자료구조

과제3

컴퓨터공학과

2019305059

이현수

[소스코드] – 이진탐색트리

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

typedef struct nodeRecord {
	int data;			//데이터저장변수
	struct nodeRecord* LChild; //왼쪽노드포인터
	struct nodeRecord* RChild; //오른쪽노드포인터
}node;
typedef node* Nptr;

void InOrder(Nptr T) { //중위순회
	if (T != NULL) { //노드포인터가 NULL이 아닐때까지
		InOrder(T->LChild);
		printf("%d ", T->data);
		InOrder(T->RChild);
	}
}

Nptr Insert(Nptr T, int Key) //트리에 데이터삽입
{
	if (T == NULL) { //노드포인터가 가리키는 노드가 없을때
		T = (node*)malloc(sizeof(node));  //동적할당
		T->data = Key;				//데이터key 삽입
		T->LChild = NULL; T->RChild = NULL; //왼쪽,오른쪽 노드포인터 NULL로
	}
	else if (T->data > Key) { //삽입데이터가 노드에있는 데이터보다 작을경우
		T->LChild = Insert(T->LChild, Key); //왼쪽노드로 이동
	}
	else{ //그외...삽입데이터가 노드에있는 데이터보다 클경우...
		T->RChild = Insert(T->RChild, Key);//오른쪽노드로 이동
	}
	return T; //노드포인터 반환
}

/*//정수30개입력
int main(void) {
	int input;
	int i;

	Nptr T = NULL;

	printf("이진탐색트리₩n");
	printf("정수 30개를 입력하세요 : ");
	for (i = 0;i < 30;i++) {
		scanf("%d", &input);  //정수입력
		T = Insert(T, input); //이진탐색트리에 정수삽입
```

```
	}
	printf("₩n");
	InOrder(T); //중위순회
}
*/

/*//정수40개입력
int main(void) {
	int input;
	int i;

	Nptr T = NULL;

	printf("이진탐색트리₩n");
	printf("정수 40개를 입력하세요 : ");
	for (i = 0;i < 40;i++) {
		scanf("%d", &input);   //정수입력
		T = Insert(T, input); //이진탐색트리에 정수삽입
	}
	printf("₩n");
	InOrder(T); //중위순회
}
*/

//정수50개입력
int main(void) {
	int input;
	int i;

	Nptr T = NULL;

	printf("이진탐색트리₩n");
	printf("정수 50개를 입력하세요 : ");
	for (i = 0;i < 50;i++) {
		scanf("%d", &input);   //정수입력
		T = Insert(T, input); //이진탐색트리에 정수삽입
	}
	printf("₩n");
	InOrder(T); //중위순회
}
```

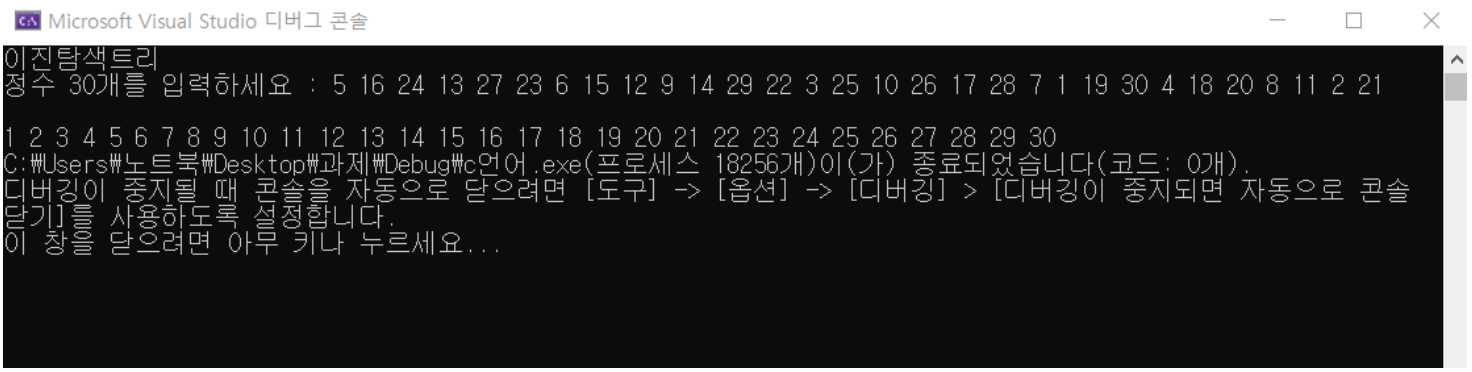
[소스코드] - Min Heap

```
#include<stdio.h>
#include<stdbool.h>
#define MAX 100

typedef struct heap {
	int heaparr[MAX];//데이터 저장할 배열
	int size;		//데이터 개수 변수
}HEAP;
typedef HEAP* Nptr;


void inint(Nptr hp) {
	hp->size = 0; //개수 0으로 초기화
}


bool Isempty(Nptr hp) {
	if (hp->size == 0)return true; //개수가 0이면 true반환
	else return false;		//아니면 false반환
}


void Add(Nptr hp, int data) {//하향식 힙
	int Current, Parent, temp;

	if (hp->size == MAX) {//힙이 꽉 찼을 경우
		printf("Heap Max₩n");
	}
	hp->heaparr[(hp->size)] = data; //배열의 마지막에 data삽입
	Current = hp->size;		//그 위치의 인덱스
	Parent = (Current - 1) / 2;	//부모 노드의 인덱스

	//루트노드가 아니고 부모노드보다 작을 때 반복
	while ((Current != 0) && (hp->heaparr[Current] < hp->heaparr[Parent])) {
		temp = hp->heaparr[Parent];			//부모와 자식스왑
		hp->heaparr[Parent] = hp->heaparr[Current];	//부모와 자식스왑
		hp->heaparr[Current] = temp;			//부모와 자식스왑
		Current = Parent;	//스왑 된 위치를  새로운 인덱스로
		Parent = (Current - 1) / 2; //스왑 된 위치에서의 부모 인덱스
	}
	hp->size++;// 수 증가
}
```

```
void DownHeap(Nptr hp, int Current) {
	int child, Rchild, temp;
	if ((2*Current+1)>= hp->size) { //리프노드까지 도착
		//아무것도 안함.
	}
	else {
		child = 2 * Current + 1; //왼쪽자식이 오른쪽 자식보다 작다고 간주
		if (hp->size>=(2*Current+2)) {//오른쪽 자식이 존재하면
			Rchild = child + 1;		//그 인덱스는 왼쪽자식보다 하나 많음
			if (hp->heaparr[Rchild] < hp->heaparr[child]) {//왼쪽보다 오른쪽자식이 작다면
				child = Rchild;  //오른쪽자식 인덱스를 child변수에 저장
			}
		}
		if (hp->heaparr[Current]> hp->heaparr[child]) {//현재 레코드가 자식보다 크면
			temp = hp->heaparr[Current];			//부모와 자식스왑
			hp->heaparr[Current] = hp->heaparr[child];  //부모와 자식스왑
			hp->heaparr[child] = temp;			//부모와 자식스왑
			DownHeap(hp, child); //내려간 위치에서 다시 재귀호출
		}
	}
}

void Remove(Nptr hp) {
	printf("%d ", hp->heaparr[0]);  //삭제되는 노드 출력
	hp->heaparr[0] = hp->heaparr[(hp->size)-1]; //마지막노드 루트노드로 이동
	(hp->size)--;  //개수 감소
	DownHeap(hp, 0); //0번 인덱스로부터 굴러 떨어지기 호출
}

/*//정수30개입력
int main(void) {
	int input;
	int i;

	HEAP H;
	inint(&H); //힙 초기화

	printf("Min Heap₩n");
	printf("정수 30개를 입력하세요 : ");
	for (i = 0;i < 30;i++) {
		scanf("%d", &input);
		Add(&H, input);
	}

	printf("₩n");

	while (!Isempty(&H)) {
		Remove(&H);
```

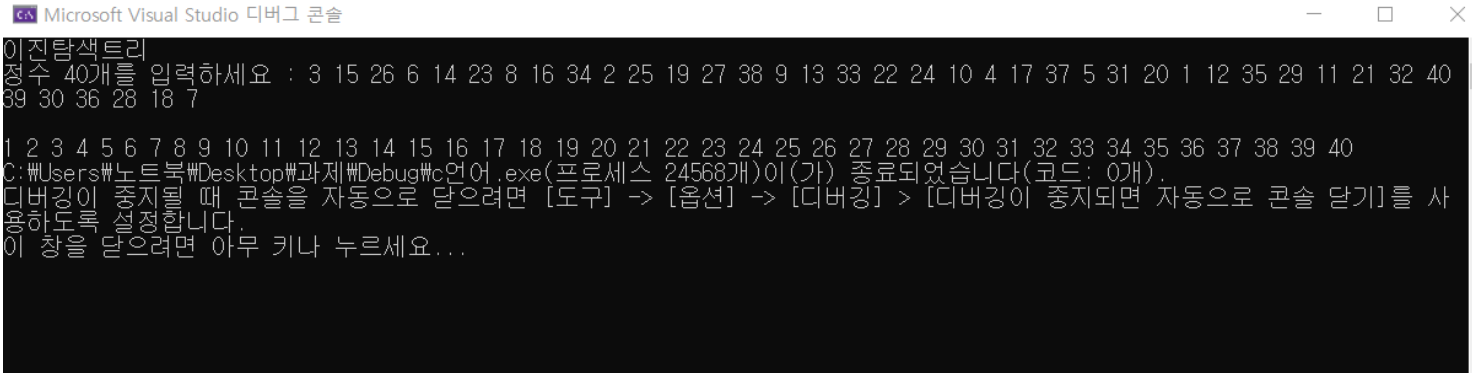
```
	}
}
*/

/*//정수40개입력
int main(void) {
	int input;
	int i;

	HEAP H;
	inint(&H); //힙 초기화

	printf("Min Heap₩n");
	printf("정수 40개를 입력하세요 : ");
	for (i = 0;i < 40;i++) {
		scanf("%d", &input);
		Add(&H, input);
	}

	printf("₩n");

	while (!Isempty(&H)) {
		Remove(&H);
	}
}
*/

//정수50개입력
int main(void) {
	int input;
	int i;

	HEAP H;
	inint(&H); //힙 초기화

	printf("Min Heap₩n");
	printf("정수 50개를 입력하세요 : ");
	for (i = 0;i < 50;i++) {
		scanf("%d", &input);
		Add(&H, input);
	}

	printf("₩n");

	while (!Isempty(&H)) {
		Remove(&H);
	}
}
```

[실행] – 이진탐색트리



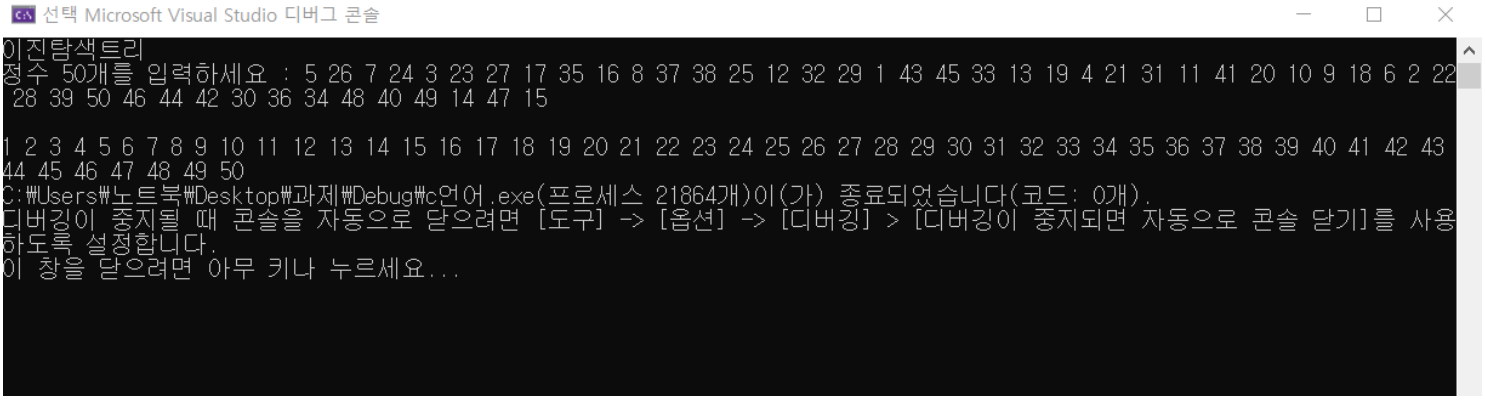
입력 : 1~30 정수를 순서없이 입력

결과 : 1~30까지 오름차순 정렬



입력 : 1~40 정수를 순서없이 입력

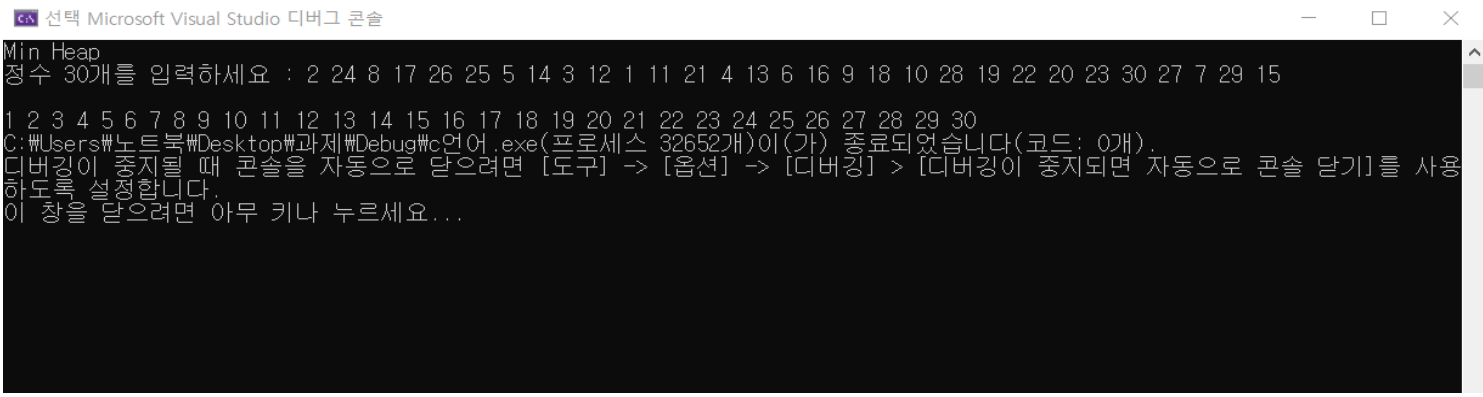
결과 : 1~40까지 오름차순 정렬



입력 : 1~50 정수를 순서없이 입력

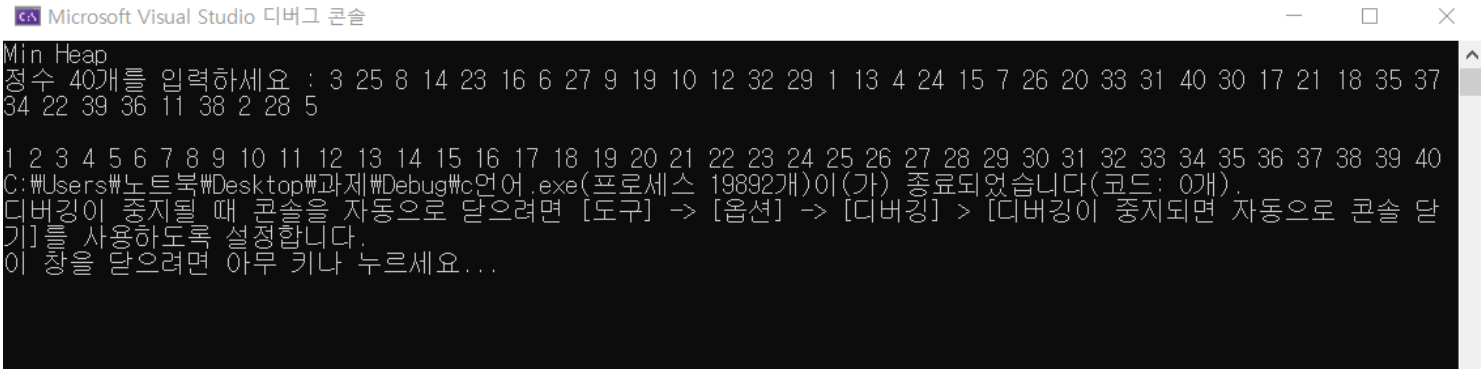
결과 : 1~50까지 오름차순 정렬

[실행] - Min Heap



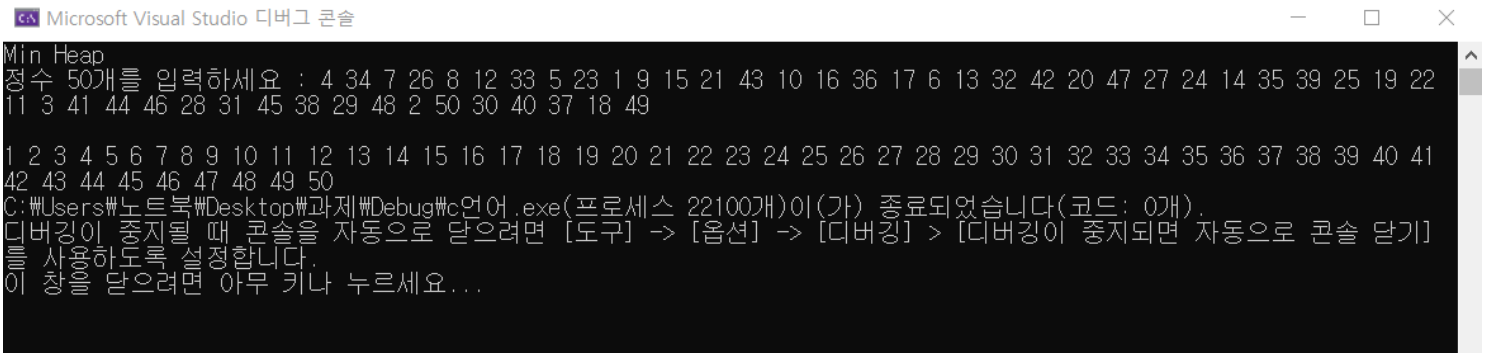
입력 : 1~30 정수를 순서없이 입력

결과 : 1~30까지 오름차순 정렬



입력 : 1~40 정수를 순서없이 입력

결과 : 1~40까지 오름차순 정렬



입력 : 1~50 정수를 순서없이 입력

결과 : 1~50까지 오름차순 정렬