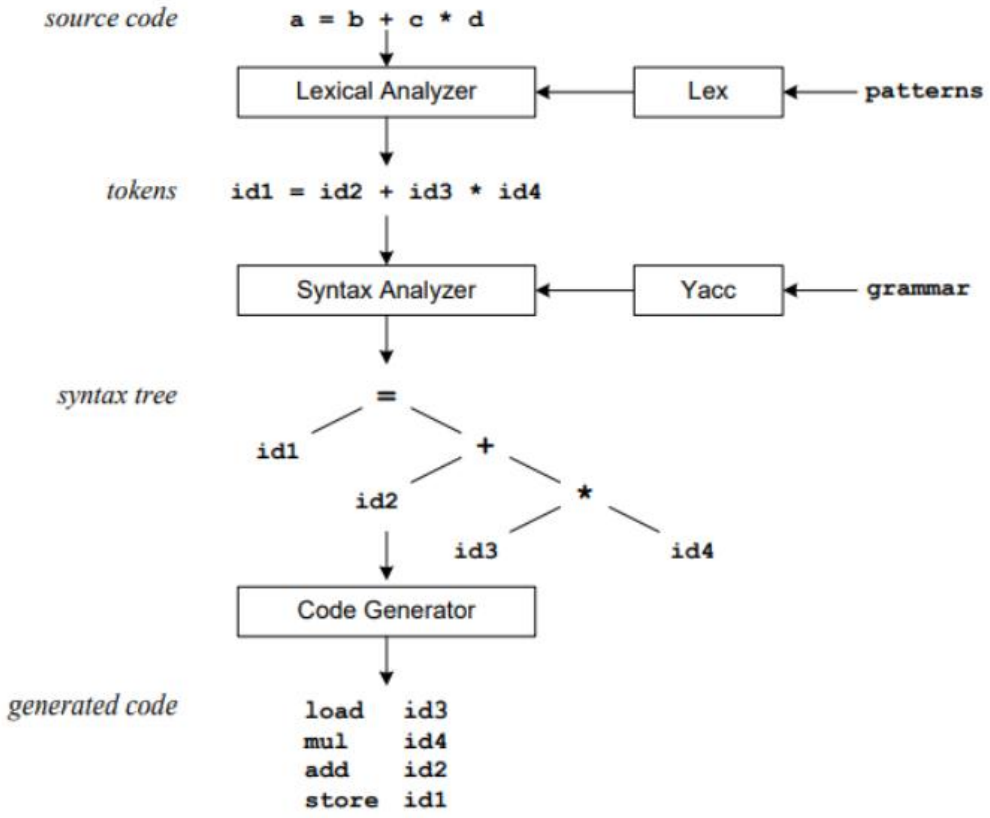
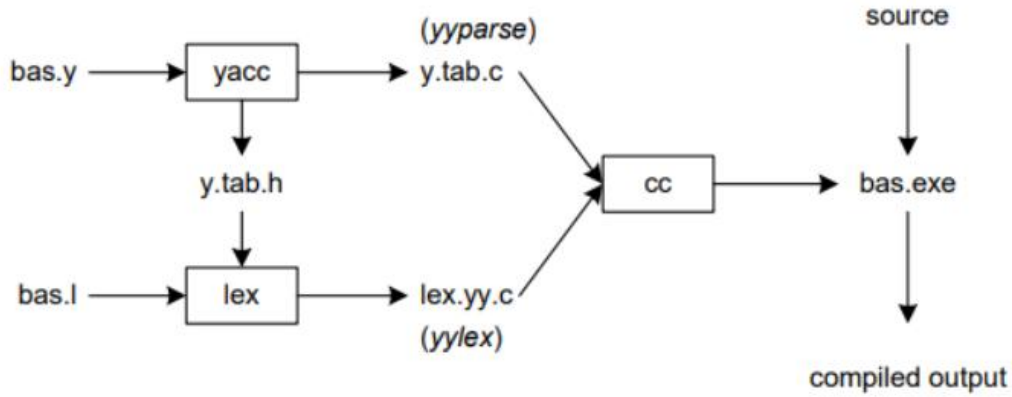
LEX and YACC

姓名：李名智
學號：1102924
製作簡易整數計算機
一、　整體流程、編譯流程　　圖片來源





問題：為何要先編譯 yacc 再編譯 lex?

因 Token 的宣告是在.y 檔內，所以在 yacc 編譯過程中會產生一個 y.tab.h 檔。Lex 會讀取.l 檔裡的樣本（pattern）敘述式及 y.tab.h 內的宣告，產生一個詞彙分析器（Lexical analyzer），並寫在 lex.yy.c 內。另外，yacc 會讀取.y 黨內的語法規則並產生一個語法分析器，並寫在

y. tab. c 中。最後一部則是透過編譯把兩者連結在一起，產生一個執行檔。
編譯步驟如下：

```
bison -d y.y
flex l.l
g++ -c y.tab.c
g++ -c lex.yy.c
g++ -c main.cpp
g++ main.o lex.yy.o y.tab.o -o main
.\main
```

二、 製作 lex（. l 檔）

主要架構：（規則定義 Token 型式）

```
... definitions ...
%%
... rules ...
%%
... subroutines ...
```

宣告、定義：

```
%{
#include "main.h"
#include "y.tab.h"
%}

number      ([0-9]+)
blank_chars ([ \f\r\t\v]+)
expressions ([-+*/()])
```

規則：

```
%%

{number}        { yylval = atoi(yytext); return NUMBER;/*將讀入字串轉成int，並將值存入yylval*/ }
{expressions}   { return yytext[0];/*讀到符號時直接回傳符號*/ }
{blank_chars}   { ;/*若為空白甚麼事都不用做*/ }
"="             { return yytext[0]; }
\n              { ; }

%%
```

子程式：

```
int yywrap(void)
{
    return 1;//若讀取完，後面沒有動作了就return 1
}
```

三、 製作 yacc（.y 檔）

主要架構：

先定義有甚麼 Token（Lex 才有辦法判斷），規則定義 Grammar 型式

```
... definitions ...
%%
... rules ...
%%
... subroutines ...
```

規則定義：

```
<加法> ::= <數字> <+> <數字>
```

（BNF 表示）

```
expr: NUMBER '+' NUMBER
```

（將 BNF 轉為 YYAC 語法）

```
expr: NUMBER '+' NUMBER
 $$    $1    $2   $3
```

（不同單位的對應屬性）

定義先乘除後加減、左相關右相關：

若是有多個符號要被定義優先級，則「後定義」的符號具有較高的優先級。

```
%left '+' '-'          /* %left:(1+2)+3 */
%left '*' '/'          /* %right:1+(2+3)*/
```

YACC 原始程式風格：

（1）終端符名全部用大寫字母，非終端符全部用小寫字母；

（2）把語法規則和語義動作放在不同的行；

（3）把左部相同的規則寫在一起，左部只寫一次，而後面所有規則都寫在豎線“|”之後；

（4）把分號“；”放在規則最後，獨佔一行；

（5）用製表符來對齊規則和動作。

宣告、定義：

```
%{
#include "main.h"

void yyerror(const char *s);
extern int yylex(); /*用來讀取檔案或文字 */
extern int yyparse();

%}

%token NUMBER
%left '+' '-'          /* %left:(1+2)+3 */
%left '*' '/'          /* %right:1+(2+3)*/
%nonassoc UMINUS       /* %nonassoc:用來表示最高優先級  解決"負號"跟"減法"為同一個符號的問題*/
```

補充：%nonassoc 的意思是沒有結合性。它一般與%prec 結合使用表示該操作有相同的優先權。

規則：

```
%%

func:
      expr '='              { printf("Result: %d\n", $1);/*輸出計算結果*/ }
    ;

expr:
      NUMBER                { $$ = $1; }
    | expr '+' expr         { $$ = $1 + $3; }
    | expr '-' expr         { $$ = $1 - $3; }
    | expr '*' expr         { $$ = $1 * $3; }
    | expr '/' expr
        {
            if ($3 == 0.0)
            { /*避免除以零的問題！*/
                yyerror("Error: divisor cannot be zero!");
                YYABORT;
            }
            else
            {
                $$ = $1 / $3;
            }
        }
    | '-' expr %prec UMINUS { $$ = -$2; }
    | '(' expr ')'          { $$ = $2; }
    ;

%%
```

子程式：

```
void yyerror(const char *s)
{
    cerr << s << endl;//yacc錯誤處理
}
```

四、 製作主程式（main.cpp）

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

extern int yyparse(void);
extern FILE* yyin; //yyin為 指向要讀取的檔案位置

int main()
{
    const char* sFile = "file.txt";//檔名
    FILE* fp = fopen(sFile, "r");//開啟該檔，並"讀取"
    if (fp == NULL)
    {
        printf("cannot open %s\n", sFile);
        return -1;
    }
    //yytext 為指向目前讀取到的文字
    yyin = fp;//檔案的位置設給yyin
    yyparse();//main函數是呼叫yacc解析入口函數yyparse()。
    fclose(fp);
    return 0;
}
```

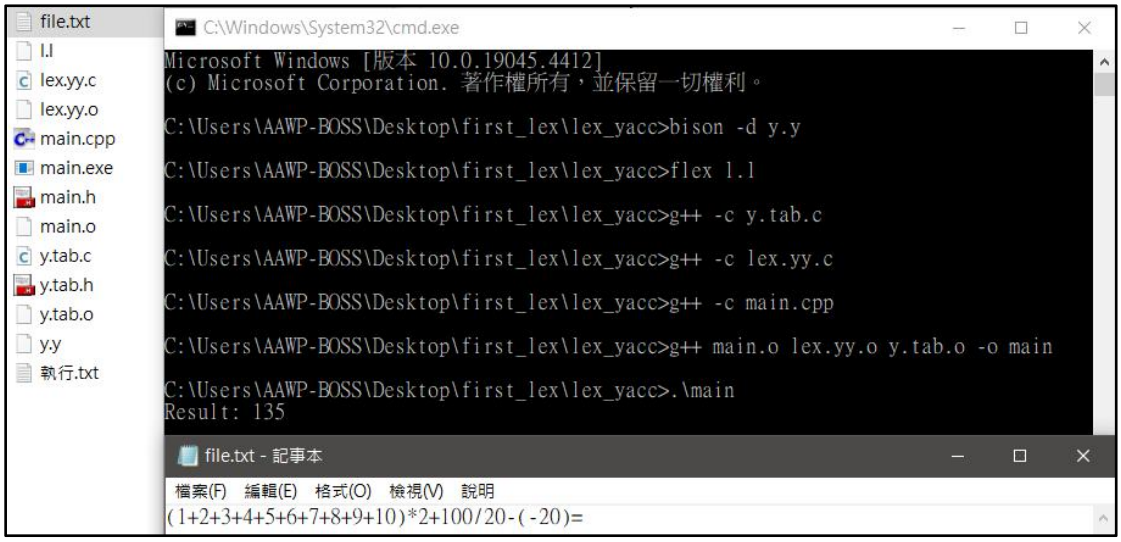
五、 製作標頭檔（main.h）

```
#ifndef MAIN_H
#define MAIN_H

#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;

#define YYSTYPE int
#endif
```

六、 Demo

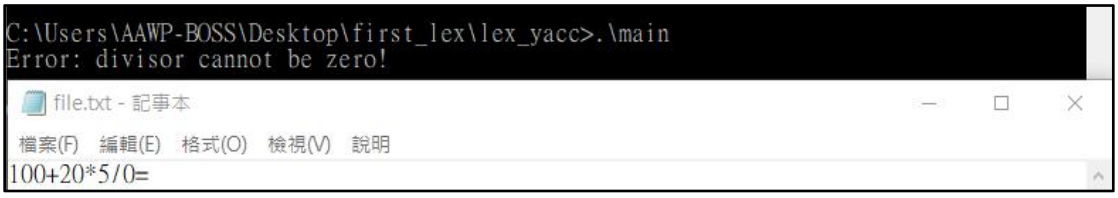
1.單純計算

正確答案應為：135



Result: 135

2.測試若除數為零



Result: Error: divisor cannot be zero!

參考資料：

https://ithelp.ithome.com.tw/articles/10314330

https://redirect.cs.umbc.edu/courses/331/papers/compactGuideLexYacc.pdf

https://www.cnblogs.com/rednodel/p/4500276.html

Team Work

三、LEX 製作流程

1. 宣告及定義：

宣告：先宣告編譯器所需的所有導入和全域變數。

```
%{
        #include "y.tab.h"
        int count=0; /*全域變數，用來記錄行號*/
%}
```

定義：（1.起始狀態的定義 2.特定記號格式的定義）

```
%option yylineno
alpha           [a-zA-Z]
digit           [0-9]
unary           "++"|"--"
```

%option yylineno："yylineno"此變數會被 LEX 自動管理，用來跟蹤當前的行號。

alpha、digit、unary：用來簡化和提高正規表示式的可讀性。

2.規則：主要列出 Lex 的正規表示式以及相對應的操作。在讀取檔案中，偵測到指定樣式的字詞時，該如何標記 token，供後續 yacc 的規則匹配。

語法注意：

(1)每一種不同的樣式(pattern)需要列在該行的最前面。

(2)樣式跟後續的操作函式需要以空白隔開。

(3)若該操作函式有複數行 C 語言，則應以大括號包起來。

```
%%
        Token的規則 {return 屬於哪種Type;}
        [-]?[0-9]+ {return NUMBER;}
        [-]?{digit}+ { return NUMBER; }
%%
```

3. 使用者子程序：一個名為 yywrap()的函式，用來將多個輸入檔打包成一個輸入。

```
int yywrap()
{
    return 1;
}
```

四、yy 功能是什麼？

name	funtion
int yylex(void)	呼叫以啟動詞彙分析器(lexer)，並return Token
char *yytext	指向匹配字串(String)的指標
yyleng	匹配字串(String)的長度
yylval	此Token的值
int yywrap	結束，若已完成return 1，若未完成return 0

五、LEX 小實作

trylex.l

trylex.l

C: > Users > AAWP-BOSS > Desktop > first_lex > 判斷屬於哪種type > trylex.l

```
%{
#include <stdio.h>
%}
%option yylineno

alpha [a-zA-Z]
digit [0-9]
unary "++"|"--"

%%
if |
else |
printf {printf("\n%s is a keyword.", yytext);}
[-]?[0-9]+ {printf("\n%s is a number.", yytext);}
[a-zA-Z]({alpha}|{digit})* {printf("\n%s is a id.",yytext);}
{unary} {printf("\n%s is a unary.", yytext); }
.|\n {ECHO;}
%%

int main()
{
    printf("\n Enter the String:");
    yylex();
}

int yywrap()
{
    return 1;
}

```

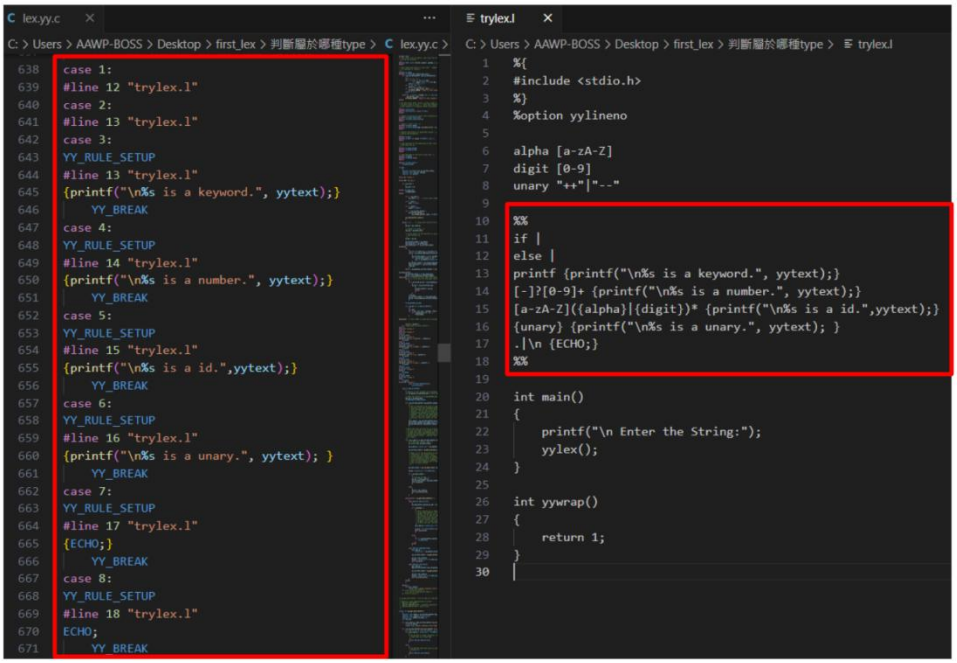
C:\Windows\System32\cmd.exe - a.exe

```
Microsoft Windows [版本 10.0.19045.4412]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\AAWP-BOSS\Desktop\first_lex\判斷屬於哪種type>flex trylex.l

C:\Users\AAWP-BOSS\Desktop\first_lex\判斷屬於哪種type>gcc lex.yy.c

C:\Users\AAWP-BOSS\Desktop\first_lex\判斷屬於哪種type>a.exe

 Enter the String:Hello if elsa else 123 printf -15 ++ -- asd789

Hello is a id.
if is a keyword.
elsa is a id.
else is a keyword.
123 is a number.
printf is a keyword.
-15 is a number.
++ is a unary.
-- is a unary.
asd789 is a id.
```

編譯步驟：

1. **flex trylex.l** → flex 會產生 lex.yy.c 檔（將 trylex.l 的規則翻譯成 C 程式碼格式的函數 yylex()，並輸出到 lex.yy.c 檔中，則 User subroutines 複製到此檔的最後面）
2. **gcc lex.yy.c** → 用 gcc 編譯，產生 a.exe
3. **a.exe** → 可執行

lex.yy.c & trylex.l

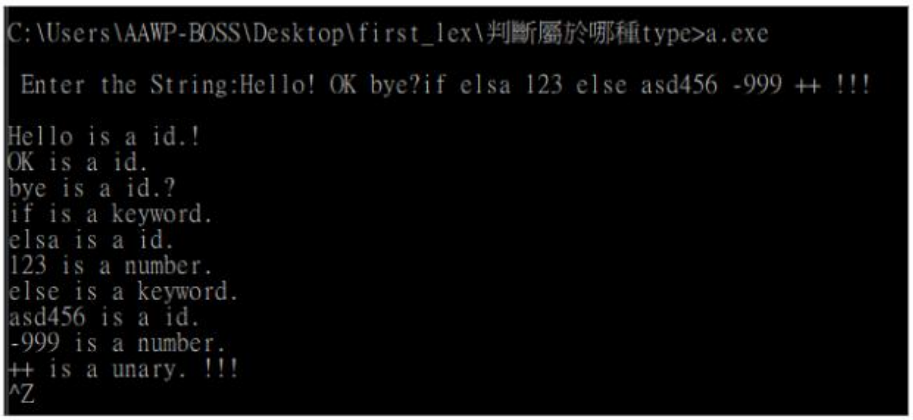


```
case 1:
#line 12 "trylex.l"
case 2:
#line 13 "trylex.l"
case 3:
YY_RULE_SETUP
#line 13 "trylex.l"
{printf("\n%s is a keyword.", yytext);}
    YY_BREAK
case 4:
YY_RULE_SETUP
#line 14 "trylex.l"
{printf("\n%s is a number.", yytext);}
    YY_BREAK
case 5:
YY_RULE_SETUP
#line 15 "trylex.l"
{printf("\n%s is a id.",yytext);}
    YY_BREAK
case 6:
YY_RULE_SETUP
#line 16 "trylex.l"
{printf("\n%s is a unary.", yytext); }
    YY_BREAK
case 7:
YY_RULE_SETUP
#line 17 "trylex.l"
{ECHO;}
    YY_BREAK
case 8:
YY_RULE_SETUP
#line 18 "trylex.l"
ECHO;
    YY_BREAK
```

```
%{
#include <stdio.h>
%}
%option yylineno

alpha [a-zA-Z]
digit [0-9]
unary "++"|"--"

%%
if |
else |
printf {printf("\n%s is a keyword.", yytext);}
[-]?[0-9]+ {printf("\n%s is a number.", yytext);}
[a-zA-Z]({alpha}|{digit})* {printf("\n%s is a id.",yytext);}
{unary} {printf("\n%s is a unary.", yytext); }
.|\n {ECHO;}
%%

int main()
{
    printf("\n Enter the String:");
    yylex();
}

int yywrap()
{
    return 1;
}
```

執行結果

執行結果



```
C:\Users\AAWP-BOSS\Desktop\first_lex\判斷屬於哪種type>a.exe

 Enter the String:Hello! OK bye?if elsa 123 else asd456 -999 ++ !!!

Hello is a id.!
OK is a id.
bye is a id.?
if is a keyword.
elsa is a id.
123 is a number.
else is a keyword.
asd456 is a id.
-999 is a number.
++ is a unary. !!!
^Z
```

參考資料:

https://redirect.cs.umbc.edu/courses/331/papers/compactGuideLexYacc.pdf

https://pandolia.net/tinyc/ch8_flex.html