**숙제 1**

1. hw1\_calc\_var2에서 경우에 따라 pow와 곱의 결과가 float의 범위를 넘어서 오버플로우를 발생하는 경우를 방지하기 위해 double의 자료형인 tmp를 사용해야한다.
2. 이 음수가 나올 수 있다. 따라서 본래 분산은 음수의 값이 나올 수 없지만 후자의 방식을 따를 경우 음수가 나올 수 있다.
3. 후자의 방식이 더 빠르다. 전자의 경우 매 데이터마다 평균을 감하여 합하지만, 후자의 경우에는 마지막 항은 평균 \* 데이터 개수에 제곱을 하면 되기 때문에 간편하다.

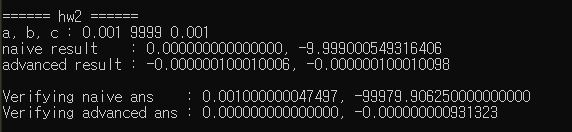
**숙제 2**

b의 값이 커서 에 잘못된 지워짐이 발생하면 오차가 예상보다 크게 발생하여 잘못된 값이 발생한다. 예를 들어 a = 1, b = 10000, c = 1와 같은 경우 근의 공식 적용에 부정확한 값을 가지도록 한다.

특히의 경우 이미 부동 소수점 곱셈을 수행하였기 때문에 반올림 오차가 반영된 상태이다. 만일 라서 에서 지워짐이 발생하지 않는다면, 단순히 라고 볼 수 있다. 이 경우 나쁜지워짐이 발생하게 된다.

1. 위와 같은 오류를 방지하고자 근의 공식의 분자 분모에 를 곱해서 다음 과 같은 근을 구해야 한다.





그리고 를 단독으로 구할 경우 또다시 비슷한 수의 뺄셈 오류를 발생시키기 때문에 을 이용하여 역수처리 한 후 다시 역수화하여 구하였다. 특히 b의 값이 클 경우에는 이 0이 나올 가능성이 있어 나눗셈에 문제가 발생할 수 있기 때문에 더욱 큰 주의가 필요했다.

**숙제 3**

1. 1번 : loop를 iterating을 증가하는 형식으로 하는 것과 감소하는 형식을 비교

2번 : if문 조건에 함수 호출하여 판별

3번 : else- if 시에 병렬형 혹은 중첩형으로 진행.

4번 : loop jamming

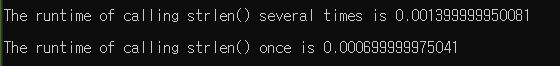
5번 : loop에서 조건 부합 시 break문 추가 유무

1번 :



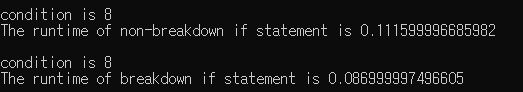
어셈블리에는 0과 비교하는 명령어가 별도로 존재해서 인덱스를 증가하며 특수한 수에 다다를 때까지 루프를 돌리는 것보다, 특수한 수에서 시작하여 0까지 도달하는지에 대해 판별하는 것이 필요하다.

2번 :



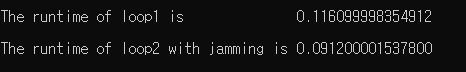
For loop의 조건으로 strlen()을 매번 호출하는 것보다 미리 별도의 변수에 strlen()의값을 저장한 뒤, 그 변수를 활용하여 for loop의 조건으로 써서 코드를 최적화하게 사용할 수 있다.

3번 :



입력에 따라 if -else if -else 처리를 할 시에 조건이 여러 개일 경우 모든 조건을 병렬적으로 처리하기 보다는 큰 분류로 나누어 판별하지 않아도 되는 조건문은 통과하지 않도록 한다.

4번 :



똑같은 조건에 대해 여러 루프를 쓰기 보다는 최대한 한 루프 내에서 처리를 해야 효율적이다.

5번 :



만약 조건에 부합하는 처리를 하고 그 뒤 별도 처리가 필요하지 않은 경우에는 break문을 써서 반복문을 나가서 불필요하게 반복하는 경우를 없앤다.