Universidad de Alcalá.

Grado Ingeniería Informática.

Asignatura: Procesadores del Lenguaje.

Profesor: José Carlos Holgado Martín

Autor: Luis Ángel Parada

DNI: 50502347M

# Memoria PECL2
# Analizador Sintáctico-Semántico

Esta practica consiste en la creación de un compilador para el lenguaje especificado en la practica PECL1, para ello primero tenemos que generar una gramática libre de contexto la que no se ambigua, la cual nos genere un árbol sintáctico capaz de procesar un archivo de tipo “prog” con la estructura especificada.

Para generar nuestro AFD utilice la herramienta CUP la cual trabaja con un LALR el cual permite tener un look ahead del siguiente token que viene del analizador léxico, esta manera podemos recuperarnos de un error sintáctico y recuperarnos y seguir procesando el archivo

Para la parte semantica de la PECL2 he implementado una clase auxiliar y una tabla de símbolos.

A continuación se explicara la gramática diseñada, luego la parte semántica los métodos, la tabla de simboles, clases auxiliares etc.

En nuestra gramática todos los no terminales están en mayúscula con un nombre claramente significativo de que token representan le asignamos la clase String ya que esta clase contiene una cantidad de métodos ya implementados en java que hacen mas sencilla el procesamiento semántico de nuestro analizador. El error lo coloco en donde se pueda cometer un error y dirigirlo a otro bloque del programa para continuar su ejecución.

Como no terminales los representamos en minúscula.

- **axioma**: representa el axioma de nuestra gramática.
- **programa**: representa nuestro bloque “main” de nuestro texto.
- **bloque**: representa lo que esta dentro del bloque main que son ecuaciones de suma y resta como de asignación.
- **expresion:** representa una ecuación que puede ser utilizada en  “ bloque” o en el cuerpo de una función.

- **argumento:** representa los argumentos de una función que pueden ser varias de tipo integre boolean separados con una coma
- **opcion:** representa el tipo del cual se puede formar una sentencia identificador, false, true, función, numero
- **valuetype:** representa el tipo de dato que puede tener una función como parámetro
- **operaciones:** representa el tipo de operadores matemáticos disponibles.
- **decls:** representa una lista de declaraciones de identificadores.
- **decl:** representa una sola declaración de una variable.
- **decfunc:** representa el bloque de declaración de una función
- **func:** representa los argumentos y los tipos que puede tomar
- **body:** el cuerpo de una función.
- **returntype:** el valor que puede devolver una función.
- **tipo:** que puede tomar un identificador integer o boolean.
- **decvar:** el bloque de declaración de variables.
- **asignación:** representa la linea de asignación de de un identificador.
- **sentencia:**
- **formato:** es el formato que puede presentar nuestro lenguaje sumas, restas, y estas operaciones entre paréntesis.
- **formato2:** para la ampliación de nuestras ecuaciones.
- **bloqueclose:** representa el token “end” seguido de “;”

Las precedencias fueron utilizadas para eliminar problemas de reducción desplazamiento o desplazamiento desplazamiento.

```
terminal   String      TK_BEGIN,TK_FUNCTION;
terminal   String      TK_END,TK_VARCDEL,TK_INTEGER;
terminal   String      TK_TRUE,TK_RETURN,TK_FALSE,TK_PROGRAM;
terminal   String      TK_RESTA, TK_IGUAL, TK_ASIGNACION;
terminal   String      TK_OPENPARENT,TK_CLOSEPARENT,
TK_COMA,TK_SEMI,TK_SUMA;
terminal   String   TK_ID,TK_FUNCID,TK_BOOL;
terminal   Integer  TK_NUMERO;

non terminal String    axioma,programa,bloque,expresion,argumento,op-
cion,valuetype,operaciones;
non terminal String   decls,decl,decfunc,func,body,returntype,tipo,dec-
var,asignacion,sentencia,formato,formato2,bloqueclose;
precedence left  TK_RESTA,error,TK_FUNCTION;


axioma::= TK_PROGRAM   decvar decfunc programa
             | error decvar decfunc  programa
programa::= TK_BEGIN  bloque bloqueclose
             | error

bloque::=  /*epsilon*/
           | expresion bloque ;

bloqueclose::= TK_END TK_SEMI
```

```
            |error TK_SEMI
            |error;

sentencia ::=  formato:e TK_SEMI ;

expresion::= TK_ID:E1 TK_IGUAL sentencia:E2
           |error     ;


formato::=  opcion:E
           | TK_OPENPARENT formato:E TK_CLOSEPARENT formato2:E3
           | opcion:E  operaciones   opcion:E1 formato2:E3
                     ;

formato2::=   operaciones  formato:E
                  {:RESULT= E;:}
                |
                ;
operaciones::= TK_SUMA
               |TK_RESTA;




opcion::= TK_ID:E
           | TK_NUMERO:E
           | TK_FUNCID:E  argumento:E2

           |TK_FALSE:E
           |TK_TRUE:E      ;


argumento ::=   TK_OPENPARENT:E valuetype:E1 TK_CLOSEPARENT:E2
                |TK_OPENPARENT valuetype:E1 TK_COMA  argumento:e2
                |valuetype:E1 TK_COMA argumento:e2
                |
                |TK_OPENPARENT:E TK_FUNCID argumento   TK_CLOSEPAREN-
T:E2
                     ;



decvar::= |
             TK_VARCDEL decls TK_END TK_SEMI
             |error decls TK_END TK_SEMI
```

```
              |error decfunc;

decls::= | decl:E1 decls


decl::=    asignacion TK_SEMI
           | error ;

asignacion::=
             TK_ID:E1 TK_ASIGNACION tipo:e
           | TK_ID:E1 TK_COMA asignacion
           ;

tipo::= TK_INTEGER:E
       | TK_BOOL:E


decfunc::=  | TK_FUNCTION TK_FUNCID:E1 func

func ::=  TK_ID:E2 TK_OPENPARENT tipo:E3 TK_CLOSEPARENT body
          |TK_ID:E2 TK_OPENPARENT tipo:E3 TK_CLOSEPARENT TK_COMA func
           |body
           |error body

body::= expresion TK_RETURN returntype:E3  TK_SEMI decfunc
     |TK_RETURN returntype:E3  TK_SEMI decfunc
     |error programa;


valuetype::=  TK_TRUE:E
           | TK_FALSE:E
           | TK_NUMERO:E
           ;
returntype::= TK_TRUE:E
           | TK_FALSE:E
           | TK_NUMERO:E
           | TK_ID:E
           ;
```

Como clase auxiliar utilizamos “Item” que es compuesta de nombre, valor,tipo, numero (numero de llamadas) y una lista de “Item” para representar los parámetros de una función.

Como estructura para representar a la tabla de símbolos utilizamos una hash table ya que permite almacenar distintas clases en un misma estructura y su búsqueda se hace por un “key” que este caso utilizamos el nombre.

Los métodos para la parte semántica son:

check: es una función que es llamada cuando alguna de las producciones de operaciones en el cuerpo de una función como en el “main” del programa se encuentra.
Lo que hace este método es verificar primero si el String que se le manda como parámetro no es nulo si no lo es prosigue a buscarlo en la tabla de símbolos ya que puede ser una función, identificador, numero o false o true, cabe acotar que nuestro lenguaje se han tomado consideraciones para hacerlo mas fuertemente tipado por ende no acepta suma de boleanos, la función devuelve un String indicando el tipo de dato que fue introducido y luego en la producción se compara si son del mismo tipo es una suma o resta valida siempre y cuando sean funciones de tipo entera, identificadores de tipo entero o numeros, devolverá error si es null el String.

check2: es una función que es llamada cuando la producción expresión se cumple esto quiere decir de la forma ( id = …..) aquí se se busca en la tabla de símbolos el identificador y devuelve su tipo, de no encontrarse en la tabla de símbolos devolverá “No declarado” para así poder indicar el error por consola las expresiones solo admiten operaciones de su mismo tipo claro esta.

checkfunc: es una función que se llama cuando una función que es llamada cuando una función es invocada para devolver la cantidad de parámetros de esa función si es que los posee y devuelve un entero de la cantidad de parámetros, esta función recibe un String que sera el TK_FUNCTIONID.

param: es una función que recibe en texto los argumentos de argumentos con la que fue llamada la función y calcula la cantidad de parámetros y devuelve la cantidad de parámetros.

parametros2: es una función la cual comprueba si los tipos de datos mandando a las funciones son los correctos, primero verifica que no sea null el String que se le manda que es el nombre de la función luego se busca en la tabla de símbolos y se almacena en una variable temporal la lista de parámetros y se compara con el otro String argumento para ver si concuerdan en tipos, si lo es devuelve “True” si no “False” y si no se encuentra en la tabla de símbolos devuelve “No declarado”.

PECL2

```
axioma::= TK_PROGRAM   decvar decfunc programa  {: reporte();:}
                    l error decvar decfunc  programa {: errors++; reporte(); :};

programa::= TK_BEGIN  bloque bloqueclose
                              |
                              l error {:  errors++;:};


bloque::=  /*epsilon*/
                    l expresion bloque ;
bloqueclose::= TK_END TK_SEMI
                               lerror TK_SEMI{:  errors++; reporte();:}
                               lerror {:  errors++; reporte();:}
                               ;

sentencia ::=  formato:e TK_SEMI
                                    {:
                                                RESULT=e; :} ;


expresion::= TK_ID:E1 TK_IGUAL sentencia:E2
                              {:

                                    String token = check2(E1);

                                    String token2= check(E2);

                                    if(token.equalsIgnoreCase(token2) )
                                    {
                                          RESULT=token;
                                    }
                                    else
                                    {
                                          if(token.equalsIgnoreCase("No declarado"))
                                          {
                                                System.out.println(" Variable no declarada: "
+""+E1+""+ " en la linea " + E1left + " en la columna " +E1right );
                                                semantic=semantic + 1;
                                                System.out.println("");
                                          }
                                          else
                                          {
                                                System.out.println(" Asignacion de tipos invalidos  "
+ ""+E1+"' tipo  " + token.toUpperCase() + " y "+ ""+E2+"' tipo  "+ token2.toUpperCase()+ " en la
linea : " + E1left + " en la columna " +E1right );
                                                semantic=semantic + 1;
                                                System.out.println("");
                                          }
                               }
```

```
                        :}
                        |error {:  errors++;:}
                        ;


formato::=  opcion:E

                        {:
                                RESULT=E;
                        :}
                | TK_OPENPARENT formato:E TK_CLOSEPARENT formato2:E3
                 {:

                        String token= check (E3);

                        if (!token.equalsIgnoreCase(E))
                        {

                                System.out.println(" Error en suma de tipos  "+  ""+E+""+ "con "+
""+token+""  +" en la linea " + Eleft + " en la columna " + Eright);
                                System.out.println("");
                                semantic++;
                                RESULT="ERROR";

                        }


                        RESULT=E;

                 :}
                | opcion:E  operaciones   opcion:E1 formato2:E3
                {:

                        String token1=check(E);
                        String token2=check(E1);

                        if(token1.equalsIgnoreCase(token2))
                        {

                                if(token1.equalsIgnoreCase("boolean"))
                                {
                                System.out.println(" Error en suma de tipos  "+  ""+E+""+ " " + to-
ken1.toUpperCase() + " " +  ""+E1+"" + " " + token2.toUpperCase() + " en la linea " + Eleft + "
en la columna " + Eright);
                                semantic=semantic + 1;

                                RESULT="ERROR";
                                }
                                else
                                {
                                RESULT=check(E);
                                }
```

```
                        }
                        else
                        {
                        System.out.println(" Error en suma de tipos  "+  """+E+"""+ " " +
token1.toUpperCase() + " " +  """+E1+"""  + " " + token2.toUpperCase() + " en la linea " + Eleft + "
en la columna " + Eright);
                                semantic=semantic + 1;
                                System.out.println("");
                                RESULT="ERROR"    ;
                        }

                :}
                ;

formato2::=   operaciones  formato:E
                         {:RESULT= E;:}
                        |
                        ;
operaciones::= TK_SUMA
                          |TK_RESTA;

opcion::= TK_ID:E {: RESULT = E;:}
                | TK_NUMERO:E {: RESULT = E.toString();:}
                | TK_FUNCID:E  argumento:E2
                {:

                        if((parametros2(E2,E).equalsIgnoreCase("No declarado")))
                        {
                                System.out.println("La funcion: " + E+ " en la columna "  + Eleft + "
linea " + Eright + " No esta declarada");
                                semantic=semantic + 1;
                                System.out.println("");
                        }
                        else
                        {
                                if(!(checkfunc(E)==param(E2)))
                                {
                                        System.out.println("Error en la llamada de la funcion: " + E
+ " en la columna "  + Eleft + " linea " + Eright + " la cantidad de parametros son incorrectos" );
                                        errors=errors + 1;
                                        System.out.println("");
                                }
                                else
                                {
                                        if((parametros2(E2,E).equalsIgnoreCase("false")))
                                {
                                        System.out.println("La funcion: " + E+ " en la columna "  +
Eleft + " linea " + Eright + " los tipos de parametros son incorrectos ");
                                        semantic=semantic + 1;
                                        System.out.println("");
```

```
                        }
                    }
                    }


                        RESULT = E;
                :}
                |TK_FALSE:E {: RESULT= E; :}
                |TK_TRUE:E {: RESULT= E; :}

        ;


argumento ::=   TK_OPENPARENT:E valuetype:E1 TK_CLOSEPARENT:E2
                                {:

                                    RESULT=E1+",";
                                    :}
                                |TK_OPENPARENT valuetype:E1 TK_COMA  argumento:e2
                                 {:
                                      RESULT=E1+","+e2;      :}
                                |valuetype:E1 TK_CLOSEPARENT
                                {:
                                  RESULT=E1+","; :}
                                |valuetype:E1 TK_COMA argumento:e2
                                {:
                                  RESULT=E1+","+e2; :}
                                | {: RESULT= ",";:}

                                |TK_OPENPARENT:E TK_FUNCID argumento   TK_CLOSEPAR-
ENT:E2
                                ;




decvar::= |
            TK_VARCDEL decls TK_END TK_SEMI
            |error decls TK_END TK_SEMI {:  errors++;:}
                ;

decls::= | decl:E1 decls
                 {:   RESULT=E1;     :};
```

PECL2

```
decl::= asignacion TK_SEMI
                        | error {:errors++;:} ;

asignacion::=
                TK_ID:E1 TK_ASIGNACION tipo:e
                {:      Item i= new Item(E1,"",e);
                        contenedor.put(E1,i);
                        type=e;
                        RESULT = e ;
                                :}
               | TK_ID:E1 TK_COMA asignacion
               {:
                        Item i = new Item(E1,"",type);
                        contenedor.put(E1,i);
               :}


               ;

tipo::= TK_INTEGER:E {: RESULT = E;:}
        | TK_BOOL:E {: RESULT = E;:} ;


decfunc::=  | TK_FUNCTION TK_FUNCID:E1 func
                        {:
                        Item i = new Item(E1,"funcion",check(returno),
((ArrayList<Item>)Tokens.clone()));
                        contenedor.put(E1,i);
                        Tokens.removeAll(Tokens);
                            :}

                         |error decfunc{:errors++;
                         :};



func ::=  TK_ID:E2 TK_OPENPARENT tipo:E3 TK_CLOSEPARENT body

                        {:    Item i = new Item (E2,"",E3);
                                contenedor.put(E2,i);

                                Tokens.add(i);
                                        :}
                      |TK_ID:E2 TK_OPENPARENT tipo:E3 TK_CLOSEPARENT TK_COMA func
                      {:
                        Item i = new Item(E2,"",E3);
                        contenedor.put(E2,i);
                        Tokens.add(i);
                            :}

                      |body
```

```
                    ;
body ::= expresion TK_RETURN returntype:E3  TK_SEMI decfunc
                    {:   returno=E3;
                     :}
                    |TK_RETURN returntype:E3  TK_SEMI decfunc
                    {:   returno=E3;
                     :}
                    |error decfunc {:  errors++;:} ;


valuetype::=  TK_TRUE:E
                                {: RESULT= E; :}
                            | TK_FALSE:E
                                {: RESULT= E; :}
                            | TK_NUMERO:E
                                {: RESULT= E.toString(); :}
                            ;
returntype::= TK_TRUE:E
                                {: RESULT = E; :}
                            | TK_FALSE:E
                                {: RESULT= E;   :}
                            | TK_NUMERO:E
                                {: RESULT= E.toString(); :}
                            | TK_ID:E
                                {: RESULT= E; :}

                                ;
```

# Pruebas Realizadas

| Tipos de Pruebas | Argv[] |
|---|---|
| Extension | 0 |
| Declaración de una o varias variables | 1 |
| Declaración de una o varias Funciones | 2 |
| Programa con varias ecuaciones válidas | 3 |
| Programa con errores sintácticos en varios cuerpo del mismo | 4 |
| Programa con problema de asignación de tipos | 5 |
| Programa con problema de suma de tipos | 6 |
| Problema con problemas en los parámetros de las funciones | 7 |

PECL2

| Tipos de Pruebas | Argv[] |
|---|---|
| Problema con problemas sintácticos, semánticos, léxicos | 8 |

Archivo0: formato invalido.

```
Comenzando el análisis de su archivo: Archivo0.txt
Su Archivo:  Archivo0.txt no es compatible
```

Archivo1:

Sin ningún problema.

```
Comenzando el análisis de su archivo: Archivo1.prog

----------------------------------------------------------------------
Numero de errores sintacticos: 0
Numero de errores semanticos: 0
Numero de errores lexicos: 0
```

Archivo2 dos funciones declaradas mas no utilizadas.

```
Comenzando el análisis de su archivo: Archivo2.prog

----------------------------------------------------------------------
Numero de errores sintacticos: 0
Numero de errores semanticos: 0
Numero de errores lexicos: 0
Warning: la funcion $nombre3 ha sido declarada y no usada
Warning: la funcion $nombre1 ha sido declarada y no usada
```

Archivo3. Ecuaciones validas utilizadas en el texto no hay problemas

```
Comenzando el análisis de su archivo: Archivo3.prog

------------------------------------------------------------------------
Numero de errores sintacticos: 0
Numero de errores semanticos: 0
Numero de errores lexicos: 0
Warning: la funcion $nombre3 ha sido declarada y no usada
```

Archivo4. Errores en la declaración de variables en la asignación de tipo de variables, en la declaración de una funcion, y en el cuerpo del programa

```
Comenzando el análisis de su archivo: Archivo4.prog

Error sintactico en la linea 2 2, token VARDEC incorrecto.
Error sintactico en la linea 7 10, token asd incorrecto.
Error sintactico en la linea 19 1, token Begin incorrecto.
Error sintactico en la linea 25 1, token id2 incorrecto.
------------------------------------------------------------------------
Numero de errores sintacticos: 4
Numero de errores semanticos: 0
Numero de errores lexicos: 0
Warning: la funcion $nombre3 ha sido declarada y no usada
```

Archivo5 asignaciones de tipos inválidos integer a variables boleadas y viceversa.

```
Comenzando el análisis de su archivo: Archivo5.prog

 Asignacion de tipos invalidos  'id' tipo  INTEGER y 'false' tipo  BOOLEAN en la
 linea : 20 en la columna 1

 Asignacion de tipos invalidos  'id4' tipo  BOOLEAN y 'INTEGER' tipo  INTEGER en
 la linea : 22 en la columna 1

 Asignacion de tipos invalidos  'id2' tipo  INTEGER y 'id6' tipo  BOOLEAN en la
linea : 24 en la columna 1

------------------------------------------------------------------------
Numero de errores sintacticos: 0
Numero de errores semanticos: 3
Numero de errores lexicos: 0
Warning: la funcion $nombre3 ha sido declarada y no usada
Warning: la funcion $nombre1 ha sido declarada y no usada
```

Archivo6. error en sumas de booleans y interos y viceversa.

```
Comenzando el análisis de su archivo: Archivo6.prog

 Error en suma de tipos  'INTEGER'con 'BOOLEAN' en la linea 21 en la columna 14

 Error en suma de tipos  '1' INTEGER 'id5' BOOLEAN en la linea 22 en la columna
5

 Error en suma de tipos  '1' INTEGER 'id6' BOOLEAN en la linea 24 en la columna
6

------------------------------------------------------------------------
Numero de errores sintacticos: 0
Numero de errores semanticos: 3
Numero de errores lexicos: 0
Warning: la funcion $nombre4 ha sido declarada y no usada
Warning: la funcion $nombre1 ha sido declarada y no usada
```

Archivo7 funciones con parámetros incorrectos de tipo y de cantidad si la función no esta declarada también se detectara este tipo de errores

```
Comenzando el análisis de su archivo: Archivo7.prog

La funcion: $nombre1 en la columna 21 linea 10 los tipos de parametros son incor
rectos

Error en la llamada de la funcion: $nombre1 en la columna 23 linea 7 la cantidad
 de parametros son incorrectos

Error en la llamada de la funcion: $nombre3 en la columna 25 linea 7 la cantidad
 de parametros son incorrectos

La funcion: $nombre5 en la columna 27 linea 7 No esta declarada

------------------------------------------------------------------------
Numero de errores sintacticos: 2
Numero de errores semanticos: 2
Numero de errores lexicos: 0
Warning: la funcion $nombre4 ha sido declarada y no usada
```

Archivo8 es una mezcla de todos los tipos de errores posibles que puede tener a la hora de la compilación y vemos como el análisis no se detiene continua hasta acaba el uso de variables no declaradas también esta implementado.

```
Comenzando el análisis de su archivo: Archivo8.prog

Error sintactico en la linea 2 2, token VARDEC incorrecto.
Error sintactico en la linea 4 5, token id4 incorrecto.
 Variable no declarada: 'id3' en la linea 9 en la columna 4

Error sintactico en la linea 20 1, token Begin incorrecto.
La funcion: $nombre1 en la columna 22 linea 10 los tipos de parametros son incor
rectos

 Variable no declarada: 'id' en la linea 22 en la columna 1

Error en la llamada de la funcion: $nombre1 en la columna 24 linea 8 la cantidad
 de parametros son incorrectos

 Error en suma de tipos  '1' INTEGER '$nombre3' BOOLEAN en la linea 25 en la col
umna 5

 Variable no declarada: 'id' en la linea 26 en la columna 1

Error en la linea: 30 columna: 1 Error lexico '%'
Error en la linea: 31 columna: 2 Error lexico '&'
 Asignacion de tipos invalidos  'id4' tipo  BOOLEAN y '1' tipo  INTEGER en la li
nea : 28 en la columna 1

 Variable no declarada: 'id19' en la linea 34 en la columna 1

--------------------------------------------------------------------------------
Numero de errores sintacticos: 4
Numero de errores semanticos: 7
Numero de errores lexicos: 2
Warning: la funcion $nombre4 ha sido declarada y no usada
```