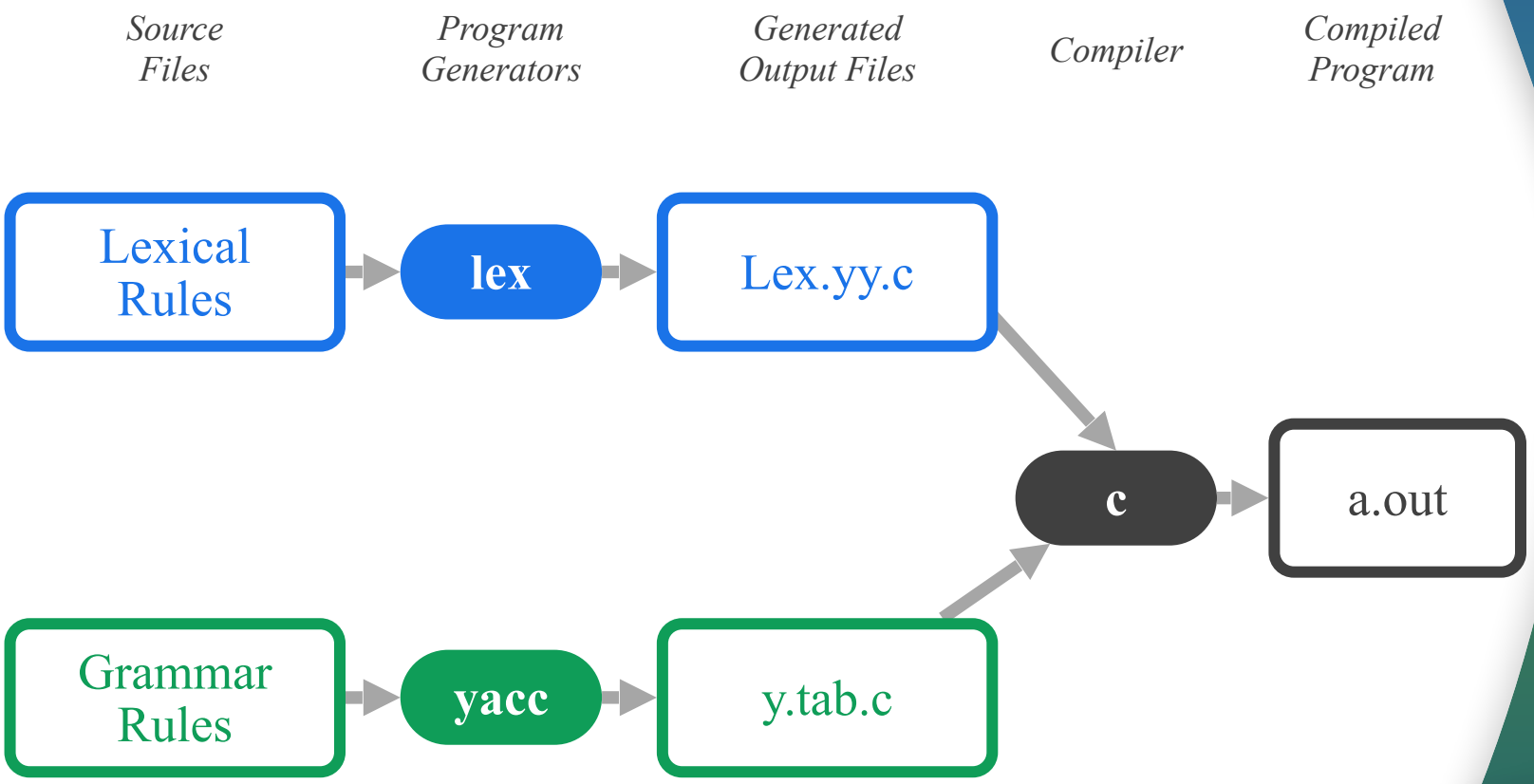
編譯器製作

Lex Tutorial

助教| 丁襄龍 Clove Dragon

Compiler 的分工



Lex 的工作

運作

Lex會把 input 當作 a sequence of characters

- 一個以上連續的character會形成一個token

目的

Lex的目的是檢查 token 是否合法

- 例如不合法的變數名稱 (identifier)

條件

Lex必須事先定義規則

- Regular Expression
  - 可以被辨識的token

Lex 的 Input

以 Java 為例

```
public static void main() {
    int c;
    int a = 5;
    int 5a;      //不合法的identifier

    c = add(a, 10);
    if (c > 10)
        print("c = " + -c);
    else
        print(c);
    print("Hello World");
}
```

Lex 的格式

分成三部分，每個部分以 %% 區隔開來

Definition

%%

Lex Rules

%%

User code

Definition


```
%{
#include <stdio.h>
unsigned charCount=1, idCount=0, lineCount=1;
%}
operator [\+\-\*\/]
space [ \t]
eol \n

/* You should write your own regular expression. */
reserved_word [ ….]
symbol [ ….]
id [ ….]

%%
```

Rules


```
%%

{operator} {
   printf("Line: %d, 1st char: %d, \"%s\" is an
                     \"operator\".\n", lineCount, charCount, yytext);
   charCount += yyleng;
}
{space} {
   charCount++;
}
{eol} {
   lineCount++;
   charCount = 1;
}
{reserved_word} {
   /* You should write your own code */
}

%%
```

Rules

- Scanner 所匹配規則的優先順序
  - 會 scan 出長度最長的 token 去進行匹配
  - 如果匹配長度一樣，
    則看被定義的先後順序 (由上到下)

User Code


```
%%
int main(){
    yylex();
    return 0;
}
.......
......
```

來看看
Test Cases

Test File

test1.java

```
public class Test1 {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

Output

```
bot@Pc-Nsysu-Lab-Curtis ~/compiler/lab1/LexDemo
$ ./demo < test1.java
Line: 1, 1st char: 1,  "public" is a "ReservedWord".
Line: 1, 1st char: 8,  "class" is a "ReservedWord".
Line: 1, 1st char: 14, "Test1" is an "ID".
Line: 1, 1st char: 20, "{" is a "symbol".
Line: 2, 1st char: 5,  "public" is a "ReservedWord".
Line: 2, 1st char: 12, "static" is a "ReservedWord".
Line: 2, 1st char: 19, "int" is a "ReservedWord".
Line: 2, 1st char: 23, "add" is an "ID".
Line: 2, 1st char: 26, "(" is a "symbol".
Line: 2, 1st char: 27, "int" is a "ReservedWord".
Line: 2, 1st char: 31, "a" is an "ID".
Line: 2, 1st char: 32, "," is a "symbol".
Line: 2, 1st char: 34, "int" is a "ReservedWord".
Line: 2, 1st char: 38, "b" is an "ID".
Line: 2, 1st char: 39, ")" is a "symbol".
Line: 2, 1st char: 41, "{" is a "symbol".
Line: 3, 1st char: 9,  "return" is a "ReservedWord".
Line: 3, 1st char: 16, "a" is an "ID".
Line: 3, 1st char: 18, "+" is an "operator".
Line: 3, 1st char: 20, "b" is an "ID".
Line: 3, 1st char: 21, ";" is a "symbol".
Line: 4, 1st char: 5,  "}" is a "symbol".
Line: 5, 1st char: 1,  "}" is a "symbol".
The symbol table contains:
Test1
add
a
b
```

Test File & Output

test1.java

Test File

```
public class Test1 {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

Output

```
bot@Pc-Nsysu-Lab-Curtis ~/compiler/lab1/LexDemo
$ ./demo < test1.java

Line: 1, 1st char: 1,  "public" is a "ReservedWord".
Line: 1, 1st char: 8,  "class" is a "ReservedWord".
Line: 1, 1st char: 14, "Test1" is an "ID".
Line: 1, 1st char: 20, "{" is a "symbol".
```

Lex 的特殊字元

- 這些字元在regular expression中有特殊意義，
  如果要當成一般字元，
  請在前面加上\這一個跳脫字元(Escape character)
  - ? * + | ( ) ^ $. [ ] { } " \

- Digit [0-9]
- Letter [a-zA-Z]
- Operator [\+\-*]

使用 Lex

如何使用 Lex File

- 我們的目的要將 demo.l 編譯成可以執行的 scanner
- 首先必須安裝flex這個程式來編譯我們的lex file

```
sudo apt-get install flex        <- 以ubuntu為例
```

- 透過 flex 將 demo.l 編譯成 C source file，
  這個 C source file 就是我們的 scanner

```
flex  demo.l
```

- C source file預設檔名為 lex.yy.c ，
  最後我們可以利用gcc將其編譯成可執行檔

```
gcc lex.yy.c –o demo -lfl
```

- 執行檔為demo，假設我們要scan的檔案為test1.java

```
./demo < test1.java
```

- 也可以直接執行demo，<Ctrl+D>可以送出EOF

Regular Expression

常用字元符號

.	任意 字元 (不含換行)
\d	任意　數字
\D	任意非 數字
\w	任意　文字、數字、底線
\W	任意非 文字、數字、底線
\s	任意　空白字元 (空白、定位、換行)
\S	任意非 空白字元 (空白、定位、換行)
\n	換行字元 (NewLine)
\t	定位字元 (Tab)
\r	回車字元 (Carriage return)
\0	Null 字元

特殊字元符號

\.	. 字元	\(	(字元
\?	? 字元	\)	) 字元
*	* 字元	\[	[字元
\+	+ 字元	\]	] 字元
|	| 字元	\{	{ 字元
\^	^ 字元	\}	} 字元
\$	$ 字元	\\	\ 字元
\”	“ 字元	\/	/ 字元

常用列舉規則

規則	說明	合法舉例
[12abc]	A single character of 1, 2, a, b, c	1, 2, a, b, c
[^12abc]	A single character except 1, 2, a, b, c	3, 4, d, e, f
[0-9A-Z]	A character in the range of 0-9 or A-Z	0, 1, A, B, C
[ab]|[0-9]	A single character of a, b, or in the range of 0-9	a, b, 0, 1, 2

常用次數符號 (加在尾巴)

符號	說明
*	重複 0 ~ ∞ 次
+	重複 1 ~ ∞ 次
?	重複 0 ~ 1 次
{n}	重複 恰好 n 次
{n,}	重複 n ~ ∞ 次
{n,m}	重複 n ~ m 次

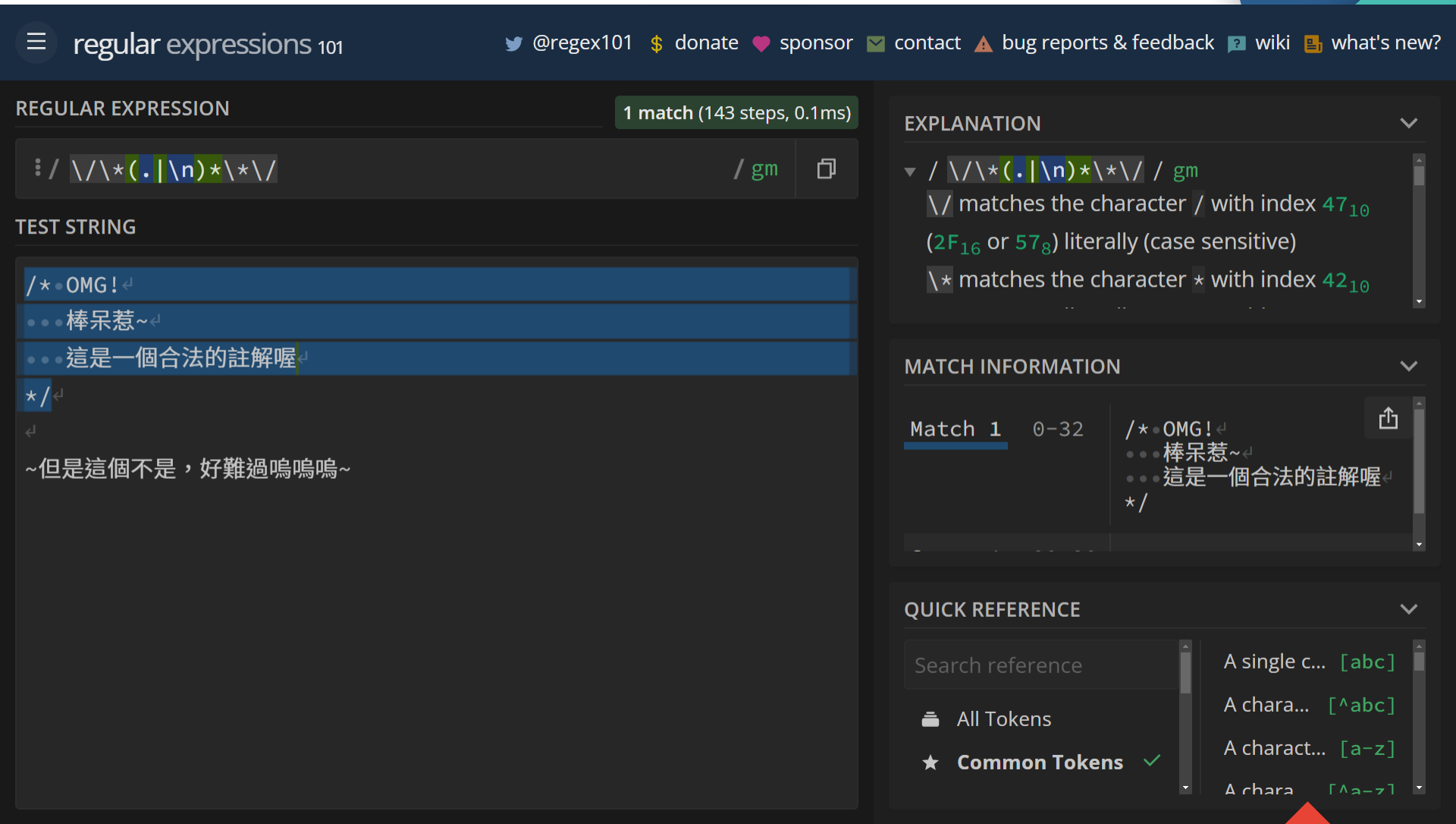
Regular Expression

- 當然不止這一些

- 如果你對正規語言不熟悉，
  網路上對有相當豐富的資源，利如
  - https://www.vixual.net/blog/archives/211

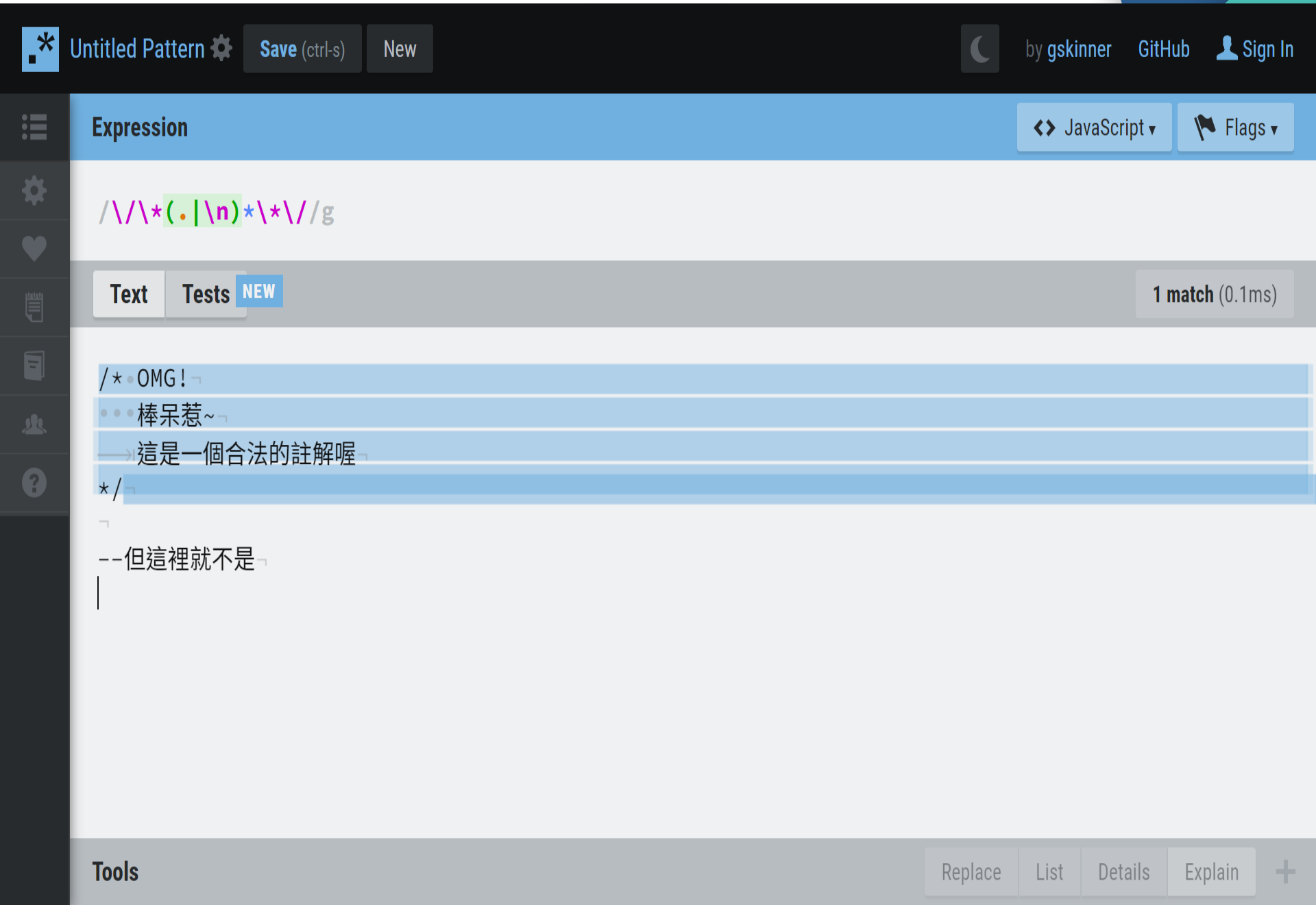
- Online Regular Expression Tester
  - https://regex101.com
  - https://regexr.com

regexr.com



這裡有相當豐富的
Regular Expression Rules 可以參考喔

regexr.com


Untitled Pattern
Save (ctrl-s)
New
by gskinner
GitHub
Sign In
Expression
JavaScript
Flags
/\/*(.|\n)**\//g
Text
Tests NEW
1 match (0.1ms)
/* OMG!
棒呆惹~
這是一個合法的註解喔
*/
--但這裡就不是
Tools
Replace
List
Details
Explain

關於作業I

推薦的環境

- 在虛擬機上裝 Ubuntu
  - Ubuntu 22

作業繳交注意事項

- **DUE DATE: 2025年04月13日 23:59**
- 程式 Demo 環境是 Ubuntu 22.04.2 LTS，
  請保證程式碼能夠編譯執行
- 請參考課程網頁中的測試檔案來驗證你的程式
- 助教會自行設計額外的測試檔案，因此請保證你所寫的 Regular Expression 可以 match 到大部分的 cases
  - 例如一些複雜的變數名稱、浮點數必須要可以是負數...
- 請準時繳交作業，作業遲交一天打七折
- 請把作業包成一個壓縮包，上傳至網路大學，
  檔名命為「學號_**hw1.zip**」。
  學號輸錯，作業分數-10；沒輸學號，作業分數-50。
- 在繳交截止後，會安排時間 Demo **(4/16~4/18)**，
  請準時到 EC5023 資料庫系統實驗室 找助教 Demo。

常見問答與提示

題目檔範例輸出

```
// print hello world
{
  print("hello world");
  int a = 5 + 5.5;
}
```

你的scanner 將會輸出下列的結果：

Line: **1**, 1st char: **1**, "**// print hello world"** is a "**comment"**.
Line: **2**, 1st char: **1**, **"{"** is a "**symbol"**.
Line: **3**, 1st char: **3**, **"print"** is a "**reserved word"**.
Line: **3**, 1st char: **8**, **"("** is a **"symbol"**.
Line: **3**, 1st char: **10**, **"hello world"** is a "**string"**.
Line: **3**, 1st char: **22**, **")"** is a **"symbol"**.
Line: **3**, 1st char: **23**, **";"** is a "**symbol"**.
Line: **4**, 1st char: **3**, **"int"** is a "**reserved word"**.
Line: **4**, 1st char: **7**, **"a"** is an "**ID"**.
Line: **4**, 1st char: **9**, **"="** is a "**operator"**.
Line: **4**, 1st char: **11**, **"5"** is an "**integer"**.
Line: **4**, 1st char: **13**, **"+"** is an "**operator"**.
Line: **4**, 1st char: **15**, **"5.5"** is a "**float"**.
Line: **4**, 1st char: **18**, **";"** is a "**symbol"**.
Line: **5**, 1st char: **1**, **"}"** is a "**symbol"**.
The symbol table contains:
a

Error Handling

- 哎呀~ 遇到錯誤該做什麼！？
- 只要可以讓程式遇到錯誤時，
  還能繼續執行下去，不致中止，就可以囉。
- 而如果你的錯誤處理，
  是包含輸出錯誤相關資訊額外加分

可以被識別的資料型態

- 合法的 Java 資料型態，都應該要被識別
  - 畢竟，這是一個 Java 的 Scanner
  - 可以自己到 Online Java Compiler 試試看
    https://www.jdoodle.com/online-java-compiler
  - 反之，Java 不合法的宣型態，則都該被視為錯誤
    (如 字串錯誤、數字溢位、數字開頭的變數名稱、…)

- 以 整數 為例
  - 0x14　　(Hexadecimal)
  - 024　　(Octal)

可以被識別的資料型態

- 以 浮點數 為例
- Java 分為 float 和 double 兩種浮點數型態

double	.2 , 2. , 2.0
float	.2f, 2.f, 2.0f

  - 本次作業允許 float 和 double 都納入 {float}

Return Value of User Code

- 既然是放在 “User Code” 裡面的 function，
  顧名思義，就是你想要有一些額外的功能，
  在利用 Lex Rules 掃描檔案的時候，
  呼叫他來達成某些特定目的。

- 所以在本次作業，你的程式需要 Return 什麼值，
  就 Return 他吧。
  - Yacc Paser 會需要有特定的回傳值，
    這部分下次 Lab 作業會再講解

這個文法錯了，要報錯嗎

- 請記得你現在實作的是
  檢查 Lexical Rules 的 Yacc Scanner，
  並不是檢查 Grammar Rules 的 Yacc Parser。

- 所以，如果他是個合法的 Token，就請印出他，
  不需要報錯。

- 例如以下 Java 程式雖文法有誤，無法被編譯，
  但因語彙無誤，因此仍可被正常辨識輸出。

```
int x = 2 + 0.3f ;
```

Symbol Table

- 請不要重複存入 Symbol ID !!
  - Symbol Table 只要存入相異的 ID 即可，
    這樣才符合 Symbol Table 的目的與精神喔！
  - 請善用 lookup function !!
- 按照讀入順序，依序輸出即可
  - 要使用其他順序排列也可以，
    但請在PDF報告書裡說明

這個邏輯要被識別嗎

- 作業說明裡面列舉的定義，
  都是 Minimum Requirements。

- 因此，如果你有考慮到
  比作業說明還多的邏輯與規則，就放膽去做吧！

- 也要記得在報告 PDF 檔裡面，以及 Demo 的時候讓
  我知道，才能給你更高的分數喔！

- (畢竟一個程式語言的規則與邏輯有千千萬萬種，
  如果全部都要實作的話，
  肯定不是幾百行程式碼就能簡單搞定的事XD)

正負號還是加減號

- 這是本次作業實作重點之一，
  所以只能給你一點小小的 Hint 喔 ：
- 如果 +/- 左側不是數字的話，
  那他就代表正負記號，對嗎？
- 如果 +/- 左側是數字的話，
  那他就代表運算子，對嗎？
- 想想看，也許可以善用變數來實作喔
- 這只是提供一個可能的想法，
  還請自己多加思考是否符合所有邏輯

如果有何問題

沒事的，歡迎詢問助教：

丁襄龍
m133040006@gmail.com
EC5023 資料庫系統實驗室

- 請附上你的 學號 + 姓名