資料結構 HW1

```
35 int main(int argc, char* argv[])
36 {
37     // require two arguments
38     if(argc < 3) return 0;
39
40     // input and output file
41     ifstream input;
42     input.open(argv[1]);
43     ofstream output;
44     output.open(argv[2]);
```

main function，從命令列讀取 argument，如果 argument 的數量小於三（執行檔案名、輸入文件名、輸出文件名）的話，因為無法正確執行程式而結束程式。用 std::fstream 開啟輸入文件和輸出文件。

```
46     // create a char array and read input
47     char* str = new char [1000001];
48     input.read(str, 1000000);
49
50     // avoid punctuation and space
51     int str_size = strlen(str);
52     for(int i = 0; i < str_size; i++){
53         if(ispunct(str[i]) || isspace(str[i])) str[i] = ' ';
54     }
```

宣告一個長度為 1000001 的 char array，並從輸入文件中讀取所有的字元。
為了分隔字，再將 char array 裡所有非 alphanumeric 的字元轉成空白。

```
56     // get pattern
57     char *pat, *s;
58     char delim[] = " ";
59     pat = s = strtok(str, delim);
60     int pat_size = strlen(pat);
61
62     // build failure function
63     int* fail = new int [pat_size];
64     failure_function(pat, pat_size, fail);
```

宣告兩個字串指標，一個存放 pattern，一個要被比對的字串。再算出 pattern 的 failure function，以提供接下來的比對用。

```
66     // freq : the number of the words same as the pattern
67     // word_idx : the index of the current word
68     int freq = 0, word_idx = 1;
69     stringstream out;
70     while(s != NULL){
71         // match pattern
72         int s_size = strlen(s);
73         int match_time = match(s, s_size, pat, pat_size, fail);
74
75         // output
76         if(match_time){
77             for(int i = 0; i < match_time; i++) out << word_idx;
78             if(s_size == pat_size) freq++;
79         }
80
81         // get new word from input
82         word_idx++;
83         s = strtok(NULL, delim);
84     }
```

宣告比對完全相同次數的變數 freq，還有目前在比對第幾個字串的變數 word_idx。stringstream out 用來儲存輸出。在 while 迴圈中，將要被比對的字串與 pattern 做一次 kmp，算出有幾個子字串和 pattern 相同。如果有比對成功（即 match_time != 0）就將 word_idx 寫 match_time 次進入輸出，而且如果要被比對的字串的長度，與 pattern 的長度一樣的話，代表兩字串相同，freq 加

一。接下來，word_idx 加一並用 strtok 找出下一個字串。當 strtok 找完整個字串之後會回傳一個 NULL 給 s，使得程式跳出 while 迴圈。

```
85      // output result
86      output << freq << endl;
87      string output_string;
88      out >> output_string;
89      output << output_string << endl;
90      input.close();
91      output.close();
92      delete [] str;
93      return 0;
```

先將 freq 寫入檔案，再將比對成功的字的位置以 1-based 輸出寫入檔案，關閉檔案並釋放使用過的記憶體空間，結束程式。

```
// generate failure function
void failure_function(char* pat, int size, int* fail)
{
    fail[0] = -1;
    for(int i = 1, j = -1; i < size; i++){
        while(j >= 0 && pat[j + 1] != pat[i]) j = fail[j];
        if(pat[j + 1] == pat[i]) j++;
        fail[i] = j;
    }
}
```

```
19 // match the pattern and the word, return the match times
20 int match(char* s, int s_size, char* pat, int pat_size, int* fail)
21 {
22     int match_time = 0;
23     if(s_size < pat_size) return 0;
24     for(int i = 0, j = -1; i < s_size; i++){
25         while(j >= 0 && pat[j + 1] != s[i]) j = fail[j];
26         if(pat[j + 1] == s[i]) j++;
27         if(j == pat_size - 1){
28             j = fail[j];
29             match_time++;
30         }
31     }
32     return match_time;
33 }
34
```

failure function 是找出在從零開始到 i 位置的字串，次長前綴後綴長度-2 的值，也就是上一次比對成功的 index-1。

match 則是利用 failure function 找出與 pattern 相同的子字串的數量。