Day16

태그	
날짜	@2022년 10월 21일

Algorithm

알고리즘은 문제를 해결하기 위한 절차

알고리즘은 비단 컴퓨터에서만 사용되는 용어가 아니라 음악의 악보, 요리의 레시피, 가전제품 사용 설명서 등은 사용방법 등이 상세하게 절차적으로 적혀있다.

알고리즘을 프로그래밍 언어 (자바)로 작성한 것이 바로 프로그램이다.

컴퓨터에게 실행하기를 원하는 내용을 상세하게 분해하여 구체적인 방법을 지시하는 것이다.

ex)

청소기 로봇

전원 켠다. → 직진한다. → 장애물을 만나면 약간 옆으로 방향을 전환한다. → 직진한다. → 배터리가 부족하면 충전소로 이동한다. → 미 청소구역이 있으면 처음부터 다시 반복한다.

프로그램을 작성하는 흐름

1. 기획
2. 설계 - 이 단계에서 알고리즘이 필요하다.
3. 프로그래밍
4. 디버깅
5. 문서화

- 좋은 알고리즘이란?

    - 알기 쉽다. - 특히 여러 사람이 작업을 하게 되는 프로그래밍에서는 타인이 이해하기 어렵다면 많은 시간이 소모되고 실수를 범하기 쉽게 된다.
    - 속도가 빠르다. - 결과가 나타날때 까지의 시간이 짧다는 것이다.
    - 효율적이다. - 메모리를 적게 사용한다.
    - 재사용이 쉽다. - 한번 작성한 프로그램을 다른 프로그램을 만들 때도 사용할 수 있다면 작성 시간이 상당히 줄일 수 있게 된다.

- 왜 알고리즘을 공부해야 할까?
    - 좋은 알고리즘을 만들 수 있다.
    - 프로그램의 좋고 나쁨을 판단할 수 있게 된다.
    - 프로그램의 전체 과정을 효율화 할 수 있다.
    - 프로그래밍 기술을 향상 시킬 수 있다.

- 알고리즘의 조건
    - 정확한 결과를 얻을 수 있어야 한다.
    - 반드시 종료 되어야 한다.

- 알고리즘의 기본형 3가지

  아무리 복잡해 보이는 알고리즘도 기본형 3가지의 조합으로 구성 된다.

    1. 순차구조 : 일반적인 흐름
    2. 선택 구조 : if문
    3. 반복 구조 : while, for문

- 알고리즘을 표현하는 방법 2가지
    - 수도 코드 (pseudo code, 컴퓨터는 이해하지 못하는 컴퓨터언어, 의사 코드)
    - 흐름도 flow chart, 그림으로 표현하는 알고리즘

삼각형의 면적을 계산하는 알고리즘

`삼각형의 면적은 = 밑변 * 높이 * / 2`

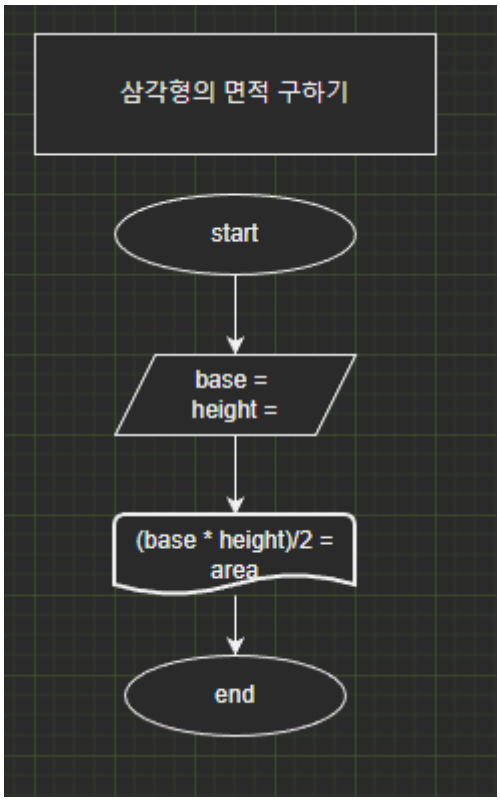
필요한 데이터는 밑변의 길이와 높이가 필요하다.

따라서 변수를 준비해서 계산

`area = base * height / 2`

밑변을 높이에 곱한 값을 2로 나누어 그 결과 값을 area 변수에 대입한다.

1. base와 height 을 입력한다.
2. base 와 height 곱한 값을 2로 나누어 그 결과를 area변수에 대입한다.
3. area 값을 출력한다.




```
package test;

public class algo {
```

```
  public static void main(String[] args) {
    // TODO Auto-generated method stub

    int base = 4;
    int height = 5;
    int area = 0;
    area = (base*height)/ 2;
    System.out.println(area);

  }

}
```

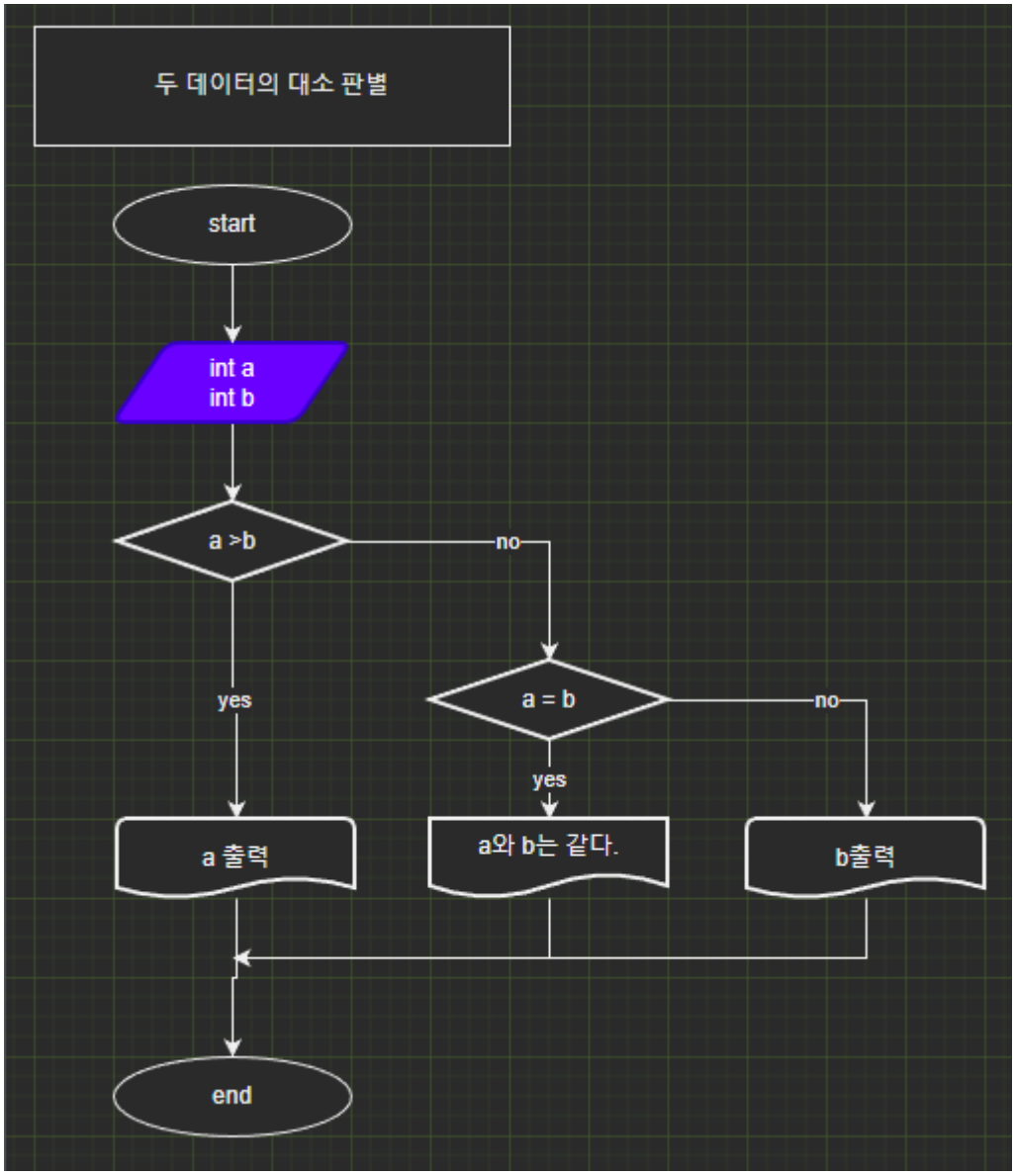
두 데이터의 대소 판별

2개의 데이터 중에서 큰 것은 어느 쪽일까

변수를 2개 준비한다. a와 b를 사용한다.

데이터를 비교하여 a가 크면 a를 출력 b가 크면 b를 출력한다.

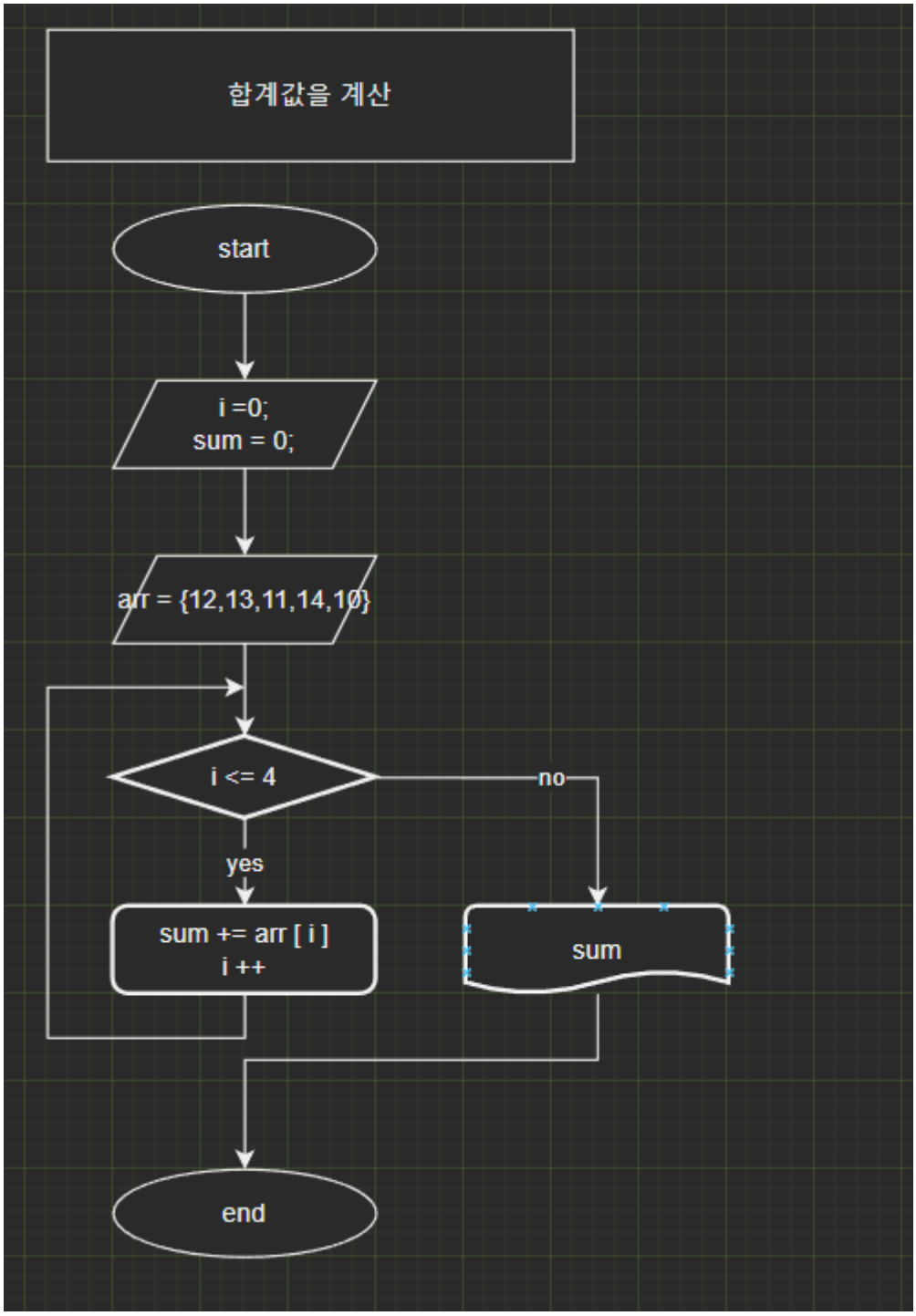
1. a, b 입력
2. a > b
    a. a 출력
3. a = b
    a. ‘a와 b는 같다’
4. a < b
    a. b 출력




```
package test;

public class aorb {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    int a = 6;
    int b = 9;

    if (a>b) {
      System.out.println("a는 b보다 크다");
    }else if(a==b) {
      System.out.println("a는 b와 같다.");
    }else {
      System.out.println("b는 a보다 크다");
    }
  }
}
```

합계값을 계산

여러 값의 함을 계산하려면 반복 구조를 사용해야 한다.




```
package test;

public class sumarr {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    int i = 0;
    int sum = 0;
```

```
    int[] arr = {12,13,11,14,15};

    for (i=0; i<=4; i++)
    if(i<=4) {
      sum += arr[i];

    }
    System.out.println(sum);
  }

}
```

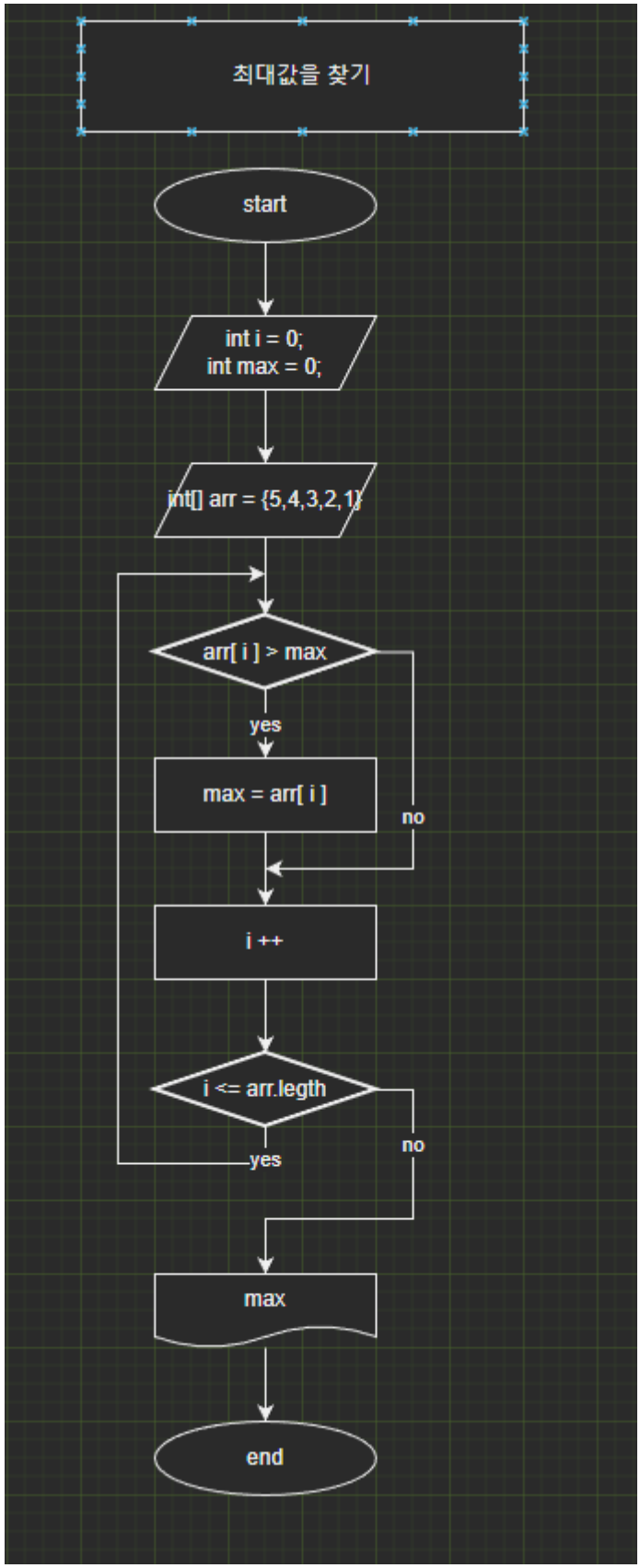
다른사람이 한 코드

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    //변수 준비

    int arr[]= {3,3,4,2};

    int sum =0;

    for(int i=0;i<arr.length;i++) {

       sum = sum+arr[i];
    }
    System.out.println(sum);

}
```

이 방법이 더 좋아보인다.

최대값을 찾는 알고리즘

예를 들어 5개의 데이터 중에서 제일 큰 값은 무엇일까.


최대값을 찾기
start
int i = 0;
int max = 0;
int[] arr = {5,4,3,2,1}
arr[i] > max
yes
max = arr[i]
no
i ++
i <= arr.legth
yes
no
max
end

알고리즘의 유형

- 탐색 알고리즘 : 선형탐색법, 이진탐색법, 해시탐색법
- 정렬 알고리즘 : 단순정렬, 단순 교환, 단순 삽입, 퀵
- 에라토스테네스의 체 알고리즘 : 소수 판별 알고리즘
- 유클리드 알고리즘 : 최대공약수 알고리즘

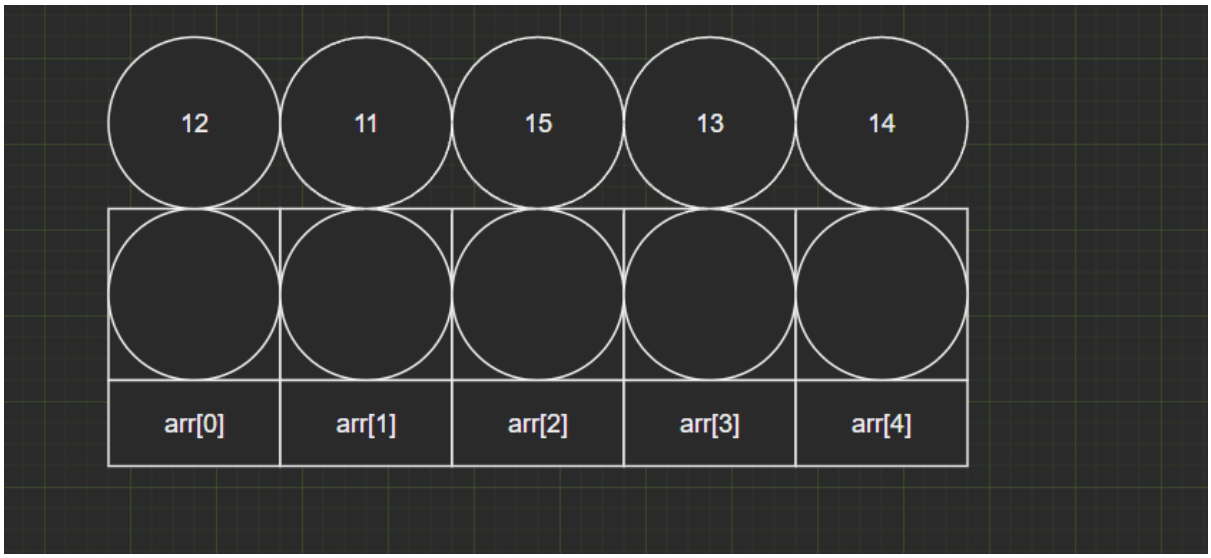
선형탐색법 : Linear Search

단순히 맨 앞에서부터 차례로 원하는 값을 찾는다.

알고리즘 자체가 아주 단순해서 이해하기 쉽다.

알고리즘의 성능은 그다지 좋지 못하다.

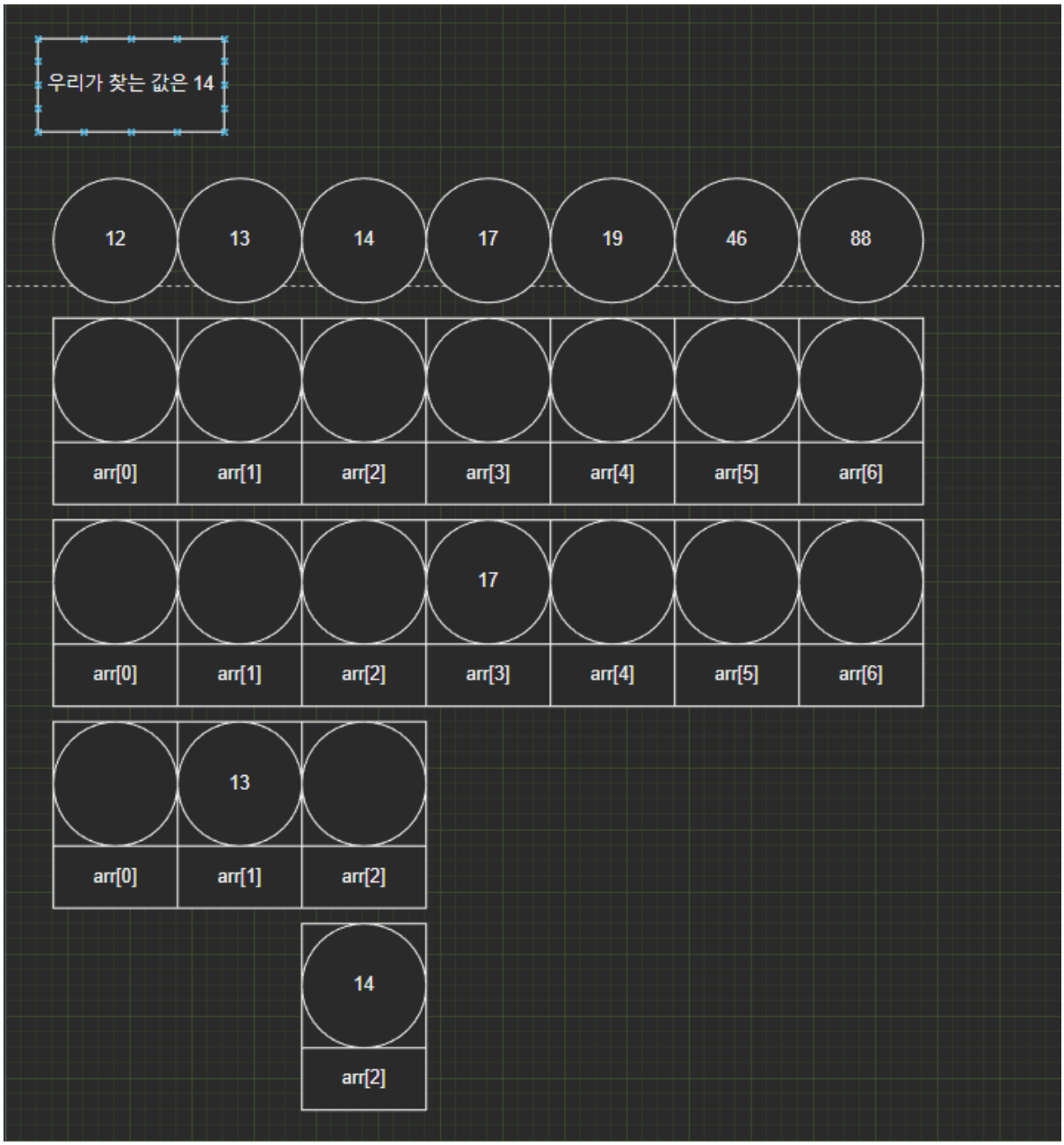
왼쪽 칸부터 원하는 값 5가 나올때 까지 차례로 하나씩 방문을 열어 확인한다.



이진 탐색법 (Binary Search)

이진 탐색법은 탐색의 범위를 한번 순환할 때마다 반으로 그 대상을 줄일 수 있기 때문에 빠른 시간에 원하는 값을 찾을 수 있는 알고리즘이다.

그러나, 탐색 대상 범위의 값들이 반드시 오름차순/내림차순 정렬된 상태 이어야만 이진 탐색법을 사용할 수 있다.



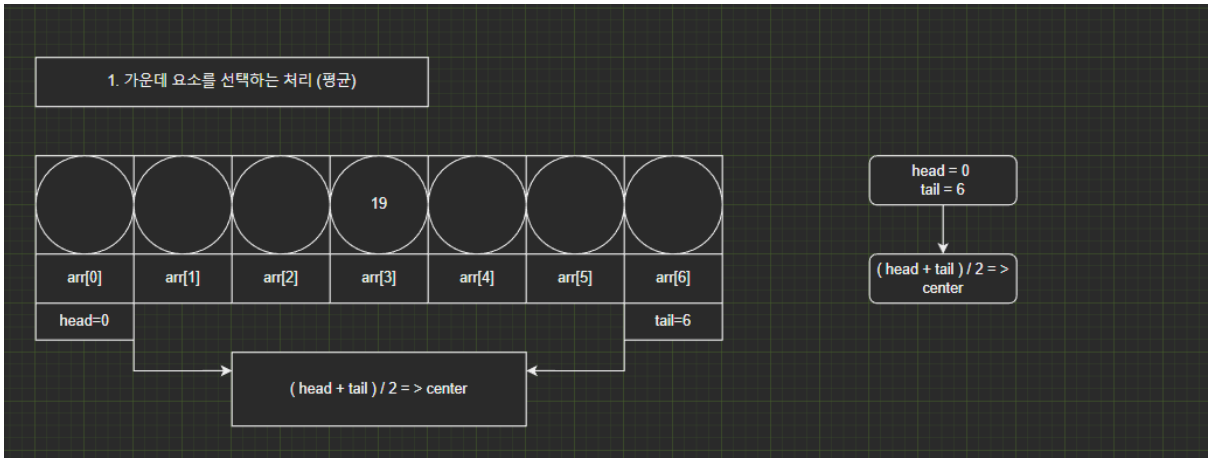
정 가운데의 데이터를 열어서 찾고 있는 값인지 비교해보고 크거나 작으면 , 앞 위쪽 대상 범위를 제거하고 해당 범위에서 다시 정 가운데 데이터를 열고 반복한다.

최종 남아있는 방을 확인한다. 이때 찾고 있는 값이 없을 수도 있다.

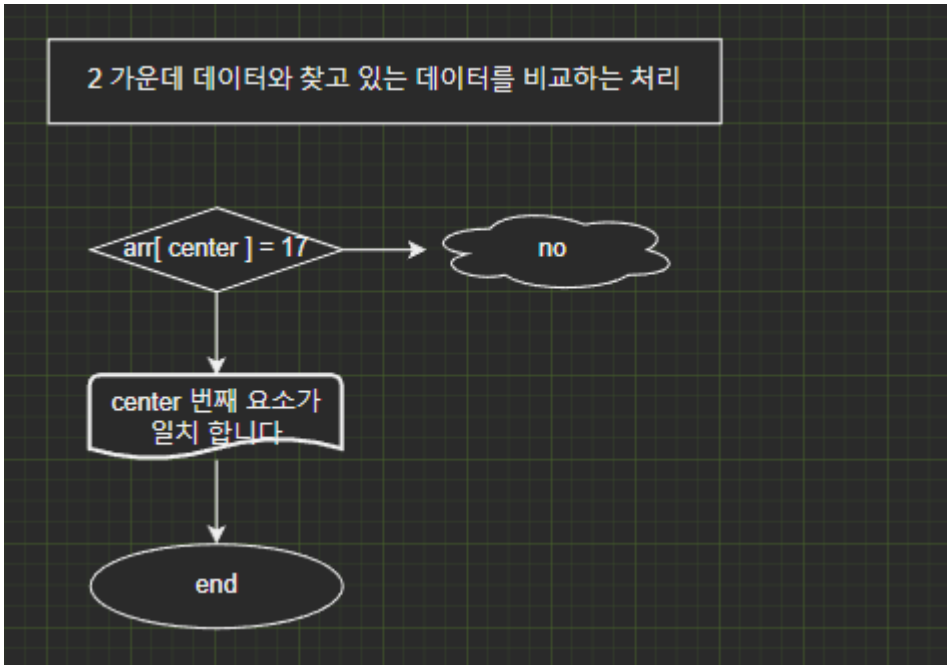
순서도로 표현하기

이진 탐색은 크게 세가지 로직을 가지게 된다.

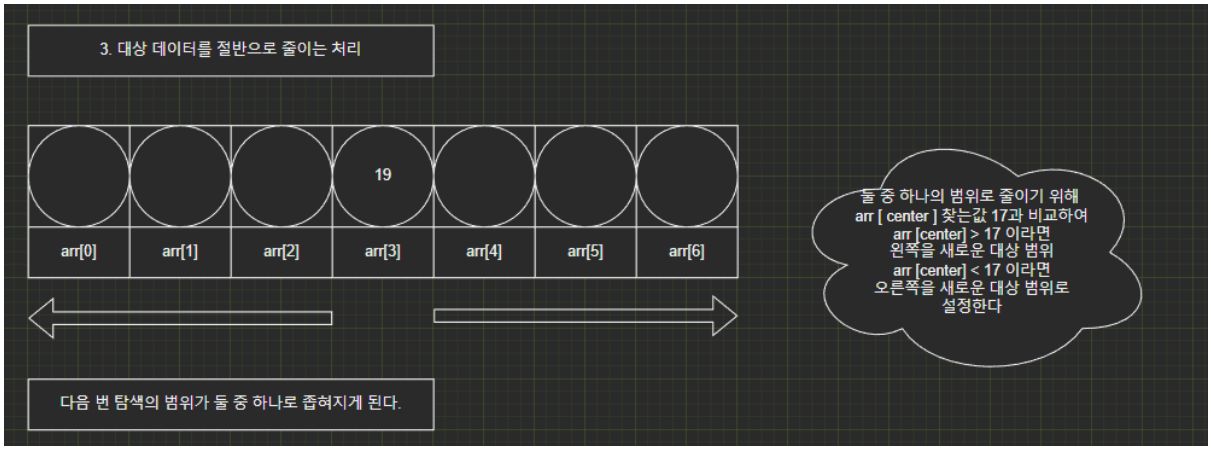
1. 가운데 요소를 선택하는 처리 (평균)
    - 범위의 가운데 요소를 선택하는 방법은 바로 평균을 이용한 것이다. 두 숫자의 가운데는 평균이다.
    - 예를 들어 0부터 10까지 중 가운데를 찾으려면
    - (0+10) /2 = 5
    - 그런데 만약 0 부터 11까지 중 가운데를 찾으려면 어떻게 해야할가
    - (0+11) /2 소수 이하는 버리고 정수 부분만 취하여 가운데 값으로 사용하게 된다. 따라서 소수 이하를 버린 5를 가운데 값으로 하여 처리하게 된다.

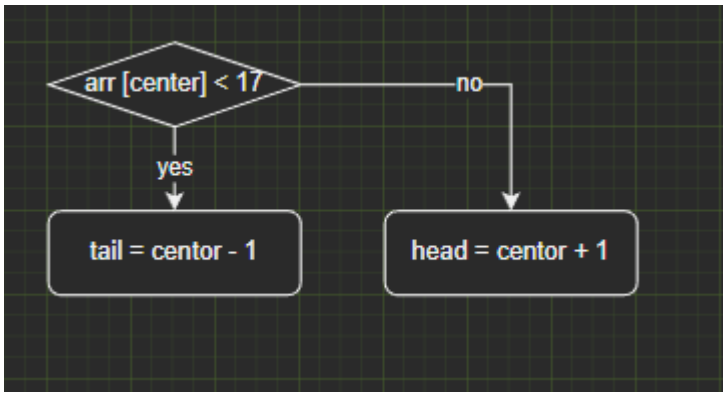


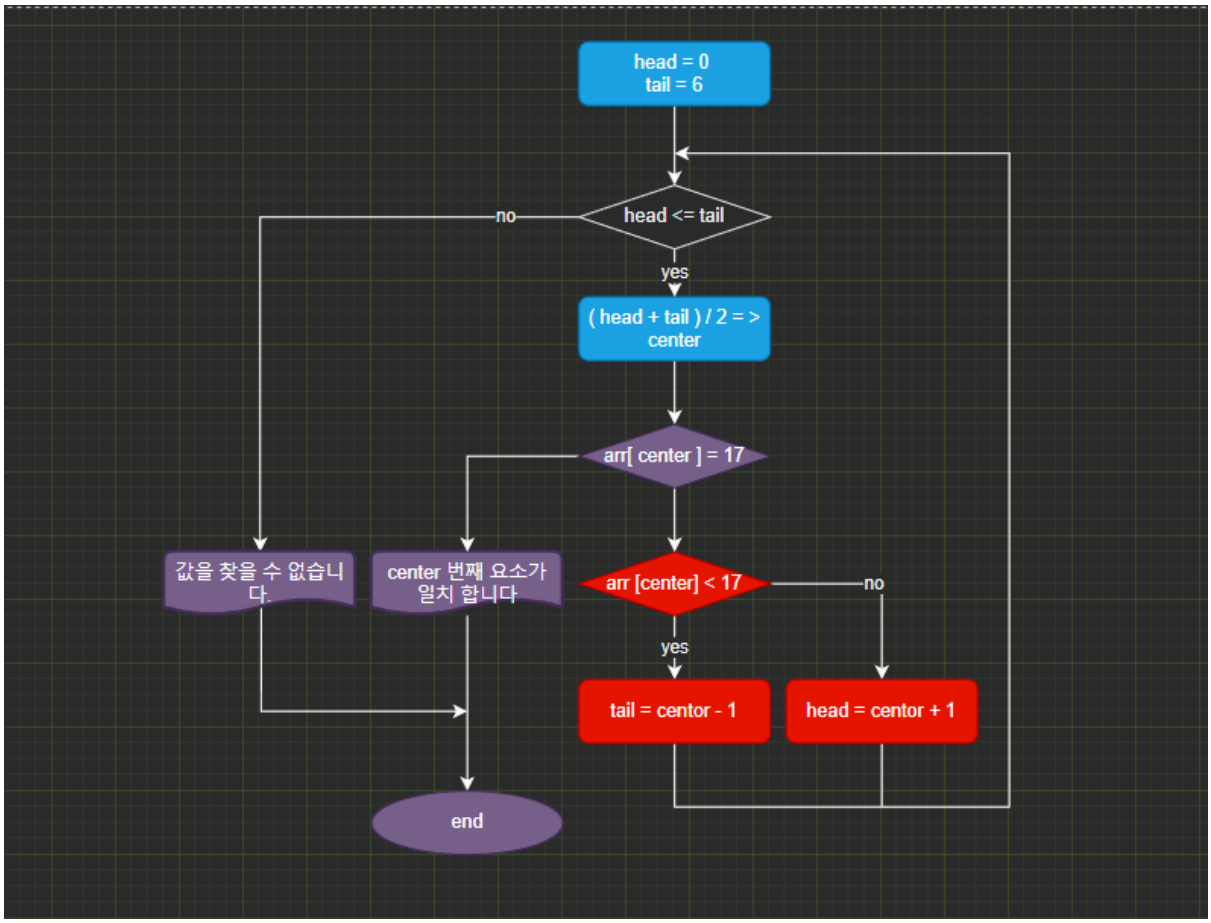
2. 가운데 데이터와 찾고 있는 데이터를 비교하는 처리



3. 대상 데이터를 절반으로 줄이는 처리






head = 0
tail = 6
head <= tail
no
yes
(head + tail) / 2 = > center
arr[center] = 17
값을 찾을 수 없습니다.
center 번째 요소가 일치 합니다
arr [center] < 17
no
yes
tail = centor - 1
head = centor + 1
end

반대로 생각해보기