



UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - CAMPUS FLORESTAL

COMPILADORES

# Trabalho Prático 1

Aymê Faustino dos Santos - [aymesantos](#)

Emily Lopes Almeida - [Emily-Lopes](#)

Ingred Fonseca de Almeida - [ingredalmeida1](#)

Letícia Oliveira Silva - [leticiasilvaa](#)

Florestal - MG

2024

# Sumário

<b>1 Introdução</b>	<b>3</b>
<b>2 Especificação da Linguagem</b>	<b>3</b>
2.1 Nome e Origem do Nome	3
2.2 Tipos de Dados Primitivos	4
2.3 Operadores	4
2.4 Variáveis	6
2.5 Estruturas de Dados	6
2.6 Comandos Disponíveis	7
2.7 Estrutura do Código	10
2.8 Palavras-Chave e Palavras Reservadas	10
<b>3 Gramática</b>	<b>11</b>
3.1 Produções da Gramática	11
3.2 Exemplo de Derivação	13
<b>4 Analisador Léxico</b>	<b>21</b>
4.1 Definições Regulares	21
4.2 Arquivo lex.l	22
<b>5 Testes de Execução</b>	<b>23</b>
5.1 Repetição	23
5.2 Condicionais	25
5.3 Funções	27
5.4 Estruturas de dados	29
5.5 Exemplo Geral	31
<b>6 Referências</b>	<b>35</b>

# 1 Introdução

Este trabalho prático consiste em especificar uma nova linguagem de programação e implementar seu analisador léxico. A nova linguagem é a linguagem procedural `TripCode`, com características que envolvem o programador na atmosfera de viagens. Sua especificação inclui: nome da linguagem, tipos de dados primitivos, comandos disponíveis e a gramática da linguagem.

O analisador léxico será implementado utilizando o gerador de analisador léxico `FLEX`, uma versão mais recente do `LEX`, juntamente com a linguagem de programação `C`. Essa ferramenta permite especificar um analisador léxico a partir da definição de expressões regulares que são responsáveis por descrever os padrões para os tokens que devem ser reconhecidos. Nesta etapa do trabalho, as ações associadas a cada token serão de impressão na tela para apenas demonstrar que os tokens estão sendo corretamente identificados.

## 2 Especificação da Linguagem

A linguagem de programação `TripCode` é procedural, possui verificação estática de tipos e, assim como na linguagem `C`, possui um programa principal denominado `trip()`. Nela, o tamanho da letra é significativo, ou seja, é *case sensitive*, e os identificadores podem conter tanto letras maiúsculas quanto minúsculas, números e o caractere sublinhado (`_`), mas devem sempre começar com letra minúscula.

O escopo da linguagem é estático e os blocos são definidos utilizando `>>>` e `<<<`, portanto, a indentação é utilizada apenas para legibilidade do código. Para realizar comentários, o programador deve colocá-los entre os símbolos `->` e `<-`, o que permite com comentários multilinhas. Vale ressaltar que em `TripCode` comentários não podem ser aninhados.

### 2.1 Nome e Origem do Nome

O nome `TripCode` reflete tanto o propósito quanto o contexto temático da linguagem. Ele é uma combinação dos termos distintos:

- `Trip`: faz referência ao contexto temático da linguagem, que é o setor de viagens.
- `Code`: sublinha a função principal da linguagem de programação, que é permitir a criação de códigos.

Além disso, a letra `C` faz uma referência implícita à linguagem `C`, linguagem na qual `TripCode` foi inspirada.



Figura 1 - Logo da linguagem `TripCode`

## 2.2 Tipos de Dados Primitivos

Os tipos de dados primitivos da linguagem `TripCode` são booleanos, inteiros, reais e strings. Como o objetivo é especificar uma linguagem temática, a escolha dos nomes para os tipos primitivos de dados reflete diretamente o tema central de férias e viagens.

Para representar o tipo booleano, `FERIAS` representa valor verdadeiro, uma vez que é um período em que viagens são mais frequentes e `DIAUTIL` representa valor falso, por ser uma expressão utilizada para representar dias de trabalho. Como a linguagem `TripCode` é uma linguagem com verificação estática de tipos, para declarar uma variável do tipo booleano é preciso utilizar `STATUS`.

Uma variável do tipo inteiro deve ser declarada utilizando `MILHAS`. Esse termo faz referência à bonificações dadas por companhias aos clientes como pontos que podem ser posteriormente trocados por viagens, e esses pontos são sempre valores inteiros. Já para números de ponto flutuante o termo que deve ser utilizado é `DOLAR` para fazer referência ao aspecto financeiro das viagens. Essa escolha reflete a precisão decimal necessária para lidar com valores monetários. Para separar a parte inteira da parte decimal deve ser utilizado ponto(.).

Na linguagem `TripCode` não existe o tipo caractere uma vez que ele pode ser representado por uma string de tamanho um. Para declarar uma variável do tipo string deve-se utilizar `VOUCHER`, considerando que para fazer uma viagem uma pessoa precisa sempre de um documento. Além disso, é permitido o uso tanto de aspas simples quanto duplas proporcionando flexibilidade ao programador na escrita e manipulação desses tipos de dados.

## 2.3 Operadores

A linguagem `TripCode` oferece suporte a operadores, que são símbolos ou palavras-chave que realizam operações específicas sobre valores ou variáveis. Com esses operadores, é possível criar expressões, que representam as operações que o programa executa para manipular dados e calcular resultados.

Operador	Descrição	Exemplo
*	multiplicação	<code>MILHAS a &lt;-&gt; a * b;</code>
/	divisão	<code>MILHAS a &lt;-&gt; a / b;</code>
+	soma	<code>MILHAS a &lt;-&gt; a + b;</code>
-	subtração	<code>MILHAS a &lt;-&gt; a - b;</code>
%	módulo	<code>MILHAS a &lt;-&gt; a % b;</code>

Tabela 1 - Operadores Aritméticos

Operador	Descrição	Exemplo
<code>DESCANSAR</code>	Incremento	<code>i DESCANSAR;</code>
<code>TRABALHAR</code>	Decremento	<code>i TRABALHAR;</code>

Tabela 2 - Operadores de Incremento e Decremento

Operador	Descrição	Exemplo
=	igual	STATUS a <-> a = b;
>	maior que	STATUS a <-> a > b;
>=	maior ou igual	STATUS a <-> a >= b;
<	menor que	STATUS a <-> a < b;
<=	menor ou igual	STATUS a <-> a <= b;
#	diferente	STATUS a <-> a # b;

**Tabela 3 - Operadores Relacionais**

Operador	Exemplo
AND	STATUS a <-> a = 5 AND b = 10;
OR	STATUS a <-> a = 5 OR b = 10;
NOT	STATUS a <-> NOT a = 5;

**Tabela 4 - Operadores Lógicos**

Operador	Descrição	Exemplo
<->	Atribuição Simples	a <-> b;

**Tabela 5 - Operador de Atribuição**

Com relação ao operador de atribuição, é importante destacar que é possível realizar múltiplas atribuições desde que todas as variáveis sejam do mesmo tipo e recebam o mesmo valor, exceto atribuições do tipo MAPA e PASSAPORTE onde cada posição ou campo da estrutura deve ser acessado individualmente para realizar a atribuição.

```
x, y, z <-> 50;
```

**Figura 2 - Exemplo de Atribuição Múltipla**

A linguagem TripCode também possui os operadores CAMBIO(), COTACAO() e CLASSE() que são responsáveis, respectivamente, pela conversão de tipo, cálculo do tamanho de um tipo e retorno do tipo.

```
-> sintaxe dos operadores <-
CAMBIO(<novo_tipo>,<variavel>);
COTACAO(<tipo>);
CLASSE(<variavel>);
```

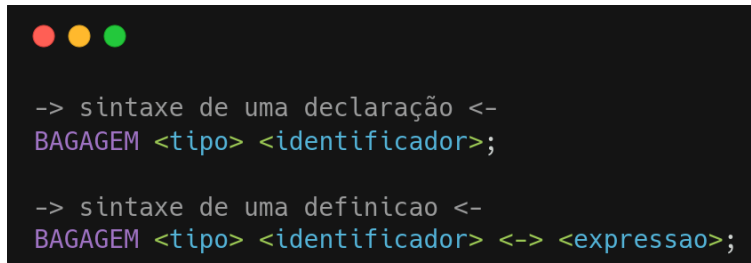
**Figura 3 - Sintaxe dos Operadores CAMBIO(), COTACAO() e CLASSE()**

```
-> exemplos <-
CAMBIO(DOLAR,num);
COTACAO(DOLAR);
CLASSE(num);
```

**Figura 4 - Exemplo dos Operadores CAMBIO(), COTACAO() e CLASSE()**

## 2.4 Variáveis

Todas as variáveis em `TripCode` devem ser declaradas e definidas antes de serem utilizadas. Para declarar variáveis de tipos primitivos, deve-se utilizar a diretiva `BAGAGEM` seguida do tipo da variável e então de seu identificador. Assim como é possível fazer múltiplas atribuições, também é possível fazer declarações e definições múltiplas, também apenas de tipos primitivos.



```
-> sintaxe de uma declaração <-  
BAGAGEM <tipo> <identificador>;  
  
-> sintaxe de uma definicao <-  
BAGAGEM <tipo> <identificador> <-> <expressao>;
```

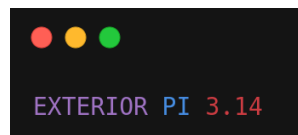
Figura 5 - Sintaxes de declaração e definição de variáveis



```
-> exemplos <-  
BAGAGEM MILHAS distancia;  
BAGAGEM DOLAR preco <-> 5.25;  
BAGAGEM VOUCHER nome <-> 'Daniel';  
BAGAGEM STATUS promocao <-> FERIAS;  
BAGAGEM MILHAS x, y, z <-> 5;
```

Figura 6 - Exemplos de declaração e definição de variáveis

Para definir constantes basta utilizar a diretiva `EXTERIOR` seguida do identificador e do valor que será associado a esse identificador, assim como mostra a figura 7. Vale ressaltar que a definição de constantes deve ser feita no início do código fonte.

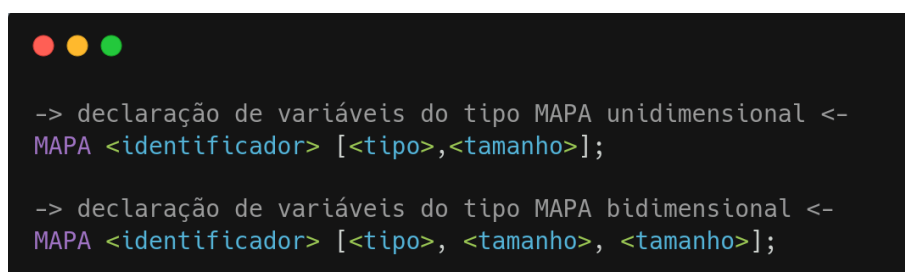


```
EXTERIOR PI 3.14
```

Figura 7 - Exemplo de declaração de constante

## 2.5 Estruturas de Dados

Como uma forma complementar aos tipos primitivos de dados, a linguagem `TripCode` dá suporte a estruturas de dados. Como referência ao array da linguagem C, tem-se o tipo `MAPA`, que podem ser unidimensionais ou bidimensionais, e como referência às estruturas, que são coleção de dados que atuam como um todo, tem-se o tipo `PASSAPORTE`. É importante ressaltar que o índice do tipo `MAPA` inicia na posição 0.



```
-> declaração de variáveis do tipo MAPA unidimensional <-  
MAPA <identificador> [<tipo>, <tamanho>];  
  
-> declaração de variáveis do tipo MAPA bidimensional <-  
MAPA <identificador> [<tipo>, <tamanho>, <tamanho>];
```

Figura 8 - Sintaxe de declaração do tipo MAPA

```

MAPA nomes [VOUCHER, 20];
nomes[5] <-> "Teste";
nome <-> nomes[5]; -> Acessar posição 5 do MAPA nomes <-

MAPA matriz [MILHAS, 10, 50]; -> 10 linhas e 50 colunas <-
matriz[0,0] <-> 5;

```

Figura 9 - Exemplos de uso do tipo MAPA

```

-> criar estrutura <-
PASSAPORTE <identificador_estrutura> >>>
    BAGAGEM <tipo> <identificador>;
    ...
<<<

-> declarar variável do tipo PASSAPORTE <-
PASSAPORTE <identificador_estrutura> <identificador>;

-> para acessar um membro de uma estrutura basta utilizar ponto(.) <-
<identificador>.<identificador_atributo>;

PASSAPORTE minha_struct >>>
    BAGAGEM DOLAR ricas2;
    -> pode ter quantas quiser <-
<<<

minha_struct.ricas2;

```

Figura 10 - Sintaxe e exemplos do tipo PASSAPORTE

## 2.6 Comandos Disponíveis

A linguagem `TripCode` possui comandos de leitura de dados do teclado e exibição dos dados na tela. Para leitura de dados o comando disponível é o `CHECKIN()` que recebe como parâmetros um vetor com os tipos de dados e em seguida uma lista com o identificador das variáveis onde o valor recebido da entrada deve ser armazenado, assim como mostra a Figura 11. Já para a exibição de dados na tela tem-se o comando `CHECKOUT()` que recebe como parâmetros uma string de controle e concatena dados utilizando o operador '+'. Os nomes dos comandos fazem referência aos termos utilizados em hotéis e viagens de avião para confirmar a entrada ou saída de hóspedes ou passageiros, da mesma forma nos comandos: `CHECKIN()` define uma entrada, ou seja que dados estão sendo recebidos e `CHECKOUT()` define saída, ou seja dados serão impressos e apresentados ao usuário.

```

-> comando de entrada <-
CHECKIN([MILHAS, DOLAR, VOUCHER], a,b,c);

-> comando de saída <-
CHECKOUT("Exibir" + MILHAS a + DOLAR b + "na Tela");

```

Figura 11 - Exemplo dos comandos de entrada e saída

Quanto aos comandos condicionais, a linguagem TripCode possui comandos equivalentes ao `if`, `if-else`, e `if aninhados` da linguagem C, porém utilizando os termos temáticos relacionados ao conceito de ALFANDEGA. `if` na linguagem TripCode, é um ponto de verificação relacionada ao pagamento de impostos, e por isso, `ISENTO` representa o `else` e `TRIBUTADO` representa `else-if`. A Figura 12 mostra alguns exemplos de como esses comandos podem ser utilizados. Nela é possível perceber, que independente se ALFANDEGA possuir uma ou mais sentenças é necessário utilizar o delimitador de bloco `>>>` e `<<<`.

```
-> 1º exemplo de condicional <-
ALFANDEGA (distancia > 20) >>>
    x <-> 30;
<<<

-> 2º exemplo de condicional <-
ALFANDEGA (distancia < 20) >>>
    CHECKOUT("Distancia =" + MILHAS distancia);
<<< ISENTO >>>
    x <-> 30;
<<<

-> 3º exemplo de condicional <-
ALFANDEGA (distancia = 90) >>>
    CHECKOUT("Você chegou ao seu destino");
<<< TRIBUTADO (distancia > 50) >>>
    CHECKOUT("Você passou da metade do caminho");
<<< TRIBUTADO (distancia = 45) >>>
    CHECKOUT("Você está exateme nte na metade do caminho");
<<< ISENTO >>>
    CHECKOUT("Você está apenas no início do caminho");
<<<
```

Figura 12 - Exemplos do uso de comando condicional

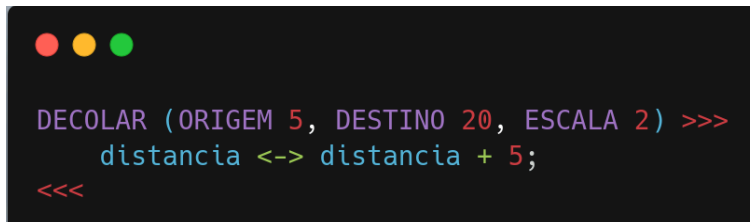
Com intuito de simplificar a tomada de decisões com base no valor de uma variável, a linguagem TripCode também possui o comando `ITINERARIO` que faz referência ao comando `switch`, no qual cada caso é precedido por `ROTA` e o termo default é representado por `IMPREVISTO`. Nesse contexto vale ressaltar que a linguagem também dá suporte ao comando de interrupção `POUSAR` que faz referência ao `break`.

```
ITINERARIO(distancia) >>>
    ROTA 90:
        CHECKOUT("Você chegou ao seu destino");
        POUSAR;
    ROTA 45:
        CHECKOUT("Você está exateme nte na metade do caminho");
        POUSAR;
    IMPREVISTO:
        CHECKOUT("Você está apenas no início do caminho");
        POUSAR;
<<<
```

Figura 13 - Exemplo do uso de switch na linguagem TripCode

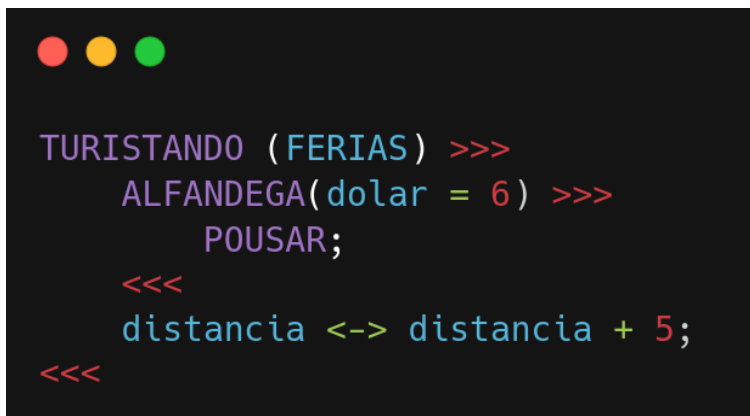


Para comandos de repetição, a linguagem possui equivalentes ao `while` e `do-while`, com sintaxes bem similares à da linguagem C, onde apenas o nome dos comandos e o delimitador de bloco foi alterado, como está apresentado nas Figuras 15 e 16. Além disso, também foi definido um equivalente para o `for`, no qual é necessário especificar o passo, denominado `ESCALA`, que pode ser positivo para incrementar, ou negativo para decrementar, o valor definido em `ORIGEM` até que ele se iguale ao valor definido em `DESTINO`, ou vice-versa, que são sempre valores inteiros e para utilizar variáveis elas precisam ter sido definidas anteriormente. O comando `POUSAR` também é utilizado nos comandos de repetição representando o tradicional `break`. Ademais a linguagem oferece suporte ao comando `continue` que é representado por `FERIADO` fazendo referência à possibilidade de prolongar uma viagem devido a existência de um feriado.



```
DECOLAR (ORIGEM 5, DESTINO 20, ESCALA 2) >>>
    distancia <-> distancia + 5;
<<<
```

Figura 14 - Exemplo do uso de FOR na linguagem TripCode



```
TURISTANDO (FERIAS) >>>
    ALFANDEGA(dolar = 6) >>>
    POUSAR;
<<<
    distancia <-> distancia + 5;
<<<
```

Figura 15 - Exemplo do uso de WHILE na linguagem TripCode



```
TURISTAR >>>
    distancia <-> distancia + 5;
<<< DURANTE (dolar < 6);
```

Figura 16 - Exemplo do uso de DO-WHILE na linguagem TripCode

Além disso, é permitido que funções sejam criadas em TripCode utilizando a sentença `ROTEIRO` antes do identificador e sempre especificando o tipo de retorno, além de que o protótipo destas funções deve ser previamente declarado antes do programa principal da linguagem `trip()` e sua especificação deve estar após o programa principal. Além disso, para ativar uma função criada pelo programador deve-se utilizar a sentença `EMBARCAR` e para devolver o controle para a função ativadora, deve-se utilizar a sentença `DESPACHAR` que faz referência ao comando `return` da linguagem C.

```
-> protótipo da função <-
ROTEIRO identificador (tipo1 param1, tipo2 param2) >>> <tipo_retorno> <<<

-> definição da função <-
ROTEIRO identificador (tipo1 param1, tipo2 param2) >>>
    declaração_variaveis_locais
    sentença
    sentença
    ...
    DESPACHAR expressao;
<<< <tipo_retorno> <<<

-> chamar função <-
EMBARCAR identificador (param1, param2);
```

Figura 17 - Exemplo de uso de funções na linguagem TripCode

2.7 Estrutura do Código

Um código da linguagem TripCode deve ser feito por completo em apenas um arquivo seguindo a seguinte ordem:

definição de constantes
declaração e/ou definição de variáveis globais
declaração de estruturas do tipo PASSAPORTE
protótipo das funções criadas pelo programador
programa principal trip()
funções definidas pelo programador

2.8 Palavras-Chave e Palavras Reservadas

Devido ao fato de palavras-chaves não serem reservadas dificultar o projeto do analisador léxico, na linguagem TripCode todas as palavras-chave são reservadas e elas estão destacadas na tabela abaixo.

FERIAS	DESCANSAR	BAGAGEM	POUSAR	TURISTANDO
DIAUTIL	TRABALHAR	EXTERIOR	DECOLAR	TURISTAR
STATUS	AND	ALFANDEGA	ORIGEM	DURANTE
VOUCHER	OR	TRIBUTADO	DESTINO	ROTEIRO
MILHAS	NOT	ISENTO	ESCALA	EMBARCAR
DOLAR	CAMBIO	ITINERARIO	FERIADO	DESPACHAR
MAPA	COTACAO	ROTA	CHECKIN	trip
PASSAPORTE	CLASSE	IMPREVISTO	CHECKOUT	

Tabela 6 - Palavras-chave e reservadas em TripCode

### 3 Gramática

A análise sintática de uma linguagem de programação é feita a partir de uma gramática livre de contexto (GLC), que é o nível de poder computacional dos autômatos de pilha. Uma gramática é composta por terminais e por variáveis. Os terminais são símbolos gerados pela gramática que pertencem à linguagem gerada por ela e por convenção são representados em negrito, enquanto as variáveis são símbolos auxiliares que por convenção são representadas em itálico. O analisador léxico é responsável por agrupar o texto de entrada em lexemas e produzir uma saída composta de tokens, em que cada token possui um nome e um valor de atributo. Os nomes dos tokens são símbolos abstratos que vão ser usados pelo analisador sintático e por isso são frequentemente chamados de terminais, porque aparecem como terminais na gramática. No entanto, como o analisador léxico da linguagem `TripCode`, por enquanto, produzirá apenas ações de impressão, ou seja, ainda não retornará tokens, a gramática será especificada considerando o código fonte da linguagem e por isso os terminais serão os possíveis padrões de caracteres.

De modo geral, ao definir a gramática livre de contexto de uma linguagem, definimos as produções gramaticais que especificam as palavras que a linguagem é capaz de gerar, ou seja, especifica em qual ordem os tokens encontrados pelo analisador léxico podem aparecer em um código da linguagem `TripCode`.

#### 3.1 Produções da Gramática

<i>P</i>	→	<i>consts</i> <i>variaveis</i> <i>decl_structs</i> <i>functions_header</i> <i>main</i> <i>functions</i>
<i>main</i>	→	<b>ROTEIRO</b> <i>trip</i> ( <i>list_params_form</i> ) >>> <i>stmts</i> <<< <b>MILHAS</b> <<<
<i>consts</i>		<i>const</i> <i>consts</i>   $\epsilon$
<i>const</i>	→	<b>EXTERIOR</b> <i>id</i> <i>term</i>
<i>decl_structs</i>		<i>decl_struct</i> <i>decl_structs</i>   $\epsilon$
<i>decl_struct</i>	→	<b>PASSAPORTE</b> <i>id</i> >>> <i>variaveis</i> <<<
<i>functions_header</i>		<i>function_header</i> <i>functions_header</i>   $\epsilon$
<i>function_header</i>	→	<b>ROTEIRO</b> <i>id</i> ( <i>list_params_form</i> ) >>> <i>type</i> <<<
<i>functions</i>	→	<i>function</i> <i>functions</i>   $\epsilon$
<i>function</i>	→	<b>ROTEIRO</b> <i>id</i> ( <i>list_params_form</i> ) >>> <i>stmts</i> <<< <i>type</i> <<<
<i>call_function</i>	→	<b>EMBARCAR</b> <i>id</i> ( <i>list_params_real</i> )
<i>return</i>	→	<b>DESPACHAR</b> <i>expr</i> ;
<i>list_params_form</i>	→	<i>param_form</i> <i>params_form</i>   $\epsilon$
<i>params_form</i>	→	, <i>param_form</i> <i>params_form</i>   $\epsilon$
<i>param_form</i>	→	<i>type</i> <i>id</i>
<i>list_params_real</i>	→	<i>param_real</i> <i>params_real</i>   $\epsilon$
<i>params_real</i>	→	, <i>param_real</i> <i>params_real</i>   $\epsilon$
<i>param_real</i>	→	<i>expr</i>
<i>expr</i>	→	<i>expr</i> <i>operator</i> <i>term</i>
		<i>term</i>
		<i>call_function</i>
		<i>increment</i>
		<i>logic_uni</i> <i>expr</i>
		( <i>expr</i> )
<i>stmts</i>	→	<i>stmt</i> <i>stmts</i>   $\epsilon$
<i>stmt</i>	→	<i>for</i>   <i>while</i>   <i>do_while</i>   <i>if</i>   <i>else</i>   <i>switch</i>   <i>command</i>
<i>for</i>	→	<b>DECOLAR</b> ( <b>ORIGEM</b> <i>term</i> , <b>DESTINO</b> <i>term</i> , <b>ESCALA</b> <i>termo</i> ) >>> <i>stmts</i> <<<
<i>while</i>	→	<b>TURISTANDO</b> ( <i>expr</i> ) >>> <i>stmts</i> <<<
<i>do_while</i>	→	<b>TURISTAR</b> >>> <i>stmts</i> <<< <b>DURANTE</b> ( <i>expr</i> );
<i>if</i>	→	<b>ALFANDEGA</b> ( <i>expr</i> ) >>> <i>stmts</i> <<< <i>else</i>

else	→	<<< ISENTO >>> stmts <<<
		<<< TRIBUTADO >>> stmts <<< else
		ε
switch	→	ITINERARIO (variavel) >>> cases <<<
cases		case cases
case		ROTA term: stmts   IMPREVISTO: stmts
command	→	variaveis
		assign
		call_function
		return
		CHECKIN([type_list], id_list);
		CHECKOUT(result_form);
		CAMBIO(type,id);
		COTACAO(type);
		CLASSE(variavel,id);
		POUSAR;
		FERIADO;
variaveis	→	decl_variavel variaveis   def_variavel variaveis   ε
decl_variavel	→	BAGAGEM type id_list;
		MAPA id[type, term];
		MAPA id[type, term, term];
		PASSAPORTE id id;
def_variavel	→	BAGAGEM primary_type id_list <-> expr;
assign	→	variavel_list <-> expr;
variavel_list		variavel vars
vars		,variavel vars   ε
variavel	→	id   array_use   struct_use   ORIGEM
increment	→	variavel TRABALHAR   variavel DESCANSAR
type_list	→	type types
types	→	,type   ε
id_list	→	id ids
ids	→	,id ids   ε
result_form	→	result results
results	→	+ result   ε
result	→	string   type id
term	→	number   float   string   bool   variavel
array_use	→	id[expr]   id[expr,expr]
struct_use	→	id.variavel
type	→	primary_type   MAPA type   PASSAPORTE
primary_type	→	STATUS   MILHAS   DOLAR   VOUCHER
id	→	[a-z][_ {letters}{digit}]*
number	→	signal digits
signal	→	+   -   ε
float	→	number.digits
string	→	"letters"   'letters'
bool		FERIAS   DIAUTIL
operator	→	<->   op   relop   logic_bi
op	→	+   -   *   /   %
relop	→	=   <   <=   >   >=   #
logic_bi	→	AND   OR
logic_uni	→	NOT
digits	→	digit digits   ε
digit	→	0   1   2   3   4   5   6   7   8   9
letters	→	letter letters   ε
letter	→	[a-z] [A-Z]

### 3.2 Exemplo de Derivação

Para verificar se as produções da gramática realmente funcionam como pensado é interessante fazer algumas derivações. Nesse sentido, abaixo temos a derivação para o código da figura 27, em que, iniciando da variável de partida, vamos aplicando produções com objetivo de alcançar um momento apenas com terminais. Para facilitar a visualização do processo, a variável à qual foi aplicada a produção seguinte está sublinhada.

*P => consts variaveis decl\_structs functions\_header main functions*

A variável de partida deriva em cinco variáveis. Para facilitar a visualização do processo, vamos derivar uma de cada vez.

```
consts
=> const consts
=> const
=> EXTERIOR id term
=> EXTERIOR id number
=> EXTERIOR id signal_digits
=> EXTERIOR id digits
=> EXTERIOR id digit digits
=> EXTERIOR id digit digit digits
=> EXTERIOR id digit digit
=> EXTERIOR id 50
=> EXTERIOR max 50

variaveis
=> def_variavel variaveis
=> def_variavel
=> BAGAGEM primary_type id_list <-> expr;
=> BAGAGEM primary_type id_list <-> term;
=> BAGAGEM primary_type id_list <-> bool;
=> BAGAGEM primary_type id ids <-> bool;
=> BAGAGEM primary_type id <-> bool;
=> BAGAGEM STATUS teste <-> FERIAS;

decl_struct
=> PASSAPORTE id >>> variaveis <<<
=> PASSAPORTE id >>> decl_variavel variaveis <<<
=> PASSAPORTE id >>> decl_variavel decl_variavel variaveis <<<
=> PASSAPORTE id >>> decl_variavel decl_variavel decl_variavel variaveis <<<
=> PASSAPORTE id >>> decl_variavel decl_variavel decl_variavel <<<
=> PASSAPORTE id >>> BAGAGEM type id_list; BAGAGEM type id_list; decl_variavel <<<
=> PASSAPORTE id >>> BAGAGEM type id_list; BAGAGEM type id_list; MAPA id[type, term]; <<<
=> PASSAPORTE id >>> BAGAGEM primary_type id_list; BAGAGEM primary_type id_list; MAPA id[primary_type, term]; <<<
=> PASSAPORTE id >>> BAGAGEM VOUCHER id_list; BAGAGEM MILHAS id_list; MAPA id[DOLAR, term]; <<<
=> PASSAPORTE id >>> BAGAGEM VOUCHER id ids; BAGAGEM MILHAS id ids; MAPA id[DOLAR, term]; <<<
=> PASSAPORTE id >>> BAGAGEM VOUCHER id; BAGAGEM MILHAS id; MAPA id[DOLAR, term]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, term]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, number]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, signal_digits]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, digits]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, digit digits]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, digit]; <<<
=> PASSAPORTE aluno >>> BAGAGEM VOUCHER nome; BAGAGEM MILHAS matricula; MAPA notas[DOLAR, 4]; <<<

functions_header
=> ROTEIRO id (list_params_form) >>> type <<<
=> ROTEIRO id (param_form params_fomr) >>> type <<<
=> ROTEIRO id (param_form,param_form params_fomr) >>> type <<<
=> ROTEIRO id (param_form,param_form) >>> type <<<
=> ROTEIRO id (type id, type id) >>> type <<<
=> ROTEIRO id (primary_type id, primary_type id) >>> primary_type <<<
=> ROTEIRO id (MILHAS id, MILHAS id) >>> MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> MILHAS <<<
```



[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

```

ROTEIRO trip() >>> BAGAGEM type id id; BAGAGEM primary_type id id id <-> number; BAGAGEM primary_type id <-> float; MAPA
id[type, number, number]; id[number,number] <-> string; PASSAPORTE id id; id.id <-> string; id.id <-> number; id <->
id.id; BAGAGEM primary_type id <-> logic_uni id; BAGAGEM primary_type id <-> id operator id; id <-> id operator id;
BAGAGEM primary_type id <-> id operator id; id <-> id TRABALHAR; id <-> EMBARCAR id(id, number); DECOLAR (ORIGEM number,
DESTINO number, ESCALA number) >>> BAGAGEM type id; CHECKIN([type], id); id.id[ORIGEM] <-> id; <<< BAGAGEM primary_type
id <-> id.id[number]; ITINERARIO (id) >>> ROTA number: CHECKOUT(string); POUSAR; IMPREVISTO: CHECKOUT(string); POUSAR;
<<< DESPACHAR number; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM primary_type id id; BAGAGEM primary_type id id id <-> number; BAGAGEM primary_type id <->
float; MAPA id[primary_type, number, number]; id[number,number] <-> string; PASSAPORTE id id; id.id <-> string; id.id <->
number; id <-> id.id; BAGAGEM primary_type id <-> logic_uni id; BAGAGEM primary_type id <-> id operator id; id <-> id
operator id; BAGAGEM primary_type id <-> id operator id; id <-> id TRABALHAR; id <-> EMBARCAR id(id, number); DECOLAR
(ORIGEM number, DESTINO number, ESCALA number) >>> BAGAGEM primary_type id; CHECKIN([primary_type], id); id.id[ORIGEM]
<-> id; <<< BAGAGEM primary_type id <-> id.id[number]; ITINERARIO (id) >>> ROTA number: CHECKOUT(string); POUSAR;
IMPREVISTO: CHECKOUT(string); POUSAR; <<< DESPACHAR number; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM primary_type id id; BAGAGEM primary_type id id id <-> number; BAGAGEM primary_type id <->
float; MAPA id[primary_type, number, number]; id[number,number] <-> string; PASSAPORTE id id; id.id <-> string; id.id <->
number; id <-> id.id; BAGAGEM primary_type id <-> logic_uni id; BAGAGEM primary_type id <-> id logic_bi id; id <-> id
logic_bi id; BAGAGEM primary_type id <-> id logic_bi id; id <-> id TRABALHAR; id <-> EMBARCAR id(id, number); DECOLAR
(ORIGEM number, DESTINO number, ESCALA number) >>> BAGAGEM primary_type id; CHECKIN([primary_type], id); id.id[ORIGEM]
<-> id; <<< BAGAGEM primary_type id <-> id.id[number]; ITINERARIO (id) >>> ROTA number: CHECKOUT(string); POUSAR;
IMPREVISTO: CHECKOUT(string); POUSAR; <<< DESPACHAR number; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM primary_type id id; BAGAGEM primary_type id id id <-> 5; BAGAGEM primary_type id <-> 5.25;
MAPA id[primary_type, 10, 6]; id[0,0] <-> string; PASSAPORTE id id; id.id <-> string; id.id <-> 4661; id <-> id.id;
BAGAGEM primary_type id <-> logic_uni id; BAGAGEM primary_type id <-> id logic_bi id; id <-> id logic_bi id; BAGAGEM
primary_type id <-> id logic_bi id; id <-> id TRABALHAR; id <-> EMBARCAR id(id, 10); DECOLAR (ORIGEM 0, DESTINO 3, ESCALA
1) >>> BAGAGEM primary_type id; CHECKIN([primary_type], id); id.id[ORIGEM] <-> id; <<< BAGAGEM primary_type id <->
id.id[0]; ITINERARIO (id) >>> ROTA 100: CHECKOUT(string); POUSAR; IMPREVISTO: CHECKOUT(string); POUSAR; <<< DESPACHAR 0;
<<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM primary_type id id; BAGAGEM primary_type id id id <-> 5; BAGAGEM primary_type id <-> 5.25;
MAPA id[primary_type, 10, 6]; id[0,0] <-> "Materia 1"; PASSAPORTE id id; id.id <-> "Leticia"; id.id <-> 4661; id <->
id.id; BAGAGEM primary_type id <-> logic_uni id; BAGAGEM primary_type id <-> id logic_bi id; id <-> id logic_bi id;
BAGAGEM primary_type id <-> id logic_bi id; id <-> id TRABALHAR; id <-> EMBARCAR id(id, 10); DECOLAR (ORIGEM 0, DESTINO
3, ESCALA 1) >>> BAGAGEM primary_type id; CHECKIN([primary_type], id); id.id[ORIGEM] <-> id; <<< BAGAGEM primary_type id
<-> id.id[0]; ITINERARIO (id) >>> ROTA 100: CHECKOUT("Parabéns!"); POUSAR; IMPREVISTO: CHECKOUT("Você é capaz de
melhorar"); POUSAR; <<< DESPACHAR 0; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM VOUCHER id id; BAGAGEM MILHAS id id id <-> 5; BAGAGEM DOLAR id <-> 5.25; MAPA id[VOUCHER, 10,
6]; id[0,0] <-> "Materia 1"; PASSAPORTE id id; id.id <-> "Leticia"; id.id <-> 4661; id <-> id.id; BAGAGEM STATUS id <->
logic_uni id; BAGAGEM STATUS id <-> id logic_bi id; id <-> id logic_bi id; BAGAGEM STATUS id <-> id logic_bi id;
id <-> id TRABALHAR; id <-> EMBARCAR id(id, 10); DECOLAR (ORIGEM 0, DESTINO 3, ESCALA 1) >>> BAGAGEM DOLAR id;
CHECKIN([DOLAR], id); id.id[ORIGEM] <-> id; <<< BAGAGEM DOLAR id <-> id.id[0]; ITINERARIO (id) >>> ROTA 100:
CHECKOUT("Parabéns!"); POUSAR; IMPREVISTO: CHECKOUT("Você é capaz de melhorar"); POUSAR; <<< DESPACHAR 0; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM VOUCHER id, id; BAGAGEM MILHAS id,id, id <-> 5; BAGAGEM DOLAR id <-> 5.25; MAPA id[VOUCHER,
10, 6]; id[0,0] <-> "Materia 1"; PASSAPORTE id id; id,id <-> "Leticia"; id,id <-> 4661; id <-> id,id; BAGAGEM STATUS id
<-> NOT id; BAGAGEM STATUS id <-> id AND id; id <-> id AND id; BAGAGEM STATUS id <-> id OR id; id <-> id TRABALHAR; id
<-> EMBARCAR id(id, 10); DECOLAR (ORIGEM 0, DESTINO 3, ESCALA 1) >>> BAGAGEM DOLAR id; CHECKIN([DOLAR], id);
id,id[ORIGEM] <-> id; <<< BAGAGEM DOLAR id <-> id,id[0]; ITINERARIO (id) >>> ROTA 100: CHECKOUT("Parabéns!"); POUSAR;
IMPREVISTO: CHECKOUT("Você é capaz de melhorar"); POUSAR; <<< DESPACHAR 0; <<< MILHAS <<<

ROTEIRO trip() >>> BAGAGEM VOUCHER nome1,nome2; BAGAGEM MILHAS x ,y,z <-> 5; BAGAGEM DOLAR media <-> 5.25; MAPA
horarios[VOUCHER, 10, 6]; horarios[0,0] <-> "Materia 1"; PASSAPORTE aluno aluno1; aluno1_nome <-> "Leticia";
aluno1_matricula <-> 4661; y <-> aluno1_matricula; BAGAGEM STATUS negação <-> NOT teste; BAGAGEM STATUS logic_and <->
negacao AND teste; logic_and <-> logic_and AND negacao; BAGAGEM STATUS logic_not <-> negacao OR teste; x <-> x
TRABALHAR; z <-> EMBARCAR soma(z, 10); DECOLAR (ORIGEM 0, DESTINO 3, ESCALA 1) >>> BAGAGEM DOLAR nota; CHECKIN([DOLAR],
nota); aluno1.nota[ORIGEM] <-> nota; <<< BAGAGEM DOLAR nota <-> aluno1.nota[0]; ITINERARIO (nota) >>> ROTA 100:
CHECKOUT("Parabéns!"); POUSAR; IMPREVISTO: CHECKOUT("Você é capaz de melhorar"); POUSAR; <<< DESPACHAR 0; <<< MILHAS <<<

```

## functions

```

=> function functions
=> function
=> ROTEIRO id (list_params_form) >>> stmts <<< type <<<
=> ROTEIRO id (param_form params_form) >>> stmts <<< type <<<
=> ROTEIRO id (param_form ,param_form params_fomr) >>> stmts <<< type <<<
=> ROTEIRO id (param_form ,param_form) >>> stmts <<< type <<<
=> ROTEIRO id (type id,type id) >>> stmts <<< type <<<
=> ROTEIRO id (type id,type id) >>> stmts <<< type <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> stmt stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> stmt stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> command stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> variaveis stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> def_variavel stmts <<< MILHAS <<<

```

```

=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id_list <-> expr; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id_list <-> expr operator term; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id_list <-> term operator term; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id_list <-> variavel operator variavel; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id_list <-> id operator id; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id <-> id operator id; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id <-> id operator id; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM primary_type id <-> id op id; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; stmt stmts <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; stmt <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; command <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; return <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; DESPACHAR expr; <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; DESPACHAR term; <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; DESPACHAR variavel; <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; DESPACHAR id; <<< MILHAS <<<
=> ROTEIRO soma (MILHAS a, MILHAS b) >>> BAGAGEM MILHAS s <-> a + b; DESPACHAR s; <<< MILHAS <<<

```

## 4 Analisador Léxico

### 4.1 Definições Regulares

Para definir as expressões regulares da linguagem, inicialmente foram estabelecidos os tokens que o analisador léxico deveria retornar. Consideraram-se os comandos existentes na linguagem e a estrutura do código, definindo os tokens relevantes e os caracteres a serem ignorados, como espaços em branco, quebras de linha e tabulações. Os demais tokens foram definidos com base nos comandos disponíveis apresentados na *Seção 4* desta documentação. As definições e expressões regulares para a linguagem *TripCode* são as seguintes:

delim	[ \t\n]
ws	{delim}+
digit	[0-9]
lower_letter	[a-z]
upper_letter	[A-Z]
letter	{lower_letter} {upper_letter}
comment	->[^<]*<-
trip	trip
ferias	FERIAS
diautil	DIAUTIL
status	STATUS
milhas	MILHAS
dolar	DOLAR
voucher	VOUCHER
descansar	DESCANSAR
trabalhar	TRABALHAR
cambio	CAMBIO
cotacao	COTACAO
classe	CLASSE
mapa	MAPA
passaporte	PASSAPORTE
bagagem	BAGAGEM
exterior	EXTERIOR
alfandega	ALFANDEGA
tributado	TRIBUTADO
isento	ISENTO
itinerario	ITINERARIO
rota	ROTA
imprevisto	IMPREVISTO
pousar	POUSAR

feriado	FERIADO
decolar	DECOLAR
origem	ORIGEM
destino	DESTINO
escala	ESCALA
turistando	TURISTANDO
turistar	TURISTAR
durante	DURANTE
checkin	CHECKIN
checkout	CHECKOUT
roteiro	ROTEIRO
embarcar	EMBARCAR
despachar	DESPACHAR
and	AND
or	OR
not	NOT
logicop	{and} {or} {not}
atribuicao	\<\->
som	\+
sub	\-
mult	\*
div	\/
mod	\%
op	{som} {sub} {mult} {div} {mod}
eq	\=
gt	\>
ge	\>\=
lt	\<
le	\<\=
diff	\#
relop	{eq} {gt} {ge} {lt} {le} {diff}
inteiro_pos	\+?{digit}+
inteiro_neg	\-{digit}+
inteiro	{inteiro_pos} {inteiro_neg}
float	{inteiro}+\.{digit}+
string	"([^\\""] \\.)*\\"' '([^\\"'] \\.)*\'
id	((lower_letter))(_ {letter} {digit})*
dot	\.
comma	\,
dot_comma	\;
colon	\:
open_bracket	\[
close_bracket	\]
open_parentheses	\(
close_parentheses	\)
open_codeblock	\>\>\>
close_codeblock	\<\<\<

## 4.2 Arquivo lex.l

Para implementar o analisador léxico, é importante observar a ordem em que cada padrão é definido ao definir as regras de tradução. Como as palavras-chaves são reservadas, elas devem aparecer após o padrão de comentários e antes do padrão para identificadores. Para os demais padrões, o primeiro critério de desempate de preferir o prefixo mais longo já é suficiente.

## 5 Testes de Execução

Para testar a corretude do funcionamento do analisador léxico desenvolvido, foram criados arquivos de código fonte da linguagem especificada neste trabalho, os quais foram utilizados como entrada para o analisador léxico. Desta forma, esta seção apresenta os arquivos utilizados como entrada e suas respectivas saídas. É importante destacar que, nesta primeira parte do trabalho o analisador léxico foi projetado para ser autônomo, e por isso as saídas exibem uma mensagem na tela indicando qual lexema foi identificado. A seguir estão apresentados casos de teste e suas respectivas saídas para cada comando presentes na linguagem `TripCode`.

### 5.1 Repetição

```
ROTEIRO trip()>>>

  BAGAGEM DOLAR dolar;
  BAGAGEM DOLAR distancia;

  distancia <-> 0;

  DECOLAR (ORIGEM 5, DESTINO 20, ESCALA 2) >>>
    distancia <-> distancia + 5;
  <<<

  dolar <-> 6;

  TURISTANDO (FERIAS) >>>
    ALFANDEGA(dolar = 6) >>>
      POUSAR;
    <<<

    distancia <-> distancia + 5;
    dolar <-> DESCANSAR dolar;
  <<<

  TURISTAR >>>
    distancia <-> distancia + 5;
  <<< DURANTE (dolar < 6);

  DESPACHAR 0;
<<< MILHAS <<<
```

Figura 18 - Arquivo de teste de comandos de repetição

Unset

```
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado a função principal trip. LEXEMA: trip
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando decolar. LEXEMA: DECOLAR
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrada o comando origem. LEXEMA: ORIGEM
```

```

Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrada o comando destino. LEXEMA: DESTINO
Foi encontrado um inteiro. LEXEMA: 20
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrada o comando escala. LEXEMA: ESCALA
Foi encontrado um inteiro. LEXEMA: 2
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrada a operação de atribuição. LEXEMA: <=>
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrado um operador aritmético. LEXEMA: +
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrada a operação de atribuição. LEXEMA: <=>
Foi encontrado um inteiro. LEXEMA: 6
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando turistando. LEXEMA: TURISTANDO
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um valor booleano ferias. LEXEMA: FERIAS
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrado o comando alfandega. LEXEMA: ALFANDEGA
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrado um operador relacional. LEXEMA: =
Foi encontrado um inteiro. LEXEMA: 6
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada o comando pousar. LEXEMA: POUSAR
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrada a operação de atribuição. LEXEMA: <=>
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrado um operador aritmético. LEXEMA: +
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrada a operação de atribuição. LEXEMA: <=>
Foi encontrada o comando descansar. LEXEMA: DESCANSAR
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrada o comando turistar. LEXEMA: TURISTAR
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrada a operação de atribuição. LEXEMA: <=>
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrado um operador aritmético. LEXEMA: +
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrada o comando durante. LEXEMA: DURANTE
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: dolar
Foi encontrado um operador relacional. LEXEMA: <
Foi encontrado um inteiro. LEXEMA: 6
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<

```

**Figura 19 - Saída de arquivo de teste de comandos de repetição**



## 5.2 Condicionais

```
-> este e um comentario <-

-> este e
outro comentario <-

ROTEIRO trip() >>>
    BAGAGEM MILHAS x <-> 5;

    ALFANDEGA (x = 8) >>>
        CHECKOUT('oi');
    <<< ISENTO >>>
        CHECKOUT('tchau');

    ALFANDEGA (x <= 4) >>>
        CHECKOUT('haha');
    <<< TRIBUTADO (x = 5) >>>
        CHECKOUT('bye');
        BAGAGEM soma <-> 5.1 + 3.26
    <<< ISENTO >>>
        CHECKOUT('hihi');
    <<<

    BAGAGEM DOLAR distancia <-> 45;

    ITINERARIO(distancia) >>>
        ROTA 90:
            CHECKOUT("Você chegou ao seu destino");
            POUSAR;
        ROTA 45:
            CHECKOUT("Você está exatemennte na metade do caminho");
            POUSAR;
        IMPREVISTO:
            CHECKOUT("Você está apenas no início do caminho");
            POUSAR;

    <<<

    DESPACHAR 0;

<<< MILHAS <<<
```

Figura 20 - Arquivo de teste de comandos condicionais

### Unset

```
Foi encontrado um comentario: -> este e um comentario <-
Foi encontrado um comentario: -> este e
outro comentario <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado a função principal trip. LEXEMA: trip
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado o comando alfanDEGA. LEXEMA: ALFANDEGA
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: x
Foi encontrado um operdador relacional. LEXEMA: =
Foi encontrado um inteiro. LEXEMA: 8
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: 'oi'
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado o comando isento. LEXEMA: ISENTO
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a função checkout. LEXEMA: CHECKOUT
```

```

Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: 'tchau'
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado o comando alfandega. LEXEMA: ALFANDEGA
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: x
Foi encontrado um operador relacional. LEXEMA: <=
Foi encontrado um inteiro. LEXEMA: 4
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: 'haha'
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado o comando tributado. LEXEMA: TRIBUTADO
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: x
Foi encontrado um operador relacional. LEXEMA: =
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: 'bye'
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um identificador. LEXEMA: soma
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um float. LEXEMA: 5.1
Foi encontrado um operador aritmético. LEXEMA: +
Foi encontrado um float. LEXEMA: 3.26
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado o comando isento. LEXEMA: ISENTO
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: 'hihi'
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um inteiro. LEXEMA: 45
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando itinerario. LEXEMA: ITINERARIO
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: distancia
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada o comando rota. LEXEMA: ROTA
Foi encontrado um inteiro. LEXEMA: 90
Foi encontrado dois pontos. LEXEMA: :
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: "Você chegou ao seu destino"
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando pousar. LEXEMA: POUSAR
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando rota. LEXEMA: ROTA
Foi encontrado um inteiro. LEXEMA: 45
Foi encontrado dois pontos. LEXEMA: :
Foi encontrada a função checkout. LEXEMA: CHECKOUT
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: "Você está exatemennte na metade do caminho"
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando pousar. LEXEMA: POUSAR
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando imprevisto. LEXEMA: IMPREVISTO
Foi encontrado dois pontos. LEXEMA: :
Foi encontrada a função checkout. LEXEMA: CHECKOUT

```

```

Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado uma string. LEXEMA: "Você está apenas no início do caminho"
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando pousar. LEXEMA: POUSAR
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<

```

Figura 21 - Saída do arquivo de teste de comandos condicionais

### 5.3 Funções

```

-> protótipo <-
ROTEIRO minha_funcao (MILHAS x, DOLAR y) >>> DOLAR <<<

ROTEIRO trip() >>>

    BAGAGEM DOLAR total;

-> ativar/chamar <-
total <-> EMBARCAR minha_funcao (MILHAS x, DOLAR y);

DESPACHAR 0;
<<< MILHAS <<<

-> definição <-
ROTEIRO minha_funcao (MILHAS x, DOLAR y) >>>
    BAGAGEM DOLAR result;
    result <-> x * y;

-> quaisquer outras operações ou comandos <-

DESPACHAR result;
<<< DOLAR <<<

```

Figura 22 - Arquivo de teste de funções

Unset

```

Foi encontrado um comentario: -> protótipo <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado um identificador. LEXEMA: minha_funcao
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: y
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado a função principal trip. LEXEMA: trip
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: total
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um comentario: -> ativar/chamar <-

```

```

Foi encontrado um identificador. LEXEMA: total
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrada o comando embarcar. LEXEMA: EMBARCAR
Foi encontrado um identificador. LEXEMA: minha_funcao
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: y
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um comentario: -> definição <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado um identificador. LEXEMA: minha_funcao
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: y
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a estrutura de dados bagagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: result
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: result
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um identificador. LEXEMA: x
Foi encontrado um operador aritmético. LEXEMA: *
Foi encontrado um identificador. LEXEMA: y
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um comentario: -> quaisquer outras operações ou comandos <-
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um identificador. LEXEMA: result
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<

```

**Figura 23 - Saída do arquivo de teste de funções**

## 5.4 Estruturas de dados

```
PASSAPORTE minha_struct >>>
  BAGAGEM DOLAR ricas2;
  -> pode ter quantas quiser <-
<<<

ROTEIRO trip() >>>

  PASSAPORTE minha_struct teste;
  teste.ricas2 <-> 1000.50;

  MAPA notas [DOLAR, 10];

  MAPA matriz [MILHAS, 3, 3];

  notas[1] <-> 6;

  BAGAGEM MILHAS x, y, z <-> 5;

  BAGAGEM VOUCHER texto <-> "oi";

  BAGAGEM VOUCHER outro_texto <-> 'tudo?';

  CAMBIO(DOLAR, x);
  COTACAO(VOUCHER);
  CLASSE(y);

  DESPACHAR 0;
<<< MILHAS <<<
```

Figura 24 - Arquivo de teste das estruturas

### Unset

Foi encontrada a estrutura de dados passaporte. LEXEMA: PASSAPORTE  
Foi encontrado um identificador. LEXEMA: minha\_struct  
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>  
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM  
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR  
Foi encontrado um identificador. LEXEMA: ricas2  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrado um comentario: -> pode ter quantas quiser <-  
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<  
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO  
Foi encontrado a função principal trip. LEXEMA: trip  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>  
Foi encontrada a estrutura de dados passaporte. LEXEMA: PASSAPORTE  
Foi encontrado um identificador. LEXEMA: minha\_struct  
Foi encontrado um identificador. LEXEMA: teste  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrado um identificador. LEXEMA: teste  
Foi encontrado um ponto. LEXEMA: .  
Foi encontrado um identificador. LEXEMA: ricas2  
Foi encontrada a operação de atribuição. LEXEMA: <->  
Foi encontrado um float. LEXEMA: 1000.50  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada a estrutura de dados mapa. LEXEMA: MAPA  
Foi encontrado um identificador. LEXEMA: notas  
Foi encontrada uma abertura de colchetes. LEXEMA: [  
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR  
Foi encontrado uma vírgula. LEXEMA: ,  
Foi encontrado um inteiro. LEXEMA: 10  
Foi encontrado um fechamento de colchetes. LEXEMA: ]  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada a estrutura de dados mapa. LEXEMA: MAPA  
Foi encontrado um identificador. LEXEMA: matriz  
Foi encontrada uma abertura de colchetes. LEXEMA: [  
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS

```

Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 3
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 3
Foi encontrado um fechamento de colchetes. LEXEMA: ]
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: notas
Foi encontrada uma abertura de colchetes. LEXEMA: [
Foi encontrado um inteiro. LEXEMA: 1
Foi encontrado um fechamento de colchetes. LEXEMA: ]
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um inteiro. LEXEMA: 6
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: y
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: z
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado um identificador. LEXEMA: texto
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado uma string. LEXEMA: "oi"
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado um identificador. LEXEMA: outro_texto
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado uma string. LEXEMA: 'tudo?'
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um operador cambio. LEXEMA: CAMBIO
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: x
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um operador cotacao. LEXEMA: COTACAO
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um operador classe. LEXEMA: CLASSE
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: y
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<

```

**Figura 25 - Saída do arquivo de teste das estruturas**

## 5.5 Exemplo Geral

```
-> definicao de constantes <-
EXTERIOR max 50

-> declaracao e/ou definicao de variaveis globais <-
BAGAGEM STATUS teste <-> FERIAS;

-> declaracao de estruturas <-
PASSAPORTE aluno >>>
    BAGAGEM VOUCHER nome;
    BAGAGEM MILHAS matricula;
    MAPA notas[DOLAR, 4];
<<<

-> prototipos de funcoes <-
ROTEIRO soma (MILHAS a, MILHAS b) >>> MILHAS <<<

-> funcao principal <-
ROTEIRO trip()>>>

    BAGAGEM VOUCHER nome1, nome2; -> declaracao multipla <-
    BAGAGEM MILHAS x, y, z <-> 5; -> definicao multipla <-
    BAGAGEM DOLAR media <-> 5.25;

    MAPA horarios[VOUCHER, 10, 6];
    horarios[0,0] <-> "Materia 1";

    PASSAPORTE aluno aluno1;
    aluno1.nome <-> "Leticia";
    aluno1.matricula <-> 4661;
    y <-> aluno1.matricula;

    BAGAGEM STATUS negacao <-> NOT teste;
    BAGAGEM STATUS logic_and <-> negacao AND teste;
    logic_and <-> logic_and AND negacao;
    BAGAGEM STATUS logic_or <-> negacao OR teste;

    x <-> x TRABALHAR;
    z <-> EMBARCAR soma(z, 10);

    DECOLAR(ORIGEM 0, DESTINO 3, ESCALA 1)>>>
        BAGAGEM DOLAR nota;
        CHECKIN([DOLAR], nota);
        aluno1.notas[ORIGEM] <-> nota;
    <<<

    BAGAGEM DOLAR nota <-> aluno1.notas[0];

    ITINERARIO(nota) >>>
        ROTA 100:
            CHECKOUT("Parabéns");
            POUSAR;
        IMPREVISTO:
            CHECKOUT("Você é capaz de melhorar");
            POUSAR;
    <<<

    DESPACHAR 0;

<<< MILHAS <<<

-> funcoes criadas pelo programador <-
ROTEIRO soma (MILHAS a, MILHAS b) >>>
    BAGAGEM MILHAS s <-> a + b;
    DESPACHAR s;
<<< MILHAS <<<
```

Figura 26 - Arquivo de teste geral

## Unset

```
Foi encontrado um comentario: -> definicao de constantes <-
Foi encontrada uma declaracao de constante exterior. LEXEMA: EXTERIOR
Foi encontrado um identificador. LEXEMA: max
Foi encontrado um inteiro. LEXEMA: 50
Foi encontrado um comentario: -> declaracao e/ou definicao de variaveis globais <-
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados status. LEXEMA: STATUS
Foi encontrado um identificador. LEXEMA: teste
Foi encontrada a operacao de atribuicao. LEXEMA: <->
Foi encontrado um valor booleano ferias. LEXEMA: FERIAS
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrado um comentario: -> declaracao de estruturas <-
Foi encontrada a estrutura de dados passaporte. LEXEMA: PASSAPORTE
Foi encontrado um identificador. LEXEMA: aluno
Foi encontrado uma abertura de bloco de codigo. LEXEMA: >>>
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado um identificador. LEXEMA: nome
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: matricula
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrada a estrutura de dados mapa. LEXEMA: MAPA
Foi encontrado um identificador. LEXEMA: notas
Foi encontrada uma abertura de colchetes. LEXEMA: [
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado uma virgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 4
Foi encontrado um fechamento de colchetes. LEXEMA: ]
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de codigo. LEXEMA: <<<
Foi encontrado um comentario: -> prototipos de funcoes <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado um identificador. LEXEMA: soma
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: a
Foi encontrado uma virgula. LEXEMA: ,
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: b
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de codigo. LEXEMA: >>>
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de codigo. LEXEMA: <<<
Foi encontrado um comentario: -> funcao principal <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado a função principal trip. LEXEMA: trip
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de codigo. LEXEMA: >>>
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado um identificador. LEXEMA: nome1
Foi encontrado uma virgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: nome2
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrado um comentario: -> declaracao multipla <-
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: x
Foi encontrado uma virgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: y
Foi encontrado uma virgula. LEXEMA: ,
Foi encontrado um identificador. LEXEMA: z
Foi encontrada a operacao de atribuicao. LEXEMA: <->
Foi encontrado um inteiro. LEXEMA: 5
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrado um comentario: -> definicao multipla <-
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR
Foi encontrado um identificador. LEXEMA: media
Foi encontrada a operacao de atribuicao. LEXEMA: <->
Foi encontrado um float. LEXEMA: 5.25
Foi encontrado um ponto e virgula. LEXEMA: ;
Foi encontrada a estrutura de dados mapa. LEXEMA: MAPA
Foi encontrado um identificador. LEXEMA: horarios
Foi encontrada uma abertura de colchetes. LEXEMA: [
```



```

Foi encontrado tipo de dados voucher. LEXEMA: VOUCHER
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 10
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 6
Foi encontrado um fechamento de colchetes. LEXEMA: ]
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: horarios
Foi encontrada uma abertura de colchetes. LEXEMA: [
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um inteiro. LEXEMA: 0
Foi encontrado um fechamento de colchetes. LEXEMA: ]
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado uma string. LEXEMA: "Materia 1"
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados passaporte. LEXEMA: PASSAPORTE
Foi encontrado um identificador. LEXEMA: aluno
Foi encontrado um identificador. LEXEMA: aluno1
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: aluno1
Foi encontrado um ponto. LEXEMA: .
Foi encontrado um identificador. LEXEMA: nome
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado uma string. LEXEMA: "Leticia"
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: aluno1
Foi encontrado um ponto. LEXEMA: .
Foi encontrado um identificador. LEXEMA: matricula
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um inteiro. LEXEMA: 4661
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: y
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um identificador. LEXEMA: aluno1
Foi encontrado um ponto. LEXEMA: .
Foi encontrado um identificador. LEXEMA: matricula
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados status. LEXEMA: STATUS
Foi encontrado um identificador. LEXEMA: negacao
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um operador lógico. LEXEMA: NOT
Foi encontrado um identificador. LEXEMA: teste
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados status. LEXEMA: STATUS
Foi encontrado um identificador. LEXEMA: logic_and
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um identificador. LEXEMA: negacao
Foi encontrado um operador lógico. LEXEMA: AND
Foi encontrado um identificador. LEXEMA: teste
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: logic_and
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um identificador. LEXEMA: logic_and
Foi encontrado um operador lógico. LEXEMA: AND
Foi encontrado um identificador. LEXEMA: negacao
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada a estrutura de dados babagem. LEXEMA: BAGAGEM
Foi encontrado tipo de dados status. LEXEMA: STATUS
Foi encontrado um identificador. LEXEMA: logic_or
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um identificador. LEXEMA: negacao
Foi encontrado um operador lógico. LEXEMA: OR
Foi encontrado um identificador. LEXEMA: teste
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: x
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrado um identificador. LEXEMA: x
Foi encontrada o comando trabalhar. LEXEMA: TRABALHAR
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um identificador. LEXEMA: z
Foi encontrada a operação de atribuição. LEXEMA: <-->
Foi encontrada o comando embarcar. LEXEMA: EMBARCAR
Foi encontrado um identificador. LEXEMA: soma
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um identificador. LEXEMA: z

```

Foi encontrado uma vírgula. LEXEMA: ,  
Foi encontrado um inteiro. LEXEMA: 10  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada o comando decolar. LEXEMA: DECOLAR  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrada o comando origem. LEXEMA: ORIGEM  
Foi encontrado um inteiro. LEXEMA: 0  
Foi encontrado uma vírgula. LEXEMA: ,  
Foi encontrada o comando destino. LEXEMA: DESTINO  
Foi encontrado um inteiro. LEXEMA: 3  
Foi encontrado uma vírgula. LEXEMA: ,  
Foi encontrada o comando escala. LEXEMA: ESCALA  
Foi encontrado um inteiro. LEXEMA: 1  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>  
Foi encontrada a estrutura de dados bagagem. LEXEMA: BAGAGEM  
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR  
Foi encontrado um identificador. LEXEMA: nota  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada a função checkin. LEXEMA: CHECKIN  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrada uma abertura de colchetes. LEXEMA: [  
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR  
Foi encontrado um fechamento de colchetes. LEXEMA: ]  
Foi encontrado uma vírgula. LEXEMA: ,  
Foi encontrado um identificador. LEXEMA: nota  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrado um identificador. LEXEMA: aluno1  
Foi encontrado um ponto. LEXEMA: .  
Foi encontrado um identificador. LEXEMA: notas  
Foi encontrada uma abertura de colchetes. LEXEMA: [  
Foi encontrada o comando origem. LEXEMA: ORIGEM  
Foi encontrado um fechamento de colchetes. LEXEMA: ]  
Foi encontrada a operação de atribuição. LEXEMA: <->  
Foi encontrado um identificador. LEXEMA: nota  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<  
Foi encontrada a estrutura de dados bagagem. LEXEMA: BAGAGEM  
Foi encontrado um tipo de dados dolar. LEXEMA: DOLAR  
Foi encontrado um identificador. LEXEMA: nota  
Foi encontrada a operação de atribuição. LEXEMA: <->  
Foi encontrado um identificador. LEXEMA: aluno1  
Foi encontrado um ponto. LEXEMA: .  
Foi encontrado um identificador. LEXEMA: notas  
Foi encontrada uma abertura de colchetes. LEXEMA: [  
Foi encontrado um inteiro. LEXEMA: 0  
Foi encontrado um fechamento de colchetes. LEXEMA: ]  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada o comando itinerario. LEXEMA: ITINERARIO  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrado um identificador. LEXEMA: nota  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>  
Foi encontrada o comando rota. LEXEMA: ROTA  
Foi encontrado um inteiro. LEXEMA: 100  
Foi encontrado dois pontos. LEXEMA: :  
Foi encontrada a função checkout. LEXEMA: CHECKOUT  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrado uma string. LEXEMA: "Parabéns"  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada o comando pousar. LEXEMA: POUSAR  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada o comando imprevisto. LEXEMA: IMPREVISTO  
Foi encontrado dois pontos. LEXEMA: :  
Foi encontrada a função checkout. LEXEMA: CHECKOUT  
Foi encontrada uma abertura de parênteses. LEXEMA: (  
Foi encontrado uma string. LEXEMA: "Você é capaz de melhorar"  
Foi encontrado um fechamento de parênteses. LEXEMA: )  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrada o comando pousar. LEXEMA: POUSAR  
Foi encontrado um ponto e vírgula. LEXEMA: ;  
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<  
Foi encontrada o comando despachar. LEXEMA: DESPACHAR  
Foi encontrado um inteiro. LEXEMA: 0  
Foi encontrado um ponto e vírgula. LEXEMA: ;

```
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um comentario: -> funcoes criadas pelo programador <-
Foi encontrada o comando roteiro. LEXEMA: ROTEIRO
Foi encontrado um identificador. LEXEMA: soma
Foi encontrada uma abertura de parênteses. LEXEMA: (
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: a
Foi encontrado uma vírgula. LEXEMA: ,
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: b
Foi encontrado um fechamento de parênteses. LEXEMA: )
Foi encontrado uma abertura de bloco de código. LEXEMA: >>>
Foi encontrada a estrutura de dados bagagem. LEXEMA: BAGAGEM
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um identificador. LEXEMA: s
Foi encontrada a operação de atribuição. LEXEMA: <->
Foi encontrado um identificador. LEXEMA: a
Foi encontrado um operador aritmético. LEXEMA: +
Foi encontrado um identificador. LEXEMA: b
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrada o comando despachar. LEXEMA: DESPACHAR
Foi encontrado um identificador. LEXEMA: s
Foi encontrado um ponto e vírgula. LEXEMA: ;
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
Foi encontrado um tipo de dados milhas. LEXEMA: MILHAS
Foi encontrado um fechamento de bloco de código. LEXEMA: <<<
```

**Figura 27 - Saída do arquivo de teste geral**

## 6 Referências

[1] AHO, Alfred V. et al. Compiladores: Princípios, técnicas e ferramentas. 2. ed. São Paulo: Pearson Addison Wesley, 2008. 638 p. Tradução de Daniel Vieira.

[2] Manual de Sintaxe da Linguagem C. Disponível em:

<https://www.feg.unesp.br/Home/PaginasPessoais/profmarcosapereira3168/programacaodecomputadoresi/manual-de-sintaxe-da-linguagem-c.pdf>

[3] Analisador Léxico Desenvolvido. Disponível em:

<https://github.com/ingredalmeida1/tripcode>