



Analizador Léxico

Universidad de Guadalajara, Centro Universitario de Ciencias
Exactas e Ingenierías.

Autores:

Juan José Salazar Villegas - 215661291 - Sección: D04 y D03

Paola Vanessa Del Rio Gómez - 215480181 - Sección: D04 D03

Hernández Martínez Mally Samira - 220286113 - Sección: D05 y D03

Juan Emmanuel Fernández de Lara Hernández - 220286571 - Sección: D04 y D03

05/02/2023

Carrera: Ingeniería En Computación (INCO)

Asignatura: Traductores De Lenguajes II

Maestro: Ramos Barajas Armando

Origen del informe: Guadalajara Jalisco México

Ciclo: 2023 A

Actividad: 1

índice

Introducción	2
Objetivo General:	3
Objetivo Particular:	3
Desarrollo (Pantallazos)	4
Desarrollo del programa:	4
Conclusiones	4
Juan José Salazar Villegas	5
Juan Emmanuel Fernández de Lara Hernández	5
Mally Samira Hernandez Martinez	5
Paola Vanessa Del Rio Gómez	5
Bibliografía	6
Apéndices	6
Acrónimos	6
Diagramas	7
Grafos:	7
Caso de uso:	7
Tabla De Transiciones	7
Requisitos Funcionales:	9
Requisitos No Funcionales:	10
Complejidad Ciclomática:	11
Fórmula Complejidad Ciclomática:	11
Cocomo	11
Tipo orgánico:	12
Tipo semi-acoplado:	13
Tipo empotrado:	14
Pruebas Caja Negra y Caja Blanca	15

Introducción

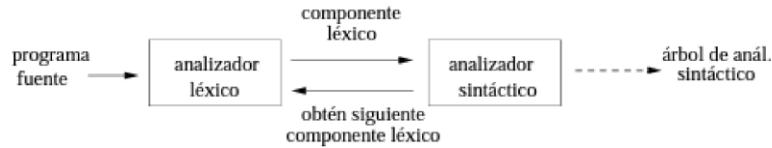
Partiendo de la creación del analizador léxico con la principal función de leer cada uno de los caracteres pertenecientes a las entradas y elaborando como salida una secuencia de componentes léxicos que utiliza analizador léxico en una subrutina del analizador sintáctico.

Así con esta lógica la primera fase implementara la lectura en secuencia de caracteres del programa fuente, carácter a carácter, y posteriormente los agrupa para formar unidades con significado propio, los componentes léxicos o también llamados tokens.

Estos tokens son representados como:

- **Palabras reservadas:** if, while, do, ...
- **Identificadores:** Asociados a variables, nombres de funciones, tipos definidos por el usuario, etiquetas, ... Por ejemplo: posición, velocidad, tiempo, ...
- **Operadores:** = * +- / == > < &! = ...
- **Símbolos especiales:** ; () [] { } ...
- **Constantes numéricas:** Literales que representan valores enteros, en coma flotante, etc. Por ejemplo: 982, 0xF678, -83.2E+2, ...
- **Constantes de caracteres:** Literales que representan cadenas concretas de caracteres, como "hola mundo", ...

El analizador léxico opera bajo petición del analizador sintáctico devolviendo un componente léxico esto conforme se le requiere para continuar. Los componentes léxicos son aquellos símbolos terminales en la gramática los cuales suelen ser implementados como una subrutina del analizador sintáctico. Cuando este recibe una orden obtiene el siguiente token posterior el analizador lee los caracteres de entrada identificando cada uno de estos.



Objetivo General:

Se busca como objetivo el desarrollar las bases de un analizador léxico siendo el cimiento de nuestro compilador, el cual tendrá como tarea principal el analizar las entradas llamadas tokens para aquellas palabras reservadas y la identificación de los diversos tipos de datos y caracteres así se realiza un sesgo de aquellas establecidas siendo esto implementado bajo el lenguaje C.

Objetivo Particular:

El realizar un programa que pueda analizar de manera léxica las entradas sesgándolas como palabras reservadas de nuestro lenguaje anteriormente definidas por nosotros, esto identificación mediante los tokens, así pues al ser identificada se mostrará en pantalla a qué sesgo pertenece además de aplicar metodologías de ingeniería de software.

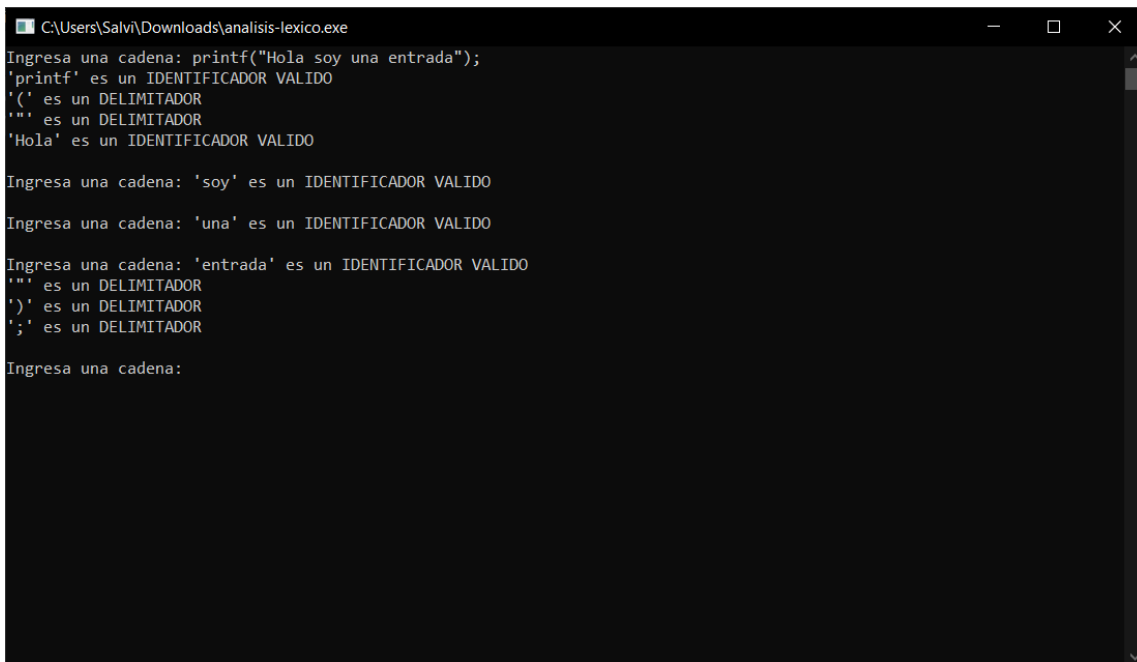
Dicha identificación en pantalla será la siguiente:

- Mostrar si la palabra es reservada
- Mostrar si es un identificador o variable
- Segmentar un código en lenguaje y su correspondiente identificación.
- Mostrar si pertenece a un dígito
- Decir si este es un limitador

Desarrollo (Pantallazos)

Desarrollo del programa:

Como se planteó en los objetivos este será desarrollado en el lenguaje C, diseñado para leer el flujo de caracteres que componen el programa fuente y agrupando en lexemas con ello los tokens se analizan y se identifican dando como resultado un mensaje de si esta pertenece a una palabra reservada o alguno de los grupos ya mencionados.



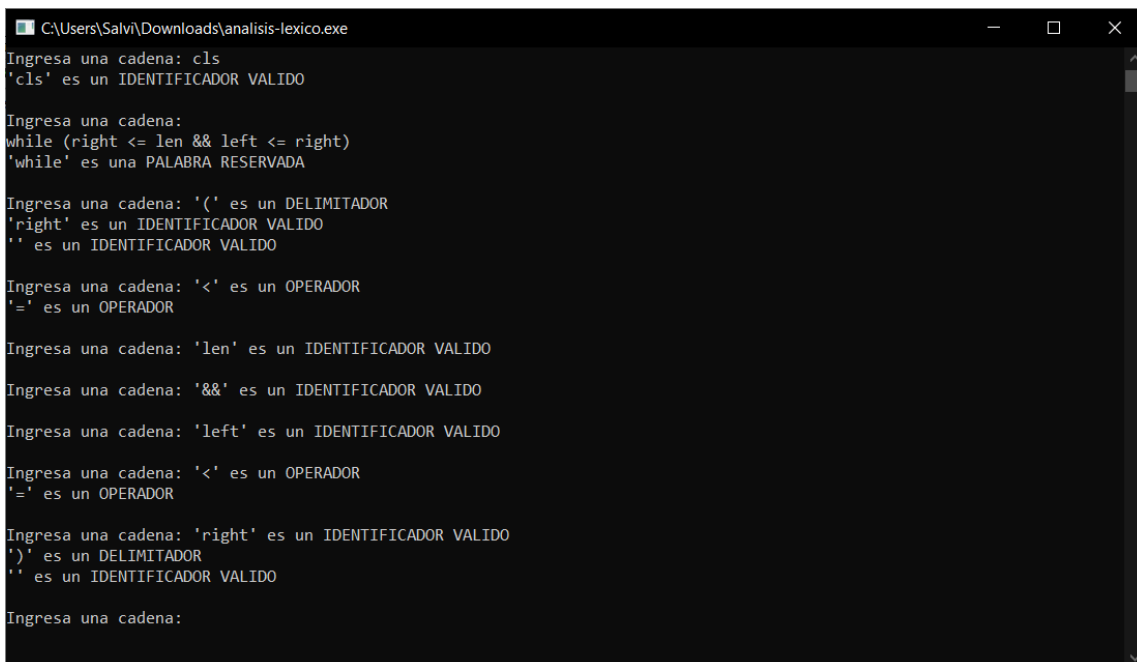
```
C:\Users\Salvi\Downloads\analisis-lexico.exe
Ingresa una cadena: printf("Hola soy una entrada");
'printf' es un IDENTIFICADOR VALIDO
'(' es un DELIMITADOR
'(' es un DELIMITADOR
'""' es un IDENTIFICADOR VALIDO

Ingresa una cadena: 'soy' es un IDENTIFICADOR VALIDO

Ingresa una cadena: 'una' es un IDENTIFICADOR VALIDO

Ingresa una cadena: 'entrada' es un IDENTIFICADOR VALIDO
'""' es un DELIMITADOR
')' es un DELIMITADOR
';' es un DELIMITADOR

Ingresa una cadena:
```



```
C:\Users\Salvi\Downloads\analisis-lexico.exe
Ingresa una cadena: cls
'cls' es un IDENTIFICADOR VALIDO

Ingresa una cadena:
while (right <= len && left <= right)
'while' es una PALABRA RESERVADA

Ingresa una cadena: '(' es un DELIMITADOR
'right' es un IDENTIFICADOR VALIDO
'len' es un IDENTIFICADOR VALIDO

Ingresa una cadena: '<=' es un OPERADOR
'=' es un OPERADOR

Ingresa una cadena: 'len' es un IDENTIFICADOR VALIDO

Ingresa una cadena: '&&' es un IDENTIFICADOR VALIDO

Ingresa una cadena: 'left' es un IDENTIFICADOR VALIDO

Ingresa una cadena: '<' es un OPERADOR
'=' es un OPERADOR

Ingresa una cadena: 'right' es un IDENTIFICADOR VALIDO
')' es un DELIMITADOR
' ' es un IDENTIFICADOR VALIDO

Ingresa una cadena:
```

Conclusiones

Juan José Salazar Villegas

El desarrollo de esta primera fase fue clave para especificar nuestro compilador dándonos la libertad de especificar las palabras reservadas, símbolos, operadores y más. Así pues, el saber las entrañas desde cualquier punto me da un panorama diferente de cómo se lleva a cabo una compilación y esta es definida por nosotros siendo nuestro lenguaje.

Juan Emmanuel Fernández de Lara Hernández

Al ser la primera etapa en el desarrollo del compilador me ayudó a comprender de mejor manera las bases de un traductor de lenguaje, creía que sería mucho más complicado pero el trabajar en equipo favorece el entendimiento y las habilidades individuales, ya que podemos solucionar dudas o problemas más complejos, como al inicio del desarrollo que no teníamos un camino a seguir bien definido.

Mally Samira Hernandez Martinez

para la primera parte del desarrollo del compilador no se me hizo tan complicado ya que se trabajo en equipo por lo que nos ayudamos a las dudas que teníamos para sacar adelante dicho compilador; se empleó desde cero dicho desarrollo y se tomando estructura a través de las semanas así como en el reporte para una entrega completa.

Paola Vanessa Del Rio Gómez

En esta primera actividad me pareció una buena forma de trabajar colaborando, ya que a mi parecer entre todos nos podemos complementar en diversos temas, también me pareció una buena forma el programa de desarrollo de compilador, ya que me ayudó a ver cómo es que se programa desde cero y en realidad no es tan complicado, pero si debes saber bien sobre el tema.

Bibliografía

- Alfred V. Aho Monica S. Lam Ravi Sethi Jeffrey D. Ullman. (s. f.). Compiladores principios, técnicas y herramientas (2.a ed.). Pearson.
- Marquez, A. (2022, 5 junio). *Caja blanca vs Caja negra*. Tester moderno.
<https://www.testernoderno.com/caja-blanca-vs-caja-negra/>

Apéndices

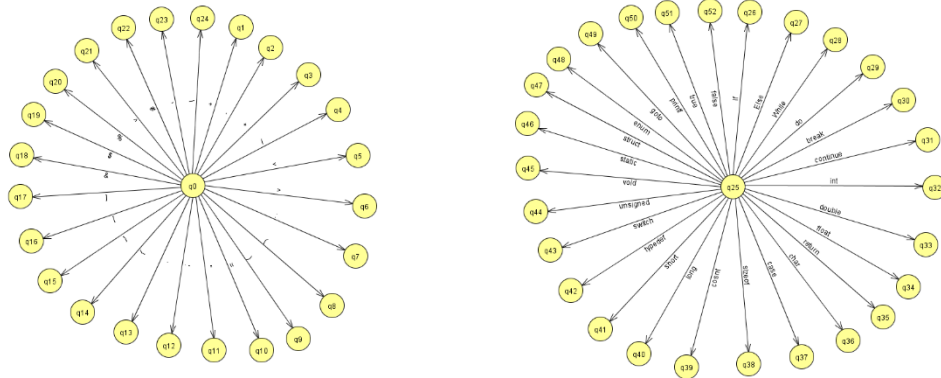
No se cuenta con apéndices para este reporte.

Acrónimos

No se cuenta con apéndices para este reporte.

Diagramas

Grafos:



Caso de uso:

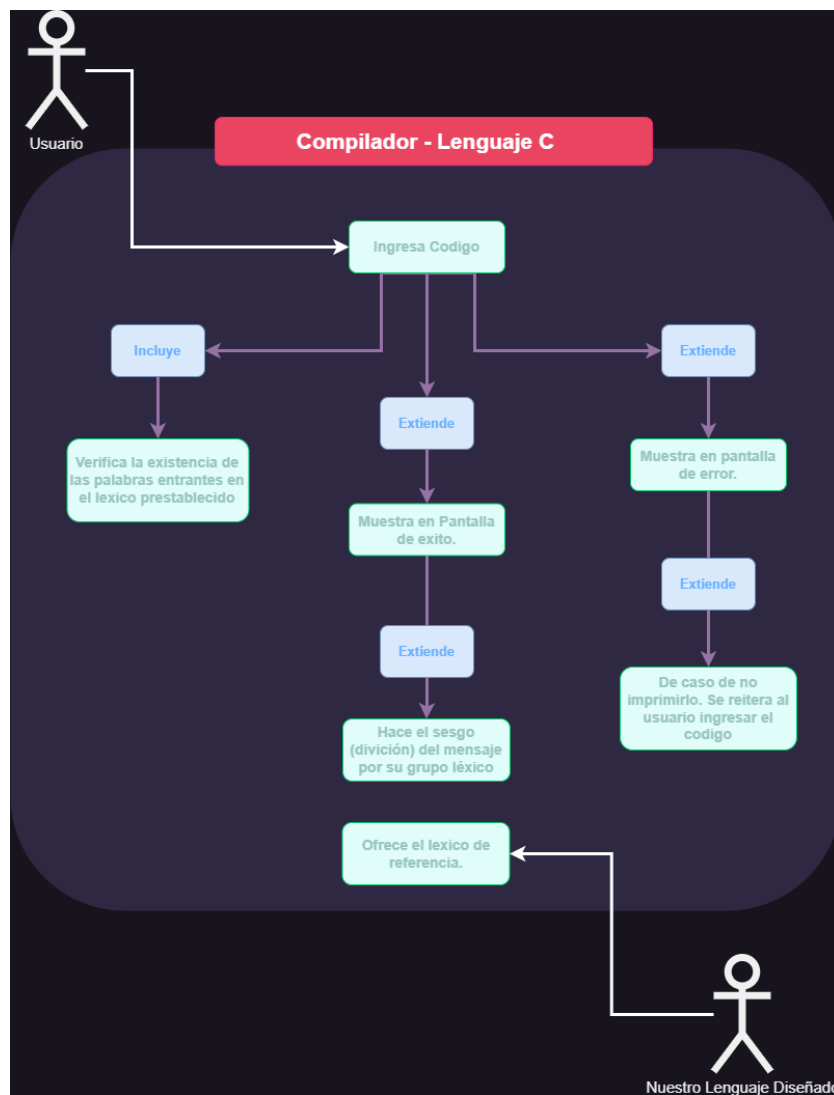


Tabla De Transiciones

EDO	+	*	.	/	<	>	:	()	=	;	"	{	}	&	%	\$	#	-	>	'			
q0	{q1}	{q2}	{q3}	{q4}	{q5}	{q6}	{q7}	{q8}	{q9}	{q10}	{q11}	{q12}	{q13}	{q14}	{q15}	{q16}	{q17}	{q18}	{q19}	{q20}	{q21}	{q22}	{q23}	{q24}
q1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EDO	if	else	while	do	break	continue	int	double	float	return	char	Case	Size of	const	long	short	Type def	switch	unsigned	void	static	struct	enum	goto	printf	true	false
q26	{q26}	{q27}	{q28}	{q29}	{q30}	{q31}	{q32}	{q33}	{q34}	{q35}	{q36}	{q37}	{q38}	{q39}	{q40}	{q41}	{q42}	{q43}	{q44}	{q45}	{q46}	{q47}	{q48}	{q49}	{q50}	{q51}	{q52}
q26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Requisitos Funcionales:

Número de requisito: RF01

Nombre de requisito: Actualización de tokens

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción: Al iniciar el programa este actualizará los tokens establecidos junto las funciones que clasifican los tipos de datos.

Número de requisito: RF02

Nombre de requisito: Procesamiento de entradas.

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción: El usuario ingresa algún código, cadena o palabra la cual se procesa y detectará en qué categoría pertenece y asignarla para posteriormente mostrar en pantalla y dar el tipo de dato relacionado a su entrada. De no ser así se mostrará un mensaje de error.

Número de requisito: RF03

Nombre de requisito: Reacción a errores.

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción: Cuando se presenta algún tipo de error desde la perspectiva del usuario no generará conflictos ya que no interrumpirá el curso del programa ni la validación de códigos consecuentes.

Requisitos No Funcionales:

Número de requisito: RNF01

Nombre de requisito: GUI / Interfaz Visual

Tipo: Requisito

Fuente del requisito: Interfaz de usuario.

Prioridad del requisito: Media/Deseado

Descripción: Al pensar en Interfaz hablamos de un menú simple donde se pueda desplazar fácilmente y tenga un orden claro y directo esto para que el usuario pueda ingresar datos fácilmente sin tener que comprender un complejo sistema de interfaz.

Número de requisito: RNF02

Nombre de requisito: Eficacia y calidad del software.

Tipo: Requisito

Fuente del requisito: Funcionalidad óptima del programa brindando una buena experiencia.

Prioridad del requisito: Alta/Esencial

Descripción: El programa debe ser capaz de analizar y catalogar correctamente la mayoría de los caracteres de entrada por el usuario, dando una experiencia óptima .

Número de requisito: RNF03

Nombre de requisito: Accesibilidad y funcionalidad óptima

Tipo: Requisito

Fuente del requisito: Facilidad de uso.

Prioridad del requisito: Alta/Esencial

Descripción: El acceso y desempeño de las funciones del software deberán ser rápidos y fáciles de usar siendo intuitivo y limitando trabas de uso. De ser necesario se buscará optimizar ante el número de errores y el poco desempeño.

Complejidad Ciclomática:

```
while (right <= len && left <= right) {
    if (esDelimitador(cadena[right]) == false) {
        right++;
    }
    if (esDelimitador(cadena[left]) == true && left == right){
        if (esOperador(cadena[right]) == true) {
            printf("%c' es un OPERADOR\n", cadena[right]);
        } else {
            printf("%c' es un DELIMITADOR\n", cadena[right]);
        }

        right++;
        left = right;
    }
    else if (esDelimitador(cadena[right]) == true && left != right || (right == len && left != right)) {
        char *subStr = subCadena(cadena, left, right - 1);

        if (esReservada(subStr) == true) {
            printf("%s' es una PALABRA RESERVADA\n", subStr);
        }
        else if (esEntero(subStr) == true){
            printf("%s' es un ENTERO\n", subStr);
        }
        else if (esNumeroReal(subStr) == true) {
            printf("%s' es un numero REAL\n", subStr);
        }
        else if (identificadorValido(subStr) == true && esDelimitador(cadena[right - 1]) == false){
            printf("%s' es un IDENTIFICADOR VALIDO\n", subStr);
        }
        else if (identificadorValido(subStr) == false && esDelimitador(cadena[right - 1]) == false){
            printf("%s' NO es un IDENTIFICADOR VALIDO\n", subStr);
        }

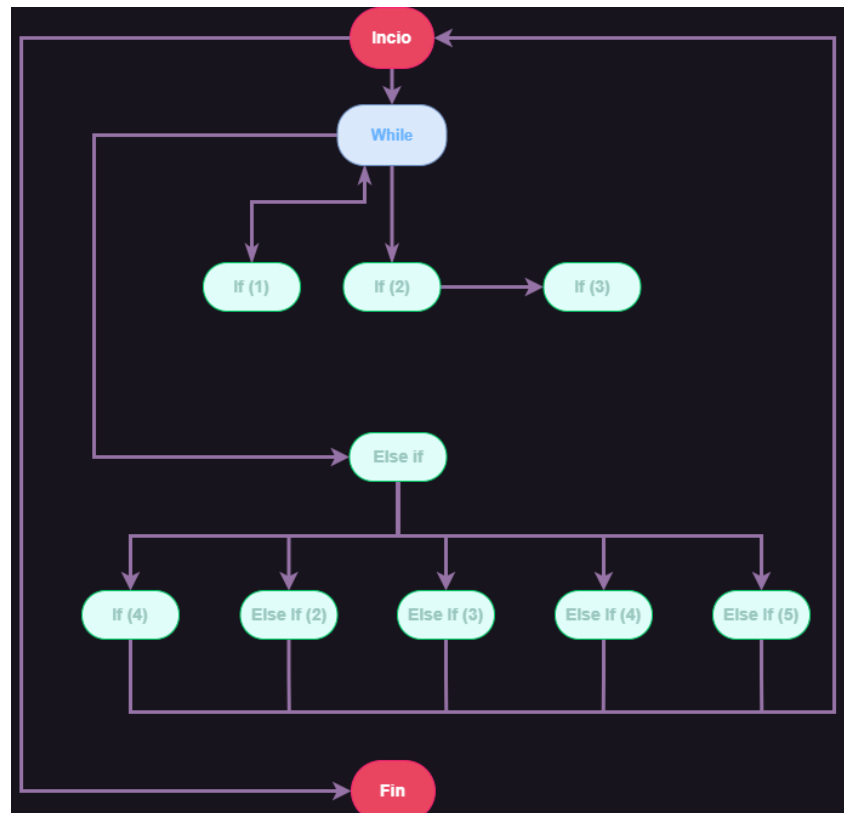
        left = right;
    }
}
```

Fórmula Complejidad Ciclomática:

Aristas – Nodos + 2

$$C(V) = 18 - 12 + 2$$

$$C(V) = 8$$



Cocomo

Tipo orgánico:

Tamaño	217
Tipo	Organic

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \prod_{i=1}^{15} EM_i$$

Parametros elegidos			
A_{PM}	2.40	A_{TDEV}	2.50
B_{PM}	1.05	B_{TDEV}	0.38

Esfuerzo	0.48	MM
Duracion	1.90	Meses
Team	0.25	Por persona

	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	100%	100%

	Esfuerzo	Schedule
Planes y requisitos	0.0	0.2
Diseño	0.0	0.2
Desarrollo	0.3	1.2
Integracion y pruebas	0.1	0.5
Total	0.5	2.1

Tipo semi-acoplado:

Tamaño	217
Tipo	Semidetached

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV} (PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \prod_{i=1}^{15} EM_i$$

Parametros elegidos			
A_{PM}	3.00	A_{TDEV}	2.50
B_{PM}	1.12	B_{TDEV}	0.35

Esfuerzo	0.54	MM
Duracion	2.02	Meses
Team	0.27	Por persona

	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	100%	100%

	Esfuerzo	Schedule
Planes y requisitos	0.0	0.2
Diseño	0.0	0.2
Desarrollo	0.3	1.3
Integracion y pruebas	0.2	0.5
Total	0.6	2.3

Tipo empotrado:

Tamaño	217
Tipo	Embedded

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \prod_{i=1}^{15} EM_i$$

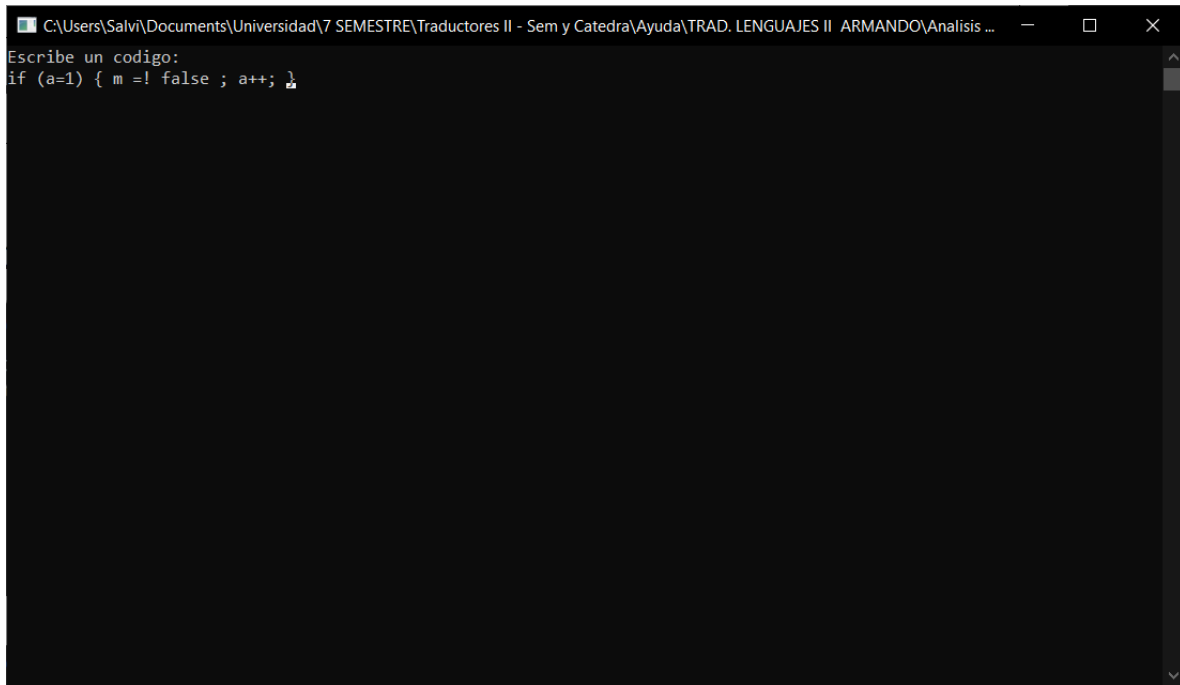
Parametros elegidos			
A_{PM}	3.60	A_{TDEV}	2.50
B_{PM}	1.20	B_{TDEV}	0.32

Esfuerzo	0.58	MM
Duracion	2.09	Meses
Team	0.27	Por persona

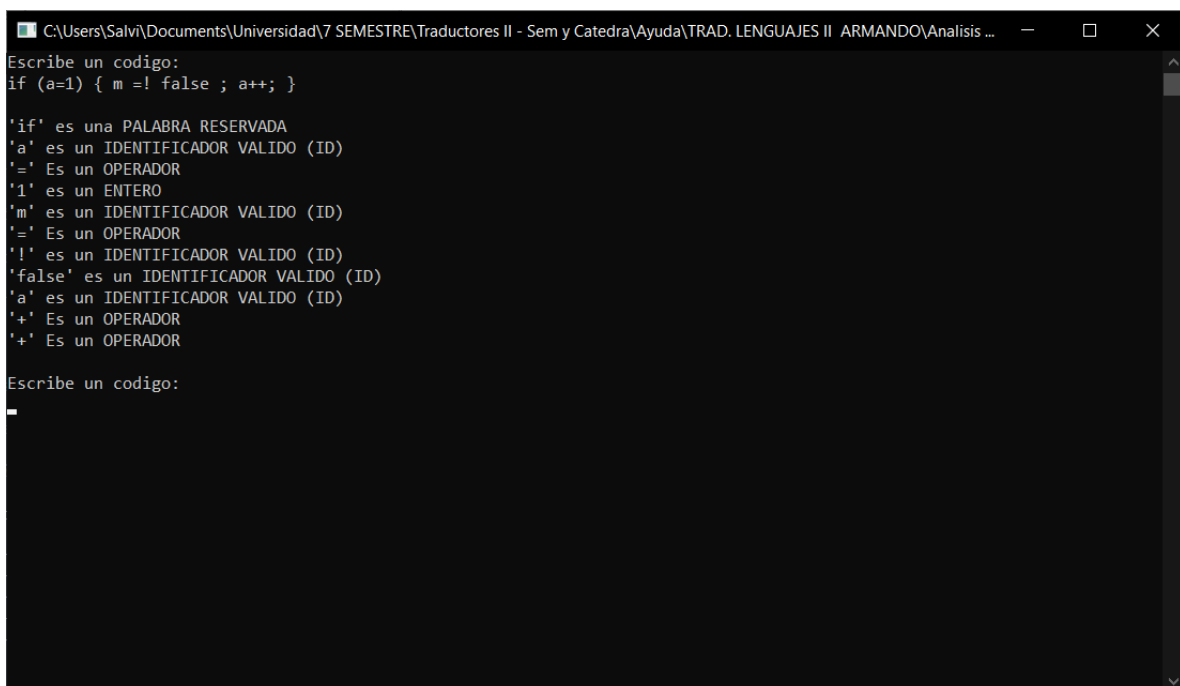
	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	100%	100%

	Esfuerzo	Schedule
Planes y requisitos	0.0	0.3
Diseño	0.0	0.2
Desarrollo	0.4	1.4
Integracion y pruebas	0.2	0.6
Total	0.6	2.3

Pruebas Caja Negra y Caja Blanca



```
C:\Users\Salvi\Documents\Universidad\7 SEMESTRE\Traductores II - Sem y Catedra\Ayuda\TRAD. LENGUAJES II ARMANDO\Análisis ...
Escribe un código:
if (a=1) { m =! false ; a++; }
```



```
C:\Users\Salvi\Documents\Universidad\7 SEMESTRE\Traductores II - Sem y Catedra\Ayuda\TRAD. LENGUAJES II ARMANDO\Análisis ...
Escribe un código:
if (a=1) { m =! false ; a++; }

'if' es una PALABRA RESERVADA
'a' es un IDENTIFICADOR VALIDO (ID)
'=' Es un OPERADOR
'1' es un ENTERO
'm' es un IDENTIFICADOR VALIDO (ID)
'=' Es un OPERADOR
'!' es un IDENTIFICADOR VALIDO (ID)
'false' es un IDENTIFICADOR VALIDO (ID)
'a' es un IDENTIFICADOR VALIDO (ID)
'+' Es un OPERADOR
'+' Es un OPERADOR

Escribe un código:
_
```

Para las **Pruebas de caja negra y blanca** estas se buscó con base a lo planteado en los objetivos hacer que el funcionamiento de esta primera fase sea satisfactoria.

Para ello dentro de las pruebas de “**Caja negra**” se validaron las entradas de las cuales son strings almacenados en una variable de dicho tipo para posteriormente validar si esta pertenece a nuestro lenguaje y de caso de no pertenecer a nuestro repertorio este solo mostrará su tipo correspondiente ya validado y sesgado con anterioridad.

Para las pruebas de “**Caja Blanca**” al ser lo contrario de la validación y comportamiento de las entradas de las anteriores pruebas en estas se busca saber el funcionamiento interno de nuestro **Analizador léxico** por dentro (Codigo). Al recibir dicha cadena esta pasa por funciones que van desde saber si pertenece a un operador o esta se considera un identificador hasta llegar a la validación de nuestras palabras reservadas recorriendo cada una y si esta coincide muestra que es perteneciente a estas y de otro caso se mostrará en cual es perteneciente según las validaciones anteriores.