实验 1 扫描器的设计

1.实验目的

熟悉并实现一个扫描器（词法分析程序）。

2.实验类型

设计型。

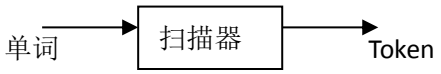
3.实验基本要求

(1) 设计扫描器的有限自动机（识别器）；
(2) 设计翻译、生成 Token 的算法（翻译器）；
(3) 编写代码并上机调试运行通过。
·输入——源程序文件或源程序字符串；
·输出——相应的 Token 序列；
关键字表和界符表；
符号表和常数表；

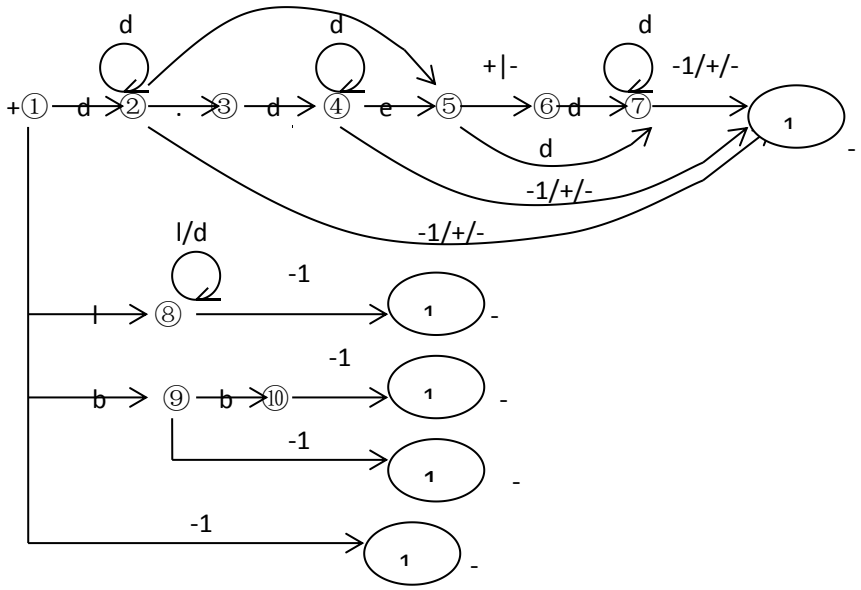
4.预习内容

(1) 有限自动机原理
(2) 词法分析器原理

5.实验基本步骤



(1) 有限自动机的状态转换图



其中：d 为数字，l 为字母，b 为界符，-1 代表其它符号（如在状态 8 处遇到了非字母或数字的其它符号，会变换到状态 12）。

状态转换矩阵：

	d	·	E|e	+|-	l	b	-1
1	2				8	9	15
2	2	3	5	11			11
3	4						
4	4		5	11			11
5	7			6			
6	7						
7	7			11			11
8	8				8		12
9						10	14
10							13
11							
12							
13							
14							
15							

(2) 关键字表和界符表

Program	;
Begin	:
End	(
Var	)
While	,
Do	:=
Repeat	+
Until	-

For	*
To	/
If	>
Then	>=
Else	==
	<
	<=

(3) 数据设计：

①状态转换矩阵：int aut[10][7]={ 2, 0, 0, 0, 8, 9, 15,
2, 3, 5,11, 0, 0, 11,
4, 0, 0, 0, 0, 0, 0,
4, 0, 5,11, 0, 0, 11,
7, 0, 0, 6, 0, 0, 0,
7, 0, 0, 0, 0, 0, 0,
7, 0, 0,11, 0, 0, 11,
8, 0, 0, 0, 8, 0, 12,
0, 0, 0, 0, 0, 10, 14,
0, 0, 0, 0, 0, 0, 13};

②关键字表：

```
char keywords[30][12]={“program”,”begin”,”end”,”var”,”while”,”do”,
                       ”repeat”,”until”,”for”,”to”,”if”,”then”,”else”,
                        “;”, ”:”, ”(“, ”)”, ”,”, ”:=”, ”+”, ”-“, ”*”, ”/”,
                        ”>”, ”>=”, ”==”, “<”, “<=”};
```

③符号表：char ID[50][12]; //表中存有源程序中的标识符

④常数表：float C[20];

⑤其它变量：

```
struct token
        { int code;
          int value};                  //Token 结构
struct token tok[100];          //Token 数组
int s;                          //当前状态
int n,p,m,e,t;        //尾数值，指数值，小数位数，指数符号，类型
float num;                //常数值
char w[50];           //源程序缓冲区
int i;                //源程序指针,当前字符为 w[i]
char strTOKEN[12];    //当前已经识别出的单词
```

(4) 语义动作：

·q_1：n=m=p=t=0; e=1; num=0; 其它变量初始化；

·q_2：n=10*n+(w[i]);

·q_3：t=1;

·q_4：n=10*n+(w[i]); m++;

·q_5：t=1;

·q_6：if '-' then e=-1;
·q_7：p=10*p+(w[i]);
·q_8：将 w[i]中的符号拼接到 strTOKEN 的尾部；
·q_9：将 w[i]中的符号拼接到 strTOKEN 的尾部；
·q_{10}：将 w[i]中的符号拼接到 strTOKEN 的尾部；
//标识符的编码（code）为 1，值（value）为其在符号表中的位置；常数的编码（code）为 2，值（value）为其在常数表中的位置；关键字和界符的编码（code）为其在关键字表中的位置，值（value）为 0。
·q_{11}：num=n*10^{e*p-m};　　//计算常数值
　　t[i].code=2;　t[i].value=InsertConst(num);　　//生成常数 Token
·q_{12}：code=Reserve(strTOKEN);　　//查关键字表
　　if code then { t[i].code=code;　t[i].value=0; } //生成关键字 Token
　　else {t[i].code=1;　t[i].value=InsertID(strTOKEN); }
　　//生成标识符 Token
·q_{13}：code=Reserve(strTOKEN);　　//查界符表
　　if code then { t[i].code=code;　t[i].value=0; }　//生成界符 Token
　　else {
　　strTOKEN[strlen(strTOKEN)-1]='\0';　　//单界符
　　源程序缓冲区指针减 1；
　　code=Reserve(strTOKEN);　　//查界符表
　　t[i].code=code;　t[i].value=0;　　//生成界符 Token
　　}
·q_{14}：code=Reserve(strTOKEN);　　//查界符表
　　t[i].code=code;　t[i].value=0;　　//生成界符 Token
·q_{15}：stop.

(5) 查状态变换表
int find(int s,char ch){　//s 是当前状态，ch 是当前字符，返回值是转换后状态
　查状态转换矩阵 aut[10][7]；
　返回新状态；
}

(6) 主程序流程：
　　初始化；
　　打开用户源程序文件；
　　while （文件未结束）{
　　读入一行到 w[i]，i=0;
　　do　　//处理一行，每次处理一个单词
　　{　滤空格，直到第一个非空的 w[i]；
　　i--;
　　s=1;　　//处理一个单词开始
　　while （s!=0）　　//拼单词并生成相应 Token
　　{

```
            act(s);                                  //执行 qs
            if （s>=11 && s<=14）                    //一个单词处理结束
                break;
            i++;                                     //getchar()
            s=find(s, w[i]);
        }
        if （s==0）
            词法错误；
    }while （w[i]!=换行符）；
}
```
关闭用户源程序文件；
生成 Token 文件；
输出关键字表；
输出 Token 序列；
输出符号表；
输出常数表；