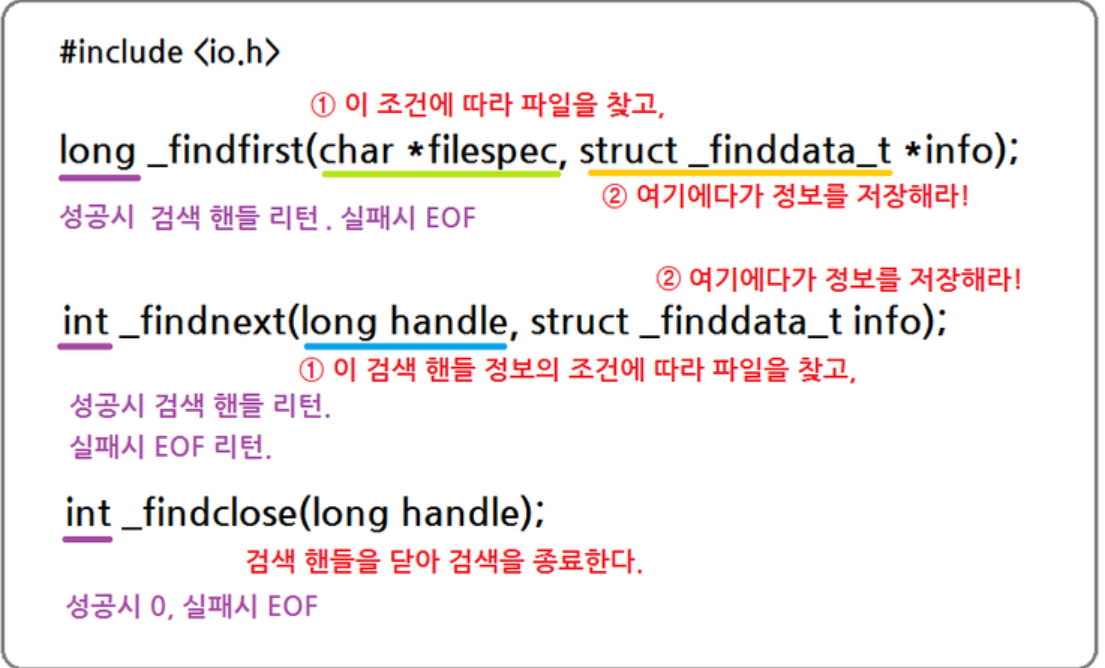
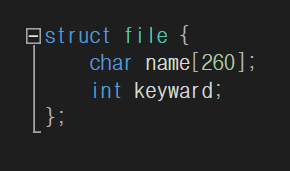
**Term Project 보고서**

**20170364한태균**

문제 : 여러 개의 파일들 중에서 키워드가 포함된 파일을 뽑고 Ranking하여라.

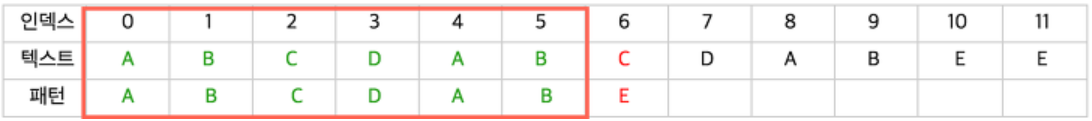
처음에는 파일들을 불러오는 것조차 갈피가 잡히지 않았다. 연결리스트를 사용하는 방법도 생각해 보았지만 아직 배우지도 않았고, 구현하는 데에는 나의 실력이 부족하다는 것을 느꼈다. 연결리스트를 사용하고자 한 것은 바이너리 트리의 형식으로 데이터를 저장하면 저장과 동시에 sorting이 가능하기 때문이었다. 이는 굉장히 효율적으로 보였지만 나는 구조체 배열을 사용하기로 하였다. 또한 디렉토리에서 파일을 불러오는 것들은 구글검색으로 해결했다. 처음 듣는 헤더파일인 <io.h> 를 선언하여 해결했다.  이 헤더파일에 있는 함수들은 내가 사용하기에 매우 적합한 것이었다. 처음 \_findfirst 함수를 사용할 적엔 편의를 위해서 텍스트파일들을 미리 옮겨 놓고 \_findfirst(“\*.txt”,&fd)의 형태로 사용했지만 그 후에는 직접 디렉토리를 찾아갈 수 있도록 수정하였다.

또한 sorting의 편의를 위해서 구조체 배열을 선언하였다.

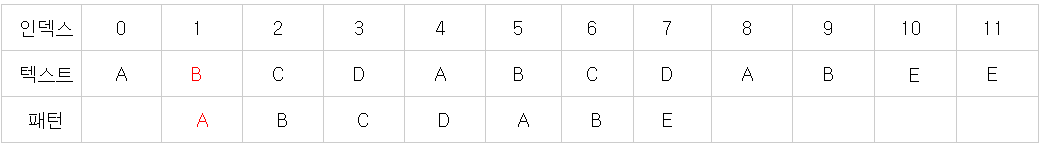


나의 생각은 키워드가 있는 파일은 구조체에 이름과 그 키워드에 개수를 저장하는 방식이었다.

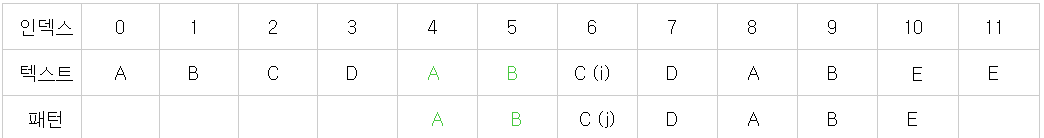
그래서 나중에 키워드 개수가 많은 순서대로 sorting 하는 것이다. 그리고 sorting한 순서대로 파일의 이름만 출력해주면 되는 일이었다.

이제 나의 고민은 키워드를 어떻게 판별하고 찾아내느냐는 것이었다. 처음에는 그냥 한글자씩 비교하자는 생각이었다. hello라는 것을 검색할 때, h 가 나올때까지 찾는다. 그 후 h다음이 e인지를 확인하고 그렇다면 그다음이 l 인지를 확인한다. 그렇지 않다면 다음으로 넘어간다. 라는 식이었다. 하지만 그 다음엔 조금 생각이 바뀌어 알고리즘 등을 찾아보기 시작했다. 처음 눈에 들어온 알고리즘은 kmp 알고리즘이었고, kmp알고리즘에 대해서 알아보기 시작했다. kmp알고리즘은 단어의 접두사와 접미사의 일치도를 이용하여 불필요한 비교과정을 없애주는 알고리즘이었다. kmp알고리즘을 이용하면

이 비교과정다음에

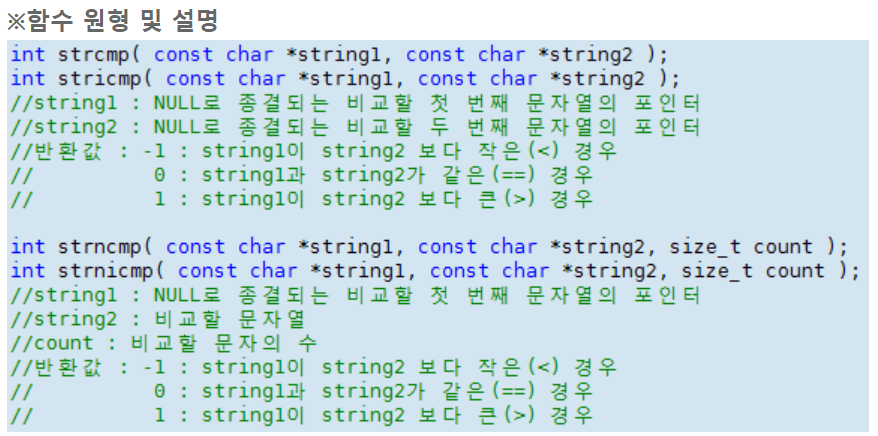


이렇게 비교하는 것이 아닌



바로 이 단계로 건너 뛸 수 있게 해준다. 하지만 kmp 알고리즘을 구현하기 위해서 알아야 하는 과정들이 복잡하고 어려웠다.

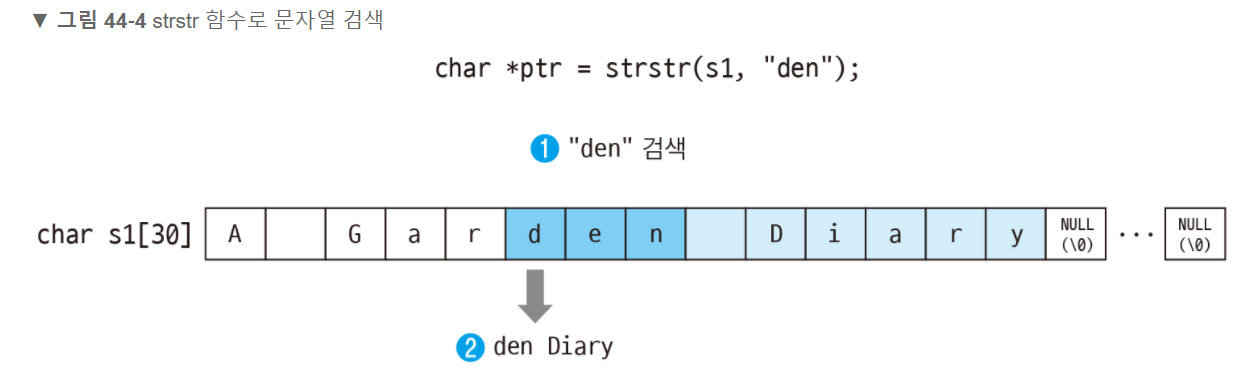
그렇기에 나는 내가 배운 것을 써먹어 보기로 하였다. fscanf로 문자열을 하나씩 받아와 키워드와 비교해 보려고 했다. 그래서 처음에는 strcmp로 문자열을 비교하였다. 하지만 문제점이 많았다. computer를 검색했을 때, computers 나 computer, 등 뒤에 추가적인 문자나 기호가 붙는 것들을 찾아내지 못하였다. 이를 개선하기 위해서 strncmp를 사용하게 되었다. 하지만 또 다른 문제가 있었다. 대소문자의 문제가 그것이었고, computer를 검색했을 때, Computer까지도혹은 Computers, COMPUTER같은 문자열도 찾아내야만 했다. 따라서 나는 strnicmp를 채용하기까지 이르렀다.



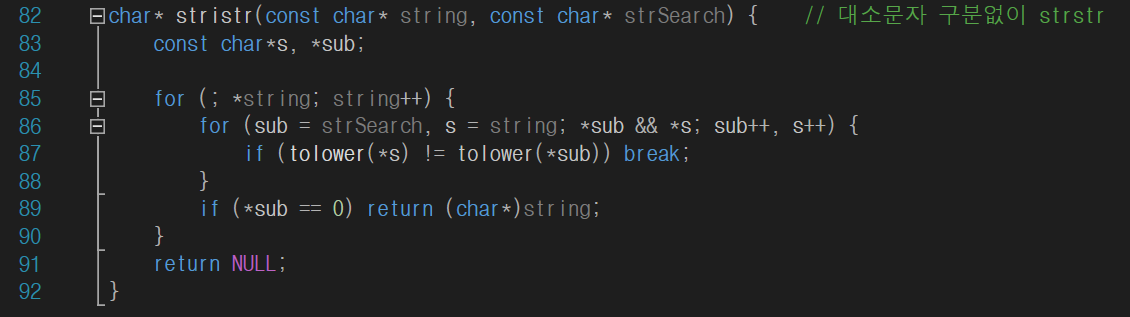
여기까지 다다른 나는 코드를 여러 번 체크해보았고 별 이상은 없어 보였다. 하지만 치명적인 오류들을 발견하였다. 내 코드는 앞에 문자나 특수문자가 있으면 키워드를 찾아내지 못한다.

“Computer ~~, &computer, mycomputer, 와 같은 경우였다.

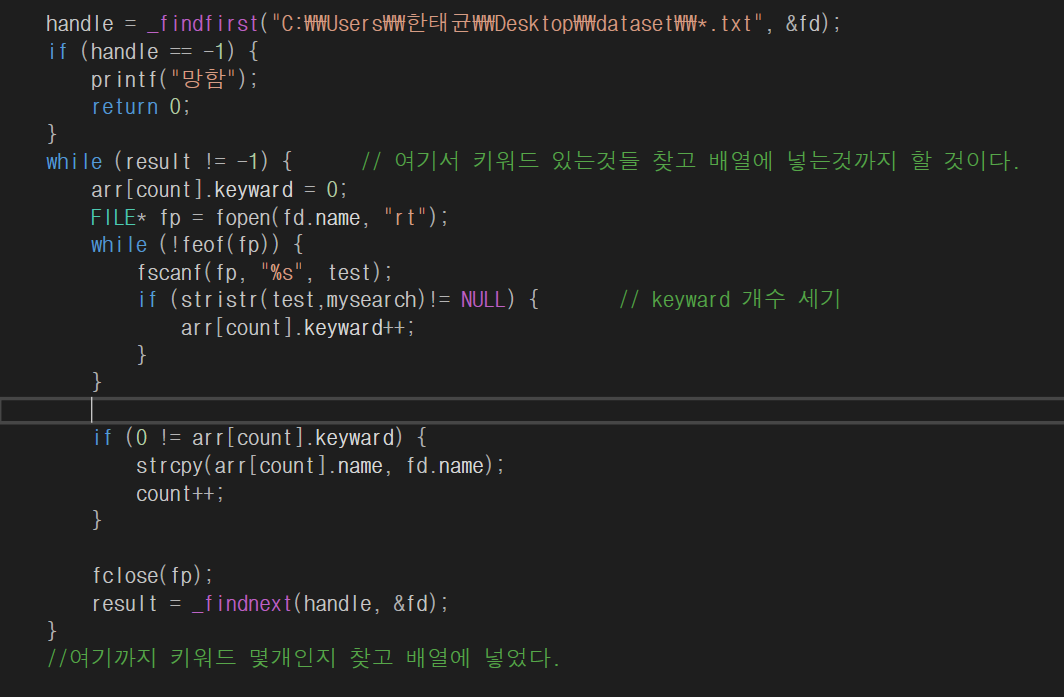
여기에서 나의 방법은 틀렸다고 생각했다. 그래서 strstr함수를 쓰기로 다짐하였다.



strstr함수를 사용함으로인해 전의 오류들은 사라졌지만 문제는 다시 생겼다. strstr함수가 대소문자를 구분하기 때문이었다. 따라서 나는 stricmp같은 stristr의 함수가 필요하였다. 하지만 표준 라이브러리에서는 stristr함수를 지원하지 않았다. 그래서 직접 만들어야만 했다.



여기까지 한 뒤 키워드를 찾아 구조체 배열에 넣었다. 다음은 파일을 가져오고 구조체 배열에 넣는 과정이다.



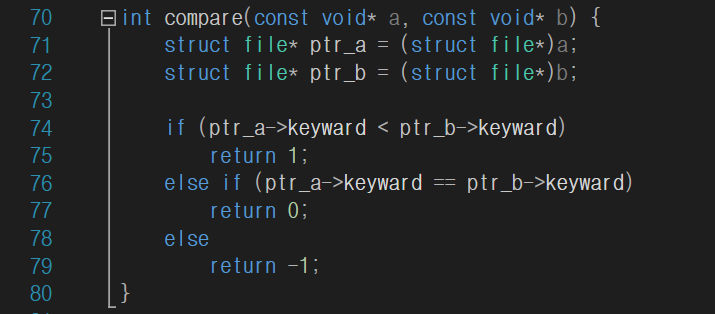
이제 sorting을 할 차례이다. 배운 sorting 방법은 버블솔팅이지만 새로운 방법을 사용해 보기로 했다. 버블솔팅은 효율이 좋지 않다고 들었기 때문이다. 그래서 qsort를 해보고 싶었는데 마침 구조체배열을 sorting하기에 적합하였다. 다음은 qsort를 사용하는데에 참고한 예제이다.



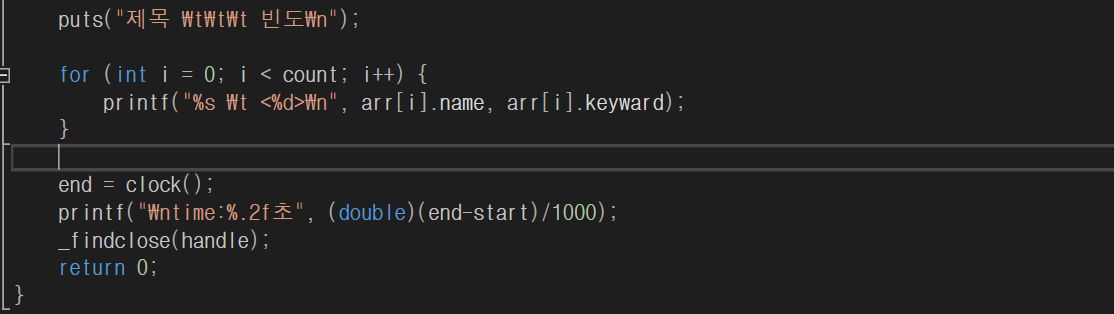


그렇게 나도 qsort함수를 구현해 보았다.

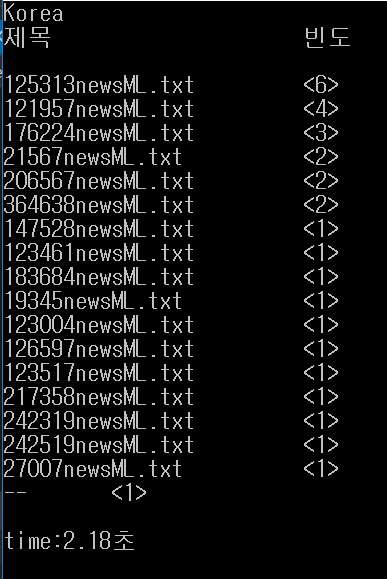




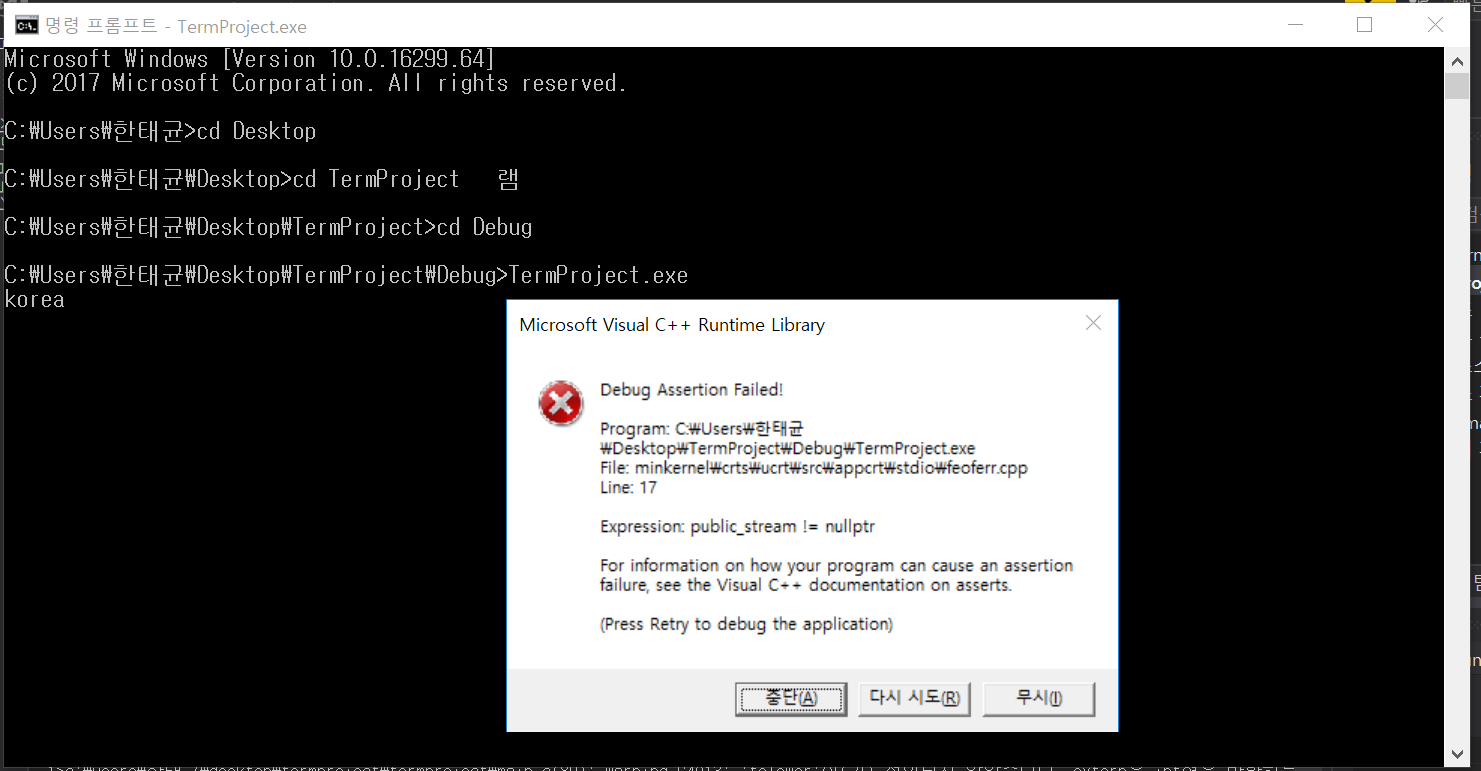
출력 파트:



Clock함수를 이용하여 시간도 측정할 수 있게 하였다.

-> 검색이 잘 되는 모습

각 파일에서 ctrl+f 해서 단어를 검색해가며 맞는지 보았다. CMD에서의 구현은 하지 못했다.



처음 문자를 받아올 때, strcpy(mysearch, argv[1]); 으로 받았는데 문제가 생겼다.

이번 프로젝트를 하면서 구글링을 굉장히 많이 했다. 정보바다에서 내가 필요한 정보를 찾고 적용하는 과정들이 나에게 도움이 된 것 같다. 내 코딩에도 아쉬운 부분들이 많고, 검색하는 데 있어서도 어떤게 정답인지 명확하지가 않다. 예를 들어 난 are을 검색하고 싶다 하지만 are과 의미가 전혀다른 rare도 검색된다. 이를 어떤 관점으로 보아야 할지도 생각해보아야 할 문제이다.

참고한 사이트들.

문자열 비교

<http://shaeod.tistory.com/219>

\_findfirst

<http://picomo.blogspot.kr/2014/04/cc.html>

kmp알고리즘

<http://bowbowbow.tistory.com/6>

io.h

<http://blog.naver.com/PostView.nhn?blogId=skout123&logNo=50132819661>

strstr

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

qsort

<https://flavono123.github.io/2017-05-03/sort-c-struct-with-qsort/>