안녕하세요?
지식캠퍼스의 김현철입니다.

지난 시간에 이어서 이번 시간에는
정보의 동작, 어떤 동작

즉, 정보의 흐름을
어떻게 정의하는가

그게 알고리즘이고 그게 자동화하고
연결되는 부분이거든요.

그래서 그 부분에 대한 얘기를
좀 나눠보도록 하겠습니다.

우리가 'Computational Thinking'
이라는 말을 우리말로 번역할 때

우리가 '컴퓨팅 사고력'이라고
번역했거든요.

그런데 어떤 분들은 또 '계산적 사고'
이렇게 번역하시는 분들이 있어요.

그 이유는 'Compute'라는 단어하고
그다음에 실제로 계산하는

우리가 더하기 빼기 계산하는
'Calculate'라는 단어가 있잖아요.

이 단어 두 개가 다 우리나라 말로
'계산'이라고 번역이 되어 있었습니다.

그래서 사실 근데 영어로 얘기하면
Compute 한다, 컴퓨팅

그다음에 Calculate 한다는
다른 의미를 가지고 있거든요.

그런데 우리는 같은 용어를 쓰니까
그런 혼란이 있습니다, 사실은요.

그래서 그것을 분리하기 위해서
컴퓨팅, Compute라고 할 때

과거에는 전자 계산, 계산은 계산인데
전자기기로 하는 계산, '전자 계산'

혹은 '전산' 이렇게
번역하기도 했습니다.

하지만 두 개의 의미는 그렇게
번역하면 안 될 것 같고요.

조금 더 어떤 의미인지,
어떤 차이가 있는지를

우리가 생각할 필요가 있습니다.

Calculate은 단순하게

계산을 하는 게 맞습니다.

그런데 우리가 Computing이라고
얘기를 하면요.

그것은 일반적인 계산하고는 달리
무엇인가를 자동화하는 형태

뭐가 자동화될 수 있는가?
무엇을 자동화시킬 수 있는가?

그런 자동화의 관점으로

Computing이라는 단어가
사용되고 있습니다.

즉, 어떠한 작업을 컴퓨터라는
전자 기계를 통해서

자동으로 처리하게 한다는 것이고요.

그다음에 컴퓨팅 모델이라고 얘기하면

거기서 자동으로 처리되는 어떤
그런 모델을 얘기하는 거예요.

다시 비교를 하면 Calculation,
계산은 뭐가 무엇을 사용하게 되냐면

식을 사용하게 됩니다.
계산식이 있습니다.

계산식에 데이터를 넣어서, 아니면
숫자를 넣어서 계산이 되게 하는 거죠.

Computing은요, 식에 대비되는 게
뭐냐 하면 '알고리즘'입니다.

우리가 지난 시간에 잠깐
살펴본 것처럼 알고리즘이에요.

알고리즘에 어떤
숫자 데이터를 집어넣으면

이 알고리즘에 의해서 자동으로
뭔가가 계산되는 것

그것을 우리가 컴퓨팅이라고
얘기할 수 있습니다.

알고리즘을 표현하는 방법이 '순서도'
라는 것을 주로 사용하고 있는데요.

이것을 과정으로, 이 순서도를 통해서

어떻게 컴퓨팅이 될 수 있는지,
데이터가

그리고 자동으로 어떤 작업이
이루어질 수 있는지를

살펴보도록 하겠습니다.

순서도는 일이 순서대로,
절차대로 흘러가는 것

즉, 흐름을 정의한 그런 모습이라고
얘기를 할 수가 있습니다.

자, 이 그림을 한번 보세요.

제가 순서도를 한번 띄워 놨는데요.

어떤 두 개의 값에 평균을 내서

평균이 100점이 넘으면 잘했다,
평균이 100점이 안 되면 못했다

이렇게 얘기해 주는 거라고
생각을 해 보세요.

그런데 가만히 보세요.

그 숫자가 어떤 것인지를
알 수가 없어요.

어떤 것인지 알 수 없는데
어떤 두 숫자가 들어오면

무조건 그 안에서는
평균을 내서 판단을 합니다.

굿(Good)인지,
배드(Bad)인지를요.

이거 계산 관점으로도
생각할 수가 있겠지만

뭔가 컴퓨터에서 자동으로
처리되게 하는 거예요.

그래서 순서도로 만든 것을
한번 보세요.

먼저 시작을 했고 그다음에
평행사변형이 보이죠.

평행사변형은
데이터를 받는 거예요.

A, B 값을 누군가 주는 거죠.

사람이 줄 수도 있고,
다른 것에서 줄 수도 있고.

A, B를 받는 것이 평행사변형
형태로 되어 있고

밑에 사각형은 뭔가 일을
처리하는 거예요.

그래서 보게 되면 A, B의 합을

2로 나눠서

그것을 '애버리지'(average)라고 하자.
라고 해놓은 거고요.

그다음에 다이아몬드 형태가 있죠.
그것은 판단하는 거예요.

"이거야? 이거야?"라고
질문을 하는 거예요.

그래서 그 질문을 보면 'average가
100보다 같거나 크냐?'라고 물었고요.

거기에서 갈래가 두 개가 나오죠?

Yes 또는 No가 나오죠.

Yes 하면 Good이야,
밑에 이렇게 꼬부라진 형태가

출력을 하는 거거든요.
Good이야 하는 것이고.

No면 bad야.
이렇게 하고 끝나는 겁니다.

자, 이 그림을 가만 보세요.

자동화 기계인데, 소프트웨어인데

두 개의 숫자를 받아서 평균 내서
100이 넘는지 안 넘는지에 대해서

Good인지 bad인지를 판단하는,
그것도 자동으로.

숫자가 100만 개가 들어가도 돼요.

연속해서 두 개씩 계속
들어갈 수가 있죠.

자동으로 계속 처리를 합니다.

옛날에 이것을 사람이 했다고
생각을 해 보세요, 직접.

얼마나 시간과 노력과
그런 게 들었겠습니까?

소프트웨어가 자동으로
할 수가 있습니다, 이런 일들을요.

소프트웨어를 몇 개 더 보도록 하죠,
이런 관점에서.

아, 알고리즘을요.

알고리즘은 아까 우리가 본
예를 가지고 살펴보면

마치 함수하고 좀 비슷한 것 같아요.

우리가 수학에서 배운
함수를 생각해 보세요.

함수는 y=f(x)라고 했잖아요.

y=f(x)란 것을 가만히 보세요.
거기서 x는 인풋(input)이거든요.

입력이 어떤 값이 x란 값이 들어가,
그다음에 f에 의해서 그 x값이

뭔가가 가공이 되고 변형이 돼서
결과가 y가 나오는 거예요.

그리고 그 f라는 것은 x에 의해서
결정이 되는 것인데

그것을 우리가 정의를 한 것이
함수라고 얘기를 했죠.

그림을 한번 보세요.

입력값이 들어가서 뭔가가
처리가 돼서 출력이 나오는 거죠.

함수와 거의 비슷합니다.

그리고 입력값을 우리는
인풋(input)이라고 하고

처리를 프로세스(process)라고
얘기하고

출력을 아웃풋(output)이라고
얘기합니다.

그래서 이러한 과정을
'IPO 차트'라고도 얘기합니다.

IPO, Input, Process
그다음에 Output 이렇게요.

자, 이 관점에서 다시 이제
순서도를 보도록 하죠.

첫 번째 순서도를 한번
보도록 하겠습니다.

이 순서도를 보게 되면, 게임이에요.

어떤 숫자를 컴퓨터가 랜덤으로
무작위로 만들어 내고

내가 그 숫자가 "이거야? 이거야?"
라고 맞추는 거라고 해 보세요.

그래서 맞으면 잘했다고 그러고,
'빙고!' 그러고

틀렸으면 '다시 해 봐, 다시 넣어 봐'
이렇게 해 주는

어떤 기계라고 생각해 보세요.

게임 기계라고도 얘기할 수 있고
사실은 소프트웨어죠.

속에서 Compute를 하는 거죠.

자, 그러면 한번 봅시다.

첫 번째, 넘버(number)를
뭐라고 하냐면

넘버는 0에서 100까지 임의로 그냥
무작위로 작동한 거예요.

그게 넘버예요.

그러면 평행사변형이 나오죠.

뭔가 데이터를 입력을
해야 되는 거예요.

숫자를 맞춰봐, 그러면 제가
숫자를 뭐라고 집어넣죠.

그거를 게스(guess)라고 하고

그러면 컴퓨터가
제너레이트(generate)한 넘버가 있고

내가 집어넣은 게스(guess)가
맞는지만 보면 되잖아요.

게스가 '넘버냐?'라고 물어서
Yes면 "빙고!" 맞췄네.

아니면 '다시 한 번 해 봐'
반복되는 거죠.

자, 이 구조를 보세요.

로드가 있고 화살표로
그 순서가 정해져 있습니다.

이 순서대로만 그대로 하면 숫자를
맞추는 자동 기계가 되는 걸까요?

자동 게임 기계가 되는 걸까요?
됩니다, 그렇죠?

다음 예를 한번 봅시다.

조금 더 어렵습니다.

무슨 기계를 만들고 싶냐면,
무슨 소프트를 만들고 싶냐면

어떠한 내가 'n'이라는

숫자를 집어넣으면

n이 뭐 10이건, 100이건
50이건 집어넣으면

1에서부터 그 숫자까지의 합을
1+2+3+5+... 계속 나가서 50까지.

그 합을 출력으로 내는
그런 기계를 만들고 싶어요.

그것을 옛날에 사람이 일일이
손으로 했다고 해보세요.

얼마나 힘들었겠습니까?

Calculation 했다고 그러면
얼마나 힘들겠어요?

그런데 Calculation이 아니고
컴퓨팅을 하고 싶은 거예요.

자동으로 그 일을 하는
그것을 만들고 싶은 거예요.

순서도로 이렇게 정의를 하면 됩니다.
한번 봅시다.

처음에 또 평행사변형이 나왔습니다.

평행사변형은 어떤 값을 내가
집어넣어 줘야 되는 거예요.

n을 집어넣습니다.

아, 1에서부터 n까지의 합을
계산하려고. n, 내가 집어넣습니다.

그러면 그다음에 뭐라고 나왔냐면
i=0이고, i=index인데요.

조금 어렵습니다.
그냥 들어만 보세요.

sum=0입니다.

그러면 그다음에
다이아몬드가 나옵니다.

또 물어보는 거죠.

'i가 n보다 크냐?'
라고 물어보는 거죠.

i가 n보다 안 크죠, 그렇죠?

그러니까 밑으로 내려와서
sum=sum+i

그다음에 sum을 하나
추가시켜주는 거예요.

그리고 반복해서
그럼 'i가 n보다 이제 크냐?'

'i가 n보다 크냐?'
언제까지 반복합니까?

i 하고 n이 똑같을 때까지
반복하는 거죠.

그리고 똑같을 때까지 계속 i가,
1이 n 번이 될 때까지 반복하는데

그 반복하는 과정에서 그 i 값이
계속 sum에다가 더해지는 거죠, 그렇죠?

그런 다음에 이제 더 커지면
Yes 해가지고 밑으로 와서

여태까지 합해진 그 합을
출력을 하는 겁니다.

어렵죠?

개념적으로, 논리적으로.
그런데 너무나 간단한 형태 아닙니까?

이 그림을 한번 보세요. 그렇죠?

저것을 만들어 놓으면 어떠한 사람이
전 세계 100만 명이 한꺼번에

'나는 이 숫자까지의 합을
더해 주세요.' 집어넣으면 자동으로

계속 반복해서 컴퓨터가
이 일을 해 주게 되는 거죠.

인지적인 노동을 대신해 주고 있는
정보의 흐름을 정의한 겁니다.

이렇게 숫자만 가지고 우리가
할 수 있는가? 또 그건 아니에요.

숫자뿐만이 아니고
다양한 것을 할 수가 있습니다.

예를 들면, 로봇을 내가
자동으로 걷는 것을 만드는데

벼랑에 떨어지지 않는
로봇을 만든다고 해 보세요.

제가 장난으로 한번 써 본 건데
이렇게 쓸 수가 있습니다.

한 발 전진,
인풋이 안 들어갑니다.

인풋이 없이 그냥
한 발 전진하는 거예요.

그다음에 벼랑 끝이냐?
아니야, 그럼 또 한 발 전진.

또 한 발 전진.
계속 반복을 하는 거죠.

그래서 언젠가는 벼랑 끝을 만나겠죠?

벼랑 끝이냐?
그러면 Yes 멈춘다, 끝.

그러면 이것의 작동은 뭡니까?

한 발씩 계속 작동을 해서
벼랑 끝까지 가서 멈추게 하는 일을

내가 정의를 해놓은 거예요.

이것을 어떤 로봇이든
로봇이 백만 개든, 천만 개든

거기다가 집어넣으면
그 로봇은 그렇게 행동합니다.

우리의 인생도 알고리즘으로, 순서도로
한번 만들어 볼 수 있을 겁니다.

자, 그리고 이 그림에서 IPO,
input이 뭐가 들어갔고

process는 뭐고, output이 뭔지
여러분들이 구분해서

생각할 수 있으면 좋겠습니다.

그리고 이러한 알고리즘들은
엄청나게 많습니다.

그리고 어려운 알고리즘들도 많고요.

그 어려운 알고리즘들을
쉽게 이해하기 위해서

최근에 많이 사용되고 있는
교수학습 방법 중에 하나가

바로 언플러그드(unplugged)입니다.

언플러그드는 굉장히 어려운 개념들을
어떤 놀이 활동을 통해서

어떻게 일들이 처리가 되는지,
어떤 순서대로.

그다음에 무엇을 고려해야 되는지를
만들어진 그런 활동들을

모아놓은 것들이 있는데 그것을
언플러그드라고 얘기하고

여기에 예로 나온 것 중에서
파인 엣 오토메타 같은 경우는

제가 대학교 때도 굉장히 어렵게
배웠었던 것들이거든요.

그런데 활동을 통해서 하니까
8살~11살짜리 아이들도

쉽게 그 개념과 알고리즘을 이해
할 수 있었다, 뭐 이런 것들이고요.

최근에 많은 인터넷 자료에
들어가서 보면

이러한 언플러그드를 이용한
알고리즘에 대한 설명, 활동들이

굉장히 많이 나와 있어서
여러분들이 원하시면

다양한 인터넷 사이트에 들어가서
그런 것들을 실제로 해보고

알고리즘을 이해하고
그렇게 해 볼 수가 있습니다.

알고리즘은 컴퓨팅 사고력에서
가장 핵심적인 부분입니다.

여러분들도 여러분들 머릿속에서
혼자 남들보다 잘하는 것

알고리즘으로 만들어 보고
표현할 수 있게 되면

여러분들은 컴퓨팅 사고력의
굉장히 중요한 부분을

한 것이라고
생각할 수가 있습니다.