Universidade Federal de Uberlândia
Ciência da Computaçõa
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

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.

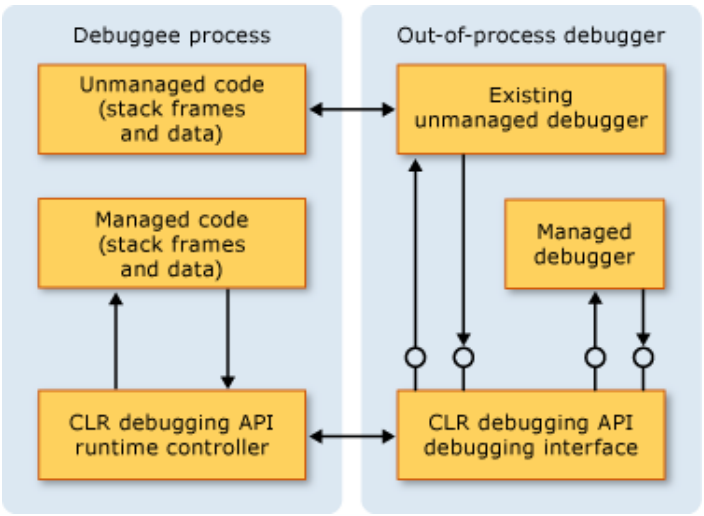
API de depuração CLR inclui dois componentes principais:

- 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 intruções 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 over- flow(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 se- gundo 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: inciliaza um bloco de memória para um valores
- jmp: pula para um método
- jmpi: 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.
- ldloca 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 códdigo
- localloc: aloca um espaço na memory pool do tamanho do primeiro elemento dap ilha 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(inverção 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 eslocamentoretiraestesdoisdapilhaecolocaoresultado.
- shr: mesmo que o shl porém com deslocamneto para a direita.
- shr.un: memso 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 poe na variavel(stloc x)
- sub: subtrai do segundo valor o primeiro e poe 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): expecifica 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 in- dex(primeira posição, topo da pilha) na pilha
- ldelema: poe o endereço do vetor na posição index na pilha.
- ldlen: poe na pilha o tamanho do vetor(topo da pilha)
- ldstr: poe 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 codigo:

sudo apt-get update

Apos autualizado, 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 codigo 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 cógido Python

python nanoC.py

Compilar o cógido C # e gerar o .exe

dmcs nanoC.cs

Gerar assembly em cima do oexecutavel C # .exe

monodis nanoC.exe > nanoC.txt

6 Códigos: MiniPython - C# - Assembly

6.1 nano01

MiniPython nano01

```
def nano01():
pass
```

MiniC# nano01

```
using System;

namespace nano01
{
    class Program
    {
        static void Main(String[] args){}
    }
}
```

Assembly nano01

```
.assembly extern mscorlib{}

.assembly 'nano01'
{
  .ver  0:0:0:0
}
.module nano01.exe

.namespace nano01
{
  .class private auto ansi beforefieldinit Program
```

```
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        ret
    }

  }
}
```

6.2 nano02

MiniPython nano02

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

MiniC# nano02

```
using System;

namespace nano02
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
        }
    }
}
```

Assembly nano02

```
.assembly extern mscorlib{}

.assembly 'nano02'
```

```
{
  .ver  0:0:0:0
}
.module nano02.exe

.namespace nano02
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ret
    }

  }
}
```

6.3 nano03

MiniPython nano03

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

MiniC# nano03

```
using System;

namespace nano03
{
    class Program
    {
        static void Main(String[] args)
```

```
        {
            int n;
            n = 1;
        }
    }

}
```

Assembly nano03

```
.assembly extern mscorlib{}

.assembly 'nano03'
{
  .ver  0:0:0:0
}
.module nano03.exe

.namespace nano03
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.1
        stloc.0
        ret
    }

  }
}
```

6.4 nano04

MiniPython nano04

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

MiniC# nano04

```
using System;

namespace nano04
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1 + 2;
        }
    }

}
```

Assembly nano04

```
.assembly extern mscorlib{}

.assembly 'nano04'
{
  .ver  0:0:0:0
}
.module nano04.exe

.namespace nano04
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
```

```
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.3
        stloc.0
        ret
    }

  }
}
```

6.5 nano05

MiniPython nano05

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

nano05()
```

MiniC# nano05

```
using System;

namespace nano05
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 2;
            Console.WriteLine(n);
        }
    }

}
```

Assembly nano05

```
.assembly extern mscorlib{}

.assembly 'nano05'
{
  .ver  0:0:0:0
}
.module nano05.exe

```

```
.namespace nano05
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
         .maxstack 8
         ldarg.0
         call instance void object::'.ctor'()
         ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
         .entrypoint
         .maxstack 0
         .locals init (int32 V_0)
         ldc.i4.2
         stloc.0
         ldloc.0
         call void class [mscorlib]System.Console::WriteLine(
            int32)
         ret
    }

  }
}
```

6.6 nano06

MiniPython nano06

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

nano06()
```

MiniC# nano06

```
using System;

namespace nano06
{
    class Program
```

```
    {
        static void Main(String[] args)
        {
            int n;
            n = 1 - 2;
            Console.WriteLine(n);
        }
    }

}
```

Assembly nano06

```
.assembly extern mscorlib{}

.assembly 'nano06'
{
  .ver  0:0:0:0
}
.module nano06.exe

.namespace nano06
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
            instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.m1
        stloc.0
        ldloc.0
        call void class [mscorlib]System.Console::WriteLine(
            int32)
        ret
    }

```

```
  }
}
```

6.7 nano07

MiniPython nano07

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

nano07()
```

MiniC# nano07

```
using System;

namespace nano07
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1;
            if(n == 1)
            {
                Console.WriteLine(n);
            }
        }
    }

}
```

Assembly nano07

```
.assembly extern mscorlib{}

.assembly 'nano07'
{
  .ver  0:0:0:0
}
.module nano07.exe

.namespace nano07
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

```

```
    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (int32 V_0)

           ldc.i4.1
           stloc.0
           ldloc.0
           ldc.i4.1
           bne.un IL_000f

           ldloc.0
           call void class [mscorlib]System.Console::
               WriteLine(int32)
  IL_000f:  ret
    }

  }
}
```

6.8 nano08

MiniPython nano08

```
def nano08():
    n=1
    if n ==1:
        print(n,end="")
    else:
        print(0,end="")

nano08()
```

MiniC# nano08

```
using System;

namespace nano08
```

```
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1;
            if(n == 1)
            {
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
            }
        }
    }

}
```

Assembly nano08

```
.assembly extern mscorlib{}

.assembly 'nano08'
{
  .ver  0:0:0:0
}
.module nano08.exe

.namespace nano08
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (int32 V_0)
```

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

```
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
            }
        }
    }

}
```

Assembly nano09

```
.assembly extern mscorlib{}

.assembly 'nano09'
{
  .ver  0:0:0:0
}
.module nano09.exe

.namespace nano09
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
            .entrypoint
            .maxstack 0
            .locals init (int32 V_0)

            ldc.i4.1
            stloc.0
            ldloc.0
            ldc.i4.1
            bne.un IL_0018

            ldloc.0
            ldc.i4.1
```

```
            add
            stloc.0
            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            br IL_001e

IL_0018:    ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
IL_001e:    ret
    }

  }
}
```

6.10 nano10

MiniPython nano10

```
def nano10():
    n=1
    m=2
    if n ==m:
        print(n,end="")
    else:
        print(0,end="")

nano10()
```

MiniC# nano10

```
using System;

namespace nano10
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            int m;
            n = 1;
            m = 2;
            if(n == m)
            {
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
```

```
            }
        }
    }

}
```

Assembly nano10

```
.assembly extern mscorlib{}

.assembly 'nano10'
{
  .ver  0:0:0:0
}
.module nano10.exe

.namespace nano10
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (
               int32   V_0,
               int32   V_1)

           ldc.i4.1
           stloc.0
           ldc.i4.2
           stloc.1
           ldloc.0
           ldloc.1
           bne.un IL_0016

           ldloc.0
           call void class [mscorlib]System.Console::
```

```
                WriteLine(int32)
            br IL_001c


IL_0016     ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
IL_001c:    ret
    }

  }
}
```

6.11 nano11

MiniPython nano11

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

nano11()
```

MiniC# nano11

```
using System;

namespace nano11
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            int m;
            int x;
            n = 1;
            m = 2;
            x = 5;
            while(x > n)
            {
                n = n + m;
                Console.WriteLine(n);
            }
        }
    }
```

```
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
```

```
            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
 IL_0015:   ldloc.2
            ldloc.0
            br IL_000b


  IL_001c:  ret
    }

  }
}
```

6.12 nano12

MiniPython nano12

```
def nano12():
    n=1
    m=2
    x=5
    while x >n:
        if n ==m:
            print(n,end="")
        else:
            print(0,end="")
        x = x -1

nano12()
```

MiniC# nano12

```
using System;

namespace nano12
{
    class Program
    {
        static void Main(String[] args)
        {
            int n = 1;
            int m = 2;
            int x = 5;
            while(x > n)
            {
                if(n == m){
                    Console.WriteLine(n);
                }
                else{
```

```
                    Console.WriteLine(0);
                }
                x = x - 1;
            }
        }
    }

}
```

Assembly nano12

```
.assembly extern mscorlib{}

.assembly 'nano12'
{
  .ver  0:0:0:0
}
.module nano12.exe

.namespace nano12
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
            .entrypoint
            .maxstack 0
            .locals init (
                int32   V_0,
                int32   V_1,
                int32   V_2)

            ldc.i4.1
            stloc.0
            ldc.i4.2
            stloc.1
            ldc.i4.5
            ldloc.2
```

```
            br IL_0027

 IL_000b:   ldloc.0
            ldloc.1
            bne.un IL_001d

            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            br IL_0023

 IL_001d:   ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            ldloc.2
 IL_0023    ldc.i4.1
            sub
            stloc.2
 IL_0027:   ldloc.2
            ldloc.0
            bgt IL_000b

 IL_002e:   ret
    }

  }
}
```

6.13 micro01

MiniPython micro01

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

micro01()
```

MiniC# micro01

```
using System;

namespace micro01
{
    class Program
    {
```

```
7           static void Main(String[] args)
8           {
9               /*
10              Fun   o: Ler uma temperatura em graus Celsius e
                    apresent -la
11              convertida em graus Fahrenheit. A f rmula de
                    convers o   :
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 converes o:
                    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

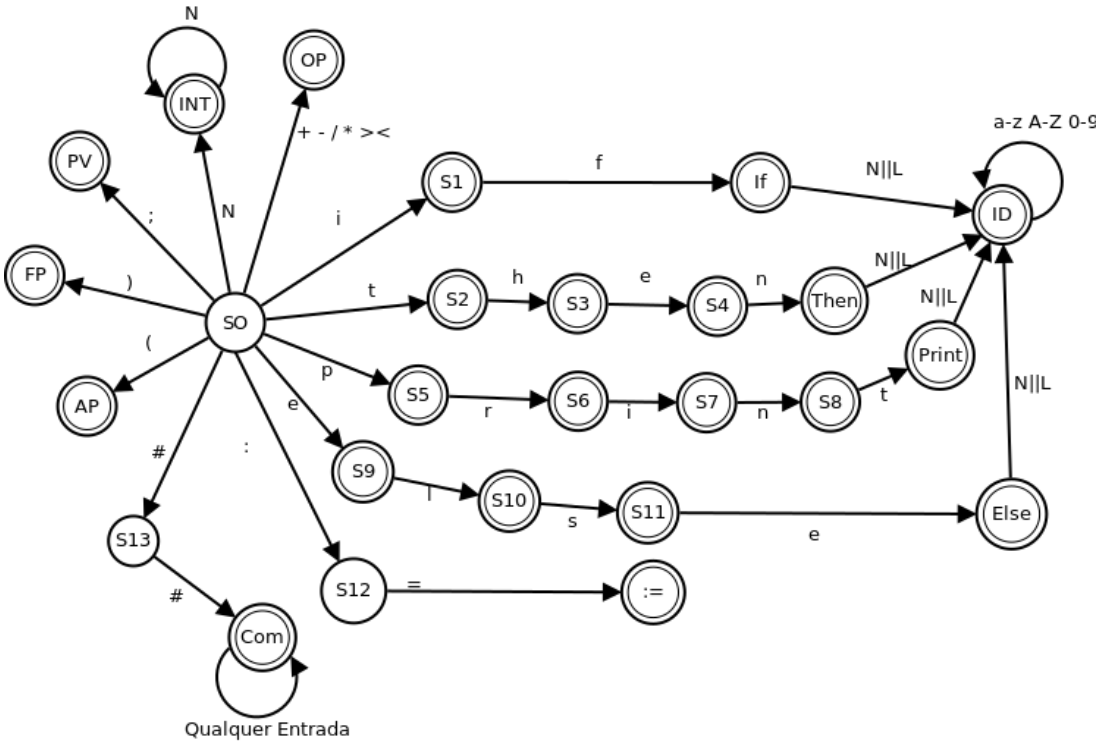
Nessa seção sera abordado um analisador léxico simples, o qual o professor Alexsandro Santos Soares aplicou em sala como atividade. A abordagem sera passo a passo mostrando a abordagem e construção pelo Autômato ate a implementação do mesmo em Ocaml

7.1 Abordagem por Autômato

Segue a versão do autômato com suas especificações, o qual será implementado em Ocaml.O automato que abordaremos nesse ponto não reconhece a linguagem Python.
É reconhecida uma linguagem convencionada e simplificada na sala de aula. Córido da linguagem foir fornecido pelo professor Alexsandro, feito durante as aulas de Compuladores;

Considerações realizadas:

- L: [a-z]U[A-Z];
- N: [0-9];
- INT: etado 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

```
let lexico (str:entrada) =
  let trans (e:estado) (c:simbolo) =
match (e,c) with
    | (0, 'i') -> 1
    | (0, 't') -> 6
    | (0, 'e') -> 10
    | (0, 'p') -> 14
    | (0, 'f') -> 25
    | (0, 'w') -> 28
    | (0, '(') -> 19
    | (0, ')') -> 20
    | (0, ';') -> 22
    | (0, ':') -> 23
    | (0, '#') -> 33
    | (0, _) when eh_operador c -> 21
    | (0, _) when eh_letra c -> 3
    | (0, _) when eh_digito c -> 4
    | (0, _) when eh_branco c -> 5
    | (0, _) -> failwith ("Erro lexico: caracter
       desconhecido " ^ Char.escaped c)

    | (1, 'f') -> 2
    | (1, _) when eh_letra c || eh_digito c -> 3

    | (2, _) when eh_letra c || eh_digito c -> 3

    | (3, _) when eh_letra c || eh_digito c -> 3

    | (4, _) when eh_digito c -> 4

    | (5, _) when eh_branco c -> 5

    | (6, 'h') -> 7
    | (6, _) when eh_letra c || eh_digito c -> 3

    | (7, 'e') -> 8
    | (7, _)  when eh_letra c || eh_digito c -> 3

    | (8, 'n') -> 9
    | (8, _)  when eh_letra c || eh_digito c -> 3

    | (9, _)  when eh_letra c || eh_digito c -> 3

    | (10, 'l') -> 11
    | (10, _) when eh_letra c || eh_digito c -> 3

    | (11, 's') -> 12
    | (11, _)  when eh_letra c || eh_digito c -> 3
```

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

```
    | _ -> estado_morto
 and rotulo e str =
  match e with
  | 2 -> If
  | 1
  | 6
  | 7
  | 8
  | 10
  | 11
  | 12
  | 14
  | 15
  | 16
  | 17
  | 25
  | 26
  | 28
  | 29
  | 30
  | 31
  | 3 -> Id str
  | 4 -> Int str
  | 5 -> Branco
  | 9 -> Then
  | 13 -> Else
  | 18 -> Print
  | 19 -> APar
  | 20 -> FPar
  | 21 -> OP str
  | 22 -> PV
  | 24 -> Atrib
  | 27 -> For
  | 32 -> While
  | _ -> failwith ("Erro lexico: sequencia desconhecida " ^
      str)
in let dfa = { transicao = trans;
               estado = estado_inicial;
               posicao = 0 }
in let estado_lexico = {
  pos_inicial = 0;
  pos_final = -1;
  ultimo_final = -1;
  rotulo = rotulo;
  dfa = dfa
} in
  analisador str (String.length str) estado_lexico
```

Seguem as funções principais feitas pelo professor que viabilizaram o fun- cionamento 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.

- analisador: responsável por analisar o estado corrente do estado léxico. Se estivermos no estado final, paramos, caso contrario, se estivermos no estado morto, vamos para o proximo estado e verificamos qual ele é

Analisador 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 empressões encontradas na sessão 7.1

```
{

  open Lexing
  open Printf

  type token =
  | LITINT of (int)
  | LITSTRING of (string)
  | ID of (string)
  | LITFLOAT of (float)
  | APAR
  | FPAR
  | VIRG
  | MAIS
  | DPONTOS
  | MENORIGUAL
  | SETA
  | E
  | ATRIB
  | RETURN
  | DEF
  | EOF
  (*tokens adicionados*)
  | OU
  | IF
  | ELSE
  | WHILE
  | FOR
  | MAIORIGUAL
  | IMPORT
  | INT
  | FLOAT
  | 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 (* O pr  processador necess rio para contabilizar a
      identa  o *)
118 rule preprocessador indentacao = parse
119   linha_em_branco          { preprocessador 0 lexbuf } (*
        ignora brancos *)
120 | [' ' '\t' ]+ '\n'        { incr_num_linha lexbuf;
121                              preprocessador 0 lexbuf } (*
                                   ignora brancos *)
122 | ' '                      { preprocessador (indentacao + 1)
      lexbuf }
123 | '\t'                     { let nova_ind = indentacao + 8 -
      (indentacao mod 8)
124                              in preprocessador nova_ind
                                   lexbuf }
125 | novalinha                { incr_num_linha lexbuf;
126                              preprocessador 0 lexbuf }
```

```
| restante as linha {
      let rec tokenize lexbuf =
          let tok = token lexbuf in
      match tok with
         EOF -> []
     | _ -> tok :: tokenize lexbuf in
      let toks = tokenize (Lexing.from_string linha) in
      (* A impress o a seguir serve apenas para depura  o
          . Retirar depois! *)
      printf "Linha(identacao=%d,nivel_par=%d)\n" indentacao
           (!nivel_par);
      Linha(indentacao,!nivel_par, toks)
}
| eof { nivel_par := 0; EOF }



(* O analisador l xico a ser chamado ap s o pr
   processador *)
and token = parse
  brancos             { token lexbuf }
| comentario          { token lexbuf }
| "'''"                { comentario_bloco 0 lexbuf; }
| ">="                { MAIORIGUAL}
| "<="                { MENORIGUAL }
| "->"                { SETA }
| "++"                { INCR }
| "--"                { DECR }
| "=="                { IGUALDADE }
| "!="                { DIFERENTE }
| "+="                { ATRIBMAIS }
| "-="                { ATRIBMENOS }
| "*="                { ATRIBMULT }
| "/="                { ATRIBDIV }
| '{'                 { ABRECHAVES }
| '}'                 { FECHACHAVES }
| '['                 { ABRECOLCHETES }
| ']'                 { FECHACOLCHETES }
| '('                 { let _ = incr(nivel_par) in APAR }
| ')'                 { let _ = decr(nivel_par) in FPAR }
| ','                 { VIRG }
| '+'                 { MAIS  }
| '-'                 { MENOS  }
| '*'                 { VEZES  }
| '/'                 { DIVIDIDO }
| '='                 { ATRIB }
| ':'                 { DPONTOS }
| ';'                 { PV }
| '<'                 { MENOR }
```

```
| '>'                 { MAIOR }
| '%'                     { MODULO }
| '.'                 { PONTO }
| int as num          { let numero = int_of_string num in
                       LITINT numero }
| "or"                { OU }
| "if"                { IF }
| "else"              { ELSE }
| "while"             { WHILE }
| "for"               { FOR }
| "return"            { RETURN }
| "def"               { DEF }
| "import"            { IMPORT }
| "int"               { INT }
| "float"             { FLOAT }
| "double"            { DOUBLE }
| "char"              { CHAR }
| "list"              { LIST }
| "and"               { E    }
| "in"                { IN }
| "range"             { RANGE }
| "pass"              { PASS }
| "void"              { VOID }
| "elif"                { ELIF }
| "print"             { PRINT }
| "str"               { STR }
| "input"             { INPUT }
| "len"               { LENGTH }
| "break"             { BREAK }
| "is"                { IS }
| "not"                 { NOT }
| "from"                { FROM }
| identificador as id { ID id }
| '"'                 { let pos = lexbuf.lex_curr_p in
                        let lin = pos.pos_lnum
                        and col = pos.pos_cnum - pos.pos_bol
                            - 1 in
                        let buffer = Buffer.create 1 in
                        let str = leia_string lin col buffer
                            lexbuf in
                            LITSTRING str }
| _ as c  { failwith (msg_erro lexbuf c) }
| eof        { EOF }



and comentario_bloco n = parse
   "'''"      { if n=0 then token lexbuf
                else comentario_bloco (n-1) lexbuf }
| "'''"       { 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 codigo escrito em um arquivo chamado "codigo":
lex "codigo";

Supondo o seguinte arquivo de código abaixo:

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

A saída do analisador léxico é a seguinte:

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
EOF
- : tokens =
            [Lexico.DEF; Lexico.ID "paco"; Lexico.APAR;
            Lexico.ID "x"; Lexico.FPAR;Lexico.DPONTOS;
            Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "x";
            Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
            Lexico.LITINT 1; Lexico.NOVALINHA;Lexico.ID "y";
            Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
            Lexico.LITINT 2;Lexico.NOVALINHA; Lexico.ID "y";
            Lexico.ATRIB; Lexico.ID "x"; Lexico.MAIS;
            Lexico.LITINT 3; Lexico.MAIS; Lexico.APAR;
            Lexico.LITINT 4; Lexico.MAIS;Lexico.LITINT 5;
            Lexico.FPAR; Lexico.NOVALINHA; Lexico.RETURN;
            Lexico.ID "x"; Lexico.NOVALINHA; Lexico.DEDENTA;
            Lexico.EOF]
```

7.2 Abordagem por Linguagem Regular

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

7.2.1 Implementação

Os principais tokens são NOVALINHA, IDENTA e DEDENTA que re- alizam o controle de identação 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 identação, O token DEDENTA é utilizado toda a vez a identação volta ao normal o que significa o fim de um comando.

Fazendo uma analogia com a linguagem C, pode-se dizer que o token NOVA- LINHA representa a vırgula, IDENTA representa abrir chaves e DE- DENTA, fechar chaves.

7.2.2 Execução

Código

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

Saída

```
# lex "nano02.py";;
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=-1)
Nivel: 8
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano02"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB; Lexico.INT;
 Lexico.ID "n"; Lexico.FPAR; Lexico.DEDENTA; Lexico.EOF]
```

Código

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

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano03"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
```

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

Código

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

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano03"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
     ATRIB; Lexico.INT;
 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;
     Lexico.ID "n";
 Lexico.ATRIB; Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.
     DEDENTA; Lexico.EOF]
```

Código

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

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano04"; Lexico.APAR; Lexico.FPAR;
    Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
     ATRIB; Lexico.INT;
 Lexico.APAR; Lexico.ID "n"; Lexico.FPAR; Lexico.NOVALINHA;
     Lexico.ID "n";
```

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

Código

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

nano05()
```

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=0,nivel_par=0)
Nivel: 0
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano05"; Lexico.APAR; Lexico.FPAR; Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.ATRIB;
 Lexico.LITINT 2; Lexico.NOVALINHA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
 Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING "";
 Lexico.FPAR; Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ID "nano05";
 Lexico.APAR; Lexico.FPAR; Lexico.NOVALINHA; Lexico.EOF]
```

Código

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

nano06()
```

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=0,nivel_par=0)
```

```
Nivel: 0
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano06"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
 Lexico.LITINT 1; Lexico.MENOS; Lexico.LITINT 2; Lexico.
    NOVALINHA;
 Lexico.PRINT; Lexico.APAR; Lexico.ID "n"; Lexico.VIRG;
    Lexico.ID "end";
 Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.FPAR; Lexico.
    NOVALINHA;
 Lexico.DEDENTA; Lexico.ID "nano06"; Lexico.APAR; Lexico.
    FPAR;
 Lexico.NOVALINHA; Lexico.EOF]
```

Código

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

nano07()
```

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=16,nivel_par=0)
Nivel: 16
Linha(identacao=0,nivel_par=0)
Nivel: 0
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano07"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
    NOVALINHA;
 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;
```

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

Código

```
def nano08():
    n=1
    if n ==1:
        print(n,end="")
    else:
        print(0,end="")

nano08()
```

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=16,nivel_par=0)
Nivel: 16
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=16,nivel_par=0)
Nivel: 16
Linha(identacao=0,nivel_par=0)
Nivel: 0
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano08"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
     ATRIB;
 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
     ;
 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
     NOVALINHA;
 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
     Lexico.VIRG;
 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
     FPAR;
 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.
     DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.PRINT; Lexico.APAR
     ;
 Lexico.LITINT 0; Lexico.VIRG; Lexico.ID "end"; Lexico.ATRIB
     ;
```

```
 Lexico.LITSTRING ""; Lexico.FPAR; Lexico.NOVALINHA; Lexico.
    DEDENTA;
 Lexico.DEDENTA; Lexico.ID "nano08"; Lexico.APAR; Lexico.
    FPAR;
 Lexico.NOVALINHA; Lexico.EOF]
```

Código

```
def nano09():
    n=1
    if n ==1:
        print(n,end="")
    else:
        print(0,end="")

nano09()
```

Saída

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=16,nivel_par=0)
Nivel: 16
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=16,nivel_par=0)
Nivel: 16
Linha(identacao=0,nivel_par=0)
Nivel: 0
EOF
- : tokens =
[Lexico.DEF; Lexico.ID "nano09"; Lexico.APAR; Lexico.FPAR;
   Lexico.DPONTOS;
 Lexico.NOVALINHA; Lexico.INDENTA; Lexico.ID "n"; Lexico.
    ATRIB;
 Lexico.LITINT 1; Lexico.NOVALINHA; Lexico.IF; Lexico.ID "n"
    ;
 Lexico.IGUALDADE; Lexico.LITINT 1; Lexico.DPONTOS; Lexico.
    NOVALINHA;
 Lexico.INDENTA; Lexico.PRINT; Lexico.APAR; Lexico.ID "n";
    Lexico.VIRG;
 Lexico.ID "end"; Lexico.ATRIB; Lexico.LITSTRING ""; Lexico.
    FPAR;
 Lexico.NOVALINHA; Lexico.DEDENTA; Lexico.ELSE; Lexico.
    DPONTOS;
 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():
```

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

Saída

Erro: Variável com caractere proibido

```
Linha(identacao=0,nivel_par=0)
Nivel: 0

Exception: Failure "1-0: caracter desconhecido @".
```

Código

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

micro01()
```

Saída

Erro: Comentario não fechado

```
Linha(identacao=0,nivel_par=0)
Nivel: 0
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8
Linha(identacao=8,nivel_par=0)
Nivel: 8

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 sımbolos que ocorrem no inıcio de uma determinada regra.
Follow: o que pode aparecer depois da ocorrência de um determinado sımbolo.

	anulável	FIRST	FOLLOW
S	Não	a c d	
X	Sim	a	b c d e f
Y	Não	c d	e f
Z	Não	e f	c d

	a	b	c	d	e	f
S	XYZ		XYZ	XYZ		
X	aXb	ε	ε	ε	ε	ε
Y			cYZcX	d		
Z					eZYe	f

figura 1

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 sintátiva 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"
```

```
  | D -> "d"
  | E -> "e"
  | F -> "f"
  | EOF -> "eof"

let erro esp =
  let msg = Printf.sprintf "Erro: esperava %s mas encontrei %s"
                           esp (to_str !tk)
  in
  failwith msg

let consome t = if (!tk == t) then prox() else erro (to_str t)

let rec ntS () =
  match !tk with
    A
   |C
   |D     ->
              let cmd1 = ntX() in
              let cmd2 = ntY() in
              let cmd3 = ntZ() in
              S (cmd1, cmd2, cmd3)
  | _ -> erro "a, c ou d"
and ntX () =
  match !tk with
     B
    |C
    |D
    |E
    |F    -> X_vazio
    |A    -> let _ = consome A in
             let cmd = ntX() in
             let _ = consome B in
             X (A, cmd, B)
    | _ -> erro "a"
and ntY () =
  match !tk with
    C    -> let _ = consome C in
            let cmd = ntY() in
            let cmd2 = ntZ() in
            let _ = consome C in
            let cmd3 = ntX() in
            Y (C,cmd,cmd2, C, cmd3)
   |D     -> let _ = consome D in
             Y_d (D)
   |_     -> erro "c ou d"
and ntZ () =
  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. Ja com os arquivos compilados abra o Ocaml

ocamla

5. Com o Ocaml executando, digite o codigo 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 de- pendências 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.cm
```

8.2.1 Execução de Testes

sintaticoArv.ml

```
def micro11() -> int:
    numero = 0
    x =0
    print("numero")
    numero = input()

    numero = verifica(numero)
    if x ==1:
        print("Positivo")
    if x ==0:
        print("zero")
    else:
        print("Negativo")

    return 0

def verifica(n:int) -> int:
    res = 0
    if n>0:
        res = 1
    if n<0:
        res = 2
    else:
        res = 0

    return res

micro11()
```

Saída

```
- : Ast.prog =
Prog ([],
 [DEFUNCAO([], Some INTEIRO,
   [ATRIBUI  0 ("numero", ExpTerm (TERMTL (LITINT 0)));
     ATRIBUI  0 ("X", ExpTerm (TERMTL (LITINT 0))); PRINT
        ("numero", []);
     LEIA "numero";
     CHAMADADEFUNCAO ("numero", Some "verifica", [TERMID "
        numero"]);
     CONDICAOIF
       (EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (
          TERMTL (LITNT 1))),
       [], None);
     CONDICAOIF
       (EXP (ExpTerm (TERMID "x"), IGUALDADE, ExpTerm (
          TERMTL (LITINT 0))),
       [], Some ELSECOND);
```

```
     RETORNO (TERMTL (LITINT 0))]);
 DEFUNCAO ([INTEIRO], Some INTEIRO,
   [ATRIBUICAO ("res", ExpTerm (TERMTL (LITINIT 0)));
    CONDICAOIF
      (EXP (ExpTerm (TERMID "n"), MAIOR, ExpTerm (TERMTL (
          LITINT 0))),
      [],None);
    CONDICAOIF
      (EXP (ExpTerm (TERMID "n"), MENOR, ExpTerm (TERMTL (
          LITINT 0))),
      [], Some ELSECOND);
RETORNO (TERMID "res"); CHAMADADEFUNCAO ("micro11", None,
    [])])])
```

sintatico.mli

```
    import x
    def micro():
        numero =0
        numero = input()
        if numero>= 100:
            if numero<= 200:
                print("e")
            else:
                print("d")
        else:
            print("f")

    micro()
```

Saida

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

9 Analisador Semântico

Essa seção contém a implementação de um analisador semântico, com a implementação foram realizadas mudanças nos arquivos do sintático e léxico com o

objetivo de facilitar a integração.

Para verificarmos a arvore semantica de um programa abra o ocaml e execute

verifica$_t$*ipos*"*nome*$_a$*rquivo.py*";;

Variáveis do código serão exibidas com os paramêtros linha e coluna, segue exemplo abaixo:

FuncaoExemplo

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

Saída

```
1  - : Tast.expressao Ast.programa * Ambiente.t =
2  (Programa
3    [Funcao
4      {fn_nome =
5        ("main",
6         {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
             pos_cnum = 4});
7       fn_tiporet = INTEIRO; fn_formais = [];
8       fn_corpo =
9        [LEIAF (Tast.EXPVAR ("valor", REAL));
10        ATRIBUICAO (Tast.EXPVAR ("x", REAL),
11         Tast.EXPOPB ((ADICAO, REAL), (Tast.EXPVAR ("valor",
               REAL), REAL),
12          (Tast.EXPFLOAT (1., REAL), REAL)));
13        RETORNO (Some (Tast.EXPINT (1, INTEIRO)))]}],
14   <abstr>)
```

9.0.1 Execução de Testes

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

Para executar os testes va ate o diretorio que contenha oarquivo Semantico.mli e execute

modulo1.py

```
def main() -> int:
    numero = 0
    x =0
    print("Digite um numero: ")
    inputi(numero)
    numero = verifica(numero)

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0


def verifica(n:int) -> int:
    res = 0
    if n > 0:
        return 1
    if n < 0:
        return 3
    else:
        return 0

main()
```

Saída

```
- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
  [Funcao
    {fn_nome =
      ("main",
       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
           pos_cnum = 4});
     fn_tiporet = INTEIRO; fn_formais = [];
     fn_corpo =
      [ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO), Tast.
         EXPINT (0, INTEIRO));
       ATRIBUICAO (Tast.EXPVAR ("x", INTEIRO), Tast.EXPINT
           (0, INTEIRO));
       PRINT (Tast.EXPSTRING ("Digite um numero: ", STRING))
           ;
       LEIAI (Tast.EXPVAR ("numero", INTEIRO));
       ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO),
```

```
        Tast.EXPCALL ("verifica", [Tast.EXPVAR ("numero",
            INTEIRO)], INTEIRO));
       CONDICAOIF
        (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
          (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tast.EXPINT (1, INTEIRO), INTEIRO)),
        [PRINT (Tast.EXPSTRING ("Positivo", STRING))],
        Some
         (CONDICAOIF
           (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
             (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
             (Tast.EXPINT (0, INTEIRO), INTEIRO)),
           [PRINT (Tast.EXPSTRING ("zero", STRING))],
           Some
            (CONDICAOElifElse [PRINT (Tast.EXPSTRING ("
                Negativo", STRING))]))));
      RETORNO (Some (Tast.EXPINT (0, INTEIRO)))]};
  Funcao
   {fn_nome =
     ("verifica",
      {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
          pos_cnum = 4});
    fn_tiporet = INTEIRO;
    fn_formais =
     [(("n",
        {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
            pos_cnum = 13}),
       INTEIRO)];
    fn_corpo =
     [ATRIBUICAO (Tast.EXPVAR ("res", INTEIRO), Tast.EXPINT
         (0, INTEIRO));
      CONDICAOIF
       (Tast.EXPOPB ((MAIORQ, BOOLEAN),
         (Tast.EXPVAR ("n", INTEIRO), INTEIRO),
         (Tast.EXPINT (0, INTEIRO), INTEIRO)),
       [RETORNO (Some (Tast.EXPINT (1, INTEIRO)))], None);
      CONDICAOIF
       (Tast.EXPOPB ((MENORQ, BOOLEAN),
         (Tast.EXPVAR ("n", INTEIRO), INTEIRO),
         (Tast.EXPINT (0, INTEIRO), INTEIRO)),
       [RETORNO (Some (Tast.EXPINT (3, INTEIRO)))],
       Some (CONDICAOElifElse [RETORNO (Some (Tast.EXPINT
           (0, INTEIRO)))]))]};],
<abstr>)
```

modulo2.py

```
def main() -> None:
    print("Digite um numero:  ")
    inputi(numero)
```

```
    if numero>= 100:
        if numero<= 200:
            print("\n numero entre 100 e 200")
        else:
            print("\nnumero maior que 200")
    else:
        print("\nnumero menor que 100")

    return
main()
```

Saída

```
- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
  [Funcao
    {fn_nome =
      ("main",
       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
           pos_cnum = 4});
     fn_tiporet = NONE; fn_formais = [];
     fn_corpo =
      [PRINT (Tast.EXPSTRING ("Digite um numero:  ", STRING)
         );
       LEIAI (Tast.EXPVAR ("numero", INTEIRO));
       CONDICAOIF
        (Tast.EXPOPB ((MAIORIGUALQ, BOOLEAN),
          (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tast.EXPINT (100, INTEIRO), INTEIRO)),
        [CONDICAOIF
          (Tast.EXPOPB ((MENORIGUALQ, BOOLEAN),
            (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
            (Tast.EXPINT (200, INTEIRO), INTEIRO)),
          [PRINT (Tast.EXPSTRING ("\n numero entre 100 e 200
              ", STRING))],
          Some
           (CONDICAOElifElse
             [PRINT (Tast.EXPSTRING ("\nnumero maior que 200
                 ", STRING))]))],
        Some
         (CONDICAOElifElse
           [PRINT (Tast.EXPSTRING ("\nnumero menor que 100",
               STRING))]));
       RETORNO None]}],
 <abstr>)
```

modulo3.py

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

```
    x=50
    while n < x:
        n = n + 4 + m
        print("n = ")
        print(n)
        print("\n")

main()
```

Saída

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

modulo4.py

```
def main() -> None:
    numero =1
    while numero < 0 or numero >0:
        print("\n Digite um numero: ")
        inputi(numero)
        if numero > 10:
            print("\n Numero maior que 10")
```

```
        else:
            print("\n Numero menor que 10")


main()
```

Saída

```
- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
  [Funcao
    {fn_nome =
      ("main",
       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
           pos_cnum = 4});
     fn_tiporet = NONE; fn_formais = [];
     fn_corpo =
      [ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO), Tast.
         EXPINT (1, INTEIRO));
       WHILELOOP
        (Tast.EXPOPB ((OR, BOOLEAN),
          (Tast.EXPOPB ((MENORQ, BOOLEAN),
            (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
            (Tast.EXPINT (0, INTEIRO), INTEIRO)),
           BOOLEAN),
          (Tast.EXPOPB ((MAIORQ, BOOLEAN),
            (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
            (Tast.EXPINT (0, INTEIRO), INTEIRO)),
           BOOLEAN)),
        [PRINT (Tast.EXPSTRING ("\n Digite um numero: ",
           STRING));
         LEIAI (Tast.EXPVAR ("numero", INTEIRO));
         CONDICAOIF
          (Tast.EXPOPB ((MAIORQ, BOOLEAN),
            (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
            (Tast.EXPINT (10, INTEIRO), INTEIRO)),
          [PRINT (Tast.EXPSTRING ("\n Numero maior que 10",
             STRING))],
          Some
           (CONDICAOElifElse
             [PRINT (Tast.EXPSTRING ("\n Numero menor que 10
                ", STRING))]))])]}],
 <abstr>)
```

modulo6.py

```
def main() -> int:
    numero = 0

    print("Digite um numero de 1 a 5: ")
    inputi(numero)
```

```
    if numero ==1:
        print("Um")
    elif numero == 2:
        print("Dois")
    elif numero ==3:
        print("Tres")
    elif numero ==4:
        print("Quatro")
    elif numero ==5:
        print("Cinco")
    else:
        print("Numero Invalido!!!")

main()
```

Saída

```
- : Tast.expressao Ast.programa * Ambiente.t =
(Programa
  [Funcao
    {fn_nome =
      ("main",
       {Lexing.pos_fname = ""; pos_lnum = 1; pos_bol = 0;
           pos_cnum = 4});
     fn_tiporet = INTEIRO; fn_formais = [];
     fn_corpo =
      [ATRIBUICAO (Tast.EXPVAR ("numero", INTEIRO), Tast.
          EXPINT (0, INTEIRO));
       PRINT (Tast.EXPSTRING ("Digite um numero de 1 a 5: ",
           STRING));
       LEIAI (Tast.EXPVAR ("numero", INTEIRO));
       CONDICAOIF
        (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
          (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
          (Tast.EXPINT (1, INTEIRO), INTEIRO)),
        [PRINT (Tast.EXPSTRING ("Um", STRING))],
        Some
         (CONDICAOIF
           (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
             (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
             (Tast.EXPINT (2, INTEIRO), INTEIRO)),
           [PRINT (Tast.EXPSTRING ("Dois", STRING))],
           Some
            (CONDICAOIF
              (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
                (Tast.EXPVAR ("numero", INTEIRO), INTEIRO),
                (Tast.EXPINT (3, INTEIRO), INTEIRO)),
              [PRINT (Tast.EXPSTRING ("Tres", STRING))],
              Some
               (CONDICAOIF
                 (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
```

```
                    (Tast.EXPVAR ("numero", INTEIRO), INTEIRO
                       ),
                    (Tast.EXPINT (4, INTEIRO), INTEIRO)),
                  [PRINT (Tast.EXPSTRING ("Quatro", STRING))
                     ],
                  Some
                   (CONDICAOIF
                     (Tast.EXPOPB ((EHIGUAL, BOOLEAN),
                       (Tast.EXPVAR ("numero", INTEIRO),
                          INTEIRO),
                       (Tast.EXPINT (5, INTEIRO), INTEIRO)),
                     [PRINT (Tast.EXPSTRING ("Cinco", STRING)
                        )],
                     Some
                      (CONDICAOElifElse
                        [PRINT (Tast.EXPSTRING ("Numero
                           Invalido!!!", STRING))])))))))))
                           ]})],
 <abstr>)
```

9.0.2 Execução de Testes

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

Para executar os testes va ate o diretorio que contenha oarquivo Semantico.mli e execute

modulo1.py

```
def main() -> int:
    numero = 0
    x =0
    print("Digite um numero: ")
    inputi(numero)
    numero = verifica("String Erro Semantico")

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0


def verifica(n:int) -> int:
    res = 0
```

```
    if n > 0:
        return 1
    if n < 0:
        return 3
    else:
        return 0

main()
```

Saída

```
# verifica_tipos "../testes/modulo1.py";;
EOF
Exception:
Failure
 "Semantico -> linha 1, coluna 39: O parametro eh do tipo
    string mas deveria ser do tipo inteiro".
```

10 Interpretador Utilizando Menhir

Nessa sessão iremos executar nossos interpretador utilizando a ferramenta Menhir, para isso siga os passos de execução abaixo para buildar e compilar os arquivos.

10.0.1 Execução Interpretador

Antes devemos excluir o diretorio build com rm -rf build e excluir o arquivo interpretadorTeste.byte caso ele existe com rm interpre- tadorTeste.byte.

Para executar o interpretador deve-se digitar:

ocamlbuild -use-ocamlfind -use-menhir -menhir

ocamlbuild"menhir –table-package menhirLib interpreteTeste.byte

E depois abrir o ocaml: **rlwrap ocaml**

Após isso, execute o interpretador com **interprete "../Codigos/NomArquivo.py";;**

10.0.2 Resultado de Execuções do Interprete

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

modulo1.py

```
def main() -> int:
    numero = 0
    x =0
    print("Digite um numero: ")
    inputi(numero)
    numero = verifica(numero)

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0


def verifica(n:int) -> int:
    res = 0
    if n > 0:
        return 1
    if n < 0:
        return 3
    else:
        return 0

main()
```

Saída

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

modulo2.py

```
def main() -> None:
    print("Digite um numero:  ")
    inputi(numero)
    if numero>= 100:
        if numero<= 200:
            print("\n numero entre 100 e 200")
        else:
```

```
            print("\nnumero maior que 200")
    else:
        print("\nnumero menor que 100")

    return
main()
```

Saída

```
# interprete "../testes/e2.py";;
EOF
Digite um numero:  50

numero menor que 100- : unit = ()
# interprete "../testes/e2.py";;
EOF
Digite um numero:  150

 numero entre 100 e 200- : unit = ()
# interprete "../testes/e2.py";;
EOF
Digite um numero:  300

numero maior que 200- : unit = ()
```

modulo7.py

```
def main() -> None:
    numero =0
    fat = 0
    print("Digite um numero: ")
    inputi(numero)
    fat = fatorial(numero)

    print("O fatorial eh ")
    print(fat)

def fatorial(n: int) -> int:
    if n <= 0:
        return 1
    else:
        return n * fatorial(n - 1)

main()
```

Saída

```
# interprete "../testes/e7.py";;
EOF
Digite um numero: 5
O fatorial eh 120- : unit = ()
```

10.0.3 Erros de Execuções do Interprete

Nesta seção, sera mostrados alguns avisos de erros gerados pelo interpretador, sera utilizado codigos em Python com erros introduzidos propositalmente.

Erro Léxico Erro Léxico induzido na declaração da variavel x

modulo1.py

```
def main() -> int:
    numero = 0
    @x =0
    print("Digite um numero: ")
    inputi(numero)
    numero = verifica(numero)

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0


def verifica(n:int) -> int:
    res = 0
    if n > 0:
        return 1
    if n < 0:
        return 3
    else:
        return 0

main()
```

Saída

```
# interprete "../testes/modulo1.py";;
Exception: Failure "1-0: caracter desconhecido @".
```

Erro Sintatico Erro Sintático induzido no não fechamento da funcao print

modulo1.py

```
def main() -> int:
    numero = 0
    x =0
    print("Digite um numero: "
    print(x)
```

```
    inputi(numero)
    numero = verifica(numero)

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0


def verifica(n:int) -> int:
    res = 0
    if n > 0:
        return 1
    if n < 0:
        return 3
    else:
        return 0

main()
```

Saída

```
# interprete "../testes/modulo1.py";;
Erro sint tico na linha 1, coluna -1 0 - <declaracao
    invalida>
```

Erro Semantico Erro Semantico induzido na passagem de mais de um parametro na chamada da função verifica.

modulo1.py

```
def main() -> int:
    numero = 0
    x =0
    print("Digite um numero: ")
    inputi(numero)
    numero = verifica(numero,x)

    if numero == 1:
        print("Positivo")
    elif numero == 0:
        print("zero")
    else:
        print("Negativo")

    return 0
```

```
16
17
18  def verifica(n:int) -> int:
19      res = 0
20      if n > 0:
21          return 1
22      if n < 0:
23          return 3
24      else:
25          return 0
26
27  main()
```

Saída

```
1  # interprete "../testes/modulo1.py";;
2  EOF
3  Exception:
4  Failure "Semantico -> linha 1, coluna 8: Numero incorreto de
       parametros".
```

11 Referências

11.1 Bibliografias

Construção de Compiladores - Gabriel Augusto Marson, Leonardo da Silva Martins, Angelo Travizan Neto, Patrícia Marcolino

11.2 Webgráfias

Python tutorial - TutorialsPointz
www.codecademy.com
www.codecademy.com