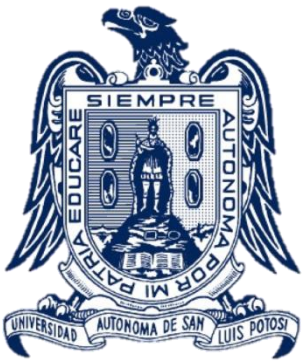
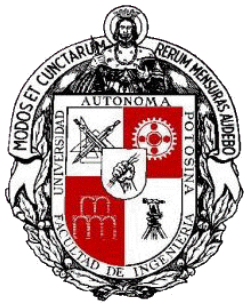




UNIVERSIDAD AUTONOMA DE SAN LUIS POTOSÍ

FACULTAD DE INGENIERÍA

ÁREA DE CIENCIAS DE LA COMPUTACIÓN

"LABORATORIO PROGRAMACIÓN DE SISTEMAS"

PRACTICA 1:
CALCULADORA DE EXPRESIONES ARITMÉTICAS

PROFESOR:
ING. AGUSTÍN HERNÁNDEZ GARCÍA

ALUMNO: Braulio A. García Rivera

CLAVE ÚNICA: 0239196

FECHA: 23/09/2020

SEMESTRE: 2020-2021/I

1. INTRODUCCIÓN

En este documento se explica cómo fue el acercamiento con la generación de una gramática con la herramienta ANTLR4 a través de Visual Studio 2017 para posteriormente utilizarla como una calculadora de expresiones aritméticas para evaluar y ver resultados.

2. INSTALACIÓN Y CONFIGURACIÓN

En Visual Studio 2017 se instalaron las siguientes extensiones:

- Antlr4Code
- ANTLR Language Support

También se instalaron las bibliotecas de Antlr4 como paquetes NuGet:

- Antlr4 v4.6.6
- Antlr4.CodeGenerator v4.6.6
- Antlr4.Runtime v4.6.6

La configuración se realizó en base a la explicación del profesor y los ejemplos de la plataforma didacTIC.

3. COMPONENTES LEXICOS

- 0,1,2,3,4,5,6,7,8,9.
- +, -, *, /, (,).

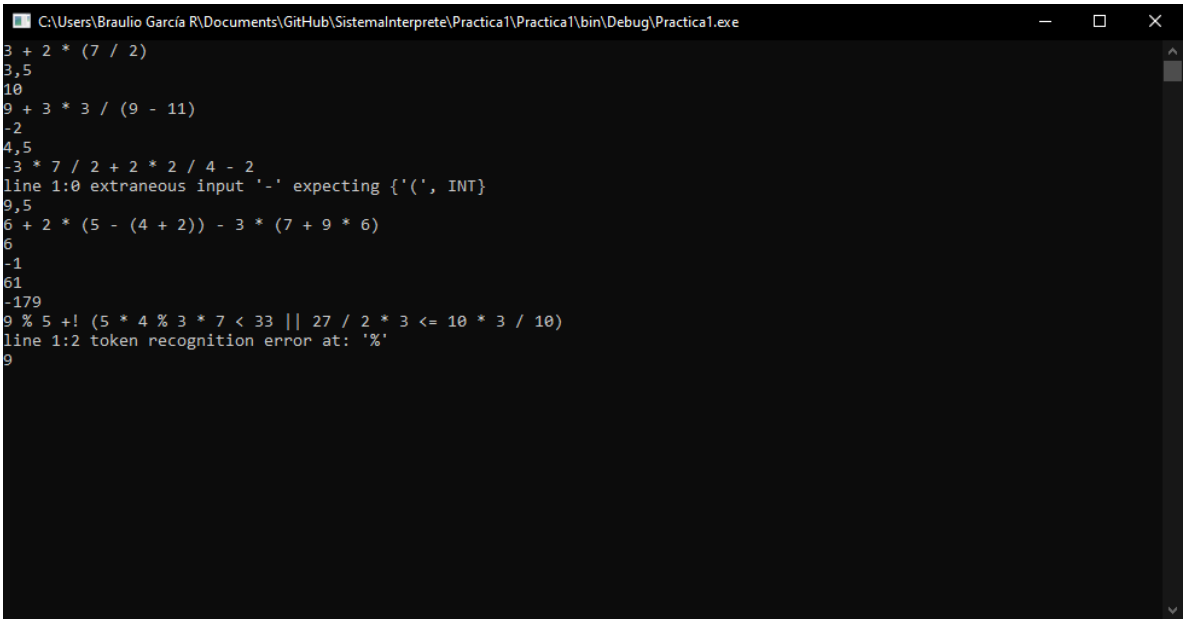
4. REGLAS GRAMATICALES

```
grammar Combined1;


/*
*opciones de compilacion de la gramatica
*/
options {
    language=CSharp2;                                                    //lenguaje
objetivo de la gramatica
}

/*
*      Reglas del Parser
*/
programa returns[float value]                                            //el
programa retornara un valor entero.
       : stat{System.Console.WriteLine($stat.value);}  //se imprime el valor que
calculo el parser.
       ;

stat returns[float value]                                                //la
expresion retornara un valor entero al programa.
       :
```

```
	c = expresion NEWLINE	{System.Console.WriteLine($c.value);}	//Se
imprime el valor adquirido.
|NEWLINE;			//no se hace nada.

expresion returns[float value]						//El valor
calculado por la expresion sera regresado como un entero.
	:
	a = multiplicacion{$value = $a.value;} (	//Se asina el valor que se
retornara en la regla.
	MAS b = multiplicacion {$value =$value + $b.value;}
	//El valor se suma con el actual en la expresion.
	|
	MENOS b = multiplicacion{$value =$value-
$b.value;})*{System.Console.WriteLine($value);} //El valor se resta con el actual y
se imprime el valor.
	;

multiplicacion returns[float value]					//La regla
retorna un entero.
	:
	a = numero{$value = $a.value;}  (			//Se asigna el valor
que se regresara.
	POR b = numero{$value =$value* $b.value;}		//Se calcula la
multiplicacion
	|
	ENTRE b = numero{$value =$value/ $b.value;})*	//Se calcula la division.
	;
numero returns[float value]						//La regla
retonara un entero.
	:
	INT	{$value = float.Parse($INT.text);}			//se convierte a
entero la cadena de entrada de la consola.
	|
	PARENI expresion PAREND		{$value = $expresion.value;}
	//se asigna el valor de la expresion dentro del parentesis.
	;

/*
*	Reglas del Lexer.
*/
PARENI
	:	'('		//token de parentesis derecho
	;
PAREND
	:	')'		//token de parentesis izquierdo.
	;
MAS
	: '+'		//token de signo mas
	;
MENOS
	: '-'		//token de signo menos
	;
POR
	: '*'		//token de signo por
	;
INT
	:('0'..'9')+	//tokens validos para numeros
	;
```

```
ENTRE
	: '/'		//token de signo entre
	;

NEWLINE				//token para identificar el final de la expresion.
	: '\n'
	;
WS
	: (' '|'\r'|'\n'|'\t')+ {Skip();} //tokens que identifican las secuencas de
escape.
	;
```

5. EVALUACIÓN DE LAS EXPRESIONES



3 + 2 * (k = 7 / 2)	= 10
9 + 3 * 3 / (9 – 11)	= 4.5
-3 * 7 / 2 + 2 * 2 / 4 – 2	= 9.5
6 + 2 * (5 – (4 + 2)) – 3 * (7 + 9 * 6)	= -179
9 % 5 +! (5 * 4 % 3 * 7 < 33 || 27 / 2 * 3 <= 10 * 3 / 10)	= 9

6. PROBLEMAS

Al inicio se trató de implementar la herramienta con Visual Studio 2019 y antlr-4.8-complete pero no se ejecutaba correctamente la herramienta de antlr, después se usó una versión más antigua y empezaba a mostrar incompatibilidad al juntarlo con el programa de consola en Visual Studio 2019, después de varios intentos opte por generarlo directo sobre visual Studio 2017 lo cual resulto un poco más entendible logrando generar la gramática correctamente.

Al dividir entre 0 el programa dejaba de funcionar por lo cual se le hizo una excepción para que avisara al usuario con un mensaje y el programa se mantuviera estable.

No reconoce la letra “k” y el símbolo igual “=”.

No reconoce el símbolo de menos “-” al inicio de las expresiones.

No reconoce el símbolo %.

7. POSIBLES MEJORAS

Mejorar el entorno de la herramienta ANTLR y modificación para poder implementarla en cualquier Visual Studio de manera genérica para evitar detales de compatibilidad.

8. CONCLUSIÓN

Con la utilización de esta calculadora de expresiones podemos observar los resultados generados respetando la jerarquía de operaciones, así como la ejecución por partes para llegar al resultado final, es complicada la herramienta al momento de generar la gramática pues tiene partes donde provoca confusión, sin embargo, se consiguió el objetivo.