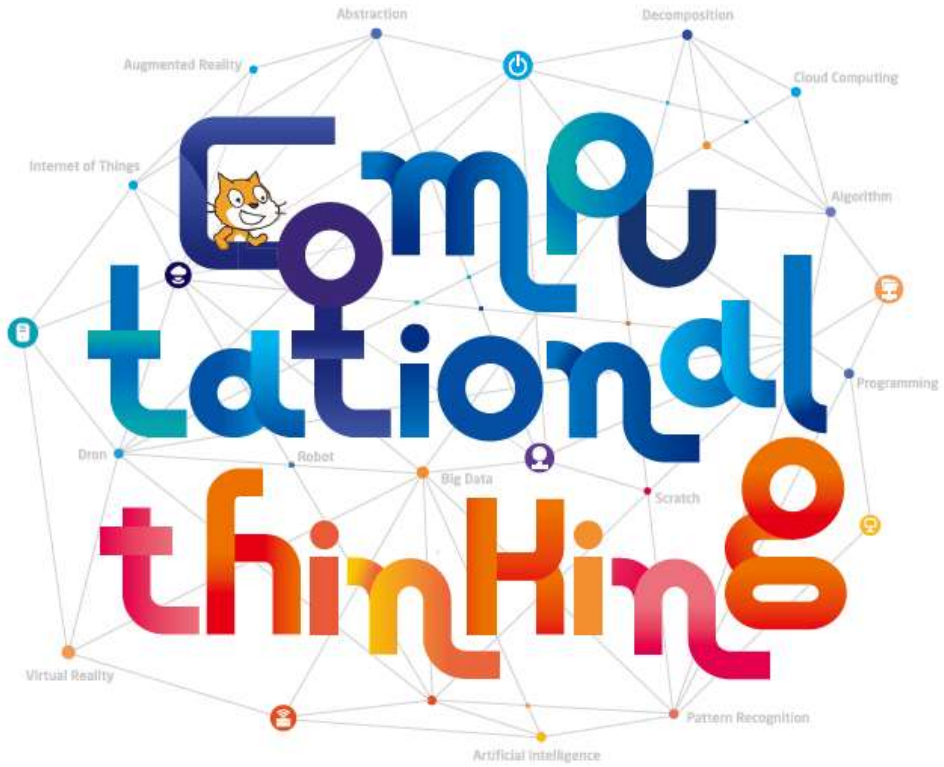
한빛아카데미 Hanbit Academy, Inc.



스크래치로 배우는

컴퓨팅 사고

Part 01. 컴퓨팅 사고 이론

Chapter 04. 알고리즘과 프로그래밍 언어

목차

01
알고리즘의 이해

01. 알고리즘의 이해

■ 알고리즘의 개념

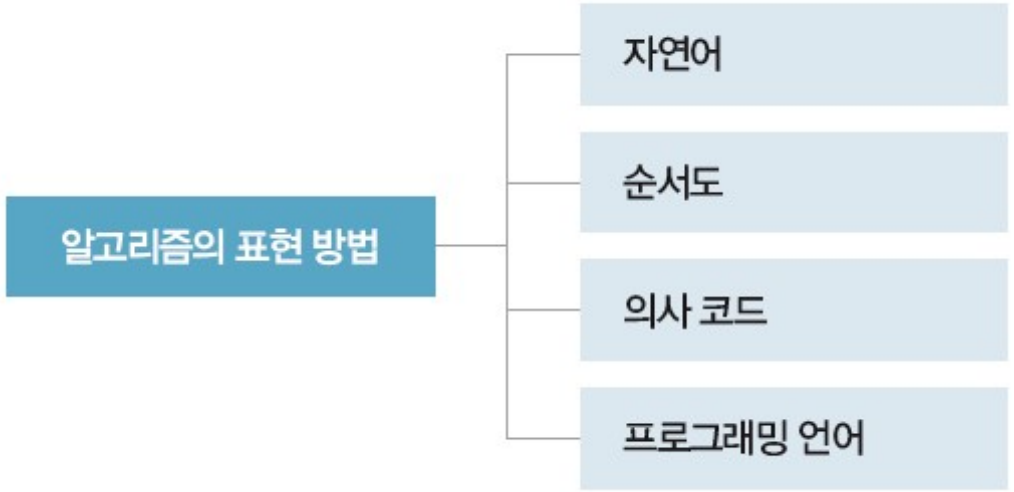
- **알고리즘(algorithm) :** 주어진 문제를 어떻게 해결할 것인지에 대해 그 방법과 절차를 기술한 것이다.

[그림 4-1] 라면 끓이기 알고리즘

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

- 알고리즘을 표현하는 방법에는 자연어, 순서도, 의사 코드, 프로그래밍 언어를 사용하는 방법이 있다.



[그림 4-2] 알고리즘의 표현 방법

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

* 자연어

- **자연어(natural language) :** 사람들이 일상생활에서 사용하는 언어로, 프로그래밍 언어와 구별하기 위해 사용한다.
- Z를 사용하여 X와 Y의 값을 바꾸는 알고리즘을 자연어로 표현한 것이다

```
시작
        X에 3, Y에 5를 대입한다.
        X의 값을 Z에 대입한다.
        Y의 값을 X에 대입한다.
        Z의 값을 Y에 대입한다.
        X와 Y 값을 출력한다.
끝
```

[그림 4-3] 자연어 사용 예시

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

* 순서도

- **순서도(flow chart) :** 약속된 기호와 선을 사용하여 문제 해결 과정을 표현하는 방법이다. 순서도를 사용하면 자연어로 파악하기 어려운 전체 구조의 흐름을 한번에 파악할 수 있다.

[표 4-1] 순서도의 기호

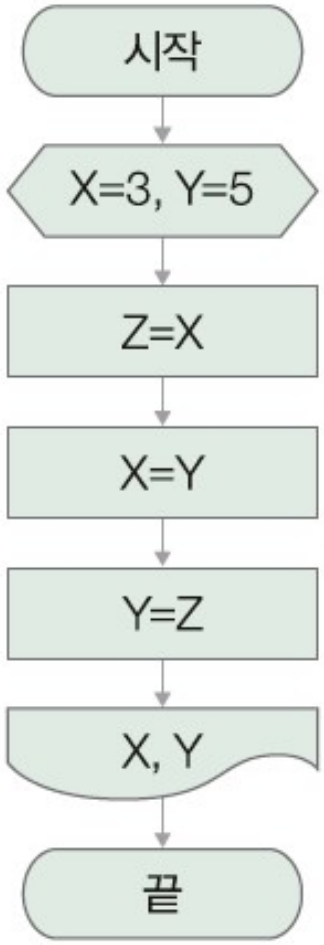
기호	명칭	의미
	단말	순서도의 시작과 끝을 의미한다.
	흐름선	각 기호를 연결하며, 순서도의 흐름을 나타낸다.
	처리	계산 등 자료의 연산 또는 처리를 나타낸다.
	준비	변수의 초깃값, 기억 장소의 설정 등 작업의 준비 과정을 나타낸다.
	판단	조건을 판단하여 '예' 또는 '아니오'로 이동한다.
	입출력	자료의 입력과 출력을 나타낸다.
	출력	출력장치를 통한 출력을 나타낸다.

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

* 순서도

- [그림 4-4]는 [그림 4-3]의 자연어를 순서도로 표현한 예이다.
- 순서도는 알고리즘의 흐름을 빠르게 파악할 수 있다는 장점이 있지만, 복잡한 프로그램을 순서도로 작성하기 까다롭다는 단점도 있다.



[그림 4-4] 순서도 사용 예시

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

* 의사 코드

- **의사 코드(pseudo code) :** 특정 프로그래밍에 사용되는 언어와 유사한 서술로 알고리즘을 표현한 것으로, 프로그래밍 언어를 흉내낸 것이다.

```
START
        X=3, Y=5
        Z=X
        X=Y
        Y=Z
        PRINT X, Y
END
```

[그림 4-5] 의사 코드 사용 예시

01. 알고리즘의 이해

■ 알고리즘의 표현 방법

* 프로그래밍 언어

- **프로그래밍 언어(programming language) :** 컴퓨터에게 작업을 지시하기 위해 사용하는 언어이다. 의사 코드를 작성하면 특정한 프로그래밍 언어로 쉽게 변환할 수 있다.

01. 알고리즘의 이해

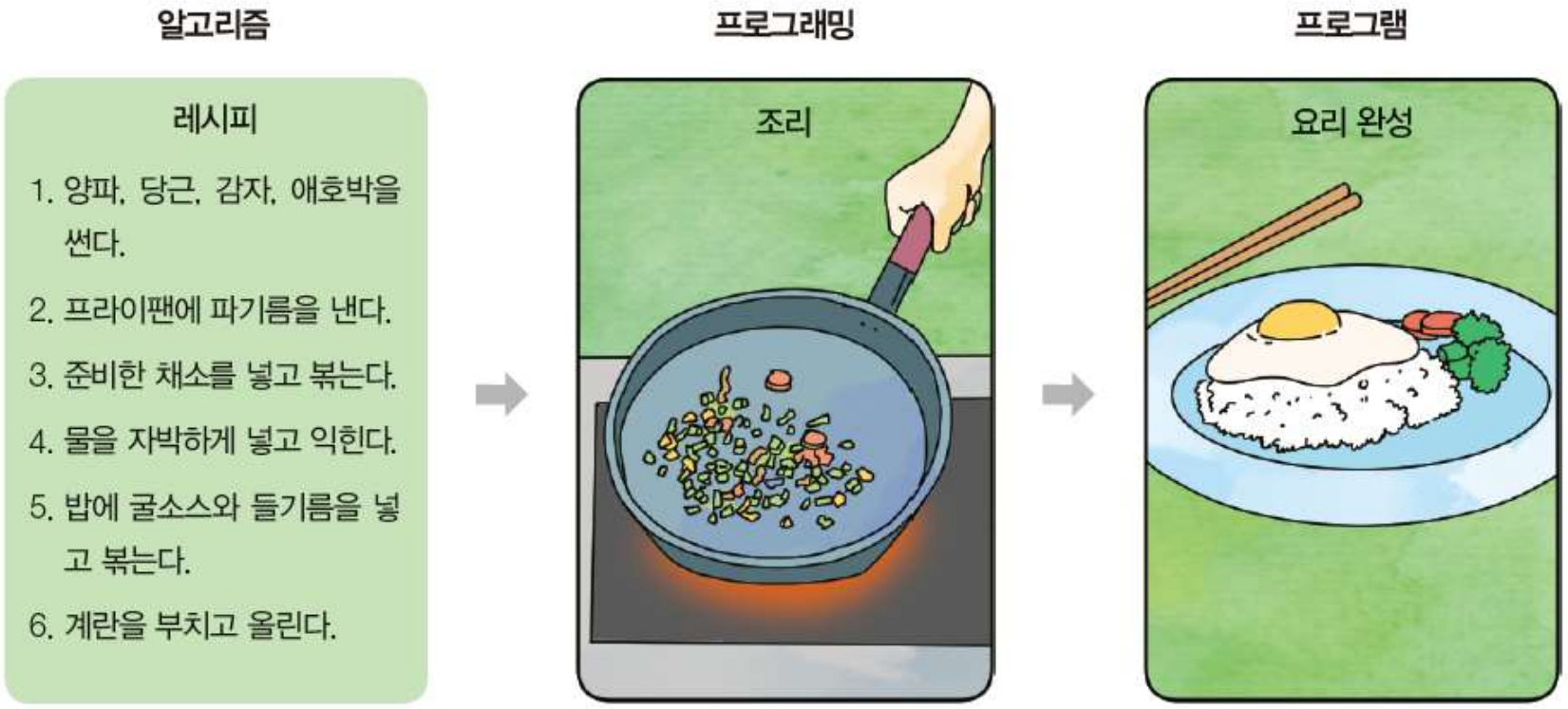
■ 알고리즘의 조건

- **입력:** 알고리즘에 입력되는 자료가 0개 이상 존재한다.
- **출력:** 알고리즘이 실행되면 결과 값이 1개 이상 나온다.
- **유한성:** 알고리즘은 종료되어야 한다.
- **명확성:** 알고리즘의 명령이 모호하지 않고 명확해야 한다.
- **수행 가능성:** 알고리즘의 명령은 수행 가능해야 한다.

01. 알고리즘의 이해

■ 알고리즘과 프로그래밍의 관계

- 알고리즘이 레시피라면, 프로그래밍은 레시피에 적힌 대로 조리하는 과정과 같고, 완성된 요리는 프로그램과 같다.



[그림 4-6] 알고리즘과 프로그래밍의 관계

02
알고리즘의 설계 및 분석

02. 알고리즘의 설계 및 분석

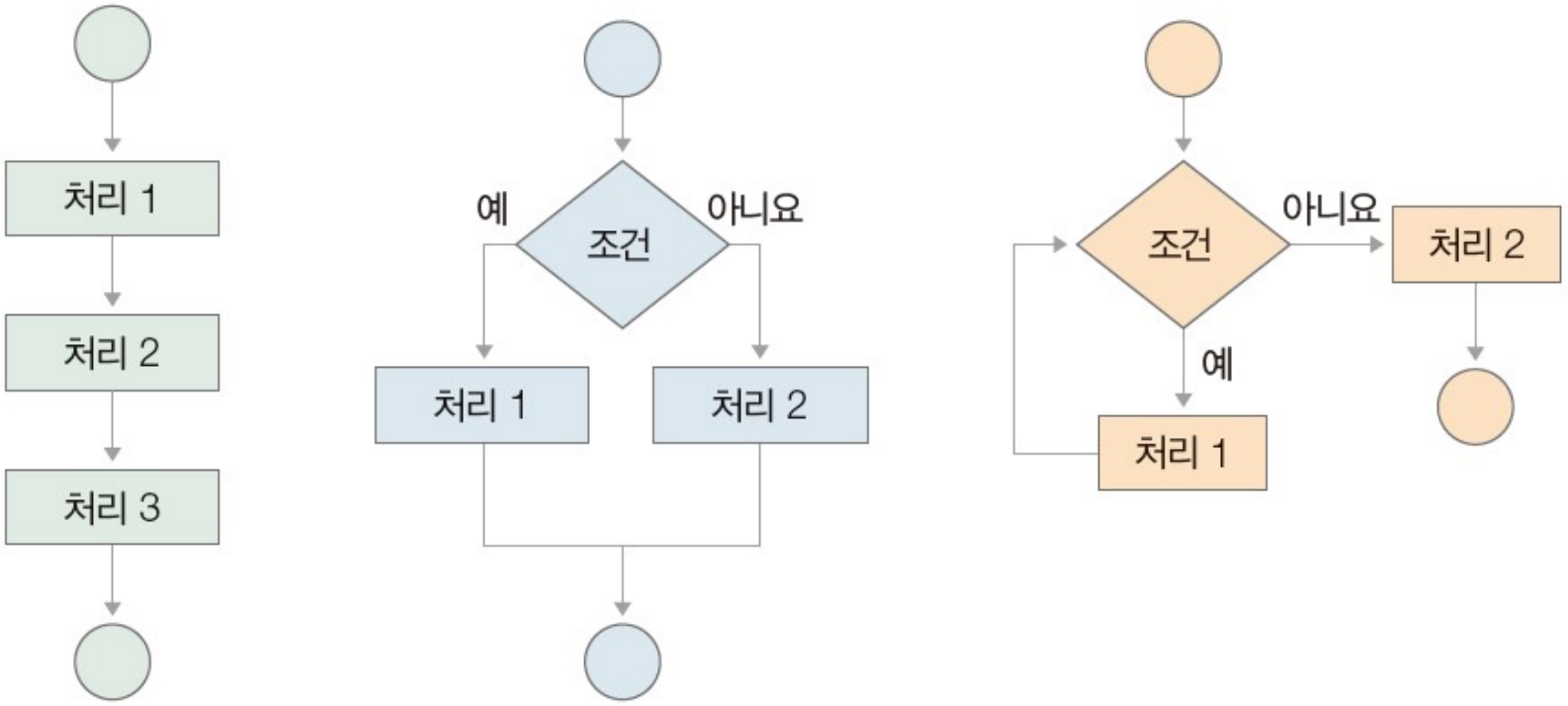
■ 알고리즘의 설계

- 알고리즘의 설계란 문제를 해결하기 위해 가장 효율적인 방법을 찾아내는 과정을 말한다.
- 알고리즘을 설계할 때에는 문제의 현재 상태와 목표 상태를 명확히 정의해야 한다.

02. 알고리즘의 설계 및 분석

■ 알고리즘의 설계

- 알고리즘을 설계할 때 알고리즘의 명령이 실행되는 순서를 결정하는 제어 구조로 순차 구조, 선택 구조, 반복 구조가 있다.



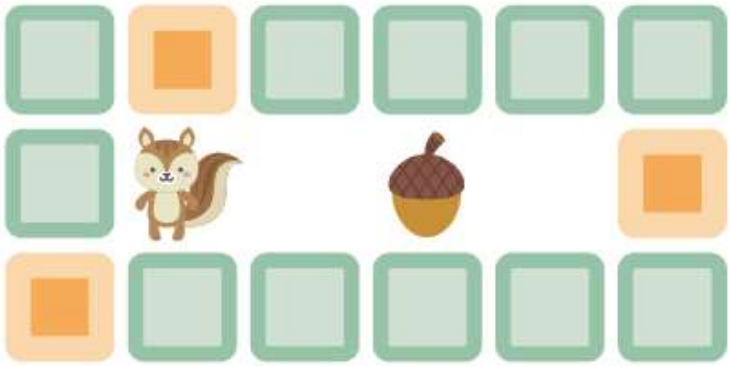
(a) 순차 구조 (b) 선택 구조 (c) 반복 구조

[그림 4-7] 제어 구조를 활용한 알고리즘

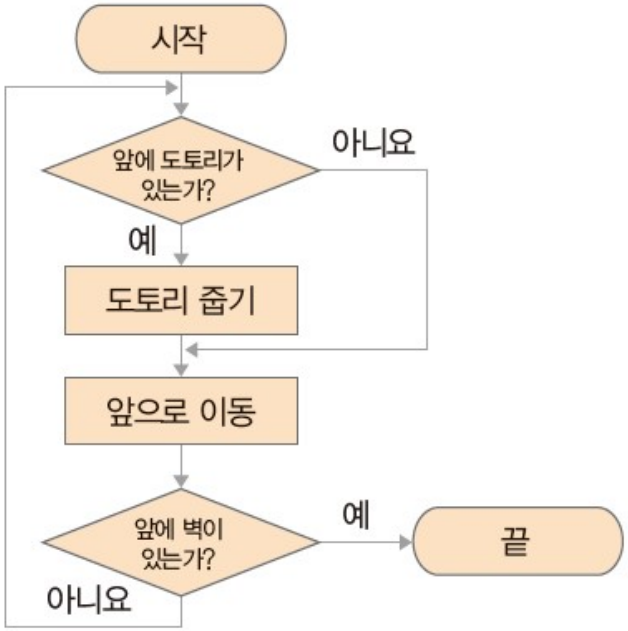
02. 알고리즘의 설계 및 분석

■ 알고리즘의 설계

- [그림 4-8](a)와 같이 다람쥐가 밀폐된 공간에서 도토리를 줍는 과정을 순차 구조, 선택 구조, 반복 구조를 이용하여 순서도로 설계하면 [그림4 -8](b)와 같다.



(a) 밀폐된 공간의 다람쥐와 도토리



(b) 순서도 구현

[그림 4-8] 순서도로 도토리 줍기 알고리즘 설계

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

- 하나의 문제 상황을 해결하기 위한 알고리즘이 다양하게 나타날 수 있는데, 이때 각 알고리즘을 비교하고 분석하는 과정을 통해 가장 효율적인 알고리즘을 선택하는 것이 중요하다.

[그림 4-9] 내비게이션 비교

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

- 컴퓨터 과학 분야에서는 프로그램을 실행하여 결과가 나올 때까지의 실행 시간이 짧고 컴퓨팅 기기의 기억 장소를 적게 사용하는 것을 효율적인 알고리즘이라고 한다.
- 가장 효율적인 알고리즘을 선택하기 위해서는 시간 복잡도와 공간 복잡도를 분석한다.

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

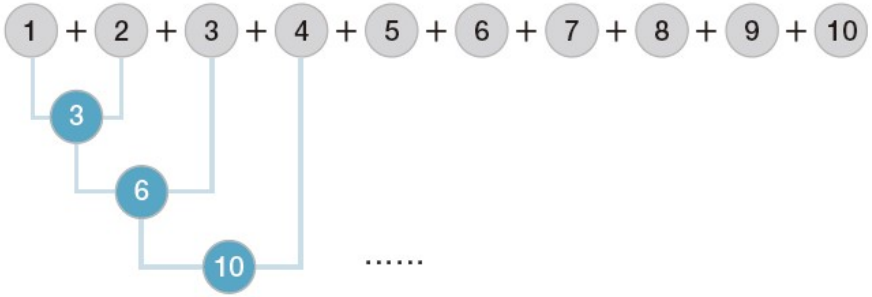
* 시간 복잡도

- **시간 복잡도 :** 알고리즘이 실행되어 종료될 때까지 어느 정도의 시간이 필요한지 측정하는 방법이다.
- 실제 컴퓨터의 실행 시간을 측정하기는 어렵기 때문에 시간 복잡도는 알고리즘의 실행문이 몇 번 실행되는지 횟수를 표시하는 방법을 사용한다.

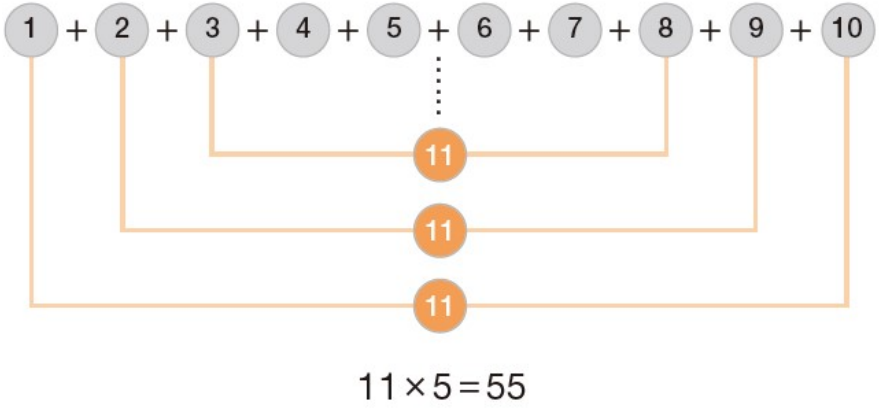
■ 알고리즘의 분석

* 시간 복잡도

- 1부터 10까지의 수를 더하는 두 가지 알고리즘을 비교하면 어느 방법이 더 효율적일까?



(a) 방법 1



(b) 방법 2

[그림 4-10] 1부터 10까지의 수를 더하는 두 가지 알고리즘

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

* 시간 복잡도

여기서 잠깐 **빅 오 표기법**

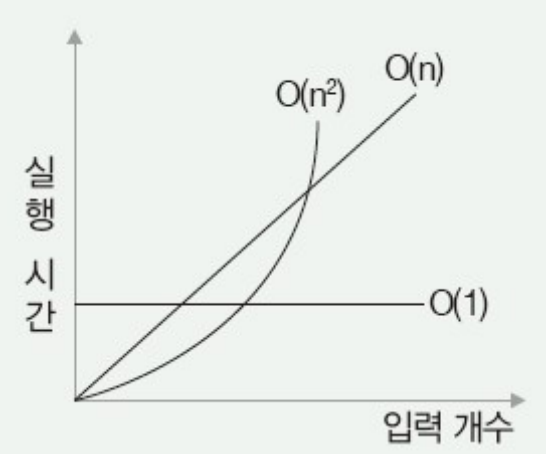
빅 오 표기법(Big O notation)은 알고리즘의 시간 복잡도를 표현하는 방법이다.

컴퓨터에서는 입력되는 값의 크기에 따라 알고리즘이 처리되는 횟수가 얼마나 증가하는지를 나타낸다.

- *O(n): 알고리즘의 수행 횟수도 n만큼 커진다.*

- *O(n2): 알고리즘의 수행 횟수도 n2만큼 커진다.*

O(n)이 O(n2)보다 시간 복잡도가 더 좋은 알고리즘이라는 것을 알 수 있다.



[그림 4-11] 빅 오 표기법 그래프

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

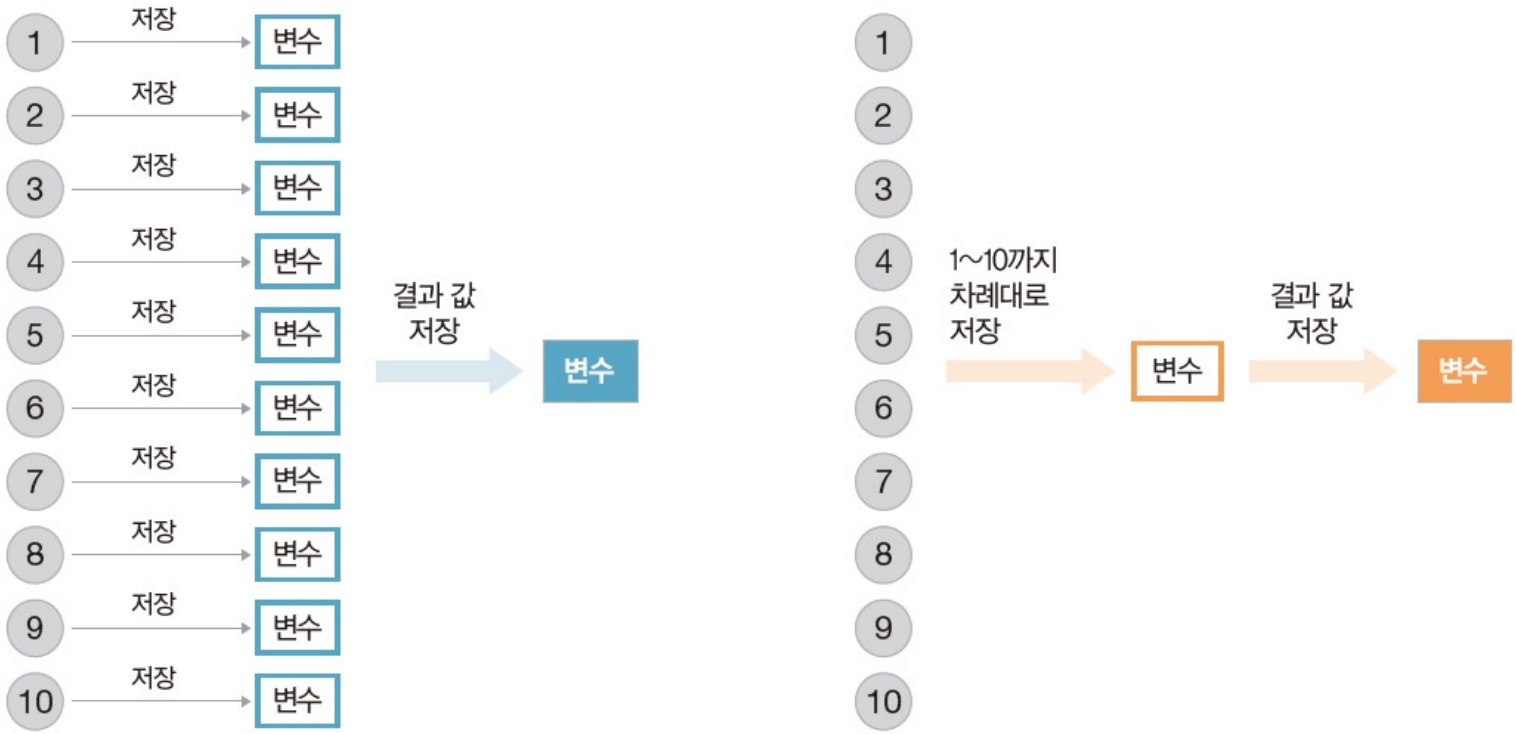
* 공간 복잡도

- **공간 복잡도 :** 알고리즘이 문제를 해결하는 데 어느 정도의 저장 공간을 필요로 하는지 측정하는 방법으로, 기억 장치 내의 공간을 얼마나 적게 사용하는지가 중요하다.

■ 알고리즘의 분석

* 공간 복잡도

- [그림 4-12]는 1부터 10까지의 수를 더하는 알고리즘에서 수를 저장하는 방법을 나타낸 것이다.



(a) 방법 1 (b) 방법 2

[그림 4-12] 1부터 10까지의 수를 더하는 알고리즘에서 수를 저장하는 방법

02. 알고리즘의 설계 및 분석

■ 알고리즘의 분석

* 공간 복잡도

여기서 잠깐 **좋은 알고리즘의 조건**

좋은 알고리즘이란 시간 복잡도와 공간 복잡도가 좋아서 처리 효율이 높은 것을 말하며, 그 밖에 다음의 조건을 만족해야 한다.

① **신뢰성:** 정밀도가 높고 올바른 결과를 얻어야 한다.

② **일반성:** 특수한 상황뿐만 아니라 다양한 상황에서도 사용할 수 있어야 한다.

③ **확장성:** 수정이 간단하고 다른 알고리즘과 결합하기 쉬어야 한다.

03

알고리즘을 이용한 문제 해결

03. 알고리즘을 이용한 문제 해결

■ 알고리즘 설계 실습: 스무고개 게임

- 스무고개 게임은 출제자가 1에서 10까지의 숫자 중 하나를 생각한다. 참가자는 출제자가 생각한 숫자를 맞추는 게임이다. 만약 참가자가 말한 숫자가 생각한 숫자보다 작으면 '작다', 크면 '크다'라고 출제자가 알려준다. 이러한 과정을 반복하여 출제자가 생각한 숫자를 찾아가는 게임이다.

[그림 4-13] 스무고개 문제 분해

[그림 4-14] 스무고개 게임의 패턴

03. 알고리즘을 이용한 문제 해결

■ 알고리즘을 이용한 문제 해결 과정

[그림 4-15] 알고리즘을 작성하여 문제를 해결하는 과정

03. 알고리즘을 이용한 문제 해결

■ 알고리즘을 이용한 문제 해결 과정

① 알고리즘 작성 : 주어진 문제에 대해 알고리즘을 짠다. 글을 쓰듯이 간단히 서술해도 되고 의사 코드 형태로 만들어도 된다.

② 알고리즘 검토: 알고리즘이 만들어지면 머릿속에서 돌려본다. 마치 내가 컴퓨터가 된 듯 한 줄씩 실행해보는 것이다.

③ 프로그래밍 및 실행: 알고리즘을 검토한 후 문제가 없다고 판단되면, 마지막으로 프로그래밍 언어로 코드를 작성하고 실행한다. 이 단계에서는 자신이 푼 문제가 맞았는지 꼭 확인한다.

만약 프로그램을 실행했는데 에러가 발생한다면, 단계 ❶로 돌아가 코드를 수정한다. 문법 에러는 에러 종류를 파악하여 입력 창에서 수정하면 된다. 만약 문법 에러가 없는데도 원하는 값을 얻지 못하거나 이상한 값이 출력되면 코드에 에러를 의심해본다. 코드 에러는 버그(bug)라 부르며, 버그를 없애는 과정을 디버깅(debugging)이라 부른다. 디버깅을 통해 자신이 실수한 부분을 찾아내어 단계 ❶~❸을 반복한다.

04
프로그래밍 언어

04. 프로그래밍 언어

■ 프로그래밍 언어의 개념

- **프로그래밍 언어(programming language) :** 컴퓨터에게 작업을 지시하기 위해 사용하는 언어를 말한다.
- 기계어, 어셈블리어, C, C++, 자바, 파이썬, 스크래치와 같이 다양한 언어가 있다.

[그림 4-16] 다양한 컴퓨터 언어들

04. 프로그래밍 언어

■ 저급 언어와 고급 언어

- **어셈블리어(assembly language) :** 기계어를 사람들이 이해할 수 있는 문자 형태로 바꾼 것이다. 어셈블리어는 숫자를 문자로 바꾸었을 뿐이어서 사람이 이해하기 어려운 구조로 되어 있다.
- **저급 언어(low level language)** : 어셈블리어와 같이 기계어에 가까워 사람이 이해하기 힘든 언어를 통틀어 저급 언어라 부른다.

(a) 기계어　　(b) 어셈블리어

[그림 4-17] 저급 언어

04. 프로그래밍 언어

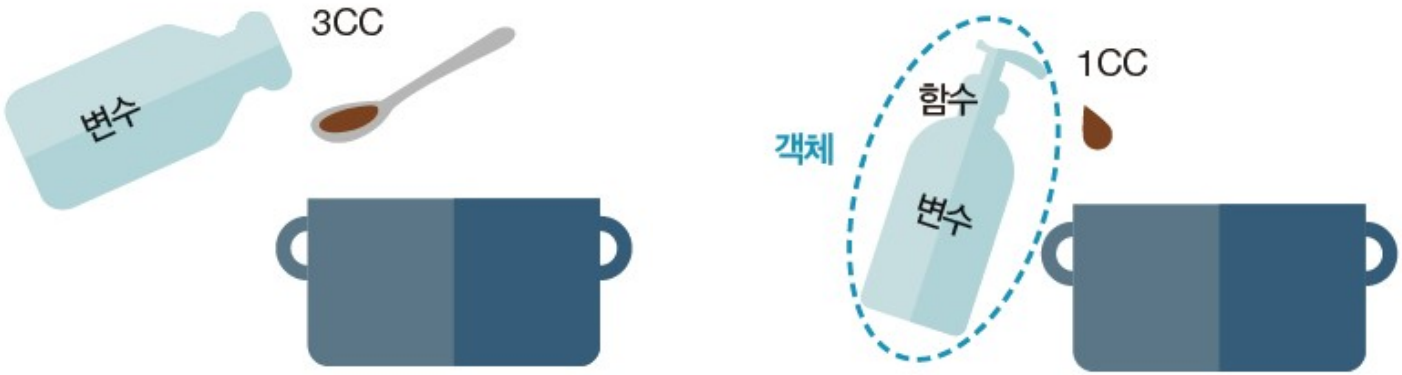
■ 저급 언어와 고급 언어

- **고급 언어(high level language) :** 사람이 사용하는 단어를 사용하여 이해하기 쉽도록 만든 언어를 말한다.
- 기계어와 어셈블리어를 제외한 대부분의 언어가 고급 언어이다.
- 고급 언어 중 가장 많이 알려진 언어가 C 언어이다.

04. 프로그래밍 언어

■ 객체지향 언어

- C 언어를 객체지향 언어로 바꾼 것이 C++ 언어다.
- **객체지향 언어(Object Oriented Language) :** 데이터를 담는 통과 통에 담긴 데이터를 처리할 수 있는 함수(메소드)를 하나로 묶어 놓은 것이다.



(a) 일반 언어 (b) 객체지향 언어

[그림 4-18] 일반 언어와 객체지향 언어의 차이

04. 프로그래밍 언어

■ 컴파일러와 인터프리터

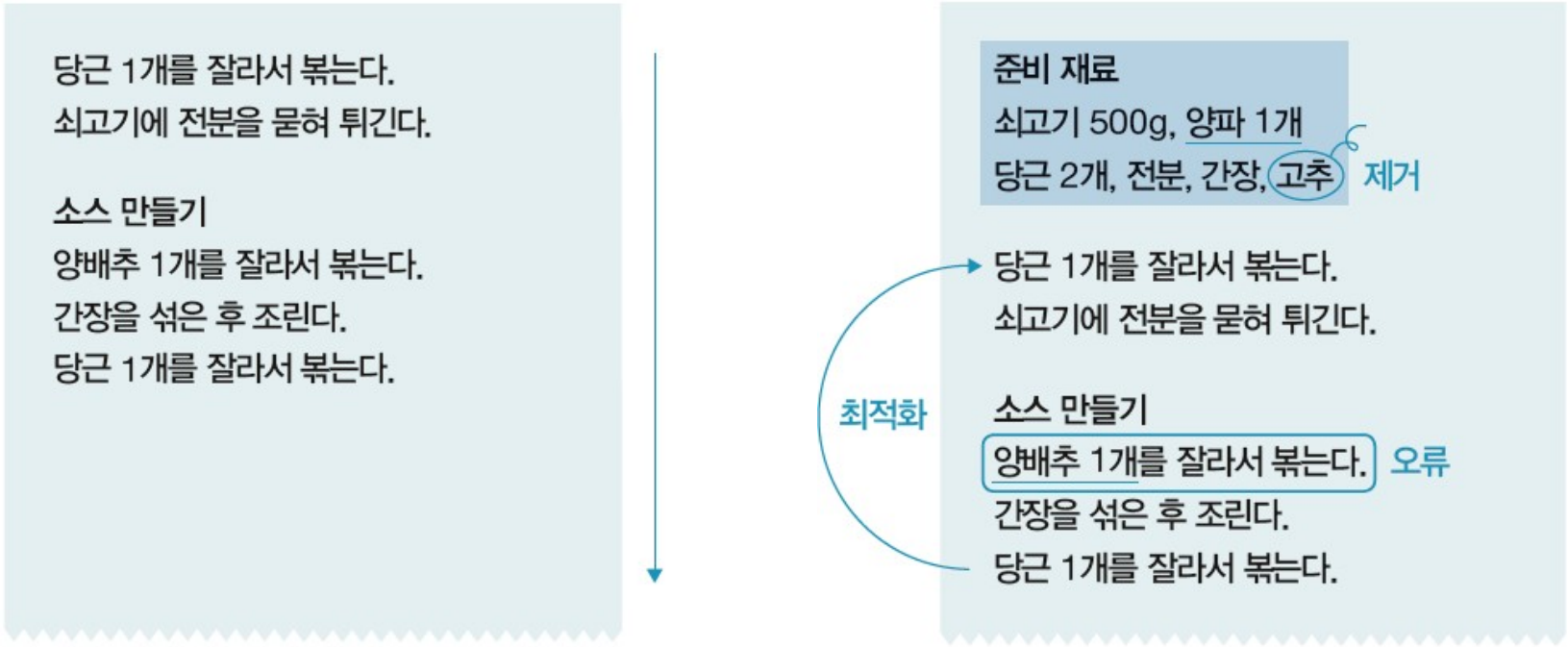
- 대표적인 언어 번역 방식에는 컴파일러와 인터프리터가 있다.
- **컴파일러(compiler) :** 소스코드를 컴퓨터가 실행할 수 있는 기계어로 번역하여 실행 파일을 만든 후 한꺼번에 실행한다. C 언어, 자바 등이 이 방식을 사용한다.
- **인터프리터(interpreter) :** 소스코드를 한번에 한 행씩 번역하여 실행한다. 자바스크립트, 파이썬, 베이직 등이 이 방식을 사용한다.

(a) 컴파일러 (b) 인터프리터

[그림 4-19] 컴파일러와 인터프리터

■ 컴파일러와 인터프리터

- 다음은 인터프리터와 컴파일러의 동작을 레시피로 나타낸 것이다.



(a) 인터프리터 (b) 컴파일러

[그림 4-20] 인터프리터와 컴파일러의 차이

04. 프로그래밍 언어

■ 컴파일러와 인터프리터

- 컴파일러와 인터프리터의 차이는 컴파일러를 사용하는 자바와 인터프리터를 사용하는 자바스크립트를 비교하면 확인할 수 있다.

[표 4-2] 자바(컴파일러)와 자바스크립트(인터프리터)의 차이

구분	자바	자바스크립트
변수	변수를 선언해야 한다.	변수를 선언할 필요가 없다.
실행	컴파일 후 실행된다.	한 줄씩 실행된다.
장점	오류 찾기와 코드 최적화, 분할 컴파일에 의한 공동 작업이 가능하다.	실행이 편리하다.
사용 프로그램	대형 프로그램	간단한 프로그램

Thank You!

한빛아카데미 Hanbit Academy, Inc.