```
//Parsear el HTML
PROGRAM -> |WEBCODE

WEBCODE -> |CCODE
           |HTMLCODE

HTMLCODE -> |HTMLTOKEN WEBCODE

CCODE ->| BeginCcode ListOfSentences EndCcode WEBCODE
        | Epsilon

//GRAMATICA PARA EL CCODE
ListOfSentences ->| Sentence ListOfSentences
                  | Epsilon

ListOfSpecialSentences -> SpecialSentence ListOfSpecialSentences
                          |Epsilon
//No permitimos declarar struct dentro de struct ni función dentro de struct
SpecialSentence   -> IF
                     | WHILE
                     | DO
                     | FORLOOP
                    | AssignmentOrFunctionCall
                    | CONST
                    | SWITCH
                    | SPECIALDECLARATION
                     | BREAK
                     | CONTINUE
                     | INCLUDE
                     | ENUM
                     | Return

Sentence -> | DECLARATION
            | IF
            | WHILE
            | DO
            | FORLOOP   //PODRIA SER UN FOR NORMAL O FOREACH
            | SWITCH
            | AssignmentOrFunctionCall
            | STRUCT
            | CONST
```

```
            | BREAK
            | CONTINUE
            | INCLUDE
            | ENUM


INCLUDE->#include Identifier;

CONTINUE -> continue ;

Return -> return EXPRESSION ;

AssignmentOrFunctionCall ->  IdentifierValueForPreId EOSTOKEN

ValueForPreId-> = EXPRESSION
              | += EXPRESSION
              | -= EXPRESSION
              |  CallFunction

//DataTypes va a ser los tipos como int, float, date, string
SPECIALDECLARATION ->GeneralDeclaration TypeOfDeclarationForFunction

TypeOfDeclarationForFunction-> ValueForId MultiDeclaration
                             | IsArrayDeclaration
                             | ;

GeneralDeclaration - > Datatype IsPointer Identifier

DECLARATION -> GeneralDeclaration TypeOfDeclaration

TypeOfDeclaration-> ValueForId MultiDeclaration
                  | IsArrayDeclaration
                  | IsFunctionDeclaration
                  |;
MultiDeclaration-> OptionalId EOSTOKEN

IsArrayDeclaration -> [SizeForArray] BidArray OptionalInitOfArray EOSTOKEN

BidArray -> [ SizeBidArray ]
           | Epsilon
```

```
//int myArray[10] = { 5, 5, 5, 5, 5, 5, 5, 5, 5, 5 };
OptionalInitOfArray -> = { ListOfExpressions }

ListOfExpressions-> EXPRESSION OptionalExpression

OptionalExpression-> , ListOfExpressions
                   | Epsilon

//Para tipos de declaración double sumar(), y; int x [5]
// la forma general de declarar una función es
//return_type function_name( parameter list ) {
//   body of the function
//}

IsFunctionDeclaration-> ( ParameterList ){ ListOfSpecialSentences }

//Después de saber el tipo, si es o no un puntero, podemos recibir más Id para
declaraciones múltiples en una sola línea o bien la inicialización de lo declarado
//Esta es para las múltiples apuntadores
IsPointer-> * IsPointer
            |Epsilon

//El size puede ser un id(variable) o un número cualquiera
SizeForArray -> |Identifier
                |LiteralNumber
                |LiteralOctal
                |LiteralHexadecimal
                |Epsilon

SizeBidArray -> Identifier
                |LiteralNumber
                |LiteralOctal
                |LiteralHexadecimal


//Para casos como  int d = 3, f = 5;  debe estar separados por coma y con valor de
asignación
OptionalId ->| , ListOfId
             | Epsilon

//Múltiples elementos en la declaración, loop.
ListOfId -> Identifier OtherIdOrValue
```

```
OtherIdOrValue -> ValueForId OptionalId
                  | Epsilon

//Asignación o inicialización
ValueForId-> = EXPRESSION
              | Epsilon


//Los tipos de datos que soportamos
DataType -> |int
            |float
            |char
            |bool
            |string
            |date

//Cuando una definicion de funcion lleva parametros int suma(int num1,int num2)
ParameterList ->| DataType CHOOSE_ID_TYPE OptionaListOfParams
                | Epsilon
//El optional es para permitir más de un parámetro, significa que acepta uno o más
parámetros
OptionaListOfParams ->| , DataType CHOOSE_ID_TYPE OptionaListOfParams
                      |Epsilon

CHOOSE_ID_TYPE -> & Identifier
                 | * IsPointer Identifier
                 |Identifier


     ************************************************************************
EXPRESSION
************************************************************************
EXPRESSION -> RelationalExpression

RelationalExpression -> ExpressionAdicion RelationalExpression'

RelationalExpression' -> RelationalOperators ExpressionAdicion RelationalExpression'
            | Epsilon

ExpressionAdicion -> ExpressionMul ExpressionAdicion'

ExpressionAdicion' -> AdditiveOperators ExpressionMul ExpressionAdicion'
                   |Epsilon
```

```
ExpressionMul -> ExpressionUnary ExpressionMul'

ExpressionMul' -> MultiplicativeOperators ExpressionUnary ExpressionMul'
                | Epsilon
              factor

ExpressionUnary -> UnaryOperators Factor
                 | Factor

//A factor can be just an Identifier, we can call a function or access a value stored in an
array
Factor -> | Identifier FactorFunctionOrArray
          | LiteralNumber
          | LiteralString
          | LiteralBoolean
          | LiteralDate
          | LiteralChar
          | ( EXPRESSION )

OptionalIncrementOrDecrement -> ++
                              | --

FactorFunctionOrArray ->  CallFunction
                   | IndexOrArrowAccess
                   | Epsilon

IndexOrArrowAccess -> [EXPRESSION] IndexOrArrowAccess
                   |ArrowOrDot Identifier IndexOrArrowAccess
                   |Epsilon

ArrowOrDot -> ->
                | .
RelationalOperators ->| <
                      | <=
                      | >
                      | >=
                      | &&
                      | ||
                      | >>
                      | <<
```

```
                        | ==
                        | !=
                        | +=
                        | -=
                        | *=
                        | /=
                        | %=
                        | &
                        | ^=
                        | |=
AdditiveOperators ->| +
                      | -

MultiplicativeOperators ->| *
                            | /
                            | %

UnaryOperators ->|~
                 | ++
                 | --
                 | &
                 | |
                 | ^

//This one is used when we have this
CallFunction-> ( ListOfExpressions )

ListOfExpressions-> EXPRESSION  OptionalListOfExpressions
                   |Epsilon

OptionalListOfExpresssions-> , ListOfExpressions
                           | Epsilon

**************************************************************************
IF SENTENCE
**************************************************************************

//La producción BLOCK puede producir solo un statement, de lo contrario, si son más
requiere Brackets
```

```
IF -> if( EXPRESION) BlOCKFORIF ELSE

BlOCKFORIF -> Sentence
            | { ListOfSentences }

BlOCKFORLOOP ->{ ListOfSentences }


ELSE -> else BlOCKFORIF
        | Epsilon
```

****

WHILE SENTENCE

```
while ( expression )
{
   Single statement
   or
   Block of statements;
}
```

****

```
WHILE -> while (EXPRESSION) BlOCKFORLOOP
```

****

SENTENCE DO

```
do
{
   Single statement
   or
   Block of statements;
}while(expression);
```

****

```
DO -> do BlOCKFORLOOP while ( EXPRESSION ) EOSTOKEN
```

****

SENTENCE FOR

```
for( expression1; expression2;
expression3)
{
   Single statement
   or
   Block of statements;
}
```

SENTENCE FOREACH

```
for (String item : someList) {
       System.out.println(item);
```

```
}

*************************************************************************

FORLOOP -> for ( ForOrForEach

ForOrForEach->  DataType Identifier : Identifier ) BlOCKFORLOOP
               | EXPRESSION ; EXPRESSION ; EXPRESSION ) BlOCKFORLOOP


*************************************************************************
SENTENCE SWITCH
switch( expression )
     {
        case constant-expression1:      statements1;
        [case constant-expression2:     statements2;]
        [case constant-expression3:     statements3;]
        [default : statements4;]
     }
*************************************************************************

SWITCH -> switch ( EXPRESSION ) { ListOfCase }

ListOfCase -> |CASE ListOfCase
              |DefaultCase
              |Epsilon

CASE -> case EXPRESSION: ListOfSpecialSentences BREAK EOSTOKEN

DefaultCase-> default : ListOfSpecialSentences EOSTOKEN

BREAK -> break;
         |Epsilon

*************************************************************************
SENTENCE STRUCT
struct [structure tag] {
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

```
**************************************************************************

STRUCT -> struct Identifier { MembersList } EOSTOKEN

MembersList -> DeclarationOfStruct

DeclarationOfStruct -> GeneralDeclaration ArrayIdentifier EOSTOKEN OptionalMember

OptionalMember -> DeclarationOfStruct
             | Epsilon

ArrayIdentifier-> [ SizeForArray ] BidArray
                  | Epsilon

//No puede ser declaration porque abarcaría declaración de funciones también.

**************************************************************************
CONST
const type variable = value;
**************************************************************************

CONST-> const DataType Identifier = EXPRESSION EOSTOKEN

**************************************************************************
ENUMS
enum-specifier:
enum identifier opt { enumerator-list }
enum identifier

**************************************************************************

ENUM -> enum Identifier { EnumeratorList } EOSTOKEN

EnumeratorList -> EnumItem OptionalEnumItem
                | Epsilon

OptionalEnumItem -> , EnumeratorList

EnumItem -> Identifier OptionalIndexPosition

//Este es Opcional Index, es para cambiar auto-enumerado de los enums
```

```
OptionalIndexPosition -> = LiteralNumber
                      | Epsilon

PostfixOperators -> | ()
                    | []
                    | ->
                    | .
                    | ++
                    | --


MultiplicativeOperators ->| *
                          | /
                          | %

AdditiveOperators ->| +
                    | -

ShiftOperators ->| <<
                 | >>

RelationalOperators ->| <
                      | <=
                      | >
                      | >=

EqualityOperators -> | ==
                     | !=

BiwiseAND -> | &

BitwiseXOR ->| ^

BitwIseOR ->| |

LogicalAND ->| &&
LogicalOR -> | ||

Conditional -> ?:
```

```
Assignment ->| =
             | +=
             | -=
             | *=
             | /=
             | %=
             | >>=
             | <<=
             | &=
             | ^=
             | |=
```

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left

Comma	,	Left to right