1. 코드 분석 및 주석

#include "stdlib.h"

#include "stdio.h"

#include "string.h"

#define MAX\_HEAP\_SIZE 10

void Decrease\_key\_min\_heap(int A[], int i, int key) {//i에 있는 key값을 더 작게 만들도록

if (key >= A[i])

printf("error new key is not smaller thatn current key\n");

A[i] = key;

int parent;

int child;

int temp;

while (i != 1) {//i=1이면 가장 높은 노드이기 때문에

parent = i / 2;//parent노드를 찾고

if (A[i] < A[parent]) {//만약 parent보다 작은 경우에는 바꿔준다

temp = A[i];

A[i] = A[parent];

A[parent] = temp;

}

i = parent;//그리고 비교대상을 위로 올려준다.

}

}

void Increase\_min\_heap(int A[], int i, int key) {

if (key <= A[i])

printf("error new key is not smaller thatn current key\n");

A[i] = key;

int child;

int temp;

while (i <= MAX\_HEAP\_SIZE / 2) {//전체 노드의 갯수/2까지만 비교해주면 가능

child = i \* 2;

if (A[child] < A[child + 1])

child++;

if (A[i] > A[child]) {//바꾸기 연산

temp = A[i];

A[i] = A[child];

A[child] = temp;

}

i = child;//하나 내려준다.

}

}

int main() {

int h1[11] = {0,1,4,2,7,5,3,3,7,8,9 };//배열

Decrease\_key\_min\_heap(h1, 4, 3);

Increase\_min\_heap(h1, 3, 10);

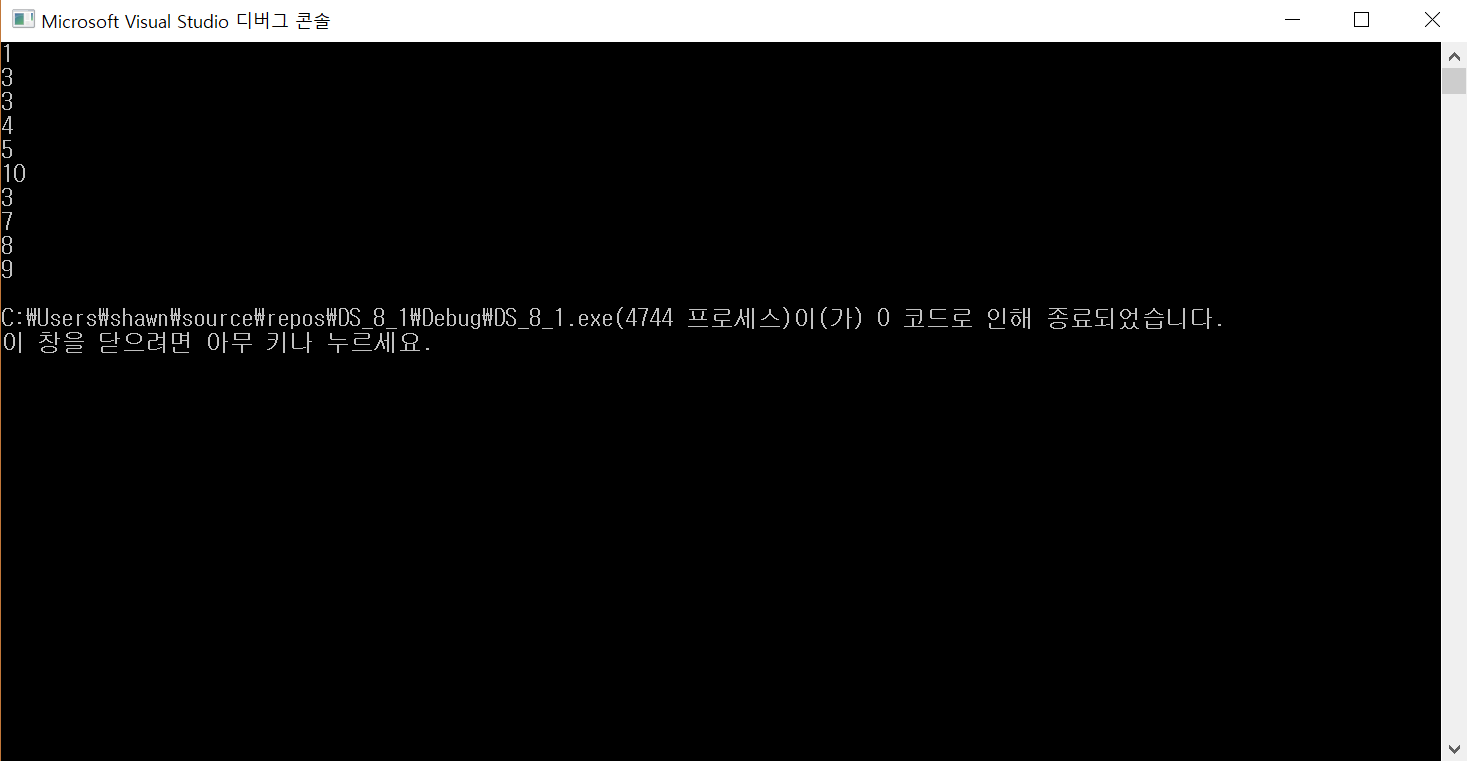
for (int i = 1; i <= MAX\_HEAP\_SIZE; i++) {

printf("%d\n", h1[i]);

}

}

1. 결과 및 콘솔 창



1. 주요 코드

Decrease, increase에서 가장 중요한 차이는 연산이 위로 가냐 앞으로 가냐 이다. Decrease는 맨 꼭대기로 가야 멈추기 때문에 조건문을 while(i!=1), i를 한단계 위로 올리도록 반복문을 짰고, increase는 아래로 가야 멈추기 때문에 반대로 짰다. 이렇게 해서 계속 단계를 올리고 내려가면서 반복문을 돌려주면 가능하다.