순서도를 활용한 프로그래밍 제어 구조 학습에 나타난 오류 유형 분석

최현종†

요 약

컴퓨팅 사고 교육에서 알고리즘의 설계는 학습자의 논리적 사고력과 절차적 사고력이 요구되는 중요한 학습 과정이다. 하지만 알고리즘 학습에 관한 연구와 학습자가 실제 학습에서 겪는 오류에 관한 연구가 부족한 실정이다. 이에 본 연구는 알고리즘 설계 학습에서 순서도를 활용한 프로그래밍 제어 구조 설계에서 발견된 학습자의 오류를 분석하여, 오류 유형을 제시하였다. 대학생을 대상으로 한 강의에서 세 가지 제어 구조에 관한 평가 문항을 제시한 결과, 순차 구조에서는 오류 유형이 발견되지 않았다. 하지만 조건 구조에서는 2개의 조건문이 중첩된 경우 조건 설정에서 오류가 발생하였다. 반복 구조에서는 반복의 횟수를 조절하는 조건, 반복되는 명령문의 위치, 중첩된 반복문에서 조건과 명령문의 위치 오류가 발견되었다. 본 연구에서 나타난 오류 유형은 초·중등학교와 대학에서 실시하고 있는 컴퓨팅 사고 교육의 알고리즘 설계 학습에 참고할 수 있는 사례가 될 것이다.

주제어 : 오류 유형, 제어 구조, 알고리즘, 컴퓨팅 사고

Analysis on Types of Errors in Learning about Control Structures of Programming using Flowchart

Choe Hyunjong†

ABSTRACT

Designing algorithms is a very important learning process in computational thinking education because it requires learner's logical and procedural thinking. But the case studies that have topics of algorithms learning and students' types of errors in learning algorithms are not enough. So the purpose of this study is to analyze students' errors that discovered in the process of learning three control structures of programming using flowchart and provide types of errors in designing algorithms. Results about tests of three types of control structures in university student's algorithms learning class showed different cases of types of errors; types of sequential control error are not presented in the class, types of conditional control error are presented in the case of setting the conditions of nested conditional control, and types of iterative control are showed in the many cases of iterative conditions, statements of single and nested iterative control structure. The results of study will be a good case study about teaching designing algorithms of computational thinking education in elementary, secondary school and university.

Keywords : Types of errors, Control structures, Algorithms, Computational thinking

† 종신회원: 서원대학교 부교수(교신저자)
논문접수: 2016년 1월 19일, 심사완료: 2016년 1월 26일, 게재확정: 2016년 1월 26일

1. 서 론

2015년 7월 21일 교육부와 미래창조과학부는 초등학교부터 대학까지의 소프트웨어 교육 청사진으로 'SW 중심 사회를 위한 인재 양성 추진 계획'을 발표하였다[1]. 이 추진 계획에 의해 교육부는 2015년 2월에 '소프트웨어 교육 운영 지침'을 고시하였고[2], 같은 해 9월에는 2015 개정 교육과정으로 초등학교부터 고등학교까지 학생들이 소프트웨어에 대한 기초 소양을 충실히 갖추어 나갈 수 있는 실과와 정보 교과의 교육과정을 제시하였다[3]. 초등학교 실과는 소프트웨어 관련 단원을 구성하고, 중학교의 정보 과목은 소프트웨어 교육 과목으로 개편하여 필수 과목으로 지정하였고, 고등학교의 정보 과목은 심화 선택에서 일반 선택 과목으로 편제를 수정하였다. 또한 미래창조과학부는 소프트웨어 중심 대학 사업을 통해 대학에서의 소프트웨어 교육을 지원하고 있다[4].

소프트웨어 교육은 최근 컴퓨터 교육학계의 새로운 교육 패러다임으로 자리잡고 있는데, 단순한 소프트웨어의 개발이 아니라 실생활의 문제를 컴퓨팅 사고(computatinal thinking)로 해결할 수 있는 역량을 기르는 교육으로 정의하고 있다[2]. 이는 소프트웨어 교육을 단순한 프로그램 개발 능력으로 한정하지 말고, 문제 해결을 위한 논리적이고 절차적인 사고 능력을 키우는 데에 집중해야 한다는 것을 의미하고 있다. 그래서 소프트웨어 교육 운영 지침과 2015 개정 교육과정의 정보 교과에서 제시하고 있는 학습 내용 영역 중 가장 중요한 영역이 바로 문제 해결을 위한 알고리즘, 프로그래밍이라 할 수 있다[5].

대학에서도 소프트웨어 교육은 컴퓨팅 사고에 기반한 해당 학문 분야의 문제 해결력을 높이는 방향으로 진행되고 있다. 특히 연세대학교의 경우 컴퓨터 과학의 원리를 활용하여 문제를 해석하고, 단계별로 풀어가는 사고 교육을 CT(Computational Thinking) 교육이라 지칭하고, 이를 위해 2016학년도 신입생 전원을 대상으로 CT 교육을 실시할 계획이다. 이 강좌는 기능적인 코딩 교육에 앞서 문제를 발견하고, 해결하는 사고력 향상을 목적으로 하고 있다[4].

즉 초등학교부터 대학에 이르기까지 소프트웨어 교육의 핵심은 결국 컴퓨팅 사고력 교육이라 할 수 있고, 이를 위한 핵심적인 학습 내용은 바로 문제 해결을 위한 알고리즘과 프로그래밍 활동임을 알 수 있다.

문제해결을 위한 알고리즘과 프로그래밍 교육은 문제의 이해와 분석, 알고리즘 설계, 프로그래밍이라는 세 가지 학습 활동이 서로 순차적으로 연결된 하나의 통합된 학습 활동이다[2]. 그렇기 때문에 소프트웨어 교육 운영 지침과 2015 개정 교육과정의 정보 교과에서는 이 세 활동이 문제 해결을 위한 각각의 단계로 제시되어 서로 연관되어 순차적으로 제시되어 있다.

하지만 실제 초·중등학교의 수업과 대학의 관련 수업을 살펴보면 프로그래밍에 집중된 수업들이 대부분이다. 컴퓨팅 사고력 교육의 관점에서 살펴보면 프로그래밍은 학습자가 문제 해결을 위해 설계한 알고리즘이 실제로 문제를 해결할 수 있는지를 확인하는 단계에 불과한데[5], 실제 수업에서는 문제를 이해하고 분석하여 문제해결방법을 알고리즘으로 설계해보는 과정보다 프로그래밍 활동에 치중하고 있다는 것이다. 따라서 최근의 컴퓨팅 사고 교육 관련 연구들을 살펴보면 알고리즘 설계 관련 주제보다 프로그래밍과 관련된 연구 주제를 가진 사례가 더 많은 것이 사실이다.

알고리즘 설계 단계는 문제를 분석하여 논리적 사고력으로 해결 방법을 절차적으로 설계해보는 지적 활동이 가장 왕성하게 일어나는 단계이므로, 소프트웨어 교육이 성공하기 위해서는 이 단계에 대한 많은 고민과 연구가 필요하다. 하지만 무엇보다도 알고리즘을 설계하는 학습 활동에서 학습자가 어떤 학습 요소를 어려워하고, 어떤 오류가 발생하는지에 관한 연구가 먼저 이루어져야 하는데, 아직 이에 관한 연구는 많이 이루어지지 않고 있다.

이에 본 연구에서는 대학생을 대상으로 알고리즘 설계 수업을 진행하였고, 수업의 평가 문제를 제시하였던 문제에서 발생하는 오류 유형과 내용을 분석하여 보았다. 평가 문제는 프로그래밍 구조의 핵심이라고 할 수 있는 세 가지 제어 구조에 관한 내용을 평가할 수 있는 문항으로 구성하였고, 이를 통해 학습자가 어떤 제어 구조를 어려

워하고, 어떤 오류 유형이 많이 발생되는지를 확인할 수 있었다.

2. 관련 연구

국내의 관련 연구들을 살펴보면 주로 프로그래밍 학습에서 학습자에게 나타나는 오류의 유형에 관한 연구들이 지속적으로 이루어졌다. 문외식(2006)은 정보영재 교육원에 재학 중인 초등학생들을 대상으로 비주얼베이식 프로그램으로 오류의 유형을 조사하였다[6]. 오류의 유형을 논리 오류, 데이터 정의 및 취급 오류, 화면 인터페이스 오류, 기타 오류로 분류하여 조사하였는데, 초등학생들의 특성과 문제의 난이도 때문에 기타 오류(명령어 타이핑 오류 등)가 가장 많이 발견되었고, 그 다음이 데이터 정의 및 취급 오류로 조사되었음을 발표하였다.

장혜선 외(2007)는 두리틀이라는 교육용 프로그래밍 언어를 활용한 초등학생 교육에서 나타나는 오류의 유형을 구문 에러, 의미 에러, 논리 에러로 구분하였다[7]. 초등학생의 특성으로 인해 역시 구문 에러가 많이 발생된 것을 확인할 수 있었다.

남재원과 유인환(2011)은 초등학생의 로봇프로그래밍 교육을 위해 오류 분석에 기반한 NXC 로봇 프로그래밍 지원시스템을 개발하는 연구를 진행하였다[8]. 이 연구에서 프로그래밍 오류의 유형을 문법 오류, 논리 오류, 코딩 오류, 로봇제어 오류로 분류하고, 실제 오류 발생 횟수를 조사하였다. 여기에서도 초등학생이기 때문에 로봇제어 명령어의 부정확한 사용이 가장 많은 오류를 보였다.

김지선과 김영식(2014)은 중·고등학생들을 대상으로 온라인 프로그래밍 학습을 통해 프로그래밍 개념 성취와 오류 유형의 관계를 분석하였다[9]. 이 연구에서는 과제 수행 결과를 문법 오류, 논리 오류, 코딩 오류로 분류하여 분석하였는데, 논리 오류의 유형이 가장 높게 관찰되었다.

최정원과 이영준(2014)의 연구는 대학생에게 코드닷오알지(code.org)의 퍼즐 문제를 제시하여, 어려워하는 부분에 대해 조사하였다[10]. 일반적인 프로그래밍 개념인 순차와 반복, 조건과 이벤트별로 어려워하는 부분들을 조사하였다.

김수환(2015)은 스크래치라는 교육용 프로그래밍 언어를 이용하여 대학의 교양 강좌에서 알고리즘과 프로그래밍 교육을 실시하였다[11]. 그는 이 연구에서 프로그래밍의 개념으로 변수와 리스트를 어려워하는 개념으로 제시하고, 프로그램의 설계 즉 알고리즘 측면에서는 아이디어를 생각하여 구현하는 과정과 어떤 명령어가 적절한지를 선택하는 것이 가장 어려워했다고 제시하고 있다.

국내의 프로그래밍 학습 오류에 관한 연구들은 초등학생과 중·고등 및 대학으로 구분할 수 있다. 초등학생과 관련된 연구에서는 대부분이 프로그래밍의 구문 오류들, 즉 익숙하지 않은 명령어들을 타이핑하여 입력해야 하는 상황에서 발생되는 오류들이 의외로 많이 발생하고 있었다. 하지만, 중·고등 및 대학의 경우에는 조금 다른 경향을 보였는데, 구문 오류보다 논리 오류의 측면이 더 두드러지게 나타났다. 이를 통해 텍스트 입력 방식의 프로그래밍 방법보다 프로그램의 논리적 구조를 설계하는 데에 더 많은 어려움을 겪는다는 것을 확인할 수 있었다.

국외의 연구들을 살펴보면 먼저 Murray(1987)는 학습자의 프로그래밍 오류를 지식 유형에 기초하여 5가지로 분류하였다[12]. 첫째 오류는 잘못된 문제 해석의 오류(mis-interpretation)로 문제의 핵심을 파악하지 못해서 발생한 오류이고, 두 번째 오류는 전략적 오류(strategic error)로 문제 해결을 위한 방법의 세부 절차를 계획하는 과정에서의 오류를 의미한다. 세 번째는 프로그래밍 언어로 번역하는 과정에서 생기는 오류이고, 네 번째는 통사적 오류로 프로그래밍 언어에 관한 지식적 오류이고, 다섯 번째는 코딩 오류로 타이핑 잘못으로 인한 오류이다.

Milne과 Rowe(2002)은 학습자들이 프로그래밍 학습에서 겪는 어려움을 확인하기 위해 대학생을 대상으로 C와 C++ 프로그래밍을 교육하고 설문을 실시하였다[13]. 이 설문을 통해 C 프로그래밍 교육에서 포인터와 메모리 할당이 가장 어려운 개념으로 파악되었다. 또한 Jenkins(2002)는 학습자들이 프로그래밍 학습을 할 때 겪게 되는 어려움을 조사하였다[14]. 그는 프로그래밍 학습의 어려움으로 프로그래밍 언어의 문제, 프로그래

밍이 어렵다는 이미지, 배워야 되는 많은 시간으로 인한 부담감, 새로운 학문에 관한 거부감, 알고리즘을 세분화하거나 코드로 옮기는 어려움, 언어의 문법적 문제로부터 의미적 구조에 이르는 복잡한 사고능력 등을 제시하였다.

Piteira와 Costa(2013)도 프로그래밍의 교육적 가치에 비해 초보자들이 프로그래밍 학습에서 겪는 어려움이 의외로 많다는 것을 주장하였다[15]. 초보자들이 겪는 어려움을 학습의 개념, 학습 환경, 교육 방법, 학습 주제와 교육 자료로 구분하였다. 대표적으로 학습의 개념에서는 포인터와 참조, 매개변수와 구조화된 자료 유형, 에러 처리 등이 있었고, 교육 방법으로 혼자 학습하는 것을 어려워하였고, 교육 자료로는 인터넷의 다양한 정보와 동영상 교육 자료 등이 쉽게 활용할 수 있는 것으로 조사되었다.

국외의 관련 연구들은 학생들이 프로그래밍 학습을 실시하면 겪게 되는 다양한 분야에 관한 연구들이 진행되고 있었다. 하지만 알고리즘을 설계하는 단계에서 발생되는 구체적인 학생들의 오류 유형에 관한 연구는 아직 많이 이루어지지 않고 있었다. 특히 중·고등학생의 경우 프로그래밍의 제어 구조에 관한 논리 오류가 많이 발생되는데[9], 이에 관한 자세한 연구 자료가 필요하다.

3. 연구의 설계

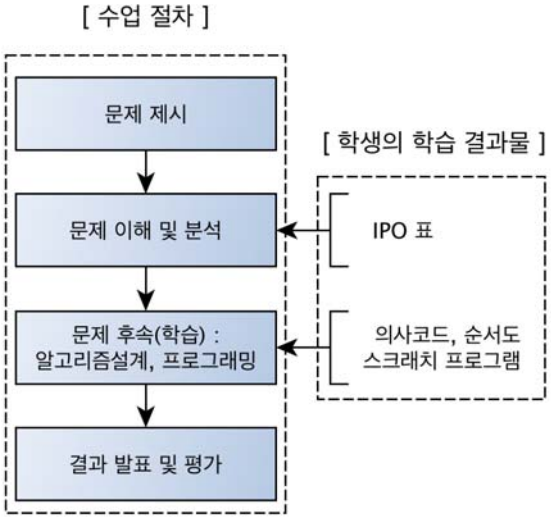
본 연구의 대상이 되는 강좌는 2015년 1학기에 실시되었고, 충청북도 청주에 소재한 대학교의 IT 학부 2학년 24명을 대상으로 하였다. 참고로 이 학생들은 모두 1학년 때에 C 프로그래밍 기초 과목을 1학기 동안 이수한 학생들이다.

연구 대상 강좌는 전공 선택 과목으로 알고리즘 사고 활동과 프로그래밍으로 구성하였는데, 프로그래밍의 경우 C와 Java와 같이 학습하기 어려운 프로그래밍 언어를 사용하지 않고 교육용 프로그래밍 언어인 스크래치(Scratch)를 사용하였다. 그 이유는 강좌의 목표가 프로그래밍 언어의 습득이 아니라, 알고리즘을 설계하는 능력을 향상시키는 것이기 때문이다. 그래서 의사 코드와 순서도로 문제 해결을 위한 알고리즘을 작성하는 활동을 강좌의 중심 활동으로 구성하였고, 스크래치 언어는 학생들이 설계한 알고리즘이 문제를 해결할 수 있는지를 확인할 수 있는 도구로 활용하였다. 주차별 강의 내용은 <표 1>과 같다.

<표 1> 강의의 주차별 학습 내용

주	학습 내용
1	강의 안내
2	문제해결과 컴퓨터
3	프로그램의 설계와 작성
4	프로그램의 논리와 구조
5	입출력과 연산자, 순차 구조
6	문제해결에서 선택 구조의 이해와 적용
7	다중 선택 구조의 이해와 적용
8	중간 시험
9	문제해결에서 반복 구조의 이해와 적용
10	다중 반복 구조의 이해와 적용
11	문제해결에서 배열의 이해와 적용
12	문제해결에서 함수의 이해와 적용
13	개인 과제(문제 제시와 분석, 설계)
14	개인 과제(프로그램 시현과 평가)
15	기말 시험

15주의 수업에서 4주까지는 강의법으로 진행되었고, 5주부터 12주까지는 문제중심학습(PBL: Problem Based Learning)을 적용하여 <그림 1>과 같은 절차로 수업을 진행되었다.



<그림 1> 강의의 절차와 학습 결과물

먼저 문제 제시 단계에서 실생활 문제를 학생들에게 제시하면, 학생들이 문제 이해 및 분석 단계에서 문제를 분석하여 IPO(Input Process Output) 모델을 적용한 간단한 IPO 표를 작성한

다. IPO 표에 표현된 개념 중에서 학습이 필요한 학습 요소들은 교수자가 설명하거나, 학생들이 학습할 수 있도록 한다. 문제후속단계인 학습 단계에서는 IPO 표에 표현된 문제해결방법을 의사코드와 순서도로 작성한다. 그리고 작성된 순서도에 따라 스크래치로 프로그램을 작성한다. 마지막 단계인 결과 발표 및 평가 단계에서는 자신이 작성한 순서도와 스크래치 결과물을 발표한다. 참고로 순서도의 작성은 무료 프로그램인 yEd 그래픽 프로그램을 사용하였다[16].

본 연구에서 사용한 검사 도구는 <표 2>와 같이 5주차부터 10주차까지 학습한 프로그램의 제어 구조인 순차, 선택, 반복을 평가할 수 있는 문항으로 구성하였다. 문항은 서술형 형태로 구성하였고, 학생들은 순서도로 답안을 작성하였다. 검사 도구의 문항은 문제 제시 단계에서 제시하였던 실생활의 문제보다 수와 관련된 문항을 구성하여 알고리즘과 프로그래밍이 쉽게 연관될 수 있도록 구성하였다.

<표 2> 평가 문항의 내용과 선정 의도

번호	문항 내용	선정 의도
1	수를 입력받아 2배수를 출력	변수와 입출력의 적용
2	두 개의 수를 입력받아, 그 곱을 출력	변수와 연산자의 적용
3	입력된 수가 90 이상이면 합격을 출력	조건문의 적용
4	90 이상이면 A, 80 이상부터 90 미만이면 B, 80 미만이면 C를 출력	다중 조건문의 적용
5	“안녕하세요.”를 5회 반복하여 출력	반복문의 적용
6	1부터 10까지의 합을 출력	변수 1개와 반복문의 적용
7	구구단 2단을 출력	변수 2개와 반복문의 적용
8	구구단을 2단부터 9단까지 출력	중첩된 반복문의 적용

4. 연구의 결과

본 연구의 평가 도구는 프로그래밍 학습에서 가장 중요한 요소인 제어 구조(control structure)를 중심으로 실시하였다. 여기에 나타난 제어 구조 학습의 오류 유형을 분석하기 이전에, 문항별 정·오답 분석을 살펴보면 <표 3>과 같다. 평가에 사용된 문항에서 완벽하게 문제를 해결한 경우(정답)와 문제를 해결하는 과정에서 오류를 범한 경우(오답)의 빈도와 백분율을 각각 표시하였다. 학생들의 답안에서 전혀 문제를 해결하지 않은 무응답은 나타나지 않았다. 주차별 학습을 한 후에 바로 평가하였는데, 제어 구조의 난이도가 높을수록 오답율이 높게 나타났다.

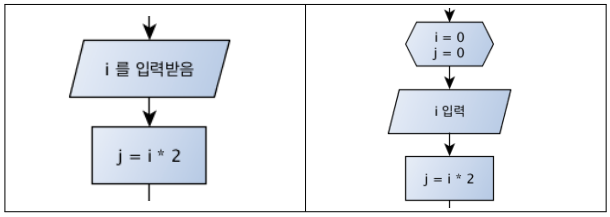
<표 3> 평가 문항별 정오답 빈도와 비율

문항	정답		오답	
	빈도(명)	백분율(%)	빈도(명)	백분율(%)
1	24	100.0	0	0
2	24	100.0	0	0
3	24	100.0	0	0
4	23	95.8	1	4.2
5	21	87.5	3	12.5
6	19	79.2	5	20.8
7	15	62.5	9	37.5
8	11	45.8	13	54.2

세 가지 제어 구조에 따라 발견된 오류 유형은 다음과 같다.

4.1 순차 구조 학습에서 발견된 오류

순차 구조에서 발생되는 오류는 크게 3가지 유형이 있다. 필요한 명령어(들)을 생략하는 경우와 필요하지 않은 명령어(들)을 추가하는 경우가 있고, 명령어들을 순서에 맞게 바르게 배열하지 못하는 경우이다. 본 연구의 1번과 2번 문항이 순차 구조에 해당하는 평가 도구인데, 비교적 단순한 평가 문항이기 때문에 오답이 발견되지 않았다. 단, 학습자에 따라 변수를 초기화하는 경우와 초기화 하지 않고 사용하는 경우로 나눌 수 있는데, 이는 사용하는 프로그래밍 언어의 변수 사용 특성에 따라 적절하게 지도하면 될 것이다.

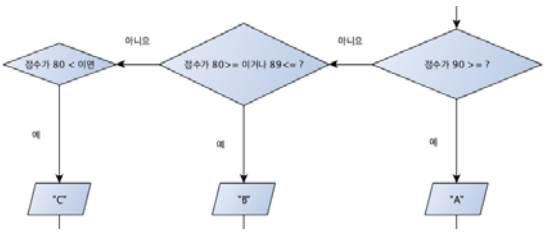


<그림 2> 초기화가 있는 순서도와 없는 순서도

4.2 선택 구조 학습에서 발견된 오류

선택 구조는 특정 조건의 참과 거짓에 따라 명

령어(들)가 결정되어 실행되는 구조이다. 이 구조의 문제에서는 크게 두 가지 오류 유형이 발생될 수 있다. 첫 번째는 선택 구조의 조건을 잘못 기술하는 오류, 두 번째는 조건의 참과 거짓에 따라 실행되는 명령어(들)가 생략되거나 불필요한 명령어(들)가 추가되어 기술된 경우이다. 본 연구의 3번과 4번 문항이 선택 구조에 해당하는 평가 도구인데, 하나의 조건문이 사용되는 3번 문항은 오답이 발생하지 않았고, 두 개의 조건문이 사용되는 4번 문항에서 오답이 발생되었다. <그림 3>에 제시된 순서도는 학생의 답안으로 2개의 조건문이 사용되어야 하는데 3개의 조건문이 사용되었고 두 번째 조건문의 조건에 2개의 조건이 사용되었다. 이 알고리즘에 따라 프로그램을 작성하면 실행은 되지만, 조건문의 조건에 대한 이해가 부족하다고 판단되어 오답으로 처리하였다.



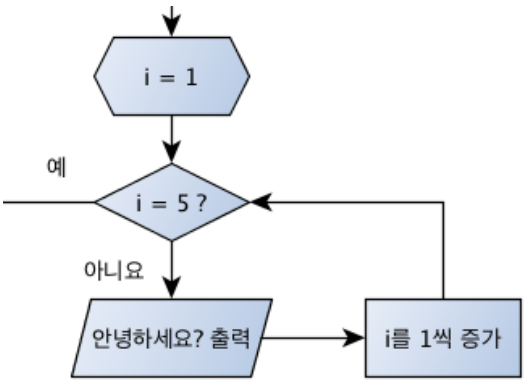
<그림 3> 선택 구조 오류 유형

4.3 반복 구조 학습에서 발견된 오류

반복 구조는 특정 조건에 따라 명령어(들)가 일정한 횟수 동안, 혹은 특정 조건이 참일 때 명령어(들)가 반복되는 구조이다. 본 연구에서는 특정 조건에 따라 일정한 횟수 동안 명령어(들)가 반복되는 경우에 한정된 문제를 제시하였다.

반복 구조의 문제에서 발견된 오유 유형은 크게 네 가지이다. 첫 번째는 반복 횟수를 잘못 설정하는 오류이었고, 두 번째는 반복되는 횟수를 설정하는 특정 조건에 대한 오류이다. 이 두 가지 유형은 서로 비슷해 보이지만, 서로 다른 오류 유형으로 첫 번째는 반복 조건을 바르게 설정했지만 단순히 반복되는 횟수를 잘못 설정하는 경우이고, 두 번째는 반복 조건 자체를 제대로 설정하지 못하는 경우이다. 세 번째 오류 유형은 반복해야 할 명령어(들)를 제대로 설정하지 못하는 경우이고, 네 번째 오류 유형은 두 개의 반복문이 서로 중첩되는 문제에서 중첩 구조를 제대로 만들지 못하는 경우이다. 순서도를 이용하여 두 개의 반복문이 서로 중첩될 경우에 한 개의 반복문 출구를 다른 반복문의 입구로 연결하는 스택킹(stacking) 기법이 필요하다[17]. 중첩 구조를 제대로 만들지 못하는 경우는 대부분 이 스택킹 기법에 대한 이해가 부족한 경우이다.

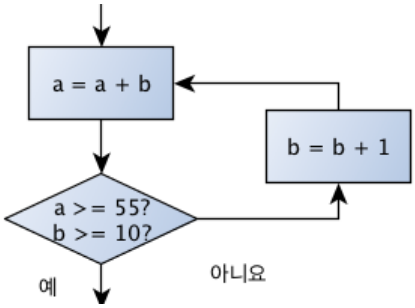
문제 5는 “안녕하세요.”를 5번 출력하는 프로그램의 순서도를 작성하는 문제이다. <그림 4>는 오답을 작성한 학생의 순서도로, 반복문의 조건과 실행되는 명령문은 잘 표현했지만, “안녕하세요.”를 4번만 출력하고 종료한다. 변수가 증가함에 따라 반복되어 실행되는 횟수에 대한 이해가 부족한 경우로 문법적인 오류가 아니라, 논리의 오류라고 판단된다.



<그림 4> 반복 구조 오류 유형 1

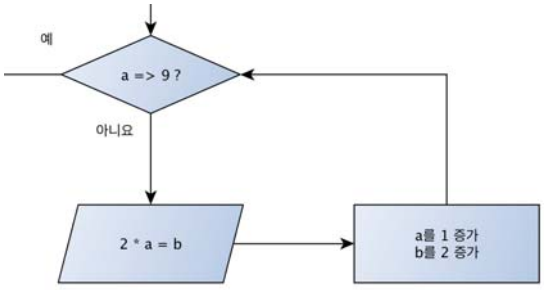
두 번째 오류 유형은 6번 문제에서 발견할 수 있었다. 6번 문제는 1부터 10까지의 합을 출력하는 프로그램의 순서도를 작성하는 문제이다. 5번 문제에서는 반복 조건을 제어하는 변수 한 개만 있는데, 8번 문제에서는 반복 조건을 제어하는 변수와 그 변수가 1씩 증가하면서 더해지는 합계를 나타내는 변수가 필요하다. 따라서 두 개의 변수가 필요한 이유와 그 변수가 반복문에 따라 값이 점차 더해지는 개념을 확실하게 이해하여야만 해결할 수 있는 문제이다. 학생의 오답은 <그림 5>와 같은데, 반복문의 조건 설정 오류라고 할 수 있다. 순서도에 제시된 알고리즘이 실행은 되지만, 반복문의 실행 조건을 판단하는 기준이 되는 변수의 처리를 두 개의 변수를 모두 처리할 필요

는 없고, 변수 b 로 반복 조건을 제어해야 효율적인 알고리즘이라고 할 수 있다.



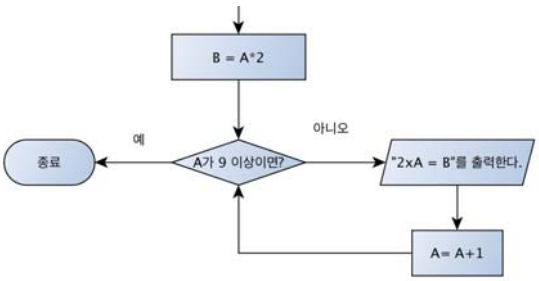
<그림 5> 반복 구조 오류 유형 2

반복 구조의 세 번째 유형은 반복되는 명령문을 제대로 처리하지 못하는 유형인데, 필요 없는 명령문을 반복되도록 하거나 반복해야 하는 명령문을 반복하지 못하게 하는 경우가 있다. 7번 문항은 구구단 2단을 순차적으로 출력하는 프로그램의 순서도를 작성하는 문제이다. <그림 6>은 필요 없는 명령문을 반복시키는 오류 유형으로, 구구단에서 1씩 증가되는 변수는 a 인데, 곱셈의 결과인 b까지 2씩 증가시키는 오류를 표현하고 있다.



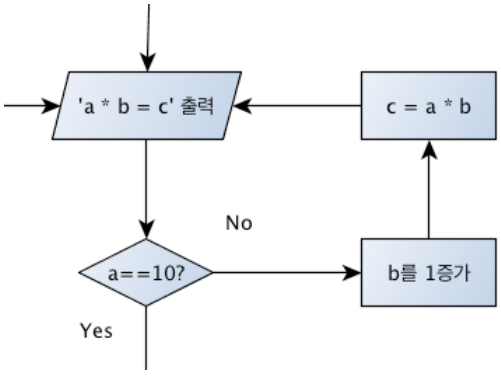
<그림 6> 반복 구조 오류 유형 3-1

구구단 문제에서의 또 다른 오류는 <그림 7>처럼 반복되어야 할 명령어를 반복 구조에 넣지 못하고, 반복문 밖에 위치시킨 경우이다. 이 경우도 세 번째 오류 유형에 속한다.



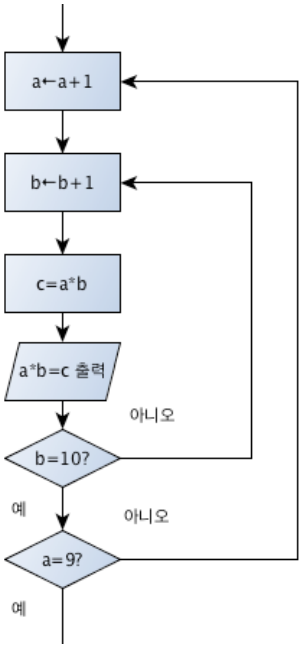
<그림 7> 반복 구조 오류 유형 3-2

반복 구조의 마지막 오류 유형은 중첩된(nested) 반복문 구조에 대한 오류이다. 문제 8번은 구구단 2단부터 9단을 순차적으로 출력하는 문제인데, 구구단의 각 단이 1부터 9배수까지 출력되기 위해서는 두 개의 반복문이 서로 중첩 구조를 이루어야 한다. 즉 한 개의 반복문은 설정된 단수의 1배부터 9배까지 출력해야 하고, 다른 한 개의 반복문은 단수를 2부터 순차적으로 9까지 1씩 증가시켜야 한다. 반복문 중첩 구조에서도 두 가지 유형의 오류 유형이 발견되었는데, 첫 번째 유형은 반복 조건 설정 오류이다. 내포된 반복문이 일정 횟수 동안 실행된 후 종료되어야 되는데, 그렇지 못한 경우이다. <그림 8>은 반복문의 조건 설정이 잘못 되어 무한 반복되는 경우이다.



<그림 8> 반복 구조 오류 유형 4-1

두 번째 유형은 중첩 구조 안에서 명령문 설정 오류이다. <그림 9>는 학생이 제시한 순서도인데, 단수 a가 하나 증가하면 출력되는 배수 b는 1부터 9까지 순차적으로 증가하게 된다. 이를 위해서 배수 b는 내포된 반복문이 종료되고 나면 다시 1로 배정하는 명령문이 필요한데, <그림 9>는 이 명령문이 없는 오류 상황이다.



<그림 9> 반복 구조 오류 유형 4-2

4.4 오류 유형 분석

프로그램의 제어 구조에 관한 오류 유형을 분석한 결과, 순차 구조는 실행되는 명령어들을 순차적으로 나열하면 되기 때문에 특별한 오류 유형을 발견하지는 못하였다. 단지 순서도에 표현하는 명령어들이 얼마나 구체적으로 실행 가능한 명령어 형태인지가 관건이었다.

선택 구조에서는 2개의 조건문이 중첩되어 사용되었을 때, 조건을 명확하게 기술하지 못하는 오류를 발견하였다. 따라서 중첩 조건문을 학습할 때에는 조건문에 사용하는 조건에 대해 좀 더 자세하고 다양한 예제를 활용한 설명이 필요하다고 판단된다. 참고로 평가 문항 4번과 같은 수 문제의 경우에는 수직선으로 조건문의 조건을 그림으로 제시하면 학생들이 좀 더 쉽게 이해할 수 있었다.

반복 구조는 학생들이 가장 어려워하는 구조이다. 왜냐하면 선택 구조는 선택 조건에 따라 명령문이 한 번만 실행되는 경우가 많은데, 반복 구조는 조건의 참과 거짓에 따라 포함된 명령문이 일정 횟수 동안 계속 반복하여 실행되는 구조이기 때문이다. 따라서 조건을 판단하여야 하고, 그 조건에 따라 실행되는 명령문과 포함된 변수의 변화를 함께 판단하여야 한다. 반복 구조의 오류 유형은 크게 반복 횟수를 조절하는 조건과 반복되는 명령문, 그리고 중첩 구조인 경우에는 중첩 구조 안에서 반복 조건과 명령문 설정 오류가 발견되었다. 반복문이 중첩 구조일 때 특히 많은 학생들이 오답을 작성하였는데, 두 개의 반복문에서 반복 조건을 변수로 제어하고, 그 반복문에서 명령문이 반복되는 경우를 논리적으로 생각하는 학습이 부족했다는 것을 확인할 수 있었다. C 프로그래밍 학습을 선수하였다고 할지라도, 순서도를 활용하여 알고리즘을 설계하는 학습에서 이런 다양한 오류 유형을 발견할 수 있다는 것은 학생들이 기존의 C 프로그래밍 실습 학습에서 알고리즘 설계보다 C 언어의 문법 학습에 치중하였다는 것을 확인할 수 있었다.

5. 결론

소프트웨어 교육의 본질은 컴퓨팅 사고를 향상하기 위한 교육이고, 이를 위해 문제를 이해하고 분석하여 해결할 수 있는 알고리즘을 설계하고, 설계한 알고리즘을 프로그래밍 언어로 평가하는 일련의 과정이 복합적으로 연결된 학습 내용으로 구성되어 있다. 하지만 많은 연구와 실제 학교의 학습에서 중요하게 다루고 있는 것은 프로그래밍 언어 교육인데, 그 이유는 프로그래밍 활동이 학습자에게 즉각적인 피드백을 제공해 주어 학습의 동기 유발이 좋고, 학습 효과를 직관적으로 관찰하기 좋은 활동이기 때문이라 판단된다. 하지만 컴퓨팅 사고에서 요구하는 학생들의 절차적인 논리에 관한 사고력은 프로그래밍 전 단계인 알고리즘 설계 단계에서 많이 개발된다. 이에 본 연구는 알고리즘 설계 학습을 위주로 강의를 진행하였고, 그 학습 과정에서 학생들이 어려워하는 개념인 프로그래밍의 세 가지 제어 구조에 관한 문제에서 발생되는 학생들의 오류를 파악하였다.

순차 구조 학습에서 발견된 오류는 거의 발견되지 않았고, 선택 구조 학습에서는 2개의 조건문이 중첩되어 사용된 경우에 조건 처리에 관한 오류가 발견되었다. 반복 구조 학습에서는 비교적 다양한 오류 유형이 발견되었다. 반복 조건과 반복되는 명령문을 잘못 설계한 경우, 2개의 반복문

이 중첩된 구조에서의 오류 등이 발견되었다.

본 연구를 통해 파악된 프로그래밍 제어 구조 학습에서의 오류 유형은 실제 초·중등학교와 대학에서 알고리즘을 설계하는 학습에서 유용한 자료로 사용될 것이다. 왜냐하면 오류 유형을 미리 파악하고 있다는 것은 교사의 입장에서 학습자가 오답을 할 가능성을 줄일 수 있는 효과적인 학습 전략이기 때문이다. 추후 본 연구를 기반으로 제어 구조에 관한 다양한 오답 사례를 중학교와 고등학교 학생들을 대상으로 수집하고, 복잡한 문제에서 제어 구조 선택 관련 오류에 대해 좀 더 조사하여 일반화된 오류 유형을 연구하고자 한다.

참 고 문 헌

[1] 교육부 (2015). **초등학교에서 대학까지, 소프트웨어(SW) 교육 청사진 나왔다!-교육부·미래부,「SW중심사회를 위한 인재양성 추진계획」발표.** 교육부·미래창조과학부 보도자료.

[2] 교육부 (2015). **소프트웨어(SW) 교육 운영 지침.** 교육부

[3] 교육부 (2015). **2015 개정 교육과정 총론 및 각론 확정 발표**. 교육부 보도자료.

[4] 미래창조과학부 (2015). **소프트웨어 중심 대학 추진 계획.** 미래창조과학부 주요정책정보.

[5] 이태욱. 최현종 (2015). **정보교과교육론**. 한빛아카데미.

[6] 문외식 (2006). 초등학생들이 프로그래밍 학습 시 발생하는 오류유형 분석. **한국컴퓨터정보학회 논문지, 11**(2), 319-327.

[7] 장혜선 (2007). **초보자의 프로그래밍 학습을 위한 에러 피드백 시스템 설계.** 박사학위 논문, 고려대학교.

[8] 남재원. 유인환 (2011). 오류분석에 기반한 NXC 로봇프로그래밍 지원시스템의 개발. **정보교육학회 논문지, 15**(3), 375-385.

[9] 김지선. 김영식 (2014). 온라인 프로그래밍 개념학습 성취수준과 오류유형과의 관계 분석. **한국컴퓨터교육학회 논문지, 17**(5), 43-51.

[10] 최정원. 이영준 (2014). 프로그래밍 학습에서 학습자의 어려움 분석. **한국컴퓨터교육학회 논문지, 17**(5), 89-98.

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

[12] Murray, W. R.(1987). Automatic program debugging for intelligent tutoring systems. Computational Intelligence, 3, 1-16.

[13] Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming-views of students and tutors. Educationa and Information Technologies, 7(1), 55-66.

[14] Jenkins, T. (2002). On The Difficulty of Learning to Program. 3rd Annual LTSN-ICS Conference, Loughborough University. 53-58.

[15] 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.

[16] yEd Graphic Editor. http://www.yworks.com/en/products/yfiles/yed/

[17] Joyce Farrell (2012). An Object-Oriented Approach to Programming Logic and Design, 4th edition. Course Technology.

최 현 종

2001 한국교원대학교 컴퓨터교육과(교육학석사)
2005 한국교원대학교 컴퓨터교육과(교육학박사)
2006~현재 서원대학교 컴퓨터교육과 부교수
관심분야: 컴퓨터교육, Semantic Web
E-Mail: blueland@seowon.ac.kr