**로고, 폰트, 텍스트, 상징이(가) 표시된 사진

자동 생성된 설명**

**[데이터구조2 리포트]**

**이름: 황지상 학번: 20224060**

**과제#1-2\_maxheap**

1. **코드**

#define \_CRT\_SECURE\_NO\_WARNINGS // 컴파일러 경고를 무시하기 위한 선언

#include <stdio.h> // 포준 입력/출력 라이브러리 포함

#include <stdlib.h> // 포준 라이브러리 포함

#define MAX\_ELEMENT 100 /// Heap의 최대 크기 정의

typedef struct {

int heap[MAX\_ELEMENT]; // 배열로 구현된 힙

int heap\_size; // 힙의 현재 크기를 저장하는 변수

} HeapType;

// 힙 초기화 함수

void init(HeapType\* h) {

h->heap\_size = 0; // 힙의 크기를 0으로 초기화

}

// 힙이 가득 찬 경우 예외 처리하는 함수

int is\_full(HeapType\* h) {

return h->heap\_size >= MAX\_ELEMENT - 1; // 힙의 크기가 최대 원소 개수보다 크거나 같으면 참 반환

}

// 힙이 비어 있는 경우 예외 처리하는 함수

int is\_empty(HeapType\* h) {

return h->heap\_size <= 0; // 힙의 크기가 0보다 작거나 같으면 참 반환

}

// 최대 힙에 원소를 삽입하는 함수

void insert\_max\_heap(HeapType\* h, int item) {

if (is\_full(h)) { // is\_full 함수를 사용하여 가득 찬 경우 처리

printf("Full\n");

return;

}

int 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 > h->heap[i / 2]) {

// 부모 노드와 비교하여 요소를 올바른 위치에 삽입

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

i /= 2;

}

h->heap[i] = item; // 요소를 최종 위치에 삽입

}

// 최대 힙에서 요소를 삭제하고 반환하는 함수

int delete\_max\_heap(HeapType\* h) {

if (is\_empty(h)) { // is\_empty 함수를 사용하여 비어 있는 경우 처리

printf("Empty\n");

return -1;

}

int parent, child;

int item, temp;

item = h->heap[1]; // 최대 원소를 저장

temp = h->heap[(h->heap\_size)--]; // 힙 크기를 감소시키고 마지막 원소를 저장

parent = 1;

child = 2;

// 힙 속성을 유지하면서 원소를 올바른 위치로 이동

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

// 현재 노드(child)와 그 자식 노드 중 더 큰 값을 찾음

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

child++;

}

// 현재 노드의 값(temp)이 자식 노드보다 크거나 같으면 루프 종료

if (temp >= h->heap[child]) {

break;

}

// 부모 노드에 자식 노드의 값을 복사하여 부모 노드를 아래로 이동

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

parent = child; // 부모 노드의 인덱스를 자식 노드의 인덱스로 갱신

child \*= 2; // 자식 노드로 이동하여 다음 레벨의 자식 노드로 이동

}

h->heap[parent] = temp; // 올바른 위치에 원소 삽입

return item; // 삭제한 최대 원소 반환

}

int main(void) {

HeapType heap; // HeapType 구조체 변수 heap을 선언

init(&heap); // heap을 초기화하는 함수 호출

char op; // 문자형 변수 op 선언

int x; // 정수형 변수 x 선언

while (1) {

scanf(" %c", &op); // 사용자로부터 문자 입력 받음

// 입력된 문자가 'I'일 경우

if (op == 'I') {

scanf("%d", &x); // 정수를 입력 받음

insert\_max\_heap(&heap, x); // 입력된 정수를 최대 힙에 삽입하는 함수 호출

}

// 입력된 문자가 'D'일 경우

else if (op == 'D') {

int result = delete\_max\_heap(&heap); // 최대 힙에서 원소 삭제하는 함수 호출

// 원소 삭제가 성공한 경우

if (result != -1) {

printf("%d\n", result); // 삭제한 최대 원소를 출력

}

}

// 입력된 문자가 'Q'일 경우

else if (op == 'Q') {

break; // 무한 루프 종료

}

}

return 0;

}

1. **실행 결과**

**텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명**

1. **설명**

* is\_full()
  + 힙이 가능 찼을 때의 예외 처리 함수, 힙이 가득 찼는지 확인
* is\_empty()
  + 힙이 비어 있을 때의 예외 처리 함수, 힙이 비어 있는지 확인
* insert\_max\_heap()
  + 최대 힙에 요소를 삽입하는 역할
  + 요소를 삽인 하기 때문에 힙의 크기를 1 증가
  + 요소가 추가한 맨 마지막에 있다고 생각하고 맨 밑에서 부터 부모 노드보다 크면 부모 노드와 위치를 바꾸면서 올바른 위치로 이동시킴
  + 힙이 가득 찬 경우 Full 출력
* delete\_max\_heap()
  + 힙이 비어 있는 경우 empty 출력
  + 최대 힙에서 요소를 삭제하고 반환하는 역할
  + 루트 노드를 삭제하고, 힙의 크기를 감소시키며, 삭제한 루트 노드를 반환
  + 삭제 후 최대 힙의 특성을 유지하기 위해 삭제한 노드의 양쪽 자식 노드를 비교하여 두 자식 노드 중 더 큰 요소를 삭제한 노드 위치로 이동시켜 올바른 위치에 있도록 함
* **사용자 입력 처리**
  + **while 무한 루프를 사용하여 사용자에게 입력 받음**
  + **입력된 문자가 I일 경우 정수를 입력 받고   
    insert\_max\_heap()를 호출하여 최대 힙에 요소를 추가**
  + **입력된 문자가 D일 경우 delet\_max\_heap()을 호출하여 최대 힙에서 요소를 삭제하고 해당 요소를 출력**
  + **입력된 문자가 Q일 경우 프로그램을 종료 합니다.**