Construção de um compilador de Python para Dalvik usando Objective Caml

Miguel Henrique de Brito Pereia
miguelhbrito@gmail.com

Faculdade de Computação
Universidade Federal de Uberlândia

7 de abril de 2019

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Sumário

Capítulo 1

Introdução

Este documento foi escrito para auxiliar na confecção do relatório da disciplina de Construção de Compiladores com a finalidade de detalhar todo o trabalho desenvolvido e os processos envolvidos da Construção de um Compilador, mais especificamente, um Compilador de Python para Dalvik, utilizando a linguagem OCaml para a construção do mesmo.

1.1 Sistema Operacional

Para esse trabalho foi utilizado o sistema operacional *Fedora 28*, sua instalação é fácil e rápida, basta acessar o site e seguir os passos descritos pela desenvolvedora do sistema.

1.2 Python

Python é uma ótima linguagem de programação orientada a objetos, interpretada e interativa é uma linguagem de programação orientada a objetos, interpretada, de script, interativa, funcional e de tipagem dinâmica. Criada por Guido van Rossum em 1991, hoje segue o modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation*.

1.3 Dalvik

Desenvolvida por Dan Bornstein e com contribuições de outros engenheiros do Google, é uma máquina virtual baseada em registradores e foi projetada para ser utilizada no sistema operacional Android. É muito conhecida pelo seu bom desempenho, pelo baixo consumo de memória e foi projetada para permitir que múltiplas instâncias da máquina virtual rodem ao mesmo tempo. A *Dalvik* é frequentemante confundida com uma Java Virtual Machine, porém, o bytecode que ela opera é bastante diferente do bytecode da JVM.
A VM do Dalvik, executa um bytecode no formato .dex (Dalvik Executable), codigos em .dex

são ilegíveis aos humanos, portanto, neste trabalho usaremos Smali Code para apresentar os códigos.

1.4 Smali/Baksmali

O *smali/baksmali* é um *assembler/disassembler* para o formato *.dex* usado pela *Dalvik*, que gera um arquivo *SmaliCode*. A sintaxe é vagamente baseada na sintaxe do *Jasmin*, e suporta a funcionalidadesdo formato *.dex* (anotações, informações de depuração, informações de linha, etc.)

1.5 OCaml

Objective Caml, ou somente *OCaml*, é uma linguagem de programação funcional e fortemente tipada, da família ML com ênfase na expressividade e na segurança. É usada em aplicações sensíveis onde um único erro pode custar milhões.
Será utilizada na implementação de nosso compilador.

Capítulo 2

Instalações

2.1 Python

Já vem instalado por padrão em sistemas *GNU/Linux*, para conferir a versão, digite no terminal:

```
>  which python
```

2.2 Java

Para checar as versões disponíveis, digite no terminal:

```
>sudo dnf search openjdk
```

Instale a versão desejada digitando no terminal:

```
>sudo dnf install <openjdk-package-name>
```

Por exemplo:

```
>sudo dnf install java-1.8.0-openjdk.x86_64
```

Para verificar se foi instalado com sucesso digite:

```
>java -version
```

2.3 Dalvik

O Dalvik é uma VM executada em android, então neste trabalha usaremos o Android Studio. Para instalar basta ir no site baixar a versão que se aplica ao seu SO e configurar o PATH no terminal:

```
>export PATH=\$PATH:/diretoriolocal/android-studio/bin
```

Para executar o Android Studio, entre no diretório android-studio/bin e digite no terminal:

```
>sh studio.sh
```

Os componentes adicionais serão instalados com a ajuda do assistente de configuração na primeira execução do programa.

2.4 OCaml

Versão utilizada: 4.07.0

```
>sudo dnf install wget
>sudo dnf install git m4 mercurial darcs
>wget https://raw.github.com/ocaml/opam/master/shell/
>opam_installer.sh -O - | sh -s /dev/bin
>opam init
>eval `opam config env`
>opam repository add git git+https://github.com/ocaml/
>opam-repository
>opam update
```

Para saber qual a versão mais atual:

```
>opam switch
```

Instalando a versão 4.0.7:

```
>opam switch 4.07.0
>eval `opam config env`
```

Instalar o rlwraper para trabalhar melhor com o OCaml:

```
> sudo dnf update
> sudo dnf install rlwrap
```

Executar o OCaml com o rlwrap:

```
>rlwraper ocaml
```

2.5 Compilação

Como gerar .smeli a partir do .dex.

2.5.1 Compilando Java em .dex

Dentro do diretório execute no terminal:

```
>javac file.java
> diretorioSDK/build-tools/version/dx --dex --output=file.dex file.class
```

Onde esta "diretorioSDK"é o caminho no qual foi instalado o SDK com o assistente de configuração do Android Studio, assim, como "version"a versão que esta sendo utilizada.
Baixe o baksmali e smali, os dois na versão 2.2.6, no site para fazer o desassembly.
É importante ressaltar que tem que deixar o .dex/.smali e o baksmali/smali no mesmo diretório.

2.5.2 Complilando .dex em .smali

Dentro do diretório execute no terminal:

```
>java -jar baksmali-2.2.6.jar disassemble file.dex
```

2.5.3 Compilando .smali code em .dex

Dentro do diretório execute no terminal:

```
> java -jar smali-2.2.6.jar assemble file.smali -o file.dex
```

2.6 Executando o arquivo .dex no Android

Depois de compilado o .dex, para executa-lo usaremos um emulador que rode o sistema operacional android. Usaremos o modelo Nexus 5, android 5.1.

2.6.1 Utilizando Emulador AVD

Após instalado o Android Studio, execute-o e crie um novo projeto em branco. Na interface do programa, vá em: *Tools>ADV Manager*.

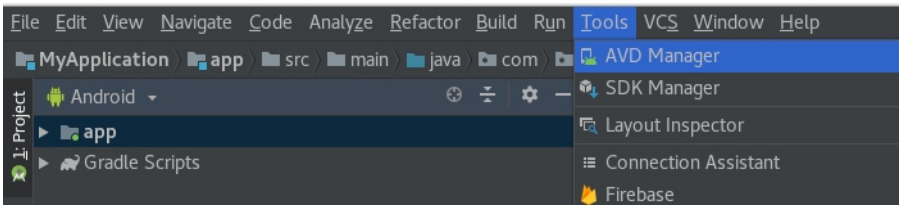

Figura 2.1: *ADV Manager*

A seguinte tela irá aparecer:

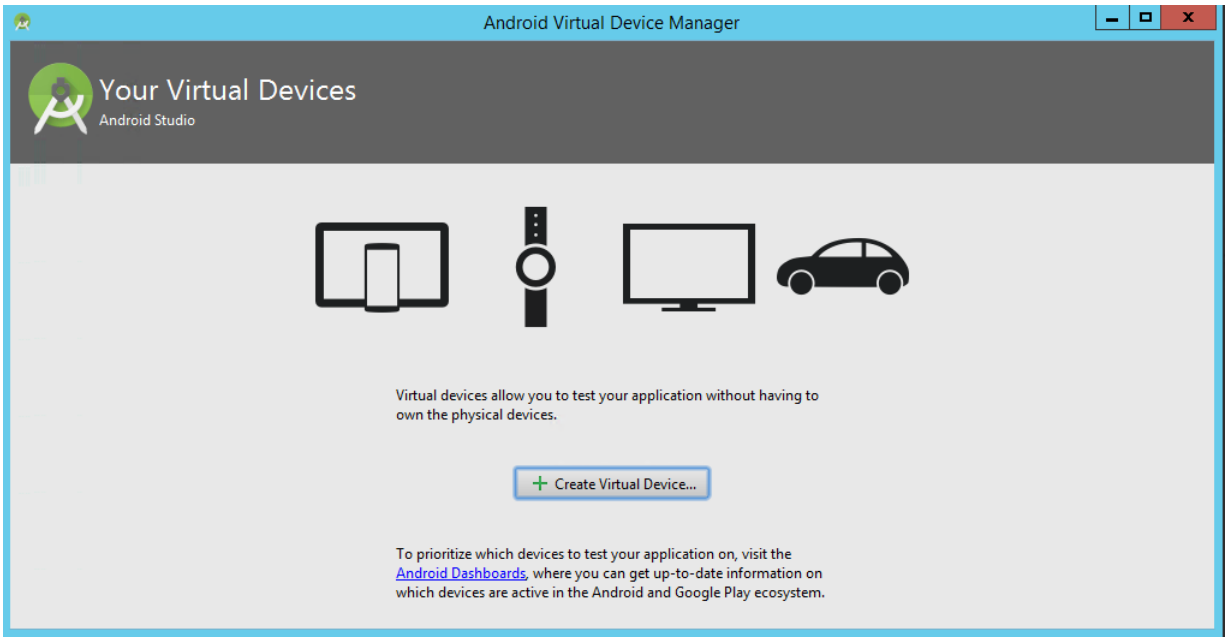

Figura 2.2: *Criar um novo device*

Nessa janela, selecione *Create Virtual Device*.
Logo em seguida irá aparecer essa janela:

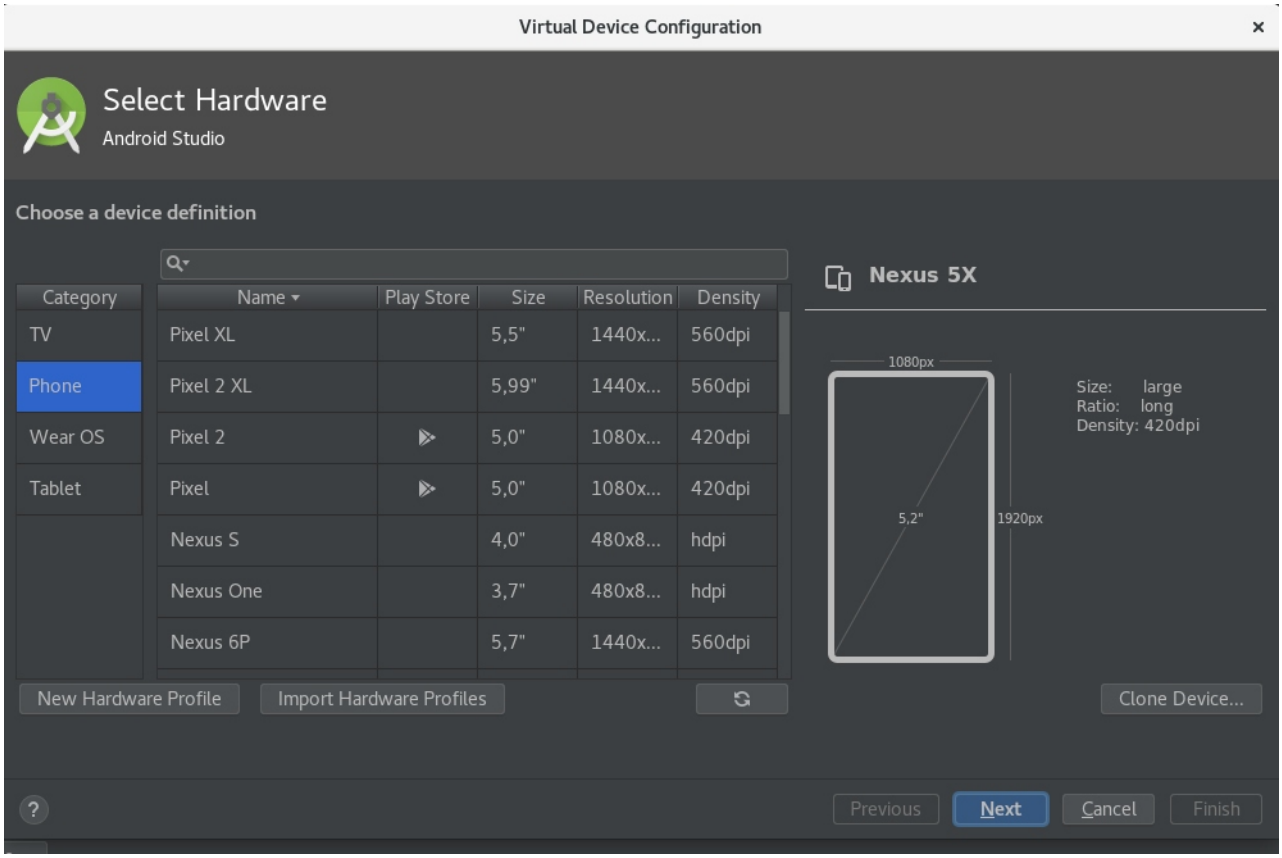

Figura 2.3: *Modelo Device*

Selecione o modelo que deseja emular, indico usar modelos Nexus, continue para as próximas

janelas clicando em next.
Abrindo o ADV Manager novamente, seu modelo emulado deverá aparecer como mostrado na figura abaixo, para iniciar, clique no icone *Play*.

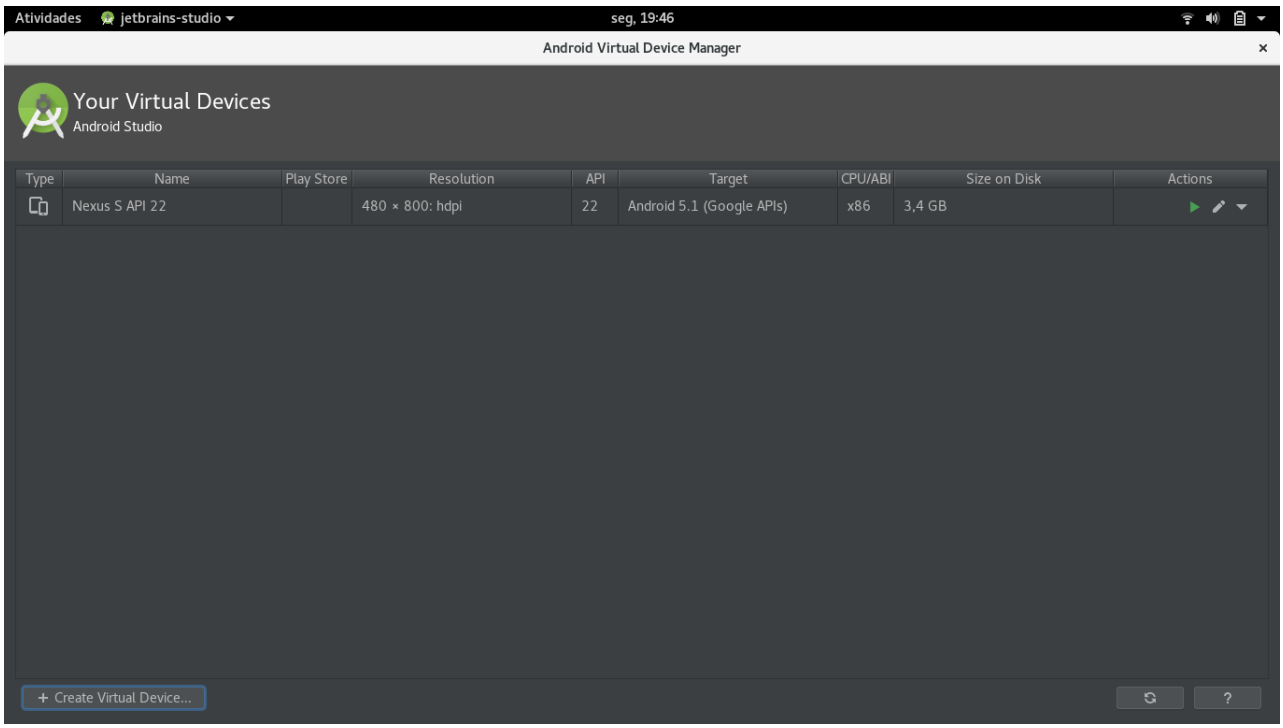

Figura 2.4: *Manager Devices*

Ao faazer esses passos ira aparecer um emulador da tela do celular, com isso ja podemos rodar arquivos .dex.
No terminal digite:

```
>./adb devices
>./adb push /home/fenrir/dev/Projetos/compilados/nano01.dex /data/local
> ./adb shell dalvikvm -cp /data/local/nano01.dex nano01
```

A execução da ferramenta adb deve ser feita dentro do diretório *androidpath/sdk/plataform-tools/*.

Capítulo 3

Nano Programas

Neste capítulo será apresentado alguns programas e suas respectivas versões em Python, Java e Smali Code.

3.1 Nano01

Listagem 3.1: Nano01.py

```
1 def main() -> None:
2     return
```

Listagem 3.2: Nano01.java

```
1 public class Nano01 {
2     public static void main(String[] args) {
3     }
4 }
```

Listagem 3.3: 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 3
22      return-void
23  .end method
```

Saída : nenhuma.

3.2 Nano02

Listagem 3.4: Nano02.py

```
1 def main() -> None:
2     n: int = 0
```

Listagem 3.5: Nano02.java

```
1 public class Nano02 {
2     public static void main(String[] args) {
3         int n;
4     }
5  }
```

Listagem 3.6: 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
```

Saída : nenhuma.

3.3 Nano03

Listagem 3.7: Nano03.py

```
1 def main() -> None:
```

```
2     n: int = 1
```

Listagem 3.8: Nano03.java

```
1 public class Nano03 {
2     public static void main(String[] args) {
3         int n;
4         n = 1;
5     }
6 }
```

Listagem 3.9: Nano03.smali

```
1 .class public LNano03;
2 .super Ljava/lang/Object;
3 .source "Nano03.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18    .registers 1
19
20    .prologue
21    .line 4
22    .line 5
23    return-void
24 .end method
```

Saída : nenhuma.

3.4 Nano04

Listagem 3.10: Nano04.py

```
1 def main() -> None:
2     n: int = 1 + 2
```

Listagem 3.11: Nano04.java

```
1 public class Nano04 {
2     public static void main(String[] args) {
3        int n;
4        n = 1+2;
5     }
6 }
```

Listagem 3.12: Nano04.smali

```
.class public LNano04;
.super Ljava/lang/Object;
.source "Nano04.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 1

    .prologue
    .line 4
    .line 5
    return-void
.end method
```

Saída : nenhuma.

3.5 Nano05

Listagem 3.13: Nano05.py

```
def main() -> None:
    n: int = 2
    print(n)

main()
```

Listagem 3.14: Nano05.java

```
public class Nano05 {
    public static void main(String[] args) {
        int n;
        n = 2;
        System.out.println(n);
    }
}
```

Listagem 3.15: 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
    const/4 v0, 0x2

    .line 5
    sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;

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

    .line 6
    return-void
.end method
```

Saída : 2.

3.6 Nano06

Listagem 3.16: Nano06.py

```
def main() -> None:
    n: int = 1 - 2
    print(n)

main()
```

Listagem 3.17: Nano06.java

```
public class Nano06 {
    public static void main(String[] args) {
        int n;
        n = 1-2;
        System.out.println(n);
    }
}
```

Listagem 3.18: Nano06.smali

```
.class public LNano06;
.super Ljava/lang/Object;
.source "Nano06.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 3
19
20      .prologue
21      .line 4
22      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
```

Saída : -1.

3.7 Nano07

Listagem 3.19: Nano07.py

```
1 def main() -> None:
2     n = 1
3     if n == 1:
4         print(n)
5
6 main()
```

Listagem 3.20: 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     }
9 }
```

Listagem 3.21: Nano07.smali

```
1 .class public LNano07;
2 .super Ljava/lang/Object;
3 .source "Nano07.java"
4
5
```

```
# 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

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

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

    .line 8
    return-void
.end method
```

Saída : 1.

3.8 Nano08

Listagem 3.22: Nano08.py

```
def main() -> None:
    n: int = 1
    if n == 1:
        print(n)
    else:
        print(0)

main()
```

Listagem 3.23: Nano08.java

```
public class Nano08 {
    public static void main(String[] args) {
        int n;
        n = 1;
        if(n==1){
            System.out.println(n);
        }else{
            System.out.println(0);
        }
    }
}
```

Listagem 3.24: Nano08.smali

```
.class public LNano08;
.super Ljava/lang/Object;
.source "Nano08.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

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

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

    .line 10
    return-void
.end method
```

Saída : 1.

3.9 Nano09

Listagem 3.25: Nano09.py

```
def main() -> None:
    n: int = 1 + (1 / 2)
    if n == 1:
        print(n)
    else:
        print(0)

main()
```

Listagem 3.26: Nano09.java

```
public class Nano09 {
    public static void main(String[] args) {
        int n;

        n = 1 + 1 / 2;
        if(n==1){
```

```
7               System.out.println(n);
8           } else {
9               System.out.println(0);
10          }
11      }
12  }
```

Listagem 3.27: Nano09.smali

```
1  .class public LNano09;
2  .super Ljava/lang/Object;
3  .source "Nano09.java"
4
5
6  # direct methods
7  .method public constructor <init>()V
8      .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 5
22     const/4 v0, 0x1
23
24     .line 7
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 11
30     return-void
31 .end method
```

Saída : 0.

3.10 Nano10

Listagem 3.28: Nano10.py

```
1 def main() -> None:
2       n: int = 1
3       m: int = 2
4       if n == m:
5           print(n)
6       else:
7           print(0)
8
9 main()
```

Listagem 3.29: 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    }
12
13 }
```

Listagem 3.30: 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 11
30    return-void
31 .end method
```

Saída : 0.

3.11 Nano11

Listagem 3.31: Nano11.py

```
1 def main() -> None:
```

```
    n: int = 1
    m: int = 2
    x: int = 5
    while x > n:
        n = n + m
        print(n)

main()
```

Listagem 3.32: Nano11.java

```
public class Nano11 {
    public static void main(String[] args) {
        int n, m, x;

        n = 1;
        m = 2;
        x = 5;
        while (x>n) {
            n = n + m;
            System.out.println(n);
        }
    }
}
```

Listagem 3.33: Nano11.smali

```
.class public LNano11;
.super Ljava/lang/Object;
.source "Nano11.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 5
    const/4 v0, 0x1

    .line 6
    const/4 v1, 0x2

    .line 7
    const/4 v2, 0x5

    .line 8
    :goto_3
    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
```

Saída : 3.
Saída : 5.

3.12 Nano12

Listagem 3.34: Nano12.py

```
1  def main() -> None:
2      n: int = 1
3      m: int = 2
4      x: int = 5
5      while x > n:
6          if n == m:
7              print(n)
8          else:
9              print(0)
10         x = x - 1
11
12 main()
```

Listagem 3.35: Nano12.java

```
1  public class Nano12 {
2      public static void main(String[] args) {
3          int n, m, x;
4          n = 1;
5          m = 2;
6          x = 5;
7          while(x>n){
8              if(n==m)
9                  System.out.println(n);
10             else
11                 System.out.println(0);
12             x = x - 1;
13         }
14     }
15 }
```

Listagem 3.36: Nano12.smali

```
1  .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
    const/4 v1, 0x1

    .line 6
    const/4 v0, 0x5

    .line 7
    :goto_2
    if-le v0, v1, :cond_d

    .line 11
    sget-object v2, Ljava/lang/System;->out:Ljava/io/PrintStream;

    const/4 v3, 0x0

    invoke-virtual {v2, v3}, Ljava/io/PrintStream;->println(I)V

    .line 12
    add-int/lit8 v0, v0, -0x1

    goto :goto_2

    .line 14
    :cond_d
    return-void
.end method
```

Saída : 0.
Saída : 0.
Saída : 0.
Saída : 0.

Capítulo 4

Analisador Léxico

Esse capítulo irá abordar como foi criado o analizador léxico para a linguagem Python.

A análise léxica é a primeira fase do compilador, e tem como tarefa analisar um alfabeto de uma determinada linguagem. Após receber uma sequência de caracteres, ele produz uma sequência de nomes, palavras-chaves e sinais de pontuação chamados de *tokens*. Ainda nessa fase, é de reponsabilidade do analisador o descarte de elementos "decorativos" do programa, tais como espaços em branco e comentários entre os tokens.

A ferramenta utilizada para construir o analisador léxico foi o ocamllex, que cria um analisador, muito semelhante ao funcionamento de um Automato Finito Deterministico, a partir de um conjunto de expressões regulares e ações semânticas para tais regras.

4.1 Lista de Tokens

Tipo	Representação	Tipo	Representação
FALSE	False	MODULO	%
NONE	None	EXP	**
TRUE	True	EQUIVALENTE	==
AND	and	NAOEQUIVALENTE	!=
AS	as	MENOR	«
BREAK	break	MAIOR	»
CONTINUE	continue	MENORIGUAL	«=
DEF	def	MAIORIGUAL	»=
DEL	del	IGUAL	=
ELIF	elif	APAR	(
ELSE	else	FPAR	)
EXCEPT	except	ACOLCHETE	[
FOR	for	FCOLCHETE	]
FROM	from	ACHAVE	{
IF	if	FCHAVE	}
IMPORT	import	PONTO	.
IN	in	VIRG	,
IS	is	DPONTOS	:
NOT	not	PVIRG	;
OR	or	ARROBA	@
RETURN	return	SOMAIGUAL	+=
WHILE	while	SUBIGUAL	-=
WITH	with	MULTIGUAL	*=
SOMA	+	DIVIGUAL	/=
SUB	-	DIVINTIGUAL	//=
MULT	*	MODULOIGUAL	%=
DIV	/	EXPIGUAL	**/
DIVINT	//		

Tabela 4.1: *Tabela de Tokens*

4.2 Códigos

Foi incluido o código pre_processador.ml, como foi pedido pelo professor para quem estevesse fazendo o trabalho em Python.

Listagem 4.1: lexico.mll

```
{

  open Lexing
  open Printf

  type token =
  | LITINT of (int)
  | LITSTRING of (string)
  | ID of (string)
  | FALSE
  | NONE
  | TRUE
  | AND
  | AS
  | BREAK
  | CONTINUE
  | DEF
  | DEL
  | ELIF
  | ELSE
  | EXCEPT
  | FOR
  | FROM
  | IF
  | IMPORT
  | IN
  | IS
  | NOT
  | OR
  | RETURN
  | WHILE
  | WITH
  | SOMA
  | SUB
  | MULT
  | DIV
  | DIVINT
  | MODULO
  | EXP
  | EQUIVALENTE
  | NAOEQUIVALENTE
  | MENOR
  | MAIOR
  | MENORIGUAL
  | MAIORIGUAL
  | IGUAL
  | APAR
  | FPAR
  | ACOLCHETE
  | FCOLCHETE
  | ACHAVE
  | FCHAVE
```

```
53    | PONTO
54    | VIRG
55    | DPONTOS
56    | PVIRG
57    | ARROBA
58    | SOMAIGUAL
59    | SUBIGUAL
60    | MULTIGUAL
61    | DIVIGUAL
62    | DIVINTIGUAL
63    | MODULOIGUAL
64    | EXPIGUAL
65    | SETA
66    | E
67    | ATRIB
68    | EOF
69    | Linha of (int * int * token list)
70    | INDENTA
71    | DEDENTA
72    | NOVALINHA
73
74
75    let nivel_par = ref 0
76
77    let incr_num_linha lexbuf =
78      let pos = lexbuf.lex_curr_p in
79        lexbuf.lex_curr_p <- { pos with
80           pos_lnum = pos.pos_lnum + 1;
81           pos_bol = pos.pos_cnum;
82        }
83
84    let msg_erro lexbuf c =
85      let pos = lexbuf.lex_curr_p in
86      let lin = pos.pos_lnum
87      and col = pos.pos_cnum - pos.pos_bol - 1 in
88      sprintf "%d-%d: caracter desconhecido %c" lin col c
89
90  }
91
92  let digito = ['0' - '9']
93  let int =  digito+
94
95  let comentario = "#"[ ^ '\n' ]*
96
97  let linha_em_branco = [' ' '\t' ]* comentario
98  let restante = [^ ' ' '\t' '\n' ] [^ '\n']+
99  let brancos = [' ' '\t']+
100 let novalinha = '\r' | '\n' | "\r\n"
101
102 let letra = [ 'a'-'z' 'A' - 'Z']
103 let identificador = letra ( letra | digito | '_' )*
104
105 rule preprocessador indentacao = parse
106   linha_em_branco          { preprocessador 0 lexbuf }
107 | [' ' '\t' ]+ '\n'        { incr_num_linha lexbuf;
108                              preprocessador 0 lexbuf }
109 | ' '                      { preprocessador (indentacao + 1) lexbuf }
110 | '\t'                     { let nova_ind = indentacao + 8 - (indentacao
      mod 8)
```

```
111                             in preprocessador nova_ind lexbuf }
112 | novalinha                 { incr_num_linha lexbuf;
113                               preprocessador 0 lexbuf }
114 | restante as linha {
115       let rec tokenize lexbuf =
116           let tok = token lexbuf in
117     match tok with
118        EOF -> []
119    | _ -> tok :: tokenize lexbuf in
120       let toks = tokenize (Lexing.from_string linha) in
121       Linha(indentacao,!nivel_par, toks)
122 }
123 | eof { nivel_par := 0; EOF }
124
125 and token = parse
126   brancos             { token lexbuf }
127 | comentario          { token lexbuf }
128 | "'''"          { comentario_bloco 0 lexbuf }
129 | "False"        { FALSE }
130 | "None"         { NONE }
131 | "True"         { TRUE }
132 | "and"               { AND }
133 | "as"             { AS }
134 | "break"        { BREAK }
135 | "continue"           { CONTINUE}
136 | "def"              { DEF }
137 | "del"              { DEL }
138 | "elif"             { ELIF }
139 | "else"         { ELSE }
140 | "except"           { EXCEPT }
141 | "for"          { FOR }
142 | "from"             { FROM }
143 | "if"               { IF }
144 | "import"           { IMPORT }
145 | "in"           { IN }
146 | "is"           { IS }
147 | "not"          { NOT }
148 | "or"               { OR }
149 | "return"          { RETURN }
150 | "while"         { WHILE }
151 | "with"         { WITH }
152 | "+"                { SOMA }
153 | "-"          { SUB }
154 | "*"              { MULT }
155 | "/"              { DIV }
156 | "//"           { DIVINT }
157 | "%"          { MODULO }
158 | "**"               { EXP }
159 | "=="               { EQUIVALENTE }
160 | "!="               { NAOEQUIVALENTE }
161 | "<"              { MENOR }
162 | ">"              { MAIOR }
163 | "<="               { MENORIGUAL }
164 | ">="           { MAIORIGUAL }
165 | "="            { IGUAL }
166 | "("              { APAR }
167 | ")"              { FPAR }
168 | "["              { ACOLCHETE }
169 | "]"              { FCOLCHETE }
```

```
170 | "{"               { ACHAVE }
171 | "}"               { FCHAVE }
172 | "."               { PONTO }
173 | ","         { VIRG }
174 | ":"         { DPONTOS }
175 | ";"               { PVIRG }
176 | "@"               { ARROBA }
177 | "+="                { SOMAIGUAL }
178 | "-="                { SUBIGUAL }
179 | "*="                { MULTIGUAL }
180 | "/="                { DIVIGUAL }
181 | "//="               { DIVINTIGUAL }
182 | "%="                { MODULOIGUAL }
183 | "**/"         { EXPIGUAL }
184
185 | int as num         { let numero = int_of_string num in
186                         LITINT numero }
187
188 | '"'                { let buffer = Buffer.create 1 in
189                         let str = leia_string buffer lexbuf in
190                             LITSTRING str }
191
192 | identificador as id { ID id }
193 | _ as c              { failwith (msg_erro lexbuf c); }
194 | eof                 { EOF }
195
196 and comentario_bloco n = parse
197       "'''" { if n=0 then token lexbuf
198               else comentario_bloco (n - 1) lexbuf }
199     | "'''" { comentario_bloco (n + 1) lexbuf; }
200     | _    { comentario_bloco n lexbuf }
201     | eof  { failwith "Comentário não fechado" }
202
203 and leia_string buffer = parse
204     | '"'       {Buffer.contents buffer }
205     | "\\t"     { Buffer.add_char buffer '\t'; leia_string buffer lexbuf }
206     | "\\n"     { Buffer.add_char buffer '\n'; leia_string buffer lexbuf }
207     | '\\' '"'  { Buffer.add_char buffer '"'; leia_string buffer lexbuf }
208     | '\\' '\\' { Buffer.add_char buffer '\\'; leia_string buffer lexbuf }
209     | _ as c    { Buffer.add_char buffer c; leia_string buffer lexbuf }
210     | eof       { failwith "A string não foi fechada." }
```

Listagem 4.2: pre-processador.ml

```
1  open Lexico
2  open Printf
3
4  (* Pré processa o arquivo gerando os tokens de indenta e dedenta *)
5
6  let preprocessa lexbuf =
7    let pilha = Stack.create ()
8    and npar = ref 0 in
9      let _ = Stack.push 0 pilha in
10     let off_side toks nivel =
11     let _ = printf "Nivel: %d\n" nivel in
12     if !npar != 0 (* nova linha entre parenteses *)
13     then toks     (* nao faz nada *)
14     else if nivel > Stack.top pilha
15          then begin
```

```
16            Stack.push nivel pilha;
17            INDENTA :: toks
18          end
19     else if nivel = Stack.top pilha
20          then toks
21     else begin
22     let prefixo = ref toks in
23     while nivel < Stack.top pilha do
24       ignore (Stack.pop pilha);
25       if nivel > Stack.top pilha
26         then failwith "Erro de indentacao"
27       else prefixo := DEDENTA :: !prefixo
28    done;
29    !prefixo
30    end
31  in
32
33  let rec dedenta sufixo =
34    if Stack.top pilha != 0
35    then let _ = Stack.pop pilha in
36         dedenta (DEDENTA :: sufixo)
37    else sufixo
38  in
39  let rec get_tokens () =
40    let tok = Lexico.preprocessador 0 lexbuf in
41    match tok with
42      Linha(nivel,npars,toks) ->
43      let new_toks = off_side toks nivel in
44      npar := npars;
45      new_toks @ (if npars = 0
46                  then NOVALINHA :: get_tokens ()
47                  else get_tokens ())
48     | _ -> dedenta []
49  in get_tokens ()
50
51
52 (* Chama o analisador léxico *)
53 let lexico =
54   let tokbuf = ref None in
55   let carrega lexbuf =
56     let toks = preprocessa lexbuf in
57     (match toks with
58        tok::toks ->
59        tokbuf := Some toks;
60        tok
61      | [] -> print_endline "EOF";
62        EOF)
63   in
64   fun lexbuf ->
65   match !tokbuf with
66     Some tokens ->
67     (match tokens with
68        tok::toks ->
69        tokbuf := Some toks;
70        tok
71      | [] -> carrega lexbuf)
72   | None -> carrega lexbuf
```

Listagem 4.3: carregador.ml

```
#load "lexico.cmo"
#load "pre_processador.cmo"

type nome_arq = string
type tokens = Lexico.token list

let rec tokens lexbuf =
  let tok = Pre_processador.lexico lexbuf in
  match tok with
  | Lexico.EOF -> (|Lexico.EOF):tokens)
  | _ -> tok :: tokens lexbuf
;;

let lexico str =
  let lexbuf = Lexing.from_string str in
  tokens lexbuf
;;

let lex (arq:nome_arq)=
  let ic = open_in arq in
  let lexbuf = Lexing.from_channel ic in
  let toks = tokens lexbuf in
  let _ = close_in ic in
  toks
```

4.3 Compilação e execução

Para compilar navague pelo terminal ate o diretório dos arquivos e execute os comandos:

```
>ocamllex lexico.mll
>ocamlc -c lexico.ml
>ocamlc -c pre_processador.ml
```

Após a compilação execute:

```
>lrwrap ocaml
```

Dentro do *O caml* execute os comandos:

```
# #use "carregador.ml";;
# lex "codigo.py";;
```

Um exemplo com o arquivo teste.py com o código:

Listagem 4.4: teste.py

```
def paco(x):
  x = x + 1
  y = x + 2
  y = x + 3 + ( 4 + 5)
  return x
```

A saída do analisador léxico:

```
- : tokens =
  [Lexico.DEF; Lexico.ID "paco"; Lexico.APAR; Lexico.ID "x"; Lexico.FPAR;
  Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "x";
  Lexico.IGUAL; Lexico.ID "x"; Lexico.SOMA; Lexico.LITINT 1; Lexico.
     NOVALINHA;
  Lexico.ID "y"; Lexico.IGUAL; Lexico.ID "x"; Lexico.SOMA; Lexico.LITINT
     2;
  Lexico.NOVALINHA; Lexico.ID "y"; Lexico.IGUAL; Lexico.ID "x"; Lexico.
     SOMA;
  Lexico.LITINT 3; Lexico.SOMA; Lexico.APAR; Lexico.LITINT 4; Lexico.SOMA;
  Lexico.LITINT 5; Lexico.FPAR; Lexico.NOVALINHA; Lexico.RETURN;
  Lexico.ID "x"; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

4.4 Analise léxica Nanos

Analise léxica dos nanos programas em python vistos no capitulo anterior.

Listagem 4.5: nano01.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
      INDENTA;
3  Lexico.RETURN; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

Listagem 4.6: nano02.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
      INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 0; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

Listagem 4.7: nano03.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
      INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]
```

Listagem 4.8: nano04.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
      INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.SOMA; Lexico.LITINT 2; Lexico.NOVALINHA;
5  Lexico.DEDENTA; Lexico.EOF]
```

Listagem 4.9: nano05.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
      INDENTA;
```

```
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 2; Lexico.NOVALINHA; Lexico.ID "print"; Lexico.APAR;
5  Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
6  Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.10: nano06.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.SUB; Lexico.LITINT 2; Lexico.NOVALINHA;
5  Lexico.ID "print"; Lexico.APAR; Lexico.ID "n"; Lexico.FPAR;
6  Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR;
7  Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.11: nano07.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.IGUAL; Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF
       ;
4  Lexico.ID "n"; Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.DPONTOS;
5  Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR;
6  Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
7  Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
8  Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.12: nano08.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n";
5  Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.NOVALINHA;
6  Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR; Lexico.ID "n"; Lexico.
       FPAR;
7  Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.DPONTOS;
8  Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR;
9  Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
10 Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
11 Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.13: nano09.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.SOMA; Lexico.APAR; Lexico.LITINT 1; Lexico.DIV;
5  Lexico.LITINT 2; Lexico.FPAR; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n";
6  Lexico.EQUIVALENTE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.NOVALINHA;
7  Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR; Lexico.ID "n"; Lexico.
       FPAR;
8  Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.DPONTOS;
9  Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR;
```

```
10  Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
11  Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
12  Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.14: nano10.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.ID "m"; Lexico.DPONTOS;
5  Lexico.ID "int"; Lexico.IGUAL; Lexico.LITINT 2; Lexico.NOVALINHA; Lexico.
       IF;
6  Lexico.ID "n"; Lexico.EQUIVALENTE; Lexico.ID "m"; Lexico.DPONTOS;
7  Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print"; Lexico.APAR;
8  Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE
       ;
9  Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print";
10 Lexico.APAR; Lexico.LITINT 0; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
       DEDENTA;
11 Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
12 Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.15: nano11.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.ID "m"; Lexico.DPONTOS;
5  Lexico.ID "int"; Lexico.IGUAL; Lexico.LITINT 2; Lexico.NOVALINHA;
6  Lexico.ID "x"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
7  Lexico.LITINT 5; Lexico.NOVALINHA; Lexico.WHILE; Lexico.ID "x";
8  Lexico.MAIOR; Lexico.ID "n"; Lexico.DPONTOS; Lexico.NOVALINHA;
9  Lexico.INDENTA; Lexico.ID "n"; Lexico.IGUAL; Lexico.ID "n"; Lexico.SOMA;
10 Lexico.ID "m"; Lexico.NOVALINHA; Lexico.ID "print"; Lexico.APAR;
11 Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA;
12 Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR;
13 Lexico.NOVALINHA; Lexico.EOF]
```

Listagem 4.16: nano12.py

```
1 [Lexico.DEF; Lexico.ID "main"; Lexico.APAR; Lexico.FPAR; Lexico.SUB;
2  Lexico.MAIOR; Lexico.NONE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.
       INDENTA;
3  Lexico.ID "n"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
4  Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.ID "m"; Lexico.DPONTOS;
5  Lexico.ID "int"; Lexico.IGUAL; Lexico.LITINT 2; Lexico.NOVALINHA;
6  Lexico.ID "x"; Lexico.DPONTOS; Lexico.ID "int"; Lexico.IGUAL;
7  Lexico.LITINT 5; Lexico.NOVALINHA; Lexico.WHILE; Lexico.ID "x";
8  Lexico.MAIOR; Lexico.ID "n"; Lexico.DPONTOS; Lexico.NOVALINHA;
9  Lexico.INDENTA; Lexico.IF; Lexico.ID "n"; Lexico.EQUIVALENTE; Lexico.ID "
       m";
10 Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "print";
11 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA
       ;
12 Lexico.ELSE; Lexico.DPONTOS; Lexico.NOVALINHA; Lexico.INDENTA;
13 Lexico.ID "print"; Lexico.APAR; Lexico.LITINT 0; Lexico.FPAR;
```

```
14 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "x"; Lexico.IGUAL;
15 Lexico.ID "x"; Lexico.SUB; Lexico.LITINT 1; Lexico.NOVALINHA;
16 Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID "main"; Lexico.APAR; Lexico.
       FPAR;
17 Lexico.NOVALINHA; Lexico.EOF]
```

Capítulo 5

Referências

Python

Dalvik

Smali

OCaml