

Laporan Milestone 3
Tugas Besar 1 IF2224 Teori Bahasa Formal dan Otomata
Semantic Analysis



Disusun oleh:
Maggie Zeta Rosida S - 13521117
Buege Mahara Putra - 13523037
Hanif Kalyana Aditya - 13523041
Jethro Jens Norbert Simatupang - 13523081

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

BAB 1

LANDASAN TEORI

Sebuah program Pascal-S dapat dikatakan valid secara sintaks apabila setiap rangkaian token dapat diturunkan dari aturan produksi grammar yang diberikan. Hal ini ditangani pada bagian syntax analysis oleh compiler. Namun, sebuah program yang sintaksnya benar belum tentu benar secara makna. Sebagai contoh, grammar dapat menerima ekspresi atau pemanggilan prosedur yang bentuknya valid, tetapi melanggar aturan bahasa seperti penggunaan variabel yang belum dideklarasikan, ketidaksesuaian tipe data, atau pelanggaran aturan lingkup (*scope*). Oleh karena itu, diperlukan tahap pemeriksaan lanjutan, yaitu semantic analysis, untuk memastikan bahwa struktur program yang sudah benar secara sintaks juga konsisten secara semantik.

Pada tahap semantic analysis, compiler melakukan pemeriksaan terhadap aspek-aspek makna program, antara lain:

1. Konsistensi tipe data (*type checking*)

Compiler memastikan operasi aritmatika/relasional/logika dilakukan pada operand yang kompatibel, serta memastikan tipe hasil ekspresi sesuai dengan konteks pemakaiannya.

2. Deklarasi dan penggunaan simbol

Compiler memastikan setiap identifier (variabel, konstanta, fungsi, prosedur, tipe) telah dideklarasikan sebelum digunakan, serta mencegah deklarasi ganda pada scope yang sama.

3. Aturan lingkup (scope rule)

Compiler membedakan identifier global dan lokal, serta memastikan akses identifier mengikuti aturan ruang lingkup blok pada Pascal-S.

4. Pernyataan dan alur kontrol

Compiler memvalidasi kondisi pada if/while bertipe boolean, batas for valid, serta aturan khusus lain yang ditetapkan bahasa.

Secara implementasi, semantic analysis umumnya bekerja dengan melakukan traversing terhadap parse tree/AST dan menambahkan informasi semantik pada node. Proses ini disebut anotasi. Anotasi dapat berupa penambahan/penyimpanan atribut seperti:

- tipe data hasil ekspresi
- referensi identifier menuju entri pada symbol table,
- informasi scope/level blok,
- serta informasi lain yang dibutuhkan untuk tahap berikutnya (misalnya intermediate code generation).

1.1 Attributed Grammar

Attributed grammar adalah grammar yang dilengkapi dengan atribut (misalnya tipe data, nilai, atau informasi simbol) beserta aturan semantik untuk menghitung atribut tersebut. Dalam praktik compiler, atribut digunakan untuk melakukan type checking dan memastikan konsistensi semantik, misalnya menentukan tipe sebuah ekspresi berdasarkan tipe operand dan operatornya.

1.2 Symbol Table

Symbol table adalah struktur data yang menyimpan informasi mengenai simbol/identifier yang muncul pada program. Informasi minimal yang umumnya disimpan mencakup:

- nama identifier
- kategori simbol (variabel/konstanta/fungsi/prosedur/tipe)
- tipe data
- scope/level blok tempat simbol didefinisikan
- (opsional) informasi tambahan seperti parameter fungsi/prosedur, tipe kembalian fungsi, alamat/offset memori, dsb.

Dalam bahasa yang memiliki blok dan scope bertingkat seperti Pascal-S, pengelolaan scope sering diimplementasikan menggunakan pendekatan stack atau stack of tables. Setiap memasuki blok baru dilakukan push tabel scope baru, dan saat keluar blok dilakukan pop. Dengan cara ini, pencarian identifier akan memprioritaskan deklarasi pada scope terdalam (paling atas stack) sebelum mencari ke scope luar.

1.3 Visitor / Visit Functions

Untuk melakukan semantic analysis secara terstruktur, digunakan pola Visitor (atau fungsi visit per jenis node). Untuk setiap jenis node pada parse tree/AST disediakan fungsi visit_<node>. Secara umum, fungsi visit akan:

1. memanggil visit pada child node (traversal),
2. mengakses atau memperbarui symbol table sesuai konstruksi bahasa (misalnya saat deklarasi),
3. menghitung atribut semantik (misalnya tipe ekspresi),
4. menambahkan anotasi pada node (misalnya node.type = ... atau node.symbol_ref = ...),
5. serta melaporkan error semantik jika ada pelanggaran aturan (misalnya “identifier belum dideklarasikan” atau “tipe tidak kompatibel”).

Dengan pendekatan ini, aturan semantik dapat diimplementasikan secara modular dan selaras dengan struktur grammar/parse tree yang sudah dibangun pada milestone sebelumnya.

BAB 2

PERANCANGAN & IMPLEMENTASI

2.1 Perancangan Sistem

2.1.1 Konsep Dasar

Semantic analyzer ini dirancang untuk menganalisis makna program Pascal-S setelah melalui tahap parsing. Komponen ini melakukan:

- Pembangunan AST dari parse tree menggunakan Syntax-Directed Translation
- Type checking untuk memvalidasi konsistensi tipe data
- Scope resolution dengan symbol table stack
- Dekorasi AST dengan informasi semantik (tipe, referensi symbol table, scope)

2.1.2 Arsitektur Sistem

```
semantic_analyzer/
└── AST Nodes (ast_nodes.py)
└── Symbol Table (symbol_table.py)
└── Semantic Analyzer (semantic_analyzer.py)
└── AST Printer (ast_printer.py)
```

Alur Kerja:

- Parse tree dari parser menjadi input
- Visitor functions mengunjungi setiap node
- Symbol table menyimpan informasi identifier
- AST dibangun dengan anotasi semantik
- Validasi type dan scope dilakukan selama traversal

2.2 Implementasi Symbol Table

2.2.1 Struktur Symbol Table

Symbol table mengimplementasi spesifikasi Pascal-S dengan tiga tabel terpisah:

```
class SymbolTable:
    def __init__(self):
        self.tab = []      # Identifier table (mulai index 29 untuk user)
        self.btab = []    # Block table
        self.atab = []    # Array table
        self.display = [] # Stack untuk scope management
        self.level = -1   # Current lexical level
```

Penjelasan:

- tab: Menyimpan semua identifier (variabel, konstanta, prosedur, fungsi)
- btab: Menyimpan informasi block (scope) dengan linked list
- atab: Menyimpan informasi array (bounds, element type)
- display: Stack untuk mengelola nested scope

2.2.2 Inisialisasi Reserved Words

Symbol table diinisialisasi dengan reserved words Pascal-S pada indices 0-28:

```

def __init_reserved_words(self):
    # Types dasar: integer, real, boolean, char, string
    # Keywords: program, variabel, mulai, selesai, jika, maka, dll
    # Built-in procedures: writeln, readln, write, read

```

Reserved words sudah predefined sehingga user identifiers mulai dari index 29.

2.2.3 Manajemen Scope dengan Block Table

```

def enter_block(self) -> int:
    self.level += 1
    block_index = len(self.btab)
    self.btab.append({
        "last": 0,      # Pointer ke identifier terakhir dalam block
        "lpar": 0,      # Pointer ke parameter terakhir
        "psze": 0,      # Total ukuran parameter
        "vsze": 0       # Total ukuran variabel lokal
    })
    self.display.append(block_index)
    return block_index

```

Mekanisme Scope:

- Setiap procedure/function/membuka block membuat scope baru
- display stack melacak scope aktif
- level menunjukkan kedalaman nesting
- Linked list dalam btab menghubungkan identifiers dalam scope yang sama

2.3 Implementasi Visitor Pattern

2.3.1 Visitor Dispatch Mechanism

```

def visit(self, node: ParseNode) -> ASTNode:
    method_name = f'visit_{node.name.replace("<", "")}.replace(">", "")\
.replace("-", "_")'
    method = getattr(self, method_name, self.visit_default)
    return method(node)

```

Cara Kerja:

- Setiap node parse tree memiliki fungsi visit khusus
- Nama method diturunkan dari nama node
- Fallback ke visit_default untuk node tanpa handler khusus

2.3.2 Semantic Actions untuk Production Rules

Assignment Statement:

```

def visit_assignment_statement(self, node: ParseNode) -> ASTNode:
    # Pattern: <variable> := <expression>
    target_node = self.visit(variable_child)
    value_node = self.visit(expression_child)

    # Type checking
    if not self.is_type_compatible(target_node.data_type,
value_node.data_type):
        self.error("Type mismatch in assignment")

    # Bangun AST node
    return AssignmentNode(target=target_node, value=value_node)

```

Binary Expressions:

```

def visit_expression(self, node: ParseNode) -> ASTNode:
    if len(node.children) == 1:
        # Single expression
        return self.visit(node.children[0])
    else:
        # Expression dengan operator
        left = self.visit(node.children[0])
        right = self.visit(node.children[2])
        operator = self._extract_operator(node.children[1])

        # Type inference untuk operasi
        result_type = self.get_expression_type(left.data_type,
right.data_type, operator)
        return BinaryExpressionNode(operator, left, right,
result_type)

```

Deklarasi Variabel:

```

def visit_var_declaraction(self, node: ParseNode) -> ASTNode:
    identifiers = self.extract_identifiers(identifier_list_child)
    type_ast = self.visit(type_child)

    for identifier in identifiers:
        # Cek duplicate dalam scope yang sama
        if not self.check_duplicate_identifier(identifier):
            # Masukkan ke symbol table
            var_idx = self.symbol_table.enter_identifier(
                identifier, ObjType.VARIABLE,
type_ast.data_type.value
            )
            # Bangun VarDeclNode dengan referensi symbol table
            var_node = VarDeclNode(identifier, type_ast.data_type,
var_idx)

    return declaration_node

```

2.4 Type Checking System

2.4.1 Type Compatibility

```

def is_type_compatible(self, target_type: BaseType, source_type: BaseType,
                      is_parameter: bool = False) -> bool:
    if target_type == source_type:
        return True

    # Implicit conversion: integer → real
    if target_type == BaseType.REAL and source_type ==
BaseType.INTEGER:
        return True

    # Strict checking untuk parameter functions
    if is_parameter:
        if target_type == BaseType.INTEGER and source_type ==
BaseType.REAL:
            return False # Tidak boleh real → integer
        # Boolean dan char harus exact match
        if target_type in [BaseType.BOOLEAN, BaseType.CHAR]:
            return target_type == source_type

    return False

```

Aturan Type Checking:

- Assignment: target dan value harus compatible
- Parameter passing: strict type matching untuk beberapa tipe

- Operasi aritmetik: integer/real dengan hasil type inference
- Operasi relasional: selalu menghasilkan boolean

2.4.2 Expression Type Inference

```
def get_expression_type(self, left_type: BaseType, right_type: BaseType,
                       operator: str) -> BaseType:
    # Arithmetic operators
    if operator in ['+', '-', '*', '/']:
        if left_type == BaseType.REAL or right_type == BaseType.REAL:
            return BaseType.REAL
        elif left_type == BaseType.INTEGER and right_type == BaseType.INTEGER:
            return BaseType.INTEGER

    # Relational operators
    elif operator in ['=', '<>', '<', '<=', '>', '>=']:
        return BaseType.BOOLEAN

    # Logical operators
    elif operator in ['dan', 'atau']:
        return BaseType.BOOLEAN

    return BaseType VOID # Error case
```

2.5 Scope Resolution

2.5.1 Identifier Lookup dengan Display Mechanism

```
def find_identifier(self, name: str) -> Optional[int]:
    # Search dari current level down ke global level
    for level in range(self.level, -1, -1):
        block_index = self.display[level]
        last_idx = self.btab[block_index]["last"]

        # Traverse linked list dalam block
        current_idx = last_idx
        while current_idx >= self.user_id_start:
            if self.tab[current_idx]["name"] == name:
                return current_idx
            current_idx = self.tab[current_idx]["link"]

    return None # Identifier tidak ditemukan
```

Algoritma Pencarian:

- Mulai dari current scope (level tertinggi)
- Traverse linked list dalam block tersebut
- Jika tidak ditemukan, turun ke scope outer
- Ulangi hingga global scope
- Cek reserved words jika masih tidak ditemukan

2.5.2 Error Detection

```
def check_duplicate_identifier(self, name: str, token: Token = None):
    existing_idx = self.symbol_table.find_identifier(name)
    if existing_idx is not None and
       self.symbol_table.tab[existing_idx]["lev"] == self.level:
        self.error(f"Duplicate identifier '{name}'", token)
    return True
```

```
        return False
```

Validasi Semantic:

- Duplicate identifier dalam scope yang sama
- Undefined identifier references
- Type mismatches dalam assignments dan operations
- Constant assignment protection

2.6 Pembangunan Decorated AST

2.6.1 Struktur AST Node dengan Anotasi

```
@dataclass
class ASTNode:
    node_type: str          # Jenis node (Program, Assignment, dll)
    children: List[ASTNode]  # Child nodes
    token: Optional[Token]   # Token source (untuk error reporting)
    data_type: Optional[BaseType]  # Tipe ekspresi (hasil type
    inference)
    tab_index: int = -1      # Referensi ke symbol table (tab)
    block_index: int = -1    # Informasi scope (btab reference)
```

Specialized Nodes:

- ProgramNode: Root node dengan nama program
- VarDeclNode: Deklarasi variabel dengan identifier dan type
- AssignmentNode: Assignment statement dengan target dan value
- BinaryExpressionNode: Operasi binary dengan operator
- ProcedureCallNode: Pemanggilan procedure dengan parameters

2.7 Proses Semantic Analysis

2.7.1 Main Analysis Loop

```
def analyze(self, parse_tree: ParseNode) -> ASTNode:
    # Inisialisasi
    self.errors.clear()
    self.symbol_table = SymbolTable()

    # Enter global scope
    global_block_idx = self.symbol_table.enter_scope()

    # Traverse parse tree dan bangun AST
    self.current_ast = self.visit(parse_tree)

    # Validasi tambahan
    self.validate_variable_initialization()
    self.check_unresolved_references()

    # Output decorated AST dan symbol tables
    return self.current_ast
```

2.7.2 Validasi Semantic Tambahan

```
def validate_variable_initialization(self):
    """Deteksi penggunaan variabel sebelum inisialisasi"""
    assigned_vars = set()

    def track_usage(node):
        if isinstance(node, AssignmentNode):
            # Track variabel yang di-assign
            target = node.children[0]
            if isinstance(target, VariableNode):
                assigned_vars.add(target.identifier)
        elif isinstance(node, VariableNode):
            # Cek jika variabel digunakan sebelum di-assign
            if (node.identifier not in assigned_vars and
                not self._is_parameter(node)):
                self.warning(f"Variable '{node.identifier}' might be
uninitialized")

    self._traverse_ast(track_usage)
```

2.8 Error Handling

2.8.1 Semantic Error Types

```
def error(self, message: str, token: Token = None):
    if token:
        location = f" at line {token.line}, column {token.column}"
    else:
        location = ""
    self.errors.append(f"Semantic Error{location}: {message}")
```

Jenis Error yang Dideteksi:

- Type mismatches: Assignment, parameter passing, operations
- Scope violations: Undefined identifiers, duplicate declarations
- Constant assignment: Attempt to modify constants
- Array bounds: Index out of range
- Parameter count: Mismatch in procedure/function calls

2.8.2 Recovery Mechanism

- Error tidak menghentikan proses analisis
- Multiple errors dapat dideteksi dalam satu run
- AST tetap dibangun untuk bagian yang valid
- Error messages informatif dengan lokasi token

BAB 3

PENGUJIAN

3.1 Kasus Uji 1

Input 1:

```
program Input1;
konstanta
    MAX = 100;
    MIN = 1;
tipe
    Range = 1..10;
    ArrayInt = larik[1..10] dari integer;
variabel
    a, b, c: integer;
    arr: ArrayInt;
mulai
    a := MAX;
    b := MIN;
    c := a + b;
selesai.
```

Output 1:

```
==== SEMANTIC ANALYSIS ====

==== DECORATED AST ====
ProgramNode(name: 'Input1')
  └─ Declarations
    └─ ConstDecl → tab_index:33, type:integer, lev:0
    └─ ConstDecl → tab_index:34, type:integer, lev:0
    └─ TypeDecl → tab_index:35, type:integer, lev:0
    └─ TypeDecl → tab_index:36, type:array, lev:0
    └─ VarDecl('a') → tab_index:37, type:integer, lev:0
    └─ VarDecl('b') → tab_index:38, type:integer, lev:0
    └─ VarDecl('c') → tab_index:39, type:integer, lev:0
    └─ VarDecl('arr') → tab_index:40, type:array, lev:0
  └─ Block → block_index:1, lev:1
    └─ Assign('a' := ConstIdentifier(type=BaseType.INTEGER,
tab_idx=33)) → type:void
      └─ target 'a' → tab_index:37, type:integer
        └─ value ConstIdentifier(type=BaseType.INTEGER, tab_idx=33)
  → type:integer
    └─ Assign('b' := ConstIdentifier(type=BaseType.INTEGER,
tab_idx=34)) → type:void
      └─ target 'b' → tab_index:38, type:integer
        └─ value ConstIdentifier(type=BaseType.INTEGER, tab_idx=34)
  → type:integer
    └─ Assign('c' := 'a'+Var('b')) → type:void
      └─ target 'c' → tab_index:39, type:integer
        └─ BinOp '+' → type:integer
          └─ 'a' → tab_index:37, type:integer
          └─ 'b' → tab_index:38, type:integer

==== SYMBOL TABLES ====

Identifier Table (tab):
Idx  Name       Obj      Type     Ref   Nrm   Lev   Adr   Link
-----
0   program    PROGRAM  VOID     0     1     0     0     0
1   variabel   TYPE    VOID     0     1     0     0     0
2   mulai      TYPE    VOID     0     1     0     0     1
3   selesai    TYPE    VOID     0     1     0     0     2
4   jika       TYPE    VOID     0     1     0     0     3
5   maka       TYPE    VOID     0     1     0     0     4
```

Screenshot 1:

```
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/input-1.pas
```

```
== SEMANTIC ANALYSIS ==
```

```
== DECORATED AST ==
ProgramNode(name: 'Input1')
|- Declarations
  |- ConstDecl → tab_index:33, type:integer, lev:0
  |- ConstDecl → tab_index:34, type:integer, lev:0
  |- TypeDecl → tab_index:35, type:integer, lev:0
  |- TypeDecl → tab_index:36, type:array, lev:0
  |- VarDecl('a') → tab_index:37, type:integer, lev:0
  |- VarDecl('b') → tab_index:38, type:integer, lev:0
  |- VarDecl('c') → tab_index:39, type:integer, lev:0
  |- VarDecl('arr') → tab_index:40, type:array, lev:0
|- Block → block_index:1, lev:1
  |- Assign('a' := ConstIdentifier(type=BaseType.INTEGER, tab_idx=33)) → type:void
    |- target 'a' → tab_index:37, type:integer
      |- value ConstIdentifier(type=BaseType.INTEGER, tab_idx=33) → type:integer
  |- Assign('b' := ConstIdentifier(type=BaseType.INTEGER, tab_idx=34)) → type:void
    |- target 'b' → tab_index:38, type:integer
      |- value ConstIdentifier(type=BaseType.INTEGER, tab_idx=34) → type:integer
  |- Assign('c' := 'a'+Var('b')) → type:void
    |- target 'c' → tab_index:39, type:integer
      |- BinOp '+' → type:integer
        |- 'a' → tab_index:37, type:integer
        |- 'b' → tab_index:38, type:integer
```

```
== SYMBOL TABLES ==
```

Identifier Table (tab):

| Idx | Name | Obj | Type | Ref | Nrm | Lev | Adr | Link |
|-----|-----------|-----------|-------|-----|-----|-----|-----|------|
| 0 | program | PROGRAM | VOID | 0 | 1 | 0 | 0 | 0 |
| 1 | variabel | TYPE | VOID | 0 | 1 | 0 | 0 | 0 |
| 2 | mulai | TYPE | VOID | 0 | 1 | 0 | 0 | 1 |
| 3 | selesai | TYPE | VOID | 0 | 1 | 0 | 0 | 2 |
| 4 | jika | TYPE | VOID | 0 | 1 | 0 | 0 | 3 |
| 5 | maka | TYPE | VOID | 0 | 1 | 0 | 0 | 4 |
| 6 | selainitu | TYPE | VOID | 0 | 1 | 0 | 0 | 5 |
| 7 | selama | TYPE | VOID | 0 | 1 | 0 | 0 | 6 |
| 8 | lakukan | TYPE | VOID | 0 | 1 | 0 | 0 | 7 |
| 9 | untuk | TYPE | VOID | 0 | 1 | 0 | 0 | 8 |
| 10 | ke | TYPE | VOID | 0 | 1 | 0 | 0 | 9 |
| 11 | turunke | TYPE | VOID | 0 | 1 | 0 | 0 | 10 |
| 12 | lanik | TYPE | ARRAY | 0 | 1 | 0 | 0 | 11 |
| 13 | dari | TYPE | VOID | 0 | 1 | 0 | 0 | 12 |
| 14 | prosedur | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 13 |
| 15 | fungsi | FUNCTION | VOID | 0 | 1 | 0 | 0 | 14 |
| 16 | konstanta | CONSTANT | VOID | 0 | 1 | 0 | 0 | 15 |
| 17 | tipe | TYPE | VOID | 0 | 1 | 0 | 0 | 16 |

```
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/input-1.pas
```

| | | | | | | | | |
|----|----------|-----------|---------|---|---|---|---|----|
| 17 | tipe | TYPE | VOID | 0 | 1 | 0 | 0 | 16 |
| 18 | kasus | TYPE | VOID | 0 | 1 | 0 | 0 | 17 |
| 19 | rekaman | TYPE | RECORD | 0 | 1 | 0 | 0 | 18 |
| 20 | ulangi | TYPE | VOID | 0 | 1 | 0 | 0 | 19 |
| 21 | sampai | TYPE | VOID | 0 | 1 | 0 | 0 | 20 |
| 22 | integer | TYPE | INTEGER | 0 | 1 | 0 | 0 | 21 |
| 23 | real | TYPE | REAL | 0 | 1 | 0 | 0 | 22 |
| 24 | boolean | TYPE | BOOLEAN | 0 | 1 | 0 | 0 | 23 |
| 25 | char | TYPE | CHAR | 0 | 1 | 0 | 0 | 24 |
| 26 | string | TYPE | STRING | 0 | 1 | 0 | 0 | 25 |
| 27 | writeln | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 26 |
| 28 | readin | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 27 |
| 29 | write | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 28 |
| 30 | read | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 29 |
| 32 | Input1 | PROGRAM | VOID | 0 | 1 | 0 | 0 | 0 |
| 33 | MAX | CONSTANT | INTEGER | 0 | 1 | 0 | 0 | 32 |
| 34 | MIN | CONSTANT | INTEGER | 0 | 1 | 0 | 0 | 33 |
| 35 | Range | TYPE | INTEGER | 0 | 1 | 0 | 0 | 34 |
| 36 | ArrayInt | TYPE | ARRAY | 0 | 1 | 0 | 0 | 35 |
| 37 | a | VARIABLE | INTEGER | 0 | 1 | 0 | 0 | 36 |
| 38 | b | VARIABLE | INTEGER | 0 | 1 | 0 | 1 | 37 |
| 39 | c | VARIABLE | INTEGER | 0 | 1 | 0 | 2 | 38 |
| 40 | arr | VARIABLE | ARRAY | 0 | 1 | 0 | 3 | 39 |

Block Table (btab):

| Idx | Last | Lpar | Psze | Vsze |
|-----|------|------|------|------|
| 0 | 40 | 0 | 0 | 4 |
| 1 | 0 | 0 | 0 | 0 |

Array Table (atab):

| Idx | IdxType | ElemType | Eref | Low | High | ElemSize | Size |
|-----|---------|----------|------|-----|------|----------|------|
| 0 | INTEGER | INTEGER | 0 | 1 | 10 | 1 | 10 |

✓ No semantic errors found

```
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224>
```

3.2 Kasus Uji 2

Input 2:

```
program Input3;
tipe
  Point = rekaman
  x, y: integer;
  selesai;
  Matrix = larik[1..3, 1..3] dari real;
variabel
```

```

p: Point;
m: Matrix;
mulai
p.x := 10;
p.y := 20;
m[1,1] := 1.5;
selesai.

```

Output 2:

```

==== SEMANTIC ANALYSIS ====

==== DECORATED AST ====
ProgramNode(name: 'Input3')
  └─ Declarations
    └─ TypeDecl → tab_index:33, type:void, lev:0
    └─ TypeDecl → tab_index:34, type:array, lev:0
    └─ VarDecl('p') → tab_index:35, type:void, lev:0
    └─ VarDecl('m') → tab_index:36, type:array, lev:0
  └─ Block → block_index:1, lev:1
    └─ Assign('p' := 10) → type:void
      └─ target 'p' → tab_index:35, type:void
        └─ value 10 → type:integer
    └─ Assign('p' := 20) → type:void
      └─ target 'p' → tab_index:35, type:void
        └─ value 20 → type:integer
    └─ Assign('m' := 1.5) → type:void
      └─ target 'm' → tab_index:36, type:real
        └─ value 1.5 → type:real

==== SYMBOL TABLES ====

Identifier Table (tab):
Idx  Name       Obj      Type     Ref   Nrm   Lev   Addr   Link
-----+
 0  program    PROGRAM  VOID     0     1     0     0     0
 1  variabel   TYPE    VOID     0     1     0     0     0
 2  mulai      TYPE    VOID     0     1     0     0     1
 3  selesai    TYPE    VOID     0     1     0     0     2
 4  jika        TYPE    VOID     0     1     0     0     3
 5  maka        TYPE    VOID     0     1     0     0     4
 6  selainitu  TYPE    VOID     0     1     0     0     5
 7  selama     TYPE    VOID     0     1     0     0     6
 8  lakukan   TYPE    VOID     0     1     0     0     7
 9  untuk      TYPE    VOID     0     1     0     0     8
10  ke          TYPE    VOID     0     1     0     0     9
11  turunke   TYPE    VOID     0     1     0     0    10
12  larik      TYPE    ARRAY   0     1     0     0    11
13  dari        TYPE    VOID     0     1     0     0    12
14  prosedur   PROCEDURE VOID     0     1     0     0    13
15  fungsi     FUNCTION VOID     0     1     0     0    14
16  konstanta  CONSTANT VOID     0     1     0     0    15
17  tipe         TYPE    VOID     0     1     0     0    16
18  kasus       TYPE    VOID     0     1     0     0    17
19  rekaman    TYPE    RECORD  0     1     0     0    18
20  ulangi     TYPE    VOID     0     1     0     0    19
21  sampai     TYPE    VOID     0     1     0     0    20
22  integer     TYPE    INTEGER 0     1     0     0    21
23  real         TYPE    REAL    0     1     0     0    22
24  boolean     TYPE    BOOLEAN 0     1     0     0    23
25  char         TYPE    CHAR    0     1     0     0    24
26  string       TYPE    STRING  0     1     0     0    25
27  writeln    PROCEDURE VOID     0     1     0     0    26
28  readln     PROCEDURE VOID     0     1     0     0    27
29  write       PROCEDURE VOID     0     1     0     0    28
30  read        PROCEDURE VOID     0     1     0     0    29

```

```

32 Input3      PROGRAM      VOID      0   1   0   0   0
33 Point       TYPE         VOID      0   1   0   0   32
34 Matrix      TYPE         ARRAY     0   1   0   0   33
35 p           VARIABLE    VOID      0   1   0   0   34
36 m           VARIABLE    ARRAY     0   1   0   1   35

Block Table (btab):
Idx  Last  Lpar  Psze  Vsze
-----
0    36    0    0    2
1    0    0    0    0

Array Table (atab):
Idx  IdxType  ElemtType  Eref  Low  High  Elemsize  Size
-----
0    INTEGER   REAL        0    1    3      1      3

✓ No semantic errors found

```

Screenshot 2:

```
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/input-3.pas
*** SEMANTIC ANALYSIS ***
```

```
*** DECORATED AST ***
ProgramNode(name: 'Input3')
  Declarations
    TypeDecl > tab_index:33, type:void, lev:0
    TypeDecl > tab_index:34, type:array, lev:0
    VarDecl('p') > tab_index:35, type:void, lev:0
    VarDecl('m') > tab_index:36, type:array, lev:0
  Block > block_index:1, lev:1
    Assign('p' := 10) + type:void
      target 'p' > tab_index:35, type:void
      value 10 > type:integer
    Assign('p' := 20) + type:void
      target 'p' > tab_index:35, type:void
      value 20 > type:integer
    Assign('m' := 1.5) + type:void
      target 'm' > tab_index:36, type:real
      value 1.5 > type:real
```

```
*** SYMBOL TABLES ***
```

| Idx | Name | Obj | Type | Ref | Nrm | Lev | Adr | Link |
|-----|-----------|-----------|---------|-----|-----|-----|-----|------|
| 0 | program | PROGRAM | VOID | 0 | 1 | 0 | 0 | 0 |
| 1 | variabel | TYPE | VOID | 0 | 1 | 0 | 0 | 0 |
| 2 | mulai | TYPE | VOID | 0 | 1 | 0 | 0 | 1 |
| 3 | selesai | TYPE | VOID | 0 | 1 | 0 | 0 | 2 |
| 4 | jika | TYPE | VOID | 0 | 1 | 0 | 0 | 3 |
| 5 | maka | TYPE | VOID | 0 | 1 | 0 | 0 | 4 |
| 6 | selainitu | TYPE | VOID | 0 | 1 | 0 | 0 | 5 |
| 7 | selama | TYPE | VOID | 0 | 1 | 0 | 0 | 6 |
| 8 | lakukan | TYPE | VOID | 0 | 1 | 0 | 0 | 7 |
| 9 | untuk | TYPE | VOID | 0 | 1 | 0 | 0 | 8 |
| 10 | ke | TYPE | VOID | 0 | 1 | 0 | 0 | 9 |
| 11 | turunke | TYPE | VOID | 0 | 1 | 0 | 0 | 10 |
| 12 | larik | TYPE | ARRAY | 0 | 1 | 0 | 0 | 11 |
| 13 | dari | TYPE | VOID | 0 | 1 | 0 | 0 | 12 |
| 14 | prosedur | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 13 |
| 15 | fungsi | FUNCTION | VOID | 0 | 1 | 0 | 0 | 14 |
| 16 | konstanta | CONSTANT | VOID | 0 | 1 | 0 | 0 | 15 |
| 17 | tipe | TYPE | VOID | 0 | 1 | 0 | 0 | 16 |
| 18 | kasus | TYPE | VOID | 0 | 1 | 0 | 0 | 17 |
| 19 | rekaman | TYPE | RECORD | 0 | 1 | 0 | 0 | 18 |
| 20 | ulangi | TYPE | VOID | 0 | 1 | 0 | 0 | 19 |
| 21 | sampai | TYPE | VOID | 0 | 1 | 0 | 0 | 20 |
| 22 | integer | TYPE | INTEGER | 0 | 1 | 0 | 0 | 21 |
| 23 | real | TYPE | REAL | 0 | 1 | 0 | 0 | 22 |

```

23 real      TYPE      REAL      0   1   0   0   22
24 boolean   TYPE      BOOLEAN   0   1   0   0   23
25 char      TYPE      CHAR     0   1   0   0   24
26 string    TYPE      STRING   0   1   0   0   25
27 writeln  PROCEDURE VOID     0   1   0   0   26
28 readln   PROCEDURE VOID     0   1   0   0   27
29 write    PROCEDURE VOID     0   1   0   0   28
30 read     PROCEDURE VOID     0   1   0   0   29
32 Input3   PROGRAM   VOID     0   1   0   0   0
33 Point    TYPE      VOID     0   1   0   0   32
34 Matrix   TYPE      ARRAY    0   1   0   0   33
35 p        VARIABLE  VOID     0   1   0   0   34
36 m        VARIABLE  ARRAY    0   1   0   1   35

Block Table (btab):
Idx Last Lpar Psze Vsze
-----
0 36 0 0 2
1 0 0 0 0

Array Table (atab):
Idx IdxType ElemtType Eref Low High ElemtSize Size
-----
0 INTEGER REAL      0   1   3   1   3

✓ No semantic errors found
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224>

```

3.3 Kasus Uji 3

Input 3:

```

program Input12;
variabel x: integer;
mulai
  ulangi
    x := x + 1;
    sampai x = 5;
    kasus x dari
      1: writeln('one');
      2: writeln('two');
    selesai;
selesai.

```

Output 3:

```

==== SEMANTIC ANALYSIS ====
==== DECORATED AST ====
ProgramNode(name: 'Input12')
  └ Declarations
    └ VarDecl('x') → tab_index:33, type:integer, lev:0
      Block → block_index:1, lev:1
        └ Assign('x' := 'x'+1) → type:void
          └ target 'x' → tab_index:33, type:integer
            BinOp '+' → type:integer
              └ 'x' → tab_index:33, type:integer
                1 → type:integer
        └ writeln(...) → predefined, tab_index:27
        └ writeln(...) → predefined, tab_index:27

==== SYMBOL TABLES ====
Identifier Table (tab):
Idx Name      Obj      Type      Ref      Nrm      Lev      Adr      Link
-----
0 program   PROGRAM  VOID      0       1       0       0       0
1 variabel  TYPE    VOID      0       1       0       0       0
2 mulai     TYPE    VOID      0       1       0       0       1
3 selesai   TYPE    VOID      0       1       0       0       2
4 jika      TYPE    VOID      0       1       0       0       3
5 maka      TYPE    VOID      0       1       0       0       4
6 selainitu TYPE    VOID      0       1       0       0       5
7 selama    TYPE    VOID      0       1       0       0       6
8 lakukan  TYPE    VOID      0       1       0       0       7
9 untuk     TYPE    VOID      0       1       0       0       8

```

| | | | | | | | | |
|----|-----------|-----------|---------|---|---|---|---|----|
| 10 | ke | TYPE | VOID | 0 | 1 | 0 | 0 | 9 |
| 11 | turunke | TYPE | VOID | 0 | 1 | 0 | 0 | 10 |
| 12 | larik | TYPE | ARRAY | 0 | 1 | 0 | 0 | 11 |
| 13 | dari | TYPE | VOID | 0 | 1 | 0 | 0 | 12 |
| 14 | prosedur | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 13 |
| 15 | fungsi | FUNCTION | VOID | 0 | 1 | 0 | 0 | 14 |
| 16 | konstanta | CONSTANT | VOID | 0 | 1 | 0 | 0 | 15 |
| 17 | tipe | TYPE | VOID | 0 | 1 | 0 | 0 | 16 |
| 18 | kasus | TYPE | VOID | 0 | 1 | 0 | 0 | 17 |
| 19 | rekaman | TYPE | RECORD | 0 | 1 | 0 | 0 | 18 |
| 20 | ulangi | TYPE | VOID | 0 | 1 | 0 | 0 | 19 |
| 21 | sampai | TYPE | VOID | 0 | 1 | 0 | 0 | 20 |
| 22 | integer | TYPE | INTEGER | 0 | 1 | 0 | 0 | 21 |
| 23 | real | TYPE | REAL | 0 | 1 | 0 | 0 | 22 |
| 24 | boolean | TYPE | BOOLEAN | 0 | 1 | 0 | 0 | 23 |
| 25 | char | TYPE | CHAR | 0 | 1 | 0 | 0 | 24 |
| 26 | string | TYPE | STRING | 0 | 1 | 0 | 0 | 25 |
| 27 | writeln | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 26 |
| 28 | readln | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 27 |
| 29 | write | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 28 |
| 30 | read | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 29 |
| 32 | Input12 | PROGRAM | VOID | 0 | 1 | 0 | 0 | 0 |
| 33 | x | VARIABLE | INTEGER | 0 | 1 | 0 | 0 | 32 |

Block Table (btab):

| Idx | Last | Lpar | Psze | Vsze |
|-----|------|------|------|------|
| 0 | 33 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |

Array Table (atab):

(empty)

✓ No semantic errors found

Screenshot 3:

```

PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/input-12.pas
*** SEMANTIC ANALYSIS ***

*** DECORATED AST ***
ProgramNode(name: 'Input12')
|- Declarations
  |- VarDecl('x') -> tab_index:33, type:integer, lev:0
  Block -> block_index:1, lev:1
    |- Assign('x' := 'x'+1) -> type:void
      target 'x' -> tab_index:33, type:integer
      BinOp '+' -> type:integer
        'x' + tab_index:33, type:integer
        1 -> type:integer
    |- writeln(...) -> predefined, tab_index:27
    |- writeln(...) -> predefined, tab_index:27

*** SYMBOL TABLES ***

Identifier Table (tab):
Idx Name Obj Type Ref Nrm Lev Adr Link
----- 
0 program PROGRAM VOID 0 1 0 0 0 0
1 variabel TYPE VOID 0 1 0 0 0 0
2 mulai TYPE VOID 0 1 0 0 0 1
3 selesai TYPE VOID 0 1 0 0 0 2
4 jika TYPE VOID 0 1 0 0 0 3
5 maka TYPE VOID 0 1 0 0 0 4
6 selainitu TYPE VOID 0 1 0 0 0 5
7 selama TYPE VOID 0 1 0 0 0 6
8 lakukan TYPE VOID 0 1 0 0 0 7
9 untuk TYPE VOID 0 1 0 0 0 8
10 ke TYPE VOID 0 1 0 0 0 9
11 turunke TYPE VOID 0 1 0 0 0 10
12 larik TYPE ARRAY 0 1 0 0 0 11
13 dari TYPE VOID 0 1 0 0 0 12
14 prosedur PROCEDURE VOID 0 1 0 0 0 13
15 fungsi FUNCTION VOID 0 1 0 0 0 14
16 konstanta CONSTANT VOID 0 1 0 0 0 15
17 tipe TYPE VOID 0 1 0 0 0 16
18 kasus TYPE VOID 0 1 0 0 0 17
19 rekaman TYPE RECORD 0 1 0 0 0 18
20 ulangi TYPE VOID 0 1 0 0 0 19
21 sampai TYPE VOID 0 1 0 0 0 20
22 integer TYPE INTEGER 0 1 0 0 0 21
23 real TYPE REAL 0 1 0 0 0 22
24 boolean TYPE BOOLEAN 0 1 0 0 0 23
25 char TYPE CHAR 0 1 0 0 0 24
26 string TYPE STRING 0 1 0 0 0 25
27 writeln PROCEDURE VOID 0 1 0 0 0 26
28 readIn PROCEDURE VOID 0 1 0 0 0 27
29 write PROCEDURE VOID 0 1 0 0 0 28
30 read PROCEDURE VOID 0 1 0 0 0 29
32 Input12 PROGRAM VOID 0 1 0 0 0 0
33 x VARIABLE INTEGER 0 1 0 0 0 32

Block Table (btab):
Idx Last Lpar Psze Vsze
----- 
0 33 0 0 1
1 0 0 0 0 0

Array Table (atab):
(empty)

✓ No semantic errors found
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224>

```

3.4 Kasus Uji 4

Input 4:

```

program ParamMismatch;

fungsi Add(a, b: integer): integer;
mulai
  Add := a + b;
selesai;

prosedur PrintNumber(num: integer);
mulai
  writeln('Number: ', num);
selesai;

variabel
  x: integer;
  y: real;
  flag: boolean;
mulai
  x := Add(5, 10);           { Valid }
  y := Add(3, 4);            { Valid (integer ke real OK) }

{ SEMANTIC ERROR: Parameter type mismatch }

```

```

x := Add(5, 3.14);           { Error: real tidak bisa ke integer
parameter }
PrintNumber(y);             { Error: real tidak bisa ke integer
parameter }
PrintNumber(flag);          { Error: boolean tidak bisa ke integer
parameter }

{ SEMANTIC ERROR: Wrong number of parameters }
x := Add(5);                 { Error: kurang parameter }
x := Add(1, 2, 3);           { Error: kelebihan parameter }

selesai.

```

Output 4:

```

==== SEMANTIC ANALYSIS ====

==== DECORATED AST ====
ProgramNode(name: 'ParamMismatch')
└ Declarations
    └ FunctionDecl → tab_index:33, return_type:integer, name:Add
    └ ProcedureDecl → tab_index:36, name:PrintNumber
    └ VarDecl('x') → tab_index:38, type:integer, lev:0
    └ VarDecl('y') → tab_index:39, type:real, lev:0
    └ VarDecl('flag') → tab_index:40, type:boolean, lev:0
    └ Block → block_index:3, lev:1
        └ Assign('x' := FunctionCall(type=BaseType.INTEGER,
tab_idx=33)) → type:void
            └ target 'x' → tab_index:38, type:integer
                └ value FunctionCall(type=BaseType.INTEGER, tab_idx=33) →
type:integer
            └ Assign('y' := FunctionCall(type=BaseType.INTEGER,
tab_idx=33)) → type:void
                └ target 'y' → tab_index:39, type:real
                    └ value FunctionCall(type=BaseType.INTEGER, tab_idx=33) →
type:integer
            └ Assign('x' := FunctionCall(type=BaseType.INTEGER,
tab_idx=33)) → type:void
                └ target 'x' → tab_index:38, type:integer
                    └ value FunctionCall(type=BaseType.INTEGER, tab_idx=33) →
type:integer
            └ PrintNumber(...) → tab_index:36
            └ PrintNumber(...) → tab_index:36
            └ Assign('x' := FunctionCall(type=BaseType.INTEGER,
tab_idx=33)) → type:void
                └ target 'x' → tab_index:38, type:integer
                    └ value FunctionCall(type=BaseType.INTEGER, tab_idx=33) →
type:integer
            └ Assign('x' := FunctionCall(type=BaseType.INTEGER,
tab_idx=33)) → type:void
                └ target 'x' → tab_index:38, type:integer
                    └ value FunctionCall(type=BaseType.INTEGER, tab_idx=33) →
type:integer
    └ PrintNumber(...)

==== SYMBOL TABLES ====

Identifier Table (tab):
Idx  Name      Obj      Type     Ref   Nrm   Lev   Adr   Link
-----
0   program   PROGRAM   VOID     0     1     0     0     0
1   variabel  TYPE     VOID     0     1     0     0     0
2   mulai     TYPE     VOID     0     1     0     0     1
3   selesai   TYPE     VOID     0     1     0     0     2
4   jika      TYPE     VOID     0     1     0     0     3
5   maka      TYPE     VOID     0     1     0     0     4
6   selainitu TYPE     VOID     0     1     0     0     5

```

| | | | | | | | | |
|----|---------------|-----------|---------|---|---|---|---|----|
| 7 | selama | TYPE | VOID | 0 | 1 | 0 | 0 | 6 |
| 8 | lakukan | TYPE | VOID | 0 | 1 | 0 | 0 | 7 |
| 9 | untuk | TYPE | VOID | 0 | 1 | 0 | 0 | 8 |
| 10 | ke | TYPE | VOID | 0 | 1 | 0 | 0 | 9 |
| 11 | turunke | TYPE | VOID | 0 | 1 | 0 | 0 | 10 |
| 12 | larik | TYPE | ARRAY | 0 | 1 | 0 | 0 | 11 |
| 13 | dari | TYPE | VOID | 0 | 1 | 0 | 0 | 12 |
| 14 | prosedur | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 13 |
| 15 | fungsi | FUNCTION | VOID | 0 | 1 | 0 | 0 | 14 |
| 16 | konstanta | CONSTANT | VOID | 0 | 1 | 0 | 0 | 15 |
| 17 | tipe | TYPE | VOID | 0 | 1 | 0 | 0 | 16 |
| 18 | kasus | TYPE | VOID | 0 | 1 | 0 | 0 | 17 |
| 19 | rekaman | TYPE | RECORD | 0 | 1 | 0 | 0 | 18 |
| 20 | ulangi | TYPE | VOID | 0 | 1 | 0 | 0 | 19 |
| 21 | sampai | TYPE | VOID | 0 | 1 | 0 | 0 | 20 |
| 22 | integer | TYPE | INTEGER | 0 | 1 | 0 | 0 | 21 |
| 23 | real | TYPE | REAL | 0 | 1 | 0 | 0 | 22 |
| 24 | boolean | TYPE | BOOLEAN | 0 | 1 | 0 | 0 | 23 |
| 25 | char | TYPE | CHAR | 0 | 1 | 0 | 0 | 24 |
| 26 | string | TYPE | STRING | 0 | 1 | 0 | 0 | 25 |
| 27 | writeln | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 26 |
| 28 | readln | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 27 |
| 29 | write | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 28 |
| 30 | read | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 29 |
| 32 | ParamMismatch | PROGRAM | VOID | 0 | 1 | 0 | 0 | 0 |
| 33 | Add | FUNCTION | INTEGER | 0 | 1 | 0 | 0 | 32 |
| 34 | a | VARIABLE | INTEGER | 0 | 1 | 1 | 0 | 0 |
| 35 | b | VARIABLE | INTEGER | 0 | 1 | 1 | 1 | 34 |
| 36 | PrintNumber | PROCEDURE | VOID | 0 | 1 | 0 | 0 | 33 |
| 37 | num | VARIABLE | INTEGER | 0 | 1 | 1 | 2 | 0 |
| 38 | x | VARIABLE | INTEGER | 0 | 1 | 0 | 3 | 36 |
| 39 | y | VARIABLE | REAL | 0 | 1 | 0 | 4 | 38 |
| 40 | flag | VARIABLE | BOOLEAN | 0 | 1 | 0 | 5 | 39 |

Block Table (btab):

| Idx | Last | Lpar | Psze | Vsze |
|-----|------|------|------|------|
| 0 | 40 | 0 | 0 | 3 |
| 1 | 35 | 0 | 0 | 2 |
| 2 | 37 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 |

Array Table (atab):

(empty)

✗ Found 5 semantic errors:

- Semantic Error at line 22, column 8: Parameter type mismatch in Add: parameter 2 (b) expects integer, got real
- Semantic Error at line 23, column 3: Parameter type mismatch in PrintNumber: parameter 1 (num) expects integer, got real
- Semantic Error at line 24, column 3: Parameter type mismatch in PrintNumber: parameter 1 (num) expects integer, got boolean
- Semantic Error at line 27, column 8: Parameter count mismatch in Add: expected 2, got 1
- Semantic Error at line 28, column 8: Parameter count mismatch in Add: expected 2, got 3

Screenshot 4:

```

PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/error-input-5.pas
*** SEMANTIC ANALYSIS ***

*** DECORATED AST ***
ProgramNode(name: 'ParamMismatch')
  Declarations
    FunctionDecl > tab_index:33, return_type:integer, name:Add
    ProcedureDecl > tab_index:36, name:PrintNumber
    VarDecl('x') > tab_index:38, type:integer, lev:0
    VarDecl('y') > tab_index:39, type:real, lev:0
    VarDecl('flag') > tab_index:40, type:boolean, lev:0
  Block > block_index:3, lev:1
    Assign('x' := FunctionCall(type=BaseType.INTEGER, tab_idx=33)) > type:void
      target 'x' > tab_index:38, type:integer
        value FunctionCall(type=BaseType.INTEGER, tab_idx=33) > type:integer
    Assign('y' := FunctionCall(type=BaseType.INTEGER, tab_idx=33)) > type:void
      target 'y' > tab_index:39, type:real
        value FunctionCall(type=BaseType.INTEGER, tab_idx=33) > type:integer
    Assign('x' := FunctionCall(type=BaseType.INTEGER, tab_idx=33)) > type:void
      target 'x' > tab_index:38, type:integer
        value FunctionCall(type=BaseType.INTEGER, tab_idx=33) > type:integer
    PrintNumber(...) > tab_index:36
    PrintNumber(...) > tab_index:36
    Assign('x' := FunctionCall(type=BaseType.INTEGER, tab_idx=33)) > type:void
      target 'x' > tab_index:38, type:integer
        value FunctionCall(type=BaseType.INTEGER, tab_idx=33) > type:integer
    Assign('x' := FunctionCall(type=BaseType.INTEGER, tab_idx=33)) > type:void
      target 'x' > tab_index:38, type:integer
        value FunctionCall(type=BaseType.INTEGER, tab_idx=33) > type:integer

*** SYMBOL TABLES ***
Identifier Table (tab):
Idx Name Obj Type Ref Nrm Lev Adr Link
-----
0 program PROGRAM VOID 0 1 0 0 0
1 variabel TYPE VOID 0 1 0 0 0
2 mulai TYPE VOID 0 1 0 0 1
3 selesai TYPE VOID 0 1 0 0 2
4 jika TYPE VOID 0 1 0 0 3
5 maka TYPE VOID 0 1 0 0 4
6 selainitu TYPE VOID 0 1 0 0 5
7 selama TYPE VOID 0 1 0 0 6
8 lakukan TYPE VOID 0 1 0 0 7
9 untuk TYPE VOID 0 1 0 0 8
10 ke TYPE VOID 0 1 0 0 9
11 turunke TYPE VOID 0 1 0 0 10
12 larik TYPE ARRAY 0 1 0 0 11
13 dari TYPE VOID 0 1 0 0 12
14 prosedur PROCEDURE VOID 0 1 0 0 13

```

```

PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/error-input-5.pas
14 prosedur PROCEDURE VOID 0 1 0 0 13
15 fungsi FUNCTION VOID 0 1 0 0 14
16 konstanta CONSTANT VOID 0 1 0 0 15
17 tipe TYPE VOID 0 1 0 0 16
18 kasus TYPE VOID 0 1 0 0 17
19 rekaman TYPE RECORD 0 1 0 0 18
20 ulangi TYPE VOID 0 1 0 0 19
21 sampai TYPE VOID 0 1 0 0 20
22 integer TYPE INTEGER 0 1 0 0 21
23 real TYPE REAL 0 1 0 0 22
24 boolean TYPE BOOLEAN 0 1 0 0 23
25 char TYPE CHAR 0 1 0 0 24
26 string TYPE STRING 0 1 0 0 25
27 writeln PROCEDURE VOID 0 1 0 0 26
28 readln PROCEDURE VOID 0 1 0 0 27
29 write PROCEDURE VOID 0 1 0 0 28
30 read PROCEDURE VOID 0 1 0 0 29
32 ParamMismatch PROGRAM VOID 0 1 0 0 0
33 Add FUNCTION INTEGER 0 1 0 0 32
34 a VARIABLE INTEGER 0 1 1 0 0
35 b VARIABLE INTEGER 0 1 1 1 34
36 PrintNumber PROCEDURE VOID 0 1 0 0 33
37 num VARIABLE INTEGER 0 1 1 2 0
38 x VARIABLE INTEGER 0 1 0 3 36
39 y VARIABLE REAL 0 1 0 4 38
40 flag VARIABLE BOOLEAN 0 1 0 5 39

Block Table (btab):
Idx Last Lpar Psze Vsze
-----
0 40 0 0 3
1 35 0 0 2
2 37 0 0 1
3 0 0 0 0

Array Table (atab):
(empty)

X Found 5 semantic errors:
- Semantic Error at line 22, column 8: Parameter type mismatch in Add: parameter 2 (b) expects integer, got real
- Semantic Error at line 23, column 3: Parameter type mismatch in PrintNumber: parameter 1 (num) expects integer, got real
- Semantic Error at line 24, column 3: Parameter type mismatch in PrintNumber: parameter 1 (num) expects integer, got boolean
- Semantic Error at line 27, column 8: Parameter count mismatch in Add: expected 2, got 1
- Semantic Error at line 28, column 8: Parameter count mismatch in Add: expected 2, got 3
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> []

```

3.5 Kasus Uji 5

Input 5:

```

program Uninitialized;
konstanta

```

```

MAX = 100;
MIN = 1;
tipe
Range = 1..10;
ArrayInt = larik[1..10] dari integer;
variabel
a, b, c: integer;
arr: ArrayInt;
mulai
b := MIN;
c := a + b;
selesai.

```

Output 5:

```

==== SEMANTIC ANALYSIS ====

==== DECORATED AST ====
ProgramNode(name: 'Uninitialized')
  └─ Declarations
    └─ ConstDecl → tab_index:33, type:integer, lev:0
    └─ ConstDecl → tab_index:34, type:integer, lev:0
    └─ TypeDecl → tab_index:35, type:integer, lev:0
    └─ TypeDecl → tab_index:36, type:array, lev:0
    └─ VarDecl('a') → tab_index:37, type:integer, lev:0
    └─ VarDecl('b') → tab_index:38, type:integer, lev:0
    └─ VarDecl('c') → tab_index:39, type:integer, lev:0
    └─ VarDecl('arr') → tab_index:40, type:array, lev:0
  └─ Block → block_index:1, lev:1
    └─ Assign('b' := ConstIdentifier(type=BaseType.INTEGER,
tab_idx=34)) → type:void
      └─ target 'b' → tab_index:38, type:integer
        └─ value ConstIdentifier(type=BaseType.INTEGER, tab_idx=34)
  → type:integer
    └─ Assign('c' := 'a'+Var('b')) → type:void
      └─ target 'c' → tab_index:39, type:integer
        └─ BinOp '+' → type:integer
          └─ 'a' → tab_index:37, type:integer
          └─ 'b' → tab_index:38, type:integer

==== SYMBOL TABLES ====

Identifier Table (tab):
Idx  Name       Obj      Type     Ref   Nrm   Lev   Adr   Link
-----
0   program    PROGRAM  VOID     0     1     0     0     0
1   variabel   TYPE    VOID     0     1     0     0     0
2   mulai      TYPE    VOID     0     1     0     0     1
3   selesai    TYPE    VOID     0     1     0     0     2
4   jika       TYPE    VOID     0     1     0     0     3
5   maka       TYPE    VOID     0     1     0     0     4
6   selainitu  TYPE    VOID     0     1     0     0     5
7   selama     TYPE    VOID     0     1     0     0     6
8   lakukan   TYPE    VOID     0     1     0     0     7
9   untuk      TYPE    VOID     0     1     0     0     8
10  ke         TYPE    VOID     0     1     0     0     9
11  turunke   TYPE    VOID     0     1     0     0     10
12  larik      TYPE   ARRAY   0     1     0     0     11
13  dari       TYPE    VOID     0     1     0     0     12
14  prosedur   PROCEDURE VOID     0     1     0     0     13
15  fungsi     FUNCTION VOID     0     1     0     0     14
16  konstanta  CONSTANT VOID     0     1     0     0     15
17  tipe       TYPE    VOID     0     1     0     0     16
18  kasus      TYPE    VOID     0     1     0     0     17
19  rekaman    TYPE   RECORD  0     1     0     0     18

```

```

20 ulangi      TYPE      VOID      0      1      0      0      19
21 sampai      TYPE      VOID      0      1      0      0      20
22 integer     TYPE      INTEGER   0      1      0      0      21
23 real         TYPE      REAL      0      1      0      0      22
24 boolean     TYPE      BOOLEAN   0      1      0      0      23
25 char         TYPE      CHAR      0      1      0      0      24
26 string       TYPE      STRING    0      1      0      0      25
27 writeln     PROCEDURE VOID      0      1      0      0      26
28 readln      PROCEDURE VOID      0      1      0      0      27
29 write        PROCEDURE VOID      0      1      0      0      28
30 read         PROCEDURE VOID      0      1      0      0      29
32 Uninitialized PROGRAM  VOID      0      1      0      0      0
33 MAX          CONSTANT  INTEGER   0      1      0      0      32
34 MIN          CONSTANT  INTEGER   0      1      0      0      33
35 Range        TYPE      INTEGER   0      1      0      0      34
36 ArrayInt    TYPE      ARRAY     0      1      0      0      35
37 a            VARIABLE INTEGER  0      1      0      0      36
38 b            VARIABLE INTEGER  0      1      0      1      37
39 c            VARIABLE INTEGER  0      1      0      2      38
40 arr          VARIABLE ARRAY    0      1      0      3      39

Block Table (btab):
Idx  Last   Lpar   Psze   Vsze
-----
0    40     0      0      4
1    0      0      0      0

Array Table (atab):
Idx  IdxType  ElemType Eref   Low    High   Elemsize  Size
-----
0    INTEGER   INTEGER   0      1      10      1      10

✗ Found 1 semantic errors:
- Semantic Error at line 13, column 8: Variable 'a' might be used
before initialization

```

Screenshot 5:

```

PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/error-input-10.pas
*** SEMANTIC ANALYSIS ***

*** DECORATED AST ***
ProgramNode(name: 'Uninitialized')
  Declarations
    ConstDecl → tab_index:33, type:integer, lev:0
    ConstDecl → tab_index:34, type:integer, lev:0
    TypeDecl → tab_index:35, type:integer, lev:0
    TypeDecl → tab_index:36, type:array, lev:0
    VarDecl('a') → tab_index:37, type:integer, lev:0
    VarDecl('b') → tab_index:38, type:integer, lev:0
    VarDecl('c') → tab_index:39, type:integer, lev:0
    VarDecl('arr') → tab_index:40, type:array, lev:0
  Block → block_index:1, lev:1
    Assign('b' := ConstIdentifier(type=BaseType.INTEGER, tab_idx=34)) → type:void
      target 'b' → tab_index:38, type:integer
      value ConstIdentifier(type=BaseType.INTEGER, tab_idx=34) → type:integer
    Assign('c' := 'a'+Var('b')) → type:void
      target 'c' → tab_index:39, type:integer
      BinOp '+' → type:integer
        'a' → tab_index:37, type:integer
        'b' → tab_index:38, type:integer

*** SYMBOL TABLES ***
Identifier Table (tab):
Idx Name Obj Type Ref Nrm Lev Adr Link
0 program PROGRAM VOID 0 1 0 0 0
1 variabel TYPE VOID 0 1 0 0 0
2 mulai TYPE VOID 0 1 0 0 1
3 selesai TYPE VOID 0 1 0 0 2
4 jika TYPE VOID 0 1 0 0 3
5 maka TYPE VOID 0 1 0 0 4
6 selainitu TYPE VOID 0 1 0 0 5
7 selama TYPE VOID 0 1 0 0 6
8 lakukan TYPE VOID 0 1 0 0 7
9 untuk TYPE VOID 0 1 0 0 8
10 ke TYPE VOID 0 1 0 0 9
11 turunke TYPE VOID 0 1 0 0 10
12 larik TYPE ARRAY 0 1 0 0 11
13 dari TYPE VOID 0 1 0 0 12
14 prosedur PROCEDURE VOID 0 1 0 0 13
15 fungsi FUNCTION VOID 0 1 0 0 14
16 konstanta CONSTANT VOID 0 1 0 0 15
17 tipe TYPE VOID 0 1 0 0 16
18 kasus TYPE VOID 0 1 0 0 17
19 rekaman TYPE RECORD 0 1 0 0 18
20 ulangi TYPE VOID 0 1 0 0 19

```

```

PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224> python -m src.compiler test/milestone-3/error-input-10.pas
20 ulangi TYPE VOID 0 1 0 0 19
21 sampai TYPE VOID 0 1 0 0 20
22 integer TYPE INTEGER 0 1 0 0 21
23 real TYPE REAL 0 1 0 0 22
24 boolean TYPE BOOLEAN 0 1 0 0 23
25 char TYPE CHAR 0 1 0 0 24
26 string TYPE STRING 0 1 0 0 25
27 writeln PROCEDURE VOID 0 1 0 0 26
28 readin PROCEDURE VOID 0 1 0 0 27
29 write PROCEDURE VOID 0 1 0 0 28
30 read PROCEDURE VOID 0 1 0 0 29
32 Uninitialized PROGRAM VOID 0 1 0 0 0 0
33 MAX CONSTANT INTEGER 0 1 0 0 32
34 MIN CONSTANT INTEGER 0 1 0 0 33
35 Range TYPE INTEGER 0 1 0 0 34
36 ArrayInt TYPE ARRAY 0 1 0 0 35
37 a VARIABLE INTEGER 0 1 0 0 36
38 b VARIABLE INTEGER 0 1 0 1 37
39 c VARIABLE INTEGER 0 1 0 2 38
40 arr VARIABLE ARRAY 0 1 0 3 39

Block Table (btab):
Idx Last Lpar Psze Vsze
0 40 0 0 4
1 0 0 0 0 0

Array Table (atab):
Idx IdxType Elemtpe Eref Low High Elemsize Size
0 INTEGER INTEGER 0 1 10 1 10

X Found 1 semantic errors:
- Semantic Error at line 13, column 8: Variable 'a' might be used before initialization
PS C:\Users\Acer\Downloads\FRT-Tubes-IF2224>

```

BAB 4

KESIMPULAN DAN SARAN

4.1 Kesimpulan

- Semantic Analysis merupakan tahap kritis dalam proses kompilasi yang memvalidasi kebenaran makna program setelah melalui tahap lexical dan syntax analysis. Tahap ini memastikan program tidak hanya benar secara sintaks tetapi juga secara semantik.
- Pendekatan L-Attributed Grammar dengan visitor pattern terbukti efektif dalam membangun Abstract Syntax Tree (AST) dari parse tree. Setiap visit function berhasil mengeksekusi semantic actions yang sesuai dan melakukan validasi aturan semantik.
- Implementasi symbol table dengan tiga tabel terpisah (tab, btab, atab) sesuai spesifikasi Pascal-S telah berhasil mengelola informasi identifier dan scope dengan baik.
- Sistem type checking yang diimplementasikan mampu mendeteksi berbagai jenis ketidaksesuaian tipe data, termasuk type mismatch dalam assignment operations, parameter passing, dan binary expressions.

4.2 Saran

- Penambahan fitur semantic warnings untuk unused variables akan membantu programmer dalam menulis kode yang lebih bersih dan menghindari bug potensial.
- Testing yang comprehensive sangat diperlukan. Perlu ditambah test case untuk berbagai skenario edge case yang mungkin dibuat programmer.
- Dokumentasi kode yang sudah cukup baik perlu dipertahankan dan dilengkapi agar memudahkan pengembangan berkelanjutan dan kolaborasi tim.

LAMPIRAN

Link Release Repository Github:

<https://github.com/JethroJNS/FRT-Tubes-IF2224.git>

Pembagian Tugas:

| NIM | Nama | Persentase Kontribusi |
|----------|--------------------------------|-----------------------|
| 13521117 | Maggie Zeta Rosida S | 25% |
| 13523037 | Buege Mahara Putra | 25% |
| 13523041 | Hanif Kalyana Aditya | 25% |
| 13523081 | Jethro Jens Norbert Simatupang | 25% |

REFERENSI

“PASCAL-S: A Subset and its implementation”

<http://pascal.hansotten.com/uploads/pascals/PASCAL-S%20A%20subset%20and%20its%20Implementation%20012.pdf>

“Let’s Build A Simple Interpreter. Part 7: Abstract Syntax Trees”

<https://ruslanspivak.com/lbasi-part7>

“Contoh Gambar Parse Trees”

https://textx.github.io/Arpeggio/latest/images/calc_parse_tree.dot.png

“Semantic Analysis (ULiège)”

<https://people.montefiore.ulg.be/geurts/Cours/compil/2015/04-semantic-2015-2016.pdf>

“Pascal for small machines”

<http://pascal.hansotten.com/niklaus-wirth/pascal-s/>

“Syntax Directed Translation in Compiler Design”

<https://www.geeksforgeeks.org/compiler-design/syntax-directed-translation-in-compiler-design/>