Warlock compiler

V6.66

Grupo
Caio Saracuza
Eduardo Ramos

In memoriam
~~*Arthur Vieira*~~
~~*Beatriz Campos*~~

Bloco

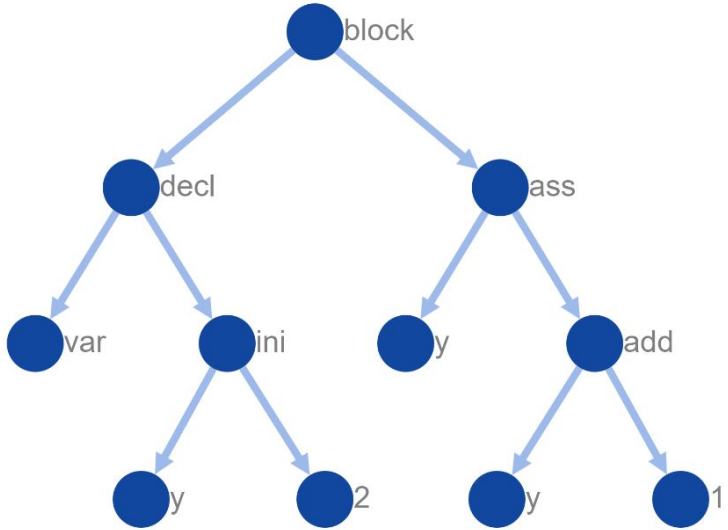
Formal:

<block> ::= ‘{‘ <decl-seq>? <cmd>+ ‘}’

ImpCompiler:

Block = _"{"_ l: DeclSeq? r: Sequence? _"}"_ { return{left:l, operator:"block", right:r}}

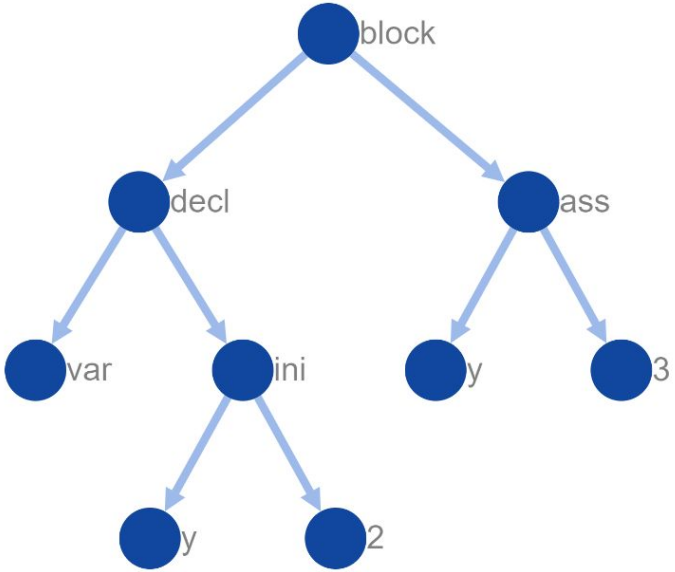
```
organizaBlock() {

    this.organizaExpressoes();

    this.empilhaValor(this.E == null ? new Map() : this.E);

    this.E = new Map(this.E);
}

resolveBlock() {
    //Tira o block do controle
    var dispose = this.desempilhaControle();

    //Retoma o ambiente externo
    this.E = this.desempilhaValor();
}
```



{var y=2; y:=y+1;}

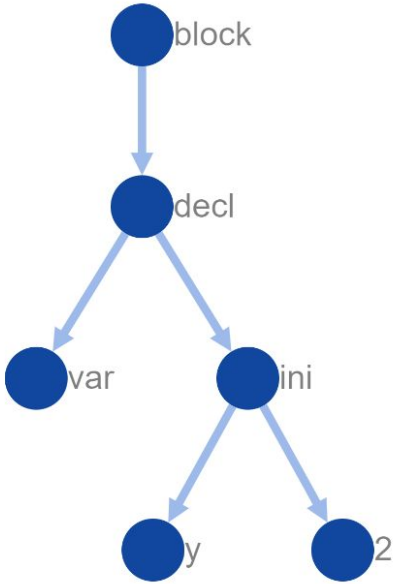
Bloco

{var y=2; y:=3;}

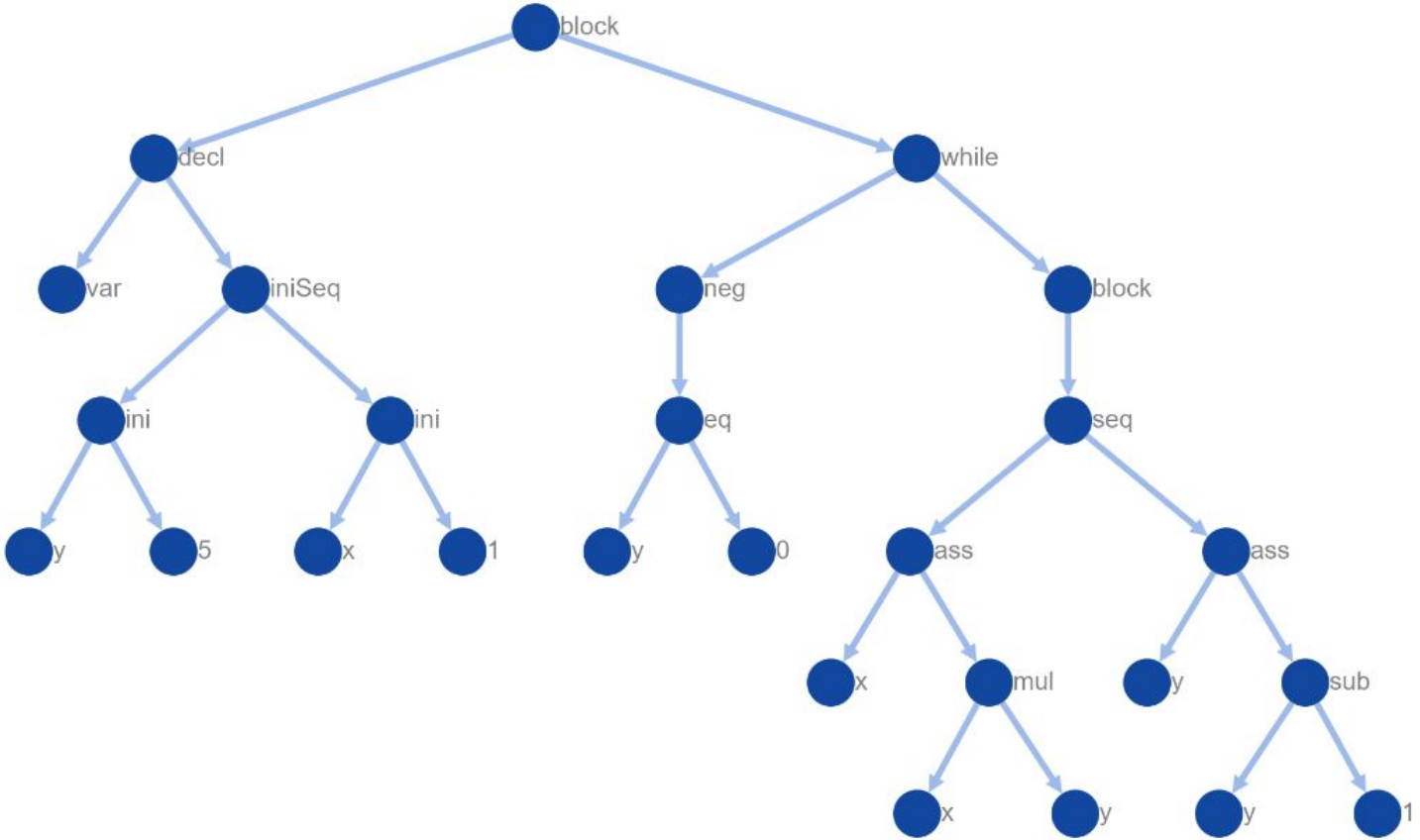


OU

{var y=2;}



Bloco


block
decl
while
var
iniSeq
neg
block
ini
ini
eq
seq
y
5
x
1
y
0
ass
ass
x
mul
y
sub
x
y
y
1


This was
NOT
in my contract!

Declarações

Formal:

<decl-seq> ::= <decl> | <decl> ‘;’ <decl-seq>

<decl> ::= <decl-op> <ini-seq>

ImpCompiler:

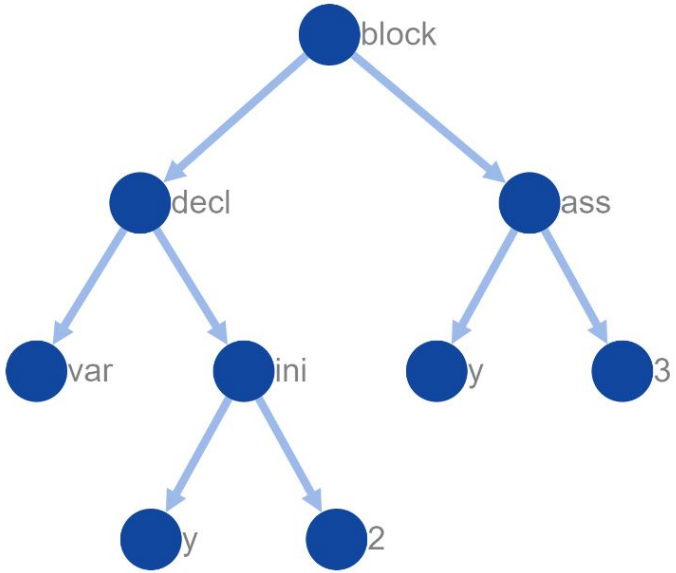
```
DeclSeq=  l: Declaration  ';'  r: DeclSeq   {return {left:l,operator:'declSeq',right:r}}

          / v: Declaration  ';'     {return v}

Declaration =  l: declop   __   r: IniSeq      {return {left:l,operator:'decl',right:r}}
```

Declarações

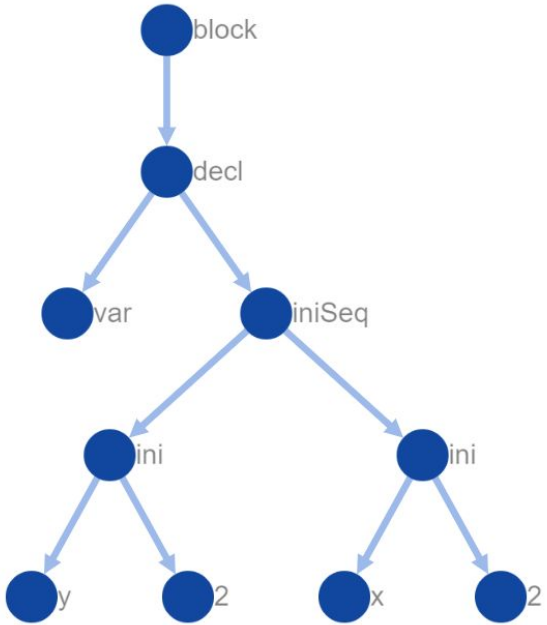
```
organizaDeclaracao() {

    var tree = this.desempilhaControle();
    //Empilha primeiro para saber quando acabar a Declaracao e assim conseguir remover o varConst da pilha de valor
    this.empilhaControle(tree.operator);

    //Empilha o resto no controle(ini, iniseq)
    this.empilhaControle(tree.right);

    //Empilha o Var ou Const em Valor
    this.empilhaValor(tree.left);


}

resolveDeclaracao() {
    //remove o decl de controle
    this.desempilhaControle();

    //remove o varConst de valor
    this.desempilhaValor();
}
```

Declarações

{var y=2; y:=3;}

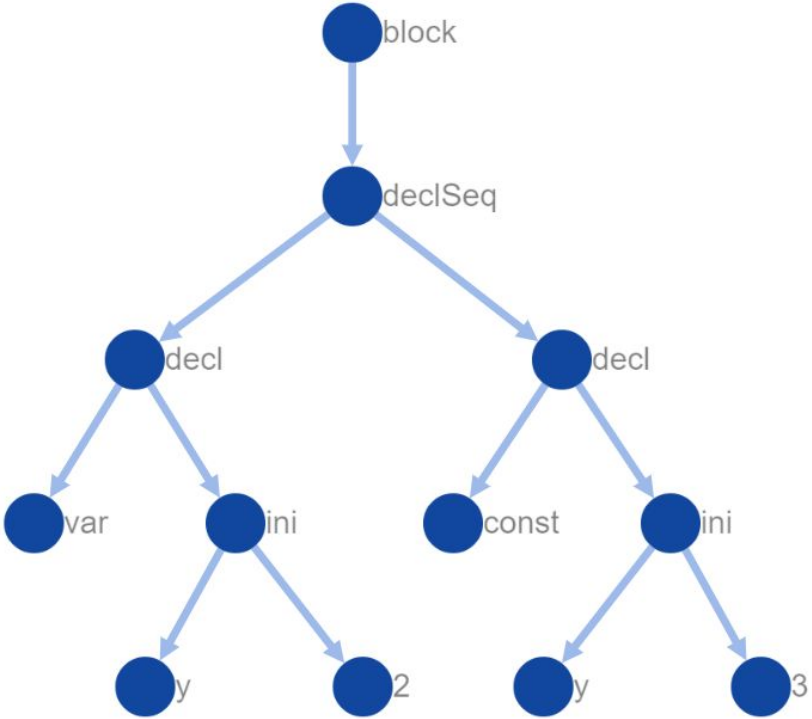


{var y=2,x=2;}

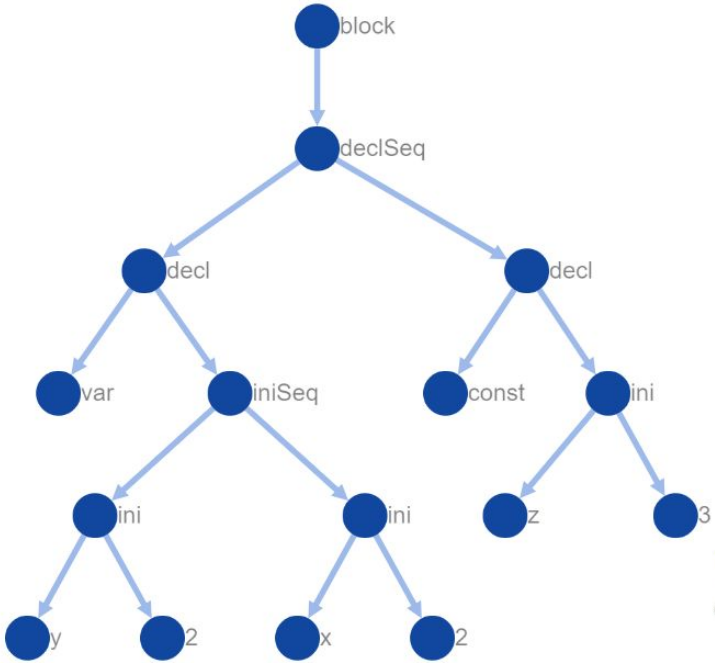


Declarações

{var y=2; const y=3;}



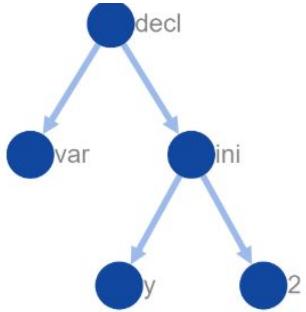
{var y=2,x=2; const z=3;}



Declarações

Fluxograma Árvore ‘decl’:

1. Encontra árvore com “operador” ‘decl’ na pilha Controle.
2. Desempilha de Controle.
3. Empilha o string ‘decl’ na pilha Controle.
4. Empilha filho direito na pilha Controle.
5. Empilha filho esquerdo na pilha Valor.



Fluxograma palavra reservada ‘decl’:

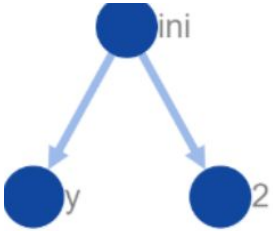
1. Encontra palavra ‘decl’ na pilha Controle.
2. Desempilha de Controle.
3. Desempilha de Valor a string que controla “var” ou “const”.

Declarações

Fluxograma Árvore ‘ini’:

1. Encontra árvore com “operador” ‘ini’ e resolve como expressão:

- Ao encontrar a árvore:



- Desmembra-a e empilha em controle na seguinte ordem:
  ini > 2 > y |início de pilha|

Fluxograma palavra reservada ‘ini’:

1. Desempilha da pilha de Valor, o valor da variável ou constante.
2. Desempilha da pilha de Valor, o identificador.
3. Desempilha da pilha de Valor, a string que identifica se é var ou const.
4. Com base nessa string, chama o método de declarar apropriado.
5. Empilha a string “var” ou “const” em valor.

Bloco

Formal:

<ini-seq> ::= <ini> | <ini> ‘,’ <ini-seq>

<ini> ::= <iden> ‘=’ <exp>

ImpCompiler:

```
IniSeq =  l:Ini ',' r:IniSeq  {return {left:l,operator:'iniSeq',right:r}}
          /Ini
Ini =  l:ident   op:'='   r:Expression  {return {left:l,operator:'ini',right:r}}
```

```
resolveIni() {
    //Tira o ini da pilha de controle
    this.desempilhaControle();

    //Desempilha da pilha de valor o identificador, seu valor e o controle de var ou const
    var value = this.desempilhaValor();
    var ident = this.desempilhaValor();
    var varConst = this.desempilhaValor();

    if (varConst == "var") {
        this.declaraVariavel(ident, value);
    } else if (varConst == "const") {
        this.declaraConstante(ident, value);
    } else {
        Console.log("======DEBUG==========Erro: Algo deu errado,era esperado 'var' ou 'const' da pilha de valor");
    }

    this.empilhaValor(varConst);
}
```

Implementação do Ambiente:

- Ambiente -> Map()
- Identificadores apontam para:
  - Caso variável: Location
  - Caso constante: Valor
- Ambiente orientado à declaração
  - Na implementação, é orientado ao bloco
- Verificação de associação

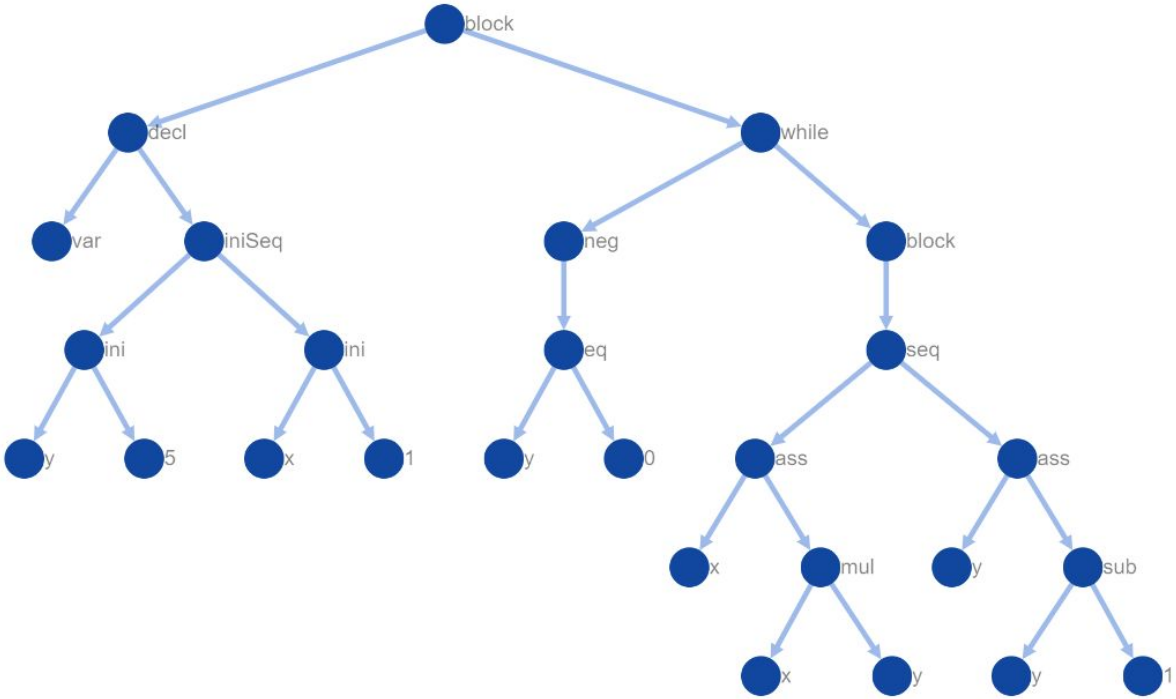
```
class Memory {

    constructor(M) {
        this.M = M;
        this.address = 0;
    }

    acessaMemoria(loc) {
        return this.M.get(loc.address);
    }

    insereMemoria(value) {
        var loc = new Location(this.address);

        //Salva na memoria o valor
        this.M.set(loc.address, value);

        //Aumenta o contador do endereço
        this.address++;

        return loc;
    }

    atualizaMemoria(loc, value) {

        if (this.M.has(loc.address)) {
            this.M.set(loc.address, value);
        }
    }

}
```

Memória

- Memória é uma Classe
- Responsável pela atribuição de endereços
- Possui um atributo Map()
  - Representa a memória

Exemplos

{var y = 5, x = 1; while ~(y==0) { x := x*y; y:= y-1;}}



FIM