编译原理实验二

Flex提取C/C++程序整数和浮点数

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

一、 实验目的

用Flex实现下面的功能：输入一个合法的C/C++程序，提取出程序中的整数和浮点数，并统计各自出现的次数。注意像变量名或函数名中包含的数字不应统计进去。

加分项：忽略掉注释里面的整数和浮点数。

提交一份简要的实验报告，实验报告应包括程序功能描述、Flex文件的代码和若干实验的结果截图。

二、 程序功能描述

由老师上周提供的Flex模板，只需修改前面初始化变量和正则表达式匹配的部分即可。后面主函数部分采用文件读入，并将结果输出。

由于要统计整数和浮点数的数量，因此在初始化区初始两个计数变量`cnt_int`和`cnt_float`，并赋初值为0。

虽然只需要提取程序中的整数和浮点数，但是为了排除一些不被提取的情况，因此另外附加对变量名和注释的识别。

1. 识别整数

对于整数，我们有下面的正则定义

$$sign \rightarrow + \mid - \mid \epsilon$$

$$digit \rightarrow 0 \mid 1 \mid \cdots \mid 9$$

$$num \rightarrow sign\ digit\ digit^*$$

因此可以得到正则表达式

```
[+-]?[0-9][0-9]*
```

其中?代表零个或一个。

2. 识别浮点数

对于浮点数，我们有如下定义

$$
\begin{aligned}
sign &\rightarrow + \mid - \mid \epsilon \\
digit &\rightarrow 0 \mid 1 \mid \cdots \mid 9 \\
digits &\rightarrow digit\ digit^{*} \\
optional\_fraction &\rightarrow .\ digits \mid \epsilon \\
optional\_exponent &\rightarrow ((E \mid e)\ (+ \mid - \mid epsilon)\ digits) \mid epsilon \\
num &\rightarrow sign\ digits\ optional\_fraction\ optional\_exponent
\end{aligned}
$$

可得正则表达式为

```
[+-]?[0-9]+(\.[0-9]+)?([Ee][+-]?[0-9]+)?
```

注意浮点数表达式和整数表达式在Flex中放置的先后次序。由于浮点数中必然包含整数，如果将浮点数的正则表达式放在前面，Flex则会提示整数的正则表达不会被匹配（warning, rule cannot be matched），因此应该先处理整数后处理浮点。

3. 变量名处理

由于C/C++的变量和函数名都是可以含有数字的，因此为了避免将这些变量名中的数字提取出来，需要在提取整数和浮点数之前先将这些变量名进行解析。

C/C++的变量名非常简单，大小写字母、数字加下划线，首个字符不能为数字，因此可得正则表达式

```
[a-zA-Z_][a-zA-Z0-9]*
```

4. 注释处理

同样，对于注释中的数字我们也是不需要将其进行统计的，因此我们在**最开始**就应该对注释语句进行处理。 C/C++中有两种注释的写法，一种是行末的注释`//`，另一种是可跨行的注释`/* */`。

对于前者，正则表达式非常简单

```
(//.*)
```

对于后者，则需要考虑到跨行的情况。初步尝试可以得到下面的正则表达式，两侧是斜杠加星号，中间可以是任意字符，也包括了换行符

```
/\*(.|[\r\n])*?\*/
```

其中?代表非贪心(non-greedy)匹配，意味着只要顺序第一个匹配上该子表达式，这个正则表达式就算匹配完成了，这样可以有效避免多个注释全部整合到一起的情况。虽然这种Lazy模式的匹配在C++<regex>库及大量文本编辑器中都支持，但是非常遗憾的是Flex并不提供这种特性，因此需要重新再对其修改。

不采用非贪心匹配的做法则和理论作业二的第二题比较类似，我们需要考虑中间出现多个星号的情况。对于注释中间的字符

- 要么不是星号，即`[^*]`
- 要么是换行符，即`[\r\n]`
- 要么星号后面不跟斜杠，而且这里的星号可以是一个或多个，有`(\*+([^*/]|[\r\n]))`

整合起来得到

```
/\*([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*\*/
```

这个正则表达式可以有效处理多行注释的情况。

再与前面的行末注释的表达式做或操作，即可得到最终匹配所有注释的正则表达式。

5. 完整代码

将上述内容整合到一起，可得下面完整的`lex.l`程序。注意实现过程中需要将斜杠和星号进行转义输出。

```
/*** Definition Section has one variable
which can be accessed inside yylex()
and main() ***/
%{
    int cnt_int = 0;
    int cnt_float = 0;
%}

%%
(\/\*([^*]|[\r\n]|(\*+([^*\/]|[\r\n])))*\*+\/)|(\/\/.*) ; /* firstly ignore the
    ↪ comments */
[a-zA-Z_][a-zA-Z0-9]* ; /* variable names */
[+-]?[0-9]+ {
    printf("Int: %s\n", yytext);
    cnt_int++;
}
[+-]?[0-9]+(\.[0-9]+)?([Ee][+-]?[0-9]+)? {
    printf("Float: %s\n", yytext);
    cnt_float++;
}
.           ;
```

```
[ \t\r\n] ; /* skip whitespace which is not part of a string */
%%

/*** Code Section prints the number of
capital letter present in the given input***/
int yywrap(){}
int main(){

// Explanation:
// yywrap() - wraps the above rule section
/* yyin - takes the file pointer
        which contains the input*/
/* yylex() - this is the main flex function
        which runs the Rule Section*/
// yytext is the text in the buffer

    FILE *fp = fopen("lex_test.cpp","r");
    yyin = fp;

    yylex();
    printf("\n# of Integers: %d"
           "\n# of Floats: %d\n", cnt_int, cnt_float);

    return 0;
}
```

三、 实验结果

我将**所有的测试样例**都整合进lex_test.cpp文件中，如下所示。

```
#include <cctype>
#include <cstdio>
using namespace std;

/*
 * A large comment
 * with numbers
 * 1, 2,3, 4, 5...
 */

#define PI 3.14159E0
typedef __int8_t myuint8;

struct A
{
    int val1 = -1;
```

```
17      float val_2 = 100.1001;
18  };
19
20  int main() {
21      /* inline f12s 234 14.234 */
22      int exp = (1 +3)* 2 / 4; // expressions
23      int asf24e =45;
24      float _123ab = -0.1;
25      printf("Print a float %f\n", 10e-5);
26      for (int i = 10; i>-2;--i);
27      A a;
28      a.val1 = 404;
29      a.val_2 = 99.9;
30      return 0;
31  }
```

这里涵盖了以下这些测试样例：

- 多行注释、多星号
- 注释内含数字、变量等
- 不同位置的多种不同注释（行内及行末）
- 科学计数法表示浮点数
- 类型名、变量名、结构体成员名含数字
- 数字前后空格数目不等
- 单一表达式内含多个数字
- 数字含符号
- 数字作为右值
- 数字以不同符号作为前导

运行结果如图1所示，可以看到我的分析器成功将程序中的整数和浮点数提取了出来，同时也排除了注释中的数字。

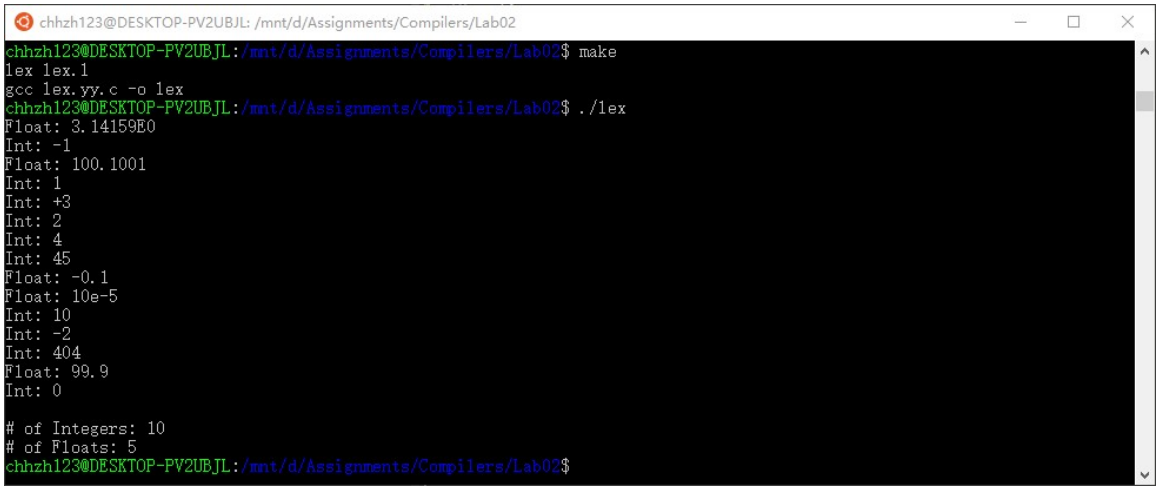
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/Compilers/Lab02
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/Compilers/Lab02$ make
lex lex.l
gcc lex.yy.c -o lex
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/Compilers/Lab02$ ./lex
Float: 3.14159E0
Int: -1
Float: 100.1001
Int: 1
Int: +3
Int: 2
Int: 4
Int: 45
Float: -0.1
Float: 10e-5
Int: 10
Int: -2
Int: 404
Float: 99.9
Int: 0
# of Integers: 10
# of Floats: 5
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/Compilers/Lab02$

图 1: 编译运行结果