实验报告

实验目的

借助flex工具实现一个词法分析器，将SysY代码中的单词和符号进行分类，然后按照单词符号出现顺序 依次输出：原始单词符号、种类、出现在源程序的位置（行数和列数）。

实验流程

本实验利用flex自动生成工具实现代码生成。为此，我们需要生成一个.l文件，其中主要包括两个部分：

声明

在声明中，我们需要引入三个头文件：<stdio.h>, <stdlib.h>, <string.h>。

用户子程序

在用户子程序中，我们只需要调用两个函数。首先将yyin即需要分析的文件设置为输入文件名，之后再调用yylex函数进行分析。

```
int main(int argc, char *argv[]) {
    yyin = fopen(argv[1], "r");
    yylex();
    return 0;
}
```

识别规则

在此part中，我们需要实现相应字符串对应的类别和该字符串在源程序中对应的位置。因此，我们需要引入一个变量column来记录列数并引入yylinno来记录行数，在每次成功匹配字符串时，都需要将字符串的长度加入column。对于字符串的类别分析，我们需要引入正规式来表示。

keyword

正规式

```
main|auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|const|float|short|unsigned|continue|for|signed|void|default|goto|sizeof|volatile|do|if|while|static
```

解释

此正规式包含了C语言的所有关键字。

identifier

正规式

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

解释

此正规式匹配由字母、数字和下划线组成的标识符，其中第一个字符必须是字母或下划线。

具体解释如下：

- `[a-zA-Z_]` 匹配一个字母或下划线。方括号 `[]` 表示一个字符集，其中包含了所有允许的字符。
- `[a-zA-Z0-9_]*` 匹配零个或多个由字母、数字和下划线组成的字符。星号 `*` 表示匹配零个或多个前面的字符。

operator

正规式

```
[\+\-\*/%&|\^~!<>=]=?|\|\||&&|<<|>>|\+\+|--|->|\.
```

解释

此正规式包含C语言全部的算符。

constant

正规式

```
0[xX][0-9A-Fa-f]+)|(0[0-7]+)|[0-9]+|([0-9]*\.[0-9]+|[0-9]+\.[0-
9]*)|'([^'\n]|\\')*'|true|false
```

解释

此正规式匹配整数、浮点数、八进制和十六进制数、布尔常量和单引号括起来的字符或转义序列。

具体解释如下:

- `(0[xX][0-9A-Fa-f]+)` 匹配十六进制数，其中 `0[xX]` 匹配前缀 `0x` 或 `0X`， `[0-9A-Fa-f]+` 匹配十六进制数的数字部分。
- `(0[0-7]+)` 匹配八进制数，其中 `0` 是前缀， `[0-7]+` 匹配八进制数的数字部分。
- `[0-9]+` 匹配十进制数。
- `([0-9]*\.[0-9]+|[0-9]+\.[0-9]*)` 匹配浮点数，其中 `[0-9]*\.[0-9]+` 匹配小数点前有可选的整数部分和小数部分， `[0-9]+\.[0-9]*` 匹配小数点前必须是整数的小数部分。
- `'([^'\n]|\\')*` 匹配单引号括起来的字符或转义序列。其中 `[^'\n]` 表示不是单引号和换行符的任意字符， `\\'` 表示一个转义的单引号。
- `true` 和 `false` 匹配布尔常量。

delimiter

正规式

```
[{}()\[\],.;:]
```

解释

此正规式包含C语言全部的界符。

note

正规式

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

解释

此正规式匹配 C/C++ 语言中的注释，包括多行注释和单行注释。

具体解释如下：

- `\/\*` 匹配多行注释的开始，即 `/*`。
- `([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*` 匹配多行注释中的内容，包括任意字符、回车换行符和星号。其中 `[^*]` 表示不是星号的任意字符，`[\r\n]` 匹配回车换行符，`\*+` 匹配一连串的星号，`([^*/]|[\r\n])` 表示星号后面不是斜线的任意字符或回车换行符。
- `\*+\/` 匹配多行注释的结束，即以一串星号结尾的斜线，例如 `*/`。
- `|` 表示或，用于匹配多行注释和单行注释。
- `"\/\/"` 匹配单行注释的开始，即 `//`。
- `(.)*` 匹配单行注释中的任意字符，包括回车换行符。

other

正规式

```
[0-9][a-zA-Z0-9_]*|.
```

解释

此正规式匹配一个数字和一个标识符（由字母、数字和下划线组成的字符串），或者一个任意字符，即以数字开头的错误标识符。

具体解释如下：

- `[0-9][a-zA-Z0-9_]*` 匹配一个标识符，其中 `[0-9]` 匹配标识符的第一个字符为数字，`[a-zA-Z0-9_]*` 匹配后续的字母、数字或下划线。
- `|` 表示或，用于匹配标识符或者任意字符。
- `.` 匹配任意字符，除了换行符。

实验结果

为了验证词法分析器的正确性，我们利用1.sy-6.sy SysY文件作为输入文件检测

1.sy

```
nt: K, (1, 1)
main: K, (1, 5)
(: D, (1, 9)
): D, (1, 10)
{: D, (1, 11)
int: K, (2, 5)
b: I, (2, 9)
[: D, (2, 10)
10: C, (2, 11)
]: D, (2, 13)
```

```
[: D, (2, 14)
10: C, (2, 15)
]: D, (2, 17)
[: D, (2, 18)
10: C, (2, 19)
]: D, (2, 21)
,: D, (2, 22)
a: I, (2, 23)
;: D, (2, 24)
b: I, (3, 5)
[: D, (3, 6)
2: C, (3, 7)
]: D, (3, 8)
[: D, (3, 9)
2: C, (3, 10)
]: D, (3, 11)
[: D, (3, 12)
2: C, (3, 13)
]: D, (3, 14)
=: O, (3, 15)
010: C, (3, 16)
;: D, (3, 19)
int: K, (4, 5)
test: I, (4, 9)
=: O, (4, 13)
21474836472147483647: C, (4, 14)
;: D, (4, 34)
int: K, (5, 5)
test: I, (5, 9)
=: O, (5, 13)
0xabcde: C, (5, 14)
;: D, (5, 21)
int: K, (6, 5)
666bbb666: T, (6, 9)
;: D, (6, 18)
int: K, (7, 5)
aaa111aaa: I, (7, 9)
;: D, (7, 18)
a: I, (8, 5)
=: O, (8, 6)
b: I, (8, 7)
[: D, (8, 8)
2: C, (8, 9)
]: D, (8, 10)
[: D, (8, 11)
2: C, (8, 12)
]: D, (8, 13)
[: D, (8, 14)
2: C, (8, 15)
]: D, (8, 16)
;: D, (8, 17)
while: K, (9, 5)
```

```
(: D, (9, 10)
a: I, (9, 11)
>: O, (9, 12)
0: C, (9, 13)
): D, (9, 14)
{: D, (9, 15)
a: I, (10, 9)
=: O, (10, 10)
a: I, (10, 11)
-: O, (10, 12)
1: C, (10, 13)
;: D, (10, 14)
if: K, (11, 9)
(: D, (11, 11)
a: I, (11, 12)
==: O, (11, 13)
5: C, (11, 15)
): D, (11, 16)
break: K, (12, 13)
;: D, (12, 18)
}: D, (13, 5)
return: K, (14, 5)
a: I, (14, 12)
;: D, (14, 13)
}: D, (15, 1)
```

2.sy

```
int: K, (2, 1)
n: I, (2, 5)
;: D, (2, 6)
int: K, (3, 1)
a: I, (3, 5)
[: D, (3, 6)
10: C, (3, 7)
]: D, (3, 9)
;: D, (3, 10)
int: K, (4, 1)
main: K, (4, 5)
(: D, (4, 9)
): D, (4, 10)
{: D, (5, 1)
n: I, (6, 2)
=: O, (6, 4)
getint: I, (6, 6)
(: D, (6, 12)
): D, (6, 13)
;: D, (6, 14)
if: K, (7, 2)
(: D, (7, 5)
n: I, (7, 6)
>: O, (7, 8)
10: C, (7, 10)
): D, (7, 12)
return: K, (8, 3)
1: C, (8, 10)
;: D, (8, 11)
int: K, (9, 2)
s: I, (9, 6)
;: D, (9, 7)
int: K, (10, 2)
i: I, (10, 6)
;: D, (10, 7)
i: I, (11, 2)
=: O, (11, 4)
0: C, (11, 6)
;: D, (11, 7)
s: I, (12, 2)
=: O, (12, 4)
i: I, (12, 6)
;: D, (12, 7)
while: K, (13, 2)
(: D, (13, 8)
i: I, (13, 9)
<: O, (13, 11)
n: I, (13, 13)
): D, (13, 14)
{: D, (13, 16)
a: I, (14, 4)
[: D, (14, 5)
i: I, (14, 6)
]: D, (14, 7)
=: O, (14, 9)
getint: I, (14, 11)
(: D, (14, 17)
): D, (14, 18)
;: D, (14, 19)
s: I, (15, 4)
=: O, (15, 6)
s: I, (15, 8)
+: O, (15, 10)
a: I, (15, 12)
[: D, (15, 13)
i: I, (15, 14)
]: D, (15, 15)
;: D, (15, 16)
i: I, (16, 4)
=: O, (16, 5)
i: I, (16, 6)
+: O, (16, 7)
1: C, (16, 8)
;: D, (16, 9)
}: D, (17, 2)
```

```
putint: I, (18, 2)
(: D, (18, 8)
s: I, (18, 9)
): D, (18, 10)
;: D, (18, 11)
int: K, (19, 2)
newline: I, (19, 6)
;: D, (19, 13)
newline: I, (20, 2)
=: O, (20, 10)
10: C, (20, 12)
;: D, (20, 14)
putch: I, (21, 2)
(: D, (21, 7)
newline: I, (21, 8)
): D, (21, 15)
;: D, (21, 16)
return: K, (22, 2)
s: I, (22, 9)
;: D, (22, 10)
}: D, (23, 1)
```

3.sy

```
int: K, (1, 1)
deepWhileBr: I, (1, 5)
(: D, (1, 16)
int: K, (1, 17)
a: I, (1, 21)
,: D, (1, 22)
int: K, (1, 24)
b: I, (1, 28)
): D, (1, 29)
{: D, (1, 31)
int: K, (2, 3)
c: I, (2, 7)
;: D, (2, 8)
c: I, (3, 3)
=: O, (3, 5)
a: I, (3, 7)
+: O, (3, 9)
b: I, (3, 11)
;: D, (3, 12)
while: K, (4, 3)
(: D, (4, 9)
c: I, (4, 10)
<: O, (4, 12)
75: C, (4, 14)
): D, (4, 16)
{: D, (4, 18)
int: K, (5, 5)
d: I, (5, 9)
;: D, (5, 10)
d: I, (6, 5)
=: O, (6, 7)
42: C, (6, 9)
;: D, (6, 11)
if: K, (7, 5)
(: D, (7, 8)
c: I, (7, 9)
<: O, (7, 11)
100: C, (7, 13)
): D, (7, 16)
{: D, (7, 18)
c: I, (8, 7)
=: O, (8, 9)
c: I, (8, 11)
+: O, (8, 13)
d: I, (8, 15)
;: D, (8, 16)
if: K, (9, 7)
(: D, (9, 10)
c: I, (9, 11)
>: O, (9, 13)
99: C, (9, 15)
): D, (9, 17)
{: D, (9, 19)
int: K, (10, 9)
e: I, (10, 13)
;: D, (10, 14)
e: I, (11, 9)
=: O, (11, 11)
d: I, (11, 13)
*: O, (11, 15)
2: C, (11, 17)
;: D, (11, 18)
if: K, (12, 9)
(: D, (12, 12)
1: C, (12, 13)
==: O, (12, 15)
1: C, (12, 18)
): D, (12, 19)
{: D, (12, 21)
c: I, (13, 11)
=: O, (13, 13)
e: I, (13, 15)
*: O, (13, 17)
2: C, (13, 19)
;: D, (13, 20)
}: D, (14, 9)
}: D, (15, 7)
}: D, (16, 5)
}: D, (17, 3)
```

```
return: K, (18, 3)
(: D, (18, 10)
c: I, (18, 11)
): D, (18, 12)
;: D, (18, 13)
}: D, (19, 1)
int: K, (21, 1)
main: K, (21, 5)
(: D, (21, 9)
): D, (21, 10)
{: D, (21, 12)
int: K, (22, 3)
p: I, (22, 7)
;: D, (22, 8)
p: I, (23, 3)
=: O, (23, 5)
2: C, (23, 7)
;: D, (23, 8)
return: K, (24, 3)
deepWhileBr: I, (24, 10)
(: D, (24, 21)
p: I, (24, 22)
,: D, (24, 23)
p: I, (24, 25)
): D, (24, 26)
;: D, (24, 27)
}: D, (25, 1)
```

4.sy

```
int: K, (1, 1)
fact: I, (1, 5)
(: D, (1, 9)
int: K, (1, 10)
n: I, (1, 14)
): D, (1, 15)
{: D, (1, 17)
if: K, (2, 3)
(: D, (2, 6)
n: I, (2, 7)
==: O, (2, 9)
0: C, (2, 12)
): D, (2, 13)
{: D, (2, 15)
return: K, (3, 5)
1: C, (3, 12)
;: D, (3, 13)
}: D, (4, 3)
int: K, (5, 3)
nn: I, (5, 7)
;: D, (5, 9)
nn: I, (6, 3)
=: O, (6, 6)
n: I, (6, 8)
-: O, (6, 9)
1: C, (6, 10)
;: D, (6, 11)
return: K, (7, 3)
(: D, (7, 10)
n: I, (7, 11)
*: O, (7, 13)
fact: I, (7, 15)
(: D, (7, 19)
nn: I, (7, 20)
): D, (7, 22)
): D, (7, 23)
;: D, (7, 24)
}: D, (8, 1)
int: K, (10, 1)
main: K, (10, 5)
(: D, (10, 9)
): D, (10, 10)
{: D, (10, 12)
int: K, (11, 3)
n: I, (11, 7)
;: D, (11, 8)
n: I, (12, 3)
=: O, (12, 5)
4: C, (12, 7)
;: D, (12, 8)
return: K, (13, 3)
fact: I, (13, 10)
(: D, (13, 14)
n: I, (13, 15)
): D, (13, 16)
;: D, (13, 17)
}: D, (14, 1)
```

5.sy

```
int: K, (1, 1)
n: I, (1, 5)
;: D, (1, 6)
int: K, (2, 1)
QuickSort: I, (2, 5)
(: D, (2, 14)
int: K, (2, 15)
arr: I, (2, 19)
[: D, (2, 22)
]: D, (2, 23)
,: D, (2, 24)
int: K, (2, 26)
```

```
low: I, (2, 30)
,: D, (2, 33)
int: K, (2, 35)
high: I, (2, 39)
): D, (2, 43)
{: D, (3, 1)
if: K, (4, 5)
(: D, (4, 8)
low: I, (4, 9)
<: O, (4, 13)
high: I, (4, 15)
): D, (4, 19)
{: D, (5, 5)
int: K, (6, 9)
i: I, (6, 13)
;: D, (6, 14)
i: I, (7, 9)
=: O, (7, 11)
low: I, (7, 13)
;: D, (7, 16)
int: K, (8, 9)
j: I, (8, 13)
;: D, (8, 14)
j: I, (9, 9)
=: O, (9, 11)
high: I, (9, 13)
;: D, (9, 17)
int: K, (10, 9)
k: I, (10, 13)
;: D, (10, 14)
k: I, (11, 9)
=: O, (11, 11)
arr: I, (11, 13)
[: D, (11, 16)
low: I, (11, 17)
]: D, (11, 20)
;: D, (11, 21)
while: K, (12, 9)
(: D, (12, 15)
i: I, (12, 16)
<: O, (12, 18)
j: I, (12, 20)
): D, (12, 21)
{: D, (13, 9)
while: K, (14, 13)
(: D, (14, 18)
i: I, (14, 19)
<: O, (14, 21)
j: I, (14, 23)
&&: O, (14, 25)
arr: I, (14, 28)
[: D, (14, 31)
j: I, (14, 32)
]: D, (14, 33)
>: O, (14, 35)
k: I, (14, 37)
-: O, (14, 39)
1: C, (14, 41)
): D, (14, 42)
{: D, (15, 13)
j: I, (16, 17)
=: O, (16, 19)
j: I, (16, 21)
-: O, (16, 23)
1: C, (16, 25)
;: D, (16, 26)
}: D, (17, 13)
if: K, (19, 13)
(: D, (19, 15)
i: I, (19, 16)
<: O, (19, 18)
j: I, (19, 20)
): D, (19, 21)
{: D, (20, 13)
arr: I, (21, 17)
[: D, (21, 20)
i: I, (21, 21)
]: D, (21, 22)
=: O, (21, 24)
arr: I, (21, 26)
[: D, (21, 29)
j: I, (21, 30)
]: D, (21, 31)
;: D, (21, 32)
i: I, (22, 17)
=: O, (22, 19)
i: I, (22, 21)
+: O, (22, 23)
1: C, (22, 25)
;: D, (22, 26)
}: D, (23, 13)
while: K, (25, 13)
(: D, (25, 18)
i: I, (25, 19)
<: O, (25, 21)
j: I, (25, 23)
&&: O, (25, 25)
arr: I, (25, 28)
[: D, (25, 31)
i: I, (25, 32)
]: D, (25, 33)
<: O, (25, 35)
k: I, (25, 37)
): D, (25, 38)
```

```
{: D, (26, 13)
i: I, (27, 17)
=: O, (27, 19)
i: I, (27, 21)
+: O, (27, 23)
1: C, (27, 25)
;: D, (27, 26)
}: D, (28, 13)
if: K, (30, 13)
(: D, (30, 15)
i: I, (30, 16)
<: O, (30, 18)
j: I, (30, 20)
): D, (30, 21)
{: D, (31, 13)
arr: I, (32, 17)
[: D, (32, 20)
j: I, (32, 21)
]: D, (32, 22)
=: O, (32, 24)
arr: I, (32, 26)
[: D, (32, 29)
i: I, (32, 30)
]: D, (32, 31)
;: D, (32, 32)
j: I, (33, 17)
=: O, (33, 19)
j: I, (33, 21)
-: O, (33, 23)
1: C, (33, 25)
;: D, (33, 26)
}: D, (34, 13)
}: D, (35, 9)
arr: I, (37, 9)
[: D, (37, 12)
i: I, (37, 13)
]: D, (37, 14)
=: O, (37, 16)
k: I, (37, 18)
;: D, (37, 19)
int: K, (38, 9)
tmp: I, (38, 13)
;: D, (38, 16)
tmp: I, (39, 9)
=: O, (39, 13)
i: I, (39, 15)
-: O, (39, 17)
1: C, (39, 19)
;: D, (39, 20)
tmp: I, (40, 9)
=: O, (40, 13)
QuickSort: I, (40, 15)
```

```
(: D, (40, 24)
arr: I, (40, 25)
,: D, (40, 28)
low: I, (40, 30)
,: D, (40, 33)
tmp: I, (40, 35)
): D, (40, 38)
;: D, (40, 39)
tmp: I, (41, 9)
=: O, (41, 13)
i: I, (41, 15)
+: O, (41, 17)
1: C, (41, 19)
;: D, (41, 20)
tmp: I, (42, 9)
=: O, (42, 13)
QuickSort: I, (42, 15)
(: D, (42, 24)
arr: I, (42, 25)
,: D, (42, 28)
tmp: I, (42, 30)
,: D, (42, 33)
high: I, (42, 35)
): D, (42, 39)
;: D, (42, 40)
}: D, (43, 5)
return: K, (44, 5)
0: C, (44, 12)
;: D, (44, 13)
}: D, (45, 1)
int: K, (47, 1)
main: K, (47, 5)
(: D, (47, 9)
): D, (47, 10)
{: D, (47, 11)
n: I, (48, 5)
=: O, (48, 7)
10: C, (48, 9)
;: D, (48, 11)
int: K, (49, 5)
a: I, (49, 9)
[: D, (49, 10)
10: C, (49, 11)
]: D, (49, 13)
;: D, (49, 14)
a: I, (50, 5)
[: D, (50, 6)
0: C, (50, 7)
]: D, (50, 8)
=: O, (50, 9)
4: C, (50, 10)
;: D, (50, 11)
```

```
a: I, (50, 12)
[: D, (50, 13)
1: C, (50, 14)
]: D, (50, 15)
=: O, (50, 16)
3: C, (50, 17)
;: D, (50, 18)
a: I, (50, 19)
[: D, (50, 20)
2: C, (50, 21)
]: D, (50, 22)
=: O, (50, 23)
9: C, (50, 24)
;: D, (50, 25)
a: I, (50, 26)
[: D, (50, 27)
3: C, (50, 28)
]: D, (50, 29)
=: O, (50, 30)
2: C, (50, 31)
;: D, (50, 32)
a: I, (50, 33)
[: D, (50, 34)
4: C, (50, 35)
]: D, (50, 36)
=: O, (50, 37)
0: C, (50, 38)
;: D, (50, 39)
a: I, (51, 5)
[: D, (51, 6)
5: C, (51, 7)
]: D, (51, 8)
=: O, (51, 9)
1: C, (51, 10)
;: D, (51, 11)
a: I, (51, 12)
[: D, (51, 13)
6: C, (51, 14)
]: D, (51, 15)
=: O, (51, 16)
6: C, (51, 17)
;: D, (51, 18)
a: I, (51, 19)
[: D, (51, 20)
7: C, (51, 21)
]: D, (51, 22)
=: O, (51, 23)
5: C, (51, 24)
;: D, (51, 25)
a: I, (51, 26)
[: D, (51, 27)
8: C, (51, 28)
```

```
]: D, (51, 29)
=: O, (51, 30)
7: C, (51, 31)
;: D, (51, 32)
a: I, (51, 33)
[: D, (51, 34)
9: C, (51, 35)
]: D, (51, 36)
=: O, (51, 37)
8: C, (51, 38)
;: D, (51, 39)
int: K, (52, 5)
i: I, (52, 9)
;: D, (52, 10)
i: I, (53, 5)
=: O, (53, 7)
0: C, (53, 9)
;: D, (53, 10)
int: K, (54, 5)
tmp: I, (54, 9)
;: D, (54, 12)
tmp: I, (55, 5)
=: O, (55, 9)
9: C, (55, 11)
;: D, (55, 12)
i: I, (56, 5)
=: O, (56, 7)
QuickSort: I, (56, 9)
(: D, (56, 18)
a: I, (56, 19)
,: D, (56, 20)
i: I, (56, 22)
,: D, (56, 23)
tmp: I, (56, 25)
): D, (56, 28)
;: D, (56, 29)
while: K, (57, 5)
(: D, (57, 11)
i: I, (57, 12)
<: O, (57, 14)
n: I, (57, 16)
): D, (57, 17)
{: D, (57, 19)
int: K, (58, 9)
tmp: I, (58, 13)
;: D, (58, 16)
tmp: I, (59, 9)
=: O, (59, 13)
a: I, (59, 15)
[: D, (59, 16)
i: I, (59, 17)
]: D, (59, 18)
```

```
;: D, (59, 19)
putint: I, (60, 9)
(: D, (60, 15)
tmp: I, (60, 16)
): D, (60, 19)
;: D, (60, 20)
tmp: I, (61, 9)
=: O, (61, 13)
10: C, (61, 15)
;: D, (61, 17)
putch: I, (62, 9)
(: D, (62, 14)
tmp: I, (62, 15)
): D, (62, 18)
;: D, (62, 19)
i: I, (63, 9)
=: O, (63, 11)
i: I, (63, 13)
+: O, (63, 15)
1: C, (63, 17)
;: D, (63, 18)
}: D, (64, 5)
return: K, (65, 5)
0: C, (65, 12)
;: D, (65, 13)
}: D, (66, 1)
```

6.sy

```
int: K, (1, 1)
main: K, (1, 6)
(: D, (1, 10)
): D, (1, 11)
{: D, (1, 13)
int: K, (9, 5)
a: I, (9, 9)
;: D, (9, 10)
}: D, (10, 1)
```