3주차 : Priority Queue

알 고 리 즘

2015. 9. 17.

충남대학교 컴퓨터공학과 임베디드 시스템 연구실
TA 권진세

Overview

- **Heap (힙)**
  - Heap이란? – 사전적 의미 : 더미 덩어리 모래산
  - 최대 힙(Max Heap)과 최소 힙(Min Heap)
  - 힙의 구현

- **Priority Queue (우선 순위 큐)**
  - 우선 순위 큐란?
  - 우선 순위 큐의 주요 기능 및 구현 원리

- **실습 / 과제**

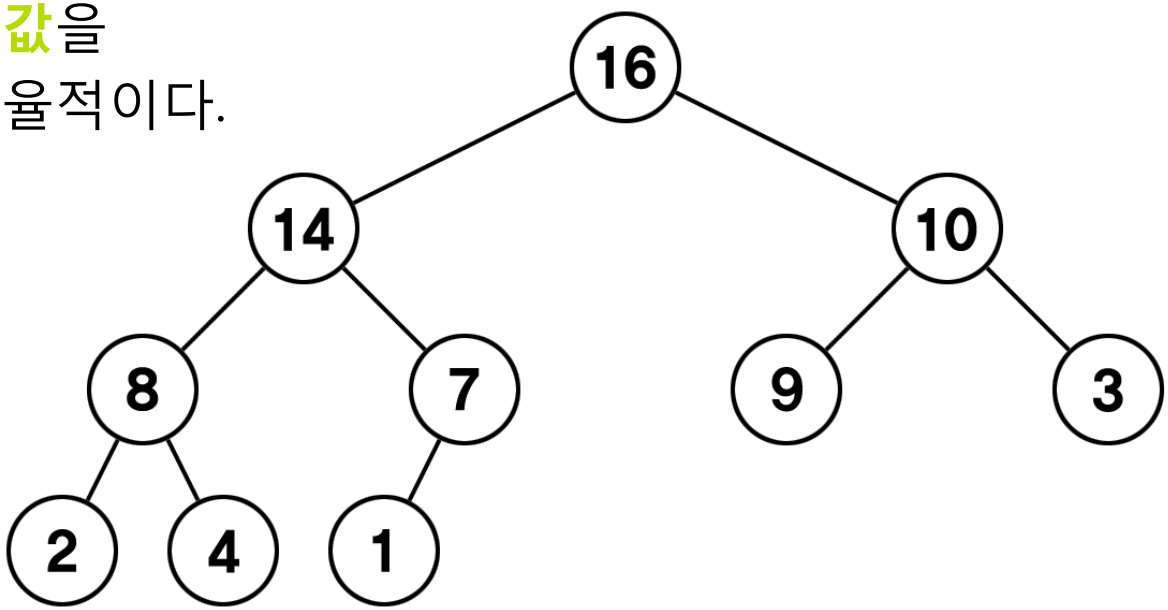
  우선 순위와 **작업명**을 갖는 Priority Queue 구현

Heap

▶ Heap이란? (1/3)

완전 이진 트리를 기반으로 하는 배열형 자료구조.

최대값 또는 **최소값**을
빠르게 찾는데 효율적이다.



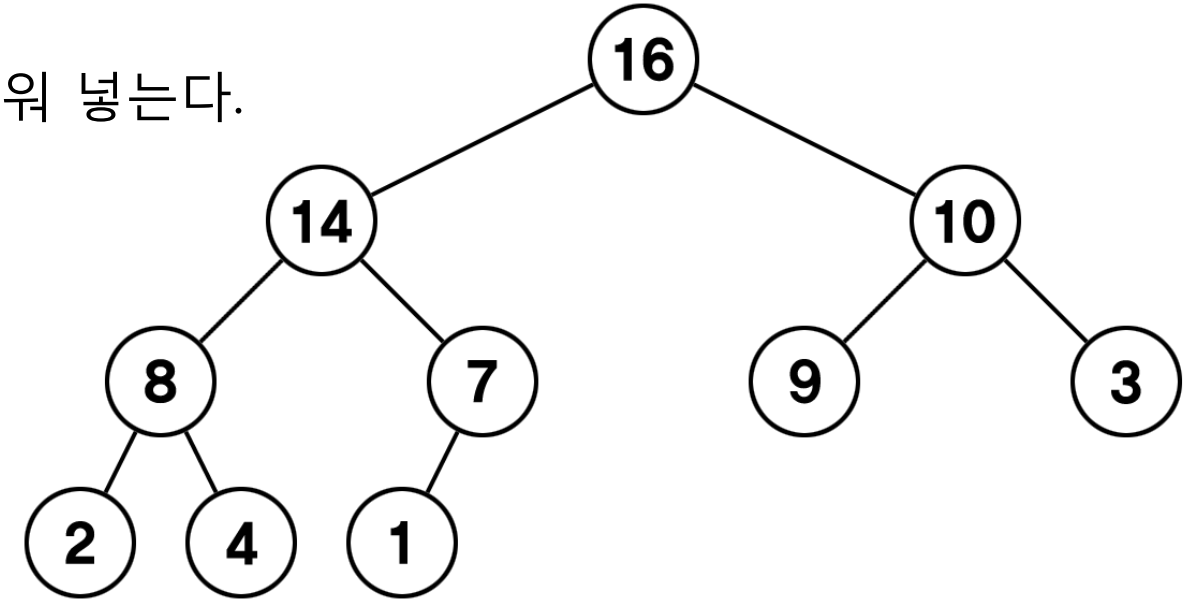
16	14	10	8	7	9	3	2	4	1
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Heap

▶ Heap이란? (2/3)

이 트리는 가장 낮은 층을 제외하고는 완전히 차 있고,

가장 낮은 층은
왼쪽에서부터 채워 넣는다.



16	14	10	8	7	9	3	2	4	1
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Heap

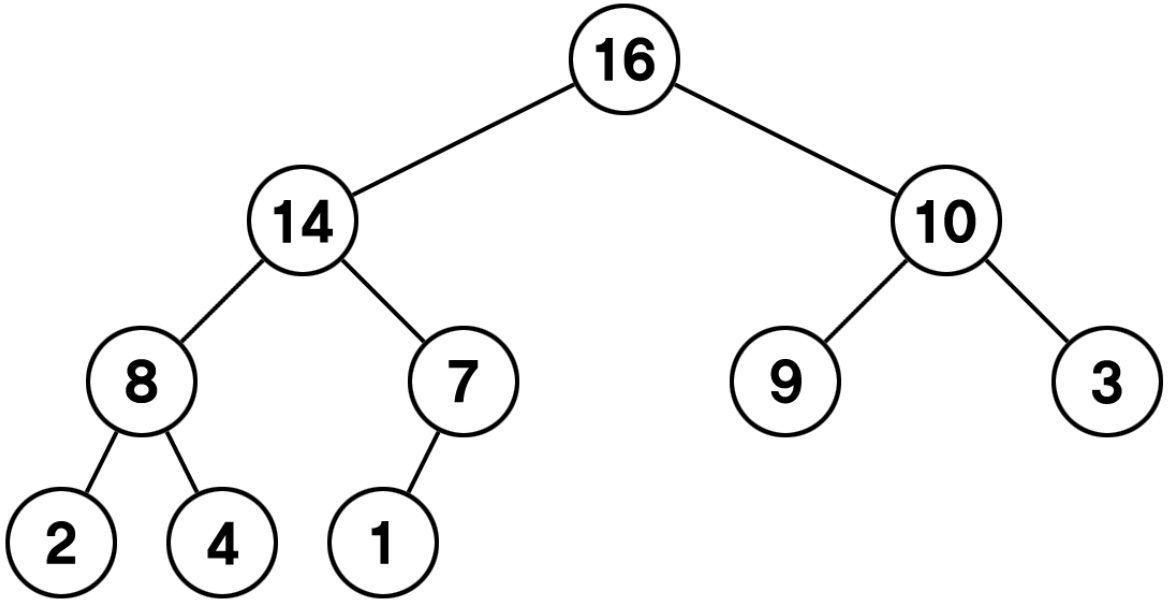
▶ Heap이란? (3/3)

어떤 노드가 저장된 인덱스를 i라 할 때,

PARENT*(i)*
  return ***⌊ i/2 ⌋***

LEFT-CHILD*(i)*
  return ***2i***

RIGHT-CHILD*(i)*
  return ***2i+1***



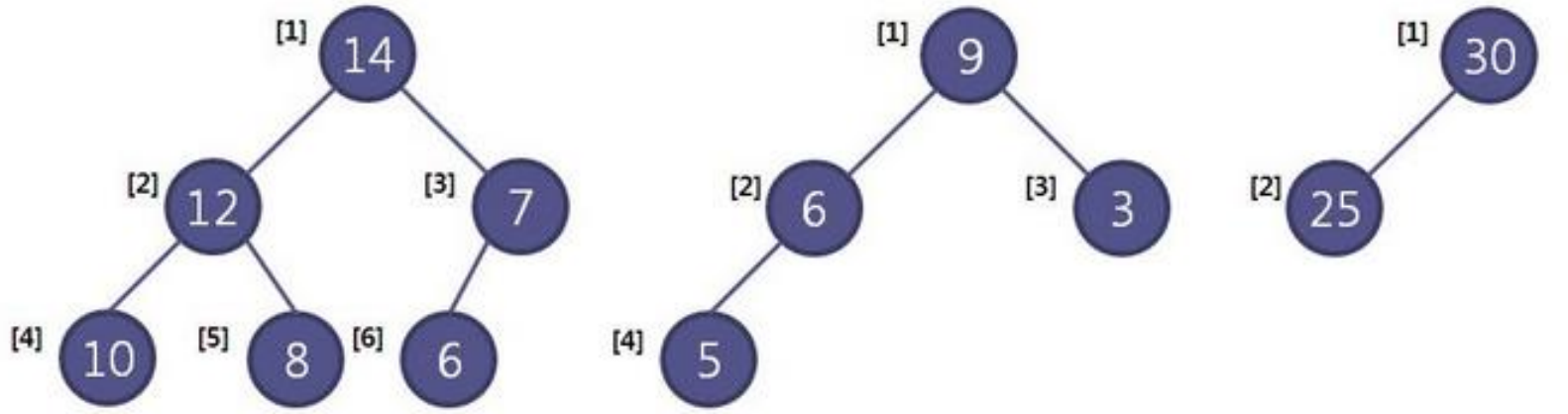
16	14	10	8	7	9	3	2	4	1
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Heap

▶ 최대 힙과 최소 힙

1) 최대 힙 (Max Heap) : **루트에 최대값이 저장**된다.

부모의 키 값 ≥ 자식의 키 값



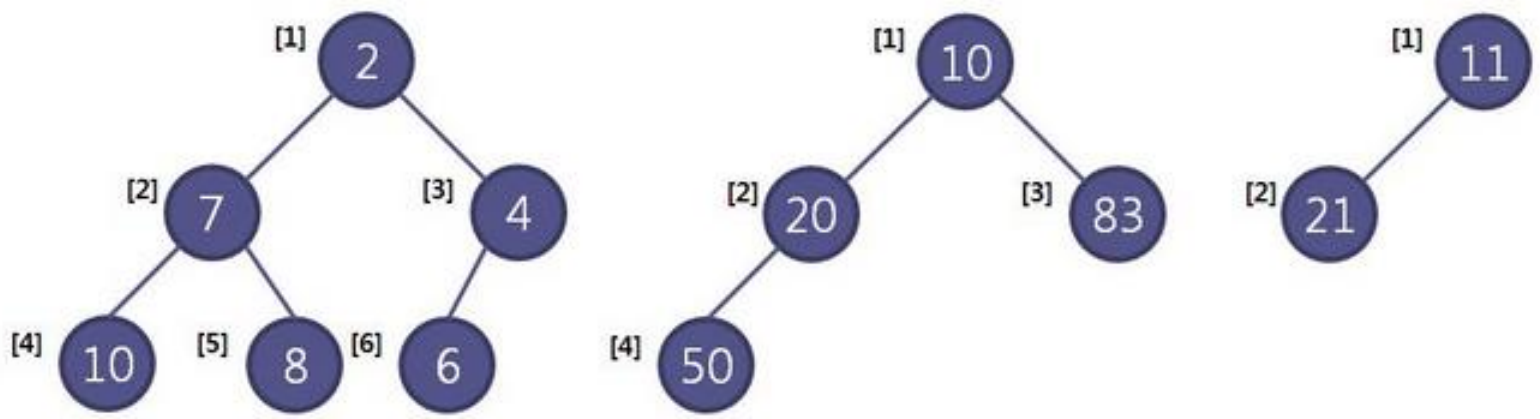
[그림 1] 최대 히프의 예

Heap

▶ 최대 힙과 최소 힙

2) 최소 힙 (Min Heap) : **루트에 최소값이 저장**된다.

부모의 키 값 ≤ 자식의 키 값



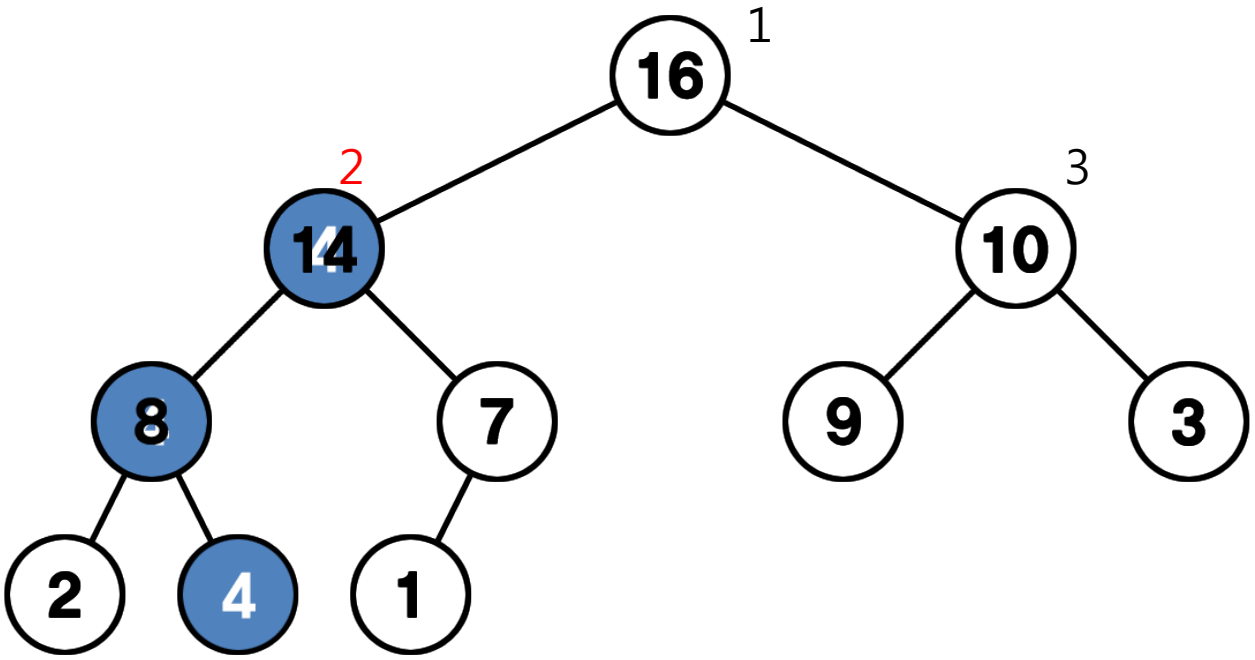
[그림 2] 최소 히프의 예

Heap

▶ 힙의 구현

MAX-HEAPIFY (S,2)

특정 노드를 기점으로, 트리를 내려가며 최대 힙이 되도록 함



Heap

▶ 힙의 구현

```
MAX-HEAPIFY(A,i)
    L ← LEFT-CHILD(i)
    R ← RIGHT-CHILD(i)
    if L ≤ heap_size[A] and A[L] > A[i]
        then largest ← L
        else largest ← i
    if R ≤ heap_size[A] and A[R] > A[largest]
        then largest ← R
    if largest ≠ i
        then A[i] ↔ A[largest]
            MAX-HEAPIFY(A,largest)
```

Heap

▶ 힙의 구현

BUILD-MAX-HEAP(A)

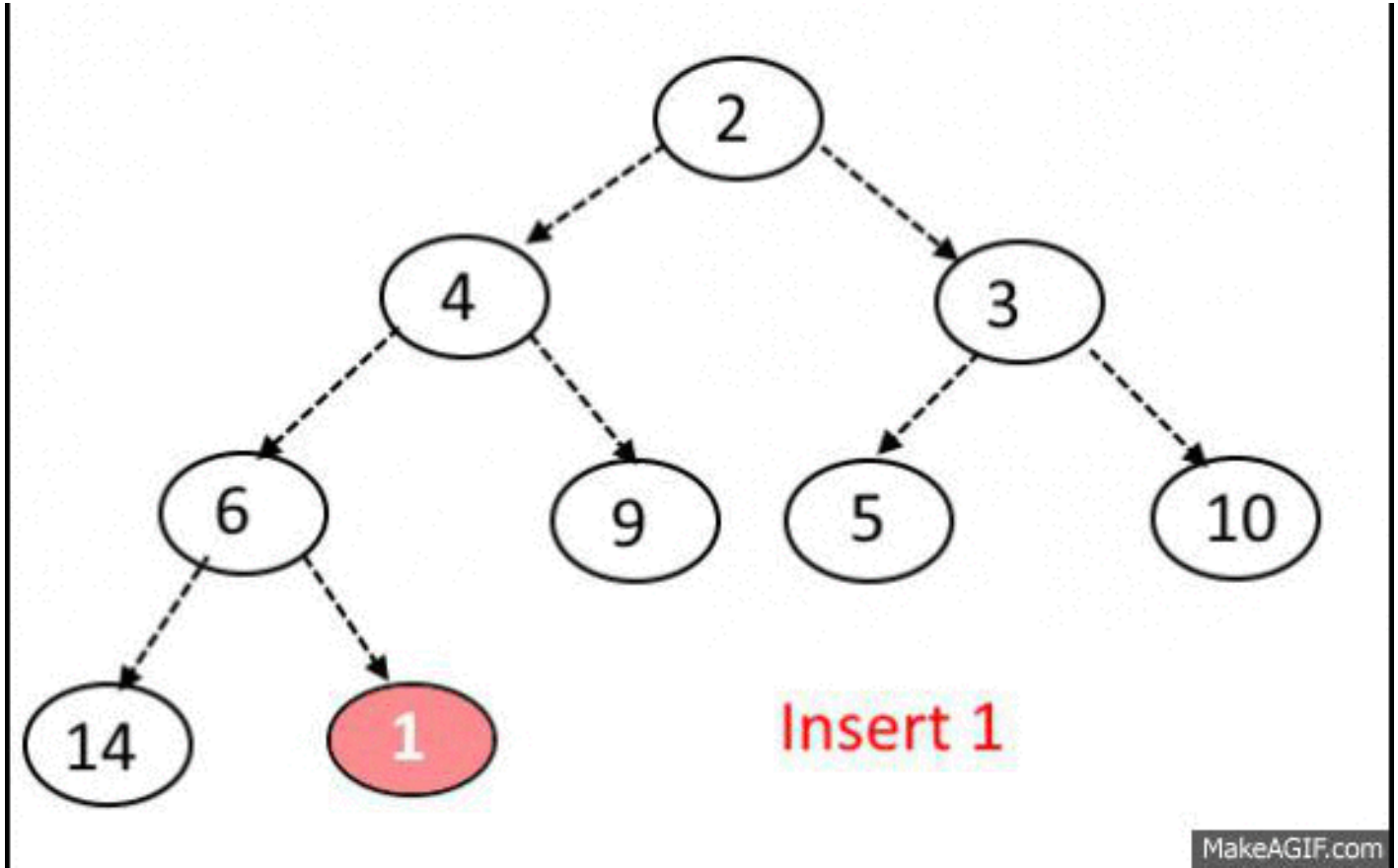
입력 받은 **배열 A를 최대 힙으로 만드는 함수**.

자식을 갖는 마지막 노드부터 루트 노드까지
순서대로 검사하며 MAX-HEAPIFY 함수를 실행한다.

BUILD-MAX-HEAP*(A)*
    heap_size[A] ← ***length[A]***
    for ***i*** ← ***⌊ length[A]/2 ⌋*** **downto** ***1***
        do MAX-HEAPIFY*(A,i)*

Heap

- **Insert(S)**
  = 배열의 마지막 원소 추가 -> Build_Max_Heap

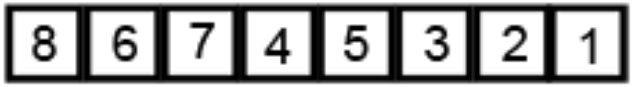


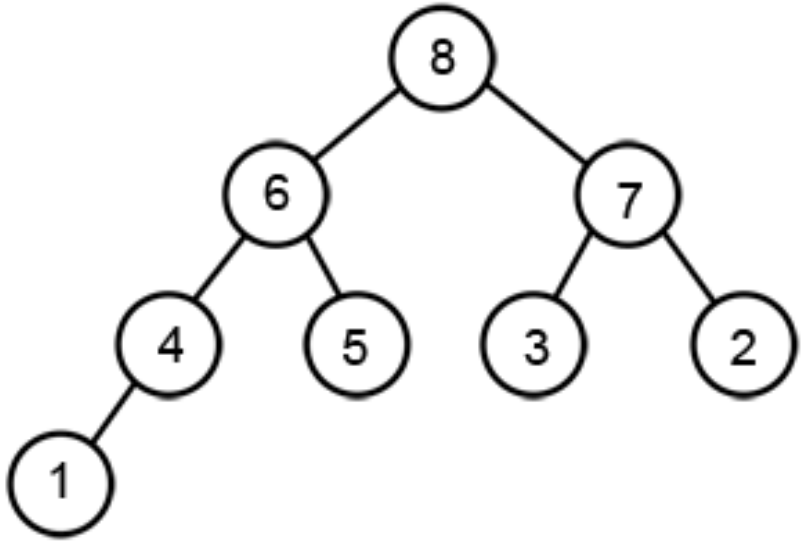
Heap

- **Complete Heap Sort**

  loop Extract_max -> BUILD-MAX-HEAP(S)

  end node = 0





Practice / Homework

data03.txt 파일의 데이터로 부터 다음과 같은 함수를 구현하라

a. insert(S, x) - 배열 S의 마지막 원소 뒤에 원소 x를 추가하라
(x = 9)

b. H_max(S) - 배열 S에서의 최대값을 리턴하라 (Heap Sort)

c. extract_max(S) - 배열 S에서 heap_sort를 수행한 후 1번째 리턴
> 부모 노드(1번)를 삭제하고 Heap Sort

d. increase_key(S, x, k) - 배열 S에서 x번째 원소를 k값으로 바꾸어라
(x = 1 , k = 5 로 고정)
(단 k값이 본래 원소보다 클 때만 바꾸어라)

e. H_delete(S, x) - 배열 S에서 x번째 원소를 삭제하라
(x = 5 로 고정)

Practice / Homework

data03.txt 파일의 데이터로 부터 다음과 같은 함수를 구현하라

f. Build-Max-Heap(S) - 배열 S에 heap sort를 구현하라
b.H_max(S)와 동일한 함수임.
(2진 트리 에서만 만족하면 된다)

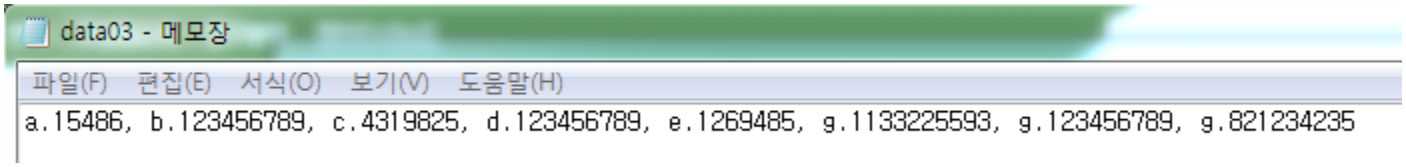
g. complete_heap_sort(S)

- 배열 S에 heap sort를 이용하여 정렬하라.
(중요한 부분이므로 신중히 코딩)
(내림차순의 완벽한 정렬을 수행하라)

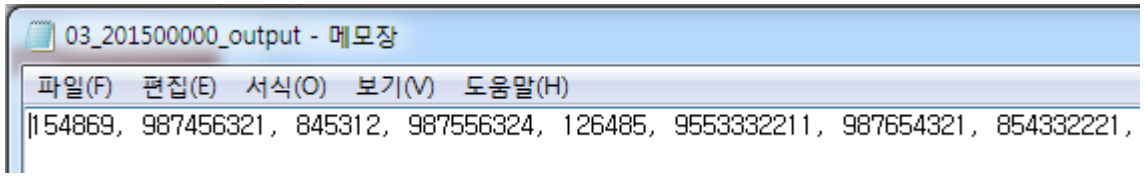
Practice / Homework

▶ **input data : data03.txt**

실제 채점 시에는 더 많은 데이터를 사용하여 채점하며 양식은 동일함.



▶ **결과 예시 : 03_201500000_output.txt**



Practice / Homework

※ 그 외 실습 과제 수행 중 유의 사항

- 포함내용 : 프로젝트 파일 전체 압축
  ※ sorting된 output 파일도 함께 압축 후 전송
  ※ 매주 실습 보고서도 함께 제출할 것

- 제출이름 :
  메일 : [알고리즘00반]_201500000_홍길동_3주차
  파일 : [알고리즘00반]_201500000_홍길동_3주차.zip

- 제출기한 : 2015-09-24 18:00까지

- 메일주소 : kwonse@cnu.ac.kr

Practice / Homework

※ 그 외 실습 과제 수행 중 유의 사항

- 보고서 꼭 작성해 주세요.

알고리즘 ()주차 실습 보고서

학번 : 201500000
이름 : 홍 길 동

Practice ( )번 소스 코드

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

void main(void){
        ...
        printf("hello algorithm₩n");
        ...
}
```

Practice ( )번 코드 설명 (및 성능 평가)

...
...
위 코드는 standard I/O 헤더의 printf 함수를 사용하여 hello algorithm을 출력하는 소스 코드입니다.
...
...

Practice ( )번 결과화면 캡쳐

혹은

APPENDIX 1. File I/O

1. 파일 입출력 방법

```
FILE* fp;          //fp : input file pointer
FILE* fop;         //fop : input file pointer

//파일 이름은 "00_201500000_insertion.txt"
//입출력 파일은 *.c 소스파일과 같은 폴더에 있어야 한다.
fp = fopen(FILENAME,"rt");                  //입력 파일 열기
fop = fopen(FILENAME2,"wt");                //출력 파일 열기

if (fp == NULL) {
        printf("**** Input File open error ****₩n");
        exit(1);
} //파일 없을 경우 예외처리로 프로그램 종료
```

APPENDIX 1. File I/O (계속)

```
while(!feof(fp)){
} //파일의 끝날때 까지 반복

fscanf(fp,"%d, ", &변수);          // ex) 123, 456,
                                   // int 값만 추출함 ', '는 제외됨

fprintf(fop, "%d", 출력할 값);  // 파일 출력 시 사용

fclose(fp);
```

APPENDIX 2. 배열 넘기기

1. Main 함수의 배열을 주소로 넘겨서 다룰 때 !! 이중포인터 사용
   - 장점 : 메모리 절약, 리턴 불필요.

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

void user_malloc(int** num);

void main(void){
    int *ptr;
    user_malloc(&ptr);      //포인터 변수 ptr의 주소를 인자로 보냄.
    printf("%d₩n", *ptr);  //출력 값은 10 이다.
    return 0;
}

void user_malloc(int** num){
    *num = (int*)malloc(sizeof(int));
    (*num)[0] = 10;
}
```

APPENDIX 3. 동적 할당 메모리 크기

Q & A.

포인터로 받은 배열의 크기를 구하는 방법? (있다)

- Malloc 함수의 선언을 보면 void* malloc(size_t size)
- Size_t는 많은경우 unsinged long int로 되어있으므로
- 메모리를 할당 할 때 이 크기만큼 더 할당해서 할당 영역의 처음 부분에 길이의 값을 저장해 두고 있음.

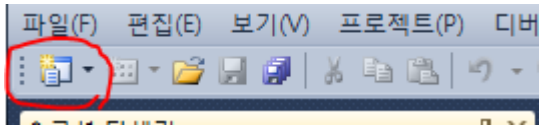
- *(ptr - sizeof(size_t))
- **다음과 같은 함수로 만들어 사용 가능**

```
int sizeof_ar(int* S){
    int size;
    size = *(S - sizeof(int));
    return size;
}
```

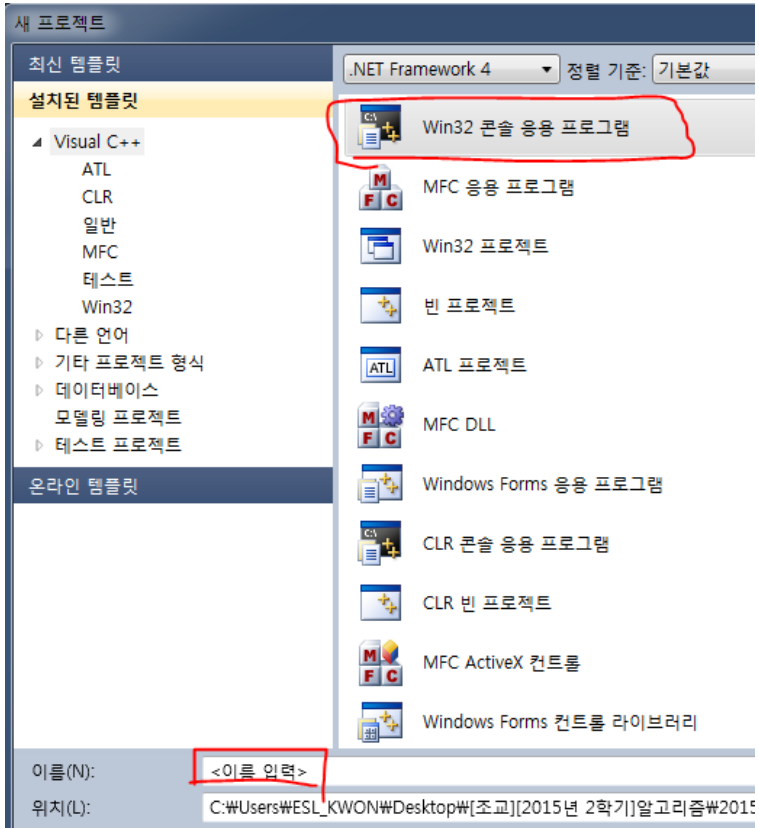
Trouble Shooting

Visual Studio 2010 프로젝트 생성

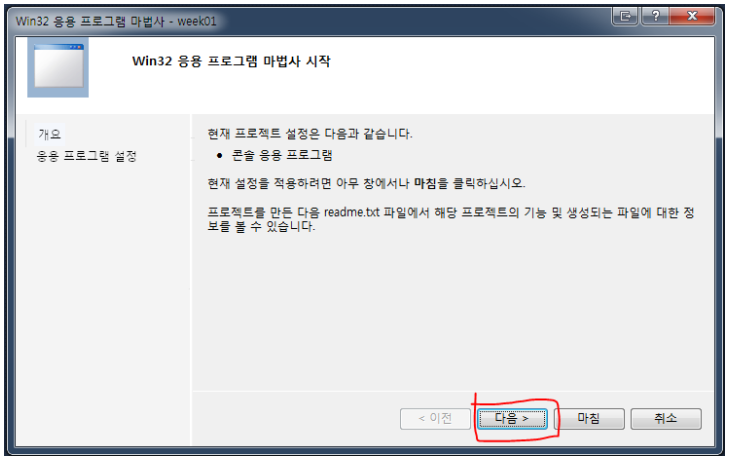
1.



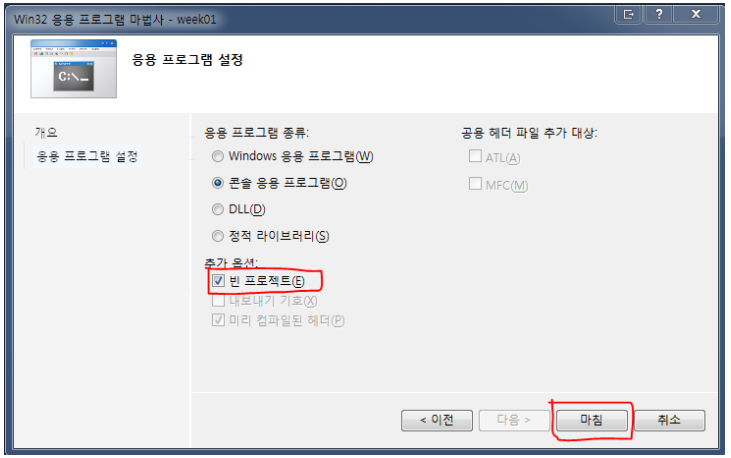
2.



3.



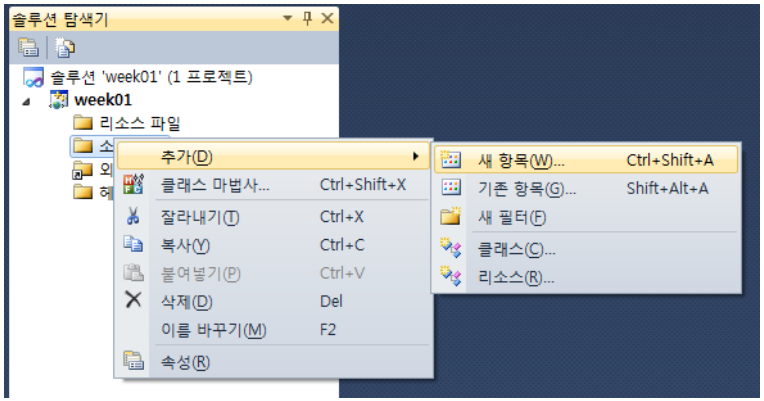
4.



Trouble Shooting

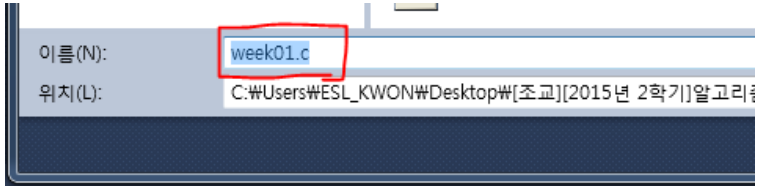
Visual Studio 2010 프로젝트 생성

5.



6.

7.



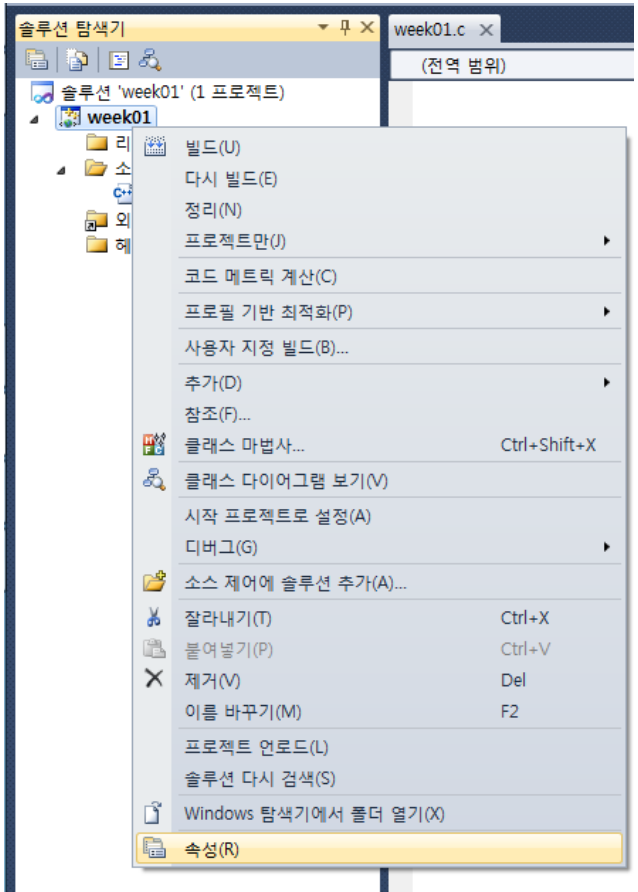
반드시 .c 로 이름 변경!!

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결

```
1>LINK : fatal error LNK1123: COFF로 변환하는 동안 오류가 발생했습니다. 파일이 잘못되었거나 손상되었습니다.
1>
1>빌드하지 못했습니다.
```

1.

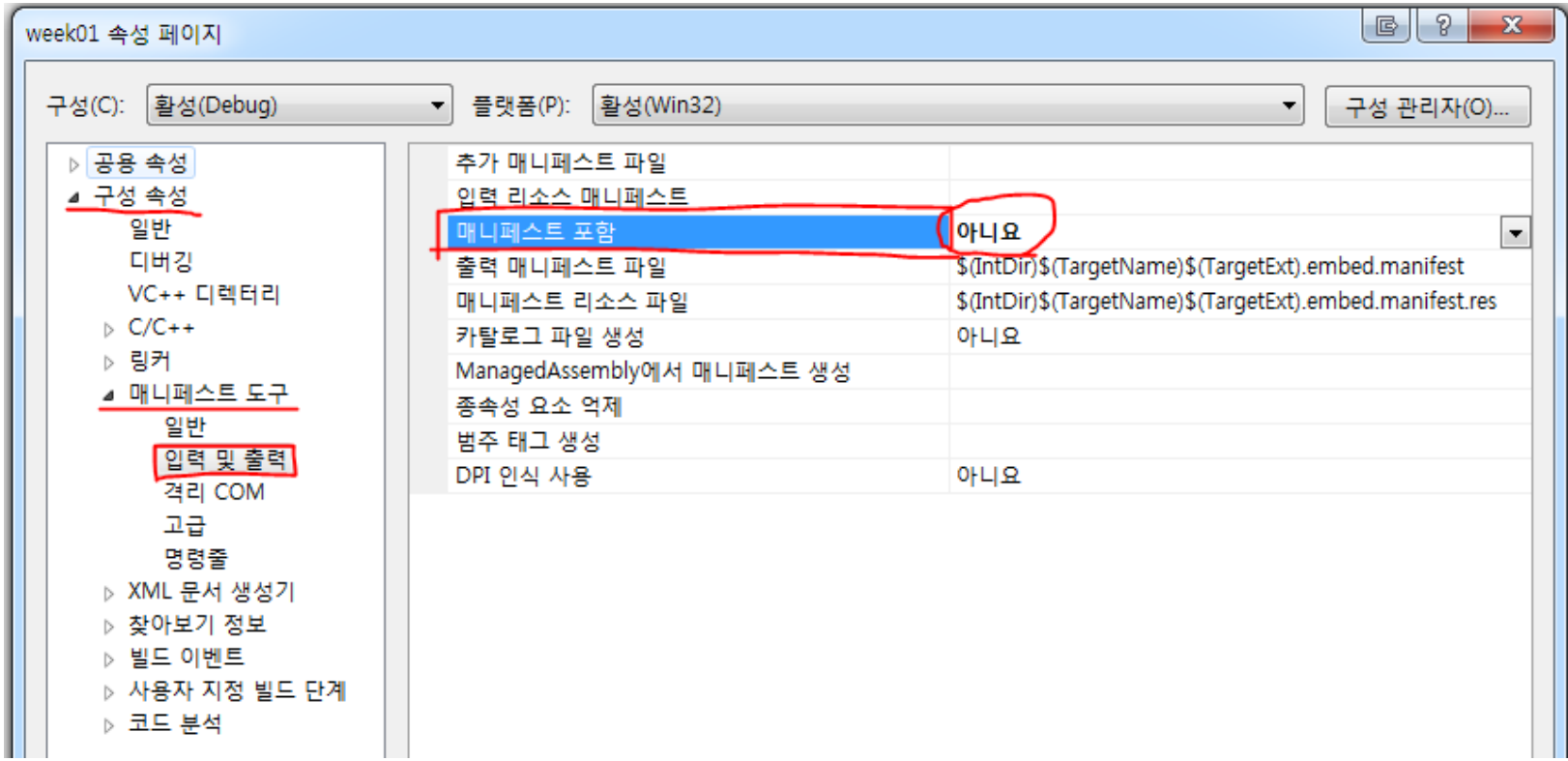


< 프로젝트 속성열기

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결

2.



구성 속성 -> 매니페스트 도구 -> 입력 및 출력 ->

메니페스트포함 : “아니요 "

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결
3. 매니페스트 문제 영구적 해결 방법

Visual Studio Service Pack 1 다운로드.
(>600MB 오래 걸림…)

https://www.microsoft.com/en-us/download/confirmation.aspx?id=23691