UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
COMPILADORES

LABORATÓRIO 10

ANÁLISE SEMÂNTICA

1. Modifique o projeto AST (Abstract Syntax Tree) fornecido no Material de Apoio de forma que ele faça conversões automáticas entre valores inteiros e booleanos, convertendo `0` para `false` e qualquer outro valor para `true`.

   O projeto AST implementa a gramática a seguir:

```
program  →  type main() block        { return block.n; }
block    →  { decls stmts }          { block.n = stmts.n; }

decls    →  decl decls
         |  ϵ

decl     →  type id index;           { symtable.insert(id, type); }

index    →  [ integer ]
         |  ϵ

stmts    →  stmt stmts_1             { stmts.n = new Seq(stmt.n, stmts_1.n); }
         |  ϵ

stmt     →  local = bool;            { stmt.n = new Assign(local.n, bool.n); }
         |  if (bool) stmt_1         { stmt.n = new If(bool.n, stmt_1.n); }
         |  while (bool) stmt_1      { stmt.n = new While(bool.n, stmt_1.n); }
         |  do stmt_1 while (bool);  { stmt.n = new Do(stmt_1.n, bool.n); }
         |  block                    { stmt.n = block.n; }

local    →  local_1 [bool]           { local.n = new Access(local_1.n, bool.n); }
         |  id                       { local.n = new Identifier(lexeme); }
```

```
bool      →  bool_1 || join        { bool.n = new Log('|', bool_1.n, join.n); }
          |  join                  { bool.n = join.n; }

join      →  join_1 && equality    { join.n = new Log('&', join_1.n, equality.n); }
          |  equality              { join.n = equality.n; }

equality  →  equality_1 == rel     { equality.n = new Rel('=', equality_1.n, rel.n); }
          |  equality_1 != rel     { equality.n = new Rel('≠', equality_1.n, rel.n); }
          |  rel                   { equality.n = rel.n; }

rel       →  rel_1 < ari           { rel.n = new Rel('<', rel_1.n, ari.n); }
          |  rel_1 <= ari          { rel.n = new Rel('≤', rel_1.n, ari.n); }
          |  rel_1 > ari           { rel.n = new Rel('>', rel_1.n, ari.n); }
          |  rel_1 >= ari          { rel.n = new Rel('≥', rel_1.n, ari.n); }
          |  ari                   { rel.n = ari.n; }

ari       →  ari_1 + term          { ari.n = new Ari('+', ari_1.n, term.n); }
          |  ari_1 - term          { ari.n = new Ari('-', ari_1.n, term.n); }
          |  term                  { ari.n = term.n; }

term      →  term_1 * unary        { term.n = new Ari('*', term_1.n, unary.n); }
          |  term_1 / unary        { term.n = new Ari('/', term_1.n, unary.n); }
          |  unary                 { term.n = unary.n; }

unary     →  !unary_1              { unary.n = new Unary('!', unary_1.n); }
          |  -unary_1              { unary.n = new Unary('-', unary_1.n); }
          |  factor                { unary.n = factor.n; }

factor    →  (bool)                { factor.n = bool.n; }
          |  local                 { factor.n = local.n; }
          |  integer               { factor.n = new Constant(INT, integer.value); }
          |  real                  { factor.n = new Constant(FLOAT, real.value); }
          |  true                  { factor.n = new Constant(BOOL, "true"); }
          |  false                 { factor.n = new Constant(BOOL, "false"); }
```