UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN

Primer Semestre 2016

Sección	Catedrática	Tutor académico
A-	Inga. Damaris Campos de López	Eiji De Paz
B-	Inga. Zulma Aguirre	Luis Fernando Lara Lemus

Enunciado de Proyecto No. 2

Objetivos:

- Comprender y aprender a escribir gramáticas libres de contexto, para poder crear un analizador sintáctico.
- Implementar analizadores sintácticos, para establecer el orden correcto de los tokens resultantes de la fase léxica del compilador.
- Integrar las dos primeras fases de un compilador; análisis léxico y sintáctico.
- Aprender a utilizar herramientas integradas en el lenguaje Visual Basic.

Descripción:

Se le ha contratado a usted para desarrollar una herramienta flexible y atractiva para niños de 5 a 10 años, que facilite la tarea de leer cuentos a maestros y padres de familia.
La herramienta consiste en tener una biblioteca de audio cuentos, capaz de convertir texto a voz, esto con el objetivo de poder agregar cuentos dinámicamente a la biblioteca, por medio de un lenguaje definido posteriormente en el enunciado y que llamaremos LF-Py, el cual será gestionado por un usuario que entiende de programación. La aplicación contará con algunas funciones definidas más adelante.
El programa contará con un editor de texto, con el cual se pueda definir el comportamiento de la aplicación, por lo cual se recibirá como entrada un archivo con la extensión .py, este será analizado léxica y sintácticamente y posteriormente, siempre y cuando no exista ningún error, tendrá como salida la biblioteca de audio cuentos.

Definición del Lenguaje:

Como se mencionó, el programa contará con un módulo para que un usuario pueda programar la aplicación, en la cual se tendrá que definir los cuentos por medio de clases, por medio del lenguaje al que llamaremos LF-Py basado en Python, el cual es un lenguaje orientado a objetos y más sencillo de entender que Java.

- Creación de clases:
  Cada cuento será representado por una clase en el lenguaje LF-Py, se puede definir N clases en un mismo archivo .py. La creación de clases es posible por medio de la palabra reservada “class” como se muestra en el ejemplo:

```
class <identificador>{
    <declaracion 1>
    .
    .
    .
    <declarion N>
}
```

En ocasiones necesitamos instanciar una clase con valores iniciales, por tal razón usamos un método especial que inicializa los valores de una clase o también llamado, método constructor.

```
class <nombreClase>{
    def __init__(self){
        <nombreAtributo> = "Caperucita Roja"
    }
}
```

Nota:

def	Palabra reservada utilizada para indicarle al lenguaje que se declarará un método/función.
__init__	Palabra reservada que está compuesto por dos guiones bajos seguidos al inicio y al final, utilizado para indicarle al lenguaje que es el método constructor.
self	Palabra reservada, equivalente a un 'this' en java, es obligatoria mencionarla como 1er parámetro en cualquier método/función.

- Atributos: son características que identifican a un objeto. Para el lenguaje LF-Py es obligatorio que cada clase, es decir cada cuento, tenga los siguientes atributos o variables:
    - ***titulo**:* esta variable contendrá el nombre o título del cuento.
    - ***cuento**:* esta variable contendrá el texto del cuento completo, el cual será mostrado y traducido a voz en el módulo Maestro/Padre (Explicado más adelante).
    - ***imagen**:* esta variable o atributo almacena la ruta donde se encuentra la imagen de portada del cuento.
    - ***tipoVoz**:* esta variable le podemos asignar 2 valores
        - *Female:* la voz que escuchará el usuario final será de una mujer.
        - *Male:* la voz que escuchará el usuario final será de un hombre.
    - ***volumen**:* numero comprendido entre 0 y 100 que representa el volumen de las bocinas.
    - ***velocidad**:* numero comprendido entre -10 y 10 que representa la velocidad con la cual se escuchará la narración (-10 es muy lento, 10 es rápido y 0 normal).
    - Para crear un atributo o variable NO es necesario declararlo, basta con asignarle un valor inicial y automáticamente se creará nuestra variable.
    - Es obligatorio asignarle un valor inicial a nuestra variable para que se cree.
    - Es posible declarar N variables con su valor respectico en una sola línea.
    - Dentro de cualquier método o función, incluyendo el constructor, es posible hacer referencia a un atributo con la palabra reservada 'self', fuera de cualquier método o función no es posible.

Nota:

True	Palabra reservada para denotar un valor booleano verdadero
False	Palabra reservada para denotar un valor booleano falso.
Cadenas	Las cadenas de texto empiezan y terminan con comillas dobles.
Caracter	Al igual que las cadenas, inician y terminan con comillas dobles.

- Métodos: utilizados para gestionar nuestra aplicación, la cual cuenta con 3 métodos propios, los cuales pueden estar presentes en una clase en cualquier orden, y como se mencionó darán funcionalidad a nuestro programa.
    - Métodos obligatorios:
        - **traducir_a_voz**: este método cuenta con un atributo obligatorio llamado 'traduccion' el cual le dirá a nuestro lenguaje, por medio de la palabra reservada 'True' o 'False' que la traducción de texto a voz estará o no disponible en el Modo-Maestro/Padre.
        - **mostrar_texto_cuento**: este método cuenta con un atributo obligatorio llamado 'texto' el cual le dirá a nuestro lenguaje, por medio de la palabra reservada 'True' o 'False' si el texto de nuestro cuento se muestra o no en el Modo-Maestro/Padre.
        - **resaltar_palabra**: este método cuenta con un atributo obligatorio llamado 'resaltar'

el cual le dirá a nuestro lenguaje, por medio de la palabra reservada 'True' o 'False' si se debe pintar de color Verde la palabra que se está escuchando en tiempo real, por lo tanto, se debe validar, que, si está activada esta variable, las otras dos anteriores también lo estén, pues no se podrá pintar de color verde la palabra actualmente en reproducción si no se encuentran activas las variables de mostrar el texto y de traducir de texto a voz en el Modo-Maestro/Padre.

  - Los métodos se declaran con la palabra reservada 'def'.
  - El primer parámetro que debe tener todo método es la palabra reservada 'self'.

- Comentarios: existen 2 tipos de comentarios de una línea y múltiples líneas.
  - Los comentarios de una sola línea inician con '#'
  - Los comentarios de varias líneas inician y terminan con 3 comillas dobles.

- Operaciones: las operaciones pueden presentarse en cualquier asignación de variable/atributo.

Precedencia	Operación	Operador
1	Suma	+
1	Resta	-
2	Multiplicación	*
2	División	/
3	Agrupación	()

Ecuación	Resultado
2*5+1	11
(2+4)*2	12
10/5*3	6

Interfaz gráfica:

La herramienta a desarrollar contará con 2 modos de ingreso, una para el usuario desarrollador y el otro para maestros o padres de familia que usarán la aplicación.

- Pantalla de Inicio:
  Esta pantalla da la bienvenida al usuario, y le pide ingresar a cualquiera de los 2 modos, según el modo seleccionado, se debe abrir otra ventana que mostrará las opciones correspondientes.
  NOTA: se debe validar que si es la primera vez que se ingresa a la aplicación, el modo Maestro/Padre no esté disponible hasta que desarrollador haya definido el comportamiento de la aplicación.

un

- Modo-Desarrollador
  Al igual de como se ha venido trabajando, la herramienta contará con un editor de texto el cual podrá manejar pestañas, es decir, abrir, guardar y escribir en cualquier pestaña los archivos con extensión .py; se debe contar con la opción de ayuda (ayuda de la aplicación en pdf y datos del estudiante). Adicional a esto, se contará con estos botones:
  - A. Léxico: botón que inicia el análisis léxico.
  - A. Sintáctico: botón que inicia el análisis sintáctico.
  - Generar Archivos: botón que, al haber algún error, ya sea en la fase léxica o sintáctica, muestre los errores en una tabla HTML, se adjunta ejemplo en la sección, archivos de salida.

- Modo-Maestro/Padre

  Es en este modo donde el usuario final podrá hacer uso realmente de la aplicación, pues se tendrá que crear y mostrar de una forma ordenada y creativa los cuentos que el usuario desarrollador haya definido.

  Funcionalidades:

  - **Reproducir cuento**: con este botón la aplicación será capaz de traducir el texto, definido por el usuario desarrollador, a voz, siempre y cuando haya sido definido como 'True' la variable 'traduccion' en el método 'traducir_a_voz'.
  - **Mostrar cuento**: debe mostrarse el cuento definido por el usuario desarrollador en alguna etiqueta o área de texto, para que el usuario final pueda leerlo, siempre y cuando haya sido definida la variable 'texto' como 'True' en el método 'mostrar_texto_cuento'.

  - **Resaltar palabras**: debe de pintarse de color verde, en el texto del cuento mostrado, la palabra que se esté escuchando actualmente, todo en tiempo real, siempre y cuando este definida la variable 'resaltar' como 'True' en el método 'resaltar_palabra'.

  NOTA: es indispensable que la aplicación pueda traducir el texto a voz, no se permite reproducir algún archivo de audio que contenta el cuento narrado.

  NOTA: el usuario desarrollador puede definir N cuentos, y debe mostrarse todos en una forma ordena con su nombre e imagen. Se recomienda mostrarlo en forma de tabla o lista.

Archivos de Salida:

El archivo de Tokens se debe mostrar en una página HTML, la información en una tabla como se muestra a continuación:

#	Fila	Columna	Lexema	Id Token	Token
1	09	25	class	11	inicioClase
2	02	02	tipoVoz	12	variable
3	07	15	def	13	inicioMetodo

El archivo de Errores, si existieran, se debe mostrar en una página HTML con la información en una tabla como se muestra a continuación:

Léxicos:

#	Fila	Columna	Carácter	Descripción
1	05	10	|	Desconocido
2	08	30	¬	Desconocido
3	10	05	~	Desconocido

Sintácticos:

#	Fila	Columna	Descripción
1	05	10	Se esperaba '{' después del identificador
2	18	03	Se esperaba '+' después de un numero
3	10	05	Se esperaba '__' en el método constructor

Entregables que se deben incluir en el CD:

- Manual de Usuario
- Manual Técnico, debe incluir la definición de los tokens (IdToken, Token, patrón –Exp.Reg.-) y el DFA (por el método del árbol a partir de las Expresiones Regulares de los tokens definidos) que se use para el analizador léxico.
  Adicionalmente se debe incluir la Gramática tipo 2 (a mano o computadora) utilizada para resolver el análisis sintáctico.
- Código Fuente y ejecutable de la aplicación.

Documentación a entregar de forma física el día de la calificación:

- Hoja de calificación (Original y una copia)

Notas importantes:

- La práctica se debe desarrollar de forma individual.
- Este proyecto se deberá desarrollar utilizando Visual Basic .Net con Visual Studio 2013.
- ***Para la traducción de texto a voz, se recomienda el uso de la librería: System.speech.dll***
  - El proceso de obtener tokens, se debe hacer a través de la implementación del autómata finito determinista obtenido por medio del método del árbol y desarrollado por el propio estudiante.
  - Es obligatorio implementar el analizador sintáctico con base a la gramática tipo 2 que se incluya en la documentación.
  - No se puede agregar o quitar algún símbolo en el archivo de entrada. El proyecto deberá funcionar con los archivos de prueba que se disponga para la calificación, sin modificación.
  - La calificación del proyecto será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido. Se debe tomar en cuenta que durante la calificación no podrán estar terceras personas alrededor, de lo contrario no se calificará la práctica.
  - No se dará prórroga para la entrega del proyecto.
  - **Copia parcial o total tendrá una nota de 0 puntos, y se notificará a la Escuela de Sistemas para que se apliquen las sanciones correspondientes.**
  - **En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.**

Fecha de entrega: 03 de Mayo de 2016

Anexos:
A continuación se muestra algunos ejemplos de archivos de entrada.

```
class cuentoPinocho{

    def __init__(self){
        self.titulo = "Pinocho"
        self.cuento = "Habia una vez, en el bosque...."
        self.imagen = "C:\Documentos\pinocho.png"
        self.tipoVoz = "Female"
        self.volumen = 7*8+(24/6)
        self.velocidad = -2
    }

    def traducir_a_voz(self){
        traduccion = False
    }

    def resaltar_palabra(self){
        resaltar = True
    }

    def mostrar_texto_cuento(self){
        texto = True
    }
}
```

```
class Caperucita_Roja{
    #fuera de los metodos no se puede usar la palabra 'self'
    #puedo declarar N atributos en una sola linea
    nombre, vol, velo = "Caperucita Roja!",99,-7

    def __init__(self){
        self.cuento = "Habia una vez, en el bosque...."
        self.imagen = "C:\Documentos\caperucita.png"
        self.tipoVoz = "Female"
        self.titulo = nombre
        self.volumen = vol
        self.velocidad = velo
    }

    def resaltar_palabra(self){
        resaltar = True
    }

    def mostrar_texto_cuento(self){
        texto = True
    }

    def traducir_a_voz(self){
        traduccion = True
    }
}
```

```
class cerdos{
    #creando atributos...
    img = "C:\Documentos\cerdos.png"
    nombre = "Los 3 cerditos"
    historia = "Habia una vez, en la granja...."

    """Declaracion de metodos
       para mi programa"""
    def __init__(self){
        self.volumen = (1+1)*50
        self.tipoVoz = "Male"
        self.velocidad = 5/5
        self.imagen = img
        self.titulo = nombre
        self.cuento = historia
    }

    def mostrar_texto_cuento(self){
        texto = False
    }

    def traducir_a_voz(self){
        traduccion = True
    }

    def resaltar_palabra(self){
        #no puedo mostrar de color la palabra
        #si no activo la variable texto en metodo correspondiente
        resaltar = False
    }
}
```