Alexandra Melo
Brenda Moreira da Silva
João Lucas Pereira Ramalho
Rafaela Valadão Nunes

Trabalho de Compiladores

Criação da linguagem BREJA

UFLA
2024

Nome da Linguagem: BREJA

Descrição das Características Gerais da Linguagem:

BREJA é uma linguagem de programação projetada para ser divertida, acessível e intuitiva, ideal tanto para iniciantes quanto para programadores experientes. Com uma sintaxe clara e amigável, BREJA permite a construção de algoritmos de forma simples e expressiva, favorecendo a criatividade e a experimentação.

Principais Características:

- Tipagem Forte: BREJA oferece suporte a seis tipos de dados fundamentais:

  Contador: Representa números inteiros.
  Quebradinho: Representa números reais.
  VaiouRacha: Representa valores booleanos (verdadeiro ou falso).
  DizAi: Representa cadeias de caracteres, facilitando a manipulação de texto.
  Pequeninho: Representa um unico caractere;

- Definição de Funções: A linguagem permite a criação de funções com e sem retorno. Sem retorno sendo definida pela palavra “soFaz” e com retorno “meDaIsso”.

- Definição de Funções: Para atribuir o valor a uma variavel utiliza o “=”.

- Operações Aritméticas: BREJA inclui uma variedade de operadores aritméticos soma, subtração, multiplicação, divisão e resto sendo respectivamente os símbolos +,-,*,/,%

- Operações Relacionais: igualdade, desigualdade, maior, maior ou igual, menor, menor ou igual sendo os simbolos ==,!=,<, <= ,>, >=, facilitando cálculos e comparações.

- Concatenação de Cadeias: A linguagem possui um operador especial para a concatenação de cadeias de caracteres, chamado “colaPalavra”.

- Estruturas de Controle: BREJA oferece suporte a estruturas de repetição (“RepeteAte” para while “perercorreTudo” para for) e estruturas condicionais (seFaz para "if" e seNao para "else").

- Comandos de Entrada e Saída: BREJA inclui comandos específicos para leitura (leiaIsso) e escrita (escreveIsso), facilitando a interação com o usuário.

- Suporte a Estruturas de Bloco: A linguagem permite a organização do código em blocos, utilizando chaves ({}) e parentese(( )), melhorando a legibilidade e a manutenção do código.

Definição Léxica da Linguagem

Classe	Padrão	Lexemas Aceitos	Sigla
Identificadores	Inicia com letra minuscula, seguido por letras ou dígitos [a-z] [a-zA-Z0-9]*	nomes, carro1, telefoneContato.	Id
Números Inteiros	Sequencias de digitos entre [0...9]	0, 123, 8	num
Numeros Reais	Sequencia de digitos separados por ponto (.)	12.3 , 1.5 , 9.8	numReal
Caracteres	Cadeia de caracteres iniciada e terminada com aspas	“APROVADO”, “Hello Word”, “rosa”	cad
Estrutura condicional	o próprio lexema	seFaz, seNao	cond
Estrutura de repeticao	o próprio lexema	repeteAte, pecorreTudo	rep
Atribuição	o próprio lexema	=	atr
Operadores Aritméticos	o próprio lexema	+,-,*,/,%	opar
Operadores Relacionais	o próprio lexema	==, !=, <, <=, >, >=	oprl
Operadores Lógicos	o próprio lexema	e, ou , não	oplog
Comandos de leitura	o próprio lexema	leiaIsso	ler
Comandos de escritta	o próprio lexema	escreveIsso	escrever

Retorno da função	o próprio lexema	meDaIsso	o próprio lexema
Função sem retorno	o próprio lexema	soFaz	o próprio lexema
Concatenação de palavras	o próprio lexema	colaPalavra	o próprio lexema
Identificador de tipo	o próprio lexema	pequeninho, contador, quebradinho, vaiouRacha, dizAi	o próprio lexema
Símbolo para terminar instruções	o próprio lexema	;	dlop
Símbolo de fechamento de bloco	o próprio lexema	}	fc
Símbolo de abetura de bloco	o próprio lexema	{	ac
abertura de parênteses	o próprio lexema	(	ap
fechamento de parênteses	o próprio lexema	}	fp

Exemplos de uso da linguagem:

Algoritmos de Fatorial

```
contador fatorial(contador n) {
    contador resultado = 1;
    percorreTudo (contador i = 2; i <= n; i++) {
        resultado = resultado *  i;
     }
    meDaIsso resultado;
}
soFaz main() {
     contador = 5;
     contador resultado = fatorial(n);
       escreveIsso resultado;
  }
```

Fibonacci

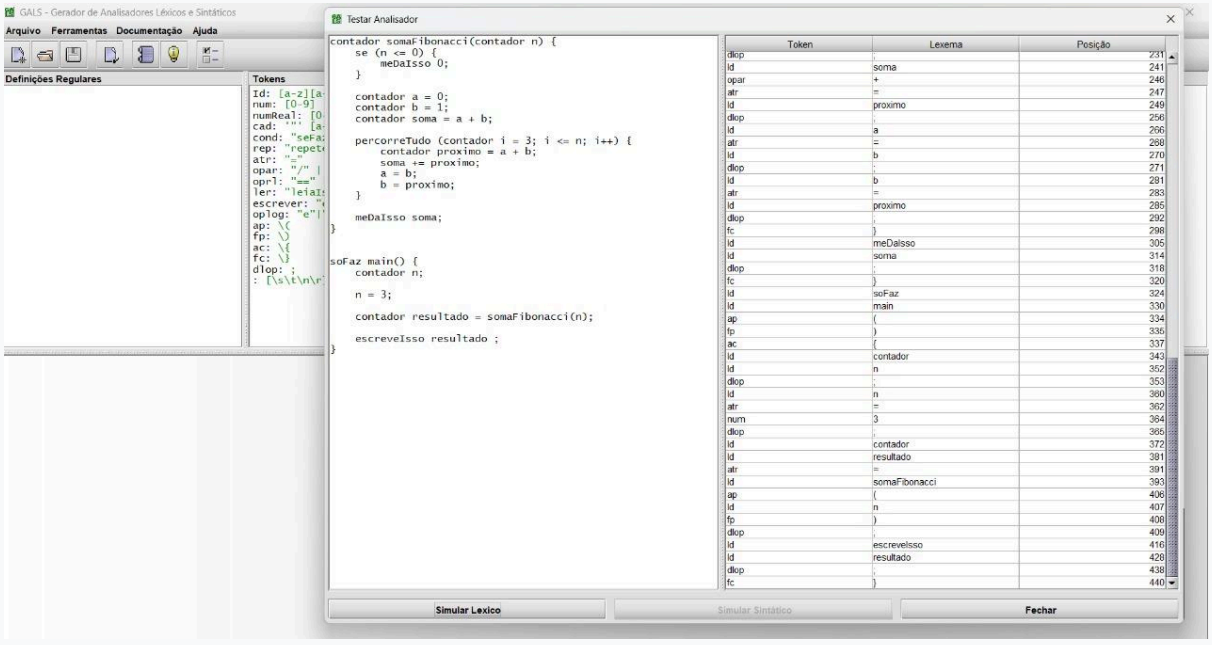
```
contador somaFibonacci(contador n) {
    se (n <= 0) {
        meDaIsso 0;
    }

    contador a = 0;
    contador b = 1;
    contador soma = a + b;

    percorreTudo (contador i = 3; i <= n; i++) {
        contador proximo = a + b;
        soma += proximo;
        a = b;
        b = proximo;
    }

    meDaIsso soma;
}


soFaz main() {
    contador n;

    n = 3;

    contador resultado = somaFibonacci(n);

    escreveIsso resultado ;
}
```

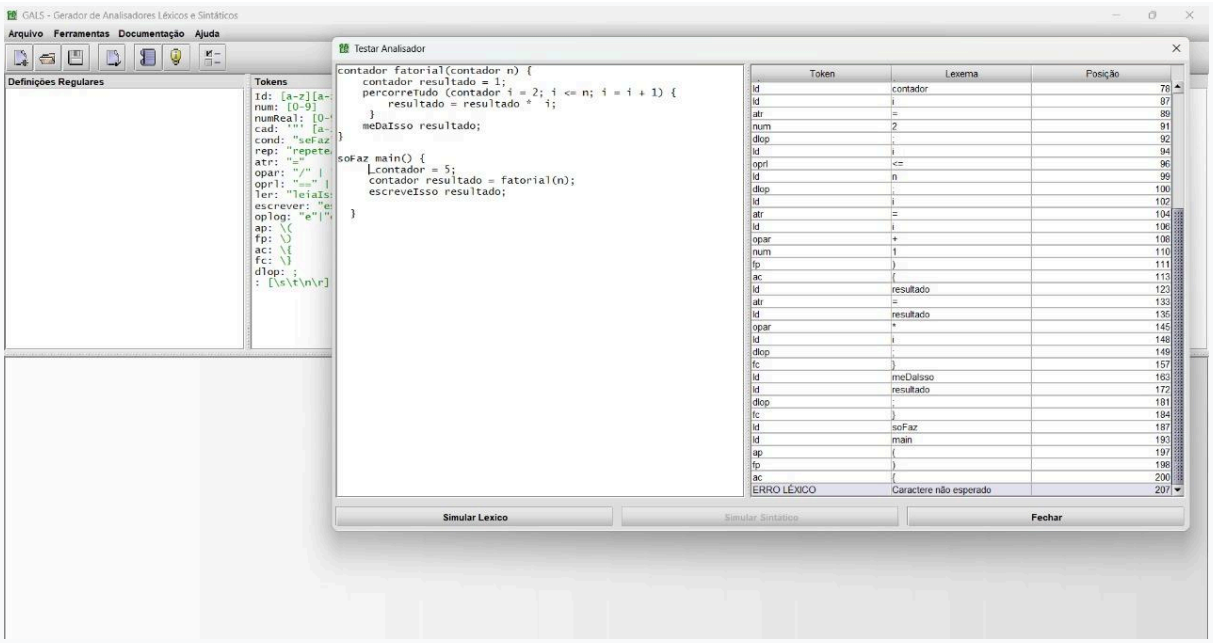
Implementação do Analisador Léxico

Criação da gramatica: transformamos os padões em expreções regularespara poder montar o nosso compilador

```
num: [0-9]&#13;
numReal: [0-9]+"."[0-9]+&#13;
cad: '"' [a-z A-Z 0-9]* '"'&#13;
cond: "seFaz"|"seNao"&#13;
rep: "repeteAte"|"percorreTudo"&#13;
atr: "="&#13;
opar: "/" | "%" | "*" | "+" | "-"&#13;
oprl: "==" | "!=" | "&lt;" | "&lt;="| "&gt;" |"&gt;="&#13;
ler: "leiaIsso"&#13;
escrever: "escrevaIsso"&#13;
oplog: "e"|"ou"|"não"&#13;
ap: \(&#13;
fp: \)&#13;
ac: \{&#13;
fc: \}&#13;
dlop: ;&#13;
: [\s\t\n\r]&#13;
```

Casos testes





Link github: https://github.com/brendamoreira06/Trabalho-de-Compiladores