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 Casi-Pascal (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 no venir.

```
Program Nombre
Constantes

Globales

Funciones

begin
…..
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;

- 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
        [ 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
```

```
For contador:=exp1 To exp2 do
 Begin
 sentencia1
 sentencia2
 …….
 End
```

```
Repeat
 sentencia 1
 sentencia 2
 ……
 Until condicion
```

```
Case selector Of
 Constante1 : sentencia1;
 Constante2 : sentencia2;
 Constanten : sentencian;
Else
 sentencia
```

If condicion Then sentencia1 [Else sentencia2]

- 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 el análisis de resultados del programa junto con unos casos de pruebas. El Análisis debe resumir el resultado de la programación, qué sirve, qué no sirve y aspectos que consideren relevantes. Para las 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.
Deben especificar, además, como compilar su parser. E incluir la gramática que están usando, sin código.

Aspectos Administrativos

- Los grupos deben permanecer iguales a la primera etapa del proyecto.
- El trabajo se debe de entregar el día 9 de Noviembre de 2014. Deben enviarlo por correo electrónico antes de las 12 de la noche, al correo emarin@ic-itcr.ac.cr y erikams79@gmail.com
- Deben entregar el código fuente junto con el ejecutable.
- Se debe de entregar la documentación IMPRESA para ser calificado, así como una copia del archivo en el correo. La documentación impresa se debe entregar el día de las revisiones.
- Recuerde que oficialmente no se recibirán trabajos con entrega tardía.