Definición general

Esta segunda etapa del proyecto conocida formalmente como parser o Análisis Sintáctico es, si se quiere la más importante del proyecto, pues de la gramática depende que el compilador se desempeñe de la mejor forma posible. De ahí que este proyecto se desarrolle con cuidado y lo mejor posible.
Para esta etapa del proyecto se debe entregar un programa que reciba un código fuente escrito en ABC (El lenguaje que inventamos para el proyecto) y realice el análisis léxico y sintáctico correspondiente. Para esto se debe utilizar la definición del lenguaje aceptado por el Scanner junto con la gramática. Por lo tanto el programa debe hacerse en Java utilizando Jflex y Cup
Al finalizar el parseo el programa deberá desplegarle al usuario el resultado del Análisis léxico y sintáctico que se efectúo. Se espera que despliegue

1. Listado de errores léxicos encontrados: El programa debe desplegar una lista de todos los errores léxicos que se encontraron en el código fuente. Debe desplegar la línea en la que se encontró el error. Es importante que el programa deba poder recuperarse del error y no desplegar los errores en cascada ni terminar de hacer el scaneo al encontrar el primer error.

2. Listado de errores sintácticos encontrados: El programa debe desplegar una lista de todos los errores sintácticos que se encontraron en el código fuente. Debe desplegar la línea en la que se encontró el error. Además, el mensaje de error debe ser lo más específico posible, con el fin de que el programador pueda llegar al error y corregirlo de forma eficiente. Es importante que el programa deba recuperarse del error y evitar no desplegar errores en cascada ni terminar de hacer el parseo al encontrar el primer error.

Descripción Detallada

Se les sugiere tomar en cuenta los siguientes aspectos para asegurar la completítud del programa.

- Las palabras reservadas que se deben incluir en la gramática son las siguientes:

  ARRAY BEGIN BOOLEAN CASE CHAR CONST DO ELSE END FALSE FOR FUNCTION IF INT LONGINT OF PROCEDURE PROGRAM READ REAL REPEAT SHORTINT STRING THEN TO TRUE UNTIL VAR WHILE WRITE

- La estructura del programa es la siguiente. Las constantes, Globales u otras funciones pueden venir o no.

```
PROGRAM Nombre
Constantes

Globales

Funciones

BEGIN   // esto se refiere al main del programa
…..
END
```

- Las variables pueden ser de tipo int, longint, shortint, char, string, boolean y real., La declaración de

variables es de la siguiente forma.

```
VAR
Nombre, apellido:STRING;
contador: INT;
nombres: ARRAY[1..100] OF CHAR;
```

- Se podrá crear variables tipo arreglos de los 4 tipos de datos numéricos: char, int, longint, shortint. Esto significa que se utilizarán los operadores [y]. Ejemplo: **colección: ARRAY [1..50] OF INT;**

- La declaración de constantes es igual solo que las precede la palabra CONST

```
CONST
PI = 3.1415926;
Nombre = 'Juan Gutiérrez';
```

- La estructura de las funciones es la siguiente

```
FUNCTION/PROCEDURE  f (tipo x, tipo y, ….  ): TipoRetorno
BEGIN
        [ declaraciones y/o Constantes (Pueden no venir y la estructura es igual que las globales)
        [ cuerpo

        f:= valor de Retorno
END
```

el returno es obligatorio en las funciones. Los procedures son funciones que no devuelven nada.

- Las asignaciones se hacen así: **Variable:= Valor;**
- Con respecto a las funciones: READ y WRITE. La primera puede o no tener un parámetro y la segunda siempre tendrá al menos un parámetro.
- Las estructuras de control tienen la siguiente estructura:

```
WHILE condicion DO
 BEGIN
  sentencia 1
  sentencia 2
  ……
 END

REPEAT
  sentencia 1
  sentencia 2
  ……
 UNTIL condicion

FOR variable:=exp1 TO exp2 DO
 BEGIN
   sentencia1
    …….
 END
```

```
IF condicion THEN
sentencia1
sentencia2
….
[ELSE
sentencia1
sentencia2
…. ] //podria no venir lo que esta en []
END


CASE variable OF
 Constante1 : sentencia1;
 Constante2 : sentencia2;
 Constanten : sentencian;
ELSE
 sentencia
```

- Los operadores que se deben tomar en cuenta son los siguientes:

Aritméticos:	"++"	"--“	:="	"+"	"-"	"*”	"/"	"MOD"	"("	")"
	"+="	"-="	“*=”	"/="	DIV					
Booleanos:	"="	">="	">"	<="	"<"	"<>"	"OR"	"AND"	"NOT"	

En este aspecto es importante recordar que las condiciones de las estructuras de control utilizan solamente expresiones booleanas. Para las asignaciones si pueden tener ambas operaciones.

- Se deben implementar buenos mensajes de error

Documentación

Se espera que sea un documento donde especifique lo siguiente:

a. Portada, índice, introducción
b. Estrategia de Solución. Debe incluir la gramática que están usando, sin código.
c. Análisis de Resultados: Deberá elaborar un listado de todas y cada una de las actividades y tareas que deben cubrirse a nivel funcional, para cada una de ellas debe aportar el porcentaje de realización y en caso de no ser el 100% debe justificarse.
d. Lecciones aprendidas: Debe prepararse un listado de las lecciones aprendidas producto del desarrollo de la tarea programada. Las lecciones aprendidas pueden ser de carácter personal y/o técnico que involucre aspectos que han logrado un aprendizaje en temas de investigación, desarrollo de habilidades técnicas y habilidades blandas como trabajo en equipo, comunicación, forma de expresar ideas, entre otros.
e. Casos de pruebas: se espera que definan claramente cada prueba, cuáles son los resultados esperados y cuáles fueron los resultados obtenidos. No es necesario que sean grandes pero deben evaluar la funcionalidad completa del programa.
f. Manual de usuario: especificar como compilar y correr su tarea.
g. Bitácora de trabajo durante las semanas de trabajo, incluyendo verificaciones realizadas (si existieran) de consultas realizadas con el profesor o asistente.
h. Bibliografía y fuentes digitales utilizadas.

Aspectos Administrativos

- Los grupos deben permanecer iguales a la primera etapa del proyecto.
- El trabajo se debe de entregar el día 2 de Noviembre de 2018 antes de media noche, enviarlo por correo o subirlo al tec digital.
- Recuerde que oficialmente no se recibirán trabajos con entrega tardía.