



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

Construção de Compiladores
Av. João Naves de Ávila 2121, Campus Santa Mônica



Relatório de Compiladores Python para Dalvik

Alunos:

Gabriel Augusto Marson - 11221BCC022

Leonardo da Silva Martins - 11321BCC034

E-mail: gabrielmarson@live.com

Profº.: Alexsandro

Sumário

1	Introdução	4
2	Configuração do Ambiente	4
2.1	Ubuntu 16	4
2.2	OCaml	4
2.3	Oracle Java JDK 8	4
2.4	Python	5
3	Android SDK	5
3.0.1	Configurando as variáveis de ambiente	6
3.0.2	Dispositivo Virtual Android	7
3.0.3	Criando um AVD	8
4	Dalvik	8
4.1	Dalvik e JVM	8
4.2	Smali/Backsmali	9
4.3	Execução de um Programa na Dalvik	10
5	Análise do comportamento de compilação de .java para .smali	11
5.1	Processo de Compilação	12
5.2	Análise dos exercícios fornecidos	13
5.2.1	Algoritmo nano01	13
5.2.2	Algoritmo nano02	15
5.2.3	Algoritmo nano03	16
5.2.4	Algoritmo nano04	18
5.2.5	Algoritmo nano05	20
5.2.6	Algoritmo nano06	22
5.2.7	Algoritmo nano07	24
5.2.8	Algoritmo nano08	26
5.2.9	Algoritmo testel_09	29
5.2.10	Algoritmo nano10	31
5.2.11	Algoritmo nano11	34
5.2.12	Algoritmo nano12	37
5.2.13	Algoritmo micro01	41
5.2.14	Algoritmo micro02	45

SUMÁRIO

5.2.15	Algoritmo micro03	53
5.2.16	Algoritmo micro04	57
5.2.17	Algoritmo micro05	63
5.2.18	Algoritmo micro06	74
5.2.19	Algoritmo micro07	79
5.2.20	Algoritmo micro08	86
5.2.21	Algoritmo micro09	92
5.2.22	Algoritmo micro10	101
5.2.23	Algoritmo micro11	106
6	Analisador Léxico	113
6.1	Abordagem por Autômato	113
6.1.1	Implementação	114
6.1.2	Testes	120
6.2	Abordagem por Linguagem Regular	123
6.2.1	Implementação	123
6.2.2	Testes	124
6.2.3	Teste de Comentários	152
7	Analisador Sintático	154
7.1	Teste de Gramática	154
7.2	Códigos Fonte	156
8	Sintático Usando Menhir	160
8.1	Testes	160
9	Analisador Semântico	165
9.1	Testes	166
9.2	Testes Com Erros	173
10	Interpretador Usando Menhir	175
10.1	Execução	175
10.2	Testes	176
11	Erros gerados pelo interpretador	184
	Apêndice	186

SUMÁRIO

12 Referências	232
12.1 Bibliográficas	232
12.2 Webgráficas	232

1 Introdução

Esse relatório contém informações à respeito da instalação das tecnologias necessárias(Dalvik, OCaml, Python, etc) para o processo de construção de compiladores. Além disso, procurou-se extrair informações à respeito das regras que o compilador Dalvik usa para processar as linguagens.

2 Configuração do Ambiente

Nesta seção, será apresentado uma breve descrição sobre as ferramentas que serão usadas e como configurá-las para o nosso experimento.

2.1 Ubuntu 16

Ubuntu é um sistema operacional com núcleo do linux. Pode-se baixá-lo no [site oficial](#) clicando em **Ubuntu Desktop**. A instalação pode ser feita via CD ou via boot pelo pendrive.

Foi baixado e instalado o Ubuntu 16.04 LTS.

2.2 OCaml

Para instalar a linguagem OCaml no Ubuntu, basta digitar no terminal:

```
> sudo apt-get install ocaml
```

A versão instalada do OCaml foi a 4.02.3.

2.3 Oracle Java JDK 8

É recomendado usar o o JDK para o desenvolvimento Android. Para simplificar o download e instalação, precisaremos adicionar um PPA(Personal Package Archive) via linha de comando:

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

3 ANDROID SDK

Para verificarmos se o JDK foi instalado com sucesso, basta digitarmos o comando `java -version`. A saída esperada é algo semelhante a isto:

```
java version "1.8.0_101"  
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

2.4 Python

Criada por Guido van Rossum em 1991, Python é um linguagem de programação imperativa, orientada objetos e de tipagem dinâmica forte. O python já vem instalado no Ubuntu. A versão do Python utilizada foi a 3.5.2.

3 Android SDK

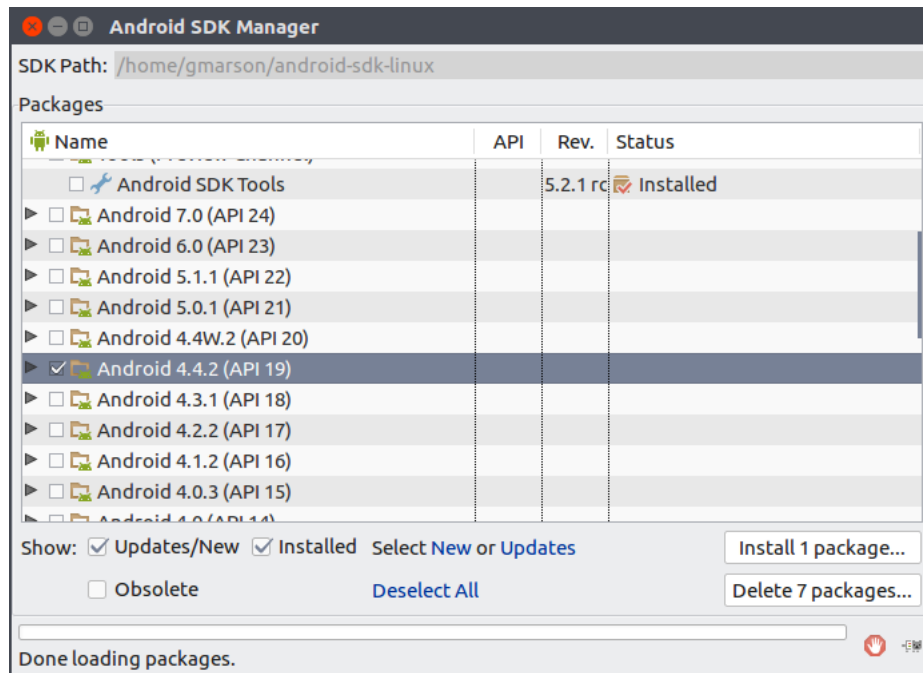
O Android SDK inclui ferramentas para que programadores consigam desenvolver as suas aplicações e, no caso desse trabalho, executar um código .smali. O download pode ser feito diretamente no [site](#). Pode-se baixar o Android Studio ou somente o Android SDK. Optamos pelo SDK.

Após baixar o arquivo, descompacte-o, acesse a pasta tools e digite.

```
> ./android
```

Será exibida uma janela com a opção de alguns arquivos default marcados para instalação. Clique no botão InstallPackages

3 ANDROID SDK



Recomenda-se o download de todas as tools e qualquer API do android. No caso, foi baixada a API 19.

3.0.1 Configurando as variáveis de ambiente

Edite o arquivo `bashrc` e acrescente as linhas a seguir salvando o arquivo posteriormente.

Listing 1: Variáveis android de ambiente

```
1 #Editando o bash
2 gedit ~/.bashrc
3
4 #Android
5 export PATH=${PATH}:~/android-sdk-linux/tools
6 export PATH=${PATH}:~/android-sdk-linux/platform-tools
7 export PATH=${PATH}:~/android-sdk-linux
```

Em seguida, basta recarregarmos o bash:

3 ANDROID SDK

```
> source ~/.bashrc
```

3.0.2 Dispositivo Virtual Android

O dispositivo Virtual Android é um emulador usado para testes de aplicações. Um Android Virtual Device (AVD) é um emulador que permite replicar um dispositivo real, especificando opções de hardware e software. Será usado um AVD para que se possa executar uma máquina Dalvik. Pelo comando:

```
> android list targets
```

É esperado que as APIs do android que foram baixadas apareçam listadas. Algo similar a esta imagem:

```
gmarson@gmarson-Inspiron-5547:~/android-sdk-linux$ android list targets
Available Android targets:
-----
id: 1 or "android-19"
  Name: Android 4.4.2
  Type: Platform
  API level: 19
  Revision: 4
  Skins: HVGA, QVGA, WQVGA400, WQVGA432, WSVGA, WVGA800 (default), WVGA854,
  WXGA720, WXGA800, WXGA800-7in
  Tag/ABIs : default/armeabi-v7a, default/x86, google_apis/armeabi-v7a, google_
  apis/x86
-----
id: 2 or "Google Inc.:Google APIs:19"
  Name: Google APIs
  Type: Add-On
  Vendor: Google Inc.
  Revision: 20
  Description: Android + Google APIs
  Based on Android 4.4.2 (API level 19)
  Libraries:
    * com.android.future.usb.accessory (usb.jar)
      API for USB Accessories
    * com.google.android.media.effects (effects.jar)
      Collection of video effects
    * com.google.android.maps (maps.jar)
      API for Google Maps
  Skins: HVGA, QVGA, WQVGA400, WQVGA432, WSVGA, WVGA800 (default), WVGA854,
  WXGA720, WXGA800, WXGA800-7in
  Tag/ABIs : no ABIs.
```


3.0.3 Criando um AVD

Para criar uma AVD, selecione um dos dispositivos previamente listados e digite o comando nesse formato: `android create avd -n < nomeDaMaquina > -t < APIMaquina >`. Segue o exemplo:

```
gnarson@gnarson-Inspiron-5547:~/teste$ android create avd -n meuAndroid19 -t a
ndroid-19 --abi default/armeabi-v7a
Android 4.4.2 is a basic Android platform.
Do you wish to create a custom hardware profile [no]no
Created AVD 'meuAndroid19' based on Android 4.4.2, ARM (armeabi-v7a) processor
with the following hardware config:
hw.lcd.density=240
hw.ramSize=512
vm.heapSize=48
gnarson@gnarson-Inspiron-5547:~/teste$
```

Seguem alguns comandos úteis:

Ver : `emulator -avd meuAndroid19`

Deletar : `android delete avd -n meuAndroid19`

4 Dalvik

Dalvik é uma máquina virtual criada para o sistema Android e que emula operações de CPU e compila códigos de uma linguagem fonte para um byte-code específico da máquina virtual. Otimizada para dispositivos móveis ou com hardware limitado, Dalvik foi projetada de modo a permitir a execução de várias instâncias ao mesmo tempo de forma eficaz.

Por trás desta eficácia, escondem-se certas medidas que determinaram o bom funcionamento dessa VM(Virtual Machine). Dentre elas, convém mencionar o processo **Zygote** que é responsável pelo compartilhamento de código entre as instâncias de programa na Dalvik.

4.1 Dalvik e JVM

A diferença entre Dalvik e JVM está no fato de que as duas possuem bytecodes diferentes. Quando é criado um aplicativo java no computador, a JVM executa tudo o que foi compilado a partir do código fonte.

Na máquina Dalvik o processo ocorre de forma similar. A diferença é que existe um compilador **dx** que pega os arquivos `.class` e os transforma para



Figura 1: Processo de Compilação em JVM

.dex. Esse último formato é processado pelo aapt(Android Packaging Tool) e convertido em um arquivo .apk .

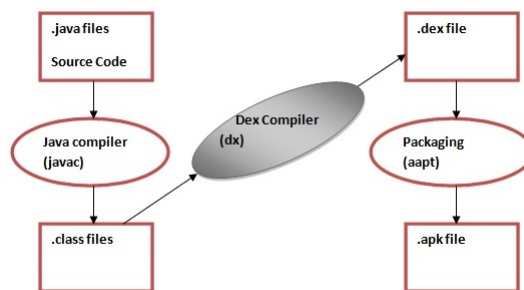


Figura 2: Processo de Compilação Dalvik

4.2 Smali/Backsmali

O assembler(programa que transforma um código Assembly, no caso, smali, para linguagem de máquina) da máquina virtual Dalvik é o Smali. O Backsmali é usado para converter um arquivo .dex em um código fonte .smali.

O download dos dois pode ser feito via bitbucket:

```
> cd
> mkdir Smali
> cd Smali
> wget https://bitbucket.org/JesusFreke/smali/downloads/smali-2.1.3.jar
> wget https://bitbucket.org/JesusFreke/smali/downloads/baksmali-2.1.3.jar
```

4.3 Execução de um Programa na Dalvik

Para exemplificar o uso do smali, usaremos o seguinte código adquirido de [2]:

Listing 2: Smali

```
1 .class public LHelloWorld
2
3 .super Ljava/lang/Object;
4 .method public static main([Ljava/lang/String;)V
5 .registers 2
6     sget-object v0,
7         Ljava/lang/System;-.>out:Ljava/io/PrintStream;
8     const-string v1, "Hello World!"
9     invoke-virtual {v0, v1},
10        Ljava/io/PrintStream;-.>println(Ljava/lang/String;)V
11     return-void
12 .end method
```

Deve-se primeiramente, salvar esse código no mesmo diretório em que estão o smali.jar e o backsmali.jar. Em seguida, digitaremos os seguintes códigos que são necessários para rodar esse programa na máquina android.

```
>cd ~/Dalvik
>java -jar smali-2.0.3.jar -o classes.dex HelloWorld.smali
> zip HelloWorld.zip classes.dex
```

É imperativo que o .dex gerado seja chamado de classes.dex. Agora, deve-se ligar o emulador do android. Relembrando que AVD significa android virtual device.

```
> emulator -avd meuAndroid19 &
```

Será usado o ADB(Android Debug Bridge) que é um comando versátil que permite a comunicação do sistema operacional com uma instância de emulador android. Esse comando diz para o SO colocar o arquivo HelloWorld.zip dentro do meuAndroid19.

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
> adb push HelloWorld.zip /data/local
```

Em seguida, basta executarmos:

```
> adb shell
> dalvikvm -cp /data/local/HelloWorld.zip HelloWorld
```

Esse comando executa a dalvikVM no arquivo HelloWorld.zip
A saída esperada é:

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/Dalvik$ adb push HelloWorld.zip
/data/local
[100%] /data/local/HelloWorld.zip
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/Dalvik$ adb shell
root@generic:/ # dalvikvm -cp /data/local/HelloWorld.zip HelloWorld
Hello World!
root@generic:/ #
```

Para fecharmos a emulação do Android, basta digitarmos o comando:

```
> adb emu kill
```

5 Análise do comportamento de compilação de .java para .smali

O processo de transformação de um arquivo java para um arquivo smile segue os seguintes passos:

1. Compila-se o arquivo .java com o compilador **javac**. Esse processo gera um arquivo .class.
2. Usa-se uma utilidade dx do android para transformar o arquivo .class em um arquivo .dex.
3. Utiliza-se o baksmali para transformar o arquivo .dex em .smali

Como não existe um compilador Python para Dalvik será feita uma análise de comportamento de Java para Dalvik utilizando o Smali/Baksmali. Dada a entrada de programas em java, será mostrado o equivalente em Smali bem como o processo que se deve fazer para alcançar esse objetivo. A partir desses dados poderá ser inferido o comportamento de compilação da máquina Dalvik.

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.1 Processo de Compilação

Segue um simples exemplo de código em java:

Listing 3: Hello World em Java

```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }
```

Para transformá-lo em smali devemos primeiro compilar o código a cima usando o comando `javac -source 1.4 -target .14 HelloWorld.java`. Source e Target são necessários para gerar compatibilidade com dx do android da API 19 uma vez que as versões mais recentes não são aceitas por este android. Após isso, colaremos o HelloWorld.java para a pasta onde está o utilitário dx. Em seguida, basta usarmos o dx do Android para transformar o .class em .dex.

```
> cd  
> #Diretorio onde estão os Códigos  
> cd GIT/Compiladores/CódigosTeste  
> cp HelloWorld.class ~/android-sdk-linux/build-tools/19.1.0  
> ./dx -dex -output=HelloWorld.dex HelloWorld.class
```

Lembrando que o HelloWorld.class tem que estar no mesmo diretório da ferramenta dx do Android.

OBS.: Para uma análise mais elaborada, será utilizado o comando `./dx --dex --no-optimize --output=HelloWorld.dex HelloWorld.class` pois assim a utilização dos registradores fica mais clara em virtude de que o código não será utilizado.

Finalmente, deve-se copiar o .dex para pasta onde está localizado o baksmali.jar. A função dele é transformar .dex para .smali. Isso pode ser feito com o seguinte comando.

```
> java -jar baksmali-2.1.3.jar -x HelloWorld.dex -o HelloWorld
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.2 Análise dos exercícios fornecidos

Nessa análise será feita uma comparação direta do código em java e do resultante em smali. Será mostrado, também, a saída do programa em java. Além disso, será observado como o código smali muda conforme forem feitas alterações no código java.

5.2.1 Algoritmo nano01

Listing 4: Código em Java

```
1 public class nano01
2 {
3     public static void main(String[] args)
4     {
5
6     }
7 }
```

Listing 5: Código em python

```
1 def nano01():
2     pass
```

Listing 6: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
13  
14     move-object v1, v0  
15  
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V  
17  
18     return-void  
19 .end method  
20  
21 .method public static main([Ljava/lang/String;)V  
22     .registers 1  
23  
24     .prologue  
25     .line 6  
26     return-void  
27 .end method
```

Pode-se perceber que o resultante em smali é grande considerando que praticamente nada tenha sido implementado no código em java. Seguem algumas considerações:

- A primeira linha (`.class`) indica qual a classe que estamos compilando que no caso é `nano01`.
- O `.super` indica de qual classe herda a nossa classe `nano01` (toda a classe em java herda de `Object`).
- o `.source` indica de qual arquivo referencia o `.smali`
- `.register` indica a quantidade de registradores usado no método
- `.prologue` indica que a próxima linha deverá marcar o fim do método.
- `.line` marca o fim do método. como podemos observar nesse código, existem dois `prologue` e dois `line`. O primeiro par é referente ao método construtor e o segundo é referente ao método estático `main`. Além disso ele é utilizado para `debugging` e `stacktraces`, ou seja, quando existir um erro no código, será mostrado o `.line` (a linha) em que esse erro ocorreu.
- Os registradores do tipo `p` são reservados para parâmetros dos métodos.

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

- Nas linhas onde são declarados os métodos, pode-se perceber um identificador para ver se o método é public, private, package seguido seguido pelo seu nome(no caso da linha 7 o nome é constructor < *init* >), seguidos pelos seus argumentos(para a linha 7 não há argumentos) e o tipo de retorno do método (no caso, V para void).
- Todos os métodos tem um comando que os finaliza, o `.end-method`.
- `move-object` é análogo ao `move`. A diferença é que passa a referência de um objeto.
- Existe um site contendo os Opcodes de Dalvik que pode ser acessado [aqui](#).

Pode-se notar o método construtor sendo declarado explicitamente na linha 7 e 16. Na chamada da linha 16 é enviado o registrador v1 que contém os parâmetros do método construtor da classe nano01 (que foram transferidos a v0 por p0 e depois para v1) além da classe de onde o método é proveniente. Por fim, ainda na mesma linha está o nome do método construtor de nano01, seus argumentos seguidos pelo tipo de retorno (V:void)

5.2.2 Algoritmo nano02

Listing 7: Código em Java

```
1 public class nano02
2 {
3     public static void main(String[] args)
4     {
5         int n;
6     }
7 }
```

Listing 8: Código em python

```
1 def nano02():
2     n = int(n)
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 9: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 1
23
24     .prologue
25     .line 6
26     return-void
27 .end method
```

Não se obteve nenhuma diferença com relação ao nano01.

5.2.3 Algoritmo nano03

Listing 10: Código em Java

```
1 public class nano03
2 {
3     public static void main(String[] args)
4     {
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
5      int n;  
6      n=1;  
7  }  
8 }
```

Listing 11: Código em python

```
1 def nano03():  
2     n = int(n)  
3     n=1
```

Listing 12: Smali resultante do .java

```
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 3  
9  
10    .prologue  
11    .line 1  
12    move-object v0, p0  
13  
14    move-object v1, v0  
15  
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V  
17  
18    return-void  
19 .end method  
20  
21 .method public static main([Ljava/lang/String;)V  
22     .registers 4  
23  
24     .prologue
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
25  .line 6
26  move-object v0, p0
27
28  const/4 v2, 0x1
29
30  move v1, v2
31
32  .line 7
33  return-void
34 .end method
```

Pode-se notar uma diferença quando atribui-se um valor a alguma variável. Podemos observar isso na linha 28, onde uma constante de 4 bytes (do tipo inteiro) é atribuída a um registrador v2 e depois movida para v1. Podemos saber que a constante é 1 por conta do que vem depois do registrador v2 em 0x1

Observe o uso dos .lines para ajudar em uma eventual construção de debugging ou stacktrace.

5.2.4 Algoritmo nano04

Listing 13: Código em Java

```
1 public class nano04
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n=1+2;
7     }
8 }
```

Listing 14: Código em python

```
1 def nano04():
2     n = int(n)
3     n = 1 + 2
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 15: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 4
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, 0x3
29
30     move v1, v2
31
32     .line 7
33     return-void
34 .end method
```

Pode-se inferir que quando a soma é entre duas constantes, o resultado

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

da soma já é atribuído como em um registrador(v2), ou seja, não foi feita nenhuma operação de soma nesse código smali.

5.2.5 Algoritmo nano05

Listing 16: Código em Java

```
1 public class nano05
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n=2;
7         System.out.print(n);
8     }
9 }
```

Listing 17: Código em python

```
1 def nano05():
2     n = 2
3     print(n,end="")
```

Listing 18: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
15
16     invoke-direct {v1}, Ljava/lang/Object; -> <init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 5
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, 0x2
29
30     move v1, v2
31
32     .line 7
33     sget-object v2,
34         Ljava/lang/System; -> out:Ljava/io/PrintStream;
35
36     move v3, v1
37
38     invoke-virtual {v2, v3}, Ljava/io/PrintStream; -> print(I)V
39
40     .line 8
41     return-void
42 .end method
```

Pode-se observar o **sget-object v2 ...** na linha 33. Isso quer dizer que em v2 será guardado uma referência para a classe `PrintStream`.

Na linha 37 são mandados os registradores v3 (possui o resultado da atribuição) e v2 que é uma referência para `PrintStream`. Essa linha é responsável por exibir o dado na tela fazendo com que na classe `PrintStream` seja invocado o método `print` onde é mandado um inteiro como argumento e esperado um void como retorno.

Saída

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

2

Obs.: Sem quebra de linha

5.2.6 Algoritmo nano06

Listing 19: Código em Java

```
1 public class nano06
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 -2;
7         System.out.print(n);
8     }
9 }
```

Listing 20: Código em python

```
1 def nano06():
2     n = 1 - 2
3     print(n,end="")
```

Listing 21: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 5
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, -0x1
29
30     move v1, v2
31
32     .line 7
33     sget-object v2,
34         Ljava/lang/System;->out:Ljava/io/PrintStream;
35
36     move v3, v1
37
38     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(I)V
39
40     .line 8
41     return-void
42 .end method
```

Na linha 28, pode-se ver o prefixo de sinal negativo em `-0x1` indicando que será atribuído uma variável negativa ao registrador `v2`.

Saída

-1

Obs.: Sem quebra de linha

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.2.7 Algoritmo nano07

Listing 22: Código em Java

```
1 public class nano07
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 ;
7         if(n == 1){
8             System.out.print(n);
9         }
10    }
11 }
```

Listing 23: Código em python

```
1 def nano07():
2     n=1
3     if n ==1:
4         print(n,end="")
```

Listing 24: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 5
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, 0x1
29
30     move v1, v2
31
32     .line 7
33     move v2, v1
34
35     const/4 v3, 0x1
36
37     if-ne v2, v3, :cond_d
38
39     .line 8
40     sget-object v2,
41         Ljava/lang/System;->out:Ljava/io/PrintStream;
42
43     move v3, v1
44
45     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(I)V
46
47     .line 10
48     :cond_d
49     return-void
50 .end method
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Em v2 e v3 são assinalados os mesmo valores. Na linha 37 há um novo comando `if-ne` o que corresponde a if-not-equal. Essa linha verifica se os registradores são iguais, se sim, então imprime, caso contrário há um marcador `:cond_d` para o qual o `if-ne` direciona o programa.

Saída

1

Obs.: Sem quebra de linha

5.2.8 Algoritmo nano08

Listing 25: Código em Java

```
1 public class nano08
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 ;
7         if(n == 1){
8             System.out.print(n);
9         }
10        else{
11            System.out.print(0);
12        }
13    }
14 }
```

Listing 26: Código em python

```
1 def nano08():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5     else:
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
6 | print(0,end="")
```

Listing 27: Smali resultante do .java

```
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 3
9
10     .prologue
11     .line 1
12     move-object v0, p0
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 5
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, 0x1
29
30     move v1, v2
31
32     .line 7
33     move v2, v1
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
34
35     const/4 v3, 0x1
36
37     if-ne v2, v3, :cond_e
38
39     .line 8
40     sget-object v2,
41         Ljava/lang/System; ->out:Ljava/io/PrintStream;
42
43     move v3, v1
44
45     invoke-virtual {v2, v3}, Ljava/io/PrintStream; ->print(I)V
46
47     .line 13
48     :goto_d
49     return-void
50
51     .line 11
52     :cond_e
53     sget-object v2,
54         Ljava/lang/System; ->out:Ljava/io/PrintStream;
55
56     const/4 v3, 0x0
57
58     invoke-virtual {v2, v3}, Ljava/io/PrintStream; ->print(I)V
59
60     goto :goto_d
61 .end method
```

Percebe-se que o if-else é tratado apenas com if em smali. É verificado se $v2 = v3$ e dependendo do resultado o programa é direcionado para imprimir zero ou um por meio de tags nos programas. O comando `goto` tem essa função.

Saída

1

Obs.: Sem quebra de linha

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.2.9 Algoritmo testel_09

Listing 28: Código em Java

```
1 public class teste1_09
2 {
3     public static void main(String[] args)
4     {
5         int n;
6         n= 1 +1 /2;
7         if(n == 1){
8             System.out.print(n);
9         }
10        else{
11            System.out.print(0);
12        }
13    }
14 }
```

Listing 29: Código em python

```
1 def teste1_9():
2     n=1 + 1/2
3     if n ==1:
4         print(n,end="")
5     else:
6         print(0,end="")
```

Listing 30: Smali resultante do .java

```
1 .class public Lteste1_09;
2 .super Ljava/lang/Object;
3 .source "teste1_09.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
8      .registers 3
9
10     .prologue
11     .line 1
12     move-object v0, p0
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 5
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v2, 0x1
29
30     move v1, v2
31
32     .line 7
33     move v2, v1
34
35     const/4 v3, 0x1
36
37     if-ne v2, v3, :cond_e
38
39     .line 8
40     sget-object v2,
41         Ljava/lang/System;->out:Ljava/io/PrintStream;
42
43     move v3, v1
44
45     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(I)V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
45  
46 .line 13  
47 :goto_d  
48 return-void  
49  
50 .line 11  
51 :cond_e  
52 sget-object v2,  
    Ljava/lang/System;->out:Ljava/io/PrintStream;  
53  
54 const/4 v3, 0x0  
55  
56 invoke-virtual {v2, v3}, Ljava/io/PrintStream;->print(I)V  
57  
58 goto :goto_d  
59 .end method
```

O smali interpreta uma conta como $1 + 1/2$ como resultado inteiro caso seja atribuída a um registrador de 4 bytes(do tipo inteiro).

Saída

1

Obs.: Sem quebra de linha

5.2.10 Algoritmo nano10

Listing 31: Código em Java

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


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
10     }
11     else{
12         System.out.print(0);
13     }
14 }
15 }
```

Listing 32: Código em python

```
1 def nano10():
2     n=1
3     m=2
4     if n ==m:
5         print(n,end="")
6     else:
7         print(0,end="")
```

Listing 33: Smali resultante do .java

```
1 .class public Lnano10;
2 .super Ljava/lang/Object;
3 .source "nano10.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22   .registers 6
23
24   .prologue
25   .line 6
26   move-object v0, p0
27
28   const/4 v3, 0x1
29
30   move v1, v3
31
32   .line 7
33   const/4 v3, 0x2
34
35   move v2, v3
36
37   .line 8
38   move v3, v1
39
40   move v4, v2
41
42   if-ne v3, v4, :cond_10
43
44   .line 9
45   sget-object v3,
46     Ljava/lang/System;.->out:Ljava/io/PrintStream;
47
48   move v4, v1
49
50   invoke-virtual {v3, v4}, Ljava/io/PrintStream;.->print(I)V
51
52   .line 14
53   :goto_f
54   return-void
55
56   .line 12
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
56 :cond_10
57 sget-object v3,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
58
59 const/4 v4, 0x0
60
61 invoke-virtual {v3, v4}, Ljava/io/PrintStream;->print(I)V
62
63 goto :goto_f
64 .end method
```

Saída

0

Obs.: Sem quebra de linha

5.2.11 Algoritmo nano11

Listing 34: Código em Java

```
1 public class nano11
2 {
3     public static void main(String[] args)
4     {
5         int n,m,x;
6         n=1;
7         m=2;
8         x=5;
9         while (x >n)
10        {
11            n = n +m;
12            System.out.print(n);
13        }
14    }
15 }
```

Listing 35: Código em python

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
1 def nano11():
2     n=1
3     m=2
4     x=5
5     while x >n:
6         n = n + m
7         print(n,end="")
```

Listing 36: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 7
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
28      const/4 v4, 0x1
29
30      move v1, v4
31
32      .line 7
33      const/4 v4, 0x2
34
35      move v2, v4
36
37      .line 8
38      const/4 v4, 0x5
39
40      move v3, v4
41
42      .line 9
43      :goto_7
44      move v4, v3
45
46      move v5, v1
47
48      if-le v4, v5, :cond_16
49
50      .line 11
51      move v4, v1
52
53      move v5, v2
54
55      add-int/2addr v4, v5
56
57      move v1, v4
58
59      .line 12
60      sget-object v4,
        Ljava/lang/System; -> out:Ljava/io/PrintStream;
61
62      move v5, v1
63
64      invoke-virtual {v4, v5}, Ljava/io/PrintStream; -> print(I)V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
65
66     goto :goto_7
67
68     .line 14
69     :cond_16
70     return-void
71 .end method
```

O comando `if-le` na linha 48 diz que se `v4` for menor ou igual a `v5` então eu vou para o target `:cond_16`. Caso contrário, vou incrementando `v5` em parcelas de `v4` como pode ser observado na linha 55.

Saída

```
35
```

Obs.: Sem quebra de linha

5.2.12 Algoritmo nano12

Listing 37: Código em Java

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

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
18     }
19     x = x -1;
20 }
21 }
22 }
```

Listing 38: Código em python

```
1 def nano12():
2     n=1
3     m=2
4     x=5
5     while x >n:
6         if n ==m:
7             print(n,end="")
8         else:
9             print(0,end="")
10    x = x -1
```

Listing 39: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 1
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 7
23
24     .prologue
25     .line 6
26     move-object v0, p0
27
28     const/4 v4, 0x1
29
30     move v1, v4
31
32     .line 7
33     const/4 v4, 0x2
34
35     move v2, v4
36
37     .line 8
38     const/4 v4, 0x5
39
40     move v3, v4
41
42     .line 9
43     :goto_7
44     move v4, v3
45
46     move v5, v1
47
48     if-le v4, v5, :cond_22
49
50     .line 11
51     move v4, v1
52
53     move v5, v2
54
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
55     if-ne v4, v5, :cond_1b
56
57     .line 13
58     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
59
60     move v5, v1
61
62     invoke-virtual {v4, v5}, Ljava/io/PrintStream;->print(I)V
63
64     .line 19
65     :goto_15
66     move v4, v3
67
68     const/4 v5, 0x1
69
70     add-int/lit8 v4, v4, -0x1
71
72     move v3, v4
73
74     goto :goto_7
75
76     .line 17
77     :cond_1b
78     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
79
80     const/4 v5, 0x0
81
82     invoke-virtual {v4, v5}, Ljava/io/PrintStream;->print(I)V
83
84     goto :goto_15
85
86     .line 21
87     :cond_22
88     return-void
89 .end method
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Pode-se observar, na linha 70, um comando diferente. Esse comando, apesar de ser do tipo add, soma com -1 e, portanto subtrai o v4 de 1.

Saída

0000

Obs.: Sem quebra de linha

5.2.13 Algoritmo micro01

Listing 40: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro01
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         float cel , far ;
9         System.out.println("  Tabela de conversao: Celsius ->
10             Fahrenheit");
11         System.out.print("Digite a temperatura em Celsius: ");
12         cel = s.nextFloat();
13         far = (9*cel+160)/5;
14         System.out.println("A nova temperatura e:"+far+"F");
15     }
16 }
```

Listing 41: Código em python

```
1 def micro01():
2     cel , far = 0.0 , 0.0
3     print("    Tabela de conversao: Celsius -> Fahrenheit")
4     print("Digite a temperatura em Celsius: ",end="")
5     cel = int(input())
6     far = (9*cel+160)/5
7     print("A nova temperatura e: j"+str(far)+"F")
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 42: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 9
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v4, Ljava/util/Scanner;
29
30     move-object v7, v4
31
32     move-object v4, v7
33
34     move-object v5, v7
35
36     sget-object v6, Ljava/lang/System;:>in:Ljava/io/InputStream;
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
37
38     invoke-direct {v5, v6},
        Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
39
40     move-object v1, v4
41
42     .line 9
43     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
44
45     const-string v5, "\t\tTabela de conversao: Celsius ->
        Fahrenheit"
46
47     invoke-virtual {v4, v5},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
48
49     .line 10
50     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
51
52     const-string v5, "Digite a temperatura em Celsius: "
53
54     invoke-virtual {v4, v5},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
55
56     .line 11
57     move-object v4, v1
58
59     invoke-virtual {v4}, Ljava/util/Scanner;->nextFloat()F
60
61     move-result v4
62
63     move v2, v4
64
65     .line 12
66     const/high16 v4, 0x41100000 # 9.0f
67
68     move v5, v2
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
69      mul-float/2addr v4, v5
70
71
72      const/high16 v5, 0x43200000 # 160.0f
73
74      add-float/2addr v4, v5
75
76      const/high16 v5, 0x40a00000 # 5.0f
77
78      div-float/2addr v4, v5
79
80      move v3, v4
81
82      .line 13
83      sget-object v4,
84          Ljava/lang/System;->out:Ljava/io/PrintStream;
85
86      new-instance v5, Ljava/lang/StringBuffer;
87
88      move-object v7, v5
89
90      move-object v5, v7
91
92      move-object v6, v7
93
94      invoke-direct {v6}, Ljava/lang/StringBuffer;-><init>()V
95
96      const-string v6, "A nova temperatura \u00e9:"
97
98      invoke-virtual {v5, v6},
99          Ljava/lang/StringBuffer;->append(Ljava/lang/String;)
100      Ljava/lang/StringBuffer;
101
102      move-result-object v5
103
104      move v6, v3
105
106      invoke-virtual {v5, v6},
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
105         Ljava/lang/StringBuffer;->append(F)Ljava/lang/StringBuffer;
106     move-result-object v5
107
108     const-string v6, "F"
109
110     invoke-virtual {v5, v6},
111         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)
112     Ljava/lang/StringBuffer;
113
114     move-result-object v5
115
116     invoke-virtual {v5},
117         Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
118
119     move-result-object v5
120
121     invoke-virtual {v4, v5},
122         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
123
124     .line 14
125     return-void
126 .end method
```

Pode-se observar comandos novos como `const-string`. Ele guarda uma string em um registrador. Pode-se reparar também em `new-instance` que declara uma nova instância da classe `Scanner`.

Saída

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro01$ java micr
o01
Tabela de conversao: Celsius -> Fahrenheit
Digite a temperatura em Celsius: 24
A nova temperatura é:75.2F
```

5.2.14 Algoritmo micro02

Listing 43: Código em Java

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
1 import java.util.Scanner;
2
3 public class micro02
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int num1 , num2 ;
9         System.out.print("Digite o primeiro numero: ");
10        num1 = s.nextInt();
11        System.out.print("Digite o segundo numero: ");
12        num2 = s.nextInt();
13        if(num1 > num2)
14            System.out.print("O primeiro numero "+num1+" e maior
15                               que o segundo "+num2);
16        else
17            System.out.print("O segundo numero "+num2+" e maior
18                               que o primeiro "+num1);
19    }
20 }
```

Listing 44: Código em python

```
1 def micro02():
2     num1, num2 = 0 , 0
3     print("Digite o primeiro numero: ")
4     num1 = int(input())
5     print("Digite o segundo numero: ")
6     num2 = int(input())
7
8     if num1 > num2:
9         print("O primeiro numero "+str(num1)+" e maior que o
10                segundo "+str(num2),end="")
11     else:
12         print("O segundo numero "+str(num2)+" e maior que o
13                primeiro "+str(num1),end="")
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 45: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 9
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v4, Ljava/util/Scanner;
29
30     move-object v7, v4
31
32     move-object v4, v7
33
34     move-object v5, v7
35
36     sget-object v6, Ljava/lang/System;:>in:Ljava/io/InputStream;
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
37
38     invoke-direct {v5, v6},
        Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
39
40     move-object v1, v4
41
42     .line 9
43     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
44
45     const-string v5, "Digite o primeiro numero: "
46
47     invoke-virtual {v4, v5},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
48
49     .line 10
50     move-object v4, v1
51
52     invoke-virtual {v4}, Ljava/util/Scanner;->nextInt()I
53
54     move-result v4
55
56     move v2, v4
57
58     .line 11
59     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
60
61     const-string v5, "Digite o segundo numero: "
62
63     invoke-virtual {v4, v5},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
64
65     .line 12
66     move-object v4, v1
67
68     invoke-virtual {v4}, Ljava/util/Scanner;->nextInt()I
69
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
70     move-result v4
71
72     move v3, v4
73
74     .line 13
75     move v4, v2
76
77     move v5, v3
78
79     if-le v4, v5, :cond_52
80
81     .line 14
82     sget-object v4,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
83
84     new-instance v5, Ljava/lang/StringBuffer;
85
86     move-object v7, v5
87
88     move-object v5, v7
89
90     move-object v6, v7
91
92     invoke-direct {v6}, Ljava/lang/StringBuffer;-><init>()V
93
94     const-string v6, "0 primeiro numero "
95
96     invoke-virtual {v5, v6},
        Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
97
98     move-result-object v5
99
100    move v6, v2
101
102    invoke-virtual {v5, v6},
        Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
103
104    move-result-object v5
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
105
106     const-string v6, " e maior que o segundo "
107
108     invoke-virtual {v5, v6},
109         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
110
111     move-result-object v5
112
113     move v6, v3
114
115     invoke-virtual {v5, v6},
116         Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
117
118     move-result-object v5
119
120     invoke-virtual {v5},
121         Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
122
123     move-result-object v5
124
125     invoke-virtual {v4, v5},
126         Ljava/io/PrintStream;->print(Ljava/lang/String;)V
127
128     .line 18
129     :goto_51
130     return-void
131
132     .line 16
133     :cond_52
134     sget-object v4,
135         Ljava/lang/System;->out:Ljava/io/PrintStream;
136
137     new-instance v5, Ljava/lang/StringBuffer;
138
139     move-object v7, v5
140
141     move-object v5, v7
```

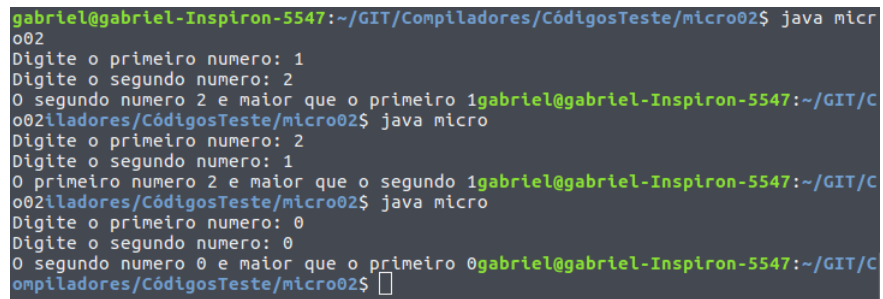
5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
138     move-object v6, v7
139
140     invoke-direct {v6}, Ljava/lang/StringBuffer;-><init>()V
141
142     const-string v6, "0 segundo numero "
143
144     invoke-virtual {v5, v6},
145         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
146
147     move-result-object v5
148
149     move v6, v3
150
151     invoke-virtual {v5, v6},
152         Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
153
154     move-result-object v5
155
156     const-string v6, " e maior que o primeiro "
157
158     invoke-virtual {v5, v6},
159         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
160
161     move-result-object v5
162
163     move v6, v2
164
165     invoke-virtual {v5, v6},
166         Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
167
168     move-result-object v5
169
170     invoke-virtual {v5, v5},
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
171         Ljava/io/PrintStream;->print(Ljava/lang/String;)V
172     goto :goto_51
173 .end method
```

Saída



```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro02$ java micr
o02
Digite o primeiro numero: 1
Digite o segundo numero: 2
O segundo numero 2 e maior que o primeiro 1gabriel@gabriel-Inspiron-5547:~/GIT/C
o02iladores/CódigosTeste/micro02$ java micro
Digite o primeiro numero: 2
Digite o segundo numero: 1
O primeiro numero 2 e maior que o segundo 1gabriel@gabriel-Inspiron-5547:~/GIT/C
o02iladores/CódigosTeste/micro02$ java micro
Digite o primeiro numero: 0
Digite o segundo numero: 0
O segundo numero 0 e maior que o primeiro 0gabriel@gabriel-Inspiron-5547:~/GIT/C
ompiladores/CódigosTeste/micro02$
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.2.15 Algoritmo micro03

Listing 46: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro03
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11
12        if(numero >= 100)
13        {
14            if(numero <=200)
15                System.out.println("O numero esta no intervalo
16                                   entre 100 e 200");
17            else
18                System.out.println("O numero nao esta no intervalo
19                                   entre 100 e 200");
20        }
21        else
22            System.out.println("O numero nao esta no intervalo
23                               entre 100 e 200");
24    }
25 }
```

Listing 47: Código em python

```
1 def micro03():
2     numero =0
3     print("Digite um numero: ",end="")
4     numero = int(input())
5     if numero>= 100:
6         if numero<= 200:
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
7         print("0 numero esta no intervalo entre 100 e 200")
8     else:
9         print("0 numero nao esta no intervalo entre 100 e 200")
10 else:
11     print("0 numero nao esta no intervalo entre 100 e 200")
```

Listing 48: Smali resultante do .java

```
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 3
9
10     .prologue
11     .line 3
12     move-object v0, p0
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 8
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v3, Ljava/util/Scanner;
29
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
30  move-object v6, v3
31
32  move-object v3, v6
33
34  move-object v4, v6
35
36  sget-object v5, Ljava/lang/System;->in:Ljava/io/InputStream;
37
38  invoke-direct {v4, v5},
    Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
39
40  move-object v1, v3
41
42  .line 9
43  sget-object v3,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
44
45  const-string v4, "Digite um numero: "
46
47  invoke-virtual {v3, v4},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
48
49  .line 10
50  move-object v3, v1
51
52  invoke-virtual {v3}, Ljava/util/Scanner;->nextInt()I
53
54  move-result v3
55
56  move v2, v3
57
58  .line 12
59  move v3, v2
60
61  const/16 v4, 0x64
62
63  if-lt v3, v4, :cond_33
64
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
65     .line 14
66     move v3, v2
67
68     const/16 v4, 0xc8
69
70     if-gt v3, v4, :cond_2b
71
72     .line 15
73     sget-object v3,
74         Ljava/lang/System; ->out:Ljava/io/PrintStream;
75
76     const-string v4, "0 numero esta no intervalo entre 100 e
77         200"
78
79     invoke-virtual {v3, v4},
80         Ljava/io/PrintStream; ->println(Ljava/lang/String;)V
81
82     .line 21
83     :goto_2a
84     return-void
85
86     .line 17
87     :cond_2b
88     sget-object v3,
89         Ljava/lang/System; ->out:Ljava/io/PrintStream;
90
91     const-string v4, "0 numero nao esta no intervalo entre 100
92         e 200"
93
94     invoke-virtual {v3, v4},
95         Ljava/io/PrintStream; ->println(Ljava/lang/String;)V
96
97     goto :goto_2a
98
99     .line 20
100    :cond_33
101    sget-object v3,
102        Ljava/lang/System; ->out:Ljava/io/PrintStream;
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
96
97     const-string v4, "0 numero nao esta no intervalo entre 100
98         e 200"
99
100     invoke-virtual {v3, v4},
101         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
102
103     goto :goto_2a
104 .end method
```

O comando `if-gt` (Greater than) compara dois registradores e se a condição for satisfeita, vai para o target, senão continua normalmente. O `if-lt` (Less than) funciona de forma análoga. O programa trata o primeiro if na linha 63 que é o caso do numero ser maior que 100. O if seguinte, na linha 70 verifica se `if-greater than v3 = 100 v4 =200`.

Os números atribuídos aos registradores sempre estão em formato hexadecimal.

Saída

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro03$ java micro03
Digite um numero: 1
0 numero nao esta no intervalo entre 100 e 200
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro03$ java micro03
Digite um numero: 123
0 numero esta no intervalo entre 100 e 200
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro03$ java micro03
Digite um numero: 201
0 numero nao esta no intervalo entre 100 e 200
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro03$
```

5.2.16 Algoritmo micro04

Listing 49: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro04
4 {
5     public static void main(String[] args)
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
6 {
7     Scanner s = new Scanner(System.in);
8     int x=0,num=0,intervalo =0;
9
10    for (x=0;x<5;x++){
11        System.out.print("Digite o numero: ");
12        num = s.nextInt();
13        if( num >=10)
14            if (num <=150)
15                intervalo = intervalo +1;
16    }
17
18    System.out.println("Ao total, foram digitados
19                        "+intervalo+" numeros no intervalo entre 10 e 150");
20 }
```

Listing 50: Código em python

```
1 def micro04():
2     x,num,intervalo = 0,0,0
3
4     for x in range(5):
5         print("Digite o numero: ",end="")
6         num = int(input())
7         if num >=10:
8             if num <=150:
9                 intervalo = intervalo +1
10
11    print("Ao total, foram digitados "+str(intervalo)+" numeros
        no intervalo entre 10 e 150")
```

Listing 51: Smali resultante do .java

```
1 .class public Lmicro04;
2 .super Ljava/lang/Object;
3 .source "micro04.java"
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 10
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v5, Ljava/util/Scanner;
29
30     move-object v8, v5
31
32     move-object v5, v8
33
34     move-object v6, v8
35
36     sget-object v7, Ljava/lang/System;-.in:Ljava/io/InputStream;
37
38     invoke-direct {v6, v7},
39         Ljava/util/Scanner;-.<init>(Ljava/io/InputStream;)V
40
41     move-object v1, v5
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
41
42     .line 8
43     const/4 v5, 0x0
44
45     move v2, v5
46
47     const/4 v5, 0x0
48
49     move v3, v5
50
51     const/4 v5, 0x0
52
53     move v4, v5
54
55     .line 10
56     const/4 v5, 0x0
57
58     move v2, v5
59
60     :goto_14
61     move v5, v2
62
63     const/4 v6, 0x5
64
65     if-ge v5, v6, :cond_37
66
67     .line 11
68     sget-object v5,
69         Ljava/lang/System; -> out:Ljava/io/PrintStream;
70
71     const-string v6, "Digite o numero: "
72
73     invoke-virtual {v5, v6},
74         Ljava/io/PrintStream; -> print(Ljava/lang/String;)V
75
76     .line 12
77     move-object v5, v1
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
77     invoke-virtual {v5}, Ljava/util/Scanner;->nextInt()I
78
79     move-result v5
80
81     move v3, v5
82
83     .line 13
84     move v5, v3
85
86     const/16 v6, 0xa
87
88     if-lt v5, v6, :cond_34
89
90     .line 14
91     move v5, v3
92
93     const/16 v6, 0x96
94
95     if-gt v5, v6, :cond_34
96
97     .line 15
98     move v5, v4
99
100    const/4 v6, 0x1
101
102    add-int/lit8 v5, v5, 0x1
103
104    move v4, v5
105
106    .line 10
107    :cond_34
108    add-int/lit8 v2, v2, 0x1
109
110    goto :goto_14
111
112    .line 18
113    :cond_37
114    sget-object v5,
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```

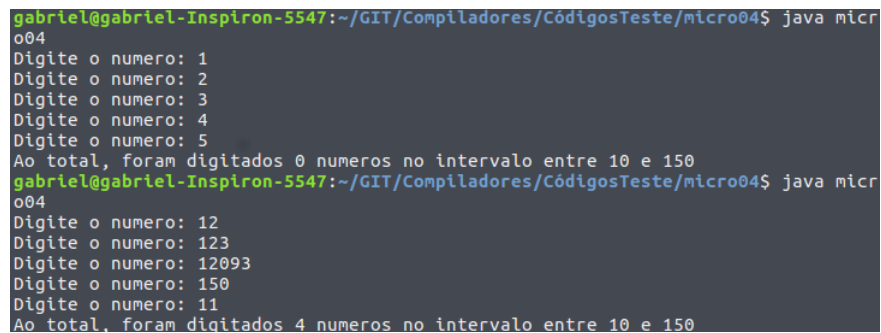
    Ljava/lang/System;->out:Ljava/io/PrintStream;
115
116 new-instance v6, Ljava/lang/StringBuffer;
117
118 move-object v8, v6
119
120 move-object v6, v8
121
122 move-object v7, v8
123
124 invoke-direct {v7}, Ljava/lang/StringBuffer;-><init>()V
125
126 const-string v7, "Ao total, foram digitados "
127
128 invoke-virtual {v6, v7},
    Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
129
130 move-result-object v6
131
132 move v7, v4
133
134 invoke-virtual {v6, v7},
    Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
135
136 move-result-object v6
137
138 const-string v7, " numeros no intervalo entre 10 e 150"
139
140 invoke-virtual {v6, v7},
    Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
141
142 move-result-object v6
143
144 invoke-virtual {v6},
    Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
145
146 move-result-object v6
147
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
148     invoke-virtual {v5, v6},  
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V  
149  
150     .line 19  
151     return-void  
152 .end method
```

Esse .smali é parecido com o anterior. A diferença principal é o uso do for que, nesse código em .smali é representado por if juntamente com o target.

Saída



```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro04$ java micr  
o04  
Digite o numero: 1  
Digite o numero: 2  
Digite o numero: 3  
Digite o numero: 4  
Digite o numero: 5  
Ao total, foram digitados 0 numeros no intervalo entre 10 e 150  
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro04$ java micr  
o04  
Digite o numero: 12  
Digite o numero: 123  
Digite o numero: 12093  
Digite o numero: 150  
Digite o numero: 11  
Ao total, foram digitados 4 numeros no intervalo entre 10 e 150
```

5.2.17 Algoritmo micro05

Algoritmos com switch são incompatíveis com o java -source 1.4. Por isso, foi usado o 1.7. Além disso, o Python não possui switch-case, então foi feita uma versão equivalente usando if e else.

Listing 52: Código em Java

```
1 import java.util.Scanner;  
2  
3 public class micro05  
4 {  
5     public static void main(String[] args)  
6     {  
7         Scanner s = new Scanner(System.in);  
8         int x=0,h=0,m =0;  
9         String nome, sexo;  
10    }
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
11     for (x=0;x<5;x++){
12         System.out.print("Digite o nome: ");
13         nome = s.nextLine();
14         System.out.print("H - Homem ou M - Mulher");
15         sexo = s.nextLine();
16
17         switch(sexo){
18             case "H":
19                 h = h +1;
20                 break;
21             case "M":
22                 m = m +1;
23                 break;
24             default:
25                 System.out.println("Sexo so pode ser H ou M!");
26         }
27     }
28
29     System.out.println("Foram inseridos "+h+" Homens");
30     System.out.println("Foram inseridas "+m+" Mulheres");
31 }
32 }
```

Listing 53: Código em python

```
1 def micro05():
2     x,h,m = 0,0,0
3     nome,sexo = "", ""
4
5     for x in range(5):
6         print("Digite o nome: ",end="")
7         nome = input()
8         print("H - Homem ou M - Mulher",end="")
9         sexo = input()
10        if sexo == "H":
11            h = h+1
12        elif sexo == "M":
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
13         m = m+1
14     else:
15         print("Sexo so pode ser H ou M!")
16
17     print("Foram inseridos "+h+" Homens")
18     print("Foram inseridas "+m+" Mulheres")
```

Listing 54: Smali resultante do .java

```
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 3
9
10     .prologue
11     .line 3
12     move-object v0, p0
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 14
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v9, Ljava/util/Scanner;
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
29
30     move-object v12, v9
31
32     move-object v9, v12
33
34     move-object v10, v12
35
36     sget-object v11,
37         Ljava/lang/System; -> in:Ljava/io/InputStream;
38
39     invoke-direct {v10, v11},
40         Ljava/util/Scanner; -> <init>(Ljava/io/InputStream;)V
41
42     move-object v1, v9
43
44     .line 8
45     const/4 v9, 0x0
46
47     move v2, v9
48
49     const/4 v9, 0x0
50
51     move v3, v9
52
53     const/4 v9, 0x0
54
55     move v4, v9
56
57     .line 11
58     const/4 v9, 0x0
59
60     move v2, v9
61
62     :goto_14
63     move v9, v2
64
65     const/4 v10, 0x5
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
65     if-ge v9, v10, :cond_70
66
67     .line 12
68     sget-object v9,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
69
70     const-string v10, "Digite o nome: "
71
72     invoke-virtual {v9, v10},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
73
74     .line 13
75     move-object v9, v1
76
77     invoke-virtual {v9},
        Ljava/util/Scanner;->nextLine()Ljava/lang/String;
78
79     move-result-object v9
80
81     move-object v5, v9
82
83     .line 14
84     sget-object v9,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
85
86     const-string v10, "H - Homem ou M - Mulher"
87
88     invoke-virtual {v9, v10},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
89
90     .line 15
91     move-object v9, v1
92
93     invoke-virtual {v9},
        Ljava/util/Scanner;->nextLine()Ljava/lang/String;
94
95     move-result-object v9
96
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
97     move-object v6, v9
98
99     .line 17
100    move-object v9, v6
101
102    move-object v7, v9
103
104    const/4 v9, -0x1
105
106    move v8, v9
107
108    move-object v9, v7
109
110    invoke-virtual {v9}, Ljava/lang/String;->hashCode()I
111
112    move-result v9
113
114    sparse-switch v9, :sswitch_data_b6
115
116    :cond_3e
117    :goto_3e
118    move v9, v8
119
120    packed-switch v9, :pswitch_data_c0
121
122    .line 25
123    sget-object v9,
124        Ljava/lang/System;->out:Ljava/io/PrintStream;
125
126    const-string v10, "Sexo so pode ser H ou M!"
127
128    invoke-virtual {v9, v10},
129        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
130
131    .line 11
132    :goto_49
133    add-int/lit8 v2, v2, 0x1
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
133      goto :goto_14
134
135      .line 17
136      :sswitch_4c
137      move-object v9, v7
138
139      const-string v10, "H"
140
141      invoke-virtual {v9, v10},
142          Ljava/lang/String;->equals(Ljava/lang/Object;)Z
143
144      move-result v9
145
146      if-eqz v9, :cond_3e
147
148      const/4 v9, 0x0
149
150      move v8, v9
151
152      goto :goto_3e
153
154      :sswitch_58
155      move-object v9, v7
156
157      const-string v10, "M"
158
159      invoke-virtual {v9, v10},
160          Ljava/lang/String;->equals(Ljava/lang/Object;)Z
161
162      move-result v9
163
164      if-eqz v9, :cond_3e
165
166      const/4 v9, 0x1
167
168      move v8, v9
169
170      goto :goto_3e
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
169
170     .line 19
171     :pswitch_64
172     move v9, v3
173
174     const/4 v10, 0x1
175
176     add-int/lit8 v9, v9, 0x1
177
178     move v3, v9
179
180     .line 20
181     goto :goto_49
182
183     .line 22
184     :pswitch_6a
185     move v9, v4
186
187     const/4 v10, 0x1
188
189     add-int/lit8 v9, v9, 0x1
190
191     move v4, v9
192
193     .line 23
194     goto :goto_49
195
196     .line 29
197     :cond_70
198     sget-object v9,
199         Ljava/lang/System; -> out:Ljava/io/PrintStream;
200
201     new-instance v10, Ljava/lang/StringBuilder;
202
203     move-object v12, v10
204
205     move-object v10, v12
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
206     move-object v11, v12
207
208     invoke-direct {v11}, Ljava/lang/StringBuilder;-><init>()V
209
210     const-string v11, "Foram inseridos "
211
212     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuild
213
214     move-result-object v10
215
216     move v11, v3
217
218     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;
219
220     move-result-object v10
221
222     const-string v11, " Homens"
223
224     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuild
225
226     move-result-object v10
227
228     invoke-virtual {v10},
        Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
229
230     move-result-object v10
231
232     invoke-virtual {v9, v10},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
233
234     .line 30
235     sget-object v9,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
236
237     new-instance v10, Ljava/lang/StringBuilder;
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
238
239     move-object v12, v10
240
241     move-object v10, v12
242
243     move-object v11, v12
244
245     invoke-direct {v11}, Ljava/lang/StringBuilder;-><init>()V
246
247     const-string v11, "Foram inseridas "
248
249     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuild
250
251     move-result-object v10
252
253     move v11, v4
254
255     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;
256
257     move-result-object v10
258
259     const-string v11, " Mulheres"
260
261     invoke-virtual {v10, v11},
        Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuild
262
263     move-result-object v10
264
265     invoke-virtual {v10},
        Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
266
267     move-result-object v10
268
269     invoke-virtual {v9, v10},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
270
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
271 .line 31
272 return-void
273
274 .line 17
275 nop
276
277 :sswitch_data_b6
278 .sparse-switch
279 0x48 -> :sswitch_4c
280 0x4d -> :sswitch_58
281 .end sparse-switch
282
283 :pswitch_data_c0
284 .packed-switch 0x0
285 :pswitch_64
286 :pswitch_6a
287 .end packed-switch
288 .end method
```

O uso do switch é representado por `.sparse-switch` `.packed-switch`

- **packed-switch:** o packed switch funciona de forma equivalente ao sparse.
- **sparse-switch:** é montado uma tabela de constantes. Se na tabela não for achado o caso esperado, o código segue para o default.

Saída

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro05$ java micr
o05
Digite o nome: Jose
H - Homem ou M - MulherH
Digite o nome: Maria
H - Homem ou M - MulherM
Digite o nome: gabriel
H - Homem ou M - MulherH
Digite o nome: Josefina
H - Homem ou M - MulherM
Digite o nome: X
H - Homem ou M - MulherX
Sexo so pode ser H ou M!
Foram inseridos 2 Homens
Foram inseridas 2 Mulheres
```

5.2.18 Algoritmo micro06

Listing 55: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro06
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0;
9         System.out.print("Digite um numero de 1 a 5: ");
10        numero = s.nextInt();
11        switch(numero)
12        {
13            case 1:
14                System.out.println("Um");
15                break;
16            case 2:
17                System.out.println("Dois");
18                break;
19            case 3:
20                System.out.println("Tres");
21                break;
22            case 4:
23                System.out.println("Quatro");
24                break;
25            case 5:
26                System.out.println("Cinco");
27                break;
28            default:
29                System.out.println("Numero Invalido");
30        }
31    }
32 }
33 }
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 56: Código em python

```
1 def micro06():
2     numero = 0
3
4     print("Digite um numero de 1 a 5: ",end="")
5     numero = int(input())
6     if numero ==1:
7         print("Um")
8     elif numero == 2:
9         print("Dois")
10    elif numero ==3:
11        print("Tres")
12    elif numero ==4:
13        print("Quatro")
14    elif numero ==5:
15        print("Cinco")
16    else:
17        print("Numero Invalido!!!")
```

Listing 57: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 8
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v3, Ljava/util/Scanner;
29
30     move-object v6, v3
31
32     move-object v3, v6
33
34     move-object v4, v6
35
36     sget-object v5, Ljava/lang/System;->in:Ljava/io/InputStream;
37
38     invoke-direct {v4, v5},
39         Ljava/util/Scanner;-<init>(Ljava/io/InputStream;)V
40
41     move-object v1, v3
42
43     .line 8
44     const/4 v3, 0x0
45
46     move v2, v3
47
48     .line 9
49     sget-object v3,
50         Ljava/lang/System;->out:Ljava/io/PrintStream;
51
52     const-string v4, "Digite um numero de 1 a 5: "
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
53
54    .line 10
55    move-object v3, v1
56
57    invoke-virtual {v3}, Ljava/util/Scanner;->nextInt()I
58
59    move-result v3
60
61    move v2, v3
62
63    .line 11
64    move v3, v2
65
66    packed-switch v3, :pswitch_data_50
67
68    .line 29
69    sget-object v3,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
70
71    const-string v4, "Numero Invalido"
72
73    invoke-virtual {v3, v4},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
74
75    .line 32
76    :goto_26
77    return-void
78
79    .line 14
80    :pswitch_27
81    sget-object v3,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
82
83    const-string v4, "Um"
84
85    invoke-virtual {v3, v4},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
86
87     .line 15
88     goto :goto_26
89
90     .line 17
91     :pswitch_2f
92     sget-object v3,
93         Ljava/lang/System; ->out:Ljava/io/PrintStream;
94
95     const-string v4, "Dois"
96
97     invoke-virtual {v3, v4},
98         Ljava/io/PrintStream; ->println(Ljava/lang/String;)V
99
100     .line 18
101     goto :goto_26
102
103     .line 20
104     :pswitch_37
105     sget-object v3,
106         Ljava/lang/System; ->out:Ljava/io/PrintStream;
107
108     const-string v4, "Tres"
109
110     invoke-virtual {v3, v4},
111         Ljava/io/PrintStream; ->println(Ljava/lang/String;)V
112
113     .line 21
114     goto :goto_26
115
116     .line 23
117     :pswitch_3f
118     sget-object v3,
119         Ljava/lang/System; ->out:Ljava/io/PrintStream;
120
121     const-string v4, "Quatro"
122
123     invoke-virtual {v3, v4},
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
119         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
120     .line 24
121     goto :goto_26
122
123     .line 26
124     :pswitch_47
125     sget-object v3,
126         Ljava/lang/System;->out:Ljava/io/PrintStream;
127
128     const-string v4, "Cinco"
129
130     invoke-virtual {v3, v4},
131         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
132
133     .line 27
134     goto :goto_26
135
136     .line 11
137     nop
138
139     :pswitch_data_50
140     .packed-switch 0x1
141     :pswitch_27
142     :pswitch_2f
143     :pswitch_37
144     :pswitch_3f
145     :pswitch_47
146     .end packed-switch
147 .end method
```

Saída

5.2.19 Algoritmo micro07

Listing 58: Código em Java

```
1 import java.util.Scanner;
2
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro06$ java micr
o06
Digite um numero de 1 a 5: 1
Um
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro06$ java micr
o06
Digite um numero de 1 a 5: 5
Cinco
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro06$ java micr
o06
Digite um numero de 1 a 5: 55
Numero Invalido
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro06$
```

```
3 public class micro07
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, programa=1;
9         char opc;
10        while( programa ==1){
11            System.out.print("Digite um numero: ");
12            numero = s.nextInt();
13
14            if (numero>0)
15                System.out.println("Positivo");
16            else
17            {
18                if (numero==0)
19                    System.out.println("0 numero e igual a 0");
20                if (numero <0)
21                    System.out.println("Negativo");
22            }
23            System.out.print("Deseja Finalizar? (S/N) ");
24            opc = s.next().charAt(0);
25            if (opc == 'S')
26                programa = 0;
27        }
28    }
29 }
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 59: Código em python

```
1 def micro07():
2     numero ,programa= 0,1
3     opc = ""
4
5     while programa ==1:
6         print("Digite um numero: ",end="")
7         numero = int(input())
8
9         if numero>0:
10            print("Positivo")
11        else:
12            if numero==0:
13                print("0 numero e igual a 0")
14            if numero <0:
15                print("Negativo")
16
17        print("Deseja Finalizar? (S/N) ",end="")
18        opc = input()
19        if opc == "S":
20            programa = 0
```

Listing 60: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 10
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v5, Ljava/util/Scanner;
29
30     move-object v8, v5
31
32     move-object v5, v8
33
34     move-object v6, v8
35
36     sget-object v7, Ljava/lang/System;->in:Ljava/io/InputStream;
37
38     invoke-direct {v6, v7},
39         Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
40
41     move-object v1, v5
42
43     .line 8
44     const/4 v5, 0x0
45
46     move v2, v5
47
48     const/4 v5, 0x1
49
50     move v3, v5
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
51  .line 10
52  :cond_10
53  :goto_10
54  move v5, v3
55
56  const/4 v6, 0x1
57
58  if-ne v5, v6, :cond_5a
59
60  .line 11
61  sget-object v5,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
62
63  const-string v6, "Digite um n\u00f3famer0: "
64
65  invoke-virtual {v5, v6},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
66
67  .line 12
68  move-object v5, v1
69
70  invoke-virtual {v5}, Ljava/util/Scanner;->nextInt()I
71
72  move-result v5
73
74  move v2, v5
75
76  .line 14
77  move v5, v2
78
79  if-lez v5, :cond_45
80
81  .line 15
82  sget-object v5,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
83
84  const-string v6, "Positivo"
85
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
86     invoke-virtual {v5, v6},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
87
88     .line 23
89     :cond_2b
90     :goto_2b
91     sget-object v5,
        Ljava/lang/System;->out:Ljava/io/PrintStream;
92
93     const-string v6, "Deseja Finalizar? (S/N) "
94
95     invoke-virtual {v5, v6},
        Ljava/io/PrintStream;->print(Ljava/lang/String;)V
96
97     .line 24
98     move-object v5, v1
99
100    invoke-virtual {v5},
        Ljava/util/Scanner;->next()Ljava/lang/String;
101
102    move-result-object v5
103
104    const/4 v6, 0x0
105
106    invoke-virtual {v5, v6}, Ljava/lang/String;->charAt(I)C
107
108    move-result v5
109
110    move v4, v5
111
112    .line 25
113    move v5, v4
114
115    const/16 v6, 0x53
116
117    if-ne v5, v6, :cond_10
118
119    .line 26
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
120     const/4 v5, 0x0
121
122     move v3, v5
123
124     goto :goto_10
125
126     .line 18
127     :cond_45
128     move v5, v2
129
130     if-nez v5, :cond_4f
131
132     .line 19
133     sget-object v5,
134         Ljava/lang/System;->out:Ljava/io/PrintStream;
135
136     const-string v6, "0 numero e igual a 0"
137
138     invoke-virtual {v5, v6},
139         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
140
141     .line 20
142     :cond_4f
143     move v5, v2
144
145     if-gez v5, :cond_2b
146
147     .line 21
148     sget-object v5,
149         Ljava/lang/System;->out:Ljava/io/PrintStream;
150
151     const-string v6, "Negativo"
152
153     invoke-virtual {v5, v6},
154         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
155
156     goto :goto_2b
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
154 .line 28
155 :cond_5a
156 return-void
157 .end method
```

Pode-se observar que a linha 58 corresponde ao while uma vez que se não for satisfeita a condição o programa vai para o final.

Saída

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro07$ java micr
o07
Digite um número: 1
Positivo
Deseja Finalizar? (S/N) N
Digite um número: -1
Negativo
Deseja Finalizar? (S/N) N
Digite um número: 0
O numero e igual a 0
Deseja Finalizar? (S/N) S
```

5.2.20 Algoritmo micro08

Listing 61: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro08
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero =1;
9         while (numero < 0 || numero >0){
10             System.out.print("Digite o numero");
11             numero = s.nextInt();
12             if (numero > 10)
13                 System.out.println("O numero "+numero+" e maior
14                                     que 10");
15             else
16                 System.out.println("O numero "+numero+" e menor
17                                     que 10");
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
16     }
17   }
18 }
```

Listing 62: Código em python

```
1 def micro08():
2     numero =1
3     while numero < 0 or numero >0:
4         print("Digite o numero",end="")
5         numero = int(input())
6         if numero > 10:
7             print("O numero "+str(numero)+" e maior que 10")
8         else:
9             print("O numero "+str(numero)+" e menor que 10")
```

Listing 63: Smali resultante do .java

```
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 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
20
21 .method public static main([Ljava/lang/String;)V
22   .registers 8
23
24   .prologue
25   .line 7
26   move-object v0, p0
27
28   new-instance v3, Ljava/util/Scanner;
29
30   move-object v6, v3
31
32   move-object v3, v6
33
34   move-object v4, v6
35
36   sget-object v5, Ljava/lang/System;-.>in:Ljava/io/InputStream;
37
38   invoke-direct {v4, v5},
39     Ljava/util/Scanner;-.<init>(Ljava/io/InputStream;)V
40
41   move-object v1, v3
42
43   .line 8
44   const/4 v3, 0x1
45
46   move v2, v3
47
48   .line 9
49   :goto_e
50   move v3, v2
51
52   if-ltz v3, :cond_14
53
54   move v3, v2
55
56   if-lez v3, :cond_6c
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
57     .line 10
58     :cond_14
59     sget-object v3,
        Ljava/lang/System; ->out:Ljava/io/PrintStream;
60
61     const-string v4, "Digite o numero"
62
63     invoke-virtual {v3, v4},
        Ljava/io/PrintStream; ->print(Ljava/lang/String;)V
64
65     .line 11
66     move-object v3, v1
67
68     invoke-virtual {v3}, Ljava/util/Scanner; ->nextInt()I
69
70     move-result v3
71
72     move v2, v3
73
74     .line 12
75     move v3, v2
76
77     const/16 v4, 0xa
78
79     if-le v3, v4, :cond_49
80
81     .line 13
82     sget-object v3,
        Ljava/lang/System; ->out:Ljava/io/PrintStream;
83
84     new-instance v4, Ljava/lang/StringBuffer;
85
86     move-object v6, v4
87
88     move-object v4, v6
89
90     move-object v5, v6
91
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
92     invoke-direct {v5}, Ljava/lang/StringBuffer;-><init>()V
93
94     const-string v5, "0 numero "
95
96     invoke-virtual {v4, v5},
97         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
98
99     move-result-object v4
100
101     move v5, v2
102
103     invoke-virtual {v4, v5},
104         Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
105
106     move-result-object v4
107
108     const-string v5, " e maior que 10"
109
110     invoke-virtual {v4, v5},
111         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
112
113     move-result-object v4
114
115     invoke-virtual {v4},
116         Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
117
118     move-result-object v4
119
120     invoke-virtual {v3, v4},
121         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
122
123     goto :goto_e
124
125     .line 15
126     :cond_49
127     sget-object v3,
128         Ljava/lang/System;->out:Ljava/io/PrintStream;
```

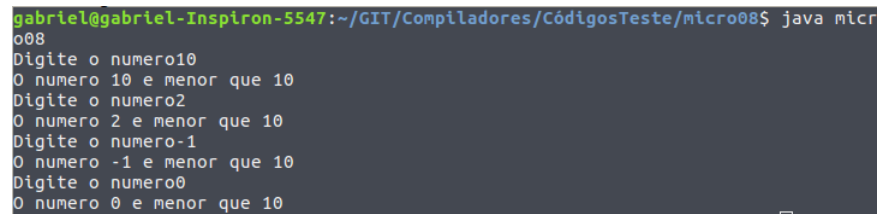
5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
124    new-instance v4, Ljava/lang/StringBuffer;
125
126    move-object v6, v4
127
128    move-object v4, v6
129
130    move-object v5, v6
131
132    invoke-direct {v5}, Ljava/lang/StringBuffer;-><init>()V
133
134    const-string v5, "0 numero "
135
136    invoke-virtual {v4, v5},
        Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
137
138    move-result-object v4
139
140    move v5, v2
141
142    invoke-virtual {v4, v5},
        Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
143
144    move-result-object v4
145
146    const-string v5, " e menor que 10"
147
148    invoke-virtual {v4, v5},
        Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
149
150    move-result-object v4
151
152    invoke-virtual {v4},
        Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
153
154    move-result-object v4
155
156    invoke-virtual {v3, v4},
        Ljava/io/PrintStream;->println(Ljava/lang/String;)V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
157
158     goto :goto_e
159
160     .line 17
161     :cond_6c
162     return-void
163 .end method
```

Saída



```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro08$ java micr
o08
Digite o numero10
0 numero 10 e menor que 10
Digite o numero2
0 numero 2 e menor que 10
Digite o numero-1
0 numero -1 e menor que 10
Digite o numero0
0 numero 0 e menor que 10
```

5.2.21 Algoritmo micro09

Listing 64: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro09
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         double preco, venda, novopreco=0 ;
9
10        System.out.print("Digite o preco: ");
11        preco = s.nextDouble();
12        System.out.print("Digite a venda: ");
13        venda = s.nextDouble();
14
15        if (venda < 500.0 || preco <30.0){
16            novopreco = preco + 10.0/100.0 *preco;
17        }
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
18     else if ((venda >= 500.0 && venda <1200.0) || (preco >=
19         30.0 && preco <80.0)){
20         novopreco = preco + 15.0/100.0 * preco;
21     }
22     else if (venda >=1200.0 || preco >=80.0){
23         novopreco = preco - 20.0/100.0 * preco;
24     }
25
26     System.out.println("O novo preco e: "+novopreco);
27 }
28 }
```

Listing 65: Código em python

```
1 def micro09():
2     preco, venda, novopreco = 0.0,0.0,0.0
3
4     print("Digite o preco: ",end="")
5     preco = int(input())
6     print("Digite a venda: ",end="")
7     venda = int(input())
8     if venda < 500 or preco <30:
9         novopreco = preco + 10/100 *preco
10    elif (venda >= 500 and venda <1200) or (preco >= 30 and
11        preco <80):
12        novopreco = preco + 15/100 * preco
13    elif venda >=1200 or preco >=80:
14        novopreco = preco - 20/100 * preco
15
16    print("O novo preco e: "+str(novopreco))
```

Listing 66: Smali resultante do .java

```
1 .class public Lmicro09;
2 .super Ljava/lang/Object;
3 .source "micro09.java"
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 3
9
10    .prologue
11    .line 3
12    move-object v0, p0
13
14    move-object v1, v0
15
16    invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18    return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22     .registers 16
23
24     .prologue
25     .line 7
26     move-object v0, p0
27
28     new-instance v8, Ljava/util/Scanner;
29
30     move-object v14, v8
31
32     move-object v8, v14
33
34     move-object v9, v14
35
36     sget-object v10,
37         Ljava/lang/System;->in:Ljava/io/InputStream;
38
39     invoke-direct {v9, v10},
40         Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
40  move-object v1, v8
41
42  .line 8
43  const-wide/16 v8, 0x0
44
45  move-wide v6, v8
46
47  .line 10
48  sget-object v8,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
49
50  const-string v9, "Digite o preco: "
51
52  invoke-virtual {v8, v9},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
53
54  .line 11
55  move-object v8, v1
56
57  invoke-virtual {v8}, Ljava/util/Scanner;->nextDouble()D
58
59  move-result-wide v8
60
61  move-wide v2, v8
62
63  .line 12
64  sget-object v8,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66  const-string v9, "Digite a venda: "
67
68  invoke-virtual {v8, v9},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
69
70  .line 13
71  move-object v8, v1
72
73  invoke-virtual {v8}, Ljava/util/Scanner;->nextDouble()D
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
74
75     move-result-wide v8
76
77     move-wide v4, v8
78
79     .line 15
80     move-wide v8, v4
81
82     const-wide v10, 0x407f400000000000L # 500.0
83
84     cmpg-double v8, v8, v10
85
86     if-ltz v8, :cond_3a
87
88     move-wide v8, v2
89
90     const-wide/high16 v10, 0x403e000000000000L # 30.0
91
92     cmpg-double v8, v8, v10
93
94     if-gez v8, :cond_61
95
96     .line 16
97     :cond_3a
98     move-wide v8, v2
99
100    const-wide v10, 0x3fb999999999999aL # 0.1
101
102    move-wide v12, v2
103
104    mul-double/2addr v10, v12
105
106    add-double/2addr v8, v10
107
108    move-wide v6, v8
109
110    .line 26
111    :cond_44
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
112      :goto_44
113      sget-object v8,
114          Ljava/lang/System; ->out:Ljava/io/PrintStream;
115
116      new-instance v9, Ljava/lang/StringBuffer;
117
118      move-object v14, v9
119
120      move-object v9, v14
121
122      move-object v10, v14
123
124      invoke-direct {v10}, Ljava/lang/StringBuffer; -><init>()V
125
126      const-string v10, "0 novo preco e: "
127
128      invoke-virtual {v9, v10},
129          Ljava/lang/StringBuffer; ->append(Ljava/lang/String;)Ljava/lang/StringBuffer
130
131      move-result-object v9
132
133      move-wide v10, v6
134
135      invoke-virtual {v9, v10, v11},
136          Ljava/lang/StringBuffer; ->append(D)Ljava/lang/StringBuffer;
137
138      move-result-object v9
139
140      invoke-virtual {v9},
141          Ljava/lang/StringBuffer; ->toString()Ljava/lang/String;
142
143      move-result-object v9
144
145      invoke-virtual {v8, v9},
146          Ljava/io/PrintStream; ->println(Ljava/lang/String;)V
147
148      .line 27
149      return-void
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
145
146     .line 18
147     :cond_61
148     move-wide v8, v4
149
150     const-wide v10, 0x407f400000000000L # 500.0
151
152     cmpl-double v8, v8, v10
153
154     if-ltz v8, :cond_75
155
156     move-wide v8, v4
157
158     const-wide v10, 0x4092c00000000000L # 1200.0
159
160     cmpg-double v8, v8, v10
161
162     if-ltz v8, :cond_83
163
164     :cond_75
165     move-wide v8, v2
166
167     const-wide/high16 v10, 0x403e000000000000L # 30.0
168
169     cmpl-double v8, v8, v10
170
171     if-ltz v8, :cond_8e
172
173     move-wide v8, v2
174
175     const-wide/high16 v10, 0x4054000000000000L # 80.0
176
177     cmpg-double v8, v8, v10
178
179     if-gez v8, :cond_8e
180
181     .line 19
182     :cond_83
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
183     move-wide v8, v2
184
185     const-wide v10, 0x3fc333333333333L # 0.15
186
187     move-wide v12, v2
188
189     mul-double/2addr v10, v12
190
191     add-double/2addr v8, v10
192
193     move-wide v6, v8
194
195     goto :goto_44
196
197     .line 21
198     :cond_8e
199     move-wide v8, v4
200
201     const-wide v10, 0x4092c00000000000L # 1200.0
202
203     cmpl-double v8, v8, v10
204
205     if-gez v8, :cond_9f
206
207     move-wide v8, v2
208
209     const-wide/high16 v10, 0x4054000000000000L # 80.0
210
211     cmpl-double v8, v8, v10
212
213     if-ltz v8, :cond_44
214
215     .line 22
216     :cond_9f
217     move-wide v8, v2
218
219     const-wide v10, 0x3fc999999999999aL # 0.2
220
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
221    move-wide v12, v2
222
223    mul-double/2addr v10, v12
224
225    sub-double/2addr v8, v10
226
227    move-wide v6, v8
228
229    goto :goto_44
230 .end method
```

Os `ifs` que contém duas condições ou mais são desmembrados em dois ou mais `ifs`. No caso do `if` que possui a condicional e são usados dois `if` em `smali` para fazer um `if` do `java` pois o primeiro verifica o lado esquerdo e o segundo o direito. Os dois `ifs` tem que ser verdadeiros. No caso do `ou`, apenas um lado da condição é verificado de cada vez e o segundo lado só é verificado se o primeiro falhar.

É interessante observar que quando o número em hexadecimal é muito extenso, o `smali` comenta automaticamente qual é esse número em decimal (linha 82 e 90)

Saída



```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro09$ java micr
o09
Digite o preço: 80
Digite a venda: 1200
O novo preço é: 64.0
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro09$ java micr
o09
Digite o preço: 30
Digite a venda: 500
O novo preço é: 34.5
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro09$
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

5.2.22 Algoritmo micro10

Listing 67: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro10
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, fat;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11        fat = fatorial(numero);
12        System.out.println("O fatorial de "+numero+" e "+fat);
13
14
15    }
16
17    public static int fatorial(int n)
18    {
19        if(n <= 0) return 1;
20        else return n* fatorial(n-1);
21    }
22
23
24 }
```

Listing 68: Código em python

```
1 def micro10():
2     numero =0
3     fat = 0
4     print("Digite um numero: ",end="")
5     numero = int(input())
6     fat = fatorial(numero)
7
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
8     print("O fatorial de "+str(numero)+" e "+str(fat),end="")
9
10
11
12 def fatorial(n):
13     if n <=0:
14         return 1
15     else:
16         return (n * fatorial(n-1))
```

Listing 69: Smali resultante do .java

```
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 3
9
10     .prologue
11     .line 3
12     move-object v0, p0
13
14     move-object v1, v0
15
16     invoke-direct {v1}, Ljava/lang/Object;-><init>()V
17
18     return-void
19 .end method
20
21 .method public static fatorial(I)I
22     .registers 5
23
24     .prologue
25     .line 18
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
26     move v0, p0
27
28     move v1, v0
29
30     if-gtz v1, :cond_7
31
32     const/4 v1, 0x1
33
34     move v0, v1
35
36     .line 19
37     :goto_6
38     return v0
39
40     :cond_7
41     move v1, v0
42
43     move v2, v0
44
45     const/4 v3, 0x1
46
47     add-int/lit8 v2, v2, -0x1
48
49     invoke-static {v2}, Lmicro10;->fatorial(I)I
50
51     move-result v2
52
53     mul-int/2addr v1, v2
54
55     move v0, v1
56
57     goto :goto_6
58     .end method
59
60     .method public static main([Ljava/lang/String;)V
61     .registers 9
62
63     .prologue
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
64 .line 7
65 move-object v0, p0
66
67 new-instance v4, Ljava/util/Scanner;
68
69 move-object v7, v4
70
71 move-object v4, v7
72
73 move-object v5, v7
74
75 sget-object v6, Ljava/lang/System;->in:Ljava/io/InputStream;
76
77 invoke-direct {v5, v6},
    Ljava/util/Scanner;-><init>(Ljava/io/InputStream;)V
78
79 move-object v1, v4
80
81 .line 8
82 const/4 v4, 0x0
83
84 move v2, v4
85
86 .line 9
87 sget-object v4,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
88
89 const-string v5, "Digite um numero: "
90
91 invoke-virtual {v4, v5},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
92
93 .line 10
94 move-object v4, v1
95
96 invoke-virtual {v4}, Ljava/util/Scanner;->nextInt()I
97
98 move-result v4
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
99
100  move v2, v4
101
102  .line 11
103  move v4, v2
104
105  invoke-static {v4}, Lmicro10;->fatorial(I)I
106
107  move-result v4
108
109  move v3, v4
110
111  .line 12
112  sget-object v4,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
113
114  new-instance v5, Ljava/lang/StringBuffer;
115
116  move-object v7, v5
117
118  move-object v5, v7
119
120  move-object v6, v7
121
122  invoke-direct {v6}, Ljava/lang/StringBuffer;-<init>()V
123
124  const-string v6, "0 fatorial de "
125
126  invoke-virtual {v5, v6},
    Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
127
128  move-result-object v5
129
130  move v6, v2
131
132  invoke-virtual {v5, v6},
    Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
133
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
134     move-result-object v5
135
136     const-string v6, " e "
137
138     invoke-virtual {v5, v6},
139         Ljava/lang/StringBuffer;->append(Ljava/lang/String;)Ljava/lang/StringBuffer
140
141     move-result-object v5
142
143     move v6, v3
144
145     invoke-virtual {v5, v6},
146         Ljava/lang/StringBuffer;->append(I)Ljava/lang/StringBuffer;
147
148     move-result-object v5
149
150     invoke-virtual {v5},
151         Ljava/lang/StringBuffer;->toString()Ljava/lang/String;
152
153     move-result-object v5
154
155     invoke-virtual {v4, v5},
156         Ljava/io/PrintStream;->println(Ljava/lang/String;)V
157
158     .line 15
159     return-void
160 .end method
```

A chamada do fatorial acontece de forma análoga ao java. Pode-se observar isso na linha 49. Na linha 30, é verificado se o parâmetro recebido é maior que 0. Se for maior que 0 o código vai para o target :cond_7 onde outra chamada de fatorial é feita na linha 49 já mencionada. Caso o número do parâmetro seja 0, o código retorna o parâmetro e começa a resolução da recursão.

Saída

5.2.23 Algoritmo micro11

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro10$ java micr
o10
Digite um numero: 1
0 fatorial de 1 e 1
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro10$ java micr
o10
Digite um numero: 7
0 fatorial de 7 e 5040
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro10$
```

Listing 70: Código em Java

```
1 import java.util.Scanner;
2
3 public class micro11
4 {
5     public static void main(String[] args)
6     {
7         Scanner s = new Scanner(System.in);
8         int numero=0, x;
9         System.out.print("Digite um numero: ");
10        numero = s.nextInt();
11        x = verifica(numero);
12        if(x ==1) System.out.println("Numero Positivo");
13        else if (x==0) System.out.println("Zero");
14        else System.out.println("Numero Negativo");
15
16    }
17
18    public static int verifica(int n){
19        int res;
20        if(n>0) res =1;
21        else if(n<0) res = -1;
22        else res = 0;
23
24
25        return res;
26    }
27
28
29 }
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

Listing 71: Código em python

```
1 def micro11():
2     numero,x =0,0
3     print("Digite um numero: ",end="")
4     numero = int(input())
5     x = verifica(numero)
6     if x ==1:
7         print("Numero Positivo")
8     elif x ==0:
9         print("Zero")
10    else:
11        print("Negativo")
12
13 def verifica(n):
14     res = 0
15     if n>0:
16         res = 1
17     elif n<0:
18         res = -1
19     else:
20         res = 0
21
22     return res
```

Listing 72: Smali resultante do .java

```
1 .class public Lmicro11;
2 .super Ljava/lang/Object;
3 .source "micro11.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 3
9
10    .prologue
11    .line 3
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
12  move-object v0, p0
13
14  move-object v1, v0
15
16  invoke-direct {v1}, Ljava/lang/Object; -> <init>()V
17
18  return-void
19 .end method
20
21 .method public static main([Ljava/lang/String;)V
22   .registers 9
23
24   .prologue
25   .line 7
26   move-object v0, p0
27
28   new-instance v4, Ljava/util/Scanner;
29
30   move-object v7, v4
31
32   move-object v4, v7
33
34   move-object v5, v7
35
36   sget-object v6, Ljava/lang/System; -> in:Ljava/io/InputStream;
37
38   invoke-direct {v5, v6},
39       Ljava/util/Scanner; -> <init>(Ljava/io/InputStream;)V
40
41   move-object v1, v4
42
43   .line 8
44   const/4 v4, 0x0
45
46   move v2, v4
47
48   .line 9
49   sget-object v4,
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```

    Ljava/lang/System;->out:Ljava/io/PrintStream;
49
50  const-string v5, "Digite um numero: "
51
52  invoke-virtual {v4, v5},
    Ljava/io/PrintStream;->print(Ljava/lang/String;)V
53
54  .line 10
55  move-object v4, v1
56
57  invoke-virtual {v4}, Ljava/util/Scanner;->nextInt()I
58
59  move-result v4
60
61  move v2, v4
62
63  .line 11
64  move v4, v2
65
66  invoke-static {v4}, Lmicro11;->verifica(I)I
67
68  move-result v4
69
70  move v3, v4
71
72  .line 12
73  move v4, v3
74
75  const/4 v5, 0x1
76
77  if-ne v4, v5, :cond_2d
78
79  sget-object v4,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
80
81  const-string v5, "Numero Positivo"
82
83  invoke-virtual {v4, v5},
```

5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```

    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
84
85  .line 16
86  :goto_2c
87  return-void
88
89  .line 13
90  :cond_2d
91  move v4, v3
92
93  if-nez v4, :cond_38
94
95  sget-object v4,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
96
97  const-string v5, "Zero"
98
99  invoke-virtual {v4, v5},
    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
100
101  goto :goto_2c
102
103  .line 14
104  :cond_38
105  sget-object v4,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
106
107  const-string v5, "Numero Negativo"
108
109  invoke-virtual {v4, v5},
    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
110
111  goto :goto_2c
112  .end method
113
114  .method public static verifica(I)I
115  .registers 4
116
```


5 ANÁLISE DO COMPORTAMENTO DE COMPILAÇÃO DE .JAVA PARA .SMALI

```
117 .prologue
118 .line 20
119 move v0, p0
120
121 move v2, v0
122
123 if-lez v2, :cond_9
124
125 const/4 v2, 0x1
126
127 move v1, v2
128
129 .line 25
130 :goto_6
131 move v2, v1
132
133 move v0, v2
134
135 return v0
136
137 .line 21
138 :cond_9
139 move v2, v0
140
141 if-gez v2, :cond_f
142
143 const/4 v2, -0x1
144
145 move v1, v2
146
147 goto :goto_6
148
149 .line 22
150 :cond_f
151 const/4 v2, 0x0
152
153 move v1, v2
154
```

```
155     goto :goto_6
156 .end method
```

Saída

```
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro11$ java micr
o11
Digite um numero: 1
Numero Positivo
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro11$ java micr
o11
Digite um numero: -1
Numero Negativo
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro11$ java micr
o11
Digite um numero: 0
Zero
gabriel@gabriel-Inspiron-5547:~/GIT/Compiladores/CódigosTeste/micro11$
```

6 Analisador Léxico

6.1 Abordagem por Autômato

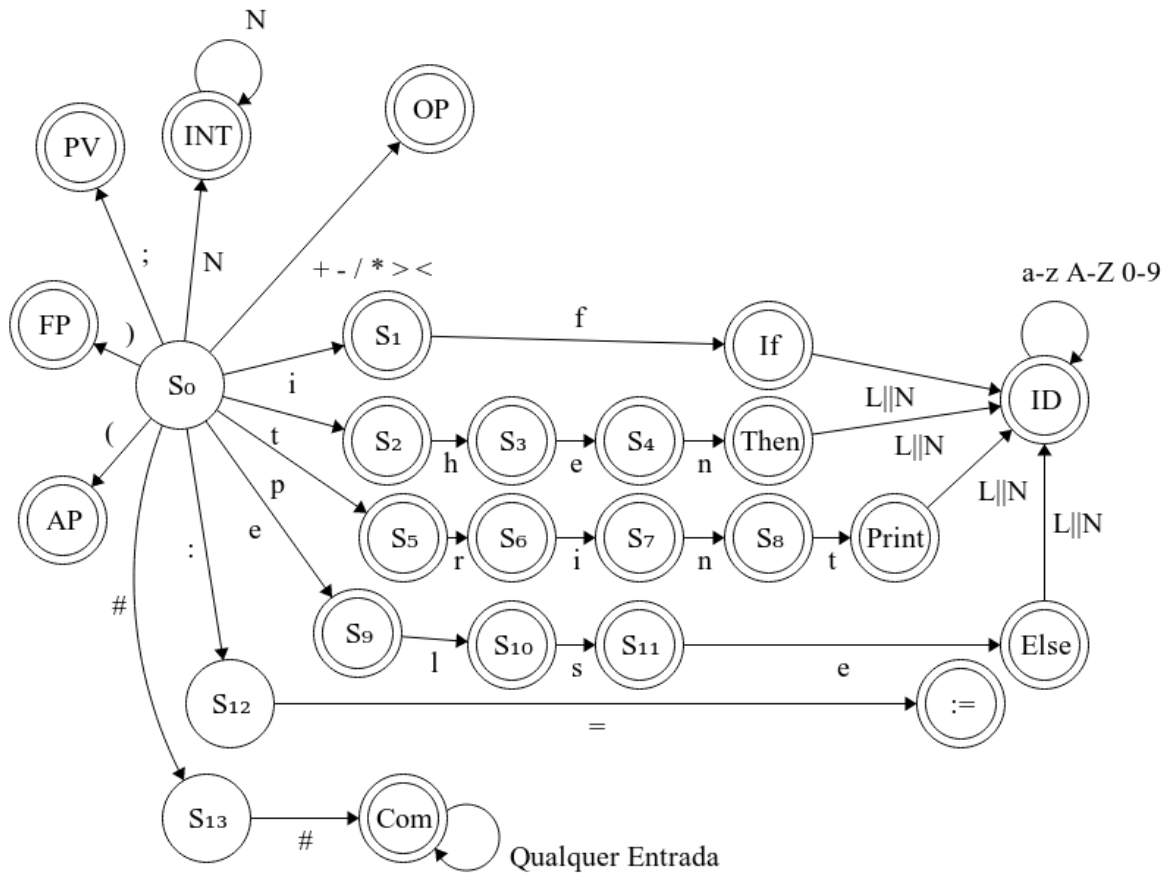
Segue uma versão simplificada (pela facilidade de entender o processo) do autômato, que será implementado em Ocaml, sendo que algumas considerações foram feitas:

1. $L = [a - z \cup A - Z]$;
2. $N = [0 - 9]$;
3. INT = estado final para reconhecimento de números inteiros ;
4. PV = estado final para reconhecimento de ponto e vírgula;
5. OP = estado final para reconhecimento de operadores;
6. AP = estado final para reconhecimento de abrir parênteses;
7. FP = estado final para reconhecimento de fechar parênteses;
8. ID = estado final para categorizar identificadores;
9. Com = estado final para categorizar um comentário;
10. *if*, *then*, *else* e *print* constituem palavras reservadas da linguagem;

11. cada estado das palavras reservadas tem uma seta até id lendo L ou N . Isso viabiliza identificadores como *el1 the1 then1 e else1*.

Vale ressaltar que o autômato não reconhece a linguagem Python. É reconhecida uma linguagem convencionada na sala de aula.

Foi utilizado o código fornecido pelo professor Alexsandro, feito durante as aulas de Compiladores.



6.1.1 Implementação

Além das palavras reservadas *if then else e print*, também foram implementadas as palavras *for e while*. Segue a função léxico que representa a implementação do autômato.

Listing 73: Código em python

```
1  let lexico (str:entrada) =
2  let trans (e:estado) (c:simbolo) =
3      match (e,c) with
4          | (0, 'i') -> 1
5          | (0, 't') -> 6
6          | (0, 'e') -> 10
7          | (0, 'p') -> 14
8          | (0, 'f') -> 25
9          | (0, 'w') -> 28
10         | (0, '(') -> 19
11         | (0, ')') -> 20
12         | (0, ';') -> 22
13         | (0, ':') -> 23
14         | (0, _) when eh_operador c -> 21
15         | (0, _) when eh_letra c -> 3
16         | (0, _) when eh_digito c -> 4
17         | (0, _) when eh_branco c -> 5
18         | (0, _) -> failwith ("Erro lexico: caracter
19                               desconhecido " ^ Char.escaped c)
20
21         | (1, 'f') -> 2
22         | (1, _) when eh_letra c || eh_digito c -> 3
23
24         | (2, _) when eh_letra c || eh_digito c -> 3
25
26         | (3, _) when eh_letra c || eh_digito c -> 3
27
28         | (4, _) when eh_digito c -> 4
29
30         | (5, _) when eh_branco c -> 5
31
32         | (6, 'h') -> 7
33         | (6, _) when eh_letra c || eh_digito c -> 3
34
35         | (7, 'e') -> 8
36         | (7, _) when eh_letra c || eh_digito c -> 3
```

6 ANALISADOR LÉXICO

```
36 | (8, 'n') -> 9
37 | (8, _) when eh_letra c || eh_digito c -> 3
38
39 | (9, _) when eh_letra c || eh_digito c -> 3
40
41 | (10, 'l') -> 11
42 | (10, _) when eh_letra c || eh_digito c -> 3
43
44 | (11, 's') -> 12
45 | (11, _) when eh_letra c || eh_digito c -> 3
46
47 | (12, 'e') -> 13
48 | (12, _) when eh_letra c || eh_digito c -> 3
49
50 | (13, _) when eh_letra c || eh_digito c -> 3
51
52 | (14, 'r') -> 15
53 | (14, _) when eh_letra c || eh_digito c -> 3
54
55 | (15, 'i') -> 16
56 | (15, _) when eh_letra c || eh_digito c -> 3
57
58 | (16, 'n') -> 17
59 | (16, _) when eh_letra c || eh_digito c -> 3
60
61 | (17, 't') -> 18
62 | (17, _) when eh_letra c || eh_digito c -> 3
63
64 | (18, _) when eh_letra c || eh_digito c -> 3
65
66 | (23, '=') -> 24
67
68 | (25, 'o') -> 26
69 | (25, _) when eh_letra c || eh_digito c -> 3
70
71 | (26, 'r') -> 27
72 | (27, _) when eh_letra c || eh_digito c -> 3
73
```

6 ANALISADOR LÉXICO

```
74         | (27, _) when eh_letra c || eh_digito c -> 3
75
76         | (28, 'h') -> 29
77         | (28, _) when eh_letra c || eh_digito c -> 3
78
79         | (29, 'i') -> 30
80         | (29, _) when eh_letra c || eh_digito c -> 3
81
82         | (30, 'l') -> 31
83         | (30, _) when eh_letra c || eh_digito c -> 3
84
85         | (31, 'e') -> 32
86         | (31, _) when eh_letra c || eh_digito c -> 3
87
88         | (32, _) when eh_letra c || eh_digito c -> 3
89
90         | _ -> estado_morto
91
92     and rotulo e str =
93         match e with
94         | 2 -> If
95         | 1
96         | 6
97         | 7
98         | 8
99         | 10
100        | 11
101        | 12
102        | 14
103        | 15
104        | 16
105        | 17
106        | 25
107        | 26
108        | 28
109        | 29
110        | 30
111        | 31
```

6 ANALISADOR LÉXICO

```
112         | 3 -> Id str
113         | 4 -> Int str
114         | 5 -> Branco
115         | 9 -> Then
116         | 13 -> Else
117         | 18 -> Print
118         | 19 -> APar
119         | 20 -> FPar
120         | 21 -> OP str
121         | 22 -> PV
122         | 24 -> Atrib
123         | 27 -> For
124         | 32 -> While
125         | _ -> failwith ("Erro lexico: sequencia desconhecida
            " ^ str)
126     in let dfa = { transicao = trans;
127       estado = estado_inicial;
128       posicao = 0 }
129     in let estado_lexico = {
130       pos_inicial = 0;
131       pos_final = -1;
132       ultimo_final = -1;
133       rotulo = rotulo;
134       dfa = dfa
135     } in
136     analisador str (String.length str) estado_lexico
```

Seguem as funções principais feitas pelo professor que viabilizaram o funcionamento da função léxico:

1. `obtem_token_e_estado`: função responsável por atualizar o autômato para o próximo estado dado uma string e um estado léxico como parâmetro.
2. `analisador`: responsável por analisar o estado corrente do estado léxico. Se estivermos no estado final, paramos, caso contrario, se estivermos no estado morto, vamos para o proximo estado e verificamos qual ele é.

Listing 74: Código em Java

```
1      let obtem_token_e_estado (str:entrada)
      (el:estado_lexico) =
2      let inicio = el.pos_inicial
      and fim = el.pos_final
3      and estado_final = el.ultimo_final
4      and rotulo = el.rotulo in
5      let tamanho = fim - inicio + 1 in
6      let lexema = String.sub str inicio tamanho in (*pega sub
7      string*)
8      let token = rotulo estado_final lexema in
9      let proximo_el = { el with pos_inicial = fim + 1;
10      pos_final = -1;
11      ultimo_final = -1;
12      dfa = { el.dfa with estado = estado_inicial;
      (*volta pro estado inicial*)
13      posicao = fim + 1 }}
14      in
15      (token, proximo_el)
16
17
18      let rec analisador (str:entrada) (tam:int)
      (el:estado_lexico) =
19      let posicao_atual = el.dfa.posicao
20      and estado_atual = el.dfa.estado in
21      if posicao_atual >= tam
22      then
23          if el.ultimo_final >= 0
24          then let token, proximo_el = obtem_token_e_estado
      str el in
25              [token; EOF]
26          else [EOF]
27      else
28          let simbolo = str.[posicao_atual]
29          and transicao = el.dfa.transicao in
30          let proximo_estado = transicao estado_atual simbolo
      in
```



```
31         if proximo_estado = estado_morto
32         then let token, proximo_el = obtem_token_e_estado
           str el in
33             token :: analisador str tam proximo_el
34     else
35         let proximo_el =
36             if eh_estado_final proximo_estado el
37             then { el with pos_final = posicao_atual;
38                   ultimo_final = proximo_estado;
39                   dfa = { el.dfa with estado = proximo_estado;
40                         posicao = posicao_atual + 1 }}
41             else { el with dfa = { el.dfa with estado =
42                                   proximo_estado;
43                                   posicao = posicao_atual + 1 }}
44         in
           analisador str tam proximo_el
```

6.1.2 Testes

Para testar o algoritmo basta abrirmos o intepretador ocaml no mesmo diretório dos arquivos fonte com **rlwrap ocaml**. Em seguida, em virtudes de utilizarmos as funções contidas nesses arquivos, digitamos **# use nome-DoArquivo.ml;;**

6 ANALISADOR LÉXICO

```
#use "dfalexer.ml";;
type estado = int
type entrada = string
type simbolo = char
type posicao = int
type dfa = {
  transicao : estado -> simbolo -> estado;
  estado : estado;
  posicao : posicao;
}
type token =
  | If
  | Then
  | Else
  | Id of string
  | Int of string
  | Print
  | APar
  | FPar
  | OP of string
  | Atrib
  | PV
  | Branco
  | EOF
  | For
  | While
type estado_lexico = {
  pos_inicial : posicao;
  pos_final : posicao;
  ultimo_final : estado;
  dfa : dfa;
  rotulo : estado -> entrada -> token;
}
val estado_morto : estado = -1
val estado_inicial : estado = 0
val eh_letra : simbolo -> bool = <fun>
val eh_digito : simbolo -> bool = <fun>
val eh_branco : simbolo -> bool = <fun>
val eh_operador : simbolo -> bool = <fun>
val eh_estado_final : estado -> estado_lexico -> bool = <fun>
val obtem_token_e_estado : entrada -> estado_lexico -> token * estado_lexico =
  <fun>
val analisador : entrada -> posicao -> estado_lexico -> token list = <fun>
val lexico : entrada -> token list = <fun>
```

Palavras Reservadas

Foram testadas as palavras reservadas **if**, **then**, **else**, **print**, **for** e **while**. Percebe-se que o autômato também pega espaços em branco.

```
lexico "if";;
- : token list = [If; EOF]
# lexico "then";;
- : token list = [Then; EOF]
# lexico "else";;
- : token list = [Else; EOF]
# lexico "for";;
- : token list = [For; EOF]
# lexico "while";;
- : token list = [While; EOF]
# lexico " for ";;
- : token list = [Branco; For; Branco; EOF]
# lexico " while ";;
- : token list = [Branco; While; Branco; EOF]
# lexico " else ";;
- : token list = [Branco; Else; Branco; EOF]
# lexico " then ";;
- : token list = [Branco; Then; Branco; EOF]
# lexico " if ";;
- : token list = [Branco; If; Branco; EOF]
```

Identificadores e Números inteiros

```
lexico "if1";;  
- : token list = [Id "if1"; EOF]  
# lexico "el2";;  
- : token list = [Id "el2"; EOF]  
# lexico "th3en";;  
- : token list = [Id "th3en"; EOF]  
# lexico "23";;  
- : token list = [Int "23"; EOF]  
# lexico "23 3 4 5 2";;  
- : token list =  
[Int "23"; Branco; Int "3"; Branco; Int "4"; Branco; Int "5"; Branco;  
Int "2"; EOF]  
# lexico "if2 then3 else4 for5 while2";;  
- : token list =  
[Id "if2"; Branco; Id "then3"; Branco; Id "else4"; Branco; Id "for5"; Branco;  
Id "while2"; EOF]
```

Operadores, Atribuição, Ponto e Vírgula e Parênteses

Repare que o autômato não reconhece o sinal de = como atribuição. Ele somente reconhece := .

```
lexico " = ";;  
Exception: Failure "Erro lexico: character desconhecido =".  
# lexico " := ";;  
- : token list = [Branco; Atrib; Branco; EOF]  
# lexico " ( ) ";;  
- : token list = [Branco; APar; FPar; Branco; EOF]  
# lexico " ; ";;  
- : token list = [Branco; PV; Branco; EOF]  
# lexico " > ";;  
- : token list = [Branco; OP ">"; Branco; EOF]  
# lexico " < ";;  
- : token list = [Branco; OP "<"; Branco; EOF]  
# lexico " + ";;  
- : token list = [Branco; OP "+"; Branco; EOF]  
# lexico " - ";;  
- : token list = [Branco; OP "-"; Branco; EOF]  
# lexico " * ";;  
- : token list = [Branco; OP "*"; Branco; EOF]  
# lexico " / ";;  
- : token list = [Branco; OP "/"; Branco; EOF]  
#
```

Comentários

Convecionou-se que os comentários serão determinados por ## em virtude da proximidade com Python.

```
lexico "then print(if1) ##print if 1";;  
- : token list = [Then; Branco; Print; APar; Id "if1"; FPar; Branco; EOF]  
# lexico "for(i:=0;i<id1;i++) ##comentario nao deve aparecer";;  
- : token list =  
[For; APar; Id "i"; Atrib; Int "0"; PV; Id "i"; OP "<"; Id "id1"; PV;  
Id "i"; OP "+"; OP "+"; FPar; Branco; EOF]  
# lexico "1 ##comentario ##nao ##deve ##aparecer";;  
- : token list = [Int "1"; Branco; EOF]  
#
```

Exemplos dados em Aula

Seguem os exemplos que serão testados:

1. $i := 1 + b$;
2. $\text{print}(a * b)$;
3. $\text{if1} := a - 2$;
4. $\text{then print}(\text{if1})$;
5. $\text{else print}(\text{if2})$;

```
lexico "i := 1 + b ;";;
- : token list =
[Id "i"; Branco; Atrib; Branco; Int "1"; Branco; OP "+"; Branco; Id "b";
 Branco; PV; EOF]
# lexico "print(a*b);";;
- : token list = [Print; APar; Id "a"; OP "**"; Id "b"; FPar; PV; EOF]
# lexico "if1 := a-2;";;
- : token list =
[Id "if1"; Branco; Atrib; Branco; Id "a"; OP "-"; Int "2"; PV; EOF]
# lexico "then print(if1);";;
- : token list = [Then; Branco; Print; APar; Id "if1"; FPar; PV; EOF]
# lexico "else print(if2);";;
- : token list = [Else; Branco; Print; APar; Id "if2"; FPar; PV; EOF]
#
```

6.2 Abordagem por Linguagem Regular

Foi feita uma implementação de linguagem regular em virtude de facilitar o processo de análise léxica.

6.2.1 Implementação

Os principais tokens são NOVALINHA, IDENTA e DEDENTA que realizam o controle de indentação do python. NOVALINHA é inserido após a quebra de linha e pode ser seguido do identa caso algum comando seja dado. EX.: for, while, ou seja, comandos que exigem indentação, O token DEDENTA é utilizado toda a vez a indentação volta ao normal o que significa o fim de um comando.

Fazendo uma analogia com a linguagem C, pode-se dizer que o token NOVALINHA representa a vírgula, IDENTA representa abrir chaves e DEDENTA, fechar chaves.

6.2.2 Testes

Código

```
1 def nano01(): -> void :  
2     pass
```

Saída

Listing 75: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "nano01"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.PASS; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

Código

```
1 def nano02(): -> void :  
2     n = 0
```

Saída

Listing 76: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "nano02"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 0;  
   Lexico.NOVALINHA;  
5 Lexico.DEDENTA; Lexico.EOF]
```

Código

```
1 def nano03(): -> void :  
2     n=1
```

Saída

6 ANALISADOR LÉXICO

Listing 77: Analisador Léxico

```
1 - : tokens =
2   [Lexico.DEF; Lexico.ID "nano03"; Lexico.APAR; Lexico.FPAR;
3     Lexico.DPONTOS;
4     Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
5     Lexico.INDENTA;
6     Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;
7     Lexico.NOVALINHA;
8     Lexico.DEDENTA; Lexico.EOF]
```

Código

```
1 def nano04(): -> void :
2   n = 1 + 2
```

Saída

Listing 78: Analisador Léxico

```
1 - : tokens =
2   [Lexico.DEF; Lexico.ID "nano04"; Lexico.APAR; Lexico.FPAR;
3     Lexico.DPONTOS;
4     Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
5     Lexico.INDENTA;
6     Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.MAIS;
7     Lexico.LITINT 2;
8     Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

Código

```
1 def nano05(): -> void :
2   n = 2
3   print(n)
4
5
6 nano05()
```

Saída

Listing 79: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "nano05"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 2;
   Lexico.NOVALINHA;
5 Lexico.PRINT; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR;
   Lexico.NOVALINHA;
6 Lexico.DEDENTA; Lexico.ID "nano05"; Lexico.APAR;
   Lexico.FPAR;
7 Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano06(): -> void :
2     n = 1 - 2
3     print(n)
4
5
6 nano06()
```

Saída

Listing 80: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "nano06"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1; Lexico.MENOS;
   Lexico.LITINT 2;
5 Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
   Lexico.FPAR;
6 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "nano06";
   Lexico.APAR;
7 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano07(): -> void :
2     n=1
3     if n ==1:
4         print(n)
5
6
7 nano07()
```

Saída

Listing 81: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "nano07"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;
   Lexico.NOVALINHA; Lexico.IF;
5 Lexico.ID "n"; Lexico.IGUALDADE; Lexico.LITINT 1;
   Lexico.DPONTOS;
6 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT;
   Lexico.APAR; Lexico.ID "n";
7 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
   Lexico.DEDENTA;
8 Lexico.ID "nano07"; Lexico.APAR; Lexico.FPAR;
   Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano08(): -> void :
2     n=1
3     if n ==1:
4         print(n)
5     else:
6         print(0)
```



```
7  
8  
9 nano08()
```

Saída

Listing 82: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "nano08"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;  
   Lexico.NOVALINHA; Lexico.IF;  
5 Lexico.ID "n"; Lexico.IGUALDADE; Lexico.LITINT 1;  
   Lexico.DPONTOS;  
6 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT;  
   Lexico.APAR; Lexico.ID "n";  
7 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;  
   Lexico.DPONTOS;  
8 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;  
9 Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA;  
   Lexico.DEDENTA;  
10 Lexico.DEDENTA; Lexico.ID "nano08"; Lexico.APAR;  
   Lexico.FPAR;  
11 Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano10(): -> void :  
2     n=1  
3     m=2  
4     if n ==m:  
5         print(n)  
6     else:  
7         print(0)  
8  
9
```

```
10 nano10()
```

Saída

Listing 83: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "nano10"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;  
   Lexico.NOVALINHA;  
5 Lexico.ID "m"; Lexico.ATRIB; Lexico.LITINT 2;  
   Lexico.NOVALINHA; Lexico.IF;  
6 Lexico.ID "n"; Lexico.IGUALDADE; Lexico.ID "m";  
   Lexico.DPONTOS;  
7 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT;  
   Lexico.APAR; Lexico.ID "n";  
8 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;  
   Lexico.DPONTOS;  
9 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;  
10 Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA;  
   Lexico.DEDENTA;  
11 Lexico.DEDENTA; Lexico.ID "nano10"; Lexico.APAR;  
   Lexico.FPAR;  
12 Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano11(): -> void :  
2     n=1  
3     m=2  
4     x=5  
5     while x >n:  
6         n = n + m  
7         print(n)  
8  
9
```

```
10 nano11()
```

Saída

Listing 84: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "nano11"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;  
   Lexico.NOVALINHA;  
5 Lexico.ID "m"; Lexico.ATRIB; Lexico.LITINT 2;  
   Lexico.NOVALINHA;  
6 Lexico.ID "x"; Lexico.ATRIB; Lexico.LITINT 5;  
   Lexico.NOVALINHA;  
7 Lexico.WHILE; Lexico.ID "x"; Lexico.MAIOR; Lexico.ID "n";  
   Lexico.DPONTOS;  
8 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n";  
   Lexico.ATRIB;  
9 Lexico.ID "n"; Lexico.MAIS; Lexico.ID "m";  
   Lexico.NOVALINHA; Lexico.PRINT;  
10 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;  
   Lexico.DEDENTA;  
11 Lexico.DEDENTA; Lexico.ID "nano11"; Lexico.APAR;  
   Lexico.FPAR;  
12 Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def nano12(): -> void :  
2     n=1  
3     m=2  
4     x=5  
5     while x >n:  
6         if n ==m:  
7             print(n)  
8         else:
```

6 ANALISADOR LÉXICO

```
9         print(0)
10         x = x -1
11
12 nano12()
```

Saída

Listing 85: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "nano12"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "n"; Lexico.ATRIB; Lexico.LITINT 1;
   Lexico.NOVALINHA;
5 Lexico.ID "m"; Lexico.ATRIB; Lexico.LITINT 2;
   Lexico.NOVALINHA;
6 Lexico.ID "x"; Lexico.ATRIB; Lexico.LITINT 5;
   Lexico.NOVALINHA;
7 Lexico.WHILE; Lexico.ID "x"; Lexico.MAIOR; Lexico.ID "n";
   Lexico.DPONTOS;
8 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.IF; Lexico.ID "n";
9 Lexico.IGUALDADE; Lexico.ID "m"; Lexico.DPONTOS;
   Lexico.NOVALINHA;
10 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
   Lexico.FPAR;
11 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;
   Lexico.DPONTOS;
12 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
13 Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA;
   Lexico.DEDENTA;
14 Lexico.ID "x"; Lexico.ATRIB; Lexico.ID "x"; Lexico.MENOS;
   Lexico.LITINT 1;
15 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID
   "nano12";
16 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

6 ANALISADOR LÉXICO

```
1 def micro01(): -> void :
2     cel , far = 1.0 , 0.0
3     print("      Tabela de conversao: Celsius -> Fahrenheit")
4     print("Digite a temperatura em Celsius: ")
5     cel = input()
6     far = (9*cel+160)/5
7     print("A nova temperatura e:"+str(far)+"F")
```

Saída

Listing 86: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro01"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "cel"; Lexico.VIRG; Lexico.ID "far"; Lexico.ATRIB;
5 Lexico.LITINT 1; Lexico.PONTO; Lexico.LITINT 0; Lexico.VIRG;
6 Lexico.LITINT 0; Lexico.PONTO; Lexico.LITINT 0;
   Lexico.NOVALINHA;
7 Lexico.PRINT; Lexico.APAR;
8 Lexico.LITSTRING "\t\tTabela de conversao: Celsius ->
   Fahrenheit";
9 Lexico.FPAR; Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
10 Lexico.LITSTRING "Digite a temperatura em Celsius: ";
   Lexico.FPAR;
11 Lexico.NOVALINHA; Lexico.ID "cel"; Lexico.ATRIB;
   Lexico.INPUT; Lexico.APAR;
12 Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "far";
   Lexico.ATRIB; Lexico.APAR;
13 Lexico.LITINT 9; Lexico.VEZES; Lexico.ID "cel"; Lexico.MAIS;
14 Lexico.LITINT 160; Lexico.FPAR; Lexico.DIVIDIDO;
   Lexico.LITINT 5;
15 Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
16 Lexico.LITSTRING "A nova temperatura e: "; Lexico.MAIS;
   Lexico.STR;
17 Lexico.APAR; Lexico.ID "far"; Lexico.FPAR; Lexico.MAIS;
18 Lexico.LITSTRING "F"; Lexico.FPAR; Lexico.NOVALINHA;
```

```
19 Lexico.NOVALINHA;  
Lexico.DEDENTA; Lexico.EOF]
```

Código

```
1 def micro02(): -> void :  
2     num1, num2 = 0 , 0  
3     print("Digite o primeiro numero: ")  
4     num1 = int(input())  
5     print("Digite o segundo numero: ")  
6     num2 = int(input())  
7  
8     if num1 > num2:  
9         print("O primeiro numero "+str(num1)+" e maior que o  
10             segundo "+str(num2))  
11     else:  
12         print("O segundo numero "+str(num2)+" e maior que o  
13             primeiro "+str(num1))  
14 micro02()
```

Saída

Listing 87: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "micro02"; Lexico.APAR; Lexico.FPAR;  
   Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
   Lexico.INDENTA;  
4 Lexico.ID "num1"; Lexico.VIRG; Lexico.ID "num2";  
   Lexico.ATRIB;  
5 Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT 0;  
   Lexico.NOVALINHA;  
6 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite o  
   primeiro numero: ";  
7 Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "num1";  
   Lexico.ATRIB; Lexico.INT;
```

```
8 Lexico.APAR; Lexico.INPUT; Lexico.APAR; Lexico.FPAR;
   Lexico.FPAR;
9 Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
10 Lexico.LITSTRING "Digite o segundo numero: "; Lexico.FPAR;
   Lexico.NOVALINHA;
11 Lexico.ID "num2"; Lexico.ATRIB; Lexico.INT; Lexico.APAR;
   Lexico.INPUT;
12 Lexico.APAR; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA;
   Lexico.IF;
13 Lexico.ID "num1"; Lexico.MAIOR; Lexico.ID "num2";
   Lexico.DPONTOS;
14 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
15 Lexico.LITSTRING "O primeiro numero "; Lexico.MAIS;
   Lexico.STR; Lexico.APAR;
16 Lexico.ID "num1"; Lexico.FPAR; Lexico.MAIS;
17 Lexico.LITSTRING " e maior que o segundo "; Lexico.MAIS;
   Lexico.STR;
18 Lexico.APAR; Lexico.ID "num2"; Lexico.FPAR; Lexico.FPAR;
   Lexico.NOVALINHA;
19 Lexico.DEDENTA; Lexico.ELSE; Lexico.DPONTOS;
   Lexico.NOVALINHA;
20 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
21 Lexico.LITSTRING "O segundo numero "; Lexico.MAIS;
   Lexico.STR; Lexico.APAR;
22 Lexico.ID "num2"; Lexico.FPAR; Lexico.MAIS;
23 Lexico.LITSTRING " e maior que o primeiro "; Lexico.MAIS;
   Lexico.STR;
24 Lexico.APAR; Lexico.ID "num1"; Lexico.FPAR; Lexico.FPAR;
   Lexico.NOVALINHA;
25 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID "micro02";
   Lexico.APAR;
26 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro03(): -> void :
2     numero =0
3     print("Digite um numero: ")
```

```
4  numero = int(input())
5  if numero >= 100:
6      if numero <= 200:
7          print("O numero esta no intervalo entre 100 e 200")
8      else:
9          print("O numero nao esta no intervalo entre 100 e 200")
10 else:
11     print("O numero nao esta no intervalo entre 100 e 200")
12
13 micro03()
```

Saída

Listing 88: Analisador Léxico

```
1  - : tokens =
2  [Lexico.DEF; Lexico.ID "micro03"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3  Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4  Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
5  Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um
   numero: ";
6  Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "numero";
   Lexico.ATRIB; Lexico.INT;
7  Lexico.APAR; Lexico.INPUT; Lexico.APAR; Lexico.FPAR;
   Lexico.FPAR;
8  Lexico.NOVALINHA; Lexico.IF; Lexico.ID "numero";
   Lexico.MAIORIGUAL;
9  Lexico.LITINT 100; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
10 Lexico.IF; Lexico.ID "numero"; Lexico.MENORIGUAL;
   Lexico.LITINT 200;
11 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
   Lexico.PRINT; Lexico.APAR;
12 Lexico.LITSTRING "O numero esta no intervalo entre 100 e
   200"; Lexico.FPAR;
13 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;
```



```
Lexico.DPONTOS;
14 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
15 Lexico.LITSTRING "0 numero nao esta no intervalo entre 100
    e 200";
16 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
    Lexico.DEDENTA; Lexico.ELSE;
17 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
    Lexico.PRINT; Lexico.APAR;
18 Lexico.LITSTRING "0 numero nao esta no intervalo entre 100
    e 200";
19 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
    Lexico.DEDENTA;
20 Lexico.ID "micro03"; Lexico.APAR; Lexico.FPAR;
    Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro04(): -> void :
2     x,num,intervalo = 0,0,0
3
4     for x in range(5):
5         print("Digite o numero: ")
6         num = int(input())
7         if num >=10:
8             if num <=150:
9                 intervalo = intervalo +1
10
11         print("Ao total, foram digitados "+str(intervalo)+" numeros
12             no intervalo entre 10 e 150")
13
14 micro04()
```

Saída

Listing 89: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro04"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
```

6 ANALISADOR LÉXICO

```
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA;
4 Lexico.ID "x"; Lexico.VIRG; Lexico.ID "num"; Lexico.VIRG;
5 Lexico.ID "intervalo"; Lexico.ATRIB; Lexico.LITINT 0;
  Lexico.VIRG;
6 Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT 0;
  Lexico.NOVALINHA; Lexico.FOR;
7 Lexico.ID "x"; Lexico.IN; Lexico.RANGE; Lexico.APAR;
  Lexico.LITINT 5;
8 Lexico.FPAR; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA; Lexico.PRINT;
9 Lexico.APAR; Lexico.LITSTRING "Digite o numero: ";
  Lexico.FPAR;
10 Lexico.NOVALINHA; Lexico.ID "num"; Lexico.ATRIB;
  Lexico.INT; Lexico.APAR;
11 Lexico.INPUT; Lexico.APAR; Lexico.FPAR; Lexico.FPAR;
  Lexico.NOVALINHA;
12 Lexico.IF; Lexico.ID "num"; Lexico.MAIORIGUAL;
  Lexico.LITINT 10;
13 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.IF;
14 Lexico.ID "num"; Lexico.MENORIGUAL; Lexico.LITINT 150;
  Lexico.DPONTOS;
15 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "intervalo";
  Lexico.ATRIB;
16 Lexico.ID "intervalo"; Lexico.MAIS; Lexico.LITINT 1;
  Lexico.NOVALINHA;
17 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.DEDENTA;
  Lexico.PRINT; Lexico.APAR;
18 Lexico.LITSTRING "Ao total, foram digitados "; Lexico.MAIS;
  Lexico.STR;
19 Lexico.APAR; Lexico.ID "intervalo"; Lexico.FPAR;
  Lexico.MAIS;
20 Lexico.LITSTRING " numeros no intervalo entre 10 e 150";
  Lexico.FPAR;
21 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "micro04";
  Lexico.APAR;
22 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro05(): -> void :
2     x,h,h = 0,0,0
3     nome,sexo = "", ""
4
5     for x in range(5):
6         print("Digite o nome: ")
7         nome = input()
8         print("H - Homem ou M - Mulher")
9         sexo = input()
10        if sexo == "H":
11            h = h+1
12        elif sexo == "M":
13            m = m+1
14        else:
15            print("Sexo so pode ser H ou M!")
16
17    print("Foram inseridos "+h+" Homens")
18    print("Foram inseridas "+m+" Mulheres")
19
20
21 micro05()
```

Saída

Listing 90: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro05"; Lexico.APAR; Lexico.FPAR;
3   Lexico.DPONTOS;
4   Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
5   Lexico.INDENTA;
6   Lexico.ID "x"; Lexico.VIRG; Lexico.ID "h"; Lexico.VIRG;
   Lexico.ID "h";
   Lexico.ATRIB; Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT
   0; Lexico.VIRG;
   Lexico.LITINT 0; Lexico.NOVALINHA; Lexico.ID "nome";
   Lexico.VIRG;
```

```
7 Lexico.ID "sexo"; Lexico.ATRIB; Lexico.LITSTRING "";
  Lexico.VIRG;
8 Lexico.LITSTRING ""; Lexico.NOVALINHA; Lexico.FOR;
  Lexico.ID "x"; Lexico.IN;
9 Lexico.RANGE; Lexico.APAR; Lexico.LITINT 5; Lexico.FPAR;
  Lexico.DPONTOS;
10 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
11 Lexico.LITSTRING "Digite o nome: "; Lexico.FPAR;
  Lexico.NOVALINHA;
12 Lexico.ID "nome"; Lexico.ATRIB; Lexico.INPUT; Lexico.APAR;
  Lexico.FPAR;
13 Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;
14 Lexico.LITSTRING "H - Homem ou M - Mulher"; Lexico.FPAR;
  Lexico.NOVALINHA;
15 Lexico.ID "sexo"; Lexico.ATRIB; Lexico.INPUT; Lexico.APAR;
  Lexico.FPAR;
16 Lexico.NOVALINHA; Lexico.IF; Lexico.ID "sexo";
  Lexico.IGUALDADE;
17 Lexico.LITSTRING "H"; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA;
18 Lexico.ID "h"; Lexico.ATRIB; Lexico.ID "h"; Lexico.MAIS;
  Lexico.LITINT 1;
19 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELIF; Lexico.ID
  "sexo";
20 Lexico.IGUALDADE; Lexico.LITSTRING "M"; Lexico.DPONTOS;
  Lexico.NOVALINHA;
21 Lexico.INDENTA; Lexico.ID "m"; Lexico.ATRIB; Lexico.ID "m";
  Lexico.MAIS;
22 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.DEDENTA;
  Lexico.ELSE;
23 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
  Lexico.PRINT; Lexico.APAR;
24 Lexico.LITSTRING "Sexo so pode ser H ou M!"; Lexico.FPAR;
  Lexico.NOVALINHA;
25 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.PRINT; Lexico.APAR;
26 Lexico.LITSTRING "Foram inseridos "; Lexico.MAIS; Lexico.ID
  "h";
27 Lexico.MAIS; Lexico.LITSTRING " Homens"; Lexico.FPAR;
```

```
Lexico.NOVALINHA;  
28 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Foram  
    inseridas "; Lexico.MAIS;  
29 Lexico.ID "m"; Lexico.MAIS; Lexico.LITSTRING " Mulheres";  
    Lexico.FPAR;  
30 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "micro05";  
    Lexico.APAR;  
31 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro06(): -> void :  
2     numero = 0  
3  
4     print("Digite um numero de 1 a 5: ")  
5     numero = int(input())  
6     if numero ==1:  
7         print("Um")  
8     elif numero == 2:  
9         print("Dois")  
10    elif numero ==3:  
11        print("Tres")  
12    elif numero ==4:  
13        print("Quatro")  
14    elif numero ==5:  
15        print("Cinco")  
16    else:  
17        print("Numero Invalido!!!")  
18 micro06()
```

Saída

Listing 91: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "micro06"; Lexico.APAR; Lexico.FPAR;  
    Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
    Lexico.INDENTA;
```

```
4 Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
5 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um
   numero de 1 a 5: ";
6 Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "numero";
   Lexico.ATRIB; Lexico.INT;
7 Lexico.APAR; Lexico.INPUT; Lexico.APAR; Lexico.FPAR;
   Lexico.FPAR;
8 Lexico.NOVALINHA; Lexico.IF; Lexico.ID "numero";
   Lexico.IGUALDADE;
9 Lexico.LITINT 1; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
10 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Um";
   Lexico.FPAR;
11 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELIF; Lexico.ID
   "numero";
12 Lexico.IGUALDADE; Lexico.LITINT 2; Lexico.DPONTOS;
   Lexico.NOVALINHA;
13 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING
   "Dois";
14 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELIF;
15 Lexico.ID "numero"; Lexico.IGUALDADE; Lexico.LITINT 3;
   Lexico.DPONTOS;
16 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
17 Lexico.LITSTRING "Tres"; Lexico.FPAR; Lexico.NOVALINHA;
   Lexico.DEDENTA;
18 Lexico.ELIF; Lexico.ID "numero"; Lexico.IGUALDADE;
   Lexico.LITINT 4;
19 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
   Lexico.PRINT; Lexico.APAR;
20 Lexico.LITSTRING "Quatro"; Lexico.FPAR; Lexico.NOVALINHA;
   Lexico.DEDENTA;
21 Lexico.ELIF; Lexico.ID "numero"; Lexico.IGUALDADE;
   Lexico.LITINT 5;
22 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
   Lexico.PRINT; Lexico.APAR;
23 Lexico.LITSTRING "Cinco"; Lexico.FPAR; Lexico.NOVALINHA;
   Lexico.DEDENTA;
```

6 ANALISADOR LÉXICO

```
24 Lexico.ELSE; Lexico.DPONTOS; Lexico.NOVALINHA;  
    Lexico.INDENTA; Lexico.PRINT;  
25 Lexico.APAR; Lexico.LITSTRING "Numero Invalido!!!";  
    Lexico.FPAR;  
26 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID  
    "micro06";  
27 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro07(): -> void :  
2     numero ,programa= 0,1  
3     opc = ""  
4  
5     while programa ==1:  
6         print("Digite um numero: ")  
7         numero = int(input())  
8  
9         if numero>0:  
10            print("Positivo")  
11        else:  
12            if numero==0:  
13                print("0 numero e igual a 0")  
14            if numero <0:  
15                print("Negativo")  
16  
17        print("Deseja Finalizar? (S/N) ")  
18        opc = input()  
19        if opc == "S":  
20            programa = 0  
21  
22 micro07()
```

Saída

Listing 92: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "micro07"; Lexico.APAR; Lexico.FPAR;
```

```
Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA;
4 Lexico.ID "numero"; Lexico.VIRG; Lexico.ID "programa";
  Lexico.ATRIB;
5 Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT 1;
  Lexico.NOVALINHA;
6 Lexico.ID "opc"; Lexico.ATRIB; Lexico.LITSTRING "";
  Lexico.NOVALINHA;
7 Lexico.WHILE; Lexico.ID "programa"; Lexico.IGUALDADE;
  Lexico.LITINT 1;
8 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
  Lexico.PRINT; Lexico.APAR;
9 Lexico.LITSTRING "Digite um numero: "; Lexico.FPAR;
  Lexico.NOVALINHA;
10 Lexico.ID "numero"; Lexico.ATRIB; Lexico.INT; Lexico.APAR;
  Lexico.INPUT;
11 Lexico.APAR; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA;
  Lexico.IF;
12 Lexico.ID "numero"; Lexico.MAIOR; Lexico.LITINT 0;
  Lexico.DPONTOS;
13 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
14 Lexico.LITSTRING "Positivo"; Lexico.FPAR; Lexico.NOVALINHA;
  Lexico.DEDENTA;
15 Lexico.ELSE; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA; Lexico.IF;
16 Lexico.ID "numero"; Lexico.IGUALDADE; Lexico.LITINT 0;
  Lexico.DPONTOS;
17 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
18 Lexico.LITSTRING "0 numero e igual a 0"; Lexico.FPAR;
  Lexico.NOVALINHA;
19 Lexico.DEDENTA; Lexico.IF; Lexico.ID "numero"; Lexico.MENOR;
20 Lexico.LITINT 0; Lexico.DPONTOS; Lexico.NOVALINHA;
  Lexico.INDENTA;
21 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Negativo";
  Lexico.FPAR;
22 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEDENTA;
  Lexico.PRINT; Lexico.APAR;
```


6 ANALISADOR LÉXICO

```
23 Lexico.LITSTRING "Deseja Finalizar? (S/N) "; Lexico.FPAR;  
    Lexico.NOVALINHA;  
24 Lexico.ID "opc"; Lexico.ATRIB; Lexico.INPUT; Lexico.APAR;  
    Lexico.FPAR;  
25 Lexico.NOVALINHA; Lexico.IF; Lexico.ID "opc";  
    Lexico.IGUALDADE;  
26 Lexico.LITSTRING "S"; Lexico.DPONTOS; Lexico.NOVALINHA;  
    Lexico.INDENTA;  
27 Lexico.ID "programa"; Lexico.ATRIB; Lexico.LITINT 0;  
    Lexico.NOVALINHA;  
28 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID  
    "micro07";  
29 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro08(): -> void :  
2     numero =1  
3     while numero < 0 or numero >0:  
4         print("Digite o numero")  
5         numero = int(input())  
6         if numero > 10:  
7             print("O numero "+str(numero)+" e maior que 10")  
8         else:  
9             print("O numero "+str(numero)+" e menor que 10")  
10  
11  
12 micro08()
```

Saída

Listing 93: Analisador Léxico

```
1 - : tokens =  
2 [Lexico.DEF; Lexico.ID "micro08"; Lexico.APAR; Lexico.FPAR;  
    Lexico.DPONTOS;  
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;  
    Lexico.INDENTA;  
4 Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 1;
```

```
Lexico.NOVALINHA;
5 Lexico.WHILE; Lexico.ID "numero"; Lexico.MENOR;
  Lexico.LITINT 0; Lexico.OU;
6 Lexico.ID "numero"; Lexico.MAIOR; Lexico.LITINT 0;
  Lexico.DPONTOS;
7 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
8 Lexico.LITSTRING "Digite o numero"; Lexico.FPAR;
  Lexico.NOVALINHA;
9 Lexico.ID "numero"; Lexico.ATRIB; Lexico.INT; Lexico.APAR;
  Lexico.INPUT;
10 Lexico.APAR; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA;
  Lexico.IF;
11 Lexico.ID "numero"; Lexico.MAIOR; Lexico.LITINT 10;
  Lexico.DPONTOS;
12 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR;
13 Lexico.LITSTRING "0 numero "; Lexico.MAIS; Lexico.STR;
  Lexico.APAR;
14 Lexico.ID "numero"; Lexico.FPAR; Lexico.MAIS;
15 Lexico.LITSTRING " e maior que 10"; Lexico.FPAR;
  Lexico.NOVALINHA;
16 Lexico.DEDENTA; Lexico.ELSE; Lexico.DPONTOS;
  Lexico.NOVALINHA;
17 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING
  "0 numero ";
18 Lexico.MAIS; Lexico.STR; Lexico.APAR; Lexico.ID "numero";
  Lexico.FPAR;
19 Lexico.MAIS; Lexico.LITSTRING " e menor que 10";
  Lexico.FPAR;
20 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEDENTA;
  Lexico.DEDENTA;
21 Lexico.ID "micro08"; Lexico.APAR; Lexico.FPAR;
  Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro09(): -> void :
2   preco, venda, novopreco = 0.0,0.0,0.0
3
```

```

4 print("Digite o preco: ")
5 preco = int(input())
6 print("Digite a venda: ")
7 venda = int(input())
8 if venda < 500 or preco <30:
9     novopreco = preco + 10/100 *preco
10 elif (venda >= 500 and venda <1200) or (preco >= 30 and
    preco <80):
11     novopreco = preco + 15/100 * preco
12 elif venda >=1200 or preco >=80:
13     novopreco = preco - 20/100 * preco
14
15
16 print("O novo preco e: "+str(novopreco))
17
18 micro09()

```

Saída

Listing 94: Analisador Léxico

```

1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro09"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "preco"; Lexico.VIRG; Lexico.ID "venda";
   Lexico.VIRG;
5 Lexico.ID "novopreco"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.PONTO;
6 Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT 0; Lexico.PONTO;
7 Lexico.LITINT 0; Lexico.VIRG; Lexico.LITINT 0; Lexico.PONTO;
8 Lexico.LITINT 0; Lexico.NOVALINHA; Lexico.PRINT;
   Lexico.APAR;
9 Lexico.LITSTRING "Digite o preco: "; Lexico.FPAR;
   Lexico.NOVALINHA;
10 Lexico.ID "preco"; Lexico.ATRIB; Lexico.INT; Lexico.APAR;
   Lexico.INPUT;
11 Lexico.APAR; Lexico.FPAR; Lexico.FPAR; Lexico.NOVALINHA;

```

```
Lexico.PRINT;
12 Lexico.APAR; Lexico.LITSTRING "Digite a venda: ";
Lexico.FPAR;
13 Lexico.NOVALINHA; Lexico.ID "venda"; Lexico.ATRIB;
Lexico.INT; Lexico.APAR;
14 Lexico.INPUT; Lexico.APAR; Lexico.FPAR; Lexico.FPAR;
Lexico.NOVALINHA;
15 Lexico.IF; Lexico.ID "venda"; Lexico.MENOR; Lexico.LITINT
500; Lexico.OU;
16 Lexico.ID "preco"; Lexico.MENOR; Lexico.LITINT 30;
Lexico.DPONTOS;
17 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "novopreco";
Lexico.ATRIB;
18 Lexico.ID "preco"; Lexico.MAIS; Lexico.LITINT 10;
Lexico.DIVIDIDO;
19 Lexico.LITINT 100; Lexico.VEZES; Lexico.ID "preco";
Lexico.NOVALINHA;
20 Lexico.DEDENTA; Lexico.ELIF; Lexico.APAR; Lexico.ID "venda";
21 Lexico.MAIORIGUAL; Lexico.LITINT 500; Lexico.E; Lexico.ID
"venda";
22 Lexico.MENOR; Lexico.LITINT 1200; Lexico.FPAR; Lexico.OU;
Lexico.APAR;
23 Lexico.ID "preco"; Lexico.MAIORIGUAL; Lexico.LITINT 30;
Lexico.E;
24 Lexico.ID "preco"; Lexico.MENOR; Lexico.LITINT 80;
Lexico.FPAR;
25 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID
"novopreco";
26 Lexico.ATRIB; Lexico.ID "preco"; Lexico.MAIS; Lexico.LITINT
15;
27 Lexico.DIVIDIDO; Lexico.LITINT 100; Lexico.VEZES; Lexico.ID
"preco";
28 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELIF; Lexico.ID
"venda";
29 Lexico.MAIORIGUAL; Lexico.LITINT 1200; Lexico.OU; Lexico.ID
"preco";
30 Lexico.MAIORIGUAL; Lexico.LITINT 80; Lexico.DPONTOS;
Lexico.NOVALINHA;
```

```
31 Lexico.INDENTA; Lexico.ID "novopreco"; Lexico.ATRIB;  
    Lexico.ID "preco";  
32 Lexico.MENOS; Lexico.LITINT 20; Lexico.DIVIDIDO;  
    Lexico.LITINT 100;  
33 Lexico.VEZES; Lexico.ID "preco"; Lexico.NOVALINHA;  
    Lexico.DEDENTA;  
34 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O novo preco  
    e: "; Lexico.MAIS;  
35 Lexico.STR; Lexico.APAR; Lexico.ID "novopreco";  
    Lexico.FPAR; Lexico.FPAR;  
36 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "micro09";  
    Lexico.APAR;  
37 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro10(): -> void :  
2     numero =0  
3     fat = 0  
4     print("Digite um numero: ")  
5     numero = int(input())  
6     fat = fatorial(numero)  
7  
8     print("O fatorial de "+str(numero)+" e "+str(fat))  
9  
10  
11  
12 def fatorial(n:int): -> int :  
13     if n <=0:  
14         return 1  
15     else:  
16         return (n * fatorial(n-1))  
17  
18 micro10()
```

Saída

Listing 95: Analisador Léxico

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro10"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4 Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
5 Lexico.ID "fat"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
6 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um
   numero: ";
7 Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "numero";
   Lexico.ATRIB; Lexico.INT;
8 Lexico.APAR; Lexico.INPUT; Lexico.APAR; Lexico.FPAR;
   Lexico.FPAR;
9 Lexico.NOVALINHA; Lexico.ID "fat"; Lexico.ATRIB; Lexico.ID
   "fatorial";
10 Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
   Lexico.NOVALINHA;
11 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O fatorial de
   "; Lexico.MAIS;
12 Lexico.STR; Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
   Lexico.MAIS;
13 Lexico.LITSTRING " e "; Lexico.MAIS; Lexico.STR;
   Lexico.APAR;
14 Lexico.ID "fat"; Lexico.FPAR; Lexico.FPAR;
   Lexico.NOVALINHA; Lexico.DEDENTA;
15 Lexico.DEF; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID
   "n";
16 Lexico.DPONTOS; Lexico.INT; Lexico.FPAR; Lexico.DPONTOS;
   Lexico.SETA;
17 Lexico.INT; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA; Lexico.IF;
18 Lexico.ID "n"; Lexico.MENORIGUAL; Lexico.LITINT 0;
   Lexico.DPONTOS;
19 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.RETURN;
   Lexico.LITINT 1;
20 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;
```

```
Lexico.DPONTOS;
21 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.RETURN;
Lexico.APAR; Lexico.ID "n";
22 Lexico.VEZES; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID
    "n";
23 Lexico.MENOS; Lexico.LITINT 1; Lexico.FPAR; Lexico.FPAR;
Lexico.NOVALINHA;
24 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID "micro10";
Lexico.APAR;
25 Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
1 def micro11(): -> void :
2     numero,x =0,0
3     print("Digite um numero: ")
4     numero = int(input())
5     x = verifica(numero)
6     if x ==1:
7         print("Numero Positivo")
8     elif x ==0:
9         print("Zero")
10    else:
11        print("Negativo")
12
13 def verifica(n:int): -> int :
14     res = 0
15     if n>0:
16         res = 1
17     elif n<0:
18         res = -1
19     else:
20         res = 0
21
22     return res
23
24 micro11()
```

Saída

Listing 96: Analisador Léxico

```

1  - : tokens =
2  [Lexico.DEF; Lexico.ID "micro10"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3  Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
4  Lexico.ID "numero"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
5  Lexico.ID "fat"; Lexico.ATRIB; Lexico.LITINT 0;
   Lexico.NOVALINHA;
6  Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "Digite um
   numero: ";
7  Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "numero";
   Lexico.ATRIB; Lexico.INT;
8  Lexico.APAR; Lexico.INPUT; Lexico.APAR; Lexico.FPAR;
   Lexico.FPAR;
9  Lexico.NOVALINHA; Lexico.ID "fat"; Lexico.ATRIB; Lexico.ID
   "fatorial";
10 Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
   Lexico.NOVALINHA;
11 Lexico.PRINT; Lexico.APAR; Lexico.LITSTRING "O fatorial de
   "; Lexico.MAIS;
12 Lexico.STR; Lexico.APAR; Lexico.ID "numero"; Lexico.FPAR;
   Lexico.MAIS;
13 Lexico.LITSTRING " e "; Lexico.MAIS; Lexico.STR;
   Lexico.APAR;
14 Lexico.ID "fat"; Lexico.FPAR; Lexico.FPAR;
   Lexico.NOVALINHA; Lexico.DEDENTA;
15 Lexico.DEF; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID
   "n";
16 Lexico.DPONTOS; Lexico.INT; Lexico.FPAR; Lexico.DPONTOS;
   Lexico.SETA;
17 Lexico.INT; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA; Lexico.IF;
18 Lexico.ID "n"; Lexico.MENORIGUAL; Lexico.LITINT 0;

```

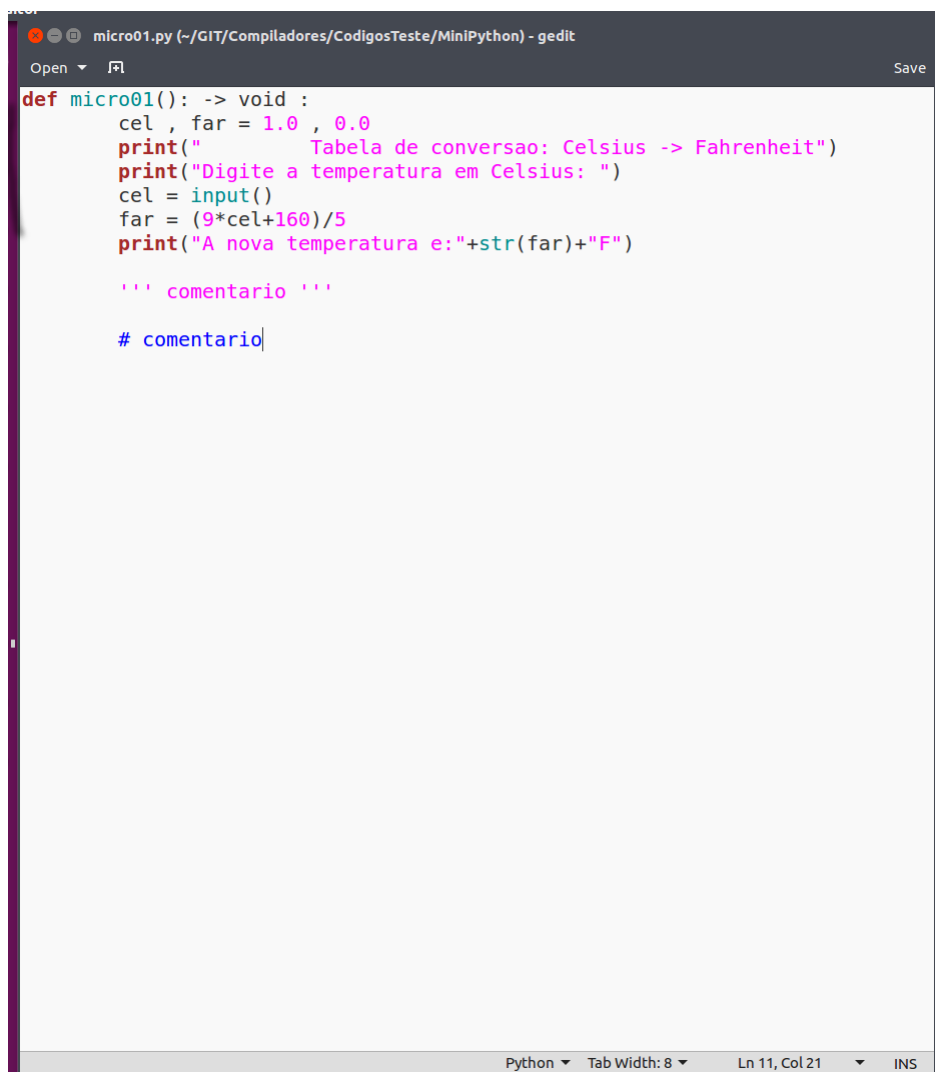


```
19      Lexico.DPONTOS;
      Lexico.NOVALINHA; Lexico.INDENTA; Lexico.RETURN;
      Lexico.LITINT 1;
20      Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE;
      Lexico.DPONTOS;
21      Lexico.NOVALINHA; Lexico.INDENTA; Lexico.RETURN;
      Lexico.APAR; Lexico.ID "n";
22      Lexico.VEZES; Lexico.ID "fatorial"; Lexico.APAR; Lexico.ID
      "n";
23      Lexico.MENOS; Lexico.LITINT 1; Lexico.FPAR; Lexico.FPAR;
      Lexico.NOVALINHA;
24      Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID "micro10";
      Lexico.APAR;
25      Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

6.2.3 Teste de Comentários

Foi usado o código micro01 para testes

6 ANALISADOR LÉXICO



```
def micro01(): -> void :
    cel , far = 1.0 , 0.0
    print("          Tabela de conversao: Celsius -> Fahrenheit")
    print("Digite a temperatura em Celsius: ")
    cel = input()
    far = (9*cel+160)/5
    print("A nova temperatura e:"+str(far)+"F")

    ''' comentario '''

    # comentario
```

A saída não considerou os comentários:

Listing 97: Analisador Léxico com Comentário

```
1 - : tokens =
2 [Lexico.DEF; Lexico.ID "micro01"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
3 Lexico.SETA; Lexico.VOID; Lexico.DPONTOS; Lexico.NOVALINHA;
   Lexico.INDENTA;
```

7 ANALISADOR SINTÁTICO

```
4 Lexico.ID "cel"; Lexico.VIRG; Lexico.ID "far"; Lexico.ATRIB;  
5 Lexico.LITINT 1; Lexico.PONTO; Lexico.LITINT 0; Lexico.VIRG;  
6 Lexico.LITINT 0; Lexico.PONTO; Lexico.LITINT 0;  
  Lexico.NOVALINHA;  
7 Lexico.PRINT; Lexico.APAR;  
8 Lexico.LITSTRING "\t\tTabela de conversao: Celsius ->  
  Fahrenheit";  
9 Lexico.FPAR; Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;  
10 Lexico.LITSTRING "Digite a temperatura em Celsius: ";  
  Lexico.FPAR;  
11 Lexico.NOVALINHA; Lexico.ID "cel"; Lexico.ATRIB;  
  Lexico.INPUT; Lexico.APAR;  
12 Lexico.FPAR; Lexico.NOVALINHA; Lexico.ID "far";  
  Lexico.ATRIB; Lexico.APAR;  
13 Lexico.LITINT 9; Lexico.VEZES; Lexico.ID "cel"; Lexico.MAIS;  
14 Lexico.LITINT 160; Lexico.FPAR; Lexico.DIVIDIDO;  
  Lexico.LITINT 5;  
15 Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR;  
16 Lexico.LITSTRING "A nova temperatura e:"; Lexico.MAIS;  
  Lexico.STR;  
17 Lexico.APAR; Lexico.ID "far"; Lexico.FPAR; Lexico.MAIS;  
18 Lexico.LITSTRING "F"; Lexico.FPAR; Lexico.NOVALINHA;  
  Lexico.NOVALINHA;  
19 Lexico.DEDENTA; Lexico.EOF]
```

7 Analisador Sintático

7.1 Teste de Gramática

Foi implementado, o algoritmo da seguinte gramática em virtude de aprofundar os conhecimentos do processo de análise sintática.

7 ANALISADOR SINTÁTICO

```
S → X Y Z

X → a X b
X →

Y → c Y Z c X
Y → d

Z → e Z Y e
Z → f
```

É importante ressaltar os conceitos de first e follow. First é o conjunto de símbolos que ocorrem no início de uma determinada regra e follow é o que pode aparecer depois da ocorrência de um determinado símbolo.

Facilita o entendimento do código quando é feita a tabela de first e follow e também a tabela com as regras de derivação. Nesse caso, as tabelas foram retiradas dos slides da aula.

Solução

	anulável	FIRST	FOLLOW		a	b	c	d	e	f
S	Não	a c d		S	XYZ		XYZ	XYZ		
X	Sim	a	b c d e f	X	aXb	ε	ε	ε	ε	ε
Y	Não	c d	e f	Y			cYZcX	d		
Z	Não	e f	c d	Z					eZYe	f

Segue a sequencia de passos para compilar e rodar os programas:

Listing 98: Analisador Léxico com Comentário

```
1  ocamllex lexico.mll
2  ocamlc -c sintatico.mli
3  ocamlc -c lexico.ml
4
5  rlwrap ocaml
6  load "lexico.cmo";;
7  "sintaticoArv.ml";;
8  teste();;
```

Seguem os testes referentes a esse processo.

Entrada: abdf

7 ANALISADOR SINTÁTICO

```
# teste();;
Ok!
- : regra = S (X (A, X_vazio, B), Y_d D, Z_f F)
"
```

Entrada: cdfcf

```
Ok!
- : regra = S (X_vazio, Y (C, Y_d D, Z_f F, C, X_vazio), Z_f F)
# teste();;
```

Para uma entrada equivocada, o código gera um erro:

Entrada: bcdac

```
# parser "bcdac";;
Exception: Failure "Erro: esperava a, c ou d mas encontrei b".
```

7.2 Códigos Fonte

Seguem os códigos fonte referentes a implementação do programa:

Listing 99: léxico.mll

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

7 ANALISADOR SINTÁTICO

```
19
20
21   }
22
23   rule token = parse
24   | 'a'      {A}
25   | 'b'      {B}
26   | 'c'      {C}
27   | 'd'      {D}
28   | 'e'      {E}
29   | 'f'      {F}
30   | _ as c { failwith (msg_error lexbuf c) }
31   | eof      { EOF }
```

Listing 100: sintatico.mli

```
1   type tokens = A
2   | B
3   | C
4   | D
5   | E
6   | F
7   | EOF
```

Listing 101: sintaticoArv.ml

```
1   (* Parser preditivo *)
2   load "lexico.cmo";;
3   open Sintatico;;
4
5   type rule = S of rule * rule * rule
6   | X of tokens * rule * tokens
7   | Y of tokens * rule * rule * tokens * rule
8   | Z of tokens * rule * rule * tokens
9   | X_empty
10  | Y_d of tokens
11  | Z_f of tokens
```

7 ANALISADOR SINTÁTICO

```
12
13 let tk = ref EOF (* variavel global para o token atual *)
14 let lexbuf = ref (Lexing.from_string "")
15
16 (* le o proximo token *)
17 let prox () = tk := Lexico.token !lexbuf
18
19 let to_str tk =
20 match tk with
21 A -> "a"
22 | B -> "b"
23 | C -> "c"
24 | D -> "d"
25 | E -> "e"
26 | F -> "f"
27 | EOF -> "eof"
28
29 let erro esp =
30 let msg = Printf.sprintf "Erro: esperava %s mas encontrei
31 %s"
32 in
33 failwith msg
34
35 let consome t = if (!tk == t) then prox() else erro (to_str
36 t)
37
38 let rec ntS () =
39 match !tk with
40 A
41 |C
42 |D ->
43 let cmd1 = ntX() in
44 let cmd2 = ntY() in
45 let cmd3 = ntZ() in
46 S (cmd1, cmd2, cmd3)
47 | _ -> erro "a, c ou d"
48 and ntX () =
```

```
48 match !tk with
49 B
50 |C
51 |D
52 |E
53 |F    -> X_empty
54 |A    -> let _ = consome A in
55 let cmd = ntX() in
56 let _ = consome B in
57 X (A, cmd, B)
58 | _ -> erro "a"
59 and ntY () =
60 match !tk with
61 C    -> let _ = consome C in
62 let cmd = ntY() in
63 let cmd2 = ntZ() in
64 let _ = consome C in
65 let cmd3 = ntX() in
66 Y (C,cmd,cmd2, C, cmd3)
67 |D    -> let _ = consome D in
68 Y_d (D)
69 | _ -> erro "c ou d"
70 and ntZ () =
71 match !tk with
72 E    -> let _ = consome E in
73 let cmd = ntZ() in
74 let cmd2 = ntY() in
75 let _ = consome E in
76 Z (E, cmd, cmd2, E)
77 |F    -> let _ = consome F in
78 Z_f (F)
79 | _ -> erro "e ou f"
80
81 let parser str =
82 lexbuf := Lexing.from_string str;
83 prox (); (* inicializa o token *)
84 let arv = ntS () in
85 match !tk with
```


8 SINTÁTICO USANDO MENHIR

```
86 EOF -> let _ = Printf.printf "Ok!\n" in arv
87 | _ -> erro "fim da entrada"
88
89 let teste str =
90 let entrada = str
91 in
92 parser entrada
```

8 Sintático Usando Menhir

Para usar o Menhir foi necessário instalar o opam (gerenciador de dependências do Ocaml)

```
> sudo apt-get install opam m4
> opam init
> eval 'opam config env'
> opam install menhir
```

Para otimizar o processo de compilação, foi utilizado o arquivo .ocamlinit

```
# directory "_build";;
# load "lexico.cmo";;
# load "parser.cmo";;
# "pre_processador.cmo";;
# "main.cmo";;
# open Main;;
# open Ast;;
```

O processo para a compilação é: **ocamlbuild -use-menhir main.byte**

8.1 Testes

Foram testados os seguintes problemas. Seguem as saídas:

Listing 102: sintatico.mli

```
1 def micro11() -> int:
2     numero = 0
3     x = 0
4     print("numero")
5     numero = input()
```

8 SINTÁTICO USANDO MENHIR

```
6     numero = verifica(numero)
7     if x ==1:
8         print("Positivo")
9     if x ==0:
10        print("zero")
11    else:
12        print("Negativo")
13
14    return 0
15
16
17    def verifica(n:int) -> int:
18        res = 0
19        if n>0:
20            res = 1
21        if n<0:
22            res = 2
23        else:
24            res = 0
25
26        return res
27
28    micro11()
```

Saída

```
- : Ast.prog =
Prog ([],
[DEFUNCAO ([], Some INTEIRO,
[ATRIBUICAO ("numero", ExpTerm (TERMTL (LITINT 0)));
ATRIBUICAO ("x", ExpTerm (TERMTL (LITINT 0))); PRINT ("numero", []);
LEIA "numero";
CHAMADADEFUNCAO ("numero", Some "verifica", [TERMID "numero"]);
CONDICAOIF
(EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (TERMTL (LITINT 1))),
[], None);
CONDICAOIF
(EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (TERMTL (LITINT 0))),
[], Some ELSECOND);
RETORNO (TERMTL (LITINT 0))]);
DEFUNCAO ([INTEIRO], Some INTEIRO,
[ATRIBUICAO ("res", ExpTerm (TERMTL (LITINT 0)));
CONDICAOIF
(EXP (ExpTerm (TERMID "n"), MAIOR, ExpTerm (TERMTL (LITINT 0))),
[], None);
CONDICAOIF
(EXP (ExpTerm (TERMID "n"), MENOR, ExpTerm (TERMTL (LITINT 0))),
[], Some ELSECOND);
RETORNO (TERMID "res"); CHAMADADEFUNCAO ("micro11", None, [])]))]
```

Listing 103: sintatico.mli

```
1  import x
2  def micro():
3      numero = 0
4      numero = input()
5      if numero >= 100:
6          if numero <= 200:
7              print("e")
8          else:
9              print("d")
10     else:
11         print("f")
12
13     micro()
```

Saída

```
- : Ast.prog =
Prog ([ComecoDeBloco],
[DEFFUNCAO ([], None,
[ATRIBUICAO ("numero", ExpTerm (TERMTL (LITINT 0))); LEIA "numero";
CONDICAOIF
(EXP (ExpTerm (TERMID "numero"), MAIORIGUAL,
ExpTerm (TERMTL (LITINT 100))),
[], Some ELSECOND);
CHAMADADEFUNCAO ("micro", None, [])])])])
```

Listing 104: sintatico.mli

```
1 def nano():
2     n=1
3     m=2
4     x=5
5     while x >n:
6         n = 4 + m
7         print("n",n)
```

Saída

```
- : Ast.prog =
Prog ([,
 [DEFFUNCAO ([, None,
 [ATRIBUICAO ("n", ExpTerm (TERMTL (LITINT 1)));
 ATRIBUICAO ("m", ExpTerm (TERMTL (LITINT 2)));
 ATRIBUICAO ("x", ExpTerm (TERMTL (LITINT 5)));
 WHILELOOP (EXP (ExpTerm (TERMID "x"), MAIOR, ExpTerm (TERMID "n")));
 ATRIBUICAO ("n",
 EXP (ExpTerm (TERMTL (LITINT 4)), MAIS, ExpTerm (TERMID "m")));
 PRINT ("n", ["n"])]))]
```

Listing 105: sintatico.mli

```
1 def micro08():
2     numero = 1
3     while numero < 0 or numero > 0:
4         print("digite")
5         numero = input()
6         if numero > 10:
7             print("maior10")
8         else:
9             print("menor10")
10
11
12     micro08()
```

Saída

```
- : Ast.prog =
Prog ([],
[DEFFUNCAO ([], None,
[ATRIBUICAO ("numero", ExpTerm (TERMTL (LITINT 1)));
WHILELOOP
(EXP (ExpTerm (TERMID "numero"), MENOR,
EXP (ExpTerm (TERMTL (LITINT 0))), OU,
EXP (ExpTerm (TERMID "numero"), MAIOR, ExpTerm (TERMTL (LITINT 0))))));
PRINT ("digite", []); LEIA "numero";
CONDICAOIF
(EXP (ExpTerm (TERMID "numero"), MAIOR, ExpTerm (TERMTL (LITINT 10))),
[], Some ELSECOND);
CHAMADADEFUNCAO ("micro08", None, [])])])])
```

Listing 106: sintatico.mli

```
1 def micro06() -> int:
2     numero = 0
3
4     print("Digite um numero de 1 a 5: ")
5     numero = input()
6     if numero == 1:
7         print("Um")
8     elif numero == 2:
9         print("Dois")
10    elif numero == 3:
11        print("Tres")
12    elif numero == 4:
```

9 ANALISADOR SEMÂNTICO

```
13     print("Quatro")
14 elif numero ==5:
15     print("Cinco")
16 else:
17     print("Numero Invalido!!!")
18
19 micro06()
```

Saída

```
- : Ast.prog =
Prog ([,
[DEFFUNCAO ([, Some INTEIRO,
[ATRIBUICAO ("numero", ExpTerm (TERMTL (LITINT 0)));
PRINT ("Digite um numero de 1 a 5: ", []); LEIA "numero";
CONDICAOIF
(EXP (ExpTerm (TERMID "numero"), IGUALDADE, ExpTerm (TERMTL (LITINT 1))),
[EXP (ExpTerm (TERMID "numero"), IGUALDADE, ExpTerm (TERMTL (LITINT 2)));
EXP (ExpTerm (TERMID "numero"), IGUALDADE, ExpTerm (TERMTL (LITINT 3)));
EXP (ExpTerm (TERMID "numero"), IGUALDADE, ExpTerm (TERMTL (LITINT 4)));
EXP (ExpTerm (TERMID "numero"), IGUALDADE, ExpTerm (TERMTL (LITINT 5))],
Some ELSECOND);
CHAMADADEFUNCAO ("micro06", None, [])])])
```

9 Analisador Semântico

Em virtude de incorporar o analisador semântico ao trabalho já feito, foram realizadas mudanças nos arquivos do sintático e léxico a fim de facilitar a integração.

Agora, as variáveis do código estão sendo exibidas com os parâmetros linha e coluna. Segue um exemplo:

Listing 107: sintatico.mli

```
1
2 def funcaozona() -> int:
3
4     inputf(valor)
5
6     x = valor + 1.0
7
8     return 1
```

Saída

```

EOF
- : Sast.expressao Ast.programa option option =
Some
  (Some
    (Programa
      [Funcao
        {fn_nome =
          ("funcaozona",
            {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
          fn_tiporet = INTEIRO; fn_formais = []};
        fn_corpo =
          [LEIAF
            (Sast.EXPVAR
              ("valor",
                {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                  pos_cnum = 7}));
            ATRIBUICAO
              (Sast.EXPVAR
                ("x",
                  {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                    pos_cnum = 8}),
                Sast.EXPOPB
                  ((ADICAO,
                    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                      pos_cnum = 10}),
                    Sast.EXPVAR
                      ("valor",
                        {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                          pos_cnum = 4}),
                    Sast.EXPFLOAT
                      (1.,
                        {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                          pos_cnum = 12})))));
            RETORNO
              (Some
                (Sast.EXPINT
                  (1,
                    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
                      pos_cnum = 7}))))]]]))

```

9.1 Testes

Seguem os testes que tem como saída a árvores semântica tipada. Para aumentar o controle de tipos, o comando input foi alterado em tres outros comandos, inputi, inputs e inputf que representam leitura de tipos inteiro, string e real.

Listing 108: sintatico.mli

```

1
2 def micro11() -> int:
3   numero = 0
4   x = 0
5   print("numero")
6   inputs(numero)
7   numero = verifica(numero)

```

```
8   if x ==1:
9       print("Positivo")
10  if x ==0:
11      print("zero")
12  else:
13      print("Negativo")
14
15  return 0
16
17
18 def verifica(n:int) -> int:
19     res = 0
20     if n>0:
21         res = 1
22     if n<0:
23         res = 2
24     else:
25         res = 0
26
27     return res
28
29 x = 2
```

Saída


```

- : Tact.expressao Ast.programa * Ambiente.t =
(Programa
 [Funcao
  {fn_name =
   ("micro11",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = INTEIRO; fn_formais = [];
   fn_corpo =
    [ATRIBUICAO (Tact.EXPVAR ("numero", INTEIRO), Tact.EXPINT (0, INTEIRO));
     ATRIBUICAO (Tact.EXPVAR ("x", INTEIRO), Tact.EXPINT (0, INTEIRO));
     PRINT (Tact.EXPSTRING ("numero", STRING));
     LEIAS (Tact.EXPVAR ("numero", INTEIRO));
     ATRIBUICAO (Tact.EXPVAR ("numero", INTEIRO),
      Tact.EXPCALL ("verifica", [Tact.EXPVAR ("numero", INTEIRO)], INTEIRO));
     CONDICAIOIF
      (Tact.EXPOPB ((EHIGUAL, BOOLEAN),
       (Tact.EXPVAR ("x", INTEIRO), INTEIRO),
       (Tact.EXPINT (1, INTEIRO), INTEIRO)),
      [PRINT (Tact.EXPSTRING ("Positivo", STRING)], None);
     CONDICAIOIF
      (Tact.EXPOPB ((EHIGUAL, BOOLEAN),
       (Tact.EXPVAR ("x", INTEIRO), INTEIRO),
       (Tact.EXPINT (0, INTEIRO), INTEIRO)),
      [PRINT (Tact.EXPSTRING ("zero", STRING)],
       Some (CONDICAIOIfElse [PRINT (Tact.EXPSTRING ("Negativo", STRING)]))];
     RETORNO (Some (Tact.EXPINT (0, INTEIRO)))]];
  Funcao
   {fn_name =
    ("verifica",
     {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
    fn_tiporet = INTEIRO;
    fn_formais =
     [((("n",
      {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 13}),
      INTEIRO))];
    fn_corpo =
     [ATRIBUICAO (Tact.EXPVAR ("res", INTEIRO), Tact.EXPINT (0, INTEIRO));
      CONDICAIOIF
       (Tact.EXPOPB ((MAIORQ, BOOLEAN),
        (Tact.EXPVAR ("n", INTEIRO), INTEIRO),
        (Tact.EXPINT (0, INTEIRO), INTEIRO)),
       [ATRIBUICAO (Tact.EXPVAR ("res", INTEIRO), Tact.EXPINT (1, INTEIRO)],
        None);
      CONDICAIOIF
       (Tact.EXPOPB ((MENORQ, BOOLEAN),
        (Tact.EXPVAR ("n", INTEIRO), INTEIRO),
        (Tact.EXPINT (0, INTEIRO), INTEIRO)),
       [ATRIBUICAO (Tact.EXPVAR ("res", INTEIRO), Tact.EXPINT (2, INTEIRO)],
        Some
         (CONDICAIOIfElse
          [ATRIBUICAO (Tact.EXPVAR ("res", INTEIRO),
           Tact.EXPINT (0, INTEIRO))]);
      RETORNO (Some (Tact.EXPVAR ("res", INTEIRO)))]],
    <abstr>

```

Listing 109: sintatico.mli

```

1 def micro() -> None:
2     numero = 0
3     inputi()
4     if numero >= 100:
5         if numero <= 200:
6             print("e")
7         else:
8             print("d")

```

```

9     else:
10         print("f")
11
12 micro()

```

Saída

```

- : Tact.expressao Ast.programa * Ambiente.t =
(Programa
 [Funcao
  {fn_nome =
   ("micro",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = NONE; fn_formals = []};
  fn_corpo =
  [LEIAI (Tact.EXPVAR ("numero", INTEIRO));
   CONDICA0IF
    (Tact.EXPOPB ((MAIORIGUALQ, BOOLEAN),
      (Tact.EXPVAR ("numero", INTEIRO), INTEIRO),
      (Tact.EXPINT (100, INTEIRO), INTEIRO)),
     [CONDICA0IF
      (Tact.EXPOPB ((MENORIGUALQ, BOOLEAN),
        (Tact.EXPVAR ("numero", INTEIRO), INTEIRO),
        (Tact.EXPINT (200, INTEIRO), INTEIRO)),
       [PRINT (Tact.EXPSTRING ("e", STRING))],
        Some (CONDICA0Else [PRINT (Tact.EXPSTRING ("d", STRING))]])),
       Some (CONDICA0Else [PRINT (Tact.EXPSTRING ("f", STRING))]]))],
   <abstr>)]

```

Listing 110: sintatico.mli

```

1
2 def nano() -> None:
3     n=1
4     m=2
5     x=5
6     while x >n:
7         n = 4 + m
8         print("n",n)

```

Saída

9 ANALISADOR SEMÂNTICO

```
- : Tact.expressao Ast.programa * Ambiente.t =
(Programa
 [Funcao
  {fn_nome =
   ("nano",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = NONE; fn_formais = []};
  fn_corpo =
  [ATRIBUICAO (Tact.EXPVAR ("n", INTEIRO), Tact.EXPINT (1, INTEIRO));
   ATRIBUICAO (Tact.EXPVAR ("m", INTEIRO), Tact.EXPINT (2, INTEIRO));
   ATRIBUICAO (Tact.EXPVAR ("x", INTEIRO), Tact.EXPINT (5, INTEIRO));
   WHILELOOP
    (Tact.EXPOPB ((MAIORQ, BOOLEAN),
      (Tact.EXPVAR ("x", INTEIRO), INTEIRO),
      (Tact.EXPVAR ("n", INTEIRO), INTEIRO)),
     [ATRIBUICAO (Tact.EXPVAR ("n", INTEIRO),
      Tact.EXPOPB ((ADICAO, INTEIRO),
        (Tact.EXPINT (4, INTEIRO), INTEIRO),
        (Tact.EXPVAR ("m", INTEIRO), INTEIRO))));
      PRINT (Tact.EXPSTRING ("n", STRING))]]]);
  <abstr>)
```

Listing 111: sintatico.mli

```
1
2 def micro08() -> None:
3     numero =1
4     while numero < 0 or numero >0:
5         print("digite")
6         inputi(numero)
7         if numero > 10:
8             print("maior10")
9         else:
10            print("menor10")
11
12
13 micro08()
```

Saída

9 ANALISADOR SEMÂNTICO

```
- : Tact.expressao Ast.programa * Ambiente.t =
(Programa
 [Funcao
  {fn_nome =
   ("micro08",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = NONE; fn_formais = [];
   fn_corpo =
    [ATRIBUICAO (Tact.EXPVAR ("numero", INTEIRO), Tact.EXPINT (1, INTEIRO));
     WHILELOOP
      (Tact.EXPOPB ((OR, BOOLEAN),
        (Tact.EXPOPB ((MENORQ, BOOLEAN),
          (Tact.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tact.EXPINT (0, INTEIRO), INTEIRO)),
          BOOLEAN),
        (Tact.EXPOPB ((MAIORQ, BOOLEAN),
          (Tact.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tact.EXPINT (0, INTEIRO), INTEIRO)),
          BOOLEAN))),
      [PRINT (Tact.EXPSTRING ("digite", STRING));
       LEIAI (Tact.EXPVAR ("numero", INTEIRO));
       CONDICAOIF
        (Tact.EXPOPB ((MAIORQ, BOOLEAN),
          (Tact.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tact.EXPINT (10, INTEIRO), INTEIRO)),
          [PRINT (Tact.EXPSTRING ("maior10", STRING)]),
          Some
           (CONDICAOelifElse [PRINT (Tact.EXPSTRING ("menor10", STRING)]]))]]],
    <abstr>)
```

Listing 112: sintatico.mli

```
1
2 def micro06() -> int:
3     numero = 0
4
5     print("Digite um numero de 1 a 5: ")
6     inputi(numero)
7     if numero ==1:
8         print("Um")
9     elif numero == 2:
10        print("Dois")
11    elif numero ==3:
12        print("Tres")
13    elif numero ==4:
14        print("Quatro")
15    elif numero ==5:
16        print("Cinco")
17    else:
18        print("Numero Invalido!!!")
19
20 micro06()
```

Saída

```

EOF
- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
[Funcao
{fn_nome =
("micro06",
{Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_crum = 4});
fn_tipolet = INTEIRO; fn_formais = [];
fn_corpo =
[ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO), Tast.EXPINT (0, INTEIRO));
PRINT (Tast.EXPSTRING ("Digite um numero de 1 a 5:", STRING));
LEIAI (Tast.EXPVAR ("numero", INTEIRO));
CONDICAOIF
(Tast.EXPOPB ((EHIGUAL, BOOLEAN),
(Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
(Tast.EXPINT (1, INTEIRO), INTEIRO)),
[PRINT (Tast.EXPSTRING ("Um", STRING)]),
Some
(CONDICAIOF
(Tast.EXPOPB ((EHIGUAL, BOOLEAN),
(Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
(Tast.EXPINT (2, INTEIRO), INTEIRO)),
[PRINT (Tast.EXPSTRING ("Dois", STRING)]),
Some
(CONDICAIOF
(Tast.EXPOPB ((EHIGUAL, BOOLEAN),
(Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
(Tast.EXPINT (3, INTEIRO), INTEIRO)),
[PRINT (Tast.EXPSTRING ("Tres", STRING)]),
Some
(CONDICAIOF
(Tast.EXPOPB ((EHIGUAL, BOOLEAN),
(Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
(Tast.EXPINT (4, INTEIRO), INTEIRO)),
[PRINT (Tast.EXPSTRING ("Quatro", STRING)]),
Some
(CONDICAIOF
(Tast.EXPOPB ((EHIGUAL, BOOLEAN),
(Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
(Tast.EXPINT (5, INTEIRO), INTEIRO)),
[PRINT (Tast.EXPSTRING ("Cinco", STRING)]),
Some
(CONDICAIOFelifElse
[PRINT (Tast.EXPSTRING ("Numero Invalido!!!", STRING))]]))))))]]],
<abstr>

```

Listing 113: sintatico.mli

```

1
2 def micro10() -> None:
3     numero = 0
4     fat = 0
5     print("Digite um numero: ")
6     inputi(numero)
7     fat = fatorial(numero)
8
9     print("O fatorial eh ",fat)
10
11 def fatorial(n: int) -> int:

```

```

12     if n <=0:
13         return 1
14     else:
15         return n * fatorial(n - 1)
16
17 micro10( )

```

Saída

```

- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
 [Funcao
  {fn_name =
   ("micro10",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = NONE; fn_formais = []};
  fn_corpo =
  [ATRIBUICAO (Tast.EXPPVAR ("numero", INTEIRO), Tast.EXPPINT (0, INTEIRO));
   ATRIBUICAO (Tast.EXPPVAR ("fat", INTEIRO), Tast.EXPPINT (0, INTEIRO));
   PRINT (Tast.EXPPSTRING ("Digite um numero: ", STRING));
   LEIAI (Tast.EXPPVAR ("numero", INTEIRO));
   ATRIBUICAO (Tast.EXPPVAR ("fat", INTEIRO),
    Tast.EXPCALL ("fatorial", [Tast.EXPPVAR ("numero", INTEIRO)], INTEIRO));
   PRINT (Tast.EXPPSTRING ("O fatorial eh ", STRING))];
  Funcao
  {fn_name =
   ("fatorial",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 4});
   fn_tiporet = INTEIRO;
   fn_formais =
   [((("n",
    {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0; pos_cnum = 13}),
    INTEIRO));
   fn_corpo =
   [CONDICAOIF
    (Tast.EXPOPBB ((MENORIGUALQ, BOOLEAN),
    (Tast.EXPPVAR ("n", INTEIRO), INTEIRO),
    (Tast.EXPPINT (0, INTEIRO), INTEIRO)),
    [RETORNO (Some (Tast.EXPPINT (1, INTEIRO)))],
    Some
    (CONDICAOElifElse
    [RETORNO
    (Some
    (Tast.EXPOPBB ((MULTIPLICACAO, INTEIRO),
    (Tast.EXPPVAR ("n", INTEIRO), INTEIRO),
    (Tast.EXPCALL ("fatorial",
    [Tast.EXPOPBB ((SUBTRACAO, INTEIRO),
    (Tast.EXPPVAR ("n", INTEIRO), INTEIRO),
    (Tast.EXPPINT (1, INTEIRO), INTEIRO))),
    INTEIRO),
    INTEIRO)))]))]

```

9.2 Testes Com Erros

Seguem os testes em que o código foi deixado propositalmente com erros semânticos.

Listing 114: sintatico.mli

1

9 ANALISADOR SEMÂNTICO

```
2 def func() -> int:
3
4     inputf(valor)
5
6     x = valor + 1.0
7
8     return "uma string que nao deveria estar aqui"
```

Saída

```
Exception:
Failure
"Semantico -> linha 1, coluna 44: O tipo retornado eh string mas foi declarado como inteiro".
```

Listing 115: sintatico.mli

```
1
2 def func() -> int:
3
4     inputf(valor)
5
6     x = valor + 1.0
7
8     return "uma string que nao deveria estar aqui"
```

Saída

```
EOF
Exception:
Failure
"Semantico -> linha 1, coluna 9: Operando esquerdo do tipo real, mas o tipo do direito eh string".
```

10 Interpretador Usando Menhir

10.1 Execução

Para executar o interpretador deve-se digitar

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

E depois entrar no ocaml usando **rlwrap ocaml**. Após isso, o interpretador pode ser executado com **# interprete "../testes/nomeDoArquivo.py"**.

OBS.: deve-se, antes disso, excluir o diretorio build com **rm -rf _build** e excluir o arquivo interpretadorTeste.byte caso ele existe com **rm interpretadorTeste.byte**.

10.2 Testes

O interpretador consiste em executar o código usando as partes léxica, sintática e semântica feitas durante o semestre. Seguem os testes

Listing 116: sintatico.mli

```
1 def main() -> int:
2     numero = 0
3     x = 0
4     print("Digite um numero: ")
5     inputi(numero)
6     numero = verifica(numero)
7
8     if numero == 1:
9         print("Positivo")
10    elif numero == 0:
11        print("zero")
12    else:
13        print("Negativo")
14
15    return 0
16
17
18 def verifica(n:int) -> int:
19     res = 0
20     if n > 0:
21         return 1
22     if n < 0:
23         return 3
24     else:
25         return 0
26
27 main()
```

Saída

```
[ interprete "../testes/e1.py";;  
EOF  
Digite um numero: 1  
Positivo- : unit = ()  
# interprete "../testes/e1.py";;  
EOF  
Digite um numero: -1  
Negativo- : unit = ()  
# interprete "../testes/e1.py";;  
EOF  
Digite um numero: 0  
zero- : unit = ()  
"
```

Listing 117: sintatico.mli

```
1 def main() -> None:  
2     print("Digite um numero: ")  
3     inputi(numero)  
4     if numero >= 100:  
5         if numero <= 200:  
6             print("\n numero entre 100 e 200")  
7         else:  
8             print("\n numero maior que 200")  
9     else:  
10        print("\n numero menor que 100")  
11  
12  
13 main()
```

Saída

```
[ interprete "../testes/e2.py";;  
EOF  
[Digite um numero: 1  
  
numero menor que 100- : unit = ()  
[# interprete "../testes/e2.py";;  
EOF  
[Digite um numero: 101  
  
numero entre 100 e 200- : unit = ()  
[# interprete "../testes/e2.py";;  
EOF  
[Digite um numero: 201  
  
numero maior que 200- : unit = ()
```

Listing 118: sintatico.mli

```
1 def main() -> None:  
2     n=1  
3     m=2  
4     x=50  
5     while n < x:  
6         n = n + 4 + m  
7         print("n = ")  
8         print(n)  
9         print("\n")  
10  
11 main()
```

Saída

```
[ interprete "../testes/e3.py";;  
EOF  
n = 7  
n = 13  
n = 19  
n = 25  
n = 31  
n = 37  
n = 43  
n = 49  
n = 55  
- : unit = ()
```

Listing 119: sintatico.mli

```
1 def main() -> None:  
2     numero =1  
3     while numero < 0 or numero >0:  
4         print("\n Digite um numero: ")  
5         inputi(numero)  
6         if numero > 10:  
7             print("\n Numero maior que 10")  
8         else:  
9             print("\n Numero menor que 10")  
10  
11  
12 main()
```

Saída

```
[ Digite um numero: 5

Numero menor que 10
[ Digite um numero: 11

Numero maior que 10
[ Digite um numero: -1

Numero menor que 10
[ Digite um numero: 0

Numero menor que 10- : unit = ()
```

Listing 120: sintatico.mli

```
1 def main() -> int:
2     numero = 0
3
4     print("Digite um numero de 1 a 5: ")
5     inputi(numero)
6     if numero == 1:
7         print("Um")
8     elif numero == 2:
9         print("Dois")
10    elif numero == 3:
11        print("Tres")
12    elif numero == 4:
13        print("Quatro")
14    elif numero == 5:
15        print("Cinco")
16    else:
17        print("Numero Invalido!!!")
18
19 main()
```

Saída

```
[ interprete "../testes/e6.py";;  
EOF  
Digite um numero de 1 a 5: 1  
Um- : unit = ()  
# interprete "../testes/e6.py";;  
EOF  
Digite um numero de 1 a 5: 3  
Tres- : unit = ()  
# interprete "../testes/e6.py";;  
EOF  
Digite um numero de 1 a 5: 5  
Cinco- : unit = ()  
# interprete "../testes/e6.py";;  
EOF  
Digite um numero de 1 a 5: -1  
Numero Invalido!!!- : unit = ()
```

Listing 121: sintatico.mli

```
1 def main() -> None:  
2     numero =0  
3     fat = 0  
4     print("Digite um numero: ")  
5     inputi(numero)  
6     fat = fatorial(numero)  
7  
8     print("O fatorial eh ")  
9     print(fat)  
10  
11 def fatorial(n: int) -> int:  
12     if n <= 0:  
13         return 1  
14     else:  
15         return n * fatorial(n - 1)  
16  
17 main()
```

Saída

```
interprete "../testes/e7.py";;  
EOF  
Digite um numero: 10  
0 fatorial eh 3628800- : unit = ()  
# interprete "../testes/e7.py";;  
EOF  
Digite um numero: 7  
0 fatorial eh 5040- : unit = ()  
# interprete "../testes/e7.py";;  
EOF  
Digite um numero: 1  
0 fatorial eh 1- : unit = ()
```

Listing 122: sintatico.mli

```
1 def main() -> None:  
2     print("Digite um numero: ")  
3     inputi(i)  
4     while i < 10:  
5         print(i)  
6         print(" ")  
7         i = i +1
```

Saída

```
interprete "../testes/ex1.py";;  
EOF  
Digite um numero: 5  
5 6 7 8 9 - : unit = ()  
# interprete "../testes/ex1.py";;  
EOF  
Digite um numero: 100  
- : unit = ()  
# interprete "../testes/ex1.py";;  
EOF  
Digite um numero: 10  
- : unit = ()  
# interprete "../testes/ex1.py";;  
EOF  
Digite um numero: 1  
1 2 3 4 5 6 7 8 9 - : unit = ()
```

Listing 123: sintatico.mli

```
1
2 def fib(n:int) -> int:
3     if n <= 1:
4         return 1
5     else:
6         return fib( n - 1 ) + fib( n - 2 )
7
8 def fat(n:int) -> int:
9     if n <= 1:
10        return 1
11    else:
12        return n * fat( n - 1 )
13
14
15 def main() -> None:
16     op = 1
17     while op != 0:
18         print("| 1 - fib\n| 2 - fat\n| 0 - sair\n-> ")
19         inputi(op)
20         if op == 1:
21             print("Digite um numero para calcular o fibonacci: ")
22             inputi(f)
23             print(" fibonacci :")
24             print(fib(f))
25         elif op == 2:
26             print("Digite um numero para calcular o fatorial: ")
27             inputi(f)
28             print(" fatorial :")
29             print(fat(f))
30         else:
31             op =0
32             print("\n")
33
34
35 main()
```


Saída

```
interprete "../testes/ex4.py";;
EOF
| 1 - fib
| 2 - fat
| 0 - sair
-> 1
Digite um numero para calcular o fibonacci: 5
fibonacci :8
| 1 - fib
| 2 - fat
| 0 - sair
-> 2
Digite um numero para calcular o fatorial: 11
fatorial :39916800
| 1 - fib
| 2 - fat
| 0 - sair
-> 0

- : unit = ()
#
```

11 Erros gerados pelo interpretador

Nesta seção, serão exibidos os erros gerados pelo interpretador dado como entrada códigos em python propositalmente errados.

Erro Semântico

Listing 124: sintatico.mli

```
1 def main() -> str:
2     n=1
3     m=2
4     x=50
5     while n < x:
6         n = n + 4 + m
7         print("n = ")
8         print(n)
9         print("\n")
```

11 ERROS GERADOS PELO INTERPRETADOR

```
10
11     return 1
12
13 main()
```

Saída

```
- : unit = ()
[# interprete "../testes/e3.py";;
EOF
Exception:
Failure
"Semantico -> linha 1, coluna 6: 0 tipo retornado eh inteiro mas foi declarado como string".
#
```

Erro Sintático

Listing 125: sintatico.mli

```
1
2 def main() -> None:
3     n=1
4     m=2
5     x=50
6     while n < x:
7         n = n + 4 + m
8         print("n = "
9         print(n)
10        print("\n")
11
12 main()
```

Saída

```
interprete "../testes/e3.py";;
Erro sintático na linha 1, coluna -1 0 - <YOUR SYNTAX ERROR MESSAGE HERE>
.
Exception: Failure "Nada a fazer!\n".
#
```

Erro Léxico

Listing 126: sintatico.mli

```
1 def main() -> None:
2     n=1
3     @=2
4     x=50
5     while n < x:
6         n = n + 4 + m
7         print("n = ")
8         print(n)
9         print("\n")
10
11 main()
```

Saída

```
interprete "../testes/e3.py";;
Exception: Failure "1-0: caracter desconhecido @".
```

Apêndice

lexico.mll

Listing 127: lexico.mll

```
1 {
2
3     open Sintatico
4     open Lexing
5     open Printf
6
7     exception Erro of string
8
9     let booleano nbool =
10     match nbool with
```

```

11 | "True" -> 1
12 | "False" -> 0
13 | _ -> failwith "Erro: nao eh valor booleano"
14
15 let nivel_par = ref 0
16
17 let incr_num_linha lexbuf =
18 let pos = lexbuf.lex_curr_p in
19 lexbuf.lex_curr_p <- { pos with
20 pos_lnum = pos.pos_lnum + 1;
21 pos_bol = pos.pos_cnum;
22 }
23
24 let msg_erro lexbuf c =
25 let pos = lexbuf.lex_curr_p in
26 let lin = pos.pos_lnum
27 and col = pos.pos_cnum - pos.pos_bol - 1 in
28 sprintf "%d-%d: caracter desconhecido %c" lin col c
29
30 let erro lin col msg =
31 let mensagem = sprintf "%d-%d: %s" lin col msg in
32 failwith mensagem
33
34 let pos_atual lexbuf = lexbuf.lex_start_p
35
36 }
37
38 let digito = ['0' - '9']
39 let int = '-'? * digito+
40 let float = '-'? * digito+ * '.' * digito+
41 let comentario = "#"[ ^ '\n' ]*
42 let linha_em_branco = [' ' '\t']* comentario
43 let restante = [^ ' '\t' '\n' ] [^ '\n']+
44 let brancos = [' ' '\t']+
45 let novalinha = '\r' | '\n' | "\r\n"
46 let letra = ['a'-'z' 'A' - 'Z']
47 let identificador = letra ( letra | digito | '_' )*
48

```

```

49  (* 0 pre processador necessario para contabilizar a
      indentacao *)
50  rule preprocessador indentacao = parse
51  linha_em_branco      { preprocessador 0 lexbuf } (* ignora
      brancos *)
52  | [' ' '\t']+ '\n'    { incr_num_linha lexbuf;
53  preprocessador 0 lexbuf } (* ignora brancos *)
54  | ' '                  { preprocessador (indentacao + 1)
      lexbuf }
55  | '\t'                  { let nova_ind = indentacao + 8 -
      (indentacao mod 8)
56  in preprocessador nova_ind lexbuf }
57  | novalinha            { incr_num_linha lexbuf;
58  preprocessador 0 lexbuf }
59  | restante as linha {
60  let rec tokenize lexbuf =
61  let tok = token lexbuf in
62  match tok with
63  EOF -> []
64  | _ -> tok :: tokenize lexbuf in
65  let toks = tokenize (Lexing.from_string linha) in
66  Linha(indentacao,!nivel_par, toks)
67  }
68  | eof { nivel_par := 0; EOF }
69
70  (* 0 analisador lexico a ser chamado apos o pre processador
      *)
71  and token = parse
72  brancos              { token lexbuf }
73  | comentario         { token lexbuf }
74  | """                { comentario_bloco 0 lexbuf; }
75  | ">="                { MAIORIGUAL (pos_atual lexbuf)}
76  | "<="                { MENORIGUAL (pos_atual lexbuf)}
77  | "->"               { SETA (pos_atual lexbuf)}
78  | "=="               { IGUALDADE (pos_atual lexbuf)}
79  | "!="               { DIFERENTE (pos_atual lexbuf)}
80  | '('                { incr(nivel_par); APAR(pos_atual
      lexbuf)}}

```

```

81 | '),'          { decr(nivel_par); FPAR(pos_atual
    lexbuf)}}
82 | ','          { VIRG (pos_atual lexbuf)}
83 | '+'          { MAIS (pos_atual lexbuf)}
84 | '-'          { MENOS (pos_atual lexbuf) }
85 | '*'          { VEZES (pos_atual lexbuf)}
86 | '/'          { DIVIDIDO (pos_atual lexbuf)}
87 | '='          { ATRIB (pos_atual lexbuf)}
88 | ':'          { DPONTOS (pos_atual lexbuf)}
89 | '<'          { MENOR (pos_atual lexbuf)}
90 | '>'          { MAIOR (pos_atual lexbuf)}
91 | '%'          { MODULO (pos_atual lexbuf)}
92 | "or"         { OU (pos_atual lexbuf)}
93 | "if"         { IF (pos_atual lexbuf)}
94 | "else"       { ELSE (pos_atual lexbuf)}
95 | "while"      { WHILE (pos_atual lexbuf)}
96 | "for"        { FOR (pos_atual lexbuf)}
97 | "return"     { RETURN (pos_atual lexbuf)}
98 | "def"        { DEF (pos_atual lexbuf)}
99 | "int"        { INT (pos_atual lexbuf)}
100 | "float"      { FLOAT (pos_atual lexbuf)}
101 | "bool"       { BOOL (pos_atual lexbuf)}
102 | "and"        { E (pos_atual lexbuf) }
103 | "in"         { IN (pos_atual lexbuf)}
104 | "range"      { RANGE (pos_atual lexbuf)}
105 | "None"       { NONE (pos_atual lexbuf)}
106 | "elif"       { ELIF (pos_atual lexbuf)}
107 | "print"      { PRINT (pos_atual lexbuf)}
108 | "str"        { STR (pos_atual lexbuf)}
109 | "inputi"     { INPUTI (pos_atual lexbuf)}
110 | "inputf"     { INPUTF (pos_atual lexbuf)}
111 | "inputs"     { INPUTS (pos_atual lexbuf)}
112 | "not"        { NOT (pos_atual lexbuf)}
113 | "True"       { LITBOOL(true,pos_atual lexbuf)}
114 | "False"      { LITBOOL(false,pos_atual lexbuf)}
115 | int as num    { LITINT (int_of_string num, pos_atual
    lexbuf) }
116 | float as num { LITFLOAT (float_of_string num,

```

```

117     pos_atual lexbuf) }
| digito+ as numint {let num = int_of_string numint in
    LITINT (num, pos_atual lexbuf)}
118 | identificador as id { ID (id, pos_atual lexbuf) }
119 | '''          { let pos = lexbuf.lex_curr_p in
120 let lin = pos.pos_lnum
121 and col = pos.pos_cnum - pos.pos_bol - 1 in
122 let buffer = Buffer.create 1 in
123 let str = leia_string lin col buffer lexbuf in
124 LITSTRING (str, pos_atual lexbuf) }
125 | _ as c { failwith (msg_erro lexbuf c) }
126 | eof      { EOF }
127
128 and comentario_bloco n = parse
129 '''      { if n=0 then token lexbuf
130 else comentario_bloco (n-1) lexbuf }
131 | '''      { comentario_bloco (n+1) lexbuf }
132 | novalinha { incr_num_linha lexbuf; comentario_bloco n
    lexbuf }
133 | _        { comentario_bloco n lexbuf }
134 | eof      { raise (Erro "Comentario nao terminado") }
135
136 and leia_string lin col buffer = parse
137 '''      { Buffer.contents buffer}
138 | "\\t"    { Buffer.add_char buffer '\t'; leia_string lin
    col buffer lexbuf }
139 | "\\n"    { Buffer.add_char buffer '\n'; leia_string lin
    col buffer lexbuf }
140 | '\\\'    { Buffer.add_char buffer '\''; leia_string lin
    col buffer lexbuf }
141 | '\\\' '\\\' { Buffer.add_char buffer '\\\''; leia_string lin
    col buffer lexbuf }
142 | _ as c    { Buffer.add_char buffer c; leia_string lin col
    buffer lexbuf }
143 | eof      { erro lin col "A string nao foi fechada"}

```

semantico.mll

Listing 128: semantico.mll

```
1
2  module Amb = Ambiente
3  module A = Ast
4  module S = Sast
5  module T = Tast
6
7  let rec posicao exp =
8  let open S in
9  match exp with
10 | EXPVAR      (_,pos)      -> pos
11 | EXPINT      (_,pos)      -> pos
12 | EXPSTRING   (_,pos)      -> pos
13 | EXPBOOL     (_,pos)      -> pos
14 | EXPFLOAT    (_,pos)      -> pos
15 | EXPOPB      ((_,pos),_,_) -> pos
16 | EXPOPU      ((_,pos),_)   -> pos
17 | EXPCALL     ((_,pos),_)   -> pos
18
19
20 type classe_op = Aritmetico | Relacional | Logico
21
22 let classifica op =
23 let open A in
24 match op with
25 ADICAO
26 | SUBTRACAO
27 | MULTIPLICACAO
28 | DIVISAO
29 | MOD        -> Aritmetico
30 | MAIORQ
31 | MENORQ
32 | MAIORIGUALQ
33 | MENORIGUALQ
34 | EHIGUAL
```

```

35 | EHDIFERENTE -> Relacional
36 | AND
37 | NEGACAO
38 | OR -> Logico
39
40 let msg_erro_pos pos msg =
41 let open Lexing in
42 let lin = pos.pos_lnum
43 and col = pos.pos_cnum - pos.pos_bol - 1 in
44 Printf.sprintf "Semantico -> linha %d, coluna %d: %s" lin
    col msg
45
46 (* argumento nome e do tipo S.tipo *)
47 let msg_erro nome msg =
48 let pos = snd nome in
49 msg_erro_pos pos msg
50
51 let nome_tipo t =
52 let open A in
53 match t with
54 INTEIRO    -> "inteiro"
55 | STRING   -> "string"
56 | BOOLEAN  -> "booleano"
57 | REAL     -> "real"
58 | NONE     -> "vazio"
59
60 let mesmo_tipo pos msg tinf tdec =
61 if tinf <> tdec then
62 let msg = Printf.sprintf msg (nome_tipo tinf) (nome_tipo
    tdec) in
63 failwith (msg_erro_pos pos msg)
64
65 let rec infere_exp amb exp =
66 match exp with
67
68 | S.EXPINT i -> (T.EXPINT (fst i, A.INTEIRO ), A.INTEIRO )
69 | S.EXPSTRING s -> (T.EXPSTRING (fst s, A.STRING ),
    A.STRING )

```

```

70 | S.EXPBOOL b -> (T.EXPBOOL (fst b, A.BOOLEAN ), A.BOOLEAN )
71 | S.EXPFLOAT f -> (T.EXPFLOAT (fst f, A.REAL), A.REAL)
72 | S.EXPVAR variavel ->
73 let nome = fst variavel in
74 (try begin
75 (match (Amb.busca amb nome) with
76 | Amb.EntVar tipo -> (T.EXPVAR (nome, tipo), tipo)
77 | Amb.EntFun _ ->
78 let msg = "Nome de funcao usado como nome de variavel:
    ^nome in
79 failwith (msg_erro variavel msg))
80 end with Not_found ->
81 let msg = "Variavel "^nome^" nao declarada" in
82 failwith (msg_erro variavel msg))
83 | S.EXPOPB (op, exp_esq, exp_dir) ->
84 let (esq, tesq) = infere_exp amb exp_esq
85 and (dir, tdir) = infere_exp amb exp_dir in
86 let verifica_aritmetico () =
87 (match tesq with
88 | A.INTEIRO
89 | A.REAL ->
90 let _ = mesmo_tipo (snd op)
91 "Operando esquerdo do tipo %s, mas o tipo do direito eh %s"
92 tesq tdir
93 in tesq (* Tipo inferido para a operacao *)
94 | demais ->
95 let msg = "O tipo "^
96 (nome_tipo demais)^
97 " nao eh valido em um operador aritmetico" in
98 failwith (msg_erro op msg))
99 and verifica_relacional () =
100 (match tesq with
101 | A.INTEIRO
102 | A.STRING
103 | A.BOOLEAN
104 | A.REAL ->
105 (let _ = mesmo_tipo (snd op)
106 "Operando esquerdo do tipo %s, mas o tipo do direito eh %s"

```

```

107 tesq tdir
108 in A.BOOLEAN) (* Tipo inferido para a operacao *)
109 | demais ->
110 (let msg = "O tipo "^
111 (nome_tipo demais)^
112 " nao eh valido em um operador relacional" in
113 failwith (msg_erro op msg)))
114 and verifica_logico () =
115 (match tesq with
116 | A.BOOLEAN ->
117 let _ = mesmo_tipo (snd op)
118 "Operando esquerdo do tipo %s, mas o tipo do direito eh %s"
119 tesq tdir
120 in A.BOOLEAN (* Tipo inferido para a operacao *)
121 | demais ->
122 let msg = "O tipo "^
123 (nome_tipo demais)^
124 " nao eh valido em um operador logico" in
125 failwith (msg_erro op msg))
126 in
127 let oper = fst op in
128 let tinf =
129 (match (classifica oper) with
130 | Aritmetico -> verifica_aritmetico ()
131 | Relacional -> verifica_relacional ()
132 | Logico -> verifica_logico () )
133 in (T.EXPOPB ((oper, tinf), (esq, tesq), (dir, tdir)), tinf)
134 | S.EXPOPU (op, exp) ->
135 let (exp, texp) = infere_exp amb exp in
136 let verifica_not () =
137 match texp with
138 | A.BOOLEAN ->
139 let _ = mesmo_tipo (snd op)
140 "O operando eh do tipo %s, mas espera-se um %s"
141 texp A.BOOLEAN
142 in A.BOOLEAN
143 | demais ->
144 let msg = "O tipo "^

```

```

145     (nome_tipo demais)^
146     " nINTEIROao eh valido para o operador not" in
147     failwith (msg_erro op msg)
148     and verifica_negativo () =
149     match texp with
150     | A.REAL ->
151     let _ = mesmo_tipo (snd op)
152     "O operando eh do tipo %s, mas espera-se um %s"
153     texp A.REAL
154     in A.REAL
155     | A.INTEIRO ->
156     let _ = mesmo_tipo (snd op)
157     "O operando eh do tipo %s, mas espera-se um %s"
158     texp A.INTEIRO
159     in A.INTEIRO
160     | demais ->
161     let msg = "O tipo "^
162     (nome_tipo demais)^
163     " nao eh valido para o operador menos" in
164     failwith (msg_erro op msg)
165     in
166     let oper = fst op in
167     let tinf =
168     let open A in
169     match oper with
170     | NEGACAO -> verifica_not ()
171     | SUBTRACAO -> verifica_negativo ()
172     | demais->
173     let msg = "Operador unario indefinido"
174     in failwith (msg_erro op msg)
175     in (T.EXPOPU ((oper, tinf), (exp, texp)), tinf)
176     | S.EXPCALL (nome, args) ->
177     let rec verifica_parametros ags ps fs =
178     match (ags, ps, fs) with
179     | (a::ags), (p::ps), (f::fs) ->
180     let _ = mesmo_tipo (posicao a)
181     "O parametro eh do tipo %s mas deveria ser do tipo %s"
182     p f

```

```

183   in verifica_parametros ags ps fs
184   | [], [], [] -> ()
185   | _ -> failwith (msg_erro nome "Numero incorreto de
      parametros")
186   in
187   let id = fst nome in
188   try
189   begin
190   let open Amb in
191   match (Amb.busca amb id) with
192   | Amb.EntFun {tipo_fn; formais} ->
193   let targs = List.map (infere_exp amb) args
194   and tformais = List.map snd formais in
195   let _ = verifica_parametros args (List.map snd targs)
      tformais in
196   (T.EXPCALL (id, (List.map fst targs), tipo_fn), tipo_fn)
197   | Amb.EntVar _ -> (* Se estiver associada a uma variavel,
      falhe *)
198   let msg = id ^ " eh uma variavel e nao uma funcao" in
199   failwith (msg_erro nome msg)
200   end
201   with Not_found ->
202   let msg = "Nao existe a funcao de nome " ^ id in
203   failwith (msg_erro nome msg)
204
205   let rec verifica_cmd amb tiporet cmd =
206   let open A in
207   match cmd with
208   | CHAMADADEFUNCAO exp -> let (exp,tinf) = infere_exp amb
      exp in CHAMADADEFUNCAO exp
209   | PRINT exp -> let expt = infere_exp amb exp in PRINT (fst
      expt)
210   | WHILELOOP (cond, cmds) ->
211   let (expCond, expT ) = infere_exp amb cond in
212   let comandos_tipados =
213   (match expT with
214   | A.BOOLEAN -> List.map (verifica_cmd amb tiporet) cmds
215   | _ -> let msg = "Condicao deve ser tipo Bool" in

```

```

216 failwith (msg_erro_pos (posicao cond) msg))
217 in WHILELOOP (expCond,comandos_tipados)
218 | LEIAI exp ->
219 (match exp with
220 S.EXPVAR (id,pos) ->
221 (try
222 begin
223 (match (Amb.busca amb id) with
224 Amb.EntVar tipo ->
225 let expt = infere_exp amb exp in
226 let _ = mesmo_tipo pos
227 "inputi com tipos diferentes: %s = %s"
228 tipo (snd expt) in
229 LEIAI (fst expt)
230 | Amb.EntFun _ ->
231 let msg = "nome de funcao usado como nome de variavel: " ^
232 id in
233 failwith (msg_erro_pos pos msg) )
234 end
235 with Not_found ->
236 let _ = Amb.insere_local amb id A.INTEIRO in
237 let expt = infere_exp amb exp in
238 LEIAI (fst expt) )
239 | _ -> failwith "Falha Inputi"
240 )
241 | LEIAF exp ->
242 (match exp with
243 S.EXPVAR (id,pos) ->
244 (try
245 begin
246 (match (Amb.busca amb id) with
247 Amb.EntVar tipo ->
248 let expt = infere_exp amb exp in
249 let _ = mesmo_tipo pos
250 "Inputf com tipos diferentes: %s = %s"
251 tipo (snd expt) in
252 LEIAF (fst expt)
253 | Amb.EntFun _ ->

```

```

253   let msg = "nome de funcao usado como nome de variavel: " ^
        id in
254   failwith (msg_erro_pos pos msg) )
255   end
256   with Not_found ->
257   let _ = Amb.insere_local amb id A.REAL in
258   let expt = infere_exp amb exp in
259   LEIAF (fst expt) )
260   | _ -> failwith "Falha Inputf"
261   )
262   | LEIAS exp ->
263   (match exp with
264   S.EXPVAR (id,pos) ->
265   (try
266   begin
267   (match (Amb.busca amb id) with
268   Amb.EntVar tipo ->
269   let expt = infere_exp amb exp in
270   let _ = mesmo_tipo pos
271   "Inputs com tipos diferentes: %s = %s"
272   tipo (snd expt) in
273   LEIAS (fst expt)
274   | Amb.EntFun _ ->
275   let msg = "nome de funcao usado como nome de variavel: " ^
        id in
276   failwith (msg_erro_pos pos msg) )
277   end
278   with Not_found ->
279   let _ = Amb.insere_local amb id A.STRING in
280   let expt = infere_exp amb exp in
281   LEIAS (fst expt) )
282   | _ -> failwith "Falha Inputs"
283   )
284   | ATRIBUICAO (elem, exp) ->
285   let (var1, tdir) = infere_exp amb exp in
286   ( match elem with
287   S.EXPVAR (id,pos) ->
288   (try

```

```

289 begin
290 (match (Amb.busca amb id) with
291   Amb.EntVar tipo ->
292     let _ = mesmo_tipo pos
293     "Atribuicao com tipos diferentes: %s = %s"
294     tipo tdir in
295     ATRIBUICAO (T.EXPVAR (id, tipo), var1)
296   | Amb.EntFun _ ->
297     let msg = "nome de funcao usado como nome de variavel: " ^
298               id in
299     failwith (msg_erro_pos pos msg) )
300 end
301 with Not_found ->
302   let _ = Amb.insere_local amb id tdir in
303   ATRIBUICAO (T.EXPVAR (id, tdir), var1))
304   | _ -> failwith "Falha CmdAtrib"
305 )
306 | RETORNO exp ->
307   (match exp with
308     (* Se a funcao nao retornar nada, verifica se ela foi
309       declarada como void *)
310     None ->
311       let _ = mesmo_tipo (Lexing.dummy_pos)
312       "0 tipo retornado eh %s mas foi declarado como %s"
313
314     NONE tiporet
315     in RETORNO None
316     | Some e ->
317
318       (* Verifica se o tipo inferido para a expressao de retorno
319         confere com o *)
320       (* tipo declarado para a funcao.
321         *)
322       let (e1,tinf) = infere_exp amb e in
323       let _ = mesmo_tipo (posicao e)
324       "0 tipo retornado eh %s mas foi declarado como %s"
325       tinf tiporet
326       in RETORNO (Some e1)

```

```

323 )
324 | CONDICAOfElse comandos ->
325 let comandos = List.map (verifica_cmd amb tiporet) comandos
326   in
327 CONDICAOfElse comandos
328 | CONDICAOf (teste, entao, senao) ->
329 let (teste1,tinf) = infere_exp amb teste in
330 let _ = mesmo_tipo (posicao teste)
331 "O teste do if deveria ser do tipo %s e nao %s"
332 BOOLEAN tinf in
333 let entao1 = List.map (verifica_cmd amb tiporet) entao in
334 let senao1 =
335 match senao with
336 | None      -> None
337 | Some bloco -> let c = verifica_cmd amb tiporet bloco in
338   Some c
339 in CONDICAOf (teste1, entao1, senao1)
340 | FORLOOP (idt, int_de,int_ate,bloco) ->
341 let (idt1,tinf) = infere_exp amb idt in
342 let (int_de1,tinf1) = infere_exp amb int_de in
343 let (int_ate1,tinf2) = infere_exp amb int_ate in
344 (* O tipo inferido para o identificador deve ser int *)
345 let _ = mesmo_tipo (posicao idt)
346 "A variavel deveria ser do tipo %s e nao %s"
347 INTEIRO tinf in
348 (* O tipo inferido para os ints devem ser inteiros *)
349 let _ = mesmo_tipo (posicao int_de)
350 "O comando DE deveria ser do tipo %s e nao %s"
351 INTEIRO tinf1 in
352 let _ = mesmo_tipo (posicao int_ate)
353 "O comando DE deveria ser do tipo %s e nao %s"
354 INTEIRO tinf2 in
355 (* Verifica a validade de cada comando do bloco *)
356 let bloco1 = List.map (verifica_cmd amb tiporet) bloco in
357 FORLOOP (idt1, int_de1,int_ate1,bloco1)
358
359 and verifica_fun amb ast =
360 let open A in

```

```

359 match ast with
360 | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
361 (* Estende o ambiente global, adicionando um ambiente local
   *)
362 let ambfn = Amb.novo_escopo amb in
363 (* Insere os parametros no novo ambiente *)
364 let insere_parametro (v,t) = Amb.insere_param ambfn (fst v)
   t in
365 let _ = List.iter insere_parametro fn_formais in
366 (* Verifica cada comando presente no corpo da funcao usando
   o novo ambiente *)
367 let corpo_tipado = List.map (verifica_cmd ambfn fn_tiporet)
   fn_corpo in
368 Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo =
   corpo_tipado}
369 | ACMD _ -> failwith "Instrucao invalida"
370
371 let rec verifica_dup xs =
372 match xs with
373 | [] -> []
374 | (nome,t)::xs ->
375 let id = fst nome in
376 if (List.for_all (fun (n,t) -> (fst n) <> id) xs)
377 then (id, t) :: verifica_dup xs
378 else let msg = "Parametro duplicado " ^ id in
379 failwith (msg_erro nome msg)
380
381 let insere_declaracao_fun amb dec =
382 let open A in
383 match dec with
384 | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
385 let formais = verifica_dup fn_formais in
386 let nome = fst fn_nome in
387 Amb.insere_fun amb nome formais fn_tiporet
388 | ACMD _ -> failwith "Instrucao invalida"
389
390 let fn_predefs =
391 let open A in [

```

```

392  ("inputi", [("x", INTEIRO )], NONE);
393  ("inputs", [("x", STRING )], NONE);
394  ("inputf", [("x", REAL)], NONE)]
395
396  let declara_predefinidas amb =
397  List.iter (fun (n,ps,tr) -> Amb.insere_fun amb n ps tr)
          fn_predefs
398
399  let semantico ast =
400  let amb_global = Amb.novo_amb [] in
401  let _ = declara_predefinidas amb_global in
402  let A.Programa instr = ast in
403  let decs_funs = List.filter(fun x ->
404  (match x with
405  | A.Funcao _ -> true
406  | _ -> false)) instr in
407  let _ = List.iter (insere_declaracao_fun amb_global)
          decs_funs in
408  let decs_funs = List.map (verifica_fun amb_global)
          decs_funs in
409  (A.Programa decs_funs, amb_global)

```

interprete.mll

Listing 129: interprete.mll

```
1  module Amb = AmbInterp
2  module A = Ast
3  module S = Sast
4  module T = Tast
5
6  exception Valor_de_retorno of T.expressao
7
8  let obtem_nome_tipo_var exp = let open T in
9  match exp with
10 | EXPVAR (nome,tipo) -> (nome,tipo)
11 | _ -> failwith "obtem_nome_tipo_var1: nao
    eh variavel"
12
13 let pega_int exp =
14 match exp with
15 | T.EXPINT (i,_) -> i
16 | _ -> failwith "pega_int: nao eh inteiro"
17
18 let pega_float exp = match exp with
19 | T.EXPFLOAT (f,_) -> f
20 | _ -> failwith "pega_float: nao eh inteiro"
21
22 let pega_str exp =
23 match exp with
24 | T.EXPSTRING (s,_) -> s
25 | _ -> failwith "pega_string: nao eh string"
26
27 let pega_bool exp =
28 match exp with
29 | T.EXPBOOL (b,_) -> b
30 | _ -> failwith "pega_bool: nao eh booleano"
31
32 type classe_op = Aritmetico | Relacional | Logico
33
```

```

34   let classifica op =
35   let open A in
36   match op with
37   OR
38   | NEGACAO
39   | AND -> Logico
40   | MENORQ
41   | MAIORQ
42   | MAIORIGUALQ
43   | MENORIGUALQ
44   | EHIGUAL
45   | EHDIFERENTE -> Relacional
46   | ADICAO
47   | SUBTRACAO
48   | MULTIPLICACAO
49   | MOD
50   | DIVISAO -> Aritmetico
51
52
53   let rec interpreta_exp amb exp =
54   let open A in
55   let open T in
56   match exp with
57   | EXPFLOAT _
58   | EXPINT _
59   | EXPSTRING _
60   | EXPBOOL _ -> exp
61   | EXPVAR (nome, tipo) ->
62   (match (Amb.busca amb nome) with
63   | Amb.EntVar (_, v) ->
64   (match v with
65   | Some valor -> valor
66   | None -> failwith "variavel nao inicializada: ")
67   )
68   | _ -> failwith "interpreta_exp: expvar"
69   )
70   | EXPOPB ((op,top), (esq, tesq), (dir,tdir)) ->
71   let vesq = interpreta_exp amb esq

```

```

72  and vdir = interpreta_exp amb dir in
73
74  let interpreta_aritmetico () =
75  (
76  match tesq with
77  | INTEIRO ->
78  (
79  match op with
80  | ADICAO -> EXPINT (pega_int vesq + pega_int vdir, top)
81  | SUBTRACAO -> EXPINT (pega_int vesq - pega_int vdir, top)
82  | MULTIPLICACAO -> EXPINT (pega_int vesq * pega_int vdir,
83                             top)
84  | DIVISAO -> EXPINT (pega_int vesq / pega_int vdir, top)
85  | _ -> failwith "interpreta_aritmetico"
86  )
87  | _ -> failwith "interpreta_aritmetico"
88  )
89  and interpreta_relacional () =
90  (match tesq with
91  | INTEIRO ->
92  (match op with
93  | MAIORIGUALQ -> EXPBOOL (pega_int vesq >= pega_int vdir,
94                             top)
95  | MENORIGUALQ -> EXPBOOL (pega_int vesq <= pega_int vdir,
96                             top)
97  | MENORQ -> EXPBOOL (pega_int vesq < pega_int vdir, top)
98  | MAIORQ -> EXPBOOL (pega_int vesq > pega_int vdir, top)
99  | EHIGUAL -> EXPBOOL (pega_int vesq == pega_int vdir,
100                        top)
101  | EHDIFERENTE -> EXPBOOL (pega_int vesq != pega_int
102                             vdir, top)
103  | _ -> failwith "interpreta_relacional"
104  )
105  | STRING ->
106  (match op with
107  | MAIORIGUALQ -> EXPBOOL (pega_str vesq >= pega_str vdir,
108                             top)
109  | MENORIGUALQ -> EXPBOOL (pega_str vesq <= pega_str vdir,

```

```

    top)
104 | MENORQ      -> EXPBOOL (pega_str vesq < pega_str vdir, top)
105 | MAIORQ      -> EXPBOOL (pega_str vesq > pega_str vdir, top)
106 | EHIGUAL     -> EXPBOOL (pega_str vesq == pega_str vdir,
    top)
107 | EHDIFERENTE -> EXPBOOL (pega_str vesq != pega_str
    vdir, top)
108 | _          -> failwith "interpreta_relacional"
109 )
110 | BOOLEAN ->
111 (match op with
112 | MAIORIGUALQ -> EXPBOOL (pega_bool vesq >= pega_bool vdir,
    top)
113 | MENORIGUALQ -> EXPBOOL (pega_bool vesq <= pega_bool vdir,
    top)
114 | MENORQ      -> EXPBOOL (pega_bool vesq < pega_bool vdir,
    top)
115 | MAIORQ      -> EXPBOOL (pega_bool vesq > pega_bool vdir,
    top)
116 | EHIGUAL     -> EXPBOOL (pega_bool vesq == pega_bool vdir,
    top)
117 | EHDIFERENTE -> EXPBOOL (pega_bool vesq != pega_bool
    vdir, top)
118 | _          -> failwith "interpreta_relacional"
119 )
120 | REAL ->
121 (match op with
122 | MAIORIGUALQ -> EXPBOOL (pega_float vesq == pega_float
    vdir, top)
123 | MENORIGUALQ -> EXPBOOL (pega_float vesq == pega_float
    vdir, top)
124 | MENORQ      -> EXPBOOL (pega_float vesq < pega_float vdir,
    top)
125 | MAIORQ      -> EXPBOOL (pega_float vesq > pega_float vdir,
    top)
126 | EHIGUAL     -> EXPBOOL (pega_float vesq == pega_float
    vdir, top)
127 | EHDIFERENTE -> EXPBOOL (pega_float vesq != pega_float

```

```

128     | _      -> failwith "interpreta_relacional"
129   )
130   | _ -> failwith "interpreta_relacional"
131   )
132
133   and interpreta_logico () =
134   (match tesq with
135   | BOOLEAN ->
136   (match op with
137   | OR -> EXPBOOL (pega_bool vesq || pega_bool vdir, top)
138   | AND -> EXPBOOL (pega_bool vesq && pega_bool vdir, top)
139   | _ -> failwith "interpreta_logico"
140   )
141   | _ -> failwith "interpreta_logico"
142   )
143
144   in
145   let valor = (match (classifica op) with
146   Aritmetico -> interpreta_aritmetico ()
147   | Relacional -> interpreta_relacional ()
148   | Logico      -> interpreta_logico ()
149   )
150   in
151   valor
152
153   | EXPOPU ((op, top), (exp, texp)) ->
154   let vexp = interpreta_exp amb exp in
155   let interpreta_not () =
156   (match texp with
157   | A.BOOLEAN -> EXPBOOL (not (pega_bool vexp), top)
158   | _ -> failwith "Operador unario indefinido")
159   and interpreta_negativo () =
160   (match texp with
161   | A.INTEIRO -> EXPINT (-1 * pega_int vexp, top)
162   | A.REAL -> EXPFLOAT (-1.0 *. pega_float vexp, top)
163   | _ -> failwith "Operador unario indefinido")
164   in

```

```

165 let valor =
166 (match op with
167 | NEGACAO -> interpreta_not ()
168 | SUBTRACAO -> interpreta_negativo ()
169 | _ -> failwith "Operador unario indefinido")
170 in valor
171 | EXPCALL (id, args, tipo) ->
172 let open Amb in
173 (match (Amb.busca amb id) with
174 | Amb.EntFun {tipo_fn; formais; corpo} ->
175 let vargs = List.map (interpreta_exp amb) args in
176 let vformais = List.map2 (fun (n,t) v -> (n, t, Some v))
177     formais vargs
178 in interpreta_fun amb vformais corpo
179 | _ -> failwith "interpreta_exp: expchamada"
180 )
181 | EXPNONE -> T.EXPNONE
182
183 and interpreta_cmd amb cmd =
184 let open A in
185 let open T in
186 match cmd with
187 RETORNO exp ->
188 (* Levantar uma excecao foi necessaria pois, pela semantica
189    do comando de *)
190 (* retorno, sempre que ele for encontrado em uma funcao, a
191    computacao *)
192 (* deve parar retornando o valor indicado, sem realizar os
193    demais comandos. *)
194 (match exp with
195 (* Se a funcao nao retornar nada, entao retorne ExpVoid *)
196 | None -> raise (Valor_de_retorno EXPNONE)
197 | Some e ->
198 (* Avalia a expressao e retorne o resultado *)
199 let e1 = interpreta_exp amb e in
200 raise (Valor_de_retorno e1))
201 | CONDICAOF (teste, entao, senao) ->
202 let teste1 = interpreta_exp amb teste in

```

```

199 (match teste1 with
200 | EXPBOOL (true,_) ->
201 (* Interpreta cada comando do bloco 'entao' *)
202 List.iter (interpreta_cmd amb) entao
203 | _ ->
204 (* Interpreta cada comando do bloco 'senao' se houver *)
205 (match senao with
206 | None -> ()
207 | Some bloco -> interpreta_cmd amb bloco))
208 | CONDICAOelifElse comandos ->
209 List.iter (interpreta_cmd amb ) comandos
210 | ATRIBUICAO (elem, exp) ->
211 let resp = interpreta_exp amb exp in
212 (match elem with
213 | T.EXPVAR (id, tipo) ->
214 (try
215 begin
216 match (Amb.busca amb id) with
217 | Amb.EntVar (t, _) -> Amb.atualiza_var amb id tipo (Some
218 resp)
219 | Amb.EntFun _ -> failwith "falha na atribuicao"
220 end
221 with Not_found ->
222 let _ = Amb.insere_local amb id tipo None in
223 Amb.atualiza_var amb id tipo (Some resp))
224 | _ -> failwith "Falha CmdAtrib"
225 )
226 | CHAMADADEFUNCAO exp -> ignore( interpreta_exp amb exp )
227 | LEIAI exp
228 | LEIAF exp
229 | LEIAS exp ->
230 (* Obtem os nomes e os tipos de cada um dos argumentos *)
231 let nt = obtem_nome_tipo_var exp in
232 let leia_var (nome, tipo) =
233 let _ =
234 (try
235 begin
236 match (Amb.busca amb nome) with

```

```

236 | Amb.EntVar (_,_) -> ()
237 | Amb.EntFun _ -> failwith "falha no input"
238 end
239 with Not_found ->
240 let _ = Amb.insere_local amb nome tipo None in ()
241 )
242 in
243 let valor =
244 (match tipo with
245 | INTEIRO -> T.EXPINT (read_int () , tipo)
246 | STRING -> T.EXPSTRING (read_line () , tipo)
247 | REAL -> T.EXPFLOAT (read_float () , tipo)
248 | _ -> failwith "Fail input")
249 in Amb.atualiza_var amb nome tipo (Some valor)
250 in leia_var nt
251 | PRINT exp ->
252 let resp = interpreta_exp amb exp in
253 (match resp with
254 | T.EXPINT (n,_) -> print_int n
255 | T.EXPFLOAT (n,_) -> print_float n
256 | T.EXPSTRING (n,_) -> print_string n
257 | T.EXPBOOL (b,_) ->
258 let _ = print_string (if b then "true" else "false")
259 in print_string " "
260 | _ -> failwith "Fail print"
261 )
262 | WHILELOOP (cond, cmds) ->
263 let rec laco cond cmds =
264 let condResp = interpreta_exp amb cond in
265 (match condResp with
266 | EXPBOOL (true,_) ->
267 (* Interpreta cada comando do bloco 'entao' *)
268 let _ = List.iter (interpreta_cmd amb) cmds in
269 laco cond cmds
270 | _ -> ())
271 in laco cond cmds
272 | FORLOOP (idt, int_de ,int_ate, bloco) ->
273 let (elem1,tipo) = obtem_nome_tipo_var idt in

```

```

274 let rec executa_para amb int_de int_ate bloco elem1 tipo =
275 if (int_de) <= (int_ate)
276 then begin
277 (*Executa o bloco de codigo: *)
278 List.iter (interpreta_cmd amb) bloco;
279 (*Atualiza o valor da variavel: *)
280 Amb.atualiza_var amb elem1 tipo (Some ( EXPINT( (int_de + 1
281 ),INTEIRO) ) );
282 (*Chamada recursiva:*)
283 executa_para amb (int_de + 1) int_ate bloco elem1 tipo;
284 end in
285 executa_para amb (pega_int int_de) (pega_int int_ate) bloco
286 elem1 tipo
287
288 and interpreta_fun amb fn_formais fn_corpo =
289 let open A in
290 (* Estende o ambiente global, adicionando um ambiente local
291 *)
292 let ambfn = Amb.novo_escopo amb in
293 (* Associa os argumento
294 s aos parametros e insere no novo ambiente *)
295 let insere_parametro (n,t,v) = Amb.insere_param ambfn n t v
296 in
297 let _ = List.iter insere_parametro fn_formais in
298 (* Interpreta cada comando presente no corpo da funcao
299 usando o novo *)
300 (* ambiente
301 *)
302 try
303 let _ = List.iter (interpreta_cmd ambfn) fn_corpo in
304 T.EXPNONE
305 with
306 Valor_de_retorno expret -> expret
307
308 let insere_declaracao_fun amb dec =
309 let open A in
310 match dec with
311 | Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->

```

```

305 let nome = fst fn_nome in
306 let formais = List.map (fun (n,t) -> ((fst n), t))
    fn_formais in
307 Amb.insere_fun amb nome formais fn_tiporet fn_corpo
308 | _ -> failwith "Erro de declaacao de funcao"
309
310
311 let fn_predefs = let open A in [
312   ("inputi", [("x", INTEIRO )], NONE, []);
313   ("inputf", [("x", REAL   )], NONE, []);
314   ("inputs", [("x", STRING )], NONE, []);
315
316 ]
317
318 (* insere as funcoes pre definidas no ambiente global *)
319 let declara_predefinidas amb =
320 List.iter (fun (n,ps,tr,c) -> Amb.insere_fun amb n ps tr c)
    fn_predefs
321
322 let interprete ast =
323 let open Amb in
324 let amb_global = Amb.novo_amb [] in
325 let _ = declara_predefinidas amb_global in
326 let A.Programa instr = ast in
327 let decs_funs = List.filter (fun x ->
328 (match x with
329 | A.Funcao _ -> true
330 | _ -> false)) instr in
331 let _ = List.iter (insere_declaracao_fun amb_global)
    decs_funs in
332 (try begin
333 (match (Amb.busca amb_global "main") with
334 | Amb.EntFun { tipo_fn ; formais ; corpo } ->
335 let vformais = List.map (fun (n,t) -> (n, t, None)) formais
    in
336 let _ = interpreta_fun amb_global vformais corpo in ()
337 | _ -> failwith "variavel declarada como 'main'")
338 end with Not_found -> failwith "Funcao main nao declarada ")

```



sintatico.mly

Listing 130: sintatico.mly

```
1
2     %{
3     open Ast
4     open Sast
5     %}
6
7     %token <int * int * token list> Linha
8     %token <float * Lexing.position> LITFLOAT
9     %token <string * Lexing.position> ID
10    %token <string * Lexing.position> LITSTRING
11    %token <int * Lexing.position> LITINT
12    %token <bool * Lexing.position> LITBOOL
13    %token <Lexing.position> DEF SETA DPONTOS
14    %token <Lexing.position> VIRG
15    %token <Lexing.position> ATRIB MAIOR MAIORIGUAL MENOR
16    %token <Lexing.position> OU E NOT MAIS MENOS DIVIDIDO
17    %token <Lexing.position> APAR FPAR
18    %token <Lexing.position> PRINT
19    %token <Lexing.position> INPUTI INPUTF INPUTS
20    %token <Lexing.position> WHILE FOR IN RANGE
21    %token <Lexing.position> IF ELIF ELSE
22    %token <Lexing.position> RETURN
23    %token <Lexing.position> NONE
24    %token <Lexing.position> STR
25    %token <Lexing.position> INT
26    %token <Lexing.position> FLOAT
27    %token <Lexing.position> BOOL
28    %token INDENTA DEDENTA NOVALINHA EOF
29
30    %left OU
31    %left E
32    %left IGUALDADE DIFERENTE
```

```

33      %left MAIOR MAIORIGUAL MENOR MENORIGUAL
34      %left MAIS MENOS
35      %left VEZES DIVIDIDO MODULO
36
37      %nonassoc unary_minus
38
39      %start <Sast.expressao Ast.programa> programa
40
41      %%
42
43      programa: ins=instrucao*
44      EOF
45      {Programa ins }
46
47      funcao:
48      DEF nome= ID
49      APAR args = separated_list(VIRG, parametro) FPAR
50      SETA retorno = tipo DPONTOS NOVALINHA
51      INDENTA
52      cmd = comandos
53      DEDENTA
54      {
55      Funcao {
56      fn_nome = nome;
57      fn_tiporet = retorno;
58      fn_formais = args;
59      fn_corpo = cmd
60      }
61      }
62
63
64      parametro:
65      | id = ID DPONTOS tp = tipo { (id,tp) }
66
67
68      /*esse eh o meu stm_block */
69      instrucao:
70      | func = funcao      {      func      }

```

```

71 | cmd = comando    { ACMD(cmd) }
72
73
74 | comandos:
75 | cmd = comando+ { cmd }
76
77 | /*esse eh o meu stm_list*/
78 | comando:
79 | | stm = atribuicao          { stm }
80 | | stm = chamadafuncao      { stm }
81 | | stm = loopWhile         { stm }
82 | | stm = condicaoIF         { stm }
83 | | stm = loopFOR           { stm }
84 | | stm = print              { stm }
85 | | stm = retorno           { stm }
86 | | stm = leiai NOVALINHA    { stm }
87 | | stm = leiaf NOVALINHA    { stm }
88 | | stm = leias NOVALINHA    { stm }
89 | ;
90
91 | retorno:
92 | | RETURN expr = exprLogicoAritmetica? NOVALINHA {
93 | |     RETORNO(expr) }
94 | ;
95
96 | print:
97 | | PRINT exprla = exprLogicoAritmetica NOVALINHA
98 | |     {PRINT(exprla) }
99 | ;
100 | /*a sacada eh emcapsular tudo dentro de expressao*/
101
102 | chamadafuncao:
103 | | exp=chamada NOVALINHA { CHAMADADEFUNCAO(exp) }
104 | ;
105
106 | chamada : nome=ID APAR args=separated_list(VIRG,
107 |         exprLogicoAritmetica) FPAR { EXPCALL (nome, args) }

```

```

106 condicaoIF:
107 | IF exprla= exprLogicoAritmetica DPONTOS NOVALINHA
108 INDENTA stm=comandos DEDENTA
109 cee = condicaoELIFELSE?
110 { CONDICAIOIF(exprla,stm,cee) }
111
112
113 condicaoELIFELSE:
114 | ELIF exprla = exprLogicoAritmetica DPONTOS NOVALINHA
115     INDENTA stm = comandos DEDENTA condee =
116     condicaoELIFELSE? { CONDICAIOIF (exprla,stm, condee) }
117 | ELSE DPONTOS NOVALINHA INDENTA stm=comandos DEDENTA
118     {CONDICAOelifElse( stm ) }
119 ;
120
121 atribuicao: id = ID ATRIB exprla = exprLogicoAritmetica
122     NOVALINHA { ATRIBUICAO (EXPVAR id , exprla) }
123
124 leiai: INPUTI exp=exprLogicoAritmetica { LEIAI exp }
125 leiaf: INPUTF exp=exprLogicoAritmetica { LEIAF exp }
126 leias: INPUTS exp=exprLogicoAritmetica { LEIAS exp }
127
128 loopFOR:
129 | FOR expid=exprLogicoAritmetica IN RANGE APAR
130     exprcomeco = exprLogicoAritmetica VIRG exprfim =
131     exprLogicoAritmetica FPAR DPONTOS NOVALINHA INDENTA
132     stm = comandos DEDENTA {
133     FORLOOP(expid,exprcomeco,exprfim,stm)}
134 ;
135
136 loopWhile: WHILE exprla = exprLogicoAritmetica DPONTOS
137     NOVALINHA INDENTA stm = comandos DEDENTA {
138     WHILELOOP(exprla,stm) }
139
140 exprLogicoAritmetica:
141 | f = chamada { f
142     }
143 | id = ID { EXPVAR(id)

```

```

133         | i = LITINT                                { EXPINT(i)
        }
134     | s = LITSTRING                                {
        EXPSTRING(s)    }
135     | f = LITFLOAT                                {
        EXPFLOAT(f)    }
136     | b = LITBOOL                                { EXPBOOL
        (b) }
137     | op=opU e=exprLogicoAritmetica %prec unary_minus {
        EXPOPU (op,e) }
138     | e1=exprLogicoAritmetica op = opB e2 =
        exprLogicoAritmetica { EXPOPB (op,e1,e2) }
139     | APAR e=exprLogicoAritmetica FPAR                { e
        }
140 ;
141
142 tipo:
143     | BOOL          { BOOLEAN  }
144     | INT            { INTEIRO  }
145     | FLOAT          { REAL     }
146     | NONE           { NONE     }
147     | STR            { STRING   }
148 ;
149
150 %inline opB:
151     | pos = MAIS                { (ADICAO, pos) }
152     | pos = MENOS                { (SUBTRACAO, pos) }
153     | pos = VEZES                { (MULTIPLICACAO,pos) }
154     | pos = DIVIDIDO            { (DIVISAO, pos) }
155     | pos = MODULO              { (MOD, pos) }
156     | pos = IGUALDADE            { (EHIGUAL, pos) }
157     | pos = MAIOR                { (MAIORQ, pos) }
158     | pos = MAIORIGUAL          { (MAIORIGUALQ, pos) }
159     | pos = MENOR                { (MENORQ, pos) }
160     | pos = MENORIGUAL          { (MENORIGUALQ, pos)}
161     | pos = DIFERENTE            { (EHDIFERENTE, pos) }
162     | pos = E                    { (AND, pos) }

```

```
163 | pos = OU          { (OR,      pos) }
164 |
165
166 %inline opU:
167 | pos = NOT  { (NEGACAO, pos)  }
168 | pos = MENOS { (SUBTRACAO, pos ) }
169 |
```

ast.mll

Listing 131: ast.mll

```
1      (* The type of the abstract syntax tree (AST). *)
2
3
4      type identificador = string
5      (*posicao no arquivo*)
6      type 'a pos = 'a * Lexing.position
7
8      type 'expr programa = Programa of 'expr instrucoes
9      and 'expr comandos = 'expr comando list
10     and 'expr instrucoes = 'expr instrucao list
11     and 'expr expressoes = 'expr list
12     and 'expr instrucao =
13     Funcao of 'expr decfn
14     | ACMD of 'expr comando
15     and 'expr decfn = {
16     fn_nome: identificador pos;
17     fn_tiporet: tipo;
18     fn_formais: (identificador pos * tipo) list;
19     fn_corpo: 'expr comandos
20     }
21
22     and tipo =
23     BOOLEAN
24     | INTEIRO
25     | REAL
26     | NONE
27     | STRING
28
29     and 'expr comando =
30     ATRIBUICAO of 'expr * 'expr
31     | CONDICAOIF of 'expr * ('expr comando) list * ('expr
32       comando option)
33     | CONDICAOelifElse of 'expr comandos
34     | WHILELOOP of 'expr * ('expr comando) list
```

```
34 | FORLOOP of 'expr * 'expr * 'expr * ('expr comando) list
35 | PRINT of 'expr
36 | RETORNO of 'expr option
37 | LEIAI of 'expr
38 | LEIAF of 'expr
39 | LEIAS of 'expr
40 | CHAMADADEFUNCAO of 'expr
41
42 and operador =
43 ADICAO
44 | SUBTRACAO
45 | MULTIPLICACAO
46 | DIVISAO
47 | MOD
48 | EHIGUAL
49 | MAIORQ
50 | MAIORIGUALQ
51 | MENORQ
52 | MENORIGUALQ
53 | EHDIFERENTE
54 | AND
55 | OR
56 | NEGACAO
```

sast.mll

Listing 132: sast.mll

```
1  open Ast
2
3  type expressao =
4  EXPOPB of operador pos * expressao * expressao
5      | EXPOPU of operador pos * expressao
6      | EXPVAR of identificador pos
7      | EXPINT of int pos
8      | EXPSTRING of string pos
9      | EXPFLOAT of float pos
10     | EXPBOOL of bool pos
11     | EXPCALL of identificador pos * (expressao expressoes)
```

tast.mll

Listing 133: tast.mll

```
1
2  open Ast
3
4  type expressao =
5  EXPOPB of (operador * tipo) * (expressao * tipo) *
        (expressao * tipo)
6      | EXPOPU of (operador * tipo) * (expressao * tipo)
7      | EXPCALL of identificador * (expressao expressoes) *
        tipo
8      | EXPINT of int * tipo
9      | EXPSTRING of string * tipo
10     | EXPFLOAT of float * tipo
11     | EXPBOOL of bool * tipo
12     | EXPVAR of identificador * tipo
13     | EXPNONE
```

interpreteTeste.ml

Listing 134: interpreteTeste.ml

```
1
2  open Printf
3  open Lexing
4
5  open Ast
6  exception Erro_Sintatico of string
7
8  module S = MenhirLib.General (* Streams *)
9  module I = Sintatico.MenhirInterpreter
10
11
12  (* This file was auto-generated based on "sintatico.msg". *)
13
14  (* Please note that the function [message] can raise
15     [Not_found]. *)
16
17  let message =
18  fun s ->
19  match s with
20  | 9 ->
21    "<esperava expressao>\n"
22  | 60 ->
23    "<esperava dois pontos>\n"
24  | 24 ->
25    "<esperava fim de expressao>\n"
26  | 27 ->
27    "<esperava fim de expressao>\n"
28  | 28 ->
29    "<esperava dois pontos>\n"
30  | 29 ->
31    "<esperava fim de expressao>\n"
32  | 31 ->
33    "<esperava fim de expressao>\n"
34  | 32 ->
```

```
34  "<esperava dois pontos>\n"
35  | 35 ->
36  "<esperava fim de expressao>\n"
37  | 36 ->
38  "<esperava dois pontos>\n"
39  | 39 ->
40  "<esperava fim de expressao>\n"
41  | 40 ->
42  "<esperava dois pontos>\n"
43  | 37 ->
44  "<esperava fim de expressao>\n"
45  | 38 ->
46  "<esperava dois pontos>\n"
47  | 41 ->
48  "<esperava fim de expressao>\n"
49  | 42 ->
50  "<esperava dois pontos>\n"
51  | 43 ->
52  "<esperava fim de expressao>\n"
53  | 44 ->
54  "<esperava dois pontos>\n"
55  | 45 ->
56  "<esperava fim de expressao>\n"
57  | 46 ->
58  "<esperava dois pontos>\n"
59  | 47 ->
60  "<esperava fim de expressao>\n"
61  | 48 ->
62  "<esperava dois pontos>\n"
63  | 61 ->
64  "<esperava nova linha>\n"
65  | 62 ->
66  "<erro de indentacao>\n"
67  | 63 ->
68  "<esperava expressao ou declaracao>\n"
69  | 94 ->
70  "<erro de indentacao>\n"
71  | 33 ->
```

```
72  "<esperava fim de expressao>\n"
73  | 49 ->
74  "<esperava fim de expressao>\n"
75  | 50 ->
76  "<esperava dois pontos>\n"
77  | 11 ->
78  "<esperava termino de expressao >\n"
79  | 18 ->
80  "<esperava dois pontos>\n"
81  | 21 ->
82  "<esperava expressao ou variavel>\n"
83  | 23 ->
84  "<esperava fechamento de parenteses>\n"
85  | 0 ->
86  "<declaracao invalida>\n"
87  | 64 ->
88  "<esperava variavel ou expressao>\n"
89  | 65 ->
90  "<esperava nova linha>\n"
91  | 162 ->
92  "<erro de indentacao>\n"
93  | 67 ->
94  "<esperava parenteses>\n"
95  | 68 ->
96  "<esperava string>\n"
97  | 69 ->
98  "<esperava fechamento de parenteses ou virgula>\n"
99  | 71 ->
100 "<esperava variavel ou expressao>\n"
101 | 73 ->
102 "<esperava nova linha>\n"
103 | 15 ->
104 "<esperava parenteses>\n"
105 | 16 ->
106 "<esperava fechamento de parenteses>\n"
107 | 100 ->
108 "<esperava nova linha>\n"
109 | 5 ->
```

```
110     "<esperava identificador>\n"
111     | 6 ->
112     "<esperava novalinha>\n"
113     | 165 ->
114     "<esperava import ou qualquer declaracao valida>\n"
115     | 75 ->
116     "<esperava variavel ou expressao>\n"
117     | 76 ->
118     "<esperava dois pontos>\n"
119     | 77 ->
120     "<esperava nova linha>\n"
121     | 78 ->
122     "<erro de indentacao>\n"
123     | 79 ->
124     "<esperava expressao ou declaracao>\n"
125     | 115 ->
126     "<esperava def de funcao, expressao ou declaracao>\n"
127     | 124 ->
128     "<esperava dois pontos>\n"
129     | 125 ->
130     "<esperava nova linha>\n"
131     | 126 ->
132     "<erro de indentacao>\n"
133     | 127 ->
134     "<esperava expressao ou declaracao>\n"
135     | 116 ->
136     "<esperava variavel ou expressao>\n"
137     | 117 ->
138     "<esperava dois pontos>\n"
139     | 118 ->
140     "<esperava nova linha>\n"
141     | 119 ->
142     "<erro de indentacao>\n"
143     | 120 ->
144     "<esperava expressao ou comando>\n"
145     | 132 ->
146     "<esperava expressao ou comando>\n"
147     | 80 ->
```

```
148     "<esperava parenteses>\n"
149     | 81 ->
150     "<esperava expressao>\n"
151     | 82 ->
152     "<esperava nova linha>\n"
153     | 19 ->
154     "<esperava variavel ou tipo>\n"
155     | 55 ->
156     "<parenteses nao fechado>\n"
157     | 56 ->
158     "<esperava variavel ou tipo>\n"
159     | 103 ->
160     "<esperava novalinha>\n"
161     | 1 ->
162     "<esperava identificador>\n"
163     | 4 ->
164     "<esperava import>\n"
165     | 84 ->
166     "<esperava variavel>\n"
167     | 85 ->
168     "<esperava in>\n"
169     | 86 ->
170     "<esperava expressao>\n"
171     | 108 ->
172     "<esperava dois pontos>\n"
173     | 109 ->
174     "<esperava nova linha>\n"
175     | 110 ->
176     "<erro de indentacao>\n"
177     | 111 ->
178     "<esperava expressao ou declaracao>\n"
179     | 87 ->
180     "<esperava expressao>\n"
181     | 88 ->
182     "<parenteses nao fechado>\n"
183     | 89 ->
184     "<parenteses nao fechado>\n"
185     | 90 ->
```

```
186     "<esperava dois pontos>\n"
187     | 91 ->
188     "<esperava nova linha>\n"
189     | 92 ->
190     "<erro de indentacao>\n"
191     | 93 ->
192     "<esperava declaracao ou expressao>\n"
193     | 136 ->
194     "<esperava identificador de funcao>\n"
195     | 137 ->
196     "<esperava parenteses>\n"
197     | 138 ->
198     "<esperava variavel e tipo como argumentos>\n"
199     | 139 ->
200     "<esperava dois pontos e tipo de variavel>\n"
201     | 140 ->
202     "<esperava tipo de variavel>\n"
203     | 146 ->
204     "<esperava virgula>\n"
205     | 148 ->
206     "<esperava variavel e tipo como argumento>\n"
207     | 151 ->
208     "<esperava seta>\n"
209     | 152 ->
210     "<esperava tipo apos seta>\n"
211     | 153 ->
212     "<esperava dois pontos>\n"
213     | 154 ->
214     "<esperava nova linha>\n"
215     | 155 ->
216     "<erro de indentacao>\n"
217     | 156 ->
218     "<esperava declaracao ou expressao>\n"
219     | _ ->
220     raise Not_found
221
222
223     open Semantico
```

```

224 let posicao lexbuf =
225 let pos = lexbuf.lex_curr_p in
226 let lin = pos.pos_lnum
227 and col = pos.pos_cnum - pos.pos_bol - 1 in
228 sprintf "linha %d, coluna %d" lin col
229
230 (* [pilha checkpoint] extrai a pilha do automato LR(1)
    contida em checkpoint *)
231
232 let pilha checkpoint =
233 match checkpoint with
234 | I.HandlingError amb -> I.stack amb
235 | _ -> assert false (* Isso nao pode acontecer *)
236
237 let estado checkpoint : int =
238 match Lazy.force (pilha checkpoint) with
239 | S.Nil -> (* 0 parser esta no estado inicial *)
240 0
241 | S.Cons (I.Element (s, _, _, _), _) ->
242 I.number s
243
244 let sucesso v = Some v
245
246 let falha lexbuf (checkpoint : (Sast.expressao
    Ast.programa) I.checkpoint) =
247 let estado_atual = estado checkpoint in
248 let msg = message estado_atual in
249 raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n"
250 (Lexing.lexeme_start lexbuf) msg))
251
252 let loop lexbuf resultado =
253 let fornecedor = I.lexer_lexbuf_to_supplier
    Pre_processador.lexico lexbuf in
254 I.loop_handle sucesso (falha lexbuf) fornecedor resultado
255
256
257 let parse_com_erro lexbuf =
258 try

```

```

259     Some (loop lexbuf (Sintatico.Incremental.programa
        lexbuf.lex_curr_p))
260   with
261   | Lexico.Erro msg ->
262     printf "Erro lexico na %s:\n\t%s\n" (posicao lexbuf) msg;
263     None
264   | Erro_Sintatico msg ->
265     printf "Erro sintatico na %s %s\n" (posicao lexbuf) msg;
266     None
267
268   let parse s =
269     let lexbuf = Lexing.from_string s in
270     let ast = parse_com_erro lexbuf in
271     ast
272
273   let parse_arq nome =
274     let ic = open_in nome in
275     let lexbuf = Lexing.from_channel ic in
276     let ast = parse_com_erro lexbuf in
277     let _ = close_in ic in
278     ast
279
280   let verifica_tipos nome =
281     let ast = parse_arq nome in
282     match ast with
283     Some (Some ast) -> semantico ast
284     | _ -> failwith "Nada a fazer!\n"
285
286
287   let interprete nome =
288     let tast,amb = verifica_tipos nome in
289     Interprete.interprete tast

```


12 Referências

12.1 Bibliográficas

1. Construção de Compiladores - **Bruna Felice da Silva, Fernando Henrique de Oliveira, Guilherme Ferreira Ribeiro**
2. Construção de Compiladores - Compilador de AnalisadorLexico para Máquina Virtual Dalvik - **Alana Rocha Santos**

12.2 Webgráficas

1. [Diagrama de Funcionamento](#)
2. [Código Smali](#)
3. [Convert Java to Smali](#)
4. [Dalvik Opcodes](#)