Computational Thinking을 통해서 초등 교육 바라보기

고영남* · 김종우*
*제주대학교 교육대학 초등컴퓨터교육전공
e-mail:arang57@naver.com

Seeing the elementary education through computational thinking

Young-Nam Ko*, Jong-Woo Kim*
*Dept. of Computer Education, Teachers College, Jeju National University

요 약

컴퓨터 사용이 우리의 일상 생활이 되면서 컴퓨터 과학은 이미 우리 사회 여러 영역을 바꾸어 놓았다. 그리고 이런 관점에서 볼 때 컴퓨터 과학자처럼 사고하고 문제를 해결하는 능력인 Computational Thinking은 앞으로의 사회에서 나타날 문제를 해결하는 해결 방법이 될 것이고 미래사회를 살아가는데 필수 능력이 될 것이다. 따라서 본 연구에서는 Computational Thinking의 정확한 개념을 살펴보고 초등교육에서는 어떤 부분에서 Computational Thinking을 찾아볼 수 있는 지 알아보고자 한다.

1. 서론

인류 역사상 가장 위대한 발명품을 꼽으라고 하면 과거에는 증기기관차, 문자 등을 얘기했지만 지금은 컴퓨터를 얘기하는 사람들이 많을 것이다. 그만큼 컴퓨터는 우리 생활에 너무나도 큰 영향을 주고 있어 컴퓨터 없는 세상을 상상하기가 쉽지 않을 정도이다.

이런 컴퓨터의 발명과 함께 21세기 정보 기술의 발전을 이끌어온 학문은 컴퓨터 과학이다. 컴퓨터 과학은 이미 사회의 여러 모습을 바꾸어 놓았고, 다른 학문의 모습도 바꾸어 놓았으며 앞으로도 계속 변화시킬 것이다. 따라서 컴퓨터 과학을 이해하는 것은 앞으로의 우리들의 미래 모습을 이해하는데 굉장히 중요하다 할 수 있다.

컴퓨터 과학은 단순히 컴퓨터라는 기계에 대한 학문으로만 국한하지는 않는다. 유명한 컴퓨터 과학자 에츠허르 데이크스트라는 "컴퓨터 과학에서 컴퓨터란, 천문학에서 망원경 이상의 것이 아니다."라고 하였다[7]. 이는 컴퓨터 과학에서 컴퓨터는 도구에 지나지 않는다는 것을 말하는 것이며, 더 나아가 컴퓨터 과학은 인간의 사고 방식에 관한 학문임을 암시하는 내용이다. 실제로 컴퓨터(computer)라는 용어는 1920년대 이전에는 계산(compute)을 담당하는 사무관을 일컫는 용어였으며 초기 컴퓨터 과학자들은 계산(compute) 가능성 문제(종이와 연필만을 가진 사무관이 철저한 지시에 따라 계산하면 풀 수 있는 문제들은 어떤 것들인가?)에 흥미를 가졌다[7]. 즉, 컴퓨터 과학은 어떻게 하면 문제를 해결할 수 있는가를 고민하는 문제해결을 위한 학문인 것이다.

컴퓨터와 컴퓨터 과학이 우리의 사회에 미친 영향이 대단하다고 볼 때 컴퓨터 과학자처럼 생각하고 문제를 해결하는 능력은 앞으로의 사회에서 나타날 문제를 해결하는데 큰 역할을 할 것임은 자명한 사실이다. 이에 Wing(2006)은 컴퓨터 과학자처럼 생각하는 사고방식을 Computational Thinking이라고 하였으며[2] 이러한 Computational Thinking은 우리가 느끼는 것 이상으로 우리의 생활에 밀접하게 자리 잡고 있다.

이에 본 연구에서는 우리 생활에 많은 영향을 주는 Computational Thinking의 개념을 살펴보고 핵심요소들에 대해 알아본 후 실제 초등 교육 속에서 교육 상황을 Computational thinking의 관점에서 접근해보고자 한다.

2. Computational Thinking

2.1 Computational Thingking이란 무엇인가?

최근 컴퓨터 과학자들과 교육 연구자들은 컴퓨터 과학 교육의 근본적인 목표로 Computational Thinking 능력의 향상을 주장하고 있다[5].

Computational thinking은 컴퓨터 과학의 기본 개념을 기반으로 문제해결, 시스템 디자인 뿐만 아니라 인간행동의 이해를 포함한다[2]. 이것은 분석적 사고의 하나로써 수학적 사고와 함께하여 문제 해결에 접근할 수도 있고, 공학적 사고와 함께하여 크고 복잡한 체계를 디자인하고 평가하는데 접근할 수도 있으며, 과학적 사고와 함께하여 지능, 인간 행동성, 계산 가능성을 이해하는데 접근할 수도 있다[2]. 즉, Computational Thinking은 광범위한 컴퓨터 과학 분야가 반영된 정신적 도구인 것이다[2].

Wing(2006)은 Computational Thinking을 단지 컴퓨터

과학자만을 위한 것이 아닌, 모든 사람들이 갖추어야 할 기본 기술로 정의하고 있고, 3R(Reading, wRiting, aRithmetic)과 더불어 모든 아이들의 분석적 능력에 포함해야 한다고 주장하였다[2].

Computational Thinking에 대해 Wing(2006)은 다음과 같이 개념을 정리하였다[2].

첫째, 환원, 변형, 시뮬레이션등의 방법을 통해 어떤 어려운 문제를 해결방법을 알고 있는 다른 문제로 재정형화(reformulating)하는 것이다.

둘째, 재귀적사고(thinking recursively)이고 이는 병렬처리(parallel processing)이다.

셋째, 커다란 문제를 해결하거나 거대하고 복잡한 시스템을 설계할 때 추상화와 분해를 사용하는 것이다.

다섯째, 예방, 보호 그리고 오류 정정과 피해방지, 중복성을 통해 최악의 경우로 부터 복원한다는 관점에서의 사고방식이다.

여섯째, 해결책을 찾기 위한 발견적 추론이다.

일곱째, 빠른 문제 해결을 위해 대량의 정보를 사용하는 것이다.

이러한 Computational Thinking은 다른 학문에도 많은 영향을 주었다. 기계학습은 통계학을 변화시켰고, 데이터 구조와 알고리즘은 생물학에 영향을 주어 컴퓨터생명공학은 생물학자들의 생각을 바꾸었다. 마찬가지로 컴퓨터 게임이론은 경제학자들의 생각을, 나노컴퓨팅은 화학자들의 생각을, 퀀텀 컴퓨팅은 물리학자들의 생각을 바꾸었다[2].

Computational Thinking은 다음의 특징을 갖는다[2].

첫째, 프로그래밍이 아닌 개념화이다.

둘째, 기계적 기술이 아닌 근본적인 기술이다.

셋째, 수학적사고와 공학적 사고의 적용과 조합이다.

넷째, 소프트웨어나 하드웨어와 같은 인공적이 아닌 컴퓨터적 개념으로 문제 해결에 접근하는 사고방식이다.

다섯째, 모든 사람들, 모든 장소를 위한 것이다.

Computational Thinking이란 결국 그 개념과 특징들을 생각해 볼 때, 단순히 컴퓨터가 문제를 해결하는 일련의 사고 과정을 이해하는 것이 아닌, 컴퓨터적인 개념들을 바탕으로 문제를 해결하는 일련의 모든 사고과정이라고 할 수 있겠다.

2.2 Computational Thingking의 핵심 요소

2.2.1 추상화 (abstraction)

경험적인 과학은 실험을 통해 구체적인 모델을 만들고 수학을 통해 추상적인 모델을 만들지만 컴퓨터 과학은 소프트웨어에 의해서 구현되기 때문에 그들의 모델은 물리적으로 구체적이지 않고, 이러한 비물리적인 관점에서 컴퓨터 과학의 모델은 추상적이다.

이러한 면에서 Wing(2008)은 Computational Thinking의 본질은 추상화(abstraction)이고 컴퓨터 과학에서는 시간과 공간의 물리적 차원을 넘어선 개념을 추상화한다고 하였다[3].

물론 수학도 기호와 숫자를 이용하여 추상적인 모델을 만들지만, 수학에서의 추상화와 컴퓨터 과학에서의 추상화는 그 본질에 있어서 다른 점을 보인다[1].

수학에서의 추상화는 추론 구조(inference structures)를 생성하는데 그 목표가 있지만 컴퓨터 과학에서의 추상화는 **상호작용 패턴**(interaction patterns)을 생성하는데 그 목표가 있다[1]. 모든 컴퓨터 소프트웨어는 컴퓨터적 처리 과정의 상호작용이며 이것은 기계 지향적인 개념이 숨어있는 추상화 단계들에 의해 가능하다[1]. 즉, 컴퓨터 소프트웨어는 메모리, 프로세서, 사용자 등의 상호작용 패턴의 총집합으로 만들어지는 것이다.

어떤 과학이던지 그들의 주제에 대해 형식적인 수학적 모델을 구성함으로써 성과를 거둔다. 이때 추상화의 과정을 거치면서 불필요한 정보나 관련이 없는 정보를 제거하게 된다. 이것을 정보 부정(information neglect)이라고 한다[1].

하지만 컴퓨터 과학에서의 추상화의 목적은 정보숨김(information hiding)이다[1]. 소프트웨어의 관점에서 볼 때 비트(bit)는 플립플롭(flip-flop)이라는 추상화에 의해 컴퓨터로 해석되지만 우리들은 플립플롭이 어떻게 만들어진 건지 알 필요가 없다. 추상화에 플립플롭에 대한 정보가 숨어있는 것이다. 플립플롭뿐만이 아니다. 바이트(byte)도 8비트가 모여서 만들어진 것이지만 어떻게 비트가 정렬이 되어 있는지에 대한 정보는 바이트라는 추상화에 숨어있을 뿐이고 우리는 그냥 바이트를 사용하는 것이다. 이것은 알고리즘도 마찬가지이다. 정렬알고리즘이 있다면 그 알고리즘 속에는 알고리즘의 동작을 위한 여러 명령어들이 숨어있을 뿐이고 우리는 명령어들에 대한 정보와 상관없이 알고리즘을 활용하여 데이터를 정렬할 뿐이다. 프로그래밍에서의 클래스, 컴포넌트 등도 전부 정보숨김이 되어있는 추상화이다.

이러한 컴퓨터 과학적 추상화에 대해 Wing(2008)은 수학적, 물리학에서의 추상화와 다음의 두 가지 차이점이 있다고 했다[3].

첫째, 컴퓨터 과학적 추상화는 우리가 실생활에서 쓰이는 수학적 추상화의 간결하고, 명료하게 적용할 수 있는 대수학적 속성을 이용할 수가 없다. 예를 들어 2개의 추상자료형인 스택을 더할 때, 두 숫자를 더하듯이 더할 수 없는 것이다[3].

둘째, 컴퓨터 과학적 추상화는 물리적 세계의 제약 내에서 수행하도록 구현되기 때문에 디스크가 꽉 차 있거나 서버가 응답을 하지 않는다면 어떤 일이 일어날까 등과 같은 특수한 사례(edge case)와 실패 사례(failure case)를 염두에 두어야 한다[3]. 이는 컴퓨터 과학적 추상화과 우리 실생활과 밀접한 관련이 있음을 시사하는 것이다.

그리고 추상화의 과정에는 레이어(layers)가 도입된다. 그리고 응용프로그램인터페이스(API)나 인터넷 레이어 구조처럼 잘 정의가 된 레이어들 간의 인터페이스는 더 크고 복잡한 시스템을 갖출 수 있게 한다.

결국 Computational Thinking의 핵심 요소인 추상화는 문제해결을 위한 추상화이고 이것은 여러 사물들 간의, 사물과 인간사이, 또는 인간과 인간 사이의 상호작용의 패턴 안에 정보를 숨김으로써 추상화하는 것이라고 할 수 있으며, 수학처럼 간결하고 명료하게 해결할 수 없고 실생활과 밀접하게 관련이 있기 때문에 더욱 복잡한 사고 능력을 요구하는 것이라고 할 수 있겠다.

즉, Computational Thinking에서의 중요한 점은 추상화를 정의하는 것, 추상화의 다중 레이어와 함께 수행하는 것, 그리고 다른 레이어 사이의 관계를 이해하는 것이다[3].

2.2.2 자동화(Automation)

추상화능력은 자동화에 의해 강화된다. 자동화는 추상적 개념들을 해석하기 위한 일종의 컴퓨터의 필요성을 의미하는 것이다[3].

가장 대표적인 자동화 장치는 정보를 처리하고, 저장하는 컴퓨터이지만 이는 꼭 컴퓨터만을 말하는 것은 아니며, 인간도 정보를 처리하고 계산할 수 있기에 인간도 자동화 장치라고 할 수 있다. 달리 말하면 Computational Thinking이 기계를 요구하는 것은 아니다. 더욱이 우리가 인간과 기계의 조합을 컴퓨터로 여긴다면 우리는 인간과 기계의 결합된 처리능력을 이용할 수 있다[3]. 즉, 자동화 장치는 기계, 인간, 기계와 인간의 조합이 될 수 있는 것이다.

컴퓨팅은 '문제해결을 위해 어떻게 컴퓨터를 얻을 것인가?' 라는 물음에 대한 대답과 관계가 깊다. 그리고 그 대답은 적합한 추상화 개념을 정립하는 것과 과제 수행을 위한 적합한 종류의 컴퓨터를 선택하는 것이다. 불행하게도 이 물음에는 올바른 추상화를 정의하는 것에 깊이 생각하지 않고 문제 해결에 강력한 힘을 가진 기계를 선택하는 것으로 대답하기가 쉽다. 하지만 Computational Thinking은 이러한 간단한 기계적인 컴퓨터의 사용보다 더 많은 것을 제공할 것이다[3].

즉, Computational Thinking은 문제 해결을 위해 적합한 **추상화 개념을 정립**하고 적합한 **자동화 장치를 선택**하는 것을 말하며, 단지 컴퓨터의 영역이 아닌 다른 영역에도 긍정적인 영향을 미칠 것이라는 것이다.

3. Computational Thinking으로 초등 교육 바라보기

Computational Thinking은 컴퓨터 과학자처럼 사고하는 것을 말한다. 이는 컴퓨터 과학의 기본 개념을 적용하여 문제를 해결하는 것을 말하며 컴퓨터 과학의 기본 개념들은 문제 해결을 위해 정립된 적합한 추상화 개념들이다. 이러한 컴퓨터적 개념들이 우리 교육에서는 어떻게 적용되는지 살펴보자.

3.1 국어과에서의 Computational Thinking

(그림 1)은 초등학교 6학년 1학기 말하기·듣기·쓰기 교과서의 2단원 4차시 '묘사의 방법으로 글쓰기'와 관련하여 수업한 국어과 교수·학습 지도안의 일부이다.

활동1 (전체학습)	▣ 활동 1.몬스터 묘사하기 - 묘사를 잘하기 위해서는? · 전체에서 부분으로 묘사하기 · 순서를 정해서 묘사하기 · 인상적인 부분을 강조해서 묘사하기 · 다른 사물에 빗대어 나타내기 · 흉내 내는 말을 사용하기 - 묘사글만 자르고 다른 친구들과 나눠 갖기	10′	◆ PPT ◆ 개별 학습지 ※ 그림을 그릴 때 너무 세세하게 그리지 않도록 한다.
활동2 (개별학습)	▣ 활동 2. 몬스터 그리기 - 묘사글을 나눠주면 글을 보고 몬스터 그리기 - 같은 몬스터 찾아서 돌려주기	10′	
활동3 (개별학습)	▣ 활동 3. 묘사 글 수정하기 - 자기의 몬스터와 친구가 그린 몬스터 비교하기 - 부족한 부분을 찾아 수정하기 - 수정한 묘사 글 발표하기	5′	

(그림 1) 국어과 교수학습지도안

이 수업은 미국의 한 초등학교에서 시작된 '몬스터 프로젝트'를 모티브로 구안된 수업이다. '몬스터 프로젝트'란 1995년 미국의 한 초등학교에서 시작하여 2010년에도 진행 중인 프로젝트로 현재 20만 명 이상의 어린이가 참여하고 있다. 자기가 좋아하는 괴물을 그리고, 글로 설명한 뒤 글만 이메일로 전송하고 글을 받은 어린이가 그 괴물을 상상해 그려서 답 메일을 보낸 다음 서로 비교하는 활동이다.

이 수업은 이런 몬스터 프로젝트의 일련의 과정을 다음과 같이 적용하였다.

첫 번째, 묘사의 방법을 알아본다.

두 번째, 나만의 몬스터를 그리고, 그 몬스터를 묘사하는 글을 쓴다.

세 번째, 묘사한 글만 잘라내어 친구들과 나누어 갖는다.

네 번째, 친구로부터 받은 몬스터를 묘사한 글을 보고 몬스터를 그린다.

다섯 번째, 묘사한 글을 쓴 주인에게 자기가 그린 몬스터와 받았던 묘사글을 돌려준다.

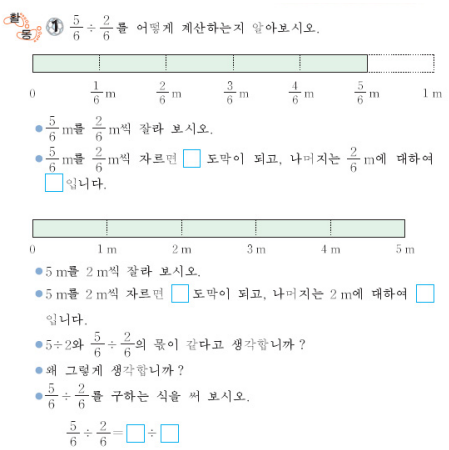
여섯 번째, 묘사한 글을 쓴 주인은 자기가 그린 몬스터와 친구가 그려준 몬스터의 차이점을 확인하면서 자신이 묘사한 글을 수정한다.

여기서 아동이 자신의 묘사 글을 수정하는 모습은 마치 프로그래머가 자신의 프로그램에서 오류를 정정하는 모습과 같다. 아동은 자신이 그린 몬스터와 친구가 그린 몬스터를 비교하면서 차이점을 찾아내 자신의 묘사 글에서 그 잘못 묘사된 부분만을 수정한다. 즉, 문제해결을 위한 추상화과정에서 상호작용 패턴을 생성하고, 재귀적 사고로 문제를 해결하는 것이다. 이는 프로그래머가 자신의 프로그램을 디버깅하는 것과 마찬가지이다. 프로그래머는 자신이 생각한 프로그램과 실제 컴퓨터에서 나타나는 프로그램의 차이점을 찾아내어 자신이 작성한 명령어들을 수정한다. 그리고 이 과정을 반복하면서 프로그래머는 자신의 프로그램을 완성한다.

이 수업을 통해 아동들은 정확히 묘사하는 방법과 더불어 글을 수정하는 것의 중요성을 깨닫게 된다. 이는 프로그래머가 디버깅의 중요성을 알게 되는 것과 더불어 Wing(2006)이 제시한 Computational Thinking의 개념 중 하나인 오류 정정의 관점에서의 사고방식과 일맥상통한다고 할 수 있다.

3.2 수학과에서의 Computational Thinking

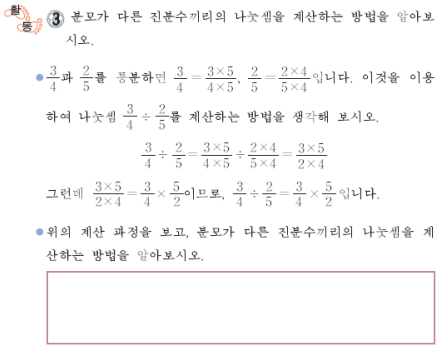
(그림 2)는 초등학교 6학년 2학기 수학 교과서의 1단원 1차시 '분모가 같은 진분수끼리의 나눗셈 알아보기'의 교과서의 일부분이다.



(그림 2) 6-나 수학 교과서 1단원 1차시 학습문제

처음으로 분수의 나눗셈을 배우는 아동들에게 분모가 같은 진분수끼리의 나눗셈인 $\frac{5}{6}\div\frac{2}{6}$를 그들이 이미 알고 있는 5÷2로 변형시켜서 가르치고 있다. 도막의 수는 2이고 나머지는 나누는 수에 대하여 반이 되는 위의 두 상황은 서로 비슷하다. 그러므로 이와 같은 활동을 통하여 나눗셈 $\frac{5}{6}\div\frac{2}{6}$와 5÷2의 몫이 같음을 학생들이 발견할 수 있다[4]. 그리고 이러한 계산 결과를 바탕으로 분모가 같은 진분수끼리의 나눗셈은 분자들의 나눗셈과 같다는 사실을 일반화한다.

(그림 3)은 (그림 2)에서 이어지는 2차시의 교과서의 일부분이다.



(그림 3) 6-나 수학 교과서 1단원 2차시 학습문제

1차시에 이어서 2차시에서는 분모가 다른 진분수끼리의 나눗셈을 1차시에서 일반화 한 사실을 바탕으로 가르치고 있다. 즉, 분모가 다른 진분수를 통분하여 분모를 같게 하고 그 다음 분모가 같은 진분수끼리의 나눗셈은 분자들의 나눗셈과 같다는 사실을 이용하여 문제를 해결하고 있다.

그런데 $\frac{3\times5}{2\times4}=\frac{3}{4}\times\frac{5}{2}$이므로 $\frac{3}{4}\div\frac{2}{5}=\frac{3}{4}\times\frac{5}{2}$임을 발견하게 한다. 이 사실을 통하여 일반적으로 분모가 다른 진분수끼리의 나눗셈을 계산하는 방법을 발견하게 한다[4].

이로써 우리가 흔히 사용하는 분수의 나눗셈 방식인 ÷를 ×로 고치고 나누는 수를 역수를 취하여 계산하는 방법을 발견하게 되는 것이다. 그리고 다음 문제부터는 새로 발견한 이 방법을 사용할 뿐 그 속에 있는 통분을 하여 분모를 같게 하는 것, 분모가 같은 진분수끼리의 나눗셈은 분자끼리의 나눗셈과 같다는 사실은 이제 더 이상 신경 쓰지 않는다. 이는 분모가 다른 진분수끼리의 나눗셈에서의 상호작용 패턴을 **정보 숨김**을 통해 추상화한 것이다.

또한 분모가 다른 분수끼리의 나눗셈 과정이 분모가 같은 분수끼리의 나눗셈을 토대로 **재정형화**되었다는 면에서도 Wing(2006)이 말한 Computational Thinking의 개념과 같다고 볼 수 있겠다.

4. 결론

컴퓨터의 사용이 이미 일상화 되어버린 이 시점에서 Computational Thinking의 개념은 더욱 중요해질 것이다. 그래서 본 연구에서는 Computational Thinking의 개념과 핵심요소, 그리고 Computational Thinking의 관점에서 초등교육의 일부분을 바라보았다.

하지만 초등교육에서 지극히 일부분이었고, 특정 부분만을 바라보았기 때문에 Computational Thinking이 초등 교육에 이미 많이 녹아 있다고 일반화하기는 어렵다. 그러나 이러한 시도가 Computational Thinking이 교육에 녹아들도록 하는 데에 견인차 역할을 할 수 있다고 생각한다.

Computational Thinking은 21세기를 이끌어갈 사고방식이다. 인쇄물이 읽기, 쓰기, 셈하기의 능력의 확산을 촉진시켰듯이, 컴퓨터는 Computational Thinking의 확산을 촉진시키고 있다. 이러한 시대적 흐름 속에서 우리는 Computational Thinking을 어떻게 교육에 스며들게 하고, 어떻게 자연스럽게 아동들이 배우게 할 것인가에 대해 더욱 고민해야 할 것이다.

참고문헌

[1] Timothy Colburn and Gary Shute. (2007). Abstraction in Computer Science. Minds & Machines. 17, 169-184.

[2] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

[3] Wing, J. M. (2008). Computational thinking and thinking about computing. Philos Transact A Math Phys Eng Sci. 366(1881), 3717-3725.

[4] 교육과학기술부. (2010). 초등 학교 교사용 지도서 6-나, 70-73.

[5] 이영준. (2008). Computational Thinking 향상을 위한 초·중등학교 정보 교육의 방향. 컴퓨터교육학회지. 2(1), 17-21.

[6] 몬스터프로젝트. http://www.monsterexchange.org/

[7] 위키피디아. http://ko.wikipedia.org/wiki/