A file describing what changes I have to make to my scanner

xxx.l

四資工三甲　B10615031　許晉捷

---

1. 新增 char 的 token：

project 1 的規格書中並沒有規定要有 char，但是 project 2 的規格書中卻有 char 的型態；因此，在 lex 中新增之。

```
284     /* TOKEN: characters */
285 (\'[^\n\t]\')|(\'\'\'\')|(\'\"\"\') {
286     // buffer the text
287     bufText(yytext);
288 
289     // special case 1: '''', which means a single-quote sign '
290     if (!strcmp("\'\'\'\'", yytext)) {
291         tokenChar('\'');
292     }
293     // special case 2: '""', which means a double-quote sign "
294     else if (!strcmp("\'\"\"\'", yytext)) {
295         tokenChar('\"');
296     }
297     // general case, e.g., 'e', '4', '+', ...
298     else {
299         char ch = yytext[1];
300         tokenChar(ch);
301     }
302 }
```

有一個地方必須提：由於沒有任何規定若該字元為**單引號**或**雙引號**時是否有特殊的表示法。因此，我在這邊自己定義：如同字串的規格中，**連續兩個雙引號**代表**一個雙引號**般；在 char 中，**單個或連續兩個**單引號皆代表**一個**單引號、**單個或連續兩個**雙引號皆代表**一個**雙引號。即：

(1) '''

(2) ''''

(3) '"'

(4) '""'

(1)及(2)都是合法的**單引號 char**；(3)及(4)都是合法的**雙引號 char**。

2. 進一步拆分 tokenString：

在 porject 1 中，tokenString 可以用於辨認 ID、字串、實數等 tokens；但在 project 2 中，辨認 ID 有它自己的 tokenId、辨認字串有它自己的 tokenString、辨認實數有它自己的 tokenReal。另外，還加入了 tokenChar 用於辨認字元。

3. 在所有 tokenXXX 的動作中皆加入 **return 敘述**：

所有的 tokenXXX：tokenChar、tokenString、tokenInteger、tokenReal、tokenId，以及 token（用於辨認除了以上五者以外的 tokens）中，最後都要 return，如圖。

```
// get a literal character token
#define tokenChar(ch) { \
    yylval.c = ch; \
    return LITERAL_CHAR; \
}

// get a literal string token
#define tokenString(str) { \
    strcpy(yylval.s, str); \
    yylval.s[strlen(str)] = 0; \
    return LITERAL_STRING; \
}

// get an integer token
#define tokenInteger(str_i) { \
    sscanf(yytext, "%d", &(yylval.i)); \
    return LITERAL_INTEGER; \
}

// get a real number token
#define tokenReal(str_r) { \
    sscanf(yytext, "%lf", &(yylval.f)); \
    return LITERAL_FLOAT; \
}

// get an identifier token
#define tokenId(id) { \
    strcpy(yylval.s, yytext); \
    yylval.s[strlen(yytext)] = 0; \
    return ID; \
}

// get a token of rest cases
#define token(t) return t
```

以上所有紅框內的 symbols（如 **LITERAL_CHAR**、**ID** 等），皆於 xxx.y（我的 yacc 檔）中定義，並放在 y.tab.h 中。

4. include 標頭檔之 **y.tab.h**：

```
12    #include "y.tab.h"
```

5. 在 lex 檔中所有的 token 辨認敘述中，呼叫 token 時所傳入的參數，全部改成**存放在 y.tab.h 中所定義的 symbols**：

例如 AND、OR，及 NOT：

```
155     /* TOKEN: "&&" (logical AND) -> logical operators */
156 "&&" { bufText("&&"); token(AND); }
157
158     /* TOKEN: "||" (logical OR) -> logical operators */
159 "||" { bufText("||"); token(OR); }
160
161     /* TOKEN: "!" (logical NOT) -> logical operators */
162 "!" { bufText("!"); token(NOT); }
```