코딩 테스트

1억번의 연산 = 1초  
  
시간복잡도 유형  
최선의 빅-오메가 보통의 빅-세타 최악의 빅-오  
  
연산횟수 계산 방법  
연산횟수 = 알고리즘 시간 복잡도 n값에 데이터의 최대 크기를 대입하여 도출  
  
시간 복잡도를 바탕으로 코드 로직 개선하기  
- 시간 복잡도는 작성한 코드의 비효율적인 로직을 개선하는 바탕으로 사용 가능  
- 코드의 시간 복잡도를 도출해야 함  
  
시간 복잡도 도출 기준  
- 상수는 시간 복잡도 계산에서 제외한다   
– 가장 많이 중첩된 반복문의 수행 횟수가 시간 복잡도의 기준이 된다  
  
코드에서 실수 하기 쉬운 오류4가지  
1. 변수 초기화 오류  
2. 반복문에서 인덱스 범위 지정 오류 찾아보기  
3. 잘못된 변수 사용 오류 찾아보기  
4. **자료형 범위 오류 찾아보기(정답이 음수가 나온다면 범위초과를 의심해보쟈)  
(long이나 longlong을 사용)**

배열  
**-연속된** **메모리 공간**에 값이 채워져 있음  
**-인덱스**를 통해 **참조**할 수 있음  
**-배열의 크기**는 **선언**할 때 지정하며 **변경 불가**

리스트: 값괴 포인터를 묶은 노드를 포인터로 연결  
**-인덱스가 없음** Head부터 순서대로 접근(**접근** 속도가 **느림**)  
**-삽입/삭제**가 속도가 **빠름**-선언시에 **별도의 크기 지정 X**(**크기가 변하는 데이터**에 적절)  
-포인터를 저장할 공간이 필요하므로 구조가 복잡

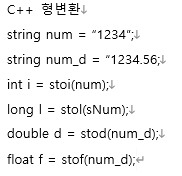
벡터: 동적배열  
- 동적으로 원소 추가 가능(**크기가 자동으로 변환**)  
- 양끝 데이터의 추가/삭제는 문제가 없지만 **중간값 조정**은 **배열과 같은** 매커니즘

백터 사용법  
-선언:   
std::vector<int> A;

-삽입 연산  
A.push\_back(1): 마지막에 추가  
A.insert(A.begin(),7): 맨앞에 7을 삽입  
A.insert(A.begin + 2, 10): index 2(= [2])에 10을 삽입

텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명-삭제 연산  
A.pop\_back(): 마지막 값 삭제  
A.erase(A.begin() + 3): index 3(=[3])의 값을 삭제  
A.clear(): 모든 값을 삭제



C++형변환

- 정보 가져오기  
A.size: 데이터 개수  
A.front(): 처음 값  
A.back(): 마지막 값  
A[3]: index 3의 값  
A.at(5): index 5의 값  
A.begin(): 첫번째 데이터 주소  
A.end(): 마지막 데이터 주소