https://doi.org/10.32431/kace.2019.22.1.005

비전공자를 위한 교육용 프로그래밍 언어의 비교 연구: 프로그래밍 언어 설계 원칙의 관점으로

김영민† · 이민정††

요 약

SW 중심사회가 도래하면서 사회 모든 분야에서 SW 기반의 문제해결 역량이 강조되고 있다. 대학에서도 비전공자를 위한 SW 기초교육을 의무화하고 프로그래밍 교육을 진행하는 추세이다. 본 연구는 프로그래밍 언어의 설계 원칙 중 간결성과 일반성을 수용하는 문법요소를 도출하고 이를 기반으로 비주얼 프로그래밍 언어(스크래치, 앱인벤터)와 다이어그래밍 언어(랩터, 플로우고리즘)를 비교 분석하였다. 그 결과 간결성과 일반성에 있어 랩터의 표현력과 효율성이 스크래치보다 우세한 것으로 분석되었으며 학습자의 결과물을 통해 이를 확인하였다. 본 연구를 통해 프로그래밍 언어의 특징에 기반한 프로그래밍 기초교육의 설계와 구현에 기여할 것으로 기대한다.

주제어 : 프로그래밍 언어, 비전공자, 프로그래밍 기초교육, 설계 원칙

A Comparative Study of Educational Programming Languages for Non-majors Students: from the Viewpoint of Programming Language Design Principles

Youngmin Kim† · Minjeong Lee††

ABSTRACT

As the SW-centered society has emerged, SW-based problem-solving capabilities is emphasized in all areas of society. It is a trend that universities are obliged to do SW basic education for non-majors students and they are carrying out programming education. This study derives grammatical elements based on conciseness, generality, and efficiency among the design principles of programming language and based on it, compares and analyzes visual programming language and diagramming language. As a result, the efficiency of Raptor is more powerful than Scratch in the simplicity and generality, and the same tendency can be confirmed in the result of the learner 's obtained in programming lesson. We hope that this study will contribute to the design and implementation of programming education based on features of programming language.

Keywords : Programming Language, Non-majors, Programming Education, Design Principles

†정 회 원: 중앙대학교 다빈치교양대학 조교수
††정 회 원: 중앙대학교 다빈치교양대학 조교수(교신저자)
논문접수: 2018년 11월 12일, 심사완료: 2018년 12월 26일, 게재확정: 2018년 12월 27일

1. 서론

4차 산업혁명 시대를 주도할 미래 인재 양성을 위해 SW 교육이 중요하게 자리매김하고 있다[1]. SW는 사회 모든 분야에서 새로운 가치를 창출할 수 있는 혁신의 기반이 되며 산업 간 벽을 허물어 이종 산업을 융합할 수 있는 결합체 역할을 할 수 있다. 사회의 다양한 분야에서 응용 프로그램의 활용 능력뿐만 아니라 SW의 개념과 동작을 이해하고 직접 개발할 수 있는 SW 역량을 강조하는 이유이다[2][3].

대학에서도 전공과 상관없이 모든 학생을 대상으로 한 SW 교과과정을 늘려가고 있다. 기존에 비전공자 대상의 SW 교육이 주로 ICT 활용 중심의 사무용 응용 프로그램의 숙련자 양성을 목표로 했다면 최근에는 SW 중심사회에 대비하여 전공 분야에 대한 이해를 바탕으로 창의적이고 융합적으로 문제를 해결할 수 있는 SW 기반 문제해결 역량의 향상에 초점을 맞추고 SW 프로그래밍 교육을 필수로 포함하는 추세이다[4]. 이에 따라 비전공 대학생들이 겪는 프로그래밍 학습에서의 어려움을 분석하고 효과적인 프로그래밍 학습 방안과 콘텐츠에 관한 연구가 활발히 진행되어 왔다[5][6][7].

한편 프로그래밍 과정은 프로그래밍 언어, 개발환경, 컴퓨팅기기와 같은 개발도구에 따라 초기 진입의 용이함, 학습 기간, 학습 성취도가 달라질 수 있다. 특히 초보자에게는 프로그래밍 언어가 텍스트 기반인지, 비주얼 프로그래밍 기법을 활용하는지, 다이어그래밍 방식의 설계를 허용하는지와 같은 언어적 특징에 따라 프로그래밍 학습 속도와 효능감에 영향을 받게 되며 따라서 학습 효과에도 차이가 나타날 수 있다. 그러나 프로그래밍 기초교육에 관련한 기존 연구는 프로그래밍 교육 효과를 높이기 위한 프로그래밍 교수 학습 모형과 전략 개발에만 초점을 두고 있다[8][9][10][11]. 기초교육용 프로그래밍 언어의 선택을 위해 프로그래밍 언어론 관점에서 언어 간 비교를 진행한 연구는 부족하다.

본 연구에서는 비전공자를 포함한 초보자를 위한 교육용 프로그래밍 언어들을 컴퓨터 과학의 프로그래밍 언어론 관점에서 그 특징과 차이를 고찰함으로써 교육용 프로그래밍 언어 자체에 내재된 특징을 분석하고자 하였다. 교육용 프로그래밍 언어의 특징을 분석하기 위해 컴퓨터 과학에서 정의하는 프로그래밍 언어의 설계 원칙 중 간결성과 일반성을 준용하는 문법적 요소들을 도출하였다. 비전공자 대상의 프로그래밍 교육에 주로 사용되는 교육용 프로그래밍 언어들에 대해 앞서 도출한 문법적 요소의 수용 범위를 분석하고 그에 따른 결과물의 특징을 비교하였다. 본 연구의 결과를 바탕으로 향후 비전공자의 SW 기초교육에 활용할 프로그래밍 언어를 선택하거나 설계할 때 선정 기준을 수립할 수 있을 것이며 이를 기반으로 비전공자 대상 프로그래밍 기초교육의 설계와 효과적 실행에 기여할 것으로 기대한다.

2. 이론적 배경

2.1 SW 기초교육용 프로그래밍 언어

1) 다이어그래밍 언어

프로그래밍은 문제 해결에 필요한 알고리즘을 프로그래밍 언어의 문법에 따라 소스코드를 작성하는 코딩과정이 수반되므로 프로그래밍 언어의 문법을 숙지해야 하고 제대로 작성되었는지 계속 점검해야 한다. 그러나 순서도, NS 차트, UML과 같은 다이어그래밍 언어는 다이어그램 작성을 위해서 간단한 기호들만 익히면 되기 때문에 고급 프로그래밍 언어의 문법을 학습하는 것에 비해 시간과 노력이 적게 들고 코딩 과정이 없으므로 문제와 해결과정 자체에 집중할 수 있다[12].

최근에는 순서도를 컴퓨터로 작성할 수 있는 랩터(raptor), 플로우고리즘(flowgorithm) 등과 같은 다이어그래밍 언어가 등장하였다[13][14]. 랩터와 플로우고리즘은 다이어그램 자체를 해석하여 실행시켜주기 때문에 다이어그램을 이용해 문제를 해결한 후 실행을 통한 검증이 가능하다. 이로써 프로그래밍 언어를 이용한 코딩 과정을 생략하고 알고리즘에 집중한 프로그래밍 학습과 문제해결 학습이 가능하다.

2) 교육용 비주얼 프로그래밍 언어

초보자를 대상으로 하는 프로그래밍 교육은 실제적인 문제해결에 주력하기 보다는 학습을 통해 논리적 사고력, 문제해결력 등의 고등 사고력 신장이

목적이라는 점에서 고급 프로그래밍 언어를 통한 교육의 한계점을 극복하기 위해 교육용 프로그래밍 언어가 등장하였다. 교육용 프로그래밍 언어는 문법 암기에서 벗어나, 자신의 생각을 블록이나 간단한 스크립트로 표현하여 프로그래밍할 수 있게 하는 언어이다. 국내에서는 스크래치(Scratch), 엔트리(Entry), 두리틀(DoLittle), 앱인벤터(App Inventor), 로고(Logo) 등을 활용하고 있으며 다양한 교과와 연계한 융합 교육에도 적용하고 있다[9][15].

3) 텍스트 기반 고급 프로그래밍 언어

일반적으로 프로그래밍 교육에 가장 많이 활용되는 언어는 C, Java, Python 등 고급 프로그래밍 언어들이다. 중·고등학교 정보·컴퓨터 과목에서도 C 언어를 기반으로 프로그래밍 교육이 이루어지고 있다. 그러나 고급 프로그래밍 언어 수업 설계를 살펴보면 문법 위주의 내용이 대부분을 차지하고 있어 실질적 알고리즘 교육이 되기 어렵다. 또한 디버깅에 많은 시간을 투자하고 있어 초보자들에게는 실제 문제를 해결하는 데 프로그래밍 과정을 적용하기도 전에 프로그래밍 학습의 흥미를 잃게 하는 요인이 되기도 한다[16]. Pietra와 Costa(2013)는 초보 학습자들이 프로그래밍 학습 과정에서 포인터와 참조, 매개변수, 구조체, 디버깅, 라이브러리 활용에서 특히 어려움을 느끼고 있음을 밝혔다[17].

2.2 교육용 프로그래밍 언어별 학습 효과

SW 교육의 확산되면서 컴퓨팅 사고 학습의 주요 문제점으로 체계적이지 못한 교수법, 학생들과의 상호 작용 부족, 수업에 대한 관심 부족이 제기되었다[18]. Bennedsen 등과 Kazimoglu 등은 이러한 이유로 프로그래밍 학습은 비전공자에게 좌절감을 느끼도록 하는 활동으로 인식되기도 한다고 주장하였다[19][20]. 초기에는 프로그래밍 기초교육 과정에 주로 텍스트 프로그래밍 언어가 사용되었는데, Crews 등은 초보자를 대상으로 하는 프로그래밍 기초교육에서 순서도를 사용하는 것이 텍스트 프로그래밍 언어를 사용하는 것보다 효율적이라는 것을 보여주는 결과를 발표하였다[13].

이후 비주얼 프로그래밍 도구들이 등장하기 시작하면서 초등학생 대상의 프로그래밍 교육에서 텍스트 프로그래밍 언어와 비주얼 프로그래밍 언어의 효과를 비교한 연구는 학습자의 동기 부여를 높이기 위해 비주얼 프로그래밍 언어를 사용하는 것이 더 적합하다는 결과를 보여준다[21]. Noone 등은 비주얼 프로그래밍 언어와 텍스트 프로그래밍 언어 사이에 첫 프로그래밍 언어로 무엇을 선택하는 것이 좋을 것인가에 대하여 스냅(snap)과 자바(java)를 비교한 연구를 통해 처음 프로그래밍 언어를 배우는 학생들은 스냅에 비해 자바를 배우는 것이 어렵다는 결과를 도출하였다[22].

Giordano는 프로그래밍 언어의 문법적 효능을 비교한 연구에서 학습자의 효능감을 평가하는 방법으로 비주얼 프로그래밍 언어, 텍스트 프로그래밍 언어, 순서도를 사용하여 프로그램을 작성하도록 했을 때 소요되는 시간과 발생하는 에러의 수를 비교하였는데 비주얼 프로그래밍과 순서도 접근 방법 모두 시간적인 측면과 오류의 통계적인 측면에서 텍스트 프로그래밍과 마찬가지로 알고리즘 설계에 적합함을 밝혔다[14].

3. 교육용 프로그래밍 언어의 비교 고찰

비전공자를 대상으로 진행하는 프로그래밍 기초교육에서 비전공자의 프로그래밍 경험 부족과 적응 속도를 고려하여 비주얼 프로그래밍과 다이어그램 언어를 많이 사용하고 있다. 비주얼 프로그래밍과 다이어그램 언어는 텍스트 언어와 달리 블록과 도형을 결합하는 형식으로 프로그래밍을 지원하고 있어 높은 효율성을 갖기 때문이다.

프로그래밍 언어는 수업 이외의 시간에도 사용 가능하도록 공개 라이선스를 지원하고 이미 많은 사용자가 확보되어 원격으로 도움말을 공유할 수 있는 커뮤니티가 형성된 것이 적합하다. 이와 같은 비주얼 프로그래밍 언어로는 스크래치와 앱인벤터, 다이어그래밍 언어로는 랩터와 플로우고리즘(flowgorithm)이 있다. 본 연구에서는 프로그래밍 기초교육을 위한 프로그래밍 언어를 프로그래밍 언어의 설계원칙 관점에서 비교한다[23].

3.1 프로그래밍 언어의 설계원칙을 수용하는 문법요소 도출

프로그래밍 언어는 사용 목적과 필요성에 따라 설계원칙이 좌우된다. 설계원칙에는 효율성(efficiency), 일반성(generality), 획일성(uniformity), 직교성(orthogonality) 등의 기본 원칙과 간결성(simplicity), 표현력(expressiveness), 정확성(preciseness) 등의 기타 원칙이 존재한다[23]. <표 1>는 프로그래밍 언어 설계원칙이 요구하는 사항을 간단히 보여준다.

<표 1> 프로그래밍 언어 설계 원칙

설계 원칙 구분	내용
효율성 (efficiency)	얼마나 쉽고, 빠르게 프로그래밍 작성이 가능한가?
일반성 (generality)	밀접하게 관련 있는 여러 개념을 일반적인 하나의 개념으로 통합하는가?
획일성 (uniformity)	유사한 것들은 유사하게 보이며, 유사한 의미를 지니는가?
직교성 (orthogonality)	같은 요소가 문맥에 따라 같게 동작하는가?
간결성 (simplicity)	언어가 복잡하지 않고 간결한가?
표현력 (expressiveness)	복잡한 구조를 쉽고 표현할 수 있는가?
정확성 (preciseness)	문법이 잘 정의되어 있는가?
보안성 (security)	프로그래밍 오류를 줄이고, 오류 발견에 쉬운가?
기계 독립성 (machine independence)	언어가 기계에 독립적인가?
확장성 (extensibility)	새로운 기능을 추가할 수 있는가?

본 연구에서는 프로그래밍 기초교육용 프로그래밍 언어를 프로그래밍 언어 설계원칙의 일반성, 간결성 측면에서 비교한다. 비교를 진행하기에 앞서 프로그래밍 언어의 문법적 측면에서 7가지 요소를 도출하였다. 프로그래밍 언어 간 차이를 보이지 않는 부분은 배제하였다. 예를 들어, 조건문의 경우 모든 언어가 지원하는 형태가 거의 유사하여 의미가 없다. 또한, 함수와 배열은 문법요소는 본 교과 일정에 포함되지 않아, 문법요소 도출 시 배제한다. <표 2>에서 도출된 문법요소의 구체적인 내용 및 문법요소와 설계 원칙 사이의 연관성을 보여준다.

<표 2> 문법요소와 설계 원칙 사이의 연관성

(+:양의 관계, -:음의 관계, X:관계없음)

문법요소	내용	간결성	일반성	효율성
변수 선언과 초기화	변수 선언과 초기화를 구분하는가?	+	X	+
변수 선언 시 유형의 구분	변수 선언 시 유형을 구분하는가?	+	-	+
변수의 값 할당	변수에 값 할당을 위한 다양한 방법을 제시하는가?	+	X	+
반복문	반복문을 위한 다양한 방법을 제시하는가?	+	-	+
연산자 우선순위(결합)	연산자 우선순위를 제시하는가?	-	+	-
더하기 연산자 데이터 유형	같은 개념으로 더하기 연산자를 문자열 데이터에 적용할 수 있는가?	-	+	-
관계 연산자 데이터 유형	같은 개념으로 관계 연산자를 문자열 데이터에 적용할 수 있는가?	-	+	-

변수의 선언과 초기화를 구분하는 언어는 프로그래밍 작성 시 선언을 위한 절차와 초기화를 위한 절차로 나누어 작성되므로 프로그래밍 언어의 구조를 간결하게 한다. 변수 선언 시 데이터 유형을 구분하면 변수의 유형과 일치하는 데이터만을 저장할 수 있어 일반성은 낮아지게 된다. 하지만, 단일 데이터 유형의 처리 시 프로그래밍 언어는 간결한 명령을 제공할 수 있다. 변수에 값 할당, 반복문 작성 시 다양한 방법을 제공할 수 있는 경우 간결성이 높은 언어이다. 특정 상황에 맞는 방법을 사용하여 간결한 명령을 통해 프로그램 작성이 가능하기 때문이다. 문자열 데이터에 더하기 연산자와 관계 연산자를 사용할 수 있는 경우 그렇지 않은 언어에 비해 일반성이 높다고 할 수 있다. 하지만 직관적이고 간결하게 문자열의 데이터를 처리하지 못하므로 간결성을 저해하는 요소로 작용한다.

<표 2>에 나타난 것처럼 본 연구에서 도출한 문법요소와 프로그래밍 언어 설계 원칙과의 연관성은 프로그래밍 언어론에서 제시하는 설계 원칙 사이의 연관성을 모두 만족하고 있다. 프로그래밍 언어 설계 시 일반성이 높으면, 간결성을 해치는 요소가 된다[21]. 프로그래밍 언어 설계 시 간결성이 높아지면, 효율성도 함께 향상된다.

<표 3> 프로그래밍 언어에서 지원하는 문법요소

(O:그렇다, X:그렇지 않다)

항목	문법요소	내용	비주얼 프로그래밍		다이어그래밍	
			스크래치	앱인벤터	랩터	플로우고리즘
1	변수 선언과 초기화	변수 선언과 초기화 구분하는가?	O	X	X	O
2	변수 선언 시 타입	변수 선언 시 타입을 구분하는가?	X	X	X	O
3	변수의 값 할당	변수에 값 할당을 위한 다양한 방법을 제시하는가?	O	X	X	X
4	반복문	반복문을 위한 다양한 방법을 제시하는가?	O	O	X	O
5	연산자 우선순위 (결합)	연산자 우선순위를 제시하고 있는가?	X	X	O	O
6	더하기 연산자 데이터 유형	같은 개념으로 더하기 연산자를 문자열 데이터에 적용할 수 있는가?	X	X	O	X
7	관계 연산자 데이터 유형	같은 개념으로 관계 연산자를 문자열 데이터에 적용할 수 있는가?	O	X	O	X

3.2 프로그래밍 언어 설계 원칙 관점에서의 교육용 프로그래밍 언어의 비교

프로그래밍 기초교육에 활용되는 프로그래밍 언어가 앞서 도출된 문법요소를 지원하는지 살펴본다. <표 3>는 스크래치, 앱인벤터, 랩터, 플로우고리즘에서 어떤 문법요소를 지원하고 있는지에 대한 결과를 보여준다. 이 결과를 통해 프로그래밍 언어 간 문법적 특징에 차이가 있음을 확인할 수 있다.

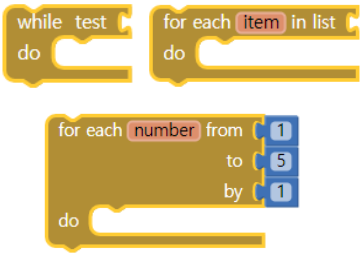
스크래치와 플로우고리즘은 변수 선언을 위한 별도의 절차를 요구한다. 반면, 앱인벤터와 랩터는 선언과 초기화가 동시에 이루어진다. 스크래치와 랩터는 변수에 값을 할당하기 위해 데이터의 유형을 고려하지 않는다.

랩터를 제외한 언어는 반복을 위한 여러 형태의 반복문을 제공한다. [그림 1]은 각 프로그래밍 언어에서 제공하는 반복문을 보여 준다.

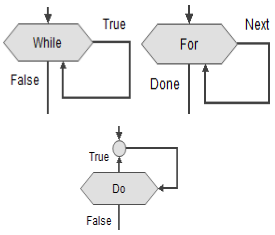
스크래치와 앱인벤터는 블록 결합 방식의 프로그래밍 언어로 연산자 우선순위가 존재하지 않는다. 연산식을 구성하는 블록 단위로 연산이 수행되기 때문이다.

랩터는 문자열 데이터에 더하기 연산을 적용하여 문자열 결합이 가능하다. 스크래치와 앱인벤터의 경우 문자열 결합을 위한 별도의 블록을 제공하며, 플로우고리즘은 문자열 결합 방식을 제공하지 않는다.

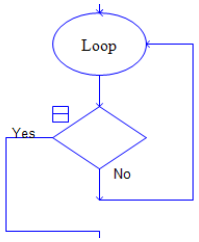
(a) 스크래치에서 제공하는 반복문



(b) 앱인벤터에서 제공하는 반복문



(c) 플로우고리즘에서 제공하는 반복문



(d) 랩터에서 제공하는 반복문

[그림 1] 프로그래밍 언어별 반복문의 유형

<표 4>에서는 간결성, 일반성에 해당하는 문법요소를 기반으로 스크래치, 앱인벤터, 랩터, 플로우고리즘을 비교한다. 비교 결과로부터 각 언어의 문법적 차이가 학습자의 프로그래밍에 미치는 영향을 분석한다. 랩터는 <표 3>의 결과로부터 프로그래밍

언어 설계 원칙의 일반성과 간결성 측면에서 높은 성향을 보일 것으로 예측된다. 이러한 이유로 랩터를 기준이 되는 언어로 정한다.

플러스(+)는 문법요소에 의해 연결된 설계 원칙이 랩터보다 우세함을 나타내고, 마이너스(-)는 문법요소에 의해 연결된 설계 원칙이 랩터보다 열세함을 나타낸다. 프로그래밍 언어 설계 관점에서 일반성이 높은 언어는 간결성을 보장하지 못하기 때문에 일반성이 높아지는 경우 <표 2>에 보인 것처럼 간결성을 낮추는 요소로 작용한다. 하지만 일반성이 낮다고 반드시 간결성이 향상되는 것은 아니다. <표 3>의 연산자 우선순위를 제공하지 않는 스크래치와 앱인벤터의 경우 랩터와 플로우고리즘보다 일반성이 상대적으로 낮다고 해서 간결성이 높은 것은 아니다. 스크래치와 앱인벤터는 연산자마다 별도의 블록을 제공하고 있으며, 이러한 블록은 연산식 작성 시 괄호의 역할을 하여 다른 언어와 같이 연산자 우선순위를 제공할 필요가 없다. 연산식을 나타내는 중첩된 블록은 중첩의 가장 안쪽부터 블록 단위로 연산이 수행된다. 따라서 개발자에게 작성하는 연산식의 모든 우선순위를 고려하여 블록을 구성해야 하는 책임이 있다.

스크래치와 앱인벤터의 경우 [그림 1]에 보인 것처럼 다양한 반복문을 제공하여 특정한 반복에서 랩터보다 간결한 구조를 보인다. 하지만, 후 조건 반복을 위한 별도의 반복문을 제공하지 않아 후 조건 반복 절차 작성 시 랩터보다 복잡하다. 랩터의 경우 단일 반복문만으로 후 조건 반복을 간결하게 표현한다.

[그림 2]는 스크래치, 앱인벤터, 랩터, 플로우고리즘의 후 조건 반복 반복문의 구조를 보여준다. 스크래치와 앱인벤터의 경우 반복해야 할 절차를 반복 시작 전에 한 번 더 작성해야 한다. 반복해야 할 절차가 많아질수록 코드의 양도 함께 증가하게 된다. 결과적으로, 스크래치와 앱인벤터는 후 조건 반복을 위한 개별적인 반복문을 제공하고 있지 않아 반복문 전체로 고려했을 때, 일반성과 간결성 모두 낮아지게 된다. 그러므로 다양하게 제공되는 구성자가 빠짐없이 모든 문법적 요소를 표현할 수 있어야 간결성의 향상을 가져올 수 있다. 구성자의 수가 많아지는 것이 반드시 간결성 향상으로 연결되는 것은 아니다.

비전공자를 대상으로 하는 프로그래밍 기초교육은 프로그래밍이 쉽고 직관적인 비주얼 프로그래밍 또는 다이어그래밍 언어를 사용한다. 프로그래밍이 쉬운 언어는 프로그래밍 언어 설계 원칙의 관점에

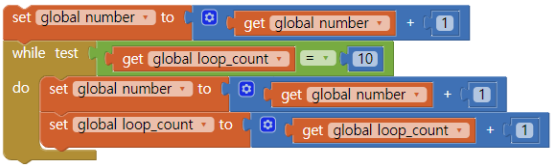
<표 4> 간결성, 일반성 설계 원칙에 따른 프로그래밍 언어의 비교

(-: 낮음, +: 높음)

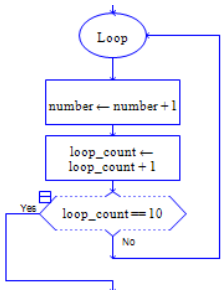
구분	내용	문법요소	비주얼 프로그래밍		다이어그래밍	
			스크래치	앱인벤터	랩터	플로우고리즘
간결성 (simplicity)	가능하고 명료하고, 단순한 표현으로 원하는 알고리즘을 기술할 수 있는가?	변수 선언과 초기화	−	·	·	−
		변수 선언 시 타입	·	·	·	−
		변수의 값 할당	+	·	·	·
		반복문	−	−	·	−
		연산자 우선순위(결합)	−	−	·	·
		더하기 연산자 데이터 유형	·	−	·	−
		관계 연산자 데이터 유형	·	−	·	−
일반성 (generality)	밀접하게 관련 있는 여러 개념을 일반적인 하나의 개념으로 통합하고 있는가?	변수 선언 시 타입 구분	·	·	·	+
		반복문	−	−	·	−
		연산자 우선순위(결합)	−	−	·	·
		더하기 연산자 데이터 유형	−	−	·	−
		관계 연산자 데이터 유형	·	−	·	−

서 높은 효율성을 갖는 언어이다. 프로그래밍 언어 설계 원칙에서 간결성의 향상은 프로그래밍의 효율성 향상과 연결된다.

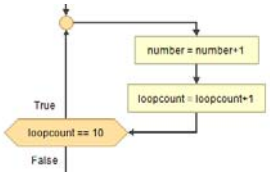
(a) 스크래치



(b)앱인벤터



(c)랩터



(d)플로우고리즘

[그림 2] 프로그래밍 언어별 후조건 반복문

4. 연구 방법

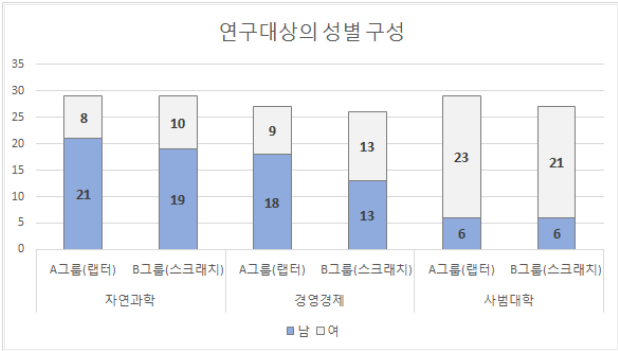
4.1 연구 대상의 선정

C 대학은 비이공계신입생 전체를 대상으로 컴퓨팅 사고력 함양을 위한 SW 기초교육을 의무적으로 시행하고 있다. 이 중 이과계열(자연과학대학), 문과계열(경영경제대학, 사범대학) 신입생으로 구성된 분반을 <표 5>와와 같이 각각 2개의 그룹으로 나누어 서로 다른 프로그래밍 언어를 적용한 프로그래밍 기초교육을 진행하였다.

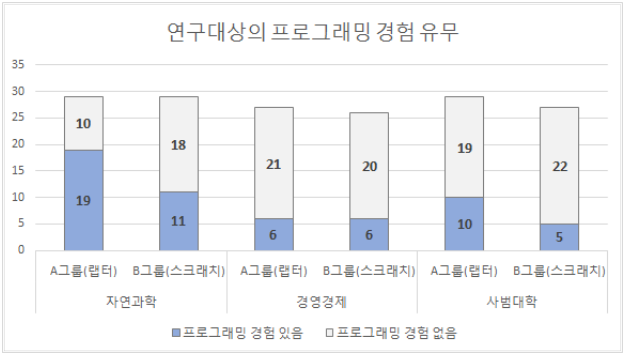
<표 5> 연구대상의 구성

	A그룹 (랩터)	B그룹 (스크래치)
자연과학대학	29명	29명
경영경제대학	27명	26명
사범대학	29명	27명
합계	85명	82명

연구대상의 그룹별 성별 분포를 함께 조사하였다. [그림 3]에 표시된 바와 같이 계열 특성 상 자연과학대는 남학생의 비율이 A그룹에서 72%, B그룹에서 65%로 상대적으로 높았으며 사범대학은 A그룹에서 79%, B그룹에서 78%로 상대적으로 높았다. 학과별 남녀 성별 구성은 다르게 나타났으나 그룹 간에 성별 비율은 유사하다는 것을 알 수 있다.



[그림 3] 연구대상의 성별 구성



[그림 4] 연구대상의 프로그래밍 경험

[그림 4]는 각 학과별로 학습자의 프로그래밍 경험을 조사하여 나타낸 것이다. 연구대상의 각 그룹 내에 프로그래밍을 이미 경험한 학생들은 컴퓨터의 동작을 어느 정도 이해하고 있을 가능성이 높으며 경험이 없는 학생들에 비해 컴퓨팅 수업에 두려움

이 덜할 수 있다.

Kwon D. 등은 초보 개발자를 대상으로 프로그래밍 학습 절차에 대한 연구 과정에서 학습자에게 프로그래밍의 목적, 필요성, 흥미에 관한 설문을 통해 프로그래밍 인식의 변화를 조사한 바 있다[24]. 본 연구에서는 Kwon D.의 연구에 활용된 프로그래밍에 대한 흥미를 조사하는 문항을 인용하여 문항1과 문항2로 구성하였다. 또한 이우숙 등은 정보가 많아지고 복잡해지는 현대 사회에서 직면할 수 있는 문제 상황을 탐색하고 대처할 수 있는 문제해결 역량을 측정하기 위한 설문 및 분석도구를 개발하였는데 본 연구에서는 문제해결을 위해 문제해석과 해법탐구에 대한 학습자의 태도를 확인하는 문항 5개를 선정하였다[25]. 즉, 본 연구에서는 실험 그룹에 속한 학습자의 프로그래밍에 관한 인식과 문제해결에 대한 태도를 비교하기 위해 사전설문을 위한 문항을 <표 6>과 같이 구성하였다.

<표 6> 프로그래밍 효능감 및 문제해결상황에 대한 태도 조사를 위한 설문문항 [24][25]

항목	설문문항
1	나는 프로그래밍 언어가 무엇인지 알고 있다.
2	나는 코딩을 배워서 컴퓨터 프로그램을 만들어 보고 싶다.
3	나는 문제의 본질을 이해하기 위해 시간을 충분히 쓰는 편이다.
4	나는 복잡한 문제의 해결방안을 찾는 것을 즐기는 편이다.
5	나는 어려운 문제나 과제를 더 잘 이해하기 위해 작은 단위로 나누어서 살펴본다.
6	나는 새로운 아이디어를 만드는데 도움이 되는 정보 조사에 시간을 들인다.
7	나는 문제를 해결하거나 새로운 아이디어를 만들기 위해 정보의 다양한 소스를 고려한다.

4.2 프로그래밍 기초교육 내용의 설계

프로그래밍 언어 설계 원칙 관점에서 프로그래밍 언어의 차이를 살펴보기 위해 비주얼 프로그래밍 언어인 스크래치와 순서도를 표현하는 다어그래밍 언어인 랩터를 적용한 7주간의 주차별 학습 과정을 <표 7>과 같이 설계하였다. <표 4>의 결과로부터 스크래치와 랩터는 다른 두 언어보다 간결성과 일반성 측면에서 높음을 확인할 수 있다. 이러한 결과는 학습자의 생각을 절차로 표현할 때 다른 두 언어보다 유연함을 나타낸다. 주차별 학습 과정 설계 시 사용할 프로그래밍 언어로 스크래치와 랩터를 선택한다.

<표 7> 프로그래밍 기초교육의 주차별 학습 내용

주차	학습목표	학습내용 및 활동
1	SW와 컴퓨터	SW와 컴퓨터의 동작 원리 컴퓨팅 사고와 프로그래밍
2	프로그래밍 언어의 활용	Raptor 또는 스크래치 사용법 익히기
3	정보의 표현	변수의 생성과 화면 출력 - 두 데이터 교환하기 - 소용돌이 사각형 그리기
4	연산자	연산식과 조건식 만들기 - 대입, 산술, 관계, 논리연산자 - 연산자 우선순위
5	조건에 의한 분기	조건문 수립 - 짝수와 홀수 판별하기 - 가위바위보 게임 승/패 조건 수립
6	패턴 도출과 반복	반복되는 패턴의 도출 - (1-10)까지 합 구하기 - (1-100) 사이의 짝수 순차 저장 - 가위바위보 게임의 반복 패턴 탐색
7	조건에 의한 반복	반복패턴의 종료 조건 수립 및 적용 - (1-100)사이의 짝수와 홀수 개수 세기 - 가위바위보 게임 반복종료 조건 수립

<표 7>에서 설계한 내용에 따라 연구대상의 A그룹에는 랩터를 활용한 프로그래밍 교육을 실시하고, B그룹에는 스크래치를 활용한 프로그래밍 교육을 실시하였다. 3주차에 학습하는 '다각형 그리기' 프로젝트는 프로그램 내 정보를 처리하기 위한 변수의 개념과 활용을 다룬다. 4주차부터 5주차에서는 산술 연산식과 비교 및 논리 연산식과 이를 이용한 조건식을 구성하는 방법을 학습하였다. 특히 단순 조건식에서 '~이고', '또는', '아니다'와 같은 관계 연산자를 활용하여 좀 더 복잡하고 풍부한 연산식을 구성할 수 있게 된다. 6주차와 7주차에는 앞에서 다룬 조건식과 반복하고자 하는 패턴을 도출하여 조건에 의한 반복문을 구현하는 것을 목표로 하였다. 이를 위해 1과 100 사이의 짝수 혹은 홀수의 개수를 세는 예제와 가위바위보 게임 예제를 통해 특정 조건에 의해 일련의 동작을 반복하는 개념을 익히도록 하였다. 7주간의 학습을 마친 후

A그룹과 B그룹의 학생들에게 공통적으로 [그림 5]의 '가위바위보 하나빼기 게임' 문제를 해결하도록 테스트하였다.

테스트 문제. "가위바위보 하나빼기 게임"

① 사용자와 컴퓨터로부터 각각 2개의 가위, 바위, 보를 입력받은 후, 둘 중의 하나를 선택하여 가위, 바위, 보를 진행한다.
② 같은 손모양을 내면 컴퓨터와 사용자 모두에게 새로운 손모양을 입력받는다.
③ 심판은 사용자의 남은 손과 컴퓨터의 남은 손으로 승부를 결정한다.
④ 사용자가 이긴 경우에 'USER WIN'을 출력하고 컴퓨터가 이긴 경우에 'COMPUTER WIN'을 출력한다.
⑤ 사용자 혹은 컴퓨터가 5승을 달성하면 게임을 종료한다. 그렇지 않다면, 다시 컴퓨터와 사용자에게 가위, 바위, 보를 입력받아 다음 게임을 진행한다.
⑥ 단, 가위, 바위, 보는 프로그램 내에서 1, 2, 3으로 디지털화한다.

[그림 5] 학습자 테스트용 프로그래밍 문제

'가위바위보 하나빼기 게임'을 6개의 문항으로 구분하였으며, <표 8>에 나타난 바와 같이 각 문항은 본 연구에서 도출한 프로그래밍 언어의 설계 원칙에 따른 문법요소와 연결된다.

<표 8> 문법요소 및 테스트 문항별 평가 기준

문법요소	관련문항	평가 기준
변수의 활용	①⑥	필요한 변수를 적절히 선언하였는가? 컴퓨터와 사용자가 낸 손의 모양을 적정한 변수로 정의하여 저장하는가? 손모양을 숫자로 변환하고 입력과 승패 결정에 활용하는가?
조건문 (연산자의 복합)	②③④	승패 조건을 정확히 수립했는가? 무승부 조건을 수립하여 반복하는가?
반복문	⑤	무승부 조건을 수립하여 반복하는가? 5승이 날 때까지 게임을 반복하는가?
입출력	①②④	사용자 입력을 받았는가 ? 승부결과를 정확히 출력하는가?

두 집단의 학생들에게 공통적으로 '가위바위보 하나빼기 게임' 문제를 해결한 결과에 대해 A그룹(랩터)과 B그룹(스크래치)의 평균과 표준편차를 구하여 성취도를 비교한다. 그룹 간 성적의 분포에 따른 t-검정($p<0.05$)을 수행하여 유의미한 수준에서 차이가 있는지 확인한다. 만약, 분석 결과 평균값이 서로 유사한 수준이라면 스크래치와 랩터 모두 학습자의 프로그래밍 역량을 유사한 수준에서 배양한다고 볼 수 있다. 여기에 덧붙여 프로그래밍 언어의 설계 원칙에 따라 도출한 프로그래밍 언어의 문법요소 별로 학습자 A그룹과 B그룹의 성취도에 차이가 있는지 살펴본다. 만약 전체 평균은 유사하나 항목별 성취도에서 차이가 있다면 이로부터 대상이 되었던 프로그래밍 언어의 고유한 특징을 도출한다.

5. 적용 결과

5.1 연구대상의 비교

본격적인 학습에 앞서 실험 그룹에 속한 학습자의 프로그래밍에 관한 인식과 문제해결에 대한 태도를 비교하기 위해 설문조사를 진행하였다.

<표 6>의 설문 항목에 대해 5점 척도로 응답을 받은 결과를 독립표본 t-검정으로 분석하였을 때 <표 9>에 나타난 바와 같이 실험에 참여한 두 그룹이 유의수준 0.05에서 프로그래밍에 대한 관심도와 문제해결을 위한 솔루션의 탐색 성향이 유동일한 집단임을 확인하였다. 따라서 두 그룹에 속한 비전공 학생들에게 익숙하지 않은 프로그래밍 과정에 대해서 유사한 학습 태도를 가진 것으로 추정할 수 있다. 본 설문도구의 신뢰도를 조사하기 위해 각 그룹의 설문결과에 Cronbach alpha 방식을 적용한 결과 A그룹은 0.64, B그룹은 0.65로 거의 근사한 신뢰도를 보였다.

<표 9> 그룹별 프로그래밍 효능감 및 문제해결상황에 대한 태도 조사 결과

설문 항목	A그룹 평균	B그룹 평균	t-검정 p=0.05
1	2.31	2.21	0.62
2	3.25	3.40	0.37
3	3.76	3.84	0.56
4	3.39	3.66	0.07
5	3.40	3.60	0.17
6	3.55	3.73	0.20
7	3.86	3.98	0.35

5.2 프로그래밍 기초교육의 적용

프로그래밍 기초교육을 받는 동안 학생들은 동일한 예제를 각각 랩터와 스크래치로 구현하였다. 그룹별 학생들의 구현 결과물을 변수의 활용, 연산자의 사용, 반복문 구성 측면에서 비교하였다. 그 결과 본 연구에서 프로그래밍 언어의 설계 원칙을 투영한 문법적 요소들을 도출하고 각 문법요소 별로 프로그래밍 언어들 간에 상대적 우위를 분석한 <표 4>와 유사한 결과와 일치함을 확인하였다. 각 주차별 학습내용을 구현한 학습자의 대표적 사례를 들어 좀 더 자세히 살펴보겠다.

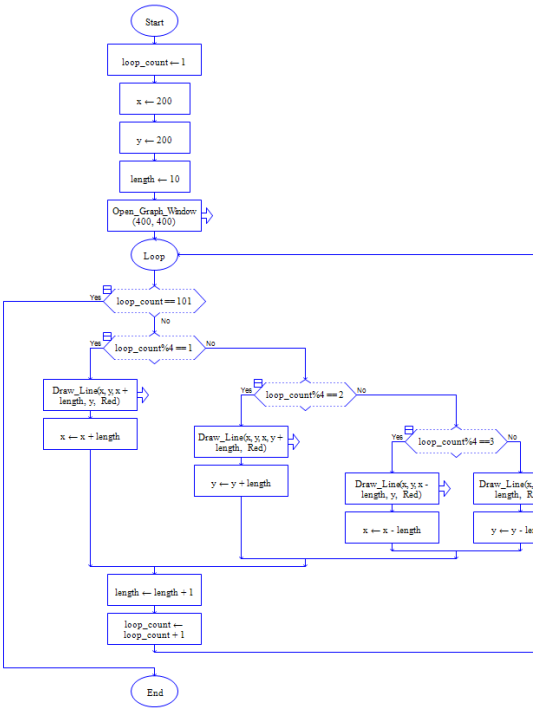
5.2.1 변수의 활용

두 프로그래밍 언어 모두가 변수를 사용하기 전에 데이터 유형을 미리 확정하지 않지만, 스크래치는 [그림 6]에서 보듯이 사용할 변수의 이름과 적용범위('지역' 혹은 '전역')를 미리 선언해야 한다. 랩터는 변수를 사용하기 전에 별도의 선언 과정이 필요하지 않아 바로 사용되고 있음을 볼 수 있다. 또한, 스크래치에서는 변수의 값을 변경하기 위해 '~로 정하기', '~만큼 바꾸기', '~만큼 움직이기', '~도 돌기' 등 다양한 블록을 활용하고 있음을 확인할 수 있다.

랩터는 변수의 값을 바꾸는 용도로 '←' 기호만 제공하므로 단순하다. 이는 여러 경우를 쉽게 표현할 수 있는 다양성이 부족할 수 있겠으나 한편 학습자로 하여금 하나의 방법으로 여러 경우를 나타내도록 유도하여 수학적 사고력을 배양할 할 수 있다.

5.2.2 조건문 - 연산자의 복합 활용

[그림 7]는 학습자에게 1부터 100까지의 수 중 짝수와 홀수의 개수를 카운트하는 예제를 구현하게 한 결과이다. 이중에서도 짝수와 홀수를 구분하기 위해 나머지(%) 연산자를 활용하는 부분을 살펴보겠다. 랩터는 나머지 연산을 위해 '%' 연산자를 제공한다. number 변수값이 홀수인지 구분하기 위해 'number % 2 = 1'을 조건식을 수립했을 때, 랩터에서는 'number % 2'의 수식이 먼저 계산되고 이후 1과 등가인지 확인하는 순서로 연산자 우선순위에 따라 수행된다.



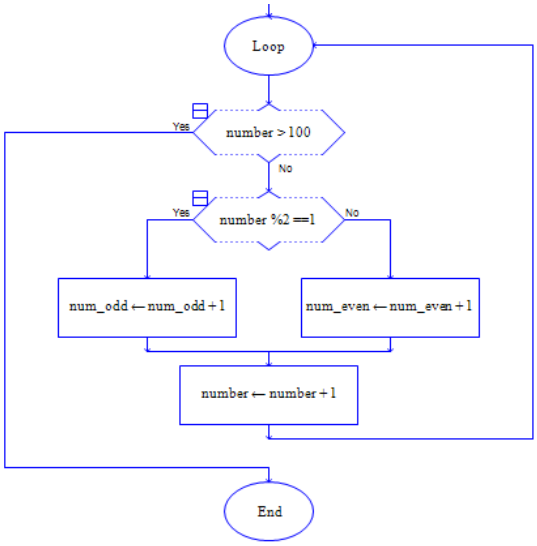
(a) 랩터

(b) 스크래치

[그림 6] 학습적용 사례 - 정보의 표현과 활용

반면 스크래치는 모든 연산 블록을 2개의 입력값만 허용하는 이항연산자 형태로만 제공하므로 연

산의 우선순위를 개발자가 정해야 한다.



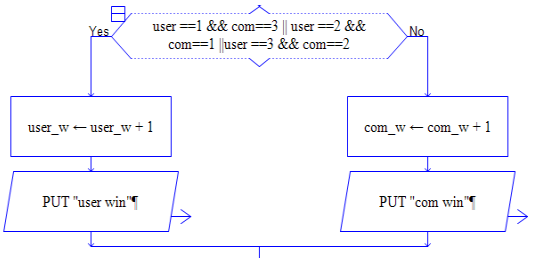
(a) 랩터

(b) 스크래치

[그림 7] 학습적용 사례 - 연산자의 구조

[그림 7]의 (b)에서 'number 나누기 2의 나머지' 연산 블록을 '~가 아니다' 블록 위에 얹어서 가장 상위의 블록부터 처리되도록 한 것에서도 알 수 있다. 스크래치의 이러한 방침은 학습자가 연산자 우선순위에 대한 이해 없이도 연산 기능을 활용할 수 있다는 관점에서 강점이라고 볼 수 있다. 그러나 대학생의 경우 기존에 연산자 우선순위에 따라 수립할 수 있었던 복합 연산식을 이항 연산 구조로 변형해야 하는 것에서 오히려 부담을 느끼는 것을 확인할 수 있었다.

복합 관계식의 사례를 들어보겠다. 5주차에 '가위바위보 게임의 승패 조건을 고려하여 수립'하는 활동을 하였는데 랩터는 비교연산자 '=='과 관계연산자 '&&'와 '||'를 활용하여 [그림 8]의 (a)과 보이는 바와 같이 조건문을 앞서 언급한 바와 같이 스크래치는 각 연산자가 이항연산자 블록으로 표기되므로 각 블록을 찾아 비교 순서에 따라 쌓아주어야 한다. [그림 8]의(b)의 조건식을 수립하기 위해 비교조건 블록, 관계연산자 블록을 들여와 계속 쌓아주어야 했는데 조건식이 너무 길게 표현되어 가독성이 떨어지는 것을 알 수 있다. [그림 8]의 (c)는 (b)에서 한 줄로 표현한 복합 조건식을 단계별 조건으로 재구성하여 보다 직관적으로 표현한 결과이다.



(a) 랩터

(b) 스크래치 - 복합관계식(1)

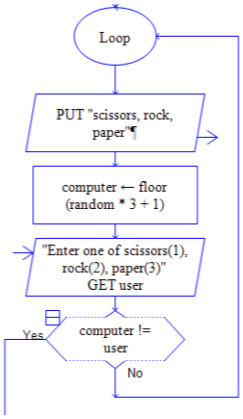
(c) 스크래치 - 복합관계식(2)

[그림 8] 학습적용 사례 - 복합조건식의 수립

5.2.3 반복문의 구성

반복문을 구성하기 위해서 의도한 동작 중 어느 구간이 반복되고 있는 지 패턴을 추출하는 것이 매우 중요하다. 패턴을 추출하여 반복문을 구성한 후에는 반복문에서 빠져나올 조건을 고려해야 하는데 만약 이 조건이 없다면 프로그램은 강제로 종료하기 전까지 무한히 반복하게 된다. 반복문의 탈출

조건이 반복 패턴의 어느 위치에서 정의될 수 있는가에 따라 반복문의 유형을 다양하게 정의할 수 있다. 랩터는 반복문의 유형은 하나이지만 반복문의 탈출 조건의 전후에 실행명령어를 위치할 수 있도록 함으로써 보다 직관적으로 알고리즘을 표현할 수 있다. 반면 스크래치는 반드시 반복문 진입 전에 탈출 조건을 점검하도록 되어 있어 학습자가 이해하고 있던 동작의 절차를 스크래치의 반복문 구성에 맞춰서 재구성하는 과정을 겪어야 하는 점에서 어려움을 호소하였다.



(a) 랩터

(b) 스크래치

[그림 9] 학습적용 사례 - 복합조건식의 수립

[그림 9]의 (a)는 랩터에서 가위바위보 게임에서 5승이 나올 때까지 반복하는 순서도를 나타내며 후조건 반복문으로 가위바위보 게임의 종결 여부를 체크하는 것을 알 수 있다. 반면 스크래치는 선조건 반복문만 제공하므로 무조건 최초의 한번 무의미한 비교를 하게 된다. 초보 학습자에게는 이와 같은 의도적인 절차가 자연스럽지 않아 반복문을 구성하는 시작점을 찾아내지 못하는 경우가 있어 프로그래밍이 어렵게 느껴지는 요인으로 작용하는 것을 발견할 수 있었다.

5.3 프로그래밍 언어별 학습 후 테스트 결과

<표 10>에 나타난 것처럼 A그룹(랩터)과 B그룹(스크래치)의 전체 평균과 표준편차가 서로 유사하다. 두 그룹의 성취도는 f-검정 결과 등분산으로 분석되며 t-검정에 의해 0.05의 유의미한 수준에서 서로 동일하다고 볼 수 있다. 따라서 비전공 학생들에게 공통의 프로그래밍 기초교육 내용을 적용한 경우 실습을 위해 비주얼 프로그래밍 언어와 다이어그래밍 언어 중 어느 것을 활용하건 평균적으로 동등한 수준에서 프로그래밍 역량을 배양한다고 볼 수 있다.

<표 10> 프로그래밍 기초교육 학습 평가 결과

항목	A그룹-랩터 (n=85)	B그룹-스크래치 (n=82)	f-검정	t-검정
평균 (표준편차)	67.64 (27,24)	68.78 (26.18)	0.72	0.78

앞서 확인한 프로그래밍 언어의 설계 원칙에서 도출한 프로그래밍 언어의 문법요소 별로 각 그룹별 성취도에 차이를 보이는 지 살펴보기 위해 <표 8>에서 제시했던 '가위바위보 하나빼기 게임' 문제의 평가 문항에 따라 변수의 활용, 입출력 기능, 복합연산, 반복문의 요소로 평가 결과를 상세화하였다.

<표 11> 프로그래밍 평가 문항별 분석

문법요소	A그룹 (n=85)	B그룹 (n=82)	f-검정	t-검정
변수의 활용	16.82 (5.46)	16.09 (3.98)	0.004	0.33
조건문 (연산자의 복합)	19.53 (10.73)	18.71 (10.50)	0.85	0.62
반복문	12.11 (0.58)	13.9 (7.33)	0.15	0.15
입출력	19.18 (9.04	21.06 (10.01)	0.35	0.55
총합	67.64 (27.24)	68.78 (26.19)	0.72	0.78

<표 11>에서 보는 바와 같이 변수를 활용하거나 조건문을 활용하는 항목은 A그룹과 B그룹이 취득한 평균 점수 간에 차이가 거의 없으며 등분산 t-

검정을 한 결과 역시 유의미한 차이는 없다고 판단할 수 있다.

다만, 변수의 활용을 점검하는 항목에서 f-검정한 결과 등분산의 구조가 아니었으며 이에 따라 t 검정은 이분산 가정 집단에 대한 t-검정을 진행하였다. 두 집단의 비슷한 수준에서 변수를 활용하였으나 변수의 정의와 사용이 쉬운 랩터 쪽이 다소간 평균점수가 높다고 볼 수 있다.

복합 관계식을 사용한 조건식은 거의 유사한 평가 결과가 나왔으나, 반복문의 경우 f-검정 값과 t-검정 값이 비교적 낮게 도출되었다. 반복문의 구성과 사용성에 있어서 랩터와 스크래치 간에 서로 차이가 있다는 점이 어느 정도 반영된 것으로 볼 수 있다.

이는 <표 4>의 간결성과 일반성과 관계하여 변수의 선언, 초기화, 변수 타입의 구분 등의 문법요소 항목에 있어 스크래치와 랩터가 큰 차이가 없다고 분석한 결과와도 일치한다.

6. 결론 및 제언

본 연구에서는 프로그래밍 기초학습에서 사용하는 비주얼 프로그래밍과 다이어그래밍 언어를 프로그래밍 언어의 설계 관점에서 비교 및 분석하였다. 프로그래밍 언어 간 차이를 보이는 7가지 문법요소를 도출하였으며, 이를 기반으로 간결성, 일반성, 효율성 측면에서 비교 수행하였다. 일반성은 간결성을 보장하지 않는다는 측면에서 보면, 스크래치의 간결성이 랩터보다 높아야 한다. 하지만 랩터가 일반성과 간결성 모두 스크래치보다 높은 결과를 보였다. 이러한 결과로부터 단순히 문법요소에 해당하는 구성자의 수를 늘리는 것이 프로그래밍 언어의 간결성 향상을 보장하지 않을 수 있다는 이론적 근거를 확인하였다.

또한, 이론적 근거의 신뢰성을 얻기 위해 자연과학대학 신입생 57명, 사범대학 신입생 56명, 경영경제대학 신입생 53명을 A와 B 그룹으로 나누고, A그룹에는 랩터를 활용한 순서도 기반 프로그래밍 교육을, B그룹에는 스크래치를 활용한 비주얼 프로그래밍 교육을 7주간 진행하였다. 교육과정에서 사용된 예제는 프로그래밍 언어 비교를 위한 문법요소의 학습이 가능하도록 설계하였다. 학습에 앞서 두 집단의 학생들이 프로그래밍과 문제해결 역량의 자기효능감에 있어 유의미한 수준에서 서로 같은 집단임을 확인하였다. 7주간의 교육과정을 마친 후 두 그룹을 대상으로 같은 문제에 대해 테스트를 진행하였다. 그 결과 랩터와 스크래치 그룹에서 모두 초보자의 프로그래밍 기초 역량을 유사한 수준으로 향상된 것으로 나타났다. 그러나 수업 과정과 테스트에서 동일한 문제를 해결하기 위해 각 그룹에서 산출된 결과물을 살펴보면 스크래치보다 랩터로 구현한 코딩 내용이 보다 간결하고 일반적인 표현으로 구성된 것을 확인할 수 있었다. 이는 프로그래밍 언어의 설계 관점에서 분석한 결과 랩터가 스크래치보다 일반성과 간결성에서 우세하다는 특징이 실제 비전공자의 코딩 내용에서 드러난 것이라 볼 수 있다.

이처럼 본 연구에서는 두 그룹의 학습자들이 작성한 랩터와 스크래치 프로그래밍 결과를 비교하여 앞서 제시된 이론적 근거가 타당함을 확인하였다. 또한, 이로부터 비전공자를 위한 교육용 프로그래밍 언어를 선택할 때 프로그래밍 언어의 문법적 기능의 다양성과 규모보다 학습자의 성향과 단계별 교육 목표에 맞게 프로그래밍 언어의 설계 관점에서의 특징을 고려하여 선택할 필요가 있다는 시사점을 얻을 수 있다.

향후 연구에서는 본 연구에서 다루고 있지 않은 프로그래밍 언어 설계 원칙에 대하여 프로그래밍 언어를 비교한다. 추가로, 현재 초·중·고 학생들에게 확대되고 있는 정보 교육의 내용을 고려하여 비주얼 프로그래밍 언어와 다이어그래밍 언어 이외에 텍스트 기반 프로그래밍 언어까지 포함할 예정이다.

참 고 문 헌

[1] 교육부 (2015). **소프트웨어 운영 지침**.

[2] Gorman, H. Bourn, L. E. (1983). Learning to think by learning LOGO: Rule learning in third-grade computer programmers, *Bulletin of the Psychonomic Society, 21*(3), 165-167.

[3] Alfred V. Aho. (2012). Computation and

computational thinking, *Computer Journal, 55*(7), 832-835.

[4] 김수환 (2015). Computational Thinking 증진을 위한 학습자 중심의 교수학습 전략의 효과. **정보교육학회논문지, 18**(3), 323-332.

[5] 최현종 (2011). 대학프로그래밍 강좌를 위한 프로그래밍 교육 프레임워크. **컴퓨터교육학회논문지, 14**(1), 69-79.

[6] 김수환 (2015). Computational Thinking 교육에서 나타난 컴퓨터 비전공 학습자들의 어려움 분석. **컴퓨터교육학회논문지, 18**(3), 49-57.

[7] 김민자·김현철 (2018). 컴퓨팅 사고력 관점에서 본 컴퓨터 비전공자 대상 교양 컴퓨팅 수업의 효과성 분석 연구. **컴퓨터교육학회논문지, 21**(1), 11-21.

[8] 홍성권·최정원·이영준 (2014). 초등학생의 게임 프로그래밍 경험이 자기효능감에 미치는 영향. **교원교육, 30**(3), 197-215.

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

[10] 송정범·조성환·이태욱 (2008). 메타인지 전략을 활용한 게임 프로그래밍 학습이 초등학생의 문제해결력에 미치는 효과. **교원교육**, 24(4), 432-447.

[11] 정웅열·이은경·이영준 (2009). 전문계 고등학교 학습자의 동기 유발 지속을 위한 프로그래밍 교수 학습 모형. **컴퓨터교육학회논문지, 12**(4), 13-21.

[12] Kanis C., Somkiat W. (2006). Visual Programming using Flowchart, *International Symposium on Communications and Information Technologies*, 1062-1065.

[13] Crews, T., Ziegler, U. (1998). The flowchart interpreter for introductory programming courses. *In Frontiers in Education Conference, FIE'98. 28(*1), 307-312.

[14] Giordano, D., Maiorana, F. (2015). Teaching algorithms: Visual language vs flowchart vs textual language. *Global Engineering Education Conference (EDUCON),* 499-504.

[15] 안성진·서영민·이영준 (2012). 교육용 프로그래밍 언어 연구 동향", **한국컴퓨터정보학회 학술발표논문집, 20**(1), 139-142.

[16] 최정원·이영준 (2014). 프로그래밍 학습에서 학습자의 어려움 분석. **컴퓨터교육학회논문지, 17**(5), 89-98.

[17] Piteira, M., & Costa, C. (2013). Learning computer programming: study of difficulties in learning programming. In Proceedings of the 2013 International Conference on Information Systems and Design of Communication, Lisboa. ACM

[18] Barker, L. J., McDowell, C., Kalahar, K. (2009) Exploring factors that influence computer science introductory course students to persist in the major. *In ACM SIGCSE Bulletin, 41*(1), 153-157.

[19] Bennedsen, J., Caspersen, M. E., Kölling, M. (Eds.) (2008). *Reflections on the teaching of programming: Methods and implementations (Vol. 4821).* Springer.

[20] Kazimoglu, C., Kiernan, M., Bacon, L., Mackinnon, L. (2010). Developing a game model for computational thinking and learning traditional programming through game-play. *In: Proceedings of the World Conference on E-learning in Corporate, Government, Healthcare and Higher Education 2010. AACE.* 1378-1386.

[21] Tsukamoto, H., Takemura, Y., Oomori, Y., Ikeda, I., Nagumo, H., Monden, A., Matsumoto, K. I. (2016). Textual vs. visual programming languages in programming education for primary schoolchildren. *In Frontiers in Education Conference (FIE),* 1-7.

[22] Noone, M., Mooney, A. (2017). First Programming Language: Visual or Textual?. *arXiv preprint arXiv:1710.11557.*

[23] Robert, W. S. (2018). *Concepts of programming languages.* Pearson.

[24] Kwon, D., Yoon, I., Lee, W. (2011). Design of programming learning process using hybrid programming environment for computing education. *KSII Transaction on Internet and Information Systems, 5*(10), 1799-1813.

[25] 이우숙·박선환·최은영 (2008). 성인의 문제해결 과정 측정도구 개발. **기본간호학회지, 15**(4), 548-557.

김 영 민

2010 중앙대학교 전자전기공학부(학사)
2012 중앙대학교 컴퓨터공학부(석사)
2017 중앙대학교 컴퓨터공학부(박사)
2017 ~ 현재 중앙대학교 다빈치교양대학 조교수
관심분야: 컴퓨터교육
E-Mail: e010120302@cau.ac.kr

이 민 정

1994 중앙대학교 컴퓨터공학과(공학학사)
1996 KAIST 전산학과(공학석사)
1996~2000 LG종합기술원 연구원
2000~2010 ㈜아이에이 수석연구원
2011~2015 ㈜삼성전자 소프트웨어센터 부장
2016~현재 중앙대학교 다빈치교양대학 조교수
관심분야: 컴퓨터교육, ITS, 인공지능, VR/AR
E-Mail: minjeonglee@cau.ac.kr