

# Construção de um compilador de Lua para Dalvik usando Objective Caml

Vitor Martins Basso

`vitorbasso93@gmail.com`

Faculdade de Computação  
Universidade Federal de Uberlândia

30 de outubro de 2018

# Lista de Listagens

2.1	enunciado do programa micro04 - Lê números e informa quais estão entre 10 e 150 . . . . .	17
2.2	micro04.java - Lê números e informa quais estão entre 10 e 150 . . . . .	17
2.3	micro04.smali - Lê números e informa quais estão entre 10 e 150 . . . . .	18
3.1	nano01.lua - Módulo mínimo que caracteriza um programa . . . . .	20
3.2	nano01.java - Módulo mínimo que caracteriza um programa . . . . .	20
3.3	nano01.smali - Módulo mínimo que caracteriza um programa . . . . .	20
3.4	nano02.lua - Declaração de uma variável . . . . .	21
3.5	nano02.java - Declaração de uma variável . . . . .	21
3.6	nano02.smali - Declaração de uma variável . . . . .	21
3.7	nano03.lua - Atribuição de um inteiro a uma variável . . . . .	21
3.8	nano03.java - Atribuição de um inteiro a uma variável . . . . .	21
3.9	nano03.smali - Atribuição de um inteiro a uma variável . . . . .	22
3.10	nano04.lua - Atribuição de uma soma de inteiros a uma variável . . . . .	22
3.11	nano04.java - Atribuição de uma soma de inteiros a uma variável . . . . .	22
3.12	nano04.smali - Atribuição de uma soma de inteiros a uma variável . . . . .	22
3.13	nano05.lua - Inclusão do comando de impressão . . . . .	23
3.14	nano05.java - Inclusão do comando de impressão . . . . .	23
3.15	nano05.smali - Inclusão do comando de impressão . . . . .	23
3.16	nano06.lua - Atribuição de uma subtração de inteiros a uma variável . . . . .	24
3.17	nano06.java - Atribuição de uma subtração de inteiros a uma variável . . . . .	24
3.18	nano06.smali - Atribuição de uma subtração de inteiros a uma variável . . . . .	24
3.19	nano07.lua - Inclusão do comando condicional . . . . .	25
3.20	nano07.java - Inclusão do comando condicional . . . . .	25
3.21	nano07.smali - Inclusão do comando condicional . . . . .	25
3.22	nano08.lua - Inclusão do comando condicional com parte senão . . . . .	26
3.23	nano08.java - Inclusão do comando condicional com parte senão . . . . .	26
3.24	nano08.smali - Inclusão do comando condicional com parte senão . . . . .	26
3.25	nano09.lua - Atribuição de duas operações aritméticas sobre inteiros a uma variável . . . . .	27
3.26	nano09.java - Atribuição de duas operações aritméticas sobre inteiros a uma variável . . . . .	27
3.27	nano09.smali - Atribuição de duas operações aritméticas sobre inteiros a uma variável . . . . .	27
3.28	nano10.lua - Atribuição de duas variáveis inteiras . . . . .	28
3.29	nano10.java - Atribuição de duas variáveis inteiras . . . . .	28
3.30	nano10.smali - Atribuição de duas variáveis inteiras . . . . .	28
3.31	nano11.lua - Introdução do comando de repetição enquanto . . . . .	29
3.32	nano11.java - Introdução do comando de repetição enquanto . . . . .	29
3.33	nano11.smali - Introdução do comando de repetição enquanto . . . . .	29

3.34	nano12.lua - Comando condicional aninhado em um comando de repetição .	30
3.35	nano12.java - Comando condicional aninhado em um comando de repetição .	31
3.36	nano12.smali - Comando condicional aninhado em um comando de repetição	31
3.37	micro01.lua - Converte graus Celsius para Fahrenheit . . . . .	32
3.38	micro01.java - Converte graus Celsius para Fahrenheit . . . . .	32
3.39	micro01.smali - Converte graus Celsius para Fahrenheit . . . . .	32
3.40	micro02.lua - Ler dois inteiros e decide qual é maior . . . . .	34
3.41	micro02.java - Ler dois inteiros e decide qual é maior . . . . .	34
3.42	micro02.smali - Ler dois inteiros e decide qual é maior . . . . .	35
3.43	micro03.lua - Lê um número e verifica se ele está entre 100 e 200 . . . . .	37
3.44	micro03.java - Lê um número e verifica se ele está entre 100 e 200 . . . . .	38
3.45	micro03.smali - Lê um número e verifica se ele está entre 100 e 200 . . . . .	38
3.46	micro04.lua - Lê números e informa quais estão entre 10 e 150 . . . . .	39
3.47	micro04.java - Lê números e informa quais estão entre 10 e 150 . . . . .	40
3.48	micro04.smali - Lê números e informa quais estão entre 10 e 150 . . . . .	40
3.49	micro05.lua - Lê strings e caracteres . . . . .	42
3.50	micro05.java - Lê strings e caracteres . . . . .	42
3.51	micro05.smali - Lê strings e caracteres . . . . .	43
3.52	micro06.lua - Escreve um número lido por extenso . . . . .	46
3.53	micro06.java - Escreve um número lido por extenso . . . . .	46
3.54	micro06.smali - Escreve um número lido por extenso . . . . .	47
3.55	micro07.lua - Decide se os números são positivos, zeros ou negativos . . . . .	49
3.56	micro07.java - Decide se os números são positivos, zeros ou negativos . . . . .	49
3.57	micro07.smali - Decide se os números são positivos, zeros ou negativos . . . . .	50
3.58	micro08.lua - Decide se um número é maior ou menor que 10 . . . . .	52
3.59	micro08.java - Decide se um número é maior ou menor que 10 . . . . .	52
3.60	micro08.smali - Decide se um número é maior ou menor que 10 . . . . .	53
3.61	micro09.lua - Cálculo de preços . . . . .	55
3.62	micro09.java - Cálculo de preços . . . . .	55
3.63	micro09.smali - Cálculo de preços . . . . .	56
3.64	micro10.lua - Calcula o fatorial de um número . . . . .	59
3.65	micro10.java - Calcula o fatorial de um número . . . . .	59
3.66	micro10.smali - Calcula o fatorial de um número . . . . .	60
3.67	micro11.lua - Decide se um número é positivo, zero ou negativo com auxílio de uma função . . . . .	61
3.68	micro11.java - Decide se um número é positivo, zero ou negativo com auxílio de uma função . . . . .	62
3.69	micro11.smali - Decide se um número é positivo, zero ou negativo com auxílio de uma função . . . . .	62
4.1	lexico.mll - Gera tokens dado código da linguagem LUA . . . . .	66
4.2	carregador.ml - Programa auxiliar para o analisador léxico . . . . .	70
4.3	Resultado de passar o analisador léxico no programa nano01.lua . . . . .	71
4.4	Resultado de passar o analisador léxico no programa nano02.lua . . . . .	71
4.5	Resultado de passar o analisador léxico no programa nano03.lua . . . . .	71
4.6	Resultado de passar o analisador léxico no programa nano04.lua . . . . .	71
4.7	Resultado de passar o analisador léxico no programa nano05.lua . . . . .	72
4.8	Resultado de passar o analisador léxico no programa nano06.lua . . . . .	72
4.9	Resultado de passar o analisador léxico no programa nano07.lua . . . . .	72
4.10	Resultado de passar o analisador léxico no programa nano08.lua . . . . .	72

4.11	Resultado de passar o analisador léxico no programa nano09.lua . . . . .	73
4.12	Resultado de passar o analisador léxico no programa nano10.lua . . . . .	73
4.13	Resultado de passar o analisador léxico no programa nano11.lua . . . . .	73
4.14	Resultado de passar o analisador léxico no programa nano12.lua . . . . .	73
4.15	Resultado de passar o analisador léxico no programa micro01.lua . . . . .	74
4.16	Resultado de passar o analisador léxico no programa micro02.lua . . . . .	74
4.17	Resultado de passar o analisador léxico no programa micro03.lua . . . . .	75
4.18	Resultado de passar o analisador léxico no programa micro04.lua . . . . .	75
4.19	Resultado de passar o analisador léxico no programa micro05.lua . . . . .	75
4.20	Resultado de passar o analisador léxico no programa micro06.lua . . . . .	76
4.21	Resultado de passar o analisador léxico no programa micro07.lua . . . . .	77
4.22	Resultado de passar o analisador léxico no programa micro08.lua . . . . .	77
4.23	Resultado de passar o analisador léxico no programa micro09.lua . . . . .	78
4.24	Resultado de passar o analisador léxico no programa micro10.lua . . . . .	78
4.25	Resultado de passar o analisador léxico no programa micro11.lua . . . . .	79
4.26	micro05Erro.lua - Lê strings e caracteres - com erro proposital de não fechar um comentário de bloco . . . . .	79
4.27	Resultado de passar o analisador léxico no programa micro05Erro.lua . . . .	80
4.28	micro06Erro.lua - Escreve um número lido por extenso - com erro proposital de adicionar caracter inválido . . . . .	80
4.29	Resultado de passar o analisador léxico no programa micro06Erro.lua . . . .	81
4.30	micro07.lua - Decide se os números são positivos, zeros ou negativos . . . .	81
4.31	Resultado de passar o analisador léxico no programa micro07Erro.lua . . . .	81
5.1	sintatico.mly - Código com a gramática do analisador sintático . . . . .	82
5.2	ast.ml - Código da árvore sintática abstrata . . . . .	86
5.3	sintaticoTest.ml - Código auxiliar . . . . .	87
5.4	lexico.mll - Novo código para o analisador léxico . . . . .	88
5.5	nano01.lua - Programa nano01 modificado . . . . .	92
5.6	nano01.txt - Resultado do analisador sintático executado sobre nano01.lua modificado . . . . .	92
5.7	nano02.lua - Programa nano02 modificado . . . . .	92
5.8	nano02.txt - Resultado do analisador sintático executado sobre nano02.lua modificado . . . . .	92
5.9	nano03.lua - Programa nano03 modificado . . . . .	93
5.10	nano03.txt - Resultado do analisador sintático executado sobre nano03.lua modificado . . . . .	93
5.11	nano04.lua - Programa nano04 modificado . . . . .	93
5.12	nano04.txt - Resultado do analisador sintático executado sobre nano04.lua modificado . . . . .	93
5.13	nano05.lua - Programa nano05 modificado . . . . .	93
5.14	nano05.txt - Resultado do analisador sintático executado sobre nano05.lua modificado . . . . .	93
5.15	nano06.lua - Programa nano06 modificado . . . . .	94
5.16	nano06.txt - Resultado do analisador sintático executado sobre nano06.lua modificado . . . . .	94
5.17	nano07.lua - Programa nano07 modificado . . . . .	94
5.18	nano07.txt - Resultado do analisador sintático executado sobre nano07.lua modificado . . . . .	94
5.19	nano08.lua - Programa nano08 modificado . . . . .	94

5.20	nano08.txt - Resultado do analisador sintático executado sobre nano08.lua modificado . . . . .	95
5.21	nano09.lua - Programa nano09 modificado . . . . .	95
5.22	nano09.txt - Resultado do analisador sintático executado sobre nano09.lua modificado . . . . .	95
5.23	nano10.lua - Programa nano10 modificado . . . . .	95
5.24	nano10.txt - Resultado do analisador sintático executado sobre nano10.lua modificado . . . . .	96
5.25	nano11.lua - Programa nano11 modificado . . . . .	96
5.26	nano11.txt - Resultado do analisador sintático executado sobre nano11.lua modificado . . . . .	96
5.27	nano12.lua - Programa nano12 modificado . . . . .	97
5.28	nano12.txt - Resultado do analisador sintático executado sobre nano12.lua modificado . . . . .	97
5.29	micro01.lua - Programa micro01 modificado . . . . .	98
5.30	micro01.txt - Resultado do analisador sintático executado sobre micro01.lua modificado . . . . .	98
5.31	micro02.lua - Programa micro02 modificado . . . . .	98
5.32	micro02.txt - Resultado do analisador sintático executado sobre micro02.lua modificado . . . . .	99
5.33	micro03.lua - Programa micro03 modificado . . . . .	99
5.34	micro03.txt - Resultado do analisador sintático executado sobre micro03.lua modificado . . . . .	100
5.35	micro04.lua - Programa micro04 modificado . . . . .	100
5.36	micro04.txt - Resultado do analisador sintático executado sobre micro04.lua modificado . . . . .	100
5.37	micro05.lua - Programa micro05 modificado . . . . .	101
5.38	micro05.txt - Resultado do analisador sintático executado sobre micro05.lua modificado . . . . .	101
5.39	micro06.lua - Programa micro06 modificado . . . . .	102
5.40	micro06.txt - Resultado do analisador sintático executado sobre micro06.lua modificado . . . . .	103
5.41	micro07.lua - Programa micro07 modificado . . . . .	103
5.42	micro07.txt - Resultado do analisador sintático executado sobre micro07.lua modificado . . . . .	104
5.43	micro08.lua - Programa micro08 modificado . . . . .	104
5.44	micro08.txt - Resultado do analisador sintático executado sobre micro08.lua modificado . . . . .	104
5.45	micro09.lua - Programa micro09 modificado . . . . .	105
5.46	micro09.txt - Resultado do analisador sintático executado sobre micro09.lua modificado . . . . .	105
5.47	micro10.lua - Programa micro10 modificado . . . . .	106
5.48	micro10.txt - Resultado do analisador sintático executado sobre micro10.lua modificado . . . . .	107
5.49	micro11.lua - Programa micro11 modificado . . . . .	107
5.50	micro11.txt - Resultado do analisador sintático executado sobre micro11.lua modificado . . . . .	108
5.51	teste01.lua - Programa de teste 01 . . . . .	109

5.52	saida01.txt - Resultado do analisador sintático executado sobre teste01.lua modificado . . . . .	109
5.53	teste02.lua - Programa de teste 02 . . . . .	109
5.54	saida02.txt - Resultado do analisador sintático executado sobre teste02.lua modificado . . . . .	109
5.55	teste03.lua - Programa de teste 03 . . . . .	109
5.56	saida03.txt - Resultado do analisador sintático executado sobre teste03.lua modificado . . . . .	109
5.57	teste04.lua - Programa de teste 04 . . . . .	109
5.58	saida04.txt - Resultado do analisador sintático executado sobre teste01.lua modificado . . . . .	110
5.59	teste05.lua - Programa de teste 05 . . . . .	110
5.60	saida05.txt - Resultado do analisador sintático executado sobre teste05.lua modificado . . . . .	110
5.61	teste06.lua - Programa de teste 06 . . . . .	110
5.62	saida06.txt - Resultado do analisador sintático executado sobre teste06.lua modificado . . . . .	111
5.63	teste07.lua - Programa de teste 07 . . . . .	111
5.64	saida07.txt - Resultado do analisador sintático executado sobre teste07.lua modificado . . . . .	111
5.65	teste08.lua - Programa de teste 08 . . . . .	111
5.66	saida08.txt - Resultado do analisador sintático executado sobre teste08.lua modificado . . . . .	111

# Sumário

<b>1</b>	<b>Introdução</b>	<b>9</b>
1.1	Sistema Operacional . . . . .	9
1.2	Lua . . . . .	9
1.2.1	Configurando Lua . . . . .	10
1.3	Java JDK . . . . .	10
1.3.1	Configurando a Java JDK . . . . .	10
1.4	Dalvik . . . . .	10
1.4.1	Configurando a Dalvik . . . . .	11
1.5	OCaml . . . . .	11
1.5.1	Configurando OCaml . . . . .	11
1.6	Smali . . . . .	11
1.7	Saindo de um arquivo .lua para um arquivo .smali . . . . .	12
1.8	Saindo de um arquivo .smali para um .dex . . . . .	13
1.9	Rodando o programa .dex no Dalvik . . . . .	13
1.9.1	O Android Debug Brigde - ADB . . . . .	13
1.9.2	Conectando-se a um aparelho . . . . .	14
1.9.3	Rodando o programa . . . . .	14
<b>2</b>	<b>A linguagem smali</b>	<b>16</b>
2.1	Entendendo um programa smali . . . . .	16
<b>3</b>	<b>label=chapter-programs</b>	<b>20</b>
3.1	Nano Programas . . . . .	20
3.2	Micro Programas . . . . .	32
<b>4</b>	<b>Analizador léxico</b>	<b>65</b>
4.1	Reconhecendo os Tokens . . . . .	65
4.2	Montagem do Analizador Léxico . . . . .	66
4.2.1	Código do Analizador Léxico . . . . .	66
4.2.2	Utilização . . . . .	70
4.3	Testando o Analizador Léxico . . . . .	71
4.3.1	Programas Nano . . . . .	71
4.3.2	Programas Micro . . . . .	74
4.4	Teste de Erros . . . . .	79
<b>5</b>	<b>Analizador Sintático</b>	<b>82</b>
5.1	Código do Analizador Sintático . . . . .	82
5.2	Código da Árvore Sintática Abstrata . . . . .	86
5.3	Código do sintaticoTest . . . . .	87
5.4	Novo código do analisador léxico . . . . .	88

5.5	Usando o analisador sintático . . . . .	91
5.5.1	Pré-requisitos . . . . .	91
5.5.2	Compilando o analisador sintático . . . . .	91
5.5.3	Executando o analisador . . . . .	91
5.6	Testando o Analisador Sintático . . . . .	92
5.6.1	Programas nano . . . . .	92
5.6.2	Programas micro . . . . .	98
5.7	Testes de Erros Sintáticos . . . . .	108



# Capítulo 1

## Introdução

Este Documento está sendo escrito como relatório para as diversas atividades propostas na disciplina de Construção de Compiladores, cuja finalidade é a de desenvolver meu conhecimento acerca do processo de construção de um compilador. Para tal, ao final da disciplina, terei construído um compilador de Lua para Dalvik, utilizando a linguagem OCaml. De forma que será necessário a aprendizagem da linguagem Lua e da linguagem OCaml de programação, assim como o entendimento do código gerado para a máquina virtual Dalvik e sua utilização.

Este capítulo contém informações acerca das tecnologias que serão usadas no decorrer deste trabalho, bem como suas configurações e modo de uso. Dessa forma, serve de base para os capítulos e atividades seguintes, possibilitando a execução deles.

### 1.1 Sistema Operacional

A versão 18.04.1 LTS da distribuição Ubuntu do sistema operacional Linux está sendo usado para a realização deste trabalho. O download da última versão do ubuntu (18.04.1 LTS quando esse relatório foi escrito) pode ser realizado pelo seguinte [link para a página oficial](#).

### 1.2 Lua

A linguagem de programação Lua é uma poderosa, leve e eficiente, projetada para entender aplicações. Ela permite programação orientada a objetos, programação orientada a dados e descrição de dados, programação procedural e programação funcional. A linguagem é tipada dinamicamente e é executada via interpretação de bytecodes para uma máquina virtual baseada em registradores. Além disso, Lua é rápida, portátil e de código aberto.

### 1.2.1 Configurando Lua

Para configurar a linguagem Lua para sua última versão (5.3.5 durante a escrita deste relatório) basta executar os seguintes comandos em um terminal linux

```
$ sudo apt install build-essential libreadline-dev
$ sudo apt-get update
$ mkdir lua_build
$ cd lua_build
$ sudo curl -R -O http://www.lua.org/ftp/lua-5.3.5.tar.gz
$ sudo tar -zxvf lua-5.3.5.tar.gz
$ cd lua-5.3.5
$ sudo make linux test
$ sudo make install
```

Para verificar a versão

```
$ lua -v
```

Para executar programas em .lua, basta executar

```
$ lua arquivo.lua
```

## 1.3 Java JDK

A Java JDK é um conjunto de utilitários cuja função é permitir criar sistemas de software para a plataforma java. Uma vez que utilizaremos a máquina virtual Dalvik, faz-se necessário instalar java e sua jdk na plataforma.

### 1.3.1 Configurando a Java JDK

Antes que se possa configurar a Dalvik, faz-se necessário a configuração do Java JDK. Será utilizado a Versão 1.8 da JDK devido a sua maior compatibilidade. Para tal, executa-se os seguintes comandos pelo terminal

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

Para verificar a versão instalada

```
$ java -version
```

## 1.4 Dalvik

A Dalvik é uma máquina virtual projetada e escrita por Dan Bornstein e com contribuições de outros engenheiros do Google para fazer parte da plataforma Android. Ao invés de

usar registradores virtuais, ela usa registradores reais, de forma que foi otimizada para utilizar pouca memória e permitir múltiplas instâncias da máquina virtual rodando ao mesmo tempo. Dalvik possui uma ferramenta chamada dx, incluída no SDK do Android, que permite transformar arquivos do formato .class em arquivos do formato .dex, que é o formato usado por ela.

### 1.4.1 Configurando a Dalvik

A dalvik está incluída no ambiente de desenvolvimento Android, portanto sua instalação é simples. Basta realizar o download do Android Studio ([link para a versão mais recente](#)) e instalar. Além disso, caso o sistema operacional que estiver sendo usado for de 64 bits, deve-se executar os seguintes comandos no terminal para que algumas bibliotecas de 32 bits necessárias sejam instaladas

```
$ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1
libbz2-1.0:i386
```

## 1.5 OCaml

OCaml (Objective Caml) é uma linguagem de programação de propósito geral, com ênfase em expressividade e segurança. A linguagem permite dois tipos de compilação, uma para bytecode (ocorrendo na máquina virtual zinc) ou para código de máquina nativo para um grande número de plataformas. Suas vantagens advêm de seu compilador, que gera código nativo rapidamente e com excelente desempenho no quesito tempo de execução, além de uma biblioteca base extensa.

### 1.5.1 Configurando OCaml

Para realizar a instalação da versão mais recente da linguagem OCaml (4.05.0 quando este relatório foi escrito) no Ubuntu, executa-se os seguintes comandos em um terminal Linux

```
$ sudo apt install ocaml-nox
$ sudo apt install ocaml
```

Para verificar sua versão, executa-se

```
$ ocaml -version
```

## 1.6 Smali

Como o arquivo .dex não é facilmente legível em sua forma bytecode, que é o formato executado pelo Dalvik, precisamos de uma ferramenta que faça a transformação de um

arquivo .dex para um arquivo .smali, que é um formato mais facilmente legível e desenvolvido justamente para isso. Smali oferece recursos para ir de .dex para .smali (baksmali) e para ir de .smali de volta para .dex (smali). Para realizar o download, entre nesse [link](#) (a versão usada nesse relatório foi a mais recente no tempo de escrita, 2.2.4). Após feito o download, mude o nome dos programas para smali e baksmali respectivamente para facilitar sua execução, de forma que não haja necessidade de digitar a versão nem o formato dos arquivos no terminal.

## 1.7 Saindo de um arquivo .lua para um arquivo .smali

Como dito anteriormente, um arquivo .smali é uma maneira prática de se ler o bytecode de um programa executável por Dalvik. Isso permite não apenas a fácil alteração de tais arquivos (uma vez que há uma maneira de conversão de volta de .smali para .dex), mas também auxilia o estudo de tal bytecode e do funcionamento de Dalvik. Porém, a linguagem lua não possui uma maneira prática de produzir um executável para Dalvik, de forma que os programas em .lua devam ser traduzidos de forma manual para programas em Java, com formato .java. A partir daí, pode-se compilar tais programas para .class por meio de um compilador java normal (o javac) e, por fim, para o arquivo final da Dalvik com formato .dex utilizando-se de uma ferramenta da própria Dalvik, o dx.

Para fazer isso, primeiramente deve-se compilar os programas em .java já traduzidos dos programas em .lua por meio da seguinte execução em um terminal Linux

```
$ javac arquivo.java
```

Sendo arquivo.java o nome do arquivo a ser compilado. Em seguida, pode-se executar os programas caso desejado com

```
$ java arquivo.class
```

Onde arquivo.class é o nome do arquivo que foi compilado a ser executado.

Após essa etapa, deve-se transformar os arquivos .class para arquivos .dex. Para isso, utiliza-se a ferramenta dx provida pela própria Dalvik. Pode-se fazer isso da seguinte forma

```
$ androidStudio-path/Sdk/build-tools/version/dx --dex --output=arquivo.dex
arquivo.class
```

Onde androidStudio-path é o caminho para a pasta onde está o Android Studio e version o número de versão do programa (home/nomeUsuario/Android/Sdk/build-tools/28.0.2/dx no meu caso, sendo nomeUsuario o nome de usuário da minha máquina).

Até esse momento, temos o arquivo .dex do programa original .java. Agora, para facilitar a leitura de seu código bytecode, vamos transformar esse arquivo para .smali utilizando o baksmali baixado anteriormente. Para isso, é necessário que o programa baksmali esteja na mesma pasta dos arquivos a serem transformados e que se execute o seguinte código

```
$ java -jar baksmali disassemble arquivo.dex
```

Onde arquivo.dex é o arquivo a ser traduzido para .smali.

Pronto! Agora já temos um arquivo de fácil leitura onde podemos analisar o bytecode rodado pela máquina virtual Dalvik. Basta abrir o arquivo .smali com um editor de texto comum qualquer.

## 1.8 Saindo de um arquivo .smali para um .dex

Com o propósito do trabalho de construir um compilador de lua para dalvik em mente, agora precisamos de uma forma de passar um arquivo do formato .smali para o formato .dex, para que ele possa ser executado pela máquina virtual. Para tal, usaremos o programa smali, baixado anteriormente. Com o arquivo do smali no mesmo endereço onde encontram-se os arquivos .smali a serem traduzidos, basta executar o seguinte comando para obter-se os arquivos .dex correspondentes

```
$ java -jar smali assemble arquivo.smali
```

Onde arquivo.smali é o arquivo a ser traduzido. Pronto! Agora é só mandar os arquivos .dex para a Dalvik para que sejam executados.

## 1.9 Rodando o programa .dex no Dalvik

Para que possamos executar os arquivos .dex em Dalvik, necessitamos de um aparelho Android, seja um aparelho físico ou um emulador que rode o sistema operacional em questão. O processo para os dois é bastante similar, mudando apenas a forma de se conectar aos dois.

### 1.9.1 O Android Debug Bridge - ADB

O ADB é um componente da SDK de ferramentas da plataforma do Android. É uma utilidade da linha de comando que permite realizar ações como controlar o seu aparelho de um computador por meio de uma conexão USB, instalar e desinstalar apps, copiar arquivos tanto de quanto para seu aparelho, executar comandos shell e entre outras funções. É uma ferramenta essencial para o desenvolvimento para a plataforma Android.

#### Configurando o ADB

Para instalar o ADB bastam as instruções a seguir serem executadas em um terminal em seu linux

```
$ sudo apt update
$ sudo apt install android-tools-adb
```

Para verificar a versão instalada

```
$ adb version
```

Por fim, uma função útil é a de verificar a lista de aparelhos disponíveis para uso com a ferramenta, podendo ser executada da seguinte forma

```
$ adb devices
```

## 1.9.2 Conectando-se a um aparelho

Aqui é onde os processos entre um simulador de aparelho Android e um aparelho real se diferem. Seguem instruções para configurar as duas alternativas.

### Aparelho real

Para configurar um aparelho real primeiramente deve-se habilitar a função de desenvolvedor nas configurações. Esse processo pode variar de aparelho para aparelho, porém de forma geral deve-se acessar as configurações e então a sobre o dispositivo. No final da lista deve-se encontrar uma informação com o Número da versão. Pressione repetidas vezes essa informação para habilitar as opções de desenvolvedor. Caso não funcione em seu aparelho Android, faça uma pesquisa na internet de como fazê-lo para seu dispositivo em específico.

Com as Opções do desenvolvedor habilitadas, ache a opção de Depuração USB e a ative. Pronto, agora toda vez que quiser usar seu aparelho Android para realizar operações pela ADB basta conectá-lo à um computador utilizando-se de um cabo USB e mantendo-o desbloqueado.

### Simulador Android

Para um simulador Android, use o AVD Manager que já vem com o programa do Android Studio. Por meio dele, basta iniciar um aparelho que ele já estará pronto para ser usado com a ADB.

## 1.9.3 Rodando o programa

Agora com essas configurações realizadas, os passos para rodar o programa .dex são os mesmos tanto para um aparelho físico quanto para um aparelho simulado. Garantindo-se que há um aparelho disponível para a adb, deve-se primeiramente mandar o arquivo .dex a ser executado para o aparelho. Faz-se isso por meio do comando

```
$ adb push arquivo.dex /sdcard/
```

Onde arquivo.dex é o arquivo a ser executado e /sdcard/ o local no aparelho para onde esse arquivo será enviado. O próximo passo é pedir para a Dalvikvm executar o programa, por meio da adb shell como se segue

```
$ adb shell dalvikvm -cp /sdcard/arquivo.dex arquivoClasse
```

Onde arquivoClasse é a especificação de qual classe do arquivo a Dalvik irá executar. No caso de um exemplo onde o programa micro01 foi mandado para a pasta /sdcard/ do aparelho e sua única classe é também chamada micro01, temos o seguinte comando para executá-lo

```
$ adb shell dalvikvm -cp /sdcard/micro01.dex micro01
```

Pronto, agora a dalvik já está executando o programa .dex utilizando o aparelho Android, seja ele simulado ou real.

Obs: Caso haja mais de um dispositivo disponível para a ADB, faz-se necessário explicitar qual deles está-se usando por meio de parâmetros adicionais.

# Capítulo 2

## A linguagem smali

O Smali e o Baksmali são um assembler e disassembler para o formato .dex usado pela Dalvik no Android. Sua sintaxe tem base em Jasmim e Dedexer, outros disassemblers, e suporta toda a funcionalidade do formato dex. Por isso, saber usar ambas as ferramentas é de grande valia no entendimento e desenvolvimento do trabalho proposto.

### 2.1 Entendendo um programa smali

Para entendermos o formato smali, vamos analisar a estrutura de um programa smali e, em seguida, estudar um exemplo do formato.

#### Estrutura de um programa smali

A estrutura de um programa smali, de forma geral, segue a estrutura apresentada na imagem a seguir

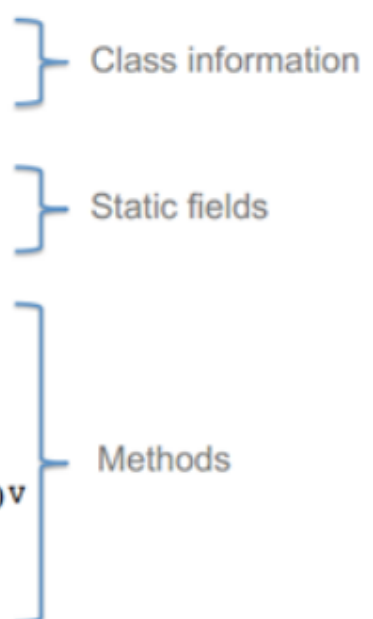
```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java"

# static fields
.field public static final TAG:Ljava/lang/String; =
    "ApkudoUtils"

# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```



The diagram illustrates the structure of a smali program. It uses blue brackets on the right side to group lines of code into three main categories:

- Class information:** This group includes the first three lines of the code: `.class public Lcom/apkudo/util/Serializer;`, `.super Ljava/lang/Object;`, and `.source "Serializer.java"`.
- Static fields:** This group includes the lines starting with `# static fields`, `.field public static final TAG:Ljava/lang/String; =`, and `"ApkudoUtils"`.
- Methods:** This group includes the lines starting with `# direct methods`, `.method public constructor <init>()V`, `.registers 1`, `.prologue`, `.line 5`, `invoke-direct {p0}, Ljava/lang/Object;-><init>()V`, `return-void`, and `.end method`.



## Exemplo de um programa smali

Realizar a análise de um programa smali permitirá melhorar nosso entendimento sobre a sintaxe desse assembler. Para isso, iremos utilizar o pseudocódigo a seguir como exemplo

**Listagem 2.1: enunciado do programa micro04 - Lê números e informa quais estão entre 10 e 150**

```

1 algoritmo "micro04"
2 /*
3 Função: Ler 5 números e ao final informar quantos número(s)
4 est(á)ão no intervalo entre 10 (inclusive) e 150 (inclusive).
5 */
6
7 var
8 x, num, intervalo: inteiro
9
10 início
11
12 para x de 1 até 5 faça
13     escreva("Digite um número: ")
14     leia(num)
15     se num >= 10 então
16         se num <= 150 então
17             intervalo      intervalo + 1
18         fim_se
19     fim_se
20 fim_para
21
22 escreval("Ao total, foram digitados ",intervalo," números no
23 intervalo entre 10 e 150")
24
25 fim_algoritmo

```

A tradução desse pseudocódigo para um código escrito em java (micro04.java) é mostrado a seguir

**Listagem 2.2: micro04.java - Lê números e informa quais estão entre 10 e 150**

```

1 public class micro04{
2     public static void main(String[] args){
3         int x, num, intervalo;
4         intervalo = 0;
5         for (x = 1; x <= 5; x++){
6             System.out.print("Digite um numero: ");
7             num = Integer.parseInt(System.console().readLine());
8             if(num >= 10){
9                 if(num <=150){
10                     intervalo = intervalo + 1;
11                 }
12             }
13         }
14         System.out.println("Ao total, foram digitados " + intervalo + "
15                             numeros no intervalo entre 10 e 150");
16     }
17 }

```

E, finalmente, a tradução desse arquivo .java para o smali resulta no seguinte arquivo

Listagem 2.3: micro04.smali - Lê números e informa quais estão entre 10 e 150

```

1 .class public Lmicro04;
2 .super Ljava/lang/Object;
3 .source "micro04.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v0, 0x0
23
24     .line 5
25     const/4 v1, 0x1
26
27     :goto_2
28     const/4 v2, 0x5
29
30     if-gt v1, v2, :cond_25
31
32     .line 6
33     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
34
35     const-string v3, "Digite um numero: "
36
37     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
38
39     .line 7
40     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
41
42     move-result-object v2
43
44     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
45
46     move-result-object v2
47
48     invoke-static {v2}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
49
50     move-result v2
51
52     .line 8
53     const/16 v3, 0xa
54
55     if-lt v2, v3, :cond_22
56
57     .line 9

```

## 2.1

```
58     const/16 v3, 0x96
59
60     if-gt v2, v3, :cond_22
61
62     .line 10
63     add-int/lit8 v0, v0, 0x1
64
65     .line 5
66     :cond_22
67     add-int/lit8 v1, v1, 0x1
68
69     goto :goto_2
70
71     .line 14
72     :cond_25
73     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
74
75     new-instance v2, Ljava/lang/StringBuilder;
76
77     invoke-direct {v2}, Ljava/lang/StringBuilder;--><init>()V
78
79     const-string v3, "Ao total, foram digitados "
80
81     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
82
83     move-result-object v2
84
85     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;-->append(I)Ljava/
        lang/StringBuilder;
86
87     move-result-object v0
88
89     const-string v2, " numeros no intervalo entre 10 e 150"
90
91     invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
92
93     move-result-object v0
94
95     invoke-virtual {v0}, Ljava/lang/StringBuilder;-->toString()Ljava/lang/
        String;
96
97     move-result-object v0
98
99     invoke-virtual {v1, v0}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
100
101     .line 15
102     return-void
103 .end method
```

---

# Capítulo 3

## Programas

### 3.1 Nano Programas

Listagem 3.1: nano01.lua - Módulo mínimo que caracteriza um programa

```
1 function main()  
2 end
```

Listagem 3.2: nano01.java - Módulo mínimo que caracteriza um programa

```
1 public class nano01{  
2     public static void main(String[] args){  
3  
4     }  
5 }
```

Listagem 3.3: nano01.smali - Módulo mínimo que caracteriza um programa

```
1 .class public Lnano01;  
2 .super Ljava/lang/Object;  
3 .source "nano01.java"  
4  
5  
6 # direct methods  
7 .method public constructor <init>()V  
8     .registers 1  
9  
10    .prologue  
11    .line 1  
12    invoke-direct {p0}, Ljava/lang/Object; -><init>()V  
13  
14    return-void  
15 .end method  
16  
17 .method public static main([Ljava/lang/String;)V  
18     .registers 1  
19  
20    .prologue  
21    .line 4  
22    return-void
```

23 **.end method**

---

#### Listagem 3.4: nano02.lua - Declaração de uma variável

```
1 function main()  
2     local n  
3 end
```

---

#### Listagem 3.5: nano02.java - Declaração de uma variável

```
1 public class nano02{  
2     public static void main(String[] args){  
3         int n;  
4     }  
5 }
```

---

#### Listagem 3.6: nano02.smali - Declaração de uma variável

```
1 .class public Lnano02;  
2 .super Ljava/lang/Object;  
3 .source "nano02.java"  
4  
5  
6 # direct methods  
7 .method public constructor <init>()V  
8     .registers 1  
9  
10    .prologue  
11    .line 1  
12    invoke-direct {p0}, Ljava/lang/Object; -> <init>()V  
13  
14    return-void  
15 .end method  
16  
17 .method public static main([Ljava/lang/String;)V  
18    .registers 1  
19  
20    .prologue  
21    .line 4  
22    return-void  
23 .end method
```

---

#### Listagem 3.7: nano03.lua - Atribuição de um inteiro a uma variável

```
1 function main()  
2     local n  
3     n = 1  
4 end  
5  
6 main()
```

---

#### Listagem 3.8: nano03.java - Atribuição de um inteiro a uma variável

```
1 public class nano03{  
2     public static void main(String[] args){  
3         int n;  
4         n = 1;  
5     }  
6 }
```

```

5     }
6 }

```

Listagem 3.9: nano03.smali - Atribuição de um inteiro a uma variável

```

1 .class public Lnano03;
2 .super Ljava/lang/Object;
3 .source "nano03.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -> <init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 1
19
20     .prologue
21     .line 4
22     .line 5
23     return-void
24 .end method

```

Listagem 3.10: nano04.lua - Atribuição de uma soma de inteiros a uma variável

```

1 function main()
2     local n
3     n = 1 + 2
4 end
5
6 main()

```

Listagem 3.11: nano04.java - Atribuição de uma soma de inteiros a uma variável

```

1 public class nano04{
2     public static void main(String[] args){
3         int n;
4         n = 1 + 2;
5     }
6 }

```

Listagem 3.12: nano04.smali - Atribuição de uma soma de inteiros a uma variável

```

1 .class public Lnano04;
2 .super Ljava/lang/Object;
3 .source "nano04.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1

```

### 3.1

```
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 1
19
20     .prologue
21     .line 4
22     .line 5
23     return-void
24 .end method
```

---

Listagem 3.13: nano05.lua - Inclusão do comando de impressão

```
1 function main()
2     local n
3     n = 2
4     print(n)
5 end
6
7 main()
```

---

Listagem 3.14: nano05.java - Inclusão do comando de impressão

```
1 public class nano05{
2     public static void main(String[] args){
3         int n;
4         n = 2;
5         System.out.print(n);
6     }
7 }
```

---

Listagem 3.15: nano05.smali - Inclusão do comando de impressão

```
1 .class public Lnano05;
2 .super Ljava/lang/Object;
3 .source "nano05.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
```

```

20     .prologue
21     .line 4
22     const/4 v0, 0x2
23
24     .line 5
25     sget-object v1, Ljava/lang/System; ->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream; ->print(I)V
28
29     .line 6
30     return-void
31 .end method

```

Listagem 3.16: nano06.lua - Atribuição de uma subtração de inteiros a uma variável

```

1 function main()
2     local n
3     n = 1 - 2
4     print(n)
5 end
6
7 main()

```

Listagem 3.17: nano06.java - Atribuição de uma subtração de inteiros a uma variável

```

1 public class nano06{
2     public static void main(String[] args){
3         int n;
4         n = 1 - 2;
5         System.out.print(n);
6     }
7 }

```

Listagem 3.18: nano06.smali - Atribuição de uma subtração de inteiros a uma variável

```

1 .class public Lnano06;
2 .super Ljava/lang/Object;
3 .source "nano06.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20    .prologue
21    .line 4
22    const/4 v0, -0x1
23

```



### 3.1

```
24     .line 5
25     sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->print(I)V
28
29     .line 6
30     return-void
31 .end method
```

---

Listagem 3.19: nano07.lua - Inclusão do comando condicional

```
1 function main()
2     local n
3     n = 1
4     if n == 1 then
5         print(n)
6     end
7 end
8
9 main()
```

---

Listagem 3.20: nano07.java - Inclusão do comando condicional

```
1 public class nano07{
2     public static void main(String[] args){
3         int n;
4         n = 1;
5         if(n == 1){
6             System.out.print(n);
7         }
8     }
9 }
```

---

Listagem 3.21: nano07.smali - Inclusão do comando condicional

```
1 .class public Lnano07;
2 .super Ljava/lang/Object;
3 .source "nano07.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20    .prologue
21    .line 4
22    const/4 v0, 0x1
23
```

```

24     .line 6
25     sget-object v1, Ljava/lang/System; ->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream; ->print(I)V
28
29     .line 8
30     return-void
31 .end method

```

Listagem 3.22: nano08.lua - Inclusão do comando condicional com parte senão

```

1 function main()
2     local n
3     n = 1
4     if n == 1 then
5         print(n)
6     else
7         print(0)
8     end
9 end
10
11 main()

```

Listagem 3.23: nano08.java - Inclusão do comando condicional com parte senão

```

1 public class nano08{
2     public static void main(String[] args){
3         int n;
4         n = 1;
5         if(n == 1){
6             System.out.print(n);
7         }else{
8             System.out.print(0);
9         }
10    }
11 }

```

Listagem 3.24: nano08.smali - Inclusão do comando condicional com parte senão

```

1 .class public Lnano08;
2 .super Ljava/lang/Object;
3 .source "nano08.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19

```

### 3.1

```
20 .prologue
21 .line 4
22 const/4 v0, 0x1
23
24 .line 6
25 sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
26
27 invoke-virtual {v1, v0}, Ljava/io/PrintStream;-->print(I)V
28
29 .line 10
30 return-void
31 .end method
```

---

Listagem 3.25: nano09.lua - Atribuição de duas operações aritméticas sobre inteiros a uma variável

```
1 function main()
2     local n
3     n = 1 + 1 / 2
4     if n == 1 then
5         print(n)
6     else
7         print(0)
8     end
9 end
10
11 main()
```

---

Listagem 3.26: nano09.java - Atribuição de duas operações aritméticas sobre inteiros a uma variável

```
1 public class nano09{
2     public static void main(String[] args){
3         int n;
4         n = 1 + 1 / 2;
5         if(n == 1){
6             System.out.print(n);
7         }else{
8             System.out.print(0);
9         }
10    }
11 }
```

---

Listagem 3.27: nano09.smali - Atribuição de duas operações aritméticas sobre inteiros a uma variável

```
1 .class public Lnano09;
2 .super Ljava/lang/Object;
3 .source "nano09.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10 .prologue
11 .line 1
12 invoke-direct {p0}, Ljava/lang/Object;--><init>()V
```

```

13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 6
25     sget-object v1, Ljava/lang/System; ->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream; ->print(I)V
28
29     .line 10
30     return-void
31 .end method

```

---

Listagem 3.28: nano10.lua - Atribuição de duas variáveis inteiras

```

1 function main()
2     local n, m
3     n, m = 1, 2
4     if n == m then
5         print(n)
6     else
7         print(0)
8     end
9 end
10
11 main()

```

---

Listagem 3.29: nano10.java - Atribuição de duas variáveis inteiras

```

1 public class nano10{
2     public static void main(String[] args){
3         int n, m;
4         n = 1;
5         m = 2;
6         if(n == m ){
7             System.out.print(n);
8         }else{
9             System.out.print(0);
10        }
11    }
12 }

```

---

Listagem 3.30: nano10.smali - Atribuição de duas variáveis inteiras

```

1 .class public Lnano10;
2 .super Ljava/lang/Object;
3 .source "nano10.java"
4
5
6 # direct methods
7 .method public constructor <init>()V

```

### 3.1

```
8      .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     .line 9
23     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
24
25     const/4 v1, 0x0
26
27     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(I)V
28
29     .line 11
30     return-void
31 .end method
```

---

Listagem 3.31: nano11.lua - Introdução do comando de repetição enquanto

```
1 function main()
2     local n, m, x
3     n, m, x = 1, 2, 5
4     while x > n do
5         n = n + m
6         print(n)
7     end
8 end
9
10 main()
```

---

Listagem 3.32: nano11.java - Introdução do comando de repetição enquanto

```
1 public class nano11{
2     public static void main(String[] args){
3         int n, m, x;
4         n = 1;
5         m = 2;
6         x = 5;
7         while(x > n){
8             n = n + m;
9             System.out.print(n);
10        }
11    }
12 }
```

---

Listagem 3.33: nano11.smali - Introdução do comando de repetição enquanto

```
1 .class public Lnano11;
2 .super Ljava/lang/Object;
3 .source "nano11.java"
```

```

4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 5
25     const/4 v1, 0x2
26
27     .line 6
28     const/4 v2, 0x5
29
30     .line 7
31     :goto_3
32     if-le v2, v0, :cond_c
33
34     .line 8
35     add-int/2addr v0, v1
36
37     .line 9
38     sget-object v3, Ljava/lang/System; ->out:Ljava/io/PrintStream;
39
40     invoke-virtual {v3, v0}, Ljava/io/PrintStream; ->print(I)V
41
42     goto :goto_3
43
44     .line 11
45     :cond_c
46     return-void
47 .end method

```

Listagem 3.34: nano12.lua - Comando condicional aninhado em um comando de repetição

```

1 function main()
2     local n, m, x
3     n, m, x = 1, 2, 5
4     while x > n do
5         if n == m then
6             print(n)
7         else
8             print(0)
9         end
10        x = x - 1
11    end
12 end

```

## 3.1

```
13
14 main()
```

Listagem 3.35: nano12.java - Comando condicional aninhado em um comando de repetição

```
1 public class nano12{
2     public static void main(String[] args){
3         int n, m, x;
4         n = 1;
5         m = 2;
6         x = 5;
7         while (x > n){
8             if (n == m){
9                 System.out.print(n);
10            }else{
11                System.out.print(0);
12            }
13            x = x - 1;
14        }
15    }
16 }
```

Listagem 3.36: nano12.smali - Comando condicional aninhado em um comando de repetição

```
1 .class public Lnano12;
2 .super Ljava/lang/Object;
3 .source "nano12.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v1, 0x1
23
24     .line 6
25     const/4 v0, 0x5
26
27     .line 7
28     :goto_2
29     if-le v0, v1, :cond_d
30
31     .line 11
32     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
33
34     const/4 v3, 0x0
35
```

```

36     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(I)V
37
38     .line 13
39     add-int/lit8 v0, v0, -0x1
40
41     goto :goto_2
42
43     .line 15
44     :cond_d
45     return-void
46 .end method

```

---

## 3.2 Micro Programas

### Listagem 3.37: micro01.lua - Converte graus Celsius para Fahrenheit

```

1 function main()
2     local cel, far
3     print(" tabela de conversao: Celsius -> Fahrenheit\n")
4     print("Digite a temperatura em Celsius: ")
5     cel = io.read("*number")
6     far = (9*cel + 160)/5
7     print("A nova temperatura e: " ..far.." F")
8 end
9
10 main()

```

---

### Listagem 3.38: micro01.java - Converte graus Celsius para Fahrenheit

```

1 public class micro01{
2     public static void main(String[] args){
3         double cel, far;
4         System.out.println("Tabela de conversao: Celsius -> Fahrenheit");
5         System.out.print("Digite a temperatura em Celsius: ");
6         cel = Double.parseDouble(System.console().readLine());
7         far = ((9 * cel) + 160) / 5;
8         System.out.println("A nova temperatura e " + far + " F");
9     }
10 }

```

---

### Listagem 3.39: micro01.smali - Converte graus Celsius para Fahrenheit

```

1 .class public Lmicro01;
2 .super Ljava/lang/Object;
3 .source "micro01.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method

```



## 3.2

```
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 6
19
20     .prologue
21     .line 4
22     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
23
24     const-string v1, "Tabela de conversao: Celsius -> Fahrenheit"
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
27
28     .line 5
29     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
30
31     const-string v1, "Digite a temperatura em Celsius: "
32
33     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(Ljava/lang/
        String;)V
34
35     .line 6
36     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
37
38     move-result-object v0
39
40     invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;
41
42     move-result-object v0
43
44     invoke-static {v0}, Ljava/lang/Double;-->parseDouble(Ljava/lang/String
        ;)D
45
46     move-result-wide v0
47
48     .line 7
49     const-wide/high16 v2, 0x4022000000000000L    # 9.0
50
51     mul-double/2addr v0, v2
52
53     const-wide/high16 v2, 0x4064000000000000L    # 160.0
54
55     add-double/2addr v0, v2
56
57     const-wide/high16 v2, 0x4014000000000000L    # 5.0
58
59     div-double/2addr v0, v2
60
61     .line 8
62     sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
63
64     new-instance v3, Ljava/lang/StringBuilder;
65
66     invoke-direct {v3}, Ljava/lang/StringBuilder;--><init>()V
67
68     const-string v4, "A nova temperatura e "
69
70     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
```

```

71
72     move-result-object v3
73
74     invoke-virtual {v3, v0, v1}, Ljava/lang/StringBuilder; ->append(D)Ljava
        /lang/StringBuilder;
75
76     move-result-object v0
77
78     const-string v1, " F"
79
80     invoke-virtual {v0, v1}, Ljava/lang/StringBuilder; ->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
81
82     move-result-object v0
83
84     invoke-virtual {v0}, Ljava/lang/StringBuilder; ->toString()Ljava/lang/
        String;
85
86     move-result-object v0
87
88     invoke-virtual {v2, v0}, Ljava/io/PrintStream; ->println(Ljava/lang/
        String;)V
89
90     .line 9
91     return-void
92 .end method

```

#### Listagem 3.40: micro02.lua - Ler dois inteiros e decide qual é maior

```

1 function main()
2     local num1, num2
3     print("Digite o primeiro numero: ")
4     num1 = io.read("*number")
5     print("Digite o segundo numero: ")
6     num2 = io.read("*number")
7
8     if num1 > num2 then
9         print("O primeiro número "..num1.." é maior que o segundo "..num2)
10    else
11        print("O segundo número "..num2.." é maior que o primeiro "..num1)
12    end
13 end
14
15 main()

```

#### Listagem 3.41: micro02.java - Ler dois inteiros e decide qual é maior

```

1 public class micro02{
2     public static void main(String[] args){
3         int num1, num2;
4         System.out.print("Digite o primeiro numero: ");
5         num1 = Integer.parseInt(System.console().readLine());
6         System.out.print("Digite o segundo numero: ");
7         num2 = Integer.parseInt(System.console().readLine());
8
9         if(num1 > num2){
10             System.out.print("O primeiro numero " + num1 + " e maior que o
                segundo " + num2);
11         }else{

```

## 3.2

```
12         System.out.print("O segundo numero " + num2 + " e maior que o  
           primeiro " + num1);  
13     }  
14 }  
15 }
```

Listagem 3.42: micro02.smali - Ler dois inteiros e decide qual é maior

```
1 .class public Lmicro02;  
2 .super Ljava/lang/Object;  
3 .source "micro02.java"  
4  
5  
6 # direct methods  
7 .method public constructor <init>()V  
8     .registers 1  
9  
10    .prologue  
11    .line 1  
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V  
13  
14    return-void  
15 .end method  
16  
17 .method public static main([Ljava/lang/String;)V  
18     .registers 6  
19  
20     .prologue  
21     .line 4  
22     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;  
23  
24     const-string v1, "Digite o primeiro numero: "  
25  
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print (Ljava/lang/  
        String;)V  
27  
28     .line 5  
29     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;  
30  
31     move-result-object v0  
32  
33     invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;  
34  
35     move-result-object v0  
36  
37     invoke-static {v0}, Ljava/lang/Integer;-->parseInt (Ljava/lang/String;)I  
38  
39     move-result v0  
40  
41     .line 6  
42     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;  
43  
44     const-string v2, "Digite o segundo numero: "  
45  
46     invoke-virtual {v1, v2}, Ljava/io/PrintStream;-->print (Ljava/lang/  
        String;)V  
47  
48     .line 7  
49     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
```

```

50
51     move-result-object v1
52
53     invoke-virtual {v1}, Ljava/io/Console;->readLine()Ljava/lang/String;
54
55     move-result-object v1
56
57     invoke-static {v1}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
58
59     move-result v1
60
61     .line 9
62     if-le v0, v1, :cond_4b
63
64     .line 10
65     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
66
67     new-instance v3, Ljava/lang/StringBuilder;
68
69     invoke-direct {v3}, Ljava/lang/StringBuilder;-<init>()V
70
71     const-string v4, "O primeiro numero "
72
73     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
74
75     move-result-object v3
76
77     invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
78
79     move-result-object v0
80
81     const-string v3, " e maior que o segundo "
82
83     invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
84
85     move-result-object v0
86
87     invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
88
89     move-result-object v0
90
91     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
92
93     move-result-object v0
94
95     invoke-virtual {v2, v0}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
96
97     .line 14
98     :goto_4a
99     return-void
100
101     .line 12
102     :cond_4b

```

```

103     sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
104
105     new-instance v3, Ljava/lang/StringBuilder;
106
107     invoke-direct {v3}, Ljava/lang/StringBuilder;--><init>()V
108
109     const-string v4, "O segundo numero "
110
111     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
112
113     move-result-object v3
114
115     invoke-virtual {v3, v1}, Ljava/lang/StringBuilder;-->append(I)Ljava/
        lang/StringBuilder;
116
117     move-result-object v1
118
119     const-string v3, " e maior que o primeiro "
120
121     invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
122
123     move-result-object v1
124
125     invoke-virtual {v1, v0}, Ljava/lang/StringBuilder;-->append(I)Ljava/
        lang/StringBuilder;
126
127     move-result-object v0
128
129     invoke-virtual {v0}, Ljava/lang/StringBuilder;-->toString()Ljava/lang/
        String;
130
131     move-result-object v0
132
133     invoke-virtual {v2, v0}, Ljava/io/PrintStream;-->print(Ljava/lang/
        String;)V
134
135     goto :goto_4a
136 .end method

```

Listagem 3.43: micro03.lua - Lê um número e verifica se ele está entre 100 e 200

```

1 function main()
2     local numero
3     print("Digite um numero: ")
4     numero = io.read("*number")
5     if numero >= 100 then
6         if numero <= 200 then
7             print("O numero esta no intervalo entre 100 e 200\n")
8         else
9             print("O numero nao esta no intervalo entre 100 e 200\n")
10        end
11    else
12        print("O numero nao esta no intervalo entre 100 e 200\n")
13    end
14 end
15
16 main()

```

Listagem 3.44: micro03.java - Lê um número e verifica se ele está entre 100 e 200

```

1 public class micro03{
2     public static void main(String[] args){
3         int numero;
4         System.out.print("Digite um numero: ");
5         numero = Integer.parseInt(System.console().readLine());
6         if(numero >= 100){
7             if(numero <= 200){
8                 System.out.println("O numero esta no intervalo entre 100 e
9                                     200");
10            }else{
11                System.out.println("O numero nao esta no intervalo entre
12                                    100 e 200");
13            }
14        }else{
15            System.out.println("O numero nao esta no intervalo entre 100 e
16                                200");
17        }
18    }
19 }

```

Listagem 3.45: micro03.smali - Lê um número e verifica se ele está entre 100 e 200

```

1 .class public Lmicro03;
2 .super Ljava/lang/Object;
3 .source "micro03.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um numero: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->print(Ljava/lang/
27         String;)V
28
29     .line 5
30     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
31
32     move-result-object v0
33
34     invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
35
36     move-result-object v0

```

## 3.2

```
36
37     invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;) I
38
39     move-result v0
40
41     .line 6
42     const/16 v1, 0x64
43
44     if-lt v0, v1, :cond_2b
45
46     .line 7
47     const/16 v1, 0xc8
48
49     if-gt v0, v1, :cond_23
50
51     .line 8
52     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
53
54     const-string v1, "O numero esta no intervalo entre 100 e 200"
55
56     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
57
58     .line 15
59     :goto_22
60     return-void
61
62     .line 10
63     :cond_23
64     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66     const-string v1, "O numero nao esta no intervalo entre 100 e 200"
67
68     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
69
70     goto :goto_22
71
72     .line 13
73     :cond_2b
74     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
75
76     const-string v1, "O numero nao esta no intervalo entre 100 e 200"
77
78     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
79
80     goto :goto_22
81 .end method
```

Listagem 3.46: micro04.lua - Lê números e informa quais estão entre 10 e 150

```
1 function main()
2     local x, num, intervalo
3     intervalo = 0
4     for x = 1, 5, 1
5     do
6         print("Digite um numero: ")
7         num = io.read("*number")
```

```

8         if num >= 10 then
9             if num <= 150 then
10                 intervalo = intervalo + 1
11             end
12         end
13     end
14     print("Ao total, foram digitados "..intervalo.." numeros no intervalo
15         entre 10 e 150")
16 end
17 main()

```

Listagem 3.47: micro04.java - Lê números e informa quais estão entre 10 e 150

```

1 public class micro04{
2     public static void main(String[] args){
3         int x, num, intervalo;
4         intervalo = 0;
5         for (x = 1; x <= 5; x++){
6             System.out.print("Digite um numero: ");
7             num = Integer.parseInt(System.console().readLine());
8             if(num >= 10){
9                 if(num <=150){
10                     intervalo = intervalo + 1;
11                 }
12             }
13         }
14         System.out.println("Ao total, foram digitados " + intervalo + "
15             numeros no intervalo entre 10 e 150");
16     }
17 }

```

Listagem 3.48: micro04.smali - Lê números e informa quais estão entre 10 e 150

```

1 .class public Lmicro04;
2 .super Ljava/lang/Object;
3 .source "micro04.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v0, 0x0
23
24     .line 5
25     const/4 v1, 0x1

```



## 3.2

```
26
27 :goto_2
28 const/4 v2, 0x5
29
30 if-gt v1, v2, :cond_25
31
32 .line 6
33 sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
34
35 const-string v3, "Digite um numero: "
36
37 invoke-virtual {v2, v3}, Ljava/io/PrintStream;-->print(Ljava/lang/
    String;)V
38
39 .line 7
40 invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
41
42 move-result-object v2
43
44 invoke-virtual {v2}, Ljava/io/Console;-->readLine()Ljava/lang/String;
45
46 move-result-object v2
47
48 invoke-static {v2}, Ljava/lang/Integer;-->parseInt(Ljava/lang/String;)I
49
50 move-result v2
51
52 .line 8
53 const/16 v3, 0xa
54
55 if-lt v2, v3, :cond_22
56
57 .line 9
58 const/16 v3, 0x96
59
60 if-gt v2, v3, :cond_22
61
62 .line 10
63 add-int/lit8 v0, v0, 0x1
64
65 .line 5
66 :cond_22
67 add-int/lit8 v1, v1, 0x1
68
69 goto :goto_2
70
71 .line 14
72 :cond_25
73 sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
74
75 new-instance v2, Ljava/lang/StringBuilder;
76
77 invoke-direct {v2}, Ljava/lang/StringBuilder;--><init>()V
78
79 const-string v3, "Ao total, foram digitados "
80
81 invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
    String;)Ljava/lang/StringBuilder;
82
```

```

83     move-result-object v2
84
85     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
86
87     move-result-object v0
88
89     const-string v2, " numeros no intervalo entre 10 e 150"
90
91     invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
92
93     move-result-object v0
94
95     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
96
97     move-result-object v0
98
99     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
100
101     .line 15
102     return-void
103 .end method

```

#### Listagem 3.49: micro05.lua - Lê strings e caracteres

```

1 function main()
2     local nome, sexo, x, h, m
3     h, m = 0, 0
4     for x = 1, 5, 1
5     do
6         print("Digite o nome: ")
7         nome = io.read("*line")
8         print("H - Homem ou M - Mulher: ")
9         sexo = io.read("*line")
10        if sexo == "H" then
11            h = h + 1
12        elseif sexo == "M" then
13            m = m + 1
14        else
15            print("Sexo so pode ser H ou M!\n")
16        end
17    end
18    print("Foram inseridos "..h.." homens\n")
19    print("Foram inseridos "..m.." mulheres\n")
20 end
21
22 main()

```

#### Listagem 3.50: micro05.java - Lê strings e caracteres

```

1 public class micro05{
2     public static void main(String[] args){
3         String nome, sexo;
4         int x, h, m;
5         h = 0;
6         m = 0;

```

```

7         for( x=1; x <= 5; x++){
8             System.out.print("Digite o nome: ");
9             nome = System.console().readLine();
10            System.out.print("H - Homem ou M - Mulher:");
11            sexo = System.console().readLine();
12            if(sexo == "H"){
13                h = h + 1;
14            }else if(sexo == "M"){
15                m = m + 1;
16            }else{
17                System.out.println("Sexo so pode ser H ou M!");
18            }
19        }
20        System.out.println("Foram inseridos " + h + " Homens");
21        System.out.println("Foram inseridos " + m + " Mulheres");
22    }
23 }

```

#### Listagem 3.51: micro05.smali - Lê strings e caracteres

```

1 .class public Lmicro05;
2 .super Ljava/lang/Object;
3 .source "micro05.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 6
19
20     .prologue
21     const/4 v2, 0x0
22
23     .line 5
24     .line 7
25     const/4 v3, 0x1
26
27     move v0, v2
28
29     move v1, v2
30
31     :goto_4
32     const/4 v2, 0x5
33
34     if-gt v3, v2, :cond_3d
35
36     .line 8
37     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
38
39     const-string v4, "Digite o nome: "

```

```

40
41     invoke-virtual {v2, v4}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
42
43     .line 9
44     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
45
46     move-result-object v2
47
48     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
49
50     .line 10
51     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
52
53     const-string v4, "H - Homem ou M - Mulher:"
54
55     invoke-virtual {v2, v4}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
56
57     .line 11
58     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
59
60     move-result-object v2
61
62     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
63
64     move-result-object v2
65
66     .line 12
67     const-string v4, "H"
68
69     if-ne v2, v4, :cond_2e
70
71     .line 13
72     add-int/lit8 v1, v1, 0x1
73
74     .line 7
75     :goto_2a
76     add-int/lit8 v2, v3, 0x1
77
78     move v3, v2
79
80     goto :goto_4
81
82     .line 14
83     :cond_2e
84     const-string v4, "M"
85
86     if-ne v2, v4, :cond_35
87
88     .line 15
89     add-int/lit8 v0, v0, 0x1
90
91     goto :goto_2a
92
93     .line 17
94     :cond_35
95     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
96

```

## 3.2

```
97     const-string v4, "Sexo so pode ser H ou M!"
98
99     invoke-virtual {v2, v4}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
100
101     goto :goto_2a
102
103     .line 20
104     :cond_3d
105     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
106
107     new-instance v3, Ljava/lang/StringBuilder;
108
109     invoke-direct {v3}, Ljava/lang/StringBuilder;-<init>()V
110
111     const-string v4, "Foram inseridos "
112
113     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
114
115     move-result-object v3
116
117     invoke-virtual {v3, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
118
119     move-result-object v1
120
121     const-string v3, " Homens"
122
123     invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
124
125     move-result-object v1
126
127     invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
128
129     move-result-object v1
130
131     invoke-virtual {v2, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
132
133     .line 21
134     sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
135
136     new-instance v2, Ljava/lang/StringBuilder;
137
138     invoke-direct {v2}, Ljava/lang/StringBuilder;-<init>()V
139
140     const-string v3, "Foram inseridos "
141
142     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
143
144     move-result-object v2
145
146     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
147
```

```

148     move-result-object v0
149
150     const-string v2, "Mulheres"
151
152     invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
153
154     move-result-object v0
155
156     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
157
158     move-result-object v0
159
160     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
161
162     .line 22
163     return-void
164 .end method

```

#### Listagem 3.52: micro06.lua - Escreve um número lido por extenso

```

1 function main()
2     local numero
3     print("Digite um numero de 1 a 5: ")
4     numero = io.read("*number")
5     if numero == 1 then
6         print("Um\n")
7     elseif numero == 2 then
8         print("Dois\n")
9     elseif numero == 3 then
10        print("Tres\n")
11    elseif numero == 4 then
12        print("Quatro\n")
13    elseif numero == 5 then
14        print("Cinco\n")
15    else
16        print("Numero invalido!!!")
17    end
18 end
19
20 main()

```

#### Listagem 3.53: micro06.java - Escreve um número lido por extenso

```

1 public class micro06{
2     public static void main(String[] args){
3         int numero;
4         System.out.print("Digite um numero de 1 a 5: ");
5         numero = Integer.parseInt(System.console().readLine());
6         if(numero == 1){
7             System.out.println("Um");
8         }else if(numero == 2){
9             System.out.println("Dois");
10        }else if(numero == 3){
11            System.out.println("Tres");
12        }else if(numero == 4){
13            System.out.println("Quatro");

```

```

14         }else if(numero == 5){
15             System.out.println("Cinco");
16         }else{
17             System.out.println("Numero invalido!!!");
18         }
19     }
20 }

```

Listagem 3.54: micro06.smali - Escreve um número lido por extenso

```

1  .class public Lmicro06;
2  .super Ljava/lang/Object;
3  .source "micro06.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um numero de 1 a 5: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(Ljava/lang/
        String;)V
27
28     .line 5
29     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
30
31     move-result-object v0
32
33     invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;
34
35     move-result-object v0
36
37     invoke-static {v0}, Ljava/lang/Integer;-->parseInt(Ljava/lang/String;)I
38
39     move-result v0
40
41     .line 6
42     const/4 v1, 0x1
43
44     if-ne v0, v1, :cond_1e
45
46     .line 7
47     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
48

```

```

49     const-string v1, "Um"
50
51     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
52
53     .line 19
54     :goto_ld
55     return-void
56
57     .line 8
58     :cond_1e
59     const/4 v1, 0x2
60
61     if-ne v0, v1, :cond_29
62
63     .line 9
64     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66     const-string v1, "Dois"
67
68     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
69
70     goto :goto_ld
71
72     .line 10
73     :cond_29
74     const/4 v1, 0x3
75
76     if-ne v0, v1, :cond_34
77
78     .line 11
79     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
80
81     const-string v1, "Tres"
82
83     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
84
85     goto :goto_ld
86
87     .line 12
88     :cond_34
89     const/4 v1, 0x4
90
91     if-ne v0, v1, :cond_3f
92
93     .line 13
94     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
95
96     const-string v1, "Quatro"
97
98     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
99
100    goto :goto_ld
101
102    .line 14
103    :cond_3f

```



```

104     const/4 v1, 0x5
105
106     if-ne v0, v1, :cond_4a
107
108     .line 15
109     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
110
111     const-string v1, "Cinco"
112
113     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
114
115     goto :goto_1d
116
117     .line 17
118     :cond_4a
119     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
120
121     const-string v1, "Numero invalido!!!"
122
123     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
124
125     goto :goto_1d
126 .end method

```

#### Listagem 3.55: micro07.lua - Decide se os números são positivos, zeros ou negativos

```

1 function main()
2     local programa, numero, opc
3     programa = 1
4     while programa == 1 do
5         print("Digite um numero: ")
6         numero = io.read("*n")
7         if numero > 0 then
8             print("Positivo\n")
9         else
10            if numero == 0 then
11                print("0 numero e igual a 0\n")
12            end
13            if numero < 0 then
14                print("Negativo\n")
15            end
16        end
17        print("Deseja finalizar? (S - 1): ")
18        opc = io.read("*n")
19        if opc == 1 then
20            programa = 0
21        end
22    end
23 end
24
25 main()

```

#### Listagem 3.56: micro07.java - Decide se os números são positivos, zeros ou negativos

```

1 public class micro07{
2     public static void main(String[] args){
3         int programa, numero, opc;

```

```

4
5     programa = 1;
6     while(programa == 1){
7         System.out.print("Digite um numero: ");
8         numero = Integer.parseInt(System.console().readLine());
9         if(numero > 0){
10            System.out.println("Positivo");
11        }else{
12            if(numero == 0){
13                System.out.println("O numero e igual a 0");
14            }
15            if(numero < 0){
16                System.out.println("Negativo");
17            }
18        }
19        System.out.print("Deseja finalizar? (S -1) ");
20        opc = Integer.parseInt(System.console().readLine());
21        if(opc == 1){
22            programa = 0;
23        }
24    }
25 }
26 }
27 }

```

Listagem 3.57: micro07.smali - Decide se os números são positivos, zeros ou negativos

```

1 .class public Lmicro07;
2 .super Ljava/lang/Object;
3 .source "micro07.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 6
19
20     .prologue
21     const/4 v1, 0x1
22
23     .line 5
24     move v0, v1
25
26     .line 6
27     :cond_2
28     :goto_2
29     if-ne v0, v1, :cond_4a
30
31     .line 7
32     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;

```

## 3.2

```
33
34     const-string v3, "Digite um numero: "
35
36     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
37
38     .line 8
39     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
40
41     move-result-object v2
42
43     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
44
45     move-result-object v2
46
47     invoke-static {v2}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
48
49     move-result v2
50
51     .line 9
52     if-lez v2, :cond_37
53
54     .line 10
55     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
56
57     const-string v3, "Positivo"
58
59     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
60
61     .line 19
62     :cond_20
63     :goto_20
64     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66     const-string v3, "Deseja finalizar? (S -1) "
67
68     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
69
70     .line 20
71     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
72
73     move-result-object v2
74
75     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
76
77     move-result-object v2
78
79     invoke-static {v2}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
80
81     move-result v2
82
83     .line 21
84     if-ne v2, v1, :cond_2
85
86     .line 22
87     const/4 v0, 0x0
88
```

```

89     goto :goto_2
90
91     .line 12
92     :cond_37
93     if-nez v2, :cond_40
94
95     .line 13
96     sget-object v3, Ljava/lang/System; ->out:Ljava/io/PrintStream;
97
98     const-string v4, "O numero e igual a 0"
99
100    invoke-virtual {v3, v4}, Ljava/io/PrintStream; ->println(Ljava/lang/
        String;)V
101
102    .line 15
103    :cond_40
104    if-gez v2, :cond_20
105
106    .line 16
107    sget-object v2, Ljava/lang/System; ->out:Ljava/io/PrintStream;
108
109    const-string v3, "Negativo"
110
111    invoke-virtual {v2, v3}, Ljava/io/PrintStream; ->println(Ljava/lang/
        String;)V
112
113    goto :goto_20
114
115    .line 26
116    :cond_4a
117    return-void
118 .end method

```

#### Listagem 3.58: micro08.lua - Decide se um número é maior ou menor que 10

```

1 function main()
2     local numero
3     numero = 1
4     while numero ~= 0 do
5         print("Digite um numero: ")
6         numero = io.read("*n")
7         if numero > 10 then
8             print("O numero "..numero.." e maior que 10\n")
9         else
10            print("O numero "..numero.." e menor que 10\n")
11        end
12    end
13 end
14
15 main()

```

#### Listagem 3.59: micro08.java - Decide se um número é maior ou menor que 10

```

1 public class micro08{
2     public static void main(String[] args){
3         int numero;
4         numero = 1;
5         while (numero != 0){
6             System.out.print("Digite um numero: ");

```

## 3.2

```
7         numero = Integer.parseInt(System.console().readLine());
8         if(numero > 10){
9             System.out.println("O numero " + numero + " e maior que 10
10                ");
11         }else{
12             System.out.println("O numero " + numero + " e menor que 10
13                ");
14         }
15 }
```

Listagem 3.60: micro08.smali - Decide se um número é maior ou menor que 10

```
1 .class public Lmicro08;
2 .super Ljava/lang/Object;
3 .source "micro08.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 5
25     :goto_1
26     if-eqz v0, :cond_58
27
28     .line 6
29     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
30
31     const-string v1, "Digite um numero: "
32
33     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(Ljava/lang/
34         String;)V
35
36     .line 7
37     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
38
39     move-result-object v0
40
41     invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;
42
43     move-result-object v0
44
45     invoke-static {v0}, Ljava/lang/Integer;-->parseInt(Ljava/lang/String;)I
```

```

45
46     move-result v0
47
48     .line 8
49     const/16 v1, 0xa
50
51     if-le v0, v1, :cond_39
52
53     .line 9
54     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
55
56     new-instance v2, Ljava/lang/StringBuilder;
57
58     invoke-direct {v2}, Ljava/lang/StringBuilder;--<init>()V
59
60     const-string v3, "O numero "
61
62     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
63
64     move-result-object v2
65
66     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;-->append(I)Ljava/
        lang/StringBuilder;
67
68     move-result-object v2
69
70     const-string v3, " e maior que 10"
71
72     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
73
74     move-result-object v2
75
76     invoke-virtual {v2}, Ljava/lang/StringBuilder;-->toString()Ljava/lang/
        String;
77
78     move-result-object v2
79
80     invoke-virtual {v1, v2}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
81
82     goto :goto_1
83
84     .line 11
85     :cond_39
86     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
87
88     new-instance v2, Ljava/lang/StringBuilder;
89
90     invoke-direct {v2}, Ljava/lang/StringBuilder;--<init>()V
91
92     const-string v3, "O numero "
93
94     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
95
96     move-result-object v2
97

```

## 3.2

```
98     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
        lang/StringBuilder;
99
100    move-result-object v2
101
102    const-string v3, " e menor que 10"
103
104    invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
105
106    move-result-object v2
107
108    invoke-virtual {v2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
109
110    move-result-object v2
111
112    invoke-virtual {v1, v2}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
113
114    goto :goto_1
115
116    .line 14
117    :cond_58
118    return-void
119 .end method
```

---

### Listagem 3.61: micro09.lua - Cálculo de preços

```
1 function main()
2     local preco, venda, novo_preco
3     print("Digite o preco: ")
4     preco = io.read("*n")
5     print("Digite a venda: ")
6     venda = io.read("*n")
7     if venda < 500 or preco < 30 then
8         novo_preco = preco + 10 / 100 * preco
9     elseif (venda >= 500 and venda <= 1200) or (preco >=30 and preco <80)
        then
10         novo_preco = preco + 15 / 100 * preco
11     elseif venda >= 1200 or preco >= 80 then
12         novo_preco = preco - 20 / 100 * preco
13     end
14     print("O novo preco e "..novo_preco.."\\n")
15 end
16
17 main()
```

---

### Listagem 3.62: micro09.java - Cálculo de preços

```
1 public class micro09{
2     public static void main(String[] args){
3         double preco, venda, novo_preco;
4         novo_preco = 0;
5         System.out.print("Digite o preco: ");
6         preco = Double.parseDouble(System.console().readLine());
7         System.out.print("Digite a venda: ");
8         venda = Double.parseDouble(System.console().readLine());
9         if((venda < 500) || (preco < 30)){
```

```

10         novo_preco = preco + (10/100) * preco;
11     }else if(((venda >= 500) && venda < 1200) || ((preco >= 30) &&
        preco >= 80)){
12         novo_preco = preco + (15/100) * preco;
13     }else if ((venda >= 1200) || (preco >=80)){
14         novo_preco = preco - (20/100) * preco;
15     }
16     System.out.println("O novo preco e " + novo_preco);
17 }
18 }

```

### Listagem 3.63: micro09.smali - Cálculo de preços

```

1 .class public Lmicro09;
2 .super Ljava/lang/Object;
3 .source "micro09.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 9
19
20     .prologue
21     .line 4
22     const-wide/16 v0, 0x0
23
24     .line 5
25     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27     const-string v3, "Digite o preco: "
28
29     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(Ljava/lang/
        String;)V
30
31     .line 6
32     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
33
34     move-result-object v2
35
36     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
37
38     move-result-object v2
39
40     invoke-static {v2}, Ljava/lang/Double;->parseDouble(Ljava/lang/String
        ;)D
41
42     move-result-wide v2
43
44     .line 7

```



## 3.2

```
45     sget-object v4, Ljava/lang/System;-->out:Ljava/io/PrintStream;
46
47     const-string v5, "Digite a venda: "
48
49     invoke-virtual {v4, v5}, Ljava/io/PrintStream;-->print(Ljava/lang/
        String;)V
50
51     .line 8
52     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
53
54     move-result-object v4
55
56     invoke-virtual {v4}, Ljava/io/Console;-->readLine()Ljava/lang/String;
57
58     move-result-object v4
59
60     invoke-static {v4}, Ljava/lang/Double;-->parseDouble(Ljava/lang/String
        ;)D
61
62     move-result-wide v4
63
64     .line 9
65     const-wide v6, 0x407f400000000000L    # 500.0
66
67     cmpg-double v6, v4, v6
68
69     if-ltz v6, :cond_37
70
71     const-wide/high16 v6, 0x403e000000000000L    # 30.0
72
73     cmpg-double v6, v2, v6
74
75     if-gez v6, :cond_54
76
77     .line 10
78     :cond_37
79     const-wide/16 v0, 0x0
80
81     mul-double/2addr v0, v2
82
83     add-double/2addr v0, v2
84
85     .line 16
86     :cond_3b
87     :goto_3b
88     sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
89
90     new-instance v3, Ljava/lang/StringBuilder;
91
92     invoke-direct {v3}, Ljava/lang/StringBuilder;--><init>()V
93
94     const-string v4, "O novo preco e "
95
96     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;-->append(Ljava/lang/
        String;)Ljava/lang/StringBuilder;
97
98     move-result-object v3
99
100    invoke-virtual {v3, v0, v1}, Ljava/lang/StringBuilder;-->append(D)Ljava
```

```

        /lang/StringBuilder;
101
102     move-result-object v0
103
104     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
        String;
105
106     move-result-object v0
107
108     invoke-virtual {v2, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
109
110     .line 17
111     return-void
112
113     .line 11
114     :cond_54
115     const-wide v6, 0x407f400000000000L      # 500.0
116
117     cmpl-double v6, v4, v6
118
119     if-ltz v6, :cond_66
120
121     const-wide v6, 0x4092c00000000000L      # 1200.0
122
123     cmpg-double v6, v4, v6
124
125     if-ltz v6, :cond_72
126
127     :cond_66
128     const-wide/high16 v6, 0x403e000000000000L      # 30.0
129
130     cmpl-double v6, v2, v6
131
132     if-ltz v6, :cond_77
133
134     const-wide/high16 v6, 0x4054000000000000L      # 80.0
135
136     cmpl-double v6, v2, v6
137
138     if-ltz v6, :cond_77
139
140     .line 12
141     :cond_72
142     const-wide/16 v0, 0x0
143
144     mul-double/2addr v0, v2
145
146     add-double/2addr v0, v2
147
148     goto :goto_3b
149
150     .line 13
151     :cond_77
152     const-wide v6, 0x4092c00000000000L      # 1200.0
153
154     cmpl-double v4, v4, v6
155
156     if-gez v4, :cond_86

```

## 3.2

```
157
158     const-wide/high16 v4, 0x4054000000000000L    # 80.0
159
160     cmpl-double v4, v2, v4
161
162     if-ltz v4, :cond_3b
163
164     .line 14
165     :cond_86
166     const-wide/16 v0, 0x0
167
168     mul-double/2addr v0, v2
169
170     sub-double v0, v2, v0
171
172     goto :goto_3b
173 .end method
```

---

### Listagem 3.64: micro10.lua - Calcula o fatorial de um número

```
1 function main()
2     local numero, fat
3     print("Digite um numero: ")
4     numero = io.read("*n")
5     fat = fatorial(numero)
6
7     print("O fatorial de "..numero.." e "..fat.."\\n")
8 end
9
10 function fatorial(n)
11     if n <= 0 then
12         return 1
13     else
14         return n * fatorial(n-1)
15     end
16 end
17
18 main()
```

---

### Listagem 3.65: micro10.java - Calcula o fatorial de um número

```
1 public class micro10{
2     public static void main(String[] args){
3         int numero, fat;
4         System.out.print("Digite o numero: ");
5         numero = Integer.parseInt(System.console().readLine());
6         fat = fatorial(numero);
7         System.out.print("O fatorial de ");
8         System.out.print(numero);
9         System.out.print(" e ");
10        System.out.print(fat);
11    }
12    public static int fatorial(int n){
13        if(n <= 0){
14            return 1;
15        }else{
16            return (n * fatorial(n-1));
17        }
18    }
```

```

19     }
20 }

```

### Listagem 3.66: micro10.smali - Calcula o fatorial de um número

```

1 .class public Lmicro10;
2 .super Ljava/lang/Object;
3 .source "micro10.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -> <init>()V
13
14    return-void
15 .end method
16
17 .method public static fatorial(I)I
18     .registers 2
19
20    .prologue
21    .line 13
22    if-gtz p0, :cond_4
23
24    .line 14
25    const/4 v0, 0x1
26
27    .line 16
28    :goto_3
29    return v0
30
31    :cond_4
32    add-int/lit8 v0, p0, -0x1
33
34    invoke-static {v0}, Lmicro10; -> fatorial(I)I
35
36    move-result v0
37
38    mul-int/2addr v0, p0
39
40    goto :goto_3
41 .end method
42
43 .method public static main([Ljava/lang/String;)V
44     .registers 5
45
46    .prologue
47    .line 4
48    sget-object v0, Ljava/lang/System; -> out:Ljava/io/PrintStream;
49
50    const-string v1, "Digite o numero: "
51
52    invoke-virtual {v0, v1}, Ljava/io/PrintStream; -> print(Ljava/lang/
        String;)V
53

```

## 3.2

```
54 .line 5
55 invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
56
57 move-result-object v0
58
59 invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;
60
61 move-result-object v0
62
63 invoke-static {v0}, Ljava/lang/Integer;-->parseInt(Ljava/lang/String;)I
64
65 move-result v0
66
67 .line 6
68 invoke-static {v0}, Lmicro10;-->fatorial(I)I
69
70 move-result v1
71
72 .line 7
73 sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
74
75 const-string v3, "O fatorial de "
76
77 invoke-virtual {v2, v3}, Ljava/io/PrintStream;-->print(Ljava/lang/
    String;)V
78
79 .line 8
80 sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
81
82 invoke-virtual {v2, v0}, Ljava/io/PrintStream;-->print(I)V
83
84 .line 9
85 sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
86
87 const-string v2, " e "
88
89 invoke-virtual {v0, v2}, Ljava/io/PrintStream;-->print(Ljava/lang/
    String;)V
90
91 .line 10
92 sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
93
94 invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(I)V
95
96 .line 11
97 return-void
98 .end method
```

Listagem 3.67: micro11.lua - Decide se um número é positivo, zero ou negativo com auxílio de uma função

```
1 function main()
2     local numero, x
3     print("Digite um numero ")
4     numero = io.read("*n")
5     x = verifica(numero)
6     if x == 1 then
7         print("Numero positivo\n")
8     elseif x == 0 then
```

```

9      print("Zero\n")
10     else
11         print("Numero negativo\n")
12     end
13 end
14
15 function verifica(n)
16     local res
17     if n > 0 then
18         res = 1
19     elseif n < 0 then
20         res = -1
21     else
22         res = 0
23     end
24     return res
25 end
26
27 main()

```

Listagem 3.68: micro11.java - Decide se um número é positivo, zero ou negativo com auxílio de uma função

```

1 public class micro11{
2     public static void main(String[] args){
3         int numero, x;
4         System.out.print("Digite um numero: ");
5         numero = Integer.parseInt(System.console().readLine());
6         x = verifica(numero);
7         if(x == 1){
8             System.out.println("Numero positivo");
9         }else if(x == 0){
10            System.out.println("Zero");
11        }else{
12            System.out.println("Numero negativo");
13        }
14    }
15    public static int verifica(int n){
16        int res;
17        if(n > 0){
18            res = 1;
19        }else if (n < 0){
20            res = -1;
21        }else{
22            res = 0;
23        }
24        return res;
25    }
26 }

```

Listagem 3.69: micro11.smali - Decide se um número é positivo, zero ou negativo com auxílio de uma função

```

1 .class public Lmicro11;
2 .super Ljava/lang/Object;
3 .source "micro11.java"
4
5
6 # direct methods

```

## 3.2

```
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um numero: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->print(Ljava/lang/
        String;)V
27
28     .line 5
29     invoke-static {}, Ljava/lang/System;-->console()Ljava/io/Console;
30
31     move-result-object v0
32
33     invoke-virtual {v0}, Ljava/io/Console;-->readLine()Ljava/lang/String;
34
35     move-result-object v0
36
37     invoke-static {v0}, Ljava/lang/Integer;-->parseInt(Ljava/lang/String;)I
38
39     move-result v0
40
41     .line 6
42     invoke-static {v0}, Lmicroll;-->verifica(I)I
43
44     move-result v0
45
46     .line 7
47     const/4 v1, 0x1
48
49     if-ne v0, v1, :cond_22
50
51     .line 8
52     sget-object v0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
53
54     const-string v1, "Numero positivo"
55
56     invoke-virtual {v0, v1}, Ljava/io/PrintStream;-->println(Ljava/lang/
        String;)V
57
58     .line 14
59     :goto_21
60     return-void
61
62     .line 9
63     :cond_22
```

```

64     if-nez v0, :cond_2c
65
66     .line 10
67     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
68
69     const-string v1, "Zero"
70
71     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
72
73     goto :goto_21
74
75     .line 12
76     :cond_2c
77     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
78
79     const-string v1, "Numero negativo"
80
81     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
82
83     goto :goto_21
84 .end method
85
86 .method public static verifica(I)I
87     .registers 2
88
89     .prologue
90     .line 17
91     if-lez p0, :cond_4
92
93     .line 18
94     const/4 v0, 0x1
95
96     .line 24
97     :goto_3
98     return v0
99
100    .line 19
101    :cond_4
102    if-gez p0, :cond_8
103
104    .line 20
105    const/4 v0, -0x1
106
107    goto :goto_3
108
109    .line 22
110    :cond_8
111    const/4 v0, 0x0
112
113    goto :goto_3
114 .end method

```

---



# Capítulo 4

## Analizador léxico

Neste capítulo teremos a implementação de um analisador léxico para a linguagem MiniLua. Ela será feita com a linguagem Ocaml, que possui uma ferramenta bastante útil para tal fim, a Ocamllex.

Essa etapa é a primeira no processo de compilação. Sua função é facilitar o trabalho da próxima etapa, a análise sintática, lendo o código e convertendo-o em uma sequência de *tokens*, que são uma espécie de identificadores da estrutura de um programa. Dessa forma, temos *tokens* para simbolizar palavras reservadas da linguagem, variáveis (e seus nomes), valores literais e seus tipos, fim de linha e outros caracteres parte da linguagem. Além disso, o analisador léxico já elimina elementos decorativos, como comentários e espaços em branco.

### 4.1 Reconhecendo os Tokens

Seguem os *tokens* criados para a linguagem MiniLua na tabela

**Tabela 4.1:** *Tokens*

Tipo	Representação	Tipo	Representação
AND	<i>and</i>	BREAK	<i>break</i>
DO	<i>do</i>	ELSE	<i>else</i>
ELSEIF	<i>elseif</i>	EOF	<i>representa fim do arquivo</i>
END	<i>end</i>	FALSE	<i>false</i>
FOR	<i>for</i>	FUNCAO	<i>function</i>
IF	<i>if</i>	IN	<i>in</i>
IO_READ	<i>io.read</i>	LOCAL	<i>local</i>
NIL	<i>nil</i>	NOT	<i>not</i>
NUMBER_INPUT	<i>*number</i>	OR	<i>or</i>
PRINT	<i>print</i>	REPEAT	<i>repeat</i>
RETURN	<i>return</i>	THEN	<i>then</i>
TRUE	<i>true</i>	UNTIL	<i>until</i>
WHILE	<i>while</i>	ABRE_CHAVE	<i>{</i>

Continued on next page

Tabela 4.1 – continued from previous page

Tipo	Representação	Tipo	Representação
ABRE_COLCHETE	/	ADICAO	+
AND_BINARIO	&	APAR	(
ATRIB	=	CONCATENA	..
DIV_POR_2	»	DIVISAO	/
DIVISAO_INTEIRO	//	DOIS_PONTOS	:
EQUIVALENTE	==	EXPONENCIACAO	**
FECHA_CHAVE	}	FECHA_COLCHETE	/
FPAR	)	MAIOR	>
MAIOR_OU_IGUAL	>=	MENOR	<
MENOR_OU_IGUAL	<=	MODULO	%
MULTIPLICACAO	*	MULT_POR_2	«
NAO_EQUIVALENTE	!=	OR_BINARIO	/
PONTO	.	PONTO_VIRGULA	;
RETICENCIAS	...	SUBTRACAO	-
TAMANHO	#	VIRGULA	,
LITINT of int	<i>dígitos</i>	LITSTRING of string	<i>"expressao"</i>
ID of string	<i>ex: variavel_nome</i>		

## 4.2 Montagem do Analisador Léxico

### 4.2.1 Código do Analisador Léxico

Segue o código do analisador léxico para a linguagem MiniLua. Um sistema de anotação de tipos (inexistente na linguagem atualmente) próprio deste projeto será adicionado futuramente para facilitar o desenvolvimento do restante do compilador.

Listagem 4.1: lexico.mll - Gera tokens dado código da linguagem LUA

```

1 {
2   open Lexing
3   open Printf
4
5   let incr_num_linha lexbuf =
6     let pos = lexbuf.lex_curr_p in
7     lexbuf.lex_curr_p <- { pos with
8       pos_lnum = pos.pos_lnum + 1;
9       pos_bol = pos.pos_cnum;
10    }
11
12   let msg_erro lexbuf c =
13     let pos = lexbuf.lex_curr_p in
14     let lin = pos.pos_lnum
15     and col = pos.pos_cnum - pos.pos_bol - 1 in
16     sprintf "%d-%d: caracter desconhecido %c" lin col c
17
18   let erro lin col msg =

```

## 4.2

```
19     let mensagem = sprintf "%d-%d: %s" lin col msg in
20         failwith mensagem
21
22 type tokens = ABRE_CHAVE
23             | ABRE_COLCHETE
24             | ADICAO
25             | AND
26             | AND_BINARIO
27             | APAR
28             | ATRIB
29             | BREAK
30             | CONCATENA
31             | DIV_POR_2
32             | DIVISAO
33             | DIVISAO_INTEIRO
34             | DO
35             | DOIS_PONTOS
36             | ELSE
37             | ELSEIF
38             | END
39             | EQUIVALENTE
40             | EXPONENCIACAO
41             | FALSE
42             | FECHA_CHAVE
43             | FECHA_COLCHETE
44             | FOR
45             | FPAR
46             | FUNCAO
47             | IF
48             | IN
49             | IO_READ
50             | LOCAL
51             | MAIOR
52             | MAIOR_OU_IGUAL
53             | MENOR
54             | MENOR_OU_IGUAL
55             | MODULO
56             | MULT_POR_2
57             | MULTIPLICACAO
58             | NAO_EQUIVALENTE
59             | NIL
60             | NOT
61             | NUMBER_INPUT
62             | OR
63             | OR_BINARIO
64             | PONTO
65             | PONTO_VIRGULA
66             | PRINT
67             | REPEAT
68             | RETICENCIAS
69             | RETURN
70             | SUBTRACAO
71             | TAMANHO
72             | THEN
73             | TRUE
74             | UNTIL
75             | VIRGULA
76             | WHILE
77             | LITINT of int
```

```

78         | LITSTRING of string
79         | ID of string
80         | EOF
81     }
82
83     let digito = ['0' - '9']
84     let inteiro = digito+
85
86     let letra = ['a' - 'z' 'A' - 'Z']
87     let identificador = letra ( letra | digito | '_' ) *
88
89     let brancos = [' ' '\t'] +
90     let novalinha = '\r' | '\n' | "\r\n"
91
92     rule token = parse
93     brancos                { token lexbuf }
94 | novalinha                { incr_num_linha lexbuf; token lexbuf }
95 | "--[" {
96         let pos = lexbuf.lex_curr_p in
97         let lin = pos.pos_lnum
98         and col = pos.pos_cnum - pos.pos_bol - 1 in
99         comentario_bloco lin col lexbuf }
100 | "--" { comentario_linha lexbuf }
101 | "(" { APAR }
102 | "{" { ABRE_CHAVE }
103 | "[" { ABRE_COLCHETE }
104 | "+" { ADICAO }
105 | "-" { SUBTRACAO }
106 | ")" { FPAR }
107 | "}" { FECHA_CHAVE }
108 | "]" { FECHA_COLCHETE }
109 | "," { VIRGULA }
110 | "." { PONTO }
111 | ";" { PONTO_VIRGULA }
112 | ":" { DOIS_PONTOS }
113 | "==" { EQUIVALENTE }
114 | "~=" { NAO_EQUIVALENTE }
115 | ">=" { MAIOR_OU_IGUAL }
116 | "<=" { MENOR_OU_IGUAL }
117 | "/" { DIVISAO }
118 | "*" { MULTIPLICACAO }
119 | "%" { MODULO }
120 | "^" { EXPONENCIACAO }
121 | ">" { MAIOR }
122 | "<" { MENOR }
123 | "=" { ATRIB }
124 | "#" { TAMANHO }
125 | "<<" { MULT_POR_2 }
126 | ">>" { DIV_POR_2 }
127 | "//" { DIVISAO_INTEIRO }
128 | "&" { AND_BINARIO }
129 | "|" { OR_BINARIO }
130 | ".." { CONCATENA }
131 | "... " { RETICENCIAS }
132 | "and" { AND }
133 | "break" { BREAK }
134 | "do" { DO }
135 | "else" { ELSE }
136 | "elseif" { ELSEIF }
137 | "end" { END }

```

```

137 | "false"           { FALSE }
138 | "for"             { FOR }
139 | "function"        { FUNCAO }
140 | "if"              { IF }
141 | "io.read"         { IO_READ }
142 | "in"              { IN }
143 | "local"           { LOCAL }
144 | "nil"             { NIL }
145 | "not"             { NOT }
146 | "print"           { PRINT }
147 | "or"              { OR }
148 | "repeat"          { REPEAT }
149 | "return"          { RETURN }
150 | "then"            { THEN }
151 | "true"            { TRUE }
152 | "until"           { UNTIL }
153 | "while"           { WHILE }
154 | inteiro as num    { let numero = int_of_string num in
155 |                   LITINT numero }
156 | identificador as id { ID id }
157 | '"'              { let pos = lexbuf.lex_curr_p in
158 |                   let lin = pos.pos_lnum and col = pos.pos_cnum -
159 |                       pos.pos_bol - 1 in
160 |                       let buffer = Buffer.create 1 in
161 |                       let str = leia_string lin col buffer lexbuf in
162 |                       LITSTRING str }
162 | _ as c           { failwith (msg_erro lexbuf c) }
163 | eof               { EOF }
164
165 and comentario_bloco lin col = parse
166   "--]]"           { token lexbuf }
167 | novalinha { incr_num_linha lexbuf; comentario_bloco lin col lexbuf }
168 | _         { comentario_bloco lin col lexbuf }
169 | eof       { erro lin col "Comentario nao fechado" }
170
171 and leia_string lin col buffer = parse
172   '"' { Buffer.contents buffer}
173 | "\\t" { Buffer.add_char buffer '\t'; leia_string lin col buffer
174 |       lexbuf }
174 | "\\n" { Buffer.add_char buffer '\n'; leia_string lin col buffer
175 |       lexbuf }
175 | '\\ ' '"' { Buffer.add_char buffer '"'; leia_string lin col buffer
176 |       lexbuf }
176 | '\\ ' '\\ ' { Buffer.add_char buffer '\\'; leia_string lin col buffer
177 |       lexbuf }
177 | novalinha {erro lin col "A string nao foi fechada"}
178 | _ as c    { Buffer.add_char buffer c; leia_string lin col buffer lexbuf
179 |           }
179 | eof      { erro lin col "A string nao foi fechada"}
180
181 and comentario_linha = parse
182   novalinha {incr_num_linha lexbuf; token lexbuf}
183 | _         {comentario_linha lexbuf}

```

---

Além do analisador léxico, usaremos um outro programa chamado *carregador.ml*. A função desse programa é automatizar o processo de passar os códigos do arquivo em lua para o programa lexico compilado no formato .cmo, pois nosso analisador léxico por si só não lê o arquivo como um todo, mas palavra por palavra no arquivo de entrada (palavra aqui no

sentido de palavra da linguagem regular que forma a linguagem de programação LUA do arquivo). Segue o código usado pelo *carregador.ml*

Listagem 4.2: carregador.ml - Programa auxiliar para o analisador léxico

```

1 #load "lexico.cmo";;
2
3 let rec tokens lexbuf =
4   let tok = Lexico.token lexbuf in
5   match tok with
6   | Lexico.EOF -> [Lexico.EOF]
7   | _ -> tok :: tokens lexbuf
8 ;;
9
10 let lexico str =
11   let lexbuf = Lexing.from_string str in
12   tokens lexbuf
13 ;;
14
15 let lex arq =
16   let ic = open_in arq in
17   let lexbuf = Lexing.from_channel ic in
18   let toks = tokens lexbuf in
19   let _ = close_in ic in
20   toks

```

### 4.2.2 Utilização

O código descrito na listagem 4.1 acima nada mais é que um conjunto de expressões regulares e instruções para podermos montar um autômato finito determinístico, usado para fazer o reconhecimento da linguagem regular dos *tokens*. Para criarmos tal autômato, utilizamos de uma extensão do ocaml, chamada ocamllex. Essa ferramenta faz essa tradução das expressões regulares no arquivo .mll para o autômato finito determinístico em um arquivo de saída .ml que pode, então, ser utilizado para produzir os tokens de um arquivo em Lua.

Começamos traduzindo o arquivo .mll para o arquivo .ml por meio do ocamllex rodando o código a seguir no local onde o arquivo se encontra

```
$ ocamllex lexico.mll
```

Isso gerará o arquivo .ml que deverá ser, então, compilado para produzir os formatos .cmi e .cmo da seguinte forma

```
$ ocamlc -c lexico.ml
```

O arquivo *lexico.cmo* gerado após a compilação será usado por nosso programa auxiliar *carregador.ml* 4.2. Para fazermos isso, entramos no interpretador do ocaml por meio do seguinte código

```
$ rlwrap ocaml
```

Para então importarmos o *carregador.ml* da seguinte maneira

```
# #use "carregador.ml";;
```

Agora podemos utilizar a funcionalidade em *carregador.ml*. Faremos isso chamando a função *lex* e passando como argumento o arquivo *.lua* a ser traduzido em tokens pelo analisador léxico.

```
# lex "arquivo.lua";;
```

Onde *arquivo.lua* é o nome do arquivo de onde queremos obter os tokens. Este código terá como resultado a lista de tokens do arquivo.

## 4.3 Testando o Analisador Léxico

A seguir serão apresentadas as saídas obtidas ao passarmos nossos programas nano e micro pelo nosso analisador léxico.

### 4.3.1 Programas Nano

Os resultados de passar os programas nano pelo analisador são os seguintes

#### nano01

Listagem 4.3: Resultado de passar o analisador léxico no programa nano01.lua

```
1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.END;
3 Lexico.EOF]
```

#### nano02

Listagem 4.4: Resultado de passar o analisador léxico no programa nano02.lua

```
1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.END; Lexico.EOF]
```

#### nano03

Listagem 4.5: Resultado de passar o analisador léxico no programa nano03.lua

```
1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.END;
4 Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]
```

#### nano04

Listagem 4.6: Resultado de passar o analisador léxico no programa nano04.lua

```
1 - : Lexico.tokens list =
```

```

2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.
  ADICAO;
4 Lexico.LITINT 2; Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
5 Lexico.EOF]

```

---

## nano05

Listagem 4.7: Resultado de passar o analisador léxico no programa nano05.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 2; Lexico.PRINT
  ;
4 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.END; Lexico.ID "main";
5 Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## nano06

Listagem 4.8: Resultado de passar o analisador léxico no programa nano06.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;
4 Lexico.SUBTRACAO; Lexico.LITINT 2; Lexico.PRINT; Lexico.APAR; Lexico.ID "
  n";
5 Lexico.FPAR; Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
6 Lexico.EOF]

```

---

## nano07

Listagem 4.9: Resultado de passar o analisador léxico no programa nano07.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.IF;
4 Lexico.ID "n"; Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.THEN;
5 Lexico.PRINT; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.END;
6 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## nano08

Listagem 4.10: Resultado de passar o analisador léxico no programa nano08.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.IF;
4 Lexico.ID "n"; Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.THEN;
5 Lexico.PRINT; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.ELSE;
6 Lexico.PRINT; Lexico.APAR; Lexico.LITINT 0; Lexico.FPAR; Lexico.END;
7 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## nano09



Listagem 4.11: Resultado de passar o analisador léxico no programa nano09.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.
  ADICAO;
4 Lexico.LITINT 1; Lexico.DIVISAO; Lexico.LITINT 2; Lexico.IF; Lexico.ID "n
  ";
5 Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.THEN; Lexico.PRINT; Lexico.
  APAR;
6 Lexico.ID "n"; Lexico.FPAR; Lexico.ELSE; Lexico.PRINT; Lexico.APAR;
7 Lexico.LITINT 0; Lexico.FPAR; Lexico.END; Lexico.END; Lexico.ID "main";
8 Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

## nano10

Listagem 4.12: Resultado de passar o analisador léxico no programa nano10.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.ID "n"; Lexico.
  VIRGULA;
4 Lexico.ID "m"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.VIRGULA;
5 Lexico.LITINT 2; Lexico.IF; Lexico.ID "n"; Lexico.EQUIVALENTE;
6 Lexico.ID "m"; Lexico.THEN; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
7 Lexico.FPAR; Lexico.ELSE; Lexico.PRINT; Lexico.APAR; Lexico.LITINT 0;
8 Lexico.FPAR; Lexico.END; Lexico.END; Lexico.ID "main"; Lexico.APAR;
9 Lexico.FPAR; Lexico.EOF]

```

## nano11

Listagem 4.13: Resultado de passar o analisador léxico no programa nano11.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.VIRGULA; Lexico.ID "
  x";
4 Lexico.ID "n"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.VIRGULA; Lexico.ID "
  x";
5 Lexico.ATRIB; Lexico.LITINT 1; Lexico.VIRGULA; Lexico.LITINT 2;
6 Lexico.VIRGULA; Lexico.LITINT 5; Lexico.WHILE; Lexico.ID "x"; Lexico.
  MAIOR;
7 Lexico.ID "n"; Lexico.DO; Lexico.ID "n"; Lexico.ATRIB; Lexico.ID "n";
8 Lexico.ADICAO; Lexico.ID "m"; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
9 Lexico.FPAR; Lexico.END; Lexico.END; Lexico.ID "main"; Lexico.APAR;
10 Lexico.FPAR; Lexico.EOF]

```

## nano12

Listagem 4.14: Resultado de passar o analisador léxico no programa nano12.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "n"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.VIRGULA; Lexico.ID "
  x";
4 Lexico.ID "n"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.VIRGULA; Lexico.ID "
  x";

```

```

5 Lexico.ATRIB; Lexico.LITINT 1; Lexico.VIRGULA; Lexico.LITINT 2;
6 Lexico.VIRGULA; Lexico.LITINT 5; Lexico.WHILE; Lexico.ID "x"; Lexico.
  MAIOR;
7 Lexico.ID "n"; Lexico.DO; Lexico.IF; Lexico.ID "n"; Lexico.EQUIVALENTE;
8 Lexico.ID "m"; Lexico.THEN; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
9 Lexico.FPAR; Lexico.ELSE; Lexico.PRINT; Lexico.APAR; Lexico.LITINT 0;
10 Lexico.FPAR; Lexico.END; Lexico.ID "x"; Lexico.ATRIB; Lexico.ID "x";
11 Lexico.SUBTRACAO; Lexico.LITINT 1; Lexico.END; Lexico.END; Lexico.ID "
  main";
12 Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

### 4.3.2 Programas Micro

Os resultados de passar os programas micro pelo analisador são os seguintes

#### micro01

Listagem 4.15: Resultado de passar o analisador léxico no programa micro01.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "cel"; Lexico.VIRGULA; Lexico.ID "far"; Lexico.PRINT; Lexico.
  APAR;
4 Lexico.LITSTRING " tabela de conversao: Celsius -> Fahrenheit\n";
5 Lexico.FPAR; Lexico.PRINT; Lexico.APAR;
6 Lexico.LITSTRING "Digite a temperatura em Celsius: "; Lexico.FPAR;
7 Lexico.ID "cel"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
8 Lexico.LITSTRING "*number"; Lexico.FPAR; Lexico.ID "far"; Lexico.ATRIB;
9 Lexico.APAR; Lexico.LITINT 9; Lexico.MULTIPLICACAO; Lexico.ID "cel";
10 Lexico.ADICAO; Lexico.LITINT 160; Lexico.FPAR; Lexico.DIVISAO;
11 Lexico.LITINT 5; Lexico.PRINT; Lexico.APAR;
12 Lexico.LITSTRING "A nova temperatura e: "; Lexico.CONCATENA;
13 Lexico.ID "far"; Lexico.CONCATENA; Lexico.LITSTRING " F"; Lexico.FPAR;
14 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

#### micro02

Listagem 4.16: Resultado de passar o analisador léxico no programa micro02.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "num1"; Lexico.VIRGULA; Lexico.ID "num2"; Lexico.PRINT;
4 Lexico.APAR; Lexico.LITSTRING "Digite o primeiro numero: "; Lexico.FPAR;
5 Lexico.ID "num1"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
6 Lexico.LITSTRING "*number"; Lexico.FPAR; Lexico.PRINT; Lexico.APAR;
7 Lexico.LITSTRING "Digite o segundo numero: "; Lexico.FPAR; Lexico.ID "
  num2";
8 Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR; Lexico.LITSTRING "*number";
9 Lexico.FPAR; Lexico.IF; Lexico.ID "num1"; Lexico.MAIOR; Lexico.ID "num2";
10 Lexico.THEN; Lexico.PRINT; Lexico.APAR;
11 Lexico.LITSTRING "O primeiro n\195\186mero "; Lexico.CONCATENA;
12 Lexico.ID "num1"; Lexico.CONCATENA;
13 Lexico.LITSTRING " \195\169 maior que o segundo "; Lexico.CONCATENA;
14 Lexico.ID "num2"; Lexico.FPAR; Lexico.ELSE; Lexico.PRINT; Lexico.APAR;
15 Lexico.LITSTRING "O segundo n\195\186mero "; Lexico.CONCATENA;

```

## 4.3

```
16 Lexico.ID "num2"; Lexico.CONCATENA;
17 Lexico.LITSTRING " \195\169 maior que o primeiro "; Lexico.CONCATENA;
18 Lexico.ID "num1"; Lexico.FPAR; Lexico.END; Lexico.END; Lexico.ID "main";
19 Lexico.APAR; Lexico.FPAR; Lexico.EOF]
```

---

### micro03

Listagem 4.17: Resultado de passar o analisador léxico no programa micro03.lua

```
1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "numero"; Lexico.PRINT; Lexico.APAR;
4 Lexico.LITSTRING "Digite um numero: "; Lexico.FPAR; Lexico.ID "numero";
5 Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR; Lexico.LITSTRING "*number";
6 Lexico.FPAR; Lexico.IF; Lexico.ID "numero"; Lexico.MAIOR_OU_IGUAL;
7 Lexico.LITINT 100; Lexico.THEN; Lexico.IF; Lexico.ID "numero";
8 Lexico.MENOR_OU_IGUAL; Lexico.LITINT 200; Lexico.THEN; Lexico.PRINT;
9 Lexico.APAR;
10 Lexico.LITSTRING "O numero esta no intervalo entre 100 e 200\n";
11 Lexico.FPAR; Lexico.ELSE; Lexico.PRINT; Lexico.APAR;
12 Lexico.LITSTRING "O numero nao esta no intervalo entre 100 e 200\n";
13 Lexico.FPAR; Lexico.END; Lexico.ELSE; Lexico.PRINT; Lexico.APAR;
14 Lexico.LITSTRING "O numero nao esta no intervalo entre 100 e 200\n";
15 Lexico.FPAR; Lexico.END; Lexico.END; Lexico.ID "main"; Lexico.APAR;
16 Lexico.FPAR; Lexico.EOF]
```

---

### micro04

Listagem 4.18: Resultado de passar o analisador léxico no programa micro04.lua

```
1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "x"; Lexico.VIRGULA; Lexico.ID "num"; Lexico.VIRGULA;
4 Lexico.ID "intervalo"; Lexico.ID "intervalo"; Lexico.ATRIB; Lexico.LITINT
  0;
5 Lexico.FOR; Lexico.ID "x"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.VIRGULA;
6 Lexico.LITINT 5; Lexico.VIRGULA; Lexico.LITINT 1; Lexico.DO; Lexico.PRINT
  ;
7 Lexico.APAR; Lexico.LITSTRING "Digite um numero: "; Lexico.FPAR;
8 Lexico.ID "num"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
9 Lexico.LITSTRING "*number"; Lexico.FPAR; Lexico.IF; Lexico.ID "num";
10 Lexico.MAIOR_OU_IGUAL; Lexico.LITINT 10; Lexico.THEN; Lexico.IF;
11 Lexico.ID "num"; Lexico.MENOR_OU_IGUAL; Lexico.LITINT 150; Lexico.THEN;
12 Lexico.ID "intervalo"; Lexico.ATRIB; Lexico.ID "intervalo"; Lexico.ADICAO
  ;
13 Lexico.LITINT 1; Lexico.END; Lexico.END; Lexico.END; Lexico.PRINT;
14 Lexico.APAR; Lexico.LITSTRING "Ao total, foram digitados ";
15 Lexico.CONCATENA; Lexico.ID "intervalo"; Lexico.CONCATENA;
16 Lexico.LITSTRING " numeros no intervalo entre 10 e 150"; Lexico.FPAR;
17 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]
```

---

### micro05

Listagem 4.19: Resultado de passar o analisador léxico no programa micro05.lua

```
1 - : Lexico.tokens list =
```

```

2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "nome"; Lexico.VIRGULA; Lexico.ID "sexo"; Lexico.VIRGULA;
4 Lexico.ID "x"; Lexico.VIRGULA; Lexico.ID "h"; Lexico.VIRGULA; Lexico.ID "
  m";
5 Lexico.ID "h"; Lexico.VIRGULA; Lexico.ID "m"; Lexico.ATRIB; Lexico.LITINT
  0;
6 Lexico.VIRGULA; Lexico.LITINT 0; Lexico.FOR; Lexico.ID "x"; Lexico.ATRIB;
7 Lexico.LITINT 1; Lexico.VIRGULA; Lexico.LITINT 5; Lexico.VIRGULA;
8 Lexico.LITINT 1; Lexico.DO; Lexico.PRINT; Lexico.APAR;
9 Lexico.LITSTRING "Digite o nome: "; Lexico.FPAR; Lexico.ID "nome";
10 Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR; Lexico.LITSTRING "*line";
11 Lexico.FPAR; Lexico.PRINT; Lexico.APAR;
12 Lexico.LITSTRING "H - Homem ou M - Mulher: "; Lexico.FPAR; Lexico.ID "
  sexo";
13 Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR; Lexico.LITSTRING "*line";
14 Lexico.FPAR; Lexico.IF; Lexico.ID "sexo"; Lexico.EQUIVALENTE;
15 Lexico.LITSTRING "H"; Lexico.THEN; Lexico.ID "h"; Lexico.ATRIB;
16 Lexico.ID "h"; Lexico.ADICAO; Lexico.LITINT 1; Lexico.ELSEIF;
17 Lexico.ID "sexo"; Lexico.EQUIVALENTE; Lexico.LITSTRING "M"; Lexico.THEN;
18 Lexico.ID "m"; Lexico.ATRIB; Lexico.ID "m"; Lexico.ADICAO; Lexico.LITINT
  1;
19 Lexico.ELSE; Lexico.PRINT; Lexico.APAR;
20 Lexico.LITSTRING "Sexo so pode ser H ou M!\n"; Lexico.FPAR; Lexico.END;
21 Lexico.END; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Foram inseridos
  ";
22 Lexico.CONCATENA; Lexico.ID "h"; Lexico.CONCATENA;
23 Lexico.LITSTRING " homens\n"; Lexico.FPAR; Lexico.PRINT; Lexico.APAR;
24 Lexico.LITSTRING "Foram inseridos "; Lexico.CONCATENA; Lexico.ID "m";
25 Lexico.CONCATENA; Lexico.LITSTRING " mulheres\n"; Lexico.FPAR; Lexico.END
  ;
26 Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

## micro06

Listagem 4.20: Resultado de passar o analisador léxico no programa micro06.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "numero"; Lexico.PRINT; Lexico.APAR;
4 Lexico.LITSTRING "Digite um numero de 1 a 5: "; Lexico.FPAR;
5 Lexico.ID "numero"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
6 Lexico.LITSTRING "*number"; Lexico.FPAR; Lexico.IF; Lexico.ID "numero";
7 Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.THEN; Lexico.PRINT; Lexico.
  APAR;
8 Lexico.LITSTRING "Um\n"; Lexico.FPAR; Lexico.ELSEIF; Lexico.ID "numero";
9 Lexico.EQUIVALENTE; Lexico.LITINT 2; Lexico.THEN; Lexico.PRINT; Lexico.
  APAR;
10 Lexico.LITSTRING "Dois\n"; Lexico.FPAR; Lexico.ELSEIF; Lexico.ID "numero"
  ;
11 Lexico.EQUIVALENTE; Lexico.LITINT 3; Lexico.THEN; Lexico.PRINT; Lexico.
  APAR;
12 Lexico.LITSTRING "Tres\n"; Lexico.FPAR; Lexico.ELSEIF; Lexico.ID "numero"
  ;
13 Lexico.EQUIVALENTE; Lexico.LITINT 4; Lexico.THEN; Lexico.PRINT; Lexico.
  APAR;
14 Lexico.LITSTRING "Quatro\n"; Lexico.FPAR; Lexico.ELSEIF; Lexico.ID "
  numero";
15 Lexico.EQUIVALENTE; Lexico.LITINT 5; Lexico.THEN; Lexico.PRINT; Lexico.

```

```

    APAR;
16 Lexico.LITSTRING "Cinco\n"; Lexico.FPAR; Lexico.ELSE; Lexico.PRINT;
17 Lexico.APAR; Lexico.LITSTRING "Numero invalido!!!"; Lexico.FPAR; Lexico.
    END;
18 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## micro07

Listagem 4.21: Resultado de passar o analisador léxico no programa micro07.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "programa"; Lexico.VIRGULA; Lexico.ID "numero"; Lexico.VIRGULA;
4 Lexico.ID "opc"; Lexico.ID "programa"; Lexico.ATRIB; Lexico.LITINT 1;
5 Lexico.WHILE; Lexico.ID "programa"; Lexico.EQUIVALENTE; Lexico.LITINT 1;
6 Lexico.DO; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um numero:
    ";
7 Lexico.FPAR; Lexico.ID "numero"; Lexico.ATRIB; Lexico.IO_READ; Lexico.
    APAR;
8 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.IF; Lexico.ID "numero";
9 Lexico.MAIOR; Lexico.LITINT 0; Lexico.THEN; Lexico.PRINT; Lexico.APAR;
10 Lexico.LITSTRING "Positivo\n"; Lexico.FPAR; Lexico.ELSE; Lexico.IF;
11 Lexico.ID "numero"; Lexico.EQUIVALENTE; Lexico.LITINT 0; Lexico.THEN;
12 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O numero e igual a 0\n";
13 Lexico.FPAR; Lexico.END; Lexico.IF; Lexico.ID "numero"; Lexico.MENOR;
14 Lexico.LITINT 0; Lexico.THEN; Lexico.PRINT; Lexico.APAR;
15 Lexico.LITSTRING "Negativo\n"; Lexico.FPAR; Lexico.END; Lexico.END;
16 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Deseja finalizar? (S - 1): "
    ;
17 Lexico.FPAR; Lexico.ID "opc"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
18 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.IF; Lexico.ID "opc";
19 Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.THEN; Lexico.ID "programa";
20 Lexico.ATRIB; Lexico.LITINT 0; Lexico.END; Lexico.END; Lexico.END;
21 Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## micro08

Listagem 4.22: Resultado de passar o analisador léxico no programa micro08.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "numero"; Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 1;
4 Lexico.WHILE; Lexico.ID "numero"; Lexico.NAO_EQUIVALENTE; Lexico.LITINT
    0;
5 Lexico.DO; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um numero:
    ";
6 Lexico.FPAR; Lexico.ID "numero"; Lexico.ATRIB; Lexico.IO_READ; Lexico.
    APAR;
7 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.IF; Lexico.ID "numero";
8 Lexico.MAIOR; Lexico.LITINT 10; Lexico.THEN; Lexico.PRINT; Lexico.APAR;
9 Lexico.LITSTRING "O numero "; Lexico.CONCATENA; Lexico.ID "numero";
10 Lexico.CONCATENA; Lexico.LITSTRING " e maior que 10\n"; Lexico.FPAR;
11 Lexico.ELSE; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O numero ";
12 Lexico.CONCATENA; Lexico.ID "numero"; Lexico.CONCATENA;
13 Lexico.LITSTRING " e menor que 10\n"; Lexico.FPAR; Lexico.END; Lexico.END
    ;
14 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

---

## micro09

Listagem 4.23: Resultado de passar o analisador léxico no programa micro09.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "preco"; Lexico.VIRGULA; Lexico.ID "venda"; Lexico.VIRGULA;
4 Lexico.ID "novo_preco"; Lexico.PRINT; Lexico.APAR;
5 Lexico.LITSTRING "Digite o preco: "; Lexico.FPAR; Lexico.ID "preco";
6 Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR; Lexico.LITSTRING "*n";
7 Lexico.FPAR; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite a venda:
8 Lexico.FPAR; Lexico.ID "venda"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR
9 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.IF; Lexico.ID "venda";
10 Lexico.MENOR; Lexico.LITINT 500; Lexico.OR; Lexico.ID "preco"; Lexico.
11 Lexico.LITINT 30; Lexico.THEN; Lexico.ID "novo_preco"; Lexico.ATRIB;
12 Lexico.ID "preco"; Lexico.ADICAO; Lexico.LITINT 10; Lexico.DIVISAO;
13 Lexico.LITINT 100; Lexico.MULTIPLICACAO; Lexico.ID "preco"; Lexico.ELSEIF
14 Lexico.APAR; Lexico.ID "venda"; Lexico.MAIOR_OU_IGUAL; Lexico.LITINT 500;
15 Lexico.AND; Lexico.ID "venda"; Lexico.MENOR_OU_IGUAL; Lexico.LITINT 1200;
16 Lexico.FPAR; Lexico.OR; Lexico.APAR; Lexico.ID "preco";
17 Lexico.MAIOR_OU_IGUAL; Lexico.LITINT 30; Lexico.AND; Lexico.ID "preco";
18 Lexico.MENOR; Lexico.LITINT 80; Lexico.FPAR; Lexico.THEN;
19 Lexico.ID "novo_preco"; Lexico.ATRIB; Lexico.ID "preco"; Lexico.ADICAO;
20 Lexico.LITINT 15; Lexico.DIVISAO; Lexico.LITINT 100; Lexico.MULTIPLICACAO
21 Lexico.ID "preco"; Lexico.ELSEIF; Lexico.ID "venda"; Lexico.
22 Lexico.LITINT 1200; Lexico.OR; Lexico.ID "preco"; Lexico.MAIOR_OU_IGUAL;
23 Lexico.LITINT 80; Lexico.THEN; Lexico.ID "novo_preco"; Lexico.ATRIB;
24 Lexico.ID "preco"; Lexico.SUBTRACAO; Lexico.LITINT 20; Lexico.DIVISAO;
25 Lexico.LITINT 100; Lexico.MULTIPLICACAO; Lexico.ID "preco"; Lexico.END;
26 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O novo preco e ";
27 Lexico.CONCATENA; Lexico.ID "novo_preco"; Lexico.CONCATENA;
28 Lexico.LITSTRING "\n"; Lexico.FPAR; Lexico.END; Lexico.ID "main";
29 Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

## micro10

Listagem 4.24: Resultado de passar o analisador léxico no programa micro10.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "numero"; Lexico.VIRGULA; Lexico.ID "fat"; Lexico.PRINT;
4 Lexico.APAR; Lexico.LITSTRING "Digite um numero: "; Lexico.FPAR;
5 Lexico.ID "numero"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
6 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.ID "fat"; Lexico.ATRIB;
7 Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
8 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O fatorial de ";
9 Lexico.CONCATENA; Lexico.ID "numero"; Lexico.CONCATENA;
10 Lexico.LITSTRING " e "; Lexico.CONCATENA; Lexico.ID "fat"; Lexico.
11 Lexico.LITSTRING "\n"; Lexico.FPAR; Lexico.END; Lexico.FUNCAO;
12 Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.IF;
13 Lexico.ID "n"; Lexico.MENOR_OU_IGUAL; Lexico.LITINT 0; Lexico.THEN;

```

```

14 Lexico.RETURN; Lexico.LITINT 1; Lexico.ELSE; Lexico.RETURN; Lexico.ID "n"
    ;
15 Lexico.MULTIPLICACAO; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID "n";
16 Lexico.SUBTRACAO; Lexico.LITINT 1; Lexico.FPAR; Lexico.END; Lexico.END;
17 Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

## micro11

Listagem 4.25: Resultado de passar o analisador léxico no programa micro11.lua

```

1 - : Lexico.tokens list =
2 [Lexico.FUNCAO; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.LOCAL;
3 Lexico.ID "numero"; Lexico.VIRGULA; Lexico.ID "x"; Lexico.PRINT;
4 Lexico.APAR; Lexico.LITSTRING "Digite um numero "; Lexico.FPAR;
5 Lexico.ID "numero"; Lexico.ATRIB; Lexico.IO_READ; Lexico.APAR;
6 Lexico.LITSTRING "*n"; Lexico.FPAR; Lexico.ID "x"; Lexico.ATRIB;
7 Lexico.ID "verifica"; Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
8 Lexico.IF; Lexico.ID "x"; Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.
    THEN;
9 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Numero positivo\n";
10 Lexico.FPAR; Lexico.ELSEIF; Lexico.ID "x"; Lexico.EQUIVALENTE;
11 Lexico.LITINT 0; Lexico.THEN; Lexico.PRINT; Lexico.APAR;
12 Lexico.LITSTRING "Zero\n"; Lexico.FPAR; Lexico.ELSE; Lexico.PRINT;
13 Lexico.APAR; Lexico.LITSTRING "Numero negativo\n"; Lexico.FPAR; Lexico.
    END;
14 Lexico.END; Lexico.FUNCAO; Lexico.ID "verifica"; Lexico.APAR; Lexico.ID "
    n";
15 Lexico.FPAR; Lexico.LOCAL; Lexico.ID "res"; Lexico.IF; Lexico.ID "n";
16 Lexico.MAIOR; Lexico.LITINT 0; Lexico.THEN; Lexico.ID "res"; Lexico.ATRIB
    ;
17 Lexico.LITINT 1; Lexico.ELSEIF; Lexico.ID "n"; Lexico.MENOR;
18 Lexico.LITINT 0; Lexico.THEN; Lexico.ID "res"; Lexico.ATRIB;
19 Lexico.SUBTRACAO; Lexico.LITINT 1; Lexico.ELSE; Lexico.ID "res";
20 Lexico.ATRIB; Lexico.LITINT 0; Lexico.END; Lexico.RETURN; Lexico.ID "res"
    ;
21 Lexico.END; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.EOF]

```

## 4.4 Teste de Erros

Nesta seção serão exibidos alguns erros léxicos que podem ocorrer e a resposta devolvida pelo analisador léxico. Detalhe na resposta do erro ele indica a linha e a coluna onde o erro foi encontrado.

### Comentário de bloco não fechado

Arquivo de entrada

Listagem 4.26: micro05Erro.lua - Lê strings e caracteres - com erro proposital de não fechar um comentário de bloco

```

1 function main()
2     local nome, sexo, x, h, m
3     h, m = 0, 0
4     for x = 1, 5, 1

```

```

5      do
6          print("Digite o nome: ")
7          nome = io.read("*line")
8          print("H - Homem ou M - Mulher: ")
9          sexo = io.read("*line")
10         --[[if sexo == "H" then
11             h = h + 1
12         elseif sexo == "M" then
13             m = m + 1
14         else
15             print("Sexo so pode ser H ou M!\n")
16         end
17     end
18     print("Foram inseridos "..h.." homens\n")
19     print("Foram inseridos "..m.." mulheres\n")
20 end
21
22 main()

```

---

Resultado do analisador

Listagem 4.27: Resultado de passar o analisador léxico no programa micro05Erro.lua

```
1 Exception: Failure "10-11: Comentario nao fechado".
```

---

## Carácter invalido

Arquivo de entrada

Listagem 4.28: micro06Erro.lua - Escreve um número lido por extenso - com erro proposital de adicionar caracter inválido

```

1 function main()
2     local numero
3     print("Digite um numero de 1 a 5: ")
4     numero = io.read("*number")
5     if numero == 1 then
6         print("Um\n")
7     elseif numero == 2 then
8         print("Dois\n")
9     elseif numero == 3 then
10        print("Tres\n")
11    elseif numero == 4 then
12        print("Quatro\n")
13    elseif numero == 5 then
14        print("Cinco\n")
15    else
16        @
17        print("Numero invalido!!!")
18    end
19 end
20
21 main()

```

---

Resultado do analisador



## Listagem 4.29: Resultado de passar o analisador léxico no programa micro06Erro.lua

```
1 Exception: Failure "16-1: caracter desconhecido @".
```

---

## String com aspas não fechada corretamente

## Arquivo de entrada

## Listagem 4.30: micro07.lua - Decide se os números são positivos, zeros ou negativos

```
1 function main()
2     local programa, numero, opc
3     programa = 1
4     while programa == 1 do
5         print("Digite um numero: ")
6         numero = io.read("*n")
7         if numero > 0 then
8             print("Positivo\n")
9         else
10            if numero == 0 then
11                print("O numero e igual a 0\n")
12            end
13            if numero < 0 then
14                print("Negativo\n")
15            end
16        end
17        print("Deseja finalizar? (S - 1): ")
18        opc = io.read("*n")
19        if opc == 1 then
20            programa = 0
21        end
22    end
23 end
24
25 main()
```

---

## Resultado do analisador

## Listagem 4.31: Resultado de passar o analisador léxico no programa micro07Erro.lua

```
1 Exception: Failure "14-22: A string nao foi fechada".
```

---

# Capítulo 5

## Analizador Sintático

O analisador sintático tem o dever de construir uma árvore gramatical para uma dada sentença de entrada. Caso uma dada sentença não pertença a gramática que o analisador está verificando, ele dará, então, uma indicação de erro. Dessa forma, neste capítulo será mostrado a implementação de tal analisador para a linguagem MiniLua, adaptada para permitir tipagem, feita com a linguagem Ocaml. Dessa forma, os programas nanos e micros do capítulo 3 foram reescritos de forma a acomodar essa adaptação, que facilitará as próximas etapas do projeto. Além disso, como a definição da gramática agora está no analisador sintático, não é mais necessária que a mesma esteja duplicada no analisador léxico, portanto reescreveu-se o mesmo afim de simplificá-lo e tornar o projeto mais eficiente de forma geral. Por fim, faz-se a análise léxica de todos esses programas exemplos em lua e alguns testes de erros para confirmar que o analisador sintático está em bom funcionamento.

### 5.1 Código do Analisador Sintático

Segue o código do analisador sintático para a linguagem MiniLua utilizada nesse trabalho

Listagem 5.1: sintatico.mly - Código com a gramática do analisador sintático

```
1 %{
2   open Ast
3 %}
4
5 %token <string>    ID
6 %token <string>    LITSTRING
7 %token <int>       LITINT
8 %token <bool>      BOOL
9 %token             ABRE_CHAVE
10 %token             ABRE_COLCHETE
11 %token             FECHA_CHAVE
12 %token             FECHA_COLCHETE
13 %token             ADICAO
14 %token             AND
15 %token             AND_BINARIO
16 %token             APAR
17 %token             ATRIB
18 %token             BREAK
```

## 5.1

19 %token	CONCATENA
20 %token	DIV_POR_2
21 %token	DIVISAO
22 %token	DIVISAO_INTEIRO
23 %token	DO
24 %token	DOIS_PONTOS
25 %token	ELSE
26 %token	ELSEIF
27 %token	END
28 %token	EQUIVALENTE
29 %token	EXPONENCIACAO
30 %token	FOR
31 %token	FPAR
32 %token	FUNCAO
33 %token	IF
34 %token	IN
35 %token	IO_READ
36 %token	LOCAL
37 %token	MAIOR
38 %token	MAIOR_OU_IGUAL
39 %token	MENOR
40 %token	MENOR_OU_IGUAL
41 %token	MODULO
42 %token	MULT_POR_2
43 %token	MULTIPLICACAO
44 %token	NAO_EQUIVALENTE
45 %token	NIL
46 %token	NOT
47 %token	NUMBER_INPUT
48 %token	OR
49 %token	OR_BINARIO
50 %token	OR_BINARIO_EXCLUSIVO
51 %token	PONTO
52 %token	PONTO_VIRGULA
53 %token	PRINT
54 %token	RETICENCIAS
55 %token	RETURN
56 %token	SUBTRACAO
57 %token	TAMANHO
58 %token	TIPO_BOOLEAN
59 %token	TIPO_INT
60 %token	TIPO_STRING
61 %token	THEN
62 %token	UNTIL
63 %token	VIRGULA
64 %token	WHILE
65 %token	EOF
66 %token	FALSE
67 %token	TRUE
68 %token	REPEAT
69	
70 %left	OR
71 %left	AND
72 %left	MAIOR MENOR MENOR_OU_IGUAL MAIOR_OU_IGUAL EQUIVALENTE NAO_EQUIVALENTE
73 %left	OR_BINARIO
74 %left	OR_BINARIO_EXCLUSIVO
75 %left	AND_BINARIO
76 %left	MULT_POR_2 DIV_POR_2

```

77 %left          CONCATENA
78 %left          ADICAO SUBTRACAO
79 %left          MULTIPLICACAO DIVISAO DIVISAO_INTEIRO MODULO
80 %left          NOT TAMANHO
81 %left          EXPONENCIACAO
82
83 %start <Ast.programa> programa
84
85 %%
86
87 programa: f = funcoes+
88     EOF { Programa (f) }
89
90 funcoes:
91     | FUNCAO tipo=tipo_simples id=ID APAR args=argumentos* FPAR ds=
92       declaracao* cs=comando*
93     (*ret=retorno*)
94     END { Funcao (tipo, id, args, ds, cs(*,ret*)) }
95     ;
96
97 argumentos:
98     | t = tipo_simples id = ID { Args (t, id) }
99     ;
100
101 declaracao:
102     | t=tipo v=variavel {DecVar (t,v)}
103     | LOCAL t=tipo v=variavel { DecVar (t,v) }
104     ;
105
106 tipo: t = tipo_simples { t }
107
108 tipo_simples: TIPO_INT          { TipoInt }
109              | TIPO_STRING      { TipoString }
110              | TIPO_BOOLEAN     { TipoBool  }
111
112 comando:  c = comando_atribuicao { c }
113          | c = comando_if        { c }
114          | c = comando_for        { c }
115          | c = comando_while      { c }
116          | c = comando_print      { c }
117          | c = comando_scan       { c }
118          | c = comando_funcao     { c }
119          | c = comando_retorno    { c }
120
121 comando_atribuicao:
122     | v = variavel ATRIB e = expressao { CmdAtrib (v,e) }
123     | v = variavel ATRIB id = ID APAR args = ID* FPAR {
124       CmdAtribRetorno (v, id, args) }
125     ;
126
127 comando_if: IF teste = expressao THEN
128     entao = comando+
129     senao = option (ELSE cs = comando+ { cs })
130     END {
131       CmdIf (teste, entao, senao)
132     }
133
134 comando_for:

```

## 5.1

```
133 | FOR v = variavel  ATRIB l1=LITINT VIRGULA l2 = LITINT VIRGULA l3 =
    LITINT DO
134     cs = comando* END {CmdFor (v, l1, l2, l3, cs) }
135 ;
136
137 comando_while:
138 | WHILE teste = expressao DO  cs = comando* END { CmdWhile  (teste, cs)
    }
139 ;
140
141 comando_print:
142 | PRINT APAR  teste = expressao FPAR  {CmdPrint (teste) }
143 ;
144
145 comando_scan:
146 | v = variavel  ATRIB IO_READ APAR  FPAR  {CmdScan  (v) }
147 ;
148
149 comando_funcao:
150 | id=ID APAR  args=ID* FPAR {CmdFunction  (id, args)  }
151 ;
152
153 comando_retorno:
154 | RETURN  exp = expressao { CmdRetorno  (exp) }
155 ;
156
157 expressao:
158 | v = variavel  { ExpVar  v }
159 | i = LITINT    { ExpInt  i }
160 | s = LITSTRING { ExpString s }
161 | b = BOOL      { ExpBool b }
162 | e1 = expressao op = oper e2 = expressao  { ExpOp (op, e1, e2)  }
163 | APAR e = expressao FPAR  { e }
164
165
166 %inline oper:
167 | OR { Or }
168 | AND { And }
169 | MAIOR { Maior }
170 | MENOR { Menor }
171 | MAIOR_OU_IGUAL { Maior_ou_Igual }
172 | MENOR_OU_IGUAL { Menor_ou_Igual }
173 | EQUIVALENTE { Equivalente }
174 | NAO_EQUIVALENTE { Nao_Equivalente }
175 | OR_BINARIO { Or_Binario }
176 | OR_BINARIO_EXCLUSIVO { Or_Binario_Exclusivo }
177 | AND_BINARIO { And_Binario }
178 | MULT_POR_2 { Mult_Por_2 }
179 | DIV_POR_2 { Div_Por_2 }
180 | CONCATENA { Concatena }
181 | ADICAO { Adicao }
182 | SUBTRACAO { Subtracao }
183 | MULTIPLICACAO { Multiplicacao }
184 | DIVISAO { Divisao }
185 | DIVISAO_INTEIRO { Divisao_Inteiro }
186 | MODULO { Modulo }
187 | NOT { Not }
188 | TAMANHO { Tamanho }
189 | EXPONENCIACAO { Exponenciacao }
```

```

190
191 variavel:
192   | x=ID      { VarSimples  x }

```

---

## 5.2 Código da Árvore Sintática Abstrata

Segue o código da árvore sintática abstrata, utilizada para construir a árvore sintática dada uma entrada

Listagem 5.2: ast.ml - Código da árvore sintática abstrata

```

1 type identificador = string
2 type programa      = Programa of funcoes list
3
4 and funcoes = Funcao of tipo * identificador * argumentos list *
   declaracoes * comandos (** retorno*)
5
6 and argumentos = Args of tipo * identificador
7
8 and declaracoes = declaracao list
9
10 and declaracao = DecVar of tipo * variavel
11
12 and comandos = comando list
13
14 and tipo =   TipoInt
15             | TipoString
16             | TipoBool
17
18 and comando = CmdAtrib of variavel * expressao
19               | CmdAtribRetorno of variavel * identificador * identificador
20                   list
21               | CmdIf of expressao * comandos * (comandos option)
22               | CmdFor of variavel * int * int * int * comandos
23               | CmdWhile of expressao * comandos
24               | CmdPrint of expressao
25               | CmdScan of variavel
26               | CmdFunction of identificador * identificador list
27               | CmdRetorno of expressao
28
29 and variaveis = variavel list
30
31 and variavel = VarSimples of identificador
32
33 and expressao = ExpVar of variavel
34               | ExpInt of int
35               | ExpString of string
36               | ExpBool of bool
37               | ExpOp of oper * expressao * expressao
38
39 and oper = Or
40           | And
41           | Maior
42           | Menor
43           | Maior_ou_Igual

```

```

44         | Menor_ou_Igual
45         | Equivalente
46         | Nao_Equivalente
47         | Or_Binario
48         | Or_Binario_Exclusivo
49         | And_Binario
50         | Mult_Por_2
51         | Div_Por_2
52         | Concatena
53         | Adicao
54         | Subtracao
55         | Multiplicacao
56         | Divisao
57         | Divisao_Inteiro
58         | Modulo
59         | Not
60         | Tamanho
61         | Exponenciacao

```

---

## 5.3 Código do sintaticoTest

Código para auxiliar a análise sintática de um arquivo inteiro

Listagem 5.3: sintaticoTest.ml - Código auxiliar

```

1  open Printf
2  open Lexing
3
4  open Ast
5  open ErroSint (*nome do modulo contendo as mensagens de erro*)
6
7  exception Erro_Sintatico of string
8
9  module S = MenhirLib.General (* Streams *)
10 module I = Sintatico.MenhirInterpreter
11
12 let posicao lexbuf =
13     let pos = lexbuf.lex_curr_p in
14     let lin = pos.pos_lnum
15     and col = pos.pos_cnum - pos.pos_bol - 1 in
16     sprintf "linha %d, coluna %d" lin col
17
18 (* [Pilha checkpoint] extrai a pilha do automato LR(1) contida em
19    checkpoint *)
20 let pilha checkpoint =
21     match checkpoint with
22     | I.HandlingError amb -> I.stack amb
23     | _ -> assert false (* Isso n'ao pode acontecer *)
24
25 let estado checkpoint : int =
26     match Lazy.force (pilha checkpoint) with
27     | S.Nil -> (*O parser esta no estado inicial *)
28         0
29     | S.Cons (I.Element (s, _, _, _), _) ->
30         I.number s

```

```

31
32 let sucesso v = Some v
33
34 let falha lexbuf (checkpoint : Ast.programa I.checkpoint) =
35   let estado_atual = estado checkpoint in
36   let msg = message estado_atual in
37   raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n" (Lexing.lexeme_start
    lexbuf) msg))
38
39 let loop lexbuf resultado =
40   let fornecedor = I.lexer_lexbuf_to_supplier Lexico.token lexbuf in
41   I.loop_handle sucesso (falha lexbuf) fornecedor resultado
42
43 let parse_com_erro lexbuf =
44   try
45     Some (loop lexbuf (Sintatico.Incremental.programa lexbuf.lex_curr_p))
46   with
47     | Lexico.Erro msg -> printf "Erro lexico na %s:\n\t%s\n" (posicao
    lexbuf) msg;
48     None
49     | Erro_Sintatico msg ->
50       printf "Erro sintatico na %s %s\n" (posicao lexbuf) msg;
51       None
52
53 let parse s =
54   let lexbuf = Lexing.from_string s in
55   let ast = parse_com_erro lexbuf in
56   ast
57
58 let parse_arq nome =
59   let ic = open_in nome in
60   let lexbuf = Lexing.from_channel ic in
61   let result = parse_com_erro lexbuf in
62   let _ = close_in ic in
63   match result with
64     | Some ast -> ast
65     | None -> failwith "A analise sintatica falhou"
66
67 (* Para compilar:
68   menhir -v --list-errors sintatico.mly > sintatico.msg
69   menhir -v sintatico.mly --compile-errors sintatico.msg > erroSint.ml
70   ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
    menhirLib sintaticoTeste.byte
71 *)

```

## 5.4 Novo código do analisador léxico

Por fim, segue o código do novo analisador léxico, depois de retirado as definições de tokens dele para que não haja duplicatas

Listagem 5.4: lexico.mll - Novo código para o analisador léxico

```

1 {
2   open Lexing
3   open Printf
4   open Sintatico

```



```

5
6  exception Erro of string
7
8  let incr_num_linha lexbuf =
9      let pos = lexbuf.lex_curr_p in
10         lexbuf.lex_curr_p <- { pos with
11             pos_lnum = pos.pos_lnum + 1;
12             pos_bol = pos.pos_cnum;
13         }
14
15 }
16
17 let digito = ['0' - '9']
18 let inteiro = '-'? digito+
19
20 let letra = ['a' - 'z' 'A' - 'Z']
21 let identificador = letra ( letra | digito | '_' ) *
22
23 let brancos = [' ' '\t']+
24 let novalinha = '\r' | '\n' | "\r\n"
25
26 rule token = parse
27 | brancos                { token lexbuf }
28 | novalinha              { incr_num_linha lexbuf; token lexbuf }
29 | "--["                 { comentario_bloco lexbuf }
30 | "--"                  { comentario_linha lexbuf }
31 | "int"                  { TIPO_INT }
32 | "bool"                 { TIPO_BOOLEAN }
33 | "string"               { TIPO_STRING }
34 | "("                    { APAR }
35 | "{"                    { ABRE_CHAVE }
36 | "["                    { ABRE_COLCHETE }
37 | "+"                    { ADICAO }
38 | "-"                    { SUBTRACAO }
39 | ")"                    { FPAR }
40 | "}"                    { FECHA_CHAVE }
41 | "]"                    { FECHA_COLCHETE }
42 | ","                    { VIRGULA }
43 | "."                    { PONTO }
44 | ";"                    { PONTO_VIRGULA }
45 | ":"                    { DOIS_PONTOS }
46 | "=="                   { EQUIVALENTE }
47 | "~="                   { NAO_EQUIVALENTE }
48 | ">="                    { MAIOR_OU_IGUAL }
49 | "<="                    { MENOR_OU_IGUAL }
50 | "/"                    { DIVISAO }
51 | "*"                    { MULTIPLICACAO }
52 | "%"                    { MODULO }
53 | "^"                    { EXPONENCIACAO }
54 | ">"                    { MAIOR }
55 | "<"                    { MENOR }
56 | "="                    { ATRIB }
57 | "#"                    { TAMANHO }
58 | "<<"                   { MULT_POR_2 }
59 | ">>"                   { DIV_POR_2 }
60 | "//"                   { DIVISAO_INTEIRO }
61 | "&"                    { AND_BINARIO }
62 | "|"                    { OR_BINARIO }
63 | ".."                   { CONCATENA }

```

```

64 | "..."          { RETICENCIAS }
65 | "and"           { AND }
66 | "break"        { BREAK }
67 | "do"           { DO }
68 | "else"         { ELSE }
69 | "elseif"       { ELSEIF }
70 | "end"          { END }
71 | "false"        { FALSE }
72 | "for"          { FOR }
73 | "function"     { FUNCAO }
74 | "if"           { IF }
75 | "io.read"      { IO_READ }
76 | "in"           { IN }
77 | "local"        { LOCAL }
78 | "nil"          { NIL }
79 | "not"          { NOT }
80 | "print"        { PRINT }
81 | "or"           { OR }
82 | "repeat"       { REPEAT }
83 | "return"       { RETURN }
84 | "then"         { THEN }
85 | "true"         { TRUE }
86 | "until"        { UNTIL }
87 | "while"        { WHILE }
88 | inteiro as num      { let numero = int_of_string num in
89 |                               LITINT numero }
90 | identificador as id { ID id }
91 | '''           { let buffer = Buffer.create 1 in
92 |               let str = leia_string buffer lexbuf in
93 |               LITSTRING str }
94 | _             { raise (Erro ("Caracter desconhecido: " ^ Lexing.
95 |               lexeme lexbuf))}
96 | eof           { EOF }
97
98 and comentario_bloco = parse
99   "--]]"       { token lexbuf }
100 | novalinha    { incr_num_linha lexbuf; comentario_bloco lexbuf }
101 | _           { comentario_bloco lexbuf }
102 | eof         { raise (Erro "Comentario nao terminado")}
103
104 and leia_string buffer = parse
105   '''          { Buffer.contents buffer}
106 | "\\t"        { Buffer.add_char buffer '\t'; leia_string buffer lexbuf }
107 | "\\n"        { Buffer.add_char buffer '\n'; leia_string buffer lexbuf }
108 | '\\\'' '''   { Buffer.add_char buffer '\''; leia_string buffer lexbuf }
109 | '\\\'' '\\\'' { Buffer.add_char buffer '\\\''; leia_string buffer lexbuf }
110 | novalinha    {raise (Erro "A string nao foi fechada")}
111 | _ as c       { Buffer.add_char buffer c; leia_string buffer lexbuf }
112 | eof         { raise (Erro "A string nao foi terminada")}
113
114 and comentario_linha = parse
115   novalinha {incr_num_linha lexbuf; token lexbuf}
116 | _        {comentario_linha lexbuf}

```

---

## 5.5 Usando o analisador sintático

### 5.5.1 Pré-requisitos

Para executar o analisador léxico, são necessários que alguns pacotes do Ocaml extras sejam instalados. Para tal, deve-se começar instalando o ocamlbuild da seguinte maneira

```
$ sudo apt-get update
$ sudo apt-get install ocamlbuild
```

Em seguida, deve-se instalar o pacote *menhir*, pois utilizaremos *menhirLib*. Para tal, deve-se realizar o download do pacote no site oficial, [link para download](#), e então executar os seguintes comandos em seu diretório após descompactá-lo

```
$ make -f Makefile PREFIX=/usr/local USE_OCAMLFIND=true TARGET=byte all
$ sudo make -f Makefile PREFIX=/usr/local TARGET=byte install
```

Pode-se verificar que tal instalação ocorreu corretamente executando

```
$ ocamlfind query menhirLib
```

### 5.5.2 Compilando o analisador sintático

Antes que se possa executar o analisador sintático, devemos configurar suas mensagens de erro e compilá-lo. Para isso, vamos primeiramente gerar suas mensagens de erro, por meio do comando

```
$ menhir -v --list-errors sintatico.mly > sintatico.msg
```

Este código gerará um arquivo *.msg* contendo os casos de erro e suas respectivas mensagens. Essas mensagens devem ser modificadas nesse arquivo neste momento com mensagens que fazem sentido para cada tipo de erro antes que possamos prosseguir. Após feito isso, compile o arquivo de mensagens de erro para que o mesmo possa ser utilizado pelo analisador sintático da seguinte maneira

```
$ menhir sintatico.mly --compile-errors sintatico.msg > erroSint.ml
```

Agora podemos compilar todo o projeto com o auxílio de ocamlbuild, que criará uma nova pasta *\_build* como arquivos do projeto. Faz — se assim da seguinte maneira

```
$ ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -
package menhirLib sintaticoTest.byte
```

Agora estamos prontos para utilizar o analisador.

### 5.5.3 Executando o analisador

Para executar o analisador sintático, primeiramente devemos entrar no ambiente Ocaml

```
$ rlwrap ocaml
```

E, finalmente, podemos realizar a análise sintática de um arquivo .lua por meio da função no arquivo auxiliar `sintaticoTest.ml` ?? `parsearqepassaronomedoarquivooseraanalisadoocomoparametroo.D`

```
# parse_arq "nome_do_arquivo_a_ser_analisador.lua";;
```

Esse comando retornará o resultado do analisador sintático.

## 5.6 Testando o Analisador Sintático

Os seguintes testes com os programas nano e micro modificados tem a finalidade de validar a corretude das árvores geradas pelo analisador sintático.

### 5.6.1 Programas nano

#### Nano01

Listagem 5.5: nano01.lua - Programa nano01 modificado

```
1 function int main()
2 end
```

Saida do analisador sintático:

Listagem 5.6: nano01.txt - Resultado do analisador sintático executado sobre nano01.lua modificado

```
1 - : Ast.programa option =
2 Some (Programa [Funcao (TipoInt, "main", [], [], [])])
```

#### Nano02

Listagem 5.7: nano02.lua - Programa nano02 modificado

```
1 function int main()
2     local int n
3 end
```

Saida do analisador sintático:

Listagem 5.8: nano02.txt - Resultado do analisador sintático executado sobre nano02.lua modificado

```
1 - : Ast.programa option =
2 Some
3 (Programa
4  [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")], [])])
```

## Nano03

### Listagem 5.9: nano03.lua - Programa nano03 modificado

```
1 function int main()
2     local int n
3     n = 1
4 end
```

Saida do analisador sintático:

### Listagem 5.10: nano03.txt - Resultado do analisador sintático executado sobre nano03.lua modificado

```
1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n", ExpInt 1)])])
```

## Nano04

### Listagem 5.11: nano04.lua - Programa nano04 modificado

```
1 function int main()
2     local int n
3     n = 1 + 2
4 end
```

Saida do analisador sintático:

### Listagem 5.12: nano04.txt - Resultado do analisador sintático executado sobre nano04.lua modificado

```
1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n", ExpOp (Adicao, ExpInt 1, ExpInt 2))])])
```

## Nano05

### Listagem 5.13: nano05.lua - Programa nano05 modificado

```
1 function int main()
2     local int n
3     n = 2
4     print(n)
5 end
```

Saida do analisador sintático:

### Listagem 5.14: nano05.txt - Resultado do analisador sintático executado sobre nano05.lua modificado

```
1 - : Ast.programa option =
2 Some
```

```

3  (Programa
4    [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5      [CmdAtrib (VarSimples "n", ExpInt 2);
6        CmdPrint (ExpVar (VarSimples "n"))])]])

```

---

## Nano06

### Listagem 5.15: nano06.lua - Programa nano06 modificado

```

1 function int main()
2   local int n
3   n = 1 - 2
4   print(n)
5 end

```

---

Saida do analisador sintático:

### Listagem 5.16: nano06.txt - Resultado do analisador sintático executado sobre nano06.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n", ExpOp (Subtracao, ExpInt 1, ExpInt 2));
6       CmdPrint (ExpVar (VarSimples "n"))])]])

```

---

## Nano07

### Listagem 5.17: nano07.lua - Programa nano07 modificado

```

1 function int main()
2   local int n
3   n = 1
4   if n == 1 then
5     print(n)
6   end
7 end

```

---

Saida do analisador sintático:

### Listagem 5.18: nano07.txt - Resultado do analisador sintático executado sobre nano07.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n", ExpInt 1);
6       CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "n"), ExpInt 1),
7         [CmdPrint (ExpVar (VarSimples "n"))], None)])]])

```

---

## Nano08

### Listagem 5.19: nano08.lua - Programa nano08 modificado

## 5.6

```
1 function int main()
2     local int n
3     n = 1
4     if n == 1 then
5         print(n)
6     else
7         print(0)
8     end
9 end
```

---

Saida do analisador sintático:

Listagem 5.20: nano08.txt - Resultado do analisador sintático executado sobre nano08.lua modificado

```
1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n", ExpInt 1);
6       CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "n"), ExpInt 1),
7         [CmdPrint (ExpVar (VarSimples "n"))], Some [CmdPrint (ExpInt 0)])])
9   ])
```

---

## Nano09

Listagem 5.21: nano09.lua - Programa nano09 modificado

```
1 function int main()
2     local int n
3     n = 1 + 1 / 2
4     if n == 1 then
5         print(n)
6     else
7         print(0)
8     end
9 end
```

---

Saida do analisador sintático:

Listagem 5.22: nano09.txt - Resultado do analisador sintático executado sobre nano09.lua modificado

```
1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "n")],
5     [CmdAtrib (VarSimples "n",
6       ExpOp (Adicao, ExpInt 1, ExpOp (Divisao, ExpInt 1, ExpInt 2)))];
7     CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "n"), ExpInt 1),
8       [CmdPrint (ExpVar (VarSimples "n"))], Some [CmdPrint (ExpInt 0)])])
9   ])
```

---

## Nano10

Listagem 5.23: nano10.lua - Programa nano10 modificado

```

1 function int main()
2     local int n
3     local int m
4     n = 1
5     m = 2
6     if n == m then
7         print(n)
8     else
9         print(0)
10    end
11 end

```

---

Saida do analisador sintático:

Listagem 5.24: nano10.txt - Resultado do analisador sintático executado sobre nano10.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "n"); DecVar (TipoInt, VarSimples "m")],
6     [CmdAtrib (VarSimples "n", ExpInt 1);
7       CmdAtrib (VarSimples "m", ExpInt 2);
8       CmdIf
9         (ExpOp (Equivalente, ExpVar (VarSimples "n"), ExpVar (VarSimples "m"
10           "))),
11     [CmdPrint (ExpVar (VarSimples "n"))], Some [CmdPrint (ExpInt 0)]]])
12 ]

```

---

## Nano11

Listagem 5.25: nano11.lua - Programa nano11 modificado

```

1 function int main()
2     local int n
3     local int m
4     local int x
5     n = 1
6     m = 2
7     x = 5
8     while x > n do
9         n = n + m
10        print(n)
11    end
12 end

```

---

Saida do analisador sintático:

Listagem 5.26: nano11.txt - Resultado do analisador sintático executado sobre nano11.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "n"); DecVar (TipoInt, VarSimples "m");
6       DecVar (TipoInt, VarSimples "x")],

```



```

7      [CmdAtrib (VarSimples "n", ExpInt 1);
8      CmdAtrib (VarSimples "m", ExpInt 2);
9      CmdAtrib (VarSimples "x", ExpInt 5);
10     CmdWhile
11     (ExpOp (Maior, ExpVar (VarSimples "x"), ExpVar (VarSimples "n")),
12     [CmdAtrib (VarSimples "n",
13     ExpOp (Adicao, ExpVar (VarSimples "n"), ExpVar (VarSimples "m"))
14     ;
15     CmdPrint (ExpVar (VarSimples "n"))]]))]

```

## Nano12

Listagem 5.27: nano12.lua - Programa nano12 modificado

```

1  function int main()
2      local int n
3      local int m
4      local int x
5      n = 1
6      m = 2
7      x = 5
8      while x > n do
9          if n == m then
10             print(n)
11          else
12             print(0)
13          end
14          x = x - 1
15      end
16 end

```

Saida do analisador sintático:

Listagem 5.28: nano12.txt - Resultado do analisador sintático executado sobre nano12.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4  [Funcao (TipoInt, "main", [],
5  [DecVar (TipoInt, VarSimples "n"); DecVar (TipoInt, VarSimples "m");
6  DecVar (TipoInt, VarSimples "x")],
7  [CmdAtrib (VarSimples "n", ExpInt 1);
8  CmdAtrib (VarSimples "m", ExpInt 2);
9  CmdAtrib (VarSimples "x", ExpInt 5);
10 CmdWhile
11 (ExpOp (Maior, ExpVar (VarSimples "x"), ExpVar (VarSimples "n")),
12 [CmdIf
13 (ExpOp (Equivalente, ExpVar (VarSimples "n"),
14 ExpVar (VarSimples "m")),
15 [CmdPrint (ExpVar (VarSimples "n"))], Some [CmdPrint (ExpInt 0)])
16 ;
17 CmdAtrib (VarSimples "x",
18 ExpOp (Subtracao, ExpVar (VarSimples "x"), ExpInt 1))]]))]

```

## 5.6.2 Programas micro

### micro01

Listagem 5.29: micro01.lua - Programa micro01 modificado

```

1 function int main()
2     local int cel
3     local int far
4     print(" tabela de conversao: Celsius -> Fahrenheit\n")
5     print("Digite a temperatura em Celsius: ")
6     cel = io.read()
7     far = (9*cel + 160)/5
8     print("A nova temperatura e: " ..far.." F")
9 end

```

Saida do analisador sintático:

Listagem 5.30: micro01.txt - Resultado do analisador sintático executado sobre micro01.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "cel"); DecVar (TipoInt, VarSimples "far
6     ")]],
7     [CmdPrint (ExpString " tabela de conversao: Celsius -> Fahrenheit\n")
8     ;
9     CmdPrint (ExpString "Digite a temperatura em Celsius: ");
10    CmdScan (VarSimples "cel");
11    CmdAtrib (VarSimples "far",
12      ExpOp (Divisao,
13        ExpOp (Adicao,
14          ExpOp (Multiplicacao, ExpInt 9, ExpVar (VarSimples "cel")),
15          ExpInt 160),
16        ExpInt 5));
17    CmdPrint
18      (ExpOp (Concatena,
19        ExpOp (Concatena, ExpString "A nova temperatura e: ",
20          ExpVar (VarSimples "far")),
21        ExpString " F")))]])

```

### micro02

Listagem 5.31: micro02.lua - Programa micro02 modificado

```

1 function int main()
2     local int num1
3     local int num2
4     print("Digite o primeiro numero: ")
5     num1 = io.read()
6     print("Digite o segundo numero: ")
7     num2 = io.read()
8
9     if num1 > num2 then
10         print("O primeiro número " ..num1.." é maior que o segundo " ..num2)

```

```

11     else
12         print("O segundo número "..num2.." é maior que o primeiro "..num1)
13     end
14 end

```

Saida do analisador sintático:

Listagem 5.32: micro02.txt - Resultado do analisador sintático executado sobre micro02.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "num1");
6     DecVar (TipoInt, VarSimples "num2")],
7     [CmdPrint (ExpString "Digite o primeiro numero: ");
8     CmdScan (VarSimples "num1");
9     CmdPrint (ExpString "Digite o segundo numero: ");
10    CmdScan (VarSimples "num2");
11    CmdIf
12      (ExpOp (Maior, ExpVar (VarSimples "num1"), ExpVar (VarSimples "num2"
13      "))),
14    [CmdPrint
15      (ExpOp (Concatena,
16        ExpOp (Concatena,
17          ExpOp (Concatena, ExpString "O primeiro n\195\186mero ",
18            ExpVar (VarSimples "num1")),
19          ExpString " \195\169 maior que o segundo "),
20          ExpVar (VarSimples "num2")))],
21    Some
22      [CmdPrint
23        (ExpOp (Concatena,
24          ExpOp (Concatena,
25            ExpOp (Concatena, ExpString "O segundo n\195\186mero ",
26              ExpVar (VarSimples "num2")),
27            ExpString " \195\169 maior que o primeiro "),
28            ExpVar (VarSimples "num1")))]))]])

```

## micro03

Listagem 5.33: micro03.lua - Programa micro03 modificado

```

1 function int main()
2   local int numero
3   print("Digite um numero: ")
4   numero = io.read()
5   if numero >= 100 then
6     if numero <= 200 then
7       print("O numero esta no intervalo entre 100 e 200\n")
8     else
9       print("O numero nao esta no intervalo entre 100 e 200\n")
10    end
11  else
12    print("O numero nao esta no intervalo entre 100 e 200\n")
13  end
14 end

```

Saida do analisador sintático:

Listagem 5.34: micro03.txt - Resultado do analisador sintático executado sobre micro03.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "numero")],
5     [CmdPrint (ExpString "Digite um numero: ");
6       CmdScan (VarSimples "numero");
7       CmdIf
8         (ExpOp (Maior_ou_Igual, ExpVar (VarSimples "numero"), ExpInt 100),
9           [CmdIf
10            (ExpOp (Menor_ou_Igual, ExpVar (VarSimples "numero"), ExpInt 200)
11              [CmdPrint (ExpString "O numero esta no intervalo entre 100 e 200\
12                n")],
13              Some
14                [CmdPrint
15                  (ExpString "O numero nao esta no intervalo entre 100 e 200\n")
16                ])],
17              Some
18                [CmdPrint
19                  (ExpString "O numero nao esta no intervalo entre 100 e 200\n")
20                ])]),
21           [CmdPrint
22             (ExpString "O numero nao esta no intervalo entre 100 e 200\n")
23           ])],
24   ])]

```

## micro04

Listagem 5.35: micro04.lua - Programa micro04 modificado

```

1 function int main()
2   local int x
3   local int num
4   local int intervalo
5   intervalo = 0
6   for x = 1, 5, 1
7   do
8     print("Digite um numero: ")
9     num = io.read()
10    if num >= 10 then
11      if num <= 150 then
12        intervalo = intervalo + 1
13      end
14    end
15  end
16  print("Ao total, foram digitados "..intervalo.." numeros no intervalo
17  entre 10 e 150")
18 end

```

Saida do analisador sintático:

Listagem 5.36: micro04.txt - Resultado do analisador sintático executado sobre micro04.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],

```

```

5      [DecVar (TipoInt, VarSimples "x"); DecVar (TipoInt, VarSimples "num")
6      ;
7      DecVar (TipoInt, VarSimples "intervalo")],
8      [CmdAtrib (VarSimples "intervalo", ExpInt 0);
9      CmdFor (VarSimples "x", 1, 5, 1,
10     [CmdPrint (ExpString "Digite um numero: ");
11     CmdScan (VarSimples "num");
12     CmdIf (ExpOp (Maior_ou_Igual, ExpVar (VarSimples "num"), ExpInt
13         10),
14         [CmdIf
15             (ExpOp (Menor_ou_Igual, ExpVar (VarSimples "num"), ExpInt 150),
16             [CmdAtrib (VarSimples "intervalo",
17                 ExpOp (Adicao, ExpVar (VarSimples "intervalo"), ExpInt 1)]),
18             None)],
19         None)]];
20 CmdPrint
21 (ExpOp (Concatena,
22     ExpOp (Concatena, ExpString "Ao total, foram digitados ",
23         ExpVar (VarSimples "intervalo")),
24     ExpString " numeros no intervalo entre 10 e 150")))]])

```

## micro05

Listagem 5.37: micro05.lua - Programa micro05 modificado

```

1  function main()
2      local string nome
3      local string sexo
4      local int x
5      local int h
6      local int m
7      x = 1
8      h = 0
9      m = 0
10     for x = 1, 5, 1 do
11         print("Digite o nome: ")
12         nome = io.read()
13         print("H - Homem ou M - Mulher: ")
14         sexo = io.read()
15         if sexo == "H" then
16             h = h + 1
17         else
18             if sexo == "M" then
19                 m = m + 1
20             else
21                 print("Sexo so pode ser H ou M!\n")
22             end
23         end
24     end
25     print("Foram inseridos "..h.." homens\n")
26     print("Foram inseridos "..m.." mulheres\n")
27 end

```

Saida do analisador sintático:

Listagem 5.38: micro05.txt - Resultado do analisador sintático executado sobre micro05.lua modificado

```

1 - : Ast.programa option =

```

```

2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoString, VarSimples "nome");
6     DecVar (TipoString, VarSimples "sexo");
7     DecVar (TipoInt, VarSimples "x"); DecVar (TipoInt, VarSimples "h");
8     DecVar (TipoInt, VarSimples "m")],
9     [CmdAtrib (VarSimples "x", ExpInt 1);
10    CmdAtrib (VarSimples "h", ExpInt 0);
11    CmdAtrib (VarSimples "m", ExpInt 0);
12    CmdFor (VarSimples "x", 1, 5, 1,
13      [CmdPrint (ExpString "Digite o nome: "); CmdScan (VarSimples "nome"
14        );
15      CmdPrint (ExpString "H - Homem ou M - Mulher: ");
16      CmdScan (VarSimples "sexo");
17      CmdIf
18        (ExpOp (Equivalente, ExpVar (VarSimples "sexo"), ExpString "H"),
19        [CmdAtrib (VarSimples "h",
20          ExpOp (Adicao, ExpVar (VarSimples "h"), ExpInt 1))],
21        Some
22          [CmdIf
23            (ExpOp (Equivalente, ExpVar (VarSimples "sexo"), ExpString "M"
24              ),
25            [CmdAtrib (VarSimples "m",
26              ExpOp (Adicao, ExpVar (VarSimples "m"), ExpInt 1))],
27            Some [CmdPrint (ExpString "Sexo so pode ser H ou M!\n")])])]);
28    CmdPrint
29      (ExpOp (Concatena,
30        ExpOp (Concatena, ExpString "Foram inseridos ",
31          ExpVar (VarSimples "h")),
32        ExpString " homens\n"));
33    CmdPrint
34      (ExpOp (Concatena,
35        ExpOp (Concatena, ExpString "Foram inseridos ",
36          ExpVar (VarSimples "m")),
37        ExpString " mulheres\n")))]))];

```

## micro06

Listagem 5.39: micro06.lua - Programa micro06 modificado

```

1 function int main()
2   local int numero
3   print("Digite um numero de 1 a 5: ")
4   numero = io.read()
5   if numero == 1 then
6     print("Um\n")
7   end
8   if numero == 2 then
9     print("Dois\n")
10  end
11  if numero == 3 then
12    print("Tres\n")
13  end
14  if numero == 4 then
15    print("Quatro\n")
16  end
17  if numero == 5 then

```

```

18     print("Cinco\n")
19 else
20     print("Numero invalido!!!")
21 end
22 end

```

Saida do analisador sintático:

Listagem 5.40: micro06.txt - Resultado do analisador sintático executado sobre micro06.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "numero")],
5     [CmdPrint (ExpString "Digite um numero de 1 a 5: ");
6       CmdScan (VarSimples "numero");
7       CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 1),
8         [CmdPrint (ExpString "Um\n")], None);
9       CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 2),
10        [CmdPrint (ExpString "Dois\n")], None);
11      CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 3),
12        [CmdPrint (ExpString "Tres\n")], None);
13      CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 4),
14        [CmdPrint (ExpString "Quatro\n")], None);
15      CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 5),
16        [CmdPrint (ExpString "Cinco\n")],
17      Some [CmdPrint (ExpString "Numero invalido!!!")])])])

```

## micro07

Listagem 5.41: micro07.lua - Programa micro07 modificado

```

1 function int main()
2   local int programa
3   local int numero
4   local int opc
5   programa = 1
6   while programa == 1 do
7     print("Digite um numero: ")
8     numero = io.read()
9     if numero > 0 then
10      print("Positivo\n")
11    else
12      if numero == 0 then
13        print("O numero e igual a 0\n")
14      end
15      if numero < 0 then
16        print("Negativo\n")
17      end
18    end
19    print("Deseja finalizar? (S - 1): ")
20    opc = io.read()
21    if opc == 1 then
22      programa = 0
23    end
24  end
25 end

```

Saída do analisador sintático:

Listagem 5.42: micro07.txt - Resultado do analisador sintático executado sobre micro07.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "programa");
6     DecVar (TipoInt, VarSimples "numero");
7     DecVar (TipoInt, VarSimples "opc")],
8   [CmdAtrib (VarSimples "programa", ExpInt 1);
9   CmdWhile
10    (ExpOp (Equivalente, ExpVar (VarSimples "programa"), ExpInt 1),
11   [CmdPrint (ExpString "Digite um numero: ");
12   CmdScan (VarSimples "numero");
13   CmdIf (ExpOp (Maior, ExpVar (VarSimples "numero"), ExpInt 0),
14    [CmdPrint (ExpString "Positivo\n")],
15    Some
16    [CmdIf
17     (ExpOp (Equivalente, ExpVar (VarSimples "numero"), ExpInt 0),
18     [CmdPrint (ExpString "O numero e igual a 0\n")], None);
19     CmdIf (ExpOp (Menor, ExpVar (VarSimples "numero"), ExpInt 0),
20     [CmdPrint (ExpString "Negativo\n")], None)]];
21   CmdPrint (ExpString "Deseja finalizar? (S - 1): ");
22   CmdScan (VarSimples "opc");
23   CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "opc"), ExpInt 1),
24    [CmdAtrib (VarSimples "programa", ExpInt 0)], None)]]))])

```

## micro08

Listagem 5.43: micro08.lua - Programa micro08 modificado

```

1 function int main()
2   local int numero
3   numero = 1
4   while numero ~= 0 do
5     print("Digite um numero: ")
6     numero = io.read()
7     if numero > 10 then
8       print("O numero "..numero.." e maior que 10\n")
9     else
10      print("O numero "..numero.." e menor que 10\n")
11    end
12  end
13 end

```

Saída do analisador sintático:

Listagem 5.44: micro08.txt - Resultado do analisador sintático executado sobre micro08.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [], [DecVar (TipoInt, VarSimples "numero")],
5   [CmdAtrib (VarSimples "numero", ExpInt 1);
6   CmdWhile

```



```

7      (ExpOp (Nao_Equivalente, ExpVar (VarSimples "numero"), ExpInt 0),
8      [CmdPrint (ExpString "Digite um numero: ");
9      CmdScan (VarSimples "numero");
10     CmdIf (ExpOp (Maior, ExpVar (VarSimples "numero"), ExpInt 10),
11     [CmdPrint
12     (ExpOp (Concatena,
13     ExpOp (Concatena, ExpString "O numero ",
14     ExpVar (VarSimples "numero")),
15     ExpString " e maior que 10\n"))],
16     Some
17     [CmdPrint
18     (ExpOp (Concatena,
19     ExpOp (Concatena, ExpString "O numero ",
20     ExpVar (VarSimples "numero")),
21     ExpString " e menor que 10\n"))]]]]))

```

## micro09

Listagem 5.45: micro09.lua - Programa micro09 modificado

```

1  function main()
2      local int preco
3      local int venda
4      local int novo_preco
5      print("Digite o preco: ")
6      preco = io.read()
7      print("Digite a venda: ")
8      venda = io.read()
9      if venda < 500 or preco < 30 then
10         novo_preco = preco + 10 / 100 * preco
11     else if (venda >= 500 and venda <= 1200) or (preco >= 30 and preco < 80)
12         then
13         novo_preco = preco + 15 / 100 * preco
14     else if venda >= 1200 or preco >= 80 then
15         novo_preco = preco - 20 / 100 * preco
16     end
17     end
18     print("O novo preco e "..novo_preco.."\\n")
19 end

```

Saida do analisador sintático:

Listagem 5.46: micro09.txt - Resultado do analisador sintático executado sobre micro09.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "preco");
6     DecVar (TipoInt, VarSimples "venda");
7     DecVar (TipoInt, VarSimples "novo_preco")],
8     [CmdPrint (ExpString "Digite o preco: "); CmdScan (VarSimples "preco"
9     );
10     CmdPrint (ExpString "Digite a venda: "); CmdScan (VarSimples "venda"
11     );
12     CmdIf

```

```

11      (ExpOp (Or, ExpOp (Menor, ExpVar (VarSimples "venda"), ExpInt 500),
12        ExpOp (Menor, ExpVar (VarSimples "preco"), ExpInt 30)),
13      [CmdAtrib (VarSimples "novo_preco",
14        ExpOp (Adicao, ExpVar (VarSimples "preco"),
15        ExpOp (Multiplicacao, ExpOp (Divisao, ExpInt 10, ExpInt 100),
16        ExpVar (VarSimples "preco")))]],
17    Some
18      [CmdIf
19        (ExpOp (Or,
20          ExpOp (And,
21            ExpOp (Maior_ou_Igual, ExpVar (VarSimples "venda"), ExpInt
22              500),
23            ExpOp (Menor_ou_Igual, ExpVar (VarSimples "venda"), ExpInt
24              1200)),
25          ExpOp (And,
26            ExpOp (Maior_ou_Igual, ExpVar (VarSimples "preco"), ExpInt
27              30),
28            ExpOp (Menor, ExpVar (VarSimples "preco"), ExpInt 80))),
29        [CmdAtrib (VarSimples "novo_preco",
30          ExpOp (Adicao, ExpVar (VarSimples "preco"),
31          ExpOp (Multiplicacao, ExpOp (Divisao, ExpInt 15, ExpInt 100),
32          ExpVar (VarSimples "preco")))]],
33        Some
34          [CmdIf
35            (ExpOp (Or,
36              ExpOp (Maior_ou_Igual, ExpVar (VarSimples "venda"),
37                ExpInt 1200),
38              ExpOp (Maior_ou_Igual, ExpVar (VarSimples "preco"), ExpInt
39                80)),
40            [CmdAtrib (VarSimples "novo_preco",
41              ExpOp (Subtracao, ExpVar (VarSimples "preco"),
42              ExpOp (Multiplicacao, ExpOp (Divisao, ExpInt 20, ExpInt
43                100),
44              ExpVar (VarSimples "preco")))]],
45            None)]])];
46    CmdPrint
47      (ExpOp (Concatena,
48        ExpOp (Concatena, ExpString "O novo preco e ",
49        ExpVar (VarSimples "novo_preco")),
50        ExpString "\n")))]])

```

## micro10

Listagem 5.47: micro10.lua - Programa micro10 modificado

```

1 function int fatorial(int n)
2   local int x
3
4   if n <= 1 then
5     return 1
6   else
7     y = n - 1
8     x = fatorial (y)
9     return n * x
10  end
11 end
12
13 function int main()

```

```

14  local int numero
15  local int fat
16  print("Digite um numero: ")
17  numero = io.read()
18  fat = fatorial(numero)
19  print("O fatorial de ")
20  print("numero")
21  print(" e ")
22  print(fat)
23
24  return 1
25 end

```

Saida do analisador sintático:

Listagem 5.48: micro10.txt - Resultado do analisador sintático executado sobre micro10.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "fatorial", [Args (TipoInt, "n")],
5     [DecVar (TipoInt, VarSimples "x")],
6     [CmdIf (ExpOp (Menor_ou_Igual, ExpVar (VarSimples "n"), ExpInt 1),
7       [CmdRetorno (ExpInt 1)],
8       Some
9         [CmdAtrib (VarSimples "y",
10           ExpOp (Subtracao, ExpVar (VarSimples "n"), ExpInt 1));
11           CmdAtribRetorno (VarSimples "x", "fatorial", ["y"]);
12           CmdRetorno
13             (ExpOp (Multiplicacao, ExpVar (VarSimples "n"),
14               ExpVar (VarSimples "x")))]))]];
15   Funcao (TipoInt, "main", [],
16     [DecVar (TipoInt, VarSimples "numero");
17     DecVar (TipoInt, VarSimples "fat")],
18     [CmdPrint (ExpString "Digite um numero: ");
19     CmdScan (VarSimples "numero");
20     CmdAtribRetorno (VarSimples "fat", "fatorial", ["numero"]);
21     CmdPrint (ExpString "O fatorial de "); CmdPrint (ExpString "numero")
22     ;
23     CmdPrint (ExpString " e "); CmdPrint (ExpVar (VarSimples "fat"));
24     CmdRetorno (ExpInt 1)]])

```

## micro11

Listagem 5.49: micro11.lua - Programa micro11 modificado

```

1 function int main()
2   local int numero
3   local int x
4   print("Digite um numero ")
5   numero = io.read()
6   x = verifica(numero)
7   if x == 1 then
8     print("Numero positivo\n")
9   else if x == 0 then
10    print("Zero\n")
11  else

```

```

12     print("Numero negativo\n")
13 end
14 end
15 end
16
17 function int verifica(int n)
18     local int res
19     if n > 0 then
20         res = 1
21     else if n < 0 then
22         res = -1
23     else
24         res = 0
25     end
26 end
27 return res
28 end

```

Saida do analisador sintático:

Listagem 5.50: micro11.txt - Resultado do analisador sintático executado sobre micro11.lua modificado

```

1 - : Ast.programa option =
2 Some
3 (Programa
4   [Funcao (TipoInt, "main", [],
5     [DecVar (TipoInt, VarSimples "numero");
6     DecVar (TipoInt, VarSimples "x")],
7     [CmdPrint (ExpString "Digite um numero ");
8     CmdScan (VarSimples "numero");
9     CmdAtribRetorno (VarSimples "x", "verifica", ["numero"]);
10    CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "x"), ExpInt 1),
11      [CmdPrint (ExpString "Numero positivo\n")],
12      Some
13        [CmdIf (ExpOp (Equivalente, ExpVar (VarSimples "x"), ExpInt 0),
14          [CmdPrint (ExpString "Zero\n")],
15          Some [CmdPrint (ExpString "Numero negativo\n")])])]),
16    Funcao (TipoInt, "verifica", [Args (TipoInt, "n")],
17      [DecVar (TipoInt, VarSimples "res")],
18      [CmdIf (ExpOp (Maior, ExpVar (VarSimples "n"), ExpInt 0),
19        [CmdAtrib (VarSimples "res", ExpInt 1)],
20        Some
21          [CmdIf (ExpOp (Menor, ExpVar (VarSimples "n"), ExpInt 0),
22            [CmdAtrib (VarSimples "res", ExpInt (-1))],
23            Some [CmdAtrib (VarSimples "res", ExpInt 0)])]),
24      CmdRetorno (ExpVar (VarSimples "res"))])])

```

## 5.7 Testes de Erros Sintáticos

A seguir são alguns dos erros sintáticos que podem ocorrer e gerar uma exceção pelo analisador sintático

### Comandos fora do escopo de uma função

## Listagem 5.51: teste01.lua - Programa de teste 01

```
1 while
```

Saida do analisador:

## Listagem 5.52: saida01.txt - Resultado do analisador sintático executado sobre teste01.lua modificado

```
1 Erro sintatico na linha 1, coluna 4 0 - "Funcao nao definida"
2 .
3
4 Exception: Failure "A analise sintatica falhou".
```

## Função com estrutura incorreta

## Listagem 5.53: teste02.lua - Programa de teste 02

```
1 function while
```

Saida do analisador:

## Listagem 5.54: saida02.txt - Resultado do analisador sintático executado sobre teste02.lua modificado

```
1 Erro sintatico na linha 1, coluna 13 9 - "Function espera o seguinte
   formato: 'function tipo nome (argumentos)'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".
```

## Declaração de variável incorreta

## Listagem 5.55: teste03.lua - Programa de teste 03

```
1 function int main()
2     int
3 end
```

Saida do analisador:

## Listagem 5.56: saida03.txt - Resultado do analisador sintático executado sobre teste03.lua modificado

```
1 Erro sintatico na linha 3, coluna 2 28 - "Comando declaracao espera o
   formato: 'tipo_string tipo'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".
```

## Atribuição incorreta de variável

## Listagem 5.57: teste04.lua - Programa de teste 04

```
1 function int main()
2     int x
```

```

3
4     x =
5
6 end

```

---

Saida do analisador:

Listagem 5.58: saida04.txt - Resultado do analisador sintático executado sobre teste01.lua modificado

```

1 Erro sintatico na linha 6, coluna 2 40 - "Atribuicao espera formato: '
    variavel ATRIB'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".

```

---

## Formato incorreto: comando IF

Listagem 5.59: teste05.lua - Programa de teste 05

```

1 function int main()
2     int x
3     x = 9
4     if x == 9 then
5
6 end

```

---

Saida do analisador:

Listagem 5.60: saida05.txt - Resultado do analisador sintático executado sobre teste05.lua modificado

```

1 Erro sintatico na linha 6, coluna 2 60 - "Comando if espera o seguinte
    formato: 'if expressao then comandos else comandos end'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".

```

---

## Formato incorreto: comando FOR

Listagem 5.61: teste06.lua - Programa de teste 06

```

1 function int main()
2     int x
3     int y
4
5     x = 9
6
7     for y = 0, 5, 1
8         x = x + 1
9     end
10
11 end

```

---

Saida do analisador:

Listagem 5.62: saida06.txt - Resultado do analisador sintático executado sobre teste06.lua modificado

```
1 Erro sintatico na linha 8, coluna 8 80 - "For espera o seguinte formato: '
    for variavel atrib litint virgula litint virgula litint do comandos end
    '".
2
3 Exception: Failure "A analise sintatica falhou".
```

---

## Formato incorreto: comando WHILE

Listagem 5.63: teste07.lua - Programa de teste 07

```
1 function int main()
2
3     int x
4     x = 9
5
6     while if do
7         x = x + 1
8     end
9 end
```

---

Saida do analisador:

Listagem 5.64: saida07.txt - Resultado do analisador sintático executado sobre teste07.lua modificado

```
1 Erro sintatico na linha 6, coluna 11 52 - "While espera o seguinte formato
    : 'while expressao do comandos end'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".
```

---

## Formato incorreto: retorno de função

Listagem 5.65: teste08.lua - Programa de teste 08

```
1 function int main()
2     int x
3     x = 9
4
5     return
6
7 end
```

---

Saida do analisador:

Listagem 5.66: saida08.txt - Resultado do analisador sintático executado sobre teste08.lua modificado

```
1 Erro sintatico na linha 7, coluna 2 54 - "Comando return espera o formato:
    'return expressao'"
2 .
3
4 Exception: Failure "A analise sintatica falhou".
```

---