삽입정렬 (Insertion Sort)

진행 순서

1. 두번째 index의 값을 temp에 저장한다.
2. temp와 이전에 있는 원소들과 비교하며 삽입해나간다.
3. 다음 위치 index의 값을 temp에 저장하고, 과정을 반복한다.

코드

```
void insertionSort() {
    int arr[5] = {5, 4, 3, 2, 1};

    for(int i=1; i<5; ++i){
        int tmp = arr[i];
        int prevIdx = i - 1;

        while( (prevIdx >= 0) && (arr[prevIdx] > temp) ){
            arr[prevIdx+1] = arr[prevIdx]
            prevIdx--;
        }

        arr[prevIdx + 1] = tmp;
    }
}
```

시간 복잡도

> 원소가 정렬되있든, 아니든 모두 탐색함

최선	평균	최악 (역으로 정렬)
$O(N)$	$O(N^2)$	$O(N^2)$

공간 복잡도

> 한개의 배열 안에서 수행

$$O(N)$$

장점

- 구현 단순
- 원소가 정렬되어 있을 수록 효율적이다.
- 정렬하고자 하는 배열안에서 정렬 교환이 발생하므로,
  다른 메모리 공간을 필요로 하지 않는다.
- stable sort
  - 동일한 정렬 기준(같은 크기의 원소)을 가진 것은 정렬 후 위치가 같다.

단점

- 최선, 평균, 최악이 O(N^2)으로 비효율
- 정렬되어 있지 않은 원소가 많으면 SWAP이 많이 발생
- unstable sort
  - 동일한 정렬 기준(같은 크기의 원소)을 가진 것은 정렬 후 위치가 다르다.
