پروژه کامپایلر

اعضای گروه :علیرضا اکرمی ،امیرحسین محمدی

فاز اول

اهداف این فاز:در این فاز قرار است با استفاده از ابزار jflex به برای یک ورودی که حاوی الگوهای زبان هست برنامه ای بنویسیم تا توکن های مربوط به آن فایل را پیدا کرده و جدول نمادی برای آن ایجاد کند

کتابخانه های مورد استفاده در این برنامه

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Iterator;

%%
%class MyC
%standalone
%column
%line
```

عبارت های منظم مورد استفاده برای یافتن توکن

```
single_multiline_Comment = \/\*[^\*]*\*\/ | \/\/[^\n]*\n
OneChar = [\(\)\{\}\,\;\+\-\*\/\%\<\>\=\!\[\]]
TwoChar = \<\= | \>\= | \=\= | \!\= | \| \| | \&\&
KW = if | else | while | for | return | break
ArrayOp = new | size
DataType = void | bool | int | float
BoolValue = true | false
FloatNum = \d+\.\d+
IntegerNum = \d+
Id = [a-zA-Z\_][a-zA-Z\_0-9]*
WhiteSpace = [\t\n ]+
EndOfFile = \z
```

ذخیره سازی توکن های پیدا شده در ArrayList برای استفاده در مرحله بعد

```
%{
    HashMap<String, Integer> symbolTable = new HashMap<>();
    ArrayList Sym_table = new ArrayList();
    BufferedWriter writer,TokenOutput;
    // Iterator i = Sym_table.iterator();
    int flag,i,it;
    {
    try{
        writer = new BufferedWriter(new FileWriter("TokenResult.txt"));
        TokenOutput = new BufferedWriter(new FileWriter("SimbolTable.txt"));
    } catch(IOException e) {
        e.printStackTrace();
    }
    }
%}
```

کد مربوط به ایجاد جدول نماد

```
%eof{
    flag=0;
    i=0;
    it=1;
    try{
        TokenOutput.append(String.valueOf(it)+"\t");
        while(i<Sym_table.size()){
            TokenOutput.append(String.valueOf(Sym_table.get(i))+"\t ");

            TokenOutput.flush();
            flag++;
            if(flag==2){
                flag=0;
                it++;
                TokenOutput.newLine();
                TokenOutput.append(String.valueOf(it)+"\t");
                TokenOutput.flush();

            }
            i++;
        }
        writer.close();
        TokenOutput.close();

    } catch(IOException e){
        e.printStackTrace();
    }
%eof}
```

Macro definition

1

```
%%
{single_multiline_Comment} {
    // System.out.print(yytext());
}
```

2

```
{TwoChar} {
        String word = "Null";
        String input = yytext();
        switch (input){
            case "<=":
                word = "LE";
                Sym_table.add("LE");
                Sym_table.add("257");
                break;
            case ">=":
                word = "GE";
                Sym_table.add("GE");
                Sym_table.add("258");
                break;
            case "==":
                word = "EQ";
                Sym_table.add("EQ");
                Sym_table.add("259");
                break;
            case "!=":
                word = "NE";
                Sym_table.add("NE");
                Sym_table.add("260");
                break;
            case "||":
                word = "OR";
                Sym_table.add("OR");
                Sym_table.add("261");
                break;
            case "&&":
                word = "AND";
                Sym_table.add("AND");
                Sym_table.add("262");
                break;
        }
```

3

```
{OneChar} {
        Sym_table.add("SingleChar");
        Sym_table.add(Integer.valueOf((int) yytext().toCharArray()[0]));
        try {

            writer.append("<SingleChar, " + Integer.valueOf((int) yytext().toCharArray()[0]) + ">");
            writer.flush();


        } catch (IOException e){
            e.printStackTrace();
        }

}
```

4

```
{KW} {

        String w=yytext();
        switch (w) {
            case "if":
                Sym_table.add("IF");
                Sym_table.add("263");
                break;
            case "else":
                Sym_table.add("ELSE");
                Sym_table.add("264");
                break;
            case "while":
                Sym_table.add("WHILE");
                Sym_table.add("265");
                break;
            case "for":
                Sym_table.add("FOR");
                Sym_table.add("266");
                break;
            case "return":
                Sym_table.add("RETURN");
                Sym_table.add("267");
                break;
            case  "break":
                Sym_table.add("BREAK");
                Sym_table.add("268");
                break;
        }
        try{
                writer.append("<KW, " + yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
```

5

```
{ArrayOp} {
        String AO =yytext();
         switch (AO) {
            case "new":
                Sym_table.add("NEW");
                Sym_table.add("269");
                break;
            case "size":
                Sym_table.add("SIZE");
                Sym_table.add("270");
                break;
            }
        try{
                writer.append("<AO, " +
yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
```

6

```
{DataType} {
    String DT=yytext();
    switch (DT){
        case "void":
            Sym_table.add("VOID");
            Sym_table.add("271");
            break;
        case "bool":
            Sym_table.add("BOOL");
            Sym_table.add("272");
            break;
        case "int":
            Sym_table.add("INT");
            Sym_table.add("273");
            break;
        case "float":
            Sym_table.add("FLOAT");
            Sym_table.add("274");
            break;
    }
    try{
        writer.append("<DT, " + yytext().toUpperCase()+">");
        writer.flush();
    } catch (IOException e){
        e.printStackTrace();
    }
}
```

7

```
{BoolValue} {
        String BOOL_LIT=yytext();
        Sym_table.add("BOOL_LIT");
        switch (BOOL_LIT){
            case "true":
              Sym_table.add("275");
              break;
            case "false":
              Sym_table.add("276");
              break;
        }
        try{
                writer.append("<BOOL_LIT, " +
yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
```

8

```
{FloatNum} {
        Sym_table.add("Float_LIT");
        Sym_table.add("278");
        try{
                writer.append("<Float_LIT, " + yytext() + ">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
{IntegerNum} {
        Sym_table.add("INT_LIT");
        Sym_table.add("279");
        try{
                writer.append("<INT_LIT, " + yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
{Id} {
        Sym_table.add("IDENT");
        Sym_table.add("280");
        int index = symbolTable.size();
        if (symbolTable.containsKey(yytext())) {
```

```
 index = symbolTable.get(yytext());
        } else {
            symbolTable.put(yytext(), index);
        }
        try {

            writer.append("<IDENT, " + index + ">");
            writer.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
{WhiteSpace} {}
```

```
{EndOfFile} {
        Sym_table.add("EOF");
        Sym_table.add("0");
        try {
            writer.append("<EOF, " + 0 + ">");
            writer.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
```

نحوه اجرای برنامه

Command Prompt

```
D:\learning\compiler\old>jflex Lexical.jflex
Reading "Lexical.jflex"
Constructing NFA : 224 states in NFA
Converting NFA to DFA :
.....................................................................................

Warning in file "Lexical.jflex" (line 276):
Rule can never be matched:
{EndOfFile} {
86 states before minimization, 57 states in minimized DFA
Old file "MyC.java" saved as "MyC.java~"
Writing code to "MyC.java"

D:\learning\compiler\old>javac MyC.java
Note: MyC.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

D:\learning\compiler\old>java MyC input.txt

D:\learning\compiler\old>
```

ورودی برنامه

```
void main(void)
{
num = read();
/* To calculate anf print num^2*/
print(num* num);
}
int main ()
{
float cos, x, n, term, eps, alt;
// compute the cosine of x to within tolerance eps
// use an alternating series
x = 3.14159;
eps = 0.1;
n = 1;
cos = 1;
term = 1;
alt = -1;
while (term>eps)
{
term = term * x * x / n / (n+1);
cos = cos + alt * term;
alt = -alt;
n = n + 2;
}
}
```

خروجی های برنامه

فایل مربوط به توکن های یافت شده

<DT, VOID><IDENT, 0><SingleChar, 40><DT, VOID><SingleChar, 41><SingleChar, 123><IDENT, 1><SingleChar, 61><IDENT, 2><SingleChar, 40><SingleChar, 41><SingleChar, 59><IDENT, 3><SingleChar, 40><IDENT, 1><SingleChar, 42><IDENT, 1><SingleChar, 41><SingleChar, 59><SingleChar, 125><DT, INT><IDENT, 0><SingleChar, 40><SingleChar, 41><SingleChar, 123><DT, FLOAT><IDENT, 4><SingleChar, 44><IDENT, 5><SingleChar, 44><IDENT, 6><SingleChar, 44><IDENT, 7><SingleChar, 44><IDENT, 8><SingleChar, 44><IDENT, 9><SingleChar, 59><IDENT, 5><SingleChar, 61><Float_LIT, 3.14159><SingleChar, 59><IDENT, 8><SingleChar, 61><Float_LIT, 0.1><SingleChar, 59><IDENT, 6><SingleChar, 61><INT_LIT, 1><SingleChar, 59><IDENT, 4><SingleChar, 61><INT_LIT, 1><SingleChar, 59><IDENT, 7><SingleChar, 61><INT_LIT, 1><SingleChar, 59><IDENT, 9><SingleChar, 61><SingleChar, 45><INT_LIT, 1><SingleChar, 59><KW, WHILE><SingleChar, 40><IDENT, 7><SingleChar, 62><IDENT, 8><SingleChar, 41><SingleChar, 123><IDENT, 7><SingleChar, 61><IDENT, 7><SingleChar, 42><IDENT, 5><SingleChar, 42><IDENT, 5><SingleChar, 47><IDENT, 6><SingleChar, 47><SingleChar, 40><IDENT, 6><SingleChar, 43><INT_LIT, 1><SingleChar, 41><SingleChar, 59><IDENT, 4><SingleChar, 61><IDENT, 4><SingleChar, 43><IDENT, 9><SingleChar, 42><IDENT, 7><SingleChar, 59><IDENT, 9><SingleChar, 61><SingleChar, 45><IDENT, 9><SingleChar, 59><IDENT, 6><SingleChar, 61><IDENT, 6><SingleChar, 43><INT_LIT, 2><SingleChar, 59><SingleChar, 125><SingleChar, 125>

جدول نماد

1	VOID	271	
2	IDENT	280	
3	SingleChar		40
4	VOID	271	
5	SingleChar		41
6	SingleChar		123
7	IDENT	280	
8	SingleChar		61
9	IDENT	280	
10	SingleChar		40
11	SingleChar		41
12	SingleChar		59
13	IDENT	280	
14	SingleChar		40
15	IDENT	280	
16	SingleChar		42
17	IDENT	280	
18	SingleChar		41
19	SingleChar		59
20	SingleChar		125
21	INT	273	
22	IDENT	280	
23	SingleChar		40
24	SingleChar		41
25	SingleChar		123
26	FLOAT	274	
27	IDENT	280	
28	SingleChar		44
29	IDENT	280	
30	SingleChar		44
31	IDENT	280	
32	SingleChar		44
33	IDENT	280	
34	SingleChar		44
35	IDENT	280	
36	SingleChar		44

فاز دوم

طراحی تجزیه کننده

کتابخانه های مورد نیاز

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Iterator;

%%
%class MyC
%standalone
%column
%line
%byaccj
```

عبارت منظم برای تشخیص خطا

ErrNum = \d+\.+ | \d+\.\d+\.[\d\.]* | \d+\.\.[\d+\.]*

ErrIdent = [0-9]+[a-zA-Z][a-zA-Z_0-9]*

عبارات های منظم برای یافتن توکن

```
single_multiline_Comment = \/\*[^\*]*\*\/ | \/\/[^\n]*\n
OneChar = [\(\)\{\}\,\;\+\-\*\/\%\<\>\=\!\[\]]
TwoChar = \<\= | \>\= | \=\= | \!\= | \| \| | \&\&
KW = if | else | while | for | return | break
ArrayOp = new | size
DataType = void | bool | int | float
BoolValue = true | false
FloatNum = \d+\.\d+
IntegerNum = \d+
Id = [a-zA-Z\_][a-zA-Z\_0-9]*
WhiteSpace = [\t\n ]+
EndOfFile = \z
```

در این ابزار با استفاده از علامت % می توان به زبان جاوا کد زد

```
%{
  // Parser yyparser;
  private Parser yyparser = new Parser();

  public MyC(java.io.Reader r, Parser yyparser) {
    this(r);
    this.yyparser = yyparser;
  }

  HashMap<String, Integer> symbolTable = new HashMap<>();
  ArrayList Sym_table = new ArrayList();
  BufferedWriter writer,TokenOutput;
  // Iterator i = Sym_table.iterator();
  int flag,i,it;
  {
  try{
    writer = new BufferedWriter(new FileWriter("TokenResult.txt"));
    TokenOutput = new BufferedWriter(new FileWriter("SimbolTable.txt"));
     error = new BufferedWriter(new FileWriter("Error.txt"));
  } catch(IOException e) {
    e.printStackTrace();
  }
  }
%}
```

با استفاده از علامت eof می توان کد هایی نوشت که به هر حال در انتهای اجرای برنامه اجرا شود در اینجا نوشتن در فایل

```
%eof{
    flag=0;
    i=0;
    it=1;
    try{
        TokenOutput.append(String.valueOf(it)+"\t");
        while(i<Sym_table.size()){
            TokenOutput.append(String.valueOf(Sym_table.get(i))+"\t ");
            // System.out.println(String.valueOf(Sym_table.get(i)));
            TokenOutput.flush();
            flag++;
            if(flag==2){
                flag=0;
                it++;
                TokenOutput.newLine();
                TokenOutput.append(String.valueOf(it)+"\t");
                TokenOutput.flush();

            }
            i++;
        }
        writer.close();
        TokenOutput.close();

    } catch(IOException e){
        e.printStackTrace();
    }
%eof}
```

Action and rules

```
{single_multiline_Comment} {
    // System.out.print(yytext());
}
{TwoChar} {
        String word = "Null";
        String input = yytext();
        switch (input){
            case "<=":
                word = "LE";
                Sym_table.add("LE");
                Sym_table.add("257");
                return Parser.LE;
                break;
            case ">=":
                word = "GE";
                Sym_table.add("GE");
                Sym_table.add("258");
                return Parser.GE;
                break;
            case "==":
                word = "EQ";
                Sym_table.add("EQ");
                Sym_table.add("259");
                return Parser.EQ;
                break;
            case "!=":
                word = "NE";
                Sym_table.add("NE");
                Sym_table.add("260");
                return Parser.NE;
                break;
            case "||":
                word = "OR";
                Sym_table.add("OR");
                Sym_table.add("261");
                return Parser.OR;
                break;
            case "&&":
                word = "AND";
                Sym_table.add("AND");
                Sym_table.add("262");
                return Parser.AND;
                break;
        }

        try {

writer.append("<DoubleChar, " +
word+">");
            writer.flush();

        }catch (IOException e){
            e.printStackTrace();
        }

}
```

```
{OneChar} {
            Sym_table.add("SingleChar");
            Sym_table.add(Integer.valueOf((int) yytext().toCharArray()[0]));
           try {

               writer.append("<SingleChar, " + Integer.valueOf((int)
yytext().toCharArray()[0]) + ">");
               writer.flush();


           } catch (IOException e){
               e.printStackTrace();
           }
           return (int) yycharat(0);
}
```

```
{KW} {
        String w=yytext();
        switch (w) {
            case "if":
                Sym_table.add("IF");
                Sym_table.add("263");
                return Parser.IF;
                break;
            case "else":
                Sym_table.add("ELSE");
                Sym_table.add("264");
                return Parser.ELSE;
                break;
            case "while":
                Sym_table.add("WHILE");
                Sym_table.add("265");
                 return Parser.WHILE;
                break;
            case "for":
                Sym_table.add("FOR");
                Sym_table.add("266");
                 return Parser.FOR;
                break;
            case "return":
Sym_table.add("RETURN");
                Sym_table.add("267");
                 return Parser.RETURN;
                break;
            case  "break":
                Sym_table.add("BREAK");
                Sym_table.add("268");
                 return Parser.BREAK;
                break;
        }
        try{
                writer.append("<KW, " +
yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
```

```
{ArrayOp} {
        String AO =yytext();
         switch (AO) {
            case "new":
                Sym_table.add("NEW");
                Sym_table.add("269");
                 return Parser.NEW;
                break;
            case "size":
                Sym_table.add("SIZE");
                Sym_table.add("270");
                 return Parser.SIZE;
                break;
            }
        try{
                writer.append("<AO, " +
yytext().toUpperCase()+">");
                writer.flush();
        } catch (IOException e){
            e.printStackTrace();
        }
}
```

```
{ErrNum} {
    try{
        error.append("Lex error in line " + yyline + ". Wrong number : " + yytext());
        error.newLine();
        error.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
{ErrId} {
    try{
        error.append("Lex error in line " + yyline + ". Id starts with a num: " +
yytext());
        error.newLine();
        error.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
.   {return (int) yycharat(0);}
```

نحوه اجرای برنامه

Command Prompt

```
D:\learning\compiler\s\s2>jflex Lexical.flex
Reading "Lexical.flex"
Constructing NFA : 276 states in NFA
Converting NFA to DFA :
..........................................................................................

Warning in file "Lexical.flex" (line 355):
Rule can never be matched:
{EndOfFile} {
97 states before minimization, 62 states in minimized DFA
Old file "MyC.java" saved as "MyC.java~"
Writing code to "MyC.java"

D:\learning\compiler\s\s2>javac MyC.java

D:\learning\compiler\s\s2>yacc.exe -J MyC.y
yacc.exe: w - the symbol size is undefined
yacc.exe: 235 shift/reduce conflicts.

D:\learning\compiler\s\s2>java MyC  input.txt

D:\learning\compiler\s\s2>
```

ورودی برنامه

```
int main (void)
{
double a=555;
float 3cos cos;
x = 3;
eps = 0.1;  0.1.1
int cos:
n = 1;
cos = 1;
term = 1;
alter = -1;
while (term>eps) 0..1
{
term = term * x * x / n / (n+1);
cos = cos + alt * term;
alt = -alt;
n = n + 2;
}
}
```

خروجی برنامه

Lexical error in line 2. Identifier starts with a number: 3cos
Lexical error in line 4. Wrong number format: 0.1.1
Lexical error in line 10. Wrong number format: 0..1