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

José Augusto Bolina
joseaugusto.bolina@hotmail.com

Faculdade de Computação
Universidade Federal de Uberlândia

12 de julho de 2018

# Lista de Listagens

# Sumário

# Capítulo 1

# Introdução

Este documento foi escrito para auxiliar no decorrer da disciplina de Construção de Compiladores, servindo de guia para o seguintes trabalhos. O projeto se trata da criação de um compilador da linguagem MiniC para a máquina virtual Dalvik, utilizando a linguagem OCaml para a construçao do compilador.

## 1.1 Dalvik VM

A Dalvik VM foi construída como parte da plataforma Android, atualmente essa máquina foi descontinuada, onde sua sucessora foi a **Android Runtime** ou **ART**, foi utilizada em versões do Android 4.4 KitKat e anteriores, utilizado principalmente em celulares e tablets. A máquina virtual Dalvik, é uma máquina baseada em registradores, o que requer menos instruções, mas em contrapartida, são instruções mais complexas, cada registrador pode comportar qualquer tipo, seja *char* ou *float*, valores como *long* ou *double* precisam ser armazenados em dois registradores consecutivos. A máquina virtual é otimizada para requerer pouca memória, e é projetada para permitir que múltiplas instâncias da máquina virtual rodem ao mesmo tempo, deixando para o sistema operacional o isolamento de processos, o gerenciamento de memória e o suporte a threading.

## 1.2 Formatos de arquivos

Dispositivos Android, normalmente recebe aplicações que são desenvolvidas em Java, que gera um arquivo *.class*. A máquina virtual Dalvik executa somente arquivos *.dex*, sendo assim necessário a utilização de uma ferramenta chamada *dx*, para realizar a transformação dos arquivos *.class* para *.dex*.
Um arquivo *.dex* é bastante otimizado, visto que em apenas um arquivo podem ser incluidas diversas classes Java, Strings e constantes duplicadas e comuns entre as classes são adicionadas somente uma vez no arquivo, assim economizando espaço.
O assembler/disassembler no formato *.dex* utilizado pelo Dalvik VM nesse trabalho é o Smali, que irá criar um arquivo de saída *.smali*. Essa conversão nos retorna um arquivo possível de ser lido, esse arquivo está na linguagem *smali*.

## 1.3 Smali

Com os programadores criando arquivos para Android em Java, o compilador cria então os arquivos em *jar*, depois o arquivos são convertidos para *dex*, assim a máquina virtual Dalvik podera executar esses arquivos.
Assim, uma das maneiras para poder criar modificações para o seu sistema é através de modificações desses arquivos, ou criando os de autoria própria. Algumas ROMs modificadas como o *Cyanogenmod* disponibiliza o código fonte, que poderia ser alterado para suas necessidades, enquanto outras OEM (*Original Equipment Manufacturer*) não disponibilizam seus códigos para modificações.
Para realizar as modificações, é necessário transformar os arquivos *dex* disponíveis para o arquivo *smali*, que assim ficará legível na linguagem *smali*. Essa transformação reversa é chamada de *baksmaling*. Quando os arquivos *Java*são transformados em *smali* algumas informações são transformadas ou irrelevantes, por essa razão não é possível transformar os arquivos de volta para *Java*.

# Capítulo 2

# Dependências

## 2.1 Ambiente

O sistema operacional utilizado foi o Arch Linux 64 bits, com o kernel Linux na versão *4.13.12-1-ARCH*, para a instalação das ferramentas necessárias foi utilizado ambos gerenciadores de pacotes disponíveis, *yaourt* ou *pacman*.

## 2.2 Ferramentas e instalação

As ferramentas necessária foram Java, na versão 8, para criação dos arquivos de *.java* para *.class*. Para instalação dessa dependência, é feito o segundo comando:

```
> yaourt java
```

Assim, selecionando o número referente ao pacote que contém a JDK da versão deseja, isso realizará a instalação da JRE de mesma versão e a criação das variáveis de ambiente automaticamente.

Também foi utilizado a SDK do Android para conversão dos arquivos *.class* para *.dex*. Para instalação da SDK é utilizado o comando:

```
> yaourt android-studio
```

Selecionando o número do pacote desejado, será baixado a IDE para Android e junto com ela virá a SDK do Android.
Por fim, foi utilizado o assembler/disassembler para *.smali*, para instalar a ferramenta Baksmali, é feito seguinte comando:

```
> yaourt smali
```

Após selecionar o número do pacote, o assembler será instalado na máquina.

## 2.3 Utilização

Para utilizar tais ferramentas na prática, é realizado uma sequência de passos, para entendimento, os arquivos terão nomes como *Teste* ou *arquivo*. Inicialmente, tendo o programa Java em um um arquivo *.java*, é realizado seguinte comando:

```
> javac Teste.java
```

Após isso, será criado um arquivo *Teste.class*, que será passado na ferramenta *dx* e será transformado em um arquivo *.dex*, para isso, siga o comando:

```
> ANDROID/Sdk/build-tools/(versão)/dx --dex --output=arquivo.dex Teste.
    class
```

Onde ANDROID é o caminho onde o Android Studio está instalado, assim acessamos a SDK e o *dx* para criação do arquivo binário *.dex*.
Para o disassembly do *.dex* para o arquivo *.smali* é realizado seguinte comando:

```
> baksmali d arquivo.dex
```

Isso irá criar uma nova pasta com nome *out*, que irá conter o arquivo *arquivo.smali*

# Capítulo 3

# Códigos

Nessa seção serão incluidos os arquivos de códigos utilizados. A linguagem inicial é a linguagem C, mas para conversão para *smali* foi utilizado como uma linguagem intermediária, a linguagem Java.
A seguir serão listados os arquivos *.c, .java* e *.smali*.

## 3.1 Exemplos iniciais

Essa seção contém alguns exemplos de códigos, mas se mantendo com exemplos somente em *Java* e *Smali*.

### 3.1.1 Sobre a linguagem smali

A linguagem *smali*, lembra em alguns aspectos a linguagem *Java*, mas, por ser uma linguagem assembly, não possue *for, while* ou *loops*, tudo é tratado com *labels* e *goto*.

Números em *smali* são representados na forma hexadecimal no padrão *IEEE*754, é possível utilizar ferramentas de conversão online, como [esta], mas existem algumas jogadas. Possuindo um valor armazenado 330 em um registrador, por exemplo o valor *0x43a5*, para verificar o valor, copiando ele e colocando no conversor é notado que a conversão não ocorre de maneira errada, pois o valor é um hexadecimal 16-bits e o conversor aceita no mínimo valores de 32-bits, sendo assim, o conversor adicionará mais *0*'s para completar, finalizando com *0x000043a5* que equivale a *2.4266E-41*.

Esse erro ocorre porque os valores são na verdade *little-endians*, sendo assim, é necessário adicionar mais *0*'s e utilizar como *input* para conversão o valor *0x43a50000*, que corresponde a 330.

Pelo funcionamento da máquina virtual Dalvik, teremos que seus registradores se com-

portarão como variáveis locais, para cada método chamado terá um novo conjunto de registradores, esses registradores não afetarão os registradores do método de onde foi invocado.

Os tipos primitivos do *smali* seguem o seguinte:

| Tipo | Representação |
|---|---|
| int | I |
| long | J |
| boolean | Z |
| double | D |
| float | F |
| short | S |
| char | C |
| void | V |

### 3.1.2 Exemplos

Um simples exemplos de *if/else* em Java seria:

```
if (flagx == 1)
    flagx = 2
else
    flagx = 3
```

Quando convertido para *smali* é equivalente a:

```
const/4 v1, 0x1             % constante 1 atribuida ao registrador v1
if-ne v0, v1, :cond_0       % se v0 diferente v1, vá para label cond_0
const/4 v2, 0x2             % constante 2 atribuida ao registrador v2
move v0, v2                 % mover valor de v2 para v0
goto :goto_0                % va para label goto_0
:cond_0                     % inicio da label cond_0
const/4 v2, 0x3             % constante 3 atribuida ao registrador v2
move v0, v2                 % mover valor de v2 para v0
:goto_0                     % label goto_0
```

Outro exemplo, com a estrutura completa do arquivo *smali* seria a seguinte:

**Listagem 3.1: Exemplo de código com a estrutura**

```
1 # nome da classe e local onde foi realizado o dump
2 .class public Lcom/packageName/example;
3
4 # Sub classe de
5 .super Ljava/lang/Object;
6
7 # Note que o a estrutura de nome segue: L<caminho da classe>
```

```
8
9  # Nome do arquivo original
10 .source "example.java"
11
12
13 # Variaveis que sao instancias de classes, observe que tipos nao
       primitivos
14 # necessitam de ponto e virgula no fim
15 .field private someString:Ljava/lang/String;
16
17 # Tipo finais não são usados diretamente, as referencias para
18 # eles são substituidos pelos proprios valores.
19 .field public final someInt:I
20 .field public final someBool:Z
21
22
23 # Arrays são feitos na forma
24 .field public final someCharArray:[C
25 .field private someStringArray:[Ljava/lang/String;
26
27
28 # Aqui sera chamado o <init> do método, ou seja, seu construtor.
29 # Seu construtor ira chamar o construtor da super classe.
30 # Observe que o construtor retornara V, que é void
31 .method public constructor <init>(ZLjava/lang/String;I)V
32
33
34 # Declarado quantas variaveis existem
35 .registers 6
36
37
38 # Parametros não costumam aparecer, dizem os nomes dos parametros
39 # quando estavam no Java
40 .parameter "someBool"
41 .parameter "someInt"
42 .parameter "exampleString"
43
44
45 # Diretivas como prologue e line podem ser ignoradas, a linha pode
46 # ser util para casos de debug
47 .prologue
48 .line 10
49
50
51 # p0 referencia o parametro 0, onde estamos chamado a super classe
52 invoke-direct {p0}, Ljava/lang/Object;-><init>()V
53
54
55 # Armazena String a v0
56 const-string v0, "fala ai, meu amigo"
57
58
59 # Armazena o valor 0xF em v0, notando que registradores podem
60 # armazenar qualquer tipo de dados
61 const/4 v0, 0xF
62
63
64 # Uma nova instancia do objeto StringBuilder
65 new-instance v1, Ljava/lang/StringBuilder;
```

```
66
67
68 # Inicializa StringBuilder com valor v2
69 const-string v2, "iniciando objeto"
70 invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/
       String;)V
71
72
73 # concatenamos p1 que é booleano, o retorno será um StringBuilder
74 # para concater utilizamos append(Z)
75 invoke-virtual {v1, p1}, Ljava/lang/StringBuilder;->append(Z)Ljava/lang/
       StringBuilder;
76
77
78 # move resultado passado para v1, caso retorno nao seja objeto se utiliza
79 # move-result ou move-result-wide
80 move-result-object v1
81
82
83 # concatenamos uma String com StringBuilder
84 # observe que append agora recebe um tipo String
85 const-string v2, "uma string aleatoria"
86 invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
       String;)Ljava/lang/StringBuilder;
87 move-result-object v1
88
89
90 # chamamos o metodo toString() do StringBuilder
91 invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
       String;
92 move-result-object v1
93
94
95 # capturando o tempo atual, observe que o tipo é float
96 invoke-static {}, Ljava/lang/System;->currentTimeMillis()J
97
98
99 # Por ser um wide-result, ira ocupar os registradores v2 e v3
100 move-result-wide v2
101
102 # Dessa forma, para não perder o valor é necessario utilizar v4
103 const-wide/16 v4, 0x300 # ira ocupar v4 e v5
104 div-long/2addr v2, v4   # divide v2 por v4
105 long-to-int v2, v2      # transforma em um inteiro
106
107
108 # Invocar um método estático tem a seguinte forma
109 invoke-static {} , Lcom/algum/pacote/classe;->metodoEstatico()Ljava/lang/
       String;
110
111 # passamos os parametros {}, o pacote -> o método() e o tipo de retorno
112
113 return-void
114 .end method
115
116 .method public static metodoEstatico()Ljava/lang/String;
117 .registers 4
118
119 new-instance v0, Ljava/lang/Long;
```

```
120 invoke-static {}, Ljava/lang/System;->currentTimeMillis()J
121 move-result-wide v1
122
123 invoke-direct {v0, v1, v2}, Ljava/lang/Long;-><init>()V
124
125 invoke-static {v0}, Ljava/lang/String;->valueOf(Ljava/lang/Object;)Ljava/
      lang/String;
126
127 move-result-object v1
128
129 # observe que é return object e não so return
130 return-object v1
131 .end method</class>
```

## 3.2 Instruções da linguagem

A linguagem *smali* possui também um conjunto de instruções para ser executado na máquina Dalvik, grande parte dessas instruções pode ser encontrada [aqui], onde possui os *opcodes*, o nome da instrução em si, como utilizar e um exemplo de utilização. As instruções são:

**Movimentação entre registradores e de resultados**

- move. São utilizados para movimentar valores entre registradores e movimentar/retornar valores/objetos retornados pelos métodos, para diferentes tamanhos utilize */from16* no fim do *opcode*. Variações são:
  - move-{wide, result, object, result-object}
  - return-{void, wide, object}
- const. Para criação de constantes no código, podendo ser variar no tamanho e espaço nos registradores, para tal, adicione */4, /16, /32* ou */high16*. Variações são:
  - const-{wide, string, class}
- array. Para manipulação de arrays na linguagem, deve-se utilizar:
  - new-array
  - array-length
  - filled-new-array

  - fill-array-data

- new-instance
- Para controle de execução, existem as seguintes instruções:
  - goto
  - packed-switch
  - sparse-switch
  - if-{eq, ne, lt, ge, gt, le, eqz, nez, ltz, gez, gtz, lez}
- Para leitura/escrita de de campos, possui:
  - iget-{wide, object, boolean, byte, char, short}
  - iput-{wide, object, boolean, byte, char, short}
- Leitura/escrita em arrays:
  - aget-{wide, object, boolean, byte, char, short}
  - aput-{wide, object, boolean, byte, char, short}
- Invocação de métodos para execução:
  - invoke-{virtual, super, direct, static, interface}
- Operações aritméticas em valores como *int, long, float, double*
  - add
  - sub
  - mul
  - div
  - rem
  - and
  - or
  - xor
  - shl

- shr
- usrh
- neg-{int, long, float, double}
- not-{int, long}

## 3.3 Nano programas

Os nano programas contemplam simples programas, desde somente a definição da função principal, atribuição de valores para variáveis, somas até estruturas de loop e decisão.

### 3.3.1 nano03

**Listagem 3.2: nano03.c**

```c
void main() {
  int n;
  n = 1;
}
```

**Listagem 3.3: nano03.java**

```java
public class nano03 {
    public static void main(String[] args) {
        int n;
        n = 1;
    }
}
```

**Listagem 3.4: nano03.smali**

```
# Nome da classe
.class public Lnano03;

# Nome da super classe
.super Ljava/lang/Object;

# Nome original do arquivo
.source "nano03.java"


# Chamado do construtor
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 1
```

```
17
18      # Invoca o construtor da super classe
19      invoke-direct {p0}, Ljava/lang/Object;-><init>()V
20
21      return-void
22  .end method
23
24  # Método main, observe que como a main definida no Java.
25  # Recebe uma String como parametro e retorna void
26  .method public static main([Ljava/lang/String;)V
27      # Utiliza um registrador
28      .registers 1
29
30      .prologue
31      .line 4
32      .line 5
33
34      # Retorna void
35      return-void
36  .end method
```

### 3.3.2 nano04

**Listagem 3.5: nano04.c**

```
1 void main() {
2   int n;
3   n = 1 + 2;
4 }
```

**Listagem 3.6: nano04.java**

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

**Listagem 3.7: nano04.smali**

```
1  # Nome da classe
2  .class public Lnano04;
3
4  # Nome da super classe
5  .super Ljava/lang/Object;
6
7  # Nome do arquivo
8  .source "nano04.java"
9
10
11 # Método construtor
12 .method public constructor <init>()V
13     .registers 1
14
15     .prologue
16     .line 1
```

```
17
18      # Invoca o construtor da super classe
19      invoke-direct {p0}, Ljava/lang/Object;-><init>()V
20
21      return-void
22  .end method
23
24  # Método main, observe que como a main definida no Java.
25  # Recebe uma String como parametro e retorna void
26  .method public static main([Ljava/lang/String;)V
27      # Utiliza um registrador
28      .registers 1
29
30      .prologue
31      .line 4
32      .line 5
33
34      # Retorna void
35      return-void
36  .end method
```

## 3.4 Micro programas

Os micro programas contemplam programas um pouco mais complicados que os nano, os micro programas utilizam de funções auxiliares, laços e tomadas de decisões.

### 3.4.1 micro10

Listagem 3.8: micro10.c

```c
#include <stdio.h>

int fatorial(int n) {
  return (n <= 0) ? 1 : (n * fatorial(n-1));
}

int main() {
  int numero, fat;

  printf("Digite um número: ");
  scanf("%d", &numero);
  fat = fatorial(numero);

  printf("O fatorial de %d é %d\n", numero, fat);
}
```

Listagem 3.9: micro10.java

```java
public class micro10 {
    static int fatorial(int n) {
        return (n <= 0) ? 1 : (n * fatorial(n-1));
    }

    public static void main(String[] args) {
```

```
7           int numero, fat;
8
9           System.out.println("Digite um número: ");
10          numero = Integer.parseInt(System.console().readLine());
11          fat = fatorial(numero);
12
13          System.out.println("O fatorial de "+numero+" é "+fat+"\n");
14      }
15
16
17  }
```

**Listagem 3.10: micro10.smali**

```
1  # Nome da classe
2  .class public Lmicro10;
3
4  # Nome da super classe
5  .super Ljava/lang/Object;
6
7  # Nome do arquivo original
8  .source "micro10.java"
9
10
11 # direct methods
12 .method public constructor <init>()V
13     .registers 1
14
15     .prologue
16     .line 1
17
18     # Chama construtor da super classe
19     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
20
21     # O método retorna void
22     return-void
23 .end method
24
25
26 # Referencia ao método static fatorial, que recebe um int como
27 # parametro e retorna outro int
28 .method static fatorial(I)I
29
30     # Utiliza dois registradores
31     .registers 2
32
33     .prologue
34     .line 3
35
36     # Se o parametro 0, que é um int, for maior que 0 vá para cond_4
37     if-gtz p0, :cond_4
38
39     # Armazene o valor 1 em v0
40     const/4 v0, 0x1
41
42     # Label goto_3
43     :goto_3
44     return v0
45
```

```
46      :cond_4
47      # Adiciona o parametro 0 com literal -0x1 e armazena resultado em v0
48      add-int/lit8 v0, p0, -0x1
49
50      # Recursivamente invoca o método fatorial passando o valor v0
51      invoke-static {v0}, Lmicro10;->fatorial(I)I
52
53      # Com o resultado da invocação é movido para v0
54      move-result v0
55
56      # Multiplica o v0 com p0 e armazena o resultado em v0
57      mul-int/2addr v0, p0
58
59      # Va para label goto_3
60      goto :goto_3
61  .end method
62
63  .method public static main([Ljava/lang/String;)V
64      .registers 6
65
66      .prologue
67      .line 9
68
69      # Lê o objeto para v0
70      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
71
72      # Armazena a String em v1
73      const-string v1, "Digite um n\u00famero: "
74
75      # Invoca o método println passando os valores v0 e v1
76      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
77
78      .line 10
79      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
80
81      # Com o objeto resultado de retorno do console é armazenado em v0
82      move-result-object v0
83
84      # Chama o método readLine, para realizar leitura
85      invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
86
87      # Move o objeto resultado para v0
88      move-result-object v0
89
90      # Chama o método estático para transformar o objeto lido para int
91      invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
92
93      # Resultado movido para v0
94      move-result v0
95
96      .line 11
97
98      # A classe micro10 chama o método fatorial, passando v0 como parametro
99      invoke-static {v0}, Lmicro10;->fatorial(I)I
100
101     # Com o resultado do método fatorial é movido para v1
102     move-result v1
103
```

```
104     .line 13
105     # Lê o objeto para v2
106     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
107
108     # Criada uma nova instancia de StringBuilder e armazenada em v3
109     new-instance v3, Ljava/lang/StringBuilder;
110
111     # Chamado construtor de StringBuilder
112     invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V
113
114     # String constante em v4
115     const-string v4, "O fatorial de "
116
117     # Concatenamos v3 com v4
118     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
119
120     # Objeto resultante movido para v3
121     move-result-object v3
122
123     # Concatenamos v1 em v3, v1 contém o resultado do método fatorial
124     invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
125
126     # Toda String do resultado é movida para v0
127     move-result-object v0
128
129     # Sao adicionados os caracteres finais na String
130     const-string v3, " \u00e9 "
131     invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
132     move-result-object v0
133     invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
134     move-result-object v0
135     const-string v1, "\n"
136     invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
137     move-result-object v0
138     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
139     move-result-object v0
140
141     # Invocado método println passando v2 que é PrintStream e a String
            final
142     invoke-virtual {v2, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
143
144     .line 14
145
146     # Return
147     return-void
148 .end method
```

## 3.5 Análise sobre a linguagem

Com os exemplos passados, e com a arquitetura básica do arquivo *smali*, é notado que o *smali* possui bastante semelhança com o próprio *Java*, herdando todos os nomes dos objetos, métodos, parâmetros, pacotes, etc. que vieram no momento da compilação, sendo assim mais fácil de se localizar entre códigos.

## 3.6 Códigos de programas

Nessa seção serão incluidos os arquivos de códigos utilizados. A linguagem inicial é a linguagem C, mas para conversão para *smali* foi utilizado como uma linguagem intermediária, a linguagem Java.
A seguir serão listados os arquivos *.c, .java* e *.smali*.

## 3.7 Nano programas

Os nano programas contemplam simples programas, desde somente a definição da função principal, atribuição de valores para variáveis, somas até estruturas de loop e decisão.

### 3.7.1 nano01

**Listagem 3.11: nano01.c**

```
1 void main() {
2 }
```

**Listagem 3.12: nano01.java**

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

**Listagem 3.13: nano01.smali**

```
1 .class public Lnano01;
2 .super Ljava/lang/Object;
3 .source "nano01.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
```

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

### 3.7.2 nano02

**Listagem 3.14: nano02.c**

```
1 void main() {
2   int n;
3 }
```

**Listagem 3.15: nano02.java**

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

**Listagem 3.16: nano02.smali**

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

### 3.7.3 nano03

Já contido anteriormente.

### 3.7.4 nano04

Já contido anteriormente.

### 3.7.5 nano05

**Listagem 3.17: nano05.c**

```c
#include <stdio.h>

void main() {
  int n;
  n = 2;
  printf("%d", n);
}
```

**Listagem 3.18: nano05.java**

```java
public class nano05 {
    public static void main(String[] args) {
        int n;
        n = 2;
        System.out.println(n);
    }
}
```

**Listagem 3.19: nano05.smali**

```
.class public Lnano05;
.super Ljava/lang/Object;
.source "nano05.java"


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

    .prologue
    .line 1
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method

.method public static main([Ljava/lang/String;)V
    .registers 3

    .prologue
    .line 4
```

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

### 3.7.6 nano06

**Listagem 3.20: nano06.c**

```c
#include <stdio.h>

void main() {
  int n;
  n = 1 - 2;
  printf("%d", n);
}
```

**Listagem 3.21: nano06.java**

```java
public class nano06 {
    public static void main(String[] args) {
        int n;
        n = 1 - 2;
        System.out.println(n);
    }
}
```

**Listagem 3.22: nano06.smali**

```
.class public Lnano06;
.super Ljava/lang/Object;
.source "nano06.java"


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

    .prologue
    .line 1
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method

.method public static main([Ljava/lang/String;)V
    .registers 3

    .prologue
    .line 4
    const/4 v0, -0x1
```

```
23
24      .line 5
25      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27      invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(I)V
28
29      .line 6
30      return-void
31  .end method
```

### 3.7.7 nano07

**Listagem 3.23: nano07.c**

```
1 #include <stdio.h>
2
3 void main() {
4   int n;
5   n = 1;
6   if (n == 1)
7     printf("%d", n);
8 }
```

**Listagem 3.24: nano07.java**

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

**Listagem 3.25: nano07.smali**

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

```
22      const/4 v0, 0x1
23
24      .line 6
25      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27      invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(I)V
28
29      .line 7
30      return-void
31  .end method
```

### 3.7.8 nano08

**Listagem 3.26: nano08.c**

```
1  #include <stdio.h>
2
3  void main() {
4    int n;
5    n = 1;
6    if (n == 1)
7      printf("%d", n);
8    else
9      printf("0");
10 }
```

**Listagem 3.27: nano08.java**

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

**Listagem 3.28: nano08.smali**

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

```
.method public static main([Ljava/lang/String;)V
    .registers 3

    .prologue
    .line 4
    const/4 v0, 0x1

    .line 6
    sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;

    invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(I)V

    .line 9
    return-void
.end method
```

### 3.7.9 nano09

**Listagem 3.29: nano09.c**

```c
#include <stdio.h>

void main() {
  int n;
  n = 1 + 1 / 2;
  if (n == 1)
    printf("%d", n);
  else
    printf("0");
}
```

**Listagem 3.30: nano09.java**

```java
public class teste1_9 {
    public static void main(String[] args) {
        int n;
        n = 1 + 1 / 2;
        if (n == 1)
            System.out.println(n);
        else
            System.out.println(0);
    }
}
```

**Listagem 3.31: nano09.smali**

```
.class public Lteste1_9;
.super Ljava/lang/Object;
.source "teste1_9.java"


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

    .prologue
    .line 1
```

```
12      invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14      return-void
15  .end method
16
17  .method public static main([Ljava/lang/String;)V
18      .registers 3
19
20      .prologue
21      .line 4
22      const/4 v0, 0x1
23
24      .line 6
25      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27      invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(I)V
28
29      .line 9
30      return-void
31  .end method
```

### 3.7.10 nano10

**Listagem 3.32: nano10.c**

```
1  #include <stdio.h>
2
3  void main() {
4    int n, m;
5    n = 1;
6    m = 2;
7    if (n == m)
8      printf("%d", n);
9    else
10     printf("0");
11 }
```

**Listagem 3.33: nano10.java**

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

**Listagem 3.34: nano10.smali**

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

### 3.7.11 nano11

Listagem 3.35: nano11.c

```
1  #include <stdio.h>
2
3  void main() {
4    int n, m, x;
5    n = 1;
6    m = 2;
7    x = 5;
8
9    while (x > n) {
10     n = n + m;
11     printf("%d", n);
12   }
13 }
```

Listagem 3.36: nano11.java

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

```
11          }
12      }
13  }
```

**Listagem 3.37: nano11.smali**

```
1  .class public Lnano11;
2  .super Ljava/lang/Object;
3  .source "nano11.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 5
25     const/4 v1, 0x2
26
27     .line 6
28     const/4 v2, 0x5
29
30     .line 8
31     :goto_3
32     if-le v2, v0, :cond_c
33
34     .line 9
35     add-int/2addr v0, v1
36
37     .line 10
38     sget-object v3, Ljava/lang/System;->out:Ljava/io/PrintStream;
39
40     invoke-virtual {v3, v0}, Ljava/io/PrintStream;->println(I)V
41
42     goto :goto_3
43
44     .line 12
45     :cond_c
46     return-void
47 .end method
```

### 3.7.12 nano12

**Listagem 3.38: nano12c**

```c
#include <stdio.h>

void main() {
  int n, m, x;
  n = 1;
  m = 2;
  x = 5;

  while (x > n) {
    if (n == m)
      printf("%d", n);
    else
      printf("0");
    x = x - 1;
  }
}
```

**Listagem 3.39: nano12.java**

```java
public class nano12 {
    public static void main(String[] args) {
        int n, m, x;
        n = 1;
        m = 2;
        x = 5;

        while (x > n) {
            if (n == m)
                System.out.println(n);
            else
                System.out.println(0);
            x = x - 1;
        }
    }
}
```

**Listagem 3.40: nano12.smali**

```
.class public Lnano12;
.super Ljava/lang/Object;
.source "nano12.java"


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

    .prologue
    .line 1
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method

.method public static main([Ljava/lang/String;)V
    .registers 5

    .prologue
    .line 4
```

```
22      const/4 v1, 0x1
23
24      .line 6
25      const/4 v0, 0x5
26
27      .line 8
28      :goto_2
29      if-le v0, v1, :cond_d
30
31      .line 12
32      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
33
34      const/4 v3, 0x0
35
36      invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(I)V
37
38      .line 13
39      add-int/lit8 v0, v0, -0x1
40
41      goto :goto_2
42
43      .line 15
44      :cond_d
45      return-void
46  .end method
```

## 3.8 Micro programas

Os micro programas contemplam programas um pouco mais complicados que os nano, os micro programas utilizam de funções auxiliares, laços e tomadas de decisões.

### 3.8.1 micro01

**Listagem 3.41: micro01.c**

```c
#include <stdio.h>

/*
  Função: Ler uma temperatura em graus Celsius e apresentá-la
    convertida em graus Fahrenheit. A fórmula de conversão é:

    F=(9*C+160) / 5,

  Sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.
*/

void main() {
  float cel, far;
  printf("Tabela de conversão: Celsius -> Fahrenheit\n");
  printf("Digite a temperatura em Celsius: ");
  scanf("%f", &cel);
  far = (9*cel+160) / 5;
  printf("A nova temperatura é: %fF\n", far);
}
```

**Listagem 3.42: micro01.java**

```
/*
  Função: Ler uma temperatura em graus Celsius e apresentá-la
    convertida em graus Fahrenheit. A fórmula de conversão é:

    F=(9*C+160) / 5,

  Sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.
*/

public class micro01 {
    public static void main(String[] args) {
        double cel, far;
        System.out.println("Tabela de conversão: Celsius -> Fahrenheit\n")
            ;
        System.out.println("Digite a temperatura em Celsius: ");
        cel = Double.parseDouble(System.console().readLine());
        far = (9*cel+160) / 5;
        System.out.println("A nova temperatura é: " + far + "F\n");
    }
}
```

**Listagem 3.43: micro01.smali**

```
.class public Lmicro01;
.super Ljava/lang/Object;
.source "micro01.java"


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

    .prologue
    .line 10
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method

.method public static main([Ljava/lang/String;)V
    .registers 6

    .prologue
    .line 13
    sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;

    const-string v1, "Tabela de convers\u00e3o: Celsius -> Fahrenheit\n"

    invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V

    .line 14
    sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;

    const-string v1, "Digite a temperatura em Celsius: "

    invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
        String;)V
```

```
34
35      .line 15
36      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
37
38      move-result-object v0
39
40      invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
41
42      move-result-object v0
43
44      invoke-static {v0}, Ljava/lang/Double;->parseDouble(Ljava/lang/String
            ;)D
45
46      move-result-wide v0
47
48      .line 16
49      const-wide/high16 v2, 0x4022000000000000L    # 9.0
50
51      mul-double/2addr v0, v2
52
53      const-wide/high16 v2, 0x4064000000000000L    # 160.0
54
55      add-double/2addr v0, v2
56
57      const-wide/high16 v2, 0x4014000000000000L    # 5.0
58
59      div-double/2addr v0, v2
60
61      .line 17
62      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
63
64      new-instance v3, Ljava/lang/StringBuilder;
65
66      invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V
67
68      const-string v4, "A nova temperatura \u00e9: "
69
70      invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
71
72      move-result-object v3
73
74      invoke-virtual {v3, v0, v1}, Ljava/lang/StringBuilder;->append(D)Ljava
            /lang/StringBuilder;
75
76      move-result-object v0
77
78      const-string v1, "F\n"
79
80      invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
81
82      move-result-object v0
83
84      invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
85
86      move-result-object v0
87
```

```
88      invoke-virtual {v2, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
89
90      .line 18
91      return-void
92  .end method
```

### 3.8.2 micro02

**Listagem 3.44: micro02.c**

```c
#include <stdio.h>

/* Função : Escrever um algoritmo que leia dois valores inteiro distintos
    e
 * informe qual é o maior.
 * */

void main() {
  int num1, num2;

  printf("Digite o primeiro número: ");
  scanf("%d", &num1);
  printf("Digite o segundo número: ");
  scanf("%d", &num2);

  if (num1 > num2) {
     printf("O primeiro número %d é maior que o segundo número %d", num1,
         num2);
  } else {
    printf("O segundo número %d é maior que o primeiro número %d", num2,
        num1);
  }
}
```

**Listagem 3.45: micro02.java**

```java
/* Função : Escrever um algoritmo que leia dois valores inteiro distintos
    e
 * informe qual é o maior.
 * */

public class micro02 {
    public static void main(String[] args) {
        int num1, num2;

        System.out.println("Digite o primeiro número: ");
        num1 = Integer.parseInt(System.console().readLine());
        System.out.println("Digite o segundo número: ");
        num2 = Integer.parseInt(System.console().readLine());

        if (num1 > num2) {
            System.out.println("O primeiro número " + num1 + " é maior que
                o segundo número " + num2);
        } else {
            System.out.println("O primeiro número " + num2 + " é maior que
                o segundo número " + num1);
        }
```

```
19      }
20  }
```

**Listagem 3.46: micro02.smali**

```
1  .class public Lmicro02;
2  .super Ljava/lang/Object;
3  .source "micro02.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 5
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 6
19
20     .prologue
21     .line 9
22     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite o primeiro n\u00famero: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
27
28     .line 10
29     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
30
31     move-result-object v0
32
33     invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
34
35     move-result-object v0
36
37     invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
38
39     move-result v0
40
41     .line 11
42     sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
43
44     const-string v2, "Digite o segundo n\u00famero: "
45
46     invoke-virtual {v1, v2}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
47
48     .line 12
49     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
50
51     move-result-object v1
52
```

```
53      invoke-virtual {v1}, Ljava/io/Console;->readLine()Ljava/lang/String;
54
55      move-result-object v1
56
57      invoke-static {v1}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
58
59      move-result v1
60
61      .line 14
62      if-le v0, v1, :cond_4b
63
64      .line 15
65      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
66
67      new-instance v3, Ljava/lang/StringBuilder;
68
69      invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V
70
71      const-string v4, "O primeiro n\u00famero "
72
73      invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
74
75      move-result-object v3
76
77      invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
78
79      move-result-object v0
80
81      const-string v3, " \u00e9 maior que o segundo n\u00famero "
82
83      invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
84
85      move-result-object v0
86
87      invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
88
89      move-result-object v0
90
91      invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
92
93      move-result-object v0
94
95      invoke-virtual {v2, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
96
97      .line 19
98      :goto_4a
99      return-void
100
101     .line 17
102     :cond_4b
103     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
104
105     new-instance v3, Ljava/lang/StringBuilder;
```

```
106
107     invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V
108
109     const-string v4, "O primeiro n\u00famero "
110
111     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
112
113     move-result-object v3
114
115     invoke-virtual {v3, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
116
117     move-result-object v1
118
119     const-string v3, " \u00e9 maior que o segundo n\u00famero "
120
121     invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
122
123     move-result-object v1
124
125     invoke-virtual {v1, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
126
127     move-result-object v0
128
129     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
130
131     move-result-object v0
132
133     invoke-virtual {v2, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
134
135     goto :goto_4a
136 .end method
```

### 3.8.3 micro03

**Listagem 3.47: micro03.c**

```
1  #include <stdio.h>
2
3  /* Função : Faça um algoritmo que receba um número e diga se este
4   * número está no intervalo entre 100 e 200.
5   * */
6
7  void main() {
8    int numero;
9
10   printf("Digite um número: ");
11   scanf("%d", &numero);
12
13   if (numero >= 100) {
14     if (numero <= 200)
15       printf("O número está no intervalo entre 100 e 200\n");
16     else
```

```
17        printf("O número não está no intervalo entre 100 e 200\n");
18    } else
19      printf("O número não está no intervalo entre 100 e 200\n");
20  }
```

**Listagem 3.48: micro03.java**

```
1  /* Função : Faça um algoritmo que receba um número e diga se este
2   * número está no intervalo entre 100 e 200.
3   * */
4
5  public class micro03 {
6      public static void main(String[] args) {
7          int numero;
8
9          System.out.println("Digite um número: ");
10         numero = Integer.parseInt(System.console().readLine());
11
12         if (numero >= 100) {
13             if (numero <= 200)
14                 System.out.println("O número está no intervalo entre 100 e
                       200\n");
15             else
16                 System.out.println("O número não está no intervalo entre
                      100 e 200\n");
17         } else
18             System.out.println("O número não está no intervalo entre 100 e
                   200\n");
19     }
20 }
```

**Listagem 3.49: micro03.smali**

```
1  .class public Lmicro03;
2  .super Ljava/lang/Object;
3  .source "micro03.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 5
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 9
22     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um n\u00famero: "
25
```

```
26      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
27
28      .line 10
29      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
30
31      move-result-object v0
32
33      invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
34
35      move-result-object v0
36
37      invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
38
39      move-result v0
40
41      .line 12
42      const/16 v1, 0x64
43
44      if-lt v0, v1, :cond_2b
45
46      .line 13
47      const/16 v1, 0xc8
48
49      if-gt v0, v1, :cond_23
50
51      .line 14
52      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
53
54      const-string v1, "O n\u00famero est\u00e1 no intervalo entre 100 e
            200\n"
55
56      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
57
58      .line 19
59      :goto_22
60      return-void
61
62      .line 16
63      :cond_23
64      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66      const-string v1, "O n\u00famero n\u00e3o est\u00e1 no intervalo entre
            100 e 200\n"
67
68      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
69
70      goto :goto_22
71
72      .line 18
73      :cond_2b
74      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
75
76      const-string v1, "O n\u00famero n\u00e3o est\u00e1 no intervalo entre
            100 e 200\n"
77
78      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
```

```
        String;)V
79
80      goto :goto_22
81  .end method
```

### 3.8.4 micro04

**Listagem 3.50: micro04.c**

```c
#include <stdio.h>

/*
 * Função: Ler 5 números e ao final informar quantos número(s)
 * est(á)ão no intervalo entre 10 (inclusive) e 150 (inclusive).
 * */

void main() {
  int x, num, intervalo;

  for (x=1; x<=5; x++) {
    printf("Digite um número: ");
    scanf("%d", &num);
    if (num >= 10) {
      if (num <= 150) {
        intervalo += 1;
      }
    }
  }

  printf("Ao total, foram digitados %d números no intervalo entre 10 e 150
      ", intervalo);
}
```

**Listagem 3.51: micro04.java**

```java
/*
 * Função: Ler 5 números e ao final informar quantos número(s)
 * est(á)ão no intervalo entre 10 (inclusive) e 150 (inclusive).
 * */

public class micro04 {
    public static void main(String[] args) {
        int x, num, intervalo = 0;

        for (x=1; x<=5; x++) {
            System.out.println("Digite um número: ");
            num = Integer.parseInt(System.console().readLine());
            if (num >= 10) {
                if (num <= 150) {
                    intervalo += 1;
                }
            }
        }

        System.out.println("Ao total, foram digitados " + intervalo + " nú
            meros no intervalo entre 10 e 150");
    }
}
```

**Listagem 3.52: micro04.smali**

```
1  .class public Lmicro04;
2  .super Ljava/lang/Object;
3  .source "micro04.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 6
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 8
22     const/4 v0, 0x0
23
24     .line 10
25     const/4 v1, 0x1
26
27     :goto_2
28     const/4 v2, 0x5
29
30     if-gt v1, v2, :cond_25
31
32     .line 11
33     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
34
35     const-string v3, "Digite um n\u00famero: "
36
37     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
38
39     .line 12
40     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
41
42     move-result-object v2
43
44     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
45
46     move-result-object v2
47
48     invoke-static {v2}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
49
50     move-result v2
51
52     .line 13
53     const/16 v3, 0xa
54
```

```
55      if-lt v2, v3, :cond_22
56
57      .line 14
58      const/16 v3, 0x96
59
60      if-gt v2, v3, :cond_22
61
62      .line 15
63      add-int/lit8 v0, v0, 0x1
64
65      .line 10
66      :cond_22
67      add-int/lit8 v1, v1, 0x1
68
69      goto :goto_2
70
71      .line 20
72      :cond_25
73      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
74
75      new-instance v2, Ljava/lang/StringBuilder;
76
77      invoke-direct {v2}, Ljava/lang/StringBuilder;-><init>()V
78
79      const-string v3, "Ao total, foram digitados "
80
81      invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
82
83      move-result-object v2
84
85      invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
86
87      move-result-object v0
88
89      const-string v2, " n\u00fameros no intervalo entre 10 e 150"
90
91      invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
92
93      move-result-object v0
94
95      invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
96
97      move-result-object v0
98
99      invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
100
101     .line 21
102     return-void
103 .end method
```

### 3.8.5 micro05

**Listagem 3.53: micro05.c**

```c
#include <stdio.h>

/*
 * Função : Escrever um algoritmo que leia o nome e o sexo de 56
 * pessoas e informe o nome e se ela é homem ou mulher.
 * No final informe o total de homens e de mulheres.
 */

void main() {
  int x, h, m;
  char nome[20], sexo;

  for (x=1; x<=5; x++) {
    printf("Digite o nome: ");
    fflush(stdout);
    fgets(nome, sizeof nome, stdin);
    printf("H - Homem ou M - Mulher: ");
    scanf("%c", &sexo);
    getchar();
    switch(sexo) {
      case 'H':
        h += 1;
        break;
      case 'M':
        m += 1;
        break;
      default:
        printf("Sexo só pode ser H ou M!\n");
    }
  }

  printf("Foram inseridos %d Homens\n", h);
  printf("Foram inseridos %d Mulheres\n", m);
}
```

**Listagem 3.54: micro05.java**

```java
/*
 * Função : Escrever um algoritmo que leia o nome e o sexo de 56
 * pessoas e informe o nome e se ela é homem ou mulher.
 * No final informe o total de homens e de mulheres.
 */

public class micro05 {
    public static void main(String[] args) {
        int x, h = 0, m = 0;
        String nome, sexo;

        for (x=1; x<=5; x++) {
            System.out.println("Digite o nome: ");
            nome = System.console().readLine();
            System.out.println("H - Homem ou M - Mulher: ");
            sexo = System.console().readLine();
            switch(sexo) {
                case "H":
                    h += 1;
                    break;
                case "M":
```

```
22                      m += 1;
23                      break;
24                  default:
25                      System.out.println("Sexo só pode ser H ou M!\n");
26              }
27          }
28
29          System.out.println("Foram inseridos "+h+" Homens\n");
30          System.out.println("Foram inseridos "+m+" Mulheres\n");
31      }
32  }
```

**Listagem 3.55: micro05.smali**

```
1  .class public Lmicro05;
2  .super Ljava/lang/Object;
3  .source "micro05.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 7
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 9
19
20     .prologue
21     const/4 v4, 0x1
22
23     const/4 v3, 0x0
24
25     .line 9
26     move v0, v3
27
28     move v1, v3
29
30     move v5, v4
31
32     .line 12
33     :goto_5
34     const/4 v2, 0x5
35
36     if-gt v5, v2, :cond_55
37
38     .line 13
39     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
40
41     const-string v6, "Digite o nome: "
42
43     invoke-virtual {v2, v6}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
44
```

```
45      .line 14
46      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
47
48      move-result-object v2
49
50      invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
51
52      .line 15
53      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
54
55      const-string v6, "H - Homem ou M - Mulher: "
56
57      invoke-virtual {v2, v6}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
58
59      .line 16
60      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
61
62      move-result-object v2
63
64      invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
65
66      move-result-object v6
67
68      .line 17
69      const/4 v2, -0x1
70
71      invoke-virtual {v6}, Ljava/lang/String;->hashCode()I
72
73      move-result v7
74
75      sparse-switch v7, :sswitch_data_92
76
77      :cond_2d
78      :goto_2d
79      packed-switch v2, :pswitch_data_9c
80
81      .line 25
82      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
83
84      const-string v6, "Sexo s\u00f3 pode ser H ou M!\n"
85
86      invoke-virtual {v2, v6}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
87
88      .line 12
89      :goto_37
90      add-int/lit8 v2, v5, 0x1
91
92      move v5, v2
93
94      goto :goto_5
95
96      .line 17
97      :sswitch_3b
98      const-string v7, "H"
99
100     invoke-virtual {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object
            ;)Z
```

```
101
102     move-result v6
103
104     if-eqz v6, :cond_2d
105
106     move v2, v3
107
108     goto :goto_2d
109
110     :sswitch_45
111     const-string v7, "M"
112
113     invoke-virtual {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object
            ;)Z
114
115     move-result v6
116
117     if-eqz v6, :cond_2d
118
119     move v2, v4
120
121     goto :goto_2d
122
123     .line 19
124     :pswitch_4f
125     add-int/lit8 v1, v1, 0x1
126
127     .line 20
128     goto :goto_37
129
130     .line 22
131     :pswitch_52
132     add-int/lit8 v0, v0, 0x1
133
134     .line 23
135     goto :goto_37
136
137     .line 29
138     :cond_55
139     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
140
141     new-instance v3, Ljava/lang/StringBuilder;
142
143     invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V
144
145     const-string v4, "Foram inseridos "
146
147     invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
148
149     move-result-object v3
150
151     invoke-virtual {v3, v1}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
152
153     move-result-object v1
154
155     const-string v3, " Homens\n"
156
```

```
157     invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
158
159     move-result-object v1
160
161     invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
162
163     move-result-object v1
164
165     invoke-virtual {v2, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
166
167     .line 30
168     sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
169
170     new-instance v2, Ljava/lang/StringBuilder;
171
172     invoke-direct {v2}, Ljava/lang/StringBuilder;-><init>()V
173
174     const-string v3, "Foram inseridos "
175
176     invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
177
178     move-result-object v2
179
180     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(I)Ljava/
            lang/StringBuilder;
181
182     move-result-object v0
183
184     const-string v2, " Mulheres\n"
185
186     invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
187
188     move-result-object v0
189
190     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
191
192     move-result-object v0
193
194     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
195
196     .line 31
197     return-void
198
199     .line 17
200     :sswitch_data_92
201     .sparse-switch
202         0x48 -> :sswitch_3b
203         0x4d -> :sswitch_45
204     .end sparse-switch
205
206     :pswitch_data_9c
207     .packed-switch 0x0
```

```
208         :pswitch_4f
209         :pswitch_52
210     .end packed-switch
211 .end method
```

### 3.8.6 micro06

**Listagem 3.56: micro06.c**

```c
#include <stdio.h>

/*
 * Função : Faça um algoritmo que leia um número de 1 a 5 e o
 * escreva por extenso. Caso o usuário digite um número que
 * não esteja neste intervalo, exibir mensagem: número inválido.
 * */

void main() {
  int numero;
  printf("Digite um número de 1 a 5: ");
  scanf("%d", &numero);
  switch(numero) {
    case 1:
      printf("Um\n");
      break;
    case 2:
      printf("Dois\n");
      break;
    case 3:
      printf("Três\n");
      break;
    case 4:
      printf("Quatro\n");
      break;
    case 5:
      printf("Cinco\n");
      break;
    default:
      printf("Número inválido!!!\n");
      break;
  }
}
```

**Listagem 3.57: micro06.java**

```java
/*
 * Função : Faça um algoritmo que leia um número de 1 a 5 e o
 * escreva por extenso. Caso o usuário digite um número que
 * não esteja neste intervalo, exibir mensagem: número inválido.
 * */

public class micro06{
    public static void main(String[] args) {
        int numero;
        System.out.println("Digite um número de 1 a 5: ");
        numero = Integer.parseInt(System.console().readLine());
        switch(numero) {
            case 1:
```

```
14                  System.out.println("Um\n");
15                  break;
16              case 2:
17                  System.out.println("Dois\n");
18                  break;
19              case 3:
20                  System.out.println("Três\n");
21                  break;
22              case 4:
23                  System.out.println("Quatro\n");
24                  break;
25              case 5:
26                  System.out.println("Cinco\n");
27                  break;
28              default:
29                  System.out.println("Número inválido!!!\n");
30                  break;
31          }
32
33
34      }
35  }
```

**Listagem 3.58: micro06.smali**

```
1  .class public Lmicro06;
2  .super Ljava/lang/Object;
3  .source "micro06.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 7
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 10
22     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um n\u00famero de 1 a 5: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
27
28     .line 11
29     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
30
31     move-result-object v0
32
33     invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
```

```
34
35      move-result-object v0
36
37      invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
38
39      move-result v0
40
41      .line 12
42      packed-switch v0, :pswitch_data_46
43
44      .line 29
45      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
46
47      const-string v1, "N\u00famero inv\u00e1lido!!!\n"
48
49      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
50
51      .line 34
52      :goto_1d
53      return-void
54
55      .line 14
56      :pswitch_1e
57      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
58
59      const-string v1, "Um\n"
60
61      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
62
63      goto :goto_1d
64
65      .line 17
66      :pswitch_26
67      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
68
69      const-string v1, "Dois\n"
70
71      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
72
73      goto :goto_1d
74
75      .line 20
76      :pswitch_2e
77      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
78
79      const-string v1, "Tr\u00eas\n"
80
81      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
82
83      goto :goto_1d
84
85      .line 23
86      :pswitch_36
87      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
88
```

```
89      const-string v1, "Quatro\n"
90
91      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
92
93      goto :goto_1d
94
95      .line 26
96      :pswitch_3e
97      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
98
99      const-string v1, "Cinco\n"
100
101     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
102
103     goto :goto_1d
104
105     .line 12
106     :pswitch_data_46
107     .packed-switch 0x1
108         :pswitch_1e
109         :pswitch_26
110         :pswitch_2e
111         :pswitch_36
112         :pswitch_3e
113     .end packed-switch
114 .end method
```

### 3.8.7 micro07

**Listagem 3.59: micro07.c**

```
1  #include <stdio.h>
2
3  /*
4   * Função : Faça um algoritmo que receba N números e mostre
5   * positivo, negativo ou zero para cada número.
6   * */
7
8  void main() {
9    int programa, numero;
10   char opc;
11
12   programa = 1;
13   while (programa == 1) {
14     printf("Digite um número: ");
15     scanf("%d", &numero);
16     if (numero > 0)
17       printf("Positivo\n");
18     else {
19       if (numero == 0)
20         printf("O número é igual a 0\n");
21       if (numero < 0)
22         printf("Negativo\n");
23     }
24
25     printf("Deseja finalizar? (S/N) ");
```

```
26      scanf("%c", &opc);
27      getchar();
28
29      if (opc == 'S')
30        programa = 0;
31    }
32  }
```

**Listagem 3.60: micro07.java**

```
1  /*
2   * Função : Faça um algoritmo que receba N números e mostre
3   * positivo, negativo ou zero para cada número.
4   * */
5
6  public class micro07 {
7      public static void main(String[] args) {
8          int programa, numero;
9          String opc;
10
11         programa = 1;
12         while (programa == 1) {
13             System.out.println("Digite um número: ");
14             numero = Integer.parseInt(System.console().readLine());
15             if (numero > 0)
16                 System.out.println("Positivo\n");
17             else {
18                 if (numero == 0)
19                     System.out.println("O número é igual a 0\n");
20                 if (numero < 0)
21                     System.out.println("Negativo\n");
22             }
23
24             System.out.println("Deseja finalizar? (S/N) ");
25             opc = System.console().readLine();
26
27             if (opc.equals("S"))
28                 programa = 0;
29         }
30
31     }
32 }
```

**Listagem 3.61: micro07.smali**

```
1  .class public Lmicro07;
2  .super Ljava/lang/Object;
3  .source "micro07.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 6
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
```

```
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 6
19
20     .prologue
21     const/4 v1, 0x1
22
23     .line 11
24     move v0, v1
25
26     .line 12
27     :cond_2
28     :goto_2
29     if-ne v0, v1, :cond_4c
30
31     .line 13
32     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
33
34     const-string v3, "Digite um n\u00famero: "
35
36     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
37
38     .line 14
39     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
40
41     move-result-object v2
42
43     invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
44
45     move-result-object v2
46
47     invoke-static {v2}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
48
49     move-result v2
50
51     .line 15
52     if-lez v2, :cond_39
53
54     .line 16
55     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
56
57     const-string v3, "Positivo\n"
58
59     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
60
61     .line 24
62     :cond_20
63     :goto_20
64     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66     const-string v3, "Deseja finalizar? (S/N) "
67
68     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
69
70     .line 25
```

```
71      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
72
73      move-result-object v2
74
75      invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
76
77      move-result-object v2
78
79      .line 27
80      const-string v3, "S"
81
82      invoke-virtual {v2, v3}, Ljava/lang/String;->equals(Ljava/lang/Object
            ;)Z
83
84      move-result v2
85
86      if-eqz v2, :cond_2
87
88      .line 28
89      const/4 v0, 0x0
90
91      goto :goto_2
92
93      .line 18
94      :cond_39
95      if-nez v2, :cond_42
96
97      .line 19
98      sget-object v3, Ljava/lang/System;->out:Ljava/io/PrintStream;
99
100     const-string v4, "O n\u00famero \u00e9 igual a 0\n"
101
102     invoke-virtual {v3, v4}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
103
104     .line 20
105     :cond_42
106     if-gez v2, :cond_20
107
108     .line 21
109     sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
110
111     const-string v3, "Negativo\n"
112
113     invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
114
115     goto :goto_20
116
117     .line 31
118     :cond_4c
119     return-void
120 .end method
```

### 3.8.8 micro08

Listagem 3.62: micro08.c

```c
#include <stdio.h>

void main() {
  int numero;
  numero = 1;

  while (numero != 0) {
    printf("Digite um numero: ");
    scanf("%d", &numero);

    if (numero > 10)
      printf("O numero %d é maior que 10\n");
    else
      printf("O numero %d é menor que 10\n");
  }
}
```

**Listagem 3.63: micro08.java**

```java

public class micro08 {
    public static void main(String[] args) {
        int numero;
        numero = 1;

        while (numero != 0) {
            System.out.println("Digite um numero: ");
            numero = Integer.parseInt(System.console().readLine());

            if (numero > 10)
                System.out.println("O numero %d é maior que 10\n");
            else
                System.out.println("O numero %d é menor que 10\n");
        }
    }
}
```

**Listagem 3.64: micro08.smali**

```
.class public Lmicro08;
.super Ljava/lang/Object;
.source "micro08.java"


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

    .prologue
    .line 2
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method

.method public static main([Ljava/lang/String;)V
    .registers 4

    .prologue
```

```
21      .line 5
22      const/4 v0, 0x1
23
24      .line 7
25      :goto_1
26      if-eqz v0, :cond_2a
27
28      .line 8
29      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
30
31      const-string v1, "Digite um numero: "
32
33      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
34
35      .line 9
36      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
37
38      move-result-object v0
39
40      invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
41
42      move-result-object v0
43
44      invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
45
46      move-result v0
47
48      .line 11
49      const/16 v1, 0xa
50
51      if-le v0, v1, :cond_22
52
53      .line 12
54      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
55
56      const-string v2, "O numero %d \u00e9 maior que 10\n"
57
58      invoke-virtual {v1, v2}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
59
60      goto :goto_1
61
62      .line 14
63      :cond_22
64      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
65
66      const-string v2, "O numero %d \u00e9 menor que 10\n"
67
68      invoke-virtual {v1, v2}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
69
70      goto :goto_1
71
72      .line 16
73      :cond_2a
74      return-void
75  .end method
```

### 3.8.9 micro09

**Listagem 3.65: micro09.c**

```c
#include <stdio.h>

void main() {
  float preco, venda, novo_preco;

  printf("Digite o preço: ");
  scanf("%f", &preco);
  printf("Digite a venda: ");
  scanf("%f", &venda);

  if (venda < 500 || preco < 30) {
    novo_preco = preco + 10/100 * preco;
  } else if ((venda >= 500 && venda < 1200) || (preco >= 30 && preco < 80)
     ) {
    novo_preco = preco + 15/100 * preco;
  } else if (venda >= 1200 || preco >= 80) {
    novo_preco = preco - 20/100 * preco;
  }
  printf("O novo preço é %f\n", novo_preco
}
```

**Listagem 3.66: micro09.java**

```java
public class micro09 {
    public static void main(String[] args) {
        float preco, venda, novo_preco = 0;

        System.out.println("Digite o preço: ");
        preco = Float.parseFloat(System.console().readLine());
        System.out.println("Digite a venda: ");
        venda = Float.parseFloat(System.console().readLine());

        if (venda < 500 || preco < 30) {
            novo_preco = preco + 10/100 * preco;
        } else if ((venda >= 500 && venda < 1200) || (preco >= 30 && preco
            < 80)) {
            novo_preco = preco + 15/100 * preco;
        } else if (venda >= 1200 || preco >= 80) {
            novo_preco = preco - 20/100 * preco;
        }
        System.out.println("O novo preço é "+novo_preco+"\n");

    }
}
```

**Listagem 3.67: micro09.smali**

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


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

```
8       .registers 1
9
10      .prologue
11      .line 1
12      invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14      return-void
15  .end method
16
17  .method public static main([Ljava/lang/String;)V
18      .registers 9
19
20      .prologue
21      const/high16 v7, 0x44960000    # 1200.0f
22
23      const/high16 v6, 0x43fa0000    # 500.0f
24
25      const/high16 v5, 0x42a00000    # 80.0f
26
27      const/high16 v4, 0x41f00000    # 30.0f
28
29      const/4 v0, 0x0
30
31      .line 3
32      .line 5
33      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
34
35      const-string v2, "Digite o pre\u00e7o: "
36
37      invoke-virtual {v1, v2}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
38
39      .line 6
40      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
41
42      move-result-object v1
43
44      invoke-virtual {v1}, Ljava/io/Console;->readLine()Ljava/lang/String;
45
46      move-result-object v1
47
48      invoke-static {v1}, Ljava/lang/Float;->parseFloat(Ljava/lang/String;)F
49
50      move-result v1
51
52      .line 7
53      sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;
54
55      const-string v3, "Digite a venda: "
56
57      invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
58
59      .line 8
60      invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
61
62      move-result-object v2
63
64      invoke-virtual {v2}, Ljava/io/Console;->readLine()Ljava/lang/String;
```

```
65
66      move-result-object v2
67
68      invoke-static {v2}, Ljava/lang/Float;->parseFloat(Ljava/lang/String;)F
69
70      move-result v2
71
72      .line 10
73      cmpg-float v3, v2, v6
74
75      if-ltz v3, :cond_37
76
77      cmpg-float v3, v1, v4
78
79      if-gez v3, :cond_58
80
81      .line 11
82      :cond_37
83      mul-float/2addr v0, v1
84
85      add-float/2addr v0, v1
86
87      .line 17
88      :cond_39
89      :goto_39
90      sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
91
92      new-instance v2, Ljava/lang/StringBuilder;
93
94      invoke-direct {v2}, Ljava/lang/StringBuilder;-><init>()V
95
96      const-string v3, "O novo pre\u00e7o \u00e9 "
97
98      invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
99
100     move-result-object v2
101
102     invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(F)Ljava/
            lang/StringBuilder;
103
104     move-result-object v0
105
106     const-string v2, "\n"
107
108     invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/
            String;)Ljava/lang/StringBuilder;
109
110     move-result-object v0
111
112     invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/
            String;
113
114     move-result-object v0
115
116     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
117
118     .line 19
```

```
119     return-void
120
121     .line 12
122     :cond_58
123     cmpl-float v3, v2, v6
124
125     if-ltz v3, :cond_60
126
127     cmpg-float v3, v2, v7
128
129     if-ltz v3, :cond_68
130
131     :cond_60
132     cmpl-float v3, v1, v4
133
134     if-ltz v3, :cond_6b
135
136     cmpg-float v3, v1, v5
137
138     if-gez v3, :cond_6b
139
140     .line 13
141     :cond_68
142     mul-float/2addr v0, v1
143
144     add-float/2addr v0, v1
145
146     goto :goto_39
147
148     .line 14
149     :cond_6b
150     cmpl-float v2, v2, v7
151
152     if-gez v2, :cond_73
153
154     cmpl-float v2, v1, v5
155
156     if-ltz v2, :cond_39
157
158     .line 15
159     :cond_73
160     mul-float/2addr v0, v1
161
162     sub-float v0, v1, v0
163
164     goto :goto_39
165 .end method
```

### 3.8.10 micro10

Já contido anteriormente.

### 3.8.11 micro11

**Listagem 3.68: micro11.c**

```c
#include <stdio.h>

/*
 * Função : recebe um número e verifica se o número é positivo, nulo
 * ou negativo com auxilio de uma função.
 * */

int verifica(int n) {
  int res;

  if (n > 0)
    res = 1;
  else if (n < 0)
    res = -1;
  else
    res = 0;

  return res;
}


void main() {
  int numero, x;

  printf("Digite um número: ");
  scanf("%d", &numero);

  x = verifica(numero);

  if (x == 1) printf("Número positivo\n");
  else if (x == 0) printf("Zero\n");
  else printf("Número negativo\n");
}
```

Listagem 3.69: micro11.java

```java
/*
 * Função : recebe um número e verifica se o número é positivo, nulo
 * ou negativo com auxilio de uma função.
 * */

public class micro11 {
    public static void main(String[] args) {
        int numero, x;

        System.out.println("Digite um número: ");
        numero = Integer.parseInt(System.console().readLine());

        x = verifica(numero);

        if (x == 1) System.out.println("Número positivo\n");
        else if (x == 0) System.out.println("Zero\n");
        else System.out.println("Número negativo\n");
    }

    static int verifica(int n) {
        int res;

        if (n > 0)
```

```
24              res = 1;
25          else if (n < 0)
26              res = -1;
27          else
28              res = 0;
29
30          return res;
31      }
32  }
```

**Listagem 3.70: micro11.smali**

```
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 1
9
10     .prologue
11     .line 6
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 10
22     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
23
24     const-string v1, "Digite um n\u00famero: "
25
26     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
           String;)V
27
28     .line 11
29     invoke-static {}, Ljava/lang/System;->console()Ljava/io/Console;
30
31     move-result-object v0
32
33     invoke-virtual {v0}, Ljava/io/Console;->readLine()Ljava/lang/String;
34
35     move-result-object v0
36
37     invoke-static {v0}, Ljava/lang/Integer;->parseInt(Ljava/lang/String;)I
38
39     move-result v0
40
41     .line 13
42     invoke-static {v0}, Lmicro11;->verifica(I)I
43
44     move-result v0
45
46     .line 15
```

```
47      const/4 v1, 0x1
48
49      if-ne v0, v1, :cond_22
50
51      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
52
53      const-string v1, "N\u00famero positivo\n"
54
55      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
56
57      .line 18
58      :goto_21
59      return-void
60
61      .line 16
62      :cond_22
63      if-nez v0, :cond_2c
64
65      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
66
67      const-string v1, "Zero\n"
68
69      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
70
71      goto :goto_21
72
73      .line 17
74      :cond_2c
75      sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
76
77      const-string v1, "N\u00famero negativo\n"
78
79      invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/
            String;)V
80
81      goto :goto_21
82  .end method
83
84  .method static verifica(I)I
85      .registers 2
86
87      .prologue
88      .line 23
89      if-lez p0, :cond_4
90
91      .line 24
92      const/4 v0, 0x1
93
94      .line 30
95      :goto_3
96      return v0
97
98      .line 25
99      :cond_4
100     if-gez p0, :cond_8
101
102     .line 26
```

```
103     const/4 v0, -0x1
104
105     goto :goto_3
106
107     .line 28
108     :cond_8
109     const/4 v0, 0x0
110
111     goto :goto_3
112 .end method
```

# Capítulo 4

# Analisador Léxico

Esse capítulo irá abordar como foi realizada a criação do analisador léxico para linguagem MiniC. Um programa em MiniC é uma instância positiva de um programa em C, ou seja, toda ação que for válida em MiniC também deve ser válida em C.

## 4.1 Sobre o Analisador Léxico

O analisador léxico, tem por função realizar uma *tokenização* de um código, por isso também é conhecido como *tokenizer* ou *lexer*. O analisador converte uma sequência de caractéres em um sequência de caractéres que carregam algum significado, assim é possível retirar informações maiores sobre o que foi lido. Como a nossa gramática é uma gramática livre de contexto, para definirmos qual o *token* associar ao valor presente no *buffer* de leitura não é necessário observar os *tokens* passados e nem os *tokens* que estão por vir, fazendo assim, a *tokenização* seguir somente para frente, não necessitando de informações provenientes de outros passos da análise do compilador.

Para criação do analisador léxico, foi utilizado o *ocamllex*, que cria um analisador a partir de um conjunto de expressões regulares e ações semânticas para tais regras. Possuindo o conjunto de regras e ações, o *ocamllex* cria um autômato finito determinístico para as regras passadas.

## 4.2 Comandos

Para criarmos o autômato com o *ocamllex*, utilizamos um arquivo no formato *.mll*, e executamos o seguinte comando:

```
> ocamllex nome_arquivo.mll
```

Isso irá criar um arquivo *.ml* que contém as transições da nossa máquina de estados, para utilizar o arquivo gerado, é necessário compilar, para realizar isso, basta executar:

```
> ocamlc -c nome_arquivo.ml
```

Após a execução do comando, dois novos arquivos serão criados, um *.cmi* e outro *.cmo*. Ambos são arquivos objetos, assim, para utilizar o autômato, basta importar o arquivo *.cmi*, seja em outro arquivo *.ml* ou na linha de comando do *OCaml*.

## 4.3 Tokens

Foram criados *tokens* para as palavras reservadas mais utilizadas e as que foram vistas nos códigos de exemplo, *nano* e *micro*, criando uma relação de *token* e palavra reservada, temos:

| Tipo | Representação | Tipo | Representação |
|---|---|---|---|
| APAR | ( | AND | && |
| ACHA | { | OU | \|\| |
| FCHA | } | NAO | ! |
| FPAR | ) | IGUAL | == |
| ATRIB | = | DIFE | != |
| LET | <= | LST | < |
| GET | >= | GTT | > |
| IF | if | INC | ++ |
| ELSE | else | FOR | for |
| WHILE | while | RET | return |
| FLOAT | float | INT | int |
| PTVIRG | ; | CHAR | char |
| VIRG | , | VOID | void |
| MAIS | + | LITINT | 1, 12, 123, ... |
| MENOS | - | LITSTRING | "string comum" |
| MULT | * | LITFLOAT | 0.8, .9, 1., ... |
| DIV | / | ID | identificador |
| MOD | % | EOF | end of file |

Assim, quando a máquina de estados é executada em cima de um programa válido em MiniC, as palavras e simbolos presentes na tabela serão representadas por seus respectivos *tokens*, tabulações, espaços, comentários e importações são descartados pelo analisador.

## 4.4 Código

O código utilizado para criar o analisador léxico é o seguinte:

**Listagem 4.1: analisador léxico**

```
1 {
2   open Lexing
3   open Printf
4   open Syntatic
```

```
5
6   exception Erro of string
7
8   let incr_num_linha lexbuf =
9     let pos = lexbuf.lex_curr_p in
10    lexbuf.lex_curr_p <-
11      { pos with pos_lnum = pos.pos_lnum + 1;
12          pos_bol = pos.pos_cnum
13      }
14
15 }
16
17
18 let digit = ['0' - '9']
19 let integer = digit+
20 let double = (digit+ "." digit*) | (digit+ "." digit+)
21 let leter = ['a' - 'z' 'A' - 'Z']
22 let lib = "(<|\")?\\s*" leter+ "\\s*(>|\")?"
23 let preproc = "#" [^ '\r' '\n']*
24 let identifier = leter(leter|digit|'_')*
25 let blank = [' ' '\t']+
26 let newline = ['\r' '\n']+
27 let comment = "//" [^ '\r' '\n']*
28
29
30 rule token = parse
31     blank   {token lexbuf}
32   | newline {incr_num_linha lexbuf; token lexbuf}
33   | comment {token lexbuf}
34   | preproc {token lexbuf}
35   | "/*"    {nested_comments 0 lexbuf}
36   | ","     {VIRG}
37   | ";"     {PTVIRG}
38   | "("     {APAR}
39   | "{"     {ACHA}
40   | "++"    {INC}
41   | "+"     {MAIS}
42   | "-"     {MENOS}
43   | "*"     {MULT}
44   | "/"     {DIV}
45   | "%"     {MOD}
46   | "&&"    {AND}
47   | "||"    {OU}
48   | "!="    {DIFE}
49   | "!"     {NAO}
50   | "<="    {LET}
51   | ">="    {GET}
52   | "<"     {LST}
53   | ">"     {GTT}
54   | ")"     {FPAR}
55   | "}"     {FCHA}
56   | "=="    {IGUAL}
57   | "="     {ATRIB}
58   | double as num {let numero = float_of_string num in LITFLOAT numero}
59   | integer as num {let numero = int_of_string num in LITINT numero}
60   | "void"  {VOID}
61   | "int"   {INT}
62   | "float" {FLOAT}
63   | "char"  {CHAR}
```

```
64   | "for"    {FOR}
65   | "if"     {IF}
66   | "else"   {ELSE}
67   | "while"  {WHILE}
68   | "return" {RET}
69   | identifier as id {ID id}
70   | "\""      {let pos = lexbuf.lex_curr_p in
71               let lin = pos.pos_lnum
72               and col = pos.pos_cnum - pos.pos_bol - 1 in
73               let buffer = Buffer.create 1 in
74               let str = read_string lin col buffer lexbuf in LITSTRING str}
75   | _        {raise (Erro ("Caracter desconhecido: " ^ Lexing.lexeme lexbuf
       ))}
76   | eof      {EOF}
77
78 and nested_comments n = parse
79     "*/"     {if n=0 then token lexbuf
80               else nested_comments (n-1) lexbuf}
81   | "/*"     {nested_comments (n+1) lexbuf}
82   | newline {incr_num_linha lexbuf; nested_comments n lexbuf}
83   | _        {nested_comments n lexbuf}
84   | eof      {raise (Erro "Comentario nao terminado!")}
85
86 and read_string lin col buffer = parse
87     "\""      {Buffer.contents buffer}
88   | "\\t"    {Buffer.add_char buffer '\t'; read_string lin col buffer
       lexbuf}
89   | "\\n"    {Buffer.add_char buffer '\n'; read_string lin col buffer
       lexbuf}
90   | "\\" "\""{Buffer.add_char buffer '"'; read_string lin col buffer
       lexbuf}
91   | "\\" "\\" {Buffer.add_char buffer '\\'; read_string lin col buffer
       lexbuf}
92   | _ as c   {Buffer.add_char buffer c; read_string lin col buffer lexbuf}
93   | eof      {raise (Erro "A string nao foi fechada!")}
```

Como observado, com a nossa lista de *tokens*, utilizamos as expressões regulares para ditar regras sobre as palavras reservadas em C e substituir pelo *token* referente da nossa lista. Foram criadas funções a parte para atender ao caso de *strings* e no caso de comentários aninhados.

Quando utilizamos o analisador léxico em um código de exemplo em C, obteremos uma lista de *tokens* causado pela substituição das palavras. Um exemplo da aplicação, com o seguinte código:

**Listagem 4.2: micro11.c**

```
1  #include <stdio.h>
2
3  /*
4   * Função : recebe um número e verifica se o número é positivo, nulo
5   * ou negativo com auxilio de uma função.
6   * */
7
8  int verifica(int n) {
9    int res;
10
11   if (n > 0)
```

```
12      res = 1;
13    else if (n < 0)
14      res = -1;
15    else
16      res = 0;
17
18    return res;
19  }
20
21
22  void main() {
23    int numero, x;
24
25    printf("Digite um número: ");
26    scanf("%d", &numero);
27
28    x = verifica(numero);
29
30    if (x == 1) printf("Número positivo\n");
31    else if (x == 0) printf("Zero\n");
32    else printf("Número negativo\n");
33  }
```

Vamos ter a seguinte saída do analisador:

**Listagem 4.3: lexical.mll**

```
1  [INT; ID "verifica"; APAR; INT; ID "n"; FPAR; ACHA;
2    INT; ID "res";
3    PTVIRG; IF; APAR; ID "n"; GTT; LITINT 0; FPAR;
4      ID "res"; ATRIB; LITINT 1; PTVIRG;
5    ELSE; IF; APAR; ID "n"; LST; LITINT 0; FPAR;
6       ID "res"; ATRIB; MENOS; LITINT 1; PTVIRG;
7    ELSE;
8      ID "res"; ATRIB; LITINT 0; PTVIRG;
9
10   RET; ID "res";
11 PTVIRG; FCHA;
12
13 VOID; ID "main"; APAR; FPAR; ACHA;
14   INT; ID "numero"; VIRG; ID "x"; PTVIRG;
15
16   ID "printf"; APAR; LITSTRING "Digite um número: "; FPAR; PTVIRG;
17   ID "scanf"; APAR; LITSTRING "%d"; VIRG; ECOM; ID "numero"; FPAR; PTVIRG;
18
19   ID "x"; ATRIB; ID "verifica"; APAR; ID "numero"; FPAR; PTVIRG;
20
21   IF; APAR; ID "x"; IGUAL; LITINT 1; FPAR;
22     ID "printf"; APAR; LITSTRING "Número positivo\n"; FPAR; PTVIRG;
23   ELSE; IF; APAR; ID "x"; IGUAL; LITINT 0; FPAR;
24     ID "printf"; APAR; LITSTRING "Zero\n"; FPAR; PTVIRG;
25   ELSE;
26     ID "printf"; APAR; LITSTRING "Número negativo\n"; FPAR; PTVIRG;
27 FCHA;
28 EOF]
```

Utilizando a nossa tabela, é possível verificar que obtivemos a saída desejada. Descartamos todos espaços em branco, comentários e tabulações. Demarcamos parênteses, chaves,

funções, literais, etc. Com o analisador funcional é possível dar o próximo passo para criação do compilador de *miniC* para *Dalvik*.

# Capítulo 5

# Analisador Sintático

O analisador sintático tem como entrada uma *stream* de *tokens* provenientes da análise léxica, onde o analisador sintático deve verificar se a sequência de *tokens* respeitam a sintaxe da gramática, caso um código possua algum erro, deve ser rejeitado. Com uma gramática livre de contexto, a análise sintática é interpretada por um autômato de pilha.

## 5.1 Sobre o Analisador Sintático

O analisador sintático utiliza um *parser*, que terá a *stream* de entrada proveniente da análise léxica e irá produzir uma estrutura de acordo com as regras fornecidas. A estrutura montada pelo *parser* é denominada *abstract syntax tree*(AST), que cria uma representação estrutural da gramática.

### 5.1.1 Parser

Existem diferentes tipos de *parsers*, alguns mais simples e rápidos conseguem se adequar para gramáticas mais simples, como, por exemplo, o *parser* LL(1). Para o presente trabalho, foi necessário a utilização do *parser* LR(1), que é mais robusto e que consegue representar as regras da gramática corretamente.

O *parser* LR(1), foi criado pelo criador do próprio TEX, Donald Knuth. O LR(1) é um autômato finito determinístico, dessa maneira, ele possui estados e transições estáticas, que são construídas a partir das regras da gramática. O *parser* utiliza o que é chamado de *"parsing tables"* e utiliza um termo de *lookahead*, que significa que uma transição só irá ocorrer se o símbolo terminal for o mesmo do *lookahead*, o que torna a expressividade da gramática maior.

$$A \rightarrow B\, a$$

Pelo exemplo 5.1.1, a transição de A para B só pode acontecer caso o símbolo terminal

for *a*. O *lookahead* é definido por 5.1.1. A linguagem *miniC* pode ser representada por um *parser* LR(1).

$$\begin{gathered}\text{Para qualquer item } (A \to \alpha \cdot X\beta, z) \text{ no estado,} \\ \text{para toda produção } X \to \gamma, \\ \text{para todo } w \,\epsilon\, FIRST(\beta z), \\ \text{inclua } (X \to \cdot\gamma, w)\end{gathered}$$

Desse maneira, temos um conjunto de símbolos em que podemos fazer determinadas reduções nas regras da gramática.

## 5.2 Código

O código foi criado utilizado a biblioteca *menhir* do *OCaml*, e o código é definido abaixo:

**Listagem 5.1: Analisador sintático**

```
%{
    open Ast;;
%}

%token <int> LITINT
%token <string> LITSTRING
%token <float> LITFLOAT
%token INT FLOAT CHAR VOID
%token ATRIB
%token APAR ACHA FPAR FCHA
%token MAIS MENOS MULT DIV MOD
%token PTVIRG VIRG
%token GTT LST NAO
%token IGUAL DIFE LET GET
%token INC OU AND
%token FOR IF ELSE WHILE
%token RET
%token <string> ID
%token EOF


%start program
%type <Ast.programa> program

%%

program: p = declarations
    EOF { p }

declarations: { [] }
    | declare declarations { $1 :: $2 }

declare: v = decl_fun { v }

decl_fun: decl_fun_int { $1 }
    | decl_fun_float { $1 }
```

```
37      | decl_fun_char { $1 }
38      | decl_fun_void { $1 }
39
40  decl_fun_int: INT ID APAR arguments FPAR ACHA
41          commands
42          RET expression PTVIRG
43      FCHA { Funcao(TInt, ExpVar $2, $4, $7, Some $9)}
44
45  decl_fun_float: FLOAT ID APAR arguments FPAR ACHA
46          commands
47          RET expression PTVIRG
48      FCHA { Funcao(TFloat, ExpVar $2, $4, $7, Some $9) }
49
50  decl_fun_char: CHAR ID APAR arguments FPAR ACHA
51          commands
52          RET expression PTVIRG
53      FCHA { Funcao(TChar, ExpVar $2, $4, $7, Some $9) }
54
55  decl_fun_void: VOID ID APAR arguments FPAR ACHA
56          commands
57      FCHA { Funcao(TVoid, ExpVar $2, $4, $7, None) }
58
59  arguments: { [] }
60      | a=seq { a }
61
62  seq: a=argument { [a] }
63      | s=seq VIRG a=argument { s @ [a] }
64
65  argument: types ID { CmdDec(ExpVar $2, $1, None) }
66
67  commands: { [] }
68      | command PTVIRG commands { $1 :: $3 }
69      | command commands { $1 :: $2 }
70
71  command:
72      | cmd_attr PTVIRG   { $1 }
73      | cmd_dec  PTVIRG   { $1 }
74      | cmd_for           { $1 }
75      | cmd_while         { $1 }
76      | cmd_if            { $1 }
77      | cmd_incr          { $1 }
78
79  cmd_attr: ID ATRIB expression { CmdAtrib (ExpVar $1, $3) }
80
81  cmd_dec: INT ID initial { CmdDec ( ExpVar $2, TInt, $3 ) }
82      | FLOAT ID initial { CmdDec ( ExpVar $2, TFloat, $3 ) }
83      | CHAR ID initial { CmdDec ( ExpVar $2, TChar, $3 ) }
84
85  initial: { None }
86      | ATRIB expression { Some($2) }
87
88  cmd_if: IF APAR expression FPAR ACHA
89          commands
90      FCHA cmd_else { CmdIf($3, $6, $8) }
91
92  cmd_else: { None }
93      | ELSE ACHA c=commands FCHA { Some(c) }
94      | ELSE cmd_if   { Some([$2]) }
95
```

```
96 cmd_while: WHILE APAR expression FPAR ACHA
97             commands
98         FCHA { CmdWhile($3,$6) }
99
100 cmd_for: FOR APAR cmd_attr PTVIRG expression PTVIRG command FPAR ACHA
101         commands
102     FCHA { CmdFor($3, $5, $7, $10) }
103
104 cmd_incr: ID INC { CmdIncr(ExpVar $1) }
105
106 expression: e=expression AND e1=expr1 { ExpBin(AND, e, e1) }
107     | e=expression OU e1=expr1 { ExpBin(OU, e, e1) }
108     | e1=expr1 { e1 }
109
110 expr1: e1=expr1 IGUAL e2=expr2 { ExpBin(IGUAL, e1, e2) }
111     | e1=expr1 DIFE e2=expr2 { ExpBin(DIFE, e1, e2) }
112     | e2=expr2 { e2 }
113
114 expr2: e2=expr2 GTT e3=expr3 { ExpBin(GTT, e2, e3) }
115     | e2=expr2 LST e3=expr3 { ExpBin(LST, e2, e3) }
116     | e2=expr2 GET e3=expr3 { ExpBin(GET, e2, e3) }
117     | e2=expr2 LET e3=expr3 { ExpBin(LET, e2, e3) }
118     | e3=expr3 { e3 }
119
120 expr3: e3=expr3 MAIS e4=expr4 { ExpBin(MAIS, e3, e4) }
121     | e3=expr3 MENOS e4=expr4 { ExpBin(MENOS, e3, e4) }
122     | e4=expr4 { e4 }
123
124 expr4: e4=expr4 MULT e5=expr5 { ExpBin(MULT, e4, e5) }
125     | e4=expr4 DIV e5=expr5 { ExpBin(DIV, e4, e5) }
126     | e4=expr4 MOD e5=expr5 { ExpBin(MOD, e4, e5) }
127     | e5=expr5 { e5 }
128
129 expr5: NAO e5=expr5 { ExpUn(NAO, e5) }
130     | MENOS e6=expr6 { ExpUn(MENOS, e6) }
131     | e6=expr6 { e6 }
132
133 expr6: LITINT               { ExpInt $1 }
134     | LITFLOAT              { ExpFloat $1 }
135     | LITSTRING             { ExpString $1 }
136     | ID                    { ExpVar $1 }
137     | APAR expression FPAR  { $2 }
138     | call_func             { $1 }
139
140 call_func: ID APAR a=args FPAR { ChamaFunc(ExpVar $1, a) }
141
142 args: { [] }
143     | seqs { $1 }
144
145 seqs: expression { [$1] }
146     | seqs VIRG expression { $1 @ [$3] }
147
148 types:
149     | INT { TInt }
150     | FLOAT { TFloat }
151     | CHAR { TChar }
152     | LITSTRING { TString }
153     | VOID { TVoid }
```

Como citado, a estrutura utiliza o 5.1, que é definido da seguinte maneira:

**Listagem 5.2: AST**

```
type programa = funcoes

and funcoes = funcao list
and funcao = Funcao of tipo * expressao * comando list * comandos *
    expressao option

and comandos = comando list
and comando = CmdAtrib of expressao * expressao
  | CmdDec of expressao * tipo * expressao option
  | CmdWhile of expressao * comandos
  | CmdFor of comando * expressao * comando * comandos
  | CmdIf of expressao * comandos * comandos option
  | CmdIncr of expressao

and expressao = ExpInt of int
    | ExpVar of string
    | ExpFloat of float
    | ExpChar of char
    | ExpString of string
    | ExpBin of operador * expressao * expressao
    | ExpUn of operador * expressao
    | ChamaFunc of expressao * expressao list

and expr = {
                valor: expressao;
                mutable tipoexp: tipo option
            }

and operador = MAIS | MENOS | MULT | DIV | MOD |
               GTT | LST | GET | LET |
               IGUAL | DIFE | OU | AND | NAO

and tipo = TInt | TFloat | TChar | TString | TVoid

type tvalor = VInt of int | VFloat of float | VString of string |
        VChar of char

type info = {
        tipos: tipo;
        inicializada: bool;
        valor: tvalor option;
        mutable endereco: int option
    }
```

Para realizar o teste de funcionamento, foi criado um arquivo para definir as funções para load de arquivos de teste, segue código:

**Listagem 5.3: sintaticoTest.ml**

```
open Printf
open Lexing

open Ast
open ErroSynt (* nome do módulo contendo as mensagens de erro *)

```

```
7  exception Erro_Sintatico of string
8
9  module S = MenhirLib.General (* Streams *)
10 module I = Syntatic.MenhirInterpreter
11
12 let posicao lexbuf =
13     let pos = lexbuf.lex_curr_p in
14     let lin = pos.pos_lnum
15     and col = pos.pos_cnum - pos.pos_bol - 1 in
16     sprintf "linha %d, coluna %d" lin col
17
18 (* [pilha checkpoint] extrai a pilha do autômato LR(1) contida em
       checkpoint *)
19
20 let pilha checkpoint =
21   match checkpoint with
22   | I.HandlingError amb -> I.stack amb
23   | _ -> assert false (* Isso não pode acontecer *)
24
25 let estado checkpoint : int =
26   match Lazy.force (pilha checkpoint) with
27   | S.Nil -> (* O parser está no estado inicial *)
28      0
29   | S.Cons (I.Element (s, _, _, _), _) ->
30      I.number s
31
32 let sucesso v = Some v
33
34 let falha lexbuf (checkpoint : Ast.programa I.checkpoint) =
35   let estado_atual = estado checkpoint in
36   let msg = message estado_atual in
37   raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n"
38                                         (Lexing.lexeme_start lexbuf) msg))
39
40 let loop lexbuf resultado =
41   let fornecedor = I.lexer_lexbuf_to_supplier Lexical.token lexbuf in
42   I.loop_handle sucesso (falha lexbuf) fornecedor resultado
43
44
45
46 let parse_com_erro lexbuf =
47   try
48     Some (loop lexbuf (Syntatic.Incremental.program lexbuf.lex_curr_p))
49   with
50   | Lexical.Erro msg ->
51      printf "Erro lexico na %s:\n\t%s\n" (posicao lexbuf) msg;
52      None
53   | Erro_Sintatico msg ->
54      printf "Erro sintático na %s %s\n" (posicao lexbuf) msg;
55      None
56
57 let parse s =
58   let lexbuf = Lexing.from_string s in
59   let ast = parse_com_erro lexbuf in
60   ast
61
62 let parse_arq nome =
63   let ic = open_in nome in
64   let lexbuf = Lexing.from_channel ic in
```

```
65   let ast = parse_com_erro lexbuf in
66   let _ = close_in ic in
67   ast
68
69
70 (* Para compilar:
71       menhir -v --list-errors sintatico.mly > sintatico.msg
72       menhir -v --list-errors sintatico.mly --compile-errors sintatico.msg
             > erroSint.ml
73       ocamlbuild -use-menhir sintaticoTest.byte
74  *)
```

Para rodar o programa, é necessário primeramente compilar os códigos, para isso, execute a seguinte sequencia de códigos

```
> menhir -v --list-errors syntatic.mly > syntatic.msg
```

Com o arquivo *.msg*, é possível definir quais serão as mensagens de erro que serão expostas quando o erro ocorrer.

```
> menhir syntatic.mly --compile-errors syntatic.msg > erroSynt.ml
> ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
    menhirLib sintaticoTest.byte
```

Para isso, é necessário que o arquivo do analisador léxico deve estar disponível no mesmo diretório. Após realizar os comandos, crie o arquivo *.ocamlinit* para carregar todas as dependências, o arquivo é definido dessa maneira:

**Listagem 5.4: .ocamlinit**

```
1 #use "topfind";;
2 #require "menhirLib";;
3 #directory "_build";;
4 #load "erroSynt.cmo";;
5 #load "syntatic.cmo";;
6 #load "lexical.cmo";;
7 #load "ast.cmo";;
8 #load "sintaticoTest.cmo";;
9 open Ast
10 open SintaticoTest
```

Com o arquivo definido, basta executar o *OCaml* e executar a função definida no arquivo *sintaticoTest.ml* para carregar um arquivo para análise léxica.

# Capítulo 6

# Analisador semântico

Esse capítulo trata da criação do analisador semântico para a linguagem, algumas regras básicas foram definidas na linguagem para o funcionamento do analisador, assim como alterações nos analisadores passados, onde tudo será explicado.

## 6.1 Analise semântica

A analise semântica é o terceiro passo na compilação de um programa, esse passo será reservado para realizar verificações de tipos, inferências sobre tipos não definidos e declarações de funções e variáveis.

Checagem de tipos é o processo de verificação se cada instrução do programa respeita o tipo definido na linguagem. Desta maneira, verifica os tipos aplicados em operandos na execução de comandos, não permitindo assim a aplicação de um operando de soma utilizando um *int* e um *float*, por exemplo. Algumas linguagens tem essa flexibilidade, mas não é o caso do *miniC*.

Checagem para declarações avalia se a variável ou função a ser utilizada foi declarada posteriormente ao uso, assim não permitindo o uso não declarado. Caso a utilização ocorra em um escopo interno a um contexto, será buscado recursivamente entre os ambientes disponíveis a definição da variável.

Para armazenar as declarações realizadas e o tipo de cada uma, são criados ambientes. Tendo um ambiente, caso aconteça uma chamada de função, é então utilizado um novo ambiente, embutido no ambiente de escopo maior. Assim, cada definição realizada, é salva nessa ambiente, e após realizada a associação a um tipo, não é possível realizar atribuições com valores de tipos diferentes.

### 6.1.1 Regras de inferência

Existem algumas regras para inferência de tipos, onde a expressão é inferida para se avaliar verdadeira:

- $\Gamma$: ambiente
- $\Gamma + e : t$: dado o ambiente $\Gamma$ a expressão $e$ possui tipo $t$
- $\frac{\Gamma + a : t_1}{\Gamma + b : t_2}$: No ambiente $\Gamma$, $a$ possui tipo $t_1$ e $b$ possui tipo $t_2$

Para a linguagem *miniC* podemos seguir as seguintes regras:

- $\frac{\Gamma + a : t \quad \Gamma + b : t}{\Gamma + (a\ op\ b) : t}$: sendo $t$=*int, float* e $op$=+, -, *, /, %, <, >, <=, >=
- $\frac{\Gamma + a : t \quad \Gamma + b : t}{\Gamma + (a\ op\ b) : t}$: sendo $t$=*int, float, string, char* e $op$===, !=

Por essas regras notamos que operadores aritmeticos e relacionais, somente são aplicados quando ambos valores são do mesmo tipo.

## 6.2 Alterações nos analisadores passados

Foram necessárias alterações nos analisadores léxico e sintático para o perfeito funcionamento com o analisador semântico. Tais alterações foram necessárias para a captura da posição no *buffer* para caso algum erro ocorrer e para captura dos tipos declarados para as funções, variáveis e literais definidos no código do programa em *miniC*. Cada tópico irá cobrir as alterações realizadas.

### 6.2.1 Regras

As regras foram introduzidas para faciltar a captura de variáveis e conseguir definir o escopo de cada uma dessas, tais regras existem em versões mais antigas da linguagem C. Agora, é necessário:

- **Declarações** quando for declarar variáveis, devem estar logo no "topo", antes do inicio do programa em si;
- **Atribuição** não é possível declarar e atribuir o valor para uma variável simutaneamente;
- **Operadores** operadores lógicos, aritmeticos passam a funcionar somente com valores do mesmo tipo.

Seguindo essas regras, um exemplo que traz essas regras em utilização:

Listagem 6.1: lexico.mll

```
1 void main() {
2     int i, j, k; // Modo correto de definir variaveis
3     float n = 1.0; // Modo errado de definir variaveis
4 
5     i = 1 + 1; // Operador aplicado corretamente
6     j = 1 + n; // Como n é float, operador aplicado em diferentes tipos,
          ERRO
7 
8     int l; // Local errado de definir variaveis
9 }
```

Caso aconteça alguma infração nessas regras, um erro sintático será lançado.

### 6.2.2 Léxico

No analisador léxico, foi criada uma nova função para definição da posição atual no *buffer* de leitura do *OCaml*. Foram também definidos construtores para os *tokens* da linguagem, onde agora é passada a posição do *buffer* quando um dos *tokens* é encontrado na leitura do arquivo. Os valores de literais passam dois parâmetros no seu construtor, sendo eles o valor em si do literal, - seja ele *char, string, float* ou *int* - e também a posição do *buffer*.
Com essas alterações, o código do analisador léxico resultou em:

Listagem 6.2: lexico.mll

```
1 {
2   open Lexing
3   open Printf
4   open Sintatico
5 
6   exception Erro of string
7 
8   let incr_num_linha lexbuf =
9     let pos = lexbuf.lex_curr_p in
10    lexbuf.lex_curr_p <-
11      { pos with pos_lnum = pos.pos_lnum + 1;
12                 pos_bol = pos.pos_cnum
13      }
14 
15  let pos_atual lexbuf = lexbuf.lex_start_p
16 }
17 
18 
19 let digit = ['0' - '9']
20 let integer = digit+
21 let double = (digit+ "." digit*) | (digit+ "." digit+)
22 let leter = ['a' - 'z' 'A' - 'Z']
23 let lib = "(<|\")?\\s*" leter+ "\\s*(>|\")?"
24 let preproc = "#" [^ '\r' '\n']*
25 let identifier = leter(leter|digit|'_')*
26 let blank = [' ' '\t']+
27 let newline = ['\r' '\n']+
28 let comment = "//" [^ '\r' '\n']*
29 
```

```
30
31 rule token = parse
32     blank   {token lexbuf}
33   | newline {incr_num_linha lexbuf; token lexbuf}
34   | comment {token lexbuf}
35   | preproc {token lexbuf}
36   | "/*"    {nested_comments 0 lexbuf}
37   | ","     {VIRG (pos_atual lexbuf)}
38   | ";"     {PTVIRG (pos_atual lexbuf)}
39   | "("     {APAR (pos_atual lexbuf)}
40   | "{"     {ACHA (pos_atual lexbuf)}
41   | "+"     {MAIS (pos_atual lexbuf)}
42   | "-"     {MENOS (pos_atual lexbuf)}
43   | "*"     {MULT (pos_atual lexbuf)}
44   | "/"     {DIV (pos_atual lexbuf)}
45   | "%"     {MOD (pos_atual lexbuf)}
46   | "&&"    {AND (pos_atual lexbuf)}
47   | "||"    {OU (pos_atual lexbuf)}
48   | "!="    {DIFE (pos_atual lexbuf)}
49   | "!"     {NAO (pos_atual lexbuf)}
50   | "<="    {LET (pos_atual lexbuf)}
51   | ">="    {GET (pos_atual lexbuf)}
52   | "<"     {LST (pos_atual lexbuf)}
53   | ">"     {GTT (pos_atual lexbuf)}
54   | ")"     {FPAR (pos_atual lexbuf)}
55   | "}"     {FCHA (pos_atual lexbuf)}
56   | "=="    {IGUAL (pos_atual lexbuf)}
57   | "="     {ATRIB (pos_atual lexbuf)}
58   | double as num { LITFLOAT (float_of_string num, pos_atual lexbuf) }
59   | integer as num { LITINT (int_of_string num, pos_atual lexbuf) }
60   | "void"  {VOID (pos_atual lexbuf)}
61   | "int"   {INT (pos_atual lexbuf)}
62   | "float" {FLOAT (pos_atual lexbuf)}
63   | "char"  {CHAR (pos_atual lexbuf)}
64   | "for"   {FOR (pos_atual lexbuf)}
65   | "if"    {IF (pos_atual lexbuf)}
66   | "else"  {ELSE (pos_atual lexbuf)}
67   | "while" {WHILE (pos_atual lexbuf)}
68   | "return" {RET (pos_atual lexbuf)}
69   | "printf" { SAIDA (pos_atual lexbuf)}
70   | "scanf" { ENTRADA (pos_atual lexbuf)}
71   | identifier as id {ID (id, pos_atual lexbuf)}
72   | "\""     { let buffer = Buffer.create 1 in
73                let str = read_string buffer lexbuf in
74                LITSTRING (str, pos_atual lexbuf) }
75   | _        {raise (Erro ("Caracter desconhecido: " ^ Lexing.lexeme lexbuf
       ))}
76   | eof      {EOF}
77
78 and nested_comments n = parse
79   "*/"    { if n=0 then token lexbuf
80             else nested_comments (n-1) lexbuf }
81   | "/*"    { nested_comments (n+1) lexbuf }
82   | _       { nested_comments n lexbuf }
83   | eof     { raise (Erro "Comentário não terminado") }
84
85 and read_string buffer = parse
86   '"'       { Buffer.contents buffer}
87   | "\\t"     { Buffer.add_char buffer '\t'; read_string buffer lexbuf }
```

```
88    | "\\n"     { Buffer.add_char buffer '\n'; read_string buffer lexbuf }
89    | '\\' '"'  { Buffer.add_char buffer '"'; read_string buffer lexbuf }
90    | '\\' '\\' { Buffer.add_char buffer '\\'; read_string buffer lexbuf }
91    | _ as c    { Buffer.add_char buffer c; read_string buffer lexbuf }
92    | eof       { raise (Erro "A string não foi terminada") }
```

### 6.2.3 Sintático

Com as alterações realizadas no analisador sintático, regras foram incluidas na linguagem, regras essas que devem ser seguidas para obter um programa lexicalmente correto em *miniC*. Além disso, foram introduzidas as alterações necessárias para captura das declarações realizadas no código, e também a captura da posição que se encontram as declarações. O novo analisador sintático é o seguinte:

**Listagem 6.3: sintatico.mly**

```
1  %{
2      open Ast
3      open Lexing
4      open Sast
5  %}
6
7  %token <int * Lexing.position> LITINT
8  %token <string * Lexing.position> LITSTRING
9  %token <float * Lexing.position> LITFLOAT
10 %token <string * Lexing.position> ID
11 %token <Lexing.position> INT FLOAT CHAR VOID
12 %token <Lexing.position> ATRIB
13 %token <Lexing.position> APAR ACHA FPAR FCHA
14 %token <Lexing.position> MAIS MENOS MULT DIV MOD
15 %token <Lexing.position> ENTRADA SAIDA
16 %token <Lexing.position> PTVIRG VIRG
17 %token <Lexing.position> GTT LST
18 %token <Lexing.position> IGUAL DIFE LET GET
19 %token <Lexing.position> OU AND NAO
20 %token <Lexing.position> FOR IF ELSE WHILE
21 %token <Lexing.position> RET
22 %token EOF
23
24 %left OU
25 %left AND
26 %left IGUAL DIFE NAO
27 %left GTT LST
28 %left GET LET
29 %left MAIS MENOS
30 %left MULT DIV MOD
31
32 %start <Sast.expressao Ast.programa> program
33
34 %%
35
36 program: p = decl_fun* c = cmd_func*
37     EOF { Programa (List.flatten [], p, c) }
38
```

```
39 decl_fun: tipo=types nome=ID APAR formais=separated_list(VIRG, argument)
     FPAR ACHA
40         locais=cmd_dec*
41         corpo=command*
42     FCHA {
43       Funcao {
44         fn_nome = nome;
45         fn_tiporet = tipo;
46         fn_formais = formais;
47         fn_locais = List.flatten locais;
48         fn_corpo = corpo;
49       }
50     }
51
52 argument: types_non_void ID { ($2, $1) }
53
54 command:
55     | cmd_attr PTVIRG   { $1 }
56     | cmd_for           { $1 }
57     | cmd_while         { $1 }
58     | cmd_if            { $1 }
59     | comando_entrada   { $1 }
60     | comando_saida     { $1 }
61     | cmd_func          { $1 }
62     | cmd_ret           { $1 }
63
64 cmd_attr: ID ATRIB expression { CmdAtrib (ExpVar $1, $3) }
65
66 cmd_dec:
67   t = types_non_void ids = separated_nonempty_list(VIRG, ID) PTVIRG {
68     List.map (fun id -> CmdDec (id,t)) ids
69   }
70
71 cmd_if: IF APAR expression FPAR ACHA
72         command*
73     FCHA cmd_else { CmdIf($3, $6, $8) }
74
75 cmd_else: { None }
76     | ELSE ACHA c=command* FCHA { Some(c) }
77     | ELSE cmd_if   { Some([$2]) }
78
79 cmd_while: WHILE APAR expression FPAR ACHA
80             command*
81         FCHA { CmdWhile($3,$6) }
82
83 cmd_for: FOR APAR cmd_attr PTVIRG expression PTVIRG cmd_attr FPAR ACHA
84         command*
85     FCHA { CmdFor($3, $5, $7, $10) }
86
87 cmd_func: call PTVIRG { CmdChamada($1) }
88
89 comando_entrada: ENTRADA xs=separated_nonempty_list(VIRG, expression)
     PTVIRG {
90                     CmdEntrada xs
91                }
92
93 comando_saida: SAIDA xs=separated_nonempty_list(VIRG, expression) PTVIRG {
94                   CmdSaida xs
95          }
```

```
96
97 call: ID APAR args=separated_list(VIRG, expression) FPAR {
98   ExpChamada ($1, args)
99 }
100
101 cmd_ret: RET expression? PTVIRG { CmdRetorno $2 }
102
103 expression: LITINT          { ExpInt $1 }
104     | LITFLOAT              { ExpFloat $1 }
105     | LITSTRING             { ExpString $1 }
106     | ID                    { ExpVar $1 }
107     | APAR expression FPAR  { $2 }
108     | call                  { $1 }
109     | op=oper e=expression  { ExpUn (op, e) }
110     | e=expression op=oper e1=expression { ExpBin(op, e, e1) }
111
112 %inline oper:
113     | pos = MAIS   { (Mais, pos)  }
114     | pos = MENOS  { (Menos, pos) }
115     | pos = MULT   { (Mult, pos)  }
116     | pos = DIV    { (Div, pos)   }
117     | pos = MOD    { (Mod, pos)   }
118     | pos = GTT    { (Maior, pos)   }
119     | pos = LST    { (Menor, pos)   }
120     | pos = GET    { (MaiorI, pos)   }
121     | pos = LET    { (MenorI, pos)   }
122     | pos = IGUAL  { (Igual, pos) }
123     | pos = DIFE   { (Difer, pos)  }
124     | pos = NAO    { (Nao, pos)  }
125     | pos = OU     { (Ou, pos)    }
126     | pos = AND    { (E, pos)   }
127
128 types:
129     | INT { TInt }
130     | FLOAT { TFloat }
131     | CHAR { TChar }
132     | LITSTRING { TString }
133     | VOID { TVoid }
134
135 types_non_void:
136     | INT { TInt }
137     | FLOAT { TFloat }
138     | CHAR { TChar }
139     | LITSTRING { TString }
```

Para atender as novas funcionalidades foi necessário a alterações da árvore sintática abstrata($AST$), assim como a criação de duas outras estruturas baseadas na $AST$, sendo elas uma árvore semântica($SAST$) e uma árvore tipada($TAST$), que seguem respectivamente nas listagens a seguir:

**Listagem 6.4: AST**

```
1 open Lexing
2
3 type ident = string
4 type 'a pos =  'a * Lexing.position
5
```

```
type 'expr programa = Programa of declaracoes * ('expr funcao) list * ('
    expr comandos)
and 'expr funcoes = ('expr funcao) list
and declaracoes = declaracao list

and declaracao = CmdDec of (ident pos) * tipo
and 'expr funcao = Funcao of ('expr decfn)
and 'expr decfn = {
  fn_nome:    ident pos;
  fn_tiporet: tipo;
  fn_formais: (ident pos * tipo) list;
  fn_locais:  declaracoes;
  fn_corpo:   'expr comandos;
}

and 'expr comandos = ('expr comando) list
and 'expr comando =
  | CmdRetorno of 'expr option
  | CmdIf of 'expr * ('expr comandos) * ('expr comandos option)
  | CmdAtrib of 'expr * 'expr
  | CmdWhile of 'expr * 'expr comandos
  | CmdFor of 'expr comando * 'expr * 'expr comando * 'expr comandos
  | CmdEntrada of ('expr expressoes)
  | CmdSaida of ('expr expressoes)
  | CmdChamada of 'expr

and 'expr variaveis = ('expr variavel) list
and 'expr variavel =
  | VarSimples of ident pos
and 'expr expressoes = 'expr list

and tipo = TInt | TFloat | TChar | TString | TVoid

and oper = Mais | Menos | Mult | Div | Mod |
           Maior | Menor | MaiorI | MenorI |
           Igual | Difer | Ou | E | Nao
```

**Listagem 6.5: SAST**

```
open Ast

type expressao = ExpInt of int pos
    | ExpVar of ident pos
    | ExpFloat of float pos
    | ExpChar of char pos
    | ExpString of string pos
    | ExpUn of (oper pos) * expressao
    | ExpBin of (oper pos) * expressao * expressao
    | ExpChamada of ident pos * (expressao expressoes)
```

**Listagem 6.6: TAST**

```
open Ast

type expressao = ExpInt of int * tipo
    | ExpVar of (expressao variavel) * tipo
    | ExpFloat of float * tipo
    | ExpChar of char * tipo
    | ExpString of string * tipo
```

```
8      | ExpBool of bool * tipo
9      | ExpVoid
10     | ExpUn of (oper * tipo) * (expressao * tipo)
11     | ExpBin of (oper * tipo) * (expressao * tipo) * (expressao * tipo)
12     | ChamaFunc of ident * (expressao list) * tipo
```

### 6.2.4 Semântico

Após citadas as alterações nos arquivos e as algumas estruturas dependentes, temos por fim o analisador semântico.

**Listagem 6.7: Definição do analisador**

```
1 val semantico : (Sast.expressao Ast.programa) -> Tast.expressao Ast.
      programa * Ambiente.t
```

**Listagem 6.8: Implementação do analisador**

```
1  module Amb = Ambiente
2  module A = Ast
3  module S = Sast
4  module T = Tast
5
6  let rec posicao exp = let open S in
7    match exp with
8    | ExpInt (_,pos) -> pos
9    | ExpVar (_,pos) -> pos
10   | ExpFloat (_,pos) -> pos
11   | ExpChar (_,pos) -> pos
12   | ExpString  (_,pos) -> pos
13   | ExpUn ((_,pos),_) -> pos
14   | ExpBin ((_,pos),_,_)  -> pos
15   | ExpChamada ((_,pos), _) -> pos
16
17 type classe_op = Aritmetico | Relacional | Unario
18
19 let classifica op =
20   let open A in
21   match op with
22   | Nao -> Unario
23   | Ou
24   | E
25   | Menor
26   | Maior
27   | MaiorI
28   | MenorI
29   | Igual
30   | Difer -> Relacional
31   | Mais
32   | Menos
33   | Mult
34   | Div
35   | Mod -> Aritmetico
36
37 let msg_erro_pos pos msg =
38   let open Lexing in
```

```
39    let lin = pos.pos_lnum
40    and col = pos.pos_cnum - pos.pos_bol - 1 in
41    Printf.sprintf "Semantico -> linha %d, coluna %d: %s" lin col msg
42
43  let msg_erro nome msg =
44    let pos = snd nome in
45    msg_erro_pos pos msg
46
47  let nome_tipo t =
48    let open A in
49      match t with
50        TInt -> "inteiro"
51      | TFloat -> "float"
52      | TChar -> "char"
53      | TString -> "string"
54      | TVoid -> "void"
55
56  let mesmo_tipo pos msg tinf tdec =
57    if tinf <> tdec
58    then
59      let msg = Printf.sprintf msg (nome_tipo tinf) (nome_tipo tdec) in
60      failwith (msg_erro_pos pos msg)
61
62  let rec infere_exp amb exp =
63    match exp with
64      S.ExpInt n    -> (T.ExpInt (fst n, A.TInt),       A.TInt)
65    | S.ExpString s -> (T.ExpString (fst s, A.TString), A.TString)
66    | S.ExpFloat f -> (T.ExpFloat (fst f, A.TFloat), A.TFloat)
67    | S.ExpChar c -> (T.ExpChar (fst c, A.TChar), A.TChar)
68    | S.ExpVar v ->
69      (match v with
70         nome ->
71         let id = fst nome in
72           (try (match (Amb.busca amb id) with
73                 | Amb.EntVar tipo -> (T.ExpVar (A.VarSimples nome, tipo),
                       tipo)
74                 | Amb.EntFun _ ->
75                   let msg = "nome de funcao usado como nome de variavel: "
                         ^ id in
76                    failwith (msg_erro nome msg)
77              )
78            with Not_found ->
79                   let msg = "A variavel " ^ id ^ " nao foi declarada" in
80                   failwith (msg_erro nome msg)
81           )
82      )
83
84    | S.ExpUn (op, valor) ->
85      let (v, tv) = infere_exp amb valor in
86
87      let verifica_unario () =
88        (match tv with
89          A.TInt -> tv
90          | A.TString -> tv
91          | A.TFloat -> tv
92          | A.TChar -> tv
93
94          | t -> let msg = "um operador relacional nao pode ser usado com o
              tipo " ^
```

```
95              (nome_tipo t)
96            in failwith (msg_erro_pos (snd op) msg)
97        )
98
99      in
100     let op = fst op in
101     let tinf = (match (classifica op) with
102           Unario     -> verifica_unario ()
103         | _ -> failwith "Encontrado operador binario para expressao unaria
                "
104       )
105     in
106       (T.ExpUn ((op, tinf), (v, tv)), tv)
107
108   | S.ExpBin (op, esq, dir) ->
109     let (esq, tesq) = infere_exp amb esq
110     and (dir, tdir) = infere_exp amb dir in
111
112     let verifica_aritmetico () =
113       (match tesq with
114         A.TInt ->
115           let _ = mesmo_tipo (snd op)
116             "O operando esquerdo eh do tipo %s mas o direito eh do tipo %s
                  "
117             tesq tdir
118           in tesq
119         | A.TFloat ->
120           let _ = mesmo_tipo (snd op)
121             "O operando esquerdo eh do tipo %s mas o direito eh do tipo %s
                  "
122             tesq tdir
123           in tesq
124
125        | t -> let msg = "um operador aritmetico nao pode ser usado com o
              tipo " ^
126                          (nome_tipo t)
127         in failwith (msg_erro_pos (snd op) msg)
128       )
129
130     and verifica_relacional () =
131       (match tesq with
132         A.TInt ->
133           let _ = mesmo_tipo (snd op)
134             "O operando esquerdo eh do tipo %s mas o direito eh do tipo %s
                  "
135             tesq tdir
136           in tesq
137         | A.TFloat ->
138           let _ = mesmo_tipo (snd op)
139             "O operando esquerdo eh do tipo %s mas o direito eh do tipo %s
                  "
140             tesq tdir
141           in tesq
142
143         | t -> let msg = "um operador relacional nao pode ser usado com o
               tipo " ^
144             (nome_tipo t)
145           in failwith (msg_erro_pos (snd op) msg)
146       )
```

```
147
148     in
149     let op = fst op in
150     let tinf = (match (classifica op) with
151           Aritmetico -> verifica_aritmetico ()
152         | Relacional -> verifica_relacional ()
153         | _ -> failwith "Unario encontrado em exp Binaria"
154       )
155     in
156       (T.ExpBin ((op,tinf), (esq, tesq), (dir, tdir)), tesq)
157
158   | S.ExpChamada (nome, args) ->
159      let rec verifica_parametros ags ps fs =
160         match (ags, ps, fs) with
161          (a::ags), (p::ps), (f::fs) ->
162             let _ = mesmo_tipo (posicao a)
163                       "O parametro eh do tipo %s mas deveria ser do tipo %s
                             " p f
164             in verifica_parametros ags ps fs
165        | [], [], [] -> ()
166        | _ -> failwith (msg_erro nome "Numero incorreto de parametros")
167      in
168      let id = fst nome in
169      try
170        begin
171          let open Amb in
172
173          match (Amb.busca amb id) with
174            Amb.EntFun {tipo_fn; formais} ->
175            let argst = List.map (infere_exp amb) args
176            and tipos_formais = List.map snd formais in
177            let _ = verifica_parametros args (List.map snd argst)
                 tipos_formais
178             in (T.ChamaFunc (id, (List.map fst argst), tipo_fn), tipo_fn)
179          | Amb.EntVar _ ->
180            let msg = id ^ " eh uma variavel e nao uma funcao" in
181            failwith (msg_erro nome msg)
182        end
183      with Not_found ->
184        let msg = "Nao existe a funcao de nome " ^ id in
185        failwith (msg_erro nome msg)
186
187 let rec verifica_cmd amb tiporet cmd =
188   let open A in
189   match cmd with
190     CmdRetorno exp ->
191     (match exp with
192        None ->
193         let _ = mesmo_tipo (Lexing.dummy_pos)
194               "O tipo retornado eh %s mas foi declarado como %s"
195               TVoid tiporet
196         in CmdRetorno None
197      | Some e ->
198         let (e1,tinf) = infere_exp amb e in
199         let _ = mesmo_tipo (posicao e)
200                   "O tipo retornado eh %s mas foi declarado como %s"
201                   tinf tiporet
202         in CmdRetorno (Some e1)
203       )
```

```
204   | CmdIf (teste, entao, senao) ->
205     let (teste1,tinf) = infere_exp amb teste
206     and entao1 = List.map (verifica_cmd amb tiporet) entao in
207     let senao1 =
208       match senao with
209         None -> None
210         | Some bloco -> Some (List.map (verifica_cmd amb tiporet) bloco)
211     in
212       CmdIf (teste1, entao1, senao1)
213
214   | CmdAtrib (elem, exp) ->
215     let (exp,  tdir) = infere_exp amb exp
216     and (elem1, tesq) = infere_exp amb elem in
217     let _ = mesmo_tipo (posicao elem)
218             "Atribuicao com tipos diferentes: %s = %s" tesq tdir
219     in CmdAtrib (elem1, exp)
220
221   | CmdWhile (teste, comandos) ->
222     let (teste1, tinf) = infere_exp amb teste
223     and corpo = List.map(verifica_cmd amb tiporet) comandos
224     in
225       CmdWhile (teste1, corpo)
226
227   | CmdFor (atrib, teste, comando, corpo) ->
228     let attr_externa = verifica_cmd amb tiporet atrib
229     and (teste1, tinf) = infere_exp amb teste
230     and attr_interna = verifica_cmd amb tiporet comando
231     and corpo1 = List.map(verifica_cmd amb tiporet) corpo
232     in
233       CmdFor(attr_externa, teste1, attr_interna, corpo1)
234
235   | CmdEntrada exps ->
236     let exps = List.map (infere_exp amb) exps in
237       CmdEntrada (List.map fst exps)
238
239   | CmdSaida exps ->
240     let exps = List.map (infere_exp amb) exps in
241       CmdSaida (List.map fst exps)
242
243   | CmdChamada exp ->
244     let (exp, tinf) = infere_exp amb exp in
245       CmdChamada exp
246
247 and verifica_fun amb ast =
248   let open A in
249   match ast with
250     A.Funcao {fn_nome; fn_tiporet; fn_formais; fn_locais; fn_corpo} ->
251       let ambfn = Amb.novo_escopo amb in
252         let insere_parametro (v,t) = Amb.insere_param ambfn (fst v) t in
253           let _ = List.iter insere_parametro fn_formais in
254             let insere_local = fun (CmdDec (v,t)) -> Amb.insere_local
                    ambfn (fst v)  t in
255               let _ = List.iter insere_local fn_locais in
256               let corpo_tipado = List.map (verifica_cmd ambfn fn_tiporet)
                      fn_corpo in
257
258               A.Funcao {
259                 fn_nome;
260                 fn_tiporet;
```

```
261                 fn_formais;
262                 fn_locais;
263                 fn_corpo = corpo_tipado
264               }
265
266 let rec verifica_dup xs =
267   match xs with
268     [] -> []
269   | (nome,t)::xs ->
270     let id = fst nome in
271     if (List.for_all (fun (n,t) -> (fst n) <> id) xs)
272     then (id, t) :: verifica_dup xs
273     else let msg = "Parametro duplicado " ^ id in
274       failwith (msg_erro nome msg)
275
276 let insere_declaracao_var amb dec =
277   let open A in
278     match dec with
279         CmdDec (nome, tipo) ->  Amb.insere_local amb (fst nome) tipo
280
281 let insere_declaracao_fun amb dec =
282   let open A in
283     match dec with
284       Funcao {fn_nome; fn_tiporet; fn_formais; fn_corpo} ->
285         let formais = verifica_dup fn_formais in
286         let nome = fst fn_nome in
287         Amb.insere_fun amb nome formais fn_tiporet
288
289 (* Lista de cabeçalhos das funções pré definidas *)
290 let fn_predefs = let open A in [
291    ("printf", [("x", TInt); ("y", TInt)], TVoid);
292    ("scanf",   [("x", TInt); ("y", TInt)], TVoid)
293 ]
294
295 (* insere as funções pré definidas no ambiente global *)
296 let declara_predefinidas amb =
297   List.iter (fun (n,ps,tr) -> Amb.insere_fun amb n ps tr) fn_predefs
298
299 let semantico ast =
300   (* cria ambiente global inicialmente vazio *)
301   let amb_global = Amb.novo_amb [] in
302   let _ = declara_predefinidas amb_global in
303   let (A.Programa (decs_globais, decs_funs, corpo)) = ast in
304   let _ = List.iter (insere_declaracao_var amb_global) decs_globais in
305   let _ = List.iter (insere_declaracao_fun amb_global) decs_funs in
306   (* Verificação de tipos nas funções *)
307   let decs_funs = List.map (verifica_fun amb_global) decs_funs in
308   (* Verificação de tipos na função principal *)
309   let corpo = List.map (verifica_cmd amb_global A.TVoid) corpo in
310      (A.Programa (decs_globais, decs_funs, corpo),  amb_global)
```

Como citado anteriormente, para o funcionamento é necessário a existência dos ambientes. As seguintes listagens trazem das definições e implementações desses ambientes.

**Listagem 6.9: Definição do ambiente**

```
1 type entrada_fn = { tipo_fn:  Ast.tipo;
2                     formais: (string * Ast.tipo) list;
```

```
3                       (*      locais: (string * Asabs.tipo) list *)
4 }
5
6 type entrada =  EntFun of entrada_fn
7              |  EntVar of Ast.tipo
8
9 type t
10
11 val novo_amb :  (string * entrada) list -> t
12 val novo_escopo : t -> t
13 val busca: t -> string -> entrada
14 val insere_local : t -> string -> Ast.tipo -> unit
15 val insere_param : t -> string -> Ast.tipo -> unit
16 val insere_fun : t -> string -> (string * Ast.tipo) list -> Ast.tipo ->
     unit
```

**Listagem 6.10: Implementação do ambiente**

```
1 module Tab = Tabsimb
2 module A = Ast
3
4 type entrada_fn = { tipo_fn:  A.tipo;
5                    formais: (string * A.tipo) list;
6 }
7
8 type entrada =  EntFun of entrada_fn
9              |  EntVar of A.tipo
10
11 type t = {
12   ambv : entrada Tab.tabela
13 }
14
15 let novo_amb xs = { ambv = Tab.cria xs }
16
17 let novo_escopo amb = { ambv = Tab.novo_escopo amb.ambv }
18
19 let busca amb ch = Tab.busca amb.ambv ch
20
21 let insere_local amb ch t =
22   Tab.insere amb.ambv ch (EntVar t)
23
24 let insere_param amb ch t =
25   Tab.insere amb.ambv ch (EntVar t)
26
27 let insere_fun amb nome params resultado =
28   let ef = EntFun { tipo_fn = resultado;
29                     formais = params }
30   in Tab.insere amb.ambv nome ef
```

As tabelas de simbolos utilizadas foram:

**Listagem 6.11: Definição da tabela de simbolos**

```
1
2 type 'a tabela
3
4 exception Entrada_existente of string
5
```

```
val insere : 'a tabela -> string -> 'a -> unit
val substitui : 'a tabela -> string -> 'a -> unit
val atualiza : 'a tabela -> string -> 'a -> unit
val busca  : 'a tabela -> string -> 'a
val cria   : (string * 'a) list -> 'a tabela

val novo_escopo : 'a tabela -> 'a tabela
```

**Listagem 6.12: Implementação da tabela de simbolos**

```
type 'a tabela = {
    tbl: (string, 'a) Hashtbl.t;
    pai: 'a tabela option;
}

exception Entrada_existente of string;;

let insere amb ch v =
  if Hashtbl.mem amb.tbl ch
  then raise (Entrada_existente ch)
  else Hashtbl.add amb.tbl ch v

let substitui amb ch v = Hashtbl.replace amb.tbl ch v

let rec atualiza amb ch v =
    if Hashtbl.mem amb.tbl ch
    then Hashtbl.replace amb.tbl ch v
    else match amb.pai with
       None -> failwith "tabsim atualiza: chave nao encontrada"
     | Some a -> atualiza a ch v

let rec busca amb ch =
  try Hashtbl.find amb.tbl ch
  with Not_found ->
    (match amb.pai with
       None -> raise Not_found
     | Some a -> busca a ch)

let rec cria cvs =
  let amb = {
    tbl = Hashtbl.create 5;
    pai = None
  } in
  let _ = List.iter (fun (c,v) -> insere amb c v) cvs
  in amb

let novo_escopo amb_pai = {
  tbl = Hashtbl.create 5;
  pai = Some amb_pai
}
```

## 6.3 Utilização

Para realizar a analise semântica de um programa, foi criado o seguinte código:

**Listagem 6.13: Arquivo para funcionamento**

```
open Printf
open Lexing

open Ast
exception Erro_Sintatico of string

module S = MenhirLib.General (* Streams *)
module I = Sintatico.MenhirInterpreter

open Semantico

let message =
  fun s ->
    match s with
    | 0 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 1 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 34 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 35 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 36 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 72 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 47 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 48 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 49 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 51 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 52 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 55 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 56 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 57 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 58 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 61 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 62 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 63 ->
        "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
    | 64 ->
```

```
52          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
53      | 73 ->
54          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
55      | 74 ->
56          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
57      | 95 ->
58          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
59      | 89 ->
60          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
61      | 97 ->
62          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
63      | 98 ->
64          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
65      | 99 ->
66          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
67      | 65 ->
68          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
69      | 66 ->
70          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
71      | 53 ->
72          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
73      | 67 ->
74          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
75      | 68 ->
76          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
77      | 59 ->
78          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
79      | 60 ->
80          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
81      | 42 ->
82          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
83      | 41 ->
84          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
85      | 70 ->
86          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
87      | 75 ->
88          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
89      | 77 ->
90          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
91      | 76 ->
92          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
93      | 105 ->
94          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
95      | 84 ->
96          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
97      | 43 ->
98          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
99      | 85 ->
100         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
101     | 86 ->
102         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
103     | 45 ->
104         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
105     | 46 ->
106         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
107     | 102 ->
108         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
109     | 103 ->
110         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
```

```
111     | 81 ->
112         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
113     | 3 ->
114         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
115     | 2 ->
116         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
117     | 6 ->
118         "estado 6: esperava um tipo. Exemplo:\n   x : inteiro;\n"
119     | 7 ->
120         "estado 7: esperava a definicao de um campo. Exemplo:\n   i:
               registro\n         parte_real: inteiro;\n         parte_imag:
               inteiro;\n      fim registro;\n      "
121     | 8 ->
122         "estado 8: esperava ':'. Exemplo:\n   x: inteiro;\n   "
123     | 9 ->
124         "estado 9: esperava um tipo. Exemplo:\n   x: inteiro;\n"
125     | 25 ->
126         "estado 25: esperva um ';'.\n"
127     | 26 ->
128         "estado 26: uma declaracao foi encontrada. Para continuar era\n
               esperado uma outra declara\195\167\195\163o ou a palavra '
               inicio'.\n"
129     | 29 ->
130         "estado 29: espera a palavra 'registro'. Exemplo:\n   i: registro\
               n         parte_real: inteiro;\n         parte_imag: inteiro;\n
                     fim registro;\n"
131     | 31 ->
132         "estado 31: esperava um ';'. \n"
133     | 107 ->
134         "estado 107: uma declaracao foi encontrada. Para continuar era\n
                    esperado uma outra declara\195\167\195\163o ou a palavra '
               inicio'.\n"
135     | 13 ->
136         "estado 13: esperava um '['. Exemplo:\n   arranjo [1..10] de
               inteiro;\n"
137     | 14 ->
138         "estado 14: esperava os limites do vetor. Exemplo:\n   arranjo
               [1..10] de inteiro;\n"
139     | 15 ->
140         "estado 15: esperava '..'. Exemplo:\n   1 .. 10\n"
141     | 16 ->
142         "estado 16: esperava um numero inteiro. Exemplo:\n   1 .. 10\n"
143     | 18 ->
144         "estado 18: esperava um ']'. Exemplo\n   arranjo [1..10] de
               inteiro;\n"
145     | 19 ->
146         "estado 19: esperava a palavra reservada 'de'. Exemplo:\n
               arranjo [1..10] de inteiro;\n"
147     | 20 ->
148         "estado 20: A variavel já foi declarada posteriormente.\n"
149     | _ ->
150         raise Not_found
151
152 let posicao lexbuf =
153     let pos = lexbuf.lex_curr_p in
154     let lin = pos.pos_lnum
155     and col = pos.pos_cnum - pos.pos_bol - 1 in
156     sprintf "linha %d, coluna %d" lin col
157
```

```
158 (* [pilha checkpoint] extrai a pilha do autômato LR(1) contida em
       checkpoint *)
159
160 let pilha checkpoint =
161   match checkpoint with
162   | I.HandlingError amb -> I.stack amb
163   | _ -> assert false (* Isso não pode acontecer *)
164
165 let estado checkpoint : int =
166   match Lazy.force (pilha checkpoint) with
167   | S.Nil -> (* O parser está no estado inicial *)
168      0
169   | S.Cons (I.Element (s, _, _, _), _) ->
170      I.number s
171
172 let sucesso v = Some v
173
174 let falha lexbuf (checkpoint : (Sast.expressao Ast.programa) I.checkpoint)
       =
175   let estado_atual = estado checkpoint in
176   let msg = message estado_atual in
177   raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n"
178                                         (Lexing.lexeme_start lexbuf) msg))
179
180 let loop lexbuf resultado =
181   let fornecedor = I.lexer_lexbuf_to_supplier Lexico.token lexbuf in
182   I.loop_handle sucesso (falha lexbuf) fornecedor resultado
183
184
185 let parse_com_erro lexbuf =
186   try
187     Some (loop lexbuf (Sintatico.Incremental.program lexbuf.lex_curr_p))
188   with
189   | Lexico.Erro msg ->
190      printf "Erro lexico na %s:\n\t%s\n" (posicao lexbuf) msg;
191      None
192   | Erro_Sintatico msg ->
193      printf "Erro sintático na %s %s\n" (posicao lexbuf) msg;
194      None
195
196 let parse s =
197   let lexbuf = Lexing.from_string s in
198   let ast = parse_com_erro lexbuf in
199   ast
200
201 let parse_arq nome =
202   let ic = open_in nome in
203   let lexbuf = Lexing.from_channel ic in
204   let ast = parse_com_erro lexbuf in
205   let _ = close_in ic in
206   ast
207
208 let verifica_tipos nome =
209   let ast = parse_arq nome in
210   match ast with
211     Some (Some ast) -> semantico ast
212   | _ -> failwith "Nada a fazer!\n"
213
214 (* Para compilar:
```

```
215     ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -
            package menhirLib semanticoTest.byte
216
217   Para usar, entre no ocaml
218
219     rlwrap ocaml
220
221   e se desejar ver apenas a árvore sintática que sai do analisador sintá
          tico, digite
222
223     parse_arq "exemplos/ex2.tip";;
224
225   Depois, para ver a saída do analisador semântico já com a árvore
          anotada com
226   o tipos, digite:
227
228   verifica_tipos "exemplos/ex2.tip";;
229
230   Note que o analisador semântico está retornando também o ambiente
          global. Se
231   quiser separá-los, digite:
232
233   let (arv, amb) = verifica_tipos "exemplos/ex2.tip";;
234
235
236
237 *)
```

E o arquivo *.ocamlinit* ficou da seguinte maneira:

**Listagem 6.14: .ocamlinit**

```
1 let () =
2   try Topdirs.dir_directory (Sys.getenv "OCAML_TOPLEVEL_PATH")
3   with Not_found -> ()
4 ;;
5
6 #use "topfind";;
7 #require "menhirLib";;
8 #directory "_build";;
9 #load "sintatico.cmo";;
10 #load "lexico.cmo";;
11 #load "ast.cmo";;
12 #load "sast.cmo";;
13 #load "tast.cmo";;
14 #load "tabsimb.cmo";;
15 #load "ambiente.cmo";;
16 #load "semantico.cmo";;
17 #load "semanticoTest.cmo";;
18
19 open Ast
20 open Ambiente
21 open SemanticoTest
```

Para realizar a compilação dos arquivos, são executados os seguintes comandos:

```
> menhir -v --list-errors sintatico.mly > sintatico.messages
```

```
> menhir -v --list-errors sintatico.mly --compile-errors sintatico.
    messages > fnmes.ml
> ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
    menhirLib semanticoTest.byte
```

Após a compilações dos arquivos, basta iniciar o *OCaml*, para analisar um arquivo é executado dentro do *shell* do *OCaml* o seguinte comando:

```
> verifica_tipos "caminho/para/o/codigo.c";;
```

Um exemplo de programa válido em *miniC*, possuindo diversos comandos diferentes é o seguinte:

**Listagem 6.15: Programa válido**

```
1  int outra_funcao(int i) {
2    int m;
3
4    m=0;
5
6    if (i==m) {
7      return 1;
8    } else {
9      return 0;
10   }
11 }
12
13 void sem_parametros() {
14   int k;
15
16   k = 2 * 15;
17 }
18
19 int fatorial(int n) {
20
21   if (n <= 0) {
22     return 1;
23   }
24   return n * fatorial(n-1);
25 }
26
27 void main() {
28   int n, i;
29   float k;
30
31   k = 1.34;
32   n = 100;
33
34   for (i=0; i<10; i = i + 1) {
35       n = i + 1;
36       n = fatorial(i);
37   }
38
39   while (n > 0) {
40       n = n / 2;
41       k = k * 2.0;
42   }
```

```
43
44    if (i != 0) {
45        n = 1;
46        outra_funcao(i);
47    } else if (n < 0) {
48        n = 0;
49    } else if (n > 1) {
50        n = 10;
51        outra_funcao(i);
52    } else if (n >= 50) {
53      k = 1.54;
54      i = 100;
55      sem_parametros();
56    } else if (k <= 100.0) {
57      n = 101;
58    } else {
59      sem_parametros();
60      i = 2 * n;
61    }
62
63 }
```

# Capítulo 7

# Interpretador

Após passar por todos os analisadores, léxico, sintático e semântico, o código em *miniC* está finalmente livre de qualquer erro possível da linguagem, erros lógicos continuam pois não são possíveis de ser capturados. Assim, um passo a ser realizado é o interpretador da linguagem.

## 7.1 Interprete e Compilador

O interprete funciona de maneira semelhanta ao analisador semântico, mas ele não avalia tipos, pois tais tipos já foram avaliados, o interprete irá executar a instrução passada pela linha do código. Alguns podem confundir o interprete com o compilador em si, mas cada um funciona de uma maneira diferente.

Um compilador irá ter como entrada um código fonte e após passar pelos analisadores para checagem de erro, quando chegar ao fim, irá produzir um arquivo objeto, que poderá ser executado pela máquina, de maneira bem rude, um compilador irá transformar um código fonte em um código de máquina para execução.

Um interprete tem um funcionamento diferente da seguinte maneira, irá também ser realizada a checagem de erros, e após toda checagem, o interprete irá passar uma linha por vez do código e executar a operação designada pela linha, realizar os comandos definidos, interpretando utilizando os valores que estão em seu ambiente e tabela de símbolos. Por esse simples explicação, já fica claro que interpretar um código é uma tarefa mais custosa do que executar um código nativamente pela linguagem de máquina gerada pelo compilador.

Para um exemplo mais claro de um código interpretado, temos o *Python*, com a sua própria interface de linha de comando, podemos executar comandos que serão interpretados quando executados, mantendo ainda as restrições da linguagem, por exemplo, executar uma função utilizando um variável não declarada. Num interpretador, se executarmos um mesmo comando $n$ vezes, ele será interpretado $n$ vezes, enquanto num compilador industrial, é pos-

sível que existam otimizações embutidas para melhorar a performance do código de máquina a ser gerado sem alterar a semântica.

## 7.2 Código do Interprete

Assim como analisador semântico, o interprete utiliza uma definição de ambiente.

**Listagem 7.1: Definição do ambiente**

```
type entrada_fn = {
  tipo_fn:  Ast.tipo;
  formais: (string * Ast.tipo) list;
  locais:  Ast.declaracoes;
  corpo: Tast.expressao Ast.comandos
}

type entrada =
  |  EntFun of entrada_fn
  |  EntVar of Ast.tipo *  (Tast.expressao option)

type t

val novo_amb :  (string * entrada) list -> t
val novo_escopo : t -> t
val busca:              t -> string -> entrada
val atualiza_var:    t -> string -> Ast.tipo ->  (Tast.expressao option)
    -> unit
val insere_local :    t -> string -> Ast.tipo -> (Tast.expressao option)
    -> unit
val insere_param : t -> string -> Ast.tipo -> (Tast.expressao option) ->
    unit
val insere_fun :  t ->
  string ->
  (string * Ast.tipo) list ->
   Ast.declaracoes ->
  Ast.tipo -> (Tast.expressao Ast.comandos) -> unit
```

Com a implementação das funções:

**Listagem 7.2: Implementação do ambiente**

```
module Tab = Tabsimb
module A = Ast
module T = Tast

type entrada_fn = {
  tipo_fn:  A.tipo;
  formais: (A.ident * A.tipo) list;
  locais:  A.declaracoes;
  corpo: T.expressao A.comandos
}

type entrada =  EntFun of entrada_fn
                        |  EntVar of A.tipo * (T.expressao option)

type t = {
```

```
16   ambv : entrada Tab.tabela
17 }
18
19 let novo_amb xs = { ambv = Tab.cria xs }
20
21 let novo_escopo amb = { ambv = Tab.novo_escopo amb.ambv }
22
23 let busca amb ch = Tab.busca amb.ambv ch
24
25 let atualiza_var amb ch t v =
26   Tab.atualiza amb.ambv ch (EntVar (t,v))
27
28 let insere_local amb nome t v =
29   Tab.insere amb.ambv nome (EntVar (t,v))
30
31 let insere_param amb nome t v =
32   Tab.insere amb.ambv nome (EntVar (t,v))
33
34 let insere_fun amb nome params locais resultado corpo =
35   let ef = EntFun { tipo_fn = resultado;
36                     formais = params;
37                     locais = locais;
38                     corpo = corpo }
39   in Tab.insere amb.ambv nome ef
```

O interpretador também possui sua definição:

**Listagem 7.3: Definição do interprete**

```
1 val interprete :  Tast.expressao Ast.programa -> unit
```

Com a implementação dos métodos para interpretar os comandos do código em *miniC*:

**Listagem 7.4: Implementação do interprete**

```
1 open Char
2
3 module Amb = AmbInterp
4 module A = Ast
5 module S = Sast
6 module T = Tast
7
8 exception Valor_de_retorno of T.expressao
9
10 let obtem_nome_tipo_var exp = let open T in
11   match exp with
12   | ExpVar (v,tipo) ->
13     (match v with
14       | A.VarSimples (nome,_) -> (nome,tipo)
15     )
16   | _ -> failwith "obtem_nome_tipo_var: nao eh variavel"
17
18 let pega_int_ou_bool exp =
19   match exp with
20   |  T.ExpInt (i,_) -> i
21   | T.ExpBool (b,_) ->
22     (match b with
23       true -> 1
24     | false -> 0
```

```
25      )
26    | _ -> failwith "pega_int: nao eh inteiro"
27
28  let pega_float_ou_bool exp =
29    match exp with
30    | T.ExpFloat (f,_)-> f
31    | T.ExpBool (b,_) ->
32      (match b with
33        true -> 1.0
34      | false -> 0.0
35      )
36    | _ -> failwith "pega_float: nao eh float"
37
38  let pega_char_ou_bool exp =
39    match exp with
40    | T.ExpChar (c,_) -> escaped c
41    | T.ExpBool (b,_) ->
42      (match b with
43        true -> "a"
44      | false -> ""
45      )
46    | _ -> failwith "pega_char: nao eh char"
47
48  let pega_string_ou_bool exp =
49    match exp with
50    |  T.ExpString (s,_) -> s
51    | T.ExpBool (b,_) ->
52      (match b with
53        true -> "a"
54      | false -> ""
55      )
56    | _ -> failwith "pega_string: nao eh string"
57
58
59  let pega_int exp =
60    match exp with
61    |  T.ExpInt (i,_) -> i
62    | _ -> failwith "pega_int: nao eh inteiro"
63
64  let pega_float exp =
65    match exp with
66    | T.ExpFloat (f,_)-> f
67    | _ -> failwith "pega_float: nao eh float"
68
69  let pega_char exp =
70    match exp with
71    | T.ExpChar (c,_) -> escaped c
72    | _ -> failwith "pega_char: nao eh char"
73
74  let pega_string exp =
75    match exp with
76    |  T.ExpString (s,_) -> s
77    | _ -> failwith "pega_string: nao eh string"
78
79  type classe_op = Aritmetico | Relacional | Unario
80
81  let classifica op =
82    let open A in
83    match op with
```

```
84    | Nao -> Unario
85    | Ou
86    | E
87    | Menor
88    | Maior
89    | MaiorI
90    | MenorI
91    | Igual
92    | Difer -> Relacional
93    | Mais
94    | Menos
95    | Mult
96    | Div
97    | Mod -> Aritmetico
98
99
100 let rec interpreta_exp amb exp =
101   let open A in
102   let open T in
103   match exp with
104   | ExpVoid
105   | ExpInt _
106   | ExpFloat _
107   | ExpString _
108   | ExpChar _
109   | ExpBool _ -> exp
110   | ExpVar _ ->
111     let (id,tipo) = obtem_nome_tipo_var exp in
112     (match (Amb.busca amb id) with
113      | Amb.EntVar (tipo, v) ->
114        (match v with
115         | None -> failwith ("variável nao inicializada: " ^ id)
116         | Some valor -> valor
117        )
118      |  _ -> failwith "interpreta_exp: expvar"
119     )
120
121   | ExpBin ((op,top), (esq, tesq), (dir,tdir)) ->
122     let vesq = interpreta_exp amb esq
123     and vdir = interpreta_exp amb dir in
124
125       let interpreta_aritmetico () =
126         (match tesq with
127           | TInt ->
128             (match op with
129               | Mais ->     ExpInt (pega_int vesq + pega_int vdir, top)
130               | Menos -> ExpInt (pega_int vesq - pega_int vdir, top)
131               | Mult ->     ExpInt (pega_int vesq * pega_int vdir, top)
132               | Div  ->      ExpInt (pega_int vesq / pega_int vdir, top)
133               | Mod  ->      ExpInt (pega_int vesq mod pega_int vdir, top)
134               | _ -> failwith "interpreta_aritmetico"
135             )
136
137           | TFloat ->
138             (match op with
139               | Mais -> ExpFloat (pega_float vesq +. pega_float vdir, top)
140               | Menos -> ExpFloat (pega_float vesq -. pega_float vdir, top
                    )
141               | Mult -> ExpFloat (pega_float vesq *. pega_float vdir, top)
```

```
142             | Div -> ExpFloat (pega_float vesq /. pega_float vdir, top)
143             | Mod -> ExpFloat (mod_float (pega_float vesq) (pega_float
                    vdir), top)
144             | _ -> failwith "interpreta_aritmetico"
145           )
146       | _ -> failwith "Valor nao permitido para operador aritmetico"
147     )
148
149     and interpreta_relacional () =
150       (match tesq with
151         | TInt ->
152           (match op with
153             | Menor -> ExpBool (pega_int vesq < pega_int vdir, top)
154             | Maior -> ExpBool (pega_int vesq > pega_int vdir, top)
155             | Igual -> ExpBool (pega_int vesq == pega_int vdir, top)
156             | Difer -> ExpBool (pega_int vesq != pega_int vdir, top)
157             | MenorI -> ExpBool (pega_int vesq <= pega_int vdir, top)
158             | MaiorI -> ExpBool (pega_int vesq >= pega_int vdir, top)
159             | Ou -> ExpBool ((pega_int_ou_bool vesq) != 0  || (
                    pega_int_ou_bool vdir) != 0, top)
160             | E -> ExpBool ((pega_int_ou_bool vesq) != 0  && (
                    pega_int_ou_bool vdir) != 0, top)
161             | _ -> failwith "interpreta_relacional"
162           )
163
164         | TFloat ->
165           (match op with
166             | Menor -> ExpBool (pega_float vesq < pega_float vdir, top
                    )
167             | Maior  -> ExpBool (pega_float vesq > pega_float vdir,
                    top)
168             | Igual   -> ExpBool (pega_float vesq == pega_float vdir,
                    top)
169             | Difer   -> ExpBool (pega_float vesq != pega_float vdir,
                    top)
170             | MenorI -> ExpBool (pega_float vesq <= pega_float vdir,
                    top)
171             | MaiorI -> ExpBool (pega_float vesq >= pega_float vdir,
                    top)
172             | Ou -> ExpBool ((pega_float_ou_bool vesq) != 0.0  || (
                    pega_float_ou_bool vdir) != 0.0, top)
173             | E -> ExpBool ((pega_float_ou_bool vesq) != 0.0  && (
                    pega_float_ou_bool vdir) != 0.0, top)
174             | _ -> failwith "interpreta_relacional"
175           )
176
177         | TString ->
178           (match op with
179             | Menor -> ExpBool (pega_string vesq < pega_string vdir,
                    top)
180             | Maior  -> ExpBool (pega_string vesq > pega_string vdir,
                    top)
181             | Igual   -> ExpBool (pega_string vesq == pega_string vdir
                    , top)
182             | Difer   -> ExpBool (pega_string vesq != pega_string vdir
                    , top)
183             | MenorI -> ExpBool (pega_string vesq <= pega_string vdir,
                     top)
184             | MaiorI -> ExpBool (pega_string vesq >= pega_string vdir,
```

```
                    top)
185               | Ou -> ExpBool ((pega_string_ou_bool vesq) != ""  || (
                     pega_string_ou_bool vdir) != "", top)
186               | E -> ExpBool ((pega_string_ou_bool vesq) != ""  && (
                     pega_string_ou_bool vdir) != "", top)
187               | _ -> failwith "interpreta_relacional"
188             )
189
190           | TChar ->
191             (match op with
192               | Menor -> ExpBool (pega_char vesq < pega_char vdir, top)
193               | Maior  -> ExpBool (pega_char vesq > pega_char vdir, top)
194               | Igual   -> ExpBool (pega_char vesq == pega_char vdir,
                     top)
195               | Difer   -> ExpBool (pega_char vesq != pega_char vdir,
                     top)
196               | MenorI -> ExpBool (pega_char vesq <= pega_char vdir, top
                     )
197               | MaiorI -> ExpBool (pega_char vesq >= pega_char vdir, top
                     )
198               | Ou -> ExpBool ((pega_char_ou_bool vesq) != ""  || (
                     pega_char_ou_bool vdir) != "", top)
199               | E -> ExpBool ((pega_char_ou_bool vesq) != ""  && (
                     pega_char_ou_bool vdir) != "", top)
200               | _ -> failwith "interpreta_relacional"
201             )
202
203           | _ ->  failwith "interpreta_relacional"
204         )
205
206   in
207   let valor = (match (classifica op) with
208         Aritmetico -> interpreta_aritmetico ()
209       | Relacional -> interpreta_relacional ()
210       | _ -> failwith "Operado diferente de binario em exp binaria"
211     )
212   in
213     valor
214
215 | ExpUn ((op,top), (dir,tdir)) ->
216   let valor = interpreta_exp amb dir in
217     let interpreta_unario () =
218       (match tdir with
219         | TInt ->
220           (match op with
221             Nao -> ExpBool (not (pega_int_ou_bool valor != 0), top)
222           | _ -> failwith "interpreta_unario")
223         | TFloat ->
224           (match op with
225             Nao -> ExpBool (not (pega_float_ou_bool valor != 0.0), top)
226           | _ -> failwith "interpreta_unario")
227         | TString ->
228           (match op with
229             Nao -> ExpBool (not (pega_string_ou_bool valor != ""), top)
230           | _ -> failwith "interpreta_unario")
231         | TChar ->
232           (match op with
233             Nao -> ExpBool (not (pega_char_ou_bool valor != ""), top)
234           | _ -> failwith "interpreta_unario")
```

```
235           | _ -> failwith "interpreta_unario"
236         )
237     in
238       let valor = (match (classifica op) with
239             Unario -> interpreta_unario ()
240           | _ -> failwith "Operado diferente de unario em exp unaria"
241         )
242       in
243         valor
244
245   | ChamaFunc (id, args, tipo) ->
246     let open Amb in
247     ( match (Amb.busca amb id) with
248       | Amb.EntFun {tipo_fn; formais; locais; corpo} ->
249           let vargs = List.map (interpreta_exp amb) args in
250           let vformais = List.map2 (fun (n,t) v -> (n, t, Some v))
                formais vargs
251           in interpreta_fun amb id vformais locais corpo
252       | _ -> failwith "interpreta_exp: expchamada"
253     )
254
255 and interpreta_fun amb fn_nome fn_formais fn_locais fn_corpo =
256   let open A in
257   let ambfn = Amb.novo_escopo amb in
258    let insere_local  d =
259     match d with
260       (CmdDec (v,t)) -> Amb.insere_local ambfn (fst v)  t None
261   in
262   let insere_parametro (n,t,v) = Amb.insere_param ambfn n t v in
263   let _ = List.iter insere_parametro fn_formais in
264     let _ = List.iter insere_local fn_locais in
265   try
266     let _ = List.iter (interpreta_cmd ambfn) fn_corpo in T.ExpVoid
267     with
268        Valor_de_retorno expret -> expret
269
270 and interpreta_cmd amb cmd =
271   let open A in
272   let open T in
273   match cmd with
274   CmdRetorno exp ->
275     (match exp with
276        None -> raise (Valor_de_retorno ExpVoid)
277      | Some e ->
278        let e1 = interpreta_exp amb e in
279        raise (Valor_de_retorno e1)
280     )
281
282   | CmdIf (teste, entao, senao) ->
283     let teste1 = interpreta_exp amb teste in
284     (match teste1 with
285        ExpBool (true,_) ->
286        List.iter (interpreta_cmd amb) entao
287      | _ ->
288        (match senao with
289           None -> ()
290         | Some bloco -> List.iter (interpreta_cmd amb) bloco
291        )
292     )
```

```
293
294   | CmdAtrib (elem, exp) ->
295       let exp = interpreta_exp amb exp
296       and (elem1,tipo) = obtem_nome_tipo_var elem in
297         Amb.atualiza_var amb elem1 tipo (Some exp)
298
299   | CmdWhile (teste, corpo) ->
300     let teste1 = interpreta_exp amb teste in
301     (match teste1 with
302       ExpBool (true, _) ->
303         let entao1 = corpo @ [CmdWhile(teste, corpo)] in
304         List.iter (interpreta_cmd amb) entao1
305       | _ -> ()
306     )
307
308   | CmdFor (attr_esq, teste, attr_dir, corpo) ->
309     let _ = interpreta_cmd amb attr_esq
310     and teste1 = interpreta_exp amb teste in
311       (match teste1 with
312         ExpBool (true, _) ->
313           let corpo1 = corpo @ [CmdFor(attr_dir, teste, attr_dir, corpo)]
                in
314           List.iter (interpreta_cmd amb) corpo1
315         | _ -> ()
316       )
317
318   | CmdEntrada exps ->
319     let nts = List.map (obtem_nome_tipo_var) exps in
320     let leia_var (nome,tipo) =
321       let valor =
322         (match tipo with
323           | A.TInt    -> T.ExpInt    (read_int (),  tipo)
324           | A.TFloat  -> T.ExpFloat  (read_float (), tipo)
325           | A.TString -> T.ExpString (read_line (), tipo)
326           | _ -> failwith "leia_var: nao implementado"
327         )
328       in Amb.atualiza_var amb nome tipo (Some valor)
329     in
330       List.iter leia_var nts
331
332   | CmdSaida exps ->
333     let exps = List.map (interpreta_exp amb) exps in
334     let imprima exp =
335       (match exp with
336         | T.ExpInt (n,_) -> print_int n
337         | T.ExpFloat (f,_) -> let _ = print_float f in print_string " "
338         | T.ExpString (s,_) -> let _ = print_string s in print_string " "
339         | T.ExpBool (b,_) ->
340           let _ = print_string (if b then "true" else "false")
341             in print_string " "
342         | _ -> failwith "imprima: nao implementado"
343       )
344     in
345       let _ = List.iter imprima exps in
346       print_newline ()
347
348   | CmdChamada exp -> ignore( interpreta_exp amb exp)
349
350 let insere_declaracao_var amb dec =
```

```
351     match dec with
352         A.CmdDec (nome, tipo) ->  Amb.insere_local amb (fst nome) tipo
                None
353
354 let insere_declaracao_fun amb dec =
355   let open A in
356     match dec with
357       Funcao {fn_nome; fn_tiporet; fn_formais; fn_locais; fn_corpo} ->
358         let nome = fst fn_nome in
359         let formais = List.map (fun (n,t) -> ((fst n), t)) fn_formais in
360         Amb.insere_fun amb nome formais fn_locais fn_tiporet fn_corpo
361
362 let fn_predefs = let open A in [
363     ("printf", [("x", TInt); ("y", TInt)], TVoid, []);
364     ("scanf",  [("x", TInt); ("y", TInt)], TVoid, []);
365 ]
366
367 let declara_predefinidas amb =
368   List.iter (fun (n,ps,tr,c) -> Amb.insere_fun amb n ps [] tr c)
        fn_predefs
369
370 let interprete ast =
371   let amb_global = Amb.novo_amb [] in
372   let _ = declara_predefinidas amb_global in
373   let (A.Programa (decs_globais, decs_funs, corpo)) = ast in
374   let _ = List.iter (insere_declaracao_var amb_global) decs_globais in
375   let _ = List.iter (insere_declaracao_fun amb_global) decs_funs in
376   let resultado = List.iter (interpreta_cmd amb_global) corpo in
377   resultado
```

Para poder testar o interprete, foi realizada um implementação que inicialmente verifica se existem erros no código do arquivo passado, a para o interprete é passado uma árvore tipada proveniente do analisador semântico, a partir dela é realizada a interpretação dos comandos, a implementação é a seguinte:

**Listagem 7.5: Implementação**

```
1  open Printf
2  open Lexing
3
4  open Ast
5  exception Erro_Sintatico of string
6
7  module S = MenhirLib.General (* Streams *)
8  module I = Sintatico.MenhirInterpreter
9
10 open Semantico
11
12 let message =
13   fun s ->
14     match s with
15     | 0 ->
16         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
17     | 1 ->
18         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
19     | 34 ->
20         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
21     | 35 ->
22         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
```

```
23      | 36 ->
24          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
25      | 72 ->
26          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
27      | 47 ->
28          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
29      | 48 ->
30          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
31      | 49 ->
32          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
33      | 51 ->
34          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
35      | 52 ->
36          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
37      | 55 ->
38          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
39      | 56 ->
40          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
41      | 57 ->
42          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
43      | 58 ->
44          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
45      | 61 ->
46          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
47      | 62 ->
48          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
49      | 63 ->
50          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
51      | 64 ->
52          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
53      | 73 ->
54          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
55      | 74 ->
56          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
57      | 95 ->
58          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
59      | 89 ->
60          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
61      | 97 ->
62          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
63      | 98 ->
64          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
65      | 99 ->
66          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
67      | 65 ->
68          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
69      | 66 ->
70          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
71      | 53 ->
72          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
73      | 67 ->
74          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
75      | 68 ->
76          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
77      | 59 ->
78          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
79      | 60 ->
80          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
81      | 42 ->
```

```
82          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
83      | 41 ->
84          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
85      | 70 ->
86          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
87      | 75 ->
88          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
89      | 77 ->
90          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
91      | 76 ->
92          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
93      | 105 ->
94          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
95      | 84 ->
96          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
97      | 43 ->
98          "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
99      | 85 ->
100         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
101     | 86 ->
102         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
103     | 45 ->
104         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
105     | 46 ->
106         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
107     | 102 ->
108         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
109     | 103 ->
110         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
111     | 81 ->
112         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
113     | 3 ->
114         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
115     | 2 ->
116         "<YOUR SYNTAX ERROR MESSAGE HERE>\n"
117     | 6 ->
118         "estado 6: esperava um tipo. Exemplo:\n   x : inteiro;\n"
119     | 7 ->
120         "estado 7: esperava a definicao de um campo. Exemplo:\n   i:
               registro\n         parte_real: inteiro;\n         parte_imag:
               inteiro;\n      fim registro;\n      "
121     | 8 ->
122         "estado 8: esperava ':'. Exemplo:\n   x: inteiro;\n   "
123     | 9 ->
124         "estado 9: esperava um tipo. Exemplo:\n   x: inteiro;\n"
125     | 25 ->
126         "estado 25: esperva um ';'.\n"
127     | 26 ->
128         "estado 26: uma declaracao foi encontrada. Para continuar era\n
               esperado uma outra declara\195\167\195\163o ou a palavra '
               inicio'.\n"
129     | 29 ->
130         "estado 29: espera a palavra 'registro'. Exemplo:\n   i: registro\
               n         parte_real: inteiro;\n         parte_imag: inteiro;\n
                     fim registro;\n"
131     | 31 ->
132         "estado 31: esperava um ';'. \n"
133     | 107 ->
134         "estado 107: uma declaracao foi encontrada. Para continuar era\n
```

```
                esperado uma outra declara\195\167\195\163o ou a palavra '
              inicio'.\n"
135     | 13 ->
136         "estado 13: esperava um '['. Exemplo:\n   arranjo [1..10] de
              inteiro;\n"
137     | 14 ->
138         "estado 14: esperava os limites do vetor. Exemplo:\n   arranjo
              [1..10] de inteiro;\n"
139     | 15 ->
140         "estado 15: esperava '..'. Exemplo:\n   1 .. 10\n"
141     | 16 ->
142         "estado 16: esperava um numero inteiro. Exemplo:\n   1 .. 10\n"
143     | 18 ->
144         "estado 18: esperava um ']'. Exemplo\n   arranjo [1..10] de
              inteiro;\n"
145     | 19 ->
146         "estado 19: esperava a palavra reservada 'de'. Exemplo:\n
              arranjo [1..10] de inteiro;\n"
147     | 20 ->
148         "estado 20: esperava um tipo. Exemplo\n   arranjo [1..10] de
              inteiro;\n"
149     | _ ->
150         raise Not_found
151
152 let posicao lexbuf =
153     let pos = lexbuf.lex_curr_p in
154     let lin = pos.pos_lnum
155     and col = pos.pos_cnum - pos.pos_bol - 1 in
156     sprintf "linha %d, coluna %d" lin col
157
158 (* [pilha checkpoint] extrai a pilha do autômato LR(1) contida em
      checkpoint *)
159
160 let pilha checkpoint =
161   match checkpoint with
162   | I.HandlingError amb -> I.stack amb
163   | _ -> assert false (* Isso não pode acontecer *)
164
165 let estado checkpoint : int =
166   match Lazy.force (pilha checkpoint) with
167   | S.Nil -> (* O parser está no estado inicial *)
168      0
169   | S.Cons (I.Element (s, _, _, _), _) ->
170      I.number s
171
172 let sucesso v = Some v
173
174 let falha lexbuf (checkpoint : (Sast.expressao Ast.programa) I.checkpoint)
      =
175   let estado_atual = estado checkpoint in
176   let msg = message estado_atual in
177   raise (Erro_Sintatico (Printf.sprintf "%d - %s.\n"
178                                         (Lexing.lexeme_start lexbuf) msg))
179
180 let loop lexbuf resultado =
181   let fornecedor = I.lexer_lexbuf_to_supplier Lexico.token lexbuf in
182   I.loop_handle sucesso (falha lexbuf) fornecedor resultado
183
184
```

```
185 let parse_com_erro lexbuf =
186   try
187     Some (loop lexbuf (Sintatico.Incremental.program lexbuf.lex_curr_p))
188   with
189   | Lexico.Erro msg ->
190      printf "Erro lexico na %s:\n\t%s\n" (posicao lexbuf) msg;
191      None
192   | Erro_Sintatico msg ->
193      printf "Erro sintático na %s %s\n" (posicao lexbuf) msg;
194      None
195
196 let parse s =
197   let lexbuf = Lexing.from_string s in
198   let ast = parse_com_erro lexbuf in
199   ast
200
201 let parse_arq nome =
202   let ic = open_in nome in
203   let lexbuf = Lexing.from_channel ic in
204   let ast = parse_com_erro lexbuf in
205   let _ = close_in ic in
206   ast
207
208 let verifica_tipos nome =
209   let ast = parse_arq nome in
210   match ast with
211     Some (Some ast) -> semantico ast
212   | _ -> failwith "Nada a fazer!\n"
213
214
215 let interprete nome =
216   let tast,amb = verifica_tipos nome in
217   Interprete.interprete tast
```

Para quando iniciarmos o *OCaml* todas as dependências serem carregadas de forma correta, o arquivo do *.ocamlinit* foi seguinte:

**Listagem 7.6: .ocamlinit para carregar os arquivos necessários**

```
1 let () =
2   try Topdirs.dir_directory (Sys.getenv "OCAML_TOPLEVEL_PATH")
3   with Not_found -> ()
4 ;;
5
6 #use "topfind";;
7 #require "menhirLib";;
8 #directory "_build";;
9 #load "sintatico.cmo";;
10 #load "lexico.cmo";;
11 #load "ast.cmo";;
12 #load "sast.cmo";;
13 #load "tast.cmo";;
14 #load "tabsimb.cmo";;
15 #load "ambiente.cmo";;
16 #load "semantico.cmo";;
17 #load "ambInterp.cmo";;
18 #load "interprete.cmo";;
19 #load "interpreteTeste.cmo";;
20
```

```
21 open Ast
22 open AmbInterp
23 open InterpreteTeste
```

### 7.2.1 Execução

Para executar o interprete, é necessário compilador o código implementado, para tal, execute no terminal:

```
> menhir -v --list-errors sintatico.mly > sintatico.messages
> menhir -v --list-errors sintatico.mly --compile-errors sintatico.
    messages > fnmes.ml
> ocamlbuild -use-ocamlfind -use-menhir -menhir "menhir --table" -package
    menhirLib interpreteTeste.byte
```

Após a compilação ser realizada de forma correta, entre no *OCaml* e execute:

```
> interprete "caminho/para/o/codigo.c";;
```

Assim, o código contido no arquivo referente ao caminho passado ao interprete será interpretado, os comandos serão executados linha por linha. Um exemplo de um programa válido em *miniC*, para ser interpretado:

**Listagem 7.7: Programa válido**

```
1 int fatorial(int n) {
2   if (n <= 0) {
3     return 1;
4   }
5   return (n * fatorial(n-1));
6 }
7
8 void main() {
9   int n, i;
10  float k;
11
12  k = 1.34;
13  n = 100;
14
15  printf("OLA");
16  printf(1);
17  printf(n || n);
18  printf("Insira valor de n");
19  scanf(n);
20
21  printf("\nFATORIAL");
22  printf(fatorial(n));
23
24  printf("\nWHILE");
25  while (n > 0) {
26      n = n - 1;
27      printf(n);
28  }
29
30  printf("\nFOR");
```

```
31    for (i=0; i<10; i = i + 1) {
32        printf(i);
33    }
34
35    printf("\nExpressao");
36    printf(!k);
37
38    printf("\nExpressao");
39    printf(!(1<2 && 2<3));
40 }
41
42
43 main();
```