**Chapter01. 자료구조와 알고리즘의 이해**

* 1. **자료구조에 대한 기본적인 이해**

〮먼저 이 책은 C언어를 기반으로 자료구조를 설명하므로 C언어를 안다는 가정하에 진행된다.

〮 “프로그램이란 데이터를 표현하고, 그렇게 표현된 데이터를 처리하는 것”

여기서 데이터를 표현하는 것 -> 자료구조, 데이터를 처리하는 것 -> 알고리즘에 해당한다.

위에서 말하는 데이터 표현은 데이터의 저장을 포함하는 개념이다.

그리고 알고리즘은 자료구조에 의존적이다.

〮 자료구조는 크게 선형구조, 비선형 구조, 파일 구조, 단순구조로 분류할 수 있는데 이 책에서는 선형구조와 비선형 구조에 대해서 다룬다.

선형 구조(리스트, 스택, 큐) / 비선형 구조(트리, 그래프)

* 자료구조 학습에 있어서 기억할 것 두가지

자료구조의 모델을 그림으로 우선 이해해야 한다.

구현이 가능해야만 의미 있는 것은 아니다.

* 1. **알고리즘의 성능분석 방법**

〮알고리즘의 평가요소 – 시간 복잡도: 얼마나 빠른가?

- 공간 복잡도: 얼마나 메모리를 적게 쓰는가?

시간복잡도를 더 중요시

알고리즘 수행 속도 평가 - > 중심이 되는 특정 연산의 횟수 세어 평가 (함수T(n))

데이터 수가 많아짐에 따른 연산횟수 증가 정도 중요

중심이 되는 연산자의 특징 (Hint)

: 주변 연산자의 연산 횟수는 중심 연산자의 연산 횟수에 의존적

\*시간복잡도 <= 최악의 경우의 함수 T(n)

〮 이진탐색 알고리즘

순차 탐색보다 훨씬 좋은 성능

전제 조건: 배열에 저장된 데이터는 정렬되어 있어야한다

알고리즘 1. mid = (first + last)/2

if(target == ar[mid]) { return mid; }

else {

If(target < ar[mid]) last = mid-1;

If(target > ar[mid]) last = mid+1;

}

2. mid? Target!

3. <- -> (찾지 못했을 때)

시간 복잡도 T(n) = log2n (중심연산: ==)

〮 빅-오 표기법

T(n)에서 실제로 영향을 미치는 부분을 가리켜 빅오라 한다.

* 데이터 수의 증가횟수에 따른 연산횟수의 증가 형태(패턴)을 나타낸 표기법
* 단순하게 구하는 법: 다항식의 최고차항
* 빅-오 표기 성능 비교 : O(1) < **O(logn)** < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n)

가장 바람직한 형태