- for each block node
    - for each vardecl
        - addsymbol to table
        - set symbol annotation on tree (useful for codegen)
    - for each funcdecl
        - addsymbol to table
        - set symbol annotation on tree (useful for codegen)
        - add new scope (current_scope is parent)
        - save current_scope and update curent_scope
        - process formals
            - add each to symbol table, set symbol tree annotation
        - recursively process the block
        - restore current_scope before leaving funcdecl
    - for the statement
        - depends on kind of statement
        - compoundstatement just loop through each one and recursively call statement visitor
        - for assign
            - lookup symbol and set assign_symbol annotation
            - visit expression
            - check that expression subtree's datatype field matches the assign_symbol's type
        - (read type_specification.md for the rest of the statements)
- for each expression node
    - for number factors always annotate datatype with int
    - for variable factors, lookup the symbol in the symbol table and set the datatype to the result, set the variable_symbol to the symbol
    - for function factors
        - lookup the symbol in the symbol table and set the datatype to the return type, set the function_symbol to the symbol
        - check the function parameters against formal types, setting datatype and variable_symbol for each
    - for unary expressions
        - recursively visit the child expression
        - check the resulting datatype for the child against the operation (int for minus, bool for not)
        - set the datatype field
    - for binary expressions
        - recursively visit the left and right children
        - check that the resulting datatype annotations match
        - check the resulting datatype annotations against the operation (int for plus, minus, mult, div, mod; bool for and, or)
        - set the datatype field