

Manual do Usuário - Simple C

Felipe Gomes da Silva
Luis Henrique Salomão Lobato

Setembro, 2025

Sumário

1	Introdução	2
2	Testando o Analisador Léxico	2
3	Tokens Reconhecidos	3
4	Análise de Erros	4
4.1	Rastreamento de Posição	4
5	Gerenciamento de Estados	4
5.1	Comentários	4
5.2	Strings	4
6	Descrição da linguagem regular	5
7	Aceitador	5
8	Conclusão	5

1 Introdução

Este manual do usuário fornece uma visão geral do Simple C, uma linguagem de programação que visa simplificar o uso de conceitos fundamentais da linguagem C. O Simple C é projetado para ser fácil de aprender e usar, tornando-o ideal para iniciantes em programação, mas removendo algumas funcionalidades avançadas da linguagem C.

Neste momento, abordaremos apenas o analisador léxico da linguagem. Utilizamos a ferramenta Flex para construir o analisador léxico, que é responsável por identificar e classificar os tokens na entrada do código-fonte.

2 Testando o Analisador Léxico

Para testar o analisador léxico do Simple C, siga os passos abaixo:

1. Certifique-se de ter o Flex instalado em seu sistema. Você pode verificar isso executando o comando `flex -version` no terminal.
2. Clone o repositório do Simple C do GitHub:

```
git clone github.com/felipe-gsilva/simple-c
```

3. Navegue até o diretório do projeto:

```
cd simple-c/flex/
```

4. Gere o analisador léxico usando o Flex e o compile com nosso Makefile:

```
make
```

5. Execute o analisador léxico com um arquivo de entrada que contenha código Simple C:

```
./simplec < input_file.c
```

Substitua `input_file.c` pelo caminho do arquivo que você deseja analisar.

Foram criados alguns arquivos de teste na pasta `tests` do repositório. Você pode usar esses arquivos para verificar o funcionamento do analisador léxico. Cada arquivo de teste contém exemplos de código Simple C que abrangem diferentes aspectos da linguagem, como declarações de variáveis, estruturas de controle, funções, entre outros.

Tenha em vista que o analisador sintático ainda não foi implementado, portanto, mudanças na gramática podem ocorrer futuramente.

3 Tokens Reconhecidos

O analisador léxico do Simple C reconhece os seguintes tokens:

Tabela 1: Tabela de Tokens do Simple C

Nome do Token	Lexema(s)	Descrição
Palavras-chave: Tipos de Dados		
KEYWORD_INT	int	Palavra-chave para tipo inteiro.
KEYWORD_FLOAT	float	Palavra-chave para tipo ponto flutuante.
KEYWORD_CHAR	char	Palavra-chave para tipo caractere.
KEYWORD_STRING	string	Palavra-chave para tipo string (cadeia de caracteres).
KEYWORD_VOID	void	Palavra-chave para tipo vazio/nulo.
KEYWORD_BOOL	bool	Palavra-chave para tipo booleano.
Palavras-chave: Controle de Fluxo e Comandos		
KEYWORD_IF	if	Inicia uma estrutura condicional.
KEYWORD_ELSE	else	Bloco alternativo de uma estrutura condicional.
KEYWORD_FOR	for	Inicia um laço de repetição 'for'.
KEYWORD_WHILE	while	Inicia um laço de repetição 'while'.
KEYWORD_DO	do	Inicia um laço de repetição 'do-while'.
KEYWORD_SWITCH	switch	Inicia uma estrutura de seleção múltipla.
KEYWORD_CASE	case	Define um rótulo dentro de um 'switch'.
KEYWORD_DEFAULT	default	Define o rótulo padrão de um 'switch'.
KEYWORD_BREAK	break	Interrompe a execução de um laço ou 'switch'.
KEYWORD_CONTINUE	continue	Pula para a próxima iteração de um laço.
KEYWORD_RETURN	return	Retorna um valor de uma função.
Identificadores e Literais		
IDENTIFICADOR	var, _x, ...	Nome de variável, função, etc.
INT	123, 42	Valor literal inteiro.
FLOAT	3.14, 0.5	Valor literal de ponto flutuante.
CHAR	'a', '\n'	Valor literal de caractere.
STRING	"hello"	Valor literal de string.
BOOLEAN_LITERAL	true, false	Valor literal booleano.
Operadores		
OP_SOMA	+	Operador de adição.
OP_SUB	-	Operador de subtração.
OP_MULT	*	Operador de multiplicação.
OP_DIV	/	Operador de divisão.
OP_MOD	%	Operador de módulo (resto da divisão).
OP_ATRIBUICAO	=	Operador de atribuição simples.
OP_INC_ATRIBUICAO	+=	Operador de atribuição com adição.
OP_DEC_ATRIBUICAO	-=	Operador de atribuição com subtração.
OP_MULT_ATRIBUICAO	*=	Operador de atribuição com multiplicação.
OP_DIV_ATRIBUICAO	/=	Operador de atribuição com divisão.
OP_INC	++	Operador de incremento.
OP_DEC	--	Operador de decremento.

Tabela 1: Tabela de Tokens do Simple C (Continuação)

Nome do Token	Lexema(s)	Descrição
OP_IGUAL	==	Operador relacional de igualdade.
OP_DIFERENTE	!=	Operador relacional de desigualdade.
OP_MENOR	<	Operador relacional menor que.
OP_MAIOR	>	Operador relacional maior que.
OP_MENOR_IGUAL	<=	Operador relacional menor ou igual que.
OP_MAIOR_IGUAL	>=	Operador relacional maior ou igual que.
OP_AND	&&	Operador lógico E (AND).
OP_OR		Operador lógico OU (OR).
OP_NOT	!	Operador lógico de negação (NOT).
Pontuadores e Delimitadores		
PONTO_VIRGULA	;	Finalizador de instrução.
DOIS_PONTOS	:	Usado em casos de 'switch'.
VIRGULA	,	Separador de elementos (ex: em listas).
ABRE_PARENTESES	(Abre lista de parâmetros ou expressão.
FECHA_PARENTESES)	Fecha lista de parâmetros ou expressão.
ABRE_CHAVES	{	Abre um bloco de código.
FECHA_CHAVES	}	Fecha um bloco de código.
ABRE_COLCHETES	[Abre a declaração/acesso de um array.
FECHA_COLCHETES]	Fecha a declaração/acesso de um array.

4 Análise de Erros

A função `print_error` exibe as mensagens de erro léxico e inclui a linha, coluna e token que causaram o erro. Para isto, utilizamos rastreamento de posição durante toda a leitura do código-fonte.

4.1 Rastreamento de Posição

Para tratar os erros léxicos de forma eficiente, o analisador léxico do Simple C implementa um mecanismo de rastreamento de posição. Para isto, são declaradas 2 variáveis globais, `current_line` e `current_col`, que armazenam a linha e a coluna atuais do analisador léxico, respectivamente. Essas variáveis são atualizadas conforme o analisador lê o código-fonte, permitindo que mensagens de erro incluam informações precisas sobre a localização do erro.

5 Gerenciamento de Estados

Para melhor gerenciar o estado do analisador léxico, utilizamos a funcionalidade de estados do Flex. Em especial, foram criados 2 estados diferentes: `IN_COMMENT` e `IN_STRING`.

5.1 Comentários

Os comentários podem aparecer de 2 formas no código: (i) comentários de linha única, que começam com `//` e se estendem até o final da linha; (ii) comentários de bloco, que começam com `/*` e terminam com `*/`.

5.2 Strings

As strings são sequências de caracteres delimitadas por aspas duplas (`"`). O analisador léxico reconhece strings e trata caracteres de escape, como `\n` para nova linha e `\t` para tabulação.

Para isso, foi-se utilizado da função `yymore()` do Flex, que permite concatenar a string lida em `yytext` com a próxima parte da string lida, até que o caractere de fechamento (") seja encontrado.

6 Descrição da linguagem regular

7 Aceitador

8 Conclusão