Nery Oswaldo Jiménez Contreras Carné: 201700381
Organización de Lenguajes y Compiladores 1 Sección: C

MANUAL TÉCNICO EXREGAN

Software desarrollado como proyecto 1 del curso de compiladores 1.

Herramientas

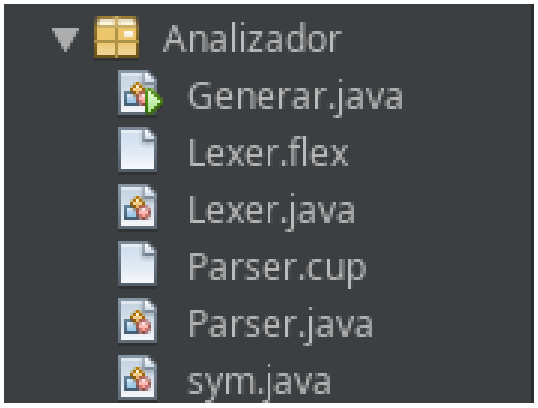
Java Cup versión 11
Java JFlex versión 1.7.0
Java JDK versión 18.0
Apache Netbeans versión 15

Sistema operativo: Manjaro Plasma

Analizador:

Para el desarrollo del analizador se crearon los archivos *Lexer.flex* y *Parser.cup* los cuales contienen la declaración del lenguaje a reconocer, así como las producciones de las gramáticas sintácticas por cumplir.

Dentro de este Package se encuentra una clase *Generar.java* la cual nos compila los dos archivos anteriormente mencionados, y así poder ejecutar nuestra aplicación.



Generar.java

```java
public class Generar {
    public static void main(String[] args) {
        try {
            String ruta = "src/Analizador/";
            String opcFlex[] = {ruta + "Lexer.flex", "-d", ruta};
            jflex.Main.generate(opcFlex);

            String opcCup[] = {"-destdir", ruta, "-parser", "Parser", ruta + "Parser.cup"};
            java_cup.Main.main(opcCup);
        } catch (Exception e){
        }
    }
}
```

En esta clase definimos la ruta donde se crearan nuestras clases *Lexer.java* y *Parser.java*.

Lexer.flex

Este archivo tiene de primero un conjunto de gramáticas para reconocer algunos patrones.

```
BLANCOS = [ \t\r\f\n]
ENTERO = [0-9]+
DECIMAL  = {ENTERO} ("."? [0-9]* )?
COMILLA = [\"]
LETRA = [a-zA-ZñÑ]+
ID = ({LETRA}|("_"{LETRA}))({LETRA}|{ENTERO}|"_")*
UNILINEA = ("//".*\r\n)|("//".*\n)|("//".*\r)
MULTILINEA = "<!""<"*([^*<]|[^*]"!"|"*"[^!])*"*"*"!>"
ASCII = [!-/]|[:-@]|[\[-`]|[{-~]

%%
```

Luego tenemos la declaración de símbolos:

```
":" { return new Symbol(sym.DOSPUNTOS, yyline, yychar, yytext());}
";" { return new Symbol(sym.PTCOMA, yyline, yychar, yytext());}
"{" { return new Symbol(sym.LLAVABRE, yyline, yychar, yytext());}
"}" { return new Symbol(sym.LLAVCIERRA, yyline, yychar, yytext());}

"+" {return new Symbol(sym.MAS, yyline, yychar, yytext());}
"*" {return new Symbol(sym.POR, yyline, yychar, yytext());}
"->" {return new Symbol(sym.FLECHA, yyline, yychar, yytext());}
"~" {return new Symbol(sym.COLOCHO, yyline, yychar, yytext());}
"." {return new Symbol(sym.PUNTO, yyline, yychar, yytext());}

"|" {return new Symbol(sym.OR, yyline, yychar, yytext());}
"%" {return new Symbol(sym.PORCENTAJE, yyline, yychar, yytext());}
"?" {return new Symbol(sym.INTERROG, yyline, yychar, yytext());}
"," {return new Symbol(sym.COMA, yyline, yychar, yytext());}
"\\n" {return new Symbol(sym.LN, yyline, yychar, yytext());}
"\\'" {return new Symbol(sym.COMSIMPLE, yyline, yychar, yytext());}
"\\\"" {return new Symbol(sym.COMDOBLE, yyline, yychar, yytext());}
```

Y también definimos para este proyecto un conjunto de palabras reservadas:

```
"CONJ" {return new Symbol(sym.PR_CONJ, yyline, yychar, yytext());}
{COMILLA} {return new Symbol(sym.COMILLA, yyline, yychar, yytext());}
{ID} {return new Symbol(sym.ID, yyline, yychar, yytext());}
{ENTERO} {return new Symbol(sym.ID, yyline, yychar, yytext());}
{DECIMAL} {return new Symbol(sym.ID, yyline, yychar, yytext());}
{ASCII} {return new Symbol(sym.ASCII, yyline, yychar, yytext());}
```

Parser.cup

En este archivo definimos nuestras reglas sintácticas, comenzando por declarar los terminales y no terminales necesarios para nuestra gramática:

```
terminal String DOSPUNTOS, PTCOMA,LLAVABRE,LLAVCIERRA;
terminal String MAS,POR, PORCENTAJE;
terminal String DECIMAL, ID, ASCII;
terminal String PR_CONJ, FLECHA, COLOCHO, PUNTO, OR, COMA, INTERROG, LN, COMSIMPLE;
terminal String COMDOBLE, COMILLA;

non terminal ini, cuerpo, conjunto, definicion, expresion, cuerpoExpresion, especiales, definido;
non terminal lexemas, usoConjuntos, data, simbolos, asciiSymb, asciiComplement;
```

Y una producción se ve de la siguiente manera:

```
start with ini;

ini ::= LLAVABRE:a cuerpo:b LLAVCIERRA:c;

cuerpo ::= conjunto:a
          |cuerpo:a conjunto:b
          | expresion:a
          | cuerpo:a expresion:b
          | lexemas:a
          | cuerpo:a lexemas:b
          | cuerpo error
;

conjunto ::= PR_CONJ:a DOSPUNTOS:b ID:c FLECHA:d definicion:e PTCOMA:f
           | error PTCOMA
;

definicion ::= ID:a COLOCHO:b ID:c
             | asciiSymb:a COLOCHO:b asciiSymb:c
             | definido:a
;

definido ::= ID:a
          | definido:a ID:b
          | definido:a COMA:b ID:c
          | asciiSymb:a
          | definido:a asciiComplement:b
          | definido:a COMA:b asciiComplement:c
          | especiales:a
          | definido:a COMA:b especiales:c
;
```

Dentro de cup podemos ingresar código java, el cuál hemos utilizado para generar mensajes de errores como el manejo de nodos para el método del árbol.

```
parser code
{:
    ArrayList<MetodoArbol> arboles = new ArrayList<>();

    //Lista para errores
    public ArrayList<Errors> Errores = new ArrayList();

    //Método para manejar algún error sintáctico
    public void syntax_error(Symbol s){
        Errores.add(new Errors("Sintáctico", "Error de sintaxis detectado. Símbolo: " + s.value, s.left + "", s.right + ""));
    }

    //Método errores sintácticos en el que ya no es posible recuperar errores
    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception{
        System.out.println("Error síntactico irrecuperable en la Línea " + (s.left)+ " Columna "+s.right+". Componente " + s.value + " no reconocido.");
    }

    public ArrayList<Errors> getErrores(){
        return Errores;
    }

:}
```

```
    cuerpoExpresion ::=  PUNTO:a cuerpoExpresion:b cuerpoExpresion:c {:
                            Nodo padre = new Nodo(a);
                            padre.setHijoIzq((Nodo)b);
                            padre.setHijoDer((Nodo)c);
                            RESULT = padre;
                        :}
                        | OR:a cuerpoExpresion:b cuerpoExpresion:c {:
                            Nodo padre = new Nodo(a);
                            padre.setHijoIzq((Nodo)b);
                            padre.setHijoDer((Nodo)c);
                            RESULT = padre;
                        :}
                        | POR:a cuerpoExpresion:b {:
                            Nodo padre = new Nodo(a);
                            padre.setHijoIzq((Nodo)b);
                            RESULT = padre;
                        :}
                        | MAS:a cuerpoExpresion:b {:
                            Nodo padre = new Nodo(a);
                            padre.setHijoIzq((Nodo)b);
                            RESULT = padre;
                        :}
                        | INTERROG:a cuerpoExpresion:b  {:
                            Nodo padre = new Nodo(a);
                            padre.setHijoIzq((Nodo)b);
                            RESULT = padre;
                        :}
                        | usoConjuntos:a {:
                            Nodo hoja = new Nodo((String)a);
                            hoja.setHoja(true);
                            hoja.setAnulable(false);
                            RESULT = hoja;
                        :}
                        | COMILLA:a data:b COMILLA:c{:
                            Nodo hoja = new Nodo((String)b);
                            hoja.setHoja(true);
                            hoja.setAnulable(false);
                            RESULT = hoja;
                        :}
    ;
```

Clase Nodo.java

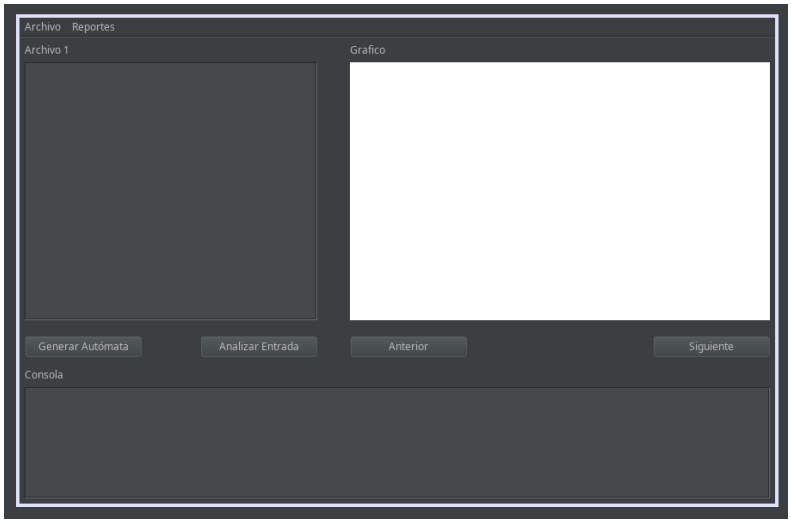
En esta clase tendremos los nodos y métodos necesarios para manejar el método del árbol.

```java
public class Nodo {
    public String token;
    public Nodo hijoIzq;
    public Nodo hijoDer;
    public boolean hoja = false;
    public int num;
    public boolean anulable;
    public ArrayList<Integer> primeros = new ArrayList<>();
    public ArrayList<Integer> ultimos = new ArrayList<>();

    public boolean isAnulable() {
        return anulable;
    }

    public void setAnulable(boolean anulable) {
        this.anulable = anulable;
    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }

    public Nodo(String token) {
        this.token = token;
    }

    public Nodo getHijoIzq() {
        return hijoIzq;
    }

    public void setHijoIzq(Nodo hijoIzq) {
        this.hijoIzq = hijoIzq;
    }
```

```java
public Nodo(String token) {
    this.token = token;
}

public Nodo getHijoIzq() {
    return hijoIzq;
}

public void setHijoIzq(Nodo hijoIzq) {
    this.hijoIzq = hijoIzq;
}

public Nodo getHijoDer() {
    return hijoDer;
}

public void setHijoDer(Nodo hijoDer) {
    this.hijoDer = hijoDer;
}

public void setHoja(boolean hoja) {
    this.hoja = hoja;
}

public boolean isHoja() {
    return hoja;
}

public String getToken() {
    return token;
}

public ArrayList<Integer> getPrimeros() {
    return primeros;
}

public ArrayList<Integer> getUltimos() {
    return ultimos;
}
```

interfazExRegan.java

Esta clase es la parte visual de la aplicación con la que el usuario estará interactuando. Entre sus métodos principales encontramos *OpenFile* para abrir archivos y cargarlos en la app, y *SaveAs* que permite guardar los textos que nosotros editemos dentro de la herramienta.




```
//Menu Abrir archivo
private void itemAbrirActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JFileChooser file = new JFileChooser();

    FileNameExtensionFilter filtro = new FileNameExtensionFilter( description: ".OLC", extensions: "olc");
    file.setFileFilter( filter: filtro);

    int seleccion = file.showOpenDialog( parent: this);

    if(seleccion == JFileChooser.APPROVE_OPTION){
        File fichero = file.getSelectedFile();

        try(FileReader fileR = new FileReader( file: fichero)){
            String cadena = "";
            int valor = fileR.read();
            while(valor != -1){
                cadena = cadena +(char) valor;
                valor = fileR.read();
            }
            this.txtArchivo.setText( t: cadena);
        }catch(IOException el){
        }
    }
}
//Menu Guardar Como
private void itemGuardarComoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JFileChooser file = new JFileChooser();
    int seleccion = file.showSaveDialog( parent: this);

    if(seleccion == JFileChooser.APPROVE_OPTION){
        File fichero = file.getSelectedFile();

        try(FileWriter fileW = new FileWriter( file: fichero)){
            fileW.write( str: this.txtArchivo.getText());
        } catch(IOException el){
        }
    }
}
```

MetodoArbol.java

En esta clase se realiza la mayor parte del proyecto, desde la generación del árbol, la tabla de siguientes y transiciones, y la generación de archivos *.dot para los reportes.

```
public class MetodoArbol {
    public Nodo arbolExpresiones;
    public int numNodo = 1;
    public int numDot = 1;
    //public ArrayList tablaSiguientes;
    ArrayList <ArrayList> tablaSiguientes = new ArrayList<>();

    public MetodoArbol(Nodo arbolExpresiones) {
        Nodo raiz = new Nodo( token: ".");
        Nodo acepta = new Nodo( token: "#");
        acepta.setHoja( hoja: true);
        acepta.setAnulable( anulable: false);
        raiz.setHijoDer( hijoDer: acepta);
        raiz.setHijoIzq( hijoIzq: arbolExpresiones);
        this.arbolExpresiones = raiz;
        asignarIndice( actual: this.arbolExpresiones);
        numNodo = 0;
//        ArrayList <ArrayList> tablaSiguientes = new ArrayList<>();
        anulables( actual: this.arbolExpresiones);
        //System.out.println("Tabla siguientes:"+tablaSiguientes);
        imprimirSiguientes( i: Integer.toString( i: numDot));
        //System.out.println(crearArbol(this.arbolExpresiones,numNodo));
        generarDot( dot: crearArbol( node: this.arbolExpresiones, padre: numNodo), i: Integer.toString( i: numDot));
        numDot++;
    }

    public void asignarIndice(Nodo actual){
        if(actual == null){
            return;
        }
        if(actual.isHoja()){
            actual.setNum( num: numNodo);
            numNodo++;
            return;
        }

        asignarIndice( actual: actual.getHijoIzq());
        asignarIndice( actual: actual.getHijoDer());

    }
```

```
public void anulables(Nodo actual){
    if(actual == null){
        return;
    }

    if(actual.isHoja()){
        actual.getPrimeros().add( e: actual.getNum());
        actual.getUltimos().add( e: actual.getNum());
        return;
    }

    anulables( actual: actual.getHijoIzq());
    anulables( actual: actual.getHijoDer());

    switch (actual.getToken()) {
        case ".":
            actual.setAnulable(actual.getHijoIzq().isAnulable() && actual.getHijoDer().isAnulable());
            if(actual.getHijoIzq().isAnulable()){
                actual.getPrimeros().addAll( c: actual.getHijoIzq().getPrimeros());
                actual.getPrimeros().addAll( c: actual.getHijoDer().getPrimeros());
            }else{
                actual.getPrimeros().addAll( c: actual.getHijoIzq().getPrimeros());
            }
            if(actual.getHijoDer().isAnulable()){
                actual.getUltimos().addAll( c: actual.getHijoIzq().getUltimos());
                actual.getUltimos().addAll( c: actual.getHijoDer().getUltimos());
            }else{
                actual.getUltimos().addAll( c: actual.getHijoDer().getUltimos()); //Validar que esté correcto
            }
```

En este apartado comienza la generación de tabla de transiciones

```
        //Creando tabla de siguientes
        ArrayList<Integer> follow = new ArrayList<>();
        follow = actual.getHijoIzq().getUltimos();
        ArrayList<Integer> listaFollow = new ArrayList<>();
        listaFollow = actual.getHijoDer().getPrimeros();
        String token = actual.getHijoIzq().getToken();
        boolean alerta = false;
        for(int item : follow){
            if(tablaSiguientes != null){
                for (ArrayList ts : tablaSiguientes) {
                    if(ts.get( index: 1).equals( obj: item)){
                        ArrayList<Integer> pivL = (ArrayList<Integer>) ts.get( index: 2);
                        for(int pivItem : listaFollow){
                            if(!pivL.contains( o: pivItem)){
                                pivL.add( e: pivItem);
                            }
                        }
                        ts.set( index: 2,  element: pivL);
                        alerta = true;
                    }
                }
            }
            if(alerta != true){
                ArrayList tabla = new ArrayList();
                tabla.add( e: token);
                tabla.add( e: item);
                tabla.add( e: listaFollow);
                tablaSiguientes.add( e: tabla);
            }
            alerta = false;
        }
        if("#".equals( anObject: actual.getHijoDer().getToken())){
            //Creando tabla de siguientes
            ArrayList tabla = new ArrayList();
            tabla.add( e: "#");
            tabla.add( e: actual.getHijoDer().getUltimos());
            tabla.add( e: "--");
            tablaSiguientes.add( e: tabla);
        }
        break;
```

Así como la generación del documento DOT.

```
public void imprimirSiguientes(String i){

    //Ordenar ascendente
    //Collections.sort(tablaSiguientes, (ArrayList a, ArrayList b) -> ((Integer) a.get(1)).compareTo((Integer) b.get(1)));

    System.out.println( x: tablaSiguientes);
    String txt = "";
    txt += "digraph { \n tbl1 [\n"+
           "shape=plaintext\n" +
           " label=<\n" +
           "<table border='0' cellborder='1' cellspacing='0'>\n"+
           "<tr><td colspan=\"2\">TABLA DE SIGUIENTES</td></tr>\n" +
           "<th><td>Hoja</td><td>Siguientes</td></th>"+
           "";
    for(ArrayList ts : tablaSiguientes){
        txt += "<tr><td bgcolor=\"#fcc8c8\">"+ts.get( index: 1)+"</td><td bgcolor=\"#fcc8c8\">"+ts.get( index: 2)+"</td></tr>\n";
        //System.out.println(tablaSiguientes.get(i));
    }

    txt += "</table>\n" +
           ">];\n" +
           "}";

    //Generando DOT
    FileWriter fichero = null;
    PrintWriter escritor = null;
    String path = "/home/n21/Documentos/USAC/OLC1/LAB/PROYECTOS/PROYECTO1/repo/OLC1-Project1/SIGUIENTES_201700381/";
    String name = "";
    String nameJPG = "";
    try{

        name +="Siguientes"+i+".dot";
        nameJPG ="Siguientes"+i+".jpg";
        path += name;
        fichero = new FileWriter( fileName: path);
        escritor = new PrintWriter( out: fichero);
        escritor.println( x: txt);
        escritor.close();
        fichero.close();
    } catch(Exception e){
        System.out.println( x: "Error al generar .dot");
    }
```

Repositorio Github

El código de este proyecto podrá ser visualizado en el siguiente repositorio:
https://github.com/NeryJim21/OLC1-Project1.git