《编译原理》课程实验报告

实验名称	实验一　构造识别符号串的自动机
实验内容	1.用高级语言编写程序：该程序能接受所有的标识符。 2.用高级语言编写程序：该程序能接受所有的常数(整数和定点小数)。 3.用高级语言编写程序：该程序能接受 PL/0 的所有保留字。 4.用高级语言编写程序：该程序能接受 PL/0 的所有界符、运算符。 实验二　词法分析程序的构造 1.单词的分类：可将所有标识符归为一类；将常数归为另一类：保留字、界符、运算符则可采取一词一类。 2.符号表的建立：可事先建立一保留字表，以备识别保留字时进行查询。变量名表及常数表则在词法分析过程中建立。 3.单词串的输出形式：所输出的每一单词，均按形如(CLASS，VALUE)的二元式编码。对于变量标识符和常数，CLASS 字段为相应的类别码，VALUE 字段是该标识符、常数在其符号表中登记项的序号(要求在变量名表登记项中存放该标识符的字符串，常数表登记项中则存放该常数)。对于保留字、界符和运算符，由于采用一词一类的编码方式，所以仅需在二元式的 CLASS 字段上放置相应的单词的类别码，VALUE 字段则为“空”。(或：为便于查看由词法分析程序输出的单词串也可以在 CLASS 字段上放置单词符号串本身)。 4.编写上述词法分析程序

一、实验目的：

进一步掌握词法分析方法的同时，锻炼设计、实现、分析和维护编译程序等方面的初步能力，

二、主要数据结构：

数组：储存单词符号表及翻译后的二元式

class ID: 储存标识符各种信息

id:标识符名称

classs:该标识符数据类型

三、主要设计思想与算法：

1. 首先分析下 PL/0 语言，将单词分为 5 类：保留字、界符、运算符、常量。整

理全部保留字、界符、运算符，由于其个数一定，整理成单词符号表。常量分为整型常数、实型常数，故标识符、整数、实数各为一类，各自另外成表。

2. 分析 PL/0 语言文法规则，构造有限自动机，根据有限自动机编制程序，其中终态是识别出不符合该分支的字符，到达终态则为识别出一个单词，储存后继续识别其他单词。

四、实验结果及测试用例：

测试用例：

```
const a=10,b=+1,c=-10,
    d=3e1,e=+2e2,f=-2e-1,
    g=0.5e-1,h=+10.1e+1;
var m18s11,mw1,m,n,r,q;
{const, m223,;}
procedure gcd;
    begin
        if odd a
            q:=m/n;
        while r#0 do
            begin
                q:=m/n;
                r:=m-q*n;
                m:=n;
                n:=r;
            end
     end;
begin
    read(m);
    read(n);
    {为了方便，规定 m>=n}
    if m<n then
```

```
        begin
            r:=m;m:=n;n:=r;
        end;
    begin
        r:=1+996;
        call gcd;
        write(m);
    end;
end.
```

实验结果：result.txt

单词符号表如下：

class	value	class	value	class	value
class	value	class	value		
0	const	1	var	2	procedure
3	begin	4	end		
5	odd	6	if	7	then
8	call	9	while		
10	do	11	read	12	write
13	.	14	,		
15	;	16	(	17	)
18	+	19	-		
20	*	21	/	22	=
23	#	24	<		
25	<=	26	>	27	>=
28	:=	29	id		

30 int 31 float

标识符表如下：

No id No id No id

No id No id

---

0 ('a', 'const') 1 ('b', 'const') 2 ('c', 'const') 3 ('d', 'const') 4 ('e', 'const')

5 ('f', 'const') 6 ('g', 'const') 7 ('h', 'const') 8 ('ml8s11', 'var') 9 ('mwl', 'var')

10 ('m', 'var') 11 ('n', 'var') 12 ('r', 'var')

13 ('q', 'var') 14 ('gcd', 'var')

整型常数表如下：

No value No value No value

No value No value

---

0 10 1 1 2 -10

3 0 4 996

实型常数表如下：

No value No value No value
No value No value

---

0 30.0 1 200.0 2 -0.2
3 0.05 4 101.0

(0, 'const')(29, 0)(22, '=')(30, 0)(14, ',')(29, 1)(22, '=')(30, 1)(14, ',')(29, 2)(22, '=')(30, 2)(14, ',')(29, 3)(22, '=')(31, 0)(14, ',')(29, 4)(22, '=')(31, 1)(14, ',')(29, 5)(22, '=')(31, 2)(14, ',')(29, 6)(22, '=')(31, 3)(14, ',')(29, 7)(22, '=')(31, 4)(15, ';')(1, 'var')(29, 8)(14, ',')(29, 9)(14, ',')(29, 10)(14, ',')(29, 11)(14, ',')(29, 12)(14, ',')(29, 13)(15, ';')(2, 'procedure')(29, 14)(15, ';')(3, 'begin')(6, 'if')(5, 'odd')(29, 0)(29, 13)(28, ':=')(29, 10)(21, '/')(29, 11)(15, ';')(9, 'while')(29, 12)(23, '#')(30, 3)(10, 'do')(3, 'begin')(29, 13)(28, ':=')(29, 10)(21, '/')(29, 11)(15, ';')(29, 12)(28, ':=')(29, 10)(19, '-')(29, 13)(20, '*')(29, 11)(15, ';')(29, 10)(28, ':=')(29, 11)(15, ';')(29, 11)(28, ':=')(29, 12)(15, ';')(4, 'end')(4, 'end')(15, ';')(3, 'begin')(11, 'read')(16, '(')(29, 10)(17, ')')(15, ';')(11, 'read')(16, '(')(29, 11)(17, ')')(15, ';')(6, 'if')(29, 10)(24, '<')(29, 11)(7, 'then')(3, 'begin')(29, 12)(28, ':=')(29, 10)(15, ';')(29,

10)(28, ':=')(29, 11)(15, ';')(29, 11)(28, ':=')(29, 12)(15, ';')(4, 'end')(15, ';')(3, 'begin')(29, 12)(28, ':=')(30, 1)(18, '+')(30, 4)(15, ';')(8, 'call')(29, 14)(15, ';')(12, 'write')(16, '(')(29, 10)(17, ')')(15, ';')(4, 'end')(15, ';')(4, 'end')(13, '.')

五、实验总结：

通过本实验，我进一步掌握了词法分析方法，自己通过分析语言并且构造有限自动机、编制词法分析程序的过程锻炼了设计、实现、分析和维护编译程序等方面的初步能力。

实验过程中由于对 PL/0 语言掌握信息不足，根据其是 Pascal 语言子集的特点，参考了 Pascal 语言的一些特性。实验的不足之处在于没有进行错误处理，当程序单词不符合规范时，不能显示相应错误。