**로고, 폰트, 텍스트, 상징이(가) 표시된 사진

자동 생성된 설명**

**[데이터구조2 리포트]**

**이름: 황지상 학번: 20224060**

**과제#1-1\_min\_heap**

1. **코드**

#include <stdio.h> // 표준 입력/출력 라이브러리를 포함

#include <stdlib.h> // 표준 라이브러리 함수를 포함

#define MAX\_ELEMENT 200 // Heap의 최대 크기 정의

// 요소를 나타내는 구조체 정의

typedef struct {

int key; // 정수형 키 값

}element;

// 최소 힙을 나타내는 구조체 정의

typedef struct {

element heap[MAX\_ELEMENT]; // 배열로 구현된 힙

int heap\_size; // 힙의 현재 크기를 저장하는 변수

}HeapType;

// 힙을 생성하는 함수

HeapType\* create() {

return (HeapType\*)malloc(sizeof(HeapType)); // 힙 구조체를 동적으로 할당하고 포인터 반환

}

// 힙을 초기화하는 함수

void init(HeapType\* h) {

h->heap\_size = 0; // 힙 크기를 0으로 초기화

}

// 최소 힙에 요소를 삽입하는 함수

void insert\_min\_heap(HeapType\* h, element item) {

int i;

i = ++(h->heap\_size); // 힙 크기를 증가하고 인덱스 i를 설정

// 부모 노드와 비교하여 힙 속성을 만족할 때까지 원소를 올바른 위치로 이동

while ((i != 1) && (item.key < h->heap[i / 2].key)) {

// 부모 노드와 비교하여 요소를 올바른 위치에 삽입

h->heap[i] = h->heap[i / 2];

i /= 2;

}

h->heap[i] = item; // 요소를 최종 위치에 삽입

}

// 최소 힙에서 요소를 삭제하고 반환하는 함수

element delete\_min\_heap(HeapType\* h) {

int parent, child;

element item, temp;

item = h->heap[1]; // 루트 노드의 요소를 삭제하고 반환

temp = h->heap[(h->heap\_size)--]; // 힙 크기를 감소하고 마지막 요소를 임시 변수에 저장

parent = 1;

child = 2;

// 힙 속성을 유지하면서 원소를 올바른 위치로 이동

while (child <= h->heap\_size) {

// 자식 노드 중 더 작은 자식을 찾음

if ((child < h->heap\_size) && (h->heap[child].key) > h->heap[child + 1].key) {

child++;

}

// 임시 변수의 키값과 비교하여 올바른 위치로 요소를 이동

if (temp.key <= h->heap[child].key) break;

h->heap[parent] = h->heap[child];

parent = child;

child \*= 2;

}

h->heap[parent] = temp; // 요소를 최종 위치에 삽입

return item; // 삭제된 요소를 반환

}

int main(void) {

int numbers[] = { 10, 40, 30, 5, 12, 6, 15, 9, 60 }; // 정수 배열 선언 및 초기화

int num[MAX\_ELEMENT] = { 0, }; // 정수 배열 선언 및 초기화

HeapType\* heap; // 힙 구조체 포인터 선언

heap = create(); // 힙을 생성하고 포인터에 할당

init(heap); // 힙을 초기화

element k; // 요소 구조체 변수 선언

for (int i = 0; i < 9; i++) {

k.key = numbers[i]; // 요소의 키 값을 설정

insert\_min\_heap(heap, k); // 최소 힙에 요소를 삽입

}

// 최소 힙을 트리 구성 과정 출력

printf("최소 힙 트리 구성 과정:\n");

for (int i = 1; i <= heap->heap\_size; i++) {

printf("%d를 추가한 후: ", heap->heap[i].key);

num[i] = heap->heap[i].key;

for (int j = 1; j <= i; j++) {

printf("%d, ", num[j]);

}

printf("\n");

}

printf("\n==============================================================\n");

// 최소 힙 트리 배열 내용을 출력

printf("\n최소 힙 트리 배열 내용: ");

while (heap->heap\_size > 0) {

element min\_item = delete\_min\_heap(heap); // 최소 힙에서 요소를 삭제하고 반환

printf("%d ", min\_item.key); // 삭제된 요소의 키 값을 출력

}

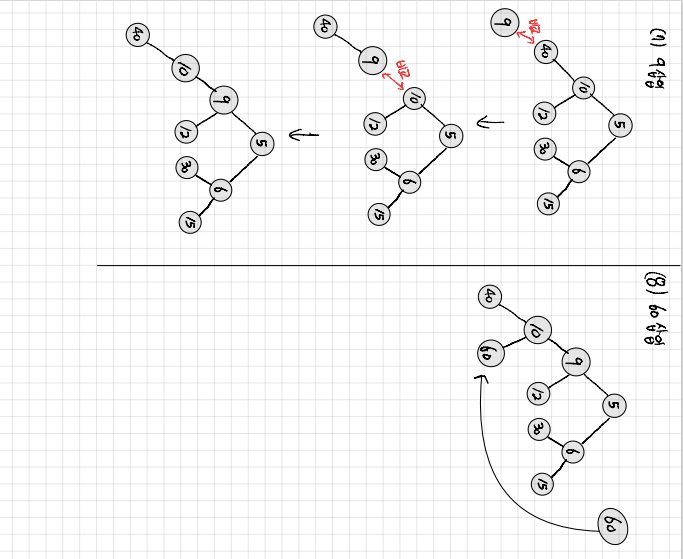
return 0;

}

1. **실행 결과**

**텍스트, 도표, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명**

****

1. **설명**

* create() 함수
  + 최소 힙을 생성하고 초기화 하기 위한 힙 구조체를 동적으로 할당하고 해당 구조체에 대한 포인터 반환
* init()
  + 힙 초기화, 힙의 크기를 0으로 설정
* insert\_min\_heap()
  + 최소 힙에 요소를 삽입하는 역할
  + 요소를 삽입 하기 때문에 힙의 크기를 1 증가
  + 요소가 추가한 맨 마지막에 있다고 생각하고 밑에서 부터 부모 노드보다 작으면 부모 노드와 위치를 바꾸면서 올바른 위치로 이동 시켜 최소 힙 특성을 만족하도록 함
* delete\_min\_heap()
  + 최소 힙에서 요소를 삭제하고 반환하는 역할
  + 루트 노드를 삭제하고, 힙의 크기를 감소시키며, 삭제한 루트 노드를 반환
  + 삭제 후 최소 힙 특성을 유지하기 위해 삭제한 노드의 양쪽 자식 노드와 비교하여 두 자식 노드 중 더 작은 요소를 삭제한 노드 위치로 이동시켜 올바른 위치에 있도록 함
* 요소 추가와 최소 힙 구성 과정 출력
  + for 루프를 사용하여 주어진 배열을 순서대로 읽어서 insert\_min\_heap() 함수를 호출하여 최소 힙에 요소를 추가
  + 동시에 각 단계에서 최소 힙의 구성 과정을 출력하고 각 단계에서 요소가 추가 될때마다 최소 힙의 내용을 출력하여 시각화 함
* 최소 힙 배열 내용 출력
  + 최소 힙 트리의 구성이 완료되면, 최소 힙의 내용을 삭제 하면서 출력함