Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ingeniería En Ciencias y Sistemas

Organización de Lenguajes y Compiladores 1

Segundo Semestre de 2019

Catedráticos: Ing. Mario Bautista, Ing. Manuel Castillo, Ing. Kevin Lajpop

Tutores académicos: Nery Galvez, Miguel Ruano, Erick Tejaxún

Práctica 1

Tabla de contenido

Objetivos

Objetivo general

Que el estudiante aplique los conocimientos adquiridos en el curso Lenguajes Formales y de Programación y conceptos del curso Organización de Lenguajes y Compiladores 1 para implementar una solución a un problema dado.

Objetivos específicos

- Que el estudiante utilice una herramienta para generar analizadores léxicos y sintácticos.
- Que el estudiante recolecte información a través de la fase de análisis léxico y la fase de análisis sintáctico.
- Que el estudiante sea capaz de manipular la información recolectada de la fase de análisis léxico y la fase de análisis sintáctico y que realice validaciones semánticas básicas.

Descripción

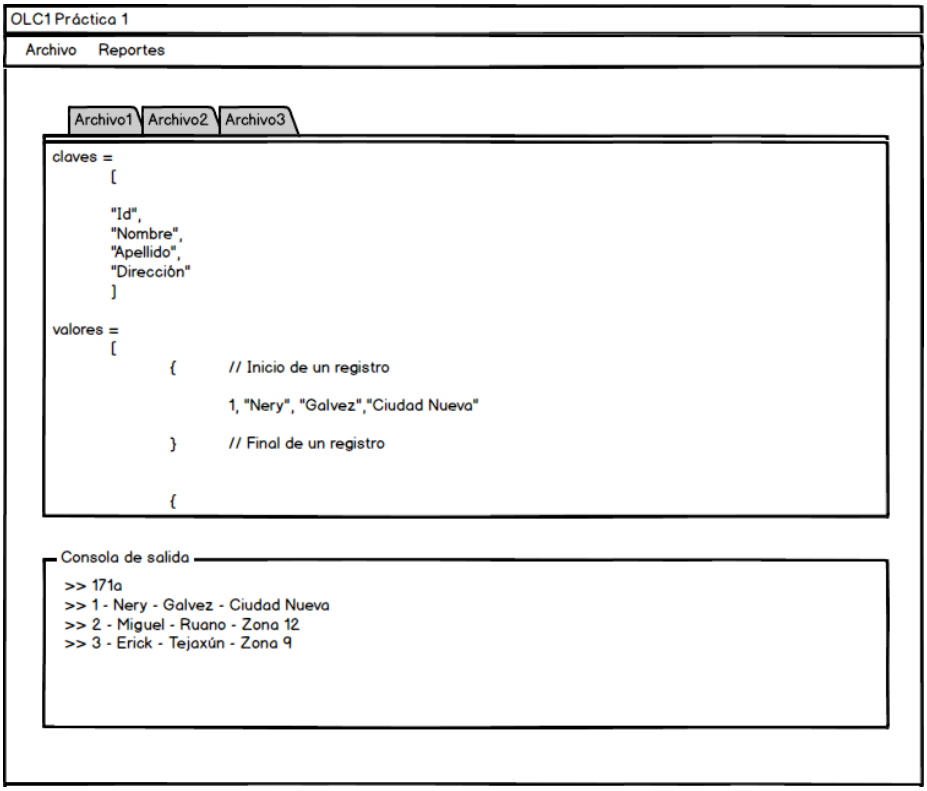
La práctica número 1 del laboratorio consiste en que el estudiante implemente una solución que sea capaz de analizar y recolectar información de un archivo con una sintaxis definida, la sintaxis y su contenido es definido más adelante. Así mismo se debe de implementar un pequeño editor para el manejo de los archivos que se quieren analizar, las funcionalidades de este también son definidas más adelante.

Descripción del editor

La solución debe de contar con un editor para un manejo más eficiente de los archivos que se quieren analizar. Las funcionalidades son las siguientes:

Área de edición

La aplicación debe de poder manejar múltiples pestañas para poder tener múltiples archivos al mismo tiempo.



Manejo de archivos

- Abrir: se seleccionará un archivo y se cargará su contenido a una nueva pestaña dentro del editor.
- Guardar: guardará el archivo y todo su contenido.
- Guardar como: se creará un nuevo archivo con todo el contenido de la pestaña actual.

Reportes

- Reporte HTML de los errores léxicos, sintácticos y semánticos.

Consola de salida

Se contará con un espacio donde se mostrará el resultado de la función Imprimir.

Definición del lenguaje

El lenguaje cuenta con dos tipos de archivos, el archivo de datos y el archivo de reportes, estos se definen a continuación.

Tipos de datos primitivos

Tipo	Descripción	Ejemplo
Cadena	Este tipo de dato acepta un conjunto de caracteres encerrados dentro de comillas dobles	“Hola mundo” “a” “”
Numérico	Este tipo de dato acepta valores tanto enteros como decimales	193 -8.8371 -37 0.8161

Comentarios

Un comentario es un componente léxico que no es tomado en cuenta en la fase de análisis sintáctico.

Comentario de una línea

Comentario que iniciará con los caracteres “***//***” y finalizará con un carácter de fin de línea.

```
//Esto es un comentario de una línea

// Ejemplos de la definición de los registros de un archivo

Registros = [      {1024} // Registro 1

                ]
```

Comentario de múltiples líneas

Comentario que iniciará con los caracteres “***/**”* y finalizará con los caracteres “****/***”

```
/*******************************************************

-----> Esto es un comentario de varias líneas

-----> Ejemplos de la definición de los registros de un archivo

*******************************************************/

Registros = [ {1024} // Registro 1]
```

Archivo de datos

Este archivo contendrá un conjunto de claves y un conjunto de registros con una lista de valores, para un valor en específico siempre existirá una clave asociada. La estructura de este archivo es la siguiente:

```
// Definiendo las claves

Claves = [ Clave1, Clave2, …, Claven]

/* Se definen diferentes registros */

Registros = [

                {val1, val2, …, valn} // Registro 1

                {val1, val2, …, valn} // Registro 2

                {val1, val2, …, valn} // Registro 3

        ]
```

Definición de las claves de un archivo

Una clave sería el equivalente a definirle un encabezado a una columna de una tabla. Las claves que se definan para un archivo siempre serán de tipo Cadena y la sintaxis para definirlas es la siguiente:

```
// Ejemplos de la definición de las claves de un archivo

Claves = [ “Id” ] // solo tendría una clave

Claves = [ “Id”, “Nombre”, “Apellido” ] // se tendrían 3 claves
```

Definición de los registros de un archivo

Un registro es una lista de valores separados por coma encerrados dentro de llaves. El valor puede ser de tipo Cadena o tipo Numérico y la sintaxis para definirlos es la siguiente:

```
/* Ejemplos de la definición de los registros de un archivo */

Registros = [

                {1024} // Registro 1

        ]

Registros = [

                {1, “Nery”, “Galvez”} // Registro 1

                {2, “Miguel”, “Ruano”} // Registro 2

                {3, “Erick”, “Tejaxún”} // Registro 3

        ]
```

Posibles errores en el archivo de datos

1. El primer registro indicará el tipo de dato de las claves, se debe de marcar un error semántico si los siguientes registros tienen tipos de datos distintos en sus valores.
2. Si un registro tiene más o menos valores que la cantidad de claves definidas se marcará un error semántico.

De ocurrir algún error con un registro no se tomará en cuenta y se continuará a evaluar el siguiente registro si existiera.

A continuación, se muestra un ejemplo:

```
Claves = [ “Nombre”, “Carrera”, “Créditos”]

Registros =

        [

                {// Inicio del primer registro, el cual me dirá el tipo de dato de las claves definidas

                        “Persona1” // La clave Nombre será de tipo Cadena

                        , 08 // La clave Carrera será de tipo Numérico

                        , 200 // La clave Créditos será de tipo Numérico

                } // Fin del primer registro

                // Definiendo más registros

                {

                        100 // Sería un error semántico porque la clave Nombre es de tipo Cadena

                        , 09 // Este sí tendría el tipo correcto

                        , 110 // Este sí tendría el tipo correcto

                }

                {“Persona3”, 03, 135} // Este registro sí está bien definido

                {“Persona4”, 02, 243} // Este registro sí está bien definido

                {100, 200} // Error, la cantidad de valores del registro es incorrecta



        ]
```

Archivo de reportes

El archivo de reportes contendrá una serie de instrucciones las cuales permitirán manipular la información obtenida de un archivo de datos. A continuación, se definen las instrucciones del lenguaje.

Declaración de variables

Para la manipulación de la información se podrán declarar variables que almacenarán valores específicos según su tipo de dato. Su definición sintáctica es la siguiente:

```
TIPO id = FUNCION ;
```

Ejemplo:

```
// Variable de tipo Archivo, almacena toda la información que contiene un archivo de datos

Archivo datos1 = leerArchivo("archivo_datos.dat");

// Función que suma todos los valores que pertenecen a la clave id y que están en el archivo datos1

Numerico suma_id = Sumar( datos1, "Id");

// Imprimiendo los resultados

Imprimir( datos1);

Imprimir(suma_id);
```

Función leerArchivo

La función **leerArchivo** permitirá cargar a memoria toda la información contenida en un archivo de datos para su posterior manipulación. La función recibirá los siguientes parámetros:

1. Ruta: cadena con la ruta relativa donde se encuentra el archivo de datos que se quiere cargar a memoria.

Sintaxis:

```
leerArchivo (Ruta)
```

Ejemplo:

```
Archivo datos1 = leerArchivo("control1.dat");

Archivo datos2 = leerArchivo("Calificacion/entradas/configuraciones.dat");
```

Nota: las estructuras donde se almacenará la información quedan a discreción del estudiante.

Función Sumar

La función **Sumar** permitirá obtener la suma de todos los valores que pertenezcan a una clave en específico. La función recibirá los siguientes parámetros:

1. Identificador: nombre de una variable de tipo Archivo.
2. Clave: cadena con el nombre de una clave, la cual será utilizada para realizar la suma de los valores correspondientes a dicha clave.

Sintaxis:

```
Sumar (Identificador, Clave)
```

Ejemplo:

```
Numerico suma = Sumar(datos1, “Edades”);

Imprimir (suma);
```

Función Contar

La función **Contar** devolverá el número de registros que contiene un archivo. La función recibirá los siguientes parámetros:

1. Identificador: nombre de una variable de tipo Archivo.

Sintaxis:

```
Contar(Identificador)
```

Ejemplo:

```
Numerico numRegistros = Contar(archivo1);

Imprimir(numRegistros);
```

Función Promedio

La función **Promedio** calculará el promedio de todos los valores que pertenezcan a una clave en específico. La función recibirá los siguientes parámetros:

1. Identificador: nombre de una variable de tipo Archivo.
2. Clave: cadena con el nombre de una clave, la cual será utilizada para calcular el promedio de los valores correspondientes a dicha clave.

Sintaxis:

```
Promedio(Identificador, Clave)
```

Ejemplo:

```
Numerico prom = Promedio(datos1, “correlativo”);

Imprimir(prom);
```

Función ContarSi

La función **ContarSi** permitirá verificar cuántos registros tienen un valor específico en una clave determinada. La función recibirá los siguientes parámetros:

1. Identificador: nombre de una variable de tipo Archivo.
2. Clave: cadena con el nombre de una clave, la cual será utilizada para saber qué valor del registro será utilizado para la comparación.
3. Operador relacional: operador que indica qué tipo de comparación se realizará para verificar si el valor del registro cumple o no la condición. Los operadores permitidos serán los siguientes:

>	<
>=	<=
==	!=

4. Valor: expresión de tipo numérico o cadena la cuál contendrá el valor que deben de tener los registros para que la condición sea verdadera.

Sintaxis:

```
ContarSi(Identificador, Clave, Operador_Relacional, Valor)
```

Ejemplo:

```
Numerico mayoresEdad = ContarSi(datos1, “Edad”, >=, 18);

Imprimir(mayoresEdad);

Numerico ingenieros = ContarSi(datos1 , “Categoría”, ==, “Ingeniero”);

Imprimir(ingenieros);
```

Función ObtenerSi

La función **ObtenerSi** permitirá obtener la información de los registros que cumplen con un valor específico en una clave determinada. A diferencia de la función **ContarSi** la cual solo obtiene cuántos registros cumplen con una cierta condición, la función **ObtenerSi** obtendrá una cadena con toda la información de los registros que cumplan la condición definida, el formato de salida de esta cadena es la siguiente:

```
[
        { id : 1 , Nombre : “Nery”, Apellido : “Galvez”, Direccion : “Zona 2, Guatemala”}
        { id : 5 , Nombre : “Julio”, Apellido : “Ruano”, Direccion : “Zona 3, Guatemala”}
        { id : 9 , Nombre : “Rodrigo”, Apellido : “Chávez”, Direccion : “Zona 15”}
]
```

La función recibirá los siguientes parámetros:

1. Identificador: nombre de una variable de tipo Archivo.
2. Clave: cadena con el nombre de una clave, la cual será utilizada para saber qué valor del registro será utilizado para la comparación.
3. Operador relacional: operador que indica qué tipo de comparación se realizará para verificar si el valor del registro cumple o no la condición. Los operadores permitidos serán los siguientes:

>	<
>=	<=
==	!=

4. Valor: expresión de tipo numérico o cadena la cuál contendrá el valor que deben de tener los registros para que la condición sea verdadera.

Sintaxis:

```
ObtenerSi(Identificador, Clave, Operador_Relacional, Valor)
```

Ejemplo:

```
Cadena mayoresEdad = ObtenerSi(datos1, “Edad”, >=, 18);

Imprimir(mayoresEdad);

Cadena ingenieros = ObtenerSi (datos1 , “Categoría”, ==, “Ingeniero”);

Imprimir(ingenieros);
```

Función Imprimir

La función **Imprimir** permitirá mostrar en la consola de la aplicación el valor de una expresión. La función recibirá los siguientes parámetros:

1. Expresiones: será una lista de expresiones de tipo numérico o cadena separadas por coma.

Sintaxis:

```
Imprimir(Expresiones);
```

Ejemplo:

```
Imprimir(“/*************** Calificación Compiladores 1 ***************/”);

Imprimir(“Mayores de edad = ” , variable1);

Imprimir(“Cantidad de ingenieros = “, variable2, “ Cantidad de arquitectos = “, variable3);
```

Función Graficar

La función **Graficar** generará una imagen en formato jpg de una gráfica de barras. La función recibirá los siguientes parámetros:

1. Nombre: cadena con el nombre que recibirá la imagen a generar.
2. Título: cadena con el título de la gráfica de barras a generar.
3. Identificador: nombre de una variable de tipo Archivo.
4. ValoresX: cadena con el nombre de una clave, la cual será utilizada para saber el título del eje X de la gráfica y para saber de qué clave se tomarán los datos del eje X de la gráfica.
5. ValoresY: cadena con el nombre de una clave, la cual será utilizada para saber el título del eje Y de la gráfica y para saber de qué clave se tomarán los datos del eje Y de la gráfica.

Sintaxis:

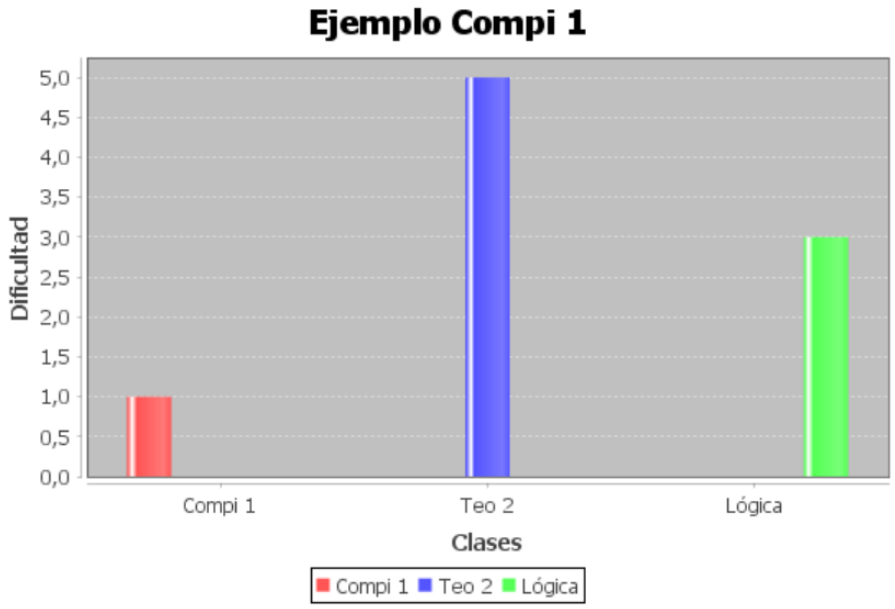
```
Graficar (Nombre, Titulo, Identificador, ValoresX, ValoresY);
```

Ejemplo:

```
Archivo datos1 = leerArchivo("Registros.dat");

Graficar("Imagen1", "Ejemplo Compi 1", datos1, "Clases", "Dificultad");
```

Imagen generada



Palabras reservadas

Claves	Registros	Archivo	leerArchivo
Numerico	Sumar	Contar	Promedio
ContarSi	Cadena	ObtenerSi	Imprimir
Graficar			

Librería para graficar

Se recomienda utilizar la librería ***JFreeChart***. Se adjunta un tutorial de cómo utilizar dicha librería: https://www.tutorialspoint.com/jfreechart/index.htm

Consideraciones y restricciones

- Se deberá de trabajar de manera individual utilizando las herramientas JFlex y Cup de Java.
- El sistema operativo y el IDE quedan a elección del estudiante.
- El lenguaje NO es case sensitive, lo que quiere decir que el analizador no encuentra diferencia entre Claves y claVES.
- No se calificará solo la implementación de los analizadores léxicos y sintácticos, se debe de poder ejecutar las instrucciones definidas para el lenguaje.
- La calificación se realizará sobre los archivos ejecutables de la aplicación, de no cumplir con esto no se tendrá derecho a calificación.
- Copias de proyectos tendrán de manera automática una nota de 0 puntos y serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas los involucrados.

Entregables

- Código fuente de la aplicación.
- Ejecutable de la aplicación.
- Manual técnico.
- Manual de usuario.

Forma de entrega

- La entrega será por medio de la plataforma Classroom.
- No se calificarán proyectos que sean entregados después de la hora límite.
- Se debe de entregar todo lo necesario en una carpeta con formato Zip con el nombre: carné_Práctica1.zip
- Fecha y hora de entrega:

15 de agosto de 2019 antes de las 23:59 horas