

**UNIVERSIDADE DO VALE DO ITAJAÍ - UNIVALI  
ESCOLA DO MAR CIÊNCIA E TECNOLOGIA  
CIÊNCIA DA COMPUTAÇÃO**

**DOCUMENTAÇÃO DO TRABALHO FINAL  
COMPILADORES**

por

**Amanda Cristine Stecz Antunes  
Elmis Rosa  
João Felipe Barbosa Cerqueira Carvalho**

**Itajaí (SC), Junho de 2021**

**CAMINHO PARA O ARQUIVO JAR:**

`\COMPILADOR_01\out\ARTIFACTS\COMPILADOR_0_1_JAR`

**RELAÇÃO DE TOKENS**

**EOF(0) → UTILIZADO PARA INDICAR FIM DE LINHA**

**PROGRAM(4)** → UTILIZADO PARA INDICAR A INICIALIZAÇÃO DO PROGRAMA  
**DEFINE(5)** → PRECEDE A DEFINIÇÃO DE VARIÁVEIS OU CONSTANTES  
**NOT(6)** → NEGAÇÃO QUE PRECEDE PALAVRA RESERVADA, UTILIZADO PARA INDICAR CONSTANTES  
**VARIABLE(7)** → UTILIZADO PARA DECLARAÇÃO DE VARIÁVEIS OU CONSTANTES (QUANDO PRECEDIDO DE NEGAÇÃO)  
**IS(8)** → UTILIZADO PARA DECLARAR UM ESTADO BOOLEANO  
**BOOLEAN(9)** → UTILIZADO PARA DEFINIR UM ESTADO BOOLEANO  
**CHAR(10)** → UTILIZADO PARA DEFINIR TIPO CHAR  
**REAL(11)** → UTILIZADO PARA DEFINIR TIPO REAL  
**NATURAL(12)** → UTILIZADO PARA DEFINIR TIPO NATURAL  
**EXECUTE(13)** → UTILIZADO PARA INDICAR A INICIALIZAÇÃO DO CORPO DO PROGRAMA  
**SET(14)** → UTILIZADO PARA COMANDO DE ATRIBUIÇÃO  
**GET(15)** → UTILIZADO PARA COMANDO DE ENTRADA DE DADOS  
**PUT(16)** → UTILIZADO PARA COMANDO DE SAÍDA DE DADOS  
**LOOP(17)** → UTILIZADO PARA COMANDO DE LAÇOS DE REPETIÇÃO  
**WHILE(18)** → UTILIZADO PARA COMANDO DE LAÇOS DE REPETIÇÃO  
**TRUE(19)** → UTILIZADO PARA DEFINIR ESTADO VERDADEIRO  
**FALSE(20)** → UTILIZADO PARA DEFINIR ESTADO FALSO  
**DO(21)** → UTILIZADO PARA DEFINIR OS COMANDOS DE LAÇOS DE REPETIÇÃO  
**TO(22)** → UTILIZADO PARA CRITÉRIO DE PARADA DE LAÇOS DE REPETIÇÃO  
**VERIFY(23)** → UTILIZADO PARA COMANDO DE SELEÇÃO  
**IDENTIFICADOR(24)** → UTILIZADO PARA UM IDENTIFICADOR  
**CONSTANTE\_NUM\_REAL(25)** → UTILIZADO PARA UM IDENTIFICADOR TIPO REAL  
**CONSTANTE\_NUM\_INT(26)** → UTILIZADO PARA UM IDENTIFICADOR NUMÉRICO INTEIRO  
**CONSTANTE\_LIT(27)** → UTILIZADO PARA UM IDENTIFICADOR LITERAL  
**ASCII(28)** → TABELA ASCII  
**LETTER(29)** → LETRAS DO ALFABETO  
**APARENT(30)** → ABRE PARENTESSES  
**FPARENT(31)** → FECHA PARENTESSES  
**ACHAVE(32)** → ABRE CHAVES  
**FCHAVE(33)** → FECHA CHAVES  
**ACOLCH(34)** → ABRE COLCHETES  
**FCOLCH(35)** → FECHA COLCHETES  
**VIRGULA(36)** → VIRGULA  
**PONTO(37)** → PONTO  
**COMENT(38)** → COMENTÁRIOS  
**ADICAO(39)** → OPERADOR ARITMÉTICO “+”  
**SUBTRACAO(40)** → OPERADOR ARITMÉTICO “-”  
**MULTIPLICACAO(41)** → OPERADOR ARITMÉTICO “\*”

**DIVISAO(42)** → OPERADOR ARITMÉTICO “/”  
**POTENCIA(43)** → OPERADOR ARITMÉTICO “\*\*”  
**DIVISAOINTEIRA(44)** → OPERADOR ARITMÉTICO “%”  
**RESTODIVISAO(45)** → OPERADOR ARITMÉTICO “%%”  
**IGUAL(46)** → OPERADOR ARITMÉTICO “=”  
**EQUIVALENTE(47)** → OPERADOR RELACIONAL “==”  
**DIFERENTE(48)** → OPERADOR RELACIONAL “!=”  
**MENOR(49)** → OPERADOR RELACIONAL “<”  
**MENOROUIGUAL(50)** → OPERADOR RELACIONAL “<=”  
**MAIOR(51)** → OPERADOR RELACIONAL “>”  
**MAIOROUIGUAL(52)** → OPERADOR RELACIONAL “>=”  
**NAO(53)** → OPERADOR LÓGICO “!”  
**OU(54)** → OPERADOR LÓGICO “|”  
**E(55)** → OPERADOR LÓGICO “&”

## **ERROS LÉXICOS:**

**ERRO(00)** -->SÍMBOLO INVÁLIDO

**ERRO(01)** -->STRING NÃO FINALIZADA  
**ERRO(02)**--> IDENTIFICADOR INVÁLIDO

### **ERROS SINTÁTICOS:**

**ERRO (03)**---> PROGRAMA FINALIZADO INCORRETAMENTE.  
**ERRO (04)**---> PROGRAMA OU FUNÇÃO NÃO FOI INICIADA CORRETAMENTE  
**ERRO (05)**---> PROGRAMA INICIALIZADO INCORRETAMENTE  
**ERRO (06)**---> CORPO DO PROGRAMA INICIALIZADO INCORRETAMENTE  
**ERRO (07)**---> EXPRESSÃO INVÁLIDA  
**ERRO (08)**---> TIPO INVÁLIDO  
**ERRO (09)**---> COMANDO INVÁLIDO  
**ERRO (10)**---> ERRO NA INICIALIZAÇÃO DE VARIÁVEIS  
**ERRO (11)**---> VETOR DECLARADO INCORRETAMENTE  
**ERRO (12)**---> COMANDO OU FUNÇÃO FINALIZADO INCORRETAMENTE  
**ERRO (13)**---> ERRO EM CORPO DO PROGRAMA

### **ESTRUTURA DE COMENTÁRIOS:**

**SINGLELINECOMENT** (1) → COMENTÁRIO DE UMA ÚNICA LINHA, **DEFINIDO POR** **“//”**  
**MULTILINECOMENT** (2) → COMENTÁRIO EM BLOCO, **DEFINIDO POR** **“/\* \*/”**

### **ERROS SEMÂNTICOS**

**ERRO (14)**--->TIPO INVÁLIDO PARA CONSTANTE EM DECLARAÇÃO DE CONSTANTE  
**ERRO (15)**--->IDENTIFICADOR JÁ DECLARADO.

**ERRO (16)---**>IDENTIFICADOR DE VARIÁVEL NÃO INDEXADA EM COMANDO DE ATRIBUIÇÃO

**ERRO (16B)---**>IDENTIFICADOR DE VARIÁVEL NÃO INDEXADA EM ENTRADA DE DADOS.

**ERRO (17)---**> IDENTIFICADOR DE VARIÁVEL INDEXADA EXIGE ÍNDICE EM COMANDO DE ATRIBUIÇÃO.

**ERRO (18)---**> IDENTIFICADOR NÃO DECLARADO OU DE CONSTANTE EM ENTRADA DE DADOS.

**ERRO (19)---**> IDENTIFICADOR NÃO DECLARADO EM COMANDO DE SAÍDA OU EXPRESSÃO .

**ERRO (20)---**> IDENTIFICADOR DE CONSTANTE OU DE VARIÁVEL NÃO INDEXADA EM COMANDO DE SAÍDA.

## ERROS DE EXECUÇÃO DA MÁQUINA VIRTUAL

**RUNTIME ERROR (1):** ESSA INSTRUÇÃO É VÁLIDA APENAS PARA VALORES INTEIROS OU REAIS.

**RUNTIME ERROR (2):** NÃO FOI POSSÍVEL EXECUTAR A INSTRUÇÃO, TIPO NULL ENCONTRADO.

**RUNTIME ERROR (3):** NÃO FOI POSSÍVEL EXECUTAR A INSTRUÇÃO, VALOR NULL ENCONTRADO.

**RUNTIME ERROR (4):** ESSA INSTRUÇÃO É VÁLIDA APENAS PARA VALORES LÓGICOS.

**RUNTIME ERROR (5):** IMPOSSÍVEL REALIZAR A OPERAÇÃO. O VALOR DO DIVIDENDO DEVE SER MAIOR QUE ZERO.

**RUNTIME ERROR (6):** CONSTATANTE NÃO IDENTIFICADA. ESPERADO VALOR LÓGICO.

**RUNTIME ERROR (7):** CONSTATANTE NÃO IDENTIFICADA. ESPERADO VALOR INTEIRO.

**RUNTIME ERROR (8):** CONSTATANTE NÃO IDENTIFICADA. ESPERADO VALOR REAL.

**RUNTIME ERROR (9):** CONSTATANTE NÃO IDENTIFICADA. ESPERADO VALOR LITERAL.

**RUNTIME ERROR (10):** TIPO INCOMPATÍVEL. ESPERADO TIPO INTEIRO.

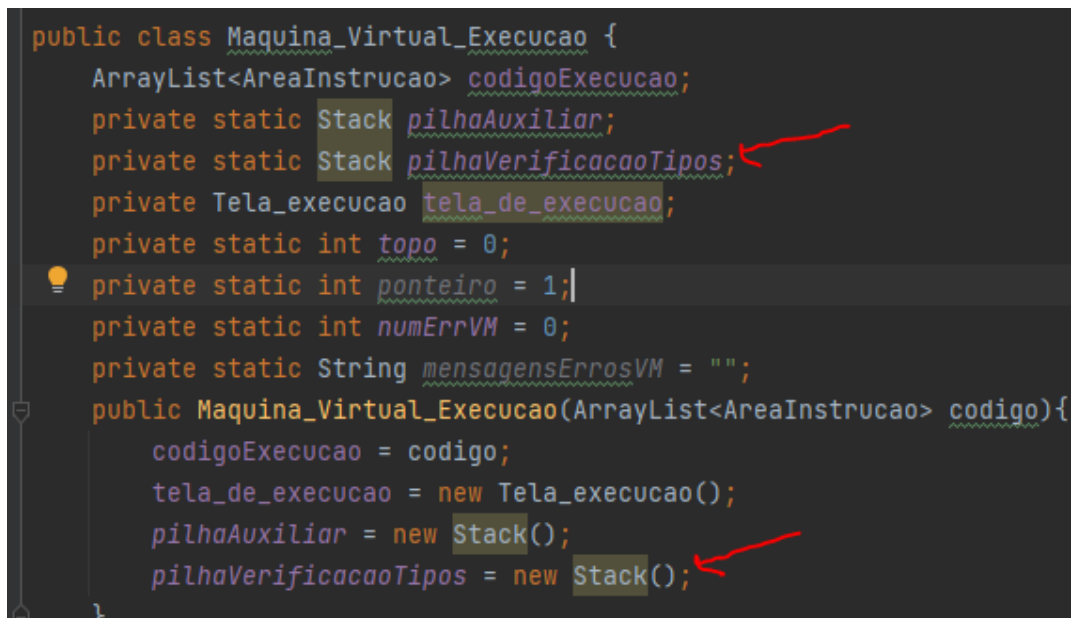
**RUNTIME ERROR (11):** TIPOS INCOMPATÍVEL. ESPERADO TIPO REAL.

**RUNTIME ERROR (12):** TIPOS INCOMPATÍVEL. ESPERADO TIPO LITERAL.

**RUNTIME ERROR (13):** TIPOS INCOMPATÍVEL. ESPERADO TIPO LÓGICO.

## DESCRIÇÃO DAS INSTRUÇÕES DA MÁQUINA VIRTUAL ESPECIFICADAS OU ALTERADAS

PARA REALIZAÇÃO DO CONTROLE DE TIPO ALOCADO NA PILHA DE EXECUÇÃO, FOI CRIADA UMA PILHA DE VERIFICAÇÃO DE TIPOS - FIGURA 1 - PARA AUXILIAR NO CONTROLE DE TIPOS.



```
public class Maquina_Virtual_Execucao {
    ArrayList<AreaInstrucao> codigoExecucao;
    private static Stack pilhaAuxiliar;
    private static Stack pilhaVerificacaoTipos;
    private Tela_execucao tela_de_execucao;
    private static int topo = 0;
    private static int ponteiro = 1;
    private static int numErrVM = 0;
    private static String mensagensErrosVM = "";
    public Maquina_Virtual_Execucao(ArrayList<AreaInstrucao> codigo){
        codigoExecucao = codigo;
        tela_de_execucao = new Tela_execucao();
        pilhaAuxiliar = new Stack();
        pilhaVerificacaoTipos = new Stack();
    }
}
```

FIGURA 1 - INSTÂNCIA DA PILHA DE VERIFICAÇÃO DE TIPOS PARA CONTROLE SINALIZADA COM UMA SETA

AO OCORRER UM EMPILHAMENTO NA PILHA PRINCIPAL (CHAMADA DE PILHA AUXILIAR), A PILHA DE VERIFICAÇÃO DE TIPOS ERA ATUALIZADA COM O VALOR CORRESPONDENTE AO VALOR EMPILHADO

NA PILHA PRINCIPAL, CONFORME DEMONSTRADO NA FIGURA 2, ONDE A PILHA AUXILIAR ESTÁ EMPILHANDO UM VALOR DO TIPO REAL E A PILHA DE VERIFICAÇÃO DE TIPOS ESTÁ RECEBENDO UMA STRING RELACIONADA AO TIPO DO VALOR, NO CASO, A PALAVRA “REAL”. E CASO OCORRESSE UM DESEMPILHAMENTO DE ALGUM VALOR DA PILHA PRINCIPAL, TAMBÉM OCORRIA UMA ATUALIZAÇÃO NA PILHA DE VERIFICAÇÃO DE TIPOS, DESEMPILHANDO O VALOR COM O TIPO CORRESPONDENTE.

```
    pilhaAuxiliar.push(instrucao.fParametro);  
    pilhaVerificacaoTipos.push("real");
```

FIGURA 2 - ATUALIZAÇÃO SINCRONIZADA ENTRE A PILHA PRINCIPAL E A PILHA DE VERIFICAÇÃO DE TIPOS

**i) INSTRUÇÕES DE ATRIBUIÇÃO ALTERADAS: “LDS”, “LDI”, “LDB”, “LDR”,** CADA INSTRUÇÃO DE ATRIBUIÇÃO FORAM ALTERADAS PARA SUPORTAR SOMENTE OS VALORES PARA AS QUAIS SÃO DESIGNADAS ATRAVÉS DE OPERAÇÕES DE IF E ELSE, USAMOS A PILA DE VERIFICAÇÃO DE TIPOS PARA GARANTIR QUE AS INSTRUÇÕES RECEBAM AS CONSTANTES DOS TIPOS CORRETOS, LITERAL, INTEIRA, LÓGICA E REAL, RESPECTIVAMENTE.

**ii) OPERAÇÕES RELACIONAIS ALTERADAS:** TODAS AS INSTRUÇÕES RELACIONAIS, FORAM CONDICIONADAS PARA RETORNAR APENAS VALORES LÓGICOS, INTERNAMENTE FIZEMOS AS VERIFICAÇÕES COM OS VALORES INTEIROS OU REAIS, E DEPOIS USAMOS A PILHA DE VERIFICAÇÃO PARA RETORNAR VALORES LÓGICOS.

**iii) INSTRUÇÕES ARITMÉTICAS ALTERADAS: “ADD”, “DIV”, “MUL” E “SUB”. INSTRUÇÕES ARITMÉTICAS ESPECIFICADAS: “DIVINT”, “MOD” E “POT”:**

TODAS AS INSTRUÇÕES ARITMÉTICAS FORAM CONDICIONADAS PARA TRABALHAREM APENAS COM VALORES REAIS OU INTEIROS, RESPEITANDO A REGRA DE QUE UMA VARIÁVEL DO TIPO INTEIRO SÓ PODE RECEBER VALORES INTEIROS E UMA VARIÁVEL DO TIPO REAL SÓ PODE RECEBER VALORES INTEIROS OU REAIS. BEM COMO OPERAÇÕES QUE ENVOLVEM DIVIDENDOS DEVEM POSSUIR VALORES MAIORES QUE ZERO. ESSE CONTROLE FOI REALIZADO ATRAVÉS DA PILHA DE VERIFICAÇÃO DE TIPOS, COM O EMPILHAMENTO E DESEMPILHAMENTO DE VALORES. CASO ALGUMAS CONDIÇÕES NÃO SEJAM VERIFICADAS, A EXCEÇÃO NÚMERO 1, 2, 3 OU 5 PODEM SER DISPARADAS.

AS INSTRUÇÕES ARITMÉTICAS “DIVINT”, “MOD” E “POT”, CORRESPONDEM A DIVISÃO INTEIRA, RESTO DA DIVISÃO E POTENCIAÇÃO, RESPECTIVAMENTE. E FORAM NOVAS INSTRUÇÕES ESPECIFICADAS DA SEGUINTE FORMA:

**DIVINT** - RETIRA OS DOIS ÚLTIMOS VALORES DA MEMÓRIA; VERIFICA SE O TOPO É DIFERENTE DE ZERO (ASSIM COMO A INSTRUÇÃO “DIV”); DIVIDE O PENÚLTIMO VALOR PELO ÚLTIMO E COLOCA O RESULTADO NO TOPO DA MEMÓRIA. O RESULTADO FINAL É ARREDONDADO PARA UM NÚMERO INTEIRO CASO HAJA NÚMEROS REAIS ENVOLVIDOS NA OPERAÇÃO.

**MOD** - RETIRA OS DOIS ÚLTIMOS VALORES DA MEMÓRIA; EFETUA A OPERAÇÃO MOD QUE RETORNA O RESTO DA DIVISÃO E COLOCA O RESULTADO NO TOPO DA MEMÓRIA. VÁLIDO APENAS PARA VALORES REAIS OU INTEIROS.

**POT** - RETIRA OS DOIS ÚLTIMOS VALORES DA MEMÓRIA; EFETUA A OPERAÇÃO DE POTENCIAÇÃO E COLOCA O RESULTADO LÓGICO NO TOPO DA MEMÓRIA.

**IV) INSTRUÇÕES RELACIONAIS ALTERADAS: “BGE”, “BGR” E “DIF”, “EQL”, “SME” E “SMR”.**

AS INSTRUÇÕES RELACIONAIS FORAM ALTERADAS PARA QUE APENAS POSSAM SER EXECUTADAS CASO ESTEJAM ENVOLVENDO NÚMEROS REAIS E INTEIROS, DE FORMA QUE O RESULTADO DA AVALIAÇÃO SEMPRE IRÁ RESULTAR NO EMPILHAMENTO DE UM VALOR LÓGICO NA PILHA PRINCIPAL E NA PILHA DE VERIFICAÇÃO DE TIPOS. UMA VARIÁVEL DO TIPO INTEIRO SÓ PODE RECEBER VALORES INTEIROS, UMA VARIÁVEL DO TIPO REAL SÓ PODE RECEBER VALORES INTEIROS OU REAIS. CASO OS VALORES ENVOLVIDOS NÃO OBEDEÇAM A ESSA REGRA DE VERIFICAÇÃO, A EXCEÇÃO NÚMERO 1 OU DISPARADA NÚMERO 2 PODEM SER DISPARADAS.

**V) INSTRUÇÕES LÓGICAS ALTERADAS: “AND”, “NOT” E “OR”.**

AS INSTRUÇÕES LÓGICAS APENAS PODEM SER EXECUTADAS SE OS VALORES ARMAZENADOS NA PILHA FOREM DO TIPO “LÓGICO”. PARA O CONTROLE DE OPERADORES LÓGICOS POSSA OCORRER ATRAVÉS DA PILHA DE VERIFICAÇÃO DE TIPOS, SÃO DESEMPILHADOS O ÚLTIMO E O PENÚLTIMO VALOR (EXCETO PARA “NOT”, ONDE É DESEMPILHADO APENAS O ÚLTIMO VALOR, O TOPO) DA PILHA DE VERIFICAÇÃO DE TIPOS E, DESTA FORMA, OCORRE A VERIFICAÇÃO ATRAVÉS DE UMA INSTRUÇÃO CONDICIONAL. CASO OS TIPOS NÃO CORRESPONDAM AO TIPO “LÓGICO”, AS EXCEÇÕES NÚMERO 4 OU NÚMERO 2 PODEM SER DISPARADAS.