Journal of Korea Multimedia Society Vol. 20, No. 12, December 2017(pp. 2000-2008)
https://doi.org/10.9717/kmms.2017.20.12.2000

프로그래밍 제어구조 개념 학습을 위한 오개념 순서도 분석 및 적용

최 영 미†

Analysis and Application of Misconception Flowchart for Programming Control Structure Concept Learning

Youngmee Choi†

ABSTRACT

The purpose of this study is to analyze the misconception flowchart of programming control structure and to suggest it as a method for improving computational thinking. In this study, we divide programming control structure concept into sequential, iteration, selection, and function, analyze what concept and principle are difficult for each learner, and what kind of misconception flowchart is drawn in the Introduction of C Programming course for beginners' programming learning. As an example, we show that a lesson learned from the process of correcting the misconception flowchart to the correct flowchart in the course.

Key words: Misconception Flowchart, Control Structure Concept, Algorithm, Computational Thinking

1. 서 론

알고리즘은 어떠한 문제를 수행하기 위하여, 잘 정의된 명령어들을 적절한 순서대로 나열하고, 그 나열된 순서대로 명령어들을 차례대로 수행하면 결국 최종적으로 그 문제가 완성되게 하는 효과적인 방법이다[8]. 컴퓨터를 사용하여 문제를 해결하기 위하여 컴퓨터에서 직접 실행이 가능하도록 알고리즘을 설계하여 프로그램을 작성하는 과정이 필요하다. 이 과정에서는 문제를 이해하고 분석하여 해결할 수 있는 알고리즘을 설계하고, 설계한 알고리즘을 프로그래밍 언어로 코딩하고 테스트하는 과정이 복합적으로 연결되어 진행된다.

컴퓨팅 사고(Computational Thinking)는 복잡한 문제를 분해하여 문제 안에 내재된 패턴을 찾고 추상화 단계를 거쳐서 문제를 해결하는 알고리즘을 작성하는 과정이다. 알고리즘은 컴퓨팅 사고의 4가지의 토대(분해, 패턴, 추상화, 알고리즘) 중 하나이다. 알고리즘은 사람이 수행하기도 하지만 대부분의 경우 컴퓨터를 이용하여 문제를 해결하고자 할 때 필요하다. 알고리즘은 프로그래밍 언어로 구현하면 컴퓨터 프로그램이 된다. 즉 컴퓨팅 사고에서 요구하는 학생들의 절차적인 논리에 관한 사고력은 프로그래밍 전 단계인 알고리즘 설계 단계에서 많이 개발된다[9]. 알고리즘을 표현하는 방법의 종류로는 자연어, 순서도, 의사 코드 등이 있으며 특히 순서도는 정해진 규칙과 기호로 프로그램의 논리를 표현할 수 있으므로 의사 코드와 더불어 가장 많이 사용하는 방법이다.

프로그래밍을 처음 시작하는 학습자들은 프로그래밍이 요구하는 엄격한 문법과 구조 이해와 고도의

※ Corresponding Author : Youngmee Choi, Address: (14097) Sungkyuldaehak-ro 53, Manan-gu, Anyang, Korea, TEL : +82-31-467-8164, FAX : +82-31-449-0529, E-mail : choiym@sungkyul.ac.kr

Receipt date : Oct. 20, 2017, Revision date : Nov. 30, 2017
Approval date : Dec. 6, 2017

† Dept. of MediaSoftware, School of Engineering, Sungkyul University

추상적인 사고 훈련 부족으로 인하여 학습의 어려움을 많이 호소하고 있다. 더욱이 학생들이 프로그래밍을 선행 학습하였다고 할지라도 순서도를 활용하여 알고리즘을 설계하는 학습 현장에서 다양한 오류 유형을 발견할 수 있었는데, 이는 학생들이 기존의 프로그래밍 수업에서 알고리즘 설계보다 프로그래밍 언어의 문법 학습에 치중한 것에 그 원인이 있다고 본다. 프로그래밍 언어는 아이디어를 실현시키는 코딩에 훌륭한 도구일 뿐이다. 그런데 대부분 수업현장에서 프로그래밍 언어 교육을 중요하게 다루고 있는 이유는 학습자에게 즉각적인 피드백을 제공해 주어 학습의 동기 유발이 좋고, 학습 효과를 직관적으로 관찰하기 좋은 활동이기 때문이었다. 그러나 문제의 구조를 파악하고 그것을 간단히 표현하여 일관성 있는 규칙으로 정리한다는 것은 프로그래머에게는 일상적인 활동이므로 코딩 전 단계인 아이디어를 체계적으로 구체화하는 컴퓨팅 사고를 향상시키는 훈련이 필요이다.

이에 본 연구는 순서도를 사용한 알고리즘 설계에 중점을 두어 대학교 1학년 프로그래밍 수업을 순서도작성 워크북을 활용하여 진행하였다[7]. 프로그래밍 기초 교육에서 초보 학습자들이 어려워하고 오류를 발생시키는 부분은 어디인지, 그 이유는 무엇인지에 대하여 프로그래밍 제어구조 개념을 중심으로 관찰하고 분석하였다. 또한 오개념 순서도를 정답 순서도로 수정하는 과정에서 프로그래밍 학습경험을 보인다.

본 논문의 구성은 2장 연구 배경에서 오개념의 교육적 의의 및 그 활용과 프로그래밍 교육을 위한 학습방안을 살펴보았다. 3장에서는 프로그래밍 기초 교육에서 초보 학습자들이 어려워하고 오류를 발생시키는 부분은 어디인지, 그 이유는 무엇인지에 대하여 프로그래밍 제어구조 개념을 중심으로 분석하였다. 4장에서는 학생들이 작성한 순서도의 오류를 프로그래밍 제어구조의 오개념 순서도 유형으로 분석하고 오개념 순서도를 정답 순서도로 수정하면서 프로그래밍 학습과정을 기술하였다. 5장에서는 적용사례로 초보 프로그래머가 스스로 오류를 찾고 개선할 수 있도록 설계한 워크북을 사용한 학생들의 활동의 일부를 소개한다. 6장 연구 결과에서는 학생들이 스스로 오류의 원인을 확인하고 수정하면서, 제어구조 개념에 대한 정확한 이해와 내면화, 정교한 알고리즘 작성할 수 있는 학습경험을 통하여 오개념 순서도를 컴퓨팅 사고력 향상에 집중할 수 있는 학습 방안으로 제시한다.

2. 연구배경

2.1 오개념(誤槪念, misconception)의 교육적 의의 및 그 활용

오개념(誤槪念, misconception)은 학습자가 스스로 알게 된 지식이나 학교에서 가르친 지식을 잘못 해석한 것으로써 학습자의 오류이다. 인간이 오류를 범하는 불완전한 존재라는 것은 곧 인간이 개선될 수 있다는 가능성을 함의한다. 교육은 바로 이러한 인간의 불완전성과 개선 가능성을 전제로 이루어지는 활동이라는 점에서 인간의 오류는 교육적으로 매우 각별한 의미를 가진다[10].

학습자가 오개념을 가지고 있는 경우, 교수가 단 한번 지적을 하거나 시범을 보이는 방법으로 학습자가 금방 자신의 생각을 포기하지 않는다. 왜냐하면, 학습자가 범하는 오류는 그보다 높은 수준의 스승이나 전문가의 입장에서 그렇게 간주되는 것이지, 그것의 주체인 학습자의 입장에서는 '정답'인 것이며, 그가 지니고 있는 인식 체계는 현재로서는 그에게 최선의 것이기 때문이다. 그래서 학습자는 교수가 자신의 오류를 지적하여도 쉽게 자신의 오류를 직면하거나 그것을 극복하기가 어렵다. 배움이 가장 잘 이루어지는 시점은 학습자 스스로가 자신의 부족함과 오류 등을 직면함으로써 그것을 극복하겠다는 동기와 의지가 갖추어질 때이다. 다른 사람에 의해 자신의 오류가 지적되었다고 하더라도 학습자 스스로 자신의 오류를 자각하는 경험이 필요하다.

프로그래밍 학습 시 자신의 인지 방식과 괴리된 가르침이 계속되는 상황 속에서 학습 결손이 누적되어가고 정서적으로는 학습된 무기력을 가지게 되어 코딩을 멀리하는 결과를 가져올 수 있다. 학습자가 보이는 순서도의 오류 유형을 제시하고 수정하는 과정은 학습자 스스로 자신의 오류를 자각하는 간접적인 경험을 할 수 있다. 프로그래밍 분야에서도 학생들이 자신의 인지 방식의 오류와 한계를 생생하게 자각하고 또 수정하는 기회를 제공하는 오개념의 교육적 활용 가치는 높다고 본다.

2.2 프로그래밍 교육을 위한 교수 학습 방안

프로그래밍 교육을 위한 교수학습 방안 수립 시 고려할 사항을 제시하면 다음과 같다[4,5].

첫째, 프로그래밍 제어구조 개념을 정확하게 이해하고 내면화하기 위한 전략이 필요하다. 제어구조 개념에 대한 충분한 이해 없이 해결만을 목적으로 하는 학습은 문제 유형을 기억해두었다가 같은 유형이 등장하면 매치되는 방법을 적용하는 수준에 그치기 때문에 익히지 못한 유형의 문제는 해결하기 어렵다. 따라서 개념에 대한 대략적인 이해 수준을 넘어 각 개념이 문제 해결에 어떤 역할을 하고 왜 필요한지, 어떤 상황에 어떻게 적용될 수 있는지를 정확히 깨닫도록 하는 것이 가장 중요하다. 프로그래밍 제어구조 개념에 대한 내면화는 학습자가 문제 해결에 적합한 개념을 선택하도록 하며 문제 해결의 효율성 향상을 꾀할 수 있다.

둘째, 문제 해결 절차를 가시화 할 수 있는 정교화된 알고리즘을 작성하는 연습이 충분히 이루어지도록 지도할 필요가 있다. 정교하게 짜여진 알고리즘은 문제 해결을 위한 심사숙고의 과정을 거치도록하기 때문에 계획 없이 작성된 프로그램을 반복적인 시행착오를 통해 해결하려는 소극인 태도를 개선할 수 있다. 또한 문제 해결 맥락을 논리적이고 체계적으로 살펴볼 수 있도록 하므로 문제 해결에 필요한 블록이 생략되거나 불필요한 블록이 추가되지 않도록 하며, 반복되는 패턴을 인식하거나 중첩 구조를 쉽게 발견하도록 한다. 문제 해결 과정에서의 논리적인 오류를 탐지하고 수정하는 과정을 용이하도록 한다. 왜냐하면 알고리즘을 머릿속으로만 구상하는 수준에서 문제를 해결하는 것은 사람이 갖는 사고의 한계로 인하여 학습자들로 하여 많은 시행착오를 겪도록 하기 때문이다.

셋째, 학습해야 하는 프로그래밍 제어구조 개념을 어떠한 순서로 구성하여 가르칠 것인가에 대한 심도 깊은 설계가 필요하다. 왜냐하면 학습자가 개념을 학습하고 개념들을 구조화함에 있어 학습의 효율성, 오개념 형성에 영향을 미칠 수 있기 때문에 학습 순서는 매우 중요하다. 실제로 학습자들은 현재 학습 해야 할 프로그래밍 제어구조의 개념 바로 직전에 배웠던 개념의 영향을 받아 오개념을 형성하거나 시행착오를 겪는 경우가 많다.

프로그래밍 교육을 위한 교수 학습은 문제 해결에 필요한 개념을 학습하고 개념을 적용하는 원리를 익히는 데에 중점을 두어야 한다. 이를 바탕으로 학습자들이 어려워하는 점은 무엇인지, 그 이유는 무엇인지에 대해 파악함으로써 학습의 효과를 높일 수 있는 전략을 수립할 수 있다. 이러한 분석은 학습 과정에서 오개념이 형성되는 것을 예방할 수 있으며, 학습 동기와 흥미를 유지시키고 프로그래밍 학습이 지속되기 위한 적절한 교육 처방이 가능하도록 한다. 왜냐하면 프로그래밍은 언어의 엄격한 문법 학습에 더하여 고도의 추상화 능력을 요구하여 학습자에게 인지 부담을 증가시켜 학습자가 중도에 포기하도록 하는 결과를 낳는 경우가 많기 때문이다.

3. 프로그래밍 제어구조 오류 원인 분석

3.1 프로그래밍 제어구조 학습에서의 경험 분석

프로그래밍 제어구조 개념을 순차, 선택, 반복, 함수로 구분하고 학습자가 경험한 어려움과 그 원인을 분석하였다[1,2,3,6].

(1) 순차: 순차는 문제 해결에 필요한 명령을 구체적인 순서에 따라 나열하는 개념이다. 순서가 올바르지 않으면 잘못된 결과를 산출하게 된다. 따라서 순서도에 표현하는 명령어들을 구체적이고 실행 가능한 형태로 기술하여야 한다. 순차에서 학습자가 경험한 어려움은 다음과 같이 컴퓨터가 폰노이만 방식으로 순차적으로 문제를 해결한다는 관점의 이해 부족에 그 원인이 있었다. ①학습자들은 학습자들이 문제 해결 절차를 체계적으로 작성하지 않고 순간적인 사고에 의존, ②큰 문제를 작은 문제로 분해하여 해결하고 합치는 과정에서 필요한 블록을 생략, ③현재까지 작성된 프로그램의 결과가 다음 프로그래밍에 향을 미친다는 사실을 고려하지 않음, ④컴퓨터가 문제를 해결하는 관점에서 사고하지 못함, ⑤필요한 자료 수집이 이루어지지 못함 등이다.

(2) 선택: 선택은 조건식의 가부에 근거한 비교판단으로 특정 동작이 발생하도록 분기가 이루어져야 할 때 사용하는 개념이다. 기본적으로 '만약'과 '만약, 그렇지 않다면'의 두 가지 블록을 포함한다. 잘못된 분기는 예기치 못한 결과를 산출하므로 의미에 합당하게 명확하게 분지의 흐름을 기술할 수 있도록

선택을 결정하는 조건에 대해 좀 더 자세하고 다양한 예제를 활용한 설명이 필요하다. 선택에서 학습자가 경험한 어려움은 다음과 같이 논리구조의 이해부족에 그 원인이 있었다. ①각 조건 내에 삽입해야할 명령문 블록을 찾아내지 못함, ②조건식 범위의 중첩, ③다중 선택문에서 각 조건식 순서의 뒤바뀜, ④문제 상황에 부적절한 명령문 선택, ⑤조건식의 올바르지 않은 표현 등이다.

(3) 반복: 반복은 문제 해결 과정에서 연속으로 나타나는 패턴을 인식하고 이 패턴이 반복 실행되도록 구성하는 개념이다. 반복은 대체로 '횟수', '카운트'와 같은 횟수가 지정된 반복과 '할 때까지', '인 동안'과 같은 조건이 만족할 때까지 반복하는 두 경우로 나뉜다. 반복에서 학습자가 경험한 어려움은 다음과 같이 반복 구조의 이해부족에 그 원인이 있었다. 은 원인에 따른 결과이다. ①반복되어야 할 블록 순차적 나열, ②반복문 내에 불필요한 블록을 삽입, ③반복 횟수 입력 오류, ④문제 해결에 적합하지 않은 반복 블록 선택, ④함수 호출 방식중첩 구조를 만들지 못함 등이다.

(4) 함수: 함수는 한 동작이 다른 동작의 발생을 위해 프로그램의 실행 제어권을 넘기는 개념이다. 함수는 문제 해결 과정 전반에 걸쳐 반복 사용될 수 있는 모듈로 주프로그램과 별도로 작성된다. 함수에서 학습자가 경험한 어려움은 다음과 같이 함수에 대한 개념과 사용 방법에 한 이해가 충분히 이루어지지 않은 점, 문제 해결 과정의 전체 맥락에서 함수에 포함되어야 하는 패턴을 정확하게 인지하지 못한 데 그 원인이 있다. ①함수의 호출 위치 파악, ②매개변수 삽입 위치 파악, ③함수에 포함되는 블록을 선정, ④함수 호출 방식 등이 있다.

위와 같이 학습자가 제어구조에서 느끼는 어려움의 공통된 특징은 학습자들이 제어구조의 기본 개념과 사용 방법에 대한 이해가 충분히 이루어지지 않은 점, 문제 해결 과정의 전체 맥락에서 제어구조에 포함되어야 하는 패턴을 정확하게 인지하지 못한 점, 중첩의 구조화에 대한 충분한 파악 부족, 시행착오적 접근 방법 등에 그 원인이 있음을 알 수 있었다.

3.2 프로그래밍 제어구조의 오류 유형 분석

대상 학생들에게 순서도를 활용한 프로그래밍 제어 구조를 학습시킨 후 이를 평가하여 나타난 오류 유형을 분석하였다[3]. Table 1은 네 가지 프로그래밍 개념에 따른 대표 오류 유형을 표로 정리한 것이다.

4. 프로그래밍 제어구조 오개념 순서도 분석 및 적용

4.1 순차구조

4.1.1 필요한 명령어를 생략하는 경우

필요한 명령어를 생략하는 경우는 두 변수의 값을 서로 바꾸는 프로그램을 예로 들 수 있다. 초보자의 경우 두 값을 교환하는 스와핑 알고리즘을 Fig. 1의 (a)와 같이 작성할 수 있다. 하지만 이의 결과는 의도

Table 1. Representative error types according to four programming concepts

구 조	오류 유형
sequential	① the required command is omitted ② add not required statement ③ the commands are not listed in order
selection	① the condition of the selection structure is misrepresented ② a command that is executed according to the true and false condition is omitted ③ an unnecessary command is added to the condition and described
iteration	① set the repeat count incorrectly ② loops that are not appropriate for problem solving ③ lack of understanding of nested structure
function	① function's parameters are set incorrectly. ② missing function prototype declaration ③ lack of understanding how to call function

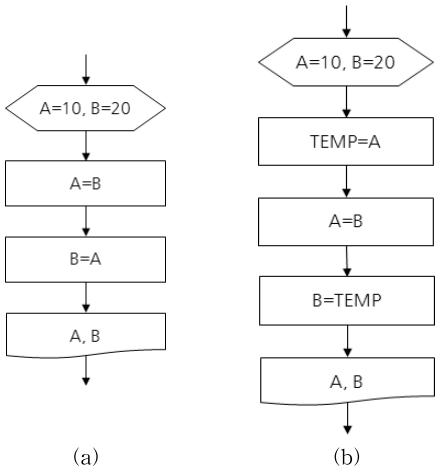

Fig. 1. The correct answer flowchart of Exchange two variables.

한 대로 나오지 않고 동일한 값을 두 번 출력하는 결과를 만든다.

두 변수의 값을 서로 바꾸기 위하여 Fig. 1의 (b)와 같이 TEMP라는 임시 변수를 추가해서 총 3개의 변수로 값을 교환할 수 있게 된다.

4.1.2 명령어들을 순서에 맞게 바르게 배열하지 못하는 경우

명령어들을 순서에 맞게 바르게 배열하지 못하는 경우는 두 수를 입력받은 뒤 그 합을 계산하여 출력하는 프로그램에서 흔히 발생하는 오류 중 하나이다.

Fig. 2의 (a)는 논리적인 오류가 존재하지 않는다. 하지만 'SUM=A+B'의 문장과 A와 B의 값을 입력받는 명령은 별개의 문장으로 동작하기 때문에 A와 B의 값을 입력받았다 하더라도 SUM에는 입력되기 이전의 A와 B의 값의 합이 저장되어 결국 쓰레기 값이 출력되는 오류를 발생한다.

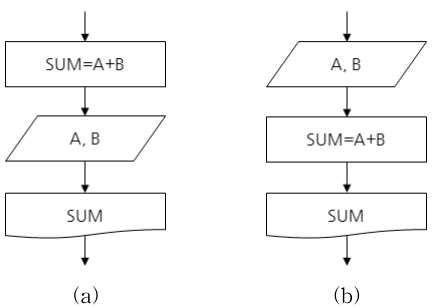

Fig. 2. The correct answer flowchart of Arithmetic expression.

4.2 선택구조

4.2.1 조건에 불필요한 명령어가 추가되어 기술된 경우

조건을 제대로 이해하지 못한 상태에서 프로그램을 작성하게 될 때 불필요한 판단기호를 추가하게 되어 문제에서 제시하는 논리를 제대로 수행하지 못하게 된다.

Fig. 3은 사용자로부터 정수를 입력받아서 90점 이상을 경우 'A', 80점 이상 90점 미만의 경우 'B', 80점 미만의 경우 'C'를 출력하는 학점 계산 프로그램의 순서도의 일부이다.

Fig. 3에서 S는 사용자로부터 입력받은 정수이다. Fig. 3의 (a)의 경우 80점 미만의 경우를 처리하기 위해 판단기호를 추가하였다. 이 경우 논리적으로는 틀리지 않아서 결과는 같지만 프로그램은 순차적으로 처리되므로 첫 번째와 두 번째 판단 기호를 No로 지나왔을 경우 이 값은 무조건 80 미만의 수가 될 수밖에 없다. 따라서 80 미만이 거짓일 경우가 발생할 수 없으므로 이는 불필요한 명령어가 된다. 불필요한 명령을 제거하면 Fig. 3의 (b)처럼 된다.

이를 잘 해결하기 위해서는 주어진 조건을 잘 파악하여 최적의 흐름을 작성할 수 있도록 한다.

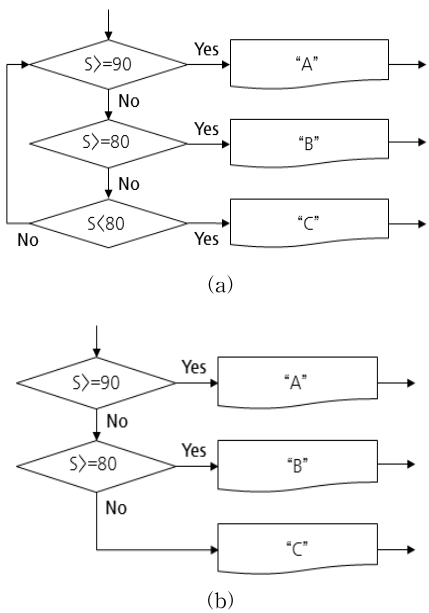

Fig. 3. The correct answer flowchart of Multi Control.

4.2.2 조건식을 잘못 세우는 경우

선택구조에서 조건식은 어떤 문장을 실행하는지를 결정하므로 그 역할이 매우 중요하다. 하지만 초보자들은 조건식을 작성하는 데 많은 어려움을 가지고 있다. 다음은 조건 변수가 10일 경우 '안녕하세요'를 출력하는 프로그램의 정오답 순서도 중 일부이다.

Fig. 4의 (a)에서는 조건식이 AAA=10으로 되어 있다. 프로그램을 실행시켜 볼 경우 '안녕하세요'가 정상적으로 출력되는 것을 확인할 수 있다. 하지만 이는 연산자 우선순위에 따라 변수 AAA에 10이 대입된 상태에서 조건식이 변수 AAA를 참조하게 되는 데, 이 때 참조된 변수 AAA의 값이 0이 아니므로 조건식은 무조건 참이 된다. 따라서 (a)의 조건식을 (b)와 같이 원래의 의도대로 비교연산자를 사용하여 오류를 수정할 수 있다.

4.3 반복구조

4.3.1 반복 횟수를 정하는 조건을 잘못 설정한 경우

반복문은 명령어를 반복 수행할 때 사용되며 종료조건이 존재한다. 종료조건이 잘못 설정되어 있을 경우 반복문을 무시하고 그냥 넘어가거나 심지어는 반복문을 벗어나지 못해 계속 반복되는 무한 루프가 될 수 있다. 따라서 반복구조를 설계할 때에는 반복문의 종료 조건이 되는 인자를 잘 다루는 것이 중요하다.

Fig. 5는 1부터 N까지의 자연수의 합이 100을 넘지 않는 N의 최대값과 그때의 합을 구하는 프로그램의 정오답 순서도이다.

(a)는 문제가 없어 보이지만, 실제 결과는 SUM과 N이 각각 105와 15가 나온다. 이는 우선 반복되는 문장에서 SUM은 이미 N이 추가된 상태이기 때문에 해당 반복문은 합이 100을 초과하는 최초 1회까지 반복된다. 또한 N 역시 반복문에서 1씩 더해지기 때문에 마지막에 1을 빼주어야 원하는 결과를 얻을 수 있다. 따라서 오답 순서도인 (a)의 이러한 문제점들을 수정하면 (b)와 같이 되며, 이 때의 결과는 각각 91과 13이 출력된다.

4.3.2 문제 해결에 적합하지 않은 반복문을 사용하는 경우

반복구조에서 while구조가 처음에 반복 종료 조건을 비교한 뒤 그 다음 반복될 명령문을 수행한다면 do-while구조는 우선 반복될 명령문을 수행한 뒤 그

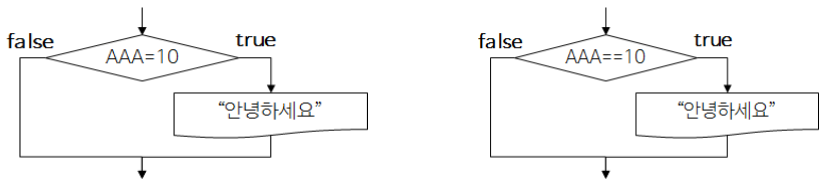

Fig. 4. The correct answer flowchart of If condition.

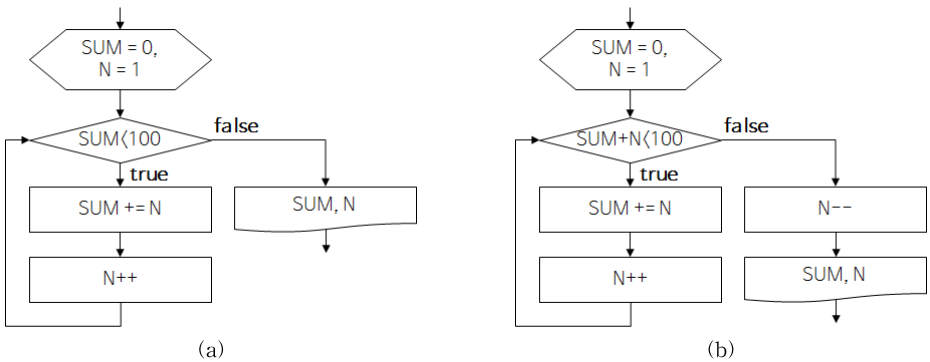

Fig. 5. The correct answer flowchart of While loop.

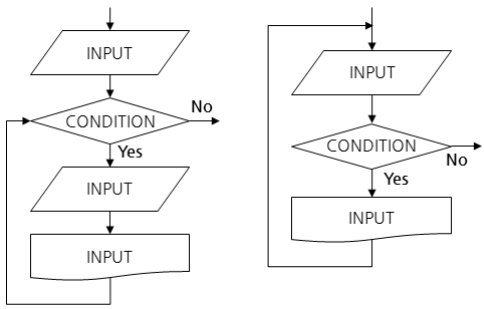

Fig. 6. The correct answer flowchart of Do while loop.

다음 조건식을 비교한다. while구조의 경우 조건식이 시작부터 거짓일 경우 반복될 명령문은 무시된 채 다음 문장으로 넘어가게 되지만 do-while구조는 조건식에 상관없이 반복될 명령문을 무조건 한 번 수행한다는 특징이 있다. 따라서 do-while구조는 주로 입력에 대한 처리나 메뉴 등 특수한 목적에 사용할 때 유용하다. 하지만 이러한 차이점을 간과하여 문제 해결에 적합하지 않은 구조를 사용할 경우 의도했던 결과에서 멀어질 수 있다.

Fig. 6은 사용자로부터 정수를 입력받아서 0부터 100 사이일 경우 그 값을 출력한 뒤 계속 입력받도록 하고 범위 이외의 값일 경우 반복문을 종료하게 되는 프로그램의 순서도 일부이다. 여기서 CONDITION은 0 <= INPUT && INPUT <= 100과 같다.

Fig. 6의 (a)를 보면 문제를 while구조로 구현하기 위해 입력을 최초로 1회 받은 뒤 그 뒤로는 반복문 안에서 처리하도록 되어있다. 논리적으로는 틀리지 않았지만 이 역시 최적화된 알고리즘으로 볼 수 없다. 프로그램의 흐름을 위해 최초 1회 입력받을 때 반복을 사용한 것이 아닌 순차 논리를 사용하였기 때문이다. 이 때 do-while구조를 사용하여 구현하면 (b)와 같이 온전히 반복문으로 구현이 가능해진다. (a)와 비교해보았을 때 좀 더 간결하게 표현할 수 있음을 확인할 수 있다. 따라서 문제에 따라서 어떠한 반복구조를 사용할 것인가를 결정하는 것도 효과적인 알고리즘을 구현하는 데에 큰 도움이 된다.

4.4 함수

4.4.1 함수의 매개변수를 잘못 설정한 경우

Fig. 7의 (a)는 swap() 함수의 매개변수로 정수형 변수를 참조하고 있다. 이럴 경우 출력되는 결과는 의도한 바와 달리 10과 20으로 그대로 출력된다. 이는 함수를 호출할 때 매개변수로 입력된 두 변수 A와 B는 함수 안에서 사용된 변수와 다르기 때문이다. 이를 해결하기 위하여는 우선 함수의 매개변수의 자료형을 int *형으로 선언한 뒤, 매개변수를 참조할 때 &A와 같이 주소연산자를 적음으로써 함수 내부에서도 매개변수의 값을 직접 참조할 수 있도록 한다. (b)는 원래 의도대로 수정한 순서도이고, 실행할 경우 (a)와는 달리 A와 B의 값이 바뀌어 출력된다.

5. 적용사례

S대학교 소프트웨어학과 1학년 프로그래밍기초 교과목을 수강하는 학생들의 컴퓨팅 사고 향상을 위해 순서도 작성에 중점을 두어 설계한 순서도작성 워크북을 교수학습자료로 개발하여 활용하였다. 각

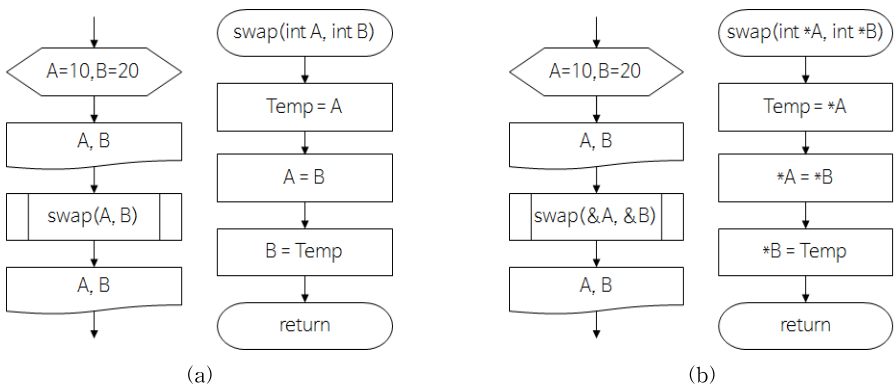

Fig. 7. The correct answer flowchart of Parameters Passing to function.

제어구조 훈련을 위하여 문제를 제시하고 문제 해결 과정을 5단계로 나누어 활동 기록을 하도록 하였다. 다음은 초보 프로그래머가 스스로 오류를 찾고 개선할 수 있도록 설계한 워크북을 사용한 학생들의 반복 제어구조 워크북 활동의 일부를 소개한다.

5.1 반복제어구조 워크북 활동

워크북은 각 제어 구조에 대한 설명과 이에 대한 예제 순서도, 그리고 이를 응용하여 학습자들이 직접 풀도록 하는 연습 문제로 구성되어 있다. Fig. 8은 대상 학습자의 워크북 연습 문제 풀이를 스캔한 것이다. 연습 문제는 다음과 같이 문제가 주어지며 순서대로 ①변수 설계, ②문제 요구사항 파악, ③오답 순서도로부터 정답 순서도 도출, ④코딩, ⑤유제문제 개발의 5단계로 해결할 수 있도록 하고 있다.

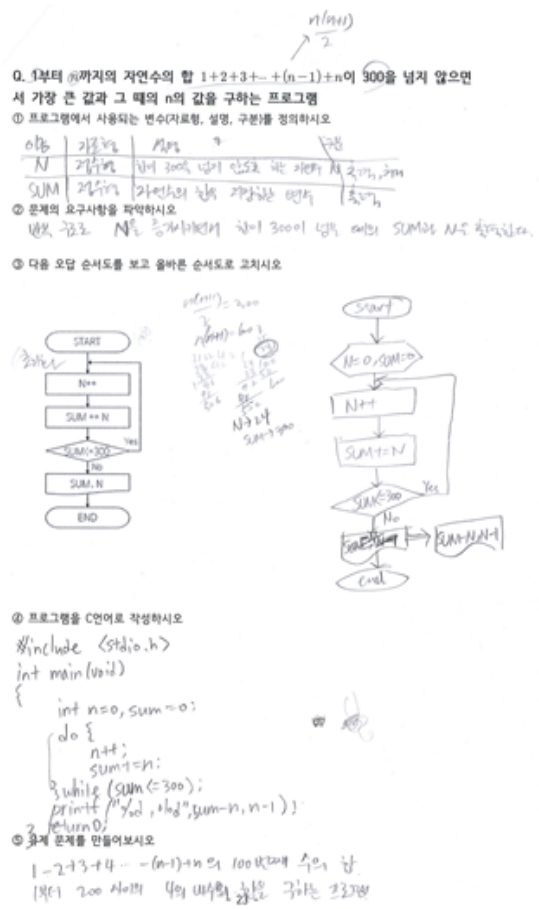

Fig. 8. activity of workbook about Iteration control structure.

5.2 학습경험

수업 적용 사례로 워크북을 사용한 학생들의 활동의 일부를 소개하였다. 학습경험을 정리하면 다음과 같다. 첫째, 프로그래밍기초 교육 시 초보 프로그래머가 스스로 오류를 찾고 개선할 수 있도록 설계한 워크북을 사용한 학습 활동을 통하여, 프로그래밍 제어구조 개념에 대한 정확한 이해와 내면화, 정교한 알고리즘 작성을 위하여 오개념 순서도 활용이 초보 프로그래머에게 효과적이었다. 둘째, 프로그래밍기초 교육 시 초보자가 실생활의 다양한 문제를 발굴하고 이해한 후, 스스로 프로그래밍 개념을 활용할 수 있는 문제를 만들고, 해결방안을 찾고(변수정의, 문제의 요구사항 파악, 순서도 작성) 해결을 할 수 있는 워크북을 활용한 학습경험을 하였다. 그 결과 간단한 알고리즘을 직접 설계하여 프로그래밍을 체험함으로써 프로그래밍 초보자가 창의적으로 문제를 해결할 수 있도록 하는 컴퓨팅 사고력을 길러주는 데 집중할 수 있었다.

6. 결 론

본 연구에서는 아이디어를 체계적으로 구체화하는 컴퓨팅사고를 향상시키는 훈련을 위하여 대학교 1학년 C프로그래밍기초 수업을 순서도작성 워크북을 활용하여 진행하였다. 특히 프로그래밍 기초 교육에서 초보 학습자들이 어려워하고 오류를 발생시키는 부분은 어디인지, 그 이유는 무엇인지에 대하여 프로그래밍 제어구조 개념을 중심으로 관찰하고 분석하였다. 또한 학습자들이 논리적인 오류를 발생시키는 오개념을 가시적인 순서도를 사용하여 오개념 순서도를 정답 순서도로 수정하는 과정에서 스스로 잘못 이해한 부분에 대한 자각과 오류 등을 직면함으로써 그것을 극복하는 과정에서 프로그래밍에 대한 흥미와 성취를 느끼는 학습 경험을 하였다. 따라서 워크북을 사용한 오개념 순서도 활용 실습을 학생들이 스스로 오류의 원인을 확인하고 수정하면서 제어구조 개념에 대한 정확한 이해와 내면화를 통한 컴퓨팅 사고력 향상에 집중할 수 있는 학습 방안으로 제시한다.

REFERENCE

[1] M. Piteria and C. Costa, "Learning Computer

Programming: Study of Difficulties in Learning Programming," *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, pp. 75-80, 2013.

[2] J. Choi and Y. Lee, "The Analysis of Learners' Difficulties in Programming Learning," *Journal of Korean Association of Computer Education*, Vol. 17, No. 5, pp. 89-98, 2014.

[3] H. Chae, "Analysis on Types of Errors in Learning about Control Structures of Programming Using Flowchart," *Journal of Korean Association of Computer Education*, Vol. 19, No. 1, pp. 101-109, 2016.

[4] S.H. Lee, *Design and Implementation of the Education System of C Programming Using Flowchart*, Master's Thesis of KookMin University, 2005.

[5] K.W. Park, "Programming Language Learning System Using Basic Algorithm," *Journal of The Korea Institute of Electronic Communication Sciences*, Vol. 5, No. 1, pp. 66-73, 2010.

[6] M. Choi, "Supplementary Process through Error Analysis of Learner Exploring Principles of Textbook Development," *The Journal of Curriculum Studies*, Vol. 21, No. 3, pp. 243-264, 2003.

[7] Y. Choi, "Design and Application of Term Project Model for Game Mathematics in Flipped Learning Environments," *Journal of Korea Multimedia Society,* Vol. 20, No. 7, pp. 1102-1112, 2017.

[8] I. Chen, *Problem Solving and Computational Thinking*, Infinity Books, Goyang, 2017.

[9] S.H. Kim, *Data-driven World the Beginning of Informational Thinking,* Korea University Press, Seoul, 2014.

[10] S.H. Choi, G.C. Nam, and H.A. Ryu, "Development of Instruction Materials for Underachieving Students to Correction of Misconception," *Mathematics Education Studies*, Vol. 23, No. 2, pp. 117-133, 2013.

최 영 미

1979년 이화여자대학교 수학과 (이학사)
1981년 이화여자대학교 전산학 전공(이학석사)
1993년 아주대학교 컴퓨터공학과 (공학박사)

1994년~현재 성결대학교 미디어소프트웨어학부 교수
관심분야 : 지능형교수시스템, 게임프로그래밍, AR게임