UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ORGANIZACION DE LENGUAJES Y COMPILADORES 1
SECCIÓN B
ING. MANUEL HAROLDO
AUX. KEVIN ALFREDO LÓPEZ RODRÍGUEZ
SEGUNDO SEMESTRE 2023

B

PROYECTO No. 1

GRAMÁTICA

ANALIZADOR JSON Y STATPY

Nombre: Julio Alejandro Zaldaña Ríos

Carné: 202110206

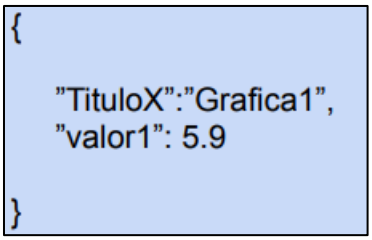
Guatemala, 20 de septiembre de 2023

ÍNDICE

GRAMÁTICAS PARA ARCHIVOS JSON

A continuación, se presentarán las gramáticas de análisis sintáctico trabajadas para la aceptación de los archivos JSON.

Ya que tienen una estructura parecida a la siguiente, se optó por trabajar de la siguiente forma:



$< inicio >$ ::= $LLAV\_A$ $< elementos >$ $LLAV\_C$

$< elementos >$ ::= $< elemento >$

| $< elementos >$ $COMA$ $< elemento >$;

$< elemento >$ ::= $STRING\ DOSPTS\ DOUBLE$

| $STRING\ DOSPTS\ STRING$

En el no terminal <elementos> se realiza una recursividad, para que se añadan más no terminales <elemento> seguidas de coma, con la estructura donde primero viene un String : String o String : Decimal/Double.

GRAMÁTICAS ARCHIVOS STATPY

A continuación, se presentarán las gramáticas de análisis sintáctico más importantes para el análisis y la traducción de los archivos StatPy.

Se necesita un comenzar con sentencias, donde vendrán la estructura del “main” que es la estructura principal de un archivo StatPy.

$< INICIO >::= < SENTENCIAS >$

$< SENTENCIAS >::= < SENTENCIAS >$

$| < S >$

$| comentario$

$| comentario2$

Estructura básica del main

```
void main ( ){
   <Sentencias>
}
```

$< S > :: = VOI\ \ MAIN\ \ PAR\_A\ \ PAR\_C\ LLAV\_A\ \ < CUERPO > \ LLAV\_C\ < COMENTARIOS >$

Notar que lleva un cuerpo, donde irán todas las sentencias, este se explica a continuación, es el que conectará a todos los funcionamientos, como sentencias de control, ciclos, switch, funciones estadísticas etc.

Para aceptar comentarios

$< COMENTARIOS >::= \ comentario$

$| comentario2$

|

Estructura del cuerpo de un main

Aquí se define la mayoría de no terminales, que abrirá paso para poder aceptar estructuras de control, ciclos, switch, declaración de variables y aceptación de la estructura de funciones estadísticas.

$< CUERPO > ::= < CUERPO >< C >$

$| < C >$

$< C > ::= < TIPO > \ ID \ \ PYCOMA$

$| < TIPO > ID \ ASIGNA < EXPRESION > PYCOMA$

$| \ < TIPO > \ ID \ ASIGNA < \ OPERACION > \ PYCOMA$

$|CONSOLE \ PUNTO \ WRITE \ PAR\_A \ < VARPRINT > \ PAR\_C \ PYCOMA$

$| IFSTATE$

$| ELIFSTATE$

$| SWITCHSTATE$

$| FORSTATE \ \ DOWHILESTATE$

$| WHILESTATE$

$| RGB\_STATE$

$| RGP\_STATE$

$| RDG\_STATE$

$| comentario$

$| comentario2 : com2$

Aceptación de palabras reservadas

$< TIPO >::= INT1$

$| CHAR1$

$| DOUBLE1$

$| BOOL1$

$| STRING1$

Vendrán como palabras reservadas, para la declaración de variables en StatPy, ya sea int,char,double,bool y string.

Sentencias IF – ELSE IF

Se considera importante el <IFSTATE> para acceder a la estructura de un IF.

$< IFSTATE >::= IF\ PAR\_A\ < IFARG >$

Al igual que <ELIFSTATE> para acceder a la estructura de un ELSE IF.

$< ELIFSTATE >::= ELSEIF\ PAR\_A\ < ELIFARG >$

<IFARG> es donde se define el argumento de un IF, donde se declaran operaciones.

$< IFARG >::= < OPERIFS >\ PAR\_C\ LLAV\_A\ < CUERPOIF >\ LLAV\_C$

$| ID\ PAR\_C\ LLAV\_A\ < CUERPOIF >\ LLAV\_C$

El <CUERPOIF> es donde se define la estructura y el cuerpo de un IF.

$< ELIFARG >::= < OPERIFS >\ PAR\_C\ LLAV\_A\ < CUERPOELIF >\ LLAV\_C$

$| ID: PAR\_C\ LLAV\_A\ CUERPOELIF\ LLAV\_C$

<OPERIFS> define todas las operaciones posibles dentro de un IF STATEMENT.

$< OPERIFS >:: = \ OP\ SIGNO\ OP\ \ < TERMINO >$

$\quad | \ OP\ SIGNO\ OP$

$\quad | \ NOT\ OP$

$\quad | \ NOT\ OP\ TERMINO$

$\quad | \ SUPERNOT\ NOT\ OP$

$\quad | \ SUPERNOT\ NOT\ OP\ TERMINO:$

$SUPERNOT :: = \ SUPERNOT\ NOT$

$\quad | \ NOT$

$< CUERPOIF > = \ CONSOLE\ PUNTO\ WRITE\ PAR_A < VARPRINT > \ PAR\_C\ PYCOMA\ MOREPRINTS$

$\quad | \ CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA$

$\quad | \ MOREIFS\ MORELIFS$

$\quad | \ MOREIFS$

$< CUERPOELIF >:: = \ CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT\ PAR\_C\ PYCOMA\ \ MOREPRINTS$

$\quad | \ CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT\ PAR\_C\ PYCOMA$

$\quad | \ MOREIFS$

$\quad | \ MORELIFS$

$\quad | \ CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT\ PAR\_C\ PYCOMA\ INT1\ ID\ PYCOMA\ WHILEDOWHIL$

Para IFS Anidados:

$< MOREIFS >::= < MOREIFS > \ IF: PAR\_A \ IF < ARGIF >$

$\quad | \ IF \ PAR\_A \ < IFARGIF >$

$< IFARGIF >::= \ OPERIFS \ PAR\_C \ LLAV\_A \ IFCUERPOIF \ LLAV\_C$

$\quad | \ ID \ PAR\_C \ LLAV\_A \ IFCUERPOIF \ LLAV\_C$

Cuerpo para ifs anidados:

$< IFCUERPOIF >::= \ CONSOLE \ PUNTO \ WRITE \ PAR\_A \ VARPRINT \ PAR\_C \ PYCOMA \ \ MOREPRINTS$

$\quad | \ CONSOLE \ PUNTO \ WRITE \ PAR\_A \ VARPRINT \ PAR\_C \ PYCOMA$

$\quad | \ MOREIFS$

$\quad | \ CONSOLE \ PUNTO \ WRITE \ PAR\_A \ VARPRINT \ PAR\_C \ PYCOMA \ WHILESTATEIF$

Lo mismo pero para else if statements anidados:

$MORELIFS ::= \ MORELIFS: morelifs \ ELSEIF \ PAR\_A \ IFARGELIF: elsearg$

$\quad | \ ELSEIF \ PAR\_A \ IFARGELIF: elsearg$

$IFARGELIF ::= \ OPERIFS: oper \ PAR\_C \ LLAV\_A \ CUERPOELIF: cuerpoelif \ LLAV\_C$

$\quad | \ ID: id \ PAR\_C \ LLAV\_A \ CUERPOELIF: cuerpoelif \ LLAV\_C$

Aceptación de sentencia imprimir

$< VARPRINT >::= CADENA\ SUMA\ ID$

$| CADENA$

$| CADENA\ SUMA\ ENTERO$

$| ID : id$

Para mas sentencias de impresión

$< MOREPRINTS >::= < MOREPRINTS > CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT\ PAR\_C\ PYCOMA$

$| CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT\ PAR\_C\ PYCOMA$

Declaración de variables:

$< EXPRESION >::= ENTERO$

$| DOUBLE$

$| ID$

$| COMILLA\_S\ ID\ COMILLA\_S$

$| COMILLA\_S\ ENTERO : COMILLA\_S :$

Para operaciones aritméticas, relacionales y lógicas:

$OPERACION ::= OP\ SIGNO\ OP\ TERMINO$

$| OP\ SIGNO\ OP$

$| NOT\ OP$

Tipos de operadores que se aceptan:

$< OP >::= ID$

$| ENTERO: nume$

$| DOUBLE$

$| CADENA$

$| DECIMAL$

Signos de las operaciones:

$SIGNO ::= SUMA$

$| RESTA$

$| MULT$

$| DIV$

$| MAYOR$

$| MENOR$

$| MAYOR\_I$

$| MENOR\_I$

$| IGUAL$

$| DIST$

$| AND$

$| OR:$

$| NOT$

Operaciones Recursivas

$< TERMINO >::= < TERMINO > \ SIGNO \ \ OP:$

$| TERMINO: term \ SIGNO: signo \ SUPERNOT: not1 \ OP: op1$

$| SIGNO: signo2 \ OP: op2$

$| SIGNO: signo3 \ SUPERNOT: not \ OP: op3$

Para Sentencias Switch:

$SWITCHSTATE ::$

$= SWITCH: swit \ PAR\_A \ ID: id \ PAR\_C \ LLAV\_A \ CASE: case1 \ ENTERO: enu1 \ DOSP \ ID: id2 \ ASIGNA \ ENTERO: enu2 \ PYCOMA$

$|SWITCH: swit \ PAR\_A \ ID: id \ PAR\_C \ LLAV\_A \ CASE: case1 \ ENTERO: enu1 \ DOSP \ ID: id2 \ ASIGNA \ ENTERO: enu2 \ PYCOMA$

$SWITCHIF ::= MOREIFS: ifstate1 \ MORELIFS: elif \ CASE1: case1$

$CASE1 ::$

$= BREAK \ PYCOMA \ CASE \ ENTERO: enu2 \ DOSP \ ID: id3 \ ASIGNA \ ENTERO: enu3 \ PYCOMA \ SWITCHIF2: switchif$

$| BREAK \ PYCOMA \ CASE \ ENTERO: enu2 \ DOSP \ ID: id3 \ ASIGNA \ ENTERO: enu3 \ PYCOMA \ CASE2: case2$

$SWITCHIF2 ::= MOREIFS: ifstate2 \ MORELIFS: elif \ CASE2: case2$

$CASE2 ::$

$= CASE \ ENTERO: enu3 \ DOSP \ ID: id4 \ ASIGNA \ ENTERO: enu4 \ PYCOMA \ SWITCHIF3: switchif \ LLAV\_C$

$SWITCHIF3 ::= MOREIFS: ifstate3\ MORELIFS: elif\ CASED: cased$

$CASED ::$
$= DEFAULT\ DOSP\ CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA$

Para Ciclos For:

$FORSTATE ::$
$= FOR: fors\ PAR\_A\ TIPO\ ID: id\ ASIGNA\ ENTERO\ PYCOMA\ OPERFOR: operfor\ PYCOMA\ ID\ SUMA\ S$

Cuerpo de un ciclo for:

$CUERPOFOR ::$
$= CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA\ MOREPRINTS: print2$

$| CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA$

$| MOREFORS: morefors$

$| CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA\ MOREFORS: morefors$

Para más ciclos for en el cuerpo de un for:

$MOREFORS ::$
$= MOREFORS\ FOR: fors\ PAR\_A\ TIPO\ ID: id\ ASIGNA\ ENTERO\ PYCOMA\ OPERFOR: operfor\ PYCOM$

$| FOR: fors\ PAR\_A\ TIPO\ ID: id\ ASIGNA\ ENTERO\ PYCOMA\ OPERFOR: operfor\ PYCOMA\ ID\ SUMA$

Operaciones de un for

$OPERFOR ::= OP: op1\ SIGNO: signo\ OP: op2$

Para Ciclos While y Do While:

$WHILESTATE$ ::

$= WHILE: whiles\ PAR\_A\ OPERIFS: operifs\ PAR\_C\ LLAV\_A\ CUERPOWHILE: cuerpowhile\ ID: id1\ AS$

$\mid WHILE: whiles\ PAR\_A\ OPERIFS: operifs\ PAR\_C\ LLAV\_A\ CUERPOWHILE: cuerpowhile\ PYCOMA$

$WHILEDOWHILE$ ::

$= WHILE: whiles\ PAR\_A\ OPERIFS: operifs\ PAR\_C\ LLAV\_A\ CUERPOWHILE: cuerpowhile\ LLAV\_C$

Cuerpo de un ciclo while:

$CUERPOWHILE$ ::

$= CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA\ MOREPRINTS: print2$

$\mid CONSOLE\ PUNTO\ WRITE\ PAR\_A\ VARPRINT: varprint\ PAR\_C\ PYCOMA$

$\mid MOREIFS: moreifs$

Un while con estados if dentro en el cuerpo:

$WHILESTATEIF$ ::

$= WHILE: whiles\ PAR\_A\ OPERIFS: operifs\ PAR\_C\ LLAV\_A\ CUERPOWHILE: cuerpowhile\ ID: id1\ AS$

$\mid WHILE: whiles\ PAR\_A\ OPERIFS: operifs\ PAR\_C\ LLAV\_A\ CUERPOWHILE: cuerpowhile\ PYCOMA$

Para funciones estadísticas:

Para definer graficas de barra

$$RGB\_STATE ::= RGB\ PAR\_A\ PAR\_C\ LLAV\_A\ INSTRUCCIONESFUNC\ LLAV\_C$$

Para definir graficas de pie

$$RGP\_STATE ::= RGP\ PAR\_A\ \ PAR\_C\ LLAV\_A\ INSTRUCCIONESFUNC\ LLAV\_C$$

Para definir globales

$$RDG\_STATE ::= RDG\ PAR\_A\ PAR\_C\ LLAV\_A\ INSTRUCCIONESFUNC\ LLAV\_C$$

Para que sea recursivo, los valores y atributos para graficar:

$$INSTRUCCIONESFUNC ::= INSTRUCCIONESFUNC\ TIPO\ INSTRUCCIONFUNC$$

$$| TIPO\ INSTRUCCIONFUNC$$

$$INSTRUCCIONFUNC ::= RTitulo\ ASIGNA\ CADENA: aPYCOMA$$

$$| RTitulo\ ASIGNA\ ID: a\ PYCOMA$$

$$| RTitulo\ ASIGNA\ DOLAR\ LLAV\_A\ NEWVALOR\ COMA\ CADENA\ COMA\ CADENA: clav$$

$$| CORCH\_A\ CORCH\_C\ REjex\ ASIGNA\ LLAV\_A\ LISTAEJEX\ LLAV\_C\ PYCOMA$$

$$| CORCH\_A\ CORCH\_C\ RValores\ ASIGNA\ LLAV\_A\ LISTAVALORES\ \ LLAV\_C\ PYCOMA$$

$$| RTituloX\ ASIGNA\ CADENA: a\ PYCOMA$$

$$| RTituloX\ ASIGNA\ ID: a\ PYCOMA$$

$$| RTituloY\ ASIGNA\ CADENA: a\ PYCOMA$$

$$| RTituloY\ ASIGNA\ ID: a\ PYCOMA$$

$$| RTituloY\ ASIGNA\ DOLAR\ LLAV\_A\ NEWVALOR\ COMA\ CADENA\ COMA\ CADENA: cla$$

$$| ID: a\ ASIGNA\ CADENA: b \quad \{: ts.add(a, b); :\} PYCOMA$$

$$| ID: a\ ASIGNA\ DOLAR\ LLAV\_A\ NEWVALOR\ COMA\ CADENA\ COMA\ CADENA: clavejs$$

$$PYCOMA$$

$$| ID: a\ ASIGNA\ DOUBLE: b\ PYCOMA$$

$$LISTAEJEX ::= LISTAEJEX\ COMA\ ELEMENTOSEJEX: a$$

$$| ELEMENTOSEJEX: a$$

$$LISTAVALORES ::= LISTAVALORES\ COMA\ ELEMENTOSVALORES: a\ \{:$$

$$|ELEMENTOSVALORES: a$$

$$ELEMENTOSEJEX ::= CADENA: a\ \{: RESULT = a; :\}$$

$$| ID: a\ \{:\ RESULT = ts.get(a);\ :\}$$

$$| DOLAR\ LLAV\_A\ NEWVALOR\ COMA\ CADENA\ COMA\ CADENA: clavejson\ LLAV\_C$$

$$ELEMENTOSVALORES ::= DOUBLE: a$$

$$| ID: a$$

$$| DOLAR\ LLAV\_A\ NEWVALOR\ COMA\ CADENA\ COMA\ CADENA: clavejson\ LLAV\_C$$