17-3-2022

MANUAL DE USUARIO

Gabriel Enrique Perez Meza - 201900221
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

INDICE

PAG

FUNCIONALIDADES DEL SISTEMA

El sistema está compuesto por 5 clases principales, estas se encargan del flujo correcto del programa. Entre sus principales funciones está el analizar un archivo con extensión *".lfp",* analiza y encuentra posibles errores sintácticos y semánticos avisando al usuario por medio de la consola.

Las clases son las siguientes:

1. Class interfaz:

    a. def __init__(): En este método se crea la ventana haciendo uso de la librería Tkinter. Se crean todas las etiquetas, botones y área de texto donde estará alojado la entrada que analizara el programa.

    b. def ObtenerItem(): Para poder redireccionar al usuario al manual o reporte que desee se creó este método para recuperar la selección que hizo el usuario en el select de las opciones, el sistema revisa que seleccionó y por medio de parámetros toma la decisión de redireccionar al archivo en *".pdf"* donde se encuentra la información deseada por el usuario.

    c. def limpiar(): Con este método se limpia el área de texto donde está alojado el archivo de entrada para que se pueda ingresar otro archivo si lo desea el usuario.

    d. def abrirArchivo(): Cuando el usuario necesite ingresar el archivo que desea analizar el botón hace referencia a

este método, el cual llama a Tkinter para abrir una ventana donde estén los archivos, la ventana únicamente mostrará archivos con la extensión ".lfp" para que el usuario tenga más facilidad a la hora de buscar el archivo. Una vez abierto el archivo el método manda toda la información del mismo al área de texto para que el archivo sea visible al usuario y pueda ser modificable si lo requiera.

e. def Analizar(): Al momento que el usuario este listo para analizar su archivo y presione el botón "Analizar" este hará referencia a este método y a su vez este hará referencia al método def leerArchivo de la clase automata, le pasa como parámetro la variable donde esta almacenado todo el texto que contiene el archivo.

2. Class automata:

a. def leerArchivo: Este método recibe como referencia la variable donde está almacenado todo el texto que contiene el archivo. Lo descompone en líneas y recorre todo el archivo analizando letra por letra tod el archivo. Se utilizó un Autómata Finito Determinista (AFD) para realizar este análisis y sea más sencillo interpretar el archivo. Una vez reconoce alguna cadena valida el método llama a la clase Tokenn para almacenar el token encontrado, este recibe como parámetros el lexema, token, línea, columna. Una vez realizado el análisis llama a los métodos: GenerarReporteTokens(), GenerarReporteErrores() y generariFrame(). También llama a la clase Sintactico().

b. def generariFrame(): Para crear las salidas posibles, este método revisa las instrucciones recibidas en el archivo de entrada, hace una validación de que instrucción esta analizando y lo agrega un archivo *".html"*. Usa dos scripts diferentes, uno para el token "tk_entrada" y otro para el

token “tk_info”. Dependiendo que elija el usuario se ejecutará un script.

c. GenerarReporteTokens(): Cuando el archivo es analizado llama a esta clase. Esta clase crea un archivo *“.html”* donde se encuentra una tabla con los siguientes componentes:

   - Lexema
   - Linea
   - Columna
   - Token

   La tabla representa todos los tokens encontrados en el archivo, nos informa el lexema encontrado, la línea y columna donde se ubica y el token asignado. En dado caso encuentre una cadena no reconocida por el lenguaje arroja un token “tk_desconcocido”.

d. GenerarReporteErrores(): De igual manera que con el reporte de tokens, este método crea un archivo “.html” donde se creará una tabla con los mismo atributos que con la tabla de tokens. Aquí estará todos los tokens “tk_desconcocido” que son errores léxicos del archivo ingresado.

e. def PalabraReservada(): Se le pasa como atributo el lexema que estemos analizando y verifica que sea una palabra reservada del lenguaje, si lo es retorna True, si no lo es retorna False.

f. def ObtenerError(): Revisa en la lista de tokens cuales son token “tk_desconocido” si lo encuentra los agrega a una lista de errores y retorna la lista.

g. def ObtenerToken(): Tiene como atributo el lexema que estamos analizando, y realiza una serie de validaciones con todos los token de las palabras reservadas si en dado

caso no lo encuentra se vuelve un token “tk_desconcocido”.

3. Class Sintactico:

    a. def init(): Se inicializan las variables que usaremos en la clase, se agrega un ultimo token a nuestra lista de tokens para saber cual es el ultimo token de nuestro archivo.

    b. def Comprobar(): Se le pasa como atributo un token para compararlo con el token actual que estamos usando, si no son los mismos tokens, arroja un error y el programa imprime el token actual como error, también imprime el token que considera que se esperaba.

    c. def Inicio(): Se llama al método formulario y al método Repetir().

    d. Los demás métodos se componen del lenguaje del programa.

4. Class Tokenn:

    a. def init(): Se tiene como constructor el lugar dónde se reconocerán los tokens recibidos, asignándoles lexema, token, línea y columna.

5. Class error:

    b. def init(): Método constructor donde reconoce tokens desconocidos a signándoles lexema, token, línea y columna.

LENGUAJE OBJETO

Se utilizó la siguiente gramática para la realización, creación y análisis del proyecto.

INICIO

<Inicio> ::== <Formulario><Repetir>

FORMULARIO

<Formulario> ::== tk_Formulario tk_GuionCurvo tk_MayorQue tk_MayorQue tk_CorcheteI <Bloque Instruccion> tk_CorcheteD

REPETIR

<Repetir> :== <Formulario><Repetir>

| <VACÍO>

BLOQUE INSTRUCCION

<Bloque Instruccion> ::== <Cuerpo Instrucción>

| <Cuerpo Instrucción> tk_Coma <Bloque Instrucción>

CUERPO INSTRUCCION

<Cuerpo Instrucción> ::== tk_MenorQue <Bloque Elementos> tk_MayorQue

BLOQUE ELEMENTOS

<Bloque Elementos> ::== <Elemento>

| <Elemento> tk_Coma <Bloque Elementos>

ELEMENTOS

<Elemento> ::== <tipo>

| <valor>

| <fondo>

| <valores>

| <evento>

TIPO

<tipo> ::== tk_tipo tk_DosPuntos tk_Cadena

VALOR

<valor> ::== tk_valor tk_DosPuntos tk_Cadena

FONDO

<fondo> ::== tk_fondo tk_DosPuntos tk_Cadena

VALORES

<valores> ::== tk_valores tk_DosPuntos tk_CorcheteI <Bloque valores> tk_CorcheteD

BLOQUE VALORES

<Bloque valores> ::== tk_Cadena

| tk_Cadena tk_Coma <Bloque valores>

EVENTO

<evento> ::== tk_evento tk_DosPuntos tk_MenorQue tk_entrada tk_MayorQue

| tk_evento tk_DosPuntos tk_MenorQue tk_info tk_MayorQue