**《编译原理》实验一报告**

**实验一题目：识别标识符**

**姓名：刘洋 学号：16121368 实验日期：2019年3月28日**

# 实验目的：

* 根据 PL/0 语言的文法规范，编写 PL/0 语言的标识符识别程序。
* 通过设计调试标识符识别程序，实现从源程序中分出各个标识符的 方法；加深对课堂教学的理解；为后序词法分析程序的实现打下基 础。
* 掌握从源程序文件中读取有效字符的方法和产生源程序的内部表示 文件的法。
* 掌握识别标识符的实现方法。
* 上机调试完成的识别标识符程序的实现

# 实验内容：

输入 PL/0 语言源程序，输出源程序中所有标识符的出现次数。

# 实验要求：

* 识别程序读入 PL/0 语言源程序（文本文件），识别结果也以文本文 件保存。
* 按标识符出现的顺序输出结果，每个标识符一行，采用二元式序列， 即：(标识符值, 标识符出现次数)
* 源程序中字符不区分大小写，即：“a1”和“A1”是同一个标识符。
* 准备至少 5 组测试用例，每组测试用例包括：输入源程序文件和输 出结果。
* 测试用例应该考虑各种合法标识符的组合情况。

# 算法描述：

该程序用python实现，调用了re库，该库的作用主要在于字符串匹配。算法思路如下：首先将程序通过txt文件按行读取，将所有行连接成为字符串后全部变成小写，然后去掉所有符号，这样得到了该程序所有词的一个list，再分析程序的每一行，将所有含有关键字“var”，“const”，“procedure”的行通过正则匹配找到，再将这几行转换成为单词列表，将这几行的内容去掉之前的三个关键词，去掉所有的单独数字，得到所有的标识符列表，然后在整个程序的而单词列表中，一个个匹配，分别计数，得到程序最终结果。

# 实验代码：

|  |
| --- |
| import re  # 文件按行读取后连接成为一个字符串，变成小写字母的单词  f=open("test.txt","r")  datalines=f.readlines()  datalines\_string="".join(datalines).lower()  # 将文件中所有文本符号删去  allword=re.findall(r"[\w’]+", datalines\_string)  # 按行读取，将所有var和const和procedure的行存下来  data=[]  list=["const","var","procedure"]  for line in datalines:  for i in list:  a=re.findall(i,line,re.I|re.S) #正则表达式匹配，忽略大小写  if a!=[]:  data.append(line)  print(data) #带有const或者var的行  #list内容变成一个字符串string,lower()全部变成小写字母  string="".join(data).lower()  # 将这一个字符串中所有符号删去，变成单词  e=re.findall(r"[\w’]+", string)  print(e)  biaoshifu=[]  # 把const var procedure 数值 都去掉  for i in e:  if re.match("const",i,re.I|re.S)==None and re.match("var",i,re.I|re.S)==None \  and re.match("\d",i,re.I|re.S)==None and re.match("procedure",i,re.I|re.S)==None:  biaoshifu.append(i)  print(biaoshifu)  # 新建一个存放计数的list  count=[0]\*len(biaoshifu)  # 打开输入文件并输入  f2 = open('result.txt','a')  for i in range(len(biaoshifu)):  for word in allword:  if biaoshifu[i]==word:  count[i]=count[i]+1  count[i]=str(count[i]) #输入文件需变成str  print("(",biaoshifu[i],":",count[i],")")  f2.write("("+biaoshifu[i]+":"+count[i]+")\n")  f2.write("\n") |

# 实验结果（输入输出样例）：

1、测试样例1：

输入：

Const num=10;

Var a1,b2;

Begin

Read(A1);

b2:=a1+num;

const a=12;

write(A1,B2);

End.

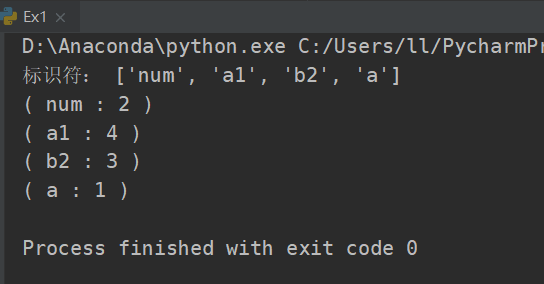
输出：

( num : 2 )

( a1 : 4 )

( b2 : 3 )

( a : 1 )



2、测试样例2：

输入：

var a,b,c,t;

procedure out;

begin

write(a);

write(b);

write(c);

end;

begin

read(a);

read(b);

read(c);

if a>b then

begin

t :=a;

a :=b;

b :=t;

end;

if a>c then

begin

t :=a;

a :=c;

c :=t;

end;

if b>c then

begin

t :=b;

b :=c;

c :=t;

end;

call out;

end.

输出：

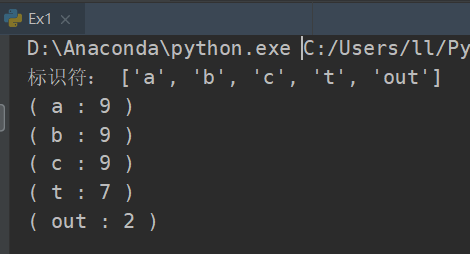
( a : 9 )

( b : 9 )

( c : 9 )

( t : 7 )

( out : 2 )



3、测试样例3：

输入：

var x,y,a,b,c;

procedure fun;

begin

a:=1;

b:=2;

c:=3;

x:=a+b;

end;

begin

call fun;

end.

输出：

( x : 2 )

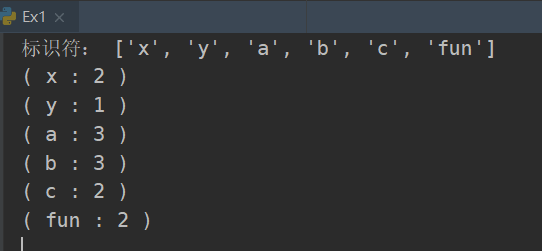
( y : 1 )

( a : 3 )

( b : 3 )

( c : 2 )

( fun : 2 )



4、测试样例4：

输入：

Const a=10;

var b,c;

procedure p;

begin

c:=a+b

end;

begin

read(b);

while b#0 do

begin

call p;write(2\*c);read(b)

end

end.

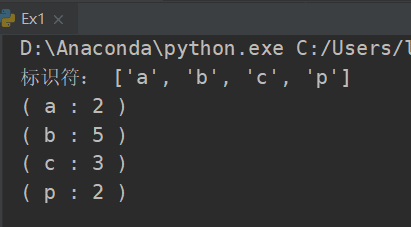
输出：

( a : 2 )

( b : 5 )

( c : 3 )

( p : 2 )



5、测试样例5：

输入：

var a,b,c,t,tag,p;

begin

read(a);

read(b);

read(c);

tag=0;

begin

if a>b then

begin

t :=a;

a :=b;

b :=t;

end;

if a>c then

begin

t :=a;

a :=c;

c :=t;

end;

if b>c then

begin

t :=b;

b :=c;

c :=t;

end

end;

if a+b>c then

tag:=1;

if tag=1 then

begin

write("yes")

if a=b then

p=1;

if b=c then

p=1;

if a=c then

p=0;

if a\*a+b\*b=c\*c then

p=2;

end;

if p=0 then

write("1")

if p=1 then

write("2")

if p=2 then

write("3")

if tag=0 then

write("no")

end.

输出：

( a : 13 )

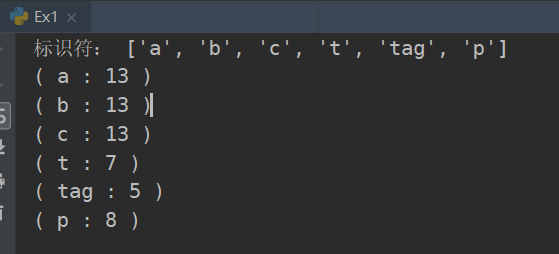
( b : 13 )

( c : 13 )

( t : 7 )

( tag : 5 )

( p : 8 )



# 实验体会：

这是第一次编译原理实验，做的是识别标识符有关的内容，难度并不大，用python的字符串操作按照自己的思路很快就做出了结果。很多意思的点在于很多同学的思路都各不相同，也可以通过一个个的字符从文件中读入来做分析。一开始我只认为var和const后的是标识符，后来才意识到定义函数时procedure后的函数名也算标识符。总体来说，这次实验很好的贴合了编译原理课程所学的内容，从词法分析开始，相信下一次的实验会更体现出词法，文法这方面的内容。