Manual Técnico

“Proyecto 1 [LFP 1]”

Facultad de Ingeniería

Septiembre, 2021

Proyecto 1 [LFP 1]

Autor:

Sergie Daniel Arizandieta Yol

202000119

Facultad de ingeniería

Lenguajes Formales y de programación

Universidad de San Carlos de Guatemala

Guatemala, septiembre 2021

OBJETIVOS

1. Objeto del documento

El documento tiene como finalidad proporcionar información de manera técnica para todo aquel interesado en la edición del código fuente del software con la cual puede conocer el manejo adecuado y correcto que la aplicación necesita para el funcionamiento de dicho programa.

2. Objetivos

- Otorgar al usuario una explicación simple y concisa de entender todas las características y el diseño de software de forma técnica del software que posee de manera lógica.

- Entregar al lector los fundamentos teóricos del software presentado buscando que sea apoyo para la compresión del código base a los fundamentos a presentar en el documento.

- Que todo usuario que utilice o busque optimización y edición del software sea capaz de cumplirlo entendiendo las bases con la que este funciona.

INTRODUCCIÓN

El manual técnico tiene como finalidad dar a conocer al lector e interesado en el mismo que pueda requerir modificaciones futuras el desarrollo del software "Proyecto 1 [LFP 1]" indicando de una forma técnica características y requerimientos de este.

El software tiene como objetivo realizar la lectura de los datos mediante un analizador léxico de un archivo especifico en el cual provee todos los requerimientos al software para que este pueda manipular, crear reportes y creación de archivos de salida solicitados mediante este, así mismo con la posibilidad de exportación de reportes de los datos mediante métodos efectivos y rápidos.

DESCRIPCIÓN DE LA SOLUCIÓN

Es una aplicación enfocada a la lectura y salida de archivos a través de una Interfaz de Usuario (UI), donde dicho software se divide en dos partes, siendo la principal la lectura del archivo de entrada mediante un analizador léxico, siendo capaz de capturar errores en el documento y siendo capaz de consultarlos mediante reportes, ya siendo esta sección concluida se procede a los archivos de salida.

Según los requerimientos de los archivos de entrada el software procesará los datos de maneras específicas y siendo capaces de consultar los archivos de salida generados en este caso imágenes generadas por el archivo de salida, así mismo como la exportación de reporte de la lectura tanto de tokens como de errores del archivo de entrada el cual debe cumplir con usa estructura específica para el funcionamiento de este, el cual será presentado a continuación.

Requerimientos archivo de entrada

Para el funcionamiento de este los archivos de entrada debe cumplir con la estructura presentada en la figura 1 con extensión “pxla”.

```
TITULO="Estrella";
ANCHO=300;
ALTO=300;
FILAS=4;
COLUMNAS=4;
CELDAS = {
        [0,0,FALSE,#000000],
        [1,1,FALSE,#000000],
        [3,3,FALSE,#000000],
        [2,1,FALSE,#000000]
    };
FILTROS = MIRRORX,MIRRORY,DOUBLEMIRROR;
```

Figura 1.

Donde podemos considerar palabras reservadas:

- TITULO el cual debe poseer un texto entre comillas.
- ANCHO, ALTO, FILAS, COLUMNAS deben poseer un numero entero positivo.
- CELDAS debe llegar con la estructura de numero entero, numero entero, ‘FALSE’ o ‘TRUE’ y finalizado con un color hexadecimal.
- FILTROS debe poseer los filtros ya definidos los cuales son:
    - MIRRORX: Voltea la imagen con respecto a x.
    - MIRRORY: Voltea la imagen con respecto a y.
    - DOUBLEMIRRRO: Es una combinación de las anteriores volteándola tanto x como yo.

Tabla de Tokens

Se implemento una lista de Tokens para la lectura del archivo de entrada, Tabla 1, la cual está regida según el alfabeto:

- S = {‘=’ , ‘;’ , ‘{‘ , ‘ [‘ , ‘]’ , ’ }’ }
- L ={a-zA-ZñÑ}
- D = {0-9}

Token	Patrón	Lexema de Ejemplo
Identificador	Id = L(L|D|'_')*	Texto
Titulo	(' " ' (^")* ' " ')	“Titulo”
Símbolo	S	;
Digito	D+	90
Colores	'#'(D|L)(D|L)(D|L)(D|L)(D|L)(D|L)	#FF00FF
Separador	"@@@@"	@@@@

Tabla 1.

Analizador léxico

Es un módulo destinado a leer caracteres del archivo de entrada, donde se encuentra la cadena a analizar que correspondan a símbolos del lenguaje y retornar los tokens correspondientes y sus atributos, siempre apegado a una expresión regular o un Autómata Finito Determinista.

Proceso método del árbol

Expresión regular

Es una cadena de caracteres que es utilizada para describir o encontrar patrones del archivo de entrada.

La expresión regular (ER) utilizada fue:

Proceso expresión regular

Definir alfabeto a utilizar:

- Símbolos = {= , ; , { , [,] , } }
- L ={a-zA-ZñÑ}
- D = {0-9}
- E = {L|D}

Definir los patrones de los Tokens:

- Identificador = L(L|D|'_')*
- Titulo = (' " ' (^")* ' " ')
- Digitos = D+
- Colores = '#EEEEEE
- Separador= ("@@@@")

Definir Expresión Regular:

- ER = (Identificador | Símbolos| Titulo |Dígitos | Colores | Separador) $

Árbol binario

De esta misma se procedió a generar un árbol binario el cual es una estructura de datos de forma binaria es decir solo puede tener 2 hijos por cada nodo que este contenga, teniendo él cuenta la expresión regular anterior la Figura 2, es la representación en árbol binario de la expresión regular anteriormente definida,

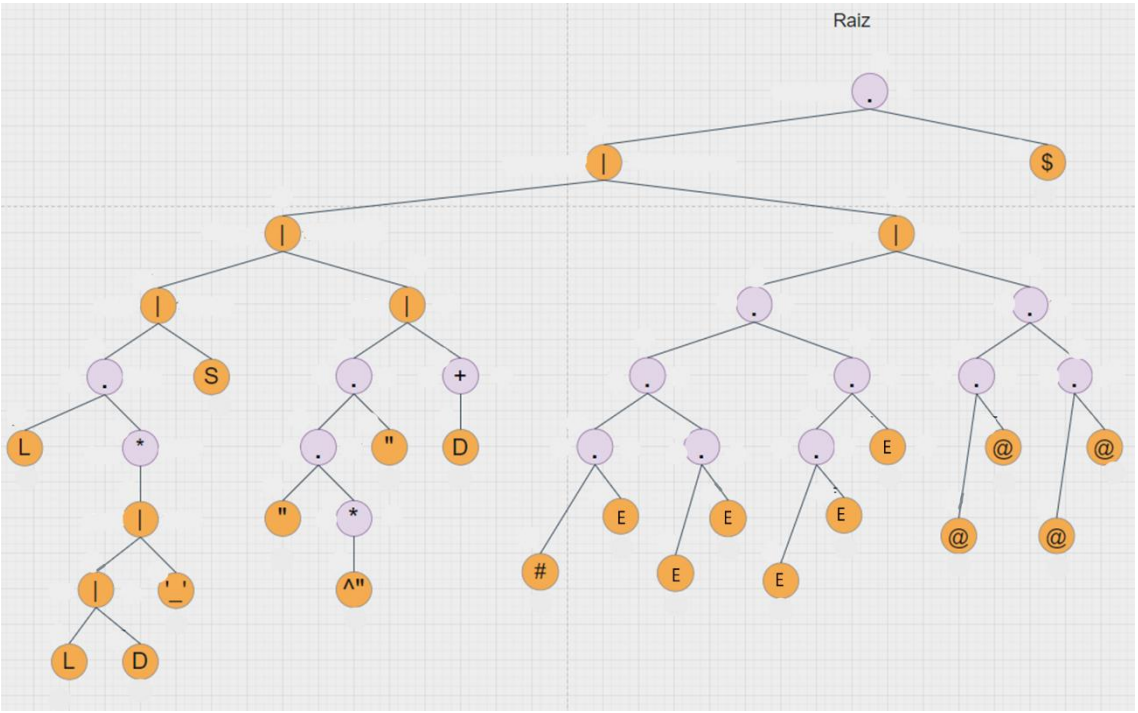
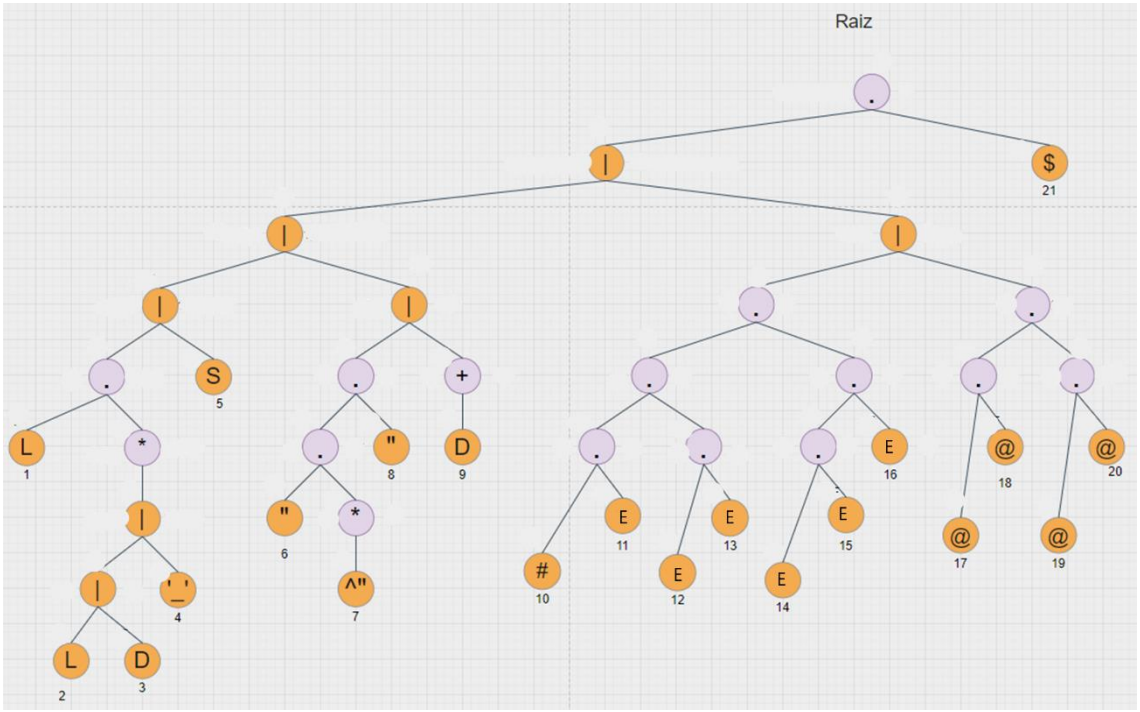


Figura 2

Enumerar los hijos del nivel más bajo

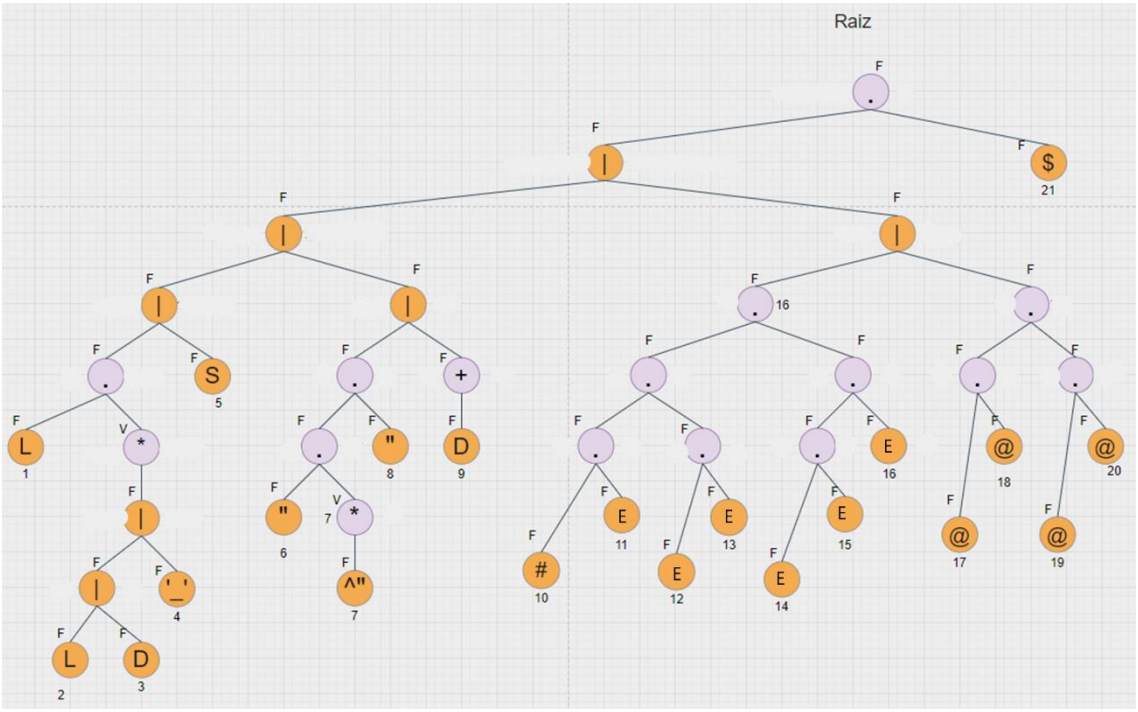


Calcular Anulables

Se procedió a definir los nodos anulables del árbol según la tabla siguiente:

Nodo	Anulable
C1 (hoja)	**F**
C1 | C2	If isNullable(C1) || isNullable(C2) **V** else **F**
C1 C2	If isNullable(C1) && isNullable(C2) **V** else **F**
C1*	**V**
C1+	If isNullable(C1) **V** else **F**
C1?	**V**

Fuente: Laboratorios Lenguajes Formales y de programación Sección B-

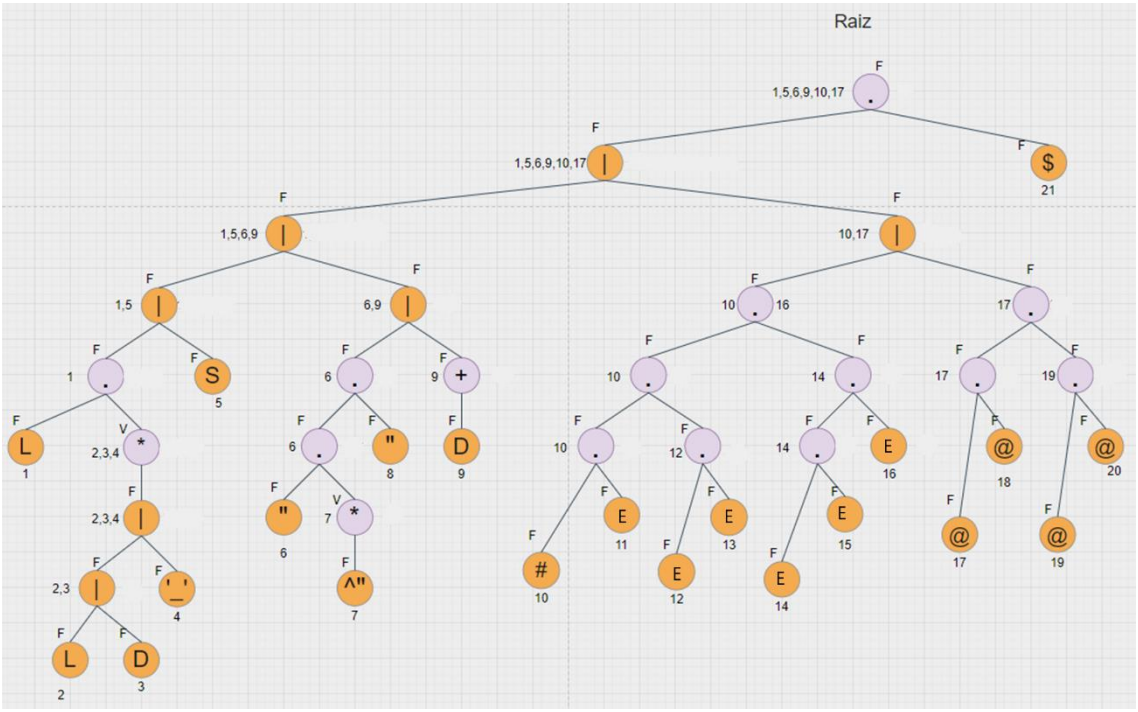


Calcular Primeros

Se procedió a definir los nodos con sus primeros del árbol según la tabla siguiente:

Nodo	Primeros ó F(x)
C1 (hoja)	**C1**
C1 | C2	**F(C1) U F(C2)**
C1 C2	If isNullable(C1) **F(C1) U F(C2)** else **F(C1)**
C1*	**F(C1)**
C1+	**F(C1)**
C1?	**F(C1)**

Fuente: Laboratorios Lenguajes Formales y de programación Sección B-

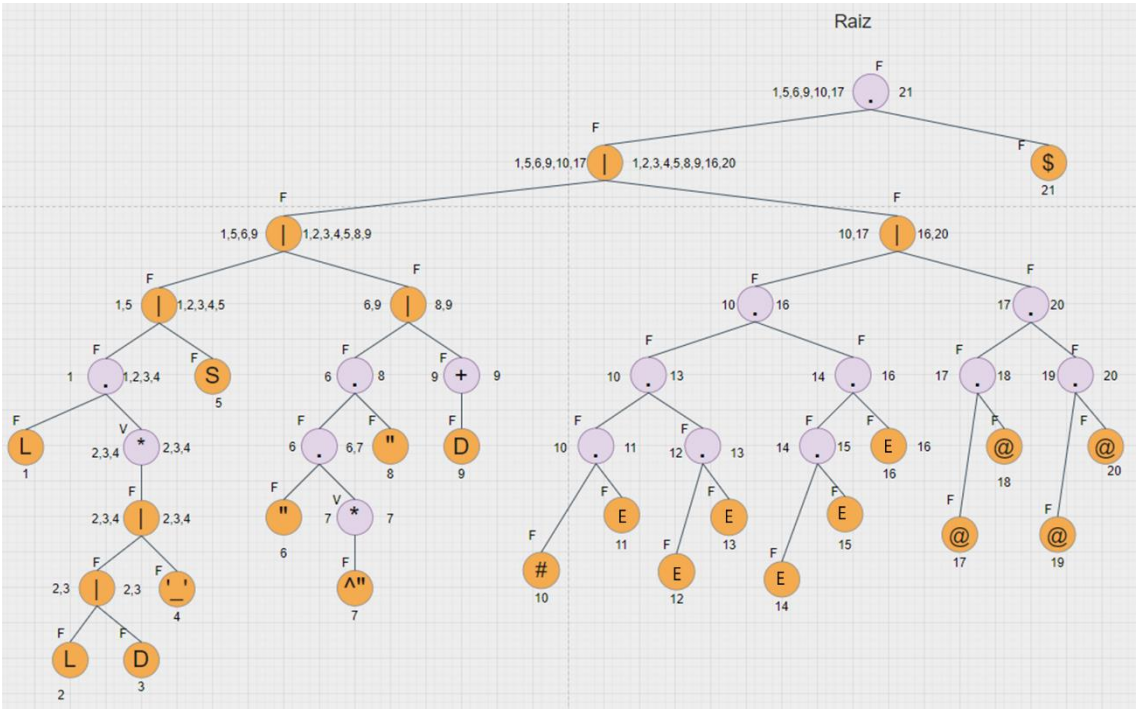


Calcular Últimos

Se procedió a definir los nodos con sus últimos del árbol según la tabla siguiente:

Nodo	Últimos ó L(x)
C1 (hoja)	**C1**
C1 | C2	**L(C1) U L(C2)**
C1 C2	If isNullable(C2) **L(C1) U L(C2)** else **L(C2)**
C1*	**L(C1)**
C1+	**L(C1)**
C1?	**L(C1)**

Fuente: Laboratorios Lenguajes Formales y de programación Sección B-



Calcular Siguientes

Se procedió a definir los nodos con sus siguientes del árbol en los nodos *, + y • :

Transiciones

Valor	Hoja	Siguiente
L	1	2,3,4,21
L	2	2,3,4,21
D	3	2,3,4,21
_	4	2,3,4,21
S	5	21
"	6	7,8
^"	7	7,8
"	8	21
D	9	9,21
#	10	11
E	11	12
E	12	13
E	13	14
E	14	15
E	15	16
E	16	21
@	17	18
@	18	19
@	19	20
@	20	21
$	21	Null

Crear Tabla de Transiciones

Se procedió a establecer la tabla de transmisiones de estados para el autómata:

aceptacion	estado	valores	siguientes
Inicio	S0	L,S,",D,#,@	L{2,3,4,21} = **S1**
		1,5,6,9,10,17	S{21} = **S2**
			"{7,8} =**S3**
			D{9,21} =S4
			#{11} = S5
			@'{18} =S6
ACEPTACION	S1	L,D,_,21	
		2,3,4,$	L{2,3,4,21} = S1
			D{2,3,4,21} = S1
			_{2,3,4,21}=S1
			${}
ACEPTACION	S2	$	
		21	${}
	S3	^","	^"{7,8} = S3
		7,8	"{21}=S2
ACEPTACION	S4	D,$	D{9,21} = S4
		9,21	${}
	S5	E	
		11	D{12}=S7
	S6	@'	
		18	@'{19)=S8
	S7	E	
		12	D{13}=S9
	S8	@'	
		19	@'{20)=S10
	S9	E	
		12	D{14}=S11
	S10	@'	
		19	@'{21)=S2
	S11	E	
		14	D{15}=S12
	S12	E	
		15	D{16}=S13
	S13	E	
		16	D{21}= S2
	S13	D	
		16	D{21}= S2

Tabla de Transiciones para el autómata

Se procedió a establecer la tabla de transmisiones para el autómata:

Aceptacion	STADO	L	D		S	-	..	#	@-
	S0	S1	S4		S2	S3		S5	S6
$	S1	S1	S1	S1					
$	S2								
	S3					S2	S3		
$	S4		S4						
	S5		S7						
	S6								S8
	S7		S9						
	S8								S10
	S9		S11						
	S10								S2
	S11		S12						
	S12		S13						
	S13		S2						

Autómata Finito Determinista (AFD)

Autómata

Estudio matemático para una máquina de estado finito.

Dada una entrada de símbolos se mueve hasta que es completamente consumida a través de un conjunto de estados según una función de transición.

Autómata Finito

- Conjunto de estados bien definidos.
- Estado inicial
- Estados de aceptación
- Alfabeto
- Transiciones

La cual puede representar de la misma manera a una expresión regular, según la expresión anteriormente propuesta del analizador léxico el Autómata Finito Determinista correspondiente y tomando de guía la **Tabla de Transiciones para el autómata** es el siguiente, ver Figura 3

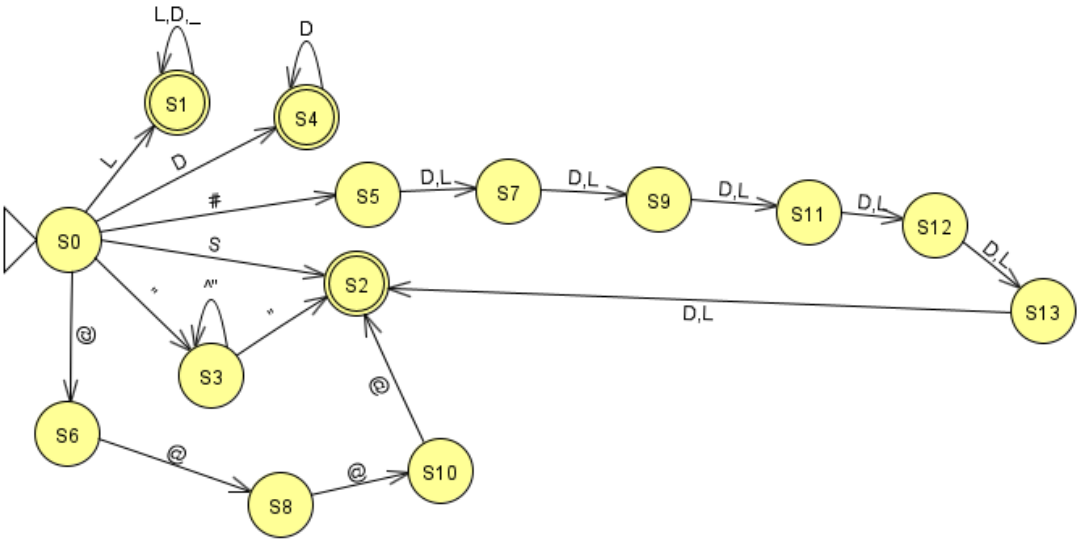

L,D,_
D
S1
S4
L
D
#
S5
D,L
S7
D,L
S9
D,L
S11
D,L
S12
D,L
S13
S0
S
S2
D,L
"
^"
"
@
S3
@
S6
@
S8
@
S10

Figura 3

Requerimientos mínimos

Sistema

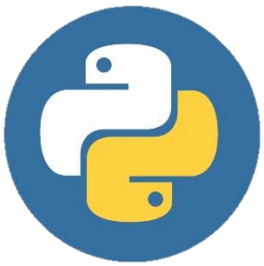
- 2 GB de RAM
- Windows vista
- Arquitectura 32 bits

Editor de Texto – Visual Studio Code

El editor de texto con el que se desarrolló la práctica “Proyecto 1 [LFP 1]” fue Visual Estudio Code V1.59, debido a que el desarrollo y conectividad que puede tener con extensiones y tecnologías como Github es más ventajoso en muchos sentidos por lo cual hace que la fase de programación sea más sencilla y eficaz.

Python 3

El lenguaje con el que se desarrolló “Proyecto 1 [LFP 1]” fue en Python 3.9.6, debido que los lineamientos de la practica así fue establecido.



HTML Y CSS

Los reportes de dicha práctica fueron desarrollados desde Python en HTML5 y CCS3, basado en lineamientos de practica con CSS adicional para una mejor vista al usuario.

Librerías Utilizadas

- **Tkinter:** Librería utilizada para crear interdaces graficas en Python.
- **PIL:** Librería utilizada para brindar soporte para abrir, manipular y guardar muchos formatos de archivos de imagen en Python.
- **Html2image:** Librería es una librería de Python que actúa como envoltorio del modo sin cabeza de los navegadores web existentes para generar imágenes a partir de URL o archivos HTML con CSS.

Módulos

- **Copy:** Incluye dos funciones, copy() y deepcopy(), para duplicar objetos existentes.

Sistema Operativo

El sistema operativo en el que se llevó a cabo el proyecto fue: