10667213 鄧偉祥

以下結果皆在 Ubuntu16.04 中測試。

1. 編譯方法：
   使用 makefile。
   編譯成功會產生叫做 mini_pascal 的執行檔。
2. 執行方法：
   ./mini_pascal test.p
   會產生 test.j 檔，接著在含有 jasmin.jar 的資料夾下輸入：
   java –jar jasmin.jar test.j
   則會產生相對應的.class 檔，檔名是 test.p 裡 program 後面接的名字(假設是 test)。
   java test
   則會執行 test.p 的內容。
3. Parsing 失敗會顯示：
   syntax error at line N
   或是有 semantic error 都不會產生.j 檔。
   如果沒有錯誤訊息即代表編譯成功。
4. 資料夾：
   (1) source 放置所有的源代碼，包含 makefile、mini_pascal.y、mini_pascal.l 以及其他產生 symbol table、做 semantic check、做 code generation 的相應源代碼(codegene.c、codegene.h、node.c、node.h、symtab.c、symtab.h)。
   (2) bin 放置我事前編好的 compiler，檔名為 mini_pascal。
   (3) doc 放置本文件。
   (4) my_test 放置老師給的測試檔、以及我寫的測試檔(.p 檔)，以及我事先用 bin 資料夾內的 mini_pascal 產生的.j 和.class 檔。
5. 完成事項：
   我總共完成 1、2、3、4、5、6、7、8、9、10、11、12、13、14、16、17、18、19、20、21、22、23、24、44、45、46 項目。
   我的 compiler 無法 run time 做 array bound 的檢查(第 15 項)，其他應該都有完成(我有檢查過，但不知道會不會有疏漏的部分)，另外有加入&&和||的功能(第 44 項)，但此項目只支援最多兩個表達式(例如 exp && exp && exp 不支援)；以及完成 for-loops(第 45 項)，但 for-loop 的 index 只支援純數字，不支援變數；最後多完成 repeat-until loops(第 46 項)。
   1~19、23、24 項的檢查可參考我寫的 test_all.p 檔，或老師提供的所有測試檔。
   20~22 項的檢查可參考老師的測試檔：fibonacci_recursive.p、

test_function.p、test_procedure.p、test_recursion.p 等檔案。
44 項的檢查請參考我寫的 test_andor.p 檔。
45 項的檢查請參考我寫的 test_for.p 檔。
46 項的檢查請參考我寫的 test_repeat.p 檔。
為了實現輸出輸入的部分，我寫了 7 個函式：printlnInt、printlnReal、printlnString、printInt、printReal、printString、readInt。
前三個函式與後三個函式的差別在於輸出會自動換行，最後一個函式是用來輸入整數用的，使用方式例如：a := readInt;
最後我寫了幾個有趣的小程式來測試我的 compiler：

(1) qsort.p
    原本老師就有給的測試檔，可輸入最多 50 個整數(輸入 0 代表停止輸入)，接著輸出 quick sort 產生的排序數列。
(2) exponential.p
    利用泰勒展開式來計算 e^x (近似值)。
(3) gcd.p
    使用 Euclidean 演算法計算輸入的兩個整數的最大公因數。
(4) power.p
    計算輸入兩個整數(x, y)的指數：x^y。
(5) knight_tour.p
    使用 backtracking 演算法解騎士巡邏問題，起始點只能是(0, 0)或(7, 7)，否則要算很久…(沒有算出來過)。
(6) knight_tour_Warnsdorff.p
    使用 Warnsdorff 演算法解騎士巡邏問題，起始點可以任意給定。