Compiler Construction Assignment No 2 - Lexical Scanner

Jawad Ahmed(20P-0165) Section: BCS-6A

March 19, 2023

1 To build your own programming language, define the following

1.1 Rules for identifier name (Regular Expression)

Identifier will always start from @ sign. The corresponding regular expression for identifying identifier will be:

```
@([a-zA-Z])([a-zA-Z0-9])*
```

Some example identifiers accepted by the above regular expression.
`@ali`, `@name`, `@deep_generative_model_params`

1.2 Reserve words including data types such as int, float, string etc

Reserved words will be:

switch	if	auto	int	struct
char	else	goto	default	while
for				

1.3 Operators

Operators will be:

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
=	Assignment
!=	Inequality
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

1.4 Parathesis

The parathesis will be the following:

Symbol	Name
( )	Round parentheses
[ ]	Square brackets
{ }	Curly braces

1.5 Symbol used for end statement (use any symbol other than ; (semicolon))

Hash (#) symbol will be used for end statement.

2 Draw a single DFA for your own language (use JFLAP).

Lexical Analyzer for recognizing,

2.1 Identifiers

Identifier will always start from @ sign. The corresponding regular expression for identifying identifier will be:

```
@([a-zA-Z])([a-zA-Z0-9])*
```

Some example identifiers accepted by the above regular expression.
`@ali`, `@name`, `@deep_generative_model_params`

2.2 Reserverd Words

- `switch`
- `if`
- `auto`
- `int`
- `struct`
- `char`
- `else`
- `goto`
- `default`
- `while`
- `for`

2.3 Operators

- `+`
- `*`
- `-`
- `/`
- `%`
- `>`
- `<`
- `!=`
- `>=`
- `<=`
- `=`

2.4 Parenthesis

- `{`
- `}`
- `(`
- `)`
- `[`
- `]`

the corresponding DFA will be,

2.5 DFA For Operators(shown in Figure 1):

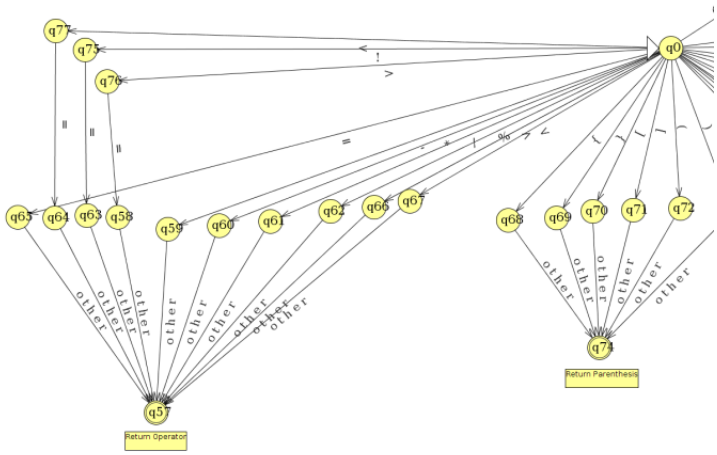

Figure 1: dfa for operator

2.6 DFA For Identifier(shown in Figure 2):

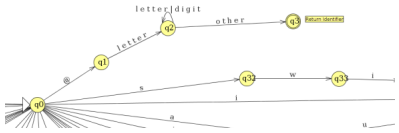

Figure 2: dfa for identifier

2.7 DFA For Reserved words(shown in Figure 3):

2.8 DFA For paranthesis(shown in Figure 4):

3 Write a lexical analyzer of Scanner code for a DFA designed in step 2.

Code is written in scanner.c file attached with the assignment.

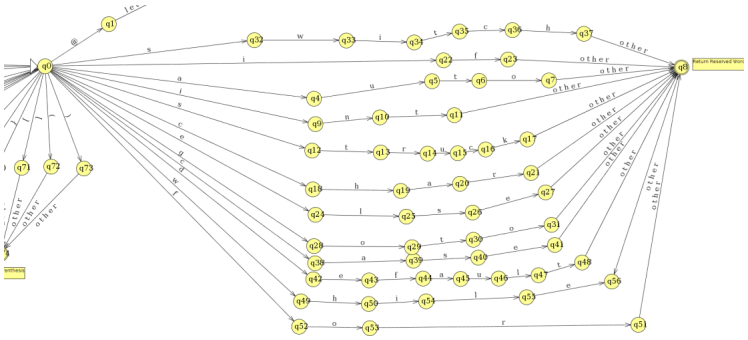

Figure 3: dfa for reserved words

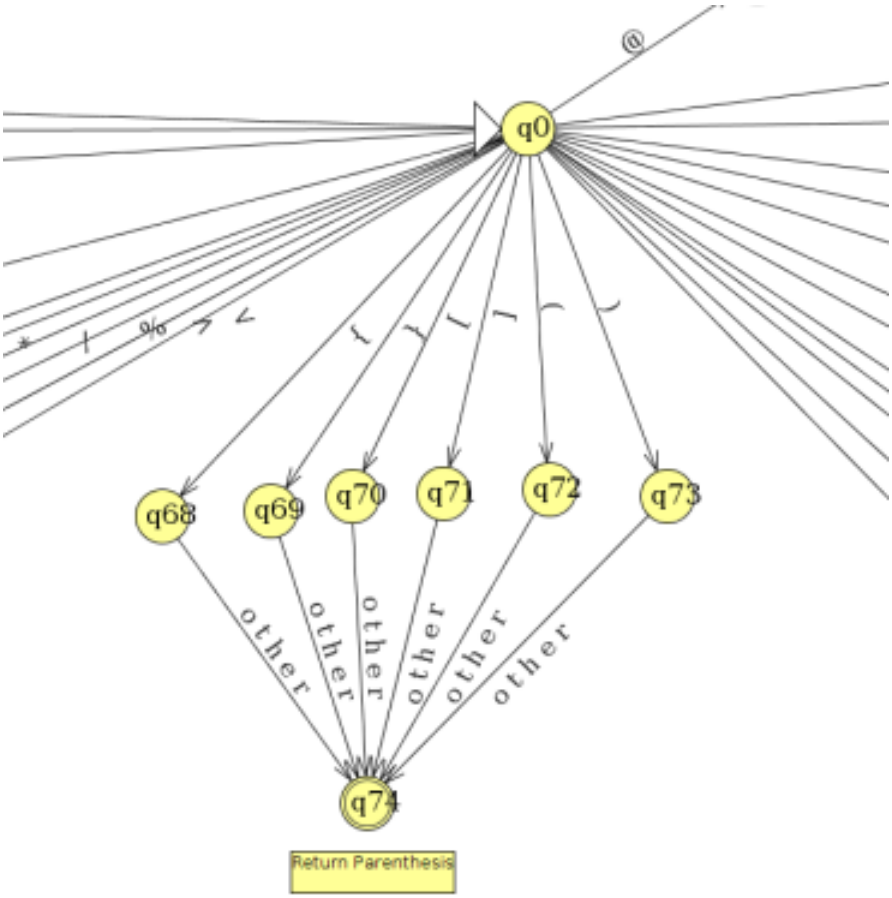

Figure 4: dfa for parathesis

3.1 Testing the code

3.1.1 Inputfile Given to the Code shown in figure 5

test.txt

```
1  int main() {
2      @name = "jawad"#
3      print("My Name is", @name)#
4      if @name print(hello jadi)#
5      @a = 1 + 2#
6      return 0#
7  }
```

Figure 5: Input file

3.1.2 Output File Generated shown in Figure 6.

output.txt

```
1   int <int>
2   {   <{>
3   @name   <id, 1>
4   =   <=>
5   if  <if>
6   @name   <id, 2>
7   @a  <id, 3>
8   =   <=>
9   +   <+>
10  }   <}>
11
```

Figure 6: Output file