

B03703004 財金五 陳冠宇

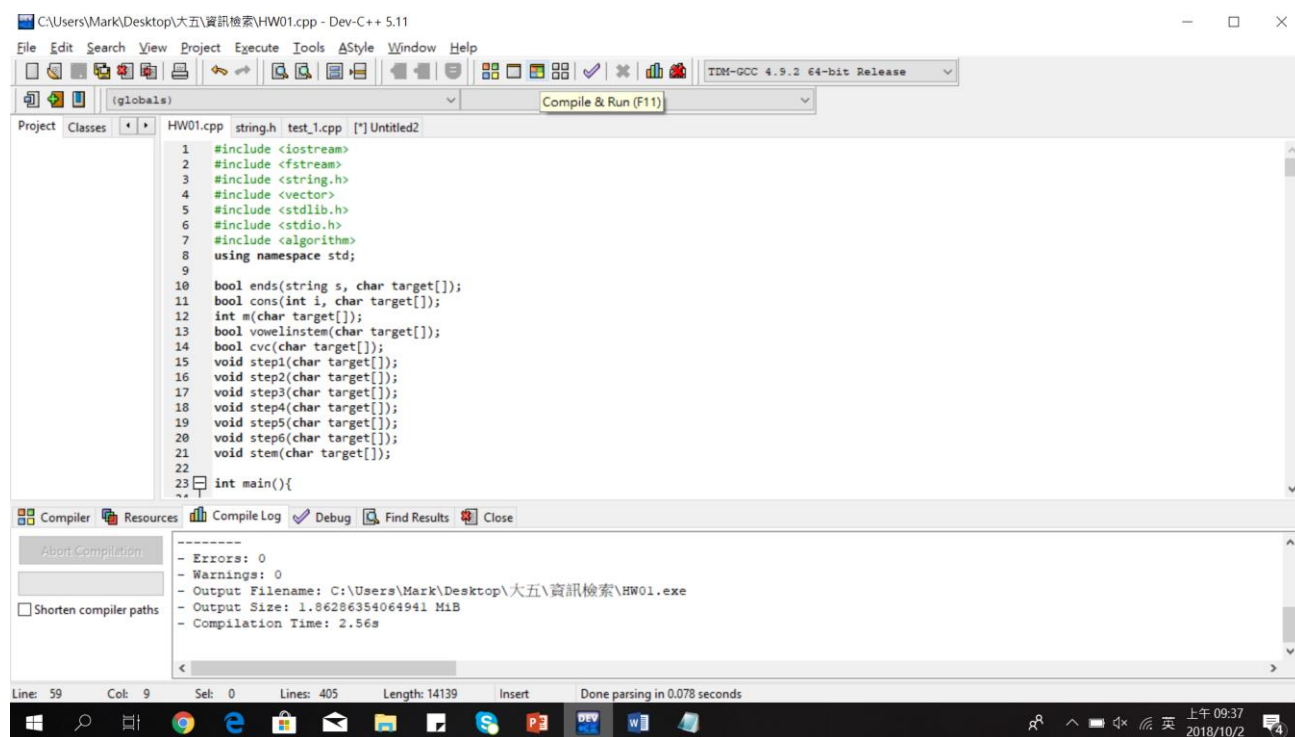
Introduction to Information Retrieval HW01

執行環境：Dev C++ 5.11

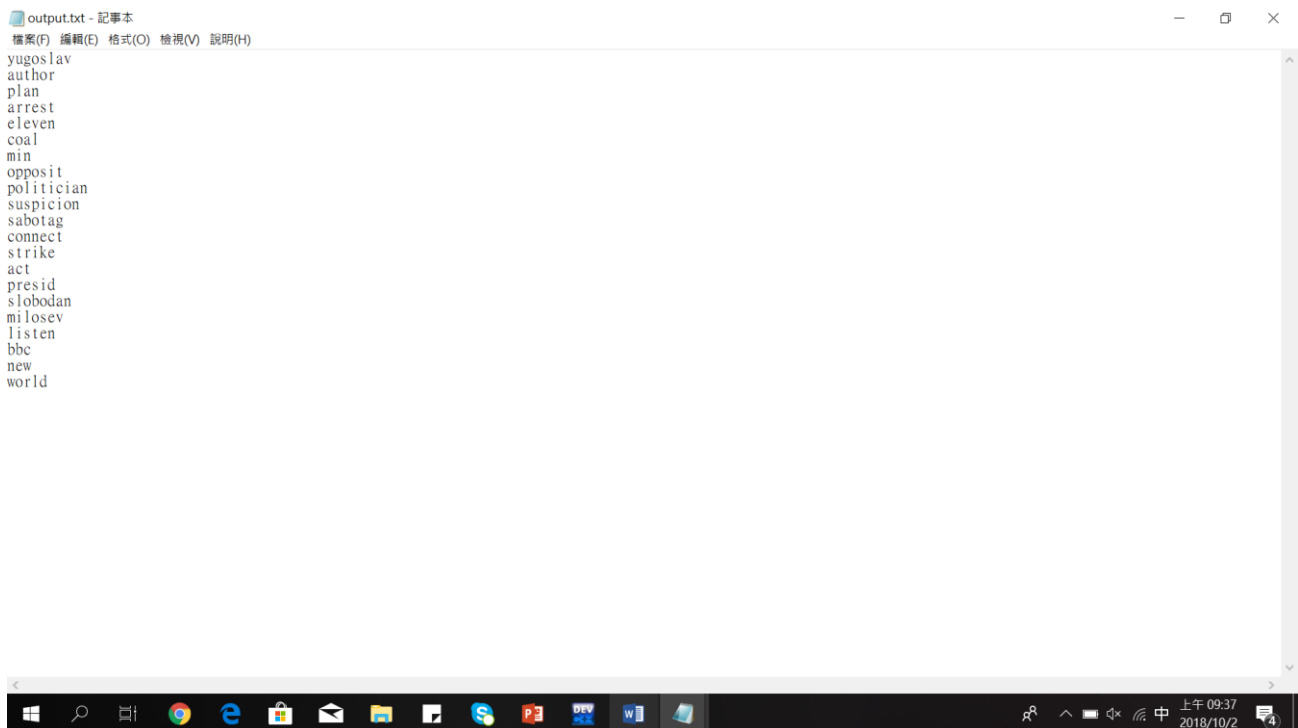
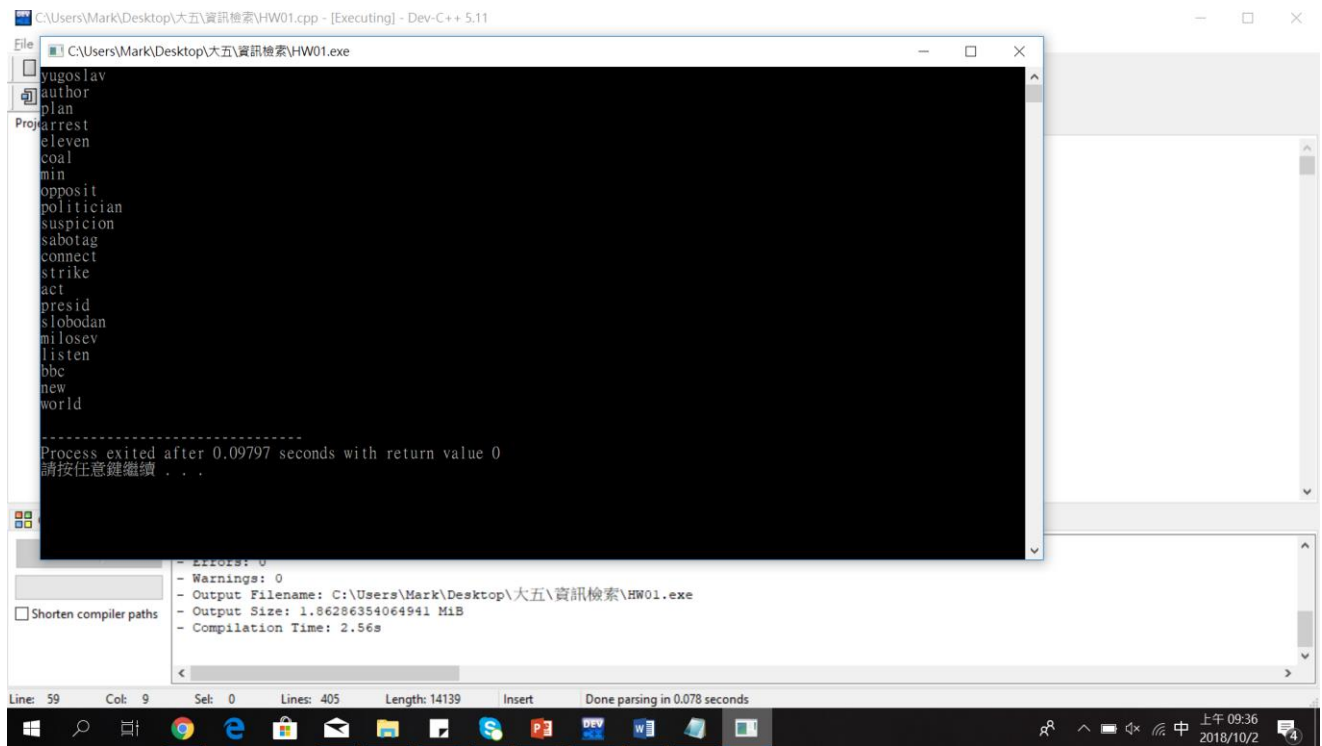
使用語言：C++

執行方式：

1. 主程式 main 內第一行 `ifstream doc("C:\\Users\\Mark\\Desktop\\text.txt"); //read file` 的引號內位置改成需要處理的文件位置
(即 C:\\Users\\Mark\\Desktop\\text.txt 這行改為您的文件讀取位置)
2. 主程式 main 內這行 `ofstream outfile ("C:\\Users\\Mark\\Desktop\\output.txt");` 引號內的位置為最後輸出結果位置，可以改為您要的目的地
(即 C:\\Users\\Mark\\Desktop\\output.txt 這行改為您的文件寫出位置)
3. 用 Dev C++ 5.11 打開 HW01.cpp

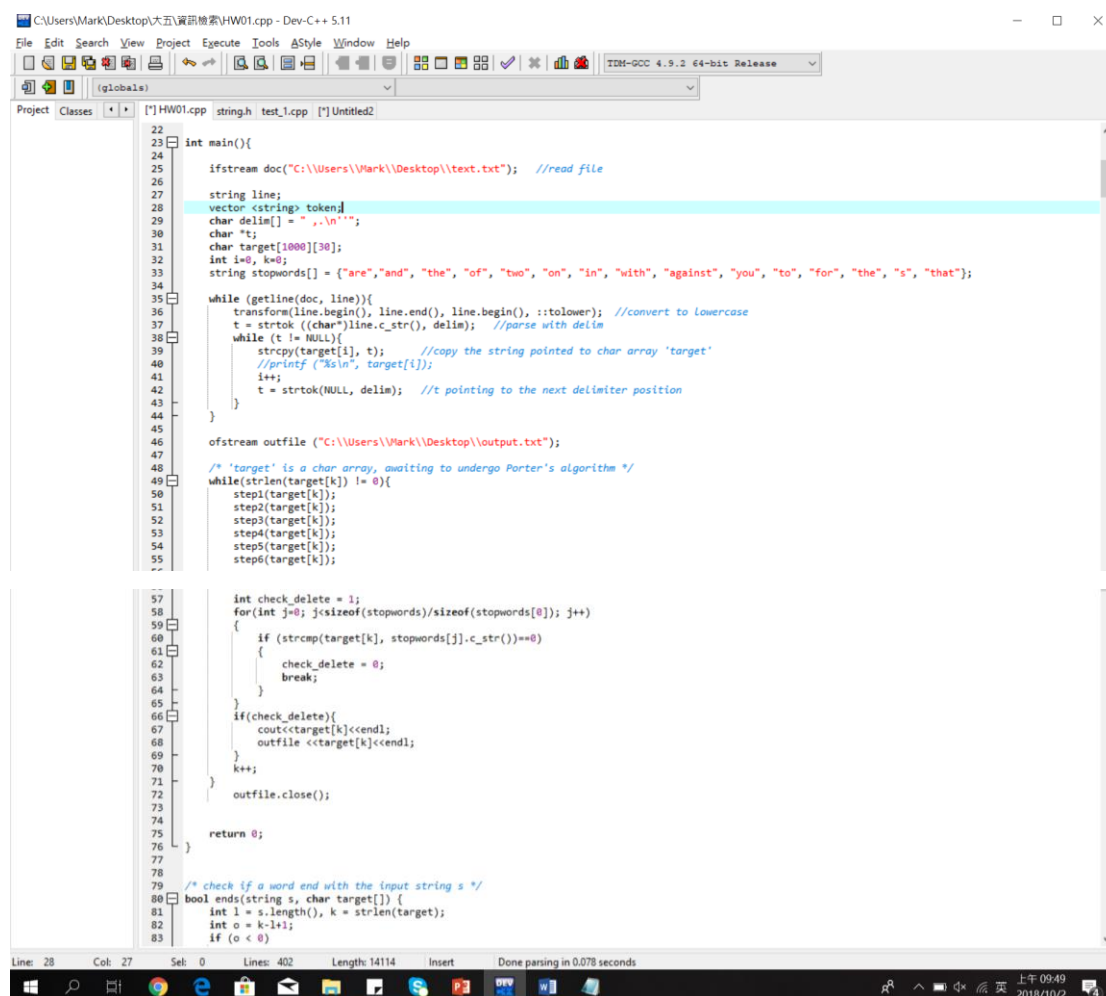


4. 點上方 Compile & Run 即可，最後會顯示出結果，並在(2)的目的地輸出 txt 檔



處理邏輯：

主程式



```
22
23 int main(){
24
25     ifstream doc("C:\\Users\\Mark\\Desktop\\text.txt"); //read file
26
27     string line;
28     vector<string> tokens;
29     char delim[] = " .,\\n";
30     char *t;
31     char target[1000][30];
32     int i=0, k=0;
33     string stopwords[] = {"are", "and", "the", "of", "two", "on", "in", "with", "against", "you", "to", "for", "the", "s", "that"};
34
35     while (getline(doc, line)){
36         transform(line.begin(), line.end(), line.begin(), ::tolower); //convert to lowercase
37         t = strtok((char*)line.c_str(), delim); //parse with delim
38         while (t != NULL){
39             strcpy(target[i], t); //copy the string pointed to char array 'target'
40             //printf("%s\\n", target[i]);
41             i++;
42             t = strtok(NULL, delim); //t pointing to the next delimiter position
43         }
44     }
45
46     ofstream outfile ("C:\\Users\\Mark\\Desktop\\output.txt");
47
48     /* 'target' is a char array, awaiting to undergo Porter's algorithm */
49     while(strlen(target[k]) != 0){
50         step1(target[k]);
51         step2(target[k]);
52         step3(target[k]);
53         step4(target[k]);
54         step5(target[k]);
55         step6(target[k]);
56
57         int check_delete = 1;
58         for(int j=0; j<sizeof(stopwords)/sizeof(stopwords[0]); j++){
59             if (strcmp(target[k], stopwords[j].c_str())==0)
60             {
61                 check_delete = 0;
62                 break;
63             }
64         }
65         if(check_delete){
66             cout<<target[k]<<endl;
67             outfile <<target[k]<<endl;
68         }
69         k++;
70     }
71     outfile.close();
72
73     return 0;
74 }
75
76
77
78
79 /* check if a word end with the input string s */
80 bool ends(string s, char target[]){
81     int l = s.length(), k = strlen(target);
82     int o = k-l+1;
83     if (o < 0)
```

甲、getline 讀 txt 檔

乙、用 transform 函數把全部轉換成小寫

丙、用 strtok 函數斷詞，以 delim[]內的符號作為斷詞依據，存在 character 2D array 'target' 內

丁、進入 while 迴圈，讓每個單詞經過 porter' s algorithm 的處理

戊、與 stopwords 比對，若為 stopword 就不輸出，若不是 stopword 就輸出且寫入輸出的 txt 檔

資料結構：2 維 char array

Library：

```
1 #include <iostream>
2 #include <fstream>
3 #include <string.h>
4 #include <vector>
5 #include <stdlib.h>
6 #include <stdio.h>
7 #include <algorithm>
```

函數介紹：

```
10 bool ends(string s, char target[]);
11 bool cons(int i, char target[]);
12 int m(char target[]);
13 bool vowelinstem(char target[]);
14 bool cvc(char target[]);
15 void step1(char target[]);
16 void step2(char target[]);
17 void step3(char target[]);
18 void step4(char target[]);
19 void step5(char target[]);
20 void step6(char target[]);
```

參考資料：An algorithm for suffix stripping, M.F.Porter, 1980

<https://tartarus.org/martin/PorterStemmer/def.txt>

bool ends(string s, char target[]);

檢查 target 的結尾是否為第一個參數中的 string

bool cons(int i, char target[]);

檢查 target[i] 是否為子音，是的話回傳 true、是母音的話回傳 false。另外，y 的前面如果是子音，他就是母音

int m(char target[]);

用來檢查中間出現幾次 <V><C>，<V> 為連續的母音，<C> 為連續的子音

bool vowelinstem(char target[]);

檢查整個字內有沒有含子音

bool cvc(char target[]);

檢查結尾結構是否為 <子音><母音><不為 XYZ 的子音>

void step1(char target[]);

1(a)

結尾取代規則：

前		後
SS	->	SS
IES	->	I
SS	->	SS
S	->	

1(b)

結尾取代規則：

前		後
(m()>0) EED	->	EE

(contains a vowel) ED ->(a)
 (contains a vowel) ING ->(b)

1(c) 若(a) or (b)成功

結尾取代規則：

前	後
AT	-> ATE
BL	-> BLE
IZ	-> IZE
(結尾重複且非 LSZ)	-> 砍掉最後一個字
(m()=1 and cvc()=true)	-> E

void step2(char target[]);

結尾取代規則：

前	後
(含子音) Y	-> I

void step3(char target[]);

結尾取代規則：

前	後
(m>0) ATIONAL	-> ATE
(m>0) TIONAL	-> TION
(m>0) ENCI	-> ENCE
(m>0) ANCI	-> ANCE
(m>0) IZER	-> IZE
(m>0) ABLI	-> ABLE
(m>0) ALLI	-> AL
(m>0) ENTLI	-> ENT
(m>0) ELI	-> E
(m>0) OUSLI	-> OUS
(m>0) IZATION	-> IZE
(m>0) ATION	-> ATE
(m>0) ATOR	-> ATE
(m>0) ALISM	-> AL
(m>0) IVENESS	-> IVE
(m>0) FULNESS	-> FUL
(m>0) OUSNESS	-> OUS
(m>0) ALITI	-> AL

(m>0) IVITI -> IVE
(m>0) BILITI -> BLE

void step4(char target[]);

結尾取代規則：

前	後
(m>0) ICATE	-> IC
(m>0) ATIVE	->
(m>0) ALIZE	-> AL
(m>0) ICITI	-> IC
(m>0) ICAL	-> IC
(m>0) FUL	->
(m>0) NESS	->

void step5(char target[]);

結尾取代規則：

前	後
(m>1) AL	->
(m>1) ANCE	->
(m>1) ENCE	->
(m>1) ER	->
(m>1) IC	->
(m>1) ABLE	->
(m>1) IBLE	->
(m>1) ANT	->
(m>1) EMENT	->
(m>1) MENT	->
(m>1) ENT	->
(m>1) and (結尾為 S 或 T)	ION ->
(m>1) OU	->
(m>1) ISM	->
(m>1) ATE	->
(m>1) ITI	->
(m>1) OUS	->
(m>1) IVE	->
(m>1) IZE	->

`void step6(char target[]);`

結尾取代規則：

前	後
---	---

`(m>1) E` ->

`(m=1 & cvc()==false) E` ->

`(m > 1 and 結尾重複且非 LL)` -> 砍掉最後一個字

*詳細實作內容請件附檔 HW01.cpp