컴퓨팅적 사고

컴퓨팅적 사고는 컴퓨터 과학자뿐만이 아니라 누구나 배워서 활용할 수 있는 보편적인 사고이자 기술이다.

컴퓨팅적 사고는 사고의 주체가 컴퓨터건 사람이건 간에 전산처리의 힘과 한계에 기반해 있다. 컴퓨팅적 방법론과 모델을 통해 우리는 혼자서는 만들 수 없는 시스템을 설계하고 어려운 문제를 해결할 수 있을 거라는 자신감을 얻을 수 있다. 우리는 컴퓨팅적 사고를 통해 수수께끼와 같은 기계 지능의 난제에 도전한다. 인간이 컴퓨터보다 잘 하는 것은 무엇이 있을까? 반대로 컴퓨터는 인간보다 무엇을 잘 할까? 그리고 궁극적으로는 계산할 수 있는 대상에 한계가 있는가 묻고 있다. 우리는 여전히 이 질문들에 대한 만족스러운 답은 구할 수 없는 상태다.

컴퓨팅적 사고는 컴퓨터 과학자뿐만이 아니라 누구나 갖춰야 하는 기본적인 역량이다. 읽기, 쓰기, 셈하기와 같이 아이들이 기본적으로 갖춰야 하는 분석 역량 항목에 컴퓨팅적 사고를 추가해야 한다. 인쇄술이 읽기, 쓰기, 셈하기의 능력을 널리 확산시켜 보편화 했듯이 컴퓨터는 오늘날 컴퓨팅적 사고를 확산시키고 있다.

컴퓨팅적 사고를 한다는 것은 컴퓨터 공학의 기본 개념을 끌어와 문제를 해결하고, 시스템을 설계하고, 인간의 행동을 이해할 수 있다는 것이다. 컴퓨팅적 사고는 컴퓨터 공학의 폭넓은 분야만큼이나 다양한 지적 도구들을 자랑한다.

우리는 어려운 문제에 직면했을 때 이 문제가 얼마나 어려운지, 또한 최선의 해결책은 무엇인지 고민할 것이다. 컴퓨터 과학자라면 이론적 근거에 기반한 정확한 답을 구하려 할 것이며 그에게 문제의 난이도는 답을 찾기 위해 사용될 기계의 사양에 따라 달라질 것이다. 또한 그는 컴퓨터의 명령어 집합 (instruction set)[1], 자원 제약, 그리고 작동 환경을 고려하지 않을 수 없을 것이다.

그리고 문제를 효율적으로 해결하기 위해서는 정확하지 않지만 근사한 해결 방법도 괜찮은지, 무작위 추출을 이용하는 것이 용이한지, 거짓 양성 (false positives)과 거짓 음성 (false negatives)[2]을 허락해도 될지 고민할 것이다. 우리는 컴퓨팅적 사고를 통해 축소, 내장, 변형이나 시뮬레이션과 같은 기법으로 무척이나 어려워 보이는 문제를 이미 해결 방법을 알고 있는 문제로 재구성할 수도 있다.

컴퓨팅적 사고는 재귀적 사고이자 병렬 처리이다. 그것은 코드를 데이터로 반대로 데이터를 코드로도 해석하는 능력이다. 그것은 차원 해석(dimensional analysis)[3]의 일반화를 위해 유형 확인(type checking)을 하는 것이며 에일리어싱 (aliasing) 또는 어떤 대상을 여러 이름으로 부르는 것에 대한 장점과 위험성을 이해하는 것이다. 또한 그것은 간접주소 지정(indirect addressing) [4]과 프로시져 호출의 비용과 혜택을 이해하는 것이다. 컴퓨팅적 사고는 어떠한 프로그램을 단순히 정확함이나 효율성의 면에서만 따지는 것이 아니라 시스템의 심미성, 디자인의 간결함과 정밀함을 판단할 수 있는 사고력이기도 하다.

컴퓨팅적 사고는 추상화(abstraction)[5]와 분해(decomposition)를 통해 복잡한 시스템을 설계하거나 어려운 문제를 해결하는 것이다. 그것은 문제 덩어리를 분리해내는 능력이기도 하다. 우리는 복잡한 문제를 접근하기 쉽도록 그것을 적절하게 묘사하거나 관련 있는 특징들을 모델링 할 수 있다. 그것은 또한 불변식(invariants)을 이용해 시스템의 작용을 간단명료하게 서술하는 것이다. 컴퓨팅적 사고를 갖추면 우리가 크고 복잡한 시스템의 세부사항을 알지 못하더라도 그것을 안전하게 사용하고 고쳐 쓸 수 있다는 확신을 가질 수 있다. 또한 그것은 수많은 다양한 사용자를 고려해 무엇인가를 모듈화를 하는 것, 혹은 미래의 사용성을 대비해 프리페칭(prefetching)이나 캐싱(caching)을 하는 사고력이기도 한다.

컴퓨팅적 사고는 또한 복제, 피해 억제, 오류 수정을 통해 최악의 시나리오에 대한 예방, 보호, 회복에 관한 사고다. 컴퓨팅적 사고의 세계에서 정체는 '교착 상태(deadlock)' [6]라 하고, 계약은 '인터페이스'[7]라고 한다. 그것은 동기화가 일어났을 때 경쟁 상황[8]을 회피하는 법을 배우는 것이다. 컴퓨팅적 사고는 발견적 추론을 통해 해결 방법을 찾는 것이다. 그것은 불확실한 상황에서 계획을 짜고, 학습을 하고 일정을 세울 수 있는 역량이다. 컴퓨팅적 사고는 끝없는 탐구와 꼬리를 무는 검색이며 이를 통해 웹 페이지 목록, 게임 공략법, 반증 사례 등을 얻을 수 있다. 또한 컴퓨팅적 사고는 더욱 빠른 계산을 위해 방대한 데이터를 사용하는 것이며 시간과 공간, 처리 능력과 기억 용량의 균형을 계산하는 것이다.

컴퓨팅적 사고는 언제쯤 보편화된 개념이 될까? 그것은 **컴퓨팅적 사고의 핵심은 프로그래밍이 아닌 개념화에 있다.** 알고리즘[11]과 전제조건과 같은 단어가 일상화되었을 때, 비결정성(nondeterminism)과 가비지 컬렉션 (garbage collection)과 같은 용어가 컴퓨터 과학자들이 쓰는 것과 같은 뜻으로 쓰일 때, 그리고 나무를 트리 구조와 같이 거꾸로 그릴 때 컴퓨팅적 사고는 일상적인 개념이 될 것이다.

우리는 오늘날 컴퓨팅적 사고가 다른 분야에 미치는 영향을 볼 수 있다. 그 예로 기계학습이 통계학을 바꾸고 있는 것을 볼 수 있다. 현재 통계적 학습 기술은 과거에는 시도하지 못한 막대한 스케일의 데이터와 고차원의 문제를 해결하는데 쓰여지고 있다. 오늘날 기업의 통계 관련 부서에서는 컴퓨터 공학자들을 채용하고 있으며 컴퓨터 공학 학과에서는 타 학과의 통계 수업을 과정에 포함시키거나 새로운 통계 강좌를 열고 있다.

컴퓨팅적 사고의 핵심은 프로그래밍이 아닌 개념화에 있다.

우리는 일상에서 쉽게 컴퓨팅적 사고를 접할 수 있다. 당신의 아이가 학교 갈 채비를 할 때 그 날 필요한 물건들을 가방 안에 챙겨 넣을 것이다. 이처럼 곧 꺼내서 써야 할 자료나 정보를 미리 준비해 놓는 것을 컴퓨터 과학에서는 프리페칭 (prefetching)과 캐싱(caching)이라고 한다. 아이가 장갑을 잃어버렸다. 아마도 당신은 아이에게 걸어온 길을 되짚어 보라고 할 것이다. 이와 같이 문제해결을 위해 과정을 거꾸로 되짚는 것을 백트래킹 (backtracking) 기술이라고 한다. 어느 시점에 스키 대여를 그만하고 스키를 구입하는 것이 이득일까? 이러한 비용에 대한 계산을 온라인 알고리즘이라고 한다. 슈퍼 계산대 앞에 온 당신, 어느 줄에 서는 게 유리할까? 정해진 수의 계산대를 많은 사람들이 가장 효율적으로 통과하는 방법을 고민하는 것, 이와 유사하게 방대한 데이터를 효율적으로 받고 보내는 방법을 고민한 결과 나온 기술이 멀티 서버 시스템의 성능 모델링의 기술이다. 전기가 나갔는데 어떻게 전화기는 될까? 이는 복제를 통한 장애극복 기술이다. 어떻게 자동화된 튜링 테스트[9] 나 CAPTCHA[10]가 사용자가 사람인지 아닌지 구분을 할 수 있는 것일까? 이것은 악성 코드를 막기 위해 어려운 AI 문제를 활용하는 기법의 예다.

컴퓨터 공학자들은 또한 최근 생물학에 대한 관심을 갖게 되었는데 이는 컴퓨팅적 사고가 생물학에서도 기여할 점이 많을 거란 아이디어에서 출발했다. 컴퓨터 공학자들은 데이터를 뒤져 패턴을 찾아내기도 하지만 그들이 생물학에 기여한 바는 그 이상이다. 컴퓨터 공학자들은 데이터 구조와 알고리즘 등의 컴퓨팅적 추상화 기법을 통해 단백질 구조를 추상화해서 그 기능에 대한 이해를 넓힐 수 있을 것이라고 기대하고 있다. 이처럼 컴퓨팅적 생물학은 생물학자들의 사고를 바꾸고 있다. 또한 컴퓨팅적 게임 이론은 경제학자들의 사고를 바꾸고 있다. 이와 같이 나노컴퓨팅이 화학자들의 사고를, 양자 컴퓨팅이 물리학자들의 사고를 바꾸고 있다.

이러한 컴퓨팅적 사고가 과학자뿐만이 아니라 모두의 능력이 될 날이 올 것이다. 오늘날이 유비쿼터스 컴퓨팅의 시대라면 미래는 컴퓨팅적 사고의 시대가 될 것이다. 어제의 꿈이었던 유비쿼터스 컴퓨팅이 오늘날의 현실이 된 것처럼 컴퓨팅적 사고는 내일의 현실이 될 것이다.

컴퓨팅적 사고의 허와 실

컴퓨터 공학은 계산에 관한 학문이다. 무엇이 계산될 수 있으며 어떻게 계산할 것인지에 관한 학문이다. 따라서 컴퓨팅적 사고는 다음과 같은 특징을 갖는다.

- **컴퓨팅적 사고의 핵심은 프로그래밍이 아닌 개념화에 있다.** 컴퓨터 공학은 컴퓨터 프로그래밍이 아니다. 컴퓨터 공학자와 같이 사고한다는 것은 컴퓨터 프로그래밍을 할 줄 아는 것, 그 이상이다. 여러 단계의 추상화를 통해 사고 하는 것이 컴퓨팅적 사고다.
- **컴퓨팅적 사고는 단순 반복적인 기술이 아닌 모든 사람이 갖춰야 하는 핵심 역량이다.** 단순 반복은 기계적인 반복을 뜻한다. 모순 같지만 컴퓨터 공학자들이 인공지능에 대한 궁극적인 과제 (AI Grand Challenge)인 인간처럼 사고하는 컴퓨터를 만들기 전까지 컴퓨팅적 사고는 기계적 사고에 머물 것이다.
- **컴퓨팅적 사고는 컴퓨터가 아닌 인간의 사고방법이다.** 컴퓨팅적 사고는 인간이 문제를 해결하는 방법의 하나로 인간이 컴퓨터처럼 사고하는 것을 뜻하는 것이 아니다. 컴퓨터는 따분하고 지루한 반면 인간은 영리하며 상상력이 풍부하다. 인간은 컴퓨터를 흥미롭게 만들 수 있다. 우리는 컴퓨터 기기에 인간의 영리함을 불어넣어 컴퓨팅 시대 이전에는 상상도 못한 문제를 해결하려고 하고 있으며 이를 구현하는데 있어 우리를 가로막는 것은 상상력의 한계뿐이다.
- **컴퓨팅적 사고는 수학적 사고와 공학적 사고를 보완하고 결합한다.** 모든 과학 분야가 수학에 기초하고 있듯 컴퓨터 공학 역시 수학적 사고에 기반하고 있다. 또한 컴퓨터 공학은 실제로 사용될 시스템을 설계하는데 쓰이기 때문에 공학 기술적 사고에 기초하고 있기도 하다. 컴퓨터 엔지니어는 컴퓨팅 기기의 한계로 인해 수학적 사고와 컴퓨팅적 사고를 발휘할 수밖에 없다. 반면에 자유롭게 가상현실을 만들 수 있기 때문에 그들은 물질로 이루어진 세상을 초월한 시스템을 구상할 수 있기도 하다.
- **컴퓨팅적 사고는 인공물이 아닌 아이디어이다.** 우리가 만든 소프트웨어와 하드웨어만이 우리의 생활의 일부가 된 것이 아니다. 문제를 해결하기 위해, 일상 생활을 꾸려나가기 위해, 다른 이들과 소통하기 위해 발전된 컴퓨팅적 개념 또한 우리의 삶의 구석구석에 막대한 영향을 끼치고 있다.
- **컴퓨팅적 사고는 모두를 위한 것이다.** 컴퓨팅적 사고가 인간 활동에 필수 요소가 되어 더 이상 특수한 철학으로 존재하지 않을 때 그것은 자연스러운 삶의 일부가 될 것이다.

많은 사람들이 컴퓨터 과학을 컴퓨터 프로그래밍과 같은 개념으로 오해한다. 어떤 부모는 컴퓨터 과학을 공부하는 자녀가 갈 수 있는 직종의 폭이 극히 한정되어 있다고 생각한다. 그리고 많은 이들이 컴퓨터 과학의 핵심 연구는 더 이상 할 게 없고 이제 기술 공학적 문제만이 남았다고 생각하기도 한다. 이 사회가 바라보는 컴퓨터 과학에 대한 좁은 시선을 바꾸는 데에 컴퓨팅적 사고는 컴퓨터 과학 분야의 교육자, 연구자, 종사자들에게 길잡이가 되는 큰 비전이 되어줄 것이다. 우리는 특히 선생, 부모, 학생들에게 두 가지 중요한 사안을 전달해야 한다.

우리를 지적으로 자극하고 흥미를 주는 과학적 난제들은 여전히 남아 있으며 우리의 해결책과 깨달음을 기다리고 있다. 문제와 해결책의 영역은 우리의 호기심과 창의력에만 국한되어 있는 것이다.

컴퓨터 과학을 전공한 사람은 무엇이든 할 수 있다. 영어나 수학을 전공한 뒤 다양한 커리어를 찾아갈 수 있듯 컴퓨터 공학도 예외는 아니다. 컴퓨터 과학을 전공하고 의학, 법, 경제, 정치, 과학과 공학, 그리고 심지어 예술 분야에도 진출할 수 있다.

컴퓨터 과학 교수들은 “컴퓨터 과학자처럼 생각하기”와 같은 수업을 전공과 관계 없이 모든 대학 신입생들에게 가르쳐야 한다. 또한 대학 진학을 앞둔 학생들에게도 컴퓨팅적 방법론과 모델을 가르쳐야 한다. 지금은 컴퓨터 과학에 대한 관심과 연구자금이 점점 줄어드는 현실을 아쉬워할 때가 아니다. 우리는 대중에게 이 분야의 지적 탐험에 대한 영감을 심어줄 필요가 있다. 컴퓨터 과학의 힘과 그로 인해 얻을 수 있는 기쁨과 감동을 널리 알려 컴퓨팅적 사고를 일상의 영역으로 가져와야 한다. ■

용어 설명:

[1] 명령어 집합 (instruction set) - 컴퓨터 중앙 처리 장치 (CPU)가 인식해서 기능을 이해하고 실행할 수 있는 기계어 명령어
[2] 허위 양성과 허위 음성(false positive and false negative) - 통계적 오류에 대한 용어로 실제 음성인 것을 양성으로, 양성인 것을 음성으로 판정하는 오류
[3] 차원 해석(dimensional analysis) - 관련 변수들의 관계를 간결하게 나타내기 위한 수학적 기법
[4] 간접주소 지정(indirect addressing) - 명령어(instruction)의 어드레스부에 간접 어드레스를 넣어두는 어드레스 지정 방식
[5] 추상화(abstraction) - 복잡한 자료, 모듈, 시스템 등으로부터 핵심적인 개념 또는 기능을 간추려 내는 것
[6] 교착 상태 (deadlock) - 동일한 자원을 공유하고 있는 두 개의 컴퓨터 프로그램들이, 상대방이 자원에 접근하는 것을 사실상 서로 방해함으로써, 두 프로그램 모두 기능이 중지되는 결과를 낳는 상황
[7] 인터페이스 - 다른 장치 혹은 시스템 영역 등이 접점에서 서로 만나 영향을 주고 받는 것
[8] 경쟁 상태 (race condition) - 장치나 시스템이 두 개 이상의 동작을 동시에 수행하려고 시도했을 때 발생하는 바람직하지 않은 상태를 가리킨다. 그러나 장치나 시스템의 속성 때문에 이러한 동작들은 적절한 순서에 따라 바르게 완료 되어야만 한다.
[9] 튜링 테스트 - 기계가 인간의 지능이 있는지를 판별하기 위해서 대화법을 이용하는 테스트로, 앨런 튜링이 1950년에 제안했다.
[10] CAPTCHA - CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart, 완전 자동화된 사람과 컴퓨터 판별,캡차)는 사람인지 컴퓨터 프로그램인지를 구별하기 위해 사용되는 방법으로 텍스트나 이미지를 일그러뜨린 형태로 변형시킨 후 인식 대상이 판별해 낼 수 있는지를 확인하는 기법이다.
[11] 알고리즘 - 어떠한 문제를 해결하기 위한 여러 동작들의 유한한 모임

저자 지넷 윙 (Jeannette M. Wing)은 현재 카네기 멜론 대학 컴퓨터 과학 학과장이자 마이크로소프트 리서치의 부사장입니다.

위 글은 윙 교수가 2006년 ‘컴퓨팅적 사고 (computational thinking)’란 개념을 현대인이 갖춰야 할 핵심 역량으로 소개한 첫 발행물을 번역한 글입니다.

원문: Wing M. Jeannette. (2006). Computational Thinking. “Communications of the ACM”, 49(3), 33-35. [링크]

번역:
이지연, Ed.M
MOOC 유닛, NHN Next 재단

감수:
김인희, Ph.D Candidate
초중등 소프트웨어 유닛, NHN Next 재단

정호영, Ph.D.
초중등 소프트웨어 유닛, NHN Next 재단