

## 자료구조 1장

### part1. 자료구조와 알고리즘

자료구조란? 데이터를 수용하여 저장하고 관리하기 위한 구조

-> 데이터를 잘 정리해 두는 방법 제공하여, 프로그램이 데이터를 더 빠르게 찾거나, 저장하거나, 계산하도록 도움.

- 1) 스택 : 데이터를 쌓아 올린 구조, **후입선출**(ex. 책을 쌓아두면 맨 위에 놓은 책을 먼저 꺼낼 수 있음.)
- 2) 큐 : 줄을 서는 구조, **선입선출**(ex. 먼저 줄 선 사람이 먼저 창구에서 서비스를 받음.)
- 3) 리스트 : **데이터를 순서대로 저장**, 배열과 유사하지만 크기 유동적으로 변경 가능함. (ex. 사고 싶은 물건을 순서대로 적어놓은 쇼핑 리스트)
- 4) 사전 : **키-값** 으로 데이터 저장함. 원하는 데이터를 키를 통해 빠르게 찾을 수 있음. (ex. 단어 사전)
- 5) 그래프 : **노드와 엣지로 이루어진 구조**. 노드는 점, 엣지는 점들을 연결하는 선(ex. 지하철 노선도 - 역(노드)와 역 사이의 선(엣지)로 구성)
- 6) 트리 : **계층 구조를 표현한 것**. 루드에서 시작해 가지처럼 뻗어 나가는 구조.(ex. 가계도)

자료구조 = 선형 자료구조 + 비선형 자료구조

선형 자료구조 : 스택, 큐, 배열 etc.

비선형 자료구조 : 그래프, 트리 etc.

< 자료구조와 알고리즘 >

```
#define MAX_ELEMENTS 100 // 배열의 최대 크기를 100으로 정의함.
```

```
int scores[MAX_ELEMENTS];
```

```
// scores라는 이름의 배열 선언하고 이 배열은 최대 100개의 정수 저장 가능함.
```

```
int get_max_score(int n) // n개의 점수가 배열에 저장되어 있다고 가정하고 그중에서 최댓값 찾을.
```

```
{
    int i , largest;
    largest = scores[0]; // 배열의 첫 번째 값을 초기 최댓값으로 설정함.
    for (i = 1; i < n; i++) // 배열의 두 번째 요소부터 마지막 요소 n에 도달하기 전까지 i의 값을 1
    씩 증가시켜 scores[i] 반복함.
    {
        if (scores[i] > largest) {
            largest = scores[i]; // scores[i] 가 기존의 largest보다 크다면 largest를 현재 점수
            score[i]로 업데이트함.
        }
    }
    return largest;
}
```

자료구조 : 배열이 자료를 저장하는 구조

알고리즘 : 문제를 해결하는 절차

# 위 코드에서 scores가 자료구조, 변수 largest를 첫 번째 요소로 초기화하고 나머지 요소들과 순차적

으로 비교하는 것은 알고리즘임.

알고리즘이란? 컴퓨터로 문제를 풀기 위한 단계적인 절차

문제와 컴퓨터가 주어진 상태에서 문제를 해결하는 방법을 정밀하게 장치가 이해할 수 있는 언어로 기술한 것. -> 특정 일을 수행하는 명령어들의 집합임.

# 모든 명령어들의 집합이 알고리즘이 되는 것은 아님. 명령어의 집합이 아래 조건을 만족해야 함.

- > 입력 : 0개 이상의 입력 존재해야 함.
- > 출력 : 1개 이상의 출력 존재해야 함.
- > 명백성 : 각 명령어의 의미는 모호하지 않고 명확해야 함.
- > 유한성 : 한정된 수의 단계 후에는 반드시 종료되어야 함.
- > 유효성 : 각 명령어들은 연필, 컴퓨터로 실행 가능한 연산이어야 함.

Q. 0으로 나누는 연산은 알고리즘에 해당하는가?

A.

->

알고리즘 표현하는 방법

1) 한글이나 영어 같은 자연어

- 자연어 특성상 약간의 모호성이 존재하기에 명령어로 사용되는 단어들을 명백하게 정의해야 함.

2) 흐름도

- 도형을 사용하여 알고리즘을 표현하기에 직관적이다 라는 장점이 있지만 알고리즘의 복잡도가 올라갈수록 표현하기 힘들.

3) 의사 코드(가장 많이 사용되는 방법1)

- 실제 코드는 아니지만, 사람이 읽기 쉬운 형태로 알고리즘을 서술하는 방법, 자연어보다 더 체계적임.
- 기술 형식이 파이썬과 유사하다고 생각하면 됨.

4) 프로그래밍 언어(가장 많이 사용되는 방법2)

- 가장 정확하게 알고리즘 표현 가능
- 실제 구현시 핵심적인 내용들의 이해를 방해할 수 있음.

< 배열에서 최대값을 찾는 알고리즘을 의사코드로 표현한 예 >

ArrayMax(list, N): // 배열 list와 그 크기 N을 입력받아 배열에서 가장 큰 값 변환하는 함수 설정.

largest = list[0] // 배열의 첫 번째 요소를 초기 최대값, largest로 설정.

for i = 1 to N-1: // 배열 두 번째 요소부터 마지막 요소까지 순차적으로 탐색.

if list[i] > largest: // 배열의 현재 요소가 largest보다 크다면 -> 조건식

largest = list[i] // largest를 배열의 현재 요소로 업데이트.

return largest

Q2. 알고리즘을 기술하기 위한 방법에는 자연어, 흐름도, -----, 프로그래밍 언어가 있다.

A2.

Q3. 알고리즘이 되기 위한 조건이 아닌 것?

1) 출력 2) 명백성 3) 유효성 4) 반복성

A3.

## part2. 추상 자료형

자료형이란? 말 그대로 데이터의 종류

-> 기초 자료형 + 파생 자료형 + 사용자 정의 자료형

# 기초 자료형 : 정수, 실수, 문자

파생 자료형 : 배열, 포인터, 문자열(배열로 구현되기 때문에 파생자료형임.)

사용자 정의 자료형 : 구조체, 공용체, 열거형

추상자료형(ADT)란? 데이터와 데이터를 조작하는 연산을 함께 묶어서 추상적으로 표현한 자료형.

추상자료형 역사. 추상화 -> 정보은닉 기법 -> 추상 자료형

# 추상화 : 내부 구현 방식과 상관없이 사용자가 **기능만 사용할** 수 있도록 한 것(ex. tv 리모콘)

정보은닉 기법 : 데이터 직접 조작하지 못하고, 제공된 연산만을 사용해 접근 가능. (**외부 직접 접근 제한**)

-> ADT 사용자는 구현세부사항이 아닌 인터페이스만 사용하기에 추상 자료형의 구현 방법은 언제든지 안전하게 변경 가능함. 즉, 전체 프로그램을 변경가능성이 있는 구현의 세부사항으로부터 보호함.

ADT에서 데이터나 연산이 무엇인지는 정의되지만, 데이터나 연산을 어떻게 컴퓨터 상에서 구현할 것인지는 정의되지 않음. 즉, 연산의 이름, 매개변수, 반환형은 정의하지만 연산을 구현하는 구체적인 코드는 주어지지 않는 것이 ADT임.

프로그래밍 언어에 따라 ADT는 여러 방법으로 구현됨.

객체지향언어에서는 클래스를 사용하여 ADT가 구현되며, private이나 protected 키워드 이용하여 내부 자료 접근 제한 가능함.