EXPA

SDD L-ATRIBUÍDA

ATRIBSTAT2→ ‘ident’ ATRIBSTAT3

ATRIBSTAT3.h = new Node(ident.text, getTypeOfIdent(ident.text))

ATRIBSTAT2→’int_constant’ D C EXPRESSION2

D.h = new node(int_constant.text, “int”)

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT2→ ’float_constant’ D C EXPRESSION2

D.h = new node(float_constant.text, “float”)

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT2→’string constant’ D C EXPRESSION2

D.h = new node(string_constant.text, “string”)

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT2→’(‘ NUMEXPRESSION ‘)’ D C EXPRESSION2

D.h = NUMEXPRESSION.sin

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT2→'+' FACTOR D C EXPRESSION2

FACTOR.h = +.text

D.h = FACTOR.sin

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT2→ '-' FACTOR D C EXPRESSION2

FACTOR.h = -.text

D.h = FACTOR.sin

C.h = D.sin

ATRIBSTAT2.expTree = addTree(C.sin)

ATRIBSTAT3 → B D C EXPRESSION2

B.type = ATRIBSTAT3.h

D.h = B.sin

C.h = D.sin

ATRIBSTAT3.expTree = addTree(C.sin)

B → T2 B'

B'.type = B.type

B.sin = new Node("array", T2.sin, B'.sin)

B → &

B.sin = B.type

T2 → '[' NUMEXPRESSION ']'

T2.sin = NUMEXPRESSION.sin

NUMEXPRESSION → TERM C

C.h = TERM.sin

NUMEXPRESSION.sin = C.sin

NUMEXPRESSION.expTree = addTree(NUMEXPRESSION.sin)

C → T3 C’

T3.h = C.h

C.h = T3.sin

C.sin = C’.sin

C → &

C.sin = C.h

T3 → ‘+’ TERM

T3.sin = new Node(+.text, T3.h, TERM.sin)

T3 → ‘-’ TERM

T3.sin = new Node(-.text, T3.h, TERM.sin)

TERM → UNARYEXPR D

D.h = UNARYEXPR.sin

TERM.sin = D.sin

D → T4 D’

T4.h = D.h

D’.h = T4.sin

D.sin = D’.sin

D → &

D.sin = D.h

T4 → '*' UNARYEXPR

T4.sin = new Node(*.text, T4.h, UNARYEXPR.sin)

T4 → '\' UNARYEXPR

T4.sin = new Node(\.text, T4.h, UNARYEXPR.sin)

T4 → '%' UNARYEXPR

T4.sin = new Node(%.text, T4.h, UNARYEXPR.sin)

UNARYEXPR → '+' FACTOR

FACTOR.h = +.text

UNARYEXPR.sin = FACTOR.sin

UNARYEXPR → '-' FACTOR

FACTOR.h = -.text

UNARYEXPR.sin = FACTOR.sin

UNARYEXPR → FACTOR

FACTOR.h = ""

UNARYEXPR.sin = FACTOR.sin

FACTOR → 'int_constant'

FACTOR.sin = new Node(h + int_constant.text, "int")

FACTOR → ‘float_constant’

FACTOR.sin = new Node(h + float_constant.text, “float”)

FACTOR → ‘string constant’

FACTOR.sin = new Node(h + string_constant.text, “string”)

FACTOR → LVALUE

LVALUE.h = FACTOR.h

FACTOR.sin = LVALUE.node

FACTOR → ‘(‘ NUMEXPRESSION ‘)’

FACTOR.sin = NUMEXPRESSION.sin

LVALUE → ‘ident’ B

B.type = new Node(h + ident.text, getTypeOfIdent(ident.text))

LVALUE.node = B.sin

LVALUE.expTree = addTree(LVALUE.node)

EXPA

SDT

ATRIBSTAT2→'ident' { ATRIBSTAT3.h = new Node(ident.text, getTypeOfIdent(ident.text))

} ATRIBSTAT3

ATRIBSTAT2→'int_constant' {D.h = new node(int_constant.text, "int")} D {C.h = D.sin} C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT2→'float_constant'{D.h = new node(float_constant.text, "float")} D {C.h = D.sin} C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT2→ 'string constant' {D.h = new node(string_constant.text, "string")} D {C.h = D.sin} C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT2→'(' NUMEXPRESSION ')' {D.h = NUMEXPRESSION.sin} D {C.h = D.sin } C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT2→'+' {FACTOR.h = +.text} FACTOR {D.h = FACTOR.sin} D {C.h = D.sin} C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT2→ '-' {FACTOR.h = -.text} FACTOR {D.h = FACTOR.sin} D {C.h = D.sin} C EXPRESSION2 {ATRIBSTAT2.expTree = addTree(C.sin)}

ATRIBSTAT3 → {B.type = ATRIBSTAT3.h} B {D.h = B.sin} D {C.h = D.sin} C {ATRIBSTAT3.expTree = addTree(C.sin)}

B → T2 {B'.type = B.type} B' {B.sin = new Nó("array", T2.sin, B'.sin)}

B → & {B.sin = B.type}

T2 → '[' NUMEXPRESSION ']' {T2.sin = NUMEXPRESSION.sin}

NUMEXPRESSION → TERM {C.h = TERM.sin} C {NUMEXPRESSION.sin = C.sin;
NUMEXPRESSION.expTree = addTree(NUMEXPRESSION.sin)}

C → {T3.h = C.h} T3 {C'.h = T3.sin } C' {C.sin = C'.sin}

C → & {C.sin = C.h}

T3 → '+' TERM {T3.sin = new Node(+.text, T3.h, TERM.sin)}

T3 → '-' TERM {T3.sin = new Node(-.text, T3.h, TERM.sin)}

TERM → UNARYEXPR {D.h = UNARYEXPR.sin} D {TERM.sin = D.sin}

D → {T4.h = D.h} T4 {D'.h = T4.sin} D' {D.sin = D'.sin}

D → & {D.sin = D.h}

T4 → '*' UNARYEXPR {T4.sin = new Node(*.text, T4.h, UNARYEXPR.sin)}

T4 →' \' UNARYEXPR {T4.sin = new Node(\.text, T4.h, UNARYEXPR.sin)}

T4 → '%' UNARYEXPR {T4.sin = new Node(%.text, T4.h, UNARYEXPR.sin)}

UNARYEXPR → ‘+’ {FACTOR.h = +.text} FACTOR { UNARYEXPR.sin = FACTOR.sin }

UNARYEXPR → ‘-’ {FACTOR.h = -.text} FACTOR { UNARYEXPR.sin = FACTOR.sin }

UNARYEXPR →{FACTOR.h = “”} FACTOR { UNARYEXPR.sin = FACTOR.sin }

FACTOR → ‘int_constant’ { FACTOR.sin = new Node(h + int_constant.text, “int”)}

FACTOR → ‘float_constant’ { FACTOR.sin = new Node(h + float_constant.text, “float”)}

FACTOR → ‘string constant’ { FACTOR.sin = new Node(h + string_constant.text, “string”) }

FACTOR → {LVALUE.h = FACTOR.h} LVALUE { FACTOR.sin = LVALUE.node }

FACTOR → ‘(‘ NUMEXPRESSION ‘)’ { FACTOR.sin = NUMEXPRESSION.sin }

LVALUE → ‘ident’ {B.type = new Node(h + ident.text, getTypeOfIdent(ident.text))} B {LVALUE.node = B.sin; LVALUE.expTree = addTree(LVALUE.node)}