

Universidade Federal de Uberlândia  
Ciência da Computação  
Construção de Compiladores

# Construção do mini compilador Python para CLR

Aluno: Arthur Xavier Giffoni  
E-mail: arthurxaviergiffoni@gmail.com  
Professor:Alexsandro Santos Soares

Abril, 2017

# Sumário

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Arquitetura</b>	<b>2</b>
<b>3</b>	<b>Instruções</b>	<b>3</b>
<b>4</b>	<b>Instalações</b>	<b>7</b>
4.1	Instalação do Mono . . . . .	7
4.2	Instalação do OCaml . . . . .	7
4.3	Instalação do Python . . . . .	7
<b>5</b>	<b>Gerando a Tradução dos Programas</b>	<b>7</b>
<b>6</b>	<b>Códigos: MiniPython - C# - Assembly</b>	<b>8</b>
6.1	nano01 . . . . .	8
6.2	nano02 . . . . .	9
6.3	nano03 . . . . .	10
6.4	nano04 . . . . .	12
6.5	nano05 . . . . .	13
6.6	nano06 . . . . .	14
6.7	nano07 . . . . .	16
6.8	nano08 . . . . .	17
6.9	nano09 . . . . .	19
6.10	nano10 . . . . .	21
6.11	nano11 . . . . .	23
6.12	nano12 . . . . .	25
6.13	micro01 . . . . .	27
<b>7</b>	<b>Analizador Léxico</b>	<b>28</b>
7.1	Abordagem por Autômato . . . . .	28
7.1.1	Implementação . . . . .	33
7.1.2	Execução . . . . .	38
7.2	Abordagem por Linguagem Regular . . . . .	39
7.2.1	Implementação . . . . .	40
7.2.2	Execução . . . . .	40
7.2.3	Erros Léxicos . . . . .	47
<b>8</b>	<b>Analizador Sintático</b>	<b>49</b>
8.1	Analizador Abordado em Aula . . . . .	49
8.1.1	Gramática . . . . .	49
8.1.2	Código . . . . .	49
8.1.3	Execução . . . . .	52
8.2	Analizador sintático Mini-Python usando Menhir . . . . .	53
8.2.1	Execução de Testes . . . . .	53

# 1 Introdução

Common Language Runtime - CLR, é um ambiente de tempo de execução do .NET Framework, o qual que executa o código e provê serviços que tornam o processo de desenvolvimento mais fácil. Compiladores e ferramentas expõem as funcionalidades do Common Language Runtime e habilitam você escrever código que se beneficia desse ambiente de execução gerenciado. Código que você desenvolve com um compilador de linguagem que tem como alvo o runtime é chamado de código gerenciado; ele se beneficia de recursos como integração entre linguagens, tratamento de exceção entre linguagens, segurança aprimorada, suporte a versionamento e implantação, um modelo simplificado para interação entre componentes, e serviços de depuração e de perfil.

O CLR descreve o código executável e o ambiente de execução que formam o núcleo do Framework .NET da Microsoft, do MONO e do Portable.NET. A Common Language Runtime é uma máquina virtual que segue um padrão internacional e a base para a criação e execução de ambientes de desenvolvimento em que as linguagens e as bibliotecas trabalham juntos.

Programas em linguagem Visual Basic, Visual C++ ou C# são compilados em um formulário intermediário de codificar chamado Common Intermediate Language (CIL) em um arquivo de execução portátil (PE) que pode, em seguida, ser gerenciado e executado pelo Common Language Runtime.

# 2 Arquitetura

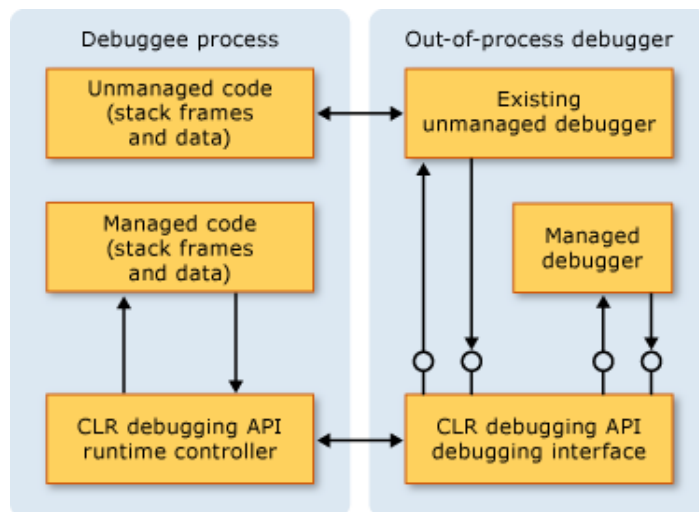
O common language runtime (CLR) API de depuração foi projetado para ser usado como se fosse parte do kernel do sistema operacional. Em código não gerenciado, quando um programa gera uma exceção, o kernel suspende a execução do processo e passa as informações de exceção para o depurador usando o Win32 API de depuração. API de depuração CLR fornece a mesma funcionalidade para código gerenciado. Quando o código gerenciado gera uma exceção, a API de depuração CLR suspende a execução do processo e passa as informações de exceção para o depurador.

API de depuração CLR inclui dois componentes principais:

- A DLL de depuração, que é sempre carregada no mesmo processo como o programa que está sendo depurado. O controlador de tempo de execução é responsável pela comunicação com o CLR e executar o controle de execução e inspeção de threads que estejam executando o código gerenciado.
- A interface do depurador, que é carregada em um processo diferente do programa que está sendo depurado. A interface do depurador é responsável pela comunicação com o controlador de tempo de execução em nome do depurador. Também é responsável pela manipulação de eventos de depuração do Win32 que vêm do processo que está sendo depurado e ambos lidando com esses eventos ou passá-los para um depurador de código não

gerenciado. A interface do depurador é a única parte da API que tem uma API exposta de depuração CLR.

O CLR API de depuração não oferece suporte ao uso remoto entre computadores ou entre processos; ou seja, um depurador que usa a API deverá fazê-lo de dentro de seu próprio processo, conforme a figura no seguinte diagrama de arquitetura da API. Esta ilustração mostra onde os diferentes componentes do CLR API de depuração estão localizados e como eles interagem com o CLR e o depurador.



### 3 Instruções

O assembly da plataforma, MSIL, é um conjunto de instruções baseado em pilha e orientado a objetos. As funcionalidade de algumas instruções em relação a manipulação da pilha listadas a seguir:

- **add:** retira os dois elementos do topo da pilha e coloca o resultado no topo.
- **add.ovf:** mesma operação que o **add** porém com uma verificação de overflow(gerando uma exceção de overflow).
- **and:** retira os dois elementos do topo da pilha, faz a operação **and** bit a bit e coloca o resultado no topo da pilha.
- **arglist:** pega argumento da lista (usado em para pegar argumentos de função)
- **beq.[length]:** branch para label se o dois valores da pilha são iguais
- **bge.[length]:** branch para label se o primeiro valor é menor que o segundo da pilha

- bge.un.[length]: branch para label se maior ou igual, comparação sem sinal
- bgt.[length]: branch para label quando o segundo valor é maior que o primeiro
- bgt.un.[length]: branch para label quando o topo é maior que o segundo valor, sem sinal
- ble.[length]: branch para label quando o topo é menor ou igual ao segundo valor
- ble.un.[length]: branch para label se topo é menor ou igual ao segundo valor, sem sinal
- blt.[length]: branch para label quando quando o topo é menor que o segundo valor
- blt.un.[length]: branch para label se topo é menor que segundo valor, sem sinal
- bne.un[length]: branch para label quando topo não for igual ou não ordenada
- br.[length]: branch incondicional
- break: instrução breakpoint
- brfalse.[length]: branch para label se falso, nulo, ou zero
- brtrue.[length]: branch para label quando não falseo ou não nulo
- call: chama um método
- calli: chama um método indireto
- ceq: compara se igual
- cgt: compara se maior que
- cgt.un: compara maior que,sem sinal e não ordenado
- ckfinite: Checa se é um número real e finito
- clt: compara se menor que
- clt.un: compara se menor que, sem sinal e não ordenado
- conv.[to type]: conversão de dados
- conv.ovf.[to type]: conversão de dados com detecção de overflow
- conv.ovf.[to type].un: conversão de dados sem sinal com detecção de overflow

- cpblk: copia dados da memória para a memória
- div: divide valores
- div.un: divide valores inteiros, sem sinal
- dup: duplica o valor do topo da pilha
- endfilter: fim do filtro da cláusula de SEH
- endfinally: finaliza a cláusula do bloco de exceção
- initblk: inicializa um bloco de memória para um valores
- jmp: pula para um método
- jmp: pula para um ponteiro de método
- ldarg.[length]: carrega um argumento na pilha
- ldarga.[length]: carrega um argumento a partir de um endereço
- ldc.[type]: carrega uma constante numérica
- ldftn : carrega um ponteiro de método
- ldind.[type]: carrega um valor indireto na pilha
- ldloc: load local variable onto the stack
- ldloc: carrega na pilha o valor da variavel local.
- ldloc index: carrega na pilha o valor da variavel local com index.
- ldnull: carrega na pilha um ponteiro pra null
- leave target: sai de uma região protegida do código
- localloc: aloca um espaço na memory pool do tamanho do primeiro elemento da pilha e retorna o endereço da área.
- mul: multiplica os dois valores do topo da pilha(retirando-os) e coloca o resultado.
- mul.ovf.[type]: multiplica valores inteiros levando em conta o overflow.
- neg: retira o valor do topo da pilha, nega ele e põe o resultado no topo, retornando o mesmo tipo de operando.
- nop: faz nada :D
- not: retira um inteiro e coloca seu complemento(inversão de bits) na pilha.
- or: faz o OU bit a bit dos dois valores inteiros no topo da pilha(retirando-os) e coloca o resultado.

- pop: remove o elemento do topo da pilha.
- rem: computa o resto da divisão do valor abaixo do topo da pilha pelo valor que está no topo(retirando-os) e coloca o resultado no topo.
- rem.un: mesmo que o rem só que para inteiros unsigned.
- ret: retorna para o método corrente, o tipo de retorno do método em questão será o utilizado.
- shl: Deslocamento de inteiro para a esquerda..., valor, qntd d e d deslocamento retira estes dois da pilha e coloca o resultado.
- shr: mesmo que o shl porém com deslocamento para a direita.
- shr.un: mesmo que shr só que para inteiros unsigned.
- starg.[length]: retira o elemento do topo da pilha e o coloca em um argumento(starg num).
- stind.[type]: coloca o valor(topo da pilha) no endereço (logo abaixo).
- stloc: retira o valor do topo da pilha e o põe na variável(stloc x)
- sub: subtrai do segundo valor o primeiro e põe o resultado no topo.
- sub.ovf.[type]: subtração de inteiros com overflow
- tail.: deve preceder imediatamente instruções de call, calli ou callvirt. Ela indica que o método corrente na pilha deve ser removido antes da chamada da função ser executada.
- unaligned. (prefix code): especifica que o endereço na pilha não está alinhado ao tamanho natural.
- volatile. (prefix code): especifica que o endereço no topo da pilha é volátil.
- xor: executa a operação XOR(bit a bit) entre os dois primeiros elementos da pilha colocando seu resultado na mesma.
- ldelem.[type]: carrega um elemento do vetor(segunda posição) no index(primeira posição, topo da pilha) na pilha
- ldelem: põe o endereço do vetor na posição index na pilha.
- lden: põe na pilha o tamanho do vetor(topo da pilha)
- ldstr: põe a string(ldstr string) no topo da pilha
- newarr: cria um array com o tipo definido(newarr int32) onde seu tamanho está no topo da pilha.
- sizeof: carrega o tamanho em bits do tipo definido(sizeof int32) na pilha.
- stelem.[type]: coloca no array(terceiro da pilha) no index(segunda da pilha) o valor(topo da pilha).

## 4 Instalações

Nessa seção sera listados os comandos para instalação das ferramentas necessárias para manipulação e execução dos códigos. Antes de iniciar qualquer instalação no Linux, devemos atualizar nosso pacote utilizando o código:

```
sudo apt-get update
```

Apos atualizado, podemos instalar as ferramentas especificas.

### 4.1 Instalação do Mono

Patrocinado pela Microsoft, Mono é uma implementação de código aberto do Microsoft .NET Framework baseado nos padrões ECMA para C # e Common Language Runtime. Para instalar o Mono com apt-get utiliza-se o pacote:

```
sudo apt-get install mono-complete
```

### 4.2 Instalação do OCaml

Para instalar o OCaml basta digitar o seguinte comando no terminal:

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

### 4.3 Instalação do Python

Nesse trabalho iremos utilizar o Python 2.7. Para instalar basta digitar o seguinte comando no terminal:

```
sudo apt-get install python2.7
```

## 5 Gerando a Tradução dos Programas

Os programas em Python devem terminar com a Extensão ".py"

Para a transformação do código o mesmo deve ser convergido para C # para depois ser lido o assembly.

Os programas em C # devem terminar com a Extensão ".cs"



Executar o código Python

```
python nanoC.py
```

Compilar o código C # e gerar o .exe

```
dmcs nanoC.cs
```

Gerar assembly em cima do executável C # .exe

```
monodis nanoC.exe > nanoC.txt
```

## 6 Códigos: MiniPython - C# - Assembly

### 6.1 nano01

MiniPython nano01

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

MiniC# nano01

```
1 using System;  
2  
3 namespace nano01  
4 {  
5     class Program  
6     {  
7         static void Main(String[] args){}  
8     }  
9 }
```

Assembly nano01

```
1 .assembly extern mscorlib{  
2  
3 .assembly 'nano01'  
4 {  
5     .ver 0:0:0:0  
6 }  
7 .module nano01.exe  
8  
9 .namespace nano01  
10 {  
11     .class private auto ansi beforefieldinit Program
```

```

12     extends [mscorlib]System.Object
13 {
14
15     .method public hidebysig specialname rtspecialname
16         instance default void '.ctor' () cil managed
17     {
18         .maxstack 8
19         ldarg.0
20         call instance void object::.ctor'()
21         ret
22     }
23
24     // method line 2
25     .method private static hidebysig
26         default void Main () cil managed
27     {
28         .entrypoint
29         .maxstack 0
30         ret
31     }
32 }
33 }
34 }

```

## 6.2 nano02

MiniPython nano02

```

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

```

MiniC# nano02

```

1 using System;
2
3 namespace nano02
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10        }
11    }
12 }

```

Assembly nano02

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano02'

```

```

4 {
5     .ver 0:0:0:0
6 }
7 .module nano02.exe
8
9 .namespace nano02
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (int32 V_0)
31             ret
32         }
33     }
34 }
35 }

```

### 6.3 nano03

MiniPython nano03

```

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

```

MiniC# nano03

```

1 using System;
2
3 namespace nano03
4 {
5     class Program
6     {
7         static void Main(String[] args)

```

```

8      {
9          int n;
10         n = 1;
11     }
12 }
13
14 }

```

Assembly nano03

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano03'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano03.exe
8
9 .namespace nano03
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (int32 V_0)
31             ldc.i4.1
32             stloc.0
33             ret
34         }
35     }
36 }
37 }

```

## 6.4 nano04

MiniPython nano04

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

MiniC# nano04

```
1 using System;
2
3 namespace nano04
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            n = 1 + 2;
11        }
12    }
13
14 }
```

Assembly nano04

```
1 .assembly extern mscorlib{}
2
3 .assembly 'nano04'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano04.exe
8
9 .namespace nano04
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
```

```

26         default void Main () cil managed
27     {
28         .entrypoint
29         .maxstack 0
30         .locals init (int32 V_0)
31         ldc.i4.3
32         stloc.0
33         ret
34     }
35
36 }
37 }

```

## 6.5 nano05

MiniPython nano05

```

1 def nano05():
2     n = 2
3     print(n,end=" ")
4
5 nano05()

```

MiniC# nano05

```

1 using System;
2
3 namespace nano05
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            n = 2;
11            Console.WriteLine(n);
12        }
13    }
14
15 }

```

Assembly nano05

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano05'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano05.exe
8

```

```

9  .namespace nano05
10 {
11  .class private auto ansi beforefieldinit Program
12  extends [mscorlib]System.Object
13  {
14
15  .method public hidebysig specialname rtspecialname
16  instance default void '.ctor' () cil managed
17  {
18  .maxstack 8
19  ldarg.0
20  call instance void object::.ctor'()
21  ret
22  }
23
24  // method line 2
25  .method private static hidebysig
26  default void Main () cil managed
27  {
28  .entrypoint
29  .maxstack 0
30  .locals init (int32 V_0)
31  ldc.i4.2
32  stloc.0
33  ldloc.0
34  call void class [mscorlib]System.Console::WriteLine(
35  int32)
36  ret
37  }
38 }
39 }

```

## 6.6 nano06

MiniPython nano06

```

1 def nano06():
2     n = 1 - 2
3     print(n,end=" ")
4
5 nano06()

```

MiniC# nano06

```

1 using System;
2
3 namespace nano06
4 {
5     class Program

```

```

6      {
7          static void Main(String[] args)
8          {
9              int n;
10             n = 1 - 2;
11             Console.WriteLine(n);
12         }
13     }
14
15 }

```

Assembly nano06

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano06'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano06.exe
8
9 .namespace nano06
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (int32 V_0)
31             ldc.i4.m1
32             stloc.0
33             ldloc.0
34             call void class [mscorlib]System.Console::WriteLine(
35                 int32)
36             ret
37         }
38     }
39 }

```



```
38 }
39 }
```

## 6.7 nano07

MiniPython nano07

```
1 def nano07():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5
6 nano07()
```

MiniC# nano07

```
1 using System;
2
3 namespace nano07
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            n = 1;
11            if(n == 1)
12            {
13                Console.WriteLine(n);
14            }
15        }
16    }
17
18 }
```

Assembly nano07

```
1 .assembly extern mscorlib{}
2
3 .assembly 'nano07'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano07.exe
8
9 .namespace nano07
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
```

```

15     .method public hidebysig specialname rtspecialname
16         instance default void '.ctor' () cil managed
17     {
18         .maxstack 8
19         ldarg.0
20         call instance void object::.ctor()
21         ret
22     }
23
24     // method line 2
25     .method private static hidebysig
26         default void Main () cil managed
27     {
28         .entrypoint
29         .maxstack 0
30         .locals init (int32 V_0)
31
32         ldc.i4.1
33         stloc.0
34         ldloc.0
35         ldc.i4.1
36         bne.un IL_000f
37
38         ldloc.0
39         call void class [mscorlib]System.Console::
            WriteLine(int32)
40     IL_000f: ret
41     }
42
43 }
44 }

```

## 6.8 nano08

MiniPython nano08

```

1 def nano08():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5     else:
6         print(0,end=" ")
7
8 nano08()

```

MiniC# nano08

```

1 using System;
2
3 namespace nano08

```

```

4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            n = 1;
11            if(n == 1)
12            {
13                Console.WriteLine(n);
14            }
15            else{
16                Console.WriteLine(0);
17            }
18        }
19    }
20 }
21 }

```

Assembly nano08

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano08'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano08.exe
8
9 .namespace nano08
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (int32 V_0)

```

```

31         ldc.i4.1
32         stloc.0
33         ldloc.0
34         ldc.i4.1
35         bne.un IL_0014
36
37         ldloc.0
38         call void class [mscorlib]System.Console::
39             WriteLine(int32)
40         br IL_001a
41
42 IL_0014:     ldc.i4.0
43             call void class [mscorlib]System.Console::
44                 WriteLine(int32)
45 IL_001a:     ret
46     }
47 }
48 }

```

## 6.9 nano09

MiniPython nano09

```

1 def nano09():
2     n=1
3     if n ==1:
4         n = n + 1
5         print(n,end=" ")
6     else:
7         print(0,end=" ")
8
9 nano09()

```

MiniC# nano09

```

1 using System;
2
3 namespace nano09
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            n = 1;
11            if(n == 1)
12            {
13                n = n + 1;

```

```

14         Console.WriteLine(n);
15     }
16     else{
17         Console.WriteLine(0);
18     }
19 }
20 }
21
22 }

```

Assembly nano09

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano09'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano09.exe
8
9 .namespace nano09
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15     .method public hidebysig specialname rtspecialname
16         instance default void '.ctor' () cil managed
17     {
18         .maxstack 8
19         ldarg.0
20         call instance void object::.ctor'()
21         ret
22     }
23
24     // method line 2
25     .method private static hidebysig
26         default void Main () cil managed
27     {
28         .entrypoint
29         .maxstack 0
30         .locals init (int32 V_0)
31
32         ldc.i4.1
33         stloc.0
34         ldloc.0
35         ldc.i4.1
36         bne.un IL_0018
37
38         ldloc.0
39         ldc.i4.1

```

```

40         add
41         stloc.0
42         ldloc.0
43         call void class [mscorlib]System.Console::
           WriteLine(int32)
44         br IL_001e
45
46 IL_0018:     ldc.i4.0
47         call void class [mscorlib]System.Console::
           WriteLine(int32)
48 IL_001e:     ret
49     }
50
51 }
52 }

```

## 6.10 nano10

MiniPython nano10

```

1 def nano10():
2     n=1
3     m=2
4     if n ==m:
5         print(n,end=" ")
6     else:
7         print(0,end=" ")
8
9 nano10()

```

MiniC# nano10

```

1 using System;
2
3 namespace nano10
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            int m;
11            n = 1;
12            m = 2;
13            if(n == m)
14            {
15                Console.WriteLine(n);
16            }
17            else{
18                Console.WriteLine(0);

```

```

19     }
20   }
21 }
22
23 }

```

Assembly nano10

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano10'
4 {
5   .ver 0:0:0:0
6 }
7 .module nano10.exe
8
9 .namespace nano10
10 {
11   .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13   {
14
15     .method public hidebysig specialname rtspecialname
16       instance default void '.ctor' () cil managed
17     {
18       .maxstack 8
19       ldarg.0
20       call instance void object::.ctor'()
21       ret
22     }
23
24     // method line 2
25     .method private static hidebysig
26       default void Main () cil managed
27     {
28       .entrypoint
29       .maxstack 0
30       .locals init (
31         int32 V_0,
32         int32 V_1)
33
34       ldc.i4.1
35       stloc.0
36       ldc.i4.2
37       stloc.1
38       ldloc.0
39       ldloc.1
40       bne.un IL_0016
41
42       ldloc.0
43       call void class [mscorlib]System.Console::

```

```

        WriteLine(int32)
44        br IL_001c
45
46
47 IL_0016    ldc.i4.0
48        call void class [mscorlib]System.Console::
        WriteLine(int32)
49 IL_001c:   ret
50    }
51
52 }
53 }

```

## 6.11 nano11

MiniPython nano11

```

1 def nano11():
2     n=1
3     m=2
4     x=5
5     while x > n:
6         n = n + m
7         print(n,end=" ")
8
9 nano11()

```

MiniC# nano11

```

1 using System;
2
3 namespace nano11
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n;
10            int m;
11            int x;
12            n = 1;
13            m = 2;
14            x = 5;
15            while(x > n)
16            {
17                n = n + m;
18                Console.WriteLine(n);
19            }
20        }
21    }

```



22  
23 }

### Assembly nano11

```
1 .assembly extern mscorlib{}
2
3 .assembly 'nano11'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano11.exe
8
9 .namespace nano11
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (
31                 int32 V_0,
32                 int32 V_1,
33                 int32 V_2)
34
35             ldc.i4.1
36             stloc.0
37             ldc.i4.2
38             stloc.1
39             ldc.i4.5
40             ldloc.2
41             bne.un IL_0015
42
43 IL_000b: ldloc.0
44           ldloc.1
45           add
46           stloc.0
```

```

47         ldloc.0
48         call void class [mscorlib]System.Console::
           WriteLine(int32)
49     IL_0015:     ldloc.2
50                 ldloc.0
51                 br IL_000b
52
53
54     IL_001c:     ret
55     }
56
57 }
58 }

```

## 6.12 nano12

MiniPython nano12

```

1 def nano12():
2     n=1
3     m=2
4     x=5
5     while x > n:
6         if n == m:
7             print(n,end=" ")
8         else:
9             print(0,end=" ")
10            x = x -1
11
12 nano12()

```

MiniC# nano12

```

1 using System;
2
3 namespace nano12
4 {
5     class Program
6     {
7         static void Main(String[] args)
8         {
9             int n = 1;
10            int m = 2;
11            int x = 5;
12            while(x > n)
13            {
14                if(n == m){
15                    Console.WriteLine(n);
16                }
17                else{

```

```

18         Console.WriteLine(0);
19     }
20     x = x - 1;
21 }
22 }
23 }
24
25 }

```

Assembly nano12

```

1 .assembly extern mscorlib{}
2
3 .assembly 'nano12'
4 {
5     .ver 0:0:0:0
6 }
7 .module nano12.exe
8
9 .namespace nano12
10 {
11     .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13     {
14
15         .method public hidebysig specialname rtspecialname
16             instance default void '.ctor' () cil managed
17         {
18             .maxstack 8
19             ldarg.0
20             call instance void object::.ctor'()
21             ret
22         }
23
24         // method line 2
25         .method private static hidebysig
26             default void Main () cil managed
27         {
28             .entrypoint
29             .maxstack 0
30             .locals init (
31                 int32 V_0,
32                 int32 V_1,
33                 int32 V_2)
34
35             ldc.i4.1
36             stloc.0
37             ldc.i4.2
38             stloc.1
39             ldc.i4.5
40             ldloc.2

```

```

41         br IL_0027
42
43     IL_000b:    ldloc.0
44               ldloc.1
45               bne.un IL_001d
46
47               ldloc.0
48               call void class [mscorlib]System.Console::
49                   WriteLine(int32)
50               br IL_0023
51
52     IL_001d:    ldc.i4.0
53               call void class [mscorlib]System.Console::
54                   WriteLine(int32)
55               ldloc.2
56     IL_0023:    ldc.i4.1
57               sub
58               stloc.2
59     IL_0027:    ldloc.2
60               ldloc.0
61               bgt IL_000b
62
63     IL_002e:    ret
64   }
65 }

```

## 6.13 micro01

MiniPython micro01

```

1 def micro01():
2     cel , far = 0.0 , 0.0
3     print("    Tabela de conversao: Celsius -> Fahrenheit")
4     print("Digite a temperatura em Celsius: ",end="")
5     cel = input()
6     far = (9*cel+160)/5
7     print("A nova temperatura    :"+str(far)+"F")
8
9 micro01()

```

MiniC# micro01

```

1 using System;
2
3 namespace micro01
4 {
5     class Program
6     {

```

```

7      static void Main(String[] args)
8      {
9          /*
10         Função: Ler uma temperatura em graus Celsius e
              apresentá-la
11         convertida em graus Fahrenheit. A fórmula de
              conversão é:
12         F=(9*C+160) / 5,
13         sendo F a temperatura em Fahrenheit e C a
              temperatura em
14         Celsius.
15         */
16
17         float cel;
18         float far;
19
20         Console.WriteLine("    Tabela de conversão:
              Celsius -> Fahrenheit");
21         Console.Write("Digite a temperatura em Celsius:
              ");
22         cel = Console.ReadLine();
23         far = (9*cel+160)/5;
24
25         Console.WriteLine("A nova temperatura : ",far,
              " F");
26     }
27 }
28
29 }

```

## 7 Analisador Léxico

Nessa seção será abordado um analisador léxico simples, o qual o professor Alexandro Santos Soares aplicou em sala como atividade. A abordagem será passo a passo mostrando a abordagem e construção pelo Autômato até a implementação do mesmo em Ocaml.

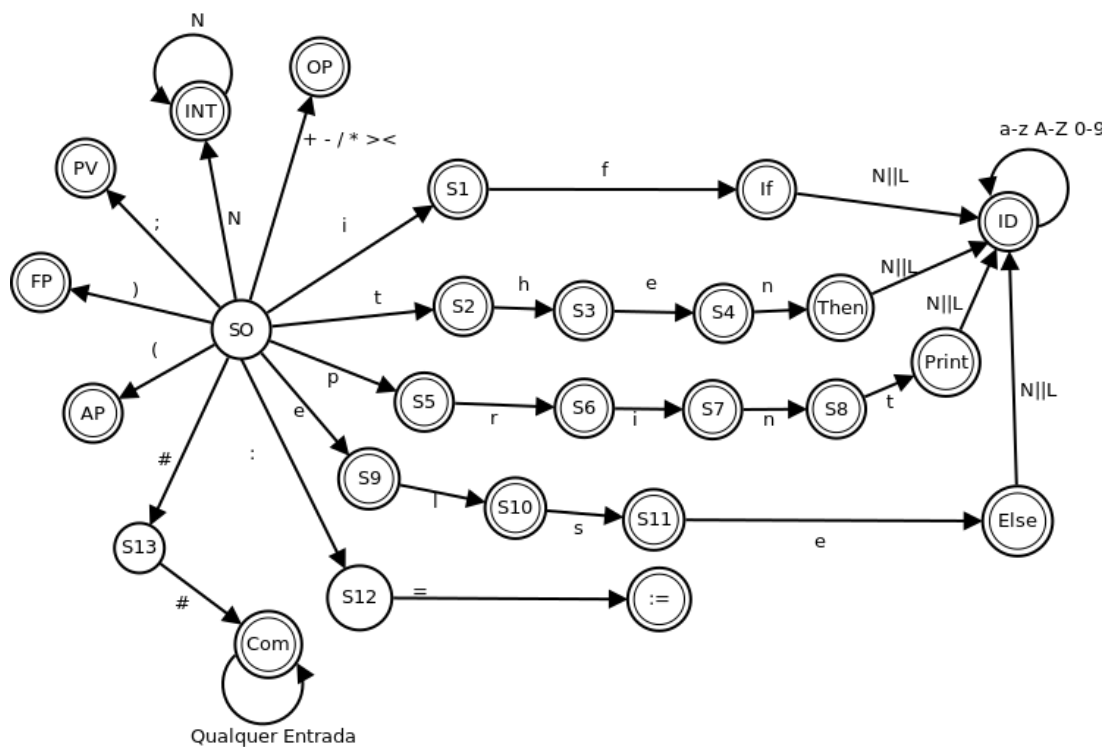
### 7.1 Abordagem por Autômato

Segue a versão do autômato com suas especificações, o qual será implementado em Ocaml. O autômato que abordaremos nesse ponto não reconhece a linguagem Python.

É reconhecida uma linguagem convencionada e simplificada na sala de aula. O código da linguagem foi fornecido pelo professor Alexandro, feito durante as aulas de Computadores;

Considerações realizadas:

- L: [a-z]U[A-Z];
- N: [0-9];
- INT: estado final para reconhecimento de números inteiros;
- PV: estado final para reconhecimento de ponto e vírgula;
- OP: estado final para reconhecimento de operadores;
- AP: estado final para reconhecimento de abrir parênteses;
- FP: estado final para reconhecimento de fechar parênteses;
- ID: estado final para categorizar identificadores;
- Com: estado final para categorizar comentários;
- if, then, else e print constituem palavras reservadas da linguagem;
- cada estado das palavras reservadas tem uma seta até id lendo L ou N .  
Isso viabiliza identificadores como el1 the1 then1 e else1.



## Automato - Ocaml

```
1 let lexico (str:entrada) =
2   let trans (e:estado) (c:simbolo) =
3   match (e,c) with
4     | (0, 'i') -> 1
5     | (0, 't') -> 6
6     | (0, 'e') -> 10
7     | (0, 'p') -> 14
8     | (0, 'f') -> 25
9     | (0, 'w') -> 28
10    | (0, '(') -> 19
11    | (0, ')') -> 20
12    | (0, ';') -> 22
13    | (0, ':') -> 23
14    | (0, '#') -> 33
15    | (0, _) when eh_operador c -> 21
16    | (0, _) when eh_letra c -> 3
17    | (0, _) when eh_digito c -> 4
18    | (0, _) when eh_branco c -> 5
19    | (0, _) -> failwith ("Erro lexico: caracter
        desconhecido " ^ Char.escaped c)
20
21    | (1, 'f') -> 2
22    | (1, _) when eh_letra c || eh_digito c -> 3
23
24    | (2, _) when eh_letra c || eh_digito c -> 3
25
26    | (3, _) when eh_letra c || eh_digito c -> 3
27
28    | (4, _) when eh_digito c -> 4
29
30    | (5, _) when eh_branco c -> 5
31
32    | (6, 'h') -> 7
33    | (6, _) when eh_letra c || eh_digito c -> 3
34
35    | (7, 'e') -> 8
36    | (7, _) when eh_letra c || eh_digito c -> 3
37
38    | (8, 'n') -> 9
39    | (8, _) when eh_letra c || eh_digito c -> 3
40
41    | (9, _) when eh_letra c || eh_digito c -> 3
42
43    | (10, 'l') -> 11
44    | (10, _) when eh_letra c || eh_digito c -> 3
45
46    | (11, 's') -> 12
47    | (11, _) when eh_letra c || eh_digito c -> 3
```

```
48 | (12, 'e') -> 13
49 | (12, _) when eh_letra c || eh_digito c -> 3
50
51 | (13, _) when eh_letra c || eh_digito c -> 3
52
53 | (14, 'r') -> 15
54 | (14, _) when eh_letra c || eh_digito c -> 3
55
56 | (15, 'i') -> 16
57 | (15, _) when eh_letra c || eh_digito c -> 3
58
59 | (16, 'n') -> 17
60 | (16, _) when eh_letra c || eh_digito c -> 3
61
62 | (17, 't') -> 18
63 | (17, _) when eh_letra c || eh_digito c -> 3
64
65 | (18, _) when eh_letra c || eh_digito c -> 3
66
67 | (23, '=') -> 24
68
69 | (25, 'o') -> 26
70 | (25, _) when eh_letra c || eh_digito c -> 3
71
72 | (26, 'r') -> 27
73 | (27, _) when eh_letra c || eh_digito c -> 3
74
75 | (27, _) when eh_letra c || eh_digito c -> 3
76
77 | (28, 'h') -> 29
78 | (28, _) when eh_letra c || eh_digito c -> 3
79
80 | (29, 'i') -> 30
81 | (29, _) when eh_letra c || eh_digito c -> 3
82
83 | (30, 'l') -> 31
84 | (30, _) when eh_letra c || eh_digito c -> 3
85
86 | (31, 'e') -> 32
87 | (31, _) when eh_letra c || eh_digito c -> 3
88
89 | (32, _) when eh_letra c || eh_digito c -> 3
90
91 | (33, '#') -> 34
92
93 | (34, _) when pulou_linha c -> 0
94 | (34, _) -> 34
95
96
97
```



```

98     | _ -> estado_morto
99 and rotulo e str =
100     match e with
101     | 2 -> If
102     | 1
103     | 6
104     | 7
105     | 8
106     | 10
107     | 11
108     | 12
109     | 14
110     | 15
111     | 16
112     | 17
113     | 25
114     | 26
115     | 28
116     | 29
117     | 30
118     | 31
119     | 3 -> Id str
120     | 4 -> Int str
121     | 5 -> Branco
122     | 9 -> Then
123     | 13 -> Else
124     | 18 -> Print
125     | 19 -> APar
126     | 20 -> FPar
127     | 21 -> OP str
128     | 22 -> PV
129     | 24 -> Atrib
130     | 27 -> For
131     | 32 -> While
132     | _ -> failwith ("Erro lexico: sequencia desconhecida " ^
        str)
133 in let dfa = { transicao = trans;
134               estado = estado_inicial;
135               posicao = 0 }
136 in let estado_lexico = {
137     pos_inicial = 0;
138     pos_final = -1;
139     ultimo_final = -1;
140     rotulo = rotulo;
141     dfa = dfa
142 } in
143     analisador str (String.length str) estado_lexico

```

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

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

Analizador Lexico

### 7.1.1 Implementação

Este analisador foi implementado utilizando a linguagem funcional Ocaml. Segue no código abaixo, todas as funções utilizadas para a implementação do mesmo para que possam ser reconhecidas as expressões encontradas na sessão 7.1

```

1 {
2
3   open Lexing
4   open Printf
5
6   type token =
7     | LITINT of (int)
8     | LITSTRING of (string)
9     | ID of (string)
10    | LITFLOAT of (float)
11    | APAR
12    | FPAR
13    | VIRG
14    | MAIS
15    | DPONTOS
16    | MENORIGUAL
17    | SETA
18    | E
19    | ATRIB
20    | RETURN
21    | DEF
22    | EOF
23    (*tokens adicionados*)
24    | OU
25    | IF
26    | ELSE
27    | WHILE
28    | FOR
29    | MAIORIGUAL
30    | IMPORT
31    | INT
32    | FLOAT
33    | LIST

```

```
34 | ABRECOLCHETES
35 | FECHACOLCHETES
36 | ABRECHAVES
37 | FECHACHAVES
38 | INCR
39 | DECR
40 | IGUALDADE
41 | MENOS
42 | VEZES
43 | DIVIDIDO
44 | MENOR
45 | MAIOR
46 | PV
47 | IN
48 | RANGE
49 | CHAR
50 | DOUBLE
51 | PONTO
52 | PASS
53 | VOID
54 | ELIF
55 | PRINT
56 | STR
57 | INPUT
58 | LENGTH
59 | DIFERENTE
60 | TRUE
61 | FALSE
62 | BREAK
63 | IS
64 | NOT
65 | MODULO
66 | FROM
67 | ATRIBMAIS
68 | ATRIBMENOS
69 | ATRIBMULT
70 | ATRIBDIV
71 (* Os tokens a seguir s o importantes para o pr
    processador *)
72 | Linha of (int * int * token list)
73 | INDENTA
74 | DEDENTA
75 | NOVALINHA
76
77 (*Adicionado*)
78 let booleano nbool =
79     match nbool with
80     | "True" -> 1
81     | "False" -> 0
82     | _ -> failwith "Erro: nao eh valor booleano"
```

```

83
84 let nivel_par = ref 0
85
86 let incr_num_linha lexbuf =
87   let pos = lexbuf.lex_curr_p in
88   lexbuf.lex_curr_p <- { pos with
89     pos_lnum = pos.pos_lnum + 1;
90     pos_bol = pos.pos_cnum;
91   }
92
93 let msg_erro lexbuf c =
94   let pos = lexbuf.lex_curr_p in
95   let lin = pos.pos_lnum
96   and col = pos.pos_cnum - pos.pos_bol - 1 in
97   sprintf "%d-%d: caracter desconhecido %c" lin col c
98
99   (*Adicionado*)
100  let erro lin col msg =
101    let mensagem = sprintf "%d-%d: %s" lin col msg in
102    failwith mensagem
103
104
105 }
106
107 let digito = ['0' - '9']
108 let int = digito+
109 let comentario = "#"[ ^ '\n' ]*
110 let linha_em_branco = [ ' '\t' ]* comentario
111 let restante = [ ^ ' '\t' '\n' ] [ ^ '\n' ]+
112 let brancos = [ ' '\t' ]+
113 let novalinha = '\r' | '\n' | "\r\n"
114 let letra = [ 'a'-'z' 'A' - 'Z' ]
115 let identificador = letra ( letra | digito | '_' ) *
116
117 (* 0 pr processador necess rio para contabilizar a
   indenta o *)
118 rule preprocessor indentacao = parse
119   linha_em_branco          { preprocessor 0 lexbuf } (*
   ignora brancos *)
120 | [ ' '\t' ]+ '\n'        { incr_num_linha lexbuf;
121                             preprocessor 0 lexbuf } (*
   ignora brancos *)
122 | ' '                    { preprocessor (indentacao + 1)
123                             lexbuf }
124 | '\t'                    { let nova_ind = indentacao + 8 -
125                             (indentacao mod 8)
126                             in preprocessor nova_ind
127                             lexbuf }
128 | novalinha                { incr_num_linha lexbuf;
129                             preprocessor 0 lexbuf }

```

```

127 | restante as linha {
128     let rec tokenize lexbuf =
129         let tok = token lexbuf in
130         match tok with
131         | EOF -> []
132         | _ -> tok :: tokenize lexbuf in
133     let toks = tokenize (Lexing.from_string linha) in
134     (* A impress o a seguir serve apenas para depura o
135        . Retirar depois! *)
135     printf "Linha(indentacao=%d,nivel_par=%d)\n" indentacao
136         (!nivel_par);
136     Linha(indentacao,!nivel_par, toks)
137 }
138 | eof { nivel_par := 0; EOF }
139
140
141
142
143 (* O analisador l xico a ser chamado ap s o pr
144    processador *)
144 and token = parse
145     brancos           { token lexbuf }
146 | comentario         { token lexbuf }
147 | " , , , "          { comentario_bloco 0 lexbuf; }
148 | ">="                { MAIORIGUAL }
149 | "<="                { MENORIGUAL }
150 | "->"               { SETA }
151 | "++"               { INCR }
152 | "--"               { DECR }
153 | "=="               { IGUALDADE }
154 | "!="               { DIFERENTE }
155 | "+="               { ATRIBMAIS }
156 | "-="               { ATRIBMENOS }
157 | "*="               { ATRIBMULT }
158 | "/="               { ATRIBDIV }
159 | '{'                { ABRECHAVES }
160 | '}'                { FECHACHAVES }
161 | '['                { ABRECOLCHETES }
162 | ']'                { FECHACOLCHETES }
163 | '('                { let _ = incr(nivel_par) in APAR }
164 | ')'                { let _ = decr(nivel_par) in FPAR }
165 | ','                { VIRG }
166 | '+'                { MAIS }
167 | '-'                { MENOS }
168 | '*'                { VEZES }
169 | '/'                { DIVIDIDO }
170 | '='                { ATRIB }
171 | ':'                { DPONTOS }
172 | ';'                { PV }
173 | '<'                { MENOR }

```

```

174 | '>' { MAIOR }
175 | '%' { MODULO }
176 | '.' { PONTO }
177 | int as num { let numero = int_of_string num in
178 |             LITINT numero }
179 | "or" { OU }
180 | "if" { IF }
181 | "else" { ELSE }
182 | "while" { WHILE }
183 | "for" { FOR }
184 | "return" { RETURN }
185 | "def" { DEF }
186 | "import" { IMPORT }
187 | "int" { INT }
188 | "float" { FLOAT }
189 | "double" { DOUBLE }
190 | "char" { CHAR }
191 | "list" { LIST }
192 | "and" { E }
193 | "in" { IN }
194 | "range" { RANGE }
195 | "pass" { PASS }
196 | "void" { VOID }
197 | "elif" { ELIF }
198 | "print" { PRINT }
199 | "str" { STR }
200 | "input" { INPUT }
201 | "len" { LENGTH }
202 | "break" { BREAK }
203 | "is" { IS }
204 | "not" { NOT }
205 | "from" { FROM }
206 | identificador as id { ID id }
207 | '""' { let pos = lexbuf.lex_curr_p in
208 |         let lin = pos.pos_lnum
209 |         and col = pos.pos_cnum - pos.pos_bol
210 |         - 1 in
211 |         let buffer = Buffer.create 1 in
212 |         let str = leia_string lin col buffer
213 |         lexbuf in
214 |         LITSTRING str }
215 | _ as c { failwith (msg_erro lexbuf c) }
216 | eof { EOF }
217
218 and comentario_bloco n = parse
219     "'''" { if n=0 then token lexbuf
220           else comentario_bloco (n-1) lexbuf }
221 | "'''" { comentario_bloco (n+1) lexbuf }

```

```

222 | novalinha { incr_num_linha lexbuf; comentario_bloco n
      lexbuf }
223 | _        { comentario_bloco n lexbuf }
224 | eof      { failwith "Coment rio n o fechado" }
225
226
227
228 and leia_string lin col buffer = parse
229   | '"'      { Buffer.contents buffer}
230   | "\\t"    { Buffer.add_char buffer '\t'; leia_string
      lin col buffer lexbuf }
231   | "\\n"    { Buffer.add_char buffer '\n'; leia_string
      lin col buffer lexbuf }
232   | '\\ ' '"' { Buffer.add_char buffer '"'; leia_string
      lin col buffer lexbuf }
233   | '\\ ' '\\ ' { Buffer.add_char buffer '\\'; leia_string
      lin col buffer lexbuf }
234   | _ as c    { Buffer.add_char buffer c; leia_string lin
      col buffer lexbuf }
235   | eof      { erro lin col "A string n o foi fechada"}

```

### 7.1.2 Execução

Para que possa executar o analisador léxico, deve estar instalado o Ocaml. Seu tutorial de instalação encontra-se em 4.2

1. Deve-se no terminal, compilar o arquivo lexico.mll para gerar o arquivo lexico.ml, utilizando o comando::

```
ocamllex lexico.mll
```

2. Agora, o compilador do OCaml deve compilar o arquivo lexico.ml, gerando o arquivo "carregador.ml", utilizando o comando:

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

3. Já compilado, deve-se abrir o OCaml utilizando o seguinte comando:

```
ocaml
```

4. Para que o OCaml possa carregar o analisador léxico, utilizando o seguinte co- mando:

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

5. Neste momento já é possível testá-lo. Pode-se utilizar da seguinte forma, para analisar o código escrito em um arquivo chamado "codigo":

```
lex "codigo";
```

Supondo o seguinte arquivo de código abaixo:

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

A saída do analisador léxico é a seguinte:

```
1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=8,nivel_par=0)
8 Nivel: 8
9 Linha(identacao=8,nivel_par=0)
10 Nivel: 8
11 EOF
12 - : tokens =
13     [Lexico.DEF; Lexico.ID "paco"; Lexico.APAR;
14     Lexico.ID "x"; Lexico.FPAR;Lexico.DPONTOS;
15     Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "x";
16     Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
17     Lexico.LITINT 1; Lexico.NOVALINHA;Lexico.ID "y";
18     Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
19     Lexico.LITINT 2;Lexico.NOVALINHA; Lexico.ID "y";
20     Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
21     Lexico.LITINT 3; Lexico.MAIS; Lexico.APAR;
22     Lexico.LITINT 4; Lexico.MAIS;Lexico.LITINT 5;
23     Lexico.FPAR; Lexico.NOVALINHA; Lexico.RETURN;
24     Lexico.ID "x"; Lexico.NOVALINHA; Lexico.DEDENTA;
25     Lexico.EOF]
```

## 7.2 Abordagem por Linguagem Regular

Foi feita uma implementação de linguagem regular.Dessa forma facilita o processo de análise léxica.



### 7.2.1 Implementação

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

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

### 7.2.2 Execução

Código

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

Saída

```
1 # lex "nano02.py";
2 Linha(identacao=0,nivel_par=0)
3 Nivel: 0
4 Linha(identacao=8,nivel_par=-1)
5 Nivel: 8
6 EOF
7 - : tokens =
8 [Lexico.DEF; Lexico.ID "nano02"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
9  Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
   ATTRIB; Lexico.INT;
10 Lexico.ID "n"; Lexico.FPAR; Lexico.DEDENTA; Lexico.EOF]
```

Código

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

Saída

```
1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 EOF
8 - : tokens =
9 [Lexico.DEF; Lexico.ID "nano03"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
```

```

10 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB; Lexico.INT;
11 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;
    Lexico.ID "n";
12 Lexico.ATRIB; Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.
    DEDENTA; Lexico.EOF]

```

Código

```

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

```

Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 EOF
8 - : tokens =
9 [Lexico.DEF; Lexico.ID "nano03"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
10 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB; Lexico.INT;
11 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;
    Lexico.ID "n";
12 Lexico.ATRIB; Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.
    DEDENTA; Lexico.EOF]

```

Código

```

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

```

Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 EOF
8 - : tokens =
9 [Lexico.DEF; Lexico.ID "nano04"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
10 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB; Lexico.INT;
11 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;
    Lexico.ID "n";

```

```

12 Lexico.ATRIB; Lexico.LITINT 1; Lexico.MAIS; Lexico.LITINT
    2;
13 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.EOF]

```

Código

```

1 def nano05():
2     n = 2
3     print(n,end=" ")
4
5 nano05()

```

Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=0,nivel_par=0)
8 Nivel: 0
9 EOF
10 - : tokens =
11 [Lexico.DEF; Lexico.ID "nano05"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
12 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
13 Lexico.LITINT 2; Lexico.NOVALINHA; Lexico.PRINT; Lexico.
    APAR; Lexico.ID "n";
14 Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB; Lexico.
    LITSTRING "";
15 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "
    nano05 ";
16 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]

```

Código

```

1 def nano06():
2     n = 1 - 2
3     print(n,end=" ")
4
5 nano06()

```

Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=0,nivel_par=0)

```

```

8 Nivel: 0
9 EOF
10 - : tokens =
11 [Lexico.DEF; Lexico.ID "nano06"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
12 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
13 Lexico.LITINT 1; Lexico.MENOS; Lexico.LITINT 2; Lexico.
    NOVALINHA;
14 Lexico.PRINT; Lexico.APAR; Lexico.ID "n"; Lexico.VIRG;
    Lexico.ID "end";
15 Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.FPAR; Lexico.
    NOVALINHA;
16 Lexico.DEDENTA; Lexico.ID "nano06"; Lexico.APAR; Lexico.
    FPAR;
17 Lexico.NOVALINHA; Lexico.EOF]

```

#### Código

```

1 def nano7():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5
6 nano7()

```

#### Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=16,nivel_par=0)
8 Nivel: 16
9 Linha(identacao=0,nivel_par=0)
10 Nivel: 0
11 EOF
12 - : tokens =
13 [Lexico.DEF; Lexico.ID "nano07"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
14 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
15 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
16 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
    NOVALINHA;
17 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
18 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;

```

```

19 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.DEDENTA; Lexico.ID
    "nano07";
20 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]

```

#### Código

```

1 def nano08():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5     else:
6         print(0,end=" ")
7
8 nano08()

```

#### Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=16,nivel_par=0)
8 Nivel: 16
9 Linha(identacao=8,nivel_par=0)
10 Nivel: 8
11 Linha(identacao=16,nivel_par=0)
12 Nivel: 16
13 Linha(identacao=0,nivel_par=0)
14 Nivel: 0
15 EOF
16 - : tokens =
17 [Lexico.DEF; Lexico.ID "nano08"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
18 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
19 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
20 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
    NOVALINHA;
21 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
22 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;
23 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.
    DPONTOS;
24 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR
    ;
25 Lexico.LITINT 0; Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB
    ;

```

```

26 Lexico.LITSTRING ""; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
    DEDENTA;
27 Lexico.DEDENTA; Lexico.ID "nano08"; Lexico.APAR; Lexico.
    FPAR;
28 Lexico.NOVALINHA; Lexico.EOF]

```

### Código

```

1 def nano09():
2     n=1
3     if n ==1:
4         print(n,end=" ")
5     else:
6         print(0,end=" ")
7
8 nano09()

```

### Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=16,nivel_par=0)
8 Nivel: 16
9 Linha(identacao=8,nivel_par=0)
10 Nivel: 8
11 Linha(identacao=16,nivel_par=0)
12 Nivel: 16
13 Linha(identacao=0,nivel_par=0)
14 Nivel: 0
15 EOF
16 - : tokens =
17 [Lexico.DEF; Lexico.ID "nano09"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
18 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
19 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
20 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
    NOVALINHA;
21 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
22 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;
23 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.
    DPONTOS;
24 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR
    ;

```

```

25 Lexico.LITINT 0; Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB
    ;
26 Lexico.LITSTRING ""; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
    DEDENTA;
27 Lexico.DEDENTA; Lexico.ID "nano09"; Lexico.APAR; Lexico.
    FPAR;
28 Lexico.NOVALINHA; Lexico.EOF]

```

#### Código

```

1 def nano10():
2     n=1
3     m=2
4     if n ==m:
5         print(n,end=" ")
6     else:
7         print(0,end=" ")
8
9
10 nano10()

```

#### Saída

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=8,nivel_par=0)
8 Nivel: 8
9 Linha(identacao=16,nivel_par=0)
10 Nivel: 16
11 Linha(identacao=8,nivel_par=0)
12 Nivel: 8
13 Linha(identacao=16,nivel_par=0)
14 Nivel: 16
15 Linha(identacao=0,nivel_par=0)
16 Nivel: 0
17 EOF
18 - : tokens =
19 [Lexico.DEF; Lexico.ID "nano10"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
20 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
21 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.ID "m"; Lexico.
    ATRIB;
22 Lexico.LITINT 2; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
23 Lexico.IGUALDADE; Lexico.ID "m"; Lexico.DPONTOS; Lexico.
    NOVALINHA;

```

```

24 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
25 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;
26 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.
    DPONTOS;
27 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR
    ;
28 Lexico.LITINT 0; Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB
    ;
29 Lexico.LITSTRING ""; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
    DEDENTA;
30 Lexico.DEDENTA; Lexico.ID "nano10"; Lexico.APAR; Lexico.
    FPAR;
31 Lexico.NOVALINHA; Lexico.EOF]

```

### 7.2.3 Erros Léxicos

Código

```

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

```

Saída

Erro: String Aberta

```

1 Linha(identacao=0,nivel_par=0)
2 Nivel: 0
3 Linha(identacao=8,nivel_par=0)
4 Nivel: 8
5 Linha(identacao=8,nivel_par=0)
6 Nivel: 8
7 Linha(identacao=8,nivel_par=0)
8 Nivel: 8
9 Linha(identacao=8,nivel_par=0)
10 Nivel: 8
11 Linha(identacao=8,nivel_par=0)
12 Nivel: 8
13
14 Exception: Failure "1-40: A string nao foi fechada".

```

Código

```

1 def micro01():

```



```

2      @cel , far = 0.0 , 0.0
3      print("      Tabela de conversao: Celsius -> Fahrenheit")
4      print("Digite a temperatura em Celsius: ",end="")
5      cel = input()
6      far = (9*cel+160)/5
7      print("A nova temperatura eh:"+str(far)+"F")
8      ''' cine'''
9  micro01()

```

Saída

Erro: Variável com caractere proibido

```

1  Linha(identacao=0,nivel_par=0)
2  Nivel: 0
3
4  Exception: Failure "1-0: caracter desconhecido @".

```

Código

```

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

```

Saída

Erro: Comentario não fechado

```

1  Linha(identacao=0,nivel_par=0)
2  Nivel: 0
3  Linha(identacao=8,nivel_par=0)
4  Nivel: 8
5  Linha(identacao=8,nivel_par=0)
6  Nivel: 8
7  Linha(identacao=8,nivel_par=0)
8  Nivel: 8
9  Linha(identacao=8,nivel_par=0)
10 Nivel: 8
11 Linha(identacao=8,nivel_par=0)
12 Nivel: 8
13 Linha(identacao=8,nivel_par=0)
14 Nivel: 8
15
16 Exception: Failure "Comentario nao fechado".

```

## 8 Analisador Sintático

Essa seção contém a implementação de um analisador sintático simples abordado em sala de aula e ourto para mini-Python.

### 8.1 Analisador Abordado em Aula

#### 8.1.1 Gramática

Como passo inicial foi passado em sala de aula, pelo professor Alexssandro Santos Soares, a gramática a seguir com o intuito de aprofundarmos os conhecimentos do processo da análise.

```
S -> XYZ
X -> aXb
X ->
Y -> cYZcX
Y -> d
Z -> eZYe
Z -> f
```

Para que seja melhor o entendimento da proxima tabela precisamos entender os conceitos de first e follow. First: conjunto de simbolos que ocorrem no inicio de uma determinada regra.

Follow: o que pode aparecer depois da ocorrência de um determinado simbolo.

#### 8.1.2 Código

Segue os códigos do analisador sintático, juntamente com os arquivos lexico.mll (o qual usaremos para efetuar analise do capitulo anterior) e sintatico.mli(tokens para efetuar a analise sintativa) e o sintativoArv.ml(contrução da arvore sintática para efetuarmos a analise).

léxico.mll

```
1 {
2   open Lexing
3   open Printf
4   open Sintatico
5
6   let incr_num_linha lexbuf =
7     let pos = lexbuf.lex_curr_p in
8     lexbuf.lex_curr_p <- { pos with
9       pos_lnum = pos.pos_lnum + 1;
10      pos_bol = pos.pos_cnum;
11    }
```

```

12
13   let msg_erro lexbuf c =
14     let pos = lexbuf.lex_curr_p in
15     let lin = pos.pos_lnum
16     and col = pos.pos_cnum - pos.pos_bol - 1 in
17     sprintf "%d-%d: caracter desconhecido %c" lin col c
18
19   }
20
21 rule token = parse
22 | 'a'      {A}
23 | 'b'      {B}
24 | 'c'      {C}
25 | 'd'      {D}
26 | 'e'      {E}
27 | 'f'      {F}
28 | _ as c   { failwith (msg_erro lexbuf c) }
29 | eof      { EOF }

```

sintaticoArv.ml

```

1 (* Parser preditivo *)
2 #load "lexico.cmo";;
3 open Sintatico;;
4
5 type rule = S of rule * rule * rule
6           | X of tokens * rule * tokens
7           | Y of tokens * rule * rule * tokens * rule
8           | Z of tokens * rule * rule * tokens
9           | X_vazio
10          | Y_d of tokens
11          | Z_f of tokens
12
13 let tk = ref EOF (* variavel global para o token atual *)
14 let lexbuf = ref (Lexing.from_string "")
15
16 (* le o proximo token *)
17 let prox () = tk := Lexico.token !lexbuf
18
19 let to_str tk =
20   match tk with
21   | A -> "a"
22   | B -> "b"
23   | C -> "c"
24   | D -> "d"
25   | E -> "e"
26   | F -> "f"
27   | EOF -> "eof"
28
29 let erro esp =
30   let msg = Printf.sprintf "Erro: esperava %s mas encontrei

```

```

31      %s"
32      esp (to_str !tk)
33      in
34      failwith msg
35  let consome t = if (!tk == t) then prox() else erro (to_str
36                  t)
37  let rec ntS () =
38      match !tk with
39      | A
40      | C
41      | D      ->
42          let cmd1 = ntX() in
43          let cmd2 = ntY() in
44          let cmd3 = ntZ() in
45          S (cmd1, cmd2, cmd3)
46      | _ -> erro "a, c ou d"
47  and ntX () =
48      match !tk with
49      | B
50      | C
51      | D
52      | E
53      | F      -> X_vazio
54      | A      -> let _ = consome A in
55                  let cmd = ntX() in
56                  let _ = consome B in
57                  X (A, cmd, B)
58      | _ -> erro "a"
59  and ntY () =
60      match !tk with
61      | C      -> let _ = consome C in
62                  let cmd = ntY() in
63                  let cmd2 = ntZ() in
64                  let _ = consome C in
65                  let cmd3 = ntX() in
66                  Y (C,cmd,cmd2, C, cmd3)
67      | D      -> let _ = consome D in
68                  Y_d (D)
69      | _      -> erro "c ou d"
70  and ntZ () =
71      match !tk with
72      | E      -> let _ = consome E in
73                  let cmd = ntZ() in
74                  let cmd2 = ntY() in
75                  let _ = consome E in
76                  Z (E, cmd, cmd2, E)
77      | F      -> let _ = consome F in
78                  Z_f (F)

```

```

79 | _      -> erro "e ou f"
80
81 let parser str =
82   lexbuf := Lexing.from_string str;
83   prox (); (* inicializa o token *)
84   let arv = ntS () in
85   match !tk with
86     EOF -> let _ = Printf.printf "Ok!\n" in arv
87   | _ -> erro "fim da entrada"
88
89 let teste str =
90   let entrada = str
91   in
92   parser entrada

```

### 8.1.3 Execução

Na execução utilizaremos novamente a ferramenta Ocaml.

1. Compilar o arquivo lexico.mll gerando o arquivo lexico.ml

```
ocamllex lexico.mll
```

2. Compilar o arquivo sintatico.mli gerando o arquivo sintatico.cmi

```
ocamlc -c sintatico.mli
```

3. compilar o arquivo lexico.ml gerando o arquivo carregador.ml

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

4. Já com os arquivos compilados abra o Ocaml

```
ocaml
```

5. Com o Ocaml executando, digite o código abaixo para que ele possa utilizar o analisador Léxico

```
load "lexico.cmo";;
```

6. Para que o Ocaml possa utilizar o analisador Sintativo, digite:

```
use "sintaticoArv.ml";;
```

7. Agora podemos executar o comando "teste()" para testar

```
teste();;
```

Para testarmos precisamos passar uma entrada como parametro e assim sera analisada a mesma

Entrada Válida

```
"cdfcf"
```

Saida do analisador para entrada válida:

```
- : variavel = S (X_vazio, Y(C, Y_dD, Z_fF, C, X_vazio), Z_fF)
```

Entrada Inválida

```
"cdfcfaaa"
```

Saida para a entrada inválida

```
Exception: Failure "Erro: esperava fim da entrada mas encontrei a".
```

## 8.2 Analisador sintático Mini-Python usando Menhir

Podemos perceber que com a complexidade do nosso projeto aumentando fica inviavel para que uma pessoa compile todos os arquivos na mão, para isso vamos utilizar a ferramenta Menhir para otimizar isso.

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

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

Para otimizarmos nossas compilações vamos deixar construir um arquivo "arquivo.ocamlinit", esse tera a função de listar quais arquivos devem ser compilados pelo Menhir

```
directory " build";; load "lexico.cmo";; load "parser.cmo";; "pre processador.cmo";; "main.cr
```

### 8.2.1 Execução de Testes

sintaticoArv.ml

```
1 def micro11() -> int :
2     numero = 0
```

```

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

```

Saída

```

1  - : Ast.prog =
2  Prog ([,
3      [DEFUNCAO([], Some INTEIRO,
4          [ATRIBUI 0 ("numero", ExpTerm (TERMTL (LITINT 0)));
5          ATRIBUI 0 ("X", ExpTerm (TERMTL (LITINT 0))); PRINT
6              ("numero", []);
7          LEIA "numero";
8          CHAMADADEFUNCAO ("numero", Some "verifica", [TERMID "
9              numero"]);
10         CONDICAIOIF
11             (EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (
12                 TERMTL (LITINT 1))),
13                 [], None);
14         CONDICAIOIF
15             (EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (
16                 TERMTL (LITINT 0))),
17                 [], Some ELSECOND);
18         RETORNO (TERMTL (LITINT 0))]);
19  DEFUNCAO ([INTEIRO], Some INTEIRO,
20      [ATRIBUICAO ("res", ExpTerm (TERMTL (LITINIT 0)));
21      CONDICAIOIF
22          (EXP (ExpTerm (TERMID "n"), MAIOR, ExpTerm (TERMTL (

```

```

    LITINT 0))),
19     [], None);
20     CONDICAIF
21     (EXP (ExpTerm (TERMID "n"), MENOR, ExpTerm (TERMTL (
        LITINT 0))),
22     [], Some ELSECOND);
23     RETORNO (TERMID "res"); CHAMADADEFUNCAO ("micro11", None,
        [])))]))

```

sintatico.mli

```

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

```

Saida

```

1     - : Ast.prog =
2     Prog ([ComecoDeBLoco],
3     [DEFFUNCAO ([], None,
4     [ATRIBUICAO ("numero", ExpTerm (TERMTL (LITINT 0))); LEIA
        "numero";
5     CONDICAIF
6     (EXP (ExpTerm (TERMID "numero"), MAIORIGUAL,
7     ExpTerm (TERMTL (LITINT 100))),
8     [], Some ELSECOND);
9     CHAMADADEFUNCAO ("micro",None, [])))]))

```