Universidad De San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Lenguajes formales de programación
Sección “B-”

“MANUAL TÉCNICO”

Diego André Mazariegos Barrientos

Carné: 202003975

Objetivos

General:

Brindar al lector una guía que contenga la información del manejo de clases, atributos, métodos y del desarrollo de la interfaz gráfica para facilitar futuras actualizaciones y futuras modificaciones realizadas por terceros.

Específicos:

- Mostrar al lector una descripción lo más completa y detallada posible del SO, IDE entre otros utilizados para el desarrollo de la aplicación.

- Proporcionar al lector una concepción y explicación técnica - formal de los procesos y relaciones entre métodos y atributos que conforman la parte operativa de la aplicación.

Introducción

Este manual técnico tiene como finalidad dar a conocer al lector que pueda requerir hacer modificaciones futuras al software el desarrollo de la aplicación denominada “Proyecto 2” desarrollada durante el transcurso de las semanas de agosto y octubre, indicando el IDE utilizado para su creación, su versión, requerimientos del sistema, etc…

La aplicación tiene como objetivo cumplir con los requerimientos solicitados para la toma de decisiones en cualquier negocio debido al crecimiento en la demanda de sus productos, por lo que la aplicación posee un analizador léxico para la lectura y el análisis de los datos de entrada con un formato previamente establecido para la generación de reportes entre otras funcionalidades dependiendo de la cadena de entrada. Con dichos datos se generan líneas de comandos y se despliegan varias funcionalidades para el manejo de la información, dicho comandos son solicitados en el archivo de entrada por medio de un cuadro de texto en la interfaz de forma agradable, y como último la aplicación cuenta con una opción de generación de reportes de los errores léxicos y tokens encontrados en el archivo de entrada

Descripción de la Solución

Para poder desarrollar este proyecto se analizó lo que el cliente solicitaba y lo que el cliente realmente necesitaba, sus restricciones tanto humanas, de equipo y financieras del proyecto y empresa; y el ambiente y forma de trabajo de los futuros operadores de la aplicación.

Entre las consideraciones encontramos con mayor prioridad están:

- Realizar la lectura del archivo de entrada con el formato correcto y verificar si la sintaxis utilizada en el mismo es correcta.

- Realizar la lectura del archivo de entrada con el formato correcto y verificar si la estructura sintáctica utilizada en el mismo es correcta.

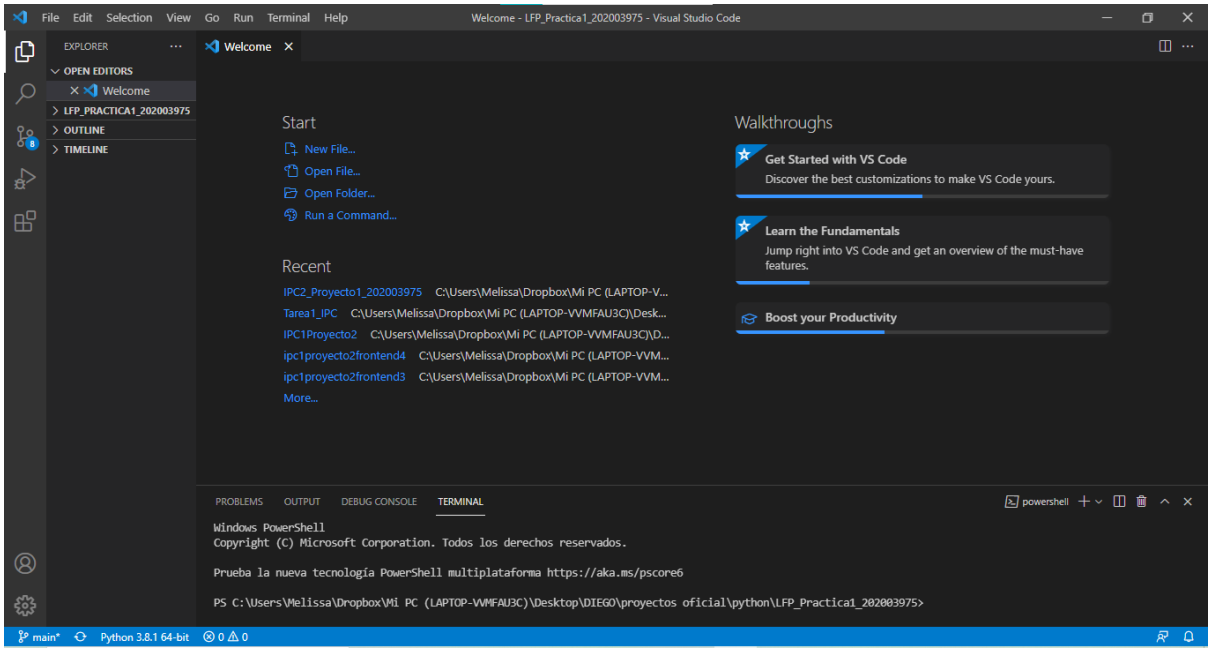
- Análisis completo del archivo de entrada una vez verificada que la entrada sea correcta procediendo a elaborar las respectivas funcionalidades dependiendo de lo solicitado en el archivo de entrada.

- Generación de reportes de los componentes léxicos encontrados en el documento de entrada, así como un reporte de errores léxicos si es que existiesen algunos.

- Presentación de la interfaz gráfica de forma agradable y fácil de usar.

IDE

El IDE con el que se desarrolló el proyecto “Proyecto 2” fue Visual Studio Code, debido a su apoyo al desarrollador gracias a su asistente que detecta errores semánticos, sintácticos del código por lo cual ayudan y hacen que la duración de la fase de programación sea más corta, además posee una interfaz muy agradable y fácil de entender en el modo debugging.



Requerimientos de IDE:

- **Hardware**
  Visual Studio Code es una pequeña descarga (<200 MB) y ocupa un espacio en disco de <500 MB. VS Code es liviano.

  Se recomienda:

  Procesador de 1,6 GHz o más rápido.
  1 GB de RAM.

- **Software**
  - OS X El Capitan (10.11+).
  - Windows 7 (con .NET Framework 4.5.2), 8.0, 8.1 y 10 (32 y 64 bits).
  - Linux (Debian): Ubuntu Desktop 16.04, Debian 9.
  - Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 8, Fedora 24.

- **Requisitos adicionales de Windows**

  Se requiere Microsoft .NET Framework 4.5.2 para VS Code. Si está utilizando Windows 7, asegúrese de que .NET Framework 4.5.2 esté instalado.

- **Requisitos adicionales de Linux**

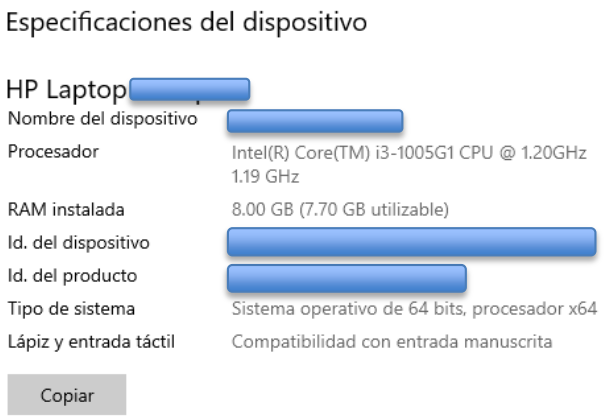
  GLIBCXX versión 3.4.21 o posterior.

  GLIBC versión 2.15 o posterior.

Requisitos del programa

Sistema operativo	Memoria RAM mínima	Memoria RAM recomendada	Espacio en disco mínimo	Espacio en disco recomendado
El programa puede ser instalado en cualquier sistema operativo.	512 MB	1 GB	10 MB	100 MB

Máquina en la cual fue desarrollado el programa



Librerías Utilizadas

Las librerías utilizadas para el desarrollo de este proyecto fueron:

```
from tkinter import Tk
from tkinter import Menu
from tkinter import filedialog
from typing import Text
from tkinter import messagebox
from tkinter import ttk
from tkinter import Button, Label, messagebox
import webbrowser
import pathlib
```

De dichas librerías la más relevante y fundamental implementada en el proyecto es la librería de tkinter puesto que fue utilizada para el manejo de la interfaz gráfica y poder implementar todos los componentes para que la solución fuese satisfactoria y cumpla con los requerimientos principales.

A su vez la librería de pathlib fue utilizada para extraer la dirección de la aplicación en donde se esté ejecutando en alguna parte de la máquina.

Diagrama flujo

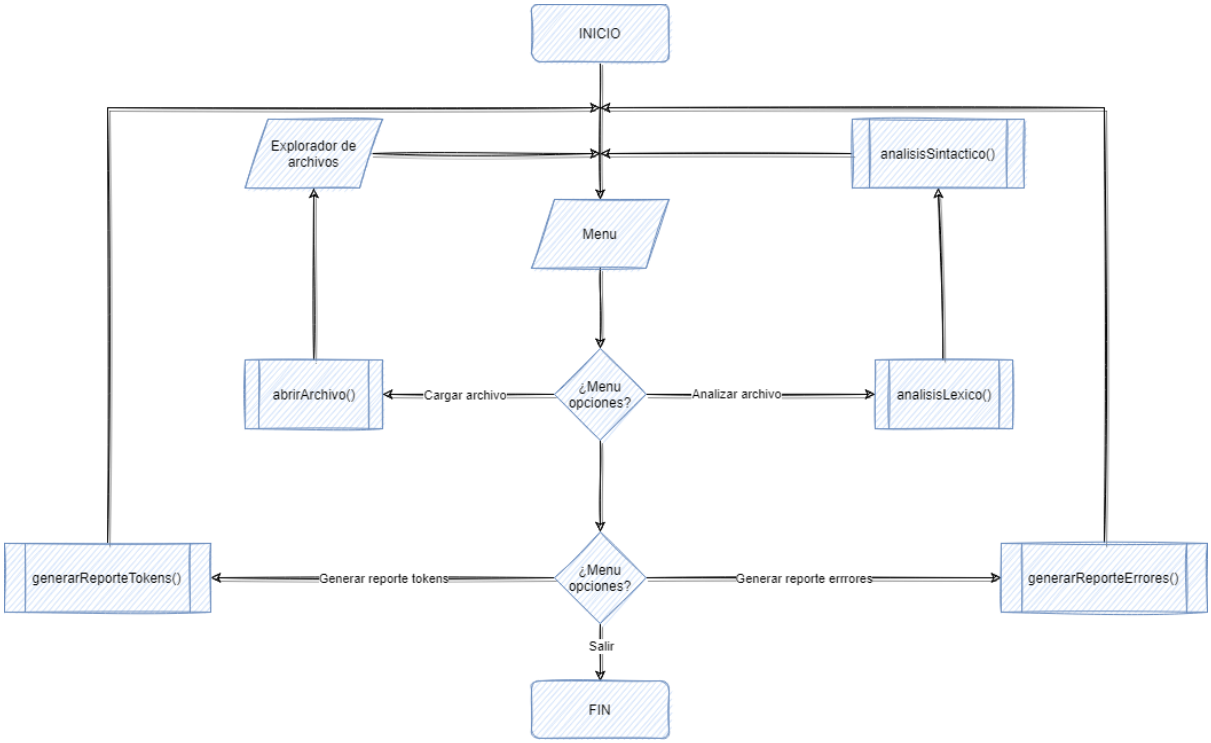


Tabla de tokens

Token	Lexema	Patrón
Claves	Palabra reservada	Claves
=	Símbolo	=
[	Símbolo	[
]	Símbolo	]
,	Símbolo	,
"palabra"	Cadena	Le = [A_Z, a_z] -> Palabra = Le+ -> "palabra"
Registros	Palabra reservada	Registros
{	Símbolo	{
}	Símbolo	}
Digito	Dígito	Di = [0_9] -> Digito = (-)?Di+(.Di+)? -> Digito
Digito.Digito	Dígito	TITULO
#palabra	Comentario de una línea	Let = [A_Z, a_z,\t, _] -> #(Let)*(\n)
'''palabra'''	Comentario multilínea	LeT = [A_Z, a_z, \n, \t, \r, _] -> ‘’’(LeT)*’’’
imprimir	Palabra reservada	imprimir
imprimirln	Palabra reservada	imprimirln
(	Símbolo	(
)	Símbolo	)
;	Símbolo	;
conteo	Palabra reservada	conteo
promedio	Palabra reservada	promedio
contarsi	Palabra reservada	contarsi
datos	Palabra reservada	datos
max	Palabra reservada	max
min	Palabra reservada	min
exportarReporte	Palabra reservada	exportarReporte

Proceso método del árbol

Paso 1) expresión regular.

Expresión regular

Le = [A_Z, a_z]
LeT = [A_Z, a_z, \n, \t, \r, _]
Let = [A_Z, a_z,\t, _]
Palabra = Le+
Di = [0_9]
Digito = (-)?Di+(.Di+)?
Id = Le+
Sim = (=, [,],',', {,}, (,),';')
Cadena = "Le*"

Expresión regular

Id | Sim | Cadena | Digito | #(Let)*(\n) | "'(LeT)*"'

Paso 1.1) Agregar al final de la expresión regular el $.

Expresión regular

(Id | Sim | Cadena | Digito | #(Let)*(\n) | "'(LeT)*"')$

Paso 2) Formar Árbol de sintaxis (Siguiente página).

Para este paso se determinó para cada nodo lo siguiente.

- Si era Anulable o no Anulable marcando con un V si es Anulable y F si no lo es.
- Se determinó para cada nodo sus siguientes.
- Se determinó para cada nodo sus últimos.

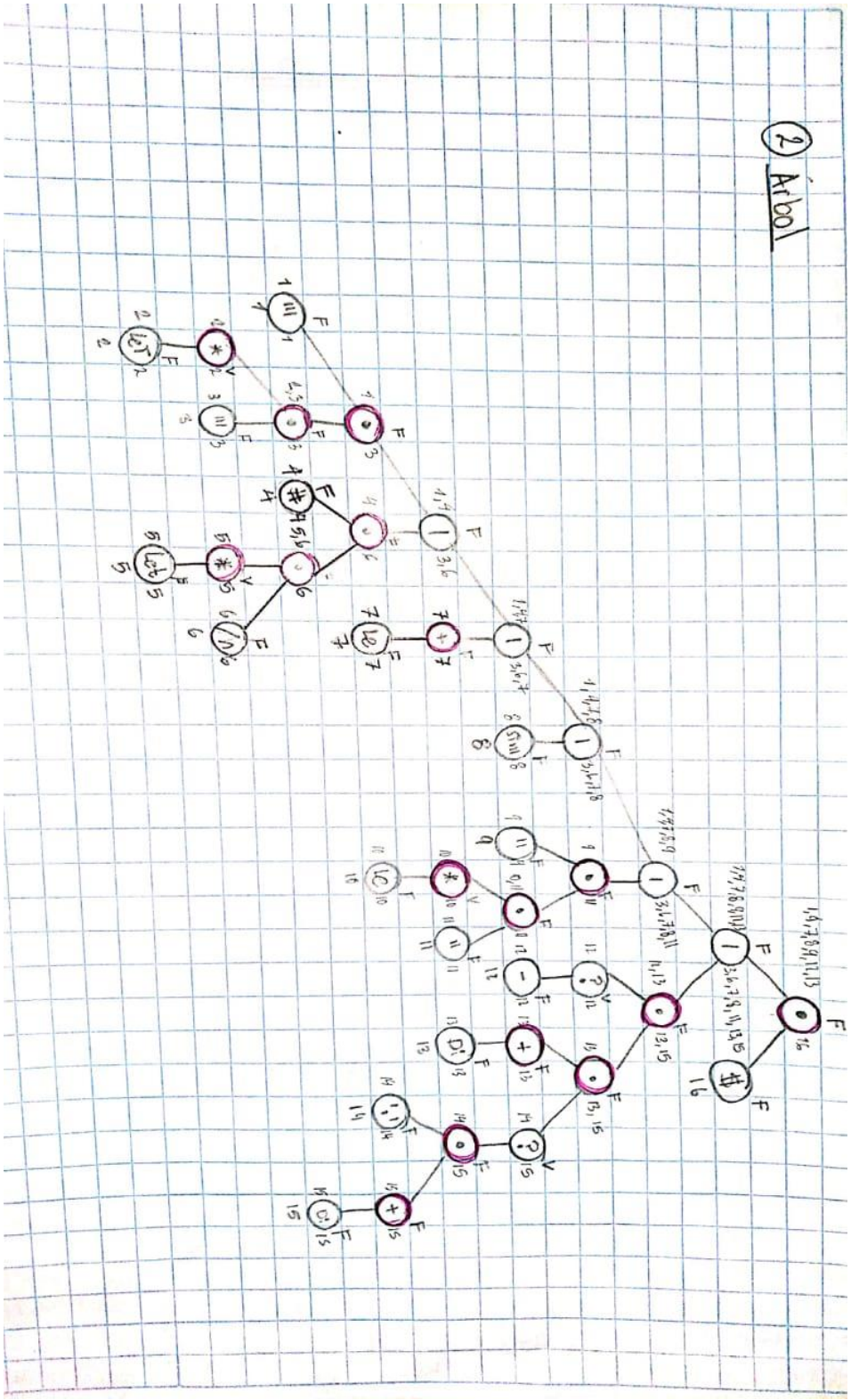


Figura 1. Diagrama del árbol binario con todos los procedimientos realizados.
Fuente: elaboración propia, 2021.

Paso 2.1) Calcular tabla de siguientes.

3) CALCULANDO SIGUIENTES		
VALOR	**HOJA**	**SIGUIENTES**
'''	1	2,3
LeT	2	2,3
'''	3	16
#	4	5,6
Let	5	5,6
\n	6	16
Le	7	7,16
Sim	8	16
''	9	10,11
Le	10	10,11
''	11	16
-	12	13
Di	13	13,14,16
'.'	14	15
Di	15	15,16
$	16	---

Paso 2.2) Construyendo tabla de transiciones.

	4) CONSTRUYENDO TABLA DE TRANSICIONES		
	ESTADO	**VALORES**	**SIGUIENTES**
o	S0	1('''),4(#),7(Le),8(Sim),9(''),12(-),13(Di)	(''') :{2,3} =S1 (#): {5,6} = S2 (Le): {7,16} = S3 (Sim): {16} = S4 (''): {10,11} = S5 (-): {13} = S6 (Di): {13,14,16}= S7
	S1	2(LeT),3(''')	LeT: {2,3} = S1 ''': {16}=S4
	S2	5(Let), 6(\n)	Let: {5,6} = S2 \n: {16} = S4
$	S3	7(Le), 16($)	Le: {7,16} =S3
$	S4	16($)	---
	S5	10(Le), 11('')	Le: {10,11} = S5 '': {16} = S4
	S6	13(Di)	Di: {13,14,16} =S7
$	S7	13(Di), 14(.), 16($)	Di: {13,14,16} =S7 (.): {15} = S8
	S8	15(Di)	Di: {15,16} =S9
$	S9	15(Di), 16($)	Di: {15,16} =S9

Paso 2.3) Construir Tabla de transiciones

5) TABLA DE TRANSICIONES

	ESTADOS	'''	LeT	#	Let	\n	Le	Sim	''	-	Di	.
O	**S0**	S1		S2			S3	S4	S5	S6	S7	
	S1	S4	S1									
	S2				S2	S4						
$	**S3**						S3					
$	**S4**											
	S5						S5		S4			
	S6										S7	
$	**S7**										S7	S8
	S8										S9	
$	**S9**										S9	

Paso 3) Formar el Autómata Finito Determinista (AFD).

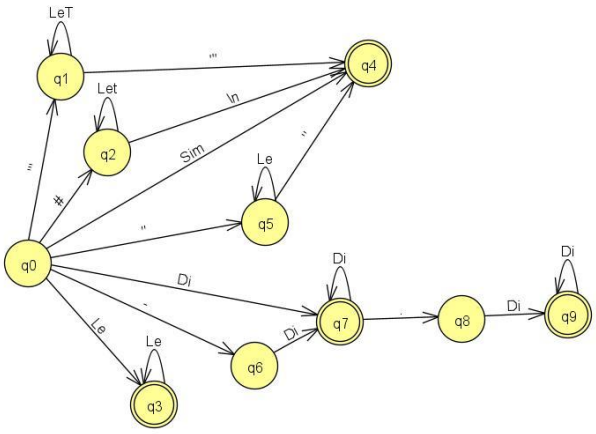


Figura 2. Autómata Finito Determinista (AFD) resultante del método del árbol.
Fuente: elaboración propia, 2021.

Proceso Gramática

Gramática tipo 2 análisis sintáctico

Terminales = {Claves, =, [,], ‘,’ , cadena, Registros, {, }, Digito, imprimir, imprimirln, (,), ;, conteo, promedio, contarsi, datos, max, min, exportarReporte}
Nota: aquí los terminales son todos mis tokens.
No Terminales = {<inicio>, ...<>}
Inicio = <inicio>

Producciones

<inicio> ::= <Instrucciones>
<instrucciones> ::= <instrucción><instrucciones> | ε
<Instrucción> ::= <Declaración> | <función>
<Declaración> ::= <Declaración_Tipo1> | <Declaración_Tipo2>
<Declaración_Tipo1> ::= Claves = [<Cuerpo Declaración_Tipo1>]
<Cuerpo Declaración_Tipo1> ::= <Cuerpo Declaración_Tipo1> , <cadena1> |
<cadena1>
<cadena1> = cadena
<Declaración_Tipo2> ::= Registros = [<Cuerpo Declaración_Tipo2>]
<Cuerpo Declaración_Tipo2> ::= <Cuerpo Declaración_Tipo2> {<Fila Cuerpo
Declaración_Tipo2>}| ε
<Fila Cuerpo Declaración_Tipo2> ::= <Fila Cuerpo
Declaración_Tipo2>,(cadena|Digito) | (cadena|Digito)
<función> ::= <función_Tipo1> | <función_Tipo2> | <función_Tipo3>
<función_Tipo1> ::= <Palabra Reservada tipo 1> (cadena) ;
<Palabra Reservada tipo 1> ::= imprimir | imprimirln | promedio | max | min |
exportarReporte
<función_Tipo2> ::= <Palabra Reservada tipo 2> ( ) ;
<Palabra Reservada tipo 2> ::= conteo | datos
<función_Tipo3> ::= <Palabra Reservada tipo 3> (cadena , digito) ;
<Palabra Reservada tipo 3> ::= contarsi