터는 사용자가 모르게 1부터 100까지의 수 중에서 무작위로 하나를 고른다. 사용자는 그 수를 맞히는 것이 목표이다.

정답보다 내가 말한 수가 크면 UP이라고 출력하고, 작으면 DOWN, 정답이면 “정답입니다.”라고 출력한다.

기회는 10번이다. 10번 이내에 맞히지 못하면 종료한다.

인공지능 로봇의 시작

가위 바위 보 게임을 만들어 봅시다. 컴퓨터와 내가 가위바위보 게임을 하려면 어떤 점들이 생각해 봐야 할까요?

문제를 해결해 봅시다.

게임을 디자인해봅시다. 화면구성을 그리거나 아이디어를 쓰세요.

시계 만들기 [소스3-14]

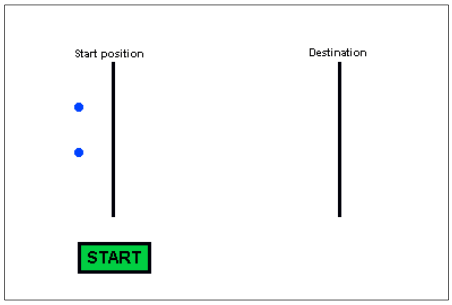
시계의 초침, 분침, 시침이 1초마다 회전하는 각을 계산할 수 있는가?

문제를 해결해 봅시다.

시뮬레이션을 디자인해봅시다. 화면구성을 그리거나 아이디어를 쓰세요.

3. 동시처리 때문에 일어나는 일들

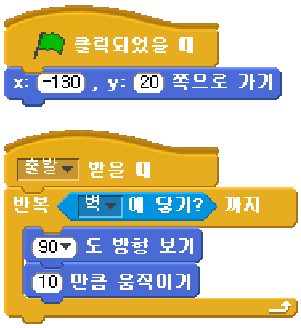
1) 다음의 두 개의 스프라이트 중에 목표지점까지 먼저 도착하는 것은? [소스3-15]



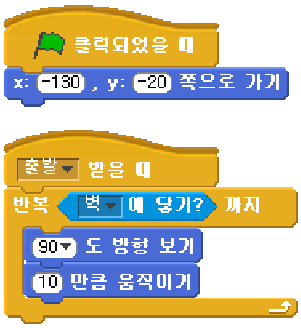
두 개의 스프라이트가 같은 거리에 있는 목적지까지 동시에 출발하려고 한다. 누가 우승일까?

우리가 프로그래밍 할 때에는 개념적으로 동시에 일어난다고 생각하지만 컴퓨터는 오로지 한 번에 한 가지 일만 한다. 이러한 차이로 인해 벌어지는 오류는 무수히 많아질 수 있다. 이러한 차이를 알게 됨으로써 내가 만든 프로그램의 버그(오류)를 좀 더 깊이 이해할 수 있다.

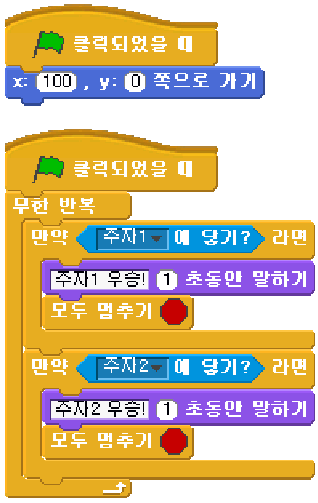
[주자1] 스프라이트



[주자2] 스프라이트



[도착지점] 스프라이트

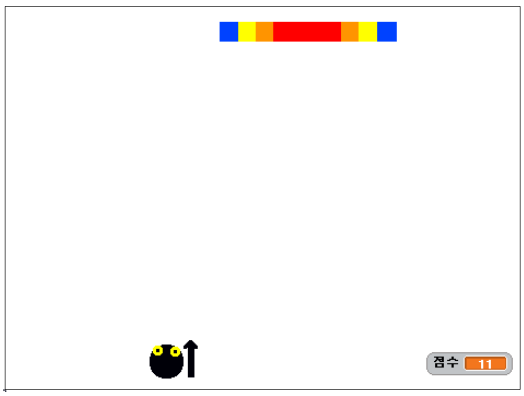


위와 같은 프로그래밍에서 생길 수 있는 오류는 무엇일지 생각해보자.

대안책 [소스3-16]

과녁 맞추기 게임

하나의 게임을 만드려고 한다. 스페이스바를 누르면 화살을 쏘게 되고, 과녁의 어느 부분에 맞느냐에 따라 점수가 계산된다.



문제를 해결해 봅시다.

게임을 디자인해봅시다. 화면구성을 그리거나 아이디어를 쓰세요.

1. 객체지향

1) 객체지향의 개념은?

2) 공유변수, 스프라이트(객체)변수 사용하기
- '속도' 변수를 공유하는 스프라이들 [소스4-1]
- '속도' 스프라이트 변수를 각자 가지고 있는 스프라이들 [소스4-2]

3) 인공지능의 개념 이해하기
- 프로그래밍으로 무엇까지 만들 수 있을까?

4) 시뮬레이션이란?

5) 함수와 재귀호출의 의미

[4강_퀴즈1] 계속 움직이며 딱 10까지만 말하는 고양이

땅을 파는 인공지능 지렁이

인공지능의 개념을 이해하고 난수를 이용하여 땅을 파고 들어가는 지렁이를 만들어봅시다.

문제를 해결해 봅시다.

지렁이가 실제 지렁이처럼 움직이게 하기 위해서는 어떤 것을 생각해야 할까요? 어떤 변수가 필요할지 생각해봅시다.

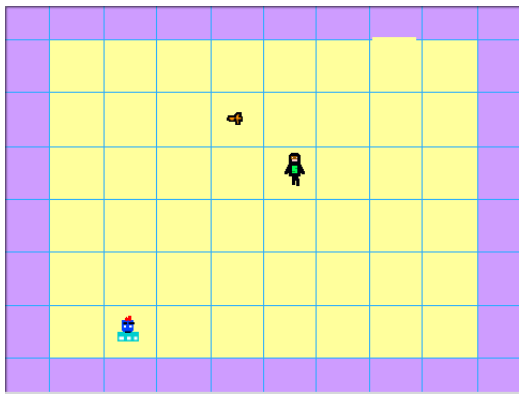
벌레의 움직임

더듬이를 이용해 장애물을 피해가는 벌레의 움직임을 프로그래밍 해봅시다.

문제를 해결해 봅시다.

벌레들이 장애물을 피해가기 위해서는 어떤 점을 고려해야 할까요?

청소로봇 #1



방에 사람, 로봇이 있다.

1. 사람과 로봇은 1초마다 무작위(위/아래/오른쪽/왼쪽)로 움직이지만 벽으로 이동할 수는 없다.
2. 사람은 50%의 확률로 현재 위치에 쓰레기를 버린다. (스탬프 기능 사용)
3. 로봇은 현재 위치에 쓰레기가 있다면 이를 줍는다. (스탬프 기능 사용)

문제를 해결해 봅시다.

50% 확률을 어떻게 표현할 것인가? 벽으로 이동시키지 않으려면?

청소로봇 #2

어느 방향에 쓰레기가 있는지를 확인하기 위해서 이번에는 레이저를 사용하여 보자.

아래와 같은 동작원리를 가진 로봇이 있다고 생각한다.

1. 로봇은 레이저를 가지고 있다. 레이저는 로봇을 중심으로 360도 회전하며 쓰레기를 찾는다.
2. 쓰레기를 찾은 로봇은 잠시 레이저 회전을 멈추고 쓰레기쪽으로 이동하여 청소한다.
3. 쓰레기가 없을 때는 무작위(난수/랜덤)로 움직인다.

생각해 봅시다.

레이저가 쓰레기를 찾았다면 로봇이 쓰레기쪽으로 이동할 수 있는 방법은 무엇인가?

박테리아와 백신 시뮬레이션

박테리아가 퍼지고 제거되는 시뮬레이션을 통해 어떠한 모양으로 퍼지는지 알아보자.

생각해 봅시다.

박테리아가 번지는 원리는 아래와 같으며 그대로 프로그래밍 하시오.

기본사항) 배경은 하얀색이다. (인간의 몸속 하얀 부분이라고 생각하자.)
박테리아는 검은색이며 점모양이다. (먼저처럼 울퉁불퉁 점모양)
백신은 파란색이이며 점모양이며 박테리아보다는 조금 작은 크기이다.

동작1. 박테리아는 좌표(0,0)으로 위치하고 무작위(난수) 방향으로 번지기 시작한다.

동작2. 하얀색(인간의 몸)에 닿을 때까지 1씩 움직인다.
움직이는 도중에 파란색(백신)에 닿으면 동작1을 다시 시작한다.

감기의 전염

같은 크기의 공간 A, B에 감기세균이 하나씩 있다.

A에는 무작위로 움직이는 사람 50명이 있고, B에는 무작위로 움직이는 사람 10명이 있다.

나는 과연 어떤 공간에 있는 것이 감기에 조금이라도 늦게 걸리겠는가?

생각해 봅시다.

사람은 검은색 점모양이며, 감기세균은 초록색 점모양입니다.

사람은 다음과 같이 행동합니다.

- 자신의 속도로 움직입니다. 벽에 닿으면 튕깁니다.
- 2~4초마다 자신의 속도가 무작위로 바뀝니다.
- 2~4초마다 자신의 방향이 무작위로 바뀝니다.
- 초록색(감기세균)에 걸리면 초록색(감기에 걸린 사람의 모양)으로 바뀌며, 다른 사람을 전염시킬 수 있습니다.

시뮬레이션으로 평가하기

횟 수	사람이 10일 때	사람이 30명일 때	사람이 50명일 때
1회			
2회			
3회			
4회			
5회			

1. 애플리케이션과 게임

1) 애플리케이션이란?

- 일반적으로 프로그램을 의미한다.
  컴퓨터의 소프트웨어 개발은 그 자체가 문제 해결을 위한 탐색이다.

ex) 이전에 우리는 '계산기'를 만들기도 했다.
이런 프로그램이 바로 애플리케이션이라고 할 수 있다.

2) 게임에서 자주 사용되는 개념에는 어떤 것들이 있나요?

3) 비디오 디지털 게임의 종류

- 롤플레잉(RPG), 시뮬레이션, 슈팅게임, 액션게임, 보드/카드 게임, 스포츠/레이싱 등

※ 앞으로의 문제들은 어떠한 예시도 없다.
스스로 아이디어를 구상하고 직접 프로그램으로 만들기 바란다.

애플리케이션 ex) 그림판

아이들이 컴퓨터를 가지고 그림을 쉽게 그릴 수 있게 “그림판”을 만들어 주려고 한다. 아이디어 구상은 아래의 그림과 같다.

아래의 아이디어 구상을 살피고 그림판 애플리케이션(응용프로그램)을 만들어보자.

① 붓은 마우스를 따라다님. 마치 마우스 포인터가 붓 모양인 것처럼.

② 물감을 마우스로 클릭하면 붓 색깔이 해당 물감색으로 바뀜.

③ 지우기 버튼을 누르면 하얀색 도화지가 다 지워짐.

④ 하얀색 도화지 위에서 마우스를 클릭하거나 클릭한 후 움직이면 선이 그려짐.

게임

다양한 게임 아이디어들이 있다. 아이디어를 읽고 자신의 아이디어를 더 보태어 자신만의 게임을 만들어보자.

1) 양치기 소년

양들이 계속 벽에 튕기며 좌우로 움직이고 있다.
마우스를 무대에 가져가서 키보드 1을 누르면 빨간색 점이 찍히고,
키보드 2를 누르면 파란색 점이 찍힌다.
양들은 움직이다가 빨간색에 닿으면 시계방향으로 회전 90도,
파란색에 닿으면 반시계방향으로 90도 회전한다.
최종 목적지에 닿으면 1단계 끝~

2) 잠수함 슈팅 게임

과녁 맞히기 게임과 비슷. 핵심 캐릭터는 배이고 물위에서 좌우로만 이동가능. 물 속에서 랜덤(난수)하게 잠수함들이 나와 위쪽으로 미사일발사. 이를 피해야 함.

3) 기억력 테스트 게임

짧은 시간 그림을 보여주고 그것이 무엇인지 알아맞혀야 함.
점수가 높아질수록 그림을 보여주는 시간이 짧아짐.

4) 한자리 수 암산 게임

1단계: 하나의 수를 보여줌
/ 3초 안에 이제까지 보여준 수들의 합을 입력해야 함.
2단계: 하나의 수 보여줌 -> 1초 후 하나의 수 또 보여줌
/ 3초 안에 이제까지의 수들의 합을 입력해야 함
3단계: 위와 같은 방법으로 계속 수들을 늘려감.

5) 점프게임

항상 점프하는 점(볼) 스프라이트가 있다. 장애물을 피하여 목적지에 도착하여 다음 단계로 넘어가는 롤플레잉 게임의 한 종류.

<부록 10> 계산적 사고력(계산적 인지력 영역) 검사지 A형

제외된 문항을 포함한 원본

계산적 사고력 검사지

(계산적 인지력 영역)

Group assessment of Computational thinking

A 형

제주대학교 대학원 컴퓨터교육과 http://cerg.jejunu.ac.kr

A형 1번

보드게임 말 옮기기

친구들과 “세계 일주”라는 보드게임을 하고 있다. 아래의 그림처럼 내 말은 ⊙ 모양이며 현재 ‘일본’ 위치에 있다. 여기에서 주사위를 굴려 3이상의 숫자가 나오면 ‘도착’ 지점으로 가는 것으로 규칙을 정했다.

출발	영국	프랑스	인도	러시아	중국	일본	캐나다	미국	도착
						⊙			

생각해 봅시다.

보드게임을 다시 시작하여 내 말이 아래의 그림처럼 현재 ‘출발’ 위치에 있다고 하자.

출발 ⊙	영국	프랑스	인도	러시아	중국	일본	캐나다	미국	도착

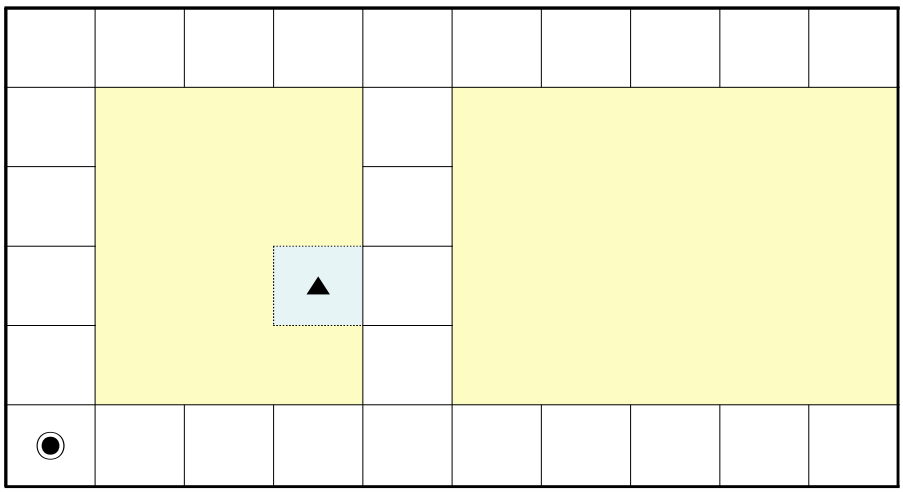
주사위를 두 번 굴려서 도착지점에 갔다고 했을 때, 첫 번째와 두 번째에 나올 수 있는 수들을 써보자.

	첫 번째 수	두 번째 수
예)	6	6

A형 2번

길 안내하기

아래의 그림은 민수의 동네 약도이다. (하늘에서 아래를 내려다 본 그림)



▲: 박물관, ◉: 현위치

길을 가던 관광객이 민수에게 박물관까지 가는 길을 물었다.
민수는 관광객에게 아래와 같이 6줄짜리 안내문을 써주었다.

순 서	동 작
①	현위치에서 길찾기 시작
②	동쪽 방향으로 보기
③	4블록(칸) 움직이기
④	북쪽 방향으로 보기
⑤	2블록(칸) 움직이기
⑥	왼쪽을 보면 목적지가 보임

(※ 안내문의 길이: 6)

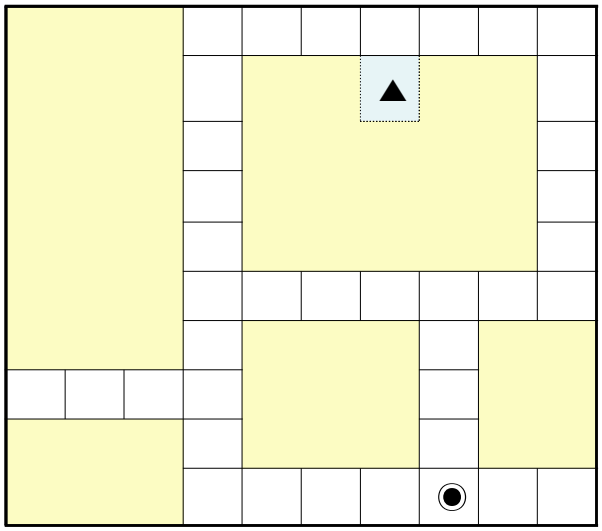
관광객은 목적지에 도착할 때까지 총 6번의 동작을 했고, 박물관을 잘 찾을 수 있었다.

A형 2번

생각해 봅시다.

아래의 그림은 우리 동네 약도이고 어떤 관광객이 나에게 미술관까지 가는 길을 물었다.

미술관까지 가는 길은 다양하지만 나는 관광객이 쉽게 길을 찾아갈 수 있도록 안내문의 길이를 줄이고 싶었다. 민수의 방법처럼 안내문을 작성하되, 현위치에서 미술관까지 가는 **길이(명령문의 줄)가 가장 짧은** 안내문을 완성하여라.

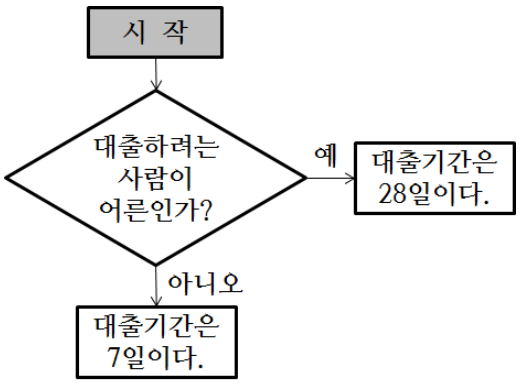


순 서	동 작
①	현위치에서 길찾기 시작

A형 3번 (제외된 문제)

대출기간 계산하기

제주시 도서관은 간단한 도서대출 시스템을 가지고 있다.
어른은 대출 기간이 28일이며 어린이는 대출 기간이 7일이다.
아래의 순서도는 이 대출기간을 계산하는 방법을 나타낸 것이다.



한편, 서귀포 도서관은 이와 비슷하지만 좀 더 복잡한 도서대출 시스템을 가지고 있다.

- '제한'으로 표시된 책은 누구나 대출기간이 2일이다.
- '제한'이 표시되지 않은 책(사전 제외)들의 경우에는, 어른이 28일, 어린이가 14일이다.
- '제한'이 표시되지 않은 '사전'일 경우에는 누구나 7일이다.
- 반납일이 지난 책을 가지고 있는 사람은 다른 책을 대출할 수 없다.

생각해 봅시다.

1. 민수는 어린이이며, 반납일이 지난 책이 없다. 만약 민수가 서귀포 도서관에서 '제한'이 표시되지 않은 책을 대출하고자 한다면 대출기간은 얼마인가? ( ) 일

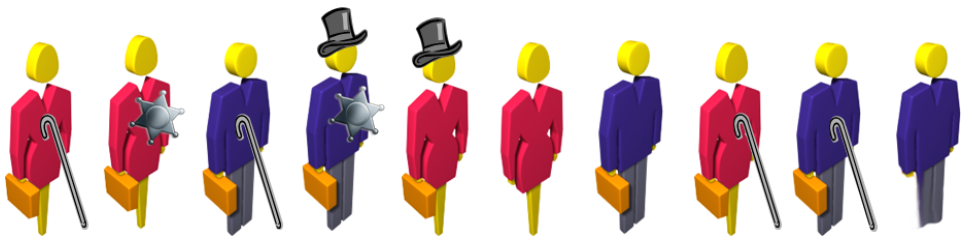
2. 지윤이는 어른이며, 반납일이 지난 책이 없다. 만약 지윤이가 서귀포 도서관에서 '제한'이 표시된 책을 대출하고자 한다면 대출기간은 얼마인가? ( ) 일

A형 4번

스파이를 찾아라.

첩보원 10명이 있다. 하지만 이 중에 스파이가 있다고 한다.
아래의 글은 스파이에 대한 정보이다. 잘 읽어보고 스파이를 찾아라.

- '가방'이 없는 첩보원은 스파이가 아니다.
- '가방'이 있고 '지팡이'도 있는 남자는 스파이가 아니다.
- '가방'이 있고 '모자'도 있는 여자는 스파이가 아니다.
- '모자'를 쓰지 않은 여자 중 '지팡이'가 있는 여자는 스파이가 아니다.
- '배지'를 가지고 있는 첩보원은 스파이가 아니다.
- 스파이가 아니라고 한 첩보원을 제외하여 남은 자는 스파이가 확실하다.



생각해 봅시다.

1. 스파이는 몇 명인지 모른다. 스파이가 확실한 첩보원의 번호를 쓰시오.

( )

A형 5번

출력을 반복하라.

컴퓨터에게 다음과 같은 명령을 주었다.

‘반복 2 { }’은 { } 안의 내용을 2번 반복해서 화면에 출력하겠다는 의미이다.

그 예시는 아래의 표와 같다.

명령문	화면 출력

반복 2 {
“안녕”
“그래”
}
	안녕
	그래
	안녕
	그래

생각해 봅시다.

1. 화면 출력이 아래와 같을 때 ‘반복’을 사용하여 명령문을 작성해 보세요.

명령문	화면 출력
	딸기 아이스크림 바나나 딸기 아이스크림 바나나 딸기 아이스크림 바나나 딸기 아이스크림 바나나

A형 5번

컴퓨터에게 다음과 같은 명령을 주었다.
반복 명령 안에 반복을 또 사용할 수 있는 예시를 보여주는 것이다.

명령문	화면 출력

반복 2 {
“안녕”
반복 2 {
“그래”
}
}
	안녕
	그래
	그래
	안녕
	그래
	그래

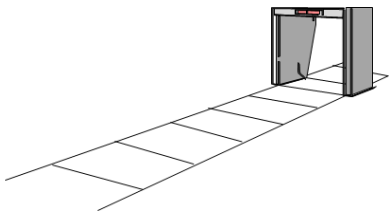
생각해 봅시다. (제외된 문항)

2. 화면 출력이 아래와 같고 반복을 활용하여 가장 짧은(명령줄의 길이가 짧은) 명령문을 만들어보자.

명령문	화면 출력
	역시 나는 천재 천재 천재 역시 나는 천재 천재 천재

A형 6번

검색로봇



검색대는 폭발물을 찾아내는 기능을 한다. 3초에 한번 검색대 바로 밑을 지나가는 롤러판 위의 물건을 레이저로 검사한다. 아래의 그림은 이를 단순화 한 것이다.

롤러판은 1초에 한 번씩 앞으로 1칸씩 이동되며 이 때 롤러판 위의 물건도 따라 움직이게 된다.

(롤러판은 이동하지만 검색대는 고정되어 있다.)

[롤러판] 이동방향 ⇒

물건 ①	물건 ②		물건 ③	물건 ④	물건 ⑤	물건 ⑥			현재 검사중	

검색대

생각해 봅시다.

1. 위의 그림처럼 검색대는 현재 검사중이다.
   롤러판에 있는 물건 ① ~ ⑥ 중에서
   검사가 안되고 지나쳐버리는 물건의 번호를 모두 쓰시오.

( )

A형 7번

동시에 일어나는 일들

아래의 그림처럼 두 사람이 활을 들고 화살을 쏘려고 하고 있다.

화살 A

화살 B

생각해 봅시다.

1. 두 사람이 동시에 과녁을 향해 화살을 쏘았습니다.
   과녁에 화살이 꽂히게 되는 순서는? ---------- ( )
   ① 화살 A가 먼저 꽂힌다. ② 화살 B가 먼저 꽂힌다.
   ③ 동시에 꽂힌다.

2. **(제외된 문항)**
   과녁이 아래와 같이 화살이 꽂히는 것을 확인한다고 한다.
   화살이 언제 날아올지는 모르며 과녁은 아래의 표처럼 계속적으로 검사를 하고 있다고 생각하자. 두 사람이 동시에 과녁을 향해 화살을 쏘았습니다. 과녁은 어느 화살이 먼저 꽂힌다고 판단할까?
   ---------- ( )

순 서	검사하는 내용
[1]	화살 A가 꽂혔는지 검사한다.
[2]	화살 B가 꽂혔는지 검사한다.
[3]	화살이 모두 꽂혀있지 않으면 [1]부터 순서대로 계속 검사.

① 항상 화살 A가 먼저 꽂힌다고 판단하게 될 것이다..
② 항상 화살 B가 먼저 꽂힌다고 판단하게 될 것이다.
③ 동시에 꽂힌다고 판단하게 될 것이다.
④ 상황에 따라 A 또는 B가 먼저 꽂힌다고 판단하게 될 것이다.

A형 8번

동시에 일어나는 일들

아래의 그림처럼 두 사람이 활을 들고 화살을 쏘려고 하고 있다.

화살 A

화살 B

생각해 봅시다.

1. 두 사람이 '시작'이라는 말을 동시에 듣고 아래와 같이 동시에 동작하려고 합니다. 과녁에 꽂히게 되는 화살의 색깔을 순서대로 써보시오.

순서	민호의 동작
①	'시작' 구호 듣기
②	기다리기
③	노란 화살 쏘기
④	빨간 화살 쏘기

순서	철수의 동작
①	'시작' 구호 듣기
②	파란 화살 쏘기
③	기다리기
④	기다리기

( ) → ( ) → ( )

A형 9번

변하는 물감 색깔

물감으로 색을 만드는 방법을 아래와 같이 정리해 보았다.

물감1 색깔	물감2 색깔	물감 1과 2를 섞은 색깔
빨강	노랑	주황
빨강	파랑	보라
빨강	녹색	검정
파랑	노랑	녹색
파랑	검정	남색
검정	흰색	회색
회색	검정	회색
회색	흰색	회색

(※ 물감1과 물감2의 색깔이 서로 바뀌어도 이들을 섞은 결과는 같다.)

생각해 봅시다.

1. 아래의 동작을 하고 나서 팔레트에 남게 되는 물감의 색깔은?

---------- ( )

순서	동작
①	깨끗한 팔레트 준비
②	팔레트에 파란색 물감 짜기
③	팔레트에 노란색 물감 짜서 섞기
④	팔레트에 빨간색 물감 짜서 섞기
⑤	팔레트에 흰색 물감 짜서 섞기

① 녹색　② 보라　③ 남색　④ 검정　⑤ 회색

2. 현재 팔레트에 빨간색 또는 파란색이 있다. 하지만 확실히 둘 중의 어느 것인지는 모른다. 이 팔레트에 노란색을 넣었다면 어떤 색깔의 물감이 만들어 질 수 있는지 모두 쓰시오.

( )

A형 10번

규칙에 따른 계산

형수는 작은 칠판을 가지고 있다. 칠판에 무엇인가를 쓸 때에는 현재 칠판의 모든 것을 지우고 써야 하는 규칙이 있다.

만약 아래와 같은 명령문이 있다면 형수는 다음과 같이 동작해야 한다.

명령문 (순서는 위에서부터 아래로)	실제 형수의 동작	동작 후 상황	
		칠판 A	칠판 B
칠판준비 A B	깨끗한 칠판 A와 B 준비하기		
칠판A ← 2	칠판A에 숫자 2 쓰기	2	
칠판A ← 칠판A + 1	현재 칠판A에 있는 수와 1을 더한 값을 칠판A를 지운 후에 쓰기	3	
칠판B ← 칠판A + 1	현재 칠판A에 있는 수와 1을 더한 값을 칠판B를 지운 후에 쓰기	3	4
칠판A ← 칠판A + 칠판B	현재 칠판A에 있는 수와 칠판B에 있는 수를 더한 값을 칠판A를 지운 후에 쓰기	7	4

생각해 봅시다.

1. **(제외된 문항)**

위의 규칙처럼 아래의 명령문을 실행한다면 칠판A에 남게 되는 수는?

칠판 A ( )

명령문
칠판준비 A
칠판A ← 6
칠판A ← 칠판A × 2
칠판A ← 칠판A ÷ 4

A형 10번

2. 위의 규칙처럼 아래의 명령문을 실행한다면 칠판A와 칠판 B에 남게 되는 수는?

칠판 A (　　), 칠판 B (　　)

명령문
칠판준비 A B
칠판A ← 3
칠판B ← 4
칠판A ← 칠판A × 칠판B
칠판B ← 칠판A ÷ 칠판B

3. 위의 규칙처럼 아래의 명령문을 실행한다면 칠판A, B, C에 남게 되는 수는?

칠판 A (　　), 칠판 B (　　), 칠판 C (　　)

명령문
칠판준비 A B C
칠판A ← 3
칠판B ← 4
칠판C ← 칠판A
칠판A ← 칠판B
칠판B ← 칠판C

A형 11번

뭐가 나올지 몰라

아래의 그림처럼 상자에 1부터 3까지의 번호가 써진 카드가 **무수히 많다.** 상자의 속을 볼 수는 없어서 상자에서 카드를 꺼낼 때 어떤 카드가 나올지는 아무도 모른다.

생각해 봅시다.

1. **(제외된 문항)** 이 상자에서 카드를 2장 꺼냈다. 두 카드의 수를 더하여 나올 수 있는 값들을 모두 쓰시오.

( )

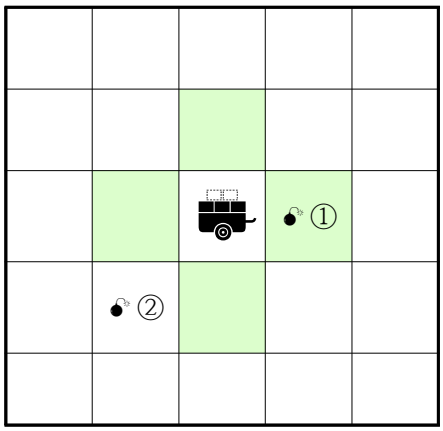
2. 이 상자에서 카드를 3장 꺼냈다. 세 카드의 수를 더하여 나올 수 있는 값들을 모두 쓰시오.

( )

A형 12번

인공지능

지뢰 탐사로봇()이 있다. 지뢰 탐사로봇은 현재의 위치에서 동/서/남/북쪽의 칸에 존재하는 지뢰만 탐색할 수 있다. 예를 들자면, 아래의 지도에서 지뢰①은 탐색할 수 있지만 지뢰②는 탐색이 불가능하다.



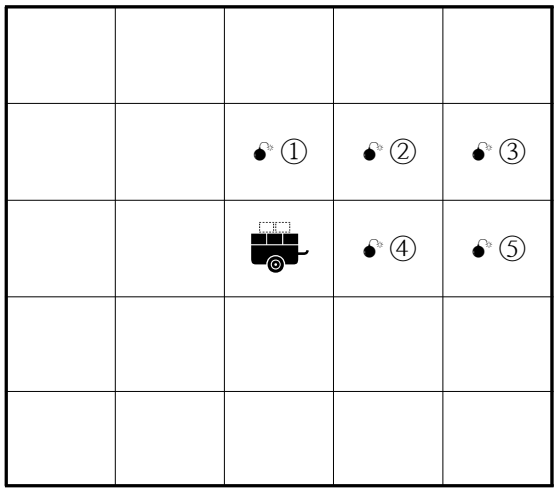
구체적으로 지뢰 탐사로봇은 아래와 같은 순서의 규칙으로 지뢰를 탐색하여 제거한다.

작동번호	작 동 내 용
①	현재 위치의 동쪽을 탐색한다. 지뢰가 있다면 동쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
②	현재 위치의 서쪽을 탐색한다. 지뢰가 있다면 서쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
③	현재 위치의 남쪽을 탐색한다. 지뢰가 있다면 남쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
④	현재 위치의 북쪽을 탐색한다. 지뢰가 있다면 북쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
⑤	동쪽, 서쪽, 남쪽, 북쪽 중 무작위(난수)로 한 군데 이동한다.

A형 12번

생각해 봅시다.

1. 아래의 지도에 있는 지뢰 탐사로봇()이 위의 규칙대로 작동하다면 지뢰가 제거되는 순서를 지뢰 번호로 쓰시오.



( ) → ( ) → ( ) → ( ) → ( )

A형 13번

객체

참새 10마리가 하늘을 날고 있다. 참새는 각자 생명을 가지고 있다. 모두 비슷한 생김새라고 생각하겠지만 조금씩 다른 성질을 가지고 있다. 예를 들자면, 참새마다 날아가는 속도는 다를 것이다. 만약 첫 번째 참새의 속도가 10이라면

참새1.속도 ← 10

으로 표현하고자 한다.

1. 하늘을 자유롭게 날고 있는 참새들이 있다고 할 때, 위 예의 '속도'처럼 **각각의 참새들이 서로 다르게 가지고 있는 성질**과 가장 **상관없는 것**은? -------- ( )

① 현재 높이　② 날고 있는 방향　③ 몸의 크기
④ 참새의 총 개수　⑤ 현재 위치

2. 교통수단 3개가 움직이고 있다. 이들의 종류를 다음과 같이 표현하였다.

교통수단1.종류 ← 기차
교통수단2.종류 ← 오토바이
교통수단3.종류 ← 택시

교통수단의 '종류'라는 성질처럼 위의 교통수단이 각각 서로 다르게 가질 수 있는 성질을 생각나는 대로 쓰시오. (3개 이상)

A형 14번

함수 호출

자판기가 하나 있다. 아래의 표는 자판기에서 구입할 수 있는 음료의 가격이다.

자판기 음료수	가 격
사이다	300원
콜라	400원
녹차	500원
쥬스	700원

자판기에 500원을 넣고 사이다를 구입하고 남은 돈을 '자판기(500원,사이다)' 라고 표현한다. 즉 자판기(500원,사이다)는 200원과 같다.

생각해 봅시다.

1. 아래의 표현이 나타내는 값은? -------- (원)

자판기(1000원,쥬스) + 자판기(500원,녹차) + 자판기(500원,콜라) + 자판기(1000원, 사이다)

2. 위와 같은 표현 방법으로 이번달 전기, 수도, 인터넷, 전화요금을 낸 후에 남은 돈을 요금(낸 돈, 요금종류)의 표현방법으로 계산하고자 한다.

요금종류	가 격
전기	4500원
수도	8500원
인터넷	4000원
전화	9000원

아래의 표현이 나타내는 값은? -------- (원)

요금(5000원,전기) + 요금(10000원,수도) + 요금(5000원,인터넷) + 요금(10000원,전화)

A형 15번

재귀

자판기(1000원,콜라)는 자판기에 1000원을 넣고 콜라를 구입했을 때 남은 돈을 의미한다고 했다. 여기 새로운 자판기가 있다. 이 자판기에서 뽑을 수 있는 음료의 종류와 가격은 아래의 표와 같다.

자판기 음료수	가 격
커 피	600원
홍 차	500원
율무차	400원
코코아	300원

새로운 표현을 하나 더 배워보자.

자판기(자판기(1000원,커피) , 코코아)

뜻: 자판기에 1000원을 넣고 커피를 구입하고 남은 돈을 다시 자판기에 넣고 코코아를 구입하여 남은 돈

즉 **자판기(자판기(1000원,커피) , 코코아)** 의 값은 100원이다.

생각해 봅시다.

1. 아래의 표현이 나타내는 값은? -------- (원)

자판기 (자판기(자판기(자판기(2000원,커피) , 홍차) , 율무차) , 율무차)

2. 아래의 상황을 위의 표현식처럼 바꾸어 써보시오.

자판기에 3000원을 넣고 율무차를 구입하여 남은 돈을
자판기에 넣고 홍차를 구입하여 남은 돈을
자판기에 넣고 코코아를 구입하여 남은 돈

<부록 11> 계산적 사고력(계산적 인지력 영역) 검사지 B형

제외된 문항을 포함한 원본

계산적 사고력 검사지

(계산적 인지력 영역)

Group assessment of Computational thinking

B 형

제주대학교 대학원 컴퓨터교육과 http://cerg.jejunu.ac.kr

B형 1번

보드게임 말 옮기기

친구들과 “세계 일주”라는 보드게임을 하고 있다. 아래의 그림처럼 내 말은 ⊙ 모양이며 현재 ‘일본’ 위치에 있다. 여기에서 주사위를 굴려 3이상의 숫자가 나오면 ‘도착’ 지점으로 가는 것으로 규칙을 정했다.

출발	영국	프랑스	인도	러시아	중국	일본	캐나다	미국	도착
						⊙			

생각해 봅시다.

내 말이 아래의 그림처럼 현재 ‘영국’ 위치에 있다고 하자.

출발	영국	프랑스	인도	러시아	중국	일본	캐나다	미국	도착
	⊙								

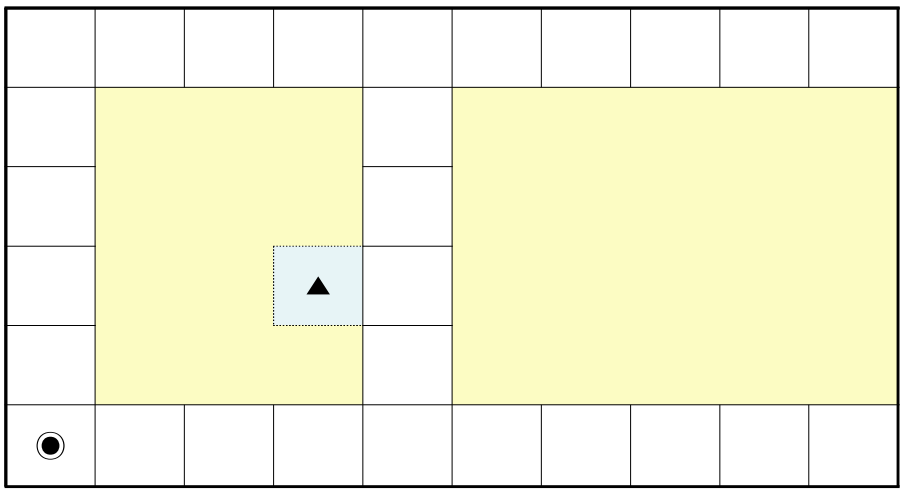
주사위를 두 번 굴려서 도착지점에 갔다고 했을 때, 첫 번째와 두 번째에 나올 수 있는 수들을 써보자.

	첫 번째 수	두 번째 수
예)	6	6

B형 2번

길 안내하기

아래의 그림은 민수의 동네 약도이다. (하늘에서 아래를 내려다 본 그림)



▲: 박물관, ◉: 현위치

길을 가던 관광객이 민수에게 박물관까지 가는 길을 물었다.
민수는 관광객에게 아래와 같이 6줄짜리 안내문을 써주었다.

순 서	동 작
①	현위치에서 길찾기 시작
②	동쪽 방향으로 보기
③	4블록(칸) 움직이기
④	북쪽 방향으로 보기
⑤	2블록(칸) 움직이기
⑥	왼쪽을 보면 목적지가 보임

(※ 안내문의 길이: 6)

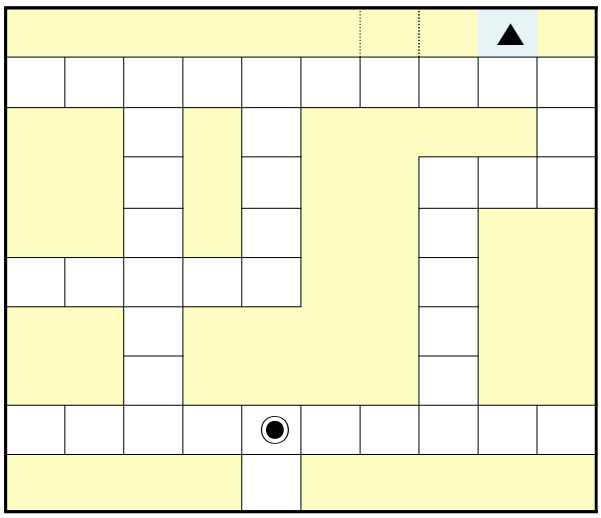
관광객은 목적지에 도착할 때까지 총 6번의 동작을 했고, 박물관을 잘 찾을 수 있었다.

A형 2번

생각해 봅시다.

아래의 그림은 우리 동네 약도이고 어떤 관광객이 나에게 미술관까지 가는 길을 물었다.

미술관까지 가는 길은 다양하지만 나는 관광객이 쉽게 길을 찾아갈 수 있도록 안내문의 길이를 줄이고 싶었다. 민수의 방법처럼 안내문을 작성하되, 현위치에서 미술관까지 가는 **길이(명령문의 줄)가 가장 짧은** 안내문을 완성하여라.



순 서	동 작
①	현위치에서 길찾기 시작

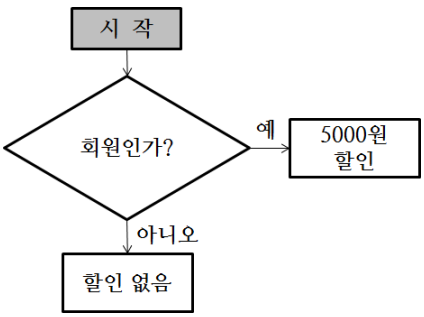
B형 3번 (제외된 문제)

대출기간 계산하기

우리 동네 '깎아요 미용실'에서는 커트, 파마, 염색 중 1개의 서비스만 받을 수 있다. 고객이 남성과, 여성인지에 따라, 그리고 머리길이(짧고 긴 정도)에 따라 요금이 다르다.

- '커트'는 머리 길이에 상관없이 남자의 경우에는 1만원, 여자의 경우에는 2만원이다.
- '파마'는 남자의 경우에는 머리 길이에 상관없이 3만원이다.
- '파마'는 여자의 경우에 머리가 길면 8만원이고, 짧은 경우에는 5만원이다.
- '염색'의 경우에는 남녀 구분 없이 머리가 길면 10만원, 짧은 경우에는 7만원이다.

이 미용실에서는 내야할 총 금액에서 회원인 경우 할인을 아래의 순서도처럼 해주고 있다.



생각해 봅시다.

1. 여학생인 서연이는 긴 머리를 가지고 있으며 이 미용실의 회원이다. 만약 '파마'를 할 때 내야할 금액은 얼마인가?

( )원

2. 남학생인 동림이는 짧은 머리를 가지고 있으며 이 미용실의 비회원이다. 만약 '커트'를 할 때 내야할 금액은 얼마인가?

( )원

B형 4번

범인을 찾아라.

용의자가 10명이 있다. 이 중에 범인이 있으며 몇 명인지는 모른다. 아래의 글은 범인에 대한 정보이다. 잘 읽어보고 범인을 찾아라.

- '동전'과 '지갑' 모두를 가지고 있는 용의자는 범인이 아니다.
- '안경'을 가지고 있는 용의자는 범인이 아니다.
- '구슬'을 가지고 있는 용의자는 범인이 아니다.
- '구두'와 '양말'을 가지고 있는 용의자는 범인이 아니다.
- 소지품이 1개밖에 없는 용의자는 범인이 아니다.
- 범인이 아니라고 한 용의자들을 제외하여 남은 자들은 범인이 확실하다.

아래의 표는 용의자들이 가지고 있던 소지품들이다.

용의자 ①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
구슬 지갑 구두	안경 구슬	동전 안경 구슬 지갑	구두 양말	동전	안경 구두	구슬 구두	동전 지갑	양말 지갑 구두	지갑 양말

생각해 봅시다.

1. 범인은 몇 명인지 모른다. 범인이 확실한 용의자의 번호를 쓰시오.

( )

B형 5번

출력을 반복하라.

청소 로봇에게 다음과 같은 명령을 주었다.

'**계속** 3 { }'은 { } 안의 내용을 3번 계속해서 동작하겠다는 의미이다. 그 예시는 아래의 표와 같다.

명령문	동작순서

계속 3 {
앞으로 1m 움직이기
바닥 쓸기
}
	앞으로 1m 움직이기
	바닥 쓸기
	앞으로 1m 움직이기
	바닥 쓸기
	앞으로 1m 움직이기
	바닥 쓸기

생각해 봅시다.

동작 순서가 아래와 같을 때 '**계속**'을 사용하여 명령문을 작성해보세요.

명령문	동장순서
	앞으로 1m 움직이기 바닥 쓸기 바닥 닦기 앞으로 1m 움직이기 바닥 쓸기 바닥 닦기 앞으로 1m 움직이기 바닥 쓸기 바닥 닦기

B형 5번

청소로봇에게 다음과 같은 명령을 주었다.
'**계속**' 명령 안에 '**계속**'을 또 사용할 수 있는 예시를 보여주는 것이다.

명령문	동작 순서

계속 2 {
  앞으로 1m 움직이기
  계속 3 {
  휴지 줍기
  }
}
	앞으로 1m 움직이기
	휴지 줍기
	휴지 줍기
	휴지 줍기
	앞으로 1m 움직이기
	휴지 줍기
	휴지 줍기
	휴지 줍기

생각해 봅시다. (제외된 문항)

2. 동작 순서가 아래와 같고 '계속'을 활용하여 가장 짧은 (명령줄의 길이가 짧은) 명령문을 만들어보자.

명령문	화면 출력
	앞으로 1m 움직이기 휴지 줍기 바닥 쓸기 바닥 닦기 휴지 줍기 바닥 쓸기 바닥 닦기 앞으로 1m 움직이기 휴지 줍기 바닥 쓸기 바닥 닦기 휴지 줍기 바닥 쓸기 바닥 닦기

B형 6번

UFO 헌터

형민이는 UFO의 존재를 믿으며 UFO를 찍기 위해 세계를 누비고 있다. 그가 가진 유일한 장비는 카메라이다. 하지만 카메라는 하늘 전체를 찍을 수 없으며 정해진 너비만 찍을 수 있었다. 아래 그림의 하나의 칸은 이 카메라로 찍을 수 있는 너비를 표시해 둔 것이다.

UFO 이동경로⇒

1 UFO	2 UFO		3 UFO	4 UFO	5 UFO	6 UFO		현재 촬영중		

현재 UFO 위치

사진이 찍히는 너비

6개의 UFO가 하늘에 있는데 위의 그림처럼 오른쪽으로 모두 이동중이며 1초에 1칸 (카메라 촬영 너비의 1칸) 이동한다. 형민이는 위의 그림처럼 카메라를 한군데 고정시켜 놓고 자동으로 2초에 한번씩 촬영되도록 설정해 두었다.

생각해 봅시다.

1. 위의 그림처럼 카메라는 현재 촬영중이다. UFO ① ~ ⑥ 중에서 카메라에 찍히게 되는 UFO의 번호를 모두 쓰시오.

( )

B형 7번

동시에 일어나는 일들

아래의 그림처럼 두 사람이 두 사람이 총을 쏘려고 하고 있다.

총알 A

총알 B

생각해 봅시다.

1. 현실 세계에서 두 사람이 동시에 같은 총, 같은 총알로 과녁을 향해 총을 쏘았습니다. 과녁에 총알이 도착하는 순서는? -- ( )

① 총알 A가 먼저 꽂힌다. ② 총알 B가 먼저 꽂힌다.
③ 동시에 꽂힌다.

2. **(제외된 문항)**

과녁은 아래의 표처럼 총알이 꽂히는 것을 확인하는 컴퓨터 프로그램으로 만들어졌다고 한다. 총알은 언제 날아올지는 모르며 과녁은 아래의 표처럼 계속적으로 검사를 하고 있다고 생각하자. 두 사람이 동시에 과녁을 향해 총을 쐈습니다.

과녁은 어느 총알이 먼저 도착한다고 판단할까? --- ( )

순서	검사하는 내용
[1]	총알 A가 도착했는지 검사한다.
[2]	총알 B가 도착했는지 검사한다.
[3]	총알이 모두 도착하지 않았으면 [1]부터 순서대로 다시 계속 검사.

① 항상 총알 A가 먼저 도착한다고 판단하게 될 것이다..
② 항상 총알 B가 먼저 도착한다고 판단하게 될 것이다.
③ 동시에 도착한다고 판단하게 될 것이다.
④ 상황에 따라 총알 A 또는 총알 B가 먼저 도착한다고 판단하게 될 것이다.

B형 8번

동시에 일어나는 일들

아래의 그림처럼 두 사람이 두 사람이 총을 쏘려고 하고 있다.

총알 A

총알 B

생각해 봅시다.

1. 두 사람이 '시작'이라는 말을 동시에 듣고 아래와 같이 동시에 동작하려고 합니다. 과녁에 꽂히게 되는 총알의 색깔을 순서대로 써보시오.

순서	민호의 동작
①	'시작' 구호 듣기
②	노란색 총알 쏘기
③	기다리기
④	보라색 총알 쏘기
⑤	기다리기

순서	철수의 동작
①	'시작' 구호 듣기
②	기다리기
③	기다리기
④	기다리기
⑤	빨간색 총알 쏘기

( ) → ( ) → ( )

B형 9번

두 개의 기호

기호 두 개를 합하여 하나의 기호를 만드는 암호문이 있다.

기호1	기호2	기호 1과 2로 만든 기호
○	○	◎
▲	▲	▢
▢	▢	■
◎	●	◉
△	△	▲
▲	●	▲
■	△	○
■	▢	◉

(※ 기호1과 기호2의 순서가 서로 바뀌어도 결과는 같다.)

생각해 봅시다.

1. 아래의 동작을 하고 나서 만들어지는 기호는? ---- ( )

순서	동작
1	기호 ▢ 가져오기
2	기호 ▢ 섞기
3	기호 △ 섞기
4	기호 ○ 섞기
5	기호 ●섞기

① ▢ ② ▲ ③ ○ ④ ● ⑤ ◉

2. A와 B를 섞어 기호 C를 만들었다고 하자. 만약 기호 B가 '▢'라면 C가 될 수 있는 기호를 모두 쓰시오.

( )

B형 10번

모래판

민수는 모래판을 가지고 있다. 모래판에 무엇인가를 쓸 때에는 이미 모래판에 쓰여진 것들을 모두 지우고 써야한다. 만약 아래와 같은 명령문이 있다면 민수는 모래판을 가지고 다음과 같이 동작해야 한다.

명령문 (순서는 위에서부터 아래로)	실제 민수의 동작	동작 후 상황	
		모래판 A	모래판 B
모래판준비 A B	깨끗한 모래판 A와 B 준비하기		
모래판A ← 2	모래판A에 숫자 2 쓰기	2	
모래판A ← 모래판A + 1	현재 모래판A에 있는 수와 1을 더한 값을 모래판A를 지운 후에 쓰기	3	
모래판B ← 모래판A + 1	현재 모래판A에 있는 수와 1을 더한 값을 모래판B를 지운 후에 쓰기	3	4
모래판A ← 모래판A +모래판B	현재 칠판A에 있는 수와 칠판B에 있는 수를 더한 값을 칠판A를 지운 후에 쓰기	7	4

생각해 봅시다.

1. **(제외된 문항)**

위의 규칙처럼 아래의 명령문을 실행한다면 모래판A에 남게 되는 수는?

칠판 A ( )

명령문
모래판준비 A
모래판A ← 5
모래판A ← 모래판A × 4
모래판A ← 모래판A ÷ 10

B형 10번

2. 위의 규칙처럼 아래의 명령문을 실행한다면 모래판A와 모래판 B에 남게 되는 수는?

모래판 A (　　), 모래판 B (　　)

명령문
모래판준비 A B
모래판A ← 4
모래판B ← 3
모래판A ← 모래판A × 모래판B
모래판B ← 모래판A ÷ 모래판B

3. 위의 규칙처럼 아래의 명령문을 실행한다면 모래판A, B, C에 남게 되는 수는?

모래판 A (　　), 모래판 B (　　), 모래판 C (　　)

명령문
모래판준비 A B C
모래판A ← 3
모래판B ← 4
모래판C ← 모래판A
모래판A ← 모래판B
모래판B ← 모래판C

B형 11번

뭐가 나올지 몰라

아래의 그림처럼 1부터 6가지의 숫자가 적힌 주사위가 있다. 주사위를 돌려서 나오는 숫자는 아무도 예측할 수 없다.

생각해 봅시다.

1. **(제외된 문항)** 이 주사위를 두 차례 굴려서 나온 수를 더한 값을 공책에 썼다. 이런 행동을 무한히 반복할 때 공책에는 어떤 값들이 써져 있을지 모두 쓰시오.

( )

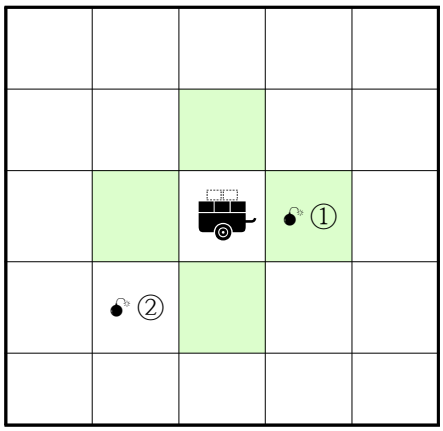
2. 1부터 3까지만 적혀져 있는 주사위를 사용하여 세 차례 굴렸다. 세 번 굴려 나온 수들을 더한 값을 공책에 썼다. 이런 행동을 무한히 반복할 때 공책에는 어떤 값들이 써져 있을지 모두 쓰시오.

( )

B형 12번

인공지능

지뢰 탐사로봇()이 있다. 지뢰 탐사로봇은 현재의 위치에서 동/서/남/북쪽의 칸에 존재하는 지뢰만 탐색할 수 있다. 예를 들자면, 아래의 지도에서 지뢰①은 탐색할 수 있지만 지뢰②는 탐색이 불가능하다.



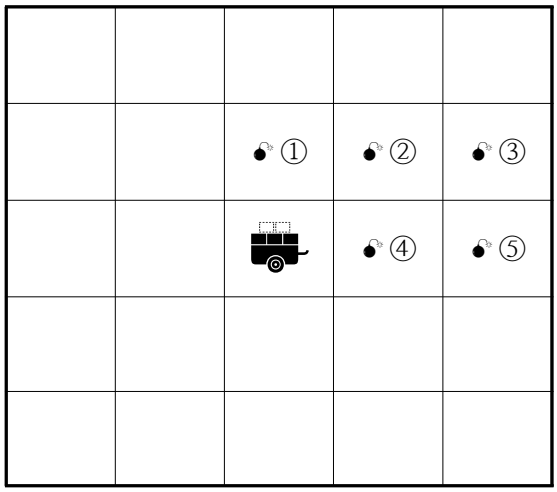
구체적으로 지뢰 탐사로봇은 아래와 같은 순서의 규칙으로 지뢰를 탐색하여 제거한다. (※ 이전 문제의 규칙과 순서가 다르니 주의 깊게 보기 바람.)

작동번호	작 동 내 용
①	현재 위치의 북쪽을 탐색한다. 지뢰가 있다면 북쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
②	현재 위치의 동쪽을 탐색한다. 지뢰가 있다면 동쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
③	현재 위치의 남쪽을 탐색한다. 지뢰가 있다면 남쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
④	현재 위치의 서쪽을 탐색한다. 지뢰가 있다면 서쪽으로 이동하여 지뢰를 제거하고 [①번 작동]부터 다시 시작한다. 지뢰가 없다면 다음 작동(↓)으로 넘어간다.
⑤	동쪽, 서쪽, 남쪽, 북쪽 중 무작위(난수)로 한 군데 이동한다.

B형 12번

생각해 봅시다.

1. 아래의 지도에 있는 지뢰 탐사로봇()이 위의 규칙대로 작동하다면 지뢰가 제거되는 순서를 지뢰 번호로 쓰시오.



( ) → ( ) → ( ) → ( ) → ( )

B형 13번

객체

붕어 10마리가 연못에서 헤엄치고 있다. 붕어는 각자 생명을 가지고 있다. 모두 비슷한 생김새라고 생각하겠지만 조금씩 다른 성질을 가지고 있다. 붕어마다 헤엄치는 속도도 다를 것이다. 만약 첫 번째 붕어의 속도가 10이라면

붕어1.속도 ← 10

으로 표현하고자 한다.

1. 연못을 자유롭게 헤엄 치고 있는 붕어들이 있다고 할 때, 위 예의 '속도'처럼 각각의 붕어들이 서로 다르게 가지고 있는 성질과 가장 상관없는 것은? ---------- ( )

① 현재 깊이　② 헤엄치는 방향　③ 몸의 크기
④ 연못 물의 양　⑤ 연못에서의 현재 위치

2. 축구 선수 3명이 축구경기를 하고 있다. 이들의 이름을 다음과 같이 표현하였다.

축구선수1.키 ← 176
축구선수2.키 ← 183
축구선수3.키 ← 165

축구선수의 '키'라는 성질처럼 축구선수 각각마다 서로 다르게 가질 수 있는 성질을 생각나는 대로 쓰시오. (3개 이상)

B형 14번

함수 호출

가전제품을 사려 한다. 아래의 표는 가전제품들의 가격이다.

제 품 명	가 격
냉장고	100만원
세탁기	50만원
전자렌지	30만원
밥솥	20만원

100만원을 지불하여 세탁기를 구입하고 남은 돈을
'구입(100만원,전자렌지)' 라고 표현한다
즉, **구입(100만원,전자렌지)** 는 70만원과 같다.

생각해 봅시다.

1. 아래의 표현이 나타내는 값은? -------- (원)

```
구입(100만원,냉장고) + 구입(50만원,세탁기) + 구입(50만원,전자렌지) + 구입(50만원,밥솥)
```

2. 위와 같은 표현 방법으로 도서를 구입하고 **구입(낸 돈, 도서종류)**의 표현방법으로 계산하고자 한다.

요금종류	가 격
그림책	5000원
잡지	9500원
소설	4500원
사전	9000원

아래의 표현이 나타내는 값은? ------ (원)

```
구입(5000원,소설) + 구입(10000원,사전) + 구입(5000원,그림책) + 요금(10000원,잡지)
```

B형 15번

재귀

구입(100만원,전자렌지)는 100만원을 지불하여 전자렌지를 구입했을 때, 남은 돈을 의미한다고 했다. 새로운 물품들을 구입하려고 한다.

자판기 음료수	가 격
TV	80만원
컴퓨터	100만원
오디오	50만원
전화기	20만원

새로운 표현을 하나 더 배워보자.

구입(구입(100만원,오디오) , 전화기)

뜻: 100만원을 지불하여 오디오를 구입하고 남은 돈을 다시 지불하여 전화기를 구입하여 남은 돈

즉 **구입(구입(100만원,오디오) , 전화기)** 의 값은 30만원이다.

생각해 봅시다.

1. 아래의 표현이 나타내는 값은? -------- (원)

구입 (구입(구입(구입(400만원,컴퓨터) , TV) , 오디오) , 컴퓨터)

2. 아래의 상황을 위의 표현식처럼 바꾸어 써보시오.

500만원을 지불하여 오디오를 구입하여 남은 돈을
다시 지불하여 컴퓨터를 구입하여 남은 돈을
다시 지불하여 전화기를 구입하여 남은 돈