Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
Segundo Semestre 2023
Catedráticos:
Ing. Mario Bautista
Ing. Manuel Castillo
Ing. Kevin Lajpop
Tutores académicos:
Walter Guerra
Kevin López
Maynor Piló
Jose Perez

StatPy Convertor

Proyecto 1

Tabla de contenido

1. Objetivo General

Aplicar los conocimientos sobre las fases de análisis léxico y sintáctico de un compilador para la construcción de una solución de software.

2. Objetivos Específicos

- Que el estudiante aprenda a generar analizadores léxicos y sintácticos utilizando las herramientas de JFLEX y CUP.
- Que el estudiante comprenda los conceptos de token, lexema, patrones y expresiones regulares.
- Que el estudiante pueda realizar correctamente el manejo de errores léxicos.
- Que el estudiante sea capaz de realizar acciones gramaticales usando el lenguaje de programación JAVA.

3.Descripción

El proyecto "StatPy Convertor" es una solución de software que consiste en aplicar los conocimientos sobre las fases de análisis léxico y sintáctico de un compilador para ofrecer dos funcionalidades principales: generación de reportes estadísticos y traducción de código de StatPy a Python.

Debido a lo tedioso que puede llegar a ser el sobrescribir una aplicación de un lenguaje a otro, se le solicita a los estudiantes de sistemas que desarrollen un traductor de lenguaje de programación StatPy a Python. Este traductor abarca algunas de las sentencias básicas del lenguaje, como la declaración y asignación de variables, sentencias de retorno, sentencias de condición y repetición, métodos, entre otros.

Además, se solicita que el programa pueda generar reportes estadísticos, con datos que se puedan leer desde archivos JSON (objetos con datos),permitiendo una mejor comprensión de los datos procesados durante el proceso de traducción.

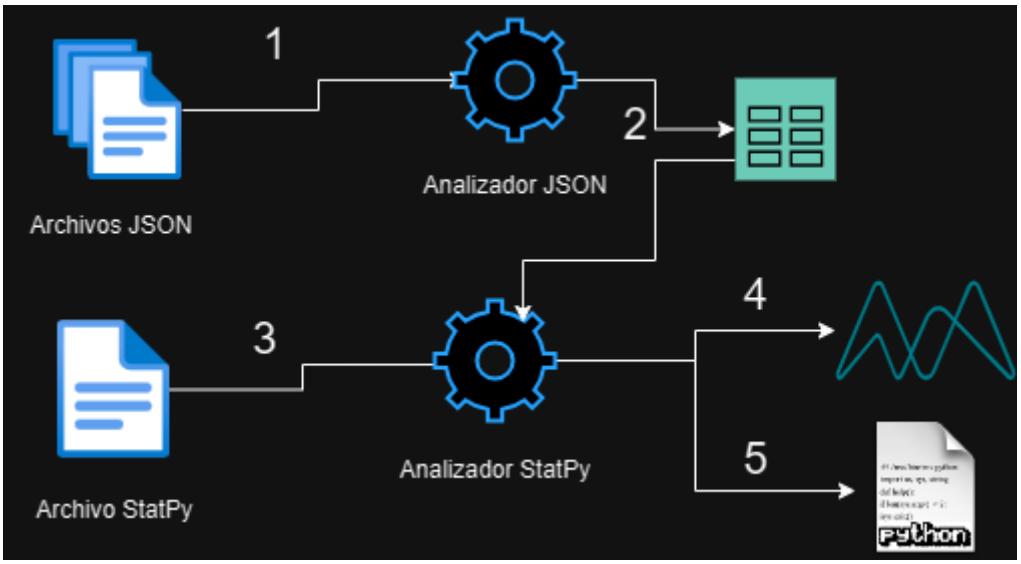
4. Descripción de la Solución

El proyecto tiene como objetivo desarrollar una aplicación que ofrezca dos funcionalidades principales: generación de reportes estadísticos y traducción de código de StatPy a Python. Para lograr esto, se utilizarán dos analizadores diferentes: uno para analizar archivos JSON y otro para traducir el código StatPy y generar reportes estadísticos.

El analizador JSON estará encargado de leer archivos JSON (se podrá cargar varios archivos) que contengan objetos con datos. Los datos podrían incluir información como números, nombres, etc. Estos datos deben ser almacenados por archivo para usarlos con el siguiente analizador.

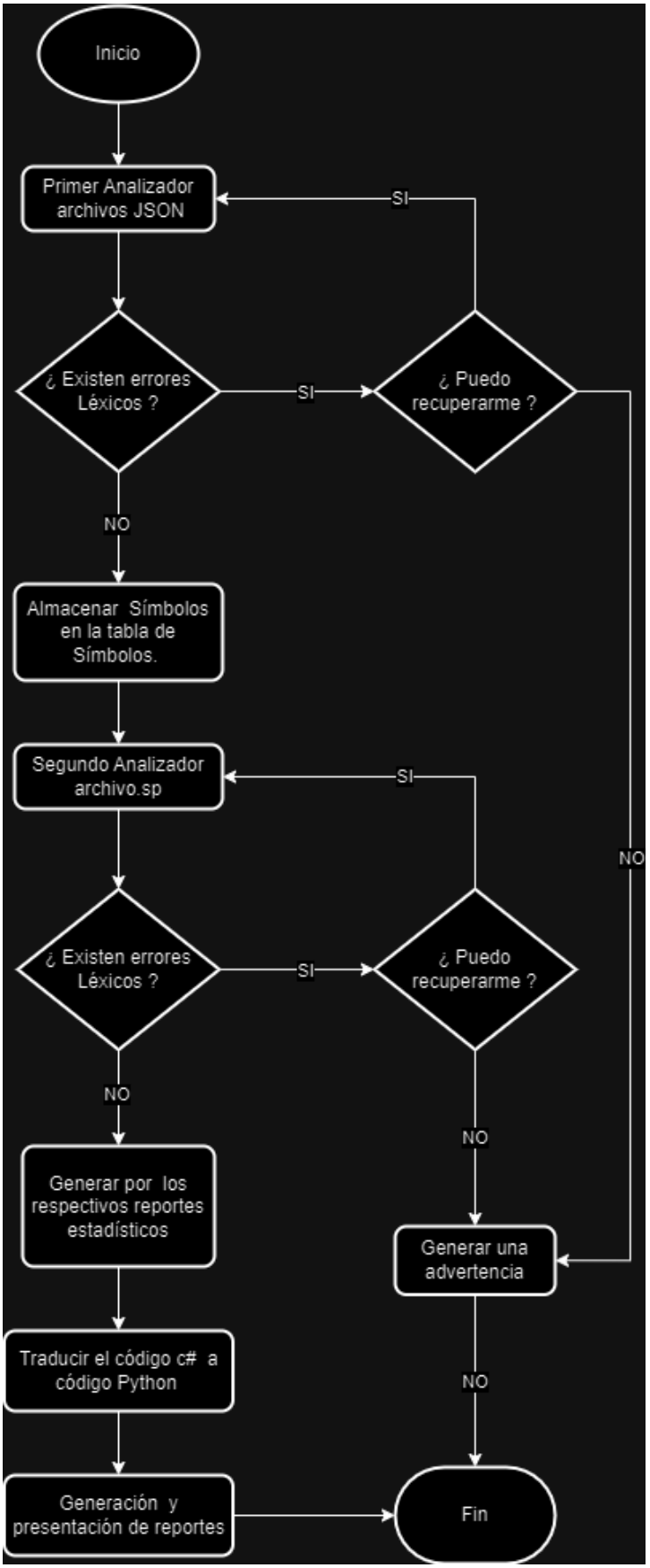
El segundo analizador se encargará de traducir el código fuente escrito en StatPy a su equivalente en Python. Adicionalmente la aplicación cuenta con sus propias funciones de generación de reportes este analizador deberá ser capaz de generar esos respectivos reportes

5. Resumen del funcionamiento



1. El estudiante ingresa los archivos JSON para ser analizados
2. Los símbolos se almacenarán por archivo en tablas de símbolos donde estos se utilizaran en el analizador StatPy
3. El estudiante ingresa el archivo con extension .sp para que sea analizado por el analizador StatPy
4. El analizador Statpy deberá de generar por medio de las funciones los respectivos reportes estadísticos
5. El analizador Statpy deberá de traducir el código StatPy (exceptuando las funciones estadísticas) a código python

6. Flujo recomendado



7. Área de Edición y Resultados

7.1 Editor de Texto:

El editor será el entorno de trabajo para el manejo de entradas, salidas y analizadores de StatPy Convertor. Su función principal será el ingreso de código fuente el cual será analizado. El código podrá ingresar directamente en el editor o mediante un archivo. El diseño de la interfaz queda a discreción del estudiante.

Interfaz Sugerida

7.2 Funcionalidades:

- **Abrir Archivo:** El editor debe tener la capacidad de abrir archivos con las extensiones .sp y .json cuyo contenido se mostrará en la entrada.
- **Guardar:** Se debe permitir guardar la entrada sobre el archivo en el que se está trabajando.
- **Guardar Como:** Se debe permitir guardar la entrada en un nuevo archivo.
- **Cambiar Analizador:** Se debe tener la opción de elegir el analizador al cual se le va a enviar el programa fuente. Siendo estos:
    - Analizador 1: StatPy
    - Analizador 2: JSON

7.3 Herramientas:

- **Editor de Entrada:** Se debe contar con un área de texto que permita ingresar y modificar el programa fuente.
- **Consola de Salida:** Mostrará los resultados de la traducción de StatPy. La consola debe tener la restricción de no ser editable.
- **Ejecutar:** Se encarga de enviar el programa fuente al analizador que esté seleccionado.

7.4 Reportes:

de json

- **Reporte de Tokens:** Mostrará el reporte de Tokens obtenidos.
- **Reporte de Errores Léxicos:** Mostrará el reporte de Errores Léxicos.

de json

8. Especificación del programa fuente

8.1 Case insensitive

El lenguaje no distinguirá entre mayúsculas o minúsculas.

```
void DefinirGlobales(){
//Instrucciones
}
void definirglobales(){
//Instrucciones
}
```

Nota: Ambos casos son lo mismo.

8.2 Comentarios

8.2.1 Comentarios de una línea

Este tipo de comentario comenzará con // y deberá terminará con un salto de línea

8.2.2 Comentarios multilínea

Este tipo de comentario comienza con /* y deberá de terminar con */.

```
// Esto es un comentario de una sola línea.
/* Esto es un comentario
multilínea */
```

8.3 Tipos de Dato

Dado que el lenguaje python es de tipado dinámico este no contiene una palabra reservada específica para cada tipo de dato por lo cual cualquier tipo de dato será sustituido por ‘var’

Nota: Esto aplica para todas las declaraciones exceptuando las declaraciones en las funciones estadísticas

StatPy	Python
int	
double	
char	
bool	
string	

8.4 Declaración y asignación de Variables

StatPy	TipoDato ID; TipoDato ID = Expresion;
Salida - Python	id = Expresion

8.5 Expresión

Dentro de una expresión pueden venir operadores aritméticos, llamadas a funciones, llamadas a variables, cadenas, etc.

Ejemplo:

Lenguaje	Ejemplo
StatPy	int var1 = 5+8*9;
Python	var1 = 5+8*9

8.6 Declaración del main

Método inicial donde las sentencias será todas las instrucciones de StatPy

Lenguaje	Ejemplo

StatPy	void main ( ){
	<Sentencias>
	}

Python	def main( ): <Sentencias> if__name__ = “__main__”: main()

8.7 Sentencias de Control

- ***if***

Lenguaje	Ejemplo

StatPy	if (b > a){
	  Console.Write("b mayor que a");
	}else if(a == b){
	  Console.Write("a y b son iguales");
	}

Python	if b > a:
	  print("b mayor que a")
	elif a == b:
	  print("a y b son iguales")

- ***switch***

Lenguaje	Ejemplo

StatPy	switch(valor){
	  case 1:
	  precio = 55;
	  break;
	  case 2:
	  precio = 25;
	  case 3:
	  precio = 40;
	  default:
	  Console.Write("No válido. Escoja 1, 2, o 3.");
	}

Python	def switch(case, precio):
	  switcher = {
	  1: precio = 55,
	  2: precio = 25,
	  3: precio = 40,
	  4: print("No válido. Escoja 1, 2, o 3."),
	  }

8.8 Sentencias de Repetición

- ***for***

Lenguaje	Ejemplo

StatPy	for (int a=0; a<10; a++){
	  Console.Write("el valor de a es: " + a);
	}

Python	for a in range(1,10): print(“el valor de a es: ”, a)

- ***while***

Lenguaje	Ejemplo

StatPy	while(a < 10){
	  Console.Write("el valor de a es: " + a);
	}

Python	while a < 10 : print("el valor de a es:”, a)

- ***do-while***

Lenguaje	Ejemplo

StatPy	int a = 1;
	do {
	  Console.Write("el valor de a es: " + a);
	} while(a < 5);

Python	i = 1
	while True:
	  print("el valor de a es: ", a)
	  a = a + 1
	  if (a < 5):
	  break

8.9 Sentencia Imprimir

La traducción de la sentencia imprimir en consola debe ser traducida de la siguiente manera:

Entrada - StatPy	Console.Write("el valor de a es: " + a);
Salida - Python	print("el valor de a es: ", a)

8.10 Operadores para las Expresiones

Las expresiones manejan operadores Aritméticos, Relacionales y Lógicos. La traducción de los operadores debe realizarse con las siguientes reglas.

Aritméticas:

Los operadores aritméticos son iguales en ambos lenguajes.

Operación	Entrada - StatPy	Salida - Python
Suma	+	+
Resta	-	-
Multiplicación	*	*
División	/	/

Relacionales:

Los operadores relacionales son iguales en ambos lenguajes.

Operación	Entrada - StatPy	Salida - Python
Mayor	>	>
Menor	<	<
Mayor igual	>=	>=
Menor igual	<=	<=
Igual	==	==

Distinto	!=	!=

Lógicas:

Las operaciones lógicos deben generar su traducción en base a las siguientes reglas:

Operación	Entrada - StatPy	Salida - Python
AND	&&	and
OR	||	or
NOT	!	not

8.11 Declaración de funciones

Las funciones **NO deberán ser traducidas a python** luego de ser analizadas. El lenguaje maneja funciones específicas enfocadas en la ejecución de gráficas estadísticas. Las funciones manejan la siguiente estructura:

```
void <Nombre_Funcion> ( ) {
    <Sentencias>
}
```

Nota: El detalle de cómo se ejecutan las funciones se especifica en la sección de Funciones Estadísticas.

9. Archivos JSON

9.1 Lectura archivos JSON

En esta sección se utilizará para hacer lectura de 1 o varios archivos JSON para ser almacenados en tablas de símbolos por archivos

Nota: Esta totalmente prohibido usar herramientas/librerías para hacer lectura de estos archivos

Ejemplo de estructura de archivo JSON

```
{

    ”<nombre>”:<valor>,
    ”<nombre>”: ”<cadena>”

}
```

Los únicos valores que vendrán en estos archivos serán de tipo double y de tipo cadena Ejemplo

```
{

    ”TituloX”:”Grafica1”,
    ”valor1”: 5.9

}
```

Cada símbolo se deberá guardar en una tabla de símbolos por archivo para que pueda ser utilizado en las funciones estadísticas

9.2 Obtención de símbolos en archivo StatPy

Para poder obtener los símbolos cargados de los archivos json y utilizarlos en las variables de las funciones estadísticas de StatPy se hará a través del uso de la siguiente sintaxis:

```
${ ”<NewValor>”,  ”<nombreArchivo.json>”,  ”<nombreSimbolo>”}
```

Nota: En la funciones estadísticas se mostrarán ejemplos del uso de esta sintaxis para obtener los símbolos de los archivos json previamente cargados

10. Funciones estadísticas

10.1 Definir Variables Globales

Está sección se utilizará para definir variables globales que podrán ser utilizadas **Únicamente en las funciones estadísticas**

10.1.1 Tipos de datos

En las funciones estadísticas se hará uso de dos tipos de datos para generar una flexibilidad en la cantidad de reportes que podamos manejar, entre ellos los siguientes

Tipo de dato	Descripción
string	Este tipo de dato almacenará únicamente cadenas de caracteres específicamente declaradas en los archivos de entrada(sin operaciones) o valores de variables de los archivos json previamente cargados
double	Este tipo de datos almacenará valores decimales específicamente declarados en la entrada (sin operaciones) o valores de variables de los archivos json previamente cargados

10.1.2 Variables Globales

Esta función permitirá la declaración y uso de variables globales, que será la primera función en ser declarada, estas variables sólo podrán ser de los tipos definidos en la tabla anterior y podrán utilizarse **Únicamente en las funciones estadísticas**, el nombre de la función es fija, estas se deben guardar en una nueva tabla de símbolos para ser utilizadas en las otras funciones

```
void DefinirGlobales()
{
    string firstReport = “Probabilidad general esperada”;
    double generalExpected = 0.8;
    string secondReport = “Probabilidad general esperada”;
    double generalExpected2 = 0.2:
}
```

10.2 Gráficas estadísticas

Tal como se mencionó en la descripción de la solución es necesario que la aplicación sea capaz de generar reportes estadísticos a través de las funciones de StatPy definidas a continuación

Nota: Se recomienda hacer uso de la librería JFreeChart para la generación de los reportes estadísticos.

Gráfica de Barras

La gráfica de barras se definirá a través de un conjunto de atributos los cuales se encontrarán acotados por la siguiente sintaxis:

```
void GraficaBarras(){
        //CARACTERÍSTICAS
}
```

```
<Tipo><Característica> = <Valor> ;
```

Donde característica es alguna de las características definidas a continuación, el valor dependerá de cada característica y el tipo de la característica. Las características podrán venir en cualquier orden. La sintaxis es la siguiente:

Características de la gráfica de Barras

Título

1. Palabra reservada: Título
2. Valor esperado: string o variable global
3. Descripción: título que se mostrará en la parte superior de la gráfica.

```
string Titulo = var1;

string  titulo = “Puntaje General”;
```

EjeX

1. Palabra reservada: EjeX
2. Valor esperado: lista de Strings o variables globales
3. Descripción: será una lista de valores de tipo string o variables globales de tipo string separadas por comas y encerradas entre {}. Esta

sentencia solo vendrá una vez dentro de una declaración de gráfica de barras y funcionará como un arreglo de tamaño indefinido.

```
string [] Ejex = {“Clase 1”, var1, “Método 1”};

string [] Ejex = {“Clase 2”, var2};

string [] Ejex = {var1, var2, var3};
```

Valores

1. Palabra reservada: Valores

2. Valor esperado: lista de valores decimales o variables globales

3. Descripción: será una lista de valores de tipo double o variables globales de tipo double separadas por comas y encerradas entre {}. Esta sentencia solo vendrá una vez dentro de una declaración de gráfica de barras y funcionará como un arreglo de tamaño indefinido.

```
double [] Valores= { 3.5, var1, var2 };

double [] Valores= { ${ NewValor, “archivo1.json”, “valor1”}, 0.3, 1.1];

double [] Valores= {var1, 0.5, ${ NewValor, “archivo1.json”, “valor2”} ];
```

Titulo Eje X

1. Palabra reservada: TituloX
2. Valor esperado: String o variable global
3. Descripción: título que se mostrará en la parte inferior de la gráfica.

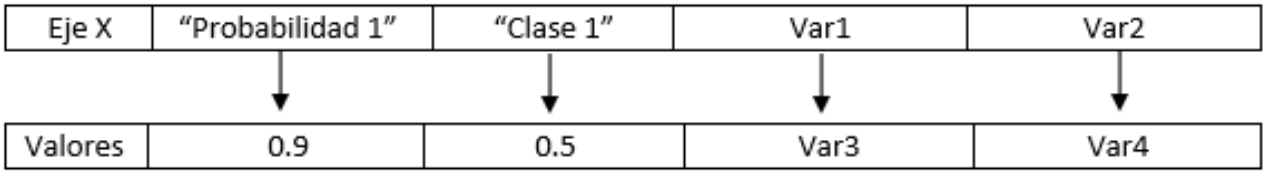
```
string TituloX= “Titulo Eje X”;

string TituloX= var1;
```

Titulo Eje Y

1. Palabra reservada: TituloY
2. Valor esperado: String o variable global
3. Descripción: título que se mostrará en la parte izquierda de la gráfica.

```
string TituloY= “Titulo Eje Y”;

string Tituloy= var1;
```

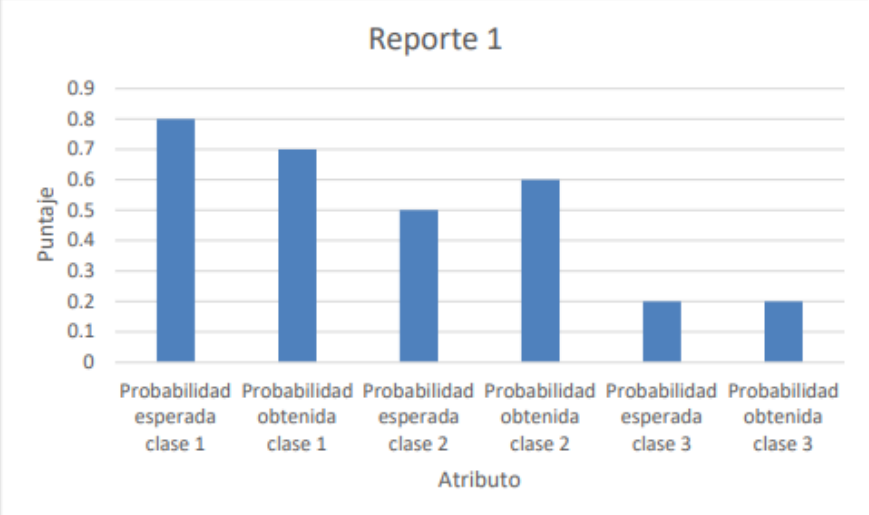
Relación Eje X y Valores



Ejemplo del funcionamiento

```
void DefinirGlobales(){
        string reporte1 = “Reporte 1”;
        double pe1 = 0.8;
        double pe2 = 0.5;
        double pe3 = 0.2;
        double po1 = ${ NewValor, “archivo2.json”, “valor1”};
        double po2 = ${ NewValor, “archivo1.json”, “valor1”};
        string vart = “Valor Obtenido”;
        string var2 = “Valor Esperado clase 1”;
        string var22 = “Valor Obtenido clase 1”;
        string var3 = “Valore Esperado clase 2”;
        string var3 = “Valor Obtenido clase 2”;

}
```

```
void GraficaBarras(){

        string Titulo= reporte1;

 string [] Ejex= { “Probabilidad Esperada clase 1”, “Probabilidad Obtenida Clase 1”, var2, var22, var3, var33};

double [] Valores= { pe1, po1, pe2, po2, pe3, ${ NewValor, “archivo1.json”, “valor2”} };

string TituloX= “Atributo”;

string TituloY= “Puntaje”;

}
```



10.3 Gráfica de Pie

La gráfica de pie se definirá a través de un conjunto de atributos los cuales se encontrarán acotados por la siguiente sintaxis:

```
void GraficaPie(){
        //CARACTERÍSTICAS
}
```

```
<Tipo><Característica> = <Valor> ;
```

Donde característica es alguna de las características definidas a continuación y valor dependerá de cada característica. Las características podrán venir en cualquier orden. La sintaxis es la siguiente:

Características de la gráfica de Pie

Titulo

1. Palabra reservada: Titulo
2. Valor esperado: string o variable global
3. Descripción: título que se mostrará en la parte superior de la gráfica.

```
string Titulo = var1;
string  titulo = “Puntaje General”;
```

EjeX

1. Palabra reservada: EjeX

2. Valor esperado: lista de Strings o variables globales

3. Descripción: será una lista de valores de tipo string o variables globales de tipo string separadas por comas y encerradas entre {}. Esta sentencia solo vendrá una vez dentro de una declaración de gráfica de barras y funcionará como un arreglo de tamaño indefinido.

```
string [] Ejex = {“Clase 1”, var1, “Método 1”};

string [] Ejex = {“Clase 2”, var2};

string [] Ejex = {var1, var2, var3};
```

Valores

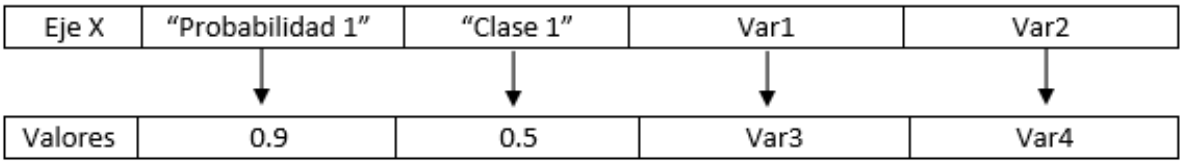
1. Palabra reservada: Valores

2. Valor esperado: lista de valores decimales o variables globales

3. Descripción: será una lista de valores de tipo double o variables globales de tipo double separadas por comas y encerradas entre {}. Esta sentencia solo vendrá una vez dentro de una declaración de gráfica de barras y funcionará como un arreglo de tamaño indefinido.

```
double [] Valores= { 3.5, var1, var2 };

double [] Valores= { ${ NewValor, “archivo1.js”, “valor1”}, 0.3, 1.1];

double [] Valores= {var1, 0.5, ${ NewValor, “archivo1.js”, “valor2”} ];
```

Relación Eje X y Valores



Ejemplo del funcionamiento

```
void DefinirGlobales(){

    string reporte1 = “Reporte 1”;

    double pe1 = 0.8;

    double pe2 = 0.5;

    double pe3 = 0.2;

    double po1 = ${ NewValor, “archivo1.json”, “valor1”};

    double po2 = ${ NewValor, “archivo2.json”, “valor2”};

    string vart = “Valor Obtenido”;

    string var2 = “Valore Esperado clase 1”;

    string var22 = “Valor Obtenido clase 1”;
```

```
        string var3 = “Valore Esperado clase 2”;

        string var3 = “Valor Obtenido clase 2”;

}

void GraficaPie(){

string Titulo= reporte1;

 string [] Ejex=  { “Probabilidad Esperada clase 1”, “Probabilidad Obtenida Clase 1”, var2, var22, var3, var33};

double [] Valores= { pe1, po1, pe2, po2, pe3, ${ NewValor, “archivo1.json”, “valor1”} };

}
```

11. Reportes

11.1 Reporte de tokens

Se deberá generar un reporte en HTML con todos los tokens y lexemas reconocidos durante la fase de análisis léxico. Se debe considerar el siguiente formato:

Lexema	Token	Línea	Columna
DefinirGlobales	Func_Es	5	1
variable1	identificador	12	5
variable2	identificador	25	10

11.2 Reporte de errores léxicos

Se deberá generar un reporte en HTML con todos los errores y lexemas reconocidos durante los análisis correspondientes. Se debe considerar el siguiente formato

Lexema	Descripción	Línea	Columna
°	error léxico	5	1
¬	error léxico	7	10

12. Entregables

12.1 Manual de Usuario

Deberá entregar un manual de usuario en donde se detalle cómo se hace uso de la aplicación, es de vital importancia que se adicional a incluir capturas de pantalla sobre el funcionamiento de la aplicación el estudiante incluya cómo funciona el lenguaje, esto con el objetivo de que en caso de que cualquier usuario haga uso de la aplicación, sepa cómo funciona tanto el lenguaje de generación de reportes como el editor de texto creado para la aplicación.

12.2 Manual técnico

En este manual el estudiante deberá incluir toda la información que considere importante para el caso en el que otro desarrollador desee darle mantenimiento o realizar mejoras en el código fuente, se recomienda incluir versiones de las herramientas utilizadas, especificaciones del entorno en donde se desarrolló la solución, diagramas de clases, etc.

12.3 Archivos de gramática

El estudiante deberá incluir la gramática que utilizo tanto para el lenguaje StatPy como también la gramática que se realizó para analizar los archivos JSON, es de vital importancia que no se realice una copia literal del archivo utilizado para la herramienta CUP, en este archivo deberán escribir las gramáticas con el formato **BNF(Backus-Naur form)**

12.4 Requerimientos mínimos

Para tener derecho a calificación, el estudiante deberá cumplir con los siguientes requerimientos mínimos:

- Carga de los archivos json como los archivos StatPy
- Análisis léxico y sintáctico para los archivos descritos para el proyecto
- Funciones estadísticas
- Manual de Usuario.
- Manual Técnico.
- Archivo de gramáticas.

12.5 Restricciones

- Se debe hacer uso de un repositorio en Github para realizar la entrega de su proyecto, Nombre del repositorio: **[OLC1]Proyecto1_#Carnet** no olvidar invitar a su auxiliar encargado
- Lenguajes de programación a usar: Java
- Herramientas de análisis léxico y sintáctico: JFlex/CUP
- Está totalmente prohibido usar alguna herramienta/librería para leer los archivos json se debe hacer con el analizador correspondiente
- Herramienta para generar gráficas: se puede utilizar cualquier librería (Se recomienda hacer uso de JFreechart)
- **El proyecto es individual**
- Copias completas/parciales de: código, gramática, etc. serán merecedoras de una nota de 0 puntos, los responsables serán reportados al catedrático de la sección y a la Escuela de Ciencias y Sistemas

12.6 Fecha de Entrega

Sábado 16 de septiembre de 2023 la entrega se realizará por medio de UEDI, en caso exista algún problema, se estará habilitando un medio alterno por medio del auxiliar del laboratorio

13 Ejemplo de archivos

Link: https://drive.google.com/drive/folders/1CJzwi9KEd-kRN_Zk-nejZdMDZndfsLuR?usp=sharing