PLY入门

实验内容

编写程序，将以下程序中的词法单元都识别出来

```
int asd = 0;
int bc = 10;
while ( asd < bc)
{
    if(bc - asd < 2)
        cout<<"they are close."<<endl;
    asd = asd + 1;
}
```

工具

- **windows**
- **python3.9**
- **ply包**

实验步骤

Token 是一个形式如下的四元组：[type, value, lineno, lexpos] 。

其中 type 表示**词的类型**，value 就是**文本中匹配上的内容**，lineno 表示**词出现的行号**，lexpos 表示**词出现在文本中的第几个字符。**

1. 编写标记列表

```
tokens = [
    'SEMICOLON', 'LB_BR', 'RB_BR','LL_BR', 'RL_BR', 'EQUAL',
    'LT','GT', 'MINUS', 'PLUS', 'LSHIFT', 'RSHIFT',
    'ID','INT', 'STRING',
    'WHILE','IF'
]
```

列表 tokens 就是一个简单的、由字符串组成的元组，其中每一个字符串表示一种 Token 的名称

用来告诉分析器，执行各种分析

2. 匹配规则的声明

定义匹配规则时，需要以**t_**为前缀，并且使用正则表达式的规则来定义匹配

其中，简单的规则匹配格式如下：

```
t_LL_BR = r'\('
```

如果需要在匹配的同时进行其他操作，则需要用如下形式定义：

```
def t_INT(t):
    r'\d+'
    t.value = int(t.value)
    return t
```

为了处理保留字符，同时也为了加快匹配的处理速度，我们加入一个匹配字典**reversed**：

```
reversed = {'if': 'IF', 'int': 'INT', 'while': 'WHILE'}
```

于是，我们在处理较为复杂的字符时，可以先到字典**reversed**中寻找是否存在，提高匹配效率，例如：

```
def t_ID(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = reversed.get(t.value, 'ID')    # Check for reserved words
    return t
```

3. 行号列号

我们定义**t_newline()**函数来处理行号列号：

```
def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)
```

4. 错误处理

当我们发现需要处理的代码中有匹配不上的字符时，我们需要打印错误信息：

```
def t_error(t):
    print("Illegal character '%s'" % t.value[0])
    t.lexer.skip(1)
```

运行结果

```
LexToken(INT,'int',1,0)
LexToken(ID,'asd',1,4)
LexToken(EQUAL,'=',1,8)
LexToken(INT,0,1,10)
LexToken(SEMICOLON,';',1,11)
LexToken(INT,'int',2,13)
LexToken(ID,'bc',2,17)
LexToken(EQUAL,'=',2,20)
LexToken(INT,10,2,22)
LexToken(SEMICOLON,';',2,24)
LexToken(WHILE,'while',3,26)
LexToken(LL_BR,'(',3,32)
LexToken(ID,'asd',3,34)
LexToken(LT,'<',3,38)
LexToken(ID,'bc',3,40)
LexToken(RL_BR,')',3,42)
LexToken(LB_BR,'{',4,44)
LexToken(IF,'if',5,47)
LexToken(LL_BR,'(',5,49)
LexToken(ID,'bc',5,50)
LexToken(MINUS,'-',5,53)
LexToken(ID,'asd',5,55)
LexToken(LT,'<',5,59)
LexToken(INT,2,5,61)
LexToken(RL_BR,')',5,62)
LexToken(ID,'cout',6,66)
LexToken(LSHIFT,'<<',6,70)
LexToken(STRING,'they are close.',6,72)
LexToken(LSHIFT,'<<',6,89)
LexToken(ID,'endl',6,91)
LexToken(SEMICOLON,';',6,95)
LexToken(ID,'asd',7,98)
LexToken(EQUAL,'=',7,102)
LexToken(ID,'asd',7,104)
LexToken(PLUS,'+',7,108)
LexToken(INT,1,7,110)
LexToken(SEMICOLON,';',7,111)
LexToken(RB_BR,'}',8,113)
```