코딩능력 : 언제나 중요

알고리즘 코딩 : 재사용 X But 그렇다고 스파게티 코드는 지양할 것

**읽기 쉬운 코드** ⟹ 프로그래밍 대회에서 좋은 성적을 거두기 위한 비결

간결하고 효율적인 프로그램을 작성하는 능력
↳ 알고리즘 대회의 가장 큰 소득 중 1개

알고리즘 대회에서 자주하는 실수 + 코딩, 디버깅에 관한 노하우

## 좋은 코드를 짜기 위한 원칙

### 1. 간결한 코드를 작성하기

짧을 수록 오타, 버그가 줄고 디버깅이 용이해진다.

대표적인 예시 : 전역변수의 광범위적인 사용 ✗

알고리즘 특성상 코드가 짧고 간결해 해당변수가 어디 사용되는지 명확히 알 수 있으므로 단점이 부각되지 않음

### 2. 적극적인 코드 재사용

→ 자주 사용하는 코드는 모듈화 할 것 . 버그발생시 디버깅에 큰도움

But 실무에서처럼 쓰기는 X

## 3. 표준 라이브러리 공부하기

큐, 스택같은 자료구조 , 정렬같은 기본적인 알고리즘

⇓

표준라이브러리에 있음

## 4. 항상 같은 형태로 프로그램을 작성하기

알고리즘 : 여러 종류의 코드를 반복적으로 짜게 된됨

[ex) BST, DFS, BFS, 좌표평면 구간 점검 확인 함수 .....]

자신이 가장 알아보고 사용하기 쉬운 코드를 정해 항상 그 방식으로 작성하기!!

앞기의 영역

## 5. 일관적이고 명료한 명명법 사용하기.

모호하지 않은 변수와 함수의 명명 + 되도록 표준 명명법을 따르기

## 6. 모든 자료를 정규화 하여 저장

각도, 분수 등의 자료 = 1가지 방식으로 정하고 사용

자료입력시, 계산시 즉시 정규화 할것

## 7. 코드와 데이터의 분리

코드의 논리와 상관없는 데이터는 분리하는 게 좋음

ex) 체스판위 나이트의 움직임

```
const knigthDx[8]={2,2,-2,-2,1,1,-1,-1}
const knigthDy[8]={1,-1 1,-1,2,-2,2,-2}
```

# 자주하는 실수

1. 오버플로
2. 배열 범위 밖 원소에 접근

   Array index out of bounds
3. 일관되지 않은 범위 표현 방식
4. off-by-one

   100미터 당장이 10미터 간격으로 설치하는 가로수 갯수

   답= 11개
5. 오타
6. 재귀함수 사용시 발생하는 스택 오버플로우
7. 고차원 배열의 인덱스 순서 바꿔 쓰기
8. 잘못된 비교함수(C++ 의여기 같음)
9. 최소, 최대 예외 잘못 다루기
10. 연산자의 우선순위
11. 너무느린 입출력 방식
12. 변수 초기화

## 디버깅과 테스팅

원하는 결과가 나오지 않았을 경우

→ assertion 사용(성질)

→ 직접 값 확인

테스팅에 유용한 기법

- 스캐폴딩: 임의의 테케를 만드는 코드