YACC-作業二

學號：D1149768

姓名：王冠傑

cal.l

```
%{
#include "y.tab.h"
%}
%option noyywrap
%%
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULTIPLY; }
"/"             { return DIVIDE; }
"("             { return LPAREN; }
")"             { return RPAREN; }
\n              { return '\n'; }
[ \t]         ;  // 忽略空格和換行
.               { printf("Unexpected character: %s\n\n", yytext); }
%%
```

數字匹配

```
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
```

[0-9]+：匹配一個或多個數字。 atoi(yytext)：將匹配到的字串 (數字) 轉換為整數並存入 yylval (供語法分析使用)。 return NUMBER;：回傳 NUMBER 這個 token。

運算符

```
"+" { return PLUS; }
"-" { return MINUS; }
"*" { return MULTIPLY; }
"/" { return DIVIDE; }
```

每個運算符號都直接返回對應的 token，如 PLUS、MINUS 等。

括號

```
"(" { return LPAREN; }
")" { return RPAREN; }
```

(和) 分別返回 LPAREN 和 RPAREN token。

換行

```
\n { return '\n'; }
```

匹配換行符，返回 '\n'。

忽略空白和縮排

```
[ \t] ;
```

匹配空格 ( ) 和 Tab (\t)，但什麼也不做 (動作是空的)。

無法識別的字元

```
. { printf("Unexpected character: %s\n\n", yytext); }
```

匹配任何其他無法識別的字元，並輸出錯誤訊息。

cal.y

程式

```
%{
#include <stdio.h>
#include <stdlib.h>
void yyerror(const char *msg);
extern int yylex();
%}

%token NUMBER
%token PLUS MINUS MULTIPLY DIVIDE LPAREN RPAREN

%left PLUS MINUS
%left MULTIPLY DIVIDE
%nonassoc UMINUS  // 定義負號的優先級

%%
calculation:
    calculation expression '\n' {
        printf("ANS is %d\n\n", $2);
    }
    | /* 空規則，允許輸入空行 */
    ;


expression:
    NUMBER { $$ = $1; }
    | expression PLUS expression {
```

```
        printf("%d plus %d equals %d\n", $1, $3, $1 + $3);
        $$ = $1 + $3;
    }
    | expression MINUS expression {
        printf("%d minus %d equals %d\n", $1, $3, $1 - $3);
        $$ = $1 - $3;
    }
    | expression MULTIPLY expression {
        printf("%d multiply %d equals %d\n", $1, $3, $1 * $3);
        $$ = $1 * $3;
    }
    | expression DIVIDE expression {
        if ($3 == 0) {
            yyerror("Division by zero");
            $$ = 0;
        } else {
            printf("%d divide %d equals %d\n", $1, $3, $1 / $3);
            $$ = $1 / $3;
        }
    }
    | LPAREN expression RPAREN { $$ = $2; }
    | MINUS expression %prec UMINUS {
        $$ = -$2;
    }
    ;
%%
int main() {
    printf("Enter expressions (end with Ctrl+D):\n");
    yyparse();
    return 0;
}

void yyerror(const char *msg) {
    fprintf(stderr, "Error: %s\n", msg);
}
```

Token 定義

```
%token NUMBER
%token PLUS MINUS MULTIPLY DIVIDE LPAREN RPAREN
```

%token: 定義詞法分析器傳遞過來的符號 (tokens)。 NUMBER 表示數字。 PLUS, MINUS, MULTIPLY, DIVIDE 分別表示加減乘除符號。 LPAREN, RPAREN 分別表示左括號 (和右括號)。

運算優先級和結合性

```
%left PLUS MINUS
%left MULTIPLY DIVIDE
%nonassoc UMINUS
```

%left: 表示運算符的左結合性。 PLUS 和 MINUS 具有相同優先級。 MULTIPLY 和 DIVIDE 具有更高的優先級。 %nonassoc: 表示符號是非結合性的，用於定義負號（單目運算符）的優先級。

語法規則

```
calculation:
    calculation expression '\n' {
        printf("ANS is %d\n\n", $2);
    }
    | /* 空規則，允許輸入空行 */
    ;
```

calculation: 主規則，表示程式接受的輸入內容。 calculation expression '\n': 每行可以是新的運算式，以換行符結束，計算結果並輸出。 $2: 表示第二個部分（expression）的值。 空規則: 允許輸入空行。

```
expression:
    NUMBER { $$ = $1; }
    | expression PLUS expression {
        printf("%d plus %d equals %d\n", $1, $3, $1 + $3);
        $$ = $1 + $3;
    }
    | expression MINUS expression {
        printf("%d minus %d equals %d\n", $1, $3, $1 - $3);
        $$ = $1 - $3;
    }
    | expression MULTIPLY expression {
        printf("%d multiply %d equals %d\n", $1, $3, $1 * $3);
        $$ = $1 * $3;
    }
    | expression DIVIDE expression {
        if ($3 == 0) {
            yyerror("Division by zero");
            $$ = 0;
        } else {
            printf("%d divide %d equals %d\n", $1, $3, $1 / $3);
            $$ = $1 / $3;
        }
    }
    | LPAREN expression RPAREN { $$ = $2; }
    | MINUS expression %prec UMINUS {
        $$ = -$2;
    }
    ;
```

expression: 定義運算的規則。 基本數字: 如果是 NUMBER，直接返回值。 $$: 表示這條規則的結果。 $1: 表示第一個部分（NUMBER）的值。 加法: expression PLUS expression。 輸出加法過程，計算並返回結果。 減法、乘法、除法: 與加法類似。 注意除法中檢查是否除以零。 括號: LPAREN expression RPAREN，返回括

號內部運算的結果。 $2: 表示括號內部的 expression。 負號運算: 使用 %prec UMINUS 明確優先級，處理單目負號。 返回負數值。

主程式

```
int main() {
    printf("Enter expressions (end with Ctrl+D):\n");
    yyparse();
    return 0;
}
```

main(): 主函式，提示使用者輸入運算式，並呼叫 yyparse()。 yyparse(): Yacc 生成的主函式，用來解析輸入並執行語法規則。

錯誤處理

```
void yyerror(const char *msg) {
    fprintf(stderr, "Error: %s\n", msg);
}
```

yyerror(): 當解析出現錯誤時，會輸出錯誤訊息，例如「除以零」。

輸入輸出

輸入：

```
12+32
56-2*3
13+6/2
(5+2)*3-5+3
3+5*(8-3)
(4+2)*(6-1)
(2+5)*(4+6)
-5+3
6+-2
(4-9)*3
(2+(3+4))*(5+1)
((8+2)*3)-(6/2)
(5*(3+2))-7
12+32+35+21-20
65+35
32-2
112-(2+32)*3
36/(30+3*2)
12+(12-(12+12))
89+77/(8-1)
```

輸出

```
Enter expressions (end with Ctrl+D):
12 plus 32 equals 44
ANS is 44

2 multiply 3 equals 6
56 minus 6 equals 50
ANS is 50

6 divide 2 equals 3
13 plus 3 equals 16
ANS is 16

5 plus 2 equals 7
7 multiply 3 equals 21
21 minus 5 equals 16
16 plus 3 equals 19
ANS is 19

8 minus 3 equals 5
5 multiply 5 equals 25
3 plus 25 equals 28
ANS is 28

4 plus 2 equals 6
6 minus 1 equals 5
6 multiply 5 equals 30
ANS is 30

2 plus 5 equals 7
4 plus 6 equals 10
7 multiply 10 equals 70
ANS is 70

-5 plus 3 equals -2
ANS is -2

6 plus -2 equals 4
ANS is 4

4 minus 9 equals -5
-5 multiply 3 equals -15
ANS is -15

3 plus 4 equals 7
2 plus 7 equals 9
5 plus 1 equals 6
9 multiply 6 equals 54
ANS is 54

8 plus 2 equals 10
10 multiply 3 equals 30
```

```
6 divide 2 equals 3
30 minus 3 equals 27
ANS is 27

3 plus 2 equals 5
5 multiply 5 equals 25
25 minus 7 equals 18
ANS is 18

12 plus 32 equals 44
44 plus 35 equals 79
79 plus 21 equals 100
100 minus 20 equals 80
ANS is 80

65 plus 35 equals 100
ANS is 100

32 minus 2 equals 30
ANS is 30

2 plus 32 equals 34
34 multiply 3 equals 102
112 minus 102 equals 10
ANS is 10

3 multiply 2 equals 6
30 plus 6 equals 36
36 divide 36 equals 1
ANS is 1

12 plus 12 equals 24
12 minus 24 equals -12
12 plus -12 equals 0
ANS is 0

8 minus 1 equals 7
77 divide 7 equals 11
89 plus 11 equals 100
ANS is 100
```

心得

一開始要搞懂 tokens 和 運算優先級 怎麼配合，還有 yylval 的傳值，真的需要花時間。尤其是 Lex 的正則表達式，一個不小心就可能匹配錯誤，或者忘記處理某些特殊情況，比如空格忽略、非法字符報錯等。 再來是 Yacc 的部分，設定優先級和結合性雖然看起來簡單，但有時候規則稍微複雜一點，debug 起來就超麻煩，因為錯誤訊息不是很直觀。