scanner generator

姓名：王冠傑

學號：D1149768

程式碼

```
%{
#include <stdio.h>
#include <string.h>

int int_count = 0;
int float_count = 0;
int id_count = 0;
int error_count = 0;
%}
%option noyywrap
%%

[a-zA-Z][a-zA-Z0-9_]*     { printf("Identifier(ID) : %s\n", yytext); id_count++; }
[a-zA-Z][a-zA-Z0-9_]*[^\x00-\x1Fa-zA-Z0-9_\n]+.* {printf("Error : %s\n", yytext);error_count++;/*只要不符合上述就會報錯
*/}

[-]?[0-9]+                { printf("Integer : %s\n", yytext); int_count++; }
[-]?[0-9]+[^0-9.\x00-\x1F\n]+.* {
    printf("Error : %s\n", yytext);
    error_count++;
}

^[-]?[0-9]+\.[0-9]+        { printf("Float : %s\n", yytext); float_count++; }
^[-]?[0-9]+\.        { printf("Error : %s\n", yytext);error_count++; }
^[-]?[0-9]+\.[^.\x00-\x1F0-9\n]+.*        {printf("Error : %s\n", yytext);error_count++;}
^[-]?[0-9]+\.[0-9]+[^.\x00-\x1F0-9\n]+.*        {printf("Error : %s\n", yytext);error_count++;}
[0-9]+\.[0-9]+\..* {printf("Error : %s\n", yytext);error_count++;}

^[^ \t\n]+[ \t]+[^ \t\n]+.*                { printf("Error : %s\n", yytext); error_count++; }
\n          { /* 忽略換行符 */ }
[ \t] {}
^[^ \-a-zA-Z0-9].* {
    if(yytext[0]>=' ')printf("Error : %s\n", yytext);//因為會把控制符號也一起掃進去
    error_count++;
}

%%

int main() {
    yylex();
    printf("Total_Integer : %d\n", int_count);
    printf("Total_Float : %d\n", float_count);
    printf("Total_Identifier(ID) : %d\n", id_count);
    printf("Total_Error : %d\n", error_count);
    return 0;
}
```

作法

開頭

```
%{
#include <stdio.h>
#include <string.h>

int int_count = 0;
int float_count = 0;
int id_count = 0;
int error_count = 0;
%}
```

- 這裡定義了四個全域變數，用來計算不同類型的匹配數量：`int_count`、`float_count`、`id_count` 和 `error_count`，分別代表整數、浮點數、識別符和錯誤的計數。
- 引入了標準的 C 函式庫 `stdio.h` 和 `string.h`。

讀取ID

- 匹配以字母開頭，並可包含字母、數字或底線 (`_`) 的字符串。
- 若匹配成功，會顯示 "Identifier(ID)" 並計數
- \x00-\x1F排除掉空白前面的控制字串
- [^\x00-\x1Fa-zA-Z0-9_\n]排除非英文、數字、底線、控制字符。也就是特殊字元
- ^放在[ ]裡面是否定的意思
- ^放在[]前面是作為第一行讀取的意思

```
[a-zA-Z][a-zA-Z0-9_]* { printf("Identifier(ID) : %s\n", yytext); id_count++; }
[a-zA-Z][a-zA-Z0-9_]*[^\x00-\x1Fa-zA-Z0-9_\n]+.* {
    printf("Error : %s\n", yytext);
    error_count++;/*只要不符合上述就會報錯*/
}
```

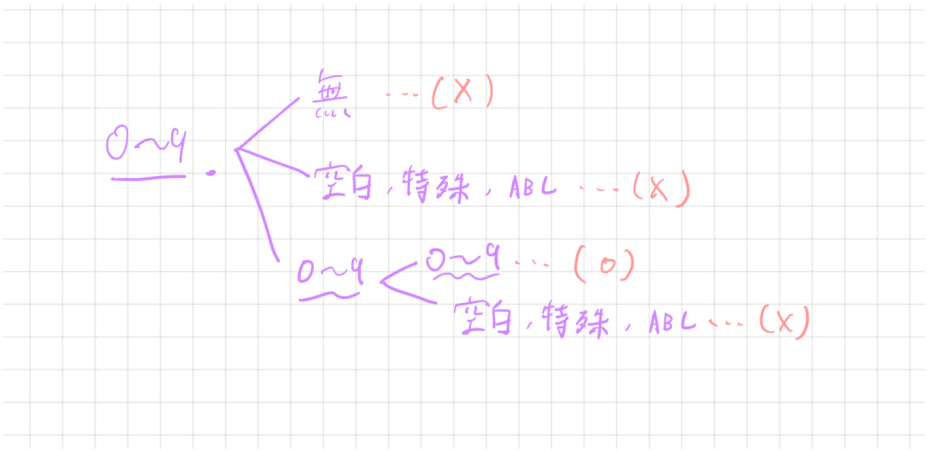
讀取整數

- 匹配整數（可選擇性地以負號 `-` 開頭），若成功則計數。
- ?表示0個或1個
- .*表示遇到特殊字元後 後面的東西不管是什麼都要讀掉

```
[-]?[0-9]+ { printf("Integer : %s\n", yytext); int_count++; }
[-]?[0-9]+[^0-9.\x00-\x1F\n]+.* {printf("Error : %s\n", yytext);error_count++;}
```

讀取小數點

- 匹配浮點數（可選擇性地以負號 `-` 開頭），並計數。



上面四種可能性 都有分別對應方法

```
^[-]?[0-9]+\.[0-9]+ 正確：1.1
^[-]?[0-9]+\. 錯誤：9.
^[-]?[0-9]+\.[^.\x00-\x1F0-9\n]+.* 錯誤：9.~
^[-]?[0-9]+\.[0-9]+[^.\x00-\x1F0-9\n]+.*  錯誤：9.9~
[0-9]+\.[0-9]+\..* 錯誤：9.9.0

c= ^[-]?[0-9]+\.[0-9]+ {printf("Float : %s\n", yytext); float_count++; }
^[-]?[0-9]+\.{printf("Float : %s\n", yytext); float_count++; }
^[-]?[0-9]+\.[^.\x00-\x1F0-9\n]+.*        {
    printf("Error : %s\n", yytext);
    error_count++;
}
^[-]?[0-9]+\.[0-9]+[^.\x00-\x1F0-9\n]+.*  {
    printf("Error : %s\n", yytext);
    error_count++;
}
[0-9]+\.[0-9]+\..*  {printf("Error : %s\n", yytext);error_count++;}
```

讀取空白以及其他

- 忽略換行符和空白字符。
- 若開頭是非法字符且非控制字符，則標記為錯誤。

```
^[^ \t\n]+[ \t]+[^ \t\n]+.* { printf("Error : %s\n", yytext); error_count++; }
//如果字與字之間有空格也是錯誤
\n { /* 忽略換行符 */ }
[ \t] { /* 忽略空白與tab */ }
^[^ \-a-zA-Z0-9].* { //只要開頭是特殊字元就跳過直到換行
    if(yytext[0]>=' ')printf("Error : %s\n", yytext);//因為會把控制符號也一起掃進去
```

```
    error_count++;
}
```

主函式

- `main()` 函式會呼叫 `yylex()` 來啟動 Lex 分析。
- 分析完成後，輸出各類型的匹配總數。

```
int main() {
    yylex();
    printf("Total_Integer : %d\n", int_count);
    printf("Total_Float : %d\n", float_count);
    printf("Total_Identifier(ID) : %d\n", id_count);
    printf("Total_Error : %d\n", error_count);
    return 0;
}
```

執行結果：

測試資料

```
123;
12jj12
-54564
12
9-
0.00002
-222.1111;
A_XX2;
 ABD_UU_X
ABD_UU_X~
.12345
1.0.1.0;
CACHE_  ;
A  A  A~
1 1 3 4
A
A
2.

~ab ~df aa
_ABX
123ABC;
```

```
~
.
```

執行結果

```
Error : 123;
Error : 12jj12
Integer : -54564
Integer : 12
Error : 9-
Float : 0.00002
Error : -222.1111;
Error : A_XX2;
Identifier(ID) : ABD_UU_X
Error : ABD_UU_X~
Error : .12345
Error : 1.0.1.0;
Error : CACHE_  ;
Error : A  A  A~
Error : 1 1 3 4
Identifier(ID) : A
Identifier(ID) : A
Error : 2.
Error : ~ab ~df aa
Error : _ABX
Error : 123ABC;
Error : ~
Error : .
Total_Integer : 2
Total_Float : 1
Total_Identifier(ID) : 3
Total_Error : 17
```

討論&心得

遇到非常多問題，花了很多時間也debug到零辰五點。遇到的問題像是，要怎麼讀到換行才結束。或是怎麼把一整行當作一個token。又或是怎麼看到字與字之間有空格就判斷是錯的。另外這個程式還有可能讀取到ascii code 空白之前的控制字元，使我找不出錯誤在哪。然後判斷id、int、float有沒有特殊字元也下了功夫，尤其是浮點數會出現的狀況最多，要做最多的判斷。

加分題

程式名字是scannerGenerator.c

程式碼

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

int total_int = 0;
int total_float = 0;
int total_id = 0;
int total_error = 0;

int isInteger(char *str) {
    if (str[0] == '+' || str[0] == '-') str++;
    if (*str == '\0') return 0;
    while (*str) {
        if (!isdigit(*str)) return 0;
        str++;
    }
    return 1;
}

int isFloat(char *str) {
    int has_dot = 0;
    int has_digit_before_dot = 0;
    int has_digit_after_dot = 0;
    if (str[0] == '+' || str[0] == '-') str++;
    if (*str == '\0') return 0;
    while (*str) {
        if (*str == '.') {
            if (has_dot) return 0;
            has_dot = 1;
        } else if (isdigit(*str)) {
            if (!has_dot) {
                has_digit_before_dot = 1;
            } else {
                has_digit_after_dot = 1;
            }
        } else {
            return 0;
        }
        str++;
    }
    return has_dot && has_digit_before_dot && has_digit_after_dot;
}

int isIdentifier(char *str) {
    if (!isalpha(str[0])) return 0;
    str++;
    while (*str) {
        if (!isalnum(*str) && *str != '_') return 0;
        str++;
    }
    return 1;
}

void classifyToken(char *token) {
    if (isInteger(token)) {
        printf("Integer : %s\n", token);
        total_int++;
    } else if (isFloat(token)) {
        printf("Float : %s\n", token);
        total_float++;
    } else if (isIdentifier(token)) {
        printf("Identifier(ID) : %s\n", token);
        total_id++;
    } else {
        printf("Error : %s\n", token);
        total_error++;
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL) {
        printf("Error opening file: %s\n", argv[1]);
        return 1;
    }

    char line[256];

    while (fgets(line, sizeof(line), file)) {
        line[strcspn(line, "\n")] = 0;

        char *start = line;
        while (isspace(*start)) start++;
        char *end = start + strlen(start) - 1;
        while (end > start && isspace(*end)) *end-- = '\0';


        if (*start == '\0') continue;

        if (strchr(start, ' ') != NULL || strchr(start, ';') != NULL) {
            printf("Error : %s\n", start);
            total_error++;
        } else {
            classifyToken(start);
        }
    }

    fclose(file);

    printf("\nTotal_Integer : %d\n", total_int);
    printf("Total_Float : %d\n", total_float);
    printf("Total_Identifier(ID) : %d\n", total_id);
    printf("Total_Error : %d\n", total_error);

    return 0;
}
```

解釋

判斷ID

- 第一個字元必須是字母。
- 其後可以是字母、數字或下劃線。

```
int isIdentifier(char *str) {
    if (!isalpha(str[0])) return 0; // 首字必須為字母
    str++;
    while (*str) {
        if (!isalnum(*str) && *str != '_') return 0; // 允許字母、數字、下劃線
        str++;
    }
    return 1;
}
```

判斷整數

- 若首字元是 `+` 或 `-`，則跳過。
- 如果字串為空，則不是整數。
- 使用 `isdigit()` 檢查每個字元是否為數字。
- 如果所有字元都是數字，則返回 `1`，表示它是整數。

```
int isInteger(char *str) {
    if (str[0] == '+' || str[0] == '-') str++;
    if (*str == '\0') return 0;
    while (*str) {
        if (!isdigit(*str)) return 0;
        str++;
    }
    return 1;
}
```

判斷浮點數

- 檢查是否有正負號。
- 使用 `has_dot` 記錄是否有小數點，並檢查它不能重複。
- 確保小數點前後都有至少一個數字。

```
int isFloat(char *str) {
    int has_dot = 0;
```

```
    int has_digit_before_dot = 0;
    int has_digit_after_dot = 0;
    if (str[0] == '+' || str[0] == '-') str++;
    if (*str == '\0') return 0;
    while (*str) {
        if (*str == '.') {
            if (has_dot) return 0; // 不能有超過一個小數點
            has_dot = 1;
        } else if (isdigit(*str)) {
            if (!has_dot) {
                has_digit_before_dot = 1; // 小數點前有數字
            } else {
                has_digit_after_dot = 1; // 小數點後有數字
            }
        } else {
            return 0; // 如果有其他符號，不是浮點數
        }
        str++;
    }
    // 要有小數點，並且小數點前後都需要至少一個數字
    return has_dot && has_digit_before_dot && has_digit_after_dot;
}
```

分類 token

- 依次檢查是否為整數、浮點數、識別符，否則將其歸類為錯誤。

```
void classifyToken(char *token) {
    if (isInteger(token)) {
        printf("Integer : %s\n", token);
        total_int++;
    } else if (isFloat(token)) {
        printf("Float : %s\n", token);
        total_float++;
    } else if (isIdentifier(token)) {
        printf("Identifier(ID) : %s\n", token);
        total_id++;
    } else {
        printf("Error : %s\n", token);
        total_error++;
    }
}
```