

# **Classificação das regras semânticas e conjunto de instruções de código intermediário**

**Alunos:** Leonardo Apicelo / Gabriel Moura

## **1 – Classificação das regras semânticas:**

- **Regras semânticas:**

1. Não declarar mais de 1 ID com mesmo nome no mesmo escopo
2. Declaração de Id no escopo antes do uso
3. Só permite atribuição e operações com tipos iguais
4. Só posso passar parâmetros para funções
5. Quantidade de parâmetros na chamada da função deve ser igual a da declaração
6. O tipo dos argumentos passados nas funções deve ser igual ao tipo dos

parâmetros

7.O tipo retornado deve ser igual ao tipo de retorno da função

8. Só pode usar índice ([]) em variáveis do tipo vetor

9. Só pode usar membros (.) em variáveis do tipo classe

10. Só posso acessar membros de classe declarado

Não terminal	Regras	Justificativa
PROGRAMA	-	Só organiza a estrutura, sem semântica própria
CORPO	-	Apenas agrupa blocos
DECLARACOES	-	Encapsula seções, sem semântica direta
DEF_CONST	-	Encapsula seções, sem semântica direta

LISTA_CONST	-	Só estrutura
LISTA_CONST'	-	Só estrutura

CONSTANTE	1	Ponto onde um ID é declarado
CONST_VALOR	-	Só estrutura
DEF_TIPOS	-	Só estrutura
LISTA_TIPOS	-	Só estrutura
LISTA_TIPOS'	-	Só estrutura
TIPO	1	Ponto onde um ID é declarado
TIPO_DADO	-	Só estrutura
DEF_VAR	-	Só estrutura
LISTA_VAR	-	Só estrutura
LISTA_VAR'	-	Só estrutura

VARIAVEL	1	Ponto onde uma lista de IDs são declarados
LISTA_ID	-	Só estrutura
LISTA_ID'	-	Só estrutura
NUMERO	-	É tratado como um token
LISTA_FUNC	-	Só estrutura
FUNCAO	-	Só estrutura
NOME_FUNCAO	1	Ponto onde um ID é declarado
BLOCO_FUNCAO	-	Só estrutura
BLOCO	-	Só estrutura
LISTA_COM	-	Só estrutura

COMANDO	2,3,7	2 : Quando usar um ID, a regra deve ser verificada
		3: Na atribuição é verificado se o tipo da variável é igual ao valor que ela recebe
		7: Na atribuição é verificado se ,dentro da função , o tipo do retorno é igual ao tipo do valor retornado
ELSE	-	Só estrutura
VALOR	2,5,6	2: Quando usar um ID, a regra deve ser verificada

		5: Aqui é onde pode ocorrer a chamada de função
		6: Aqui é onde pode ocorrer a chamada de função

VALOR'	4	Após for identificado o uso de parênteses no VALOR', a regra deve ser verificada
LISTA_NOME	2	Quando usar um ID, a regra deve ser verificada
LISTA_NOME'	-	Só estrutura

EXP_LOGICA	3	Nesse ponto (tendo em vista que o processamento da árvore é em pós- ordem) deve-se verificar a compatibilida de dos valores comparados
EXP_LOGICA'	-	Só estrutura
EXP_MAT	2,3	2: Quando usar um ID, a regra deve ser verificada
		3: Nesse ponto (tendo em vista que o

		processamento da árvore é em pós ordem) deve-se verificar a compatibilidade dos valores comparados
EXP_MAT'	-	Só estrutura
OP_LOGICO	-	Só estrutura
OP_MAT	-	Só estrutura
NOME	2,8,9,10	2 : Quando usar um ID como índice ( ex: vetor[i]) deve se verificar essa regra
		8: Quando identificar o terminal '[', a regra deve ser



		validada
		9: Semelhante a regra 8, quando identificar o terminal '.', a regra deve ser validada
		10: É um complemento da

		regra 9, então a validação ocorre logo após a 9 ser validada
ID	-	É tratado como um token

OBS: sobre a regra 7, foi considerando a forma de retorno do Pascal (para retornar

um valor de uma função nessa linguagem, é atribuído um valor dentro de uma variável do mesmo nome da função) ex:

```
function soma(a,b : integer) : integer;
```

```
begin
```

```
    soma := a + b; (aqui é realizado o retorno)
```

```
end
```

OBS2: sobre a regra 9, consideramos **record** equivalente ao tipo classe

## 2 – Conjunto de instruções de código intermediário:

Grupo	Instrução	Endereço 1	Endereço 2	Endereço 3	Descrição
Transferência de informação	atrib	Variável	Parâmetro	-	<b>Atribuí</b> o valor do parâmetro na variável
Operações aritméticas	add	Variável	Parâmetro 1	Parâmetro 2	<b>Soma</b> os dois parâmetros e coloca o resultado na variável
	sub	Variável	Parâmetro 1	Parâmetro 2	<b>Subtrai</b> os dois parâmetros e coloca o resultado na variável

	mul	Variável	Parâmetro 1	Parâmetro 2	<b>Multiplica</b> os dois parâmetros e coloca o resultado na variável
	div	Variável	Parâmetro 1	Parâmetro 2	<b>Divide</b> os dois parâmetros e coloca o resultado na variável
<b>Operações lógicas</b>	eql	Resultado	Parâmetro 1	Parâmetro 2	Compara o valor dos dois parâmetros e armazena ( 1 se forem <b>iguais</b> , 0 caso contrário) no resultado
	dif	Resultado	Parâmetro 1	Parâmetro 2	Compara o valor dos dois parâmetros e armazena ( 1 se forem <b>diferentes</b> , 0 caso contrário) no resultado
	grt	Resultado	Parâmetro 1	Parâmetro 2	Compara o valor dos dois parâmetros e armazena ( 1 se o primeiro for <b>maior</b> que o segundo, 0 caso contrário) no resultado

	les	Resultado	Parâmetro 1	Parâmetro 2	Compara o valor dos dois parâmetros e armazena ( 1 se o primeiro for <b>menor</b> que o segundo, 0 caso contrário) no resultado
<b>Instruções de salto</b>	lbl	label	-	-	<b>Marca a linha do código</b> onde pode ocorrer um salto com o nome presente no label

	jmp	label	-	-	<b>Salta</b> para linha do código onde está o label
	jz	label	Parâmetro	-	<b>Salta</b> para linha do código onde está o label se o valor no parâmetro for 0
	jnz	label	Parâmetro	-	<b>Salta</b> para linha do código onde está o label se o valor no parâmetro <b>não</b> for 0