ESQUEMA DE TRADUÇÃO (nº8)

```
<programa>          ::= #15 main  constante.caracter
                              { <variáveis>  <lista comandos> } #16

<variáveis>         ::= ε  |  <tipo> #21  :  <lista id> #23 ;  <variáveis>
<tipo>              ::= int  |  real
<lista id>          ::= identificador #22  |  identificador #22  ,  <lista id>

<lista comandos>::= ε  |  <comando>  ;  <lista comandos>
<comando>           ::= <atribuição>  |  <entrada>  |  <saída>

<atribuição>        ::=  identificador #22  :=  <relacional> #26

<entrada>           ::=  read  (  <lista id> #24 )

<saída>             ::= write ( <lista expressão> )
<lista expressão>::= <lógica> #14                          |
                        <lógica> #14  , <lista expressão>

<lógica>            ::= true                  #11          |
                        false                 #12          |
                        not ( <lógica> )      #13          |
                        <relacional>

<relacional>::=       <expressão>  <operador> #9 <expressão> #10
                    |  <expressão>
<operador>::=         <  |  >  |  =

<expressão>::=        <termo>  <expressão_>
<expressão_>::=       + <termo>  #1  <expressão_>
                    |  -  <termo>  #2  <expressão_>
                    |  ε

<termo>::=            <elemento>  <termo_>
<termo_>::=           * <elemento>  #3  <termo_>
                    |  / <elemento>  #4  <termo_>
                    |  ε

<elemento>::=         constante.inteira  #5
                    |  constante.real     #6
                    |  + <elemento>       #7
                    |  - <elemento>       #8
                    |  ( <expressão> )
                    |  identificador      #25
```

registros semânticos:

```
TS          // tabela de símbolos, usada para armazenar os identificadores e respecitivos tipos, conforme
            // declaração de variáveis
tipovar     // tipo, usado para armazenar o tipo reconhecido na ação #21
listaid     // lista de identificadores, usada para armazenar os identificadores reconhecidos na ação #23
```

ações semânticas:

ação #1:

```
tipo1:= pilha.desempilha
tipo2:= pilha.desempilha
se (tipo1=float64) ou (tipo2=float64)
então pilha.empilha (float64)
senão pilha.empilha (int64)
fimse
código.adiciona (add)
```

ação #5:

```
pilha.empilha (int64)
código.adiciona (ldc.i8 token.getLexeme)
código.adiciona (conv.r8)
```

ação #21:

```
caso token.getlexeme
  int  tipovar:= int64
  real tipovar:= float64
fimcaso
```

ação #22:

```
listaid.adiciona(token.getlexeme)
```

ação #23:

```
para cada id in listaid faça
  se TS.tem (id)
  então erro semântico, parar
  fimse
  TS.adiciona(id, tipovar)
  código.adiciona (.locals(tipovar id))
fimpara
listaid.limpa
```

ação #25:

```
id:= token.getlexeme
se não TS.tem (id)
então erro semântico, parar
fimse
tipoid:= TS.recupera_tipo(id)
pilha.empilha (tipoid)
código.adiciona (ldloc id)
se (tipoid=int64)
então código.adiciona (conv.r8)
fimse
```

ação #26:

```
id:= listaid.retira
se não TS.tem (id)
então erro semântico, parar
fimse
tipoid:= TS.recupera_tipo(id)
tipoexp:= pilha.desempilha
se (tipoid≠tipoexp)
então erro semântico, parar
fimse
se (tipoid=int64)
então código.adiciona (conv.i8)
fimse
código.adiciona (stloc id)
```

ação #24:

```
para cada id in listaid faça
  se não TS.tem (id)
  então erro semântico, parar
  fimse
  tipoid:= TS.recupera_tipo(id)
  caso tipoid
    int64      classe:= Int64
    float64    classe:= Double
  fimcaso
  código.adiciona (call string [mscorlib]System.Console::ReadLine())
  código.adiciona (call tipoid [mscorlib]System.classe::Parse(string))
  código.adiciona (stloc id)
fimpara
listaid.limpa
```