Tech University of Korea(TUK)
자료구조 과제 노트

202#-0#학기

담당교수	박정민
학번	•
이름	• 황##

한국공학대학교
컴퓨터공학부 (자료구조) 박정민 교수

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

■ # 공부하는 방법

- 1.타이밍 하지 마라-> 예제소스 단순히 실행->결과
- 2.디버깅(2개)
- - 1.손->2.그림->수동

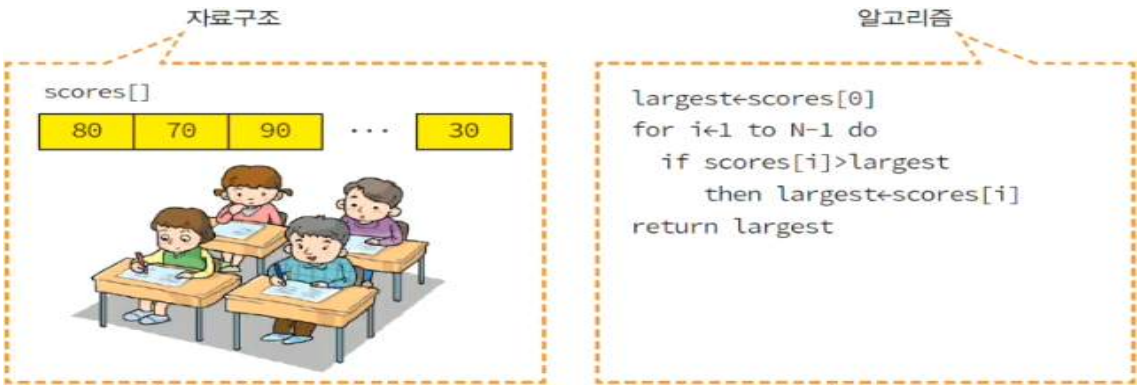
■ 일상생활과 자료구조의 비교

〈표 1-1〉 일상생활과 자료구조의 유사성

일상생활에서의 예	해당하는 자료구조
그릇을 쌓아서 보관하는 것	스택
마트 계산대의 줄	큐
버킷 리스트	리스트
영어사전	사전
지도	그래프
컴퓨터의 디렉토리 구조	트리

■ 자료구조와 알고리즘

- 프로그램= 자료구조 + 알고리즘ㅍ



■ 알고리즘(algorithm): 컴퓨터로 문제를 풀기 위한 단계적인 절차

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘: 컴퓨터로 문제를 풀기 위한 단계적인 절차

-
-

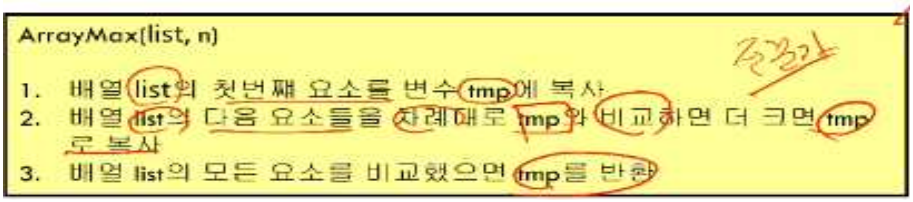
■ 알고리즘의 기술 방법

- 영어나 한국어와 같은 자연어
- 순서도(flow chart)
- 유사 코드(pseudo-code)
- C와 같은 프로그래밍 언어
-

■ 자연어로 표기된 알고리즘

- 인간이 읽기가 쉬움
- 그러나 자연어의 단어들을 정확하게 정의하지 않으면 의미 전달이 모호해질 우려가 있음

(예) 배열에서 최대값 찾기 알고리즘

- 

-

■ 흐름도로 표기된 알고리즘

- 직관적이고 이해하기 쉬운 알고리즘 기술 방법
- 그러나 복잡한 알고리즘의 경우,
- 상당히 복잡해짐
-

■ 유사코드로 표현된 알고리즘

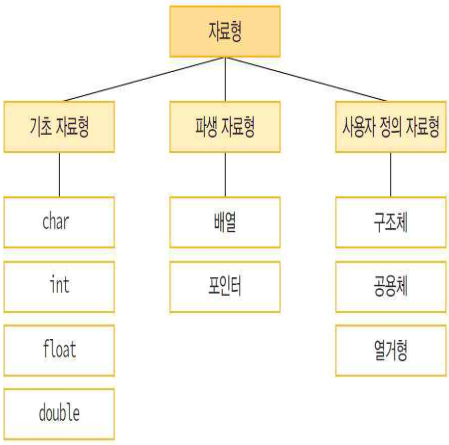
- 알고리즘 기술에 가장 많이 사용
- 프로그램을 구현할 때의 여러 가지 문제들을 감출 수 있다.
- 즉 알고리즘의 핵심적인 내용에만 집중할 수 있다.
-

■ C로 표현된 알고리즘

- 알고리즘의 가장 정확한 기술이 가능
- 반면 실제 구현 시, 많은 구체적인 사항들이 알고리즘의 핵심적인
- 내용에 대한 이해를 방해할 수 있다.

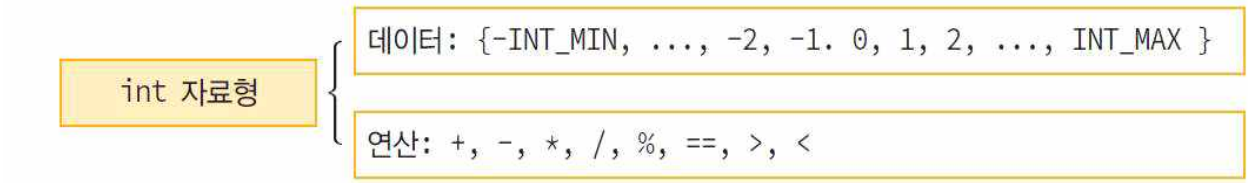
-

■ 자료형

- 자료형(data type):"데이터의 종류"
- 정수,실수,문자열 등이 기초적인 자료형의 예
- 

-

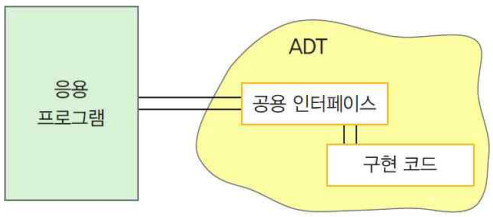
■ 자료형(data type)

- 데이터의 집합과 연산의 집합
- 

■ 추상 데이터 타입

- 데이터 타입을 추상적(수학적)으로 정의한 것
- 데이터나 연산이 무엇(what)인가는 정의되지만 데이터나 연산을 어떻게(how) 컴퓨터 상에서 구현할 것인지는 정의되지 않는다.
-

■ 추상데이터타입의 유래

- 추상화->정보은닉기법->추상 자료형
- 추상화란 사용자에게 중요한 정보는 강조되고 반면 중요하지 않은 구현 세부 사항은 제거하는 것
- 

■ 추상 데이터 타입의 정의

- 객체: 추상 데이터 타입에 속하는 객체가 정의된다.
- 연산: 이들 객체들 사이의 연산이 정의된다. 이 연산은 추상 데이터 타입과 외부를 연결하는 인터페이스의 역할을 한다.

■ 알고리즘의 성능 분석

■ 알고리즘의 성능 분석 기법

- **수행 시간 측정**(직접 수행)
- - 두 개의 알고리즘의 실제 수행 시간을 측정하는 것
- - 실제로 구현하는 것이 필요
- - **동일한 하드웨어**를 사용하여야 함(시험문제 출제)
-

■ 알고리즘의 복잡도 분석 (빅오)

- 직접 구현하지 않고서도 수행 시간을 분석하는 것
- 알고리즘이 수행하는 연산의 횟수를 측정하여 비교
- 일반적으로 연산의 횟수는 n의 함수
-

■ 왜 프로그램의 효율성이 중요한가?

입력 자료의 개수	프로그램 A: n^2	프로그램 B: 2^n
$n = 6$	36초	64초
$n = 100$	10000초	2^{100}초=4×10^{22}년

-

■ 수행시간측정

- 알고리즘을 프로그래밍 언어로 작성하여 실제 컴퓨터상에서 실행시킨 다음, 그 수행시간을 측정
- 조건) 1. 데이터의 개수(n)이 같아야 한다.
- 2. 하드웨어가 같아야 한다.
- 상세내용 2

■ 복잡도 분석

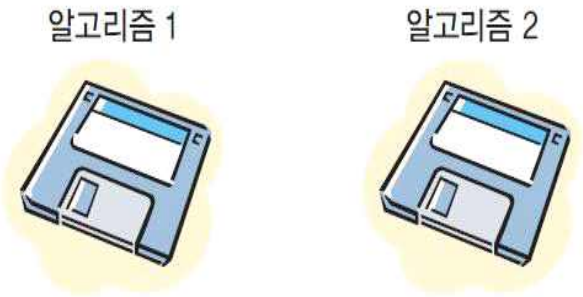
- 시간 복잡도는 알고리즘을 이루고 있는 연산들이 몇 번이나 수행되는지를 숫자로 표시

```
largest←scores[0]
for i←1 to N-1 do
    if scores[i]>largest
        then largest←scores[i]
return largest
```

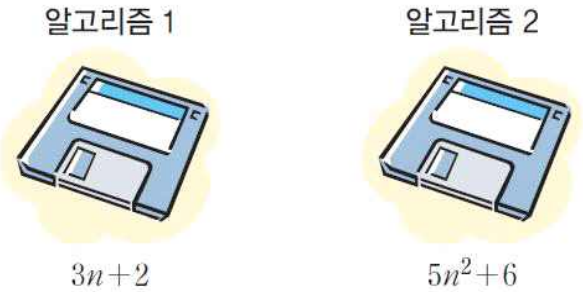
■ 복잡도 분석의 종류

- 직접 측정

- - 시간 측정
- 간접 측정
- - 자료 개수 n개로 고정
- - 연산의 횟수를 계산
- 시간 복잡도(time complexity)
- 공간 복잡도(space complexity)



■ 입력의 개수 고려



■ 복잡도 분석의 예

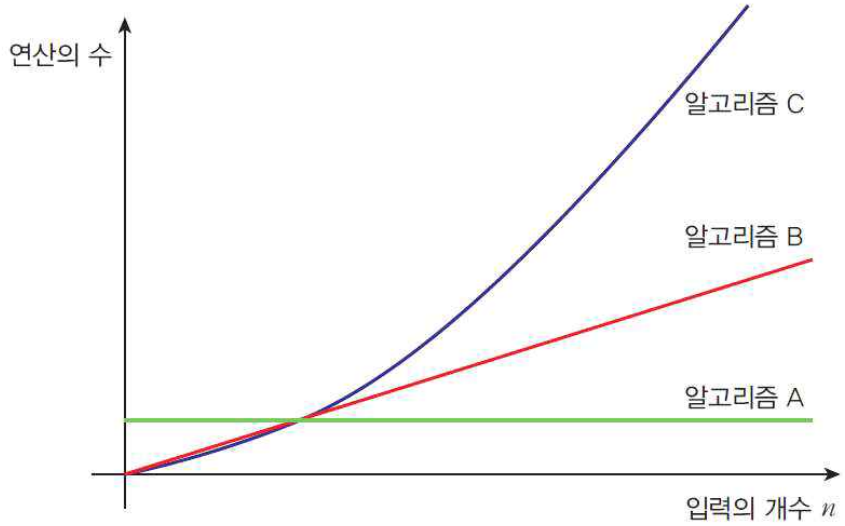
- - 양의 정수을 n번 더하는 문제를 생각해보자

알고리즘 A	알고리즘 B	알고리즘 C

sum ←n*n;	for i←1 to n do
	sum ←sum + n;
		for i←1 to n do
		for j←1 to n do
		sum ←sum + 1;

	알고리즘 A	알고리즘 B	알고리즘 C
대입연산	1	n	$n*n$
덧셈연산		n	$n*n$
곱셈연산	1		
나눗셈연산			
전체연산수	2	$2n$	$2n^2$

- O(1) O(n) O(n^2)

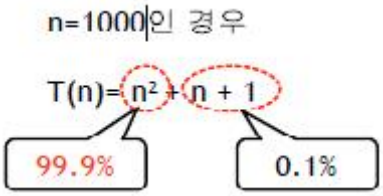


■ 복습!

- - 직접 측정: time을 재는 것.
- - 간접 측정: 연산의 횟수를 재는 것.

■ 빅오 표기법

- - 자료의 개수가 많은 경우에는 차수가 가장 큰 항이 가장 큰 항이 가장 영향을 크게 미치고 다른 항들은 상대적으로 무시될 수 있다.



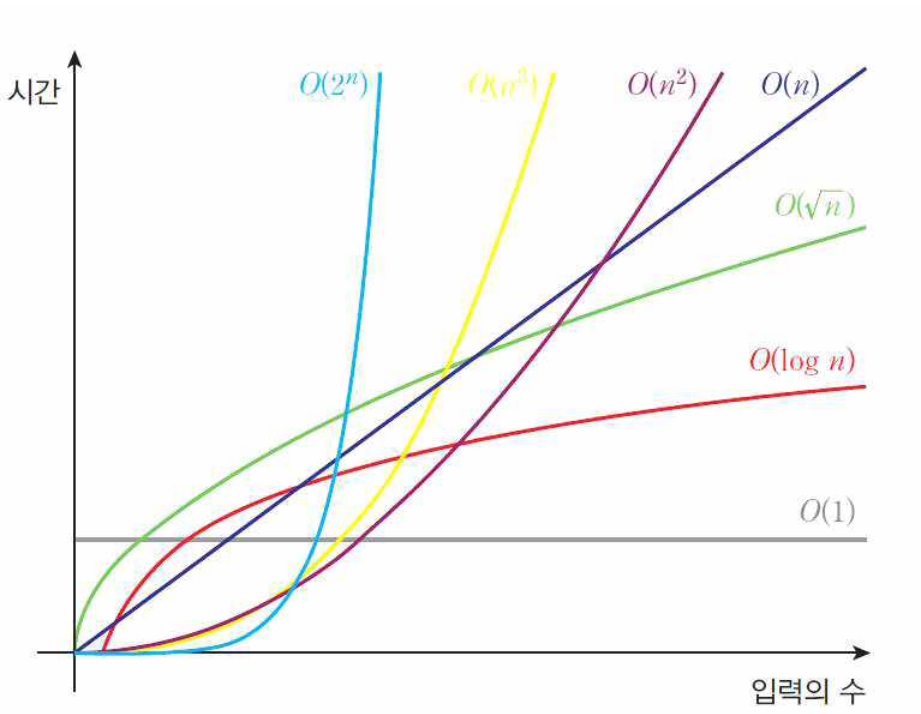
- 23p~29p 예제 반드시 공부하자!

■ 빅오 표기법의 종류

- $O(1)$: 상수형
- $O(\log n)$: 로그형
- $O(n)$: 선형
- $O(n \log n)$: 선형로그형
- $O(n^2)$: 2차형
- $O(n^3)$: 3차형
- $O(2^n)$: 지수형
- $O(n!)$: 팩토리얼형

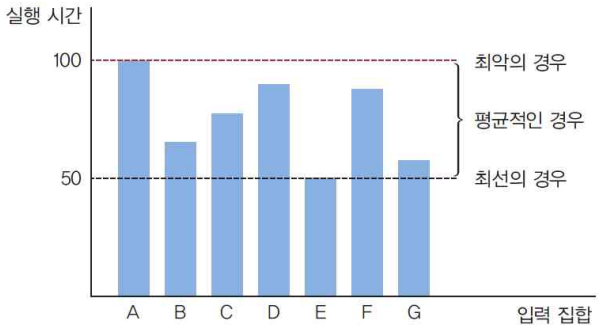
$f(n)$	$O(f(n))$
10	$O(1)$
$5n^2+6$	$O(n^2)$
$2n^3+1$	$O(n^3)$
$2n^3+5n^2+6$	$O(n^3)$

시간복잡도	n					
	1	2	4	8	16	32
1	1	1	1	1	1	1
$\log n$	0	1	2	3	4	5
n	1	2	4	8	16	32
$n\log n$	0	2	8	24	64	160
n^2	1	4	16	64	256	1024
n^3	1	8	64	512	4096	32768
2^n	2	4	16	256	65536	4294967296
$n!$	1	2	24	40326	20922789888000	26313×10^{33}



■ 최선, 평균, 최악의 경우

- - 알고리즘의 수행시간은 입력 자료(n) 집합에 따라 다를 수 있다.
- - 최선의 경우(best case): 수행 시간이 가장 빠른 경우
- - 평균의 경우(average case): 수행시간이 평균적인 경우
- - 최악의 경우(worst case): 수행 시간이 가장 늦은 겨우



■ (예) 최선, 평균, 최악의 경우

■ (예) 순차탐색

- - 최선의 경우: 찾고자 하는 숫자가 맨 앞에 있는 경우 : O(1)
- 

- - 최악의 경우: 찾고자 하는 숫자가 맨 뒤에 있는 경우 : O(n)
- 

- - 평균적인 경우: 각 요소들이 균일하게 탐색된다고 가정하면 : (1+2+...+n)/n=(n+1)/2
- : O(n)
- 

■ 프로그램 1.3

- - 최선의 경우: 의미가 없는 경우가 많다.
- - 평균적인 경우: 계산하기가 상당히 어려움
- - 최악의 경우: 가장 널리 사용된다. 계산하기 쉽고 응요에 따라서 중요한 의미를 가질 수도 있다. (예) 비행기 관제업무, 게임, 로보틱스

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함

- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함

■ 알고리즘의 조건

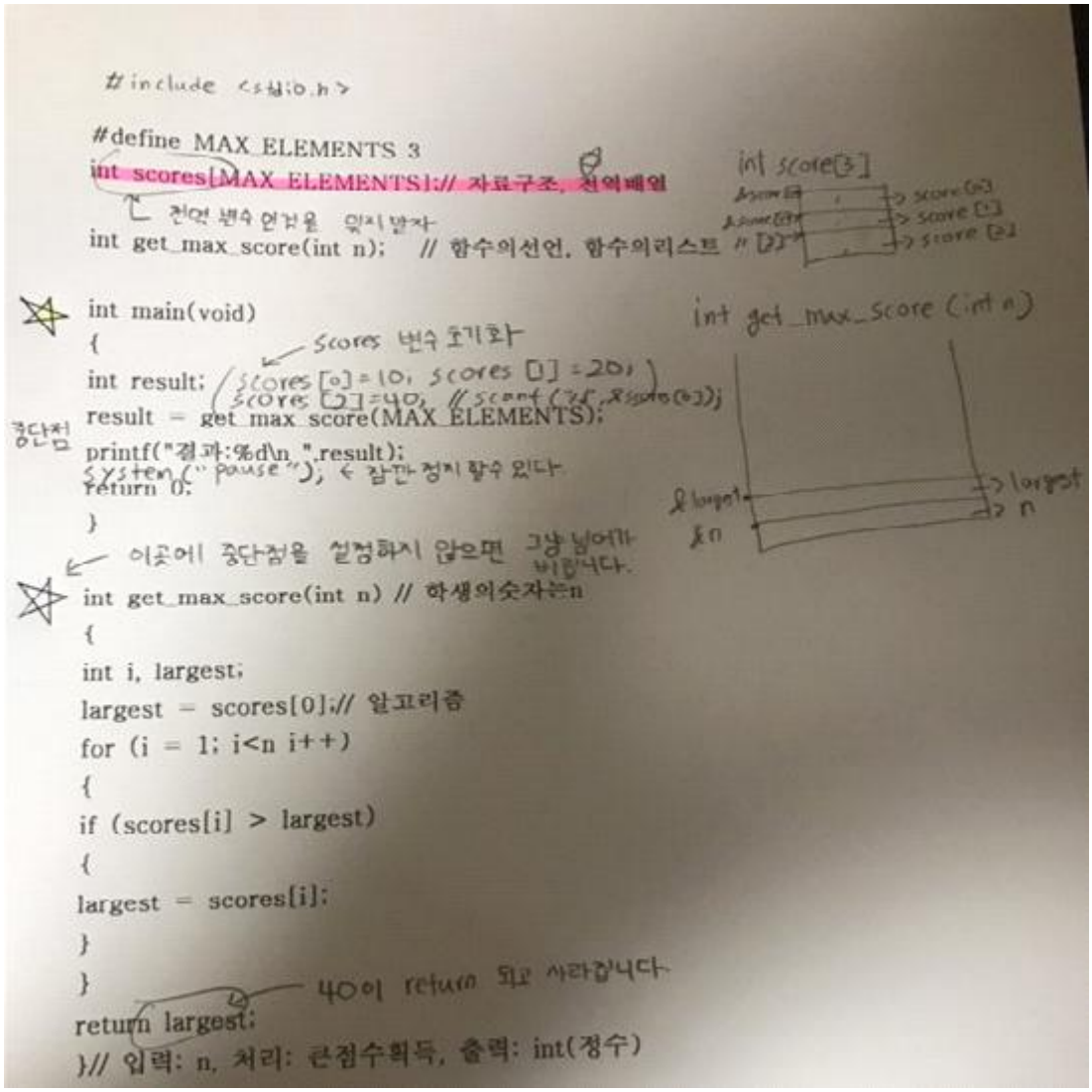
- - 입 력 : 입력이 존재하여야 함
- - 출 력 : 출력이 존재하여야 함
-

1-2 소스코드 디버깅 분석(역공부)

■ '손' 분석내용

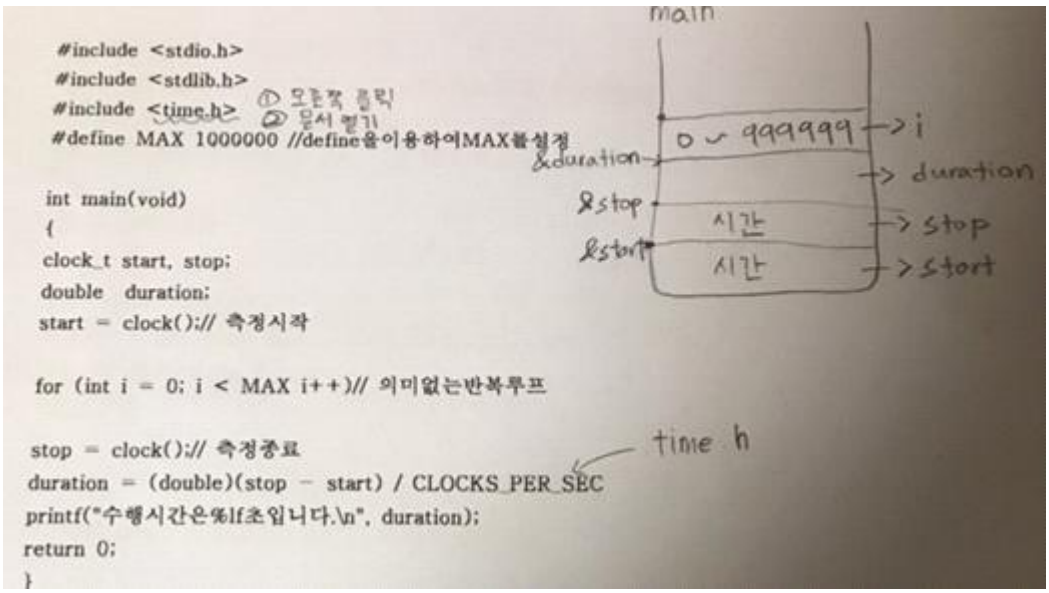
-

■ 01 calc_scores.c



```
#include <stdio.h>

#define MAX_ELEMENTS 3
int scores[MAX_ELEMENTS];// 자료구조, 전역배열
int get_max_score(int n);   // 함수의선언, 함수의리스트

int main(void)
{
int result;
result = get_max_score(MAX_ELEMENTS);
printf("결과:%d\n",result);
system("pause");
return 0;
}

int get_max_score(int n) // 학생의숫자는n
{
int i, largest;
largest = scores[0];// 알고리즘
for (i = 1; i<n i++)
{
if (scores[i] > largest)
{
largest = scores[i];
}
}
return largest;
}// 입력: n, 처리: 큰점수획득, 출력: int(정수)
```

■ 02 calc_time.c

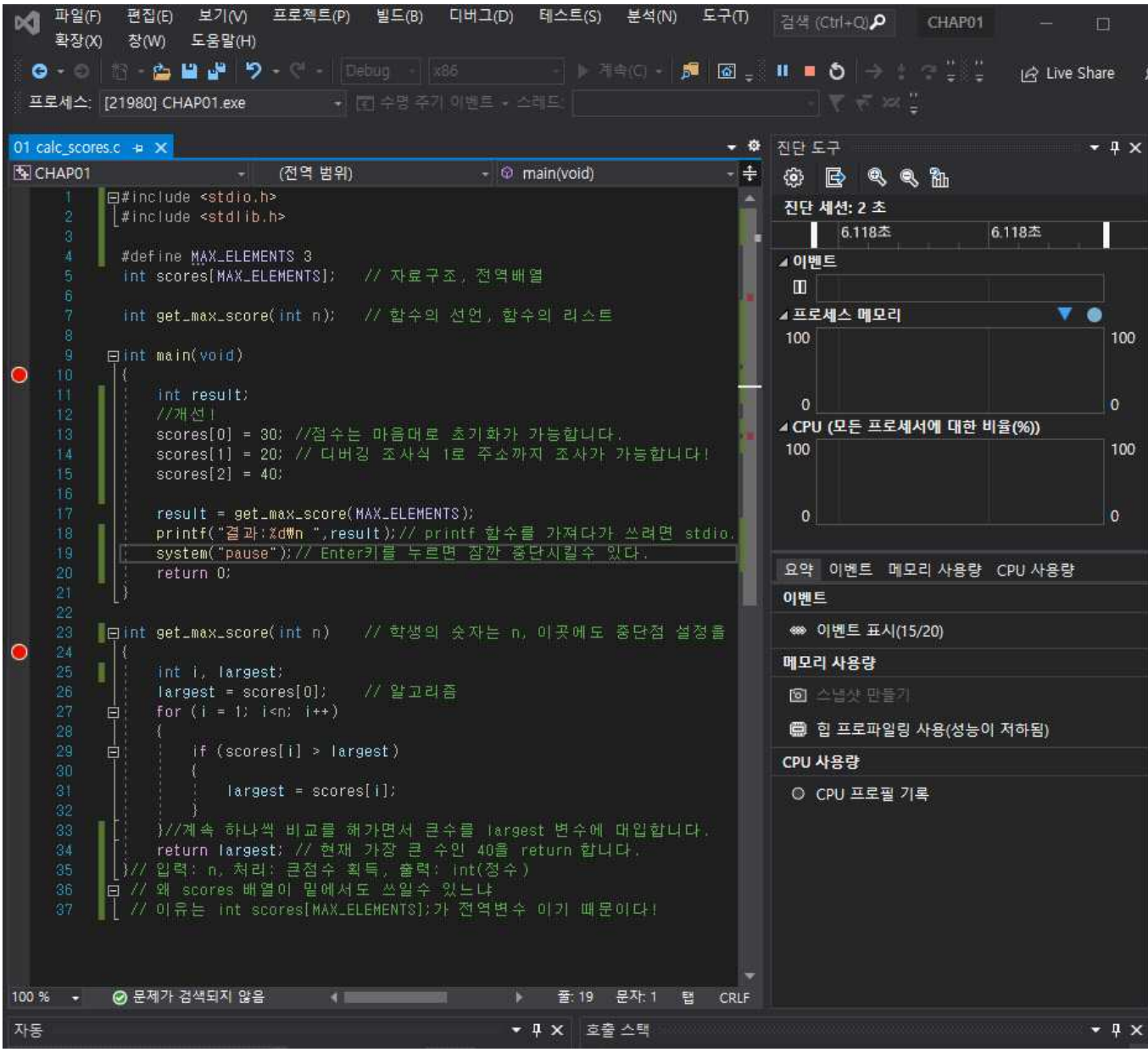


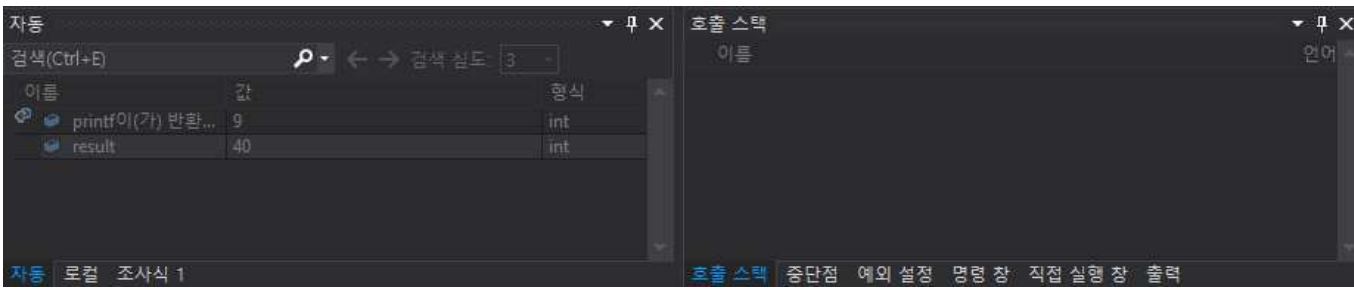

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 1000000 //define을이용하여MAX를설정

int main(void)
{
clock_t start, stop;
double  duration;
start = clock();// 측정시작

for (int i = 0; i < MAX i++)// 의미없는반복루프

stop = clock();// 측정종료
duration = (double)(stop - start) / CLOCKS_PER_SEC
printf("수행시간은%lf초입니다.\n", duration);
return 0;
}
```

■ 03 seq_search.c

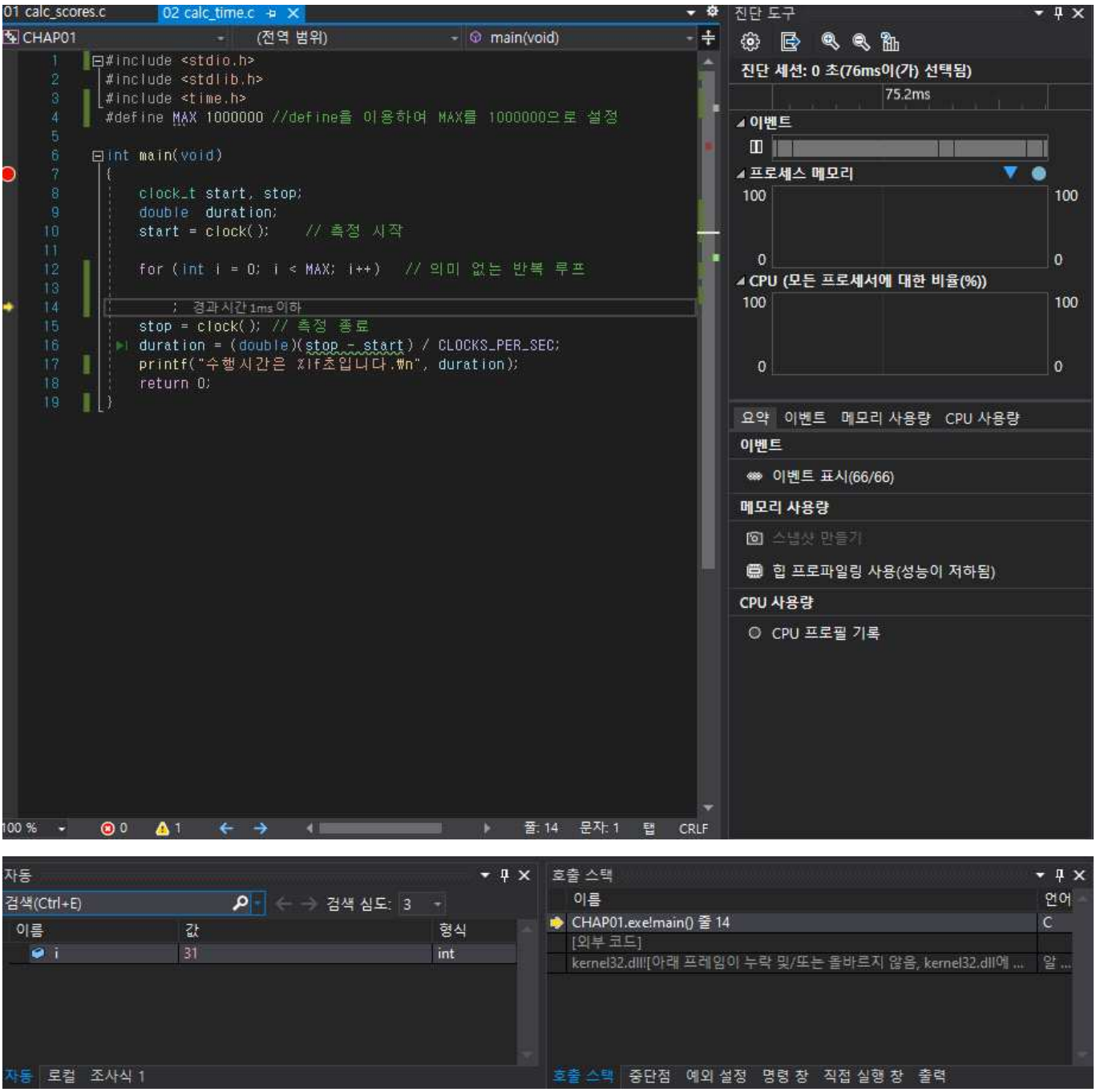
```
#include <stdio.h>
#define MAX 10

int main(void)
{
int search;
int result;
int array1[MAX] = {23,32,42,35,216,23,62,33,15,6}; //배열의이름은배열의시작주소
scanf_s("%d", &search); //scanf를이용해서좀더개선된코드를이용할수있습니다.
//start = clock();
result=seq_search(array1, MAX, search);
//stop = clock();
printf("결과: %d\n", result);
return 0;

}
int seq_search(int* list, int n, int key) // 순차탐색
// list는주소를저장하는포인터변수: 지금은array1을포인팅하고있다
// int* list=array; int n=10; int key=15
// 입력: 3개, 처리: 순차적으로data를찾는다. 출력: int 정수로한다
{
int i; // i는0~9까지의인덱스번호
for (i = 0; i < n i++)
if (list[i] == key)
return i;  /* 탐색에성공하면키값의인덱스반환*/
return -1;    /* 탐색에실패하면-1 반환*/
}
```

■ 디버깅SW를 이용한 분석내용

- 
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 테스트(S) 분석(N) 도구(T)
확장(X) 창(W) 도움말(H)
검색 (Ctrl+Q)
CHAP01
Debug
x86
Live Share
프로세스: [21980] CHAP01.exe
01 calc_scores.c
CHAP01
(전역 범위)
main(void)
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENTS 3
int scores[MAX_ELEMENTS]; // 자료구조, 전역배열
int get_max_score(int n); // 함수의 선언, 함수의 리스트
int main(void)
{
int result;
//개선!
scores[0] = 30; //점수는 마음대로 초기화가 가능합니다.
scores[1] = 20; // 디버깅 조사식 1로 주소까지 조사가 가능합니다!
scores[2] = 40;
result = get_max_score(MAX_ELEMENTS);
printf("결과:%d₩n ",result);// printf 함수를 가져다가 쓰려면 stdio.
system("pause");// Enter키를 누르면 잠깐 중단시킬수 있다.
return 0;
}
int get_max_score(int n) // 학생의 숫자는 n, 이곳에도 중단점 설정을
{
int i, largest;
largest = scores[0]; // 알고리즘
for (i = 1; i<n; i++)
{
if (scores[i] > largest)
{
largest = scores[i];
}
}//계속 하나씩 비교를 해가면서 큰수를 largest 변수에 대입합니다.
return largest; // 현재 가장 큰 수인 40을 return 합니다.
}// 입력: n, 처리: 큰점수 획득, 출력: int(정수)
// 왜 scores 배열이 밑에서도 쓰일수 있느냐
// 이유는 int scores[MAX_ELEMENTS];가 전역변수 이기 때문이다!
진단 도구
진단 세션: 2 초
6.118초
6.118초
이벤트
프로세스 메모리
100
0
CPU (모든 프로세서에 대한 비율(%))
요약 이벤트 메모리 사용량 CPU 사용량
이벤트 표시(15/20)
스냅샷 만들기
힙 프로파일링 사용(성능이 저하됨)
CPU 사용량
CPU 프로필 기록
100 %
문제가 검색되지 않음
줄: 19 문자: 1 탭 CRLF
자동
호출 스택

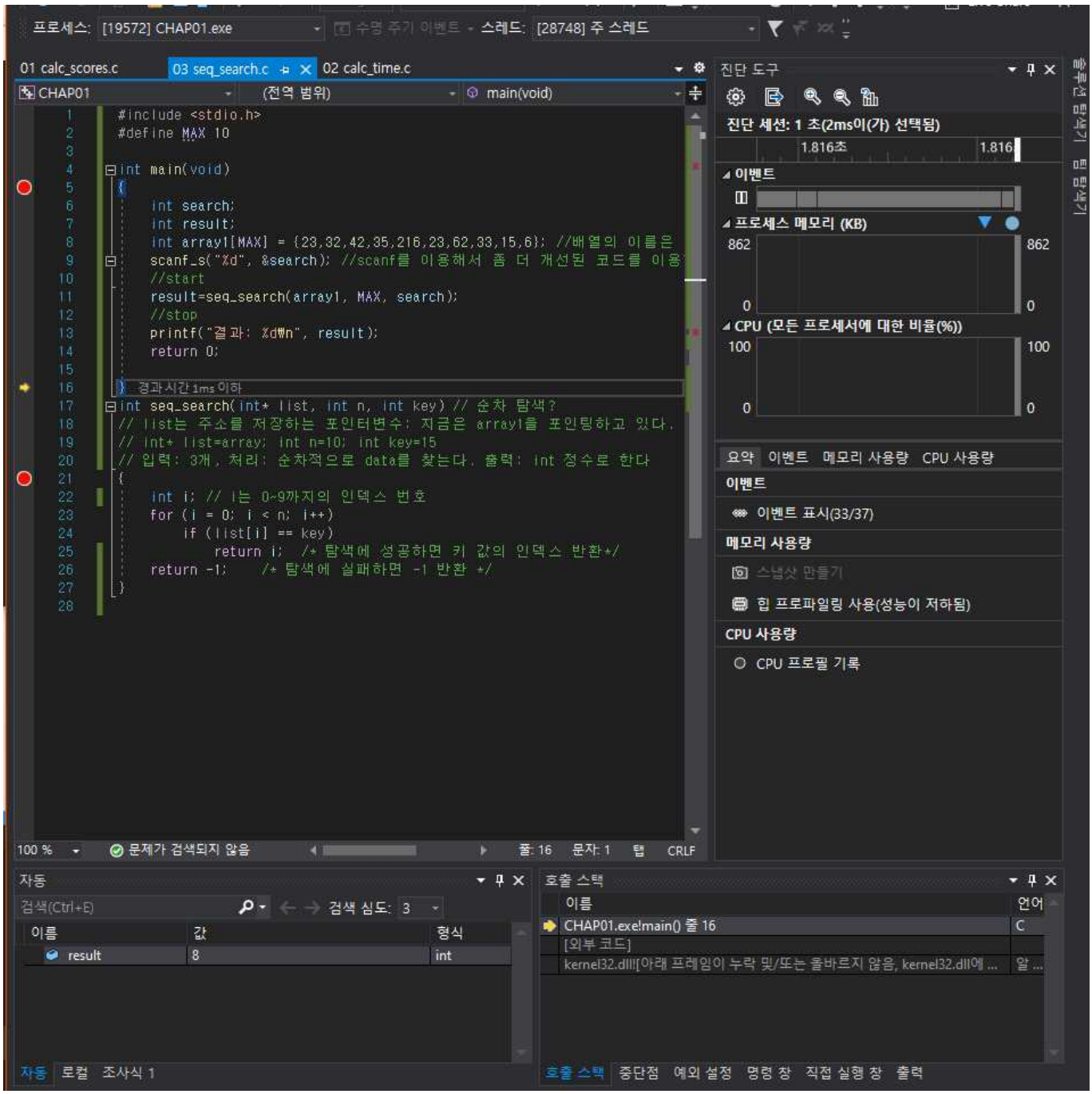
- 
자동
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
printf이(가) 반환... 9 int
result 40 int
자동 로컬 조사식 1
호출 스택
이름
언어
호출 스택 중단점 예외 설정 명령 창 직접 실행 창 출력

■ 02 calc_time.c



clock_t = long double 타입이다!

■ 03 seq_search.c



1-3 소스코드 수정/개선 & 주석(순공부)

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

■ 01 calc_scores.c

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

#define MAX_ELEMENTS 3
int scores[MAX_ELEMENTS];   // 자료구조, 전역배열

int get_max_score(int n);   // 함수의 선언, 함수의 리스트

int main(void)
{
    int result;
    //개선!
    scores[0] = 30; //점수는 마음대로 초기화가 가능합니다.
    scores[1] = 20; // 디버깅 조사식 1로 주소까지 조사가 가능합니다!
    scores[2] = 40;

    result = get_max_score(MAX_ELEMENTS);
    printf("결과:%d₩n ",result);// printf 함수를 가져다가 쓰려면 stdio.h가 필요하다.
    system("pause");// Enter키를 누르면 잠깐 중단시킬수 있다.
    return 0;
}

int get_max_score(int n)    // 학생의 숫자는 n, 이곳에도 중단점 설정을 꼭 해놓읍시다!
{
    int i, largest;
    largest = scores[0];    // 알고리즘
    for (i = 1; i<n; i++)
    {
        if (scores[i] > largest)
        {
            largest = scores[i];
        }
    }//계속 하나씩 비교를 해가면서 큰수를 largest 변수에 대입합니다.
    return largest; // 현재 가장 큰 수인 40를 return 합니다.
}// 입력: n, 처리: 큰점수 획득, 출력: int(정수)
 // 왜 scores 배열이 밑에서도 쓰일수 있느냐
 // 이유는 int scores[MAX_ELEMENTS];가 전역변수 이기 때문이다!
```

- - scores[2] = 40; //scanf("%d",&scores[2]); 이런식으로 scanf로 바로 입력받는 방법도 있습니다.

■ 02 calc_time.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 1000000 //define을 이용하여 MAX를 1000000으로 설정

int main(void)
{
    clock_t start, stop; // start,stop: 측정시작과 종료를 나타내는 변수
    double  duration;
    start = clock();    // 측정 시작

    for (int i = 0; i < MAX; i++)   // 의미 없는 반복 루프
        // i가 0~99999번 돌동안의 시간을 측정합니다.

        ;
    stop = clock(); // 측정 종료
    duration = (double)(stop - start) / CLOCKS_PER_SEC;
    // 형변환을 해준이유 duration이 double형이기 때문입니다.
    printf("수행시간은 %lf초입니다.\n", duration);
    //결과값을 출력하는 문장
    return 0;
}
```

■ 03 seq_search.c

```c
#include <stdio.h>
#define MAX 10

int main(void)
{
    int search;
    int result;
    int array1[MAX] = {23,32,42,35,216,23,62,33,15,6}; //배열의 이름은 배열의 시작주소
    scanf("%d", &search); //scanf를 이용해서 좀 더 개선된 코드를 이용할 수 있습니다.
    //start
    result=seq_search(array1, MAX, search);
    //stop
    printf("결과: %d₩n", result);
    return 0;

}
int seq_search(int* list, int n, int key) // 순차 탐색?
// list는 주소를 저장하는 포인터변수; 지금은 array1을 포인팅하고 있다.
// int* list=array; int n=10; int key=15
// 입력: 3개, 처리: 순차적으로 data를 찾는다, 출력: int 정수로 한다
{
    int i; // i는 0~9까지의 인덱스 번호
    for (i = 0; i < n; i++)
        if (list[i] == key)
            return i;  /* 탐색에 성공하면 키 값의 인덱스 반환*/
    return -1;    /* 탐색에 실패하면 -1 반환 */
}
```

시퀀스서치에서는 절대로 메인함수의 array를 사용할 수가 없다. 따라서 주소값을 저장해서 놓아야 한다.

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 장의 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	27 점		

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

노력지수 (10점 만점)	• 7점 - 아직 너무 부족하다. 불안하다. 다른 학우들은 1학기때 다 끝냈는데 나만 2학기에 공부하는 느낌이다. 하지만 A+를 위해 열심히 공부하겠다. 파이팅이다
느낀 점 (10점 만점)	• 8점 - **열심히 공부해서 좋은 성적을 받기 위해 달려가고 있는 나의 모습을 발견했다. 파이팅하자**

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