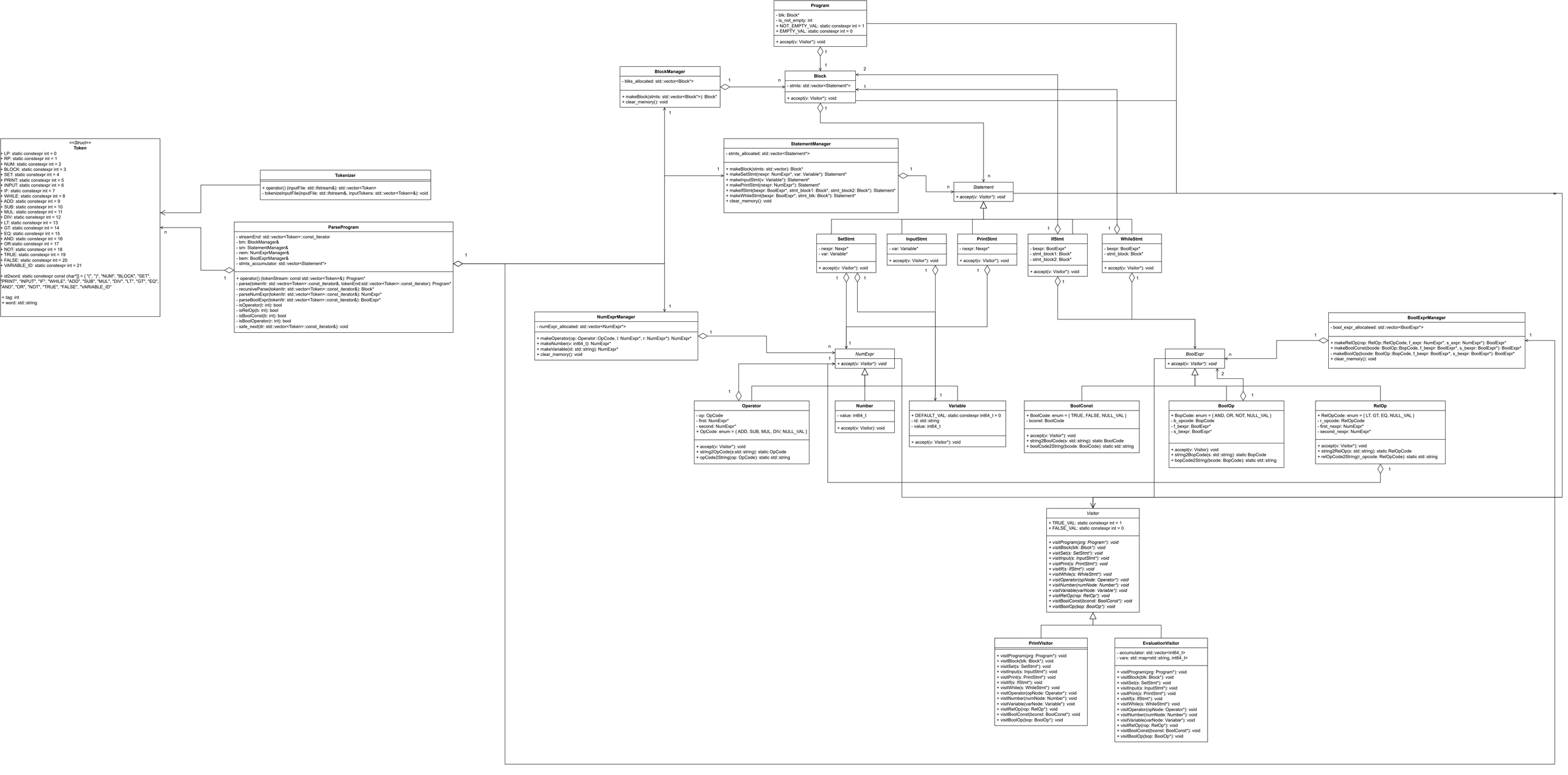
Program
- blk: Block*
- is_not_empty: int
+ NOT_EMPTY_VAL: static constexpr int = 1
+ EMPTY_VAL: static constexpr int = 0
+ accept(v: Visitor*): void
BlockManager
- blks_allocated: std::vector<Block*>
+ makeBlock(stmts: std::vector<Block*>): Block*
+ clear_memory(): void
Block
- stmts: std::vector<Statement*>
+ accept(v: Visitor*): void
StatementManager
- stmts_allocated: std::vector<Statement*>
+ makeBlock(stmts: std::vector): Block*
+ makeSetStmt(nexpr: NumExpr*, var: Variable*): Statement*
+ makeInputStmt(v: Variable*): Statement*
+ makePrintStmt(nexpr: NumExpr*): Statement*
+ makeIfStmt(bexpr: BoolExpr*, stmt_block1: Block*, stmt_block2: Block*): Statement*
+ makeWhileStmt(bexpr: BoolExpr*, stmt_blk: Block*): Statement*
+ clear_memory(): void
Statement
+ accept(v: Visitor*): void
<<Struct>>
Token
+ LP: static constexpr int = 0
+ RP: static constexpr int = 1
+ NUM: static constexpr int = 2
+ BLOCK: static constexpr int = 3
+ SET: static constexpr int = 4
+ PRINT: static constexpr int = 5
+ INPUT: static constexpr int = 6
+ IF: static constexpr int = 7
+ WHILE: static constexpr int = 8
+ ADD: static constexpr int = 9
+ SUB: static constexpr int = 10
+ MUL: static constexpr int = 11
+ DIV: static constexpr int = 12
+ LT: static constexpr int = 13
+ GT: static constexpr int = 14
+ EQ: static constexpr int = 15
+ AND: static constexpr int = 16
+ OR:static constexpr int = 17
+ NOT: static constexpr int = 18
+ TRUE: static constexpr int = 19
+ FALSE: static constexpr int = 20
+ VARIABLE_ID: static constexpr int = 21
+ id2word: static constexpr const char*[] = { "(", ")", "NUM", "BLOCK", "SET", "PRINT", "INPUT", "IF", "WHILE", "ADD", "SUB", "MUL", "DIV", "LT", "GT", "EQ", "AND", "OR", "NOT", "TRUE", "FALSE", "VARIABLE_ID"
+ tag: int
+ word: std::string
Tokenizer
+ operator() (inputFile: std::ifstream&): std::vector<Token>
- tokenizeInputFile(inputFile: std::ifstream&, inputTokens: std::vector<Token>&): void
ParseProgram
- streamEnd: std::vector<Token>::const_iterator
- bm: BlockManager&
- sm: StatementManager&
- nem: NumExprManager&
- bem: BoolExprManager&
- stmts_accumulator: std::vector<Statement*>
+ operator() (tokenStream: const std::vector<Token>&): Program*
- parse(tokenItr: std::vectro<Token>::const_iterator&, tokenEnd:std::vector<Token>::const_iterator): Program*
- recursiveParse(tokenItr: std::vector<Token>::const_iterator&): Block*
- parseNumExpr(tokenItr: std::vector<Token>::const_iterator&): NumExpr*
- parseBoolExpr(tokenItr: std::vector<Token>::const_iterator&): BoolExpr*
- isOperator(t: int): bool
- isRelOp(b: int): bool
- isBoolConst(b: int): bool
- isBoolOperator(r: int): bool
- safe_next(itr: std::vector<Token>::const_iterator&): void
SetStmt
- nexpr: Nexpr*
- var: Variable*
+ accept(v: Visitor*): void
InputStmt
- var: Variable*
+ accept(v: Visitor*): void
PrintStmt
- nexpr: Nexpr*
+ accept(v: Visitor*): void
IfStmt
- bexpr: BoolExpr*
- stmt_block1: Block*
- stmt_block2: Block*
+ accept(v: Visitor*): void
WhileStmt
- bexpr: BoolExpr*
- stmt_block: Block*
+ accept(v: Visitor*): void
NumExprManager
- numExpr_allocated: std::vector<NumExpr*>
+ makeOperator(op: Operator::OpCode, l: NumExpr*, r: NumExpr*): NumExpr*
+ makeNumber(v: int64_t): NumExpr*
+ makeVariable(id: std::string): NumExpr*
+ clear_memory(): void
NumExpr
+ accept(v: Visitor*): void
Operator
- op: OpCode
- first: NumExpr*
- second: NumExpr*
+ OpCode: enum = { ADD, SUB, MUL, DIV, NULL_VAL }
+ accept(v: Visitor*): void
+ string2OpCode(s:std::string): static OpCode
+ opCode2String(op: OpCode): static std::string
Number
- value: int64_t
+ accept(v: Visitor): void
Variable
+ DEFAULT_VAL: static constexpr int64_t = 0
- id: std::string
- value: int64_t
+ accept(v: Visitor*): void
BoolExprManager
- bool_expr_allocateed: std::vector<BoolExpr*>
+ makeRelOp(rop: RelOp::RelOpCode, f_expr: NumExpr*, s_expr: NumExpr*): BoolExpr*
+ makeBoolConst(bcode: BoolOp::BopCode, f_bexpr: BoolExpr*, s_bexpr: BoolExpr*): BoolExpr*
- makeBoolOp(bcode: BoolOp::BopCode, f_bexpr: BoolExpr*, s_bexpr: BoolExpr*): BoolExpr*
+ clear_memory(): void
BoolExpr
+ accept(v: Visitor*): void
BoolConst
+ BoolCode: enum = { TRUE, FALSE, NULL_VAL }
- bconst: BoolCode
+ accept(v: Visitor*): void
+ string2BoolCode(s: std::string): static BoolCode
+ boolCode2String(bcode: BoolCode): static std::string
BoolOp
+ BopCode: enum = { AND, OR, NOT, NULL_VAL }
- b_opcode: BopCode
- f_bexpr: BoolExpr*
- s_bexpr: BoolExpr*
+ accept(v: Visitor): void
+ string2BopCode(s: std::string): static BopCode
+ bopCode2String(bcode: BopCode): static std::string
RelOp
+ RelOpCode: enum = { LT, GT, EQ, NULL_VAL }
- r_opcode: RelOpCode
- first_nexpr: NumExpr*
- second_nexpr: NumExpr*
+ accept(v: Visitor*): void
+ string2RelOp(s: std::string): static RelOpCode
+ relOpCode2String(r_opcode: RelOpCode): static std::string
Visitor
+ TRUE_VAL: static constexpr int = 1
+ FALSE_VAL: static constexpr int = 0
+ visitProgram(prg: Program*): void
+ visitBlock(blk: Block*): void
+ visitSet(s: SetStmt*): void
+ visitInput(s: InputStmt*): void
+ visitPrint(s: PrintStmt*): void
+ visitIf(s: IfStmt*): void
+ visitWhile(s: WhileStmt*): void
+ visitOperator(opNode: Operator*): void
+ visitNumber(numNode: Number*): void
+ visitVariable(varNode: Variable*): void
+ visitRelOp(rop: RelOp*): void
+ visitBoolConst(bconst: BoolConst*): void
+ visitBoolOp(bop: BoolOp*): void
PrintVisitor
+ visitProgram(prg: Program*): void
+ visitBlock(blk: Block*): void
+ visitSet(s: SetStmt*): void
+ visitInput(s: InputStmt*): void
+ visitPrint(s: PrintStmt*): void
+ visitIf(s: IfStmt*): void
+ visitWhile(s: WhileStmt*): void
+ visitOperator(opNode: Operator*): void
+ visitNumber(numNode: Number*): void
+ visitVariable(varNode: Variable*): void
+ visitRelOp(rop: RelOp*): void
+ visitBoolConst(bconst: BoolConst*): void
+ visitBoolOp(bop: BoolOp*): void
EvaluationVisitor
- accumulator: std::vector<int64_t>
- vars: std::map<std::string, int64_t>
+ visitProgram(prg: Program*): void
+ visitBlock(blk: Block*): void
+ visitSet(s: SetStmt*): void
+ visitInput(s: InputStmt*): void
+ visitPrint(s: PrintStmt*): void
+ visitIf(s: IfStmt*): void
+ visitWhile(s: WhileStmt*): void
+ visitOperator(opNode: Operator*): void
+ visitNumber(numNode: Number*): void
+ visitVariable(varNode: Variable*): void
+ visitRelOp(rop: RelOp*): void
+ visitBoolConst(bconst: BoolConst*): void
+ visitBoolOp(bop: BoolOp*): void