

# 과제 5

전자사전 만들기

끝말 잇기

단어 분석

20232907 정현승

## 1. 문제 1

각종 단어와 속어가 저장되어 있는 파일을 이용해 전자사전을 만든다. 영어 단어를 입력하면 그에 해당하는 뜻을 출력한다. 단 파일에 정해진 양식과 다른 데이터가 있는 오류는 무시하고 프로그램을 작성한다.

## 2. 문제 1 해결 방안

먼저 텍스트 파일을 열어 단어와 속어가 저장되어 있는 텍스트 파일을 모두 읽고 한 문자열에 저장했다. 문자열에 텍스트 파일 내용을 옮기는 작업이 모두 끝났으면 텍스트 파일을 닫고 나머지 작업을 수행한다. 이후 단어와 단어 사이가 줄 변경 문자로, 영어 단어와 뜻 사이가 콜론(':')으로 구분되어 있다는 점을 이용해, strtok 함수로 각 단어와, 영어 단어와 뜻을 분리하였다. 이는 먼저 strtok 함수에 문자열을 자르는 기준이 되는 문자(함수의 2번째 매개변수에 들어가는 값)에 ':'을 넣어 영어 단어를 분리해 내고, 이후 다시 strtok 함수에 문자열을 자르는 기준이 되는 문자(함수의 2번째 매개변수에 들어가는 값)에 'wn'을 넣어 뜻을 분리해 낸 후, ':' 앞뒤로 있는 공백을 제거하는 순서로 진행하였다. 그러니까 영어 단어는 ':'을, 뜻은 'wn'을 기준으로 잘랐다는 의미이다. 이렇게 문자열을 자른 후 그 결과값을 문자열의 2차원 배열에 모두 저장했다. 이때 단어의 수가 많아질 것을 고려해, 2차원 배열을 동적으로 할당받아 사용했다. 코드에서 char\*(\*)[2]같은 자료형이 등장한 이유는 이 2차원 배열을 포인터로 담기 위한 것이다. (char\*[2]의 포인터이므로 char\*(\*)[2] 형태가 된다.) 코드 내에서 (\*dict)[0] 같은 형태도 저 자료형에서 문자열(char\*)에 접근하기 위하여 자주 사용된다. 이렇게 2차원 배열을 완성하면 사전 텍스트 파일을 읽고 사용하기 적합하게 바꾸는 과정은 끝이며, 이는 문제 2와 3에도 동일한 방법으로 적용된다.

사전에 있는 단어인지 비교하는 함수는, 비교 대상이 되는 문자열과 사전 정보를 매개변수로 받은 뒤, 첫 번째 단어부터 하나하나 문자열을 비교하는 방식으로 진행했다. 이진 탐색을 사용하지 않은 이유는, 사전 정보를 담은 배열의 동적할당 크기를 어디서도 저장하지 않기 때문이다. 이진 탐색을 진행하려면 이를 계산하는 과정부터 시작해야 하는 문제가 있어 선형 탐색으로 구현했다. 사전에 일치하는 단어가 없어 사전 끝까지 확인을 진행한 경우, 사전이 끝났음은 배열의 마지막 원소가 NULL을 저장하고 있다는 점을 기준으로 삼았다(앞에서 사전 정보를 만들 때 1개 단어가 더 들어갈 공간까지 여유롭게 동적할당 받아서 메모리 접근에 문제가 생기지 않는다). 사전에 저장된 단어 중 비교 대상이 되는 문자열과 완전히 같은 단어가 있으면 그 단어의 저장 위치(포인터)를 반환하고, 다음 단어의 정보가 NULL인 경우, 사전에 찾는 단어가 없다는 의미인 NULL을 return 했다.

사전 파일을 열어 메모리로 올리는 것을 위에 설명한 방식으로 진행한 후, 사용자에게 단어를 입력받는 것은 과제 3에서 만들어 사용했던 sgets 함수를 사용했다. 역시 동적할당에 실패하면, 에러메시지는 함수 내에서 출력 될 테니 바로 프로그램을 종료해주고, 그렇지 않으면 사전에 있

는 단어인지 비교하는 함수를 이용해 그 단어의 의미와 뜻을 화면에 출력했다. 마지막으로 sgets 함수에서 동적으로 할당된 공간을 해제해주고, 이를 무한 반복하여 문제를 해결하였다.

이를 적용하여 문제 1을 해결한 결과는 다음과 같다.

### 3. 문제 1의 결과

그림 1

그림 2

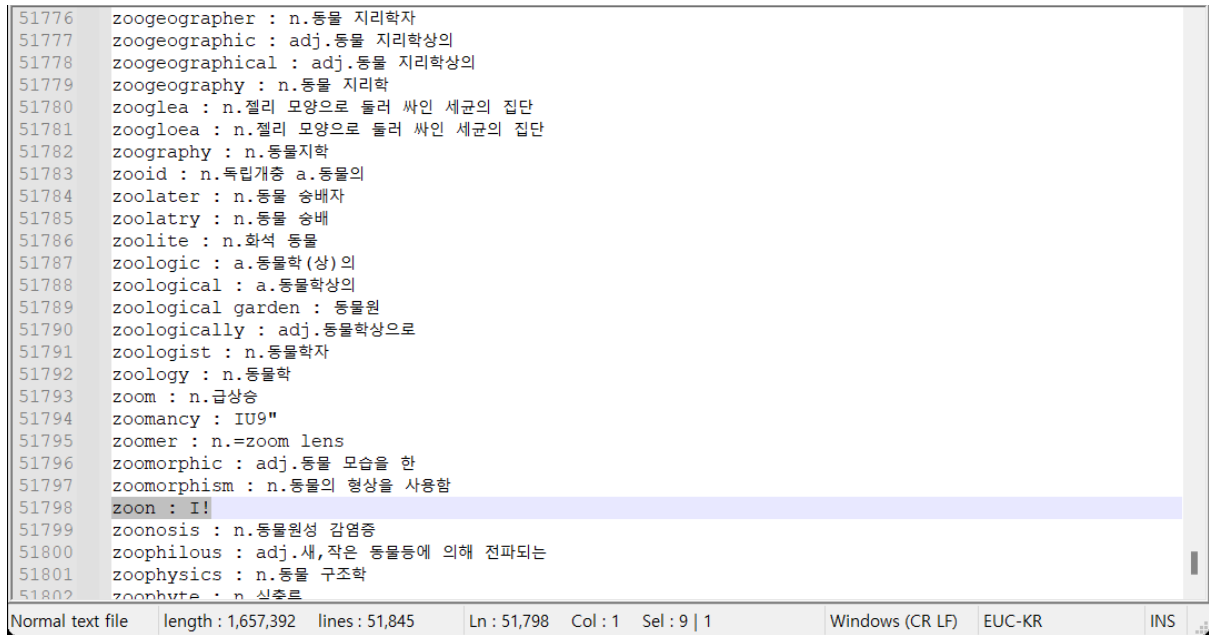


그림 3

그림 1은 실행 결과 화면이고, 그림 2와 3은 이를 설명하기 위한 사전 파일의 캡처본이다. 그림 1에서 단어가 사전에 있는 경우 입력한 단어와 그 뜻이 출력되고, 입력한 단어가 사전에 없으면 해당 단어가 사전에 없다는 메시지를 출력하고 있다는 점을 볼 수 있다. 또한 사전 파일에 있는 오류를 무시했기 때문에, zoon이나 zabaglione 같은 단어의 경우 이상한 값이 나왔지만, 저것도 사전 파일에 있는 값을 그대로 출력한 것이다. 이는 사전 파일을 열고 캡처한 화면인 위의 그림 2와 3에서 확인할 수 있다.

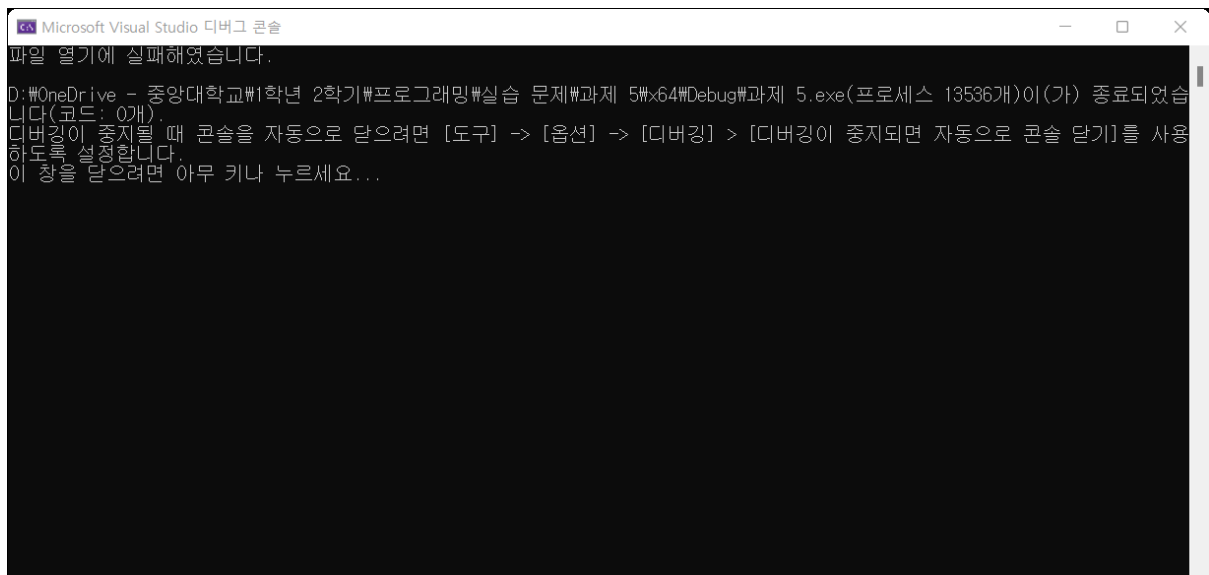


그림 4

위 그림 4는 파일 열기에 실패했을 때의 실행 결과이다. Break point를 이용해 fopen 함수의 반환 값을 0으로, 임의로 바꾼 것을 제외하고는 코드를 만지지 않았다. 위처럼 파일 열기에 실패했을 때도 메시지 출력과 프로그램 종료가 정상적으로 작동하는 것을 볼 수 있다.

## 4. 문제 2

혼자서 하는 끝말잇기를 구현한다. 단 시작에 한해서 컴퓨터가 제시어를 하나 고른다. 사용자는 3~10글자의 영어 단어를 입력하며, 유효한 단어인 경우 1점을 얻고, 단어의 길이가 맞지 않는 경우, 이미 나온 단어, 사전에 없는 단어, 끝말잇기가 안 되는 단어가 입력되면 기회를 한 번 잃는다. 기회를 3번 잃으면 게임이 종료된다.

## 5. 문제 2 해결 방안

사전 파일을 메모리에 올리는 것과 단어 검색은 문제 1에서 만든 코드를 그대로 사용하였고, 사용자의 단어 입력도 과제 3에서 만든 `sgets` 함수를 사용하였다. 컴퓨터의 단어 선택은 사전에 있는 단어 중 무작위로 하나를 고르는 방법으로 구현하였으며, 해당 단어에 대해 다음 문단에서 설명할, 사람이 맞는 단어를 골랐을 때와 같은 처리를 하고 컴퓨터가 고른 단어를 출력하는 방식으로 하였다. 그러나 사전 파일을 배열로 만들 때 배열의 크기를 함수에서 반환하지 않기 때문에, 그 크기를 직접 다시 계산하여 무작위 값을 계산한 값으로 나누었다. 이렇게 사전의 크기를 다시 계산하는 작업은 문자열을 입력받을 때 `gets` 함수나 `fgets` 함수에서 이번에 입력받은 문자열의 길이를 알려주지 않아 별도로 `strlen` 함수를 써야 하는 상황과 비슷하다고 할 수 있다. 또 `rand` 함수의 무작위 값 최댓값이 사전의 단어 개수보다 작은 상황이 발생할 수 있다. 따라서 무작위 값의 최댓값이 몇 비트로 나타낼 수 있는 최댓값인지를 계산하여, 무작위 값의 최댓값이 32비트가 되도록 무작위 값을 계속 얻어낸 후 `shift` 연산을 활용해 값을 저장했다. 무작위 값의 최댓값이 15비트일 때를 예로 들면, 무작위 값을 뽑아 `unsigned int` 형 변수에 그대로 넣고, 다시 무작위 값을 뽑아 이번에는 왼쪽으로 15비트만큼 `shift` 연산을 진행해 방금 활용한 `unsigned int` 형 변수에 더한다. 또다시 뽑아 이번에는 왼쪽으로 30비트만큼 `shift` 연산해 `unsigned int` 형 변수에 더하여 32비트를 전부 무작위 값으로 채우는 것이다.

이후 단어가 입력되면 먼저 단어의 길이를 확인한다. 단어의 길이가 맞지 않는 경우 길이가 올바르게 맞지 않는다는 메시지를 출력하고 기회를 한 번 차감한다. 그리고 끝말이 이어지는지도 확인하고 이어지지 않으면 단어가 안 이어진다는 메시지를 출력 후 기회를 한 번 차감한다. 이 둘에 모두 해당하지 않으면, 사용자가 입력한 단어가 사전에 있는지 검색한다. 그 단어가 사전에 있고 이전에 입력된 단어가 아니면, 점수를 1점 올리고 이번 단어의 끝말을 저장한 뒤 사전 2차원 배열에서 뜻이 저장된 포인터를 지운다. 이는 이 프로그램에서 단어의 뜻은 사용하지 않기 때문에, 뜻을 가리키는 포인터 변수가 있는지, 아니면 뜻이 `NULL`을 가리키고 있는지로 이 단어가 이미 나온 단어인지를 판정하도록 프로그램을 설계하였기 때문이다. 따라서 사전에 일치하는 단어는 있는데 뜻을 저장하는 문자열이 `NULL`을 가리키면, 이미 나온 단어를 입력한 것이라 보고 기회를 한 번 차감한다. 이외에도 사전에서 단어를 찾지 못하면 이 정보를 출력하고 기회를 한 번 차감한다. 기회 값을 저장하는 변수는 처음에 3으로 시작하고 0이 될 때까지 위 과정을 반복하며, 모든 기회

이를 이용하여 문제 2를 해결한 결과는 다음과 같다.

파일이 열리지 않는 경우는 문제 1에서 이미 확인했으므로 다시 테스트하지는 않았다.



```

C:\Microsoft Visual Studio 디버그 콘솔
컴퓨터: poet
말하기 단어: t
바르지 않은 단어의 길이입니다.
재점수 0점
말하기 단어: thrill
재점수 1점
말하기 단어: lily
재점수 2점
말하기 단어: yacht
재점수 3점
말하기 단어: ttrreeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
메모리가 부족합니다.
장치를 입력받을 메모리가 부족합니다. 사전에 있는 어떤 단어보다 긴 단어를 입력하였으므로 더 이상 입력을 받지 않습니다.
재점수 3점
말하기 단어: ttrreeee
재점수 3점
말하기 단어: ttrreeee
재점수 3점
D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 5\64\Debug\과제 5.exe(프로세스 2880개)이(가) 종료되었습니다.(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

```

그림 6

위 그림 6에서 단어의 길이가 3글자보다 짧은 경우와 입력 메모리 할당에 실패했을 때, 그리고 사전에 없는 단어를 입력한 경우에 대해 메시지와 프로그램의 처리를 볼 수 있다. 여기서 메모리 할당에 실패한 경우를 만들기 위해, break point를 사용하여 메모리 할당에 실패한 것처럼 변수에 저장된 값을 직접 바꾸었으며, 이 이외에는 프로그램 실행 중 변수나 과정 등을 일절 건드리지 않았다.

## 7. 문제 3

사전 파일에서 단어의 길이가 긴 순서대로 단어 10개, 모음이 가장 많이 포함된 단어 10개를 표시한다. 이때 다른 사전 파일이 들어와도 다른 파일에 대해서 정상적으로 계산될 수 있도록 분석 작업을 직접 프로그램이 해야 하며, 미리 분석한 후 결과만을 출력하게 하면 안 된다. 해당 분석 작업을 하는 프로그램을 구현한다.

## 8. 문제 3 해결 방안

사전 파일을 메모리로 올리는 것까지는 문제 1과 같으나, 여기서는 분석 2개를 한 번에 수행하고 출력을 한꺼번에 하는 방향으로 구현했기 때문에 자료형이 `char*(*)[2]`인, 사전 단어별로 단어와 뜻의 저장 위치를 저장하는 2차원 배열이 2개 있어야 한다. 그러니까 문자열이 2개 있을 필요는 없지만, 이를 가리키는 포인터 배열은 2개가 있어야 한다. 따라서 2차원 배열을 하나 더 만들었으며, 그 과정은 사전 배열의 길이를 문제 2에서처럼 계산해 그 크기만큼 2번째 2차원 배열을 동적으로 할당받고, 첫 번째 2차원 배열에 저장된 값을 그대로 복사하는 방법으로 해결했다. 이후 `qsort` 함수를 이용해 각각 단어의 길이가 긴 순서와 모음이 가장 많이 포함된 순서로 정렬하였다. 마지막으로 정렬된 각 단어 배열 중 첫 10개씩을 순서대로 출력한 뒤, 동적으로 할당받은 모든 공간을 해제하는 방법으로 문제를 해결하였다. 다만 사전의 길이가 10보다 작으면 사전의 길이만큼만 출력했다.

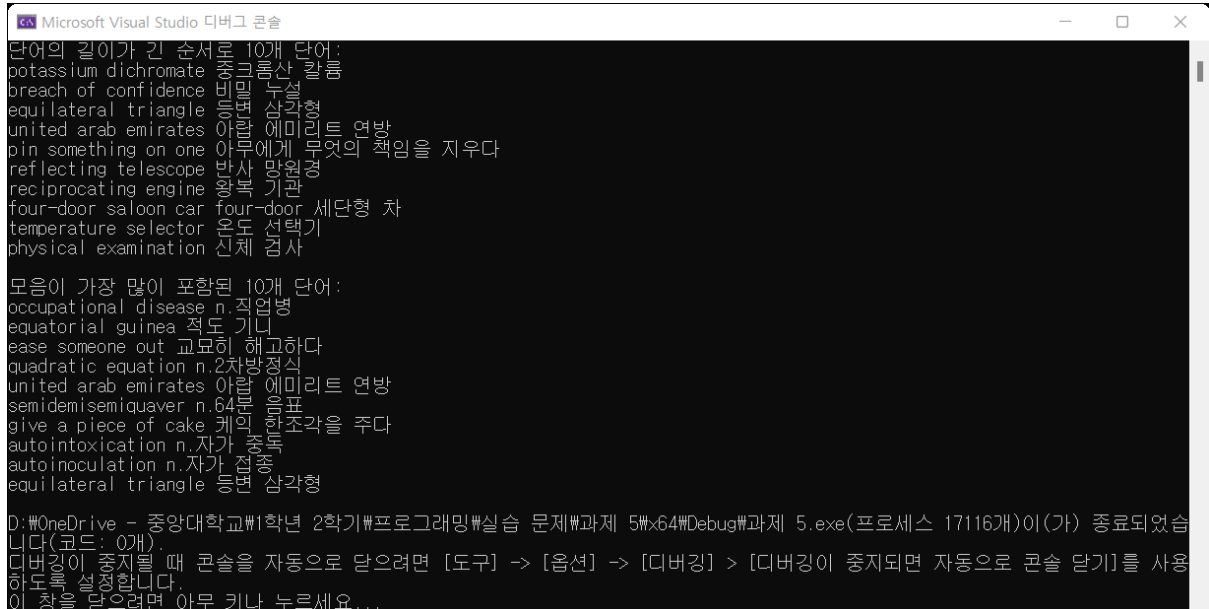
`qsort`에 들어가는 함수는, 단어의 길이 순서로 정렬하는 경우 2번째 단어의 길이에서 1번째 단어의 길이를 뺀 값을 반환했고, 모음이 가장 많이 포함된 단어 순서로 정렬하는 경우 단어별 글자별로 모음인지 확인해, 0부터 시작해 1번째 단어에 모음인 단어가 있으면 1씩 빼고, 2번째 단어에 모음인 단어가 있으면 1씩 더한 후 최종 경과를 반환하는 함수를 작성해 이용했다.

문제에서 단어의 길이와 모음의 길이가 같을 때의 처리 방법은 아무것도 제공하지 않아서 프로그램 또한 이 사례를 고려하지 않고 만들었기 때문에, 다른 방법으로 만든 프로그램과 비교해 그 결과가 다를 수도 있다.

이를 이용해 문제 3을 해결한 결과는 다음과 같다.

## 9. 문제 3의 결과

파일이 열리지 않는 경우는 문제 1에서 이미 확인했으므로 다시 테스트하지는 않았다.



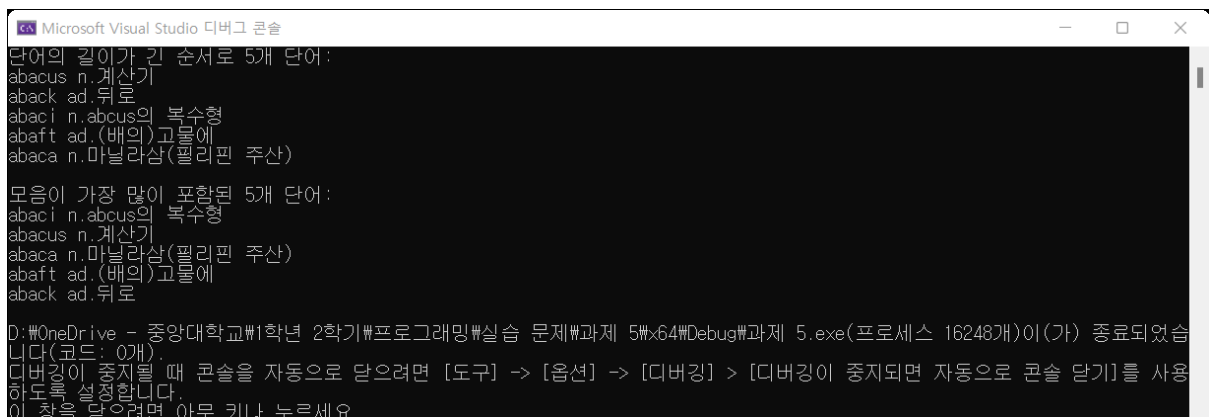
```
Microsoft Visual Studio 디버그 콘솔
단어의 길이가 긴 순서로 10개 단어:
potassium dichromate 중크롬산 칼륨
breach of confidence 비밀 누설
equilateral triangle 등변 삼각형
united arab emirates 아랍 에미리트 연방
pin something on one 아무에게 무엇을 책임을 지우다
reflecting telescope 반사 망원경
reciprocating engine 왕복 기관
four-door saloon car four-door 세단형 차
temperature selector 온도 선택기
physical examination 신체 검사

모음이 가장 많이 포함된 10개 단어:
occupational disease n. 직업병
equatorial guinea 적도 기니
ease someone out 교묘히 해고하다
quadratic equation n. 2차방정식
united arab emirates 아랍 에미리트 연방
semidemisiquaver n. 64분 음표
give a piece of cake 케익 한 조각을 주다
autointoxication n. 자가 중독
autoinoculation n. 자가 접종
equilateral triangle 등변 삼각형

D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 5\64\Debug\과제 5.exe(프로세스 17116개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 7

위 그림 7은 사전 파일을 별도로 조작하지 않고 실행한 결과이다. 단어의 길이가 긴 순서대로 10개 단어, 그리고 모음이 가장 많이 포함된 순서로 10개 단어가 정렬된 것을 볼 수 있다. 물론 문제에서 단어의 길이와 모음의 길이가 같을 때의 처리 방법은 아무것도 제공하지 않아 다른 방법으로 만든 프로그램과 비교해 그 결과가 다를 수도 있다. 당장 이 결과에서도 단어의 길이가 가장 긴 순서로 10개 단어를 출력했지만 10개 단어 모두 단어의 길이는 같고, 그 10개 단어의 정렬 기준은 찾을 수 없다.



```
Microsoft Visual Studio 디버그 콘솔
단어의 길이가 긴 순서로 5개 단어:
abacus n. 계산기
aback ad. 뒤로
abaci n. abacus의 복수형
abaft ad. (배의) 고물에
abaca n. 마닐라삼(필리핀 주산)

모음이 가장 많이 포함된 5개 단어:
abaci n. abacus의 복수형
abacus n. 계산기
abaca n. 마닐라삼(필리핀 주산)
abaft ad. (배의) 고물에
aback ad. 뒤로

D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 5\64\Debug\과제 5.exe(프로세스 16248개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 8

위 그림 8은 사전 파일을 앞 5개 단어만 남기고 모두 지운 상태에서 실행한 결과이다. 사전의 단어 자체가 5개밖에 남지 않았으므로 각 10개 단어가 아니라 5개 단어가 출력된 것을 확인할 수 있다.



## 10. 소스코드

첨부파일 참조

문제별 코드 실행 시 다음 파일이 모두 필요합니다.

문제 1 코드 실행 시: 과제 5 공통 코드, 과제 5 공통 헤더, 문제 1 코드 필요

문제 2 코드 실행 시: 과제 5 공통 코드, 과제 5 공통 헤더, 문제 2 코드 필요

문제 3 코드 실행 시: 과제 5 공통 코드, 과제 5 공통 헤더, 문제 3 코드 필요

공통 코드와 공통 헤더 모두 과제 3의 공통 코드와 헤더를 포함하고 있습니다.