**Segundos**

S(program) = {$}

S(def-list) = {$}

S(def-list-P) = {$}

S(def) = {var, id, $}

S(var-def) = {var, id, $}

S(var-list) = { ; }

S(id-list) = { ; , ) }

S(id-list-cont) = { ; , ) }

S(fun-def) = { var, id, $}

S(param-list) = { ) }

S(var-def-list) = { id,inc, dec, if, while, do, break, return, ; , var, $ }

S(var-def-list-P) = { id,inc, dec, if, while, do, break, return, ; , var, $ }

S(stmt-list) = { } }

S(stmt-list-P) = { } }

S(stmt) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-assign) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-incr) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-decr) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-fun-call) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-if) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-while) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-do-while) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-break) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-return) = { id,inc, dec, if, while, do, break, return, ; , } }

S(stmt-empty) = { id,inc, dec, if, while, do, break, return, ; , } }

\*\*S(fun-call) = { ; , \*,/,%, +, -, <, >, <=, >= , ==,<>, and, or, ) , , , ] }

S(expr-list) = { ), ] }

S(expr-list-cont) = { ), ] }

S(else-if-list) = { else, id,inc, dec, if, while, do, break, return, ; , } }

S(else-if-list-P) = { else, id,inc, dec, if, while, do, break, return, ; , } }

S(else) = { id,inc, dec, if, while, do, break, return, ; , } }

S(expr) = { ; , ), , , ] }

S(expr-or) = { ; , ), , , ] }

S(expr-or-P) = { ; , ), , , ] }

S(expr-and) = { or, ; , ), , , ] }

S(expr-and-P) = { or, ; , ), , , ] }

S(expr-comp) = {and, or, ; , ), , , ] }

S(expr-comp-P) = {and, or, ; , ), , , ] }

S(op-comp) = { +, -, id, [, (, lit-bool, lit-int, lit-char, lit-str }

S(expr-rel) = { ==, <>, and, or, ; , ), , , ] }

S(expr-rel-P) = { ==, <>, and, or, ; , ), , , ] }

S(op-rel) = { +, -, not, id, [, (, lit-bool, lit-int, lit-char, lit-str }

S(expr-add) = { <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(expr-add-P) = { <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(op-add) = { +, -, not, id, [, (, lit-bool, lit-int, lit-char, lit-str }

S(expr-mul) = { +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(expr-mul-P) = { +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(op-mul) = {+, -, not, id, [, (, lit-bool, lit-int, lit-char, lit-str }

S(expr-unary) = {\*, /, %, +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(op-unary) = {+, -, not, id, [, (, lit-bool, lit-int, lit-char, lit-str }

S(expr-primary) = {\*, /, %, +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(array) = {\*, /, %, +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }

S(lit) = {\*, /, %, +, -, <, <=, >, >=, ==, <>, and, or, ; , ), , , ] }