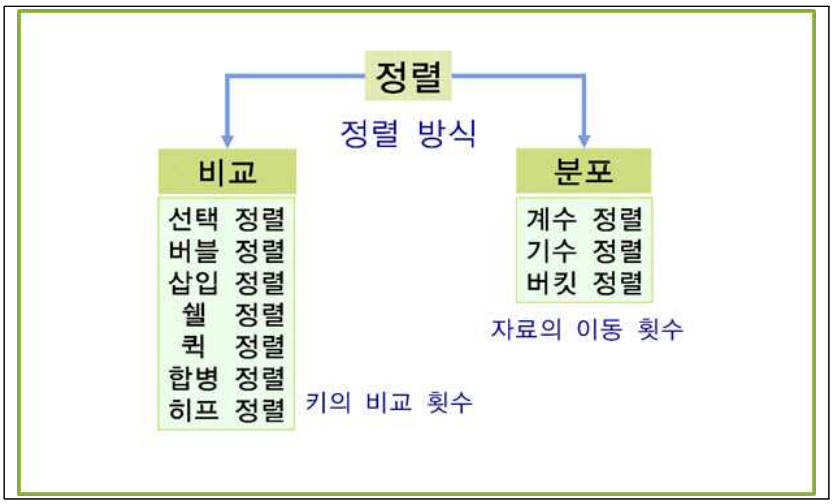
4주차 1차시 선택, 버블 정렬

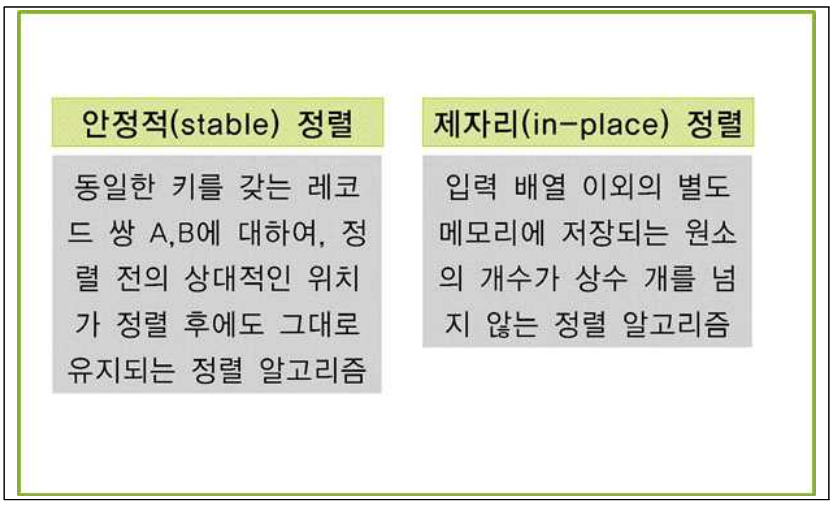
【학습목표】

1. 선택 정렬을 설명할 수 있다.
2. 버블 정렬을 설명할 수 있다.

학습내용1 : 정렬이란

정렬(sort)이란 컴퓨터 내부에 있는 수많은 자료들을 사용자의 특정 목적에 따라 순서대로 재배열하는 것을 의미한다. 데이터의 양이 많아지면서 사용자는 해당 데이터를 단시간에 검색하여 찾을 수 있어야 할 필요성이 생기게 되었는데, 검색작업의 효율성을 위해서는 데이터의 정렬이 필수적인 요소가 되었다.
단적인 예로 한 마을의 동사무소에서 인감증명을 발급받기 위해 찾아갔을 경우 그곳에는 그동에 살고 있는 사람들의 인감이 모두 기록되어 있을 것이다. 이때 인감이 오름차순으로 정렬되어 있지 않다면 동사무소 직원은 매번 찾아오는 사람들의 인감을 찾기 위해서 업무의 모든 시간을 낭비해야 할 것이다. 이렇게 보면 지금 우리가 살고 있는 주변의 거의 모든 데이터는 검색의 효율성을 위해서 정렬되어 있다고 보아도 무리가 아닐 것이다.
데이터를 정렬하기 위해서는 레코드의 특정 부분을 키(key)로 사용해야 한다. 이러한 키를 기준으로 하여 작은 값에서부터 큰 값으로 정렬하는 것을 오름차순(ascending)이라고 하고 반대로 큰 값에서 작은 값으로 정렬하는 것을 내림차순(descending)이라고 한다.





학습내용2 : 선택 정렬

1. 선택 정렬이란?

선택 정렬은 정렬 방법 중에 가장 간단한 방법 중의 하나로 제일 처음에 있는 키(key)를 가지고 나머지 키와 비교하여 제일 작은 작은 키 값을 가지는 데이터와 위치를 바꾼다. 다음 두 번째 위치한 데이터의 키를 가지고 세 번째 이후의 데이터가 가지고 있는 키와 비교하여 마찬가지로 제일 적은 키 값을 가지는 데이터와 위치를 바꾼다. 이렇게 해서 맨 마지막 바로 전 데이터와 마지막 데이터를 비교하는 순서까지 반복하면 데이터가 오름차순으로 정렬되게 된다. 이를 그림으로 보면 다음과 같다.

<원본 데이터>

5	1	6	2	4	3	7	10	9	8
52	6	57	11	22	17	62	99	85	73

<1단계>
첫 번째 있는 데이터 52와 나머지 데이터를 비교하여 제일 작은 데이터인 6과 위치를 바꾼다.

1	5	6	2	4	3	7	10	9	8
6	52	57	11	22	17	62	99	85	73

<2단계>
두 번째 있는 데이터 52와 나머지 데이터를 비교하여 가장 작은 데이터인 11과 위치를 바꾼다.

1	2	6	5	4	3	7	10	9	8
6	11	57	52	22	17	62	99	85	73

<3단계>
세 번째 있는 데이터 57과 나머지 데이터를 비교하여 가장 작은 데이터인 17과 위치를 바꾼다.

1	2	3	5	4	6	7	10	9	8
6	11	17	52	22	57	62	99	85	73

<4단계>
네 번째 있는 데이터 52와 나머지 데이터를 비교하여 가장 작은 데이터인 22와 위치를 바꾼다.

1	2	3	4	5	6	7	10	9	8
6	11	17	22	52	57	62	99	85	73

<5단계>
다섯 번째 있는 데이터 52와 나머지 데이터를 비교하였으나 52가 가장 작은 데이터이므로 건너뛴다.

1	2	3	4	5	6	7	10	9	8
6	11	17	22	52	57	62	99	85	73

<6단계>
여섯 번째 있는 데이터 57과 나머지 데이터를 비교하였으나 57이 가장 작은 데이터이므로 건너뛴다.

1	2	3	4	5	6	7	10	9	8
6	11	17	22	52	57	62	99	85	73

<7단계>
일곱 번째 있는 데이터 62와 나머지 데이터를 비교하였으나 62가 가장 작은 데이터이므로 건너뛴다.

1	2	3	4	5	6	7	10	9	8
6	11	17	22	52	57	62	99	85	73

<8단계>
여덟 번째 있는 데이터 99와 나머지 데이터를 비교하여 가장 작은 데이터인 73과 위치를 바꾼다.

1	2	3	4	5	6	7	8	9	10
6	11	17	22	52	57	62	73	85	99

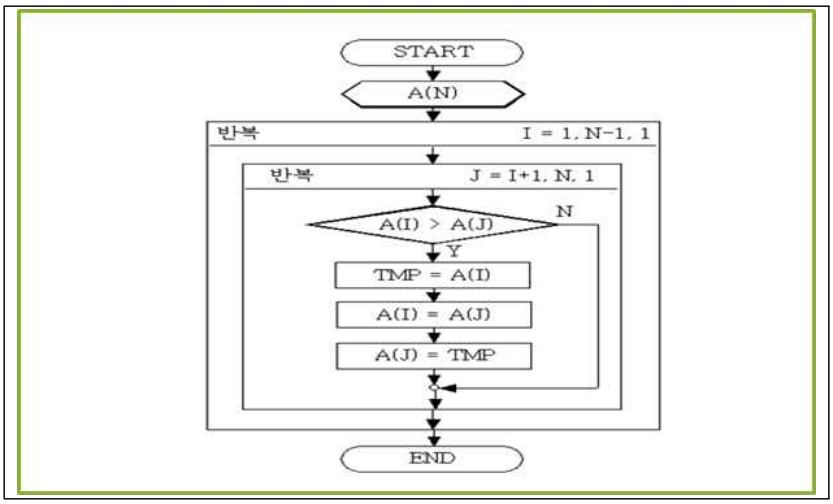
<9단계>
아홉 번째 있는 데이터 85와 나머지 데이터를 비교하였으나 85가 작으므로 건너뛴다.
결과적으로 모든 데이터가 오름차순으로 정렬되었다.

1	2	3	4	5	6	7	8	9	10
6	11	17	22	52	57	62	73	85	99

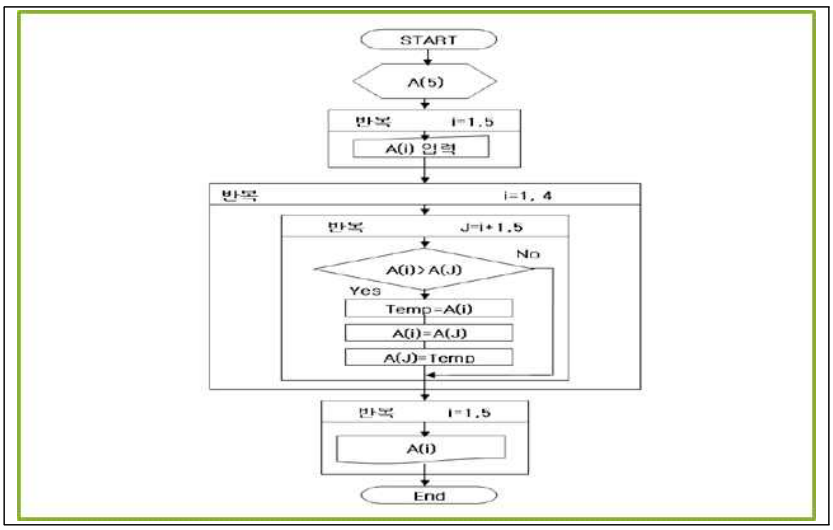
이에 대한 알고리즘을 프로그램 언어로 표현하면 다음과 같다.

```
void selectionSort(int keys[], int n){
        int i, j;
        for(i = 1, i<n, i++)
                for(j=1, j<=n; j++)
                swap(&keys[i], &keys[j]);
}
```

선택 정렬은 두가지의 기본 연산으로 이루어진다. 먼저 최소값을 찾기 위한 비교연산이 사용되며 두 번째로는 찾은 값과 비교한 대상의 위치를 교환하여 주는 교환연산이 사용된다.



* 5개의 자료를 정렬하는 예제



2. 선택정렬(selection sort)의 요약

정렬자료의 배열에서 최소의 원소를 찾아 이를 첫 위치의 원소와 교환하고 다음 제2의 최소 원소를 찾아 둘째 위치의 원소와 교환한다. 이러한 과정을 모든 원소가 정렬될 때까지 반복되는 방법이 선택정렬이다.
n개의 데이터 중 i개가 정렬되었다고 가정할 때 나머지 n-i개 중 최소인 것을 찾아 i+1번째 위치에 정렬시킨다.

① 입력 배열 외에 또 다른 배열이 필요 없는 제자리 정렬이다.
② 정렬 후에도 같은 키들의 상대적 위치가 그대로 유지되는 안정적 정렬 알고리즘이다.
③선택정렬 알고리즘의 시간복잡도는 최악/평균/최선 모두 $O(n^2)$이며 비교회수 n(n-1)/2 이 된다.

* 선택정렬의 예

초기상태 : 8, 3, 4, 9, 7
1 PASS : 3, 8, 4, 9, 7
2 PASS : 3, 4, 8, 9, 7
3 PASS : 3, 4, 7, 9, 8
4 PASS : 3, 4, 7, 8, 9

학습내용3 : 버블 정렬

1. 버블 정렬이란?

버블정렬은 주어진 데이터에서 두 개의 인접한 데이터를 비교하여 오름차순이면 작은 데이터를 왼쪽으로 내림차순이면 큰 데이터를 오른쪽으로 위치시키면서 마지막 데이터까지 반복하는 것을 순차적으로 반복하여 정렬하는 방법을 말한다.
버블정렬을 한 번 실행할 때마다 오름차순의 경우 최대로 큰 값이 배열의 가장 마지막에 위치하게 된다. 따라서 열 개의 데이터가 있다면 최초 실행 시에 가장 큰 값이 10번째에 위치하게 되고 두 번째 실행 시에는 9번째 위치하게 되며 세 번째 실행 시에는 8번째 위치하게 된다. 이러한 반복을 계속 하면 결과적으로 가장 작은 데이터가 가장 왼쪽에 위치하면서 정렬이 끝나게 된다.
이러한 과정을 그림으로 나타내면 다음과 같다.

<원본데이터>

2	1	4	5	3	10	11	8	6	7	9
10	0	30	40	20	90	100	70	50	60	80

<1단계>
첫 번째 데이터와 두 번째 데이터를 비교하여 10이 0 보다 크므로 위치를 바꾼다.

1	2	4	5	3	10	11	8	6	7	9
0	10	30	40	20	90	100	70	50	60	80

<2단계>
두 번째 데이터와 세 번째 데이터를 비교하여 10이 30보다 크지 않으므로 바꾸지 않는다.

1	2	4	5	3	10	11	8	6	7	9
0	10	30	40	20	90	100	70	50	60	80

<3단계>
세 번째 데이터와 네 번째 데이터를 비교하여 30이 40보다 크지 않으므로 바꾸지 않는다.

1	2	4	5	3	10	11	8	6	7	9
0	10	30	40	20	90	100	70	50	60	80

<4단계>
네 번째 데이터와 다섯 번째 데이터를 비교하여 40이 20보다 크므로 위치를 바꾼다.

1	2	4	3	5	10	11	8	6	7	9
0	10	30	20	40	90	100	70	50	60	80

<5단계>
다섯 번째 데이터와 여섯 번째 데이터를 비교하여 40이 90보다 크지 않으므로 바꾸지 않는다.

1	2	4	3	5	10	11	8	6	7	9
0	10	30	20	40	90	100	70	50	60	80

<6단계>
여섯 번째 데이터와 일곱 번째 데이터를 비교하여 90이 100보다 크지 않으므로 바꾸지 않는다.

1	2	4	3	5	10	11	8	6	7	9
0	10	30	20	40	90	100	70	50	60	80

<7단계>
일곱 번째 데이터와 여덟 번째 데이터를 비교하여 100이 70보다 크므로 위치를 바꾼다.

1	2	4	3	5	10	8	11	6	7	9
0	10	30	20	40	90	70	100	50	60	80

<8단계>
여덟 번째 데이터와 아홉 번째 데이터를 비교하여 100이 50보다 크므로 위치를 바꾼다.

1	2	4	3	5	10	8	6	11	7	9
0	10	30	20	40	90	70	50	100	60	80

<9단계>
아홉 번째 데이터와 열 번째 데이터를 비교하여 100이 60보다 크므로 위치를 바꾼다.

1	2	4	3	5	10	8	6	7	11	9
0	10	30	20	40	90	70	50	60	100	80

<10단계>
열 번째 데이터와 열 한번째 데이터를 비교하여 100이 80보다 크므로 위치를 바꾼다.

1	2	4	3	5	10	8	6	7	9	11
0	10	30	20	40	90	70	50	60	80	100

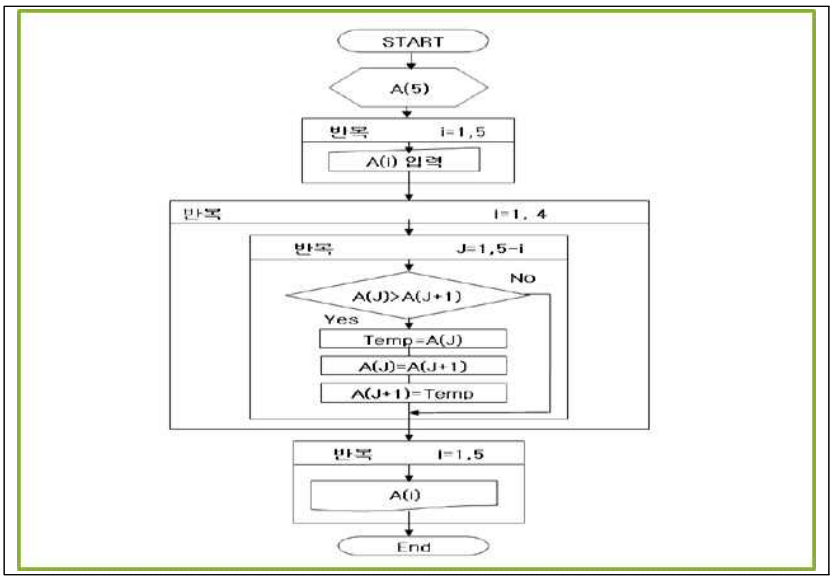
가장 큰 값이 맨 왼쪽에 위치하게 되면 가장 큰 값을 이동하는 단계가 끝난다. 이러한 방식으로 다시 처음부터 위치교환을 시작하여 9단계까지 반복하면 두 번째로 큰 값이 왼쪽에서 두 번째에 위치하며 두 번째 단계가 끝난다. 계속 반복하면 다음과 같은 결과를 얻을 수 있다.

1	2	3	4	5	6	7	8	9	10	11
0	10	20	30	40	50	60	70	80	90	100

버블 정렬의 성능은 첫 번째 단계에서는 (n-1) 번의 비교가 수행되고, 두 번째 단계에서는 (n-2) 번의 비교가 수행된다. i 번째 단계에서는 (n-i) 번을 비교한다. 따라서 평균 비교 횟수는 $O(n^2)$가 된다.
버블 정렬은 가장 단순하여 정렬 알고리즘을 처음 시작하는 사람들이 이해하기 쉬운 알고리즘으로 간단하지만 수행속도가 느리므로 잘 사용하지 않는다. 또한 안전성은 유지되나 성능이 아주 미흡하고, 최소값부터 찾아나가는 선택 정렬과 반대로 최대값을 찾는 작업을 수행하며, 대충 정렬하는 것을 동시에 반복 수행하므로 수행시간이 길어지는 단점이 있다.

이에 대한 알고리즘을 표현해보면 다음과 같다.

```
void bubbleSort(int keys[], int n){
        int i, j;
        for(i=1, i<n; i++)
                for(j=1; j<=i; j++)
                        if(keys[j] < keys[j+1])
                                swap(&keys[j], &keys[j+1]);
};
```



2. 버블정렬(bubble sort)의 요약

버블정렬은 인접한 두 키를 비교하여 정렬해 나가는 방법이다.
① 주어진 키가 모두 내림차순으로 이미 정렬되어 있는 경우가 최악의 경우이다. 이때 비교횟수는 n(n-1)/2이다.
② 선택정렬과 비교해 볼 때, 자료의 이동이 평균적으로 더 많다.
③ 두 개의 인접한 키가 동일한 값을 가지는 경우에는 키 사이의 자리바꿈이 발생하지 않고 상수 개의 변수만이 여분으로 필요하므로 안정적인 제자리 정렬이다.

* 버블정렬의 실행 예

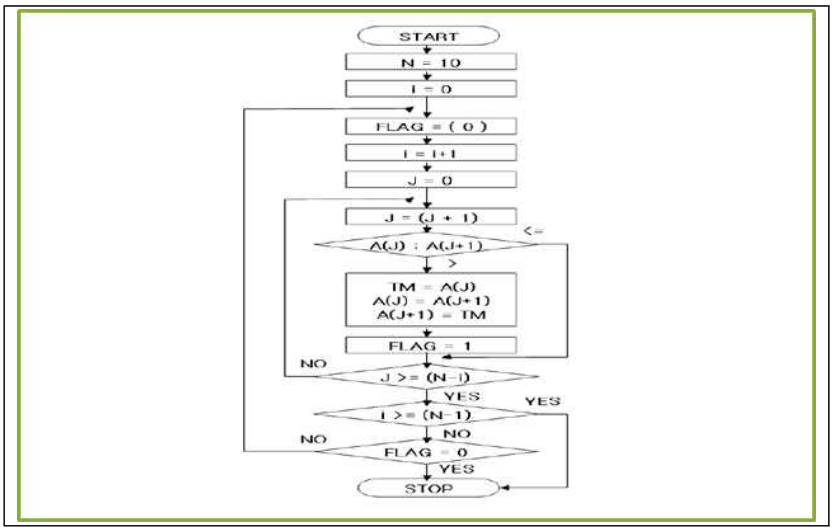
0	35	25	45	15	10	15	35	20
1	25	35	15	10	15	35	20	45
2	25	15	10	15	35	20	35	45
3	15	10	15	25	20	35	35	45
4	10	15	15	20	25	35	35	45
5	10	15	15	20	25	35	35	40

* 버블정렬의 다른 예시

1) 문제 : 제시된 <그림>은 배열 A(10)에 기억된 10개의 수치 데이터에 대하여 버블정렬(Bubble Sort)을 이용하여 오름차순으로 정렬하는 순서도이다.

2) 처리조건

- N: 정렬하고자 하는 수치 데이터의 갯수
- i : 정렬의 회전수를 계산하기 위한 변수
- J : 배열의 첨자 등을 위한 변수
- TM: 주 변수간의 값을 서로 바꾸기 위한 변수
- FLAG : 임의의 회전 작업시 데이터의 교환이 발생하지 않을 경우 비교가 반복되는 것을 방지하기 위한 변수



【학습정리】

1. 정렬이란

- 데이터를 정렬하기 위해서는 레코드의 특정 부분을 키(key)로 사용해야 한다. 이러한 키를
- 기준으로 하여 작은 값에서부터 큰 값으로 정렬하는 것을 오름차순(ascending)이라고 하고
- 반대로 큰 값에서 작은 값으로 정렬하는 것을 내림차순(descending)이라고 한다.

2. 선택정렬(selection sort)

- 정렬자료의 배열에서 최소의 원소를 찾아 이를 첫 위치의 원소와 교환하고 다음 제2의 최소 원소를 찾아 둘째 위치의 원소와 교환한다. 이러한 과정을 모든 원소가 정렬될 때까지 반복되는 방법이 선택정렬이다.
- n개의 데이터 중 i개가 정렬되었다고 가정할 때 나머지 n-i개 중 최소인 것을 찾아 i+1번째 위치에 정렬시킨다.

① 입력 배열 외에 또 다른 배열이 필요 없는 제자리 정렬이다.
② 정렬 후에도 같은 키들의 상대적 위치가 그대로 유지되는 안정적 정렬 알고리즘이다.
③선택정렬 알고리즘의 시간복잡도는 최악/평균/최선 모두 $O(n^2)$이며 비교회수 n(n-1)/2 이 된다.

* 선택정열의 예
초기상태 : 8, 3, 4, 9, 7
1 PASS : 3, 8, 4, 9, 7
2 PASS : 3, 4, 8, 9, 7
3 PASS : 3, 4, 7, 9, 8
4 PASS : 3, 4, 7, 8, 9

3. 버블정렬(bubble sort)의 요약

- 버블정렬은 인접한 두 키를 비교하여 정렬해 나가는 방법이다.

① 주어진 키가 모두 내림차순으로 이미 정렬되어 있는 경우가 최악의 경우이다. 이때 비교횟수는 n(n-1)/2이다.
② 선택정렬과 비교해 볼 때, 자료의 이동이 평균적으로 더 많다.
③ 두 개의 인접한 키가 동일한 값을 가지는 경우에는 키 사이의 자리바꿈이 발생하지 않고 상수 개의 변수만이 여분으로 필요하므로 안정적인 제자리 정렬이다.