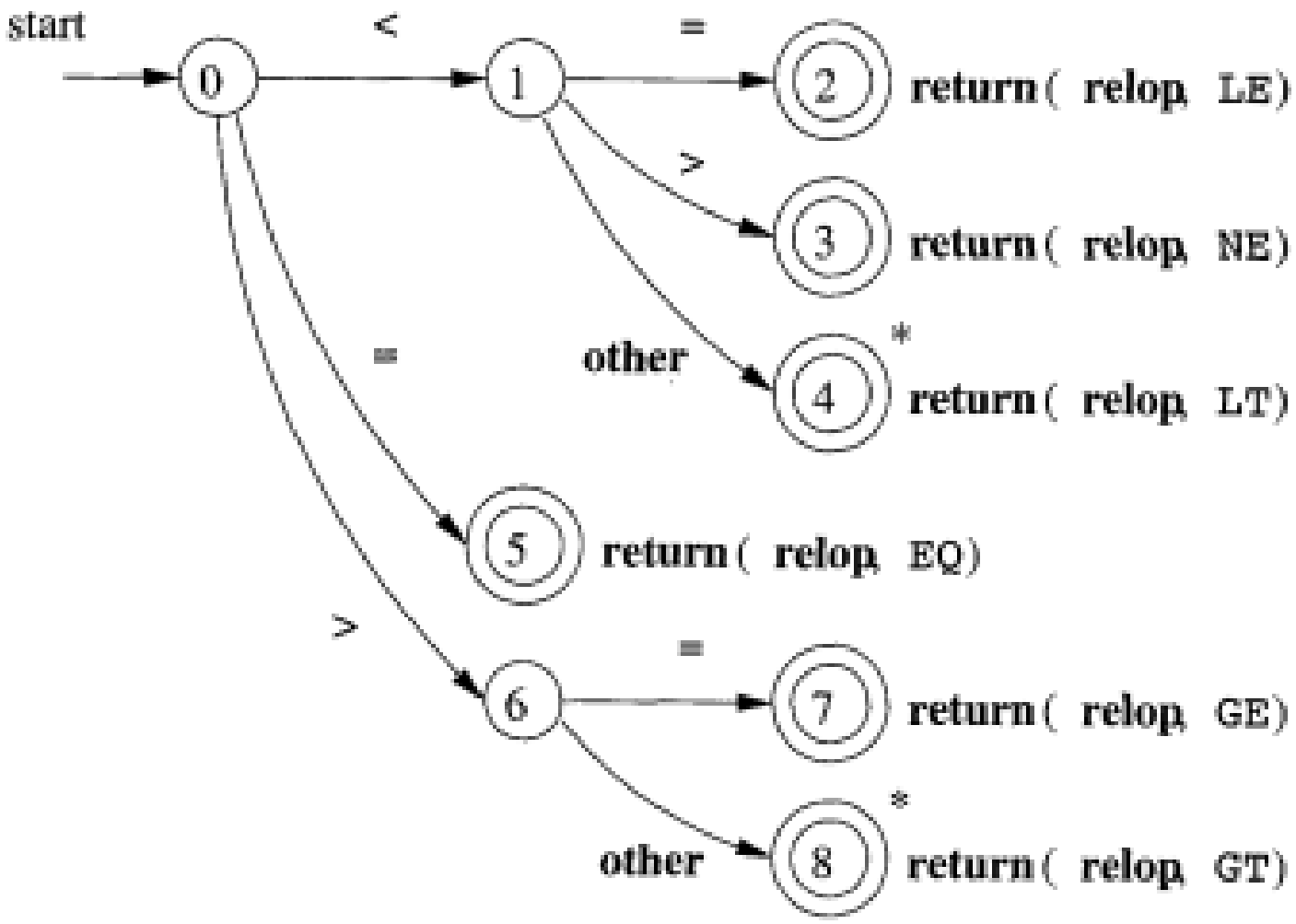
Experiment 1

Aim: Design and implement a lexical analyzer for given language using C and the lexical analyzer should ignore redundant spaces, tabs and new lines.
The lexical analyzer must do the following.

- Open a file containing a C program and read from it.
- Accept the following keywords in the C program and ignore the rest.
  `int,float,char,long,double,if,else,for,while,void,do,switch,case,break`
- Recognize identifiers and numbers.
- Recognize a subset of operators, which is only the relational operators $<, >, <=, >=, !=, ==$
- Draw the complete DFA accepting all of the above, and implement it in C code.
- A portion of the DFA is added in the next slide for your reference.

DFA for accepting relational operators



Things to do

- Expand the DFA to recognize identifiers, keywords, numbers etc.
- Write a C program which does the following.
  - Open a C program file.
  - Read each lexeme and categorize it into a token.
  - Ignore comments and white spaces in the program.
  - Also ignore the other keywords and operators which are not mentioned.
  - Print each recognized token.
- To implement a DFA, we use an integer variable state, and switch statement is used to implement the transitions.

A code fragment recognizing operators

```
1       void main()
2       {
3          int state=0,flag;
4          FILE *f;
5          f=fopen("input.c","r");
6          while(flag!=1)
7          {
8             switch(state)
9             {
10               case 0:
11                    ch=fgetc(f);
12                    if(ch=='<')
13                       state=1;
14                    else if(ch=='!')
15                       state=3;
16                    else if(ch=='>')
17                       state=5;
18                    /* write code to handle other cases such
      as ids, numbers, keywords etc.*/
19
```

A code fragment recognizing operators

```
20              case 1:
21                   ch=fgetc(f);
22                   if(ch=='=')
23                     state=2;
24                   /* other transitions */
25              case 2: //A final state, where we recognize a
     token
26                   printf("<=: is a relational operator\n");
27                   state=0; /*After recognizing a token
    automata must go to its initial state.*/
28
29              /* Implement the remaining part,
30                 first the DFA has to be constructed and
31                 then it has to be implemented in C */
```