**資訊三甲 10427101 馬若芸**

1. **開發平台**

Windows 10。

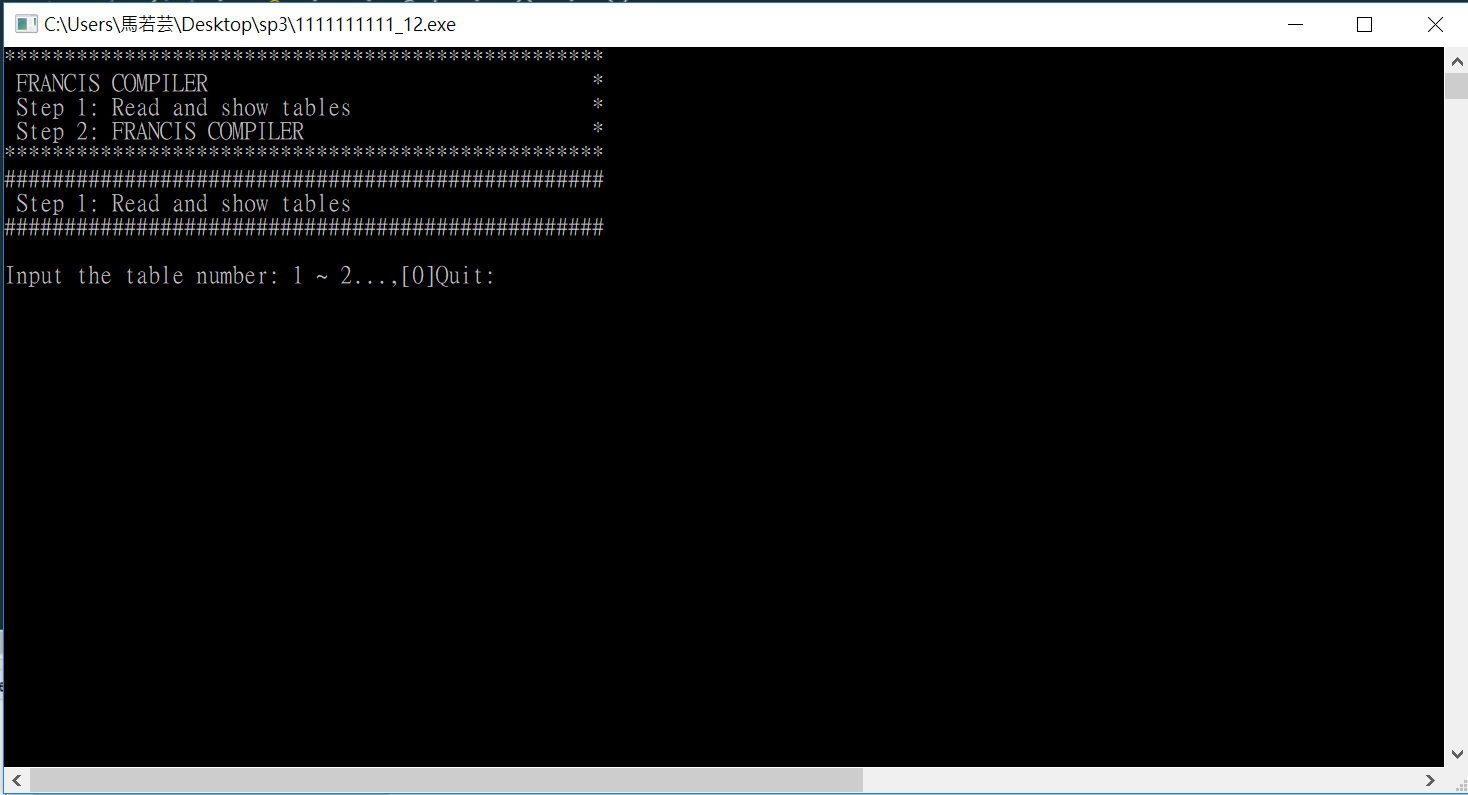
1. **使用開發環境**

Dev-C++。

1. **使用的程式語言**

C、C++。

1. **程式設計說明**



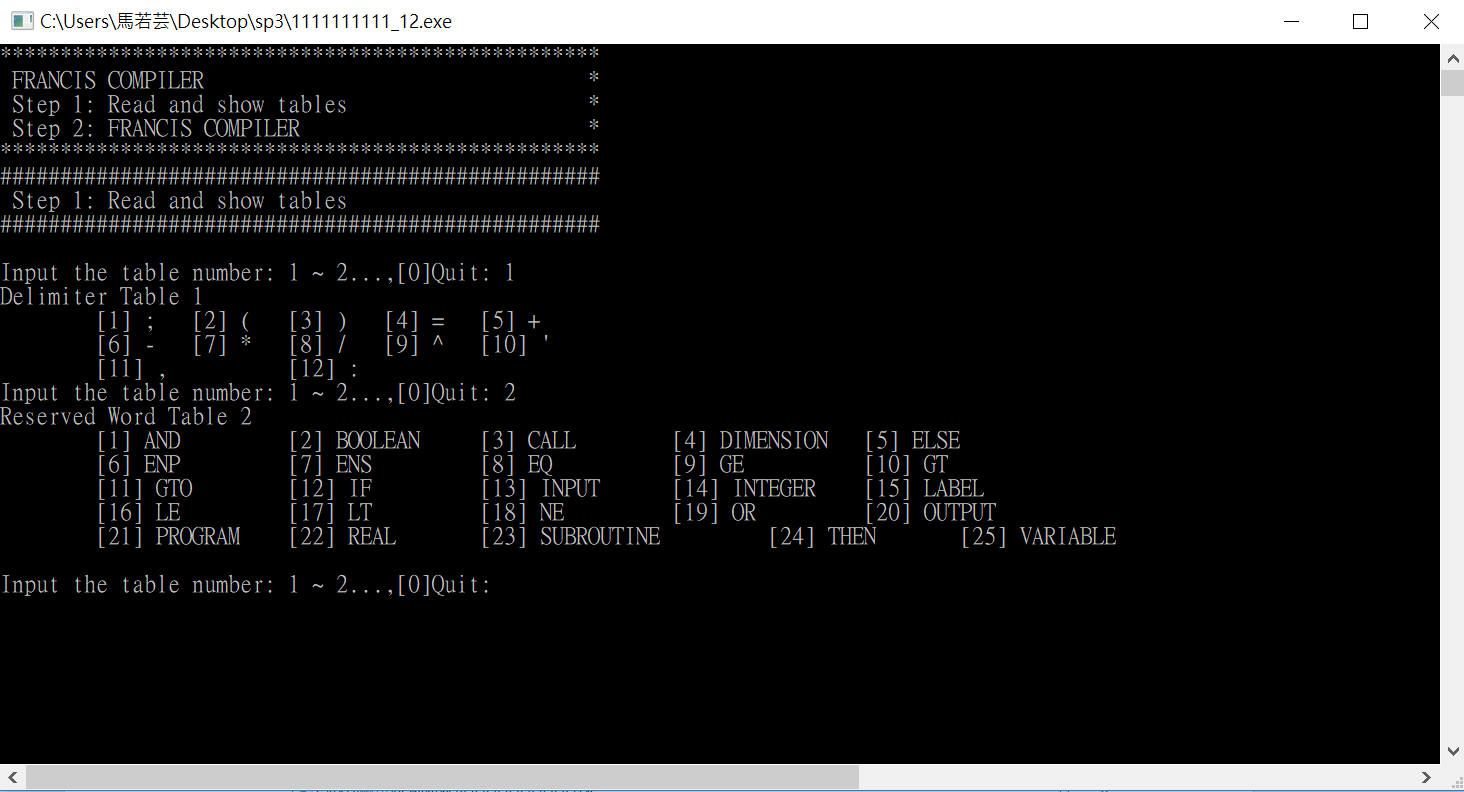
**(初始畫面)**

* 功能

程式分成兩步驟進行，先輸入完需要的delimiter、reserved word table後，即可進行FRANCIS Compiler。

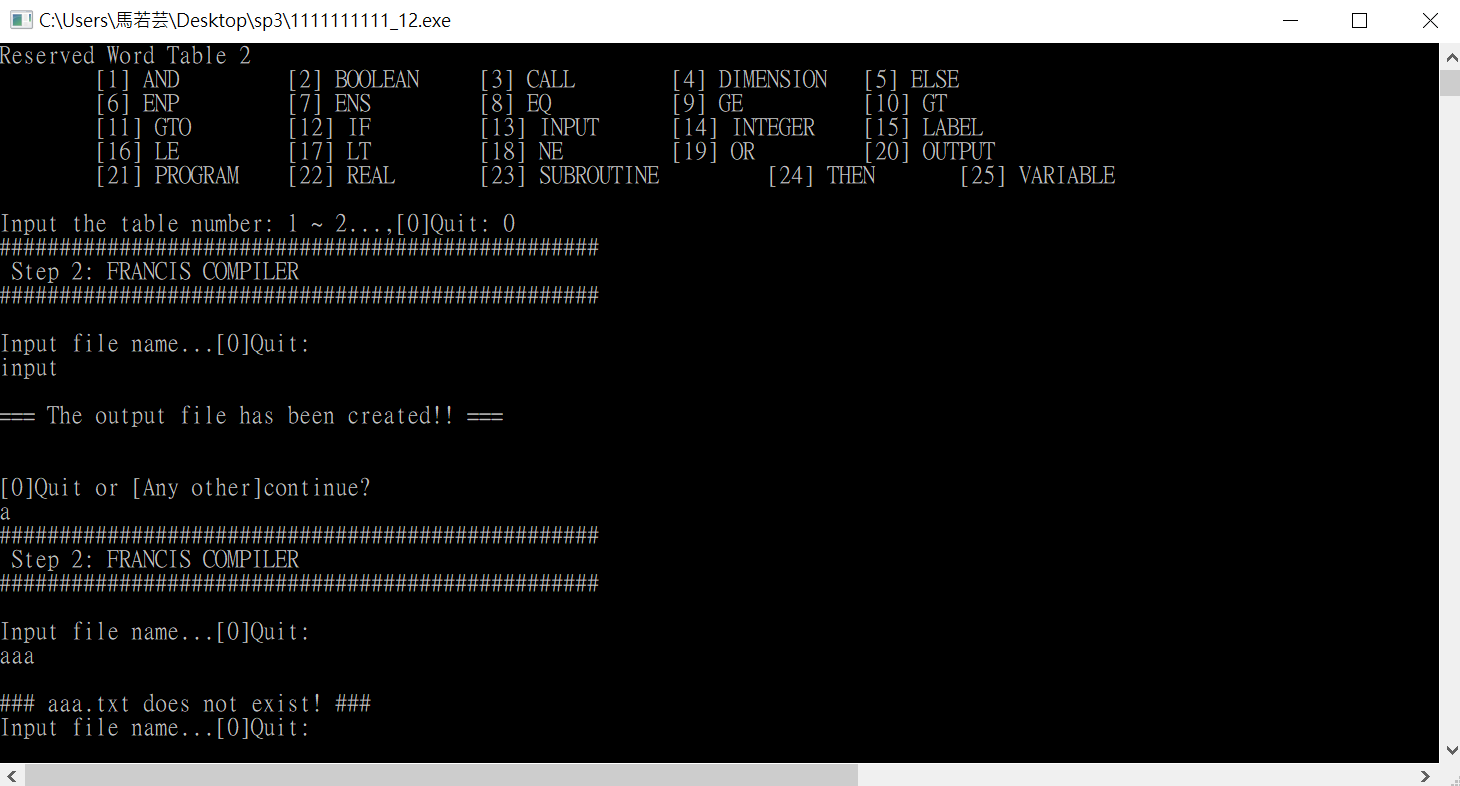
* 使用流程

1. 輸入並讀入提供的table1~2，每輸入table的編號後會輸出table的內容(圖二)。輸入需要的delimiter、reserved word table後，輸入”0”進行下一步驟。



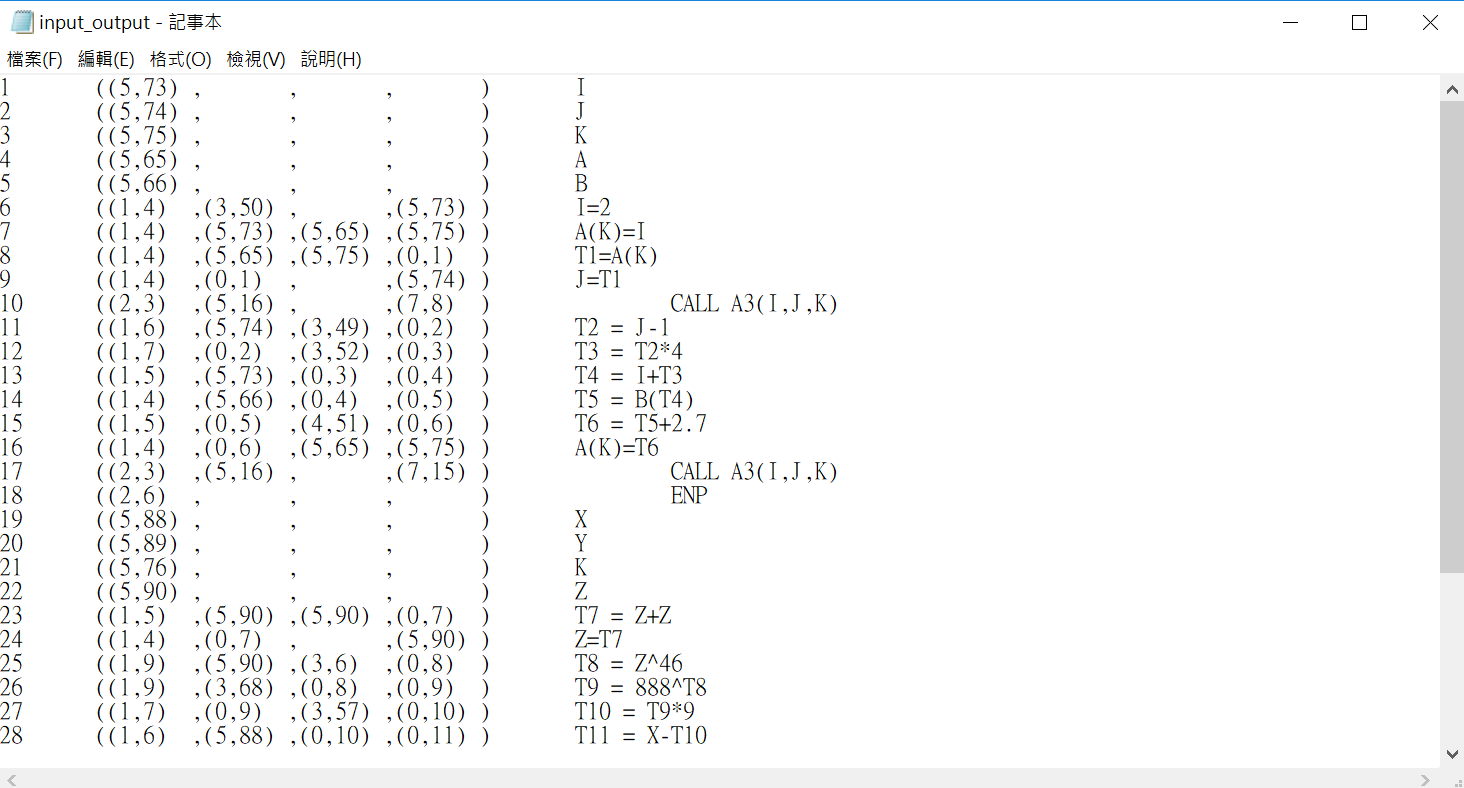
(圖二)

1. 完成步驟一接著開始FRANCIS Compiler。輸入檔案名稱，若找到該檔名並執行成功則出現”=== The output file has been created!! ===”字樣；若無此檔案則出現”### 名稱.txt does not exist! ###”(圖三)。



(圖三)

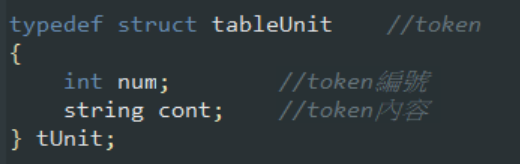
1. 完成FRANCIS Compiler後若還想再執行下一份資料時，輸入任意見繼續執行，輸入”0”結束離開。
2. FRANCIS Compiler完成產生的檔案，檔名:輸入檔\_output.txt，如在步驟二輸入”file1\_input”，則輸出檔為”file1\_input\_output”。輸出檔內容格式為每行包含欄位行數、產生的中間碼、與中間碼對應的原始程式敘述(圖四)。



(圖四)

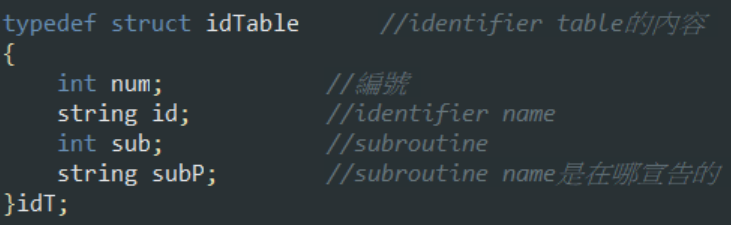
* 使用的data structure

Table的struct為tableUnit，tableUnit內有紀錄token號碼的num和此token內容的cont。(如下圖)

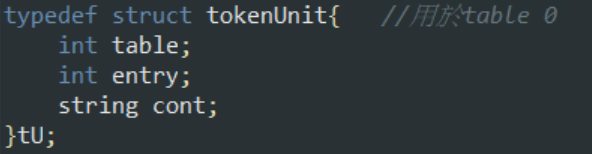


Delimiter、reserved word 皆使用vector儲存，每讀入table中一個token依類型放入相對應的vector中，兩個table對應的vector分別為tSetDeli、tSetRW。剩下的integer、real number是用linear hash方式存入空間大小為100的陣列中，相對應的陣列分別為tSetInt、tSetRN。在進行lexical analysis時，若在對應的陣列中沒有該token，則使用mod100的hash function放入hash table中，若產生碰撞則向後遞增至空的位置。

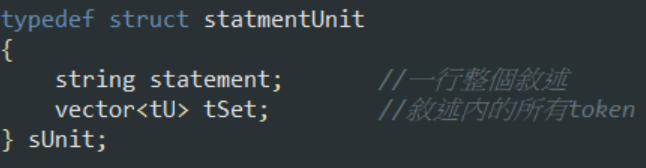
針對identifier table的idTable為了記下scope，所以以下此struct是負責記錄identifier table並mod 100用linear hash方式存入空間大小為100的陣列中(tSetId)。內容有編號(num)、identifier name(id)、scope(sub)、subroutine name(subP)。 (如下圖)



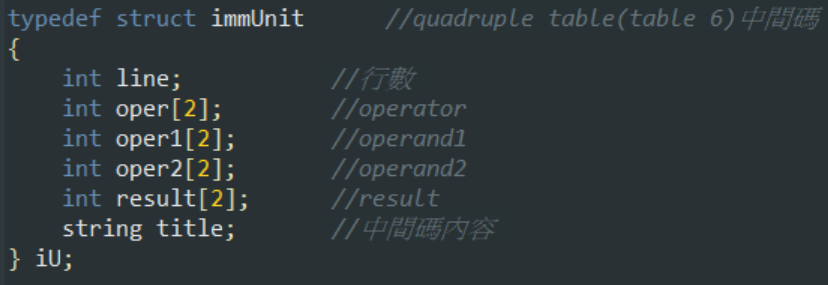
針對table 0產生的token的struct tokenUnit，利用vector(tSetLab)儲存，lexical analysis(table、entry)、token內容(cont)。(如下圖)



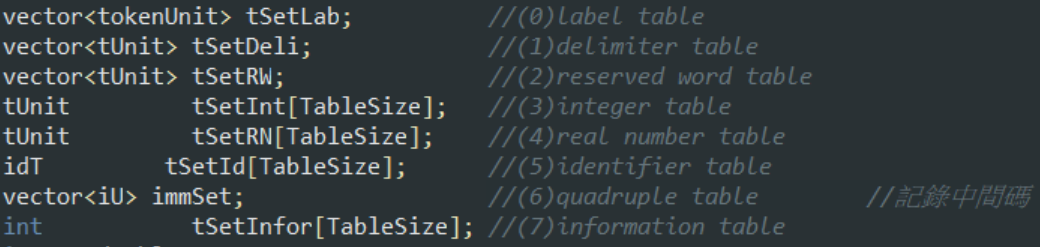
接著這個struct statementUnit是除了存一整個敘述外(statement)，還存此敘述內所有的token(tSet)，因為一個敘述內有多個token，所以用vector儲存，以便作lexical analysis。(如下圖)



struct immUnit負責儲存中間碼的內容，內有中間碼編號(line)、operator’s lexical analysis(oper)、operand1’s lexical analysis(oper1) 、operand2’s lexical analysis(oper2)、、result’s lexical analysis(result)、與中間碼對應產生的程式敘述(title)。(如下圖)



以下圖示表示所有table的生成型態:



在處理assignment使用的reverse polish notation，用到的資料結構是以vector<string>構成的operand、operators分別當作operand stack和operator stack，並根據reverse polish notation規則，適時地把運算元、運算子用vetcor.push\_back()進stack，以及要產生中間碼時用vetcor.pop\_back()從stack拿出。

* 程式碼解說

**Function說明:**

1. bool ReadWriteFile():讀進input檔，每一次都只讀一行，讀入該行後呼叫GetToken()進行切token，直到讀完input檔結束，並開始寫入output檔。若成功讀寫檔案，回傳true讓螢幕輸出完成訊息；若失敗，回傳false。
2. void WhichTable():根據輸入的table編號，來決定table內容要放入對應的vector中，並把table的內容輸出至螢幕上。
3. void InputTable():讀進table檔，一個個放入vector中，最後再呼叫WhichTable()來決定是vector是屬於哪個table的。
4. void CheckTable():分析token，用迴圈一一比對來找該token屬於哪一table的內容，若找到則把分析結果寫入output檔中並return；若無表示此token可能因為沒有以white space間隔導致是多個token組成的ex: X=X+2，或是屬於需要使用hash的table但未在該table中的token，所以需要呼叫ExtraTable()來作進一步的切割、分析。
5. void ExtraTable():把在CheckTable()無法完成分析的token來依不同情況再切割，切割完成後呼叫CheckTable()進行分析。還有要新增在table的token，會呼叫insertHashLinear()來新增至hash表。
6. void GetToken():因為每一次都讀取一行，所以用white space分割一個個token，呼叫CheckTable()分析token。
7. bool isDeli():檢查token是否為符號，若是則回傳true。
8. bool isNum():檢查token是否為數字，若是則回傳true。
9. int SetHashKey():計算hash值，hash functoin是把token中每一字元的ASCII碼相加後除以100(hash table size)取餘數，最後回傳該token的hash值。
10. void insertHashLinear():先呼叫SetHashKey()算出hash值要放入哪一位置，若產生碰撞則向後遞增至空的位置，並把完成的結果寫入output檔中。
11. void resetList():初始化hash table。
12. void InsertIdentifier():建立identifier table，記錄變數的編號、名稱、scope。
13. void insertHashLinearInfor():建立information table，依序記錄參數個數、變數類型、其他等。
14. void InsertTable0():建出作assignment時會出現的table 0內容，ex: T1、T2…..。
15. int NumForDeli():根據符號的優先次序來給定數字，方便比大小。
16. bool CheckDeli():處理assignment，比符號大小，回傳true表示stack外的符號比stack裡的符號大，所以要放入stack；若回傳false，則從stack裡pop出符號來產生中間碼。
17. tU ArrayToken():處理array計算，將新增運算元產生lexical analysis，例如遇到B(I,J)，必須新增”(J-1)\*M+I”並分別放入operator、operand的stack中以便產生中間碼。
18. void ForArray():把從ArrayToken()得到的operator、operand stack，從中pop出來產生中間碼。
19. CheckLabel():檢查使用變數前是否已被宣告，有一labelList專門儲存已被宣告過的變數，所以當呼叫的變數名稱在labelList中找不到時就回傳false，表示使用一個沒有宣告過的變數；回傳true則反之。
20. int IFstatement():處理條件式IF之後的statement，處理statement與SetImmCode()的方法相似。
21. void ForGTO():做GTO的forward reference，來補回GTO之後的變數在table 6的位置。
22. void SetImmCode():根據FRANCIS的文法規則，產生出相對應的中間碼，並存入immSet中。

**main說明:**

呼叫ReadWriteFile()來進行上述一連串各function間運作，若ReadWriteFile()回傳true表已成功輸出output檔。

1. **未完成功能**

1.沒處理lexical error: 因為找不到該在哪裡設定檢查lexical error。

2.次方符號(↑)不能讀: 這個符號是unicode，考慮到作業期限所以放棄處理，改用”^”來撰寫FRANCIS compiler。

3.assignment 不能處理三維以上array: 雖然照理說寫的方法是以多維陣列計算所在的位置，不過實際執行程式卻當機了，因為之前寫得再加上程式碼落落長，變的很難除錯，所以目前只能夠處理到二維陣列。

1. **心得**

這次得作業也是花了不少時間、趕在最後才勉強完成的，畢竟聽說下學期的PL會比這個作業難上好幾倍，心想如果這時候不完成的話，未來PL要過的話是希望渺茫，所以才努力壓抑住想放鬆的心情，考完期末考趕回來繼續寫這個作業。過程中會因為也有其他科目要作作業，所以寫的過程中總是斷斷續續的，有些寫法現在回顧還搞不清楚在寫甚麼，寫得有零零落落的，不過整體來看其實FRANCIS compiler寫起來並不是很困難，如果有更多的時間的話，可能能夠再改善一些寫法，讓整體程式碼不會看起來很繁複。