

# Tech University of Korea(TUK)

## 자료구조 과제 노트

202#-0#학기

담당교수	박정민
학번	•
이름	• 한##

## 과제수행 요청서

### 과제기간

#### ■ (과제기간) 과제는 교재의 각 장이 끝난 일주일 후, 24시까지 제출

- (제출방법) e-class의 ▲과제게시판에 업로드
- (제출파일명) [학번이름].hwp 예시) 01[2017131023박정민].hwp
- (지각제출불가) 과제를 제출하는 기간에 반드시 제출, ▲지각제출불가
- (개인제출) 과제제출은 ▲개인적으로 수행

### 과제내용

#### ■ (과제내용) 과제의 내용은 총 4가지 1)요점정리, 2)역공부, 3)순공부, 4)자기성찰

- (요점정리) 수업시간의 내용을 ▲재정리 ▲ 개별적으로 공부한 것이 있다면 추가 요점정리
- (역공부) 코드를 분석한 내용정리 ▲손으로 분석한 그림, ▲디버깅 SW를 이용한 분석그림
- (순공부) 역공학을 통해 분석한 소스코드를 근거로 ▲코드를 수정/개선시켜보기, ▲주석 상세히 달기
- (자기성찰) ▲수업을 통해서 배운 것, ▲디버깅을 통해 집중적으로 공부한 것

### 결과물활용

#### ■ (결과물) 결과물은 1)수시고사, 2)과제평가, 3) 개인 정리를 위한 지침

- (수시고사를 위한 참조) 과제들은 수시고사와 연계 ▲오픈북 시에 참조 문서로 활용
- (과제평가) 각 장별로 자료구조 수업이 종강될 때까지 작성하여 매주 평가
- (개인정리를 위한 지침) 과제내용 스스로 잘 정리하기 위한 좋은 가이드라인으로 활용

## 자료구조 과제 목차

### 1. 제1장 자료구조와 알고리즘

1-1 1장 자료구조와 알고리즘 요점정리

1-2 소스코드 디버깅 분석(역공부)

1-3 소스코드 수정/개선 & 주석(순공부)

### 2. 자기성찰

2-1 평가내용 및 느낀 점 (총 50점)

## 1. 제1장 자료구조와 알고리즘

### 1-1 1장 자료구조와 알고리즘 요점정리

#### ■ 자료구조와 알고리즘

- 프로그램은 자료구조와 알고리즘으로 구성되어있다.
- 알고리즘의 조건
  - 입력 : 0개 이상의 입력이 존재하여야 한다.
  - 출력 : 1개 이상의 출력이 존재하여야 한다.
  - 명백성 : 각 명령어의 의미는 모호하지 않고 명확해야 한다.
  - 유한성 : 한정된 수의 단계 후엔 반드시 종료되어야 한다.
  - 유효성 : 각 명령어들은 반드시 실행가능한 연산이어야 한다.
- 알고리즘
  - 기술방법
    - 자연어 (영어, 한국어) : 인간이 이해하기 쉽지만 의미가 모호해질 우려가 있다.
    - 순서도 : 직관적이고 이해하기 쉽지만 복잡한 알고리즘의 경우 상당히 복잡해진다.
    - 의사 코드 : 가장 많이 사용되고 알고리즘의 핵심적인 내용에만 집중할 수 있다.
    - 프로그래밍 언어 : 가장 정확하게 기술이 가능하지만 실제 구현 시, 구체적 내용들이 알고리즘의 핵심적인 내용을 방해할 수 있다.
- 추상 자료형(ADT)
  - 실제적인 구현으로부터 분리되어 정의된 자료형
  - 데이터나 연산이 무엇인지는 정의되지만 데이터나 연산을 어떻게 컴퓨터 상에서 구현할 것인지는 정의되지 않는다.
  - 객체 : 추상 데이터 타입에 속하는 객체
  - 연산 : 추상 데이터 타입과 외부를 연결하는 인터페이스 역할
- 알고리즘 성능분석
  - 수행 시간 측정 (직접)
    - 두 개의 알고리즘의 실제 수행 시간을 측정하는 것
    - 실제로 구현하는 것이 필요하고 동일한 하드웨어에서 수행해야 한다.
    - 알고리즘의 시작과 끝을 소스코드에서 측정하여 수행시간을 측정
  - 알고리즘의 복잡도 분석 (간접)
    - 직접 구현하지 않고 수행 시간을 측정하는 것
    - 알고리즘이 수행하는 연산의 횟수를 측정하여 비교
    - 연산의 횟수가 적을수록 더 효율적인 알고리즘이라고 할 수 있다. (시간복잡도 방법)
- 빅오 표기법
  - 연산의 횟수를 대략적으로 표기한 것
  - $T_{(n)} = n^2 + n + 1$  의 경우  $n^2$ 가 전체의 값을 주도함으로 이하의 항들은 고려대상이 아니다.

## 1-2 소스코드 디버깅 분석(역공부)

### ■ '손' 분석내용

<calc\_scores.c>

```
#define MAX_ELEMENTS 3;
int scores[MAX_ELEMENTS]
```

전역변수 scores, 함수 밖이 선언된 변수이므로  
각 배열에는 0으로 초기화되어있음

main

&score[2]	40	score[2]
&score[1]	20	score[1]
&score[0]	30	score[0]
&result	40	result

get\_max\_score

&largest	40
i	3

```
result = get_max_score
get_max_score 함수의 반환값 40이
result에 저장

return largest;
```

for 문 안

```
scores[1] > largest : 거짓
30 30
if문 수행 X

scores[2] > largest : 거짓
20 30
if문 수행 X

scores[3] > largest : 참
40 30
if문 수행

largest 값 40으로 변경
```

<calc\_time.c>

main

0 ... 10000	i
종료종료	stop
종료시작	start

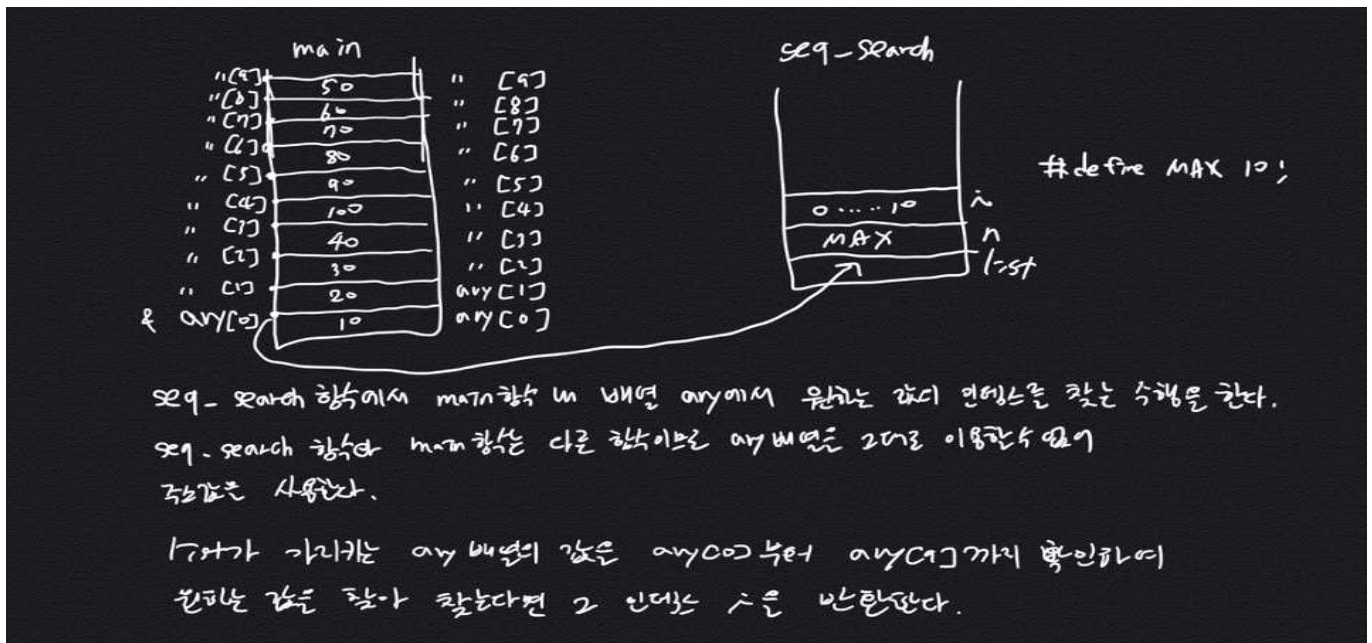
시작과 종료를 저장할 변수 start, stop 선언  
수행시간 저장에 위한 변수 duration 선언

for문에서 i가 0부터 10000까지를 반복하는  
수행시간 측정을 위한

for문 시작전 start = clock(); // 시작시간  
for문 종료시 stop = clock(); // 종료시간

종료시간에서 시작시간은 when clocks\_per\_sec(100)으로  
나눈 값은 duration이 되므로  
프로그래밍 수행시간은 직접적으로 구한다.

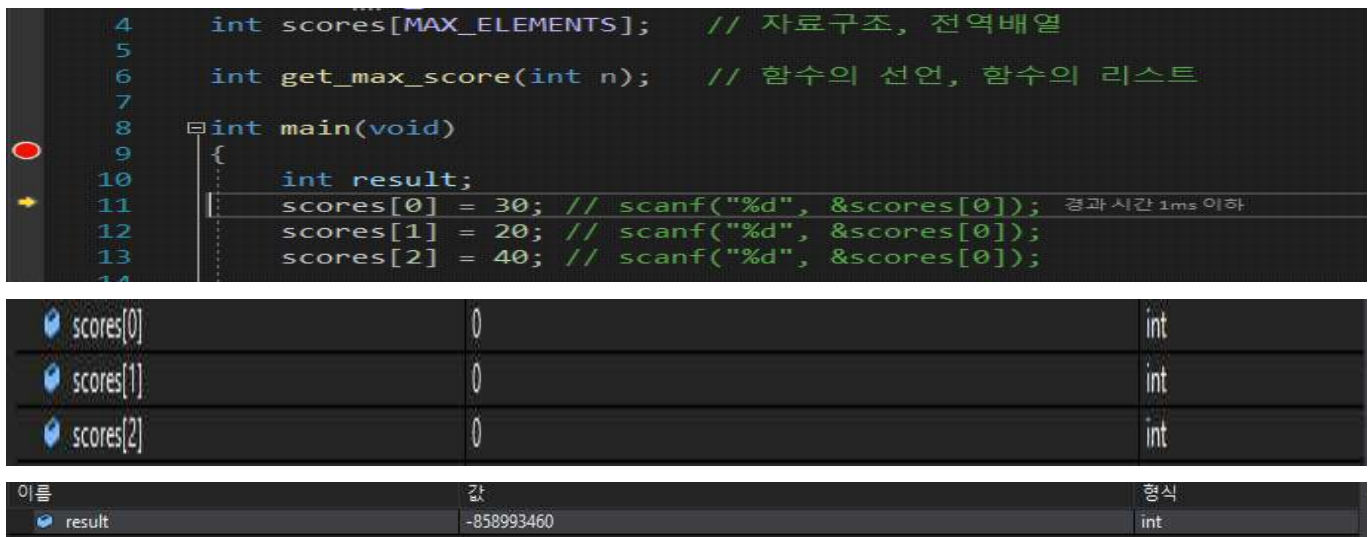
## <seq\_search.c>



seq-search 함수에서 main 함수의 배열 array에서 원하는 값의 인덱스를 찾는 수행을 한다.  
seq-search 함수와 main 함수는 다른 함수이므로 array 배열은 그대로 이용하지 않고 주소를 사용한다.  
하위가 가리키는 array 배열의 값은 array[0]부터 array[9]까지 확인하여 원하는 값을 찾아 찾았다면 그 인덱스 값을 반환한다.

## ■ 디버깅SW를 이용한 분석내용

### <cals.scores.c>



이름	값	형식
scores[0]	0	int
scores[1]	0	int
scores[2]	0	int
result	-858993460	int

- int scores[MAX\_ELEMENTS] 배열은 main 함수밖에 선언, 즉 전역변수이다.  
따라서 각 배열의 값들은 0으로 초기화되어 저장된다. result도 선언만 되어있는 상태이기에 쓰레기값이 들어있다.

이름	값	형식
scores[0]	30	int
scores[1]	20	int
scores[2]	40	int



```

11     scores[0] = 30; // scanf("%d", &scores[0]);
12     scores[1] = 20; // scanf("%d", &scores[0]);
13     scores[2] = 40; // scanf("%d", &scores[0]);
14
15     result = get_max_score(MAX_ELEMENTS); 경과 시간 1ms 이하

```

- scores 배열에 각각 30, 20, 40을 저장한다.

이름	값	형식
i	-858993460	int
largest	-858993460	int
n	3	int

```

21     int get_max_score(int n) // 학생의 숫자는 n
22     {
23         int i, largest;
24         largest = scores[0]; // 알고리즘 경과 시간 1ms 이하
25         for (i = 1; i < n; i++)
26

```

- get\_max\_score 함수가 호출되고 함수안에 i, largest 변수 선언되어 쓰레기값 저장  
n은 MAX\_ELEMENTS에 저장된 3이 호출된다.

이름	값	형식
i	1	int
largest	30	int
n	3	int

이름	값	형식
i	2	int
largest	40	int
n	3	int

```

26         {
27             if (scores[i] > largest)
28             {
29                 largest = scores[i];
30             }
31         }

```

- for문을 위해 i가 1로 초기화되고 n에 저장된 3보다 작을때까지 총 2번 반복된다.  
가장큰 값을 위해 scores[0]에 저장된 30을 일단 largest에 저장한다. i가 1일때 if문에 충족하지  
않아 for문이 끝나고 i=2일때 다시 실행한 결과 if문을 만족해서 largest값이 40으로 변경된다.

## <calc\_time.c>

```

6     int main(void)
7     {
8         clock_t start, stop;
9         double duration;
10        start = clock(); // 측정 시작 경과 시간 1ms 이하
11        for (int i = 0; i < MAX; i++) // 이미 어느 바보 르표.

```

이름	값	형식
duration	-9.2559631349317831e+61	double
start	-858993460	long
stop	-858993460	long

- main함수에서 수행시간 측정을 위한 측정시작과 종료를 저장할 변수 start, stop을 저장  
그 수행시간을 저장하 변수 duration을 선언한다. 그 변수들에는 당연히 쓰레기값들이 저장된다.

```

10  start = clock(); // 측정 시작
11  for (int i = 0; i < MAX; i++) // 의미 없는 반복 루프; 경과 시간 1ms 이하

```

start	57432	long
stop	-858993460	long

- for문을 작동될 수행시간을 측정하는 것이기 때문에 for문 직전에 측정 시작을 의미하는  
코드를 넣는다. start에는 그 값이 들어가 있음을 알 수 있다.

```

13  stop = clock(); // 측정 종료
14  duration = (double)(stop - start) / CLOCKS_PER_SEC; 경과 시간 1ms 이하
15  printf("수행시간은 %.3lf초입니다.\n", duration);
16  return 0;

```

start	57432	long
stop	138692	long

- for문 종료 직후 측정 종료를 의미하는 코드를 집어 넣어 측정종료를 알린다.

```

14  duration = (double)(stop - start) / CLOCKS_PER_SEC;
15  printf("수행시간은 %.3lf초입니다.\n", duration); 경과 시간 1ms 이하
16  return 0;
17

```

duration	81.260000000000005	double
start	57432	long
stop	138692	long

- 종료와 시작의 차를 구한후 CLOCKS\_PER\_SEC로 나눈다. duration은 double 형이고 start와  
stop은 long 형이기 때문에 이과정에선 강제 형변환이 필요하다. 나온 값을 duration에 저장함으로  
프로그램의 수행시간을 계산할 수 있다.



## <calc\_time.c>

```
4 {
5     int ary1[MAX] = { 10,20,30,40,100,90,80,70,60,50 }; // 배열의 이름은 배열의 크기
```

이름	값	형식
ary1	0x008ff9dc {10, 20, 30, 40, 100, 90, 80, 70, 60, 50}	int[10]

- main함수에 정수형 배열 ary1을 선언하고 각 배열에 값을 입력해준다.

```
6     int result = seq_search(ary1, MAX, 90); 경과시간 1ms 이하
```

```
11 int seq_search(int* list, int n, int key) // int* list = ary, int n = 10, int key = 71
12 // 입력 : 3개, 처리 : 순차적으로 data를 찾는다, 출력 : int 정수
13 {
    경과시간 1ms 이하
```

- main함수에서 seq\_search함수가 호출되어 seq\_search함수로 이동한다.  
이때 seq\_search함수의 매개변수는 int\* list, int n, int key이다. int\* list인 이유는 활용해야하는 배열 ary1배열은 main함수에 선언된 배열이기 때문에 seq\_search함수에서 직접참조할 수 없어 배열의 주소를 이용하는 간접참조 방식을 사용해야하기 때문이다. n은 배열의 크기 MAX를 의미하고 key는 배열에서 찾으려는 값을 의미한다.

```
15     for (i = 0; i < n; i++) // i는 0~9
16     {
17         if (list[i] == key) 경과시간 1ms 이하
18             return i; /* 탐색에 성공하면 키 값의 인덱스 반환 */
19     }
```

- 90을 찾기 위해 for문을 이용하여 list가 가리키는 ary1배열 [0]부터 [9]까지를 if문으로 찾고 90과 같다면 그 배열의 인덱스값을 반환하는 for문이다.

이름	값	형식
i	5	int
key	90	int
list	0x008ff9dc {10}	int*
n	10	int

```
5     int ary1[MAX] = { 10,20,30,40,100,90,80,70,60,50 }; // 배열의 이름은 배열의 크기
6     int result = seq_search(ary1, MAX, 90);
7     printf("%d\n", result);
8     return 0; 경과시간 1ms 이하
9 }
```

- ary[5]에 90이 key의 값과 일치 하기에 5가 seq\_search함수에서 반환되고 그 return 값이 main함수의 result에 저장되어 출력되는 모습이다.

## 1-3 소스코드 수정/개선 & 주석(순공부)

### ■ 수정/개선 & 주석1 (수정/개선된 코드에 주석을 달고 & 설명)

```

8   int main(void)
9   {
10      int result,i;
11      /*15열부터 17열까지의 score를 입력하는 부분을
12      for문을 이용해 표현가능해 횟수를 제어할 변수 i를 추가하여 표현*/
13      for (i = 0; i < MAX_ELEMENTS; i++)
14      {
15          scanf_s("%d", &scores[i]);
16      }
17      /*scores[0] = 30; // scanf("%d", &scores[0]);
18      scores[1] = 20; // scanf("%d", &scores[0]);
19      scores[2] = 40; // scanf("%d", &scores[0]);
20      */

```

### ■ 수정/개선 & 주석2

```

1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <time.h> // index찾는 시간측정을 위한 헤더파일
4   #define MAX 10
5
6   int main()
7   {
8       int ary1[MAX]; // 배열의 이름은 배열의 시작소
9       int i, search;
10
11       //수행시간 측정을 위한 변수 선언
12       clock_t start, stop;
13       double duration;
14
15       //ary1 배열을 직접 정할 수 있는 for문을 사용
16       printf("ary1배열의 값을 입력해주세요\n");
17       for (i = 0; i < MAX; i++)
18       {
19           scanf_s("%d", &ary1[i]);
20       }
21
22       //배열에서 찾을 값을 scanf함수를 이용해서 사용자가 직접 받는다
23       printf("배열에서 찾을 값을 입력해주세요 : ");
24       scanf_s("%d", &search);
25
26       start = clock(); // 측정 시작
27       int result = seq_search(ary1, MAX, search);
28       stop = clock(); // 측정 완료
29       duration = (double)(stop - start) / CLOCKS_PER_SEC; // 수행시간 계산 값 저장
30       printf("찾는 값의 index는 %d입니다.\n", result);
31       printf("찾는데 걸린 시간은 %lf 초입니다.\n");
32       return 0;
33   }
34
35   int seq_search(int* list, int n, int key) // int* list = ary, int n = 10, int key = 71
36   // 입력 : 3개, 처리 : 순차적으로 data를 찾는다, 출력 : int 정수
37   {
38       int i;
39       for (i = 0; i < n; i++) // i는 0~9
40       {
41           if (list[i] == key)
42               return i; /* 탐색에 성공하면 키 값의 인덱스 반환 */
43       }
44       return -1; /* 탐색에 실패하면 -1 반환 */
45   }
46   // ary1 배열은 seq_search 함수에선 사용할 수 없음
47   // 시간측정도 해보기

```

## 2. 자기성찰

### 2-1 평가내용 및 느낀 점 (총 50점)

평가 내용 (30점 만점)	3점	2점	1점
1. 온라인강의는 주어진 기한 내 듣기 완료를 하였나?	●		
2. 이번 강의 수업은 이해도가 높았나?	●		
3. 역공부, 주어진 소스코드 분석을 위해 손으로 그림을 그려보았나?	●		
4. 역공부, 주어진 소스코드 분석을 위해 디버깅 SW로 분석해보았나?	●		
5. 역공부, 소스코드를 구현한 개발자의 노력을 생각해보았나?	●		
6. 순공부, 3번과 4번을 수행하면서 코드를 개선시킬 노력은 하였나?	●		
7. 순공부, 개선코드 주석은 꼼꼼히 작성하였나?	●		
8. 과제 '양식(hwp)'을 준수하여 작성하였나?	●		
9. 오타 없이 표의 양식, 그림의 넓이&높이 등 정돈된 보고서인가?	●		
10. 시간 마감에 급급한 대충 작성한 보고서가 아닌, 열심히 공부한 정직한 보고서인가?	●		
총 점	30 점		

‘객관식 30점, 주관식 20점, 총 50점을 스스로 자기성찰’ 하고, 정직한 보고서를 위해 최선을 다하자!

노력지수 (10점 만점)	<ul style="list-style-type: none"> <li>• 9점 <ul style="list-style-type: none"> <li>- 첫 시간의 수업이라 내용적으로 크게 이해되지 않는 부분이 없었지만 오랜만에 학교 수업을 듣는만큼 최선을 다하려고 노력하였다. 특히 방학동안 약간 소홀했던 프로그래밍 감각을 올릴 수 있게 되었다.</li> </ul> </li> </ul>
느낀 점 (10점 만점)	<ul style="list-style-type: none"> <li>• 10점 <ul style="list-style-type: none"> <li>- 기본적인 틀은 지난 학기 들었던 프로그래밍과 크게 다르지 않았지만 알고리즘과 추상 자료형 등 새로운 내용을 접목할 수 있는 방법에 대해 새롭게 알아가는 것 같아 흥미가 생기기 시작했다. 특히 손으로 그리고 직접 sw를 통해 디버깅하는 과정을 통해 완벽하게 체득할 수 있었다.</li> </ul> </li> </ul>

- ‘자기성찰’ 부분은 과제 채점 시, 많은 부분 참고가 됩니다. -