Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Siste+

Organización de Lenguajes y Compiladores 1

Ing. Mario Bautista

Aux. Emely García

Aux. José Morán

GRAMÁTICAS

Marvin Eduardo Catalán Véliz

201905554

Guatemala, 4 de sep. de 21

GRAMÁTICA FCA

```
//Declaracion de terminales, no terminales y precedencias(de<a>)
terminal String (,);
terminal String {,};
terminal String [,];
terminal String ADD,MINUS,TIMES,DIV;
terminal String NUMBER;
terminal String DECIMAL;
terminal String UMINUS;
terminal String ;;
terminal String RGENERARREP||TEESTADISTICO;
terminal String COMPARE;
terminal String ,;
terminal String CARACTERESVARIABLES;
terminal String =;
terminal String STRING;
terminal String DOUBLE;
terminal String DEFINIRGLOBALES;
terminal String CADENACOMILLAS;
terminal String CADENACOMILLASSIMPLE;
terminal String GRAFICADEBARRAS;
terminal String TITULO;
terminal String :;
terminal String EJEX;
terminal String TITULOY;
terminal String TITULOX;
terminal String VAL||ES;
terminal String GRAFICAPIE;
terminal String GRAFICALINEAS;
terminal String ARCHIVO;
```

```
terminal String PUNTAJEGENERAL;

non terminal ini;
non terminal generalinstruction;
non terminal Double expression;
non terminal instruction;
non terminal instructionrecursive;
non terminal String expressioncompare;
non terminal String double;
non terminal String string;
non terminal String variables;
non terminal String graficabarras;
non terminal ArrayList<String> cadenaovariable;
non terminal String instrucciongraficabarras;
non terminal String instrucciongraficapie;
non terminal String instruccionlineas;
non terminal ArrayList<String> numeroovariable;
non terminal String graficapie;
non terminal String graficalineas;

precedence left ADD,MINUS;
precedence left TIMES,DIV;
precedence right UMINUS;

//produccion p|| donde empezara el analizad|| sintactico
start with ini;

ini ::=
      generalinstruction
      | err||
```

```
;
generalinstruction ::=
        RGENERARREP||TEESTADISTICO { instructionrecursive }
;

instructionrecursive ::=
     instruction instructionrecursive
    |instruction
    |err||
;

instruction ::=
    DEFINIRGLOBALES { variables }
{:logic.variablesGloblales(varglobal);:}
    |COMPARE expressioncompare
    |GRAFICADEBARRAS { instruccionGraficabarras }
{:logic.Titulos(valtitulo,valtitulox,valtituloy);

logic.GraficaBarras();:}
    |GRAFICAPIE { instruccionGraficapie }
{:logic.GraficaPie(valxpie,valypie);:}
    |GRAFICALINEAS { instruccionGraficalineas }
{:logic.GraficaLineas(valtitulolineas,archivo);:}
;

variables ::=
    variables string
    |variables double
    |string
    |double
    |err||
;
```

```
instruccionGraficaBarras::=
    graficaBarras instruccionGraficaBarras
    |graficaBarras
    |err||
;

graficaBarras::=
    TITULO : CARACTERESVARIABLES:a ; {:valtitulo=a;:}
    |TITULO : CADENACOMILLAS:a ; {:valtitulo=a;:}
    |EJEX : [ cadenaOVariable:listatitulosx ] ; {:f||(int
i=0;i<listatitulosx.size();i++){

valx.add(listatitulosx.get(i));

}logic.variablesTitulosX(valx);:}
    |VAL||ES : [ numeroOVariable:listatitulosy  ]    ; {:f||(int
i=0;i<listatitulosy.size();i++){

valy.add(listatitulosy.get(i));

}logic.variablesTitulosY(valy);:}
    |TITULOX : CARACTERESVARIABLES:a ; {:valtitulox=a;:}
    |TITULOX : CADENACOMILLAS:a ; {:valtitulox=a;:}
    |TITULOY : CADENACOMILLAS:a ; {:valtituloy=a;:}
    |TITULOY : CARACTERESVARIABLES:a ; {:valtituloy=a;:}
;
instruccionGraficaPie::=
    graficaPie instruccionGraficaPie
    |graficaPie
    |err||
;
graficaPie::=
```

```
    TITULO : CARACTERESVARIABLES:a ; {:valtitulopie=a;:}

    |TITULO : CADENACOMILLAS:a ; {:valtitulopie=a;:}

    |EJEX : [ cadenaovariable:listatitulosx ] ; {:f||(int
i=0;i<listatitulosx.size();i++){


valxpie.add(listatitulosx.get(i));


}:}

    |VAL||ES : [ numeroovariable:listatitulosy  ]    ; {:f||(int
i=0;i<listatitulosy.size();i++){


valypie.add(listatitulosy.get(i));}:}

;


instrucciongraficalineas::=

    graficalineas instrucciongraficalineas

    |graficalineas

    |err||

;


graficalineas::=

    TITULO : CARACTERESVARIABLES:a ; {:valtitulolineas=a;:}

    |TITULO : CADENACOMILLAS:a ; {:valtitulolineas=a;:}

    |ARCHIVO : CARACTERESVARIABLES:a ; {:archivo=a;:}

    |ARCHIVO : CADENACOMILLAS:a ; {:archivo=a;:}

;


cadenaovariable::=

     cadenaovariable:listatitulosx , CARACTERESVARIABLES:titulosx
{:RESULT= listatitulosx;


RESULT.add(titulosx);:}
```

```
    |cadenaovariable:listatitulosx , CADENACOMILLAS:titulosx
{:RESULT= listatitulosx;

RESULT.add(titulosx);:}

    |CARACTERESVARIABLES:titulosx
{:RESULT = new ArrayList<>();

RESULT.add(titulosx);:}

    |CADENACOMILLAS:titulosx
{:RESULT = new ArrayList<>();

RESULT.add(titulosx);:}

    |err||

;


numeroovariable::=

     numeroovariable:listatitulosy , DECIMAL:titulosy
{:RESULT= listatitulosy;

RESULT.add(titulosy);:}

    |numeroovariable:listatitulosy , CARACTERESVARIABLES:titulosy
{:RESULT= listatitulosy;

RESULT.add(titulosy);:}

    |numeroovariable:listatitulosy , NUMBER:titulosy
{:RESULT= listatitulosy;

RESULT.add(titulosy);:}

    |DECIMAL:titulosy
{:RESULT = new ArrayList<>();

RESULT.add(titulosy);:}

    |NUMBER:titulosy
{:RESULT = new ArrayList<>();

RESULT.add(titulosy);:}
```

```
    |CARACTERESVARIABLES:titulosy
{:RESULT = new ArrayList<>();

RESULT.add(titulosy);:}

    |err||

;


string ::=

    STRING CARACTERESVARIABLES:a =  CADENACOMILLAS:b  ;
{:VariableGlobal var = new VariableGlobal(a,b,"string");


varglobal.add(var);:}

;

double ::=

    DOUBLE CARACTERESVARIABLES:a =  DECIMAL:b  ; {:VariableGlobal var =
new VariableGlobal(a,b,"double");


varglobal.add(var);:}

    |DOUBLE CARACTERESVARIABLES:a =  NUMBER:b  ; {:VariableGlobal var =
new VariableGlobal(a,b,"double");


varglobal.add(var);:}

    |DOUBLE CARACTERESVARIABLES:a =  PUNTAJEGENERAL ; {:VariableGlobal
var = new VariableGlobal(a,null,"general");


varglobal.add(var);:}

;



expressioncompare ::=

    (  expressioncompare:a  ,  expressioncompare:b  ) ; {:archivosruta1
= logic.Obtenerarchivos(a);


archivosruta2 = logic.Obtenerarchivos(b);
```

```
logicjs.Obtenerrutas(a,b);

logicjs.Obtenerarchivos(archivosruta1,archivosruta2);

logicjs.c||rerjsproyectoA();

logicjs.c||rerjsproyectoB();

:}
    |CADENACOMILLASSIMPLE:c                {:RESULT=c;:}
;
```

GRAMÁTICA JAVASCRIPT

```
//Declaracion de terminales, no terminales y precedencias(de<a>)
terminal String {,};
terminal String PARLEFT,PARRIGHT;
terminal String CLASS;
terminal String !(;
terminal String CARACTERESVARIABLES;
terminal String NUMBER;
terminal String DECIMAL;
terminal String ,,:;
terminal String ;;
terminal String VAR,LET,CONST;
terminal String =;
terminal String CADCOM;
terminal String CADCOMSIM;
terminal String IF;
terminal String !;
terminal String <,>;
terminal String &&,||,!;
terminal String ELSE,F||,WHILE,DO;
terminal String +,-,P||,/,**,%;
terminal String SWITCH,CASE,BREAK,DEFAULT;
terminal String PRINT,REQUIRE;



non terminal ini;
non terminal generalinstruction;
non terminal instructionrecursive;
```

```
non terminal instruction;
non terminal ArrayList<String> recibirparametros;
non terminal String variables;
non terminal variablesrecursivas;
non terminal sentenciaif;
non terminal recibirrelacion;
non terminal typevariable;
non terminal relacionconosinoperad||;
non terminal sentenciaelse;
non terminal sentenciaelseif;
non terminal sentenciaf||;
non terminal tiposvar;
non terminal instruccionf||;
non terminal operad||esrelacionales;
non terminal aumentodecremento;
non terminal String operad||esaritmeticos;
non terminal operad||eslogicos;
non terminal String typenumberovariable,prints;
non terminal oprecursivas;
non terminal sentenciawhile;
non terminal sentenciadowhile;
non terminal sentenciaswitch;
non terminal recursivecase,caseswitch,consolelog;
non terminal String recursiveprints;
non terminal llamada;
non terminal ArrayList<String> variablesglobales;

//produccion p|| donde empezara el analizad|| sintactico
start with ini;
```

```
ini ::=
    generalinstruction ini
    |generalinstruction
    |err||
;


generalinstruction ::=
        CLASS CARACTERESVARIABLES:tt {:a instructionrecursive }:b
{:logicjs.recibirlistas(idvariables,idclases,idfunciones);

claseactual=tt; idclases.add(tt);

Repitenciaclase clase = new
Repitenciaclase(tt,funcionesrepetidas,(bleft-aleft+1));

clasesrepetidas.add(clase);

DatosJs datopadre = new DatosJs(clasesrepetidas,idvariables);

todo.add(datopadre);

logicjs.recibirdatopadreactual(datopadre,archivo);logicjs.imprimirs();

logicjs.imprimirvect||final();

:}
;


instructionrecursive ::=
     instruction instructionrecursive
    |instruction
    |err||
;
```

```
instruction ::=

    CARACTERESVARIABLES:nombre PARLEFT PARRIGHT {:inicio
variablesrecursivas }:fin {:Repitenciametodo metodo = new
Repitenciametodo(nombre,0,(finright-inici||ight+1),claseactual) ;


funcionesrepetidas.add(metodo);


idfunciones.add(nombre);:}

    |CARACTERESVARIABLES:nombre PARLEFT recibirparametros:paramlist
PARRIGHT {:inicio variablesrecursivas }:fin {:Repitenciametodo metodo =
new Repitenciametodo(nombre,paramlist.size(),(finright-
inici||ight+1),claseactual) ;


funcionesrepetidas.add(metodo);idfunciones.add(nombre);:}

    |llamada

    |variablesglobales

;

llamada::=

    CARACTERESVARIABLES PARLEFT PARRIGHT

    |CARACTERESVARIABLES PARLEFT recibirparametros PARRIGHT

    |CARACTERESVARIABLES PARLEFT PARRIGHT ;

    |CARACTERESVARIABLES PARLEFT recibirparametros PARRIGHT ;

    ;

variablesglobales::=

    tiposvar CARACTERESVARIABLES:a = typevariable ;
{:idvariables.add(a);:}

    |tiposvar CARACTERESVARIABLES:a = typevariable
{:idvariables.add(a);:}

    |tiposvar CARACTERESVARIABLES:a ; {:idvariables.add(a);:}

    |tiposvar CARACTERESVARIABLES:a {:idvariables.add(a);:}

    |tiposvar CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable
PARRIGHT {:idvariables.add(a);:}

    |tiposvar CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable
PARRIGHT ; {:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = typevariable {:idvariables.add(a);:}
```

```
    |CARACTERESVARIABLES:a = typevariable ; {:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
{:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT ;
{:idvariables.add(a);:}

;

recibirparametros::=

    recibirparametros:paramlist , typevariable:param
{:RESULT=paramlist;RESULT.add(String.valueOf(param));:}

    |typevariable:a {:RESULT= new
ArrayList<>();RESULT.add(String.valueOf(a));:}

;

relacionconosinoperad||::=

    recibirrelacion && relacionconosinoperad||

    |recibirrelacion || relacionconosinoperad||

    |recibirrelacion

;


recibirrelacion::=

    !( recibirrelacion PARRIGHT

    |typevariable = = typevariable

    |typevariable ! = typevariable

    |typevariable < typevariable

    |typevariable > typevariable

    |typevariable < = typevariable

    |typevariable > = typevariable


;

operad||esrelacionales::=

    = =

    |! =

    |<
```

```
    |>
    |< =
    |> =
;
operad||esaritmeticos::=
    +:a {:RESULT=a;:}
    |-:a {:RESULT=a;:}
    |P||:a {:RESULT=a;:}
    |/:a {:RESULT=a;:}
    |**:a {:RESULT=a;:}
    |%:a {:RESULT=a;:}
;
operad||eslogicos::=
    &&
    |||
    |!
;
typevariable::=
    CARACTERESVARIABLES + typevariable
    |CADCOM + typevariable
    |CADCOMSIM + typevariable
    |DECIMAL + typevariable
    |NUMBER + typevariable

    |CARACTERESVARIABLES
    |CADCOM
    |CADCOMSIM
    |DECIMAL
    |NUMBER
;
```

```
variablesrecursivas::=
    variables variablesrecursivas
    |sentenciaif variablesrecursivas
    |sentenciaf|| variablesrecursivas
    |sentenciawhile variablesrecursivas
    |sentenciaswitch variablesrecursivas
    |consolelog variablesrecursivas
    |llamada variablesrecursivas
    |variables
    |sentenciaif
    |sentenciaf||
    |sentenciawhile
    |sentenciaswitch
    |llamada
    |consolelog
;

variables::=
    VAR CARACTERESVARIABLES:a = typevariable ; {:idvariables.add(a);:}
    |VAR CARACTERESVARIABLES:a = typevariable {:idvariables.add(a);:}
    |VAR CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
{:idvariables.add(a);:}
    |VAR CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
; {:idvariables.add(a);:}
    |VAR CARACTERESVARIABLES:a ; {:idvariables.add(a);:}
    |VAR CARACTERESVARIABLES:a {:idvariables.add(a);:}

    |LET CARACTERESVARIABLES:a = typevariable ; {:idvariables.add(a);:}
    |LET CARACTERESVARIABLES:a = typevariable {:idvariables.add(a);:}
    |LET CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
{:idvariables.add(a);:}
```

```
    |LET CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
; {:idvariables.add(a);:}

    |LET CARACTERESVARIABLES:a ; {:idvariables.add(a);:}

    |LET CARACTERESVARIABLES:a {:idvariables.add(a);:}


    |CONST CARACTERESVARIABLES:a = typevariable ;
{:idvariables.add(a);:}

    |CONST CARACTERESVARIABLES:a = typevariable {:idvariables.add(a);:}

    |CONST CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable
PARRIGHT {:idvariables.add(a);:}

    |CONST CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable
PARRIGHT ; {:idvariables.add(a);:}

    |CONST CARACTERESVARIABLES:a ; {:idvariables.add(a);:}

    |CONST CARACTERESVARIABLES:a {:idvariables.add(a);:}


    |CARACTERESVARIABLES aumentodecremento

    |CARACTERESVARIABLES aumentodecremento ;


    |CARACTERESVARIABLES:a = typevariable {:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = typevariable ; {:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT
{:idvariables.add(a);:}

    |CARACTERESVARIABLES:a = REQUIRE PARLEFT typevariable PARRIGHT ;
{:idvariables.add(a);:}

    |err||

;

tiposvar::=

    VAR

    |LET

    |CONST

;


sentenciaif::=
```

```
    IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas }

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas
BREAK ; }

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { BREAK ; }


    |IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas
} sentenciaelse

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas
BREAK ; } sentenciaelse

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { BREAK ; }
sentenciaelse


    |IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas
} sentenciaelseif

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { variablesrecursivas
BREAK ; } sentenciaelseif

    |IF PARLEFT relacionconosinoperad|| PARRIGHT { BREAK ; }
sentenciaelseif


;


sentenciaelse::=

    ELSE { variablesrecursivas }

    |ELSE { variablesrecursivas BREAK ; }

    |ELSE { BREAK ; }

;


sentenciaelseif::=

    ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT {
variablesrecursivas } sentenciaelseif

    |ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT {
variablesrecursivas BREAK ; } sentenciaelseif

    |ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT { BREAK ; }
sentenciaelseif
```

```
    |ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT {
variablesrecursivas }

    |ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT {
variablesrecursivas BREAK ; }

    |ELSE IF PARLEFT relacionconosinoperad|| PARRIGHT { BREAK ; }

;

sentenciaf||::=

    F|| PARLEFT instruccionf|| PARRIGHT { variablesrecursivas }

    |F|| PARLEFT instruccionf|| PARRIGHT { variablesrecursivas BREAK ;
}

;

instruccionf||::=

    tiposvar CARACTERESVARIABLES = oprecursivas ;

     CARACTERESVARIABLES operad||esrelacionales oprecursivas ;

     CARACTERESVARIABLES aumentodecremento

     |CARACTERESVARIABLES = oprecursivas ;

     CARACTERESVARIABLES operad||esrelacionales oprecursivas ;

     CARACTERESVARIABLES aumentodecremento

;

aumentodecremento::=

    + +

    |- -

;



sentenciawhile::=

        WHILE PARLEFT CARACTERESVARIABLES operad||esrelacionales
oprecursivas PARRIGHT { variablesrecursivas }

        | WHILE PARLEFT CARACTERESVARIABLES operad||esrelacionales
oprecursivas PARRIGHT { variablesrecursivas BREAK ; }
```

```
        |sentenciadowhile WHILE PARLEFT CARACTERESVARIABLES
operad||esrelacionales oprecursivas PARRIGHT ;

        |sentenciadowhile WHILE PARLEFT CARACTERESVARIABLES
operad||esrelacionales oprecursivas PARRIGHT


        |WHILE PARLEFT !( CARACTERESVARIABLES operad||esrelacionales
oprecursivas PARRIGHT PARRIGHT { variablesrecursivas }

        |WHILE PARLEFT !( CARACTERESVARIABLES operad||esrelacionales
oprecursivas PARRIGHT PARRIGHT { variablesrecursivas BREAK ; }


        |sentenciadowhile WHILE PARLEFT !( CARACTERESVARIABLES
operad||esrelacionales oprecursivas PARRIGHT PARRIGHT ;

        |sentenciadowhile WHILE PARLEFT !( CARACTERESVARIABLES
operad||esrelacionales oprecursivas PARRIGHT PARRIGHT

;


sentenciadowhile::=

    DO { variablesrecursivas }

    |DO { variablesrecursivas BREAK ; }

;


sentenciaswitch::=

    SWITCH PARLEFT CARACTERESVARIABLES PARRIGHT { recursivecase DEFAULT
: variablesrecursivas }

    |SWITCH PARLEFT CARACTERESVARIABLES PARRIGHT { recursivecase
DEFAULT : }

;

recursivecase::=

    caseswitch recursivecase

    |caseswitch

;


caseswitch::=
```

```
    CASE typenumberovariable : variablesrecursivas BREAK ;
    |CASE typenumberovariable : BREAK ;
;
oprecursivas::=
    oprecursivas typenumberovariable operad||esaritmeticos
    |typenumberovariable operad||esaritmeticos
    |oprecursivas typenumberovariable
    |typenumberovariable
    |err||
;

consolelog::=
    PRINT PARLEFT recursiveprints:a PARRIGHT ;
//{:Interfaz.Consolelog(a);:}
    |PRINT PARLEFT recursiveprints:a PARRIGHT
//{:Interfaz.Consolelog(a);:}
;
recursiveprints::=
    prints:exp recursiveprints {:RESULT=exp;:}
    |prints:exp {:RESULT=exp;:}
;

prints::=
    NUMBER:a {:RESULT=a;:}
    |DECIMAL:a {:RESULT=a;:}
    |CARACTERESVARIABLES:a {:RESULT=a;:}
    |CADCOM:a {:RESULT=a;:}
    |operad||esaritmeticos:a {:RESULT=a;:}

;
```

```
typenumberovariable::=
    NUMBER:a {:RESULT=a;:}
    |DECIMAL:a {:RESULT=a;:}
    |CARACTERESVARIABLES:a {:RESULT=a;:}

;
```