프로그래밍이 어려운 이유와 컴퓨팅사고력간의 관계성 연구

오경선† · 안성진††

요 약

지금은 국내외적으로 소프트웨어 중심 사회로 컴퓨팅사고력기반의 문제해결능력을 갖춘 창의적 인재 양성을 하기 위해 많은 나라들이 힘을 쏟고 있다. 더욱이 직업생활과 밀접한 관련이 있는 대학에서는 소프트교육에 대한 중요성을 인식하며 IT계열이 아닌 전공자들을 대상으로 한 프로그래밍 교육을 진행하고 있으며 이는 더욱 확산되는 추세이다. 따라서 대학에서 프로그래밍을 처음 접하는 학습자들이 초보 학습단계에서 보이는 어려움의 정도를 사고단계별로 설문하여 이를 통해 얻어진 결과를 토대로 컴퓨팅 사고영역과의 관련성을 탐색하는 연구를 실시함으로써 모든 학생들이 배우는 프로그래밍 교육설계에 도움을 주고자 한다.

주제어 : 프로그래밍 교육, Computational Thinking, SW교육

A study on the relationship between difficulty in learning to program and Computational Thinking

Kyungsun Oh† · Seongjin Ahn††

ABSTRACT

Today, living in a software-centered global community. lots of countries are putting most efforts on breeding a creative manpower well equipped with problem-solving abilities based on computational thinking. Upon this colleges and universities where most strongly related to job activities are proceeding instruction related to programming for the IT majors realizing the importance of the software education. Hereby we intend to be of a proper help for every colleges and universities students who are working on educational programming design for the first time performing a research probing the relationship with the full gamut of computational thinking based on the result acquired through a phased survey showing the amount of difficulties at their beginning stages.

Keywords : Programming Education, Computational Thinking, SoftWare Education

†정 회 원: 성균관대학교 교과교육학과 컴퓨터교육전공 박사수료
††종신회원: 성균관대학교 교과교육학과 교수(교신저자)
논문접수: 2015년 8월 6일, 심사완료: 2015년 9월 9일, 게재확정: 2015년 9월 22일

1. 서론

IT와 밀접한 생활환경을 제공하는 첨단사회에서는 터치한번 말 한마디로 모든 것을 조작할 수 있는 세상의 구현이 가능하게 되었다. 이러한 시대를 살아가는 사람들에게 요구되는 능력은 단순히 컴퓨터를 잘 하는 것이 아니라 컴퓨팅사고력(Computational thinking)를 토대로 문제를 해결할 수 있는 힘이라 할 수 있다.[1] 이것은 사람들이 자신의 영역에서 컴퓨터 과학적 사고를 통하여 새로운 가치를 창출 할 수 있도록 한다. 이러한 능력은 개인에게는 더 나은 삶을 살아갈 수 있도록 하고 더 나아가 국가의 경쟁력을 강활 시킬 수 있는 원동력이 된다. 이러한 이유로 각 나라에서는 SW교육의 중요성을 인식하여 영국의 경우는 초중등교육과정에서 SW교육의 필수과목으로 지정하고 있으며 프랑스의 경우 2015년 9월부터 SW를 정규 과목으로 지정할 예정이며 미국의 경우는 스팀(STEM)분야의 인재를 잡기 위한 정책을 내놓고 있고 중국의 경우 해외고급인력을 끌어들이기 위한 범국가적 프로젝트를 내놓고 있는 실정이다.[2] 국내에서는 교육부와 미래부의 SW중심사회를 위한 인재양성 추진 계획을 통하여 초등학교에서 대학까지의 SW교육의 청사진을 내놓았다.[3] 이러한 SW교육의 중심에는 프로그래밍 교육이 있음을 확인할 수 있고 이는 컴퓨터 과학적 원리를 통한 문제해결능력의 사고 능력을 함양할 수 있는 목적을 지니고 있다. 더욱이 국내의 많은 대학에서는 비전공자들에게도 SW교육을 필수적으로 이수할 수 있도록 할 예정이라 하여 프로그래밍 교육은 더욱 강화될 전망이다.[4][5]

그러나 처음 프로그래밍 수업을 접하고 한 한기를 마치는 시점에서 대다수의 대학생들은 프로그래밍 교육은 ‘어렵고 지루하고 힘들다’ 라는 인식이 많이 있으며 IT계열전공자 조차 프로그래밍과 관련된 수업은 피하고 싶은 과목으로 인식되고 있다.[6] 프로그래밍 교육에서 있어서 대학생들에게 프로그래밍은 왜 어려운 것일까? 프로그래밍을 본격적으로 학습하기 전에 필요한 영역을 배운다면 조금 더 쉽게 학습할 수 있지 않을까? 라는 질문으로부터 본 연구를 시작하게 되었다. 이를 알아보기 위해 어려워하는 항목 중 지식영역이 아닌 사고영역은 무엇이 있는지 알아내기 위하여 전공에 상관없이 처음 프로그래밍을 접하는 대학생들을 대상으로 조사를 하여 항목들을 추출한다. 이러한 항목들의 세부 사고 요소를 찾아내기 위해 컴퓨팅사고력(Computational thinking)의 세부 요소와의 관계성을 찾아내기 위해 전문가들을 대상으로 설문 조사를 하여 그 결과를 분석한 후 관련요소들을 제시한다.

2. 이론적 배경

2.1 사전프로그래밍교육

프로그래밍 언어를 어려워하는 이유에 대한 연구들이 있었고 더 나아가 이를 근거로 효과적인 프로그래밍 학습을 위한 다양한 방법들이 제안되었다.

이 중에서 특히 프로그래밍 언어를 본격적으로 학습하기에 앞서 사전교육의 필요성을 인식하고 이를 위한 연구가 시도되었다.

장선희는 프로그래밍 언어를 배제한 사전 프로그래밍 교육의 필요성을 인식하고 퍼즐교육에서 그 대안을 찾았다.[7] Judith(1995)의 경우는 코딩을 제외한 교수 학습연구로 다중차트를 활용하여 알고리즘의 개발 및 서례도입을 하도록 하였으며[8] Rob(2005)의 연구에서는 통합적인 문제를 해결하기 위해 알고리즘 개발로 의사코드나 다이어그램을 도입할 필요성을 인식하였다.[9] 또한 Parsons 와 Haden(2006)의 연구에서는 사전 프로그래밍 교육의 필요성을 인식하고 이를 위한 퍼즐 활용을 제시하였다.[10]

이상의 연구들은 프로그래밍 언어를 교육하기 전에 사전 교육에 필요성을 인식하는데 부터 출발하였다. 또한 대부분 코딩을 작성하기에 앞서 문제해결에 필요한 알고리즘을 작성 방법에 초점을 두고 있음을 알 수 있다. 그러나 이러한 연구들은 알고리즘을 선행하되 어떤 학습도구를 사용할 것인지에 대한 이야기를 하여 그 효과성을 입증하였을 뿐 문제를 해결하기 위한 전반적인 사고과정에 대한 연구는 진행되지 않았다. 따라서 문제를 이해하고 분석하여 절차화한 후 표현하는

일련의 과정에서 필요한 세부 사고 능력에 대한 연구의 필요성이 필요하다 하겠다.

2.2 컴퓨팅 사고력(Computational Thinking)

프로그래밍 교육을 처음 접하는 학생들에게 필요한 것은 선행연구 분석에서 보았듯이 '문제 해결하는 사고과정' 이라 볼 수 있다. 이러한 문제해결하는 사고의 과정을 컴퓨팅 사고력(Computational Thinking)이라하며 이는 단순히 문제를 발견하고 분석하는 것에 끝나는 것이 아니라 컴퓨터 과학적 원리를 토대로 해결 할 수 있는 사고의 과정을 의미한다. 즉 컴퓨팅 사고력(Computational Thinking)는 문제를 해결하는 과정이며 만약 좋은 해결책이 존재하지 않는다면 왜 그런지에 대한 설명까지 할 수 있어야한다.[11]

2.2.1 정의

2009년National Research Council의 주도로 진행된 워크숍에서는 컴퓨팅 사고력(Computational thinking)에 대한 여러 학자들이 모여 다양한 의견을 나누었다. 이 워크숍에서 논의된 다양한 컴퓨팅사고력의 정의를 살펴보면 다음과 같다.[12]

① 컴퓨터 과학에 기초한 문제해결의 방법이다
② 문제해결의 논리적 구체적 계산 형태의 명확한 기호적 표현방법이다
③ 문제해결의 효율적 실행을 위한 절차적사고의 과정이며 인간의 지적능력을 확장하는 추론의 과정이다
④ 문제해결을 위해 하드웨어 소프트웨어를 인지적 도구로 사용 한다
⑤ 계산사고는 기술과 사고과정의 한정된 집합이 아니라 기술과 학습의 역동적 성격을 반영하는 개방적이고 성장하는 개념이다
⑥ 계산사고의 핵심적 개념은 명확한 절차의 반복적용 탐색 패턴매칭 반복적 정교화 무작위 기법 모델링 시뮬레이션 시각화 등이다

또한 ISTE와 CSTA 컴퓨팅사고능력을 K-12 교육에 적용하기 위하여 내린 조작적 정의는 다음과 같다.[13]

- 컴퓨터나 다른 도구를 사용하여 문제를 해결할 수 있도록 문제를 공식화하는 것
- 자료를 논리적으로 조직하고 분석하는 것
- 모델이나 시뮬레이션처럼 추상화를 통해 자료를 표현하는 것
- 알고리즘적인 사고를 통해 해결과정을 자동화하는 것
- 가장 효율적이고 효과적인 자료와 단계를 통해 가능한 해결책을 정의하고 분석하고 적용하는 것
- 이러한 문제해결 과정을 일반화하고 다양한 문제 상황으로 전이시키는 것

결국, 컴퓨팅사고능력은 문제를 해결하는 단순 과정이나 프로그래밍이 아니라, 논리화되고 절차화된 사고와 방법론을 통해 컴퓨터과학의 원리와 개념을 바탕으로 문제를 해결하는 인지적/정의적 사고과정이라고 할 수 있다.[14]

CETA와 ISTE에서는 David Barr, John Harrison, & Leslie Conery(2011)의 연구결과를 토대로 정보과학적 사고의 핵심 9가지 개념인 자료수집(Data Collection), 자료 분석(Data Analysis), 자료표현(Data Representation), 문제분해(Problem Decomposition), 추상화(Abstraction), 알고리즘과 절차(Algorithms & Procedures), 자동화(Automation), 시뮬레이션(Simulation), 병렬화(Parallelization)가 있다.[13]

3. 연구방법

이번 연구의 목적은 프로그래밍을 처음 접하는 학생들이 어려워하는 사고와 컴퓨팅사고능력간의 관계성을 분석하여 이를 토대로 효과적인 SW교육이 될 수 있는 방향성을 제시하고자 한다.

3.1 연구대상

본 연구에서 학생 대상은 수도권 지역의 대학에서 프로그래밍과목을 처음 수강하고 있는 1,2학년 학생들 122명이다. 학과에 상관없이 무작위로 설문하였으며 연구대상의 일반적 특성은 다음과 같다.

<표 1> 연구대상 특성

변인			빈도	비고
현재 수강중인 프로그래밍 언어	텍스트 기반언어	JAVA	48(39.3%)	※복수 응답 가능
		c++	2(1.6%)	
	스크립트기반 언어	HTML5	23(18.9%)	
		Javascript	37(30.3%)	

또한 설문지에서 도출된 학생들이 어려워하는 사고들과 컴퓨팅사고능력간의 관계를 분석하기 위해 컴퓨터교육관련 전문가들 16명의 설문을 받았으며 전문가들의 특성은 다음과 같다.

<표 2> 전문가특성

전문가		빈도	비고
컴퓨터교육 관련 교수자	대학	6(25%)	교육 10년 이상
	중등 교육 기관	4(38%)	
컴퓨터관련 업무 실무자	박사 과정 수료	3(19%)	
	박사	1(6%)	
컴퓨터관련 연구자	박사 과정	2(13%)	

3.2 측정도구

본 연구는 처음으로 프로그래밍 언어를 배우는 대학생들이 어려워하는 영역의 사고를 도출한 후 이와 관련된 컴퓨팅사고능력과의 관계를 분석하고자 한다. 이에 처음 프로그래밍 언어를 접하는 대학생들이 어려워하는지 사고가 무엇인지 알아보기 위해 국내외 관련 연구 자료와 전문가 의견을 통해 문항 개발을 하였다. 이러한 문항들 중 프로그래밍을 작성하기 전의 사고과정을 총12개의 예비 문항으로 측정하고자 하였다. 측정도구의 척도는 5점 척도 (매우 낮음, 낮음, 보통, 높음, 매우 높음)으로 구성하였다.

이렇게 구성된 설문조사지는 아래 표와 같다.

<표 3> 설문지 구성

구분	내용	문 항수	비고
사고과정	문제의 이해, 문제의 분석, 추상화, 알고리즘, 자동화, 시뮬레이션, 병렬화	12문항	Likert 5점 척도

또한 학생설문 응답을 토대로 컴퓨팅사고능력과의 관계성을 묻는 전문가들의 설문지에 대한 응답에서는 복수로 체크할 수 있도록 하였다.

<표 4> 설문문항내용

단계	문항	내용
문제이해	1	문제독해
	2	제시된 자료 분석
	3	필요한 자료 탐색
	4	조건탐색
문제분석	5	데이터 분석과 데이터 표현
	6	문제분해
	7	분해된 문제 통합
	8	추상화
	12	시뮬레이션
	13	병렬화
알고리즘적사고단계	9	간단한 해결절차 표현
	10	알고리즘 표현
	11	코딩

3.3 연구절차

본 연구에서는 수집된 자료를 분석하기 위하여 다음과 같은 연구방법을 진행하였다.

첫째, 처음으로 프로그래밍 강의를 수강하는 대학생들의 통계적 특성을 조사하기 위하여 빈도분석을 실시하였다.

둘째, 측정하고자하는 변수들을 유사항목으로 묶기 위하여 요인분석을 하였고 그룹 화된 문항들의 신뢰도를 측정하기 위하여 신뢰도분석을 하였다.

셋째, 유사항목으로 묶인 요인에 속한 항목 중 설문을 통하여 학생들의 어려워하는 사고의 항목들을 탐색하였다.

넷째, 전문가들을 대상으로 학생들이 어려워하는 항목과 관련 있는 컴퓨팅사고능력 요소를 찾기 위한 설문을 실시하였으며 빈도분석을 실시하

여 결과를 도출하였다.

이상의 분석을 위하여 한글 PASW 18.0을 사용하였다.

4. 자료 분석

4.1 요인분석

본 연구에서는 유사항목들을 분석하기 위하여 요인분석을 하였고 측정된 변수들은 크론바흐 알파계수를 이용해 문항 내적일 치도를 증명하였다. 또한 모형의 유의성을 검증하기 위하여 KMO와 Bartlett검정을 실시하였다. 본 연구의 표본은 KMO값이 0.896이고 Bartlett값이 유의 수준 0.000으로 유의한 것으로 나타나 요인분석을 하는 것이 적절하다는 것을 검증하였다.

<표 5> KMO와 Bartlett의 검정

표준형성 적절성의 Kaiser-Meyer-Olkin 측도.		.896
Bartlett의 구형성 검정	근사 카이제곱	876.729
	자유도	78
	유의확률	.000

정보의 손실을 최소화하면서 요약하고자 주성분분석을 사용하였으며 요인 적재치는 베리맥스 방법을 선택하여 '단 수화' 하였다. 요인 적재치는 각 변수와 요인간의 상관관계의 정도를 나타내므로 각 변수는 이 수치가 가장 높은 요인에 속하게 된다. 또한 고유 값이 1.0이상이고 요인적재치가 0.4이상을 기준으로 하였다.(일반적으로 사회과학분야에서의 요인과 문항의 선택기준은 고유값이 1.0이상 요인의 적재치는 0.4이상이면 유의한 변수로 간주한다.) [15]

<표 6> 측정도구 요인분석 결과

측정변수	성분			크론바하알파 알파값
	1	2	3	
13번	.789			.882
8번	.743			
6번	.736			
7번	.719			
12번	.653			
5번	.598			
2번		.861		.896
3번		.854		
4번		.788		
1번		.773		
9번			.864	.881
10번			.841	
11번			.777	
고유값	7.015	1.435	1.109	
KMO				.896
Bartlett구형성 검증			Chi-Squre	876.729
			df(p)	78(.000)

이 설문 문항에 대한 요인분석 결과를 요인별로 정리해 보면 다음과 같다.

<표 7> 설문문항 요인별 구성

구분	문항	명칭	크론바하알파 알파값
요인1	5번,6번,7번, 8번,12번,13번	문제분석단계	.882
요인2	1번,2번, 3번,4번	문제이해 단계	.896
요인3	9번,10번,11번	알고리즘적사 고단계	.881

5. 연구결과

지적영역 부분은 사고영역이라기 보다는 지식에 관련된 것으로 이는 사고과정에서 도움을 줄 수 있는 영역이지만 사고과정 자체가 아니므로 본 연구에서는 사고과정영역만을 분석해 보도록 하겠다.

5.1 단계별 사고과정의 어려움 정도

프로그래밍을 처음 접하는 대학생들이 프로그래밍언어가 어렵다고 느끼는 이유를 사고과정의 단계로 나누어 각 요인의 항목들을 분석하였다.

측정도구에서 사용한 리커트 5점 척도에서 응답항목 가운데 중간 항이 '들어 있느냐 아니냐도' 응답에 영향을 미칠 수 있다. 많은 사람들에게 응답 시 양극단을 피하고 중간 쪽으로 기우는 응답경향이 있기 때문에 중간 응답 항이 제시되어 있을 때에는 그것이 택해질 가능성이 그 만큼 더 높아질 수 있기 때문에 응답자의 '보통' 의 의미는 보통이상과 보통이하로 범주화하여 해석하여 분

석하였다.[16]

<표 8> 단계별 어려움의 정도 (단위:%)

요인	항목	어려움의 정도				
		매우 낮음	낮음	보통	높음	매우 높음
문제 분석 단계	5	1.8	12.3	39.5	35.1	11.4
	6		18.1	39.7	32.8	9.5
	7		11.4	43.0	38.6	7.0
	8	.9	15.8	42.1	32.5	8.8
	12	.9	14.9	40.4	37.7	6.1
	13	.9	11.2	48.3	31.9	7.8
문제 이해 단계	1		19.5	42.4	31.4	6.8
	2		21.2	42.4	30.5	5.9
	3	1.7	18.6	42.4	31.4	5.9
	4	.9	25.9	37.1	30.2	6.0
알고리즘적 사고 단계	9	.9	16.4	47.4	26.7	8.6
	10	.9	11.4	43.9	34.2	9.6
	11	2.6	9.5	44.0	37.9	6.0

전체적으로 볼 때 프로그래밍을 처음 접하는 학생들의 경우 모든 단계(문제이해단계, 문제분석단계, 알고리즘적 사고단계)에서 어려움(보통이상)을 느끼고 있다는 응답을 보였다. 즉 프로그래밍 언어를 사용하여 코딩을 작성하기 전까지의 모든 단계에서 어려움을 느끼는 것으로 해석 할 수 있다.

5.2 단계별 사고과정과 컴퓨팅 사고력 간의 관계

그렇다면 코딩하기 전까지의 단계는 컴퓨팅사고능력과 어떠한 관계를 가지고 있는 것일까? 이에 컴퓨팅 사고력의 9가지 요소를 살펴보고 관계를 분석하고자 한다.

학생들이 모든 단계에서 어려움을 겪고 있으며 이러한 문항들과 컴퓨팅사고능력과의 관계성과 관련된 질문의 응답률(%)을 보면 <표 9>와 같다. 여기서 각 문항과 컴퓨팅사고력과의 관련성 질문에 대한 응답은 사고가 복수로 할 수 있도록 하였다.

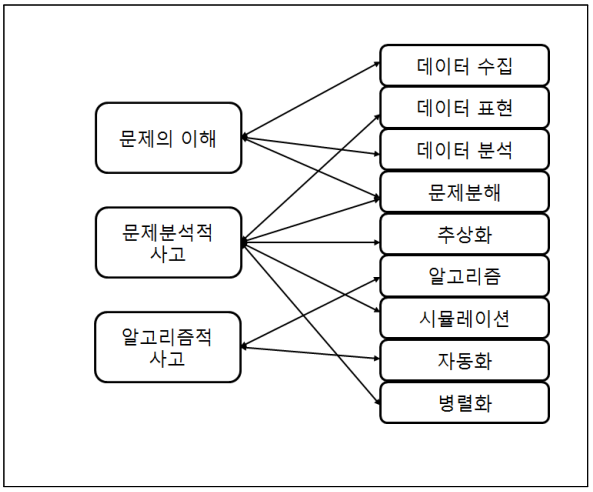
이를 바탕으로 문제의 이해, 문제분석, 알고리즘적사고와 관련된 컴퓨팅사고능력과 지적영역의 관계를 분석하여 보면 <그림 1>과 같다.

문제의 이해 단계에서는 데이터 수집과 데이터 분석과 문제 분해 개념과 밀접한 관련이 있으며 분석적 사고 단계에서는 데이터 표현과 추상화, 문제분해, 병렬화, 시뮬레이션과 관련이 있다는

<표 9> 단계별 컴퓨팅 사고력과의 관계

구분	문항 / 영역	문제의 이해 단계				분석적 사고 단계						알고리즘적 사고 단계		
		문제 독해	데이터 분석	필요한 자료의 탐색	조건 탐색	데이터 분석과 표현	문제분해	분해된 문제 통합	추상화	병렬화	시뮬레이션	간략한 해결 절차	알고리즘	코딩
컴퓨팅 사고력	데이터 수집	58.82%	17.65%	35.29%	5.88%		5.88%	5.88%						
	데이터 표현		17.65%	29.41%	17.65%	64.71%	11.76%	5.88%		10.00%	5.88%	11.76%	6.25%	5.88%
	데이터 분석	52.94%	88.24%	41.18%	52.94%	11.76%	11.76%	29.41%	5.88%	10.00%	5.88%	5.88%	6.25%	5.88%
	문제 분해	23.53%	29.41%	47.06%	58.82%	11.76%	82.35%	47.06%	23.53%	10.00%	5.88%	17.65%	6.25%	5.88%
	추상화	35.29%	11.76%	23.53%	23.53%	52.94%	23.53%	58.82%	70.59%	10.00%	11.76%	47.06%	12.50%	11.76%
	알고리즘	11.76%	11.76%	17.65%	11.76%	41.18%	17.65%	23.53%	23.53%	20.00	11.76%	58.82%	87.50%	41.18%
	자동화	5.88%	5.88%	5.88%	5.88%	5.88%	5.88%	17.65%	29.41%	10.00%	29.41%	23.53%	31.25%	76.47%
	병렬화	11.76%	5.88%	5.88%	5.88%	5.88%	11.76%	5.88%	5.88%	90.00%	11.76%	5.88%	12.50%	23.53%
	시뮬레이션	5.88%	5.88%	5.88%	5.88%	17.65%	5.88%	11.76%	17.65%	10.00%	94.12%	11.76%	31.25%	23.53%

것을 알 수 있다. 또한 알고리즘적 사고 단계에서는 알고리즘과 자동화와 관련이 있음을 확인 할 수 있다.



<그림 1> 단계별 사고과정과 컴퓨팅사고력관계도

문제이해단계에서는 문제를 읽어가며 자료를 분석함으로써 문제를 이해하기 위한 절차로 구성되어진다. 이는 컴퓨팅사고력 세부요소에서 데이터분석, 데이터 수집, 문제분해의 사고와 관련이 있다고 볼 수 있다.

또한 문제 분석적 단계에서는 이해한 문제를 토대로 문제를 해결 하고자 다양한 방법으로 분석을 하는 사고들로 구성되어진다. 이는 컴퓨팅사고력 세부 요소에서 데이터표현, 문제분해, 추상화, 병렬화, 시뮬레이션의 사고 요소와 관련이 있다고 볼 수 있다.

마지막으로 알고리즘적 사고 단계에서는 컴퓨터가 해결할 수 있는 문제를 해결하기 위한 절차를 나타낸다. 이는 컴퓨팅사고력 세부 요소에서 알고리즘과 자동화와 관계가 있는 것으로 볼 수 있다.

따라서 문제를 이해하고 분석하고 알고리즘화 하는 일련의 사고과정들은 컴퓨팅사고력의 세부 영역과 모두 '관련 있음'으로 요약할 수 있다.

6. 결론

지금은 소프트웨어 중심 사회로 많은 나라들은 컴퓨팅사고력기반의 문제해결능력을 갖춘 창의적 인재 양성을 위해 온 힘을 쏟고 있다. 이에 직업생활과 밀접한 관련이 있는 대학에서는 소프트웨어 교육에 대한 중요성을 인식하며 비IT계열전공자들을 대상으로 한 프로그래밍 교육을 진행하고 있으며 이는 더욱 확산되는 추세이다.

이에 본 연구는 대학에서의 대학생들이 프로그래밍을 어려워하는 이유 중 사고 영역과 관련된 요소들을 밝히고 그 요소들과 컴퓨팅사고력와의 관련성을 분석하여 관계있는 요소들을 탐색하였다.

대학생들이 어려워하는 사고영역은 문제를 읽어 이해하는 단계부터 코딩하는 단계까지 모두 어려워한다는 것을 알 수 있었고 이 상태에 코딩을 한다면 학생들의 인지적 부담은 가중 될 수밖에 없음을 생각 해 볼 수 있다. 따라서 프로그래밍 전과 후로 나누어 프로그래밍과목을 학습하기 전에 사고에 대한 학습이 이루어져야 하며 이를 학습한 후 프로그래밍 과목을 학습한다면 학생들의 인지적 부담이 감소되어 프로그래밍을 통한 문제 해결학습에 집중할 수 있을 것이다.

코딩하기까지의 모든 과정의 사고영역의 세부 사고요소를 알아내기 위해 컴퓨팅사고력과의 관련 있는 항목들을 탐색해 본 결과 문제 이해 단계에서는 데이터 수집, 데이터분석, 문제 분해와 관련 있음을 알 수 있고 문제 분석적 사고과정에서는 데이터표현, 문제 분해, 추상화, 시뮬레이션, 병렬 화와 관련 있음을 알 수 있었으며 알고리즘적 사고과정에서는 알고리즘과 자동화와 관련이 있음을 알 수 있었다. 즉 프로그래밍을 어려워하는 사고영역들은 컴퓨팅사고력의 9가지 사고 요소와 모두 관계가 있음을 알 수 있다. 이는 프로그래밍 과정에서 필요한 사고과정은 컴퓨팅사고력의 세부 요소를 학습함으로 사고능력을 함양할 수 있다는 것으로 해석 할 수 있다.

따라서 프로그래밍과목을 학습하기 전에 대학생들에게 컴퓨팅사고력의 9가지 세부 요소를 토대로 프로그래밍 사전 사고 학습을 하도록 한 후 이러한 사고 능력을 바탕으로 본격적으로 프로그래밍을 학습한다면 창의적으로 문제를 해결하는 능력을 함양하는데 집중할 수 있을 것이다.

참 고 문 헌

[1] http://news1.kr/articles/?2184259

[2] http://www.hankyung.com/news/app/newsview.php?aid=2015072036951

[3]http://www.hankyung.com/news/app/newsview.php?aid=2014072381388&intype=1

[4]http://www.usline.kr/news/articleView.html?idxno=4924

[5] Douglas Rushkoff (2010). *Program or be programmed: Ten Commands for a digital age. OR Books,* New York.

[6] 최현종(2011). 대학 프로그래밍 강좌를 위한 프로그래밍 교육 프레임워크. **한국컴퓨터교육학회논문지** 14(1), 69-79.

[7] 장선희(2011).**정보교과교육에서 사전 프로그래밍 교육의 도구로써 퍼즐의 활용 제안**. 석사학위 논문, 고려대학교.

[8] Judith Gal-Ezer(1995). *A Pre-Programming Introduction to Algorithmics.*

[9]RobFaux(2005).Impact of Preprogramming Course Curriculum on Learninginthe First Programming Course. Education, *IEEE Transactions* on ,49(1)

[10] RussellL.Shackelford and Richard J.LeBlanc(1997).*Introducing Computer Science Fundamentals Before Programming,* Jr.College of Computing Georgia Institute of Technology Atlanta, GA 30332-0280.

[11]http://www.open.edu/openlearn/science-maths-technology/computing-and-ict/introduction-computational-thinking/content-section-5

[12]http://www.nationalacademies.org/

[13]http://www.csta.acm.org/Curriculum/sub/CompThinking.html

[14] 최형신(2013), Computational Thinking 교육 및 평가 접근에 대한 고찰. **한국정보교육학회 학술논문**,4(1), pp.283-288.

[15] 노경섭(2014).**제대로 알고 쓰는 논문 통계분석 : SPSS & AMOS 21.**한빛아카데미(주)

[16] 오인환(1992). **사회조사 방법론-오차요인 집중 연구**. 나남.

오 경 선

1999 상명대학교 전산학과(학사)
2002 상명대학교 컴퓨터교육과 (교육학석사)
2011 ~ 현재 성균관대학교 컴퓨터교육과 박사과정
관심분야: 컴퓨터교육, SW교육, 컴퓨팅사고력
E-Mail: skyal@skku.edu

안 성 진

1988 성균관대학교 정보공학과 (학사)
1990 성균관대학교 정보공학과 (석사)
1998 성균관대학교 정보공학과(박사)
1990 ~ 1995 KIST/SERI 연구원
1996 정보통신기술사
1999 ~ 현재 성균관대학교 컴퓨터교육과 교수
관심분야: SW교육, 정보윤리, 정보보안
E-Mail: sjahn@skku.edu