

## 자료구조 실습과제 18

- 솔루션 및 프로젝트 명칭 : Proj\_18\_이름이니셜
- 제출방법 : 아래 문제를 해결하기 위한 프로그램을 구현한 후 컴파일 및 실행한 후, 오류가 없으면 메뉴에서 솔루션 정리를 수행한 후 윈도우 탐색기에서 솔루션 폴더를 찾아 압축하여 E-class에 올림

각 문제를 테스트하기 위하여 아래와 같이 구현하시오.

```
#define _CRT_SECURE_NO_WARNINGS

#define PROB 1 // 각각의 문제를 구현하고 해당 문제 번호를 변경하여 테스트

#if PROB == 1
// 1번 문제를 해결하기 위한 프로그램 작성

#elif PROB == 2
// 2번 문제를 해결하기 위한 프로그램 작성

#endif
```

### 문제 1) 정렬된 데이터에 대하여 순차탐색, 이진탐색, 보간탐색의 성능 분석

100만개의 정수를 대상으로 오름차순으로 정렬되어 있는 데이터에서 원하는 DATA를 검색하는 프로그램을 작성하시오. 순차탐색, 이진탐색, 보간탐색 기법의 함수를 구현하고, 아래 main() 함수에서 데이터 초기화 및 해당 함수를 호출하여 검색하는데 총 비교하는 횟수, 함수를 수행하는 데 걸리는 시간, 검색 결과를 실행 예와 같이 출력하는 프로그램을 작성하시오.

```
#define SIZE    1000000 // 저장된 데이터의 갯수
#define DATA    802    // 찾는 키 값
int main(void)
{
    int i, s_time, e_time, count, result;
    int* list = (int*)malloc(sizeof(int) * SIZE);
    if (list == NULL) { printf("메모리 할당 오류 \n"); exit(-1); }

    // 오름차순으로 정수 데이터 생성 및 저장
    list[0] = 0;
    for (i = 1; i < SIZE; i++)
        list[i] = list[i-1] + (rand() % 3);

    printf("Data 크기 : %d, 검색 데이터 : %d \n", SIZE, DATA);
    printf("-----\n");

    s_time = clock();
    result = seqsearch(list, 0, SIZE-1, DATA, &count); //순차탐색 호출
    e_time = clock();
    if (result == -1) printf("검색 결과 : 검색 데이터 없음 \n");
    else printf("검색결과 : %d 위치에서 찾음 \n", result);
    printf("순차탐색 소요시간 %d, 비교횟수 : %d\n\n", e_time - s_time, count);

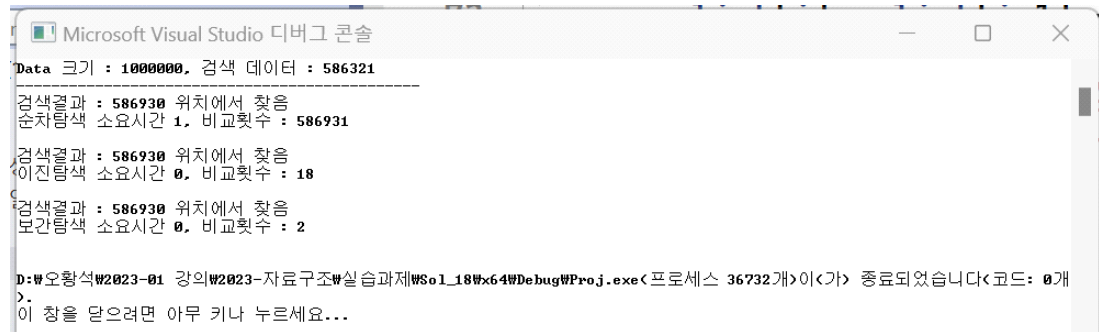
    s_time = clock();
    result = binsearch(list, 0, SIZE - 1, DATA, &count); //이진탐색 호출
    e_time = clock();
    if (result == -1) printf("검색 결과 : 검색 데이터 없음 \n");
    else printf("검색결과 : %d 위치에서 찾음 \n", result);
    printf("이진탐색 소요시간 %d, 비교횟수 : %d\n\n", e_time - s_time, count);

    s_time = clock();
    result = search_interpolation(list, 0, SIZE - 1, DATA, &count); //보간탐색 호출
    e_time = clock();
    if (result == -1) printf("검색 결과 : 검색 데이터 없음 \n");
    else printf("검색결과 : %d 위치에서 찾음 \n", result);
    printf("보간탐색 소요시간 %d, 비교횟수 : %d\n\n", e_time - s_time, count);

    free(list);

    return 0;
}
```

실행 예 : (검색 성공 시)

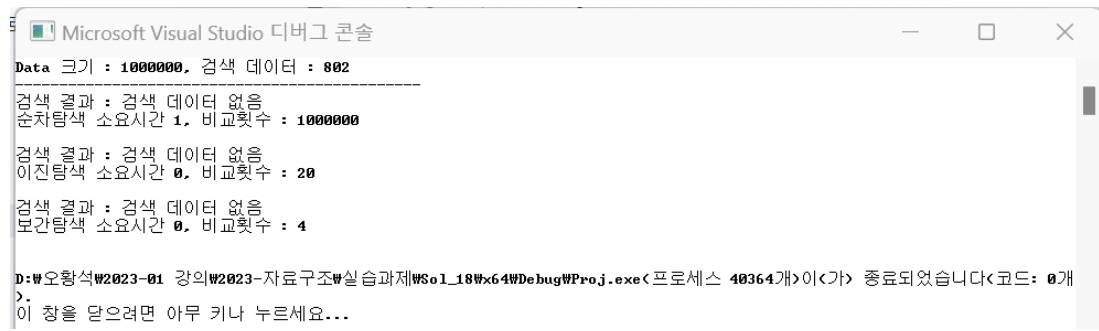


```
Microsoft Visual Studio 디버그 콘솔

Data 크기 : 1000000, 검색 데이터 : 586321
-----
검색결과 : 586930 위치에서 찾음
순차탐색 소요시간 1, 비교횟수 : 586931
검색결과 : 586930 위치에서 찾음
이진탐색 소요시간 0, 비교횟수 : 18
검색결과 : 586930 위치에서 찾음
보간탐색 소요시간 0, 비교횟수 : 2

D:\보오황석\W2023-01 강의\W2023-자료구조\실습과제\WSo1_18\Wx64\Debug\WProj.exe <프로세스 36732개>이<가> 종료되었습니다<코드: 0개>.
이 창을 닫으려면 아무 키나 누르세요...
```

실행 예 : (검색 실패 시)



```
Microsoft Visual Studio 디버그 콘솔

Data 크기 : 1000000, 검색 데이터 : 802
-----
검색 결과 : 검색 데이터 없음
순차탐색 소요시간 1, 비교횟수 : 1000000
검색 결과 : 검색 데이터 없음
이진탐색 소요시간 0, 비교횟수 : 20
검색 결과 : 검색 데이터 없음
보간탐색 소요시간 0, 비교횟수 : 4

D:\보오황석\W2023-01 강의\W2023-자료구조\실습과제\WSo1_18\Wx64\Debug\WProj.exe <프로세스 40364개>이<가> 종료되었습니다<코드: 0개>.
이 창을 닫으려면 아무 키나 누르세요...
```

## 문제 2) 균형잡힌 이진탐색트리(AVL) 구현

이진탐색트리(BST)의 단점인 트리의 깊이가 크게 차이나는 문제를 해결하기 위해 균형잡힌 균형잡힌 이진탐색트리(AVL 트리)를 구현하는 프로그램을 작성하시오. AVL에서 데이터를 검색하는 과정은 BST와 동일하나 데이터를 삽입하거나 삭제할 경우 새로운 노드 추가 또는 삭제 후에도 AVL의 조건을 만족하도록 트리를 조정해야 한다. 교재에서 소개된 LL, RR, LR, RL 방법을 적용하여 정수를 대상으로 데이터를 추가하거나 삭제할 경우 균형잡힌 이진탐색트리를 유지하도록 프로그램 하시오. 결과를 보이기 위해 트리에서 학습한 레벨 단위로 출력하는 level order 알고리즘을 활용하여 삽입/삭제 후 각 레벨에 있는 노드 값을 출력하시오. 아래 main 함수와 실행 결과를 참고하시오.

```
// AVL 트리 노드 정의
typedef struct AVLNode
{
    int key;
    struct AVLNode* left;
    struct AVLNode* right;
} AVLNode;

// 테스트를 위한 main 함수
int main(void)
{
    AVLNode* root = NULL;
    // 60 50 20 80 90 70 55 10 40 35
    // 예제 트리 구축
    printf("Insert %d\n", 60); root = AVL_insert(root, 60);
    printf("Insert %d\n", 50); root = AVL_insert(root, 50);
    printf("Insert %d\n", 20); root = AVL_insert(root, 20);
    level_order(root);
    printf("Insert %d\n", 80); root = AVL_insert(root, 80);
    level_order(root);
    printf("Insert %d\n", 90); root = AVL_insert(root, 90);
    level_order(root);
    printf("Insert %d\n", 70); root = AVL_insert(root, 70);
    level_order(root);
    printf("Insert %d\n", 55); root = AVL_insert(root, 55);
    level_order(root);
    printf("Insert %d\n", 10); root = AVL_insert(root, 10);
    level_order(root);
    printf("Insert %d\n", 40); root = AVL_insert(root, 40);
    level_order(root);
    printf("Insert %d\n", 35); root = AVL_insert(root, 35);
    level_order(root);
    printf("Remove %d\n", 50); root = AVL_remove(root, 50);
    level_order(root);
    printf("Remove %d\n", 55); root = AVL_remove(root, 55);
    level_order(root);
    return 0;
}
```

실행 예 :

```
Microsoft Visual Studio 디버그 콘솔

Insert 60
Insert 50
Insert 20
LL : 60
Levle Print
Levle 1 : [50],
Levle 2 : [20], [60],

Insert 80
Levle Print
Levle 1 : [50],
Levle 2 : [20], [60],
Levle 3 : [80],

Insert 90
RR : 60
Levle Print
Levle 1 : [50],
Levle 2 : [20], [80],
Levle 3 : [60], [90],

Insert 70
RL : 50
Levle Print
Levle 1 : [60],
Levle 2 : [50], [80],
Levle 3 : [20], [70], [90],

Insert 55
Levle Print
Levle 1 : [60],
Levle 2 : [50], [80],
Levle 3 : [20], [55], [70], [90],

Insert 10
Levle Print
Levle 1 : [60],
Levle 2 : [50], [80],
Levle 3 : [20], [55], [70], [90],
Levle 4 : [10],

Insert 40
Levle Print
Levle 1 : [60],
Levle 2 : [50], [80],
Levle 3 : [20], [55], [70], [90],
Levle 4 : [10], [40],

Insert 35
LR : 50
Levle Print
Levle 1 : [60],
Levle 2 : [40], [80],
Levle 3 : [20], [50], [70], [90],
Levle 4 : [10], [35], [55],

Remove 50
Levle Print
Levle 1 : [60],
Levle 2 : [40], [80],
Levle 3 : [20], [55], [70], [90],
Levle 4 : [10], [35],

Remove 55
LL : 40
Levle Print
Levle 1 : [60],
Levle 2 : [20], [80],
Levle 3 : [10], [40], [70], [90],
Levle 4 : [35],

D:\오형석\2023-01 강의\2023-자료구조\실습과제\So1_18\64\Debug\Proj.exe <프로세스 10288 개> 이 <가> 종료되었습니다 <코드: 0 개>.
이 창을 닫으려면 아무 키나 누르세요...
```