스크래치 프로그램을 활용한 프로그래밍 교육에 대한 비전공자의 인식 연구

오미자†

요 약

컴퓨팅 사고력의 중요성이 강조되면서, 대학에서는 소프트웨어 교육을 필수 강좌로 오픈하고 있다. 따라서 본 연구에서는 비전공자 학생들이 느끼는 프로그래밍에 대한 기존의 인식과 실제 수업 후 느끼는 인식을 살펴보고자 하였다. 이를 위해 15주 동안 스크래치 프로그램을 활용하여 프로그래밍 교육을 진행하였으며, 214명의 답변을 받아 내용을 분석하였다. 분석 결과 비전공자 학생들의 74%가 프로그래밍에 대해 이전의 경험이 없었으며, 87%가 프로그래밍에 대해 어려움을 느끼고 있었고, 69.7%가 프로그래밍 교육이 필요없다고 답하였다. 이러한 부정적 인식을 변화시키기 위해 몇 가지를 제안하고자 한다. 첫째, 교수는 수업 전 프로그래밍 교육의 필요성과 목적 및 내용을 분명히 전달해야한다. 둘째, 필수보다는 선택교과목으로 지정되어야 한다. 셋째, 전공과의 통합 내용이나, 취·창업에 연계한 교육과정 및 내용 개발이 필요하다.

주제어 : 대학생, 프로그래밍 교육, 인식, 스크래치

Non-Major Students' Perceptions of Programming Education Using the Scratch Programming Language

Mi-Ja Oh†

ABSTRACT

As an emphasis has been put on the importance of computational thinking, universities have opened software educational programs as required basic courses.. Therefore this study aimed to examine non-major students' perceptions of programming before and after they had programing education. To this end, this study performed programming education for 15 weeks using the Scratch programming language, and then conducted a questionnaire survey. This study analyzed responses from 214 students. According to the results of the analysis, 74 % of the non-major students had no previous experience with programming, 87% felt that programming was difficult, and 69.7% answered that they did not need programming education. To change these negative perceptions of programming, this study made the following suggestions. First, the professor should clearly convey the needs, purposes, and content of programming education to students prior to class. Second, programming should be designated as an optional course rather than required one. Third, it is necessary to develop content integrated with majors, or educational programs or content connected to getting a job or starting a business.

Keywords: University Students, Programming Education, Perceptions, Scratch

† 정 회 원: 건국대학교 대학교육혁신원 교수
논문접수: 2016년 10월 13일, 심사완료: 2017년 1월 16일, 게재확정: 2017년 1월 26일

1. 서론

21세기 사회는 첨단과학과 통신기술의 발달로 정보와 지식이 급증하는 지능 정보화 사회가 될 것이라 전망한다[1]. 이러한 사회에서 교육의 방향은 단순히 지식을 전달, 습득하는 기존의 형태가 아닌 '학습하는 방법을 학습'할 수 있도록 하는 능력을 갖추어야 한다고 말한다[2]. 이러한 능력은 문제 해결을 위해 스스로 정보를 찾고, 선별하여, 필요에 따라 기존의 정보에서 새롭게 정보를 창출하는 사고를 할 줄 아는 '사고의 유연성'을 갖춘 인간상을 요구하고 있는 것이다. 이러한 변화는 최근 '컴퓨팅 사고력(Computational Thinking)'의 중요성에 대한 인식이 높아지는 점에서도 확인할 수 있다[3].

컴퓨팅 사고력을 높이기 위해 여러 국가에서는 소프트웨어 교육 의무화 정책을 시행하고 있다. 우리나라에서도 국가교육과정 정보센터에서 발표한 2015 정보 개정 교육과정을 살펴보면 소프트웨어 교육이 강화된 것을 확인할 수 있으며, 세부적으로는 컴퓨팅 사고력 향상을 위해 추상화(abstraction) 능력, 프로그래밍 능력 등이 핵심 개념으로 설정되었다[4]. 실제 프로그래밍은 특정 문제를 해결하기 위한 논리적인 사고 능력을 필요로 한다. 예를 들어 프로그래밍을 통해 게임을 개발하거나, 캐릭터를 움직이기 위해 고민을 하고, 고민을 하는 과정에서 체계적인 사고능력이나 논리력이 수반되기 때문이다[5][6]. 따라서, 컴퓨팅 사고력 향상을 위한 소프트웨어의 교육 방법으로 프로그래밍이 제시되는 것은 어쩌면 당연한 결과라고 볼 수 있다.

이러한 컴퓨팅 사고력의 필요성은 앞서 제시한 대로 2015 정보 교과의 교육과정 개정으로 이어졌으며, 2018년도부터 초·중등 교과과정에서 소프트웨어 교육이 의무화로 지정되는 결과를 가져왔다. 또한, 정부 주도하에 2014년도부터 소프트웨어 교육 선도학교 및 연구학교를 운영하고 있고, 2016년 현재 초등학교(479개), 중학교(300개), 고등학교(121개)등 총 900개 학교가 선정되어 학생들의 논리력과 창의력, 문제해결력을 증진시키기 위한 목적으로 운영되고 있다[7]. 이러한 목적과 동일하게 미래창조과학부에서도 '소프트웨어 중심 대학 사업'으로 소프트웨어 교육을 강조하고 있고 이에 맞춰 여러 대학에서는 신입생 전체를 대상으로 소프트웨어 교육과정을 개설하고 프로그래밍 교육을 운영하고 있다[8].

프로그래밍에 대한 교육적 효과는 여러 연구 결과에서 확인할 수 있다. 이태욱 외(2011)는 프로그래밍을 하는 과정 속에서 학습자는 학습의 주체로써 해결 방안을 찾고 이를 즉각적인 피드백을 통해 문제해결 능력, 절차적 사고 능력, 추론 능력 등을 기를 수 있다고 하였다[9]. 박지영(2016)은 프로그래밍 과정에서 오류가 발생할 경우, 오류를 해결하는 과정 안에서 학습자 스스로 자신감이 향상될 수 있다고 제시하였고[10], 이외에도 팀 과제로 수행하면 협동학습, 자기주도학습이 가능하다는 등의 프로그래밍에 대한 효과를 제시한 바 있다[9][11].

그러나 이와 같은 연구 결과는 대부분이 초·중·고를 대상으로 한 연구 결과로, 대학에서 이를 그대로 적용하기란 무리가 있다. 대학의 경우는 '소프트웨어 중심 대학 사업'으로 소프트웨어 교육의 중요성에 대해 생각하게 되었고, 개선을 위한 고민이 시작되었다. 이러한 고민에는 실제 대학생도 프로그래밍에 대한 교육적 효과를 기대할 수 있을지, 효과가 있다면 지속적으로 수행하기 위해 필요한 것들은 무엇일지, 프로그래밍에 대한 비전공자와 전공자 학생의 이해와 수준에 따라 어떠한 교육과정이 고려되어야 할지 등이 포함되어야 한다. 만약, 이러한 것들이 고려되지 않으면 학습자는 프로그래밍에 대한 어려움이나 복잡함을 느끼게 되고, 이는 학습자 부담으로 이어져, 흥미나 성취도 등이 떨어지게 될 것이기 때문이다.

따라서, 본 연구에서는 비전공자 학생들이 프로그래밍에 대해 기존에 갖고 있던 생각과 실제 약 15주 동안 프로그래밍 수업 후 느꼈던 효과에 대해 설문을 활용하여 인식을 살펴보고, 향후 개선 방향을 제안하고자 한다. 이를 위해 인문·사회·예체능 계열 학생들에게 교육용 프로그래밍의 한 종류인 스크래치를 활용하여 교육을 진행하였다. 물론 일반 프로그래밍 언어를 통해 프로그래밍 교육이 가능하지만, 스크래치를 활용한 이유는 교육용 프로그래밍 언어로써 비전공자도 쉽게 학습

이 가능한 것으로 알려져 있기 때문이다[6].

2. 이론적 배경

2.1 컴퓨팅 사고력

21세기 미래 사회의 중요한 능력으로 컴퓨팅 사고력을 꼽는다. 컴퓨팅 사고력에 대해서는 여러 정의들이 있지만 지넷 윙(Jennette M. Wing)의 정의가 주로 회자된다. 지넷 윙은 컴퓨팅 사고력은 컴퓨터 과학자뿐만 아니라 누구나 배워서 활용해야 한다는 점을 강조하면서 '문제를 수립하고 해결책을 만들어 컴퓨터 시스템을 통해 효과적으로 수행하는 사고 과정'이라고 하였다. 또한, 디지털 네이티브(Digital Native)세대 학생들에겐 컴퓨팅 사고력은 읽기, 쓰기, 셈하기 등과 마찬가지로 매우 기본적인 능력임을 강조하며 <표 1>과 같이 9가지의 핵심 요소를 설명하였다[12][16].

ISTE&CSTA(2011)는 컴퓨팅 사고력을 자료를 논리적으로 분석·추상화를 통해 표현하고, 알고리즘화 및 자동화하는 등의 문제해결과정을 일반화하고 전이하는 능력이라고 제시하였다[13]. Dode, Mishra, Voogt(2013)은 컴퓨팅 사고력 개념을 토대로 고등학교 컴퓨터 과학의 교육과정을 개발하기도 하였다[14]. 위에서 제시한 정의를 토대로 컴퓨팅 사고력은 '데이터를 수집 분석하여, 알고리즘을 생성하고, 이를 시뮬레이션하여 문제해결방법을 모색하는 것이며, 이를 위해 다양한 전략을 사용하는 것'이라고 정의해볼 수 있다.

<표 1> 컴퓨팅 사고력의 9가지 핵심 요소

요소	설명
자료수집	• 해결해야 하는 문제와 관련하여 알맞은 자료를 모으는 과정
자료분석	• 수집된 자료를 이해하고, 해당 자료에 대해 분류하고 분석하는 과정
자료표현	• 분석된 자료의 내용을 그래프, 차트, 이미지 등으로 표현하는 과정
문제분해	• 문제에 대해 이해하고 이를 해결하기 위해 문제를 나누어 분석하는 단계
추상화	• 복잡한 문제를 기본적인 개념 정의를 기반으로 설정하는 과정
알고리즘과 절차화	• 문제를 해결하기 위한 과정을 순서적으로 표현해내는 과정
자동화	• 순서적으로 표현되어 있는 것을 컴퓨터가 수행할 수 있도록 해결책을 제시하는 과정
시뮬레이션	• 자동화를 통해 나온 결과물로써, 문제 해결을 위해 만든 모델 실행 과정
병렬화	• 목표 달성을 위한 작업을 동시에 수행하도록 자원을 구성하는 과정

<출처: Jennette M. Wing(2011). Computational Thinking. CSTA Workshop. 4 March[16]>

2.2 국내외 소프트웨어 교육 정책

앞서 언급한대로 컴퓨팅 사고력을 향상시키기 위해 국내외적으로 소프트웨어 교육 혹은 정보교과 교육을 필수로 지정하여 운영하고 있고 점점 확대되어가는 추세이다. 이러한 국내외 교육과정을 확인해보면 공통적으로는 일방향형태의 지식 전달 위주 교육 보다는 활동 위주의 교육들로 진행되고 있다는 점을 확인할 수 있다. 소프트웨어 교육의 중요성을 깨닫고 앞서 확대하여 시행한 나라들로는 미국, 영국, 일본, 중국, 인도, 이스라엘 등이 있다[6].

2.2.1 국외

미국의 경우 소프트웨어 교육을 발빠르게 적용·실행하고 있다. 국가차원에서 K-12를 대상으로 2011년도에 Computer Science Standards(컴퓨터과학표준)를 개발하여 소프트웨어 교육 방향을 제시하였고, 컴퓨터 과학을 필수로 지정하는 학교가 매년 증가하고 있다. 교육 과정은 총 3단계 레벨로 구성되어 있는데, 1단계는 저학년~6학년까지로 Computer Science and Me(컴퓨터 과학과 나), 2단계는 6학년~9학년 대상으로 Computer Science and Community(컴퓨터 과학과 사회), 3단계는 9학년~12학년 대상으로 3개의 과정으로(Computer Science in the Modern World-Principles-Topic in Computer Science) 세분화하여 제시하고 있다[6][7][15].

영국의 경우는 2013년 9월 National curriculum in England: Computing Programmes of study를 통해 1~4단계의 Key Stage단계로 나누어 5세~16세까지를 대상으로 정보 교육 과정을 제공하고 있다. 이러한 커리큘럼을 기반으로 2014년 9월부터 코딩 및 프로그래밍 교육을 초·중·고 필수 교과 과정으로 시행 중이며, 초등학교의 경우 주당 50분 이상의 교육을 진행하고 있다. 내용도 기존

의 워드, 엑셀 등의 소프트웨어 활용 능력 중심에서 스스로 소프트웨어를 만들 수 있도록 프로그래밍(코딩교육)으로 변화하는 추세이다[7][8][15]. 다음 <표 2>는 국외 소프트웨어 교육 현황을 정리한 표이다.

<표 2> 국외 소프트웨어 교육 현황

국가	필수/선택	과목명	대상
영국	필수	Computing	초 · 중 · 고
일본	둘중선택	사회와정보, 정보과학	고등학교
	필수	기술 175시간중 55시간	중학교
중국	필수2+선택2	정보기술 인프라	고등학교
	선택	종합실천활동 5개 영역	초3~중학교
인도	필수	Computer Masti	초 · 중 · 고
미국	선택	정보과학	초 · 중 · 고

2.2.2 국내

국내의 경우 1990년에 처음 시작된 컴퓨터 정규 교육이 2008년도에 완전 폐지된 후 특수 고등학교를 제외하고 일반 고등학교에서는 선택적으로 '정보'교과를 채택하여 시행해왔다[17]. 그러나 2015년 7월 미래창조과학부와 교육부는 초·중등을 대상으로 2018년부터 소프트웨어 교육 필수화 계획을 발표하였다. 주요 내용은 2015년 9월 교육과정(시수, 교육내용 등)에 대한 내용 발표, 2016년 이에 맞는 교과서 개발, 2018년 전체 확대 적용을 목표로 진행한다는 내용이었다. 이를 실행하기 위해 2016년 10월 현재 약 900여개의 소프트웨어 선도학교가 선정되어 운영 중이며, 대학에서도 소프트웨어 중심 대학으로 14개 대학이 선정되어 운영 중이다. 소프트웨어 중심대학의 목표는 소프트웨어 교육은 창의적 아이디어를 소프트웨어로 구현하는 사고력 교육으로, 이에 대한 목표는 창의적 아이디어를 소프트웨어로 구현할 수 있는 문제해결력을 갖추는 것이다[7][15].

2.3 교육용 프로그래밍 언어

교육용 프로그래밍 언어란, 실제적 문제를 해결하고자 프로그램을 개발하는 형태의 범용적 프로그램이 아닌, 교육적 목적을 달성하기 위해 개발된 프로그래밍 언어라고 말한다[18][19]. 물론 일반 프로그래밍 언어도 교육에 활용이 가능하긴 하지만, 일반 프로그래밍 언어는 문법이나 함수 사용법을 익혀야하고, 이러한 것들은 대부분이 이미지가 아닌 텍스트로 구성되어 있다. 하지만, 교육용 프로그래밍 언어는 문법이나 함수를 외우지 않아도 프로그래밍에 필요한 알고리즘에 대해 이해, 사고, 해결 능력을 키울 수 있는 장점을 가지고 있고, 이미지로 표현되기 때문에 나이가 어리거나, 비전공자도 쉽게 학습이 가능하다는 장점을 가지고 있다[6].

이러한 이미지 및 블록 형태의 교육용 프로그래밍 언어로는 대표적으로 한국에서 개발한 엔트리와 미국 MIT에서 개발한 Scratch 등이 있다.

2.3.1 엔트리(Entry)

엔트리는 무료 소프트웨어 교육을 위해 개발한 교육용 프로그래밍 언어로 2014년 4월에 서비스를 시작하여, 현재까지 2만 2천여 명의 회원을 보유하고 있다(2015년 7월 기준). 엔트리의 경우, 블록형 프로그래밍 언어로 직관적이고, 한글 명령어로 이루어져 있는 것이 장점이다. 또한, 스크래치에 비해서는 스스로 학습이 가능하도록 기본 미션 20단계와 응용 미션 10단계 등의 문제해결과정을 튜토리얼 형태로 갖추고 있어 학생들도 쉽게 명령어를 이해하고 습득할 수 있다. 이외에도 엔트리봇 보드 게임을 활용하여 언플러그드 교육이 가능하도록 지원하고 있으며, 특별한 기능으로는 학급관리기능이 있다. 이 기능은 교사가 나만의 학습공간을 만들어 학생의 아이디어와 비번을 생성하거나 이전에 가입된 학생들의 초대도 가능하다. 이러한 기능을 통해 작품을 모아보거나, 공유하고, 소통의 기회를 제공할 수 있다[17].

2.3.2 스크래치(Scratch)

스크래치는 2007년 6월 MIT대학의 미디어랩과

UCLA의 연구자가 공동으로 개발한 초보 프로그래머를 위한 멀티미디어 기반의 프로그래밍 언어라고 볼 수 있다. 최근 스크래치는 교육용 프로그래밍 언어로 많이 활용되고 있는데, 그 이유는 블록 형태로 프로그래밍이 가능하고, 프로그래밍과 관련한 개념 즉 구조와 원리를 쉽게 이해하고 습득할 수 있도록 구성되어 있기 때문이다[17]. 다양한 멀티미디어 기능을 제공하기 때문에 게임, 애니메이션, 스토리를 구성하는 창의적 활동이 가능하며, 온라인을 기반으로 하기 때문에 다른 학습자와도 상호작용이 가능하다[18]. 총 102개의 블록으로 기능에 따라 색깔별로 구성되어 있어 시각적으로도 뛰어나다. 추가적인 장점으로는 총 42개 언어로 개발되어있어, 외국인 학생들도 활용이 가능하다. 이러한 이유로 본 연구에서도 스크래치를 활용하였다. 한글 지원은 2008년부터 시작되었다.

2.4 스크래치 프로그래밍 관련 선행 연구

프로그래밍은 문제를 파악하고, 해결을 위해 알고리즘을 설계·실행함으로써 문제해결에 바탕을 둔 과정으로 이를 통해 논리적 사고, 창의적 사고를 경험할 수 있어, 여러 연구에서 LOGO와 같은 교육용 프로그래밍 언어에 대한 교육적 효과 및 필요성을 주장한다[21]. 본 연구에서 활용한 스크래치에 대한 연구 결과들을 살펴보면 다음과 같다

조성환 외(2008)은 중학교 1학년을 대상으로 스크래치 교육을 한 결과, 문제해결력 향상, 논리적 사고 향상[22], 긍정적 태도에 효과적이라고 하였으며, 송정범(2008)은 스크래치를 활용한 프로그래밍 교육이 내재적 동기와 문제해결력에 도움을 준다고 하였다[23].

노희진(2015)은 과학 수업에 스크래치를 적용한 결과 학생의 흥미도가 향상되었고, 수업참여율이 높아졌다고 하였다[24]. 또한 산출물을 얻는 과정에서 뿌듯함을 얻을 수 있고, 학생들은 이를 더 오래 기억한다고 하였다.

안형진,마대성(2013)은 초등학생을 대상으로 알고리즘 교육에 필요한 교육과정을 개발하였으며, 직접 구성해보는 활동을 통해 자기 주도적 학습 능력과 문제 해결력을 신장하는데 도움이 될 것이라 제시하였다.

이러한 효과성 연구결과는 <표 3>과 같이 정리되었지만 대부분이 초·중등에 대한 사례일 뿐, 고등교육에서 성인학습자를 대상으로 한 연구 사례는 극히 적다. 따라서 본 연구에서는 성인학습자인 대학생을 대상으로 스크래치를 교육하고 이에 대한 인식을 살펴보고자 하였다.

<표 3> 스크래치의 교육적 효과 선행 연구비교

제목	연구자	대상	결과
스크래치를 이용한 프로그래밍 수업 효과	조성환 외 (2008)	중등	사고력향상 (문제해결력, 논리적 사고)
CPS에 기반한 스크래치 EPL이 문제해결력과 프로그래밍 태도에 미치는 효과	송정범 (2008)	초등	사고력향상 (문제해결력), 성취도향상
Scratch 프로그래밍 교육이 논리적 사고력에 미치는 영향	이영주 (2010)	초등	사고력향상 (문제해결력)
스크래치를 이용한 STEAM기반 교육프로그램 개발 및 적용	오정철 외 (2012)	초등	흥미도 향상 성취도 향상
문제해결력 증진을 위한 초등학교 Scratch 교육과정 개발	안형진 외 (2013)	초등	사고력향상 (자기주도, 문제해결력)
문제해결력 증진을 위한 초등학교 scratch 교육과정 개발	노희진 (2015)	중등	흥미, 성취도향상

3. 연구 방법

3.1 연구 대상과 연구 설계

본 연구의 연구 대상은 서울특별시 소재의 K대학의 2016학년도 인문·사회·예체능 계열의 신입생이다. 해당 학생들은 '컴퓨팅적사고'라는 교양 필수 교과목을 수강하였으며 1주부터 15주까지 스크래치라는 교육용 프로그래밍 언어를 활용하여, 프로그래밍(코딩교육)교육을 받았다. 연구를 위한 설문조사는 15주 수업 이후인 2016. 5. 31 ~ 6. 10에 약 10일 간 실시하였다. 설문 구성은 <표 3>의 연구 결과를 토대로 장희선(2013)에서 활용한 프로그래밍 학습 능력에 관한 사고력 문항과 박효

선(2016), 유지은(2014)에서 활용한 흥미도, 성취도 문항을 활용하여 구성하였다[26][27][28][29]. 전체 문항은 <표 4>와 같다.

<표 4> 설문 문항 내용

번호	문항 내용		형태
1	일반적인 변인(성별, 대학)		
2	프로그래밍(코딩교육)에 대한 이전 경험 유무		Y/N
3	프로그래밍(코딩교육)에 대한 흥미		Y/N
4	프로그래밍(코딩교육)에 어려움(난이도)		Y/N
4-1	프로그래밍(코딩교육)에 난이도에 따른 의견		서술
5	프로그래밍(코딩교육)의 필요성 유무		Y/N
5-1	필요한 이유 의견		서술
5-2	필요하지 않은 이유 의견		서술
6	흥미 향상	주입식 교육에 학습 동기와 성취를 증가시키는데 도움이 된다.	5점
7		컴퓨터 언어(기술) 교과목에 대하여 흥미를 가지는데 도움이 된다.	5점
8		컴퓨터 언어(기술) 교과목에 대하여 관심 및 이해를 증가시키는데 도움이 된다.	5점
9	사고력 향상	문제해결력 향상에 도움이 된다	5점
10		논리적사고 및 통찰력 향상에 도움이 된다.	5점
11		자기주도적 학습 능력 향상에 도움이 된다.	5점
12	성취도 향상	나는 스크래치(코딩교육)을 통해 새로운 지식이나 경험을 배울 수 있었다.	5점
13		스크래치(코딩교육)은 추후에 다른 교과나 사회에서 도움이 될 것이라 생각한다.	5점

추가적으로 기본적인 성별과, 대학구분 등의 일반적 정보를 작성하도록 하였으며, 프로그래밍(코딩교육)에 대한 이전 경험 유무, 흥미 유무, 어려움 유무와 어려움에 따른 의견 등은 서술 형태로 작성하도록 하여 응답을 받았으며, 아래와 같은 연구문제를 설정하여 확인하고자 하였다.

연구문제1. 비전공자 학생들의 프로그래밍(코딩교육)에 대한 이전의 경험은 어떠한가?

연구문제2. 프로그래밍(코딩교육) 후 필요성에 대한 인식은 어떻게 변화하였는가?

연구문제3. 성별, 이전 경험・흥미・필요성 유・무에 따라 흥미, 사고력, 성취도 향상 등 효과성을 지각하는데 차이가 있는가?

양적 통계적 자료처리를 위해 SPSS 18.0 프로그램을 사용하여, 성별, 대학별, 이전 경험 등에 대한 기술통계를 실시하였으며, 서술형 형태의 설문 결과 처리를 위해서 Nvivo10 프로그램이 활용되었다. Nvivo10은 일정한 분류체계를 가지고 복잡한 자료를 의미 있는 주제나 범주로 조직·단순화가 가능하며, 특정 주제에 대한 정보 추적이 가능하여 체계성과 투명성을 보장할 수 있기 때문이다[31][32][33].

앞서 제시한 설문 일정에 따라 설문을 진행한 결과 최종적으로 설문에 응답한 응답자는 214명으로, 세부적인 정보는 <표 5>와 같다.

대학별로는 문과대학 50명(23.3%), 상경대학 47명(21.9%), 경영대학 33명(15.4%)순으로 나타났다. 성별로는 남자 94명(44%), 여자 120명(56%)로 여자 응답자가 많았다.

<표 5> 설문 참여 대학별 결과

대학	남	여	합계
경영대학	17	16	33
문과대학	16	34	50
사범대학	8	14	22
상경대학	22	25	47
예술대학	6	16	22
정치대학	23	8	31
글로벌 융합대학	2	7	9
총(명)	94	120	214

4. 연구 결과

4.1 연구문제 1의 결과

비전공자 학생들의 프로그래밍(코딩교육)에 대한 이전 경험의 응답 결과는<표 6>과 같이 정리되었다. 응답자의 74.2%가 프로그래밍(코딩교육)에 대한 이전 경험이 없었으며, 흥미의 경우 69.7%가 흥미가 없었다고 답하였다. 또한, 실제 수업 후 느끼는 어려움의 경우는 87.0%가 어려움을 느끼고 있다고 답했다.

<표 6> 이전경험, 흥미, 어려움에 대한 대학별 결과

대학	N	경험		흥미		어려움	
		유 (명,%)	무 (명,%)	유 (명,%)	무 (명,%)	유 (명,%)	무 (명,%)
경영	33	12 (36.4)	21 (63.6)	11 (33.3)	22 (66.7)	27 (81.8)	6 (18.2)
문과	50	11 (22.0)	39 (78.0)	13 (26.0)	37 (74.0)	42 (84.0)	8 (16.0)
사범	22	4 (18.2)	18 (81.8)	8 (36.4)	14 (63.6)	20 (90.9)	2 (9.1)
상경	47	13 (27.7)	34 (72.3)	15 (31.9)	32 (68.1)	36 (76.6)	11 (23.4)
예술	22	6 (27.3)	16 (72.7)	5 (22.7)	17 (77.3)	22 (100)	0 (0.0)
정치	31	5 (16.1)	26 (83.9)	9 (29.0)	22 (71.0)	27 (87.1)	4 (12.9)
글로벌 융합	9	3 (33.3)	6 (66.7)	3 (33.3)	6 (66.7)	8 (88.9)	1 (11.1)
평균		25.8%	74.2%	30.3%	69.7%	87.0%	13.0%

프로그래밍(코딩교육)이 어려운 이유는 <표 7>과 같이 정리되었다.

<표 7> 프로그래밍(코딩교육)이 어려운 이유

어려운 이유	N(명)	비율(%)
프로그래밍 문법의 어려움 (반복문, 함수 등)	38	17.8
코딩 시 발생하는 오류에 대한 수정 어려움	34	15.9
프로그래밍에 대한 생소함	46	21.5
관련 전공이 아니어서	13	6.1
흥미 없음	31	14.5
기타	52	24.3

1위는 기타 답변으로 52명(24.3%)이 답하였다. 의견을 살펴보면 '생소하고', '어려우며', '관련 전공도 아니다'라는 복합적 답변을 한 학생들이었으며, 2위는 생소함 46명(21.5%), 3위는 문법의 어려움이 38명(17.8%)로 나타났다.

4.2 연구문제 2의 결과

프로그래밍 수업 후 필요성에 대한 인식 변화에 대한 결과는 <표 8>과 같다. '필요하다'의 응답 평균은 30.2%, '필요하지 않다'는 69.7%로 필요하지 않다고 답한 비율이 2배 이상 높게 나타났다.

<표 8> 프로그래밍(코딩교육)의 필요성 유・무

대학	N	필요성			
		유(명)	비율(%)	무(명)	비율(%)
경영	33	9	27.3	24	72.7
문과	50	14	28.0	36	72.0
사범	22	9	40.9	13	59.1
상경	47	15	31.9	32	68.1
예술	22	3	13.6	19	86.4
정치	31	8	25.8	23	74.1
글로벌융합	9	4	44.4	5	55.6
	214	62	30.27	152	69.71

프로그래밍(코딩교육)이 '필요하다'라고 답한 62명의 서술 응답 결과를 분석한 결과는 <표 9>와 같다. 분석에는 Nvivo가 활용되었으며, 분석 결과 필요한 이유로는 개인적·사회적 요인 2가지로 범주화하여 분류할 수 있었다.

<표 9> 필요한 이유

구분	내용	References	비율
개인적	교육 필요성 인식	10	11.6
	컴퓨터와 친밀감	3	3.5
	고차원적 사고에 도움	15	17.4
	새로움과 흥미로움	36	41.9
사회적	필요성 인식	19	22.1
	취업에 도움	3	3.5

개인적 요인으로는 프로그래밍 교육이 '새롭고, 흥미로운' 교육이었으며, 고차원적 사고에 도움이 될 것이라 생각하는 것으로 나타났으며, 사회적 요인은 사회에서 요구하는 흐름 때문에 필요하다고 생각하는 것으로 나타났다. 세부 의견은 <표 10>과 같다.

<표 10> 필요한 이유에 대한 의견

필요성 의견
• 컴퓨터와 친해지는 계기가 되는 것 같다. • 막상 취업을 하려할 때 길은 많을수록 좋다고 생각해서. • 정보화시대에서 사회가 요구하는 것들을 수월하게 충족. • 정보화 사회이므로 기본 코딩정도는 알면 좋다고 생각. • 문과계열에서도 이공계적으로 생각할 수 있는 힘을 배워야 한다고 생각하기 때문이다. • 체계적으로 생각하는 힘을 길러준다. • 프로그래밍(코딩교육)을 통해 생각하는 습관을 기를 수 있고 창의력 향상에 큰 도움이 될 것.

프로그래밍(코딩교육)이 '필요없다'라고 답한 152명의 응답 결과 또한 개인적·사회적 요인으로 범주화 되었으며 그 결과는 <표 11>와 같다.

<표 11> 필요하지 않은 이유

	내용	References	비율
개인적	필수교과목 선정 부정	26	15.3
	흥미 없음	4	2.4
사회적	전공과목과 무관	32	18.8
	필요성에 대한 인식 공감 미흡	60	35.3
	실용성 부족	48	28.2

개인적 요인은 프로그래밍(코딩교육)에 대한 학교의 일방적인 필수 교과목 지정에 대한 불만족 요인이 가장 컸다. 사회적 요인으로는 프로그래밍 교육에 대한 필요성에 대한 인식, 어떤 곳에 활용되는지에 대한 공감 인식이 미흡한 것으로 나타났다. 세부 의견은 <표 12>과 같다.

<표 12> 필요하지 않은 이유에 대한 의견

필요하지 않다는 의견
• 관련 전공이 아니라면 일상에서 사용할 일이 거의 없다. • 전공과목과 관련이 없는 내용이라 집중하지 않는다. • 필요성 인식 부재 • 배워서 어디에 정확히 사용하는지 모르겠다. • 인문계 학생들의 적성, 흥미에 맞지 않는 수업. • 문과 특성상 평소에 코딩 교육을 활용할만한 일이 없다.

4.3 연구문제 3의 결과

연구문제 3에 대해 확인하고자 문항에 대한 타당도 검증을 위해 탐색적 요인분석을 실시하였다.

요인분석 결과 3개 요인으로 구분되었으며, 그 결과는 다음과 <표 13>과 같다.

<표 13> 요인분석 결과

요인	내용	성분			공통성	신뢰도
		1	2	3		
흥미	기술력 향상			.568	0.758	0.885
	기술 분야 흥미 향상			.870	0.931	
	기술 분야 관심 및 이해 향상			.622	0.774	
사고력	문제 해결력향상	.799			0.858	0.89
	논리적 사고 및 통찰력 향상	.827			0.895	
	주의력집중향상	.662			0.791	
성취도	성취도 향상1		.809		0.881	0.899
	성취도 향상2		.840		0.9	

모든 측정변수는 구성요인을 추출하기 위해 주성분 분석(Principle component analysis)을 사용하였으며, 베리맥스방식을 채택하였다. 또한, 문항 선택 기준은 고유값(eigen valus)은 1.0이상, 요인 적재치 0.6이상을 기준으로 하였다. 그 결과 Bartlett 구형성 검정 .897, 유의확률도 .000으로 요인분석에 적합하였으며, 신뢰도(Cronbach's α)도 0.6이상으로 신뢰도가 있었다. 따라서, 구분되어진 요인들이 성별, 경험 유무, 흥미유무 등에 대한 차이가 있는지를 살펴보고자 t-분석을 실시하였다.

연구문제 3인 성별에 따라 흥미 향상, 사고력 향상, 성취도 향상에 대한 지각의 차이가 있을 것이다. 라는 가설을 확인하고자 분석한 결과 성별의 경우 <표 14>와 같이 나타났다.

<표 14> 성별에 따른 t-분석 결과

구분	평균		표준편차		t값	p값
	남 (n=94)	여 (n=120)	남	여		
흥미	2.64	2.61	1.19	1.06	0.244	0.808
사고력	2.58	2.62	1.17	1.06	-0.265	0.791
성취도	2.83	2.73	1.28	1.19	0.573	0.568

*p<.05

그 결과 흥미의 t값은 0.244, 사고력은 -0.265, 성취도는 0.573로 t값이 ±1.96에서 차이가 발생하지 않았고, 유의확률이 0.05보다 큼에 따라서 성

별에 차이가 없는 것으로 나타났다.

이전 프로그래밍(코딩교육)에 대한 경험에 따라 흥미 향상, 사고력 향상, 성취도 향상에 대한 지각의 차이가 있을 것이다. 라는 가설을 확인하고자 분석한 결과 t값은 1.135, 사고력은 1.235, 성취도는 0.238로 t값이 ±1.96에서 차이가 발생하지 않았고, 유의확률이 0.05보다 큼에 따라서 이전 경험 여부에는 차이가 없는 것으로 나타났다.

<표 15> 경험 유무에 따른 t-분석 결과

구분	평균		표준편차		t값	p값
	경험 有 (n=94)	경험 無 (n=120)	有	無		
흥미	2.77	2.57	1.03	1.15	1.135	0.258
사고력	2.76	2.55	1.06	1.12	1.235	0.218
성취도	2.95	2.72	1.16	1.25	0.238	0.219

*p<.05

기존 흥미 유무에 따라 흥미 향상, 사고력 향상, 성취도 향상에 대한 지각의 차이가 있을 것이다. 라는 가설을 확인하고자 분석한 결과 t값은 6.800, 사고력은 5.387, 성취도는 6.393로 t값이 ±1.96에서 차이가 발생하였으며, 유의확률(p값)이 0.05보다 작음에 따라 흥미 유·무에 따라 차이가 있는 것으로 나타났다. 구체적으로 흥미가 있는 집단의 평균이 3.35, 3.19. 3.53으로 없는 집단에 비해 더 향상되었다고 지각하고 있는 것으로 나타났다.

<표 16> 흥미 유무에 따른 t-분석 결과

구분	평균		표준편차		t값	p값
	흥미 有 (n=94)	흥미 無 (n=120)	有	無		
흥미	3.35	2.31	0.87	1.07	6.800	0.000*
사고력	3.19	2.35	0.94	1.08	5.387	0.000*
성취도	3.53	2.45	1.03	1.17	6.393	0.000*

*p<.05

프로그래밍(코딩교육)의 필요성 유·무에 따라 흥미 향상, 사고력 향상, 성취도 향상에 대한 지각의 차이가 있을 것이다. 라는 가설을 확인하고자 분석한 결과 t값은 7.417, 사고력은 6.566, 성취도는 8.089로 t값이 ±1.96에서 차이가 발생하였으며, 유의확률(p값)이 0.05보다 작음에 따라 필요성 유·무에 따라 필요성이 있다고 느끼는 집단이 더 향상되었다고 지각하는 것으로 나타났다.

<표 17> 필요 유무에 따른 t-분석 결과

구분	평균		표준편차		t값	p값
	필요 有 (n=94)	필요 無 (n=120)	有	無		
흥미	3.42	2.3	0.87	1.05	7.417	0.000*
사고력	3.31	2.31	0.9	1.05	6.566	0.000*
성취도	3.71	2.39	0.97	1.12	8.089	0.000*

*p<.05

5. 결론

소프트웨어 교육의 효과성으로 안성진(2015)은 프로그래밍(코딩교육)을 통해 문제해결력 20.4% 향상, 논리적 사고능력 37.5%향상 등 교육적 효과를 제시한 바 있으며, 피수영(2016)은 프로그래밍을 통해 비전공자가 전공영역과 융합하여 다양한 문제해결 능력 향상을 가져올 수 있다고 하였다[30][31]. 또한, 프로그래밍(코딩교육)은 실행 후에 결과에 대해 반성의 단계를 거치게 되고, 동료 학습자와 해결 방향을 모색하고, 이러한 과정을 통해 학습에 대한 재미, 열정, 긍정적인 학습의 태도를 갖추게 되며, 성취감과 문제해결 능력, 절차적 사고 능력을 기를 수 있다[9][10][11].

이러한 프로그래밍 교육에 대한 효과는 대학에서도 일부 공감하고 있고, 실제 전체 교육과정에 반영에도 노력하고 있지만, 비전공자에 대한 배려나 구체적 방향이나 계획, 준비 등이 이루어지지 않은 상태에서 진행되고 있어 반발이 있는 상태이다. 따라서 본 연구에서는 이러한 문제의식을 가지고 비전공자인 대학 1학년생을 대상으로 15주의 스크래치를 통해 프로그래밍(코딩교육)을 진행하고 효과에 대한 인식을 살펴보고자하였으며, 연구 결과는 다음과 같다.

첫째, 비전공자 학생들의 74%가 프로그래밍에 대해 이전의 경험이 없었으며, 87%가 프로그래밍에 대해 어려움을 느끼고 있는 것으로 나타났다. 이러한 문제는 학생이 프로그래밍(코딩교육)에 대한 이해 부족과 필요성 인식 부족에서 오는 것으로 판단할 수 있다. 즉, 프로그래밍 교육이 단순히 프로그램을 개발하는 사람들을 위해 필요한

교육이라고 인식하고 있는데 따른 것이다. 이를 해결하기 위해서는 교수자는 프로그래밍 교육의 필요성과 목적, 이를 통해 가질 수 있는 효과 등에 대해 명확히 전달해야 필요가 있다. 또한, 단순히 코드를 외우는 형태의 교육 방법이 아닌 문제해결학습의 형태로의 교육 방법의 변화가 필요하다. 또한 팀 기반의 활동을 통해 서로의 의견을 지속적으로 논의하는 방법을 병행함으로써 함께 풀어가는 학습의 형태를 가지고 가야할 것이다. 따라서, 추후 교육 방법의 변화에 따른 집단의 차이에 대한 분석이 필요할 것으로 보인다. 또한, 비전공자 집단에서도 계열별 차이를 보는 것도 이루어져야할 것이다.

둘째, 비전공자 학생들은 30.3%가 프로그래밍 교육이 '필요하다'라고 말했으며, 69.7%가 프로그래밍 교육은 '필요없다'고 느껴 부정적 인식이 2배 이상으로 높았다. 이 같은 결과 결국 대학에서 교육과정 개편 시 충분한 이해와 의견 수렴이 필요함을 보여주고 있다. 대학의 교육과정은 교수요목을 분석하고, 사회 변화에 따른 시대적 요구를 반영해야 하며, 교양 및 전공 영역에서 다양화를 추구하는 것이 맞다. 다만 국가재정지원사업 수주를 목적으로 짧은 시간에, 일방적 교육과정으로의 단행은 최대한 지양해야할 것으로 보인다.

셋째, 프로그래밍 교육의 흥미 향상, 사고력 향상, 성취도 향상 등의 효과성에 대해 성별 및 이전의 경험 유·무의 집단 간 차이는 발생하지 않았지만, 흥미 유·무, 필요성 유·무에 대해서는 집단 간 차이가 나타났다. 이와 같은 결과는 흥미가 있는 집단, 필요성이 있다고 느끼는 집단에서 더 높은 사고력 향상과 성취도 향상을 가져올 수 있다는 것으로 해석할 수 있다. 즉, 대학이 사회변화에 대해 대처하기 위해 프로그래밍(코딩교육)에 대한 교과목을 개발하여 운영할 수 있지만, 이와 같은 교과목이 일방향적인 필수 교과목으로의 지정보다는 선택적 교과목으로 지정되어야하며, 선택적 교과목 중에서도 수준별로 구분하여, 전공자가 아닌 비전공자에게도 새로운 영역 혹은 융합 영역에 대해 기회를 제공하는 방향으로 개선되어야 함을 확인할 수 있었다. 또한, 수업의 방법도 일방향적 문법의 이해나 실습에서 벗어나 팀 기반의 문제해결학습 혹은 학습자 개인별 수준을 고려하여 수준별 과제를 제시가 가능하도록 하여 성취도를 높이는 형태의 수업 방법으로의 변화를 고려해야한다. 또한, 전공 내용 안에서도 통합된 형태로의 교육 내용을 개발하거나, 취·창업과 연계된 교육과정 및 내용 개발이 필요할 것으로 보인다.

참 고 문 헌

[1] 정제영(2016). **지능정보사회 대비 미래 교육 정책방향과 과제**. 한국교육학회.

[2] 이상우(2008). **아이를 바꾸는 학습 코칭론**. Family&Relationships.

[3] 나정은(2015). 컴퓨팅적사고(Computational Thinking)교과과정 개발. **한국교양교육학회** 학술대회 자료집, 161-166.

[4] 국가교육과정정보센터, http://ncic.re.kr

[5] 한국인터넷진흥원(2014). **글로벌 소프트웨어 교육 현황 및 교육 도구 동향**.

[6] 남상현 외(2015). 초,중등정보 S/W교육: 미니컴퓨터 기반 과학실험에서 SW교육. **한국컴퓨터교육학회, 19**(1), 51-55.

[7] 교육부 사이트, www.moe.go.kr/

[8] 미래창조과학부 사이트, www.msip.go.kr/

[9] 이태욱, 유인환, 이철현(2001). **ICT교육론**. 형설출판사.

[10] 박지영(2016). **학습자특성에 따른 프로그래밍 교육의 효과**. 석사학위논문, 고려대학교 교육대학원.

[11] 김종훈, 김태훈, 문현국(2011). **생각을 키우는 LOGO프로그래밍**. 서울:학지사.

[12] 이데일리(2015. 10. 8). 컴퓨팅적 사고가 미래 인재들의 핵심 경쟁력. Retrieved from http://www.edaily.co.kr/news/realtime/realtime_NewsRead.asp?newsid=02702726609530952.

[13] Barr V., & Stephenson C.,(2011). Bring Computation Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?. *Magazine ACM Inroads. 2*(1), 48-54.

[14] Chris D., Punya M., & Joke V(2013), "Advancing computational thinking in

21stcentury learning," *International Summit on ICT Education.* 2013.

[15] 미래창조과학부(2015). **주요국 초중고 SW 교육 현황 및 시사점**.

[16] J. M. Wing(2011). *Computational Thinking*. OurCS Workshop, Carnegie Mellon University. 4 March.

[17] 박민우(2014). 국내 프로그래밍 교육 실태와 소프트웨어 산업의 미래, 디지에코 보고서.

[18] 김수환(2009). 교육용 프로그래밍 언어, Online Available http://blog.daum.net/cl_education/15.

[19] 신수범, 구진희(2014). 교육용 프로그래밍 언어의 선택 기준 개발. **컴퓨터교육학회, 17**(4), 13-21.

[20] 이영주(2010). Scratch 프로그래밍 교육이 논리적 사고력에 미치는 영향 비교. **한국컴퓨터교육학회 하계 학술발표 논문**, 14(2), 267-268.

[21] Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: Programming for all. *Communications of the ACM, 52*, 60-67.

[22] 조성환, 송정범, 김성식, 이경화(2008), CPS에 기반한 Scratch EPL이 문제해결력과 프로그래밍 태도에 미치는 효과. **한국정보교육학회, 12**(1), 77-88.

[23] 송정범, 조성환, 이태욱(2008). Scratch 프로그래밍 학습이 학습자의 동기와 문제해결력에 미치는 영향. **한국정보교육학회**, **12**(4), 323-332.

[24] 노희진(2015). 스크래치를 활용한 고등학교 과학 수업에 대한 학생 인식. **한국과학교육학회. 35**(1), 53-64.

[25] 오정철, 이지휜, 김정아, 김종훈 (2012). 스크래치를 활용한 STEAM 기반 교육 프로그램 개발 및 적용-초등학교 6학년 과학교과를 중심으로. **한국컴퓨터학회 논문지, 15**(3), 11-23.

[26] 안형진, 마대성(2013). 문제해결력 증진을 위한 초등학교 Scratch 교육과정 개발. **정보교육학회논문지. 17**(3), 317-327.

[27] 장희선(2013). **Netlogo 프로그래밍 활용학습이 학습동기와 학습능력 향상에 미치는 영향분석**. 석사학위논문, 아주대학교 대학원.

[28] 박효선(2016). **프로젝트기반학습에서 문제해결력, 협력적 자기효능감, 협력적 자기조절, 인지된 성취도 간의 관계**. 석사학위논문, 이화여자대학교 대학원.

[29] 유지은(2014). **스마트교육환경에서 학습자의 미디어 리터러시, 자기조절학습능력, 협력지향성이 초등학생의 인지된 학업성취에 미치는 영향**. 석사학위논문, 한국교원대 대학원.

[30] 안성진(2015). 소프트웨어교육의 방향과 전망 Online Available http://edzine.kedi.re.kr/2015/autumn/article/policy_01.jsp.

[31] 피수영(2016). IT융합교육을 위한 비전공자 코딩교육의 발전방향. **디지털 융합학회, 14**(10), 1-8.

오 미 자

2009 광운대학교 멀티미디어교육공학 (교육학석사)
2015 성균관대학교 컴퓨터교육과(교육학박사)
2011~2013 한국교육학술정보원 연구원
2013~2016 건국대학교 대학교육혁신원 연구원
2016~현재 건국대학교 대학교육혁신원 교수
관심분야: 이러닝, 스마트러닝, OER, MOOC, SW 교육
E-Mail: skyomj@konkuk.ac.kr