Tech University of Korea(TUK)
자료구조 과제 노트

202#-0#학기

담당교수	박정민
학번	•
이름	• 박##

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

■ 일상생활과 자료구조의 비교

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

- 프로그램 = 자료구조 + 알고리즘

■ 알고리즘의 조건

- 알고리즘(algorithm): 컴퓨터로 문제를 풀기 위한 단계적인 절차
  - 오른쪽에서 왼쪽으로 해석.
  - 입력, 처리, 출력을 파악하기.
  - 자료구조가 바탕이되어 알고리즘을 처리

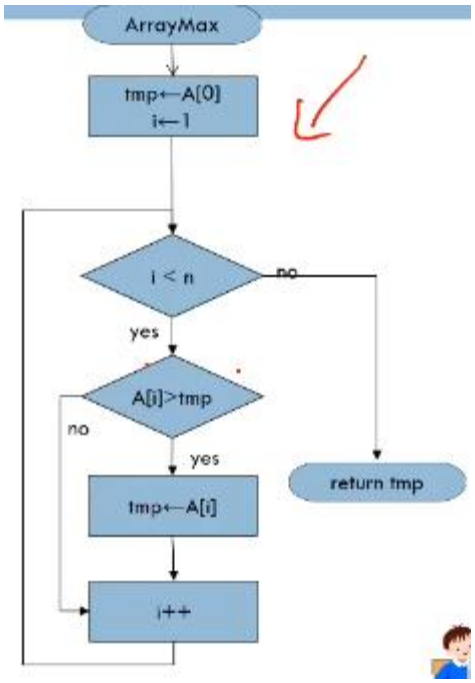
- 알고리즘의 조건
  - 입 력 : 입력이 존재하여야 한다.
  - 출 력 : 출력이 존재하여야 한다.

  - 자연어 (영어나 한국어같은)
  - 순서도 (flow chart)
  - 유사코드 (pseudo-code)
  - C와 같은 프로그래밍 언어

- 자연어로 표기된 알고리즘
  - 인간이 읽기가 쉬움
  - 그러나 자연어의 단어들을 정확하게 정의하지 않으면 의미 전달이 모호해질 우려가 있음
  - 예)

```
ArrayMax(list, n)

1. 배열 list의 첫번째 요소를 변수 tmp에 복사
2. 배열 list의 다음 요소들을 차례대로 tmp와 비교하면 더 크면 tmp
   로 복사
3. 배열 list의 모든 요소를 비교했으면 tmp를 반환
```

- 순서도(흐름도)로 표기된 알고리즘
  - 직관적이고 이해하기 쉬운 알고리즘 기술 방법
  - 그러나 복잡한 알고리즘의 경우, 상당히 복잡해짐.
  - 예)



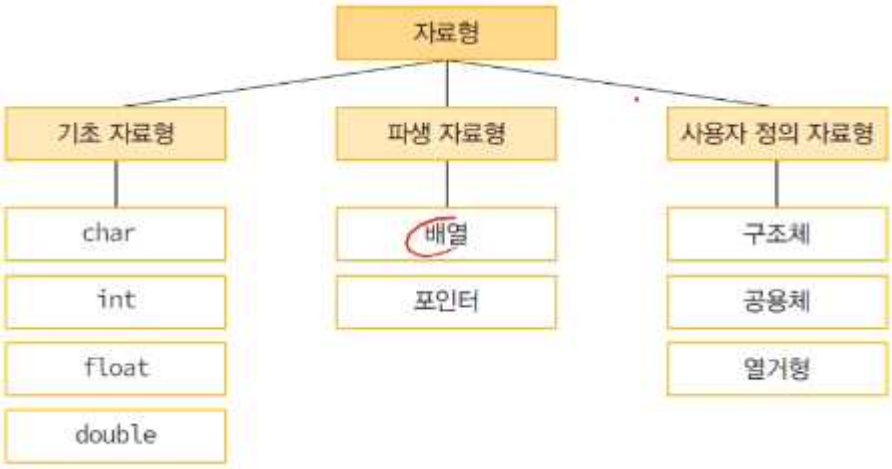
- 유사코드(슈더코드)로 표현된 알고리즘
  - 알고리즘 기술에 가장 많이 사용
  - 프로그램을 구현할 때의 여러 가지 문제들을 감출 수 있다.
  - 즉, 알고리즘의 핵심적인 내용에만 집중할 수 있다.
  - 예)

```
ArrayMax(list, N):
    largest←list[0]
    for i←1 to N-1 do
        if list[i]>largest
            then largest←list[i]
    return largest
```

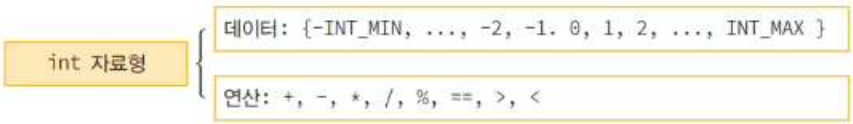
- C로 표현된 알고리즘
  - 알고리즘의 가장 정확한 기술이 가능
  - 반면 실제 구현 시, 많은 구체적인 사항들이 알고리즘의 핵심적인 내용에 대한 이해를 방해할 수 있다.

■ 자료형

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



  - 데이터의 집합과 연산의 집합



  - limits.h 사용하면 자료형의 최솟값과 최댓값을 사용할 수 있음.

■ 추상 데이터 타입

- 추상 데이터 타입(ADT : Abstract Data Type)
  - 데이터 타입을 추상적으로 정의한 것
  - 데이터나 연산이 **무엇(What)**인가는 정의되지만 데이터나 연산을 **어떻게(How)** 컴퓨터 상에서 구현할 것인지는 정의되지 않는다.

- 추상데이터 타입의 유래
  - 추상화 -> 정보은닉기법 -> 추상 자료형
  - 추상화란 사용자에게 **중요한 정보는 강조**되고 반면 **중요하지 않은 구현 세부 사항은 제거**하는 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.

  - 사용자들은 ADT가 제공하는 연산만을 사용할 수 있다. (TV의 인터페이스가 제공하는 특정한 작업만을 할 수 있다.)
  - 사용자들은 ADT내부의 데이터를 접근할 수 없다. (사용자는 TV의 내부를 볼 수 없다.)
  - 사용자들은 ADT가 어떻게 구현되는지 모르더라도 ADT를 사용할 수 있다. (TV의 내부에서 무엇이 일어나고 있는지를 몰라도 이용할 수 있다.)

■ 알고리즘의 성능분석 (21P)

- 직접 수행 (수행시간측정)
- 간접 수행 (약 얼마냐. 대문자O사용)

● 알고리즘의 성능 분석 기법

- 수행 시간 측정 (직접)

(1) 두 개의 알고리즘의 실제 수행 시간을 측정하는 것.
(2) 실제로 구현하는 것이 필요.
(3) 동일한 하드웨어를 사용하여야 한다. (시험)

- 알고리즘의 복잡도 분석 (빅오)

(1) 직접 구현하지 않고서도 수행 시간을 분석하는 것.
(2) 알고리즘이 수행하는 연산의 횟수를 측정하여 비교
(3) 일반적으로 연산의 횟수는 n의 함수
(4) n은 자료의 개수

■ 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

복잡도 낮아야 좋은 것.

■ 수행시간 측정

● 알고리즘을 프로그래밍 언어로 작성하여 실제 컴퓨터상에서 실행시킨 다음, 그 수행시간을 측정
● 비교할 하드웨어 동일, 입력(n)의 개수가 같아야 함.

● 측정 2가지 방법

방법 #1

```
#include <time.h>

start = clock();
...
stop = clock();
double duration = (double)(stop - start) /
CLOCKS_PER_SEC;
```

방법 #2

```
#include <time.h>

start = time(NULL);
...
stop = time(NULL);
double duration = (double) difftime(stop,
start);
```

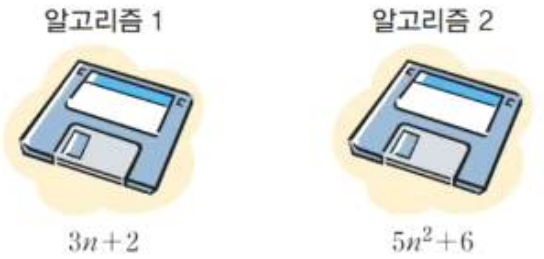
- duration = finish - start;
- time.h에 CLOCKS_PER_SEC; 정의되어 있음.

■ 복잡도 분석

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

- 복잡도 분석의 종류
  - 시간 복잡도(time complexity)
  - 공간 복잡도(space complexity)

- 시간 복잡도
  - 간접 : 연산의 수
  - 직접 : 시간



  - n=2일 때 1은 8 2는 26
  - 2가 더 복잡하다.

- 분석의 예

알고리즘 A	알고리즘 B	알고리즘 C

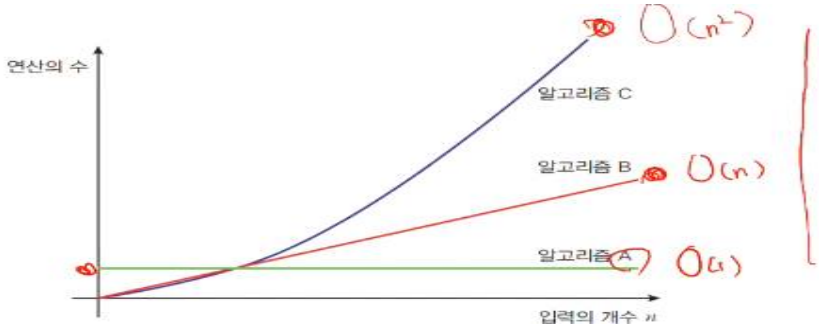
sum ←n*n;	for i←1 to n do
	sum ←sum + n;
		for i←1 to n do
		for j←1 to n do
		sum ←sum + 1;

  - n은 10일때

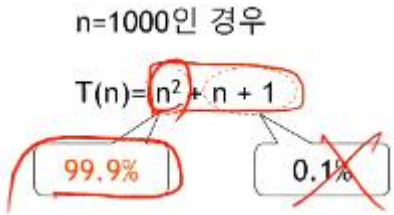
	알고리즘 A	알고리즘 B	알고리즘 C
대입연산	1	n	$n*n$
덧셈연산		n	$n*n$
곱셈연산	1		
나눗셈연산			
전체연산수	2	$2n$	$2n^2$

  - 알고리즘 A는 O(1)
  - 알고리즘 B는 O(n)
  - 알고리즘 C는 O(n^2)

- 그래프로 표기



위로 갈수록 복잡도 높아짐.

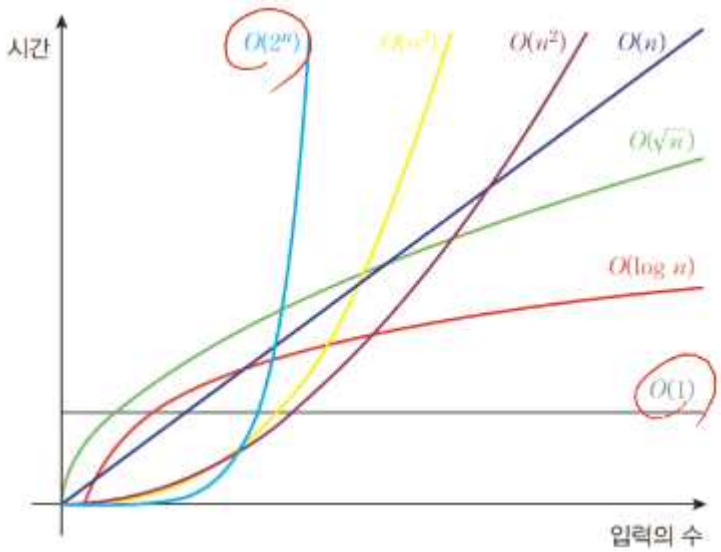
- 빅오 표기법
  - 자료의 개수가 많은 경우에는 차수가 가장 큰 항이 가장 영향을 크게 미치고 다른 항들은 상대적으로 무시될 수 있다.
  - 

  - 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

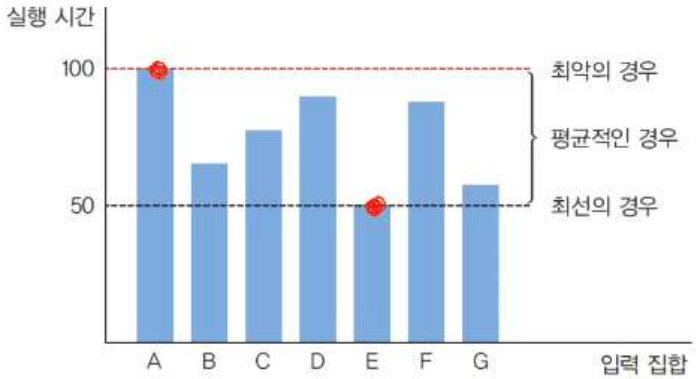
$f(n)$	$O(f(n))$
10	$O(1)$
$5n^2+6$	$O(n^2)$
$2n^3+1$	$O(n^3)$
$2n^3+5n^2+6$	$O(n^3)$

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
  - 알고리즘으 수행시간은 입력 자료 집합에 따라 다를 수 있다.

  - 최선의 경우: 수행시간이 가장 빠른 경우
  - 평균의 경우: 수행시간이 평균적인 경우
  - 최악의 경우: 수행 시간이 가장 늦은 경우



  - 예)

□ 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

$(1+2+\cdots+n)/n=(n+1)/2$

∴ O(n)

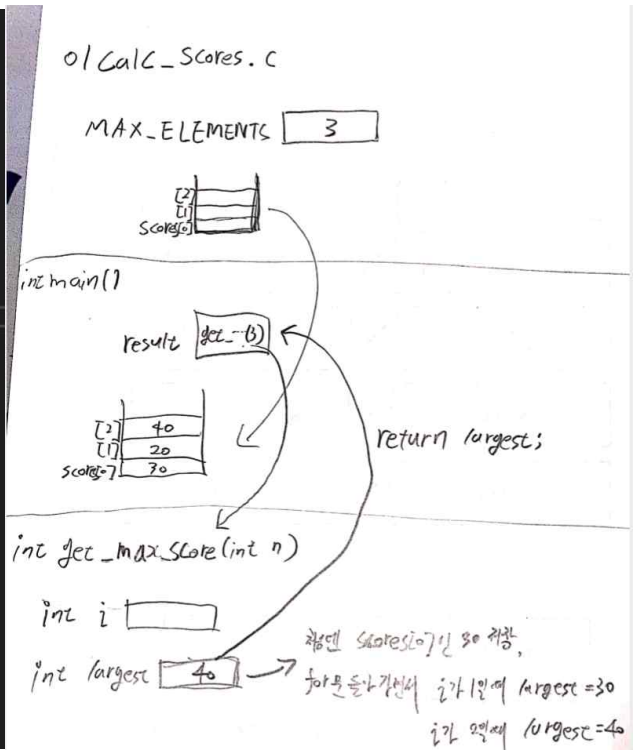
- 최선의 경우 : 의미가 없는 경우가 많다.
- 평균적인 경우 : 계산하기가 상당히 어려움.
- **최악의 경우 : 가장 널리 사용된다.** 계산하기 쉽고 응용에 따라서 중요한 의미를 가질 수도 있다.

1-2 소스코드 디버깅 분석(역공부)

■ '손' 분석내용

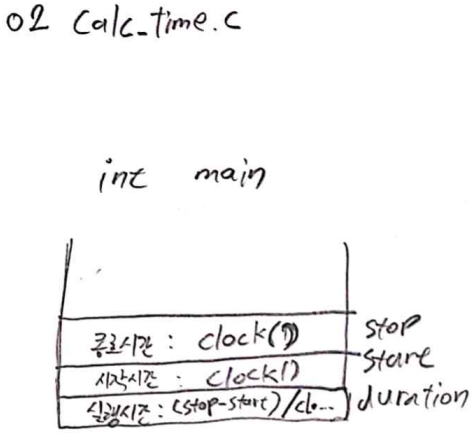
- 01 calc_scores.c

```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;
    scores[0] = 30;
    scores[1] = 20;
    scores[2] = 40;

    result = get_max_score(MAX_ELEMENTS);
    printf("결과:%d₩n ",result);
    system("pause");
    return 0;
}

int get_max_score(int n)    // 학생의 숫자는 n
{
    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: 큰점수 획득, 출력: int(정수)
```



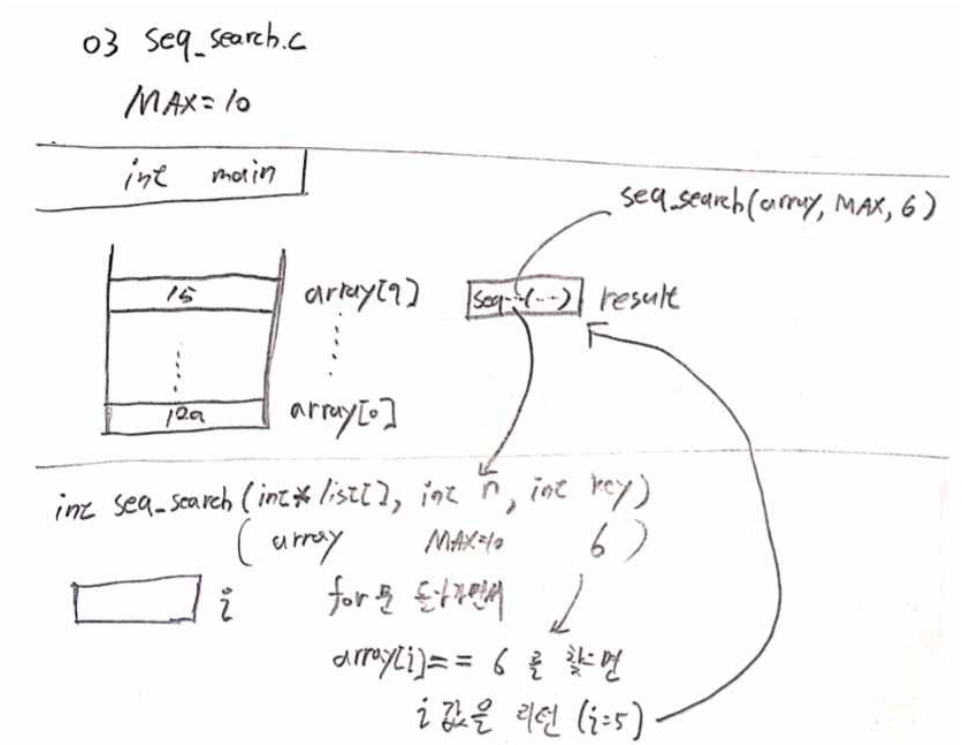
- 02 calc_time.c

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

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

    for (int i = 0; i < 1000000; i++)   // 의미 없는 반복 루프
        ;
    stop = clock(); // 측정 종료
    duration = (double)(stop - start) / CLOCKS_PER_SEC;
    printf("수행시간은 %f초입니다.₩n", duration);
    return 0;
}
```



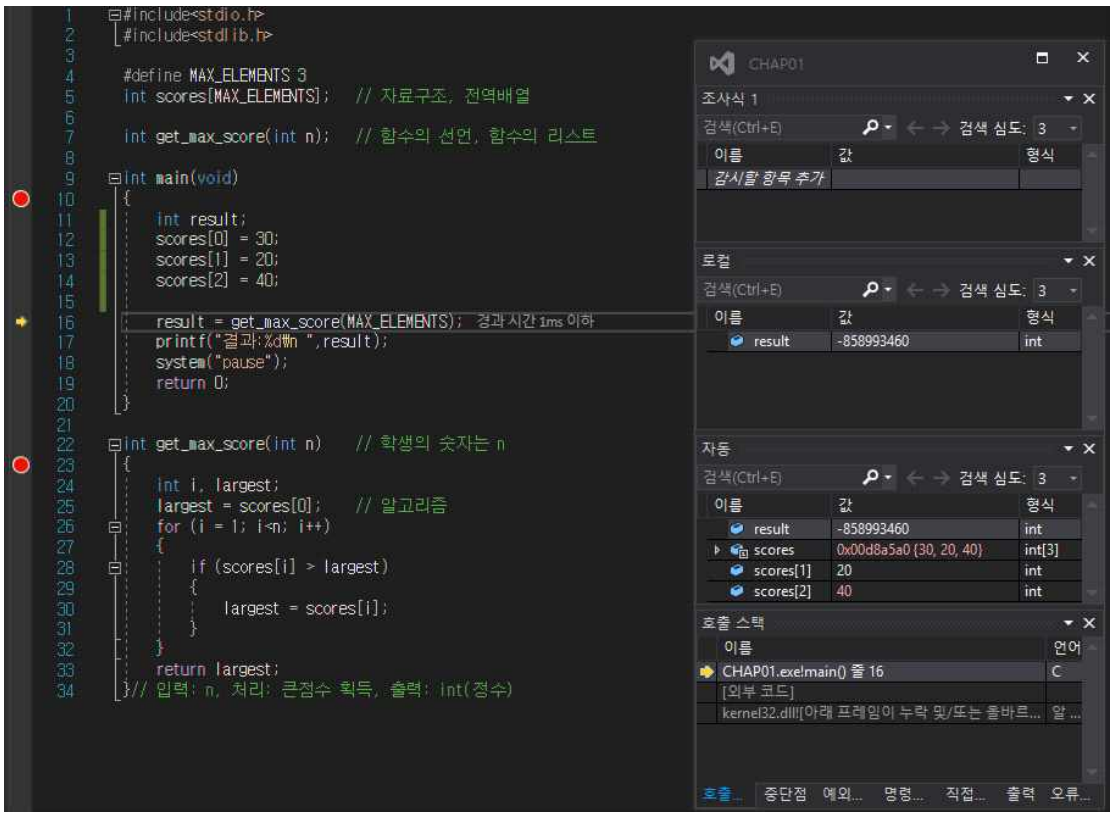
- 03 calc_search.c

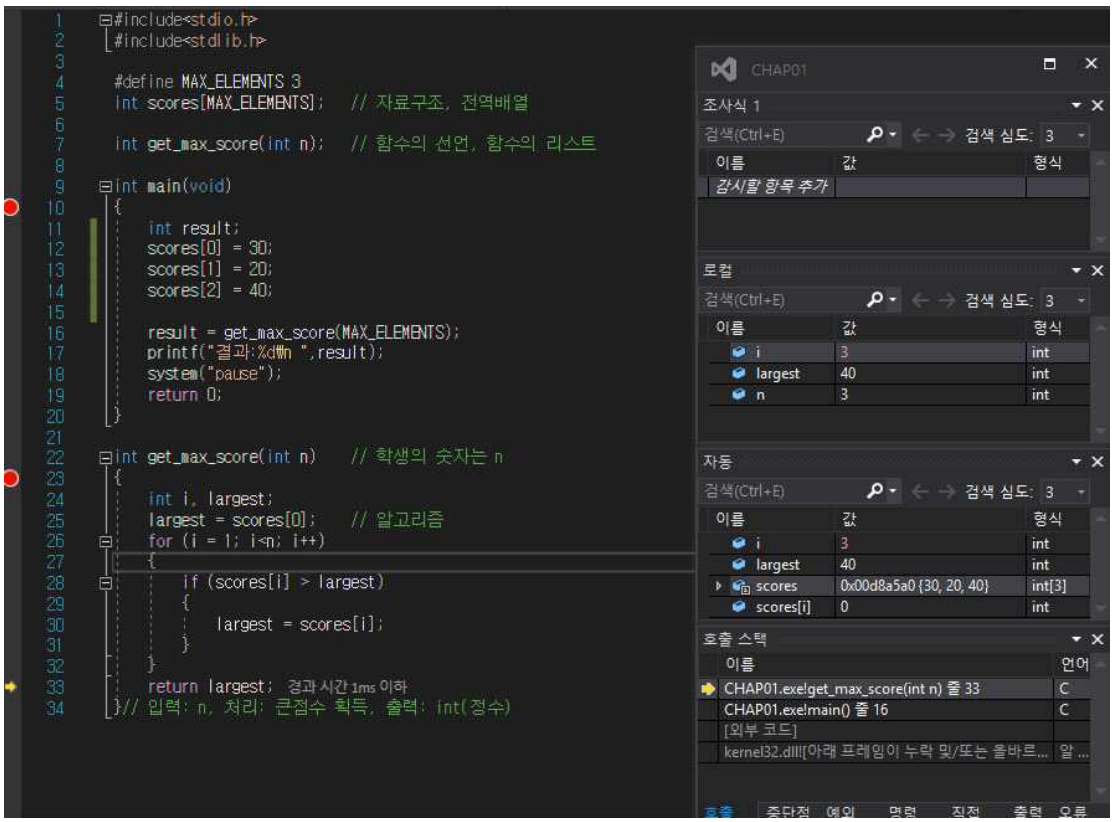
```c
#include<stdio.h>
#define MAX 10

int main()
{
    int array[MAX] = { 123,41,23,41,23,6,51,2,34,15 }; //배열의 이름은 배열의 시작 주소
    int result;

    result = seq_search(array, MAX, 6);
    printf("%d\n", result);

    return 0;
}
int seq_search(int* list[], int n, int key)// int* list= array, int n = 10, int key =6
{
    int i;
    for (i = 0; i < n; i++)
        if (list[i] == key)
            return i;  /* 탐색에 성공하면 키 값의 인덱스 반환 */
    return -1;    /* 탐색에 실패하면 -1 반환 */
}
```

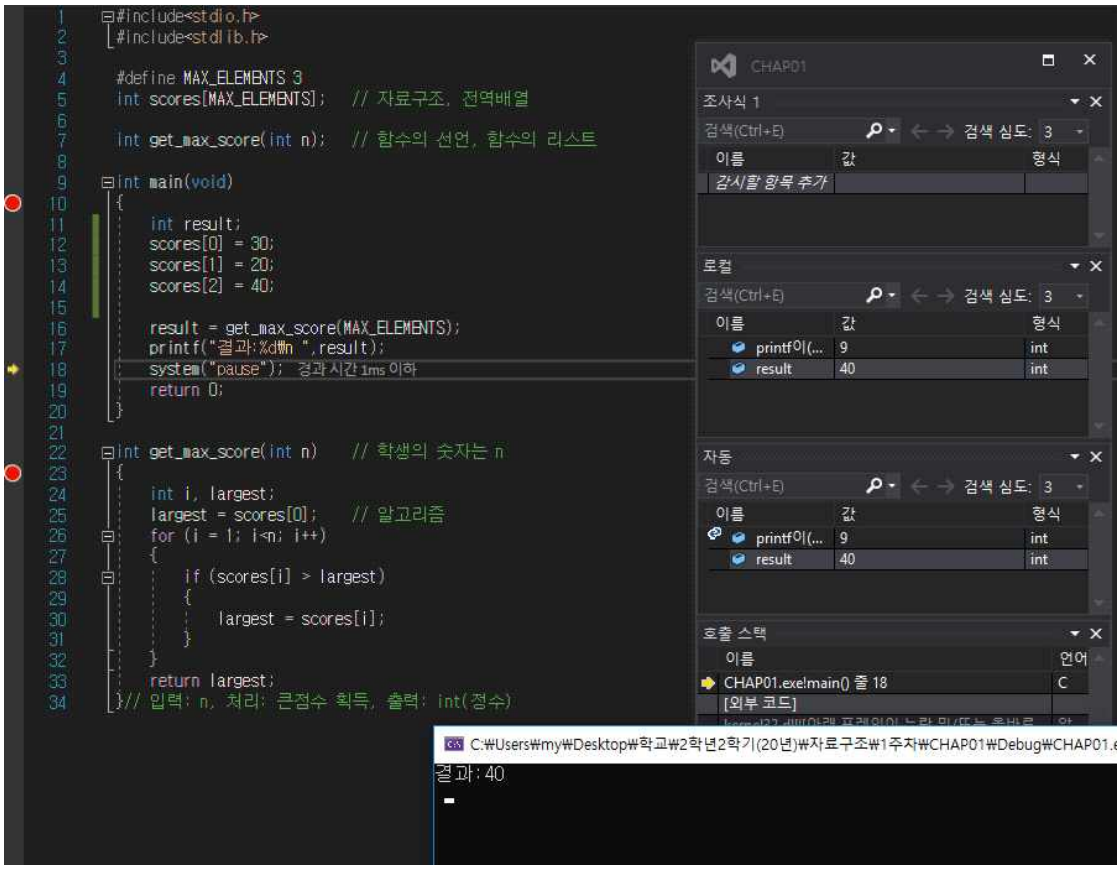


■ 디버깅SW를 이용한 분석내용

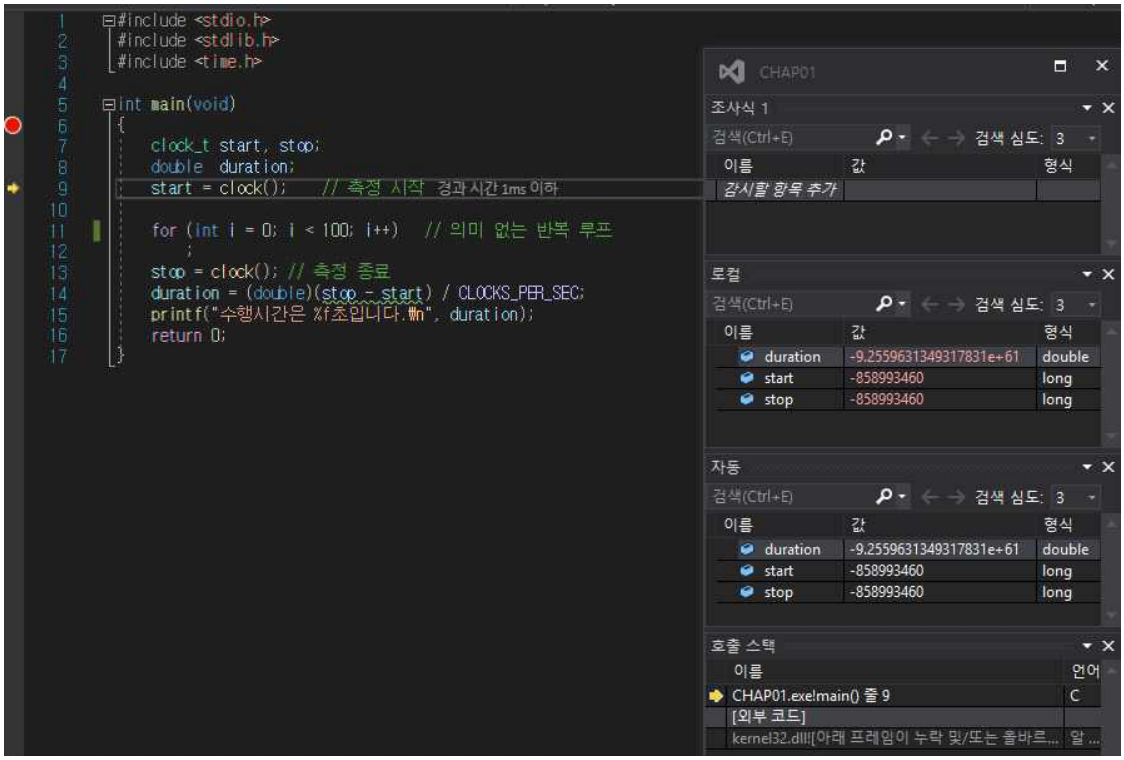
- 01 calc_scores.c

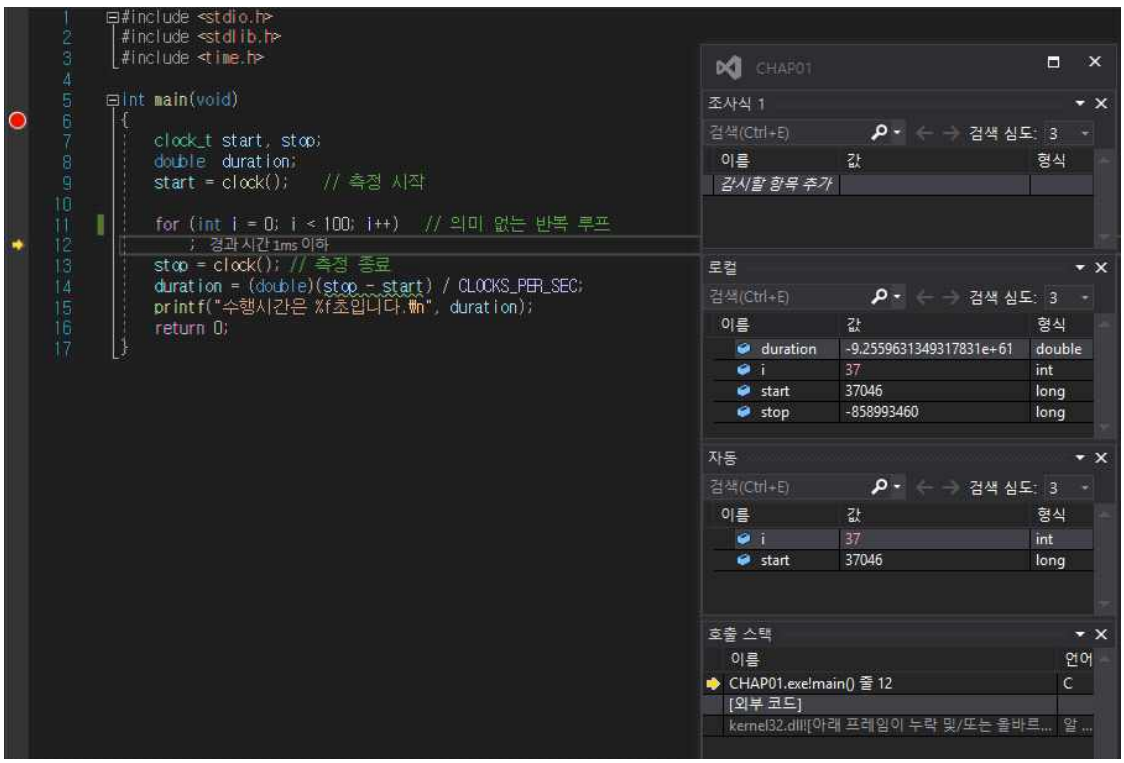


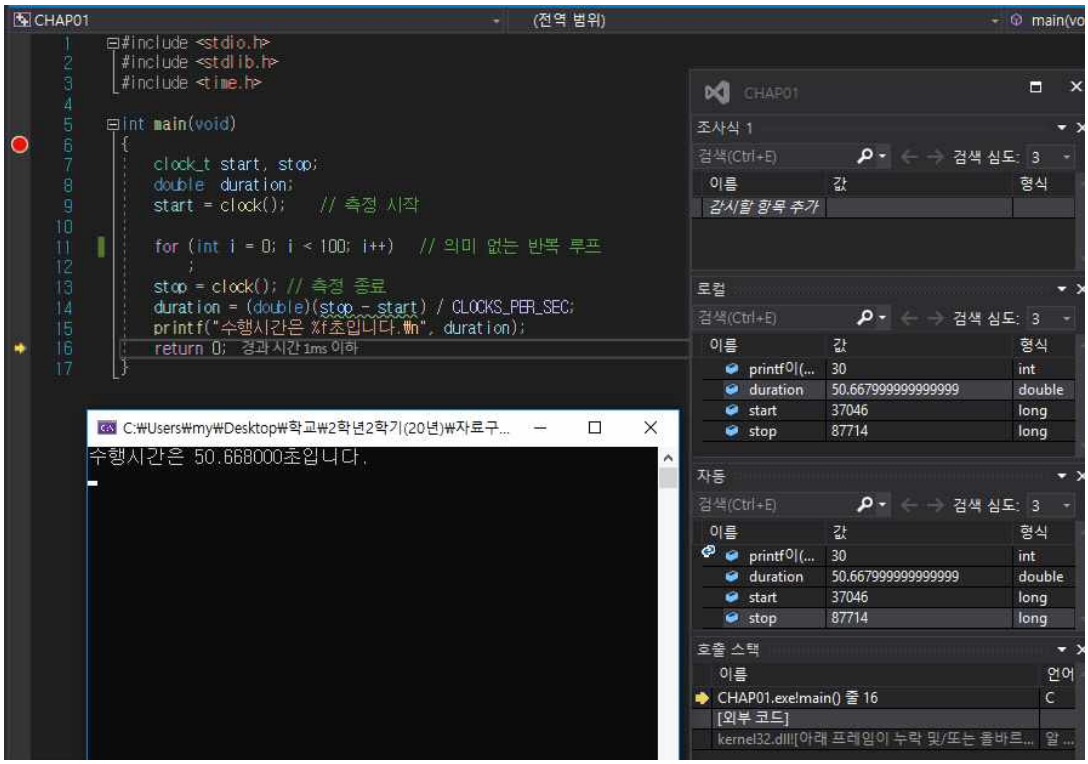



#include<stdio.h>
#include<stdlib.h>
#define MAX_ELEMENTS 3
int scores[MAX_ELEMENTS]; // 자료구조, 전역배열
int get_max_score(int n); // 함수의 선언, 함수의 리스트
int main(void)
{
int result;
scores[0] = 30;
scores[1] = 20;
scores[2] = 40;
result = get_max_score(MAX_ELEMENTS);
printf("결과:%d₩n ",result);
system("pause"); 경과 시간 1ms 이하
return 0;
}
int get_max_score(int n) // 학생의 숫자는 n
{
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: 큰점수 획득, 출력: int(정수)
CHAP01
조사식 1
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
감시할 항목 추가
로컬
printf이(... 9 int
result 40 int
자동
호출 스택
CHAP01.exe!main() 줄 18 C
[외부 코드]
C:₩Users₩my₩Desktop₩학교₩2학년2학기(20년)₩자료구조₩1주차₩CHAP01₩Debug₩CHAP01.
결과:40

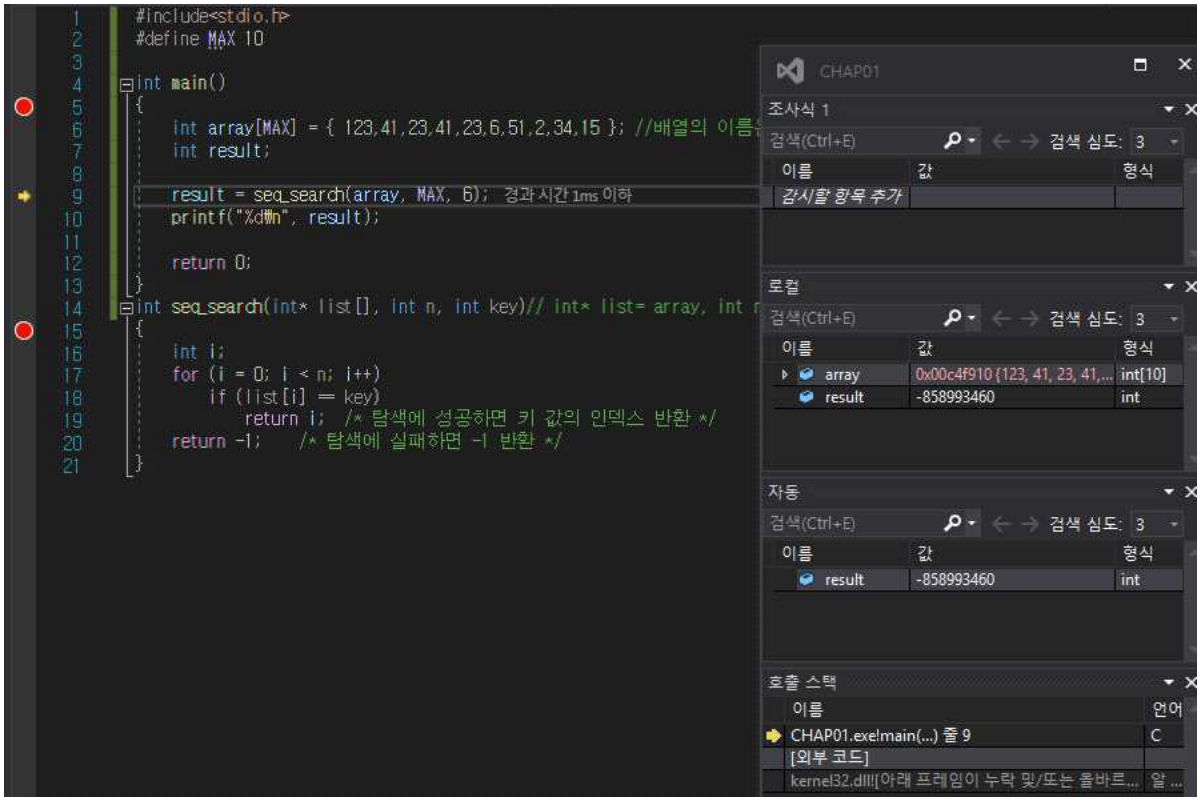
- 02 calc_time.c

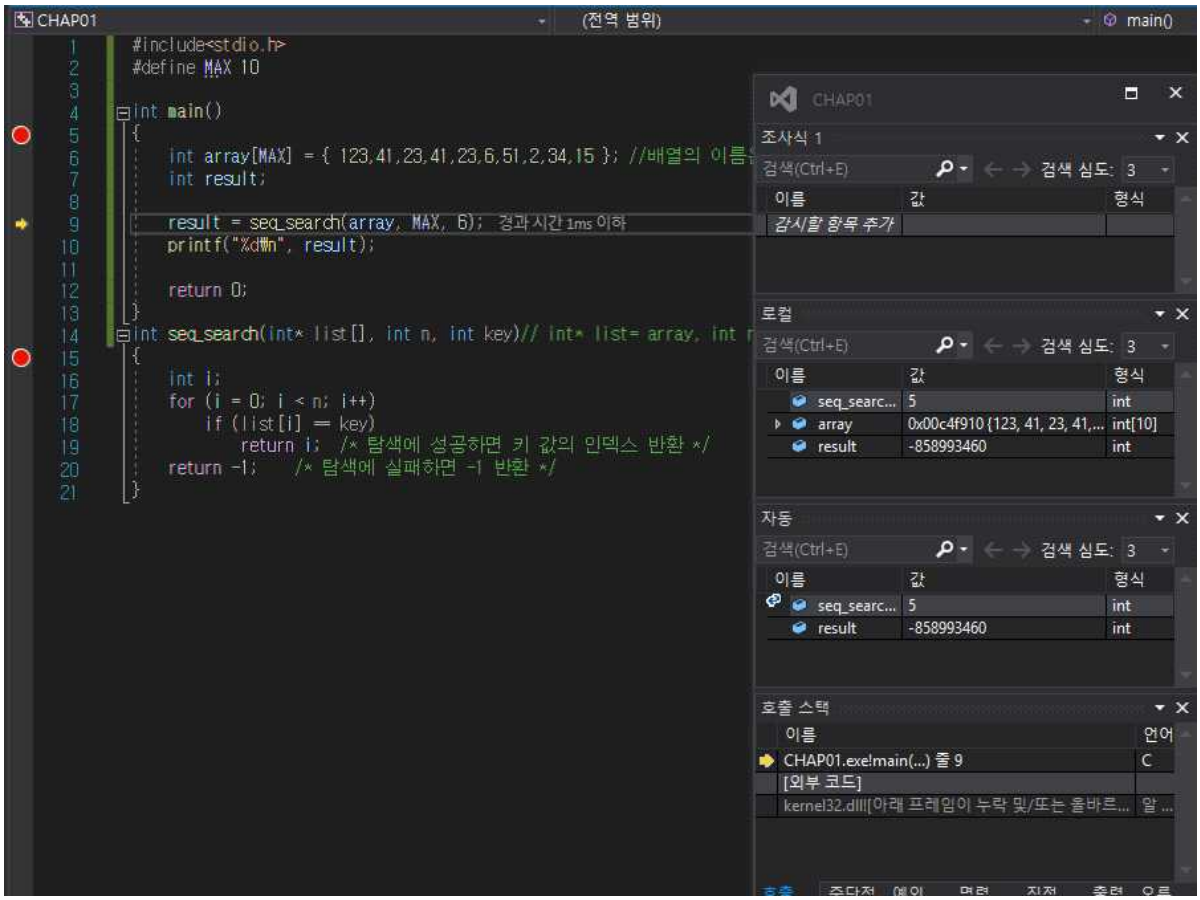


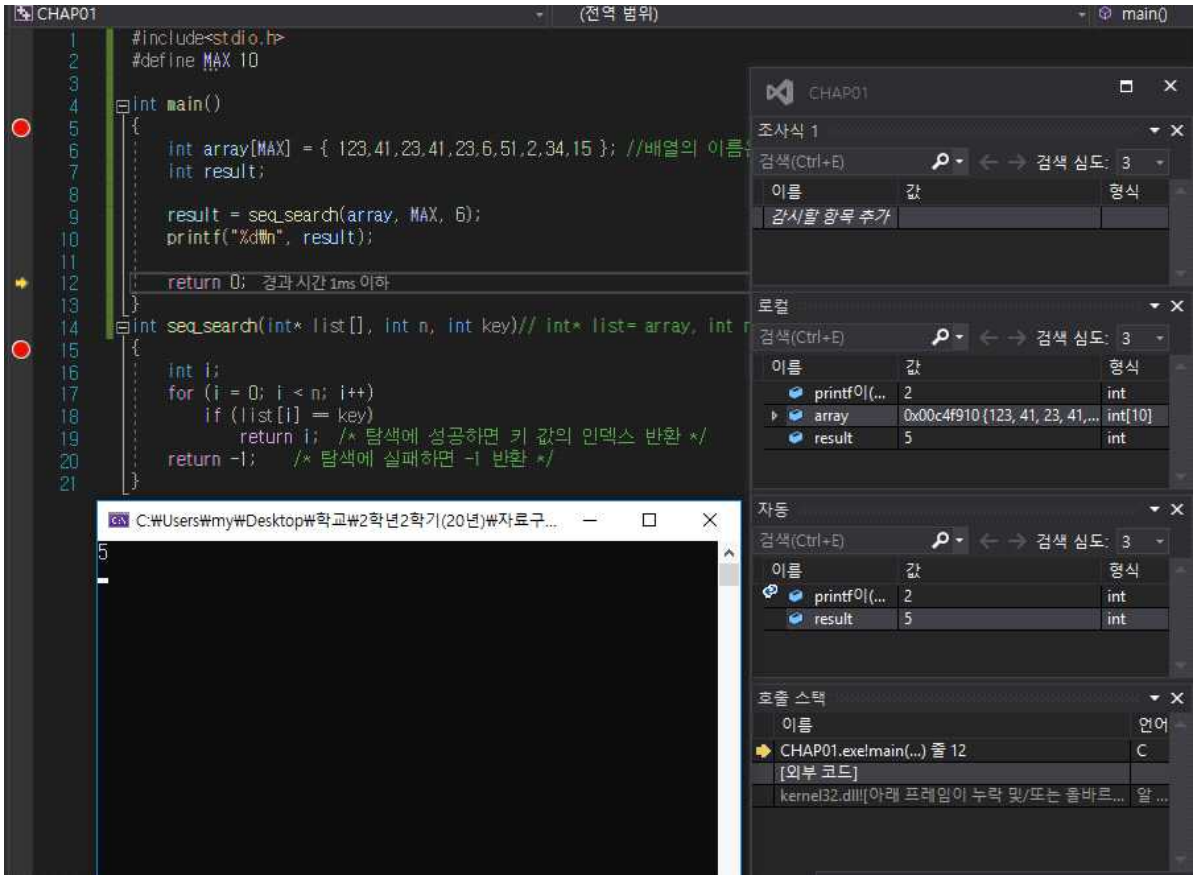



CHAP01
(전역 범위)
main(vo
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
clock_t start, stop;
double duration;
start = clock(); // 측정 시작
for (int i = 0; i < 100; i++) // 의미 없는 반복 루프
;
stop = clock(); // 측정 종료
duration = (double)(stop - start) / CLOCKS_PER_SEC;
printf("수행시간은 %f초입니다.₩n", duration);
return 0; 경과 시간 1ms 이하
}
C:₩Users₩my₩Desktop₩학교₩2학년2학기(20년)₩자료구...
수행시간은 50.668000초입니다.
CHAP01
조사식 1
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
감시할 항목 추가
로컬
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
printf이(... 30 int
duration 50.667999999999999 double
start 37046 long
stop 87714 long
자동
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
printf이(... 30 int
duration 50.667999999999999 double
start 37046 long
stop 87714 long
호출 스택
이름
언어
CHAP01.exe!main() 줄 16 C
[외부 코드]
kernel32.dll![아래 프레임이 누락 및/또는 올바르...

- 03 calc_search.c






CHAP01
(전역 범위)
main()
#include<stdio.h>
#define MAX 10
int main()
{
int array[MAX] = { 123,41,23,41,23,6,51,2,34,15 }; //배열의 이름
int result;
result = seq_search(array, MAX, 6);
printf("%d₩n", result);
return 0; 경과 시간 1ms 이하
}
int seq_search(int* list[], int n, int key)// int* list= array, int
{
int i;
for (i = 0; i < n; i++)
if (list[i] == key)
return i; /* 탐색에 성공하면 키 값의 인덱스 반환 */
return -1; /* 탐색에 실패하면 -1 반환 */
}
C:₩Users₩my₩Desktop₩학교₩2학년2학기(20년)₩자료구...
5
CHAP01
조사식 1
검색(Ctrl+E)
검색 심도: 3
이름
값
형식
감시할 항목 추가
로컬
printf이(... 2 int
array 0x00c4f910 {123, 41, 23, 41,... int[10]
result 5 int
자동
printf이(... 2 int
result 5 int
호출 스택
이름
언어
CHAP01.exe!main(...) 줄 12 C
[외부 코드]
kernel32.dll![아래 프레임이 누락 및/또는 올바르...

1-3 소스코드 수정/개선 & 주석(순공부)

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

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

#define MAX_ELEMENTS 100
int scores[MAX_ELEMENTS];   // 자료구조, 전역배열

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

int main(void)
{
    int num;
    printf("학생이 몇명인지 입력하시오(최대100명) : "); //반복문을 몇번 돌려야하는지 알기 위해
    scanf("%d", &num);
    int i;
    for (i = 0; i < num; i++) {
        printf("학생%d의 성적을 입력하세요 : ", i+1);
        scanf("%d", &scores[i]);    //scanf로 학생들의 성적을 받아서scores[]에 학생들의 성적을 차례로 입력
    }

    int result;
    result = get_max_score(MAX_ELEMENTS);
    printf("가장 점수가 높은 점수는 : %d 입니다.₩n ",result); //학생들의 점수 중 가장 높은 점수를 출력.
    system("pause");
    return 0;
}

int get_max_score(int n)    // 학생의 숫자는 n
{
    int i, largest;
    largest = scores[0];    // 알고리즘
    for (i = 1; i<n; i++)
    {
        if (scores[i] > largest)
        {
            largest = scores[i];
        }
    }
```

- 학생들의 성적을 입력받아서 그 중 가장 높은 점수를 출력하는 코드로 만들어 보았습니다.
- 일단 MAX_ELEMENTS를 100으로 define 해 놓았기 때문에 처음 학생의 수를 입력할 때 100명 이하로 입력해 달라고 언급했습니다.
- 학생의 수를 입력 받는 이유는 학생들의 점수를 입력 받을 때 반복문을 돌리는데 이 때 반복문을 몇 번 돌릴지 정하기 위해서입니다.
- 그 이후 for문을 돌려서 학생들의 성적을 차례로 입력받게 했습니다.
- 함수는 따로 건들이지 않고 간단하게 수정 및 개선해보았습니다.

■ 수정/개선 & 주석2

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

#define MAX 10

int main()
{
    clock_t start, stop;
    double  duration;


    int array[MAX] = { 123,41,23,41,23,6,51,2,34,15 }; //배열의 이름은 배열의 시작 주소
    int result;
    int want;
    printf("찾는 숫자를 입력해주세요:");
    scanf("%d", &want);     //원하는 숫자를 입력받기

    start = clock();    // 측정 시작

    result = seq_search(array, MAX, want);  //

    stop = clock(); // 측정 종료

    duration = (double)(stop - start) / CLOCKS_PER_SEC;
    if (result == -1) {     //숫자를 못 찾았을 경우를 언급
        printf("당신이 원하는 숫자를 찾지 못했습니다.₩n");
    }
    else{       //숫자를 찾았을 경우 찾는데 걸린 시간과 함께 찾은 숫자가 있는 사물함을 언급해준다.
        printf("당신이 원하는 숫자는 %d번째 사물함안에 있습니다.₩n", result+1);

        printf("사물함을 찾는데 %lf초 걸렸습니다.");
    }

    return 0;
}
int seq_search(int* list[], int n, int key)// int* list= array, int n = 10, int key =6
{
    int i;
    for (i = 0; i < n; i++)
        if (list[i] == key)
            return i;  /* 탐색에 성공하면 키 값의 인덱스 반환 */
    return -1;    /* 탐색에 실패하면 -1 반환 */
}
```

- 2번과 3번을 합쳐서 원하는 숫자가 들어있는 사물함을 찾아주고 찾는데 걸린 시간을 알려주는 코드를 만들어 보았습니다.
- time.h와 stdlib.h파일을 이용해서 사물함을 찾는데 걸리는 시간을 측정하게 해줬습니다.
- 찾는 숫자를 입력받도록 scanf를 이용해서 want에 저장해 주었습니다.
- 그 후 함수를 돌려 result에 사물함의 번호를 입력받습니다. 여기서 사물함을 찾지 못했을 경우 -1을 return 받는데 이 경우에는 원하는 숫자를 찾지 못했다는 언급을 해주기 위해 if문을 사용해서 경우를 나누었습니다.
- 사물함의 번호를 알려줄 때 0번째 사물함을 없애기 위해 result+1을 해주어서 1번째 사물함부터 나오도록 해주었습니다.

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	30 점		

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

노력지수 (10점 만점)	• 10점 - 충분한 여유를 가지고 여러 시간을 들여 하루에 몰아서 공부하지 않고 할 수 있는만큼 나눠서 충분히 공부했습니다. 과제와 영상에서 언급한 부분들을 열심히 공부했고 나름의 시선으로 좋은 결과를 낳기 위해 노력했습니다.
느낀 점 (10점 만점)	• 10점 - 첫 주인데 방학동안 자바를 공부 했고, 이번학기에 파이썬, 자료구조, 자바 3가지의 언어를 한번에 배우다보니 여러 헷갈리는 부분이 있어서 조금 힘들었다. 자료구조라는 과목이지만 첫 주에 배운 부분들은 공부하면서 내가 알고리즘이라는 수업을 듣고 있는 건지 헷갈릴 정도였다. 원래 C언어를 공부할 때 역공부만큼은 자신 있다고 생각했는데 자료구조를 들어와서 역공부를 요구해서 공부하는데 재미가 들렸다.

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