HW2 資訊系 F74009038 顧凱云

環境：linux (ubuntu)

執行方式：./Hw2

程式解說：

這個作業主要是用資料結構 vector 以及 自己宣告的class Node來輔助完成。

Class Node :

要產生tree時所需的stack要push pop的element，一開始會將start symbol S 以及 $ push進stack，經過一連串的操作後當stack中的element都pop完後即產生derivation tree。

這個class 包含兩個 member data : data(type為string)-用來存push進來的non-terminals or terminals，而num則用來存tree level。

此外這個class包含constructor和許多access這兩個變數的function (get和set)。

Vector:

這個程式用了相當多的vector：

vector<string> nonTerminals : 用來存non-terminals

vector<string> rhs : 用來存每個rule 的 right hand side

vector<string> terminals: 用來存 terminals

vector<string> derRule[500] : 這是一個 array or vector，array size=500，每個array的element都為一個vector，存每個left hand side(terminals)所對應之right hand side

vector<string> nullableSet：用來存含epsilon的non-terminals

vector<string> firstSet[nonTerminals.size()+terminals.size()]:

為一個 array of vector，array中的vector先存terminals的first set再存non-terminals的first set

vector<string> LLtableNontoTer[nonterminals.size()]:為一個array of vector，存放每個non-terminals代換成terminals的過程中所有步驟的non-terminals和terminals

vector<string> rhsToken[rhs.size()]:為一個array of vector，存放每個rule的right hand side所切成token

vector<string> followSet[terminals.size()+nonTerminals.size()]: 存放每個terminals和non-termnials的follow set，這個程式只用到non-terminals的follow set

vector<string> LLtableTer :存放LLtable中的non-terminals

vector<string> LLtableNon:存放LLtable中的terminals

vector<string> LLtableRule:存放LLtable中的derivation rule

vector<string> mainC : 存放讀入main.c的每個token以及一個$，作為stack用

vector<Node> stackList : 這個vector存放的element為Node這個class的obejct，作stack用

程式流程：

所有程式的判斷non-terminals，terminals以及其他部分的方式為一次讀一行string進來處理，用兩個變數head 和 tail，一開始都指到string的一開始，接個一個個字元來判斷，若不為’ ‘則tail++直到遇到’ ‘，在用substring(head,tail)的方式就可以取出token。

一開始先讀grammar.txt，依照前面有沒有Tab區分出terminals和non-terminals存入vector，再來開始找first set和follow set 並且輸出到set.txt。

在找first set時會把每次找到non-terminals的代換都存到LLtableNontoTer中，在建構LLtable時會依照non-terminals以及對應的terminals由LLtableNontoTer找到是哪個rule所代換的，進而完成LLtable的輸出。

最後建tree時，用mainC以及stackList來完成，由stackList的top和mainC的top比較若一樣的話就pop mainC以及stackList，如果不同則pop stackList的top並LLtable將對應的rule push到stackList中，直到最後為空停止，並產生tree。

mainC的部分是用作業一的lexical analyzer修改來完成，主要功能和作業一差不多，只是把結果丟到mainC中，遇到 id 和 num時，會把 id 和對應的變數，num以及對應的數字都丟到mainC中，其他部分就按照原本讀main.c所獨到的token都到mainC中。

Note : 詳細程式碼處理細節解說寫在程式碼的注解中