Universidade Federal de Santa Catarina
Departamento de Computação

Disciplina: Construção de Compiladores
Docente: Marlon de Matos de Oliveira
Discentes: Gabriel Estevam - 15104138
Vinicius Zanon - 15102833

Analisador Semântico

1 Análise Semântica

A análise semântica refere-se à terceira etapa de análise da construção de um compilador. As tarefas básicas desempenhadas durante a análise semântica incluem a verificação de tipos, a verificação do fluxo de controle e a verificação da unicidade da declaração de variáveis. Dessa forma, é analisado o contexto formado pelos termos léxicos e sintáticos e determinado se a instrução possui algum significado na gramática, ou seja, se as ações/regras semânticas estão coerentes e corretas segundo a gramática da linguagem. Caso o analisador encontre um instância não satisfatória é reportado um erro semântico.

2 Ações Semânticas

Para a implementação do analisador semântico é necessário a construção de uma tabela de símbolos que gerencia as ações semânticas encontradas no decorrer da gramática. A tabela de símbolos possui a seguinte estrutura:

Nível	Categoria	Nome	Tipo

- ➔ O atributo **Nível** representa o contexto ou escopo em que a entrada da tabela está no programa.
- ➔ O atributo **Categoria** caracteriza entrada da tabela como variável, função ou parâmetro.
- ➔ O atributo **Nome** identifica a entrada da tabela.
- ➔ O atributo **Tipo** caracteriza a entrada da tabela em integer, float, string ou char, quando for aplicável.

Para a manipulação da tabela de símbolos, existem três principais operações: busca, inserção e remoção.

Busca: A operação de busca verifica a existência de um identificador na tabela de símbolos. Caso afirmativo efetua as ações pertinentes. Caso contrário, indica a condição de erro: “identificador não declarado”.
Inserção: A operação de inserção verifica a existência de um identificador na tabela de símbolos. Caso afirmativo, indica condição de erro: “identificador já declarado”, caso negativo insere na tabela.
Remoção: A operação de remoção retira o identificador da tabela de símbolos, verificando o nível em que está retirando.

Para entradas da tabela com a categoria igual a função, é atribuído também um lista de tipos. Esta lista de tipos representa os parâmetros da função. Ainda, para auxiliar a verificação dos parâmetros das funções existem variáveis chamadas *current_function* e *count_param*.

Além disso, uma pilha chamada *stack_level* auxilia na identificação do nível atual durante a análise. E uma variável chamada *type* auxilia na identificação dos tipos em uma expressão e retornos de funções.

Com todos os recursos projetados para o analisador semântico é possível definir as ações semânticas que foram implementadas nesta etapa. Ações semânticas podem ser associadas às regras de produção da gramática de modo que, quando uma dada produção é processada, essas ações sejam executadas.

Abaixo, será listado as ações semânticas que foram implementadas no analisador semântico.

1 Tabela das Ações Semânticas

ID	Ações Semânticas
1	***BLOCO**:* insere *main* em stack_level.
2	***nomevariavel***: insere entrada na tabela com nível igual ao topo de stack_level, nome igual a *nomevariavel* e categoria *variável*.
3	***nomevariavel***: consulta se existe uma entrada na tabela com nome igual a *nomevariavel*, categoria variável e nível igual ao topo de stack_level. Se sim, erro semântico: variável já declarada. Senão, insere entrada na tabela com nível igual a topo de stack_level, nome igual a *nomevariavel* e categoria *variável*.
4	***integer***: se a última entrada na tabela for de categoria função, insere *integer* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *integer* e categoria variável.
5	***float***: se a última entrada na tabela for de categoria função, insere *float* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *float* e categoria variável.
6	***string***: se a última entrada na tabela for de categoria função, insere *string* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *string* e categoria variável.

7	***char***: se a última entrada na tabela for de categoria função, insere *char* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *char* e categoria variável.
8	***nomevariavel***: preenche última entrada da tabela com nome igual a *nomevariavel*; e insere *nomevariavel* em stack_level. Se houver outra entrada na tabela com mesmo nome igual, com categoria função, mesmo tipo e mesmo nível, erro semântico: função já declarada.
9	***}***: remove topo de stack_level.
10	***integer***: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *integer*.
11	***void***: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *void*.
12	***char***: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *char*.
13	***float***: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *float*.
14	***string***: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *string*.
15	***numerointeiro***: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a inteiro. Caso contrário, erro semântico: tipo de retorno incoerente.
16	***numerofloat***: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a float. Caso contrário, erro semântico: tipo de retorno incoerente.
17	***nomedavariavel***: consulta se *nomedavariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. E, consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual ao tipo da entrada na tabela de *nomedavariavel* com categoria variável e nível igual ao topo de stack_level. Caso contrário, erro semântico: tipo de retorno incoerente.
18	***nomedochar***: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a char. Caso contrário, erro semântico: tipo de retorno incoerente.
19	***nomedastring***: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a float. Caso contrário, erro semântico: tipo de retorno incoerente.
20	***VALORRETORNO***: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a void. Caso contrário, erro semântico: tipo de retorno incoerente.
21	***nomevariavel***: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Atribui a type o tipo da entrada de *nomevariavel* com nível igual ao topo de stack_level.
22	***nomedastring***: atribui string a type.
23	***nomedochar***: atribui char a type.
24	***COMANDO***: atribui void a type.
25	***nomevariavel****:* consulta se *nomevariavel* está na tabela, se é da categoria função e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: função não definida. Consulta se o tipo de *nomevariavel* com categoria função e nível igual ao topo de stack_level é void. Atribui o índice da função com nome igual *nomevariável* e nível igual a topo de stack_level a current_function.
26	***nomevariavel***: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.

27	***nomevariavel****:* consulta se *nomevariavel* está na tabela, se é da categoria função e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: função não definida. Consulta se entrada na tabela com nome *nomevariavel*, com categoria função e nível igual ao topo de stack_level possui tipo igual a type. Caso contrário, erro semântico: tipo de retorno incoerente. Atribui o índice da função com nome igual *nomevariável* e nível igual a topo de stack_level a current_function.
28	***numerointeiro***: consulta se type é igual a inteiro. Caso contrário, erro semântico: tipo incoerente com expressão.
29	***numerofloat***: consulta se type é igual a float. Caso contrário, erro semântico: tipo incoerente com expressão.
30	***nomevariavel***: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Consulta se entrada na tabela com nome *nomevariavel* com categoria variavel e nível igual ao topo de stack_level possui tipo igual a type. Caso contrário, erro semântico: tipo incoerente com expressão.
31	***nomedastring***: consulta se type é igual a string. Caso contrário, erro semântico: tipo incoerente com expressão.
32	***nomedochar***: consulta se type é igual a char. Caso contrário, erro semântico: tipo incoerente com expressão.
33	***nomevariavel***: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Atribui o tipo de *nomevariavel* a type.
34	***(*** : atribui zero para count_param.
35	***)*** : verifica se count_param acrescentado em um é igual ao tamanho da lista de parâmetros de current_function. Caso contrário, erro semântico: número de parâmetros inconsistente.
36	***,*** : incrementa em um o count_param.
37	***PARAMETROS*** : verifica se o tamanho da lista de parâmetros de current_function é zero. Caso contrário, erro semântico: número de parâmetros inconsistente.
38	***numerointeiro*** : verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *integer* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
39	***nomedastring*** : verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *string* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
40	***numerofloat*** : verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *float* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
41	***nomedochar*** : verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *char* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
42	***nomevariavel*** : consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Verifica se o

	tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui o mesmo tipo de *nomevariavel* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.

3 Gramática Modificada

A cada regra sintática pode ser associado um conjunto de ações semânticas. Mais especificamente, cada token de uma regra sintática, seja da cabeça ou produção, pode ter ou não ações semânticas. As ações semânticas são aplicadas quando os tokens são removidos da pilha de análise sintática.

Para aplicação das ações semânticas podem ser utilizados os atributos do token que está sendo processado, a tabela de símbolos, *current_function*, *count_param*, *stack_level* e *type*.

Na tabela abaixo é mostrado a gramática modificada, com as respectivas regras semânticas implementadas.

2 Aplicação das Ações Semânticas na Gramática Modificada

ID	Regra Sintática	Ação Semântica
1	BLOCO::= void main { DCLVAR DCLFUNC CORPO }	(1) *BLOCO:* insere *main* em stack_level.
2	DCLVAR::= nomevariavel REPIDENT : TIPO ; LDVAR	(2) *nomevariavel*: insere entrada na tabela com nível igual ao topo de stack_level, nome igual a *nomevariavel* e categoria *variável*.
3	DCLVAR::= î	
4	REPIDENT::= î	
5	REPIDENT::= , nomevariavel REPIDENT	(3) *nomevariavel*: consulta se existe uma entrada na tabela com nome igual a *nomevariavel*, categoria variável e nível igual ao topo de stack_level. Se sim, erro semântico: variável já declarada. Senão, insere entrada na tabela com nível igual a topo de stack_level, nome igual a *nomevariavel* e categoria *variável*.
6	TIPO::= integer	(4) *integer*: se a última entrada na tabela for de categoria função, insere *integer* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *integer* e categoria variável.
7	TIPO::= float	(5) *float*: se a última entrada na tabela for de categoria função, insere *float* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com

		tipo vazio em sequência com tipo *float* e categoria variável.
8	TIPO::= string	(6) *string*: se a última entrada na tabela for de categoria função, insere *string* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *string* e categoria variável.
9	TIPO::= char	(7) *char*: se a última entrada na tabela for de categoria função, insere *char* na lista de parâmetros da última entrada da tabela. Caso contrário, preenche todas as últimas entradas da tabela com tipo vazio em sequência com tipo *char* e categoria variável.
10	LDVAR::= î	
11	LDVAR::= LID : TIPO ; LDVAR	
12	LID::= nomevariavel REPIDENT	(3) *nomevariavel*: consulta se existe uma entrada na tabela com nome igual a *nomevariavel*, categoria variável e nível igual ao topo de stack_level. Se sim, erro semântico: variável já declarada. Senão, insere entrada na tabela com nível igual a topo de stack_level, nome igual a *nomevariavel* e categoria *variável*.
13	DCLFUNC::= TIPO_RETORNO nomevariavel DEFPAR{ DCLVAR DCLFUNC CORPO return (VALORRETORNO)} DCLFUNC	(8) *nomevariavel*: preenche última entrada da tabela com nome igual a *nomevariavel*; e insere *nomevariavel* em stack_level. Se houver outra entrada na tabela com mesmo nome igual, com categoria função, mesmo tipo e mesmo nível, erro semântico: função já declarada. (9) *}*: remove topo de stack_level.
14	TIPO_RETORNO::= integer	(10) *integer*: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *integer*.
15	TIPO_RETORNO::= void	(11) *void*: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *void*.
16	TIPO_RETORNO::= char	(12) *char*: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *char*.
17	TIPO_RETORNO::= float	(13) *float*: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *float*.
18	TIPO_RETORNO::= string	(14) *string*: insere entrada na tabela com nível igual a topo de stack_level, categoria *função* e tipo *string*.
19	DCLFUNC::= î	
20	VALORRETORNO::= numerointeiro	(15) *numerointeiro*: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a inteiro. Caso contrário, erro

		semântico: tipo de retorno incoerente.
21	VALORRETORNO::= numerofloat	(16) *numerofloat*: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a float. Caso contrário, erro semântico: tipo de retorno incoerente.
22	VALORRETORNO::= nomedavariavel	(17) *nomedavariavel*: consulta se *nomedavariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. E, consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual ao tipo da entrada na tabela de *nomedavariavel* com categoria variável e nível igual ao topo de stack_level. Caso contrário, erro semântico: tipo de retorno incoerente.
23	VALORRETORNO::= nomedochar	(18) *nomedochar*: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a char. Caso contrário, erro semântico: tipo de retorno incoerente.
24	VALORRETORNO::= nomedastring	(19) *nomedastring*: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a float. Caso contrário, erro semântico: tipo de retorno incoerente.
25	VALORRETORNO::= î	(20) *VALORRETORNO*: consulta se a entrada com nome igual ao topo de stack_level e categoria função possui tipo igual a void. Caso contrário, erro semântico: tipo de retorno incoerente.
26	DEFPAR::= î	
27	DEFPAR::= (PARAM)	
28	PARAM::= TIPO LPARAM	
29	LPARAM::= ; TIPO LPARAM	
30	LPARAM::= î	
31	CORPO::= inicio COMANDO ; REPCOMANDO fim	
32	REPCOMANDO ::= î	
33	REPCOMANDO::= COMANDO ; REPCOMANDO	
34	COMANDO::= nomevariavel = EXPRESSAO	(21) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Atribui a type o tipo da entrada de *nomevariavel* com nível igual ao topo de stack_level.
35	COMANDO::= nomedastring = EXPRESSAO	(22) *nomedastring*: atribui string a type.
36	COMANDO::= nomedochar = EXPRESSAO	(23) *nomedochar*: atribui char a type.

37	*Regra sintática removida.*	
38	COMANDO::= î	
39	COMANDO::= callfuncao nomevariavel PARAMETROS	(24) *COMANDO*: atribui void a type. (25) *nomevariavel:* consulta se *nomevariavel* está na tabela, se é da categoria função e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: função não definida. Consulta se o tipo de *nomevariavel* com categoria função e nível igual ao topo de stack_level é void. Atribui o índice da função com nome igual *nomevariavel* e nível igual a topo de stack_level a current_function.
40	PARAMETROS::= î	(37) *PARAMETROS*: verifica se o tamanho da lista de parâmetros de current_function é zero. Caso contrário, erro semântico: número de parâmetros inconsistente.
41	PARAMETROS::= (TPARAM REPPAR)	(34) *(* : atribui zero para count_param. (35) *)* : verifica se count_param acrescentado em um é igual ao tamanho da lista de parâmetros de current_function. Caso contrário, erro semântico: número de parâmetros inconsistente.
42	REPPAR::= î	
43	REPPAR::= , TPARAM REPPAR	(36) *,* : incrementa em um o count_param.
44	TPARAM::= numerointeiro	(38) *numerointeiro*: verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *integer* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
45	TPARAM::= nomedastring	(39) *nomedastring*: verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *string* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
46	TPARAM::= numerofloat	(40) *numerofloat*: verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui *float* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
47	TPARAM::= nomedochar	(41) *nomedochar*: verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a

		lista de parâmetros de current_function possui *char* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
48	TPARAM::= nomevariavel	(42) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Verifica se o tamanho da lista de parâmetros de current_function é válida conforme count_param. Caso contrário, erro semântico: número de parâmetros inconsistente. Verifica se a lista de parâmetros de current_function possui o mesmo tipo de *nomevariavel* na posição count_param. Caso contrário, erro semântico: tipo do parâmetro inconsistente.
49	COMANDO::= if (nomevariavel COMPARACAO) { COMANDO ; REPCOMANDO } ELSEPARTE	(33) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Atribui o tipo de *nomevariavel* a type.
50	ELSEPARTE::= else { COMANDO ; REPCOMANDO }	
51	ELSEPARTE::= î	
52	COMANDO::= while (nomevariavel COMPARACAO) { COMANDO ; REPCOMANDO }	(33) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Atribui o tipo de *nomevariavel* a type.
53	COMPARACAO::= == CONTCOMPARACAO	
54	COMPARACAO::= != CONTCOMPARACAO	
55	COMPARACAO::= > CONTCOMPARACAO	
56	COMPARACAO::= >= CONTCOMPARACAO	
57	COMPARACAO::= < CONTCOMPARACAO	
58	COMPARACAO::= <= CONTCOMPARACAO	
59	CONTCOMPARACAO::= numerointeiro	(28) *numerointeiro*: consulta se type é igual a inteiro. Caso contrário, erro semântico: tipo incoerente com expressão.
60	CONTCOMPARACAO::= numerofloat	(29) *numerofloat*: consulta se type é igual a float. Caso contrário, erro semântico: tipo incoerente com expressão.
61	CONTCOMPARACAO::= nomedastring	(31) *nomedastring*: consulta se type é igual a string. Caso contrário, erro semântico: tipo incoerente com expressão.
62	CONTCOMPARACAO::= nomedochar	(32) *nomedochar*: consulta se type é igual a char. Caso contrário, erro semântico: tipo incoerente com

		expressão.
63	CONTCOMPARACAO::= nomevariavel	(30) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Consulta se entrada na tabela com nome *nomevariavel* com categoria variavel e nível igual ao topo de stack_level possui tipo igual a type. Caso contrário, erro semântico: tipo incoerente com expressão.
64	COMANDO::= for (nomevariavel = CONTCOMPARACAO; nomevariavel COMPARACAO; INCREMENTO) { COMANDO ; REPCOMANDO }	(33) *nomevariavel* (1): consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. E atribui o tipo de *nomevariavel* a type. (33) *nomevariavel* (2): consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.E atribui o tipo de *nomevariavel* a type.
65	INCREMENTO::= ++ numerointeiro	
66	INCREMENTO::= -- numerointeiro	
67	COMANDO::= do { COMANDO ; REPCOMANDO } while (nomevariavel COMPARACAO)	(33) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.E atribui o tipo de *nomevariavel* a type.
68	COMANDO::= cin >> nomevariavel	(26) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.
69	COMANDO::= cout << literal SEQCOUT	
70	SEQCOUT::= î	
71	X::= nomevariavel SEQUENCIA SEQCOUT	(26) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.
72	X::= literal SEQCOUT	
73	SEQUENCIA::= î	
74	SEQUENCIA::= , nomevariavel SEQUENCIA	(26) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida.
75	EXPRESSAO::= TERMO REPEXP	
76	EXPRESSAO::= callfuncao nomevariavel	(27) *nomevariavel:* consulta se *nomevariavel* está

	PARAMETROS	na tabela, se é da categoria função e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: função não definida. Consulta se entrada na tabela com nome *nomevariavel*, com categoria função e nível igual ao topo de stack_level possui tipo igual a type. Caso contrário, erro semântico: tipo de retorno incoerente. Atribui o índice da função com nome igual *nomevariável* e nível igual a topo de stack_level a current_function.
77	REPEXP::= + TERMO REPEXP	
78	REPEXP::= - TERMO REPEXP	
79	REPEXP::= î	
80	TERMO::= FATOR REPTERMO	
81	REPTERMO::= î	
82	REPTERMO::= * FATOR REPTERMO	
83	REPTERMO::= / FATOR REPTERMO	
84	FATOR::= numerointeiro	(28) *numerointeiro*: consulta se type é igual a inteiro. Caso contrário, erro semântico: tipo incoerente com expressão.
85	FATOR::= numerofloat	(29) *numerofloat*: consulta se type é igual a float. Caso contrário, erro semântico: tipo incoerente com expressão.
86	FATOR::= nomevariavel	(30) *nomevariavel*: consulta se *nomevariavel* está na tabela, se é da categoria variável e se possui nível igual ao topo de stack_level. Caso contrário, erro semântico: variável não definida. Consulta se entrada na tabela com nome *nomevariavel* com categoria variavel e nível igual ao topo de stack_level possui tipo igual a type. Caso contrário, erro semântico: tipo incoerente com expressão.
87	FATOR::= nomedastring	(31) *nomestring*: consulta se type é igual a string. Caso contrário, erro semântico: tipo incoerente com expressão.
88	FATOR::= nomedochar	(32) *nomedochar*: consulta se type é igual a char. Caso contrário, erro semântico: tipo incoerente com expressão.
89	FATOR::= (EXPRESSAO)	
90	SEQCOUT::= << X	

5 Erros Semânticos

Os erros semânticos podem ser disparados em casos onde há incoerência das ações/regras gramaticais. No entanto, pelas ações semânticas é possível inferir se há ou não erros presentes nesta classe de análise. Neste trabalho foi possível inferir os seguintes erros semânticos:

- ➔ Erro Semântico: Variável já declarada: Esse erro indica a unicidade de declaração de variáveis.
- ➔ Erro Semântico: Função já declarada: Esse erro indica a unicidade de declaração de funções.
- ➔ Erro Semântico: Variável não definida: Esse erro indica a ausência de declaração de variável quando há a solicitação de seu uso.
- ➔ Erro Semântico: Função não definida: Esse erro indica a ausência de declaração de função quando há a solicitação de seu uso.
- ➔ Erro Semântico: Tipo de retorno incoerente: Esse erro indica a incompatibilidade do tipo de retorno usado no corpo da função.
- ➔ Erro Semântico: Tipo incoerente com expressão: Esse erro indica a incompatibilidade do tipo entre as variáveis presentes em uma expressão, seja ela relacional, condicional ou atribuição.
- ➔ Erro Semântico: Tipo do parâmetro inconsistente: Esse erro indica que o tipo do parâmetro na declaração de função é diferente do tipo do parâmetro na chamada de função.
- ➔ Erro Semântico: Número de parâmetros inconsistente: Esse erro indica que o número de parâmetros na declaração de função é diferente no número de parâmetros na chamada de função.