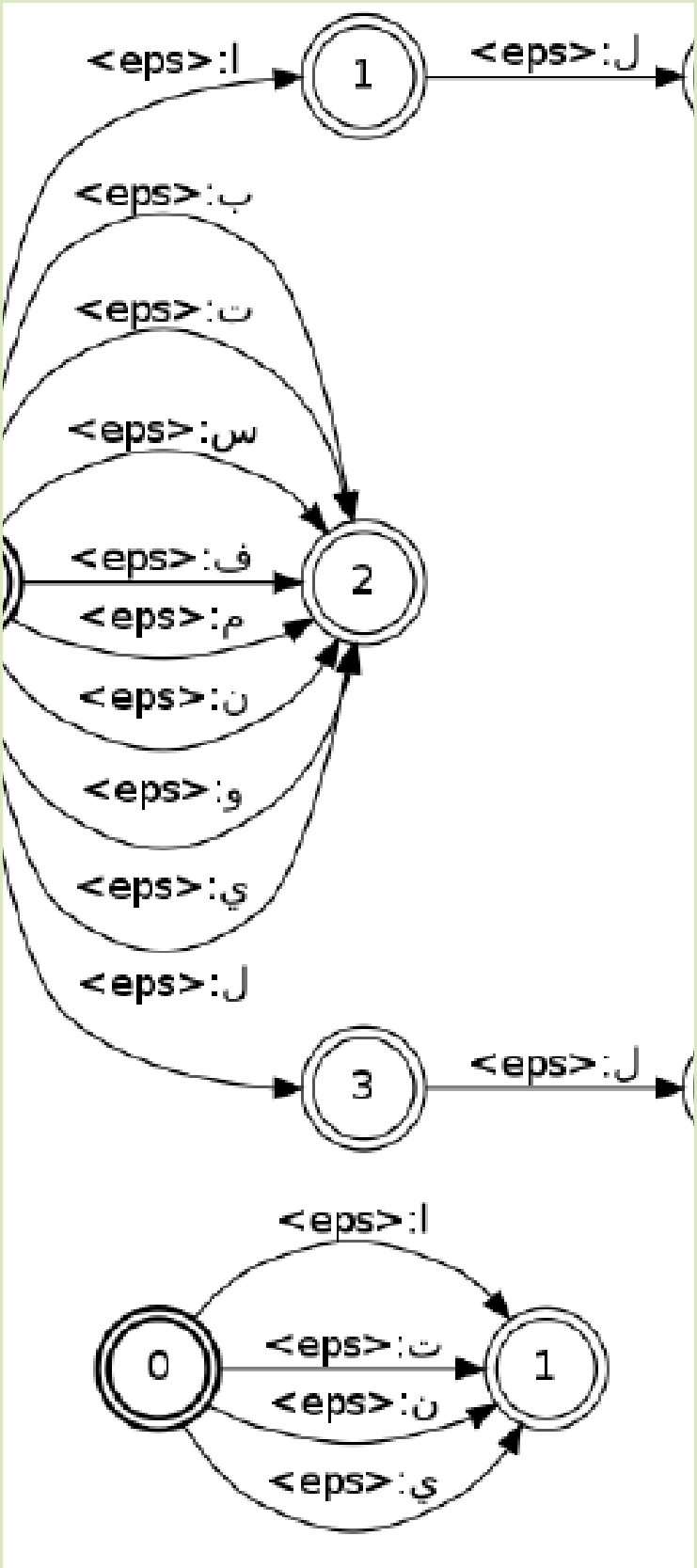
202201131

MANUAL TECNICO



ANALIZADOR LEXICO

ÍNDICE

N/P

OBJETIVO

EL PROPÓSITO DEL CÓDIGO ES ANALIZAR UN CONTENIDO ESTRUCTURADO QUE REPRESENTA CONTINENTES Y PAÍSES, Y GENERAR UN ARCHIVO DE SALIDA EN FORMATO GRAPHVIZ (DOT) PARA REPRESENTAR VISUALMENTE LA SATURACIÓN DE CADA CONTINENTE Y PAÍS EN UN GRÁFICO. EL GRÁFICO SE VISUALIZARÁ EN FUNCIÓN DE UN PORCENTAJE DE SATURACIÓN, REPRESENTANDO ASÍ LA INFORMACIÓN DE FORMA CLARA Y CONCISA.

La finalidad de éste manual técnico es instruir a la persona que quiera administrar, editar o configurar el software usando las debidas herramientas.

¿Claro, el objetivo principal del analizador léxico es leer los caracteres de entrada de un programa fuente y convertirlos en una secuencia de componentes léxicos, también conocidos como tokens1. Estos tokens son utilizados posteriormente por el analizador sintáctico para realizar el análisis sintáctico del código1.
El analizador léxico también realiza algunas funciones secundarias, como eliminar comentarios y espacios en blanco, y asociar mensajes de error con el programa fuente1.

FUNCIONAMIENTO CODIGO

COMPARACIONES CON OTRAS EMPRESAS DEL SECTOR

Variables Principales

- graf: Variable que acumula la representación en formato DOT del gráfico.
- saturacioncontinente: Almacena la saturación total de un continente.
- sumarpaises: Cuenta el número de países dentro de un continente.
- nombrecontinente, nombrepais: Variables que almacenan los nombres actuales de los continentes y países que se están analizando.
- tokens: Array que registra los tokens generados durante el análisis del contenido.

Estructura de Control

- Se utiliza un bucle do while que itera a través de cada carácter del contenido, permitiendo el análisis caracter por caracter.
- A través de estructuras if y select case, el código maneja diferentes tipos de caracteres (por ejemplo, saltos de línea, espacios, llaves, y otros caracteres especiales) y determina la acción apropiada.

Generación del Gráfico

- Cuando se identifica un continente ('continente'), se calcula su saturación en relación con el número de países y se genera un nodo en el gráfico en función de la saturación:
- Colores: Dependiendo del valor de saturación (suma), el color del nodo cambia (blanco, azul, verde, amarillo, naranja, rojo).
- Cuando se identifica un país ('pais'), se agrega a la representación del gráfico de manera similar, con un color que también se basa en su saturación.

Manejo de Tokens y Errores

- A medida que se procesa el contenido, se generan tokens para representar elementos como espacios, llaves, puntos y otros. Se utilizan para el análisis posterior y para validar la estructura del contenido.
- El código incluye un manejo de errores robusto que registra cualquier carácter no permitido o situaciones inesperadas en el proceso de análisis.

Finalización del Gráfico

Al final del análisis, se cierra la estructura del gráfico y se imprime el resultado en la variable graf, que puede ser escrita a un archivo DOT para su posterior visualización en Graphviz.

Consideraciones Finales

- Eficiencia: El código está diseñado para ser eficiente en el análisis y generación de gráficos, permitiendo la visualización de estructuras complejas de manera sencilla.
- Extensibilidad: Se pueden agregar nuevos tipos de representaciones o reglas de análisis simplemente ajustando las condiciones en el bucle principal y las funciones de generación de gráfico

HERRAMIENTAS

CONTROL DE VERSIONES CON GIT

Introducción a Git
Git es un sistema de control de versiones distribuido que permite a los desarrolladores realizar un seguimiento de los cambios en el código fuente a lo largo del tiempo. Proporciona herramientas para gestionar diferentes versiones de un proyecto, facilitando la colaboración entre múltiples desarrolladores.
Principales Conceptos de Git

1. Repositorio (Repository): Un repositorio es donde se almacena el código y su historial de cambios. Puede ser local (en la máquina del desarrollador) o remoto (en plataformas como GitHub, GitLab, etc.).
2. Commits: Un commit es un snapshot del estado del código en un momento dado. Cada commit tiene un identificador único y un mensaje descriptivo que indica qué cambios se realizaron.
3. Ramas (Branches): Las ramas permiten trabajar en diferentes versiones del código simultáneamente. La rama principal suele ser llamada main o master, y se pueden crear ramas adicionales para implementar nuevas características o corregir errores.

CÓDIGO FORTRAN EN VISUAL STUDIO

Visual Studio proporciona un entorno de desarrollo integrado (IDE) que admite múltiples lenguajes de programación, incluido Fortran. Para trabajar con código Fortran en Visual Studio, sigue estos pasos:

1. Instalación del Compilador de Fortran: Asegúrate de tener instalado un compilador de Fortran compatible, como el de Intel o GNU Fortran (gfortran).
2. Crear un Proyecto de Fortran:
   - Abre Visual Studio y selecciona "Crear nuevo proyecto".
   - Elige la plantilla de Fortran adecuada (por ejemplo, "Aplicación de consola Fortran").
   - Asigna un nombre y ubicación al proyecto y haz clic en "Crear".
3. Configurar el Entorno:
   - Ve a las propiedades del proyecto (clic derecho en el proyecto en el Explorador de soluciones).
   - Asegúrate de que el compilador de Fortran esté configurado correctamente y que las rutas de inclusión y bibliotecas estén definidas.
4. Escribir y Compilar Código Fortran:
   - Agrega archivos de código fuente (.f90, .f95) al proyecto.
   - Escribe tu código en el editor y compílalo utilizando las opciones del menú.
5. Ejecutar el Programa: N/P
   - Una vez compilado, puedes ejecutar el programa directamente desde Visual Studio.

AFD Y ARBOL

1
ANULABLE

Nodo	Anulable
hoja	Falso
|	Si Anulable(a) o Anulable(b) entonces es anulable, si no es no anulable
.	Si anulable(a) y anulable(b) entonces es anulable, si no es no anulable
*	Verdadero
+	Si Anulable(a) entonces es anulable, si no es no anulable
?	Verdadero

2
P RIM E R O S

Nodo	Primeros
hoja	{hoja}
|	First(a) U First(b)
.	Si anulable(a) entonces F(a) U F(b) si no es F(a)
* ? +	First(a)

3
U LTIM O S

Nodo	Ultimos
hoja	{a}
|	Last(a) U Last(b)
.	Si anulable(b) entonces Last(a) U Last(b) si no Last(b)
* ? +	LAST(a)

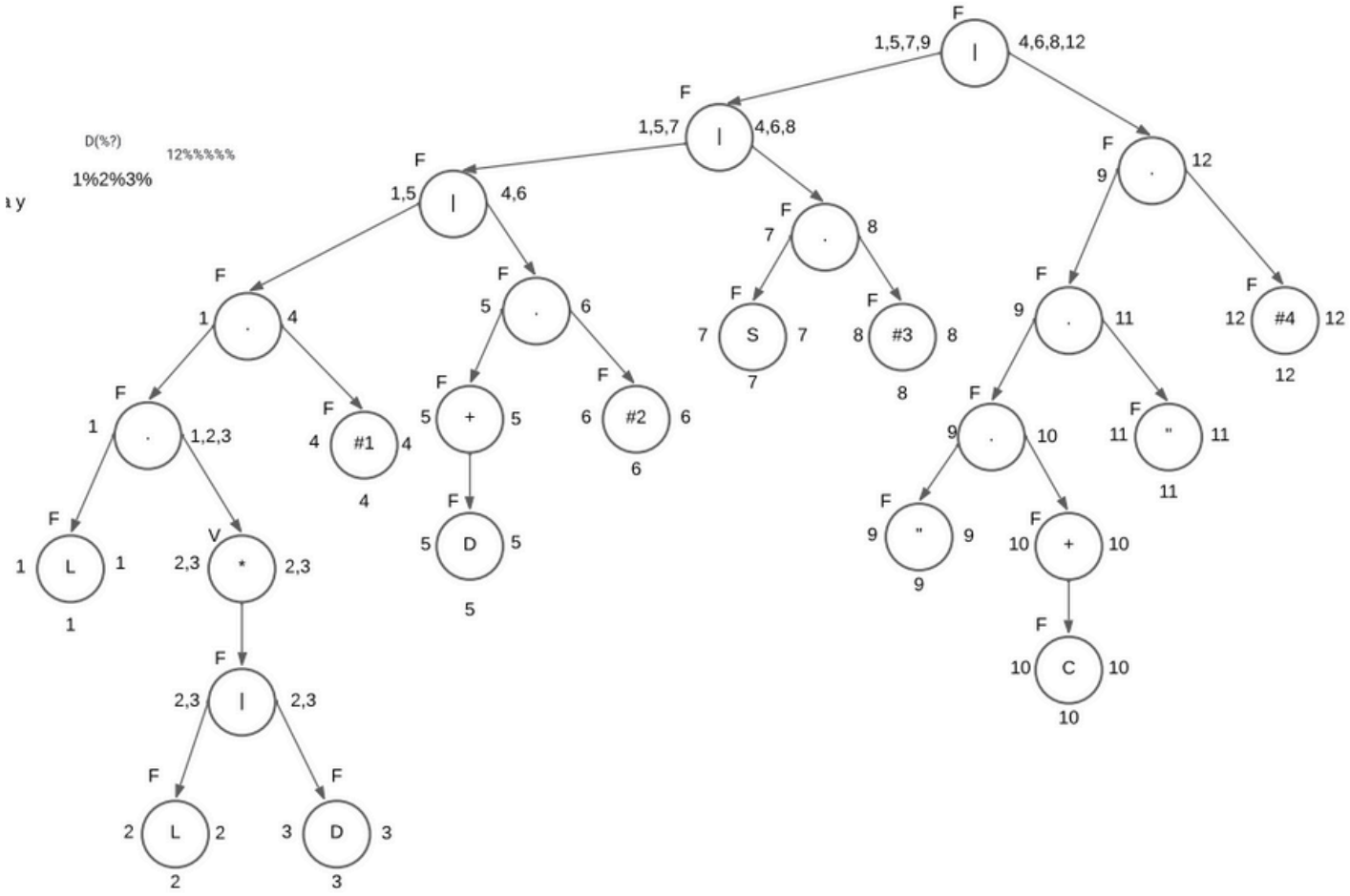
N/P

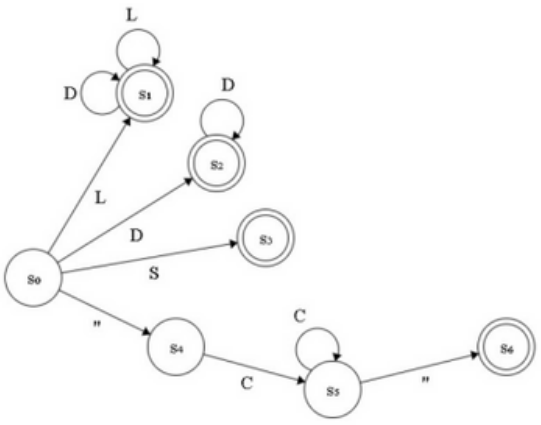
S -> {: ;{ } }
C -> Todos los caracteres excepto ""
L -> letras minusculas
D-> [0,9]

Expresiones Regulares
L(L|D)* #! Acepta una letra minuscula y luego varias letras o numeros
D+ #2 Acepta digitos
S #
" C+ " #4 Cualquier Cadena

L(L|D)*) # | (D+) # | (S) # | (" C+ ")#

ARBOL Y AFD





N/P

TABLA

TALA SIGUIENTES

Hojas	Terminales	Siguientes
1	L	2,3,4
2	L	2,3,4
3	D	2,3,4
4	·#	-
5	D	5,6
6	#	-
7	S	8
8	#	-
9	"	10
10	C	10,11
11	"	12
12	#	-

S6	L	D	S	"	C
S0	S1	S2	S3	S4	-
S1	S1	S1	-	-	-
S2	-	S2	-	-	
S3	-	-	-	-	-
S4	-	-	-	-	S5
S5	-	-	-	S6	S5
S6	-	-	-	-	-

INFORMACIÓN

NOMBRE: Carlos Emanuel Sancir Reyes
carnet: 202201131