# 09 프로그래밍 프로젝트

* 지금까지 배운 정렬 방법 들을 서로 비교하여 보자. 대상이 되는 정렬 알고리즘은 다음과 같다.

(1) 삽입 정렬
(2) 선택 정렬
(3) 버블 정렬
(4) 퀵 정렬
(5) 히프 정렬
(6) 합병 정렬

**01** 난수 생성기(random number generator)를 이용하여 데스트 데이터를 생성한다. 난수의 범위는 0에서 $n$까지로 한다.

```
// 난수로 배열을 초기화한다.
fill_test_data(int list, int n)
{
    int i;
    for(i=0; i<n; i++){
        list[i] = rand() % n;
    }
}
```

**02** 평균적인 경우에 대하여 수행을 측정하여 보자. 사실 평균적인 데이터는 아주 생성하기가 어렵다. 따라서 그냥 1번의 난수 발생기를 이용하여 데이터를 만든 다음, 정렬시켜서 측정하도록 한다. 다양한 $n$값에 대하여 실험을 수행하고 역시 그래프로 정리하라. 다음과 같은 프로그램 구조를 사용하라. 다양한 배열의 크기 $n$값에 대하여 실험하라. $n$이 적은 경우에는 대개 수행시간이 0이 된다. 따라서 같은 데이터를 여러 번 반복하여 정렬시킨 다음에 수행 시간을 측정하도록 하자. 다음과 같은 프로그램 구조를 사용하자.

```
#include <time.h>
#define MAX_REPETITION 10000

main()
{
    int r;
    clock_t start, finish;
    double  duration;
    start = clock();
    for(r=0;r<MAX_REPETITION;r++){
        fill_test_data(list, n);
        insertion_sort(list, n);
    }
    finish = clock();
    duration=(double)(finish - start) / CLOCKS_PER_SEC;
    printf("수행시간은 %f초입니다.\n", duration);
}
```

**03** 각 정렬 방법의 최악의 경우의 수행 시간을 측정하여 보자. 먼저 각 정렬 알고리즘에 대한 최악의 데이터가 필요하다. 삽입 정렬의 경우에는 역으로 정렬된 데이터가 최악이다. 즉 $n$개의 데이터라면 $n, n-1, n-2, \cdots, 1$의 입력 데이터를 사용하면 된다. 히프와 합병, 선택 정렬의 경우는 그냥 난수 데이터를 사용한다. 버블 정렬의 경우에는 역순으로 정렬된 데이터를 사용하라. 퀵정열은 정렬된 데이터가 최악이다. 테스트 데이터가 만들어지면 각 알고리즘을 구현하여 실험을 수행한다. 실험 결과값을 그래프로 표시하고 각 $n$값에 대하여 가장 빠른 정렬 알고리즘을 결정하라. 삽입 정렬과 선택, 버블 정렬은 예측했던 대로 $O(n^2)$의 성능을 보여주는지를 살펴보라.

**04** 정렬알고리즘을 비교하는 또 하나의 방법은 여러 가지 입력 데이터에 대하여 정렬에 필요한 비교 연산의 횟수와 교환 연산(또는 이동 연산)의 횟수를 계산하는 것이다. 각 정렬 알고리즘에 대하여 최악의 경우에 비교와 교환의 횟수를 계산하여 보라. 비교와 교환 연산의 횟수를 측정하기 위하여 각 함수의 몸체에 comp_count(비교)와 swap_count(교환) 변수를 추가한다. 다음에 선택 정렬의 경우의 예를 보였다. 다양한 $n$의 값에 대하여 실험을 수행하라.

```
int comp_count=0;
int swap_count=0;

void selection_sort()
{
    int i, j, least, temp;
    for(i=0; i<n-1; i++) {
       least = i;
        for(j=i+1; j<n; j++){
            if(list[j]<list[least]) least = j;
                comp_count++;
        }
        SWAP(list[i], list[least], temp);
        swap_count++;
    }
}
```