

作業題目: lex - a scanner generator

班級: 2436

學號: D0661526

姓名: 謝安冬

題目:

1. 輸入為 test.txt 檔，內有浮點數(float)、整數(int)、變數(id)，並用分號及換行隔開
2. 變數規則：第一個字必須為字母，後面可以接字母、底線、數字，變數長度不限。
3. 利用 lex 辨別出 test.txt 中浮點數 整數 變數 並在每次辨識完後輸出 float、int、id 或 error，並在最後印出各種 Token 的出現次數。

作法:

創建規則，利用正則表達式匹配浮點數,整數和變數

在匹配的過程中記錄各種 token 出現的個數，并輸出識別出的類型

最後在 main 函數中輸出統計的結果

再用 C++實現全部過程

代碼: test.l

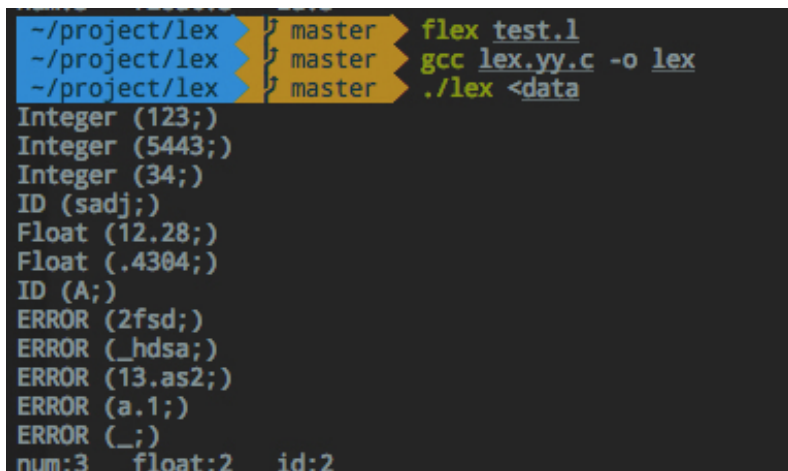
```
%{
    #include<stdio.h>
    int int_num=0;
    int float_num=0;
    int id_num=0;
    int error_num=0;
}%
%option noyywrap
%%
[0-9]+; {
    printf("Integer (%s)\n",yytext);
    int_num++;
}
```

```

}
[0-9]*\.[0-9]+; {
    printf("Float (%s)\n",yytext);
    float_num++;
}
[a-zA-Z][a-zA-Z0-9_]*; {
    printf("ID (%s)\n",yytext);
    id_num++;
}
[n] {}
[0-9a-zA-Z_!@#$.^*()_+={};]+ {
    printf("ERROR (%s)\n",yytext);
}
%%

int main(){
    yylex();
    printf("num:%d   float:%d   id:%d\n",int_num,float_num,id_num);
    return 0;
}

```



```

~/project/lex master flex test.l
~/project/lex master gcc lex.yy.c -o lex
~/project/lex master ./lex <data
Integer (123;)
Integer (5443;)
Integer (34;)
ID (sadj;)
Float (12.28;)
Float (.4304;)
ID (A;)
ERROR (2fsd;)
ERROR (_hdsa;)
ERROR (13.as2;)
ERROR (a.1;)
ERROR (_;)
num:3   float:2   id:2

```

代碼: scanner.cpp

```

//scanner using c++
#include<cstdio>
#include<iostream>
#include<regex>
#include<string>
using namespace std;

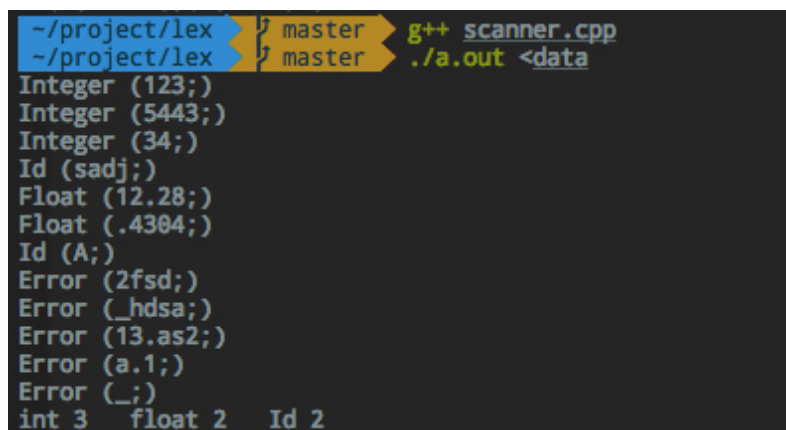
```

```

int main(){
    string str;
    int int_num=0;
    int float_num=0;
    int id_num=0;
    int error_num=0;
    while(cin>>str){
        if(regex_match(str,regex("[0-9]+;"))){
            cout<<"Integer"<<" ("<<str<<")"<<endl;
            int_num++;
        }
        else if(regex_match(str,regex("[0-9]*.[0-9]+;"))){
            cout<<"Float"<<" ("<<str<<")"<<endl;
            float_num++;
        }
        else if(regex_match(str,regex("[a-zA-Z][a-zA-Z0-9_]*;"))){
            cout<<"Id"<<" ("<<str<<")"<<endl;
            id_num++;
        }
        else if(regex_match(str,regex("[0-9a-zA-Z_!@#$.^*()_+=-{};]+"))){
            cout<<"Error"<<" ("<<str<<")"<<endl;
            error_num++;
        }
    }
    cout<<"int "<<int_num<<"    "<<"float "<<float_num<<"    "<<"Id "<<id_num<<endl;

    return 0;
}

```



```

~/project/lex master g++ scanner.cpp
~/project/lex master ./a.out <data
Integer (123;)
Integer (5443;)
Integer (34;)
Id (sadj;)
Float (12.28;)
Float (.4304;)
Id (A;)
Error (2fsd;)
Error (_hdsa;)
Error (13.as2;)
Error (a.1;)
Error ( _;)
int 3    float 2    Id 2

```

討論和心得:

1. 在編寫規則時想不出如何處理無法匹配的情況(輸出 Error), 後來想到利用匹配規則的先後順序, 將 int,float,id 的規則寫在前面, 最後將所有字符匹配, 這樣就可以實現識別 Error 的情況, 類似於 `not(not(token))` 的思想
2. 使用 C++ 寫 scanner 的時候發現 `std::regex` 特性使用不熟練, 查文檔解決