Journal of The Korean Association
of Information Education Vol. 21, No. 6,
December 2017, pp. 629-638

http://dx.doi.org/10.14352/jkaie.2017.21.6.629

컴퓨팅사고력 향상을 위한 알고리즘 기반의 교수학습방법 개발

임서은* · 정영식**

군포초등학교* · 전주교육대학교 컴퓨터교육과**

요 약

정보 교과에서 컴퓨팅사고력을 통해 해결하고자 하는 정보 교과 문제의 정의와 특성을 알아보고, 정보 교과 문제의 유형과 사례, 정보 교과 문제 해결을 위한 교수학습방법 등에 대하여 탐구하였다. 정보 교과 문제에 대해 알아보기 전, 먼저 타 교과에서 문제의 정의와 문제의 종류, 문제 해결의 중요성에 대하 조사하였다. 이러한 조사를 바탕으로 정보 교과는 컴퓨팅사고력과 컴퓨팅 파워를 통해 비구조화된 문제를 해결할 수 있고, 그러한 문제에는 카운팅 문제, 결정 문제, 검색 문제, 최적화 문제 등이 있다는 것을 알게 되었다. 이러한 컴퓨팅 과학 문제를 해결하기 위해 알고리즘 기반의 교수학습 방법, 즉, 탐색기반 교수학습방법과 관계기반 교수학습방법을 제안하였다. 선생님들은 문제의 특성에 따라 두 가지 방법을 활용함으로써, 학생들이 문제를 해결하는 과정에서 추상화, 자동화, 일반화 등의 컴퓨팅사고력을 향상시킬 수 있다.

키워드 : 컴퓨팅사고력, 정보 교과 문제, 탐색기반 설계, 관계기반 설계, 추상화, 자동화, 일반화

Development of Teaching and Learning Methods Based on Algorithms for Improving Computational Thinking

Seoeun Lim*, Youngsik Jeong**

Kunpo Elementary School*, Jeonju National Univ. of Education**

ABSTRACT

This study investigated the definition and characteristics of computer science problem to be solved through computational thinking. It also explored types and cases of both computer science problems and teaching learning methods to solve computer science problems. Before studying computer science problems, I examined the definition, type, and the importance of problem solving in other subjects. Based on this research, We found that informatics can solve ill-structured problems through computational thinking and the power of computing. This includes counting, decision, retrieval, and optimization problems. Teachers can improve their students' skills in computational thinking, particularly as related to abstraction, automation, and generalization, by choosing the appropriate teaching and learning method or based on the characteristics of the problem.

Keywords : computational thinking, computer science problems, search-based methods, relationship-based methods, abstraction, automation, generalization

교신저자 : 정영식(전주교육대학교)
논문투고 : 2017-10-04
논문심사 : 2017-10-11
심사완료 : 2017-12-11

1. 연구의 필요성 및 목적

컴퓨팅사고력은 컴퓨터의 능력을 바탕으로 문제를 해결하기 위해 깊은 사고를 융합하는 능력이다. 따라서 문제를 효율적으로 해결할 수 있는 방법을 찾아내기 위해 깊은 사고를 통해 문제를 분석하고, 분해하고, 패턴을 찾아내고, 추상화한 후 그것을 알고리즘으로 연결해야 한다. 그 후 프로그래밍을 통해 컴퓨터를 작동시켜 구현하게 된다[7].

여러 학자 및 학회에서 제시한 컴퓨팅사고력의 구성요소를 고찰하여 한국정보교육학회의 '정보과 교육과정 표준 모델 개발'이 정리한 바에 따르면[5], <Table 1>과 같이 컴퓨팅 사고력 영역의 첫 단계에 문제 분석에 대한 내용이 제시되어 있다. 즉, '문제 분석'은 컴퓨팅사고력의 영역의 출발점이자 문제 해결의 과정에서 매우 중요한 단계이다.

모든 문제의 해결은 결국 '문제'에서부터 시작하며, '문제'에 대한 이해의 과정을 거친다. 그러나 컴퓨팅사고력과 관련해 '문제'에 관한 내용은 추상화 나 자동화 등 영역 내 다른 개념들에 비해 매우 부실하게 다뤄져왔다. 이 때문에 정보과에서 컴퓨팅을 통해 해결할 수 있는 문제가 무엇인지에 대한 정의가 명확하지 않았으며, 다른 도구가 아닌 컴퓨팅을 통해서 문제를 해결하는 것의 장점에 대한 설득력도 부족했다.

더불어 알고리즘 기반의 문제해결능력을 향상시키기 위한 교수학습방법도 체계적으로 구축되어 있지 않다. 한국교육개발원과 한국교육학술정보원은 'SW교육 교수학습 모형 개발 연구'[12]에서 SW교육 교수학습 모델로 시연중심모델(DMM 모델), 재구성중심모델(UMC 모델), 개발중심모델(DDD 모델), 디자인중심모델(NDIS 모델), CT 요소 중심모델(DPAA(P)모델)을 제시하였다. 제시한 교수학습모델들은 프로그래밍 및 자동화는 강조하고 있지만, 컴퓨팅사고력의 핵심인 추상화와 관련된 알고리즘 기반의 문제 해결력은 강조하고 있지 않다. 즉, 해당 교수학습모델들로 프로그래밍 방법은 학습할 수 있으나, 실생활 문제를 해결하기 위한 알고리즘 기반의 사고력을 증진시키기는 한계가 있다.

따라서 본 연구는 그 동안 중요하게 논의되어 오지 않았던, 컴퓨터과학에서 컴퓨팅을 통해 해결하고자 하는 '문제'란 무엇인가에 대한 정의와 함께, 컴퓨팅사고력을 신장시킬 수 있도록 알고리즘을 기반의 교수학습방법을 제시하였다.

<Table 1> Area of Computational Thinking

Area	Element	Description
Problem analysis	Understanding problem	Identify the current status and target status of problem.
	Problem definition	Identifying problem situations and expressing problem
	Problem decomposition	Decomposing complex problem
Data analysis	Data collecting	Collecting data for solving problem
	Data representation	After analyzing the data, it is expressed as words, texts, pictures, etc.
	Data structure	Structuring analysis results by making tables or graphs
Abstraction	Pattern analysis	Founding recurring elements for solving the problem
	Logical reasoning	Predicting the results of algorithm and program
	Modelling	Eliminating unnecessary elements for solving the problem
	Abstraction	Simplifying and representing the procedures and methods required for solving the problem
	Algorithm	Express the problem-solving process in diagrams, flowcharts, and pseudocode
Automation	Programming	Write a program to solve the problem
	Debugging	Find and fix the program errors
	Automation	Solving the problem through programming
Generalization	Optimization	Improve with better troubleshooting
	Evaluation	Determine whether the problem is solved effectively for the purpose
	Applying	Apply the solving methods to similar problems

구체적인 연구의 방법 및 절차는 다음과 같다.

첫째, 타 교과의 문제 이해와 해결 방법에 대한 선행연구를 분석한다.

둘째, 컴퓨터과학 문제의 이해와 특징, 유형에 대한 선행 연구를 분석한다.

셋째, 컴퓨팅사고력 향상을 위한 알고리즘 기반 교수학습방법을 제안한다.

2. 문제 이해와 해결

컴퓨터과학에서 컴퓨팅을 통해 해결하고자 하는 문제에 대해서 이해하고, 그것을 해결하기 위한 방법을 가르치기 위한 교수학습방법을 개발하기 위해 먼저 타 교과에서 다루는 문제의 개념과 유형, 그것을 해결하기 위한 방법을 분석하였다.

2.1 문제의 이해

Jonsassen(1997)은 문제를 다양성을 기준으로 크게 구조화된(well-structured) 문제와 비구조화된(ill-structured) 문제로 구분하였다[11].

첫째, 구조화된(well-structured) 문제는 문제 진술에 문제에 대한 모든 요소가 제시되고, 문제 진술에 구체화된 문제의 범위가 나타나며, 수렴적인 답을 가지는 문제이다. 정보의 취사선택과 문제의 모든 조건 사이의 관련성이 이미 알려져 있거나, 개연적인 곳에서 알기 쉽고 이해하기 쉬운 해를 가진다[25].

둘째, 비구조화된 문제(ill-structured)는 상황의 측면들이 구체화되어 있지 않고(Chi & Glasr, 1985), 보다 현실적이고 실제적인 상황을 바탕으로 하고 개방성을 지니고 있으며, 단순하기보다는 복잡한 상황이 제시되는 문제이다. 또한, 실제성과 복잡성 및 개방성을 바탕으로 특정한 실세계의 맥락이 구체화되지 않은 문제 상황으로 만들어져 학습자로 하여금 스스로 문제를 해석하고 해결 방법을 모색하여 적절한 해를 찾도록 한다[9].

모든 문제가 반드시 일정한 유형에 따라 완전하게 분류되는 것은 아니며, 오히려 어떤 문제들은 여러 유형의 특성을 함께 가지고 있는 경우도 있다. 그러나 교사는 문제의 종류와 특성을 연구하고 이해하여 다양한 유형의 문제들을 학생들에게 제공하여 줌으로써, 학생들의 사고 능력을 신장시킬 수 있다[13].

타 교과에서 다루는 문제를 이해하기 위하여, 정보 교과와 가장 관련이 깊은 수학과 과학에서 다루는 문제를 중심으로 분석하였다. 즉, 수학에서 다루는 문제, 과학에서 다루는 문제에의 개념과 유형에 대하여 선행 연구를 분석하였다.

2.1.1 수학 문제

수학과는 정보 교과와 마찬가지로 알고리즘을 문제해결을 위한 구체적이고 체계적인 절차를 만들어내는 방법을 강조하고 있다[4]. 수학에서의 문제는 처음에는 정확한 해법을 알지 못할 수 있으나, 해법의 결과를 요하는 개인 또는 단체에게 부과된 양적인 장면(quantitative situation)이라 말할 수 있다. 수학적 문제는 식이나 수학적 사실들과 같이 주어진 것(givens)이 있으며, 그것을 다른 식으로 전환하는 연산(operating)이 있고, 마지막에 발견하려는 목표(terminal goal)가 있다[13]. 이러한 수학적 문제를 유형별로 제시하면 다음과 같이 정형 문제, 비정형 문제, 실생활 문제로 구분할 수 있다[16].

첫째, '정형 문제'는 이미 제시된 일반적인 알고리즘을 회상하고 그 변수에 특별한 수치를 대입하여 해결할 수 있는 문제로, 이미 알려진 해법에 따라 해결할 수 있다.

둘째, '비정형 문제'는 문제해결 전략의 사용이나 공식화되지 않은 독자적인 해결 방법을 요구하는 진정한 의미로서 문제로, 공식화되지 않은 독자적인 해결 방법을 요구하므로, 해결 자체보다는 그 해결 과정이 더욱 중요하다.

셋째, '실생활 문제'는 취급하는 소재가 학생의 실생활에서 얻어지며 그 해결법은 정형적, 비정형적 방안을 모두 요구하는 문제로, 학생들로 하여금 수학적 지식과 문제해결 전략을 활용하여 실생활에서 일어날 수 있는 문제들을 해결해보게 한다.

2.1.2 과학 문제

정보 교과는 과학 교과도 밀접한 관련성이 있다. 정보 교과에서 다루는 정보 및 전산의 이론적 기초와 그것을 구현하고 응용하는 기술은 과학적 근거를 기반으로 하기 때문이다[22]. 과학에서의 문제는 '이론과 양립하지 않는 관찰 혹은 실험 간의 불일치'이다. 불일치란 이론과 데이터 사이의 불일치, 이론들 사이의 불일치를 의미한다[10]. 과학 교과에서는 인식된 문제가 탐구의 시작이며, 탐구의 과정이란 가설을 설정하고 그것을 입증하는 과정이다[19].

과학 교과에서 다루는 문제인 탐구 문제는 5가지 유형이 있다[10].

첫째, '새로운 결과 탐구 문제'는 측정 방법이나 범위, 실험 상황 등을 변화시키면 어떠한 새로운 결과가 나올 것인지 알아내는 문제이다.

둘째, '관계 탐구 문제'는 주어진 실험 상황에서 얻은 결과들 사이에 어떠한 관계가 있는지 알아내는 문제이다.

셋째, '왜-어떻게 탐구 문제'는 어떤 결과가 나오게 된 인과적 변인을 찾아보거나 어떠한 과정을 통해 그러한 결과가 나오게 되었는지 알아내는 문제이다.

넷째, '적용 탐구문제'는 주어진 상황이나 결과가 다른 현상이나 상황에서 어떻게 적용될 수 있는지 알아내는 문제이다.

다섯째, '실험방법 탐구문제'는 구체적인 측정 방법이나 실험방법을 개발하거나 알아내는 문제이다.

2.1.3 컴퓨터과학 문제

컴퓨터과학은 구조화된(well-structured) 문제는 물론, 비구조화된(ill-structured) 문제를 가장 효율적으로 해결할 수 있는 학문이다. 비구조화된 문제는 학생들로 하여금 정보를 재조직하고 사고를 명확히 하여 새로운 이해에 이르게 하고, 문제 해결을 위한 대안들을 평가하여 가장 적합한 해결 방안을 찾을 수 있어 컴퓨팅시스템을 활용할 경우 보다 효율적으로 해결될 수 있다.

컴퓨팅시스템을 통해 해결하고자 하는 컴퓨터과학 문제는 비구조화된(ill-structured) 문제이면서, 프로그래밍 언어와 연관된 컴퓨팅을 활용하여 해결할 수 있는 문제이어야 한다. 컴퓨팅 문제에서 우리는 기호 {0,1}을 부호화된 것으로 가정하고, 이를 통해 몇 가지 조건을 만족하는 해를 출력할 수 있다. 즉, 컴퓨팅 문제에서 출력은 주어진 조건과 입력을 만족시켜야 한다[14]. 컴퓨팅을 통해 해결 가능한 정보 교과 문제의 특성을 살펴보면 다음과 같다[15].

첫째, 실제성(authenticity)으로, 일상생활과 학교 밖의 현실 상황을 묘사한 과제 및 문제가 일치하는 것(Fitzpatrick & Morrison, 1971)을 의미한다.

둘째, 복잡성(complexity)으로, 문제를 해결하는데 요구되는 개념, 원리, 법칙 혹은 이들이 어떻게 조직되는지에 대하여 불확실성을 의미한다.

셋째, 개방성(openness)으로, 문제 해결에 관하여 여러 개의 평가 준거가 존재하며, 적절한 행동을 위한 명확한 수단이 제시되지 않는 것을 의미한다.

컴퓨터과학의 문제 중 대표적인 유형에는 의사결정의 문제, 카운팅의 문제, 검색의 문제, 최적화의 문제가 있다. 물론, 컴퓨터과학의 문제가 이 4가지 유형으로 모두 분류되는 것은 아니다. 사용자가 문제를 바라보는 관점에 따라 달라질 수 있고, 대부분의 실생활 문제들은 4가지 유형이 혼재되어 있다.

첫째, 의사결정의 문제는 입력 값에 대한 '예' 또는 '아니오'로만 출력하는 것으로서 다음과 같이 소수인지를 확인하는 문제가 대표적이다[23]. 의사결정의 문제는 일반적으로 대답이 '예'인 모든 사례들의 집합으로 표현된다. 즉, 소수는 {2, 3, 5, 7, 11, ...}같은 무한 집합으로 나타낼 수 있으며, 이 중에 특정 숫자가 해당 집합에 존재하는지를 판단하여 소수 여부를 확인할 수 있다. 컴퓨팅으로 해결이 가능한 의사결정 문제는 가치판단의 문제가 아니라 수학이나 과학의 문제와 연계되며, 가장 간단한 문제는 단순 분기 구조이다. 입력된 자료가 특정 속성을 충족시키는지 여부를 확인한 후 오직 2가지로만 응답하면 되기 때문에 비교적 쉬운 문제이지만, 중요한 이론을 만드는 데에 활용될 수 있다[3].

둘째, 카운팅의 문제는 가능한 해결책들의 수를 찾는 것으로, 주어진 수식이나 조건을 만족시키는 해결 방법이 몇 개가 있는지를 찾는 것이다[1]. 예를 들면, 자연수 n의 약수가 몇 개인지를 구하는 것이다[21]. 컴퓨팅을 활용하면 아주 간단한 작업을 통해 특정 자연수의 약수 개수를 구할 수 있다. 자연수 n을 나누어떨어지게 하는 자연수를 1부터 n까지 컴퓨터가 일일이 탐색해보게 하는 프로그램을 작성하면 된다. 컴퓨터는 매우 짧은 시간에 빠른 속도로 탐색할 수 있기 때문에 카운팅 문제를 효율적으로 해결할 수 있다.

셋째, 검색의 문제는 원하는 해결 방법이 있는지 뿐만 아니라, 실제 해결 방법이었는지 확인하는 것이다. 예를 들면, n개의 정수를 배열로 저장해 두고, 원하는 정수 k가 저장되어있는지, 만약 저장되어 있다면 첫 번째 위치를 출력하게 할 수 있다[20]. 검색의 문제에는 크게 2가지 유형이 있다. 하나는 주어진 조건에 완전한 일치하는 것을 찾는 유형과, 주어진 부울 수식을 만족시키는 해답을 찾는 유형이다[2].

넷째, 최적화의 문제는 문제에 대한 최상의 해결책을 찾는 것으로서 특정 조건에 맞는 최소값과 최대값을 찾는 문

제이다. 예를 들면, ‘TSP(traveling salesman problem)’는 물건을 판매하기 위해 여행하는 세일즈맨의 문제로, 최소한의 비용으로 n개의 도시들을 방문하여 물건을 팔기 위해서는 어떤 순서로 도시를 순회해야 하는지 구해야 하는 문제이다. TSP는 컴퓨터 사이언스의 대표적인 난제로, 정보교과 분야에서 가장 많은 연구가 이루어지고 있다[8].

2.2 문제의 해결

다양성을 기준으로 분류한 Jonsassen의 구조화된 문제와 비구조화된 문제를 중심으로 문제 해결 방법을 찾아보면 다음과 같다[11].

첫째, 구조화된 문제는 문제에 대한 모든 요소를 제시하며, 해결 과정이 미리 정해져 있다. 또한, 구조화된 문제는 제한된 한도에서 예언적이고 처방적인 방식으로 조직된 제한된 수의 규칙 및 원리의 적용을 유도하여 수렴적인 답을 가진다[24]. 예를 들면, 특정 넓이의 벽면에 동일한 크기의 타일을 붙이려고 할 때, 타일이 몇 개가 필요할지를 구하는 문제 등이 있다.

둘째, 비구조화된 문제는 하나 이상의 문제 요소들이 알려지지 않거나, 불확실하여 모호하게 정의되거나, 불명확한 목적과 제약을 가지며, 문제 해결에 필요한 개념, 규칙, 원리들이 어떻게 조직되는지가 불확실하다[24]. 예를 들면, 특정 제조사가 고객이 요구하는 조건에 맞는 다양한 시제품을 만드는 사례가 그러하다. 비구조화된 문제는 여러 가지 해결법을 가지거나, 혹은 해를 전혀 갖지 않고 해결법을 평가하는 여러 준거를 가진다. 비구조화된 문제는 문제 상황에서 어떤 것을 중요하게 생각하느냐에 따라 해결 방법이 달라지기 때문에 문제 해결에 사용된 규칙, 개념, 원리들은 일관성이 없고, 일반적인 원리나 규칙을 제공하지 않는다[24].

컴퓨터과학에서도 타 학문과 마찬가지로 문제 해결을 목표로 하고 있지만, 다른 점은 컴퓨팅을 통해 해결 가능한 의사결정의 문제, 카운팅의 문제, 검색의 문제, 최적화 문제 등을 중심으로 다룬다는 점이다. 또한, 그 해결 방법이 정확한 답을 얻었는지, 가장 빠르고 논리적인지, 시간이나 메모리 등 최소한의 자원을 사용했는지, 다른 문제들을 해결하는 데도 쓰일 수 있는지 등을 컴퓨팅시스템을 활용하여 보다 쉽고, 빠르고, 정확하게 파악할 수 있다[7].

3. 알고리즘 기반 교수학습방법

학생들의 컴퓨팅사고력을 향상시키기 위해서는 앞서 나열한 컴퓨터과학의 문제를 학생 수준에 맞게 제공하고, 학생이 문제를 해결해나가는 과정에서 알고리즘 기반의 사고를 할 수 있도록 해야 한다.

본 연구에서는 문제를 해결하기 위한 절차인 알고리즘을 설계하는 방법을 활용한 교수학습방법을 개발하였다. 즉, 알고리즘 설계 방법인 탐색기반 설계와 관계기반 설계를 중심으로[18] 교사들이 어떤 절차에 따라 학생들을 지도해야 할지를 제시하였다.

3.1 탐색기반 교수학습방법

탐색기반 교수학습방법은 문제를 탐색 가능한 형태로 구조화한 후, 문제 해결 방법을 지도할 때 유용하다.

3.1.1 절차

탐색기반 설계 과정을 바탕으로 한 탐색기반 교수학습방법의 절차는 다음과 같다.

첫째, ‘문제 제시’ 단계에서는 문제 상황에서 해결할 문제를 파악한다. 교사는 발문을 통해 학생들로 하여금 문제가 탐색기반 설계로 해결될 수 있는지, 문제에 주어진 조건은 무엇인지, 최종적으로 출력해야 할 해는 무엇인지를 이끌어낸다.

둘째, ‘구조화’ 단계에서는 문제를 해결하기 위해 탐색할 자료가 저장되어 있는 구조를 파악한다. 탐색의 대상이 되는 구조가 배열이나 연결리스트로 표현될 수 있는지, 트리나 그래프의 형태로 표현될 수 있는지 등을 판단한다. 일반적으로 전자를 선형구조, 후자를 비선형구조라 한다. 따라서 문제를 해결하기 전에 배열, 연결리스트, 트리, 그래프 등에 대한 이해는 반드시 선행되어야 한다. 문제에서 탐색 구조가 명시적으로 드러나는 경우는 쉬운 문제에 속하며, 탐색 구조가 직접 드러나지 않아 문제를 해결하는 과정에서 자체적으로 구조화해야 하는 경우는 비교적 어려운 문제에 속한다[18]. ‘구조화’ 단계는 추상화와 가장 관련이 깊은 단계이며, 어려운 문제일수록 구조화를 위해 더 깊은 사고력을 요구하므로 탐색기반 교수학습방법의 핵심 단계이다.

셋째, ‘구현’ 단계에서는 구조화한 것을 프로그래밍을 통해 구현한다. ‘구조화’ 단계가 문제 해결에 불필요한 요소를 제거하고, 문제 해결에 필요한 절차와 방법을 단순화하여 알고리즘으로 표현하는 ′추상화′와 연관되어 있다면, ′구현′ 단계는 프로그램을 작성해 문제를 해결하는 단계로서 프로그래밍 능력이 뒷받침되어야 한다.

넷째, ‘탐색’ 단계에서는 ‘구현’ 단계에서 작성한 프로그램에 값을 입력하여, 컴퓨터가 해를 탐색하도록 프로그램을 실행한다. 이 때, 한 두 개 정도의 간단한 값을 입력하여 자신이 설계한 프로그램이 의도대로 실행되는지 확인할 수 있다. 컴퓨터의 빠른 연산 속도를 이용해 짧은 시간에 가능한 해의 집합을 탐색하며 최적의 해를 구할 수 있다는 것이 장점이다.

다섯째, ‘반성’ 단계에서는 구조화와 프로그래밍이 잘 되었는지 평가하고, 더 편리하거나 효율적인 방법은 없었는지에 대해 탐구한다. 하나의 문제도 여러 가지 방법으로 해결될 수 있으며, 더 좋은 해결 방법이 있는지를 고민하는 과정에서 사고력이 확장된다.

3.1.2 사례

예를 들면 1부터 n까지의 숫자 중에서 8이라는 숫자가 총 몇 번 나오는지 파악하는 문제가 있을 경우, 그것을 해결하기 위한 절차를 탐색기반 교수학습방법을 통해 지도한다면 다음과 같은 절차가 요구된다.

첫째, ‘문제 제시’ 단계에서는 이 문제가 제한된 범위 내에서 특정한 숫자를 탐색하도록 하는 문제이므로 탐색기반 설계로 해결될 수 있다는 것을 학생들이 이해하게 한다. 또한, ‘문제에서 주어진 것들 중 문제를 해결하기 위해 필요한 요소는 무엇일까?’, ‘최종적으로 출력해야 하는 것은 무엇일까?’ 등의 발문으로 학생들이 문제 해결에 필요한 요소들을 추출할 수 있도록 돕는다.

둘째, ‘구조화’ 단계에서는 문제의 난이도, 학생이 정보 교과 문제를 접해본 정도, 학생의 연령에 따라 발문의 수준을 조절해야 한다. 예를 들어, 정보 교과 문제를 접해보지 않은 학생이나, 상대적으로 연령이 낮은 초등학생에게는 ‘이 문제의 답을 컴퓨터가 일일이 탐색하게 만들기 위해서는 문제를 구조를 어떻게 바꿔서 주는 게 좋을까?’등의 발문으로 구조화를 하도록 한다. 학생들이 구조화를 어려워하더라도 교사가 곧바로 정답을 제시하기 보다는, 학생들이 최대한 스스로 문제를 구조화할 수 있도록 충분한 시간을 제공해야 한다. 이 문제의 경우는 숫자 리스트를 모두 문자열로 변환하여 합친 뒤, 합쳐진 문자열에서 8이라는 문자가 몇 개인지 조사하는 방법으로 구조화할 수 있다[6].

셋째, ‘구현’ 단계에서는 구조화한 내용을 프로그래밍을 통해 구현한다. 이 교수학습방법의 목적은 학생들의 프로그래밍 기술을 향상시키는 것보다 컴퓨팅사고력을 증진시키는 데에 있다. 따라서 이 단계에서 학생들이 구조화한 내용을 프로그램으로 설계하는 것을 어려워할 경우, 교사가 적극적인 도움을 주는 것이 중요하다. 구조화에 성공했으나 프로그래밍에 실패하면 학생들이 학습 의욕을 상실할 수 있기 때문이다.

넷째, ‘탐색’ 단계에서는 설계한 프로그램에 문제에서 제시한 입력값을 입력하여, 컴퓨터가 최적의 해를 탐색하게 한다. 이 문제에서는 합쳐진 문자열에서 8이라는 문자가 몇 개인지 프로그램이 ‘탐색’하도록 한다.

다섯째, ‘반성’ 단계에서는 문제를 해결한 방법에 대해서 토의하는 시간을 가지는 것이 좋다. 자신의 문제 해결 방법을 설명하는 과정에서 자기 평가를 할 수 있으며, 다른 학생들의 해결 방법을 듣고 사고를 확장시킬 수 있다.

3.2 관계기반 교수학습방법

탐색기반 교수학습방법이 컴퓨터의 빠른 연산 속도를 이용해 짧은 시간에 가능한 해의 집합을 탐색하며 최적의 해를 구하는 문제에 적합한 교수학습방법이라면, 관계기반 교수학습방법은 해를 구하는 행위를 하나의 함수로 표현하고 이 함수들의 관계를 이용해 해를 구하도록 할 때 적합하다[17]. 즉, 관계기반 교수학습방법은 학생들이 문제의 정의 및 상태를 함수로 정의하고 이 함수들 간의 관계를 표현하는 방법을 익히게 한다.

3.2.1 절차

관계기반 교수학습방법의 절차는 문제 인식, 함수화, 관계 발견, 구현 및 실행, 반성 등으로 구분할 수 있다.

첫째, ‘문제 인식’ 단계에서는 제시된 문제를 파악한다. 이 단계에서는 제시된 문제를 관계기반 설계로 해결해야 하는 것인지 여부를 판단하는 것이 가장 중요하다.

즉, 제시된 문제를 함수를 정의할 수 있는지, 함수들 간의 관계를 점화식 혹은 이와 유사한 형태로 표현할 수 있는지를 판단한다.

둘째, '함수화' 단계에서는 문제의 정의 및 상태를 함수로 정의하게 한다. 보통 입력 값이 n인 문제의 해를 f(n)으로 정의한다. 이 때, 문제에서 요구하는 해가 무엇인지 학생들이 정확하게 파악하고 f(n)을 정의하도록 유도하는 것이 중요하다. 관계기반 설계는 수학적 귀납법의 논리를 따르므로, f(n)의 정의에 의해 f(1)을 직접 구한다.

셋째, '관계 발견' 단계에서는 f(k)를 이미 구해두었다고 가정하고, f(k)를 통해서 f(n)을 구한다. 가장 손 쉬운 방법은 임의의 자연수를 대입하여 관계를 찾도록 한 뒤, 이를 일반화하게 하는 것이다. 예를 들어 f(7)과 f(8)을 나타낸 뒤, 둘 사이에 모든 함수식에 일반화할 수 있는 관계를 파악하면 이를 f(k)와 f(n)의 식으로 나타낼 수 있다. 여기서 중요한 것은 k를 문제에 따라 적절한 형태로 설계해야 한다는 점이다[17]. 이 과정에서 추상화와 일반화의 사고는 필수적이다.

넷째, '구현 및 실행' 단계에서는 학생들이 작성한 함수식을 프로그래밍을 통해 구현하고 해를 구한다. 탐색기반 교수학습방법과 마찬가지로 이 교수학습방법 역시 학생들의 프로그래밍 기술을 향상시키는 것보다 컴퓨팅 사고력을 증진시키는 데에 목적이 있다. 따라서 이 단계에서 학생들이 작성한 함수식을 프로그램으로 설계하는 것을 어려워할 경우, 교사가 적극적인 도움을 주는 것이 좋다. 함수식을 완성했으나 프로그래밍에 실패하면 학생들이 학습 의욕을 상실할 수 있기 때문이다.

다섯째, '반성' 단계에서는 함수식이 잘 정의되었는지, 프로그래밍이 잘 되었는지 평가한다. 또한 더 편리하거나 효율적인 방법은 없었는지 탐구해본다. 관계식을 설계하는 방법에 따라 다양한 알고리즘을 설계할 수 있고, 설계방법에 따라 계산량이 달라지므로[17] 학생들이 각자 세운 함수식에 대해 토의하면서 평가를 하고 개선할 수 있다.

3.2.2 사례

예를 들면, 임의의 정수를 입력받아 1부터 입력받은 숫자까지 합을 구하는 프로그램을 작성할 때 관계기반 교수학습방법을 적용한 사례는 다음과 같다.

첫째, '문제 인식' 단계에서는 이 문제가 관계기반 설계로 해결해야 하는 것인지 여부를 판단한다. 예를 들어 제시된 문제의 경우, "임의의 자연수 k를 가정하고, '1부터 k까지의 합'과 '1부터 k+1까지의 합'의 관계를 구할 수 있을까?"를 고민하게 한다.

둘째, '함수화' 단계에서는 문제의 정의 및 상태를 함수로 정의하게 한다. 이 문제에서 구하고자 하는 해는 '1부터 n까지의 합'이므로, 그에 맞게 함수 f(n)을 f(n)='1부터 n까지의 합'으로 정의하게 할 수 있다. 이 때, 학생들이 함수식을 정의하는 것을 어려워하더라도 교사가 곧바로 정답을 제시하기보다는, 스스로 함수식을 정의할 수 있도록 충분한 시간을 제공하는 것이 더 좋다. f(n)을 정의했다면, 정의에 의해 f(1)을 구한다. 이 문제에서 f(1)=1이다.

셋째, '관계 발견' 단계는 이 교수학습방법에서 가장 핵심적인 단계이다. f(k)를 이미 구해두었다고 가정하고, f(k)를 통해 f(n)을 구해야 한다. 이 문제의 경우 n이 자연수이기 때문에 간단하게 k를 n-1이라고 가정할 수 있다. n에 임의의 자연수 11을 대입하고, f(11)을 f(10)과의 관계로 나타내보면, f(11) = f(10) + 11이 되고, 이를 일반화 하면 f(n) = f(n-1) + n이 되는 관계를 발견할 수 있다[17]. 함수식의 관계를 발견하는 것이 관계기반 설계에서 가장 중요하므로, 교사는 단계적인 발문과 비계를 통해 학생들이 관계를 발견할 수 있도록 해야 한다.

넷째, '구현 및 실행' 단계에서는 설계한 함수식을 프로그래밍을 통해 구현하고 해를 구한다. 마찬가지로, 프로그래밍이 정확히 되었는지 여부를 판단하기 위해 임의의 수를 입력하고 결과가 제대로 출력되는지를 통해 문제 해결 여부를 확인한다.

다섯째, '반성' 단계에서는 함수식이 잘 정의되었는지, 프로그래밍이 잘 되었는지 평가하고, 다른 해결 방법이 있는지를 탐구해본다. 예를 들어, 이 문제의 경우 앞에서 f(k)를 f(n-1)로 정의하였지만, f(k)를 f($\frac{n}{2}$)이라고 정의할 경우 함수식이 다르게 표현된다.

처음에는 탐색기반 설계로 해결할 수 있는 문제와 관계기반 설계로 해결할 수 있는 문제를 구분하여 제시하고, 학생들이 해결 방법을 익히도록 할 수 있다. 그러나 궁극적으로는 유형과 해결 방법을 알 수 없는 새로운 문제를 만났을 때, 학생들이 그 문제를 탐색기반 설계로 해결할지, 관계기반 설계로 해결할지 판단하고 해결하는

능력을 기르게 하는 것이 중요하다. 그러한 능력이 진정한 의미의 문제해결력이라고 할 수 있을 것이다.

4. 결론 및 제언

컴퓨터과학에서 컴퓨팅을 통해 해결하고자 하는 '문제'란 무엇인가에 대한 정의와 함께, 컴퓨팅을 통한 문제 해결의 필요성과 해결 방법, 컴퓨팅 문제 해결을 위한 교수학습방법을 개발하였다. 이것을 정리하면 다음과 같다.

첫째, 구조화된 문제는 쉽게 해결할 수 있지만 비구조화된 문제 해결에는 한계가 있는 다른 교과와 달리, 정보 교과에서는 구조화된 문제는 물론 비구조화된(ill-structured) 문제도 효율적으로 해결할 수 있었다. 다만, 정보 교과에서 다루는 문제는 프로그래밍 언어와 연관된 컴퓨팅으로 해결 가능한 형태여야 한다. 이러한 문제들을 정보 교과 문제라 하며, 정보 교과 문제를 해결하는 과정을 통해 학생들은 컴퓨팅사고력을 향상시킬 수 있다.

둘째, 정보 교과 문제에는 카운팅 문제, 결정 문제, 검색 문제, 최적화 문제 등이 있다. 카운팅 문제는 조건을 만족하는 해결책들의 경우의 수를 찾는 문제, 결정 문제는 입력 값에 대한 '예' 또는 '아니오'로만 출력 값이 나타나는 문제, 검색 문제는 주어진 그래프 또는 조건에서 완전한 일치를 찾거나 가능한 한 그것을 만족시키는 해답을 찾는 문제, 최적화 문제는 용어와 같이 문제에 대한 최상의 해결 방법을 요구하는 문제이다. 그러나 정보 교과 문제가 모두 제시한 네 가지 유형으로 분류되는 것은 아니며, 문제를 해결하는 사람에 따라 같은 문제를 서로 다른 방식으로 접근하기도 한다.

셋째, 정보 교과 문제를 해결하며 학생들이 컴퓨팅사고력을 향상시키기 위해서 알고리즘 기반의 교수학습방법을 제안하였다. 탐색기반 교수학습방법은 문제를 탐색 가능한 형태로 구조화한 후, 문제를 해결하도록 지도하고, 관계기반 교수학습방법은 학생들이 문제의 정의 및 상태를 함수로 정의하고 이 함수들 간의 관계를 점화식 혹은 이와 유사한 형태로 표현하는 방법을 익히도록 한다. 이러한 방법으로 정보 교과 문제를 해결함으로써 학생들은 문제 분석, 추상화, 자동화, 일반화 등의 컴퓨팅 사고력을 향상시킬 수 있을 것이다.

이상과 같이 컴퓨팅 시스템을 활용하여 비구조화된 문제를 편리하고 효율적으로 해결할 수 있다는 것은 다른 교과와 차별화되는 컴퓨터과학만의 문제 해결 방법이다. 그러나 본 논문에서 제시한 교수학습방법의 목적은 결코 프로그래밍 기술을 향상시키기 위한 것이 아니라, 알고리즘 기반의 문제 해결력을 기르기 위한 것이다. 따라서 교사는 프로그램 작성을 위한 문법이나 기술보다는 학생들이 문제 해결을 위한 깊은 사고를 할 수 있게 하는 데에 초점을 맞추어야 한다. 또한, 학생들의 컴퓨팅사고력을 향상시키기 위해 교사들이 탐색기반 교수학습방법과 관계기반 교수학습방법을 실생활 문제 해결에 적용하고, 그에 따른 문제점을 개선해 나간다면, 알고리즘과 프로그래밍 중심의 SW 교육 내실화에 기여하게 될 것이다.

참고문헌

[1] Agnes Azzolino(2010). Basic Probability and Counting Problems. online: http://www.mathnstuff.com/math/spoken/here/2class/90/basic.htm.

[2] Andrej Boddanov and Andrej Bogdanov(2007). 80240233:Computational Complexity, Lecture 1. 2. Tsinghua University.

[3] Bauhaus - universitat weimar. unit-en-search-introduction Chapter S:I. online: https://www.uni-weimar.de/en/university/start.

[4] Changhyeob Shin(2010). A Study on Mathematical Materials and Algorithms in Teaching Number Operations. 25. Department of Mathematics Education Graduate School Dankook University.

[5] Cheol Kim, Youngsik Jeong, Yeonghun Seong, Namje Park, & Subeom Sin(2016). Development of a standard model for informatics curriculum. 9-10. The Korean Association of Information Education.

[6] Codingdojang(2017). online:http://codingdojang.com/scode/393.

[7] Daesu Kim(2016). Software and Computational Thinking. booksr.

[8] Frank Stephan(2014). 7. Problems. NUS Computing.

[9] Hong, N. S(1999). The relationship between

well-structured and ill-structured problem solving in multimedia simulation, 2-3. The Pennsylvania State University.

[10] Hyeyoung Yun(2008). Comparison of Characteristics of The Processes of Generaty of Scientific Inquiry Problems According to the Inquiry Contexts and Subjects, 22-23. Graduate School of Education Major in Physics Education, Dept. of Education Chonnam National University.

[11] JeeYun Hong(2013). Study on the mathematical abstraction and proportional reasoning of elementary school students in the process of solving an ill-structured problem. 7. Department of Elementary Education The Graduate School of Ewha Womans University.

[12] Jinsuk Kim, Seongwan Han and et al.(2015). Study on Development of Teaching and Learning Methods of SW Education. 78. Korean Educational Development Institute and Korea Education and Research Information Service.

[13] Jinsuk Sim(2004). The theory of the problem-solving studies and its applications, 13-14. Major in Mathematics Education The Graduate School of Education Chung-Ang University.

[14] Luca Trevisan(2010). Standford University-CS254: Computational Complexity. Handout 2, March 31. Standford University.

[15] Minkyeong Kim, Jiyoung Lee, Jeeyun Hong , and Eunkyung Kim(2011). A Study of 'Ill-Structured' Status from Mathematics Problems in Elementary School Textbooks, 9. Learner-Centered Curriculum and Instruction.

[16] Okgi Kang(1989). Development of Teaching-Learning Materials for Improving Mathematics Problem Solving Ability, 20-21. Korean Educational Development Institute.

[17] Seongjin Ann(2017). Creative algorithm for problem solving, high level, 7-8. online: https://www.digitalculture.or.kr/upload/algorithm_ad.pdf

[18] Seongjin Ann et al.(2017). Creative algorithm for problem solving, middle level, 37. online: https://www.digitalculture.or.kr/upload/algorithm_md.pdf.

[19] Seungjae Park and Huiyeong Cho(1995). Science Learning·Teaching Method. kyoyookbook.

[20] T. HS Jeon(2017a). Problem Number 1866, Return the position of the desired value with a function. online: http://koistudy.net/?mid=prob_page&NO=1866&SEARCH=0.

[21] T. HS Jeon(2017b). Problem Number 1873, Return the number of divisors of n with a function. online: http://koistudy.net/?mid=prob_page&NO=1873&SEARCH=0.

[22] Wikipedia(2017a). Informatics. online: https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%EA%B3%BC%ED%95%99#바탕이_되는_이론.

[23] Wikipedia(2017b). Computational problem. online: https://en.wikipedia.org/wiki/ Computational_problem.

[24] Wooyung Jang(2017). Analysis of relationship between mathematical thinking and core competency of primary mathematical gifted students when solving ill-structured problem, 8. Major in Elementary Mathematically Gifted Education Graduate School of Education, Gyeongin National University of Education.

[25] Wood, P. K.(1983). Inquiring systems and problem structures: Implications for cognitive development, 13.

저자소개

임 서 은

2016 전주교육대학교(학사)

2016~현재 군포초등학교 교사

관심분야 : 정보교육, SW교육, 프로그래밍

e-mail : yhsung@cue.ac.kr

정 영 식

1996 춘천교육대학교 수학교육학과(교육학학사)

2001 한국교원대학교 컴퓨터교육과(교육학석사)

2004 한국교원대학교 컴퓨터교육과(교육학박사)

2004~2011 한국교육개발원 연구위원

2004~현재 전주교육대학교 컴퓨터교육과 교수

관심분야 : 컴퓨터교육, 프로그래밍, 이러닝

e-mail : nurunso@jnue.kr