

8장 우선순위 큐(배열을 이용해 구현한)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENT 200
#define TRUE 1
#define FALSE 0

typedef struct {
    int key;
} element;
typedef struct {
    element heap[MAX_ELEMENT];
    int heap_size;
} HeapType;

// 초기화 함수
void init(HeapType *h)
{

}

// 삽입 함수: 현재 요소의 개수가 heap_size 인 힙 h 에 item 을 삽입한다.
void insert_max_heap(HeapType *h, element item)
{
    int i;

}

// 삭제 함수
element delete_max_heap(HeapType *h)
{

}

}
```

```
void preorder(HeapType *h, int root) // 삭제
{
```

```
}
```

```
int find(HeapType *h, int root, int key) // 삭제
{
```

```
}
```

```
void print_sorted_value(HeapType heap) // delete_max_heap 을 이용한다
{
```

```
    ...
```

```
}
```

```
void print_heap(HeapType *h)
{
```

```
    ...
```

```
void modify_priority(HeapType *h, int oldkey, int newkey)
{
```

```
    ...
```

```
}
```

```
int main(void)// 주함수
```

```
{
```

```
    element e1={10}, e2={5}, e3={30}, eA = {9}, eB = {19}, eC = {39};
    element e4;
    HeapType heap; // 힙 생성
```

```
    init(&heap);           // 초기화
    // 삽입
    insert_max_heap(&heap, e1);
    insert_max_heap(&heap, e2);
    insert_max_heap(&heap, e3);
    insert_max_heap(&heap, eA);
    insert_max_heap(&heap, eB);
    insert_max_heap(&heap, eC);
    preorder(&heap, 1);
    printf("%n");
    print_heap(&heap);
```

```
    // find 함수 테스트
    ...
```

```
    // print_sorted_value 함수 테스트
    print_sorted_value(heap);
```

```
    // 삭제
    e4 = delete_max_heap(&heap);
    printf("%n<%d>%n ", e4.key);
    print_heap(&heap);
```

```
    e4 = delete_max_heap(&heap);
    printf("%n<%d>%n ", e4.key);
    print_heap(&heap);
```

HW 5 : 우선순위 큐(HEAP)

HW5_0

교재 9장 우선순위 큐의 Quiz/연습문제 중 일부

#1 ~ #12 (#11은 제외)

HW5_1(배열로 구현된 Heap 트리)

Step 1: main 함수 변경

주어진 main 함수를 아래와 같다.

10, 5, 30, 9, 19, 39의 매 삽입마다 Heap의 모양이 어떻게 바뀌는지 그림으로 그려보라.

<삭제>가 일어나면 Heap의 모양이 어떻게 바뀌는지 그림으로 그려보라.

Step 2: preorder과 print_heap 함수 구현

- 전위 순회함수 preorder를 작성하여 main 함수에서 호출하여 자신의 추측과 일치하나 확인하라.
(7장의 p271에 주어진 preorder함수는 링크로 구현된 이진트리에 대한 함수이므로 배열 구현을 위해서는 조금 변형해야한다.
힌트: void preorder(HeapType *h, int root)로 구현한다. 자식트리 는 root*2, root*2+1로 표현함에 착안하라.)
Step1처럼 원소가 삽입되었다면 출력 결과는 다음과 같다.
39 15 5 9 30 10

- print_heap(HeapType *h) 함수는 Heap h에 저장된 값들을 레벨 순회 순서로 출력하되, 레벨 단위로 줄바꿈을 하여 출력하도록 한다. 예를 들면 교재 p.____ [그림 ____]의 Heap은 다음과 같이 출력되어야 한다.
39
19 30
5 9 10

Step 3: find 함수 구현

- Heap 트리 내에 주어진 key값을 포함하는 element를 찾아 그 위치(index)를 반환하는 find 함수를 작성하여 main 함수에서 테스트해보라. Key 값이 없으면 0을 반환한다.
순환적으로 작성하라.
함수의 원형은 int find(HeapType *h, int root, int key);
- (try)최대 Heap을 레벨 순회 순서로 탐색하면서 주어진 key 값을 포함하는 element를 찾아 그 위치(index)를 반환하는 find2 함수를 작성하고 테스트해 보라. 탐색 도중에 더 이상 탐색을 계속할 필요가 없을 때 (즉, Heap에 key 값이 존재하지 않음이 확실할 때) 탐색을 중단하도록 프로그램을 작성해야 한다. Heap에서 Key 값은 중복되지 않는다고 가정한다.
함수의 원형은 int find2(HeapType *h, int key)

Step4: print_sorted_value 함수 구현

Heap 에 들어있는 값들을 내림차순으로 정렬하여 출력하는 함수를 작성하여 main 함수에서 호출해보라..

힌트:

교재 <프로그램 ____>의 heap_sort 함수를 참조하라.

Step5: (try)modify_priority 함수 구현

max Heap 과 두 개의 key 값(oldkey 와 newkey)이 주어졌을 때 oldkey 를 갖고 있는 element 를 Heap 에서 찾아 key 값을 newkey 로 변경하는 함수를 작성하라. 함수가 수행된 다음에는 최대 Heap 의 조건을 만족해야 한다.
Heap 에서 Key 값은 중복되지 않는다고 가정한다.
힌트: Heap 에서의 삽입과 삭제 알고리즘을 참조하라.

```
int main(void)
{
    element e1={10}, e2={5}, e3={30}, eA = {9}, eB = {19}, eC = {39};
    element e4, e5, e6;
    int index;
    int key, oldKey, newKey;
    HeapType heap; // 힙 생성

    init(&heap);        // 초기화

    printf("Step1: 삽입된 10, 5, 30 에 추가적으로 9, 19, 39 를 <삽입> 한다");
    insert_max_heap(&heap, e1);
    insert_max_heap(&heap, e2);
    insert_max_heap(&heap, e3);
    insert_max_heap(&heap, eA);
    insert_max_heap(&heap, eB);
    insert_max_heap(&heap, eC);

    printf("\nStep2: preorder, print_heap 함수 테스트\n");
    preorder(&heap, 1);
    printf("\n\n");
    print_heap(&heap);

    e4 = delete_max_heap(&heap);
    printf("\n 삭제: 루트가 삭제 됨\n", e4.key);
    print_heap(&heap);

    printf("\nStep3: find 함수 테스트\n");
    printf("찾을 key 입력(-1 for exit):");
    scanf("%d", &key);
    while (key != -1) {
        if ((index = find(&heap, 1, key)) == 0)
            printf("%d 는 없음\n", key);
        else
            printf("%d 은 [%d]에 있음\n", key, index);
        printf("찾을 key 입력(-1 for exit):");
        scanf("%d", &key);
    }

    printf("\nStep4: print_sorted_value 함수 테스트\n");
    print_sorted_value(heap);

    printf("\nStep5: modify_priority 함수 테스트\n");
    printf("바꿀 key 입력(-1 for exit):");
    scanf("%d", &oldKey);
    while (oldKey != -1) {
        printf("새 key 입력:");
        scanf("%d", &newKey);
        modify_priority(&heap, oldKey, newKey);
        print_heap(&heap);

        printf("바꿀 key 입력(-1 for exit):");
        scanf("%d", &oldKey);
    }
}
```

■ HW5_2(배열로 구현된 Heap 트리, 입출력파일 처리 연습)

Max Heap 에 저장될 정보가 다음과 같은 형식으로 텍스트 파일(파일이름 input.txt)에 저장된다고 가정하자.

```
10
5
30
9
19
39
```

(1) 아래와 같은 함수를 정의하라.

교재에서 다른 Heap 의 연산이나 위에서 정의한 함수를 사용하여도 좋다.

- read_heap(HeapType *h, char *filename) - 파일에서 읽어서 max Heap 에 저장한다.
- write_heap_array(HeapType *h, char *filename) - max Heap 에 저장된 내용을 배열적 표현 그대로 파일에 출력한다.
- write_descending_order(HeapType *h, char *filename) - max Heap 에 저장된 내용을 내림차순으로 파일에 출력한다.

위의 함수들을 정의하기위해 아래의 함수들을 사용한다.

```
FILE *fopen(const char *filename, const char *mode)
int fclose(FILE *stream)
int fscanf(FILE *stream, const char *format, ...)
int fprintf(FILE *stream, const char *format, ...)
```

(2) 위의 두 함수를 테스트하기 위한 main 함수는 다음과 같다. 실행결과를 예측해보고 프로그램을 실행시켜보라.

```
int main(void)
{
    HeapType heap;
    element e1 = {20}, e2 = {40};

    init(&heap);

    read_heap(&heap, "input.txt");
    insert_max_heap(&heap, e1);
    insert_max_heap(&heap, e2);

    write_heap_array(&heap, "heapArray.txt");
    write_descending_order(&heap, "sorted.txt");
}
```

HW 5(추가) : HUFFMAN CODE

■ HW5_Huffman

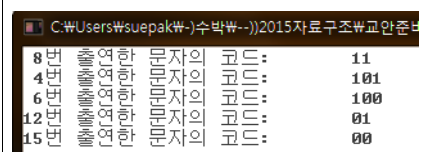
교재의 프로그램 ____를 이용하여 허프만 코드를 아래의 실행에처럼 화면에 출력하는 프로그램을 작성하라.

생성되는 허프만 코드는 수업시간에 다룬 그림 ____의 결과와 다를 수 있다(단말노드의 배치가 달라지면 허프만 코드 결과가 달라진다). 이때, 코드 해독시 문제를 일으키지 않는 코드(허프만 코드)로 제대로 생성되었음을 관찰, 확인하라. 즉 어떤 문자의 코드도 다른 문자의 코드의 첫부분이 아니다. 즉 만약 00, 001이 생성되었다면 이는 허프만 코드가 아니다. .

힌트: 교재의 프로그램은 이미 허프만 트리를 생성한다. 이것을 이용하여 허프만 코드를 출력하는 print_huffman_code 함수를 재귀적으로 정의하여 호출한다.

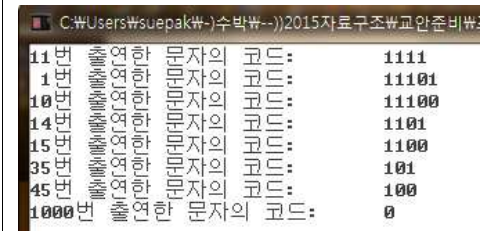
실행예:

```
int freq[MAX_ELEMENT] = {15, 12, 8, 6, 4};
```



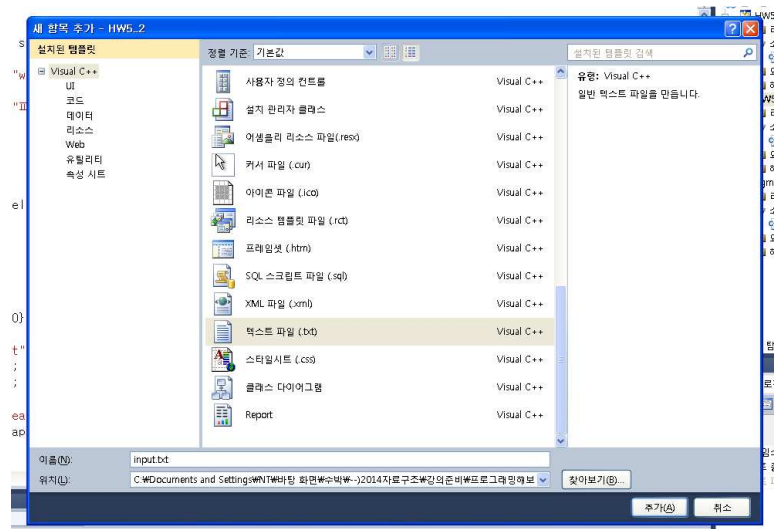
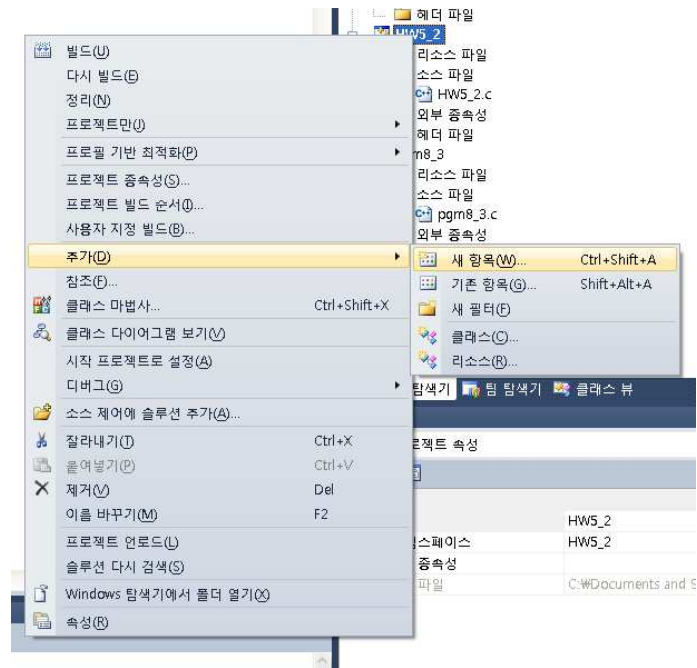
8번	출역한 문자의 코드:	11
4번	출역한 문자의 코드:	101
6번	출역한 문자의 코드:	100
12번	출역한 문자의 코드:	01
15번	출역한 문자의 코드:	00

```
int freq[MAX_ELEMENT] = {1, 10, 11, 14, 15, 35, 45, 1000};
```

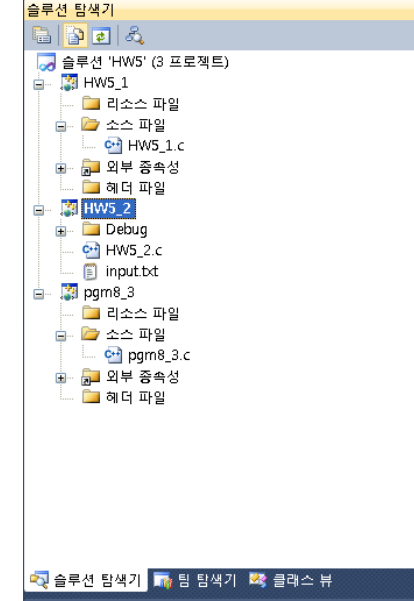


11번	출역한 문자의 코드:	1111
1번	출역한 문자의 코드:	11101
10번	출역한 문자의 코드:	11100
14번	출역한 문자의 코드:	1101
15번	출역한 문자의 코드:	1100
35번	출역한 문자의 코드:	101
45번	출역한 문자의 코드:	100
1000번	출역한 문자의 코드:	0

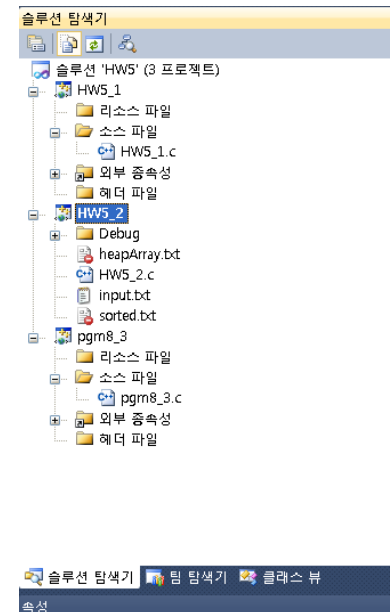
◆ 입력파일 input.txt를 준비한다.



◆ HW5_2 실행 전: input.txt를 준비



◆ 실행 후



➤ 아래는 파일 처리를 위한 뼈대코드를 보여준다. 참고하라.

```
void read_heap(HeapType *h, char *filename)
{
    // 필요한 변수

    FILE *fp = fopen(filename, "r");
    if (fp == NULL) {
        fprintf(stderr, "파일 %s을 열 수 없음!\n", filename); return;
    }

    // 구현: while (fscanf(fp, "%d\n", &n) != EOF) {...} 을 사용한다.

    fclose(fp);
}

void write_heap_array(HeapType *h, char *filename)
{
    // 필요한 변수

    FILE *fp;
    if (filename == NULL) fp = stdout;
    else {
        fp = fopen(filename, "w");
        if (fp == NULL) {
            fprintf(stderr, "파일 %s을 열 수 없음!\n", filename); return;
        }
    }

    // 구현: fprintf(fp, "%d\n", ...)을 사용한다

    fclose(fp);
}
```