소프트웨어 시스템 구현: 텍스트 파일의 검색 및 순위지정

20175201 김민수

<목차>

1. 서론-----------------------------------------------------------------------------------

2. Problem modeling--------------------------------------------------------------------

1)텍스트파일 검색 ------------------------------------------------------------------------------

2)순위지정----------------------------------------------------------------------------------------

3. Algorithm & Solution-----------------------------------------------------------------

1)검색함수----------------------------------------------------------------------------------------

2)정렬 알고리즘----------------------------------------------------------------------------------

4. Implement & evaluation--------------------------------------------------------------

1)구현---------------------------------------------------------------------------------------------

(1) 단어지정------------------------------------------------------------------------------------------------

(2) 파일 목록 받아오기------------------------------------------------------------------------------------

(3) 파일 읽기-----------------------------------------------------------------------------------------------

(4) 파일 검색-----------------------------------------------------------------------------------------------

(5) 정렬-----------------------------------------------------------------------------------------------------

2) 평가 및 수정----------------------------------------------------------------------------------

(1) 주소값--------------------------------------------------------------------------------------------------

(2) quick sort-----------------------------------------------------------------------------------------------

(3) 대소문자 구별------------------------------------------------------------------------------------------

(4) 평가 – 미흡했던점 ------------------------------------------------------------------------------------

5. 최종코드------------------------------------------------------------------------------

6. 참고문헌------------------------------------------------------------------------------

1. 서론

1학기동안 배웠던 c언어를 응용하기 위한 프로젝트로 텍스트 파일의 검색 및 순위지정이 진행되었습니다. 이 레포트는 각 주마다 목표였던 Problem modeling, Algorithm & Solution, Implement & evaluation 순으로 서술되었고 마지막에 최종 코드를 실었습니다.

2. Problem modeling

일단 프로그래밍에 들어가기 전에 문제 모델링에 들어갔습니다.

1)텍스트파일 검색

텍스트 파일을 찾는 부분은 4단계로 생각했습니다. 첫째로 찾을 단어 입력 받기. 둘째로 수업 때 배 웠던 파일 입출력으로 파일 열어서 단어 확인하고, 확인된 문서명, 수치(문서 내 단어의 개수 등) 리스트에 저장하기. 셋째로 반복문으로 두번째 단계를 반복하기. 마지막으로 저장된 리스트를 수치에 따라 정렬 후 출력하기.

2)순위지정

리스트를 정리할 기준은 2가지로 생각하였습니다. 우선 단어의 개수를 생각했습니다. 예를들어 사과에 대해 검색한다면 ‘사과는 맛있다’라는 하나의 문장보다 ‘사과는 장미목 장미과 배나무아과 사과나무속에 딸린 종으로 4월에서 5월 사이에 꽃을 피운다. 사과는 8월~9월에 꽃받침이 자라면서 생긴다.’ 라는 문장이 더 원하는 정보에 가깝다고 판단 되었기 때문입니다. 빈도의 경우 적은 문장에 적은 단어가 많은 문장에 많은 단어보다 높게 잡힐 확률이 높다는 판단에 고려하지 않기로 하였습니다. 두번째로 관련성에 대해 생각해보았습니다. 단어가 괄호안에 존재한다든가 큰따옴표, 작은따옴표안에 존재한다든가 단어 앞뒤에 공백이 있다던가 뒤에 마침표가 붙는다던가 한다면 원하는 정보에 가깝다고 판단하였습니다.

3. Algorithm & Solution

1) 검색 함수

C언어 문자열 검색 함수로 strstr함수가 있습니다. strstr함수 string.h에 선언되어 있는 함수로 문자열 내에서 특정 문자열이 나오면 그 문자열을 포함한 뒤의 문자열을 출력해주는 함수입니다. 만약에 찾는 문자열이 없다면 NULL을 반환합니다. 그림 1은 strstr함수를 사용한 예시입니다.



그림 1

2) 정렬 알고리즘

정렬알고리즘은 다양합니다. 대표적으로 bubble sort, insertion sort, quicksort, merge sort등이 있습니다. bubble sort, 거품 정렬은 시간복잡도가 O(n2)으로 상당히 느리지만 코드가 쉬운 정렬입니다. Insertion sort, 삽입 정렬은 자료배열의 모든 요소를 앞에서부터 비교해가는 정렬로 이미 한번 정렬된 데이터를 정렬할 때 효율이 좋은 정렬입니다. 마찬가지로 시간복잡도는 O(n2)입니다. Quick sort, 퀵 정렬은 최악의 경우 O(n2)번 비교하고 평균적으로 O(n log n)번 비교하는 정렬입니다. 퀵정렬은 대부분의 컴퓨터 아키텍쳐에서 효율적으로 작동하도록 설계되어 일반적으로 다른 O(n log n)에 비해 훨씬 빠르게 작동한다고 합니다. 마지막으로 merge sort, 병합정렬은 정렬되지 않은 리스트들을 절반으로 나누고 각 부분 리스트를 재귀적으로 합병정렬을 이용하는 정렬입니다. 시간복잡도가 O(n log n)인 정렬입니다.

저는 검색함수로 strstr을 이용하고 정렬알고리즘으로는 bubble sort로 구현한 후 quicksort로 바꾸도록 계획하였습니다.

4. Implement & evaluation

1) 구현

(1) 단어 지정

우선 찾고자 하는 단어를 gets 함수를 통해 입력받았습니다.

(2) 파일목록 가져오기

우선 주어진 텍스트 파일들을 불러오기 위해서 io.h 안에 들어 있는 struct 타입의 \_finddata\_t를 이용했습니다. 그림 2의 코드를 변형해서 사용하였습니다. 우선 \_findfirst를 통해 디렉토리 내의 첫 파일의 이름을 가져오고 strcpy를 이용해 복사했습니다. 그 후 \_findnext를 이용해 그 다음 파일의 이름을 가져오고 같은 함수를 사용해 복사했습니다. 이 복사해온 파일 이름들은 구조체 배열 안에 저장했습니다.



그림 2

(3) 파일 읽기

수업시간에 배웠던 파일 입출력을 이용해 파일을 읽어왔습니다. 파일 포인터와 fopen함수로 파일 목록 내의 파일 중 하나를 가져옵니다. char \* 를 이용 읽어온 파일을 2차원 배열에 임시로 옮겼습니다. 4)을 진행한후 목록안의 다음 파일을 읽었습니다.

(4) 파일 검색

파일 검색은 앞에서 제시했던 strstr함수를 이용했습니다. strstr함수의 결과가 null이 나올때까지 strstr함수를 사용하고 그 결과에서 맨 앞 글자를 제외하는 것을 반복하여 반복횟수를 통해 파일내 단어 개수를 구했습니다. 그리고 검색된 단어의 수는 파일 이름을 가져올 때 썼던 구조체 배열에 저장했습니다.

(5) 정렬

앞에서 제시했던 bubble sort를 이용해 구현하였습니다. 그림3이 구현한 코드입니다.

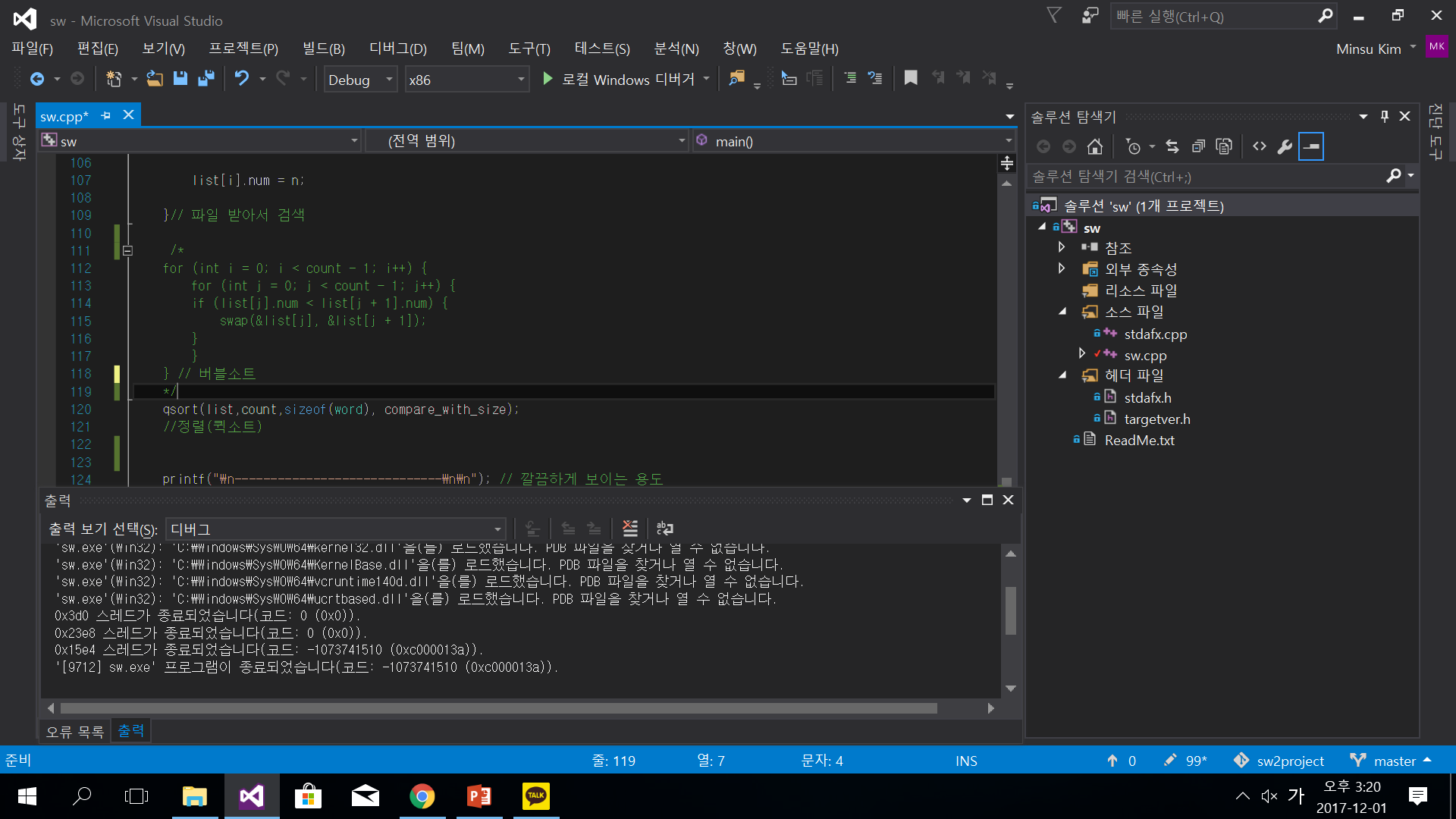


그림 3

2) 평가 및 수정

(1) 주소값

기존에는 정해진 경로의 txt파일들을 읽어왔습니다. 하지만 사용되는 기기마다 경로가 다를 수 있다는 점을 파악 경로를 사용자가 입력하도록 수정하였습니다. 그림4는 수정되어 경로를 입력받는 사진입니다.

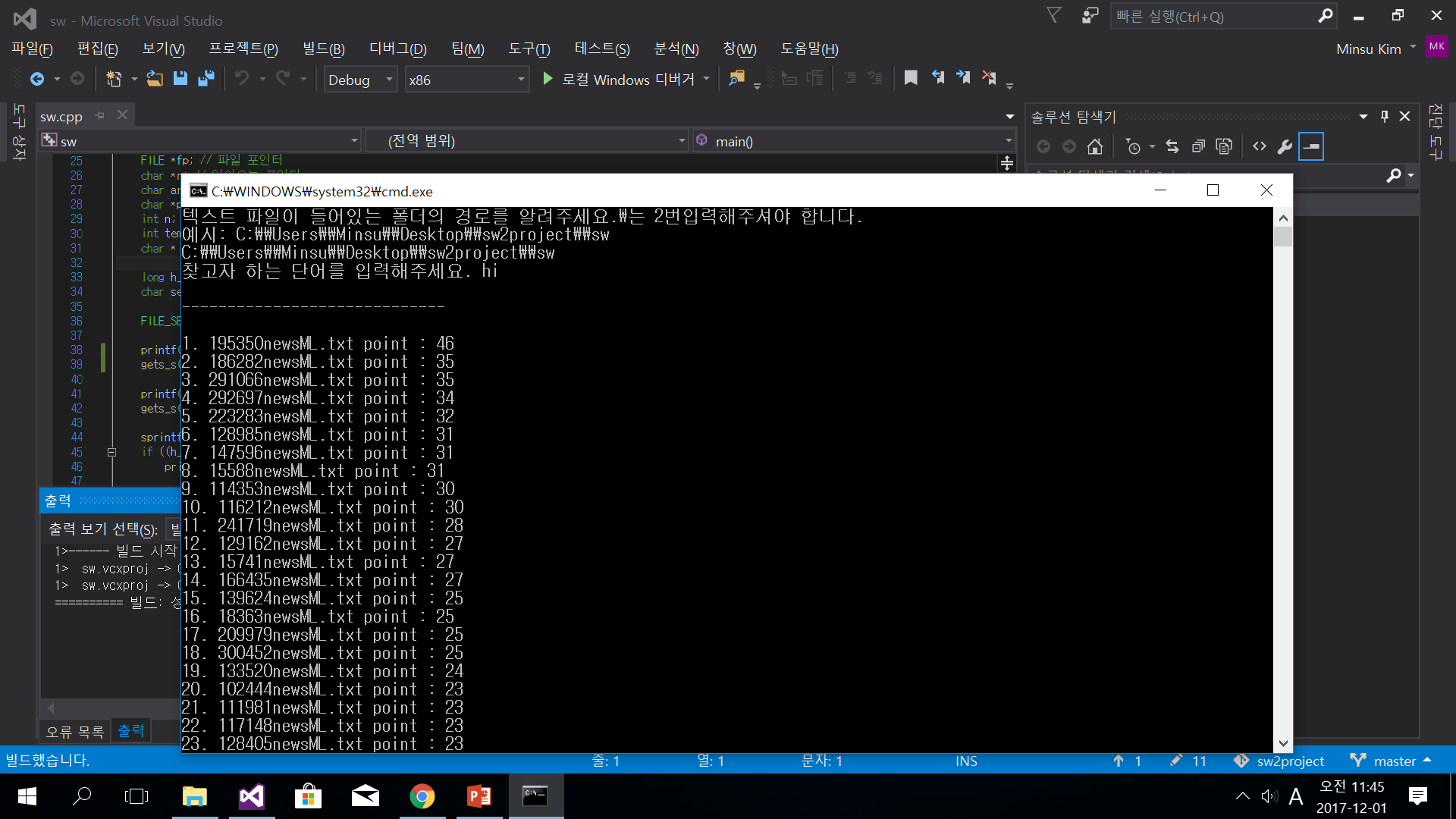


그림 4

(2) quick sort

기존의 bubble sort는 구현하기는 쉽지만 효율성은 quick sort에 비해 낮습니다. 그래서 앞의 bubble sort를 stlib.h에 있는 qsort함수를 이용하여 수정하였습니다. 아래의 그림5는 수정된 사진입니다.

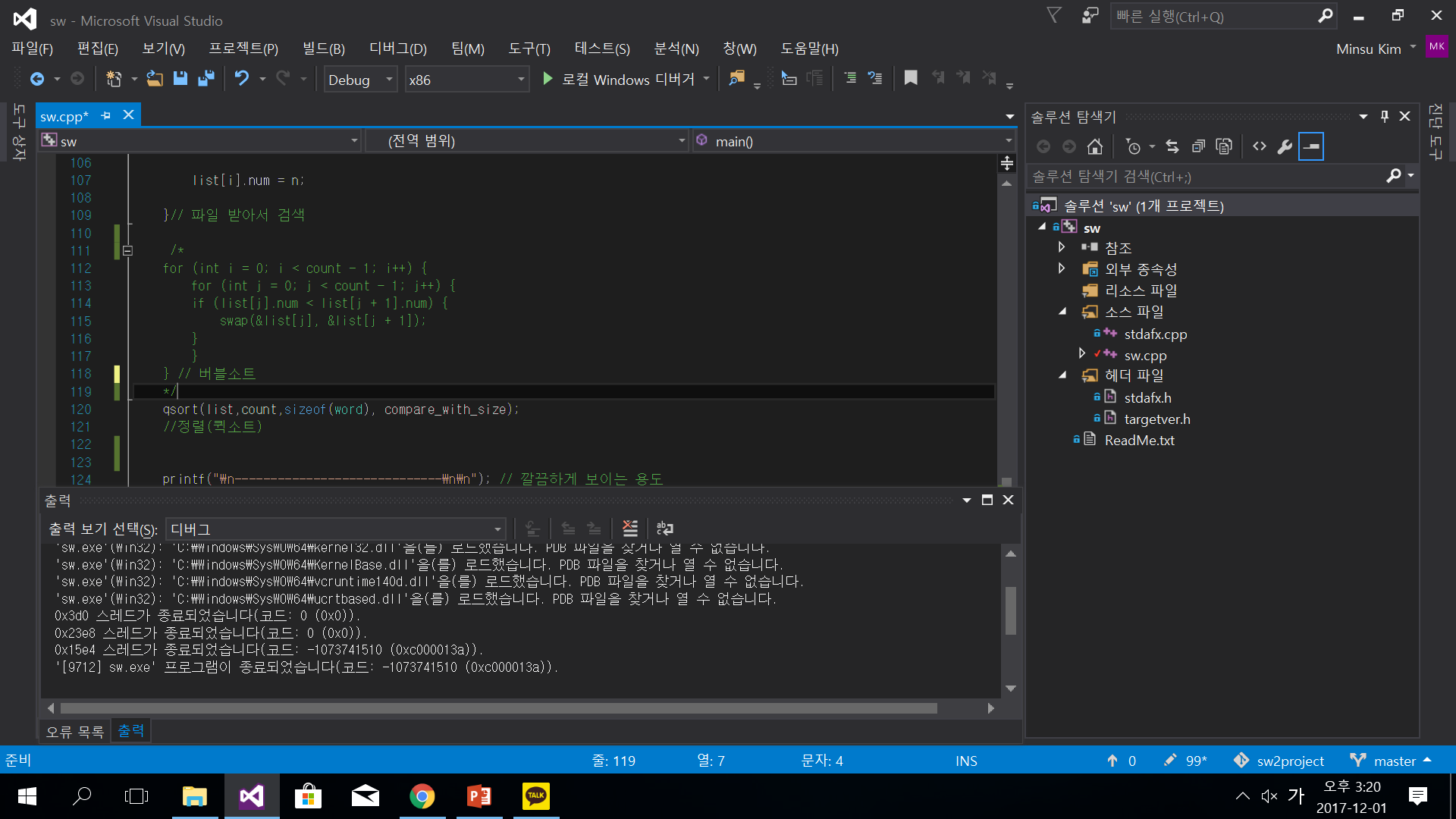


그림 5

(3) 대소문자구별

영어에서 문장의 첫문장, 고유명사, 편지의 인사말, 마지막 문장에 직함이나 이름, 역사적 사건, 기간 등등 첫글자가 대문자로 쓰이는 경우가 많습니다. 강조하기위해서 모든 글자를 대문자로 표기하는 경우 또한 있습니다.

수정전에는 대소문자에 관한 처리가 따로 없었습니다. 그 결과 그림 6, 7, 8에 나타난 것처럼 Korea를 검색하면 0건, Korea를 검색하면 17건, KOREA를 검색하면 1건이 나왔습니다.

이를 수정하여 단어를 입력하면 모든 단어가 소문자인 경우, 첫글자가 대문자인 경우, 모든 글자가 대문자인 경우를 탐색하도록 하였습니다. 그 결과 그림 9처럼 korea를 검색하면 18건 모두가 나타났습니다.

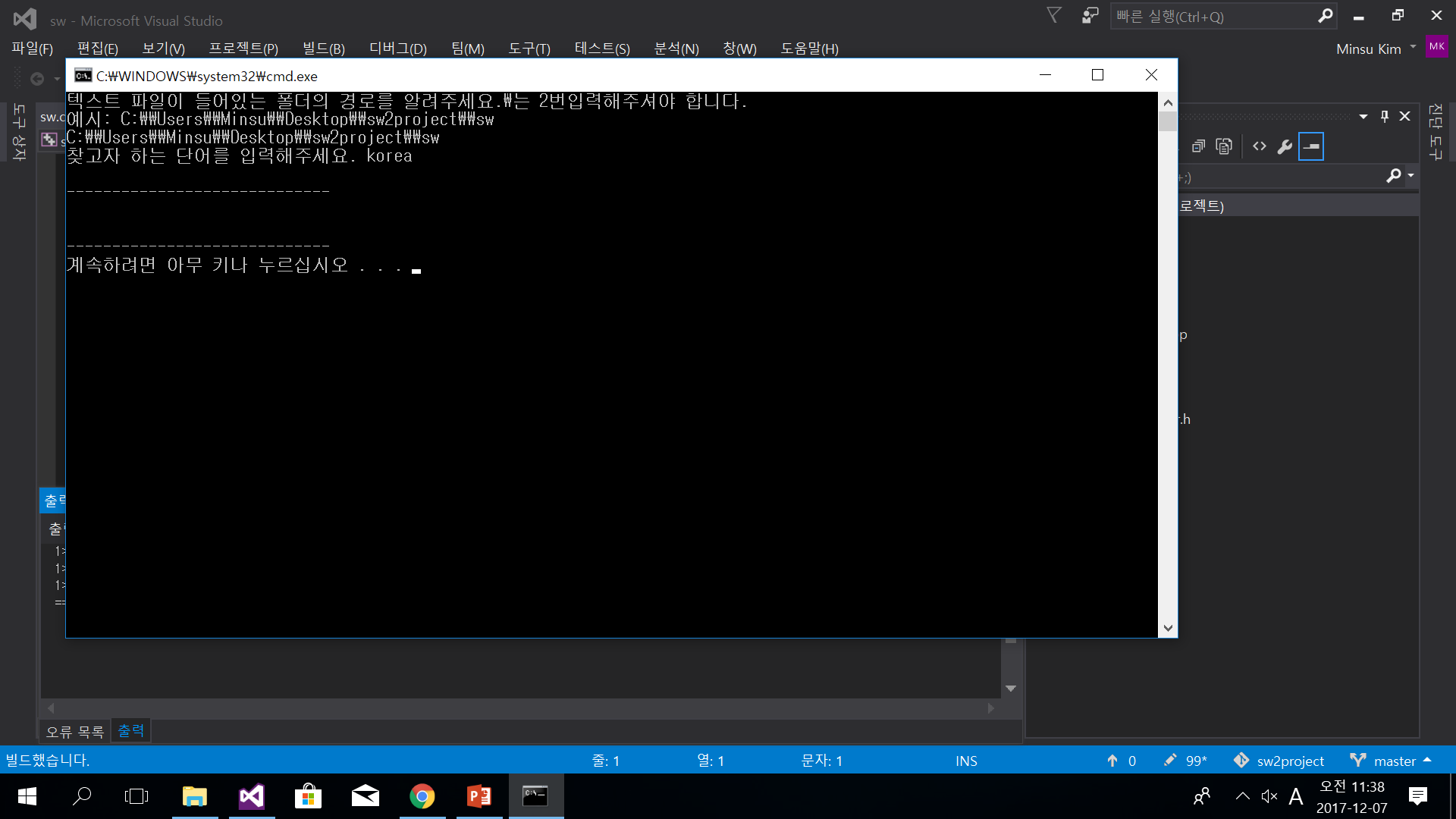


그림 6

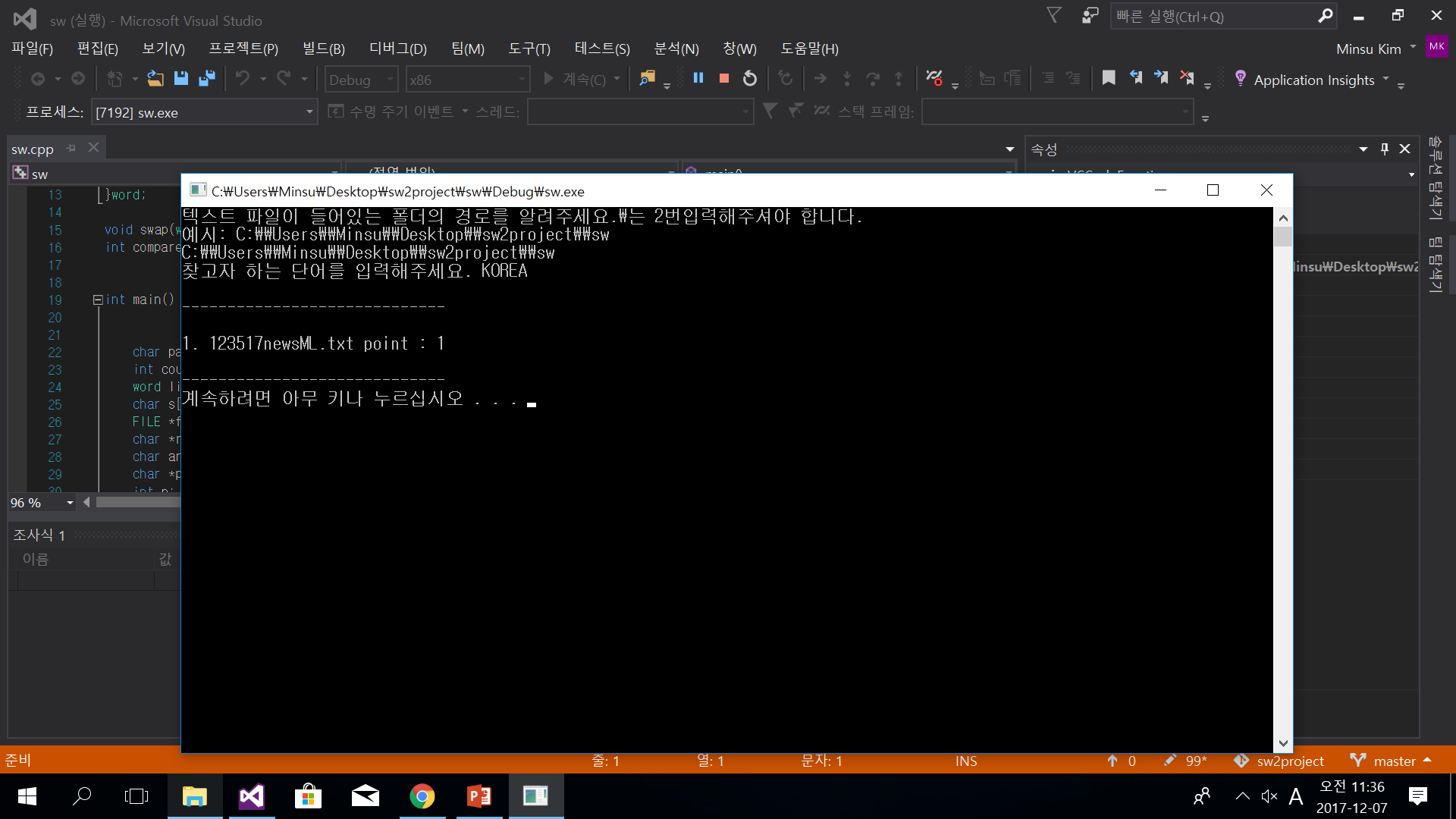


그림 7

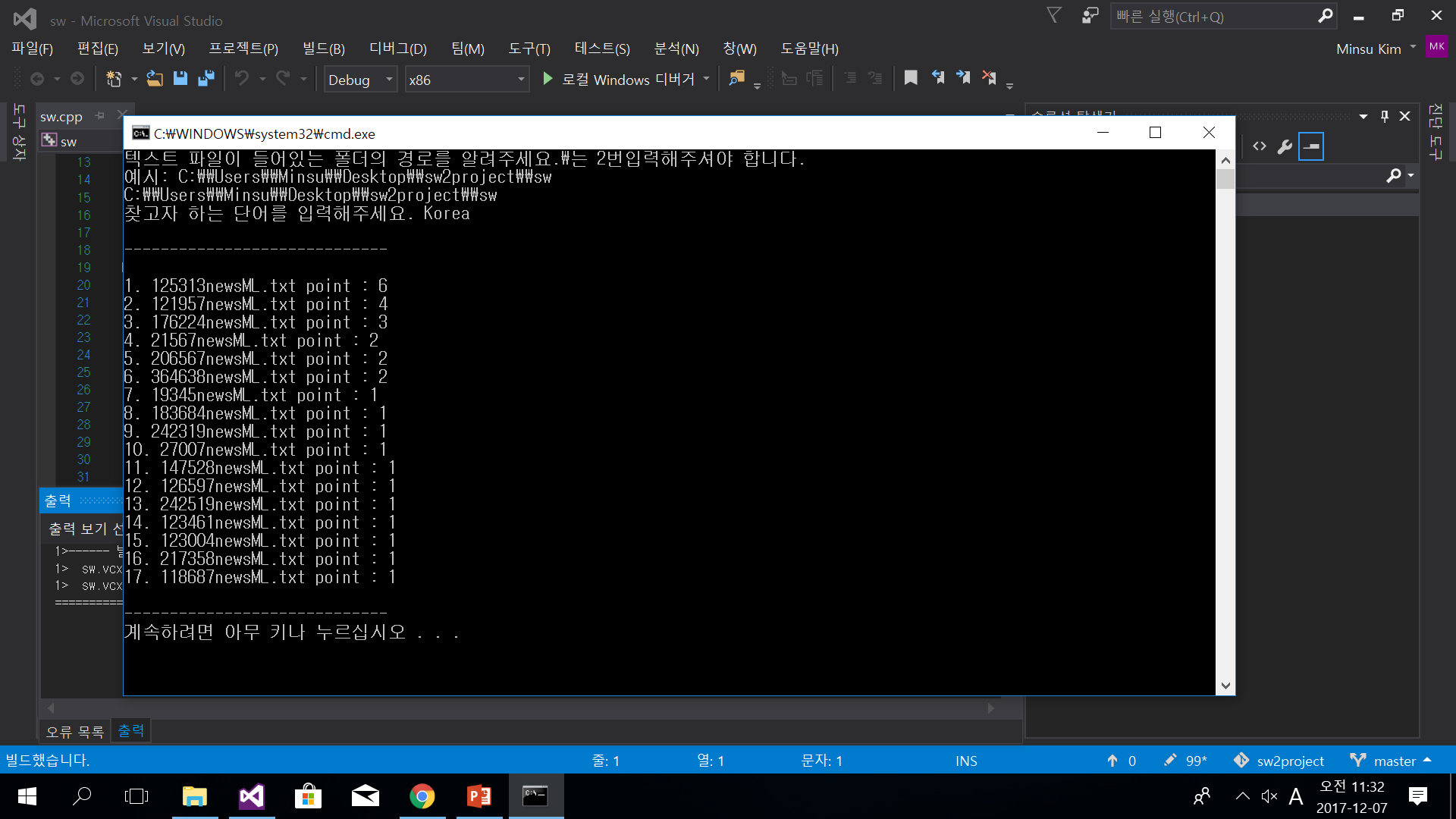


그림 8

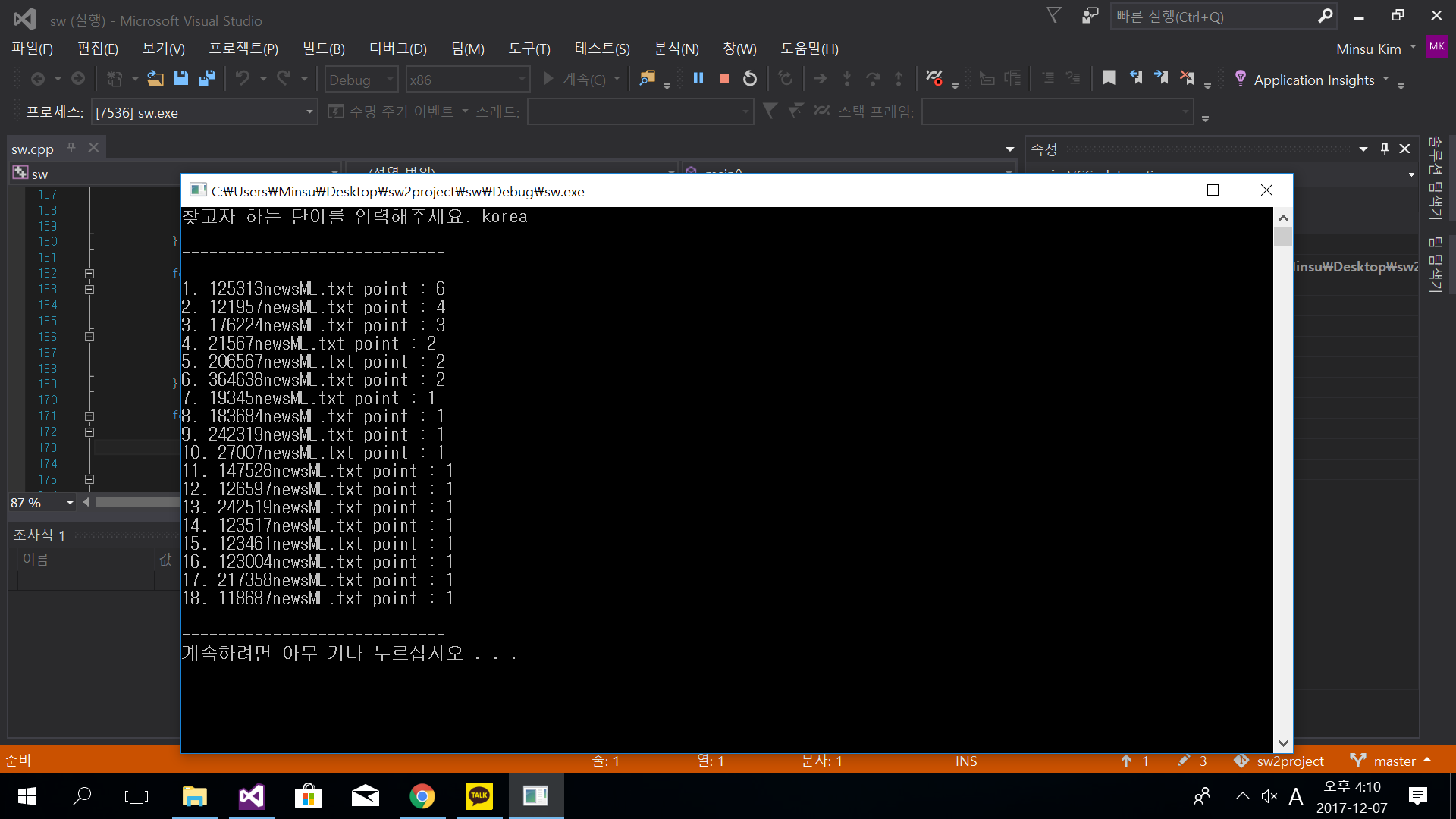
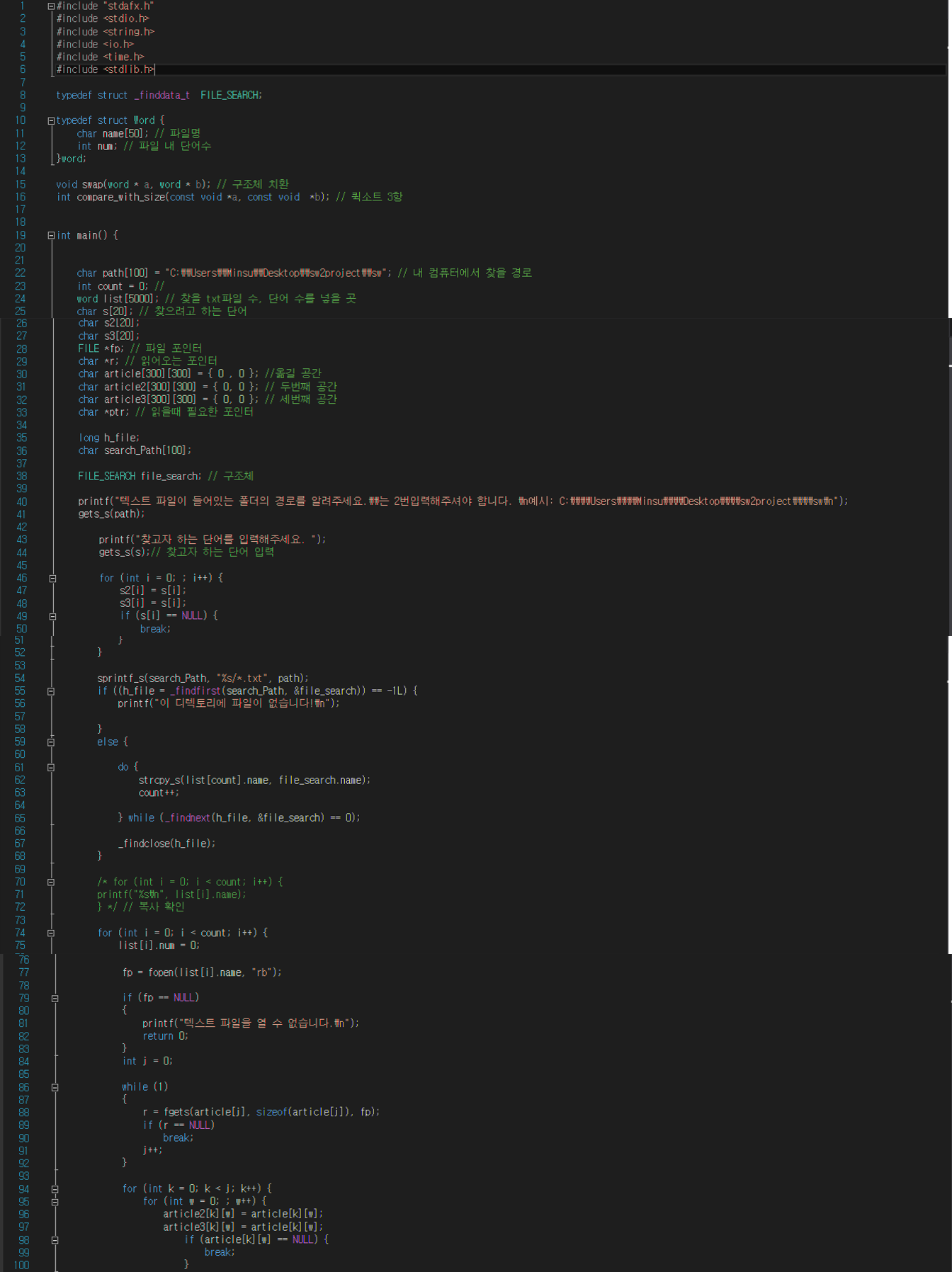
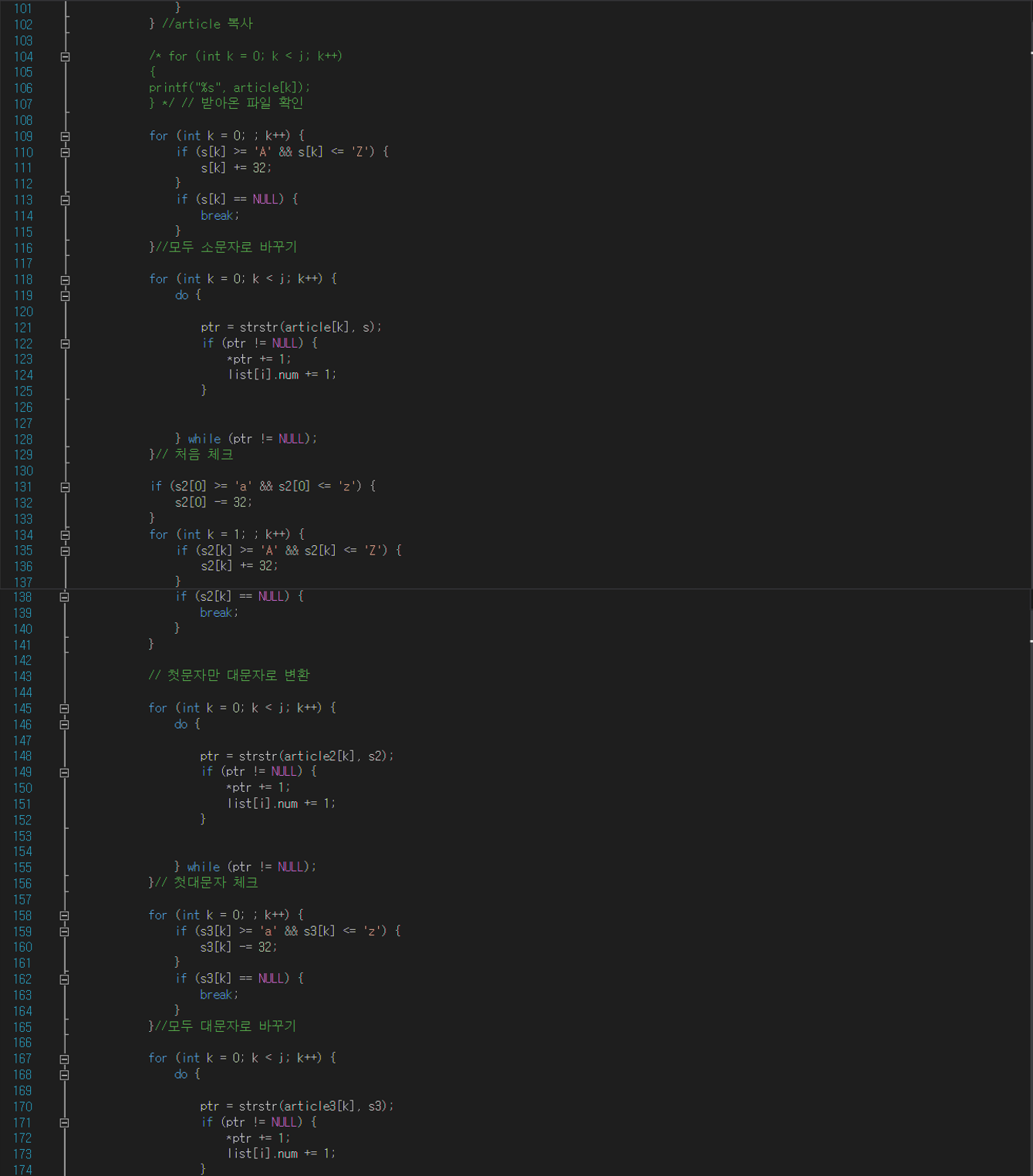


그림 9

(4) 평가 – 미흡했던점

처음 프로그램을 설계할 때 띄어쓰기나 따옴표와 같은 문장부호 등의 유무도 고려하여 우선순위를 배분하려 하였다. 하지만 본인의 역량부족으로 단어수로만 판단하게 되었다.

5. 최종 코드



6. 참고문헌

위키백과, “정렬알고리즘”

<https://ko.wikipedia.org/wiki/%EC%A0%95%EB%A0%AC_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98>

나무위키, “정렬알고리즘”

https://namu.wiki/w/%EC%A0%95%EB%A0%AC%20%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

Tistory, “\_finddata\_t”

http://sks3297.tistory.com/entry/C언어-파일목록-가져오기

Tistory “파일입출력”

http://blog.eairship.kr/21

마이크로소프트, “\_finddata\_t”

<https://msdn.microsoft.com/ko-kr/library/kda16keh.aspx>

마이크로소프트, “qsort”

https://msdn.microsoft.com/ko-kr/library/zes7xw0h.aspx

코딩도장, “문자열 검색”

<https://dojang.io/mod/page/view.php?id=371>

코딩도장, “구조체 배열”

<https://dojang.io/mod/page/view.php?id=446>