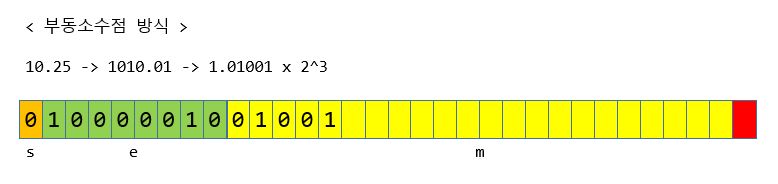
<고정소수점 방식>

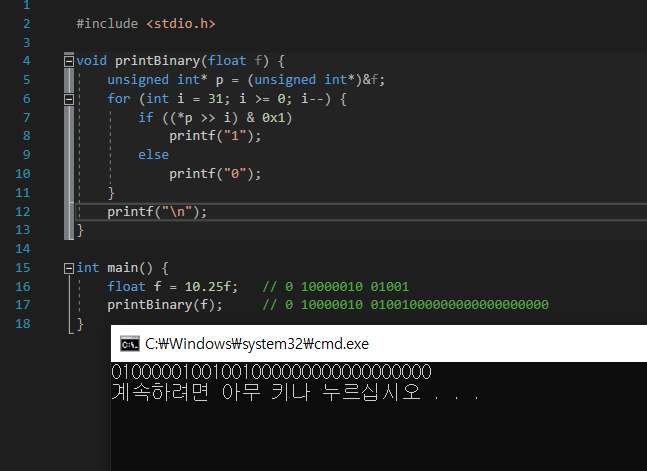
소수부분은 계속 x2를 하여 1이될때까지하여 2진수로 바꿀수 있다.

0.25 x 2 = 0.5 ‘0’ 0.5 x 2 = 1.0 ‘1’ 즉 01 이된다.

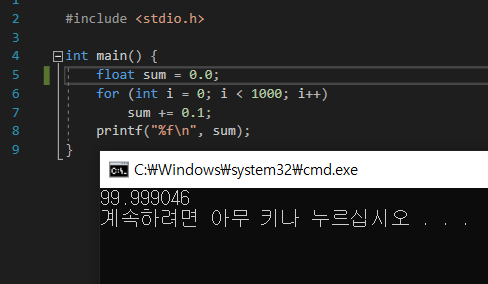
정수부분만 크기에 해당되므로 실수를 사용한다고해서 매우 큰 수를 사용한다고 볼수없다.

또한 비트의 수가 많은것도 아니므로 매우 정밀한 수를 표현한다고 볼수도 없다.

<부동소수점 방식>



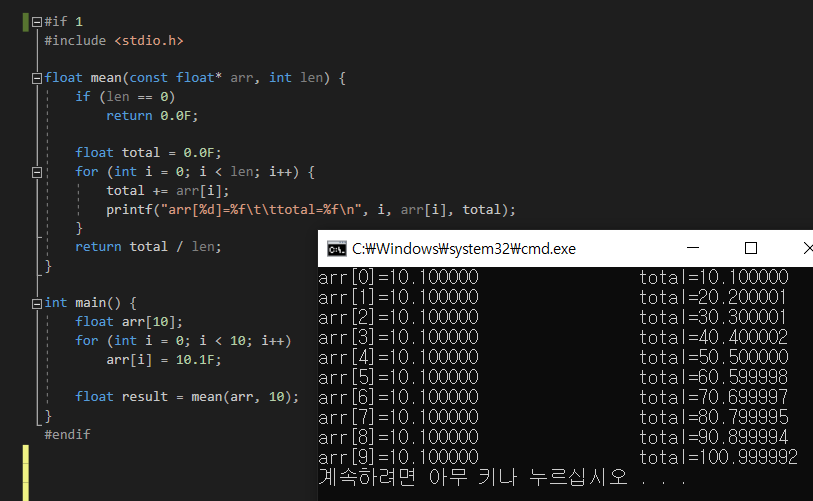
부동소수점을 통해 float형을 2진수로 바꾸는 코드



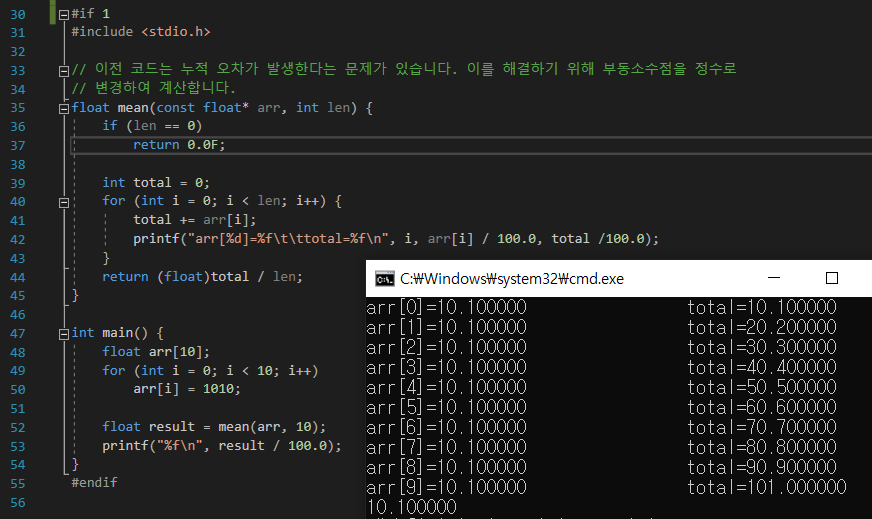
부동소수점은 매우 큰수/정밀한수를 표현하기위한것이지, 계산을 위한 것이 아니므로

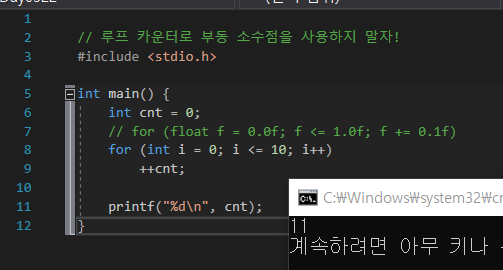
위 코드처럼 99.99~~처럼 정확하지 않은 수가 나오는 것 즉 계산을 위한 것이 아니다.

위 코드를 해결하기 위한 것으로는 float대신에 double을 쓰면된다. 그러나 1000을 돌리는게 아니라 엄청 큰 수의 루프를 돌리면 이것또한 오차가 생긴다.. 그래서 해결책은 부동소수를 정수로 변환한다음 계산후 다시 부동소수로 변환하는 방식을 사용하면된다.



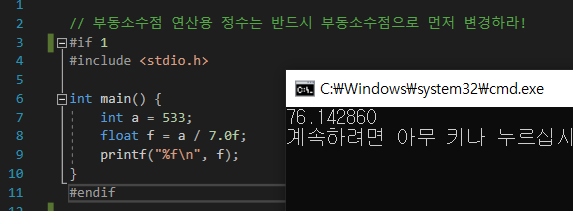
이 코드는 누적 오차가 발생한다.

이전 코드의 문제를 해결하기 위해 부동소수점을 **‘정수로 변경’**하여 계산한다.

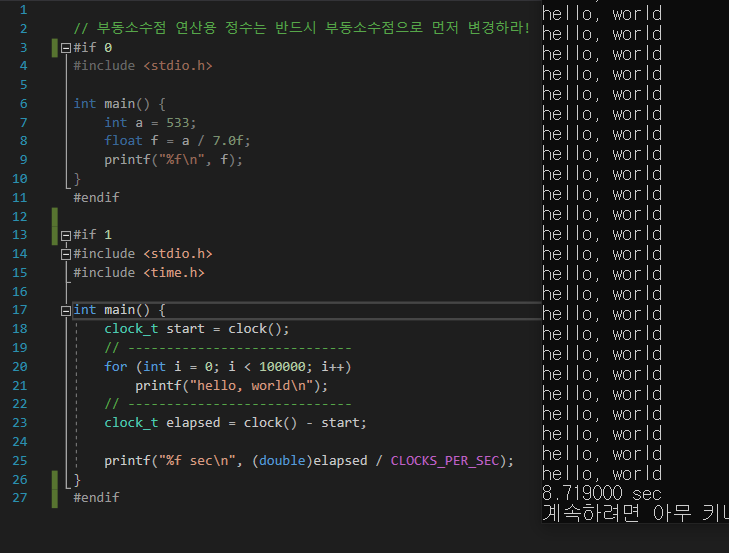


루프카운트로 부동소수점을 사용하지말자! -> float(부동소수점) 사용시 -> 10번

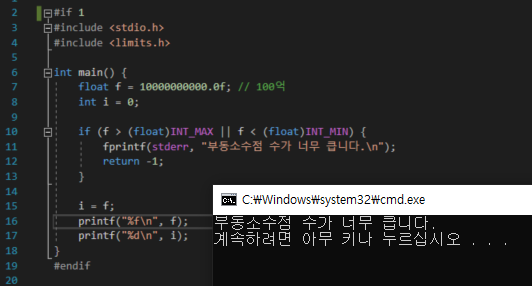
int 사용시 -> 11번(정상)



부동소수점 연산용 정수는 반드시 부동소수점으로 먼저 형변환후 계산하라

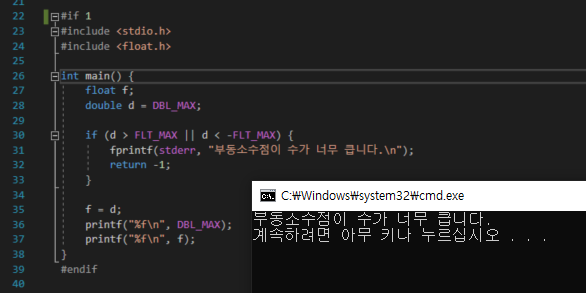


start에서 end까지 clock의수가 계속 증가하는데 그 개수를 clock\_per\_sec으로 나눠주면 초가 나온다. 나누기 할 때 둘다 정수형이므로 데이터가 사라지므로 이를 (double)로 변환하여 계산한다.



int의 범위를 넘는 float가 있을 때(100억) 그것을 int에 넣기위한 조건문 사용

정수의 max와 min을 알기위해 limits.h 를 사용

double의 min max와 float의 max min을 알기위해 float.h를 사용하여 조건문을 통해 걸러야한다.