

정렬된 배열 `data`에 저장된 값들 중에서 특정값을 검색하는 프로그램을 작성한다.

- 함수 `main`

- 배열 `data`의 초기값은 {12, 18, 24, 30, 43, 52, 65, 76, 88, 91}
- 함수 `print_array`를 이용하고 실행 예제를 참조해 배열 `data`를 출력
- 무한 루프로 아래 작업들을 반복
- + 검색하는 값을 저장하는 변수로 `value`를 이용
- + 검색하는 값을 실행 예제와 같이 입력 받음
- + 입력 받은 값이 -1이면 루프를 빠져나옴
- + 그렇지 않으면

* 함수 `search`를 이용해 검색하는 값의 배열내 인덱스를 변수 `index`에 저장

* 실행 예제를 참조해 변수 `index`의 값이 -1이 아니면 배열내 위치를 출력하고 -1이면 배열내 없다고 출력

- 함수 `void print_array(int *arr, int size)`

- 배열 `arr`의 요소들을 실행 예제와 같이 출력
- 배열의 요소들은 포인터 주소 방식으로 Access
- 위 문제 2에서 작성한 함수 `print_array`를 이용해도 좋음

- 함수 `int search(int *arr, int size, int x)`

- 정수 `x`가 배열 `arr`에서 발견되면 배열내 인덱스를 반환
- 정수 `x`가 배열 `arr`에서 발견되지 않으면 -1을 반환
- 배열의 요소들은 배열의 인덱스를 이용하는 방식으로 Access
- Binary search 알고리즘을 이용

- 실행 예제

Array data: 12 18 24 30 43 52 65 76 88 91

Enter an integer to search (-1 to quit): 24

24 is at index 2.

Enter an integer to search (-1 to quit): 91

91 is at index 9.

Enter an integer to search (-1 to quit): 75

75 is not in the array.

Enter an integer to search (-1 to quit): -1

`print_array` 함수는 출력할때 인덱스를 사용하지 않고 포인터의 주소값을 늘려가면 출력해줘



아래는 주어진 요구사항에 따라 작성된 C 프로그램의 예제 코드입니다. `print_array` 함수는 배열의 요소들을 포인터 주소 방식으로 출력하며, `search` 함수는 이진 검색 알고리즘을 사용하여 값을 검색합니다.



c

Copy code

```
#include <stdio.h>

void print_array(int *arr, int size);
int search(int *arr, int size, int x);

int main() {
    int data[] = {12, 18, 24, 30, 43, 52, 65, 76, 88, 91};
    int size = sizeof(data) / sizeof(data[0]);

    printf("Array data: ");
    print_array(data, size);

    while (1) {
        int value;
        printf("Enter an integer to search (-1 to quit): ");
        scanf("%d", &value);

        if (value == -1) {
            break;
        } else {
            int index = search(data, size, value);
            if (index != -1) {
                printf("%d is at index %d.\n", value, index);
            } else {
                printf("%d is not in the array.\n", value);
            }
        }
    }
}
```

```

    return 0;
}

void print_array(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", *arr);
        arr++; // 포인터 주소를 증가시켜 다음 요소로 이동
    }
    printf("\n");
}

int search(int *arr, int size, int x) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;


        if (arr[mid] == x) {
            return mid; // 발견된 경우 인덱스를 반환
        }

        if (arr[mid] < x) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1; // 발견되지 않은 경우 -1을 반환
}

```

이 코드는 주어진 요구사항에 따라 작성되었으며, `print_array` 함수는 요구사항에 따라 포인터 주소 방식으로 배열 요소를 출력합니다. `search` 함수는 이진 검색 알고리즘을 사용하여 값을 검색하고, 결과를 출력합니다.

 Regenerate