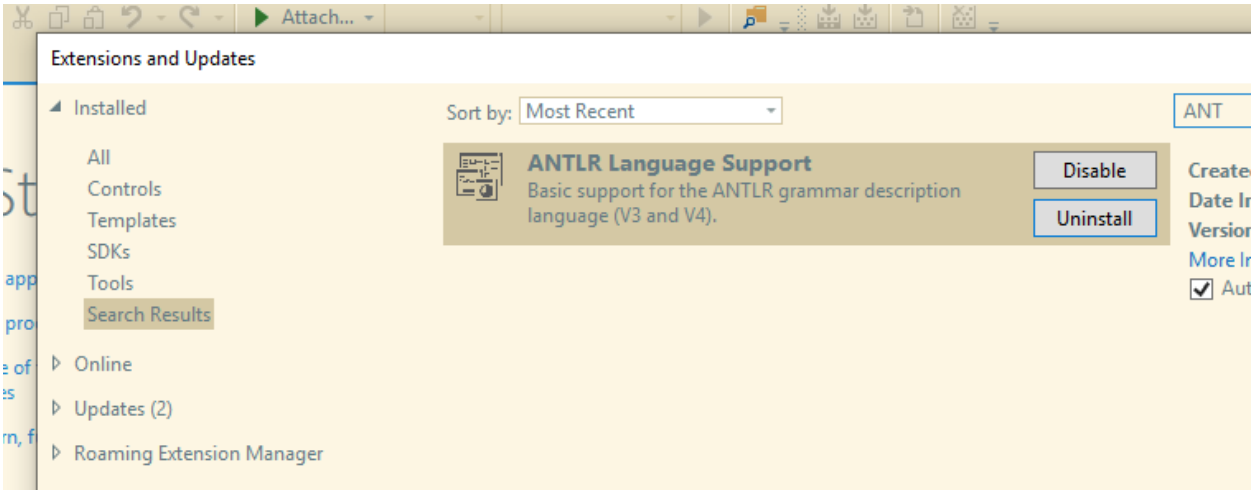
PRÁCTICA 1

Laboratorio de Programación de sistemas

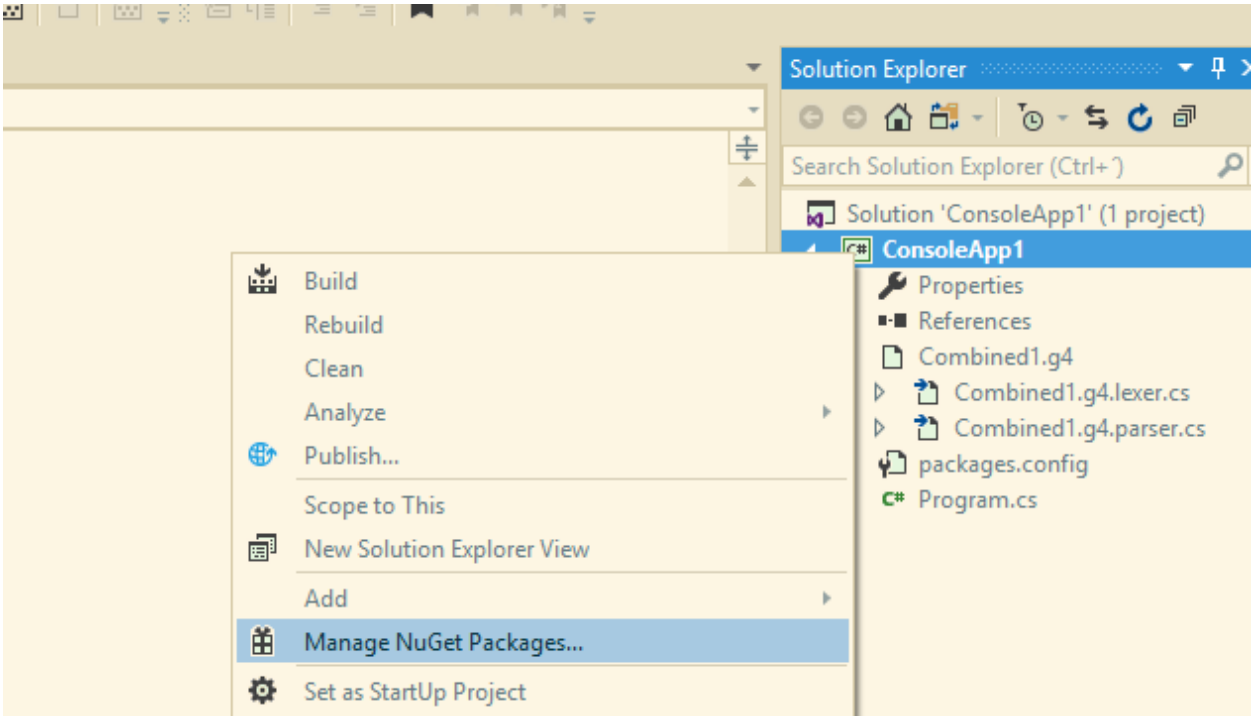
Oscar Armando González Patiño
Laboratorio día jueves

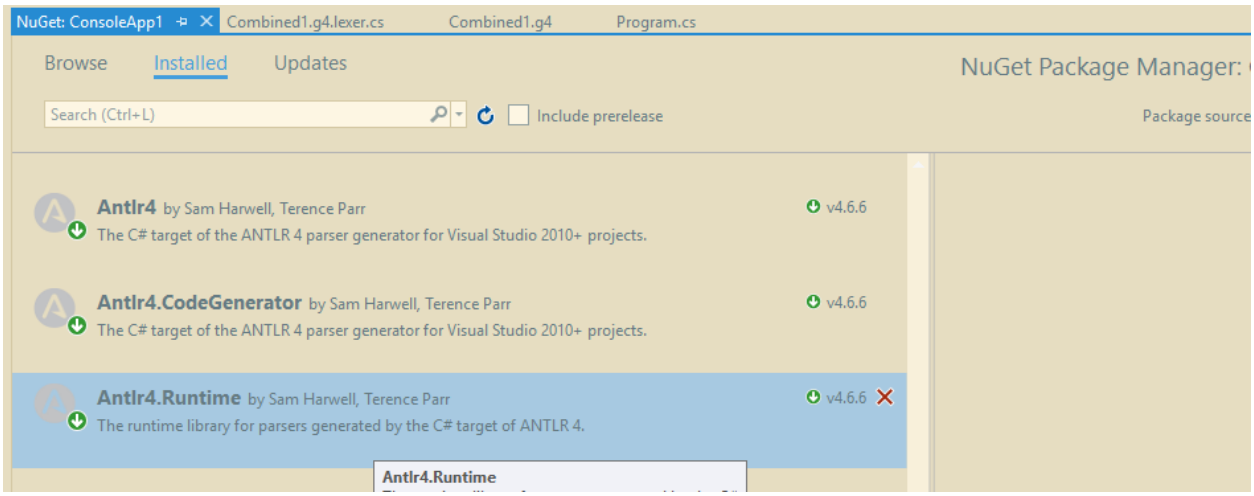
Descripción del procedimiento de instalación y configuración de herramientas de programación

Para instalar el programa ANTLR se instala como un módulo de Visual Studio. En el menú de herramientas, seleccionar la opción de extensiones, en la ventana se busca ANTLR y se instala la versión 4

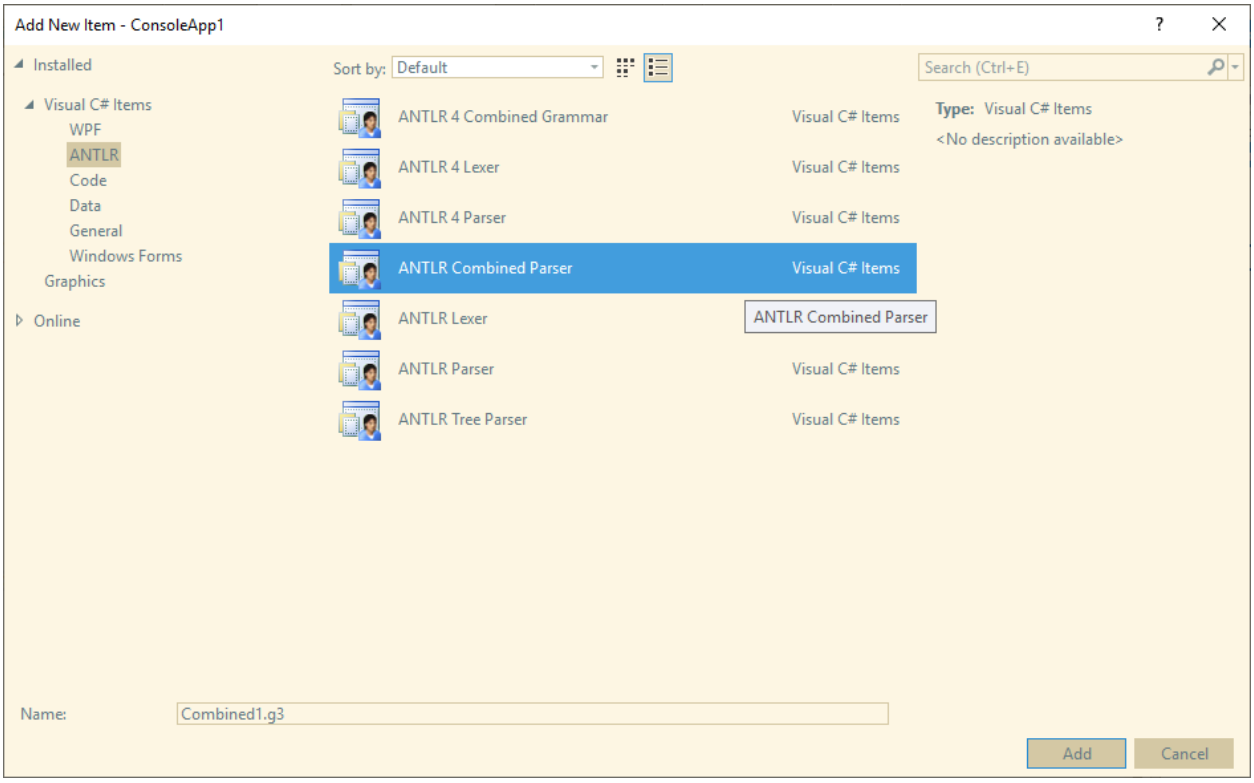


Al hacer un nuevo proyecto, se tiene que administrar los controles NuGet del lado del explorador del proyecto, y así se tiene que hacer con cada proyecto que requiera la utilización de ANTLR. Se importan la versión 4 de los siguientes módulos.





Y se tiene que agregar una gramática combinada que contiene el **leer** y el **parser,** agregando un nuevo objeto al proyecto. El objeto es de tipo **combined** si se quiere manejar el leer y el parser



Después de ingresar el modulo ya se puede empezar a editar la gramática "***.g4".** Para compilar y hacer el leer y el parser, solo se tiene que guardar el archivo "**.g4",** y seguido de esto, se puede compilar el programa. El tipo de proyecto puede ser de consola o con Windows forms.

Descripción de los componentes léxicos utilizados

Se hizo la gramática para una calculadora básica que realiza las operaciones de suma, resta, multiplicación y división, y con el uso de paréntesis. La gramática consta de una serie de reglas que van a tener como entrada una cadena de texto y la van a analizar y van a hacer un resultado.

Explicación de las reglas gramaticales utilizadas

```
stat returns[int value]:  //la expresion retornara un valor entero al programa.
	c = exp NEWLINE    {
		System.Console.Write($c.value); }     //Se imprime el valor adquirido.
	|NEWLINE                        //no se hace nada.
;

exp returns[int value]:   //El valor calculado por la expresion sera regresado como
un entero.

	a = exp2{$value = $a.value;} (     //Se asina el valor que se retornara
		ADD b = exp2 { $value = $value + $b.value; } //se suma con el actual en
la expresion.
		|SUB b = exp2 { $value = $value - $b.value; }
	)*{System.Console.WriteLine($value);}
;

exp2 returns[int value]:                    //La regla retorna un entero.
	a = num { $value = $a.value; } (       //Se asigna el valor que se regresara.
		MUL b = num {      $value = $value * $b.value; }
		|
		DIV b = num {if($b.value != 0){
                                $value = $value / $b.value;
                            }
                            else{
                                System.Console.WriteLine("Error: division entre ");
                                $value = 0;
                            }}
	)*
;

num returns[int value]:                //Ser egresa un entero
	INT    { $value = int.Parse($INT.text); }
|     PARENI exp PAREND  { $value = $exp.value; }
;

compileUnit
	:     EOF
	;

/*
 * Lexer Rules
 */

PARENI:            '(';               //parentesis derecho
PAREND:            ')';               //parentesis izquierdo.
ADD:         '+';               //signo mas
SUB:         '-';               //signo menos
MUL:         '*';               //signo por
INT:         ('0'..'9')+|SUB('0'..'9')+;     //numeros
DIV:         '/';               //signo entre
NEWLINE:     '\n';              //final de la expresion.
WS:          (' '|'\r'|'\n'|'\t') -> channel(HIDDEN);     //las secuencas de escape.
```

Descripción de los problemas encontrados y una explicación detallada de cómo fueron solucionados

Los problemas encontrados fueron principalmente la compilación del proyecto. No se podía compilar por las ciertas reglas que deben seguir al escribir los archivos al hacerlos.

Primero, la gramática tiene que tener el mismo nombre que tiene en el archivo in la extensión .g4. También se tiene que referenciar en el programa principal hacia la gramática, pero solamente **después** de guardar la gramática ya que así va a generar los componentes lexer y parser para poder agregar al proyecto.

Para ejecutar en analizador se ejecuta la primera regla que tenga que ejecutarse de acuerdo con la gramática. Puede ejecutarse otra regla, pero no va a detectar las reglas que están sobre ella. El analizador recibe una cadena de entrada que corresponde a un posible resultado de la gramática.

El analizador puede programarse para que regrese un valor específico d acuerdo a la gramática que está definida.

Conclusiones y posibles mejoras

El analizador de ANTLR puede instalarse como un módulo en el programa visual studio, y se puede implementar utilizando C# o C++. El programa sirve para manejo de gramáticas y expresiones regulares y puede ser de gran utilidad para entender los compiladores. El programa de ejemplo utiliza los elementos básicos de ANTLR para poder comprender mejor la extensión.

Una posible mejora es que el programa pueda manejar todo tipo de gramáticas para cualquier lenguaje y que proporcione las herramientas para programarlo de forma visual