**Proyecto - Fase #2 - Analizador sintáctico ascendente**

**Mini C# - Mini Java**

Fecha de entrega: 27-28 de octubre, 07:00 pm

**El objetivo**

La segunda fase del proyecto consistirá en analizar sintácticamente un programa escrito en el lenguaje definido. En esta fase, deberán definir la gramática que reconozca dicho lenguaje y analizarla implementando alguno de los métodos de análisis ascendente vistos. Además, deberán integrarlo con su analizador léxico de la primera fase. Al final de esta fase debe ser posible determinar que el programa fuente escrito está sintácticamente correcto.

**Estructura Lexicográfica**

Todas las especificaciones de la fase #1, con el agregado de las palabras reservadas:

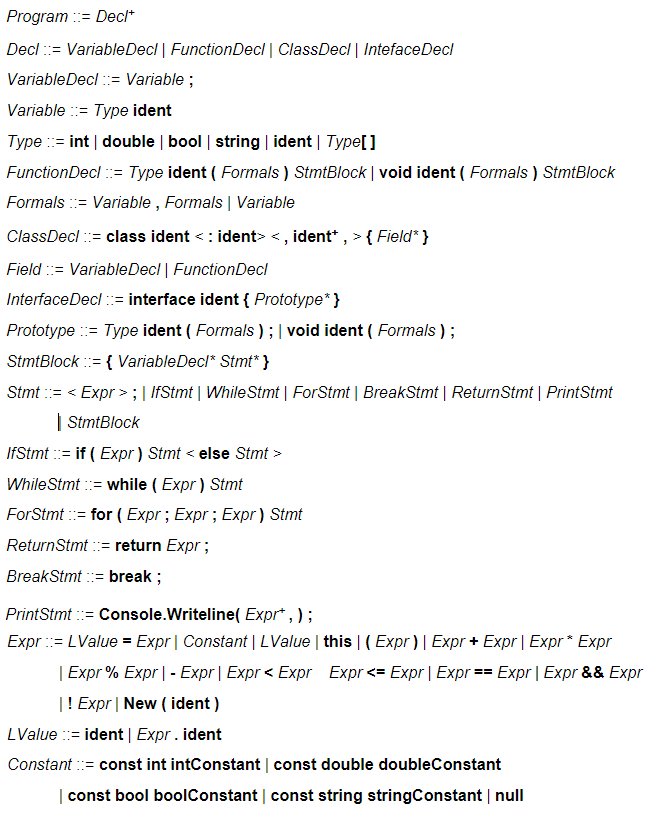
* Mini-Java: **static** (en lugar de **const** de la primera fase)

**Gramática**

Generalidades de la gramática:

* **x** significa que x es un terminal, un token devuelto por el analizador léxico. Los terminales están en minúscula y negrita excepto aquellos que son palabras reservadas
* *x* en itálico es un no terminal. Todos los no terminales tienen la primera letra en mayúscula
* <x> significa cero o una ocurrencia de x, es decir, x es opcional
* x\* significa cero o más ocurrencias de x
* x+ significa una o más ocurrencias de x (Listadas, seguidas por coma excepto el último elemento)
* | significa las alternativas de las producciones

La gramática del lenguaje Mini C# es la siguiente:



La gramática del lenguaje Mini Java es la siguiente:



Tome en cuenta que estas gramáticas no están en una notación usual de definición. Por ejemplo, las producciones con \*, +, <> deberá modificarlas para implementar la gramática en su analizador. Puede rediseñar la gramática si lo considera pertinente, solo asegúrese de reconocer el mismo lenguaje que reconoce la gramática base.

**Implementación**

1. Queda a criterio de cada equipo cuál método de análisis ascendente implementar en su analizador (SLR o LR(1)).
2. El programa lee un archivo de entrada en lenguaje Mini C# o Mini Java y debe indicar en pantalla únicamente si está sintácticamente correcto o no. No debe realizar ninguna acción adicional si está correcto.
3. De encontrar algún error, el analizador debe continuar hasta el final de archivo.
   1. Los errores deberá reportarlos en pantalla indicando:

Línea y columna, símbolo que provocó el error, Error: <descripción del error>.

**Requerimientos generales**

* El ejecutable debe ser llamado minic o minij y debe pedir el archivo de entrada y escribir todos los resultados en pantalla.
* El proyecto se entrega en el mismo repositorio en GitHub de la fase anterior, sobre la rama master o principal. Por tanto, se deberá crear una rama con su analizador léxico y una nueva donde incorpore esta fase. Se calificará el proyecto hasta el último commit realizado antes de la hora de entrega. Se revisará el historial del repositorio. No se aceptarán cambios en el momento de la calificación.
* En el repositorio deberá incluir:
  + un archivo con la gramática final implementada
  + Un archivo con la tabla de análisis diseñada indicando el método usado.
  + Un archivo README.txt donde se detalle cómo su analizador maneja los errores.

**¡Buena suerte!**