112-2 國立成功大學編譯系統課程

NCKU Compiler Construction - 2024 Spring

Homework 1. Lexical Analysis (Scanner)

Deadline: 2024/04/05 23:59:59

Late submissions are not accepted

作業規範

- 請不要攻擊我們的伺服器，否則這堂課會直接當掉
- 作業不接受任何形式的補交或遲交，請大家在截止前完成
- 作業截止後，我們會將所有同學的程式碼送到我們的比對系統中，如有發生作業抄襲等情形，抄的人與被抄的人學期作業成績 (40 %) 以 0 分計算

本學期的作業配分

- 這學期總共有 3 份作業，作業成績佔學期成績 40%
  - Homework 1: 10%
  - Homework 2: 15%
  - Homework 3: 15%

作業目標

- 實作出一個簡易的 C++ Compiler
- 三份作業的內容合起來就是一個可以動的 Compiler
- 第二份與第三份作業都會需要使用到前一份作業的內容，因此如果其中一份作業沒有完成會完成不了後面的!

簡易的 C++ (?

- 沒有 header file
- 沒有 OOP (物件導向程式設計)
- 沒有 namespace
- 只保留了最基本的語法元素
- 也許，你可以把它叫做輕量級的 C++

Homework 1: Lexical Analysis (Scanner)

- 這是實作 Compiler 的第一個步驟!
- 我們要先把程式碼切成一小片一小片的東西 (Token)
    - int number = 2024;
        - int
        - number
        - =
        - 2024
        - ;

Homework 1: Lexical Analysis (Scanner)

- 切下來的 Token 會被分為兩種類型
  - 有用的 Token (就是對程式來說有功能的東西)
  - 沒用的 Token (需忽略掉，像是註解裡面的東西)
  - 像是下面我們切出 Hello world! 的時候我們必須要知道他們是註解內的東西，需要忽略

```
int number = 10; // Hello world !
```

Homework 1: Lexical Analysis (Scanner)

- 有用的 Token 全部被切下來之後，下一步就會丟給 Parser
- Parser 要做的事情就是幫你看語法是否正確
- 但這個就是下一個步驟了 (Homework 2)
- 這一個作業我們只需要先把所有 Token 切下來就好!
- 接下來會列出輕量版 C++ 的所有 Token

Homework 1: Lexical Analysis (Scanner)

- 你的 Scanner 必須要依序輸出你切下來的 Token
- 並在最後輸出程式碼共有幾行
- 輸出的格式我們已經寫好一個 Function (printToken)
- printToken([參數 1],[參數 2])
- 參數 1 放的是你切下來的 Token 類型
- 參數 2 是這個 Token 的內容 (如果沒有內容則放 NULL)

Homework 1: Lexical Analysis (Scanner)

```
1 int main(string argv[]) {
2     cout << "Hello World" << endl;
3     return 0;
4 }
```

```
 1 INT_T
 2 VARIABLE      main
 3 '('
 4 STR_T
 5 VARIABLE      argv
 6 '['
 7 ']'
 8 ')'
 9 '{'
10 COUT
11 SHL
12 STRING_LIT    "Hello World"
13 SHL
14 VARIABLE      endl
15 ';'
16 RETURN
17 INT_LIT       0
18 ';'
19 '}'
20
21 Total line: 4
```

Homework 1: Lexical Analysis (Scanner)

- 註解的註解內容視為同一個 Token

```
1  int main(string argv[]) {
2      cout << 'H' << 'e' << 'l' << 'l' << 'o' << ' ' << 'w' << 'o' << 'r' << 'l' << 'd' << '!' << endl;
3      // this is a single comment !
4      return 0;
5  }
```

```
COMMENT       // this is a single comment !
RETURN
INT_LIT       0
';'
'}'

Total line: 5
```

Homework 1: Lexical Analysis (Scanner)

- 實作的部分我們使用 C 語言
- 實作涉及到稍微複雜的字串處理，所以我們準備了好東西
- Flex (一個 Scanner 的工具)
- 我們助教也把模板幫忙寫好了，大家剩下的任務只要把 Token 填入就完成了！

Homework 1: Lexical Analysis (Scanner)

- Flex 安裝：sudo apt install flex (系統應該幫你裝好了)
- 只要在 Rules section (我們有幫忙註解標示) 的區塊寫下你要切的字串，Flex 就會在掃到這個 Token 的時候幫你切下來
- 語法：[Token] { 你要做的事情 }
    - 以這個作業來說，你掃到 Token 要做的事情就是輸出

```
"void"      { printToken("VOID_T", NULL); }
"char"      { printToken("CHAR_T", NULL); }
"int"       { printToken("INT_T", NULL); }
```

Homework 1: Lexical Analysis (Scanner)

- 相信有人發現其實你不用別人寫好的工具也可以自己手刻
- 如果不想用 Flex 的你也可以自己手刻一個
- 但用 Flex 肯定寫起來輕鬆不少就是了 ><
- 題外話：如果你想要手刻一個而且希望他切 Token 的速度很快，你可能會需要一個資料結構來輔助你: Trie

Homework 1 作業上傳

- 請先 ssh 到 140.116.154.66 (非成大網路者，請使用 VPN)
- 接著在你自己的資料夾中，clone 下方的 GitHub repo
- https://github.com/ColtenOuO/2024-Spring-NCKU-CompilerHW1/tree/main
- 進到 repo 資料夾後開始編寫 scanner.l
- 如果你是想要自己手刻的，就自己改 makefile

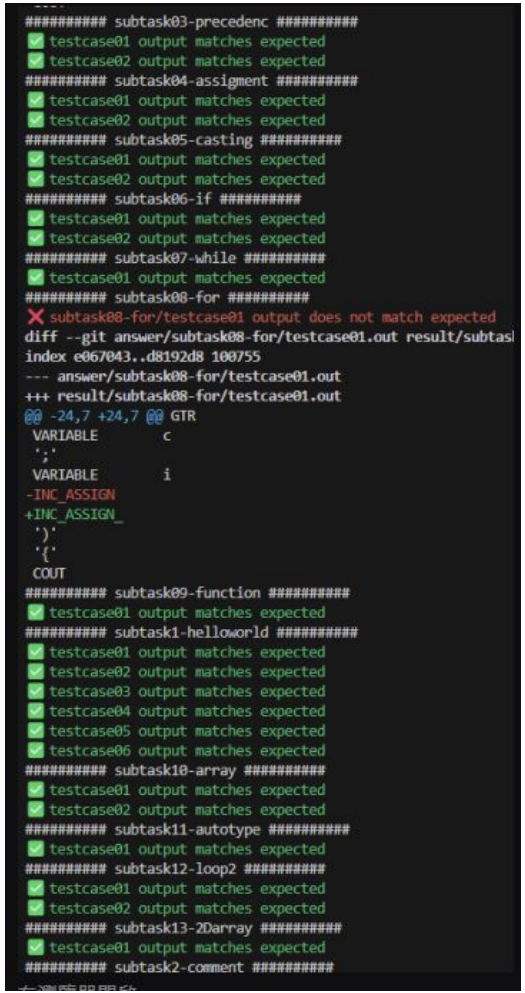
Homework 1 作業上傳

- Username 是學號，密碼也是學號，登入後請改密碼
- ~~也請不要亂登入其他同學的帳號去改他密碼~~
- 需要辨識的 Token 表已經放在 repo 的 README 中

Homework 1: 評分

- 我們將測試資料分成非常多個子題，每一個子題有會有一些測資，該子題必須要通過所有測資才會拿到該子題的分數
- 我們有寫好一個 bash 腳本可以直接協助測試你的程式
- 本次作業滿分為 120 分

Homework 1: 評分



Homework 1: Subtasks

- subtask 01: helloworld (8 %)
- subtask 02: 註解 (8 %)
- subtask 03: 運算優先順序 (8 %)
- subtask 04: 運算符號 (8 %)
- subtask 05: 強制轉型 (8 %)
- subtask 06: 條件判斷 (8 %)
- subtask 07: while 迴圈 (8 %)

Homework 1: Subtasks

- subtask 09: 函式 (8 %)
- subtask 10: 一維陣列 (8 %)
- subtask 11: auto (8 %)
- subtask 12: 多層迴圈 + if (10 %)
- subtask 13: 二維陣列 (8 %)
- subtask 14: 綜合版測試資料 (14 %)

Homework 1: 作業提示 1

- Flex 支援正則表達式，只要一開始先定義好 (類似變數宣告)
  接下來在切 Token 的時候就可以直接使用
- 某些 Token 你會需要好好善用這一個東西
- 定義方式：[名稱] [正則表達式]
- 使用時：將名稱用大括號包起來即可使用
- Example:
  - 定義：digit [0-9]
  - 使用：{digit}

Homework 1: 作業提示 2

- Flex 有以下兩種規則
- 最長匹配：不用擔心 <= 會被拆解成 < 跟 =，因為 Flex 會盡可能的找到 Token 長度最長的去做匹配
- 先定義先匹配: 如果有一段字串同時被 2 個 Token 規則匹配，而且這兩個 Token 長度一樣，那麼 Flex 會選先被定義的那一個 Token

Homework 1: 作業提示 3

- 這份作業最困難的地方是多行註解的處理
- 善用 yymore()，他的功能是把當前讀到的東西先丟到緩衝區去，之後要拿出來時會一次全部丟給 yytext
- 在註解的部分善用 yymore() 會輕鬆很多!

Homework 1: 作業提示 4

- Flex 有 state 的功能
- 你可以設立一個 state，的起始條件跟終止條件，這些條件可以放入指定的 Token，協助你判斷更多東西
- 像是多行註解你就可以把起始 Token 設定成 /* 終止 Token 設定成 */
- 最後再搭配 yymore() 就好寫很多了

Homework 1: 作業提示 5

- 簡單來說可以想像，如果遇到起始 Token，就會開始進入那一個 state 裡面做事情
- 使用 state 之前，記得先將你的 state 做宣告 (%x state名稱)

```
起始Token                          { BEGIN(state 名稱); }
<state 名稱>終止 Token               { BEGIN(INITIAL); }
<state 名稱>state 內判斷的 Token { 要做的事情 }
<state 名稱>state 內判斷的 Token { 要做的事情 }
<state 名稱>state 內判斷的 Token { 要做的事情 }
```

Homework 1: 作業提示 6

- Flex 語法資源：
    - 找 Stackoverflow
    - https://westes.github.io/flex/manual/

Homework 1: Final

- 最後祝大家作業順利! Good luck & have fun
- 測資是我們熬夜肝好幾個晚上用出來的，可能難免失手，如果有覺得不對勁的地方可以跟我們說 ><
- f74114744@gs.ncku.edu.tw (資訊 115 陳俊安)
- f74114760@gs.ncku.edu.tw (資訊 115 張羿軒)