程式語言與編譯器報告

作業一

(Programming Assignment1 - MiniJ)

系級：資工二

組員學號&姓名：

410921202 林芷萱

410921309 陳采瑜

1. The problem description

　　此次作業要求我們使用flex及bison完成MiniJ的前端編譯器，我們需要完成的部分有.l檔(lex)及.y檔(yacc)。先使用flex生成scanner(掃描器)，再使用bison讀取scanner分析出來的token(記號)建立並返回一顆“Parse Tree”(也稱作syntax tree)，詳細的關係圖如下圖一。



* 圖一、scanner及parser關係圖

　　在這次作業中，當完成.l檔及.y檔後，使用CMD(命令提示字元)叫出flex打上參數-o將.l檔轉成.c檔，並叫出bison打上參數-d，這個參數的意思是將.y檔編譯的結果分拆成.c檔及.h檔，此時應有三個.c檔(另一個為作業檔案夾裡附上的主程式)及一個.h檔，再叫出gcc將三個.c檔都轉成.o檔後，將三者link在一起，最終會產生一個叫做“mjparse”的.exe檔。

\*補充1：flex的前身為lex；bison的前身為yacc。

\*補充2：flex的作用為lexical analysis(詞法分析)；

bison的作用為syntactic analysis(語法分析)。

1. Highlight of the way you write the program

minij\_lex.l

這個檔案主要參考的資料有：

1. lex維基百科(<https://zh.wikipedia.org/wiki/Lex>)
2. The Fundamentals of lex Rules中的lex Operators部分(<https://docs.oracle.com/cd/E19504-01/802-5880/lex-6/index.html>)
3. Lex Practice (<https://www.epaperpress.com/lexandyacc/prl.html>)

此檔案的結構分為三大塊，以%%來區隔：

定義區塊 // 用來定義巨集以及匯入C寫成的標頭檔

%%

規則區塊 // 將樣式(patterns)與C的陳述串連在一起

%%

C程式碼區塊 // C的陳述與函式

以下是關於lex結構的詳細說明：

* **定義區塊**可細拆分成兩個區塊：

1. %{資料結構&函式定義&匯入標頭檔}%
2. 寫在(1)下方的token名稱及其匹配方式，舉例：

NONNL [^\n] // NONNL：「不是換行」，^為NOT的意思

* **規則區塊**是最重要的區塊，當lexer看到輸入裡面有合乎給定的樣式時，則會操作相對應的C程式碼。

在這次作業中我們要在這裡寫10個rules，其中

"//"{NONNL}\* { /\* DO NOTHING \*/ }

　　因為在此要引用定義區塊的NONNL，故在此須將其用{}框起來。

　　在寫code的時候透過參考資料來知道lex的編譯規則，以完成作業需要完成的部分，下圖為根據參考資料整理出來.l檔中主要會用到的部分：



* 圖二、lex常規表達式

minij\_parse.y

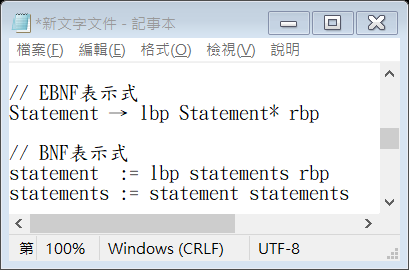
這個檔案主要參考的資料有：

1. Bison Grammar Files (<http://web.mit.edu/gnu/doc/html/bison_6.html#SEC34>)
2. lex yacc 學習(<https://iter01.com/178869.html>)
3. 以lex/yacc實作算式計算機中將EBNF表示式改寫成BNF再改寫成給yacc看的定義檔的部分

(<http://good-ed.blogspot.com/2010/04/lexyacc.html>)

　　yacc的結構跟lex非常類似，在規則區塊的部分lex放置的rules就是每個正規表示式要對應的動作，一般是返回一個token；而yacc放置的rules就是滿足一個語法描述時要執行的動作。

　　作業的要求為將The MiniJ Grammar(EBNF表示式寫成)改寫成給yacc看的定義檔。在此我的作法為參考資料3中的做法，以下為我的解題步驟：



* 圖三、stmt改寫部分



* 圖四、exp及expr改寫部分

再將其改為符合Bison文法規則的格式，其文法規則一般形式為：

*result*： *rule1-components...*

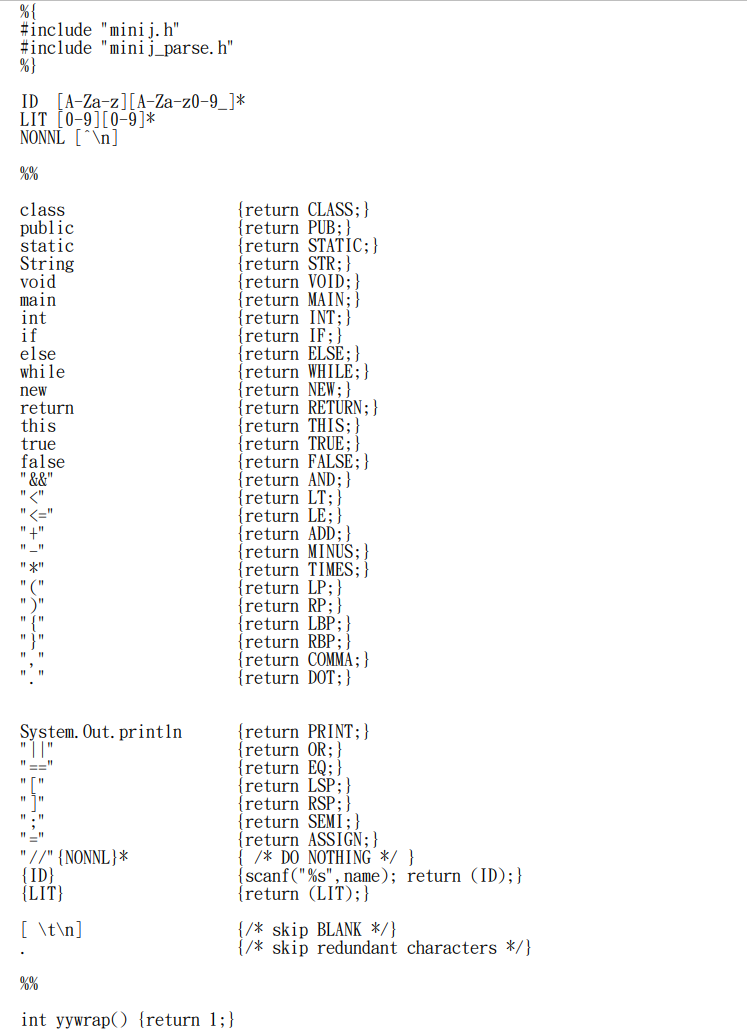
| *rule2-components...*

...

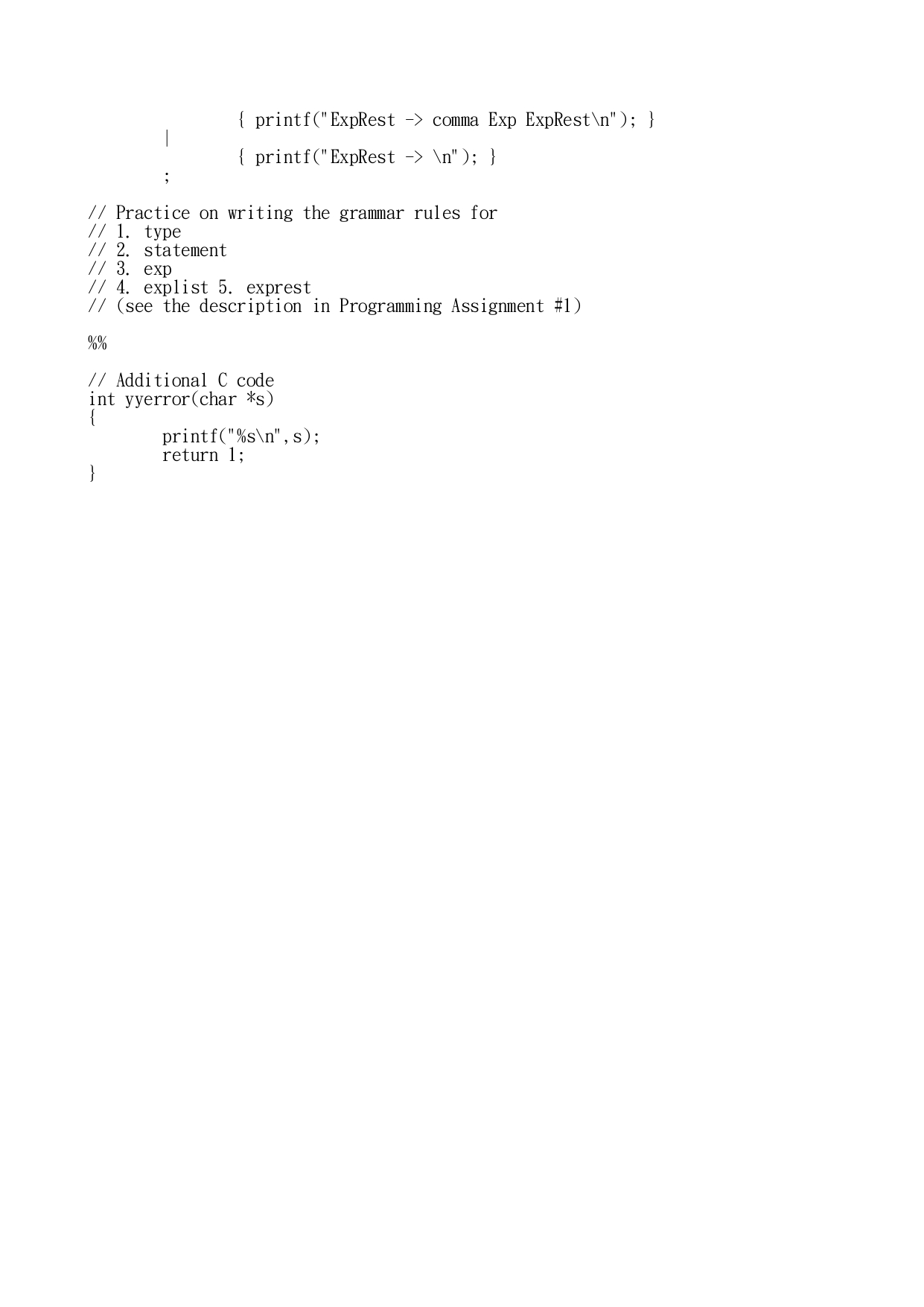
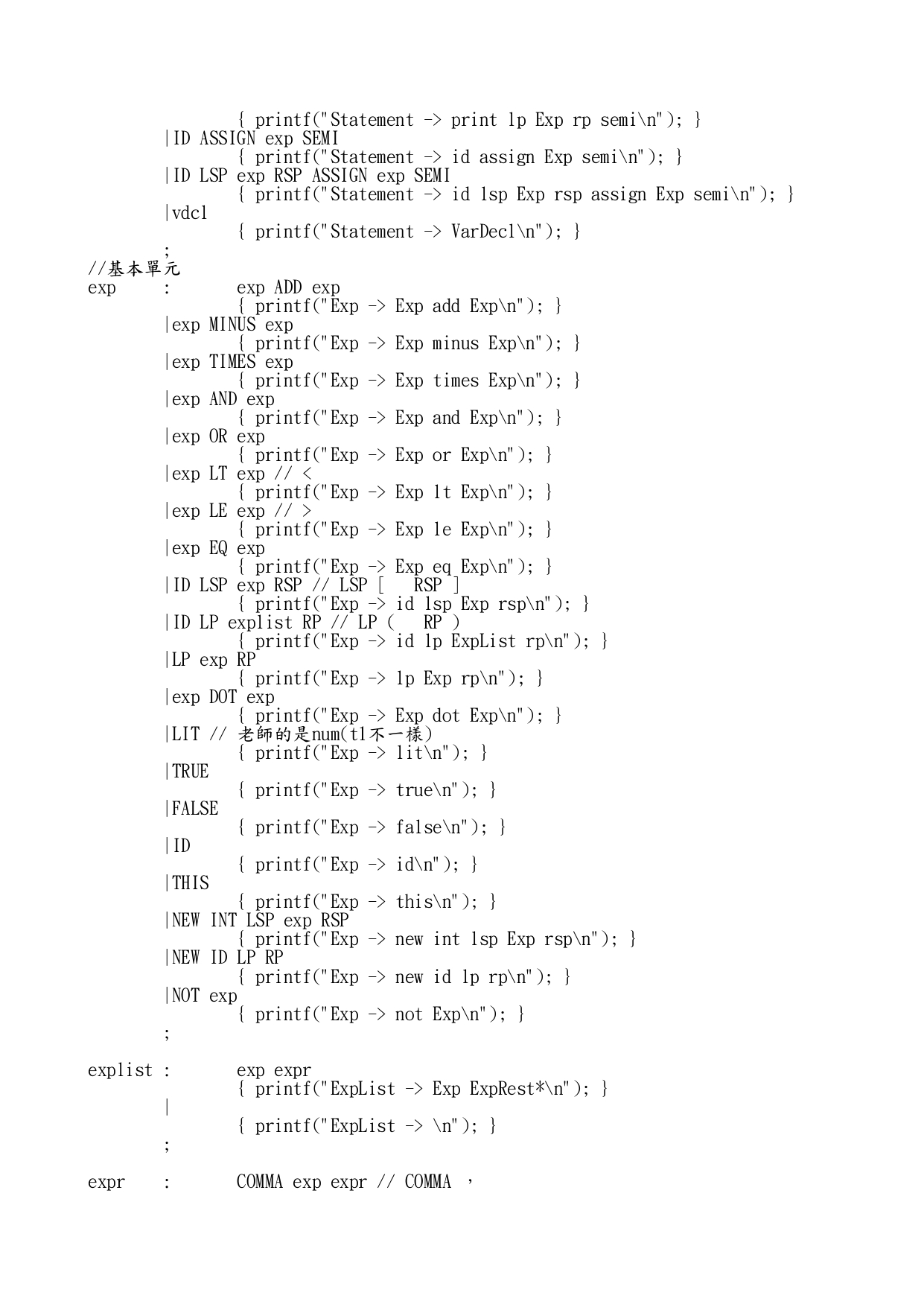
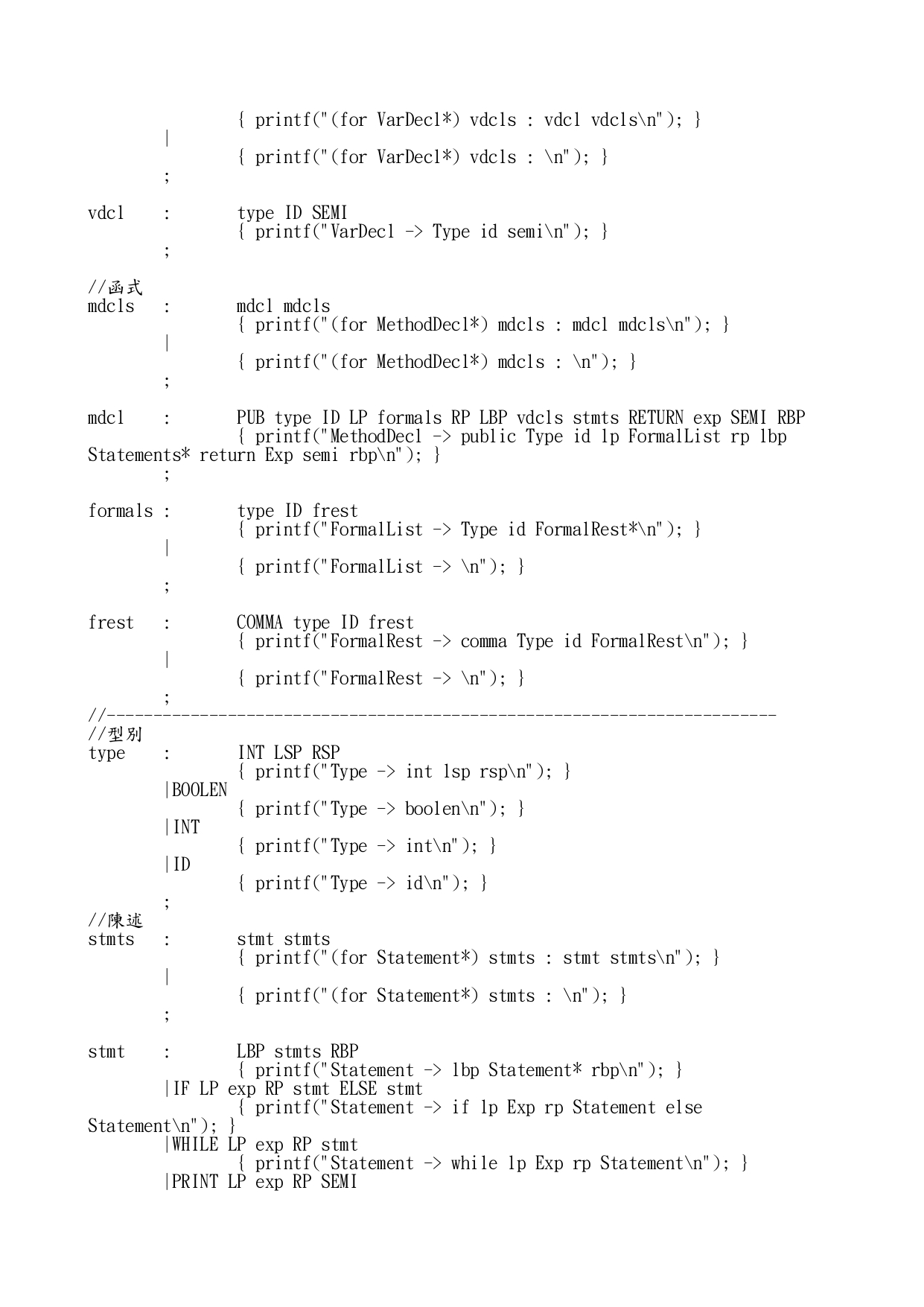
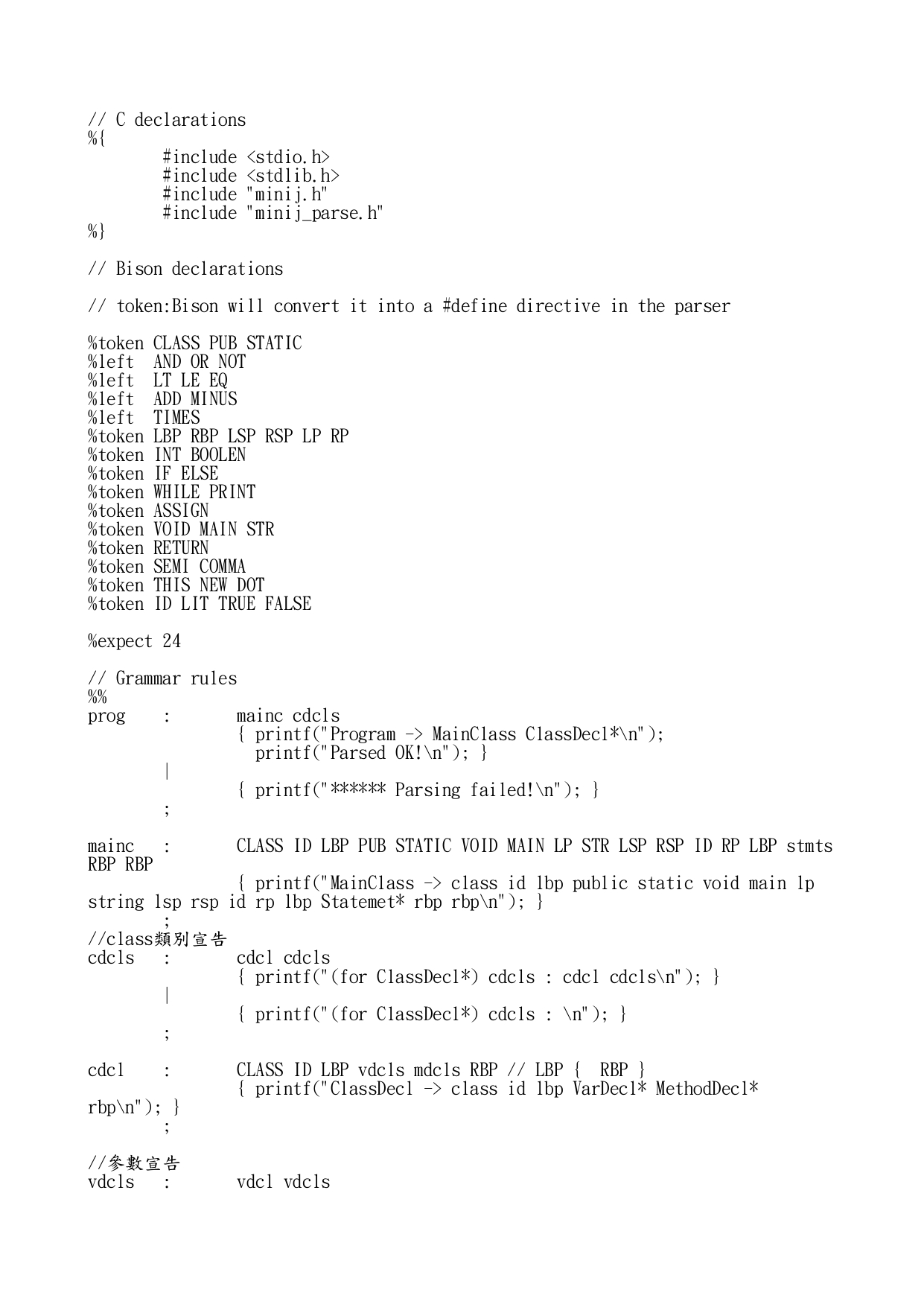
;

1. The program listing

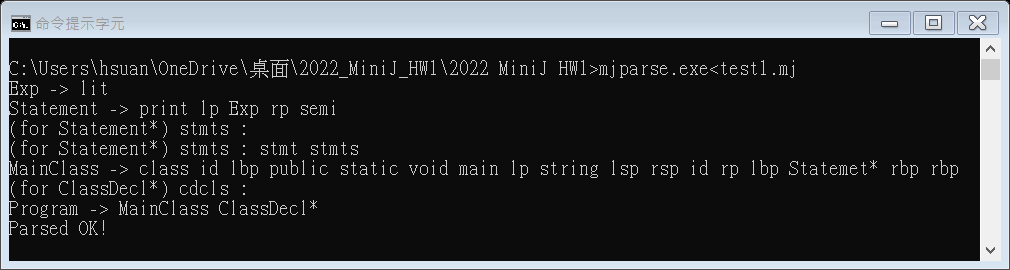
minij\_lex.l程式碼



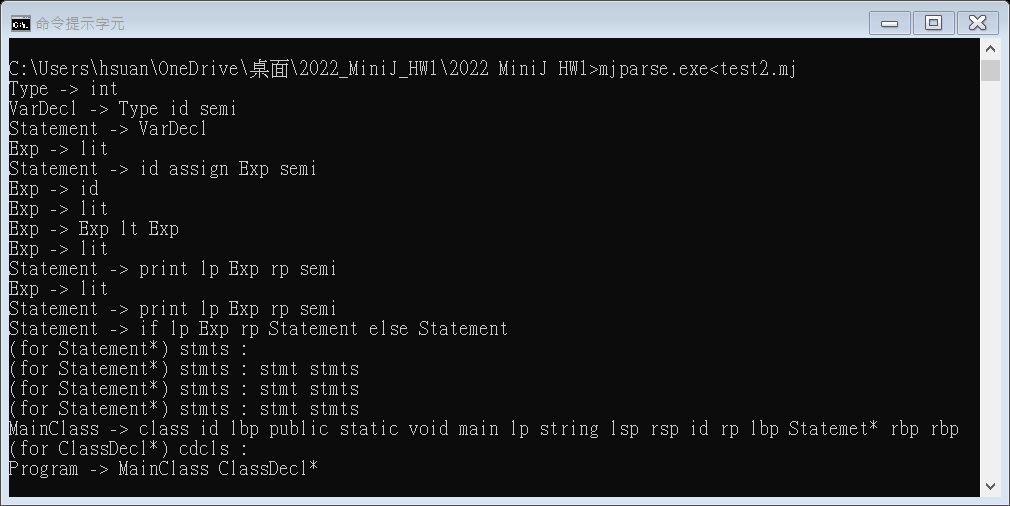
minij\_parse.y 程式碼



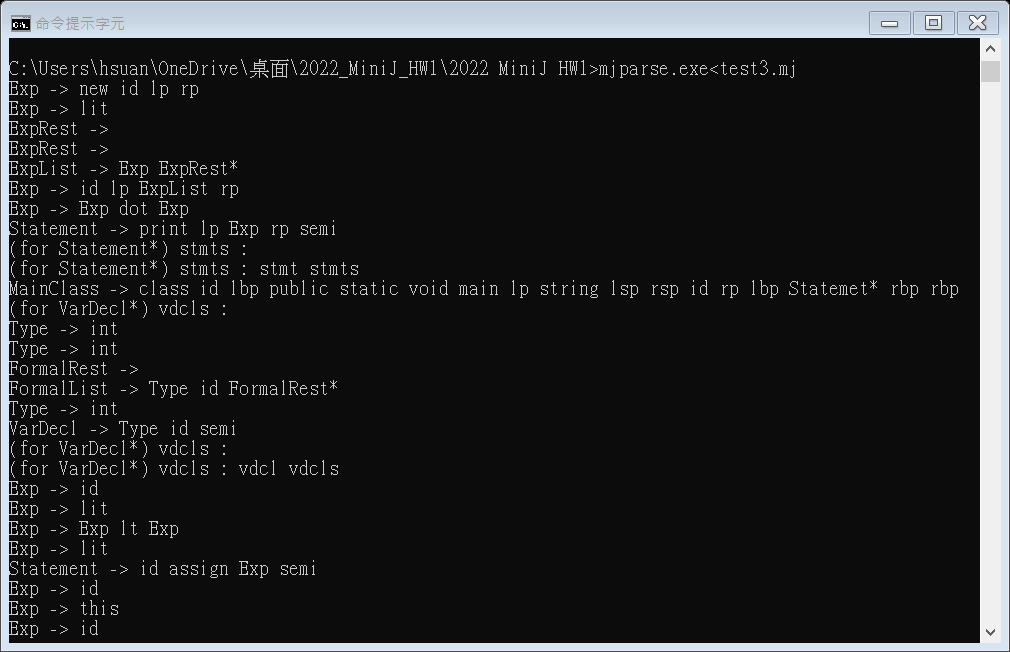
1. Test run results

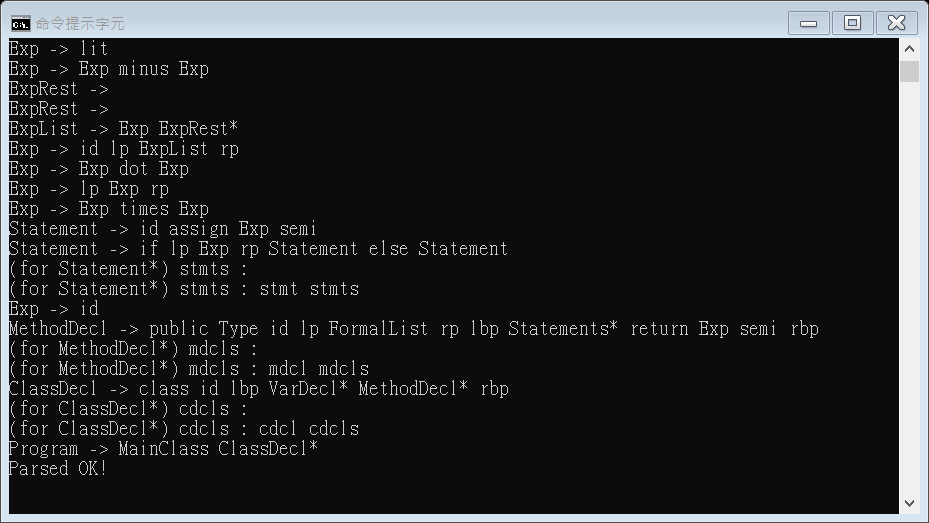


* + - * 圖七、test1.mj執行結果



* + - * 圖八、test2.mj執行結果





* + - * 圖九、test3.mj執行結果

1. Discussion

這部分我們的參考資料有：

1. Bison Declaration Summary([Bison - Bison Grammar Files (mit.edu)](http://web.mit.edu/gnu/doc/html/bison_6.html#SEC54)
2. postgresql核心語法解析器詳解的除錯衝突部分(<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/617577/>)

　　在將.y檔debug完之後，發現在編譯時出現了一個問題：

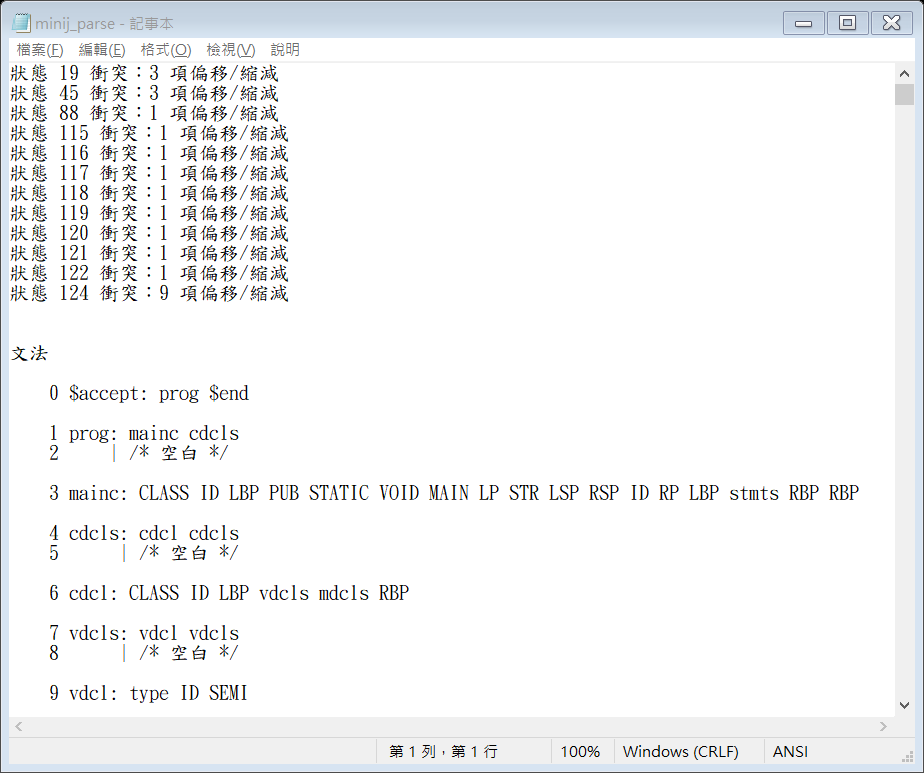


* + - * + 圖十、bison編譯問題

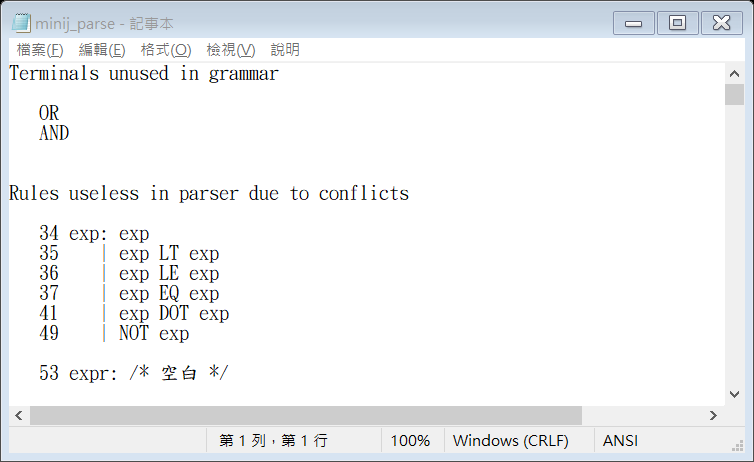
在詢問老師要怎麼解決這個問題後得知可將%expect 28 改為%expect 24，就可以讓bison完成編譯。

關於%expect的用途，在參考資料1中的說明為「Declare the expected number of shift-reduce conflicts」，意即此指令是告訴Bison解析器應有N個shift/reduce衝突，如果不匹配，Bison將報告編譯時錯誤(如圖十)。

在參考資料2中說明如果想要除錯移進/規約衝突時，可在CMD叫出bison時打上-v可生成一個OUTPUT檔，他會將語法解析過程中的某個具體節點的推導路徑給打印出來(如圖十一)，此外，在遇到其他問題例如語法錯誤需要debug時，透過output檔就能夠很容易地定位到自己規則哪一部分出現異常(如圖十二)。



* + - * 圖十一、minij\_parse.out截圖



* + - * 圖十二、output檔報告之錯誤部分