建立了一個struct來傳入thread的參數

struct data\_term\_frequency{

string list\_id; (DocID)

vector <int> list\_num; (詞頻向量)

double avg\_cosine;

int var = 0;

};

一開始先讀檔，我是使用get的方式來一個一個字元讀取，順便處理詞頻，並且結果是vector<struct>來存

方法如下，

string temp、

int sentence\_num來記錄行數(預設值為0)、

bool error\_letter代表 (預設值為false)，

while (inFile.get(byte))

但byte會有4種情況

1. byte == '\n'
2. byte為英文字母的大寫和小寫
3. byte為0~9的數字
4. byte 為其餘字元

且行數分為兩種奇數(DocID)和偶數(詞)

當byte第一種

奇數行且temp不為空字串

代表這一行結束了 temp值即為此DocID的最後一個單字，計算單字出現的次數，並且將temp、error\_letter值重新預設

偶數行且temp不為空字串

temp值為DocID，vector size變為目前的size多一個

而vector.back()存入DocID值

當byte第二種

奇數行

temp+=byte

偶數行

temp+=byte

當byte第三種

奇數行

error\_letter為true -> 有數字代表此單字不能算

偶數行

temp+=byte

當byte第四種

奇數行

和byte第一種的奇數行處理相同事情

因為byte第一種和第四種的奇數行都在處理單字出現的次數，所以我是用一個function處理，且有一全域變數list\_text (vector<string>)來儲存出現過的單字 ->

首先判斷error\_letter是否為false

false的話繼續處理下面事務，

先int i=0，如果list\_text的size > 0，就先用迴圈看看list\_text有沒有等於temp的值(while(temp != list\_text[i]))，裡面設立了當i值>list\_text的size會中斷，也就是說當迴圈執行完後，i=list\_text的size的話代表temp存的單字是第一次出現，而前面的vector<struct>的list\_num是代表單字出現過的次數，因此所以list\_text和vector<struct>的list\_num都要先把size變成比原來多一，而list\_text最後一個值則為temp，vector<struct>的list\_num的最後一個值則為1；而i!=list\_text的size的話代表temp存的單字已經出現過了，所以直接將vector<struct>的list\_num[i]+1就好。

thread為vector<struct>的size，那我是怎麼計算var的，因為vector<struct>中的list\_num的size每一個不一樣，所以要先決定大小

length = vector[i]的 > vector[j]的 ? vector[i]的: vector[j]的

for (int i=0;i<length;i++)

sum+=vector[i]的list\_num\* vector[j]的list\_num

cosine (i,j)為上述的 sum(i,j)/(sqrt\_count(sum(i,i)\* sum(j,j)))

因為不能用套件或函示庫，所以我建立了一個sqrt\_count的function，而我是透過google查詢開平方計算的公式來寫的。

avg\_cosine則為全部相加並且除以(vector<struct>.size()-1)

那這些和thread該列印出來的都處理完後，就是要找出關鍵文件，因為先前vector<struct>中有一個是儲存avg\_cosine的值，而值等thread都執行完，才會有，那我是就是要找到這個avg\_cosine在vector<struct>中的第幾個

我先預設max為vector<struct>[0]. avg\_cosine，和一個vector<int> max\_i來儲存(vector<struct>中的第幾個)，

如果某個值(i)的avg\_cosine大於max則max被取代，且max\_i要先被clear，再將size變為1，然後max\_i.back()為剛才那個值(i)

如果某個值(i)的avg\_cosine等於max，則max\_i的size多一個，max\_i.back()為剛才那個值(i)

因為關鍵文件是DocID小優先，而剛才的max\_i僅僅是代表在vector<struct>中的第幾個，所以需要一個for迴圈來比較彼此vector<struct>的list\_id的大小

CPU Time我是透過clock()來處理，先設立一個const clock\_t begin=clock()在頂部，並在尾端以(clock()-begin)/ CLOCKS\_PER\_SEC得到CPU Time運行的秒數，最後乘以1000，變為毫秒。

每個項目都有做到

i. 從命令列讀入檔名參數。

ii. 能產生正確數量的 pthread。

iii. 子執行緒可以印出本身的tid。本項滿分10分。

iv. 正確計算出文件的詞頻向量。本項滿分20分。

v. 正確進行餘弦相似係數計算。程式碼中不可以使用任何套件或函式庫，必須有完整的程式

碼。

vi. 每一個 thread 都印出執行過程所用的總共CPU時間，以ms為單位。

vii. 主執行緒找出關鍵文件並印出它的平均餘弦相似係數。

執行方式:

g++ s1091447\_02.c -lpthread -o out

./out test.txt