UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
COM10791 - COMPILADORES

UM ANALISADOR LÉXICO PARA A LINGUAGEM C--

CATTERINA VITTORAZZI SALVADOR
JOÃO VICTOR MASCARENHAS DE FARIA SANTOS
PEDRO HENRIQUE PASSOS ROCHA
RUAN VIEIRA RIBEIRO

ALEGRE
2025

CATTERINA VITTORAZZI SALVADOR - 2021201420
JOÃO VICTOR MASCARENHAS DE FARIA SANTOS - 2021200497
PEDRO HENRIQUE PASSOS ROCHA - 2021200617
RUAN VIEIRA RIBEIRO - 2021201419

UM ANALISADOR LÉXICO PARA A LINGUAGEM C--

ALEGRE
2025

SUMÁRIO

1. DESCRIÇÃO GERAL

Este trabalho tem como objetivo desenvolver um analisador léxico para a linguagem C--, uma versão simplificada da linguagem C. O analisador será responsável por processar o código-fonte escrito em C--, identificando e classificando corretamente os elementos léxicos (tokens) da linguagem.

Além do reconhecimento de tokens, a ferramenta será capaz de detectar erros léxicos e semânticos como sequências de caracteres inválidas, falta de ponto e vírgula no final de cada comando e uso de variáveis ou funções não declaradas e com isso fornecer ao usuário informações sobre essas ocorrências.

2. FUNCIONAMENTO DA IDE E DO PROGRAMA

O IDE desenvolvido para este trabalho é uma ferramenta que permite a escrita, análise e compilação de programas na linguagem C--. O ambiente foi projetado para detecção de erros léxicos e semânticos.

O analisador léxico foi implementado utilizando o JavaCC (Java Compiler Compiler), uma ferramenta que gera analisadores léxicos e sintáticos a partir de uma gramática formalmente definida. Ele é responsável por processar o fluxo de caracteres da entrada e transformá-lo em uma sequência de tokens reconhecidos pela linguagem C--. Para isso, o analisador utiliza expressões regulares para identificar lexemas correspondentes a palavras-chave, identificadores, literais numéricos, operadores e símbolos de pontuação.

Além da extração de tokens, o analisador léxico executa a remoção de comentários e espaços em branco, garantindo que apenas elementos sintaticamente relevantes sejam repassados para a análise posterior. Outra funcionalidade essencial é a detecção de erros léxicos, que ocorre quando a entrada contém sequências de caracteres não reconhecidas pela gramática definida. Nesses casos, o analisador reporta o erro, indicando precisamente a linha e a coluna onde ocorreu, permitindo uma depuração eficiente do código-fonte.

Também na gramática foi feita a configuração para evitar erros semânticos, que são aqueles referentes ao significado. Um erro semântico ocorre quando um código está sintaticamente correto, ou seja, segue as regras da linguagem de programação, mas não executa a ação desejada pelo programador. Isso acontece porque, embora o código seja válido, sua lógica está incorreta ou não corresponde à intenção original.

3. DESENVOLVIMENTO

3.1 Implementação do Analisador Léxico e Semântico

Utilizando JavaCC, foram definidos os tokens válidos e as expressões regulares correspondentes. O analisador léxico foi configurado para diferenciar letras maiúsculas de minúsculas, limitar o tamanho dos identificadores a 31 caracteres e remover comentários e espaços em branco.

Após isso o analisador foi configurado para detectar erros semânticos, como uso de variáveis ou funções não declaradas, uso de break e continue fora de laços de iteração e se o tipo de declaração da variável é compatível com o valor que será atribuído a ela.

3.2 Implementação da IDE

O IDE foi desenvolvido para integrar o analisador léxico, permitindo que o usuário escreva código, veja os erros léxicos e caso a análise tenha sucesso, o usuário também consegue visualizar os tokens consumidos em ordem, além disso a IDE marca em vermelho o local do erro.

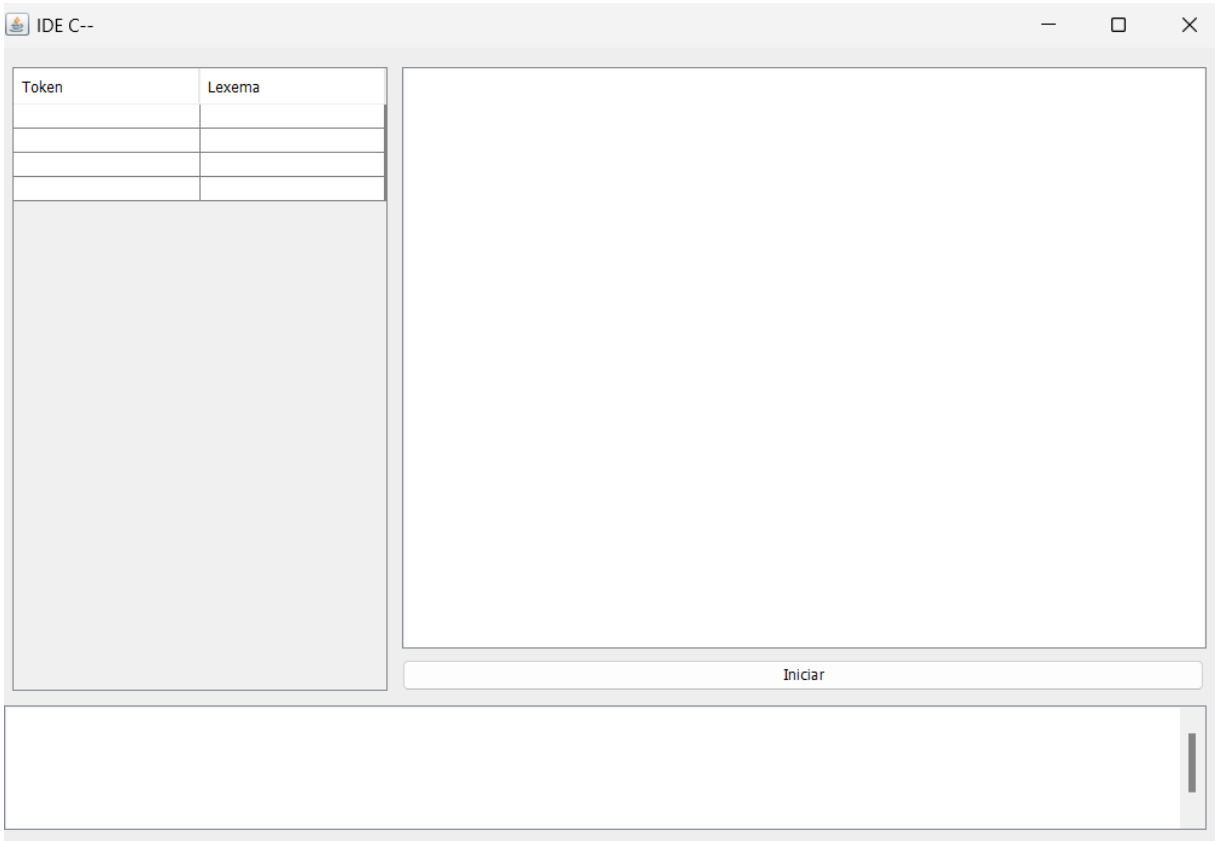


Figura 1: Visualização inicial da IDE

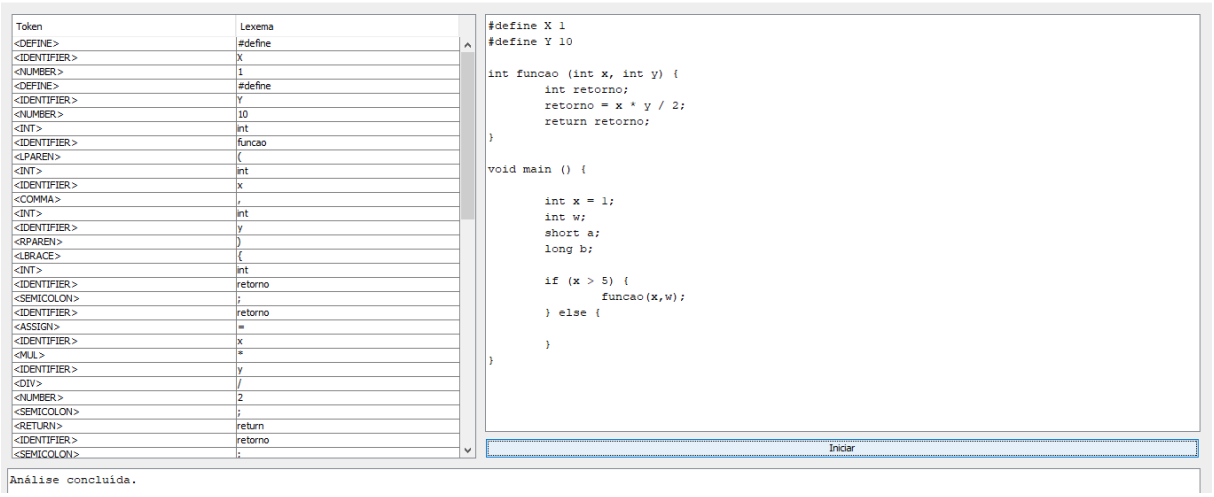


Figura 2: Lista de Tokens após análise

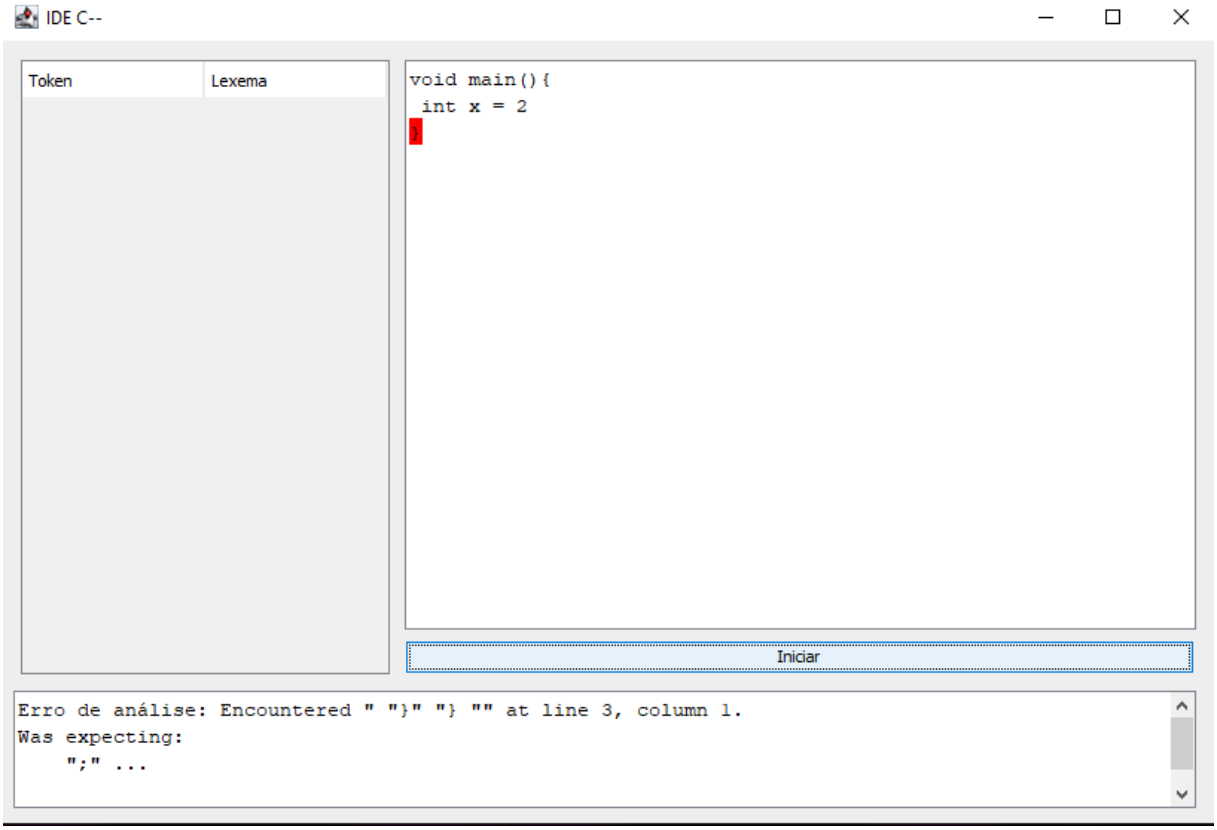


Figura 3: Marcação em vermelho do local do erro

3.3 Testes e Validação

O analisador léxico foi testado com diversos programas escritos em C-- para garantir que todos os tokens fossem reconhecidos corretamente e que os erros léxicos ou semânticos fossem detectados e reportados.

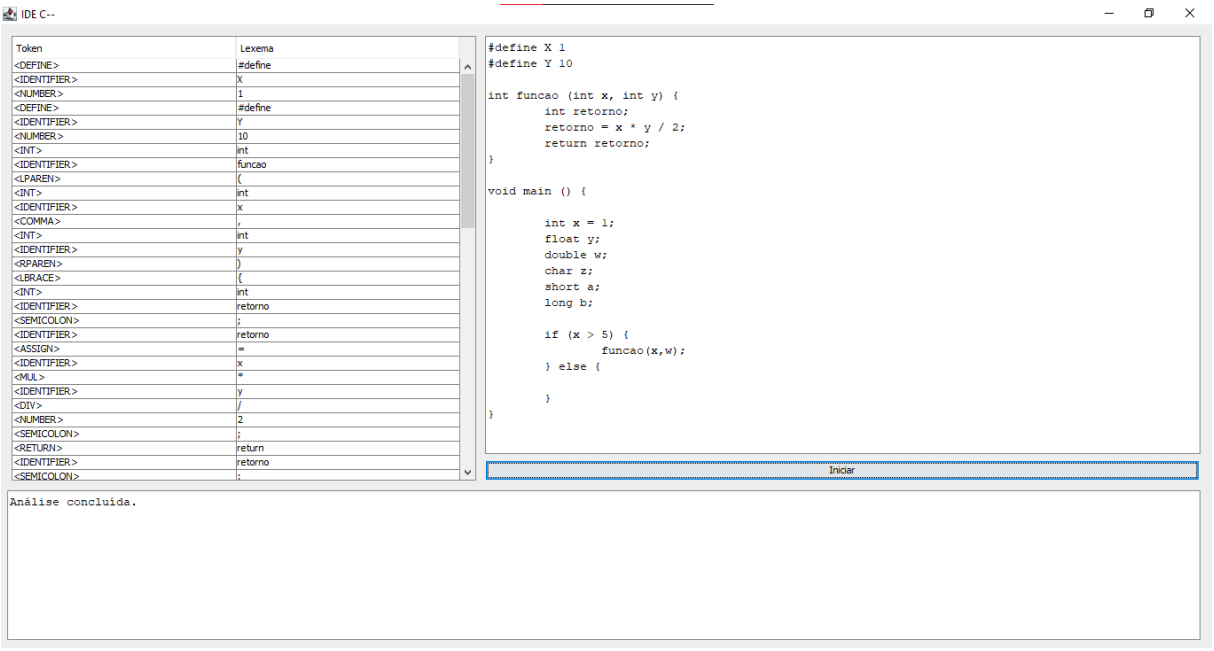


Figura 4: Exemplo de código de teste

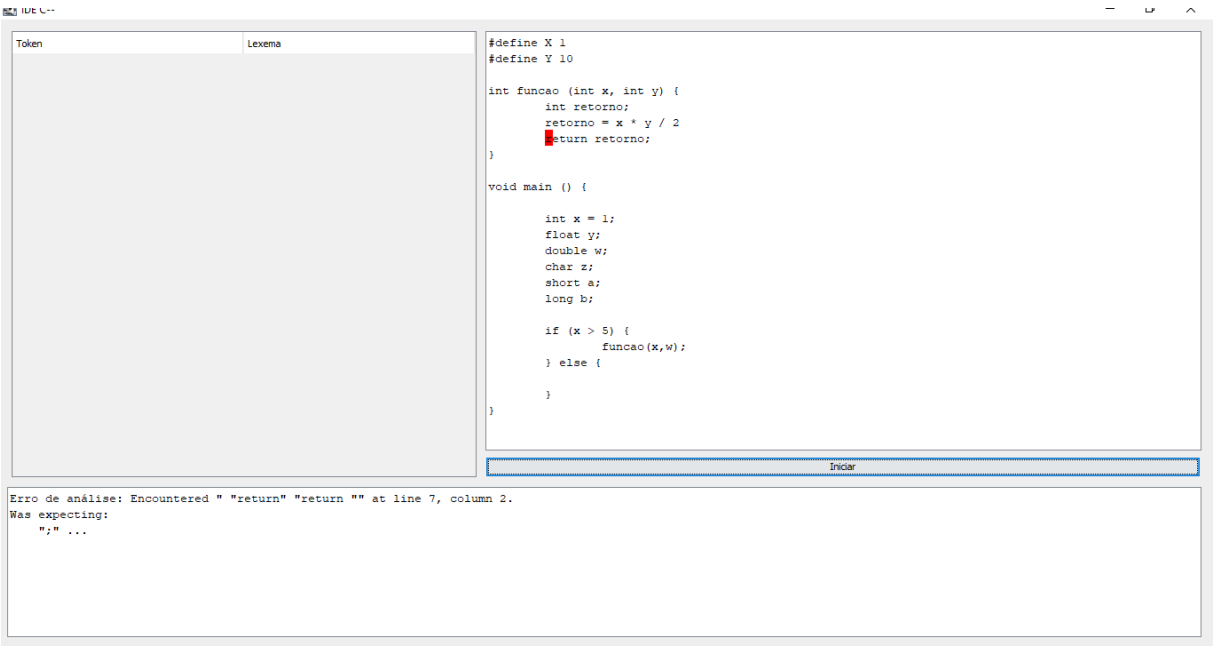


Figura 5: Exemplo de código de teste com erro léxico

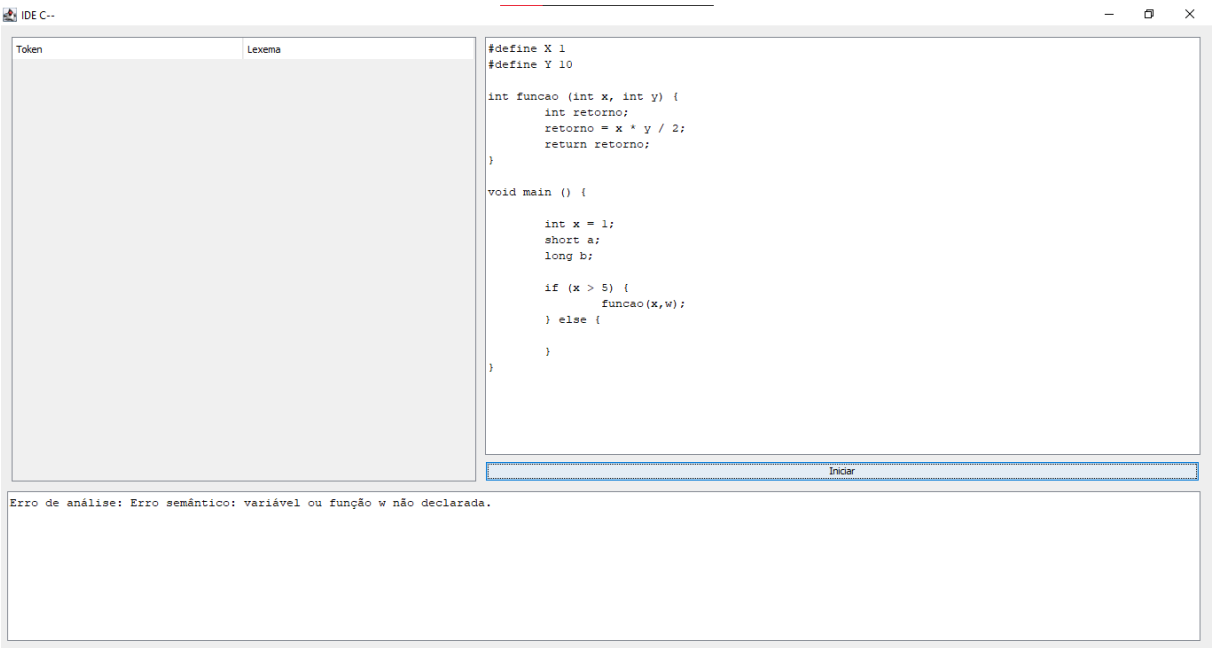
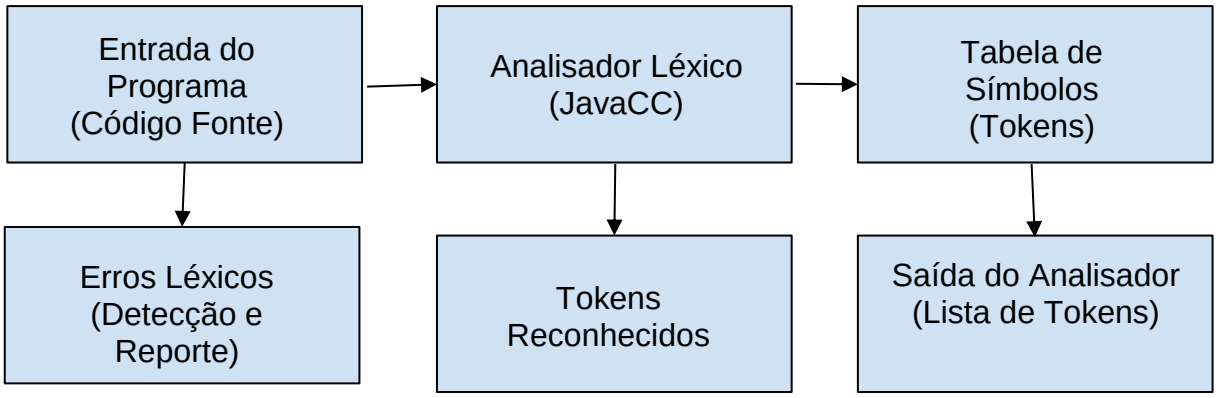


Figura 6: Exemplo de código de teste com erro semântico

4. DIAGRAMA RELACIONANDO OS MÓDULOS



5. TABELAS (TOKENS, LEXEMAS, EXPRESSÕES REGULARES, ATRIBUTOS E VALORES)

Token	Lexema	Expressão Regular	Atributo	Valor
Define	#define	#define	Palavra-Chave	#define
NUMBER	Números	[0-9]+(\.[0-9]+)?	Literal	Valor numérico
IDENTIFIER	Identificadores	[a-zA-Z_][a-zA-Z0-9_]*	Identificador	Nome da variável
ADD	+	+	Operador	+
SUB	-	-	Operador	-
MUL	*	*	Operador	*
DIV	/	/	Operador	/
MOD	%	%	Operador	%
LPAREN	(	(	Símbolo	(
RPAREN	)	)	Símbolo	)
SEMICOLON	;	;	Símbolo	;
AUTO	Auto	Auto	Palavra-Chave	Auto
STATIC	Static	Static	Palavra-Chave	Static
EXTERN	extern	extern	Palavra-Chave	extern
CONST	const	const	Palavra-Chave	const
VOID	void	void	Palavra-Chave	void

CHAR	char	char	Palavra-Chave	char
FLOAT	float	float	Palavra-Chave	float
DOUBLE	double	double	Palavra-Chave	double
SIGNED	signed	signed	Palavra-Chave	signed
UNSIGNED	unsigned	unsigned	Palavra-Chave	unsigned
SHORT	short	short	Palavra-Chave	short
INT	int	int	Palavra-Chave	int
LONG	long	long	Palavra-Chave	long
RETURN	return	return	Palavra-Chave	return
PRINTF	printf	printf	Palavra-Chave	printf
BREAK	break	break	Palavra-Chave	break
CONTINUE	continue	continue	Palavra-Chave	continue
IF	if	if	Palavra-Chave	if
ELSE	else	else	Palavra-Chave	else
OR	||	||	Operador	||
AND	&&	&&	Operador	&&
EQ	==	==	Operador	==
NEQ	!=	!=	Operador	!=
LT	<	<	Operador	<
LTE	<=	<=	Operador	<=
GT	>	>	Operador	>

GTE	>=	>=	Operador	>=
ASSIGN	=	=	Operador	=
PLUS_AS_ASSIGN	+=	+=	Operador	+=
MINUS_ASSIGN	-=	-=	Operador	-=
MUL_ASSIGN	*=	*=	Operador	*=
DIV_ASSIGN	/=	/=	Operador	/=
MOD_ASSIGN	%=	%=	Operador	%=
COMMA	,	,	Símbolo	,
LBRACKET	[	[	Símbolo	[
RBRACKET	]	]	Símbolo	]
LBRACE	{	{	Símbolo	{
RBRACE	}	}	Símbolo	}
DOT	.	.	Símbolo	.
LITERAL	Literais	“([^”]*)”	Literal	Valor do texto

5. CONCLUSÃO

Este trabalho teve como foco a implementação de um analisador léxico para a linguagem C--, utilizando a ferramenta JavaCC. O analisador foi integrado a um ambiente de desenvolvimento (IDE) que permite a escrita, análise e compilação de programas, além de oferecer detecção automática de erros léxicos para facilitar a depuração do código.

O desenvolvimento seguiu as especificações fornecidas, garantindo que todos os tokens da linguagem fossem reconhecidos corretamente. Para validar seu funcionamento, o analisador foi testado com diversos programas escritos em C--, verificando tanto o reconhecimento preciso dos tokens quanto a identificação e o reporte adequado de erros léxicos. Este relatório apresenta o funcionamento do IDE, o processo de desenvolvimento do analisador, um diagrama dos módulos que compõem o sistema e uma tabela detalhada dos tokens reconhecidos.

Além disso, na terceira parte foi feita a análise semântica do código, que verifica os erros de significado do código, fornecendo também avisos ao usuário sobre este tipo de erro.