



순천향대학교
SOON CHUN HYANG
UNIVERSITY

자료구조 실습 HW 7주차

자료구조2 실습

담당교수	홍민 교수님
학과	컴퓨터소프트웨어공학과
학번	20204005
이름	김필중
제출일	2021년 10월 12일

| 목 차 |

1. 배열 동물 히프 출력

- 1.1 문제 분석
- 1.2 소스 코드
- 1.3 소스 코드 분석
- 1.4 실행 결과
- 1.5 느낀점

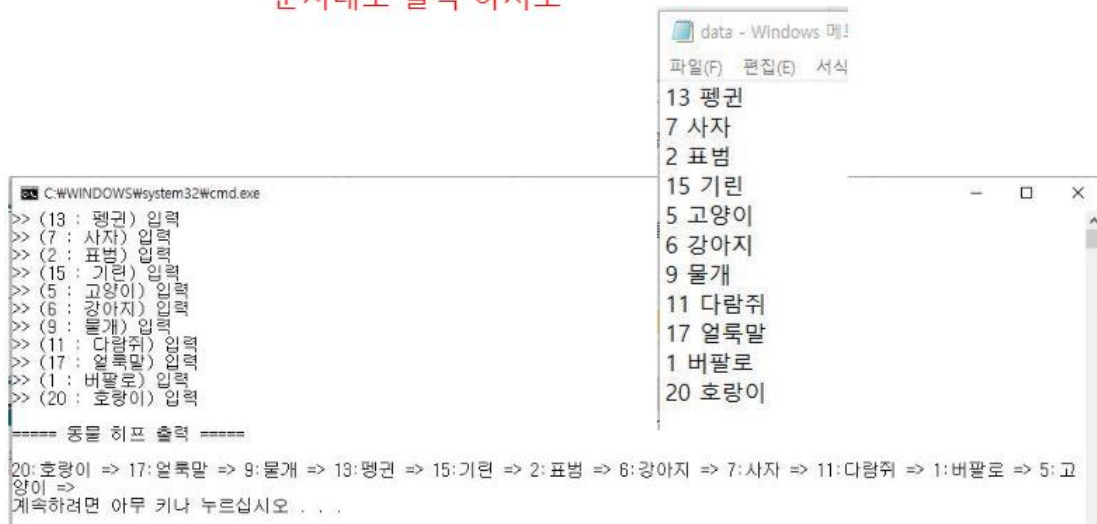
2. 배열 히프 손님 관리

- 2.1 문제 분석
- 2.2 소스 코드
- 2.3 소스 코드 분석
- 2.4 실행 결과
- 2.5 느낀점

1.1 문제 분석

■ 동물 히프

- 배열을 이용한 히프를 사용하여 data.txt에 있는 우선 순위와 동물들의 이름을 저장하여 히프에 추가하는 프로그램을 작성하시오.
 - 히프에 입력된 데이터를 히프의 루트부터 시작하여 저장되어 있는 순서대로 출력 하시오



The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe' and a text file named 'data.txt'.

The command prompt shows the following input and output:

```
>> (13 : 펭귄) 입력
>> (7 : 사자) 입력
>> (2 : 표범) 입력
>> (15 : 기린) 입력
>> (5 : 고양이) 입력
>> (6 : 강아지) 입력
>> (9 : 물개) 입력
>> (11 : 다람쥐) 입력
>> (17 : 얼룩말) 입력
>> (1 : 버팔로) 입력
>> (20 : 호랑이) 입력

===== 동물 히프 출력 =====
20: 호랑이 => 17: 얼룩말 => 9: 물개 => 13: 펭귄 => 15: 기린 => 2: 표범 => 6: 강아지 => 7: 사자 => 11: 다람쥐 => 1: 버팔로 => 5: 고양이 =>
계속하려면 아무 키나 누르십시오 . . .
```

The 'data.txt' file contains the following text:

```
13 펭귄
7 사자
2 표범
15 기린
5 고양이
6 강아지
9 물개
11 다람쥐
17 얼룩말
1 버팔로
20 호랑이
```

이번 문제는 배열을 이용해 히프를 만들어 텍스트 파일을 읽은 후 히프를 출력하는 것이다. 그러기 위해서는 히프 구조체와 데이터 구조체를 만든 후 초기화를 진행하고 삽입함수를 만들어 최대 히프를 만들어야 한다.

우선 히프를 정의해야 한다. 히프의 요소가 들어가 있는 구조체와 히프 요소의 1차원 배열을 만들어 히프를 구현한다 히프 구조체에는 저장된 요소의 개수 변수도 추가한다

히프 초기화 함수와 히프 동적 생성 함수를 만들어 준다.

히프에 새로운 요소를 넣어 줄 삽입 함수를 만들어 준다. 이 문제에서는 숫자가 큰 것이 우선이기 때문에 최대 히프 삽입 함수를 만들어 준다. 부모노드와 자식노드를 비교해 큰 것을 계속 바꿔주면서 코드를 진행하게 만들면 된다.

히프 디스플레이 함수를 만들어 준다. 각 인덱스 번호로 보여주면 되는것이기 때문에 for문을 통해 printf 를 진행하면 된다

또한 동적할당을 통해 이름의 크기를 동적할당을 해준다. 포인터를 이용해 이름의 크기만큼 동적할당 해준다.

1.2 소스 코드

```
1 //
2 // 제작일 : 21년 10월 6일 ~ 10월 10일
3 // 제작자 : 20204005 김필중
4 // 프로그램명: 배열 동물 히프 출력
5 //
6
7 // 필요한 헤더파일 선언
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11
12 // 오류 방지 구문
13 #pragma warning (disable : 4996)
14
15
16 // element 구조체 선언
17 typedef struct element
18 {
19     int num; // 숫자 크기
20     char *name; // 이름 포인터 선언
21 }element;
22
23 // 힙 트리 구조체 선언
24 typedef struct HeapType
25 {
26     element heap[100]; // 1차원 배열 만들어 힙 구현
27     int heap_size; // 힙 안에 저장된 요소의 개수
28 }HeapType;
29
30 // 힙 생성 함수
31 HeapType *create()
32 {
33     return (HeapType *)malloc(sizeof(HeapType)); // 동적 할당
34 }
35
36 // 힙 초기화 함수
37 void init(HeapType *h)
38 {
39     h->heap_size = 0; // 힙 저장 요소 0으로 초기화
40 }
41
42 // 힙 최대 힙 삽입 함수
43 void insert_max_heap(HeapType *h, element data)
44 {
45     int i; // i 변수 선언
46
47     i = ++(h->heap_size); // 저장 요소 1개 추가
48
49     while ((i != 1) && (data.num > h->heap[i / 2].num)) // 만약 i가 루트노드가 아니고 들어온 값이 현재 요소 / 2의 값 보다 클 경우 계속 반복
50     {
51         h->heap[i] = h->heap[i / 2]; // i번째 노드와 i/2 노드를 교환
52         i /= 2; // i를 2로 나뉘준다(레벨 1개 올라감)
53     }
54     h->heap[i] = data; // 들어온 값을 저장
55 }
56
57 // 표시 함수
58 void display(HeapType *h, int size)
59 {
60     int i; // i 변수 선언
61     printf("===== 동물 히프 출력 =====\n");
62     for (i = 1; i <= size; i++) // size 만큼 반복한다
63     {
64         if (i == size) // 만약 size(총 개수)와 i랑 같을 경우
65         {
66             printf("%d: %s", h->heap[i].num, h->heap[i].name); // 화살표 없이 출력한다
67             return; // 종료한다
68         }
69         printf("%d: %s => ", h->heap[i].num, h->heap[i].name); // i번째 요소를 출력한다
70     }
71 }
72 }
```

```

73 // 메인 함수
74 int main(void)
75 {
76     FILE *fp; // 파일 포인터 fp 선언
77     HeapType *heap; // heap 트리 선언
78     char temp[100]; // temp 임시 문자열 저장소 설정
79     int cnt = 0; // 파일 개수 확인 변수 선언
80     element *t; // 임시 저장 구조체 선언
81     element tmp; // 히프에 넣을 구조체 선언
82     int i = 0; // 구조체 인덱스 변수
83     int size; // 문자열 크기 변수
84
85     heap = create(); // 히프 트리를 생성한다
86     init(heap); // 히프 트리를 초기화한다
87
88     fp = fopen("data.txt", "r"); // 파일을 오픈한다
89
90     // 만약 오픈에 실패하면 종료한다
91     if (fp == NULL)
92     {
93         printf("파일 오픈 실패\n");
94         return 0;
95     }
96
97     // 파일 끝까지 반복한다
98     while (!feof(fp))
99     {
100         fscanf(fp, "%d %s\n", &tmp.num, temp); // 파일을 읽어 온다
101         cnt++; // 총 개수를 증가시킨다
102     }
103
104     t = (element *)malloc(sizeof(element) * cnt); // 총 추가되는 개수만큼 동적할당을 한다
105
106     rewind(fp); // 파일 포인터를 처음으로 돌린다
107
108
109
110     while (!feof(fp)) // 파일 끝까지 반복한다
111     {
112         fscanf(fp, "%d %s\n", &t[i].num, temp); // 파일에 있는 숫자와 문자열을 읽어온다
113         size = strlen(temp); // 이름의 크기를 확인한다
114         t[i].name = (char *)malloc(sizeof(char) * (size + 1)); // 이름 크기 + 1 만큼 동적할당을 한다
115         strcpy(t[i].name, temp); // 문자열을 저장한다
116         insert_max_heap(heap, t[i]); // 저장한 구조체를 insert_max_heap 함수를 통해 대입한다
117         printf(">> (%d : %s) 입력\n", t[i].num, t[i].name);
118         i++;
119     }
120
121     printf("\n");
122
123
124
125     printf("\n");
126
127     display(heap, cnt); // 히프 트리를 출력한다
128
129     fclose(fp); // 파일을 종료한다
130     for (int j = 0; j < i; j++)
131         free(t[j].name); // 2차원 배열 먼저 해제한다
132     free(heap); // 히프 트리를 해제한다
133     return 0; // 종료한다

```

1.3 소스 코드 분석

```
// 필요한 헤더파일 선언
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 오류 방지 구문
#pragma warning(disable : 4996)
```

1. 필요한 헤더파일을 선언한다
2. 오류 방지 구문을 선언한다.

```
// element 구조체 선언
typedef struct element
{
    int num; // 숫자 크기
    char *name; // 이름 포인터 선언
}element;
```

3. element 구조체를 선언해 각 요소들의 구조체를 만든다.
4. 이름을 포인터로 선언해 크기만큼 동적할당을 하기 위함이다.

```
// 힙 트리 구조체 선언
typedef struct HeapType
{
    element heap[100]; // 1차원 배열 만들어 힙 구현
    int heap_size; // 힙 안에 저장된 요소의 개수
}HeapType;
```

5. 힙 트리 구조체를 선언한다
6. 1차원 배열을 만들어 힙을 구현한다
7. 힙 안에 저장된 요소의 개수 변수를 정한다

```

// 힙 생성 함수
HeapType *create()
{
    return (HeapType *)malloc(sizeof(HeapType)); // 동적 할당
}

// 힙프 초기화 함수
void init(HeapType *h)
{
    h->heap_size = 0; // 힙프 저장 요소 0으로 초기화
}

```

8. 힙프 생성 함수를 통해 동적할당을 한다

9. 힙프 초기화 함수를 통해 힙프 저장 요소를 0으로 초기화 한다.

```

// 힙프 최대 힙프 삽입 함수
void insert_max_heap(HeapType *h, element data)
{
    int i; // i 변수 선언
    i = ++(h->heap_size); // 저장 요소 1개 추가
    while ((i != 1) && (data.num > h->heap[i / 2].num)) // 만약 i가 루트노드가 아니고 들어온 값이 현재 요소 / 2 의 값 보다 클 경우 계속 반복
    {
        h->heap[i] = h->heap[i / 2]; // i번째 노드와 i/2 노드를 교환
        i /= 2; // i를 2로 나눴준다(레벨 1개 올라감)
    }
    h->heap[i] = data; // 들어온 값을 저장
}

```

10. 최대 힙프 삽입 함수를 만든다

11. 우선 변수 i를 선언한 후 저장 요소 1개를 추가한다

12. 루트노드가 1이 아니고 들어온 값이 현재 i/2보다 클 경우 계속해서 반복한다

13. i번째와 i/2 노드를 교환 후 i를 2로 나눠주면서 레벨 1칸을 올린다

14. 들어온 값을 저장한다


```

// 표시 함수
void display(HeapType *h, int size)
{
    int i; // i 변수 선언
    printf("===== 동물 히프 출력 =====\n");
    for (i = 1; i <= size; i++) // size 만큼 반복한다
    {
        if (i == size) // 만약 size(총 개수)와 i랑 같을 경우
        {
            printf("%d: %s", h->heap[i].num, h->heap[i].name); // 화살표 없이 출력한다
            return; // 종료한다
        }
        printf("%d: %s => ", h->heap[i].num, h->heap[i].name); // i번째 요소를 출력한다
    }
}

```

15. display 함수는 먼저 i 변수를 선언한다

16. size 만큼 반복하면서 만약 size와 i가 같을 경우 마지막 데이터 이므로 화살표 없이 출력

17. 아닐경우는 그냥 출력

```

// 메인 함수
int main(void)
{
    FILE *fp; // 파일 포인터 fp 선언
    HeapType *heap; // heap 트리 선언
    char temp[100]; // temp 임시 문자열 저장소 설정
    int cnt = 0; // 파일 개수 확인 변수 선언
    element *t; // 임시 저장 구조체 선언
    element tmp; // 히프에 넣어줄 구조체 선언
    int i = 0; // 구조체 인덱스 변수
    int size; // 문자열 크기 변수
}

```

18. 파일 포인터와 트리를 선언한다

19. temp 임시 문자열 저장소, 파일 개수 변수, 히프에 넣어줄 구조체 t와 임시 저장 구조체 tmp를 선언한다

20. 문자열의 크기를 저장하기 위한 size 변수를 선언한다

```

heap = create(); // 히프 트리를 생성한다
init(heap); // 히프 트리를 초기화한다

```

21. 히프 트리를 생성하고 히프 트리를 초기화 한다

```

fp = fopen("data.txt", "r"); // 파일을 오픈한다

// 만약 오픈에 실패하면 종료한다
if (fp == NULL)
{
    printf("파일 오픈 실패\n");
    return 0;
}

// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%d %s\n", &tmp.num, temp); // 파일을 읽어 온다
    cnt++; // 총 개수를 증가시킨다
}

```

22. 파일을 오픈하고 오픈 실패 시 종료한다

23. 파일 끝까지 반복하면서 입력되는 데이터의 개수를 CNT에 저장한다

```

t = (element *)malloc(sizeof(element) * cnt); // 총 추가되는 개수만큼 동적할당을 한다

rewind(fp); // 파일 포인터를 처음으로 돌린다

```

24. t에 개수만큼 동적할당 한 후 CNT의 개수만큼 동적할당을 하고 포인터를 맨 앞으로 돌린다

```

while (!feof(fp)) // 파일 끝까지 반복한다
{
    fscanf(fp, "%d %s\n", &t[i].num, temp); // 파일에 있는 숫자와 문자열을 읽어온다
    size = strlen(temp); // 이름의 크기를 확인한다
    t[i].name = (char *)malloc(sizeof(char) * (size + 1)); // 이름 크기 + 1 만큼 동적할당을 한다
    strcpy(t[i].name, temp); // 문자열을 저장한다
    insert_max_heap(heap, t[i]); // 저장한 구조체를 insert_max_heap 함수를 통해 대입한다
    printf(">> (%d : %s) 입력\n", t[i].num, t[i].name);
    i++;
}

```

25. 파일 끝까지 반복하면서 파일에 있는 숫자와 문자열을 스캔한다

26. 이름의 크기를 저장한 후 이름 크기 + 1만큼 동적할당을 한다

27. 문자열을 저장한 후 INSERT_MAX_HEAP을 통해 삽입한다

28. i++를 통해 총 개수를 늘려준다

```

display(heap, cnt); // 힙 트리를 출력한다

fclose(fp); // 파일을 종료한다
for (int j = 0; j < i; j++)
    free(t[j].name); // 2차원 배열 먼저 해제한다
free(t); // 구조체 해제
free(heap); // 힙 트리를 해제한다
return 0; // 종료한다

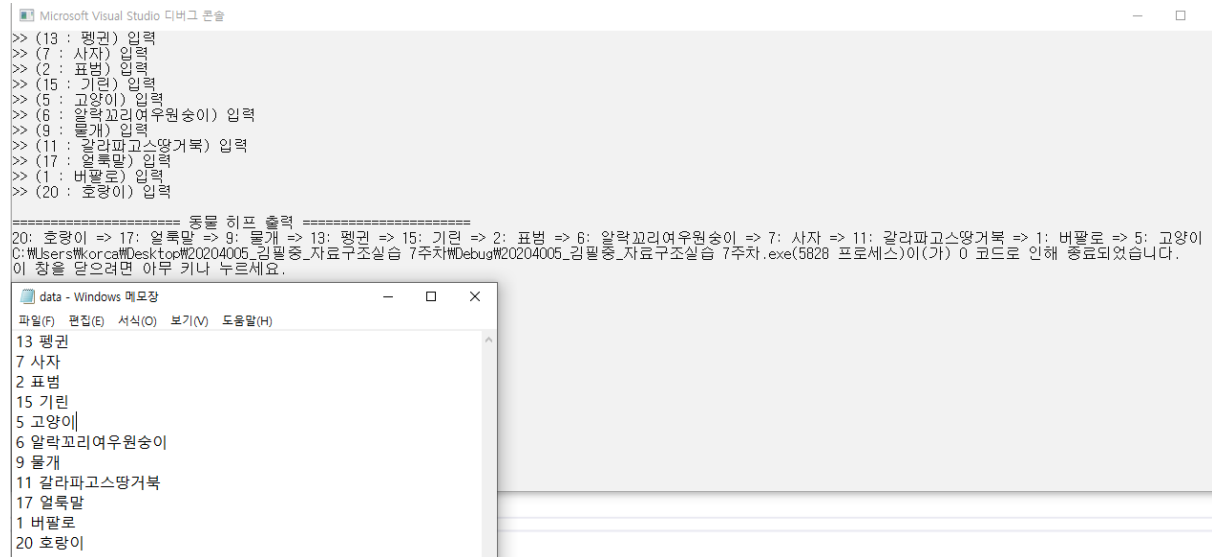
```

29. 힙 트리를 출력한다

30 . 파일을 종료한 후 2차원 배열을 먼저 해제하고 동적할당된 구조체를 해제한다

31. 힙 트리를 해제하고 종료한다

1.4 실행 결과



The screenshot shows the Microsoft Visual Studio IDE. The main window displays the output of a program execution. The output text is as follows:

```
>> (13 : 펭귄) 입력
>> (7 : 사자) 입력
>> (2 : 표범) 입력
>> (15 : 기린) 입력
>> (5 : 고양이) 입력
>> (6 : 알락꼬리여우원숭이) 입력
>> (9 : 물개) 입력
>> (11 : 갈라파고스양귀비) 입력
>> (17 : 얼룩말) 입력
>> (1 : 버팔로) 입력
>> (20 : 호랑이) 입력

===== 동물 히프 출력 =====
20: 호랑이 => 17: 얼룩말 => 9: 물개 => 13: 펭귄 => 15: 기린 => 2: 표범 => 6: 알락꼬리여우원숭이 => 7: 사자 => 11: 갈라파고스양귀비 => 1: 버팔로 => 5: 고양이
C:\Users\korca\Desktop\20204005_김필중_자료구조실습 7주차\Debug\20204005_김필중_자료구조실습 7주차.exe(5828 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

Below the main window, a smaller window titled "data - Windows 메모장" (data - Windows Notepad) is open, showing the same list of animals as entered in the program:

```
13 펭귄
7 사자
2 표범
15 기린
5 고양이
6 알락꼬리여우원숭이
9 물개
11 갈라파고스양귀비
17 얼룩말
1 버팔로
20 호랑이
```

1.5 느낀점

우선 히프 트리를 처음 만들어 봤는데 그림을 안그리고 눈으로만 문제를 보고 풀려고 하니 이해가 잘 안가서 그림을 그리는 것을 제일 먼저 해야겠다고 생각했다. 그리고 일일이 데이터를 넣어가면서 최대 히프 트리의 동작 원리를 알고 코드를 작성하기 시작했다. 이번에 가장 중요하게 생각한 부분은 히프 트리는 배열이고 출력시 배열의 인덱스 순서대로 출력한다 와 삽입함수에서는 최대와 최소의 차이가 존재하고 완전 이진트리 맨 마지막 노드에서 출발해서 하나씩 비교하는 반복문이 중요하다 생각했다. 확실히 이진 트리처럼 히프 트리도 많은 곳에서 사용될 수 있다고 느꼈다.

2.1 문제 분석

■ 손님 관리

- data.txt에 손님의 입장과 퇴장이 이름과 함께 입력되어있다. 각 입장과 퇴장을 히프에 적용하고 이를 히프에 저장된 순서대로 출력하시오.
 - i는 입장, o는 퇴장을 뜻 함
 - 우선 순위는 앞 사람이 퇴장하면 이름순(가나다순)으로 적용됨

```
C:\WINDOWS\system32\cmd.exe
>>손님(손흥민) 입장
< 히프 출력 >
1: 손흥민 =>

>>손님(기성용) 입장
< 히프 출력 >
1: 기성용 => 2: 손흥민 =>

>>손님(조현우) 입장
< 히프 출력 >
1: 기성용 => 2: 손흥민 => 3: 조현우 =>

>>손님(기성용) 퇴장
< 히프 출력 >
1: 손흥민 => 2: 조현우 =>

>>손님(손흥민) 퇴장
< 히프 출력 >
1: 조현우 =>

>>손님(구자철) 입장
< 히프 출력 >
1: 구자철 => 2: 조현우 =>

>>손님(이청용) 입장
< 히프 출력 >
1: 구자철 => 2: 조현우 => 3: 이청용 =>

>>손님(구자철) 퇴장
< 히프 출력 >
1: 이청용 => 2: 조현우 =>

>>손님(이동국) 입장
< 히프 출력 >
1: 이동국 => 2: 조현우 => 3: 이청용 =>

>>손님(박주호) 입장
< 히프 출력 >
1: 박주호 => 2: 이동국 => 3: 이청용 => 4: 조현우 =>

계속하려면 아무 키나 누르십시오 . . .
```

파일(F) 편집(E) 서식(O) 보

i 손흥민
i 기성용
i 조현우
o
o
i 구자철
i 이청용
o
i 이동국
i 박주호

이번 문제는 배열을 이용해 히프를 만들어 텍스트 파일을 읽은 후 히프를 출력하는 것이다. 또한 삭제 함수를 통해 루트 노드를 삭제한 후 맨 마지막 노드를 올려서 다시 위치를 재배치 해야한다. 그러기 위해서는 히프 구조체와 데이터 구조체를 만든 후 초기화를 진행하고 삽입함수, 삭제 함수가 필요하다. 이름이 ㄱ ㄴ ㄷ 순으로 되어있으므로 최소 히프로 제작한다

우선 히프를 정의해야 한다. 히프의 요소가 들어가 있는 구조체와 히프 요소의 1차원 배열을 만들어 히프를 구현한다 히프 구조체에는 저장된 요소의 개수 변수도 추가한다

히프 초기화 함수와 히프 동적 생성 함수를 만들어 준다.

히프에 새로운 요소를 넣어 줄 삽입 함수를 만들어 준다. 이 문제에서는 숫자가 큰 것이 우선이기 때문에 최소 히프 삽입 함수를 만들어 준다. 부모노드와 자식노드를 비교해 사전 순 기준으로 작은 것을 계속 바꿔주면서 코드를 진행하게 만들면 된다.

히프 디스플레이 함수를 만들어 준다. 각 인덱스 번호로 보여주면 되는것이기 때문에 for문을 통해 printf 를 진행하면 된다

삭제 함수를 만들어서 루트 노드를 제거하고 맨 마지막 노드를 가져 온 후 자리를 다시 배치해 주는 함수를 만들어 준다. 부모 노드와 자식 노드를 비교해 작은 것이 위로 가도록 해준다.

또한 동적할당을 통해 이름의 크기를 동적할당을 해준다. 포인터를 이용해 이름의 크기만큼 동적할당 해준다.

2.2 소스 코드

```
1 //-----
2 // 제작일 : 21년 10월 6일 ~ 10월 10일
3 // 제작자 : 20204005 김필중
4 // 프로그램명: 배열 힙프 손님 관리
5 //-----
6
7 // 필요한 헤더파일 선언
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11
12 // 오류 방지 구문
13 #pragma warning(disable : 4996)
14
15
16 // element 구조체 선언
17 typedef struct element
18 {
19     char *name; // 문자열 포인터 선언
20 }element;
21
22 // 힙 트리 구조체 선언
23 typedef struct HeapType
24 {
25     element heap[100]; // 1차원 배열 만들어 힙 구현
26     int heap_size; // 힙프 안에 저장된 요소의 개수
27 }HeapType;
28
29 // 힙 생성 함수
30 HeapType *create()
31 {
32     return (HeapType *)malloc(sizeof(HeapType)); // 동적 할당
33 }
34
35 // 힙프 초기화 함수
36 void init(HeapType *h)
37 {
38     h->heap_size = 0; // 힙프 저장 요소 0으로 초기화
39 }
40
41 // 힙프 최소 힙프 삽입 함수
42 void insert_min_heap(HeapType *h, element data)
43 {
44     int i; // i 변수 선언
45     i = ++(h->heap_size); // 저장 요소 1개 추가
46
47     while ((i != 1) && (strcmp(data.name, h->heap[i / 2].name) < 0)) // 만약 i가 루트노드가 아니고 들어온 이름이 현재 요소 / 2 의 이름 보다 작을 경우
48     {
49         h->heap[i] = h->heap[i / 2]; // i번째 노드와 i/2 노드를 교환
50         i /= 2; // i를 2로 나눴준다(레벨 1개 올라감)
51     }
52     h->heap[i] = data; // 들어온 값을 저장
53 }
54
55
56 // 최소 힙프 제거 함수
57 void delete_min_heap(HeapType *h)
58 {
59     int parent, child; // 부모, 자식 int형 변수 생성
60     element item, temp; // item, temp 구조체 생성
61
62     item = h->heap[1]; // item은 루트 노드
63     temp = h->heap[(h->heap_size)--]; // temp는 마지막 노드를 가리키고 개수 -1을 한다.
64
65     parent = 1; // 부모 1
66     child = 2; // 자식 2
67
68     printf(">> 손님(%s) 퇴장\n", item.name);
69     while (child <= h->heap_size) // 만약 자식 값이 현재 힙프 안에 저장된 요소의 개수보다 작거나 같으면 반복
70     {
71         // 만약 자식 값이 현재 힙프 안에 저장된 요소의 개수보다 작으면서 왼쪽 자식의 이름이 오른쪽 자식의 이름보다 사전 기준으로 앞일 경우
72         if ((child < h->heap_size) && (strcmp(h->heap[child].name, h->heap[child + 1].name) < 0))
73             child++; // 자식 변수에 1을 더한다
74
75         if (strcmp(temp.name, h->heap[child].name) <= 0) // 만약 마지막 노드의 이름이 자식 노드의 이름보다 사전 기준으로 앞일 경우
76             break; // 반복문 종료
77
78         h->heap[parent] = h->heap[child]; // 부모자리에 자식을 저장
79         parent = child; // 부모 변수에 자식 변수 저장
80         child *= 2; // 레벨 하나를 내려간다
81     }
82
83     h->heap[parent] = temp; // 마지막 노드 저장값을 부모자리에 저장한다
84
85
86 // 표시 함수
87 void display(HeapType *h, int size)
88 {
89     int i; // i 변수 선언
90     printf("<힙프 출력>\n"); // size 만큼 반복한다
91     for (i = 1; i <= size; i++)
92     {
93         if(i == size) // 만약 size(총 개수) 와 i랑 같을 경우
94         {
95             printf("%d: %s ", i, h->heap[i].name); // 화살표 없이 출력한다
96             return; // 종료한다
97         }
98         printf("%d: %s => ", i, h->heap[i].name); // i번째 요소를 출력한다
99     }
100 }
```

```

102 int main(void)
103 {
104     FILE *fp; // 파일 포인터 fp 선언
105     HeapType *heap; // heap 트리 선언
106     element *tmp; // 히프에 넣을 구조체 선언
107     char temp[100]; // temp 임시 문자열 저장소 설정
108     int cnt = 0; // 파일 개수 확인 변수 선언
109     int count = 0; // 동적할당 개수 설정
110     int i = 0; // 구조체 인덱스 변수
111     int size; // 문자열 크기 변수
112
113     heap = create(); // 히프 트리를 생성한다
114     init(heap); // 히프 트리를 초기화한다
115
116     fp = fopen("data2.txt", "r"); // 파일을 오픈한다
117
118     // 만약 오픈에 실패하면 종료한다
119     if (fp == NULL)
120     {
121         printf("파일 오픈 실패\n");
122         return 0;
123     }
124
125     // 파일 끝까지 반복한다
126     while (!feof(fp))
127     {
128         fscanf(fp, "%s\n", temp); // 파일을 읽어 온다
129         if (strcmp(temp, "i") == 0) // 만약 i일 경우
130             count++; // 총 개수를 증가시킨다
131     }
132     rewind(fp); // 파일 포인터를 처음으로 돌린다
133     tmp = (element *)malloc(sizeof(element) * count); // 총 추가되는 개수만큼 동적할당을 한다
134
135
136     while (!feof(fp)) // 파일 끝까지 반복한다
137     {
138         fscanf(fp, "%s", temp); // 파일 첫번째를 읽는다
139
140         if (strcmp(temp, "i") == 0) // 만약 첫번째가 i 일경우는 삽입 함수 실행
141         {
142             fscanf(fp, "%s\n", temp); // temp에 이름을 임시로 저장한다
143             size = strlen(temp); // 이름의 크기를 확인한다
144             tmp[i].name = (char *)malloc(sizeof(char) * (size + 1)); // 이름 크기 + 1 만큼 동적할당을 한다
145             strcpy(tmp[i].name, temp); // tmp 구조체에 삽입한다
146             insert_min_heap(heap, tmp[i]); // 최소 히프 삽입 함수에 넣어준다
147             printf(">> 손님(%s) 입장\n", tmp[i].name);
148             cnt++; // 총 인원수를 증가시킨다
149             display(heap, cnt); // 표시한다
150             printf("\n\n");
151             i++; // 인덱스 1 증가시킨다
152         }
153
154         if (strcmp(temp, "o") == 0) // 만약 첫번째가 o 일경우는 제거 함수 실행
155         {
156             delete_min_heap(heap); // 최소 히프 제거 함수 실행
157             cnt--; // 총 인원수 - 하기
158             display(heap, cnt); // 표시한다
159             printf("\n\n");
160         }
161     }
162
163
164
165     fclose(fp); // 파일을 종료한다
166     for (int j = 0; j < i; j++)
167         free(tmp[j].name); // 2차원 배열 먼저 해제한다
168     free(tmp); // 동적할당을 해제한다
169     free(heap); // 히프 트리를 해제한다
170     return 0; // 종료한다
171 }
172
173

```


2.3 소스 코드 분석

```
// 필요한 헤더파일 선언
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 오류 방지 구문
#pragma warning (disable : 4996)
```

1. 필요한 헤더파일을 선언한다
2. 오류 방지 구문을 선언한다.

```
// element 구조체 선언
typedef struct element
{
    char *name; // 문자열 포인터 선언
}element;

// 힙 트리 구조체 선언
typedef struct HeapType
{
    element heap[100]; // 1차원 배열 만들어 힙 구현
    int heap_size; // 힙 안에 저장된 요소의 개수
}HeapType;
```

3. element 구조체를 선언해 요소들의 구조체를 만든다.
4. 이름을 포인터로 선언해 크기만큼 동적할당을 하기 위함이다.
5. 힙 트리 구조체를 선언한다
6. 1차원 배열을 만들어 힙을 구현한다
7. 힙 안에 저장된 요소의 개수 변수를 정한다

```

// 힙 생성 함수
HeapType *create()
{
    return (HeapType *)malloc(sizeof(HeapType)); // 동적 할당
}

// 힙프 초기화 함수
void init(HeapType *h)
{
    h->heap_size = 0; // 힙프 저장 요소 0으로 초기화
}

```

8. 힙프 생성 함수를 통해 동적할당을 한다

9. 힙프 초기화 함수를 통해 힙프 저장 요소를 0으로 초기화 한다.

```

// 힙프 최소 힙프 삽입 함수
void insert_min_heap(HeapType *h, element data)
{
    int i; // i 변수 선언
    i = ++(h->heap_size); // 저장 요소 1개 추가
    while ((i != 1) && (strcmp(data.name, h->heap[i / 2].name) < 0)) // 만약 i가 루트노드가 아니고 들어온 이름이 현재 요소 / 2 의 이름 보다 작을 경우
    {
        h->heap[i] = h->heap[i / 2]; // i번째 노드와 i/2 노드를 교환
        i /= 2; // i를 2로 나눠준다(레벨 1개 올라감)
    }
    h->heap[i] = data; // 들어온 값을 저장
}

```

10. 최소 힙프 삽입 함수를 만든다

11. 우선 변수 i를 선언한 후 저장 요소 1개를 추가한다

12. 루트노드가 1이 아니고 들어온 값이 현재 i/2보다 사전 순 기준으로 더 앞일 경우 반복한다

13. i번째와 i/2 노드를 교환 후 i를 2로 나눠주면서 레벨 1칸을 올린다

14. 들어온 값을 저장한다

```

// 최소 힙 제거 함수
void delete_min_heap(HeapType *h)
{
    int parent, child; // 부모, 자식 int형 변수 생성
    element item, temp; // item, temp 구조체 생성

    item = h->heap[1]; // item은 루트 노드
    temp = h->heap[(h->heap_size)--]; // temp는 마지막 노드를 가리키고 개수 -1을 한다.

    parent = 1; // 부모 1
    child = 2; // 자식 2

    printf(">> 손님(%s) 퇴장\n", item.name);
}

```

15. 부모 자식 변수를 생성, item, temp 구조체를 생성한다

16. item에는 루트 노드를 저장하고, temp에는 마지막 노드를 가리킨 후 개수를 -1 한다

17. 부모는 1 자식은 2를 저장한다(부모 자식 노드를 맨 위에서부터 비교하기 위함)

```

while (child <= h->heap_size) // 만약 자식 값이 현재 힙에 저장된 요소의 개수보다 작거나 같으면 반복
{
    // 만약 자식 값이 현재 힙에 저장된 요소의 개수보다 작으면서 왼쪽 자식의 이름이 오른쪽 자식의 이름보다 사전 기준으로 앞일 경우
    if ((child < h->heap_size) && (strcmp(h->heap[child].name, h->heap[child + 1].name) < 0))
        child++; // 자식 변수에 1을 더한다

    if (strcmp(temp.name, h->heap[child].name) <= 0) // 만약 마지막 노드의 이름이 자식 노드의 이름보다 사전 기준으로 앞일 경우
        break; // 반복문 종료

    h->heap[parent] = h->heap[child]; // 부모자리에 자식을 저장
    parent = child; // 부모 변수에 자식 변수 저장
    child *= 2; // 레벨 하나를 내려간다
}

h->heap[parent] = temp; // 마지막 노드 저장값을 부모자리에 저장한다

```

18. 만약 자식의 값이 현재 힙에 저장된 요소의 개수보다 작거나 같으면 반복한다

19. 만약 자식의 값이 현재 힙에 저장된 요소의 개수보다 작으면서 왼쪽 자식의 이름이 오른쪽 자식의 이름보다 사전 순으로 앞일 경우 자식 변수에 1을 더한다

20. 마지막 노드의 이름이 (현재 맨 위로 올라 와있다고 생각) child 번째 인덱스와 비교해서 앞이거나 같을 경우는 반복문을 빠져나온다

21. 아닐 경우는 루트 자리에 자식을 저장하고 부모 변수에 자식 변수 저장 레벨 하나 다운시킨다

22. 마지막 노드의 값을 부모자리에 저장한다.

```

// 표시 함수
void display(HeapType *h, int size)
{
    int i; // i 변수 선언
    printf("<히프 출력>\n"); // size 만큼 반복한다
    for (i = 1; i <= size; i++)
    {
        if(i == size) // 만약 size(총 개수)와 i랑 같을 경우
        {
            printf("%d: %s ", i, h->heap[i].name); // 화살표 없이 출력한다
            return; // 종료한다
        }
        printf("%d: %s => ", i, h->heap[i].name); // i번째 요소를 출력한다
    }
}

```

23. display 함수는 먼저 i 변수를 선언한다

24. size 만큼 반복하면서 만약 size와 i가 같을 경우 마지막 데이터 이므로 화살표 없이 출력

25. 아닐경우는 i번째 요소 그냥 출력

```

FILE *fp; // 파일 포인터 fp 선언
HeapType *heap; // heap 트리 선언
element *tmp; // 히프에 넣어줄 구조체 선언
char temp[100]; // temp 임시 문자열 저장소 설정
int cnt = 0; // 파일 개수 확인 변수 선언
int count = 0; // 동적할당 개수 설정
int i = 0; // 구조체 인덱스 변수
int size; // 문자열 크기 변수

heap = create(); // 히프 트리를 생성한다
init(heap); // 히프 트리를 초기화한다

fp = fopen("data2.txt", "r"); // 파일을 오픈한다

// 만약 오픈에 실패하면 종료한다
if (fp == NULL)
{
    printf("파일 오픈 실패\n");
    return 0;
}

```

26. 파일 포인터와 히프 트리, 구조체 포인터를 선언한다

27. 임시 저장 구조체 tmp, 파일 개수 확인 변수, 동적할당 개수 변수, 문자열 크기 변수를 선언한다

28. 힙 트리를 생성하고 힙 트리를 초기화 한다

29. 파일을 열고 열고 실패 시 종료한다

```
// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%s\n", temp); // 파일을 읽어 온다
    if (strcmp(temp, "i") == 0) // 만약 i일 경우
        count++; // 총 개수를 증가시킨다
}
rewind(fp); // 파일 포인터를 처음으로 돌린다
tmp = (element *)malloc(sizeof(element) * count); // 총 추가되는 개수만큼 동적할당을 한다
```

30. 파일 끝까지 반복하면서 맨 앞이 i일 때

31. 입력되는 데이터의 개수를 CNT에 저장한다

32. count의 개수만큼 동적할당을 하고 포인터를 맨 앞으로 돌린다

```
while (!feof(fp)) // 파일 끝까지 반복한다
{
    fscanf(fp, "%s", temp); // 파일 첫번째를 읽는다

    if (strcmp(temp, "i") == 0) // 만약 첫번째가 i 일경우는 삽입 함수 실행
    {
        fscanf(fp, "%s\n", temp); // temp에 이름을 임시로 저장한다
        size = strlen(temp); // 이름의 크기를 확인한다
        tmp[i].name = (char *)malloc(sizeof(char) * (size + 1)); // 이름 크기 + 1 만큼 동적할당을 한다
        strcpy(tmp[i].name, temp); // tmp 구조체에 삽입한다
        insert_min_heap(heap, tmp[i]); // 최소 힙 삽입 함수에 넣어준다
        printf(">> 손님(%s) 입장\n", tmp[i].name);
        cnt++; // 총 인원수를 증가시킨다
        display(heap, cnt); // 표시한다
        printf("\n\n");
        i++; // 인덱스 1 증가시킨다
    }

    if (strcmp(temp, "o") == 0) // 만약 첫번째가 o 일경우는 제거 함수 실행
    {
        delete_min_heap(heap); // 최소 힙 제거 함수 실행
        cnt--; // 총 인원수 - 하기
        display(heap, cnt); // 표시한다
        printf("\n\n");
    }
}
```

33. 파일 첫번째를 읽은 후 i일 경우

34. 삽입 함수를 진행한다.

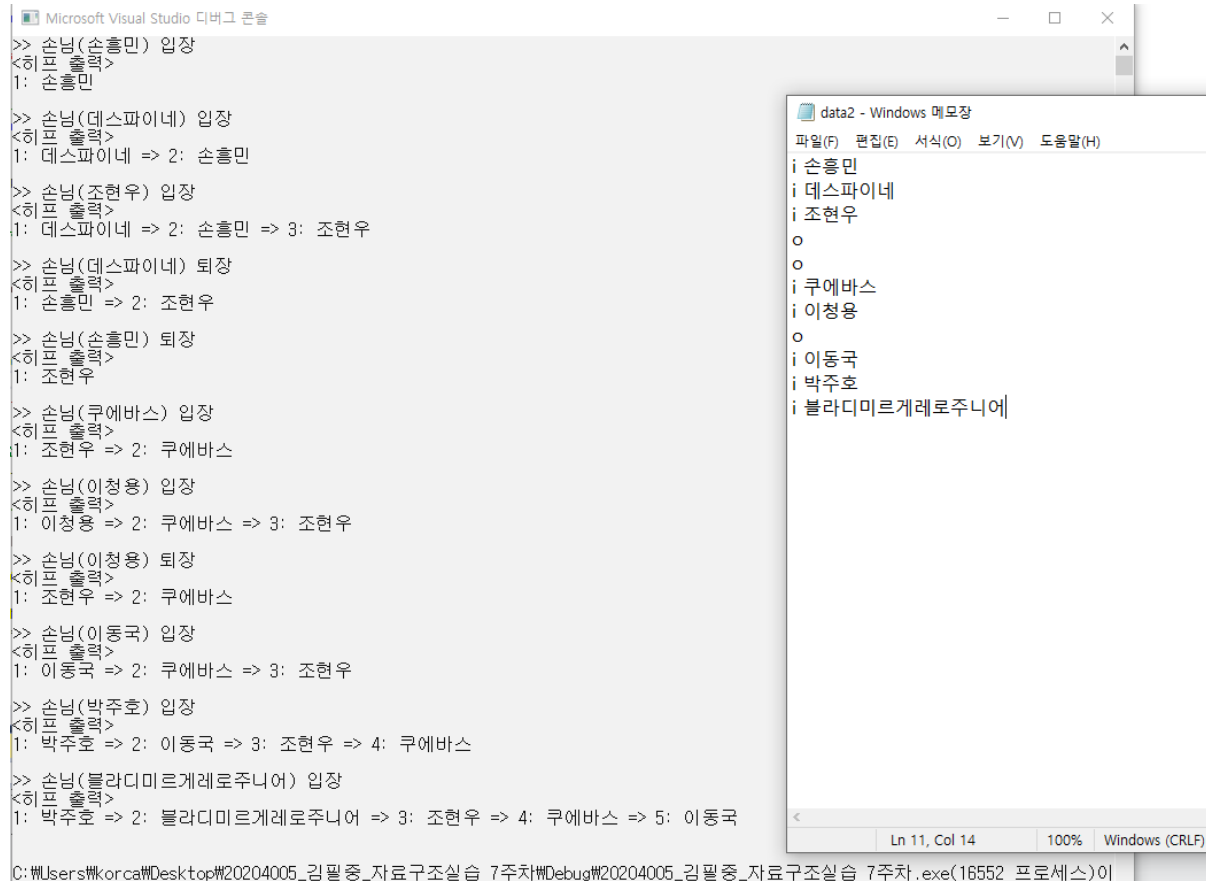
35. 파일에 있는 이름을 읽은 후 크기를 저장, 크기 + 1만큼 동적할당을 한다

- 36. 문자열을 저장한 후 INSERT_min_HEAP을 통해 삽입한다
- 37. 총 인원수를 증가시키고 display를 통해 표시한다.
- 38. l++를 통해 총 개수를 늘려준다
- 39. 만약 o일 경우는 제거 함수를 실행한다
- 40. 최소 힙 제거 함수를 실행시키고 총 인원수를 1 제거한다
- 41. 표시하고 종료한다

```
fclose(fp); // 파일을 종료한다
for (int j = 0; j < i; j++)
    free(tmp[j].name); // 2차원 배열 먼저 해제한다
free(tmp); // 동적할당을 해제한다
free(heap); // 힙 트리를 해제한다
return 0; // 종료한다
```

- 42. 파일을 종료한 후 2차원 배열을 먼저 해제하고 동적할당 된 구조체를 해제한다
- 43. 힙 트리를 해제하고 종료한다

2.4 실행 결과



```
Microsoft Visual Studio 디버그 콘솔
>> 손님(손흥민) 입장
<히프 출력>
1: 손흥민

>> 손님(데스파이네) 입장
<히프 출력>
1: 데스파이네 => 2: 손흥민

>> 손님(조현우) 입장
<히프 출력>
1: 데스파이네 => 2: 손흥민 => 3: 조현우

>> 손님(데스파이네) 퇴장
<히프 출력>
1: 손흥민 => 2: 조현우

>> 손님(손흥민) 퇴장
<히프 출력>
1: 조현우

>> 손님(쿠에바스) 입장
<히프 출력>
1: 조현우 => 2: 쿠에바스

>> 손님(이청용) 입장
<히프 출력>
1: 이청용 => 2: 쿠에바스 => 3: 조현우

>> 손님(이청용) 퇴장
<히프 출력>
1: 조현우 => 2: 쿠에바스

>> 손님(이동국) 입장
<히프 출력>
1: 이동국 => 2: 쿠에바스 => 3: 조현우

>> 손님(박주호) 입장
<히프 출력>
1: 박주호 => 2: 이동국 => 3: 조현우 => 4: 쿠에바스

>> 손님(블라디미르게레로주니어) 입장
<히프 출력>
1: 박주호 => 2: 블라디미르게레로주니어 => 3: 조현우 => 4: 쿠에바스 => 5: 이동국

data2 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
i 손흥민
i 데스파이네
i 조현우
o
o
i 쿠에바스
i 이청용
o
i 이동국
i 박주호
i 블라디미르게레로주니어
Ln 11, Col 14 100% Windows (CRLF)
```

2.5 느낀점

이번에는 최소 히프 트리를 이용해 이름 순으로 히프 트리를 짤 후 o가 들어오면 루트 노드를 제거하고 마지막 노드를 맨 위로 올리고 다시 재배치하는 문제이다. Strcmp를 통해 사전순으로 만드는 방법을 계속해서 해봤지만 아직은 더 생각을 많이 해야겠다고 생각했다. 또한 이번에는 최소이기 때문에 작은 순이라 앞에 했던 과제랑 반대로 해야한다는 것을 계속해서 생각했다. 이번에도 이름 여러 개 써놓고 직접 그리면서 과제를 진행했다. 최소 히프 트리도 많은 곳에서 사용될 수 도 있다는 사실을 알게 되었다.