

알고리즘 2021 보고서

보고서 제출서약서

나는 숭실대학교 컴퓨터학부의 일원으로 명예를 지키면서 생활하고 있습니다.

나는 보고서를 작성하면서 다음과 같은 사항을 준수하였음을 엄숙히 서약합니다.

1. 나는 자력으로 보고서를 작성하였습니다.
 - 1.1. 나는 동료의 보고서를 베끼지 않았습니다.
 - 1.2. 나는 비공식적으로 얻은 해답/해설을 기초로 보고서를 작성하지 않았습니다.
2. 나는 보고서에서 참조한 문헌의 출처를 밝혔으며 표절하지 않았습니다. (나는 특히 인터넷에서 다운로드한 내용을 보고서에 거의 그대로 복사하여 사용하지 않았습니다.)
3. 나는 보고서를 제출하기 전에 동료에게 보여주지 않았습니다.
4. 나는 보고서의 내용을 조작하거나 날조하지 않았습니다.

과목	알고리즘 2021
과제명	과제1
담당교수	최 재 영 교 수
제출인	컴퓨터학부 20163340 강원경 (출석번호 201번)
제출일	2021년 09월 10일

차 례

1장 프로그램 개요 ----- 3p

2장 설계/구현 방법 ----- 3p

3장 실행결과 ----- 4p

4장 디버깅 ----- 5p

1장. 프로그램 개요

이번 학기동안 주로 사용하게 될 C언어와 그것을 다루기 위한 IDE의 디버거에 익숙해지는 것이 이번 과제의 중요한 목표이다. 지난 학기에 시스템 프로그래밍 과제를 하면서 Windows 환경에서 Visual Studio를 사용하여 프로그램을 작성한 적이 있기 때문에, 이번 학기의 과제는 조금 더 넓은 식견을 위해 이전과는 다른 환경에서의 프로그램 개발을 시도해보기로 했다. 그렇게 선택한 것이 macOS 환경에서의 프로그래밍이다.

지난 수년간 사용해왔던 개발환경을 하루 아침에 다른 플랫폼과 다른 IDE를 사용한다는 것은 거의 새로운 모험을 떠나는 일과 같다. 하지만, 이 또한 나의 프로그래밍 실력에 도움이 될 것이라 생각했고, 이러한 시도도 학부생일 때 가능한 일이라 생각했다.

따라서 이번 과제는 현 과제 수행 시점에서의 macOS 버전과 IDE의 최신버전을 채용하였고, 각각 macOS 11.5.2 (Big Sur), Xcode 12.5.1 버전이다.

2장. 설계/구현 방법

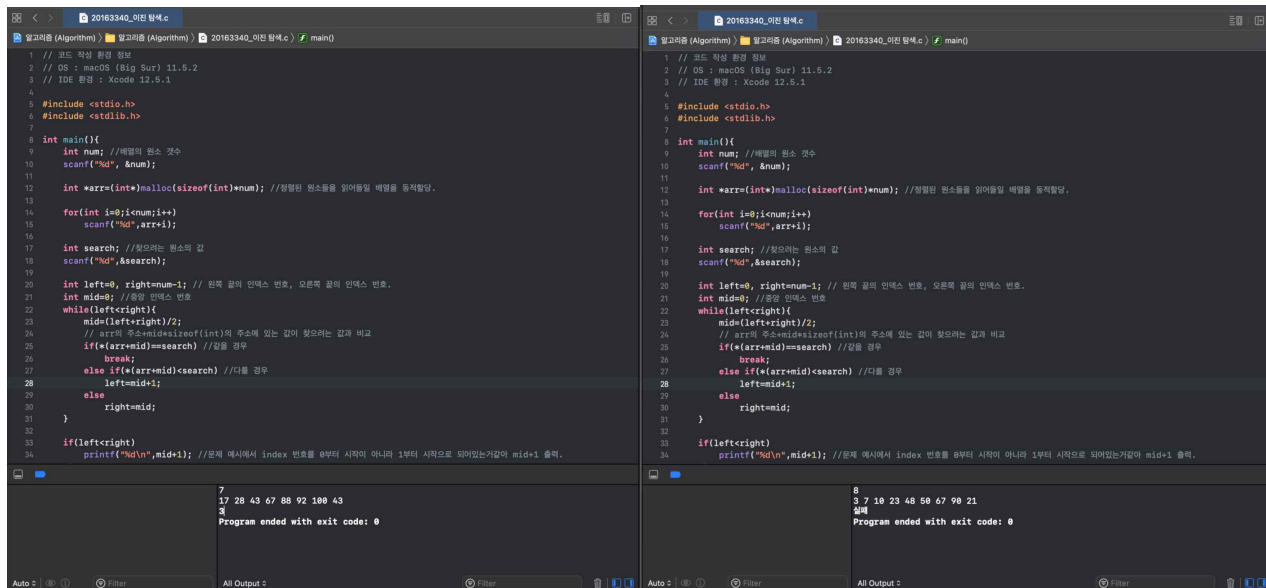
프로그램을 작성하는데에 앞서서 이번 과제에서 중요하게 생각한 것 중 하나가 있다. 전역변수를 사용하지 않는 것이 바로 그것이다. 전역변수를 사용하면 코드가 간결해지고 다른 함수에서 변수를 참조할 때 변수의 위치나 유효범위를 생각하지 않아도 되는 장점이 있기는 하나, 프로그래밍 실력, 더 나아가 실무에서의 내 능력을 키우기 위해서는 전역변수의 사용보다 포인터를 이용한 다른 함수 내에서의 변수 참조를 사용하는 것이 바람직하다고 생각했다.

이제 본문으로 넘어가보겠다. 이번 과제에서는 재귀함수를 이용한 프로그램 하나와 반복문을 사용한 프로그램 하나를 작성하였다. 재귀함수는 main 함수 이외에 `binarySearch`라는 함수를 작성하였고, 이 함수에서 리턴하는 값에 따라 검색 결과의 성공/실패 여부를 판단하였다.

반복문을 사용한 이진 탐색은 `while`문을 사용하였으며, 탐색하려는 범위의 인덱스를 나타내는 `left`와 `right` 변수를 두었다. `left`의 값이 `right` 값보다 같거나 커지기 전까지 반복을 수행하며, 탐색에 성공한 경우 더 이상 반복을 수행하지 않고 `break`를 사용하여 그 데이터가 있는 인덱스 번호를 출력한다.

3장. 수행 결과

1. 반복문을 사용한 이진탐색

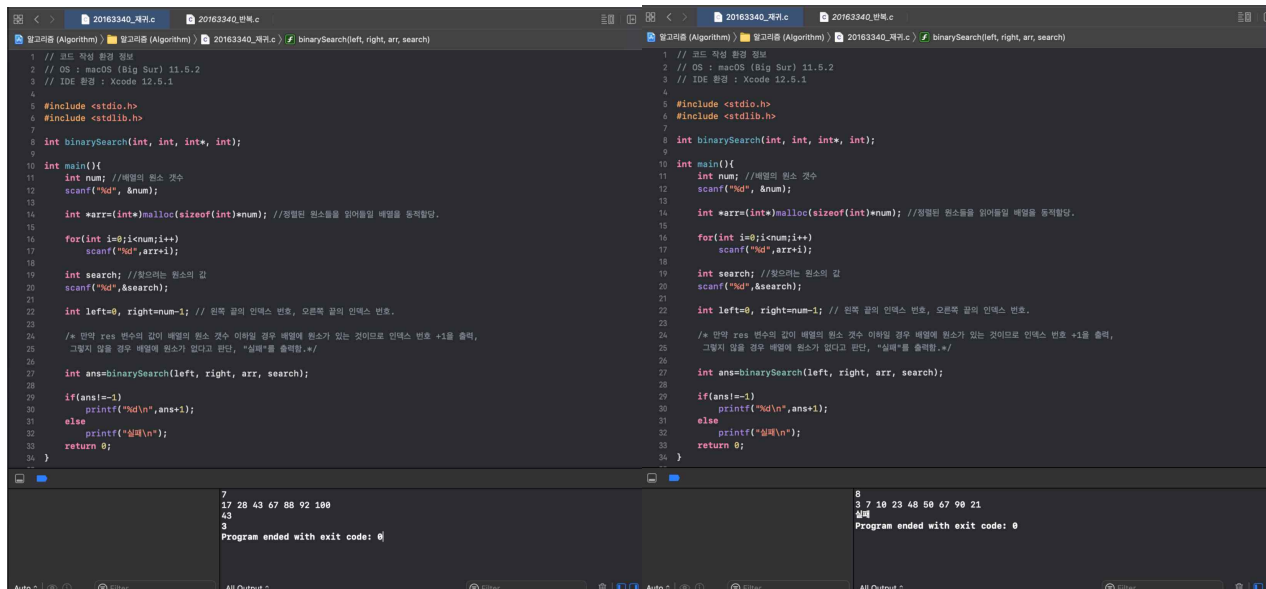


```
1 // 코드 작성 환경 정보
2 // OS : macOS (Big Sur) 11.5.2
3 // IDE 환경 : Xcode 12.5.1
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(){
9     int num; //배열의 원소 값
10    scanf("%d", &num);
11
12    int *arr=(int*)malloc(sizeof(int)*num); //정렬된 원소들을 읽어올 배열을 동적할당.
13
14    for(int i=0;i<num;i++){
15        scanf("%d", &arr[i]);
16    }
17
18    int search; //찾으려는 원소의 값
19    scanf("%d", &search);
20
21    int left=0, right=num-1; // 왼쪽 끝의 인덱스 번호, 오른쪽 끝의 인덱스 번호.
22    int mid=0; //중앙 인덱스 번호
23    while(left<right){
24        mid=(left+right)/2;
25        // arr의 주소=mid*sizeof(int)의 주소에 있는 값이 찾으려는 값과 비교
26        if(arr[mid]==search) //찾을 경우
27            break;
28        else if(arr[mid]<search) //더할 경우
29            left=mid+1;
30        else
31            right=mid;
32    }
33
34    if(left<right)
35        printf("%d\n", mid+1); //문제 예제에서 index 번호를 0부터 시작이 아니라 1부터 시작으로 되어있는것과 mid+1 출력.
36
37    7
38    17 28 43 67 88 92 100 43
39    실패
40    Program ended with exit code: 0
```

```
1 // 코드 작성 환경 정보
2 // OS : macOS (Big Sur) 11.5.2
3 // IDE 환경 : Xcode 12.5.1
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int main(){
9     int num; //배열의 원소 값
10    scanf("%d", &num);
11
12    int *arr=(int*)malloc(sizeof(int)*num); //정렬된 원소들을 읽어올 배열을 동적할당.
13
14    for(int i=0;i<num;i++){
15        scanf("%d", &arr[i]);
16    }
17
18    int search; //찾으려는 원소의 값
19    scanf("%d", &search);
20
21    int left=0, right=num-1; // 왼쪽 끝의 인덱스 번호, 오른쪽 끝의 인덱스 번호.
22    int mid=0; //중앙 인덱스 번호
23    while(left<right){
24        mid=(left+right)/2;
25        // arr의 주소=mid*sizeof(int)의 주소에 있는 값이 찾으려는 값과 비교
26        if(arr[mid]==search) //찾을 경우
27            break;
28        else if(arr[mid]<search) //더할 경우
29            left=mid+1;
30        else
31            right=mid;
32    }
33
34    if(left<right)
35        printf("%d\n", mid+1); //문제 예제에서 index 번호를 0부터 시작이 아니라 1부터 시작으로 되어있는것과 mid+1 출력.
36
37    8
38    3 7 10 23 48 50 67 90 21
39    실패
40    Program ended with exit code: 0
```

위의 두 사진과 같이 문제 명세에서의 예제 1번 (왼쪽 사진) 의 결과값인 3과, 예제 2번 (오른쪽 사진)의 결과값인 "실패"가 정상적으로 출력되었다.

2. 재귀함수를 사용한 이진 탐색



```
1 // 코드 작성 환경 정보
2 // OS : macOS (Big Sur) 11.5.2
3 // IDE 환경 : Xcode 12.5.1
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int binarySearch(int, int, int*, int);
9
10 int main(){
11     int num; //배열의 원소 값
12     scanf("%d", &num);
13
14     int *arr=(int*)malloc(sizeof(int)*num); //정렬된 원소들을 읽어올 배열을 동적할당.
15
16     for(int i=0;i<num;i++){
17         scanf("%d", &arr[i]);
18     }
19
20     int search; //찾으려는 원소의 값
21     scanf("%d", &search);
22
23     int left=0, right=num-1; // 왼쪽 끝의 인덱스 번호, 오른쪽 끝의 인덱스 번호.
24
25     /* 만약 res 변수의 값이 배열의 원소 값 수 이하일 경우 배열에 원소가 있는 것이므로 인덱스 번호 +1을 출력,
26        그렇지 않을 경우 배열에 원소가 없다고 판단, "실패"를 출력함.*/
27
28     int ans=binarySearch(left, right, arr, search);
29
30     if(ans!=-1)
31         printf("%d\n", ans+1);
32     else
33         printf("실패\n");
34     return 0;
35 }
36
37 7
38 17 28 43 67 88 92 100
39 43
40 3
41 실패
42 Program ended with exit code: 0
```

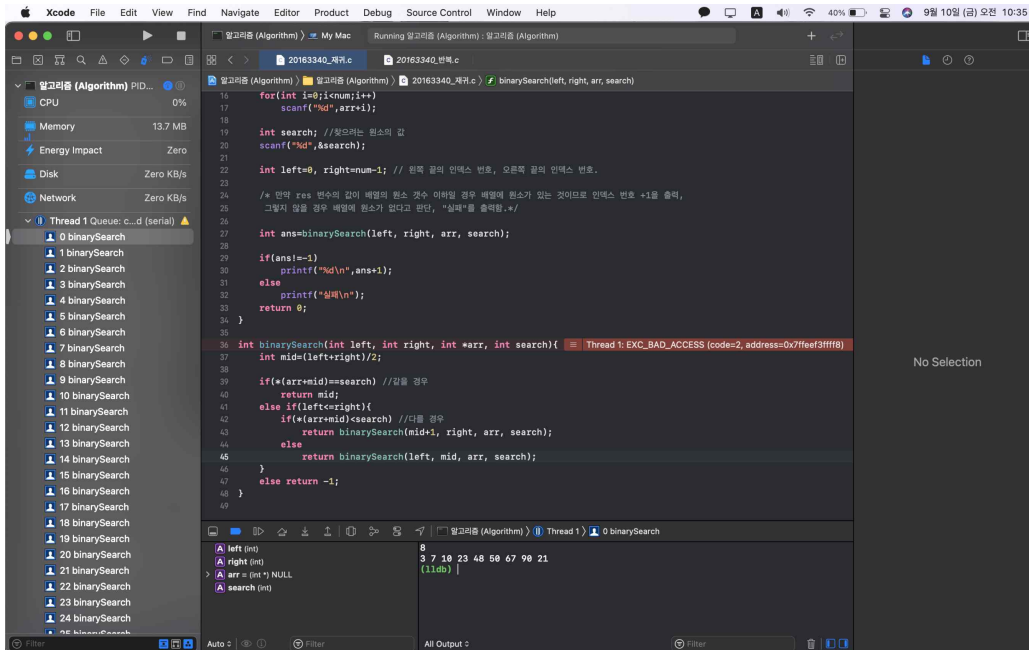
```
1 // 코드 작성 환경 정보
2 // OS : macOS (Big Sur) 11.5.2
3 // IDE 환경 : Xcode 12.5.1
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 int binarySearch(int, int, int*, int);
9
10 int main(){
11     int num; //배열의 원소 값
12     scanf("%d", &num);
13
14     int *arr=(int*)malloc(sizeof(int)*num); //정렬된 원소들을 읽어올 배열을 동적할당.
15
16     for(int i=0;i<num;i++){
17         scanf("%d", &arr[i]);
18     }
19
20     int search; //찾으려는 원소의 값
21     scanf("%d", &search);
22
23     int left=0, right=num-1; // 왼쪽 끝의 인덱스 번호, 오른쪽 끝의 인덱스 번호.
24
25     /* 만약 res 변수의 값이 배열의 원소 값 수 이하일 경우 배열에 원소가 있는 것이므로 인덱스 번호 +1을 출력,
26        그렇지 않을 경우 배열에 원소가 없다고 판단, "실패"를 출력함.*/
27
28     int ans=binarySearch(left, right, arr, search);
29
30     if(ans!=-1)
31         printf("%d\n", ans+1);
32     else
33         printf("실패\n");
34     return 0;
35 }
36
37 8
38 3 7 10 23 48 50 67 90 21
39 실패
40 Program ended with exit code: 0
```

재귀함수를 이용한 프로그램 역시 문제의 명세에 주어진 예제 입력 출력이 모두 정상적으로 이뤄지는 것을 확인할 수 있다.

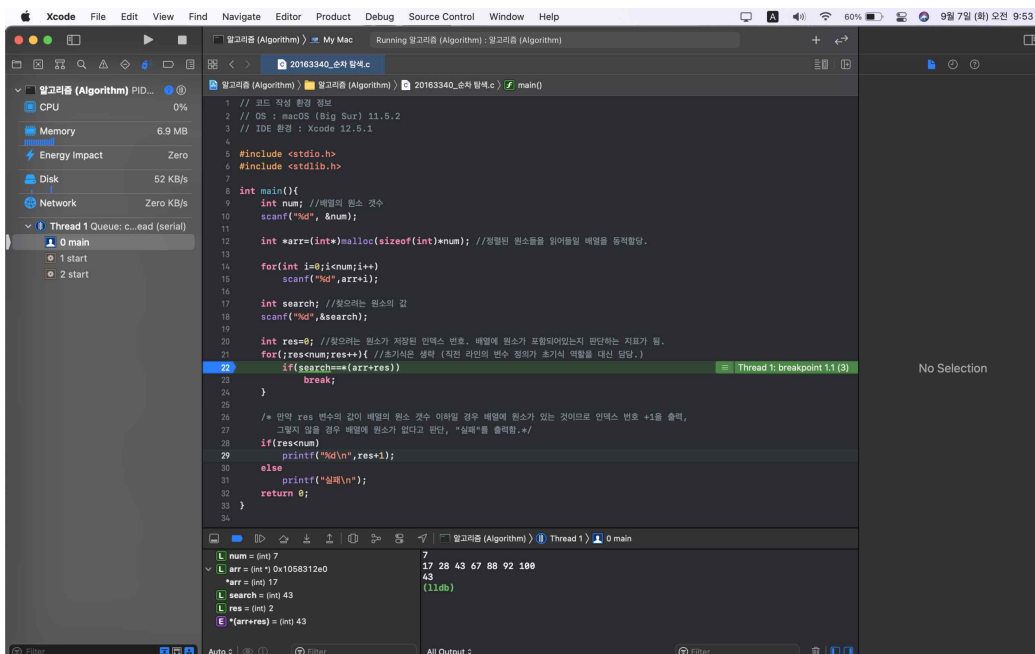
4장. 디버깅

Xcode의 디버거는 비주얼 스튜디오와 상당히 비슷하게 생겼지만, 매우 직관적이라는 느낌을 받았다. 왼쪽의 사이드바에서 지금 프로그램이 이 컴퓨터의 리소스를 얼마나 사용하는지 간략하게 보여주고, 클릭을 하게 되면 각각의 리소스를 표현하는데 적합한 그래프 (꺾은선 그래프, 원형 그래프 등)를 통해 나타내 주는 것이 인상깊었다.

또한, 아래의 사진과 같이 재귀함수의 수행 조건문을 잘못 설정하여 무한 루프가 도는 것을 알 수 있는데, 단순히 그 위치에 "!"를 출력하는 Visual Studio와는 달리 간략하게나마 에러 코드를 텍스트 영역에 표시해준다는 것이 IntelliJ를 쓰는 느낌이었고, 프로그램을 작성하는 데 있어서 편리했다.



(재귀 함수의 수행 조건문을 잘못 설정하여 무한 stack에 빠진 것을 디버깅 하는 과정.)



(중단점+표현식을 이용해 반복문 방식의 이진 탐색이 정상적으로 수행되고 있는지 확인하는 과정.)