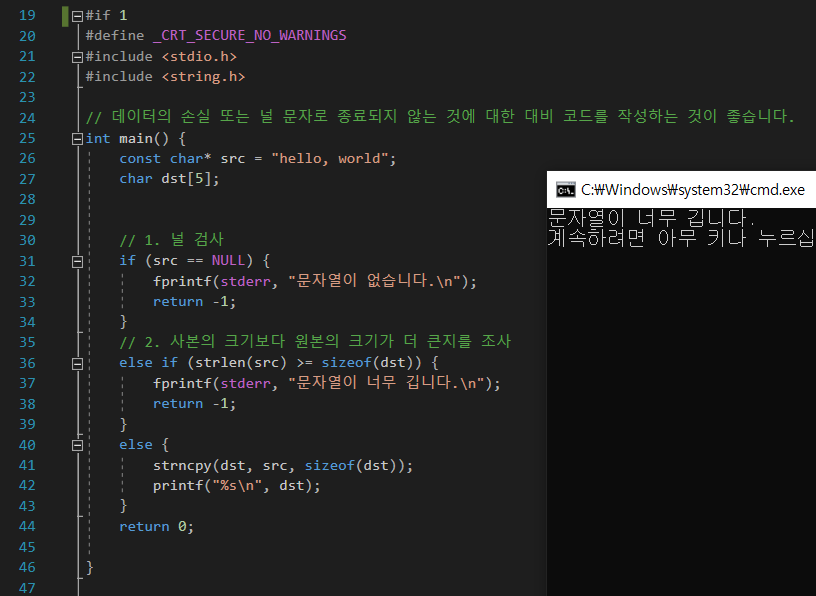
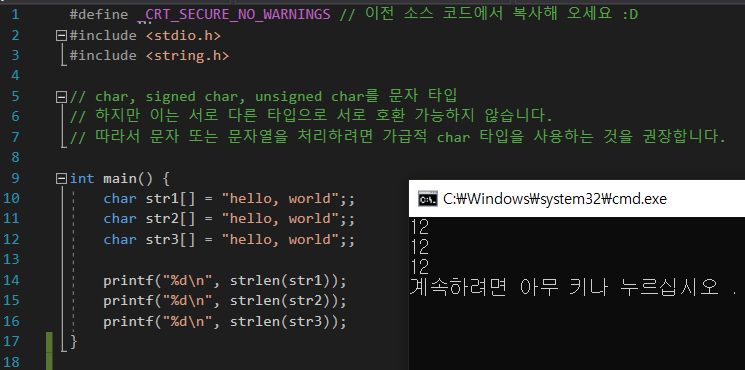
strncpy 내가 원하는 바이트만 복사를 수행해라! 라고 권장함

->strncpy, fgets 등은 버퍼 오버플로를 회피하기위해 사용되는 대표적 함수들이다. 하지만 이러한 함수들은 지정된 제한을 넘는 문자열에 대해서는 단순히 잘라버린다는 단점이 있다. 이는 데이터 손실과 더불어 소프트웨어 취약성이 될수있다.



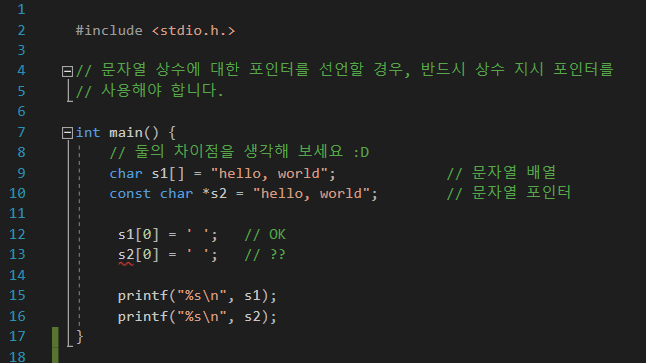
‘데이터의 손실’ 또는 ‘널문자로 종료되지 않는 것’에 대한 대비 코드를 작성하는 것이 좋다.

1. 널검사
2. 사본의 크기보다 원본의 크기가 더 큰지를 조사



char / signed char / unsigned char 문자타입은 서로 다른 타입으로 서로 호환가능하지 않다.

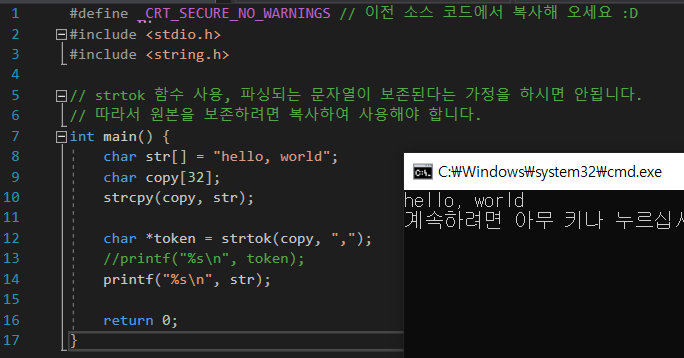
따라서 문자 또는 문자열을 처리하려면 가급적 char 타입을 사용하는 것을 권장한다.



어차피 hello는 상수이므로 바뀌지 않으므로 rodata에는 하나만 잡힌다.(굳이 사본 필요x)

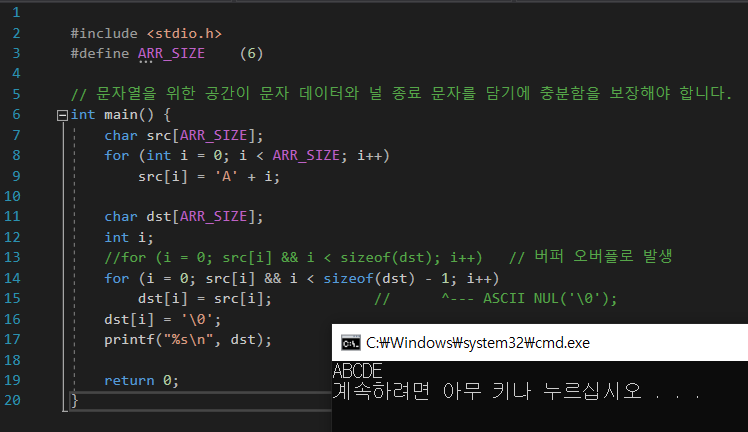
s2 컴파일타임에선 정상적으로 ok되지만 런타임에서 비정상적으로 종료된다.(왜냐하면 컴파일타임에는 rodata의 영역을 알수없으므로 넘어가지만 런타임에서 걸린다) 그래서 **s2 앞에 const써주면** 대상체가 상수라는 것을 알려주면 런타임이 아닌 컴파일타임에 오류가 발생한다.

->문자열 상수에 대한 포인터를 선언할 경우, 반드시 상수 지시 포인터를 사용해야한다.



strtok을 쓰게되면 “,” 가 \0로 바뀌게 되므로 str을 %s로 출력하게 되면 \0을 만나면 그 전만 출력되므로 잘려나가게된다.

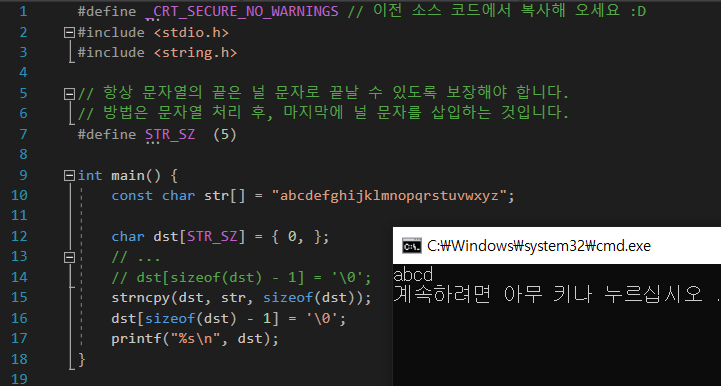
->따라서 원본을 보존하기 위해서는 복사하여 사용해야한다.( strtok함수 사용할 때 파싱되는 문자열이 보존된다는 가정을 하며 안된다.) 즉, 복사한 사본을 잘라서 사용하고 원본은 그대로 유지시킨다.



src배열에는 \0이 없으며 처음엔 null이 아니라 단순 쓰레기값들이 들어있다.

즉 문자열의 끝은 항상\0로 끝나게 해줘야한다. 그래서 sizeof(dst) 에 -1을 해줘야한다.

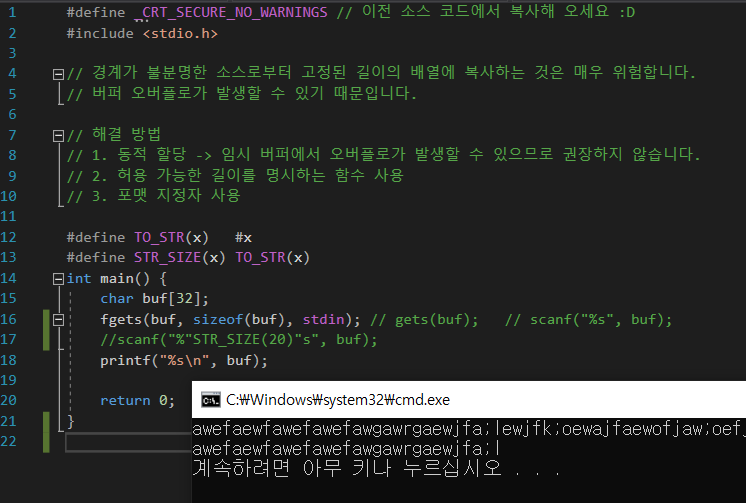
->문자열을 위한 공간이 문자 데이터와 널종료 문자들 담기에 충분함을 보장해야한다!!!



미리 \0을 넣어놔도 strncpy를 하게 되면 덮어쓰게 된다.

그래서 복사를 한후에 \0을 넣어야한다.

->항상 문자열의 끝은 널문자로 끝날수 있도록 보장해야한다. 방법은 문자열 처리 후, 마지막에 널 문자를 삽입하는 것이다.

경계가 불분명한 소스로부터 고정된 길이의 배열에 복사하는 것은 매우 위험하다(ex 32자가 넘어가는 많은 수를 입력받으면 죽게된다..)그래서 해결방법은

1. 동적할당 -> 임시 버퍼에서 오버플로가 발생할수 있으므로 권장하진 않는다.
2. 허용 가능한 길이를 명시하는 함수 사용 (o) fgets사용( 많은 수를 입력해도 잘라서 출력된다)
3. 포맷 지정자 사용 -> scanf에 %15s 같이 입력받을 길이를 지정하면된다.