Definiendo reglas sintácticas

Para el análisis sintáctico, las **terminales** serán los tokens generados en el análisis lexicográfico. Regularmente se escriben con inicio en mayúscula para diferenciarlos de los no terminales.

Por ejemplo:

```
terminal Comillas, T_dato, Cadena, If, Else, Do, While, For,
    Igual, Suma, Resta, Multiplicacion, Division, Op_logico, Op_relacional,
    Op_atribucion, Op_incremento, Op_booleano, Parentesis_a, Parentesis_c,
    Llave_a, Llave_c, Corchete_a, Corchete_c, Main, P_coma, Identificador, Frase
    Numero, ERROR;
```

Los **no terminales** son los nombres para los axiomas que permiten generar las producciones. Se deberán de construir de acuerdo con las producciones que requiera el lenguaje. Regularmente se escriben con mayúsculas para diferenciarlos de los terminales.

Por ejemplo:

```
non terminal INICIO, SENTENCIA, DECLARACION, DECLARACION_FOR, IF, IF_ELSE,
    WHILE, DO_WHILE, FOR, SENTENCIA_BOOLEANA, SENTENCIA_FOR;
```

Uno de los no terminales señalará el **inicio** de la secuencia para el análisis sintáctico.

```
start with INICIO;
```

Actividad

Instrucciones: Genera las siguiente producciones o reglas sintácticas, en su caso la explicación correspondiente. Para indicar que se terminó el axioma usa un punto y coma. Respeta el uso de mayúsculas y minúsculas conforme a los terminales y no terminales definidos previamente.

Ejemplo

```
INICIO ::=
    T_dato Main Parentesis_a Parentesis_c Llave_a SENTENCIA Llave_c |
    Main Parentesis_a Parentesis_c Llave_a SENTENCIA Llave_c
;
```

Ejercicios

//1. La sentencia puede contener una estructura antecedida de una sentencia o sin sentencia. Las estructuras para considerar son: declaración, if, if_else, while, do_while, for

```
SENTENCIA ::=
        SENTENCIA DECLARACION | DECLARACION |
        SENTENCIA IF | IF |
        SENTENCIA IF_ELSE | IF_ELSE |
        SENTENCIA WHILE | WHILE |
        SENTENCIA DO_WHILE | DO_WHILE |
        SENTENCIA FOR | FOR
;
```

//2. La declaración puede ser definida por tipo de dato o cadena, después debe llevar el identificador, al final puede o no llevar una asignación a un número a una frase según sea el caso.

```
DECLARACION ::=
        T_DATO IDENTIFICADOR Op_atribucion NUMERO |
        T_DATO IDENTIFICADOR |
        CADENA IDENTIFICADOR Op_ atribucion FRASE |
        CADENA IDENTIFICADOR
;
```

//3. El if debe contener una sentencia booleana entre paréntesis y una sentencia entre llaves.

```
IF ::=
        Parentesis_a  SENTENCIA BOOLEANA Parentesis_c Llave_a SENTENCIA Llave_c
```

```
;
```

//4. La sentencia booleana debe contener un operador booleando o un identificador seguido de un operador relacional con un operador booleano o un número o un identificador o comillas más identificador más comillas o comillas más comillas

```
SENTENCIA_BOOLEANA ::=
    Op_booleano |
    Identificador Op_relacional Op_booleano |
    Identificador Op_relacional Numero |
    Identificador Op_relacional Identificador |
    Identificador Op_relacional Comillas Comillas |
    Identificador Op_relacional Comillas Identificador Comillas
;
```

//5. Es parecido al if, pero incluye la parte del else

```
IF_ELSE ::=
        If Paréntesis_a SENTENCIA_BOOLEANA Paréntesis_c
        Llave_a SENTENCIA Llave_c
        Else Llave_a SENTECIA Llave_c
;
```

//6. Utilizar la estructura de C++ para while

```
WHILE ::=
        While Paréntesis_a SENTENCIA_BOOLEANA Paréntesis_c
        Llave_a SENTENCIA  Llave_c
;
```

//7. Utilizar la estructura de C++ para do while

```
DO_WHILE ::=
        Do Llave_a SENTENCIA Llave_c
        While Paréntesis_a SENTENCIA_BOOLEANA Paréntesis_c
;
```

//8. Utilizar la estructura de C++ para for

```
FOR ::= For Parentesis_a SENTENCIA_FOR Parentesis_c Llave_a SENTENCIA Llave_c
```

```
;
```

//9. Las dos primeras secciones, usar lo que se tiene hasta ahora y la última sección usar un nuevo axioma llamado declaración_for. Considera que en la primera sección se puede declarar o no la variable.

```
SENTENCIA_FOR ::=
    T_dato Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma DECLARACION_FOR |
    Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma DECLARACION_FOR
;
```

//10. Es la tercera sección del for. Considera que puede usarse el operador de incremento o algún operador de atribución.

```
DECLARACION_FOR ::=
    Identificador Op_atribucion Numero |
    Identificador Op_incremento |
    Op_incremento Identificador
;
```