Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 1

Escuela de Vacaciones

Ing. Mario Bautista

Aux. Francisco Puac

MANUAL DE USUARIO

Proyecto

JPR

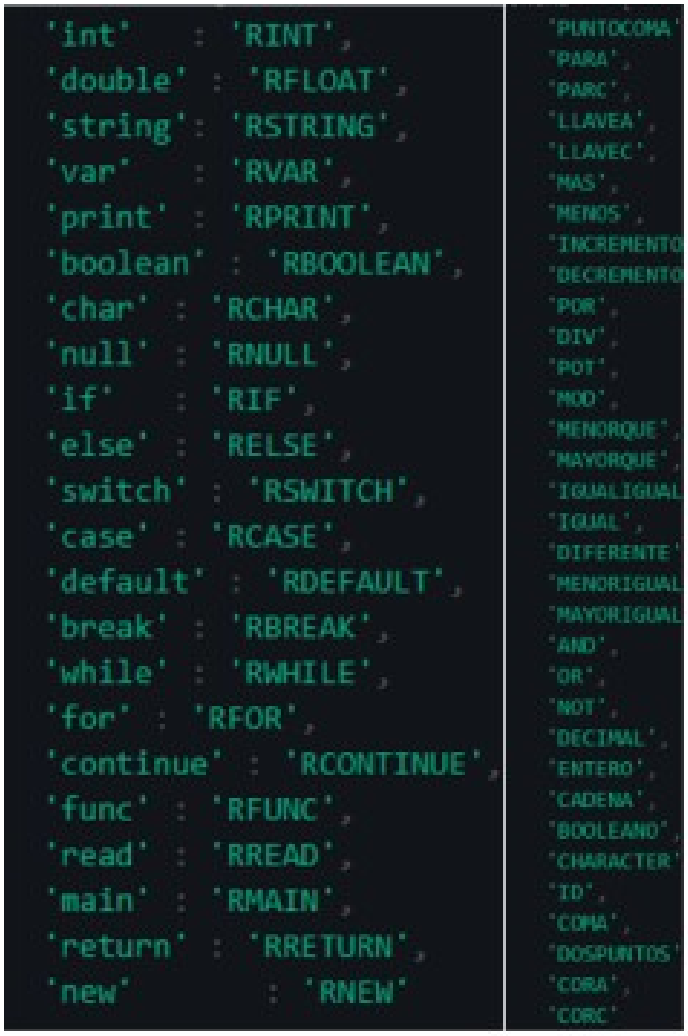
Realizado por:

3149675670901 - Jose Alejandro Barrios Rodas

Fecha: 4 de Julio de 2021

Gramatica

Terminales



```
'int'    : 'RINT',
'double' : 'RFLOAT',
'string': 'RSTRING',
'var'    : 'RVAR',
'print'  : 'RPRINT',
'boolean' : 'RBOOLEAN',
'char'  : 'RCHAR',
'null'  : 'RNULL',
'if'    : 'RIF',
'else'  : 'RELSE',
'switch' : 'RSWITCH',
'case'  : 'RCASE',
'default' : 'RDEFAULT',
'break'  : 'RBREAK',
'while'  : 'RWHILE',
'for'  : 'RFOR',
'continue' : 'RCONTINUE',
'func'  : 'RFUNC',
'read'  : 'RREAD',
'main'  : 'RMAIN',
'return' : 'RRETURN',
'new'        : 'RNEW'
```

```
'PUNTOCOMA'
'PARA',
'PARC',
'LLAVEA',
'LLAVEC',
'MAS',
'MENOS',
'INCREMENTO
'DECREMENTO
'POR',
'DIV',
'POT',
'MOD',
'MENORQUE',
'MAYORQUE',
'IGUALIGUAL
'IGUAL',
'DIFERENTE'
'MENORIGUAL
'MAYORIGUAL
'AND',
'OR',
'NOT',
'DECIMAL',
'ENTERO',
'CADENA',
'BOOLEANO',
'CHARACTER'
'ID',
'COMA',
'DOSPUNTOS'
'CORA',
'CORC'
```

No terminales

```
def p_init(t) :
    'init                : instrucciones'

def p_instrucciones_instrucciones_instruccion(t) :
    'instrucciones       : instrucciones instruccion'

def p_instrucciones_instruccion(t) :
    'instrucciones       : instruccion'
```

```
def p_instruccion(t) :
    '''instruccion          : imprimir_instr finins
                            | declaracion_instr finins
                            | declaracion_instr2 finins
                            | asignacion_instr finins
                            | asignacion2_instr finins
                            | if_instr
                            | switch_instr
                            | while_instr
                            | break_instr finins
                            | continue_instr finins
                            | for_instr
                            | main_instr
                            | funcion_instr
                            | llamada_instr finins
                            | return_instr finins
                            | declArr_instr finins
                            | modArr_instr finins
                           '''

def p_finins(t) :
    '''finins           : PUNTOCOMA
                        | '''

def p_instruccion_error(t):
    'instruccion            : error PUNTOCOMA'
    errores
```

```
def p_imprimir(t) :
    'imprimir_instr        : RPRINT PARA expresion PARC'

def p_declaracion(t) :
    'declaracion_instr        : tipo ID IGUAL expresion'

def p_declaracion_nula(t) :
    'declaracion_instr2        : tipo ID'

def p_declArr(t) :
    '''declArr_instr        : tipo1
                            | tipo2
                            | arreglo_referencia'''

def p_tipo1_arreglo(t) :
    '''tipo1        : tipo lista_Dim ID IGUAL RNEW tipo lista_expresiones'''

def p_arreglo_referencia(t):
    'arreglo_referencia     : tipo lista_Dim ID IGUAL ID'

def p_lista_Dim1(t) :
    'lista_Dim        : lista_Dim CORA CORC'

def p_lista_Dim2(t) :
    'lista_Dim       : CORA CORC'

def p_lista_expresiones_1(t) :
```

```
    'lista_expresiones        : lista_expresiones CORA expresion CORC'

def p_lista_expresiones_2(t) :
    'lista_expresiones       : CORA expresion CORC'

def p_tipo2_arreglo(t):
    ' tipo2 : tipo lista_Dim ID IGUAL lst_values '

def p_lst_values(t) :
    ' lst_values      :   lst_values COMA LLAVEA value LLAVEC '

def p_lst_value(t) :
    ' lst_values      : LLAVEA value LLAVEC '

def p_value(t):
    '''

def p_lst_values_expresio(t) :
    ' lst_expresion       : lst_expresion COMA expresion '

def p_lst_value_expresion_final(t) :
    ' lst_expresion       : expresion '

def p_modArr(t) :
    '''modArr_instr        :   ID lista_expresiones IGUAL expresion'''

def p_asignacion(t) :
    'asignacion_instr        : ID IGUAL expresion'
```

```
def p_asignacion2(t) :
    '''asignacion2_instr        : ID MASMAS
                                | ID MENOSMENOS '''
def p_if1(t) :
    'if_instr       : RIF PARA expresion PARC LLAVEA instrucciones LLAVEC'

def p_if2(t) :
    'if_instr       : RIF PARA expresion PARC LLAVEA instrucciones LLAVEC
RELSE LLAVEA instrucciones LLAVEC'

def p_if3(t) :
    'if_instr       : RIF PARA expresion PARC LLAVEA instrucciones LLAVEC
RELSE if_instr'

def p_switch_instr(t):
    'switch_instr : RSWITCH PARA expresion PARC LLAVEA lista_case
RDEFAULT DOSPUNTOS instrucciones LLAVEC'

def p_switch_instr2(t):
    'switch_instr : RSWITCH PARA expresion PARC LLAVEA lista_case
LLAVEC'

def p_switch_instr3(t):
    'switch_instr : RSWITCH PARA expresion PARC LLAVEA RDEFAULT
DOSPUNTOS instrucciones LLAVEC'

def p_switch_lista_case(t):
```

```
    'lista_case : lista_case case_instrucciones'

def p_caseInstrucciones(t):
    'lista_case : case_instrucciones'

def p_switch_case(t):
    'case_instrucciones : RCASE expresion DOSPUNTOS instrucciones'

def p_for_instr_asignacion(t):
    'for_instr : RFOR PARA asignacion_instr PUNTOCOMA expresion
PUNTOCOMA asignacion2_instr PARC LLAVEA instrucciones LLAVEC'

def p_for_instr_declaracion(t):
    'for_instr : RFOR PARA declaracion_for PUNTOCOMA expresion
PUNTOCOMA asignacion2_instr PARC LLAVEA instrucciones LLAVEC'

def p_declaracion_for(t):
    'declaracion_for :  tipo_declaracion_for ID IGUAL expresion'

def p_tipo_declaracion_for(t):
    '''tipo_declaracion_for : RINT
                            | RVAR '''

def p_while(t) :
    'while_instr        : RWHILE PARA expresion PARC LLAVEA instrucciones
LLAVEC'

def p_break(t) :
```

```
    'break_instr        : RBREAK'

def p_continue(t) :
    'continue_instr        : RCONTINUE'

def p_main(t) :
    'main_instr        : RMAIN PARA PARC LLAVEA instrucciones LLAVEC'

def p_funcion_1(t) :
    'funcion_instr        : RFUNC ID PARA parametros PARC LLAVEA
instrucciones LLAVEC'

def p_funcion_2(t) :
    'funcion_instr        : RFUNC ID PARA PARC LLAVEA instrucciones
LLAVEC'

def p_parametros_1(t) :
    'parametros        : parametros COMA parametro'

def p_parametros_2(t) :
    'parametros       : parametro'

def p_parametro(t) :
    'parametro        : tipo ID'

def p_parametro_arreglo(t) :
    'parametro        : tipo lista_Dim ID'
```

```
def p_llamada1(t) :
    'llamada_instr       : ID PARA PARC'

def p_llamada2(t) :
    'llamada_instr       : ID PARA parametros_llamada PARC'

def p_parametrosLL_1(t) :
    'parametros_llamada       : parametros_llamada COMA parametro_llamada'

def p_parametrosLL_2(t) :
    'parametros_llamada      : parametro_llamada'

def p_parametroLL(t) :
    'parametro_llamada       : expresion'

def p_return(t) :
    'return_instr       : RRETURN expresion'

def p_tipo(t) :
    '''tipo       : RINT
                  | RDOUBLE
                  | RSTRING
                  | RBOOLEAN
                  | RCHAR
                  | RVAR   '''

def p_expresion_binaria(t):
    '''
```

```
    expresion : expresion MAS expresion
              | expresion MENOS expresion
              | expresion POR expresion
              | expresion DIV expresion
              | expresion POT expresion
              | expresion MOD expresion
              | expresion MENORQUE expresion
              | expresion MAYORQUE expresion
              | expresion IGUALIGUAL expresion
              | expresion DIFERENTE expresion
              | expresion MENORIGUAL expresion
              | expresion MAYORIGUAL expresion
              | expresion AND expresion
              | expresion OR expresion
              | expresion MASMAS
              | expresion MENOSMENOS
    '''

def p_expresion_unaria(t):
    '''
    expresion : MENOS expresion %prec UMENOS
              | NOT expresion %prec UNOT
    '''

def p_expresion_agrupacion(t):
    '''
    expresion :    PARA expresion PARC
    '''
```

```
def p_expresion_llamada(t):
    '''expresion : llamada_instr'''

def p_expresion_identificador(t):
    '''expresion : ID'''

def p_expresion_entero(t):
    '''expresion : ENTERO'''

def p_expresion_decimal(t):
    '''expresion : DECIMAL'''

def p_expresion_cadena(t):
    '''expresion : CADENA'''

def p_expresion_booleano(t):
    '''expresion : BOOLEANO'''

def p_expresion_caracter(t):
    '''expresion : CARACTER'''

def p_expresion_null(t):
    '''expresion : RNULL'''

def p_expresion_read(t):
    '''expresion : RREAD PARA PARC'''
```

```
def p_expresion_cast(t):
    '''expresion : PARA tipo PARC expresion'''

def p_expresion_arreglo(t):
    '''expresion : ID lista_expresiones'''



def crearNativas(ast):
    nombre = "toupper"

    nombre = "tolower"

    nombre = "length"

    nombre = "truncate"

    nombre = "round"

    nombre = "typeof"
```