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 MyPhyton (El lenguaje que definimos 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 completitud del programa.

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

  and break class continue def elif else except finally for if in input is not or print return try while int list float string boolean char

- La estructura del programa puede ser de cualquiera de las siguientes formas.

  Funciones

  Definición de variables

  Codigo principal

  Class nombre:
  Definición de variables
  Definición de Funciones
  Codigo Principal

- Las variables pueden ser de tipo int, float, list, string, boolean y char., La declaración de variables es de la siguiente forma.

Tipo nombre;

- La estructura de las funciones es la siguiente

def f (tipo x, tipo y, ….):
[declaración de variables
[cuerpo funcion

- El parser debe validar todo tipo de expresiones aritméticas y booleanas. Recuerde que una llamada a función también se considera como una expresión.
- Las asignaciones se hacen así: **Variable = Expresion**
- Definir la estructura para la función print e input
- Recuerde que una llamada a función puede ser del tipo: **Funcion()** o puede ser una función de una clase: **clase.funcion()**
- También son válidas la creación de clases: c = Clase()
- Las estructuras de control tienen la siguiente estructura:

while condición:
bloque

for i in range():
bloque

if condición:

bloque

[Elif

Bloque

[Else

bloque

- En un bloque pueden venir: asignaciones, otras estructuras de control, llamadas a funciones o print o input. Dentro de las estructuras de control de ciclo pueden venir las sentencias break y continue.
- Para las condiciones deben contemplar la estructura para usar el in, range,
- Implementar la estructura para el control de los errores

try:
bloque
except Identificador:
bloque
finally: (No es obligatorio)
bloque

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

Aritméticos:

+	-	*	/	//	%	**	=	+=
-=	*=	/=	**=	//=	/=	(	)	

Booleanos:

!=	<>	>	<	>=	<=	==	AND	OR	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 3 de Junio de 2016. Deben enviarlo por correo electrónico antes de las 12 de la noche, al correo emarin@ic-itcr.ac.cr y erikams79@gmail.com
- 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.