08 프로그래밍 프로젝트

01 먼저 정렬되지 않은 1차원 배열 array를 이용하여 우선 순위 큐를 구현하여 보자. 정수가 입력되는 것으로 가정하고 입력될때는 배열의 맨 마지막에 저장한다. 삭제되는 경우에는 배열에서 가장 큰 값을 순차탐색으로 찾아서 반환한다. 본문에서 언급한 바와 같이 이러한 방식의 우선 순위 큐의 시간 복잡도는 삽입 연산이 이고 삭제 연산은 이 된다. 우선 순위 큐 연산들 중에서 삽입과 삭제연산만을 구현하고 이것을 insert1(int item) 와 delete1()라고 하자. main 함수에서는 10000개의 요소를 삽입하고 삭제하는데 소요되는 시간을 측정하여 보라.

```
#include <stdio.h>
#include <time.h>

#define MAX_ARRAY_SIZE 100000
#define TEST_SIZE 10000

int count;      // 우선 순위 큐의 요소의 개수
int array[MAX_ARRAY_SIZE]; // 1차원 배열

//
void insert1(int item)
{
_____________________ ;
}
//
int delete1()
{
_____________________ ;
_____________________ ;
_____________________ ;
}
//
int main()
{
    int i;
    time_t start, finish;

    start=clock();
    for(i=0;i<TEST_SIZE;i++){
        insert1(i);
    }
    for(i=0;i<TEST_SIZE;i++){
        delete1();
    }
```

```
    finish=clock();
    printf("%d\n", finish-start);
}
```

02 1번에서 사용했던 array 배열과 count 변수를 이용하여 히프 삽입 함수와 삭제 함수를 작성하여 보라. 본문에서는 element 타입과 구조체 타입을 정의하였지만 여기서는 1번과의 비교를 위하여 저장되는 데이터는 정수라고 가정하고 heap 배열 대신에 1번의 전역 변수 array를 사용하고 heap_size 대신에 1번에서 사용하였던 count 변수를 사용하여 본문의 삽입함수와 삭제 함수를 변경하라. 이렇게 변경하는 이유는 히프도 마찬가지로 1차원 배열만을 사용하는 방법이라는 것을 강조하기 위함이다. 이 삽입과 삭제 함수를 insert2(int i)와 delete2() 라고 부르자. 역시 main 함수에서는 10000개의 요소를 삽입하고 삭제하는데 소요되는 시간을 측정하여 보라.

```
#include <stdio.h>
#include <time.h>

#define MAX_ARRAY_SIZE 100000
#define TEST_SIZE 10000

int count;      // 우선 순위 큐의 요소의 개수
int array[MAX_ARRAY_SIZE]; // 1차원 배열

// 삽입 함수
void insert2(int item)
{
    int i;
    i = ++count;

    //  트리를 거슬러 올라가면서 부모 노드와 비교하는 과정
    while((i != 1) && (item > array[i/2])){
        array[i] = array[i/2];
        i /= 2;
    }
    array[i] = item;     // 새로운 노드를 삽입
}
// 삭제 함수
int delete2()
{
// 삽입 함수와 비슷하게 본문의 소스를 변경하여 보라.
}
//
void main()
{
    int i;
    time_t start, finish;

    start=clock();
    for(i=0;i<TEST_SIZE;i++){
```

```
        insert2(i);
    }
    for(i=0;i<TEST_SIZE;i++){
        delete2();
    }
    finish=clock();
    printf("%d\n", finish-start);
}
```

03 TEST_SIZE를 여러 가지 값으로 변경하여 두 가지의 방법의 수행시간을 비교하여 보라. 두 가지 방법은 똑같이 array 1차원 배열과 count 변수만을 사용하는 방법임을 유의하여야 한다. 다만 삽입과 삭제 알고리즘이 다를 뿐이다. 어떤 결론을 내릴 수 있는가? 이론적인 시간 복잡도와는 일치하는가?

04 3번의 히프의 크기는 고정되어 있다. 삽입 함수를 약간 변경하여 현재 히프의 크기보다 더 많은 데이터가 삽입되면 자동으로 더 큰 배열을 동적으로 생성하고 기존의 데이터를 새로 생성된 배열로 복사하도록 변경하여 보라. 초기의 array 배열도 동적으로 할당하여 만들도록 변경한다.