


| | | |
|---|--|--|
|  | TRABAJO PRÁCTICO Diseño de Compiladores | TP Final Versión: 1.5 |
|---|--|--|

Carrera: Licenciatura en Sistemas
Materia: Diseño de Compiladores
Fecha requerida: 15/11/2018

Año de Cursada: 2018 **Turno: Noche**
Docente: Ing. Pablo Pandolfo
Fecha entregada: 10/11/2018

Integrantes:

| Matrícula | APELLIDO, Nombres | Correo Electrónico |
|------------------|--------------------------|--|
| 501-11188 | García Huidobro, Germán | german.garcia@comunidad.edu.ub.ar |
| 501-11209 | Secatore, Miguel Angel | miguelangel.secare@comunidad.ub.edu.ar |
| | | |

Grilla de calificación

| Indicador | 1 | 2 | 3 | 4 | 5 |
|-------------------|----------|----------|----------|----------|----------|
| Muy Bien | | | | | |
| Bien | | | | | |
| A corregir | | | | | |
| NOTA | | | | | |

Indicadores de Contenido:

- Competencia técnica: incluye todos los materiales técnicos necesarios, incorpora correctamente la teoría aprendida
Comentario:.....
- Compleitud: grado de completitud técnica del producto entregado
Comentario:.....

Indicadores de Presentación:

- Claridad y Estructura: Trabajo escrito en forma clara y sucinta. Gramática, puntuación y variedad de vocabulario.
Comentario:.....
- Formato: Legibilidad, apariencia general, claridad de presentación
Comentario:.....
- Compleitud de presentación: incluye todos los materiales de presentación necesarios: carátula estándar, objetivo del TP, índice, cuerpo, referencias, etc (Materiales de presentación a acordar con el docente)
Comentario:.....

Comentario adicional del Estudiante:

Comentario adicional del Profesor:

Firma del profesor que corrige el TP:

Enunciado:

Lenguaje de expresiones: Little Quilt (“colchita de retazos”).

- Los constructores del lenguaje son expresiones que denotan objetos geométricos llamados “retazos” con una altura, un ancho y un patrón dibujado en ellos.
 - Reglas:
 1. Un retazo es una de las piezas primitivas, o
 2. se forma girando 90° un retazo hacia la derecha, o
 3. se forma cosiendo un retazo a la derecha de otro de igual altura.
 4. Ninguna otra cosa es un retazo.
- Entrada de estudio: coser(girar(r1), r2)

1. Análisis Lexicográfico:

- 1.1. Identificar las palabras reservadas.
- 1.2. Construir diagrama de transiciones para reconocer los componentes léxicos del lenguaje.
- 1.3. Generar la tabla de transiciones.
- 1.4. Implementar la tabla en Java.
- 1.5. Declarar directivas jlex (patrones – expresiones regulares).
- 1.6. Mostrar código generado por la herramienta.
- 1.7. Probar la ejecución de la entrada de estudio.

2. Análisis Sintáctico:

- 2.1. Diseñar una gramática no ambigua del lenguaje.
- 2.2. Generar el árbol de análisis sintáctico de la entrada de estudio.
- 2.3. Crear los diagramas de sintaxis.
- 2.4. Implementar la gramática en Java.
- 2.5. Diseñar Analizador Sintáctico Descendente con retroceso, construir la tabla de análisis sintáctico y analizar la entrada de estudio.
- 2.6. Comprobar si es LL(1) (si no lo es, hacer las modificaciones pertinentes para convertirla en LL(1)).
- 2.7. Construir su tabla de análisis y verificar si la entrada de estudio es analizada correctamente.
- 2.8. Diseñar Analizador Sintáctico Ascendente con retroceso, construir la tabla de análisis sintáctico y analizar la entrada de estudio.
- 2.9. Construir el autómata y la tabla de análisis SLR.
- 2.10. Analizar la entrada de estudio.
- 2.11. Construir la Tabla de Tipos.
- 2.12. Construir la Tabla de Símbolos.
- 2.13. Declarar directivas cup (gramática).
- 2.14. Mostrar código generado por la herramienta.
- 2.15. Probar la ejecución de la entrada de estudio.
- 2.16. Obtener conclusiones.

Actividad:

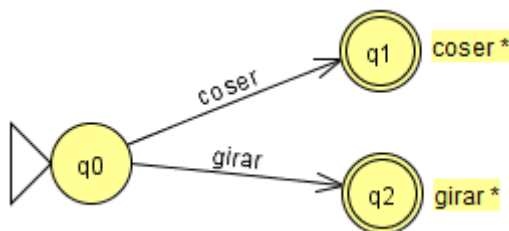
En base a la presentación del Trabajo Práctico en el classroom de la cátedra, se responderá los siguientes puntos dentro del desarrollo del trabajo.

Desarrollo:

1. Análisis Lexicográfico

1.1. Las palabras reservadas son:
coser y girar

1.2. Diagrama de transiciones:



1.3. Tabla de transiciones

| Estado | coser | girar |
|--------|-------|-------|
| >q0 | q1 | q2 |
| *q1 | - | - |
| *q2 | - | - |

1.4. Implementación de la tabla en Java

```

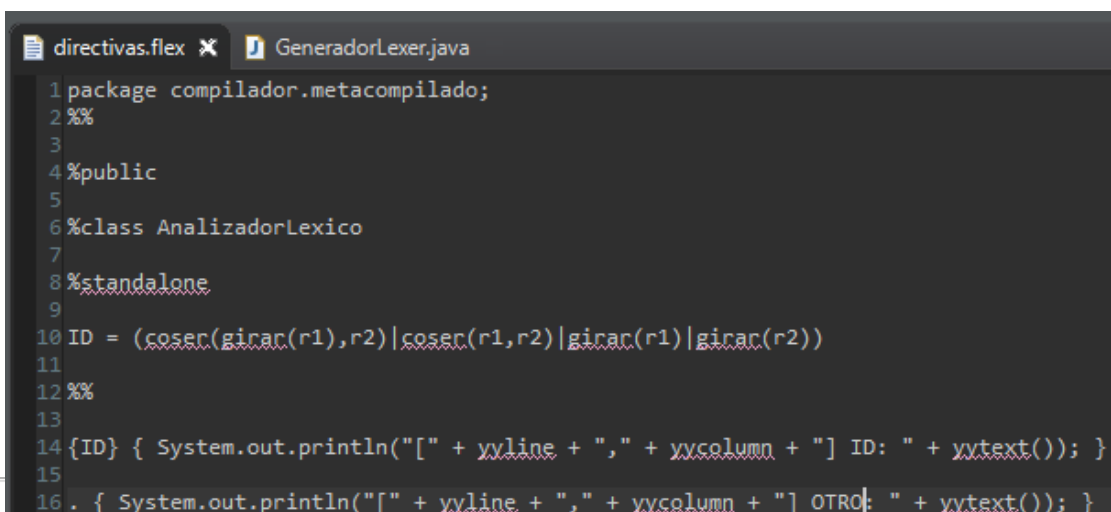
public class Lexer {
    private String componente = "";
    public static void main(String[] args) {
        System.out.println(new Lexer().q0());
    }
    private String q0() {
        switch (componente) {
            case "coser":
                return q1();
            case "girar":
                return q2();
        }
    }
}
  
```

```
        default:
            return "Error";
        }
    }
    private String q1() {
        switch (componente) {
            case "coser":
                return "Aceptar";
            default:
                return "Error";
        }
    }
    private String q2() {
        switch (componente) {
            case "girar":
                return "Aceptar";
            default:
                return "Error";
        }
    }
}
```

1.5. Declarar directivas jlex (patrones – expresiones regulares)

Ver Anexo 1

1.6. Mostrar código generado por la herramienta



```
1 package compilador.metacompilado;
2 %%
3
4 %public
5
6 %class AnalizadorLexico
7
8 %standalone
9
10 ID = (coser(girar(r1),r2)|coser(r1,r2)|girar(r1)|girar(r2))
11
12 %%
13
14 {ID} { System.out.println "[" + yyline + "," + yycolumn + "] ID: " + yytext(); }
15
16 . { System.out.println "[" + yyline + "," + yycolumn + "] OTRO: " + yytext(); }
```

El código Generado se encuentra en el Anexo 1

```

Problems Javadoc Declaration Console X
<terminated> GeneradorLexer (1) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15 nov. 2018 20:49:53)
Reading "src\compilador\metacompilado\directivas.flex"
Constructing NFA : 30 states in NFA
Converting NFA to DFA :
.....
20 states before minimization, 19 states in minimized DFA
Writing code to "src\compilador\metacompilado\AnalizadorLexico.java"

```

1.7. Probar la ejecución de la entrada de estudio.

Ver anexo 1

2. Análisis Sintáctico

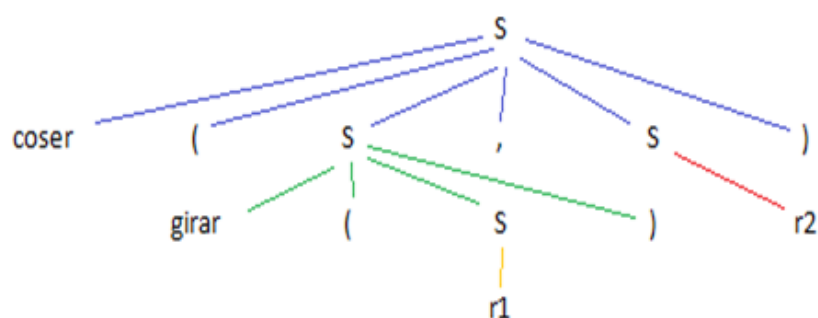
2.1. Diseñar una gramática no ambigua del lenguaje.

Es una gramática de tipo 2, libre de contexto

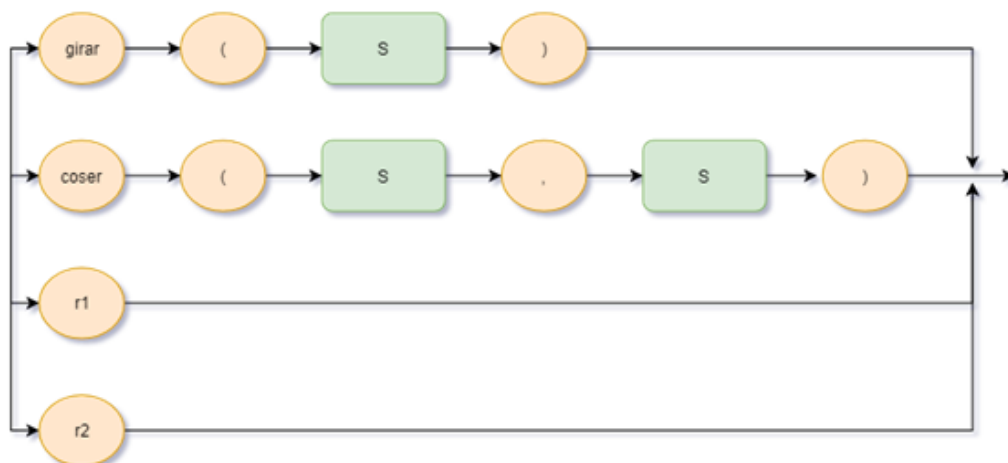
$S \rightarrow \text{coser}(S,S)|\text{girar}(S)|r1|r2$

$G_{t2} = \langle \{S, \text{coser}, \text{girar}\}, \{r1, r2\}, \{S\}, \{P_0 = \text{coser}(S,S), P_1 = \text{girar}(S,S), P_2 = r1, P_3 = r2\} \rangle$

2.2. Generar el árbol de análisis sintáctico de la entrada de estudio.



2.3. Crear los diagramas de sintaxis.



2.4. Implementar la gramática en Java.

```
private Boolean estado_q0() {
    String simbolo = obtenerSimbolo();
    switch(simbolo) {
        case "girar":
            return estado_q1();
        case "coser":
            return estado_q1();
        case "r1":
            return estado_q1();
        case "r2":
            return estado_q1();
        case "(":
            return estado_q1();
        case ")":
            return estado_q1();
        case ",":
            return estado_q1();
        default:
            return Boolean.FALSE;
    }
}

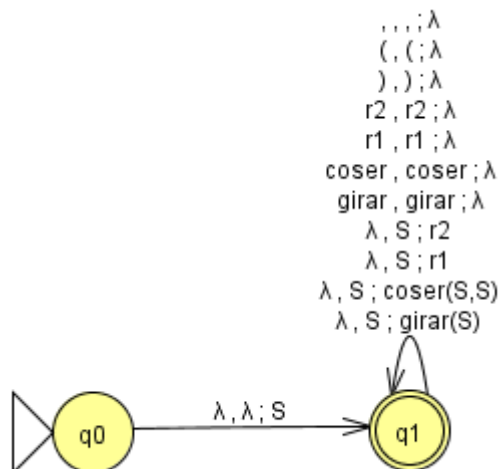
private Boolean estado_q1() {
    return Boolean.TRUE;
}
```

2.5. Diseñar Analizador Sintáctico Descendente con retroceso, construir la tabla de análisis sintáctico y analizar la entrada de estudio.

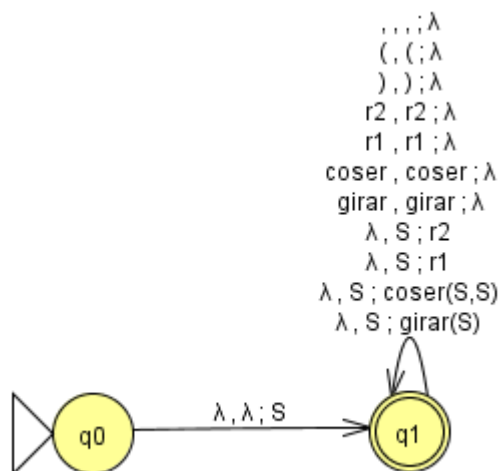
| Autómata de Pila | | |
|------------------|-------|-------|
| Leer | Sacar | Poner |

| | | |
|---------------------|-----------|----------------------|
| coser(girar(r1),r2) | λ | S |
| coser(girar(r1),r2) | λ | coser(S,S) |
| coser(girar(r1),r2) | coser | (S,S) |
| (girar(r1),r2) | (| S,S) |
| girar(r1),r2) | λ | girar(S),S) |
| girar(r1),r2) | girar | (S),S) |
| (r1),r2) | (| S),S) |
| r1),r2) | λ | r1),S) |
| r1),r2) | r1 |),S) |
|),r2) |) | ,S) |
| ,r2) | , | S) |
| r2) | λ | r2) |
| r2) | r2 |) |
|) |) | λ (ACEPTADO) |

- 2.6. Comprobar si es LL(1) (si no lo es, hacer las modificaciones pertinentes para convertirla en LL(1)).



- 2.7. Construir su tabla de análisis y verificar si la entrada de estudio es analizada correctamente.



$S \rightarrow coser(S,S) | girar(S) | r1 | r2$

$Pred(S \rightarrow coser(S,S)) \cap Pred(S \rightarrow girar(S)) \cap Pred(S \rightarrow r1) \cap Pred(S \rightarrow r2)$

$Prim(S \rightarrow coser(S,S)) \cup Prim(S \rightarrow girar(S)) \cup Prim(S \rightarrow r1) \cup Prim(S \rightarrow r2)$

$Prim(S) = \{coser\} \cup \{girar\} \cup \{r1\} \cup \{r2\}$

$Sig(S) = \{\$, \{, \} \} = \{\$, ,, \}$

$\{coser\} \cap \{girar\} \cap \{r1\} \cap \{r2\}$

| \ | coser | girar | R1 | R2 | (|) | , | \$ |
|---|----------------------------|--------------------------|--------------------|--------------------|-------|-------|-------|-------|
| S | $S \rightarrow coser(S,S)$ | $S \rightarrow girar(S)$ | $S \rightarrow r1$ | $S \rightarrow r2$ | error | Error | Error | error |

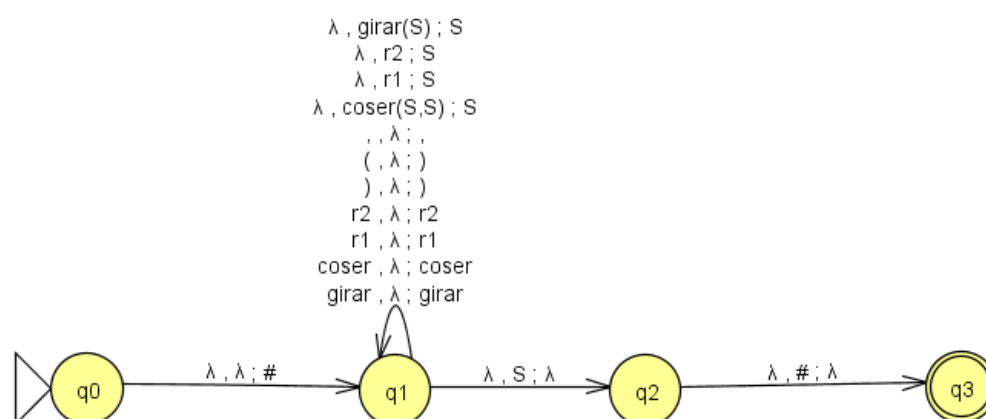
| Pila | Entrada | Regla o Acción |
|---------------|-------------------------------|-----------------------------------|
| \$S | <u>c</u> oser(girar(r1),r2)\$ | $S \rightarrow \text{coser}(S,S)$ |
| \$)S,S(coser | <u>c</u> oser(girar(r1),r2)\$ | Emparejar(coser) |
| \$)S,S(| (<u>g</u> irar(r1),r2)\$ | Emparejar((|
| \$)S,S | <u>g</u> irar(r1),r2)\$ | $S \rightarrow \text{girar}(S)$ |
| \$)S,)S(girar | <u>g</u> irar(r1),r2)\$ | Emparejar(girar) |
| \$)S,)S(| (<u>r</u> 1),r2)\$ | Emparejar((|
| \$)S,)S | <u>r</u> 1),r2)\$ | $S \rightarrow r1$ |
| \$)S,)r1 | <u>r</u> 1),r2)\$ | Emparejar(r1) |
| \$)S,) |),r2)\$ | Emparejar(()) |
| \$)S, | ,r2)\$ | Emparejar(,) |
| \$)S | <u>r</u> 2)\$ | $S \rightarrow r2$ |
| \$)r2 | <u>r</u> 2)\$ | Emparejar(r2) |
| \$) |)\$ | Emparejar()) |
| \$ | \$ | ACEPTAR |

2.8. Diseñar Analizador Sintáctico Ascendente con retroceso SLR, construir la tabla de análisis sintáctico y analizar la entrada de estudio.

| Autómata de Pila SLR | | |
|----------------------|-----------------|------------------------------------|
| Falta Leer | PILA | Acción |
| coser(girar(r1),r2) | λ | (q0, λ , λ)(q1,#) |
| coser(girar(r1),r2) | # | DESP |
| (girar(r1),r2) | #coser | DESP |
| girar(r1),r2) | #coser(| DESP |
| (r1),r2) | #coser(girar | DESP |
| r1),r2) | #coser(girar(| DESP |
|),r2) | #coser(girar(r1 | RED |
|),r2) | #coser(girar(S | DESP |
| ,r2) | #coser(girar(S) | RED |
| ,r2) | #coser(S | DESP |
| r2) | #coser(S, | DESP |
|) | #coser(S,r2 | RED |
|) | #coser(S,S | DESP |
| λ | #coser(S,S) | RED |

| | | |
|-----------|-----------|----------------------------------|
| λ | #S | $(q1, \lambda, S)(q2, \lambda)$ |
| λ | # | $(q2, \lambda, \#)(q3, \lambda)$ |
| λ | λ | FIN |

2.9. Construir el autómata y la tabla de análisis SLR.




2.10. Analizar la entrada de estudio.

2.11. Construir la Tabla de Tipos.

| Código | Nombre | TipoBase | Padre | Dimensión | Mínimo | Máximo | Ámbito |
|--------|--------|----------|-------|-----------|--------|--------|--------|
| 0 | retazo | -1 | -1 | 1 | -1 | -1 | 0 |

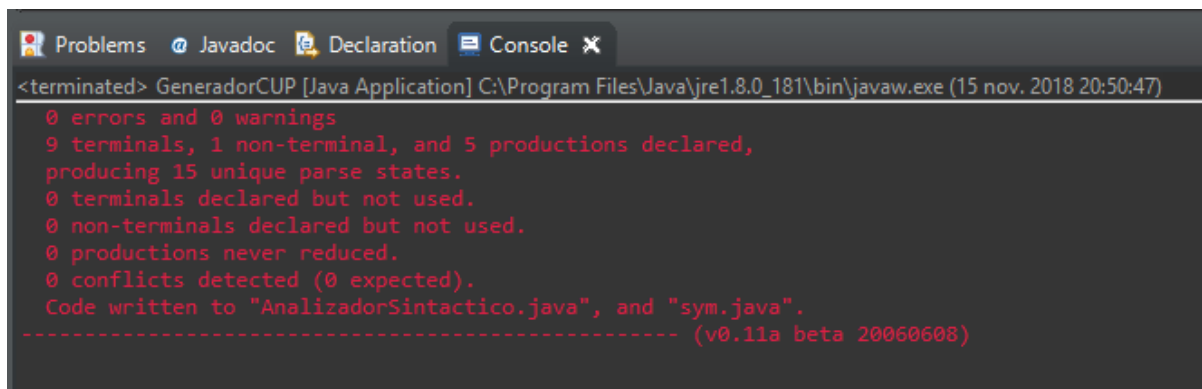
2.12. Construir la Tabla de Símbolos.

| Código | Nombre | Categoría | Tipo | NumPar | ListaPar | Dirección | Ámbito |
|--------|--------|-----------|------|--------|----------|-----------|--------|
| 0 | r1 | variable | 1 | -1 | null | null | 0 |
| 1 | r2 | variable | 1 | -1 | null | null | 0 |

| | | |
|---|---|---|
|  | <p align="center">TRABAJO PRÁCTICO</p> <p align="center">Diseño de Compiladores</p> | <p align="center">TP Final</p> <p align="center">Versión: 1.5</p> |
|---|---|---|

| | | | | | | | |
|---|-------|---------|----|----|------|------|---|
| 2 | coser | funcion | -1 | -1 | null | null | 0 |
| 3 | girar | funcion | -1 | -1 | null | nul | 0 |

- 2.13. Declarar directivas cup (gramática).
Ver anexo 1



```

<terminated> GeneradorCUP [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (15 nov. 2018 20:50:47)
0 errors and 0 warnings
9 terminals, 1 non-terminal, and 5 productions declared,
producing 15 unique parse states.
0 terminals declared but not used.
0 non-terminals declared but not used.
0 productions never reduced.
0 conflicts detected (0 expected).
Code written to "AnalizadorSintactico.java", and "sym.java".
----- (v0.11a beta 20060608)

```

- 2.14. Mostrar código generado por la herramienta.

- 2.15. Probar la ejecución de la entrada de estudio.
Ver anexo 1


- 2.16. Obtener conclusiones.

En este trabajo práctico y en el transcurso del desarrollo de la cátedra hemos aprendido y desarrollado las habilidades para interpretar de manera fehaciente el comportamiento de un compilador y todas sus etapas.

Observamos en detalle la etapa de análisis léxico gráfico, simulando un lenguaje denominado “colchita”.

Luego también vimos distintas estrategias dentro del análisis sintáctico que nos permitió acercarnos a la magnitud de complejidad que deben sortear los ingenieros que han desarrollado compiladores.⁽¹⁾


En términos generales, en informática, un compilador es un programa informático que transforma código fuente escrito en un lenguaje de programación o lenguaje informático (el lenguaje fuente), en otro lenguaje

| | | |
|---|---|---|
|  | <p style="text-align: center;">TRABAJO PRÁCTICO</p> <p style="text-align: center;">Diseño de Compiladores</p> | <p style="text-align: center;">TP Final</p> <p style="text-align: center;">Versión: 1.5</p> |
|---|---|---|

informático (el lenguaje objetivo, estando a menudo en formato binario conocido como código objeto). La razón más común para querer transformar código fuente es crear un programa ejecutable.

Cualquier programa escrito en un lenguaje de programación de alto nivel debe ser traducido a código objeto antes de que pueda ser ejecutado, para que todos los programadores que usen tal lenguaje usen un compilador o un intérprete. Por esto, los compiladores son muy importantes para los programadores. Cualquier mejora hecha a un compilador lleva a un gran número de programas mejorados.

Los compiladores son programas grandes y complejos, pero el análisis sistemático y la investigación de los científicos informáticos ha llevado a un entendimiento más claro de la construcción de los compiladores y una gran cantidad de teoría ha sido desarrollada sobre ellos. La investigación en la construcción de compiladores ha conducido a herramientas que hacen mucho más fácil crear compiladores, de modo que los estudiantes de informática de hoy en día pueden crear sus propios lenguajes pequeños y desarrollar un compilador simple en pocas semanas. (2)

| | | |
|--|--|-------------------------------------|
|  <p>UNIVERSIDAD DE Belgrano BUENOS AIRES - ARGENTINA</p> | <p>TRABAJO PRÁCTICO</p> <p>Diseño de Compiladores</p> | <p>TP Final</p> <p>Versión: 1.5</p> |
|--|--|-------------------------------------|

BIBLIOGRAFÍA

- ❖ Material de la clase para el desarrollo de los ejercicios.
Cátedra Diseño de Compiladores,
Ing. Pablo Pandolfo
- ❖ (1) Conclusión de los alumnos.
- ❖ (2) Historia de los Compiladores - Wikipedia
https://es.wikipedia.org/wiki/Historia_de_la_construcci%C3%B3n_de_los_compiladores

ANEXO 1

JFLEX (Analizador Léxico Gráfico)

AnalizadorLexico.java

```
package compilador.metacompilado;
import java_cup.runtime.Symbol;
%%

%public

%class AnalizadorLexico

%standalone
%cup
%%
"girar" {return new Symbol(sym.girar);}
"coser" {return new Symbol(sym.coser);}
"r1" {return new Symbol(sym.r1);}
"r2" {return new Symbol(sym.r2);}
"(" {return new Symbol(sym.para);}
")" {return new Symbol(sym.parc);}
"," {return new Symbol(sym.coma);}
. {System.err.println("Caracter Invalido");}
```

GeneradorCUP.java

```
package compilador.metacompilado;

import java.io.File;

public class GeneradorCUP {

    public static void main(String[] args) {

        String opciones[] = new String[] {"-destdir" , "src" + File.separator + "compilador" + File.separator +
"metacompilado", "-parser", "AnalizadorSintactico", "src" + File.separator + "compilador" + File.separator + "metacompilado" +
File.separator + "Sintactico.cup"};

        try { //
            java_cup.Main.main(opciones);
        }
        catch (Exception e) { System.out.print(e);}
    }
}
```

CUP (Analizador Sintáctico)

```
package compilador.metacompilado;
import java_cup.runtime.*;

terminal girar, coser, r1, r2, para, parc, coma;

non terminal S;
```



```
start with S;  
S ::= r1;  
S ::= r2;  
S ::= girar para S parc;  
S ::= coser para S coma S parc;
```

Generador CUP (Analizador Sintáctico)

```
package compilador.metacompilado;  
  
import java.io.File;  
  
public class GeneradorCUP {  
  
    public static void main(String[] args) {  
  
        String opciones[] = new String[] {"-destdir" , "src" + File.separator + "compilador" + File.separator +  
"metacompilado", "-parser", "AnalizadorSintactico", "src" + File.separator + "compilador" + File.separator + "metacompilado" +  
File.separator + "Sintactico.cup"};  
        try { //  
            java_cup.Main.main(opciones);  
        }  
        catch (Exception e) { System.out.print(e);}  
    }  
}
```

Analizador Sintáctico

```
//-----  
// The following code was generated by CUP v0.11a beta 20060608  
// Thu Nov 15 20:36:32 ART 2018  
//-----  
  
package compilador.metacompilado;  
  
import java_cup.runtime.*;  
  
/** CUP v0.11a beta 20060608 generated parser.  
 * @version Thu Nov 15 20:36:32 ART 2018  
 */  
public class AnalizadorSintactico extends java_cup.runtime.lr_parser {  
  
    /** Default constructor. */  
    public AnalizadorSintactico() {super();}  
  
    /** Constructor which sets the default scanner. */  
    public AnalizadorSintactico(java_cup.runtime.Scanner s) {super(s);}  
  
    /** Constructor which sets the default scanner. */  
    public AnalizadorSintactico(java_cup.runtime.Scanner s, java_cup.runtime.SymbolFactory sf) {super(s,sf);}  
  
    /** Production table. */  
    protected static final short _production_table[][] =
```

```
unpackFromStrings(new String[] {
    "\000\005\000\002\002\004\000\002\002\003\000\002\002" +
    "\003\000\002\002\006\000\002\002\010" });

/** Access to production table. */
public short[][] production_table() {return _production_table;}

/** Parse-action table. */
protected static final short[][] _action_table =
    unpackFromStrings(new String[] {
        "\000\017\000\012\004\006\005\010\006\005\007\004\001" +
        "\002\000\010\002\uffff\011\uffff\012\uffff\001\002\000\010" +
        "\002\000\011\000\012\000\001\002\000\004\010\017\001" +
        "\002\000\004\002\016\001\002\000\004\010\011\001\002" +
        "\000\012\004\006\005\010\006\005\007\004\001\002\000" +
        "\004\012\013\001\002\000\012\004\006\005\010\006\005" +
        "\007\004\001\002\000\004\011\015\001\002\000\010\002" +
        "\ufffd\011\ufffd\012\ufffd\001\002\000\004\002\001\001\002" +
        "\000\012\004\006\005\010\006\005\007\004\001\002\000" +
        "\004\011\021\001\002\000\010\002\ufffe\011\ufffe\012\ufffe" +
        "\001\002" });

/** Access to parse-action table. */
public short[][] action_table() {return _action_table;}

/** <code>reduce_goto</code> table. */
protected static final short[][] _reduce_table =
    unpackFromStrings(new String[] {
        "\000\017\000\004\002\006\001\001\000\002\001\001\000" +
        "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
        "\001\001\000\004\002\011\001\001\000\002\001\001\000" +
        "\004\002\013\001\001\000\002\001\001\000\002\001\001" +
        "\000\002\001\001\000\004\002\017\001\001\000\002\001" +
        "\001\000\002\001\001" });

/** Access to <code>reduce_goto</code> table. */
public short[][] reduce_table() {return _reduce_table;}

/** Instance of action encapsulation class. */
protected CUP$AnalizadorSintactico$actions action_obj;

/** Action encapsulation object initializer. */
protected void init_actions()
{
    action_obj = new CUP$AnalizadorSintactico$actions(this);
}

/** Invoke a user supplied parse action. */
public java_cup.runtime.Symbol do_action(
    int          act_num,
    java_cup.runtime.lr_parser parser,
    java.util.Stack stack,
    int          top)
    throws java.lang.Exception
{
    /* call code in generated class */
    return action_obj.CUP$AnalizadorSintactico$do_action(act_num, parser, stack, top);
}
```

```

}

/** Indicates start state. */
public int start_state() {return 0;}
/** Indicates start production. */
public int start_production() {return 0;}

/** <code>EOF</code> Symbol index. */
public int EOF_sym() {return 0;}

/** <code>error</code> Symbol index. */
public int error_sym() {return 1;}

}

/** Cup generated class to encapsulate user supplied action code.*/
class CUP$AnalizadorSintactico$actions {
    private final AnalizadorSintactico parser;

    /** Constructor */
    CUP$AnalizadorSintactico$actions(AnalizadorSintactico parser) {
        this.parser = parser;
    }

    /** Method with the actual generated action code. */
    public final java_cup.runtime.Symbol CUP$AnalizadorSintactico$do_action(
        int CUP$AnalizadorSintactico$act_num,
        java_cup.runtime.lr_parser CUP$AnalizadorSintactico$parser,
        java.util.Stack CUP$AnalizadorSintactico$stack,
        int CUP$AnalizadorSintactico$top)
        throws java.lang.Exception
    {
        /* Symbol object for return from actions */
        java_cup.runtime.Symbol CUP$AnalizadorSintactico$result;

        /* select the action based on the action number */
        switch (CUP$AnalizadorSintactico$act_num)
        {
            /* ..... */
            case 4: // S ::= coser para S coma S parc
            {
                Object RESULT =null;

                CUP$AnalizadorSintactico$result = parser.getSymbolFactory().newSymbol("S",0,
                ((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-5)),
                ((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()), RESULT);
            }
            return CUP$AnalizadorSintactico$result;

            /* ..... */
            case 3: // S ::= girar para S parc
            {
                Object RESULT =null;

                CUP$AnalizadorSintactico$result = parser.getSymbolFactory().newSymbol("S",0,
                ((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-3)),
                ((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()), RESULT);
            }
        }
    }
}

```

```

return CUP$AnalizadorSintactico$result;

/* ..... */
case 2: // S ::= r2
{
    Object RESULT =null;

    CUP$AnalizadorSintactico$result = parser.getSymbolFactory().newSymbol("S",0,
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()),
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()), RESULT);
}
return CUP$AnalizadorSintactico$result;

/* ..... */
case 1: // S ::= r1
{
    Object RESULT =null;

    CUP$AnalizadorSintactico$result = parser.getSymbolFactory().newSymbol("S",0,
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()),
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()), RESULT);
}
return CUP$AnalizadorSintactico$result;

/* ..... */
case 0: // $START ::= S EOF
{
    Object RESULT =null;
    int start_valleft =
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-1)).left;
    int start_valright =
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-1)).right;
    Object start_val = (Object)((java_cup.runtime.Symbol)
CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-1)).value;
    RESULT = start_val;

    CUP$AnalizadorSintactico$result = parser.getSymbolFactory().newSymbol("$START",0,
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.elementAt(CUP$AnalizadorSintactico$top-1)),
((java_cup.runtime.Symbol)CUP$AnalizadorSintactico$stack.peek()), RESULT);
}
/* ACCEPT */
CUP$AnalizadorSintactico$parser.done_parsing();
return CUP$AnalizadorSintactico$result;

/* ..... */
default:
    throw new Exception(
        "Invalid action number found in internal parse table");
}
}
}

```

Main

```

package compilador.metacompilado;

import java.io.FileReader;

```

```
public class Main {  
  
    public static void main(String[] args) {  
  
        try {  
            System.out.println(new AnalizadorSintactico(new AnalizadorLexico(new  
FileReader("C:\\Users\\German\\Desktop\\compiladores\\CompiladorColchita\\src\\compilador\\metacompilado\\programaTest.tx  
t")).parse().toString());  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Test

```
coser(girar(r1),r2)
```