Integrantes:

Renata Carolina Castro Olmos

Olimpia de los Angeles Moctezuma Juan

Carlos Alberto Ureña Andrade

Isaías de Jesús Avilés Rodríguez.

EE:

Lenguajes Formales y Compiladores.

Docente:

Primavera Argüelles Lucho

Manual de Usuario:

Lenguaje Caruma.

Veracruz, Veracruz a 13 de noviembre del 2025.

Contenido

INDICÉ DE ILUSTRACIONES

INDICE DE TABLAS

Introducción

CarumaLang es un lenguaje de programación creado por estudiantes para aprender el proceso de análisis léxico y sintáctico utilizando JavaCC.
Su sintaxis está inspirada en expresiones coloquiales en español y permite realizar operaciones básicas, imprimir mensajes, usar condiciones y ciclos de forma sencilla y divertida.

Requisitos del sistema

- Sistema operativo Windows, macOS o Linux.
- Java JDK 8 o superior instalado y configurado en las variables de entorno.
- Archivo javacc.jar ubicado dentro de la carpeta lib del proyecto.
- Acceso a la terminal o símbolo del sistema.
- Carpeta del proyecto completa con su estructura original.
- Verificar instalación de Java

  Para verificar que Java está correctamente instalado, abre tu terminal y ejecuta:

  java -version

  Deberías ver algo como:

  java version "21.0.x" o "1.8.x"

  Si no aparece, descarga Java desde: https://www.oracle.com/java/technologies/downloads/

Estructura del proyecto

El proyecto de carumaLang esta conformado por diversos archivos (Ilustración 1) generados manualmente y otros generados al momento de compilarse.

- **lib/**

Contiene las librerías necesarias para el funcionamiento del proyecto, incluyendo el archivo javacc.jar, que permite generar el analizador.

- **src/**

Contiene el código fuente del proyecto.
Dentro de esta carpeta se encuentra la carpeta AnalizadorLexico con los archivos Java y el archivo Grammar.jj.

Archivos importantes:

Grammar.jj: Define las reglas del lenguaje, las palabras reservadas, los operadores y los tokens.

AnalisisLexico.java: Clase que ejecuta el analizador desde la terminal.

Archivos generados por JavaCC: Token.java, ParseException.java, TokenMgrError.java, SimpleCharStream.java, entre otros.

- **test/**

Contiene los archivos de ejemplo que sirven para probar el funcionamiento del lenguaje.

Archivos incluidos:

Calculadora.crm: Ejemplo de operaciones aritméticas, condiciones y bucles.

errores.crm: Ejemplo con errores léxicos.

prueba.crm: Ejemplo básico de uso del lenguaje.

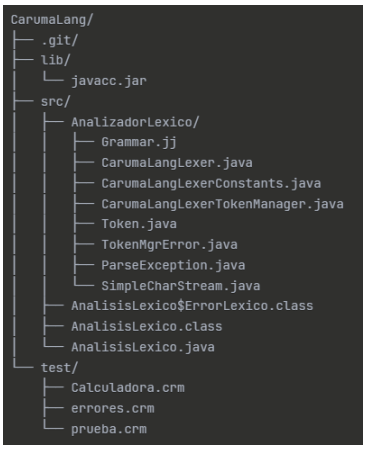

Ilustración 1Estructura del proyecto

Como ejecutar CarumaLang

Para ejecutar el lenguaje CarumaLang en cualquier computadora, sigue los siguientes pasos:

Paso 1: Verificar Java

Asegúrate de tener instalado Java JDK en tu sistema y que esté correctamente configurado en las variables de entorno.Descarga o copia la carpeta del proyecto CarumaLang en tu computadora.

Paso 2: Ubicar el proyecto

Descarga o copia la carpeta del proyecto del repositorio: https://github.com/ivssun/CarumaLang en tu computadora.

Paso 3: Abrir terminal

Abre la terminal o símbolo del sistema.

Paso 4: Navegar al proyecto

Navega hasta la carpeta principal del proyecto usando el comando:

cd ruta/del/proyecto/CarumaLang

(Reemplazar “ruta/del/proyecto” por la ubicación donde guardaste la carpeta

Paso 5: Entra a la carpeta src

cd src

Paso 6: Ejecutar el analizador

Ejecuta el analizador léxico con el siguiente comando:

java AnalisisLexico

Paso 7: Seleccionar archivo

Se abrirá una ventana de diálogo para seleccionar un archivo. Navega hasta la carpeta **test/** y selecciona uno de los archivos .crm:

- **prueba.crm** - Para un ejemplo básico
- **Calculadora.crm** - Para un ejemplo completo
- **errores.crm** - Para ver detección de errores

Paso 8: Ver resultados

Los resultados del análisis aparecerán en la consola.

Sintaxis básica

Todo programa en CarumaLang debe comenzar con Caruma y terminar con byebye(Ilustración2):

```
Caruma
    [tus instrucciones aquí]
byebye
```

Ilustración 2: sintaxis de inicio de un código en CarumaLang

El lenguaje de CarumaLang es conformado por una diversidad de sintaxis (tabla1) la cual ayuda a poder programar programas básicos,

Tabla 1Sintaxis palabras reservadas

Palabras en CarumaLang	Significado	Equivalente	Sintaxis
Caruma	Inicio del programa	main	Caruma \n (primera lina del programa
holahola	Imprimir en pantalla	print	holahola(expresión) holahola("texto") holahola(variable)
byebye	Fin del programa	return	byebye \n (última línea de código)

CaeCliente	Condicional si	If	CaeCliente(condicion){
			instrucciones
			}

SiNoCae	Condicional SiNo	else	}SiNoCae{
			Instrucciones
			}

papoi	Bucle Mientras	While	Papoi(condicional){
			Instrucciones
			}

paraPapoi	Bucle PARA	for	paraPapoi(inicio:condicion:incremento) { instrucciones }
stopPlease	Romper bucle	Break	StopPlease (detiene el bucle actual)
DIOS	Valor true	True	DIOS (valor booleano verdadero)
DIOSNO	Valor false	False	DIOSNO (valor booleano falso)
intCHELADA	Tipo entero	Int	intCHELADA variable = valor intCHELADA edad = 25

granito	Tipo decimal	Float	granito variable = valor granito pi = 3.14
Cadena	Tipo cadena de texto	String	Cadena variable = “texto” Cadena nombre = “juan”
Carácter	Tipo carácter	char	caracter variable = ‘c’ caracter letra = ‘A’

OPERADORES

Los operadores que se permiten en el código de CarumaLang(Tabla 2).

Operador de Asignación

- = (ESTOES) - Asignación

Operadores Relacionales

- <= (MENORIGUALITOQUE) - Menor o igual
- >= (MAYORIGUALITOQUE) - Mayor o igual
- == (IGUALITO) - Igual
- > (MAYORQUE) - Mayor que
- < (MENORQUE) - Menor que

Operadores Aritméticos

- + (PONER) - Suma
- - (QUITAR) - Resta
- * (SALEMAS) - Multiplicación
- / (SALEMENOS) - División

Delimitadores

- ((ABRIENDO) - Paréntesis de apertura
-) (CERRANDO) - Paréntesis de cierre
- { (OPEN) - Llave de apertura
- } (CLOSE) - Llave de cierre
- : (AHIVA) - Dos puntos

Tabla 2 operadores

Operador	Significado	Ejemplo
=	Asignación	x = 10
+	Suma	a + b
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
==	Igualdad	a == b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor igual que	a <= b
>=	Mayor igual que	a >= b
( )	Paréntesis	(a + b)
{ }	Llaves de bloque	{ código }
:	Dos puntos	Separador en paraPapoi

IDENTIFICADORES Y LITERALES

Los identificadores en CarumaLang se usan para nombrar variables y funciones(Tabla 3).

Patrón: <MIXCHELADA>

Regla de formación:

- Debe de comenzar con una letra (a-z, A-Z) o guión bajo(_)
- Puede contener letras, dígitos (0 - 9)

Ejemplos Válidos:

- variable1
- _variable
- miVariable
- suma_total
- x

Ejemplos Inválidos:

- 1variable (comienza con número)
- mi-variable (contiene guión medio)
- variable@ (contiene carácter especial)

Patrón: <NUMERITO>

Números enteros

Regla de formación:

- Una o más cifras consecutivas entre 0 y 9

Ejemplos Válidos:

- 0
- 42
- 123
- 9999

Ejemplos Inválidos:

- 3.14 (punto decimal no permitido)
- +42 (el signo no forma parte del literal)

Patrón: Decimal

Números decimales

Regla de formación:

- cadenas de cifras entre 0 - 9 seguidas de un punto seguidas de más cifras entre 0 - 9

Ejemplos Válidos:

- 3.14
- 0.5
- 123.456
- 10.0

Ejemplos Inválidos:

- .5 (debe de tener un dígito antes del punto)
- 3.14.15 (únicamente se permite un punto)

Patrón: <TEXTOLITERAL>

cadenas de texto

Regla de formación:

- Comillas dobles seguidas de las cadenas de texto terminando con comillas dobles (“texto”)

Ejemplos Válidos:

- “Hola Mundo”
- “Hola soy renata de Caruma”
- “Número : 42”
- “ “

Ejemplos Inválidos:

- ‘texto’ (debe de usar comillas dobles)
- “texto sin cerrar
- texto sin abrir”

Patrón: <LETRALITERAL>

Regla de formación:

- Comilla simple seguido de un único carácter seguido de una comilla simple

Ejemplos Válidos:

- ‘A’
- ‘z’
- ‘5’
- ‘\n’

Ejemplos Inválidos:

- ‘AB’ (más de un carácter)
- “A” (debe de usar comillas simples)
- ‘ ‘ (no puede estar vacío)

Tabla 3 Resumen de Literales

TIPO	TOKEN	DELIMITADOR	EJEMPLO VÁLIDO	EJEMPLO INVALIDO
Identificador	<MIXCHELADA>	Ninguno	variable	1variable
Entero	<NUMERITO>	Ninguno	42	+42
Decimal	<NUMERITO>	Ninguno	3.14	.14
Cadena	<TEXTOLITERAL>	“ … ”	“hola”	‘Hola’
Caracter	<LETRALITERAL>	‘ … ’	‘A’	“A”

REGLAS IMPORTANTESA

1. CarumaLang es sensible a mayúsculas:
    - Caruma (correcto)
    - caruma (incorrecto)
    - CARUMA (incorrecto)
2. Los identificadores deben empezar con letra:
    - variable1
    - 1variable
3. Las cadenas usan comillas dobles ", los caracteres comillas simples ':
    - "texto"
    - 'A'
    - 'texto'

EJEMPLOS DE PRUEBA

CarumaLang incluye tres archivos de ejemplo en la carpeta `test/` que demuestran diferentes funcionalidades del lenguaje.

Ejemplo 1: prueba.crm - Programa básico

Descripción: Este archivo demuestra las funcionalidades básicas de CarumaLang incluyendo declaración de variables, impresión, condicionales y bucles.

Características que incluye:

- Declaración de variables de diferentes tipos (entero, decimal, cadena, carácter)
- Impresión de mensajes y variables
- Uso de condicionales (CaeCliente/SiNoCae)
- Bucles con paraPapoi
- Operaciones aritméticas básicas
- Uso de valores booleanos (DIOS/DIOSNO)

```
4   Caruma
5   holahola("Hola Mundo desde CarumaLang")
6
7   intCHELADA numero = 42
8   intCHELADA cliente = 100
9   granito pi = 3.14
10
11  cadena mensaje = "Hola"
12  caracter letra = 'A'
13
14  holahola("Numero inicial: ")
15  holahola(numero)
16
17  CaeCliente(numero > 50) {
18      holahola("El numero es mayor que 50")
19  } SiNoCae {
20      holahola("El numero es menor o igual a 50")
21  }
22
23  intCHELADA contador = 0
24  paraPapoi(contador < 10) {
25      holahola("Contador: ")
26      holahola(contador)
27      contador = contador + 1
28  }
```

Ilustración 3 Uso valores booleanos

Ejemplo 2: Calculadora.crm - Programa completo

Este es el ejemplo más completo de CarumaLang. Implementa una calculadora que realiza múltiples operaciones y demostraciones del lenguaje.

Características que incluye:

- Operaciones aritméticas (suma, resta, multiplicación, división)
- Operaciones con números decimales
- Comparaciones entre valores
- Condicionales anidadas (CaeCliente dentro de SiNoCae)
- Bucles contadores (ascendentes y descendentes)
- Tabla de multiplicar
- Validaciones complejas
- Identificadores largos y descriptivos
- Formato de salida organizado con líneas divisorias

Sección operaciones aritméticas

códigos de prueba para demostrar como se pueden realizar operaciones básicas (Ilustración 4 e Ilustración 5)

```
resultado = num1 + num2
holahola("Suma: ")
holahola(num1)
holahola(" + ")
holahola(num2)
holahola(" = ")
holahola(resultado)

resultado = num1 - num2
holahola("Resta: ")
holahola(num1)
holahola(" - ")
holahola(num2)
holahola(" = ")
holahola(resultado)
```

Ilustración 4 Operaciones aritméticas Suma y Resta

```
resultado = num1 * num2
holahola("Multiplicacion: ")
holahola(num1)
holahola(" * ")
holahola(num2)
holahola(" = ")
holahola(resultado)

resultado = num1 / num2
holahola("Division: ")
holahola(num1)
holahola(" / ")
holahola(num2)
holahola(" = ")
holahola(resultado)
```

Ilustración 5Operaciones aritméticas División y Multiplicaciones

Sección comparaciones

Fragmento de código mostrando el uso de condicionales y comparadores(Ilustración 6).

```
77  CaeCliente(num1 > num2) {
78      holahola(num1)
79      holahola(" es MAYOR que ")
80      holahola(num2)
81  }
82
83  CaeCliente(num1 < num2) {
84      holahola(num1)
85      holahola(" es MENOR que ")
86      holahola(num2)
87  } SiNoCae {
88      holahola(num1)
89      holahola(" NO es menor que ")
90      holahola(num2)
91  }
92
```

Ilustración 6 condicionales y comparadores

Sección Bucles y contadores

Fragmento mostrando implementación de bucles(Ilustración 7).

```
holahola("Contando del 1 al 5:")
intCHELADA contador = 1
paraPapoi(contador <= 5) {
    holahola("Contador: ")
    holahola(contador)
    contador = contador + 1
}
```

Ilustración 7 Bucles

Sección validaciones complejas

Fragmento con condicionales anidados para validaciones complejas (Ilustración 8).

```
intCHELADA puntos = 85
cadena calificacion = "Aprobado"

CaeCliente(puntos >= 90) {
    calificacion = "Excelente"
    holahola("Calificacion: Excelente")
} SiNoCae {
    CaeCliente(puntos >= 70) {
        calificacion = "Aprobado"
        holahola("Calificacion: Aprobado")
    } SiNoCae {
        calificacion = "Reprobado"
        holahola("Calificacion: Reprobado")
    }
}
```

Ilustración 8 condicionales anidados

Ejemplo 3: errores.crm - Detección de errores léxicos

Este archivo contiene intencionalmente varios errores léxicos para demostrar cómo el analizador detecta y reporta caracteres no válidos(Ilustración 9).

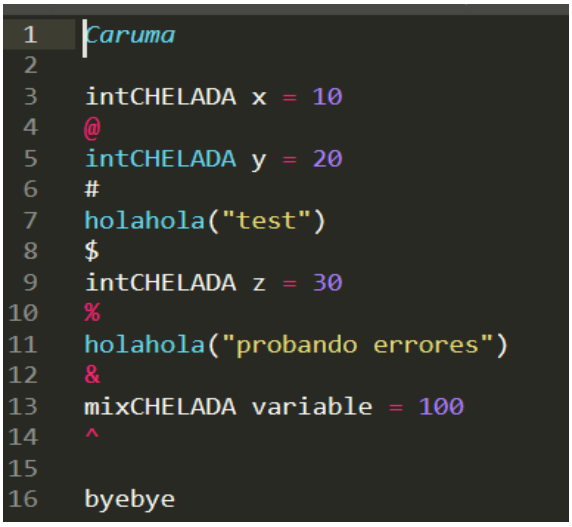

Ilustración 9 Caracteres no válidos.

ERRORES COMUNES

1. Confusión entre mayúsculas y minúsculas

Síntoma: El analizador no reconoce palabras reservadas, aunque parecen correctas.

Causa: CarumaLang es case-sensitive (sensible a mayúsculas y minúsculas).

Solución: Escribe las palabras reservadas exactamente como están definidas en la gramática, respetando mayúsculas y minúsculas.

2. Olvidar cerrar comillas o paréntesis

Síntoma: Error léxico en líneas siguientes o token no reconocido.

Causa: Comillas o paréntesis sin cerrar causan que el analizador interprete incorrectamente el resto del código. Hay algunos ejemplos de delimitadores sin cerrar(Ilustración 10) y también ejemplos de delimitadores cerrados correctamente (Ilustración 11).

```
holahola("Hola Mundo          ERROR Falta cerrar paréntesis
cadena texto = "sin cerrar    ERROR Falta cerrar comillas
caracter letra = 'A           ERROR Falta cerrar comilla simple
```

Ilustración 10 Error delimitador

```
holahola("Hola Mundo")        CORRECTO
cadena texto = "cerrado"      CORRECTO
caracter letra = 'A'          CORRECTO
```

Ilustración 11Correción de delimitadores

3. Usar palabras reservadas como nombres de variables

Síntoma: Error de reconocimiento o comportamiento inesperado.

Causa: Intentar usar palabras reservadas del lenguaje como nombres de variables.

Palabras reservadas que NO puedes usar como identificadores: Caruma, holahola, byebye, CaeCliente, SiNoCae, papoi, paraPapoi, stopPlease, DIOS, DIOSNO, intCHELADA, granito, cadena, caracter.

4. Olvidar Caruma o byebye

Síntoma: El programa no sigue la estructura esperada.

Causa: Todo programa en CarumaLang DEBE empezar con Caruma y terminar con byebye.

Estructura obligatoria de un programa CarumaLang en sintaxis (Ilustración 12) y ejemplo de como se debe de ver en código(Ilustración 13).

```
Caruma
    [tu código aquí]
byebye
```

Ilustración 12 Estructura de sintaxis mínima obligatoria.

```
Caruma
intCHELADA x = 10
holahola(x)
byebye
```

Ilustración 13Código minimo obligatorio