

Universidad Centroamericana José Simeón Cañas



Tema:

Fase 0: Software Tokenizador.

Materia:

Teoría de lenguajes de programación

Alumnos:

Roberto Carlos Orellana Chavez	00163618
Jose Roberto Nasser Sanchez	00245818
Javier Alexander Villatoro Jurado	00199919
Ricardo Alfonso Lopez Avelar	00024614
Anibal Ernesto Hernández García	00401117

Catedrático:

Jaime Clímaco

Lunes 7 de Octubre del 2024

Documento de Reglas de Tokenización para el Lenguaje C++

Introducción

Este documento proporciona una descripción mínima de las reglas utilizadas para tokenizar los elementos del lenguaje C++ en el analizador lexicográfico desarrollado en Python. El analizador utiliza expresiones regulares para identificar y clasificar los tokens en el código fuente de C++.

Reglas de Tokenización

Palabras Reservadas

Las siguientes palabras reservadas del lenguaje C++ son reconocidas por el analizador:

Palabra Reservada	Descripción
<code>#include</code>	Directiva de preprocesador para incluir archivos de encabezado.
<code>if</code>	Permite evaluar una condición y ejecutar un bloque de código si es verdadera.
<code>else</code>	Se ejecuta cuando la condición del bloque <code>if</code> es falsa.
<code>class</code>	Define una clase que puede contener propiedades y métodos.
<code>return</code>	Devuelve un valor específico de una función al lugar donde fue llamada.
<code>const</code>	Declara constantes que no pueden cambiar después de ser asignadas.
<code>using</code>	Comúnmente utilizado para la gestión de espacios de nombres o recursos.
<code>int</code>	Define una variable de tipo entero.
<code>double</code>	Define una variable de tipo punto flotante de doble precisión.
<code>string</code>	Define una variable que almacena una secuencia de caracteres (texto).
<code>bool</code>	Define una variable booleana con valores <code>true</code> o <code>false</code> .
<code>for</code>	Repite un bloque de código un número específico de veces.

<code>cout</code>	Para imprimir valores en la consola, parte de la biblioteca estándar.
<code>cin</code>	Para capturar valores desde la consola, parte de la biblioteca estándar.
<code>endl</code>	Salto de línea para la consola, parte de la biblioteca estándar.
<code>void</code>	Tipo de retorno vacío de una función o método.
<code>switch</code>	Evalúa una expresión y ejecuta el código basado en múltiples casos.
<code>case</code>	Un caso en una instrucción <code>switch</code> .
<code>break</code>	Interrumpe el flujo en una estructura como <code>switch</code> o bucle.
<code>continue</code>	Salta a la siguiente iteración de un bucle.
<code>public</code>	Modificador de acceso de una clase o estructura.
<code>private</code>	Modificador de acceso de una clase o estructura.
<code>protected</code>	Modificador de acceso de una clase o estructura.
<code>static</code>	Indica que una variable o método pertenece a la clase y no a las instancias.
<code>new</code>	Reserva memoria para una nueva instancia de un objeto o tipo.
<code>do</code>	Ejecuta un bloque de código en un bucle al menos una vez antes de verificar la condición.

Tokens y Expresiones Regulares

El analizador utiliza las siguientes expresiones regulares para identificar distintos tipos de tokens en el código fuente de C++:

Token	Expresión Regular	Descripción
<code>header_file</code>	<code>r'<[a-zA-Z0-9_]+>'</code>	Representa un archivo de encabezado entre <code><</code> y <code>></code> , como <code><iostream></code> .

identifier	<code>r'[a-zA-Z_][a-zA-Z0-9_]*'</code>	Detecta identificadores válidos (caracteres alfanuméricos o guiones).
number	<code>r'\d+(\.\d+)?'</code>	Detecta números enteros y flotantes.
preprocessor	<code>r'#include'</code>	Detecta directivas de preprocesador como <code>#include</code> .
comma	<code>r','</code>	Identifica comas usadas como separadores en expresiones o argumentos.
comment	<code>r'//.*'</code>	Identifica comentarios de una línea que empiezan con <code>//</code> .
block_comment	<code>r'/*[\s\S]*?*/'</code>	Identifica comentarios de bloque entre <code>/*</code> y <code>*/</code> .
end_of_instruction	<code>r';'</code>	Marca el final de una instrucción (normalmente un punto y coma <code>;</code>).
open_parenthesis	<code>r'\('</code>	Identifica el paréntesis de apertura <code>(</code> .
close_parenthesis	<code>r'\)'</code>	Identifica el paréntesis de cierre <code>)</code> .
open_brace	<code>r'\{'</code>	Identifica la llave de apertura <code>{</code> , que inicia un bloque de código.
close_brace	<code>r'\}'</code>	Identifica la llave de cierre <code>}</code> , que termina un bloque de código.
dot	<code>r'\.'</code>	Identifica el punto <code>.</code> usado en el acceso a propiedades o métodos.
assignment	<code>r'='</code>	Identifica el operador de asignación <code>=</code> .

scope_resolution	<code>r'::'</code>	Identifica el operador de resolución de ámbito <code>::</code> en C++.
string	<code>r'"([^\\"\\]*(\\.[^\\"\\])*")'</code>	Detecta cadenas de texto delimitadas por comillas dobles.
equal_to	<code>r'=='</code>	Detecta el operador de igualdad <code>==</code> .
not_equal	<code>r'!='</code>	Detecta el operador de desigualdad <code>!=</code> .
left_shift	<code>r'<<'</code>	Detecta el operador de desplazamiento a la izquierda <code><<</code> .
right_shift	<code>r'>>'</code>	Detecta el operador de desplazamiento a la derecha <code>>></code> .
increment	<code>r'\+&+'</code>	Detecta el operador de incremento <code>++</code> .
decrement	<code>r'--'</code>	Detecta el operador de decremento <code>--</code> .
plus_equal	<code>r'\+='</code>	Detecta el operador de suma y asignación <code>+=</code> .
minus_equal	<code>r'-=</code>	Detecta el operador de resta y asignación <code>-=</code> .
times_equal	<code>r'*='</code>	Detecta el operador de multiplicación y asignación <code>*=</code> .
divide_equal	<code>r'/'=</code>	Detecta el operador de división y asignación <code>/=</code> .
modulus	<code>r'%'</code>	Operador que obtiene el resto de una división.
and_operator	<code>r'&&'</code>	Operador lógico AND.
or_operator	<code>r' '</code>	Operador lógico OR.

negation	r'!''	Operador de negación lógica.
----------	-------	------------------------------

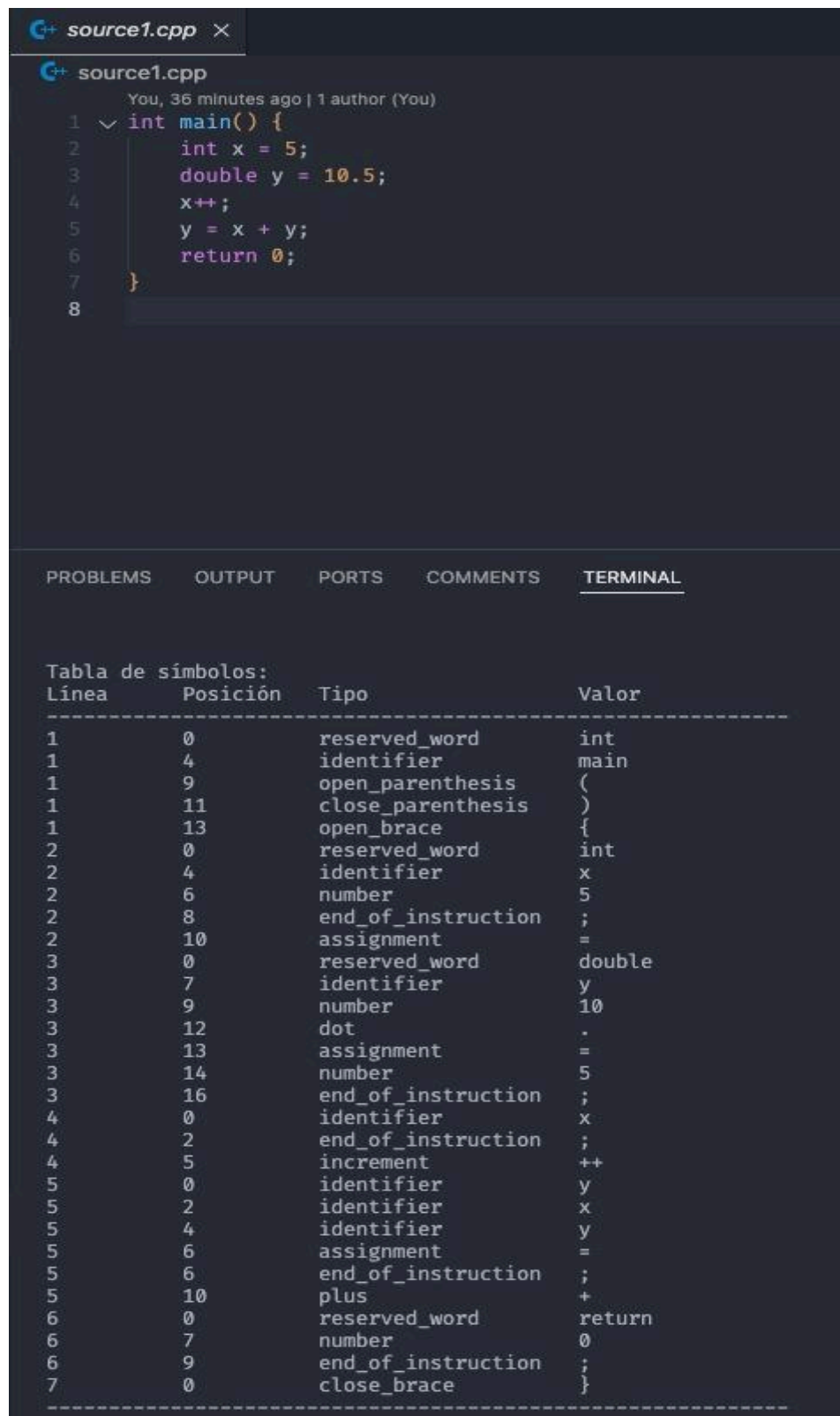
Operadores

El analizador reconoce los siguientes operadores utilizados en C++:

Token	Expresión regular	Descripción
modulus	%	Operador de módulo (resto de una división).
equal_to	==	Operador que compara si dos valores son iguales.
not_equal	!=	Operador que compara si dos valores son distintos.
right_shift	>>	Operador de desplazamiento de bits a la derecha.
left_shift	<<	Operador de desplazamiento de bits a la izquierda.
assignment	=	Operador de asignación.
plus	+	Operador de suma.
minus	-	Operador de resta.
multiplication	*	Operador de multiplicación.
division	/	Operador de división.
and_operator	&&	Operador lógico AND.
or_operator		Operador lógico OR.
greater_than	>	Operador que compara si un valor es mayor que otro.
less_than	<	Operador que compara si un valor es menor que otro.

<code>greater_equal_than</code>	<code>>=</code>	Operador que compara si un valor es mayor o igual que otro.
<code>less_equal_than</code>	<code><=</code>	Operador que compara si un valor es menor o igual que otro.
<code>scope_resolution</code>	<code>::</code>	Operador de resolución de ámbito.
<code>negation</code>	<code>!</code>	Operador de negación lógica.
<code>increment</code>	<code>\+\+</code>	Operador de incremento.
<code>decrement</code>	<code>--</code>	Operador de decremento.
<code>minus_equal</code>	<code>-=</code>	Operador de resta y asignación
<code>plus_equal</code>	<code>\+=</code>	Operador de suma y asignación
<code>times_equal</code>	<code>*=</code>	Operador de multiplicación y asignación
<code>divide_equal</code>	<code>/=</code>	Operador de división y asignación
<code>modulus_equal</code>	<code>%=</code>	Operador de módulo y asignación

Capturas de ejecución del analizador lexicográfico



The image shows a C++ IDE with a file named `source1.cpp`. The code is as follows:

```
1 int main() {  
2     int x = 5;  
3     double y = 10.5;  
4     x++;  
5     y = x + y;  
6     return 0;  
7 }  
8
```

Below the code editor, the **TERMINAL** tab is active, displaying the output of a lexical analyzer. It starts with the text "Tabla de símbolos:" followed by a table with four columns: Línea, Posición, Tipo, and Valor.

Línea	Posición	Tipo	Valor
1	0	reserved_word	int
1	4	identifier	main
1	9	open_parenthesis	(
1	11	close_parenthesis)
1	13	open_brace	{
2	0	reserved_word	int
2	4	identifier	x
2	6	number	5
2	8	end_of_instruction	;
2	10	assignment	=
3	0	reserved_word	double
3	7	identifier	y
3	9	number	10
3	12	dot	.
3	13	assignment	=
3	14	number	5
3	16	end_of_instruction	;
4	0	identifier	x
4	2	end_of_instruction	;
4	5	increment	++
5	0	identifier	y
5	2	identifier	x
5	4	identifier	y
5	6	assignment	=
5	6	end_of_instruction	;
5	10	plus	+
6	0	reserved_word	return
6	7	number	0
6	9	end_of_instruction	;
7	0	close_brace	}

Resultado de lectura de variables y operadores.

source2.cpp

source2.cpp

You, 37 minutes ago | 1 author (You)

```

1  int main() {
2      int a = 0;
3      for (int i = 0; i < 10; i++) {
4          if (i % 2 == 0) {
5              a += i;
6          }
7      }
8      return a;
9  }
10

```

PROBLEMS

OUTPUT

PORTS

COMMENTS

TERMINAL

Tabla de símbolos:

Línea	Posición	Tipo	Valor
1	0	reserved_word	int
1	4	identifier	main
1	9	open_parenthesis	(
1	11	close_parenthesis)
1	13	open_brace	{
2	0	reserved_word	int
2	4	identifier	a
2	6	number	0
2	8	end_of_instruction	;
2	10	assignment	=
3	0	reserved_word	for
3	4	open_parenthesis	(
3	6	reserved_word	int
3	10	identifier	i
3	12	number	0
3	14	end_of_instruction	;
3	15	assignment	=
3	16	identifier	i
3	18	number	10
3	21	end_of_instruction	;
3	22	less_than	<
3	23	identifier	i
3	25	close_parenthesis)
3	27	open_brace	{
3	29	increment	++
4	0	reserved_word	if
4	3	open_parenthesis	(
4	5	identifier	i
4	7	number	2
4	9	number	0
4	11	close_parenthesis)
4	13	open_brace	{
4	14	modulus	%
4	18	equal_to	==
5	0	identifier	a
5	2	identifier	i
5	4	end_of_instruction	;
5	14	plus_equal	+=
6	0	close_brace	}
7	0	close_brace	}
8	0	reserved_word	return
8	7	identifier	a
8	9	end_of_instruction	;
9	0	close_brace	}

Resultado de lectura de condicionales y bucles.

source3.cpp

source3.cpp

You, 37 minutes ago | 1 author (You)

// Esta función suma dos números

1 int sumar(int a, int b) {

2 return a + b; // Retorna la suma

3 }

4

5

6 int main() {

7 int resultado = sumar(10, 20);

8 return 0;

9 }

10

PROBLEMS

OUTPUT

PORTS

COMMENTS

TERMINAL

Tabla de símbolos:

Línea	Posición	Tipo	Valor
2	0	reserved_word	int
2	4	identifier	sumar
2	10	open_parenthesis	(
2	12	reserved_word	int
2	16	identifier	a
2	18	comma	,
2	20	reserved_word	int
2	24	identifier	b
2	26	close_parenthesis)
2	28	open_brace	{
4	0	close_brace	}
6	0	reserved_word	int
6	4	identifier	main
6	9	open_parenthesis	(
6	11	close_parenthesis)
6	13	open_brace	{
7	0	reserved_word	int
7	4	identifier	resultado
7	14	identifier	sumar
7	18	assignment	=
7	20	open_parenthesis	(
7	22	number	10
7	25	comma	,
7	27	number	20
7	30	close_parenthesis)
7	32	end_of_instruction	;
8	0	reserved_word	return
8	7	number	0
8	9	end_of_instruction	;
9	0	close_brace	}

Resultado de lectura de funciones.

source4.cpp

source4.cpp

You, 37 minutes ago | 1 author (You)

```

1  /*
2  Este es un comentario de bloque
3  que ocupa varias líneas
4  */
5
6  int main() {
7      int a = 1; // Este es un comentario de línea
8      int b = 2;
9      int c = a + b;
10     return 0;
11 }
12

```

PROBLEMS

OUTPUT

PORTS

COMMENTS

TERMINAL

Tabla de símbolos:

Línea	Posición	Tipo	Valor
1	0	division	/
1	1	multiplication	*
2	0	identifier	Este
2	5	identifier	es
2	8	identifier	un
2	11	identifier	comentario
2	22	identifier	de
2	25	identifier	bloque
3	0	identifier	que
3	4	identifier	ocupa
3	10	identifier	varias
3	17	identifier	líneas
4	0	multiplication	*
4	1	division	/
6	0	reserved_word	int
6	4	identifier	main
6	9	open_parenthesis	(
6	11	close_parenthesis)
6	13	open_brace	{
8	0	reserved_word	int
8	4	identifier	b
8	6	number	2
8	8	end_of_instruction	;
8	10	assignment	=
9	0	reserved_word	int
9	4	identifier	c
9	6	identifier	a
9	8	identifier	b
9	10	assignment	=
9	10	end_of_instruction	;
9	14	plus	+
10	0	reserved_word	return
10	7	number	0
10	9	end_of_instruction	;
11	0	close_brace	}

Resultado de lectura de comentarios de bloque y/o de línea.

source5.cpp

source5.cpp

You, 38 minutes ago | 1 author (You)

```

1  #include <iostream>
2  #include <string>
3
4  int main() {
5      std::string saludo = "Hola, mundo!";
6      if (saludo == "Hola, mundo!") {
7          std::cout << saludo << std::endl;
8      }
9      return 0;
10 }
11

```

PROBLEMS

OUTPUT

PORTS

COMMENTS

TERMINAL

Tabla de símbolos:

Línea	Posición	Tipo	Valor
1	0	preprocessor_include	#include
2	0	preprocessor_include	#include
4	0	reserved_word	int
4	4	identifier	main
4	9	open_parenthesis	(
4	11	close_parenthesis)
4	13	open_brace	{
5	0	identifier	std
5	4	reserved_word	string
5	7	scope_resolution	::
5	11	identifier	saludo
5	18	end_of_instruction	;
5	23	assignment	=
5	25	string	"Hola, mundo!"
6	0	reserved_word	if
6	3	open_parenthesis	(
6	5	identifier	saludo
6	12	close_parenthesis)
6	14	open_brace	{
6	15	equal_to	==
6	18	string	"Hola, mundo!"
7	0	identifier	std
7	4	reserved_word	cout
7	9	identifier	saludo
7	11	scope_resolution	::
7	16	identifier	std
7	18	left_shift	<<
7	20	reserved_word	endl
7	25	end_of_instruction	;
7	28	left_shift	<<
7	34	scope_resolution	::
8	0	close_brace	}
9	0	reserved_word	return
9	7	number	0
9	9	end_of_instruction	;
10	0	close_brace	}

Resultado de lectura de manejo de cadenas de texto y estructuras de control.